This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Statistical Simulator for Block Coded Channels with Long Residual Interference

Natasa Blitvic Vladimir Stojanovic
Massachusetts Institute of Technology
Cambridge, Massachussetts, 02139
{blitvic, vlada}@mit.edu

Abstract— **In this paper, we present a simple statistical simulation technique for channels with long memory operating under coded data. The proposed technique employs a divide-and-conquer approach, both at the level of the channel response and the individual codeword. We introduce an efficient algorithm for parity tracking during the generation and combining of the voltage distributions computed in individual sub-problems. The resulting computational complexity increases linearly both with the channel and the codeword length, keeping the number of parity bits constant. The complexity increases exponentially with the number of parity bits in a codeword. Thus, the technique is of most use for high-rate codes. Finally, the technique is applied to examine the effect of coding on high-speed links which operate at BERs smaller than 1e-15 and where residual interference typically spans several hundreds of bits. This simulator enables systematic evaluation of power-performance trade-offs in introducing error correction/detection codes into high-speed link systems.**

I. Introduction

Exact performance analysis of a coded communication system over a channel with long memory typically requires a combinatorial search, and is therefore impractical, if not prohibitive, due to the size of the resulting state-space. Many communication systems have little need for exhaustive simulations of this type, as they employ sophisticated equalization and/or modulation techniques with channel shortening to limit the number of channel states presented to the encoder/decoder blocks. These powerful methods bring the unwanted channel memory to a negligible level so that it can be treated as mean-distortion and added to the noise term [1] in the Gaussian Signal-to-Interference-and-Noise (SINR) representation of the Bit-Error-Rate (BER). In some systems however, additional constraints on the system complexity (power, area) limit the extent to which the channel-memory can be controlled. This leaves a significant portion of unwanted channel memory as residual intersymbol-interference (ISI) and causes difficulty in accurate performance estimation, especially for systems operating with coded data.

Monte Carlo-type simulators have the disadvantage of requiring a large number of samples for accurate performance estimation at low error rates. Sample-size reduction methods, such as importance sampling, exist, but long channel memory reduces their effectiveness of due to the dimensionality effect [2]. For channels with long memory, analytically computing the probability distributions associated with received voltages is common practice [3]–[5] but all such simulators assume the transmitted bits to be independent. We present an analytical simulation method that accurately accounts for the effect of data correlation due to both channel coding and channel memory on the probability distribution of the received voltage. The system is assumed to operate under a systematic (n, k) binary linear block code, where n is the codeword length and k the corresponding number of information bits. Ordinarily, computing the probability distribution of the received voltage involves considering all the 2^k possible codewords, and is therefore impractical. To reduce the problem complexity, the method employs a divide and conquer approach, both at the level of the channel response and the individual codeword. For a given code, the complexity increases linearly with channel length. In contrast to the combinatorial approach, the complexity increases *linearly* with the number of information bits k. However, the complexity increases exponentially with the number of parity bits, $n - k$, so the technique is of most use for high-rate codes. The accuracy of the method is controllable, so, given an adequate channel model, the results are valid at arbitrarily low error rates. The resulting probability distributions can be integrated to yield bit cross-over probabilities. On channels that can be well approximated as binary symmetric, these cross-over probabilities can be further used to estimate error rates after decoding.

Although the resulting simulation tool can be applied to a variety of systems, the development of a simulator for block-coded channels with long memory is motivated by recent research in high-speed links. These high-speed chip-to-chip interconnects can be found in large numbers in processor-memory interfaces, as well as server/router backplanes and crossbar switches. They present a challenging, bandlimited communication environment, especially due to long residual inter-symbol interference caused by signal reflections off impedance discontinuities. Due to severe power, area and complexity constraints, only limited equalization and modulation are used, leaving the residual channel memory typically longer than a hundred symbols. This residual ISI limits the achievable link data rates to an order of magnitude below their projected capacity [6].

It is currently an open question how channel coding affects the power/performance tradeoff in a high-speed link. Its promise was first demonstrated in [7] where a shortened Hamming code yielded an improvement of 7 orders of magnitude. However, at the time of this writing, no systematic study has yet been documented. The lack of general results is principally due to the difficulty of accurately simulating the system's behavior under different codes at typical high-speed link BERs, on the order of 10^{-15}. The simulation technique proposed in this paper extends the scope of statistical link simulations to channels operating with coded data.

In the following sections we first formulate the system model and discuss the previous work. We then describe the proposed simulation method. Finally, we illustrate the significance of the method by analyzing the effect of some codes on a high-speed link channel.

II. System Model

The simplified model of a communication system assumed throughout this document is shown in Fig. 1. Encoder/decoder blocks operate under a linear block code. The system employs PAM2 modulation, where, for the convenience of representing binary addition over {0,1} as multiplication over {-1,1}, a logical '1' maps to a negative voltage. The transmitter and receiver may contain equalizers, in which case the channel response is due to the residual inter-symbol interference (ISI). In high-speed links, the two main mechanisms that account for the most significant portion of the residual ISI are dispersion and reflection. In addition, residual interference may include co-channel interference, caused, for instance, by electromagnetic coupling (crosstalk) [8], [9]. As accounting for co-channel interference involves the same set of mathematical tools as accounting for the ISI, the remainder of the paper first develops the tools for computing the effects of the ISI and then generalizes the results to other sources of interference.

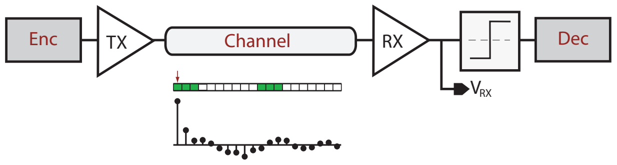


Fig. 1. Simplified Model of a Communication System

The quantities of interest are the received voltage at the input to the decision circuit, V_{RX}. The received voltage bears a signal component, V_S, which includes the ISI, and the noise component V_N, assumed to be additive and independent of the data. As $V_{RX} = V_S + V_N$ and the signal and noise quantities are statistically independent, the probability distribution of V_{RX} is given as:

$$\begin{aligned} f_{V_{RX}}(v) &= \int_{\Psi} f_{V_{RX}|V_S}(v|\psi) \times f_{V_S}(\psi)\, d\psi \\ &= \int_{\Psi} f_{V_N}(v-\psi) \times f_{V_S}(\psi)\, d\psi \end{aligned} \tag{1}$$

where Ψ is the sample space associated with V_S. The subsequent development concerns a noiseless system, therefore focusing on V_S. Adequate noise models whose effects can be quantified in either closed-form or numerical form already exist for a wide range of systems and can therefore be included as shown above. Timing and voltage noise models for high-speed links are developed in [6].

The distribution of V_S is computed from the channel's pulse response, sampled evenly at symbol intervals. The length of the pulse response typically spans multiple bits, as illustrated in Fig. 2. Assuming the pulse response is L symbols long, each of the possible L-symbol input patterns maps to a specific voltage at the output of the channel. The resulting voltage is simply given as a weighted sum of the transmitted symbols, X_1 to X_L, and channel coefficients, c_1 to c_L:

$$V_S = \sum_{i=1}^{L} X_i c_i \tag{2}$$

Since, for a given channel response, V_S is inherently discrete, its distribution is entirely determined by a probability mass function (PMF), or, alternatively, a cumulative mass function (CMF).

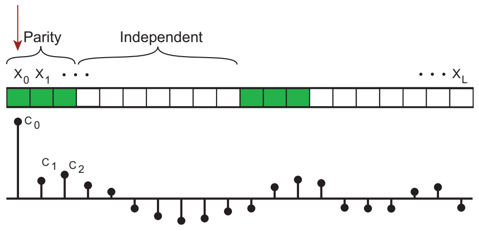


Fig. 2. Sample channel with long memory – The channel response spans not only multiple bits, but also multiple codewords. The shading indicates the location of parity bits. The arrow indicates the location of the most recently transmitted bit, chosen here to be the last parity bit. The choice is arbitrary, as any bit location can be selected by shifting and padding the pulse response. For the ease of representation, the channel response is assumed to be causal. However, all the techniques presented in this paper also apply to systems with pre-cursor ISI.

III. Previous Work

For systems operating at low error rates, Monte Carlo simulation is impractical due to the large sample size requirements. Estimating the performance of such systems relies principally on analytical computation of accurate probability distributions. Given that the residual ISI can still be an important noise mechanism, especially at low BER, it is not surprising that several analytical tools for estimating the system's performance address this issue. In [3] and [4], to calculate f_{V_S}, the authors consider individual distributions associated with each bit in the channel. More precisely, the contribution of a transmitted symbol X_i to the overall received voltage is $-c_i$ when $X_i = -1$ and c_i when $X_i = 1$. Assuming the transmitted bits are independent, the distribution of the overall received voltage becomes the convolution of the individual densities. Similarly, [5] operates on the same principle, except that the individual probability distributions are computed for 4 symbols at a time.

Expressing the probability density of the total received voltage as the convolution of probability densities due to each individual bit in the channel holds only for independent bits. The above approaches are therefore not valid for coded data.

Furthermore, it is not clear whether and where a quantization step should take place. To decrease the memory and runtime requirements, it is beneficial to quantize the resulting voltages to some adequate precision. However, quantizing voltages too early in the process results in an unnecessary loss of accuracy. This issue is further discussed in the next section.

IV. Accounting For Data Correlation Due To Channel Coding

Channel codes introduce controlled redundancy in order to improve the reliability of the transmission. In a $(n,\, k)$ linear block code, each n-bit codeword is composed of k information bits and the corresponding $m = n - k$ parity bits. The parity bits can be expressed as linear combinations of information bits over $\mathbb{F}_2$. Since the k information bits entirely determine the values of the corresponding m parity bits, the stream of bits transmitted through a communication channel can no longer be considered independent.

This section presents a technique for accounting for data correlation due to a systematic binary linear block code. We first consider a basic case where all the possible bit patterns are computed combinatorially prior to being translated into a probability mass function. For simplicity, it is assumed that the codeword length equals the length of the channel response. We then present a method of reducing the combinatorial complexity of the problem. Finally, we generalize this method to the case where the channel length spans multiple codewords and includes additional sources of interference.

A. Basic Case

A straightforward solution to calculating voltage distributions for coded data consists in computing the received voltage due to k independent and m parity bits combinatorially. Each of the 2^k possible patterns formed by the information bits results in a specific state of the parity. Translating the resulting 2^k n-bit patterns into voltages and noting that the patterns are equally likely yields the probability distribution of the received voltage due to one codeword.

The size of the resulting probability mass function can be reduced by quantizing the resulting voltages to some precision Δ and adequately grouping the corresponding probabilities. This mechanism is illustrated in Fig. 3 for a simple (3,2) single-parity-check code with $\Delta = 0.1$.

B. Reducing the Combinatorial Complexity

While the above method is adequate for dealing with codewords of relatively short to moderate lengths, it quickly becomes burdensome as the number of independent bits in a codeword increases. For instance, following the above method, computing the probabilities associated with a codeword of $k = 100$ information bits and any number m of dependent bits over a realistically long channel[1] requires considering

[1]The results of Section V were obtained on a backplane channel of comparable length

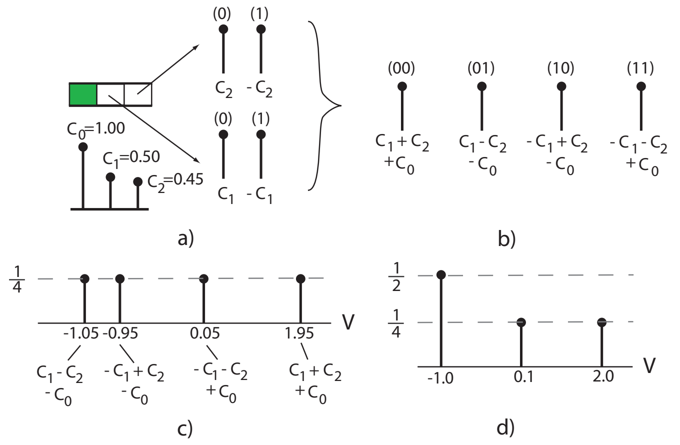


Fig. 3. Example: Distribution of voltage over one codeword – a) The corresponding portion of the channel response. b) The combinatorial expansion. c) The resulting PMF. d) The quantized PMF.

$2^{100} \approx 10^{30}$ distinct possibilities. In order to handle arbitrarily long codewords, it is therefore necessary to reduce the combinatorial complexity of the method.

An intuitive approach consists of subdividing the codeword's information bits into non-overlapping sets. The set of possible bit patterns associated with each resulting subcodeword is mapped to a set of possible voltages, thus forming a partial PMF. Each partial PMF also keeps track of its contribution to the overall state of the parity. To effectively reduce the storage/processing requirements, it is convenient to quantize the voltages to some adequate precision and group the identical elements. The combination of partial PMFs leads to the complete PMF for one full codeword. However, for the result to be equivalent to the PMF obtained by full combinatorial expansion outlined previously, it becomes necessary to develop methods for:

1) *Computing* the partial PMFs efficiently, while keeping track of the parity information for each possible voltage.
2) *Grouping* elements of a partial PMF that correspond to the same point on the voltage axis while preserving the parity information.
3) *Combining* the partial PMFs from each subcodeword while updating the parity information.

We address the issue of *computing* partial PMFs in the following manner. Subdivide the k information bits of a codeword into non-overlapping blocks of d information bits. Each block of d information bits can take on any of the 2^d possible patterns. For each pattern, compute two quantities:

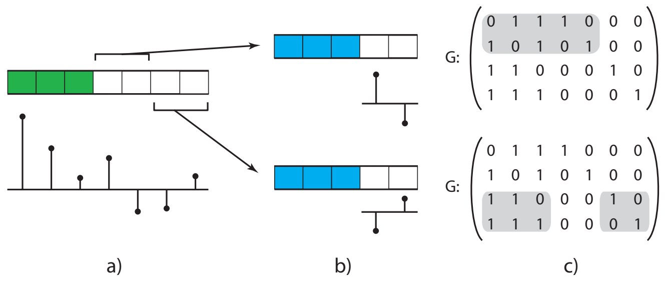
- the contribution to the total[2] received voltage;
- the contribution to the overall state of the parity.

The result is a *partial PMF* composed of 2^d $(m + 1)$-dimensional *coordinates*, where each coordinate corresponds to one possible d-bit pattern. A coordinate is of the form

$$(b_1, b_2, ..., b_m)^T @ v$$

[2]Unless otherwise stated, qualifiers such as *total* and *overall* refer to one codeword. For instance, the *total* received voltage is the voltage due to the portion of the channel response spanning the present codeword.

where b_i is the pattern's contribution to the i^{th} parity bit and v its contribution to the received voltage. Since each coordinate corresponds to a different pattern of information bits, and all such patterns are equally likely, it unnecessary to record any probabilities. The following is a useful alternative interpretation of the outlined process. Computing the voltage and parity contributions of different sections of one codeword is analogous to computing voltage PMFs (while separately storing the parity states) of different *subcodewords*. A subcodeword is obtained by appending m parity bits to each block of d information bits. In this manner, computing the contribution to the overall state of the parity due to d bits becomes equivalent to computing individual parities for each subcodeword. The code associated with a given subcodeword is the original code shortened to include only the d bits present. This is illustrated in Figure 4 for a (7,4) Hamming code.



$$G: \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 4. Example: Partitioning a (7,4) Hamming codeword for d = 2 – a) The original codeword. b) Two subcodewords. c) The portions of the generator matrix associated with each subcodeword. We assume, without loss of generality, the partitioning of the independent bits to be consecutive.

Grouping the coordinates associated with the same voltage reduces the size of the partial PMF. In order to further reduce the memory and processing requirements, all voltage contributions can also be quantized to a lesser accuracy and regrouped to a greater degree. Fig. 5 depicts the partial PMF computed for for $d = 2$ and quantized to $\Delta = 0.1$.

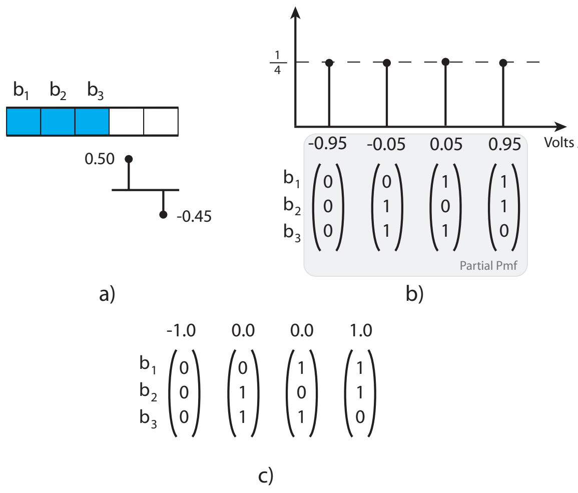


Fig. 5. Example: Partial PMF for one subcodeword – a) The corresponding portion of the channel response. b) The combinatorial expansion and the partial PMF. c) The quantized partial PMF.

Grouping several coordinates associated with the same voltage requires keeping track of the parity patterns contained in each coordinate. Also, it becomes necessary to record the probability counts, that is, the number of information patterns that produced a given voltage/parity combination. To preserve the exact states of parity bits and their probability counts while grouping two voltages, we assign a *type* to every possible parity pattern. The type of a specific pattern of parity bits in one coordinate is simply defined as the decimal equivalent of the corresponding binary string. Consequently, there are 2^m possible parity pattern types, ranging from 0 to $2^m - 1$. It then becomes sufficient to record the number of different parity types at each distinct voltage present in the partial PMF to track accurately both the probability count and the state of the parity bits. This process is illustrated in Fig. 6. For an alternative view of the method, we draw on some computer science terminology. A coordinate associated with a given voltage v contains a *record* of all the parity patterns (types) associated with v and present in the partial PMF and their corresponding probability counts. A record is composed of 2^m fields, each corresponding to a different parity pattern. The value at each field is the number of the information patterns (probability counts) that produced v and that specific parity pattern. To distinguish the more primitive partial PMF representation of Fig. 6 a) from that of Fig. 6 c), we refer to the latter as the *grouped coordinate representation*.

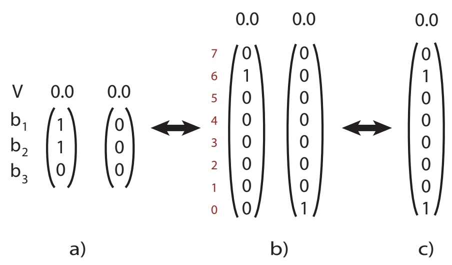


Fig. 6. Example: Grouping several coordinates – a) Possible parity types for k = 2. b) Transforming two voltage/parity coordinates to reflect their parity patterns. c) Grouping two voltage/parity coordinates. Note that the grouping is reversible and therefore causes no loss of information.

The issue of *combining* the partial PMFs associated with distinct codewords remains to be addressed. Assume that two non-overlapping sets of d information bits produce M_1 and M_2 partial PMF coordinates, respectively, where each coordinate follows the "voltage + record" format defined previously. Since no information bits are shared between subcodewords, the subcodewords are statistically independent. Thus, the equivalent contribution of $2d$ information bits results in $M_1 \times M_2$ coordinates[3]. To illustrate the basic combining principle, we revert briefly to the original coordinate representation. A given pattern of d independent bits from subcodeword l produces a voltage contribution v and a parity state $(b_{l1}, b_{l2}, ..., b_{lm})^T$, where b_{li} is once again the impact on the i^{th} parity bit in the complete codeword due to subcodeword l. The task at hand is

[3]These need not be distinct, so the combining step is again followed by grouping coordinates with identical voltages

to combine two coordinates, namely $(b_{r1}, b_{r2}, \ldots, b_{rm})^T @v$ and $(b_{s1}, b_{s2}, \ldots, b_{sm})^T @w$, resulting from subcodewords r and s. From the manner in which we define voltage and parity contributions, it follows that the resulting combined codeword is of the form

$$(b_{r1} \oplus b_{s1}, b_{r2} \oplus b_{s2}, \ldots, b_{rm} \oplus b_{sm})^T @v + w \tag{3}$$

where $\oplus$ denotes a *bit-wise xor* operation. Extending the above result to the grouped coordinate representation described previously requires introducing some additional notation. Denote by $\mathbf{p}$ and $\mathbf{q}$ two partial PMF coordinates associated with two distinct subcodewords, r and s. As described earlier, each coordinate is associated with a certain voltage and contains a record of parity patterns and probability counts. More precisely,

$$\mathbf{p} = \begin{bmatrix} v \\ \mathbf{x} \end{bmatrix} \qquad \mathbf{q} = \begin{bmatrix} w \\ \mathbf{y} \end{bmatrix}$$

where v and w are the voltages and $\mathbf{x}$ and $\mathbf{y}$ the corresponding records. From (3), the combination of $\mathbf{p}$ and $\mathbf{q}$ yields a coordinate at $v + w$. The updated record is computed one parity type at a time. From (3), a pattern of *type i* combined with a pattern of *type j* yields a pattern of type type $i \oplus j$, where the $\oplus$ operator is again bit-wise. Adding the fact that the subcodewords are independent, it follows that x_i patterns of *type i* combined with y_j patterns of *type j* produce $x_i \times y_j$ patterns of type $i \oplus j$, where x_i and y_i are the i^{th} fields of records $\mathbf{x}$ and $\mathbf{y}$, respectively. Summing over all the combinations of i and j that produce a coordinate of *type k*, we obtain that

$$z_k = \mathbf{x}^\mathbf{T} \mathbf{y}(\mathbf{I_k}) \tag{4}$$

where $\mathbf{z}$ is the record of the combined coordinate whose field z_k denotes the probability count associated with $v + w$ and a parity pattern of *type k*. Also in the above equation, $\mathbf{I_k}$ represents the permutation vector specifying the order in which the elements of one of the two coordinates need to be appear to yield the correct number of elements of *type k*. From (3), it directly follows that:

$$\mathbf{I_k} = \begin{pmatrix} k \oplus 2^m - 1 \\ \vdots \\ k \oplus 1 \\ k \oplus 0 \end{pmatrix} \tag{5}$$

where the indexing starts at 0. It follows that the combining operation can be efficiently implemented as a series of 2^m permutations and dot products. Table I shows all the permutation vectors for $m = 3$, while Fig. 7 illustrates the result of the combining process for two coordinates. Finally, all the resulting coordinates associated with the same voltage are regrouped as outlined previously. Once all the subcodewords have been combined, the total probability count and the effects of parity on the total codeword voltage are computed as described in the previous subsection. This is further illustrated in Fig. 8.

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
7	6	5	4	3	2	1	0
6	7	4	5	2	3	0	1
5	4	7	6	1	0	3	2
4	5	6	7	0	1	2	3
3	2	1	0	7	6	5	4
2	3	0	1	6	7	4	5
1	0	3	2	5	4	7	6
0	1	2	3	4	5	6	7

TABLE I
PERMUTATION VECTORS FOR m = 3

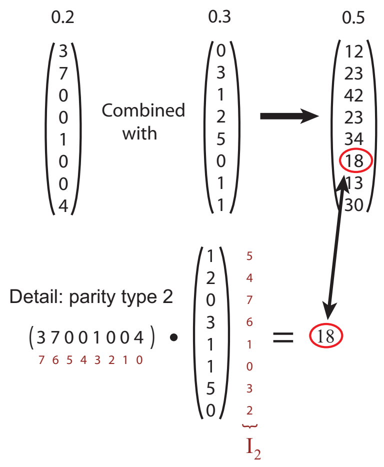


Fig. 7. Example: Combining two coordinates from two different subcodewords with m = 3. The principle of the record permutation is shown in more detail for the parity pattern of type 2 where $\mathbf{I_2}$ is used to compute the resulting probability counts.

C. Extension To Multiple Codewords And Additional Interference

The following simple generalization allows computing the distribution of the received voltage due to the full channel response for codewords whose length is less than the length of the channel. Assume without loss of generality[4] that the channel length L can be subdivided into an integer number of codewords K. Then the received voltage due to the full channel response, V_{RX}, can be written as a sum of the responses to individual codewords, V_i. More precisely, $V_{RX} = V_1 + \ldots + V_K$, as illustrated in the following figure.

Owing to the statistical independence between codewords, the distribution of the total received voltage, $f_{V_{RX}}$ is simply the convolution of the probability mass functions associated with each codeword. Namely, $f_{V_{RX}} = f_{V_1} * f_{V_2} * \ldots * f_{V_K}$ where each of the individual responses is computed as outlined previously.

Based on this principle, Fig. 9 also illustrates the method of accounting for co-channel interference. As the interfering symbols are independent from the transmitted data, their effect can be treated in the same manner as the effect of previously transmitted codewords. Note that the proposed simulation technique is general enough to allow the code to

[4]The channels response can be zero-padded to provide for an integer K.

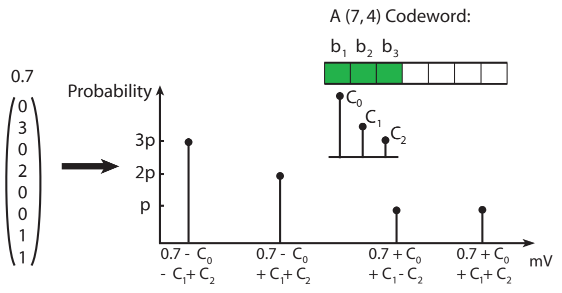


Fig. 8. Example: Obtaining the full probability distribution from the combination of all partial PMFs. After all the partial PMFs have been combined, the result is converted into a standard probability distribution. This conversion is illustrated for one coordinate. The parity patterns present are converted to PAM2 (here, '$0'$ $\rightarrow +1$) and mapped to the corresponding channel coefficients as given in (2). The probability counts are extracted from the field associated with each type and normalized by $p^{-1} = 2^k$, corresponding to the the total number of possible bit patterns associated with one codeword.

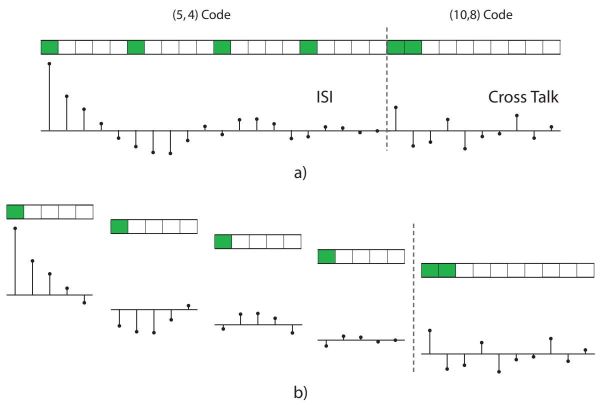


Fig. 9. Contributions of individual codewords to the total received voltage – a) Full channel response. b) Individual codeword responses.

vary between different codewords. Therefore, the interfering channel is allowed to operate under a different code.

D. Complexity Issues

Finally, it is useful to consider the degree of complexity reduction achieved through the proposed method. For a given codeword length, an increase in the length of the channel translates into an increase in the number of convolutions of codeword PMFs. The complexity increase is therefore linear. Similarly, for a codeword with a given number of parity bits m, increasing the number of independent bits k translates into an increase in the number of combinations of partial PMFs. This causes the complexity to increase linearly as well. However, due to the parity tracking mechanism, or, more particularly, the mapping of parity patterns to 2^m parity pattern types, the complexity increase resulting from increasing the number of parity bits is exponential. Nevertheless, for high-rate codes, the number of parity bits is typically much less than the number of independent bits and the overall reduction in complexity is still significant. Section V illustrates the runtime performances for different codes based on a MATLAB implementation of the technique.

E. Quantization Error

The quantization error depends on the quantizing Δ, the ratio of the channel length L to the partitioning length (size of the subcodeword excluding the parity) d, and on where the quantizing step takes place. Pre-quantizing each coefficient yields a maximum absolute quantization error of $L\Delta$, while quantizing only at the level of each partial PMF yields a reduced error of at most $\lceil L/d \rceil \Delta$. It is therefore advisable to assign a large, but practical, value to d. As an illustration, consider a system with $L = 200$ and choose $\Delta = 10^{-4}$ (0.1 mV) and $d = 10$. Then, for some (100,90) random code, there are two codewords to consider, each subdivided into 9 subcodewords. The resulting maximum quantization error becomes 1.8×10^{-3}. Assuming the distribution of the quantization error to be symmetric around 0 and individual quantizing errors for each voltage to be independent, the maximum quantization error occurs with probability 2^{-17}. More realistic bounds on the quantization error can be obtained by using any of the well-known bounds for sums of independent identically distributed random variables with finite variance.

V. Performance Evaluation And Simulation Results

In this section, we first quantify the runtime performance of the proposed algorithm when implemented in MATLAB. We then illustrate the effect of accounting for data correlation due to a linear block code on the distribution of the received voltage. Finally, we show how the accurately computed cross-over probabilities can be used to estimate the error probabilities *after decoding*. All the results are obtained with the equalized channel responses shown in Fig. 10.

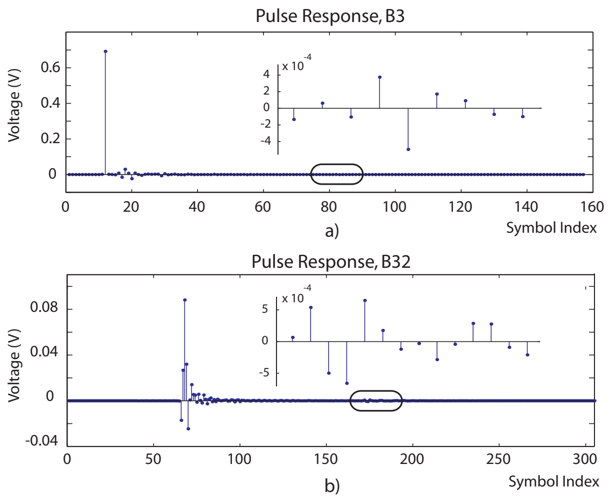


Fig. 10. Two High-speed Link Channels – The equalized pulse responses shown are based on the Peters ATCA measurement data for a 3" [top] and 32 " [bottom] backplane with bottom-layer routing [10]. The equalizer has 5 taps. The links operate at 5 Gb/s and 10 Gb/s respectively.

A. Runtime Statistics

All the distributions computed in this section are obtained with $d = 10$ bits and $\Delta = 10^{-4}$ Volts. The runtime statistics were obtained on a Pentium 1.60 GHz processor with 504 MB of RAM. Table II a) presents the runtimes for Hamming codes of different sizes. For codewords whose length exceeds the length of the channel, the channel response is zero-padded. As this increases the effective channel length the runtime results present a pessimistic view of the efficiency of the simulator for particularly long codewords. To illustrate the increase in the runtime incurred by an increase in the number of parity bits alone, Table IIb) provides the runtime statistics for a random $(290 + m, 290)$ code.

(a)

Code (n,k)	Runtime (s)
(31,26)	5.15
(63,57)	9.86
(127,120)	15.5
(255,247)	38.0
(511,502)	178
(1023,1013)	1.04×10^3

(b)

Code (n,k)	Runtime (s)
(295,290)	11.0
(296,290)	12.8
(297,290)	18.4
(298,290)	36.0
(299,290)	104
(300,290)	453

TABLE II
RUNTIME STATISTICS – A) HAMMING CODES. B) INSTANCES OF RANDOM CODES OF FIXED CODEWORD LENGTH.

B. Link Performance

To illustrate the extent to which the independent data assumption can affect the accuracy of the system's performance picture, we consider a hypothetical set of codes implemented in a high-speed link. This set is chosen as the set of all possible (10,8) systematic linear block codes and is motivated by the runtime-limiting/DC-balancing 8b/10b codes [11] traditionally implemented in some high-speed links. While the 8b/10b codes are not systematic and therefore cannot be simulated with the proposed tool, it is interesting to consider possible effects of systematic linear block codes with the same overhead. The resulting probability distributions are shown in Fig. 11. The effects of adding a realistic amount of system noise to these distributions are shown in Fig. 12.

The above figures show that the probability distributions calculated for block-coded inputs can largely deviate from the corresponding uncoded distributions. Furthermore, some codes can significantly reduce the effect of the residual ISI on the cross-over probabilities for a given bit location. The difference was on the order of 10^3 for the B3 channel operating at an uncoded BER of 10^{-10} and $>> 10^{21}$ for the B32 channel under the same conditions. This is achieved by eliminating certain worst-case patterns which cause the largest ISI from the set of all possible bit patterns. A simulation technique able to accurately quantify this difference thus becomes a useful tool.

Finally, the accurately computed crossover probabilities can be used to determine the decoded error rates on channels that can be well approximated as binary symmetric. When a bit is in error independently from any other bit, the decoded error probabilities are computed simply from a binomial distribution[5]. On channels where the dispersion is well-compensated, it is plausible to expect the error dependencies to be weak and this approximation to hold. An example of such a channel is the B3 channel shown in Fig. 10 a). With the dispersion taps removed, the power in the residual interference is typically sufficiently low for the measured received voltages at two time instants to be considered uncorrelated. Furthermore, with the absence of strong dispersion taps, no channel taps are

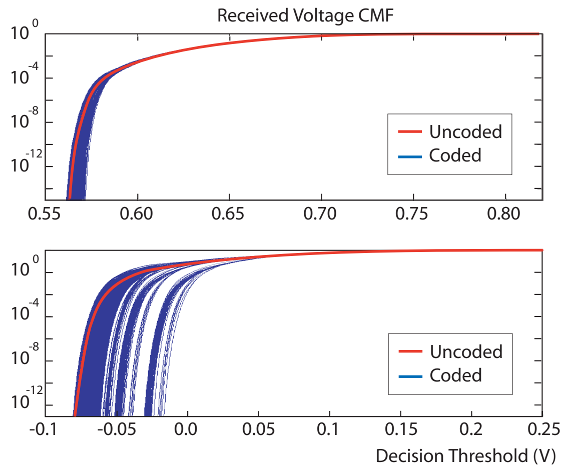


Fig. 11. Voltage distributions for the set of all (10,8) linear block codes. Shown are the cumulative mass functions (CMF) for two different channels: a) B3 b) B32. The plots also show the voltage CMF computed under the assumption that the data is uncoded. For consistency, both the coded and the uncoded CMF are quantized $d = 4$ independent bits at a time. The quantization step is 0.5mV. The probability distributions shown correspond to possible voltages associated with the first information bit in a codeword, bit X_n.

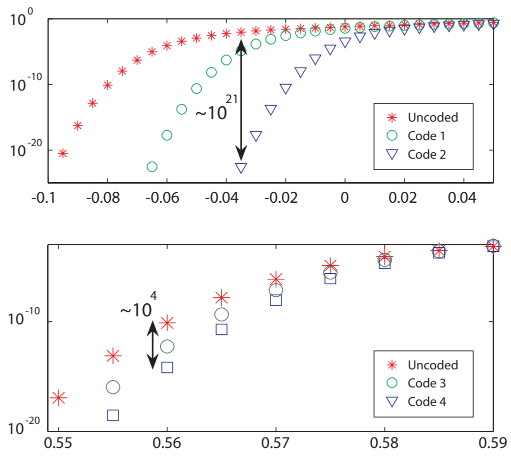


Fig. 12. Bit crossover probabilities for the uncoded case and two different (10,8) linear block codes on a B3 channel [top] and B32 channel [bottom]. Code 2 was chosen to yield the maximum deviation from the uncoded CMF, while Code 1 was chosen roughly in between the two extremes. The results are computed under the common assumption that the link noise is additive, white and Gaussian with $\sigma \approx$ 3mV,

[5]If the individual bit errors are independent but not identically distributed, all error patterns are no longer equally likely. The answer requires more computation – computing the probability of every pattern – but requires the same type of information: marginal probability densities.

dominant. Thus, for long channels and uncoded data, the sum of independent, identically distributed random variables given in (2) is approximately Gaussian. The weak correlation therefore translates into a weak dependency.

Data dependencies due to channel coding somewhat weaken this argument. Besides being tied to channel characteristics, the accuracy of the BSC approximation therefore also depends on the code itself and also potentially the target BER. It is beyond the scope of this document to make precise the conditions where the approximation holds with certain accuracy. Instead, we conclude by presenting an example of a realistic dispersion-compensated channel where the approximation holds sufficiently well for most practical purposes. The comparison between measured and predicted results for the B3 channel described previously is shown in Tables II and III, for a Hamming (31,26) and an extended Golay code respectively. The crossover probabilities are obtained from voltage distributions computed analytically by setting $\Delta = 10^{-4}$V and $d = 10$ and averaged over all the bit locations. The predicted word error rates (WER) for the Hamming code are calculated according to the BSC assumption, namely $WER = 1 - (1-p)^31 - 31p(1-p)^{30}$. This is a standard expression corresponding to the binomial probability that a codeword of length $n = 31$ contains more than a single bit error. Similarly, the WER for the Golay code is given by

$$1-(1-p)^24-24p(1-p)^{23}-276p^2(1-p)^{22}-2024p^3(1-p)^{21}$$

where the decoding error occurs if more than 3 bit errors are present in the codeword. The Monte Carlo simulator operates with 10^9 codewords. At word error rates that can be accurately captured by Monte Carlo simulation, the approximation holds very well for the Hamming code and holds relatively well even for the extended Golay code, despite the fact that one half of each 24-bit codeword corresponds to parity. The discrepancies between the predicted and the observed WER values are attributable to both the approximation error and the variance of the Monte Carlo estimate, where the latter is particularly pronounced at low error rates. For the cross-over probabilities, such discrepancies can result from the estimator variance and the quantization Δ.

Decision	Predicted		Measured	
Threshold (V)	p_{xover}	WER	p_{xover}	WER
0.56	1.45e-8	9.70e-14	1.32e-8	0
0.57	1.81e-5	1.53e-7	1.80e-5	1.30e-7
0.58	5.10e-4	1.20e-4	5.09e-4	2.68e-4
0.59	3.16e-3	4.38e-3	3.16e-3	7.56e-3
0.60	1.12e-2	4.67e-2	1.11e-2	5.36e-2

TABLE III
PREDICTED AND MEASURED CROSS-OVER PROBABILITIES (p_{xover}) AND WORD ERROR RATES (WER) FOR A (31,26) HAMMING CODE

Decision	Predicted		Measured	
Threshold (V)	p_{xover}	WER	p_{xover}	WER
0.56	1.02e-8	1.20e-15	4.17e-8	0
0.57	2.33e-5	2.65e-15	2.21e-5	0
0.58	4.77e-4	5.47-10	5.20e-4	0
0.59	3.22e-3	1.08-6	3.17e-3	4.00e-6
0.60	1.12e-2	1.40e-4	1.12e-2	6.31e-4
0.61	2.79e-2	4.11e-3	2.76e-2	1.00e-2

TABLE IV
PREDICTED AND MEASURED CROSS-OVER PROBABILITIES (p_{xover}) AND WORD ERROR RATES (WER) FOR AN EXTENDED GOLAY CODE, (24,12)

VI. CONCLUSION

We have developed a general statistical technique for evaluating the effect of block channel coding on the distribution of the received voltage over channels with long residual interference. The proposed technique uses a divide-an-conquer approach to computing the voltage distribution functions and efficient parity tracking algorithm. Coupled with a binary symmetric channel model, this technique forms the basis for coding gain evaluation over such channels. This framework for the first time enables a systematic study of power-performance trade-offs associated with introduction of block codes to high-speed links. We illustrate this on a simple example of a random (10,8) code showing that received voltage distributions of coded data can be significantly different from uncoded data, potentially leading to large differences in computed cross-over probabilities.

ACKNOWLEDGMENTS

This research is supported by the FCRP Interconnect Focus Centre.

The authors would also like to thank Professor Lizhong Zheng of the Massachusetts Institute of Technology for valuable technical discussions.

REFERENCES

[1] J. M. Cioffi, *EE379A course reader*, Stanford University (available at `http://www.stanford.edu/class/ee379a/`)
[2] P.J. Smith, M. Shafi, H. Gao "Quick simulation: a review of importance sampling techniques in communications systems," *IEEE Journal on Selected Areas in Communications,*vol. 15, no. 4, pp. 597-613, 1997.
[3] B.K. Casper, M. Haycock, R. Mooney "An accurate and efficient analysis method for multi-Gb/s chip-to-chip signaling schemes," *IEEE Symposium on VLSI Circuits*, pp. 54-57, June 2002.
[4] B. Ahmad, "Performance Specification of Interconnects," *DesignCon 2003*.
[5] A. Sanders, M. Resso, J. D'Ambrosia "Channel Compliance Testing Utilizing Novel Statistical Eye Methodology," *DesignCon 2004*.
[6] V. Stojanovic, A. Amirkhany and M.A. Horowitz, "Optimal linear precoding with theoretical and practical data rates in high-speed serial-Link backplane communication," *IEEE International Conference on Communications,*June 2004.
[7] L. E. Thon, H-J. Liaw, "Error-Correction Coding for 10Gb/s Backplane Transmission," *DesignCon* 2004.
[8] V. Stojanovic, M. Horowitz "Modeling and analysis of high-speed links," *IEEE* *Custom Integrated Circuits Conference*, pp. 589-594, Sept. 2003.
[9] J. Zerbe, C. Werner, V. Stojanovic, F. Chen, J. Wei, G. Tsang, D. Kim, W. Stonecypher, A. Ho, T. Thrush, R. Kollipara , M. Horowitz, K. Donnelly, "Equalization and Clock Recovery for a 2.5 - 10Gb/s 2-PAM/4-PAM Backplane Transceiver Cell," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 12, pp. 2121-2130, Dec. 2003.
[10] IEEE P802.3ap Task Force Channel Model Material (available at `www.ieee802.org/3/ap/public/channel_model`)
[11] A.X. Widmer, P.A. Franaszek "A DC-Balanced, Partitioned-Block, 8b/1Ob Transmission Code," IBM Journal of Research and Development, vol. 27, no. 5, pp. 440-451, Sept. 1983.