

Lightweight Hierarchical Error Control Codes for Multi-Bit Differential Channels

by

Jason D. Bakos

Ph.D. Proposal

University of Pittsburgh

2004

UNIVERSITY OF PITTSBURGH
FACULTY OF ARTS AND SCIENCES

This proposal was presented

by

Jason D. Bakos

It was defended on

May 27, 2004

Dissertation Committee:

Donald M. Chiarulli

Steven P. Levitan

Bruce R. Childers

Patchrawat Uthaisombut

Donald M. Chiarulli

Dissertation Director

Lightweight Hierarchical Error Control Codes for Multi-Bit Differential Channels

Jason D. Bakos

University of Pittsburgh, 2004

In this thesis we describe a new class of error control codes that is designed to be used over “Multi-Bit Differential Signaling (MBDS),” a new off-chip signaling technology. This new class of codes, called “Lightweight Hierarchical Error Control Codes (LHECC),” takes advantage of inherent structural properties of MBDS’s channel encoding mechanism, called “N choose M (nCm)” encoding. These properties are used to make LHECC codes extremely lightweight by offsetting much of the overhead cost normally required to add error control support to a channel. Lightweight Hierarchical Error Control Codes exploit both the inherent error detection characteristics and extra code words offered by nCm encoding.

High-performance signaling technology for chip-to-chip links is an increasingly important area of research in computer system design. The growing gap between on-chip and off-chip signaling speeds is creating a potential bottleneck and future system design. This phenomenon is caused by the inherent challenges in transmitting high-speed signals through off-chip signaling paths. These challenges stem from additional vulnerability to noise and a larger amount of conductive material associated with off-chip signaling paths relative to on-chip signaling paths.

For any given signaling technology there are various parameters that affect its maximum achievable signaling speed. Many of these parameters are influenced by the way in which data is encoded prior to transmission. In this work, we build an additional layer of data encoding on top of a newly developed high-performance off-chip signaling technology. This technology utilizes a base channel encoding mechanism to achieve high code density while offering noise rejection. The additional encoding further improves signal integrity and allows for higher total throughput.

In addition to developing LHECC codes, we present simulation experiments from a prototype MBDS system. These preliminary results demonstrate that the error correction and detection capabilities of LHECC codes are well suited to recover from the types of signaling errors that are likely to occur in an MBDS link. Signal errors are caused by various noise sources that are intrinsic to any signaling technology. The increased signal integrity that is gained from LHECC codes allow MBDS links to cope with noise and operate at higher speeds that can otherwise be achieved without an additional level of data encoding for error control.

TABLE OF CONTENTS

1. Introduction.....	1
2. Motivation.....	5
2.1. Challenges in Off-chip Signaling.....	5
2.2. Definitions of Signaling Metrics.....	7
2.2. Definitions of Signaling Metrics.....	8
2.3. Circuit-Level Data Encoding	9
3. Background	11
3.1. Existing Signaling Technologies: Single-ended and Differential.....	11
3.2. Multi-bit Differential Signaling	14
3.3. “N choose m” (nCm) Encoding and Support for Additional Coding Features.....	15
4. Previous Work.....	19
4.1. Overview	19
4.2. Error Control Coding	19
4.3. Survey of State-of-the-art Signaling Technology	25
4.4. Current Research in High-Performance Signaling Technology	28
4.5. Current Research in Non-electrical Signaling Technology	32
5. Problem Statement	35
6. Approach.....	39
6.1. Overview	39
6.2. Inherent Error Detection Capability of nCm Code Sets	39
6.3. Excess Code Words Available in nCm Code Sets	41
6.4. Error Control Codes using nCm Code Words	42
7. Preliminary Data	45
7.1. Experimental Results from Prototype Design.....	45
7.2. Bandwidth / Eye Diagram Analysis.....	47
7.3. Common-Mode Noise Analysis.....	48
7.4. Code Word Bit Error Rate Analysis	50
8. Preliminary Analysis.....	53
8.1. Error Control Coding over MBDS Signaling	53
8.2. Encoding and Decoding Hierarchical Error Correction Codes.....	56
8.3. Example	58
8.4. Partitioning an nCm Code Set.....	59
8.5. Choosing a Symbolic ECC Mechanism.....	61
8.6. Computing Code Rate as a Function of ECC Parameters.....	62
8.7. Required Hardware for Hierarchical ECC Implementation.....	63
8.8. Results for Hierarchical ECC Mechanisms	64
9. Research Plan.....	71
Appendix A: Demonstrator Chip Design.....	73
Driver Design.....	73

Receiver Design	74
Test Setup and Results	76
BIBLIOGRAPHY	81

1. Introduction

In this thesis we present a new type of error control codes called lightweight hierarchical error control codes (LHECC). These codes add capability for signal error recovery over links that are built using a new high-performance off-chip signaling technology, called ‘Multi-Bit Differential Signaling (MBDS),’ which has been recently developed by our research group. MBDS is an extension of Low Voltage Differential Signaling (LVDS), a state-of-the-art technology that is currently used to construct high-speed off-chip channels. The primary advantage that MBDS offers over LVDS is higher code density while retaining equivalent noise immunity. This means that an MBDS link has a higher information capacity than an LVDS link when both technologies utilize the same number of physical interconnections. MBDS technology accomplishes this partly through its use of a channel code called ‘N choose M (nCm)’ encoding. This channel code defines a fixed set of allowable messages that may be transmitted over an MBDS channel. We developed LHECC codes to operate seamlessly over top of the nCm code set in order to provide MBDS links with error control capability.

General purpose ECC codes are used in environments where a channel can transmit any arbitrary bit pattern and where code rate overhead is not a critical consideration. For these reasons, our new class of ECC codes must fulfill two requirements. First, the code must be effective over MBDS channels and operate while being constrained to the nCm code set for transmission.

Second, when compared to traditional ECC codes, it must require minimal code overhead relative to the intrinsic code rate of MBDS channels. As a result, we refer to this new class of code as being *lightweight*. This second requirement exists in order for MBDS links coupled with error control support to provide high code density relative to differential signaling.

LHECC codes satisfy these requirements and effectively provide error control support to MBDS channels. This is achieved by exploiting two properties of the nCm channel code. First, because only certain binary values constitute valid nCm code words, nCm code sets have an inherent ability to detect certain types of bit errors. Second, because the size of nCm code sets is not an integral power of 2 (for small code word widths), nCm code sets offer potentially unmapped code words when used to represent binary data. These two properties are used to offset the costs required for adding error control capability to an MBDS channel.

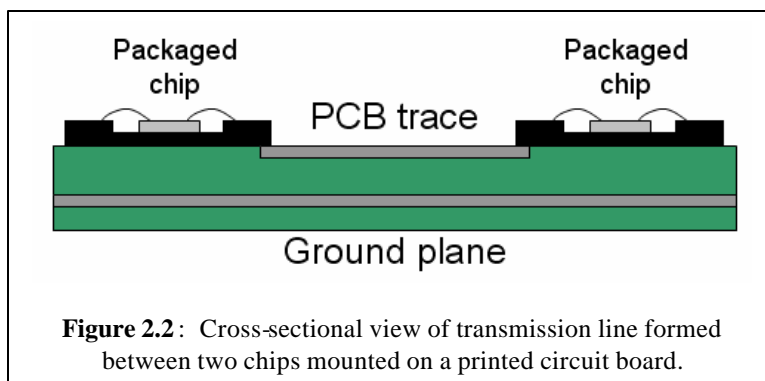
LHECC adds error control through hierarchical encoding. Multiple MBDS channels in parallel effectively form a code block used for a top-level symbolic ECC code. The top-level code is used to encode raw data by choosing a sequence of symbols. These symbols are also used to place rules on which nCm code words may be selected to in order to represent each symbol. Additional raw data is encoded in the selection of nCm code words for each symbol. Bit errors may be detected in individual nCm code words and may be corrected with information encoded in the top-level code. Additional code space is gained from the multiplicative increase in the number of unused nCm code words when multiple MBDS channels are used in parallel.

The balance of this proposal is as follows. In chapter 2 we describe our motivation for this work. Chapter 3 provides background information on traditional off-chip signaling techniques as well as Multi-bit Differential Signaling and its channel encoding mechanism. Chapter 4 is a survey of previous work in error control coding and high-performance off-chip signaling technologies. Chapter 5 formalizes our research problem. Chapter 6 describes our approach. Chapter 7 shows preliminary results from our prototype system. Chapter 8 describes preliminary analysis of our approach. Chapter 9 describes our research plan. Finally, an appendix is provided that provides technical details of our prototype system along with test results.

There are several challenges in developing high-performance electrical off-chip signaling technology. Many of these challenges stem from a general motivation to minimize the chip resources reserved for off-chip signaling. The only way to achieve this is for chip designers to multiplex off-chip I/O signaling through high-speed serial links. There are several reasons for this. First, I/O pins and real estate are the most precious resources in chip design. For I/O pins, this is especially true in highly integrated designs where power and ground connections must account for a large portion of total chip pin-out, due to high power requirements and minimization of power supply noise. In addition, off-chip driver circuits must be physically large and thus consume a large amount of chip real estate. This is due to an electrical phenomenon called the skin effect, where the effective impedance of transmission lines increases with signal frequency. This creates an inverse relationship between signal frequency and signal amplitude (gain). High-frequency drivers must support high amplification in order to be capable of sourcing enough electrical current to compensate. The amount of current that can be drawn through driver transistors is independent from the fabrication technology and is a function of the size of the transistor. These transistors become proportionally larger in area as compared to minimum chip features over successive generations of fabrication technology. In addition, these high-amplification drivers have power requirements that are relative to their size, creating another resource constraint for high-amplification drivers.

In order to describe the challenges in developing high-speed off-chip signaling, we must provide a brief overview of the physics of transmission lines. In this context, transmission lines model the signaling paths through the packaging of each chip in a link (including wire bonds and solder balls on either side of the wire bond) as well as the printed circuit board traces that connect the

chips. This type of transmission line is shown in Figure 2.2. In general, the integrity of signals sent through transmission lines degrades as signal frequency increases, beginning at the point in which the signal wavelength matches the length of the transmission line [78]. Transmission lines formed by chip-to-chip interconnect contains a relatively large amount of metal as compared to on-chip traces. As a result, these transmission lines contain capacitance and inductance that create a low-pass frequency response and filter out high-frequency signals and signal components. For example, an off-chip trace is likely to have a capacitive load of 2-3 orders of magnitude larger than an on-chip trace [76]. In addition, reliable reception of high-frequency signals is affected by noise, crosstalk, signal reflection, jitter, and PCB resistive and dielectric loss. These problems become increasingly serious when signaling channels are heavily loaded by multiple drops, run adjacent to parallel channels, or distributed over relatively long distances. High-performance signaling requires careful design of driver and receiver circuits along with techniques for data encoding or signal modulation. These techniques may be used to compensate for the loss of signal integrity at high signaling frequencies and provide advanced capability for synchronization, noise filtering, channel interference, and error recovery.



Definitions of Signaling Metrics

In this section we define the terminology used in this thesis. First, we define throughput, which is an important metric for characterizing off-chip signaling technology. Throughput is measured in bits per second, and defines how much information can be reliably sent across a link per unit of time. Equation 2.1 defines throughput for a chip-to-chip link.

$$\text{throughput} = (\text{channels}) \bullet (\text{connections} / \text{channel}) \bullet (\text{code rate}) \bullet (\text{channel frequency})$$

Equation 2.1: Link Throughput

In this work, we refer to a channel as a logical connection shared between a transmitter and one or more receivers. A channel may be implemented with a single physical connection (wire) or multiple physical connections in parallel. In Equation 2.1, channels refers to the number of channels in a link, while connections/channel refers to the number of physical connections that make up a channel. Raw data is carried over a channel as a random sequence of code words chosen from a predefined code set. The code set defines a set of allowable messages that may be sent over the channel. Code rate defines the information capacity of each physical connection in the channel. Code rate is measured in bits per code word per connection, and is computed by taking the log (base 2) of the size of code set and dividing by the number of physical connections per channel. Channel frequency is defined as the maximum switching speed (or transmission rate) for an individual physical connection where the signal state (through the detection of voltage or current) can be reliably sampled and interpreted by the receiver while remaining synchronized with the transmitter. The maximum frequency of a signaling technology depends on other metrics, such as signal integrity or signal-to-noise ratio. Channel frequency is measured in code words per second.

2.3. Circuit-Level Data Encoding

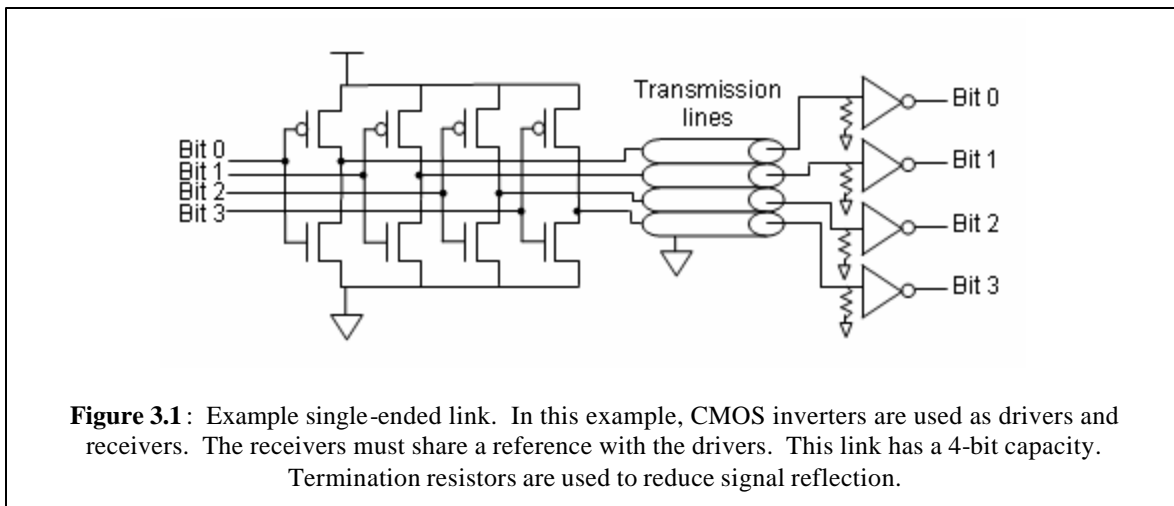
In general, data encoding techniques may be used to improve certain characteristics of a particular signaling technology such as the maximum signal frequency or the code rate. For example, many forms of data encoding are used to encode additional information into a link used for recovering timing/synchronization information in the presence of jitter (such as Manchester encoding), handling signal reception errors (such as error control encoding), or reducing the effect of noise (such as differential encoding). These types of data encoding come at a cost to code rate but improve maximum signal frequency. Other types of data encoding allow single physical connections to carry more than one binary value using signal modulation techniques (such as 4level pulse amplitude modulation). This type of encoding comes at a cost of maximum signal frequency but improves code rate. In this work, we utilize several data encoding techniques in order to manipulate various terms in Equation 2.1 to maximize link throughput.

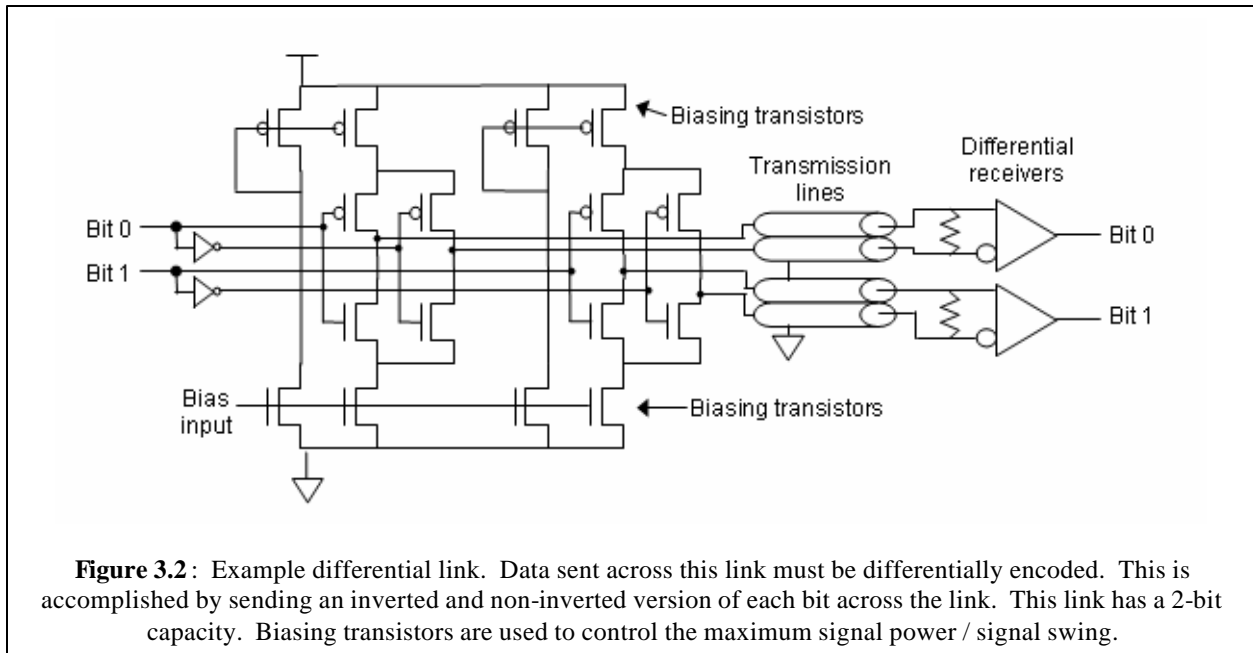
3. Background

3.1. Existing Signaling Technologies: Single-ended and Differential

Two conventional methods for building digital chip-to-chip links are single-ended strategies and differential strategies. Examples of these types of links are shown in Figures 3.1 and 3.2.

In a single-ended strategy, channels are formed by a single physical connection. The code set is $\{0, 1\}$ and each channel carries one bit of information. Therefore, the maximum code rate of a single-ended channel is one bit per code word per connection, assuming no additional channel encoding is used (although this is usually not the case). However, single-ended links are extremely limited in their maximum signal frequency due to their susceptibility to various noise sources. Signals carried by single-ended links must share a signal reference among all drivers and receivers that form the link. This shared reference may be a power supply ground or an off-





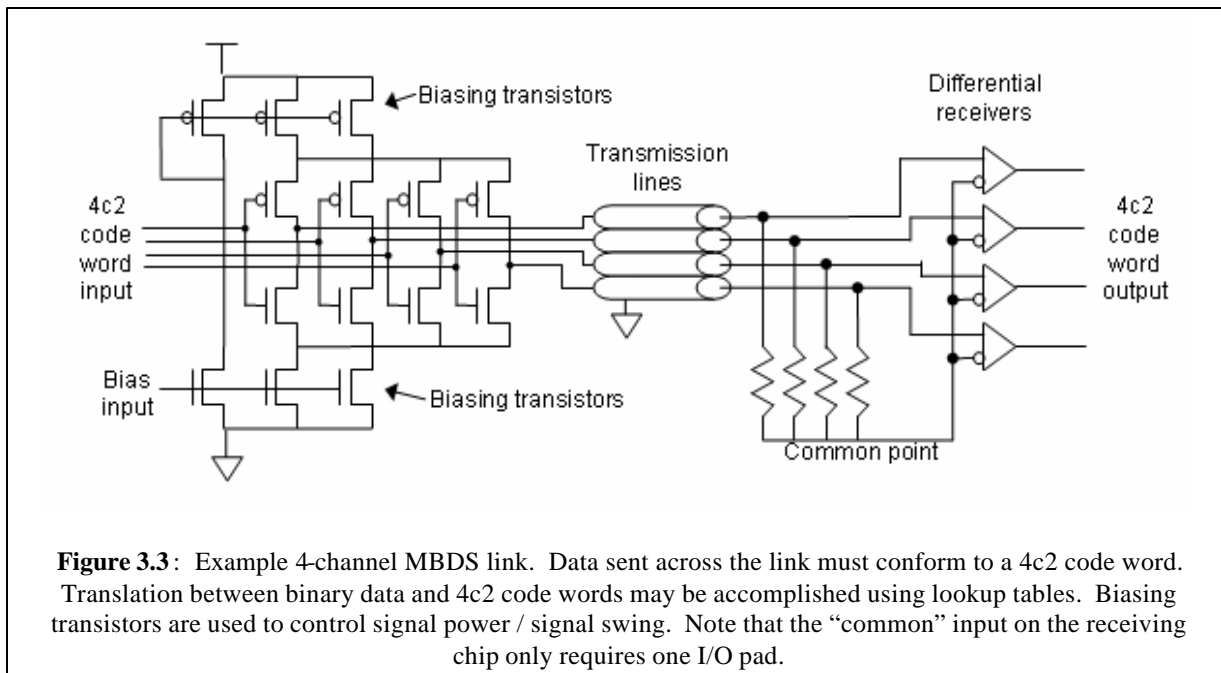
chip voltage source. In any case, this shared reference is a source of noise that ultimately distorts the received signal. Noise also affects the signal over the physical connections. There is no simple mechanism to compensate for this type of noise with a single-ended link.

Differential strategies are used in applications where high-performance chip-to-chip signaling is required. As an example, we examine a type of differential signaling called Low Voltage Differential Signaling (LVDS) [2]. LVDS is defined by the ANSI/TIA/EIA-644-A standard and is currently used in such high-speed signaling technologies such as Firewire (IEEE 1394), SCSI LVD, Flat Panel Display (FPD) Link, LVDS Display Interface (LDI) and OpenLDI standards [77]. An LVDS channel is formed by two physical connections. The code set is defined as {01, 10} and each channel is capable of carrying one bit of information. Therefore the maximum code rate of an LVDS channel is 0.5 bits per code word per connection assuming no additional channel encoding is used. A LVDS driver operates by sending current through one of its two physical connections in order to transmit a binary value. LVDS requires no shared reference so

it is immune to reference noise. In addition, LVDS drivers have relatively little “switching noise” because they draw a constant amount of supply current at all times. At the receiver, a termination resistor is connected between the two physical connections in order to match transmission line impedance and minimize signal reflection. A LVDS receiver interprets a positive or negative voltage difference over the termination resistor as a binary value and converts this signal into a voltage suitable for on-chip transmission. Common-mode noise affecting the physical connections generally affects both connections equally. This type of noise is canceled out at the receiver because the receiver only senses the voltage difference between the two physical connections. This type of noise is referred to as common-mode noise. This high level of noise immunity allows LVDS receivers to operate using small signal voltage swing. The ability to utilize small voltage swing reduces the overall detrimental effect of the low-pass frequency response caused by transmission line inductance and capacitance, allowing such links to operate with high signal frequency. However, LVDS links suffer from a significant disadvantage due to their low ratio of information capacity to number of I/O pads. This makes LVDS extremely costly in I/O pad resources when used to construct a high-throughput serial link. However, LVDS provides an example of how data encoding may be used to improve signal integrity. In an LVDS link, raw data at the driver is encoded such that each bit is converted into a differential code word ($\{01, 10\}$). The benefits of this encoding are a result of the physical properties of the differential driver, receiver, and termination design. However, this type of encoding is inefficient because two I/O pads are required to transmit a single bit.

3.2. Multi-bit Differential Signaling

Multi-bit differential signaling (MDBS) is a new link technology that retains the noise immunity of LVDS links while having a higher code rate. An example of an MDBS link is shown in Figure 2.3. To construct an MDBS channel, a “current-steering” differential driver design is scaled up to drive more than two physical connections. At the receiver, all physical connections are terminated into a common reference point. The values of the termination resistors may be matched to any desired termination load relative to the common point, and may be tuned to the transmission line impedance in order to minimize signal reflection. The voltage potential of the common point (relative to ground) will always be the average of the voltages of the physical connections in the termination network. This value is held constant by the code set used in MDBS channels. This code set forms the basis of “n choose m” (nC_m) encoding. We discuss nC_m encoding later in this chapter. LVDS receivers are used with this link technology to sense the voltage across each physical connection and the common point. A positive voltage or negative value of a channel relative to the common point is interpreted as a binary value.



MBDS drivers share the high performance electrical characteristics of differential drivers through their use of current-steering drivers, coupled transmission lines, and differential reception. This makes MBDS links and differential links comparable in signaling frequency. However, because MBDS links have a higher code rate than differential links, the overall throughput of an MBDS link is higher than a differential link that has the same number of physical connections. As a result, MBDS links may use fewer I/O pins than a differential link when required to carry an equivalent amount of information. Also, when compared to a differential link carrying an equivalent amount of information, an MBDS link requires fewer channels acting as current sources (corresponding to fewer 1-bits in the code words). This allows MBDS links to consume less power. In addition, nCm coding provides inherent error detecting capabilities as well as unused code words that may be used as a basis for additional encoding benefits. One such benefit is the ability to recover from signal transmission errors caused by jitter, noise sources, and other transient electrical effects. This additional level of data encoding may be tightly coupled into the link's driver and receiver design.

3.3. “N choose m” (nCm) Encoding and Support for Additional Coding Features

In an MBDS channel, the voltage of the common point remains fixed relative to the signaling channels by restricting the data that may be sent by each driver to a set of predefined code words, where each code word has a fixed number of 1-bits. In order to maximize the size of the code set, the number of 1-bits in each code word is always half of the number of physical connections (rounded down). In other words, such a link is comprised of n physical connections, where m of these connections are always set to 1-bits and $n-m$ channels are always set to 0-bits. We refer to

this coding system as “n choose m” (nCm) codes. For example, a 4 choose 2 (4C2) channel uses four physical connections such that at any time exactly two of the connections will be energized with 1-bits. When compared to the same four connections configured as two LVDS channels, the 4C2 channel has roughly 25% greater information throughput capacity. This is because the 4C2 channel can transmit any of the code symbols in the set {0011, 0101, 0110, 1001, 1010, 1100} while the two differential channels are restricted to the code symbols {0101, 1010, 0110, 1001}. This corresponds to approximately 2.58 bits of link capacity for the 4C2 channel versus 2 bits for the LVDS link.

Consider the set X_{nm} such that X_{nm} is the set of all valid code symbol encodings in an nCm channel. The size of X_{nm} , which is the number of available code symbols, is computed as:

$$f\{X_{nm}\} = \frac{n!}{(n-m)! m!}$$

Regardless of the number of code words allowed with a particular link configuration, each code word must be mapped to a binary data value at the inputs and outputs of the link. Since incoming and outgoing data will always be an integral number of binary bits (assuming a digital system), the effective bit width, bit_{eff} , defined as the number of bits coming into and out of the channel before encoding and after decoding is computed as:

$$bit_{eff} = \text{floor}(\log_2(\phi\{X_{mn}\}))$$

Using effective bit width as a metric, Table 3.1 compares the relative power consumption, pad count, and code word utilization for several MBDS link configurations to a differential link that carries an equivalent number of bits. Each physical connection in a link that sends a 1-bit as part of a code word acts as a current source and dissipates energy across the termination resistor. As a result, the relative power consumption of the two links is computed as $P_{eff} = m / bit_{eff}$, the

ratio of m , the number of wires energized to ‘1’ in the nCm link, to bit_{eff} , the number of wires energized to ‘1’ in an equivalent differential link. The relative pad count is computed as $RP=n/(2* bit_{eff})$, the ratio of n , the number of wires in the nCm link to $2*bit_{eff}$, the number of differential pads required in an equivalent differential link. From the data in the table it is clear that a 30-40% improvement in power efficiency and pad utilization can be achieved with relatively small values of n . Further, by selecting odd values of n , it is possible to trade smaller pad count reductions for greater power efficiency.

The nCm code set provides unused code words as a result of mapping binary values into nCm code words. The number of unused code words for each nCm code set is shown in Table 1 under ‘% Code Utilization.’ This value is computed as the largest power of two fewer than the number of code words divided by total size of the code set. These extra code words can be used to encode additional information into the link. One way to utilize these extra code words is to offset costs to code rate incurred by adding an error control mechanism over nCm coding. The addition of error control coding over MBDS signaling increases the maximum signaling

Channel type	Available Code symbols $f\{X_{mn}\}$	Effective bit width bit_{eff}	Relative Power Consumption $P_{eff} = m/bit_{eff}$	Relative Pad Count $RP=n/(2* bit_{eff})$	% Code Utilization	% Bit Utilization	Raw Code Rate $rate=bit_{eff}/n$
2C1	2	1	100%	100%	100%	50%	50%
4C2	6	2	100%	100%	66%	38%	50%
5C2	10	3	66%	83%	80%	31%	60%
6C3	20	4	75%	75%	80%	31%	67%
7C3	35	5	60%	71%	91%	27%	71%
8C4	70	6	66%	66%	91%	27%	75%
10C5	252	7	71%	71%	51%	25%	70%
11C5	462	8	63%	69%	55%	23%	73%
12C6	924	9	66%	66%	55%	23%	75%

Table 3.1: Comparison between MBDS links and differential links

frequency of the link because channel signal errors caused by noise and jitter may be recovered.

In addition to the extra code words available within nCm code sets, nCm code sets provide another property that makes them suitable for a channel error control mechanism. Valid nCm code words only represent a subset of possible n-bit binary values. As a result, nCm code sets have a built-in ability to detect many types of channel errors. This is shown in Table 1 under “% Bit Utilization”, which is computed as the number of code words divided by the number of possible binary values for each value of n (2^n). An nCm receiver may detect many types of bit errors if the result of the channel error yields a symbol that does not match an nCm code word. This error detecting ability can be used along with a more advanced coding mechanism to provide more powerful channel error correction ability.

In this work, we utilize these properties of the nCm code set to build a new error correction code that takes advantage of the features of the nCm code set. In this scenario, single or multiple signal errors are potentially caused when, during the instant of sampling at the receiver, the relative voltage between any particular physical connection and its associated common point does not exceed the minimum threshold voltage for that channel’s differential receiver. Signal conditioning on the output of the differential receivers will induce a stable CMOS value in this situation. However that resulting code word value may be in error. The most common causes of this type of error are jitter and noise. When constructing an ECC mechanism over the nCm code set, our ultimate goal is to maximize code rate and error correction capability while keeping the number of physical connections in the link as low as possible.

4. Previous Work

4.1. Overview

In this chapter, we provide an overview of previous work in four areas that are relevant to the work presented in this thesis. First, we present a brief history of coding theory and several current research projects in this field. Second, we provide a survey of chip-to-chip signaling technology that is currently employed or proposed for state-of-the-art / next generation systems. We examine this technology from a circuit, mechanical, and data encoding standpoint. Third, we present several research projects in high-performance electrical signaling technologies that employ data encoding, signal modulation, and pin/power reduction techniques. In each of these projects, a form of data encoding or signal modulation is utilized in order to increase signal integrity, reduce I/O pads, or reduce power. Lastly, we present a brief introduction to current research in high-performance non-electrical signaling technologies.

4.2. Error Control Coding

In this section, we describe an aspect of coding theory that deals specifically with dealing with signal errors introduced by noise in a transmission medium. The field of coding theory began in 1948 when Claude E. Shannon [37] wrote a landmark paper that showed that proper encoding of data can reduce errors in a noisy channel or storage medium. Shannon also derived the famous Shannon Limit, which defines the maximum amount of error-free information that may be transmitted over a channel given the channel's bandwidth and its signal-to-noise ratio (assuming random noise). Since then, there has been continuous work to develop new codes and efficient

encoding and decoding mechanisms in order to develop a communication channel that meets the Shannon Limit. In this section, we will review the relevant research that has made error correction coding possible. We then describe several relevant research projects in this field.

An error control code (ECC) is a mechanism for encoding information using more information symbols than are necessary to convey the raw information. ECC codes may be used over a communication channel when signals must be transmitted in the presence of noise. In such cases, the information contained in the signal is subject to random reception errors. The goal of an error control code is to detect or recover from such errors.

In 1950, Hamming [43] introduced the theory of linear codes. Linear codes are defined as (n,k,d,q) codes, where raw information at the information source is divided into blocks of length k , and encoded into a code word of length n . The code rate of such a code is defined as k / n . Any linear code has a distance d that defines its maximum error detection and error correction ability. The symbols that form the elements of a code are values of base- q (i.e. in a binary code, $q = 2$). Linear codes are defined as vector subspaces formed by a set of basis vectors referred to as a generator matrix. Every linear code may be defined in terms of its generator matrix. Every code also has a parity-check matrix, derived directly from the generator matrix, and is used for detecting and correcting errors in the code word. In order to encode linear codes, k -digits of raw data, represented by a $1 \times k$ information vector, are multiplied by the $k \times n$ generator matrix to yield the $1 \times n$ code word. In order to decode linear codes, the receiver multiplies the $1 \times n$ code vector by the transpose of the $(n - k) \times n$ parity-check matrix to yield a $1 \times (n - k)$ syndrome. If the syndrome is a zero vector, then transmission errors have not occurred in the code word. If

transmission errors have occurred, the error vector, a vector that has effectively been added to the code word to produce the error(s), may be determined from the value of the syndrome. Linear codes may be systematic or non-systematic. In a systematic code, every code word consists of the original k , unencoded values along with $n - k$ additional values called parity digits. Retrieving the original k information bits from the codeword is trivial in a systematic code. Non-systematic codes have code words that bare no resemblance to the original unencoded information. Systematic and non-systematic codes are equivalent in their relationship between the number of parity symbols and the resultant distance of the code. Building good linear codes and designing architectures to decode such codes forms the basis for most of the work in error control coding theory.

Several authors [38, 39, 40, 41, 42, 44] have developed and analyzed methods of systematically constructing, encoding, and decoding block codes. Prange [45] introduced the theory of cyclic codes, a subclass of linear codes that allow efficient hardware encoding using shift registers. Several authors [46, 47, 48, 49, 50] have devised architectures and algorithms that efficiently decode several types of cyclic codes. BCH codes, which form a subclass of cyclic codes, were discovered by Hocquenghem in 1959 [51] and independently by Bose and Chaudhuri in 1960 [52]. BCH codes were the first codes that could be systematically constructed given a desired block length and distance. These codes were proven to have a cyclic structure by Peterson in 1960 [53]. Peterson also developed the first decoding algorithm for these codes. Gorenstein and Zierler [54] were the first to apply BCH codes to nonbinary base p^m symbols, where p is a prime number. Further work in decoding algorithms for BCH codes was performed by Gorenstein and Zierler [54], Chien [55], Forney [56], Berlekamp [57, 58], Massey [59, 60], and Burton [61].

Reed and Solomon introduced the most widely used class of symbolic BCH codes in 1960 [62]. Reed-Solomon codes are used for error control in magnetic/optical disks and in several HDTV transmission standards. Reed and Solomon's most important result is the proof that their codes meet the Singleton bound. This means that their codes provided the highest possible error correction capability (highest distance) for valid values of n and k over symbol alphabets formed using Galois Fields.

For cyclic codes whose data symbols conform to a Galois Field (including BCH codes and Reed-Solomon codes), encoding is equivalent to representing the raw data (message) and generator matrix as polynomials. The coefficients of these polynomials are elements of the Galois Field. The coefficients of the message polynomial are the message symbols themselves. Encoding is equivalent to multiplying these two polynomials modulo- q (using polynomial convolution). The resultant polynomial's coefficients represent the code word. For a systematic ECC code, the parity digits may be computed by themselves by computing the remainder when dividing a shifted message polynomial by the generator polynomial. In hardware, encoding may be performed by lookup table, but this type of encoding is usually not feasible for large code word sizes. Encoding may also be performed using shift registers and XOR gates.

Reed-Solomon codes and any other code that meets the Singleton bound is known to be maximum distance separable (MDS). MDS codes have basic restrictions when applied as a symbolic ECC code. Such a code must be applied over base- p m symbols and is limited to a block length of p m. Seroussi and Roth [32] developed extensions to MDS codes that allow the maximum block size to reach p m + 1. De Boer [24] and Faldum and Willems [23] have

discovered a class of codes called “almost-MDS codes” or “near-MDS codes” that allow the maximum block size to reach $pm+1 - 1$ at the cost of at least one additional parity symbol (reducing code rate).

Convolutional codes, introduced by Elias [79] in 1955, form another subclass of binary linear codes. Convolutional codes differ from block codes in that the encoder and decoder contain memory. The n encoder outputs at any given time depend on the k inputs and the previous m input blocks. Convolutional codes are implemented as a linear sequential circuit and are typically used with small values of k and n and a large input memory m . Convolutional codes are also defined by their constraint length, defined by $n * (m + 1)$, which defines the maximum number of encoder outputs that can be effected by a single information bit. There is no known mechanism for constructing convolutional codes with a desired distance property. Convolutional codes must be constructed using computer search, restricting effective convolutional codes to small constraint lengths. Convolutional codes are used in extremely noisy channels where high reliability is needed and low code rate is acceptable. These codes are used in various radio applications such as CDMA/GSM cellular networks, 802.11 wireless ethernet, satellites, and deep-space probes. Decoding convolutional codes is typically performed by a Viterbi decoder [80], which is implemented as a finite state machine (sometimes represented by a “Trellis diagram”).

Forney [65] was the first to propose the use of concatenated codes, which is a method of constructing long codes from shorter codes. When using a concatenated code, binary data is broken into a number of equal-sized segments. A symbolic ECC code, called the “outer code”,

is applied such that each segment is treated as a symbol. After parity symbols (having the same length as the original segments) are added to the block, a binary ECC mechanism, called the “inner code”, is applied over each symbol. This causes parity bits to be appended to each symbol in the block including the outer code’s parity symbols. This method is used to decode extremely long codes with less decoding hardware. Concatenated convolutional codes used with bit interleaving techniques and iterative decoding form the basis of the popular Turbo Codes [67] that are seeing widespread use in modern radio communication systems.

Error correction codes also have the ability to correct special types of errors, such as erasures. An erasure is a symbol within a code word that is known by the receiver to be in error. Luby et al [63] has constructed an efficient mechanism for correcting erasures in certain types of symbolic block codes. Roth and Seroussi [64] have developed codes called “location correcting codes” that can correct symbols where the error value is known rather than the error location (as with an erasure).

“N choose m” coding, sometimes referred to as “m-out-of-n codes,” “balanced codes,” and “constant weight codes,” have been previously used for error detection and self-checking circuits. Ramabadran [66] has proposed using nC_m codes for perfect error detection in asymmetric channels, where bit errors are modeled as either a 0-to-1 flip or a 1-to-0 flip but not both. In such an error model, any bit error would result in an invalid codeword, making nC_m codes highly effective in this case. There has also been much work [73, 74] in computing the probability of an undetected error in binary symmetric systems that use nC_m codes along with an automatic repeat-request (ARQ) mechanism. Several researchers have developed self-checking

circuits and logic gates, called Totally Self-Checking Checker (TSC) circuits, through the use of nCm encoding [69, 70, 71, 72].

Szymanski [68] has proposed a forward error correcting mechanism for short-distance VCSEL-based optical links that uses concatenated codes along with an automatic repeat-request (ARQ) mechanism. In this case, small binary BCH codes form the inner codes and an error-detecting cyclic-redundancy-check (CRC) code is used for the outer code. A CRC code is a simple way to add parity bits to long binary strings and allows for detection for some types of multiple bit errors.

4.3. Survey of State-of-the-art Signaling Technology

In this section, we examine current state-of-the-art chip-to-chip signaling technologies and standards for next generation signaling technology. First, we examine the technology used by Intel and Rambus, who currently manufacture products with high-performance processor-to-memory interconnect. These technologies use a hybrid form of single-ended and differential signaling techniques. Next we examine Hypertransport and RapidIO, which are two competing interconnect standards for high-performance chip-to-chip and backplane applications. Both of these standards use differential signaling techniques along with additional data encoding as the basis for their interconnect standard.

The Pentium IV processor chip, manufactured by Intel Corporation, represents the most state-of-the-art and most widely used processor chip currently available [8]. This processor, manufactured in a 90 nm process, operates with a 3.4 GHz internal clock speed but is limited to an off-chip system bus speed of 800 MHz. This clearly illustrates the growing gap between on-

chip and off-chip signaling rates in modern systems, when compared to the original Pentium architecture that has a core speed of 100 MHz and a system bus speed of 66 MHz. The Pentium IV is packaged in a 478-pin flip-chip pin grid array package, of which 473 pins are used by the die. 85 pins are used for power and 179 are used for ground, which makes up 56% of the total pinout. It is clear that I/O pins are a limited resource in this design. This chip has an extremely high level of integration (~100 million transistors) and thus requires a large portion of its pinout to be dedicated to power and ground connections. Future chips that have higher levels of integration will have greater power demands and require an even larger portion of their pinout to be dedicated to power and ground connections, while leaving even less pins available for parallel I/O channels. This trend indicates that Intel will soon be forced to adopt a serial signaling technology for the system bus interface. The chip accepts a differential signal for its internal clock because this signal, generated off-chip, must be capable of running well into the GHz range. The system bus operates in the MHz range and is a highly parallel link using “quasi-differential” signaling. Quasi-differential signaling is essentially single-ended; however, instead of using ground connection as a signal reference, an explicit off-chip reference is used for all physical channels in the link. The system bus is made up of a 33-channel unidirectional address bus and a 64-channel bidirectional data bus. Both the address and data busses are divided into 16-channel segments that may selectively enabled. The 4 signal reference pins are located on the four inner corners of the PGA packages. The supply voltage is user-specified and is typically set 1 V, and the signal swing for the system address and data channels is specified as $\text{ref} \pm 10\% * V_{cc}$. This means that the signaling operates ± 100 mV relative to the reference points for a total single swing of 200 mV. Each 16-channel segment of the address and data busses has a single parity signal that is used for single-bit error detection. This represents a relatively primitive form

of error control coding as there is no mechanism in place for error recovery or request for data retransmission. The chip offers signal termination on die, and high-frequency decoupling capacitors on the package.

Another company who designs high-speed chip-to-chip links is Rambus Corporation. They have pioneered a serial bidirectional signaling technology, called Rambus Signaling Levels (RSL), which is used for their high-performance memory systems [35]. RSL and its updated counterpart, QRSL, are both quasi-differential signaling mechanisms based on explicitly referenced signaling channels. RSL is very similar to the signaling technology used in Intel's processor chip and thus has comparable transmission rate limitations. Rambus's most advantage signaling technology, called QRSL (Quad Rambus Signaling Levels), uses current-mode drivers to encode two bits on each channel by associating four unique voltage values across a 40-ohm termination. The four voltage ranges are recognized by a receiver that compares the input voltage relative to three fixed reference voltages. The set of voltage values, corresponding to each possible two-bit value, has a separation of 270 mV. This signaling technology is a variation on 4-Level Pulse Amplitude Modulation (4-PAM), discussed below. Rambus has developed and tested 2 Gbs links using this technology. Rambus also offers a separate signaling technology standard, called SerDes (Quad Serializer/Deserializer) that is intended for chip-to-chip and backplane applications. This is a fully differential link and uses 20 mA current-mode drivers over a 50-ohm terminated connection. This link offers clock extraction using 8-bit / 10-bit Manchester data encoding. It is designed to run over transmission lines of up to 30 inches. Links using this technology have been tested at 3.125 Gbs.

In his paper describing future interconnect standards, Nicholas Cravotta provides an examination [4] of RapidIO and Hypertransport and how these technologies seek to implement high-speed serial links rather than follow the longstanding trend of increasing interconnect bandwidth by designing increasingly parallel links. Both of these standards are competing for widespread acceptance, both are being updated, and both seek to reach practical implementation at 2 Gbs. RapidIO uses the Low Voltage Differential Signaling (LVDS) [2] standard for its physical layer, while Hypertransport uses Lightning Data Transfer (LDT) for its physical layer. LVDS and LDT are both fully differential standards with slightly different specifications (LDT is more power-aware than LVDS). Both RapidIO and Hypertransport are packet-switched protocols, and primarily specify a standard for chip-to-chip interconnect (as well as board-to-board interconnect). In addition, both standards employ primitive coding mechanisms to prevent data and/or clock loss. RapidIO uses an 8 bit / 10 bit encoding for clock extraction and error detection, while Hypertransport uses CRC over 512 data units for error detection. These approaches do not offer channel error correction ability at the link level, but offer retransmission mechanisms if a data or clock fault is detected.

4.4. Current Research in High-Performance Signaling Technology

Recent work in high-performance electrical link technology is concentrated on serial links that employ advanced circuit designs along with signal modulation and data encoding for pin and power reduction techniques. In this section, we will review several articles that characterize the problems governing high-speed link technology. Next, we will discuss and evaluate specific projects aimed at overcoming these problems through circuit and data encoding or modulation

techniques. These projects are representative of current trends in chip-to-chip signaling technology and provide a basis with which to compare our work.

An article written by Mark Horowitz and his group from Stanford University [3] is an exhaustive study on what factors contribute to signaling latency and throughput in device-level links. Included is effect of RLC parameters on driver and receiver circuits, modulation techniques, and considerations for coping with clock extraction, signal referencing, and jitter. His conclusion is that next-generation signaling technologies should employ high-speed serial links that include data encoding or signal modulation to address these problems. He proposes pulse amplitude modulation as a possible solution. Details on this technology are explained later in this chapter. In a similar article, [1], John Poulton from the University of North Carolina at Chapel Hill concludes that narrow, high-speed serial links, along with data encoding techniques are the keys to high-speed chip-to-chip interconnect.

Following similar motivations as the work presented in this thesis, the Stanford group [5] and a group from the University of Toronto [29] use signal modulation for reducing pin count, increasing code rate, and increasing signal integrity by optimizing the frequency characteristics for high performance signaling. Both groups are using variations on 4PAM (4-level pulse amplitude modulation), a multi-level signal modulation scheme similar to Rambus's QRSL technology. The basic idea of this modulation scheme is to switch from a digital "square-wave" signal to a modulated signal where pulses are sent with n different amplitudes, each representing $\log_2 n$ bits of data. The set of possible values that may be sent across this link is referred to as a

constellation. This modulation technique is very similar in concept to audio modulation applied to increase the throughput of telecom-based modems in the 1980's and early 1990's.

One of the projects from Mark Horowitz's Stanford group [6] proposes a solution to reduce I/O pin count and describes their work on a 2.4 Gbs link. Their solution for reducing I/O pin count on parallel single-ended links is to make the channels simultaneously bidirectional, where signals in both directions are superimposed on the same physical channel. Simultaneous bidirectional links effectively double the interconnect between two chips while using the same number of physical channels. The receiver in each transceiver subtracts its own transmit signal from the line voltage to generate the receive signal. They note that coupling the transmit signal to the receive signal creates a number of extra noise sources, which makes noise filtering and data synchronization an important issue in this design. They propose an aggressive receiver design that uses current integration techniques to filter out high-frequency noise on the receiver reference. In order to synchronize data in the link, both chips generate clock signals running in opposite directions. Each chip's incoming clock signal is used along with a phase-locked-loop to generate a receive clock that is delayed by a half-bit time in order to minimize the effects of skew and jitter. This technology does reduce pin count and provide a synchronization mechanism but introduces additional noise sources that must be handled by complex analog circuitry. MBDS signaling does not have this disadvantage. Very similar work in simultaneous bidirectional links was performed by the IBM Server Group in [7]. Unlike the Stanford design, they designed bidirectional differential links. The receivers work by having the ability to measure four unique voltages across the termination resistors. This voltage is determined by the single-bit transmit state of both chips in the link. Each chip, with knowledge of what signal its

driver is sending on the link, can use the termination voltage at its receiver to determine its receive signal from other transmitting chip. Once again, while this technology serves to reduce pin count, the price paid is in relatively complex analog receiver circuits (in this case, comparators).

Despite the high I/O requirement of differential signaling, there are several research projects that seek to design differential links that support ultra-high transmission rates. In [2], Stefan Hirsch and Hans-Jörg Pfeiderer present and characterize several high-speed differential receiver circuits. In [21], Motorola produces several high-speed differential (LVDS) drivers and receivers implemented using silicon-on-insulator technology. These links are fundamentally limited by their low code rate.

Several groups are investigating data encoding mechanisms for reduction of signal frequency, handling data synchronization, or reducing power. A group from the University of Ferrara, Italy [30] utilizes an encoding called minimum run-length guaranteed codes (MRLG). This type of encoding prevents single, isolated bits from occurring in a bit stream and allows encoding data with as much as twice the bit rate of the wire that is it transmitted on. They have also designed simple encoding and decoding logic. This encoding provides energy savings and increased signal integrity due to the overall reduction in signal frequency. In [33], a group from Cornell proposes a data encoding mechanism for a high speed asynchronous serial link. They use high fan-in multiplexing drivers and high fan-out demultiplexing receivers to multiplex eight data streams through a single-bit serial channel. In this design, three wires connect a transmitting chip to a receiving chip. These three wires implement a single-bit serial channel. Both chips

operate a three-state state machine and change states after each bit based on the current state and binary value sent or received. In each state, a pulse sent along one of the three wires represents a one-bit and a pulse sent along another wire represents a zero-bit. This allows the receiving chip to extract timing information and data from the channel without the need for a high-speed clock signal. They also propose a way to recover from driver and receiver state machine desynchronization due to bit errors. Binary error detecting codes are used over data sent over their link. If a bit error is detected, the driver and receiver reinitialize themselves and reset their current state through a resynchronization mechanism. Shin et al [75] proposes data encoding methods for reducing power consumption by reducing the number of bit transitions on data busses. These codes are variants of the classical Bit-Invert method, where bus data is inverted prior to transmission if the distance between the current data and the previously transmitted data is greater than half the width of the bus. Stan et al [76] proposes similar low power encodings over both time and space to reduce bus switching (time) and number of transmission lines driven high during any transmission period (space).

4.5. Current Research in Non-electrical Signaling Technology

Optical interconnect has been proposed as a solution for solving many of the problems inherent in electrical signaling. In optical links, signals are typically transmitted and received orthogonal to the chip's top or bottom surface. This technology promises ultra-dense, low-latency, and energy efficient 2-dimensional parallel links that have reduced capacitance, crosstalk, and reflections relative to electrical signaling technology. Vertical cavity surface emitting laser (VCSEL) and photodetector technology has made optical chip-to-chip links feasible, but

effective packaging technology is required to make such links practical. In this section, we review projects in optical chip-to-chip signaling technologies.

Our own research group is involved in guided-wave optoelectronic signaling research. One of our current projects is the construction of an optoelectronic multi-chip module demonstrator system [9, 10, 11]. This system is built by directly coupling optoelectronic components to multiple surfaces of a rigid fiber image guide (FIG) [9] structure. The structure is built using multiple layers of FIG cut at various angles, which allows both arbitrary signaling paths and signal fanout to be created among the chips mounted on each surface. Each optoelectronic component is constructed from a transparent-substrate silicon-on-sapphire logic die that is flip-chip-bonded to an 8x8 vertical cavity surface emitting laser array and an 8x8 detector array, both arrays having a 250 um pitch. The fiber image guide structure acts as both the structure and optical transmission medium for the multi-chip module. In this case, the goal is to provide extremely dense, highly parallel optical links to provide high bandwidth, low latency links between chips. We have also presented research on optoelectronic packaging technology based on similar principals [12, 13]. There are several other groups working with fiber image guides for parallel, chip-to-chip optoelectronic signaling. These include the NEC Research Institute, who integrate polymer FIGs directly into circuit boards to create optical signaling paths between chips [14, 15], and McGill University who has used flexible fiber image guides to create 10-channel parallel links [16]. There has also been interest in creating optical interconnects using 2-dimensional fiber ribbon cables rather than fiber image guides. Such work was performed in [25].

Many groups are currently developing free-space optoelectronic signaling technology. In this case, instead of using fiber to propagate optical signals, point-to-point optical links are created among chips sharing a common circuit board and using microlenses to direct (refract) vertically fired lasers and mirrors to reflect those signals back to neighboring chips. Groups from UCSD [17] and from George Mason University [18] have built and tested several such chip-to-chip links.

5. Problem Statement

Our research problem consists of the development of lightweight error control codes (ECC) that may be used with Multi-Bit Differential Signaling (MBDS) channels. These ECC codes must operate over MBDS channel coding and utilize the inherent properties of this channel coding in order to minimize code rate overhead. Once this is complete, we must prove that this error control code is optimal in that it achieves the best balance between code rate and error control capability. Next, in order to determine the effectiveness of our ECC code, we must first design an efficient architecture that implements this ECC code. Finally, we must develop accurate and realistic noise models for simulation and analysis in order to verify code and architecture functionality.

Traditional ECC codes (for error correction) are best suited for channels that both are capable of transmitting arbitrary bit patterns and are tolerant of significant overhead. As such, there are two reasons why these types of codes cannot be directly applied over MBDS channels. First, MBDS channels are constrained in that they can only transmit valid nC_m code words. Any ECC code applied over MBDS channels must be compatible with this constraint. This is not a trivial consideration. For example, all linear ECC codes (which include all state-of-the-art codes in use today) must by definition include an all-zero code word. Such a code word cannot be transmitted over an MBDS channel. Second, with MBDS it is not necessary to pay full ECC overhead relative to the raw, unencoded data prior to transmission. This is because there is

natural redundancy in the nCm code set which may be exploited for error control. The nCm channel code used by MBDS technology has two properties that can be used to offset the overhead cost incurred by adding error control capability. The first property is an inherent ability to detect bit errors. The second property is its unused code words when mapping MBDS code words to binary values.

The requirements for adding ECC support to MBDS channels lead to a hierarchical approach to encoding data. For an MBDS link consisting of multiple MBDS drivers in parallel, a top-level symbolic ECC code may govern selection of nCm code words transmitted across the link. This link will effectively form the code block for the top-level code. This top-level code may be used for code word error correction and detection. At the lower level, the inherent properties of nCm code words may be used to detect bit errors within individual code words. This additional level of error detection ability can be used to reduce overhead required in the top-level code. This overhead may further be reduced by the multiplicative increase in number of unused nCm code words when multiple MBDS channels are used in parallel.

Once an optimal ECC code for MBDS channels is developed, we must design an efficient architecture for encoding and decoding. In order to achieve efficiency, the architecture must satisfy two constraints. First, the encoder and decoder must be fast enough to operate at MBDS I/O speeds. In addition, they must be compact enough to fit into chip real estate reserved for the section of the I/O pad frame dedicated for off-chip I/O signaling.

After developing a hardware implementation of our error control code, we must determine precisely how various noise sources affect signals over MBDS channels and how effectively our error control mechanism compensates for these noise sources. This involves accurate modeling of various noise sources, running simulations and tests of MBDS channels with these noise models, and performing analysis of the results. These noise models include CMOS noise sources such as power supply noise, ground noise, switching noise, substrate/parasitic noise, and transistor shot noise, as well as transmission line noise (such as common-mode noise).

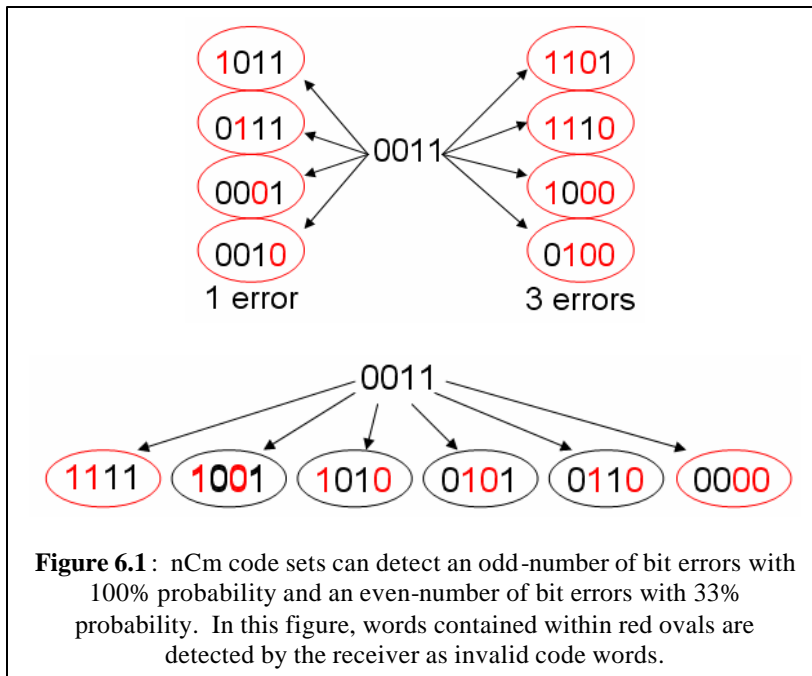
6. Approach

6.1. Overview

Our approach involves exploiting natural properties of the nCm code set in order to add additional data encoding features over MBDS channels. As stated in the previous chapter, the high code rate of MBDS technology is its primary advantage over differential signaling. Therefore, additional encoding mechanisms must come with minimal cost to code rate. In addition, these additional features must have a reasonable hardware implementation in order to be tightly integrated with MBDS drivers and receivers. In this chapter we highlight some of the properties of the nCm code set that may potentially be used for adding additional encoding features.

6.2. Inherent Error Detection Capability of nCm Code Sets

Any nCm code set has an inherent ability to detect certain types of bit errors that occur within an nCm code word. Bit errors will be detected by the receiving chip in an MBDS link if a word is sampled that does not contain an appropriate number of one-bits or zero-bits, as defined by the m parameter of the code set. As a result, any occurrence of bit errors occurring within an nCm code word will be detected by the receiver unless an equal number of 0-bits to 1-bits in the original code word are changed. This cannot be the case if an odd number of bits are changed within any nCm code word, meaning that any odd number of signal errors may be detected by the receiver. This includes any single bit errors, which most likely form the most common type



of error occurring within an MBDS channel. The receiver will detect the presence of an even number of bit errors only if an unequal number of 1-bits to 0-bits are changed.

The probability of detection of an even number of bit errors can be determined in the

following manner. Assume a code word is received that contains e bit errors where e is an even number. We will represent the occurrence of the errors as a vector of length n called an error vector. The error vector may be added (in modulo 2) to the original transmitted code word to yield the received word that potentially contains bit errors. Therefore the error vector will contain a 1-bit corresponding to each bit that is changed in the original codeword. In this case, there are exactly e 1-bits in the error vector representing the bits that are changed in the original code word. There are n bits in the original code word, so there are $C(n,e)$ unique error vectors. There are m one-bits and $n - m$ zero bits in the original code word, so only $C(m,e/2) * C(n-m,e/2)$ possible error vectors will yield a valid code word. Therefore, $1 - ([C(m,e/2) * C(n-m,e/2)] / C(n,e))$ defines the portion of even-numbered bit errors that can be detected by a particular nCm code set, given e . For example, if 2 bit errors occur to a $4c2$ code word, there are $C(4,2) = 6$ unique error vectors, and $C(2,1) * C(2,1) = 4$ possible error vectors that yield a valid code word. Therefore, $1 - (4 / 6) \approx 33\%$ of cases where two bits are changed will yield invalid code words

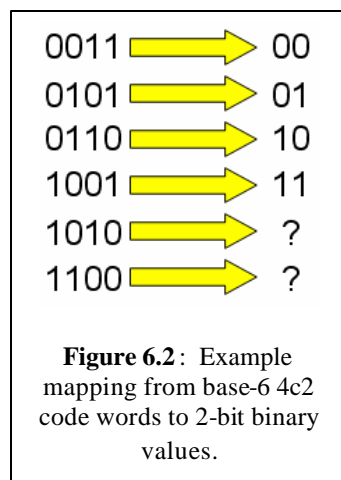
and will be detected by the receiver. Table 6.1 shows what portion of even numbered bit-errors are detectable by various nC_m code sets. Figure 6.1 shows an example of how various numbers of bit errors affect a $4c_2$ code word.

Code set	P(detect 2-bit error)	P(detect 4-bit error)	P(detect 6-bit error)
4c2	33%		
5c2	40%	40%	
6c3	40%	40%	
7c3	43%	49%	43%
8c4	43%	49%	43%

Table 6.1: Probability of detecting even-numbers of bit error for various nC_m code sets.

6.3. Excess Code Words Available in nC_m Code Sets

The nC_m code set constitutes a fixed set of allowable messages that may be transmitted over an MBDS channel. One way to view this is that these messages conform to a base- n numerical system, where n is the number of code words and is not an integral power of 2 for any small code word width. As a result, nC_m code sets provide excess code words that may be used to facilitate additional encoding features. In a digital link, an arbitrary mapping must be established between base-2 values and nC_m code words. For example, in a $4C_2$ code set there are 6 code words. Of these, we expect to use only the code space that spans an integral number of bits, e.g. $\text{floor}(\log_2 6) = 2$. For 2 bits, 4 code words are mapped to nC_m symbols, leaving 2 symbols potentially



unused. An example of this mapping is shown in Figure 6.2. If an error control mechanism is applied over nC_m code sets, these additional code words may be used to recover code rate lost to ECC overhead.

One way to exploit excess code words is to use multiple nC_m drivers

in parallel. The effective code rate of the link increases as the number of drivers used in parallel grows. For example, a single 4C2 driver has 6 code words in its code set and may carry $\text{floor}(\log_2(6)) = 2$ bits. This yields a code rate of 2 bits for 4 physical connections equaling 50%. However, two 4C2 drivers are used in parallel is equivalent to $\text{floor}(\log_2(6^2)) = \text{floor}(\log_2(36)) = 5$ bits, yielding an overall code rate of 5 bits for 8 physical connections, equaling 62.5%. In general, in order to increase the code rate by using parallel nCm drivers requires d drivers, where $d = \text{floor}(1 / (\log_2 cw - \text{floor}(\log_2 cw)))$. That is, for every d drivers used in a link, the total throughput is increased by one bit. This is because the sum of the fractional portion of the “bit width” of the nCm code set meets or exceeds 1.

One limiting factor to increasing code rate in this way is the lookup table sizes required to perform the mapping between code word combinations and binary values. Assuming that extra bit(s) are gained from the multiplicative increase in unused code words, the lookup table size for any arbitrary link built using d nCm drivers rises exponentially with respect to d . Clearly, additional encoding features which take advantage of these unused code words must also address this problem.

6.4. Error Control Codes using nCm Code Words

Assume an MBDS link that consists of several MBDS channels in parallel (in order to take advantage of unused code words). These channels all conform to the same nCm code set and thus each has n physical connections and a code set of size C . There are two ways that traditional error control codes may applied to this link.

1. Treat each MBDS channel as a symbol used in a symbolic ECC code (such as a Reed-Solomon code)

A symbolic ECC code may be applied over this link such that each MBDS channel, carrying an nCm code word, is considered a symbol. These symbols can be considered to originate from an alphabet of size 2^n or of size C . In either case, some of these symbols will carry data and others will carry parity information. In this arrangement, the inherent error detection capability of the nCm code set is unused. In addition, the code rate of such a link will be low because several MBDS channels will be used only to transmit parity information.

2. Treat all the bits that form the link as one binary ECC code word

In this case, we associate the concatenation of each nCm code word in the link as a binary ECC code word constructed from a linear block or convolutional code. There are several issues with this strategy. First, a binary ECC code set which includes only combinations of valid nCm code words is impossible to build for several reasons. First, all binary ECC code sets must (by definition) include a code word consisting of all zeros. No nCm code set contains an all zero code word. Second, there is no reasonable mechanism for developing a generator matrix for a code based upon a predetermined set of code words. Third, balanced-weight binary ECC codes are extremely inefficient where asymmetric errors may occur.

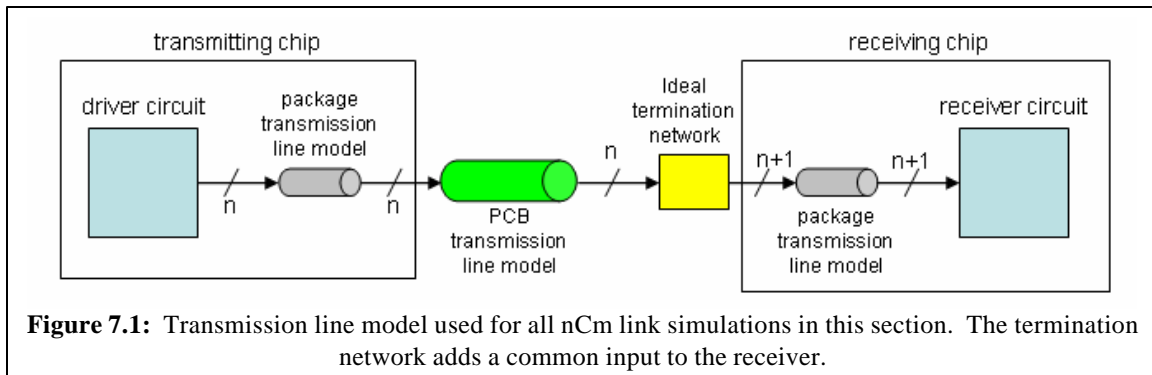
Neither of these methods is acceptable. As such, our approach is to use a hierarchical method where nCm code sets are partitioned into equal-sized subsets such that each subset has a

predetermined Hamming distance that is consistent for each subset. This binary ECC mechanism forms the lower-level of the code. These subsets are arbitrarily enumerated and used as a basis for an upper-level symbolic code. If an MBDS receiver detects an invalid code word in the link, the upper-level symbolic code can be used to determine the original subset of the code word in error. Using this knowledge, along with the binary value of the code word that is in error, the receiver can correct the error. This algorithm is explained in detail in chapter 8.

7. Preliminary Data

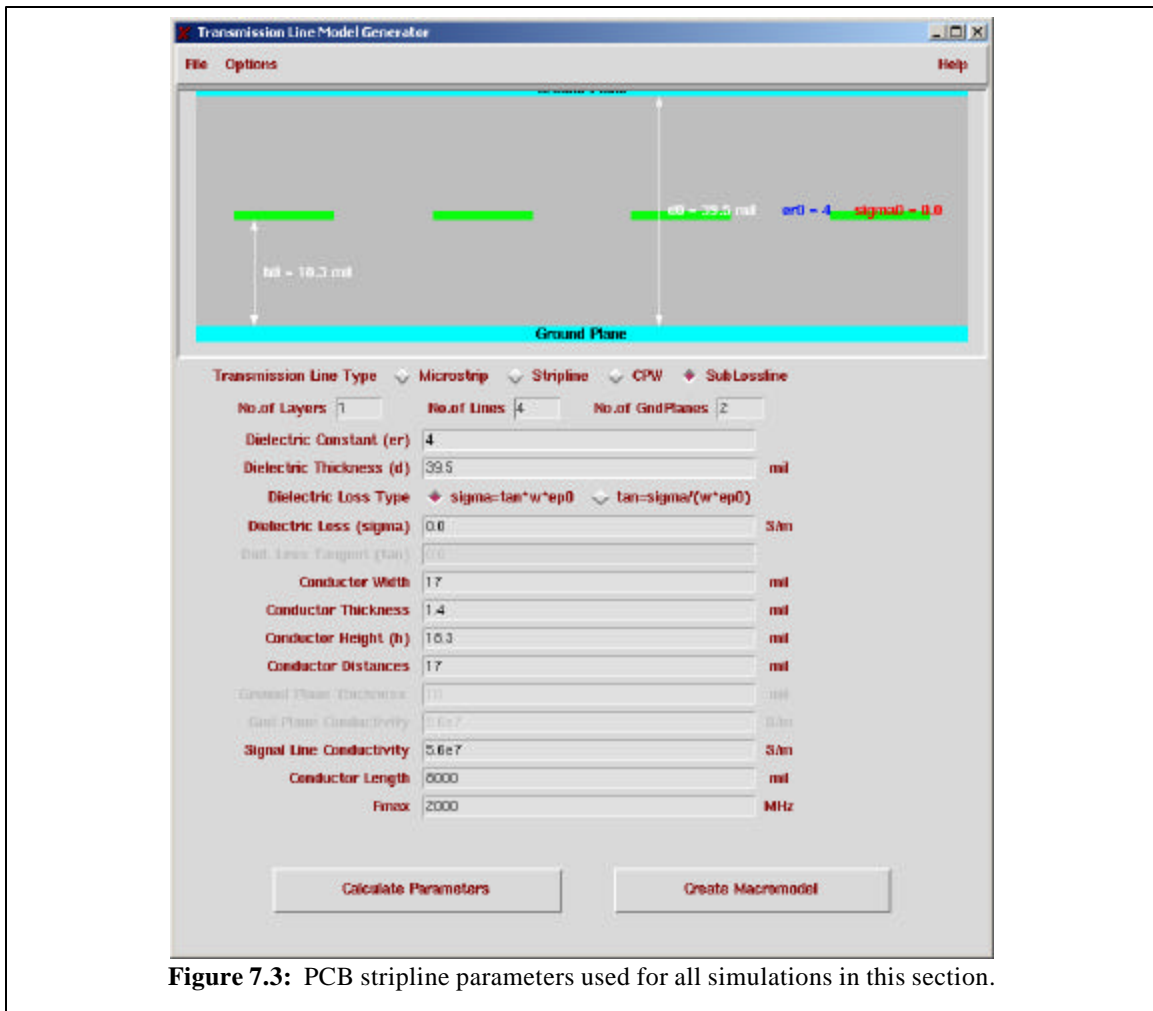
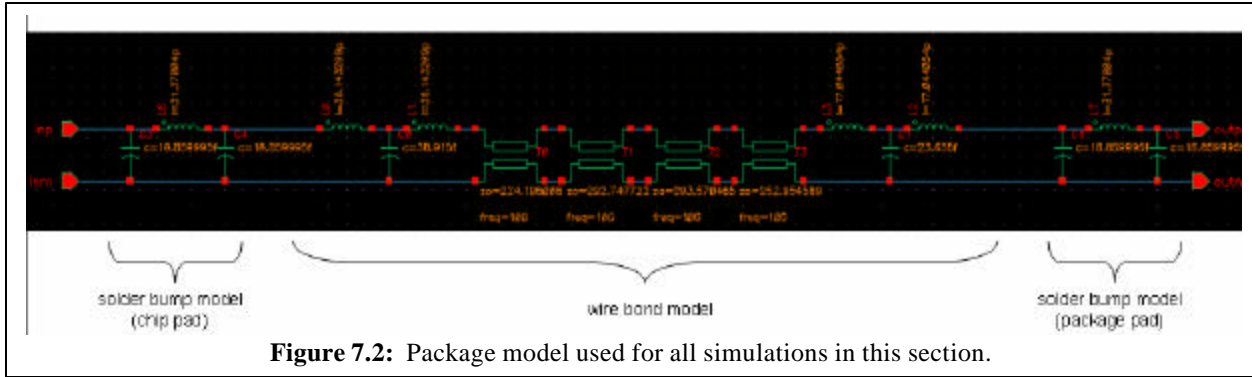
7.1. Experimental Results from Prototype Design

Using results from design simulation, we have made investigations into several primary areas. Each simulation includes extracted netlists for the nCm driver and receiver designs used in the fabricated SiGe prototype chip. These netlists include technology models for the SiGe CMOS transistors and layout parasitics. Also, each simulation includes complete transmission line models for the chip packages (including models for the chip pad and package pad solder bump and the wire bond) and an 8" PCB lossless coupled transmission line, designed to match a 50 ohm characteristic impedance. The termination network is simulated as an ideal resistor network using 50 ohm resistors. The top level setup is shown in Figure 7.1.



The chip package model is shown in Figure 7.2. In this case, reasonable values were chosen for the inductor, capacitor, and transmission line primitives.

The PCB transmission line was designed using the Cadence Transmission Line modeler as a lumped, lossless 8” stripline trace. An image of the transmission line model parameters is shown in Figure 7.3.



In the first study, we designed and simulated several implementations of MBDS links and tested the relative bandwidth of each. We present these results as a set of eye diagrams for a single channel in the link. The purpose of this investigation was to determine if there is any change in the behavior of the links at high speeds as the number of channels increase. In a second study, we study the effect of additive white Gaussian noise on the bit-error-rate of a 4C2 link. In a third study, we examine the driver bit error rates under various transmission rates.

7.2. Bandwidth / Eye Diagram Analysis

In this study, we compare the performance of a 2C1, 4C2, and 8C4 links for 500Mbps and 1 Gbps. Table 7.1 shows eye-diagrams for our 2-, 4-, and 8-channel driver designs. The input was a random sequence of valid code words. In the simulation, we can see the effects of increased link width with respect to signal integrity resulting from parasitic capacitance associated with the biasing transistor(s) and the fanouts at the drains of the n-transistors and p-transistors. The significance of these results is the comparable performance at each signal frequency between the differential (2C1) link and the 4C2 and 8C4 links. This suggests that there is little degradation in performance for wider links.

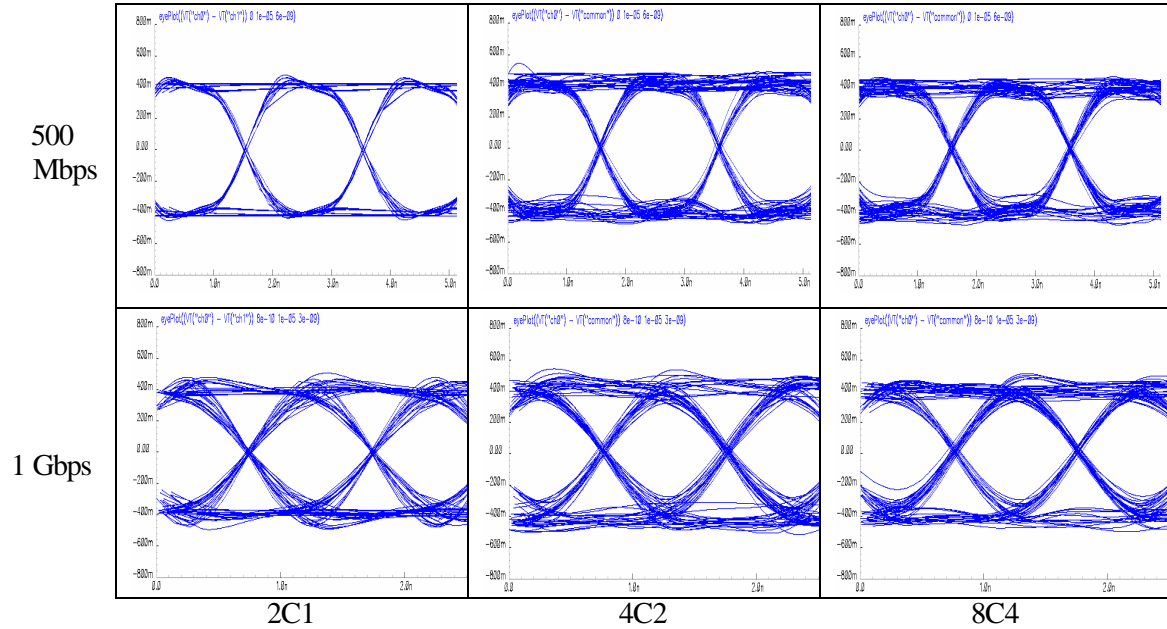


Table 7.1: Simulation eye diagrams from termination network for test channels (Y-axis is scaled from -800 mV to 800 mV in each plot)

7.3. Common-Mode Noise Analysis

In a second study we examined the effect of common-mode noise on the symbol error rate of a 4C2 link. We recorded the receiver input voltages of each of the four physical connections relative to the common point receiver input. These voltages were measured at the receiver chip I/O pad, which is after the package model (within the package). We then compared these voltages to the code word that was transmitted by the driver (assuming transmission latency). A code word error was recorded whenever the receiver input voltages were not consistent with the code word being transmitted by the driver. We ran these simulations using a consistent set of 10,000 random code words at 1.25 Gbps and 1.6 Gbps. In addition, we ran this analysis over a range of transmission latencies (transmission delay). In this case, latency is the difference in time from when the output is sampled at the termination network to when the input is sent by the driver. This is illustrated in Figure 7.4. Code word bit errors are modeled according to the

LVDS-standard [2], which states that in order for a bit to be transmitted without error, the voltage differential between the physical connection and the common point at the time of sampling must be $\geq 100\text{mV}$ for a 1-bit and $\leq -100\text{mV}$ for a 0-bit. This is shown graphically in Figure 7.5. In this simulation, any code word containing any number of bit errors is considered a code word error. We repeated these simulations after adding white Gaussian noise to each of the physical connections in the link. The noise was added by attaching a noise voltage source in series with each of the four connections in the termination network (except for common), which is before the package model for the receiver (outside the receiver chip package). The noise source follows a Normal distribution with a mean of 0 and a standard deviation of 1. This noise value is multiplied by 0.04 before being added to each node in the termination network (except for common). This yields a noise that typically ranges between 10 mV and 100 mV, consisting of 1-20% of the total signal. A sample plot of this noise is shown in Figure 7.6.

error-free code words						
latency (ps)	1.25 GHz (800 ps symbol time)			1.60 GHz (625 ps symbol time)		
	no noise	CM noise	% difference	no noise	CM noise	% difference
1350	9998	9989	0.04%	9885	9867	0.18%
1400	9998	9997	0.01%	9995	9990	0.05%
1450	9998	9998	nil	9998	9998	nil
1500	9998	9997	0.01%	9998	9997	0.01%
1550	9988	9974	0.14%	9993	9994	-0.01%
1600	9940	9899	0.59%	9983	9984	-0.01%
1650	9849	9675	1.74%	9939	9931	0.08%

Table 7.2: Effect of noise on channel code word error rate. Experiment run over 9,998 random code words.

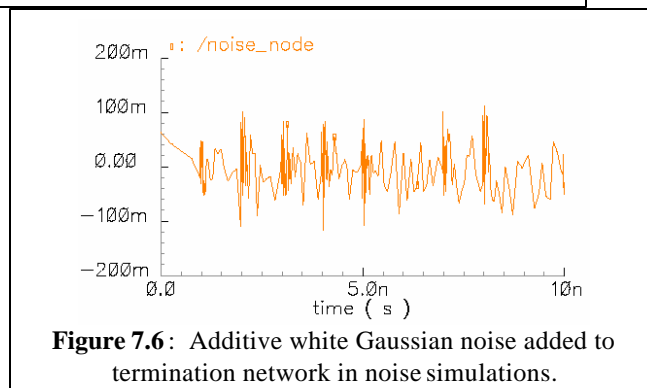
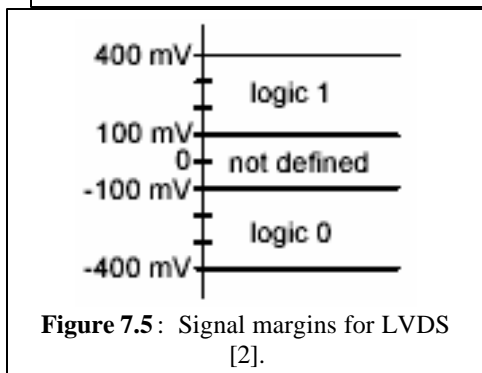
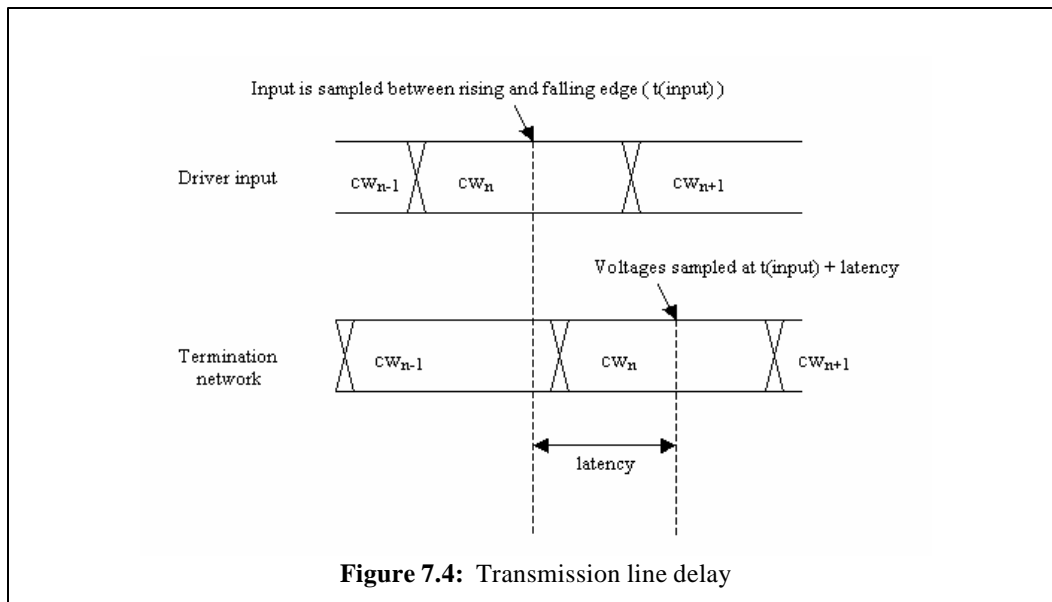
Table 7.2 shows the results of the bit error rate simulations, showing the relative difference between the numbers of error-free code words in the noise-free versus the common-mode noise simulations. “% difference” is computed as:

$$(\text{cw_correct}_{\text{no_noise}} - \text{cw_correct}_{\text{noise}}) / \text{total_codewords}.$$

These simulations show that the effect of common-mode noise on the bit error rate of the link was nonexistent at the optimal latency for the link, which was 1450 ps.

7.4. Code Word Bit Error Rate Analysis

In a third study, we wish to determine the types of symbol errors that occur when a 4C2 channel is run at high signaling speeds. In the case of our demonstrator system, we estimate this speed to be approximately 1.25 - 1.6 Gbps. As in the last study, the input set was a consistent set of 10,000 random, valid code words. The simulation was run using a 1.3 V bias voltage at 1.25 and 1.6 Gbps. The number of received symbols having 0, 1, 2, 3, and 4 bit errors was recorded as a function of latency at the receiver. As before, an error occurred when, at the time of sampling by the



receiver, the differential voltage between the physical connection and its associated common point did not have the correct sign and/or the magnitude did not meet the LVDS threshold of 100 mV. No Gaussian noise was added in this simulation. The results are shown in Table 7.3 and Table 7.4.

latency (ps)	0 errors	1 error	2 errors	3 errors	4 errors	total
1250	7568	1428	969	3	30	9998
1300	9913	73	12	0	0	9998
1350	9998	0	0	0	0	9998
1400	9998	0	0	0	0	9998
1450	9998	0	0	0	0	9998
1500	9998	0	0	0	0	9998
1550	9988	6	2	1	1	9998
1600	9940	20	28	4	6	9998
1650	9849	32	91	6	20	9998

Table 7.3: Bit error rate simulation results for 4C2 link at 1.25 GHz (800 ps bit time), 1.3 V bias (no noise). Latency is measured in picoseconds.

latency (ps)	0 errors	1 error	2 errors	3 errors	4 errors	total
1250	2711	3250	3586	60	391	9998
1300	4589	3383	1909	40	77	9998
1350	9885	95	18	0	0	9998
1400	9995	3	0	0	0	9998
1450	9998	0	0	0	0	9998
1500	9998	0	0	0	0	9998
1550	9993	5	0	0	0	9998
1600	9983	10	3	0	1	9997
1650	9939	36	18	1	3	9997

Table 7.4: Bit error rate simulation results for 4C2 link at 1.6 GHz (625 ps bit time), 1.3 V bias (no noise). Latency is measured in picoseconds. In this simulation, the number of “error-free” code words is at the maximum when the receiver samples the signal 1450 ps after driver transmission.

It is clear from these simulations that a 4C2 link can operate with no errors at 1.25 Gbps, assuming a jitter margin of 100 ps (1400 ps – 1500 ps). This frequency represents the maximum inherent error-free limit of the link. At 1.6 Gbps, infrequent single bit errors occur within this same 100 ps jitter margin. This is unacceptable if there is no mechanism to detect or recover from these errors. However, adding an error recovery mechanism that is capable of recovering from these types of channel errors will allow the link to reliably run at the faster signaling

frequency. This speed increase represents a 28% improvement. However, this is only worthwhile if the code rate overhead required for the ECC mechanism is less than 28%.

8. Preliminary Analysis

8.1. Error Control Coding over MBDS Signaling

In this chapter, we explain our approach to building hierarchical error correction codes over chip-to-chip links constructed using MBDS technology. These links are formed by multiple MBDS channels in parallel, distributed spatially or temporally, between a transmitting chip and one or more receiving chips. We also assume that each nCm driver in this link has a consistent number of channels and this use identical nCm code sets. The nCm code set associated with these drivers is partitioned into equal-sized subsets such that the Hamming distance of each subset will be a specified value, d_{symbol} , meaning that any pair of code words chosen from such a subset will differ in at least d_{symbol} bit positions. This partitioning is performed off-line, and the transmitting chip and the receiving chip will have *a priori* knowledge of this partitioning. The value chosen for d_{symbol} will determine the number of random bit errors that may be corrected with 100% probability within any single nCm code word, defined as t_{symbol} . t_{symbol} is computed in the traditional way for binary ECC mechanisms, i.e. $t_{symbol} = \text{floor}((d_{symbol} - 1) / 2)$.

For example, for $d_{symbol} = 4$, a 7c3 code set, consisting of 35 code words, may be divided into 7 equal-sized subsets, each having 5 code words. This partitioning will guarantee a single random bit error may be corrected within an nCm symbol. A valid partitioning of an nCm code set is not unique: there may be several possible partitionings of a code set. Table 8.1 shows an example

partitioning of the $7c3$ code set. These subsets are arbitrarily enumerated. An example enumeration is shown in the left-hand column of Table 8.1.

subset	code words
0	0000111, 0110001, 1001001, 1010010, 1100100
1	0001011, 0110010, 1000110, 1010001, 1101000
2	0001101, 0101010, 0110100, 1011000, 1100001
3	0001110, 0100101, 0111000, 1010100, 1100010
4	0010011, 0011100, 0101001, 1001010, 1110000
5	0010101, 0011010, 0100110, 1000011, 1001100
6	0010110, 0011001, 0100011, 0101100, 1000101

Table 8.1: Partitioning of a $7c3$ code set with $d = 4$.

Once an nCm code set has been partitioned and the subsets have been enumerated, we use the enumerated values of these subsets as a symbol alphabet for a second-level ECC mechanism. The second-level code is a linear block code, organized as an (n,k,s) code, consisting of a block of n symbols, where k symbols carry data and $n-k$ symbols carry parity. The code symbols are base- s values, where s is defined as the number of subsets in the nCm code set partitioning.

An (n,k,s) block code implies that the symbolic code has the ability to correct e_{block} symbol erasures and t_{block} symbol errors, where e_{block} and t_{block} are defined as functions of the symbolic code's distance, d_{block} . Specifically, these functions are the following:

$$e_{block} = d_{block} - 1$$

$$t_{block} = \text{floor}((d_{block} - 1) / 2)$$

The distance of the code is a function of the type of code used and the number of parity symbols. The Singleton Bound defines the maximum distance any block code can achieve given values for n and k . This bound is defined as:

$$d \leq n - k + 1$$

It follows that block codes that achieve the Singleton Bound will have the following erasure and correction capability as a function of n and k defined as:

$$e_{block} = n - k$$

$$t_{block} = \text{floor}((n-k) / 2)$$

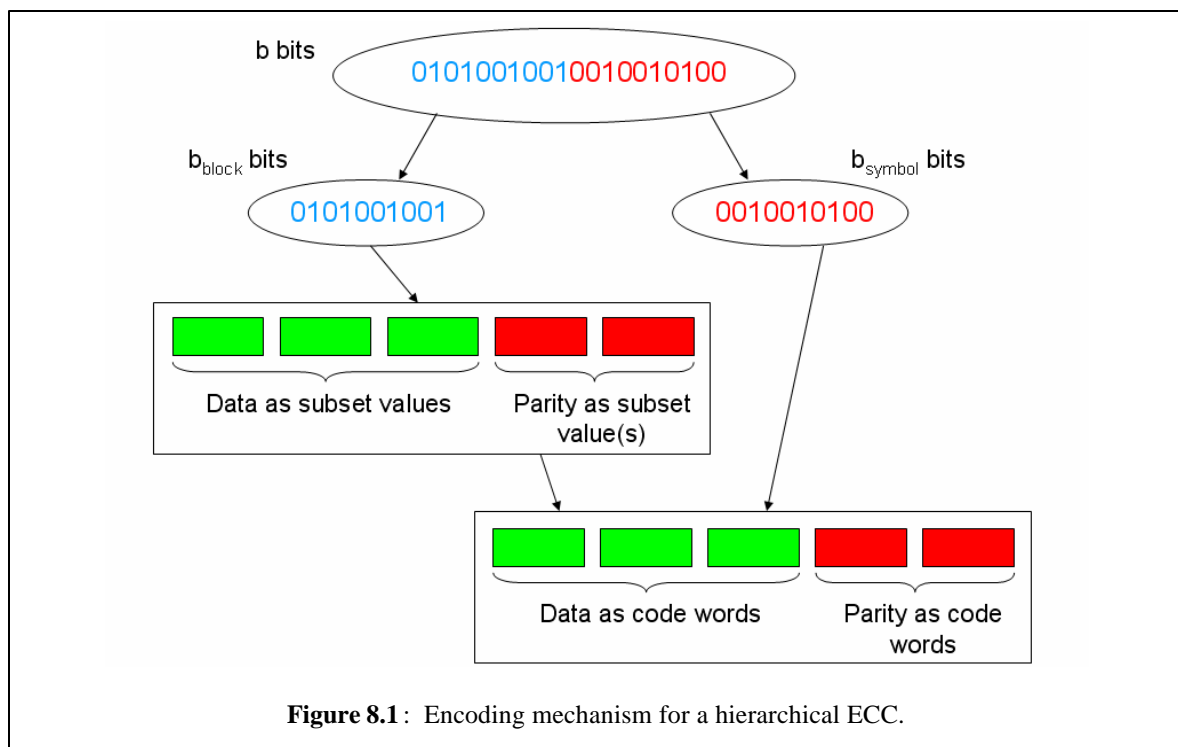
If single or multiple bit errors affect an nCm symbol in a link, the bit errors will manifest themselves in one of two ways. If the result of the error(s) yields a word that is not a valid nCm codeword, the receiver will detect that symbol as an *erasure*. In this case, the receiver has knowledge of the index of the symbol within the block that contains the bit error(s). If the result of the error(s) yields a word that is a valid nCm codeword, the receiver will most likely receive a code block where the parity value computation from the data symbols is inconsistent with the actual parity values received. In this case, the bit errors will constitute a symbol *error*. If this occurs, the index of the symbol that contains the bit error is not known and must be determined by the receiver using the error correction capability of the second-level block code.

Recall that the symbols in the block code reference the enumerated subset number to which the actual nCm codeword belongs. In the event of bit error(s), the receiver must be able to determine the subset value of the nCm code word(s) in error. Once this information is determined, the receiver has knowledge of the set of possible original code word values for the code word(s). The receiver must then determine which of these candidate code words have minimum distance to the received word.

In order to build an effective hierarchical ECC code, the nCm code set, partitioning, and block code type must be chosen to find a suitable balance of code rate and ECC capability.

8.2. Encoding and Decoding Hierarchical Error Correction Codes

Assume a link consisting of n nCm drivers. The nCm code set has been divided into s subgroups, each having c code words with distance d_{symbol} . An (n, k) block code with distance d_{block} is chosen as the second-level ECC code. A link that is capable of carrying b bits of data requires encoding of the data into two portions, b_{block} and b_{symbol} , where $b = b_{block} + b_{symbol}$. The first b_{block} bits will be encoded by choosing a sequence of k base- s digits through base-translation (binary to base- s). $n-k$ base- s parity symbols are computed based on these digits and added to the code block. For each of the digits in the code block, any of the c nCm code words that belong to the subset specified by the digit may be chosen (binary to base- c translation). The sequence of choices made will encode the remaining b_{symbol} bits of the message. This encoding process is shown graphically in Figure 8.1.



Any bit error(s) that occurs within a received nCm symbol within the code block will fall into one of the following categories:

1. A received nCm code word contains *less than or equal to* $\text{floor}((d_{\text{symbol}} - 1) / 2)$ bit errors, and the resultant word *is not* a valid nCm codeword. Any odd number of bit errors is guaranteed to produce an invalid codeword. For even numbers of bit errors, there is a fixed probability that the errors will yield an invalid nCm codeword, depending on the nCm code set that is chosen. This type of error may be corrected as an *erasure* if this type of error does not occur to more than $d_{\text{block}} - 1$ nCm code words in the code block. If this type of error occurs to more than $d_{\text{block}} - 1$ nCm code words, errors will be detected within those particular nCm code words but they cannot be corrected.

2. A received nCm code word contains *less than or equal to* $\text{floor}((d_{\text{symbol}} - 1) / 2)$ bit errors, and the resultant word *is* a valid nCm codeword. This type of error may be corrected as a *symbol error* if this type of error does not occur to more than $\text{floor}((d_{\text{block}} - 1) / 2)$ nCm code words in the code block. If this type of error occurs to more than $\text{floor}((d_{\text{block}} - 1) / 2)$ code words, but in less than or equal to $d_{\text{block}} - 1$ code words, errors will be detected in the code block but they cannot be corrected. If this type of error occurs to more than $d_{\text{block}} - 1$ code words, errors will only be detected if the second-level (symbolic) code is invalid. In any (n,k,q) code, only q^k / q^n code words are valid.

3. The received nCm code word contains *in excess of* $\text{floor}((d_{\text{symbol}} - 1) / 2)$ bit errors, and the resultant word *is not* a valid nCm codeword. This type of error will be detected as an invalid nCm code word but cannot be corrected.

4. The received nCm code word contains *in excess of* $\text{floor}((d_{symbol} - 1) / 2)$ bit errors, and the resultant word *is* a valid nCm codeword. A code block error may be detected if this type of error does not occur to more than $d_{block} - 1$ nCm code words in the code block. If this type of error occurs to more than $d_{block} - 1$ code words in the code block, code block errors will only be detected if the second-level (symbolic) code is invalid. In any (n, k, q) code, only q^k / q^n code words are valid.

8.3. Example

For example, assume an MBDS link consisting of three 4c2 drivers. We partition the 4c2 code set into three subsets with $d_{symbol} = 4$, allowing for the correction of a single bit error within any driver. The subsets are listed and enumerated in Table 8.2.

subset	code words
0	0011, 1100
1	0101, 1010
2	0110, 1001

Table 8.2: 4c2 code set partitioning.

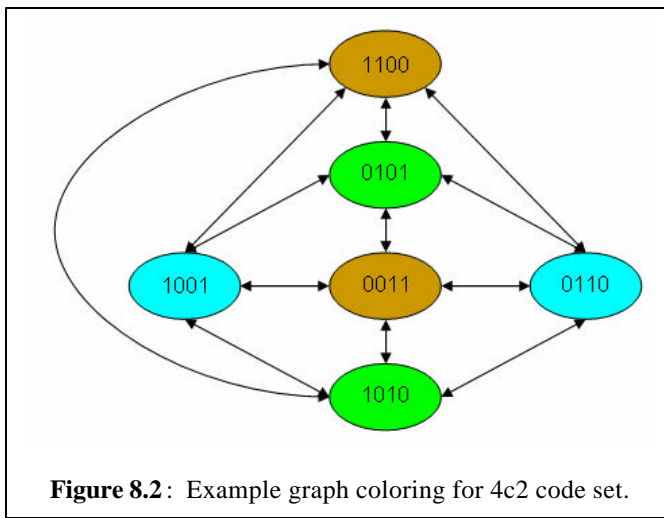
Assume a (3,2,3) checksum-based code is chosen as the second-level ECC block code. This hierarchical ECC code allows $b_{block} = \text{floor}(\log_2(s^k)) = \text{floor}(\log_2(3^2)) = 3$ bits to be encoded into the “data” portion of the block ECC code and $b_{symbol} = \text{floor}(\log_2(c^n)) = \text{floor}(\log_2(2^3)) = 3$ bits to be encoded into the code word choices for each symbol in the code block. This code allows a total of 6 bits to be encoded over 12 channels, yielding a code rate of 50%, matching the code rate of a differential link. Recall that 4c2 channels have a natural code rate of 50%, so error control is gained without penalty because the overhead is absorbed by the extra code words in the 4c2 code set.

Now, assume we wish to encode the binary value 111101. The first three bits, 111, are translated into a k -digit base- s value (2-digit base-3 value), or 21. A base- s checksum parity symbol is computed by adding the digits modulo- s ($2 + 1 \pmod 3$) to yield 0. The block code encoding is now 210. The remaining three bits, 101, are encoding by choosing $4c2$ code words from the subsets specified by the block code. The resultant code word sequence is 1001, 0101, 1100.

Now, assume these code words are transmitted across the MBDS link and are received as **1101**, 0101, 1100. The first code word is detected as an erasure because it contains three 1-bits. The first step in correcting this error is to determine the subset from which this code word originated. This is performed by subtracting the subset value of the last code word (the parity/checksum codeword) from the subset value of the data code word not containing bit errors, in modulo- s . In this case, the operation is performed as $0 - 1 = 2 \pmod 3$. The corrected code word is then determined by comparing the relative distances between the received code word and the two code words in subset 2. In this case, the distance between 1101 and 0110 is 3, while the distance between 1101 and 1001 is 1. The corrected code word is therefore 1001.

8.4. Partitioning an nCm Code Set

Partitioning an nCm code set into equal-sized subsets for a given distance d_{symbol} is equivalent to a graph coloring problem. In this case, each node in the graph represents



a code word in the nCm code set. Vertices are formed between any two nodes whose code words differ in less than d_{symbol} bit positions. An example graph coloring for a $4c2$ code set is shown in Figure 8.2.

We have developed a search algorithm over the space of nCm code words that performs a depth-first search in order to partition 4-, 5-, 6-, 7-, and 8-channel nCm code sets into subsets. The algorithm is as follows:

```
function partition (code_set, num_subsets, subset_size, distance), returns assign_state
initialize assign_state to empty
create empty stack of assignment states
push empty assignment state to stack

while unassigned codewords exist {
    pop assign_state from stack
    find smallest subset number that hasn't been assigned to subset_size codewords
    repeat for all possibilities {
        find an unassigned codeword that fits into subset given distance
        assign the codeword to subset and push resultant state onto stack
    }
}

return assign_state
```

Using this algorithm as a starting point and applying several pruning methods, we have successfully found the following partitionings shown in Table 8.3. In some cases, nCm code sets may be partitioned such that the number of subsets multiplied by the number of code words in each subset does not equal the original number of code words in the nCm code set. In this case, the code word utilization is under 100%. Surprisingly, this type of partitioning yields a higher code rate after an ECC code is applied. This represents another way to utilize excess code words, by purposely omitting certain code words that are counteractive for the purposes of error correction.

code set	code words (c)	subsets (s)	code words/subset	d_{symbol}	utilization
4c2	6	3	2	4	100%
5c2	10	5	2	4	100%
6c3	20	10	2	6	100%
6c3	20	6	3	4	90%
6c3	20	4	4	4	80%
7c3	35	7	5	4	100%
7c3	35	5	6	4	86%
8c4	70	10	7	4	100%
8c4	70	14	5	4	100%
8c4	70	35	2	8	100%
8c4	70	8	7	4	80%
8c4	70	7	8	4	80%
8c4	70	7	9	4	90%
8c4	70	8	8	4	91%

Table 8.3: Code set partitioning

8.5. Choosing a Symbolic ECC Mechanism

We have identified three classes of block codes for use in our hierarchical ECC mechanism.

Each class has unique properties that make it suitable for use under various circumstances.

The simplest type of block code is *checksum*. This is a $(n, n-1, q)$ code that supports at most a single parity symbol. Its distance is 2, which indicates that this code meets the Singleton bound. It has the ability to correct a single erasure, but it cannot correct symbol error(s). Its most important advantage is that it places no restrictions on q , the symbol base or the number of subsets for a given nCm code set. In order to calculate the parity symbol for a checksum code, the encoder performs a base- q addition of the symbols in the code block and uses the result as the parity symbol.

Another class of block code is called *maximum distance separable* codes, or *MDS* codes. MDS codes are defined as any $(n, n-1, q)$ code that meets the Singleton Bound. The most popular type of MDS code is the Reed-Solomon Code. MDS codes have two important restrictions which

place limits on the conditions under which these types of codes may be used. First, the value of q must be either a prime number or a power of a prime number. This restriction limits the types of nCm code sets and partitionings that can be used with this block code. These code sets include the $4C2$, $5C2$, and $7C3$ code sets. The second restriction on this class of codes is that the maximum block size, or n , is defined as $q+1$ [32]. This restriction will place limits on the maximum code rate that may be achieved under certain conditions.

The third class of block code is called *almost maximum distance separable* codes, or AMDS codes [23,24]. AMDS codes are defined as any $(n,n-1,q)$ code having a Singleton defect of 1. This means that one additional parity symbol is required to achieve the same distance as an MDS code, or $d = n - k$. This class of code also requires that q be a prime number or a power of a prime. However, it does not have the same restriction on block size as MDS codes. The maximum size of an AMDS code is $n = p^{m+1} - 1$ where $q = p^{m+1}$.

8.6. Computing Code Rate as a Function of ECC Parameters

Code rate is the most important metric for evaluating the success of the codes developed in this chapter. In this case, code rate is defined as the maximum number of bits that may be coded in a code block divided by the total number of channels required to transmit the code block. As the number of channels used to transmit a code block increases, the code rate also increases and the relative error correction capability decreases. This property is caused by two factors. First, nCm code sets have “unused” code words that multiplicatively increase as more nCm code words are added to a code block. Secondly, if the error correction capacity of the code block is held constant while the code block size is increased, the ratio of information to error correction

information increases, meaning the code rate increases but the relative error correction capability decreases.

Given an nCm code set with n channels, the number of subsets as s , the number of code words per subset as c , and the block code parameters n and k , the code rate is defined as:

$$rate = \lfloor n \cdot \log_2 c \rfloor + \lfloor k \cdot \log_2 s \rfloor$$

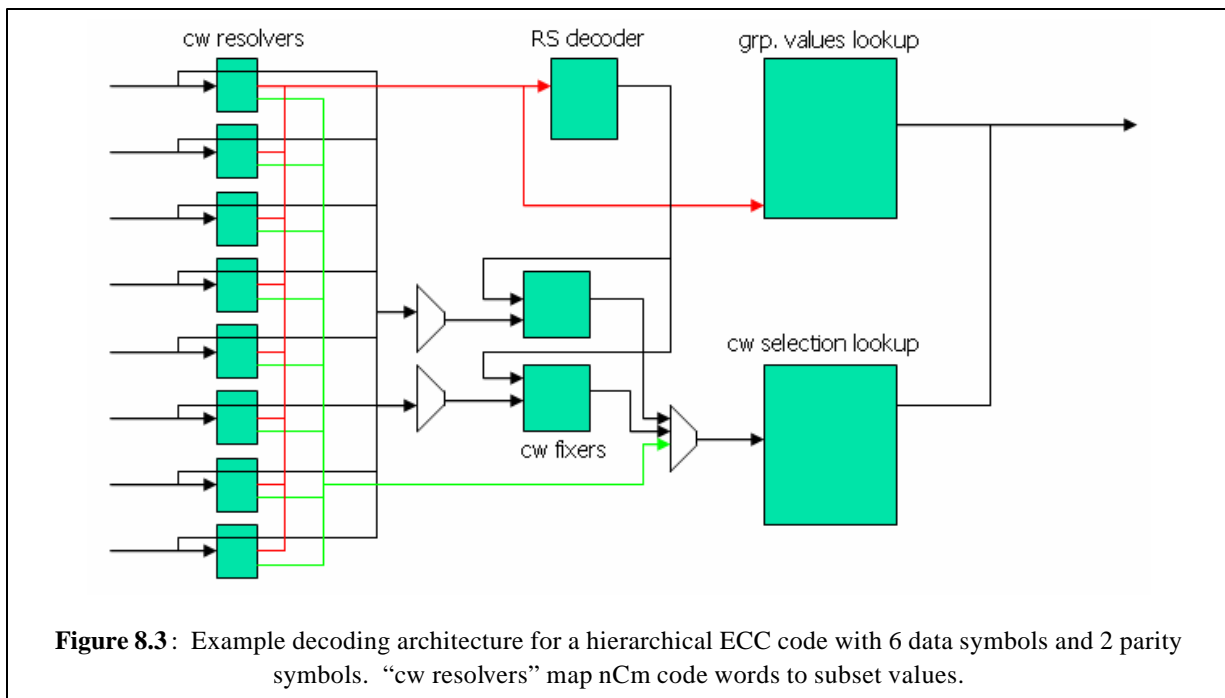
8.7. Required Hardware for Hierarchical ECC Implementation

The transmitting chip is required to translate a segment of its raw transmission data into a sequence of k subset values. This may be achieved using lookup tables or an array of small base- s adders and multipliers. Next, $n-k$ parity symbol(s) must be generated based on this resolution. This may be achieved using standard ECC hardware implementations. Next, it must resolve the remainder of its data into a sequence of n nCm code word choices for each block code symbol, including data symbols and parity symbols. This may also be achieved using a sequence of lookup tables.

The receiver must be capable of detecting both erasures (for invalid code words) and symbol errors (for block code decoding traps) on the incoming nC_m code words. In the event of an erasure or symbol error, the receiver must determine the corrected subset value(s) of any invalid code word(s) using the symbolic ECC mechanism. In this case, the receiver must be able to compute the corrected code word based on the received code word and the corresponding subset value. This may be accomplished using a minimum-distance decoder. Finally, the receiving chip must be able to convert the sequence of base- s and base- c values into a sequence of bits. This may also be achieved using lookup tables or an array of small adders and multipliers. An example architecture for performing decoding is shown in Figure 8.3.

8.8. Results for Hierarchical ECC Mechanisms

In this study, we have chosen parameters for several different hierarchical ECC codes. The purpose of this analysis is to determine a balance between code rate and error correction

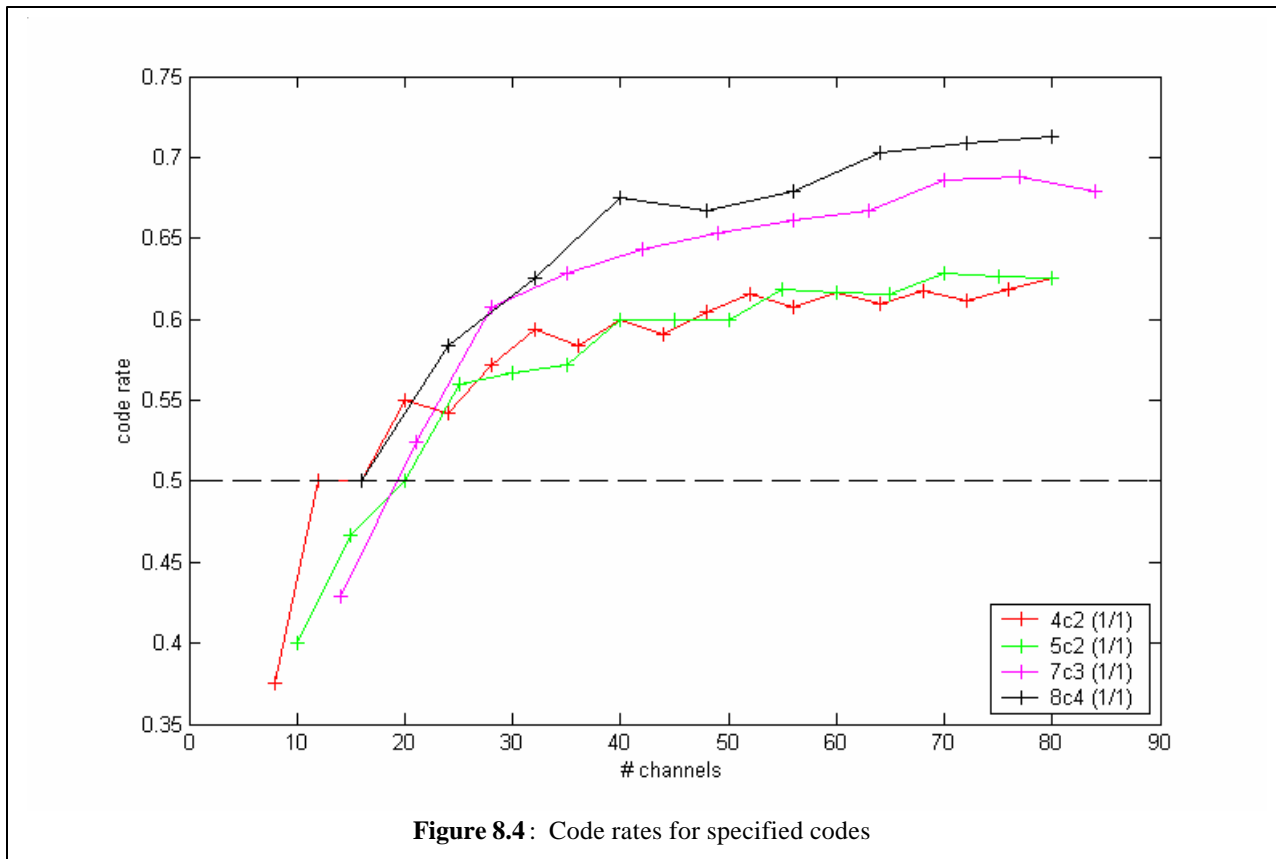


capability. In general, our goal is to build codes that match or exceed the code rate of a differential link (50%).

In the first experiment, we have built several types of codes which have the ability to correct any single bit error in any single code symbol. In each case, the nCm code sets were partitioned into subsets with $d_{symbol}=4$, such that any one bit error within an nCm code word may be corrected with 100% probability. Checksum is used as the block code, such that one parity symbol is used to yield a total symbol distance of 2 and an erasure correction ability of 1. We computed the code rate of these codes versus the number of channels required for the code block. Figure 8.4 is a plot of the results. A dotted line shows the code rate of a pure differential code with no ECC capability. Table 8.4 shows these same results for situations where the code rate is near 50%.

nCm	subsets	cw per subset	n	k	bits	wires	code rate
4c2	3	2	3	2	6	12	50.00%
4c2	3	2	5	4	11	20	55.00%
5c2	5	2	4	3	10	20	50.00%
5c2	5	2	5	4	14	25	56.00%
7c3	7	5	3	2	11	21	52.38%
7c3	7	5	4	3	17	28	60.71%
8c4	10	7	2	1	8	16	50.00%
8c4	10	7	3	2	14	24	58.33%

Table 8.4: Selected results for specified codes



In a second experiment, we have built several types of codes which have the ability to correct multiple bit errors. In some of these codes, multiple bits can be corrected within a single symbol. In others, single bits can be corrected in multiple symbols.

In the codes that use 5c2 and 7c3 symbols, a symbolic MDS block code with two parity symbols is used until the point at which the number of symbols exceeds the limit for an MDS code, given the code set partitioning. At this point, the code is switched to an equivalent AMDS code by adding a third parity symbol. The code built from 5c2 drivers uses base-5 symbols, so the MDS block limit is 6, or 30 channels. The code built from 7c3 drivers uses base-7 symbols, so the MDS block limit is 8 symbols, or 56 channels. In both cases, the nCm code sets are partitioned

into subsets whose distance is 4, while the symbolic code distance is 3. Under this arrangement, one physical bit error can be corrected in any two symbols.

In the codes that use $6c3$ and $8c4$ symbols, a checksum code is used as the symbolic code while the subsets have distances of 6 and 8, respectively. In the code formed using $6c3$ drivers, one bit error can be corrected in any single symbol. Two bit errors may be corrected in a single symbol if the result of the errors is not a valid nCm code word. Given any $6c3$ code word, there are $C(6,1) * C(6,1)$ possible valid code words that may result from a 2-bit error, while there are $C(6,2)$ total possible words that may result from a 2-bit error. This yields a 40% probability that any 2-bit error will result in an invalid code word and thus may be corrected as an erasure. Likewise, in the code formed using $8c3$ drivers, one or three bit errors can be corrected in any single symbol and two bit errors may be corrected with 43% probability.

We computed the code rate of these codes versus the number of channels required for the code block. Figure 8.5 is a plot of the results. A dotted line shows the code rate of a pure differential code with no ECC capability. Table 8.5 shows these same results for situations where the code rate is near 50%.

In a third experiment, we have built several types of codes in which the product of the number of subsets and the subset size yields a value less than the total number of code words in the nCm code set. In each of these cases, one parity value is used, which allows for the correction of a single bit error within a single symbol. Figure 8.6 shows the results of this experiment. Selected results are given in Table 8.6, where the number of total channels is ≤ 30 .

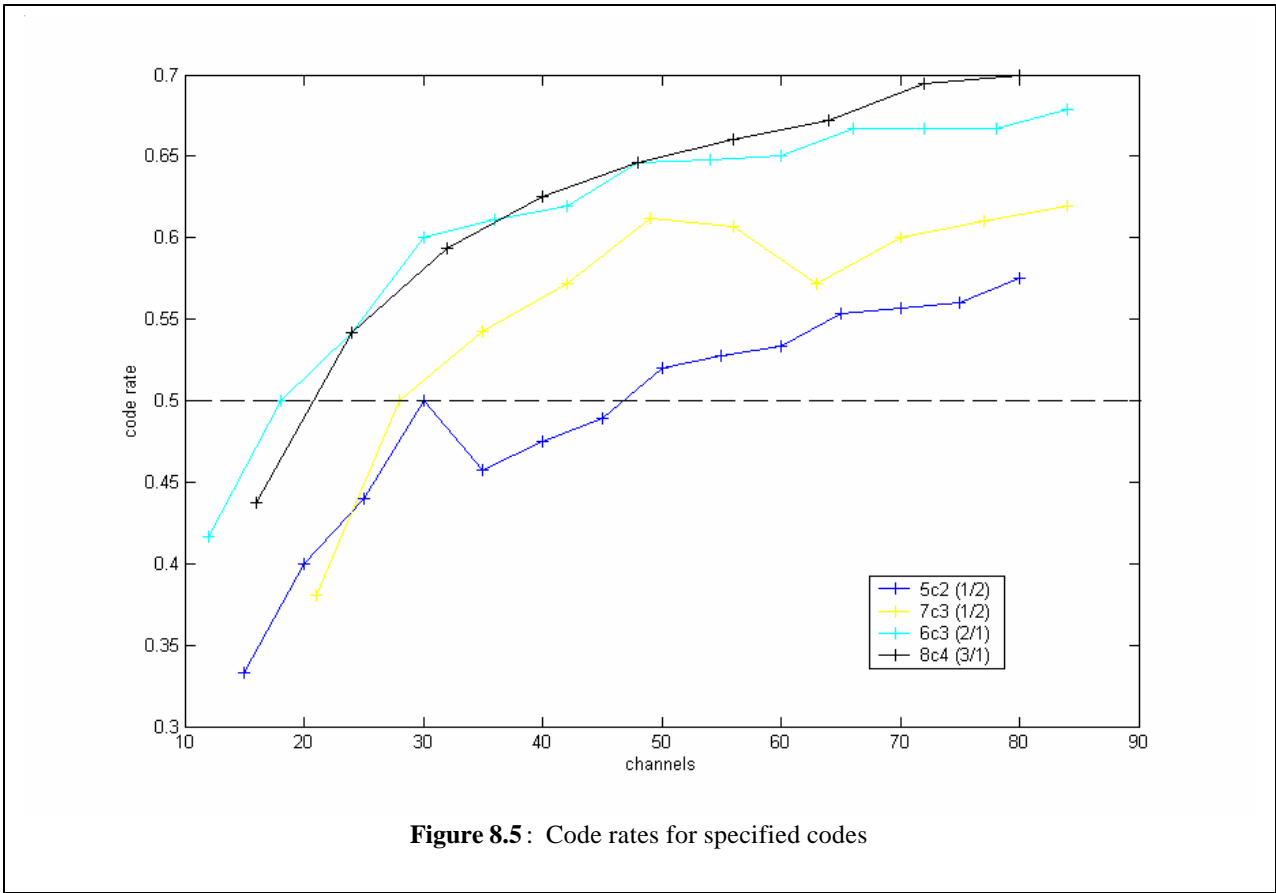


Figure 8.5: Code rates for specified codes

nCm	subsets	cw per subset	n	k	bits	wires	code rate
5c2	5	2	6	4	15	30	50.00%
7c3	7	5	4	2	14	28	50.00%
6c3	10	2	3	2	9	18	50.00%
8c4	35	2	3	2	13	24	54.17%

Table 8.5: Selected results for specified codes

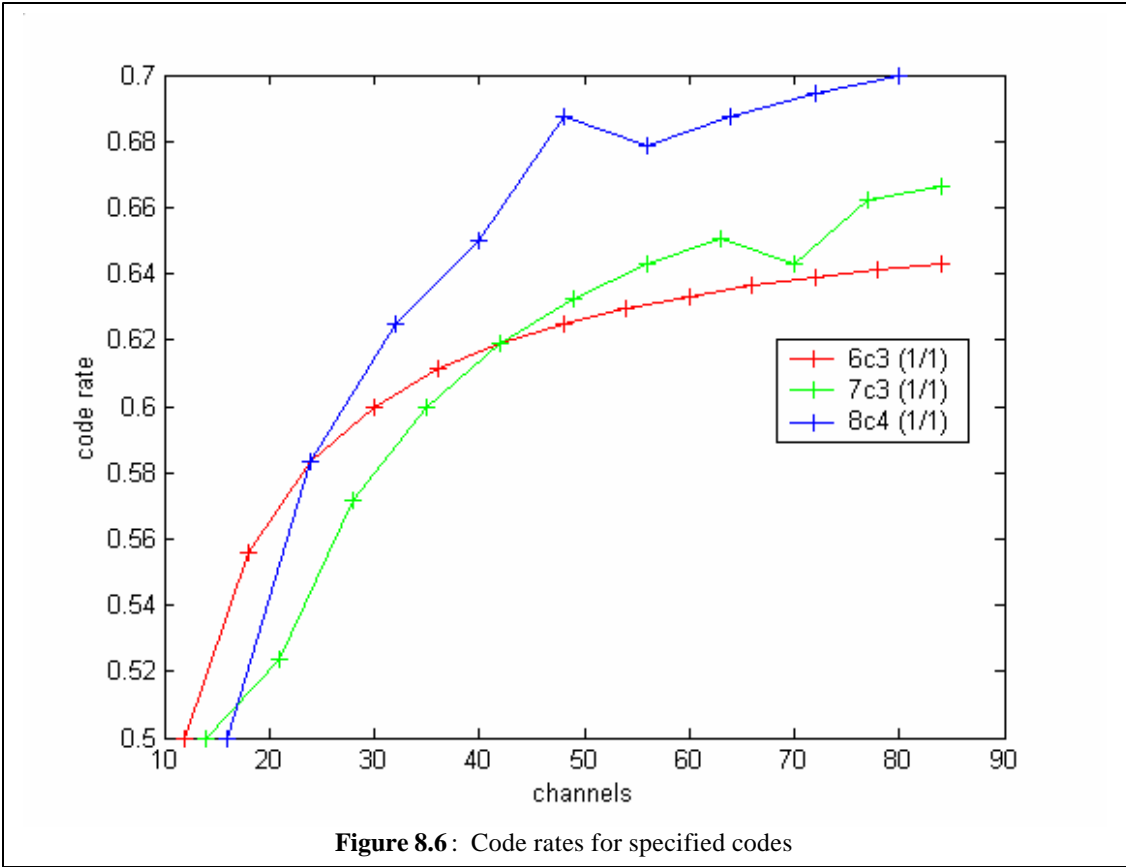


Figure 8.6: Code rates for specified codes

nCm	subsets	cw per subset	n	k	bits	wires	code rate
6c3	4	4	5	4	18	30	60.00%
7c3	5	6	4	3	16	28	57.14%
8c4	7	9	3	2	14	24	58.33%

Table 8.6: Selected results for specified codes

9. Research Plan

Our research plan involves three primary areas. First, we must determine if the hierarchical error control codes (LHECC) developed in this work are optimal relative to code rate when applied over Multi-Bit Differential Signaling (MBDS) channels. In other words, we must guarantee that there exists no other code that can add support for error control over MBDS channels and require less code rate overhead. Our LHECC codes are based on maximum distance separable (MDS) codes, which have been proven optimal relative to code rate for traditional binary channels. However, our codes use embedded error detection capability of the underlying channel code of MBDS in order to achieve higher code density relative to a direct implementation of MDS codes to MBDS channels. Therefore our first task is to verify that LHECC codes achieve the highest possible code rate relative to any other possible implementation of ECC over MBDS.

Our second task is to design an efficient architecture for encoding and decoding LHECC codes. Such encoding and decoding logic must be capable of operating as fast as the off-chip I/O frequency of the MBDS channels for which the code is used. In addition, this logic must be compact enough that it can reasonably fit into the real estate reserved for chip I/O (usually referred to as the *pad frame*). Such an architecture may be optimized by utilizing variability in the LHECC code, such as how the nCm code set is partitioned. In chapter 8 we provide an overview of a simple decoding architecture for an LHECC decoder. There are several issues that must be resolved, such as efficient mechanisms for encoding and decoding symbols that are

based on values that are not a power of 2 and performing base conversion for relatively large values.

Once we have designed a suitable architecture for encoding and decoding our ECC codes, our next task is to develop realistic noise models for simulation in order to determine the effectiveness of our ECC implementation. In Chapter 7 we have shown simulation results from our prototype MBDS system in order to determine the nature of signal errors occurring within MBDS channels. In these simulations, we modeled three types of error sources. The first type of noise source was common-mode noise, modeled by injecting additive white Gaussian voltage to each transmission line. The second type of noise source was jitter, where signal errors appeared when the receivers sampled the signals around the optimal transmission line delay relative to the time of driver transmission. The third type of noise source was caused by increasing the signal frequency at the point where transmission line capacitance and inductance seriously distorted the signal. This analysis constitutes a somewhat incomplete and naïve model for signal errors. Our goal is to provide more accurate and complete models for signal errors in order to determine the types of errors that may occur within MDDBS channels.

In summary, we plan to perform the following tasks:

- determine if LHECC codes are optimal for MBDS channels,
- design an efficient encoding and decoding architecture, and
- develop improved noise models to demonstrate functionality of LHECC codes.

APPENDIX

Appendix A: Demonstrator Chip Design

In order to test several MBDS link configurations we have designed a prototype MBDS driver and receiver chip in IBM's SiGe 5HP process (using MOSFET-only transistors). 2-, 4-, 6-, and 8-channel driver and receiver designs are located on the periphery of the die. A micrograph of the die is shown in Figure A.1. In this section we discuss the design of the prototype chip and test results.

Driver Design

We laid out 2, 4, 6, and 8 – channel MBDS current-steering drivers in this design, using standard NFET and PFET cells from the IBM SiGe 5HP standard cell library. The 4-channel driver design corresponds to the design shown in Figure 3.3. 6- and 8- channel drivers were designed as scaled-up versions of the 4

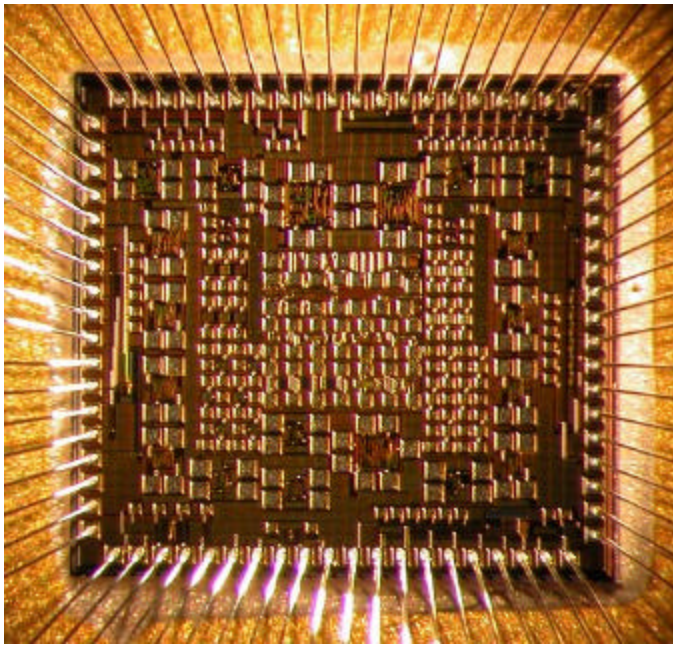


Figure A.1: Micrograph of SiGe test die

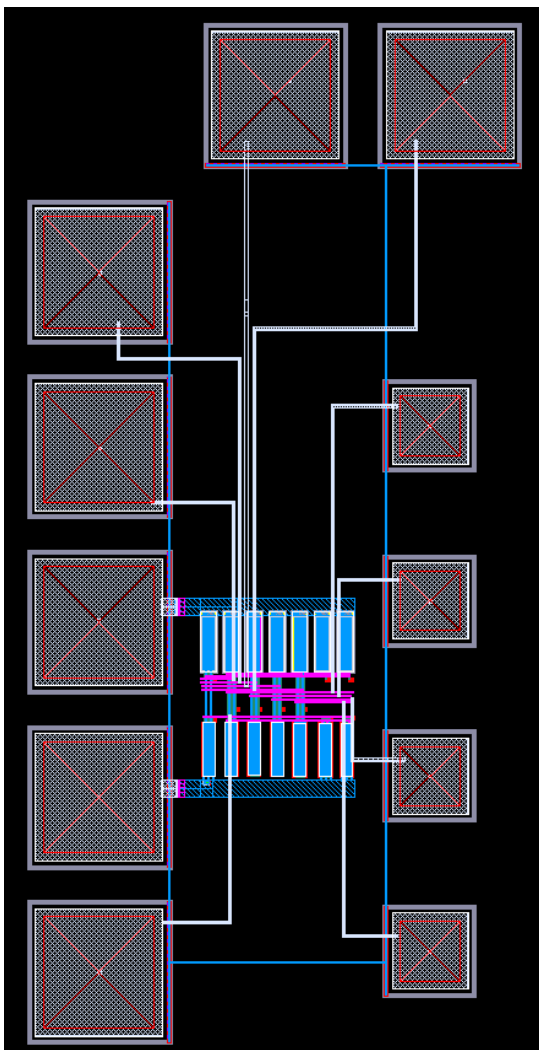


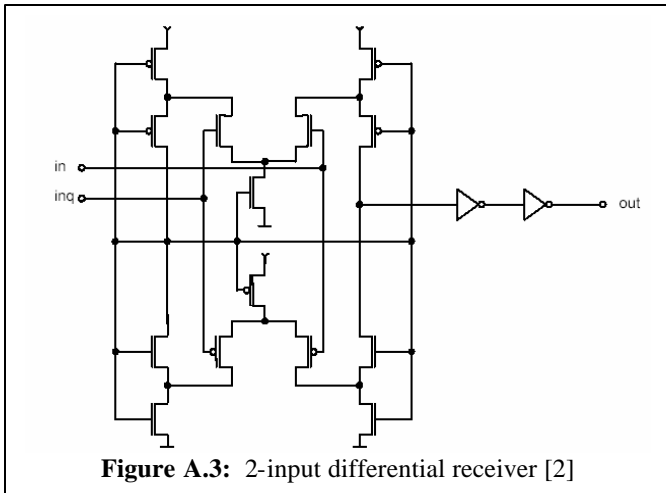
Figure A.2: Layout of 4-channel driver cell

channel driver by adding additional current-steering “legs” for each additional output. In each driver design, we added one biasing transistor for every two outputs. In these designs, the goal was to size the transistors such that, for any channel in the “on” state, the corresponding driver leg must be capable of sourcing up to 10 mA of current through a 100 ohm equivalent termination relative to ground at 10 GHz (assuming full biasing voltage). Initial simulation results indicated that this goal may be achieved if the P-transistors had $W=180$ μm assuming the minimum channel length of $L=0.5$ μm . Thus, all PFETs were designed to be 4-fingered transistors with each finger using a width of 45 μm . Simulation results also suggested that a balanced rise/fall time was reached when the NFETs are sized at half the width of the PFETs. Thus, the NFETs were sized at 90 μm (2 fingers, 45 μm each). These transistor sizes are consistent throughout the cells, including biasing transistors.

The drivers are physically situated on the periphery of the die, as close to the wirebond pads as possible. The outputs of the driver are connected to standard 110 μm square wirebond pad cells. Power and ground for each driver is brought in from wirebond pads as well. The 6- and 8-channel drivers have two power and ground connections each. The CMOS-level digital inputs to the drivers are provided by 65 μm square probe pads, which are located within the driver cells. Figure A.2 shows the layout for the 4-channel driver.

Receiver Design

The schematic for the differential receiver is shown in Figure A.3. The receiver conforms to the LVDS standard [2]. We designed each nCm receiver using an array of n 2-input LVDS receivers shown in Figure A.3. When building an MBDS link, each of the 2-input receivers has one input



connected to the common point of the termination network and the other input is connected to each channel's transmission line.

The transistor sizes for the receiver match those of the driver designs. 2-, 4-, 6-, and 8-channel receivers were designed by instancing one

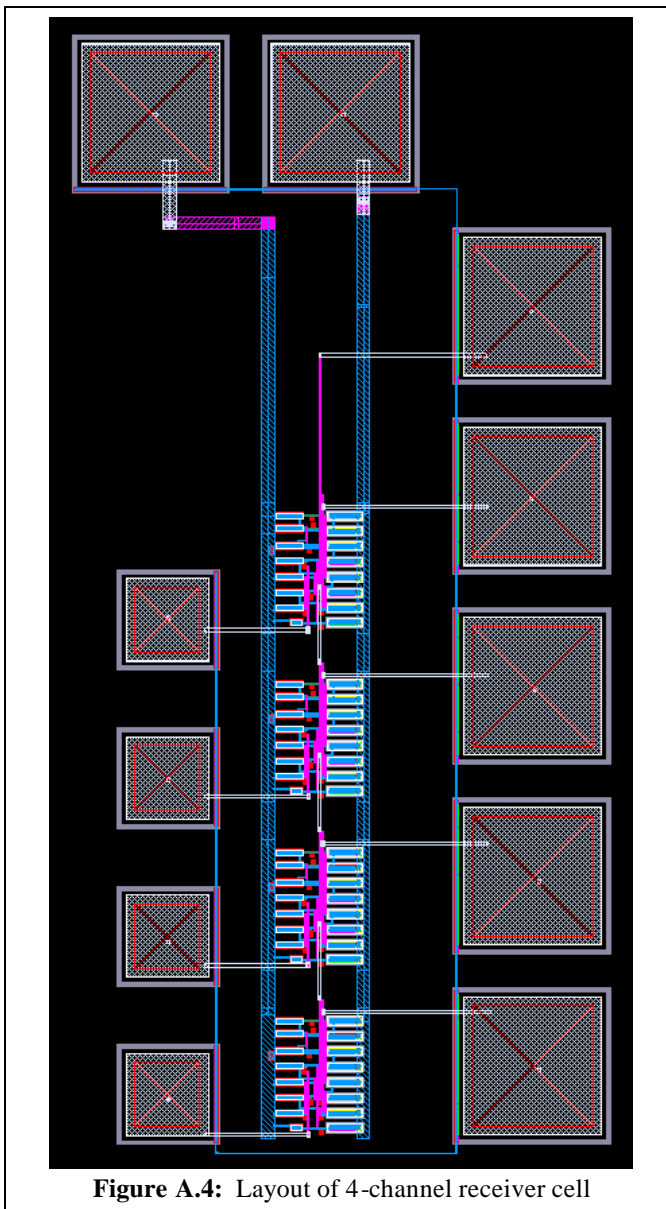
differential receiver cell for each channel. The receiver cells amplify the differential signal created between any particular channel and its

terminator network common point to CMOS levels. According to the LVDS standard, the differential voltage of the input may be as low as 100 mV [2].

The inputs to the receivers (including the channels and common point connection) are formed using the standard 110 um square wirebond pad cells, aligned on the periphery of the die.

As with the drivers, power and ground are also connected to the wirebond pads. The 6- and 8- channel receivers have two power and ground connections.

The CMOS-level outputs of the receivers are sampled using



65 um square probe pads. The layout for the 4-channel receiver is shown in Figure A.4.

The main disadvantage of this receiver design is that it is significantly slower than the driver designs. In future work, our goal is to test faster differential receiver designs with the MBDS drivers.

Test Setup and Results

Our test setup consists of several packaged test chips arranged in different link configurations implemented on a custom circuit board. In one configuration, a unidirectional link is created between the 4c2 and 8c4 drivers on an MBDS chip and off-the-shelf LVDS receiver chips. Other link configurations are created between two pairs of MBDS test chips by utilizing the drivers and receivers on each chip. The PCB microstrips in each link were designed using unique 3-dimensional structures in the inner layers of the board. We did this to test the coupled transmission line behavior in each candidate structure.

The MBDS chips are packaged in a surface mount LCC84M package. The circuit board measures 9.5” x 6.5” and the test dies are arranged on each side of the board on the horizontal axis. This effectively makes the transmission lines approximately 8 inches each. An image of the test setup is shown in Figure A.5. A diagram of the test setup is shown in Figure A.6. We have performed three tests using a 4c2 link with this setup.

In the first test, we brought in a random sequence of 4c2 code words at 100 Mbps (per channel) through the driver input area pads on the die, using 50-ohm terminated probes (Picoprobe model 10) and a Tektronix DG2020 data generator. We measured each channel as well as the

termination common point with an active probe (Picoprobe model 12C). The signals were probed at the termination resistor surface pads. The termination network is made up of 75-ohm surface-mount (1210 package) resistors. The signals were sampled and recorded using a Tektronix 11420 digitizing scope. The results for this test are shown in Table A.1 as an eye plot for each channel. Each point in these plots is computed as the common point voltage subtracted from the channel voltage, as this is the effective input signal at the differential receiver in the link.

In the second test, we measured results from a link created between a 4c2 driver and a 4-input LVDS receiver chip (National Semiconductor DS90IV048A). The drivers were stimulated with random code words at 200 Mbps (per channel). The output signals from the LVDS receiver chip were measured through SMA connectors on the PCB, connected through cables to the same scope. The results for this test are shown in Table A.2. In this test, we were limited to 200 Mbps by the LVDS receivers.

In a third test, we performed the same experiment as in the previous test but instead of using the

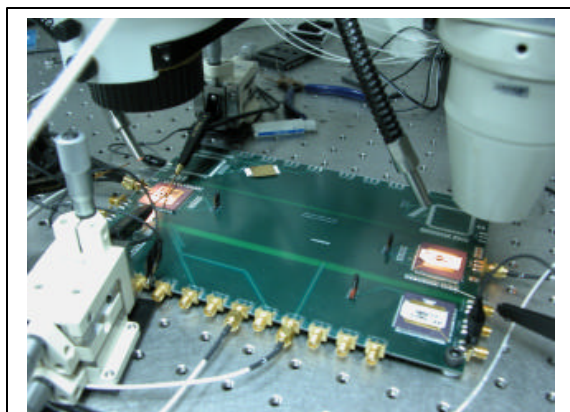
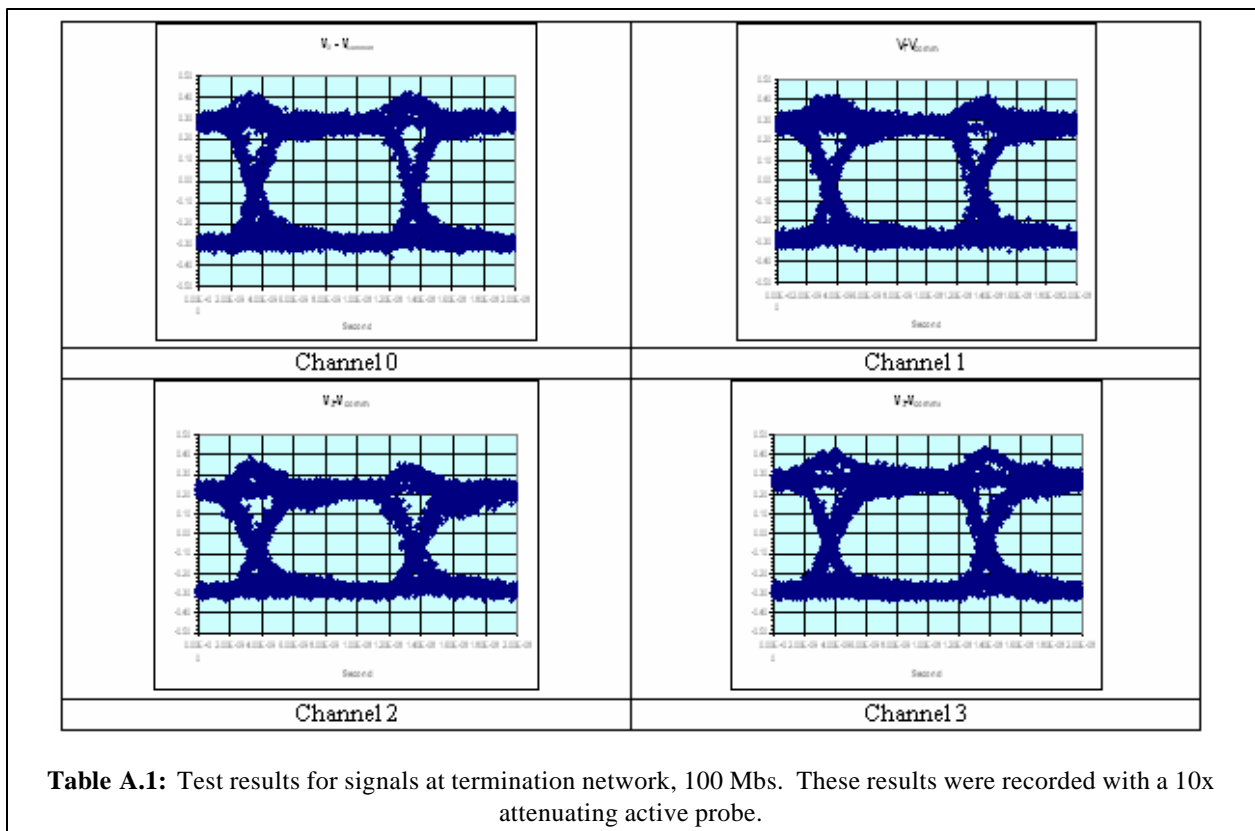
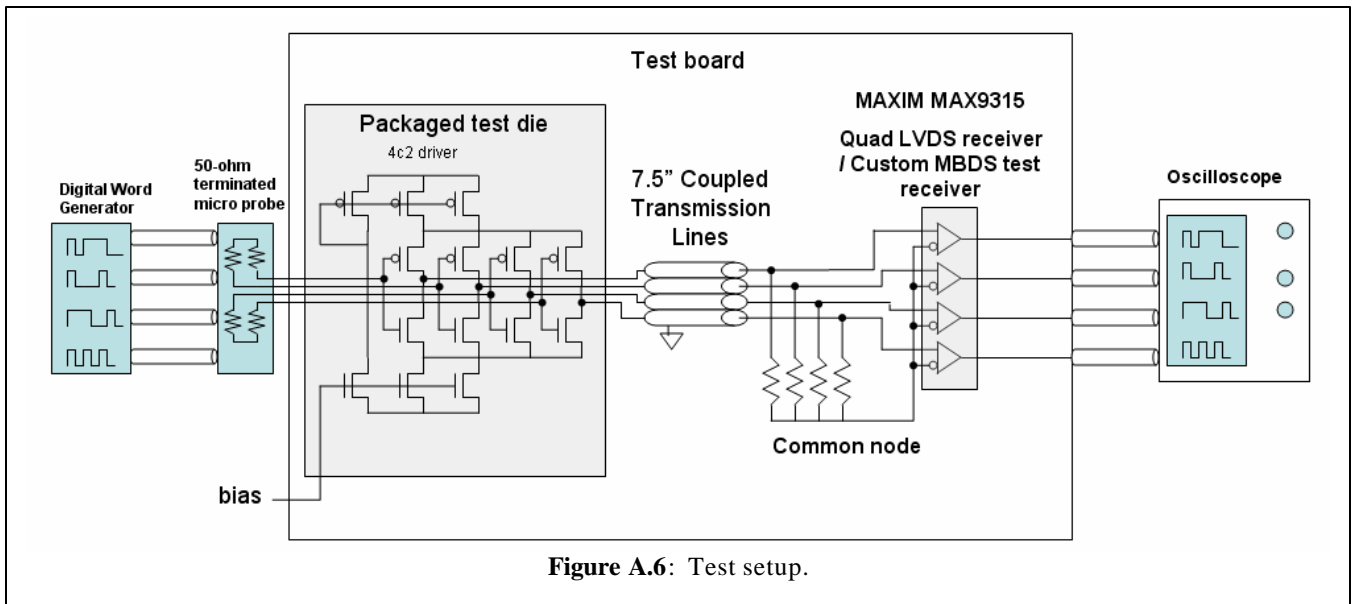


Figure A.5: Test setup

off-the-shelf LVDS receiver chip for signal reception, we use the LVDS receivers implemented on our test die. These receivers are capable of higher speeds than the National Semiconductor chips. The outputs of these receivers were sampled from probe pads on the die. These results are shown in Table A.3.



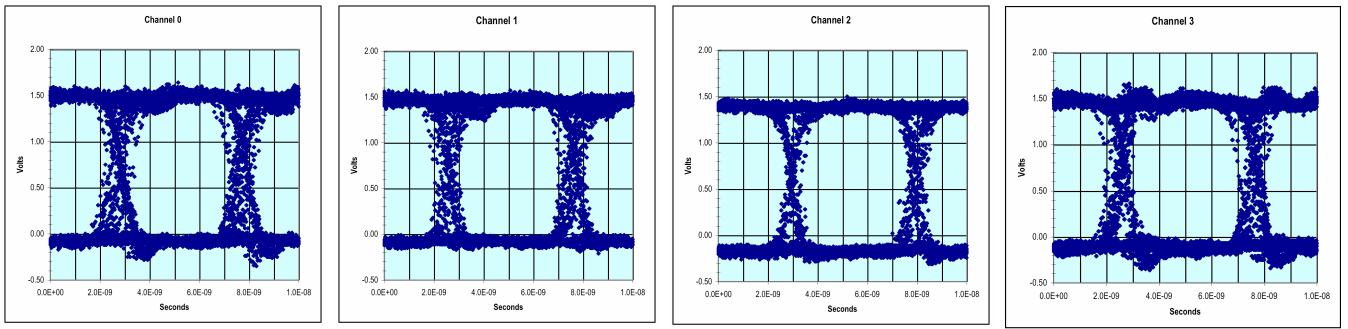


Table A.2: Receiver output of 4C2 MBDS link between test chip and commercial quad LVDS receiver running at 200Mbps.

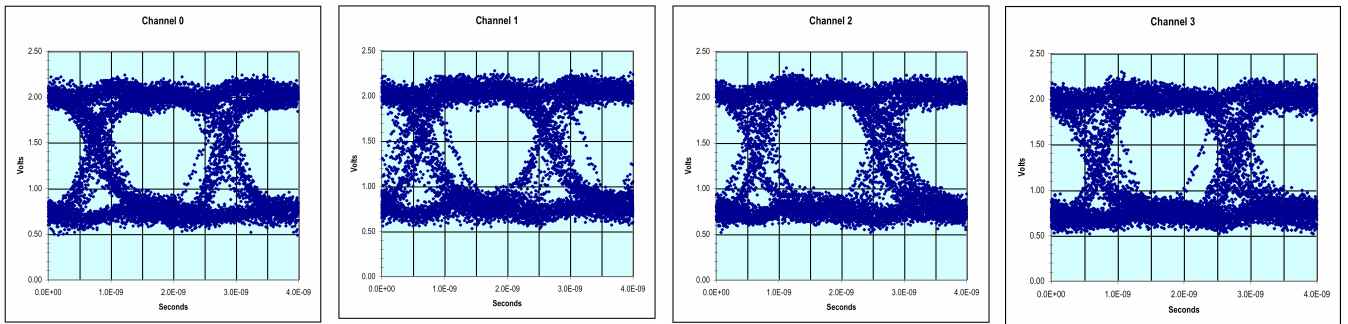


Table A.3: Receiver output of a second 4C2 test link using custom differential receivers running at 500Mbps.

BIBLIOGRAPHY

- [1] John Poulton, "*Problems and prospects for electrical signaling [VLSI]*," Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI, 21-24 March 1999, Page(s): 326.

- [2] Stefan Hirsch, Hans-Jörg Pfeleiderer, "*CMOS receiver circuits for high-speed data transmission according to LVDS-standard*," Proceedings of SPIE Vol. 5117 (2003).

- [3] M. Horowitz, C. K. Yang, S. Sidiropoulos, "*High-Speed Electrical Signaling: Overview and Limitations*," IEEE Micro, Jan/Feb 1998.

- [4] Nicholas Cravotta, "*RapidIO versus Hypertransport: A battle between equals or unintentional marketing confusion?*," EDN, June 27, 2002.

- [5] Ramin Farjad-Rod, Chih-Kong Ken Yang, Mark Horowitz, Thomas H. Lee, "*A 0.4-um CMOS 10-Gb/s 4-PAM Pre-Emphasis Serial Link Transmitter*," IEEE Journal of Solid-State Circuits, Vol. 34, No. 5, May 1999.

- [6] Evelina Young, Mark Horowitz, "*A 2.4 Gb/s/pin Simultaneous Bidirectional Parallel Link with Per-Pin Skew Compensation*," IEEE Journal of Solid-State Circuits, Vol. 35, No. 11, November 2000.

- [7] D. Cecchi, C. Hans, C. Preuss, "*A 2 GB/s high speed link with differential simultaneous bi-directional IO*," IEEE Conference on Custom Integrated Circuits, 6-9 May 2000, Pages: 505 – 508.

- [8] Intel, "*Intel Pentium 4 Processors on 90 nm Process Datasheet*," www.intel.com.

- [9] J. D. Bakos, D. M. Chiarulli, and S. P. Levitan, "*Optoelectronic Multi-Chip-Module Implementation of a 64 channel crossbar switch*," International Conference of Optics in Computing (OC2002) pp. 161-163, Taipei, Taiwan, April 8 - 11, 2002.
- [10] D. Chiarulli, S. Levitan, J. Bakos, "*Optoelectronic Multi-Chip Modules*," IEEE Mixed Design of Integrated Circuits and Systems, Poland 2003.
- [11] Jason Bakos, Donald Chiarulli, and Steven Levitan, "*Optoelectronic Multi-Chip Module Demonstrator System*" in Optics in Computing, OSA Technical Digest, (Optical Society of America, Washington DC, 2003) pp 117-119.
- [12] Donald Chiarulli, Jason Bakos, Leo Selavo, Steven Levitan, John Hansson, Michael Weisser, "*Photonic Packaging for Mixed-Technology Sensor Systems*," Integrated Photonics Research and Optics in Computing (IPR-OiC'2004), Engelberg, Switzerland, April 21-23, 2004.
- [13] Donald M. Chiarulli, Steven P. Levitan, Jason Bakos, Charles Kuznia, "*Active Substrates for Optoelectronic Interconnect*," IEEE International Symposium on Circuits and Systems, Vancouver, Canada, May 23, 2004.
- [14] Jun Ai and Yao Li, "*Polymer fiber-image-guide-based embedded optical circuit board*," Applied Optics, Vol. 38, No. 2, January 10, 1999.
- [15] Yao Li, Ting Wang, Hideo Kosaka, Shigeru Kawai, and Kenichi Kasahara, "*Fiber-image-guide-based bit-parallel optical interconnects*," Applied Optics, Vol. 35, No. 35, December 10, 1996.
- [16] Tomasz Maj, Andrew G. Kirk, David V. Plant, Joseph F. Ahadian, Clifton G. Fonstad, Kevin L. Lear, Karim Tatah, Matthew S. Robinson, and John A. Trezza, "*Interconnection of a two-dimensional array of vertical-cavity surface-emitting lasers to a receiver array by means of a fiber image guide*," Applied Optics, Vol. 39, No. 5, February 10, 2000.

- [17] Xuezhe Zheng, Philippe J. Marchand, Dawei Huang, and Sadik C. Esener, '*Free-space parallel multichip interconnection system*,' Applied Optics, Vol. 39, No. 20, July 10, 2000.
- [18] Michael W. Haney, Marc P. Christensen, Predrag Milojkovic, Jeremy Ekman, Premanand Chandramani, Richard Rozier, Fouad Kiamilev, Yue Liu, and Mary Hibbs-Brenner, "*Multichip free-space global optical interconnection demonstration with integrated arrays of vertical-cavity surface-emitting lasers and photodetectors*," Applied Optics, Vol. 38, No. 29, October 10, 1999.
- [19] S. Mick, J. Wilson, P. Franzon, "*4 Gbps high-density AC coupled interconnection*," Proceedings of the IEEE Custom Integrated Circuits Conference, May 12-15 2002, Pages:133 – 140.
- [20] S. Sidiropoulos, M. Horowitz, "*A 700-Mb/s/pin CMOS signaling interface using current integrating receivers*," IEEE Journal of Solid-State Circuits, Volume: 32, Issue: 5, May 5, 1997, Pages: 681 – 690.
- [21] B. Young, "*An SOI CMOS LVDS driver and receiver pair*," Digest of Technical Papers. 2001 Symposium on VLSI Circuits, 14-16 June 2001, Pages:153 – 154.
- [22] F. Devisch, J. Stiens, R. Vounckx, M. Kuijk, "*A power reduction method for off-chip interconnects*," The 2000 IEEE International Symposium on Circuits and Systems, 2000, Volume: 4, May 28-31, 2000, Pages:265 - 268 vol.4.
- [23] A. Faldum, W. Willems, "*Codes of Small Defect*," Designs, Codes, and Cryptography, 10, 341-350, 1997. Kluwer Academic Publishers, Boston.
- [24] Mario A. De Boer, "*Almost MDS Codes*," Designs, Codes, and Cryptography, 9, 143-155, 1996. Kluwer Academic Publishers, Boston.
- [25] L. Vanwassenhove, R. Bockstaele, R. Baets, M. Brunfaut, W. Meeus, J. Van Campenhout, J. Hall, H. Melchior, A. Neyer, J. Van Koetsem, R. King, K.Ebeling, P. Heremans, "*Demonstration of 2-D Plastic Optical Fibre based optical interconnect between CMOS*

- IC's," Optical Fiber Communication Conference and Exhibit, 2001. OFC 2001, Volume: 3, 2001, Pages: WDD74-1 - WDD74-3 vol.3.
- [26] H. Blennemann, Yao-Chao Yang; R. Nikel "Off-chip 400 Mbps signal transmission: noise reduction using non-resonant lengths and other techniques," Multi-Chip Module Conference, 1996. MCMC-96, Proceedings., 1996 IEEE, Feb. 6-7, 1996, Pages:138 – 142.
- [27] H. I. Hanafi, R. H. Dennard, C. -L. Chen, R. J. Weiss, D. S. Zicherman, "Design and characterization of a CMOS off-chip driver/receiver with reduced power-supply disturbance," IEEE Journal of Solid-State Circuits, Volume: 27, Issue: 5, May 1992, Pages: 783 – 791.
- [28] H. Zhang, V. George, J. M. Rabaey, "Low-swing on-chip signaling techniques: effectiveness and robustness," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume: 8 , Issue: 3, June 2000, Pages: 264 – 272.
- [29] K. Farzan, D. A. Johns, "Power efficient chip-to-chip signaling schemes," IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002., Volume: 2, May 26-29, 2002, Pages:II-560 - II-563 vol.2.
- [30] A. Bogliolo, "Encodings for high-performance energy-efficient signaling," International Symposium on Low Power Electronics and Design, 6-7 Aug 6-7, 2001, Pages:170 – 175.
- [31] H. Tamura, K. Gotoh, H. Araki, S. Wakayama, T. S. Cheung, M. Saito, J. Ogawa, Y. Kato, T. Nishi, M. Kawano, M. Taguchi, T. Imamura, "PRD-based global-mean-time signaling for high-speed chip-to-chip communications," Solid-State Circuits Conference, 1998. Digest of Technical Papers. 45th ISSCC 1998 IEEE International, Feb 5-7, 1998, Pages:164 - 165, 432.
- [32] Gadiel Seroussi, Ron M. Roth, "On MDS Extensions of Generalized Reed-Solomon Codes," IEEE Transactions of Information Theory, Vol IT-32, No. 3, May 1986, Pages: 349-354.

- [33] John Teifel, Rajit Manohar, "A *High-Speed Clockless Serial Link Transceiver*," Proceedings of the Ninth Symposium of Asynchronous Circuits and Systems ASYNC'03, IEEE 2003.
- [34] B. W. Langley, R. F. W. Pease, "Superconducting Interconnects for VLSI Multi-Chip System Integration," Symposium on VLSI Technology, IEEE 1990.
- [35] Rambus Corp., "Gigahertz Rambus Signaling Technologies: RSL, QRSL, and SerDes Technology Overview," <http://www.rambus.com>.
- [36] Randy Mooney (Intel Corp.), "Scaling Methods for Electrical Interconnects to Meet the Performance Requirements of Microprocessor Platforms," Workshop Notes, IEEE 14th Annual Workshop on Interconnects Within High-Speed Digital Systems, May 4-7, 2003.
- [37] C. E. Shannon, "A *Mathematical Theory of Communications*," Bell Syst. Tech. J. 27, pp. 379-423 (Part I), 623-656 (Part II), July 1948.
- [38] E. R. Berlekamp, "Algebraic Coding Theory," McGraw-Hill, New York, 1968.
- [39] W. W. Peterson, E. J. Weldon, Jr., "Error-Correcting Codes," 2nd Edition, MIT Press, Cambridge, Mass., 1972.
- [40] F. J. MacWilliams, J. J. A. Sloane, "The Theory of Error-Correcting Codes," North-Holland, Amsterdam, 1977.
- [41] R. J. McEliece, "The Theory of Information and Coding," Addison-Wesley, Reading, Mass., 1977.
- [42] F. J. MacWilliams, "A Theory on the Distribution of Weights in a Systematic Code," Bell Sys. Tech. J., 42, pp. 79-94, 1963.

- [43] R. W. Hamming, "*Error Detecting and Error Correcting Codes*," Bell Sys. Tech. J., 29, pp. 147-160, April 1950.
- [44] S. K. Leung-Yan-Cheong, M. E. Hellman, "*Concerning a Bound on Undetected Error Probability*," IEEE Transactions on Information Theory, IT-22, pp. 235-237, March 1976.
- [45] E. Prange, "*Cyclic Error-Correcting Codes in Two Symbols*," AFCRC-TN-57, 103, Air Force Cambridge Research Center, Cambridge, Mass., September 1957.
- [46] T. Kasami, "*A Decoding Method for Multiple-Error-Correcting Cyclic Codes by using Threshold Logics*," Conv. Rec. Inf. Process. Soc. Jap. (in Japanese), Tokyo, November 1961.
- [47] M. E. Mitchell et al, "*Coding and Decoding Operation Research*," G. E. Advanced Electronics Final Report on Contract AF 19 (604)-6183, Air Force Cambridge Research Labs., Cambridge, Mass., 1961.
- [48] M. E. Mitchell, "*Error-Trap Decoding of Cyclic Codes*," G. R. Report No. 62MCD3, G. E. Military Communications Dept., Oklahoma City, Okla., December 1962.
- [49] L. Rudolph, "*Easily Implemented Error-Correction Encoding-Decoding*," G. E. Report No. 62MCD2, G. E. Corporation, Oklahoma City, Okla., December 1962.
- [50] T. Kasami, "*A Decoding Procedure For Multiple-Error-Correction Codes*," IEEE Transactions on Information Theory, IT-10, pp. 134-139, April 1964.
- [51] A. Hocquenghem, "*Codes correcteurs d'erreurs*," Chiffres, 2, pp. 147-156, 1959.
- [52] R. C. Bose, D. K. Ray-Chaudhuri, "*On a Class of Error Correcting Binary Group Codes*," Information Control, 3, pp. 68-69, March 1960.

- [53] W. W. Peterson, "Encoding and Error-Correction Procedures for the Bose-Chaudhuri-Codes," IRE Transactions on Information Theory, IT-6, pp. 459-470, September 1960.
- [54] D. Gorenstein, N. Zierler, "A Class of Cyclic Linear Error-Correcting Codes in p^m Symbols," J. Soc. Ind. Appl. Math., 9, pp. 107-214, June 1961.
- [55] R. T. Chien, "Cyclic Decoding Procedure for the Bose-Chaudhuri-Hocquenghen Codes," IEEE Transactions on Information Theory, IT-10, pp 357-363, October 1965.
- [56] G. D. Forney, "On Decoding BCH Codes," IEEE Transactions on Information Theory, IT-11, pp. 549-557, October 1965.
- [57] E. R. Berlekamp, "On Decoding Bose-Chaudhuri-Hocquenghen Codes," IEEE Transactions on Information Theory, IT-11, pp. 577-580, October 1965.
- [58] E. R. Berlekamp, "Algebraic Coding Theory," McGraw-Hill, New York, 1968.
- [59] J. L. Massey, "Step-by-Step Decoding of Bose-Chaudhuri-Hocquenghen Codes," IEEE Transactions on Information Theory, IT-11, pp. 580-585, October 1965.
- [60] J. L. Massey, "Shift-Register Synthesis and BCH Decoding," IEEE Transactions on Information Theory, IT-15, pp.122-127, January 1969.
- [61] H. O. Burton, "Inversionless Decoding of Binary BCH Codes," IEEE Transactions on Information Theory, IT-17, pp. 464-466, July 1971.
- [62] I. S. Reed, G. Solomon, "Polynomial Codes over Certain Finite Fields," J. Soc. Ind. Appl. Math., 8, pp. 300-304, June 1960.

- [63] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Efficient Erasure Correcting Codes," IEEE Transactions on Information Theory, Vol. 47, No. 2, February 2001.
- [64] R. M. Roth, G. Seroussi, "Location-Correcting Codes," IEEE International Symposium on Information Theory, Trondheim, Norway, June 1994.
- [65] G. D. Forney, Jr., "Concatenated Codes," MIT Press, Cambridge, Mass., 1966.
- [66] T. V. Ramabadran, "A Coding Scheme for m -out-of- n Codes," IEEE Transactions on Communications, Vol. 38, No. 8, August 1990, pp. 1156-1163.
- [67] A. Burr, "Turbo-Codes: the ultimate error control codes?," IEEE Electronics and Communication Engineering Journal, August 2001, pp. 155-165.
- [68] T. H. Szymanski, "Optical Link Optimization Using Embedded Forward Error Correcting Codes," IEEE Journal of Selected Topics in Quantum Electronics, Vol. 9, No. 2, March/April 2003, pp. 647-656.
- [69] W. F. Chang, C. W. Wu, "Low-Cost Modular Totally Self-Checking Checker Design for m -out-of- n Code," IEEE Transactions on Computers, Vol. 48, No. 8, August 1999, pp. 815-826.
- [70] C. Bolchini, F. Salice, D. Sciuto, "Conditions for the Design of Circuits with Concurrent Error Detection Properties," 1997 IEEE International Symposium on Circuits and Systems, June 9-12, 1997, Hong Kong, pp. 2741-2744.
- [71] J. C. Lo, "A Hyper Optimal Encoding Scheme for Self-Checking Circuits," IEEE Transactions on Computers, Vol. 45, No. 9, September 1996, pp. 1022-1030.

- [72] A. J. Goor, M. S. Abadir, A. Carlin, “*Minimal Test for Coupling Faults in Word-Oriented Memories*,” Proceedings of the 2002 Design, Automation, and Test in Europe Conference and Exhibition, (DATE’02).
- [73] X. M. Wang, Y. X. Yang, “*On the Undetected Error Probability on Nonlinear Binary Constant Weight Codes*,” IEEE Transactions on Communications, Vol. 42, No. 7, July 1994, pp. 2390-2394.
- [74] F. W. Fu, T. Klove, “*The Undetected Error Probability Threshold of m-out-of-n Codes*,” IEEE Transactions on Information Theory, Vol. 46, No. 4, July 2000, pp. 1597-1599.
- [75] Y. Shin, K. Choi, Y. H. Chang, “*Narrow Bus Encoding for Low-Power DSP Systems*,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9, No. 5, October 2001.
- [76] M. Stan, W. P. Burleson, “*Low-Power Encodings for Global Communication in CMOS VLSI*,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 5, No. 4, December 1997.
- [77] <http://en.wikipedia.org/wiki/LVDS>.
- [78] M. N. O. Sadiku, “*Elements of Electromagnetics Third Edition*,” Oxford University Press.
- [79] P. Elias, “*Coding for Noisy Channels*,” IRE Conv. Rec., Part 4, pp. 37-47, 1955.
- [80] A. J. Viterbi, “*Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*,” IEEE Transactions on Information Theory, IT-13, pp. 260-269, April 1967.