RZ 3706 (# 99716) 04/04/08
Electrical Engineering 5 pages

Research Report

Reverse Concatenation of Product and Modulation Codes

Thomas Mittelholzer and Evangelos Eleftheriou

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Reverse Concatenation of Product and Modulation Codes

Thomas Mittelholzer and Evangelos Eleftheriou
IBM Zurich Research Laboratory
8803 Rüschlikon, Switzerland
e-mail: {tmi, ele}@ zurich.ibm.com

***Abstract*—Reverse concatenation (RC) architectures, which recently have been deployed in hard-disk-drive (HDD) products, offer crucial advantages in coding such as (i) avoiding error propagation through the modulation decoder, (ii) allowing the use of efficient high-rate modulation codes, and (iii) passing of soft information from the detector to the decoder, which facilitates parity-post processing and iterative coding schemes. In HDDs, error-correcting codes essentially consist of a single high-rate Reed–Solomon code, whereas in tape recording, large product codes are used that require a new RC architecture. Such a novel RC architecture for product codes is presented and illustrated by an example based on the Linear Tape Open Standard, Generation 4 (LTO-4). Compared with the rate-16/17 modulation code of the LTO-4 standard, the proposed RC scheme has a modulation scheme of rate 0.9951, i.e., achieves 5.7% improvement in rate while maintaining the same interleaved $I = 11$ modulation constraint, but at the cost of a slight weakening of the G-constraint.**

I. Introduction

Recently, reverse concatenation (RC) architectures have received increased attention, and read channel chips based on RC have already been implemented by the hard-disk-drive (HDD) industry. Reverse concatenation was introduced in [1]. In a RC scheme, the order of the error-correcting code (ECC) encoder and modulation encoder is reversed, i.e., the data is first passed through a modulation encoder and then ECC-encoded using a systematic encoder for the error-correcting code. The ECC parity symbols are either encoded using a second modulation code [1] or inserted into the data symbol stream at the bit [2] or symbol level [3]. There are three major benefits that make RC attractive:

- There is no error propagation through the modulation decoder.
- As error propagation is no issue, the first modulation code can be taken to be very long, allowing the use of capacity-efficient and high-rate modulation codes, thereby achieving code rate gains.
- In the read-back path, the ECC decoding block comes immediately after the channel detection block, i.e., soft information can be passed from the detector to the decoder on a bit-by-bit basis. This creates the appropriate framework for using novel ECC techniques that are based on turbo and LDPC codes and hold the promise of large performance improvements. Furthermore, in this framework, parity post-processing schemes can easily be implemented.

These three benefits could also be exploited in the framework of tape recording or optical recording. However, the ECC used in HDDs has a different structure than the one used in tape recording. In HDDs, ECC is essentially based on a single high-rate Reed–Solomon (RS) code. Data storage systems, which use removable media and typically record mass data, such as tape drives and optical disks, rely on strong ECCs to ensure bit error rates below 10^{-17}. In particular, tape drives and CD devices employ powerful and complexity-efficient ECC, which are based on code concatenation of an outer C2 and an inner C1 code. This requires a new RC architecture for ECC based on product or concatenated codes, which will be presented in this paper.

II. Concatenated Codes

The product code specified in the LTO-4 standard [4] is a particular instance of a concatenated coding scheme [5], where both the inner and outer codes are RS-based codes of length 480 and 64, respectively (see Table I). A codeword is a 64 × 480 array of bytes, i.e., it contains 30,720 bytes, with 54×460= 24,840 data bytes, resulting in a code rate of 0.8086. More specifically, the outer C2 code is an $[N_2 = 64, K_2 = 54, d_2 = 11]$ RS code over the Galois field GF(2^8), where N_2 denotes the length, K_2 the dimension, and d_2 the minimum Hamming distance of the code. The inner C1 code is obtained by even/odd interleaving of an [240,230,11] RS code over GF(2^8).

TABLE I
LTO-4 Product Codeword Layout

	0	**1**	**. . .**	**458**	**459**	**460**	**. . .**	**479**
0	0	1	. . .	458	459			
1	460	461	. . .	918	919		**C1**	
⋮							**Parity**	
52							**Bytes**	
53	24380	24381	. . .	24838	24839			
54								
⋮		**C2**	**Parity**	**Bytes**				
63								

One can envision other concatenated ECC schemes in which, for example, C2 is a RS code and C1 is an LDPC or turbo code.

III. ECC AND MODULATION CODES

A. Forward Concatenation

In magnetic and optical recording, modulation codes are used to enable timing recovery from the read-back signal and to allow short path memories in the detector without substantial performance loss as well as to eliminate other undesirable patterns. Thus, in the write path prior to writing ECC-encoded data onto the medium, the data is passed through a modulation encoder as shown in Fig. 1.

A scheme in which the user data is first encoded by ECC and then passed through a modulation encoder will be called a

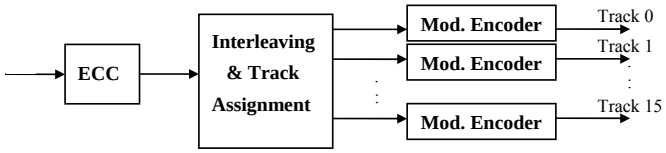

Figure 1. Forward concatenation architecture in LTO-4.

forward concatenation (FC) scheme. To improve ECC performance, there is a long block interleaver in the LTO-4 write path, denoted as Interleaving & Track Assignment block. This block buffers 64 consecutive product codewords, thereby accumulating a total of 64×64 = 4096 rows. These 4096 rows are assigned to the 16 tracks in a predefined order. For each track there is a rate-16/17 modulation encoder, which encodes the assigned rows and guarantees the predetermined modulation constraints, namely, a global G = 13 and an interleaved I = 11 constraint. Thus, prior to precoding, the maximum length of zeros in the coded sequences is limited to G and the maximum length of zeros in both the even and odd interleaves of the coded sequences is limited to I.

B. Reverse Concatenation

RC has been proposed for one-dimensional ECC architectures, in which the ECC typically consists of a single code such as an RS or an LDPC code. The RC schemes known, such as [1]-[3],[6], have not addressed the particular issues that arise from ECC, which is based on concatenated or product codes. The output of the inner code is mapped to the tracks/channels, and thus all rows should satisfy a predetermined modulation constraint. Accordingly, one is faced with the following problem, which is illustrated by the LTO-4 product code (see Table I): Putting the modulation encoder in front of a systematic ECC encoder will result in only K_2 rows that meet the modulation constraint except for the C1-parity part. The remaining $N_2 - K_2$ rows, which consist of C2-parity bytes, do not meet the modulation constraint. The C1-parity part poses a minor problem because it can be treated separately as in the case of one-dimensional ECC. However, for the C2-parity part, no efficient solution has been proposed. Thus, a substantial number of rows would not satisfy a modulation constraint and hence would need further processing. If one were to follow a one-dimensional RC strategy, these rows would need to be passed through a second modulation encoder or be dealt with using a parity-insertion strategy. Both techniques would result in undesired features: i) A second modulation code would lead to error propagation and would not allow soft information to be passed from the channel detector to the ECC decoder on a bit-by-bit basis, and ii) partial symbol interleaving would lead to very bad performance in the case of a dead track because entire faulty rows would be subdivided and spread into other rows, causing many errors in many rows.

IV. REVERSE CONCATENATION FOR PRODUCT-CODE-BASED ECC

The new RC architecture is motivated by the observation that these problems can be overcome if the following condition holds: After modulation encoding with a first modulation code and outer C2 encoding, a pre-determined modulation constraint is enforced in all rows. If this condition is met, the problem of designing a RC scheme for a product code or, more general, a concatenated code will be reduced to designing an RC scheme with a (one-dimensional) single error-correcting code. After the C2 encoder, the C1 code would play the role of the one-dimensional ECC. Thus, one would encode the rows using C1 and either employ a second modulation code or use partial symbol interleaving to meet the modulation constraints for the C1-parity part.

We will show that this condition can be achieved by *reorganizing the unencoded user data* array such that its size is based on the length rather than the dimension of the outer code C2, by introducing a *formatting block* before the C2 encoder, and by *modifying the C2-encoding* procedure.

A. LTO-based RC Scheme

A suitable RC architecture for ECC based on product codes is illustrated in Fig. 2. This scheme will be explained by an example with LTO-4-like ECC/modulation parameters. A generalization of this scheme will be treated in the next subsection.

User data reorganization: In contrast to the usual encoding of the C2 code of length N_2, where the user data is organized in K_2 rows, where K_2 is the dimension of C2, the proposed unencoded user data array consists of N_2 rows, which are generated by a serial/parallel block. An example of such an unencoded user data array is given in Table II, which is a modification of the subdata set array of the LTO-4 standard; specifically, it contains 952 user bytes more than the LTO-4 subdata set does. Each row of the unencoder user data array is passed through a first modulation encoder (denoted as Mod Enc 1 in Fig. 2) and thus satisfies a modulation constraint at the input of the formatting block. At this point, the modulated user data array still contains N_2 rows, which are by a few bytes longer because of Mod Enc 1. For the LTO-4-like scheme, Mod Enc 1 is derived from a rate-223/224 modulation code with a global G = 14 and interleaved I = 7 constraint. This interleaved Fibonacci code was constructed based on techniques described in [6]. The first modulation code transforms

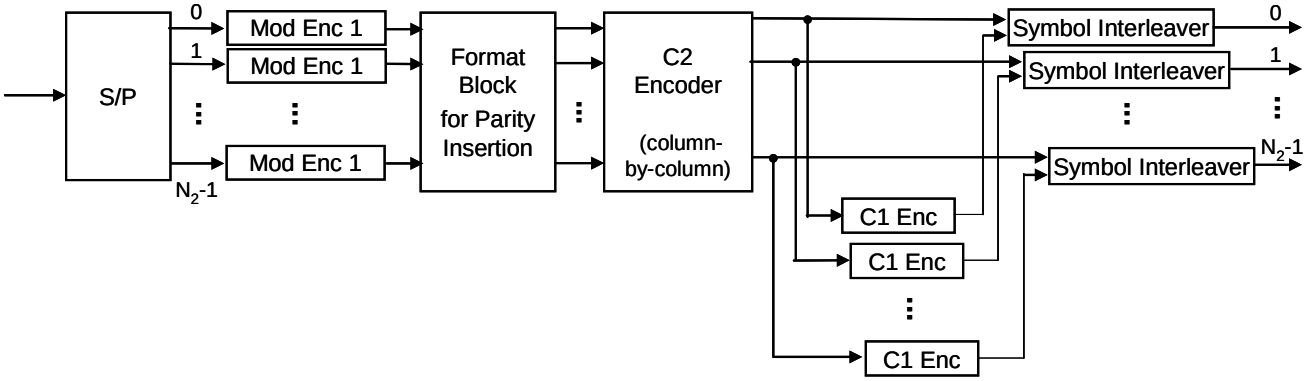

Figure 2. Reverse concatenation architecture for ECC based on product codes.

TABLE II
2D UNENCODED DATA LAYOUT FOR REVERSE CONCATENATION

	0	**1**	**2**	**. . .**	**402**
0	0	1	2	. . .	402
1	403	404	405	. . .	805
⋮					
52					
53	21359	21360	21361	. . .	21761
54					
⋮					
63	25389	25390	25391	. . .	25791

the unencoded user data array of size 64×403 into the modulated user data array of size 64×405.

Formatting block: The formatting block transforms the modulated user data array of size 64×405 into an array that has $N_2 - K_2$ "empty" components in each column. These $N_2 - K_2$ empty locations are the positions at which the parity symbols of the C2 code will be introduced. In the design phase of the formatting block, a parity pattern array is determined. Given the parity pattern array, the formatting block interleaves the empty cells into the modulated user data array row-by-row, thereby extending the length of the rows by some L bytes. This interleaving operation is similar to partial symbol interleaving [3],[6]; in particular, it weakens the modulation constraint of the first modulation code. In the above example, the insertion of unmodulated 8-bit parity symbols will weaken the global and interleaved constraints from $(G,I) = (14,7)$ to $(G,I) = (22,11)$.

To find a parity pattern array, the dimensions of the modulated user data array must satisfy a Diophantine equation, namely,

$$K_1 \times (N_2 - K_2) = N_2 \times L \tag{1}$$

where L is the number of C2-parity symbols per row and K_1 is the dimension of the C1 code on a C2-symbol basis (that is, the dimension of the C1 code must be expressed in C2-symbol units, e.g., in bytes). This Diophantine equation might necessitate an adjustment of the parameters of the C1 code. In the LTO-4-based example above, $K_1 = 480$ satisfies the equation with $L = 75$ C2-parity bytes positions in each row. Furthermore, the parity bytes should be separated by a predetermined minimum amount to prevent a destruction of the modulation constraint of the first modulation code. In the example, a spacing by at least two byte locations is sufficient to obtain a $(G,I) = (22,11)$ constraint. As there are 64 rows in each parity pattern array, there is a total of 64×75 = 4800 C2-parity bytes per parity pattern array. The insertion locations for these 4800 C2-parity bytes are specified by the following 10 linear equations modulo $N_2 = 64$, which relate the column indices x to the row indices y:

$$y \equiv x + c_i \pmod{N_2}, \tag{2}$$

where $c_i \in \{0, 6, 13, 19, 26, 32, 38, 45, 51, 58\}$ and $0 \le x < 480$. The parity pattern was selected such that each column contains $N_2 - K_2 = 10$ parity locations. Moreover, the offsets c_i have been selected to obtain an essentially even partition of the interval $[0, N_2 - 1]$. This results in an essentially even distribution of the L parity locations along each row and thus guarantees the $(G,I) = (22,11)$ constraint after parity insertion.

Column-dependent C2 encoding: The C2 code is typically an RS code, but other codes are also possible. However, the code should be a maximum-distance separable code, which has the useful property that every set of K_2 components forms an information set [7]. In other words, every set of K_2 components uniquely determines the remaining $N_2 - K_2$ parity symbols. At the input of the C2 encoder, every column contains K_2 modulated data bytes and $N_2 - K_2$ empty parity locations. In each column, the C2 encoder determines the $N_2 - K_2$ parity bytes from the K_2 modulated data bytes and inserts them at the empty parity locations. The output of the C2 encoder is a C2-encoded array of size $N_2 \times K_1$. Moreover, the C2-encoded array satisfies a predetermined modulation constraint along each row. Thus, the C2-encoded array, at the output of the column-dependent C2 encoder, meets the desired condition mentioned at the beginning of Section IV.

Note that — despite the column-dependency of the encoder — each column is a codeword of the same C2 code. Thus, during read-back, the error-correction algorithm is the same for

all columns, which is desirable from a complexity point of view.

C1 encoding and final modulation coding: The rows of the C2-encoded array are passed through an encoder for the C1 code. The resulting C1-parity bytes are either processed by a second modulation encoder [1] or partially interleaved (bit- or byte-wise) into the data stream of the C1 encoder, as shown in Fig. 2 [2],[3]. In the above LTO-4-based example, the C1 code of dimension $K_1 = 480$ and length $N_1 = 500$ is obtained as an even/odd-interleaved RS code of dimension 240 and length 250 over GF(2^8). Instead of placing the 20 parity bytes at the end of each row, they are interleaved into the modulated byte stream. In each row, the 20 insertion points for the C1-parity bytes are chosen such that they lie evenly between the last 21 C2-parity bytes. In this way, all parity bytes in a row are surrounded by at least two modulated bytes. This property is crucial to guarantee an $(G,I) = (22,11)$ constraint after parity insertion provided that the first modulation code satisfy a $(G,I) = (14,7)$ constraint before parity insertion.

In the LTO-based RC scheme, each row contains 403×8 = 3224 data bits as well as 16 additional bits from the first modulation code. Thus, the rate of the RC modulation scheme is 0.9951, whereas the rate of the LTO-4 modulation code is 16/17. Hence, the RC scheme has a 5.73% higher rate than LTO-4 does. Moreover, the RC scheme satisfies the same $I = 11$ constraint as LTO-4, but with $G = 22$ has a weaker constraint than the $G = 13$ in LTO-4.

B. *Generalized RC Scheme*

The Diophantine equation (1) imposes limitations on the choice of the parameters of the C1 and C2 code, which can make it impossible to meet specific ECC design targets. Thus, it is desirable to have a more general RC scheme with fewer limitations. The generalized RC scheme is based on the same design principles as the LTO-based RC scheme, namely:

1. The primary modulation code supports sparse parity insertions at all locations.
2. The format block shifts the bytes in each row to match a suitable parity insertion pattern.
3. C2 encoding is performed column-by-column with varying parity locations.
4. C1 encoding generates parity bytes at the end of each row.
5. The C1-parity bytes are "modulation encoded" by applying partial symbol interleaving.

The new feature of this scheme is that the parity insertion pattern will be used to insert both parity symbols and unconstrained user symbols, i.e., symbols that are not encoded by the first modulation encoder (see Fig. 3). The insertion of the unconstrained data bytes has to take place before the C2 encoder because the C2 encoder needs these bytes for computing the parity bytes. The unconstrained data bytes are generated by a de-multiplexer. The de-multiplexer splits the user data into a part consisting of U bytes per row that is processed by the first modulation encoder, and a second part consisting of D bytes per row that is processed by the insertion block before the C2 encoder. On each row, the first modulation encoder encodes the U bytes (together with a few optional padding bits) into M bytes and, thereby, enforces a tight modulation constraint, which supports partial symbol interleaving. The corresponding Diophantine equation for the number of "empty" locations in each data array is

$$(M+L) \times (N_2 - K_2) = N_2 \times (L - D) \,, \qquad (3)$$

where L is the number of "empty" locations per row and $K_1 = L + M$ is the dimension of the C1 code. Note that (3) offers more design flexibility than (1) does.

The generalized RC scheme is illustrated by the following example. The RC scheme uses as C2 code an RS code with parameters $[N_2 = 96, K_2 = 81, d_2 = 16]$ over the Galois field GF(2^8). Each data array contains $N_2 \times (U + D) = 96 \times 399 = 38{,}304$ bytes of user data, which are split into $N_2 \times U = 96 \times 394 = 37{,}824$ bytes that are encoded by the first modulation encoder and $N_2 \times D = 96 \times 5 = 480$ bytes that are processed by the insertion block before the C2 encoder. Thus, the unencoded user data array has size $N_2 \times U = 96 \times 394$. Each row of this array is encoded by a rate-197/200 interleaved Fibonacci code with global $G = 10$ and interleaved $I = 5$ constraint. Applying the rate-197/200 modulation encoder 16 times per row, each row of the unencoded subdata set with its $8 \times 394 = 3152$ bits is mapped into a row of the modulated user data array of size $N_2 \times M = 96 \times 400$.

The formatting block transforms the modulated user data array into an array, which has $N_2 - K_2 + 1 = 16$ "empty" components in each column. One of these 16 empty locations will be filled by an unconstrained data byte, and $N_2 - K_2$ empty

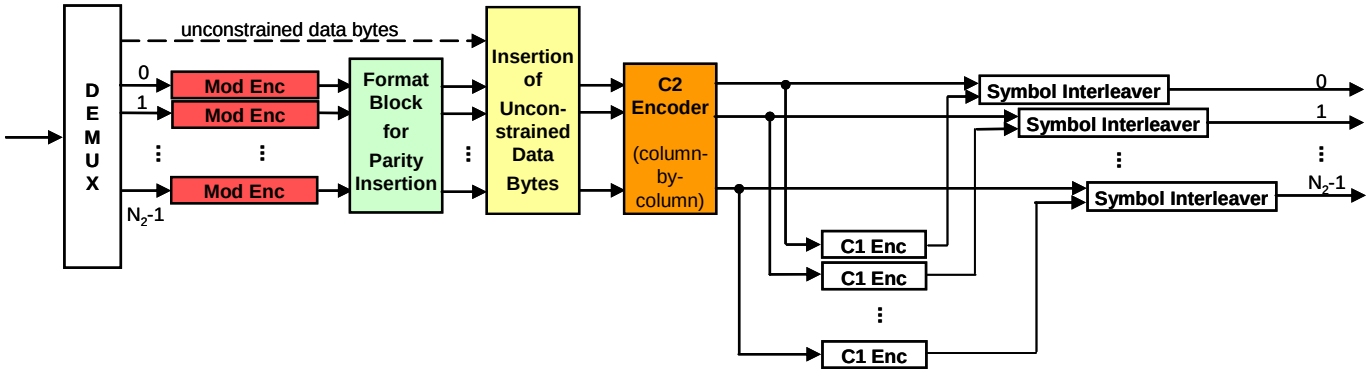

Figure 3. Generalized reverse concatenation architecture for ECC based on product codes.

locations will be filled with C2-parity bytes. To meet the Diophantine equation (3) with $M = 400$, the number of "empty" locations per row must be $L = 80$, resulting in $K_1 = L + M = 480$ for the dimension of the C1 code in terms of bytes. As there are 96 rows in each parity pattern array, there is a total of $96 \times 80 = 7680$ empty locations per parity pattern array. The 7680 insertion locations are specified by the following 16 linear equations (modulo 96), which relate the column indices x to the row indices y:

$$y \equiv x + c_i \pmod{96}, \tag{4}$$

where $c_i = 6i$ for $i = 0, 1, 2, ..., 15$ and $0 \le x < 480$.

The unconstrained $N_2 \times D = 96 \times 5 = 480$ data bytes are inserted into all of the 480 empty locations in the first six rows of the array, i.e., at locations specified by the above equations subject to the additional condition $0 \le y < 6$ on the row index. Note that per column exactly one unconstrained data byte is inserted and that there remain $N_2 - K_2 = 15$ empty locations to be filled with C2-parity bytes.

At the input of the C2 encoder, every column contains $K_2 = 81$ modulated or unconstrained data bytes and $N_2 - K_2 = 15$ empty parity locations. In each column, the C2 encoder determines the $N_2 - K_2 = 15$ parity bytes from these $K_2 = 81$ bytes and inserts them at the empty parity locations. The output of the C2 encoder is a C2-encoded array of size $N_2 \times K_1 = 96 \times 480$, which is shown in Table III, where the locations of the C2-parity bytes are denoted by 'p' and the previously inserted unconstrained data bytes are denoted by 'd'. Note that one needs at most six different column-dependent C2 encoders because the parity pattern repeats every sixth column. The insertion of parity and unconstrained data bytes weakens the $I = 5$ and $G = 10$ modulation constraints of the first modulation code along each row to $I = 9$ and $G = 18$ after C2 encoding. Finally, the C1 code and the partial symbol interleaving scheme are selected as in the LTO-based RC scheme described above. As a result, the overall scheme enforces in an $I = 9$ and $G = 18$ constraint throughout all rows.

TABLE III
2D C2-ENCODED DATA LAYOUT FOR REVERSE CONCATENATION

	0	1	2	3	4	5	6	7	. . .	479
0	d						d			
1		d						d	. . .	
2			d							
3				d						
4					d					
5						d				d
6	p						p			
7		p						p		
8			p							
9				p						
10					p					
11						p				p
12	p						p			
13		p						p		
⋮										⋮
95						p			. . .	p

V. CONCLUSION

A novel RC architecture for concatenated codes has been presented that is based on (i) a reorganization of the unencoded data array, (ii) a formatting block to perform interleaving of a suitably generated C2-parity pattern, (iii) column-dependent C2 encoding, and (iv) modulation coding of the C1-parity symbols based on a systematic modulation encoder.

The new RC architecture has been illustrated using a LTO-4-like example. Compared with the rate-16/17 code of the LTO-4 standard, the new RC scheme has a modulation scheme of rate 0.9951, which is a 5.7% improvement in rate, while maintaining the same $I = 11$ constraint and weakening the G-constraint from 13 to 22. The additional flexibility of a more general RC scheme was illustrated by a second example.

Moreover, the RC scheme presented supports the use of LDPC or turbo codes for the inner C1 code, which hold the promise of large performance improvements. In particular, the C1/C2-based ECC structure is an ideal setting for LDPC or turbo codes because the typical error floor issue that comes along with these codes is resolved by the outer C2 RS code, which can reduce the error rates to the desired 10^{-17} level.

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, Mar. 1981.

[2] A.J. van Wijngaarden and K.A.S. Immink, "Maximum runlength-limited codes with error control capabilities," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 602-611, Apr. 2001.

[3] M. Blaum, R. Cideciyan, E. Eleftheriou, K. Lakovic, T. Mittelholzer, and B. Wilson, "High-rate modulation codes for reverse concatenation," *IEEE Trans. Magn.*, vol. 43, no. 2, part 2, pp. 740-743, Feb. 2007.

[4] Linear Tape Open Standard, Ultrium Generation 4, 16-Channel Format Specification Document U-416, Revision D, December 25, 2006.

[5] I. Dumer, Concatenated Codes and Their Multilevel Generalizations, Chapter 23 in *Handbook of Coding Theory*, Vol. II, Eds.: V.S. Pless and W.C. Huffman, Elsevier 1998.

[6] M. Blaum, R. Cideciyan, E. Eleftheriou, K. Lakovic, T. Mittelholzer, and B. Wilson, "Enumerative encoding with non-uniform modulation constraints," *Proc. IEEE Intl Symp. on Information Theory*, Nice, France, June 2007, pp. 1831-1835.

[7] V.S. Pless, W.C. Huffman, and R.A. Brualdi, An Introduction to Algebraic Codes, Chapter 1 in *Handbook of Coding Theory*, Vol. I, Eds.: V.S. Pless and W.C. Huffman, Elsevier 1998.