Facilitating Wireless Communications through Intelligent Resource Management on Software-Defined Radios in Dynamic Spectrum Environments

Joseph D. Gaeddert

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

January 18, 2011
Blacksburg, Virginia

Keywords: software-defined radio, cognitive radio, OFDM/OQAM, synchronization, cognitive engine

Facilitating Wireless Communications through Intelligent Resource Management on Software-Defined Radios in Dynamic Spectrum Environments

Joseph D. Gaeddert

ABSTRACT

This dissertation provides theory and analysis on the impact resource management has on software-defined radio platforms by investigating the inherent trade-off between spectrum and processing efficiencies with their relation to both the power consumed by the host processor and the complexity of the algorithm which it can support. The analysis demonstrates that considerable resource savings can be gained without compromising the resulting quality of service to the user, concentrating specifically on physical-layer signal processing elements commonly found in software definitions of single- and multi-carrier communications signals.

Novel synchronization techniques and estimators for unknown physical layer reference parameters are introduced which complement the energy-quality scalability of software-defined receivers. A new framing structure is proposed for single-carrier systems which enables fast synchronization of short packet bursts, applicable for use in dynamic spectrum access. The frame is embedded with information describing its own structure, permitting the receiver to automatically modify its software configuration, promoting full waveform flexibility for adapting to quickly changing wireless channels. The synchronizer's acquisition time is reduced by exploiting cyclostationary properties in the preamble of transmitted framing structure, and the results are validated over the air in a wireless multi-path laboratory environment. Multi-carrier analysis is concentrated on synchronizing orthogonal frequency-division multiplexing (OFDM) using offset quadrature amplitude modulation (OFDM/OQAM) which is shown to have significant spectral compactness advantages over traditional OFDM. Demodulation of OFDM/OQAM is accomplished using computationally efficient polyphase analysis filterbanks, enabled by a novel approximate square-root Nyquist filter design based on the near-optimum Kaiser-Bessel window. Furthermore, recovery of sample timing and carrier frequency offsets are shown to be possible entirely in the frequency domain, enabling demodulation in the presence of strong interference signals while promoting heterogeneous signal coexistence in dynamic spectrum environments.

Resource management is accomplished through the introduction of a self-monitoring framework which permits system-level feedback to the radio at run time. The architecture permits the radio to monitor its own processor usage, demonstrating considerable savings in computation bandwidths on the tested platform. Resource management is assisted by supervised intelligent heuristic-based learning algorithms which use software-level feedback of the radio's active resource consumption to optimize energy and processing efficiencies in dynamic spectrum environments. In particular, a case database-enabled cognitive engine is proposed which abstracts from the radio application by using specific knowledge of previous experience rather than relying on general knowledge within a specific problem domain.

Acknowledgments

The work presented in this dissertation would not have been possible without the endless support of my caring family, friends, and the precious faculty and students at Virginia Tech. The kindness and encouragement they have shared over the years constitutes a great portion of my continuing motivation for progress. In particular I want to thank my advisor, Jeffrey Reed, for inspiring my work and encouraging the pursuit of this research, Christopher Phelps for challenging the very thought of "fair and balanced," Christopher Headley for showing me that knife-wrench isn't really just for kids, Carlos Aguayo for constantly reminding me "You broke it," my sister Susan, the first "Dr. Gaeddert," for expertly knitted socks and hats, and Maria Licher, my loving wife of just four weeks, for fortifying my resolve.

This dissertation is dedicated to my mother for showing me the meaning of "screw your courage to the sticking place" and to my father for teaching me the significance of perseverance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The wireless industry has an ever-growing appetite for broadband internet services with increased global coverage, and thus technological advances in wireless communications has expanded at an astounding pace. With this growing demand for wireless connectivity comes the need to adapt to quickly-changing standards. Software-defined radio (SDR) is heralded by many in the wireless engineering community as the future of standards implementation; however this mentality might be somewhat short-sighted. Reconfigurability, however advantageous in theory, comes at the expense of increased power consumption and hardware costs; despite all the proposed benefits, SDR platforms consume considerably more power than their hardware counterparts. The energy efficiency lost through the flexibility of baseband processing severely limits the computational capacity of the target reconfigurable platform and has considerable implications on its physical size, weight, and battery life. Furthermore, the steady increase of hardware capabilities (e.g. Moore's law) is easily overwhelmed by the escalating demand for high-quality bandwidth on such mobile wireless devices. As such, the performance of software-defined radio platforms will continue to lag behind conventional hardware designs.

1.1 Problem Description

While there does exist a significant performance gap between hardware and software radio platforms, the flexibility within the SDR platform can be leveraged to mitigate its additional power demands. The immense variability in re-programmable software components permits the system to be dynamically rebalanced for new scenarios, allowing for more efficient use of hardware. Today, wireless communications engineers have the option to trade physical performance (such as transmit power and signal bandwidth) for signal processing complexity [69, p. 6]. This has permitted the application of sophisticated algorithms which have pushed link capacities towards their information theoretic bounds and has consequently shaped the entire area of wireless communications.

Many SDR implementations seek to optimize DSP algorithms for a range of operating conditions; however because the performance of mobile radios depends so significantly upon channel conditions, it is not possible to guarantee minimal power consumption for a specific platform unless the dynamics of the algorithm itself are designed to be energy scalable. Therefore the algorithm which minimizes the energy consumed for a particular application while achieving a certain quality of service cannot be determined without first considering the complex dynamics of the wireless environment in addition to the platform upon which the application is running. Resource management on reconfigurable platforms is therefore a critical step in deploying mobile reconfigurable radios, particularly those operating in dynamic spectrum environments. Many of the traditional methods for reducing energy consumption on mobile radios, however, are inapplicable to reconfigurable platforms because hardware and software are no longer strictly bound to one another.

One of the most apparent applications to this concept resides in cognitive radios, where resource management can additionally incorporate processing complexity within protocol negotiation. By employing a self-adapting engine, cognitive radios (usually considered to be a superset of SDR) use goal-driven autonomous operation to improve link performance transparent to the user. Despite their many promises, however, cognitive radios (and otherwise adaptive radios) still require measurable system-level performance feedback in order to appropriately switch operational modes. It is therefore necessary to allow the radio to monitor its own dynamic resource consumption in addition to its link-level performance as to compete with fixed-mode hardware platforms. Traditional radio resources take the form of energy, power, and spectrum; once the software platform has been introduced, this list extends to computational complexity, memory, and processor usage. The challenge therefore becomes an issue of how to efficiently model and measure the platform's consumption of finite resources—a particularly difficult task as software becomes portable to a growing population of heterogeneous platforms. Highly parallelized architectures (such as FPGAs and multi-core processors) exacerbate the difficulty in estimating the processing time and complexity of executing particular algorithms as their platform-specific implementations become non-linear.

With the current onset of cognitive radio technologies seeking access to dynamic spectrum environments comes the need for software radio platforms to modify their protocols to adapting conditions. As discussion of spectrum sensing modules becomes more prevalent in current protocol descriptions, the need for an extensible physical layer description becomes requisite. Furthermore, dynamic frequency selection in congested spectrum environments necessitates that the transmitter be disabled during the spectrum-sensing period. This requirement completely changes the transceiver design and operation over traditional systems. As such, these qualifications imply two methods for dynamic spectrum access: the first employs a single carrier which must transmit data in bursts, requiring the receiver to synchronize to each frame very quickly; the second issues a multi-carrier physical layer description capable of indexing data subcarriers on frequencies that are deemed to be unoccupied, and disabling subcarriers which overlap occupied channels. Both cases give strong implications on the resources consumed by the baseband processor as each mandates that the receiver acquire the data signal very quickly in hostile channel conditions, potentially in the presence of interference. Consequently, the physical layer is the limiting factor in realizing such flexible receivers on a software platform, and is the driving motivation behind the research presented in this dissertation.

1.2 The Role of Software-Defined Radios

The role of SDR in the current wireless market has traditionally fallen into two major categories: seamless coverage of wireless connectivity and integration with existing standards. The flexibility gained through reconfigurable and reusable software, however, should not limit its application to merely re-implementing existing standards. At present, the cost of manufacturing a single reconfigurable platform far exceeds that of multiple application-specific integrated circuits, and consequently the wireless market has seen limited room for deploying a software-defined radio in a mobile handset. Software-defined radios do, however, have the ability to future-proof standards and consequently their value to wireless infrastructure is steadily growing. Traditionally, wireless standards undergo a number of revisions in which standards bodies continually revise protocols and weigh the benefits of certain technologies over others. This revision process takes a considerable amount of time during which equipment manufacturers will release preliminary versions of the end product. With the flexibility of software, these protocols can evolve more organically in a similar fashion to open-source software. This allows for standards bodies to roll out revisions much more quickly and reliably than is possible on hardware platforms.

While integration with existing standards has its place, the real benefits of SDR are realized through its ability to to be completely flexible in baseband processing. This permits the radio to re-define its own physical layer description, permitting it to leverage channel information to improve the user experience. In a heterogeneous network (one which is composed of may reconfigurable radios with different capabilities) interoperability is contingent upon such devices finding a common ground. In this regard, wireless nodes must have some set of overlapping capabilities in order to inter-communicate. As such, a hardware platform can only support protocols for which it has specific integrated circuitry, yielding a potentially inferior service quality. This limits the functionality of communication between nodes, and can severely reduce the link's performance, particularly if these protocols are ill-suited to the required application or insufficient to cope with the wireless environment. For example, WiFi is ubiquitously offered on mobile handsets; however its applicability towards applications with low data rate requirements and/or long-range communications is known to be inferior. Furthermore, its physical layer and medium access control specifications are ill-suited towards environments with high mobility (large Doppler shifts) and highly congested spectrum.

Software platforms, however, can negotiate protocols tailored specifically to a particular problem and realize the solution through software reconfiguration. Consider an open protocol system in which existing signal processing software can be reconfigured to implement a variety of systems, including packet-based links, multimedia broadcast streaming, massive sensor node networks, or even dynamic spectrum access networks. The functionality of the link is now limited only by the capabilities of the RF front end (tunable frequencies, transmit power, etc.) and the computational bandwidth of the baseband processor. The focus of efficiency shifts towards reducing the platform's resource consumption so as to compete with hardware radios. As such, efficient implementations of baseband synchronizes is paramount to the success of such heterogeneous networks. Much of this functionality crosses over with cognitive radio which is why they are so closely related.

1.3 The Role of Cognitive Radios

Reconfigurability becomes particularly useful as the growing demand for high data-rate wireless communications exceeds the pace of well-defined protocol standards. Consequently, the available spectrum is becoming more and more a valuable and scarce commodity, particularly as bands are subdivided into licensed partitions and therefore unavailable to secondary users. One potential solution is to allow unlicensed radios to autonomously detect and use empty spectrum bands. Such dynamic spectrum access has been proposed by a number of researchers and adopted in part by industry [70]. Cognitive radios (CR) are predominantly considered to be an extension of SDR and have been proposed to enable such technologies through dynamic spectrum access for efficient white-space re-use. Furthermore, accommodating the ever-increasing demands of high data-rate communications in dynamic spectrum environments requires fast synchronization of spectrally compact multi-carrier communications signals such as orthogonal frequency-division multiplexing using offset quadrature amplitude modulation (OFDM/OQAM). This signal acquisition must be robust to the presence of narrowband interference and motivates the investigation of parametric estimation of reference parameters strictly in the frequency domain. Technologies in the area of cognitive radios additionally promise intelligent parametric optimization of data links to improve quality of service (QoS) over traditional radios, particularly in actively dense spectrum environments through intelligent situation-aware reconfiguration. The radio itself, through the immense flexibility of software reconfiguration, can determine the best course of action as the situation unfolds. Without knowledge of the underlying radio structure, its operator can describe the problem and permit the radio to provide an environmentally-aware solution completely online. This level of autonomy is enabled by a concept known as *radio learning* in which the radio is contextually aware of its operating environment.

A plethora of definitions exist for cognitive radio—one for each application—usually targeting dynamic spectrum allocation and improved spectral efficiency. While compact frequency use is an important component in radio networks, spectrum management is too specific a problem to encompass the purpose of radio cognition. These goals are actually byproducts of the underlying motivation behind the introduction of the cognitive radios. In truth, the sole purpose for introducing artificial intelligence and cognition in radios is to facilitate communication. This simple concept is often difficult to actualize, particularly on low-profile embedded systems where power consumption is a critical factor. This motivation resides in the fact that observations about the surroundings are incomplete and mathematical models used to describe the environment are imprecise. Truthfully, radios operating on wireless protocols with strict spectrum regulation have no need for intelligent algorithms beyond simple adaptation. In this regard cognition serves no purpose in radios as the information gathered about its environment is dictated by the boundaries of the protocol in which it operates. For example, cellular networks use adaptive power control algorithms to ensure that the base station receives equal power from each node in the cell, improving network throughput. The power adjustment in the feedback loop is often driven by the packet error-rate of each link. Because cellular networks have enforced restrictions on those who can operate within their frequency bands, the assumption of the algorithm is that error rates are due to noise and in-network interference alone, and not a result of any unlicensed users. Similarly, IEEE 802.11 protocols use carrier sense multiple access to help with collision avoidance. This is in part necessary when operating in the industrial, scientific, and medical (ISM) band, in which a significant amount of electromagnetic interference is emitted from devices such as microwave ovens for purposes other than communications. The standard itself has a well-defined protocol for handling scenarios in which interference can cause significant errors in the link. In this regard, system design practice traditionally has been to contend with interference through an explicit protocol description.

To this end it must first be acknowledged that fixed-mode radios cannot operate outside the bounds of their design. It is certain that wireless standards are constantly evolving to contend with a broadening set of usage models, covering everything from high mobility to multi-user access. In dynamic spectrum environments, however, unlicensed users contend for spectrum without the aid of specific protocols. The challenge here is to design a radio that is capable of contending with scenarios for which it has neither prepared nor expected. This is particularly relevant when the radio is not strictly guided by protocol, or when theoretical models upon which its design is based are either inaccurate or are insufficient to dictate its own performance. With the introduction of radio learning into its physical layer description, the system can potentially contend with unknown situations. As a consequence, cognitive radios are capable not only of choosing a particular spectral allocation, but additionally managing the underlying protocol which suits the requirements of the application and the needs of the user.

1.4 Specific Research Goals

While previous efforts have provided initial insight into the resource management problem, most results are bound to a specific application, protocol, or platform. The intend of this work is to seek the joint optimization of radio link performance and power consumption in dynamic spectrum environments. Under the context of jointly optimizing radio link performance while reducing energy consumption, this work seeks to provide answers to the following questions:

1. Polyphase filterbanks for dynamic spectrum access
   - (a) How can power consumption in software-defined radio platforms be reduced when operating in dynamic spectrum environments?
   - (b) How can efficient multiplexing schemes be implemented with polyphase filterbanks?
   - (c) What performance metrics are appropriate for analysis?
   - (d) What additional technologies are required to enable frequency-division multiplexing schemes for cognitive radios?
2. Power/resource management on SDR platforms
   - (a) What applicable power-monitoring techniques can enable feedback into the system?
   - (b) How can resources be managed in a dynamic spectrum environment to meet the user's needs?
   - (c) What implications does energy scalability have on system performance/reliability?
3. Radio cognition
   - (a) What algorithms are appropriate for radio learning?

(b) How does the additional computational load to enable radio learning burden the system?

(c) What utility functions are appropriate for radio learning through optimization of nodes?

(d) What performance metrics are appropriate for analyzing cognitive radios?

1.5 Thesis

The appropriate access mechanism by which to convey information over a wireless link depends upon constraints on energy consumption, channel conditions (spectrum availability, multi-path fading, additive noise, etc.), and the user's service quality needs. Incorporating learning algorithms into radios has been shown to yield improvements in spectral efficiency over traditional adaptive techniques. This document observes the impact resource management has on software-defined radio platforms by investigating the inherent trade-off between spectrum and processing efficiencies as they relate strongly to both the power consumed by the processor and the complexity of the algorithm which it can support. The analysis is specifically concentrated on digital signal processing elements commonly found in software-defined radio implementations of dynamic spectrum access-enabled cognitive nodes. This includes (but is not limited to) primary aspects of the radio's physical layer as it is defined in software, demonstrating that considerable resource savings can be gained without compromising the resulting quality of service to the user.

This dissertation takes a renewed look at the use of online resource monitoring to reduce the computational complexity of spectrum-agile software-defined radios, specifically by demonstrating that physical layer system performance can be minimally sacrificed to reduce computational complexity without compromising quality of service to the user. Spectrum agility is achieved by introducing new synchronization techniques for spectrally efficient multi-carrier signals capable of recovering carrier frequency and timing information in the presence of interference. Resource management is achieved by supervised intelligent heuristic-based learning algorithms which use software-level feedback of the radio's active resource consumption to optimize energy and processing efficiencies in dynamic spectrum environments.

1.6 Original Contributions

This document makes the following original contributions to the subject of resource management of software-defined radios operating in dynamic spectrum environments:

1. *Synchronization of Narrowband Channels*

   (1) a new maximum likelihood-based estimator for the timing offset parameter τ based on the matched-filter output and its derivative. The performance of this estimator is empirically efficient (it achieves the minimum estimator variance in noisy channels);

   (2) enhanced loop architectures and stability analysis for timing recovery of narrowband signals using polyphase filterbanks as a mechanism for resampling and interpolation;

(3) a novel preamble technique for quickly synchronizing the receiver to the beginning of a single-carrier frame in the presence of carrier frequency and sample timing offsets, as well as multi-path fading;

(4) a novel flexible framing structure capable of dynamically adjusting the modulation and forward error-correction schemes, payload length, and preamble phasing lengths. An accompanying synchronizer is introduced which can automatically detect the transmitter's configuration and correct for channel impairments. Its performance is characterized by simulation and validated by over-the-air laboratory experiments;

2. *Synchronization of Multichannel Communications Systems*

(5) a multi-carrier communications scheme (OFDM/OQAM) with spectral improvements over conventional OFDM is investigated to enable simultaneous dynamic spectrum sensing and access. Relying on polyphase filterbanks and offset quadrature amplitude modulation, the multi-carrier scheme is subject to well-known filter performance tradeoffs between computational complexity and spectral dynamic range. Literature on simultaneous spectrum sensing and access is scarce; further provided is a formal theoretical discussion of how OFDM/OQAM can be used for perfect reconstruction of detected signals for modulation classification while simultaneously decoding data subcarriers on other channels;

(6) new synchronization techniques for OFDM/OQAM are proposed which involves estimation of unknown channel parameters τ, ϕ, and ν as well as equalizer taps G_k. These techniques exist after the demultiplexer—in essence, operating in the frequency domain—and consequently are robust to narrowband interference disruption, providing significant performance improvement over conventional time-correlation techniques;

(7) a thorough performance comparison of OFDM and OFDM/OQAM in regards to their ability to be synchronized in various hostile channel conditions as well as their required computational complexities and theoretical data throughputs;

3. *Synchronization of Multichannel Communications Systems*

(8) a unifying methodology to monitor dynamic resource consumption on reconfigurable platforms is provided to facilitate radio resource management;

(9) comparison of various resource-monitoring techniques, both online and offline, and analysis on the efficacy of deploying them in embedded systems;

4. *Cognitive Engine Design*

(10) a demonstrable protocol to enable dynamic spectrum access while maintaining energy-quality scalability;

(11) theoretical modeling for cognitive engine design with a unifying framework to generically describe the relationship between radio parameters, goals, and environmental observables;

(12) a novel intelligent radio engine derived from a case-based reasoner to monitor the performance of the system using the above framework;

5. *Miscellany*

    (13) a new approximate square-root Nyquist filter design using the I_0-sinh Kaiser window. The proposed filter provides significant performance improvements over the root raised-cosine filter yet is much faster to compute than the best known filter which relies on iterating over the Parks-McClellan algorithm.

1.7 Organization

The organization of this work is as follows: Chapter 2 introduces the requirements to enable wireless access in dynamic spectrum environments, highlights the benefits of using OFDM/OQAM as a multi-carrier technology over traditional OFDM, and introduces a novel approximate square-root Nyquist filter design with nearly optimal performance while being simple to compute. Chapter 3 introduces several new techniques for synchronizing single-carrier communications signals to enable DSA systems. These techniques are based on maximum likelihood estimators and are computationally efficient, providing a solution applicable to software-defined radios. A resulting innovative framing structure is further provided with a performance validated over the air in a wireless indoor multi-path environment. Chapter 4 continues the results of the previous chapter by extending analysis to the synchronization of multi-carrier signals with an emphasis on OFDM/OQAM systems. New estimators for the unknown reference parameters are proposed which exhibit comparable performance to existing estimators while providing significant improvements in interference-laden channels, proving the applicability of OFDM/OQAM in dynamic spectrum environments. Chapter 5 proposes an architecture for online resource management in software-defined radios, demonstrating considerable savings in required computational bandwidths in adaptive radios. The system's performance is characterized through baseband simulations and confirmed with over-the-air experiments using the aforementioned synchronization techniques. Chapter 6 presents a unifying definition and framework for cognitive radio design by generically describing the relationship between a radio's parameters, service quality metrics, goals, and observed environment. Further introduced is a cognitive engine designed from a case database which abstracts from the radio application and demonstrates blind adaptation to unique scenarios while internally managing its own size.

Chapter 2

Physical Layer Mechanisms for Dynamic Spectrum Access

Because power consumption is strongly tied to a platform's physical layer, the portability of software-defined radios is heavily dependent upon the platform's capabilities. Additionally, link capacity is strongly affected by channel conditions and the changing wireless environment; consequently, power consumption has significant implications on the the ability of mobile cognitive radio platforms to enable dynamic spectrum agility. Polyphase filterbanks have been proposed for use in dynamic spectrum access in [4, 22] primarily due to their improved spectral estimation dynamic range over traditional OFDM. This chapter investigates the use of polyphase filterbanks as an energy-scalable dynamic spectrum access scheme for the reduction of power consumption on SDR platforms. A novel approximate square-root Nyquist filter design is presented which has significant performance improvements over the traditional square-root raised-cosine filter in both stop-band attenuation and inter-symbol interference. The resulting filter has similar properties to the best known hM-3 filter without requiring iterations over the Parks-McClellan algorithm.

2.1 Background

Dynamic spectrum access (DSA) is comprised primarily of two components: *sensing and awareness*—employing signal detection and classification to determine which frequency bands are unoccupied—and *access*—utilizing available frequency bands for data communication. The actual mechanisms used for sensing (e.g. signal detection and classification) are not the focus of this research but rather the design of the physical radio layer to enable these techniques simultaneously with access schemes with the explicit goal of minimizing resource consumption on the target platform. Although the task of spectrum sensing is handled externally, a physical radio layer strategy to enable sensing must be provided. These mechanisms must enable the two DSA components described above and preferably do so in a way that is efficient to the radio platform. This implies that a single medium access scheme that incorporates both spectrum sensing and spectrum access is preferable over two individual ones due to power and processing considerations. Because signal recognition is a critical component to DSA [23], this chapter investigates the application of both single- and multi-carrier methods to accommodate these techniques.

2.1.1 System Description

In order to facilitate spectrum access, spectrum sensing is categorized into two major groups. In the first category only the power spectral density of the signal is needed to make an decision about which frequency bands are available. Because the phase information is lost, this category is typically useful only for broad signal detection. For example, power spectral density alone cannot distinguish between different PSK-modulated signals as the underlying data are carried within the signal's phase. The second category also requires that the phase of the signal of interest is preserved so that the time series of the interference signal can be fed into a classifier to determine its modulation properties to help with avoidance. As such there are two basic categories for spectrum access: single- and multi-carrier methods. When non-contiguous spectrum use is required, multi-carrier methods are preferred as they can efficiently combine the channelizer with the demodulator and preserve the channel capacity. Alternatively, single-carrier methods are less complex, but are ill-suited for situations where the spectrum must be subdivided into many partitions.

A number of other access schemes are available for spectrum contention including direct-sequence and frequency-hopping spread spectrum signals. Radios using these techniques are designed to handle interference through spectrum diversity rather than avoiding it, and regard their own RF emissions as negligible interference to other receivers in the network. The Bluetooth [52] and 802.11b [53] standards, for example, operate in the 2.4 GHz ISM band and contend with interference from each other as well as other devices such as microwave ovens and cordless telephones. These underlay systems treat all other signals in the band as secondary users and have interference mitigation built into their protocols. When such devices operate alone in a network, the total throughput is not typically reduced by any significant amount; however, the impact underlay signals has on legacy users and other devices with protocols with sensitive forms of modulation can be significant. In this capacity only overlay techniques are considered to be viable DSA techniques as they are explicitly designed to avoid causing any interference to primary users.

2.1.2 Design Considerations

Regardless of which spectrum access mechanism is used, disabling the transmitter in the band of interest is requisite for spectrum sensing. Depending upon the protocol, any secondary users must vacate the band once primary users have been detected by the sensing mechanism. During its sensing time, the receiver must gather statistics about the band in order to accurately determine its availability. This must happen periodically to ensure that the occupied bandwidth remains vacant and therefore available for use by secondary users. As such, data links in single-carrier DSA environments are established by transmitting packetized frames in short bursts, followed by a quiet period for sensing. Alternatively, multi-carrier systems have the additional capability of disabling subcarriers which overlap the band of interest while simultaneously transmitting information on those which don't. In either case, this behavior has strong implications on the DSA node's physical layer design and suggests that in order to preserve spectral efficiency, the receiver must be able to synchronize very quickly to each frame. Fast acquisition in this regard is a key element for successful spectrum sharing in congested environments.

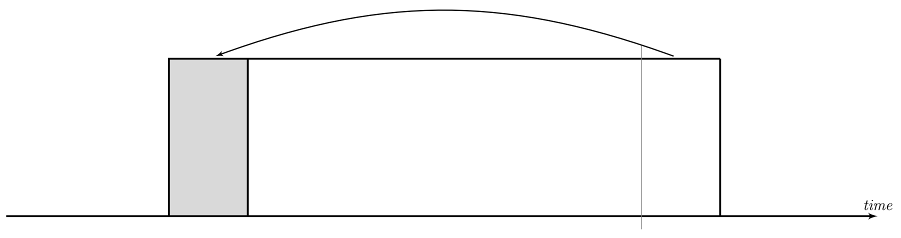
2.1.3 Single-carrier Methods

Spectrum allocation for single-carrier systems mandates the radio transmitter to use just one channel for communication between other nodes in the network. Bi-directional communication is accomplished through time-divsional duplexing on one channel (half duplex) or two channels simultaneously (full duplex). The channel is typically occupied for as long as it is available for use—that is, no primary user occupying that particular frequency band is in range and in danger of receiving interference. Nodes using single-carrier access schemes must guarantee a channel's availability by constantly scanning the band for the presence of other users and vacating occupied frequencies. Consequently, these nodes must incorporate time-division duplexing into their protocols in order to disable their transmitters; this is necessary even if full duplexing is used. Once the channel has been deemed no longer available for secondary use, the radios are required to move to a different, unoccupied frequency band. With the potential for switching channels frequently—particularly as the spectrum becomes more congested—secondary nodes are likely to disseminate sensing information throughout the network, sharing statistical usage information and conveying their preferred backup channels.

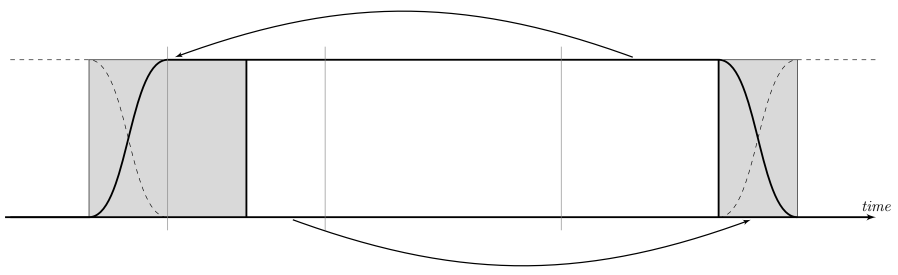
2.1.4 Multi-carrier Methods

Multi-carrier communications schemes slice a frequency band into many parallel narrowband partitions. This has several advantages over using a single wideband carrier with an equivalent data rate, particularly in hostile spectrum environments. Parallel transmission is effective in combating amplitude and delay distortion due to channel impairments as well as nonlinearities in the transmitter's and receiver's analog circuitry. Furthermore, multi-carrier communications are robust to frequency-selective fading channels in dense multi-path environments as equalization can be performed after the down-sampler with just a single tap per channel. While numerous multi-carrier systems exist, the majority of discussion is reserved to focus directly on orthogonal frequency-division multiplexing (OFDM) and OFDM using offset quadrature amplitude modulation (OFDM/OQAM), both of which support the maximum theoretical capacities of the spectrum.[1] Unlike single-carrier methods, these schemes can directly support simultaneous communication with spectrum sensing. This is accomplished by *notching* the transmitted spectrum—disabling certain subcarriers on the transmitter while using them for detection on the receiver side of the link. The next several sections provide detailed analysis on the efficacy of OFDM's applicability to this methodology and demonstrate that OFDM/OQAM is far better suited to the task despite its relative lack of interest in the radio engineering community. This is true both for spectral resolution (sensing) as well as improving channel capacity (access). The bulk of the discussion in the remainder of this chapter is reserved for multi-carrier methods for dynamic spectrum access.

[1]Other frequency-division multiplexing schemes are available, but the majority of them waste spectrum by not allowing subcarriers to overlap. For this reason their analysis has been omitted in this document.



(a) cyclic extension



(b) cyclic extension with tapering extension

Figure 2.1: OFDM cyclic extension diagram; highlighted regions are copied portions of the original symbol

2.2 OFDM

Orthogonal frequency division multiplexing (OFDM) is a parallel data transmissions mechanism, prevalent in wireless standards and has recently been proposed for spectrum sensing in conjunction with dynamic spectrum access [95, 2, 75]. Subcarriers which detect signal energy from primary users can easily be disabled, allowing for notches in the spectral bands. However, it has been demonstrated [94, 22, 26] that the side-lobe leakage due to the square pulse shape in OFDM is impractical for DSA systems with strict dynamic range requirements. This is due to the transform's lack of temporal windowing and results in a $\sin(\omega)/\omega$ spectral response. In fact, this lack of temporal windowing is precisely *why* OFDM is used for communications; the cyclic nature of the FFT allows the symbol to be extended by appending a portion of its tail to its beginning, as seen in Figure 2.1(a). The circular nature of the inverse FFT permits the receiver to begin sampling the symbol anywhere within the window of this extension without loss of orthogonality after the forward transform is taken. This is a fundamental component of Fourier analysis in which a time delay translates to a phase shift in frequency; all that needs to be done at the receiver is correct for this linear phase shift in frequency before demodulation. This provides a much simpler (and computationally efficient) solution to timing recovery than single-carrier systems. This is particularly useful for multi-access systems where several nodes all upload to a central point in which they can never be perfectly time-aligned; so long as their frames are all received by the base station within the duration of this cyclic prefix window; however, they can be uniquely separated without inter-symbol interference.

2.2.1 Window Tapering

The usefulness of OFDM for data communications comes at a steep cost when considering spectral compactness. The rectangular pulse shape amounts to a very poor $1/\omega$ side-lobe roll-off but is necessary to guarantee orthogonality between subcarriers. One would like to apply any of the well-known windowing functions such as the prolate spheroid [85], Kaiser-Bessel [55], or Blackman-harris [27] window to improve the stop-band attenuation of the symbol's spectral response. These windowing functions considerably reduce the side-lobe attenuation at the cost of slightly increasing the width of each subcarrier. Applying a temporal window, however, would consequently break the orthogonality conditions of OFDM and would result in considerable inter-symbol interference and symbol errors. One possible solution is to further increase the cyclic extension of the symbol and apply a tapering window which is flat over the duration of the symbol. An example of OFDM window tapering is presented in Figure 2.1(b); notice that applying the tapering window requires both a shaping prefix and a shaping postfix. Adjacent symbols are permitted to overlap at their edges which can significantly improve the spectral distortion by eliminating sharp transitions between symbols. While window tapering of OFDM symbols has been shown to significantly improve the out-of-band sidelobe leakage [88], this technique has several drawbacks:

1. the advantage of spectral side-lobe suppression is poor when the transition region between symbols is short. Consequently, the symbol prefix and postfix extensions need to be considerably large to see much benefit in spectral resolution;
2. these additional symbol extensions increase overhead and reduce the spectral efficiency (and therefore the theoretical capacity) of the system;
3. tapering only improves the transmitted spectrum. the receiver's FFT still results in poor time-frequency compactness.

2.2.2 Subcarrier Injection

Another proposed solution is to inject subcarriers which do not carry data, but whose purpose is explicitly to reduce the level of out-of-band energy [77]. Typically these subcarriers are placed at the edge of the guard band between the data subcarriers and the target spectral notch. Computation of these special subcarriers, however, can be cumbersome and time-consuming as they need to be generated for each data symbol. Furthermore, the increase of the guard band necessary for subcarrier injection reduces the spectral efficiency of the system. For the numerical results given in [77], a band just seven subcarriers wide and centered within the transmitted signal required 17 subcarriers to pull the side-lobes down to roughly 70dB. Furthermore, [77] demonstrate a self-interference performance hit just by injecting these sidelobes and increase their symbol error rate significantly as the stop-band attenuation increases.

2.2.3 Increasing Number of Subcarriers

Although techniques such as cancellation subcarriers, window tapering, and subcarrier deactivation have mitigated OFDM's spectral compactness problem [16], the overhead associated with these strategies significantly reduces OFDM's spectral efficiency and throughput. A simpler solution is to simply increase the number of OFDM subcarriers. This increases the length of the symbol which in turn compresses the width of each subcarrier. However, this technique has several problems:

1. the receiver is now more sensitive to frequency offsets as the subcarriers are more narrow, and spaced closer to one another;
2. this increases the coherence time and reduces the ability for the system to mitigate the effects of time-dependent fading channels (the channel is not ergodic) because now the symbol length is temporally extended;
3. this increases the computational complexity of the receiver (non-linearly) as the size of the transform is increased;
4. increasing the number of subcarriers degrades the system's peak-to-average power ratio which causes non-linear distortion in the analog circuitry (mixers, power amplifier). Backing off the transmit power to mitigate these effects is undesirable as it degrades the capacity of the link;
5. the $1/\omega$ side-lobe rolloff proves diminishing returns for increasing the number of subcarriers.

The cyclic prefix in OFDM is required by the nature of the inverse transform which results in temporal discontinuities. The lack of smooth transitions between symbols results in ringing when the signal propagates through RF hardware (mixers, up-converters, transmission lines, and the antenna) which would otherwise result in inter-symbol interference. Interestingly, much of the literature on OFDM touts the cyclic extension as a mechanism to eliminate inter-symbol interference due to the channel, while neglecting to mention that the interference is actually a byproduct of the transmission scheme itself. As it turns out, with a sufficient number of subcarriers (relative to the channel's coherence time), impairments due to inter-symbol interference fall secondary to spectral nulls due to multi-path fading. Zhou *et al.* stated the problem well in [3]: "in order for non-contiguous OFDM to be feasible in the presence of primary users' signals, stringent constraints need to be imposed…and if not met, the interference caused is detrimental to both the primary and secondary users." Permitting smooth transitions between symbols eliminates the need for the cyclic prefix as a means to absorb inter-symbol interference due to the RF hardware. This chapter will show it is indeed possible to devise an orthogonal multiplexing scheme which theoretically achieves the Shannon capacity while using filtered, overlapping subcarriers.

2.3 Polyphase Filterbank Channelizers

Parallel data communication channels subdivide a relatively wide bandwidth into smaller subchannels. Because each subchannel is relatively narrow, mulitichannel systems are robust to fading and impulsive noise. Furthermore, dynamic spectrum access can be achieved by simply enabling the subchannels (or "subcarriers") which exhibit the best link characteristics. Section 2.2 detailed the many drawbacks of using OFDM for dynamic spectrum sensing and access. Polyphase filterbanks have been proposed for spectrum sensing in [4, 22] due to their spectral compactness and spectral dynamic range improvements over traditional OFDM. This section gives a brief overview of the polyphase filterbank channelizer without going into great detail about the mathematics behind its design; it is sufficient to note, however, that a single low-pass filter and discrete Fourier transform can be used to service all aliased bands simultaneously. A more detailed description can be found in [28, 93], Chapter 4.

2.3.1 Filter design using Kaiser window

Optimum digital filters are those which meet a set of constraints related to the shape of its spectral response for a given length. These constraints are most often related to minimizing the maximum error (equiripple) in the pass-band and/or stop-band, fitting a prescribed shape in the least-squares sense, flattening the magnitude and/or phase response, or some combination of the above. Recursive digital filter designs (infinite impulse response) for these constraints have long been solved; however theory behind designing corresponding non-recursive filters (finite impulse response) has lagged. Within the past several decades strong advancements in the field of non-recursive digital filters have been made with regard to equiripple filters [74, 66] which use the Remez exchange algorithm to solve the minimax problem. This filter design process, however, is complex and subject to errors from finite machine precision. One of the simpler methods for non-recursive filter design is to apply a band-limiting window to a truncated sinc function [54], providing a moderate compromise between performance and complexity. The parameters for a low-pass filter are

$$\begin{aligned} N &= \text{filter order} \\ \delta_1 &= \text{pass-band ripple} \\ \delta_2 &= \text{stop-band ripple} \\ \omega_p &= \text{pass-band cutoff} \\ \omega_s &= \text{stop-band cutoff} \end{aligned}$$

all angular frequencies can be converted to cycle frequencies (e.g. $f_s = \omega_s/2\pi$) and are all relative to the sampling frequency F_s. The normalized quantities for transition bandwidth and and stop-band attenuation can be defined as $\Delta_f = (f_s - f_p)/2\pi$ and $A_s = -20\log_{10}(\delta_2)$, respectively.

The ideal low-pass filter has a rectangular response in the frequency domain and an infinite $\sin(t)/t$ response in the time domain. Because all time-dependent filters must be causal, this type of filter is unrealizable; furthermore, truncating its response results in poor pass-band ripple stop-band rejection. An improvement over truncation is offered by use of a band-limiting window. Let the finite impulse response of a filter be defined as

$$h(n) = h_i(n)w(n) \tag{2.1}$$

where $w(n)$ is a time-limited symmetric window and $h_i(n)$ is the impulse response of the ideal filter with a cutoff frequency ω_c, viz.

$$h_i(n) = \frac{\omega_c}{\pi}\left(\frac{\sin\omega_c n}{\omega_c n}\right), \quad \forall n \tag{2.2}$$

A number of possible windows could be used, but this work focuses on the Kaiser window for its systematic ability to trade transition bandwidth for stop-band rejection. The Kaiser window is defined as

$$w(n) = \frac{I_0\left[\pi\alpha\sqrt{1-\left(\frac{n}{N/2}\right)^2}\right]}{I_0\left(\pi\alpha\right)} \quad -N/2 \leq n \leq N/2,\ \alpha \geq 0 \tag{2.3}$$

where $I_\nu(z)$ is the modified Bessel function of the first kind of order ν and α is a shape parameter controlling the window decay. $I_\nu(z)$ can be expanded as

$$I_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{1}{4}z^2\right)^k}{k!\Gamma(k+\nu+1)} \tag{2.4}$$

The sum in (2.4) converges quickly due to the denominator increasing rapidly, (and in particular for $\nu = 0$ the denominator reduces to $(k!)^2$) and thus only a few terms are necessary for sufficient approximation. Computation of $I_0(z)$ for (2.3) can be difficult on computers with limited precision, particularly without the use of floating-point math. This is made evident by the fact that the magnitude of $I_0(z)$ grows extremely large while the ratio in (2.3) is bounded by $[0, 1]$. Appendix C gives a good approximation for computing $I_0(z)$ for the Kaiser window with little noticeable difference in the resulting filter response.

Kaiser gives an approximation for the value of α to give a particular sidelobe level for the window as [93, (3.2.7)]

$$\alpha = \begin{cases} 0.1102(A_s - 8.7) & A_s > 50 \\ 0.5842(A_s - 21)^{0.4} & 21 < A_s \leq 50 \\ 0 & \text{else} \end{cases} \tag{2.5}$$

where $A_s > 0$ is the stop-band attenuation in decibels. While exact the analytical relationship between a non-recursive low-pass filter's parameters (length, transition bandwidth, stop-band attenuation, and pass-band ripple) are not known, there exist many formulas which give approximations valid within a reasonable range and can serve as reasonable bounds. Kaiser has proposed a relatively simple formula in [54] for predicting the filter length from the ripple specifications and transition bandwidth as

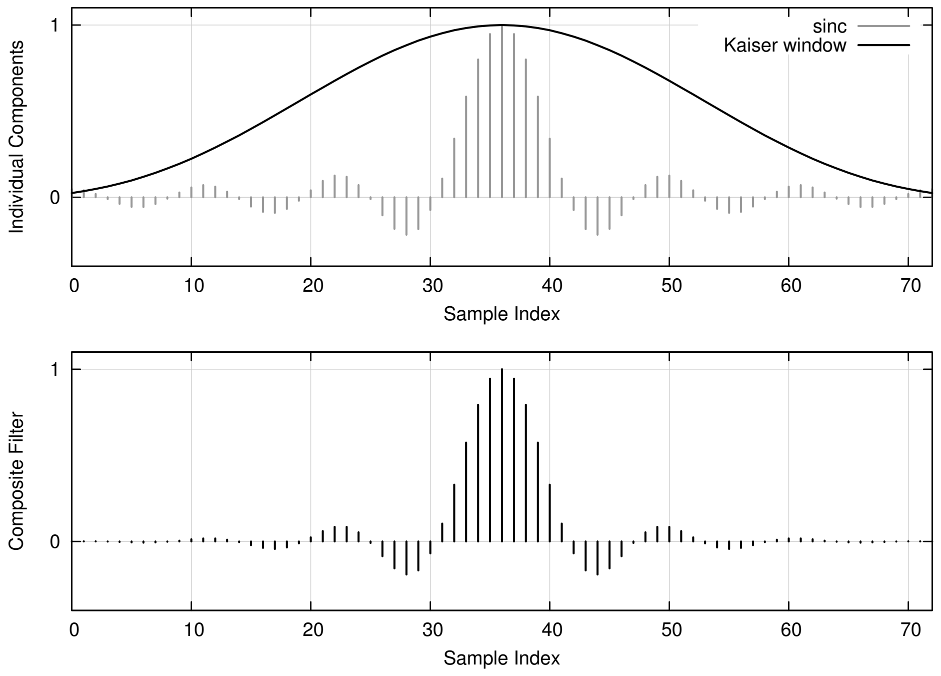
$$N \approx \frac{A_s - 7.95}{4.36\Delta f} \tag{2.6}$$

where again A_s and Δf represent the stop-band attenuation and transition bandwidth, respectively. The above equation provides a good, low-complexity approximation to the bounds on filter length and can be inverted easily for the stop-band attenuation as

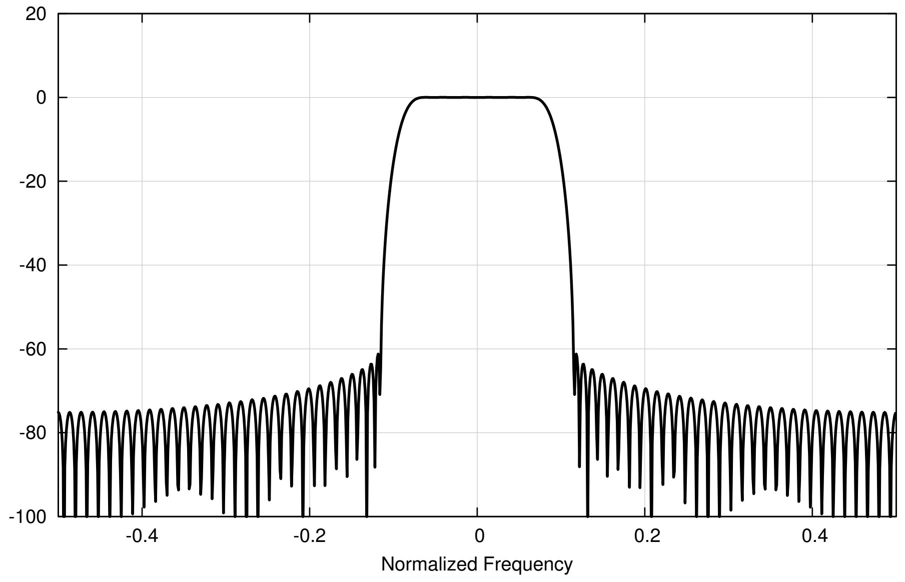
$$A_s \approx 14.36\Delta f N + 7.95 \tag{2.7}$$

Herrmann *et al.* gives a somewhat more accurate set of formulas in [48], but are non-invertible and are considerably more complex.

As an example, assume the design specifications for a low-pass filter are to have a cutoff frequency of $f_c = 0.09$, a transition bandwidth of $\Delta f = 0.05$, and a desired stop-band suppression of $A_s = 60$ dB ($\delta_1 = \delta_2 = 10^{-3}$). The resulting filter would need approximately $N = 73$ coefficients and a



(a) Time series



(b) Power Spectral Density, $H(\omega)$

Figure 2.2: Kaiser window filter design example, $f_c = 0.09$, $\Delta f = 0.05$, $A_s = 60\text{dB}$

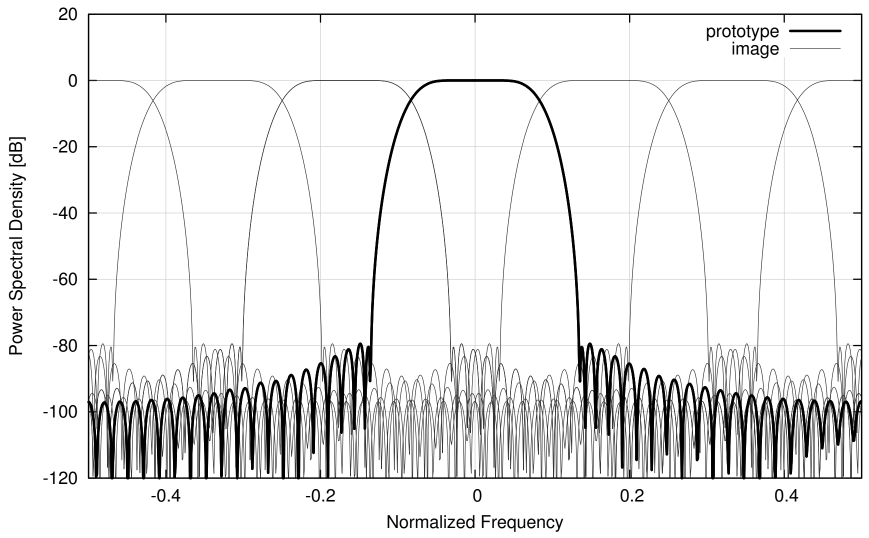


Figure 2.3: Polyphase channelizer spectrum design for $M = 6$ channels and a stop-band suppression of $A_s = 80\text{dB}$.

Kaiser window with a shape parameter $\alpha = 5.653$. Figure 2.2 demonstrates such a filter with Figure 2.2(a) showing the sinc and Kaiser window components to formulate a composite time series while Figure 2.2(b) depicts the frequency response of the resulting filter.

2.3.2 Polyphase Channelizer Design

The filter prototype described in the previous section can easily be used to create a multi-band channelizer. A channelizer is a combination of parallel down-converters and decimators which mixes the signal to baseband, applies an image-rejection filter, and then reduces the signal's sampling rate. This can be accomplished quite efficiently using a polyphase filter bank which aliases all the signals on top of each other, and then selects the appropriate alias by applying a set of phase rotators to the separate path of the polyphase partition. Conveniently, if all the bands are evenly spaced with center frequencies $\omega_k = 2\pi k/M$, they may be simultaneously preserved by applying a discrete Fourier transform to the output of the polyphase partition. This type of channelizer is known as a uniform DFT filterbank [93, Section 4.3.2] and can be used to realize a channelizer in a very efficient manner. An example of the aliased spectrum can be seen in Figure 2.3. The mixing process can be implemented very efficiently by constructing a polyphase partition of the filter by decimating its finite impulse response time series. The aliased output of each partition is selected by using the discrete Fourier transform. Figure 2.4 demonstrates this concept. For a more detailed description of polyphase filterbank channelizers for use in digital communications, the interested reader is referred to [28, Chapter 9].

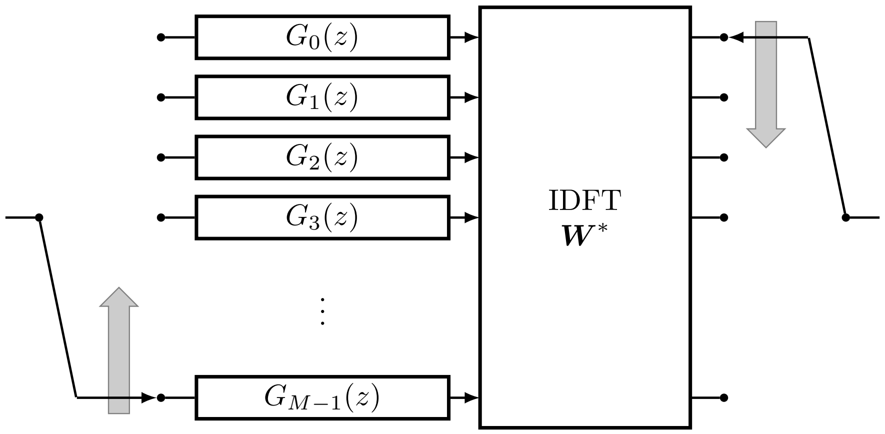


Figure 2.4: DFT polyphase filterbank channelizer system diagram.

2.3.3 (Almost) Perfect Reconstruction Filters

It is worth noting that polyphase filterbanks have the additional capability of (almost) perfectly reconstructing a signal spanning multiple channels [28]. The qualifier “almost” is necessary because while DFT filterbanks can never truly perfectly reconstruct the signal, [24, p. 189], the amount of alias distortion can be kept arbitrarily small through appropriate filter design techniques. Perfect reconstruction is particularly useful for advanced sensing algorithms which rely on methods other than simple energy detection [23]. Section 4.7 gives an example of how polyphase filterbanks can simultaneously be used for communications and interference signal reconstruction.

2.4 OFDM/OQAM

While traditional OFDM provides many benefits to radio design (e.g. the parallel down-converter uses computationally efficient FFTs), its limitations for spectrum sensing are apparent through its problems with inferior sidelobe leakage [94] due to unfiltered pulses. Section 2.2 has highlighted many of the drawbacks OFDM has for dynamic spectrum usage, particularly due to its lack of spectral compactness, and have demonstrated that a parallel channelizer can be efficiently realized through a polyphase filterbank and a single FFT in Section 2.3. It would be convenient to combine the benefits of efficient parallel data communication (OFDM) with the spectral compactness of polyphase filterbank channelizers. As it turns out, this is possible with a few minor modifications to the basic filterbank channelizer design.

Pioneering work on the spectrally efficient orthogonal frequency division multiplexed offset quadrature amplitude modulation (OFDM/OQAM) was done by Chang and Saltzberg in the 1960s [17, 83] and was later extended to use the discrete Fourier transform and polyphase filterbanks in [49, 12]. OFDM/OQAM improves upon OFDM by allowing any square-root Nyquist filter to be used as a prototype for each subcarrier. This promotes both improved spectral efficiency (in terms of throughput) and out-of-band interference suppression. In this approach successive carriers are overlapped by 50% causing cross-talk between the real and imaginary components of successive subcarriers. This cross-talk is suppressed by staggering the in-phase and quadrature components of successive channels by half a symbol. Chapter 4 presents a more detailed discussion of OFDM/OQAM design.

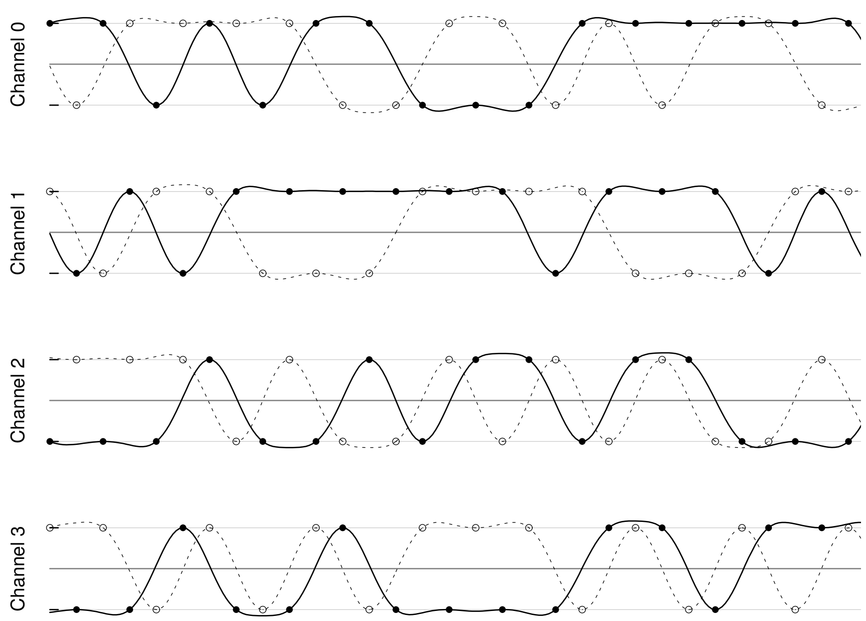
Figure 2.5 demonstrates a four channel OFDM/OQAM system in both time- and frequency-domain representations. Each of the four channels in Figure 2.5(a) uses offset 16-QAM—two bits per I/Q channel. Notice that the even channels (0 and 2) delay the quadrature component by $T/2$ while the odd channels (1 and 3) delay the in-phase component. Unlike OFDM, transitions between symbols are smooth which implies that each channel is spectrally compact; this is verified in Figure 2.5(b). Furthermore, adjacent channels are permitted to overlap by as much as half the symbol rate $(1/2T)$ without inter-channel interference. This relaxes the constraint on the prototype filter's transition bandwidth (Δf) which reduces its length for a specified stop-band attenuation (A_s). Furthermore, most of the filter's parameters can be arbitrarily specified, giving the radio flexibility to adapt to different environments, and potentially trade computational complexity for spectral efficiency.

The results presented in this section suggest that the performance of OFDM/OQAM is significantly dependent upon its filter design. Indeed this is the case and has been proven in [83] for half- and full-cosine rolloff filters. The next section introduces a novel approximate root Nyquist filter with significant improvements over the well-known root raised-cosine pulse which can be expressed in closed form.

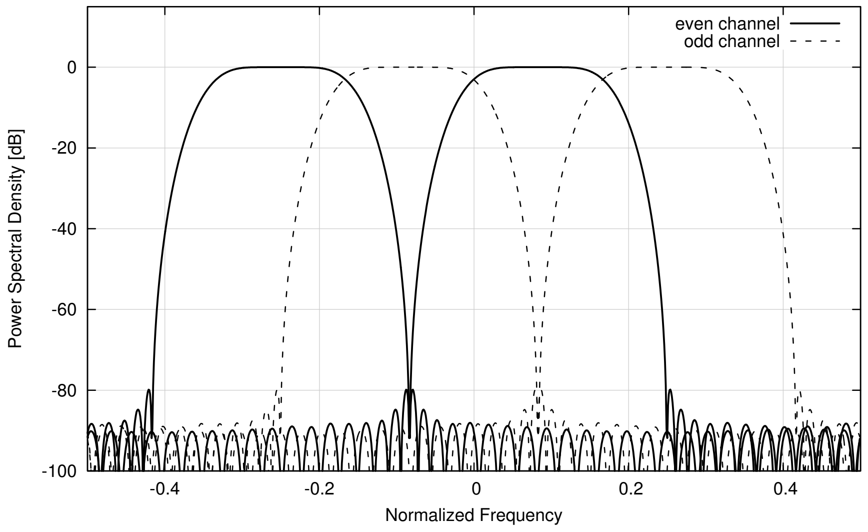
2.5 New Approximate Square-root Nyquist Filter Design

Nyquist's criteria for designing a band-limited filter exhibiting zero inter-symbol interference (ISI) is for the magnitude of the spectral response of a linear phase filter to be symmetric about the down-sampling frequency, $H(\omega) + H(\pi/T - \omega) = 1$ for $0 \leq |\omega| \leq \pi/2T$. Splitting the filtering operation into two components (one at the transmitter and one at the receiver) assures optimal performance in the presence of additive white Gauss noise. One particular pulse used commonly in academics is the raised-cosine pulse, which is likely used so often because its time-domain representation can be easily described in closed form [76, p. 560]. However it is well-understood that the spectral response of the raised-cosine filter degrades quickly as the pulse duration is truncated to a realizable finite time series. Furthermore, its spectral factorization (the root raised-cosine pulse) exhibits even worse spectral response when truncated.

An improved square-root Nyquist filter proposed in [29] called the hM-3 filter provides a strong improvement over the raised-cosine filter. The hM-3 filter iterates over the well-known Parks-McClellan algorithm which uses Chebyshev polynomials to solve the minimax problem for non-recursive filter design. The hM-3 design strategy is to modify the band edges of the Nyquist filter in the Parks-McClellan algorithm such that the resulting filter approximates a square-root Nyquist response and has a minimal inter-symbol interference. The result is a high-precision digital filter whose stop-band attenuation, inter-symbol interference, and transition bandwidths are significantly improved over the root raised cosine filter of the same length. Computing its solution, however, can prove to be a cumbersome task, particularly for filters of a high order. The memory requirements of the Parks-McClellan algorithm increase with the filter order (the suggested grid density is $20N$)



(a) Time Series



(b) Power Spectral Density

Figure 2.5: Example of a 4-channel OFDM/4-OQAM baseband signal and frequency power spectral density representations. Odd channels delay in-phase (solid) by half a symbol, even channels delay quadrature (dashed) by half a symbol. Frequency overlap is 50% for each adjacent channel.

and the algorithm can potentially fail to converge if the machine precision isn't sufficient. This is particularly noticeable for filters with very large stop-band attentions where the resulting Chebyshev error, δ, is very small in which case the computer's numerical representation of $1 \pm \delta$ can easily have insufficient resolution. Furthermore, the hM-3 filter requires many iterations of the Parks-McClellan algorithm for an optimal solution, which results in significant computational overhead. One possible work-around is to pre-compute the filter coefficients offline for a range of potential filters, and then use polynomial interpolation to resample the filter taps (if necessary); however this restricts the availability to design any filter desired at run-time, and can potentially lead to a sub-optimal solution.

As described in Section 2.3.1, a well-known non-recursive filter design technique is to apply a symmetric tapering window to a sinc function. Setting the fundamental frequency of the sinc to the down-sampling rate guarantees the resulting filter meets Nyquist's zero inter-symbol interference criteria and is the motivation behind using this as a design technique for square-root Nyquist filters. Transformation of a Nyquist filter to its square-root Nyquist counterpart can be complicated. One solution is to compute a spectral factorization on the time series of the derived filter [93, Section 3.2.5] by decomposing the Nyquist filter polynomial $H(z)$

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + \cdots + h_{2N-2} z^{2N-2}$$

into a square-root Nyquist filter $G(z)$ from its root components, viz.

$$H(z) = G^2(z) = \left[g_0 + g_1 z^{-1} + \cdots + g_{N-1} z^{N-1} \right]^2$$

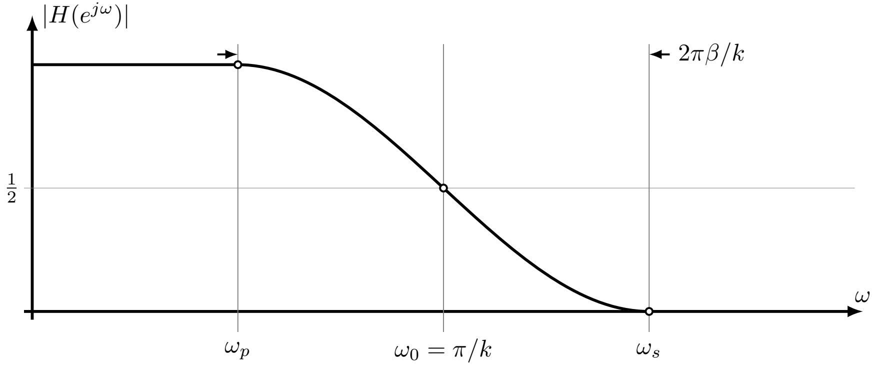
This decomposition can be accomplished by performing a search for the $2N-2$ complex roots of $H(z)$ as $\sum_{k=0}^{2N-3} (1 - r_k z^{-1})$ and then grouping them appropriately such that once expanded, $G(z)$ has only real coefficients. While this option is possible, it is unrealistic for several reasons:

1. the roots of $G(z)$ must be real and thus $H(z)$ cannot have a negative frequency response (i.e. $H(e^{j\omega}) \geq 0, \forall_\omega$). It is unlikely that the Nyquist filter's spectral response will be positive for all frequencies, as most windowed sinc functions do have a slight negative response in the stop-band. A possible work-around is to force a non-negative spectral response by adding a small offset of δ_2 (the stop-band ripple); however, this effectively halves the stop-band attenuation of the filter [93, Figure 3.2-9];

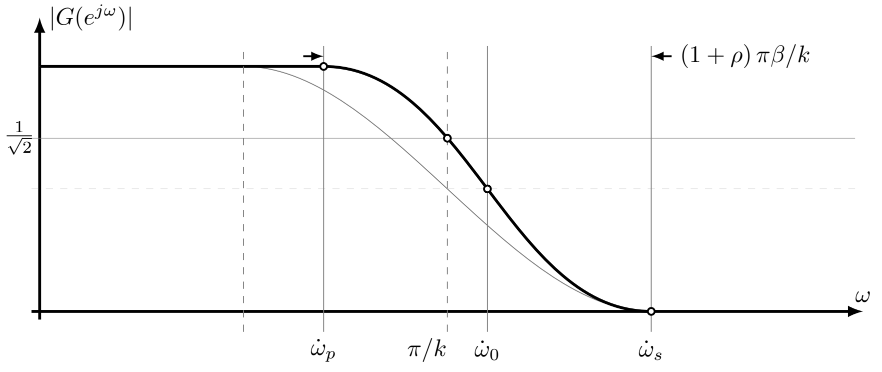
2. determining the polynomial roots can be time-consuming and computationally intense. Convergence of many well-known root-finding algorithms (such as the Durand-Kerner method and Bairstow's method) is often contingent upon the polynomial being well-conditioned, and can even fail to converge if the initial conditions are not appropriately set;

3. in order for the the zero-phase criteria to be met, the roots of the polynomials must be appropriately grouped together—a cumbersome task in and of itself [79, Section XII]—and is particularly difficult as the size of the filter increases;

4. algorithms which do not rely on computing the zeros of $H(z)$ exist [93, Appendix D]; however, they are typically more computationally involved than is desirable.



(a) Nyquist filter prototype



(b) Approximate square-root Nyquist filter prototype

Figure 2.6: Kaiser approximate square-root Nyquist filter prototype with bandwidth adjustment factor $0 < \rho < 1$.

Alternatively, The hM-3 filter design approach can be used by searching for the appropriate band edges which minimize the resulting filter's inter-symbol interference. Motivated by [29] in which the authors accept that the filter's spectral response won't be perfectly symmetric, this section investigates the effects of adjusting the pass-band cutoff frequency of a non-recursive filter design using a temporal windowed sinc in order to approximate a square-root Nyquist filter.

2.5.1 Filter Design Description

To begin, observe the properties of an ideal Nyquist filter $H(z)$ which has the frequency response $H(e^{j\omega})$, operating at k samples per symbol with a delay of m symbols and an excess bandwidth factor β. The resulting filter has $N = 2km + 1$ coefficients. Figure 2.6(a) demonstrates the ideal properties of a Nyquist filter where $\omega_0 = \pi/k$ is the half-power cutoff frequency of the filter, and $\omega_p = \pi(1-\beta)/k$ and $\omega_s = \pi(1+\beta)/k$ are the pass-band and stop-band cutoff frequencies of $H(z)$, respectively, normalized to the sampling frequency. The resulting transition bandwidth is

$\Delta\omega = \omega_s - \omega_p = 2\pi\beta/k$. An approximation to the filter's square root response is a new filter whose transition bandwidth is adjusted such that it passes directly through $1/\sqrt{2}$ at the critical sampling frequency, ω_0, as demonstrated in Figure 2.6(b). The results given in [29] conclude that the approximate root Nyquist filter $G(z)$ has a transition band similar to that of $H(z)$, but takes a compressed form in which the pass-band and half-power cutoff frequencies are increased ($\omega_p \to \dot{\omega}_p$ and $\omega_0 \to \dot{\omega}_0$). Furthermore, The stop-band frequency of the filter (ω_s) remains the same in order to preserve the excess bandwidth design parameter. This effectively decreases the transition bandwidth of the filter, and as a result its stop-band attenuation is slightly higher. This performance degradation, however, is not nearly as poor as with taking the true spectral factorization of $H(z)$ in which A_s is reduced by half.

The filter transformation of $H(z)$ to $G(z)$ in Figure 2.6(b) can be described by a single parameter ρ which simultaneously accounts for both the increase in the cut-off frequency as well as the decrease in transition bandwidth. Effectively, the transition bandwidth is decreased from $\Delta\omega = 2\pi\beta/k$ to $\Delta\dot{\omega} = (1+\rho)\,\pi\beta/k$ for $0 < \rho < 1$. The resulting filter parameters can be re-defined as

$$\dot{\omega}_p = \frac{\pi}{k}\Big[1 - \beta\rho\Big] \tag{2.8}$$

$$\dot{\omega}_0 = \frac{\pi}{k}\Big[1 + \beta(1-\rho)\Big] \tag{2.9}$$

$$\dot{\omega}_s = \frac{\pi}{k}\Big[1 + \beta\Big] \tag{2.10}$$

Therefore, for a given square-root Nyquist filter design with k samples per symbol, m symbols of delay, and an excess bandwidth factor of β, all that needs to be done is to search for the value of ρ which minimizes its inter-symbol interference.

2.5.2 Theory

Filter design using the Kaiser-Bessel window (see Section 2.3.1) has been shown to produce a near optimal prototype filter in terms of minimizing stop-band energy [93, Section 3.2.2]. Additionally, it is an accurate yet computationally efficient approximation to the well-known family of filters using prolate spheroid sequences which maximize the ratio of in-band to out-of-band energy for a given cut-off frequency. Furthermore, it provides an extensible mechanism to trade transition bandwidth for stop-band attenuation, and thus is the basis of the new approximate square-root Nyquist filter design. The Kaiser window, defined by (2.3), has a spectral response [55]

$$W(\omega) = \frac{N\sinh\left[\sqrt{\alpha^2 - \left(\frac{N\omega}{2}\right)^2}\right]}{I_0(\alpha)\sqrt{\alpha^2 - \left(\frac{N\omega}{2}\right)^2}} \tag{2.11}$$

where $\sinh(z) = \frac{1}{2}\left[e^z - e^{-z}\right]$ is the hyperbolic sine function and and I_0 is the modified Bessel function of the first kind. Let $f(z) = \sinh\sqrt{z}/\sqrt{z}$. The first observation is that $f(z)$ is purely real for all z:

$$f(z) = \begin{cases} \frac{1}{2\sqrt{z}}\left[e^{\sqrt{z}} - e^{-\sqrt{z}}\right] & z \geq 0 \\ \sin\sqrt{-z}/\sqrt{-z} & z < 0 \end{cases}$$

Furthermore, the first negative zero-crossing of $f(z)$ is at $z = -\pi^2$. Therefore, the first null of the Kaiser window is at $\omega = \frac{2}{N}\sqrt{\alpha^2 + \pi^2}$, or $f = \frac{1}{\pi N}\sqrt{\alpha^2 + \pi^2}$. The first side-lobe of $f(z)$ is located at $z = -20.19$ and has a value of approximately -0.217234. Therefore, the first side-lobe of the Kaiser window is at $\omega \approx \frac{2}{N}\sqrt{\alpha^2 + 20.19}$, or $f \approx \frac{1}{\pi N}\sqrt{\alpha^2 + 20.19}$ which gives a side-lobe value of $W(\omega) = -0.217234N/I_0(\alpha)$. Notice that as the filter bandwidth increases with α, its first side-lobe (commensurate of its stop-band attenuation) decreases with α.

The finite impulse response of a filter designed by applying a windowing mask to a sinc function is defined in (2.1). The resulting filter's frequency response is therefore the convolution of the Fourier transform of the individual components, viz.

$$H(\omega) = H_i(\omega) * W(\omega) = \int_{-\infty}^{\infty} H_i(\xi)W(\omega - \xi)d\xi \tag{2.12}$$

where $H_i(\omega)$ is rectangular with a one-sided bandwidth of ω_c. $H(\omega)$ can then be reduced to

$$H(\omega) = \int_{-\omega_c}^{\omega_c} W(\omega - \xi)d\xi = \int_{\omega-\omega_c}^{\omega+\omega_c} W(\xi)d\xi \tag{2.13}$$

It is also important to keep in mind that as the filter's cutoff frequency is increased, it is necessary to simultaneously decrease the transition bandwidth to preserve the excess bandwidth design parameter. The challenge is therefore to find the cutoff frequency ω_c and Kaiser window parameter α which minimize the resulting inter-symbol interference.

2.5.3 Evaluating Inter-Symbol Interference

Inter-symbol interference is the auto-correlation of the root-Nyquist filter response evaluated at non-zero integer multiples of the symbol period. The resulting matched-filter output with a delay τ is

$$h(\tau) = \int_{-\infty}^{\infty} g(t)g(t - \tau)dt \tag{2.14}$$

which, when expressed in discrete form with an integer lag ℓ, gives

$$h(\ell) = \sum_{j=|\ell|}^{N-1} g(j)g(j - |\ell|), \quad -N \leq \ell \leq N \tag{2.15}$$

The inter-symbol interference spans the full duration of $h(\ell)$ and so it is useful to express ISI in terms of its root mean-square value, viz.

$$\zeta_{rms}^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{h(ki)}{h(0)}\right)^2 \tag{2.16}$$

2.5.4 Optimizing and Approximating ρ

By nature of using a Kaiser-Bessel window for filter design, its response will automatically have the nearly optimum performance in terms of stop-band rejection for the specified length and transition

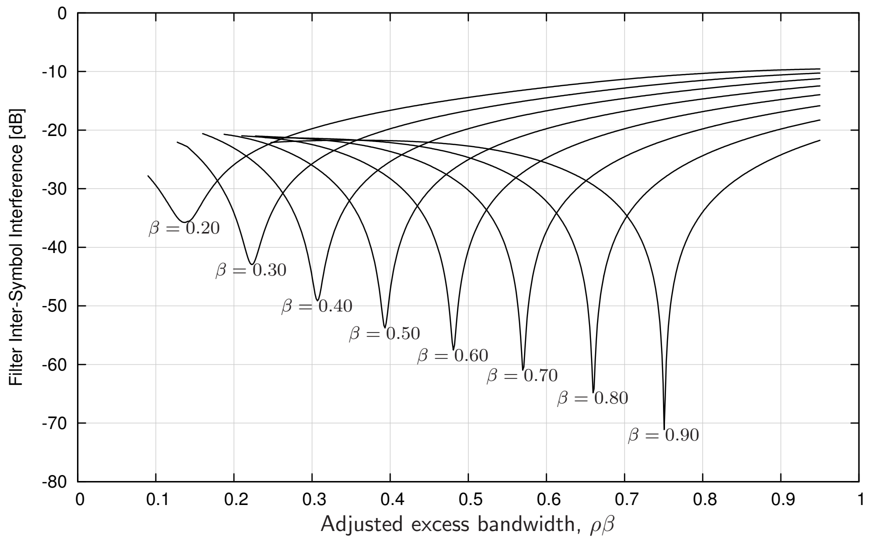


Figure 2.7: r-Kaiser square-root Nyquist filter design, ISI vs. adjusted excess bandwidth ($\rho\beta$) for a prototype filter with $k = 2$ samples/symbol and a delay of $m = 3$ symbols.

bandwidth. Furthermore, the stop-band attenuation is for all practical purposes independent of the filter's cutoff frequency, and thus independent of k. It is therefore only necessary to adjust the transition band of the filter to minimize the inter-symbol interference of the resulting approximate square-root Nyquist filter. The filter can be completely determined by specifying its over-sampling factor k, its delay in terms of symbols m, and its excess bandwidth factor β. Because the filter response is relative to k, the dependency ρ has on k is negligible and can be ignored.

As Figure 2.7 demonstrates, inter-symbol interference is clearly dependent upon the choice of ρ for a particular excess bandwidth factor β and filter delay m. Additionally, the minima in Figure 2.7 for each curve increase with β and are increasingly more sensitive to the bandwidth adjustment. Now all that remains is to perform a search for the value of ρ which minimizes ζ_{rms}. Iterative search methods, while robust, are typically undesirable as they require time-consuming computation. Figure 2.8 plots β vs. ρ for several values of the filter delay, in which the strong relationship between the two is clearly visible, and suggests that an approximating function can be used to relate the two. Empirically, it was found that the following formula provides a very good approximation to the sample data:

$$\hat{\rho} = c_0 + c_1 \log\big(\beta\big) + c_2 \log^2\big(\beta\big) \tag{2.17}$$

The values of c_0, c_1, and c_2 were optimized to the sample data in the minimum mean square error sense using a gradient search method and are given in Table 2.1 for different filter delays m. The approximation given by (2.17) is also plotted in Figure 2.8, represented by the solid lines, and suggests a good fit to the sample data.

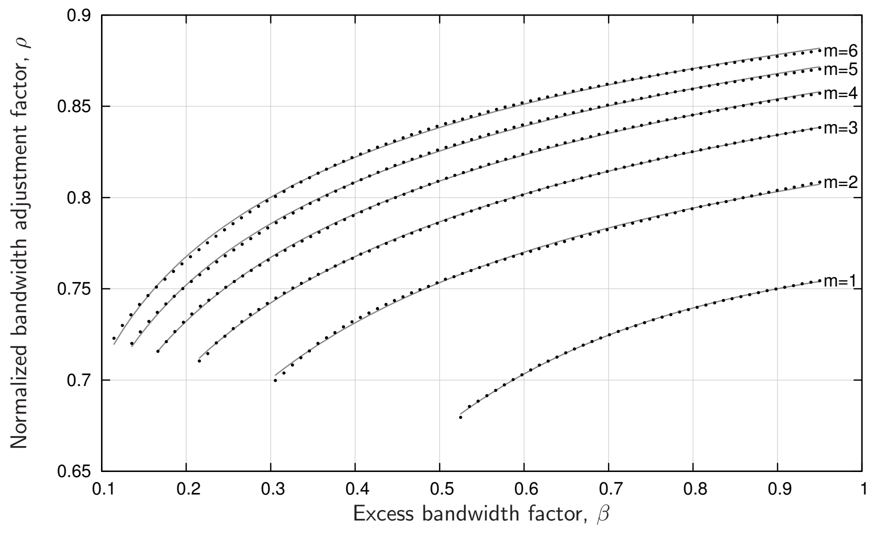
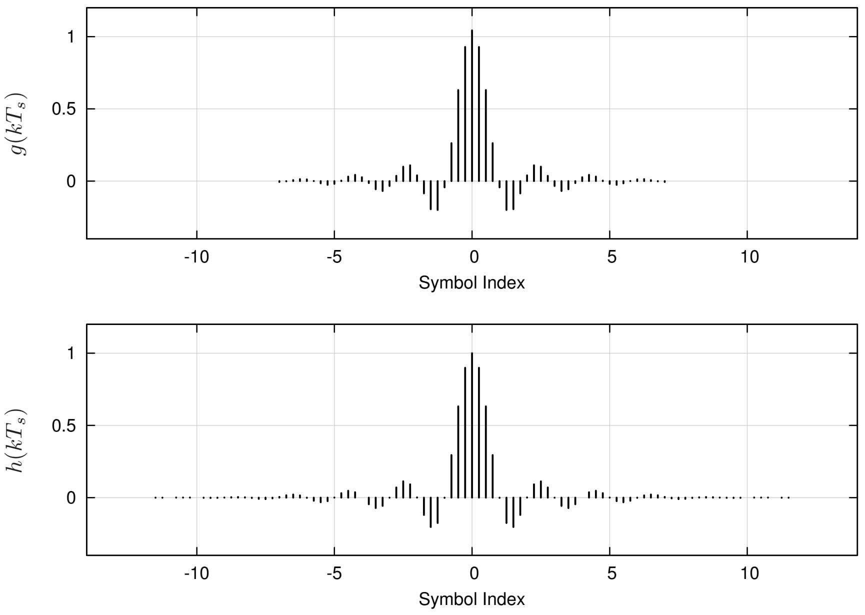


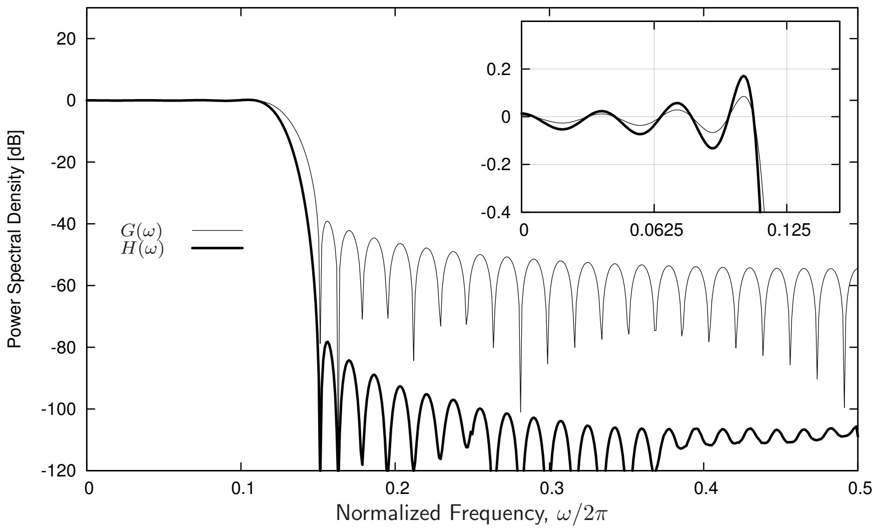
Figure 2.8: r-Kaiser optimum value of ρ vs. excess bandwidth factor β for varying filter delays $m \in [1, 6]$ along with the proposed approximation given by (2.17).

Table 2.1: Coefficients to approximate $\hat{\rho}$, given by (2.17)

m	c_0	c_1	c_2
1	0.75731975	0.05968561	-0.08998869
2	0.81099409	0.07149210	-0.01679019
3	0.84243834	0.07660834	-0.00545734
4	0.86171466	0.07264388	-0.00484673
5	0.87516469	0.06799082	-0.00525903
6	0.88511246	0.06330929	-0.00600636
7	0.89297986	0.05937212	-0.00660814
8	0.89985144	0.05688106	-0.00667160
9	0.90810889	0.05885705	-0.00531144
10	0.81103307	-0.12123501	-0.06670734
11	0.73827326	-0.23771790	-0.10142587
12	0.69563413	-0.29050779	-0.11261396
13	0.67214876	-0.31101137	-0.11386558
14	0.65340453	-0.31693947	-0.10999438
$m > 14$	$0.056873 \log(m) + 0.781388$	0.05596561	-0.00388152

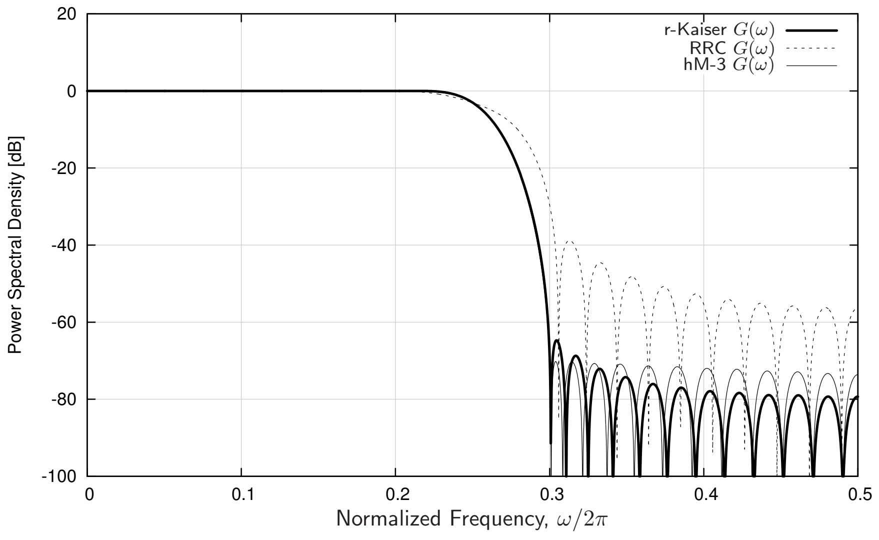


(a) Time series

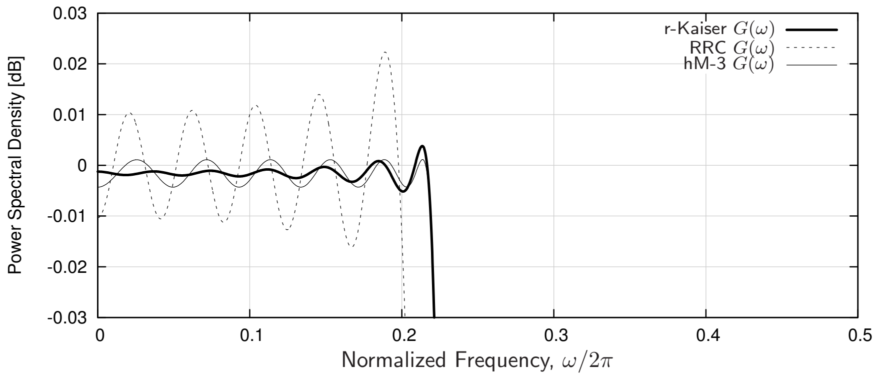


(b) Power Spectral Density

Figure 2.9: r-Kaiser example, $k = 4$, $m = 7$, $\beta = 0.2$



(a) Frequency Response



(b) Frequency Response, Pass-band Detail

Figure 2.10: $G(\omega)$ for r-Kaiser, root raised-cosine, hM-3 with $k = 2$ samples/symbol, a filter delay of $m = 12$ symbols, and an excess bandwidth factor of $\beta = 0.2$

2.5.5 Results

An example of designing an r-Kaiser filter can be found in Figure 2.9 for $k = 4$ samples/symbol, a filter delay of $m = 7$ symbols, and an excess bandwidth factor of $\beta = 0.2$. Computing $\hat{\rho}$ from (2.17) using coefficients from Table 2.1 with $m = 7$ gives $\hat{\rho} = 0.780351$. After optimization the true bandwidth adjustment factor is $\rho = 0.775543$ which leads to a transition bandwidth of $\Delta f = \rho\beta/k = 0.0388$, a cutoff frequency $f = [1 + \beta(1 - \rho)]/k = 0.261$, and a stop-band attenuation $A_s = (2km + 1)14.6\Delta f + 13 = 39.47$dB, resulting in a Kaiser window parameter $\alpha = 3.332$ as given by (2.5). The resulting filter has a root mean-square inter-symbol interference of -54.6dB and a maximum ISI of -45.6dB.

Another example comparing the spectral response of the r-Kaiser filter with the hM-3 and root raised cosine filters is depicted in Figure 2.10 for $k = 2$ samples/symbol, a filter delay of $m = 12$ symbols, and an excess bandwidth factor of $\beta = 0.2$. Notice that in Figure 2.10(a) that the r-Kaiser and hM-3 filters out-perform the RRC filter in terms of stop-band attenuation by more than 20dB. For the design parameters, the first null of $G(\omega)$ should ideally exist at $\omega/2\pi = (1 + \beta)/2k = 0.3$; while both the r-Kaiser and hM-3 filters meet this criteria, the RRC filter's first null exists at approximately 0.31, which effectively increases its spectral bandwidth. A detailed view of the filters' pass-bands can be found in Figure 2.10(b) in which the large pass-band ripple of the RRC filter is clearly visible. Furthermore, the spectral responses of the r-Kaiser and hM-3 filters are quite similar, in terms of pass-band ripple, stop-band attenuation, and transition bandwidth. To compare, the r-Kaiser filter has an RMS ISI of -73.9dB while the root raised-cosine filter has only -65.1dB.

2.6 OFDM/OQAM vs. OFDM

The proposed benefits of OFDM/OQAM over traditional OFDM have been highlighted previously in this chapter. This section compares the theoretical performance of the two in terms of spectral efficiency and theoretical computational complexity. The intent here is to demonstrate that OFDM/OQAM has a significant improvement of spectral resolution over OFDM while being only marginally more complex to compute on a reconfigurable platform. The r-Kaiser filter design, proposed in the previous section, has demonstrated an improved spectral response over the conventional root raised-cosine filter while proving much easier to compute than the hM-3 filter. As such it will be used throughout the discussion as a prototype filter for OFDM/OQAM.

2.6.1 Spectral Response

A comparison of spectral efficiency for OFDM/OQAM versus traditional OFDM can be found in Figure 2.11. With the same data rate, OFDM/OQAM demonstrates more than 20dB sidelobe suppression in its notched band over OFDM without a significant increase in computational complexity. This spectral compactness holds even with nearly eight times fewer subcarriers than OFDM. It is important to reiterate that the prototype filter can be designed with an arbitrary length and excess bandwidth factor, allowing the designer any reasonable stop-band attenuation that is desired.

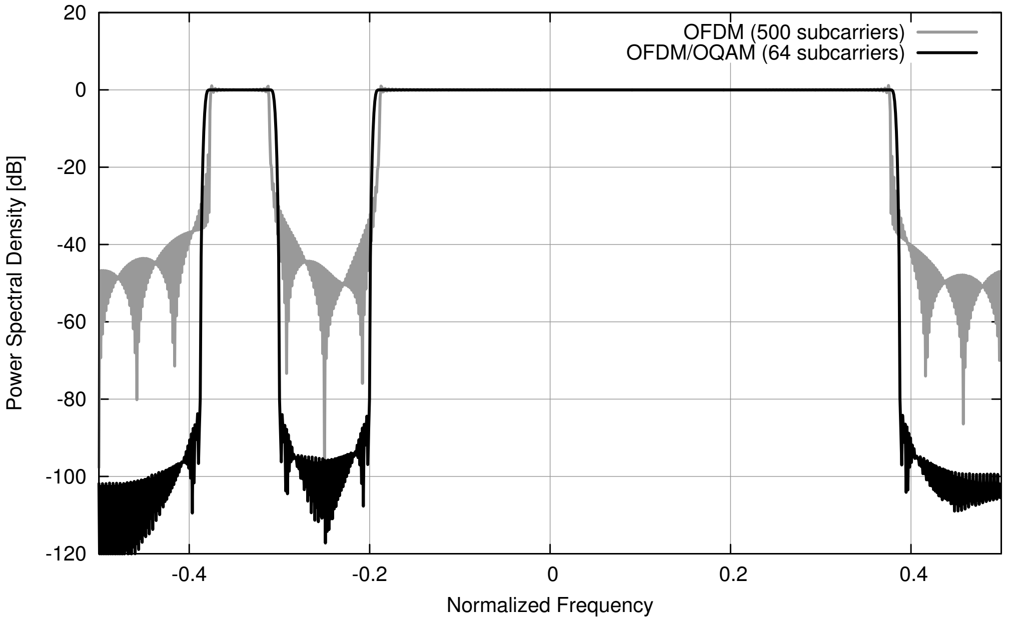


Figure 2.11: Power spectral density: OFDM/OQAM vs. traditional OFDM, 12.5% subcarriers disabled.

2.6.2 Computational Complexity

Figure 2.11 demonstrates incontrovertibly that OFDM/OQAM has many spectrum compactness advantages over traditional OFDM with only a fraction of the subcarriers. However, one of the potential drawbacks to allowing the subcarriers to be filtered is the increase in baseband processing required to compute the additional multiply-and-accumulate operations. This section compares the respective computational complexities of computing the down-conversion for both OFDM and OFDM/OQAM, neglecting overhead due to synchronization (e.g. carrier frequency offset correction).

Well-known fast Fourier transform (FFT) algorithms compute the N-point discrete Fourier transform (DFT) in $\mathcal{O}(N \log N)$ floating-point operations (flops). The fastest of these [31] can compute a split-radix (i.e. $N = 2^m$) FFT having a flop count of

$$\frac{34}{9} N \log_2 N - \frac{124}{27} N - \frac{2}{9} (-1)^{\log_2 N} \log_2 N + \frac{16}{27} (-1)^{\log_2 N} + 8$$

which is approximately $(89/18) N \log_2 N$ floating-point operations [86]. The OFDM receiver only computes the forward DFT once for every symbol, and therefore the complexity of the receiver is approximately

$$\mathcal{K}_{ofdm} = F_s \left(\frac{M}{M + N_{cp}} \right) \frac{89}{18} M \log_2 M \tag{2.18}$$

where F_s is the sample rate of the receiver, M is the number of subcarriers of the multiplexer, and N_{cp} is the length of the cyclic prefix in samples. This estimate, of course, provides a tight lower bound on the number of flops required to receive the signal, and does not incorporate any of the necessary synchronization operations.

For an M-channel filterbank, the DFT can be performed efficiently using FFT techniques. The results given in Section 2.6 suggest that OFDM/OQAM does not require as many subcarriers as OFDM due to its spectral compactness resulting from its temporal shaping; however, filtering the subcarriers introduces additional complexities. Furthermore, because the in-phase and quadrature-phase components of the signal are staggered by half a symbol, the receiver requires two independent analysis filterbanks [24, Fig. 7.12]. This effectively doubles the complexity of the receiver as each down-converter needs to run independently of the other. If the filter has a delay of m OFDM/OQAM symbols, then it has $N = 2Mm + 1$ coefficients. A single analysis filterbank requires both the M-point FFT as well as a single N-tap filter for each output symbol. Therefore the complexity of the OFDM/OQAM analysis filterbank is

$$\mathcal{K}_{ofdm/oqam} = 2F_s \left(\frac{89}{18} M \log_2 M + 2Mm + 1 \right) \tag{2.19}$$

For a fixed number of subcarriers M, OFDM/OQAM has a complexity roughly double that of OFDM; however, because of its spectral compactness, fewer subcarriers are needed.

2.7 Conclusions

This chapter has investigated the efficacy of using polyphase filterbanks as an energy-scalable dynamic spectrum access technique, and has highlighted the advantages OFDM/OQAM has over traditional OFDM in terms of spectral compactness and efficiency. The consequence of applying a band-limiting pulse to the subcarriers of a multichannel communications scheme are highlighted in Chapter 4. This chapter has also introduced a novel approximate square-root Nyquist filter using the Kaiser-Bessel window. The resulting filter has demonstrated significant performance improvements over the traditional square-root raised-cosine filter design in terms of stop-band rejection and inter-symbol interference. Furthermore, the filter's characteristics are very similar to the best known hM-3 filter but do not require iterations over the Parks-McClellan algorithm which can be sensitive to finite machine precision. The proposed r-Kaiser filter is used to demonstrate the superior spectral compactness of OFDM/OQAM over traditional OFDM.

Chapter 3

Synchronization of Narrowband Channels

In synchronous narrowband digital communications systems data are conveyed over a wireless link by transmitting uniformly-spaced symbols, filtered with a band-limited pulse. The signal at the receiver is completely known except for the data symbols and several unknown *reference parameters* [67]. While these reference parameters carry no inherent information, the ultimate task of the receiver is to recover the data symbols; without sufficient knowledge of these reference parameters, the receiver's detection process to recover the data will ultimately fail.

Synchronous data communications systems on SDR platforms rely on DSP algorithms to correct for symbol timing and carrier phase impairments observable at the receiver after the signal has been converted into a discrete sampled and quantized time series. The computational complexity of such algorithms often comprises the majority of the processing load burdening the baseband receiver. Before going into great detail in synchronization of multi-carrier systems such as OFDM/OQAM, it is necessary to first discuss synchronization of narrowband signals. This chapter gives a brief overview of synchronization techniques for narrowband signals and provides several original estimators for reference parameters, as well as preamble patterns, and general design strategies for quickly synchronizing and decoding frames transmitted in in short bursts for the sake of dynamic spectrum access. A new maximum likelihood-based estimator for the timing offset parameter using the output of the matched filter and its own derivative is provided. An enhanced dual loop architecture and framing preamble are introduced to enable timing and carrier synchronization of single-carrier signals with a minimal number of initial symbols. Additionally provided is a novel framing structure is described which permits fast acquisition and automatic reconfiguration at the receiver. The efficacy of the proposed system is supported with baseband simulation and validated with over-the-air reception in a multi-path wireless laboratory environment.

3.1 System Description

This work refers to *narrowband* signals as single-carrier systems occupying a bandwidth commensurate of its symbol rate; compare this to multi-carrier systems in which the occupied bandwidth is split into many narrow subcarriers—the symbol rate on each being relatively small. This chapter will only focus on the recovery of linear modulation transmissions and for the sake of brevity ignore continuous-phase modulations (e.g. minimum-shift keying) which typically exhibit a lower spectral efficiency.

Let the transmitted analog signal $s(t)$ be the sum of a discrete set of complex-valued symbols $a(p)$ filtered by a pulse shape $g(t)$ spanning $2L$ symbols, each separated by a unit time T, viz.

$$s(t) = \sum_{p=-\infty}^{\infty} a(p)g(t - pT) \tag{3.1}$$

The received signal is assumed to be a delayed version of $s(t)$ corrupted by carrier offsets, multi-path fading, and noise, viz.

$$r(t) = e^{j(\nu t+\theta)}\gamma s(t - \tau) * h_c(t) + n(t) \tag{3.2}$$

where ν is the carrier frequency offset, θ is the carrier phase offset, γ is the channel gain due to path loss, τ is the sample timing phase offset, $h_c(t)$ is the the impulse response of the channel (ignored for now), and $n(t)$ is the baseband noise term with a two-sided power-spectral density N_0. In order for the data symbols $a(p)$ to be recovered without error, it is necessary to remove the effects of the channel by estimating the reference parameters imparted by the channel.[1]

This chapter introduces new estimators for many of these reference parameters and provide detailed analysis on their performance under different assumptions about the channel characteristics (e.g. no fading, minimal carrier frequency offset, etc.) Many of these techniques will then be applied to multi-carrier synchronization techniques in Chapter 4. The design of the estimators focuses specifically on computationally simplicity and scalability at the receiver. The final section of this chapter describes a flexible frame descriptor which is capable of fast synchronization and computational scalability. The performance of the framing structure is tested under a number of typical operating conditions in wireless environments.

3.1.1 Synchronization and Estimation

Commonly-cited literature suggests that synchronization techniques are typically derived from two sets of approaches: heuristically-based [25, Section III], [71] and maximum likelihood-based [25, Section IV], [67] estimators. Because heuristic approaches cannot be derived in any structured manner, this work will focus efforts primarily on those based on maximum likelihood estimation.

[1]Realistically, carrier offsets are a result of mistuning the receiver's oscillators to the transmitter's and are not a result of channel distortion unless significant Doppler speeds are involved. The same could be said for the timing offset which is due in part to the transmission delay time, but also to sampling clock phase of the analog-to-digital converter. For simplicity's sake, however, this work refers to all the reference parameters as impairments which have been introduced by channel effects for which the receiver must correct digitally in software.

3.1.2 Maximum Likelihood Estimation

Maximum likelihood (ML) estimation is a method by which a statistical model is fit to an observed data set, providing estimators for the model's parameters. For a particular data set observation, the maximum likelihood method chooses the parameter set which results in a distribution that most likely resulted from the data. Mengali and D'Andrea give an excellent description of the derivation behind maximum likelihood estimation in [67, Section 2.3.1] with an emphasis on synchronization of digitally modulated signals. To begin, the expression in (3.2) can be rewritten generically as

$$r(t) = s(t, \boldsymbol{\gamma}) + n(t) \tag{3.3}$$

where $\boldsymbol{\gamma}$ represents a vector of the unknown reference parameters. The generalized likelihood function for the continuous-time domain for an unknown parameter set $\boldsymbol{\gamma}$ over a duration of T_0 seconds is given by [67, (2.3.33)]

$$\Lambda(\boldsymbol{r}|\tilde{\boldsymbol{\gamma}}) = \exp\left\{ \frac{2}{N_0} \int_0^{T_0} r(t) s(t, \tilde{\boldsymbol{\gamma}}) dt - \frac{1}{N_0} \int_0^{T_0} s^2(t, \tilde{\boldsymbol{\gamma}}) dt \right\} \tag{3.4}$$

where $\tilde{\boldsymbol{\gamma}}$ is the trial set of unknown parameters, and N_0 is the noise power spectral density of $n(t)$ in (3.3). The maximum likelihood estimate of the unknown parameters (e.g. ν, θ, τ, and $\boldsymbol{a}$) are those which maximize $\Lambda(\boldsymbol{r}|\tilde{\boldsymbol{\gamma}})$, or equivalently $\ln\{\Lambda(\boldsymbol{r}|\tilde{\boldsymbol{\gamma}})\}$

Turning to software-defined radios, the full extent of baseband processing is carried out in the digital domain. As such the received signal can be assumed to have been passed through a linear-phase rectangular (ideal) low-pass image-rejection filter and is then sampled at a rate of $1/T_s$. The unknown parameter set $\boldsymbol{\gamma}$ are then estimated on the resulting sample sequence alone. The generalized likelihood function for a sequence $x(t)$ sampled at a a rate $1/T_s$ over a period of L_0 samples is given by [67, (2.3.71)]

$$\Lambda(\boldsymbol{x}|\tilde{\boldsymbol{\gamma}}) = \exp\left\{ \frac{2T_s}{N_0} \sum_{k=0}^{L_0-1} x(kT_s) s(kT_s, \tilde{\boldsymbol{\gamma}}) - \frac{T_s}{N_0} \sum_{k=0}^{L_0-1} s^2(kT_s, \tilde{\boldsymbol{\gamma}}) \right\} \tag{3.5}$$

where $\boldsymbol{x} = \{x_0, x_1, \ldots, x_{L_0-1}\}$ is the sampled sequence. For the extent of the analysis in this document, the received signal sequence has been passed through a Hilbert transform [76, p. 150] after sampling and is relatively near complex baseband (i.e. $\nu \ll 1/T$). Furthermore, the underlying information symbols in (3.1) are independent and identically distributed. Section 3.2.6, however, demonstrates that choosing correlated symbols can significantly reduce the estimator's variance and provide a fast acquisition preamble for synchronizing to short frames.

3.2 Timing Recovery

Timing recovery is the process of aligning the receiver's sampling clock to that of the transmitter. When the receiver's sampling clock is misaligned to the transmitter's, inter-symbol interference occurs at the matched-filter output and appears as an increased noise variance in the resulting signal constellation. So long as the carrier offset is not a significant portion of the symbol rate, timing recovery can be achieved asynchronously to the carrier. Many carrier phase recovery techniques for linear modulations first require properly time-sampled symbols as the phase is the information-bearing component of the signal. As such, the analysis for single-carrier synchronization begins with timing recovery through estimation and tracking of τ.

Physical layer synchronization of symbol timing is required when samples of the received signal are misaligned with the data symbols generated by the transmitter. Several options to align the matched filter output samples are available including over-sampling the received signal and interpolating using using piecewise polynomials or polyphase filterbanks. Baseband filtering, however, consumes a significant portion of processing, and, as a result, oversampling the received signal in excess of the Nyquist rate is undesirable, considering that much of the work expended through the filtering operation results in output samples that are to be discarded. Alternatively, interpolating between available sample points has proven to be computationally efficient. Lagrange interpolating polynomials minimize inter-symbol interference by curve-fitting the past several samples to an n^{th}-order polynomial [67, Section 7.3.2]. The output symbol may be appropriately computed by simply evaluating the polynomial at the proper timing instance. The downside to using Lagrange polynomials in this fashion is that the polynomial coefficients must be re-computed for each new symbol output. Furthermore, the interpolator is applied after the down-sampler and as a result cannot completely remove inter-symbol interference.

Rice and harris proposed the use of polyphase filter banks for symbol synchronization in digital receivers in [30] and developed loop control architectures for choosing the optimal filter in the bank in addition to input sample flow control. The use of a polyphase filterbank as an effective phase shifter in the sampling clock allows for greater flexibility and efficiency in the receiver by computing only those multiplications necessary for matched filtering while simultaneously interpolating to achieve a sample point sufficiently close to the optimum. Missing from the discussion in [30], however, was an integral discussion on control loop filter coefficients; the application of the matched filter within the loop introduces a constant delay between the timing error detector and its control over the input sample flow. This additional delay creates potential instability within the loop and can corrupt the output signal, causing errors in the detector. By designing the filter coefficients appropriately, stability with in the control loop is maintained. This section expands on the work demonstrated in [30] by extending the loop control design to include stability analysis of the timing recovery system under a number of signal types (excess bandwidth factor, phased vs. random data, etc.). Furthermore, the input control flow algorithm is given explicitly. Analysis is verified by baseband simulations in Section 3.2.5. The efficacy of the timing recovery structure is later validated via over-the-air reception in Section 3.5.

3.2.1 Multi-rate Synchronization Theory

This section provides an abbreviated overview of timing synchronization through interpolation in digital-sample receivers. For a more thorough investigation on the topic of linear interpolation techniques in digital receivers the interested reader may refer to [40].

Continuous and Discrete Time Representations

Let the transmitted analog signal $s(t)$ be the sum of M-ary complex-valued symbols filtered by a pulse shape $g(t)$ spanning $2L$ symbols, each separated by a unit time T, as given by (3.1). The received signal is assumed to be a delayed version of $s(t)$ corrupted by the addition of white Gaussian noise, $n(t)$, viz.

$$r(t) = s(t - \tau) + n(t) \tag{3.6}$$

Notice that $r(t)$ here is the same as in (3.2) with $\nu = 0$, $\theta = 0$, and $h_c(t) = \delta(t)$. For the sake of simplicity in this discussion, all other reference parameters—including carrier offsets and multipath fading—will be ignored. The optimum receiver uses the output $y(t)$ of a matched filter whose impulse response is $h(t) = g(-t)$.

As is the practice for software-defined receivers, $r(t)$ is sampled using an analog-to-digital converter (ADC) before filtering in the discrete-time domain. The discrete version of the received signal at a uniform sampling interval T_s is $r(nTs)$ where the ratio of symbol period to sampling period T/T_s is assumed to be irrational, but often nearly integral. This will hold true for all communications links whose transceiver hardware ADCs are driven by independent clock sources; this is the most common situation for modern wireless transceivers which generally avoid transmitting clock information on a separate channel. Consequently, the receiver must recover the transmitter's sampling clock by recovering such information from the received signal alone. The received signal samples are applied to an interpolator that computes interpolants $y(kT_i)$ at a reduced rate $T_i = T/N$. The synchronizer must adjust the interval T_i to match the rate at which data symbols are generated by the transmitter. The new samples for an interpolating filter with an impulse response $h_I(t)$ over the time span $I_1 \leq i \leq I_2$ are given by [40, (6)],

$$y(kT_i) = \sum_{i=I_1}^{I_2} r\left[(m_k - i)T_s\right] h_I\left[(i + \mu_k)T_s\right] \tag{3.7}$$

where $m_k = \lfloor kT_i/T_s \rfloor$ is the base-point index and $\mu_k = kT_i/T_s - m_k$ is the irrational fractional interval [40]. Because T_i/T_s is likely irrational, μ_k will change for each interpolant and can take on an infinite number of possible values. Figure 3.1 demonstrates the relationship between optimum and available sampling points. In this example the sample rate T_s is approximately twice the symbol rate; however, the position of optimum timing slides to the right of the available sample points as time progresses indicating that the actual sampling frequency is slightly greater than 2 samples/symbol.

Polyphase Filterbanks as a Mechanism for Resampling

By using a polyphase decomposition of a sampled version of the matched filter, a polyphase filterbank can be used to calculate the interpolants in (3.7). The discrete impulse response of the m^{th} filter in a bank of M filters is given by [30, Eq. (15)]

$$h_m\left(nT_s\right) = h\left(nT_s + \frac{m}{M}T_s\right) \tag{3.8}$$

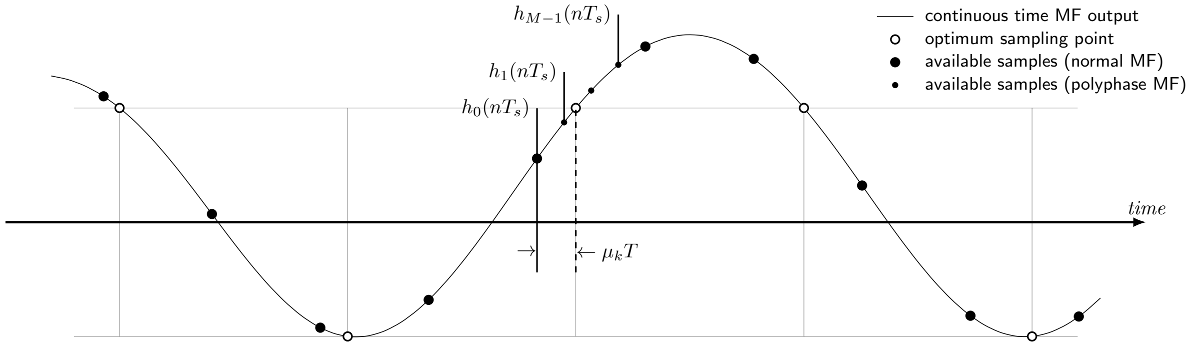


Figure 3.1: Matched filter output $y(t)$ and the relationship between available sample points, optimum sample points, and interpolants.

The goal of the synchronizer is to choose the filterbank at index m such that the fractional portion mT_s/M is as close to the true timing offset μ_k as possible. As demonstrated in Figure 3.1, none of the interpolants from the filterbank lie directly on the optimum output. Increasing the filterbank size can ensure a sufficiently fine resolution and keep the inter-symbol interference due to any residual timing offset negligible. Note that in the example in Figure 3.1 the bank consists of $M = 4$ filters. While none of the available samples lie directly on the optimum sampling point, the resolution can be set sufficiently small by increasing M.

The benefit of using filterbanks is apparent; each of the M filters in the bank operate at a minimal over-sampling rate which reduces the computational load of the receiver. This holds true regardless of the number of filters in the bank which serve only to increase the resolution of the output interpolant at the expense of increased memory size. The filterbank described above is a special case in that the bank consists of M coefficient arrays, but only one input buffer. The input samples are loaded serially into the buffer and then at the appropriate time, the filter coefficients which closely match the ideal interpolator are chosen. In contrast the filterbank channelizer described in Section 2.3 uses a bank of M coefficients but also a bank of M input buffers.

3.2.2 Loop Architecture

The loop control serves to track small variations in the timing offset between the transmit and receive sampling clocks. A timing phase-locked loop controls both the base-point sampling index and the optimum filter in the bank to calculate the interpolant. The timing tracking loop therefore operates quite differently from standard carrier PLLs; unlike carrier PLL systems which operate one sample out for each sample in, the timing PLL needs an additional mechanism to control the input and output flow of data samples. If the sample rate between the transmit and receive clocks is mismatched, μ_k will consistently change; more specifically, μ_k will increase with each sample if the sampling rate of the receiver exceeds that of the transmitter, stay constant if they are equal, and decrease otherwise. Eventually the base-point index will need to be changed as the fractional sample interval μ_k wraps around $[0, 1)$. In such a case the time-span $I_1 \leq i \leq I_2$ in (3.7) needs to be adjusted, effectively governing the input sample flow control to the matched-filter bank.

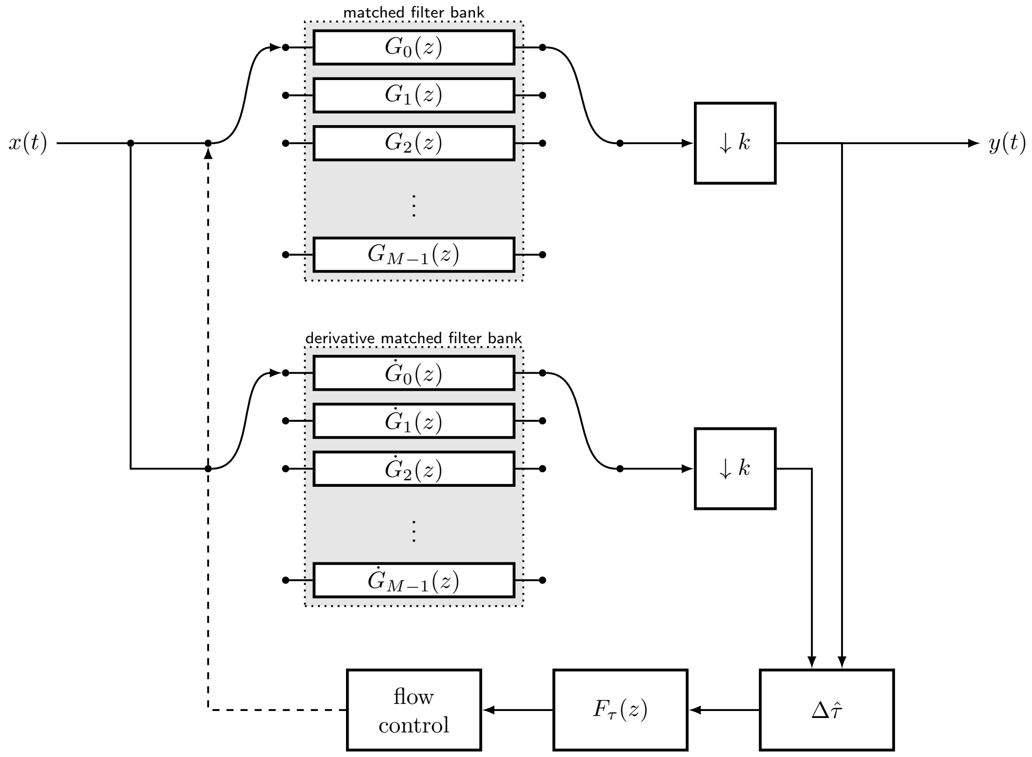


Figure 3.2: Symbol synchronizer block diagram with the control loop operating at one sample per symbol.

harris and Rice describe three possible loop control architectures in [30], the main differences being the rate at which the loop operates. Keeping the goal of minimizing processing complexity in mind, this work uses the architecture operating at one sample/symbol. This not only reduces the number of computational instructions, but uses only the optimal samples for timing error detection. The timing loop consists of a dual pair of matched filter (MF) and derivative matched filter (dMF) polyphase filterbanks which produce an output once every symbol period. The resulting outputs of the MF and dMF are fed into a timing error generator, discussed below. The error signal is filtered with a second-order low-pass timing loop filter $F_\tau(z)$ before feeding the filterbank control. Auxiliary control via the "shift/skip/stuff" mechanism described in [81] is provided to accommodate the incommensurate relationship between the data and sample clocks. The adopted system block diagram can be seen in Figure 3.2 which consists of a bank of M matched filters, a bank of M derivative matched filters, a timing error detector, and a loop filter with flow control.

3.2.3 Timing Error Generator

This system bases its timing error generator on the maximum likelihood timing error detector [67] which relies on the derivative to the matched filter impulse response,

$$\dot{h}(t) = \frac{\partial h(t)}{\partial t} \tag{3.9}$$

The ML estimator, therefore, relies on sampling the derivative of the matched filter impulse response,

$$\dot{h}_m(z) = \frac{\partial h_m(z)}{\partial z}, \quad 0 \leq m \leq M-1 \tag{3.10}$$

Computing the true derivative of the continuous-time filter impulse response is difficult as many designs do not have a closed-form solution. One particular example is the harris-Moerder filter [29] which computes its coefficients by iterating over the Parks-McClellan algorithm, deriving its impulse response from an inverse discrete Fourier transform, and has no closed-form expression. Section 2.5 has provided a filter design which does have a closed-form impulse response; however, the need for deriving such an expression can be circumvented by using an approximation method applicable to any filter. Mengali provides an efficient means for approximating $\dot{h}_m(z)$ from the MF coefficients [67, (7.4.57)], also used by [30], viz.

$$\dot{h}_m(z) \approx \begin{cases} h_1(z) - h_{M-1}(z) & m = 0 \\ h_{m+1}(z) - h_{m-1}(z) & 0 < m < M-1 \\ h_{M-2}(z) - h_0(z) & m = M-1 \end{cases} \tag{3.11}$$

This structure allows for only two polyphase filterbanks, and can additionally operate at a minimum sampling rate, e.g. $k = 2$.

The error averaged over many symbols provides feedback for controlling the filterbank index. As demonstrated by the S-curve for the maximum likelihood timing error detector depicted in [67, Figure 7.13], early sampling results in a negative error, while late sampling results in a positive error. When the filtered error signal is negative, the control loop will choose successively lower filterbank indices, enabling less delay. When the filterbank index pointer underflows, the input and output clocks are aligned by repeating the last input sample. Conversely, a positive error will increase the filterbank index pointer, enabling successively more delay. When the filterbank index pointer overflows, the next input sample is skipped.

Each symbol's matched-filter output $y(k)$ and derivative matched-filter output $\dot{y}(k)$ are combined to provide an instantaneous timing error estimate. This error is pushed through an integrating loop filter which drives the filterbank index selection and input sample control. The instantaneous timing error is derived from the maximum likelihood timing offset estimate [67, (8.3.5)]

$$\Delta\tilde{\tau} = \Re\big\{y^*(k)\dot{y}(k)\big\} \tag{3.12}$$

Normalizing to the signal's power level limits the proportion of the timing error estimate to the symbol's signal-to-noise ratio [30, (5)], viz.

$$\Delta\hat{\tau} = \tanh\Big(\Re\big\{y^*(k)\dot{y}(k)\big\}\Big) \tag{3.13}$$

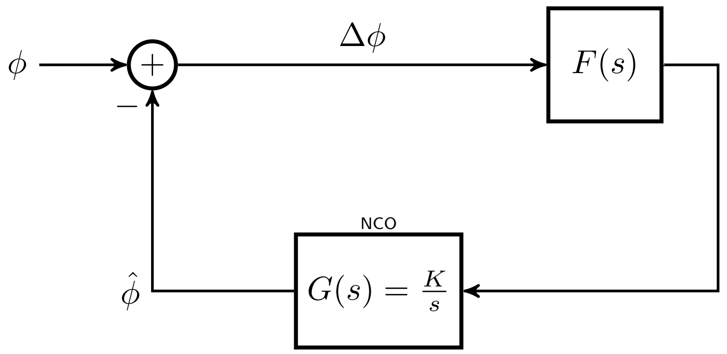


Figure 3.3: Analog phase-locked loop block diagram

A loop filter integrates over $\Delta\hat{\tau}$ to give a timing offset estimate $\hat{\tau}$ which is used to produce the next output symbol. The resulting filterbank index chosen for the next output symbol is $[\hat{\tau}M]$ where $[\cdot]$ denotes the rounding operation.

3.2.4 Timing Loop Filter, Delay Considerations, and Stability Analysis

The discrete-time phase-locked loop (DT-PLL) is a key component in software radio, particularly for timing and carrier offset recovery tracking loops. The PLL consists of three components: the phase detector, the loop filter, and the integrator. A block diagram of the typical analog PLL can be seen in Figure 3.3 in which the phase detector is represented by the summing node, the loop filter is $F(s)$, and the integrator has a transfer function $G(s) = K/s$. For a given loop filter $F(s)$, the closed-loop transfer function for the phase becomes

$$H(s) = \frac{G(s)F(s)}{1 + G(s)F(s)} = \frac{KF(s)}{s + KF(s)} \tag{3.14}$$

where the loop gain K absorbs all the gains in the loop.

There are several well-known options for designing the loop filter $F(s)$, which is, in general, a first-order low-pass filter. In particular it is convenient to express $H(s)$ such that its denominator is in standard form $s^2 + 2\zeta\omega_n s + \omega_n^2$ where ω_n is the natural frequency of the filter and ζ is the damping factor. This simplifies analysis of the overall transfer function and allows the parameters of $F(s)$ to ensure stability. The active lag design has a loop filter transfer function [9]

$$F(s) = \frac{1 + \tau_2 s}{1 + \tau_1 s} \tag{3.15}$$

When added to an integrator K/s the closed-loop transfer function becomes [76, (6.2-18)]

$$H(s) = \frac{KF(s)}{s + KF(s)} = \frac{\frac{K}{\tau_1}(1 + s\tau_2)}{s^2 + s\frac{1+K\tau_2}{\tau_1} + \frac{K}{\tau_1}} \tag{3.16}$$

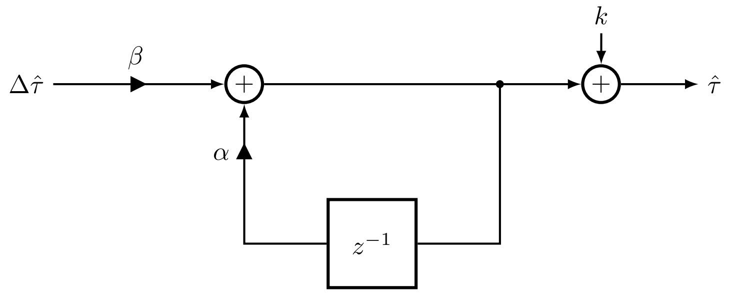


Figure 3.4: Symbol synchronizer timing loop diagram

which results in the parameter transformations

$$\omega_n = \sqrt{\frac{K}{\tau_1}}, \quad \zeta = \frac{\omega_n}{2}\left(\tau_2 + \frac{1}{K}\right) \rightarrow \tau_1 = \frac{K}{\omega_n^2}, \quad \tau_2 = \frac{2\zeta}{\omega_n} - \frac{1}{K} \tag{3.17}$$

Conversion of $H(s)$ to its discrete form can be approximated by taking the bilinear z-transform such that $s \rightarrow 2(1-z^{-1})/T(1+z^{-1})$ [100, p. 248] which results in

$$H'(z) = H'(s)\Big|_{s=\frac{1}{2}\frac{1-z^{-1}}{1+z^{-1}}} = 2K\frac{(1+\tau_2/2) + 2z^{-1} + (1-\tau_2/2)z^{-2}}{(1+\tau_1/2) - \tau_1 z^{-1} + (-1+\tau_1/2)z^{-2}} \tag{3.18}$$

As demonstrated in [7, 8], the introduction of delay inside the control loop leads to instability in the discrete-time phase-locked loop under certain considerations for the natural frequency and damping factor. Additionally, [7, 42] demonstrate that the stability region of the PLL diminishes quickly as the delay increases to only a few samples. To maintain stability the loop's natural frequency is consequently decreased.

The feedback loop in Figure 3.2 has a nominal delay of k samples due to the matched filter $G(z)$. The timing control loop can be seen in Figure 3.4. The filter parameters for a control loop bandwidth of ω_τ are empirically given as $\alpha = 1 - \omega_\tau/2\pi$ and $\beta = 0.22\omega_\tau/2\pi$ which yield a stable loop filter for a minimal over-sampling rate $k = 2$.

Algorithm 1 Symbol synchronizer loop control

```
1: $\hat{q} \leftarrow 0$   (filtered timing error)
2: $\hat{\tau} \leftarrow 0$   (timing offset estimate)
3: $b \leftarrow 0$   (filterbank index)
4: $j \leftarrow 0$   (output symbol counter)
5: for $i = 0$ to $N_i - 1$ do
6:     push $x(i)$ into MF, dMF
7:     while $b < M$ do
8:         $y(j) \leftarrow$ MF output using $G_b(z)$
9:         $\dot{y}(j) \leftarrow$ dMF output at index using $\dot{G}_b(z)$
10:        $q \leftarrow \tanh\Big(\Re\big\{y^*(k)\dot{y}(k)\big\}\Big)$
11:        $\hat{q} \leftarrow q\beta + \hat{q}\alpha$
12:        $\hat{\tau} \leftarrow \hat{\tau} + k + \hat{q}$
13:        $b \leftarrow [\hat{\tau}M]$
14:        $j \leftarrow j + 1$
15:    end while
16:    $\hat{\tau} \leftarrow \hat{\tau} - 1$
17:    $b \leftarrow b - M$
18: end for
```

The full control loop is described in Algorithm 1. Notice that an output is only computed when the filterbank index rolls over M.

3.2.5 Performance Results

For all simulations the received signal is generated at $k = 2$ samples/symbol with a nominal signal-to-noise ratio of 30dB. The timing loop filter operates with a noise bandwidth of approximately 2% of the symbol rate. A polyphase filterbank with $M = 32$ stages provides matched filtering. The timing error generator is given by (3.13).

Symbol Timing Phase Step Response

The first simulation demonstrates the response of the synchronizer when a timing step offset of $\tau = -0.3T$ is introduced into the system with a nominal SNR of 30dB. The results of the symbol timing recovery loop with $k = 2$ samples per symbol and $M = 32$ polyphase filters can be found in Figure 3.5. Because the timing error is negative, the control slides forward through the filters in the bank and eventually settles on 19 after about 400 symbols. The ideal filterbank index is $m = -\tau k M = 19.2$ which is not an integer and as a result the loop settles on index 19, but sometimes jumps up to 20. Consequently, the residual error is kept relatively small with sufficient M, as can be seen by the tight constellation in Figure 3.5.

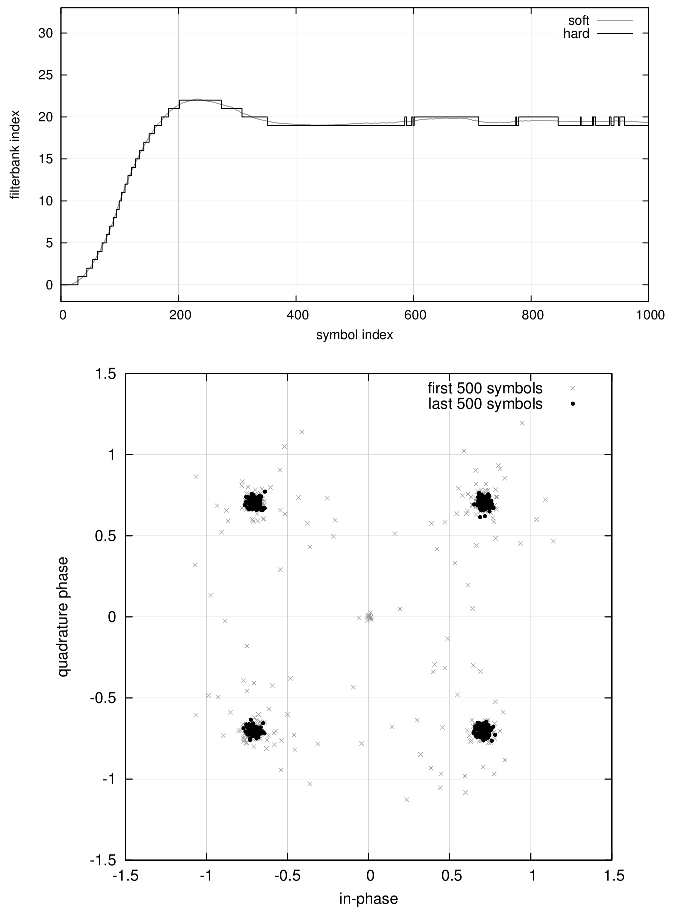


Figure 3.5: Symbol synchronizer performance with a timing phase offset of $\tau = -0.3T$ and $M = 32$ polyphase filters.

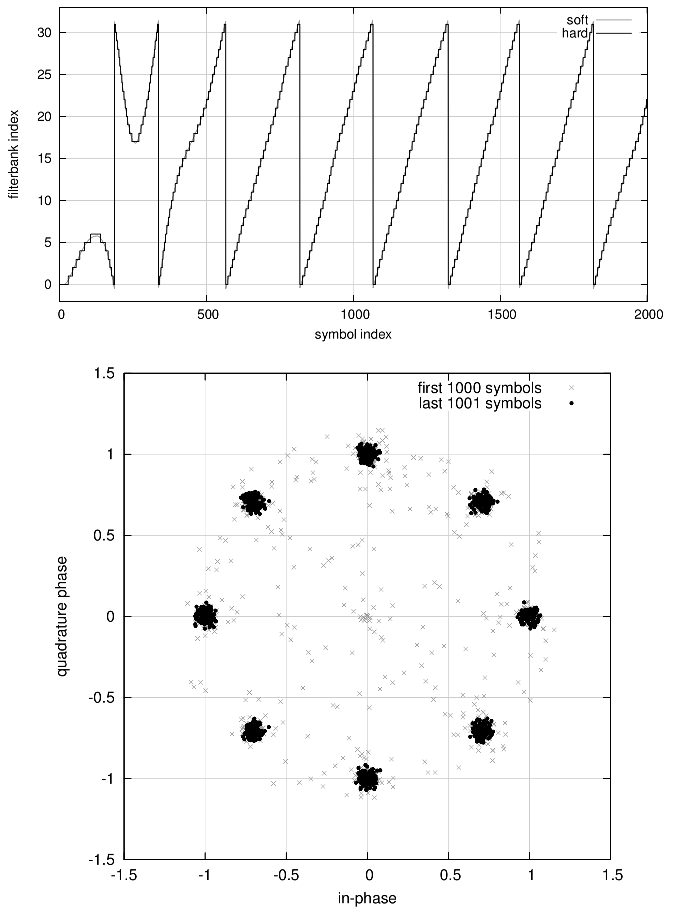


Figure 3.6: Symbol synchronizer performance with a timing frequency offset of 0.4% of the symbol rate and $M = 32$ polyphase filters.

Symbol Timing Frequency Step Response

The second simulation introduces a timing frequency offset by which the receive sample clock is offset from the transmit clock by $T/250$ (0.4% of the symbol rate). As a result, the optimum sampling instant (the fractional interval) appears to slide through the available samples. The results can be found in Figure 3.6 in which the synchronizer eventually sweeps through a full period every 250 symbols.

3.2.6 Maximum Likelihood-based Estimators

The previous section has described a symbol timing recovery algorithm which uses a pair of polyphase filterbanks to compute the matched-filter output interpolant and track to any residual timing offset. Sometimes, however, it is necessary to make an initial estimate of the timing offset given a number of input samples rather than relying on tracking loops. This is particularly convenient for the acquisition of framing headers for the transmission of packets in short bursts. Going back to ML estimation, the log-likelihood function for $\tilde{\tau}$ is

$$L(\boldsymbol{r}|\tilde{\tau}) = \frac{2}{N_0}\int_0^{T_0} r(t)\tilde{s}(t)dt - \frac{1}{N_0}\int_0^{T_0} \tilde{s}^2(t) \tag{3.19}$$

where $\tilde{s}(t)$ is the trial signal, defined as

$$\tilde{s}(t) \triangleq \sum_i \hat{a}_i g(t - iT - \tilde{\tau}) \tag{3.20}$$

After some mathematical manipulations and approximating assumptions, the derivative of the log-likelihood function is

$$\frac{\partial}{\partial\tilde{\tau}} L(\boldsymbol{r}|\tilde{\boldsymbol{\gamma}}) = \sum_{i=0}^{L_0-1} \hat{a}_i \Big[\dot{y}(iT+\tilde{\tau}) - \sum_{m=-\infty}^{\infty} \hat{a}_i h'(i-m)T\Big] \tag{3.21}$$

where $\dot{y}(t)$ is the response to the received signal $r(t)$ to the derivative matched filter $-\dot{g}(-t)$, $h'(t) = g(t) * \dot{g}(-t)$ is the response of the matched filter to its own derivative, and L_0 is the observation time in symbols. The response of $g(t)$ to $\dot{g}(-t)$ can be found in Figure 3.7 for the square root-raised cosine filter and the proposed r-Kaiser filter (Section 2.5) for an excess bandwidth factor of $\beta = 0.7$. It is useful to observe some qualities of $h'(t)$:

1. $h'(t)$ crosses through zero at $t = 0$ which implies that the Nyquist filter $h(t)$ has a maximum response at the origin. This is an expected and is verified in Figure 2.9(a);

2. $h'(t)$ has odd symmetry, e.g. $h'(-t) = h'(t)$.

Notice the familiar S-curve for $-T \leq t \leq T$, and how similar its response is to $\sin(\pi t/T)$. This suggests that a timing offset τ will present itself in the S-curve as a simple phase offset as $\sin(\pi t/T + \theta)$. The need for deriving the estimates for the transmitted symbols $\hat{a}_i$ in (3.20) can be circumvented by assuming that the expression given by (3.21) is dominated by the inner summation

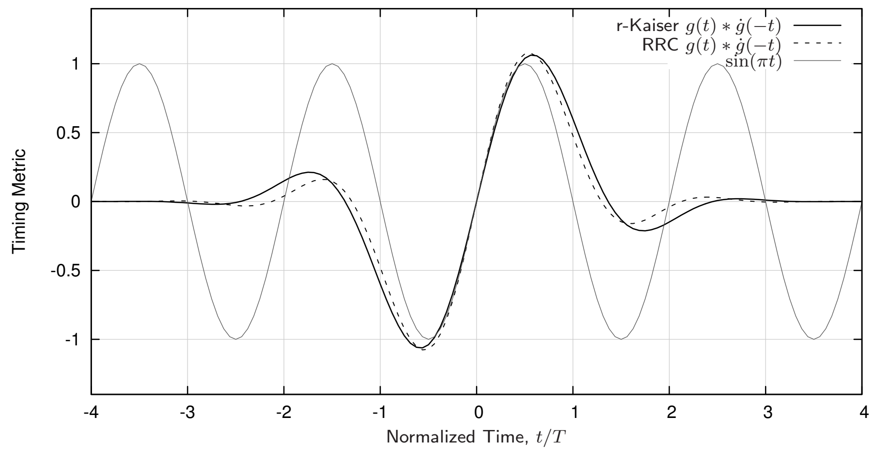


Figure 3.7: Matched-filter/derivative matched-filter output, $g(t) * \dot{g}(-t)$, for an excess bandwidth factor $\beta = 0.7$

over $h'((i-m)T)$. Furthermore, this summation can be truncated for only a few symbols as the response of $h'(t)$ decays quickly for just a few symbols, as is clearly visible in Figure 3.7. Therefore averaging the output of the received signal to its matched- and derivative matched-filter product yields a sinusoid whose period is $2T$ and phase is proportional to τ. Furthermore, the maximum likelihood estimator for the phase of a sinusoid with a known frequency f_c is well-known as

$$\hat{\theta} = \arg\left\{ \sum_k x(kT_s) e^{-j2\pi f_c k} \right\} \tag{3.22}$$

Therefore, a new approximate maximum likelihood timing offset estimator based on (3.21) and (3.22) may be expressed as

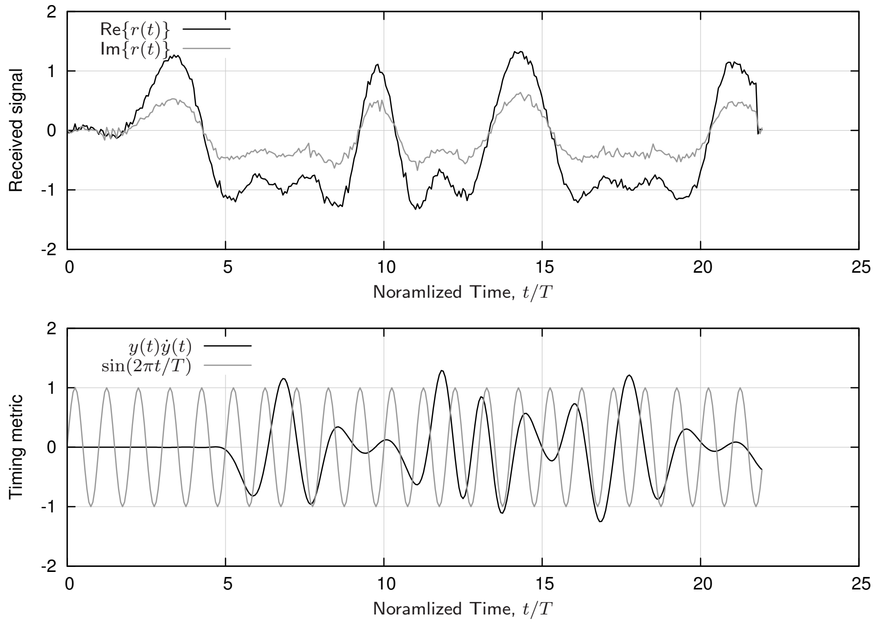
$$\hat{\tau}_G = -\frac{T}{2\pi} \arg\left\{ \sum_{i=0}^{NL_0-1} y(iT_s)\dot{y}(iT_s) e^{j(2\pi i/N + \pi/2)} \right\} \tag{3.23}$$

where the additional $\pi/2$ term in the exponent is necessary because $h'(t)$ has odd symmetry. Upon further observation, (3.23) is remarkably similar to the well-known Oerder & Meyr estimator [67]

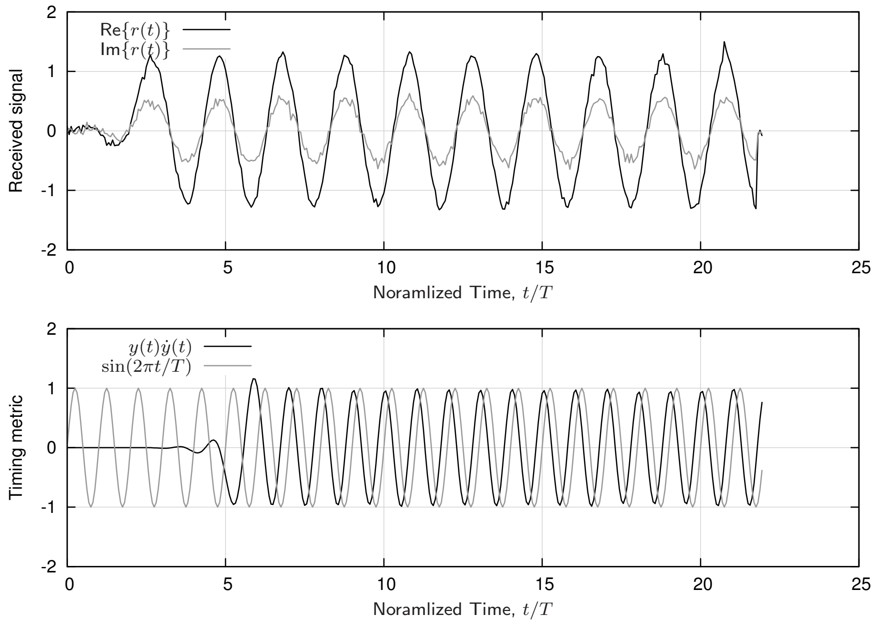
$$\hat{\tau}_{OM} = -\frac{T}{2\pi} \arg\left\{ \sum_{i=0}^{NL_0-1} y^2(iT_s) e^{j(2\pi i/N)} \right\} \tag{3.24}$$

which relies solely on the matched-filter output alone. Both of these estimators are based on the cyclostationary properties of $s(t)$—each involves computing $s^2(t)$ in some form—and exploit the energy carried within the transition bandwidth of $g(t)$.

Figure 3.8 depicts the timing metric for a BPSK signal with random carrier phase and a nominal SNR of 30dB. In the top pair of figures, the underlying data symbols are random and the timing metric doesn't appear to align well with the sinusoid. In the second figure, the data are antipodally



(a) Random Data



(b) Phased Data

Figure 3.8: Timing synchronizer metric for a BPSK signal with a timing offset of $\tau = 0.2T$

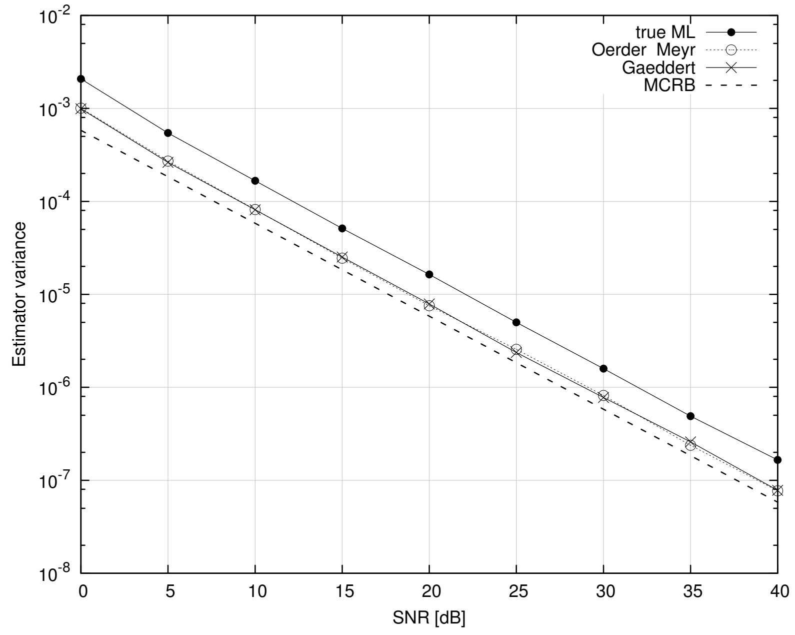


Figure 3.9: Timing offset estimator variance vs. SNR for a root raised-cosine filter with $k = 4$ samples per symbol and an excess bandwidth factor of $\beta = 0.5$ averaged over 2,000 trials.

phased as to maximize the number of transitions between successive symbols. Here, the alignment with the sinusoid is nearly perfect with only a delay, due to the timing offset of $\tau = 0.2T$.

Figure 3.9 demonstrates the performance of the proposed estimator in a noisy channel, as well as the Oerder & Meyr estimator given by (3.24), the true maximum likelihood estimator and the modified Cramér-Rao bound (MCRB) given in [67]. The variance of the estimator error takes the form

$$\sigma^2 = \frac{1}{L_0}\Big[K_{ss} + K_{sn}/\gamma + K_{nn}/\gamma^2\Big]$$

where γ is signal-to-noise ratio (linear), K_{ss} is the signal auto-correlation coefficient, K_{sn} is the signal/noise cross-correlation coefficient, and K_{nn} is the noise auto-correlation coefficient. The performance of $\hat{\tau}_G$ is nearly identical to that of $\hat{\tau}_{OM}$, while out-performing the true maximum likelihood estimator and nearly achieving the minimum theoretical error variance. One of the benefits of this estimator is that it can be computed completely blind of the transmitting symbols c_k and carrier offsets θ and ν (to within reason) while being nearly efficient. The timing metric, however, has periodicity $2T$ which requires an over-sampling factor of $k = 4$ which is undesirable for software-defined radios where minimizing computational complexity is a key factor in design.

3.3 Carrier Recovery

Carrier phase recovery with linear modulations is usually accomplished after the down-sampler once timing has been properly recovered. This allows the decision-directed phase offset estimate after demodulation to be used to drive the phase-locked loop. Phase recovery algorithms with incoherent timing offsets do exist, but are typically limited to PSK modulation and have relatively poor performance, particularly when signal strength is low. As an example, the well-known Costas loop removes the data modulation of a BPSK-modulated signal by squaring it (or using higher exponents for higher PSK modulation indices) [72]. The problem with this technique is that the error introduced when the signal crosses through or near zero (typical with linearly-modulated signals) is amplified and injected into the control loop. With timing-coherent carrier recovery only the phase error taken at the maximum signal output is used to drive the loop.

Carrier recovery in SDR operates at the symbol rate of the system and therefore requires a high amount of processing relative to the data rate of the system. In fact the numerically-controlled oscillator computes one complex rotation per modulated symbol, and as a result the computational bandwidth to synchronize the carrier of the receiver is quite high. It is therefore necessary to observe the resource consumption of carrier synchronization in software-defined receivers. This section gives a brief theoretical background on carrier recovery with an emphasis on computational complexity in software-defined radios.

3.3.1 Phase Error Estimation

Assuming coherent timing, the phase error estimate for a linear modulation scheme is [67, (8.2.6)]

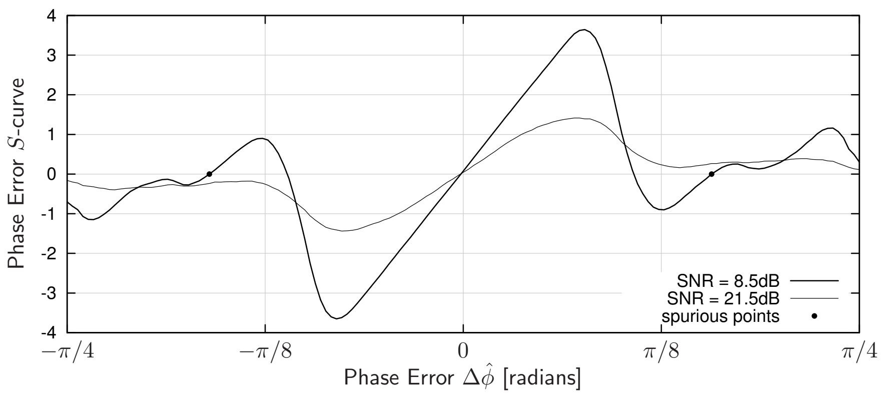
$$\Delta\hat{\phi}_k = \Im\Big\{\hat{a}_k^* y(k)\Big\} \tag{3.25}$$

where $y(k)$ is the received symbol and $\hat{a}_k$ is the output of the detector at index k. Notice that unlike timing recovery, the phase error estimate requires an estimate of the transmitted symbol. As before it is assumed at this point that $E\big\{a_k^2\big\} = 1$ and that gain control has normalized the received signal level such that $E\big\{|r(t)|^2\big\} = 1$. With transmitted symbols a_k of a smaller magnitude, the product in (3.25) is reduced, which reduces noise introduced into the loop by giving more emphasis to symbols with higher energy.

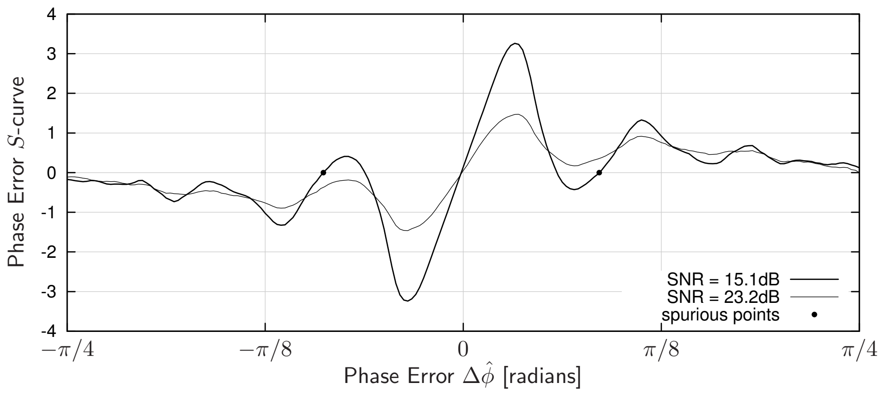
3.3.2 Linear Modulation S-Curves

Clearly the shape of (3.25) is a function of the carrier phase offset θ and depends on the constellation of the underlying data. For a particular modulation scheme, the phase error S-curve can be derived empirically for different noise levels. Figure 3.10 depicts the error signal for 16-, 64-, and 256-QAM, each with a high and low SNR level, relative to the constellation size. The standard phased-locked loop will advance the reference phase for a positive error and retard it when the phase error is negative. As such, any zero-crossing of the S-curve with a positive slope will cause the PLL to settle. Interestingly, when the SNR is relatively high, each of the modulation schemes produces a pair of spurious points which could result in a false lock in the PLL. These are depicted as the solid points in Figure 3.10 for each of the three square QAM constellations. A noisier signal has a higher probability of kicking the loop out of the false lock, allowing it to eventually settle on true phase.

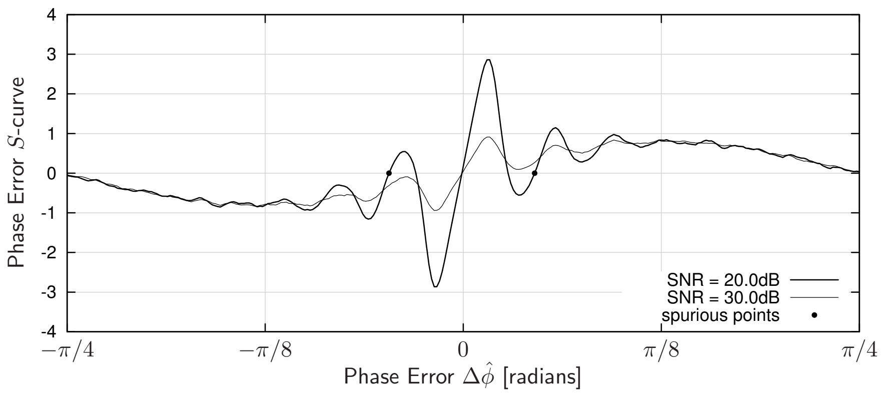
Figure 3.11 demonstrates how a 16-QAM signal with a higher signal level can lock onto the wrong phase due to a spurious response in the phase error S-curve. As such, it is important for systems



(a) 16-QAM

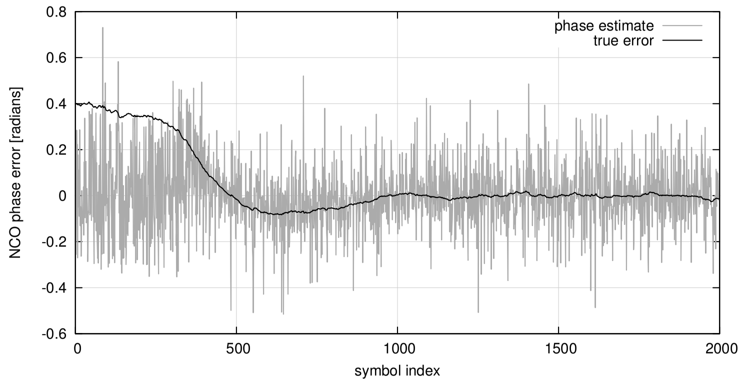


(b) 64-QAM

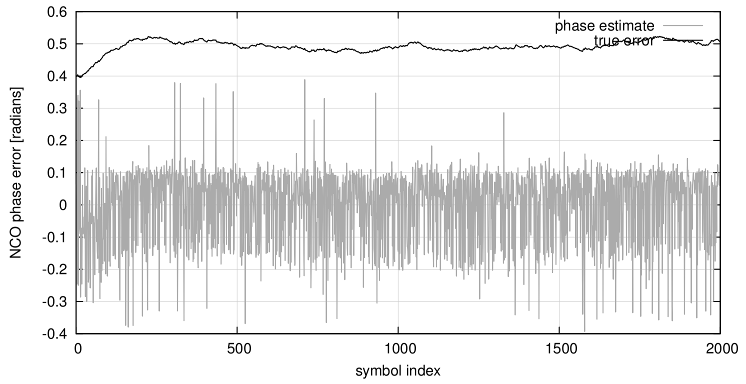


(c) 256-QAM

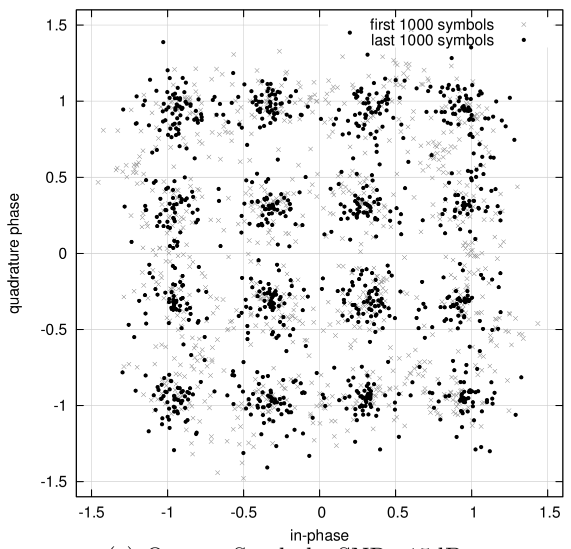
Figure 3.10: S-curves for the phase error detector of certain modulation schemes, both high and low relative SNR levels displayed.



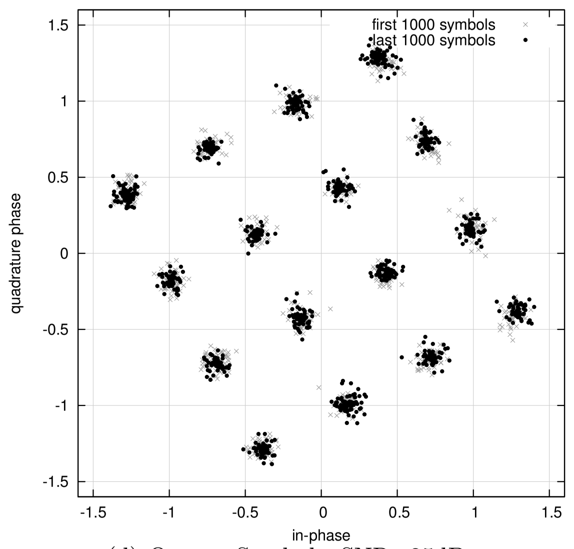
(a) Phase Response, SNR=15dB



(b) Phase Response, SNR=25dB



(c) Output Symbols, SNR=15dB



(d) Output Symbols, SNR=25dB

Figure 3.11: Carrier phase response for 16-QAM with different SNR levels. The signal with a higher SNR result in a false lock on spurious phase points, demonstrated in Figure 3.10(a).

using high-order modulation schemes to make an initial estimate of the phase such that the tracking loop operates within the pull-in range of the phase error and does not lock onto spurious points in the signal. Section 3.5 will investigate this further where a new framing structure is introduced which provides an initial phase estimate for the received signal.

3.4 Joint Carrier and Timing Recovery

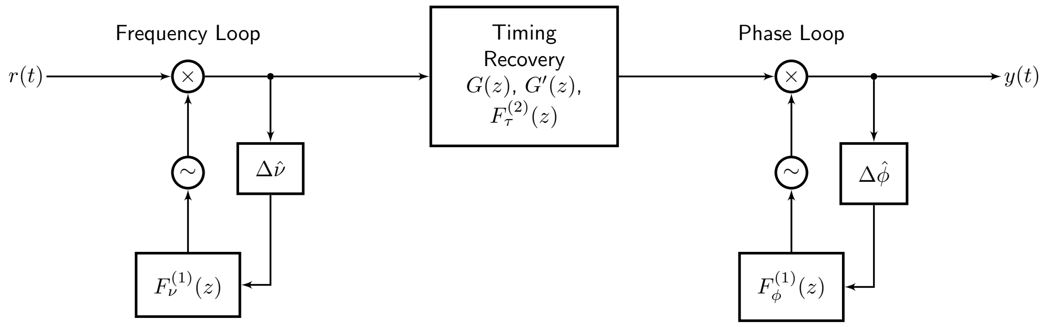
The joint maximum likelihood estimate of carrier phase and timing offset results in error signals for θ and τ which are dependent upon both parameters [59, 68]. The preceding sections have assumed independent estimates of θ, ν, and τ. While joint estimators cannot be worse than estimating parameters individually, it is typically convenient to design them independently. Furthermore, interactions of a joint estimator can lead to synchronization failures. In particular, Mengali and D'Andrea demonstrated that a synchronizer for a 64-QAM signal can settle into a false phase lock due to spurious points in the two-parameter timing and phase S-curves [67, Fig. 8.3–8.5]. These are typically a result of symmetries in the signal constellation themselves on which synchronizers can settle depending upon their initial conditions. As such, joint estimators are susceptible to the same complications as the coalition of independent estimators.

As discussed in Section 3.2 timing recovery can be accomplished asynchronously so long as the carrier frequency offset is not a significant portion of the signal's symbol rate. In this regard it is desirable to devise a joint carrier and timing recovery system which merges independent estimates of the unknown reference parameters. This section introduces three primary architectures for the joint recovery of timing and carrier information using independent estimates of each. The efficacy of implementation for each from a software radio standpoint is discussed, and the performance of the proposed design is demonstrated.

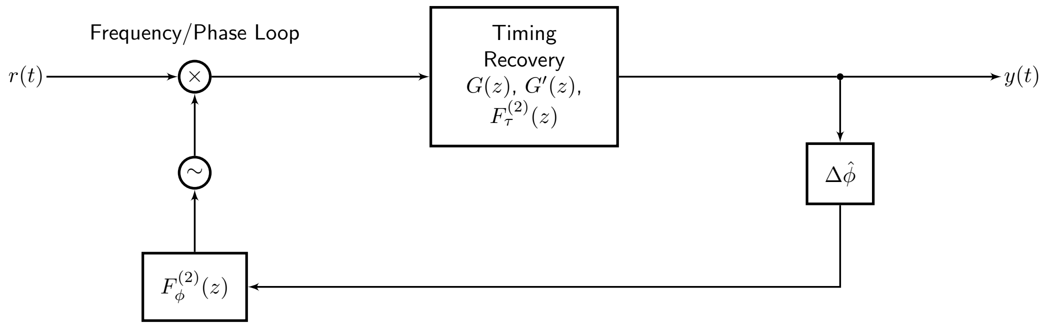
3.4.1 Dual Loop Architectures

Figure 3.12 depicts the three possible joint carrier and timing recovery architectures. The timing recovery loop acts as a down-sampler such that it accepts an input at a rate of approximately k samples/symbol and outputs exactly one sample/symbol. The demodulator computes an estimate of the phase error $\Delta\hat{\phi}$ as given by (3.25).

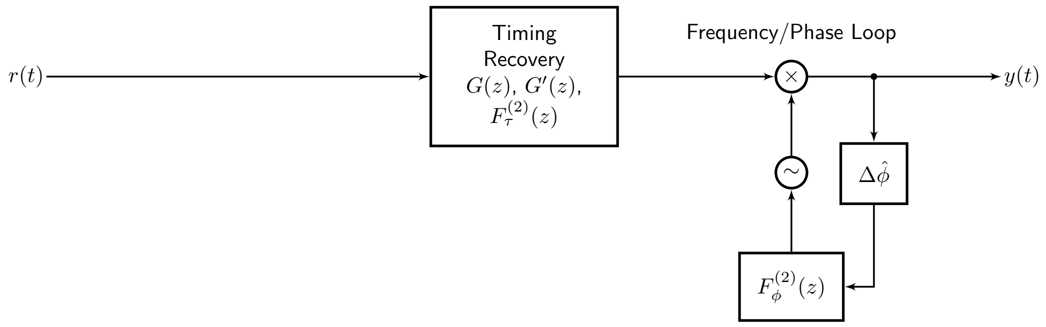
The first architecture, shown in Figure 3.12(a), contains two carrier loops; one to compensate for coarse frequency offsets and the other for residual phase correction. This is perhaps the most robust solution as it can correct for significant carrier frequency offsets between the transmit and receive oscillators, so long as the first-order frequency-locked loop filter $F_\nu^{(1)}(z)$ is relatively stable and free of oscillations. Furthermore, the introduction of a framing preamble permits the explicit estimation of the initial carrier frequency offset which can be used in place of the frequency loop altogether. Because the first carrier loop can only recover frequency information, a first-order phased-locked loop placed after the down-sampler is necessary to recover the phase. Consequently, this loop architecture has the additional computational load of mixing the signal twice; a process which consumes valuable clock cycles at the receiver. This is particularly detrimental to SDR as these operations operate at the highest sample rate in the receiver. Furthermore, it is desirable to



(a) Independent carrier frequency/phase loops



(b) Pre-timing carrier recovery



(c) Post-timing carrier recovery (preferred)

Figure 3.12: Joint carrier and timing recovery loop architecture diagrams.

first filter the signal from noise before making an estimate of the carrier frequency offset, ν.

The second architecture aims to reduce the computational requirements in the first design by placing just a single mixer/oscillator pair before the down-sampler and is shown in Figure 3.12(b). Because coherent phase recovery is necessarily computed *after* the down-sampler, the control loop must loop back around the timing recovery control circuit. This type of architecture has been investigated in [39]. While a feasible design, the difficulty is designing a stable carrier tracking loop which accounts for the delay introduced by the matched filter. This instability is equivalent to that discussed in Section 3.2.4, only it now applies to the carrier recovery loop as well. As a result of the delay introduced by the matched filter, the bandwidth of the carrier loop must be significantly reduced to maintain stability of the system. This is undesirable as reducing the bandwidth increases the lock time of the synchronizer.

The third architecture, shown in Figure 3.12(c), places the mixer/oscillator pair as well as the phase error generator after the down-sampler. This eliminates the effects of introducing loop delay found in pre-timing carrier recovery architecture of Figure 3.12(b). Furthermore, because the mixing process occurs after the matched filter/down sampler, it operates at a minimal rate (one sample/symbol). The major drawback is that the initial carrier frequency offset can cause distortion in the matched-filter output if it is large relative to the symbol rate. Furthermore, the pull-in range of the PLL is limited by the signal's modulation scheme. However, these drawbacks can be mitigated by choosing a proper phasing pattern on which to initialize the carrier loop. Section 3.5 provides such a preamble and discusses its performance in the presence of gross carrier frequency offsets. As such, this architecture is preferred to the other two for use in software radio implementation.

3.4.2 Proposed Joint Tracking Loop

The remainder of this section extends the work presented in [30, 39] and Section 3.2 by introducing an additional control loop for carrier synchronization after matched filtering. A numerically-controlled oscillator placed after matched-filtering the received signal ensures stability by eliminating delay in the feedback path, and reduces computational complexity by operating at just one sample per symbol. This design, depicted in Figure 3.12(c), allows for simultaneous tracking of both symbol timing and carrier phase mismatches and is particularly tractable for burst-mode duplexing schemes where carrier phase and symbol timing acquisition needs to be performed on each transmission burst. The loop architecture described in this section incorporates coarse frequency and symbol timing offset estimators in a natural way; however, the discussion is limited to tracking mechanisms and assumes acquisition of timing and carrier frequencies has been established. Carrier phase synchronization before matched filtering can be corrupted by out-of-band noise and adjacent-channel interference, and applying the carrier recovery algorithm before the decimator can consume valuable processing resources. This motivates an investigation on alternative techniques that combine the two into a single processing block with independent control loops. Common concerns issued to systems operating with two feedback loops can be avoided by realizing that the timing recovery algorithm operates independently of the input phase, so long as the input carrier frequency offset remains sufficiently small. Issues with carrier loop perturbations due to tracking before timing synchronization is achieved is also addressed.

The carrier loop operates in a similar fashion to the timing phase recovery loop; a generated phase error signal is fed through a second-order low-pass loop filter $F_{\phi}^{(2)}(z)$ that drives an NCO. One of the driving motivations for combining carrier phase recovery with symbol timing is that matched filtering can be performed before generating the phase error, thus reducing out-of-band interference and noise applied to the NCO. Several possibilities exist for generating phase error signals where timing information is known. For systems operating at 1 sample/symbol, perhaps the most obvious solution is a decision-directed loop when knowledge of the modulation scheme utilized is known, e.g. (3.25).

3.4.3 Performance Results

For all simulations the received signal is generated at 2 samples/symbol and without the addition of noise. The timing loop filter and carrier loop filter operate with a noise bandwidth of approximately 0.5% of the symbol rate and 1% of the sampling frequency, respectively. A set of 32-stage polyphase filterbanks are used in the timing loop for matched filtering. The timing error generator is an approximation to the maximum likelihood error detector as defined by (3.13).

Timing & Carrier Phase Step Response

Figure 3.13 depicts the results of a simulation in which both a timing and a carrier phase offset are introduced. Because the timing offset ($0.3T$) is positive, the timing recovery loop slides backwards through the filterbank indices until it settles on $m = 13$ which is closest to $(0.5 - \tau)kM = 12.8$. Additionally, the carrier loop tracks to the phase offset and eventually settles on the appropriate -0.4 radians. Because the carrier loop operates at one sample/symbol and depends so highly upon the accuracy of the matched filter, acquisition of timing is critical to this system. One very important thing to note is that the carrier phase recovery loop depends on the accurate knowledge of symbol timing. Due to the initial error in symbol timing, the phase error detector results in a non-zero NCO control value in the carrier recovery loop, despite there being no initial carrier phase error in the simulation. Although the carrier loop recovers quickly, the system could make use of a timing lock mechanism to disable the carrier phase control loop before timing synchronization is achieved.

Timing Phase & Carrier Frequency Step Response

Figure 3.14 depicts the results of a simulation in which both a timing phase and a carrier frequency offset are introduced. The timing control settles on the filter in the bank at index $m = 26$ which is closest to $\tau kM = 25.6$ after about 400 symbols. Because the output of the timing recovery loop takes longer to settle, the phase error estimate is poor and prevents the NCO from tracking to the carrier frequency offset until about 150 symbols. Eventually, however, the phase error has sufficient noise suppression to linearly increase the NCO control and track to the input carrier frequency offset.

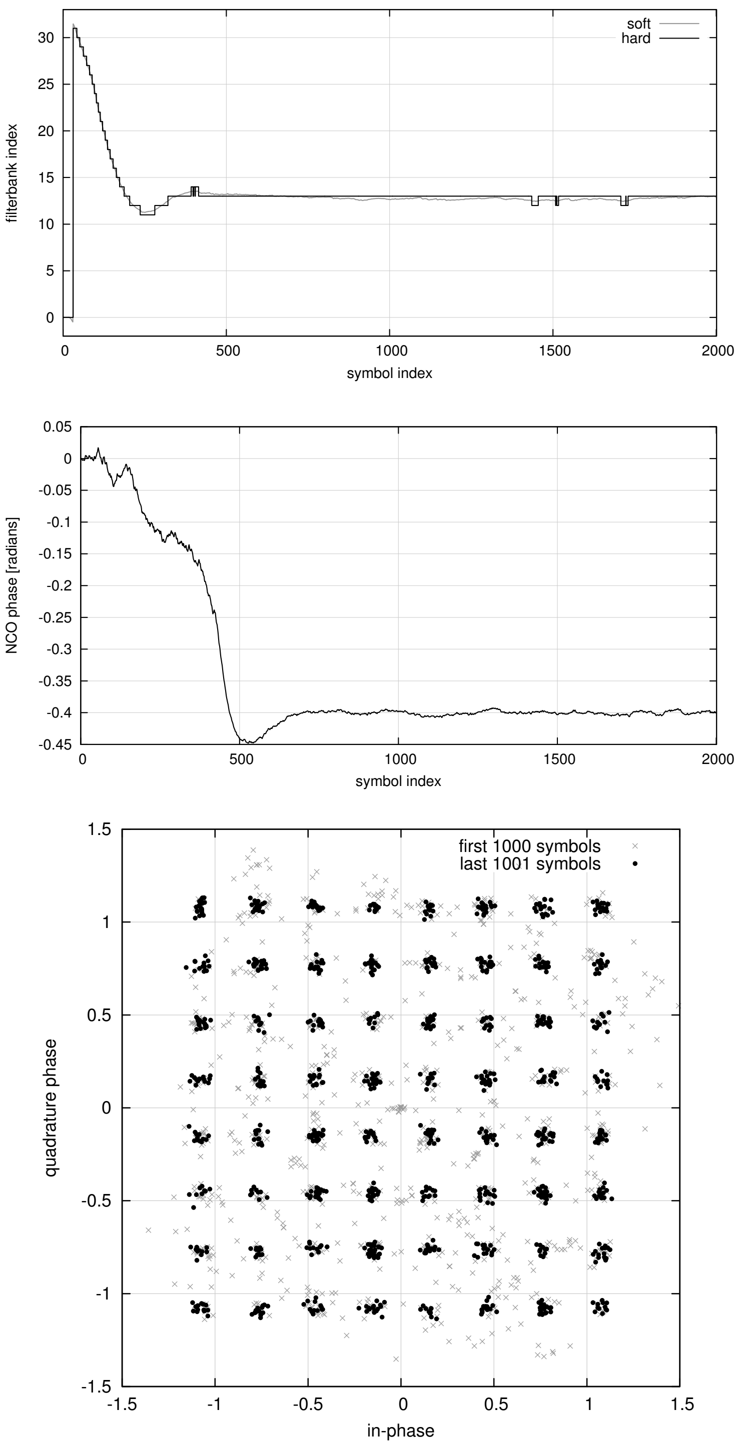


Figure 3.13: Joint symbol timing and carrier phase/frequency tracking with a timing phase offset of $\tau = 0.3T$ $(M = 32)$, a carrier phase offset of $\theta = -0.4$ radians, and with an SNR of 30dB.

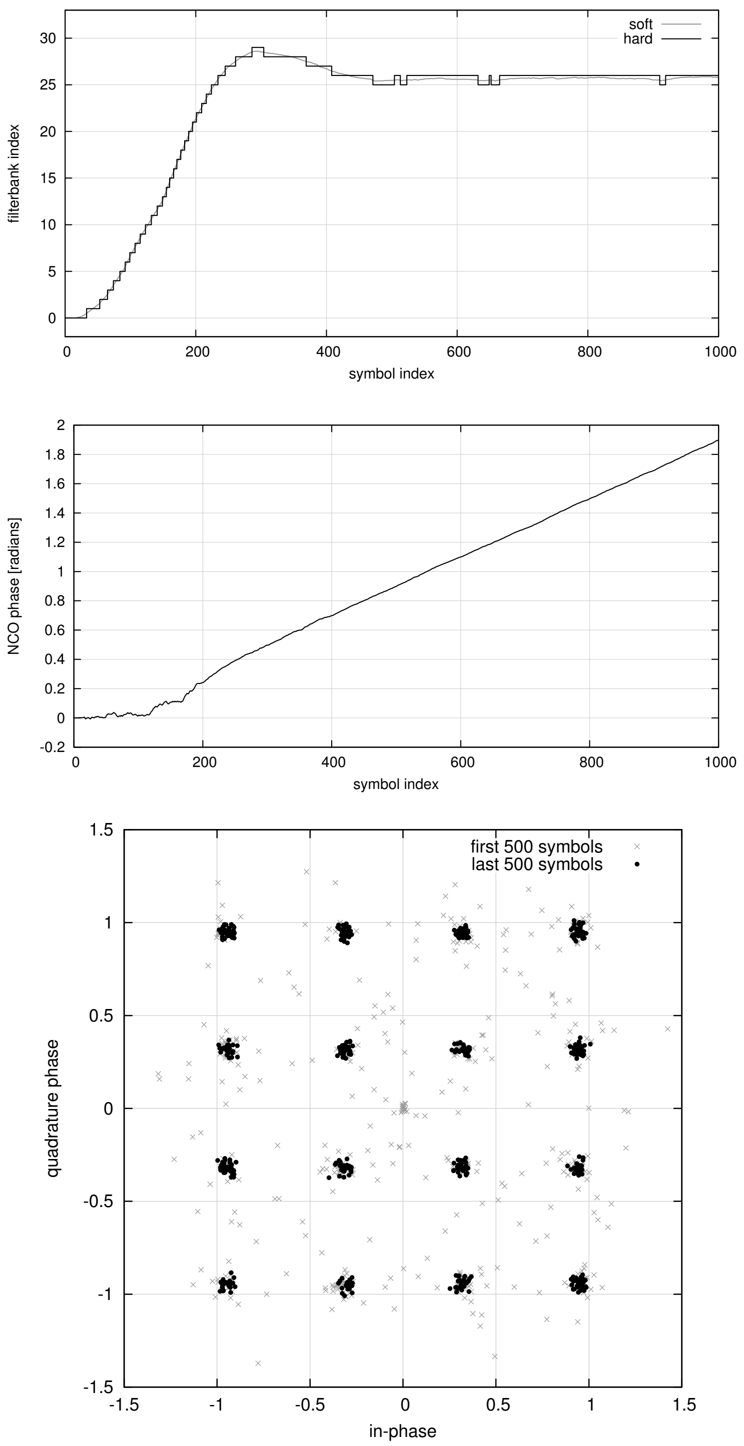


Figure 3.14: Joint symbol timing and carrier phase/frequency tracking with a timing phase offset of $\tau = -0.4T$ ($M = 32$), carrier frequency offset of $\nu = 0.002$ radians/symbol, with an SNR of 30dB.

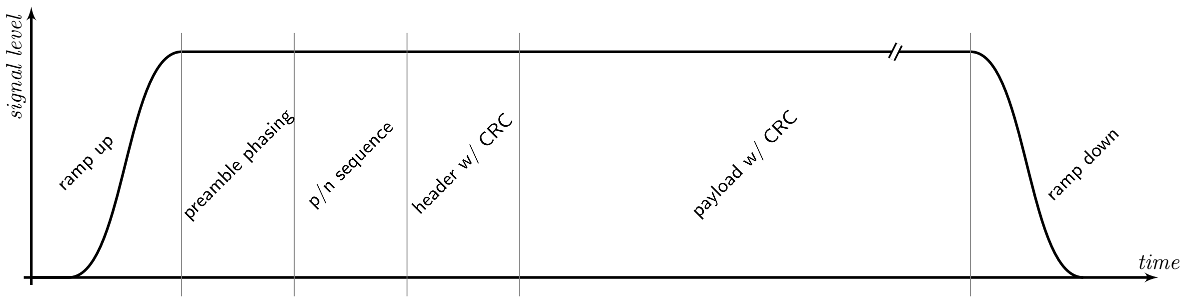


Figure 3.15: Framing structure used in over-the-air experimentation. The payload length is variable and is typically much longer than the preamble.

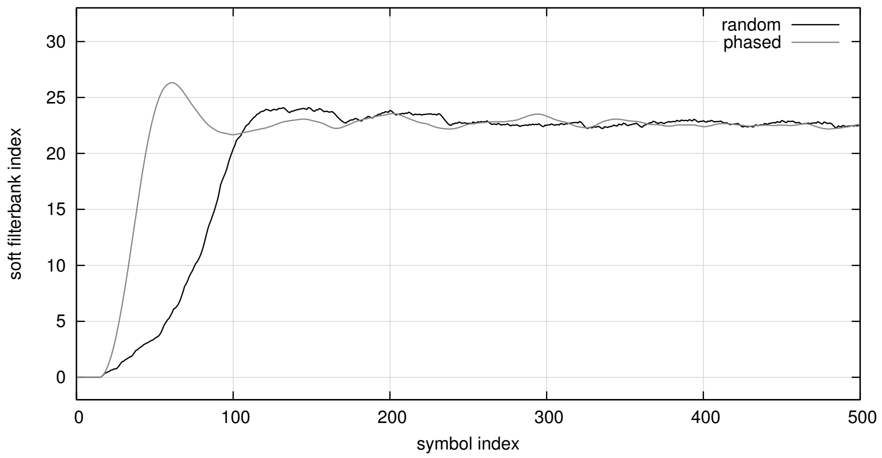
3.5 `flexframe`, a Versatile Framing Descriptor

With the necessity for dynamic spectrum access networks to periodically sense the wireless environment comes the need for their data links to quickly and efficiently synchronize to each frame. This section introduces a novel framing structure for software-defined radio which takes advantages of the efficient synchronization processing elements discussed in this chapter. The framing structure, called `flexframe`, is capable of receiving virtually any linear modulation scheme and combines timing and carrier frequency acquisition and tracking, automatic gain control, equalization, and forward error-correction decoding within a single module. Performance is verified with baseband simulation and validated with over-the-air transmission in a laboratory environment.

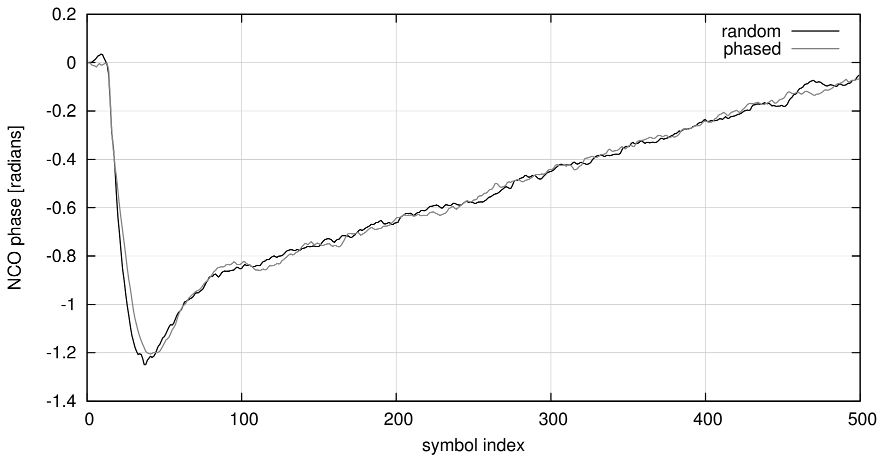
3.5.1 Framing Structure

The framing structure consists of six basic components, as depicted by Figure 3.15. The ramp up, phasing pattern, and P/N sequence are together known as the physical layer convergence procedure and are used for fast acquisition of the frame at the receiver through packet detection, automatic gain control (AGC) lock, carrier phase-locked loop (PLL) lock, and symbol timing PLL lock. The ramp up and preamble phasing sequences are a BPSK pattern which flips phase for each transmitted symbol, i.e. $a(p) = (-1)^p = \{+1, -1, +1, -1, \ldots\}$. This sequence serves several purposes but primarily to help the receiver's symbol synchronization control loop lock onto the proper timing phase. As Figure 3.16 demonstrates, the timing recovery tracking loop converges much most quickly for a phased sequence than a random one. The ramping nature of the beginning of the frame gracefully increases the output signal level to avoid "key clicking" and reduce spectral side-lobes in the transmitted signal. Furthermore, it allows the receiver's automatic gain control unit to lock on to the incoming signal, preventing sharp transitions in its output.

The P/N sequence is a 63-bit m-sequence (maximal length) exhibiting good auto- and cross-correlation properties [76, pp. 435—437]. This sequence aligns the synchronizers to the remainder of the frame, indicating when to start receiving and decoding the frame header, as well as indicating the coherent phase of the received signal. It also provides a unique training sequence upon which to initialize the equalizer. At this point it is assumed that the receiver's AGC, carrier PLL, and



(a) Timing recovery



(b) Carrier frequency/phase recovery

Figure 3.16: Preamble BPSK phasing pattern performance vs. random data bits. Symbol timing PLL bandwidth is 0.05, phase PLL bandwidth 0.02, $\theta = -\pi/3$ radians, $\nu = 0.001/F_s$, SNR= 20dB.

timing PLL all have locked to the signal at which time the bandwidths of each are reduced to a suitable tracking mode.

The header consists of several data bytes describing to the receiver how to decode the payload. In particular, it includes the modulation scheme, modulation depth, and the inner and outer forward error-correction codes used on the payload data. The header adds a minimal amount of overhead to the frame as a whole and permits the receiver to reconfigure itself on-the-fly for each packet it detects. It includes a 32-bit cyclic redundancy check to validate the integrity of the encapsulated data and is encoded using a simple 2/3-rate Hamming(12,8) block code and modulated using BPSK. The header needs to be highly reliable as decoding the remainder of the frame is contingent upon its proper reception. The choice of using a Hamming(12,8) code was made to provide a certain amount of error correction capabilities without burdening the receiver with computationally complex alternatives. Convolutional coding was specifically avoided in the header as the Viterbi decoder provides only marginal performance improvements on such a short data sequence while increasing the computation required at the receiver. See Section 5.5 for more details on the computational efficiency of forward error-correction codes.

At the core of the frame is the payload, containing the raw data to be transferred across the link. Its length in terms of number of symbols is variable depending upon the number of raw payload bits, forward error-correction used (if any), and modulation scheme used. Before any error-correction encoding is applied, a 32-bit cyclic redundancy check is appended to the raw data to validate that the frame was properly received. Finally, the ramp down sequence serves to gracefully lower the transmitted power of the signal.

3.5.2 Spectral Efficiency

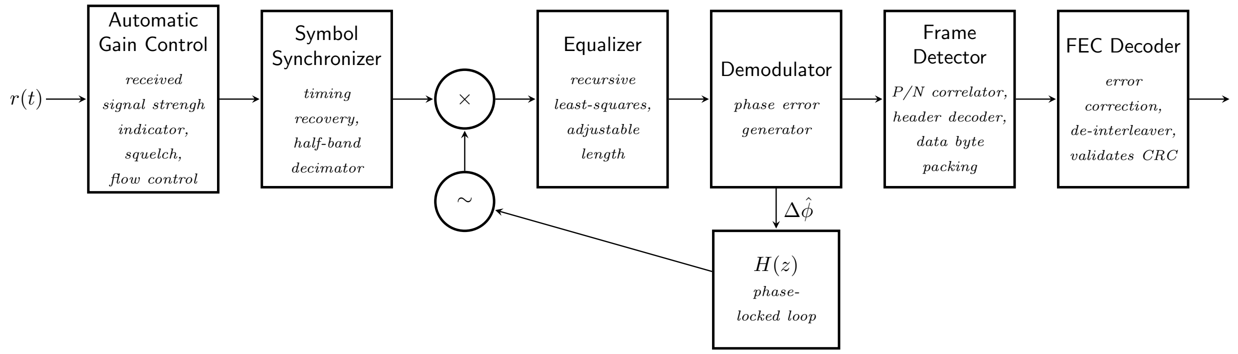
The overhead due to the ramp up/down sequences, phasing pattern, P/N sequence, and header reduces the spectral efficiency of the frame. A payload of n uncoded data bits encapsulated within a pair of cascaded forward error-correction codes with rates r_0 and r_1 respectively and modulated with a scheme using k bits per symbol results in a payload of n/r_0r_1k symbols. The framing overhead introduces an additional $\bar{n}$ symbols which is the sum of the ramp up/down sequences, the phasing pattern, the P/N sequence, and the header. The true spectral efficiency is therefore

$$\hat{\eta} = r_0 r_1 k \left[\frac{(n+32)/r_0 r_1 k}{(n+32)/r_0 r_1 k + \bar{n}} \right] \tag{3.26}$$

The overhead can be reduced by decreasing the length of the ramp up, phasing pattern, and ramp down sequences at the risk of increasing the probability of missing a packet. The spectral efficiency can also be increased by packing the frame with more payload bits at the cost of increased memory usage and latency.

3.5.3 Synchronization

A block diagram of the `flexframe` synchronizer is shown in Figure 3.17. The automatic gain control block provides not only gain correction, but provides a received signal strength indication as well



$r(t)$
Automatic Gain Control
received signal strengh indicator, squelch, flow control
Symbol Synchronizer
timing recovery, half-band decimator
×
~
Equalizer
recursive least-squares, adjustable length
Demodulator
phase error generator
$\Delta\hat{\phi}$
$H(z)$
phase-locked loop
Frame Detector
P/N correlator, header decoder, data byte packing
FEC Decoder
error correction, de-interleaver, validates CRC

Figure 3.17: `flexframe` synchronizer block diagram.

as squelch and flow control to the rest of the synchronizer. The symbol synchronizer implements the polyphase filterbank structure for timing recover described in Section 3.2. It operates at $k = 2$ samples/symbol and can be adjusted for any practical filter delay m, excess bandwidth β and timing resolution M. Notice that placing the symbol synchronizer in front of the NCO and mixer implies that it operates incoherently of the carrier phase and assumes that the received signal has a small carrier frequency offset relative to the symbol rate (e.g. $\nu T \ll 1$).

The equalizer performs a recursive least-squares optimization search over an array of complex filter coefficients [76, Section 11.4]. Training is carried out after the frame has been detected from the P/N correlator output. The equalizer trains its coefficients on the error between the received sequence of 64 symbols to the ideal transmitted pattern. Once initialized, the equalizer filter is fixed and does not change for the remainder of the frame. This is to reduce the computational complexity of the receiver and assumes that the frame is relatively short compared to the coherence time of the channel. The filter has a nominal delay of zero samples because it is designed such that its first coefficient is prominent. With no delay, the equalizer conveniently has no effect on the carrier phase-locked loop, unlike in the symbol synchronizer where the matched filter's time delay can significantly affect its loop stability (see Section 3.2.4). The equalizer's length is adjustable and can be specified at any time, but the synchronizer only reconfigures the equalizer whenever it receives a frame.

The demodulator component has several tasks. Not only is it responsible for estimating which symbol was transmitted, it also provides an estimate of the carrier phase error to the phase-locked loop. Furthermore, it keeps track of the signal constellation's error vector magnitude which is a good indicator of the ratio of the signal level to noise, distortion, inter-symbol interference, and other channel impairments. This information is useful in dynamic spectrum access networks where the signal level might be high, but the error rate is still large. A radio controller can use this information to make a decision about how to switch its transmission scheme in order to improve its performance.

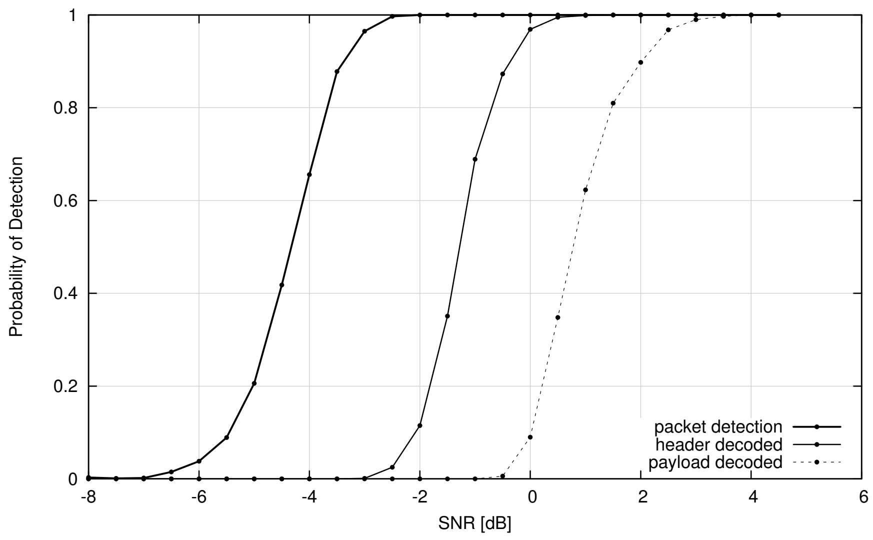


Figure 3.18: Probability of detection for the proposed frame synchronizer vs. signal-to-noise ratio. The payload is 512 bits of uncoded random BPSK symbols, 10,000 trials.

3.5.4 Performance

Figure 3.18 demonstrates the performance of the framing structure's detection probability vs. SNR. The simulation was conducted by fixing the noise power and adjusting the channel gain, forcing the synchronizer's AGC module to dynamically adjust to the varying signal levels. By running 10,000 frame trials and measuring how many of these were properly decoded, the likelihood of finding a frame buried in noise can be determined. The two curves to the left are the most telling: The packet can be detected with a 0.9 probability at about -3.5dB SNR; however, the frame header needs about -0.5dB SNR to have a probability of 0.90 to being properly decoded. This is a limitation of the modulation and coding scheme used in the frame header which must be consistent across all frames.

The limiting factor of placing the carrier recovery loop after the down-sampler, as discussed in Section 3.4.1, is that any initial carrier phase offset can cause significant distortion in the matched-filter output if it is large relative to the symbol rate. As stated in Section 3.5.2, framing overhead can be reduced by limiting the length of the initial phasing pattern at the cost of decreasing detection probability. Figure 3.19 depicts the probability of frame detection when a significant carrier offset is present. The simulation was conducted by computing the statistics of 1,000 trials each for three different phasing lengths. As demonstrated by the figure, the probability of detection can be kept sufficiently high even with manageable phasing lengths. In fact, with even just a few phasing symbols the detection probability is over 90% with a carrier frequency offset less than 12% of the symbol rate.

Figure 3.20 demonstrates the `flexframe` synchronizer over the air. From the received signal strength indication of the automatic gain control, the signal level is approximately 20dB above the noise floor. The received signal starts at around the 400 sample time mark at which point the

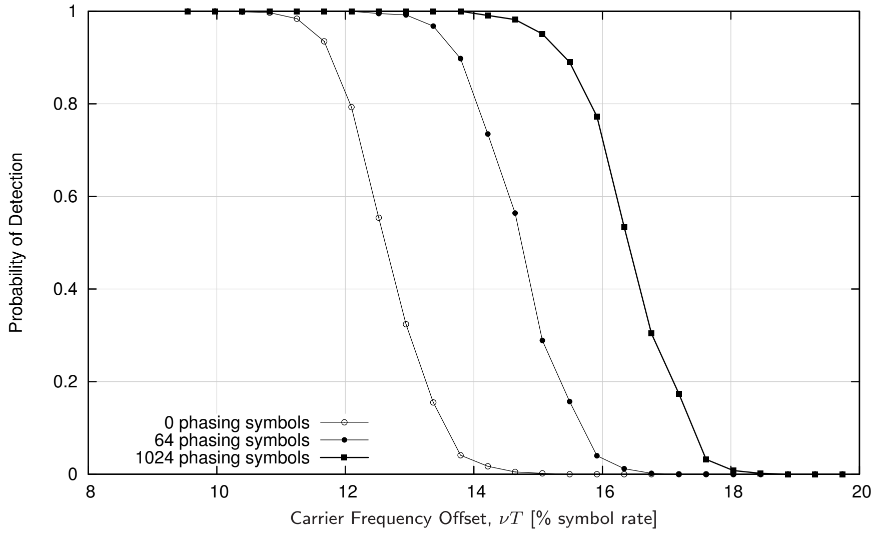


Figure 3.19: Probability of detection for the proposed frame synchronizer vs. carrier frequency offset as a percentage of symbol rate. The payload is 512 bits of uncoded random BPSK symbols, in a AWGN channel with an SNR of 20dB, averaged over 1,000 trials.

BPSK phasing pattern is clearly visible. The slight carrier frequency offset is visible until about the 600^{th} symbol as the BPSK signal rotates through the in-phase and quadrature channels periodically. The PLL locks onto the carrier quickly as the phase of the output symbols is linear. The phase jumps at around symbol 380 by π radians due to the P/N sequence correlator flipping the phase of the BPSK sequence so that the remainder of the frame can be demodulated coherently. Once the header is received, the remainder of the frame is demodulated as 16-QAM with a relatively tight signal constellation.

3.6 Conclusions

This chapter has introduced the problem of synchronization of wireless digital communications signals and introduced several new methods for recovering symbol timing and carrier offsets on software-defined receivers. The new synchronizers specifically focus on maximum likelihood-based estimators which are shown to be computationally efficient while exhibiting a variance very near the lower bound. Additionally a novel framing structure for narrowband signals is proposed which permits dynamic adjustment of the modulation parameters and payload length with little overhead. The framing receiver is capable of fast acquisition while recovering data with very low SNR levels, and its performance is validated with over-the-air reception in a wireless multi-path laboratory environment.

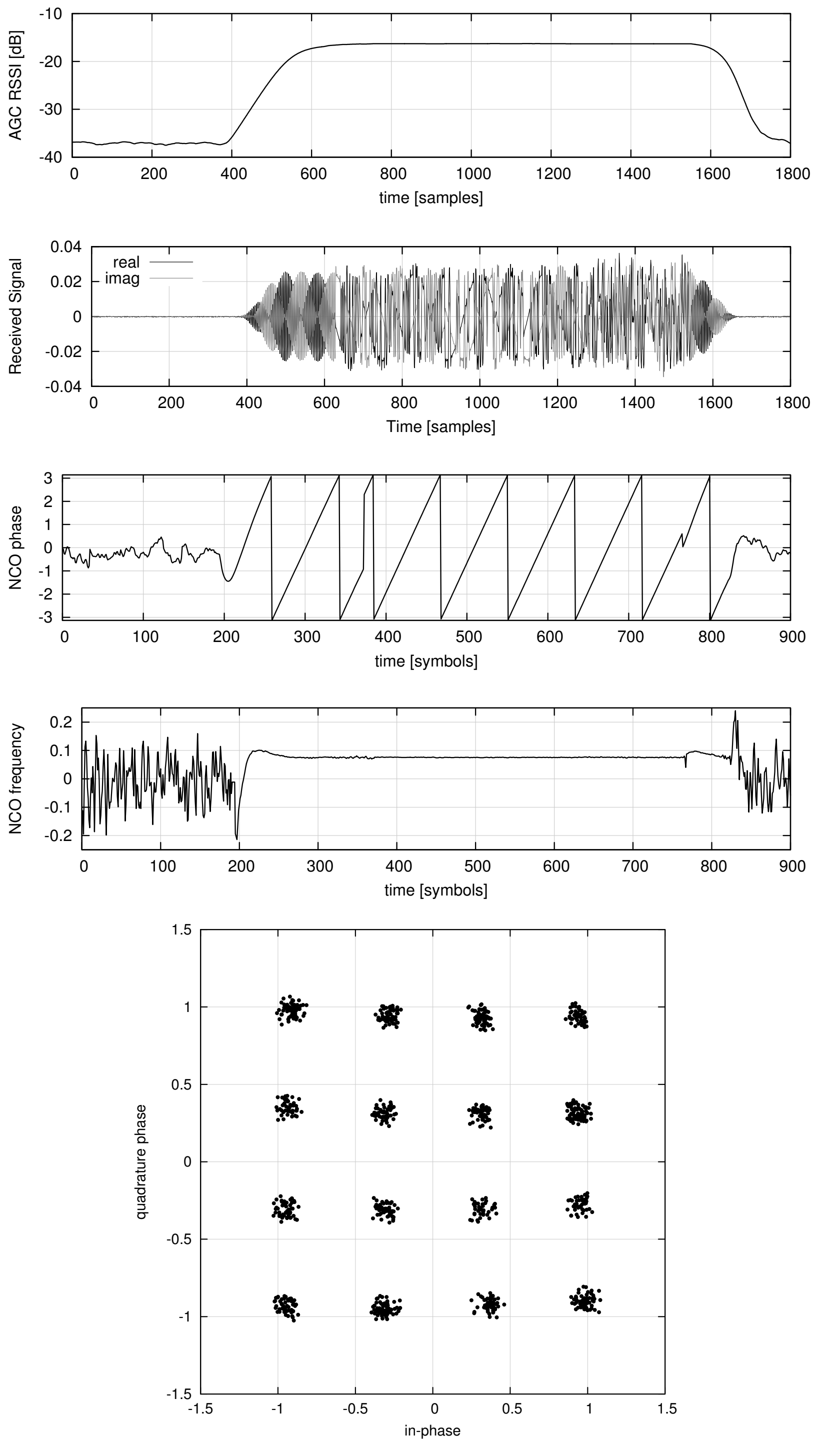


Figure 3.20: `flexframe` synchronizer, over-the-air example

Chapter 4

Synchronization of Multichannel Communications Systems

Pulse-shaping filters used in digital communications systems should have good time-frequency localization properties. As demonstrated in Chapter 2, an appropriate multi-carrier system (OFDM/OQAM) can be derived from a prototype pulse-shaping filter which results in significant spectral compactness improvements over traditional OFDM, even when window tapering is applied to the symbol transitions. The nature of filtering permits the symbols to overlap in time and the subcarriers to overlap in frequency. As a consequence, synchronization of OFDM/OQAM is not as trivial as that of OFDM. Still, the promise of improving spectral efficiency motivates the investigation of synchronization techniques for OFDM/OQAM.

The previous chapter introduced several new maximum likelihood-based techniques for synchronization of digitally modulated single carrier signals. This chapter extends this work by investigating similar methods for estimation of unknown reference parameters for OFDM/OQAM. Specifically, this work provides several new estimators for the sample timing offset, carrier frequency/phase, and equalization coefficients, with their performance characterized in channels exhibiting noise, multi-path fading, and narrow-band interference. Further included is theory on estimating the complex gains of pilot subcarriers as well as how OFDM/OQAM may be used to (almost) perfectly reconstruct the time series of an interfering signal simply by re-routing the channelized output of the the down-converter into a synthesizer polyphase filter bank.

4.1 System Description

Parallel data telecommunications schemes divide available frequency bands into several channels by independently modulating a number of carriers of different frequency. The bandwidth of each channel is much narrower in relation to the entire occupied bandwidth. As a result parallel transmission is effective in combating amplitude and delay distortion and is robust to impulsive noise as well as narrowband interference. It is tempting to choose subcarriers which do not overlap in frequency as to avoid inter-channel interference; however, this would result in inefficient use of the spectrum and loss of channel capacity. By permitting adjacent subcarriers to overlap, the system approaches the theoretical maximum capacity of the channel.

Traditional OFDM guarantees orthogonality between subcarriers by restricting the pulse shape to be rectangular over the symbol duration. The carriers are chosen such that an even multiple of sinusoidal periods fits within the pulse duration. The combination of these constraints guarantees no inter-symbol or inter-channel interference in an additive noise channel. Furthermore, the receiver (and transmitter) can be efficiently realized with a fast (inverse) Fourier transform. Section 2.2, however, has discussed the many drawbacks to using OFDM in dynamic spectrum environments, particularly when used as a spectrum sensing mechanism. One particular drawbacks is the spectral rolloff of each subcarrier due to the rectangular pulse shape.

4.1.1 OFDM/OQAM

By permitting the baseband channels to be filtered the spectral characteristics of the system can be precisely controlled. The spectral rolloff of each OFDM subcarrier could be improved if a windowing function were applied to each symbol; however, this would break the strict orthogonality conditions of OFDM. Chang and Saltzberg in [17, 83] presented general conditions under which a set of band-limited channels with a symbol rate $1/T$ can be separated by $1/2T$ without the introduction of inter-symbol and inter-channel interference. The pulse-shaping filter $g(t)$ for each subcarrier is identical and assumed to be a square-root Nyquist with a rolloff factor β. This eliminates inter-symbol interference from symbols within the same channel. The center frequencies of successive subcarriers are separated by $1/2T$ which is precisely half the signaling rate of each channel. This implies that adjacent subcarriers overlap by as much as 50% (see Figure 2.5(b)). This overlap causes cross-talk in the in-phase and quadrature components in neighboring bins. Inter-channel interference, however, is eliminated when adjacent channels are in phase quadrature with one another, and therefore the in-phase and quadrature components are delayed by $T/2$ on alternating subcarriers. Both filters $G(z)$ in the transmitter and receiver are assumed to be identical and real, thus assuring optimum performance in the presence of additive white Gauss noise. Because the filters are band-limited, cross-talk between channels which are not immediately adjacent is eliminated.

$$G(\omega) \approx 0, \quad |\omega| \geq \pi/T$$

Furthermore, the matched-filter response $H(\omega)$ of the transmit and receive filters $G(\omega)$ in conjunction has a Nyquist rolloff,

$$G^2(\omega) + G^2(\pi/T - \omega) = 1, \quad 0 \leq |\omega| \leq \pi/2T$$

The system is known as orthogonal frequency-division multiplexing using offset quadrature amplitude modulation (OFDM/OQAM). A proof of orthogonality for OFDM/OQAM is available in [83] and [24, Section 7.3.3].

Consider an M-channel OFDM/OQAM system of which only M_u subcarriers are enabled. The real and imaginary components for the m^{th} subcarrier transmitted at the p^{th} symbol index are $a_m^{\mathcal{R}}(p)$ and $a_m^{\mathcal{I}}(p)$, respectively, and are separated in time by $T/2$ (half a symbol period). Furthermore, the delay between in-phase and quadrature components alternates between successive subcarriers.

The resulting transmitted signal is therefore

$$s(t) = \sqrt{\frac{M}{2M_u}} \sum_{p=-\infty}^{\infty} \sum_{m \in \mathcal{A}} e^{jm(2\pi t/T + \pi/2)} \Big[a_m^{\mathcal{R}}(p) g(t - pT) + j a_m^{\mathcal{I}}(p) g(t - pT - T/2) \Big] \tag{4.1}$$

where m is the subcarrier index of M total subcarriers, p is the symbol index at time pMT_s, $a_m^{\mathcal{R}}(p)$ and $a_m^{\mathcal{I}}(p)$ are the respective real and imaginary symbols at index p on subcarrier m, $g(t)$ is a square-root Nyquist pulse shaping filter, and $\mathcal{A}$ is the set of all enabled subcarriers. Converting to discrete time samples gives

$$s(kT_s) = \sqrt{\frac{M}{2M_u}} \sum_{p=-\infty}^{\infty} \sum_{m \in \mathcal{A}} e^{jm\left(\frac{2\pi}{T} kT_s + \pi/2\right)} \Big[a_m^{\mathcal{R}}(p) g(kT_s - pMT_s) + j a_m^{\mathcal{I}}(p) g(kT_s - pMT_s - MT_s/2) \Big] \tag{4.2}$$

The orthogonality conditions give no restriction on $g(t)$ other than it being a square-root Nyquist filter (e.g. root-raised cosine). Consequently the filter may have arbitrary length, spectral overlap (transition bandwidth) and stop-band suppression. These relaxed constraints on $g(t)$ make OFDM/OQAM particularly tractable for use in dynamic spectrum environments.

4.1.2 Distortion from Timing and Carrier Offsets

As in Section 3.1 the received signal is defined as a delayed version of $s(t)$ corrupted by carrier offsets, multi-path fading, noise, and with an additional interference term, viz.

$$r(t) = e^{j(\nu t + \theta)} \gamma s(t - \tau) * h_c(t) + n(t) + \psi(t) \tag{4.3}$$

where as before ν is the carrier frequency offset, θ is the carrier phase offset, γ is the channel gain (due to path loss, shadowing, etc.), τ is the sample timing phase offset, $h_c(t)$ is the the impulse response of the channel, $n(t)$ is the baseband noise term with single-sided power-spectral density $N_0/2$. and $\psi(t)$ is an additional narrow-band interference signal with power P_ψ. Saltzberg demonstrated in [83] that the performance of OFDM/OQAM is particularly sensitive to timing, carrier phase, and carrier frequency offsets, and that the resulting distortion to the input of the demodulator can have a significant impact on the error rate at the receiver. Furthermore, the distortion was found to be significantly affected by the choice of the baseband pulse shape $g(t)$, and in particular it is essential to minimize the amount of overlap between adjacent subcarriers to reduce inter-channel interference when timing and carrier offsets are introduced.

This differs significantly from traditional OFDM in which timing offsets are compensated through cyclically extending each OFDM symbol, and carrier phase offsets are translated to a simple phase rotation in each down-converted subchannel. In this regard, carrier phase offset in OFDM/OQAM does not translate to a demodulator phase shift in the down-sampled subcarrier because of the $T/2$ time delay between the in-phase and quadrature components. Distortions due to carrier frequency offsets, however, are still present on OFDM systems and are exacerbated as the spectral width of each subcarrier decreases (e.g. as a result of partitioning the spectrum into more subcarriers). This chapter will later demonstrate that properly designing an OFDM/OQAM preamble results in accurate knowledge of the channel reference parameters, providing a minimal amount of distortion in the system.

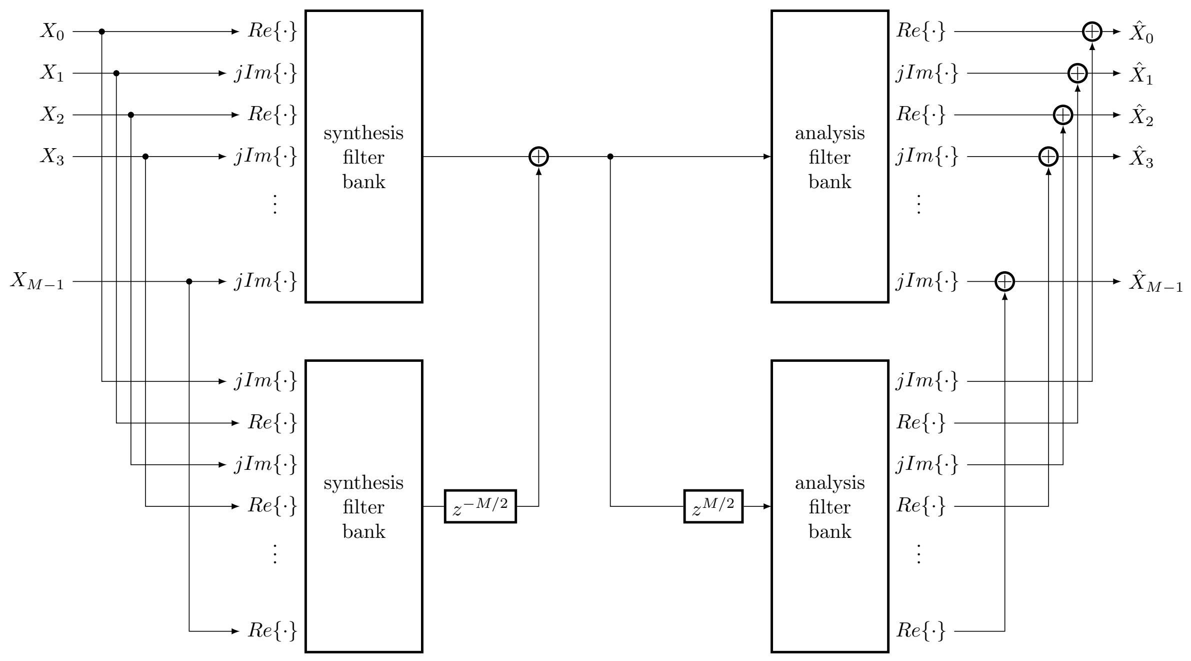


Figure 4.1: OFDM/OQAM channelizer using two identical yet separate DFT filterbanks.

4.1.3 Polyphase Filterbank Representation

The OFDM/OQAM transceiver can be represented by a pair of separate but identical DFT polyphase filterbank channelizers with a $T/2$ delay between their inputs [24, Section 8.6.9]. The system employing both the synthesis and analysis filterbanks can be seen in Figure 4.1; in effect, the channelizers efficiently realize a set of parallel down-sampling filters which recover the original data symbols when no channel impairments are present. This chapter shows that this basic prototype needs only slight modification in order to mitigate the effects of channel impairments.

4.1.4 Organization

This chapter will focus on recovering the unknown reference parameters in (4.3) to be congruent with a receiver based on DFT polyphase filterbanks as to provide as computationally efficient a receiver as possible. Multi-channel communications systems can extract information for estimation of the reference parameters in two ways:

1. *Before* de-multiplexing the subcarriers, typically by taking advantage of correlating whit explicit training data in the system. Traditional OFDM systems can also exploit cyclostationary features from the symbol's cyclic extension, regardless of whether or not the underlying data are known. The remainder of this chapter will refer to these as *time-domain* techniques. This work will later demonstrate that the success of many of these techniques are contingent upon good channel conditions as well as the absence of narrow-band interference.

2. *After* de-multiplexing the subcarriers, usually by relying on training symbols embedded into the data pattern. Such pilots carry enough information about the transmitted signal to guide the receiver towards synchronization. In the remainder of this chapter these will be known as *frequency-domain* techniques.

The majority of proposed multi-channel synchronizers employ a combination of both time- and frequency-domain techniques, typically exploiting temporal auto- and cross-correlation for acquisition and relying on training symbols embedded into the data sequence for tracking.

4.2 Timing Recovery

Section 3.2 has demonstrated how polyphase filterbanks can be used for timing recovery in narrowband systems, namely by devising a control loop which drives the interpolating filter in a bank. This motivates investigation into how such techniques can be used to synchronize OFDM/OQAM multi-carrier communications systems. As mentioned in the previous section, there are two main categories of timing estimation techniques: those which are run in the time domain, and those which are run in the frequency (down-sampled) domain. This section observes several estimators in both categories, provide performance results, and describe their importance for different operating conditions.

4.2.1 Time-domain Estimation

As demonstrated in Section 3.5, the introduction of a special preamble sequence with particular cyclostationary features can aid the convergence of carrier and timing phase-locked loops. In fact most existing systems (multi-carrier and otherwise) employ specific preamble convergence procedures for this purpose. For example, 802.11a is a 64-subcarrier OFDM system which uses a physical layer convergence procedure (PLCP) consisting of two short-sequence and two long-sequence OFDM symbols. The short sequences are generated by loading a particular QPSK pilot on every fourth subcarrier, corresponding to exactly four 16-sample temporal repetitions once the inverse transform is taken, hence the name "short" sequence. Furthermore, because 802.11a employs a 16-sample cyclic prefix, the resulting time series consists of ten 16-sample repetitions. The long sequences are generated by loading a different QPSK pilot sequence on all subcarriers. The resulting time series represents a unique sequence for which the receiver may correlate against. A number of resources exist on the topic of synchronizing 802.11a and its derivative protocols. The most common references deal with estimating the carrier frequency and timing offset parameters through temporal auto- and cross-correlation.

These 802.11a sequences provide motivation for a timing recovery estimator based on a sequence cross-correlator. One such an estimator integrates the product of received signal with the known transmitted sequence. The resulting timing offset estimate $\hat{\tau}$ is that which maximizes the correlator's output, viz.

$$\hat{\tau} = \arg\max_{\tilde{\tau}} \left\{ \int_0^{T_0} r^*(t-\tilde{\tau}) s(t) dt \right\} \tag{4.4}$$

where as in Section 3.1 T_0 is the observation time and $\tilde{\tau}$ is the trial value of the timing offset. For discrete-time systems, this is equivalent to finding the index which maximizes the cross-correlator output. Fractional timing offsets can be resolved by performing parabolic interpolation between samples. For a sampling rate $1/T_s$ and an observation period of L_0 samples, (4.4) can be rewritten in its discrete form as

$$\hat{\tau}_t = \arg\max_{\tilde{\tau}} \left\{ \sum_{k=0}^{L_0-1} r^*(kT_s - \tilde{\tau}) s(kT_s) \right\} \tag{4.5}$$

Applying this estimator for OFDM is trivial as each frequency-domain symbol set is transformed into a unique time series via the inverse DFT. This is not the case with OFDM/OQAM as the symbols overlap in time, an artifact of the pulse-shaping filter $g(t)$ as well as the subsequent I/Q offset by $T/2$ seconds. Consequently, estimators based on temporal correlation for OFDM/OQAM are considerably more difficult.

This estimation technique has several drawbacks. To begin with, any residual carrier frequency offset will cause the received sequence to rotate its phase, reducing the output of the cross-correlator. This makes (4.5) particularly sensitive to ν when L_0 is large. In fact, if $\nu = 2\pi/L_0$ then the phase of $r(t)$ is reversed through the period L_0 and the cross-correlator output will equal zero on average, viz.

$$E\left\{ \sum_{k=0}^{L_0-1} \left[s^*(kT_s) e^{-j2\pi k/L_0} + n(kT_s) \right] s(kT_s) \right\} = 0$$

Additionally, the introduction of narrowband interference (when $\psi(t) \neq 0$) corrupts the temporal correlator output and increases the estimator's variance. One potential solution is to apply a filter before the input to the transform as to eliminate this interference; however, this requires more processing than is desirable. Furthermore, multi-carrier systems are designed to run in the frequency domain such that data subcarriers which do not overlap the interference are orthogonal to it. In effect, pre-filtering the signal is only to the benefit of the initial estimation stage and not for demodulation. This chapter later introduces several novel estimators for the carrier and timing offset reference parameters, as well as equalization due to multi-path fading. The performance is characterized in channels undergoing additive white noise, multi-path fading, and interference.

4.2.2 Frequency-Domain Estimation

With the potential of narrowband interference to corrupt temporal estimation techniques, it is logical to move to estimation methods which operate in the frequency domain. Consider the output of of the m^{th} down-sampler as

$$y_m(t) = r(t) e^{-j(\omega_m t)} * g(-t) \tag{4.6}$$

where the received signal is a delayed version of the transmitted signal corrupted only by noise such that $r(t) = s(t-\tau) + n(t)$. When converted into a discretely sampled time series, the output of the m^{th} down-sampler after running through a matched filter $g(t)$ over the time span $L_1 \leq i \leq L_2$ is given by

$$y_m(kT_s) = \sum_{i=L_1}^{L_2} r\big((k-i)T_s\big) e^{-j\left(\omega_m (k-i) T_s\right)} g\left(iT_s\right) \tag{4.7}$$

When $\tau = 0$ and the sampling index k is taken at the maximum eye opening of the matched-filter output, the output $y_m(kT_s) \approx a_m(p)$ may be fed into the detector for demodulation. If a slight timing offset exists the distortion due to timing mismatch by not sampling at the peak of the eye is negligible and that the channelized output can be approximated as

$$y_m(kT_s) \approx a_m(p) e^{j2\pi\tau m/M} + \tilde{n}(kT_s) \tag{4.8}$$

where $\tilde{n}(kT_s)$ is the in-band noise term after matched-filtering. As such, the complex gain of the channel may be expressed as $G_m = Y_m/S_m$ where the abbreviations $Y_m = y_m(kT_s)$ and $S_m = a_m(p)$ are used for convenience. Clearly τ corresponds to the frequency of the sinusoid in (4.8), and the phase of the complex gain terms G_m will vary linearly with m. Therefore an estimator for the timing offset τ may be derived by using the maximum likelihood estimator for the frequency of this sinusoid. This is accomplished by by summing the phase differences of the complex gain terms G_m over adjacent subcarriers, viz.

$$\hat{\tau}_f = \frac{M}{2\pi} \arg\left\{ \sum_{m=0}^{M-2} G_{m+1} G_m^* \right\} \tag{4.9}$$

Notice that the above expression can be computed for any non-integer value of τ, while the time-domain correlation estimator $\hat{\tau}_t$ in (4.5) required parabolic interpolation for estimating fractional sample delays.

This particular estimator has several drawbacks in its applicability to OFDM/OQAM systems. For instance, $\hat{\tau}_f$ neglects the effects of any carrier frequency offset which can cause substantial distortion in the output of the channelizer. In this situation, the approximation given by (4.8) breaks down due to inter-channel interference. The fact that the in-phase and quadrature components of $s(t)$ are delayed from one another causes further problems. Additionally, the assumption that distortion due to a timing mismatch won't hold for even moderate timing offsets. The use of $\hat{\tau}$ is convenient for OFDM where $g(t) = 1$ over the duration of the symbol plus its cyclic extension, and therefore the expression in (4.8) is exact. This is not the case with OFDM/OQAM where distortion due to a mismatch in sample timing is an artifact of inter-symbol interference. As Section 4.4.3 will later show, this problem can be circumvented by repeating the OFDM/OQAM symbol pilots to force a flat temporal response, mimicking the behavior of OFDM.

4.2.3 Timing Correction in OFDM/OQAM

Once the timing offset has been properly estimated, it is necessary to correct this error by compensating for the sampling mismatch at the receiver. In OFDM systems this is accomplished by simply applying a linear complex phase rotation for each subcarrier after the transform. Orthogonality is preserved so long as the transform is taken within the duration of the cyclic prefix window (in other words, the buffered samples do not overlap symbols). Unfortunately for OFDM/OQAM this is not a sufficient condition as the envelope of power changes over the duration of one symbol, thus, timing offsets cannot be perfectly corrected by applying complex phase rotation after taking the transform. Consider a single narrowband channel in which a root raised-cosine pulse is applied to a BPSK signal. If the timing of the matched-filter output is misaligned then the receiver will observe inter-symbol interference at the filter output. This can be corrected before down-sampling by applying an interpolating filter which can effectively all but eliminate ISI from the filter output.

This technique was presented in Section 3.2 with the use of a polyphase filterbank interpolator. Due to the frequency overlap of OFDM/OQAM, once the signal has been down-sampled, any residual timing offset will result in not only inter-symbol interference, but inter-channel interference as well. This prevents the use of any simple interpolating filter at the output of the channelizer to easily perform post-filtering timing offset correction.

Consider a simple non-recursive filter which is comprised of an array coefficients and a buffer of input samples of the same length. The samples are loaded in the buffer from the right one at a time. Each time the sample is loaded, the filter output is computed by taking the vector dot product between the buffer and the coefficients array. Section 3.2 has demonstrated that timing recovery of a narrowband signal requires oversampling by only the Nyquist rate; the resulting symbols can be recovered by choosing the appropriate filter in a bank to compute the interpolants between available sample points. Furthermore, the timing could be recovered by controlling the input data flow into the the filter buffer and selecting the appropriate index to a bank of filter coefficients interpolate between available samples. Additionally, Section 3.2.5 demonstrated that a bank of just 32 filters is sufficient to provide interpolation without a significant increase in inter-symbol interference. Conveniently, the interpolating filter also doubles as the matched filter. This method is equivalent to, but more computationally efficient that just grossly over-sampling the signal and simply choosing the optimal matched-filter output sample.

Consider, however, that in an OFDM/OQAM multi-carrier system, each narrowband carrier is already oversampled by a factor of M due to the nature of the channelizer. This suggests that if the system contains a sufficient number of subcarriers, the filterbank channelizer itself can be used for interpolation to correct for timing offsets. The DFT analysis filterbank operates similarly to the filter banks of Section 3.2; however, the input sequence is sub-sampled by a factor of M into M buffers. The input buffer for each filter is effectively a set of parallel down-samplers, storing the interleaved samples in consecutive arrays. But as depicted in Figure 2.4, the samples have already been loaded into the commutator which are internally buffered. This can easily be solved; if the samples have been loaded into the wrong buffer by virtue of the input commutator having started sampling at the wrong time, the alignment can be corrected by permuting the proper buffer to its corresponding filter. Figure 4.2 depicts the modified DFT filterbank allowing arbitrary alignment with the sub-sampled input time series.

4.3 Carrier Recovery

As stated in Section 3.3, digital communications systems are limited by carrier offset relative to their symbol rates. In the case of narrowband systems this carrier offset is commensurate of the signal's occupied bandwidth. For multi-carrier systems, however, the occupied bandwidth is much higher than the symbol rate as many subcarriers are transmitted in parallel. Consequently, multichannel communications systems are much more sensitive to carrier offsets than single channel systems. This section investigates several estimators for ν, the carrier frequency offset reference parameter, in both the time and frequency domains. My intent here is to introduce estimators for ν commonly used with synchronizing traditional OFDM signals and discuss their viability for OFDM/OQAM systems.

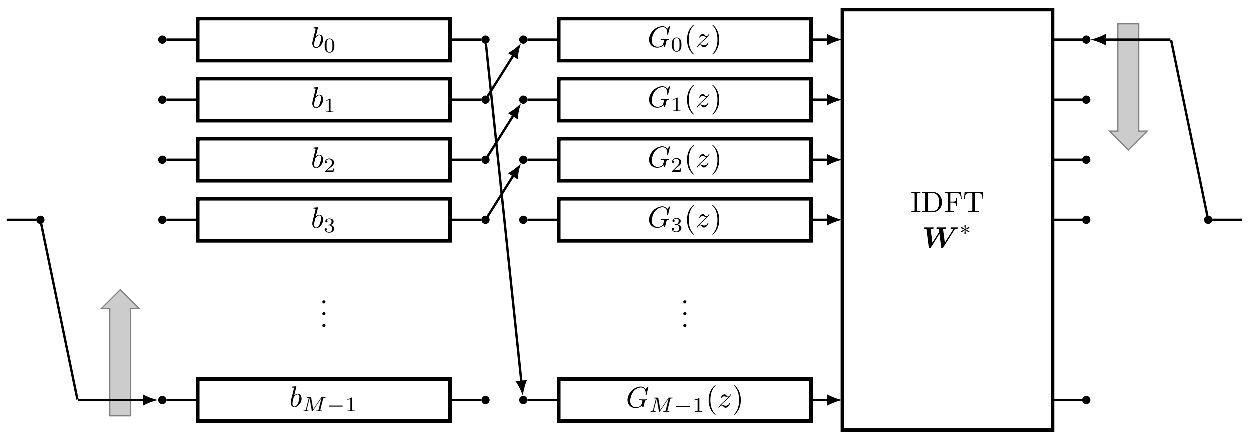


Figure 4.2: Modified DFT polyphase filterbank channelizer system diagram. The filter coefficients $G_m(z)$ are separated from their buffers b_m allowing arbitrary sample alignment. A shift of one sample is shown here.

4.3.1 Time-domain estimation

Carrier recovery in OFDM systems is typically accomplished by taking advantage of a certain property of inverse DFTs in which disabling all but every N^{th} subcarrier causes N repetitions in the output of the transform. Effectively, this amounts to multiplying the spectrum with a sequence of equally spaced Dirac delta functions and thus causes repetitions in the time domain over the duration of one symbol. As such the transmitted signal can be represented as

$$s(t) = s(t-\zeta) \tag{4.10}$$

where ζ is the time between repetitions, equal to the inverse of the bandwidth between enabled subcarriers. If a carrier frequency offset ν is present, then the received signal will rotate $\nu\zeta$ radians between repetitions, thus $r(t) = r(t-\zeta)e^{j\nu\zeta t}$. Sampling the received signal at the interval T_s gives the relationship

$$r(kT_s) = r\big((k-P)T_s\big)e^{j\nu P} \tag{4.11}$$

where $P = \zeta T_s$ is the delay between repetitions in samples. The cyclostationary properties of the transmitted signal can be exploited to derive an estimator for ν. Regardless of the time index k, the phase difference between $r(kT_s)$ and $r\big((k-P)T_s\big)$ is constant and implies that $E\big\{r(kT_s)r^*\big((k-P)T_s\big)\big\} = \nu P$. The resulting estimator for carrier frequency offset is

$$\hat{\nu}_t = \frac{1}{P}\arg\left\{\sum_{k=0}^{L_0-1} r(kT_s)r^*\Big((k-P)T_s\Big)\right\} \tag{4.12}$$

The above estimator has a phase ambiguity of π/P; if the phase difference is π radians over P samples, the estimator cannot differentiate between $\nu = \pi/P$ and $\nu = -\pi/P$. As a result, the range for $\hat{\nu}_t$ is $-\pi/P \le \nu \le \pi/P$. Clearly, the estimator's range can be improved by decreasing the delay between repetitions with the drawback of concentrating more energy into fewer subcarriers.

A more reliable estimator can be derived which exploits knowledge of the transmitted signal $s(t)$. This implies that timing information has been recovered—recall that $\hat{\tau}_t$ in Section 4.2.1 made the assumption that carrier frequency information had been recovered. Conveniently, $\hat{\nu}_t$ can be made blind, independent of the transmitted symbol sequence as well as timing information and carrier phase offset.

As before with OFDM/OQAM, however, the consequence of applying a band-limiting filter to each subcarrier is that symbols overlap in time, and therefore (4.10) will only hold true after as many symbols as the delay of the prototype filter are transmitted. Furthermore, the introduction of narrowband interference can significantly degrade the performance of $\hat{\nu}_t$ as discussed with $\hat{\tau}_t$.

4.3.2 Frequency-domain estimation

As with $\hat{\tau}$, the potential for narrowband interference to corrupt temporal estimation techniques motivates the estimation of ν in the frequency domain. Assuming optimal timing at the receiver, the complex gain of the channel is $G_m(p) = Y_m(p)/a_m(p)$ where $Y_m(p)$ is the output of the m^{th} down-sampler at symbol index p. For small carrier frequency offsets it may be assumed that $\nu \ll 1/T$ and that the distortion due to inter-channel interference from matched-filter misalignment is negligible. The resulting carrier offset will cause the complex channel gain $G_m(p)$ to rotate with successive symbols equal to ν/M. An estimator for ν can be derived completely in the frequency domain by averaging this phase rotation between successive symbols for all subcarriers, viz.

$$\hat{\nu}_f = \frac{1}{M} \arg\left\{ \sum_{m=0}^{M-1} G_m(p) G_m^*(p-1) \right\} \tag{4.13}$$

As with $\hat{\nu}_t$, this estimator has a limited range of operation. Because successive symbols are spaced M samples apart (neglecting any cyclic extension), $\hat{\nu}_f$ has a theoretical range $-\pi/M \leq \nu \leq \pi/M$. Significant frequency shifts, however, can cause distortion in the matched-filter output which further limits the practical range of $\hat{\nu}_f$.

4.4 Joint Timing and Carrier Recovery

The preceding sections have demonstrated that while independent estimators for τ and ν are available, the performance of each is conditional upon knowledge of the other. Interestingly, the best estimator for timing occurs in the frequency domain, while the best estimator for carrier frequency offset occurs in the time domain. This section introduces a joint estimator for τ and ν for OFDM/OQAM signals by exploiting cyclostationary properties within a novel preamble sequence. These new estimators are compared to several others which have recently been proposed, and demonstrate a significant improvement in wireless channels exhibiting narrowband interference. The performance is characterized in terms of error variance in a variety of channel environments and is compared to the Cramér-Rao lower bounds.

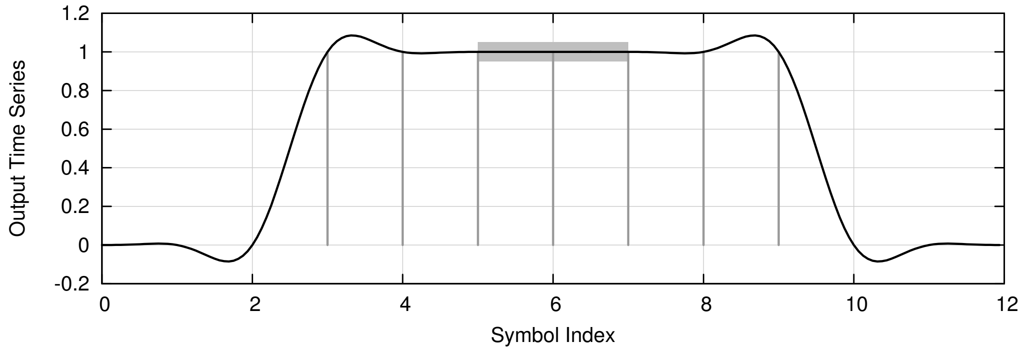


Figure 4.3: OFDM/OQAM filter preamble for a delay of $m = 3$ symbols and an excess bandwidth factor of $\beta = 0.9$ for $R = 2$ repetitions using the r-Kaiser filter prototype. The region of repetition is highlighted.

4.4.1 Temporal Repetition in OFDM/OQAM

Section 4.3 has demonstrated that carrier recovery of multi-channel communications signals often exploits temporal repetitions in the signal and that OFDM can synthesize this response simply by disabling all but several equally-spaced subcarriers. With OFDM/OQAM, however, the band-limiting filter causes the symbols to overlap in time and thus (4.10) no longer holds. Furthermore, timing recovery in the frequency domain becomes impractical for timing offsets of half a symbol period where the output of the down-sampler is corrupted by inter-symbol interference.

These limitations, however, can be circumvented if the time series were permitted to repeat. While trivial with OFDM, this is indeed also possible with OFDM/OQAM by pushing the same symbol repeatedly through the synthesizer filter bank. Figure 4.3 demonstrates the effect of repeating the same symbol into a square-root Nyquist filter at the transmitter, and match-filtered at the receiver. Notice that not only does the time series repeat after $2m$ symbols—the the full delay of both the transmit and receive filters—but that the magnitude response is nearly flat over its duration. This is an artifact of the allowing the excess bandwidth factor of the filter to approach 100%. For R total symbol repetitions, the transmitter needs to push $R + 2m - 1$ symbols through the filter. These repetitions allow estimators based on (4.5) and (4.12) to be used for OFDM/OQAM signals.

4.4.2 Existing Estimators

Several joint estimators proposed in the literature take advantage of the repetition property described in the previous section. Fusco *et al.* in [33] derive a least-squares (LS) estimator by exploiting the known structure of the training sequence which leads to the following joint estimators for ν and τ:

$$\hat{\tau}_{LS} = \arg\max_{\tilde{\tau}} \left\{ 2\left|R(\tilde{\tau})\right| - Q_1(\tilde{\tau}) - Q_2(\tilde{\tau}) \right\} \tag{4.14}$$

$$\hat{\nu}_{LS} = \frac{1}{PT_s} \arg\{R(\hat{\tau}_{LS})\} \tag{4.15}$$

with

$$R(\tilde{\tau}) \triangleq \sum_{k=(2m-1)M}^{(R+2m-1)M-1} r^*(kT_s + \tilde{\tau}) r(kT_s + PT_s + \tilde{\tau}) \tag{4.16}$$

and

$$Q_i(\tilde{\tau}) \triangleq \sum_{k=(2m-1)M}^{(R+2m-1)M-1} \left| r\big(kT_s + (i-1)PT_s\tilde{\tau}\big) \right|^2 \tag{4.17}$$

where P is the time delay in samples between repetitions and is equal to M (the number of subcarriers) in these estimators. The authors go on to derive a modified least-squares (MLS) estimator by dividing the timing metric in (4.14) by the term $Q(\tilde{\tau}) \triangleq Q_1(\tilde{\tau}) + Q_2(\tilde{\tau})$:

$$\hat{\tau}_{MLS} = \arg\max_{\tilde{\tau}} \left\{ \frac{|R(\tilde{\tau})|}{Q(\tilde{\tau})} \right\} \tag{4.18}$$

$$\hat{\nu}_{MLS} = \frac{1}{PT_s} \arg\{R(\hat{\tau}_{MLS})\} \tag{4.19}$$

Notice that both of these joint estimators are blind and make no assumption of knowledge of the training sequence $s(kT_s)$. Similarly, Tonello and Rossi in [90] propose a similar least-squares blind joint estimator which takes advantage of the periodicity in the training burst

$$\hat{\tau}_{TR1} = \arg\max_{\tilde{\tau}} \left\{ \frac{|R(\tilde{\tau})|^2}{Q_2(\tilde{\tau})^2} \right\} \tag{4.20}$$

$$\hat{\nu}_{TR1} = \frac{1}{PT_s} \arg\{R(\hat{\tau}_{TR1})\} \tag{4.21}$$

Tonello and Rossi also proposed a set of joint estimators which exploit knowledge of the transmitted training burst $s(kT_s)$ [90],

$$\hat{\tau}_{TR2} = \arg\max_{\tilde{\tau}} \left\{ \frac{|S(\tilde{\tau})|^2}{T(\tilde{\tau})} \right\} \tag{4.22}$$

$$\hat{\nu}_{TR2} = \frac{1}{PT_s} \arg\{S(\hat{\tau}_{TR2})\} \tag{4.23}$$

where

$$S(\tilde{\tau}) = \sum_{k=(2m-1)M}^{(R+2m-1)M-1} r^*(kT_s - \tilde{\tau}) r(kT_s + PT_s - \tilde{\tau}) s^*(kT_s + PT_s) \tag{4.24}$$

and

$$T(\tilde{\tau}) = \sum_{k=(2m-1)M}^{(R+2m-1)M-1} \left| s(kT_s) \right|^2 \left| s(kT_s + PT_s) \right|^2 \tag{4.25}$$

Furthermore, blind estimators are proposed in [11] by exploiting the unconjugate cyclostationarity of the received OFDM/OQAM, while in [34] it is shown that accurate estimators for ν can be obtained using both the conjugate and the unconjugate cyclostationarity properties of the received OFDM/OQAM signal. Additionally, a maximum likelihood estimator has been proposed in [32], but all of these estimators are much more computationally complex than those presented here.

4.4.3 Proposed Estimators

All of the joint estimators in the preceding section exploited temporal correlations of the time series before the down-sampler. It is desirable in dynamic spectrum environments to estimate the reference parameters in the frequency domain as to remove interfering signals from the estimates. Frequency-domain estimators in Section 4.2 and Section 4.3, however, make strong assumptions about the underlying signal structure. In particular, distortion due to timing and carrier phase offsets are negligible after the down-sampler. Saltzberg, demonstrated in [83] that even a slight timing offset causes both inter-symbol and inter-channel interference, a carrier phase offset results in cross-talk between adjacent subcarriers' in-phase and quadrature channels. However, these distortions can be mitigated or even eliminated altogether by devising a special preamble structure for initial acquisition. Inter-symbol interference can be eliminated by permitting the input symbols to repeat as in Figure 4.3. Furthermore, inter-channel interference can be completely removed if all odd subcarriers are disabled, which effectively yields two repetitions per symbol period. Finally, the I/Q mismatch is handled by disabling the quadrature component and only transmitting symbols on the in-phase channel. This preamble sequence is only several symbols long and only needs to be transmitted at the beginning of each data frame. Once acquired, the receiver can track to any residual errors using pilot symbols.

Conveniently, the polyphase OFDM/OQAM receiver channelizer uses two separate yet identical DFT filterbanks separated in time by $M/2$ (half a symbol period), as depicted in Figure 4.1. Considering that data are only transmitted on the even subcarriers, the time series will repeat every $P = M/2$ samples (twice a symbol period). As such, the output of the two analysis filterbanks will reflect sufficient information about the channel impairments to yield estimators for ν and τ. Specifically, complex gain of the channel may be expressed as $G_m^{(i)} = Y_m^{(i)}/a_m$ where $i = 0$ and $i = 1$ correspond to the upper and lower analysis filterbanks, respectively (see Figure 4.1). As in (4.5), the sinusoidal frequency of the gains over the subcarrier frequencies is proportional to the sample timing offset at the receiver. An appropriate estimator for τ may be derived as

$$\hat{\tau}_G = \frac{M}{4\pi} \arg\left\{ \sum_{m\in\mathcal{P}} G_m^{(0)} G_{m-2}^{*(0)} + G_m^{(1)} G_{m-2}^{*(1)} \right\} \tag{4.26}$$

Notice that the above estimator differs from (4.5) in that only every other subcarrier is used and that the time spacing between the outputs of the down-sampler are $M/2$ instead of M. Notice that unlike the estimators given by Fusco *et al.* and Tonello *et al.*, $\hat{\tau}_G$ is derived in closed-form and does not require an extra search step or parabolic interpolation. The estimator effectively performs convolution of the known sequence with the received sequence in the frequency domain rather than the time domain as done with $\hat{\tau}_t$ and $\hat{\tau}_{TR2}$. This helps to reduce the computational burden of the receiver as well as increasing the timing resolution.

A frequency-domain estimator for ν is derived in a similar to $\hat{\nu}_f$ of (4.13). The observable phase difference between the outputs of the upper and lower analysis filterbanks is proportional to the carrier frequency offset. Because the output of lower analysis bank has a delay of $M/2$ samples from the upper bank, the average phase difference between the two will be proportional to the carrier frequency offset. Ignoring any distortion due to mis-aligned matched filters, ν may be expressed as the accumulated phase differences between $G_m^{(0)}$ and $G_m^{(1)}$, viz.

$$\hat{\nu}_G = \frac{2}{M} \arg\left\{ \sum_{m \in \mathcal{P}} G_m^{(0)} G_m^{*(1)} \right\} \tag{4.27}$$

What is convenient about this estimator, as demonstrated later, is that any interference on a particular subcarrier can be ignored from the set $\mathcal{P}$.

4.4.4 Working Example

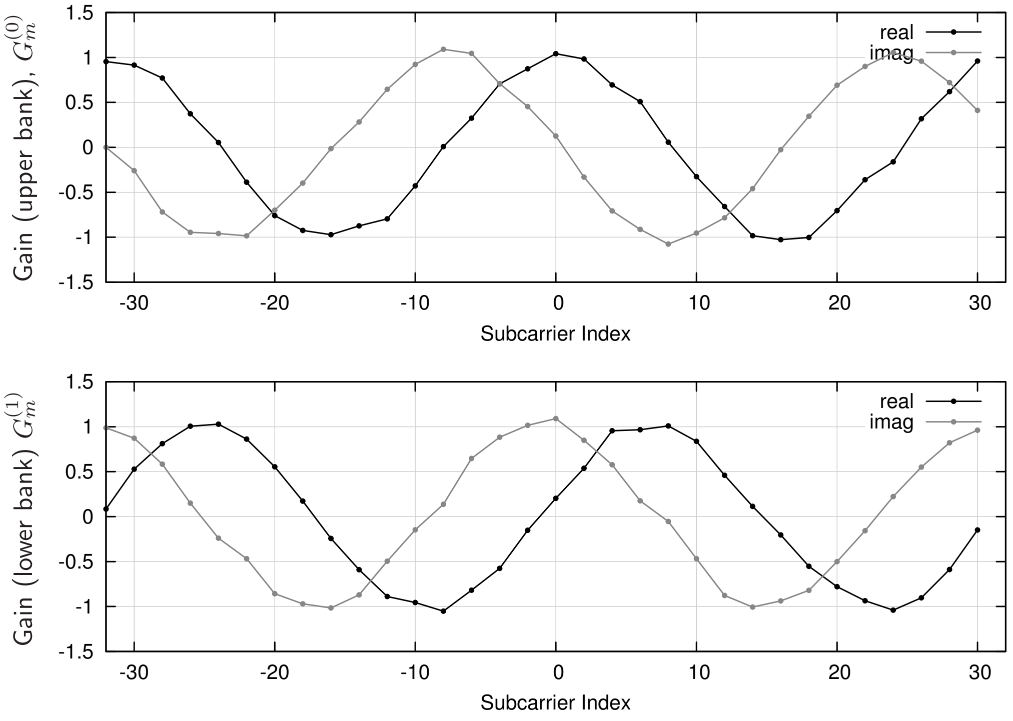
Figure 4.4 gives an example of how the the proposed OFDM/OQAM sample timing and carrier frequency estimators operate. The complex channel gains $G_m^{(0)}$ and $G_m^{(1)}$ are depicted in Figure 4.4(a) in which a sinusoidal response due to a timing offset is clearly visible. Figure 4.4(b) shows the relationship of the phase response of the complex channel gains. The slope of the phase relates to the timing offset while the phase difference is proportional to the carrier frequency offset.

4.5 Numerical Results

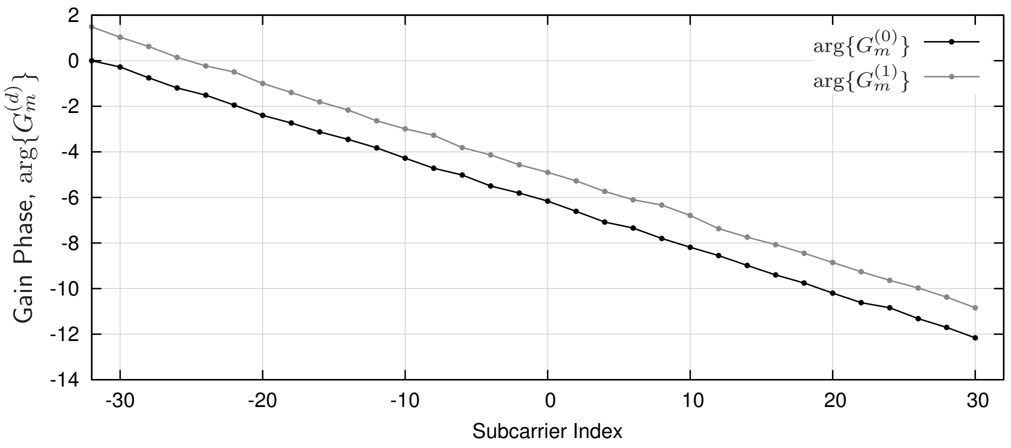
This section compares the performance of the existing estimators for τ and ν given by (4.5), (4.12), given in Section 4.4.2, and proposed in Section 4.4.3 under a variety of channel conditions. The specific channels of interest are those which exhibit additive white Gauss noise (AWGN), multi-path frequency-selective fading, and narrow-band interference. The performance of each estimator is characterized by its root mean-squared error which incorporates both its variance as well as its bias. Statistics were gathered by running 1000 independent simulations across varying signal-to-noise/interference ratios. The values of normalized carrier frequency offset ν, carrier phase offset θ, and sample timing offset τ were uniformly distributed in $[-\pi/M, \pi/M)$, $[-\pi, \pi)$, and $[-M/4, M/4)$, respectively. The OFDM/OQAM subsystem has $M = 64$ subcarriers with the r-Kaiser prototype filter designed in Section 2.5 with a delay of $m = 3$ and an excess bandwidth factor of $\beta = 0.6$. The multi-path channel has been modeled to consist of $N_m + 1 = 5$ independent fading taps with an exponentially decaying power delay profile such that $E\{|h(\ell)|^2\} = Ce^{-2\ell}$, $\ell \in \{0, \ldots, N_m\}$ where C is a constant such that $\sum_{\ell=0}^{N_m} E\{|h(\ell)|^2\} = 1$. The first tap is drawn from a Rice-K distribution with a fading factor of $K = 10$ dB; all subsequent taps are Rayleigh. This model was chosen to exhibit a weak line-of-sight component so as to compare the estimators' performances in a dense multi-path environment. The Cramér-Rao variance bound (CRVB) for $\hat{\tau}$ and $\hat{\nu}$ are derived from [33, (32)] and [33, (33)] respectively as

$$\mathrm{CRVB}(\hat{\tau}) = \frac{M^2 T_s^2}{8\pi^2(\gamma/N_0)\left[\frac{1}{M_u}\sum\limits_{\ell\in\mathcal{A}}\left(\ell - \frac{1}{M_u}\sum\limits_{\ell\in\mathcal{A}}\ell\right)^2\right]} \tag{4.28}$$

$$\mathrm{CRVB}(\hat{\nu}) = \frac{3}{2\pi^2 T_s^2 (\gamma/N_0) L_0^3} \tag{4.29}$$



(a) Complex channel gains



(b) Phase of channel gains

Figure 4.4: OFDM/OQAM preamble synchronization example for $M = 64$ subcarriers with $\tau = 2T_s$, $\nu = 0.02/T_s$, and SNR=20dB.

where γ/N_0 is the received signal-to-noise ratio, $L_0 = RM$ is the number of observed samples. Notice that both (4.28) and (4.29) are inversely proportional to the received signal's SNR, and in particular the variance of $\hat{\tau}$ depends on the location of the enabled subcarriers.

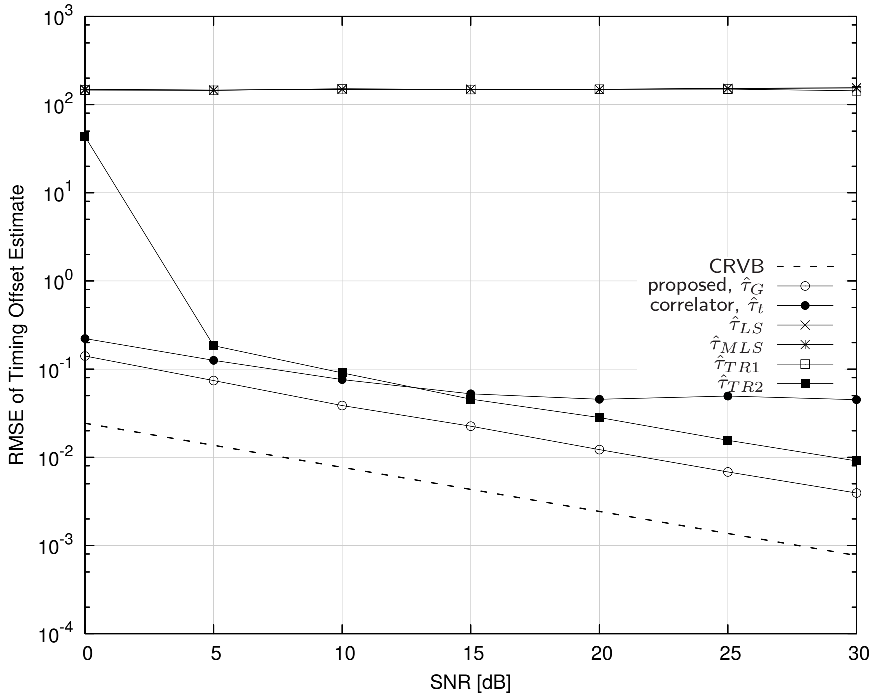
Figure 4.5 demonstrates the performance of the estimators in an AWGN channel. The estimators with the worst timing performance are $\hat{\tau}_{LS}$, $\hat{\tau}_{MLS}$, and $\hat{\tau}_{TR1}$ which do not rely on knowledge of the transmitted symbol set and therefore cannot accurately align the receiver with the transmitter. Interestingly the frequency-domain estimator $\hat{\tau}_G$ out-performs the time-domain estimator $\hat{\tau}_t$ as it does not rely on interpolating between samples. Nearly all the estimators for ν have the same performance with the exception of $\hat{\nu}_G$ which does not rely on time-domain correlation of the received signal. The performance of $\hat{\nu}_G$ flattens out for higher SNR levels as distortion due to matched-filter mismatch tends to dominate.

Figure 4.6 demonstrates the performance of the estimators in a dense multi-path environment. Because the multi-path model exhibits only a weak line-of-sight component, the timing performance for each of the estimators is poor, even for high SNR levels. Estimators based on cross-correlation techniques perform poorly in multi-path environments, particularly for low SNR levels. Interestingly, the proposed $\hat{\tau}_G$ estimator is less affected by multi-path and outperforms the other estimators at 0 dB SNR. This can be explained by the fact that $\hat{\tau}_G$ computes its timing estimate in the frequency domain; frequency components within a multi-path fade naturally contribute less to the estimate of τ and therefore are not as critical to its performance. The carrier frequency offset estimators, however, perform almost identically to the AWGN channel in Figure 4.5 but with a slightly higher variance.

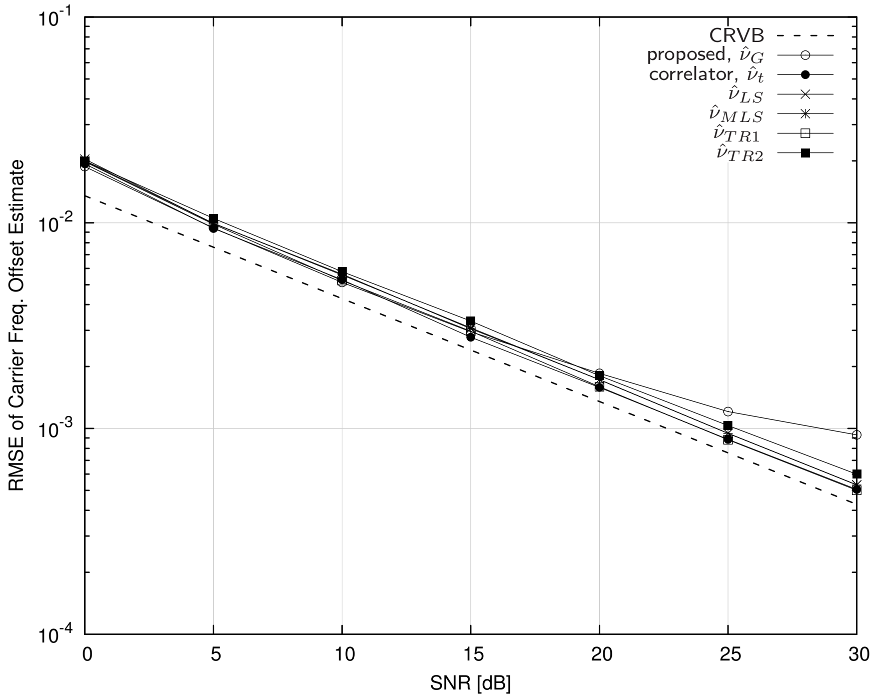
Finally, Figure 4.7 demonstrates the performance of each estimator in the presence of a narrow-band interfering tone. The received signal has a nominal SNR of 20dB before an interfering carrier tone is injected into the receiver. This carrier tone emulates the presence of a narrow-band legacy user near the OFDM/OQAM receiver. It is clear from the figures that the performance of the time-domain estimators is poor when the interfering signal is significantly stronger than the OFDM/OQAM signal. This is an artifact of the cross-correlation methods which cannot remove the interference before estimation. Alternatively, because $\hat{\tau}_G$ and $\hat{\nu}_G$ operate strictly in the frequency-domain, a narrowband tone operating in just one or two subcarrier bands has little contribution to the estimator as a whole. Furthermore, if the frequency of the interference is known *a priori* then the contribution to the corresponding subcarriers to the estimators for τ and ν can be disabled completely. Consequently the proposed estimator can accurately synchronize to the OFDM/OQAM signal even when the SIR level is -20dB, and has a practical limitation of the prototype filter stop-band attenuation.

4.6 Equalization

Suppose that the transmitted signal $s(t)$ passes through an unknown channel filter $h_c(t)$ of length N_m. For narrow-band single-carrier systems where the mean excess delay of the channel is a considerable portion of the symbol period, the receiver would see considerable inter-symbol interference, causing significant distortion to the input of the demodulator and possibly resulting in symbol errors. The receiver's equalizer seeks to remove this inter-symbol interference by designing a filter

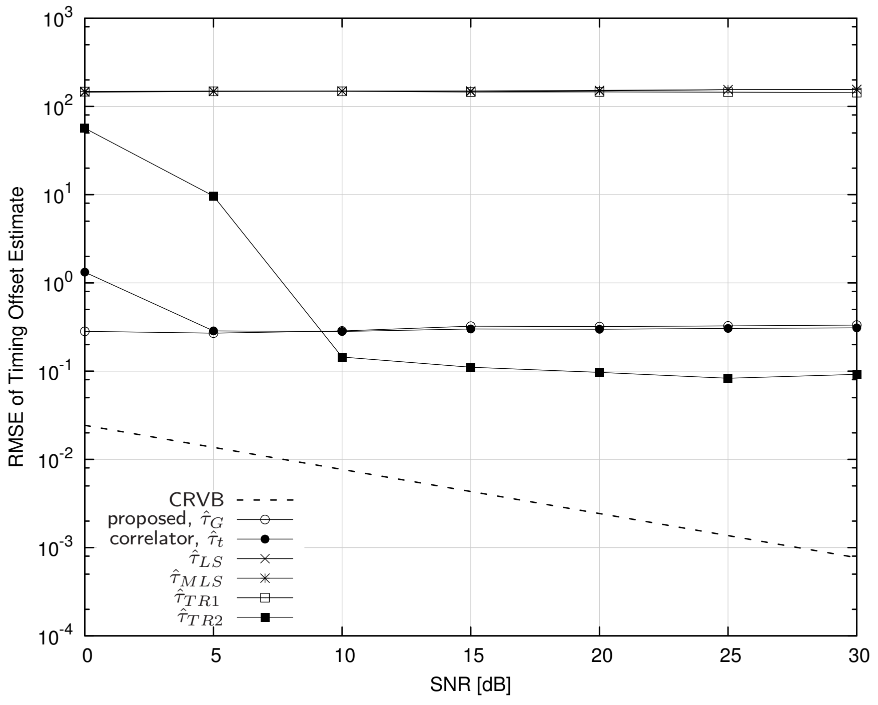


(a) Timing offset estimator, $\hat{\tau}$

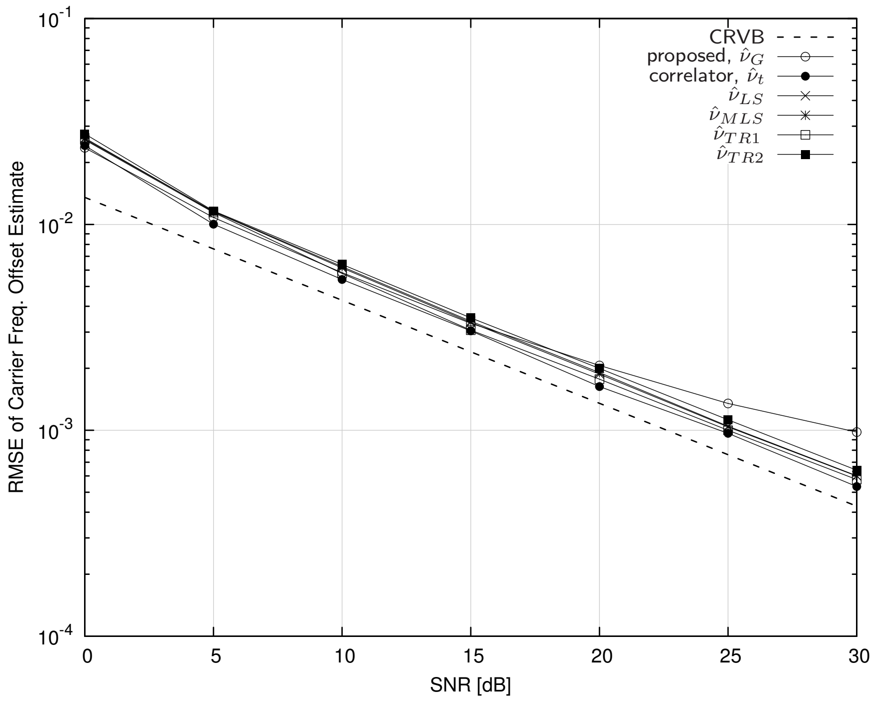


(b) Carrier frequency offset estimator, $\hat{\nu}$

Figure 4.5: OFDM/OQAM parametric estimation in AWGN channels.

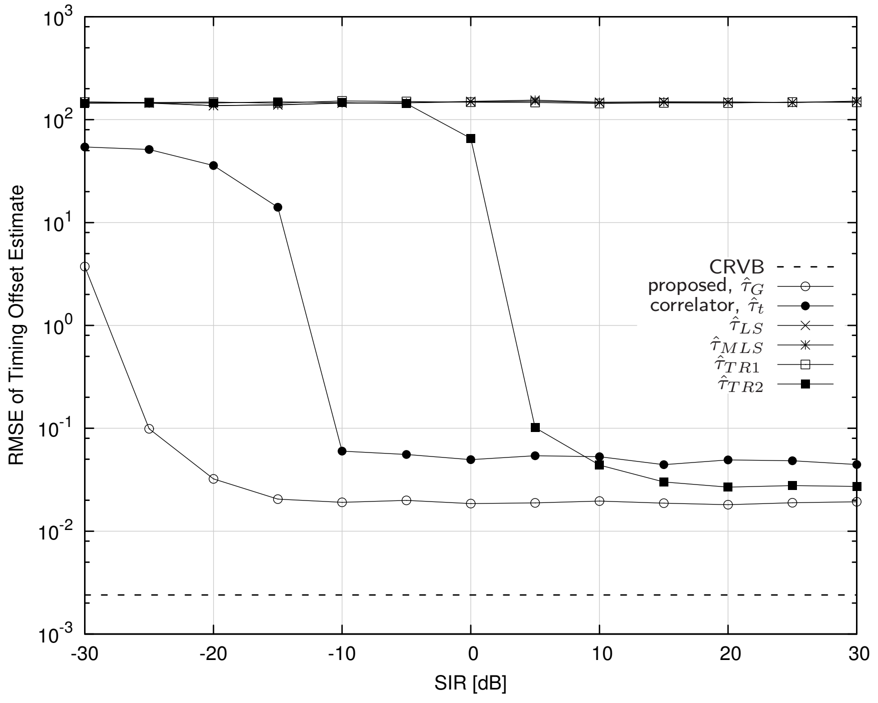


(a) Timing offset estimator, $\hat{\tau}$

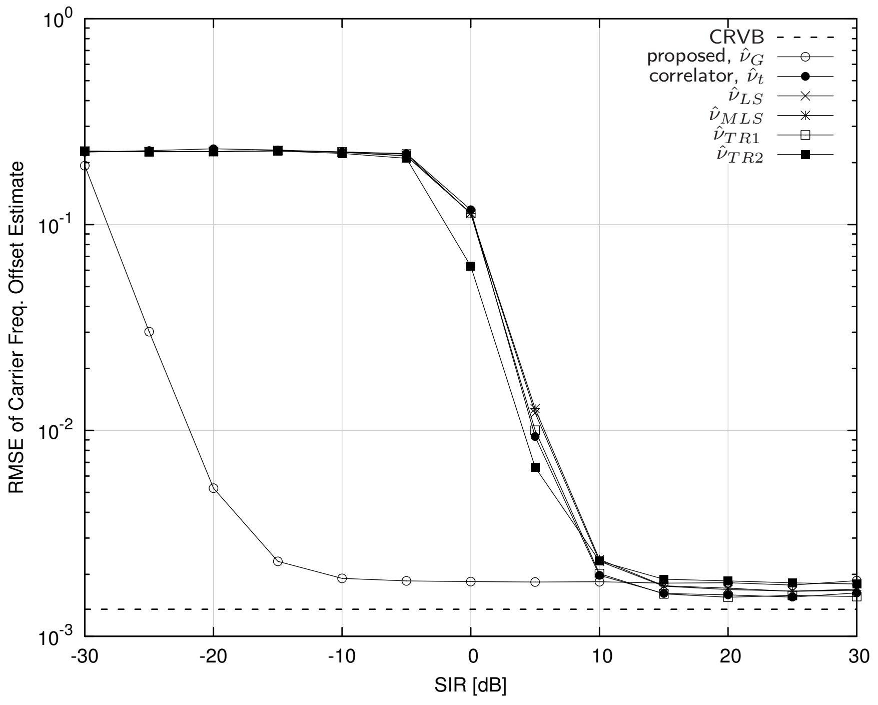


(b) Carrier frequency offset estimator, $\hat{\nu}$

Figure 4.6: OFDM/OQAM parametric estimation in multi-path channels.



(a) Timing offset estimator, $\hat{\tau}$



(b) Carrier frequency offset estimator, $\hat{\nu}$

Figure 4.7: OFDM/OQAM parametric estimation in channels with narrowband interference and a nominal SNR of 20dB.

which minimizes the error between the transmitted sequence and the received data symbols [76, 45].

For multichannel systems the total bandwidth is assumed to have been split into many smaller partitions, extending the symbol duration on each particular subcarrier. As a result, the mean excess delay of the channel impulse response is now much smaller than the symbol period such that the fading on any particular subchannel is approximately flat. This permits equalization to be accomplished after the down-sampler with just one coefficient per channel, a computationally efficient prospect [65]. Conventional multi-channel equalizers take advantage of a set of pilot subcarriers with known amplitude and phase responses. For traditional OFDM systems, the signal's cyclic prefix is exploited to mitigate the effects of sub-optimal symbol timing. Any residual timing error is presented to the down-sampler as simply a linear phase shift with frequency (see Figure 4.4(b)).

Saltzberg has shown how carrier phase error can introduce inter-channel interference in OFDM/OQAM receivers, causing distortion in the signal before demodulation [83]. It is therefore necessary to correct for this carrier phase error due to the multi-path channel without the presence of subcarrier overlap. As before with timing and carrier frequency offset, the preamble sequence is assumed to illuminate only a subset of non-adjacent subcarriers. The equalizer therefore must estimate the complex channel gain corrections for all subcarriers from only those which were transmitted. Effectively, the receiver needs to interpolate between available sample points in the frequency domain to accurately estimate the complex channel gain on those subcarriers which were disabled. While a number of equalizers have been proposed for traditional OFDM systems, almost no literature exists on the subject for OFDM/OQAM. In this section a novel equalizer for OFDM/OQAM is proposed based on the same preamble sequence used to estimate the reference parameters τ and ν in Section 4.4. Its performance is characterized in terms of the mean-squared error difference between the ideal equalizer in which the channel gain is known.

4.6.1 Proposed Equalizer

A number of equalization algorithms exist for multi-rate systems, particularly for OFDM. With traditional OFDM, the gain can be estimated on each subcarrier independently, so long as the carrier frequency offset isn't sufficiently high as to risk inter-channel interference, and that the transform is taken within the duration of the cyclic prefix as to eliminate inter-symbol interference. With OFDM/OQAM, however, ensuring no interference is more difficult. The preamble pattern described in Section 4.4.3, however, eliminates inter-symbol and inter-channel interference by disabling all odd subcarriers on the transmitter, disabling the quadrature channel, and repeating the training pattern several times. As a result the complex gain on those subcarriers which were disabled must be interpolated from those which were enabled. As such, the gain estimate on all pilot subcarriers (the set $\mathcal{P}$) is the ratio of the received symbol to the transmitted symbol as $G_m = Y_m/a_m$ where $m \in \mathcal{P}$. The gain can be approximated on all subcarriers by applying a smoothing function which serves to both interpolate between pilot subcarriers as well as filter out some of the noise in the estimation. The estimate of equalizer gain on all subcarriers is therefore

$$\hat{G}_m = \frac{1}{\bar{w}_m} \sum_{\ell \in \mathcal{P}} w(\ell - m) G_\ell \tag{4.30}$$

where

$$\bar{w}_m = \sum_{\ell \in \mathcal{P}} w(\ell - m) \tag{4.31}$$

is the normalized gain for the m^{th} subcarrier. The remaining task is to choose the appropriate windowing function in (4.30) for estimating G_m.

For estimating the complex gain of a finite-length channel response, its time series is truncated to a fixed number of taps using rectangular window. Such a window with a width N_t samples is defined by

$$w_t(n) = \begin{cases} 1 & 0 \leq n < N_t \\ 0 & \text{else} \end{cases} \tag{4.32}$$

The resulting frequency-domain window is computed by taking the discrete Fourier transform of $w_t(n)$ as

$$w_f(n) = \sum_{k=0}^{N_t - 1} w_t(k) e^{-j2\pi nk/M} \tag{4.33}$$

and can be used for interpolation within (4.30). The expression in (4.33) has no closed-form expression but is similar to the sinc function; due to the discrete nature of the transform the tails of the sinc are aliased back into $w_f(n)$. An approximation to (4.33) can be derived by ignoring the aliased components as

$$w_s(n) = \operatorname{sinc}\left(N_t \frac{n}{M}\right) \exp\left\{-j2\pi \left(\frac{n}{M}\right)\left(\frac{N_t - 1}{2}\right)\right\} \tag{4.34}$$

where the complex exponential term denotes the fact that the temporal window in (4.32) is not centered around the time $n = 0$.

A heuristic window can be derived from the temporal Gauss curve $w_t(n) = \exp(-n^2/2\sigma_t^2)$ which gives preference to multi-path taps closer to $n = 0$ and decays quickly with time. The width of the window is adjusted by σ_t which can be chosen to meet a specific value p at the desired temporal width (e.g. when $n = N_t$). Specifically if $\exp\left\{-n^2/2\sigma_t^2\right\} = p$ when $n = N_t$ then $\sigma_t = N_t/\sqrt{-2\log(p)}$. Letting $p = 0.2$ yields $\sigma_t = 0.55738 N_t$. The corresponding frequency-domain window is simply another Gauss curve

$$w_g(n) = \exp\left\{-\left(n/M\right)^2/2\sigma_f^2\right\} \tag{4.35}$$

where $\sigma_f = 1/M\sigma_t = 1.7941/MN_t$. The justification for this window is that the multi-path components decay with time, and therefore their contribution to the equalizer's complex gain estimates should be less. Samples with a significant time delay are unlikely to contain much energy in the power delay profile but will instead contain noise.

4.6.2 Performance Results

The frequency response due to the channel is simply the Fourier transform of its impulse response $h_c(t)$, viz.

$$H_c(\omega) = \int_{-\infty}^{\infty} h_c(t) e^{-j\omega t} dt \tag{4.36}$$

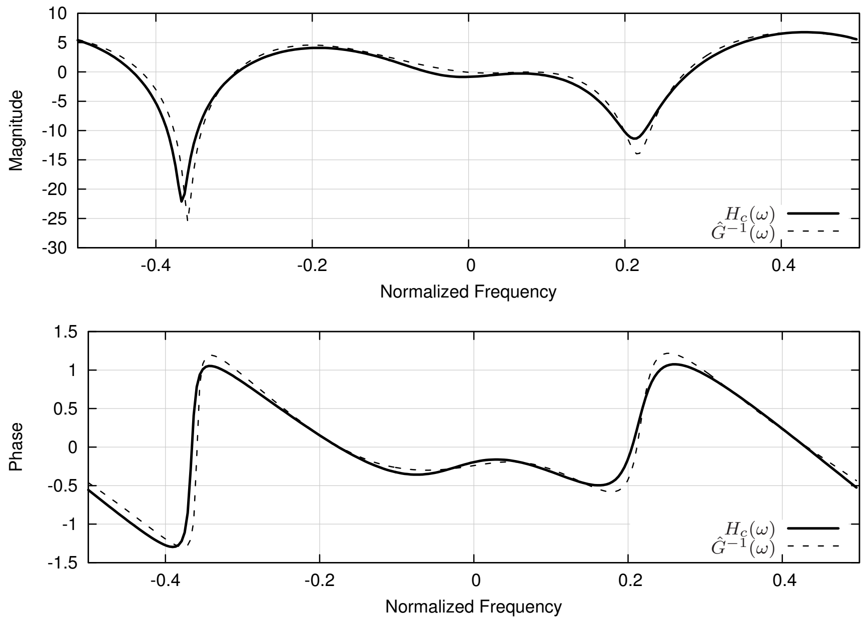


Figure 4.8: OFDM/OQAM equalization example using the $w_f(n)$ window for $M = 256$ subcarriers, $N_m = 4$ paths, $N_t = 6$ samples, and a nominal SNR of 3 dB.

The performance of the equalizer can be quantified by comparing the inverse of the gain estimates $\hat{G}_m$ to the actual channel response $H_c(\omega)$ in terms of mean-squared error. Specifically the error is computed as

$$\text{MSE} = \frac{1}{M} \sum_{m=0}^{M-1} \left[H_c(\omega_m) - \hat{G}^{-1}(\omega_m) \right]^2 \tag{4.37}$$

Figure 4.8 demonstrates the magnitude and phase response of a 4-tap channel as well as the inverse of the gain estimate from the equalizer using $w_f(n)$ in (4.33) with a nominal SNR of just 3 dB. Notice that both the magnitude and phase of the equalizer align well with the channel response suggesting that the proposed equalizer can properly correct for multi-path channel impairments.

Figure 4.9 demonstrates the performance of the equalizer in terms of mean-squared error versus SNR for the three windows introduced in Section 4.6.1. The channel impulse responses were generated as in Section 4.5 with the first tap having a Rice-K distribution with $K = 10$ dB and the remaining taps following a Rayleigh distribution with an exponentially decaying average power. As expected, $w_f(n)$—the true transform of the square window in (4.32)—gives the best results. Interestingly the sinc approximation performs well for low SNR, but eventually hits a performance floor around 20 dB as its RMSE does not improve with SNR. This is a direct consequence of ignoring the aliasing component in the true Fourier transform of (4.32). The Gauss window $w_g(n)$ of (4.35) has a poor performance for low SNR levels but eventually out-performs the $w_s(n)$ window at around 21 dB, even though it is derived completely heuristically.

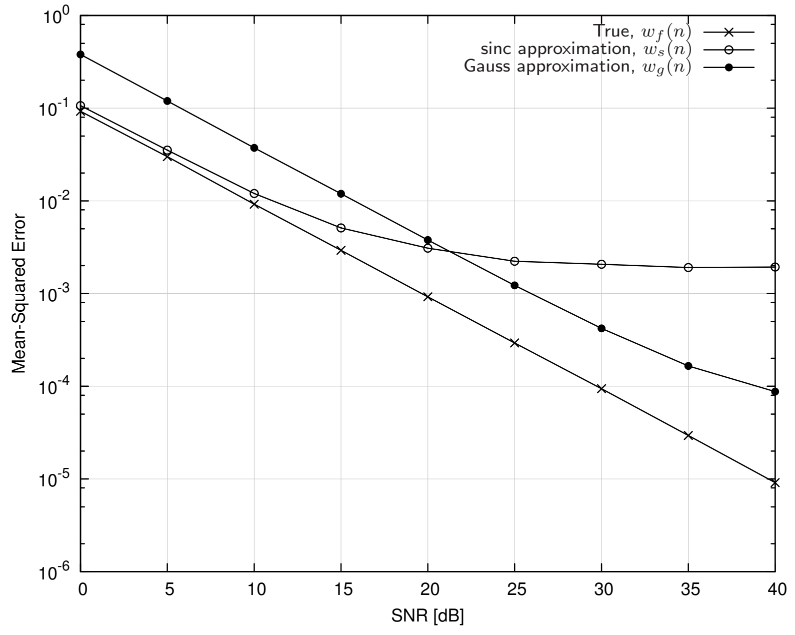


Figure 4.9: OFDM/OQAM equalization mean-squared error vs. SNR for different window types with $M = 64$ subcarriers, $N_m = 4$ paths, $N_t = 6$ samples, and averaged over 1000 trials.

4.7 (Almost) Perfect Reconstruction of Unknown Signals

Perhaps the most appealing promise of cognitive radio is its ability to sense the unknown radio frequency environment and adapt accordingly. CR lends itself well to the detection of unused licensed spectrum through classifying signals emitted from primary user transmitters. This chapter has demonstrated that OFDM/OQAM can be used to perfectly reconstruct the original down-sampled time series (with added noise) under the condition that the transform is time-aligned and any residual carrier frequency and phase offsets are properly compensated. This suggests that a signal of interest received in a contiguous block of subchannels can be also perfectly reconstructed and passed to a signal classification algorithm, thus providing a single receiver solution for both data communication, and interference channelization. This is not possible with traditional OFDM because the addition of the cyclic prefix removes the continuity of the received signal, even though the forward and reverse transforms have a unique one-to-one mapping. This means that the received signal must be passed to a separate channelizer in addition to the OFDM receiver. Furthermore, the stop-band attenuation of OFDM is insufficient for signal separation, and sidelobe suppression techniques such as window tapering [88] and subcarrier injection [77] only apply to the spectrum of the transmitted signal and do not assist in subcarrier isolation in the receiver's down-converter.

Many multichannel communications systems consist of a synthesis filter bank on the transmitter and an analysis filter bank at the receiver. The output of the analysis bank at the receiver can be connected back to back with a synthesis bank, and is said to be a perfect reconstruction system if its output $\hat{x}(n)$ is just a delayed and scaled version of its input $x(n)$ [93, p. 132]. That is,

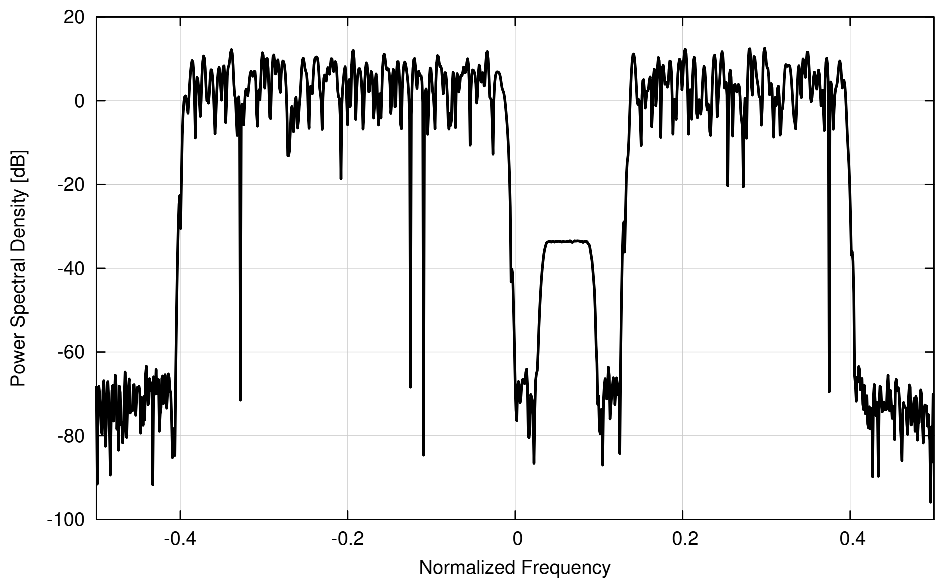
$$\hat{x}(n) = cx(n - n_0) \tag{4.38}$$

The necessary and sufficient conditions for perfect reconstruction are given more formally for any IIR or FIR system in [93, (5.6.6)]. Because OFDM/OQAM uses a matched filter on both the transmitter and receiver, it is capable of satisfying these conditions for (almost) perfect reconstruction. In actuality, this system cannot truly perfectly reconstruct the original time series as DFT filterbanks allow a minimal amount of energy to alias back into the band [24, p. 189]; however, this distortion can be kept negligible with proper filter design strategies. Conveniently, the prototype filter for OFDM/OQAM has relaxed constraints and can be adjusted to reduce stop-band attenuation to any desired level at the cost of increased delay and computational complexity.

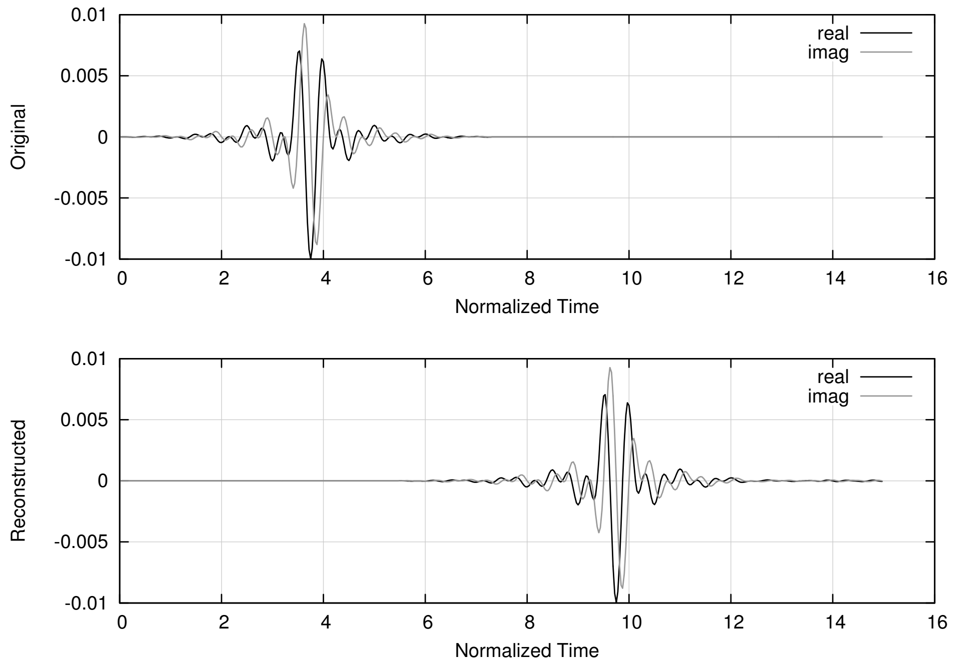
Figure 4.10 depicts an example of an OFDM/OQAM system in which several of the subcarriers are disabled at the transmitter and are used to perfectly reconstruct an interesting signal observed at the receiver. The interfering signal's power spectral density is clearly visible in the notched band at $f = 0.06/F_s$ in Figure 4.10(a). Notice that in Figure 4.10(b) the reconstructed time series of the interfering signal is identical to that of the original, demonstrating that this system indeed satisfies the criteria given by (4.38) and is a near-perfect reconstruction system. The data subcarriers are recovered simultaneously by the same down-sampler without any modification to the receiver.

4.8 Conclusions

This chapter has highlighted the benefits of using multi-carrier communications techniques for dynamic spectrum access, focusing primarily on OFDM/OQAM due to its superior spectral compactness over traditional OFDM. Several new maximum likelihood-based estimators for the timing and carrier offset parameters were introduced which are derived in the frequency domain (after the down-sampler) and are thus robust to interfering signals. The performance of the proposed estimators are comparable to other known estimators in AWGN and multi-path channels but exhibit significantly less variance in channels with narrowband interference, a likely scenario for dynamic spectrum environments. Furthermore, the estimators do not rely on temporal correlation techniques which help to reduce computational complexity. This chapter also demonstrated how the same receiver can be used to demodulate data subcarriers while simultaneously reconstructing the time series of an interfering signal to be used in a classifier.



(a) Power spectral density



(b) Time series

Figure 4.10: OFDM/OQAM perfect reconstruction example where an unknown signal spanning several subchannels is synthesized using an additional polyphase filterbank.

Chapter 5

Resource Management

The tradeoffs between platform programmability and its energy consumption are well-known [78], yet SDR platforms are virtually crippled by their lack of power efficiency; performing baseband processing on a reconfigurable platform compromises its portability due the consequential power demands on its battery. This loss of efficiency through the flexibility of baseband processing has significant implications to the size, weight, and power requirements of the mobile platform [92]. With the onset of software-defined radio, wireless communications engineers have the option to trade physical performance characteristics (e.g. transmit power and occupied bandwidth) for signal processing complexity [69, p. 6]. Recently a number of papers have been published on the subject of cross-layer power management in software-defined and cognitive radios [15, 14, 18]. Dejohnghe *et al.* discuss cross-layer power optimization for embedded, mobile wireless multimedia platforms in [18]. Furthermore, [18] suggest that service demand rates exceed that of technology scaling. To compete with the size, weight, and power gap between software and hardware radios, designs can be modified to enable and exploit flexibility of reconfigurable platforms. As a result, energy savings of up to 40% have been demonstrated under cross-layer optimization [15] through platform-modeling virtualization. Khajeh *et al.* in [57] discuss power management in cognitive radios, demonstrating a 20% overall reduction in energy consumption. Their analysis, however, is focused primarily on the hardware through voltage scaling and is limited to a hypothetical SDR platform without considering dynamic spectrum usage.

While previous efforts have provided significant insight into reducing power consumption for software-defined radios, most results are specific to the application, platform, and wireless environment. Furthermore, energy and power consumption are not the only finite resources in SDR that can significantly affect system performance. Computational bandwidth becomes a major component when the baseband algorithms are implemented in software. One might argue that optimization of source code for the target platform is the solution to data rates throttled by computational bandwidth on software radio platforms. This is partially true; code optimization plays a significant role in the amount of bandwidth the host machine can receive. As such, taking advantage of specific machine architectures (such as machine-level multimedia processor extensions with vector floating-point units) and hardware capabilities (such as graphics processors with highly parallelized architectures) significantly increases the computational bandwidth of the platform. While these technologies continue to close the performance gap between hardware and software platforms, they might not be sufficient for realizing portable reconfigurable radios. For example, the complexity of performing a one-dimensional N-point split-radix FFT was described in Section 2.6.2 as being approximately $(89/18)N \log_2 N$ floating-point operations by the fastest known algorithm [31]. This is the limiting performance factor for an OFDM or OFDM/OQAM transceiver and cannot be significantly improved unless the value of N is reduced. Consequently, the only solution for reducing the complexity of the receiver is not to modify the algorithm itself, but to modify its parametric inputs.

Chapter 1 posited that software-defined radio platforms are much more useful than just re-implementing the same standards on reconfigurable hardware. Such a use only promotes reuse of hardware, and does little for facilitating data communications in opportunistic networks. As such, the dynamic processing complexity of the baseband algorithm as well as its memory usage govern the data rate of the platform. This chapter seeks insight into jointly optimizing radio performance while minimizing resource consumption in dynamic spectrum environments by modifying algorithmic-level parameters in communications systems rather than relying on source code optimization. The analysis is specifically concerned with resources pertaining to energy and power, and those related to computational complexity of the baseband receiver. This chapter introduces several techniques for measuring the dynamic resource consumption of SDR platforms, details several new performance metrics for quantifying the performance of telecommunications links for software receivers, and provides results on DSP benchmarks for baseband processing in order to demonstrate trade-offs between system accuracy, quality of service, and resource consumption. The system is validated with the assessment of real-time software simulation of slowly fading channels and validated by over-the-air wireless reconfiguration in a laboratory environment.

5.1 Resource Consumption in SDR

For a wireless communications link, the energy and power loads can be split into those consumed by the transmitter and the receiver. The transmitter is typically burdened by the radio frequency front end, of which most of the power drain is attributed to the power amplifier. The software-defined receiver, however, is limited by its signal processing hardware. Lien et al. demonstrated that power consumed by the baseband processor increases non-linearly as a function of processor usage [63, Eq. (2)]. Furthermore, power consumption typically increases with a linear relation to sampling frequency. This is apparent in analog converter (ADC) chips which demonstrate a strong relationship between clock frequency and power source current draw [5, TPC 27]. As a result, both the receiver's computational complexity and power consumption are strongly related to its occupied signal bandwidth.

5.1.1 Specific Resources

Radios in general require a variety of resources in order to operate, including power, energy, and spectrum. Software-defined radios in particular require not only more resources than their hardware-defined counterparts, but a different set of resources altogether. Specifically, computational bandwidth and memory are limited resources for software-defined receivers, particularly when operating on embedded platforms. Consequently, the power consumption of a software-defined receiver is strongly tied to its processing requirements and is typically higher than its hardware counterpart. Meyr *et al.* acknowledge that while performance measures for bandwidth and power are clearly defined, there exists no such universally applicable measures of complexity [69, p. 6]. As such, minimizing the processor usage on SDR platforms is the primary concern of the work presented in this chapter.

5.1.2 Performance Metrics

Communication link quality is measured through application-driven metrics, such as bit-error rate (BER), data throughput, and latency. Link quality is contingent upon many controllable factors (modulation scheme, transmit power, forward error-correction coding) and uncontrollable factors (noise power spectral density, interference levels, hardware limitations). Some of these factors in particular can be applied uniquely to the receiver and do not require link-level adaptation. Equalization is a strong example of this; the length of the equalizer directly impacts both the receiver's computational complexity and its ability to reduce inter-symbol interference, thus reducing its error rate. Other factors require radios on both sides of the link to adapt, such as switching modulation and forward error-correction schemes. The relationship between link-level QoS metrics and radio parameters for a given channel state is well-known, but quantification of processing complexity and its impact on these metrics is not. Furthermore, these algorithms' energy consumption significantly depends on the implementation and the target platform's hardware configuration.

For a platform with specific capabilities (e.g. ADC with dynamic sampling frequency, available processors, IF oscillator tunable range, processor architecture) the radio needs to observe the minimum processing requirements to transfer a set of data given the current channel statistics. It is unnecessary for the moment to make any assumptions of the limitations that protocol might restrict the system's adaptation, be this effective isotropic radiated power, occupied spectrum, pulse shaping, etc. as to gain insight as to the upper bound of what sort of processing is necessary. The maximum throughput for a link constrained by transmit power is well-known and is achievable as the occupied bandwidth approaches infinity, viz. [76, (7.1-34)]

$$C = W \log_2 \left(1 + \frac{\bar{P} L_p}{W N_0} \right) \tag{5.1}$$

where $\bar{P}$ is the average transmit power, L_p is the path loss, W is occupied bandwidth, and N_0 is the noise power spectral density. However increasing bandwidth has considerable implications on the computational complexity required of the baseband processor of the receiver as well as the power which it consumes. The receiver's processing energy efficiency can therefore be defined as

$$\eta_e = \frac{C}{P_c} = \frac{W \log_2 \left(1 + \frac{\bar{P} L_P}{W N_0} \right)}{P_c(W)} \tag{5.2}$$

where $P_c(W) \propto W$ is the power required by the processor to recover the bits at the receiver. Note that η_e is measured in bits per Joule and can be though of as the number of usable bits pushed through the channel per Joule of expended energy by the receiver. Although theoretical channel throughput increases with bandwidth for a given average transmit power, the efficiency eventually

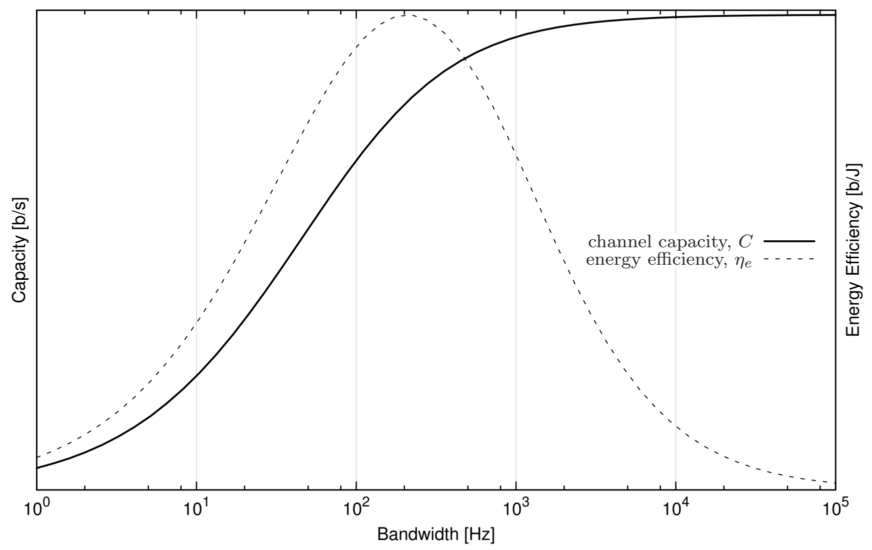


Figure 5.1: Channel throughput [b/s] and energy efficiency [b/J] as a function of bandwidth for a fixed transmit power. Energy efficiency assumes power consumption to be a linear function of occupied bandwidth.

degrades due to an increase in processor complexity and energy consumption at the receiver. The link can therefore sacrifice throughput for bandwidth efficiency and energy consumption. Several publications make reference to measure bits per expended energy as a metric for power efficiency in software-defined radios [15, Fig. 15], [18, Fig. 8]. Figure 5.1 demonstrates this strong relationship between processing efficiency and bandwidth while assuming that the power consumption has a linear relationship to occupied bandwidth. Interestingly, (5.2) relates the efficiency of the link through the transmitter's and receiver's expended power, and suggests that reducing $\bar{P}$ will require the receiver to work harder to achieve a comparable efficiency.

While the throughput of the link may be easily measured, the task of quantifying the amount of energy expended attaining it is more difficult for many software radio platforms. Section 5.3 describes several possible techniques for monitoring resources such as energy, including modeling the characteristics of the hardware using measurable quantities such as processor load and relating it to immeasurable aspects. It is sufficient to say, however, that the limiting factor for the performance of many software radio platforms is its processor capabilities. Consequently, it is more practical to use computational efficiency as a performance metric rather than energy efficiency. The receiver's computational complexity $\mathcal{K}$ is the average number of clock cycles required to decode a single bit of information in a wireless channel. The inverse of $\mathcal{K}$ can be thought of as computational efficiency, measured in bits per clock cycle which relates strongly to η_e, measured in bits per Joule. The following sections make use of $\mathcal{K}$ for analysis.

5.2 Methodology

The relationship described by (5.2) suggests a non-linear relationship between channel quality and processing efficiency. For wireless communications, the optimal operating point for the receiver cannot typically be determined *a priori* without extensive testing. As such it is necessary for the software radio to monitor its own resource consumption online to maintain a quality of service comparable to hardware platforms. This section details the methodology behind the resource-aware SDR platform, introduces energy-quality scalability for wireless systems, and provides an example relating filter length to BER degradation.

5.2.1 Fault Tolerance

Recently, a number of articles have been written which relate power management in wireless communications links to quality of service for multimedia applications. Khajeh *et al.* discuss power management at the hardware level in [57], demonstrating energy savings exceeding 20% while running a 3G WCDMA modem and H.264 video decoder. Zhang *et al.* investigate the power consumption in relation to the rate-distortion curve of decoding video in wireless channels in [97]. When not constrained to any particular standard, Bougard *et al.* in [13] demonstrated though simulation an energy consumption savings of a factor of 5 to 20 while considering system-level parameters only. In all of these examples, the radio was given information about how its link-level parameters affected both is power consumption and its performance. At a higher level, Khajeh *et al.* in [57] acknowledge that not all applications require 100% data correctness; a broad family of fault-tolerant applications exist, particularly in the wireless multimedia realm.

This methodology can be applied to any wireless link, regardless of the underlying application. For the case of fault-intolerant applications, erroneous packets are simply detected with a moderate-length cyclic redundancy check and re-transmitted. The probability of receiving packet errors is kept sufficiently low by adapting link-level parameters (e.g. power, packet length, modulation scheme). One can think of packets that need to be re-transmitted as wasted energy because no useful data was conveyed over the channel. However, one must also recognize that if packets *never* need to be retransmitted, it is likely that the radio link isn't taking full advantage of the capacity of the channel; again in this situation energy is being wasted, either by expending too many resources into each individual bit at the link level, or by over-processing the signal at the receiver.

5.2.2 Energy-Quality Scalability

The notion of energy-quality (E-Q) scalability was introduced in [84] for VLSI systems. Conceptually, the energy consumed through processing can be dramatically reduced while only minimally compromising quality of service. The trade-off between accuracy and energy shows diminishing returns as considerable amount of resources is wasted reducing error below a practical quality tolerance. The optimal solution is one that achieves a sufficient quality measure without expending unnecessary resources attaining it. This is particularly tractable to software-defined radio where performance is typically measured as actual data throughput; users care little about the underlying metrics determining it so long as the radio itself doesn't consume an inordinate amount of

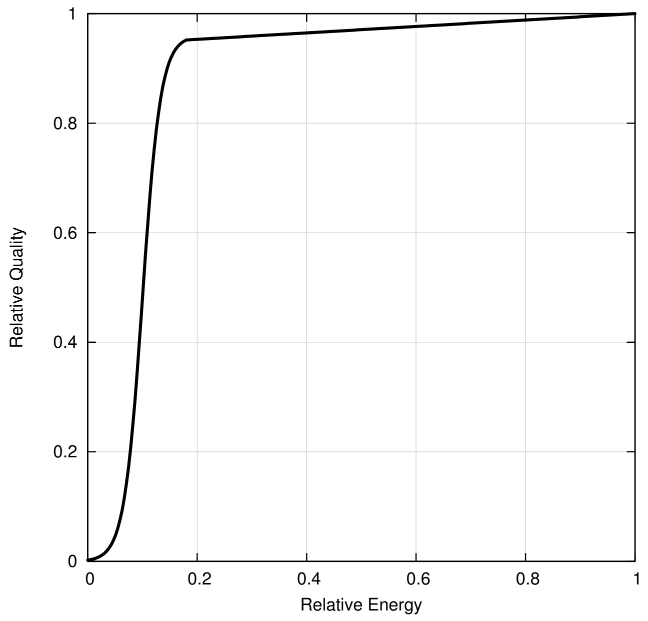


Figure 5.2: Energy-quality scalability relationship.

resources during its operation. From a high-level perspective this primarily amounts to battery life and system latency and translates easily to adaptive radios, particularly adjusting the modulation and forward error-correction schemes to a level "just good enough" while avoiding unnecessary processing. This will be demonstrated by the experiments in Sections 5.6 and 5.6.

As described in [84], the quality of the system (bit error rate, throughput, mean-square error, etc.) is represented by $Q(R)$ as in Figure 5.2 as a function of the computational resources required to attain it. This relationship exhibits a strong non-linear relationship in which the majority of the quality is attained with only a small amount of energy. The remaining quality is achieved with a disproportionate amount of energy. For the majority of wireless applications—at least at the physical layer—the radio transceiver does not need to attain 100% functional correctness. Furthermore, error-correction is typically built into the system, and has contingencies for dealing with faults. As such, with simple algorithmic modifications, the E-Q behavior of the system can be modified such that if the available computational bandwidth is reduced the resulting quality degradation is minimal.

5.2.3 Practical Example: Matched-filter Length

This section provides an example of how a simple modification of a DSP algorithm can translate to significant energy savings. For digital communications systems, filtering consumes a considerable portion of energy in baseband processing. The output $y(n)$ of a finite impulse response (FIR) filter $h(n)$ with M coefficients for an input signal $x(n)$ is defined as $y(n) = \sum_{k=0}^{M-1} x(n-k)h(k)$. Sinha *et al.* discuss only computing the N^{th} most significant filter taps for a FIR filter. From [84],

"...when we analyze the FIR filtering operation from a pure inner product perspective, it simply involves N multiply and accumulate (MAC) cycles. For a desired E-Q behavior, the MAC cycles that contribute most significantly to the output $y(n)$ should be done first."

As a result, considerable energy savings were demonstrated without sacrificing significant output quality. While it is impractical to truncate a filter in a radio communications system (each coefficient plays a significant role in the filter's frequency response), the same analysis can be applied to the *length* of the matched FIR filter before demodulation which contributes to both the inter-symbol interference as well as the number of clock cycles it requires to operate.

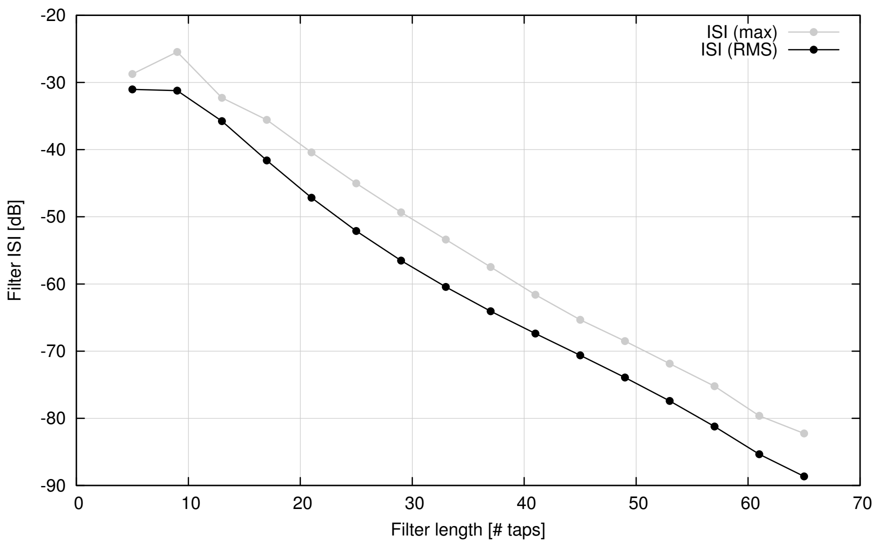
The bit error probability for uncoded QPSK in an AWGN channel with a linear SNR of γ is approximately $P_b = Q(\sqrt{\gamma})$. The error rate increases when inter-symbol interference is introduced as a result of imperfect matched filtering. For an additional ISI ζ, the error rate increases to $\hat{P}_b = Q(\sqrt{\gamma\zeta/(\gamma+\zeta)})$. The ratio $\hat{P}_b/P_b$ gives an indication of the BER degradation due to improper matched filtering. Figure 5.3 depicts the impact of a matched-filter's length on its inter-symbol interference as well as its BER degradation to a nominal SNR of 20dB. Notice in Figure 5.3(b) the non-linear relationship between the length of the filter and its resulting BER degradation as well as its similar shape to Figure 5.2. Depending upon the noise power and the modulation scheme used, the filter length can be reduced while still achieving a target BER. This example demonstrates how significant energy savings can be achieved through flexible software design. Furthermore, these energy savings can be accomplished without affecting the policy; only changes at the receiver are necessary.

The following sections will demonstrate that the parameters which affect both the system performance and energy consumption are numerous. Some of which can be adjusted to accommodate existing protocols, yet some require a more flexible policy to operate.

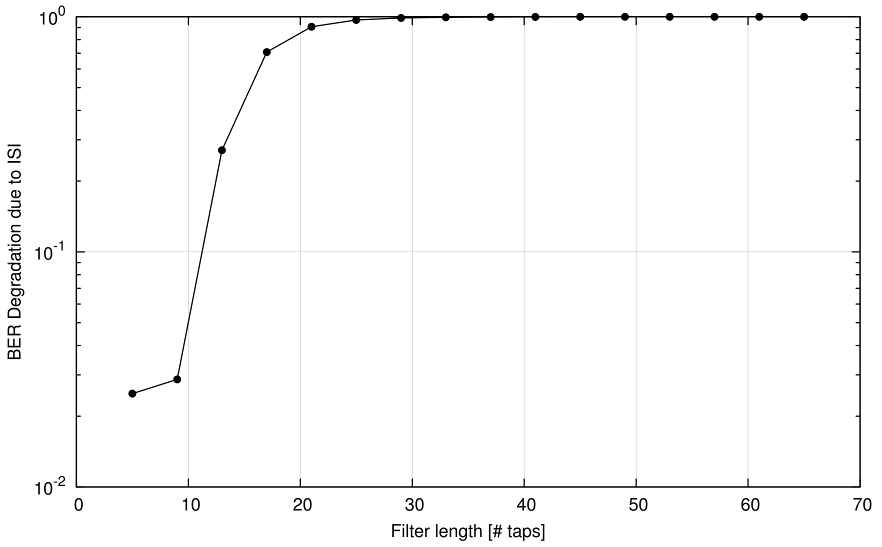
5.2.4 The Resource-aware SDR Platform

This chapter proposes to extend the typical SDR model to incorporate real-time resource monitoring modules necessary for reducing energy consumption. SDR designs consist primarily of a set of inter-connected processing components perhaps running on independent processors. The scenario is formalized as a classic non-linear optimization problem with the following axioms:

1. both QoS and resource consumption are measurable quantities and are affected by digital processing blocks' deployment, allocation, and configuration;
2. QoS performance is measured through the system as a whole. Any solution is considered valid so long as the constraints of the system are met;
3. processing complexity and power consumption performance is measured as the sum of independent processors;
4. the relationship between system performance and the amount of resources consumed to attain it is highly non-linear with significant favorability towards energy-quality scalability.



(a) Inter-symbol Interference



(b) BER degradation from ISI, γ =20dB

Figure 5.3: Inter-symbol interference vs. matched filter length using the r-Kaiser filter from Section 2.5 with an excess bandwidth factor $\beta = 0.2$

Therefore the performance monitor and radio control components must be aware of the impact energy-quality trade-off one block has on the entire system performance. For example, power savings through matched filtering (as demonstrated in Section 5.2.3) might degrade the uncoded BER to a point where forward error-correction codes cannot sufficiently recover the packet data. Conversely, there is no need to over-process an otherwise clean signal by applying unnecessary filtering.

5.3 Monitoring Real-time Resource Consumption

Understanding how algorithm adaptions affect resource consumption in processors is necessary for an energy-scalable system to adapt itself to changing environments. This section introduces three methods of monitoring real-time resource consumption on DSP platforms.

5.3.1 Direct Inference

Perhaps the most obvious method for resource monitoring is to measure it directly in real-time as the system runs. As an example, several DSP platforms support real-time power monitoring such as the Lyrtech Small Form Factor SDR development platform [64] which can monitor each processor independently. This involves specialized hardware which samples the voltage and current drawn from the platform's power source and supplies the digitized result to the signal processor. In regards to processor usage, the POSIX kernels offer several possibilities for real-time monitoring of CPU and memory usage through the `rusage` interface in the standard `sys/resource.h` header. The benefit of direct inference is that the system's resources can be measured simultaneously with the algorithms running on the different processors. The algorithms can therefore be adjusted based on this information to manage resource usage. This method provides the most accurate solution, but often platforms do not have such capabilities.

5.3.2 Modeling

While directly measuring the platform's resource consumption is the ideal solution, certain resources such as energy and power are difficult to measure on a number of platforms lacking specialized hardware support. Alternatively, certain resources may be modeled through measurable quantities. For example, [63, 50] demonstrate the strong relationship between CPU usage for general purpose processors and power that the device consumes. Power can be estimated to within just a few percent of the actual consumption by measuring CPU usage alone. The relationship between the two can be characterized offline using a look-up table or low-order polynomial curve-fitting. Because many platforms can monitor their own CPU usage in real-time, online modeling provides a tractable solution to power monitoring and energy management. Furthermore, it has been shown that significant power savings can result from selectively reducing processing to meet a target quality of service for such applications [96]. It is apparent from these results that the same benefits can be realized in SDR platforms which provide flexibility over baseband processing blocks (such as filters, oscillators, and synchronizers). Typical QoS metrics for data radios measure throughput, latency, and error probabilities; all highly sensitive to these baseband processing algorithms. Processing complexity can therefore be minimized such that a predefined QoS is guaranteed while reducing power consumption.

5.3.3 Offline Benchmarks

While Section 5.3.2 generalizes resource management by relating one immeasurable resource to a quantifiable one, modeling in this way provides little information about resource consumption due to specific algorithms and routines. For example, monitoring CPU usage might indicate the amount of resources used by the entire receiver chain, but gives little information about the individual filters, demodulators, and decoders which comprise it. Energy consumption of such processing blocks can be measured offline on the target platform such that they can be efficiently partitioned when the application is to be executed. The energy consumption of each processing block can be either stored in a look-up table or modeled using simple curve-fitting techniques for a variety of typical operating ranges of the algorithm. Table 5.1 gives an example of several DSP benchmarks running on a 867MHz PowerPC.

Signal processing algorithms with flexible implementations must be able to be deployed on a variety of hardware. These implementations, however, might not consume comparable amounts of power on different platforms, thus making modeling more difficult as each platform needs to be individually characterized. For example, the same source code when compiled for a PowerPC architecture can have vastly different binaries from an Intel version. Each processing block might even have different implementations for each architecture; a vector dot product using SIMD instruction extensions for the the PowerPC's AltiVec engine looks vastly different from the Intel's MMX or SSE instruction set. As such, the resource consumption of each must be modeled on each processor independently.

5.4 Framework

This section discusses the design of a software framework to manage radio resources as well as monitor system performance. The framework is built out of necessity to enable the radio to monitor its own resources in real-time, and is designed using a combination of the resource-monitoring methods described in Section 5.3. Chapter 6 will show that allowing an intelligent agent to hook into the framework can provide an engine capable of algorithmic adaptations as unique and unforeseen scenarios present themselves. Only with a resource-monitoring framework can this intelligent engine be enabled.

5.4.1 Online Processor Usage Monitor

It is fairly straightforward to benchmark and profile signal processing algorithms running on a general purpose processor, and the open-source software community has provided a number of tools to accomplish these tasks. To maintain portability, hardware monitoring methods are ignored and analysis is condensed to strictly observing processor usage alone. Furthermore, specifying a hardware-specific interface seems hardly relevant for software-defined radio. Regarding

Table 5.1: DSP benchmarks for various SDR processing algorithms running on a PowerPC platform with an instruction clock rate of 867MHz

```
agc_benchmark
    agc                 :    40000 trials in  35.18 ms (  1.14 M t/s,   762.6 cycles/t)
dotprod_rrrf_benchmark
    dotprod_rrrf_4      :    20000 trials in   1.26 ms ( 15.87 M t/s,    54.6 cycles/t)
    dotprod_rrrf_16     :    20000 trials in   2.70 ms (  7.42 M t/s,   116.9 cycles/t)
    dotprod_rrrf_64     :    20000 trials in   8.27 ms (  2.42 M t/s,   358.4 cycles/t)
    dotprod_rrrf_256    :    20000 trials in  30.57 ms (654.32 k t/s,  1325.0 cycles/t)
iir_filter_benchmark
    iir_filter_4        :    40000 trials in   4.40 ms (  9.09 M t/s,    95.3 cycles/t)
    iir_filter_8        :    40000 trials in   7.62 ms (  5.25 M t/s,   165.2 cycles/t)
    iir_filter_16       :    40000 trials in  13.78 ms (  2.90 M t/s,   298.8 cycles/t)
interp_benchmark
    interp_m2_h8        :    40000 trials in  17.05 ms (  2.35 M t/s,   369.6 cycles/t)
    interp_m4_h16       :    40000 trials in  40.04 ms (998.95 k t/s,   867.9 cycles/t)
    interp_m8_h64       :    40000 trials in 174.95 ms (228.64 k t/s,  3792.0 cycles/t)
decim_benchmark
    decim_m2_h8         :    40000 trials in  11.39 ms (  3.51 M t/s,   246.8 cycles/t)
    decim_m4_h16        :    40000 trials in  17.85 ms (  2.24 M t/s,   386.9 cycles/t)
    decim_m8_h64        :    40000 trials in  39.09 ms (  1.02 M t/s,   847.3 cycles/t)
resamp2_benchmark
    cresamp2_decim_h13  :    40000 trials in  13.32 ms (  3.00 M t/s,   288.7 cycles/t)
    cresamp2_decim_h21  :    40000 trials in  15.86 ms (  2.52 M t/s,   343.8 cycles/t)
    cresamp2_decim_h37  :    40000 trials in  20.78 ms (  1.93 M t/s,   450.3 cycles/t)
    cresamp2_decim_h53  :    40000 trials in  25.75 ms (  1.55 M t/s,   558.1 cycles/t)
fft_benchmark
    fft_16              :    10000 trials in  86.53 ms (115.57 k t/s,  7502.0 cycles/t)
    fft_32              :    10000 trials in 250.54 ms ( 39.91 k t/s, 21721.8 cycles/t)
    fft_64              :    10000 trials in 512.53 ms ( 19.51 k t/s, 44436.4 cycles/t)
    fft_128             :    10000 trials in   1.05  s (  9.55 k t/s, 90739.8 cycles/t)
modem_demodulate_benchmark
    demodulate_psk2     :    40000 trials in  18.86 ms (  2.12 M t/s,   408.9 cycles/t)
    demodulate_psk4     :    40000 trials in  19.18 ms (  2.08 M t/s,   415.8 cycles/t)
    demodulate_psk8     :    40000 trials in  19.46 ms (  2.06 M t/s,   421.7 cycles/t)
    demodulate_psk16    :    40000 trials in  19.77 ms (  2.02 M t/s,   428.5 cycles/t)
    demodulate_psk32    :    40000 trials in  20.02 ms (  2.00 M t/s,   433.9 cycles/t)
    demodulate_psk64    :    40000 trials in  20.29 ms (  1.97 M t/s,   439.7 cycles/t)
    demodulate_qam4     :    40000 trials in   8.17 ms (  4.90 M t/s,   177.1 cycles/t)
    demodulate_qam8     :    40000 trials in   8.49 ms (  4.71 M t/s,   184.1 cycles/t)
    demodulate_qam16    :    40000 trials in   8.66 ms (  4.62 M t/s,   187.6 cycles/t)
    demodulate_qam32    :    40000 trials in  10.29 ms (  3.89 M t/s,   223.1 cycles/t)
    demodulate_qam64    :    40000 trials in   9.24 ms (  4.33 M t/s,   200.3 cycles/t)
    demodulate_qam128   :    40000 trials in  10.12 ms (  3.95 M t/s,   219.3 cycles/t)
    demodulate_qam256   :    40000 trials in  11.20 ms (  3.57 M t/s,   242.7 cycles/t)
nco_benchmark
    nco_sincos          :    10000 trials in   4.53 ms (  2.21 M t/s,   392.7 cycles/t)
    nco_mix_up          :   160000 trials in 127.12 ms (  1.26 M t/s,   688.8 cycles/t)
    nco_mix_block_up    :   160000 trials in  87.82 ms (  1.82 M t/s,   475.9 cycles/t)
```

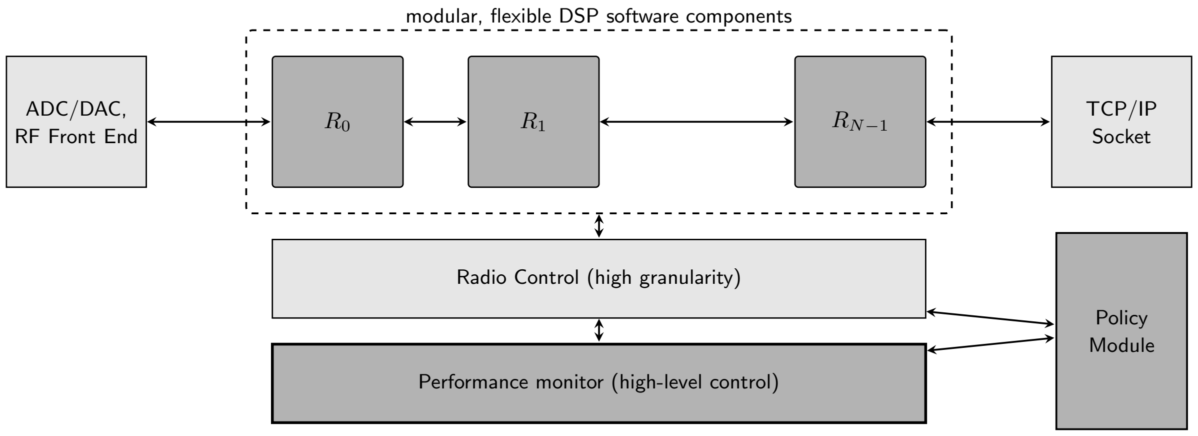

modular, flexible DSP software components
ADC/DAC, RF Front End
R_0
R_1
R_{N-1}
TCP/IP Socket
Radio Control (high granularity)
Performance monitor (high-level control)
Policy Module

Figure 5.4: Resource management framework

processor usage, POSIX kernels offer several options for monitoring of CPU and memory usage in real time and provide a programming interface through the `rusage` structure in the standard `sys/resource.h` library header. The `getrusage()` interface reports resource usage for a specific process running on the system, having the capability to estimate not only the amount of time in which the processor has spent executing instructions but other useful metrics such as memory footprint and number of page faults. This interface provides an invaluable window into the processor's core, granting the radio access to information which it can use to improve system performance.

While this interface provides the ability to monitor several resource metrics, only the computational complexity of running the algorithm is considered out of simplicity. The complexity of the algorithm is directly proportional to the processor load r_p, which is simply the relative time the processor spends executing instructions for the algorithm (T_a) divided by the total amount of time the program has spent in execution (T_e), viz

$$r_p = T_a/T_e \tag{5.3}$$

The quantities T_a and T_e are relatively easy to compute, and empirical evidence on several target platforms shows that these resource usage measurements are both accurate and consistent. Furthermore, invoking these interfaces adds minimal if not negligible overhead to the radio's performance during its operation. This implies that a performance monitor can periodically probe the state of the radio platform and continually observe its resource usage. Figure 5.4 depicts the proposed real-time resource management framework for monitoring performance of the system on a software platform. The parametrized resources $\{R_0, R_1, \ldots, R_{N-1}\}$ are adjusted by the *Radio Control* module which provides an interface for setting the radio's physical layer parameters (filter parameters, modulation scheme, transmit power, etc.) while the *Performance Monitor* evaluates the online system performance. The purpose of the *Policy Module* is to ensure that the parametrized solution set is valid for the regulated protocol.

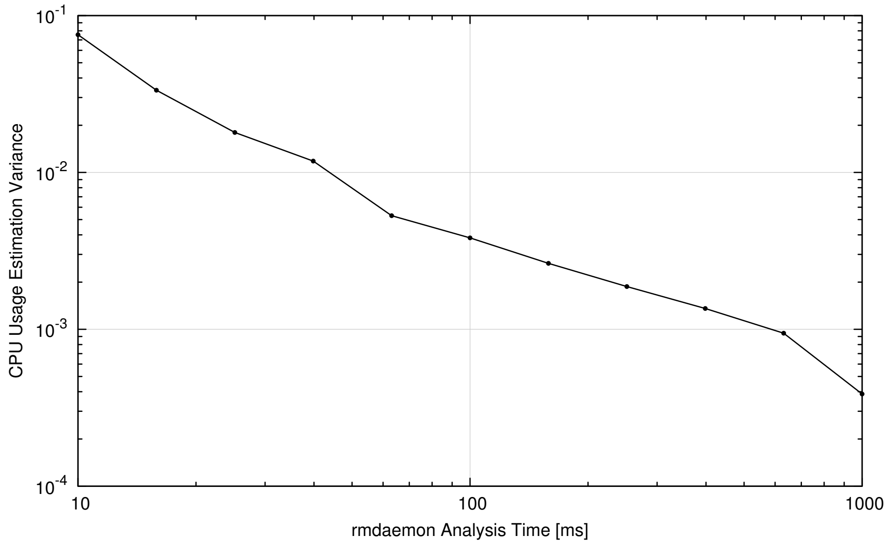


Figure 5.5: `rmdaemon` variance of CPU load estimation vs. analysis time with an average CPU load of approximately 80%. Statistics were drawn over a 180 second duration.

5.4.2 `rmdaemon` Application Program Interface

Resource monitoring is accomplished in the proceeding sections through a custom interface designed to observe processor usage. The `rmdaemon` is a background process which monitors the resource use of a particular thread and reports this information periodically back to a central controlling unit. The daemon works by periodically invoking the `getrusage()` method and comparing the difference in resource usage.

To test the daemon, a thread was initialized in which the CPU was required to perform a computational task for some amount of time, and then was allowed to sleep. By adjusting the duty cycle of computation and sleep, the program allows a fine control over the CPU's average computational load. The period of this cycle was designed to be approximately 10ms and is adjustable from nearly 0% to nearly 100% duty cycle. The daemon then monitored the CPU usage many times over the course of several minutes and reported its estimate $\hat{r}_p$ of the processor load. Figure 5.5 shows the variance of the estimate of the CPU load r_p versus analysis time T_e. Clearly increasing the analysis time reduces the variance in the estimation of the CPU load. This is due to the daemon's averaging out the processor usage over a longer amount of time. While it is desirable to increase the observation rate of the processor load to gain a more granular representation of how the CPU is being used, the error in estimation is kept manageable by increasing the observation time to a minimum of 100ms.

5.5 Experimentation: Adaptive Modulation/Error-Correction in Slowly-Varying Fading Channels

Several publications make reference to measure expended energy per bit as a metric for power efficiency is software-defined radios [15, Fig. 15], [18, Fig. 8]. Figure 5.1 demonstrates the strong relationship between processing efficiency and signal bandwidth. Although theoretical channel throughput increases with bandwidth for a given average transmit power, the efficiency eventually degrades due to an increase in processor complexity and energy consumption by the receiver. The link can therefore sacrifice throughput for bandwidth efficiency and energy consumption. This section presents the results on measuring energy efficiency on a software-defined radio platform with reconfigurable DSP components.

Modeling processing complexity for DSP algorithms and the power they consume on a variety of platforms is a difficult task. When mapping platform-independent to platform-specific models one must incorporate a host of hardware design considerations, such as floating-point capabilities, available cache size, memory constraints, instruction sets/extensions, etc. Typical communications processing blocks which consume a significant portion of processing power are filters, channelizers, FEC coding (e.g. Viterbi decoder), channel estimation (e.g. RAKE receiver), and equalization [60, Table 1]. The analysis presented in this section is focused on a specific case study of running a modulator/demodulator and forward error-correction encoder/decoder pair on a on a GPP with a PowerPC architecture. Simulations demonstrate energy-quality trade-off for error rates through adaptive modulation and forward error-correction schemes.

5.5.1 Algorithm/Complexity Relationship

Analysis begins by formulating a model to demonstrate the trade-off between bit error rate and energy consumed by the GPP for each coded bit. The average bit-error probability ($\bar{P}_b$) for an uncoded modulation scheme with a linear signal-to-noise ratio γ can be approximated as

$$\bar{P}_b \approx c_0 e^{-c_1 \gamma} + c_2 e^{-2c_1 \gamma} \tag{5.4}$$

For a coded modulation scheme, the approximation takes another form, viz.

$$\bar{P}_b \approx \frac{c_0}{1 + c_2 \gamma^{c_1}} \tag{5.5}$$

These approximations were chosen on the basis of being uniquely invertible to give a required signal-to-noise ratio γ for a target average bit error rate $\bar{P}_b$. The values for the coefficients c_0, c_1, and c_2 are given in Table 5.2 and the accuracy of the approximation is shown in Figure 5.6. Trials were conducted on 1024-bit blocks running a minimum of 1000 errors using hard-decision input to the FEC decoders. It is apparent from the BER performance of Figure 5.6 and the processing efficiency, $\mathcal{K}$, in Table 5.2 (bold column) that a strong performance trade-off exists between the two. The processing efficiency—in terms of the number of CPU clock cycles per transmitted bit—can be directly related to the energy efficiency of the radio. This preliminary work demonstrates the need for SDR platforms to incorporate resource management capabilities in order to reduce power consumption without significantly sacrificing quality of service. This is a requisite feature for portable cognitive radio platforms.

Table 5.2: Computational Complexity of Modulation/FEC Sets Running on a 867MHz PPC

Modulation	*FEC*	$\eta, [b/s/Hz]$	$\mathcal{K}, [cycles/bit]$	c_0	c_1	c_2	label
BPSK	*uncoded*	1	**86.2**	0.2013	1.11	0.13136	*a*
BPSK	$r = 1/2$, $K = 9$	0.5	**514.6**	0.456	24.799	10.667	*b*
BPSK	$r = 1/3$, $K = 9$	0.33	**726.6**	0.478	5641.4	10.6	*c*
QPSK	*uncoded*	2	**46.7**	0.193	0.5512	0.15594	*d*
QPSK	$r = 1/2$, $K = 9$	1	**435.5**	0.4773	0.0121	10.998	*e*
QPSK	$r = 1/3$, $K = 9$	0.67	**608.0**	0.4429	3.2696	10.400	*f*
16-QAM	*uncoded*	4	**69.5**	0.1358	0.1088	0.17282	*g*
16-QAM	$r = 1/2$, $K = 9$	2	**481.2**	0.491	1.09e-7	9.497	*h*
16-QAM	$r = 1/3$, $K = 9$	1.33	**676.5**	0.494	7.48e-5	9.2219	*i*
64-QAM	*uncoded*	6	**48.0**	0.084	0.025	0.2111	*j*
64-QAM	$r = 1/2$, $K = 9$	3	**438.1**	0.4973	1.4999e-10	8.2301	*k*
64-QAM	$r = 1/3$, $K = 9$	2	**611.9**	0.493	5.50e-9	9.222	*l*

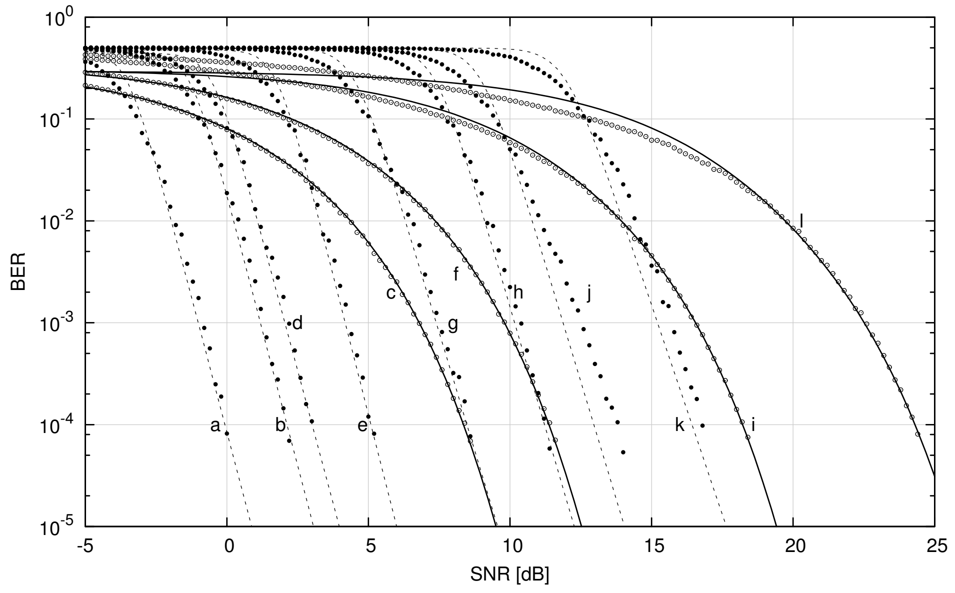


Figure 5.6: Bit error probabilities for example modulation and FEC coding scheme pairs with approximations defined by (5.4) and (5.5). Labels are given in Table 5.2.

5.5.2 Selecting appropriate modulation/coding schemes

Figure 5.7(a) demonstrates the simulated capacity of different modulation/coding scheme pairs vs. E_b/N_0 to achieve a bit error rate of $P_b = 10^{-5}$ along with the theoretical Shannon capacity bound. The pairs were chosen from all combinations of available modulation (BPSK, QPSK, $\{8, 16, 32, 64\}$-PSK, $\{16, 32, 64, 128, 256\}$-QAM, $\{8, 16, 32, 64, 128\}$-APSK) and forward error-correction schemes. The pairs are grouped into two sets: those with low-complexity FEC schemes (no coding, repeat $r1/3$, $r1/5$, Hamming(7,4) block code, and a Hamming(12,8) block code) and those with high-complexity FEC schemes (convolutional with $r1/2$ (K=7,9), and punctured rates from 2/3 through 7/8, convolutional with $r1/3$ (K=9), and convolutional with $r1/6$, (K=15)). Values nearest to the capacity bound are traditionally preferable due to their high spectral efficiency in relation to the energy required to achieve a particularly low error rate. The pairs lying along the empirical Pareto frontier (see Section 6.2.3) are listed in Table 5.3. As one can observe from the figure, certain combinations are particularly spectrally inefficient, and typically include pairs with a high modulation index and low coding rate (e.g. 64-QAM with $r\,1/3$ convolutional encoding).

Conversely, Figure 5.7(b) shows the same set of modulation and FEC scheme pairs' computational complexity plotted against E_b/N_0. Notice that while the "low complexity" pairs achieve a poor capacity for low E_B/N_0, some have a capacity comparable to the "high complexity" pairs while expending much fewer clock cycles. The pairs lying along the Pareto frontier are listed in Table 5.4. As will be demonstrated later in this section, certain pairs achieve nearly the same efficiency while requiring significantly less computational complexity.

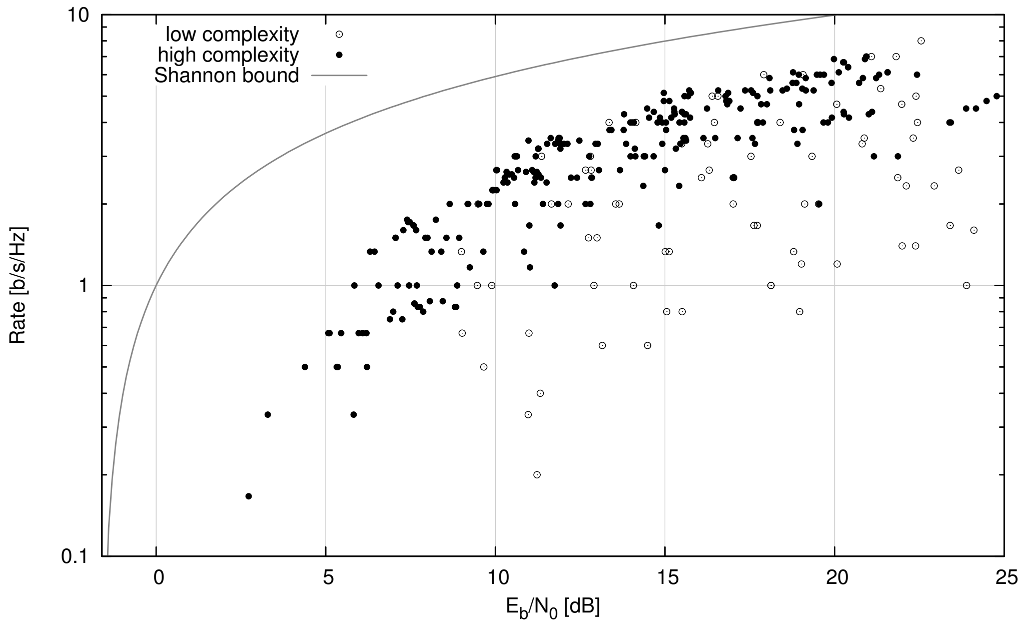
Adaptively switching modulation and coding schemes based on instantaneous received power is a technique which as been incorporated in a wide population of wireless radios; however, the impact these switching mechanisms have on baseband processors implemented on reconfigurable platforms has garnered little investigation. This section takes a renewed look at adaptive switching modulation/coding schemes not simply for the sake of improving spectral efficiency but how appropriately choosing a set of schemes can significantly reduce the complexity of the receiver without unnecessarily sacrificing capacity [38].

5.5.3 System Model

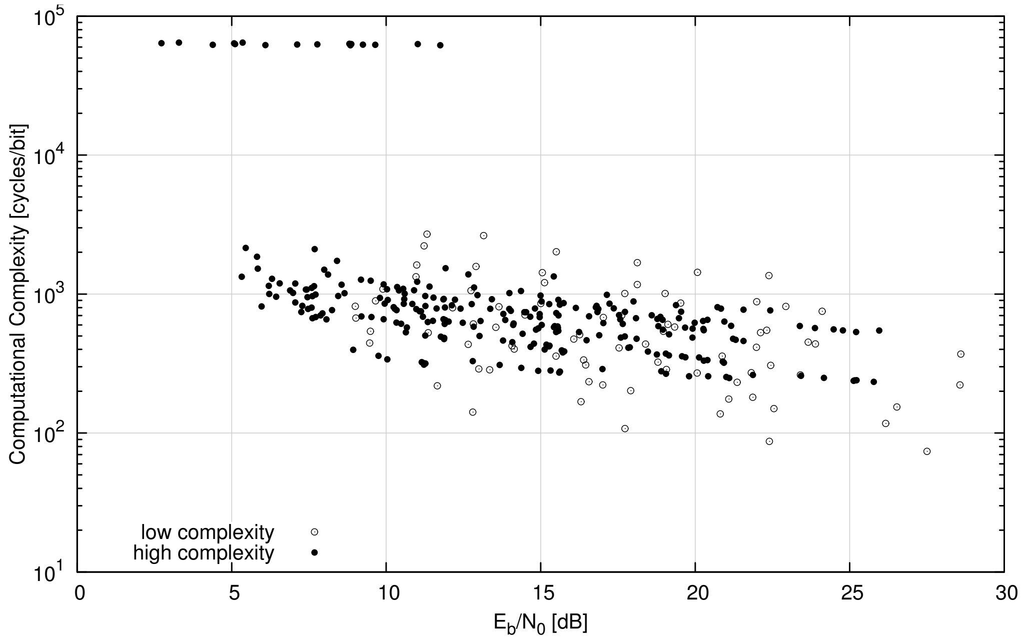
Given a fading distribution $f_\gamma(\gamma; \Omega)$ for instantaneous received signal power γ and average power Ω, an upper bound on the average bit error rate for a modulation/coding scheme pair can be computed by averaging its instantaneous BER performance over the distribution, viz. [91]

$$\bar{P}_b^{m,f}(\Omega) = \int_0^\infty P_b^{m,f}(\gamma) f_\gamma(\gamma; \Omega) d\gamma \tag{5.6}$$

For slowly-varying channels, the received signal power is assumed to be flat over the duration of a data packet, and thus the switching procedure chooses the "optimal" modulation/coding scheme pair in the set which matches the estimated target error probability to the received signal power. The appropriate pairs for subsequent packets are chosen on the basis of signal strength alone. The average BER of an adaptive switching scheme is therefore computed by summing the marginal



(a) Capacity, optimum values listed in Table 5.3



(b) Complexity, optimum values listed in Table 5.4

Figure 5.7: Simulated capacity/complexity vs. E_b/N_0 for $P_b = 10^{-5}$ with each point representing a specific modulation/coding scheme pair.

Table 5.3: Spectrally Efficient Modulation/FEC Sets

Modulation	*FEC*	$\eta, [b/s/Hz]$	$\mathcal{K}, [cycle/bit]$	E_b/N_0
BPSK	conv. $r = 1/6$, $K = 15$	0.167	63954.0	2.73
QPSK	conv. $r = 1/6$, $K = 15$	0.333	64529.0	3.29
8-PSK	conv. $r = 1/6$, $K = 15$	0.500	62136.0	4.39
16-APSK	conv. $r = 1/6$, $K = 15$	0.667	63704.0	5.08
QPSK	conv. $r = 1/2$, $K = 9$	1.000	1524.2	5.85
QPSK	conv. $r = 2/3$, $K = 9$	1.333	1289.2	6.31
QPSK	conv. $r = 4/5$, $K = 7$	1.600	822.5	7.29
QPSK	conv. $r = 7/8$, $K = 9$	1.750	1079.5	7.40
16-QAM	conv. $r = 1/2$, $K = 9$	2.000	1014.2	8.65
8-APSK	conv. $r = 3/4$, $K = 9$	2.250	1173.6	9.92
16-APSK	conv. $r = 2/3$, $K = 9$	2.667	1083.0	10.03
16-QAM	conv. $r = 3/4$, $K = 9$	3.000	851.4	10.57
16-QAM	conv. $r = 6/7$, $K = 9$	3.429	779.8	10.98
16-QAM	conv. $r = 7/8$, $K = 9$	3.500	788.1	11.63
16-QAM	uncoded	4.000	285.1	13.35
32-QAM	conv. $r = 6/7$, $K = 9$	4.286	720.3	13.79
64-QAM	conv. $r = 3/4$, $K = 9$	4.500	740.3	14.47
64-QAM	conv. $r = 4/5$, $K = 9$	4.800	717.5	14.97
64-QAM	conv. $r = 6/7$, $K = 9$	5.143	682.6	14.97
64-QAM	conv. $r = 7/8$, $K = 7$	5.250	379.9	15.71
64-QAM	uncoded	6.000	201.8	17.91
128-QAM	conv. $r = 7/8$, $K = 9$	6.125	663.2	18.77
256-QAM	conv. $r = 6/7$, $K = 9$	6.857	622.3	19.98
256-QAM	conv. $r = 7/8$, $K = 7$	7.000	320.9	20.93
256-QAM	uncoded	8.000	150.1	22.55

Table 5.4: Computationally Efficient Modulation/FEC Sets

Modulation	*FEC*	$\eta, [b/s/Hz]$	$\mathcal{K}, [cycle/bit]$	E_b/N_0
BPSK	conv. $r = 1/6$, $K = 15$	0.167	63954.0	2.73
8-PSK	conv. $r = 1/6$, $K = 15$	0.500	62136.0	4.39
BPSK	conv. $r = 1/2$, $K = 9$	0.500	1332.5	5.33
BPSK	conv. $r = 2/3$, $K = 7$	0.667	814.2	5.97
BPSK	conv. $r = 3/4$, $K = 7$	0.750	743.4	7.25
BPSK	conv. $r = 6/7$, $K = 7$	0.857	670.0	7.61
BPSK	conv. $r = 7/8$, $K = 7$	0.875	657.0	8.07
8-PSK	conv. $r = 1/2$, $K = 7$	1.500	396.9	8.93
8-PSK	conv. $r = 2/3$, $K = 7$	2.000	359.8	9.75
8-PSK	conv. $r = 3/4$, $K = 7$	2.250	339.6	10.03
8-PSK	conv. $r = 4/5$, $K = 7$	2.400	324.0	11.15
8-PSK	conv. $r = 5/6$, $K = 7$	2.500	320.9	11.18
8-PSK	conv. $r = 7/8$, $K = 7$	2.625	310.8	11.22
8-PSK	Hamming $(12, 8)$	2.000	218.7	11.65
8-PSK	uncoded	3.000	141.3	12.80
16-PSK	uncoded	4.000	107.7	17.73
32-PSK	uncoded	5.000	87.2	22.40
64-PSK	uncoded	6.000	73.8	27.50

BER over the partitioned distribution:

$$\bar{P}_b(\Omega) = \frac{\sum\limits_{i=0}^{N-1} k_i r_i \int_{\gamma_i}^{\gamma_{i+1}} P_i(\gamma) f_\gamma(\gamma, \Omega) d\gamma}{\sum\limits_{i=0}^{N-1} k_i r_i \int_{\gamma_i}^{\gamma_{i+1}} f_\gamma(\gamma, \Omega) d\gamma} \tag{5.7}$$

where Ω is the average signal power, $k_i = \log_2(M_i)$ bits per symbol for the modulation type, r_i coding rate, γ_i is the threshold SNR, and $P_i(\gamma)$ is the BER distribution for the i^{th} of N modulation/coding pairs under an AWGN channel. Note that the expression for average BER is now no longer a function of the instantaneous received signal power γ but the average received signal power Ω. Note also that the denominator in (5.7) is the average spectral efficiency, $\bar{\eta}(\Omega)$, viz

$$\bar{\eta}(\Omega) = \sum_{i=0}^{N-1} k_i r_i \int_{\gamma_i}^{\gamma_{i+1}} f_\gamma(\gamma, \Omega) d\gamma \tag{5.8}$$

In a similar fashion the average receiver complexity $\bar{\mathcal{K}}$ can be defined as

$$\bar{\mathcal{K}}(\Omega) = \sum_{i=0}^{N-1} \mathcal{K}_i \int_{\gamma_i}^{\gamma_{i+1}} f_\gamma(\gamma, \Omega) d\gamma \tag{5.9}$$

where $\mathcal{K}_i$ is the computational complexity of the modulation/coding pair measured in terms of CPU clock cycles per uncoded bit.

5.5.4 Results

In order to gain insight into the tradeoffs between computational complexity and link quality this analysis adopts the Rayleigh fading model, a typically pessimistic assumption about mobile channel conditions. The Rayleigh fading model for instantaneous received signal power γ is

$$f_\gamma(\gamma; \Omega) = \frac{1}{\Omega} e^{-\gamma/\Omega} \tag{5.10}$$

where Ω is the average power.[1] Error rates were generated by encoding data packets using a discrete set of modulation/coding scheme pairs, pushing the resulting symbols through an AWGN channel, and measuring the resulting bit errors. For each data point, a minimum of 10,000 trials were performed ensuring at least 2000 observed errors. A block interleaver was inserted after the encoder to help randomize bit errors within the data packet to aid the decoder. Computational complexity measurements were conducted by running 100,000 iterations of each demodulator and FEC decoder on an embedded 867 MHz PowerPC processor with AltiVec SIMD extensions [56].

Two sets of adaptive modulation/coding switching schemes were compared; the first set contains typical half-rate convolutional codes with constraint lengths $K = 7$, $K = 9$, and punctured codes

[1] It is important to note that while fading models typically are expressed in terms of signal amplitude, (5.10) denotes fading in terms of signal power. This is a trivial variable transformation necessary for analysis.

Table 5.5: Available modulation/FEC scheme pairs, broken into 2 sets (experiment 5.5)

Modulation scheme	FEC scheme	$\eta, [b/s/Hz]$	$\mathcal{K}_i, [cycles/bit]$	γ_i [dB]
		$\mathcal{S}_\eta$, spectrum efficiency-driven set		
QPSK	conv. r=1/3, K=9	0.67	612.5	6.45
QPSK	conv. r=1/2, K=9	1.00	492.9	7.84
QPSK	conv. r=2/3, K=9	1.33	521.2	9.57
QPSK	conv. r=3/4, K=9	1.50	511.5	10.34
16-QAM	conv. r=1/2, K=9	2.00	519.1	13.52
16-QAM	conv. r=2/3, K=9	2.67	540.8	15.32
16-QAM	conv. r=3/4, K=7	3.00	211.7	16.61
16-QAM	conv. r=4/5, K=9	3.20	526.0	17.02
16-QAM	conv. r=7/8, K=9	3.50	501.3	18.06
64-QAM	conv. r=2/3, K=9	4.00	520.7	20.39
64-QAM	conv. r=3/4, K=9	4.50	511.0	21.48
64-QAM	conv. r=4/5, K=9	4.80	509.2	22.23
64-QAM	conv. r=7/8, K=9	5.25	485.9	23.40
64-QAM	uncoded	6.00	33.8	24.28
		$\mathcal{S}_\mathcal{K}$, computational efficiency-driven set		
BPSK	Hamming (7,4)	0.57	104.8	8.92
BPSK	uncoded	1.00	56.2	10.07
QPSK	Hamming (7,4)	1.14	66.3	10.49
QPSK	uncoded	2.00	34.2	12.77
8-PSK	uncoded	3.00	141.4	17.28
16-QAM	uncoded	4.00	47.3	18.90
64-QAM	uncoded	6.00	33.8	24.28

with rates from ranging from 2/3 to 7/8; the second contains either no error-correction (uncoded) or only a computationally efficient Hamming$(7,4)$ block code. These pairs are presented in Table 5.5 along with the average computational complexity to receive each coded bit, and the required instantaneous SNR to achieve a BER of 10^{-3} in an AWGN channel.

Given the simplicity of the distribution of signal power for the Rayleigh fading model, the expressions for average spectral efficiency in (5.8) and average computational complexity (5.9) can be rewritten in closed form respectively as

$$\bar{\eta}(\Omega) = \sum_{i=0}^{N-1} k_i r_i \left(e^{-\gamma_i/\Omega} - e^{-\gamma_{i+1}/\Omega} \right) \tag{5.11}$$

$$\bar{\mathcal{K}}(\Omega) = \sum_{i=0}^{N-1} \mathcal{K}_i \left(e^{-\gamma_i/\Omega} - e^{-\gamma_{i+1}/\Omega} \right) \tag{5.12}$$

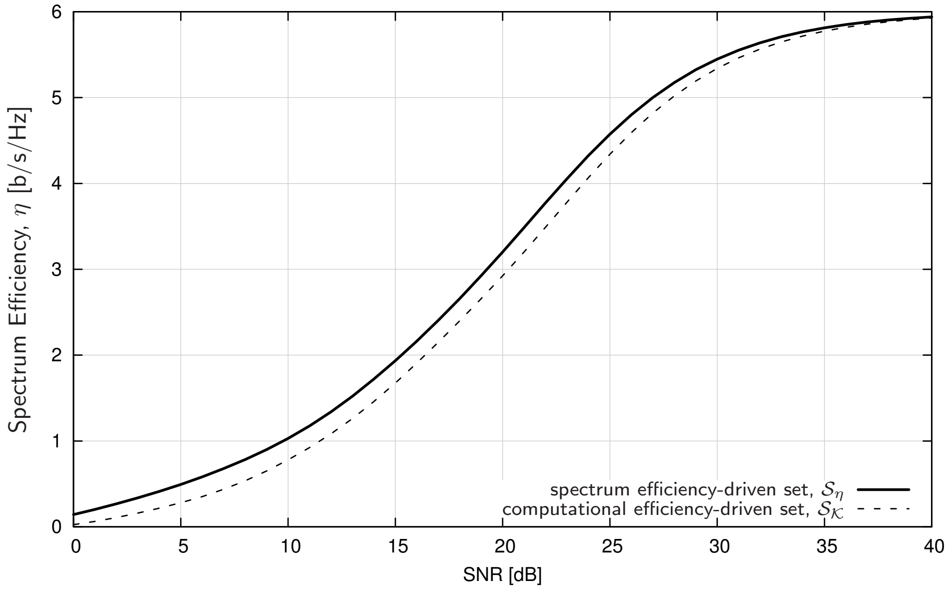
The values for average error probability given by (5.7), were computed through numerical integration. The switching levels for adaptation are given in Table 5.5.

The marginal BER in (5.7) was computed through numerical integration of the instantaneous BER curves, evaluated through simulation. Finally, a fair assessment of throughput was established by multiplying the average spectral efficiency by the probability of receiving a packet correctly, measured empirically through simulation. While the average BER was targeted to be 10^{-3}, this value cannot be achieved for small values of Ω unless stronger forward error-correction codes are used.

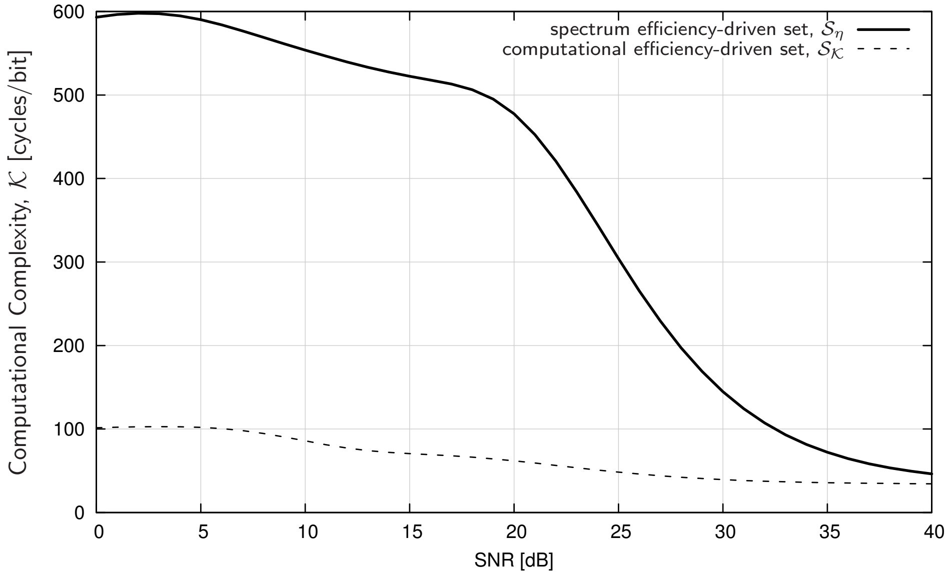
Figure 5.8 depicts the results of the simulations. While it is hardly surprising that the stronger convolutional codes provide a higher capacity than the uncoded/weakly-coded set, the actual spectral efficiency does not suffer a significant hit. The strongest discrepancy of $\bar{\eta}$ between the two sets exists at $\Omega \approx 23$ dB where the difference in spectral efficiency is only about 0.25 b/s/Hz (a 7% degradation) as seen in Figure 5.8(a). In contrast, Figure 5.8(b) demonstrates that the difference in computational complexity between the two sets is nearly a factor 6 for low SNR in favor of the weaker codes. For both sets, the average complexity decreases with Ω. Eventually the complexity for the spectrum efficiency-driven set declines steeply (around 20 dB) as the probability of running the computationally efficient uncoded 64-QAM dominates. Analysis could easily be run for any number of modulation/coding combinations, with appropriate sets sought to maximize spectral efficiency, minimize complexity, or some middle-ground combination of the two; the particular sets were chosen specifically to accentuate this trade-off.

5.6 Experimentation: Adaptive Modulation/Error Correction Over the Air

The previous section demonstrated the strong relationship between channel conditions and computational complexity required by the receiver, specifically through the adaptation of forward error-correction (FEC) coding in conjunction with the scheme used to modulate the underlying data. While the tradeoffs between received signal strength (relative to the noise level) to error rate performance is well known within the wireless engineering community, its impact on computational resources is not. This is in part due to the complexities of modeling complex algorithms on various hardware devices and is significantly dependent upon the algorithm, the physical characteristics of the platform, and the implementation. Professional signal processing engineers use sophisticated benchmarking and profiling tools to eliminate any and every unnecessary instruction in the algorithm to ensure its performance is streamlined and as near to optimum as possible. While a useful design approach, it is impractical to apply it to the modern model-based software design in which there exists a strict disconnect between the platform-independent and platform-specific models. This section demonstrates the use of online processor complexity monitoring specifically to improve the computational efficiency of the receiver in a dynamic wireless environment.



(a) Average spectral efficiency, $\bar{\eta}(\Omega)$



(b) Average computational complexity, $\bar{\mathcal{K}}(\Omega)$

Figure 5.8: Performance of adaptive switched modulation/coding scheme sets in a slowly-fading Rayleigh channel.

5.6.1 System model

As demonstrated in the previous section through extensive base-band simulation, receivers need to work harder to recover signals which are weaker, noisier, and in general more corrupted. Traditional radio designs implement standards in hardware; the physical layer is designed to operate under the predicted worst-case scenarios and has little flexibility for on-the-fly modification to adapt to changing wireless conditions aside from switching between modulation and forward error-correction coding schemes. While these adaptations play a significant role in link throughput, data reliability, and energy consumption, other common physical-layer parameters such as filter length, number of equalizer taps, rake fingers, and image-rejection filter length are often left untouched by the receiver and viewed as pre-specified. Neglecting these dimensions of communications signal processing during the design process results in an acute sub-optimal solution. Just as any embedded software development must consider resource consumption on its host platform, so must SDR in this regard.

Understanding the dynamics behind algorithms and the resources they consume is necessary in gaining back what SDR sacrifices for flexibility. This chapter proposed the use of computational channel efficiency $\mathcal{K}$ as a performance metric for comparing receiver designs. Computational channel efficiency is simply the number of properly synchronized and decoded bits of information conveyed over a communications link per clock cycle on the host platform. It is therefore necessary to monitor waveform complexity at the receiver in order to gain insight as to how good the receiver structure performs. The processor's clock cycles are a restricted commodity and limited resource for facilitating communications. The complexity $\mathcal{K}$ has units clock cycles per bit and can be measured by several methods. For most platforms, processor usage can be easily measured on certain running processes as a strict percentage of its maximum load. As such, a processor with a master clock frequency of F_p cycles/second operating at r_p percent of its maximum while running a receiver capable of decoding η bits per second per Hz has a processing efficiency of

$$\mathcal{K} = F_p r_p / \eta \tag{5.13}$$

While the quantity in (5.13) gives an indication of peak computational efficiency, it does not thoroughly describe the actual throughput of the link given its channel quality. As demonstrated by the results in the previous section, channel quality has significant implications on the workload required by the receiver. Spectral efficiency, in this case, is a measure of properly received data in the link, compensating for erroneous packets. Normalized spectral efficiency is computed as

$$\zeta(\gamma) = \eta\big(1 - P_p(\gamma)\big) \tag{5.14}$$

where again η is the theoretical spectral efficiency of the transmission scheme (e.g. modulation/-coding scheme pair with framing overhead) and P_p is the packet error rate at the receiver under an instantaneous signal-to-noise ratio (SNR), γ. Both η and P_p are strongly affected by the choice of radio transmission scheme, as is the resulting receiver complexity. This normalized quantity describes the actual efficiency of data being conveyed over the wireless channel. The quantity in (5.14) is theoretically bounded by γ and is well known in information theory. Typically channel capacity is referenced not in terms of the ratio of signal energy to noise (E_s/N_0), but of bit energy to noise (E_b/N_0). The use of E_b/N_0 is viewed as a fair comparison of different transmission schemes as it incorporates the rate of the transmission into its analysis of signal strength. However the analysis in this section uses E_s/N_0 because the receiver's signal-to-noise ratio incorporates its ability to synchronize the entire physical-layer frame, including detection, acquisition, and tracking. The ratio E_b/N_0 is really only useful once the receiver has synchronized, ignoring a very important and practical component of wireless communication.

For a fair comparison of transmission schemes, the complexity of the receiver must be normalized to the actual throughput. Incorporating (5.13) and (5.14) gives

$$\dot{\mathcal{K}}(\gamma) = \mathcal{K}/P_p(\gamma) \tag{5.15}$$

which can be understood as the average number of clock cycles to decode a single bit of data through a communications link, while accounting for erroneous data.

5.6.2 Scenario

Consider a scenario in which two software radios need to establish a communications link in a wireless network. These nodes need to negotiate a protocol which achieves the highest data rate possible without exceeding the capabilities of either node. For most protocols, the burden of computation is much higher at the receiving end of the link and is therefore the limiting factor. Furthermore heterogeneous networks are likely to contain nodes of vastly different capabilities; some with higher computational bandwidth than others. Should they just select one of a pre-determined set of protocols, it is likely that if their capabilities overlap their solution will be sub-optimal and the resulting performance poor. The previous section demonstrated that simply choosing an appropriate set of modulation and FEC coding scheme pairs can result in a significant performance improvement. Here, these results are validated through over-the-air data transfer in a packet radio network. While a number of other radio parameters significantly affects the computational efficiency of the receiver, the analysis is limited to simply modulation and coding schemes as to be consistent with the results in Section 5.5.

5.6.3 Framing structure

In order to validate the results in Section 5.6, a simple physical-layer wireless protocol was developed in which narrowband packets were transmitted between two software-defined radio platforms. The structure for framing the packets was designed to be as versatile as possible, allowing for fast acquisition, reliable detection, variable payload lengths, coherent demodulation of any number of modulation schemes, and decoding of layered, interleaved FEC schemes. The framing structure consists of six basic components, as depicted by Figure 3.15. A fully-reconfigurable software transceiver was implemented in the C programming language capable of synchronizing to packets transmitted over a wireless channel. The receiver consists of the following modules: automatic gain control with squelch to disable unnecessary processing when the signal strength is low, received signal strength and signal-to-noise radio estimation, numerically-controlled oscillator and second-order PLL for carrier synchronization, multi-rate decimator for symbol timing recovery [39], frame lock detection, variable-level demodulation, block de-interleaver, forward error-correction decoder, and cyclic redundancy check validator. The details of the framing structure and synchronizer can be found in Section 3.5.

5.6.4 Experimental Setup

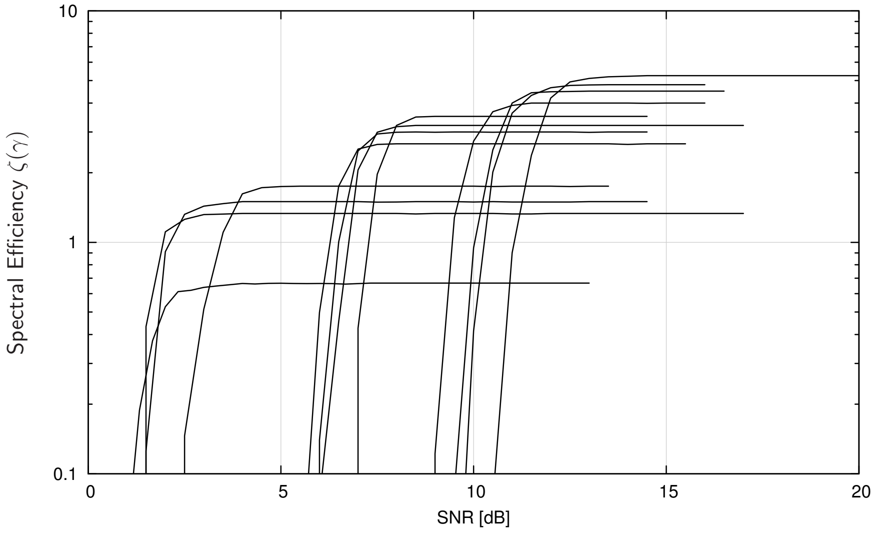
The goal of the experiment was to achieve a maximum spectral efficiency (ζ) given an instantaneous signal-to-noise ratio (γ) by switching between available modulation and forward error-correction schemes from within a set. As with the previous experiment, two different sets were chosen: one on the basis of maximizing spectral efficiency, and the other on the basis of minimizing computation complexity. The first set, $\mathcal{S}_\eta$ (designed for spectrum efficiency), contains typical $r = 1/2$ and $r = 1/3$ convolutional codes with a constraint length $K = 9$ and punctured at rates ranging from 2/3 to 7/8. The second set, $\mathcal{S}_\mathcal{K}$ (designed for computation efficiency), uses either no error-correction (uncoded) or only a computationally efficient Hamming$(12, 8)$ 2/3-rate block code. The two sets are depicted in Table 5.6 along with their average computational loads measured in the number of processor clock cycles (in thousands) required to decode a single bit of data through the entire receiver chain. These complexity measurements were conducted by invoking `getrusage()` as described in Section 5.3 using (5.13) and (5.3). Notice that their spectral efficiencies have nearly the same range (from 0.6667 b/s/Hz at a minimum to about 5 b/s/Hz at a maximum), yet have starkly different receiver complexities.

Frames were transmitted over a wireless link in a laboratory environment with varying transmit powers such that the receiver's signal-to-noise ratio spanned approximately -2 dB to 23 dB. The transmitter and receivers ran on independent Intel Pentium processors, each connected to a USRP [21] and operating at a symbol rate of W=225 kHz with a payload of 1024 bits (128 bytes). The receiver made an estimate $\hat{\gamma}$ of each frame's signal-to-noise ratio using a combination of the received signal strength relative to its pre-characterized noise level and the variance of its demodulator error vector magnitude. Statistics were gathered for the data throughput and computational complexity under each modulation and coding scheme pair from each of the two sets depicted in Table 5.6. Computational complexity analysis was conducted by way of the methods described in Section 5.3. It is important to recognize that the clock cycle measurements in Table 5.6 incorporate the entire receiver including filtering, gain control, frame detection, carrier frequency/phase recovery, symbol timing synchronization, decimation, demodulation, and forward error-correction decoding, while the results in the first experiment only incorporated the demodulator and FEC decoder.

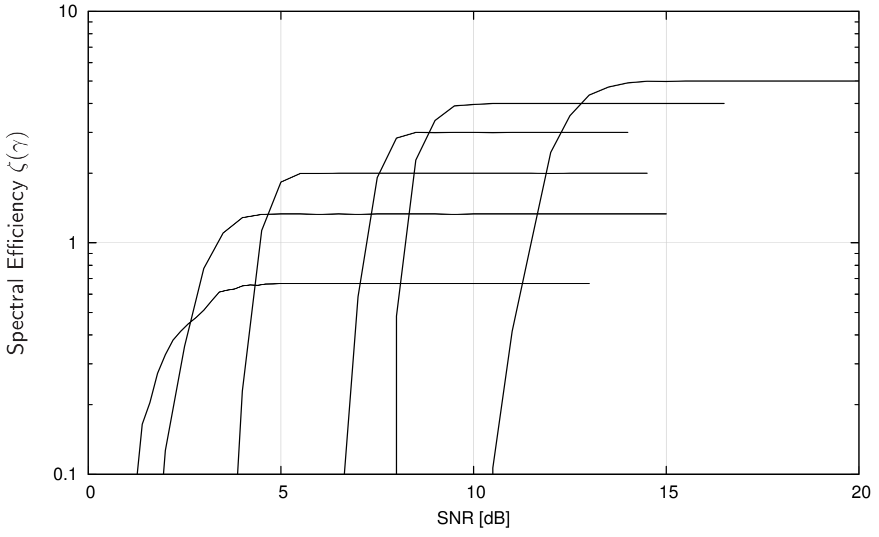
5.6.5 Results

Figure 5.9 depicts the measured spectral efficiencies for the two modulation/coding scheme sets measured over the air in a laboratory environment, computed using (5.14). As the received signal level degrades, the packet error rate approaches unity, and therefore the throughput approaches zero. As expected, this transition occurs sharply with γ such that an SNR threshold is required to achieve any suitable throughput. This threshold varies for each modulation/coding scheme pair, as does the actual peak throughput.

In an adaptively switching protocol, the radios choose the pair with the highest spectral efficiency for a given γ in order to maximize the total throughput for the link. To compare, the peak spectral efficiency for each set—the value riding along the crest of Figures 5.9(a) and 5.9(b)—is plotted in Figure 5.10(a). This shows the maximum spectral efficiency possible for the system given a specific channel condition. Finally, Figure 5.10(b) shows the normalized computational complexity $\dot{K}$ of

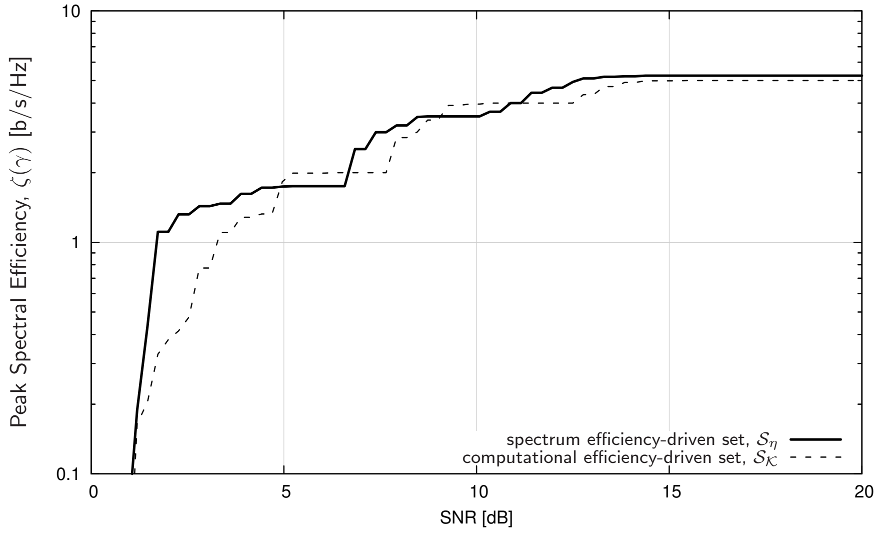


(a) $\mathcal{S}_\eta$, spectrum efficiency-driven set

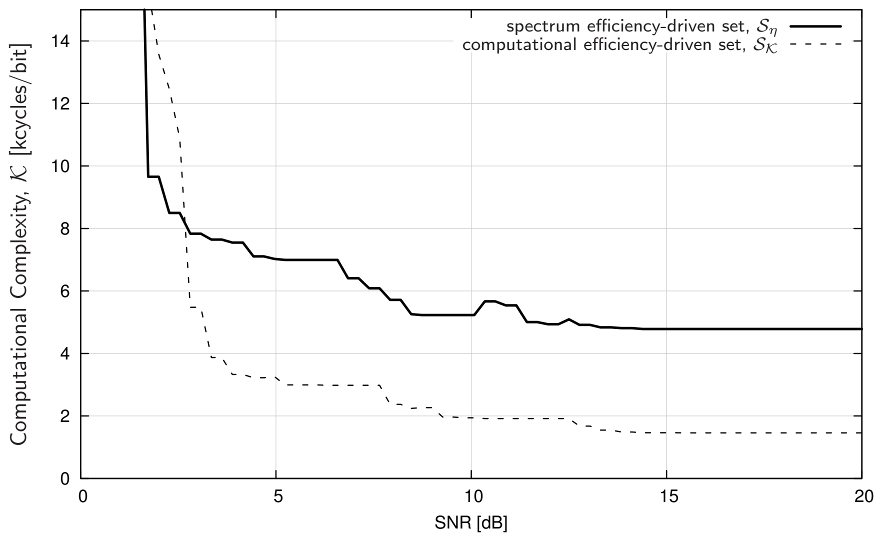


(b) $\mathcal{S}_\mathcal{K}$, computational efficiency-driven set

Figure 5.9: Performance of adaptive switched modulation/coding scheme sets' average spectral efficiencies $\zeta(\gamma)$ measured in an over-the-air in a laboratory test environment.



(a) Peak spectral efficiency $\zeta(\gamma)$



(b) Average computational complexity, $\dot{\mathcal{K}}$

Figure 5.10: Performance of adaptive switched modulation/coding scheme sets over the air in a laboratory wireless environment.

Table 5.6: Available modulation/FEC scheme pairs, broken into 2 sets, $\mathcal{S}_\eta$ and $\mathcal{S}_\mathcal{K}$ (over-the-air experiment in Section 5.5)

Modulation scheme	FEC scheme	$\eta, [b/s/Hz]$	$\mathcal{K}_i, [kcycles/bit]$
	$\mathcal{S}_\eta$, spectrum efficiency-driven set		
QPSK	conv. $r = 1/3$, $K = 9$	0.667	12.512
QPSK	conv. $r = 2/3$, $K = 9$	1.333	8.135
QPSK	conv. $r = 3/4$, $K = 9$	1.500	7.568
QPSK	conv. $r = 7/8$, $K = 9$	1.750	7.064
16-QAM	conv. $r = 2/3$, $K = 9$	2.667	6.138
16-QAM	conv. $r = 3/4$, $K = 9$	3.000	5.733
16-QAM	conv. $r = 4/5$, $K = 9$	3.200	5.753
16-QAM	conv. $r = 7/8$, $K = 9$	3.500	5.281
64-QAM	conv. $r = 2/3$, $K = 9$	4.000	5.254
64-QAM	conv. $r = 3/4$, $K = 9$	4.500	4.975
64-QAM	conv. $r = 4/5$, $K = 9$	4.800	4.840
64-QAM	conv. $r = 7/8$, $K = 9$	5.250	4.831
	$\mathcal{S}_\mathcal{K}$, computational efficiency-driven set		
BPSK	Hamming $(12, 8)$	0.667	7.874
QPSK	Hamming $(12, 8)$	1.333	3.239
QPSK	uncoded	2.000	3.013
8-PSK	uncoded	3.000	2.264
16-QAM	uncoded	4.000	1.938
32-QAM	uncoded	5.000	1.474

the two sets, as computed by (5.15).

As expected, $\mathcal{S}_\eta$ has typically a higher spectrum efficiency than $\mathcal{S}_\mathcal{K}$, but for certain values of γ it is actually slightly lower. This is demonstrated in Figure 5.10(a) near $\gamma = 6$dB and again near $\gamma = 10$dB where the computational efficiency-driven set exhibits a slightly higher spectral efficiency than that of $\mathcal{S}_\eta$. This discrepancy can be attributed to the synchronizer's inability to lock onto signals with moderate carrier phase and frequency offsets as well as timing phase offsets. While simulation suggests that the packet error rate should be lower for certain modulation schemes, the receiver is unable to extract an adequate channel phase estimate due to noise and therefore cannot establish a carrier lock. This results in a constellation with a gross carrier phase offset and thus a higher bit error-rate than theory under ideal carrier recovery assumptions. In a practical sense, the limiting factor of the physical layer is the distance between signal points as they are responsible for proper demodulation and proper carrier recovery. As a result, the errors introduced by increasing the modulation scheme cannot easily be recovered by using forward error correction.

Clearly the introduction of convolutional decoding increases the required number of computations at the receiver. This is a well-known phenomenon and makes a significant impact on the receiver's computational complexity. Not only does using a stronger forward error-correction scheme typically increase computational requirements of the receiver by running the algorithm alone, but decreasing the rate of the code also increases the length of the frame. This, in turn, requires the receiver to run more filtering and gain operations which directly impact its computational load. The processing complexity of $\mathcal{S}_\eta$ is roughly three times higher than that of $\mathcal{S}_\mathcal{K}$ at the highest signal-to-noise ratio; this suggests that the receiver could actually increase its spectrum bandwidth running at several times its nominal speed, increasing its data rate. Alternatively, the same receiver could be deployed on a more primitive piece of hardware, allowing for nearly the same throughput to be achieved on nodes without higher capabilities. In either case, choosing appropriate schemes implies a more efficient receiver.

For very low signal-to-noise ratios (less than 3dB) the spectrum efficiency-driven set actually outperforms the computation efficiency-driven set; this is because the receiver's throughput is so low that the denominator in (5.15) approaches zero causing $\dot{\mathcal{K}}$ to explode. However, this is only an issue for low values of γ in which both sets suffer.

5.7 Conclusions

This chapter has analyzed and demonstrated the advantages of monitoring resource consumption of software-defined radio platforms, and the results are promising; adapting link-level parameters to match channel conditions suggest not only performance improvements through spectral efficiency but computational complexity (and therefore power efficiency) as well. Simulated results have been confirmed with an over-the-air implementation of an adaptive radio capable of switching modulation and coding schemes, saving the host receiver valuable computational bandwidth while sacrificing minimal spectral efficiency. These results suggest an even greater increase in performance is possible by extending the analysis to encompass the full receiver architecture and allowing the radio to adjust nearly all aspects of its receiver. The next chapter extends these results by incorporating the physical layer synchronizers of the preceding chapters with the proposed resource monitoring framework into an adaptive packet radio system controlled by an intelligent engine.

Chapter 6

Cognitive Engine Design

While the modern concept of cognitive radio promises specific enhancements such as spectrum compactness, efficiency, and higher throughput, the sole purpose in introducing artificial intelligence and cognition into radio design [46] is simply to facilitate communication. The motivation in this Chapter for employing cognition in radios and wireless networks resides in the fact that observations about the surroundings are incomplete and mathematical models used to describe the environment are imprecise. As such, it is necessary for the radio to learn specific characteristics about its environment in order to provide a sufficient quality of service to its users. This is particularly critical for dynamic spectrum environments which are not governed by any particular protocol, and as a result link quality is highly dependent upon spectrum availability and interference levels.

This chapter presents a generalized informal definition of cognitive radio and discusses optimization methodology for its autonomous operation. A generalized theory for describing the relationship between radio parameters, goals, and environmental observables is provided. A novel utility function is introduced which provides metric abstraction and allows for generalized used in cognitive engines. Furthermore, a generic case database-enabled cognitive engine is proposed which uses specific knowledge and experience rather than general knowledge from within a problem domain. The engine is used to control the parameters of a wireless communications link between a pair of radios operating in a laboratory environment, and its performance is validated by adapting its transmission scheme to varying channel conditions while reducing the computational burden at the receiver.

6.1 Taxonomy of Radio Cognition

With the onset of cognitive radio technologies comes a definition for each implementation. Predominantly considered to be an extension of software-defined radio, cognitive radios have been proposed to enable such technologies as dynamic spectrum access for efficient whitespace re-use. While DSA is a useful application for CR, it is by no means the only one. This section gives a broad definition for cognitive radio and breaks down its objectives into several levels within the domain of machine learning. The intent here is to abstract from any particular application, and

Table 6.1: Taxonomy of the Cognitive Domain

Category	*Description*	*Radio Domain Examples*
Knowledge	the ability to recall data and information	look-up table; memory; case database
Comprehension	understand meaning, symbolic representation	pre-programmed rules
Application	apply knowledge to a new situation	statistical inference of data to draw trends
Analysis	separation of concepts into its component parts to understand organizational structure	determine reason for low performance by observing individual processing blocks; "recognize logical fallacies in reasoning"
Synthesis	building structure or pattern from diverse elements	on-the-fly reconfigurability; creating new policy
Evaluation	make judgments about the value of ideas or materials	select the most effective solution by understanding its performance characteristics

provide a unified definition for comprehension.

6.1.1 Bloom's Taxonomy of the Cognitive Domain

The *Taxonomy of Educational Objectives* was developed by Benjamin Bloom in the 1950s as a set of learning objectives for educators [10]. Bloom's Taxonomy is a hierarchical classification with three domains: *affective*, *psychomotor*, and *cognitive*. Most references to Bloom's Taxonomy refer to the cognitive domain, which is broken up into six categories: *knowledge*, *comprehension*, *application*, *analysis*, *synthesis*, and *evaluation*, each being a requisite for all higher categories. While classically used in educational psychology, Bloom's cognitive domain relates very strongly to other types of learning; in this case radio intelligence. Table 6.1 dissects each category and gives a brief example of its application in the radio domain.

6.1.2 Cognitive Engine

The radio itself is supervised by a controlling entity called the cognitive engine. The goal of a cognitive engine may simply be stated as *optimally manage the radio's finite resources within a dynamic environment*; however the details of this action are specific to its application. Resources can take many forms, both the tangible and intangible, and typically include energy, power, memory, computational bandwidth, and shared electromagnetic spectrum. Despite the flexibility in software-defined radio platforms, the engine must still operate within a strictly confined set of policies. Depending upon the scenario its operator may grant leniency to the engine, allowing it to create its own protocol with relaxed constraints. As a result, cognitive radios have been shown to provide significant resource savings through waveform adaption [57].

6.1.3 Informal Definition

A number of definitions exist for cognitive radio which attempt to tie its intended application formally into its description; the most common application being dynamic spectrum access. Binding a potential application to its definition is short-sighted as its potential applications may change with time. Cognitive radio is further defined by some to be built on a software-defined platform; while SDR is a likely candidate for cognitive radio whose operation likely requires waveform flexibility, it is not a necessity.

Within the context of the cognitive domain, a cognitive radio must be driven by a set of goals and must be capable of understanding and learning how its actions impact these goals. This might involve recalling and correlating past actions, scenarios, and performance, and requires awareness of its environments and its own capabilities. At the lowest level, the radio must be capable of the statistical inference of data to draw trends on performance bounds, and at the very least perform simple adaptations using a set of pre-programmed rules to improve its performance. At a higher level it can make judgments about the value of certain adaptations in order to select the most effective solution, and understand its performance characteristics by evaluating its experience. Finally, all of its actions must be made transparent to its user. Using Section 6.1.1 as a guide, this chapter conforms the following simple, informal working definition of a cognitive radio:

> *A cognitive radio is one in which autonomous goal-driven operation is supervised by the evaluation of its current operating environment and past experiences.*

6.2 Theory

A radio can have multiple objectives other than simply the reliable transmission of data. Goals such as increasing battery life by reducing power consumption, occupying less spectrum, and conserving static memory are all examples of radio resource management (see Section 5.1). While it is possible that many solutions exist to achieve a particular desired performance, the engine itself must be capable of evaluating each solution to determine which is optimal. In this section I formalize the radio's operation as a classic non-linear multi-objective optimization problem and demonstrate exactly how the engine can reduce its search space without loss of generality.

6.2.1 Subspaces

In order for the radio to make informed decisions about its adaptation strategy, the radio must first retain knowledge about its experiences. There are three primary subspaces to cognitive radio information: *utilities*, *observables*, and *parameters*. The design of the engine itself abstracts from explicitly specifying these subspaces in the context of the radio's domain. As such, a single engine design can be used for many vastly different applications whose operation can be formalized as a classic non-linear optimization problem; the radio must maximize a set of goal-driven utilities $\boldsymbol{u} = \{u_0, u_1, \ldots, u_{N_u-1}\}$ by adjusting a set of parameters $\boldsymbol{\theta} = \{\theta_0, \theta_1, \ldots, \theta_{N_\theta-1}\}$ subject to a set of observable operating conditions $\boldsymbol{\phi} = \left\{\phi_0, \phi_1, \ldots, \phi_{N_\phi-1}\right\}$.

The *utilities* supply the engine with quantitative information about system performance in relation to the engine's goals. The engine's utilities are directly related to the radio's performance metrics, $\boldsymbol{\beta} = \{\beta_0, \beta_1, \ldots, \beta_{N_u-1}\}$ through a monotonic utility function. For example, if one goal of the engine is to achieve a data rate of at least 100 kbps, yet the established link is only supplying 25 kbps, then resulting utility should reflect this poor performance. The utility does not necessarily (and perhaps *should not*) scale linearly with performance.

The *observables* (or "operating conditions") are quantities relating to the environment over which the radio has no control. These should provide a complete scenario description to the engine and can amount to such metrics such as path loss, Doppler shift, or fading index. As a good design principle, the observables should be independent of the radio's parameters and should only reflect information about the operating environment itself. For example, using bit error rate as an observable would be a poor choice if the transmitter can adapt its modulation scheme, as the two are strongly related to one another. As a result, the observable is linked to its parameter set, making the search more obfuscated. Similarly, using signal-to-noise ratio is again a poor choice if the transmitter can adjust its radiated power. A more robust choice would be to use path loss which is only affected by the distance between the transmitter and receiver, antenna gain, and multi-path power delay profile: values over which typically the radio has no control. In fact, modification of any of the parameters should ideally have no affect on the measured environmental conditions. The reason for this is due to how the engine needs to access its knowledge base.

The *parameters* are chosen by the engine such that the utilities are maximized, given the current state of the environment. As such, the engine will need to retain its experience in order to later recall this information. It is important to recognize that the radio only has control over $\boldsymbol{\theta}$ (transmit power, modulation scheme, etc.); however, it cannot change $\boldsymbol{\phi}$ (path loss, Doppler shift, etc.). Examples of such goals include *achieve a data rate of 100 kbps*, or *maximize the link's spectral efficiency.* The solution space is strictly bounded by policy and the radio's own physical capabilities. Furthermore, parameters can take either a continuous or discrete form, and can be either ordered or unordered.

6.2.2 Mapping subspaces

The subspaces defined by $\boldsymbol{\theta}$ (parameter), $\boldsymbol{\phi}$ (observable), $\boldsymbol{\beta}$ (goal), $\boldsymbol{u}$ (utility), and v are related by their mapping functions. Consequently the N_θ-dimensional parameter space and N_ϕ-dimensional observable space are mapped onto N_u-dimensional goal space using f, viz

$$f\left(\boldsymbol{\theta}|\boldsymbol{\phi}\right) : \mathbb{R}^{N_\theta+N_\phi} \rightarrow \mathbb{R}^{N_u}$$

The mapping function f is typically non-realizable and is measured empirically. Within the context of cognitive radios, the goals are often values measured by the system itself, and include some random variation. For example, throughput of a system degrades with packet error rate, which

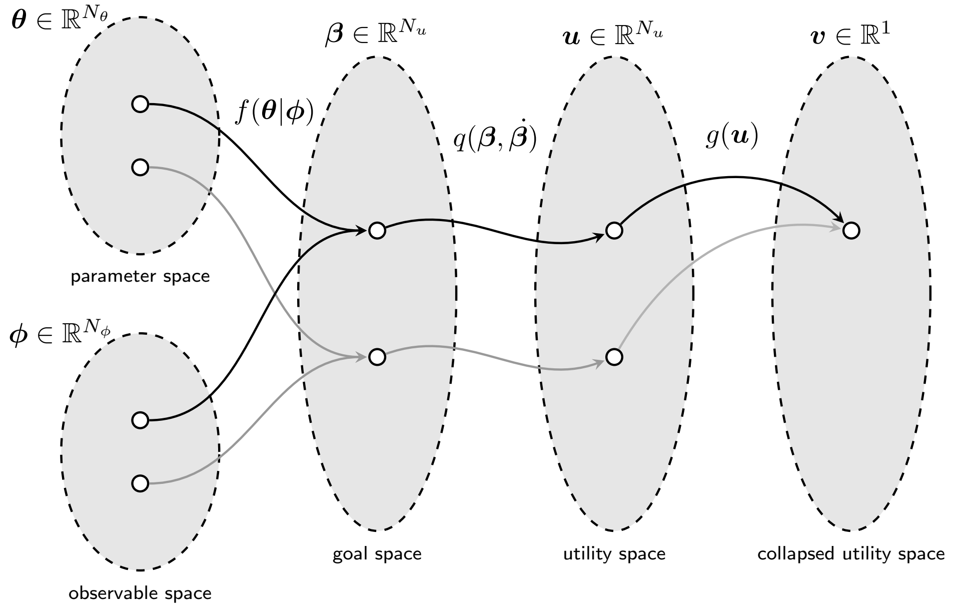

$\boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$
$\boldsymbol{\beta} \in \mathbb{R}^{N_u}$
$\boldsymbol{u} \in \mathbb{R}^{N_u}$
$\boldsymbol{v} \in \mathbb{R}^{1}$
$f(\boldsymbol{\theta}|\boldsymbol{\phi})$
$q(\boldsymbol{\beta}, \dot{\boldsymbol{\beta}})$
$g(\boldsymbol{u})$
parameter space
$\boldsymbol{\phi} \in \mathbb{R}^{N_\phi}$
goal space
utility space
collapsed utility space
observable space

Figure 6.1: Mapping optimization spaces. The mapping functions f and g are not strictly one-to-one as multiple solutions can produce the same set of utilities. The utility function q, however, has a strict one-to-one mapping between the goal and utility spaces.

can fluctuate with each measurement. The goal space is converted to the utility space through the injective utility function:

$$q(\boldsymbol{\beta}) : \mathbb{R}^{N_u} \to \mathbb{R}^{N_u}$$

It is necessary to collapse the N_u-dimensional utility space into a one-dimensional subspace so that the performance of each adaptation can be properly quantified. This is done through the simple mapping function, g:

$$g(\boldsymbol{u}) : \mathbb{R}^{N_u} \to \mathbb{R}^{1}$$

This collapsed utility space mapping function is completely user-defined and can significantly affect the engine's choice of adaptation. While a number of subspace mapping functions have been proposed, their usefulness to radio learning in the context of non-simulated data is limited. Figure 6.1 depicts the relationships between $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, $\boldsymbol{\beta}$, $\boldsymbol{u}$, v, f, q, and g through solution space mapping. Due to stochastic observations, however, the values of the observables $\boldsymbol{\phi}$ and performance metrics $\boldsymbol{\beta}$ are only estimates, viz.

$$\begin{array}{ccc} \boldsymbol{\phi} & \to & \hat{\boldsymbol{\phi}} \\ \boldsymbol{\beta} & \to & \hat{\boldsymbol{\beta}} \end{array}$$

As a result, the error due to the variance in the estimates are propagated downstream to the utility and collapsed utility spaces. This can give rise to errors in the engine's decisions if not properly accounted for. For example, let one the performance metrics, $\hat{\beta}_k$, represent the packet error rate observed over a finite number of packets. The variance of $\hat{\beta}_k$ can cause the engine to

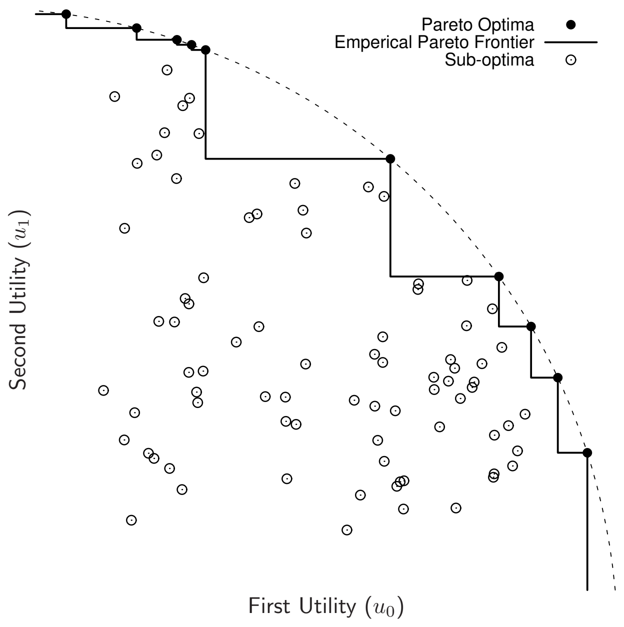


Figure 6.2: Example of Pareto efficiency with the utility space $\boldsymbol{u} \in \mathbb{R}^2$

make errors in its decisions due to imprecise information about its environment. However, with sufficient observation and knowledge collection these errors can be kept to a minimum.

6.2.3 Pareto Efficiency

The purpose of the cognitive engine is to retain empirical data about channel conditions and to search the available parameter space leading to a solution as near to optimal as possible. Both the terms *channel condition* and *optimal solution*, however, are very broad in the context of radio performance. Impairments such as multi-path fading, path loss, and interference all fall under the umbrella of channel conditions. The engine itself must handle the optimization of multiple objectives; performance of radio systems is typically focused on improving spectral efficiency, increasing throughput, and reducing errors introduced by the channel. In multi-objective optimization theory, a Pareto improvement is one in which choosing a new parameter set results in at least one utility improvement without degrading any others. A solution set is deemed to be Pareto efficient (or Pareto optimum) when no more Pareto improvements can be made. The boundary along this optimum is called the Pareto frontier. Formally, a parameter set $\boldsymbol{\theta}_k$ is Pareto efficient if and only at least one of its utilities is greater than or equal to the that of every other parameter set. This, conveniently, allows the solution space to be significantly reduced as the majority of possible solutions can easily be determined to be inefficient.

An example of a Pareto-efficient system can be found in Figure 6.2 in which a discrete parameter set yields the displayed utility space. In this example, the objective is to maximize the two utilities u_0 and u_1. The solid line connecting the Pareto efficient solutions represents the empirical Pareto frontier, providing the boundary of the performance of the system. All of the sub-optimal solutions (open circles) are inferior to any of the Pareto efficient solutions and can be discarded from the set of candidates as their performance is strictly dominated by one or more of the Pareto optima. However, if a new utility is introduced, then all of the potential solutions will need to be re-evaluated. In such a case it is possible that solutions initially deemed inferior now could provide a feasible solution. A practical example of this can be seen in Figure 5.7 in which channel capacity and computational complexity are compared against E_b/N_0 for various modulation and forward error-correction coding schemes. Considering just capacity, it seems obvious that the low complexity pairs are inferior as they require a higher bit energy to convey less data. However, as soon as the utility to minimize computational complexity is introduced, pairs such as QPSK/Hamming$(12,8)$ are re-introduced into contention because of their reduced computational requirements.

6.2.4 Utility Function: Abstracting Performance Metrics

The engine uses performance metrics to determine the quality of the current adaptation relative to the operating state of the system. In the context of radios, these usually refer to bit error rate, throughput, and occupied spectrum. These metrics, however, operate on different scales and have completely different units, giving the engine the difficult task of making a fair comparison of cumulative performance. This section introduces the utility function which maps the received metric to a normalized quantity. The intent here is to abstract the physical representation about the metrics describing the outcome of a particular adaptation to the engine. Consider, for example, an engine whose purpose is to maximize the rate of a communications link subject to minimizing the transmitter's radiated power. Is a link that achieves an average throughput of 100 kbps twice as good as one which provides 50 kbps? Given that the radio operates with a finite resource set, were the tradeoffs made to achieve this rate worthwhile? If one of the goals were to achieve a data rate of 200 kbps then the tradeoffs were indeed probably worthwhile as the adaptation has moved the link quality significantly closer to this achievement. The resulting utility should reflect this change, and suggest that a significant improvement has been made with this adaptation. Alternatively, if the goal was initially to create a link with just 50 kbps, then it is likely that significant resources were wasted in this adaptation, and again, the utility should reflect this.

A generic utility function used to abstract from the metrics observed by the cognitive engine has been proposed in [98]. Specifically, the utility function for a metric β is defined as

$$q\left(\beta, \dot{\beta}; \eta, \sigma\right) = \frac{1}{2} + \frac{1}{2}\tanh\left\{\sigma\left[\log\left(\frac{\beta}{\dot{\beta}}\right) - \eta\right]\right\} \tag{6.1}$$

where $\dot{\beta}$ is the target metric, σ is the spread parameter, η is the threshold parameter, and $\tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$ is the hyperbolic tangent. Therefore (6.1) is monotonic and strictly bounded by $q \in (0,1)$. The threshold and spread parameters can be chosen arbitrarily and govern at exactly what utility value $q(\beta)$ gives when the metric β meets its target. More specifically, choosing $q(\dot{\beta}) = q_0$ and $q(p\dot{\beta}) = q_p$ results in

$$\sigma = \frac{1}{2\log p}\left[\log\left(\frac{1}{q_0} - 1\right) + \log\left(\frac{q_p}{1-q_p}\right)\right] \tag{6.2}$$

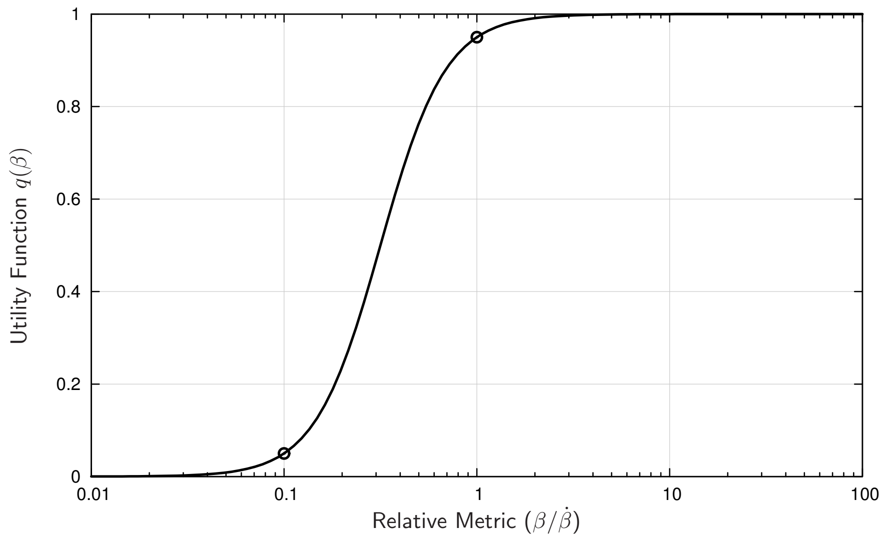


Figure 6.3: Generic utility function for metric abstraction, defined by (6.1) for a target of 0.95 when $\beta = \dot{\beta}$ and 0.05 when $\beta = \dot{\beta}/10$.

and

$$\eta = \frac{1}{2\sigma} \log\left(\frac{1}{q_0} - 1\right) \tag{6.3}$$

Notice that there is no restriction on the choice of q_0, q_p, and p except that $p \neq 1$ and q_0, $q_p \in (0, 1)$, $q_0 \neq q_p$. This flexibility permits total control over the shape, transition region, and slope of the utility function. For example, choosing the utility to be 95% when the metric achieves the target ($\beta = \dot{\beta}$) and 5% when the metric is one decade below ($\beta = \dot{\beta}/10$) gives $q_0 = 0.95$, $q_p = 0.05$, and $p = 0.1$. Inserting these numbers into (6.2) and (6.3) gives $\sigma = 1.27875352$ and $\eta = -1.15129256$, respectively. The resulting utility function is plotted in Figure 6.3. Notice that η is negative which implies that $p < 1$ and the knee defined by $\beta = p\dot{\beta}$ is below $\beta = \dot{\beta}$. If $p > 1$ then the shape of the utility function is reversed.

The design choice of having a monotonic function begs the question: *why not adopt a utility function which is maximized by the metric meeting its goal?* This could be accomplished by adopting a utility such as the normal curve $q = \exp\{-(\beta - \dot{\beta})^2/2\sigma^2\}$ which achieves a maximum of 1 at $\beta = \dot{\beta}$. The reason why this is avoided relates to the injective nature of the utility function which allows its input/output mapping to be unique between domains. This helps significantly with reasoning within the engine which would not be able to distinguish between a poor utility due to a metric being too low from a poor utility due to a metric being too high. One might think that a bell-shaped utility function has a multi-objective optimization nature built into it. For example, if β_k represents the throughput of a link with the goal $\dot{\beta}_k$=100 kbps, then providing a data rate of 200 kbps must be in some way wasting resources; This demonstrated in Chapter 5 for radios adaptively switching modulation and forward error-correction codes. Properly designed engines, however, will weigh all of the utilities proportionately and make a decision as to appropriately trade the improvement of one utility for another. It is therefore necessary to choose sufficient utilities within the engine itself which encompass its true objectives. For this example, another utility chosen to minimize radiated power at the transmitter might outweigh the benefit of the additional data rate. It is well known that one cannot simultaneously decrease transmit power while increasing channel capacity (see Figure 5.7(a)); consequently, one utility will increase while the other decreases. To put it another way, it is up to the engine to decide if the cost attributed to increasing one utility only marginally is worth the cost of decreasing another substantially.

The benefits of using the utility function in (6.1) are highlighted below.

1. The shape of $q(\beta)$ is monotonically increasing which improves convergence of most search algorithms. Because it is never perfectly flat (its derivative is always positive), $q(\beta)$ has a strict one-to-one mapping to its input utility. Furthermore, this injective nature prevents gradient-based search algorithms from hitting a plateau, increasing the speed of convergence. It also assists in the engine's reasoning capabilities by allowing it to distinguish between unique performance characteristics.

2. The utility function can be parametrized to describe the desired relationship between any number of metrics and the respective goal of the system. Specifically, when $p < 1$ it is used to describe metrics which need to be maximized (throughput, spectral efficiency, etc.) and when $p > 1$ its shape is inverted and describes metrics which need to be minimized (transmit power, bit-error rate, etc.). The steepness of the curve can also be adjusted, and provides emphasis on the relative importance of meeting a particular goal.

3. Because (6.1) accepts the ratio of the input metric to its goal, the utility function abstracts any dependency of units from the resulting utility. Furthermore, $q(\beta)$ is strictly bounded by the interval $(0, 1)$ which provides a standardized output scaling to the engine. This is particularly useful in problems involving multiple objectives where the resulting utilities need the quantification abstracted from the metrics they represent.

4. By forcing an unattainable upper bound, the utility function suggests to the engine that there is always room for improvement, even if the target metric has been achieved. The particular shape of $q(\beta)$, however, provides a diminishing marginal return on the utility for its particular metric.

6.2.5 Production Function: Collapsing the Utility Space

The dimensionality of radio parameters can be staggering, providing such a massive number of combinations that it would be impractical to retain information about each one. However, once the solution space is pruned to its Pareto efficient candidates, the search process becomes more manageable. The issue now becomes one of choosing an appropriate solution set from within those lying on the Pareto frontier. A classic example given in the literature involves a company which must choose a processes for manufacturing a product. The company's goal is to choose a process which maximizes both manufacturing ability (labor) and production (capital goods). The two utilities, however, have an inverse relationship to one another on the system's Pareto frontier; one cannot simultaneously increase labor while also increasing capital. There exists an inherent trade-off between the two utilities, and consequently the company must choose a process which serves its purposes accordingly by compromising one utility for improvement of the other. In neoclassical economics, the output quality of a firm can be specified by a production function which accepts a set of input quantities (e.g. capital, labor, raw materials).

Similarly, the cognitive engine can have contradicting utilities. For example, keeping all other factors at a constant, it is impossible to reduce the occupied spectrum while increasing the capacity of the channel (see Figure 5.1). The engine must therefore compromise some utilities for the sake of the improvement of others. The collapsed utility space exists strictly in $\mathbb{R}^1$. The mapping function $g : \mathbb{R}^{N_u} \to \mathbb{R}^1$ to select the appropriate solution is therefore open to choice. The collapsed utility may be therefore computed by taking the weighted geometric mean of all utilities, viz.

$$v = g(\boldsymbol{u}) = \left[\prod_{k=0}^{N_u-1} q(\boldsymbol{u}_k)^{\alpha_k} \right]^{1/N_u} \tag{6.4}$$

where $\alpha_k > 0$ denotes the relative importance of utility u_k. In essence the collapsed utility is considered a *global* utility for the system, providing a balanced metric for comparing potential solutions. The equation in (6.4) is quite similar to the Cobb-Douglas production function to relate inputs to outputs–probably the most widely used functional representation in neoclassical economics [87]. By assigning a multiplicative sub-space mapping onto the collapsed utility space, the result is inherently fair. That is, if any one particular utility is low, so is the global utility v. By contrast consider an additive production function $v = \sum_{k=0}^{N_u-1} \alpha_k q(u_k)$ where $\sum_{k=0}^{N_u-1} \alpha_k = 1$; this function tends give an unbalanced preference to utilities with a higher importance while completely ignoring others. The drawbacks of selecting this type of production function are further discussed in Section 6.4.1. Because $q(\boldsymbol{u}_k)$ is conveniently bounded by the interval $(0, 1)$, so will v be bounded by this same interval, regardless of the values of the weighting factors α_k. This helps to compare potential solutions by providing a normalized reference. Because the engine only needs to compare solutions by observing relative values of v, the logarithm of v may be computed for simplicity:

$$\log(v) = \frac{1}{N_u} \sum_{k=0}^{N_u-1} \alpha_k \log q(\boldsymbol{u}_k) \tag{6.5}$$

Note that it is not necessary for the weighting factors α_k to sum to unity, nor are their individual values required to be less than one. In fact the only constraint on the weighting factors are that they must be greater than zero. This ensures that utilities with higher values of α are given a higher preference. However, the examples given in this chapter impose that $0 < \alpha_k < 1$ and $\sum_{k=0}^{N_u-1} \alpha_k = 1$ for consistency. Conveniently for a system with two goals the collapsed utility v will equal the individual utilities u_0 and u_1 when $\alpha_0 + \alpha_1 = 1$. Therefore when $u = u_0 = u_1$,

$$g(\boldsymbol{u}) = u_0^{\alpha_0} u_1^{\alpha_1} = u^{\alpha_0 + \alpha_1} = u$$

Furthermore, v will be at a maximum where $u_0 = u_1$ if and only if $\alpha = \alpha_0 = \alpha_1$.

Example #1: Bandwidth vs. Capacity

Consider a communications link which operates at a fixed transmit power and has control over a single parameter, bandwidth ($\theta_0 = W$). The link undergoes a constant path loss, which translates to a static ratio of energy per bit to noise power at the receiver ($\phi_0 = E_b/N_0$). The system has two goals, each with a target: maximize the channel capacity ($\beta_0 = C$) while achieving a minimum rate ($\dot{\beta}_0 = 200$ kbps), and minimize occupied bandwidth ($\beta_1 = W$) while not exceeding a maximum ($\dot{\beta}_1 = 100$ kHz). The resulting utilities from these performance metrics (u_0 and u_1, respectively) are weighted such that maximizing the rate of the link is given significantly more emphasis than minimizing occupied bandwidth ($\alpha_0 = 0.8$ and $\alpha_1 = 0.2$).

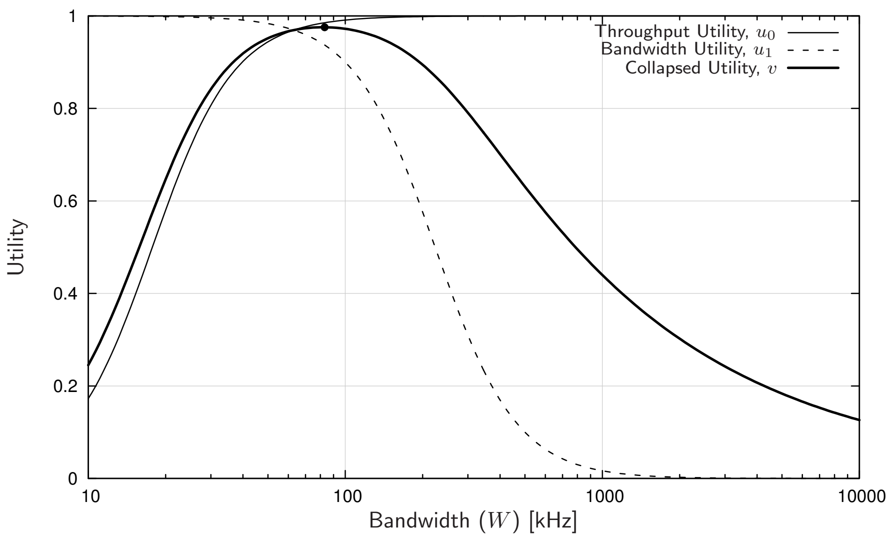
The channel capacity can be computed by solving the following equation for the rate C/W [76, (7.1-34)] measured in b/s/Hz (plotted in Figure 5.7(a))

$$\frac{C}{W} = \log_2\left(1 + \frac{C}{W}\frac{E_b}{N_0}\right) \tag{6.6}$$

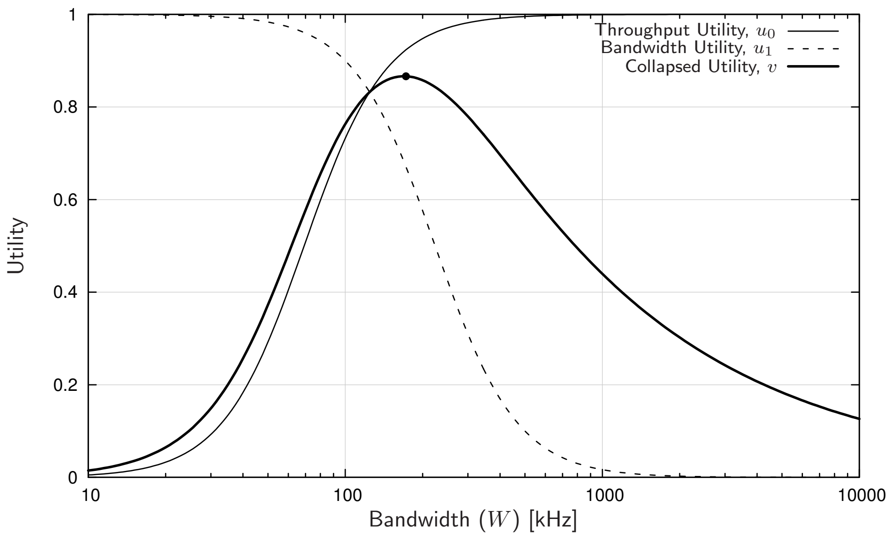
which exhibits a theoretical lower limit at $E_b/N_0 = 10\log_{10}(\log(2)) \approx -1.6$ dB. Figure 6.4 demonstrates the production function under two separate scenarios. In the first case, the channel quality is high as $E_b/N_0 = 8$ dB. The resulting production function achieves its maximum at $W = 83.1$ kHz which corresponds to a capacity of 418.4 kbps. In this particular case both the target metrics were achieved and the shape of the production function is quite wide with its maximum at $v = 0.976$, as can be seen in Figure 6.4(a). In the second case, however, the channel quality is low as $E_b/N_0 = 1/2$ dB which gives a theoretical maximum capacity at the upper bandwidth limit of 100 kHz is only 129.3 kbps. Consequently it is not possible to meet both target metrics. Given that the system emphasizes the first utility (maximizing throughput) over the second (minimizing bandwidth), the maximum of the utility function is achieved at a bandwidth of 172.3 kHz which yields a data rate of 222.8 kbps. In this case, the shape of the production function is narrower with its maximum at only $v = 0.866$, as can be seen in Figure 6.4(b). Notice that the capacity utility u_0 has shifted considerably to the right from the previous example. Further degradations in E_b/N_0 cause the production function to degrade quickly, sacrificing even more bandwidth for data rate.

Example #2: Bandwidth and Power vs. Capacity

Considering the results gathered from the first example, it is interesting to observe the effects of allowing the transmitter to adjust its power as well as its bandwidth. This example uses a two-parameter system: transmit power (θ_0) and bandwidth (θ_1). From a theoretical standpoint, the capacity of the channel increases logarithmically with transmit power and can increase infinitely. In a practical scenario, a wireless system is limited by the amount of power it can transmit; Furthermore, from a standpoint of efficiency, it is usually desirable to minimize the amount of power radiated by the transmitter. Consequently, a new utility is introduced with a goal to minimize transmit power. The system now has three goals: maximize the channel capacity ($\beta_0 = C$) while achieving a minimum of $\dot{\beta}_0 = 100$ kbps, minimize occupied bandwidth ($\beta_1 = W$) while not exceeding a maximum of $\dot{\beta}_1 = 100$ kHz, and minimize the transmit power ($\beta_2 = E_b/N_0$) while not exceeding a of $\dot{\beta}_2 = 10$ dB. As before, the channel capacity is computed by solving (6.6) for C/W and then multiplying by the bandwidth. The relative weights were given this time as $\alpha_0 = 0.5$, $\alpha_1 = 0.3$,



(a) $E_b/N_0 = 8$ dB, $\beta_0 = 418.4$ kbps, $\beta_1 = 83.1$ kHz



(b) $E_b/N_0 = 1/2$ dB, $\beta_0 = 222.8$ kbps, $\beta_1 = 172.3$ kHz

Figure 6.4: Collapsed utility space example vs. occupied bandwidth $\theta_0 = W$ given the observable $\phi_0 = E_b/N_0$ where u_0 and u_1 represent maximizing throughput for the channel and minimizing occupied bandwidth, respectively. The target metrics are $\dot{\beta}_0$ =200 kbps and $\dot{\beta}_1$ =100 kHz.

and $\alpha_2 = 0.2$ for the three respective utilities. Figure 6.5 demonstrates the production function. The corresponding peak is $v = 0.977$ at $W = 69.0$ kHz and $E_b/N_0 = 8.2$ dB which yields a rate of $C = 353.7$ kbps. Notice that for this particular scenario, all three targets were met.

6.3 Components

The cognitive engine can be formally dissected into four primary components: the *knowledge base*, *performance monitor*, *fitness estimator*, and *adjudicator*. It is not imperative to discuss either the implementation of each component or the interaction between them. Instead, the discussion shall be limited to the distinct purpose of each.

6.3.1 Knowledge Base

As discussed in Section 6.1.1, the first category of Bloom's Taxonomy of the cognitive domain is knowledge. The ability to recall information is the first and most critical step towards radio cognition. Knowledge retention can be realized on software-defined radio platforms by any number of methods, including look-up tables, random access memory, filters, databases, and artificial neural networks. In the context of cognitive radios it is perhaps most logical to store events in a case database in which the performance of a particular adaptation under a scenario is retained. This is an important aspect to radio learning as recalling previous experience is paramount to adapting quickly to changing conditions. Given the previous discussion of optimization, each case should accurately represent the sets parameters, observables, and utilities.

6.3.2 Performance Monitor

The learning process is enabled on intelligent systems by relaying information about the current solution state to its adaptation algorithm. The potential of each adaptation is valued in relation to achieving each goal, typically realized through a well-defined utility function. The performance monitor retains this information in the cognitive engine's internal knowledge base.

6.3.3 Fitness Estimator

Because both the utilities and the operating conditions are measured values and therefore stochastic, the result of an adaptation cannot be determined by direct observation alone. The fitness estimator provides an estimate of how a particular adaptation will perform for the operating condition of the radio. This aspect of cognition is extremely important for the efficient convergence of non-linear optimization techniques. Evaluating the performance of each candidate parameter set by direct observation is inherently flawed in two ways:

1. Actual fitness measurements require considerable observation time. This in itself is impractical as channel conditions in dynamic spectrum environments are changing too quickly for the fitness of any particular adaptation to be reliably observed;

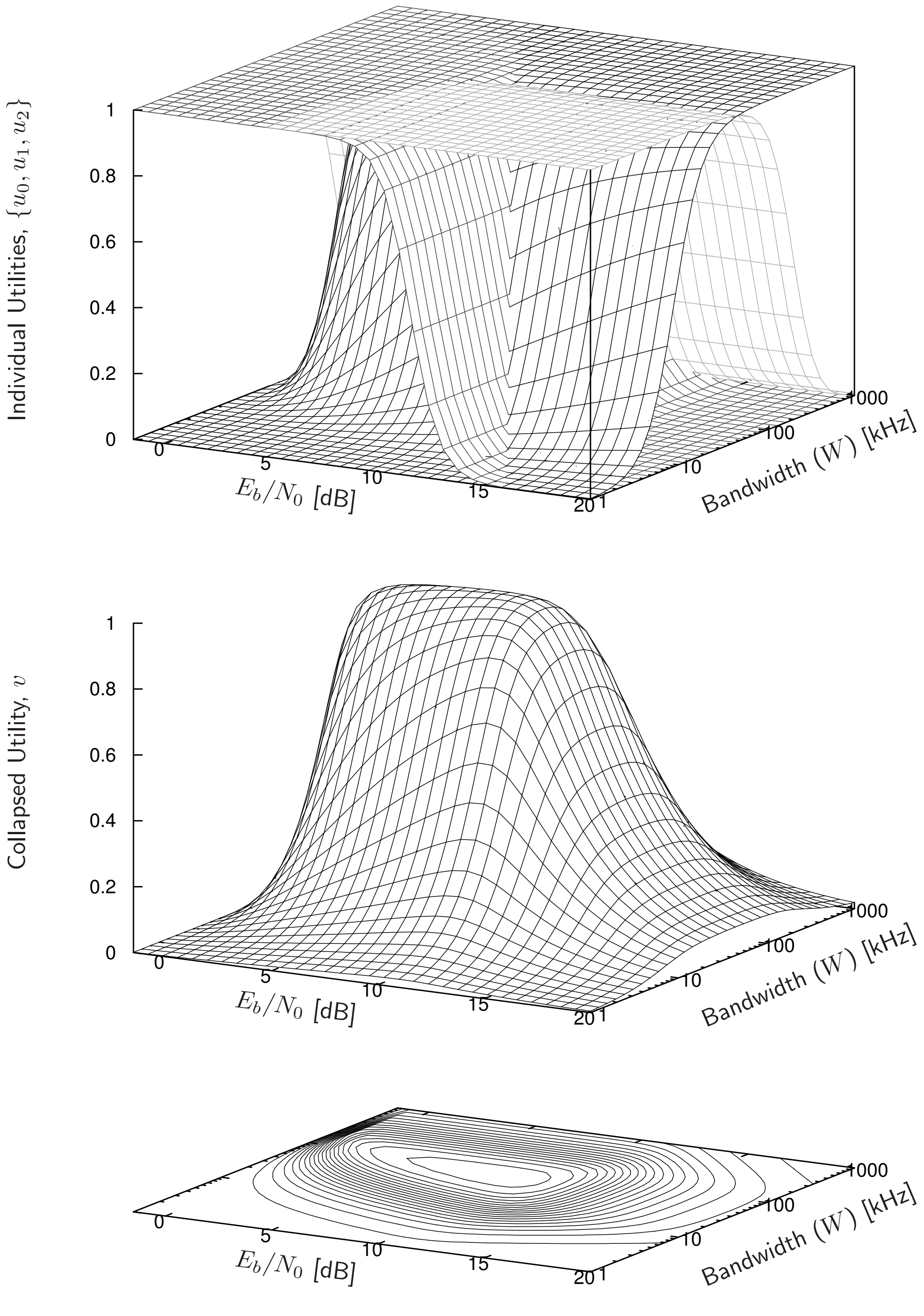


Figure 6.5: Collapsed utility space example: power and bandwidth vs. capacity where u_0, u_1, and u_2 represent maximizing throughput, minimizing occupied bandwidth, and minimizing transmit power (E_b/N_0) with targets $\dot{\beta}_0 = 200$ kbps, $\dot{\beta}_1 = 100$ kHz, and $\dot{\beta}_2 = 10$ dB, respectively.

2. In the presence of other cognitive radios, channel state conditions are exacerbated. This is an artifact of the channel state being a non-ergodic process, under which condition the observation is fallacious.

Direct-observation feedback methods eliminate classical optimization techniques such as Newton-Raphson which assume a smooth error surface. Figure 6.6 demonstrates the importance of fitness estimation in cognitive radios. The observed of the relationship between the operating parameters θ_1 and θ_2 and the fitness function $F(\theta_1, \theta_2)$ is noisy, as depicted by Figure 6.6(a). The fitness estimator function $\zeta(\theta_1, \theta_2)$ maps the observation to a continuously smooth surface, Figure 6.6(b). The mapping can be achieved by any number of methods, particularly multi-dimensional polynomial curve-fitting or artificial neural networks [43, 58, 82, 23]. This permits the use of classical non-linear optimization techniques such as Newton-Raphson, gradient-based, and hill-climbing search algorithms.

6.3.4 Adjudicator

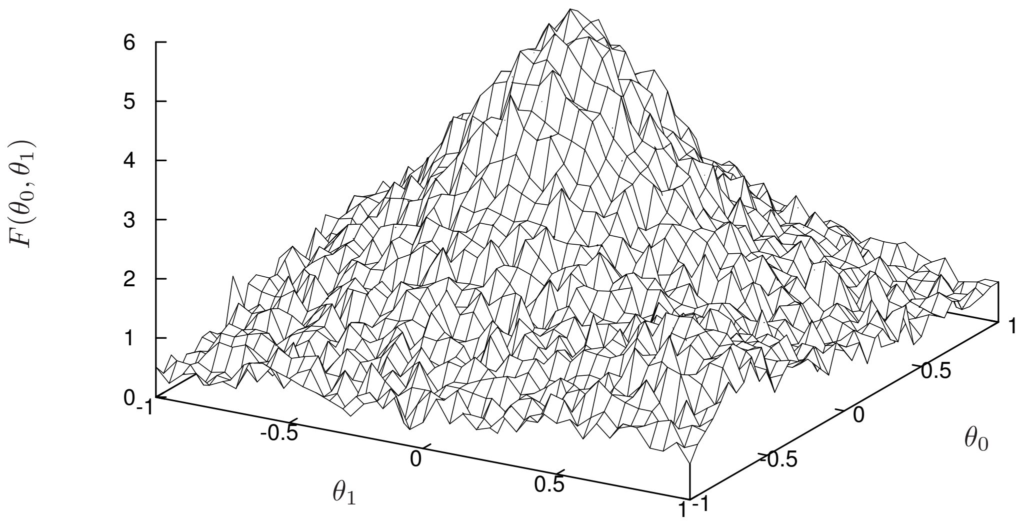
The adjudicator is the decision-making entity of the cognitive engine. It's purpose is to balance the tradeoffs observed in multi-objective optimization problems prevalent in cognitive radio scenarios. This is commonly implemented by the weighted combination of the output of individual utility functions [98, (6)]. Section 6.2.5 has shown how merging multiple utilities collapses its quantitative representation into $\mathbb{R}^1$.

6.3.5 Policy Module

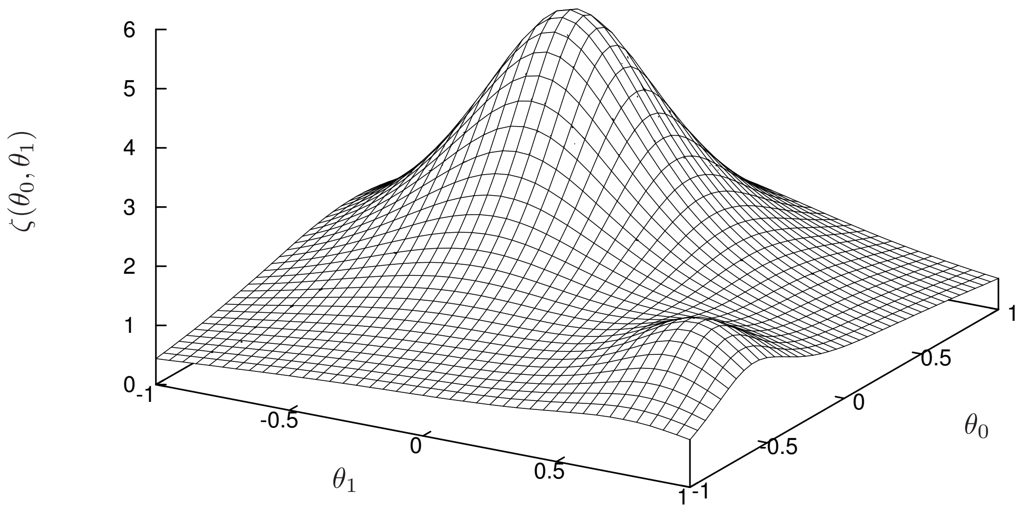
As cognitive networks are proposed to overlay legacy system bands, such radios must comply with spectrum policy to minimize electromagnetic interference to primary users. The amount of interference to which primary users can handle before legacy systems exhibit a noticeable quality of service degradation is a point of contention between regulators and standards committees. The Federal Communications Commission strictly limits the amount of transmit power allowable on particular frequency bands, as well as the permissible modulation and multiplexing schemes. For example, the IEEE 802.22 standard dynamically adjusts the transmit power spectral density mask depending upon the primary users (television broadcast stations, Part 74 devices, wireless microphones, etc) [51]. The policy module holds information about the admissible system parameters, based on location, time of day, and current spectrum occupancy. The radio environment map (REM) has been suggested to drive the information stored within the policy module and is particularly useful for wireless mobile networks [99, 98].

6.4 Evolution of the Cognitive Engine Architecture

There exist nearly as many cognitive radio designs as there are applications for them. This section details several cognitive engine development projects designed for software radio platforms and



(a) Fitness Observation



(b) Fitness Estimation

Figure 6.6: Fitness function F and its estimator ζ for a two-parameter system

provides the motivation behind the case database-enabled cognitive engine architecture proposed in Section 6.5.

6.4.1 CoRTekS

A preliminary cognitive radio design was created in 2005 using an artificial neural network-based cognitive engine to adjust modulation, power, and carrier frequency, all while complying with a programmable user-defined policy that restricted the permutation of each of these parameters. The radio itself was constructed from Tektronix test equipment and a personal computer. As such, it was dubbed *CoRTekS* for **co**gnitive **r**adio **Tek**tronix **s**ystem. The transmitter was built from a Tektronix arbitrary waveform generator AWG430 acting as baseband generator, a Tektronix AWG710B as the local oscillator, and a variety of external SMA components such as mixers, amplifiers, filters, cables, and a UHF antenna. Together, these components acted as a flexible RF transmitter, capable of switching carrier frequency, modulation scheme, and transmit power. A Tektronix real-time spectrum analyzer 3408A and another UHF antenna acted as the receiver, capable of both signal demodulation and spectrum sensing. A personal computer running an early version of Virginia Tech's open-source software communications architecture implementation, OSSIE, [73] acted as the waveform controller. All the components were connected using a set of general purpose interface bug cables to the host PC which could control both the transmitter and receiver independently. The data transmitted over the air represented a bit-mapped image of a chameleon sitting on a limb. A graphical user interface was designed to display the policy, spectrum usage, current adaptation, performance, as well as both the transmitted and received images; as errors were received, the image would be corrupted allowing a relatively simple visualization of the radio's performance.

In order to demonstrate spectrum robustness an RF emitter was designed consisting of a battery and a filtered resistor network connected to a voltage-controlled oscillator and a line-matched antenna. The emitter acted as a narrowband jammer and could be tuned from the operating range of the radio, 480-490MHz, using a potentiometer. Because the transmit power of the radio was so low, the jammer, when enabled and tuned to the radio's center frequency, could easily disrupt the communications link. When the receiver detected this resulting increase in error rate, the engine would switch into a search mode, disabling the transmitter and putting the RTSA into its spectrum analyzer mode. A snapshot of the received spectrum was then sent to the engine which would then estimate which frequency bands were being occupied by primary users. This detection process simply used the received power spectral density to estimate which frequencies exhibited power above the estimated noise floor. A simple search algorithm then determined the best available frequency slot (if any existed). In fact, the engine would request that the radio enter its search mode periodically, even if the received image had not been corrupted. This intermittent sensing enabled the engine to continually monitor spectrum usage and ensure that it was not interfering with incumbent devices.

After the appropriate channel selection, the engine would choose the best transmit power and modulation scheme; by adjusting these parameters the engine could improve upon its utilities to improve spectral efficiency, decrease the bit error rate, and minimize transmit power (used to keep the amount of interference the radio imparts to incumbent devices to a minimum). The engine itself was built upon an artificial neural network and a case database. The concept of the engine was to use the database to retain the radio's experiences by storing the resulting utility of its previous decisions given the channel state condition. The artificial neural network was used to interpolate between knowledge points and acted as the fitness estimator in the engine. The engine observed the channel conditions by employing a channel reliability metric which gave an indication of the stability of the link. When the channel was stressed by interference, the metric was decreased; otherwise its value was steadily increased. The metric itself was a combination of the time-averaged bit error rate measured at the receiver as well as the amount of interference measured by the spectrum manager. The neural network accepted three inputs: one observable and the two parameters:

ϕ_0 : the channel reliability metric

θ_0 : the transmit power of the receiver, measured in dBm

θ_1 : one of five available modulation schemes: QPSK, 8-PSK, 16-QAM, 32-QAM, or 64-QAM

These inputs fully described the capability of the system in regards to the current status of the link. The outputs of the network were defined as the three utilities

u_0 : decrease the bit error rate

u_1 : improve spectral efficiency, measured in b/s/Hz

u_2 : minimize transmit power

The utilities were computed by pushing the resulting three empirical metrics (BER, spectral efficiency, and transmit power) through a normalization function similar to that of (6.1) such that they were strictly bounded by $[0, 1]$. The case database stored the six-element state/decision/outcome vector[1] as

$$s_k : \{\phi_0,\ \theta_0,\ \theta_1,\ u_0,\ u_1,\ u_2\} \tag{6.7}$$

in a first-in/first-out queue, retaining only the most recent N vectors. When the radio operated in its search mode, it would train the neural network on its experience collected in the case database. Once complete, the engine would perform a search over the network by adjusting θ_0 and θ_1 while holding the empirical channel reliability metric ϕ_0 a constant. The resulting output solutions were compared using an additive weighting function of their utilities as $u = 0.5u_0 + 0.3u_1 + 0.2u_2$; the adaptation producing the maximum global utility was chosen. To prevent over-training as well as promote the exploration of additional solutions, the engine would intermittently add a random permutation to the resulting parameter set. Introducing random perturbations into the decision of the engine caused it to periodically explore solutions which could potentially result in better performance, driving the engine to a more well-balanced knowledge base. This draws strong parallels to genetic algorithm theory which permits its potential solution sets to merge

[1]Retaining the outcome of a particular decision given the channel state became the precursor to this work in attempt to fully describing the scenario to the cognitive engine, and later became the *parameter*, *observable*, *utility* subspace set that are described in Section 6.2.1. This analysis is congruent with the analysis of Kolodner who suggests in [61] that all cases should include at least a *problem-situation description*, a *proposed solution*, and an *outcome*. Section 6.5.2 gives a more detailed explanation of case representation.

("crossover") and transform ("mutation"). As both the transmitter and receiver were connected to the same machine, feedback between them was relatively simple. In a true wireless network, feedback takes more effort on the part of the receiving node in that it must convey to the transmitter the results of its adaptation process. As Section 6.4.3 will discuss, this can be accomplished using a packet feedback response mechanism between master and slave nodes, each capable of both the transmission and the reception of data.

The radio was able to learn its mistakes and adapt to them, even with no prior understanding of the relationships between radio parameters. This learning process, however, was relatively slow, and required a considerable amount of experience before the radio was able to adapt in accordance to environmental changes. As a result, the database was initialized with a knowledge set representing pre-defined solutions. One of the key aspects of this experiment was that while the artificial intelligence can be quite simple and yet provide significant gains, it also reinforced the need for exhaustive training with simulation before applying real-world experience. Additionally, the case database needed constant pruning as it contained a number of contradictory entries; due to interference and noise, the same parameter set under similar channel reliability observations could produce different outcomes. This was exacerbated when link would undergo channel impairments outside the scope of the engine's ability to measure. For example, if someone were to touch or move the receiving antenna, the received signal strength would visibly drop on the RTSA causing an increase in bit error rate. However, because the engine had no way to measure signal degradation (aside from the consequential increase in errors), the resulting drop of service quality would contradict an earlier adaptation. This was in fact the reasoning behind introducing the channel reliability metric ϕ_0 in the first place as it was intended to incorporate any aspect of signal degradation, regardless of the reason. Furthermore, any contradictions within the case database was to be handled explicitly by the artificial neural network which could "smooth out" radical jumps in case entries. Clearly the engine needed an enhanced situational awareness, and required more sophisticated information about the link's dynamic channel conditions.

Interestingly, another drawback existed when link conditions were ideal; when running for an extended period after having found a global optimum would result in wiping the case database's knowledge about other adaptations, causing the neural network's training set to under-represent the engine's experience. As a result the engine would eventually "forget" its past experience and lose its ability to adapt to environmental changes, even the radio was presented a situation it had previously encountered. Furthermore, increasing the size of the database would cause the adaptation time for the engine to increase dramatically. Information stored in the case database needed to be retained longer than just in a first-in/first-out manner. This has motivated the design of the case database-enabled cognitive engine, described in Section 6.5.

Although one of its goals was to minimize transmit power, the radio often chose the highest modulation type available with the largest transmit power. This is because the goals *maximize spectral efficiency* and *meet quality of service* were given a significantly higher collective utility than the *minimize transmit power* goal. The radio thus determined that the small reduction in utility caused by increasing its transmit power was offset by the improvement in data reliability. Without other radios to contend with, the benefits perhaps warranted this decision, but a radio acting alone is a poor representation of a practical scenario. This can also be described by the neural network's tendency to try to describe the channel's reliability as a continuous variable. In a practical sense, if the channel exhibits interference or has encountered a deep fade, the channel is completely unreliable. These sudden changes make adaptivity in the radio's memory difficult; good decisions it made one moment are suddenly poor the very next. The radio thus determines that the only good way to mitigate this uncertainty is to ensure that the channel is always as reliable as possible all the time, and therefore makes a "greedy" decision, in this case, to increase transmitted power, perhaps unnecessarily. This is the drawback of applying an additive production function instead of the multiplicative one as in (6.4).

This work was sponsored by Tektronix Instruments [89] and was personally demonstrated to Tektronix's hardware research laboratory in November of 2005 as well as at the 2005 Software-Defined Radio Forum in Orange County California.

6.4.2 ETRI's 802.22 Cognitive Engine

The first cognitive engine to support the Electronic and Telecommunications Research Institute's (ETRI) implementation of IEEE 802.22 [20] was designed in 2006 [80, 98]. Using a case-based reasoning in conjunction with non-linear multi-objective optimizations, the engine allocated resources to users in the form of OFDM subcarriers, power, and adaptive bit loading. Much of the insight into designing this engine was derived from a 2005 study supported by the Army Research Office to compare and contrast various artificial intelligence algorithms in their applicability to radio resource management. The 802.22 specification gives strict guidelines as to how quickly secondary users must vacate channels occupied by incumbent devices. As such, fast adaptation was paramount to the engine's success. The conclusions of the project were that the use of evolutionary and genetic algorithms could be circumvented by seeding a simple hill-climbing search with a case database. The resulting performance was nearly optimum, achieving its solution in a fraction of the time. The engine, while effective, operated completely in a simulated environment making its applicability towards laboratory tests impractical.

6.4.3 Ciren

In 2007 the Software-Defined Radio Forum hosted its first annual student radio challenge competition [36, 44]. The challenge involved designing and implementing dynamic spectrum wireless radios capable of communicating in a network, particularly targeting emergency scenarios where communications infrastructure has been disabled. This involved implementation of the baseband digital signal processing software, as well as designing the protocol, cognitive engine, components, and waveform. The wireless capabilities of this project far exceeded those of CoRTekS, yet the embedded cognitive engine was significantly less complicated.

System Description

First response teams following major disasters such as earthquakes or floods require accurate and reliable radio links to provide voice and data communications between team members. Frequently, however, emergency teams cannot inter-communicate due to relying on fixed-frequency legacy radios. Furthermore, current systems are incapable of relaying information such as high resolution area maps, streaming audio, building floor plans, images, and other pertinent data that can significantly assist both rescue personnel and the people they are trying to help. The system, named the cognitively intrepid radio emergency network (Ciren), automatically establishes data links in the licensed family radio service (FRS) band by employing a novel rendezvous protocol while avoiding interference and primary users with dynamic frequency access algorithms. Furthermore, simple cognitive algorithms assist the radio in determining not only which channels are most likely to be unoccupied, but appropriate adaptation strategies when necessary. The Ciren radios support both point-to-point data links as well as broadcast modes all without the use of an expensive base station. Furthermore, periodic channel scanning allows the nodes to disseminate spectrum usage information to other nodes, mitigating the hidden incumbent user problem. The Ciren system is designed to be deployed on embedded and portable platforms running OSSIE [73], an open-source implementation of the Software Communications Architecture, an open industry standard. The test platform has sufficiently low power requirements as to provide a demonstrable mobile and portable cognitive transceiver.

The radios operate in a master/slave mode with either a point-to-point (PTP) connection, or in a voice broadcast (VB) mode. The protocol specifications are flexible enough to support most basic digital services over voice or data channels. One of the most difficult tasks for this problem was developing a reliable protocol for rendezvousing with the intended receiver to establish a connection, maintaining link quality, and avoiding incumbent FRS users.

Ad-hoc Rendezvous

Ad-hoc rendezvous is a difficult problem in wireless communications, particularly when no base station or reliable control channel exist. This is exacerbated by the necessity to share the precious spectrum with incumbent/primary users entering and leaving the FRS channels in a stochastic manner, making this quite possibly the most critical development for this problem. The link is first established even before the user of one Ciren radios invokes the push-to-talk button. This radio, the mater node, transmits a beacon on one of the free FRS channels. The framing structure used in this system, in fact, was a precursor to the one described in Section 3.5; however its preamble phasing pattern and payload were a fixed length, and the encoded header was slightly different. While not connected, each receiver scans the 14 FRS channels looking for the beacon signal and gathering information about channel availability. Detected Ciren signals are decoded by the slave node and correlated with its own UID m-sequence (and a special m-sequence for VB mode). The slave node sends a reply when the correlator output is high. Otherwise, after a short time, the receiver will move on to the next channel. This scanning process also allows for the radios to determine which channels are available or, if occupied, what type of user in which the channel currently resides.

Once nodes have established a connection, they immediately provide each other with a list of ordered backup channels such that if their communications link is broken–presumably due to a legacy FRS radio entering the channel–they can instantly re-connect on a different carrier. If they haven't yet initially established a connection, they have no way to determine on which frequency bands the other nodes exist. Therefore they operate in a scanning mode, regardless of the user's request to send data. This primarily serves two purposes: to sense the spectrum for primary users as well as other CIREN nodes, and to establish a connection with another node to reduce latency in communication when the user requests a data transfer. While in scanning mode, the slave node keeps a database of available channels through the use of a channel selection score for each of the 14 FRS channels. The appropriate selection score is increased if the channel is empty and decreased, otherwise. The channel selection is in fact random, but its probability distribution follows the resulting channel selection score. This means that if a channel is repeatedly found to be occupied then its score is low, reducing the probability of it being sensed again. This significantly increases rendezvous in dynamic spectrum environments between nodes which have not previously established a connection. Ultimately, this primitive spectrum sensing information is shared amongst all of the nodes in the network.

A flowchart of the operational mode in the CIREN radios can be found in Figure 6.7. This protocol allows for great flexibility in radio networks and can be extended to incorporate ad hoc network packet routing for scenarios where more bandwidth is available.

6.5 The Case Database-Enabled Cognitive Engine

The preceding sections have described the basic necessities of a cognitive engine; this section discusses practical engine design. Case-based reasoning (CBR) is an area in artificial intelligence that focuses on applying similar experience to guide the problem-solving process has been actively investigated in the fields of optimization and expert systems [1, 62]. What sets CBR apart from other artificial intelligence techniques is that rather than relying on general knowledge within a problem domain, CBR is able to use *specific* knowledge of previously-experienced solutions (cases). This distinction has proven to be exceedingly useful in the development of cognitive radios as the details of the communications link which each wireless receiver is unique. In this sense, models attempting to generalize channel conditions prove to be imprecise and lead to inferior solutions. CBR has recently been investigated for use within cognitive engines [37, 80, 98, 47], particularly to amalgamate radio learning with adaptation. This section extends the analysis from specific protocols to a generalized design strategy.

6.5.1 Issues with Radio Applications

The fact that the radio's parameters can be a various mixture of types only exacerbates the search procedure. These types include continuous/discrete, bounded/unbounded, and ordered/unordered. Discrete parameters, as the name would suggest, only account for one which takes a finite set of values. The modulation index for a linear M-ary QAM modem is an example, as $M \in \{4, 8, 16, 32, \ldots\}$. For the most part, continuous parameters are usually bounded as well—e.g. the available transmit power is typically limited not only by policy, but by the radio's own supply. Furthermore, such values are often limited to finite precision in practical systems. For example, the transmit power on an embedded platform might be capable of a range of 1 mW to 100 mW but can only be adjusted in 0.1 dB steps. Consequently it is common practice to represent continuous parameters with discrete values. For example, transmit power can take on the values $\{0, 0.1, 0.2, 0.3, \ldots, 20\}$ dBm.

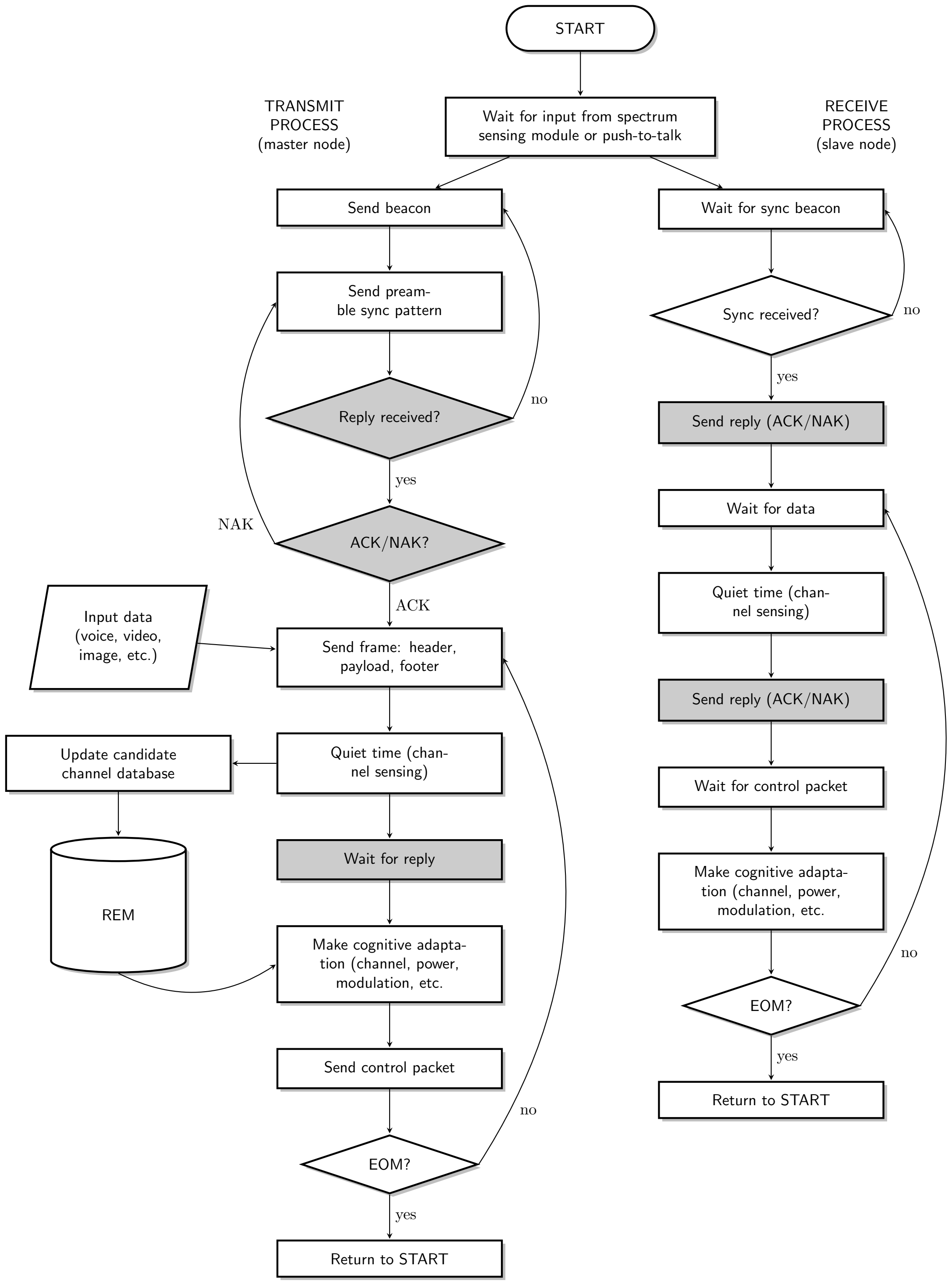

START
TRANSMIT PROCESS (master node)
Wait for input from spectrum sensing module or push-to-talk
RECEIVE PROCESS (slave node)
Send beacon
Send preamble sync pattern
Reply received?
no
yes
ACK/NAK?
NAK
ACK
Input data (voice, video, image, etc.)
Send frame: header, payload, footer
Update candidate channel database
Quiet time (channel sensing)
REM
Wait for reply
Make cognitive adaptation (channel, power, modulation, etc.
Send control packet
EOM?
no
yes
Return to START
Wait for sync beacon
Sync received?
no
yes
Send reply (ACK/NAK)
Wait for data
Quiet time (channel sensing)
Send reply (ACK/NAK)
Wait for control packet
Make cognitive adaptation (channel, power, modulation, etc.
EOM?
no
yes
Return to START

Figure 6.7: Flow chart of the CIREN protocol. Highlighted blocks are used in PTP mode only (ignored in VB mode).

Parameters can take the form of ordered or unordered values. Ordered parameters suggest that the utilities could be correlated with the value of the parameter itself. For example, increasing the payload length in number of bytes can directly improve the forward error-correction capability at the receiver while simultaneously increasing latency. The engine can therefore learn that *increasing the payload improves packet error rates but degrades link latency.* Unordered parameters, however, give no indication of correlation to utility. With the case of discrete parameters, its values take the form of a list. As an example, carrier frequency gives no direct bearing on throughput. This does not mean that throughput is independent of carrier frequency; interference in a particular channel can certainly increase errors at the receiver and reduce the link's capacity. The significance is that there does not exist a direct correlation between the value of carrier frequency and throughput.

6.5.2 Case Representation

In [61], Kolodner focuses on the content in which cases need to have to support the human decision-making process, suggesting that cases should include a *problem-situation description* (the "state of the world at the time the case was happening"), the *proposed solution* (sometimes including traces of how the problem was solved), and the *outcome* (the resulting "state of the world when the solution was carried out"). This draws parallels to the state/decision/outcome vector in which CoRTekS stored its knowledge (see section 6.4.1). Kolodner continues to suggest that the case database can additionally represent predictive features which can project the output of the case before it is even applied. There exist a wide variety of methods for representing cases in the database including feature vectors, object-oriented expressions, hierarchical representations, and even textual entries [6].

For the purposes of describing radio ontology, cases represented as feature vectors as certainly the most relevant. The case is structured by concatenating the parameters, observables, and performance metrics into a single vector, viz.

$$s_k : \Big\{\theta_0, \theta_1, \ldots, \theta_{N_\theta - 1}\Big\}\Big\{\phi_0, \phi_1, \ldots, \phi_{N_\phi - 1}\Big\}\Big\{\beta_0, \beta_1, \ldots, \beta_{N_u - 1}\Big\} \tag{6.8}$$

Notice that (6.8) uses performance metrics ($\boldsymbol{\beta}$) instead of utilities ($\boldsymbol{u}$). This is a design choice made only out of convenience–recall that the utility function $q(\beta)$ is injective and is therefore fully reversible. Section 6.2.4 explained the importance of abstracting the performance metrics' quantities from the engine and its decision. This means that adjusting the goal priorities is simply a matter of re-calculating the utilities from the performance metrics stored in the case. This also brings up an interesting point regarding which cases should be stored in the database itself. Ideally, all instances of all adaptations should be retained as they contain valuable information about each experience. Retaining all experience, however, would result in the database growing in memory beyond the capabilities of the embedded platform. Furthermore, searching the database can become exhaustively slow if its size is not properly managed. Further analysis on case database management is provided in Section 6.5.4.

6.5.3 Case Selection

Case selection is process of finding the case in the database most relevant to the current scenario. While this is a broad definition, case selection typically involves computing a distance metric between what is currently being observed in the environment and what has already been seen by the engine and stored in the database. This work defines distance between an input vector of observables $\hat{\phi}$ and the i^{th} entry in the case database is

$$D_{\phi}^{(i)} = \left[\frac{1}{N_{\phi}} \sum_{k=0}^{N_{\phi}} \left| \frac{\hat{\phi}(k) - \phi_i(k)}{\max\{\phi(k)\} - \min\{\phi(k)\}} \right|^2 \right]^{1/2} \tag{6.9}$$

where N_{ϕ} are the number of observables in each vector and $\min\{\phi(k)\}$ and $\max\{\phi(k)\}$ are the minimum and maximum values of ϕ in the database along the k^{th} dimension. This is necessary to normalize the distance metric along each dimension as the observable's values can have vastly different quantities. For example, suppose a database were designed for a wireless system such that ϕ_0 represented the path loss in dB and ϕ_1 represented the packet error rate. It is possible that the database would contain entries for ϕ_0 spanning 50 to 100 dB while ϕ_1 spanning 10^{-6} to 10^{-1}. Clearly these values require normalization before they can be aggregated into the distance metric. Retaining the maximum and minimum is trivial and in fact does not need to be updated when entries in the database are removed.

Similar to (6.9), a distance metric can be derived for the utility as

$$D^{(i)} = (1 - \zeta) D_{\phi}^{(i)} + \zeta \left(1 - g(\boldsymbol{u}_i)\right) \tag{6.10}$$

where $D_{\phi}^{(i)}$ is the case distance defined in (6.9), $g(\boldsymbol{u})$ is the production function defined in (6.4) and ζ is a parameter which controls the relative weight between the two. The introduction of ζ permits the engine to give preference to a solution which closely matches the scenario over one which has a better utility. This is an important concept for case selection, particularly when the database is sparse and does not have sufficient information to describe the situation. Suppose, for example, that the engine is presented with a scenario which it has not yet encountered and consequently does not have any relevant entries in its database. Selecting the case with the least distance will likely result in a poor utility. But simply choosing an entry based on utility alone is sure to have similarly poor results considering that the environment has been ignored altogether. The introduction of ζ in (6.10) effectively fuses the results of cases by driving the engine's choice in a direction of improvement.

6.5.4 Case Management

The previous chapters have discussed the importance of radio resource management on reconfigurable SDR platforms. One of the key requirements for embedded SDR is the reduction of the memory footprint on such devices. The case database consists simply of a dynamically reconfigurable block of memory which stores relational information about the radio's previous experiences. As entries are appended to the database unchecked, two things happen. First, the memory footprint grows beyond the capabilities of the host platform which reduces its performance. Second, the search time for the engine increases $\mathcal{O}(N)$ as each new case needs to be compared to each one existing in the database. The size of the database can be managed by periodically merging cases which are relatively similar. This reduces the necessary memory footprint for the database and never allows the number of entries to expand beyond a manageable level.

Trimming

Trimming is the simplest method for managing the size of a database by removing entries whose utilities are below a certain value. Because each adaptation is retained within the database initial solutions with poor performance will consume a considerable amount of memory. As the database matures and the engine's knowledge grows, the old entries need to be removed. Trimming the case library based on utility alone, however, has its drawbacks. The maximum global utility might not be known given the state of the channel, and scenarios where the channel is poor result in a similarly poor utility. In such a situation removing an entry based on its utility alone might result in removing the best solution given the situation.

Pruning

An improvement on database trimming is pruning; this technique searches for similar entries in the database using the metric described by (6.9). When a pair of entries are found which are too similar, the entry with the lower utility is removed from the database. This process is repeated for each entry in the database. The maximum complexity of this algorithm for a database of N entries is $\mathcal{O}(N^2)$ as each entry needs to be compared to every other entry. Realistically this complexity is pessimistic as entries which are removed from the database no longer need comparing.

The advantage pruning has over trimming is that the best entry for each scenario is retained regardless of its utility. Case pruning, however, has a significant computational increase over trimming by requiring all the case entries to be compared to one another. The complexity can be reduced by choosing a random selection of entries for comparison rather than pruning all entries at a time. Pruning has the additional drawback of perpetuating solutions with falsely optimistic utilities. Consider a solution for a given observation set which initially gives a high utility. Subsequent applications of the same solution might not yield such a high utility due in part to immeasurable quantities and otherwise random factors. However the database will always retain the initial case due to its high utility, even if it wasn't actually good on average. Ideally the cognitive engine should realize that the initial adaptation was optimistic and decrease the stored utility as a consequence. This is effectively what case merging does and is discussed in the following section.

Merging

Case merging operates similarly to pruning; however, instead of simply retaining the case entry with the best utility all nearby entries contribute to a new case. The steps for case merging are as follows:

- The case $\mathcal{C}_k$ at index k is chosen as an initial starting point.

- Using the distance metric given by (6.9), all entries within a specified distance D_{min} in the database are selected to be in the subset $\mathcal{S}_k$.
- A new case $\tilde{\mathcal{C}}_k$ is generated by averaging each of observables and each of the performance metrics in $\mathcal{S}_k$; the parameters from the case with the best utility in $\mathcal{S}_k$ are retained in $\tilde{\mathcal{C}}_k$.
- All the cases within $\mathcal{S}_k$ are removed from the database and $\tilde{\mathcal{C}}_k$ is retained.
- The process is repeated for all cases in the database.

Merging has the benefit of smoothing the resulting performance metrics by filtering the result of each adaptation with similar cases. Merging existing cases together not only reduces the database size by eliminating redundant cases but it additionally provides the effect of the fitness estimator (see Figure 6.6).

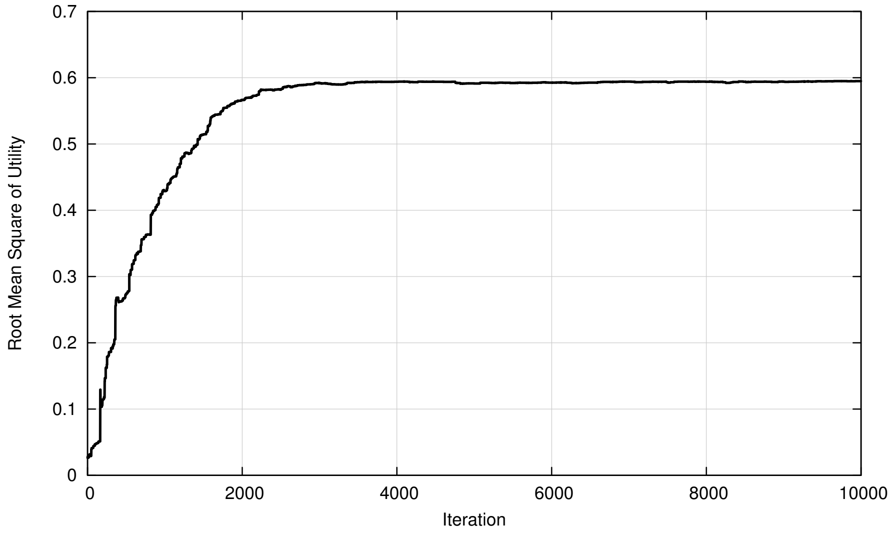
6.6 Experiment 1: Global Optimization

For the sake of demonstrating the efficacy of the proposed engine, this experiment provides a basic conditional optimization problem. Consider a two-parameter system with just one observable and one utility:

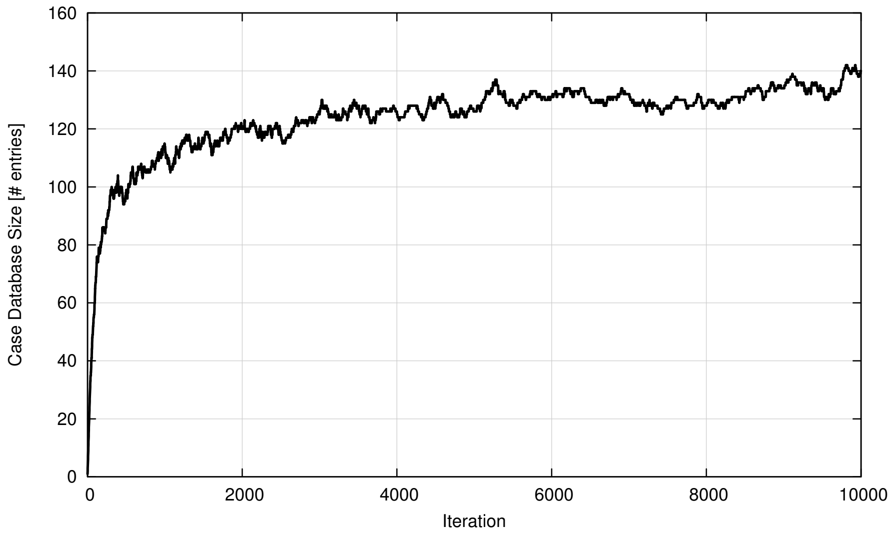
$$u = u_{max} \cdot \exp\left\{-\left[(\cos\phi - \theta_0)^2 + (\sin\phi - \theta_1)^2\right]/2\sigma^2\right\} \tag{6.11}$$

where $\sigma = 0.3$ is a constant spreading factor and $u_{max} = \exp\{-0.80472(1 - \cos\phi)\}$ is the maximum possible utility given ϕ. The shape of (6.11) is a two-parameter Gauss curve having a peak which lies on the circle defined by $\theta_0 = \cos\phi$, $\theta_1 = \sin\phi$. Furthermore, the value of the utility at this optimum is a function of ϕ; a maximum of 1.0 when $\phi = 0$ and a minimum of 0.2 when $\phi = \pm\pi$. This choice was made to emulate a practical situation where the constraints of the system limit the optimal utility. Throughput in wireless communications links, for example, degrades significantly with received power, and therefore the resulting best solution might provide a poor quality of service. Consequently, both the position of the global optimum and its value cannot be known *a priori*; it is left up to the engine to draw appropriate relationships between its environment and available parameters in order to determine which solutions are best.

In this example, the goal of the engine is to find the values of θ_0 and θ_1 given the observation ϕ which maximize u through its own learning process. For each iteration, the engine is given a single value of ϕ for which it must choose an appropriate pair θ_0 and θ_1 to maximize u. The values of ϕ are uniformly distributed on the interval $[-\pi, \pi]$ and are uncorrelated. After each iteration, the engine's performance is evaluated by sweeping ϕ through its full span and computing the RMS of the utility from the engine's resulting estimates of θ_0 and θ_1. The results of the simulation are shown in Figure 6.8. Figure 6.8(a) depicts the root mean-square of the utility after each epoch. During the initial stages, the engine's case database is empty as it has no knowledge of the relationship between u, θ_0, θ_1, and ϕ, and the resulting utility is poor. As demonstrated in Figure 6.8(b), however, the number of entries in the database grow quickly with the first fifty iterations and the resulting performance increases sharply. Furthermore the size of the database is kept at a manageable level by periodically pruning entries that are too similar. As described in Section 6.5.4, this serves to keep the memory footprint small as well as reduce the engine's search time.



(a) Utility root mean square



(b) Case database size

Figure 6.8: Convergence of the case database-enabled cognitive engine; root mean-square global utility and case database size vs. adaptation time.

6.7 Experiment 2: Over-the-air Performance

The previous experiment demonstrated the performance of a case database-enabled cognitive engine in a controlled simulation. This experiment extends the previous by enabling the engine to control a wireless link between a pair of radios operating in a laboratory environment.

6.7.1 System Description

Each radio was equipped with a transceiver capable of half-duplex medium access operating in the FRS band. A simple packet network protocol was created to enable bi-directional communication between the nodes such that packet acknowledgments could be sent to validate the data transmitted across the link. If the master node had not received an acknowledgment packet after a timeout period, the packet would be retransmitted. This would occur for all lost and erroneous packets, providing persistent data integrity of the link. Before transmitting each data packet, the master node would first sense the channel for interference by observing the average received signal energy as an indication of channel availability. Only after the energy was consistently below a threshold would the master node transmit a data packet.

Both nodes assembled packets according to the `flexframe` structure described in Section 3.5. Each frame includes detailed information about its underlying structure including payload length, multi-level forward error-correction, modulation scheme, and modulation depth. The advantage to this structures is that the receiving node can automatically reconfigure its physical layer description to each individual packet, allowing the transmitting node flexibility in choosing the best configuration for a given channel state. The packet transmission protocol can be seen in Figure 6.9 in which packet `123` was never received by the slave node and needs to be re-transmitted. Clearly the throughput is limited by the latency of the radio's ability to respond to incoming packets as well as the number of packets that need to be re-transmitted across the channel.

The domain of the cognitive engine is shown in Table 6.2 which lists the engine's five parameters, one observable, and three performance metrics/utilities. Specifically the engine, residing in the master node, had the objectives of maximizing link throughput while reducing transmit power and computational complexity at the slave node. The occupied bandwidth of each link was 100 kHz. Each data packet contained a header describing to the slave node the parameter set, including the transmit power used. Upon reception of each data packet the slave node would compute its average computational load r_p (see Section 5.4.2) and estimate the channel path loss from the master node's transmit power and the received packet's signal strength. For each packet that was properly decoded the slave node would send a short acknowledgment frame back to the master node (see Figure 6.9) which would report both the path loss and its computational complexity for decoding the data payload. A varying path loss was emulated by modifying the transmit gain of the master node (independent of the *transmit gain* parameter θ_4) to exhibit a sinusoidally periodic response in time with a range of 18 dB.

The master node would gather performance statistics and update its case database every 500 ms. Based on the channel conditions and the performance of the link the engine would take action and adjust its parameters accordingly. Each adaptation consisted of modifying the master node's

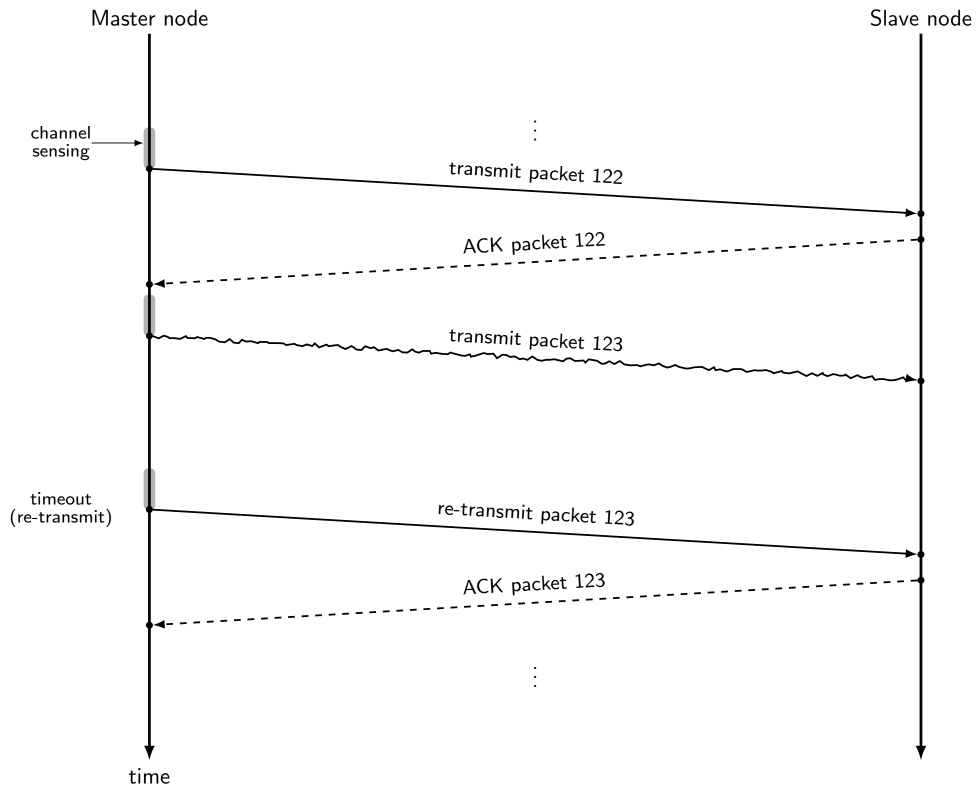


Figure 6.9: Cognitive engine over-the-air example protocol flowchart. Spectrum sensing by the master node is highlighted before each packet transmission. Acknowledgment (ACK) frames are sent after the proper reception of each data packet.

Table 6.2: Cognitive Engine Over-the-Air Example Domain

Item	Description
	Parameters
θ_0, Modulation Scheme	Linear modulation scheme used in the payload of each packet: BPSK, QPSK, {8,16}-PSK, {16,32,64,128,256}-QAM
θ_1, θ_2, FEC (inner/outer)	Forward error-correction codes: uncoded, Hamming(7,4), Hamming(12,8) block codes, convolutional with $r1/2$ (K=7,9), and punctured rates from 2/3 through 7/8, convolutional with $r1/3$ (K=9), and a $r = 0.875$ Reed-Solomon (255,223) block code
θ_3, Payload length	Number of bytes of the raw payload before applying forward error correction and modulating, ranging from 1 to 1024
θ_4, Transmit gain	Transmit power gain of the master node, ranging from -25 to 0 dB
	Observables
ϕ_0, Path loss	Average signal strength degradation due to path loss between master and slave nodes
	Utilities
β_0, Throughput	Maximize link throughput, achieving an average data rate of 130 kbps ($\alpha_0 = 0.5$)
β_1, Transmit power	Minimize transmit power of the master node ($\alpha_1 = 0.3$)
β_2, Complexity	Minimize computational complexity of the receiver node ($\alpha_2 = 0.2$)

modulation and dual-level forward error-correction schemes, transmit power gain, and payload length. It is important to highlight that like any real system this link had no perfect control channel; all feedback in the link (e.g. slave node packet acknowledgments) was subject to error and consequently required the engine to compensate accordingly. Furthermore, all of the data measured in this experiment was completely empirical and not derived from models programmed within the system. In particular the data throughput of the link—in terms of bits per second—incorporates erroneous packets, channel sensing times, and latency between the nodes.

6.7.2 Numerical Results

The results of the experiment can be found in Figures 6.10 and 6.11. Figure 6.10 shows the measured path loss, average global (collapsed) utility, actual throughput for the link, transmit power gain, and receiver computational complexity as a percentage of the CPU under a full load. The throughput was calculated by simply counting the number of bits for which the master node received an acknowledgment divided divided by the time of observation. Figure 6.11 re-plots the empirical path loss measurement for convenience and shows the average computational efficiency (measured in valid bits per 1,000 clock cycles) of the receiver. The following observations can be made from the figures:

1. The utility within the first 100 iterations of the engine is initially very poor, evidenced by the resulting low throughput, high transmit power, and high receiver complexity. This can be attributed to the fact that the engine's case database is initially empty and that the preliminary solutions are inferior and little more than guesses;

2. The engine's average performance improves with time, achieving a maximum average utility of about 0.98 when the path loss is low. This implies that the engine gains experience with each iteration, applying it efficiently to each scenario. Not until the 600^{th} iteration is the utility reliably high which suggests that the engine initially spends a considerable amount of effort exploring alternative solutions rather than benefiting from reliable experience;

3. In general the utility has an inverse relationship to path loss and the metrics are highly dependent upon the channel conditions. When the path loss is high the resulting throughput is low as the receiver's SNR consequently degrades and more packets are received in error. The engine's initial adaptation is to increase transmit power which degrades the metric β_1. Ultimately the engine increases the reliability of the transmission scheme by reducing the rate of the link and dropping back to a more conservative modulation depth and forward error-correction coding at the expense of increasing the receiver's complexity. When the path loss eventually decreases the engine reverts to a more spectrally and computationally efficient scheme and the link performance is improved;

4. After about 600 iterations the global utility never drops below about 0.65 demonstrating that even in poor channel conditions the engine finds a solution suitable to the problem. The performance of the engine becomes more consistent with time as its case database is populated with improved adaptations;

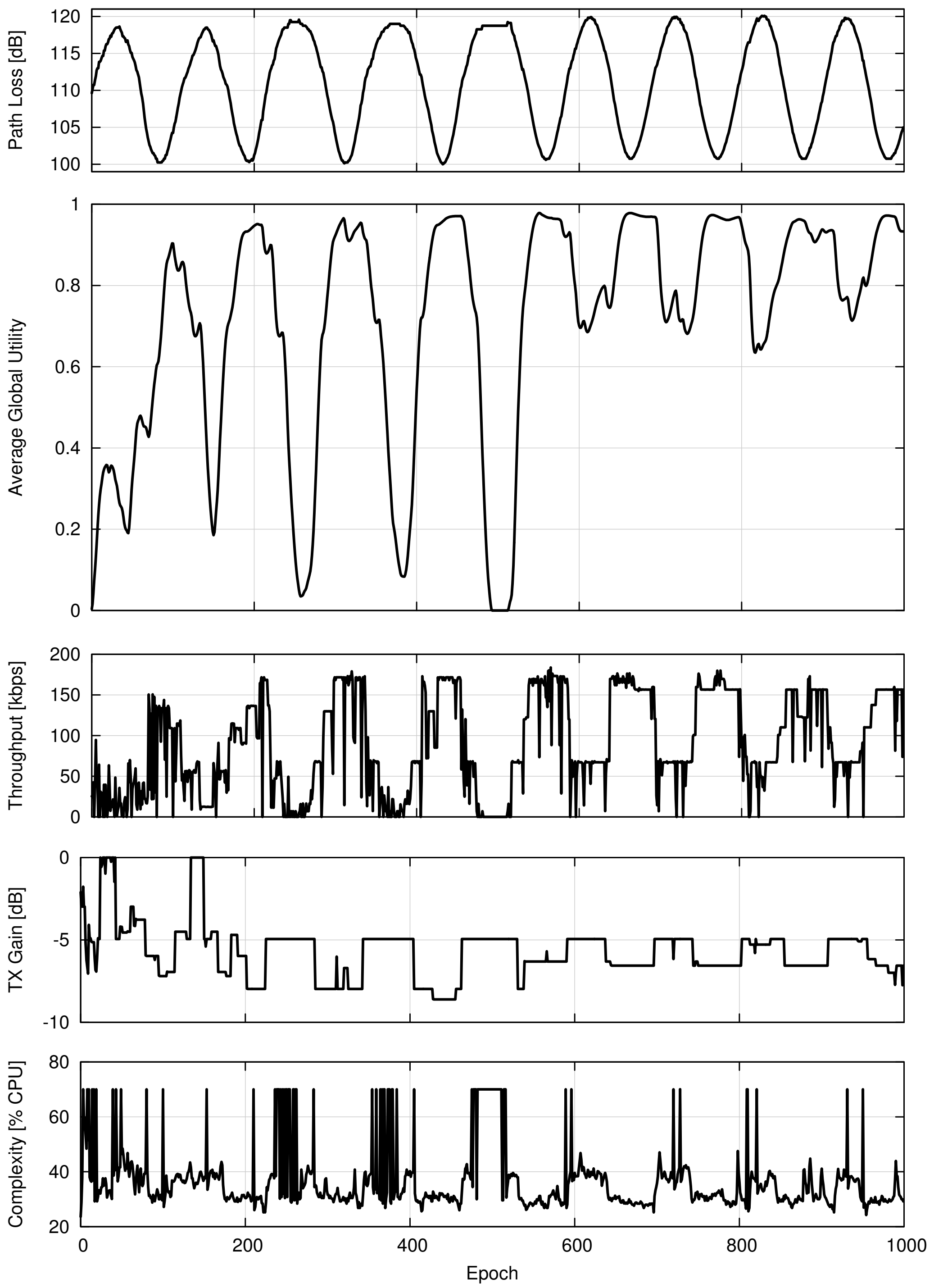


Figure 6.10: Cognitive Engine performance, over-the-air example. All values are measured empirically.

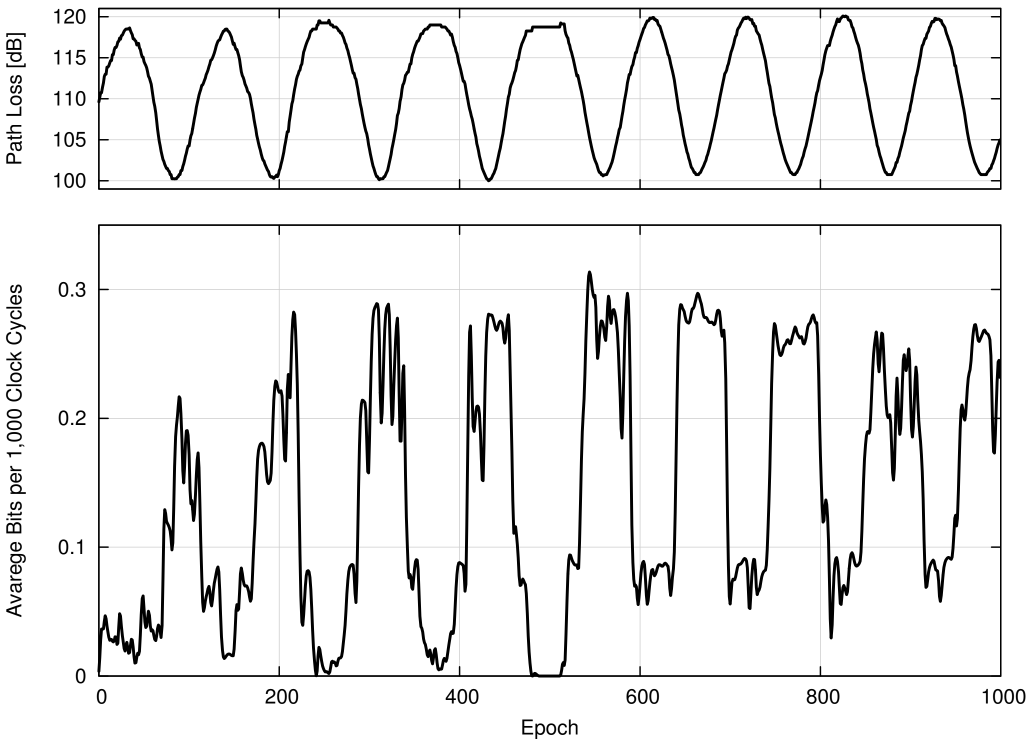


Figure 6.11: Slave node average computational efficiency $1/\mathcal{K}$ (bits per 1,000 clock cycles) over time.

5. The computational efficiency of the slave node shown in Figure 6.11 also demonstrates a strong relationship to channel quality. Specifically the slave node needs to work harder when the path loss is high (channel quality is poor) which is particularly visible within the first 200 iterations of the engine. When the channel improves (path loss is low) the engine drops back to a less computationally complex transmission scheme allowing for an improved resource efficiency at the slave node. There is roughly a factor of three difference in computational efficiency of the receiver between maximum and minimum path loss extrema, corroborating the results of Sections 5.5 and 5.6.

6.8 Conclusions

This chapter has introduced a generalized theory for cognitive radio which describes the relationship between radio parameters, goals, and environmental observable subspaces. In addition, a novel utility function was proposed which provides metric abstraction and allows for generalized use in cognitive engines. The utility function is monotonically increasing and parametrized to describe the relationship between any number of metrics. This chapter additionally contributed theory behind case database reasoners for cognitive engine design including algorithms for case selection and merging in abstract applications. A case database-enabled cognitive engine was devised, demonstrating blind learning with fast adaptation and resource management. The performance of the engine was validated by controlling the parameters of wireless link in a laboratory multi-path environment to maximize throughput while simultaneously limiting transmit power and receiver complexity. The engine demonstrated real-time learning and was able to adapt to varying channel conditions while retaining a relatively high global utility. In particular the computational complexity of the receiving node was kept to a minimum by reserving spectrally efficient yet computationally intensive algorithms (e.g. convolutional decoding) to situations where no other options were available. As a result the engine demonstrated a minimal quality of service penalty for a significant increase in resource efficiency.

Chapter 7

Conclusions

The demand for global wireless internet connectivity has caused an astounding expansion in technological advances, motivating the research community to explore software-defined radios as a mechanism for swift reconfiguration and waveform adaptation. Furthermore, the desire for increased broadband coverage has motivated the exploration of dynamic spectrum access networks, typically through cognitive radio technologies. The accompanying flexibility has come at the cost of increased power demands and a reduction of energy efficiency on the targeted platforms. This work has demonstrated the benefits of resource management in software-defined radios, specifically those operating in dynamic spectrum environments. Spectrum access is achieved by devising new synchronization techniques for spectrally efficient OFDM/OQAM multi-carrier signals capable of recovering carrier frequency and timing information in the presence of strong interference. The developed system has demonstrated significant performance improvements by allowing the radio to monitor its own dynamic resource consumption and adjust its operational modes accordingly.

7.1 Original Contributions

The design requirements to enable wireless access to dynamic spectrum environments, both for single- and multi-carrier operations were introduced in Chapter 2. In particular the benefits of using OFDM/OQAM as a multi-carrier technology over traditional OFDM were highlighted. A novel approximate square-root Nyquist filter design was discovered to have significant improvements over the raised-cosine filter in both stop-band attenuation and inter-symbol interference performance. The filter has similar properties to best known hM-3 design without relying on cumbersome iterations over the Parks-McClellan algorithm, providing a simpler implementation capable of being designed at run time. The resulting r-Kaiser filter was used to demonstrate the superior spectral compactness of OFDM/OQAM over traditional OFDM while using only a fraction of the subcarriers.

Several novel techniques for synchronizing single-carrier communications signals were introduced in Chapter 3. These techniques exploit maximum likelihood-based timing and carrier frequency offset estimators, and were shown to be both computationally efficient while achieving the minimum variance of any linear unbiased estimator. A joint timing and carrier recovery architecture was introduced based on polyphase filterbanks. Furthermore, an innovative framing structure was proposed to enable narrowband dynamic spectrum access through fast synchronization of short packet bursts. The frame exploits cyclostationary properties in the preamble to reduce the synchronizer's acquisition time and includes a short internal descriptor allowing the receiver to automatically reconfigure to the frame's dynamic properties (modulation and forward error-correction schemes, payload length, etc.). Its performance was characterized through baseband simulation and validated with over-the-air experiments conducted in a wireless indoor environment demonstrating signal detection and decoding in very low SNR regions.

The results shown for single-carrier signals motivates the investigation of synchronizing multi-carrier systems in Chapter 4, emphasizing the recovery of OFDM/OQAM signals using polyphase analysis filterbanks. In particular, the compact spectral advantages of OFDM/OQAM gives rise to temporally overlapping symbols, complicating the necessary estimation of reference parameters. As with the single-carrier chapter, a novel preamble signature was proposed to enable both time- and frequency-domain estimators for sample timing and carrier frequency offsets. Furthermore, a pair of novel closed-form estimators were shown to exhibit comparable performance to known estimators in AWGN and fading channels while providing significant performance improvements in interference-laden channels. The results confirm the applicability of using OFDM/OQAM in dynamic spectrum environments where legacy signals can operate in the center of the desired band while neither exhibiting interference to nor experiencing interference from the secondary user. Additionally, OFDM/OQAM was demonstrated to perfectly reconstruct signals overlaying its band while simultaneously decoding data subcarriers, all without the use of an additional receiver. Consequently, the proposed system promotes true spectrum coexistence in dynamic spectrum environments.

An architecture was proposed in Chapter 5 to enable online resource management in software-defined radios by permitting the platform to monitor its own resources and adapt accordingly. Additionally, an online processor usage monitor was proposed capable of accurately estimating the CPU load without a significant impact on the processor itself. The framework was tested by simulating a system capable of adapting modulation and forward error-correction schemes in slowly-fading fading channels, and validated through over-the-air experimentation in a wireless multi-path environment. As a result, a $2.5\times$ reduction in computational requirements was observed without significantly affecting the service quality. The results suggest that even greater performance improvements can be achieved by granting the system full access to the receiver, permitting all aspects of the radio to be adapted based on the dynamics of the wireless channel.

A unifying definition and framework for cognitive radios was presented in Chapter 6, generically describing the relationship between a radio's goals, adjustable parameters, and observed environment. Additionally proposed was a novel utility function benefiting from a monotonic shape and capable of being parametrized to describe the relationship between any number of metrics and the respective goals of the radio, all while abstracting the radio's underlying performance metrics. Furthermore, a generic cognitive engine was proposed based on a case database using specific knowledge of previously-experienced solutions rather than relying on general knowledge from within a problem domain. Consequently, the engine itself is generic to the specific radio application by representing cases as abstract feature vectors and normalizing distance metrics between new and past experiences. Its performance was demonstrated to quickly and blindly adapt to unique scenarios without the use of an initial knowledge base. Furthermore, the size of the database is automatically managed by periodically merging similar cases, retaining only those which have proven to work best in relevant situations.

7.2 Closing Remarks

The interactions between a radio's operational modes and its performance in a dynamic wireless environment are complex, and while waveform adaptation has suggested performance improvements the most significant challenge of this research has been to demonstrate this response in a practical sense. Synchronization of radio signals has a significant impact on the quality of service in dynamic spectrum environments and is the limiting factor for the portability of broadband wireless handsets. The benefits of dynamic online resource management can be easily observed in software-defined radios when considering the limited processing bandwidth of an embedded platform. However, these benefits can still be extended to hardware platforms when considering that a significant amount of energy is expended performing tasks in the physical layer that might not be necessary given the conditions of the wireless environment. In this regard, the platform's computational complexity demands are less of an issue as the signal processing is accomplished with dedicated hardware, and while individual tasks are not shared by common silicon as they are in SDR, one would still expect to see similar performance improvements. However, as wireless systems are constantly evolving, the eventual migration to platforms defining their physical characteristics in software seems imminent. Consequently, machine learning has been promised to improve network coverage by opening unlicensed spectrum to secondary users; however, the application of artificial intelligence into radios is still an unproven technology in everyday life. As such, the future of software-defined and cognitive radios is still undetermined as each has yet to make its mark in technological history.

Appendix A

List of Abbreviations

ACK acknowledgment (packet)

APSK amplitude/phase-shift keying

AWGN additive white Gauss noise

BER bit error rate

CE cognitive engine

CPU central processing unit

CR cognitive radio

CRC cyclic redundancy check

CRVB Cramér-Rao variance bound

BPSK binary phase-shift keying

dB decibels

DFT discrete Fourier transform

DSA dynamic spectrum access

DSP digital signal processor

dMF dervative matched filter

FEC forward error correction

FFT fast (discrete) Fourier transform

FPGA field-programmable gate array

hM-3 harris-Moerder-3 filter [29]

ISI inter-symbol interference

ISM industrial, scientific, and medical (frequency band)

LS least squares

MCRB modified Cramér-Rao bound

MF matched filter

MLS modified least squares

NCO numerically-controlled oscillator

OFDM orthogonal frequency-division multiplexing

OFDM/OQAM OFDM using offset quadrature amplitude modulation

PLL phase-locked loop

P/N pseudo-noise (sequence)

PSK phase-shift keying

QAM quadrature amplitude modulation (quaternary amplitude-shift keying)

QPSK quaternary phase-shift keying

PER packet error rate

RF radio frequency

RMS root mean-square (error)

r-Kaiser approximate square-root Nyquist Kaiser filter

SDR software-defined radio

SIR signal-to-interference ratio

SNR signal-to-noise ratio

USRP univeral software radio peripheral [21]

Appendix B

Lists of Symbols

Table B.1: Symbols for Chapter 2: Physical Layer Mechanisms for Dynamic Spectrum Access

Symbol	Description
k	matched-filter over-sampling rate [samples/symbol]
m	matched-filter delay [symbols]
β	matched-filter excess bandwidth factor
N	non-recursive filter order (number of coefficients)for (root) Nyquist filters $N = 2km+1$
F_s	sampling frequency [samples/s]
ω_p	filter pass-band cutoff frequency [radians/s]
ω_s	filter stop-band cutoff frequency [radians/s]
$\Delta\omega$	filter bandwidth [radians/s], $\Delta\omega = \omega_p - \omega_s$
ω_0	critical sampling frequency [radians/s],$\omega_0 = \pi F_s/k$
δ_p	filter pass-band ripple
δ_s	filter stop-band ripple
A_s	filter stop-band attenuation [dB], $A_s = -20\log_{10}(\delta_s)$
α	Kaiser-Bessel window shape parameter
$I_\nu(z)$	modified Bessel function of the first kind, order ν
$H(z)$	Nyquist filter
$G(z)$	square-root Nyquist filter
γ	r-Kaiser bandwidth correction factor
ρ	r-Kaiser bandwidth correction factor (normalized to β)

Table B.2: Symbols for Chapter 3: Synchronization of Narrowband Channels

Symbol	Description
a	transmitted symbol
$g(t)$	square-root Nyquist transmit and receive filters
$s(t)$	transmitted signal
$r(t)$	received signal
ν	carrier frequency offset [radians/s]
θ	carrier phase offset [radians]
τ	timing offset [samples]
γ	channel gain, relating to signal-to-noise ratio
h_c	multi-path channel impulse response
$n(t)$	noise
p	symbol index
k	sample time index
T	symbol period
T_s	sampling period
T_0	observation time [s], continuous
L_0	observation time [samples], discrete
μ	irrational fractional sampling interval
m	polyphase filterbank index
M	number of polyphase filters in the bank
y	interpolating (matched) filter output sample
η	spectral efficiency [b/s/Hz]
$\Re\{\cdot\}$	real component
$\Im\{\cdot\}$	imaginary component

Table B.3: Symbols for Chapter 4: Synchronization of Multichannel Communications Systems

Symbol	Description
$G(\omega)$	square-root Nyquist filter response
$s(t)$	transmitted signal
$r(t)$	received signal
$a^{\mathcal{R}}$	OFDM/OQAM symbol (real component)
$a^{\mathcal{I}}$	OFDM/OQAM symbol (imaginary component)
p	symbol index
m	subcarrier index (also matched filter delay)
M	number of subcarriers
M_u	number of all enabled subcarriers
$g(t)$	square-root Nyquist transmit and receive filters
ν	carrier frequency offset [radians/s]
θ	carrier phase offset [radians]
τ	timing offset [samples]
γ	channel gain, relating to signal-to-noise ratio
h_c	multi-path channel impulse response
$n(t)$	noise
$\psi(t)$	interference
T_0	observation time [s], continuous
L_0	observation time [samples], discrete
k	sample time index
T	symbol period
T_s	sampling period
y	downsampler symbol output
G_m	complex channel gain on m^{th} subcarrier
R	number of repetitions in OFDM/OQAM preamble
P	time delay between repetitions [samples]
$\mathcal{A}$	set of all enabled subcarriers
$\mathcal{P}$	set of all pilot subcarriers
$w(\ell)$	equalizer window
$\bar{w}_m$	equalizer window gain normalization

Table B.4: Symbols for Chapter 5: Resource Management

Symbol	Description
C	channel capacity [b/s]
W	bandwidth [Hz]
$\bar{P}$	average transmit power [W]
L_p	path loss
N_0	noise power spectral density [W/Hz]
P_c	receiver power consumption [W]
η_e	receiver processing energy efficiency [b/J]
η	spectral efficiency [b/s/Hz]
$\mathcal{K}$	computational complexity [cycles/bit]
r_p	processor load rate
T_a	processor time spent running algorithm [s]
T_e	total processor time spent in execution [s]
P_b	bit error rate
P_p	packet error rate
γ	instantaneous signal-to-noise ratio (SNR)
Ω	average signal-to-noise ratio (SNR)
E_b/N_0	bit energy to noise ratio
$f_\gamma(\gamma)$	fading distribution for instantaneous received signal power
F_p	processor master clock frequency [cycles/s]
ζ	normalized spectral efficiency, compensating for packet loss [b/s/Hz]
$\mathcal{S}_\eta$	set of spectrally efficient modulation/error-correction scheme pairs
$\mathcal{S}_\mathcal{K}$	set of computationally efficient modulation/error-correction scheme pairs

Table B.5: Symbols for Chapter 6: Cognitive Engine Design

Symbol	Description
$\boldsymbol{\beta}$	set of radio performance metrics
$\boldsymbol{\theta}$	set of adjustable radio parameters
$\boldsymbol{\phi}$	set of environmental observable operating conditions
v	collapsed utility (production function output)
$q(\beta)$	utility function
$\dot{\beta}$	target radio performance metric
η	utility function threshold parameter
σ	utility function spread parameter
$g(\boldsymbol{u})$	production function
α	production function weighting coefficients
s_k	case representation vector
D	case selection distance metric
ζ	case selection weighting parameter

Appendix C

Approximations for the I_0 Kaiser Window for Non-recursive Filter Design

This appendix deals with approximating the modified Bessel function of the first kind for its use in non-recursive (finite impulse response) filter design. A commonly used expansion of the modified Bessel function of the first kind is given by

$$I_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{1}{4}z^2\right)^k}{k!\Gamma(k+\nu+1)} \tag{C.1}$$

which, for $\nu = 0$ collapses to

$$I_0(z) = \sum_{k=0}^{\infty} \left[\frac{(0.5z)^k}{k!}\right]^2 \tag{C.2}$$

The challenge for finite precision machines is to compute the ratio of two large numbers for the Kaiser window. As it happens, the ratio, once computed, is manageable and in $[0, 1]$. Observing that the first term of the series ($k = 0$) yields 1 (C.2) can be rewritten as

$$I_0(z) = 1 + \sum_{k=1}^{\infty} \left[\frac{(0.5z)^k}{k!}\right]^2 \tag{C.3}$$

Additionally, the shape of I_0 grows very quickly. Performing a variable transformation $t = \log(z)$ and plotting it against $I_0(t)$ on a log scale reveals a piece-wise linear dependency, as depicted by Figure C.1 Therefore the modified Bessel function of the first kind (order 0) may be approximated using piecewise polynomials as

$$\log\Big(\log\big(I_0(z)\big)\Big) \approx c_0 + c_1 t + c_2 t^2 + c_3 t^3 \tag{C.4}$$

where $t = \log(z)$ and the coefficients c_0, c_1, c_2, and c_3 are given in Table C.1. This is a particularly useful approximation for the Kaiser window in fixed-point math where $w[n]$ is computed as the ratio of two large numbers. The absolute error of (C.4) is depicted in Figure C.1, demonstrating the efficacy of this approximation.

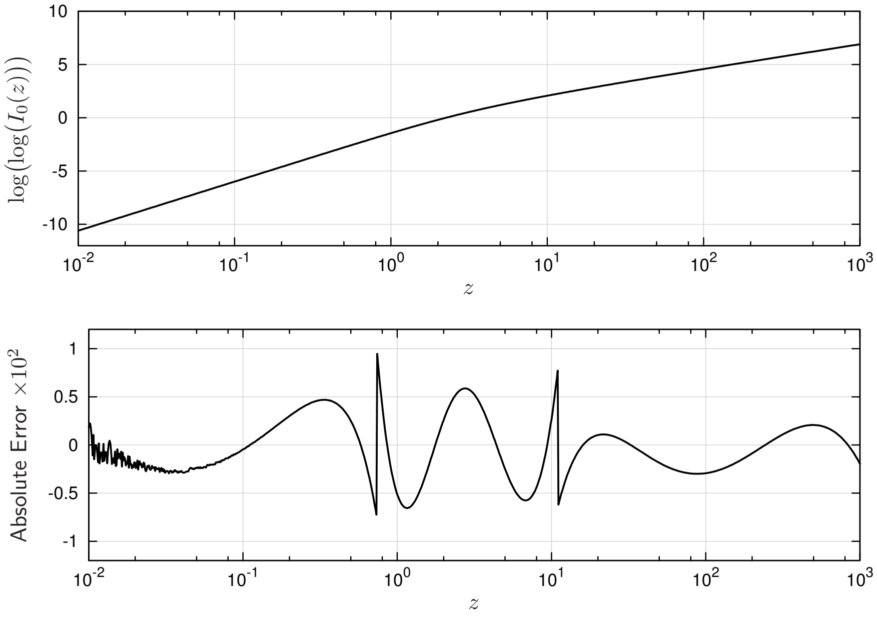


Figure C.1: Piecewise polynomial approximation to the modified Bessel function of the first kind.

Table C.1: Piece-wise polynomial coefficients to approximation for $\log\big(\log\big(I_0(z)\big)\big)$ given by (C.4)

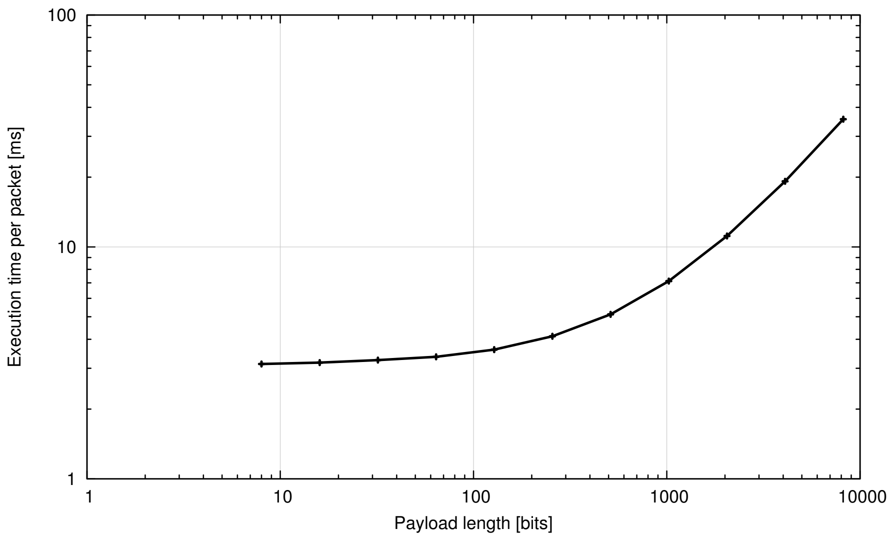
Region	c_0	c_1	c_2	c_3
$\log(z) \leq -0.3$	-1.41802979	1.97680664	-0.00466919	-0.00024033
$-0.3 < \log(z) < 2.4$	-1.43916631	1.91274643	-0.17395020	0.00218964
else	-0.85644531	1.41308594	-0.06835938	0.00382996

Appendix D

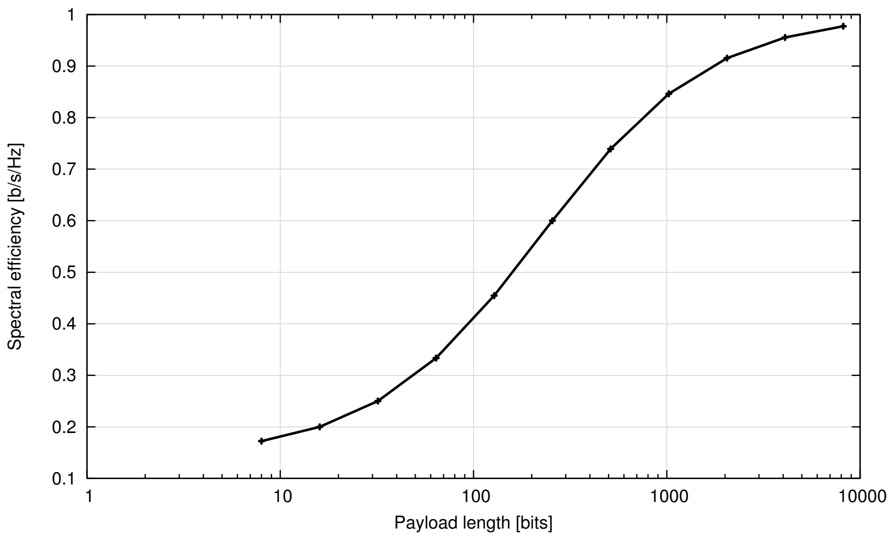
`flexframe` Synchronizer Performance

The flexible framing structure described in Section 3.5 promises fast synchronization of This section observe the performance of the synchronizer in terms of both average execution time and spectral efficiency as the length of the payload is increased. The true spectral efficiency is calculated in (3.26) and is clearly impacted by the framing overhead, the length of the data payload, and the rate of the underlying modulation and forward error-correction coding schemes.

Figure D.1 depicts the performance of the synchronizer as the payload length is adjusted. As the length of the payload is increased so does the receiver's performance as the relative effects of the framing overhead diminish. In particular, the execution time becomes linearly dependent with the payload as depicted in Figure D.1(a) after approximately 1000 bits. This suggests that an initial overhead in framing acquisition accounts for a considerable portion of processing at the receiver. When the payload is sufficiently long, this contribution is negligible and the majority of computation is spent in synchronizing the payload, thus its linear response. Furthermore the spectral efficiency exhibits a similar response and asymptotically reaches its maximum as the payload increases. This is made evident both by (3.26) and Figure D.1(b).



(a) Average execution time



(b) Spectral efficiency

Figure D.1: Impact of packet length on `flexframe` synchronizer performance in terms of average execution time and normalized spectral efficiency using an uncoded payload with BPSK modulation.

Appendix E

Introduction to *liquid*

liquid is a free and open-source digital signal processing (DSP) library designed specifically for software-defined radios on embedded platforms [35]. Its aim is to provide a lightweight DSP library that does not rely on a myriad of external dependencies or proprietary and otherwise cumbersome frameworks. Included in the package are modules for automatic gain control, audio source encoding, internal sample buffering, vector dot product computations, adaptive equalization, basic forward error correction and cyclic redundancy checks, simple fast Fourier transforms, filtering, math and matrix operations, modulation and demodulation, non-linear optimization, random number generation, and much more. All signal processing elements are designed to be flexible, scalable, and dynamic, including filters, filter design, oscillators, modems, synchronizers, and complex mathematical operations. *liquid* was created out of necessity to perform complex digital signal processing algorithms on embedded devices without relying on proprietary and otherwise cumbersome frameworks. Its intent is to be portable to embedded platforms while using flexible configurations with a minimal dependence on external libraries. All the software in this dissertation used *liquid* in analysis and is free for anyone to download and try.

E.1 A Brief Tutorial

Most of *liquid*'s signal processing elements are C structures which retain the object's parameters, state, and other useful information. The naming convention is `basename_xxxt_method` where `basename` is the base object name (e.g. `interp`), `xxxt` is the type definition, and `method` is the object method. The type definition describes respective output, internal, and input type. Types are usually `f` to denote standard 32-bit *floating point* precision values and can either be represented as `r` (real) or `c` (complex). For example, a `dotprod` (vector dot product) object with complex input and output types but real internal coefficients operating on 32-bit floating-point precision values is `dotprod_crcf`.

Most objects have at least four standard methods: `create()`, `destroy()`, `print()`, and `execute()`. Certain objects also implement a `recreate()` method which operates similar to that of `realloc()` in C and are used to restructure or reconfigure an object without completely destroying it and creating it again. Typically, the user will create the signal processing object independent of the external (user-defined) data array. The object will manage its own memory until its `destroy()` method is invoked. A few points to note:

1. The object is only used to maintain the state of the signal processing algorithm. For example, a finite impulse response filter needs to retain the filter coefficients and a buffer of input samples. Certain algorithms which do not retain information (those which are memory-less) do not use objects. For example, `design_rrc_filter()` calculates the coefficients of a root raised-cosine filter, a processing algorithm which does not need to maintain a state after its completion.

2. While the objects do retain internal memory, they typically operate on external user-defined arrays. As such, it is strictly up to the user to manage his/her own memory. Shared pointers are a great way to cause memory leaks, double-free bugs, and severe headaches.

E.1.1 Learning by example

While this document contains numerous examples listed in the text, they are typically condensed to demonstrate only the interface. The `examples/` subdirectory includes more extensive demonstrations and numerous examples for all the signal processing components. Many of these examples write an output file which can be read by octave [19] to display the results. For a brief description of each of these examples, see `examples/README`.

E.1.2 Why C?

A commonly asked question is "why C and not C++?" The answer is simple: *portability.* The aim is to provide a lightweight DSP library for software-defined radio that does not rely on a myriad of dependencies. While C++ is a fine language for many purposes (and theoretically runs just as fast as C), it is not as portable to embedded platforms as C is. Furthermore, the majority of functions simply perform complex operations on a data sequence and do not require a high-level object-oriented programming interface. The importance in object-oriented programming is the techniques used, not the languages describing it.

While a number of signal processing elements in *liquid* use structures, these are simply to save the internal state of the object. For instance, a `firfilt_crcf` (finite impulse response filter) object is just a structure which contains–among other things–the filter taps (coefficients) and an input buffer. This simplifies the interface to the user; one only needs to "push" elements into the filter's internal buffer and "execute" the dot product when desired. This could also be accomplished with classes, a construct specific to C++ and other high-level object-oriented programming languages; however, for the most part, C++ polymorphic data types and abstract base classes are unnecessary for basic signal processing, and primarily just serve to reduce the code base of a project. Furthermore, while C++ templates can certainly be useful for library development, their benefits are of limited use to signal processing and can be circumvented through the use of pre-processor macros at the gain of targeting more embedded processors. Under the hood, the C++ compiler's pre-processor expands templates and classes before actually compiling the source anyway, so in this sense they are equivalent to the second-order macros used in *liquid*.

The C programming language has a rich history in system programming– specifically targeting embedded applications–and is the basis behind many well-known projects including the Linux kernel and the python programming language. Having said this, high-level frameworks and graphical interfaces are much more suited to be written in C++ and will beat an implementation in C any day, but lie far outside the scope of this project.

E.1.3 Data Types

The majority of signal processing for SDR is performed at complex baseband. Complex numbers are handled in *liquid* by defining data type `liquid_float_complex` which is binary compatible with the standard C math type `float complex` and C++ type `std::complex<float>`. Although *liquid* is written in C, it can be seamlessly compiled and linked with C++ source files. Here is an example:

```
// file:    listings/nco.c++
// build:   g++ -Wall nco.c++ -lliquid -o nco_test

#include <iostream>
#include <math.h>

#include <liquid/liquid.h>

// NOTE: the definition for liquid_float_complex will change
//       depending upon whether the standard C++ <complex>
//       header file is included before or after including
//       <liquid/liquid.h>
#include <complex>

int main() {
    // create nco object and initialize
    nco_crcf n = nco_crcf_create(LIQUID_NCO);
    nco_crcf_set_phase(n,0.3f);
    nco_crcf_set_frequency(n,0.0f);

    // Test liquid complex data type
    liquid_float_complex x;
    nco_crcf_cexpf(n, &x);
    std::cout << "liquid complex:     "
              << x[0] << " + j" << x[1] << std::endl;

    // Test native c++ complex data type
    std::complex<float> y;
    nco_crcf_cexpf(n, reinterpret_cast<liquid_float_complex*>(&y));
    std::cout << "c++ native complex: "
              << y.real() << " + j" << y.imag() << std::endl;

    // destroy nco object
    nco_crcf_destroy(n);

    std::cout << "done." << std::endl;
    return 0;
}
```

E.2 Installation Guide

liquid was designed to be portable, and in doing so requires minimal dependencies to build and run. The required software packages are

1. `gcc`, the GNU compiler collection
2. `libc`, the standard C library
3. `libm`, the standard math library (eventually will be phased out to optional)

The project takes advantage of other libraries if they are installed on the target machine, including `fftw3` (a fast Fourier transform library [31]) and `libfec` (a forward error-correction encoding/decoding library [56]). The build system checks to see if they are installed during the `configure` process and will generate an appropriate `config.h` if they are.

The core development of *liquid* uses `git` distributed version control [41] which is open source and freely available. Building *liquid* is easily accomplished by moving into the unpacked directory, running the bootstrap configuration scrips, and invoking `make`, viz

```
$ cd /path/to/liquid/
$ ./reconf
$ ./configure
$ make
# make install
```

The `install` target copies the global header file `liquid.h` and the shared object file `libliquid.a` to the system directories. You must have root access to run make with the `install` target. Additionally, the `uninstall` target removes these files from the system directory. You may also want to build and run the optional validation program via

```
$ make check
```

and the benchmarking tool

```
$ make bench
```

In order to keep the project relatively organized, the source code is broken up into separate "modules" under the top `src/` directory. Each module consists of a top-level included makefile, a `README`, the library source files, a set of test scripts, and a set of benchmarks.

Bibliography

[1] Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):35–9, March 1994.

[2] Joydeep Acharya, Harish Viswanathan, and Sivarama Venkatesan. Timing Acquisition for Non Contiguous OFDM based Dynamic Spectrum Access. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, 2008.

[3] Joydeep Acharya, Harish Viswanathan, and Sivarama Venkatesan. Feasibility of NC-OFDM Transmission in Dynamic Spectrum Access Environments. In *IEEE Military Communications Conference*, 2009.

[4] P. Amini, R. Kempter, and B. Farhang-Boroujeny. A Comparison of Alternative Filterbank Multicarrier Methods for Cognitive Radio Systems. In *Proceedings of the Software-Defined Radio Forum*, 2006.

[5] Mixed-Signal Front-End (MxFE^{TM}) Processor for Broadband Communications. Analog Devices Product Datasheet, 2002.

[6] Ralph Bergmann, Janet Kolodner, and Enric Plaza. Representation in case-based reasoning. *The Knoweldge Engineering Review*, 0(0):1–4, 2005.

[7] Jan W. M. Bergmans. Effect of Loop Delay on Stability of Discrete-Time PLL. *IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications*, 42(4), April 1995.

[8] Jan W. M. Bergmans. Effect of Loop Delay on Phase Margin of First-Order and Second-Order Control Loops. *IEEE Transactions on Circuits and Systems–II: Express Briefs*, 52(10), October 2005.

[9] Roland E. Best. *Phase-Locked Loops: Design, Simulation, and Applications.* McGraw-Hill, 3 edition, 1997.

[10] Benjamin S. Bloom, editor. *Taxonomy of Educational Objectives, Handbook I: Cognitive Domain*, volume 1. David McKay Company, Inc., New York, 1956.

[11] Helmut Bölcskei. Blind Estimation of Symbol Timing and Carrier Frequency Offset in Wireless OFDM Systems. *IEEE Transactions on Communications*, 49(6):988—999, 2001.

[12] Helmut Bölcskei, Pierre Duhamel, and Rima Hleiss. Design of Pulse Shaping OFDM/OQAM Systems for High Data-Rate Transmission over Wireless Channels. In *IEEE International Conference on Communications*, volume 1, 1999.

[13] Bruno Bougard, Gregory Lenoir, Francky Catthoor, and Wim Dehaene. A New Approach to Dynamically Trade Off Performance and Energy Consumption in Wireless Communication Systems. In *IEEE Workshop on Signal Processing Systems*, 8 2003.

[14] Bruno Bougard, David Novo, Frederick Naessens, Lieven Hollevoet, Thomas Schuster, Miguel Glassee, Antoine Dejonghe, and Liesbet Van der Perre. A Scalable Baseband Platform for Energy-Efficient Reactive Software-Defined-Radio. In *CROWNCOM*, volume 1, pages 1–5, 6 2006.

[15] Bruno Bougard, Sofie Pollin, Antoine Dejonghe, Francky Catthoor, and Wim Dehaene. Cross-layer power management in wireless networks and consequences on system-level architecture. *EURASIP Signal Processing Journal, Special Issue on Advances in Signal Processing*, 86(8):1792–1803, 2006.

[16] Ibrahim Budiarjo, Homayoun Nikookar, and Leo. P. Ligthart. Cognitive Radio Modulation Techniques. *IEEE Signal Processing Magazine*, 25(6):24–34, 11 2008.

[17] R. W. Chang. Synthesis of band-limited orthogonal signals for multi-channel data transmission. *Bell Systems Technical Journal*, 45:1775–1796, 12 1966.

[18] A. Dejonghe, J. Declerck, F. Nassens, M. Glassée, A. Dusa, E. Umans, A. Ng, B. Bougard, G. Lenoir, J. Craninckx, and L. Van der Perre. Low-power SDRs through cross-layer control: concepts at work. In *Mobile and Wireless Communications Summit*, pages 1–6, 2007.

[19] John W. Eaton. Octave Website. http://www.gnu.org/software/octave/, 2011.

[20] Electronics and Telecommunications Research Institude. *http://www.etri.re.kr/eng/*, October 2010.

[21] Matt Ettus. Ettus Research LLC. http://www.ettus.com/, July 2010.

[22] Behrouz Farhang-Boroujeny. Filter Bank Spectrum Sensing for Cognitive Radio. *IEEE Transactions on Signal Processing*, 56(5):1801–1811, May 2008.

[23] Albrecht J. Fehske, Joseph D. Gaeddert, and Jeffrey H. Reed. A new approach to signal classification using spectral correlation and neural networks. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, pages 144–50. DySPAN, November 2005.

[24] N. J. Fliege. *Multirate Digital Signal Processing.* John Wiley & Sons, Ltd, 1994.

[25] L. E. Franks. Carrier and bit synchronization in data communication–a tutorial review. *IEEE Transactions on Communications*, COM-28(8), August 1980.

[26] fred harris and Erik Kieldsen. A Novel Interpolated Tree Orthogonal Multiplexing (ITOM) Scheme with Compact Time-Frequency Localization: an Introduction and Comparison to Wavelet Filter Banks and Polyphase Filter Banks. In *Proceedings of the Software-Defined Radio Forum*, 2006.

[27] fredric j. harris. On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform. *Proceedings of the IEEE*, 66(1):51–83, January 1978.

[28] fredric j. harris. *Multirate Signal Processing for Communication Systems.* Prentice Hall, 2004.

[29] fredric j. harris, Chris Dick, Sridhar Seshagiri, and Karl Moerder. An Improved Square-Root Nyquist Shaping Filter. In *Proceedings of the Software-Defined Radio Forum*, 2005.

[30] fredric j. harris and Michael Rice. Multirate Digital Filters for Symbol Timing Synchronization in Software Defined Radios. *IEEE Journal on Selected Areas of Communications*, 19(12):2346–2357, December 2001.

[31] Matteo Frigo and Steven G. Johnson. The Design and Implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[32] Tilde Fusco, Luciano Izzo, Angelo Petrella, and Mario Tanda. Blind Symbol Timing Estimation for OFDM/OQAM systems. In *International Conference on Communications and Signal Processing*, 2008.

[33] Tilde Fusco, Angelo Petrella, and Mario Tanda. Data-Aided Symbol Timing and CFO Synchronization for Filter Bank Multicarrier Systems. *IEEE Transactions on Wireless Communications*, 8(5), May 2009.

[34] Tilde Fusco and Mario Tanda. Blind Frequency-Offset Estimation for OFDM/OQAM Systems. *IEEE Transactions on Signal Processing*, 55(8):1828—1838, May 2007.

[35] Joseph D. Gaeddert. liquid-dsp website. https://github.com/jgaeddert/, 2011.

[36] Joseph D. Gaeddert, Carl B. Dietrich, Philip J. Balister, Carlos R. Aguayo Gonzalez, Drew Cormier, S.M. Shajedul Hasan, Kyehun Lee, Shereef Sayed, Haris I. Volos, and Chen Zhang. Spectrum Access for First Responders; Ciren: Cognitively Intrepid Radio Emergency Network. Technical report, Wireless @ Virginia Tech, September 2007.

[37] Joseph D. Gaeddert, Kyouwoong Kim, Rekha Menon, Lizdabel Morales, Youping Zhao, Kyung K. Bae, and Jeffrey H. Reed. Development of a Cognitive Engine and Analysis of WRAN Cognitive Radio Algorithms. Technical report, Wireless @ Virginia Tech, December 2006.

[38] Joseph D. Gaeddert and Jeffrey H. Reed. Leveraging Flexibility in Reconfigurable Baseband Processors for Resource Management in Software-Defined and Cognitive Radios. In *Proceedings of the Software-Defined Radio Forum*, December 2009.

[39] Joseph D. Gaeddert, Haris I. Volos, Drew Cormier, and Jeffrey H. Reed. Multi-rate Synchronization of Digital Receivers in Software-Defined Radios. In *Proceedings of the Software-Defined Radio Forum*, 2007.

[40] Floyd M. Gardner. Interpolation in Digital Modems–Part I: Fundamentals. *IEEE Transactions on Communications*, 41(3):501–507, March 1993.

[41] official git website. http://git-scm.com/, 2011.

[42] Alessandro De Gloria, Daniele Grosso, Mauro Olivieri, and Giuseppe Restani. A Novel Stability Analysis of a PLL for Timing Recovery in Hard Disk Drives. *IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications*, 46(8):1026–1031, August 1999.

[43] Madan M. Gupta. *Static and Dynamic Neural Networks: from Fundamentals to Advanced Theory.* Wiley, New York, 2003.

[44] S.M. Hasan, Philip Balister, Joseph D. Gaeddert, Carlos Aguayo Gonzalez, Kye Hun Lee, Haris I. Volos, Shereef Sayed, Chen Zhang, Xuetao Chen, Timothy R. Newman, Carl B. Dietrich, and Andrew Cormier. Spectrum Access for First Responders using Cognitively Intrepid Radio Emergency Network. In *Proceedings of the Software-Defined Radio Forum*, December 2009.

[45] Simon S. Haykin. *Adaptive Filter Theory.* Prentice Hall, Upper Saddle River, N.J., 4 edition, 2002.

[46] An He, Kyung K. Bae, Timothy R. Newman, Joseph D. Gaeddert, Kyouwoong Kim, Rekha Menon, Lizdabel Morales-Tirado, James “Jody” Neel, Youping Zhao, and Jeffrey H. Reed. A Survey of Artificial Intelligence for Cognitive Radios. *IEEE Transactions on Vehicular Technology*, 59(4):1578–92, May 2010.

[47] An He, Joseph D. Gaeddert, Kyung K. Bae, Timothy R. Newman, Jeffrey H. Reed, Lizdabel Morales, and Chang-Hyun Park. Development of a case-based reasoning cognitive engine for IEEE 802.22 WRAN applications. *ACM Mobile Computing and Communications Review*, 13(2):37–48, 2009.

[48] O. Herrmann, Lawrence R. Rabiner, and D. S. K. Chan. Practical Design Rules for Optimum Finite Impulse Response Lowpass Digital Filters. *Bell Systems Technical Journal*, 52:769–799, July-August 1973.

[49] Botaro Hirosaki. An Orthgonally Multiplexed QAM System Using the Discrete Fourier Transform. *IEEE Transactions on Communications*, COM-29(7):982–989, 7 1981.

[50] Yu Hu, Qing Li, and C.-C. Jay Kuo. Run-time Power Consumption Modeling for Embedded Multimedia Systems. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005.

[51] Functional Requirements for the 802.22 WRAN. *http://www.ieee802.org/22/*, September 2005.

[52] IEEE Computer Society. IEEE Standard 802.15.1. 10.1109/IEEESTD.2005.96290, June 2005. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements. Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).

[53] IEEE Computer Society. IEEE Standard 802.11. 10.1109/IEEESTD.2007.373646, June 2007. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[54] James F. Kaiser. Nonrecursive Digital Filter Design Using the I_0-sinh Window Function. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2O–23, April 1974.

[55] James F. Kaiser and Ronald W. Schafer. On the Use of the I_0-sinh Window for Spectrum Analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-28(1):105–7, February 1980.

[56] Phil Karn. http://www.ka9q.net/code/fec/, August 2007.

[57] Amin Khajeh, Shih-Yang Chen, Ahmen M. Eltawil, and Fadi J. Kurdahi. Power Management for Cognitive Radio Platforms. In *IEEE Global Telecommunications Conference*, pages 4066–4070, 11 2007.

[58] Namjim Kim, Nasser Kehtarnavaz, Mark B. Yeary, and Steve Thornton. DSP-Based Hierarchical Neural Network Modulation Signal Classification. *IEEE Transactions on Neural Networks*, 14:1065–71, September 2003.

[59] Hisashi Kobayashi. Simultaneous adaptive estimation and decision algorithm for carrier modulated data transmission systems. *IEEE Transactions on Communication Technology*, COM-19(3):268–80, June 1971.

[60] Rainer Kokozinski, Dieter Greifendorf, Joerg Stammen, and Peter Jung. The Evolution of Hardware Platforms for Mobile 'Software Defined Radio' Terminals. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 5, pages 2389–2393, 9 2002.

[61] Janet L. Kolodner. Improving Human Decision Making through Case-Based Decision Aiding. *AI Magazine*, 12(2), 1991.

[62] David B. Leake, editor. *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* MIT Press, Cambridge, MA, 1996.

[63] Chia-Hung Lien, Ying-Wen Bai, and Ming-Bo Lin. Estimation by Software for the Power Consumption of Streaming-Media Servers. *IEEE Transactions of Instrumentation and Measurements*, 56(5):1859–1870, October 2007.

[64] Lyrtech SFF SDR Development Platform. http://www.lyrtech.com/DSP-development/dsp_fpga/sffsdrdevelopmentplatform.php.

[65] Damián Marelli and Minyue Fu. A subband approach to channel estimation and equalization for dmt and ofdm systems. *IEEE Transactions on Communications*, 53(11):1850–8, November 2005.

[66] James H. McClellan, Thomas W. Parks, and Lawrence R. Rabiner. A Computer Program for Designing Optimum FIR Linear Phase Digital Filters. *IEEE Transactions on Audio and Electroacoustics*, AU-21(6), December 1973.

[67] Umberto Mengali and Aldo N. D'Andrea. *Synchronization Techniques for Digital Receivers.* Applications of Communications Theory. Springer, New York, 1 edition, 1997.

[68] M. H. Meyers and L. E. Franks. Joint carrier phase and symbol timing recovery for PAM systems. *IEEE Transactions on Communications*, COM-28(8):1121–9, August 1980.

[69] Heinrich Meyr, Marc Moeneclaey, and Stefan A. Fechtel. *Digital Communications Receivers: Synchronization, Channel Estimation, and Signal Processing*, volume 2 of *Wiley Series in Telecommunications and Signal Processing.* John Wiley & Sons, New York, 1998.

[70] Apurva N. Mody. IEEE 802.22 Working Group on Wireless Regional Area Networks. *http://www.ieee802.org/22/*, September 2010.

[71] Kurt H. Mueller and Markus Müller. Timing Recovery in Digital Synchronous Data Receivers. *IEEE Transactions on Communications*, COM-24(5), May 1976.

[72] Holly C. Osborne. A Generalized "Polarity-Type" Costas Loop for Tracking MPSK Signals. *IEEE Transactions on Communications*, COM-30(10):2289–2296, October 1982.

[73] Open-Source SCA Implementation::Embedded (OSSIE) Development Site. *http://ossie.wireless.vt.edu/*, October 2010.

[74] Thomas W. Parks and James H. McClellan. Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase. *IEEE Transactions on Circuit Theory*, CT-19(2), March 1972.

[75] Jeffrey D. Poston and William D. Horne. Discontiguous OFDM Considerations for Dynamic Spectrum Access in Idle TV Channels. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, pages 607–610, 2005.

[76] John G. Proakis. *Digital Communications.* McGraw-Hill, New York, 4 edition, 2001.

[77] Daiming Qu, Zhiqiang Wand, and Tao Jiang. Extended Ative Interference Cancellation for Sidelobe Suppression in Cognitive Radio OFDM Systems With Cyclic Prefix. *IEEE Transactions on Vehicular Technology*, 59(4), May 2010.

[78] Jan M. Rabaey. Reconfigurable Processing: The Solution to Low-Power Programmable DSP. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 275–278, April 1997.

[79] Lawrence R. Rabiner, James H. McClellan, and Thomas W. Parks. FIR Digital filter Design Techniques Using Weighted Chebyshev Approximations. *Proceedings of the IEEE*, 63(4):595–610, March 1975.

[80] Jeffrey H. Reed, Joseph D. Gaeddert, Kyouwoong Kim, Rekha Menon, Lizdabel Morales, Youping Zhao, Kyung K. Bae, and Jeffrey H. Reed. Development of a Cognitive Engine and Analysis of WRAN Cognitive Radio Algorithms. Technical report, Wireless @ Virginia Tech, December 2007.

[81] Michael Rice and fred harris. Loop Control Architectures for Symbol Timing Synchronization in Sampled Data Receivers. In *MILCOMM Proceedings*, volume 2, pages 987–991, October 2002.

[82] G.A. Rovithakis, M. Maniadakis, and M. Zervakis. A hybrid neural network/genetic algorithm approach to optimizing feature extraction for signal classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(1):695–703, 2004.

[83] Burton R. Saltzberg. Performance of an Efficient Parallel Data Transmission System. *IEEE Transactions on Communication Technology*, COM-15(6):805–811, December 1967.

[84] Amit Sinha, Alice Wang, and Anantha Chandrakasan. Energy scalable system design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(2):135–145, April 2002.

[85] David Slepian. Prolate Spheroidal Wave Functions, Fourier Analysis, and Uncertainty—V: the Discrete Case. *Bell Systems Technical Journal*, 57:1371–1430, May-June 1978.

[86] Steven G. Johnson and Matteo Frigo. A modified split-radix FFT with fewer arithmetic operations. *IEEE Transactions on Signal Processing*, 55(1):111–119, 2007.

[87] George J. Stigler. *Production and Distribution Theories: the Formitive Period.* Macmillan, New York, 1941.

[88] Paul Sutton, Baris Ozgul, Irene Macaluso, and Linda Doyle. OFDM Pulse-Shaped Waveforms for Dynamic Spectrum Access Networks. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, 2010.

[89] Tektronix Test and Measurement Instrumentation. *http://www.tek.com/*, October 2010.

[90] Andrea M. Tonello and Fabio Rossi. Synchronization and Channel Estimation for Filtered Multitone Modulation. In *Proceedings of International Symposium on Wireless Personal Multimedia Communications, 2004*, pages 590–4, September 2004.

[91] J.M. Torrance and L. Hanzo. Upper bound performance of adaptive modulation in a slow Rayleigh fading channel. *IEEE Electronics Letters*, 32(8):718–9, April 1996.

[92] Manuel Uhm. The Power of Software Defined Radio: A Holistic Approach to Designing SDRs for Power. In *COTS Journal*, 1 2007.

[93] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Prentice Hall Signal Processing Series. Prentice Hall, New Jersey, 1993.

[94] T.A. Weiss, J. Hillenbrand, A. Krohn, and F.K. Johndral. Mutual Interference in OFDM-Based Spectrum Pooling Systems. In *IEEE 59th vehicular Technology Conference*, 2004.

[95] Marilynn P. Wylie-Green. Dynamic Spectrum sensing by Multiband OFDM Radio for Interference Mitigation. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, pages 619–625, 2005.

[96] Shirang M. Yardi, Michael S. Hsiao, Thomas L. Martin, and Dong S. Ha. Quality-Driven Proactive Computation Elimination for Power-Aware Multimedia Processing. *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2005.

[97] Qian Zhang, Zhu Ji, and Ya-Qin Zhang. Power-Minimized Bit Allocation for Video Communication Over Wireless Channels. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6), June 2002.

[98] Youping Zhao, Joseph D. Gaeddert, Lizdabel Morales, Kyung Bae, Jung-Sun Um, and Jeffrey H. Reed. Development of Radio Environment Map Enabled Case- and Knowledge-Based Learning Algorithms for IEEE 802.22 WRAN Cognitive Engines. In *IEEE International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, pages 44–49, 8 2007.

[99] Youping Zhao, Lizdabel Morales, Joseph D. Gaeddert, Kyung Bae, Jung-Sun Um, and Jeffrey H. Reed. Applying Radio Environment Maps to Cognitive Wireless Regional Area Networks. In *IEEE International Symposium on Dynamic Spectrum Access Networks*, pages 115–8, 2007.

[100] Rodger E. Ziemer, William H. Tranter, and D. Ronald Fannin. *Signals & Systems: Continuous and Discrete*. Prentice Hall, Upper Saddle River, NJ, 4 edition, 1998.