Spring 2014

Mesh-Bus, a double-layer coded, time-transparent digital distributed single-wire bus

Hamid Fahim Rezaei
University of Iowa

MESH-BUS, A DOUBLE-LAYER CODED TIME-TRANSPARENT DIGITAL DISTRIBUTED SINGLE-WIRE BUS

by

Hamid Fahim Rezaei

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

May 2014

Thesis Supervisor: Professor Anton Kruger

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

PH.D. THESIS

_____

This is to certify that the Ph.D. thesis of

Hamid Fahim Rezaei

has been approved by the Examining Committee
for the thesis requirement for the Doctor of Philosophy
degree in Electrical and Computer Engineering at the May 2014 graduation.

Thesis Committee: _____
Anton Kruger, Thesis Supervisor

_____
David R. Andersen

_____
Weiyu Xu

_____
Mathews Jacob

_____
Craig Just

ACKNOWLEDGMENTS

I would like to thank my committee members for accepting to serve on my committee. I would also like to thank Dr. James Niemeier for his helps and inputs during the whole period of my Ph.D. program. My deepest gratitude is for Professor Anton Kruger, not only for his brilliant ideas and extensive supports during the last for years as my Ph.D. advisor, but also for his kindness, understanding and insights as a life mentor. Finally I would like to thank my wife, Raheleh, and my family back in Iran, whose mental support helped me endure difficult times.

ABSTRACT

Medium access mechanisms are one of the most important aspects of buses, which are shared mediums. Almost all standard buses use Time Division Multiple Access (TDMA) as their medium access scheme. Such buses usually are multi-wire, very sensitive to time synchronization, and often managed by a master node.

In this thesis, we develop new non-TDMA schemes for bus communications which are based on Code Division Multiple Access (CDMA), or Frequency Division Multiple Access (FDMA) that do not have the intrinsic limitations of traditional buses. The proposed schemes are based on a single wire bus setting. Since, in theory, any node can have a dedicated communication link to any other node on the bus, the nodes virtually form a fully-connected mesh, hence the name *Mesh-Bus*.

In such schemes, no master node is required on the bus; therefore, we can have a distributed bus in which all the nodes have the same functionality. Also, no time synchronization is required. Every node, using its unique code/frequency, creates a virtual private link to the other nodes, and using such interference-resistant virtual private links, the nodes communicate data to each other.

This dissertation explores the underlying principles of such non-TDMA schemes and through extensive software simulations investigates various scenarios in for CDMA scheme, and studies the performance for the system. Finally, a hardware implementation of the CDMA scheme is presented, and some experimental results are provided to validate the simulation results.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The purpose of this study is to develop the underlying concepts for a new type of bus [2] for serial data communications. This type of bus is full-duplex, digital, binary, and shares just one wire among all the nodes on the bus for bidirectional data communications. We call this the *Mesh-Bus*, see Figure 1.

The goal of the Mesh-Bus is to provide a simple uncoordinated communications channel [3], similar to the concept of the '*ether*', for low data rate communications to a group of low complexity sensors. In such a bus, new sensors are able to join the bus at any desired time. New nodes can join an existing communication channel, or form their own, independent channel(s) using the existing communications wire. As long as such new nodes follow a set of primitive rules of the host Mesh-Bus, their communications do not interfere with the existing communications, or will have a minimum and tolerable effect on the performance of the bus. Further, no node is responsible for managing the

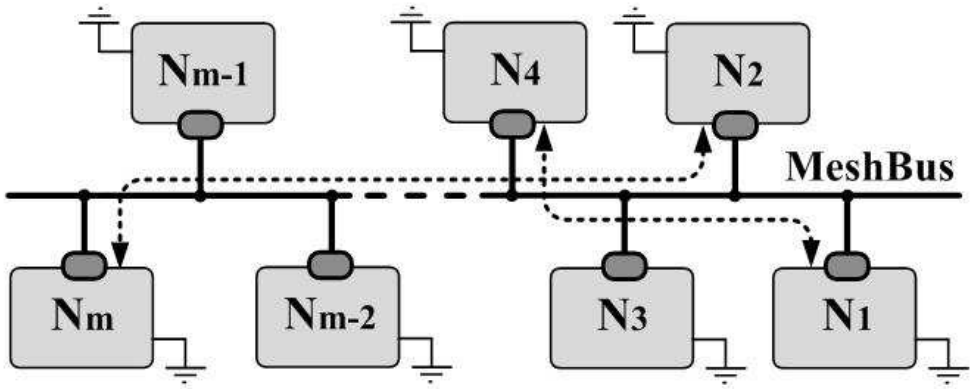


Figure 1 The Mesh-Bus, through which *N* nodes communicate bidirectionally over a single wire. A common ground-connection is implied. Although it is a bus, but the nodes are able to form a virtual mesh configuration, because every node is capable of communicationg with any other node at any desired time.

bus, and in this sense, the Mesh-Bus is a distributed [4] bus, without the requirement of a master-slave configuration [5].

Current bus protocols are typically based on Time Division Multiple Access (TDMA) [6], i.e., the shared access to the bus occurs through time sharing. In such protocols, every node has a non-overlapping *timeslot*, during which that node is the sole owner of the bus. Accurate timing of the timeslots is the only mechanism of interference avoidance among the nodes. Thus, nodes need sophisticated hardware resources (such as good clocks) to facilitate accurate timing.

Additionally, the task of assigning and managing the timeslots is a major issue in TDMA-based buses. In the case of dynamic timeslot assignment, there must be a master node on the bus to do the job, and in the case of pre-determined static timeslot assignment, the bus is not able to serve in an uncoordinated way anymore.

Such limitations eliminate the choice of a TDMA-based medium-access for Mesh-Bus. Therefore, the medium access of the Mesh-Bus should be based on an non-TDMA scheme, e.g., Frequency Division Multiple Access (FDMA) [7] or Code Division Multiple Access (CDMA) [8].

The traditional multiple access schemes such as FDMA and CDMA were originally designed for analog, linear communication channels[1]. The communication channel of the Mesh-Bus is nonlinear and digital, and cannot use such schemes directly. Therefore, the main challenge of this study is to develop an appropriate CDMA-based, or FDMA-based, multiple-access scheme for the Mesh-Bus.

---

[1] The nonlinearity of the wireless channels as the result of nonlinear power amplification, analog-digital conversion, etc. [9],the frequency-selectiveness of the fading channels and so on., are addressed by various techniques such as the channel parameter estimation and so on a part of physical layer; therefore, the multiple access schemes, as of MAC layer protocols, function based on the assumption of linearity of the channels.

Chapter 2, the literature review, covers different topologies [10] of network interconnection, and explores the characteristics of buses, as one of such topologies. Then, different types of buses are explored briefly, and a few well-known serial buses, such as RS-485, RS-232, 1-Wire, I^2C, and SPI are examined. The next section of the literature review explores the concepts of different multiple access schemes such as TDMA, FDMA, and CDMA. In the next section, digital binary channels [11], and their error models are introduced and examined. The chapter concludes with the introduction of Optical Orthogonal Codes [12]. These codes are the unipolar equivalents of the bipolar codes used in traditional CDMA, and were developed for Optical CDMA or OCDMA [13].

Chapter 3 starts with the characteristics and requirements of the Mesh-Bus, and introduces *Bus-Modulators*, which are hardware interfaces between the physical bus and the nodes, and act as bidirectional translators. Next, the concept of *NOR-bus*, as a Wired-OR [14] scheme with the Z-Channel [15] error model, is introduced, and a general detection rule for the NOR-bus is formulized. The concept of *Atoms*, as the specific unary massages with the property of being rather immune to the interference, is introduced and formalized. Atoms provide the required *channels-coding* [16] on a Mesh-Bus. Then, in order to transmit information over the Atom-based channels, an additional *information-coding* scheme is suggested. This information-coding scheme, which relies on the time-distancing of the transmissions of the Atoms, accompanied with the channel-coding scheme, form a double layer coded communication system called *double layered time-distanced unary coding scheme*. This coding scheme, as a general physical layer [17] scheme for the Mesh-Bus, could be employed for both the CDMA-based and the FDMA-based Mesh-Buses of the two following chapters, 4 and 5.

Chapter 4, Wired-CDMA, investigates CDMA-based Atoms, as the channel codes of the Mesh-Bus. It starts with exploring the traditional CDMA scheme and the properties of its codes. This section ends with introducing the near-far problem [18, 19], as the most

problematic issues of the wireless CDMA scheme. Next, OOCs as CDMA-based Atoms are presented in more depth, and their properties are examined. Then, a detection process with a linear computational complexity is suggested for such a scheme. Next, the derivation process of a specific group of OOCs is presented. Some error rate simulations are performed for these OOCs. This section concludes with studying the effects of the dime-distanced information coding scheme on the error rate, and the data rate of the Wired-CDMA.

Chapter 5 explores the hardware implementation of the Bus-Modulators. Such a hardware implementation deals with the electronic circuitry, which is a microcontroller based board, as well as the firmware of the microcontroller which is responsible for the operation of the Bus-Modulator. In this chapter the hardware and the firmware details are presented, and the difficulties and limitations of such an implementation are studied.

Chapter 6 presents the performance analysis for the Wired-CDMA, namely the rate of detection error versus the number of the nodes on the bus. In this chapter, the results of both software simulations and experimental data of the hardware implementation are used for these analyses.

Chapter 7 introduces Wired-FDMA as the FDMA equivalent of Wired-CDMA. For this new scheme, an Atom model and a detection process is suggested and examined. The chapter ends with the presentation of computer simulation results of the error rate in Wired-FDMA.

Chapter 8 introduces the *WirelessBus*, as the wireless equivalent of the Mesh-Bus scheme. The origin of the WirelessBus is the similarities between the NOR-bus and the OOK modulation scheme [20], as a special case of the analog ASK modulation scheme. Wireless communication systems use analog signals for data transmission, and assumes a linear (time-variant) channel models [21]. To address the problems which arise because of the differences between the wireless channel model and the channel model of the Mesh-Bus, a solution, namely *active listening/virtual transmission*, is suggested. This

chapter concludes with presenting the hardware test bed we developed for the WirelessBus, and will be used for experimental evaluation of this scheme in future work.

CHAPTER 2

LITERATURE REVIEW

Network Topology and Embedded/Field Bus Protocols

In this section we explore different internetworking topologies that most of the current communication networks are based on, and try to emphasize proper features of a specific one, the *bus* [10] topology. Next, we identify a set of important characteristics of the buses and categorize them. This sections ends with a brief exploration of some of the most popular buses.

Bus Topology versus the Others

When it comes to the interconnection of more than two devices in a way that they become able to communicate together, there is not much variety of topology choices. Such a topology should be chosen based on some requirements such as: either the communications are bidirectional or unidirectional, or whether the communications are needed to be possible for every pair of the devices or just among a specific subset of them. The devices are needed to get connected together either pair wise and form a fully-connected mesh, or all the communications no matter from which node toward which one, must pass through a single device which acts as a hub, which in this case they form a star topology. A bus is another popular interconnection topology. A bus is formed when the devices share a common communications medium, and are able to differentiate the communications and specify the ownership of the information content. There are other topologies such as a ring topology, which are outside the scope of our study.

The first and simplest topology is the *mesh* [4] topology. A mesh can be a fully- or partially-connected mesh. Figure 2 shows the case of a fully-connected mesh, where every single pair of devices or nodes has a direct and dedicated connection. In such a topology, there is no need for a management scheme for the interconnection of the nodes. If the pair-wise connection of the nodes supports full duplex communication, every single

node is capable of initiating a communication session with every other node in the network, at any desired time.

The problem with such topologies is the large amount of resources they consume in order to form. In a fully-connected graph of size of n nodes, number of the communication links has complexity of $O(n^2)$ [10], which is a quite big growth rate. For example, connecting 10 nodes with an average distance of 1 m requires 45 m of wire and 90 connection interfaces. When number of nodes double to 20, the required quantities grow to 190 m of wire, and 380 connection interfaces, which is a quadratic growth rate.

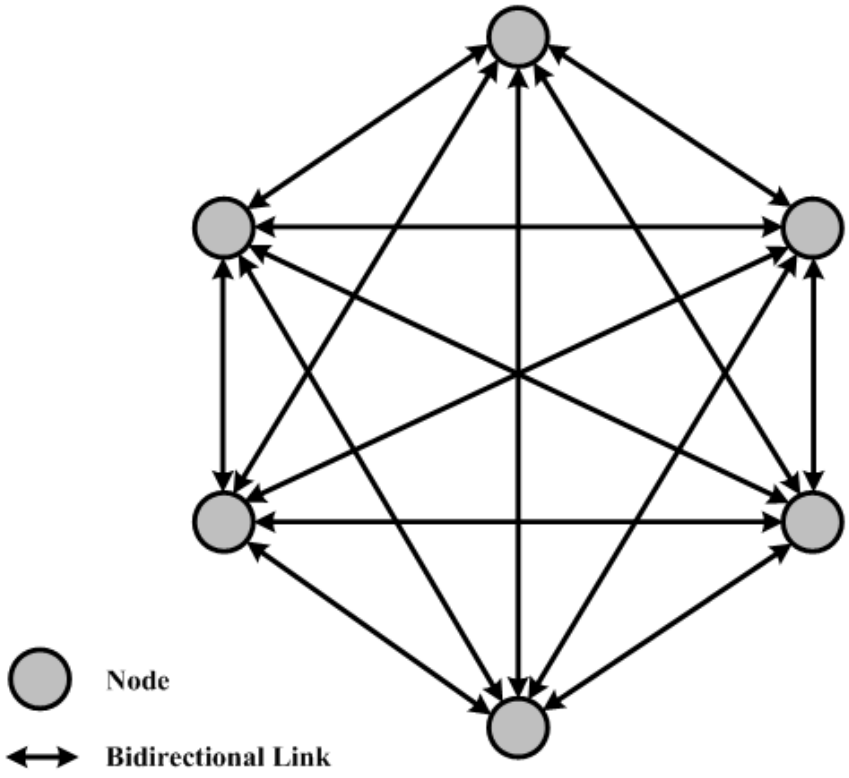


Figure 2 A fully-connected mesh topology. In a fully-connected mesh with n nodes, every node has a direct and dedicated connection to every $n-1$ other node of the network.

In addition to complete connection between all nodes that the mesh topology provides, it has another very important feature. Namely, all the nodes in such a network could be identical. In other words, the full mesh topology allows that the functionality of all nodes of the network to be the same. There is no need for special node to manage the operation of the network.

Another popular interconnection topology is the *star* [2] or *hub* topology, which in fact is a special case of mesh topology. Here, there is at least one special node in the network, and this node has a direct and dedicated link to every other node in the network. This special node or *hub*, acts as a bridge between different nodes, and manages the connections of all the nodes in the network. Figure 3 shows a star topology interconnection.

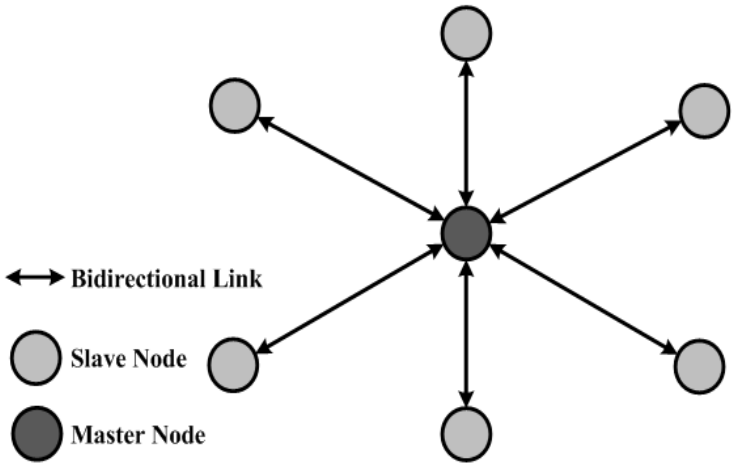


Figure 3 A Star topology. The dark gray node is the hub or the master of the network. It manages all the communications of the network. Communication between every two nodes occurs through the master.

Clearly, since adding a new node to the network requires adding one direct link to the hub, therefore the resource growth is $O(n)$. In this configuration, the hub node is a critical node, and must have enough processing power, bandwidth, and number of the hardware interfaces to service all the other nodes on the bus.

All the communication between the nodes must occur through the hub. Thus, on average, for n nodes, each node gets $1/n$ of the hub's time and other resources.

Another interconnection topology is the *bus* topology. In such a topology, every node in the network shares a common medium (e.g., set of wires.) Therefore, adding an extra node to the network does not cost any additional resource. However, since all the nodes use a shared resource. Thus, there must be rules to manage the access of the nodes to the shared communication medium. A popular medium access scheme uses the time sharing. In time sharing, every node has a time slot during which it is able to access the medium to read or write data to or from the bus. In order to avoid access interferences, the time slots must be mutually exclusive and non-overlapping. Figure 4 shows the bus topology.

Assigning the mutually exclusive time slots to the nodes of a bus is a highly important task which a special node on the bus decides about. The special node, namely the master node, is the orchestrator of the bus. Communication between the nodes is direct, and does not need to pass through the bus master. Still, the master has a crucial role in the operation of the network. Similar to the star topology resources are shared among the nodes. For n nodes on the bus, a node gets (on average) $1/n$ of the resources. Table 1 summarizes the features of each the three topologies.

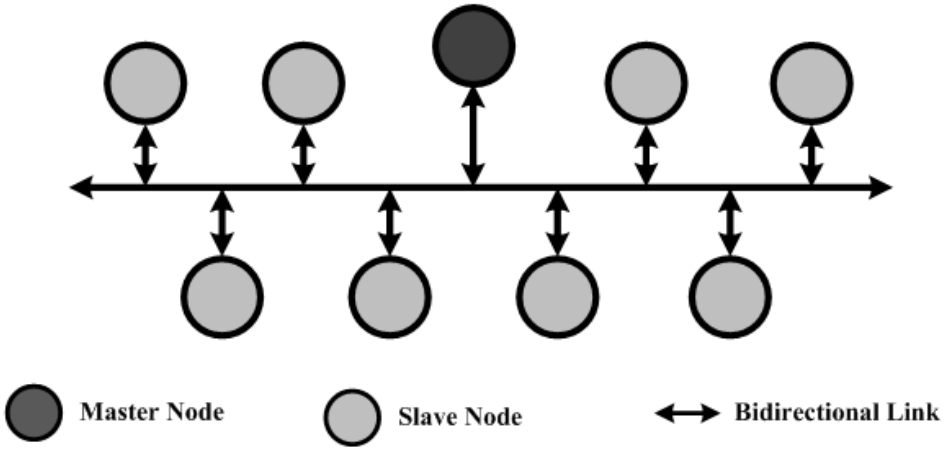


Figure 4 A bus topology. Similar to the star topology, the dark gray node is the master node which orchestrates all the communications which occur through the bus.

Table 1 Comparison between different interconnection topologies.

Topology	Resource	Distributed	Response Time
Mesh	$O\ (n^2)$	Yes	Any time
Star	$O\ (n)$	No	$1/n$ of times
Bus	$O(C)$	No	$1/n$ of times

Bus Types

The features of the bus topology makes it an attractive candidate for being used in embedded systems applications for interconnecting peripheral devices to the main microcontroller or to each other. Therefore, we will focus on the bus topology for the rest of the study.

The serial buses, based on the distance and data rate they can support, are divided in two groups. One group has some features that make the best option for interconnecting the microcontrollers to the peripheral devices, such as different types of sensors, external low volume memories and so on, which all are on the same board. We call them the embedded buses. The other groups of the buses, namely the field buses, usually are being used for longer distances, and off the board system-to-system communications.

Another important aspect of the serial buses is their dependence on a shared clock signal. Synchronous serial buses [22] are the buses that timing signal is provided to the nodes by the master of the bus. In contrast, there are another group of serial buses, the asynchronous [23] ones, which do not need for such a central timing signal. In the case of asynchronous buses, time synchronization between the nodes occurs through some special sequencing of data bits on the prior to every transmission.

In this study, our focus is on the serial buses. In addition to be a synchronous/ asynchronous or embedded/field bus, followings are some other important characteristics of the buses, which one should consider.

- Data rate
- Bus length
- Signaling type (differential, single ended)
- Number of the nodes on the bus
- Number of the data lines
- Voltage levels

Next, we will introduce some of the well-known buses, and the above mentioned parameters will be specified for them, to show their advantages and disadvantages.

Current Serial Bus Standards

There are many industrial standards for serial bus communication that today's devices use, some as very popular ones and some other as proprietary standards that only some licensed product are allowed to use.

RS- XXX Family: RS-485, RS-232, RS-422, RS-423

One of the most popular and widely used standards for serial communications and bus bases interconnection is Telecommunications Industry Association/Electronic Industries Alliance (TIA/EIA) [24], [25], family of asynchronous serial interfaces. The best-known members of this family are RS-485 and RS-232 standards [26]. Also, RS-422 and RS-423 [27] are other two standards that being used to some extent.

The most-widely used bus protocol of this family, RS-485, is a very flexible protocol. It can be used in both full duplex and half duplex configurations, and it is capable of supporting baud rates as high as $35\ Mbit/s$ over short distances. RS-485 uses differential signaling. That is, every bit is transmitted through an inverted signal on one wire, and non-inverted one on another wire. At the receiver, the difference between the two signals is calculated, and original signal is retrieved this way. In such a differential signaling scheme, the effect of interference on the wires signals are often nearly identical and get canceled at the receiver. Figure 5 shows the signal and noise in a differential signaling scheme.

The differential signaling scheme of the RS-485 protocol allows reliable communication up to $1200\ m$. The signal voltage levels of the standard are $-/+\ 7$ volts for such range of transmission.

The standard in full duplex mode has four signal lines and well as a reference pin. Similar to other members of its family, RS-485, encapsulates the ASCII values (7 bits of

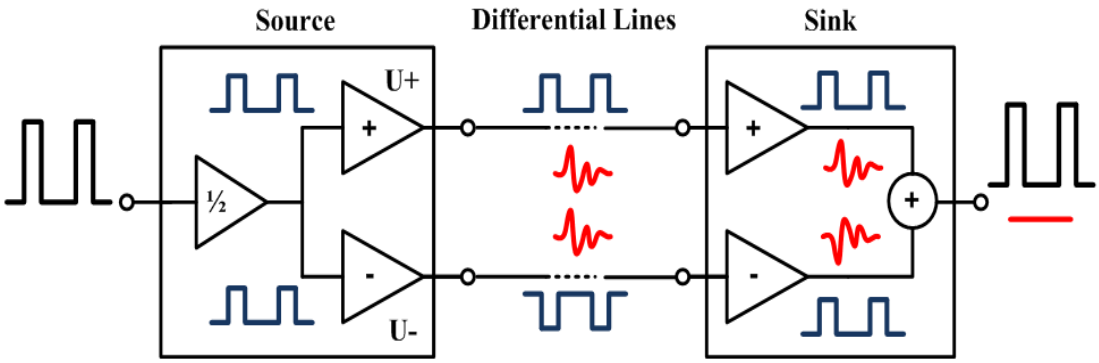


Figure 5 Differential signaling scheme. Since the end signal is the difference of the signals on the lines, the effect of noise on the line gets canceled out.

data) between a start bit and some (one or two) stop bits. Figure 6 shows an example of data communication in RS-484, where the sequence of is encapsulated between a start and a stop bit. In this figure, U_+ shows the non-inverted signal and U_- shows the inverted one.

As another widely-used member of TIA/EIA family, RS-232, is an asynchronous serial communication protocol, which is mainly used in point-to-point, short-distance

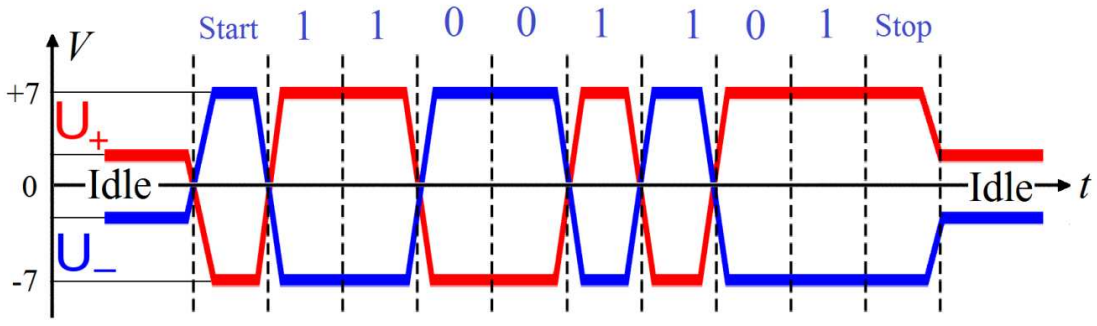


Figure 6 Differential signaling and data encapsulation of the binary string: in RS-485.

communications. Its signaling scheme, in contrast with RS-485, is not differential, and it does not support long distance communication. Its data rate is also much less that the RS-485. It supports the baud rate of for short distances, up to , and it is a full duplex protocol.

The two most common configurations of the RS-232 are 3-wires and 5-wires configurations. The 5-wire configuration provides the handshaking capabilities, which is a powerful flow control tool. In RS-232, as in RS-485, the data bits are packed in a data frame. For example, when transmitting ASCII characters, every character's bits are encapsulated between a *start* bit, one or two *stop* bits, and in some cases, as an error-detection mean, a *parity* bit. Figure 7 shows the data encapsulation and additional bits of data frame for RS-232.

The other two well-known members of this family, namely RS-422 and RS-423, are similar to RS-485 and RS-323 protocols. Both are high data rate protocols, where RS-422 uses differential signaling and supports distances up to , and RS-423 uses unbalanced (non-differential) signaling and supports distances up to .

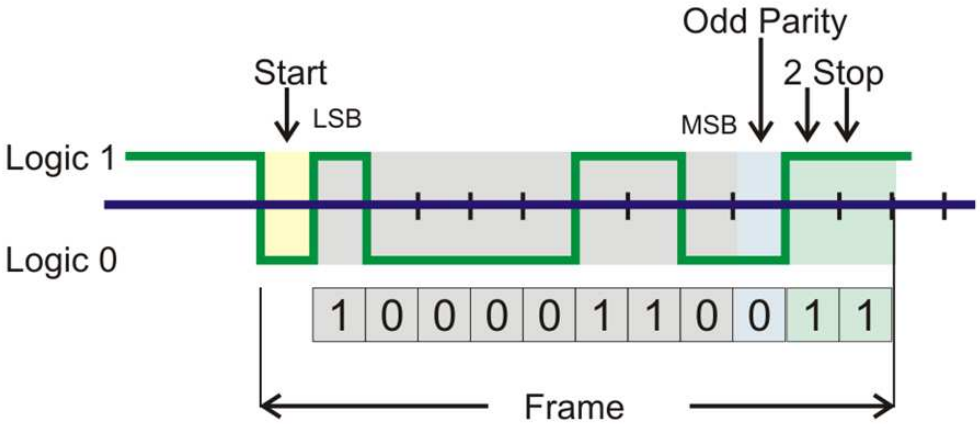


Figure 7 Data encapsulation for USART. It begins with a start bit, the data, one odd parity check bit, and finally two stop bits.

1-Wire Bus

1-Wire [28] [29] is another asynchronous bus protocol which is developed by Dallas Semiconductor Corporation. It is a bidirectional, half duplex, single wire bus protocol that has data rate of $15.4\,kbps$ in its standard mode. Every 1-Wire setting requires the presence of a master node on the bus. Communicate through 1-Wire bus, requires that every device has a factory-programmed, unalterable ID [1].

Figure 8 shows the internal circuit diagram of the nodes in a 1-Wire bus. As shown, in a 1-Wire communication setting, the bus itself is consisted of a resistively pulled up single strand of wire, as well as an implied ground wire, and all the nodes are connected to that wire through their open-drain interfaces.

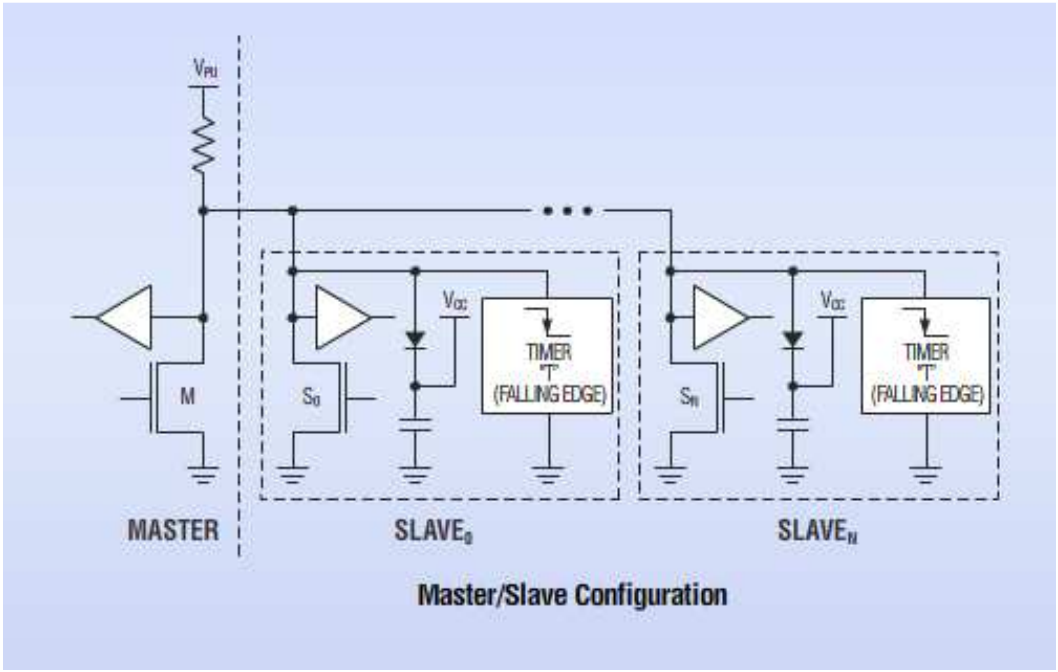


Figure 8 The circuit diagrams of the master and slave nodes in a 1-Wire bus [1].

In such a setting, every communication is initialized by the master, and all the slave nodes synchronize themselves by the falling edge of the clock sequence of the master.

In every time slot, a single bit of data is transferred, where a long duration of low pulse is interpreted as zero, and a short duration low pulse is binary one. The duration of long pulses in standard mode is $60\ \mu s$. Figure 9 shows the time slot in 1-Wire protocol, and Figure 10 shows the read/write request of the master node and the response of a slave.

Every communication in a 1-Wire setting has the three following phases of transaction; (1) device reset/synchronization, (2) ROM command sequence, and (3) function sequence. Figure 2.11 shows these three sequences, which are explained below.

1. Device Reset/Synchronization

   In this phase, the master pulls the bus low for more than 480 µs, and then releases it for the period of 15 to 60 μs. Now all the slave nodes pull the line low for 60 to

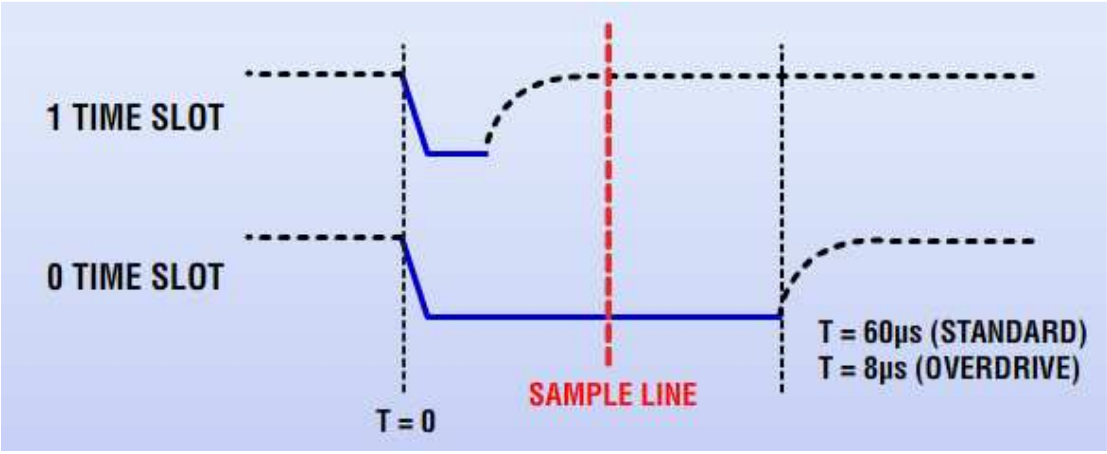


Figure 9 The time slot in the standard mode of 1-Wire bus [1].

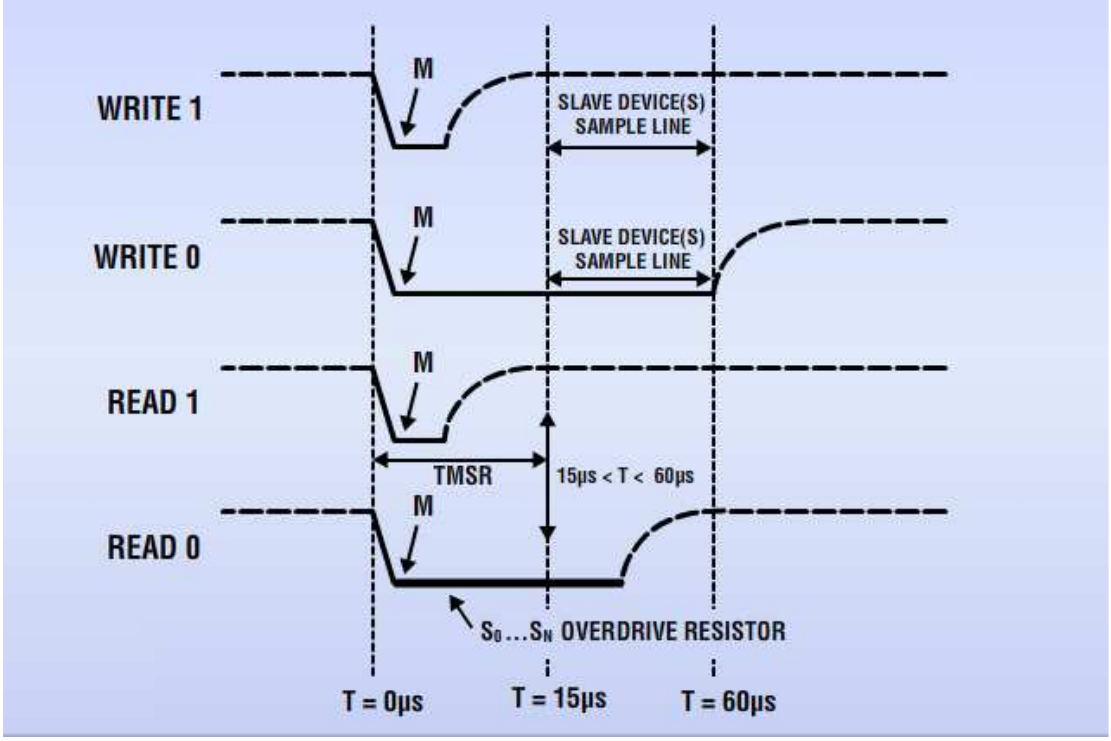


Figure 10 The master's read/write request in standard mode 1-Wire bus [1].

$240\ \mu s$. This is known as the presence pulse. By the end of the period of the presence pulse, the slaves release the line and let go back to high state or recovery. In such a state, all the nodes are in known synchronized state, and ready to communicate. Figure 12 shows the reset phase.

2. ROM command sequence

   In this phase, the master chooses a specific slave node based on its ID. In this way, all nodes one the bus one go to waiting mode and stay passive until the next reset phase. The ID which the master uses to identify and choose the target slave node is an unalterable, preprogrammed 64 bit string which all the 1-Wire capable devices must have.

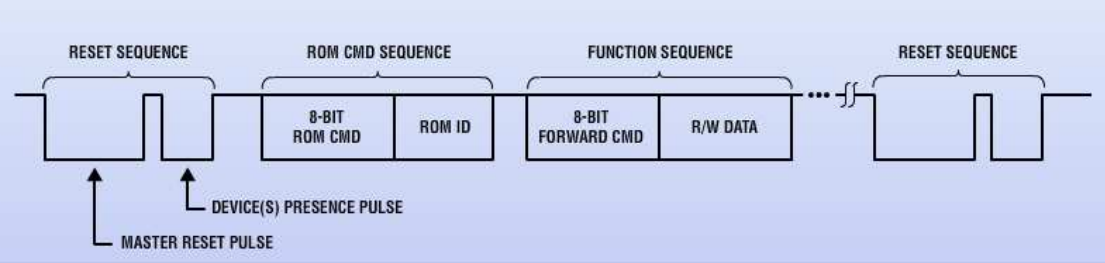


Figure 11 The three phases of communication on a 1-Wire bus [1].

3. Function sequence

   In the third phase, or the function sequence, the actual data transfer occurs through read and write requests initiated by the master, and responded by the selected slave node in phase two.

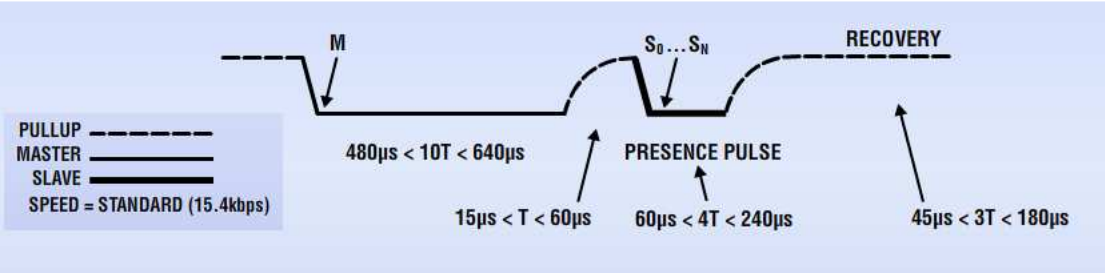


Figure 12 The first phase of 1-Wire communication: device reset/synchronization [1].

I^2C Bus

The Inter-Integrated Circuit, or I^2C [30], protocol is a synchronous serial bus originally developed by Philips Semiconductors in the 1980's. It is a low-data rate bus protocol which is widely used in current days' embedded systems in order to provide the required interconnection between the microcontrollers and the peripheral devices.

The I^2C standard has two bidirectional open drain interfaces which both are resistively pulled up to , one signal line (SDA) for half-duplex data communication, and the other one (SCL) for the clock signal. A ground wire is also required. Figure 13 shows the I^2C bus configuration. The configuration of the bus is a master-slave, and more than one master is allowed on the bus.

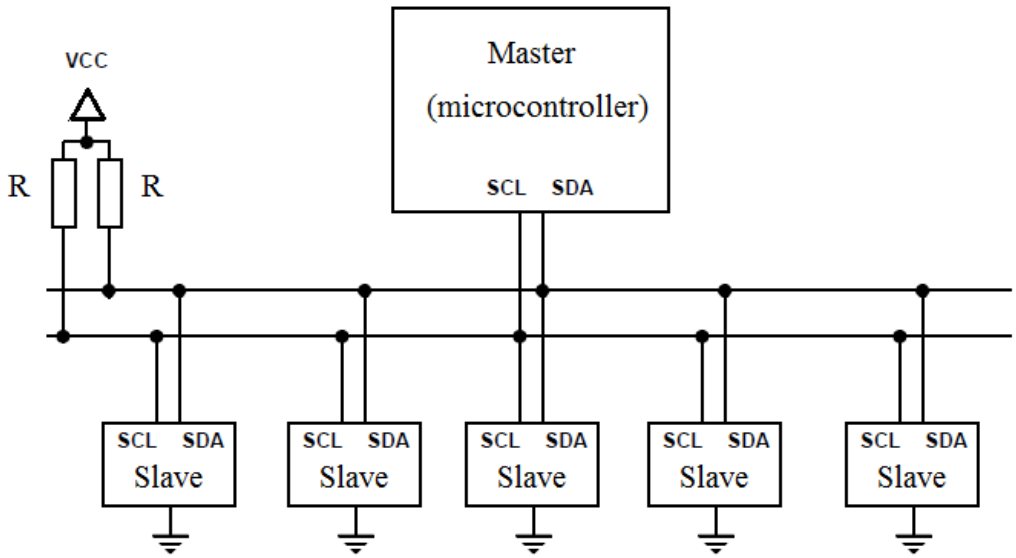


Figure 13 The bus. Both lines, the data and clock lines, are pulled up to through resistors.

For every communication, the master node puts the address of the slave node it wants to communicate with on the bus. All slaves on the bus see the address, but only the node that matches the address responds and the master-slave communication channel is formed.

In the I^2C protocol, the data is encapsulated between a start and a stop signal. As shown in Figure 14, both data and clock lines remain High when the bus is not busy. A High-to-Low transition of the data line, while the clock is High is defined as the start condition (see the shadowed area S). When a Low-to-High transition of the data line [30]occurs while the clock is High the stop condition take places (see the shadowed area P). The I^2C protocol has a 7 bits (10 bit) address space, and supports the data rate of 10 kbps to 100 $kbps$. The mechanism of error detection of I^2C bus is through ACK/NACK messages. Each byte of eight bits is followed by an acknowledgement bit (ACK). Upon transmission of the 8^{th} bit, the master releases the SDA line, which goes HIGH, the master generates an ACK clock pulse, and the slave acknowledges by pulling the SDA line low. A master receiver must generate an acknowledgement after the reception of each byte that has been clocked out of the slave transmitter. Also, a master receiver must

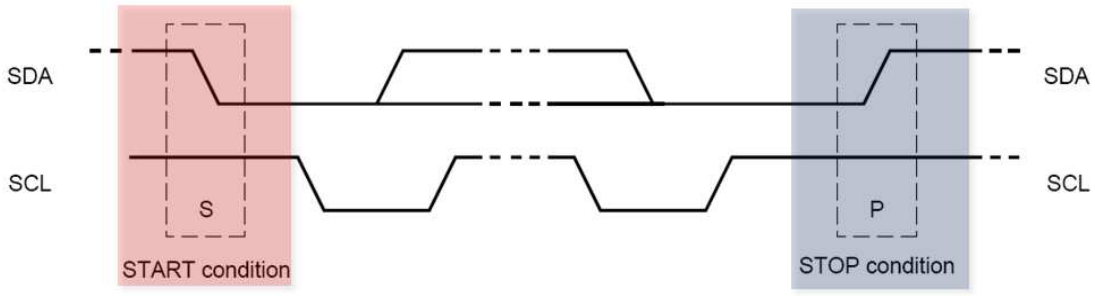


Figure 14 The start (the shadowed area S) and stop (the shadowed area P) conditions of I2C bus protocol.

generate an acknowledgement after the reception of each byte that has been clocked out of the slave transmitter. Figure 15 shows this process.

A master receiver must signal an end of data to the transmitter by not generating an acknowledgement on the last byte that has been clocked out of the slave (NACK). In this event, the transmitter must leave the data line HIGH to enable the master to generate a stop condition.

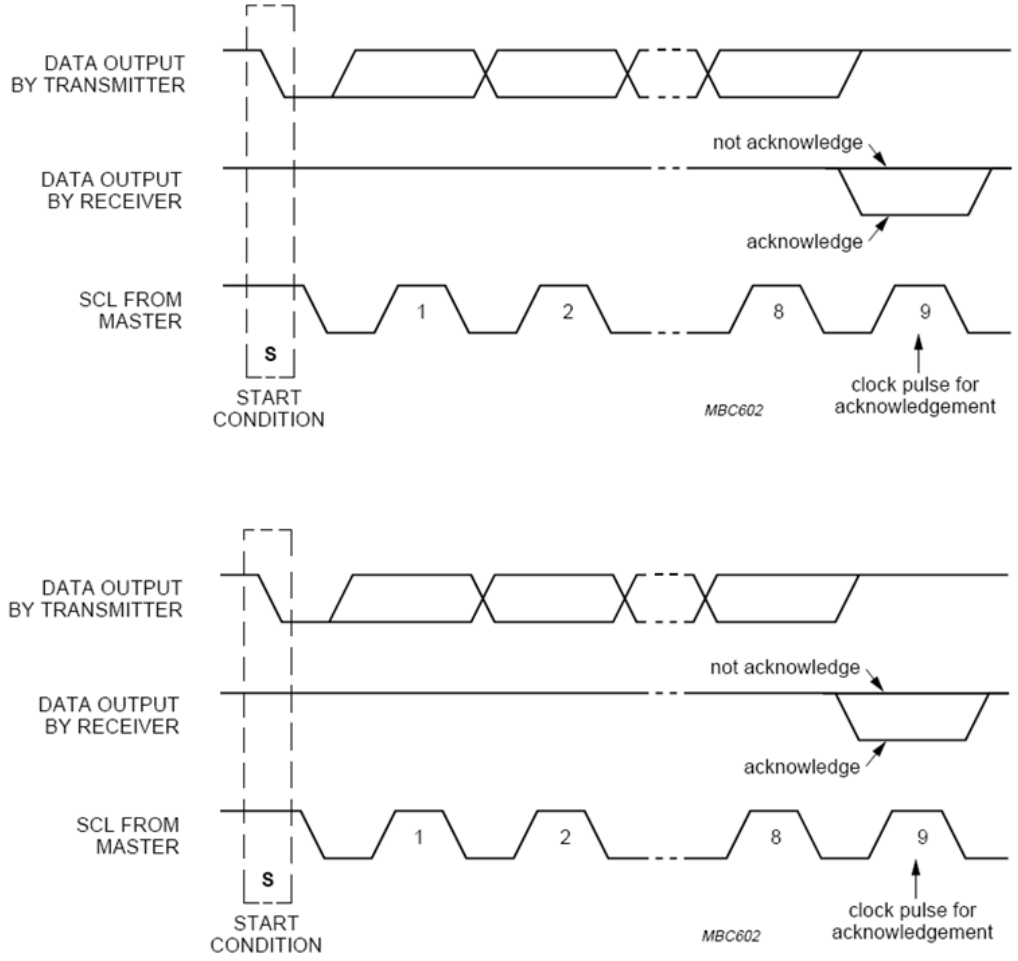


Figure 15 Process of data acknowledgement between the master and slave in the I^2C bus.

SPI Bus

The Serial Peripheral Interface, SPI [30], is a synchronous bus protocol. It is a full-duplex communication protocol, and the presence of a master node is necessary for every communication.

The main disadvantage of the SPI buses is that for every slave node on the bus, a chip select signal is required. Apart from the chip select signals, all the nodes on the bus share three signals of, serial clock (SCLK), master output, slave input (MOSI), and master input, slave output (MISO). Figure 16 and 17 show the SPI bus, as well as its signaling scheme.

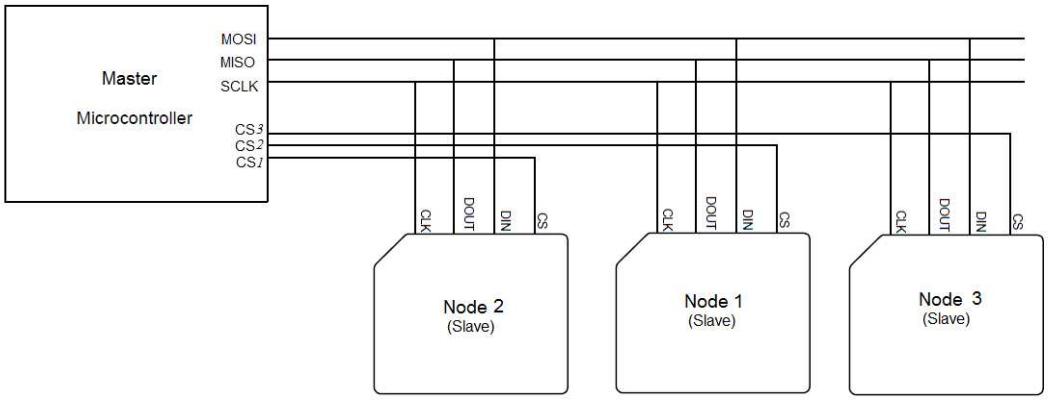


Figure 16 The SPI bus, and the data and chip select lines.

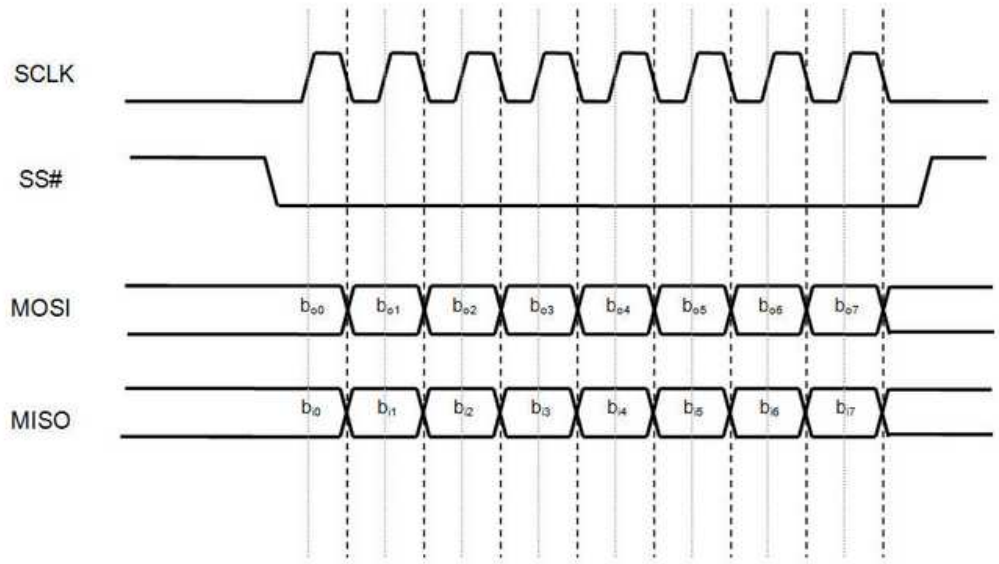


Figure 17 Signaling of the SPI bus.

Multiple Access Schemes

In situations where the communication medium as a resource (time, frequency spectrum, and physical links) is limited, the communicating devices must share the available resource. There are some well-known schemes that regulate this medium sharing process, which is called the multiple access schemes (MACs).

There are different multiple access schemes available such as TDMA, FDMA, CDMA, SDMA [31], OFDM [32] and so on. In the following sections, we briefly introduce the most important ones of them, and show some of their features. Also, Figure 18 conceptually summarizes some of these multiple access schemes.

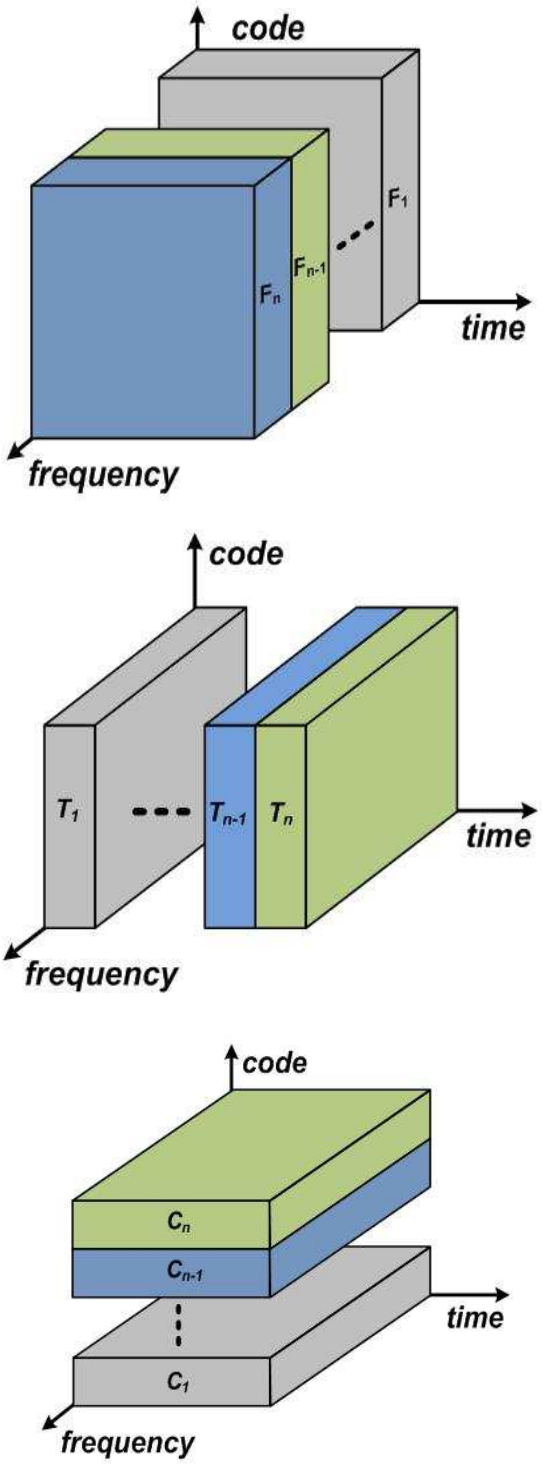


Figure 18 The multiple access schemes of FDMA, TDMA, and CDMA.

TDMA

TDMA [6], which stands for time division multiple access, is a popular multiple access scheme. TDMA is based on the concept of time sharing, and requires a predetermined setting for every node that wants to use such a scheme. The total available time in which the communication media is accessible is divided to specific number of non-overlapping time intervals or time slots. Every node in the scheme is allowed to access the bus only during the time slot which is assigned to it.

The structure of such scheme mandates that at any given time instance, only one node access the medium i.e., just one node is allowed to put data on the bus. However, if the time slots are designed short enough that all the nodes be able to have adequate number of time slots frequently enough for their data communications, one can say that every node has access to the medium all the time, and in essence, the nodes share the medium.

In such scheme (in contrast with FDMA, see next section) the every node is allowed to occupy all the frequency spectrum that such a TDMA scheme allocates, and usually (in contrast with CDMA, see the following section) the transmitted data is not specifically coded for the communication channel.

Consider a signal $S_i\ (t)$, where $i\ \in \{\ 1, 2, \dots, N\}$ and assume its information content is confined to time duration of T_n defined as follows:

$$S_i(t) = \begin{cases} info & 0 \le t\ \le T_n \\ 0 & T_n < t\ < T \end{cases}$$

Where, $T_n \le \left\lfloor \frac{T}{N} \right\rfloor$

Then, is a timed-delayed non-overlapping summation of the signals arranged side by side in a time frame, as Figure 19 shows.

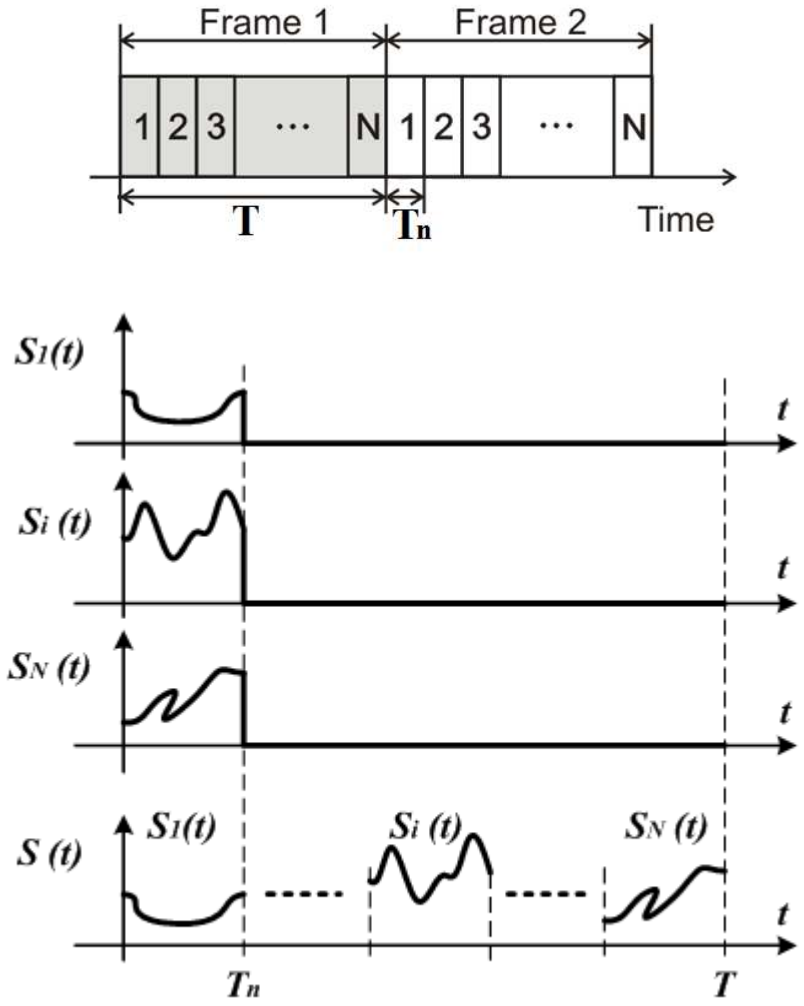


Figure 19 The frames of TDMA with *N* time slots. The time slots do not overlap with each other, and usually there are some guard time between every two time slots.

FDMA

The next shared medium access scheme, frequency division medium access, FDMA [7], takes advantage of the available frequency spectrum by forming different communication channels in different frequency ranges. In this way, every pair of nodes can have an all-time accessible communication channel.

FDMA operates by modulating the different baseband signals to non-overlapping frequency bands, and transmits all of them together. Since the information transmissions are at different frequency bands, they do not interfere with each other. Of course, the frequency separation must be chosen in a way that the frequency bands have minimum possible overlapping when they get filtered out at the receivers.

Assuming that signal $s_i(t)$ with the frequency spectrum of $S_i(f)$ with bandwidth less that $B/2$ Hz ($S_i(f) = 0,\ |f| \geq B/2$), One can up-convert it to the frequency f_i, where:

$$\forall\, i,j\ \in \{1,2,\dots,N\},\quad \left|f_i - f_j\right| \geq B$$

Then, it is guaranteed that the superposition of all the signals, $s(t)$, with frequency spectrum of $S(f)$ as follows, can be transmitted without any interference by other signals.

$$S(f) = \sum_{i=1}^{N} S(f - f_i)$$

Figure 20 show the process of up-conversion, transmission, filtering, and down conversion of hypothetical signal $s_1(t)$. Part (a) shows the frequency spectrum of the band limited signal as $S_1(f)$ which is a baseband signal with bandwidth of B Hz. Part (b)

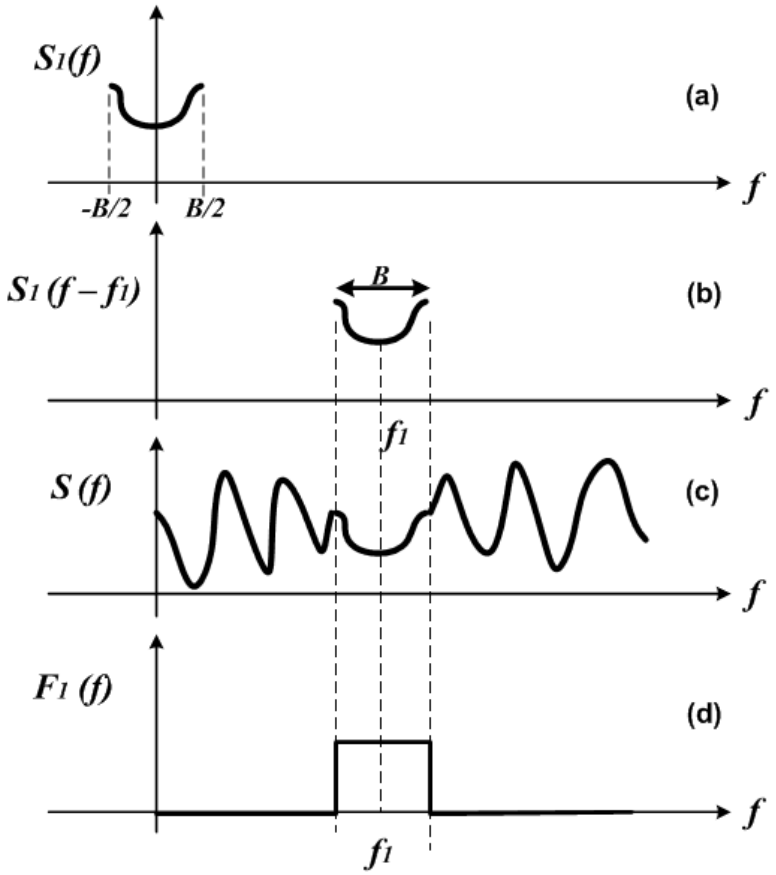


Figure 20 The processes of FDMA. (a) The baseband signal (b) up-converted signal to frequency f_1 (c) the occupation of frequency spectrum, (d) the filter at the receiver.

shows the very same signal translated to the center frequency of f_1. The next signal, $S(t)$, shown in part (c), is the superposition of all the *N* translated (up converted) signals. Part (d) shows filter $F_1(f)$, a band pass filter with the bandwidth of B, which is the same bandwidth as $S_{1(f)}$. Such a filter is designed to filter out every frequency components of $S(t)$ except for the ones for $S_1(f - f_1)$. The result of down-conversion of band pass $S_1(f - f_1)$ to its baseband equivalent is the $S_1(f)$, the original signal.

Clearly, that as long as the information content of every channel is limited to the specified frequency limit, all the nodes are capable of transmitting and receiving their

signals at any time instant they want to (in contrast with TDMA), without any channel coding (in contrast with CDMA.)

Examples of FDMA are very widespread: first generation of mobile telephone communication, ordinary commercial radio, TV all use FDMA as their multiple access schemes. Further, OFDM, as a variation of FDMA with overlapping frequency channels, is widely used in modern wired and wireless application, such as the various variants of DSL technologies (e.g., ADSL, HDSL, VDSL) as well as WLANs (e.g., IEEE 802.11a/n), WMAN (WiMAX or IEEE 802.16), LTE, DVB, DAB and so on.

CDMA

Code Division Multiple Access, CDMA [8], as another multiple access schemes, allows the simultaneous access of the medium through coding the information by some *spreading* sequences. In this scheme, every node has a unique code, or spreading sequence, and though this code, it is allowed to transmit its data at any time instant. Such codes usually have much higher bandwidth than the actual data; therefore, when the data is coded by them, the frequency spectrum of the data expands as well, here comes the name *Direct Sequence Spread Spectrum* [33] [34] technique. The advantage of Spread Spectrum Spread Spectrum (DSSS for short) is providing better signal to noise ratio[35], as well as multiple access communication channel through in case of CDMA i.e., it allows multiple accesses to medium if the codes of the spreading sequences are chosen properly.

In a DSSS scheme, if the codes are chosen as follows, they can be used as codes in a CDMA scheme to provide multiple accesses to a communication system.

1. Every pair of codewords in a code is orthogonal
2. The codewords of every code is normalized by its power

Such codes are called orthonormal [33] codes. The following formulas show the continuous and discrete forms of the orthonormality.

$$< C_i , C_j > = \int_{-\infty}^{+\infty} C_i(t) \cdot C_j^*(t)\, dt = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

$$< C_i , C_j > = \sum_{k=-\infty}^{+\infty} C_i[k] \cdot C_j^*[k] = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Assuming that there are n pairs of transmitter and receivers in a CDMA scheme, and they want to transmit their messages. Transmitter i, multiplies its message $m_1(t)$ by its unique code $c_1(t)$ and transmits it. Therefore, the transmission of node i, is $S_i(t) = m_i(t) \cdot c_i(t)$. Assuming all the n nodes do the same thing, the total transmitted signal of all of the transmitters together is as follows:

$$S(t) = \sum_{i=0}^{n-1} S_i(t) = \sum_{i=0}^{n-1} m_i(t) \cdot c_i(t)$$

$$= m_1(t) \cdot c_1(t) + \cdots + m_i(t) \cdot c_i(t) + \cdots + m_n(t) \cdot c_n(t)$$

At the receiving end, if a receiver wants to receive the message of the transmitter i, it calculates the inner-product of the $S(t)$, and the $c_i(t)$ through its correlation-based detector, and the result is as follows:

$$< S, c_i > = < \sum_{i=0}^{n-1} m_i(t) \cdot c_i(t) \, , c_i > =$$

$$< \{m_1(t) \cdot\ c_1(t) + \cdots +\ m_i(t) \cdot c_i(t) + \cdots +\ m_n(t) \cdot\ c_n(t)\}, c_i\ > =$$

$$m_1(t) \cdot< c_1, c_i\ > + \cdots +\ m_i(t) \cdot< c_i, c_i\ > + \cdots +\ m_n(t) \cdot< c_n, c_i\ >$$

$$= m_i(t)$$

Figure 21 shows the CDMA process transmission and detection processes

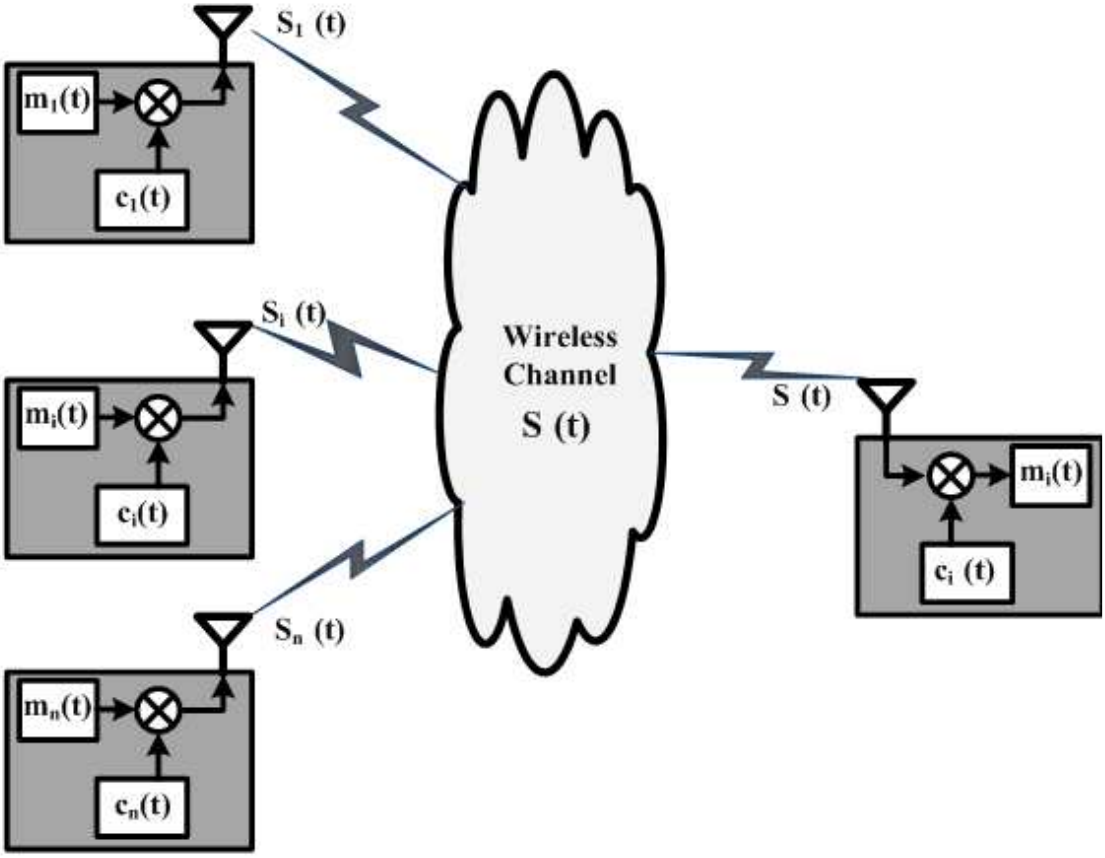


Figure 21 The processes of transmission and detection in CDMA scheme.

There are groups of well-studied codes for CDMA scheme with very interesting properties such as Walsh codes, Maximal Length sequences, Gold Codes, and etc., which will be explored in chapter 4.

Z-Channel

There several models for a digital communications channel. For example, binary symmetric channel (BSC), binary asymmetric channel (BAC), binary erasure channel (BEC), and so on [36]. The choice of the proper model depends on the characteristics of the application. Figure 22 shows the general model of a binary channel and associated error probability.

In our study, since we are going to use a NOR channel (NOR-bus will be introduced in next chapter). It is a nonlinear, asymmetric channel, which is best modeled with the binary asymmetric channel.

In a binary symmetric channel, where $p_0 = p_1$, the probability of a '0'→'1' error is the same as the probability of a '0' →'1' error. On the other hand, in a binary asymmetric channel, in contrast with binary symmetric channel the probability of error

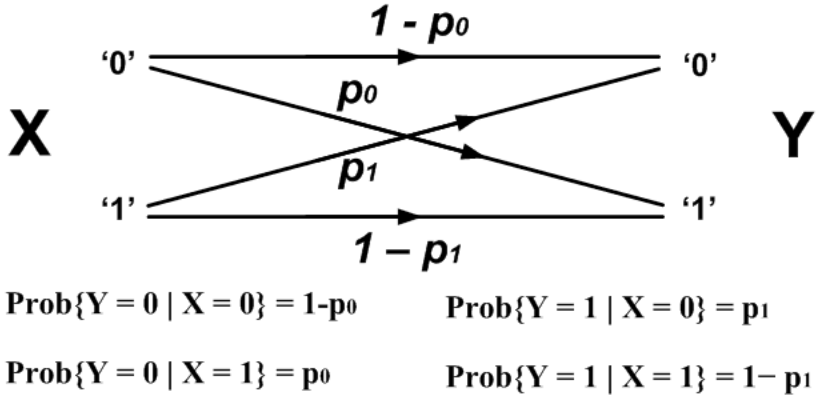


Figure 22 The Binary symmetric channel model. X is the input random variable, and Y is the output random variable.

for ‘0’→ ‘1’ is not the same as ‘1’ → ‘0’, therefore it is a asymmetric channel.

A special case of the binary asymmetric channels, where $p_0 = 0$, (the probability of ‘0’→ ‘1’ error is zero,) the shape of the error model of the channel is in form of English letter ‘Z’; therefore, it is named Z-Channel [15]. Figure 23 shows the Z-Channel model and the error probability model of it is as follows:

$$Prob\{Y = 0 \mid X = 0\} = 1$$

$$Prob\{Y = 0 \mid X = 1\} = p$$

$$Prob\{Y = 1 \mid X = 0\} = 0$$

$$Prob\{Y = 1 \mid X = 1\} = 1 - p$$

The Z-channel finds application in modeling the errors in storage media such as disk drives and RAM chips.

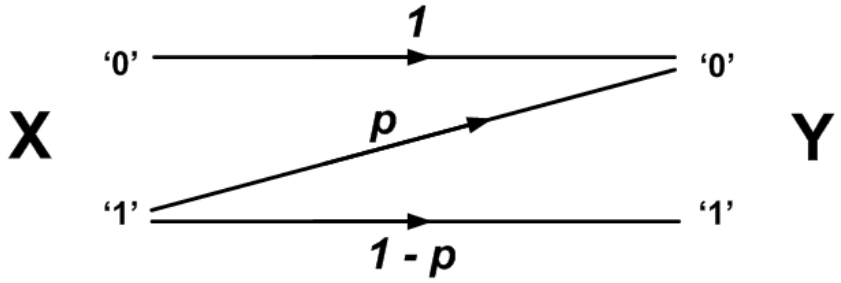


Figure 23 The Binary asymmetric channel model. X is the input random variable, and Y is the output random variable.

Optical Orthogonal Codes (OOCs)

The unipolar optical orthogonal codes [12] [13], which will be used in this study, are a group of codes that are used in unipolar fiber optic communications. An OOC code is a group of binary (‘0’ and ‘1’ bits) strings with general notation of:

$$(n, w, \lambda_a, \lambda_c) - OOC$$

Where n the length of the string is, w is the Hamming weight (number of the ‘1’ bits in the string) of the strings, and λ_a and λ_c are circular autocorrelation and cross correlation functions of the codewords, respectively. λ_a and λ_c are defined as follows. Assuming $c^i = (c_0^i, c_1^i, \ldots c_{n-1}^i)$ and $c^j = (c_0^j, c_1^j, \ldots c_{n-1}^j)$, (where $c_l^k \in \{'0', '1'\}$,) are two distinct codewords of $(n, w, \lambda_a, \lambda_c) - OOC$. The circular autocorrelation function of a codeword defined as follows. For every codeword $c^i \in (n, w, \lambda_a, \lambda_c) - OOC$

$$ACF(\tau) = \sum_{\substack{l=0 \\ \tau \neq 0}}^{n-1} c_l^i \cdot c_{l \oplus \tau}^i \leq \lambda_a$$

Since this formula is a circular function, $l \oplus \tau \triangleq (l + \tau) \bmod n$ and for the special case of $\tau = 0$, we will have

$$ACF(0) = \sum_{l=0}^{n-1} c_l^i = w$$

For circular cross correlation, the definition is as follows:

$$CCF\ (\tau) = \sum_{\substack{l=0 \\ i \neq j}}^{n-1} c_l^i\ .c_{l\oplus\tau}^j \leq \lambda_c$$

The Figures 24 shows the autocorrelation function and cross correlation function a $(n, w, \lambda_a, \lambda_c) - OOC$ respectively.

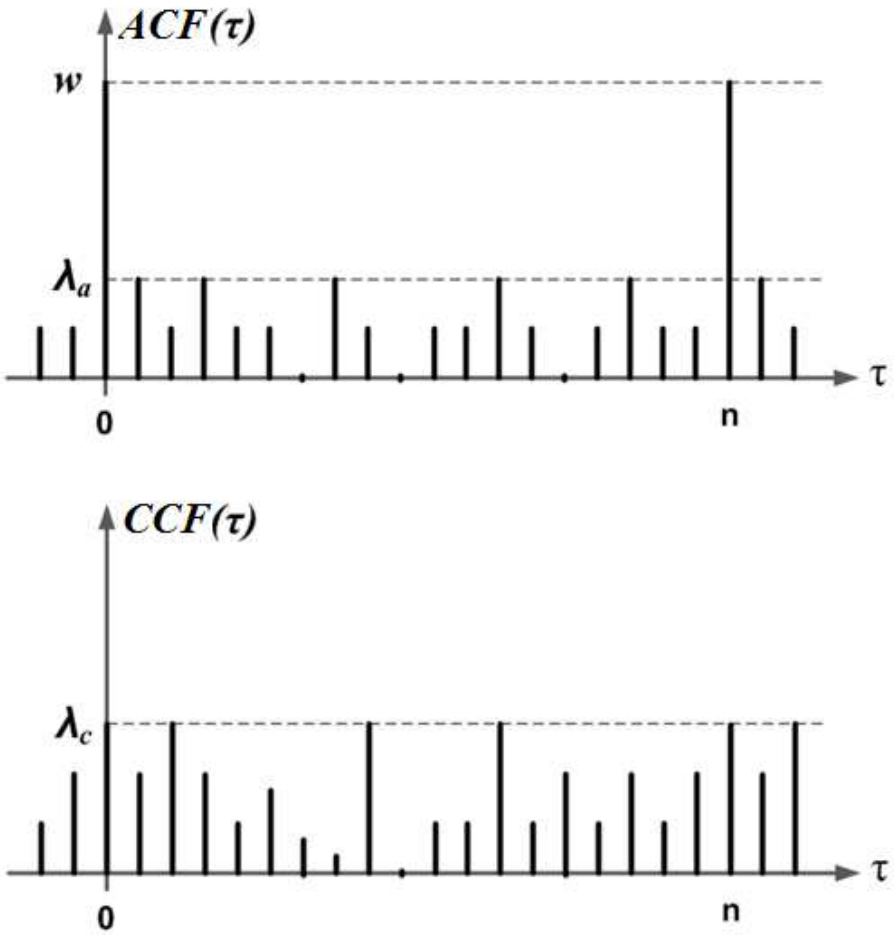


Figure 24 The autocorrelation function (top,) and cross correlation function (bottom) of the $(n, w, \lambda_a, \lambda_c) - OOC$.

The number of the codewords in a $(n, w, \lambda_a, \lambda_c) - OOC$ is called the *cardinality* of the code, and is denoted by $|C|$. On the other hand, the maximum possible number of an OOC code ($|C|_{max}$) is denoted by $\Phi(n, w, \lambda_a, \lambda_c)$. In general, there is no formula for calculating the maximum cardinality of an OOC, but as [12] shows, it is possible to derive the following upper bound (based on the Johnson upper bound for error correcting codes [37]) and lower bound (based on the greedy algorithm for construction of OOC's) for the cardinality of the $(n, w, \lambda_a, \lambda_c) - OOC$.

$$\phi(n, w, \lambda_a, \lambda_c) \leq \frac{(n-1).(n-2)\ldots(n-\max\{\lambda_a, \lambda_c\})}{w.(w-1)\ldots(w-\max\{\lambda_a, \lambda_c\})}$$

$$\phi(n, w, \lambda_a, \lambda_c) \geq \frac{\binom{n}{w} - \frac{n-1}{2}\binom{w}{\lambda_a+1}\binom{n}{w-\lambda_a-1}}{n.\sum_{i=\lambda_c+1}^{\min\{n-w,w\}}\binom{n-w}{w-i}\binom{w}{i}}$$

When the number of discovered codewords for a $(n, w, \lambda_a, \lambda_c) - OOC$ reaches the maximum possible (i.e., $|C| = |C|_{max} = \Phi(n, w, \lambda_a, \lambda_c)$), the code is considered an *optimal* [13, 38] code.

CHAPTER 3

THE MESH-BUS

Introduction

Experimental research in geosciences often involves deploying a number of sensors along with an attendant data logger. Deploying redundant sensors can enhance data reliability and quality. However, increasing the number of the sensors also increases the complexity and cost of connecting the sensors to the data-logging equipment.

By far the most common method of connecting sensors to a data logger is through a star configuration: the logger is at the canter and there is a terminal block with a set of connectors and wires dedicated to each sensor. This works well with a few sensors, but becomes problematic for a large number of sensors. For one, a user may not have enough ports on the data logger. Secondly, the number of wires one has to manage becomes unwieldy and expensive. The expense of high-quality cabling that will hold up to harsh weather conditions comes as a surprise to those that have not deployed and operated environmental field equipment.

An alternative to the star configuration is a bus where multiple sensors share a common set of wires that provide a shared communication channel, and may also supply power. Figure 25 depicts this arrangement. Since the bus is a shared resource, sensors on the bus need a compatible electrical or physical (PHY) interface. Sensors also need a scheme for medium access control (MAC) to the bus [17]. The payoff for this added complexity is lower cabling costs and simpler cable management.

Buses [10] are widely used in embedded systems. A non-exhaustive list of bus standards are I^2C, SPI, CAN, Modbus. Some of these busses such as SPI are CPU/MCU busses, designed for interfacing peripherals located in close proximity (often on the same PCB) to a microcontroller. Other busses such as CAN are field busses designed for integrating heterogeneous sensors and actuators into a larger system. Field busses are

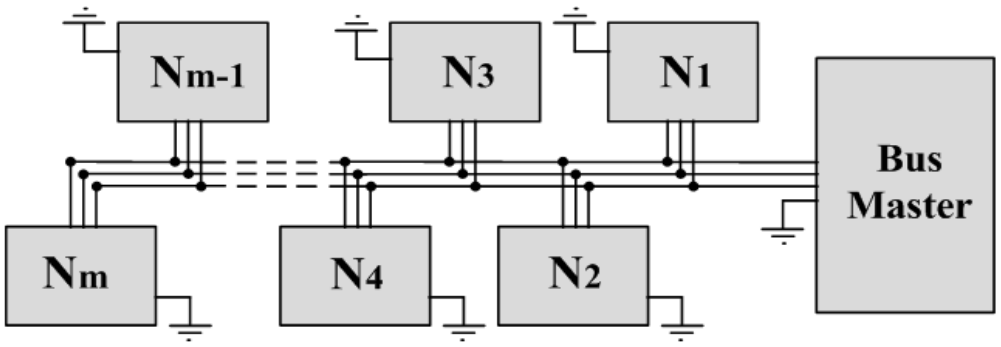


Figure 25 Traditional TDMA-based buses use sets of wires that form the buss, and the nodes on the bus must be configured as one Master and some Slave nodes.

widely used in industrial and automotive applications. *Field busses* are fully developed, providing definitions for the physical (PHY) layer, medium access control (MAC), data link layer, application layer, and so on. There are also bus standards that are not full field busses, but address the PHY and to some extent the MAC layers. An example is the RS-485 bus standard that is used in industry and also finds application in environmental sensing.

The aforementioned busses use Time Division Multiple Access as Medium Access Control (TDMA-MAC) to manage the access of the sensors to the common bus. Only a single sensor is allowed to transmit on the bus at any given time. Either sensors have pre-assigned time slots or, perhaps more commonly, there is a bus master that manages the other devices' (slaves) access to the bus.

The use of TDMA for MAC has inherent problems. Only one user at a time can transmit on the bus, so that time synchronization is critical. The timing constraint dictates slew rates, cable capacitance and, consequently, cable length. Further, the master-slave

model does not allow for one-to-one communication between the arbitrary sensors, because all the communication must take place through the master node.

Mesh-Bus Setting, Bus-Modulator, and NOR Bus

In this study we develop new techniques that have the advantage of bus architecture, but without the TDMA limitation. The techniques provide the nodes transparent medium access to the bus. Thus, there is no need for special bus master that manages the bus. We call such a bus setting a *Mesh-Bus*. Figure 26 shows the Mesh-Bus setting. An important feature of the Mesh-Bus is that it is non-TDMA and multiple independent communications can occur simultaneously. Further, the bus is largely unmanaged. That is, subject to some broad parameters there is no need to set fixed communication parameters such as data. Rather, communication between one pair of nodes on the bus can occur at a different data rate than the data rate for another pair of nodes. Nodes can adjust their data rate dynamically. One can add or remove nodes from the bus with minimal impact on the operation of the bus. As it is shown in the figure,

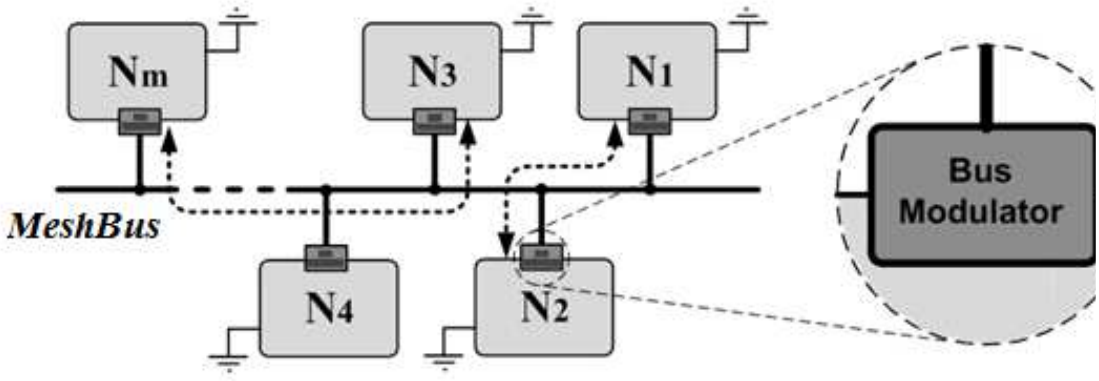


Figure 26 The Mesh-Bus setting. Every node has a intermediary device named the Bus-Modulator, through which connects to the bus

every node is connected to the single strand of wire, as the bus, through a device called *Bus-Modulator*. The Bus-Modulator is a bidirectional translator that provides physical communication services to the nodes.

Bus-Modulators have two interfaces. The high-level interface interfaces to a sensor through the sensor's native interface and protocol. For example, an anemometer may have an RS232 interface with specific Baud rate, and use plain ASCII string to communicate information. The anemometer will communicate with the bus via the Bus-Modulator's high-level interface. The anemometer will send and receive the data in ASCII. The Bus-Modulator will translate between the ASCII and Mesh-Bus format. The Bus-Modulator's low level interface is the actual hardware connection to the bus. This low-level interface is an open-drain of a FET or open-collector of a BJT [14] that connects to the Mesh-Bus. Figure 27 shows such a Bus-Modulator.

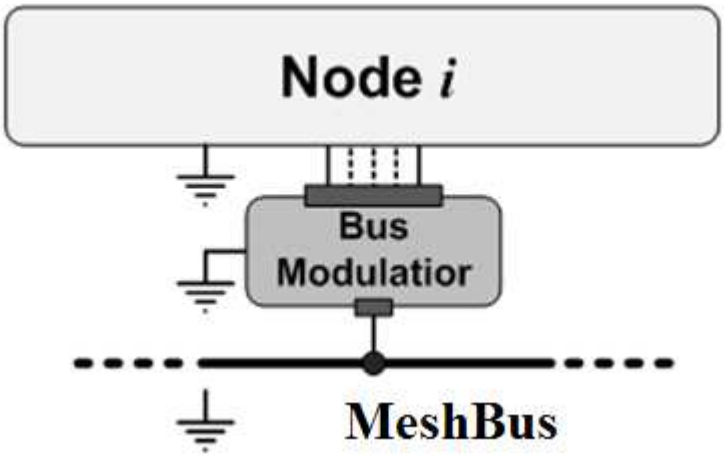


Figure 27 Bus-Modulators interfaces nodes to the Mesh-Bus. They provide high-level interfaces to sensors using the sensors' native interfaces and protocol. Bus-Modulators is a translator between the original data and the Mesh-Bus format.

The Mesh-Bus is a so-called NOR bus with the rest state of logic high (‘1’), which is provided through resistive pull up to the supply voltage. Figure 28 shows a hardware implementation of such a NOR bus through an open-drain configuration.

A NOR bus operates as follows. By default, the bus is the rest state and in logic low. When a node transmits logic ‘1’, it pulls the bus low ($\overline{1 \vee x} = 0 \wedge \bar{x} = 0$). When a node transmits a logic ‘0’ it does nothing, ($\overline{0 \vee x} = 1 \wedge \bar{x} = \bar{x}$). Figure 29 shows the effect on the NOR bus. Whenever either $N_i$or $N_j$transmits ‘1’, the bus is pulled to ‘0’.

The electrical interface for the NOR bus in the Bus-Modulator is extremely simple to implement in microcontroller. For example, it could be the input-output (I/O) pin of a microcontroller connected to the bus. When there is logic ‘1’ to transmit, the controller pulls the pin low. When there is a ‘0’ to transmit, the controller switches the

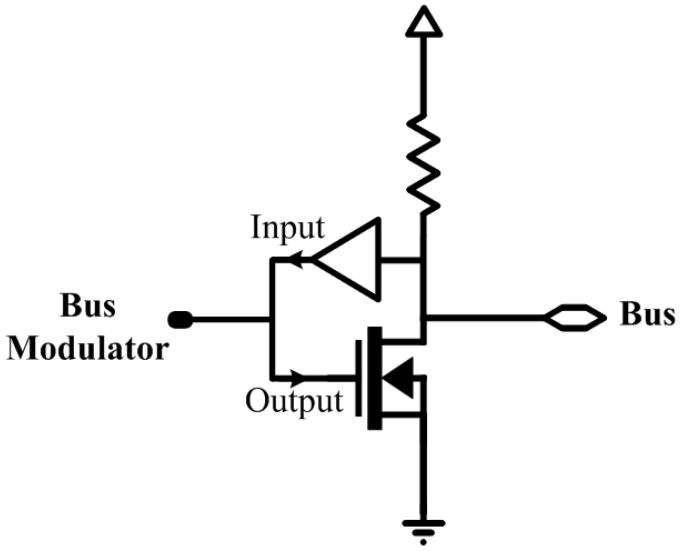
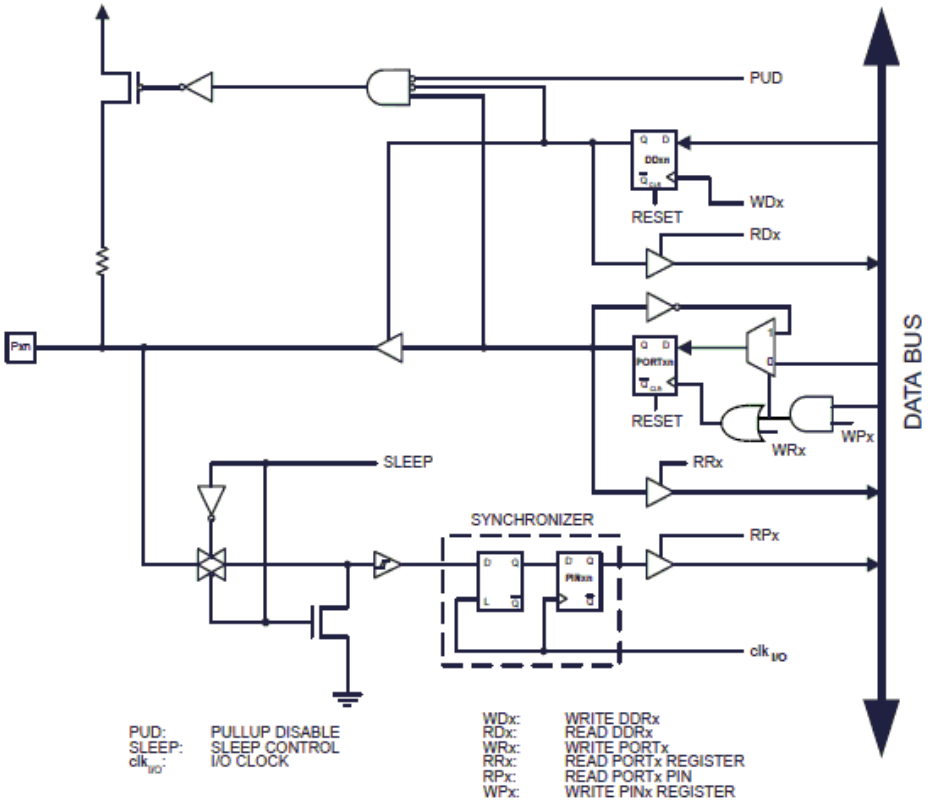


Figure 28 The open-drain interface of the Bus-Modulator. Such a configuration is very popular in microcontroller pins, so it is easily implementable using ordinary microcontrollers

pin to the high-impedance input (High-Z) state and the bus' pull-up resistor will pull the bus to the V_{cc}. Figure 29 shows the diagram of an I/O pin in Atmel AVR microcontroller [39] family.



DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

Figure 29 The diagram of general digital I/O of the ATMEL AVR microcontroller. The table shows the port pin configuration of such and I/O pin.

Operational Logic of Mesh-Bus

All the communications on a Mesh-Bus are based on the transmission and reception (detection) of messages called *Atoms*. Atoms are special types of messages carefully designed so that several can be transmitted concurrently on a bus with minimum interference between them. The lack of interference results from borrowing concepts of multiple access schemes from CDMA and FDMA.

In the most general case, an Atom, denoted as $Atom(n, w)$, is a family of binary string of length n, and the Hamming weight of w. The Hamming weight w is the number of positions where there is a '1' and the $n - w$ other position are '0'. We use *chip* as the unit of length in the Atoms, and based on the sampling rate of the Bus-Modulators, such chips could have various time durations. Figure 30 shows a sample Atom.

In the Mesh-Bus, every node has a set of unique Atoms. If node i, wants to transmit data to node j, it uses the Atom of node j to transmit the data. On the other end, node j continuously watches the content of the bus to detect the own Atoms, in case one has been transmitted by a node. Since the whole scheme is based on the uniqueness or

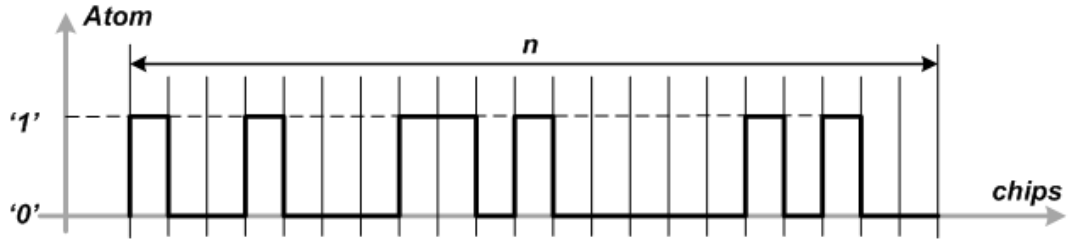


Figure 30 A sample Atom (21, 7). This Atom has length of 21 chips and the Hamming weight is 7.

distinguishability of the Atoms of the nodes, the choice of Atoms is a crucial task. Generally, Atoms are scarce, and for fixed values of n and w, there are limited number of Atoms that will provide low interference.

Detection process in a Mesh-Bus Setting

Similar to many detection schemes in communication system, our detection schemes use correlation. Correlation-based detectors compare the received signal with an expected pattern. Whenever such a comparison results in a maximum similarity between the received signal and the expected pattern, they declare the received signal a successful detection. Assume the expected pattern E is a binary string of length n as $E = (e_0, e_1, \dots, e_{n-1})$ and the received signal is an infinitely long binary signal of $S = (s_{-\infty}, \dots, s_{-1}, s_0, s_1, \dots, s_{n-1}, \dots)$. Every time the receiver receives a new sample s_i, it adds it to the end of the list of $n-1$ last samples and computes the cross correlation between such a list and the expected signal E through the following formula:

$$CF\ (i) = \sum_{j=0}^{n-1} s_{i-j}\, e_{n-1-j}$$

If such a cross correlation function reaches a maximum at index i, there has been a successful detection. This means that a transmitter message started transmitting $n-1$ chips before the reception of sample i, and the i-th sample was its last chip.

The computational complexity of such a n-chip cross correlation is $O(n^2)$, because for every new sample it computes n multiplications, and every message has n samples (chips).

In bipolar binary systems, where the channel alphabet consists of '-1' and '+1' symbols, the cross correlation can be interpreted as follows:

$$\sum(\#\text{ of positions } E \text{ and } S \text{ are the same}) - \sum(\#\text{ of positions } E \text{ and } S \text{ are different})$$

The detection occurs when the cross correlation reaches the value n. That is, all the positions are the same, and number of the similar positions add up to n.

In contrast with bipolar binary systems, in a unipolar binary system, where the alphabet symbols are '0' and '1', if the E and S differ in a position, their product is zero (in bipolar it's '-1'); therefore, the cross correlation function is as follows:

$$\sum \#\text{ of positions } E \text{ and } S \text{ are the same}$$

The computational complexity of such a unipolar binary system is $O(wn)$; because, for every new sample, just w multiplication is required to be checked and the rest are zero. Considering the fact that usually w is chosen to smaller than n by one or two orders of magnitude, this detection scheme is almost linear, compared to the quadratic complexity of general correlators.

Coding: *M*-ary versus Unary

The obvious and simplest scheme to communicate data in an Mesh-Bus using Atoms is a binary scheme as follows. Every node is assigned two unique Atoms drawn from the pool of Atoms, $(n, w) - C$. *One Atom in the pair represents '1' and the other represents '0'.* This binary or 2-ary scheme limits the maximum number of nodes on the bus to half of the number of codes in the pool of codes.

Let Φ_e denotes the number of the codes with maximum error rate of e. In general, in an *M*-ary communication system, the maximum number of the nodes with the same communication error rate is limited to Φ_e / M .

Reducing the possible number of the nodes on the bus is not the only drawback of an *M*-ary communication scheme for a Mesh-Bus setting. A more important factor is the direct relation of M with the complexity required for detection at every node. In an *M*-ary scheme, every node needs to have M active and concurrent processes (e.g., correlation processes) for decoding content of the bus. Such added complexity may not be important in software simulations, but has dramatic negative impact on the performance low complexity hardware, which are the main target platforms for such protocols.

These factors persuaded the authors to use the least *M*-ary system, which is an unary system ($M = 1$). One can view unary messages as communication Atoms. In unary communication systems just one active detection process is needed. While unary systems are the most hardware efficient, in order to use a unary system, one needs an additional level of coding. Therefore, the unary communication scheme in Wired-CDMA systems are essentially a double layer coded system, where the CDMA code, as the channel code, is the first layer, and the coding for communication data over the coded channel, is the second layer of it.

One possible coding scheme for the data coding layer could be through time distancing of the communication of channel codes. In other words, we can use q time units, as the time interval between transmissions of Atom for every channel in order to communicate k bits of binary data. Assuming every Atom has n chips and the end-to-end time duration of double layer coded k bits of data is lq units.

$$lq = q + 2n$$

A simple way to think about this type of information coding is, for example, to transmit the character ‘A’, q is linear transformation of the ASCII value of ‘A’ $(0x65)$,

where the multiplicative factor of such a transformation is defined as resolution, r , and r_0 is a constant offset. Figure 3.9 shows such a scheme.

$$q = r\,[information] + r_0$$

Since l_q is not constant, its average value will be used in our analysis as the length of general frame of such a coding scheme. Assuming that all the numbers of a k bits data are equi-probable (the values $0, 1, 2, \ldots . 2^k - 1$ are i.i.d.) the average value of a k bits data is:

$$Value_{Ave} = \frac{1}{2^b} \sum_{i=0}^{2^b-1} i = \frac{1}{2^b} \frac{(2^b - 1)2^b}{2} = 2^{b-1} - \frac{1}{2}$$

Therefore, the average of q is:

$$\bar{q} = r\,(2^{b-1} - \frac{1}{2}) + r_0$$

The following equation shows the average length of a unary code message of k bits of information based on the average-value of the q.

$$l_{q-Ave}(k,n,r,r_0) = r_0 + r\,(2^{b-1} - \frac{1}{2}) + 2n$$

Also, in an *M*-ary system, in order to transmit k bits of data we can represent it with k/log_2^M symbols. Considering the length of n chips for every symbol, the length of the package (i.e., number of the chips,) is:

$$l_m(m,k,n) = k\,n\,/log_2^M$$

Consider, example, a setting of $(1000, 100) - C$, with chip rate of 10^4. For a binary scheme to communicate 8 bits, it takes $800\,ms$ to transmit the information. By

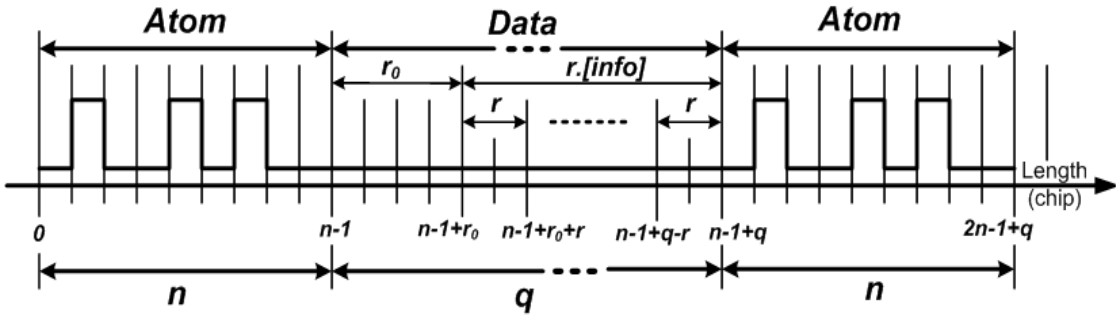


Figure 31 The time-distanced unary coding scheme.

contrast, in a unary system with a resolution factor $r = 10$ (exactly one of magnitude greater than the chip duration of $100\,\mu s$), and $r_0 = 0$ it takes $327.5\,ms$ on average to transmit the information.

In this case, using a double layered unary scheme provides the system with almost 4 dB (more than twice) average coding gain. This is, quite beneficial especially when one considers the binary scheme is twice as complex to implement in hardware. Generally, the coding gain of time-distanced, double layered unary coding over an *m*-ary coding system is given by following equation. Clearly, number of the bits per package, b, has crucial role in coding gain of such a unary coding scheme.

$$g_{c-Ave} = 10\log\left(\frac{l_m(m,k,n)}{l_{u-Ave}(k,n,r,r_0)}\right)$$

Figure 32 shows the length of a k bits package both in a binary (M-ary, where $M = 2$) system and the unary coding system, where fixed parameters are $n = 1000$, $r = 10$ and $r_0 = 0$. It also shows the coding gain of time-distanced, double

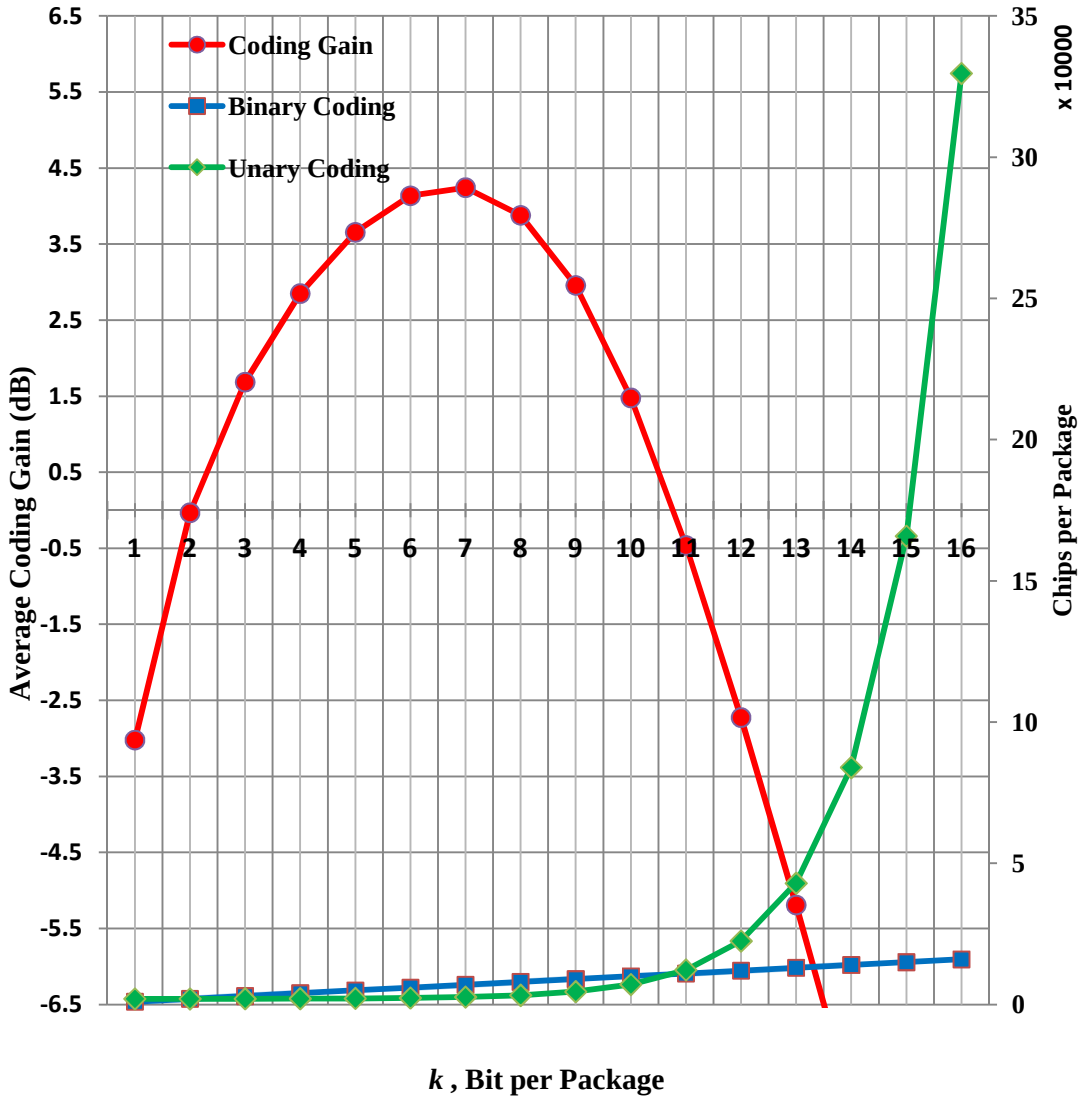


Figure 32 The message length of binary and unary coding, and the average coding gain.

layered unary coding over the binary system, versus k. The figure shows that the time distanced double layered unary coding is capable of providing more than $4\,dB$ gain when the bits per package reaches to 7, and starts decreasing after that. Figures 33 and 34 show the coding gain versus k for different values of n and r.

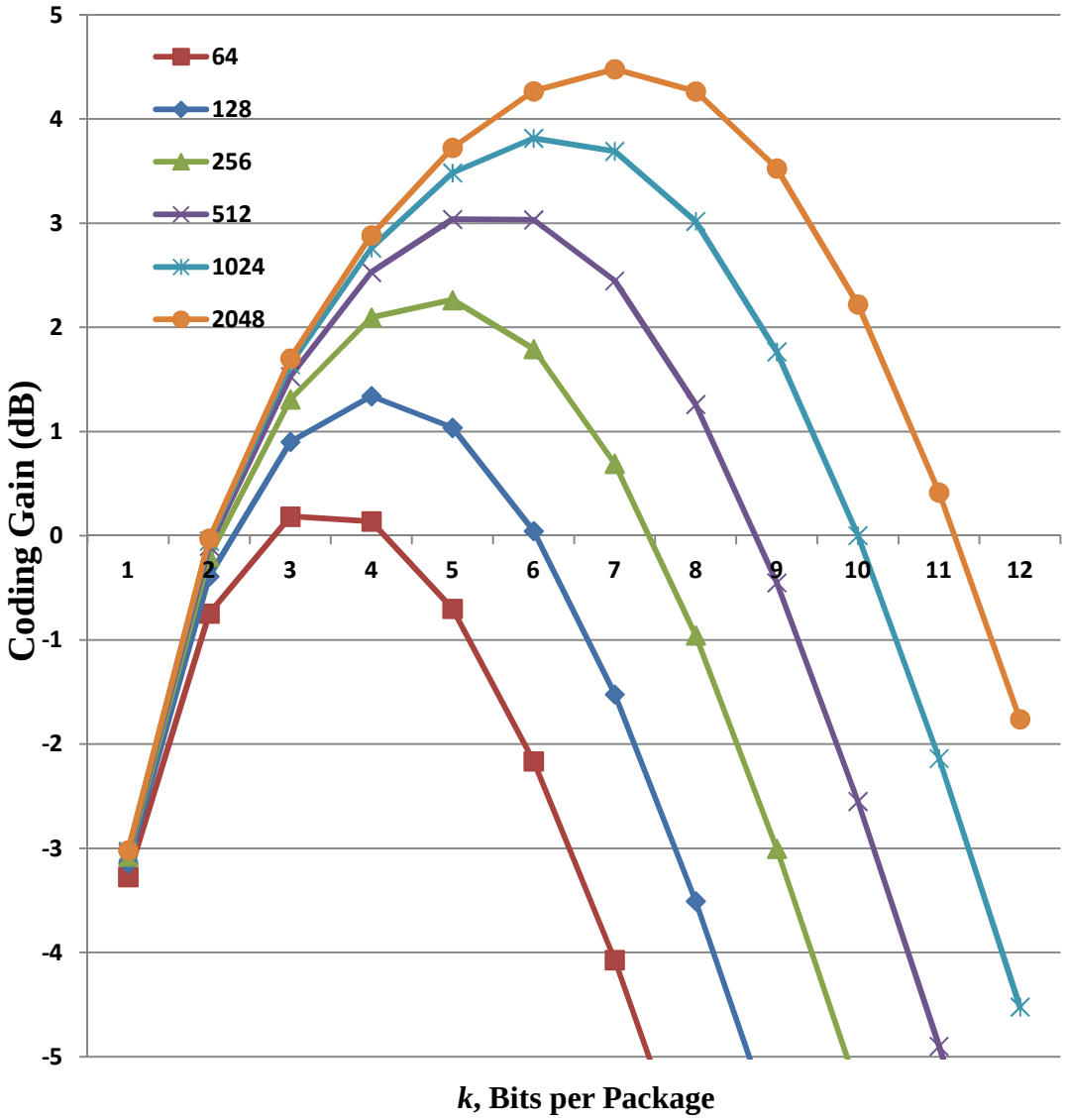


Figure 33 The coding gain of time-distanced unary coding versus k, for different Atom length, where $r = 16$.

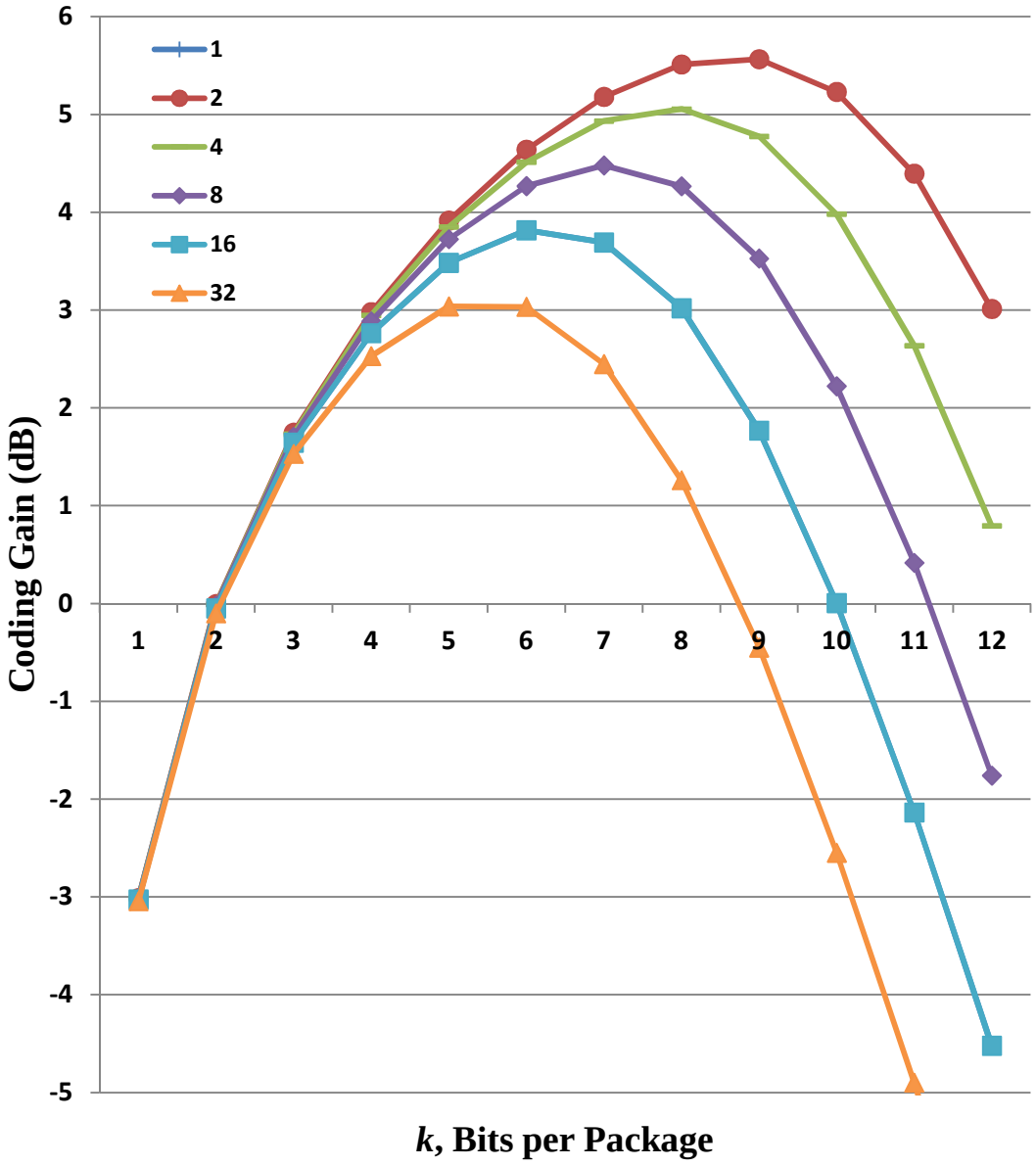


Figure 34 The average coding gain of unary scheme versus k for different values of r, where $n = 1024$.

Low Weight Atoms

The low state of the bus masks all the possible high states of all other concurrent transmissions. After increasing the number of the concurrent transmission above a certain number, the bus gets saturated (i.e., remains low for most of the time) and no meaningful communication can occur. This motivates the usage of the codes with lower ratio of the one to zeros.

We can define the code density as as follows.

As the simulation results (these simulations are based on the Wired-CDMA, see next chapter) of Figure 35 suggests, the larger the for a code, the higher the chance of

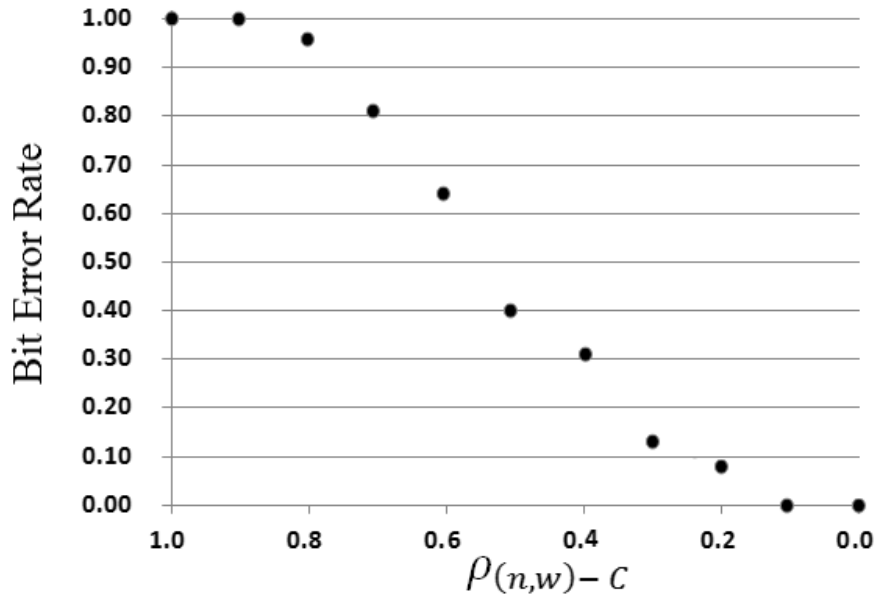


Figure 35 Code density versus error rate for 8 concurrent transmission on the bus.

saturation of the channel is. Clearly, for the density of 0.5, the error rate for 8 concurrent communications is 40%; while, reduction of ρ to 0.1 reduces it to zero for the same number of simultaneous communications.

CHAPTER 4

THE WIRED-CDMA SCHEME

Introduction

In this chapter we explore the CDMA scheme as a physical layer protocol for communication over the Mesh-Buses. In the traditional wireless or wired CDMA communication systems, the multiple access scheme is developed based on the linearity of physical layer. Since the Mesh-Bus is an asymmetric binary channel and a nonlinear communication medium, traditional CDMA techniques cannot be used directly.

In following sections of this chapter, some fundamental concepts such as Spread Spectrum, and the choice of coding schemes of the traditional CDMA are explored. Then, the nonlinear equivalent of such a communication schemes is introduced for Wired-CDMA.

Traditional CDMA

Traditional CDMA with the Spread Spectrum techniques are usually used in wireless communication systems. There are several Spread Spectrum techniques. In this study we mainly are interested in Direct Sequence Spread Spectrum [33], or in short, DSSS.

DSSS, as the name suggests, spread the frequency spectrum of the original message DSSS can provide higher signal-to-noise ratio (SNR) compared to non-spread signals. DSSS can also facilitate multiple accesses to the communication channel, where group of transceivers share the same frequency band, and simultaneously communicate using CDMA.

Figure 4.1 shows the DSSS spreading scheme. A signal is multiplied by a pseudo noise (PN) [40]sequence, as the spreading sequence. The PN sequence is a spreading code that broadens or spreads the spectrum of the original signal. The broadened or spread signal is then transmitted.

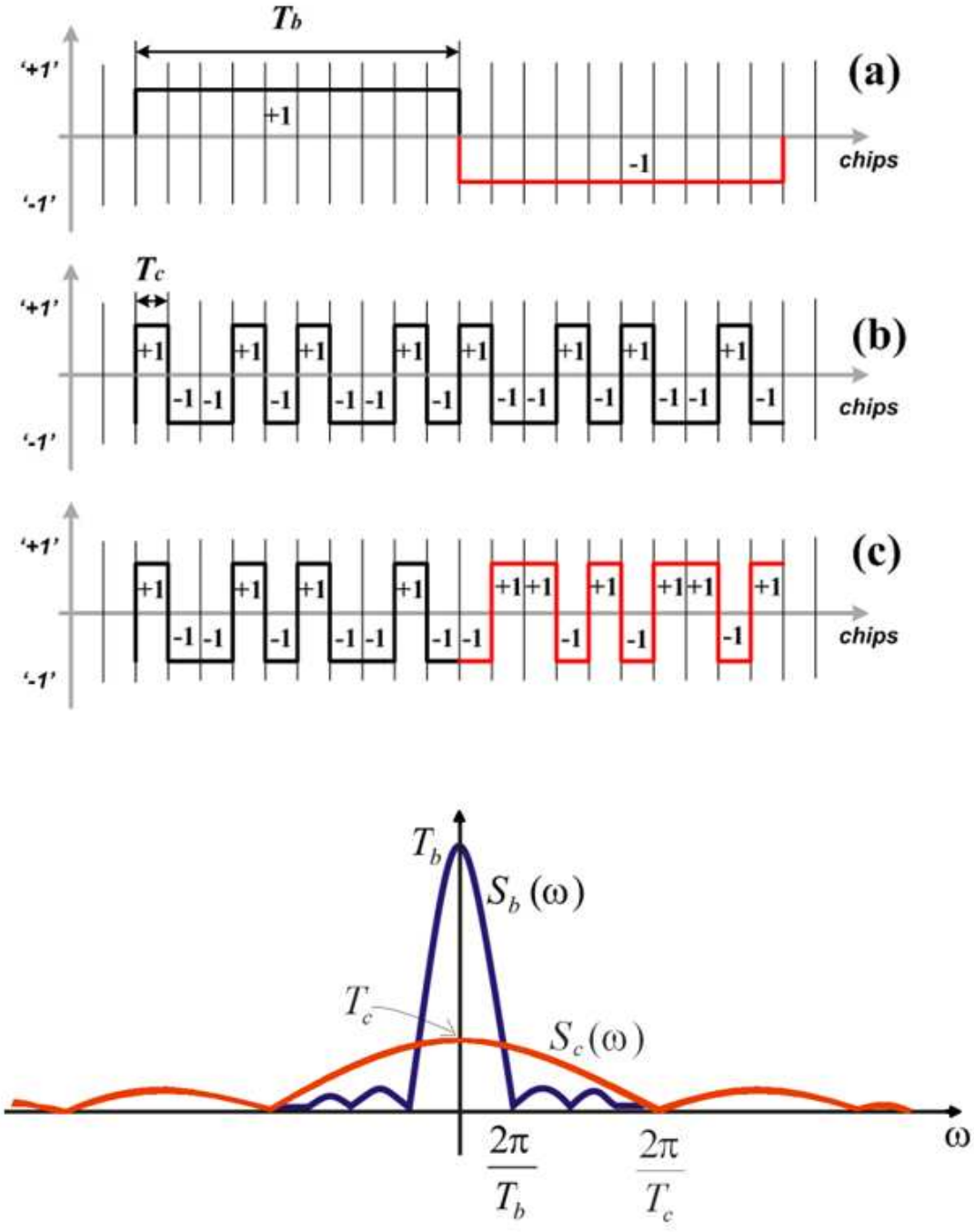


Figure 36 The process of spreading of information signal (a) by the spreading sequence (b). The signal (c) shows the spread data signal. At the bottom, the frequency spectrum of original information signal, $S_b(\omega)$, and the spreading sequence, $S_c(\omega)$ are shown.

As the figure shows, the spectrum of the PN sequence is much broader than the original message, while its spectral density is lower.

The processing gain is one of the most important parameters in a DSSS communication system, and is defined as follow, where, T_b is the bit rate of the information sequence, and T_c is the chip rate of the spreading sequence.

$$G_P = \frac{T_b}{T_c}$$

As the following formula shows, the signal to noise ratio, SNR, in a DSSS scheme has a direct linear relation with the processing gain [35]. The higher the processing gain, better SNR will be. In this formula, the P_S and P_I are the power of signal, and the power of interference (i.e., noise) respectively.

$$\left(\frac{S}{N}\right)_{DSSS} = 2\, G_p\, \frac{P_S}{P_I}$$

The multiple access aspect of DSSS schemes comes to play when one chooses the PN sequences in a way that provide the ability of differentiating the different spread signals from each other.

The randomness is the most important feature of a CDMA code. However, since we are not able to use a complete random signal, such as white noise, as the codes, the PN sequences are the next best option to provide us with the closet possible feature to the random sequences. The following two properties are the most important features of such PN codes:

1. Having an impulse-like autocorrelation function. That is, the code has its maximum (peak) value of autocorrelation at zero delay, and a very low value at the other delay values (off-peak). In case of complete random codes, such a minimum value would be zero.

2. Having very low value of cross correlation between any two different codes. In case of complete random codes, such a low value would be minimized to zero.

The first feature takes care of the differentiating a sequence from its own delayed version, therefore the transmitters and receiver are able to synchronize through the transmission of such codes. The second feature is similar to the definition of orthogonality; therefore, it the maximum value of such cross correlation is low enough, the codes could be used in a multiple access scheme. These two properties together provide us a proper code to be used in CDMA communication systems.

CDMA Codes

There are different coding scheme for the traditional CDMA communication systems. We will briefly examine them as follows.

A. Walsh Codes

In the simplest case, assume that there is set of codes or sequences where the codes are mutually orthogonal to each other (i.e. have zero cross correlation.) Such an orthogonality of the codes is capable of providing a multiple access scheme. An example of such a group of codes is the *Walsh* [41]codes.

For example, the following two codewords belong to a Walsh code of length eight. Their inner-product is zero, so that the codes are orthogonal to each other.

$$c_1: (+1, +1, +1, +1, +1, +1, +1, +1)$$

$$c_2: (+1, +1, +1, +1, -1, -1, -1, -1)$$

$$< c_1, c_2 > =$$

$$(+1 * +1) + (+1 * +1) + (+1 * +1) + (+1 * +1) + (+1 * -1)$$

$$+ (+1 * -1) + (+1 * -1) + (+1 * -1) =$$

$$+1 + 1 + 1 + 1 - 1 - 1 - 1 - 1 = 0$$

In general, to form a Walsh code with different length, we need to generate a Hadamard matrix of proper size. Every row of a Hadamard matrix is a Walsh codeword, and the following recursive formula shows how to generate the Hadamard matrices of different sizes, where every row of H_n is a Walsh codeword of length 2^n. Table 2 shows the families of Walsh codes with different length.

$$H_1 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & \overline{H_{n-1}} \end{bmatrix}$$

Considering the peak and off-peak values of the Walsh codes, in general, they do not have good autocorrelation [42]. Therefore, they are useful only when the communications are synchronized. In the case of codewords, it means the first element of

Table 2 The codewords of the Walsh codes of length 2, 4, 8, and 16.

n, where the Walsh code has length of 2^n			
1	2	3	4
(+1, +1)	(+1,+1+ 1,+1)	(+1,+1,+1,+1, +1,+1,+1,+1)	(+1,+1,+1,+1,+1,+1,+1,+1,+1,+1,+1,+1,+1,+1,+1,+1)
			(+1,-1,+1,-1, +1,-1,+1,-1, +1,-1,+1,-1, +1,-1,+1,-1)
		(+1,-1,+1,-1, +1,-1,+1,-1)	(+1,+1,-1,-1, +1,+1,-1,-1, +1,+1,-1,-1, +1,+1,-1,-1)
			(+1,-1,-1,+1,+1,-1,-1,+1,+1,-1,-1,+1,+1,-1,-1,+1)
	(+1,-1, +1,-1)	(+1,+1,-1,-1, +1,+1,-1,-1)	(+1,+1,+1,+1,-1,-1,-1,-1,+1,+1,+1,+1,-1,-1,-1,-1)
			(+1,-1,+1,-1,-1,+1,-1,+1,-1,-1,+1,-1,+1,-1,-1,+1)
		(+1,-1,-1,+1, +1,-1,-1,+1)	(+1,+1,-1,-1,-1,-1,+1,+1,+1,+1,-1,-1,-1,-1,+1,+1)
			(+1,-1,-1,+1,-1,+1,+1,-1,+1,-1,-1,+1,-1,+1,+1,-1)
(+1, -1)	(+1,+1, -1,-1)	(+1,+1,+1,+1, -1,-1,-1,-1)	(+1,+1,+1,+1,+1,+1,+1,+1,-1,-1,-1,-1,-1,-1,-1,-1)
			(+1,-1,+1,-1,+1,-1,+1,-1,-1,+1,-1,+1,-1,+1,-1,+1)
		(+1,-1,+1,-1, -1,+1,-1,+1)	(+1,+1,-1,-1,+1,+1,-1,-1,-1,-1,+1,+1,-1,-1,+1,+1)
			(+1,-1,-1,+1,+1,-1,-1,+1,-1,+1,+1,-1,-1,+1,+1,-1)
	(+1,-1, -1,+1)	(+1,+1,-1,-1, -1,-1,+1,+1)	(+1,+1,+1,+1,-1,-1,-1,-1,-1,-1,-1,-1,+1,+1,+1,+1)
			(+1,-1,+1,-1,-1,+1,-1,+1,-1,+1,-1,+1,+1,-1,+1,-1)
		(+1,-1,-1,+1, -1,+1,+1,-1)	(+1,+1,-1,-1,-1,-1,+1,+1,-1,-1,+1,+1,+1,+1,-1,-1)
			(+1,-1,-1,+1,-1,+1,+1,-1,-1,+1,+1,-1,+1,-1,-1,+1)

the first code is multiplied by the first element of the second code, the second element by the second one and so on. In real world communication systems, synchronization is not a realistic assumption. Other codes are such as *m*-sequence codes or Gold Codes are used. We will explain both of them shortly.

B. *m*-sequences

One of the most interesting group codes is the *m-sequences* [35]or the *maximal length* sequences. An m-sequence is a periodic bipolar binary string of length $2^m - 1$ and is the output of any one cell (flip-flop) of a Linear Feedback Shift Register (LFSR) [35]with a specific weight set. Figure 37 shows a sample LFSR of length m. In an LFSR of length m, the input signal to the first cell, a_0 in the Figure, is a weighted modulo−2

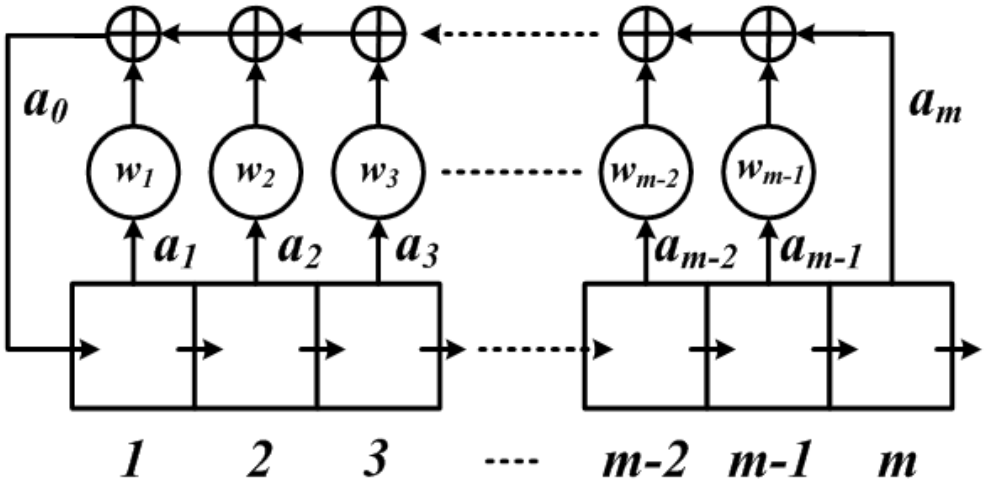


Figure 37 A Linear Feedback Shift Register.

summation (i.e., XOR) of the other m cell as follows, where the a_i and $w_i \in \{0, 1\}$.

$$a_0 = a_m \oplus \sum_{i=1}^{m-1} \oplus (w_i \cdot a_i)$$

In an LFSR of length m, if the weight set is chosen according to a primitive polynomial of the degree m, the output of every cell of that LFSR is an m-sequence.

Figure 4.5 shows the LFSR with the primitive polynomial of $1 + X + X^3$ and the resulting *m*-sequence. For example, the output of the last cell s_3, is an *m*-sequence with length $2^3 - 1 = 7$.

The most important feature of the *m*-sequences is their autocorrelation as shown in the following formula. Such an autocorrelation function is exactly compliant with the feature one, the impulse-like function. Figure 38 shows the previous m-sequence, its autocorrelation function and its spectral density.

$$R_c(\tau) = \frac{1}{T_b} \int_{-\frac{T_b}{2}}^{\frac{T_b}{2}} c(t)c(t - \tau)\, dt$$

$$R_c(\tau) = \begin{cases} 1 - \dfrac{N+1}{NT_c}|\tau| & |\tau| \le T_c \\ -\dfrac{1}{N} & \text{Rest of the Period} \end{cases}$$

In a discrete form, as Figure 38 shows, the autocorrelation is as follows:

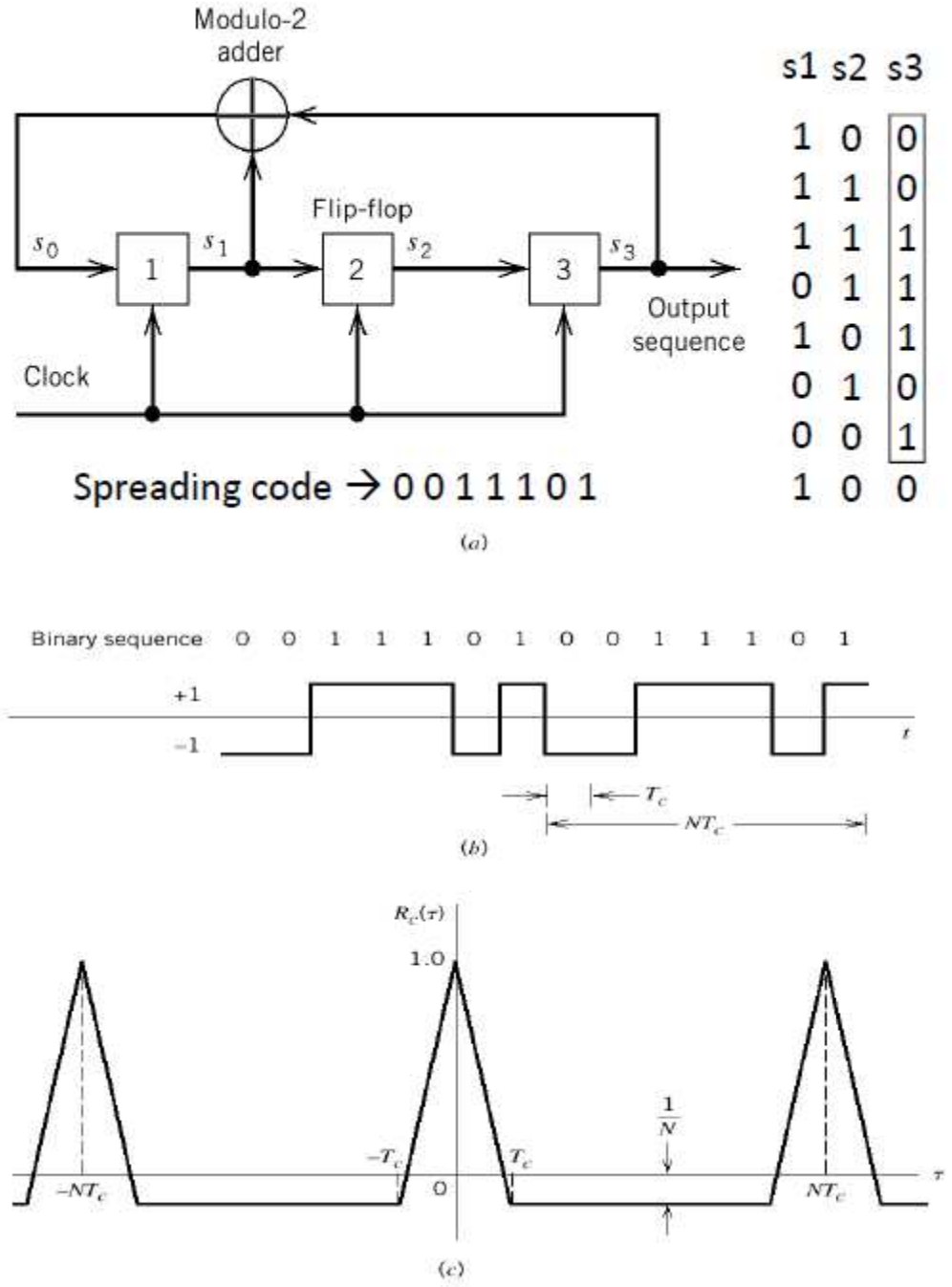


Figure 38 m-Sequences and their properties: (b) The m-Sequence of ‘0011101, generated by (a) the LFSR of the primitive polynomial of . (c) The normalized autocorrelation of the m-sequence versus .

$$R_c(\tau) = \sum_{i=0}^{N-1} c_i \, c_{i\oplus\tau} = \begin{cases} N & (\tau \bmod N) = 0 \\ -1 & (\tau \bmod N) \neq 0 \end{cases}$$

C. Gold Codes

As we can see, m-sequences are very good for providing the impulse-like autocorrelation functions. Unfortunately, the cross-correlation of m-sequences is large, also, there are limited number of unique m-sequences available for a fixed length.

However, starting with the basic *m*-sequences, one can generate so-called *Gold Codes* [35] that have both excellent auto- and cross correlation properties. Gold codes are binary summations of two different m-sequences with minimum cross correlation, as Figure 39 shows. In this figure, two LFSR with primitive polynomials of $1 + X^2 + X^5$ and $1 + X^2 + X^3 + X^4 + X^5$ are producing m-sequences of a_n and b_n.

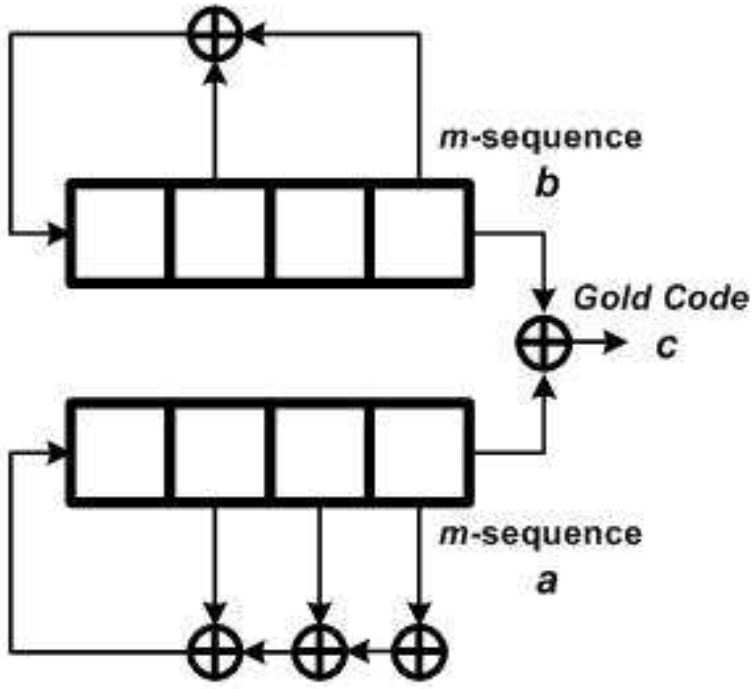


Figure 39 The sequence c_n is a Gold code generated by the two m-sequences of a_n and b_n.

The Gold Code c_n, as it is shown, is their shifted modulo-2 addition. Since the length of the m-sequences are $n = 2^m - 1$, there are n Gold Codes as the result of such modulo-2 additions. Adding the two original m-sequences of a_n and b_n, there are $n + 2$ Gold Code for every two properly chosen m-sequences.

As has shown, the Gold Codes, not only have the desirable property of impulse-like autocorrelation, but also have very low cross correlation compared to the *m*-sequences. Having these two properties at the same time, makes the Gold code one of the best options to be used in CDMA communication systems.

Near-Far Problem

The wireless communication system experience power loss during the signal transmission. In such systems, the path loss has exponential relation to the length of the communication channel (i.e., the distance between the transmitter and the receiver via line of sight, or the reflected/diffracted path that the wave travels to reach the destination.)Therefore, the power of the transmission from a near transmitter can easily overwhelm the power of the transmitted signal of the farther ones, and makes it impossible for the receiver to detect such far transmissions. This phenomena is called the near-far problem [18, 19].

Also there are some solutions to the near-far problem, such as dynamic power control [43], this problem makes the application of CDMA difficult.

Wired-CDMA

All the coding schemes we have explored thus far are digital binary schemes, but with a bipolar alphabet (e.g., '-1' and '+1'). Further, all of them are designed for a linear channel. That is, the instantaneous values of a code could get added to another's one in a linear fashion, and the result could be something other than the original alphabet. In fact, based on the number of the participants sharing the medium, the result could be some value belongs to $\mathbb{Z}$ (i.e., $\{\ldots -3, -2, -1, 0, 1, 2, 3 \ \ldots\}$). On the other hand, in Wired-

CDMA, as a physical layer of the Mesh-Bus communication buses, the communication medium is a NOR bus. The NOR operation and bus are nonlinear. Also, since the bus is a binary medium, only two valid states are allowed on the bus, logic ‘0’ and logic ‘1’ (it is a unipolar system).,The result of the mixing of the codes together belongs to the same alphabet.

These differences make all the standard coding schemes of the CDMA communication incompatible with Wired-CDMA. However, the two properties of the codes, the impulse-like autocorrelation function, and low cross correlation, are still valid requirements for any coding scheme that will be used in Wired-CDMA.

Based on the similarities between the fiber optic communication, and the wired-NOR we have decided to use the coding scheme used in optical CDMA in this study, and develop a new set of codes based on the research that has been done in that area.

There are two main similarities between fiber optic communications and NOR buses. First, in optical communications, the presence and absence of light (assuming single wavelength) represents the bits of information. Therefore it is a binary channel. Second, fiber optic communication neither has an asymmetric channel interference model, exactly the same as NOR bus.

There exist families of so-called Optical Orthogonal Codes (OOC) codes that were specifically developed for unipolar fiber optic communications. For the rest of this study we use OOCs for the Wired-CDMA

Optical Orthogonal Codes

Optical Orthogonal Codes or OOC, are a family of codes that were designed for unipolar fiber optic CDMA communication systems. A codeword of an OOC code is binary string, and uses ‘0’ and ‘1’ characters as its alphabet. In the most general case an OOC is denoted as the following 4-tuple of $(n, w, \lambda_a, \lambda_c)-OOC$, where n is the length of every codeword in the code, w is the Hamming weight of the code (i.e., it is the number

of the ‘1’s in a codeword). The two properties that we explored in previous chapter come to play in terms of λ_a and λ_c .These are the circular autocorrelation and circular cross correlation respectively, defined as follows.

Assuming $c^i = (\, c_0^i,\, c_1^i, \ldots\, c_{n-1}^i)$ and $c^j = (\, c_0^j,\, c_1^j, \ldots\, c_{n-1}^j)$, where $c_l^k \in \{'0', '1'\}$, both are two codeword of$(n, w, \lambda_a, \lambda_c)-OOC$, the circular autocorrelation function of a codeword is:

For every codeword $c^i \in (n, w, \lambda_a, \lambda_c) - OOC$

$$ACF\ (\tau) = \sum_{\substack{l=0 \\ \tau \neq 0}}^{n-1} c_l^i\ .c_{l\oplus\tau}^i \leq \lambda_a$$

Since this formula is a circular function, $l \oplus \tau = (\, l + \tau\,) \overset{m}{=} n$, and for the special case of $\tau = 0$, we will have

$$ACF\ (0) = \sum_{l=0}^{n-1} c_l^i\ = w$$

For circular cross correlation, the definition is as follows:

$$CCF\ (\tau) = \sum_{\substack{l=0 \\ i \neq j}}^{n-1} c_l^i\ .c_{l\oplus\tau}^j \leq \lambda_c$$

Clearly, the autocorrelation property of the OOCs takes care of the task of differentiation of a codeword from the delayed version of the same codeword. The cross correlation property, on the other hand, is responsible of providing some measure of

differentiation between every two different codewords of a codebook. Figure 40 shows the autocorrelation and cross correlation functions for two codewords of a $(123, 5, 1, 1)-OOC$.

The number of the codewords of a $(n, w, \lambda_a, \lambda_c)-OOC$code, namely cardinality of the OOC code, usually denoted as $|C|$ and the maximum number possible of the codewords or maximum cardinality of an OOC code usually is denoted by $\Phi(n, w, \lambda_a, \lambda_c)$. In case number of the found codeword equals $\Phi(n, w, \lambda_a, \lambda_c)$ we call that code an optimal code, and if the cardinality of a code is less that $\Phi(n, w, \lambda_a, \lambda_c)$, we call it a suboptimal code.

If one required that $\lambda_a = \lambda_c = \lambda$, the OOC is called a symmetric OOC. In this case one can use the more compact notation $(n, w, \lambda) - OOC$. In general, there does not exist an analytical formula for the cardinality of an OOC. However, one can determine the upper and lower bounds as follows

$$\phi(n, w, \lambda_a, \lambda_c) \leq \frac{(n-1).(n-2)\ldots(n-\lambda)}{w.(w-1)\ldots(w-\lambda)}$$

$$\phi(n, w, \lambda_a, \lambda_c) \geq \frac{\binom{n}{w} - \frac{n-1}{2}\binom{w}{\lambda_a+1}\binom{n}{w-\lambda_a-1}}{n.\sum_{i=\lambda_c+1}^{\min\{n-w,w\}}\binom{n-w}{w-i}\binom{w}{i}}$$

In our study, we have decided to start exploring OOCs in order to be the simplest case of the Atoms in Wired-CDMA. Therefore, to start, we focus on a simple family of $(n, 3, 1) - OOC$ and build on that as we proceed.

As shows, for the family of the optimal $(n, w, 1) - OOC$ we have

$$\phi(n, w, 1) \leq \left\lfloor \frac{n-1}{w(w-1)} \right\rfloor$$

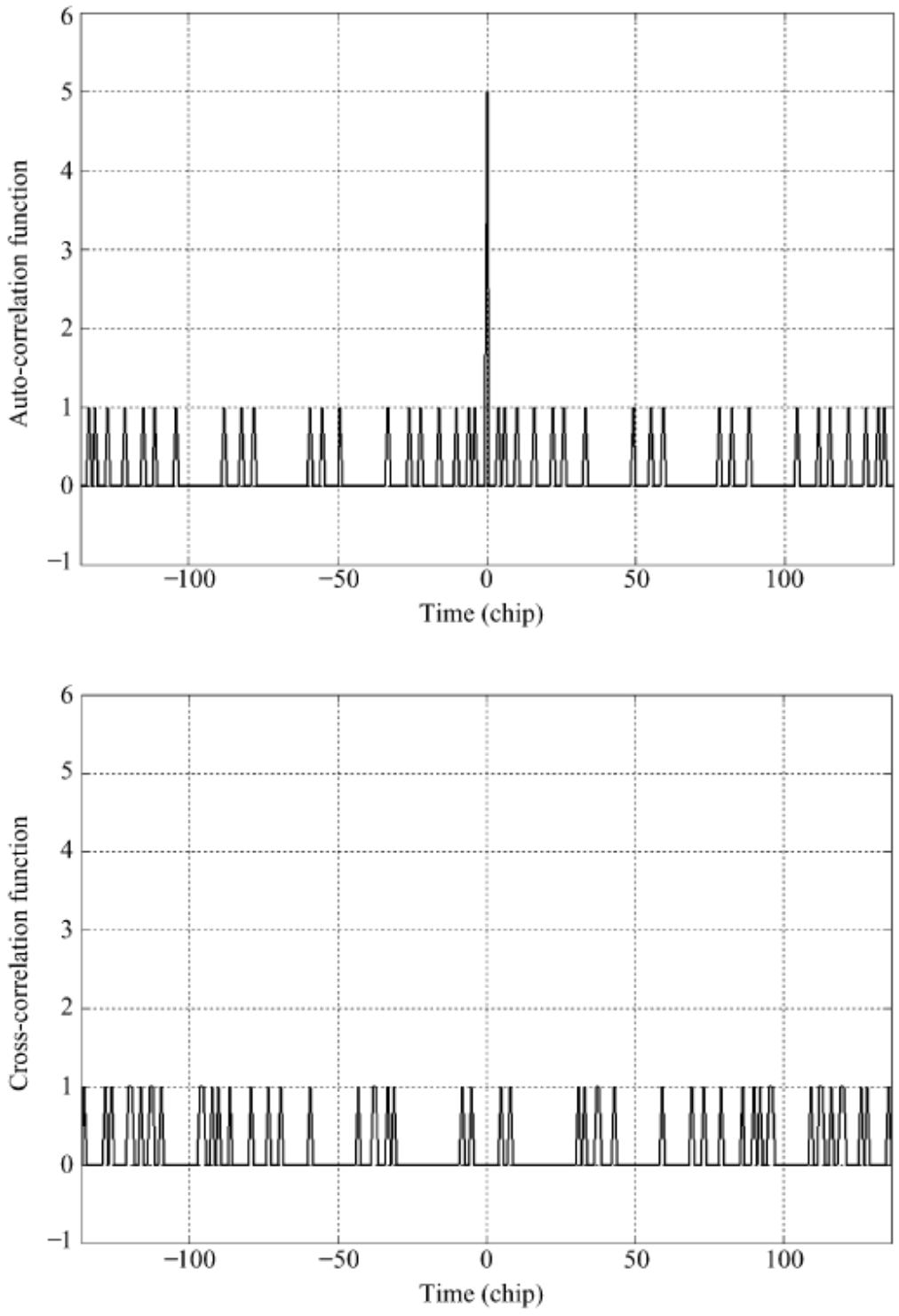


Figure 40 Correlation properties of $(123, 5, 1, 1)-OOC$: (Top) the autocorrelation of a codeword.(Bottom) The cross correlation functions between the codewords [13].

Therefore, for $(n, 3, 1) - OOC$ we have:

$$\phi(n, 3, 1) \leq \left\lfloor \frac{n-1}{6} \right\rfloor$$

As an example, $(13, 3, 1) - OOC$ has the cardinality of two, and following two codewords could belong to it.

$$c^1 = (0,1,0,1,\ 0,0,0,0,0,1,000)$$

$$c^2 = (0,0,1,0,\ 0,1,1,0,0,0,000)$$

Specifying n digits for every codeword in the form of an n-tuple becomes awkward, especially for large n . The following compact notation is useful. Codeword are represented as:

$$c^1 = (1, 3, 9)$$

$$c^2 = (2, 5, 6)$$

In c_1, there are 1s in position 1, 3, and 9, and 0s in all other positions. In c_2 there are 1s in positions 2, 5, and 6, and 0s in all other positions. Figure 41 depicts these codes.

There are different approaches to generating the codewords of an OOC such as iterative construction, combinational construction, algebraic construction, and so on. In this study, we chose to use the simple way of combinational construction to achieve the codewords of $(n, 3, 1) - OOC$. The method is based on the solution to the set packing problem and as follows:

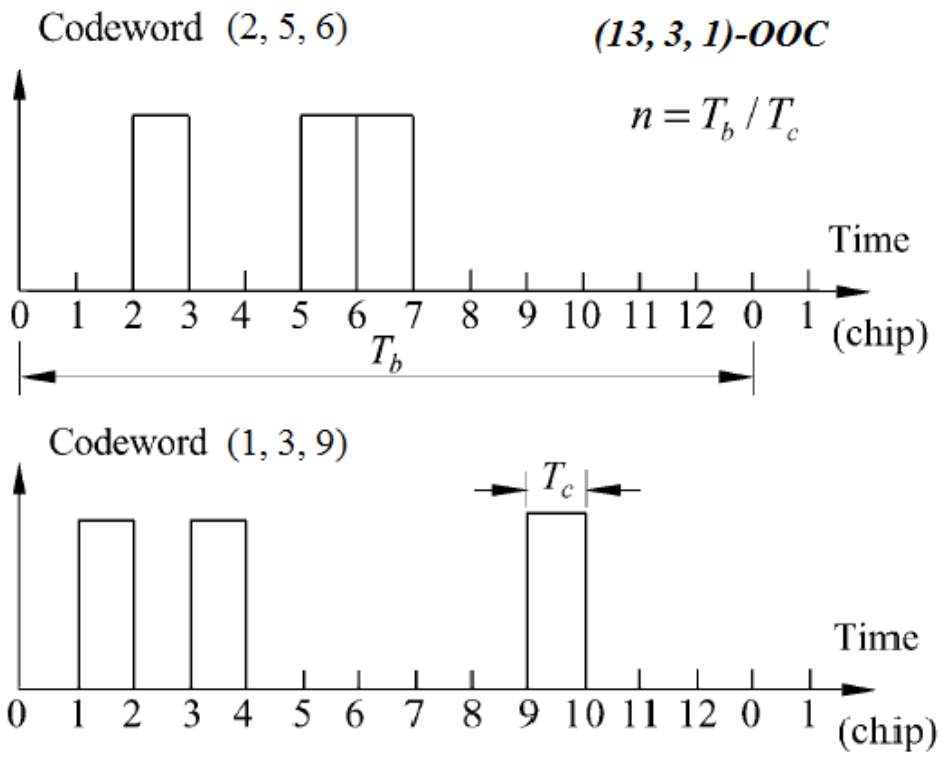


Figure 41 The two codewords of , (2,5,6) and (1, 3, 9).

Assume that:

Case one: if

Assume

The codes are:

$$(0, 6u, 10u), (0, 9u, 9u + 1), (0, 10u + 1, 12u)$$

Case two: if $l \equiv 1\ (mod\ 4)$

Assume $l = 4u + 1 \geq 9$

The codes are:

$$(0, l + i, 2l + 1 - i) for\ (1 \leq i \leq 2u)$$

$$(0, 2l + i, 3l - i) for\ (1 \leq i \leq u)$$

$$(0, 2l + u + 2 + i, 3l - u - i) for\ (1 \leq i \leq u - 2)$$

$$(0, l + 2u + 1, 2l + 2u + 1), (0, 2l + u + 1, 2l + u + 2), (0, 2l + 2u + 2, 3l)$$

Case three: if $l \equiv 2\ (mod\ 4)$

Assume $l = 4u + 2 \geq 6$

The codes are:

$$(0, l + i, 2l - i) for\ \left(1 \leq i \leq \frac{l}{2} - 1\right)$$

$$(0, 2l - 1 + i, 3l - i) for\ (1 \leq i \leq u)$$

$$(0, 2l + u + 1 + i, 3l - u - i) for\ (1 \leq i \leq u - 1)$$

$$\left(0, \frac{3}{2}l, \frac{5}{2}l\right), (0, 2l + u, 2l + u + 1), \left(0, \frac{5}{2}l + 1, 3l + 1\right)$$

Case four: if $l \equiv 3\ (mod\ 4)$

Assume $l = 4u + 3 \geq 7$

The codes are:

$$(0, l + i, 2l + 1 - i) for\ (1 \leq i \leq 2u + 1)$$

$$(0, 2l + i, 3l + 1 - i) for\ (1 \leq i \leq u + 1)$$

$$(0, 2l + u + 3 + i, 3l - u - 1 - i) fo\ (1 \leq i \leq u - 2)$$

$$(0, l + 2u + 2, 2l + 2u + 2), (0, 2l + u + 2, 3l + 3), (0, 2l + 2u + 3, 3l + 1)$$

The following are the 10 codewords of $(65, 3, 1) - OOC$, which are generated by a Java simulation engine for the purpose using in Wired-CDMA simulations. Table 3 shows the results of the calculations of their autocorrelation and cross correlation, based on the formula presented before. In the table, the diagonal elements are the autocorrelations, and all have the values of 3-1, where 3 is the autocorrelation with delay zero, and 1 is the maximum result of autocorrelation with a nonzero delay.

$$C(65, 3, 1) - OOC$$

$$N = 10, |C| = 10$$

$$Code\ (0): (0, 11, 19)$$

$$Code\ (1): (0, 12, 18)$$

$$Code\ (2): (0, 13, 17)$$

$$Code\ (3): (0, 14, 16)$$

$$Code\ (4): (0, 20, 29)$$

$$Code\ (5): (0, 21, 28)$$

$$Code\ (6): (0, 24, 27)$$

$$Code\ (7): (0, 15, 25)$$

$$Code\ (8): (0, 22, 23)$$

$$Code\ (9): (0, 26, 31)$$

Table 3 The auto and cross correlation of the codewords of *(65, 3, 1)-OOC.*

Codes	0	1	2	3	4	5	6	7	8	9
0	3-1	1	1	1	1	1	1	1	1	1
1	1	3-1	1	1	1	1	1	1	1	1
2	1	1	3-1	1	1	1	1	1	1	1
3	1	1	1	3-1	1	1	1	1	1	1
4	1	1	1	1	3-1	1	1	1	1	1
5	1	1	1	1	1	3-1	1	1	1	1
6	1	1	1	1	1	1	3-1	1	1	1
7	1	1	1	1	1	1	1	3-1	1	1
8	1	1	1	1	1	1	1	1	3-1	1
9	1	1	1	1	1	1	1	1	1	3-1

In a Wired-CDMA scheme, all the communications are assumed to be based on unary messages named Atoms. Now, we consider a simple Atom could be a codeword of an OCC, and based on such an assumption, try to simulate the Atom transmissions and detections in a multi node communication channel, such as Mesh-Bus's NOR bus.

Simulations

In order to evaluate the performance of OOC's codewords as the Atoms of Wired-CDMA we designed and developed a Java simulation engine. This simulation engine was designed as modular as possible, allowing us to investigate different aspect of the problem in a flexible manner.

The simulation engine includes an independent class for generation of OOC of different length, and is able to analysis the properties of every code as its input. This feature will allow us in the future to test and accelerate our new codes.

It also neither includes a specific class for NOR buses. Every node on the bus assumed to be an object with its own unique Atom (in Wired-CDMA case, its OOC codeword), and it is capable of transmitting the Atom at any specified time, as well as, monitoring the bus in order to detect any possible transmission toward it from other nodes.

In the first set of simulations we just focus on transmission and detection of Atoms, as unary massages, and not the any specific information content. In the simulations we assumed there are N nodes on the bus, and they all transmit their Atoms concurrently. The Atoms of the nodes are programmed to overlap with each other from 0.1 % to 100% randomly, and every experiment is repeated 10^6 to 10^8 times.

The detection rule for Wired-CDMA is the same as general detection rule. Whenever there is a '1' in the Atom, the bus must be '0'. Therefore every time there is such a situation on the bus, it could be an Atom which is transmitted by some node. But in reality there are situations in which there is not any transmitted Atom, the transmissions of other Atoms accidentally has formed a pattern similar to an Atom. In such cases, there will be detection error.

There are two types of such errors. First, when an Atom is transmitted, but as the result of other transmissions interferences, a detection candidate appeared in some time other that the time of the actual transmission. We call this type of error as *Type One* or *detection time errors*. The second and the most important type of errors are those which occur when no actual atom is transmitted, but as the result of interference, the Atom is detected. These types of the errors are *Type Two* or *false positive detections*.

In order to evaluate both Type One and Type Two errors, two sets of experiments are done. In the first set, one specific Atom is transmitted in a specific time, where other

overlapping Atoms are also being transmitted. On the other hand, one node tries to detect that specific Atom. When it is detected, the detection compared to the transmission time, and if they do not match, a Type One error is occurred. The other set of experiments is design to evaluate Type Two error. In this set of experiments, a specific Atom it tried to be detected by a node, when it actually has not been transmitted. Every time a non-transmitted Atom is successfully detected, this is a false positive detection.

Figure 42 shows the result from one such simulation. In this simulation Atoms (codewords) belong to $(1000, 3, 1) - OOC$. As it shows, higher the number of the nodes on the bus, the more errors in communicating the Atoms will be.

Data Rate Evaluation

Based on the simulation result above, we can have up to 20 nodes on the bus communicating their Atoms concurrently with at most 7% of error. Now according to the time-distanced unary coding scheme, we are able to calculate the data rate and bit error rate as follows.

As we showed in time-distanced unary coding scheme, in which for every *k* bits of data, two Atoms encapsulate some function of the numerical value of that k bits. Based on the average frame length for such double-layered coding, the bit rate is calculated as follows

$$Data\ Rate_{\ Ave} = \frac{k}{Length_{\ Ave}} \times\ Sample\ rate$$

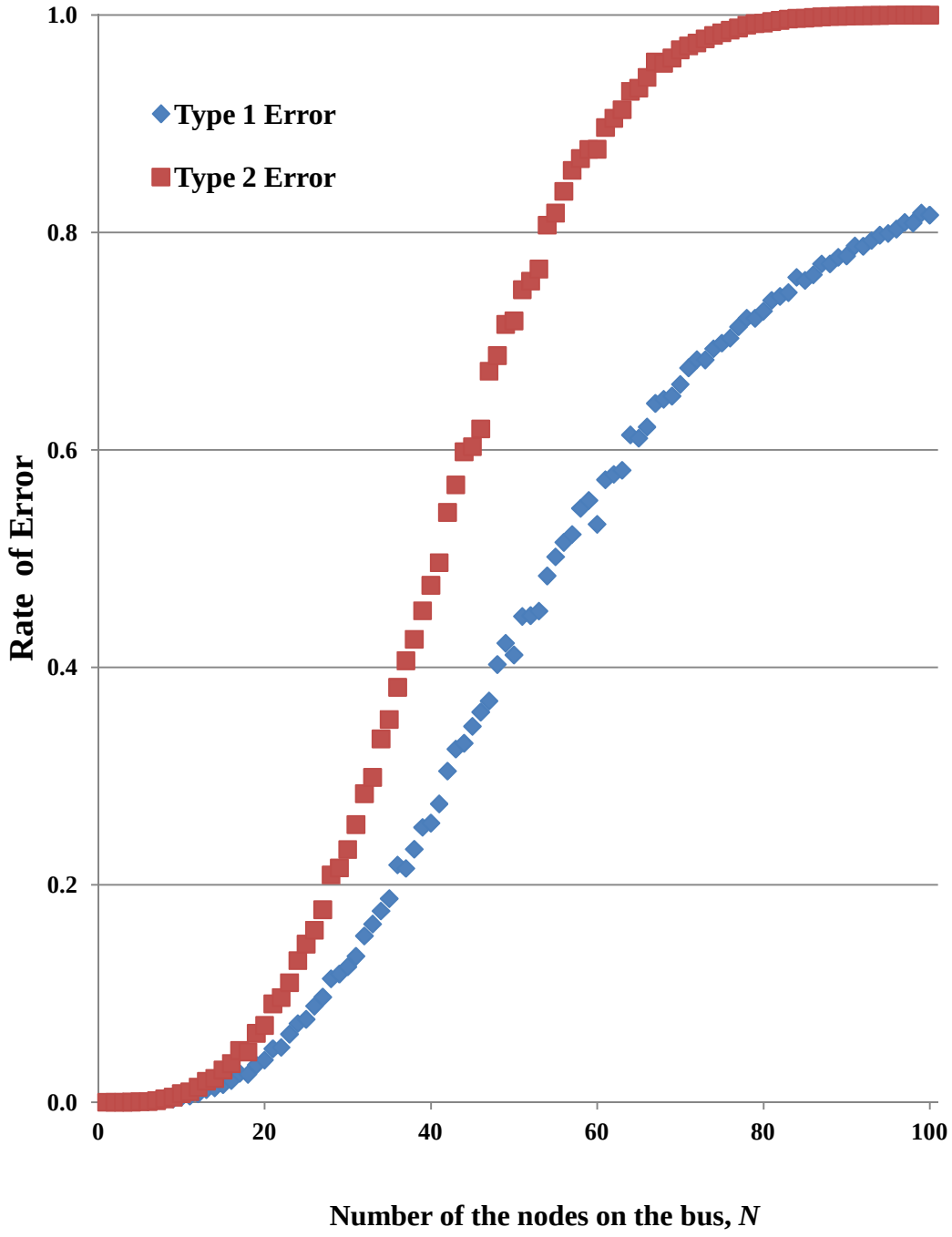


Figure 42 Simulation results for the error rate evaluation of the $(1000, 3, 1) - OOC$. In the graph, the red curve shows the Type Two (the false positive detection), and the blue curve shows the Type One error rate.

Figure 43 shows he bit rate versus k for different values of n, where $r = 16$. As the Figure suggests, the maximum data rate for $n = 1024$ is at 7 bit, in which the average bit rate $0.002609\ bit\ per\ chip$.

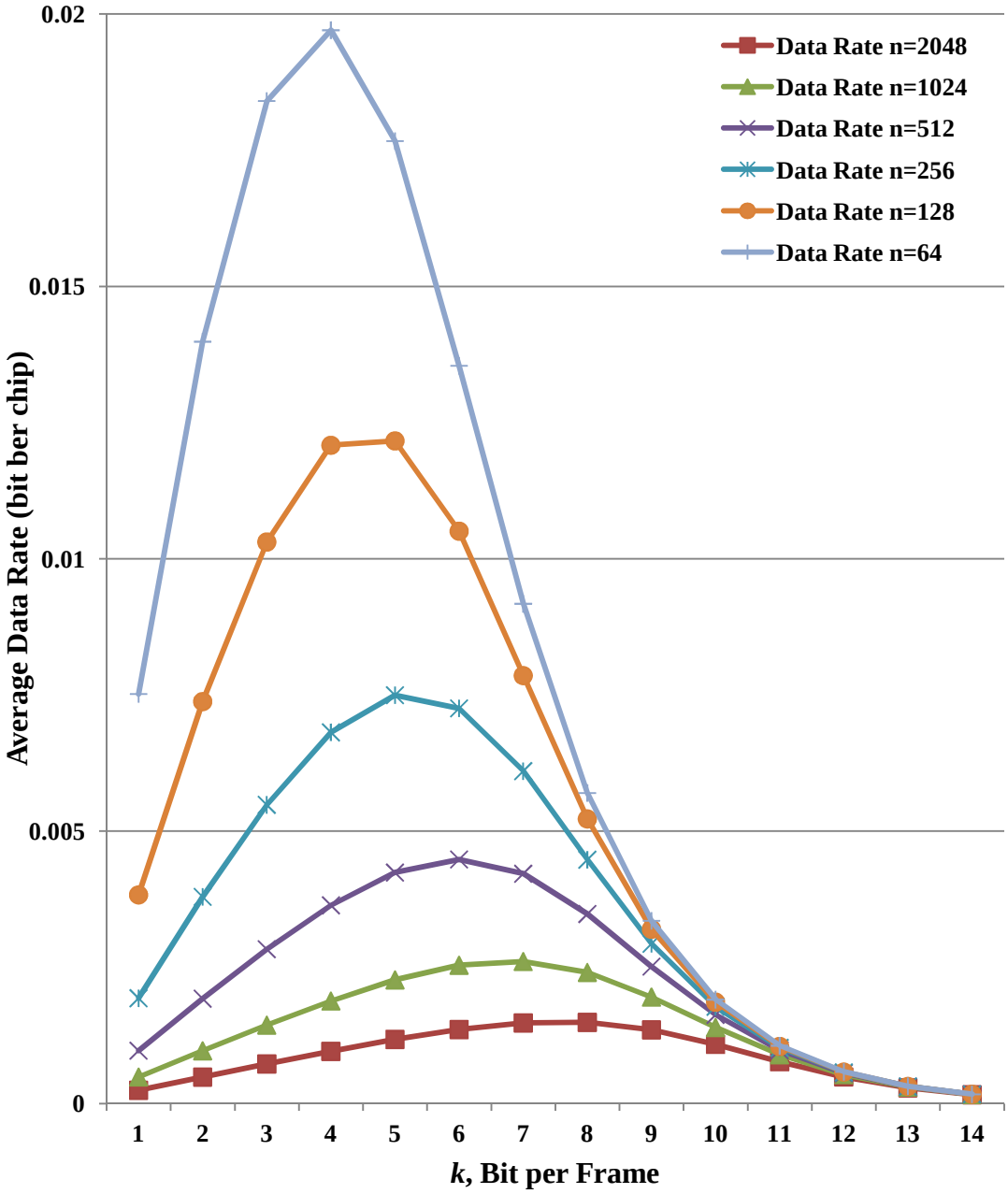


Figure 43 The data rate of the Wired-CDMA per chip, versus k, for different length of Atoms. In this graph and $r = 10$.

We assume that the sample rate of the notes are 1 Mega chip per second, therefore the time duration of every chip is 1 μs, and the data rate is 2.609 kbs

Error Correcting through Time-Distanced Information Coding

The only source of error in the Wired-CDMA is misdetection. Both types of the error, detection time error and false positive detection error, happen when a wrong detection takes place because of interference of the other Atoms. Therefore, when there are fewer Atoms on the bus, the chance of error decreases.

The other fact is that, since the transmission time of the Atoms for every node is a random process with a uniform probability distribution. Therefore, the formation of the non-transmitted Atoms that causes the wrong is also random uniformly distributed. In other words, for a fixed number of the Atoms on the bus with uniform distribution of time of transmission, the probability of the wrong detection has a uniform distribution.

In case of using linear transformation of the ASCII values for the information coding scheme ($q = r\,[\text{information}] + r_0$) the two parameters of r_0 and r play an important role in reducing the error rate in following ways

r_0, as a fix offset for all the ASCII values, guarantees a minimum distance for ever two Atoms of an information frame q. Therefore, the density of the presence of the Atoms on the bus reduces, and so the chance formation of the non-transmitted Atoms.

r, as the multiplicative factor of the transformation, distance every two Atoms by at least r chips (assume $r_0 = 0$). Therefore, if a detection happens less than r chips after the previous Atoms, it is a wrong detection. In general, if the first Atom of the frame is detected at time t_1, only the detection at the time t (as follows) could be the transmitted Atoms.

$$t = r_0 + \hat{q} \cdot r$$

Where, $\hat{q} \in \mathbb{N}$ is the transmitted ASCII value.

Therefore, considering the uniform distribution of wrong formation of non-transmitted Atoms, the number of the wrong detection could be reduced by factor of $1/r$ due to this type of coding.

As an example, Figure 4.6 shows that the error rate for a Wired-CDMA scheme with 20 nodes is less that 7%. This value could be reduced by proper choice of r to $7/r$ % through elimination of wrong detection. Such an error reduction process if called *resolution based* error reduction. Also, there is an indirect process for error reduction through reducing the chance of interference through distancing the Atoms by the proper choice of r_0.

CHAPTER 5

HARDWARE IMPLEMENTATION

Introduction

In this chapter, the hardware implementation of the Bus-Modulators is studied and various parameters which affect their operation are explored. The vision for this study, as explained in previous chapters, is to have intermediary devices, namely the Bus-Modulators, which are very simple to work with, and allow the communicating nodes to use the bus without knowing about and dealing with the bus complexities. Such Bus-Modulators already are programmed with the addresses and chip rates and other necessary parameters to be able to operate over the NOR-Bus. A known address book and an agreed upon chip rate are the only two pieces of information which the nodes are required to be aware of in order to have an operational bus. Such information, as well as some other parameters, could be hardcoded in the firmware of the Bus-Modulators, or could be configured on existing devices. Another option is to have all the information programmed in the firmware, and users select a set during a very quick setup phase prior to the use a Bus-Modulator. Figure 44 shows the Bus-Modulators.

The Bus-Modulators in this study are implemented using a general purpose microcontroller. The reason for using microcontrollers is their simplicity both in hardware and software. However, as we will see in following sections, they are not the best option for some specific implementations. There are other choices of hardware platforms for an ideal Bus-Modulator, such as CPLDs or FPGAs, which are too complex and costly for our purpose.

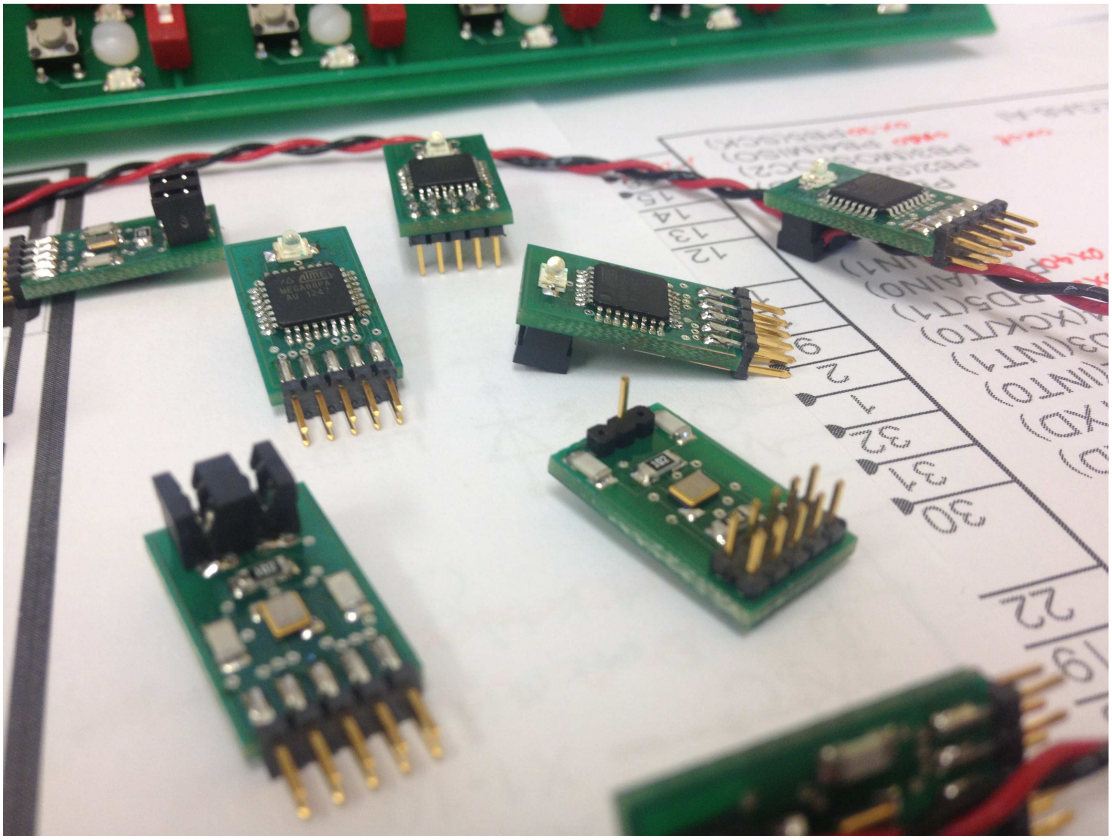


Figure 44 Top, bottom, and side view of the implemented Bus-Modulator. It has a 10-pin connector for high level interfacing and a 2-pin IDC connector for interfacing the bus.

The hardware implementation of the Bus-Modulator for this thesis is consists of two major parts: the task of design and implementation of the physical circuitry and interfaces of Bus-Modulators, which we call it hardware for short from now on, and the more important one, the program which such a hardware runs in order to the tasks of the Bus-Modulator, or the firmware. The following section describes these two parts in details. Figure 45 shows a high level schematic of the Implemented Bus-Modulator and the internal structure of its hardware and firmware

Hardware

The hardware implementation of the Bus-Modulators are based on Atmel 8-bit AVR microcontrollers family. But in order to find a proper microcontroller in the family, there are some parameters which must be considered, such as:

1- High frequency. Bus-Modulators are designed for CDMA which is a spread spectrum scheme. In the Spread Spectrum schemes, chip rate is often much higher than the data rate, so the system must be able to operate very fast in order to achieve a proper data rate. Therefore, a high frequency microcontroller helps achieving high chip rate, and accordingly. The highest frequency that the 8-bit AVR microcontrollers support is 20 MHz, and in most cases each CPU instruction takes only one CPU cycle. Therefore, the maximum throughput is 20 MIPS (million instructions per second.)

2- Serial communication. Another important parameter for choosing the right AVR microcontroller for the implementation of Bus-Modulator is the ability of communication via USART. Not all the AVR microcontrollers have this feature.

In addition to the mentioned features, having a low pin count in order to make the Bus-Modulator as small as possible, as well as large memory and large flash memory are some other characteristics which help narrow the search.

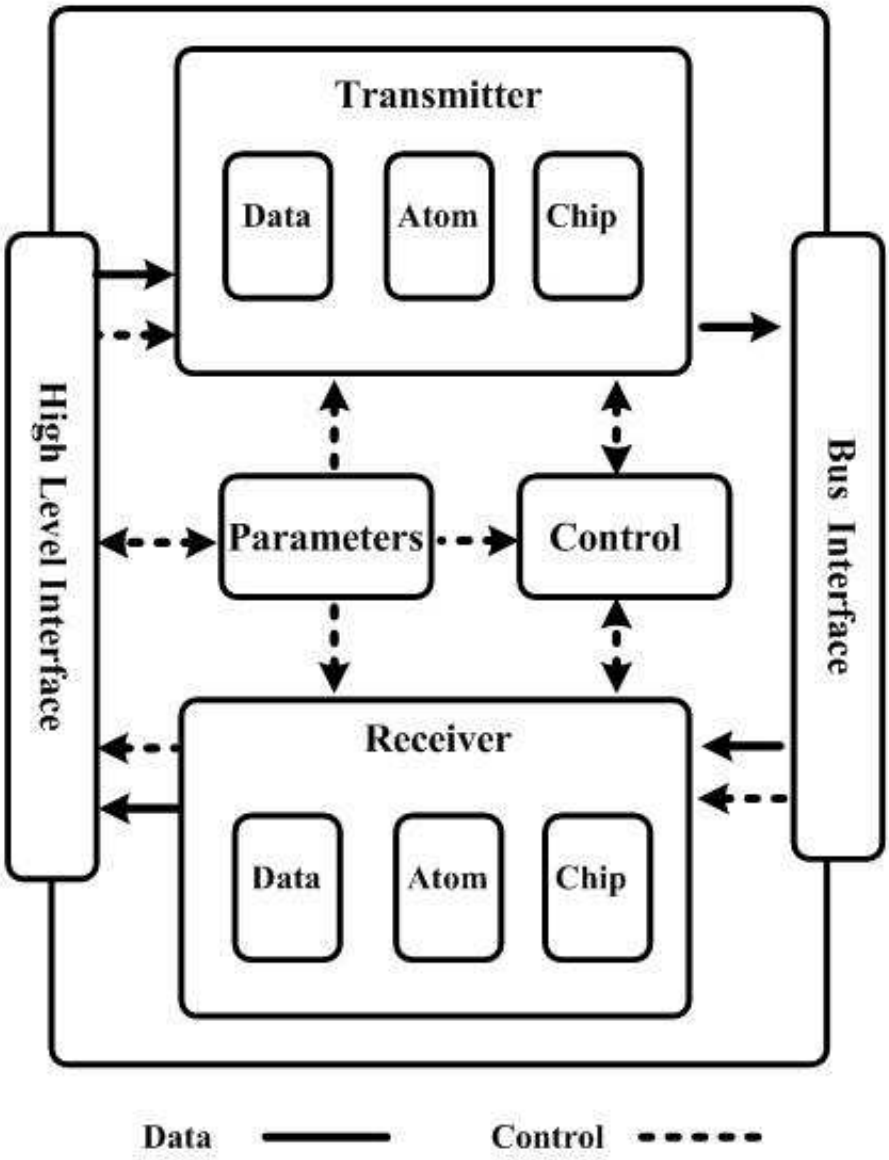


Figure 45 The schematic of the Bus-Modulator. It has two interfaces, namely, high and low level communications, and four functional block of parameters, control, transmitter and receiver.

The families of ATmega 48/ ATmega 88/ ATmega 168 and ATmega 328 [39] all have the USART capability, and operate on 20 MHz. ATmega 328 does have more memory and flash memory, but its inductions and memory access are word-based.

Therefore, in our application, its effective memory is basically the same as ATmega 168 family. Considering the mentioned features, ATmega 168PA seems to be a proper candidate. Atmel website describes this microcontroller as “The high-performance Atmel picoPower 8-bit AVR RISC-based microcontroller combines 16KB ISP flash memory with read-while-write capabilities, 512B EEPROM, 1KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8–5.5 volts.”

In addition to the microcontroller itself, only an external oscillation circuit is required to have running microcontroller. Such a circuit consists of two capacitors and a crystal. Figure 46 shows the schematic for the hardware implementation. In our implementation, some additional components are added to the design too. The general design has one LED and its current limiting resistor, for the purpose of troubleshooting. It also has an on board resistor for pulling up reset pin to V_{CC}. This resistor allows the microcontroller to start its operation upon the application of power to it. The reset pull-up resistor is a necessary component if the reset bottom is only used for resetting the Bus-Modulator, but it could be left out of the Bus-Modulator implementation, and let the communicating node implement it as a part of enable/disable decision making process for the Bus-Modulator. In our implementations, the Bus-Modulator V1.0 which is designed for testing the whole scheme on the Bus-Board has this resistor, but the Bus-Modulator V2.0, which is designed for real world application does not implement this resistor.

As the figure shows, the two connected pin 1 (PD3) and pin 2 (PD4), of the microcontroller, which are interconnected, are chosen to interface the NOR-bus. In this way, one pin could be the output pin and at the same time, the other one could act as the input. In this case pin 1, which is also the pin for external interrupt 1, which makes it perfect candidate for the interrupt enabled input for detection of the signal when it is in low power/ sleep mode.

On the other hand, for the high level communication of the Bus-Modulator, a 10-pin connector is used. This 10-pin connector provides power and reset signal to the

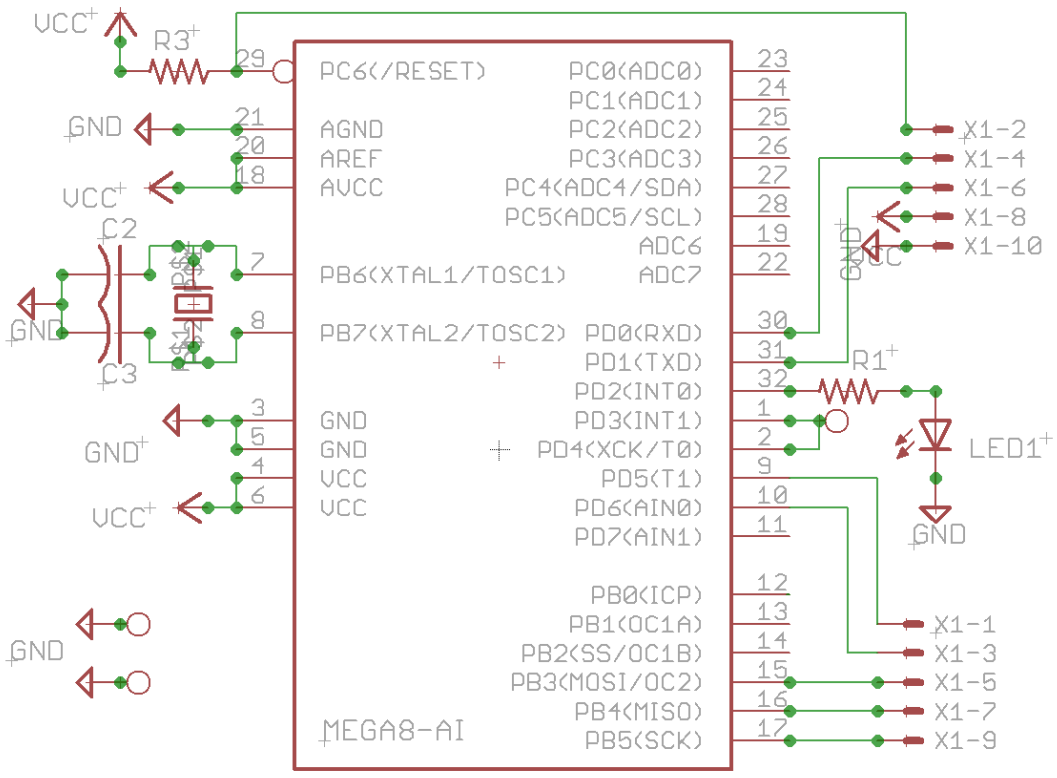


Figure 46 The schematic of the Bus-Modulator hardware, which is based on Atmel ATmega168PA.

microcontroller, as well as serial and parallel communication connections. The connection pins are chosen in a way that the 10-pin connector serves as in system programming (ISP) [47] port too. Table 4 shows the connections of the 10-pin connector, the microcontroller and the functionality of each pin.

Figure 47 shows the schematic and the actual board of the external high level interfacing board, which also is compatible with the standard AVRISP mkII [48] connector. And finally, figures 48 and 49 show the printed circuit board layout of both Bus-Modulators version 1.0 and version 2.0.

Table 4 Functionality and interconnection of the 10-pin connector of the Bus-Modulator

Connector Pin	Microcontroller	Functionality
1	09 - PD5	General I/O
2	29 - PC6	Reset
3	10 - PD6	General I/O
4	30 - PD0	General I/O - USART Rx
5	15 - PB3	General I/O - MOSI
6	31 - PD1	General I/O - USART Tx
7	16 - PB4	General I/O - MISO
8	04 - 06 VCC	VCC
9	17 – PB5	General I/O - SCK
10	03 - 05 - GND	GND

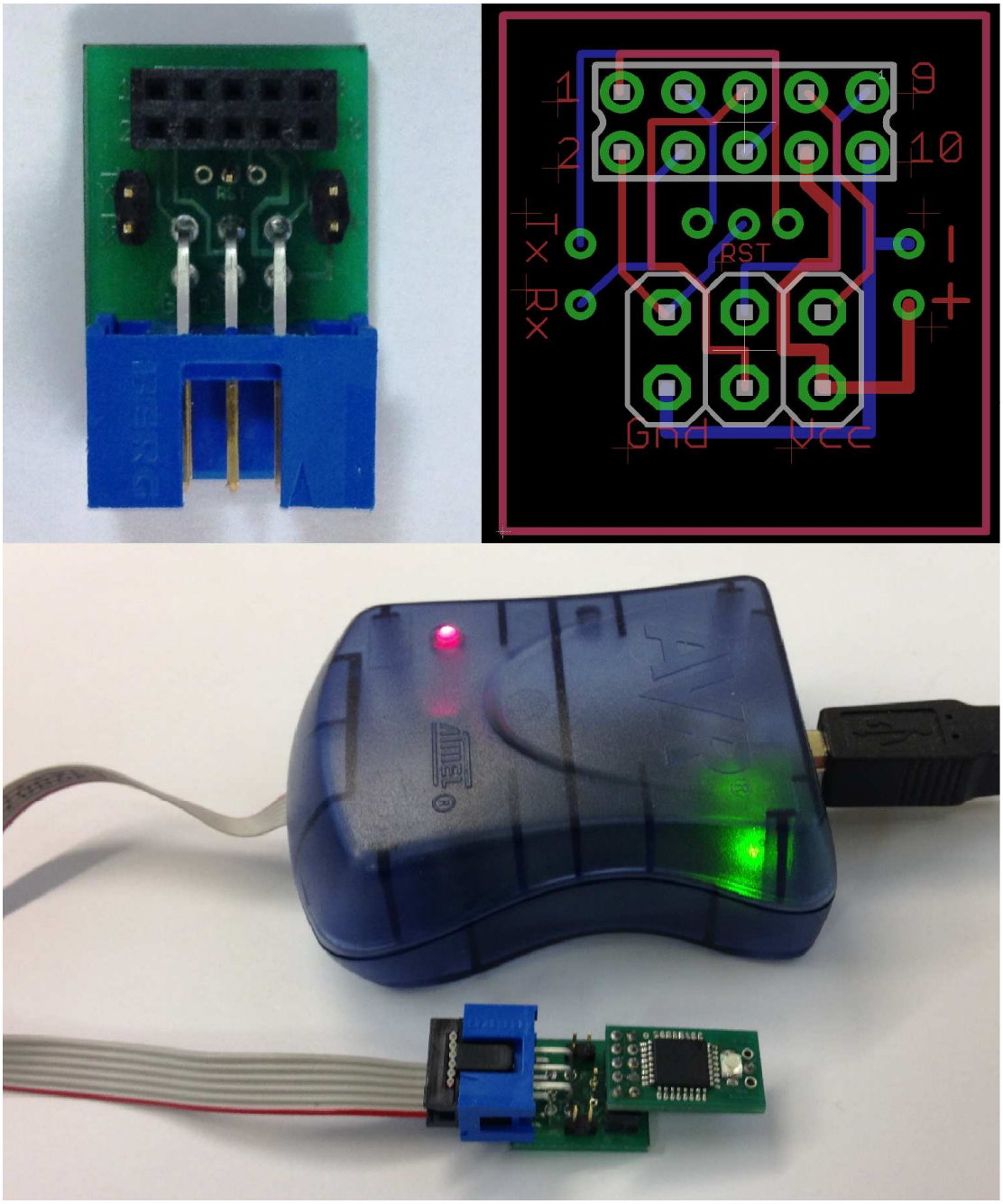


Figure 47 Printed circuit board for the external high level interfacing board (top right). The external high level interfacing board (top left). Interfacing of the Bus-Modulator V1.0 and The AVRISP programmer.

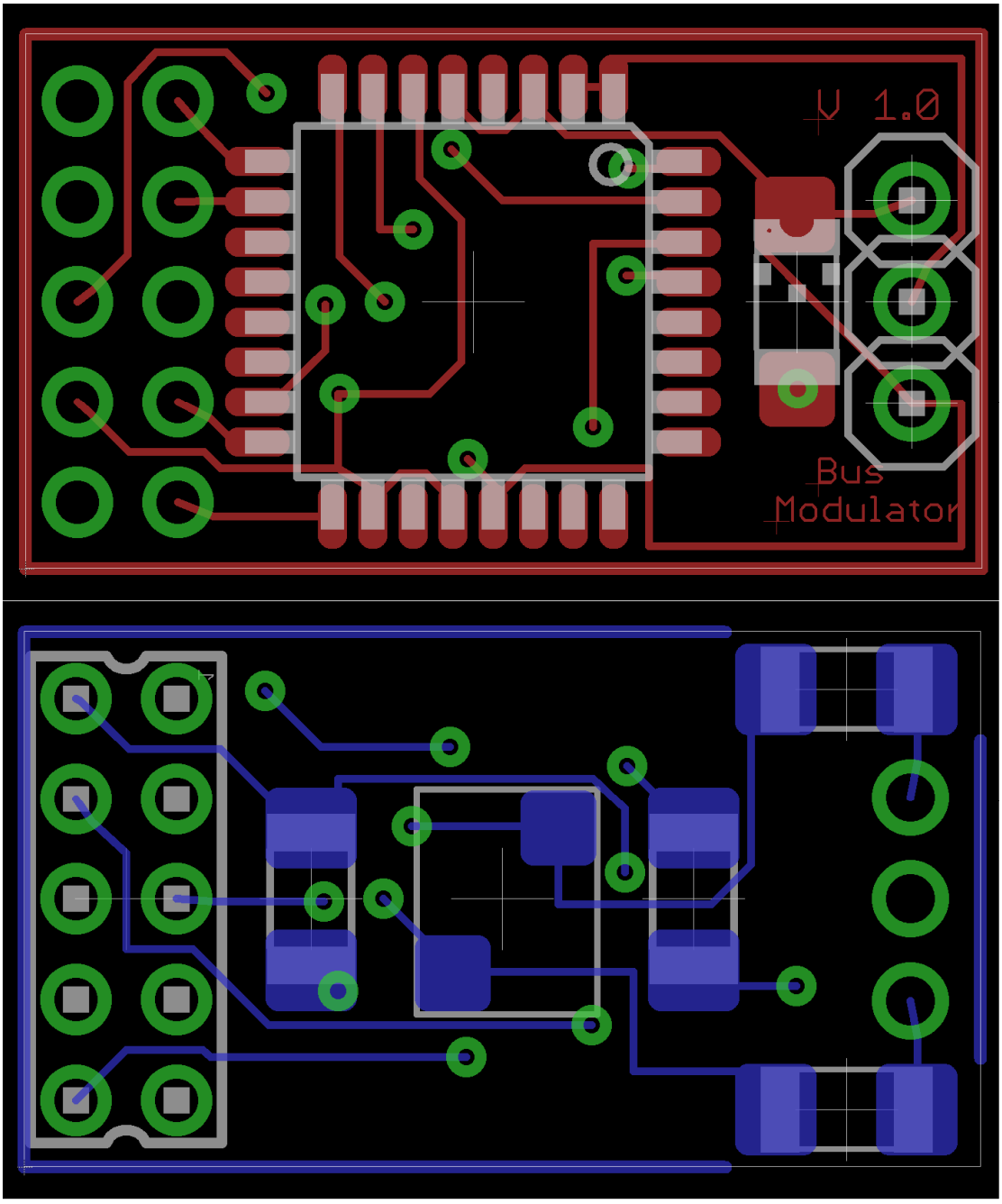


Figure 48 Top layer of the printed circuit board of the Bus-Modulator V1.0 (top). Bottom layer of the printed circuit board of the Bus-Modulator V1.0 (bottom).

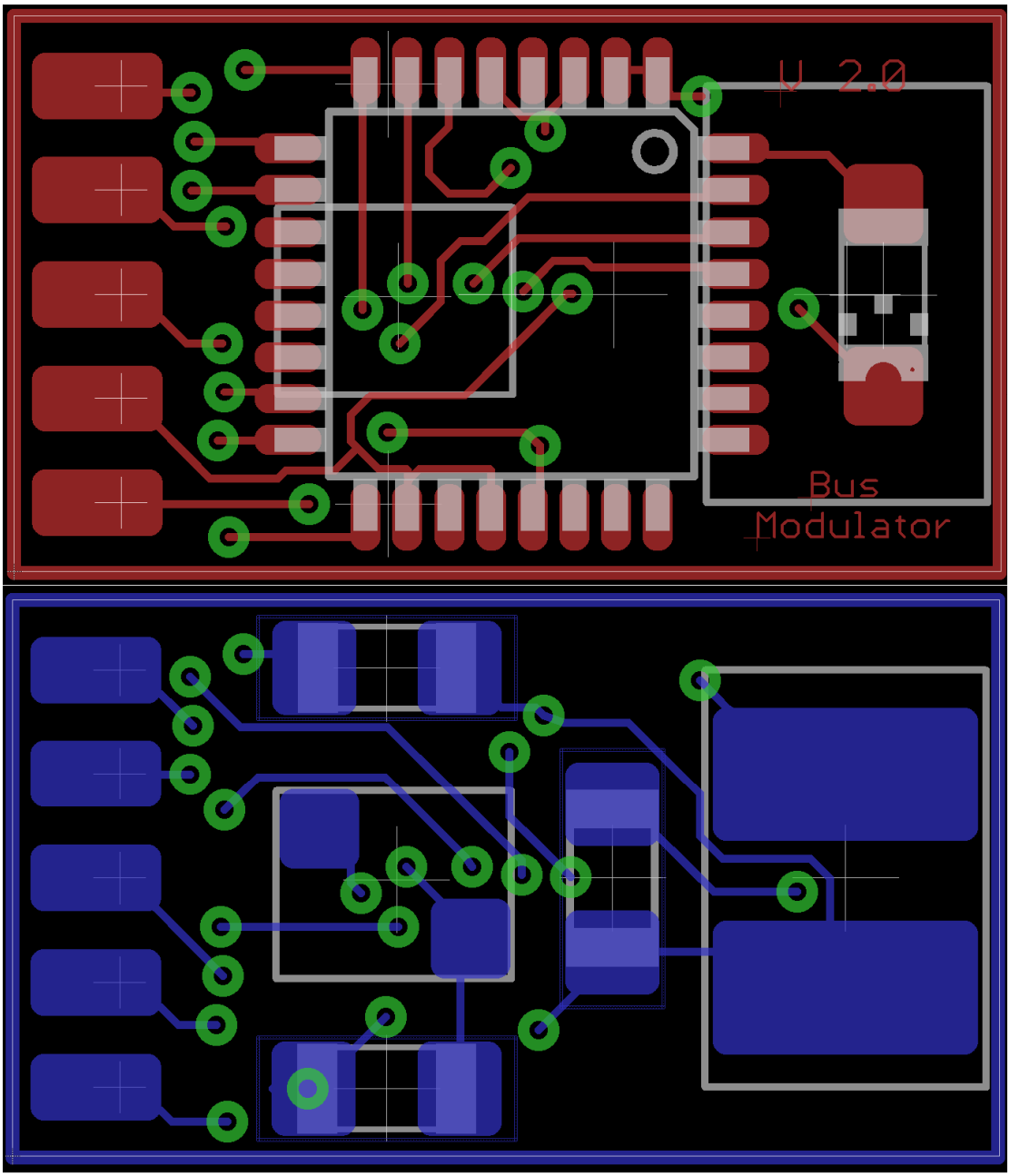


Figure 49 Top layer of the printed circuit board of the Bus-Modulator V2.0 (top). Bottom layer of the printed circuit board of the Bus-Modulator V2.0 (bottom).

Figure 50 and 51 show the actual Bus-Modulator version 10 and version 2.0.

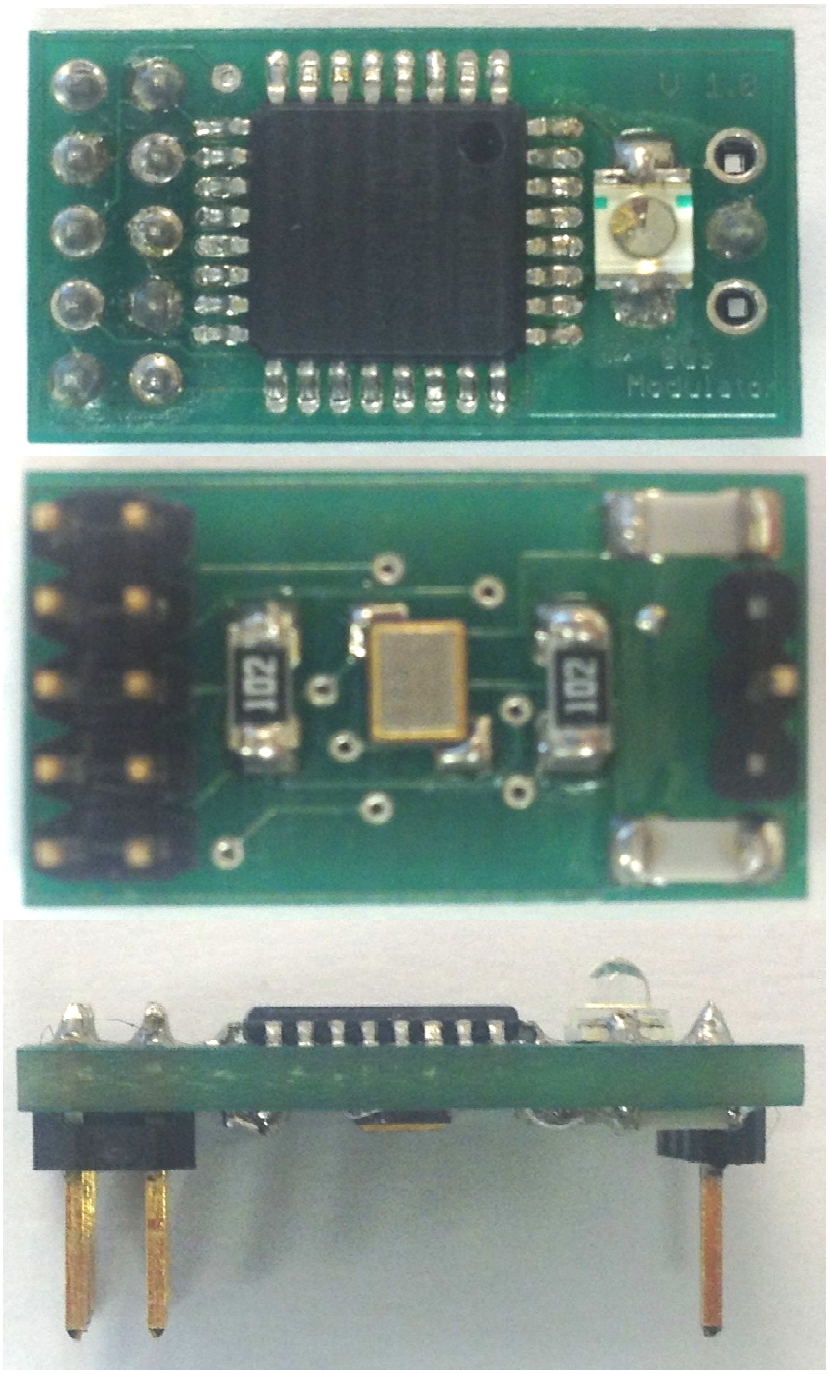


Figure 50 Top view, bottom view, and side view of the Bus-ModulatorV1.0.

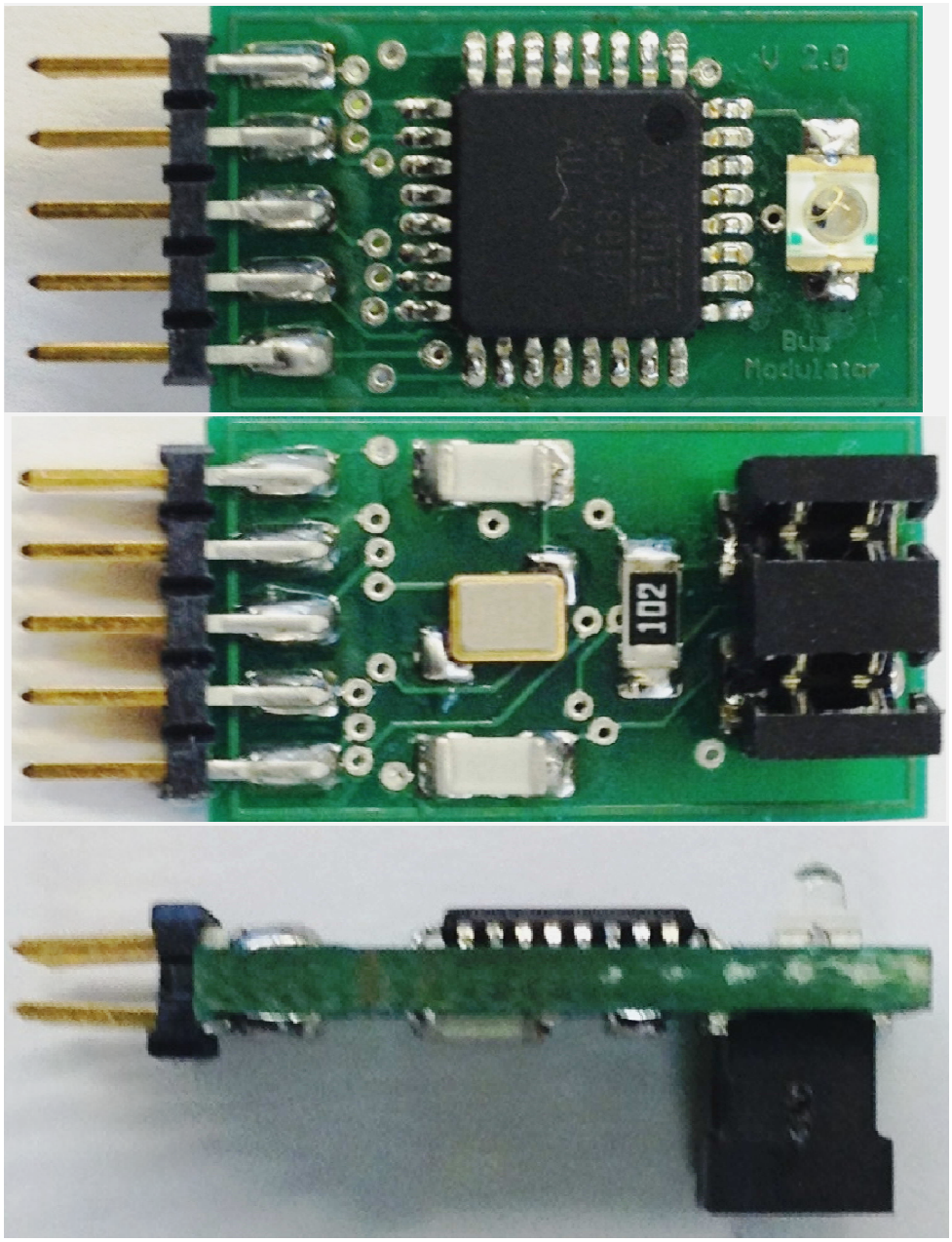


Figure 51 Top view, bottom view, and side view of the Bus-ModulatorV2.0.

In order to implement a bus set for testing the Wired-CDMA scheme, a motherboard, which provides a single wire connection for the Bus-Modulators, is designed. It is called Bus-Board. Up to thirteen Bus-Modulators could be plugged on to the Bus-Board in order to communicate with each other. When plugged, every single Bu-Modulator is able to set up independently from the rest of them. Figure 52 shows the Bus-Board.

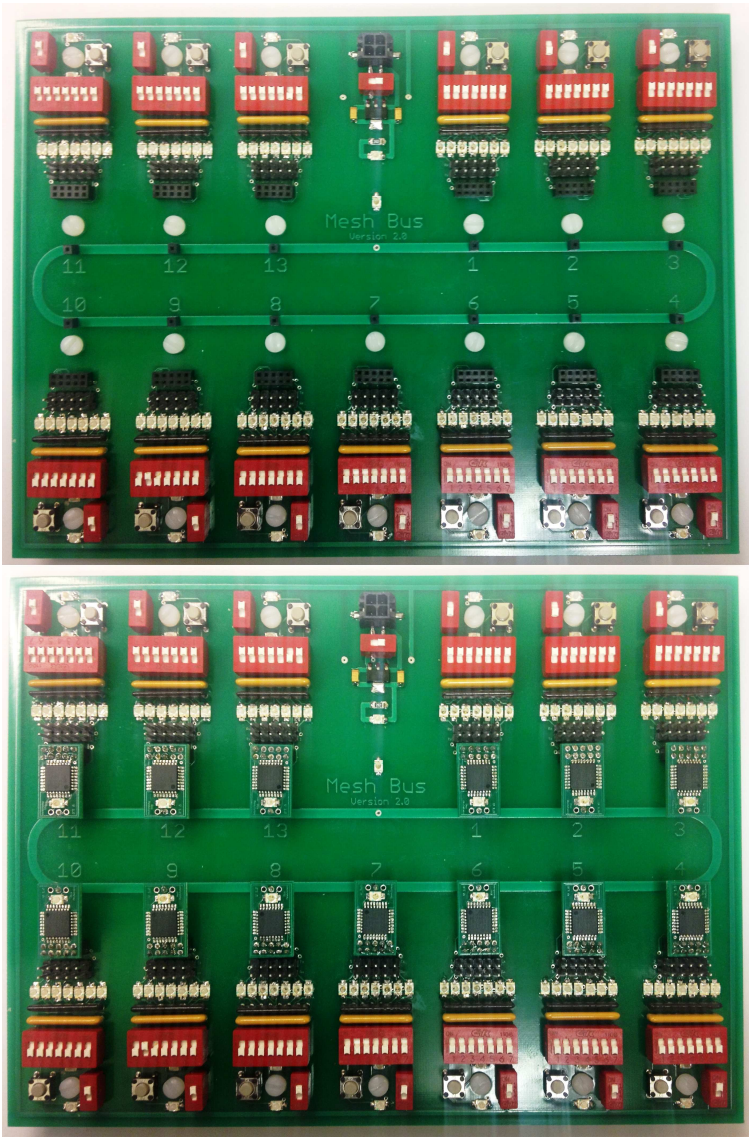


Figure 52 Top view of the Bus-Board (top). The Bus-Board with thirteen Bus-ModulatorV1.0 plugged in.

Firmware

The firmware for the Bus-Modulators is written in the standard C language of the Atmel AVR Studio. There are different sections to the program, but the most important one are the initialization processes, and the while loop in the main body of the programs

After the microcontroller is powered on, an initialization function, namely *init()* is called which is responsible to set the default values of required parameters and address book and registers. Then, the program enters an infinite loop, as the main states of the system. The rest of the operation of the Bus-Modulators occurs as the responses to the interrupts. In order to avoid in efficiencies of parameter passing to the functions, most of the variables are defined as global variables and are accessible within all the functions.

The main cycle of the program, named *chip operation cycle*, is based on an internal timer which is responsible for the duration of the chips. Every time the timer overflows, an interrupt occurs, and within the interrupt service routine, the two main operations of the Bus-Modulators take place: transmission and detection. The rest of this section will mainly discuss the implementation of these two operations, but before that some issues about the timers and interrupts must be address.

The ATmega168PA has 3 timers; the time 1 is a 16 bit timer and the timers 2 and timer 3 are 8 bit ones. In this implementation, time 1 is assigned to the chip operation cycle (chip transmission/ chip detection). Considering the $20\ MHz$ operation frequency of the ATmega168AP, the 16 bit timer could be initialized quite flexibly in order to allow for different chip rates. For example the chip duration could be from $\sim 1\mu s$ to $1ms$, and in case of using prescaler up to a few seconds.

Every time the time 1 interrupts, the program jumps to function associated with the timer 1 named interrupt service routine, and performs the required tasks within it. After that, the program jumps back to the original flow of the program until the timer 1 is activated again. The tasks inside the interrupt service routine could be different according to the state of the system; therefore, the time of their execution times are different. In

order to have constant chip operation rate, the interrupt must be activated manually inside the interrupt service routine instead of being automatically activation by finishing the interrupt service routine. In this way, we can make sure that after fixed amount of time of the interrupts of the timer, it is activated again without being worried about the variable time of the execution of the interrupt service routine.

However, the manual activation of the interrupts creates the risk of occurrence of other interrupts in the middle of the interrupt service routine which is currently being executed. This subject will be discussed later in this section. For now, we just simply assume that the time 1 provides us with a fixed chip operation rate.

Every node has an ID which is programmed to the chip, or could be configured for any node. Also, all the Bus-Modulators are preprogrammed with codebooks of different length. Every time a Bus-Modulator is powered up, knowing its own ID and the code length n, it is able to choose a codeword from the codebook. On the other hand, every time a Bus-Modulator requires transmitting data to another node, it uses the ID of the destination to specify the codeword of the destination from the same address book.

Transmission

In the transmission section of the firmware, we deal with three types of transmissions: chip transmission, Atom transmission, and data transmission. One of the variable which is involved in transmissions is *TxEn*. This variable is used in the interrupt service routine to enable or disable the transmission of a chip (therefore, Atom and data) at any moment. If it is set to enable the transmission, in every chip operation cycle, the function *transmisChip()* is called. Transmission of the chips occurs through resetting the output pin for transmission of 1 and activation of the internal pull-up resistor for transmission of 0.

The variable $index$ is a counter which counts from zero to n. It is responsible for transmission of the Atoms. For transmission of an Atom, the function *transmiAtom()* is

called in which the *index* is set to zero. Inside the *transmisChip()* the variable *index* is incremented by one every time a chip is transmitted while it is still less than n the corresponding chip (zero or one) is transmitted. Every time the variable index equals the position of one of the destination code 1 chips (i.e., *codeBook[TXto][0]*, *codeBook[TXto][1], and codeBook[TXto][2]*), it transmits 1, otherwise a zero is transmitted.

In our scheme, the communication of actual data occurs based on the time distancing of the transmissions of multiple Atoms. This job is done through the *transmitData()* function which takes the actual data for the transmission as an argument. The variable *TxEn* is set to enable the transmission process as the first operation of this function, and is reset to disable it as its last one. The variable *timeCount* which is the main clock of the program helps time distancing the Atoms. The *timeCount* which is a unsigned long variable (i.e., it ranges from 0 to $2^{32}-1$) is incremented inside the interrupt service routine of the chip operation cycle.

The time distance coding for this implementation uses three Atoms as follows: the two first Atoms are transmitted with zero distance, and the third one is transmitted after a linear function the data. Figure 53 shows such a scheme.

The reason to have three Atoms instead of two is that when two Atoms are used for data communication, and there is continues communication, if there is no reference point, there is no way to figure out which Atom is the first Atom and which one is the time distanced one. And since the communication is not error free, it is very easy to lose the original reference point for the receivers. In the case of three Atom transmissions, there is no need to have a global reference point; detection of two consecutive Atoms with distance zero determines that the next Atom should be the data-bearing time distanced one. Figure 54 shows the effect of losing global reference in the two-Atom data transmission scenario.

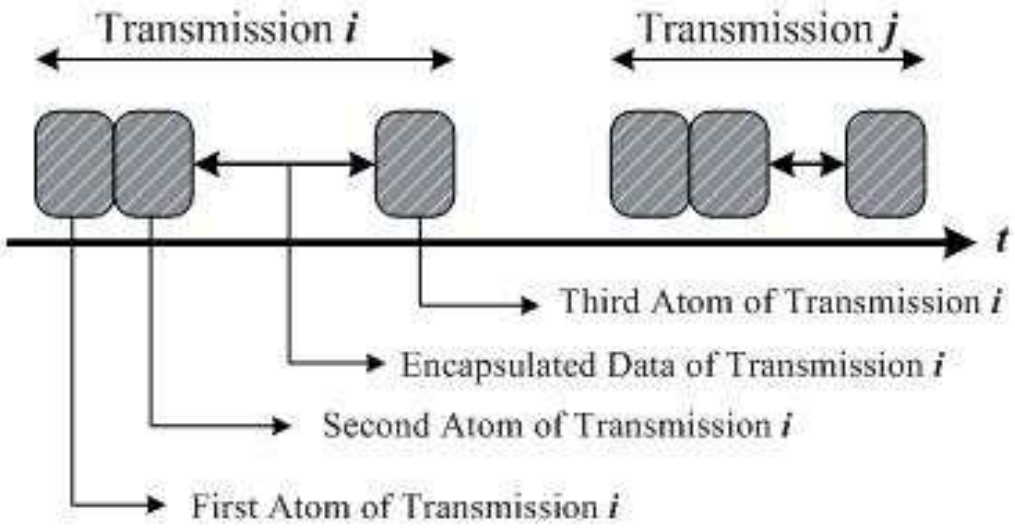


Figure 53 Arrangement of Atoms for Complex Data 1. Every information packet starts with two zero-distanced Atoms which followed with an Atom timed distanced by the data.

Detection

Similar to transmission process, detection also occurs in three different levels: chip detection, Atom detection, and data detection. Since the transmission in our scheme is not synchronized, and there could be more than one asynchronous transmission on the bus, there is no way for a node to know when to sample the bus. The simplest scheme for a node is to sample the bus at the chip rate, but it could cause over detection. Over detection happens when just one chip is transmitted but the receiving node percepts the resulting state of the bus as is there are two consecutive transmissions. Over detection occurs because of smoothed rising and falling edges of the transmitted chips. In such a scenario, as figure 55 shows, is the sampling occurs at the falling edge of a transmitted chip, there is high chance that the next sample also perceived as detection, because the rising edge of the transmitted chip is not sharp, and at the time of the next sample, its values is still considered as low.

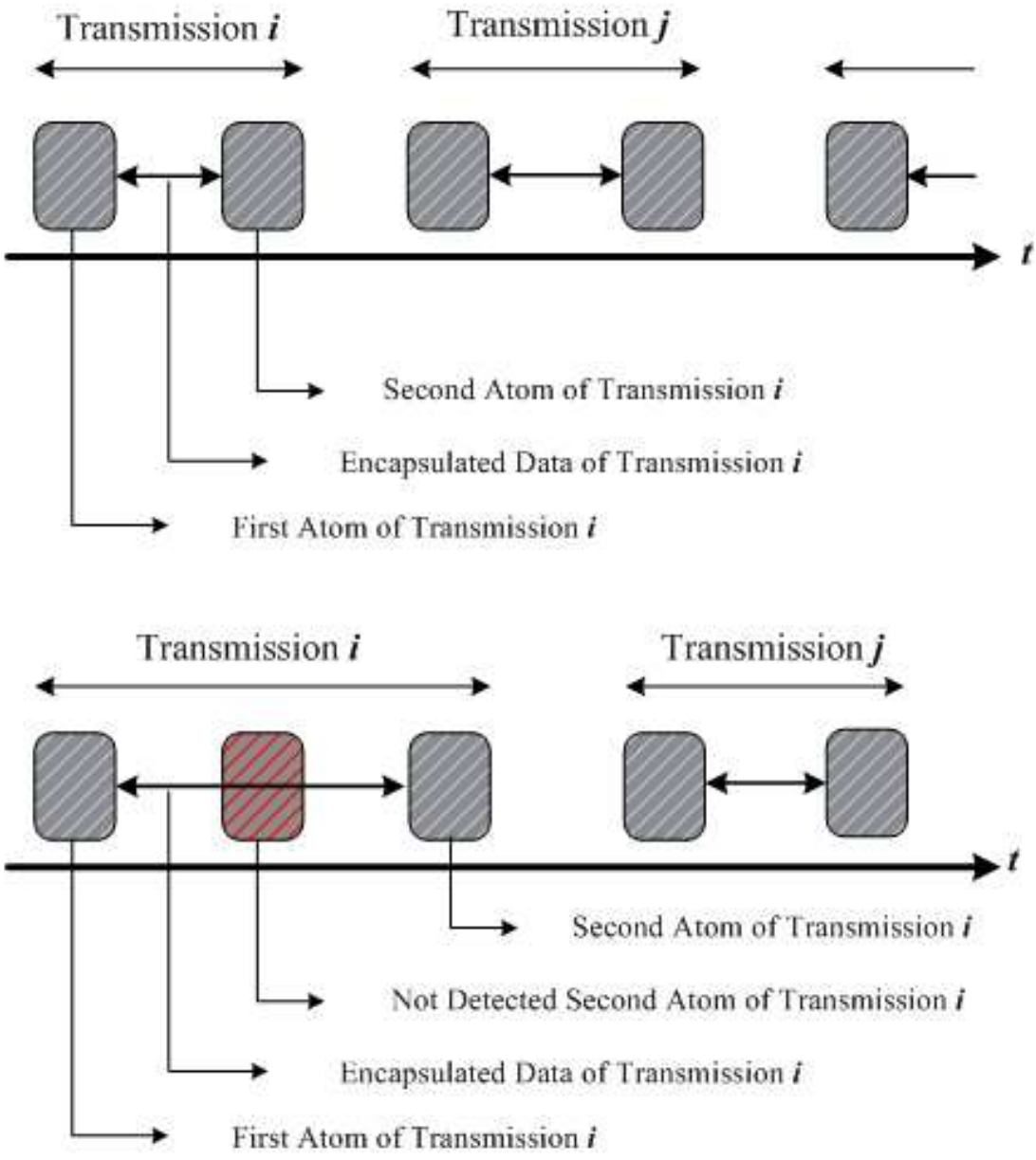


Figure 54 Effect of the losing one Atom detection on miss interpreting the encapsulated data in a two-Atom arrangement system.

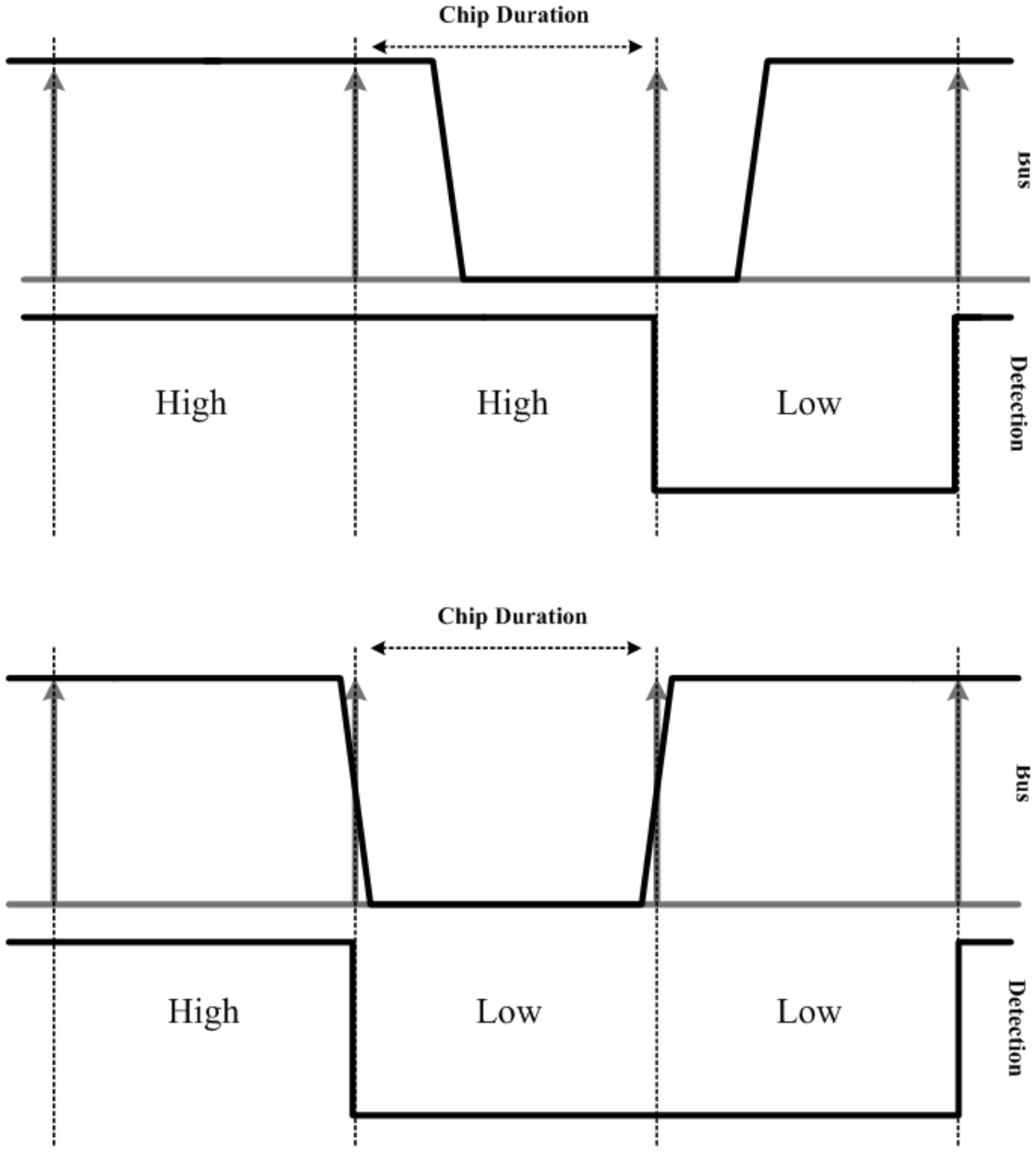


Figure 55 Top diagram shows the situation where the sampling occurs in the middle of a chip, and detection of right. The bottom diagram shows the over detection, where the sampling occurs at the falling edge of a low chip, therefore the next sample coincides with the rising edge of the same chip and may be detected as another low chip.

One way to overcome the over detection problem is sub sampling the bus. In other words, during each chip, sample the bus more than just once and based on such sub sampling, just one sub sample coincide with the rising edge of the same chip, and the rest of the sub samples of the new chip duration, truly represent the state of the bus.

There are two ways to implement sub sampling. One method is using just one interrupt for sub sampling and after a certain numbers of such sub samples do the decision about them. In this case, the time interval between the samples are an integer multiple of the time interval between sub samples. This method of sub sampling is similar to the sampling, because of the imperfection of the rising and falling edges of the chips. Since the transmissions are asynchronous, if the falling edge of a chip co inside with the first sub sample, there are chances that the chip lasts until the first sub sample of the next chip. Figure 56 and 57 show such a scenario, on an oscilloscope, where number of the sub samples per chip could be different because of the imperfect shape of the chips.

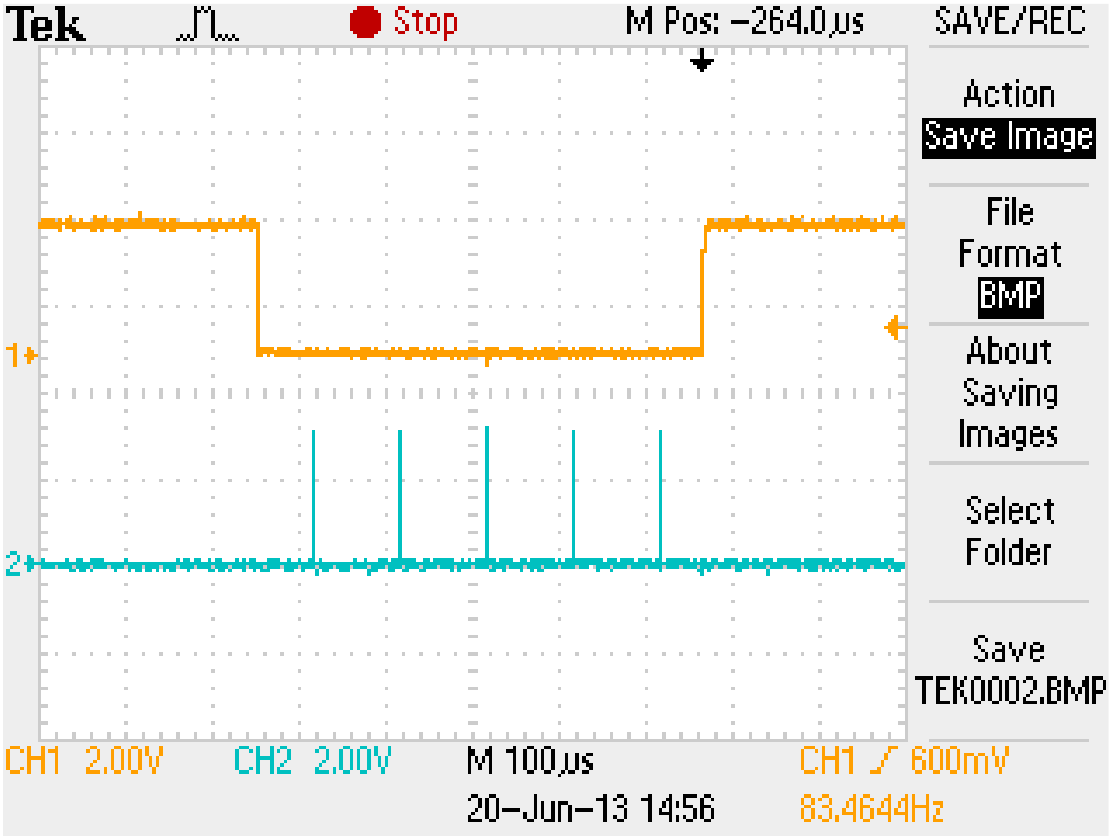


Figure 56 Five sub samples (green spikes) are accommodated in a single chip duration (yellow signal.)

In order to solve the mentioned problem, is to design the time interval of sample and sub samples in a way that it is impossible to have more than some numbers of sub samples in a sample duration. In other words the sample intervals and sub samples intervals are prime relative to each other. For example, the sub samples time interval is 3 time units and the sample intervals are 10 time units. In this, even if the first sub sample coincides with the falling edge of the chip or not, no more than 3 sub sample are possible to happen during one sample.

In order to implement this type of sub sampling, two independent interrupts are required. One interrupt is for sub sampling, which periodically occurs and probes the bus, and the other one which is responsible for chip cycle operation. It must be mentioned that

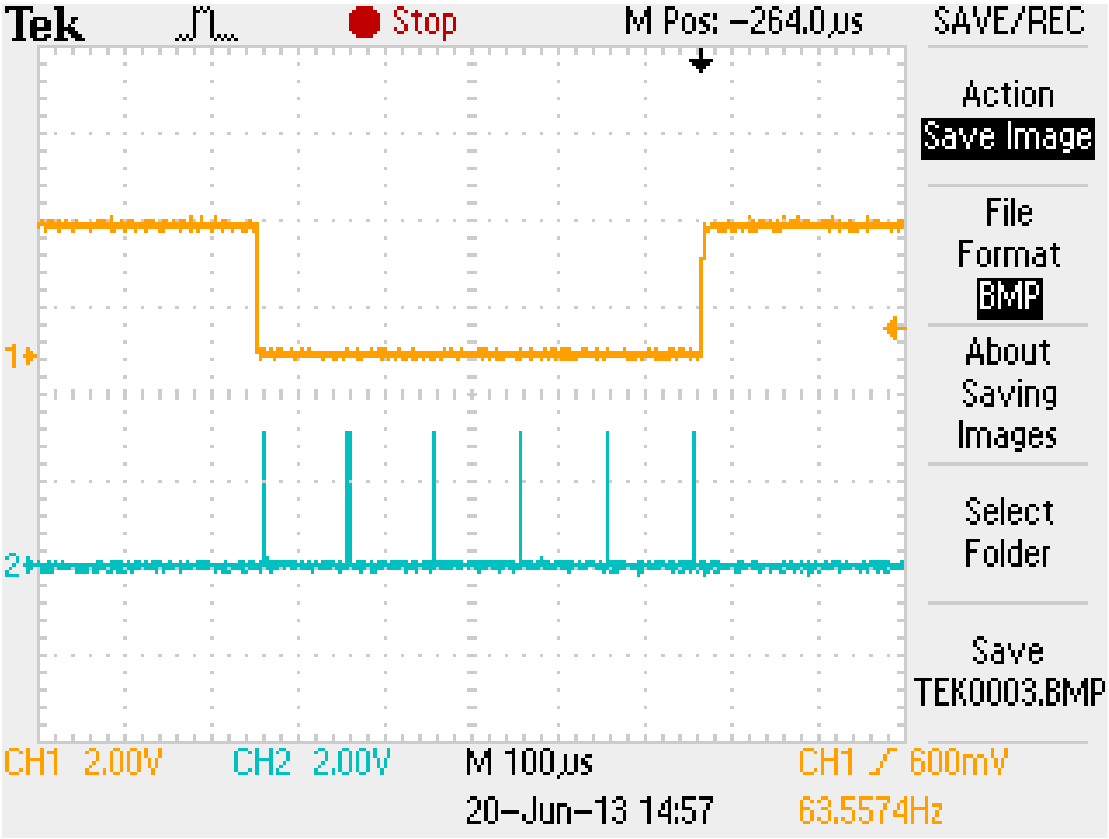


Figure 57 The chip (yellow signal) is wide enough to accommodate six sub samples.

one of the sampling interrupt's tasks is to check on the sub samples, which are independently gathered, then, determine whether the last chip was low or high.

Unfortunately, The Atmel AVR family is not able to handle more than one interrupt at a time, or prioritize them. Therefore, implementing the scenario with two independent interrupts for sampling and sub sampling is not very straight forward. In the AVR family, when an interrupt occurs, the rest of the interrupts are masked until the end of the execution of current interrupt service routine. In our implementation, there are times that one of the interrupts occurs in the middle of the other one; therefore, it is masked, as if it never happened. Such interrupts conflict causes some problem both in transmission and detection both for chip operation and sub sampling. Frequency of such

interrupts conflict is related to the frequency and duration of the interrupt service routines which mask the other interrupts. In the next chapter, some numerical values for such interrupt conflicts and their effects of performance are presented.

Having the correct detections at chip level, the next step is the detection of the Atoms. Every node Bus-Modulator has a unique Atom which constantly correlates the string of the previously detected chips to it in order to find a high correlation, thus a correct detection. In order to implement such a correlation mechanism, the character array named *buffer* with the length of n, and a character variable named *bufferIndex* are defined globally. At every chip cycle operation, within the interrupt service routine, if the variable *index* is set, the function *detection()* is called. In this function, the *buffer* gets updated with the latest chip on the bus. The *bufferIndex* is the pointer to the currently last updated cell of the *buffer*, and since it has the length of n, it is overwritten after every n chips. In order to have such a circular *buffer*, all the arithmetic operations on the pointers such as *bufferIndex* are in modulo n.

On every call of the function *detection()*, is the latest update of the *buffer*, i.e., the current chip on the bus is a low, the last $w - 1$ positions of the Atom, here the two other chips, are check in the *buffer* and if they all are zeros, a new correct Atom is detected on the bus. Setting the variable *dtcSucs* signals the new detection.

Inside the *detection()*, the time of the detection, i.e. the value of *tiemCount* is recorded in a circular array by the name of *dtcArray*. The variable *dtcArrayMax* is the pointer to the latest detection time, and all the operation on it is being done with respect to the length of the *dtcArray*. Figure 58 shows multiple detection of the Atoms whose time of detections are recorded as the values of *dtcArray*.

In our implementation, the length of *dtcArray* is 15. It means that after the first 15 detection, the array will remain full of the detection times of the last 15 detection. Having such recorded times; we need a mechanism to extract the data from them.

Such a data extraction occurs through calling the function *dtcData()* in which a state machine is implemented. The state machine should be designed based on the coding scheme which codes the actual data by the Atoms. For example, In our implementation of *transmitData()*, two Atoms are transmitted with time distance of zero, and the third one is transmitted with the specific time distance which represent the data.

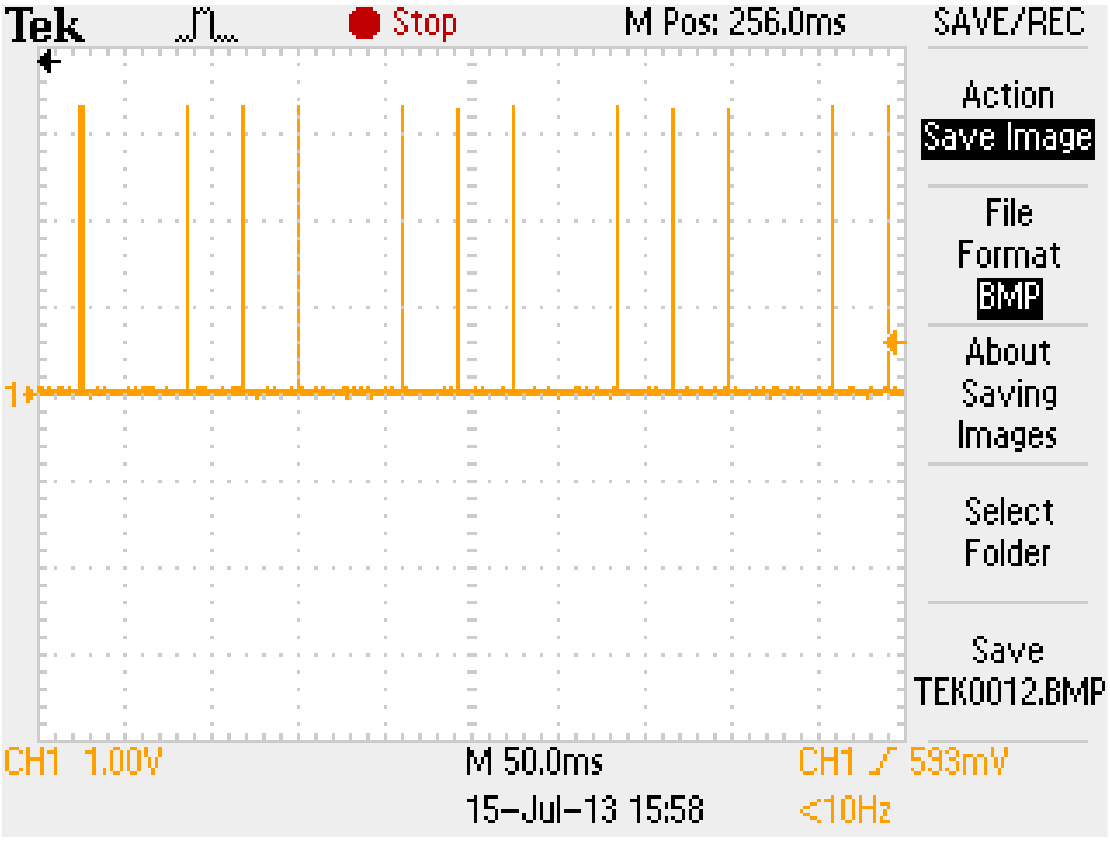


Figure 58 Detected Atoms. Every yellow spike shows a detected Atom whose detection time is recorded in the *dtcArray*.

CHAPTER 6

PERFORMANCE ANALYSIS OF WIRED-CDMA

In this section, we present the results of the extensive simulations whish are done in order to analyze the performance of the Wired-CDMA systems. The simulations are based on a software simulation engine which is written in Java, and allows for implementing various scenarios with different parameters. The main objective of the simulations is to assess the error rate for Atom and data communications based on the Wired-CDMA scheme.

However, the simulations are based on software that may not be able to catch all the imperfections of a real world implementation of such a scheme. Therefore, a substantial question is needed to be answered: how can we make sure that the results of the software simulations are relevant to real world, physical implementation of the same scheme. How can we make sure that there is not something fundamentally wrong the way that we have simulated the whole setting? To answer this question, some means are necessary to validate the results of the simulations to some extents. One of such a validation means is a hardware implementation of the scheme. Of course there are different levels of precision and robustness in any hardware or software implementation, but the more accurate and robust implementation, the more resources such an implementation requires.

In this thesis, the main objective is not a perfect hardware implementation of the Wired-CDMA, but an implementation that shows that the simulations are not completely out of touch with reality. Such an objective is achievable if the implemented hardware provides experimental results that follow the same trend of the software simulation, and as we see in this chapter, the hardware implementation shows that the software simulations are on the right track, and the result they provide could be trusted within some confidence interval.

Knowing that the simulations are reliable enough, many more simulation could be done using the same simulation engine to evaluate the performance of the system, and study the effects of different parameters on the performance.

Predictable errors in the hardware implementation

As mention in the previous chapter, our implementation hardware for Bus-Modulators is based on the idea of sub sampling for the chip detection. Implementation of such a scheme requires two interrupts, which the ATmega 168PA is not able to handle. Therefore, some errors are associated with this specific hardware implementation. Every time one of the two interrupts happens during the interrupt service routine of the other one, one sub sample is missed, or one chip detection/transmission is missed. Therefore, we can calculate the frequency of such double interrupts situation and expect that the hardware acts in a way that the same ratio of errors is taken to the consideration as the hardware implementation error rate. This way, we can compare the simulation result with the hardware implementation results.

For example, for one scenario of hardware implementation, the chip operation cycle is $1010\ \mu s$, which $20\ \mu s$ of it is the time during which all the interrupts are masked. On the other hand, the period of sub samples is $290\ \mu s$ and the interrupt service routine lasts for $2.5\ \mu s$. Calculation shows that such a double interrupts occurs 1344 times in every 60 s. If we assume that every one of such a double interrupts prevent one chip transmission or prevent chip detection, 1344 chips are in error. But not all the erroneous chips cause trouble. The only chips that their erroneous transmission/redetection affects the transmission of the Atoms are the “1” chips, which pull the bus to low.

In the simulations, the Atoms are of length of 128 chips, and are continuously transmitted with the distance of one Atom’s length from each other. In this setting, considering that the weight of the Atoms is 3, only $3/(128 + 128)$ of times there is a

"1" chip to transmit. Therefore effective number of the erroneous chips is $1344 * 3/256$, or 15.75 "1" chip per second. The worst case scenario is the case that just one erroneous chip belongs to every Atom; therefore, theoretically, there will be 15.57 misdetections of Atoms per second. The duration of each chip is $1010\ \mu s$ and every 256 chip, one Atom is transmitted; the theoretical transmission rate is 3.861 Atom per second.

On the other hand, the hardware implementation of the same scenario shows a different number. In the hardware scenario, one Bus-Modulator continuously transmits one Atoms of length 128, with the time interval of two Atoms, and another Bus-Modulator detects such an Atom. Number of the detections for half an hour, i.e. $1800\ s$, is 6481. Therefore, the practical transmission rate is 3.6005 transmission/detection per second.

The ratio of practical transmission/detection rate, over the theoretical transmission rate, here $3.6005/3.861 = 0.9325$, shows the ratio that we should expect when we compare the theoretical analysis to the hardware implementations. This 6.74% error rate is the mere result of using two independent interrupts for implementation of sub sampling and sampling processes.

Simulation of transmission/Detection

The first sets of simulations are based on the transmission of the unary messages called Atoms. Basically, an Atom is transmitted by one node at a specific time and another node tries to detect it. These two nodes are not synchronized; therefore the detecting node should be able to detect the transmitted Atom by delay of at most one chip, unless an error occurs. The transmitter and receiver are not alone on the bus, and of course there are other nodes which are transmitting different Atoms on the bus. Such transmissions are considered as interference, which could cause errors in detection. Figure 59 shows the result of Atom transmission/detection software simulation for

different length of Atoms, n, and different number of the nodes on the bus, N. In this simulation, the ratio of error shows the relative number of the times that an Atom is transmitted, but it is not detected at the same time by the receiver node. The experiment is repeated 10^9 times. As the graph shows, when longer Atoms are used, ratio of error is smaller compared to situation where shorter Atoms are used, for the same number of the nodes on the bus.

In order to validate the software simulation results, we need repeat the experiment for the Bus-Modulators, and to be able to measure the exact time of transmission and detection. Although it is possible to do that, but it is time consuming. Therefore, another scenario is defined and simulated which does not require such exact time measurements, and is much easier to evaluate in hardware.

In this scenario, a node tries to detect an Atom which is not transmitted. Therefore, if it succeed detecting such an Atom, it means the presence of Atom on the bus, is the mere result of the interference by the transmissions of the other nodes. Therefore, such a detection of not transmitted Atoms is an error. Figure 60 shows the simulation result for this scenario.

As the graph shows, the simulation results for the ratio of error versus the number of the nodes on the bus, N, demonstrate the same trends as the previous set of simulations. For the same number of the nodes on the bus, longer Atoms, show less error.

This scenario is easy to validate through the hardware implementation, because it does not require any time measurement. Different numbers of the Bus-Modulators are on the bus transmitting their Atoms, and one specific Bus-Modulator keeps detecting and Atom which is not transmitted. The experiment is done for five different numbers of the nodes on the bus, and for time duration of $1800\,s$, which is equivalent to 6950 transmissions. Figure 61 shows the hardware and software simulation results. As the graph shows, the Hardware implementation shows the same trend as the software simulation, but of course with different values. The graph also shows the ratio of error for

hardware implementation over the software simulation. The ratio is between 1.1 to 1.2, which accounts for 10% to 20% errors of hardware implementation over the software simulation.

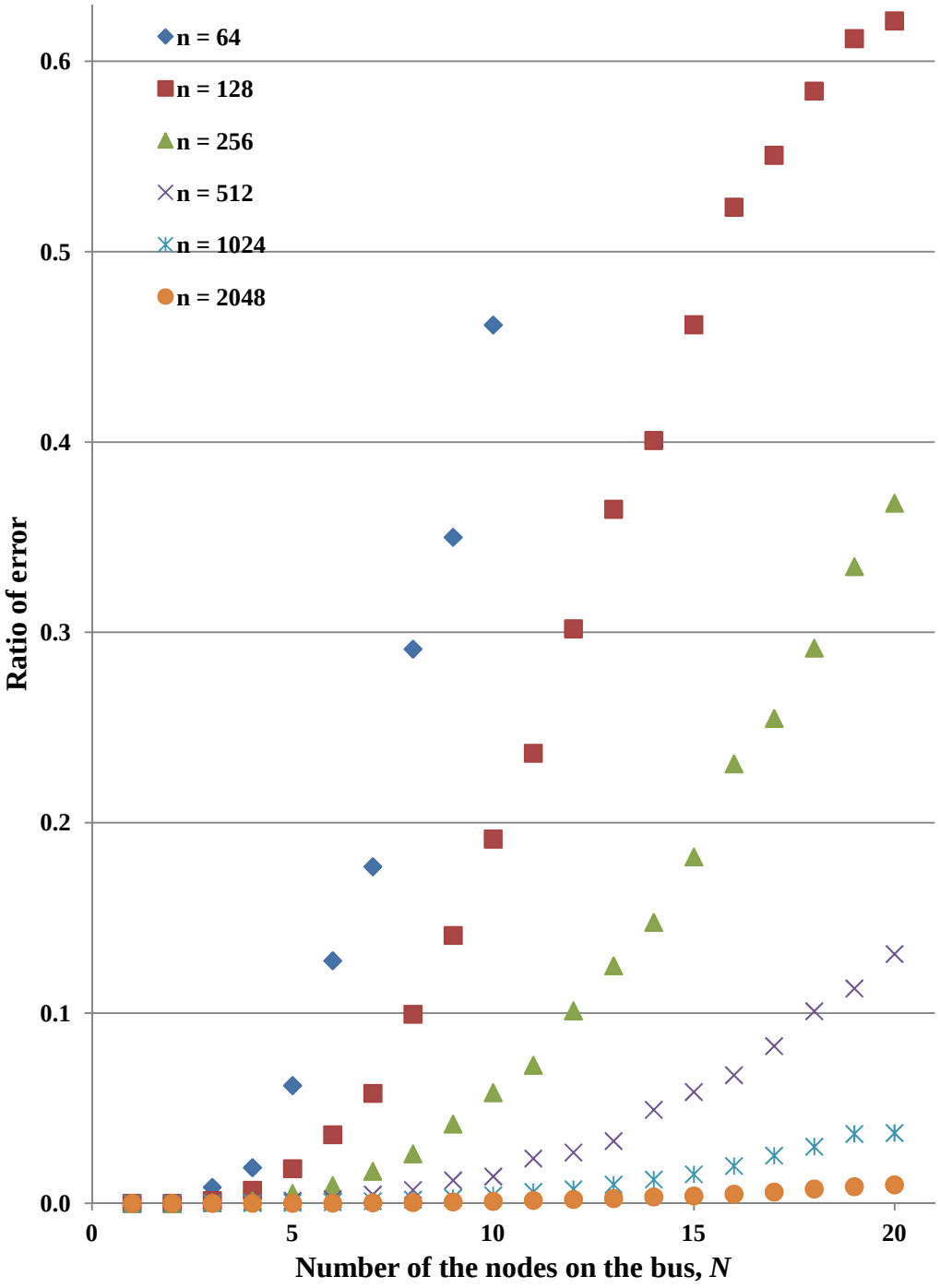


Figure 59 Software simulation of the performance analysis of the detection of the transmitted Atoms versus the number of the nodes on the bus, for different length of the Atoms.

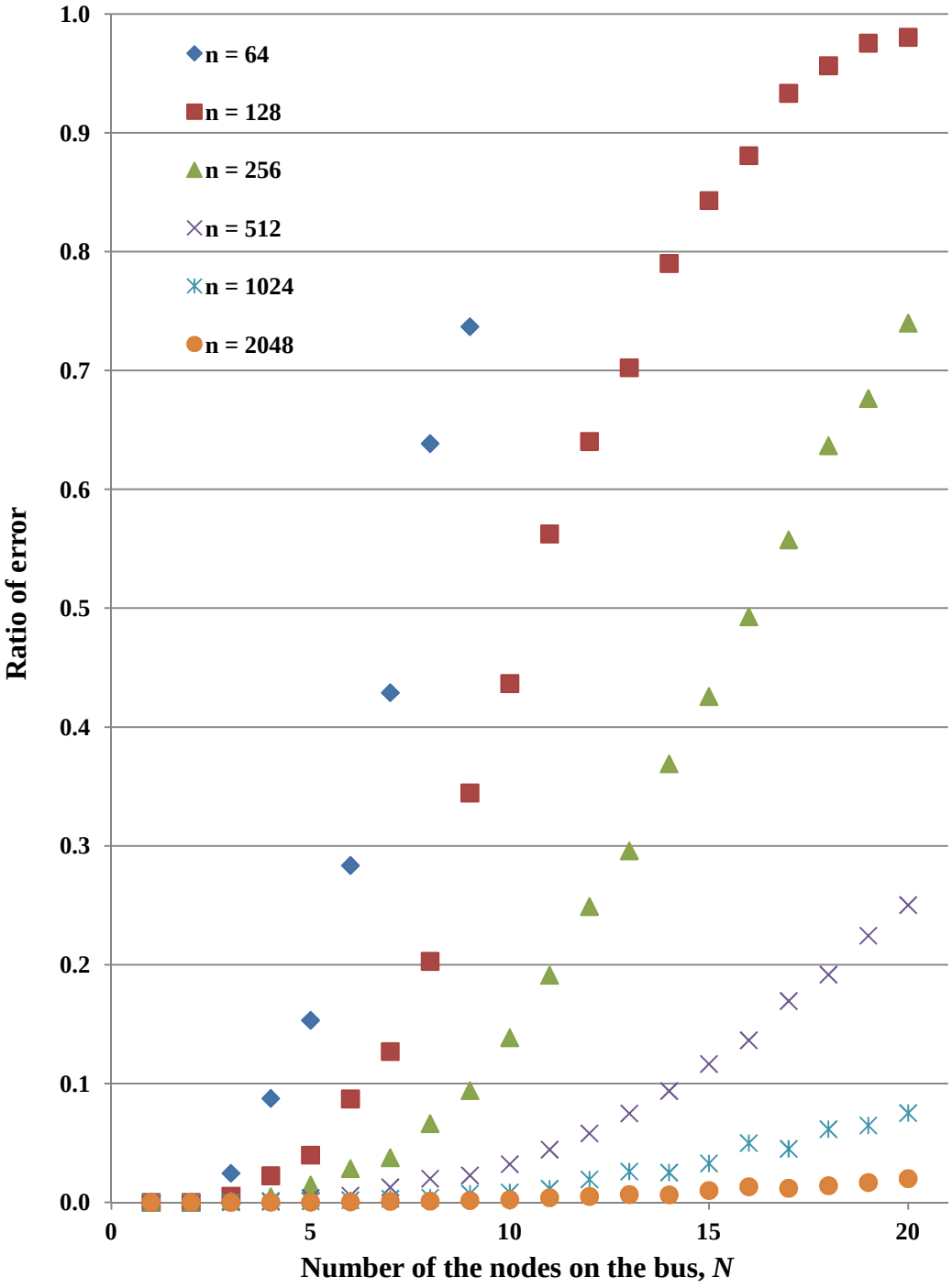


Figure 60 Software simulation of the performance analysis of the detection of the not-transmitted Atoms versus the number of the nodes on the bus, for different length of the Atoms.

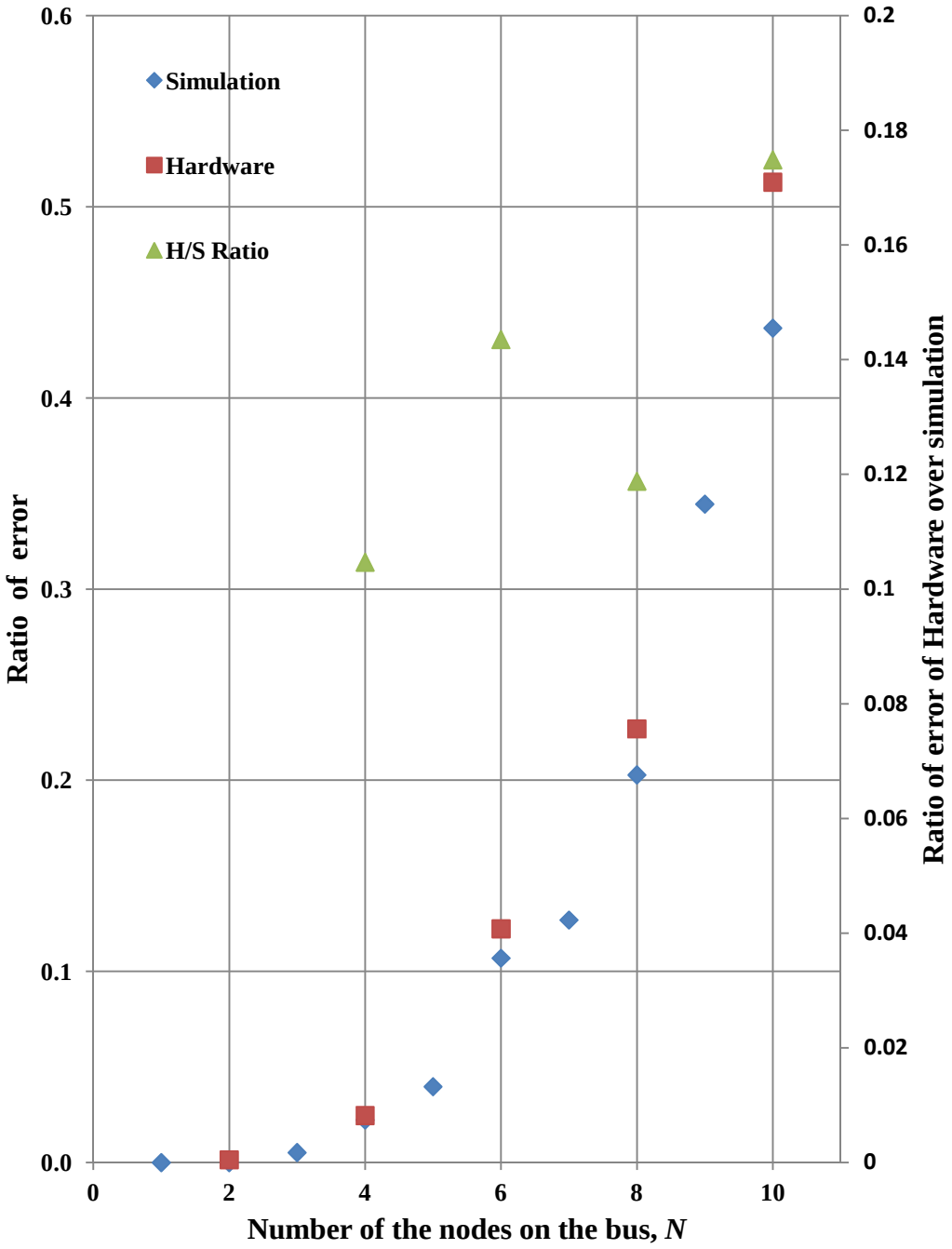


Figure 61 Comparison between the software simulated performance analysis, and experimental result of the hardware, for detection of not-transmitted Atoms.

Data Transmission

In our scheme, the transmission of the data from one node to another is done through transmissions of Atoms. How to sequence the Atoms to convey specific data could be different. The simplest scheme is transmissions of two Atoms with zero distance and the third one with a distance proportionate to the data, as Figure 53 shows. This scheme is called complex data 1. The simulation shows that this scheme follows the general trend of the less error for longer Atoms. Figure 62 shod the simulation results.

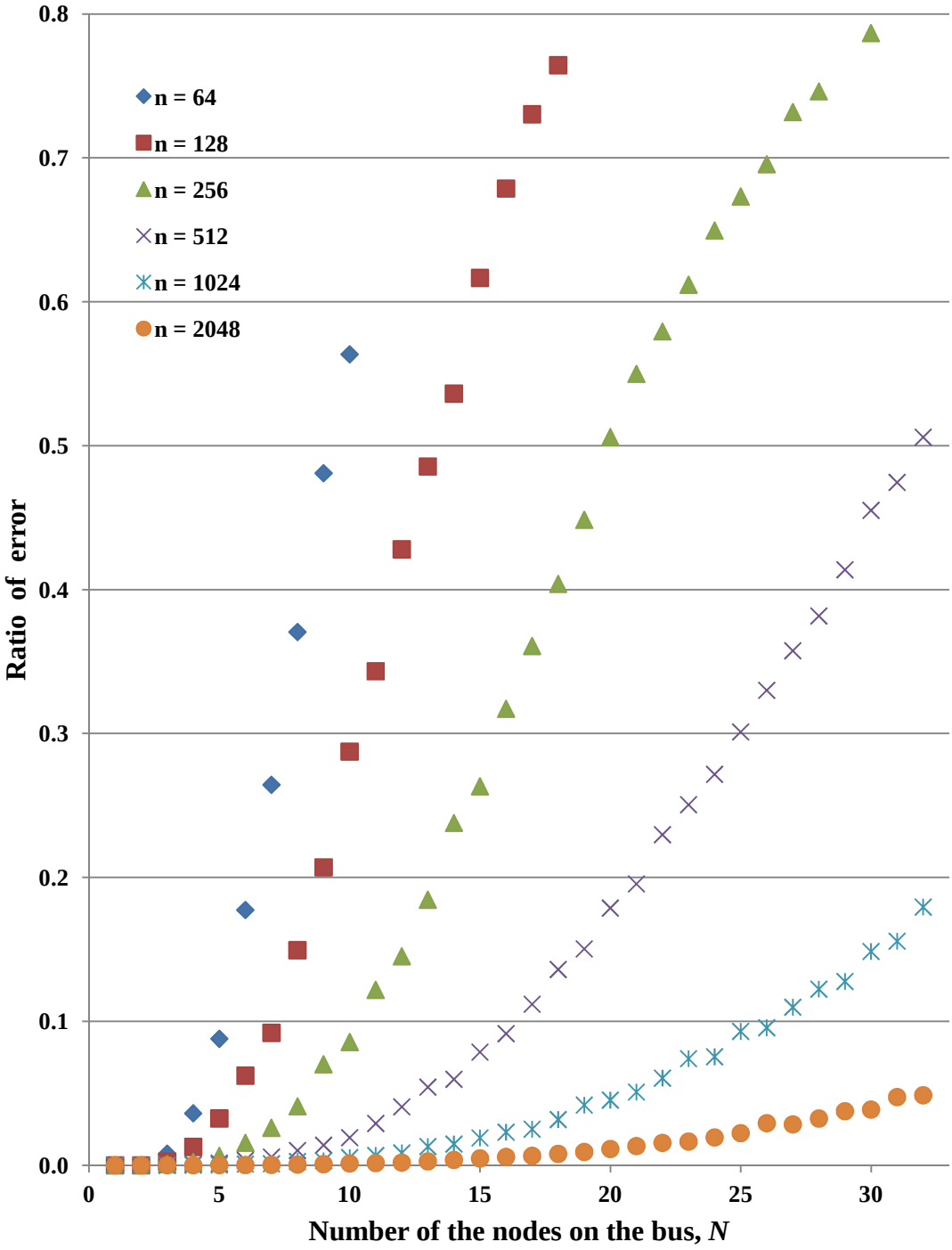


Figure 62 Software simulated performance analysis of the complex data 1 Atom arrangement versus the number of the nodes on bus, for different length of Atoms.

Figure 63 shows the comparison between the software simulation results and the

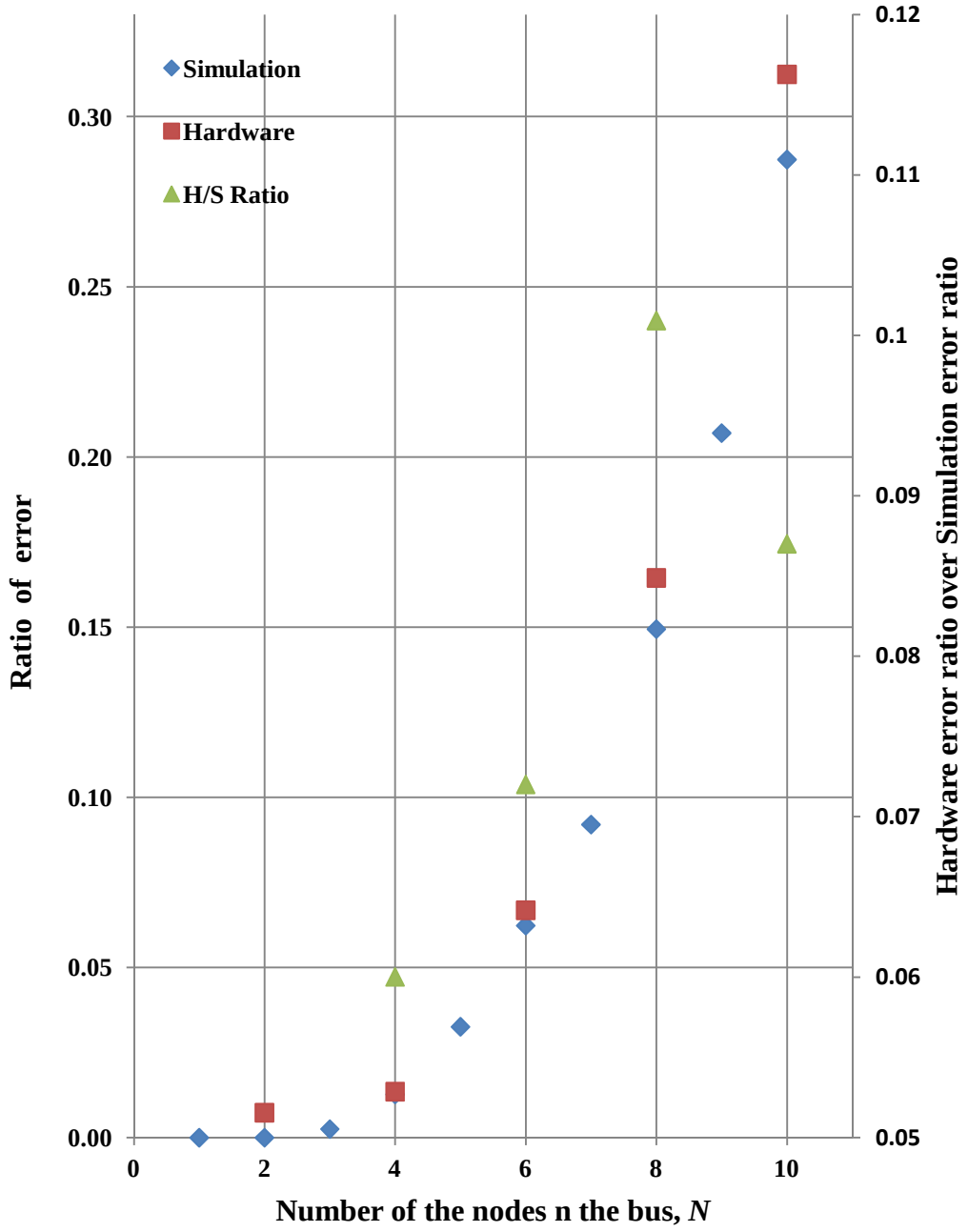


Figure 63 Comparison between the software simulated performance analysis, and experimental result of the hardware, for complex data 1 arrangement.

experimental data gathered from the hardware implementation results complex data 1 for the Atoms of length 128.

As the graph shows, the hardware implementation behaves in the same trend that the simulation results show. Of course there are some discrepancies between the results from the hardware implementation and the software simulation and the hardware implementation, which accounts for the imperfections of the hardware implementation. As the graph suggests, the difference between the four nodes out of the five measured points ranges from 5% to 10%.

The general scheme to reduce the detection errors, as explained in chapter 3, is through using the multiplicative factor for the linear transformation of the time distancing. The following simulation results, figure 64, shows the effects of such multiplicative factors for different length of the Atoms and different number of the nodes of complex data 1. As the graph shows, error rate has an inverse relation to the multiplicative factor.

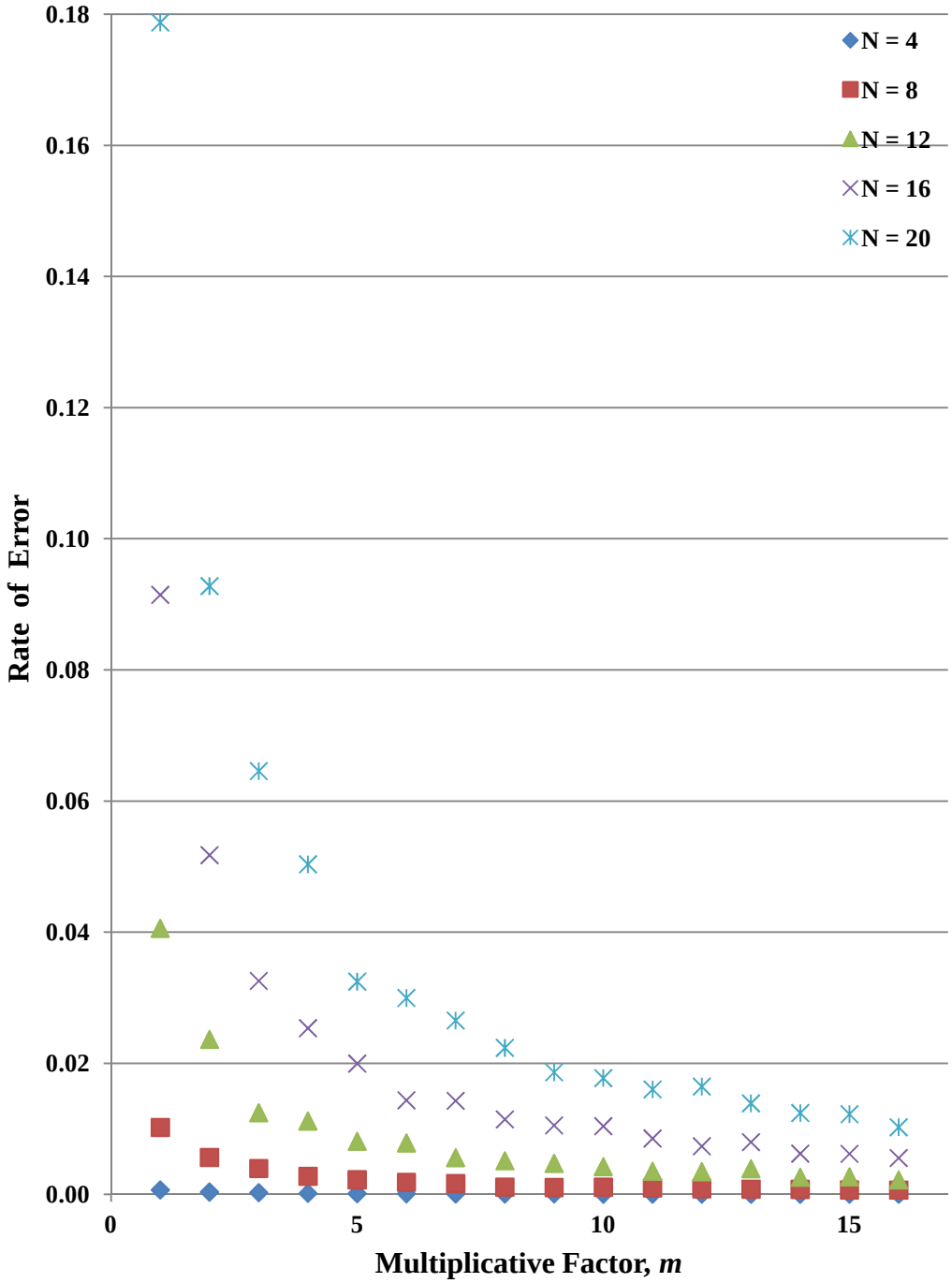


Figure 64 Effect of the multiplicative factor on the rate of error for different length of Atoms with complex data 1 arrangement.

Having the arrangement of the Atom similar to the complex data 1 is one of the simplest arrangements possible. In order to go one step further, the arrangement named complex data 2 is designed and simulated. Every transmission in a complex data 2 is consisted of six Atoms, and a check-sum. Similar to complex data 1, the complex data 2 starts with two Atoms, and follows with the time distanced third Atom, which encapsulate the linear transformation of data 1. In addition to that, another, a fourth Atom follows the third one witch encapsulates the linear transformation of another data, named data 2. Finally, two zero distanced Atom follow the forth Atom with the distance that corresponds to the modulo 128 summation of data1 and data 2. Figure 65 shows the complex data 2 arrangements.

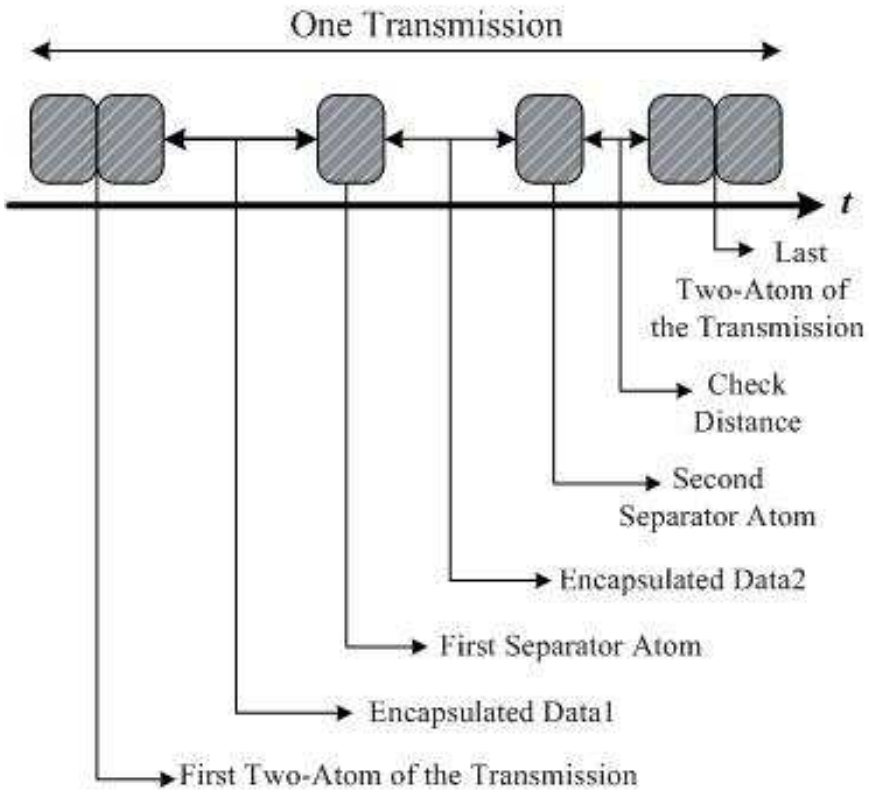


Figure 65 Arrangement of Atoms in complex data 2 format, in which every packet consisted of six Atoms and to k bit data.

Simulation shows that such an arrangement decreases the rate of detection error significantly. However, there is a price to pay. Complexity of the detection process which is $O(n!)$ where n is number of the detected Atoms between the first and last two-Atoms. In our simulation, the average value of n is fairly low especially for large multiplicative factor; therefore, growth rate is manageable. Figure 6 shows the simulation results for complex data 2, versus number of the nodes on the bus, for different length of Atoms.

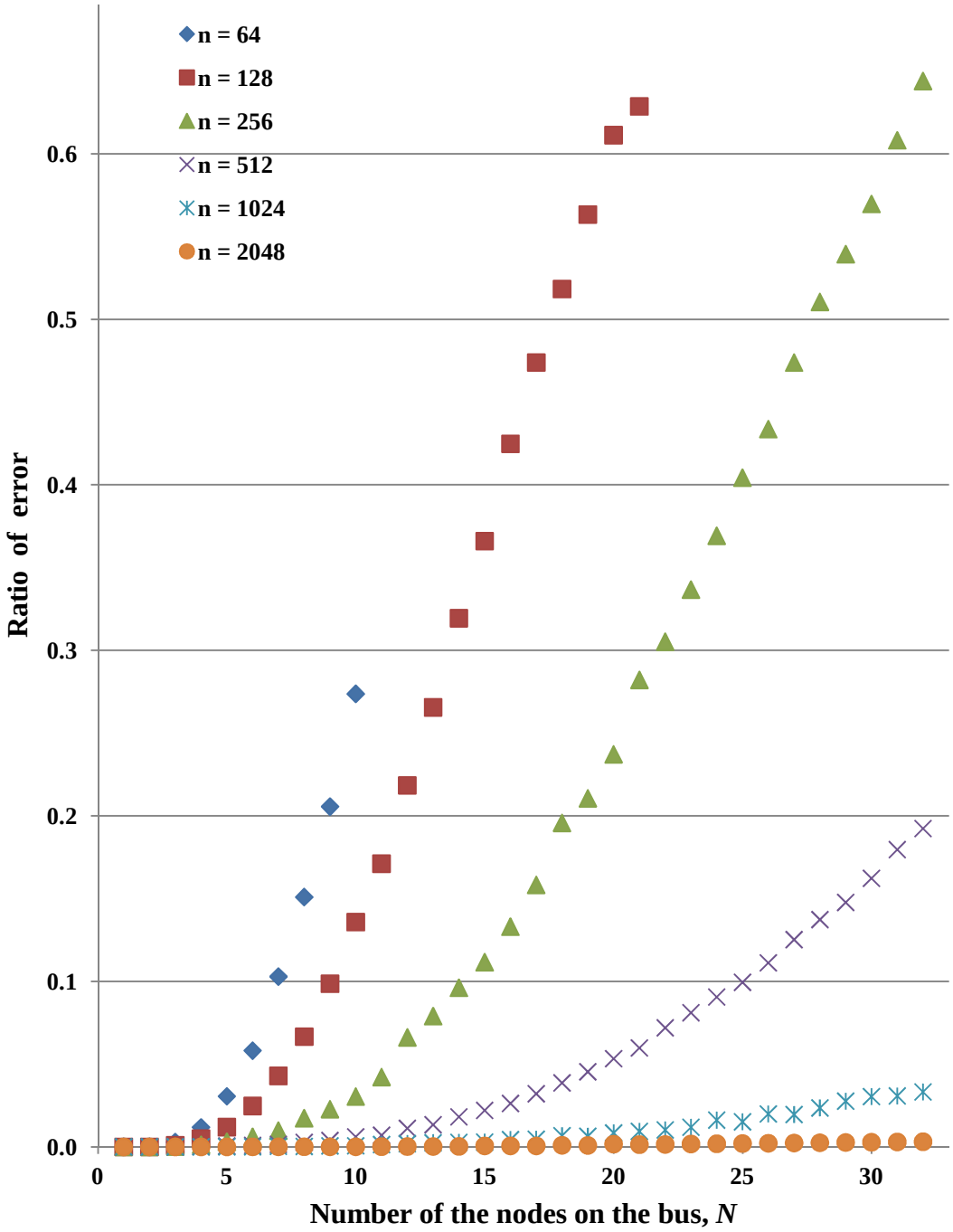


Figure 66 Software simulated performance analysis for complex data 2 arrangement versus the number of the nodes on the bus, for different length Atoms.

CHAPTER 7

THE WIRED-FDMA SCHEME

Introduction

In a Mesh-Bus setting, the nodes on the bus communicate their data through their Bus-Modulators. Such Bus-Modulators, which act as bidirectional translators between the nodes and the NOR bus, use different multiple access schemes to provide shared access to the bus. Wired-CDMA, as one of such multiple access schemes, was introduced and studied in previous chapters. In this chapter we look at a different multiple access scheme, namely *Wired-FDMA.* As the name suggest, it uses frequency rather than codes as the basis for medium access. Similar to Wired-CDMA, Wired-FDMA uses the NOR bus and it also has the same underlying concept of transmission based on the unary massages i.e., the Atoms, but with a completely different character set for an Atom. In the following section, we will introduce the Wired-FDMA Atoms, and the other aspects of this new physical layer scheme [49].

Wired-FDMA Signaling

Similar to the traditional FDMA [35], in Wired-FDMA, the different nodes use different frequency bands to communicate over the bus. In a Wired-FDMA scheme, every node S_i $(i \epsilon \{1, 2, \dots, m\})$ on the bus has a unique frequency F_i. When N_i needs to transmit a message to N_j, it modulates the message with the square wave at F_j and NORs the resulting modulated sequence onto the bus. For example, if the node N_1 transmits a message to node N_2, it codes the message by the unique frequency of node N_2. When node N_2 responds, it codes its response with node N_1's frequency.

Figure 67 depicts the bus, pulled high in the rest state, when two different and concurrent messages from N_1 and N_2 are NORed onto the bus.

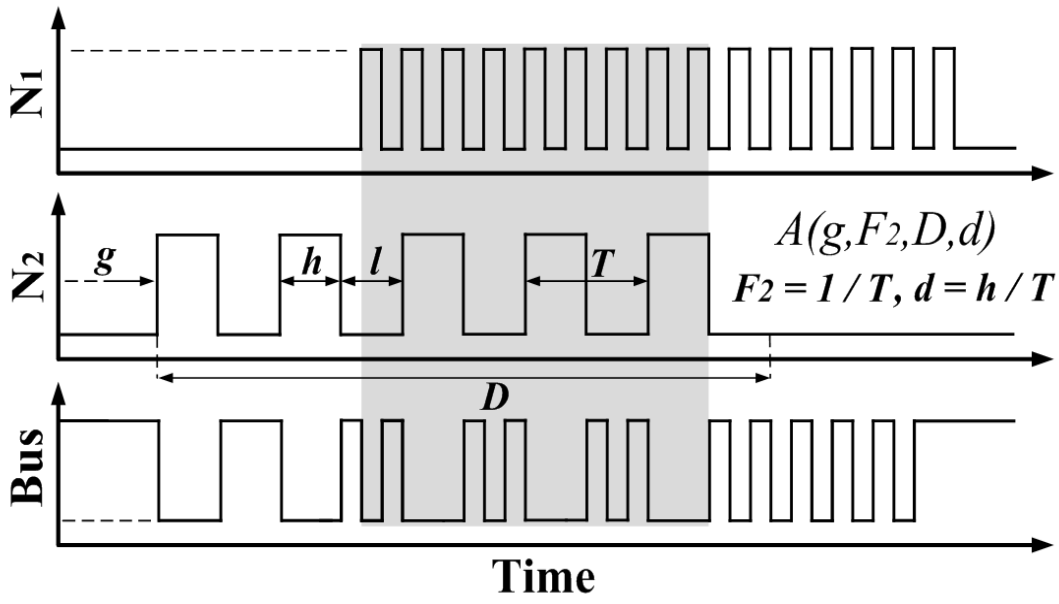


Figure 67 The Wired-FDMA based Atoms, and their interaction with the NOR-bus.

Transmission

It is useful to introduce the concept of an *Atom* for Wired-FDMA. In Wired-FDMA, message bits are coded through sets of unary Atoms. A typical Atom has the form of $A(g, F_i, D, d)$, where g is the guard time (time latency that tells the node how long it must wait after the end of last Atom it has already sent out, to start transmitting the new one), F_i is the frequency of the destination of the message, D is the time duration which the Atom will be transmitted, and d is the duty cycle of the modulating square wave.

The transmission phase of Wired-FDMA signaling consists simply of performing a logical NOR operation between the Atoms which have been generated by the nodes, and the current state of the bus.

Detection

Even though Wired-FDMA transmission has similarities with traditional wireless/radio FM, the same detection schemes are not useful. In traditional FDMA, one assumes the superposition of waves with different frequencies, and for detection one can apply techniques such as extracting the I/Q components, linear filtering, etc. as Figure 68 shows [44]. We explored such techniques, but because the NOR-bus is nonlinear, they did not perform well.

Below, one possible detection scheme is investigated which is based on the following key observation: if any single node transmits logic 1(high), the bus goes to the low state. Therefore, the bus can be in the high state if and only if all the nodes on the bus are transmitting logic 0s (low).

In the other words, as long as one node drives the bus to the low state, all possible transmissions by other nodes are being masked, because as long as the bus is

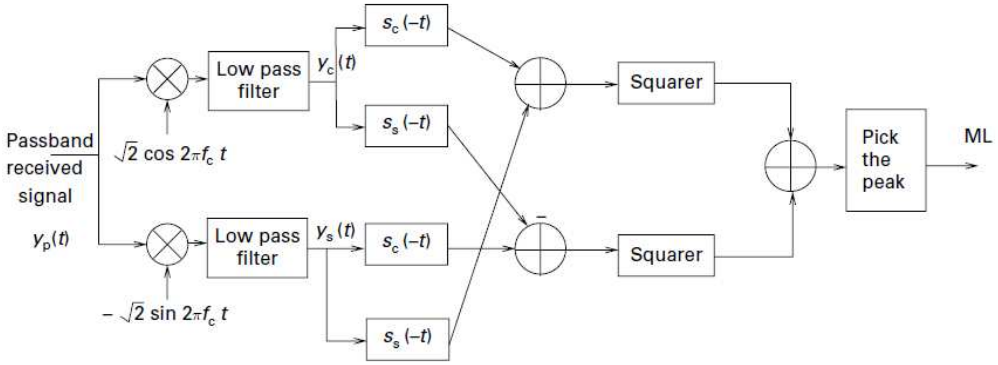


Figure 68 Detection process in the traditional DFMA schemes [44].

low, their transmission (high or low) do not change the state of the bus. Conversely, high states of the bus imply no transmission. This can be used to infer that particular Atoms were *not* transmitted, since if they were, the bus would be pulled low.

Figure 69 shows some detection examples. In the figure, E_j represents the signal on the bus node destined for node N_j, when there are no other concurrent transmissions on the bus. In this Figure S_1, S_2, and S_3 are examples of the bus when there are other concurrent transmissions. The following rule allows node N_j to eliminate transmission not destined for it:

For node N_j, a signal is a candidate for detection if it is never high when E_j is low, and at least sometimes high when E_j is high.

S_1 and S_3 are detection candidates for N_j because they follow the *rule*—they are

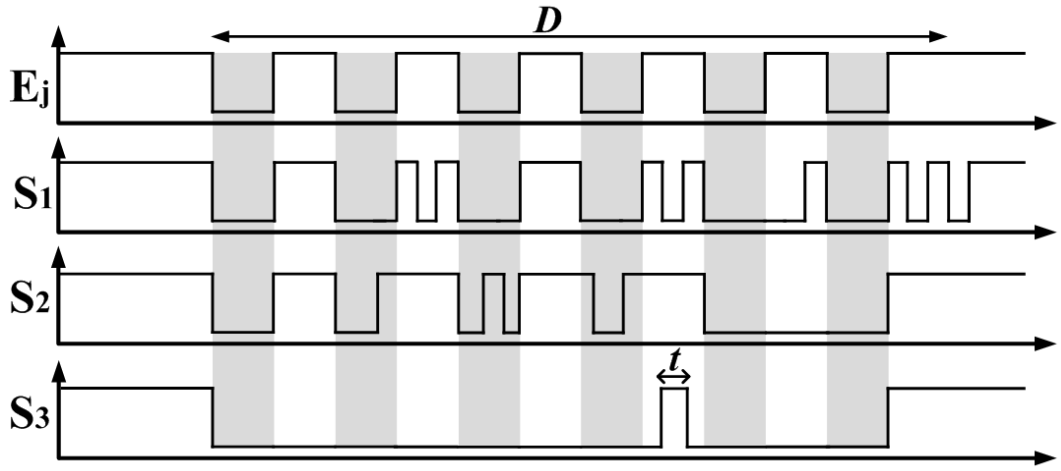


Figure 69 Detection phase of the Wired-FDMA. N_j is the expected *Atom* at the receiver side when there is no other transmission on the bus. S_1, S_2 and S_3 are three sample signals which the S_1 and S_3 are candidates, but S_2 is not, because in contrast with other two, it has high value in non-allowed (gray) areas.

always low when E_j is low, and at least sometimes high when E_j is high. However, S_2 is not a detection candidate for N_j. Clearly, it is high at some points when E_j is low (the gray areas), and does not follow the *rule*. Returning to S_1 and S_3 —since they are low at times when E_j is high, it follows that there are other concurrent transmissions on the bus.

Because the *rule* does enforce any hard restriction on the value of the signal during the high periods of E_j, there are many possible different detection candidates which follow the *rule*. To discriminate between the various candidates, the detection scheme uses a *detection ratio* (D_{Ratio}) defined throug the following formula, and has value $D_{Ratio} \epsilon\ (0, 1]$. The D_{Ratio} is simply the ratio of the sum of the high periods of the candidate signal, to the sum of the high periods of E_j.

$$D_{Ratio} = \frac{C}{h\ F_j\ D}$$

S_3 in Figure 69 could be (a) an Atom transmitted toward N_j, but highly deformed by other concurrent transmissions, or (b) an undeformed Atom, and a false positive. However, S_3 is high only for a short time t during times when E_j is high, and consequently has a low D_{Ratio}. Other the other hand, S_1, another candidate, is high many times when E_j is high. Thus, it has a higher D_{Ratio} and is a better match than S_3. By optimizing the threshold for neglecting signals with low D_{Ratio}, one can minimize false positives.

The more similar the signal on the bus to is to E_j, the closer D_{Ratio} is to 1, and the higher the probability that the detection candidate is a transmission destined to N_j. The cases in which D_{Ratio} equals one implies that there is perfect detection i.e., no other concurrent transmission.

Figure 70 suggest a hardware implementation model for Wired-FDMA scheme.

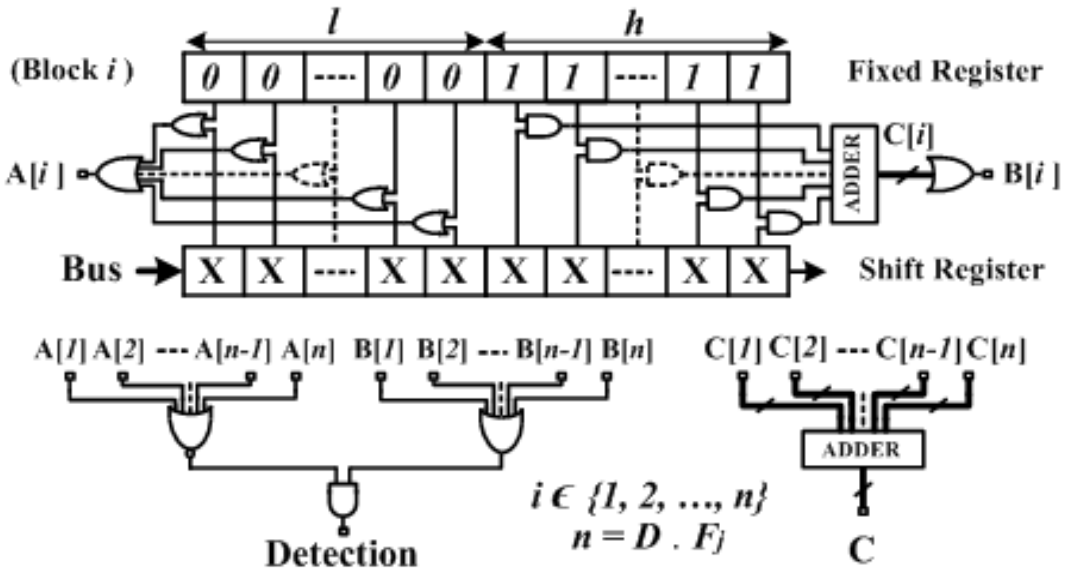


Figure 70 A hardware scheme for the detection process of the Wired-FDMA.

Simulation and Analysis

To evaluate and optimize some of the Atom parameters, we created a flexible simulation engine. It provides the ability to explore the effects of parameters such as g, D, d and ΔF (i.e., frequency separation between F_i s) in a quantitative way. Very briefly, the simulation engine is implemented in Java and fully exploits the object-oriented nature of this language. Each node on the Wired-FDMA bus is an object that has capacity of transmission and detection independently and has access to the bus, which is an object itself.

Figure 71 shows the results of a simulation designed to determine error rate versus the number of the nodes concurrently communicating. For this simulation, $F \in [9000, 436500]$ and the minimum ΔF is 4.5 kHz. D and d are constant equals to

1,000 μs, and 50% respectively. In order to simulate the concurrent transmission, the transmission of each node overlaps with the others' randomly from 0.1% to 100%. And finally, number of the repetition is 10^6 times.

In each of the 10^6 experiments, in order to consider a transmission correct or incorrect, two criterions have to be met. First, if an Atom has been transmitted, it must be detected, and second, which gets more important especially when the number of the nodes increases, if a specific Atom has not been transmitted, it must not be detected, in other words, no false positive detection is allowed.

As Figure 71 shows, this specific setting allows 7 nodes to concurrently transmit with absolutely no error, but as the number of the nodes increases above 8, some errors are introduced to the transmissions, and as it grows over 12 nodes, the bus starts getting saturated and almost no error free transmission could occur.

Note that the presented results (less than 10 concurrent communications with

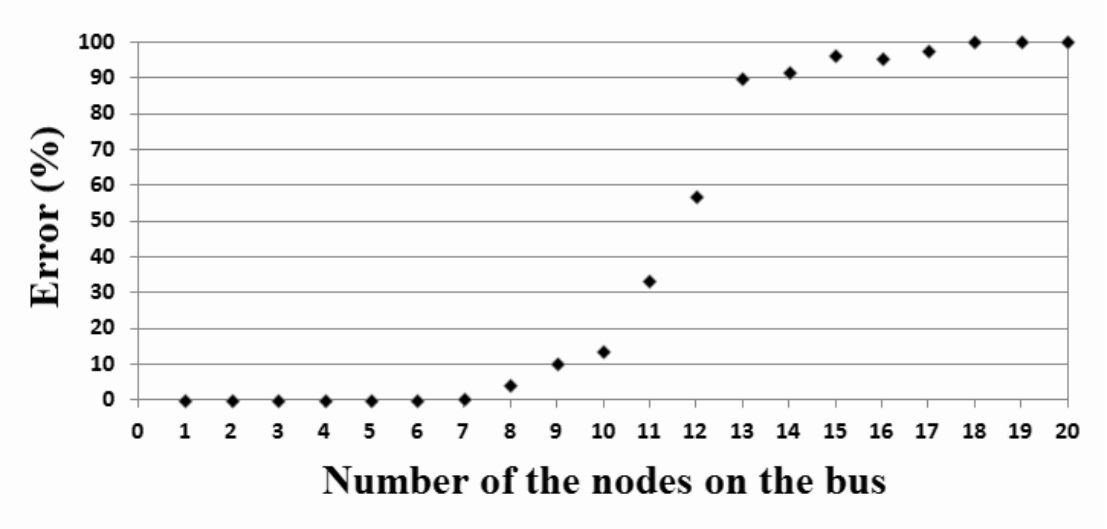


Figure 71 The simulation result for Wired-FDMA.

small error rate) are just for the proof of concept and do not present the maximum capacity of the Wired-FDMA scheme. With some other settings, through optimizing the parameters of the Atoms, as well as employing some error control coding (both are subjects of the future work), the scheme could allow higher number concurrent communication on the bus within an acceptable error margin.

CHAPTER 8

THE WIRELESS BUS

Introduction

The Mesh-Bus setting, which is a single wire multi-access communication protocol, may also be used for any type of communication medium with the same characteristics, for example, in a wireless environment, or as we previously discussed, in single wavelength fiber optical communication systems. In this chapter, we will adapt the Mesh-Bus setting for a wireless communication network, in which all the transceiver nodes operate at the same frequency [52].

Wireless Bus, as a modified Mesh-Bus Scheme

The idea of using the Mesh-Bus, as presented in the last chapters, is tied to the employment some intermediary devices, named Bus-Modulators which through some high level standard protocols communicated with the sensors. At the other end, they employ some non-TDMA schemes such as CDMA or FDMA, they connect to a NOR bus. In a big picture, such a single wire bus could be seen as the wireless medium, in which all the nodes that operate at the same frequency, share it. Such a sharing mechanism in general is quite similar to the Mesh-Bus setting. Figure 72 shows such wireless shared access as well as the Mesh-Bus.

In the Mesh-Bus setting using the NOR bus, when there is not any communication through the bus, the bus is at logic level ‘1’, or the High state. As soon as one of the nodes transmits logic ‘1’, the bus goes to logic ‘0’, or the Low state, and remains Low until none of the nodes on the bus transmits ‘1’ any more. The signal on the bus is simply the result of NOR-ing the outputs of the sensors.

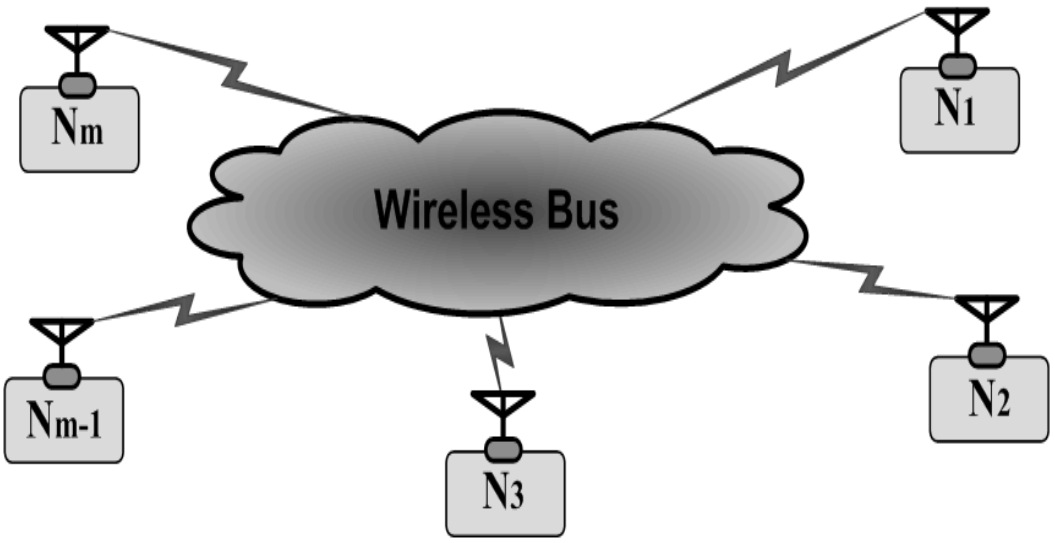


Figure 72 The WirelessBus. This wireless communication is quite similar to the Mesh-Bus setting.

On the other hand, in the wireless communication systems, the radios transceivers which use unipolar amplitude modulation [45] (i.e., On Off Keying - OOK), could be considered as the radio equivalents of Bus-Modulators of the nodes in the Mesh-Bus. High state corresponds to presence of the carrier frequency of the radios, and Low state corresponds to absence of it. Parts (a) and (b) of Figure 6.2 show such correspondences in a sample radio with an OOK modulation.

According to this analogy, when none of the sensors are transmitting, there is no carrier frequency on the air, and when one of the nodes transmits, there will be a carrier frequency present. However, unlike the NOR bus which is a nonlinear bus, and superposition does not hold (e.g., when ‘1’ NOR ‘1’ equals ‘1’ in contrast to a linear system which ‘1’ + ‘1’ equals ‘2’), the wireless transmission medium is almost a linear system. When there are n ($n \geq 2$) carriers on the air, they are either constructively or destructively summed together depending on the phase difference of the carriers.

Therefore, the result of such a summation at any time instant is random variable uniformly distributed over $[-nA, +nA]$.

In order to be able to use Mesh-Bus physical layer protocols such as Wired-CDMA or Wired-FDMA, some mechanisms are needed to guarantee that there will not be any interference, neither destructive nor constructive, when more than one node transmits at any moment. If such interferences take place, there will some additional errors introduced to the system because of them. One possible solution to this problem is the process of *active listening/virtual transmission*. This solution does not guarantee the prevention of such interferences, but it could be used in order to dramatically minimize the chance of the possible interferences.

Active Listening operates in the following way: every node constantly listens to the medium; when it has '1' to transmit, if it does not sense any carrier on the air, it transmits for the duration of time it desires; basically, it is the owner of the channel. However, if it senses the carrier at the time when it wants to transmit '1', it *virtually* transmits. It means as long as it senses the carrier, it does not transmit but it could be assumed that it is transmitting. If by the end of the duration of its own transmission the other carrier still is present, the job is done; the intended signal had been present for the required period of time through as the result of the transmission of some other nodes.

Figure 73 shows the process of active listening/virtual transmission. The waveform shown in (b) is the analog signal correspondent of the digital data, (a), which is required to be transmitted. In part (c), all the signals due to the transmissions of other nodes, which could be possible source of interference, is shown, and (d) shows the actual transmission of the node. As it could be seen in (c), at time t_2 there is no signal on the air, so as the owner of the medium, it physically transmits for the whole duration it requires, in this case, until t_3. At t_4, when it requires transmitting again, through the process of active listening it already knows that there exist other transmissions; therefore, it virtually transmits until it does not sense any other signal, the moment it recognizes the absence of

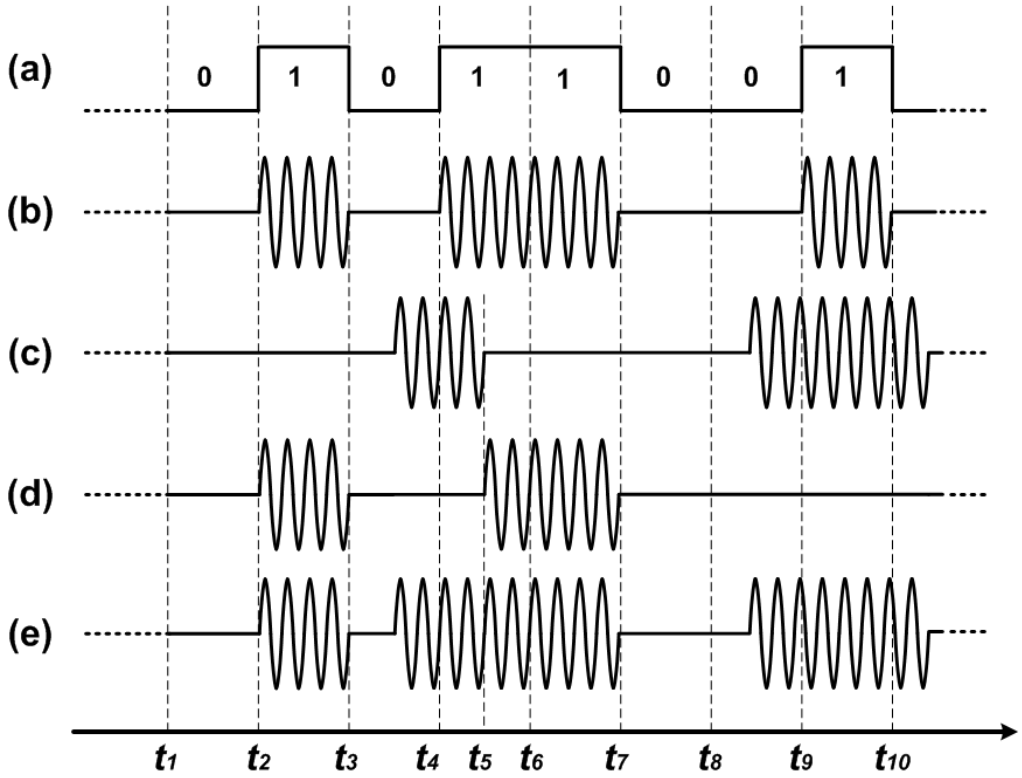


Figure 73 The process of active listening/ virtual transmission. Signals (a) and (b) show the data which is required to be transmitted in two forms of analog waveform and digital string. (c) shows the pre existed signal on the medium, and (d) is what the node physically transmits, base on the condition of the medium. Finally, (e) shows the medium after the transmission of (b).

the previously existed signal, t_5 in (c), it starts and keeps transmitting until it has something to transmit, here, until t_7. And finally, at t_9 when requires again to transmit, since already another transmission has been going on, it virtually transmits instead of actual physical transmission, but this time it does not requires to switch from virtual transmission to physical one, because the existing signal lasts more that the duration it requires to transmit, so the whole transmission remains virtual.

There are some complications with the active listening/virtual transmission scheme, such as the cases in which more than one node is in the state of virtual transmission and they all together start switching to physical transmission. There will be interference in such situations which will be addressed in future work. Another possible complication arises when the switching time from receive mode to transmission mode of the radios are considerable. Therefore, this delay must be considered as an important parameter for specifying the minimum bit duration. Figure 74 shows such a switching delay for the LINX transceiver modules [46].

Hardware implementation

In order to provide a real world example, and investigate the probable software and hardware complications in the proposed scheme, a hardware platform, which is consisted of two subsystems (a radio transceiver and a general purpose microcontroller

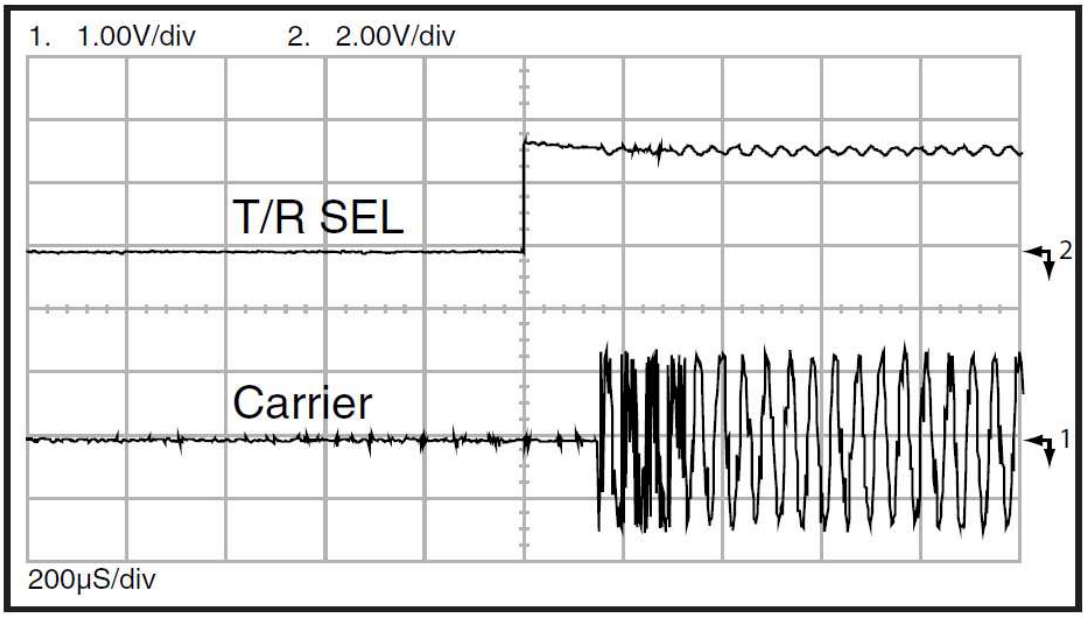


Figure 74 Delay of switching from Rx mode to Tx mode. As the diagram shows, the delay for switching between TX and RX mode is about 150 µs.

board) is designed and produced. Following subsections provide some details on these subsystems.

Radio Transceiver Module

A very simple single frequency LINX TRM-433-LT radio transceiver module has been chosen for hardware implementation. As the name suggests, it operates on 433.92 MHz and employs OOK modulation scheme. The average link budget of that 121.2 dBm (average maximum output power: +9.2 dBm, average receiver sensitivity: -112 dBm) makes it proper candidate for many WSN application. Figure 6.4 demonstrates this radio module and its internal circuitry diagram, as well as the functionality of every pin of it.

Since it is just a module without any proper antenna connector and other required peripherals, a breakout board is designed for it to extracts the pins and provides required connectors. Also, for experimental reasons, four LEDs show the activity of power, transmit/ receive, and data lines. Top portion of the Figure 75 shows this top and bottom of this breakout board.

Microcontroller Board

Based on the *ATmega 88/168/328* family of the Atmel AVR 8-bit microcontrollers, a general purpose microcontroller board is designed and implemented. This board is designed such that it houses the radio transceiver module, and provides various bitwise, parallel, and serial input/output connections to it. The microcontroller board has provided us a good amount of flexibility to experiment different schemes, and the ability to incorporate new ideas and enhancement to the existing design. The microcontroller operates at the clock frequency of 20 MHz to enable us process the data as fast as possible. It also provides USART communication for any interaction between the board and a computer. Finally, it is designed in a way that both radio transceiver module and microcontroller are able to operate independently in case they are needed to.

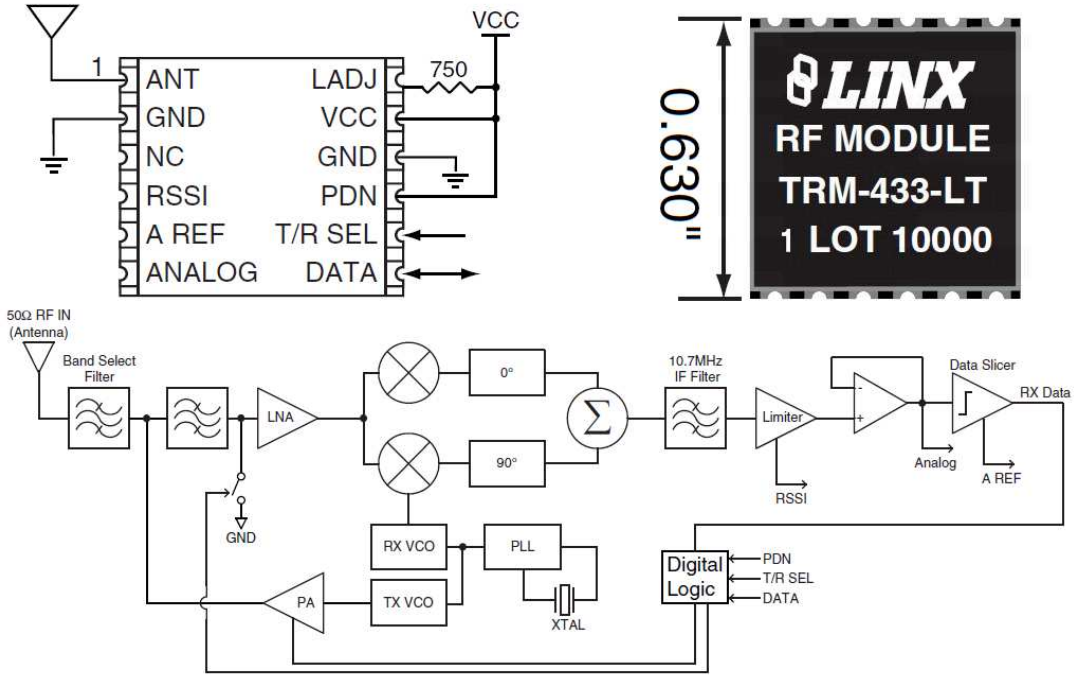


Figure 75 LINX TRM-433-LT radio transceiver module.

Figure 76 (bottom) shows the both sides of the microcontroller board. Figure 77 shows the schematic of the microcontroller board.

Experiment Setting

An scenario for testing the Wireless Bus could be as follows: five radio transmitters, which each one of them is assigned a unique OOC codeword from the , as table 6.1 shows, and one receiver, which has all the five OOC codewords of the five transmitters.

The transmitters are programmed in a way that they keep transmitting their very same OOC codewords continuously every 1280 ms. Since the lengths of the codewords are 256 *chip*s, every chip will be on air for 5 ms. On the other hand, the receiver, which is

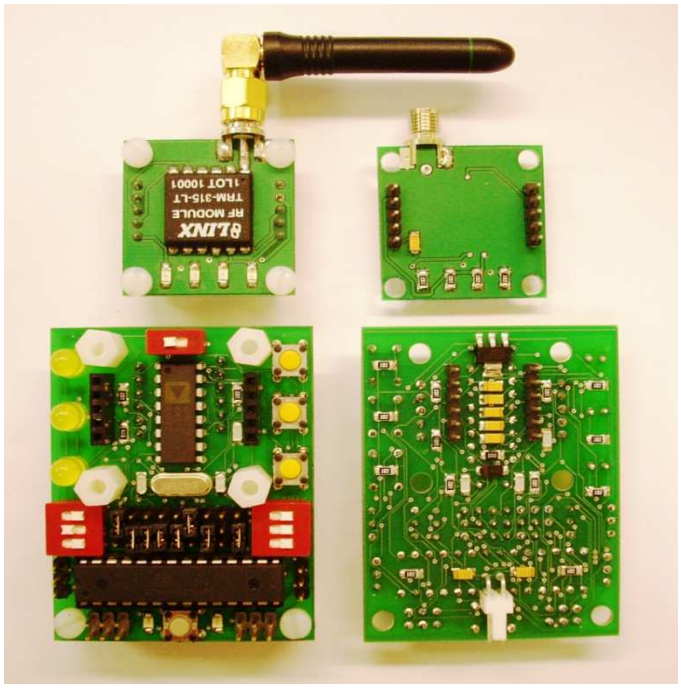


Figure 77 Radio breakout and the microcontroller boards.

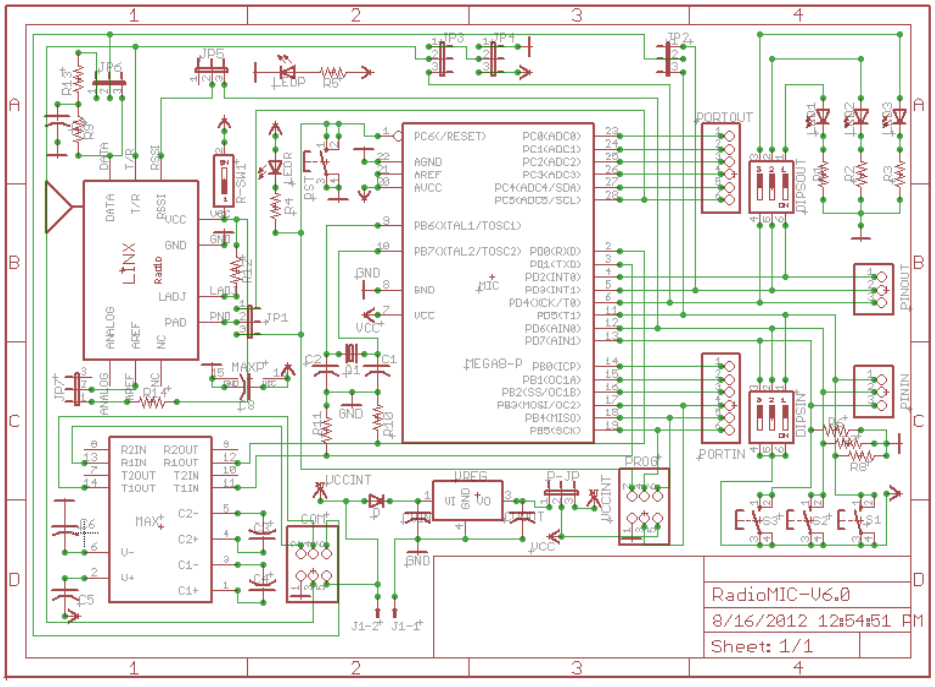


Figure 76 The schematic of the Radio-microcontroller board.

Table 5 Five codewords of the (256,3,1)-OOC.

Node	Codeword
1	(0, 15, 85)
2	(0, 25, 125)
3	(0, 35, 115)
4	(0, 45, 105)

in the radio range of all the transmitters, receives the mixed version of all the transmitted signals, and based on the code it uses (selection of one code out of the five is through the DIP switch which is connected to the input pins of the microcontroller), at any time it is able to receive just one of the transmissions. Figure 78 shows such an experiment setting.

In every transmitter a counter could count the number of the transmissions, and in the receiver, one counter for every transmitter could keep track of the detections for that transmitter. At the end of transmitting a known number of codes by one transmitter, in the presence of other transmission of other transmitters, number of the detected codewords for that specific transmitter could be used as a measure for error rate.

In case the radios perform frame synchronization, it is expected that there will be no error in such scheme of transmission, mainly because the codes are non-overlapping. Therefore, there is no way that the codes detected mistakenly. But in reality, even in this case, there still exist some error; however low, which mainly are because of environmental noise, and could be addressed future work.

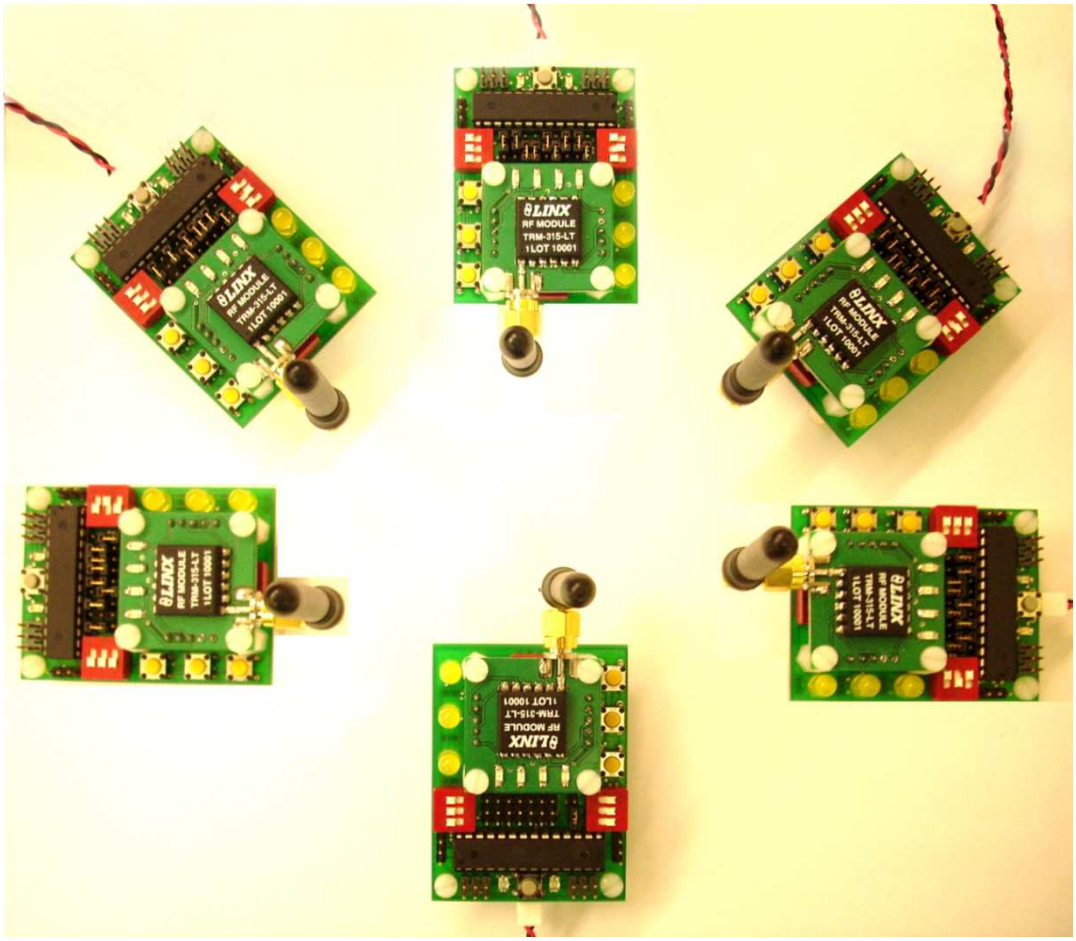


Figure 78 A scenario for testng the WirelessBus scheme consisted of six radios, five transmitters and a receiver.

REFERENCES

[1] *1-Wire Tutorial*. Available: http://www.maximintegrated.com/products/1-wire/flash/overview/index.cfm

[2] M. S. Gurdeep S. Hura and M. Singhal, *Data Computer Communications: Networking and Internetworking*: CRC PressINC, 2001.

[3] K. S. Zigangirov, *Theory of Code Division Multiple Access Communication*: Wiley, 2004.

[4] D. Barrett and D. B. T. King, *Computer Networking Illuminated*: Jones & Bartlett Learning, 2005.

[5] F. L. A. Marshall C. Yovits, *ADVANCES IN COMPUTERS*: Elsevier Science, 1977.

[6] V. Vij, *Wireless Communication*: Laxmi Publications Pvt Ltd, 2010.

[7] V. K. Garg, *Wireless Communications and Networking*: Elsevier Morgan Kaufmann, 2007.

[8] M. A. Abu-Rgheff, *Introduction to CDMA Wireless Communications*: Elsevier Science, 2007.

[9] J. Minkoff, "The Role of AM-to-PM Conversion in Memoryless Nonlinear Systems," *Communications, IEEE Transactions on,* vol. 33, pp. 139-144, 1985.

[10] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*: Elsevier Science, 2003.

[11] N. Benvenuto and M. Zorzi, *Principles of Communications Networks and Systems*: John Wiley & Sons, 2011.

[12] 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.

[13] H. Yin and D. J. Richardson, *Optical Code Division Multiple Access Communication Networks: Theory and Applications*: Springer, 2009.

[14] P. Horowitz and W. Hill, *The Art of Electronics*: Cambridge University Press, 1989.

[15] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*: Cambridge University Press, 2009.

[16] S. M. John B. Anderson and S. Mohan, *Source and Channel Coding: An Algorithmic Approach*: Springer-Verlag GmbH, 1991.

[17] R. Zurawski, *The Industrial Communication Technology Handbook*: CRC PressINC, 2005.

[18] H. Sasaoka, *Mobile Communications*: Ohmsha, 2000.

[19] A. F. Mohammed, "Near-far problem in direct-sequence code-division multiple-access systems," in *Mobile and Personal Communications, 1993., Seventh IEE European Conference on*, 1993, pp. 151-154.

[20] A. Das, *Digital Communication: Principles and System Modelling*: Springer, 2010.

[21] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*: Cambridge University Press, 2005.

[22] N. Mathivanan, *Microprocessors Pc Hardware And Interfacing*: Prentice-Hall Of India Pvt. Limited, 2004.

[23] J. W. Valvano, *Embedded Microcomputer Systems: Real Time Interfacing*: Nelson Education Limited, 2011.

[24] *Telecommunications Industry Association*. Available: http://www.tiaonline.org/about/

[25] *Electronic Industries Alliance*. Available: http://standards.ec-central.org/kwspub/home/

[26] J. L. Axelson, *Serial Port Complete: Programming and Circuits for RS-232 and RS-485*: Lakview Research, 1998.

[27] J. R. Freer, *Computer Communications And Networks, 2nd Edition*: Taylor & Francis Group, 1996.

[28] Y. Wu and Q. Luo, *High Performance Networking, Computing, Communication Systems, and Mathematical Foundations: International Conferences, ICHCC 2009-ICTMF 2009, Sanya, Hainan Island, China, December 13-14, 2009. Proceedings*: Springer, 2010.

[29] P. M. T. S. Bernhard Linke. (2008). *Overview of 1-Wire® Technology and Its Use*. Available: http://www.maximintegrated.com/app-notes/index.mvp/id/1796

[30] H. W. Huang, *The HCS12 / 9S12: An Introduction to Software and Hardware Interfacing*: Delmar Cengage Learning, 2009.

[31] I. P. Kovalev, *Spatial Division Multiple Access for Multipath Wireless Channels: Limiting Characteristics and Stochastic Models*: Springer-Verlag GmbH, 2004.

[32] H. Schulze and C. Lueders, *Theory and Applications of OFDM and CDMA: Wideband Wireless Communications*: Wiley, 2005.

[33] V. P. Ipatov, *Spread Spectrum and CDMA: Principles and Applications*: John Wiley & Sons, 2005.

[34] D. Torrieri, *Principles of Spread-Spectrum Communication Systems, Second Edition*: Springer, 2011.

[35] J. G. Proakis and M. Salehi, *Fundamentals of communication systems*. Upper Saddle River, N.J.: Pearson Prentice Hall, 2005.

[36] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*: Prentice Hall, 2004.

[37] S. Johnson, "A new upper bound for error-correcting codes," *Information Theory, IRE Transactions on*, vol. 8, pp. 203-207, 1962.

[38] A. Xiaoqiang and Q. Kun, "Construction for optimal optical orthogonal codes," in *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on*, 2002, pp. 96-100 vol.1.

[39] *ATmega48A/PA/88A/PA/168A/PA/328/P Summary*. Available: http://www.atmel.com/Images/8271S.pdf

[40] J. G. Proakis and M. Salehi, *Digital communications*, 5th ed. Boston: McGraw-Hill Higher Education, 2008.

[41] I. Glover and P. M. Grant, *Digital communications*, 3rd ed. Harlow, England ; New York: Prentice Hall, 2010.

[42] P. D. Roberto Aiello and P. D. Anuj Batra, *Ultra Wideband Systems: Technologies and Applications*: Elsevier Science, 2006.

[43] J. K. Cavers, *Mobile Channel Characteristics*: Kluwer Academic Pub, 2000.

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

[45] J. G. Proakis and M. Salehi, *Fundamentals of communication systems*: Pearson Prentice Hall, 2005.

[46] *LT Series RF Transceiver Module*. Available: https://www.linxtechnologies.com/resources/data-guides/trm-xxx-lt.pdf

[47] *In System Programming.* Available: http://www.atmel.com/images/doc0943.pdf

[48] *AVRISP mkII*. Available: http://www.atmel.com/tools/AVRISPMKII.aspx?tab=overview

[49] Rezaei, H.F.; Kruger, A.; , "Wired-FM, a novel distributed digital single-wire field bus," Electro/Information Technology (EIT), 2012 IEEE International Conference on , vol., no., pp.1-4, 6-8 May 2012.

[50] Peiffer, B.; Kruger, A.; , "Physical layer architecture for 1-wire sensor communication bus: Binary channel Code Division Multiple Access," Sensors Applications Symposium (SAS), 2011 IEEE , vol., no., pp.100-105, 22-24 Feb. 2011.

[51] Rezaei, H.F.; Kruger, A., "Low weight double layer coded CDMA as a novel physical layer for OneWire bus communication in sensor networks," Sensors Applications Symposium (SAS), 2013 IEEE , vol., no., pp.15,20, 19-21 Feb. 2013.

[52] Rezaei, H.F.; Kruger, A., "Wireless multichannel bus communication using CDMA coding for single frequency radios," Sensors Applications Symposium (SAS), 2013 IEEE , vol., no., pp.1,6, 19-21 Feb. 2013.