

# Hierarchical Error Correction Codes over Multi-Bit-Differential Signaling

Jason D. Bakos  
Department of Computer Science  
University of Pittsburgh

## Abstract

We present a mechanism for implementation of error control coding over a new chip-to-chip signaling technology called multi-bit differential signaling (MBDS). MBDS is a low-power, area and pin efficient alternative to serial differential links, and consists of a new driver and link termination network design coupled with a novel coding system based on “*n* choose *m*” ( $nCm$ ) codes. Our error handling coding technique takes advantage of the natural redundancy and additional code space that  $nCm$  codes offer.

## 1. Introduction

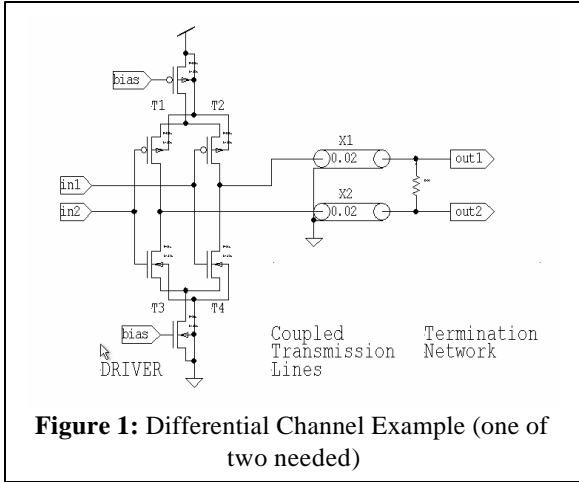
In CMOS digital and mixed-signal technology, there is a growing gap between on-chip data transfer speeds and chip-to-chip I/O throughput. Advancements in chip fabrication technology have allowed on-chip data rates to grow much faster than chip-to-chip signaling rates. The solution to this problem lies in high-performance chip-to-chip signaling. There are several trends that drive this technology. First, the physical properties of chip-to-chip transmission lines cause signal integrity to degrade at high speeds. High-performance driver and receiver circuits seek to compensate for loss of signal integrity. Second, because I/O pads are a critical resource in chip design, high-speed serial links are sought to conserve chip I/O resources. Third, data encoding that is tightly coupled with a particular link technology is emerging as a tool for increasing signal integrity and bandwidth over chip-to-chip links. In sections 2 and 3 we present an overview of MBDS signaling technology. In section 4 we discuss an extension to  $nCm$  coding that adds channel error correction capability to the link.

## 2. Multi-Bit Differential Signaling (MBDS)

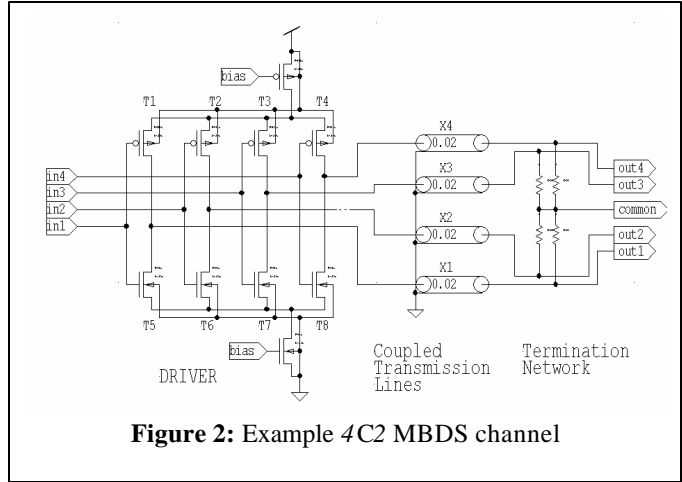
MBDS links share the high performance electrical attributes of differential signaling (signal-to-noise characteristics) while supporting a much larger code set. This larger code set means that MBDS links have significantly higher throughput than differential links, which translates to  $\leq 40\%$  better power efficiency and  $1/3$  reduction in pad count versus differential links. The key components to MBDS links are the following: a new driver circuit design, termination network design, and data encoding method. In the new system, we replace the familiar two wire per bit format of differential signaling with an  $n$ -channel system in which all of the code symbols in the link must conform to an “*n* choose *m*” (written  $nCm$ ) encoding rule. In an  $nCm$  code, each  $n$ -bit symbol encoding must have exactly  $m$  1-bits and  $n-m$  0-bits. We refer to this method as multiple-bit differential signaling based on the constant number of 1-bits in the channel at all times. For example, a 4 choose 2 ( $4C2$ ) channel uses a four-wire interconnect such that at any time exactly two of the wires will be energized with 1-bits. When compared to the same four wires configured as two differential 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 2.5 bits of throughput for the  $4C2$  channel versus 2 bits for the differential channels.

Now consider the two driver circuits and example channels shown in Figure 1 and Figure 2. Figure 1 shows a model of a differential link. Figure 2 is a model of a  $4C2$  link. Both consist of a driver circuit, a set of coupled transmission lines, and a termination network between outputs of the transmission lines. Both of the driver circuits operate by steering a constant supply current between legs of the circuit and thus through paths in the termination network. All of the resistors in the termination network have the same value and are sized to match a 100 ohm load to the transmission line. Since every code symbol is encoded with exactly two 1-bits, the common node in the center of the termination network has a constant voltage. The outputs are sensed across each resistor between the transmission line output and this common mode reference. Like the differential case, the data is sensed as the voltage polarity across this resistor and thus the two circuits can use the same type of receivers. This comparison is important because it demonstrates that MDDBS channels have substantial encoding advantages yet they retain the electrical characteristics of a differential channel. These characteristics are specifically highlighted below.

- **Better power efficiency** can be achieved since  $nCm$  code symbols are encoded with fewer ‘1’ bits used to send an equivalent amount of information.
- **Less silicon area and lower pad/wire count** is required for an  $nCm$  channel than an equivalent set of differential channels.



**Figure 1:** Differential Channel Example (one of two needed)



**Figure 2:** Example 4C2 MBDS channel

- **Higher Effective Bandwidth** is available because more information is delivered to the receiver per symbol. The larger symbol set of an  $nCm$  channel means that each code word imparts more information per symbol received.
- **Low noise in the driver circuit** comes from the current steering design of the driver that operates with constant current in all code states.
- **Coupled transmission line behavior** will occur assuming that the link transmission lines or printed circuit board are properly designed and the signals are routed in parallel and in close proximity. The embedded set of transmission lines will be electromagnetically coupled resulting in lower loss and greater signal integrity.
- **Common mode noise rejection in a pair-wise differential receiver** is achieved because the link output signal as delivered to the receiver is a relative voltage drop across the termination resistor network. The receiver sees each encoding state as a change in the polarity of this voltage. Common mode noise appears identically on both sides of the resistor in either state. Thus, it is cancelled by differential receiver circuitry, substantially enhancing the signal to noise ratio of the output.

### 3. $nCm$ Coding Fundamentals

In this section we present some basic combinatorial relationships that govern  $nCm$  encoding. We will also show how these relationships affect the power and area requirements of the required encoding for any particular MBDS link. Consider the set  $X$  such that  $X_{nm} = \{x_{nm} : x \in nCm\}$ . In other words,  $X_{nm}$  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:

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

For any value of  $n$ , the number of  $nCm$  code symbols is maximum when  $m$  is equal to  $n/2$ , rounded either up or down if the value of  $n$  is odd. Regardless of the number of code symbols for the channel, each code symbol 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, the effective bit width,  $bit_{eff}$ , the number of bits coming into and out of the channel before

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

**Table 1:** Comparison MBDS channels to equivalent single ended channels

encoding and after decoding is given by:  $bit_{eff} = \text{floor}(\log_2(\mathbf{f}\{X_{mn}\}))$ .

Using effective bit width as a metric, Table 1 compares the relative power consumption, pad count, and code utilization for several MBDS channel configurations to a set of differential channels with an equivalent effective bit width. Specifically, 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  $nCm$  channel, to  $bit_{eff}$ , the number of wires energized to '1' in a differential link required to send the same information. The relative pad count is computed as  $RP = n / (2 * bit_{eff})$ , the ratio of  $n$ , the number of wires in the  $nCm$  channel to  $2 * bit_{eff}$ , the number of differential channels required to send the same information times two wires per channel. 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 two properties that make it suitable for the addition of a channel error correction mechanism. First,  $nCm$  code sets have a built-in natural redundancy. 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  $n$ . This level of pre-existing redundancy is sufficient for single channel error detection. This redundancy can be used with a more advanced coding mechanism to provide more powerful channel error correction ability. Secondly,  $nCm$  code sets provide unused code words as a result of mapping binary values into  $nCm$  code words. These excess code words can also be used to offset any costs to code rate incurred by overlaying an error correction mechanism over  $nCm$  coding. The amount of excess code words for each  $nCm$  code set is shown in Table 1 under "% Code Utilization," which is computed as the largest power of two less than the number of code words divided by the number of codewords. The addition of error correction coding over MBDS signaling increases signal integrity, which ultimately increases the maximum bandwidth of the link.

#### 4. Application of Error Correcting Codes over MBDS Signaling

We have shown that, given a serial link of width  $n > 4$ ,  $nCm$  coding provides a higher code rate than differential coding while retaining similar electrical characteristics. However, the  $nCm$  code set provides two important properties that make it more suitable for the addition of a channel error correction mechanism than a traditional link. First,  $nCm$  code sets have a natural redundancy that provides a built-in level of error detection capacity. Secondly, after adding additional required redundancy to provide error correction capacity to an  $nCm$  link, the detrimental effect on the code rate may be softened by utilizing additional available codewords, which would otherwise go unused given a direct mapping between  $nCm$  code words to binary values. The built-in redundancy of an  $nCm$  code set is easily quantifiable. Given an  $nCm$  code set, any two codewords from the code set have a relative distance of at least 2. Given any code word in an  $nCm$  code set, flipping any bit will result in a word that no longer conforms to the "n choose m" rule because the resultant word will no longer have the appropriate number of 1-bits. This property provides the  $nCm$  code set enough redundancy to detect any single channel error with 100% probability. Now, assume a valid  $nCm$  codeword is transmitted across a link and multiple channel errors occur. The ability for this type of error to be detected by the receiver depends upon which bits in the codeword are flipped. If a multiple channel error occurs where an unequal number of the original 1-bits and original 0-bits are flipped, the resultant word will no longer conform to the "n choose m" rule and will result in an invalid codeword. This type of error can be detected by the receiver. However, if a multiple channel error occurs where an equal number of 1-bits and 0-bits are flipped, then the resultant word is a valid  $nCm$  codeword and the channel error may not be detected. The probability of detecting a multiple channel error is computed as  $P = 1 - m / ((m+1) * (n-m+1) - 3)$ . Extra  $nCm$  code words may be used to reduce ECC overhead. As the number of  $nCm$  drivers used in parallel in a link grows, the effective code rate increases. For example, a single 4C2 driver, which has 6 code words in its codeset, may carry  $\text{floor}(\log_2(6)) = 2$  bits. This yields a code rate of 2 bits for 4 channels, or 50%. However, if two 4C2 drivers are used in parallel, this is equivalent to  $\text{floor}(2 * \log_2(6)) = \text{floor}(\log_2(36)) = 5$  bits, for an overall code rate of 5 bits for 8 channels, or 62.5%.

A link that is formed using multiple  $nCm$  drivers encode  $b$  bits into  $c$  channels, where  $b < c$ . The channels are grouped in  $d$   $nCm$  drivers, which have  $n$  channels each. Therefore, a single *channel error* does not necessarily indicate a single *bit error*, because multiple bit errors may result from a single *channel error*. Each group of  $nCm$  drivers have independent common points, which means that any potential electrical phenomena that induces a channel error will occur locally to an  $nCm$  driver's common point. Therefore, our goal is to apply ECC coding that will operate over the granularity of individual  $nCm$  drivers and their associated termination networks. In this scenario, bit errors are potentially caused when, during the instant of sampling at the receiver, the relative voltage between any particular channel and its associated common point does not exceed the minimum threshold value for the channel's differential receiver. Signal conditioning on the output of the differential receivers will induce a stable CMOS value in this situation; however that value may constitute a channel error. The most common causes of this type of error are the following: jitter from clock extraction or board-level clock skew, transient transmission line capacitance, and random noise sources from within the driver, receiver, or transmission lines.

This situation lends itself to a hierarchical ECC mechanism, where a binary ECC is applied over the nCm code words and a non-binary, symbolic ECC mechanism is applied over a group of nCm code word symbols. This ECC mechanism will allow capacity for single or multiple channel correction within single or multiple nCm “symbols”. The generalized form of our hierarchical ECC mechanism operates as follows. A link is built using two or more nCm drivers, each having a consistent number of channels. The nCm code sets associated with these drivers is divided into equal-sized subsets. The distances of the code words in each subset will be a specified value,  $d_{symbol}$ . This means that for every possible pair of two code words within any of these subsets will have a relative distance of  $d_{symbol}$  or greater. The value chosen for  $d_{symbol}$  will determine the maximum number of channel errors that may be corrected within an nCm symbol. The number of correctible channels is defined in the traditional way for binary ECC mechanisms,  $t = \text{floor}((d_{symbol} - 1) / 2)$ . For example, for  $d_{symbol} = 4$ , a 7c3 code set, which has 35 code words, may be divided into 7 equal-sized subsets, each having 5 code words apiece. This partitioning will allow a single channel error to 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. The following is an example partitioning of the 7c3 code set:

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 2:** Partitioning of a 7c3 code set with  $d = 4$ .

These subsets are arbitrarily enumerated. An example enumeration is shown in the left-hand column of Table 2. 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  $(n,k,q)$  code, consisting of a block of  $n$  symbols, where  $k$  symbols carry data and  $n-k$  symbols carry parity. The code symbols are base- $q$  values, where  $q$  is the number of subsets in the nCm code set partitioning. An  $(n,k,q)$  block code implies that the symbolic code has the ability to correct  $e$  erasures and  $t$  symbol errors, where  $e$  and  $t$  are defined as functions of the symbolic code’s distance. Specifically, these functions are the following:

$$e = d - 1$$

$$t = \text{floor}((d - 1) / 2)$$

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  $n$  and  $k$ . This bound is defined as:

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

If single or multiple channel errors affect an nCm symbol in a link, the channel errors will manifest themselves in one of two ways. If the result of the error(s) yield a word that is not a valid nCm codeword, the receiver will detect that symbol as an *erasure*. In this case, the index of the symbol that contains the channel error is known to the receiver. 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 does not match the actual parity values received. In this case, the channel errors will constitute a symbol *error*. If this occurs, the index of the symbol that contains the channel error is not known and must be determined by the receiver. Recall that the symbols in the block code reference the enumerated subset number to which the actual nCm codeword belongs. In the event of channel error(s), the role of the block code is to determine from which subset the errored symbols originate. Once this information is found, the receiver knows the set of possible values for the symbol(s) in error. The receiver must then determine which of these candidate codewords have minimum distance to the received word. Assume a link consisting of nCm drivers, whose code set has been divided into  $q$  subgroups, each having  $c$  codewords 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- $q$  digits.  $n-k$  base- $q$  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 codewords that belong to the subset specified by the digit may be chosen. The sequence of choices made will encode the remaining  $b_{symbol}$  bits of the message.

We have identified three classes of block codes for use in our hierarchical ECC mechanism. Each class has unique properties which 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 a symbol error. It's most important advantage is that it places no restrictions on  $q$ , 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 Codes. These types of 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 which 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  $q+1$ . This restriction will place limits on the maximum code rate which may be achieved under certain conditions. The third class of block code is called *almost maximum distance separable* codes, or *AMDS* codes. 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  $n = q^2 - 1$ .

Code rate is the most important metric for evaluating the success of the codes developed in this work. 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:

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

In this study, we have chosen parameters for several different hierarchical ECC codes. In the first experiment, we have built several types of codes which have the ability to correct any single channel error in any single code symbol. In each case, the nCm code sets were partitioned into subsets with  $d=4$ , such that any one channel error 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. Table 3 shows these same results for situations where the code rate is near 50%. In a second experiment, we have built several types of codes which have the ability to correct multiple channel 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 each case, the nCm code sets were partitioned into subsets with various distances. In the codes that use 5c2 and 7c3 symbols, an MDS block code with two parity symbols is used until the point at which the number of symbols exceeded the limit for an MDS code. At this point, the code is switched to an equivalent AMDS code by replacing one data symbol with a parity symbol, for a total of three parity symbols. In any case, the distance of these codes is 3, while the subset distances are 4. Therefore, one channel error can be corrected in any two symbols. In the codes that use 6c3 and 8c4 symbols, a checksum code with distance=2 is used while the subsets have distances of 6 and 8, respectively. In the code that uses 6c3 symbols, one channel error can be corrected in any single symbol, or two channel errors may be corrected with  $\geq 50\%$  probability (the errors cannot yield a valid code word). The code that uses 8c3 symbols, one or three channel errors can be corrected in any single symbol and two channel errors may be corrected with 50% probability. Table 4 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 channel error within a single symbol. Selected results are given in Table 5, where the number of total channels is  $\leq 30$  are shown.

nCm	subsets	cw per subset	n	k	bits	channels	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 3:** Selected results for specified codes

nCm	subsets	cw per subset	n	k	bits	channels	code rate
5c2	3	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 4:** Selected results for specified codes

nCm	subsets	cw per subset	n	k	bits	channels	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 5:** Selected results for specified codes