Discrete-Time Implementation, Antenna Design, and MIMO for Near-Field Magnetic Induction Communications

Ronald Brett Gottula
Brigham Young University - Provo
Discrete-Time Implementation, Antenna Design, and MIMO for Near-Field Magnetic Induction Communications

R. Brett Gottula

A thesis submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Master of Science

Karl F. Warnick, Chair
Michael D. Rice
Michael J. Wirthlin

Department of Electrical and Computer Engineering
Brigham Young University
August 2012

Copyright © 2012 R. Brett Gottula
All Rights Reserved
ABSTRACT

Discrete-Time Implementation, Antenna Design, and MIMO for Near-Field Magnetic Induction Communications

R. Brett Gottula
Department of Electrical and Computer Engineering, BYU
Master of Science

Near-field magnetic induction (NFMI) is a short range wireless technology that uses loop antennas coupled by a magnetic field. NFMI antennas are electrically small and thus extremely inefficient and narrow band, making system design for multi-user and high-bitrate applications challenging. The goals of this thesis are to develop a test platform suitable for NFMI antenna testing, to model, design and test NFMI antennas that have high bandwidth-efficiency, and to explore the possibility of using MIMO (multiple-input multiple-output) to increase the capacity of the NFMI channel.

This thesis provides system implementations, test results, and channel modeling to aid in the design of future NFMI systems. Implementation of a multi-channel discrete-time wireless system are provided for PC-based software and FPGA-based firmware as a platform for antenna testing. Optimized antenna designs in terms of efficiency and bandwidth are presented, achieving the theoretical bandwidth-efficiency bound for small antennas. Preliminary modeling and simulation results for the NFMI-MIMO channel are included, which show that the information-theoretic capacity of the NFMI-MIMO channel is approximately double the standard single-antenna NFMI capacity at 10 bits/s/Hz.

Keywords: near field magnetic induction, wireless communications, MIMO
ACKNOWLEDGMENTS

I have been privileged to work with many outstanding individuals who have supported and contributed to this research. Dr. Warnick and Dr. Rice have provided many hours of mentoring and guidance. I also express gratitude for the many student researchers who have contributed to the FreeLinc project: Ben Bush, Michael McCarty, Chris Shaw, Josh Bradley, Danny Savory, Matt Manwaring, Patrick Getz, Steven Freestone, McKay Killpack, Nick Valletta, Levi Brown, David Nuttall, Kyle Browning, Sean Kerman, Rajan Komanduri, Bradford Law, Jorge Conde, Sushant Shrestha, and James Smith.

I would also like to thank FreeLinc executives Doug Dobyns and Randy Bailey for their great encouragement. The BYU Near-Field Wireless Link project is sponsored by a research grant from FreeLinc, Inc., which has been a tremendous luxury.

Finally, I would like to express appreciation to my father Ron Gottula, who encouraged me to go to graduate school and whose hard work, ambition and academic excellence throughout his life have been a constant source of inspiration.
Contents

List of Figures viii

List of Tables xi

List of Listings xii

1 Introduction 1

1.1 Motivations .......................................................... 2

1.2 Contributions ......................................................... 2

1.3 Thesis Organization ................................................. 3

2 Background 4

2.1 Applications of NFMI ................................................. 5

2.1.1 On-Body Networks ............................................. 5

2.1.2 NFC .............................................................. 5

2.1.3 Medical Sensors ................................................ 6

2.2 Electrically Small Antennas ....................................... 6

3 Discrete-Time Wireless System 8

3.1 Transmitter ............................................................ 9

3.2 Receiver .............................................................. 10

3.2.1 Channelizer ....................................................... 10

3.2.2 Matched Filter .................................................. 13

3.2.3 Synchronizer ..................................................... 13
3.2.4 Interpolator ....................................................... 15
3.2.5 DQPSK Detector .................................................. 15
3.2.6 Receiver Implementations ...................................... 17
3.2.7 Receiver Performance Verification .......................... 18
3.3 $E_b/N_0$ Estimation .............................................. 22
   3.3.1 Spectral SNR Estimator .................................... 22
   3.3.2 SNV Estimator ............................................. 24
   3.3.3 Constellation-based Estimator ............................ 25

4 Antennas .................................................................. 27
   4.1 Introduction ...................................................... 27
   4.2 Hertzian Magnetic Dipole Model ........................... 28
   4.3 Comparison of Antenna Performance ..................... 30
      4.3.1 Transmit Power ........................................... 31
      4.3.2 Hollow-Core Ferrite-Loaded Loop Antenna .......... 32
      4.3.3 Quad-Core Ferrite-Loaded Loop Antenna .......... 32
      4.3.4 Solid Ferrite-Loaded Loop Antenna ................ 32
      4.3.5 Dual-Tuned Ferrite-Loaded Loop Antenna .......... 34
   4.4 Conclusions ...................................................... 34

5 Time Division Multiple Access System ......................... 37
   5.1 Definitions ...................................................... 37
   5.2 Requirements .................................................. 38
   5.3 Design .......................................................... 39
   5.4 Implementation ................................................ 41
      5.4.1 Device ID .................................................. 42
      5.4.2 SOPC Builder System .................................. 42
      5.4.3 Software .................................................. 42
5.4.4 Interval Timers ................................. 44
5.4.5 Custom FPGA Circuits .......................... 44
  5.4.5.1 Packet Detection ........................... 45
  5.4.5.2 Transmitter On Block ....................... 45
  5.4.5.3 Configuration Registers ..................... 48
5.5 Calibration and Test .............................. 48

6 MIMO ........................................... 51
  6.1 Introduction .................................... 51
  6.2 MIMO Channel Capacity .......................... 52
  6.3 Simulated Channel Capacity ....................... 53
    6.3.1 Fixed Transmit Power Simulation .......... 56
    6.3.2 Fixed SNR Simulation ....................... 56
  6.4 Conclusions .................................... 58

7 Conclusions .................................... 60
  7.1 Future Work ..................................... 61
    7.1.1 Matching Networks ........................... 61
    7.1.2 Channel Sounding ............................ 61
    7.1.3 MIMO Implementations ....................... 61

References ....................................... 63

A Antenna Test Results ............................ 66
  A.1 Hollow-Core Ferrite-Loaded Loop Antenna ........ 66
  A.2 Quad-Core Ferrite-Loaded Loop Antenna .......... 71
  A.3 Solid-Core Ferrite-Loaded Loop Antenna .......... 76
  A.4 Dual-Tuned Ferrite-Loaded Loop Antenna .......... 81
B Automated Test System

B.1 ATS Components ................................................. 90
B.2 Tests Performed .................................................. 91
B.3 Data Set Contents ................................................ 92
B.4 Data Set Index ..................................................... 93
B.5 MATLAB Utilities ............................................... 93
B.6 Automatic Update ............................................... 93
B.7 Wiki Usage Guide ............................................... 95

C TDMA Configuration Registers ...................................... 96

D Quasistatic Field Model Source Code .............................. 98

E Software Receiver Code Listing .................................. 104
List of Figures

1.1 NFMI power versus distance ................................................. 2

2.1 Equivalent circuit for near-field magnetic induction ...................... 4

3.1 Block diagram of the discrete-time transmitter ................................. 9

3.2 System Generator transmitter .............................................. 11

3.3 Block diagram of the discrete-time receiver ................................ 12

3.4 Channelizer frequency allocation ............................................ 12

3.5 Channelizer block diagram .................................................. 12

3.6 Channelizer low-pass filter .................................................. 13

3.7 Synchronizer block diagram .................................................. 14

3.8 Farrow interpolator structure .............................................. 16

3.9 System Generator receiver .................................................. 19

3.10 Simulated software receiver performance ................................ 21

3.11 $E_b/N_0$ estimator performance comparison .............................. 23

3.12 Simulated performance of constellation-based $E_b/N_0$ estimator ...... 26

4.1 Magnetic field of Hertzian dipole ........................................... 29

4.2 Antenna S parameter response comparison ................................ 33

5.1 Topology of the TDMA network ............................................ 38

5.2 SOPC Builder screenshot .................................................... 43

5.3 Receiver bit sink circuit ..................................................... 46

5.4 Packet detector circuit ...................................................... 47
5.5  Tx On circuit ................................................................. 49
5.6  TDMA on oscilloscope .................................................. 50
6.1  Simulated NFMI-MIMO singular values for fixed Tx power ..... 57
6.2  Simulated NFMI-MIMO capacity for fixed Tx power .......... 58
6.3  Simulated NFMI-MIMO singular values ............................ 59
6.4  Simulated NFMI-MIMO capacity .................................... 59
A.1  Hollow-core antenna at 0.5 meters ............................... 67
A.2  Hollow-core antenna at 0.75 meters ............................. 68
A.3  Hollow-core antenna at 1.0 meters ............................... 69
A.4  Hollow-core antenna at 1.25 meters ............................. 70
A.5  Quad-core antenna at 0.5 meters .................................. 72
A.6  Quad-core antenna at 0.75 meters ............................... 73
A.7  Quad-core antenna at 1.0 meters .................................. 74
A.8  Quad-core antenna at 1.25 meters ............................... 75
A.9  Solid-core antenna at 0.5 meters ................................. 77
A.10 Solid-core antenna at 0.75 meters ............................... 78
A.11 Solid-core antenna at 1.0 meters .................................. 79
A.12 Solid-core antenna at 1.25 meters ............................... 80
A.13 Dual-tuned antenna at 0.5 meters, 12.72 MHz .................. 82
A.14 Dual-tuned antenna at 0.75 meters, 12.72 MHz ............... 83
A.15 Dual-tuned antenna at 1.0 meters, 12.72 MHz ................. 84
A.16 Dual-tuned antenna at 1.25 meters, 12.72 MHz ............... 85
A.17 Dual-tuned antenna at 0.5 meters, 13.56 MHz ............... 86
A.18 Dual-tuned antenna at 0.75 meters, 13.56 MHz ............... 87
A.19 Dual-tuned antenna at 1.0 meters, 13.56 MHz ............... 88
A.20 Dual-tuned antenna at 1.25 meters, 13.56 MHz ............... 89
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Properties of the discrete-time wireless system.</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>Differential encoding definition.</td>
<td>9</td>
</tr>
<tr>
<td>3.3</td>
<td>Channelizer low-pass filter design criteria.</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Piecewise parabolic interpolator coefficients $b_l(i)$.</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Antenna design summary.</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Antenna performance summary.</td>
<td>34</td>
</tr>
<tr>
<td>5.1</td>
<td>TDMA data rate requirements.</td>
<td>39</td>
</tr>
<tr>
<td>5.2</td>
<td>TDMA frame definition.</td>
<td>40</td>
</tr>
<tr>
<td>5.3</td>
<td>TDMA packet definition.</td>
<td>40</td>
</tr>
<tr>
<td>5.4</td>
<td>TDMA packet header definition.</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>Antenna properties.</td>
<td>54</td>
</tr>
</tbody>
</table>
List of Listings

D.1  driver.m .......................................................... 98
D.2  loadFerriteParameters.m ......................................... 99
D.3  R_ant.m ............................................................ 99
D.4  R_fer.m ............................................................ 99
D.5  R_loss.m ............................................................ 99
D.6  R_rad.m ............................................................ 100
D.7  L_ant.m ............................................................ 100
D.8  matchingC.m ........................................................ 100
D.9  L_ant.m ............................................................ 100
D.10 I_0.m ............................................................... 101
D.11 mu_rod.m ........................................................... 101
D.12 H_field.m .......................................................... 101
D.13 v_oc.m ............................................................. 102
D.14 v_L.m .............................................................. 102
E.1  DSP.h ............................................................... 104
E.2  DSP.cpp ............................................................ 108
Chapter 1

Introduction

Near-field magnetic induction (NFMI) is a physical layer for short-range wireless communication. NFMI operates on the same basic principles as a transformer, where current in a loop of wire produces a magnetic field which induces a current in a separate loop of wire. Unlike transformers, however, the two coils are separated by a greater distance and are only weakly coupled. NFMI operates in the near field, where most of the energy is confined to a compact region near the transmitter and does not radiate away. This is in contrast to conventional radio frequency (RF) wireless communication, where most of the energy propagates away from the transmitter into free space.

NFMI has several advantages over traditional RF technology for short-range wireless communication [1]. Near-field signal power rolls off at $-60$ dB per decade of distance, compared to $-20$ dB per decade for the far field, as shown in Figure 1.1. This steep rolloff rate results in a well-defined operational boundary or “bubble” where communication can occur. This boundary provides a natural form of security since detection of the signal outside the operating radius is challenging or impossible. The bubble also allows for generous bandwidth reuse, because users located beyond the operational range of a neighboring NFMI system can use the same frequency without interference. NFMI systems do not suffer the fading effects of multipath that plague traditional RF systems, and devices placed on opposite sides of the human body experience no link degradation. Finally, NFMI systems generally require less power than comparable RF systems.
Figure 1.1: Signal power versus distance for an NFMI system operating at 13.5 MHz. Power is expressed in decibels referenced to the signal power at 0.1 meters. The rolloff in the near field region is −60 dB per decade, and in the far field the rolloff is −20 dB per decade. At 13.56 MHz the knee in the curve is at approximately 3.5 meters—well beyond the typical 1.5 meter operational range of an NFMI communications system.

1.1 Motivations

Some of the most exciting potential applications of NFMI come in the form of on-body networks, where the security and reliability of NFMI technology provides a distinct advantage over alternatives such as Bluetooth. However, many of these applications require multi-channel capabilities and data rates that are not possible with existing NFMI systems due to limited power and bandwidth. These limitations motivate research to discover new antenna designs and signal processing techniques that can increase the capabilities of NFMI without drastically impacting power consumption, size, and complexity.

1.2 Contributions

This thesis contributes the following:

1. Design and implementation of a discrete-time signal processing system for multi-channel NFMI communications;
2. Recommendations for NFMI antenna design based on modeling, experimentation, and testing, which also includes

(a) An automated software/hardware system for antenna testing
(b) Code for estimation of system parameters such as received signal power, SNR, and bit error rate
(c) An detailed archive of thousands of individual antenna tests
(d) MATLAB scripts for statistical analysis of data in the archive;

3. Design and implementation of an efficient time division multiple access (TDMA) layer for on-body NFMI systems hosted on FPGA platforms;

4. Simulation results for the 3-by-3 NFMI channel, including channel capacity analysis and recommendations for future NFMI MIMO research.

1.3 Thesis Organization

Chapter 2 gives some background on NFMI and a brief survey of relevant literature. The design and implementation of the discrete-time signal processing system for multi-channel NFMI is presented in Chapter 3. Antenna modeling, design, test, and evaluation is the topic of Chapter 4. Chapter 5 documents the design and implementation of TDMA for NFMI. Channel sounding experiments and results for NFMI MIMO are discussed in Chapter 6, and concluding remarks and recommendations for future work are contained in Chapter 7.

Code listings, system documentation, and plots of experimental results which are too lengthy for inclusion in the body of the text have been included in several appendices. These include antenna test results (Appendix A), a description of the Automated Test System (Appendix B), configuration registers for the TDMA system (Appendix C), source code for the quasistatic NFMI field model (Appendix D), and source code for significant DSP portions of the software receiver (Appendix E).
Chapter 2

Background

Near-field magnetic induction (NFMI) communication works by modulating the magnetic near field using an inductive coil, which induces current flow in a receiver coil. Figure 2.1 is an equivalent circuit for a typical NFMI system, showing the inductive antennas, simple matching circuit consisting of a single capacitor, and the source and load elements. This arrangement is possible in the magnetic near field, which is loosely defined as the region within a half-wavelength of the transmitter. For small antennas, the radiation resistance is small, and thus most of the energy is reflected back to the source and very little is radiated into free space. As explained in Chapter 1, the received signal power in the near field
typically rolls off at the rapid rate of -60 dB per decade. The near field is also distinct in
terms of polarization, because all three polarization states are present rather than the two
transverse polarizations of a propagating far field signal.

2.1 Applications of NFMI

NFMI has many applications to short-range wireless communication. Advantages
in terms of power consumption, security, spectrum reuse, immunity to interference, and
human body penetration over traditional RF solutions have sparked renewed interest in
NFMI technology. On-body networks, NFC, and medical sensors are emerging examples of
NFMI short range communications.

2.1.1 On-Body Networks

For personal communication, NFMI can be used to form a body-area network (BAN)
[2, 3]. For this application, several devices worn on the body communicate within a near
field “bubble” that is just large enough to encircle most of the body. This limited region
provides a significant security benefit for sensitive communications because eavesdropping
from a distance is virtually impossible, even without encryption. Products produced by
FreeLinc, Inc. promise secure voice links between headsets and larger waist-worn radios for
use by public safety officers and federal agents [4]. In addition to voice communication,
on-body applications include streaming video, sensors, push-to-talk remotes, headsets, and
medical implants.

2.1.2 NFC

Near Field Communication (NFC) is a standardized subset of NFMI defined by an
industry group known as the NFC Forum. NFC is intended to facilitate short-range wireless
applications such as access control, point-of-sale payment, information exchange, proximity-
based advertising, and coupons [5, 6]. NFC typically operates at distances of 4 cm or
less, meaning that the devices must be nearly touching for a link to be established. The quick rolloff of near-field power provides intrinsic security for NFC transactions even when encryption or other security measures are not used.

2.1.3 Medical Sensors

Another exciting application of NFMI technology is medical monitoring and implants [7]. Most medical sensors are wired, which inconvenient and in some cases hazardous. Implanted sensors present a particular challenge, because any breach in the skin increases risk of infection and the human body is opaque to most RF signals. Even where RF solutions such as Bluetooth are feasible, the short battery life and the potential for eavesdropping on sensitive health information is a great concern. NFMI can potentially solve many of these problems by providing a solution that is secure, able to penetrate the human body, and low-power. Simple NFMI sensors can even harvest power from the magnetic fields, eliminating the requirement to replace or recharge a battery.

2.2 Electrically Small Antennas

NFMI systems typically use the unlicensed 13.56 MHz industrial, scientific and medical (ISM) band. This band has a half-wavelength of approximately 11 meters, which allows link distances of a few meters within the near field region. However, the long wavelength also implies that NFMI antennas that are unavoidably electrically small. The bandwidth and efficiency of NFMI antennas are constrained by the theoretical bounds on the performance of electrically small antennas.

Research has shown that electrically small antennas have high quality factor (Q), narrow bandwidth, and high efficiency, or broad bandwidth and low efficiency. Early work on small antennas confined to a sphere of a given radius provided lower bounds on Q, which limits the impedance matching bandwidth of an antenna driven by an amplifier [8, 9]. Increasing availability of low-loss, high-permeability ferrite material in the 1950’s improved the
efficiency-bandwidth of receive antennas by increasing the impedance [10]. Analytical models were later developed to aid in the design of optimal volumetrically constrained antennas [11]. More recent work has shown that the Q of an electrically small antenna with fixed radius can be minimized by utilizing the volume of a minimum enclosing sphere to the maximum extent possible, and hollow spheroidal core design results in maximum bandwidth-efficiency for a fixed volume of ferrite material [12, 13].

Despite the availability of theory and design guidelines for small antennas, many designs fall far short of the theoretical bound on the bandwidth-efficiency product. In a recent comparison of 110 antenna designs appearing in the IEEE Transactions on Antennas and Propagation, only a few approached the theoretical bound [14]. Nevertheless, the study does confirm that achieving bandwidth-efficiency performance near the bound is a realistic goal. Many NFMI applications require more bandwidth than is available with current antennas. Consequently, the design of an efficient, small, broadband NFMI antenna with practical geometric constraints is an important topic of research.
Chapter 3

Discrete-Time Wireless System

Before antenna testing is possible, a complete wireless system suitable for such experimental antenna testing must exist. In this chapter, the design and implementation of the discrete-time portions of a complete NFMI wireless system are described.

The discrete-time signal processing algorithms for the wireless system were developed in consideration of the need for multi-channel operation and the research team’s need to produce a working system in a short time frame. Design work was performed with significant assistance from Dr. Michael Rice, a professor in the BYU Electrical and Computer Engineering Department, and Chris Shaw, a graduate student in the same department. The final design is a 13-channel system with 250 kHz bandwidth per channel, 250 kbps bitrate, and differential QPSK modulation.

Several options for implementation of the discrete-time algorithms exist. An ideal system for research would be flexible, reprogrammable, and provide visibility into the signal processing algorithms for study, evaluation, and debugging. For discrete-time signal processing, field programmable gate array (FPGA) technology and traditional PC-based software processing provide two possible implementation platforms. Ultimately both software and FPGA implementations of the transmitter and receiver were developed. Software was written in the C programming language, and FPGA development was done using Xilinx System Generator and Altera DSP Builder.
Figure 3.1: Block diagram of the discrete-time transmitter.

Table 3.1: Properties of the discrete-time wireless system.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulation</td>
<td>Differential QPSK</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>250 kbps</td>
</tr>
<tr>
<td>Symbol Rate</td>
<td>125 ksymbols /s</td>
</tr>
<tr>
<td>Pulse Shape</td>
<td>Square root raised cosine (SRRC)</td>
</tr>
<tr>
<td>Pulse Shape Excess Bandwidth</td>
<td>100%</td>
</tr>
<tr>
<td>Spectral Efficiency</td>
<td>1 bit /s /Hz</td>
</tr>
</tbody>
</table>

3.1 Transmitter

The discrete-time transmitter consists of a binary source, differential QPSK encoder, upsamplers, pulse shaping filters, and a discrete-time mixer. A block diagram of the transmitter is shown in Figure 3.1. The bit rate, modulation, and pulse shaping filter design parameters are given in Table 3.1, and the differential encoding definition is given in Table 3.2. The design was proven by implementing and testing it in MATLAB.

The MATLAB implementation served as a prototype for the first FPGA implementation, which was targeted for the Xilinx Xtreme DSP Development Kit, Virtex IV Edition.

Table 3.2: Differential encoding definition.

<table>
<thead>
<tr>
<th>Transition Angle</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>( \pi/2 )</td>
<td>01</td>
</tr>
<tr>
<td>( \pi )</td>
<td>00</td>
</tr>
<tr>
<td>( 3\pi/2 )</td>
<td>10</td>
</tr>
</tbody>
</table>
Early coding was done in VHDL, but development quickly transitioned to Xilinx System Generator, which allows system designers to develop in the Simulink environment using a special blockset that generates HDL code. The System Generator transmitter in Figure 3.2 loosely matches transmitter block diagram in Figure 3.1. The counter and ROM blocks are the binary source, the BitBasher and ROM3 blocks are the DQPSK encoder, the FIR compiler blocks are the pulse-shaping filters $H(z)$, and the discrete-time mixer includes the DDS block and the multiply blocks. The upsampling is handled automatically by System Generator and thus does not have a dedicated block.

A second FPGA implementation was created for the Altera DSP Development Kit, Cyclone III Edition using Altera’s DSP Builder, which is equivalent to System Generator but for Altera FPGAs. This system was developed at FreeLinc due to the difficulty and expense of obtaining the older Xilinx Xtreme DSP development kit initially used at BYU. The Altera transmitter is functionally identical to the Xilinx version.

3.2 Receiver

The discrete-time receiver consists of a channelizer, matched filter, synchronizer, interpolator, and detector. A block diagram of the receiver is shown in Figure 3.3. The same system properties of Table 3.1 also apply to the receiver.

3.2.1 Channelizer

The channelizer performs channel selection, downsampling, and conversion to baseband.

Channel selection filters the incoming broadband signal to accept a narrowband signal centered at a chosen carrier frequency and rejects signals at other frequencies. The 12–15 MHz band is divided into thirteen 250 kHz wide channels centered at 12.00, 12.25, 12.50, 12.75, 13.00, 13.25, 13.50, 13.75, 14.00, 14.25, 14.50, 14.75, and 15.00 MHz as shown in Figure 3.4.
Figure 3.2: The transmitter as implemented using System Generator for Xilinx FPGAs.
Because FreeLinc devices may use multiple channels simultaneously, the channelizer was designed with a polyphase filter architecture. The polyphase architecture makes all 13 channels available at the channelizer output simultaneously with minimal extra processing required. The structure of this filter is shown in Figure 3.5. The filters $H_0(z^{M/D})$ to $H_{M-1}(z^{M/D})$ in Figure 3.5 are subfilters formed out of a larger low-pass filter $H(z)$. The filter $H(z)$ was designed with the criteria given in Table 3.3. For $M = 72$, $D = 36$, and

$$y(m) = y_1(nT/D)$$
Table 3.3: Channelizer low-pass filter design criteria.

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>passband</td>
<td>$0 \leq F \leq \frac{1}{2M}$ cycles/sample</td>
</tr>
<tr>
<td>transition band</td>
<td>$\frac{1}{2M} &lt; F \leq \frac{1}{M}$ cycles/sample</td>
</tr>
<tr>
<td>stop band</td>
<td>$\frac{1}{M} &lt; F \leq \frac{1}{2}$ cycles/sample</td>
</tr>
<tr>
<td>stop band attenuation</td>
<td>60 dB</td>
</tr>
<tr>
<td>filter length</td>
<td>multiple of $M$</td>
</tr>
</tbody>
</table>

a filter length of 432, the filter shown in Figure 3.6 results, in which case each of the 72 polyphase filters are length $432/72 = 6$.

3.2.2 Matched Filter

The matched filter is identical to the square root raised cosine pulse shaping filter used in the transmitter. The parameters of this filter are given in Table 3.1.

3.2.3 Synchronizer

Because the receiver and transmitter use independent clocks, the receiver must recover symbol timing information and interpolate between the available samples in order for the detector to make optimum decisions. Timing information is recovered by using the discrete
square timing recovery algorithm proposed by Oerder and Meyr in [15]. This algorithm operates in block mode, and assumes that the timing offset varies slowly over many symbols. For a sample rate of 4 samples per symbol, the algorithm simplifies to the form shown in Figure 3.7. The output of the synchronizer, $\hat{\varepsilon}$, is converted to a basepoint index $m$ and a fractional interval $\mu$ by

$$m = \begin{cases} 
0 & 0 \leq \hat{\varepsilon} < \frac{1}{2} \\
1 & \frac{1}{2} \leq \hat{\varepsilon} < 1 \\
2 & -1 \leq \hat{\varepsilon} < -\frac{1}{2} \\
3 & -\frac{1}{2} \leq \hat{\varepsilon} < 0
\end{cases}$$

(3.1)

and

$$\mu = \begin{cases} 
2\hat{\varepsilon} & 0 \leq \hat{\varepsilon} < \frac{1}{2} \\
2(\hat{\varepsilon} - \frac{1}{2}) & \frac{1}{2} \leq \hat{\varepsilon} < 1 \\
2(\hat{\varepsilon} + 1) & -1 \leq \hat{\varepsilon} < -\frac{1}{2} \\
2(\hat{\varepsilon} + \frac{1}{2}) & -\frac{1}{2} \leq \hat{\varepsilon} < 0
\end{cases}.$$

(3.2)
Table 3.4: Piecewise parabolic interpolator coefficients \( b_l(i) \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( b_2(i) )</th>
<th>( b_1(i) )</th>
<th>( b_0(i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>( \alpha )</td>
<td>( -\alpha )</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>( -\alpha )</td>
<td>( 1 + \alpha )</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>( -\alpha )</td>
<td>( \alpha - 1 )</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>( \alpha )</td>
<td>( -\alpha )</td>
<td>0</td>
</tr>
</tbody>
</table>

3.2.4 Interpolator

The basepoint index \( m \) and fractional interval \( \mu \) produced by the timing synchronizer are used to interpolate between the matched filter output samples. Interpolation is performed using a piecewise parabolic interpolator of the form

\[
x ( [m(k) + \mu(k)] T) = [v(2)\mu(k) + v(1)] \mu(k) + v(0),
\]

where

\[
v(l) = \sum_{i=-2}^{1} b_l(i)x ([m(k) - i] T),
\]

and \( b_l(i) \) is as defined in Table 3.4. This is implemented using the Farrow structure shown in Figure 3.8. [16]

3.2.5 DQPSK Detector

The interpolator output, which has one sample per symbol, forms the input to a maximum likelihood differential-QPSK detector. The detector takes the previous symbol and computes three rotations of the previous symbol at angles \( \pi/2 \), \( \pi \), and \( 3\pi/2 \). The Euclidean distance between the four resulting points and the current symbol are then computed, and the detector chooses the rotation with the minimum result as the maximum likelihood decision. The detector outputs the two bits corresponding to the detected rotation according to the differential encoding definition in Table 3.2.
Figure 3.8: Farrow interpolator structure. Based on Figure 8.4.16 of [16].
3.2.6 Receiver Implementations

The receiver was implemented following same methodology used for the transmitter, starting with MATLAB and progressing to FPGA-based systems. Unlike the transmitter, however, a near real-time implementation of the receiver was developed in the C programming language to run on a desktop PC. The software version was developed due to the then-urgent need for an antenna testing platform, which could not wait for an FPGA version to become available. The receiver is more complicated than the transmitter, and consequently development and debugging consumed more time. After a basic version of the software receiver was complete, an FPGA implementation was created.

The software implementation of the receiver was at first a direct translation of the MATLAB version into C syntax, but has since evolved into an elaborate architecture with support for bit error rate estimation, $E_b/N_0$ estimation (see Section 3.3), synchronization with the transmitter, and automatic archiving of antenna test data (see Appendix E for a description of the Automated Test System). Analog signals are sampled by an Adlink ADC situated on the PCI bus of the PC. Processing of the samples is performed block-mode such that each buffer acquired from the ADC is completely processed before the next buffer is retrieved. The software receiver is capable of processing multiple frequency channels simultaneously by means of a threaded architecture, where each active channelizer output is processed in a separate thread. The Hewlett-Packard PC that hosts the software receiver is capable of processing up to two channels simultaneously in real time.\footnote{The PC includes a dual-core Intel processor. It is likely that the number of channels that can be processed in real time is directly related to the parallel processing capabilities of the processor.} As of April 2012 the entire receiver project contained over 6,500 lines of code split between 67 source files, thus only the signal processing portions of the code have been included in this thesis in Appendix E.

After the software receiver was available for antenna testing, an FPGA version was developed. The first FPGA receiver was implemented using System Generator for the Xilinx
Xtreme DSP Development Kit, Virtex IV Edition. This development effort was more challenging to accomplish using System Generator because the most complex algorithms—the channelizer, timing synchronizer, and interpolator—could not be implemented using single blocks out of the System Generator blockset. Instead these algorithms were implemented from scratch using low-level blocks such as delay elements, adders, and multipliers. Limited understanding of efficient FPGA architectures for these algorithms complicated the development process, but after much effort and experimentation a working FPGA implementation of the receiver was completed. Unfortunately this version was not integrated into the antenna testing system due to bugs in some Xilinx APIs for transferring data between the host PC and the FPGA, but it did demonstrate the feasibility of using System Generator and similar tools to develop complex DSP algorithms for FPGAs. Later an Altera version of the receiver was developed at FreeLinc for product development purposes using Altera’s DSP Builder, which is very similar to Xilinx System Generator.

The top-level view of the System Generator receiver is presented in Figure 3.9. Unlike the transmitter, each block in Figure 3.9 is a Simulink subsystem rather than a block from the System Generator blockset. Each subsystem contains its own set of blocks and signal connections, and may contain nested subsystems.

3.2.7 Receiver Performance Verification

The quality of the discrete-time receiver signal processing can be evaluated by measuring its bit error rate performance in an additive white Gaussian noise (AWGN) channel. The FreeLinc receiver uses a differential quadrature phase-shift keying (DQPSK) detector with Gray coding, which has a theoretical bit error rate performance given by

\[
P_b = Q_1 (a, b) - \frac{1}{2} I_0 (ab) e^{-\frac{\sigma^2}{2}},
\]

(3.5)
Figure 3.9: The receiver as implemented using System Generator for Xilinx FPGAs.
where \( Q_1(a, b) \) is the Marcum Q function given by

\[
Q_1(a, b) = \int_b^\infty x e^{-\frac{x^2 + a^2}{2}} I_0(ax) \, dx, \tag{3.6}
\]

\( I_0(x) \) is the 0th order modified Bessel function

\[
I_0(x) = \sum_{k=0}^\infty \left( \frac{x^k}{2^k k!} \right)^2, \tag{3.7}
\]

and \( a \) and \( b \) are

\[
a = \sqrt{\frac{2E_b}{N_0} \left( 1 - \sqrt{\frac{1}{2}} \right)}, \tag{3.8}
\]

and

\[
b = \sqrt{\frac{2E_b}{N_0} \left( 1 + \sqrt{\frac{1}{2}} \right)}, \tag{3.9}
\]

respectively, where \( E_b/N_0 \) is the normalized SNR per bit [17]. A good implementation of a DQPSK receiver should achieve this predicted performance in an ideal AWGN channel.

The software receiver achieves the theoretical bit error rate performance using simulated input samples. The input samples were generated using a MATLAB implementation of the discrete-time transmitter using a pseudorandom binary sequence as its input. This signal, sampled at the same rate as the ADC of the software receiver (20 MHz), was corrupted with MATLAB-generated white Gaussian noise to produce a signal with known signal-to-noise ratio \( (E_b/N_0) \). The noisy MATLAB-generated samples were then introduced to the software receiver in place of the ADC. The receiver processed the signal and estimated the bit error rate of the detected binary sequence by comparing to the original pseudorandom pattern. This process was repeated for many realizations of the simulated samples and for several values of \( E_b/N_0 \). The results of this simulation are plotted in Figure 3.10, which shows excellent agreement between theory and simulation.
Figure 3.10: C software receiver performance using MATLAB generated input samples with AWGN. The solid curve is the theoretical bit error rate performance of a differential QPSK detector in the AWGN channel.
3.3 $E_b/N_0$ Estimation

The discrete-time wireless system was developed in part to facilitate antenna testing, where one of the fundamental figures of merit is the signal-to-noise ratio (SNR) at the receiver. In a communications channel with additive white Gaussian noise, the bit error rate of the system is determined by a normalized SNR denoted $E_b/N_0$, which is the ratio of the energy per bit to the power spectral density of the noise. In a real system, the value of $E_b/N_0$ is not known at the receiver but can be estimated from the received signal and prior knowledge of the transmitted signal. Over the course of this project three separate $E_b/N_0$ estimators have been implemented to support antenna testing. These estimators are described and compared in this section.

3.3.1 Spectral SNR Estimator

The first estimator uses estimates of the time-average signal power to determine $E_b/N_0$. The time-average signal power is estimated for all 13 frequency channels, where the estimate for a single channel is $\hat{P}_{av,j}$, $0 \leq j \leq 12$. If the transmitter is transmitting on channel $i$, $0 \leq i \leq 12$, then $\hat{P}_{av,i}$ includes power from both the signal and the noise, and the $\hat{P}_{av,k}$, $k \neq i$ are assumed to include only noise power. The noise power estimate $\hat{\eta}_i$ is

$$\hat{\eta}_i = \frac{1}{12} \sum_k \hat{P}_{av,k} \quad k \neq i,$$

which is the sample mean of the 12 $\hat{P}_{av,k}$. The SNR estimate $\hat{\rho}_i$ is given by

$$\hat{\rho}_i = \frac{\hat{P}_{av,i}}{\hat{\eta}_i} - 1,$$
Figure 3.11: Comparison of $E_b/N_0$ estimator performance using real data obtained during antenna testing. Estimated values of $E_b/N_0$ are plotted versus the expected values, and the ideal case with unit slope is shown by the solid line.
where the subtraction by 1 is required because the estimate \( \hat{P}_{av,i} \) includes both signal and noise power. To convert SNR to \( \frac{E_b}{N_0} \), \( \hat{\rho}_i \) is normalized such that

\[
\frac{E_b}{N_0} = \frac{B}{f_b} \hat{\rho}_i, \tag{3.12}
\]

where \( B = 250 \text{ kHz} \) is the channel bandwidth and \( f_b = 250 \text{ kbps} \) is the net bitrate.

This estimator was abandoned because its outputs did not correlate well with the expected values of \( \frac{E_b}{N_0} \). Figure 3.11a is a plot of the estimated values of \( \frac{E_b}{N_0} \) versus the expected values over dozens of individual antenna tests, where the expected values were taken to be those values that correspond to the observed bit error rate according to Equation 3.5. The poor correlation between these quantities in Figure 3.11a is clear.

### 3.3.2 SNV Estimator

After abandoning the spectral SNR estimator, the Squared Signal-to-Noise Variance (SNV) Estimator based on [18] was implemented. This proved challenging because the SNV estimator assumes coherent detection, but the FreeLinc receiver uses differential detection and does nothing to estimate carrier frequency offset. To make the FreeLinc receiver compatible with the SNV algorithm, a frequency offset estimator was implemented and the matched filter outputs were processed using this information to approximate coherent detection. Under ideal conditions this configuration produced good estimates, but the frequency offset estimator was unreliable. This poor reliability resulted in the bimodal behavior evident in Figure 3.11b, which shows two distinct regions of estimates, one reasonable and one unusable. After major efforts to improve the reliability of this estimator failed, a new approach was explored.
3.3.3 Constellation-based Estimator

After abandoning both the spectral and SNV $E_b/N_0$ estimators, a new constellation-based estimator was derived. To describe this estimator, it is convenient to start with a system model that neglects carrier frequency offset. In such a system, the matched filter outputs can be modeled as a sequence of QPSK symbols in the complex plane added to a complex white Gaussian noise process with zero mean and variance $\sigma^2$. To estimate the variance of the noise, the phase rotations introduced by the differential QPSK encoder must first be removed such that all of the matched filter outputs are centered around a single point in the IQ plane. After this operation, the sample mean and sample variance estimators are used to form the estimate of $E_b/N_0$ by

$$E_b/N_0 = \frac{|\hat{\mu}|}{\hat{\sigma}^2},$$

(3.13)

where

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} y_i,$$

(3.14)

and

$$\hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^{N} |y_i - \hat{\mu}|^2.$$

(3.15)

This estimator can be applied to a non-coherent differential QPSK system with the use of a phase locked loop.

The quality of the constellation-based $E_b/N_0$ estimator was characterized in a simulated AWGN channel. Input samples with known $E_b/N_0$ were generated in MATLAB and presented to the $E_b/N_0$ estimator. In Figure 3.11c the estimates are plotted versus the known values to show the strong correlation between these quantities. The bias of the estimator was computed as

$$\text{Bias}\left\{\hat{\theta}\right\}_j = \frac{1}{N} \sum_{i=1}^{N} \hat{\theta}_i - \theta_i,$$

(3.16)
Figure 3.12: Simulated performance of the constellation-based $E_b/N_0$ estimator. The estimates were obtained from simulations performed with the software receiver using simulated input samples with additive Gaussian noise. Estimated values of $E_b/N_0$ plotted versus the known values. The ideal case with unit slope is shown by the solid line.

and the mean squared error (MSE) was computed as

$$\text{MSE} \left\{ \hat{\theta} \right\}_j = \frac{1}{N} \sum_{i=1}^{N} \left( \hat{\theta}_i - \theta_i \right).$$ (3.17)

In both cases $j$ indexes a bin and $i$ indexes a simulated data point. In simulation the estimator has a small negative bias of about $-0.25$ dB and a MSE of about $0.126$ dB, and both values were approximately constant across the range of $E_b/N_0$ values.

Finally, the constellation-based $E_b/N_0$ estimator was evaluated using antenna test data. As seen in Figure 3.11c, the vast majority of the data points lie within 4 dB of the theoretical ideal performance, which is significantly better than the previous two estimators and sufficiently accurate for NFMI antenna performance studies where $E_b/N_0$ changes by tens of decibels when the antennas are moved by tens of centimeters. The bias of the estimator using real test data varies linearly over the range of values, starting at about $+0.5$ dB at $5$ dB $E_b/N_0$ and increasing to about $+2.5$ dB at $13$ dB. The MSE increases from a low of about $1$ dB at $5$ dB to a high of $9$ dB at $13$ dB $E_b/N_0$. 

26
Chapter 4

Antennas

4.1 Introduction

The purpose of this chapter is to study the achievable range and bandwidth of a size and power-limited broadband near field communication system, with emphasis on how the link efficiency and bandwidth are determined by the design and properties of the transmit and receive antennas. Achieving near field communications that is both power-efficient and broadband with an electrically small antenna is challenging due to physical limitations. Factors that determine performance include transmitter modulation scheme and digital signal processing (DSP), transmit power amplifier (PA), transmit antenna, propagation environment, receive antenna, receive low noise amplifier (LNA), and the receiver DSP.

Several of these factors are relatively straightforward to deal with and can be optimized using careful engineering. These include the transmit and receive DSP algorithms and implementation. Other factors, such as the transmit and receive antenna efficiency and impedance matching bandwidth, involve pushing the design as close to fundamental physical limits as possible without violating size restrictions, which can be difficult.

Basic theorems of antenna theory limit the efficiency of antennas that are much smaller than the electromagnetic wavelength at the system operating frequency, which limits the near field strength that can be produced for a given amount of current through the antenna. These theorems also limit the impedance matching bandwidth that can be achieved, even with a lossless, ideal multistage matching network.
For small antennas, low efficiency and limited impedance matching bandwidth actually interact in such a way that compounds the difficulty of achieving power-efficient system with a low battery power supply voltage. A low-efficiency antenna requires a relatively large input current into the antenna to achieve a reasonably high near field strength that is above the environmental noise floor. High input current and power-added efficiency with a low power supply voltage dictates that the power amplifier input impedance must be low. Low power amplifier output impedance in turn reduces the impedance matching bandwidth of the amplifier-antenna system.

Furthermore, the propagation environment and external noise at 13 MHz are both highly variable, which confounds attempts to model the system and understand it theoretically using techniques that are straightforward at microwave frequencies. External noise at 13 MHz is higher by 60–100 dB than at microwave frequencies, and can vary based on location and time of day by a remarkable 40 dB or more. Impulse noise and other non-stationary, non-Gaussian noise sources are common. These effects obscure the fundamental physical behavior of the antennas in the system and make it difficult to compare link efficiency and system bandwidth for different antenna designs. To remove these effects, hundreds of measurements at different times and locations were averaged in order to produce the results in this chapter.

4.2 Hertzian Magnetic Dipole Model

The magnetic field generated by a ferrite-loaded loop antenna can be modeled as the quasi-static magnetic field of a Hertzian dipole. Thus the magnetic field $\mathbf{H}(r)$ of a ferrite loop antenna situated at the origin of a spherical coordinate system is [19]

$$
\mathbf{H}(r) = j \omega \varepsilon \frac{e^{-j kr}}{4\pi r} I_{ml} \left\{ \left[ \frac{-j}{k r} + \left( \frac{-j}{k r} \right)^2 \right] 2 \cos \theta \hat{r} + \left[ 1 - \frac{j}{k r} + \left( \frac{-j}{k r} \right)^2 \right] \sin \theta \hat{\theta} \right\}, \tag{4.1}
$$
where $\omega$ is the operating frequency, $k$ is the wave number, and $r$ and $\theta$ are components of the vector $\mathbf{r}$ in spherical coordinates. Substituting $I_{ml} = j\omega \mu I_0 A$ and simplifying gives

$$
H(\mathbf{r}) = -k^2 I_0 A e^{-jkr} \left\{ -\frac{j}{kr} + \frac{1}{k^2 r^2} \right\} 2 \cos \theta \hat{\mathbf{r}} + \left[ 1 - \frac{j}{kr} + \frac{1}{k^2 r^2} \right] \sin \theta \hat{\mathbf{\theta}} \right\}, \quad (4.2)
$$

where $I_0$ is the magnitude of the current in the loop and $A$ is the area of the loop.

Equation 4.2 also provides several useful insights that aid in NFMI antenna design and test. First, it provides information about the direction of the magnetic field lines, which is important for setting up effective antenna test configurations where the antennas are oriented parallel to the predicted direction of the field lines. Figure 4.1 shows the magnetic field lines generated by a Hertzian dipole antenna in the near field regime at 13.5 MHz. Equation 4.2 also shows that the field strength is proportional to the area of the loop $A$ and the magnitude of the current in the loop $I_0$. The area of the loop is straightforward
to modify, but is constrained by the practical need for a compact design. The current $I_0$

is influenced by the number of windings, the dimensions and magnetic properties of the

ferrite material loading the antenna, and how the antenna impedance is matched to the

corresponding amplifier. These properties affect both the efficiency and bandwidth of the

antenna.

### 4.3 Comparison of Antenna Performance

The magnetic field model of Section 4.2 aided in the design of dozens of antennas,

including air core antennas, multiply-resonant antennas, ferrite cores of various types and
diameters, and antennas with multistage impedance matching networks. Four of these de-
signs were selected for exhaustive study and comparison in this chapter. Table 4.1 gives an

overview of these four designs.

Each antenna of interest was tested rigorously in a carefully controlled manner. Test-
ing was performed using the Automated Test System, which is a hardware and software

system developed to speed rigorous antenna testing while maintaining a detailed archive of
test results and associated setup information (see Appendix B for a detailed description of

the Automated Test System). Prior to each test, the transmit and receive antennas were

mounted on a test fixture that allowed for careful control of the link distance and antenna

orientation. Four link distances were considered: 0.5 m, 0.75 m, 1.0 m, and 1.25 m. Because

the antenna performance is location and time dependent, 15-20 measurements were taken at

<table>
<thead>
<tr>
<th>Antenna</th>
<th>Core Dimensions</th>
<th>Coil(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hollow core</td>
<td>25.4 mm 12.3 cm</td>
<td>5.1 mm</td>
</tr>
<tr>
<td>Quad core</td>
<td>20 mm 2.5 mm</td>
<td>N/A</td>
</tr>
<tr>
<td>Single core</td>
<td>45 mm 3.5 mm</td>
<td>N/A</td>
</tr>
<tr>
<td>Dual tuned</td>
<td>45 mm 3.5 mm</td>
<td>N/A</td>
</tr>
</tbody>
</table>
each distance. To cover a wide range of propagation environments, the location of the setup was changed randomly.

The transmit antenna is driven by a current gain amplifier and the received signal is amplified by a high gain low noise amplifier (LNA) with 50 Ω input impedance. The high gain LNA used on the receive side is a commercially available ZFL-500LN from MiniCircuits, Inc. The high input impedance of the LNA is important for the study, since it effectively increases the bandwidth of the receive antenna and allows all of the antennas to be tested with the same antenna on the receive side with a somewhat reduced effect on overall system bandwidth from the receive side of the system.

4.3.1 Transmit Power

The power consumed by the transmitter can be divided into three categories: power drawn from the battery, power transferred to the antenna, and power converted to magnetic field by the antenna. For the tests reported here, the DC power drawn from the battery was typically between 50 and 80 mW, and did not change significantly with the impedance of the antenna load. Power transferred to the antenna, however, is a function of the efficiency and impedance of the antenna. High impedance antennas such as the hollow-core antennas accept less power from the transmit amplifier, and low impedance antennas such as the solid core antenna accept more power from the amplifier. The amount of power received by the antenna that is converted to magnetic field energy is a function of the antenna efficiency. The hollow-core ferrites are larger in diameter and thus more efficient at converting input power into magnetic field energy than the smaller solid core antennas.

The transmit signal amplitude was set such that the transmit power amplifier peak voltage at the tuned amplifier input terminals was approximately 0.7 V. The power accepted by the antenna from the power amplifier varied from roughly 80 mW for the solid core antenna to 10 mW for the hollow core antenna. Due to the differences in transmit power,
the BER and SNR results given in the following sections are biased in favor of the solid core antenna.

4.3.2 Hollow-Core Ferrite-Loaded Loop Antenna

Previous research suggests that hollow core antennas offer optimal bandwidth and efficiency for a given volume of ferrite [13]. After much study, a 10-turn hollow core ferrite loaded loop antenna design was selected for further experimentation. The antenna was tuned using a variable capacitor to the frequency 13.56 MHz. The hollow core antenna was used on the transmit side of the link, and the receive side used a solid core antenna. Figure 4.2a shows Smith chart plot of the tuned response of the hollow core antenna. The test results for this antenna can be found in Section A.1.

4.3.3 Quad-Core Ferrite-Loaded Loop Antenna

The small quad-core ferrite loaded loop antenna consisted of four small ferrite cores with windings connected in series and arranged in square topology. This design improves the range performance due to the strong inter-element coupling between the cores and a consequent increase in the effective size of the antenna relative to the total ferrite volume. Figure 4.2b shows the Smith chart plot of the tuned response of the transmit antenna, and Section A.2 gives the test results.

4.3.4 Solid Ferrite-Loaded Loop Antenna

For these results, identical solid core antennas were used on both the transmit and receive sides of the link. Figure 4.2c shows the response of the antenna, and Section A.3 shows the performance for various link distances.
(a) Hollow core antenna.  
(b) Quad-core antenna.  
(c) Solid core antenna.  
(d) Dual-tuned antenna.  

Figure 4.2: S parameter response comparison of the four antenna designs from 10 to 20 MHz.
Table 4.2: Antenna performance summary.

<table>
<thead>
<tr>
<th>Antenna</th>
<th>Efficiency</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hollow core</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Quad core</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Solid core</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Dual tuned</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

4.3.5 Dual-Tuned Ferrite-Loaded Loop Antenna

The dual-tuned ferrite loaded loop antenna is constructed from two solid core ferrite cores electrically connected in parallel. The two elements are arranged orthogonal to each other to minimize the inter-element coupling and to achieve polarization diversity. The antenna is designed to maximize bandwidth performance by independently tuning the elements at two distinct resonant frequencies. Test results for this antenna are in Section A.4, and the S-parameter response is shown in Figure 4.2d.

4.4 Conclusions

This chapter has focused on the influence of antenna geometry and design on the bandwidth, bit error rate, SNR, and power performance of a near field communication system. The primary goal was to study the effect of antenna design on the bandwidth and range of the link.

From the antenna performance comparison results, summarized in Table 4.2, it is apparent that the primary factors affecting link performance are the antenna near field link efficiency and impedance matching bandwidth. The maximum environment-averaged received signal power over frequency with the hollow core antenna as transmitter was $-65$ dBm, whereas the received power with the other three antennas was much lower (quad-core $-74$ dBm, solid core $-87$ dBm, and dual tuned $-77$ dBm). This indicates that the hollow-core antenna is more efficient as a near field radiator than the other antennas. Taking into account the lower input impedance and higher input power accepted by the solid core
antenna, the difference is even greater than these figures indicate. This effect adds roughly 9 dB to the efficiency difference between the hollow-core and solid core antennas. The increased efficiency of the hollow core antenna is due to the larger diameter of the ferrite core. Efficiency also increases significantly as the core becomes longer, so it is important to note that the high efficiency of the hollow-core antenna is achieved with a core of roughly half the length of that of the solid core antenna.

While the hollow-core antenna is more efficient than the others, its higher inductance leads to a lower impedance matching bandwidth. One of the fundamental theorems of antenna theory is that the efficiency and bandwidth of a small antenna are roughly inversely related. The solid core antenna has a very low efficiency, and consequently has a higher impedance matching bandwidth than the other single-tuned antennas. To achieve bandwidth comparable to that of the solid core antenna with increased efficiency, a dual-resonant design is required.

The ultimate measure of performance of the broadband near field communication system is bit error rate (BER) over all frequency channels. Using 1 meter separation distance as a benchmark, the hollow core, quad-core, and dual tuned antennas all achieve 2–3 channels with BER near $10^{-6}$ or lower with 10–20 mW transmit power accepted by the antenna, whereas the solid core single core antenna achieves a best BER that is poorer by two orders of magnitude than the other antennas, with a much higher accepted transmit power (approx. 80 mW). This is due to the smaller link efficiency of the solid core antenna, and consequently lower SNR at the receiver.

It can be shown theoretically that the best of these antennas (hollow core, quad-core, and dual-tuned designs) achieve close to optimal performance for the bandwidth-efficiency product based on fundamental physical bounds. Therefore, to achieve significantly better performance than the considered antennas would require changing the requirements that dictate antenna size, power supply, or other parameters. Within the performance bounds, however, it is still possible to select and adjust geometrical parameters of the antenna or
detailed aspects of the amplifier design, such as impedance, aspect ratio, coil size, ferrite length, and so forth in order to realize a small, broadband, power-efficient near field communications system that meets a given set of physically reasonable performance requirements. The presented results provide a framework for understanding these bounds and optimizing system physical and electrical properties within them.
Chapter 5

Time Division Multiple Access System

A time division multiple access (TDMA) system for a wireless body area network (BAN) based on near-field magnetic induction (NFMI) technology.

A wireless body area network is a collection of portable electronic devices worn on or near the body and the wireless communication links between those devices. When the physical layer of the body area network is NFMI, frequency bandwidth is scarce due to the narrowband characteristics of the antennas and matching networks. This limited frequency space might be divided into several channels to allow multiple body area networks to operate in close proximity without interference. But within a single body area network, multiple devices share a single channel. This scenario motivates the development of a TDMA protocol that meets the particular needs of an on-body NFMI network. In this chapter, the requirements, design, and implementation of such a protocol are presented.

5.1 Definitions

TDMA allows multiple devices to share a single frequency channel by allowing only one device to transmit at a time. A time slot is defined as the time interval during which a particular node is permitted to transmit on the channel. Consecutive time slots are assigned to different nodes in the network such that each node is permitted to transmit. Short guard periods are inserted between each time slot to prevent interference resulting from imperfect timing synchronization. In the simplest TDMA protocols, each node is assigned a time slot round-robin fashion until all nodes have been assigned a slot, after which the cycle repeats
starting with the first node. A TDMA frame for the round-robin protocol is defined as the minimum set of contiguous time slots that include all nodes in the network.

5.2 Requirements

The requirements for the TDMA protocol were set by Doug Dobyns, Executive Vice President of Product Development at FreeLinc, Inc., in collaboration with Chris Shaw, a FreeLinc R&D Engineer, and myself. Our objective was to create a TDMA system that would meet the needs of an actual product prototype in development by FreeLinc, but which would be simple enough to be built and demonstrated by two relatively inexperienced engineers in a short time frame. Our intention from the outset was not to create a finished product, but to create a starting point suitable for demonstrating capability which could be extended later.

The network topology is shown in Figure 5.1, and will include one master node and zero to three slave nodes. The master node will have bidirectional data links to each of the slave nodes, which communicate only with the master node. The data rates of each
Table 5.1: TDMA data rate requirements.

<table>
<thead>
<tr>
<th>Link</th>
<th>Minimum Data Rate Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master to Slave 1</td>
<td>50 kbps</td>
</tr>
<tr>
<td>Slave 1 to Master</td>
<td>50 kbps</td>
</tr>
<tr>
<td>Master to Slave 2</td>
<td>50 kbps</td>
</tr>
<tr>
<td>Slave 2 to Master</td>
<td>50 kbps</td>
</tr>
<tr>
<td>Master to Slave 3</td>
<td>10 kbps</td>
</tr>
<tr>
<td>Slave 3 to Master</td>
<td>1 kbps</td>
</tr>
</tbody>
</table>

link and of each direction within a given link will be statically predetermined by software parameters, and are not dynamically allocated or reallocated. The required data rates will be as defined in Table 5.1, and the raw data rate of the link without TDMA will be 250 kbps. The identity of each node will be statically determined (i.e., by DIP switch setting), and will not be reassigned during operation. It is assumed that one master node and zero or more slave nodes will be present at all times, and that duplicate nodes will not exist. The behavior of a system that violates these assumptions is undefined. The maximum latency of the system will be such that audio links have no perceptible delay, preferably less than 50 ms.

5.3 Design

The design objectives were to meet the requirements and to create an efficient, robust protocol that would be flexible enough to accommodate requirement changes and the future addition of new capabilities.

The TDMA protocol designed for the FreeLinc prototype follows a master-slave paradigm. The master initiates communication with a slave by sending a packet, which occupies one TDMA time slot, and the slave, if present, responds by sending a packet back to the master in the next time slot. The full TDMA frame is defined in Table 5.2. The frame is static, meaning that the duration and ordering of time slots in the frame does not change while the device is in operation; however, the frame can be easily re-defined to accommodate different data rate requirements and different numbers of slave nodes. Despite the static
Table 5.2: TDMA frame definition.

<table>
<thead>
<tr>
<th>Type</th>
<th>Source</th>
<th>Destination</th>
<th>Duration (bit periods)</th>
<th>Payload Size (bits)</th>
<th>Net Data Rate (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet</td>
<td>Master</td>
<td>Slave 1</td>
<td>1632</td>
<td>1600</td>
<td>50</td>
</tr>
<tr>
<td>Guard</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet</td>
<td>Slave 1</td>
<td>Master</td>
<td>1632</td>
<td>1600</td>
<td>50</td>
</tr>
<tr>
<td>Guard</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet</td>
<td>Master</td>
<td>Slave 2</td>
<td>1632</td>
<td>1600</td>
<td>50</td>
</tr>
<tr>
<td>Guard</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet</td>
<td>Slave 2</td>
<td>Master</td>
<td>1632</td>
<td>1600</td>
<td>50</td>
</tr>
<tr>
<td>Guard</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet</td>
<td>Master</td>
<td>Slave 3</td>
<td>896</td>
<td>864</td>
<td>27</td>
</tr>
<tr>
<td>Guard</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packet</td>
<td>Slave 3</td>
<td>Master</td>
<td>544</td>
<td>512</td>
<td>16</td>
</tr>
<tr>
<td>Guard</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Frame Total</strong></td>
<td></td>
<td></td>
<td><strong>8000</strong></td>
<td><strong>7776</strong></td>
<td><strong>243</strong></td>
</tr>
</tbody>
</table>

Table 5.3: TDMA packet definition.

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>header (32 bits)</td>
</tr>
<tr>
<td>payload (variable length, measured in bytes)</td>
</tr>
<tr>
<td>unused space</td>
</tr>
</tbody>
</table>

nature of the frame, the protocol is robust to slave nodes entering or leaving the network at any time. The master sends packets directed to each slave node at the appropriate time slot regardless of whether it detects a response.

During each time slot, a master or slave node sends a TDMA packet with the structure shown in Table 5.3. Each packet begins with a 32-bit header containing a preamble, source and destination device codes, and payload length in bytes. The structure of the packet header is defined in Table 5.4. The remainder of the packet is a payload which contains arbitrary data. Each node in the network examines the header information of incoming packets and only accepts packets with a correct preamble and which have a destination code.
Table 5.4: TDMA packet header definition.

<table>
<thead>
<tr>
<th>preamble (16 bits)</th>
<th>src code (4 bits)</th>
<th>dst code (4 bits)</th>
<th>payload size in bytes (8 bits)</th>
</tr>
</thead>
</table>

that matches its own device code. Packets with corrupt headers or which are addressed to other nodes are simply ignored.

Because each node in a wireless network is an independent physical device with its own clock, a TDMA protocol must provide a way for all nodes to become synchronized. In this design, the master node sets the timing of the TDMA frame, and the slaves derive their timing from packets received from the master. This gives the master complete control of the frame timing, which ensures that the time elapsed between frames is both precise and accurate, which in turn guarantees that the protocol will achieve the specified data rates with negligible deviation. When a slave receives a packet from the master, it waits for the incoming packet to end and adds an appropriate guard period as defined in the frame definition before transmitting its response packet back to the master. Thus the arrival of a packet from the master node is the event that synchronizes a slave node to the master. Slaves always listen for packets from the master, and never transmit unless they successfully detect a packet from the master addressed to them.

5.4 Implementation

The TDMA prototype system was implemented on an Altera Cyclone III DSP Development Kit and integrated with an existing 250 kbps wireless transceiver system designed and implemented previously. To provide the desired flexibility, the protocol and data management algorithms were implemented primarily in software running on a NIOS II soft-core processor programmed into the Cyclone III FPGA. Supporting dedicated hardware circuits for time-critical operations ensure good performance.
5.4.1 Device ID

The identity of a node is determined by a DIP switch on the device. This four-bit ID value becomes the source field in packet headers originating from the device. Device 0 is the master, and devices 1, 2, and 3 are slaves. Other possible device codes (4 to 15) are available for use in networks requiring up to 15 slave devices.

5.4.2 SOPC Builder System

The NIOS II is a configurable processor provided with the Altera development tools as a set of licensable intellectual property (IP) cores. NIOS II is part of a larger ecosystem of peripherals, controllers, and other hardware modules that are configured using Altera’s SOPC (System On a Programmable Chip) Builder. The TDMA system uses a basic NIOS II core, two hardware interval timers, two DMA controllers, and several other components. The complete system as seen in SOPC Builder is shown in Figure 5.2.

5.4.3 Software

The software is written in C and uses provided device drivers and the NIOS II Hardware Abstraction Layer (HAL) provided by Altera. The code consists of configuration settings, initialization code, and interrupt service routines (ISRs).

The operation of the TDMA system is highly configurable. Most of the configuration settings are stored in software header files that can be modified prior to compilation. The settings available in these files include timing parameters for fine-tuning device synchronization, constants that define number of slave devices and the lengths of packets to and from those devices, the value to use for the packet preamble, and the duration of the guard period between packets.

When initialization begins, the software reads four bits from the DIP switch to determine the device ID (see Section 5.4.1). The remaining initializations are conditional on whether the device is a master or slave device, including the particular interrupt service
Figure 5.2: Screenshot of the SOPC Builder window containing the TDMA system, showing the memory address space, IRQ assignments, and clock domains. The soft-core NIOS II processor is the top component, and the transceiver system designed in DSP Builder is the component at the bottom named “trx_Interface_0.” Other components include hardware timers, DMA controllers, flash memory, on-chip RAM, and debugging tools.
routines (ISRs) registered. After initialization is complete, all software activity occurs in response to hardware interrupt requests (IRQs), which cause corresponding ISRs to run.

### 5.4.4 Interval Timers

Accurate measurement of time is critical to the operation of the TDMA protocol. Two interval timers in the SOPC builder system provide the desired accuracy and precision. The timers are configured with an interval, which is a multiple of the clock period, and generate an interrupt to the processor each time the interval expires. A timer can be configured to run continuously or to stop at each expiration until software restarts it. The way these timers are used in the TDMA system is described in the following paragraphs.

Master nodes use one interval timer to signal the start of each new TDMA frame, and a separate interval timer to measure the interval between each packet within the frame. The TDMA frame timer is configured to run continuously, which guarantees that the interrupts generated by that timer will always occur precisely one frame interval apart. The frame timer’s operation is not influenced in any way by the behavior of the software or the TDMA network. Each time an interrupt from the frame timer is generated, the master sends a packet to the first slave node in the network to initiate the TDMA frame.

The second timer measures smaller intervals within a single TDMA frame. It is set by software in response to events in the TDMA network, such as the start of the frame, the arrival of a packet from a slave node, or an interrupt generated by itself. The master node responds to interrupts generated by this timer by sending packets to slave nodes (excluding the first slave, which occurs in response to the frame timer) and enabling the receiver packet detector when incoming packets are expected from slave nodes.

### 5.4.5 Custom FPGA Circuits

Most of the TDMA protocol management is done in software, but some supporting operations are implemented as dedicated hardware units in the FPGA. These include packet
header detection, storage of configuration parameters, and some miscellaneous control logic. Examples of these are found in the receiver’s bit sink unit, which is show in Figure 5.3. The Packet Detector block near the center of the figure is described in the next section.

5.4.5.1 Packet Detection

Each node in the TDMA network must detect arriving packets. Detected packets are identified and stored, and slave nodes start hardware timers when packets from the master node arrive. Because slave node timing must be as accurate as possible, the packet detection algorithm is implemented as a custom circuit in the FPGA. The packet detection circuit, shown in Figure 5.4, searches for a sequence of contiguous bits that match the expected values of the preamble, source, and destination fields in the packet header (see Table 5.3 for the packet header definition). When a match is detected, the entire packet is added to a FIFO where it remains until accessed by software and an interrupt is generated. Slave nodes use this interrupt to reset hardware timers as described in Section 5.4.4.

The expected values of the preamble and source fields are stored in software-accessible registers, and the expected value of the destination field is the device ID. A separate configuration register allows software to disable source or destination field checking. For example, a slave node only expects to receive packets with a source address corresponding to the master node and a destination address corresponding to itself, so checking for both the source and destination fields is appropriate. In contrast, the master node expects to receive packets from any of the three slave nodes, therefore it disables source field checking during initialization.

5.4.5.2 Transmitter On Block

Fundamental to the operation of TDMA is the requirement that nodes in the network take turns transmitting, which implies that each node must keep its transmitter turned off and only turn it on during its assigned time slot. Prior to the implementation of TDMA, the transmitter of the FreeLinc wireless system was always on. To provide on/off capability, a
The packet detector generates a pulse when it finds a packet with (1) a matching preamble, (2) a matching source code, and (3) a matching destination code. (2) and/or (3) can be disabled via the configuration register. The detector is enabled by asserting Start, and disabled by Reset (if both are asserted, Start wins).

RxOn turns on when an RxStart pulse is received, and turns off when fifo full goes high. The intent of this signal is that it can be used to enable/disable Rx signal processing, ADC, analog circuitry, etc.

RxOn turns on when an RxStart pulse is received, and turns off when fifo full goes high. The intent of this signal is that it can be used to enable/disable Rx signal processing, ADC, analog circuitry, etc.

Figure 5.3: The receiver bit sink circuit. This includes the packet detector, the FIFO where detected packets are stored, and control logic.
Figure 5.4: Packet detector circuit. This circuit generates a pulse when all of the active header field detector blocks find a match. The source and destination code detectors can be disabled using two bits of a configuration register.
“Tx On” block was constructed (Figure 5.5). This block includes control logic that turns the transmitter on when a packet is being transmitted and then shuts it off immediately when the end of the packet is reached. The accuracy of the timing is critical: If the “on” state lasts too long the node might interfere with adjacent time slots, but if too short the node will cut its own transmission short. Software-configurable registers can be used to calibrate the counters in the Tx On block to achieve the desired timing requirements.

5.4.5.3 Configuration Registers

Several hardware registers are mapped to the software address space, which allows for configuration of certain hardware settings without recompiling the FPGA hardware. These registers are described in Appendix C.

5.5 Calibration and Test

The completed TDMA system was calibrated and tested by real-time operation with three Cyclone III DSP Development Kits. Because only three development boards were available, the test networks consisted of one master node and a maximum of two slave nodes rather than three. However, the results of this testing are sufficient to demonstrate with high confidence that a full network with three slave nodes would also work.

The test configuration is designed to exercise the system in a way that closely matches the expected conditions of actual operation. The tests run in real time and transmit modulated data streams wirelessly or over coaxial cables. Packets payloads from each node are filled with repetitive data unique to each node such that receiving nodes can confirm that each received packet contains the expected data without errors. Missing packets or corrupted packets are counted and the aggregate statistics are displayed in a terminal window for debugging.

Calibration of the system is primarily concerned with adjustment of timing parameters. These include parameters that affect the start and stop time of the transmitter, the
Figure 5.5: The Tx On circuit. The input signal TxOn in the top-left corner goes high when the TDMA software initiates a packet transmission, and goes low when the packet ends. The rightmost multiplexer selects between a constant representing 0 V and the output samples of the transmitter signal processor. The output of this multiplexer is sent directly to the DAC.
Figure 5.6: Oscilloscope capture of a TDMA frame produced by the system. The green signals originate from the master device, and the cyan and magenta signals originate from two slave devices. Because only three FPGA prototyping boards were available, the third slave device was not included in this test. The signal from a third slave device would occupy the two empty regions near the edges of the window. Guard periods between the packets are too narrow to be observed at this zoom level.

The expected duration of each packet, how long a receiver should wait to receive an expected packet before giving up, and other minor adjustments. This calibration was achieved with the help of an oscilloscope, which exposes overlapping transmissions and allows for measurement of packet and guard period duration (see Figure 5.6). When these timing parameters are carefully calibrated the guard periods between packets can be shortened, which allows for longer packet payloads. This reduced overhead or increased efficiency is what allowed the achieved data rates to and from slave node 3 to exceed the required rates (compare Table 5.1 to Table 5.2).
Chapter 6

MIMO

6.1 Introduction

Near-field magnetic induction (NFMI) is short-range wireless technology for on-body networks, financial transactions, biomedical sensors, and proximity based advertising. NFMI systems typically operate in the 13.56 MHz industrial, scientific and medical (ISM) band with a wavelength of over 22 meters, resulting in antennas that are electrically small, inefficient, and narrow-band. The efficiency and bandwidth limitations of the antennas bound the achievable capacity of an NFMI system with constrained physical size and battery capacity. A possible improvement in capacity may be possible with the application of multiple-input, multiple-output (MIMO) techniques. In a sufficiently diverse channel, MIMO can increase the information-theoretic channel capacity for the same power and bandwidth [20].

To be effective, MIMO must be tailored to the diversity characteristics of the channel. In a typical far-field RF system, MIMO uses the spatial diversity produced by rich scattering environments. For near field systems operating at 13.56 MHz it is impractical to separate the antennas by a significant fraction of the wavelength, thus spatial diversity cannot be exploited effectively. However, the near-field channel has strong polarization diversity with three distinct polarizations, greater than the two polarizations available in the far field. Thus a 3-by-3 NFMI channel matrix should be nearly full-rank.

Diversity techniques have already been applied to NFMI systems to mitigate fading [4]. Selection diversity and maximal ratio combining have be applied to mitigate fading caused by changing antenna orientation, and similar diversity gains can be achieved using
space-time coding for SIMO configurations. Full MIMO implementations are also possible in the near field. In [21, 22], transmitter weights are chosen to enhance the signal strength at a particular receiver while minimizing the strength at all others in a technique similar to spatial beamforming.

MIMO improves performance by exploiting diversity to increase link reliability in a fading channel or to increase the channel capacity via multiplexing. MIMO can increase the channel capacity linearly with the number of transmit/receive antenna pairs for fixed power and bandwidth [23]. Some diversity techniques that mitigate multipath fading can also be applied to degenerate cases of MIMO, which include MISO (multiple-input single-output) where the receiver has only one antenna, and SIMO (single-input multiple-output) where the transmitter has only one antenna.

Near field MIMO is closely related to line-of-sight (LOS) MIMO. In LOS MIMO, line-of-sight paths between $N$ transmit and $N$ receive antennas result in channel matrices $\mathbf{H}$ with close to $N$ non-negligible eigenvalues [24]. The line of sight paths of the LOS channel correspond to the polarizations of the NFMI channel, where the maximum rank of the channel matrix is 3.

### 6.2 MIMO Channel Capacity

The capacity of the MIMO channel can be significantly greater than the SISO capacity for a system with constrained transmitter power. The information-theoretic Gaussian channel capacity for a SISO system is

$$ C = E [\log_2 (1 + \text{SNR})] \text{ bits/s/Hz}, \quad (6.1) $$

which grows logarithmically with SNR. The capacity of a MIMO channel may be found using the “log-det” formula. Different capacities are possible for different allocations of transmitter power, which can be optimized for a given channel matrix to maximum capacity. A simpler
instance of the log-det capacity formula results when the transmitter has no information about the channel, in which case the power is divided equally among all transmitters. For this uninformed transmitter scenario, the capacity becomes

$$C = E \left[ \log_2 \left\{ \det \left( \frac{1}{\rho N_t} \mathbf{I}_{N_r} + \frac{\rho}{N_t} \mathbf{H} \mathbf{H}^\dagger \right) \right\} \right] \text{ bits/s/Hz,} \quad (6.2)$$

where $\dagger$ denotes the conjugate transpose, $\rho$ is the SNR at the receiver, $N_r$ is the number of transmitters and $N_r$ is the number of receivers [23]. The channel matrix $\mathbf{H}$ is an $N_r \times N_t$ matrix where each element $H_{i,j}$ relates the input voltage at transmit antenna $j$ to the induced voltage at receive antenna $i$. If $\mathbf{H}$ is full-rank, the capacity grows linearly with its smaller dimension. For a full-rank 3-by-3 NFMI channel, this implies a MIMO capacity of up to three times the SISO capacity—a tremendous benefit. In this chapter, a 3-by-3 NFMI channel is simulated to compute the singular values and capacity to assess the feasibility of implementing MIMO for a near-field system.

### 6.3 Simulated Channel Capacity

The NFMI MIMO channel was simulated for two sets of three mutually-orthogonal antennas, one transmitter and one receiver, separated by a distance $r$. For each antenna, the ferrites are designated by the letters $A$, $B$, and $C$, where ferrite $A$ is used as the reference for specifying orientation. The origin of a spherical coordinate system is defined as the position and orientation of the transmitter, such that the $z$ axis is parallel to transmitter ferrite $A$. The receiver antenna may be located at any arbitrary position and orientation in this coordinate system.

The magnetic field $\mathbf{H} (r)$ generated by the transmitting loop antenna situated at the origin of the spherical coordinate system can be modeled as the quasi-static magnetic field
of a Hertzian dipole by

\[
H(r) = j \omega \frac{e^{-jkr}}{4\pi r} I_{ml} \left\{ \left[ -\frac{j}{kr} + \left( \frac{-j}{kr} \right)^2 \right] 2 \cos \theta \hat{r} + \left[ 1 - \frac{j}{kr} + \left( \frac{-j}{kr} \right)^2 \right] \sin \theta \hat{\theta} \right\},
\]

(6.3)

where \( \omega \) is the operating frequency, \( k \) is the wave number, and \( r \) and \( \theta \) are components of the vector \( r \) in spherical coordinates [19]. Substituting \( I_{ml} = j \omega \mu I_0 A \) and simplifying gives

\[
H(r) = -k^2 I_0 A \frac{e^{-jkr}}{4\pi r} \left\{ \left[ -\frac{j}{kr} + \frac{1}{k^2 r^2} \right] 2 \cos \theta \hat{r} + \left[ 1 - \frac{j}{kr} + \frac{1}{k^2 r^2} \right] \sin \theta \hat{\theta} \right\},
\]

(6.4)

where \( I_0 \) is the magnitude of the current in the loop and \( A \) is the area of the loop.

The current \( I_0 \) in the transmitter antenna loop is chosen such that the total power dissipated in the antenna is equal to \( P_t = I_0^2 R_{\text{ant}} \), where \( R_{\text{ant}} \) is the real part of the antenna impedance. The antennas are taken to be ferrite-loaded loop antennas with the properties listed in Table 6.1.

**Table 6.1: Antenna properties.**

<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ferrite length</td>
<td>0.045 m</td>
</tr>
<tr>
<td>Ferrite radius</td>
<td>0.002 m</td>
</tr>
<tr>
<td>Ferrite relative permeability</td>
<td>( 120 + j2.59 )</td>
</tr>
<tr>
<td>Number of turns</td>
<td>6</td>
</tr>
<tr>
<td>Wire diameter</td>
<td>( 2.55 \times 10^{-4} ) m</td>
</tr>
<tr>
<td>Wire resistivity (copper)</td>
<td>( 1.68 \times 10^{-8} ) ( \Omega ) m</td>
</tr>
</tbody>
</table>

The resistance of a matched ferrite-loaded antenna is \( R_{\text{ant}} = R_{\text{ferrite}} + R_{\text{loss}} + R_{\text{rad}} \). The three terms in this formula are found using the approximations given in [25]. The ferrite resistance is

\[
R_{\text{ferrite}} = 2\pi f \left( \mu_{\text{rod}}/\mu' \right)^2 F_R \mu'' \mu_0 n^2 A/l_c,
\]

(6.5)

where \( \mu' \) and \( \mu'' \) are the real and imaginary parts of the relative permeability of the ferrite material, respectively, \( l_c \) is the length of the coil, and \( A = \pi a^2 \) is the cross sectional area of
the ferrite rod. The factor $F_R$ is given by the fourth-order polynomial approximation

$$F_R \approx 0.4371 \left(\frac{l_c}{l_r}\right)^4 - 0.7556 \left(\frac{l_c}{l_r}\right)^3 - 0.1384 \left(\frac{l_c}{l_r}\right)^2 + 1.0210 \left(\frac{l_c}{l_r}\right) + 0.0169. \quad (6.6)$$

Next, the resistance of the copper wire, $R_{\text{loss}}$, is

$$R_{\text{loss}} = 2\pi \rho a n / S, \quad (6.7)$$

where $S$ is the cross sectional area where current flows, given by

$$S = \begin{cases} \pi \left(\frac{d}{2}\right)^2 - \pi \left(\frac{d}{2} - \delta\right)^2 & \frac{d}{2} > \delta, \\ \pi \left(\frac{d}{2}\right)^2 & \frac{d}{2} < \delta, \end{cases} \quad (6.8)$$

where the skin depth is $\delta = \sqrt{2/2\pi f \mu_0 \sigma}$. Finally, the radiation resistance is

$$R_{\text{rad}} = \frac{Z_m}{6\pi} \beta_m^4 \left(\mu_{\text{rod}} F_V n \pi a^2\right)^2, \quad (6.9)$$

where $a$ is the radius of the loop,

$$\mu_{\text{rod}} = \frac{\mu'}{1 + 0.37 \left(l_r/d\right)^{-1.44} \left(\mu' - 1\right)} \quad (6.10)$$

is the effective permeability, $Z_m$ is the characteristic impedance of the medium, $\beta_m$ is the wave number of the medium, and $F_V$ is a third-order polynomial approximation factor given by

$$F_V \approx -0.103 \left(\frac{l_c}{l_r}\right)^3 - 0.0332 \left(\frac{l_c}{l_r}\right)^2 - 0.074 \left(\frac{l_c}{l_r}\right) + 0.9987. \quad (6.11)$$

The open-circuit voltage at the terminals of the receiving antenna is

$$v_{oc} = \mu_{\text{rod}} \mu_0 \omega A n \mathbf{H}(r_2) \cdot \mathbf{p}_2, \quad (6.12)$$
where \( \cdot \) denotes the inner product and \( r_2 \) and \( p_2 \) are the position and orientation of the receiver antenna, respectively. Finally, the voltage produced in the receive antenna load is

\[
v_l = v_{oc} \frac{Z_l}{Z_l + Z_{ant}},
\]

where \( Z_l \) is the impedance of the load and \( Z_{ant} \) is the impedance of the matched antenna.

### 6.3.1 Fixed Transmit Power Simulation

Equations (6.3–6.13) were evaluated using MATLAB for each of the nine antenna pairs, corresponding to each of the \( H_{i,j} \) elements of the channel matrix \( H \). The voltage \( v_l \) in the receive antenna load for antenna pair \((i, j)\) is assigned to \( H_{i,j} \). This process was repeated for a range of distances from 0.1 to 1.5 meters with the same antenna orientation.

The singular values of the resulting channel matrices are shown in Figure 6.1, and the uninformed transmitter MIMO capacity evaluated using equation (6.2) is shown in Figure 6.2. The transmit power was chosen such that the SISO SNR at 1 meter separation distance was near 15 dB. As expected, the capacity of the MIMO system approaches three times the SISO capacity when the SNR is high. Unexpectedly, one of the three singular values is stronger than the other two for distances less than up to about eight meters. This results from the relative orientation of the transmit and receive antenna pairs. One ferrite in both the transmitter and receiver is oriented parallel to the axis of separation, such that the ferrites are in aligned coaxially rather than in parallel. The coaxial alignment results in stronger coupling than parallel alignment at a given distance, which produces the observed disparity in singular values.

### 6.3.2 Fixed SNR Simulation

The singular values and capacity can be presented in another form that constrains SNR rather than total transmit power. This is accomplished by normalizing the channel
matrix such that the SNR is a parameter in the capacity formula. The normalized channel matrix is

$$H_n = \frac{\sqrt{N_r}}{\|H\|_{\text{Frobenius}}} H,$$

(6.14)

where $\|\cdot\|_{\text{Frobenius}}$ is the Frobenius matrix norm [26]. The unknown channel capacity formula becomes

$$C = \log_2 \det \left[ I + \text{SNR}_r H_n H_n^\dagger \right],$$

(6.15)

where $\dagger$ is the conjugate transpose. The simulated results of the NFMI channel using this normalized approach with an SNR of 15 dB from 0.01 to 10 meters are shown in Figures 6.3 and 6.4.

It is clear from Figure 6.3 that the largest singular value produced by the coaxially aligned antennas in the near field decays to zero in the far field, leaving the two remaining singular values produced by the parallel ferrites. The maximum capacity occurs when all three singular values have equal magnitude, which occurs at roughly 7 meters in distance.
Figure 6.2: Simulated NFMI-MIMO capacity versus distance for the unknown channel with fixed transmitter power. The SISO capacity for a system with the same transmit power is shown for comparison.

This is in the transition zone between the near and far field regimes, and beyond the range of most NFMI systems.

6.4 Conclusions

The simulated 3-by-3 NFMI channel has high rank and three non-negligible singular values, supporting the hypothesis that MIMO can provide a significant boost in effective bandwidth. The particular orientation of the antennas appears to be an important factor, suggesting that the capacity of an uninformed transmitter system will vary between 9 and 11 bits/s/Hz due. Despite the unexpected discovery of orientation-dependent and unequal singular values, the capacity gain of the simulated MIMO channel is significant compared to the SISO channel. Further investigation of MIMO for near field systems is fully justified by these results.
Figure 6.3: Simulated NFMI-MIMO channel matrix singular values (normalized).

Figure 6.4: Simulated NFMI-MIMO capacity versus distance for the unknown channel normalized to 15 dB SNR. The SISO capacity at 15 dB SNR is shown for comparison.
Chapter 7

Conclusions

This work makes several contributions that address current needs of NFMI systems. First, a multi-channel, discrete-time transceiver was designed and implemented on software and FPGA platforms to support research and development efforts. An automated antenna testing system was built on top of the communication time system to facilitate rapid and rigorous antenna testing, including the estimation of signal power, SNR, and bit error rate.

These systems, along with a quasistatic near-field model, were used to design, test, and evaluate dozens of unique antennas. Thousands of antenna tests in a variety of environments provide sufficient data for rigorous statistical analysis. The performance results of the best designs in terms of efficiency and bandwidth were presented in this thesis, which included solid core, hollow core, dual-tuned and quad-core designs.

This work also contributes an efficient TDMA implementation tailored to the unique needs of an on-body NFMI network. Tight timing synchronization is achieved using a combination of dedicated hardware and software interrupts to minimize overhead. A simple, reconfigurable protocol provides device identification, robustness to loss of slave nodes, and variable length packet structure.

Finally, preliminary work in the area of NFMI MIMO was performed. Simulation of the 3-by-3 NFMI channel was performed using a quasistatic field model to evaluate the theoretical capacity of NFMI MIMO for a typical system. Initial results indicate that MIMO may provide a substantial gain in effective bandwidth for NFMI systems.
7.1 Future Work

The contributions of this work have furthered the NFMI body of knowledge, but many problems and opportunities remain. Further work focused on matching networks and NFMI-MIMO seem particularly promising.

7.1.1 Matching Networks

Many antenna designs were explored as part of this work, but the bandwidth-efficiency performance of the antenna is heavily influenced by the matching network. Further work to study traditional reactive matching networks and non-Foster matching networks will complement the antenna research presented in this thesis. Non-Foster matching seems especially promising, because an inductive antenna matched by an active circuit with negative inductance could result in a very broadband front-end. Demonstration of a working negative inductor circuit and evaluation of its power/bandwidth performance will be highly valuable to the community.

7.1.2 Channel Sounding

Time did not permit extensive measurement of the 3-by-3 NFMI channel for this thesis. Experience with NFMI antenna testing suggests that a large set of experimental measurements is necessary to produce high-quality statistical results. Measurements should also be taken in different environments—including near the human body—to provide a more comprehensive picture.

7.1.3 MIMO Implementations

After good channel sounding results are available, a full NFMI MIMO implementation should be attempted. The choice of algorithm will be informed by the channel sounding data, and the platform for implementation should follow practical considerations. The easiest approach may be an uninformed transmitter scenario such as the V-BLAST algorithm [27], in
which case the algorithms may be applied in MATLAB to sampled data collected previously. Research should attempt to measure the capacity gain versus increased power, complexity, and size that will accompany a MIMO system.
References


Appendix A

Antenna Test Results

This appendix contains the antenna test results for the four antennas described in Chapter 4. These results were obtained using the Automated Test System, which is described in Appendix B. Each figure contains information from dozens of individual tests taken at a variety of locations. The central curve is the mean of all data points at a given frequency, and 70 percent of the data points at that frequency lie within the asymmetric error bars. Each antenna was tested at four distances: 0.5 meters, 0.75 meters, 1.0 meters, and 1.25 meters.

A.1 Hollow-Core Ferrite-Loaded Loop Antenna

The figures on the following four pages show the test results for the hollow-core ferrite-loaded loop antenna.
Figure A.1: Hollow-core antenna performance at 0.5 meters.
Figure A.2: Hollow-core antenna performance at 0.75 meters.
Figure A.3: Hollow-core antenna performance at 1.0 meters.
Figure A.4: Hollow-core antenna performance at 1.25 meters.
A.2 Quad-Core Ferrite-Loaded Loop Antenna

The next four pages of figures are the test results for the quad-core ferrite-loaded loop antenna.
Figure A.5: Quad-core antenna performance at 0.5 meters.
Figure A.6: Quad-core antenna performance at 0.75 meters.
Figure A.7: Quad-core antenna performance at 1.0 meters.
Figure A.8: Quad-core antenna performance at 1.25 meters.
A.3 Solid-Core Ferrite-Loaded Loop Antenna

This section includes the test results for the solid-core ferrite-loaded loop antenna.
Figure A.9: Solid-core antenna performance at 0.5 meters.
Figure A.10: Solid-core antenna performance at 0.75 meters.
(a) Bit error rate versus frequency.

(b) $E_b/N_0$ versus frequency.

(c) Power at LNA input versus frequency.

Figure A.11: Solid-core antenna performance at 1.0 meters.
Figure A.12: Solid-core antenna performance at 1.25 meters.
A.4 Dual-Tuned Ferrite-Loaded Loop Antenna

The test results for the dual-tuned ferrite-loaded loop antenna are on the pages that follow. Two sets of results were collected for this antenna because the two ferrites are oriented orthogonally with respect to each other and are each tuned to different resonant frequencies. For the first four figures, the antenna was oriented such that the 12.72 MHz coil was parallel to the receiving antenna, and the receiving antenna was tuned to 12.72 MHz. For the second set of figures the receiver antenna was tuned to 13.56 MHz, and the dual-tuned transmitting antenna was oriented with the 13.56 MHz coil parallel to the receiving antenna.
Figure A.13: Dual-tuned antenna performance at 0.5 meters in the 12.72 MHz orientation.
Figure A.14: Dual-tuned antenna performance at 0.75 meters in the 12.72 MHz orientation.
Figure A.15: Dual-tuned antenna performance at 1.0 meters in the 12.72 MHz orientation.
Figure A.16: Dual-tuned antenna performance at 1.25 meters in the 12.72 MHz orientation.
(a) Bit error rate versus frequency.

(b) $E_b/N_0$ versus frequency.

(c) Power at LNA input versus frequency.

Figure A.17: Dual-tuned antenna performance at 0.5 meters in the 13.56 MHz orientation.
Figure A.18: Dual-tuned antenna performance at 0.75 meters in the 13.56 MHz orientation.
Figure A.19: Dual-tuned antenna performance at 1.0 meters in the 13.56 MHz orientation.
Figure A.20: Dual-tuned antenna performance at 1.25 meters in the 13.56 MHz orientation.
Appendix B

Automated Test System

The Automated Test System (ATS) is an integrated hardware/software system that facilitates rapid testing of the antennas and amplifiers. The ATS transmits a modulated signal on each of the 13 frequency channels in succession and estimates signal and noise power levels, bit error rates, signal-to-noise ratios, and other statistics for each channel. This information is automatically indexed and archived on the project for later analysis.

B.1 ATS Components

The ATS is not a stand-alone program but rather an integrated system of several hardware and software components. The ATS includes the Xilinx System Generator transmitter, Xilinx Xtreme DSP Development Kit, transmitter software, receiver software, Adlink ADC, the Hubble and Galaxy PCs, MATLAB, and the project server space hosted on the CAEDM servers. The roles of these components in the ATS are described here.

The bulk of the ATS code is contained within a special task in the receiver software project, and it is this code that integrates and drives the rest of the ATS components. It is responsible for prompting the user for test parameters, establishing a network connection with the transmitter software on Galaxy, sending transmitter frequency and amplitude requests to the Tx software, initializing and running the Rx signal processing code, and storing the test results on the project server. The portion of the ATS in the Rx software also performs extensive error checking to prevent corruption of data sets and to warn users of unexpected behavior.

The Tx software, Xilinx System Generator Tx, Xtreme DSP Development Kit, and Galaxy form the transmitter portion of the ATS. The Tx software listens for and accepts incoming TCP/IP connections from the Tx software on Hubble, initializes the transmitter hardware, and changes the center frequency and amplitude of the transmitted signal. All Tx software operations are initiated and controlled by the Rx software on Hubble via a TCP/IP socket.
connection. The Xtreme DSP Development Kit is a PCI card with a Virtex IV FPGA [check], PCI interface hardware, DACs and ADCs. The Tx signal processing is all performed on the Virtex IV on this card. The FPGA implementation of the Tx signal processing was performed using Xilinx System Generator, which is a set of Simulink blocksets provided by Xilinx that allow generation of HDL code using the Simulink environment.

B.2 Tests Performed

After gathering test parameters from the user, the ATS follows this procedure to test the system:

1. Estimate noise floor power for all 13 frequency channels with the transmitter turned off.
2. Turn on transmitter with PN9 bit sequence as input starting at the 12.00 MHz channel.
3. Sample and demodulate the 12.00 MHz signal with the receiver.
4. Estimate signal power, $E_b/N_0$, bit error rate, carrier phase drift, and ... for the 12.00 MHz channel.
5. Repeat 2–4 for the remaining 12 frequency channels.
6. Archive test results on the project server.

The number of symbols transmitted at each frequency channel depends on the desired resolution of the bit error rate estimator. For example, to estimate bit error rates down to $10^{-6}$ with at least 5 bit errors per test on average, the system must transmit at least $5 \times 10^6$ bits. There is a direct relationship between the desired resolution of the bit error rate estimator and the run time of the test; thus an order of magnitude improvement in resolution requires an order of magnitude increase in run time.

For the purposes of the FreeLinc research project I chose to use two fixed test lengths in the ATS, dubbed “short” and “long.” The short test runs in seconds but can only estimate bit error rates down to $10^{-4}$, which is appropriate for quick tests where accurate estimation of small bit error rates is not critical. When smaller bit error rate resolution is required, the long test should be chosen. The long test runs for about three minutes and is accurate down to bit error rates of $10^{-6}$, which is the smallest bit error rate of interest to FreeLinc. The ATS assumes that users of the test results will interpret bit error rates smaller than
the test resolution appropriately. For example, if the estimated bit error rate for a particular test is $5 \times 10^{-7}$ and the test resolution is $10^{-6}$, all that should be assumed is that the actual bit error rate is less than or equal to $10^{-6}$.

### B.3 Data Set Contents

Each test performed by the ATS generates a data set that is stored on the project server as a MATLAB .mat file. Each data set contains the following elements:

- **Per-channel vectors**: Each of the following vectors has 13 elements, one per frequency channel:
  
  - bitsReceived: Total number of bits received on this channel during the test
  - ber: Bit error rate estimates
  - snr: $E_b/N_0$ estimates in dB
  - pow: Signal power estimates in dBm
  - noiseFloor: Noise floor power estimates in dBm
  - phaseDrift: Carrier phase drift estimates in radians per symbol

- **User-entered parameters**
  
  - distance: Link distance in meters
  - txAntName: Tx antenna name
  - rxAntName: Rx antenna name
  - txAmpName: Tx amplifier name
  - rxAmpName: Rx amplifier name
  - longTest: Boolean value: true for long tests, false for short tests
  - txGain: Gain value requested for the transmitter

- **Other values**
  
  - anomalous: Boolean value set to false when test is performed. If the test is later found to be anomalous, this is set to true.
  - timestamp: Date and time of test
  - filename: Filename of the .mat file
  - snrMethod: Indicates which SNR estimator was used. The SNR estimator has been replaced several times during the lifetime of the ATS.
B.4 Data Set Index

The data set archive also includes a global index of all tests performed. When the ATS adds a new data set to the archive, it also adds a new entry to this index. The index is in the form of a MATLAB cell array, where each row in the array represents a data set and each column stores a significant test parameter. The columns include filename, timestamp, antenna and amplifier names, link distance, test duration, and other concise information. This information can be used to select a subset of data sets to analyze together. For example, it might be desirable to analyze the set of tests with a link distance of 1 meter and with a particular set of antennas and amplifiers. The custom function createSubindex described in Section B.5 provides this filtering capability.

B.5 MATLAB Utilities

Several custom MATLAB functions facilitate easy analysis of the archived data sets. Two of these functions, plotTestSet and multiPlot, plot test results in a MATLAB figure window. plotTestSet plots the bit error rate, $E_b/N_0$, signal power, and noise power data from a single data set versus frequency on three separate plots. An example of the output from plotTestSet is given in Figure B.1. multiPlot also produces three plots, but includes data from multiple data sets. To display information in a visually meaningful manner, multiPlot generates a mean curve and error bars for each frequency channel. Multiple examples of this can be found in Appendix A.

createSubindex filters the contents of the global test index to create a subindex. It takes one or more pairs of arguments, where the first member of the pair is a column number $c_i$ and the second member is a match value $v_i$. createSubindex generates a cell array that includes all rows of the global index where the value stored in column $c_1$ is equal to $v_1$ and the value stored in column $c_2$ is equal to $v_2$, etc. The resulting cell array can be used as an argument to multiPlot, which uses the data sets found in the array to generate its plots.

B.6 Automatic Update

After the initial deployment of the ATS, it became clear that a more robust system was needed to accommodate frequent software updates. For performance reasons the software executables need to be stored on the local hard drives of the project PCs, but for backup and development the source code is best kept on the project server. To meet these needs,
Figure B.1: Example of a figure window generated by the plotTestSet MATLAB function. The plots show results from single test performed using the ATS.
an automatic update system was designed and implemented. This system allows the current master copy of the software to reside on the project server at all times, but also ensures that the local copies are kept up to date. Whenever the ATS is launched, the updater checks the timestamp of the master copy on the project server and compares it to the timestamp of the local copy. If the master copy is more recent, it is downloaded from the server to replace the local copy prior to execution. No user intervention is necessary to check for or install updates. Developers provide an update by replacing the old master copy with a newer version, and can optionally archive the old version in a separate folder.

To make this updating system possible, extra functionality was added to the Automated Test Launcher (ATL) software. The ATL runs on both Galaxy and Hubble, but its behavior on each machine is distinct. On Galaxy, the ATL is a background process that starts when a user logs on and listens for TCP/IP connections from the ATS software on Hubble. When a connection is established, the ATL checks for a software update, downloads the update if it exists, and then launches the transmitter software. On Hubble, the ATL runs only when launched by the user and is not a persistent background process. The update check occurs on each launch, after which the ATL runs the local copy of the receiver software.

### B.7 Wiki Usage Guide

A comprehensive usage guide for the ATS is available on the FreeLine project wiki at http://csas.ee.byu.edu/dokuwiki/doku.php?id=freeline:automated_test_system. The guide includes instructions for first-time setup, performing tests, and viewing test results in MATLAB. The guide also includes information on new features added since the initial release, bug fixes, feature requests, and known issues.
Appendix C

TDMA Configuration Registers

The following paragraphs describe the configuration registers available to software. Each register is readable and writable and mapped to an address in the software address space.

**Config**  This is a 32-bit register where one bit is used to disable source code matching in the packet header detector, and another bit is used to disable destination code matching. The other 30 bits are unused.

**Preamble**  This 16-bit register stores the preamble value used by the packet detector. The receiver will only accept packets that start with this value.

**Rx Timeout**  This 16-bit register stores the length of the Rx timeout period in units of bit periods (where one bit period is 1/250 kHz = 4 μs). Setting this register to 0 will result in an infinite timeout period. The timeout timer in the Rx is reset each time the Rx Start signal is sent by software. When the timer reaches zero, it causes the Rx to stop waiting for a matching packet header to appear if it has not already found one.

**Source Code**  This 4-bit register stores the packet source code. When the source code match inhibit bit of the config register equals zero, the Rx will only accept packets that have a source code matching the contents of this register.

**Tx On Start Delay**  This 16-bit register stores the number of clock cycles to delay the rising edge of the TxOn signal. When the software sends the Tx Start signal, the TxOn signal goes high this many clock cycles later. This register is used to fine-tune the timing of the TxOn signal to account for delays in the hardware signal processing blocks and elsewhere in the system.

**Tx On Stop Delay**  This 16-bit register stores the number of clock cycles to delay the falling edge of the TxOn signal. When the Tx FIFO becomes empty, the TxOn signal goes.

96
low this many clock cycles later. This register is used to fine-tune the timing of the TxOn signal to account for delays in the hardware signal processing blocks and elsewhere in the system.

**PN9 Correlator FIFO IRQ Threshold**  This 8-bit register stores sets the threshold for the CorrOut FIFO IRQ. When the number of items in the CorrOut FIFO is greater than or equal to this value, its IRQ is asserted. The only way to clear the IRQ is to read from the FIFO until the number of items is less than this threshold. To disable the IRQ, set this register to 0 (default).
Appendix D

Quasistatic Field Model Source Code

The following code listings comprise the MATLAB source code for the ferrite-loaded loop antenna quasistatic magnetic field model.

Listing D.1: driver.m

```
addpath('J:\research\code\NFMModel\model\');

%% parameters

% load ferrite parameters
[a1 d_wire1 rho_wire1 n1 lr1 mu_real1 mu_imag1] = loadFerriteParameters('fl_500079_01');
[a2 d_wire2 rho_wire2 n2 lr2 mu_real2 mu_imag2] = loadFerriteParameters('fl_500079_01');

f = 13.5e6;  % center frequency

v_g1 = 1;  % Thevenin peak voltage amplitude of source to antenna 1
Z_g1 = 1;  % Thevenin impedance of source to antenna 1
Z_L2 = 50;  % input impedance of receiver amplifier (antenna load)

R_ant1 = R_ant(mu_real1, mu_imag1, lr1, n1, a1, d_wire1, rho_wire1, f);
L_ant1 = L_ant(mu_real1, n1*d_wire1, lr1, n1, a1);
C_1 = matchingC(L_ant1, f);  % capacitance of matching capacitor in antenna 1
R_ant2 = R_ant(mu_real2, mu_imag2, lr2, n2, a2, d_wire2, rho_wire2, f);
L_ant2 = L_ant(mu_real2, n2*d_wire2, lr2, n2, a2);
C_2 = matchingC(L_ant2, f);  % capacitance of matching capacitor in antenna 2

r1 = [0; 0; 0];  % position of antenna 1
p1 = [0; 0; 1];  % orientation of antenna 1
r2 = [1; 0; 0];  % position of antenna 2
p2 = [0; 0; 1];  % orientation of antenna 2

%% function calls
I_1 = I_0(v_g1, Z_g1, R_ant1, L_ant1, C_1, f);
mu_rod2 = mu_rod(mu_real2, lr2, a2);
H1 = H_field(I_1, a1, lr1, n1, mu_real1, f, 'Cartesian');
v_oc2 = v_oc(mu_rod2, f, a2, n2, 1, H1, r1, p1, r2, p2);

v_L2 = v_L(v_oc2, R_ant2, L_ant2, C_2, f, Z_L2);
```

98
Listing D.2: loadFerriteParameters.m

function [a d_wire rho_wire n lr mu_real mu_imag] = loadFerriteParameters(filename)

    load(["J:\research\code\NFMIAModel\model\parameters"]; filename);
end

Listing D.3: R_ant.m

% R_ant Resistance of a ferrite-loaded loop antenna.
% R = R_ant(mu_real, mu_imag, lr, n, a, d_wire, rho_wire, f) gives the
% resistance of a ferrite-loaded loop antenna in ohms where
% mu_real, mu_imag are the permeability of the ferrite material
% lr is the length of the ferrite rod, in meters
% n is the number of turns
% a is the radius of the ferrite rod in meters
% d_wire is the diameter of the wire in meters
% rho_wire is the resistivity of the wire in Ohm-meters
% f is frequency in Hertz

d = d_wire/2; % assumes no gaps between wraps

R_1 = R_loss(rho_wire, f, d_wire, a, n);
R_r = R_rad(mu_real, lc, lr, n, a, f);
R_f = R_fer(mu_real, mu_imag, lc, lr, n, a, f);

R_ant = R_1 + R_r + R_f;
end

Listing D.4: R_fer.m

function R_fer = R_fer(mu_real, mu_imag, lc, lr, n, a, f)

mu_0 = 4*pi*1e-7;
mu_r = mu_rod(mu_real, lr, a);
F_R = 0.4371*(lc/lr)^4 - 0.7556*(lc/lr)^3 - 0.1384*(lc/lr)^2 + 1.0210*(lc/lr) + 0.0169;

R_fer = 2*pi*f*(mu_r/mu_real)^2*F_R*mu_imag*mu_0*n^2*pi*a^2/lc;
end

Listing D.5: R_loss.m

function R_loss = R_loss(rho, f, d_wire, a, n)

mu_0 = 4*pi*1e-7;
sigma = 1/rho; % conductivity

skinDepth = sqrt(2/(2*pi*f*mu_0*sigma));
S = pi*(d_wire/2)^2 - pi*(d_wire/2 - skinDepth)^2;
\[ R_{\text{loss}} = \rho \ast 2 \pi \ast a \ast n / S; \]

end

---

**Listing D.6: R_rad.m**

```matlab
function R_rad = R_rad(mu_real, lc, lr, n, a, f)

    c = 299792458; % speed of light in vacuum
    mu_0 = 4*pi*1e-7; % vacuum permeability
    Z_m = sqrt(mu_0^2*c^2); % characteristic impedance of free space
    beta_m = 2*pi*f/c; % wavenumber
    mu_r = mu_real(mu_real, lr, a);
    F_V = -0.103*(1c/lr)^3 - 0.0332*(1c/lr)^2 - 0.074*(1c/lr) + 0.9987;

    R_rad = Z_m/(6*pi)*beta_m^2*(mu_r*F_V*n*pi*a^2)^2;

end
```

---

**Listing D.7: L_ant.m**

```matlab
function L = L_ant(mu_real, lc, lr, n, a)

    mu_0 = 4*pi*1e-7;
    F_L = 0.3419*(1c/lr)^3 - 1.3636*(1c/lr)^2 + 1.6882*(1c/lr) + 0.0501;
    L = mu_real(mu_real, lr, a)*F_L*mu_0*n^2*pi*a^2/lc;

end
```

---

**Listing D.8: matchingC.m**

```matlab
function C = matchingC(L, f)

    omega = 2*pi*f;
    C = 1/(omega^2*L);

end
```

---

**Listing D.9: L_ant.m**

```matlab
function L = L_ant(mu_real, lc, lr, n, a)

    % L = L_ant(mu_real, lc, lr, n, a) gives the inductance of a
```

---

100
% ferrite-loaded loop antenna in henries where
% mu_real is the real part of the permeability of the ferrite material
% lc is the length of the coils (turns * wire diameter) in meters
% lr is the length of the ferrite rod in meters
% n is the number of turns
% a is the radius of the ferrite rod in meters

function L = L_ant(mu_real, lc, lr, n, a)
    mu_0 = 4*pi*1e-7;
    F_L = 0.3419*(lc/lr)^3 - 1.3636*(lc/lr)^2 + 1.6882*(lc/lr) + 0.0501;
    L = mu_rod(mu_real,lr,a)*F_L*mu_0*n^2*pi*a^2/lc;
end

Listing D.10: I_0.m

function I_0 = I_0(v_g, Z_g, Z_L)
    I_0 = v_g / (Z_g + Z_L);
end

Listing D.11: mu_rod.m

% Source: paper by Monte

function mu_rod = mu_rod(mu_prime, lr, a)
    d = 2*a;
    if (lr/d < 2 || lr/d > 20)
        mu_rod = NaN;
        error('Function only valid for 2 <= lr/(2*a) <= 20. ');
    end
    X = 0.37*(lr/d)^-1.44;
    mu_rod = mu_prime./(1 + X*(mu_prime - 1));
end

Listing D.12: H_field.m

function H = H_field(I_0, a1, lr1, n1, mu_real1, f, varargin)

    % derived parameters
    lambda = 3e8/f;    % wavelength (m)
    k = 2*pi/lambda;   % wavenumber (m^-1)

    % H field
    syms r theta phi;

101
syms x y z;

coeff = -k^2*I_0*pi*a1^2*exp(-1j*k*r)/(4*pi*r)*mu_rod(mu_real1,lr1,a1)*n1;

% H in Cartesian coordinates (default)
H = coeff * [(1 - 3j/(k*r) - 3/(k^2*r^2))*z/r^2*x; ...
(1 - 3j/(k*r) - 3/(k^2*r^2))*z/r^2*y; ...
2*(-1j/(k*r) - 1/(k^2*r^2))*z^2/r^2 - (1 - 1j/(k*r) - 1/(k^2*r^2))*(x^2 + y^2)/r^2];

H = subs(H, r, sqrt(x^2 + y^2 + z^2));

if (length(varargin) == 1)
    if (strcmpi(varargin{1}, 'spherical'))
        % H in spherical coordinates
        H = coeff * [(-1j/(k*r) + 1/(k^2*r^2))*2*cos(theta); ...
(1 - 1j/(k*r) - 1/(k^2*r^2))*sin(theta); 0];
    elseif (~strcmpi(varargin{1}, 'Cartesian'))
        error('Coordinate system argument must be ''spherical'' or ''Cartesian''.');
    end
elseif (length(varargin) > 1)
    error('Too many arguments.');
end
end

---

Listing D.13: v_oc.m

% Finds the open-circuit voltage of a ferrite-loaded coil antenna (2) in the presence of a magnetic field generated by a transmitting antenna (1), taking into account the positions and orientations of the coils.

function v_oc = v_oc(mu_rod, f, a, n, F_A, H1, r1, p1, r2, p2)

mu_0 = 4*pi*1e-7; % vacuum permeability
omega = 2*pi*f; % angular frequency
A = pi*a^2; % area of coil 2

coeff = mu_rod*mu_0*omega*A*n*F_A;

[r2_prime p2_prime] = trans_rotate(r1,p1,r2,p2);

syms x y z;
H1 = subs(H1, {x, y, z}, {r2_prime(1), r2_prime(2), r2_prime(3)});

v_oc = coeff*dot(H1,p2_prime);
end
Listing D.14: v_L.m

function v_L = v_L(v_oc, Z_S, Z_L)
    v_L = v_oc*Z_L/(Z_L + Z_S);
end
Appendix E

Software Receiver Code Listing

The following code listings comprise the digital signal processing (DSP) source code for the C software receiver.

Listing E.1: DSP.h

/*
Filenames: DSP.h & DSP.cpp

Purpose: This library contains the discrete-time signal processing functions that are at the heart of the digital receiver, along with additional nonessential signal processing utilities.
*/

#ifndef DSP_H
#define DSP_H

#include "chanFilters.h"
#include "matched_filter.h"

// Each signal processing function produces an output sample // by processing a window of input samples. The following // constants define the lengths of these windows.
const int CHAN_WINDOW_LEN = M * CHAN_FILTER_LEN;
const int MATCHED_FILTER_WINDOW_LEN = MATCHED_FILTER_LEN - 1;
const int INTERPOLATOR_WINDOW_LEN = 4;
const int DPSSK_DETECTOR_WINDOW_LEN = 1;
const int RX_WINDOW_LEN = CHAN_WINDOW_LEN
 + (MATCHED_FILTER_WINDOW_LEN + INTERPOLATOR_WINDOW_LEN) * D
 + DPSSK_DETECTOR_WINDOW_LEN * 4 * D;

/*
Purpose: Performs frequency selection and downsampling on sampled input signal. Produces outputs for 1 to NUM_CHANS channels.

Parameters:
outReal,
outImg: (return value) Result of the channelization. Caller passes an array of pointers to pre-allocated memory. The vectors accessed are only those defined by the chansToUse parameter.
outLen: Length of passed-in out vectors. If this parameter is set to zero, the function will return the required length of the output vector without performing any other computation (all pointers can be set to null; only the numSamples parameter needs to have a meaningful value in this case).
chansToUse: input vector specifying which frequency channels are to be
output. The first "numChansToUse" elements of this array will be used to index the outReal and outImag arrays. For example, if numChansToUse is 1, the function will store the real components of the results in the vector at address outReal[chanstoUse[0]].

numChansToUse: Specifies the number of channels desired.
samples: Pointer to vector of real-valued input samples.
umSamples: Number of elements in the samples vector.

Return Value: Negative values indicate error conditions. Non-negative values give the number of valid entries in the out vector (or, if lenOut is 0, the number of valid entries that would have been computed).

Error condition codes:
-1: When outLen is less than the minimum required. To find the minimum required, call the function again with outLen set to 0.

extern int channelizer(double **outReal, double **outImag, int outLen, int *chanstoUse, int numChansToUse, double *samples, int numSamples);


Purpose: Performs matched filtering.

Parameters:
outReal,
outImag: (return value) Output of the matched filter. Caller passes pointers to pre-allocated memory.
outLen: Length of passed-in out vectors. If this parameter is set to zero, the function will return the required length of the output vector without performing any other computation (all pointers can be set to null; only the inLen parameter needs to have a meaningful value in this case).
inReal,
inImag: Pointers to vectors of input samples to be filtered.
inLen: Number of elements in inReal and inImag.

Return Value: Negative values indicate error conditions. Non-negative values give the number of valid entries in the out vector (or, if lenOut is 0, the number of valid entries that would have been computed).

Error condition codes:
-3: When outLen is less than the minimum required. To find the minimum required, call the function again with outLen set to 0.

extern int matchedFilter(double *outReal, double *outImag, int outLen, double *inReal, double *inImag, int inLen);


Purpose: Estimates the basepoint index and fractional interval required by the interpolator. See section 8.4.2 in Dr. Rice's textbook for details.

Parameters:
u: (return value) Fractional interval.
offset: (return value) Basepoint index.
sReal,  sImag:   Pointers to vectors of input samples to be processed.
sLen:   Number of elements in all input vectors.
*/
extern void timingParameterEstimation(double *mu, int *offset, double *
sReal, double *sImag, int sLen);

/
Purpose: Performs piecewise polynomial interpolation to produce an
output signal downsampled by a factor of four from the input
signal. See section 8.4.2 in Dr. Rice's textbook for details.
This is implemented to match the block diagram given in the
upper portion of figure 8.4.16.

Parameters:
outReal,
outImag:   (return value) Output of the operation. Caller passes
           pointers to pre-allocated memory.
outLen:   Length of passed-in out vectors. If this parameter is set to
           zero, the function will return the required length of the
           output vector without performing any other computation
           (all pointers can be set to null; only the inLen and offset
           parameters need to have a meaningful values in this case).
inReal,
inImag:   Pointers to vectors of input samples to be processed.
inLen:   Number of elements in all input vectors.
mu:      fractional interval.
offset:  basepoint index.

Return Value: Negative values indicate error conditions. Non-negative
values
give the number of valid entries in the out vector (or, if outLen is 0,
the number of valid entries that would have been computed).

Error condition codes:
-1:   When outLen is less than the minimum required. To find the
minimum required, call the function again with outLen
set to 0.
*/
extern int farrowInterp(double *outReal, double *outImag, int outLen,
double *inReal, double *inImag, int inLen, double mu, int offset);

/
Purpose: Performs differential QPSK detection.

Parameters:
outBits:   (return value) Vector of output bits. Caller passes
           pointers to pre-allocated memory.
outBitsLen: Length of passed-in outBits vector. If this parameter is set to
           zero, the function will return the required length of the
           output vector without performing any other computation
           (all pointers can be set to null; only the sLen
           parameter needs to have a meaningful values in this case).
sReal,  sImag:   Pointers to vectors of input samples to be processed.
sLen:   Number of elements in input vectors.
Return Value: Negative values indicate error conditions. Non-negative values give the number of valid entries in the output vector (or, if `outLen` is 0, the number of valid entries that would have been computed).

Error condition codes:
-1: When `outBitsLen` is less than the minimum required. To find the minimum required, call the function again with `outBitsLen` set to 0.

/*
extern int DQPSKDetector(char *outBits, int outBitsLen, double *sReal, double *sImag, int sLen);

/*
Purpose: Estimates the time-average power in a complex signal.

WARNING: This is scaled by 2 to account for channelizer attenuation!!

Parameters:
- `power`: (return value) Result of computation. Units: Watts.
- `sReal`, `sImag`: Pointers to vectors of input samples to be processed.
- `sLen`: Number of elements in input vectors.

extern void estimatePower(double *power, double *sReal, double *sImag, int sLen);

/*
Purpose: Estimates the time-average power in a real signal.

Parameters:
- `power`: (return value) Result of computation. Units: Watts or dBm.
- `s`: Pointer to vector of input samples to be processed.
- `sLen`: Number of elements in input vector s.

extern void estimatePower(double *power, double *s, int sLen);
extern void estimatePower_dBm(double *power, double *s, int sLen);

/*
Purpose: Produces a zero-mean version of the input signal. Computes
the mean of the real and imaginary parts of the input signal, and
subtracts those mean values from each element of the input signal
to form the output signal.

Parameters:
- `outReal`, `outImag`: (return value) Output of the operation. Caller passes
  pointers to pre-allocated memory. Can point to same
  memory as `inReal` and `inImag` if desired.
- `inReal`, `inImag`: Pointers to vectors of input samples to be processed.
- `inLen`: Number of elements in all input vectors.

extern void subtractMean(double *outReal, double *outImag, double *inReal, double *inImag, int len);

#endif
Listing E.2: DSP.cpp

#include "DSP.h"
define _USE_MATH_DEFINES
#include <math.h>
#include "conv.h"

#define MAX(X, Y) (((X) > (Y)) ? (X) : (Y))

/* NOTE ON CHANNELIZER IMPLEMENTATIONS
Two competing implementations of the channelizer are available.
Both implementations are based on the architecture presented in
"The Freeline Receiver" by Dr. Rice. A detailed derivation is
presented in Dr. Rice's textbook "Communications: A Discrete-Time
Approach," section 10.2. Performance comparisons favor the new
implementation when using all 13 channels, but favor the old
implementation when using a single channel. Commenting out the
following preprocessor definition will select the old
implementation. */
define NEW_CHANNELIZER

#undef NEW_CHANNELIZER

// the old channelizer implementation
int chansToUse
int chansToUse, int numChansToUse, double *samples, int numSamples) {
    double databank[][CHANNELIZER_LEN][chanOutReal, double **outImg, int outLen, int *
        chanOutLen = (numSamples + D - (numSamples % D)) / D;

        if (outLen == 0) return chanOutLen;
        else if (outLen < chanOutLen) return -1;

        // initialize databank to zeros
        for (int i = 0; i < M; i++) {
            for (int j = 0; j < CHANNELIZER_LEN; j++) {
                databank[i][j] = 0.0;
            }
        }

        for (int i = 0; i < chanOutLen + CHANNELIZER_LEN; i++ ) {

            // shift elements in databank and initialize with new sample values
            if (i == 0) {
                databank[0][i] = samples[0];
```c
k = 1;
}
else {

    // shift data within databank
    for (int j = CHAN_FILTER_LEN - 1; j > 0; j--) {
        for (int l = 0; l < D; l++) {
            databank[D + 1][j] = databank[1][j];
            databank[1][j] = databank[D + 1][j - 1];
        }
    }

    // edge case, still shifting data within databank
    for (int l = 0; l < D; l++) {
        databank[D + 1][0] = databank[1][0];
    }

    // introducing new elements from the sample buffer
    if (k > D) {
        for (int l = 0; l < D; l++) {
            databank[1][0] = samples[k - 1 - 1];
        }
    }
}

// array initialization
for (int j = 0; j < numChansToUse; j++) {
    currentChanOutReal[j] = 0.0;
    currentChanOutImag[j] = 0.0;
}

// for each branch of the polyphase filter bank...
for (int m = 0; m < M; m++) {
    temp[m] = 0.0;

    // polyphase filter convolution
    for (int j = 0; j < CHAN_FILTER_LEN; j++) {
        temp[m] += databank[m][j] * CHAN_FILTER[m][j];
    }

    // bank of M heterodynes summed together
    for (int j = 0; j < numChansToUse; j++) {
        currentChanOutReal[j] += CHAN_HETER_REAL[m][CHAN_12_00_MHZ -
                                             chansToUse[j] - MIN_K_VALUE] * temp[m];
        currentChanOutImag[j] += CHAN_HETER_IMG[m][CHAN_12_00_MHZ -
                                                chansToUse[j] - MIN_K_VALUE] * temp[m];
    }
}

// final heterodyne (Reduced to multiplication by 1 or -1. See eqn. (10.58) in Rice.)
for (int j = 0; j < numChansToUse; j++) {
    if((CHAN_12_00_MHZ - chansToUse[j]) % 2 == 0) {
```

109
```c
outReal[chansToUse[j]][i] = currentChanOutReal[j];
outImag[chansToUse[j]][i] = currentChanOutImag[j];
} else {
    if(i % 2 == 0) {
        outReal[chansToUse[j]][i] = currentChanOutReal[j];
        outImag[chansToUse[j]][i] = currentChanOutImag[j];
    } else {
        outReal[chansToUse[j]][i] = -currentChanOutReal[j];
        outImag[chansToUse[j]][i] = -currentChanOutImag[j];
    }
}
}
k += D;

if (k >= numSamples) {
    break;
}
}

return chanOutLen;
}

# else

// the new channelizer implementation
int channelizer(double **outReal, double **outImag, int outLen, int *chansToUse, int numChansToUse, double *samples, int numSamples) {

double polyFilterOut[M];
double sumReal, sumImag;
int sampleIndex;
int indexOffset = (int)ceil((double)(M - 1) / D);
int chanOutLen = MAX(((numSamples - 1) - M * (CHAN_FILTER_LEN - 1)) / D - indexOffset + 1, 0);

if (outLen == 0)
    return chanOutLen;
else if (outLen < chanOutLen)
    return -1;

// for each channelizer output
for (int i = 0; i < chanOutLen; i++) {

    // for each branch of the polyphase filter
    for (int m = 0; m < M; m++) {
        polyFilterOut[m] = 0.0;
        for (int j = 0; j < CHAN_FILTER_LEN; j++) {

            // this complicated index is at the heart of the
            // commutation/downsample operation
            sampleIndex = (i + indexOffset) * D + M * j - m;

            // one of the M subfilters
````
polyFilterOut[m] += samples[sampleIndex] * CHAN_FILTER[m][CHAN_FILTER_LEN - j - 1];
}
}

for (int j = 0; j < numChansToUse; j++) {
    sumReal = 0.0;
    sumImag = 0.0;

    // bank of M phase shifter heterodynes
    for (int m = 0; m < M; m++) {
        sumReal += polyFilterOut[m] * CHAN_HETER_REAL[m][CHAN_12_00_MHZ - chansToUse[j] - MIN_K_VALUE];
        sumImag += polyFilterOut[m] * CHAN_HETER_IMAG[m][CHAN_12_00_MHZ - chansToUse[j] - MIN_K_VALUE];
    }

    // final heterodyne (reduced to multiplication by 1 or -1)
    if (((CHAN_12_00_MHZ - chansToUse[j]) % 2 == 0) {
        outReal[chansToUse[j]][i] = sumReal;
        outImag[chansToUse[j]][i] = sumImag;
    } else {
        if (i % 2 == 0) {
            outReal[chansToUse[j]][i] = sumReal;
            outImag[chansToUse[j]][i] = sumImag;
        } else {
            outReal[chansToUse[j]][i] = -sumReal;
            outImag[chansToUse[j]][i] = -sumImag;
        }
    }
}

return chanOutLen;
}

int matchedFilter(double *outReal, double *outImag, int outLen, double *inReal, double *inImag, int inLen) {
    
    conv(outReal, outLen, inReal, inLen, (double *)MATCHED_FILTER, MATCHED_FILTER_LEN, CONV_VALID);
    return conv(outImag, outLen, inImag, inLen, (double *)MATCHED_FILTER, MATCHED_FILTER_LEN, CONV_VALID);
}

void timingParameterEstimation(double *mu, int *offset, double *sReal, double *sImag, int sLen) {
    double X_OM_real = 0.0;
    double X_OM_imag = 0.0;
    double x_OM;
    double epsilon;

    for (int i = 0; i < sLen; i++) {

x_0M = sReal[i] * sReal[i] + sImag[i] * sImag[i];

switch(i % 4) {
    case 0:
        X_0M_real += x_0M;
        break;
    case 1:
        X_0M_imag += x_0M;
        break;
    case 2:
        X_0M_real -= x_0M;
        break;
    case 3:
        X_0M_imag -= x_0M;
        break;
}
}

epsilon = atan2(X_0M_imag, X_0M_real) / M_PI;

if (0.0 <= epsilon && epsilon < 0.5) {
    *offset = 0;
    *mu = epsilon * 2.0;
} else if (0.5 <= epsilon && epsilon <= 1.0) {
    *offset = 1;
    *mu = (epsilon - 0.5) * 2.0;
} else if (-1.0 <= epsilon && epsilon < -0.5) {
    *offset = 2;
    *mu = (epsilon + 1.0) * 2.0;
} else if (-0.5 <= epsilon && epsilon < 0.0) {
    *offset = 3;
    *mu = (epsilon + 0.5) * 2.0;
}

int farrowInterp(double *outReal, double *outImag, int outLen, double *inReal, double *inImag, int inLen, double mu, int int offset) {

double v2r, v1r, v0r, v2i, v1i, v0i;
in outIndex = 0;
in computedOutLen;

// compute needed outLen
computedOutLen = MAX(inLen / 4 - 1, 0);

if (outLen == 0)
    return computedOutLen;
else if (outLen < computedOutLen)
    return -1;

int iStart = (offset == 0) ? 4 : offset;

for (int i = iStart; i < inLen - 2; i += 4) {
v2r = 0.5 * (inReal[i + 2] - inReal[i + 1] - inReal[i] + inReal[i - 1]);
v2i = 0.5 * (inImag[i + 2] - inImag[i + 1] - inImag[i] + inImag[i - 1]);
v1r = 0.5 * (-inReal[i + 2] + inReal[i + 1] - inReal[i] - inReal[i - 1]) + inReal[i + 1];
v1i = 0.5 * (-inImag[i + 2] + inImag[i + 1] - inImag[i] - inImag[i - 1]) + inImag[i + 1];

v0r = inReal[i];
v0i = inImag[i];

outReal[outIndex] = (mu * v2r + v1r) * mu + v0r;
outImag[outIndex] = (mu * v2i + v1i) * mu + v0i;

outIndex++;

if (outIndex == computedOutLen) break;
}

return computedOutLen;
}

int DQPSKDetector(char *outBits, int outBitsLen, double *sReal, double *sImag, int sLen) {

double test0_real;
double test0_imag;
double test90_real;
double test90_imag;
double test180_real;
double test180_imag;
double test270_real;
double test270_imag;

double d2_0;
double d2_90;
double d2_180;
double d2_270;

int numBits = (sLen - 1) * 2;

if (outBitsLen == 0) return numBits;
else if (outBitsLen < numBits) return -1;

for (int i = 1; i < sLen; i++) {

test0_real = sReal[i - 1];
test0_imag = sImag[i - 1];
test90_real = -sImag[i - 1];
test90_imag = sReal[i - 1];
test180_real = -sReal[i - 1];
test180_imag = -sImag[i - 1];
test270_real = sImag[i - 1];
test270_imag = -sReal[i - 1];
d2_0 = (sReal[i] - test0_real) * (sReal[i] - test0_real) + (sImag[i] - test0_imag) * (sImag[i] - test0_imag);
d2_90 = (sReal[i] - test90_real) * (sReal[i] - test90_real) + (sImag[i] - test90_imag) * (sImag[i] - test90_imag);
d2_180 = (sReal[i] - test180_real) * (sReal[i] - test180_real) + (sImag[i] - test180_imag) * (sImag[i] - test180_imag);
d2_270 = (sReal[i] - test270_real) * (sReal[i] - test270_real) + (sImag[i] - test270_imag) * (sImag[i] - test270_imag);

// Decide Rotation 0
if (d2_0 <= d2_90 && d2_0 <= d2_180 && d2_0 <= d2_270) {
    outBits[2*(i-1)] = 1; //MSB
    outBits[2*(i-1)+1] = 1; //LSB
}
// Decide Rotation 90
else if (d2_90 <= d2_0 && d2_90 <= d2_180 && d2_90 <= d2_270) {
    outBits[2*(i-1)] = 0; //MSB
    outBits[2*(i-1)+1] = 1; //LSB
}
// Decide Rotation 180
else if (d2_180 <= d2_0 && d2_180 <= d2_90 && d2_180 <= d2_270) {
    outBits[2*(i-1)] = 0; //MSB
    outBits[2*(i-1)+1] = 0; //LSB
}
// Decide Rotation 270
else {
    outBits[2*(i-1)] = 1; //MSB
    outBits[2*(i-1)+1] = 0; //LSB
}
}
return numBits;
}

void estimatePower(double *power, double *sReal, double *sImag, int sLen)
{
double p;
p = 0.0;
for(int i = 0; i < sLen; i++) {
p += (sReal[i] * sReal[i]) + (sImag[i] * sImag[i]);
}
*power = p / (double)sLen / 50.0 * 2.0; // 50 Ohm load, factor of 2 because channelizer attenuates signal by 3 dB (?)
}

void estimatePower(double *power, double *s, int sLen)
{
double p = 0.0;
for(int i = 0; i < sLen; i++) {
p += pow(s[i], 2);
}
```c
}
*power = p / (double)sLen / 50.0;
}

void estimatePower_dBm(double *power, double *s, int sLen) {
  double p;
  estimatePower(&p, s, sLen);
  *power = 10.0 * log10(p) + 30.0;
}

void subtractMean(double *outReal, double *outImag, double *inReal, double *inImag, int len) {
  double meanReal = 0.0;
  double meanImag = 0.0;

  // compute mean
  for (int i = 0; i < len; i++) {
    meanReal += inReal[i];
    meanImag += inImag[i];
  }
  meanReal /= len;
  meanImag /= len;

  // subtract mean
  for (int i = 0; i < len; i++) {
    outReal[i] = inReal[i] - meanReal;
    outImag[i] = inImag[i] - meanImag;
  }
}
```