Tensor-Product Parity Codes: Combination With Constrained Codes and Application to Perpendicular Recording

Panu Chaichanavong[1] and Paul H. Siegel[2], *Fellow, IEEE*

[1]Marvell Semiconductor, Santa Clara, CA 95054 USA
[2]Center for Magnetic Recording Research, University of California at San Diego, La Jolla, CA 92093 USA

A parity code and a distance enhancing constrained code are often concatenated with a Reed–Solomon code to form a coding system for magnetic recording. The tensor-product parity coding scheme helps to improve efficiency of the parity code while retaining the same level of performance. In this paper, we present two methods for combining a tensor-product parity code with a distance-enhancing constrained code. The first method incorporates a constrained code with unconstrained positions. The second method uses a new technique, which we call word-set partitioning, to achieve a higher code rate relative to the first method. We simulate the performance of several coding systems based upon the two combination methods on a perpendicular recording channel, and we compare their symbol error rates and sector error rates with those of a system that uses only a Reed–Solomon code.

***Index Terms*—Constrained coding, parity code, perpendicular magnetic recording, tensor-product code.**

I. Introduction

IN DIGITAL recording systems, it is essential that the data are stored and retrieved reliably. This is achieved by an error correction coding system, which often includes a Reed–Solomon (RS) code. RS codes have excellent burst error correction capabilities, which make them particularly well suited to recording applications. However, magnetic recording channels also induce short, randomly occurring errors, against which some other classes of error correcting codes are known to perform better than RS codes. Moreover, RS decoders typically expect hard decisions at their input, and cannot take advantage of the soft information from the channel. For these reasons, several coding techniques have been devised to improve the overall system performance beyond that provided by an RS code alone.

One approach is to use graph-based codes and iterative decoding methods, such as low-density parity-check codes with message-passing decoding. Empirically, these codes have been found to offer better perfomance than an RS code, but they have not seen wide use in practice. This is due, in part, to the relatively high complexity and large memory requirements of the iterative decoders. Moreover, the decoder behavior is not fully understood, making it difficult to estimate the system performance at high signal-to-noise ratio (SNR).

A more practical approach is to concatenate the RS code with an inner code that reduces the error rate at the RS decoder input. Several inner codes have been proposed including parity codes [4] and distance enhancing codes [12]. A parity code can detect and sometimes correct short error events, while a distance enhancing code can eliminate some long error events. In this paper, we will consider both types of inner codes.

Choosing the block size of a parity code involves a tradeoff between code rate and performance. A short parity code performs better, as multiple error events within one block and miscorrections are less likely. However, the rate penalty of such a code is often too severe to be practical. To overcome the rate penalty, we propose the use of a "tensor-product parity code" whose parity-check matrix is the tensor-product of the parity-check matrices of a short parity code and a BCH code [3]. The motivation for the tensor-product parity coding scheme is the observation that usually only a few parity code blocks within a sector contain errors. So, instead of recording the parity bits of all blocks, we protect them with the BCH code; the redundancy introduced is then equal to that of the BCH code alone. If the BCH code is strong enough, the tensor-product parity code will provide the same performance as the short parity code, but with far less overhead.

A well-known class of distance enhancing codes is maximum transition run (MTR) codes [12]. These widely used codes, discussed in more detail in Section III, improve performance by limiting the runlength of consecutive transitions in the recorded sequence. However, combining a tensor-product parity code with an MTR code and an RS code is not a trivial task, for several reasons.

First, to take full advantage of the MTR code, we use a modified trellis that reflects the runlength constraints of the code as well as the partial-response channel constraints. With this trellis, the error rate after the Viterbi detector is greatly improved, but the recorded sequence must never violate the MTR constraint. Next, the recorded sequence must also satisfy the parity-check condition of the tensor-product parity code, since that code is decoded first. Finally, the decoding steps prior to the RS decoder must not propagate errors.

In this paper, we explore two methods for combining tensor-product codes and MTR constrained codes that satisfy these requirements. The first method is based on constrained systems with unconstrained positions [2]. This method is simple, but it

Digital Object Identifier 10.1109/TMAG.2005.861747

may suffer from an undesirable rate penalty. To reduce the rate penalty, we propose a second method, which we call "word-set partitioning." Both methods are described in detail in Section IV.

In Section V, we present results of error rate simulations on a perpendicular recording channel for coding systems based upon these two methods. As a benchmark for comparison, we also give results for a system incorporating an RS code without an inner code. Finally, in Section VI, we offer some concluding remarks.

II. Tensor-Product Parity Codes

The following description of the tensor-product parity code is taken from [3].

A. Code Construction

Let C_1 be an (n_1, k_1) parity code and let C_2 be an (n_2, k_2) binary linear code such that $n_1 - k_1$ divides n_2. We consider the code C with the following properties.

- C is an (n, k) binary linear code such that $n = k_1 n_2/(n_1 - k_1)$ and $k = n - n_2 + k_2$.
- Suppose we divide a codeword of C into disjoint blocks of length k_1, and compute the parity bits for each block using C_1. The parity bits of all blocks must form a valid codeword of C_2.

We construct a parity-check matrix for the code C, as follows. We assume that the parity-check matrix of C_1 has the systematic form $H_1 = [P \;\; I]$, where P is an $(n_1 - k_1) \times k_1$ matrix and I is the identity matrix of size $n_1 - k_1$. Let H_2 be a parity-check matrix of C_2. From the second property above, every codeword $c \in C$ must satisfy

$$c \begin{bmatrix} P^T & & \\ & \ddots & \\ & & P^T \end{bmatrix} H_2^T = 0.$$

The matrix H_2 can be divided into sub-blocks of size $(n_2 - k_2) \times (n_1 - k_1)$: $H_2 = [B_1 \;\; B_2 \;\; \cdots \;\; B_{n_2/(n_1-k_1)}]$. Then the parity-check matrix of C is given by

$$H = [B_1 P \quad B_2 P \quad \cdots \quad B_{n_2/(n_1-k_1)} P].$$

When C_1 is a single-parity code, the code C is a special case of a tensor-product code [15] and an error-location code [16], [11]. In the general case, the code C can be viewed as a generalized tensor-product code [6]. We therefore refer to C as a *tensor-product parity code*. The message length k_1 of C_1 is called the *block size* of C.

B. Encoding

For simplicity, we assume that C_1 is a single-parity code. The following encoding method can be easily extended to the general case. First, observe that if H_2 is in the systematic form, columns $k_1, 2k_1, \ldots, (n_2 - k_2)k_1$ of H form the identity matrix. Thus, we can use H directly to compute the redundancy bits and insert them at the corresponding positions. To be more efficient, take $k_2 k_1$ information bits and divide them into k_2 blocks. Compute the parity of each block and encode the parity using C_2. Take the rest of the information bits ($(n_2 - k_2)(k_1 - 1)$ bits) and divide them into $n_2 - k_2$ blocks. Compute the parity of each block. The redundancy bits of the tensor-product parity code are the modulo-2 sum of these parity bits and the parity part of C_2.

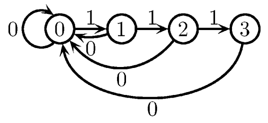

Fig. 1. Graph presentation of MTR(3).

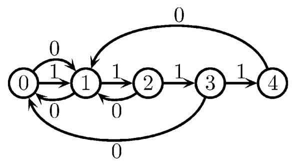

Fig. 2. Graph presentation of TMTR(3, 4).

C. Decoding

Decoding of the tensor-product parity-coded system begins with detection of the recorded bits using the Viterbi algorithm matched to the channel only. Then, the parity bits are computed using the encoder for C_1, and the resulting word is decoded using the decoder for C_2. The corrected parity bits are provided to a Viterbi detector reflecting channel states and parity code states, which computes the decoder output.

III. Constrained Codes

A constrained code is an important ingredient in almost every coding system for magnetic recording. It transforms the input into a sequence satisfying some desirable constraint. The most well-known constraint is the runlength-limited (RLL(d, k)) constraint, which limits the run-length of nontransitions to be at least d and at most k. The constraints considered in this paper are the maximum-transition-run (MTR) constraint and the time-varying version of this constraint (TMTR). The MTR(j) constraint limits the run-length of transitions to be at most j. The TMTR$(j, j+1)$ constraint limits the run-length of transitions to be at most j and $j+1$ at odd and even time indices, respectively.

A constraint is usually presented by a labeled directed graph (or graph, for short) $G = (V, E, L)$ consisting of a set of states $V = V_G$, a set of edges $E = E_G$, and an edge labeling $L = L_G$. Every sequence satisfying the constraint is the label of a path in the graph, and vice versa. The graphs presenting MTR(3) and TMTR(3, 4) constraints are shown in Figs. 1 and 2, respectively.

The MTR constraint has several desirable properties. It reduces the transition noise since it reduces the average number of transitions. Together with an appropriate detector, MTR(j) and TMTR$(j-1, j)$ codes reduce the occurrences of error events longer than j. Thus, an MTR code can enhance the effectiveness of a tensor-product parity code. With the TMTR(3, 4) code, most error events have length one, two, or three, which can be detected by a double-parity code.

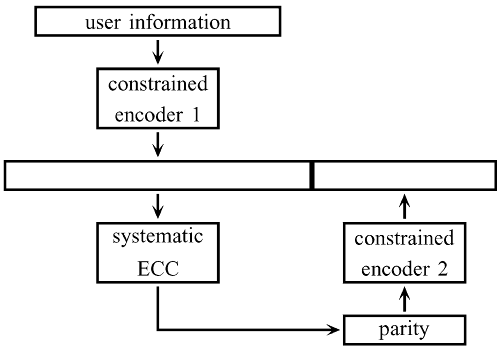

Fig. 3. Encoding diagram of the reverse concatenation scheme.

IV. Combining a Tensor-Product Parity Code With a Constrained Code

Traditionally, an error-correction code is combined with a constrained code in a standard concatenation architecture, with the error-correction code being the outer code and the constrained code being the inner code. This ensures that the recorded sequence satisfies the desired constraint. The drawback of this concatenation scheme is that the constrained code is decoded first, and it may cause error propagation that degrades the performance of the outer error-correction code. To combat this problem, a new concatenation scheme was proposed to bring the error-correction code "closer" to the channel [1], [9], [7], [5]. This so-called reverse concatenation encodes the user information using a constrained encoder first. The constrained sequence is then encoded by a systematic error-correction encoder. Finally, the parities generated by the error-correction encoder are encoded by a second constrained encoder, which usually has lower rate and reduced decoder error propagation. (The two constrained encoders are usually designed for the same constraint, and this is true for every coding system in this paper with multiple constrained codes.) The encoding process for this concatenation scheme is illustrated in Fig. 3. In the decoding process, the decoder for the second constrained code is applied first, followed by the error correction decoder, and finally the decoder for the first constrained code.

Our objective is to find a method for combining a tensor-product parity code and a constrained code that satisfies the following requirements: 1) The recorded sequence must be a codeword of both codes. This is essential since the tensor-product parity code will be decoded first. Moreover, to take full advantage of the constrained code, the constraint is incorporated into the trellis of our Viterbi detector, and hence the sequence must also satisfy the constraint. 2) The combined code must have a decoding algorithm that does not cause excessive error propagation. This is required because the output of the decoder will typically be processed by an RS decoder.

Unfortunately, direct application of the reverse concatenation scheme does not satisfy these requirements. Hence, we must seek alternative methods for combining the codes.

A. *Constrained Systems With Unconstrained Positions*

The first method that we use to combine a tensor-product parity code and a constrained code is based on the idea of constrained systems with unconstrained positions [14], [2], [13]. This method requires two constrained codes. The first code reserves some predetermined locations in the coded sequence, e.g., every tenth bit position, to be unconstrained. These positions can take on the value of 0 or 1 independently without violating the constraint. They will be replaced by the redundancy bits of the tensor-product parity code. The second code is a standard constrained code. The encoder for the first code is fed enough of the user information bits to ensure sufficiently many unconstrained positions for the redundancy bits of the tensor-product parity code. Then the encoder for the second code processes the rest of the user information bits. The tensor-product parity encoder will compute the redundancy bits and insert them into the unconstrained positions. The diagram in Fig. 4 shows the encoding process. The constraint assumed in the figure is MTR(2). The unconstrained positions are represented by the square symbol. Note that, unlike Figs. 1 and 2, this figure assumes that bits are in NRZ format.

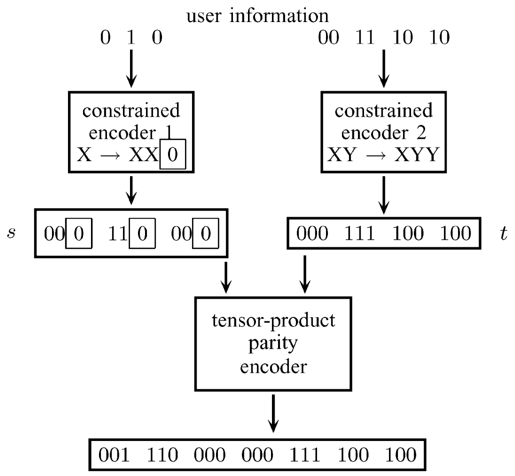

Fig. 4. Example of the unconstrained position scheme.

TABLE I
Capacities and the Highest Code Rates of the Systems With Unconstrained Position at Every Tenth Bit Position for Various MTR Constraints.

constraint	capacity	NRZI	NRZ
TMTR(2,3)	0.9163	0.8000	0.7762
MTR(3)	0.9468	0.8262	0.8195
TMTR(3,4)	0.9613	0.8456	0.8456
MTR(4)	0.9752	0.8639	0.8610
TMTR(4,5)	0.9818	0.8761	0.8710
MTR(5)	0.9881	0.8817	0.8804

The disadvantage of this method is that the overall rate is low. To see this, consider the examples in Table I. The second column of the table shows the capacities of the constraints, representing upper bounds on the code rates achievable when there are no unconstrained positions. The third and fourth columns show the highest achievable code rates if we insert an unconstrained bit in every tenth bit position, for NRZI and NRZ formats, respectively. As a specific example, consider the TMTR(2, 3) constraint, whose capacity is 0.9163. Since the unconstrained positions, which hold the redundancy bits of the tensor-product parity code, constitute only one-tenth of the coded sequence, the highest overall rate for a standard concatenation scheme is 0.8163. This is significantly higher than the maximum rates

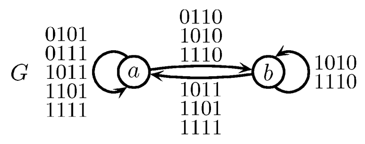

Fig. 5. Graph G for $S = \mathrm{RLL}(0,1)$ and $\ell = 4$.

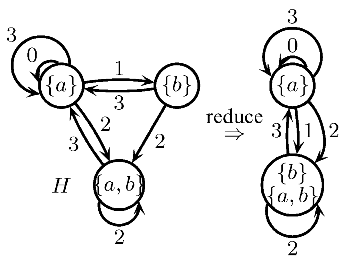

Fig. 6. Graph H and the reduced graph obtained from H.

achievable with this coding approach, namely 0.8 and 0.7762 for the NRZI and NRZ schemes, respectively.

B. Word-Set Partitioning

We now propose another combined coding method that can reduce the rate loss incurred by the unconstrained position method. We call this the word-set partitioning method. The description of this method requires some background on constrained coding, for which the reader is referred to [10].

Let S be a constraint. Suppose that the tensor-product parity code consists of the single parity code with block size ℓ. Let $G = (V_G, E_G, L_G)$ be an irreducible component of the ℓth power of a deterministic presentation of S. (We usually choose G to have the largest capacity among the irreducible components.)

Let W be the set of labels of G. Let W_0 and W_1 be ordered subsets of W such that 1) $|W_0| = |W_1| = m+1$, and 2) every word in W_0 has even parity and every word in W_1 has odd parity. Write W_0 as $(w_{0,0}, w_{0,1}, \ldots, w_{0,m})$ and W_1 as $(w_{1,0}, w_{1,1}, \ldots, w_{1,m})$. Let $w \in W$. We define $I(w)$ to be the set of initial states of the edges with label w. Assuming that G has memory one (which is true when S is an RLL(d,k), MTR(j) or TMTR$(j-1,j)$ constraint with $j,k \le \ell$), define $t(w)$ to be the terminal state of an edge with label w. This state is unique under the memory assumption. We define a graph $H = (V_H, E_H, L_H)$ as follows. First, let V_H be the set of nonempty subsets of V_G. Next, for each $i \in \{0,1,\ldots,m\}$, let $I_i = I(w_{0,i}) \cap I(w_{1,i})$ and $T_i = \{t(w_{0,i}), t(w_{1,i})\}$. Finally, assign an edge from every nonempty subset of I_i to T_i labeled by i.

Example 1: Let S be the RLL(0,1) constraint and $\ell = 4$. Fig. 5 shows the graph G.

A possible choice for W_0 and W_1 is the following: $W_0 = (0101, 0110, 1010, 1111)$ and $W_1 = (0111, 1110, 1011, 1101)$. From the above W_0 and W_1 and the graph G, we compute I_i and T_i, which are shown in Table II. The graph H is given in Fig. 6. We remark that by definition, the graph H is large since it has an exponential number of states. But we will only consider the constraint presented by H. Thus, it is sufficient to consider the reduced graph, which presents the same constraint but is typically much smaller.

TABLE II
CHOICE OF W_0 AND W_1 AND THE CORRESPONDING I_i AND T_i FOR RLL(0,1) AND $\ell = 4$

i	$w_{0,i}$	$w_{1,i}$	I_i	T_i
0	0101	0111	$\{a\}$	$\{a\}$
1	0110	1110	$\{a\}$	$\{b\}$
2	1010	1011	$\{a,b\}$	$\{a,b\}$
3	1111	1101	$\{a,b\}$	$\{a\}$

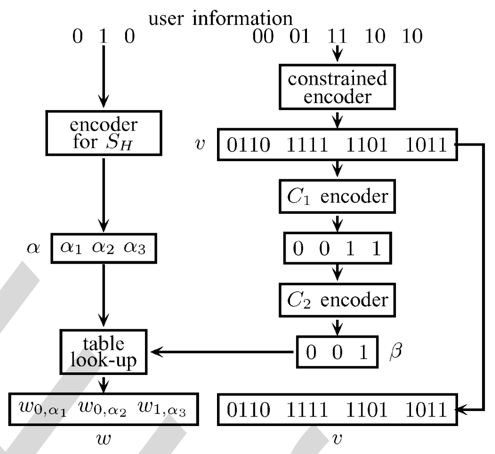

Fig. 7. Encoding algorithm for the word-set partitioning scheme.

Let $W_{01} = W_0 \cup W_1$ and $M = \{0,1,\ldots,m\}$. It is easy to see that there is a bijective mapping from $M^n \times \{0,1\}^n$ to W_{01}^n. More precisely, letting $\alpha = \alpha_1 \ldots \alpha_n \in M^n$ and $\beta = \beta_1 \ldots \beta_n \in \{0,1\}^n$, the following mapping is bijective:

$$(\alpha, \beta) \to w = w_{\beta_1,\alpha_1} \cdots w_{\beta_n,\alpha_n}.$$

(For the example above, the mapping is given in Table II.)

Let S_H be the constraint presented by H, and consider $\alpha \in M^n$ and $\beta \in \{0,1\}^n$. Using the mapping above, we claim that if α satisfies S_H, then w satisfies S. To see this, suppose that H has a path from a state A to a state C passing through B with label $\alpha_1\alpha_2$. Then G must have edges with labels w_{0,α_1} and w_{1,α_1} terminating at some states in B. In graph G, every state in B has outgoing edges with labels w_{0,α_2} and w_{0,α_1}. Thus, all words $w_{0,\alpha_1}w_{0,\alpha_2}$, $w_{0,\alpha_1}w_{1,\alpha_2}$, $w_{1,\alpha_1}w_{0,\alpha_2}$, and $w_{1,\alpha_1}w_{1,\alpha_2}$ satisfy S.

The encoding diagram for the word-set partitioning scheme is shown in Fig. 7. The algorithm requires a conventional constrained encoder for the original constraint S, an encoder for S_H, and a table look-up that implements the bijective mapping. First, the encoder for S_H takes part of the user information and produces a sequence α. Then the encoder for S takes the rest of the user information and produces a constrained sequence v. The parity β of the inner code C_2 is computed from v. Finally, the table look-up maps α and β to a sequence w. The sequence w satisfies S, as explained above. It also satisfies the parity condition by the properties of W_0 and W_1. The recorded sequence is wv. (Special care may be needed to guarantee that the constraint is satisfied at the boundary of w and v. If the encoders are properly designed, no extra bits are needed between w and v.)

In the word-set partitioning scheme, the set of words is divided into two partitions, W_0 and W_1, corresponding to odd

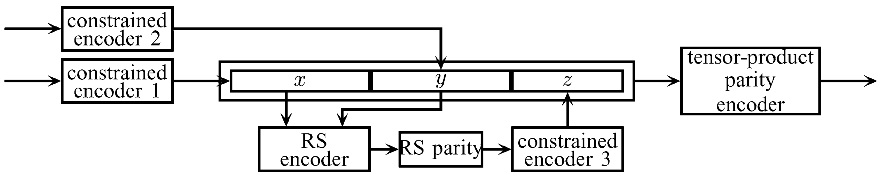

Fig. 8. Encoding diagram of the entire coding system.

parity and even parity. The approach can be easily generalized to four partitions, which will be equivalent to a double-parity code. In the most general setting, partitioning can be arbitrary. The partitions should be chosen so that an error usually corrupts a word from one partition to a word in another.

C. Incorporating the Reed–Solomon Code

The entire encoder including the RS code can be described by Fig. 8. For the unconstrained position scheme, constrained encoders 1 and 2 correspond to the respective encoders in Fig. 4. The outputs of these encoders are denoted by x and y. For the word-set partitioning scheme, constrained encoders 1 and 2 correspond to the encoder for S_H and the constrained encoder in Fig. 7, respectively. In this scheme, x is the sequence α. The RS code then encodes xy and produces its parity, which will be encoded by constrained encoder 3. This encoder has a low rate and does not allow error propagation since its output z will be decoded before the RS code. Finally the whole sequence xyz will be encoded by the tensor-product parity encoder. For the unconstrained position scheme, this step is the final step in Fig. 4, where x and yz correspond to s and t, respectively. For the word-set partitioning scheme, x and yz correspond to α and v. The tensor-product parity encoder computes β from yz and then uses the table look-up to find w from x and β.

V. Simulation Results

Our codes are tested in the perpendicular recording channel model shown in Fig. 9. The transition response is $s(t) = \tanh(\log(3)t/T_{50})$, where $T_{50} = 1.7$. The dibit response $h(t)$ is given by $s(t) - s(t-1)$. The jitter noise is modeled as $\delta_k * s'(k)$, where δ_k is normally distributed with mean zero and variance σ_j^2 if there is a transition and zero otherwise; s' is the derivative of s. The additive noise is normally distributed with mean zero and variance σ_a^2. The total noise power is defined as $\sigma^2 = \sigma_a^2 + \sigma_j^2\|s'\|^2$, and the ratio of the jitter noise to the total noise, $\sigma_j^2\|s'\|^2/\sigma^2$, is set to 0.9. The read signal is equalized to a DC-full target, $0.44 + 0.79\,D + 0.44\,D^2$.

We test six coding systems, denoted by RS-only, reverse, uncon1, uncon2, partition1, and partition2. These systems have approximately the same block length (4800 bits) and rate (0.854). The RS-only system consists of an RS code only. The reverse system consists of an RS code and a TMTR(3, 4) code, combined using reverse concatenation. The last four systems consist of an RS code, a TMTR(3, 4) code, and a tensor-product parity code. The parity code of the tensor-product parity code has a block size of 10 bits. The two uncon systems use the unconstrained position scheme to combine the codes. The parity codes of uncon1 and uncon2 are single- and double-parity codes, respectively. The two partition systems use the word-set partitioning scheme to combine the codes. The word sets of partition1 and partition2 are divided into two and four partitions, respectively. The parameters of the component codes are shown in Table III, where t denotes the correction power. Every RS code is over the field GF(1024). The BCH codes with lengths 480 and 960 are binary codes with underlying fields GF(512) and GF(1024), respectively. Not shown in the table is the constrained encoder 3 of the last four systems (see Fig. 8). This encoder is a block encoder with rate $10/11$.

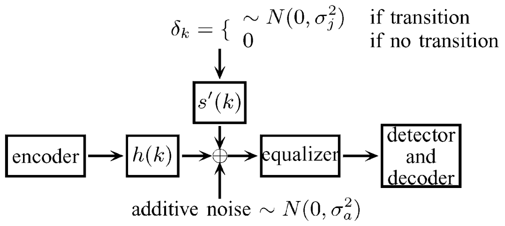
$\delta_k = \{ \begin{array}{l} \sim N(0,\sigma_j^2) \\ 0 \end{array}$
if transition
if no transition
$s'(k)$
encoder
$h(k)$
equalizer
detector and decoder
additive noise $\sim N(0,\sigma_a^2)$

Fig. 9. Perpendicular recording channel model.

TABLE III
Parameters of the six Coding Systems.

	outer	inner	constrained1	constrained2
RS-only	RS $(480,410)$ $t=35$			
reverse	RS $(475,433)$ $t=21$		rate $19/20$	rate $10/11$
uncon1	RS $(477,451)$ $t=13$	BCH $(480,318)$ $t=19$	uncon rate $42/50$	rate $19/20$
uncon2	RS $(478,464)$ $t=7$	BCH $(960,685)$ $t=28$	uncon rate $42/50$	rate $19/20$
partition1	RS $(477,451)$ $t=13$	BCH $(480,309)$ $t=20$	partition rate $17/20$	rate $19/20$
partition2	RS $(478,464)$ $t=7$	BCH $(960,665)$ $t=30$	partition rate $17/20$	rate $19/20$

The symbol error rates at the input to the RS decoders are plotted in Fig. 10. The symbol error rate of the RS-only system represents the uncoded error rate, while the symbol error rate of the reverse system represents the error rate when the TMTR constraint is imposed. The symbol error rates of the uncon1 and uncon2 systems correspond to the use of a TMTR code with single- and double-parity codes, respectively. At low SNR, the symbol error rates of the tensor-product parity codes suffer from the failure of the BCH codes. At extremely low SNR, the BCH codes almost always fail to recover the correct parity, and hence the symbol error rates are comparable to that of the TMTR code.

Fig. 11 shows sector error rates estimated by the block multinomial method [8]. For this channel, which is dominated

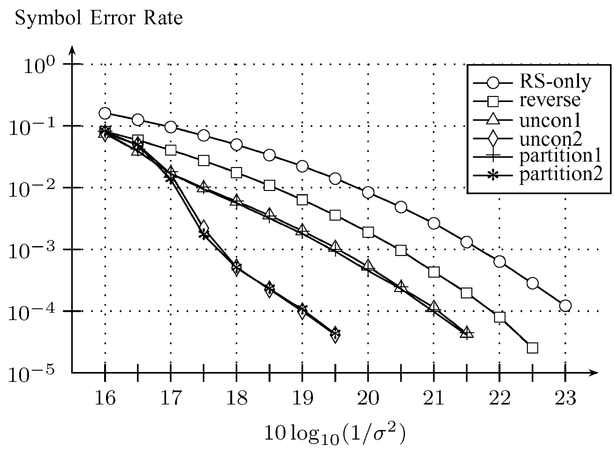

Fig. 10. Symbol error rate at the input to the RS decoder of each system.

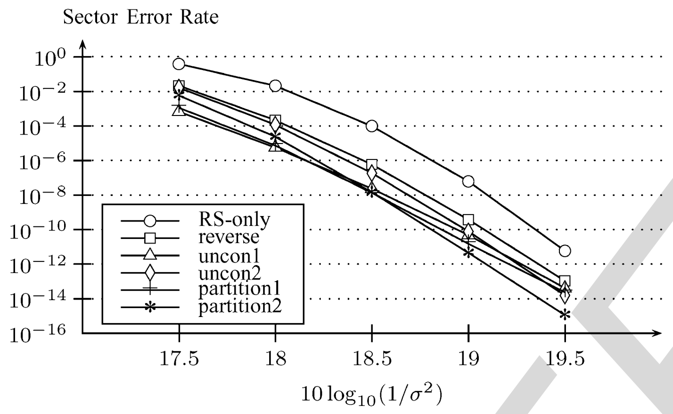

Fig. 11. Sector error rate of each coding system.

by jitter noise, the system benefits greatly from the use of the TMTR code, as can be seen by the 0.25 dB gain of the reverse system over the RS-only system. As much as 0.25 dB additional gain can be achieved by the tensor-product parity code. Since the word-set partitioning scheme is more efficient than the unconstrained position scheme, we can afford stronger BCH codes. Although there is little difference in the symbol error rates (Fig. 10), we can see some improvement in sector error rate with the word-set partitioning scheme.

Since the RS code in a tensor-product parity system is weaker, the system is more vulnerable to channel impairments that give burst errors. Hence, the actual gain may be smaller.

VI. Conclusion

The tensor-product parity code scheme helps to implement a parity code with less redundancy. We described two different methods to combine a tensor-product parity code with a constrained code and a Reed–Solomon code, and we simulated the performance of several combined coding systems on a perpendicular recording channel with jitter noise. The simulation results demonstrate that combined coding can provide some improvement over a system consisting of only a Reed–Solomon code.

Acknowledgment

This work was supported in part by the Information Storage Industry Consortium (INSIC), the Center for Magnetic Recording Research (CMRR), and the National Science Foundation (NSF) under Grant CCR-0219582. P. Chaichanavong was with the Center for Magnetic Recording Research, University of California at San Diego, during this work. The authors would like to thank the referees who provided comments that helped to improve the paper.

References

[1] W. G. Bliss, "Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation," *IBM Tech. Discl. Bull.*, vol. 23, pp. 4633–4634, 1981.
[2] J. C. de Souza, B. H. Marcus, R. New, and B. A. Wilson, "Constrained systems with unconstrained positions," *IEEE Trans. Inf. Theory*, vol. 48, no. 4, pp. 866–879, Apr. 2002.
[3] P. Chaichanavong and P. H. Siegel, "A tensor-product parity code for magnetic recording," *IEEE Trans. Magn.*, vol. 42, no. 2, Feb. 2006.
[4] T. Conway, "A new target response with parity coding for high density magnetic recording channels," *IEEE Trans. Magn.*, vol. 34, no. 4, pp. 2382–2386, Jul. 1998.
[5] J. L. Fan and A. R. Calderbank, "A modified concatenated coding scheme, with applications to magnetic storage," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 1565–1574, Jul. 1998.
[6] H. Imai and H. Fujiya, "Generalized tensor product codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 181–187, Mar. 1981.
[7] K. A. S. Immink, "A practical method for approaching the channel capacity of constrained channels," *IEEE Trans. Inf. Theory*, vol. 43, no. 5, pp. 1389–1399, Sep. 1997.
[8] Z. A. Keirn, V. Y. Krachkovsky, E. F. Haratsch, and H. Burger, "Use of redundant bits for magnetic recording: Single-parity codes and Reed–Solomon error-correcting code," *IEEE Trans. Magn.*, vol. 40, no. 1, pp. 225–230, Jan. 2004.
[9] M. Mansuripur, "Enumerative modulation coding with arbitrary constraints and post-modulation error correction coding for data storage systems," *Proc. SPIE*, vol. 1499, pp. 72–86, 1991.
[10] B. H. Marcus, R. M. Roth, and P. H. Siegel, *Handbook of Coding Theory*. Amsterdam, The Netherlands: Elsevier, 1998, ch. 20.
[11] J. Maucher, V. V. Zyablov, and M. Bossert, "On the equivalence of generalized concatenated codes and generalized error location codes," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 642–649, Mar. 2000.
[12] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Trans. Magn.*, vol. 32, no. 5, pp. 3992–3994, Sep. 1996.
[13] T. L. Poo, P. Chaichanavong, and B. H. Marcus, "Tradeoff functions for constrained systems with unconstrained positions," *IEEE Trans. Inf. Theory*, to be published.
[14] A. J. van Wijngaarden and K. A. S. Immink, "Maximum runlength-limited codes with error control capabilities," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 602–611, Apr. 2001.
[15] J. K. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 2, pp. 281–284, Apr. 1965.
[16] J. K. Wolf and B. Elspas, "Error-locating codes—A new concept in error control," *IEEE Trans. Inf. Theory*, vol. IT-9, no. 2, pp. 113–117, Apr. 1963.

Manuscript received July 1, 2005 (e-mail: panuc@marvell.com; psiegel@ucsd.edu).