Spring 2011

Design and verification of physical layer architecture for a 1-wire sensor communication bus

Benjamin Michael Peiffer
University of Iowa

DESIGN AND VERIFICATION OF PHYSICAL LAYER ARCHITECTURE FOR A 1-WIRE SENSOR COMMUNICATION BUS

by

Benjamin Michael Peiffer

A thesis submitted in partial fulfillment of the
requirements for the Master of Science degree
in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

May 2011

Thesis Supervisor: Associate Professor Anton Kruger

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

MASTER'S THESIS

_____

This is to certify that the Master's thesis of

Benjamin Michael Peiffer

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Electrical and Computer Engineering at the May 2011 graduation.

Thesis Committee: _____
Anton Kruger, Thesis Supervisor

_____
Mark Andersland

_____
Soura Dasgupta

_____
Raghuraman Mudumbai

To my family.

People think it's an obsession. A compulsion. As if there were an irresistible impulse to act. It's never been like that. I chose this life. I know what I'm doing. And on any given day, I could stop doing it. Today, however, isn't that day. And tomorrow won't be either.

Batman, in Brad Meltzer's *Identity Crisis*

ACKNOWLEDGMENTS

I would like to thank my committee members for serving on my committee. I would like to thank all of my colleagues at IIHR for their input, especially Dr. Jim Niemeier for helping me to understand the architecture of the soil moisture network that was previously deployed in Ames, Hamid Fahim Rezaei for discussing various communications topics that helped me to come up with new ideas, and Nick Sitter for his help with the parasitic power circuit, the build of the first hardware prototype, and advice on using the ATTiny45 chip. I would also like to thank IIHR for their financial support during the time that I was working on this project. Finally, I would like to thank my advisor, Dr. Anton Kruger, for his support. This development would not have happened if it were not for his ideas and our many long discussions about the fundamental issues of implementing this type of network.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Within this thesis, the problem of device and sensor communication is discussed. Recent sensor network implementations and deployments at the University of Iowa IIHR-Hydroscience and Engineering (IIHR) and the Iowa Flood Center (IFC) have motivated the need for an efficient, shared, 1-wire communication scheme. Possible solutions will be explored and discussed briefly within this chapter. The focus of this thesis is the development of a particular solution for a 1-wire physical layer utilizing the concept of Code Division Multiple Access (CDMA).

Instrumentation, Sensor Networks, Environmental Sensors

The most basic interface for commonly used sensors in environmental and geosciences research such as temperature and humidity sensors and optical radiometers is a simple analog voltage output. The attendant data logger samples and digitizes the voltages. The signal levels and the analog nature of the signals limit the distance between sensors and data logger since long wires can cause voltage drops and are susceptible to noise. To address the issues, many sensors have associated electronics that amplify, modulate and otherwise manipulate the signals making them more amenable to transmission. For example, in one scheme the voltage at the sensor is converted to a square wave with a frequency proportional to the voltage. The data logger measures the frequency of the square wave. Other schemes include pulse-width-modulated square waves and current loops.

The sensor interfaces discussed up to this point are essentially analog. Many expensive sensors and virtually all full-blown instrumentation have digital interfaces. In this instance, electronics at the sensor digitize the sensor voltage and provide the information in digital format to the data logger. Thus a wind-speed and wind-direction sensor may have an RS-232 interface that one can connect to, for example, a PC. One

would then send simple ASCII commands to the sensors or instruments via a terminal emulator and interrogate the sensor for wind-speed and direction. Commercial data loggers have a number of serial interfaces including RS-232.

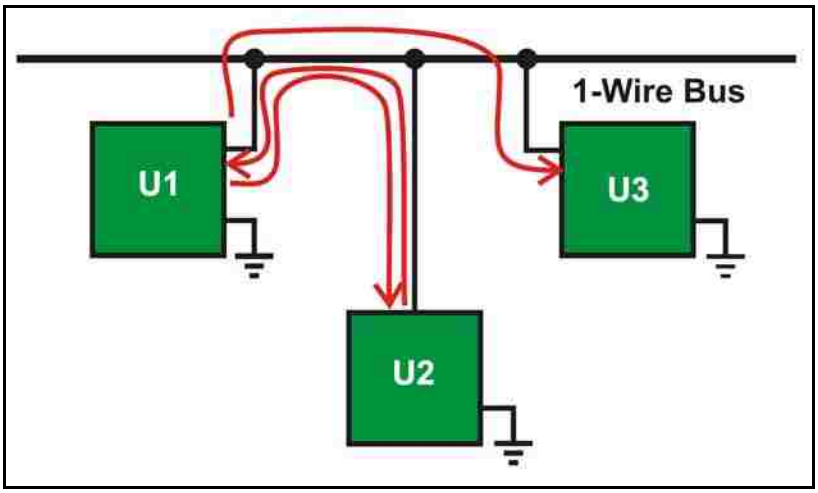

Figure 1. A 1-Wire Bus Where Multiple Users Can Transmit Simultaneously. This figure illustrates the objective of this thesis. A 1-wire bus where users can communicate asynchronously, simultaneously, and independent of other bus users, perhaps even at different rates.

Thus far, a star configuration has been assumed—all the sensors interface directly to a data logger. However, there are also sensor bus interfaces available. The advantages of using a bus are clear: it simplifies the physical interconnection, is cost effective, and is flexible. One widely used and well-documented bus is the RS-485 standard. Another common bus is Serial Data Interface At 1200 Baud (SDI-12). All busses, including SDI-12 and RS-485, need some form of Medium Access Control (MAC). This is typically implemented using Time Division Multiple Access (TDMA) with a bus master that manages access to the communication channel. The use of a TDMA as a MAC has several problems. Only one user at a time can own the bus and timing is absolutely

critical. The timing constraint is especially problematic because it dictates slew rates, cable capacitance and, consequently, cable length.

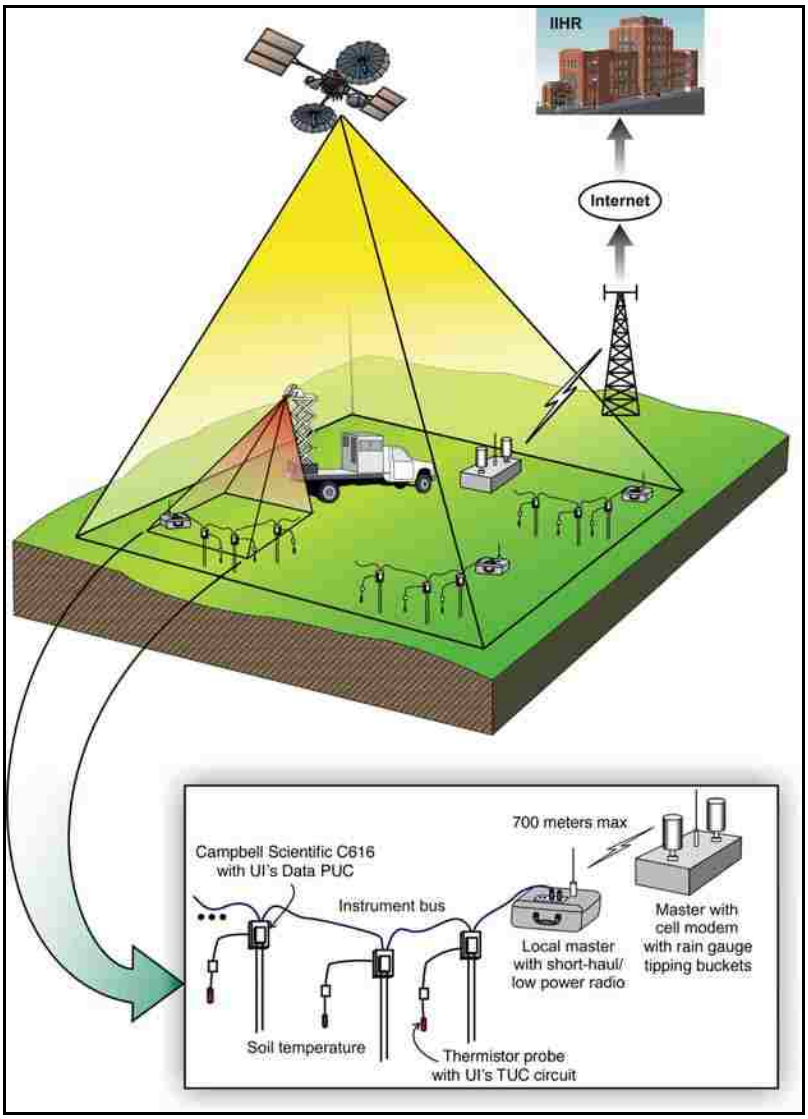

Figure 2. IIHR Soil Moisture Network Overview from [21]. This network provides significant motivation for the bus described in this thesis. The network contains wireless nodes, each with its own instrument bus. Several soil moisture and temperature probes are connected to each instrument bus and communicate using Time Division Multiple Access in timeslots assigned by the local master.

Ideally, a 1-wire communication interface as shown in Figure 1 could be developed. This ideal interface would lessen the need for strict synchronization and eliminate the star configuration by allowing many users to communicate asynchronously without a bus master. 1-wire communication protocols do exist and are in use. For instance, the Dallas Semiconductor 1-Wire interface uses a single wire. However, this system provides a good contrast to the ideal 1-wire interface. In Dallas 1-Wire a single bus-master is required, users must wait for a turn to transmit (TDMA), the master must announce the time-slots, and devices must synchronize to a certain data rate.

Case Study: Soil Moisture Network

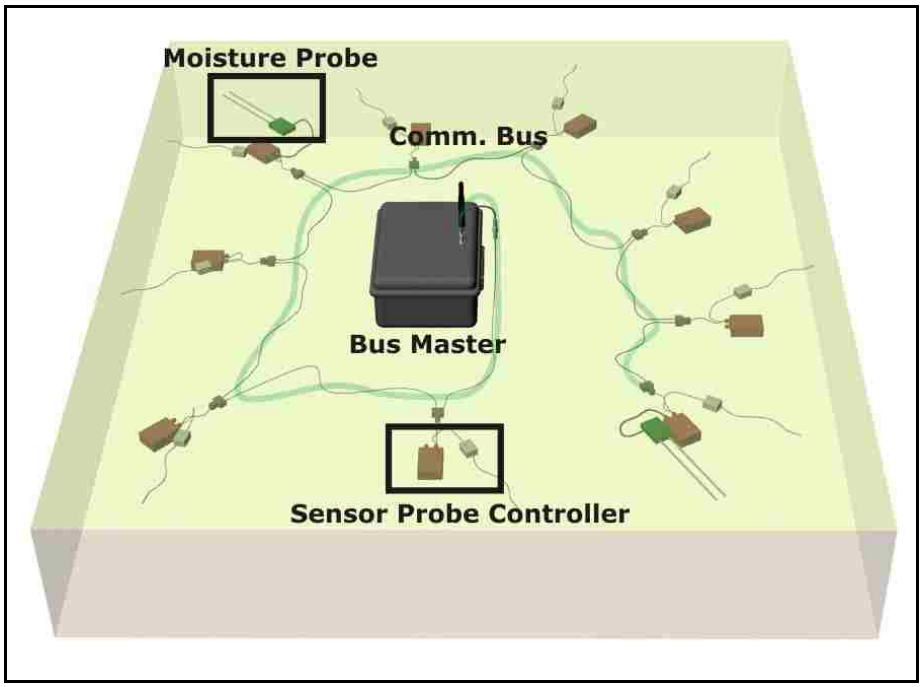

Figure 3. Soil Moisture Network Sensor Bus Illustration from [21]. Annotations have been added to show relevant communication and sensor hardware and highlight the 1-wire communication bus. The bus master is responsible for assigning time-slots to the sensor probe controllers and relaying the data to a relay node that has a cellular modem.

In [21], Niemeier discusses a Wireless Sensor Network (WSN), shown in Figure 2, that was designed and manufactured to measure soil moisture and soil temperature. The network contains several subnets, as shown in Figure 3, each with its own instrument bus that wirelessly relays data to a system master with a cell modem for data storage. The system uses TDMA within each subnet to provide the physical layer communication among sensors on the bus. Although this was a successful approach, it came at the cost of high software complexity and the need for a common clock at each node in the network.

A primary purpose of this research is to reconsider this system and ask the question: is there a simpler way to implement the bus interface? Ideally, a system could be developed that requires no strict synchronization, low hardware complexity, low software complexity, and supports multiple transceivers on a single wire.

Wireless Multiplexing Strategies

As a first approach to developing a wired shared-channel communication scheme, one could consider previous work that has been done in wireless communications. Perhaps it is possible to convert some of these strategies to a wired channel without significant performance losses. Popular methods of wireless multiplexing are Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA). All of these techniques are discussed in mo communications courses and are explored in introductory texts [19,25].

Each of these multiplexing techniques could be used to share a wired channel with a few minor adjustments from their wireless implementations. TDMA is widely used for wired networks. While both FDMA and CDMA would be interesting to explore in a wired channel, the focus of this thesis will be CDMA.

Developing a wired CDMA provides significant opportunities and challenges. In wireless CDMA, designers must combat the *near-far problem*. Dealing with this

problem adds additional difficulty to the design of the transmitter and receiver hardware. In a wired CDMA, whether it is analog or digital, signals will not take on nearly the dynamic range that wireless signals do, and no near-far problem will exist. Depending on the chosen channel implementation, other problems may make the design more difficult, but a major implementation obstacle has been overcome. Another problem in wireless CDMA is assigning codes to users as they drift from cell to cell in the network. Again, this problem does not apply in a wired CDMA, where users are fixed on a bus for their entire lifetime. This means that codes can be assigned to users at build time. The largest opportunity for wired CDMA is the transparent access that it provides. Users do not need to know that other users are on the bus. Users can be added or removed ad-hoc and the bus could be used to host conversations between different groups of users, possibly even at different bit-rates if desired.

Implementations of Wired CDMA

In [22,24], Nikoclic describes the use of a CDMA protocol within the context of Very Large Scale Integration (VLSI) and System on a Chip (SoC). The primary goal is to implement a shared channel between logical units in a SoC. In this case, CDMA is used to communicate over either a k-bit wide peripheral bus or a 1-wire analog channel. In the *k*-bit wide bus, a receiver can view the *k*-lines as bits of a signal with 2^k possible values. At each time instant, the receiver identifies the input value, and then behaves exactly as a wireless CDMA receiver.

While both of the approaches described in [22,24] are valid and appropriate for VLSI systems, they do not accomplish the goals outlined in this thesis. In the single-wire case, the hardware complexity can be high, as is shown in [3,5]. In either case, a true bus interface is no longer being implemented. Instead, there are two star configurations that are interconnected, by either 1 or *k* wires with the center of the star performing a summation operation.

As an alternative to the common TDMA-based busses and the multi-wire or analog CDMA, a simple bus interface for environmental instrumentation will be discussed. The bus is a 1-wire, all digital, CDMA implementation. There is no bus master or TDMA involved in the protocol. Devices on the bus can transmit and receive asynchronously without interfering with other channel sharers. First, work that has been done in Optical Channel implementations of CDMA will be explored.

Optical CDMA

CDMA has been utilized as a MAC for optical channels in addition to wireless channels. In [26], the author gives a historical perspective on Optical CDMA. The first significant research was reported in the mid-1980s. This thesis will not explore all of this research−[26,32] provides a good overview of the subject. Of particular interest in this thesis are the codes used in Optical CDMA. There are two classes of codes that have been used: coherent and incoherent codes. Several different coding schemes have been designed for each class of codes. In a coherent optical system, optical fields are induced by the use of phase modulators, allowing for use of bipolar codes (i.e. +1/-1 values can be sent). In an incoherent optical system, no phase information is contained and optical intensity is used to encode information, this implies that only unipolar codes can be used (i.e. +1/0 values can be sent). A unique characteristic of incoherent optical systems is the assymmetric nature of channel errors. For instance, if two users are sharing a piece of optical fiber and both are transmitting a '1', light exists in the optical channel and no error can conceivably occur. An error only occurs in the when one user is transmitting a '1' and the other user is attempting to transmit a '0' at the same time. In this case, any user attempting to receive the transmitted '0' is subject to a $0 \rightarrow 1$ error. On the other hand, no $1 \rightarrow 0$ error can occur in such a system. Two interesting incoherent codes that have been used for optical systems are the *Optical Orthogonal Codes* (OOC) and *Prime Codes*. Prime Codes have been developed for and used in various other applications, but

can be adapted to optical systems. These code sets are both designed to be low weight, meaning the number of logic-1s is minimized. A fair amount of research has been conducted in the area of OOCs. This literature will not be fully explored here, but instead clear similarities will be identified between the desired system and an Optical CDMA implementation, and thus the choice of similar coding schemes will be motivated.

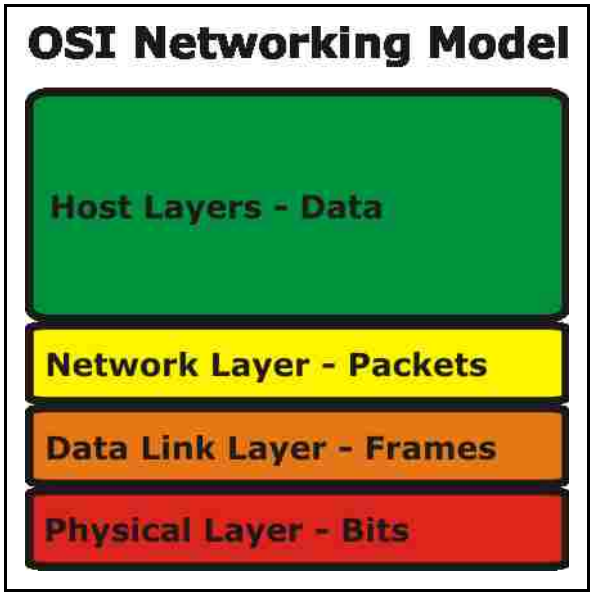

Figure 4. The Open Systems Interconnection (OSI) Networking Model is shown here. This model is referred to throughout this thesis, especially the Physical Layer. See [4] and [17] for a general reference on networking and the OSI Model.

System Proposal

The proposed system is a 1-wire, digital, asynchronous CDMA system in a bus (rather than star) configuration. The shared channel will be binary and a wired-OR channel model will be applied. At a minimum, the channel will be capable of supporting 8 users at relatively low data rates (<1kbps). The system is primarily targeted at low data rate environmental sensor applications, but the ability to apply this concept to other environment will also be considered in principle. This thesis will focus on the

development of the Physical Layer (PHY) of the OSI networking model for the CDMA Communication System. The Data Link and Network Layers will be considered when necessary to field a practical system, but most of these higher-level details will be implemented by the host platform.

Conclusion

In this chapter, the problem of a 1-wire communication network was proposed. The IIHR Soil Moisture Network was introduced as an example of such a communication network. Alternative strategies for the implementation of such a network were discussed and the desire to further explore a 1-Wire CDMA strategy was motivated. Previous work in wired CDMA and optical CDMA was explored, with a particular interest in how this previous work can help to create a system that better meets the needs outlined here.

CHAPTER II

WIRED CDMA SYSTEM OVERVIEW

In this chapter, the proposed system for the Physical Layer of a 1-Wire CDMA communication system is introduced in detail. This includes a discussion of system requirements for the system. The channel model is also explored briefly and the basic transmitter and receiver schemes are introduced.

System Requirements

In order to understand the reasons for designing a protocol like the one described in the introduction, detailed requirements must be formed and analyzed. The analysis of these requirements is what leads to the conclusion that the use of CDMA with a Binary Channel model would be convenient in certain applications.

1. No Strict Synchronization - The physical layer should have no requirement for phase or frame synchronization. Transmitter and receiver pairs need to agree on a data rate only and no bus master clock is needed.
2. Low Hardware Complexity - Each wire on the bus should have digital values to minimize the amount of analog circuitry needed. This will help to reduce the real cost of implementation, which is an important factor for a transceiver that will theoretically be included many times as a component in a single system design. This will also help reduce size, which is, again, an important factor for a component that will be reused many times.
3. Low Software Complexity - This will reduce the non-recurring engineering costs (NRE) associated with software development time. These NRE costs will be realized whenever the communication protocol is ported to a new device, so it is important to keep them as low as possible.
4. Multiple Users on a Single Data Wire - The physical layer should have the capability to support full duplex communication on a single wire and should be

operable when multiple users simultaneously share the channel. This requirement will reduce the variable costs associated with interconnection of transceivers. This could be in the form of traces on a printed circuit board (PCB), board-to-board interconnects, or cabling between physically separate nodes.

It is also important to outline the non-goals of the system. In other words, there are certain objectives that are explicitly unimportant in the system design. They are defined here.

1. High Data Rate Connection – A high data rate connection is not necessary since the data transfers in the typical target network are a few bytes at a time and are bursty in nature.
2. Hundreds of Simultaneous Users – It is not necessary to provide support for hundreds of users because it is difficult to physically collocate so many nodes and is unnecessary to have so many measurements in such a close area when it comes to measurements like soil moisture, soil temperature, rainfall, humidity, etc. This thesis will aim to support at least 8 users on the bus.
3. 0% Bit Error Rate – Error free transmissions are not necessary for a few reasons. The most important reason is that error correction techniques can be layered into the network.

Based on these requirements, it is logical that a binary channel must be used. This will reduce the cost of hardware by requiring only digital components. In order to further reduce complexity, a wired-OR is chosen as the channel model. This channel model can be implemented with very inexpensive and simple hardware components, and the implementation does not depend on the number of users. Potential implementations of this channel model will be discussed in more detail later on. With these assumptions, each transmitter can behave in a way that is inspired by wireless CDMA environment and each receiver can behave in a similar way, but will likely require optimizations that are specific to the channel model.

Channel Model and Logical Values

It is extremely important to note that the choice of a wired-OR channel is somewhat arbitrary here. Certainly, the channel could be viewed as or implemented as a wired-NOR, wired-AND, or wired-NAND depending on choices made in software or hardware. For clarity, the channel is and will be referenced as wired-OR, which could be considered an abstraction layer for the real channel.

Along with this choice of channel abstraction layer, the system will be described with 2-valued logic. Logic values will be referred to as logic-1 and logic-0. In real hardware, these values may be implemented using positive or negative logic, depending on the channel model chosen. With the wired-OR abstraction, it is obvious that logic-1 is the value that drives the channel to logic-1 and logic-0 is the value that lets the channel either be driven to logic-1 or remain at logic-0, depending on the transmissions of other users.

Binary Wired-OR Channel Model

In wireless CDMA the message is transmitted through an additive, fading channel. While the additive nature of the channel is helpful in the application of CDMA, the fading forces the designer to deal with and find a solution for the near-far problem. In a binary wired-OR CDMA the benefits of the additive channel are lost, but the costs associated with the fading channel do not need to be accounted for. The channel noise need be modeled only in the form of other users, which are significant only when they are signaling a one to the channel, in which case any user asserting a zero suffers an error. This noise source will be referred to as *Co-Channel Interference* throughout the remainder of this thesis. This phenomenon is very similar to the Z-Channel, which is commonly discussed in Information Theory, especially with regard to Optical Communications. The Z-Channel, shown in Figure 5, is defined by a probability of 0 for $1 \rightarrow 0$ errors and a probability of p for $0 \rightarrow 1$ errors. For a more detailed discussion of Z-

Channel see [28]. The primary difference between the Z-Channel and the binary wired-OR channel is that the value of p will be dependent on both the number of users currently asserting transmissions on the bus and the probability of each of those users asserting a 1 on the bus.

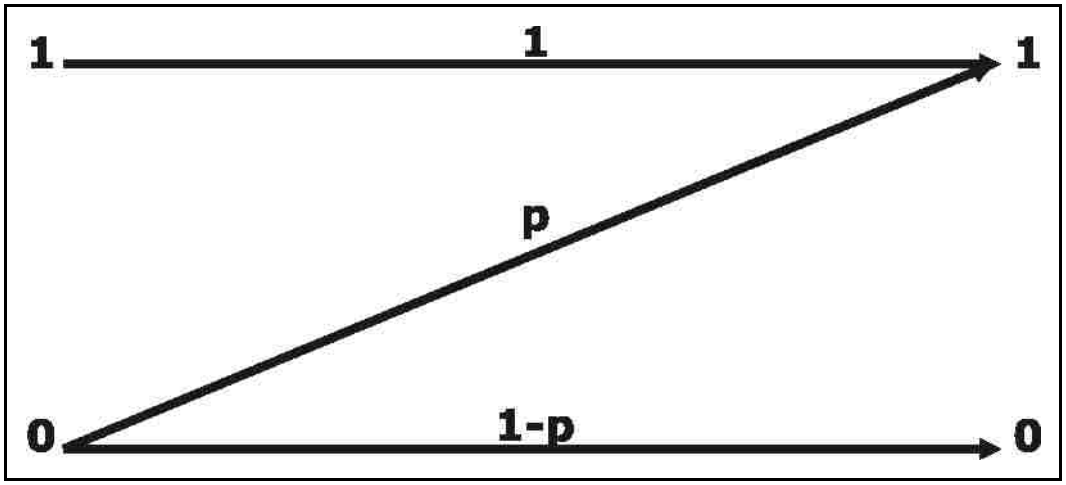

Figure 5. Information Theory Z-Channel Model. This model is commonly used to emulate the channels found in optical communication and DRAM storage.

System Components

The system can be described by four blocks, shown in Figure 6. The first block is the transmitter of interest, also referred to as the encoder. At any time, the input to the transmitter is a single bit of data, either a 1 or a 0. The output of the transmitter is a sequence of N digital values, referred to as *chips*. These N values are random in nature and are logic-1 valued with probability p. N is referred to as the *code length*. In a wireless CDMA system, the value of p=.5 is optimal.

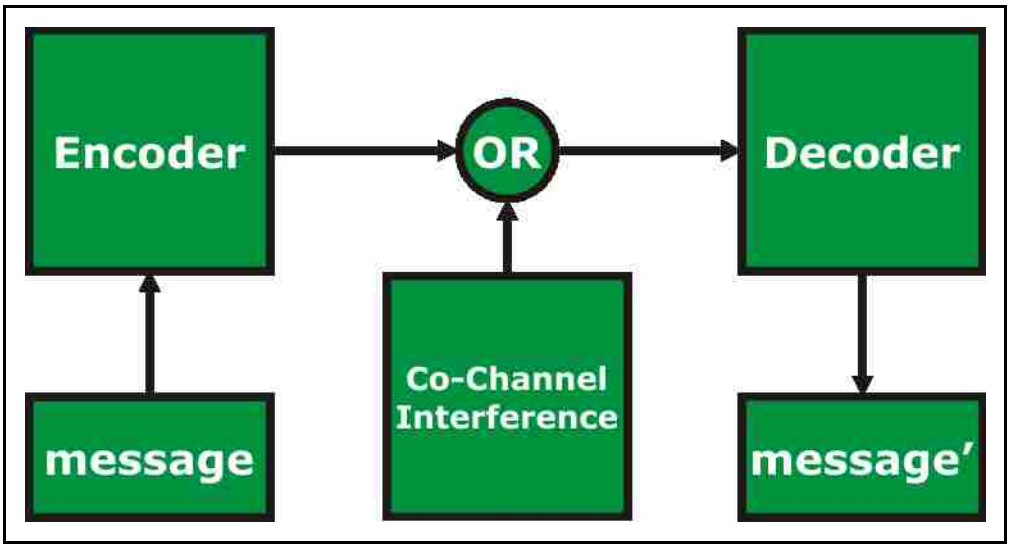

Figure 6. System Model Block Diagram. The block diagram shows the unique features of this system compared to a typical communications scheme. These include the wired-OR channel, the Co-Channel Interference, and the lack of other noise sources.

The second block is the binary wired-OR channel. This system takes in up to M binary values and performs an OR operation on these values. M is the maximum number of users allowed to share the system. The output of the system is the OR of the inputs. In the simplified system model, this block has two inputs, the output from the transmitter and the output from the co-channel interference block, which will be discussed next.

The third block is the co-channel interference block. This system produces a single digital value with probability producing a logic-1 value equal to $\overline{p}$, which is a function of the individual probability of a transmitter producing a logic-1 value and the number of users included in the co-channel interference block.

The final block is the receiver or decoder system. The input to the receiver is the output of the wired-OR channel. The output of the receiver is a single digital value. If the sequence of digital outputs at the receiver matches the sequence digital inputs at the transmitter, then no bits were in error. Typically, a physical layer will allow some errors

to be propagated. Thus, the system performs with bit error rate of a/b, where a is the number of bits in error and b is the number of bits that were transmitted.

Key Design Parameters Discussed

Now that the system requirements and structure have been roughly defined, it is important to clarify the degrees of freedom in the design of the system. A large part of this thesis will be spent evaluating various choices for these design parameters and considering the merits of varying those choices. In fact, the primary goal is to make design choices that will allow a real system to be implemented on hardware and used in a wired network.

The first design parameter is the set of codes that will be used in the system. To be more specific, the value of p, the probability of a particular transmitter asserting a logic-1 value, should be investigated to determine its usefulness in the reduction of bit error rate. For determining an optimal value, research in optical CDMA can be reviewed because of the similar channel models.

The second design parameter is the receiver algorithm to recover both in the presence and absence of the co-channel interference. A naïve algorithm could be implemented to mimic a wireless CDMA receiver, but ultimately new optimizations will have to be made that are specific to this particular design problem. This design parameter is the primary concern of this research and is explored in detail.

The third and final design parameter is that of code length, N. By increasing the code length, the number of users that can be handled by the system is increased. This research also considers the effect of increasing code length on the number of simultaneous transmissions that can be handled by the system and the corresponding bit error rate.

Conclusion

The system requirements of a wired CDMA system were motivated and presented. The need for the use of a binary channel was explained and a wired-OR binary channel was introduced. The simple system model for the 1-Wire CDMA system was given and each system component was explained. Finally, the key design parameters of the system were introduced. With a formal system model and knowledge of the key design parameters, a detailed system design can take place. In the following chapters, the detailed system design will be explained and then taken through simulation and hardware investigation.

CHAPTER III

WIRED CDMA PHYSICAL LAYER SYSTEM DESIGN

In this chapter, the design of the 1-Wire CDMA Communication System is discussed. Each system component is described and diagrams are given where appropriate. Since more than one approach could be taken in the final system design, two different options are described in this chapter. First, a system where codes are naively designed just as in a wireless CDMA system is described. Next, a system where codes are designed to take advantage of the Z-Channel by adjusting code weight is considered.

Transmitter Design

In the proposed system, the transmitter sends a codeword, $c_{x,y}$ for each message bit, where x is the index of the code to be used, and is determined by either the Data Link Layer or the Network Layer, and y is the value of the message bit (bits of the codeword will be referred to as chipping bits). An example of an equally weighted $c_{x,y}$ of length 8 could be *b10010101*. An individual message chip at position i will be denoted by $c_{x,y}^{(i)}$. The length of the message is irrelevant because each bit will be processed independently and each code will be assumed to have identical properties. Furthermore, because codes will be pseudorandom in nature with a predefined probability of a logic-1 or logic-0 occurring in each code chip, the content of the message does not matter either. The length of the codeword will be N chipping bits. Each of the data chips is sent into the channel through a wired-OR circuit. Thus, the output of one transmitter is ORed with the outputs of all other transmitters on the bus at the same time instant. The output of all other transmitters is modeled as a co-channel interference signal r.

Design Parameter: Codeword Weight

The chips of the code are pseudorandom and thus can be assigned a weighting factor to affect the probability of logic-1s and logic-0s. This weighting will be described

in two weights in this thesis. A percentage or decimal number will be given to describe the percentage of logic-1s out of the total codeword or a code will be referred to as weight *n* where *n* is the number of logic-1s that appear in a codeword. If codes are equally weighted, they will contain an equal number of logic-1s and logic-0s, similar to codes in wireless implementations of CDMA. One advantage of equally weighted codes is that a 0-bit code can be made the compliment of a 1-bit code while still maintaining the weight factor. By making the codes opposites, the receiver hardware can be implemented more efficiently when determining whether a message is intended for the receiver.

Co-Channel Interference Design

Because the co-channel interference signal is the OR of the output of all other transmitters using the channel, it can be indirectly designed through the codeword weight factor. For any instant of time, the co-channel interference will be a function of the system's chosen code weight and the number of users transmitting at that time. This signal will be the chip-wise OR of *M-1* transmitter outputs, where *M* is the number of active transmitters in the system. It is also important to note that when the co-channel interference is logic-0 valued, it does not affect the message signal.

The co-channel interference signal will be logic-0 when all *M-1* transmitters are asserting a logic-0 on the bus. The co-channel interference signal will be logic-1 valued when any of the *M-1* transmitters are asserting a logic-1 on the bus.

Probability of *1→0* Chip Error

As discussed previously, the probability of a *1→0* Chip Error is zero when hardware and software errors are not considered. Even when applied to real hardware, a *1→0* error would be extremely unlikely and only caused by clock-drift problems.

Probability of *0→1* Chip Error

The probability of a *0→1* error is a function of the system's chosen code weight and the number of users transmitting at that time.

Receiver Design

Receiver Scheme for Equally Weighted Codes

One obvious scheme for decoding messages would be to design a receiver that mimics a receiver in a wireless CDMA system. This is an especially useful topology in the case where code weight is equal. To accomplish this, a shift register of length *N* is implemented and the incoming channel bits are shifted into the register at each time instant. The received bits could be XORed with the relevant codeword to form a correlation vector. The correlation vector can then be used to calculate the Hamming distance between the two vectors.

For binary vectors of length *N*, the correlation will be between 0 and *N*. A correlation of *N*/2 indicates that the vectors are completed uncorrelated; in other words, the receiver should reject the signal because it is not a message to be received. If the correlation is 0, the codeword and the message signal are identical and thus a '1' bit has been decoded. If the correlation is *N*, the codeword and the message signal are opposite and thus a '0' bit has been decoded.

In order to provide some degree of noise immunity, the receiver could include a threshold concept as shown in Figure 8. If the correlation is less than 0 plus the threshold value, the signals are nearly identical and thus there is a high probability that a '1' bit has been decoded. If the correlation is greater than *N* minus the threshold value, then there is a high probability that a '0' bit has been decoded.

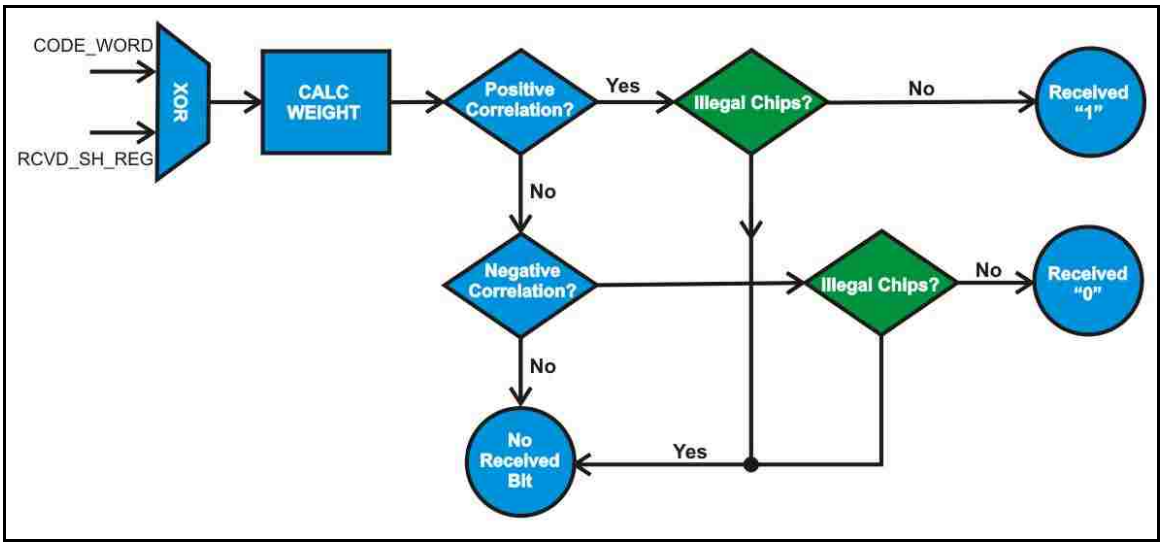

Figure 7. Basic Receiver Block Diagram for Equally Weighted Codes. This algorithm is very similar to the basic block diagram of a wireless CDMA decoder. The *Illegal Chips?* block searches for any logic-0s in slots where a code-word expects a logic-1, this allows the receiver to reject potential codewords that meet the correlation requirement but do not satisfy the condition of no *1→0* chip errors.

Finally, the receiver could provide additional noise immunity by implementing an algorithm to detect "illegal" message chips. For instance, imagine the transmitter codeword is (1010). At the receiver, the message (1000) is received. The XOR operation is performed on these two vectors is (0010). The Hamming weight of this result is 1. If the system correlation threshold offset is 1, this vector will be decoded as a 0 bit sent from the transmitter. However, since the only type of channel error assumed possible is *0→1*, the received word (1000) could not correspond to the code word (1010) since a *1→0* error is implied in the third chip. This concept takes advantage the unidirectional nature of errors in the channel.

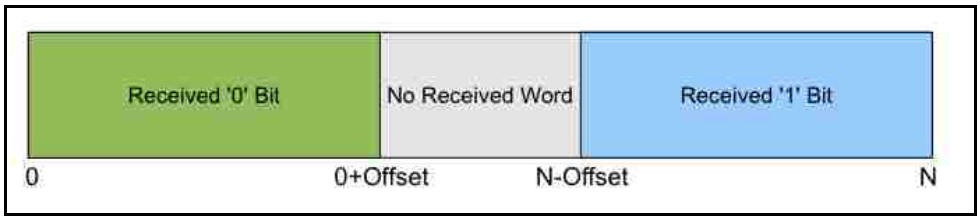

Figure 8. Decoder Threshold Concept Illustrated. The basic receiver for equally weighted codes contains a threshold setting that allows the receiver to make intelligent decisions about what was received, if anything. Three decision regions are created by using an offset value, which in practice could be different for the logic-1 and logic-0 regions, and the receiver decodes inputs based on which of these three decision regions the correlation of the code and the received vector falls into.

Although the scheme described here has merit, it is clear that limitations exist for even a smaller number of channel sharers. For an increasing number of transmitters, the probability of chip errors increases exponentially. Intuitively, it is clear that reducing average codeword weight will increase the number of transmitters that can share the channel before error rates grow out of control. Reducing the codeword weight also reduces the number of codes available, so balancing these two parameters is important. This concept is supported in the literature on optical CDMA communications, including [26,32], and supported by the figures earlier in this section. The exact probability does not need to be known in order to perform an optimal design of the receiver, it will be enough to know that there will be significantly fewer logic-1s than logic-0s in each code.

Receiver Scheme for Unequally Weighted Codes

In an unequally weighted scheme where far more logic-0s are used than logic-1s in each code, the logic-0s can be seen as carrying no information, except perhaps about channel utilization. For this reason, the basic receiver algorithm should not perform XOR to calculate correlation between codes and the contents of the receive shift register. Instead, an AND operation is used. If the vector resulting from this AND operation is equal to the code, the received signal is initially considered a match.

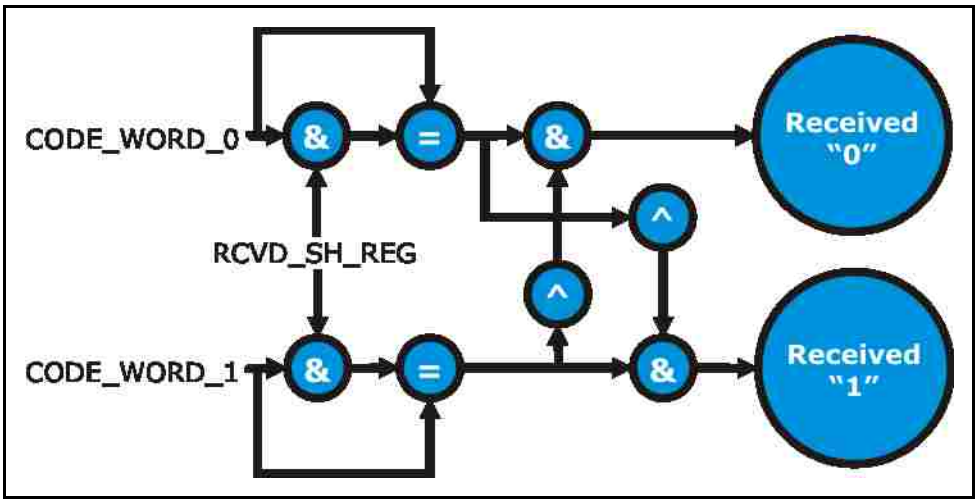

Figure 9. Receiver Algorithm for Low Weight Codes. This algorithm is very different from the basic block diagram of a wireless CDMA decoder. The correlation is essentially checked by the logical-AND operation. This block verifies that all logic-1s are present in their expected locations. The receiver prevents a 1-bit and 0-bit from being received simultaneously. This architecture inherently allows the receiver to reject potential codewords that do not satisfy the condition of no *1→0* chip errors and to ignore all other logic-0s.

In this architecture it is necessary to have a separate code for transmitting a 1 and a 0 for each user. Although this is likely to cause inefficiency in the software of a practical receiver, it can be used to reduce the amount of incorrectly decoded bits received. If the receiver decodes both a 1 and a 0 in any given cycle, the data can be marked and dealt with accordingly. For instance, the bit could be discarded by assuming that this unlikely event is occurring due to a channel that is saturated with transmitter outputs. Alternatively, the value could be randomly assigned and left to error correction and detection of a higher networking layer, the value could be returned as in contention and assigned by a higher layer deliberately based on the type of data being sent, or an *Automatic Repeat Query* (ARQ) scheme could be initiated. For a general reference on ARQ see [4]. It is important to note that if any re-transmission scheme, such as ARQ, is to be used, it would be necessary to include some pseudo-random delay or channel-listen mechanism to wait for the channel to become unsaturated by transmitters.

Identifying the channel utilization would also be a useful characteristic. The receiver could correlate the incoming data stream with the codeword to compute the distance in order to get an idea of how much co-channel interference is present in the channel. Alternatively, the weight of the receive shift register could be calculated. If the receive shift register is large, there is a higher probability that errors have occurred in decoding. This could be used to mark bits as potential error bits, so that a higher networking layer could use an intelligent error correction scheme to make better guesses about which, if any, bits are in error.

There are many other receiver optimizations that could reduce the end user bit error rate, but are not easily quantifiable. For this reason, further optimizations will be discussed in sections on hardware design and will not be applied to the model given here.

Conclusion

A high-level system design has been described, with a focus on the Physical Layer of the networking stack. The basic transmitter scheme was developed with a specific goal of similarity to wireless CDMA systems. Minor changes were made to the transmitter scheme to allow for design of the co-channel interference by changing code weights. Two basic receiver designs were given and explained. In the end, a practical implementation of the system must strive to optimize three key parameters: receiver topology, codeword length, and codeword weight and orthogonality. The remainder of this thesis will focus on making suitable decisions for these parameters so that a working hardware prototype can be implemented.

CHAPTER IV

CODE LIBRARY DESIGN

In order to successfully implement the transceiver system with minimal bit error rate, it is desirable to build a code library or codebook with optimal or suboptimal codes, rather than completely random codes. In this chapter, the similarity to incoherent optical channels that was discussed in the introduction is exploited and research in Optical Orthogonal Codes (OOC) is drawn on. Because their autocorrelation and cross correlation are generally limited to 1, these codes are considered optimal for a given weight when used with a Z-Channel. Because of these strict requirements, very few of these codes are available for a code length that could be reasonably implemented on low-cost hardware. For example, only five 32-bit optical codes are available. Codes any longer than this become difficult to implement on a typical microcontroller. Fortunately, optimal codes are not necessary to achieve acceptable performance, but instead can be extended to suit the needs of the system. Suboptimal codes can be derived by relaxing the requirements of a typical OOC set. In this chapter, goals of the code library design problem are discussed, general-purpose code library design software that has been developed is explained, a set of OOCs are also discussed and, finally, an example code library is developed to satisfy certain implementations of the transceiver.

Goals

In the code library design problem, the primary goal is to design a set of codes that are maximally orthogonal. This design goal is somewhat unique for the Z-channel. Because there is no frame synchronization in the system, the problem becomes even more difficult to solve because the set of codes must be maximally orthogonal with a 0 to N-chip time delay applied to either code. In this case, N is the number of chips in the code. To state this problem another way, each code must be considered against rotations of all other codes in the codebook.

From earlier chapters, it is evident that low weight optical codes could be the basis for any code library in this system. In order to gauge the performance of a code library on this metric, the average Hamming weight per code word can be calculated.

Code Library Design Software

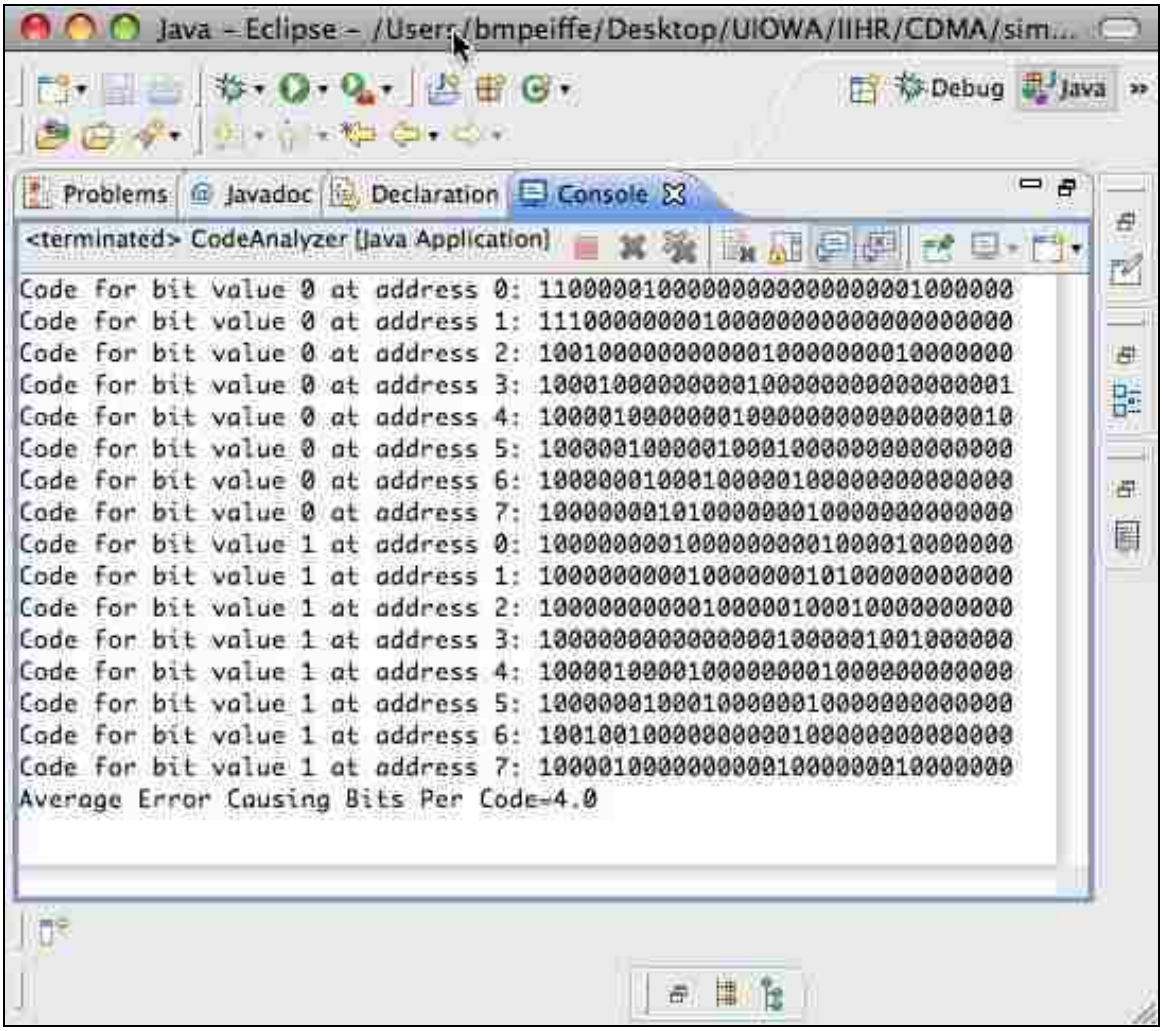

Figure 10. Code Design Software, Codes Meet Requirements. This screenshot shows how the code design software gives the user feedback to indicate that all codes are correctly designed. The software shows each code vector that has been designed. The software also shows the average codeword weight.

In order to aid in calculating the orthogonality and average Hamming weight of a code library, Code Library Design Software was written in the Java Programming Language. The software takes in a codebook size, a peak correlation requirement, and the proposed set of codes. The codes are then iteratively compared to one another and the Hamming distance of each pair is calculated. If the Hamming distance between a pair of codes is below the threshold value, the pair is reported to the user along with the actual distance between the codes. These pairs of codes are recommended for redesign. During the sequence of distance calculations, the software also accounts for delayed versions of one code with respect to the other. The software also reports the average Hamming weight of a codeword. This calculation is performed by counting the number of logic-1s in each code and adding it to a running total, then finally dividing by the total number of codes in the library.

Optical Orthogonal Codes

In general, the CDMA code design problem for the wired-OR channel is very similar to the design problem presented in optical communications systems. For some design parameters, sets of OOCs have been developed and presented in various studies including [6,10,13,19]. Ideally, a set of 32-bit OOCs with 16 codes (2 for each of 8 users) would exist. 32-bit codes would be inexpensive to implement on a modern microprocessor or microcontroller and support for 8 users would be adequate in the target application. Unfortunately, this is not the case; the strict correlation requirements of the codes prevent so many codes from being formed with such a short code length. This problem is easily solved by relaxing the correlation requirements of the codes. By using a set of previously developed OOCs as the basis for a codebook for the wired-OR system, the design problem is greatly simplified.

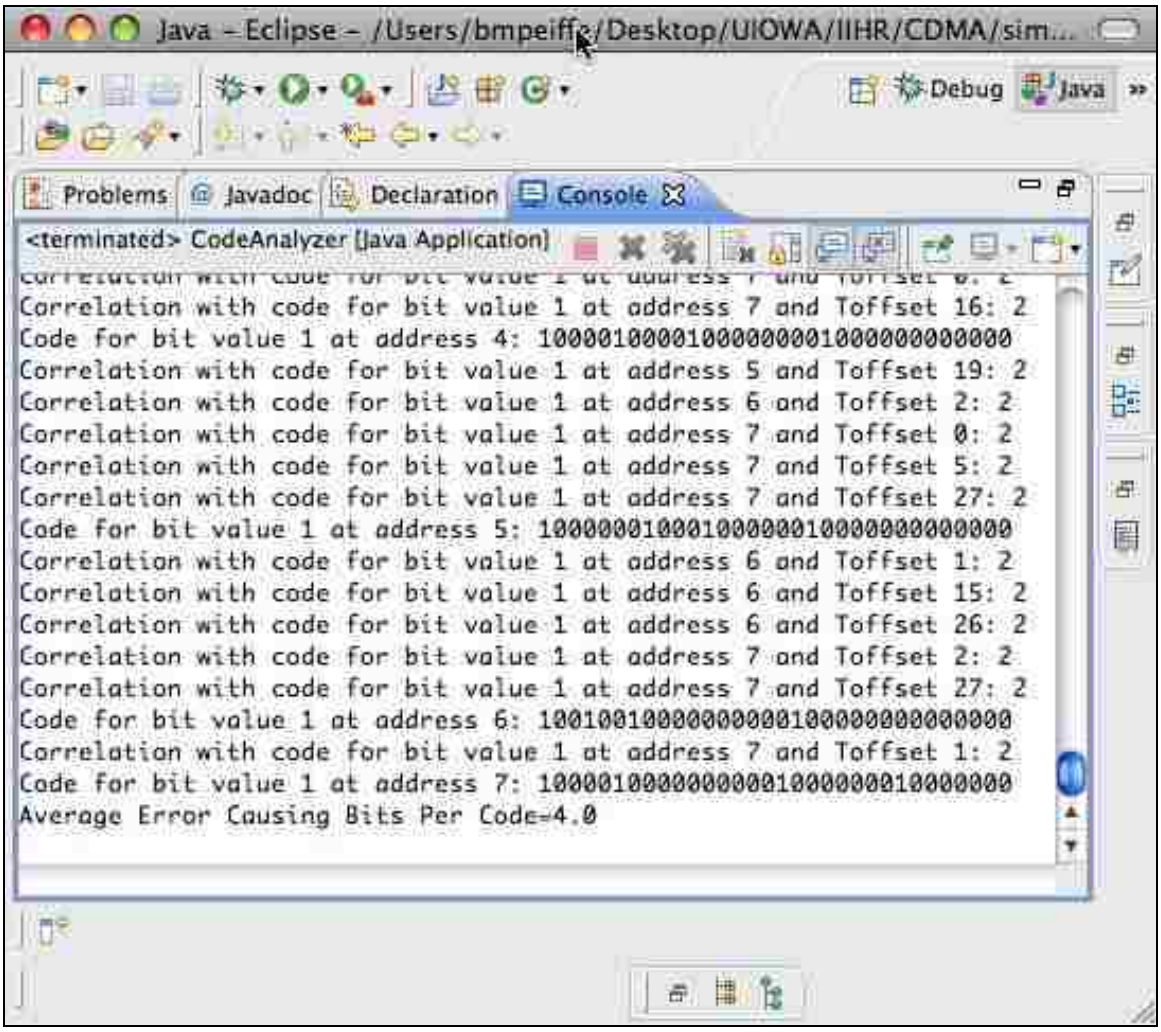

Figure 11. Code Design Software, Codes Do Not Meet All Requirements. This screenshot shows how the code design software gives the user feedback to indicate that some codes are not correctly designed. The software gives feedback about which two codes are in conflict, what relative rotation of those codes is in conflict, and what the correlation at that position is (to give an indication of how badly the codes are in conflict).

Code Library Design Example with 16 Codes of Length 32

In order to show the simplicity of the Code Library Design problem when the proper tools are used, an example is given here. This example codebook will then be used as the basis for later system implementations.

Optimal (31,3,1,1) Code
{0,1,7}
{0,2,11}
{0,3,15}
{0,4,14}
{0,5,13}

Table 1. Optimal (31,3,1,1) Codes from [13] Used in Code Design

First, an optical orthogonal codebook is taken from [13]. The codebook is specified for (31,3,1,1) codes and shown in Table 1. The (31,3,1,1) notation specifies (code length, code weight, peak cross-correlation, peak autocorrelation). The length of each code is 31 bits, which can be extended to 32-bits by adding a trailing 0 to the end of each vector. The weight of each code is 3, meaning that the codebook optimal average weight is 3 bits. The peak cross-correlation is 1, meaning that a maximum of one logic-1 can be in overlap between two vectors and their delayed versions. The peak auto-correlation is also 1, meaning that a maximum of one logic-1 can be in overlap between a vector and any of its own delayed versions.

This codebook provides 5 of the necessary 16 codes. The codes given in Table 1 are in the format $\{a,b,c\}$, where a is the position of the first logic-1, b is the position of the second logic-1, and so-on until all logic-1s for the code have been specified. This is a more compact notation than fully specifying a binary string. For instance, codeword {0,5,13} is equivalent to the binary string specified by b1000010000000100000000000000000. In order to build the codebook the maximum limits on auto-correlation and cross-correlation can be increased and the average code weight can be increased beyond 3 by adding weight-4 codes.

User Index	0-bit Code	1-bit Code
0	{0,1,7,25}	{0,9,19,24}
1	{0,1,2,11}	{0,10,18,20}
2	{0,3,15,24}	{0,11,17,21}
3	{0,4,14,31}	{0,16,22,25}
4	{0,5,13,30}	{0,4,10,19}
5	{0,6,12,16}	{0,7,11,18}
6	{0,7,11,17}	{0,3,6,17}
7	{0,8,10,18}	{0,5,16,24}

Table 2. (32,4,2,2) Code Library

For this system, it seems reasonable to attempt to design a (32,4,2,2) codebook. In order to do this, the optimal (31,3,1,1) codes can first be entered into the code design software. The peak correlation value can also be set to 2, the same as the auto and cross-correlation requirement. If all the remaining vectors are entered as 0x00 and a code check is run, the optimal codes can be verified against one another. The remaining codes can be designed by first entering a logic-1 in the first position. This will help the designer to avoid designing new codes as rotations of the existing codes. One simple approach to the design problem is to perform a guess and check while following the pattern of the optimal codes. In order to do this, a single additional logic-1 must be added to each code that is added. In this way, a code with logic-1s at positions {0,6,12,16} can be added. This code was formed by taking the code {0,5,13}, shifting the position 5 bit up one and the position 13 bit down one, and finally adding a bit to prevent the new code from being a rotation of another code. Although there are more

precise design strategies, this one proves sufficient and provides a codebook with good properties for a 16-code application. The completed codebook is shown in Table 2.

Conclusion

The code library design problem has been briefly discussed. General goals for the design problem were explained and a Code Design Software package that was developed was shown and its application explained. Optical Orthogonal Codes were revisited and their usefulness in the wired-OR channel was also explained. A codebook design example and general design procedure were given and worked through with a set of optimal optical orthogonal codes as a basis. The codes discussed in the example were given in full and will be utilized in future implementations of the wired-OR CDMA system.

CHAPTER V

BINARY CHANNEL CDMA SIMULATION ENGINE

In this chapter, the simulation engine that was designed to test the 1-Wire CDMA system design is discussed. This includes a discussion of the motivation and high-level design for such a software package, the Java implementation of the software, and the experimental results that were gained by using the software package.

Goals

The simulation engine serves two primary purposes. First, the simulator could be used to verify the system architecture. Second, the simulator could be used to test proposed optimizations to the scheme, such as certain coding and receiver designs.

In order to satisfy the first goal, it is important to allow a varying number of users, codeword lengths, and message lengths utilize the system. In order to satisfy the second goal, it is important to allow a varying of important receiver parameters and important codeword parameters. Two such parameters are decoding threshold offset and codeword weight. The simulator was also designed to allow custom code sets to be loaded into the software from a file. This would allow for the code designer to simulate the system with their codes. The simulator receiver architecture should also be easy to reprogram for experimentation with the receiver algorithm.

Design

The simulator was designed using the Java programming language. The simulator was designed to report three important metrics to the user: number of errors, bit error rate, and number of dropped bits.

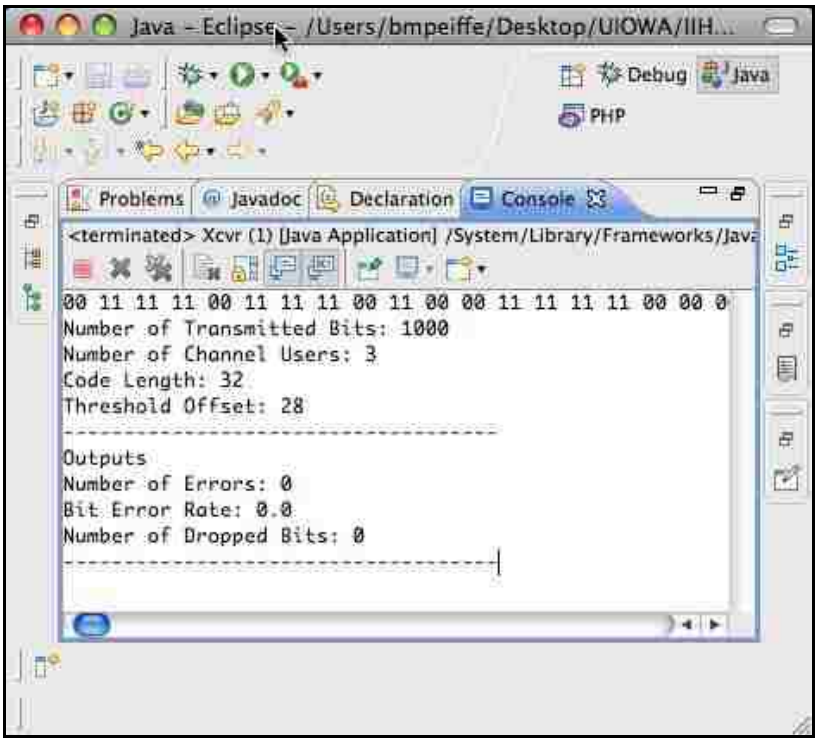

Figure 12. Simulator Inputs and Outputs. This screenshot shows the console output of the simulator that was designed. In the image, the simulator has been used to run a simulation for 3 channel users with 32-bit codes. The users have all been asked to send 1000 bits, which is essentially the interval over which to average for the chosen set of codewords. The simulator outputs the raw number of errors and the raw number of dropped bits, along with the bit error rate for the simulation run.

The simulator design contains four classes, described here:

1. *CDMA Transceiver*

   The class *CDMA Transceiver* represents one message node. It is assigned two codes, which can be either random or loaded in from a file, and a random message of the required length. The *CDMA Transceiver* was built to allow for easy instantiation of an arbitrary number of channel user objects. The *CDMA Transceiver* contains object instances of the *Message Encoder* and *Message Decoder*.

2. *Message Encoder*

The class *Message Encoder* encodes a given message with a given codeword. For each message bit, the encoder selects the appropriate code word for sequential transmission. The *Message Encoder* simulates the transmitter design given in previous chapters.

3. *Binary Channel Model*

The class *Binary Channel* takes in either an arbitrary number of Transceiver objects or an arbitrary number of message arrays and performs a bitwise OR operation on them. It then outputs an array representing the time series that the channel output would see.

4. *Message Decoder*

The class *Message Decoder* implements a decoding algorithm. During the simulator development, two different designs for the *Message Decoder* were implemented, one for each of the receiver algorithms that was previously discussed. It returns the decoded message for post processing.

Simulation Results

Equally Weighted Codes

The simulator has been used to investigate the properties of the system. The code length, threshold offset and number of simultaneous users have been varied to gain an understanding of the system capacity. The outcomes of two different test cases are shown in Figure 13 and Figure 14. Both of these test cases are for randomly generated code words with no weighting adjustment, i.e. probability of a logic-1 is 50%. Figure 13 shows simulation data for a 32-bit codeword length, the same code length as the hardware implementations that will be discussed in later chapters, and a varying number of users. The 32-bit code length was chosen here, and in the following chapters, because it is realistic to think that the 32-bit codes could be implemented on reasonably affordable

hardware, such as a 32-bit microprocessor or even an 8-bit microcontroller. The vertical axis shows the probability of error. In this case, the probability of error refers to both the number of incorrectly decoded bits and the number of dropped bits. As shown in Figure 12, the system also records the number of dropped bits, so this could be subtracted from the BER to gain an idea of the system capabilities if ARQ functionality was added. For the 32-bit codeword test case, the system maintains 0% BER up to 2 simultaneous users. The system has reasonably low BER up to 3 users, but at this point it would be useful to implement channel coding. Beyond 3 users, the error rate increases exponentially and it would be necessary to implement ARQ and error coding.

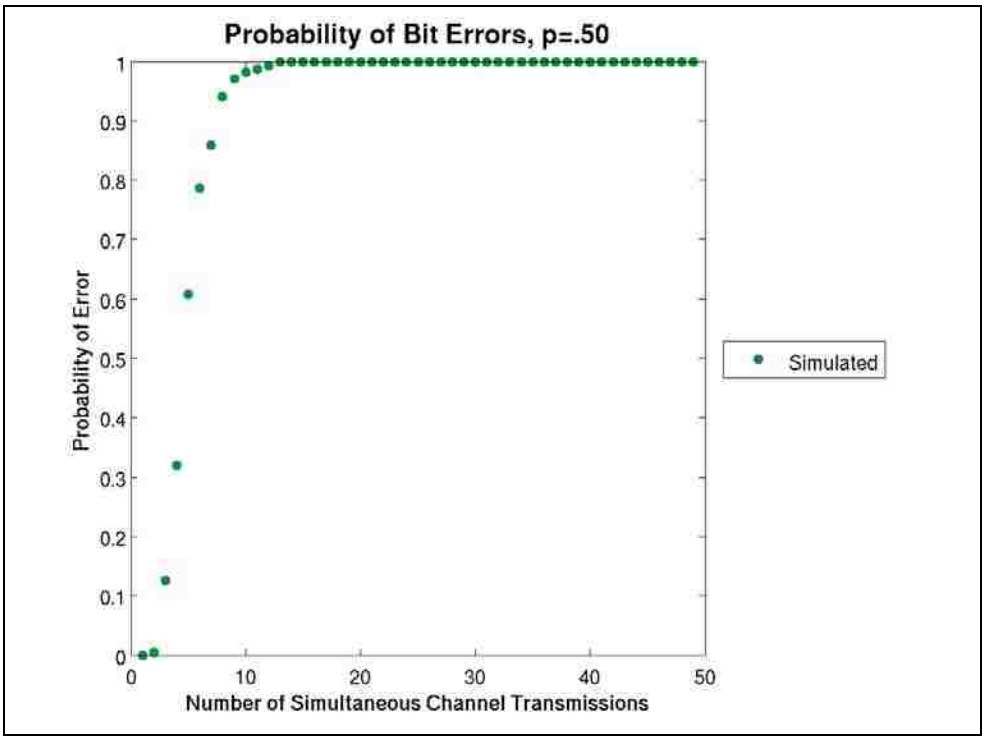

Figure 13. Average Probability of Bit Errors, Including Dropped Bits, From 50 Trials with length 32, weight 16 codes (i.e., N=32, p=.5). Codes for each of the 50 trials were generated randomly by the simulator at run-time.

Figure 14 shows a test case for 1024-bit codeword length. In this case, up to 6 users can simultaneously share the channel with a 0% BER. As the number of users increases, it again becomes clear that ARQ and error coding are necessary.

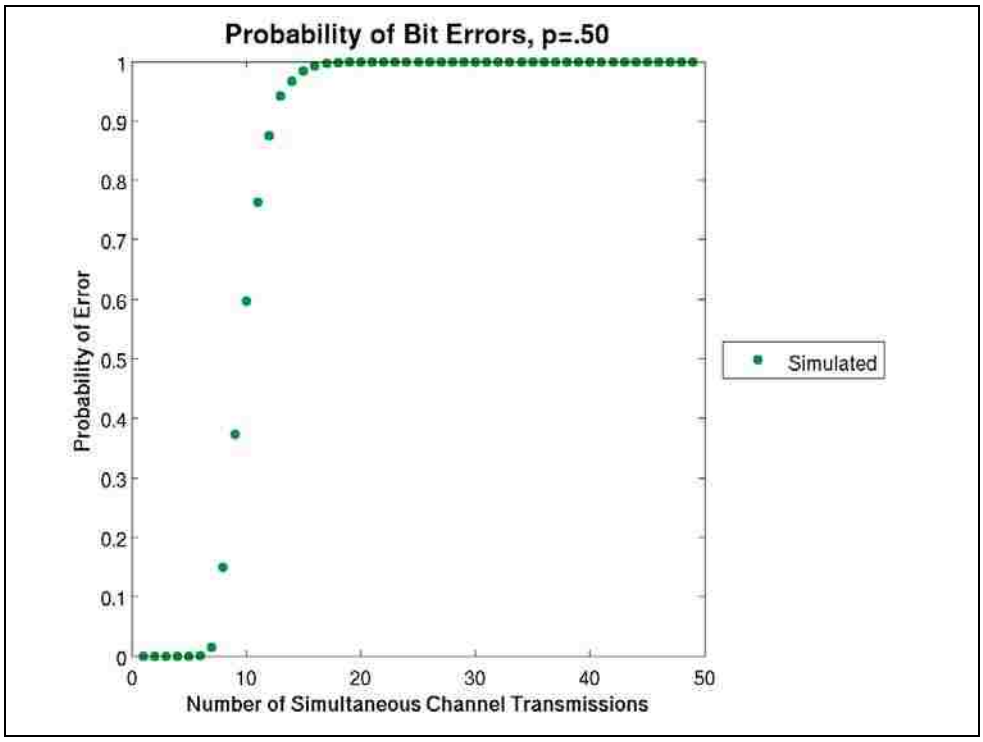

Figure 14. Average Probability of Bit Errors, Including Dropped Bits, From 50 Trials with length 1024, weight 512 codes (i.e., N=1024, p=.5). Codes for each of the 50 trials were generated randomly by the simulator at run-time.

Low Weight Codes

The data for two test cases with low weight codes were also captured. In these test cases, the probability of a logic-1 in the codeword was modified to less than 50%. This is much closer to how the actual system should be designed, based on previous work in optical communications.

The first test case is for a 1024-bit codeword length with the probability of a logic-1 in the codeword set to 10%. For this case, an automated program was run to test

the system with up to 170 users. This was done mainly to show the large difference in rate of increase of bit error rate for increasing number of users with lower logic-1 probability. Up to 24 users are able to share the channel simultaneously with virtually no errors. Up to 40 users are able to share the channel with a less than 10% bit error rate, suitable for an error-detection-and-correction scheme. For more than 40 users, it would be necessary to implement ARQ and the system probably will not support more than 60 users.

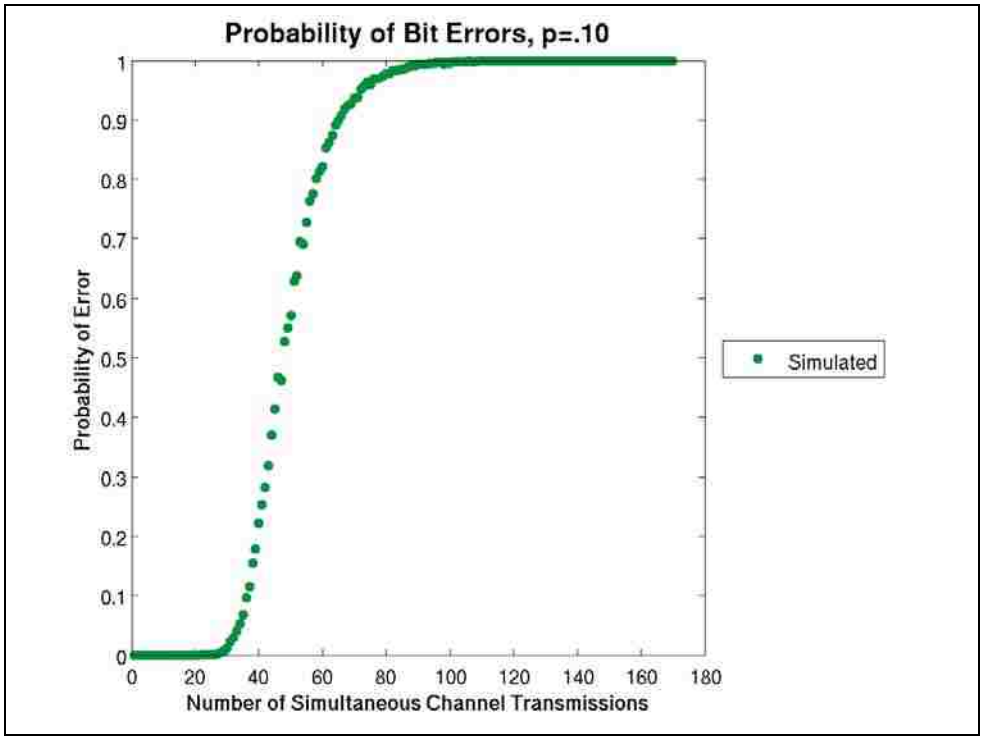

Figure 15. Average Probability of Bit Errors, Including Dropped Bits, From 50 Trials with length 1024, weight 102 codes (i.e., N=1024, p=.1). Codes for each of the 50 trials were generated randomly by the simulator at run-time.

The second test case is for a 32-bit codeword length, the same codeword length that will be used for basic microcontroller implementations of the bus that will be discussed later in this thesis. For this test case, the probability of a logic-1 in the

codeword is set to 12.5%. This choice was made to provide a direct comparison to the codebook that was previously developed, which had length 32, weight 4 codes (probability of a logic-1 equal to 12.5%). An automated program was run to test the system with up to 49 users. Again, this number of users helps to illustrate the difference in bit error rate for increasing number of users with low logic-1 probability. Up to 6 users were able to share the channel with virtually no errors. Up to 10 users are able to share the channel with around a 15% bit error rate, suitable for an error-detection-and-correction scheme. For more than 10 users, it would be necessary to implement ARQ and the system will probably not support more than 25 users.

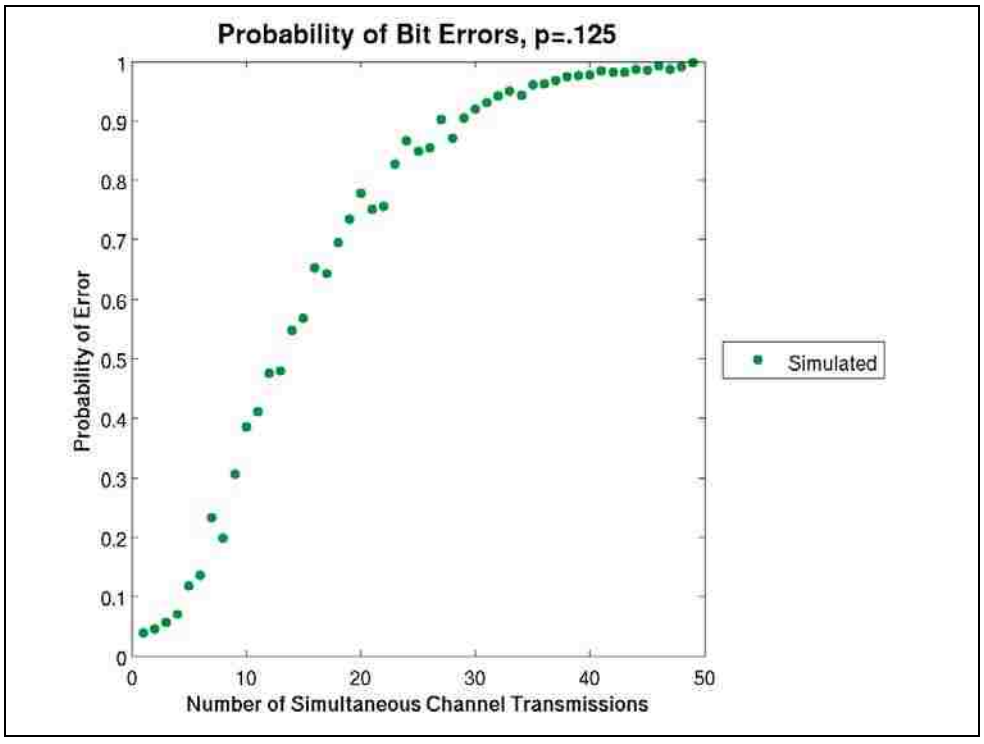

Figure 16. Average Probability of Bit Errors, Including Dropped Bits, From 50 Trials with length 32, weight 4 codes (i.e., N=32, p=.125). Codes for each of the 50 trials were generated randomly by the simulator at run-time. These results track somewhat closely with the results calculated from the probability functions.

It is also important to note that, in this configuration, the number of available codes is very small compared to previous simulations, so this also limits the system performance. The small number of available codes causes both code word collisions and unacceptably low distance between the randomly generated codes. For the majority of simulation runs, this does not cause any problems and simulations run normally. However, in somewhat rare cases, catastrophic errors occur and bit error rate will spike to nearly 100%. The large averaging interval minimizes this effect, but it is still noticeable when comparing the graphs of this simulation scenario to those previously shown. Because of these errors, it is desirable to run the simulator with actual code sets, instead of randomly generated ones, to get a better idea of system performance with length 32, low weight codes.

Sub-Optimal Codes Extended from Optical Orthogonal Codes

By using a set of non-random codes as input to the simulator, the simulator can model system performance more practically. In the previous chapter, the code library design problem was discussed. An example codebook was developed and analyzed. Here, the example codebook is used as input to the simulator system. The codes from the codebook are sub-optimal in nature, but were generated from a set of optimal optical orthogonal codes. The primary difference between the optimal OOCs and the sub-optimal codes is the peak correlation values; the optimal OOCs have peak cross and autocorrelation of 1, while the sub-optimal codes have peak cross and autocorrelation of 2.

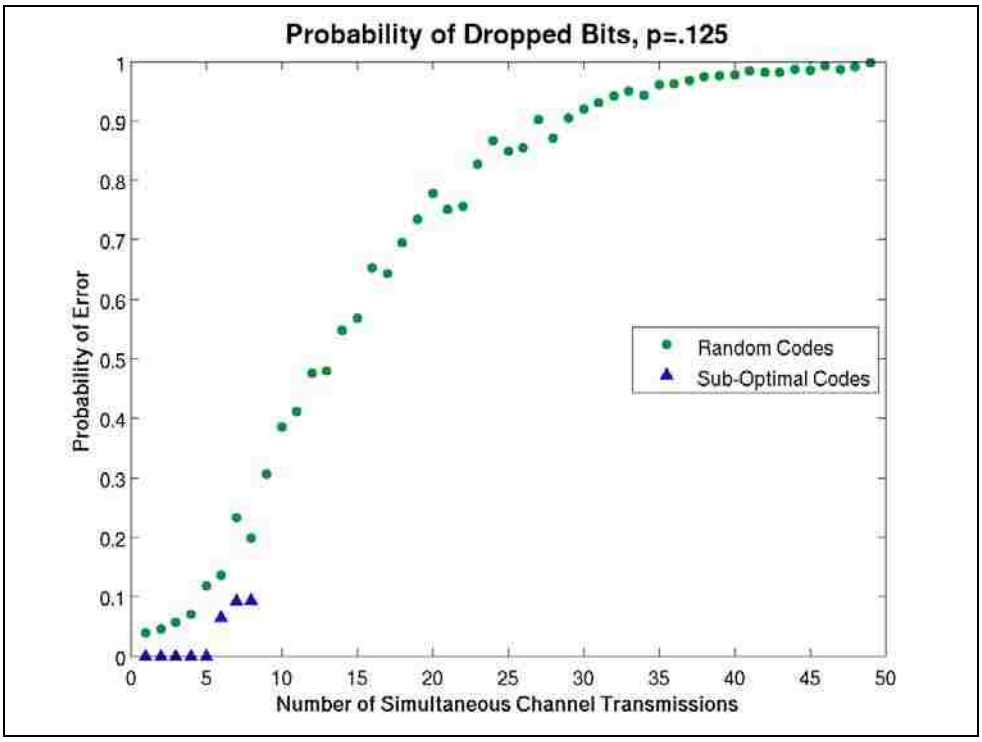

Figure 17. Average Probability of Bit Errors, Including Dropped Bits, From 50 Trials with a set of sub-optimal codes with length 32, weight 4 (i.e., N=32, p=.125). The codes used here were those generated in the chapter on Code Library Design. Since a limited number of sub-optimal codes can be generated from any set of requirements, only the 8-designed codes were plotted here.

This library can be used as a comparison to the particular case of the 32-bit, 12.5% error-chip probability codes used in the previous section. The example codebook has approximately 12.5% error-chip probability, i.e. the codes are weight 4. With the sub-optimal codebook, all 8 users could simultaneously share the channel with approximately 10% error rate. Although the simulator has the advantage of frame synchronization, the real system should behave in a very similar way.

Simulator Compared to Set of OOCs

The simulator was also used to compare the performance of a set of random codes to a set of OOCs. This test verifies that the receiver algorithm will perform better with best-case codes than with random codes. For this test, a feature was added to the

simulator to perform random rotations of each codeword. The (217,4,1,1) OOCs that were chosen from [32] support a limited number of users. Figure 18 shows that the performance tracks very closely until too many users are added to the bus, at which point the performance of the random codes diverge.

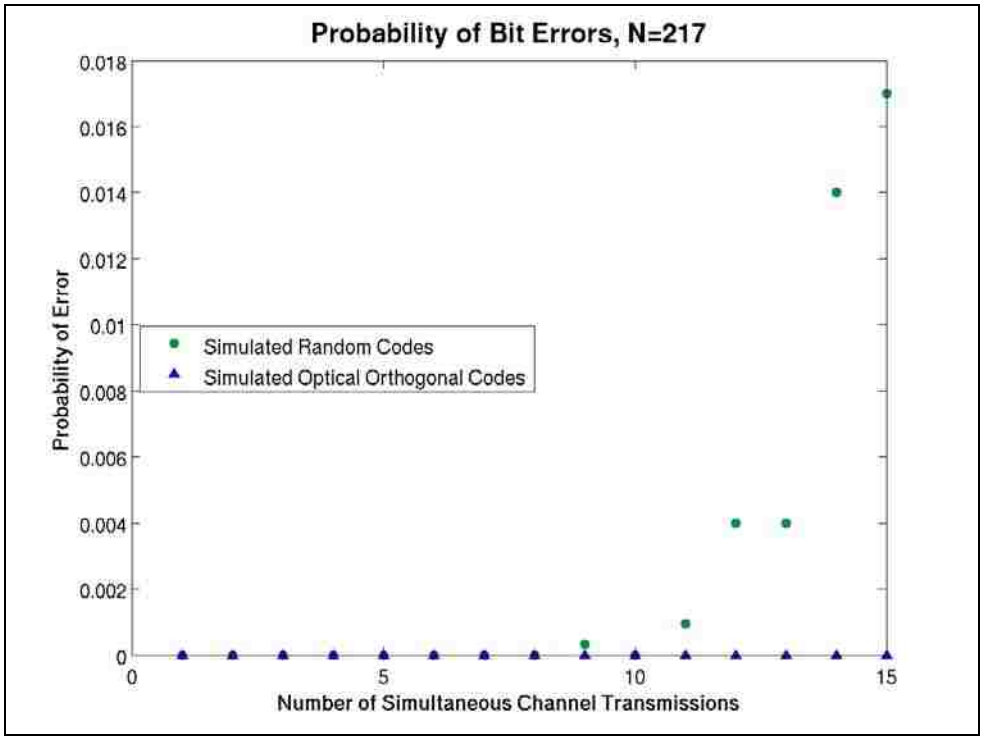

Figure 18. Average Probability of Bit Errors, Including Dropped Bits, From 50 Trials with length 217, weight 4 codes (i.e., N=217, p=.02). One set of codes for each of the 50 trials were generated randomly by the simulator at run-time, the other set of codes was a group of OOCs from [32]. These results track closely until too many users are added to the bus, at which point the performance diverges. The (217,4,1,1) OOCs that were chosen support a limited number of users.

Conclusion

A robust simulation engine has been developed and discussed. The simulator was used to verify the system architecture and shows that the system will work effectively for a small number of users with the most naive implementation and can also be made to

work with a significantly larger number of users under certain circumstances, such as increasing the code length or using better code words. The simulator was also used to verify that a lower probability of logic-1s, which is also the information chip, will yield significantly greater system capacity. The simulator was also used to test a particular codebook.

CHAPTER VI

BOARD LEVEL HARDWARE IMPLEMENTATIONS

In order to prove the practicality of the CDMA Bus system design in real hardware, a board level hardware design has been developed. The hardware was designed to allow experimentation and investigation of the bit error rate of message signals being passed through the system with additional noise degradation from real hardware components. The system supports up to 8 users, has a 32-bit codeword length, and contains a software adjustable decoder threshold level. After detailed use of the board level hardware, the results of the simulation engine experiments can be empirically verified.

Goals

The board level hardware design served two primary purposes. First, the hardware could be used to verify the practicality of the system architecture. Although the system was successfully simulated, it was necessary to prove that adequate speeds could be achieved with low-cost hardware. Second, the hardware could serve as a future platform for actual sensor deployments, especially in cases where larger channel bandwidth may be required.

In order to satisfy the first goal, it is important to allow for a reasonable number of users to utilize the system at an adequate data rate. Thus, the system should support at least 8 users at a rate of 100 bps. In order to satisfy the second goal, it is important to provide a reasonable host interface, small size, and maintain low cost.

Design

The basic design includes two logical Digital Signal Processing (DSP) units, one for the receive path and one for the transmit path. These logical DSPs can be separated in hardware or software, but they are two distinct processing paths. Aside from the

processing hardware, any implementation should define a hardware input/output configuration, the shared channel data rate, a code library, and the host software interface.

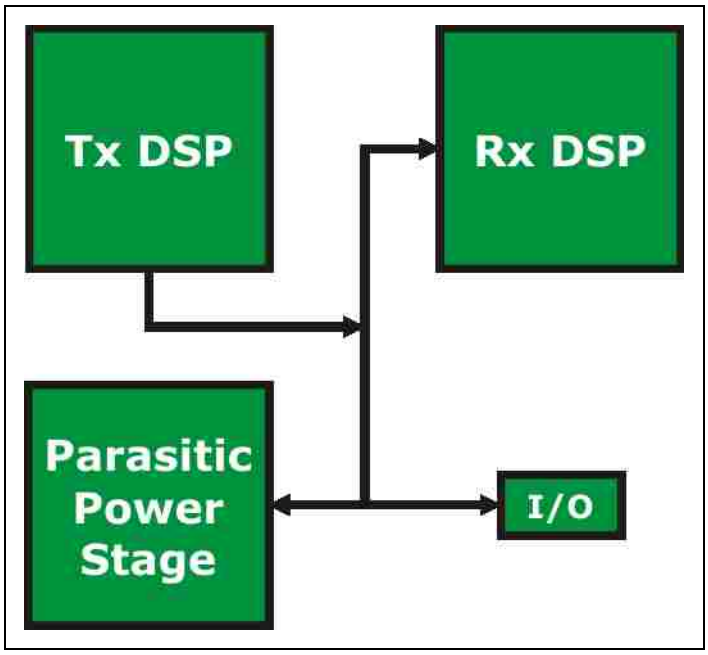

Figure 19. High Level Hardware Block Diagram. This diagram shows the logical division of the 4-primary hardware blocks, the I/O, RX and TX path DSP, and an optional parasitic power stage.

Processing Hardware

It is important to note that the logical DSP units could be implemented with one of many types of processing hardware or a combination of processing units. The available hardware includes Field Programmable Gate Arrays (FPGA), fixed point Digital Signal Processors (DSP), high-end General Purpose Processors (GPP), or Microcontrollers (MCU). One could also combine some of these processing units, either by implementing custom hardware or purchasing an off-the-shelf solution such as a Texas Instruments Open Multimedia Application Processor (TI OMAP), which combines

an ARM Core GPP with a TI DSP, among other specialized cores. For this design, a pair of MCUs was chosen in the first iteration to simplify the design process. This will be discussed further in the next section.

Channel Implementation

The supporting hardware for the transmit path must implement the wired-OR channel (although it is acceptable to implement wired-NOR, wired-AND, wired-NAND, etc. as long as appropriate software and codes are present). Most MCU and FPGA devices contain a configuration option for open drain or open collector output pins. In an open collector configuration, for instance, many outputs can be wired together to form a wired-OR. Discrete components could also be used to form the open collector. This would be necessary in the case where a high voltage channel is to be used. If the desired channel voltage is larger than the maximum input voltage of the processing unit inputs, then the wired-OR must be formed with discrete components and must also include level shifting circuits.

Parasitic Power

The receive path hardware can be minimal, but could also include parasitic power circuitry to eliminate the need for a power interconnect between devices. The basic schematic for the parasitic power circuit was borrowed from the work in [27]. The parasitic power circuitry could feed off of the channel voltage, so at least one device would need a power source. A simple method for drawing power from a communication line is to connect the anode of a diode to the communication line and connect the cathode to the positive terminal of a low Equivalent Series Resistance (ESR), high capacitance, tantalum capacitor. The diode prevents discharge of the capacitor back into the channel and the capacitor provides a short-term storage reservoir that could be used to charge a battery or, if a large enough capacitor is used, power the electronics connected to Vo. In Figure 20, this architecture was modeled and parasitic resistance was placed in series

with the capacitor to model a real capacitor with a finite Equivalent Series Resistance. A load was attached to the output and a simulation was run to show that the circuit could hold sufficient output voltage through the full cycle of input voltage.

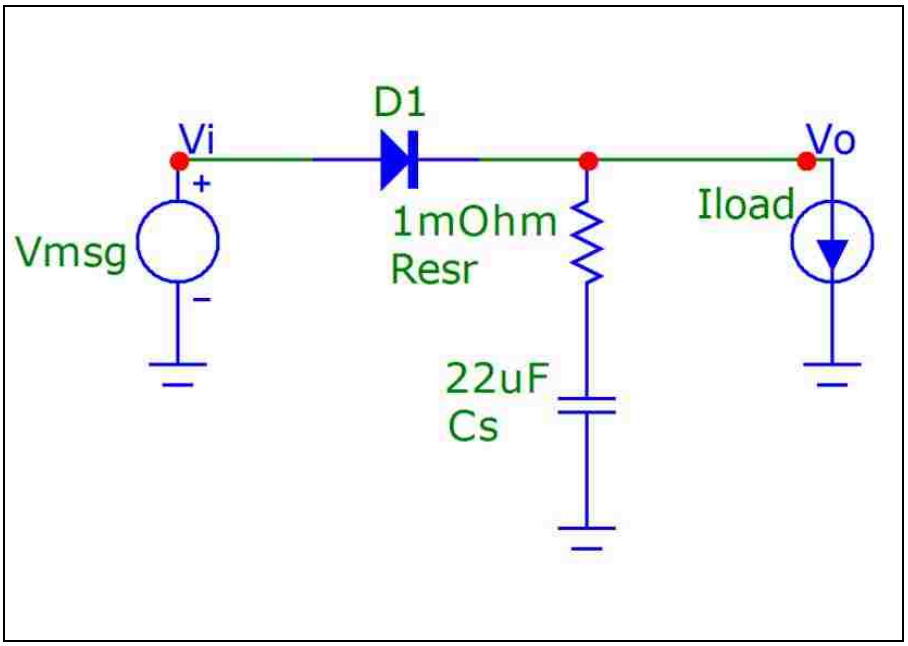

Figure 20. Parasitic Power Concept Model in Micro-Cap SPICE. The parasitic power model was borrowed from the work in [27]. The simulation in SPICE, shown here, included a small *Equivalent Series Resistance* (ESR) on the capacitor and a load that drew a constant current. A fifty percent duty cycle voltage source simulated the typical channel voltage transfer function.

Data Link Layer Implementation Details

Although this thesis is primarily concerned with the implementation of a robust physical layer, it is also necessary to consider some higher level networking layers in order to implement a practical transceiver. In particular, some details of the Data Link Layer must be defined. The frame size is defined as 1-byte or 8-bits. The Data Link Layer implemented in hardware does not contain any Error Detection and Correction or ARQ, but this could be implemented by the host machine if desired. The 8-bit frame

does not contain a stop-bit or start-bit and frames can be asynchronously initiated by any host.

Host Interface

The hardware must also include a host interface communication protocol. This protocol could be UART or I^2C but should adhere to the general user interface given here. The hardware interface for the device should be documented in each implementation, but should contain two lines for the UART or I^2C interface.

In order to send a message, the user sends a minimum of 4 bytes. The first byte is the message start byte, denoted by the ASCII character ‘s’. The second byte is the address of the destination, an integer between 0 and 255. A number of the possible addresses will be invalid and this will be indicated by the return of an error code. As an example, the implementation that supports 8 addresses will contain valid address IDs from 0 to 7. The next *N* bytes will contain message information. *N* is a number between 1 and the maximum message size in bytes. For this implementation, the maximum message size was chosen as 16 bytes. The final byte of the message will be the terminating character, represented by the ASCII newline character. It is recommended that data is sent 1 byte at a time.

Received messages will be stored in a message queue on the device and sent to the host one message at a time when interrogated. For interrupt driven networks, a message ready line will transition from low to high whenever the queue contains at least one message. The receive message queue can be interrogated by sending the byte denoted by the ASCII character ‘r’. When the queue is interrogated, the device will return a sequence of up to *N* bytes followed by a terminating new line character. The maximum size of *N* is the maximum message size, which in this implementation is 16 bytes.

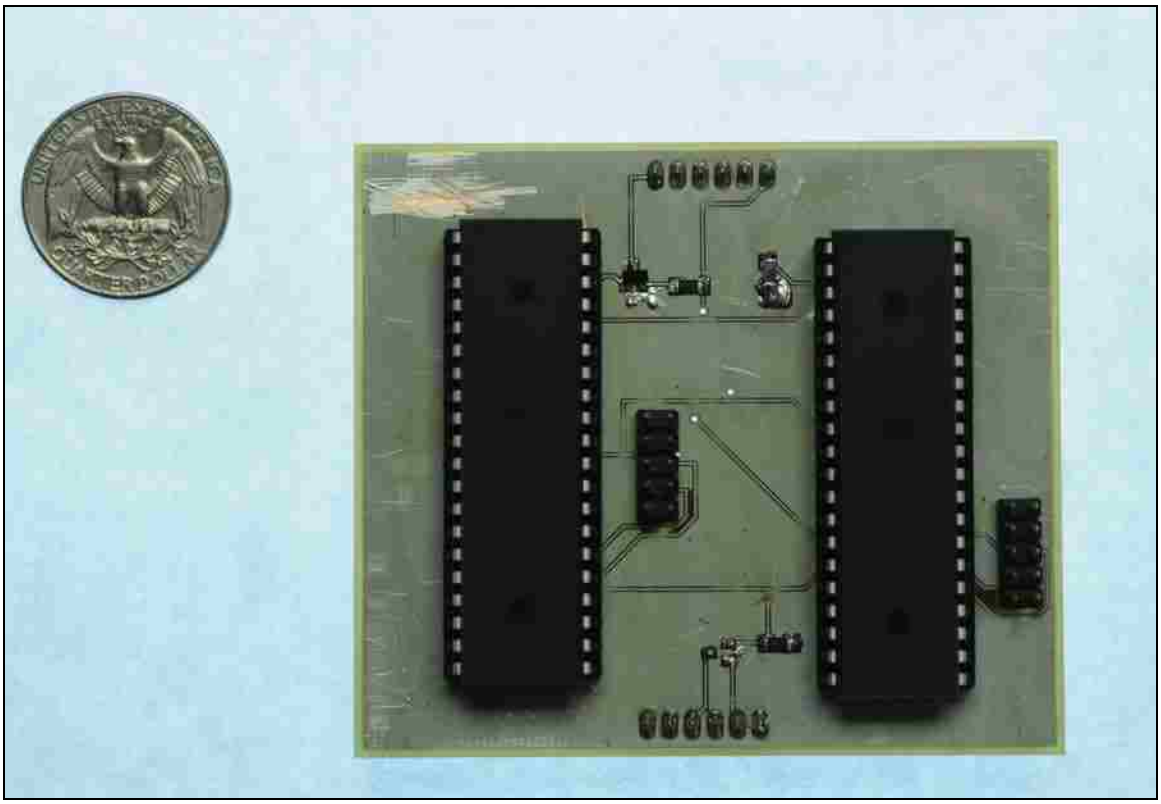

Figure 21. Prototype Hardware Size Reference. The prototype hardware board can be seen in this image. The prototype hardware used 1 or 2 Atmel ATMega164P microcontrollers for signal processing. The prototype also included parasitic power circuits and external wired-OR circuitry to allow for an arbitrary channel voltage (up to 28V).

To set the receiver address, the add address byte, represented by the ASCII character ‘a’, must first be sent. Next, the address index byte must be sent. The address index byte is represented by an integer between 0 and 255. As in the send message case, a number of the possible addresses will be invalid and this will be indicated by the return of an error code. In general, it is recommended that the host itself sets the address in hardware and then sends this information to the device during an initialization routine; however, the address can be changed at any time and it is not necessary to set the address in hardware at the host end. This protocol is documented in Appendix A Host Interface Definition.

Prototype Hardware Design

Hardware

The basic prototype hardware contains two Atmel ATMega164P microcontrollers. The Mega164 was chosen because it is good general purpose MCU, appropriate for a first design iteration. If the 164 was determined to be inadequate, it could be swapped for the ATMega324, 644 or 1284 depending on how much additional functionality was needed. If the Mega164 was determined to be overkill, the software portion could be reused and placed in a device with less functionality.

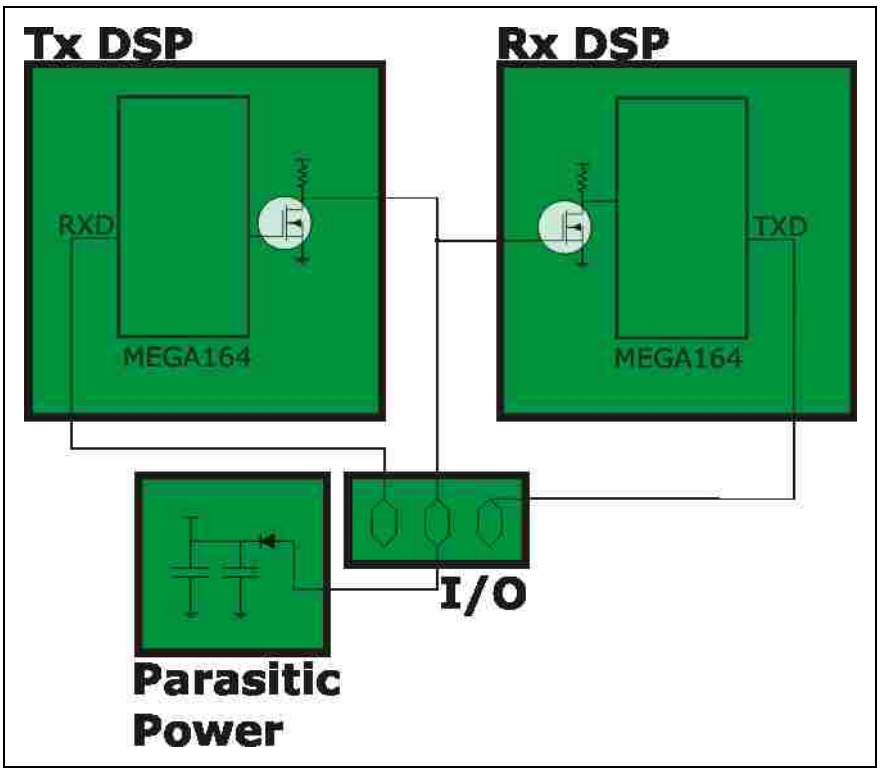

Figure 22. Prototype Hardware Detailed Block Diagram. This diagram shows a high level interconnection between the 4-primary hardware blocks, the I/O, RX and TX path DSP, and an optional parasitic power stage. It also shows some of the implementation details of each block.

As shown in Figure 22, one of these processors is wired to perform receive path Digital Signal Processing (DSP) operations and the other is wired to perform both transmit and receive path DSP operations. To establish the wired-OR channel, each transmitter is connected to the gate of an N-Channel MOSFET, the DMN3150, in an inverter configuration. If any transmitter on the channel outputs logic-1, the channel will be pulled low. As shown in Figure 22, the channel data line is also connected to the gate of another DMN3150 N-Channel MOSFET in an inverter configuration. The drain of this FET is connected to the receive path DSPs. Thus, when the channel goes low, the receiver reads a logic-1 value.

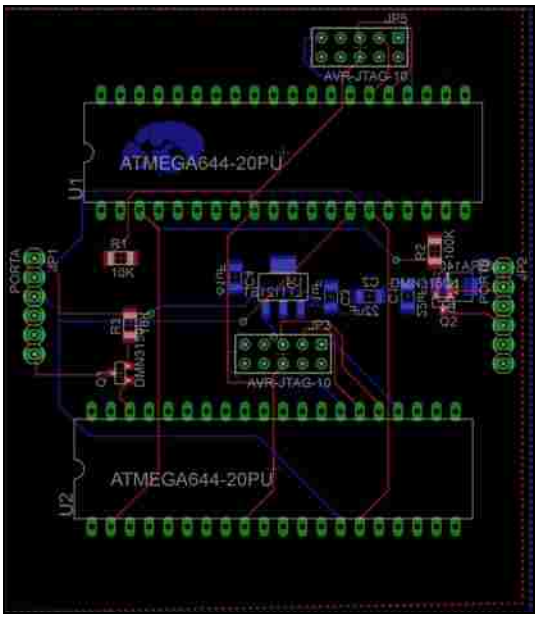

Figure 23. Prototype Hardware Printed Circuit Board Drawing. The PCB was created to allow for JTAG debugging, arbitrary channel voltage (up to 28V), and the option to use either a combined TXRX DSP or separate TX and RX DSPs.

Size

For the prototype, through-hole microcontrollers were used for ease of adding white-wiring/cut-and-jumps on the board where necessary. These 40-pin Dual-Inline Package parts could be replaced with small surface mount parts in a future revision in order to significantly reduce board size. Further, the TX/RX DSPs were separated in hardware in the initial revision to allow for determination of max throughput with dedicated hardware. However, the TX DSP was wired so that it could perform RX operations if software was modified. This potential for hardware reduction would allow for even smaller board size. For the prototype unit, the board was 3.15 inches by 2.75 inches.

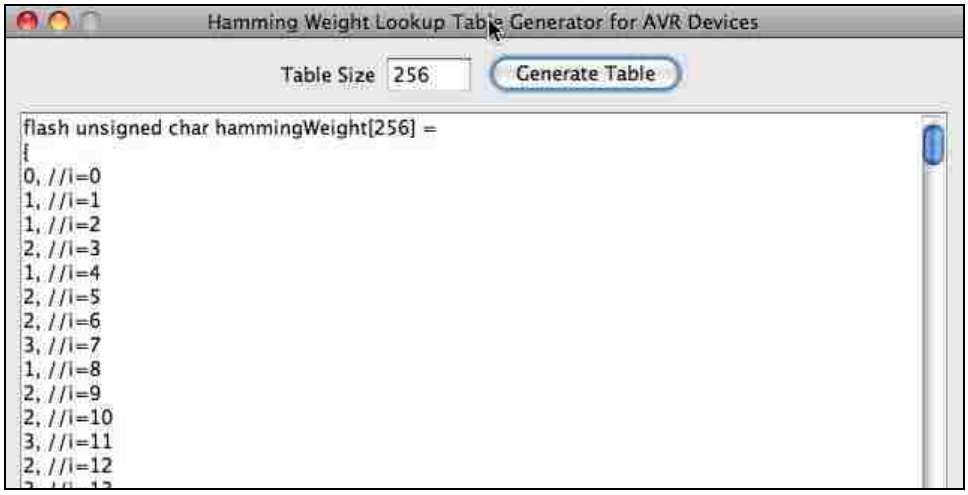

Figure 24. Hamming Weight LUT Generator Screenshot. Calculation of the Hamming weight of a vector is one of the more computationally intensive tasks that receiver must perform. Because this operation must be performed every time a new chip is shifted in, reducing the run-time of this operation can significantly increase system bit-rate. Several different algorithms were implemented and tested to compare performance. In the end, sufficient memory was available to include an 8-bit weight lookup table. Each codeword was then broken down into four 8-bit chunks to calculate the total weight. This screenshot shows the lookup table generator that was created to allow experimentation with lookup table size.

User Support

The prototype hardware iteration could have supported the maximum number of theoretical users with 32-bit codes because addresses were stored in software at compile time. In order to ease memory requirements, 8 users were supported initially.

Parasitic Power

The prototype hardware supports parasitic power with an acceptable channel voltage of 5-12V. The hardware components are regulated at the board level using an efficient LDO Linear 3.3V regulator. Parasitic power is supported using a low V(on) Schottky Diode and two 47 uF capacitors.

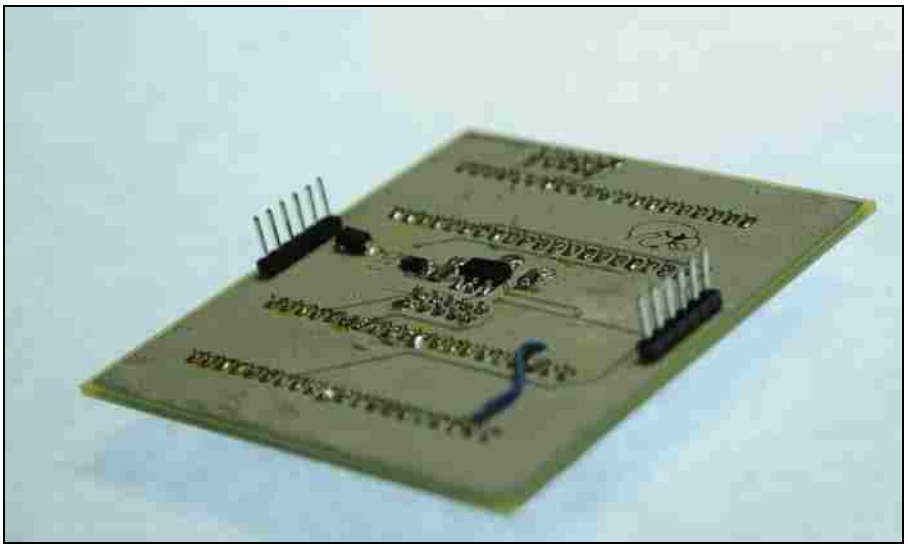

Figure 25. Prototype Cleanup-Wires. Design errors were fixed by cleanup-wires for testing purposes. Shown here is a fix for the connection between the receiver and the channel I/O pin, which was dropped in the schematic design.

Prototype Errors

Several errors were made in the design and layout of the prototype design. For testing purposes, these errors were fixed by cleanup-wires as shown in Figure 25. The primary issues were in the routing of the channel, in particular the connection between the receiver and the channel I/O pin was dropped in the schematic design. These errors were repaired by cleanup-wiring for test purposes, some of the repairs are shown in Figure 25.

Testing and Results

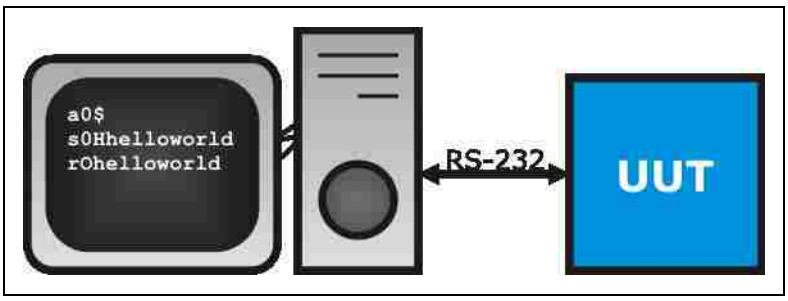

Figure 26. Test Diagram for Loopback Test. A PC is used to control the Unit Under Test (UUT) over RS-232. The RS-232 connection is level shifted to TTL levels before connecting to the device.

Codes Tested

During the time when this version of the prototype hardware was being tested, the code library from a previous chapter had not been developed. At the time of this hardware's testing, equally weighted randomly generated codes were used. The next chapter discusses results for hardware that was used to test the codes developed for the low-weight code library developed in a previous chapter.

Test Results

Two testing scenarios have been conducted with the prototype hardware. First, a loopback test was conducted. In this scenario, a transceiver sends messages to itself. A PC is used to control the node over RS-232. The PC can then command the transmitter to send certain messages and view the messages as the receiver decodes them. This allows for verification of the test setup, prototype hardware build-quality, software correctness and that the communication system will work in a noise free environment. After minor debugging, the system operated with a 0% Bit Error Rate at a data rate of 62,500 chips per second or 1.95 kbps. This matches our expectations from the simulation environment.

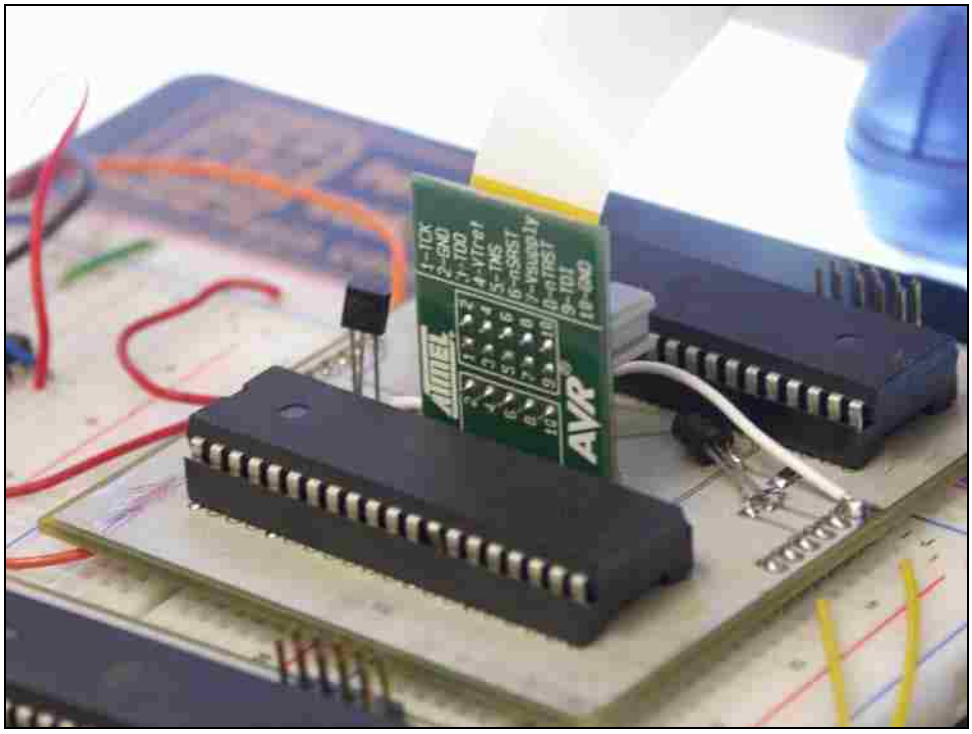

Figure 27. Prototype Hardware in Test Environment. This image shows one of the prototype boards connected to the Atmel JTAG debugger hardware. This image also shows the 2N7000 transistors which were used as a replacement for the desired transistors, whose footprint was designed incorrectly.

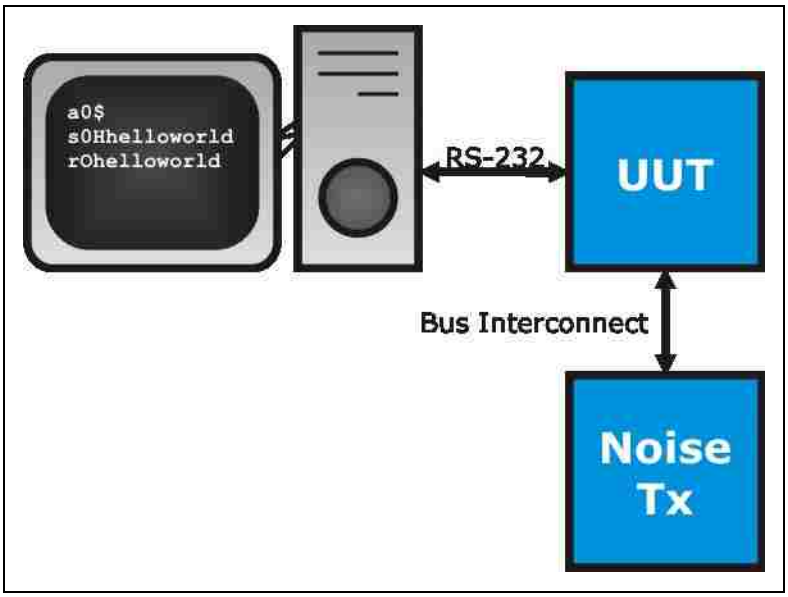

Figure 28. Test Diagram for 2-User Test. A PC is used to control the Unit Under Test (UUT) over RS-232. The second prototype is programmed to send random messages onto the bus; this test provides a test of noise immunity for the other transceiver.

The second hardware test places two transceivers on the same bus. One of these transceivers is operated identically to the first test that was described. The second transmitter is programmed with a special software load as a "co-channel interference generator". This transmitter continuously takes the XOR of its codeword with a random message bit and sends this information. Preliminary tests of the system with two users transmitting simultaneously have shown the receiver connected to the test PC is able to decode messages with approximately a 10% Bit Error Rate. This empirical result shows a similar outcome to the results of the simulator, where the simulator shows slightly better results than the real hardware. This can be explained by slight differences in the simulator and the real hardware, which can be rectified with additional development of the decoding algorithm.

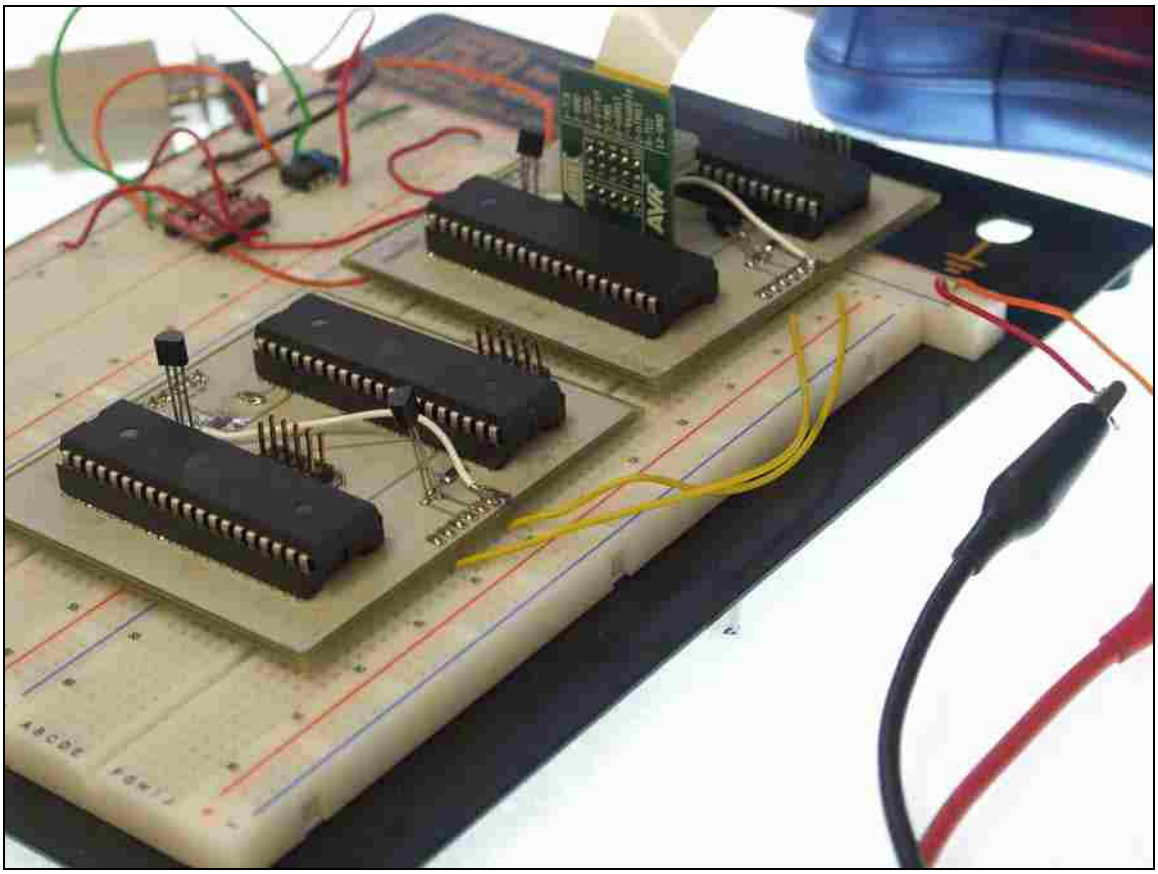

Figure 29. Testing Two Prototype Devices on Bus. This image shows two prototype boards communicating on the same bus. One of the prototypes is connected to the PC for debugging purposes.

Future Work

Board level hardware provides the greatest opportunity for future work. An investigation into the benefits of utilizing a DSP chip, rather than an FPGA or MCU, could be performed. A processor with a combination of MCU and DSP capabilities, for instance the TI OMAP processor, could also be performed. Finally, the development of an ASIC device to replace the board or to replace the FPGA of the Engineering Design Revision would improve speed and, with sufficient volume, would not be likely to increase unit cost significantly.

Conclusion

The development of a versatile, board level transceiver with a 2-pin channel interface was undertaken. Prototype hardware was initially developed using the ATMega164 family of microcontrollers as the centerpiece. The prototype hardware was used to investigate possible features for so-called production hardware. The prototype hardware performed well and at a reasonable bit-rate for low cost. Next, an investigation into the potential use of Field Programmable Gate Arrays (FPGA) was launched. A pure Physical Layer implementation of the 1-Wire CDMA protocol was written in HDL and subsequently synthesized for several FPGA hardware targets. The results of this investigation showed the potential for over 100 times speed improvements in analogous systems and 1:1 speed devices that would support more users by extending codeword length. The board level hardware investigation has yielded a selection of hardware that will cost around 10 dollars per device in low volume. In the next chapter, a method for cost reduction will be investigated by collapsing all functionality into a single low cost processing device that could be embedded in another design.

CHAPTER VII

LOW COST SYSTEM ON A CHIP (SOC) IMPLEMENTATION

In order to provide the lowest cost and most compact possible system hardware, a System on a Chip (SoC) implementation was explored. The system target is the ATTiny family of microcontrollers from Atmel; in particular the 14-pin ATTiny84 Family and the 8-pin ATTiny85 Family are considered. The 6-pin microcontrollers, such as the ATTiny10, were also considered, but were ruled out for various reasons that will be explained.

Goals

The SoC hardware design was intended for immediate adoption. First, the hardware could be used to verify the practicality of the system architecture. Although the system was successfully simulated, it was necessary to prove that adequate speeds could be achieved with low cost hardware. Second, the hardware could serve as a future platform for actual sensor deployments.

In order to satisfy the first goal, it is important to allow for a reasonable number of users to utilize the system at an adequate data rate. Thus, the system should support 8 users at a rate of a few bits per second. In order to satisfy the second goal, it is important to provide a reasonable user interface, small size, and maintain a low cost.

Design

Hardware Design

Although better performance could be gained by use of other processing devices such as Field Programmable Gate Areas (FPGA), fixed point Digital Signal Processors (DSP), or high end General Purpose Processors (GPP), a single microcontroller (MCU) will provide adequate performance at an absolute minimum cost in terms of both development time and material cost.

An MCU provides a particular advantage in its embedded nature. MCUs usually contain non-volatile memory in the form of one to a few hundred kilobytes of Flash and Electrically Erasable Programmable Read-Only Memory (EEPROM). This memory is used to store program memory and constants values, for instance a code lookup table could be contained in EEPROM. MCUs also usually contain anywhere from 32 to a few kilobytes of volatile memory in the form of Static Random Access Memory (SRAM). The MCU also contains an internal clock that generally runs at a few MHz and is calibrated to within 10%. Generally, the only external component needed for an MCU is a decoupling capacitor from VCC to GND. The MCU also generally costs almost nothing in bulk quantity, one would expect to pay anywhere from .5 to 20 USD for a microcontroller in single quantities. The ATTiny85 can be purchased for as little as 1.25 USD per unit in bulk quantity from Digikey.

The advantages described of microcontrollers are unique when compared to the characteristics of FPGA, DSP and GPP implementations in embedded system designs. Generally an FPGA will require at least one EEPROM per FPGA to load program code when power is applied. This is because most FPGAs do not provide non-volatile memory. FPGAs also generally require an external clock module. GPP and DSP implementations often require external RAM modules and ROMs as well. The price floor for these processing units is also higher than that of an MCU, somewhere in the 5 to 10 USD range. This may appear to be a small amount at first, but with the addition of the cost of the external components, and when the fact that this is a transceiver hardware unit that will be reused many times in one system, the price difference can become quite large.

For the reasons mentioned in the above paragraphs, microcontrollers are considered to be the best choice of processing unit. The next important thing to consider is the amount of volatile and non-volatile memory required for the transceiver. No more than 1K or 2K of flash should be required to store the program, as the algorithm is quite

short, and no large libraries will need to be included in the program. The volatile memory requirements are more likely to be the driver. For the 32-bit system that was used in the board-level hardware design, 4-bytes of SRAM are needed for a Receive Shift Register, 1-byte is needed to store the device address, and at least 16-bytes are needed for the received message queue. The device will also need some volatile memory to provide a buffer for the host serial interface (which could be I^2C, UART, SPI, etc). If a modest 16-byte transmit and receive buffer is included, the system will require about 50-bytes of SRAM. By doubling or tripling that amount, enough margin is included to ensure room for future growth, but not so much extra that the chip is overpowered for the application. This SRAM requirement is what allows consideration of the ATTiny family of MCUs. The more powerful ATMega family includes a minimum of 512 bytes of SRAM and 4 kilobytes of Flash. The ATTiny25/45/85 and ATTiny24/44/84 families provide a small hardware profile with 8 and 14-pins respectively and 128/256/512 bytes of SRAM and 2/4/8 kilobytes of Flash. These devices also provide the embedded Universal Serial Interface (USI) device, which provides a convenient mechanism for partial hardware implementation of the previously mentioned host-side serial interfaces, i.e. I^2C, UART, and SPI. As was mentioned in the introduction, 6-pin microcontrollers such as the ATTiny10 were also considered and coveted because of their extremely small form factor. However, these devices provide a maximum of 1 kilobyte of flash and 32 bytes of SRAM. There is also no built-in Universal Serial Interface in these devices, so the implementation of a UART, for instance, would require more SRAM than the partial hardware version with a USI. For these reasons, it is clear that the 6-pin microcontrollers, however appealing, are currently not usable for this application. If a 6-pin MCU with more SRAM were to become available in the near future, one could simply port the code found in this thesis to run on that device.

Hardware Interface

Other portions of the design follow closely with what is described in the board-level hardware implementation. One key difference is the device hardware interface. The device hardware interface for the ATTiny45 configuration is shown in Figure 30. The pins are described in detail in Appendix A Host Interface Definition, but they will also briefly be described here. VCC and GND are required to provide power to the device. VCC must be the same on every device on the bus in order to prevent over-voltage on the channel input pins and to closely match clock speeds across devices on the bus. The host interface provides 2-pins whose functions vary depending on the chosen host interface. In the case of a UART implementation, the pins are Data In (DI), or TXD, and Data Out (DO), or RXD. The default baud rate for UART communications is 9600 baud. The channel interface is 2-pins that should be tied together. These pins function as transmit and receive lines of the node and perform our CDMA communication scheme at approximately 15 bps.

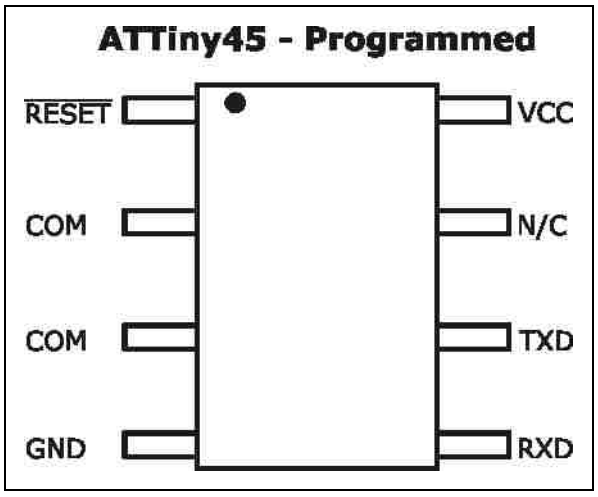

Figure 30. CDMA SOC Pinout on Programmed ATTiny45. This schematic shows the small footprint and nice interface that is possible with a device like the ATTiny45.

Data Link Layer Implementation Details

For this hardware, the same Data Link Layer as was used for the board level hardware was used, along with a few optimizations to significantly decrease the odds of receiving a byte in error.

Three optimizations were made and are explained here. First, if the receiver decodes a bit from a series of N chips, the receive shift register is cleared to all logic-0s.

Second, if the receiver decodes a bit from a series of N chips, the receiver will reject the next bit if it is decoded less than N chips later. This provides protection against incorrectly decoding when the channel is saturated. The implemented data link layer also resets the receive shift register to all logic-0s in this case, this is a provision in the event that the first bit and not the second bit was incorrectly decoded.

Finally, if the receiver decodes a bit from a series of N chips, the receiver will discard the first bit and accept a new bit if it is decoded more than N chips later. This takes advantage of the fact that if bits are decoded incorrectly it is more likely to happen sparsely than densely. This sparseness allows the data link layer to identify past bits that do not belong with current bits.

With these details added to the data link layer, it is highly unlikely that errors will occur. The data link layer has been designed with the mindset that retransmissions are preferred to errors.

Host Interface

For this hardware, the same host interface that was used for the board level hardware was used.

Code Library

For this hardware version, the low weight code library that was developed in a previous chapter was used.

Test Cases

Loopback Test

The first set of tests that will be performed is similar to the first test in the board level hardware implementation. A single transceiver is installed in the system. Two tests are performed with this hardware configuration. First, transmit data is intentionally routed back to the receiver; this step will verify that the node can correctly receive data intended for it independent of co-channel interference. Second, transmit data is routed to each other address in the system; this step will verify that the node can correctly reject data intended for each of the other system users independent of other co-channel interference.

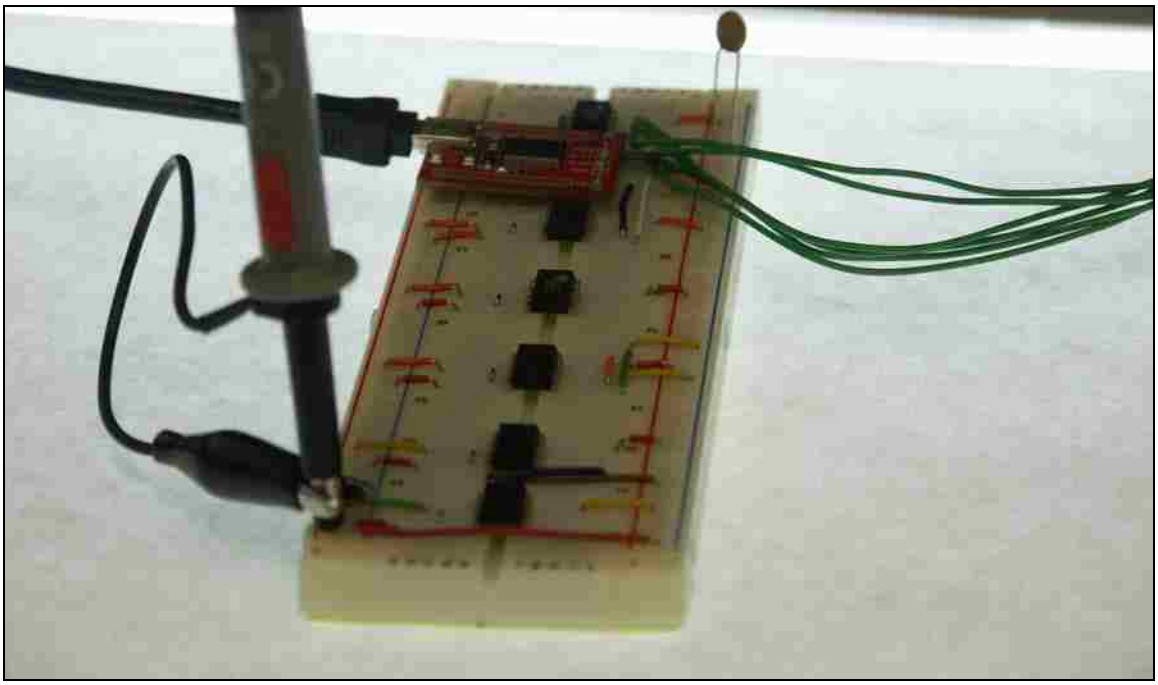

Figure 31. SOC Test Setup. The SOC design was tested with up to 8 users. A test is shown here with 5 users. The sixth MCU (at top) is shown in the programming circuit. A separate programming circuit was necessary since the Tiny45 could not be programmed in-circuit with the desired I/O configuration.

Loopback Test with Co-Channel Interference Transmitter

The second set of tests that will be performed is similar to the second test in the board level hardware implementation. Two transceivers are installed in the system. The first transceiver has a normal software load on it. The second transceiver is referred to as a co-channel interference transmitter; this transceiver continuously injects messages into the system in the form of its own codes. Two tests are performed with this hardware configuration. First, the primary transceiver sends data to itself; this step will verify that the node can correctly receive data intended for it in the presence of some co-channel interference. Second, the primary transceiver will transmit data intended for each other address in the system; this step will verify that the node can correctly reject data intended for each of the other system users in the presence of some co-channel interference.

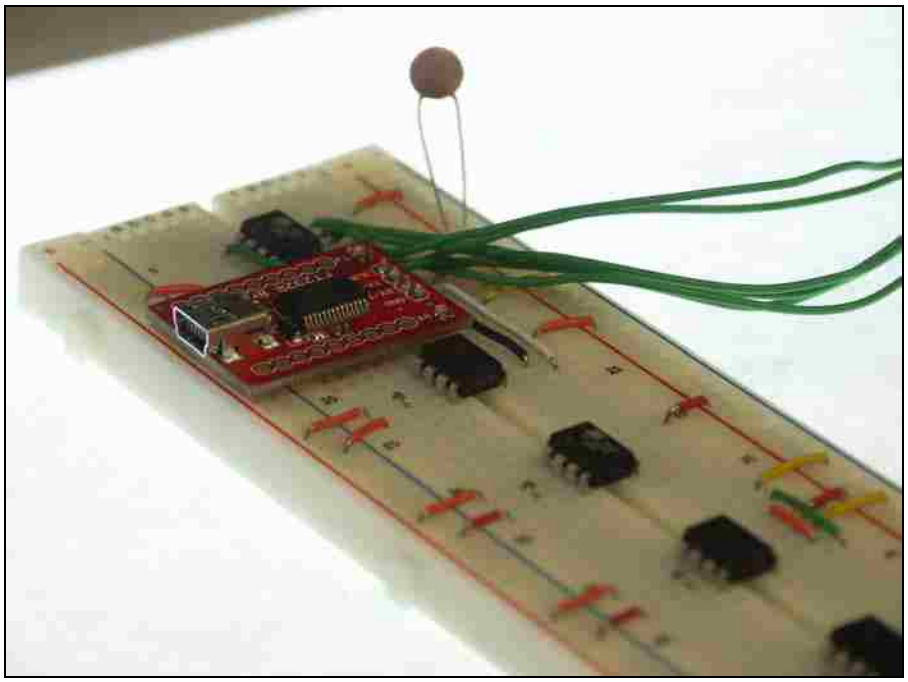

Figure 32. ATTiny45 Chips in Circuit. The MCU at top is shown in the programming circuit. A separate programming circuit was necessary since the Tiny45 could not be programmed in-circuit with the desired I/O configuration. The second MCU is connected to a USB to RS232 converter. This USB circuit is also used to power the test-bed at 3.3VDC.

Scanning Receiver Test with *M* Continuous Message Transmitters

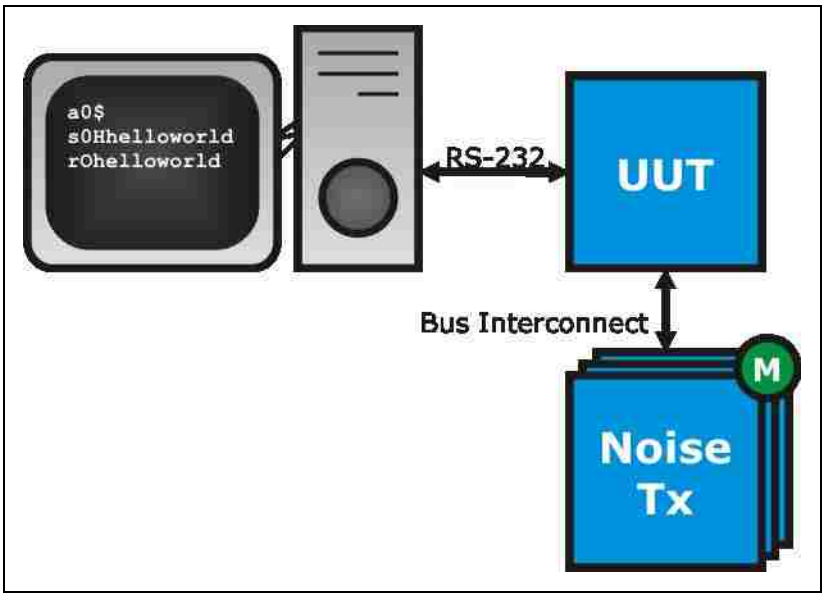

Figure 33. Test Diagram for Test Multiple Message Transmitters. A PC is used to control the Unit Under Test (UUT) over RS-232. The other chips are programmed to send random messages onto the bus; this test provides a test of noise immunity for the other transceiver. *M* can range from 1 to 7, since the code library has 8 sets of codes.

The third test that will be performed tests the performance of each codeword in the presence of maximum co-channel interference. *M* continuous message transmitters are placed in a configuration similar to the co-channel interference transmitter from the second set of tests, where *M* is the number of codes in the system. An additional transceiver is placed in the system and connected to the test PC. This transceiver is considered a dedicated receiver. This dedicated receiver is then used to iterate through each receiver address and receive the data that is being sent. This test verifies that each code can be used to receive data even when every other user in the system is transmitting, which is an unlikely event.

Results

Loopback Test

The loopback test was performed successfully on the system with little difficulty. The system encountered some errors, which were used to fix bugs in the decoding algorithm software. The impact of the software UART was noticeable and caused errors when the CDMA transceiver was run at a high rate of speed. After the decoding algorithm software was fixed, the system received data with 0% bit error rate. Data was transmitted at a rate of 488 chips per second or 15.3 bits per second. The system successfully received the intended data and also rejected data from all unintended users.

Loopback Test with Co-Channel Interference Transmitter

The loopback test with a single co-channel interference transmitter was performed successfully. Data was transmitted with a 0% bit error rate. The system successfully received the intended data and also rejected data from the co-channel interference transmitter.

Scanning Receiver Test with *N* Continuous Message Transmitters

The *N* continuous message transmitters test was somewhat successful. The network showed the expected result: as the number of transmitters on the bus increased, the number of dropped bytes increased. The system experienced 0% bit error rate for single byte transmissions with up to six devices transmitting on the bus. As the number of devices transmitting increased, the number of drops also increased, so more retransmissions would be required. With all eight devices transmitting simultaneously, all bytes were dropped.

Future Work

At the cost of a few extra pins and a minor increase in unit cost, a significant increase in speed could probably be gained by utilizing the ATTiny2313 platform, which includes a hardware UART. A full hardware UART would free significant processor resources, allowing an increase in the frequency of interrupts for the CDMA transceiver portion of the design.

The platform could gain benefits from extension to longer codes. This would reduce the speed of the platform, but would significantly reduce bit-error rate and the need for complicated error-detection and correction by the user. The development of a 64-bit math library could allow 64 bit codes to be investigated.

This device will be integrated into a hardware revision of the Soil Moisture Network that was discussed earlier. The device will also be investigated as a tool in wireless sensor networks that utilize low cost radios with On-Off Keying (OOK) modulation.

Conclusion

The development of a single chip transceiver with a 2-pin channel interface was undertaken. Prototype hardware was developed using the ATTiny25 family of microcontrollers as the centerpiece. The use of this hardware allows for an absolute minimum cost per unit, costing around $1 even in small quantities, but performs at a lower speed than the prototype hardware utilizing the ATMega164. Although the hardware takes a performance hit, the speed will be adequate for the low data rate transfers required in the typical environmental sensing application. If faster speeds are required, a chip such as the ATTiny2313 could be examined as a slightly more expensive, but faster alternative.

CHAPTER VIII

CONCLUSION

A new method for multiplexing data over a wired communication channel has been explored. Although the limits of the Binary Channel CDMA technique are apparent, this method can be exploited in certain applications, particularly when taking lessons from attempted usage of CDMA in an Optical channel. The example of a distributed sensor network was given, and the implementation of a system to meet the needs of such a network was investigated using a customized Java Simulation Engine and an embedded hardware design. Two sets of embedded hardware were investigated. The simulator and hardware investigations supported the validity of the approach for low data rate networks with a small number of users.

Investigation of FPGA Hardware for Improved Reliability

Hardware

An investigation into FPGA Hardware was undertaken, the goal was to maximize the potential number of users and the speed of the system. If this system is adequately simulated and synthesized, future work could create a board to serve as a replacement for the previously described hardware when better performance is necessary. The system could be updated while still attempting to maintain reasonable cost. To this end, the best development platform was determined to be a Field Programmable Gate Array (FPGA). The hardware development included the simulation of the device firmware, the synthesis of the device. For this design flow, the real FPGA device would be chosen at the synthesis phase based on need. The FPGA design solely implements the physical layer. Higher networking layers that were added in software to the other designs will be considered a separate topic and could either be implemented on the same device, somewhere else on the board, or in between the board and the user.

Firmware Design

The firmware was designed with three distinct hardware pieces. The design top is the CDMA_PHY_FPGA module. This module provides the input and output to the FPGA hardware that will be chosen and performs routing between other hardware modules within the device. The second module is the CDMA_ENCODER. This module performs the transmitter function by taking the message bit and determining which chips should be sent at which time. The third module is the CDMA_DECODER. This module performs the receiver function by filling a receive shift register with the incoming message chips and continuously determining whether the chips are intended for the receiver based on the RX Code ID.

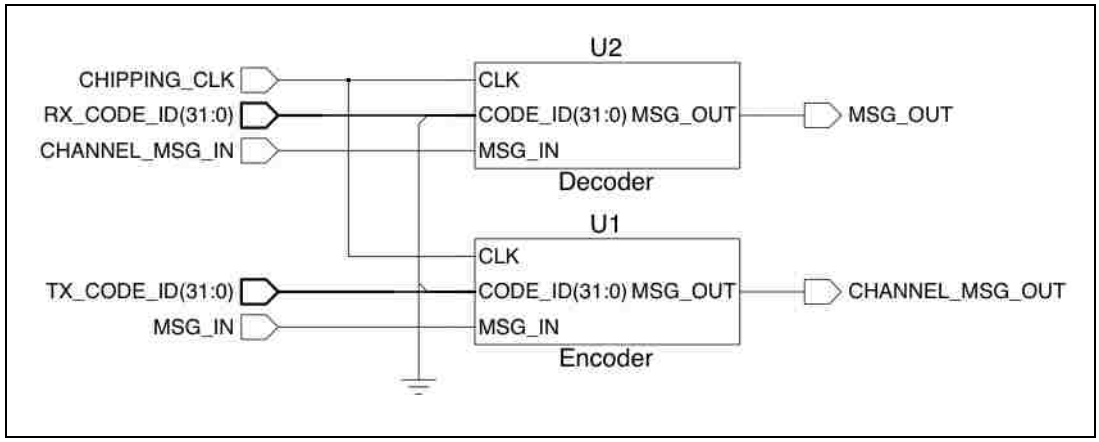

Figure 34. CDMA FPGA RTL Schematic. This schematic clearly shows the necessary pin-out for a device that would be built from this firmware.

HDL Simulation

Hardware Description Language (HDL) simulation was performed by designing several test bench units and test circuits. The test benches mimic the real hardware test benches that were used for checking out the prototype board hardware. The firmware

was verified to perform bit-level transfers in a loopback test and in the presence of co-channel interference.

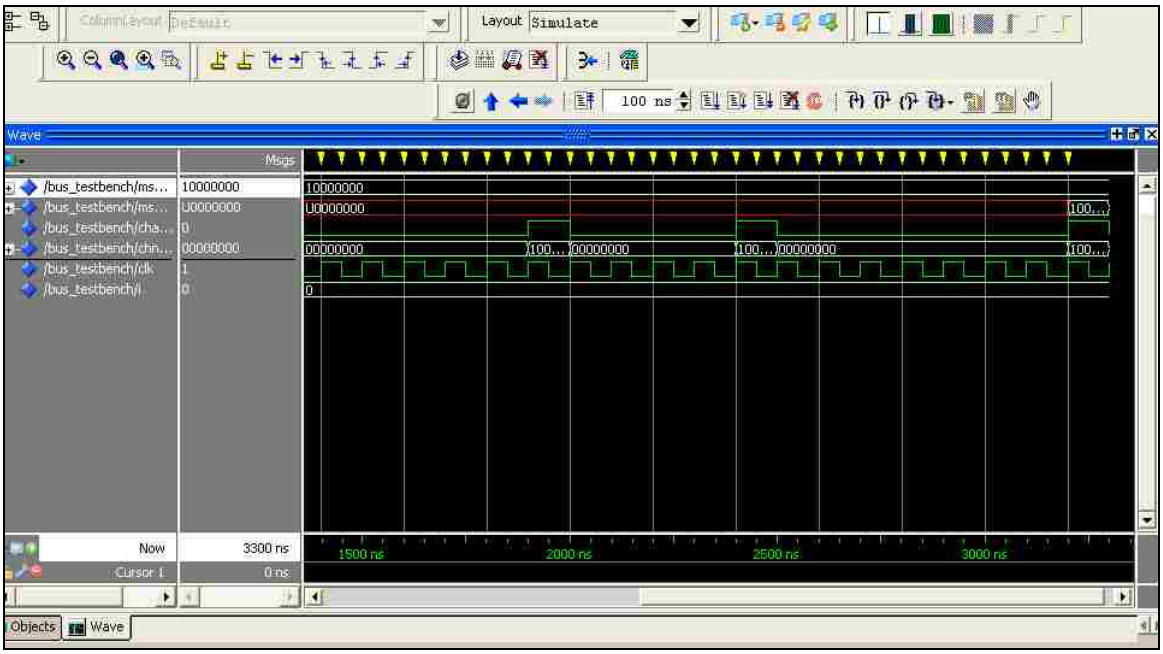

Figure 35. HDL Simulation Wave Outputs

Synthesis

Synthesis was performed to verify that the system could be practically fielded on a variety of devices and to investigate the speed of the device when placed on low cost to high-end FPGA devices. Four devices were synthesized as a part of this investigation.

Low Cost, 32-Bit Device

First, a low cost device with 32-bit codes was investigated. For this device, a search of vendors revealed that the Spartan3A series of FPGAs would be best suited. At the time of writing, the XC3S50A costs around 5 dollars in single quantities. Synthesis was performed on this hardware with a set of 32-bit codes. Timing reports revealed a

maximum chipping clock rate of approximately 148 MHz. With this chip rate, the system would have a physical layer bit rate of 4.6 Mbps.

Low Cost, 1024-Bit Device

Next, a low cost device with 1024-bit codes was investigated. For this device, the XC3S50A was also used. Synthesis was performed on this hardware with a set of 1024-bit codes. Timing reports revealed a maximum chipping clock rate of approximately 116 MHz. With this chip rate, the system would have a physical layer bit rate of 113.6 kbps. This synthesis shows that a sufficiently fast device with very low error rate for a small number of users could be generated with minimal cost.

Fastest Speed, 32-Bit Device

A highest performance device with 32-bit codes was investigated. For this device, the highest performance FPGA from Xilinx were considered. Xilinx was chosen to maintain consistency with the low cost platform that was chosen. The synthesis tool that was used did not provide support for the latest Xilinx devices, the Virtex 7 Series, so instead the Virtex 6 Series was used. The CX240T family of parts was chosen for their best in class performance. At the time of writing, the CX240T costs over 1000 dollars in single quantities. Clearly, this device exceeds the cost threshold for the system that is being built, but it is useful as a comparison for what would be physically possible without going to full-on ASIC fabrication at this point in time. Synthesis was performed on this hardware with a set of 32-bit codes. Timing reports revealed a maximum chipping clock rate of approximately 340 MHz. With this chip rate, the system would have a physical layer bit rate of 10.7 Mbps.

Fastest Speed, 1024-Bit Device

A highest performance device with 1024-bit codes was investigated. For this device, the CX240T family of parts was also used. Synthesis was performed on this

hardware with a set of 1024-bit codes. Timing reports revealed a maximum chipping clock rate of approximately 284 MHz. With this chip rate, the system would have a physical layer bit rate of 277 kbps.

Testing and Results

Two testing scenarios have been simulated with the Engineering Design Revision HDL Firmware. First, a loopback test was conducted. In this scenario, a transceiver sends messages to itself. An HDL test bench is used to control the node. The test bench commands the transmitter to send certain messages and then verifies that those messages are also received after an expected transmission delay. This allows for verification of the test setup, firmware correctness and that the communication system will work in a noise free environment. After minor debugging, the 1024-bit system operated with a 0% Bit Error Rate. This matches our expectations from the simulation environment.

The second hardware simulation places two transceivers on the same bus. One of these transceivers is operated identically to the first test that was described. The test bench causes the second transceiver to send garbage messages onto the bus. Preliminary tests of the system with two users transmitting simultaneously have shown the primary receiver is able to decode messages with approximately a 0% Bit Error Rate using 1024-bit codes. This empirical result shows a similar outcome to the results of the software simulator.

Future Work

Although the platform development was a success, there are still upgrades that could be made. The synthesized FPGA firmware could be loaded onto a device and a board could be designed to allow use of that platform in more advanced systems. This would allow for improved reliability or speed, depending on the user preference.

The primary basis for the research was the need to find a 1-wire, all digital scheme that would allow users to communicate simultaneously without significant

management from higher networking layers. A CDMA-like system was chosen as an obvious approach to this problem. With the choice of the wired-OR channel, one can ask the question: is there another technique that is more efficient and will further minimize bit-error rate and the need for retransmission? Immediately, the consideration of the other wireless digital modulation schemes comes to mind. In particular, researchers at IIHR will investigate a Frequency Modulation type system, a Phase Modulation type system, and an Orthogonal Frequency Division Multiplexing type system.

Results

The results of this research are significant. Simulation based testing shows that on the order of 50 users, using 1024-bit codes, can communicate simultaneously using a single wire without the aide of higher networking layers. Synthesized VHDL hardware shows that such a 1024-bit system could be realized on extremely low-cost hardware and yield speeds of a few 100 kilobits per second. Meanwhile, simulation results for a 32-bit system were verified in two hardware scenarios. The 32-bit system showed adequate performance for a small-scale, low-speed sensor deployment. Along with the development of higher networking layers, this system could allow huge numbers of sensors to be chained together at a single wireless node.

Conclusion

The problem of inter-device communication has been discussed. The Wired-OR CDMA technique could be adopted in embedded systems designs, board-to-board communication, and systems on a chip. The primary purpose of this thesis was to prove the validity of this approach with respect to environmental sensor networks. Physical devices have been designed and tested, and are ready to be applied to a new sensor network deployment.

APPENDIX A

HOST INTERFACE DEFINITION

Hardware Interface

Pin Description

Pin Name	**Pin Direction**	**ATTiny25/45/85**
VCC	In	Vcc
CHANNEL_TX	Out	PB4
CHANNEL_RX	In	PB3
UART_TX	Out	PB1
UART_RX	In	PB0
RSVD (MSG_RDY)	N/C	PB2
N_RESET	In (Active Low)	PB5
GND	In	GND

Table A1. Host Interface Pin List

VCC: 2.7-5V must be provided on this pin.

UART_TX: This is the TX pin of a 3-pin 9600 Baud 8N1 UART serial interface. The command set for this interface is documented in section 2 of this document.

UART_RX: This is the TX pin of a 3-pin 9600 Baud 8N1 UART serial interface. The command set for this interface is documented in section 2 of this document.

RESET: This pin is active low and should be pulled high to VCC during normal operation.

CHANNEL_TX: For reference, this is the channel transmit pin, but it should be tied directly to CHANNEL_RX under a normal configuration in order to form a proper Communication Channel connection. The logic levels for this pin are 0 to VCC with a .5V margin at both levels.

CHANNEL_RX: For reference, this is the channel receive pin, but it should be tied directly to CHANNEL_TX under a normal configuration in order to form a proper Communication Channel connection. The logic levels for this pin are 0 to VCC with a .5V margin at both levels.

GND: This is the system common ground.

User Interface

UART Functions

Send Message

The following sequence of characters should be sent:

s[id][msg text]\n

id: The code id with which to transmit (1 byte).

msg text: The information to encode and transmit.

Response:

H

Receive Message

The following sequence of characters should be sent to receive the first message out of the received queue (messages will be served on a first in first out basis (FIFO)):

r

Response:

If there are characters to be sent: *O[msg byte]*

Else: *E*

Set Receive Code ID

The following sequence of characters should be sent:

a[id]

id: The code id to set (1 byte)

Response:

$

APPENDIX B

SOFTWARE DESCRIPTION FOR ATMEGA164 BOARD LEVEL HARDWARE IMPLEMENTATION

C Source File: main.c

This file contains the main application and helper functions. It also contains the decoder and encoder algorithms and interrupt service routines for sending and receiving data.

C Header File: config.h

This file contains constants that abstract details of the hardware implementation of the device. This allows for redesign of the board without modification of the source files.

C Header File: constants.h

This file contains constants that abstract details of the transceiver operation, this includes threshold values, duty cycles, clock speeds, code lengths, etc.

C Header File: xcvr_common.h

This file contains the code library, constants for calculating Hamming weight, and miscellaneous global variables that are used by the transceiver.

C Source File: init.c

This file is the microcontroller dependent initialization file. It properly initializes all of the necessary data registers, timers, interrupt sources, etc.

APPENDIX C

SOFTWARE DESCRIPTION FOR ATTINY45 SYSTEM ON A CHIP IMPLEMENTATION

C Source File: main.c

This file contains the main application and helper functions. It also contains the decoder and encoder algorithms and interrupt service routines for sending and receiving data.

C Header File: xcvr_common.h

This file contains the code library, constants for calculating Hamming weight, and miscellaneous global variables that are used by the transceiver.

C Header File: config.h

This file contains constants that abstract details of the hardware implementation of the device. This allows for redesign of the board without modification of the source files.

C Header File: constants.h

This file contains constants that abstract details of the transceiver operation, this includes threshold values, duty cycles, clock speeds, code lengths, etc.

C Header File: USI_UART_config.h

This file contains constants to configure the SW UART module, including baud rate and message format.

C Source File: init.c

This file is the microcontroller dependent initialization file. It properly initializes all of the necessary data registers, timers, interrupt sources, etc.

C Source File: USI_UART.c

This file contains the UART that was designed using the Universal Serial Interface port on the ATTiny45. This code is based on Atmel application note AVR307.

APPENDIX D

FIRMWARE DESCRIPTION FOR ENGINEERING REVISION HARDWARE

VHDL Source File: CDMA_FPGA.vhd

This file is the top-level implementation of the FGPA, it contains all of the modules for the device and the physical I/O pins.

VHDL Source File: CDMACoder.vhd

This file contains the physical layer transmitter code. When it is in a transmit cycle and at the chip clock rising edge, the transmitter shifts out a chip.

VHDL Source File: CDMADecoder.vhd

This file contains the physical layer receiver code. When it is enabled and at the chip clock rising edge, the receiver shifts in the channel level and attempts to decode a bit.

VHDL Package File: constants.vhd

This file holds all of the constants that configure the device, including code length and the code library.

REFERENCES

[1] K. Alishahi, F. Marvasti, V. Aref and P. Pad, "Bounds on the Sum Capacity of Synchronous Binary CDMA Channels," *IEEE Trans. Inf. Theory,* vol. 55, pp. 3577-3593, 08, 2009.

[2] M. Azizoglu, J. A. Salehi and Y. Li, "Optical CDMA via temporal codes," *Communications, IEEE Transactions on,* vol. 40, pp. 1162-1170, 1992.

[3] R. H. Bell Jr., Chang Yong Kang, L. John and E. E. Swartzlander Jr., "CDMA as a multiprocessor interconnect strategy," in *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on,* 2001, pp. 1246-1250 vol.2.

[4] D. P. Bertsekas and R. G. Gallager, *Data Networks / Dimitri Bertsekas, Robert Gallager.* Englewood Cliffs, N.J. : Prentice Hall, 1992.

[5] Boon-Keat Tan, R. Yoshimura, T. Matsuoka and K. Taniguchi, "An efficient data transmission interface for VLSI systems using code-division multiple access technique," in *Solid-State Circuits Conference, 2001. ESSCIRC 2001. Proceedings of the 27th European,* 2001, pp. 149-152.

[6] F. R. K. Chung, J. A. Salehi and V. K. Wei, "Optical orthogonal codes: design, analysis and applications," *Information Theory, IEEE Transactions on,* vol. 35, pp. 595-604, 1989.

[7] M. Clendenin, "CDMA gets a spin for backplanes," *Electronic Engineering Times (01921541),* pp. 37, 02/17, 2003.

[8] L. Cottatellucci, R. R. Müller and M. Debbah, "Asynchronous CDMA Systems With Random Spreading--Part I: Fundamental Limits," *IEEE Trans. Inf. Theory,* vol. 56, pp. 1477-1497, 04, 2010.

[9] L. Cottatellucci, R. R. Müller and M. Debbah, "Asynchronous CDMA Systems With Random Spreading--Part II: Design Criteria," *IEEE Trans. Inf. Theory,* vol. 56, pp. 1498-1520, 04, 2010.

[10] G. Farhadi and S. H. Jamali, "Performance Analysis of Fiber-Optic BPPM CDMA Systems With Single Parity-Check Product Codes," *IEEE Trans. Commun.,* vol. 54, pp. 1643-1653, 09, 2006.

[11] J. P. F. Glas, "An embedded CDMA-receiver A design example," *Integration, the VLSI Journal,* vol. 23, pp. 95-111, 10, 1997.

[12] E. Haas. Dr. Erik Haas - DLR homepage. 2010(12/27), http://www.kn-s.dlr.de/People/Haas/.

[13] S. 'Han, "Optical CDMA with Optical Orthogonal Code," *Multiuser Wireless Communication (EE318K) Class Project,* 2002.

[14] T. Ida, S. Shimizu, T. Matsuoka and K. Taniguchi, "Wired CDMA Interface with Adaptivity for Interconnect Capacitances," *IEICE Trans.Fundam.Electron.Commun.Comput.Sci.,* vol. E88-A, pp. 2702-2706, October, 2005.

[15] V. P. Ipatov, *Spread Spectrum and CDMA : Principles and Applications / Valery P. Ipatov.* Chichester, West Sussex, England ; Hoboken, NJ, USA : J. Wiley, 2005.

[16] Jongsun Kim, I. Verbauwhede and M. -. F. Chang, "Design of an Interconnect Architecture and Signaling Technology for Parallelism in Communication," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 15, pp. 881-894, 2007.

[17] S. Kasera, N. Narang and S. Narang, *Communication Networks : Principles and Practice / Sumit Kasera, Nishit Narang, Sumita Narang.* New York : McGraw-Hill, 2007.

[18] J. Kim, Bo-Cheng Lai, F. C. Mau-Chung and I. Verbauwhede, "A Cost-Effective Latency-Aware Memory Bus for Symmetric Multiprocessor Systems," *IEEE Trans. Comput.,* vol. 57, pp. 1714-1720, 12, 2008.

[19] W. C. Kwong and Guu-Chang Yang, "Design of multilength optical orthogonal codes for optical CDMA multimedia networks," *Communications, IEEE Transactions on,* vol. 50, pp. 1258-1265, 2002.

[20] U. Madhow, *Fundamentals of Digital Communication.* Cambridge: Cambridge : Cambridge University Press, 2008.

[21] J. J. Niemeier, "Embedded instrumentation for soil temperature and soil moisture monitoring," 2007.

[22] T. Nikolic, G. Djordjevic and M. Stojcev, "Simultaneous data transfers over peripheral bus using CDMA technique," in *Microelectronics, 2008. MIEL 2008. 26th International Conference on,* 2008, pp. 437-440.

[23] T. Nikolic, M. Stojcev and Z. Stamenković, "Wrapper design for a CDMA bus in SOC," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on,* 2010, pp. 243-248.

[24] T. Nikolic, M. Stojcev and G. Djordjevic, "CDMA bus-based on-chip interconnect infrastructure," *Microelectronics Reliability,* vol. 49, pp. 448-459, 4, 2009.

[25] J. G. Proakis and M. Salehi, *Communication Systems Engineering / John G. Proakis, Masoud Salehi.* Upper Saddle River, N.J. : Prentice Hall, 2002.

[26] P. R. Prucnal, Ed., *Optical Code Division Multiple Access: Fundamentals and Applications.* Boca Raton, FL: CRC Press, 2006.

[27] H. F. Rezaei, N. Sitter, and A. Kruger, "Next Generation System for Real-Time Monitoring of Rainfall, Soil Moisture, and Soil Temperature," Sensors Applications Symposium (SAS), 2011 IEEE, 2011, pp. 70-75.

[28] T. J. Richardson, *Modern Coding Theory.* Cambridge ; New York: Cambridge ; New York : Cambridge University Press, 2008.

[29] H. Schulze and C. Lüders, *Theory and Applications of OFDM and CDMA : Wideband Wireless Communications / Henrik Schulze and Christian Lüders.* Chichester, England ; Hoboken, NJ : Wiley, 2005.

[30] A. Tripathi and M. S. Korde. 2010, DSSS based CDMA modem using FPGA & microcontroller. *Proceedings of the 4th International Conference on Communications and Information Technology* pp. 128.

[31] L. Wirbel, "CDMA chip targets local loop," *Electronic Engineering Times (01921541),* pp. 92, 05/23, 1994.

[32] H. Yin, *Optical Code Division Multiple Access Communication Networks Theory and Applications.* Beijing : New York: Beijing : Tsinghua University Press ; New York : Springer, 2008.

[33] Q. Zhao, P. Cosman and L. B. Milstein, "Optimal Allocation of Bandwidth for Source Coding, Channel Coding, and Spreading in CDMA Systems," *IEEE Trans. Commun.,* vol. 52, pp. 1797-1807, 10, 2004.