Error-Correction of Multidimensional Bursts

Tuvi Etzion, *Fellow, IEEE*, and Eitan Yaakobi, *Student Member, IEEE*

***Abstract*—We present several methods and constructions to generate binary codes for correction of a multidimensional cluster-error, whose shape can be a box-error, a Lee sphere error, or an error with an arbitrary shape. Our codes have very low redundancy, close to optimal, and a large range of parameters of arrays and clusters. Our main results are summarized as follows.**

1) **A construction of two-dimensional codes capable to correct a rectangular-error with considerably more flexible parameters from previously known constructions. This construction is easily generalized for D dimensions.**
2) **A novel method based on D colorings of the D-dimensional space for constructing D-dimensional codes correcting a D-dimensional cluster-error of various shapes.**
3) **A transformation of the D-dimensional space into another D-dimensional space in a way that a D-dimensional Lee sphere is transformed into a shape located in a D-dimensional box of a relatively small size.**
4) **Applying the coloring method to correct more efficiently a two-dimensional error whose shape is a Lee sphere.**
5) **A construction of D-dimensional codes capable to correct a D-dimensional cluster-error of size b in which the number of erroneous positions is relatively small compared to b.**
6) **We present a code which corrects a D-dimensional arbitrary cluster-error with relatively small redundancy.**

***Index Terms*—Burst-error, burst-locator code, cluster-correcting code, coloring, Lee sphere, multidimensional code.**

I. Introduction

In current memory devices for advanced storage systems the information is stored in two or more dimensions. In such systems errors usually take the form of multidimensional bursts. These applications include optical recordings such as page-oriented optical memories [1], and volume holographic storage [2]–[4]. Usually, a cluster of errors either will be affected by the position in which the error event occurred or will be of an arbitrary shape. But, since an arbitrary cluster-error is hard to correct efficiently it is common to assume some type of cluster-error (as any arbitrary cluster is located inside a cluster with a certain shape, e.g., any two-dimensional cluster is located inside a rectangle). These types of errors can be of specific shapes like rectangles or Lee spheres. We will consider these types of errors as well as arbitrary cluster-errors. The main measure to compute the efficiency of a cluster-error correcting code is its redundancy. If we want to design a code which corrects one multidimensional cluster-error with volume B (of an arbitrary or a specific shape) then the redundancy r of the code satisfies $r \geq 2B$. This is a Singleton-type bound proposed by Bossert and Sidorenko [5]. This bound, known for the one-dimensional case as the Reiger bound [6], is attained for binary two-dimensional codes, which correct a rectangular error, constructed recently by Boyarinov [7]. If the volume of the array is N then the redundancy of the code must also satisfy $r \geq \log_2 N + B - 1$ (usually $r - \lceil \log_2 N \rceil \geq B$). The difference $r - \lceil \log_2 N \rceil$ will be called the *excess redundancy* of the code [8], [9] (even so our definition is slightly different). Abdel-Ghaffar [8] constructed a binary two-dimensional code which corrects a burst with a rectangle shape for which $r = \lceil \log_2 N \rceil + B$. The code has a few disadvantages: very limited size, complicated construction, and there is no obvious generalization for higher dimensions.

Our goal is to design codes which are capable to correct a cluster-error whose shape is a box, a Lee sphere, or an arbitrary shape. The method should be able to work on two-dimensional codes and multidimensional codes, and the parameters of the size of the codewords and the size of the cluster are as flexible as possible. There will be a price for our flexibility and our ability to generalize a two-dimensional construction into multidimensional construction. This price will be in excess redundancy. While the two-dimensional codes which correct a rectangle-error of Abdel-Ghaffar [8] have optimal excess redundancy, the excess redundancy of our codes is only close to optimality. Moreover, the novel methods enable us to correct a cluster whose shape is a Lee sphere and an arbitrary cluster, with excess redundancy close to optimal or very low, depending on the exact parameters. Previous to our methods, the way to correct such a cluster-error was to use a code which corrects a box-error in which the cluster-error is located, a method for which the excess redundancy is far from optimality.

The rest of the paper is organized as follows. In Section II, we briefly survey some of the known constructions which are essential to understand our results. In Section III, we present a construction for codes which correct a multidimensional box-error. The construction is a generalization and a modification of the construction of Breitbach, Bossert, Zyablov, and Sidorenko [10] for correction of bursts of size $b_1 \times b_2$. Better codes are constructed when the volume of the D-dimensional box-error is an odd integer. These constructions and the constructions which follow use auxiliary codes, called component codes, one code for each dimension. The key in our constructions is a new type of component code called a locator code. This code is locating the position of an error given its shape up to a cyclic shift. The power of this new tool is demonstrated in our constructions throughout the paper. In Section IV, we present a novel method for correction of a D-dimensional cluster. The construction uses

Manuscript received December 20, 2007; revised July 17, 2008. Current version published February 25, 2009. This work is part of E. Yaakobi's M.Sc. thesis work performed at the Technion–Israel Institute of Technology, Haifa. This work was supported in part by the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, under Grant 2006097.

T. Etzion is with the Department of Computer Science, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: etzion@cs.technion.ac.il).

E. Yaakobi was with the Department of Computer Science, Technion–Israel Institute of Technology, Haifa 32000, Israel. He is now with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: eyaakobi@ucsd.edu).

Communicated by L. M. G. M. Tolhuizen, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2008.2011520

D colorings of the D-dimensional space. The construction of Section III is a special case of this construction. The new construction enables us to handle different burst patterns. In Section IV, we use this method to handle D-dimensional box-error, where the volume of the box is an even integer. In Section V, we discuss how to correct a D-dimensional cluster-error whose shape is a Lee sphere. Two types of constructions are used. The first one uses a transformation of the D-dimensional space into another D-dimensional space in a way that each Lee sphere is transformed into a shape located inside a reasonably small D-dimensional box, so that we can use the constructions of the previous sections. The transformation is especially efficient for two dimensions. The second construction uses colorings as in Section IV. For two-dimensional array, the colorings that we use result in codes with excess redundancy close to optimality (where our measure for optimality is the lower bound on the excess redundancy), which improves on the construction obtained by the two-dimensional transformation. The generalization for multidimensional Lee sphere errors usually does not make the same improvement. This is also discussed in Section V. In Section VI, we first discuss bursts of size b, where the number of erroneous positions is limited. We present a construction of D-dimensional codes for this purpose. We combine the construction for codes which are capable to correct a Lee sphere error and a construction capable to correct a one-dimensional burst with a limited number of erroneous positions, to provide a construction of codes which correct arbitrary bursts. In Section VII, we describe codes with the same or slightly better parameters than the parameters of the codes from the previous sections by using parity-check matrices. Finally, a conclusion and a list of problems for further research are given in Section VIII. Before turning to our discussion we would like to mention that all our multidimensional codes are binary.

II. Known Constructions

Five constructions are important to understand our construction and their comparison with previous results.

- The Abdel-Ghaffar, McEliece, Odlyzko, and van Tilborg [11] construction of optimum binary cyclic burst-correcting codes.
- The Abdel-Ghaffar [12] construction of optimum cyclic burst-correcting codes over GF(q). A cyclic burst-correcting code $\mathcal{C}$ of length n, redundancy r, which corrects a burst of length b is called *optimum* if $n = \frac{q^{r-b+1}-1}{q-1}$. Throughout this paper, we need only the case $q = 2^\beta$ in our discussion.
- The Abdel-Ghaffar construction [8] of two-dimensional codes which correct a rectangular-error of size $b_1 \times b_2$ with excess redundancy $b_1 b_2$.
- The Breitbach, Bossert, Zyablov, and Sidorenko construction [10] of two-dimensional codes for correction of a $(b_1 \times b_2)$ rectangular-error by using vertical and horizontal component codes.
- The Abdel-Ghaffar, McEliece, and van Tilborg construction [9] of two-dimensional burst identification codes, which are used to identify the shape of an error and together with burst location codes are used for correction of a two-dimensional cluster.

Most of two-dimensional codes known in the literature are designed to correct a single cluster-error of size $b_1 \times b_2$ [7]–[10], [13] (only in some recent papers [14]–[16] it is assumed that the cluster-error can have an arbitrary shape). Two of these methods are important in our discussion. Abdel-Ghaffar [8] gave a construction of such $n_1 \times n_2$ code with excess redundancy $b_1 b_2$. One disadvantage of his method is that n_2 must be considerably larger than n_1 (with a possible exception when $b_2 \leq 2$, subject to a list of restrictive conditions), and the existence of the code depends on series of restricted conditions. The main goal of his construction was to show that for any given integers b_1 and b_2 there exists a cyclic $(b_1 \times b_2)$ cluster-correcting code of some size $n_1 \times n_2$ having optimal excess redundancy. Therefore, the size of the array was not a factor in his construction. His construction is a generalization of the optimum cyclic one-dimensional codes which correct a single cyclic burst of length b [11], [12]. Over GF(q) such code has length n, redundancy r, and it can correct a single cyclic burst of length $b \geq 1$, where $n = \frac{q^{r-b+1}-1}{q-1}$. The existence of such codes was obtained by the following necessary and sufficient conditions. For this purpose, the following three definitions are needed. A polynomial $f(x)$ is called *square-free* if $f(x)$ is not divisible by $(g(x))^2$ for nontrivial polynomial $g(x)$. The *period* h of the polynomial $f(x)$ is the least integer h such that $f(x)$ divides $x^h - 1$. Let $e(x)$ be a polynomial over GF(q) of positive degree. Let m_e be the least common multiple of the degrees of its irreducible factors over GF(q). We say that m_e is the *degree of the splitting field* of $e(x)$.

Theorem 1: [12] If a polynomial $g(x)$ generates an optimum b-burst-correcting code over GF(q), then it can be factored as $g(x) = e(x)p(x)$, where $e(x)$ and $p(x)$ satisfy the following conditions:

1) $e(x)$ is a square-free polynomial of degree $b-1$ which is not divisible by x such that h_e and m_e are relatively primes to $q-1$, where h_e and m_e are the period of $e(x)$ and the degree of the splitting field of $e(x)$, respectively;
2) $p(x)$ is an irreducible polynomial of degree $m \geq b+1$ and period $\frac{q^m-1}{q-1}$ such that m and $q-1$ are relatively primes and $m \equiv 0 \pmod{m_e}$.

A monic polynomial over GF(q) which satisfies condition 1) of Theorem 1 will be called a b-*polynomial* [12].

Theorem 2: [12] Let $e(x)$ be a b-polynomial over GF(q). Then, for all sufficiently large m relatively prime to $q-1$ such that $m \equiv 0 \pmod{m_e}$, where m_e is the degree of the splitting field of $e(x)$, there exists an irreducible polynomial $p(x)$ of degree m such that $e(x)p(x)$ generates an optimum b-burst correcting code of length $\frac{q^m-1}{q-1}$.

Remark: If the polynomial $p(x)$ in Theorems 1 and 2 is binary, then $p(x)$ in the theorems is a primitive polynomial [11].

The second method is due to Breitbach, Bossert, Zyablov, and Sidorenko [10], who gave three constructions of binary two-dimensional codes of size $n_1 \times n_2$ which correct a rectangular-error of size $b_1 \times b_2$. Their goal in presenting these constructions was not to obtain low excess redundancy, which is one of the goals in our constructions, but to present new constructions of

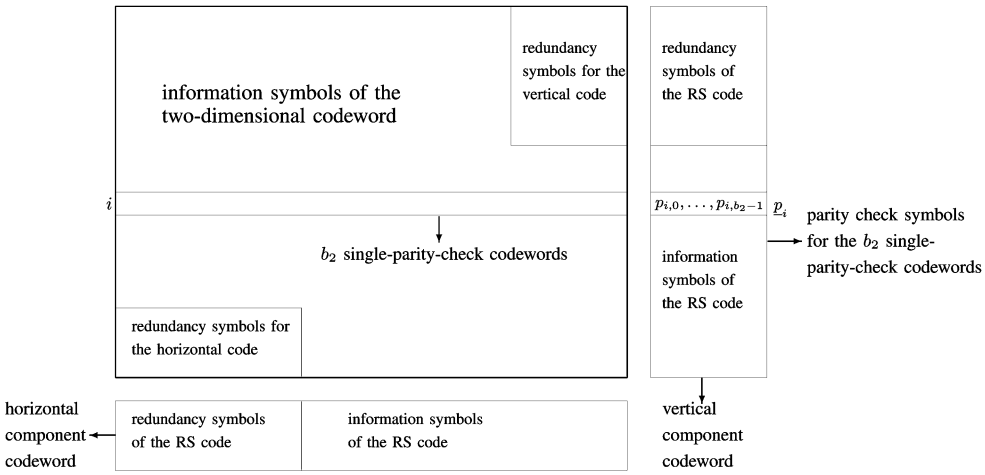

Fig. 1. Construction BBZS.

codes with relatively large array size and redundancy close to the Reiger's bound. We will use ideas from one of the constructions which will be called Construction BBZS.

A codeword $\{c_{ij}\}$ of the construction has size $n_1 \times n_2 = 2^{b_2} \times 2^{b_1}$ with $4b_1b_2$ redundancy bits located in positions $\{(i,j) : 0 \leq i \leq 2b_1 - 1, n_2 - b_2 \leq j \leq n_2 - 1\} \cup \{(i,j) : n_1 - b_1 \leq i \leq n_1 - 1, 0 \leq j \leq 2b_2 - 1\}$ (see Fig. 1). These bits are set initially to be zeros. Two temporary *component codes* are being used, a vertical code and a horizontal code (see Fig. 1). We will describe the construction of the vertical code. We note an earlier construction [17] which uses similar component codes.

For each row $i = 2b_1, \ldots, n_1 - 1$, b_2 parity-check bits are generated. $p_{i\ell}$, $\ell = 0, 1, \ldots, b_2 - 1$ is computed as

$$p_{i\ell} = \sum_{j=\ell, \ell+b_2, \ell+2b_2, \ldots, j<n_2} c_{ij}. \tag{1}$$

The parity bits $p_{i,\ell}$, $\ell = 0, 1, \ldots, b_2 - 1$ generate afterward a symbol $\underline{p}_i = (p_{i,0}, p_{i,1}, \ldots, p_{i,b_2-1})$ from the extension field GF(2^{b_2}). The symbols $\underline{p}_i$, $i = 2b_1, \ldots, n_1 - 1$ are considered as the information symbols of a Reed–Solomon (RS) code of length n_1, dimension $n_1 - 2b_1$, and minimum distance $d = 2b_1 + 1$. By the encoding procedure of the RS code we obtain $2b_1$ redundancy symbols $\underline{p}_i$, $i = 0, \ldots, 2b_1 - 1$, and the $2b_1b_2$ upper right corner redundancy bits of the array are computed in a way that (1) will hold for $i = 0, 1, \ldots, 2b_1 - 1$ and $\ell = 0, 1, \ldots, b_2 - 1$. The encoding procedure of the horizontal code is done in the same manner, where all the $4b_1b_2$ redundancy bits of the array are assumed to be zeroes.

Remark: It is possible to encode the horizontal code also with the new computed values of the $2b_1b_2$ upper right corner redundancy bits. We shall not do it as we want to follow the construction exactly as in [10]. There is no advantage of one method over the other. But, in our new encoding procedures in the following sections, we will have to encode the horizontal code when the other redundancy bits have their new computed values; otherwise these procedures will not work.

In the decoding procedure, each row generates b_2 parity bits according to (1) such that a word of length n_1 over GF(2^{b_2}) is received (the redundancy bits of the horizontal code are assumed to be zeroes). Assuming that the error occurred in the array can be confined inside a rectangle of size $b_1 \times b_2$. The generated word, of the vertical code, has at most b_1 erroneous symbols, which can be corrected by the decoding procedure of the RS code. The same process is implemented for the horizontal code. The positions of the erroneous elements in the vertical codeword induce the rows in which errors occurred in the array. The positions of the erroneous elements in the horizontal codeword induce the columns in which errors occurred in the array. Hence, we locate the positions of the $b_1 \times b_2$ cluster-error in the array. The shape of the cluster, up to b_2 horizontal cyclic shifts is found by the vertical code. The shape of the cluster up to b_1 vertical cyclic shifts is found by the horizontal code. As we know the location of the cluster, we can use one of the component codes to identify the exact shape of the error and to correct it.

Remark 1: Note that the vertical code cannot find errors inside the lower left corner redundancy bits. The horizontal code cannot find errors inside the upper right corner redundancy bits. But these facts do not affect the decoding procedure, i.e., the vertical code is able to know the erroneous rows even if some of the erroneous bits are the lower left redundancy bits.

Remark 2: A code obtained from construction BBZS is a linear code and can be described by a related generator matrix and parity-check matrix. One advantage will be that the redundancy bits will not be confined to the upper right corner and the lower left corner. It will simplify the construction, but in our opinion the understanding of the code, its encoding, and decoding, is much better when we consider the description as in this section and in [10]. This will be our point of view in the rest of this paper. In Section VII, we will discuss the representation of our code by parity-check matrices.

There is no obvious generalization of the construction of Abdel-Ghaffar [8] for multidimensional codes, while immediate

generalizations of Construction BBZS cannot support good redundancy or excess redundancy. One simple way to generalize this construction is to use the optimum burst-correcting codes of [11], [12] instead of the RS codes. The vertical component code over GF(2^{b_2}) has length $\frac{(2^{b_2})^{r_1-b_1+1}-1}{2^{b_2}-1}$, redundancy r_1, and it can correct a burst of length b_1. The horizontal component code over GF(2^{b_1}) has length $\frac{(2^{b_1})^{r_2-b_2+1}-1}{2^{b_1}-1}$, redundancy r_2, and it can correct a burst of length b_2. Instead of $4b_1b_2$ redundancy bits we will use r_1b_2 redundancy bits for the vertical code and b_1r_2 redundancy bits for the horizontal code. The excess redundancy of this construction is $2b_1b_2 - 1$ and the excess redundancy of its generalization for D dimensions is $DB - 1$, where B is the volume of the D-dimensional box error.

Further improvements of this construction are presented in Section III. Henceforth, we assume that if a D-dimensional code is discussed then D is a constant. Furthermore, we assume $b_i > 1$ for $1 \leq i \leq D$; this assumption can be made since if for some j, $b_j = 1$ then the cluster can be corrected as a $(D-1)$-dimensional cluster in a D-dimensional array.

Remark: A D-dimensional array can be always considered as a $(D-1)$-dimensional array by using unfolding. For example, a two-dimensional array can be considered as a one-dimensional array either by presenting it row by row or by presenting it column by column. Thus, the problem of correcting a $\tilde{D}$-dimensional cluster-error in a D-dimensional array, $\tilde{D} < D$, can be reduced to correcting a $\tilde{D}$-dimensional cluster in a $\tilde{D}$-dimensional array.

III. Construction for Multidimensional Arrays

In this section, we present our first idea for construction of multidimensional codes capable to correct a box-error of size $b_1 \times b_2 \times \cdots \times b_D$. First, we will present the two-dimensional version of the construction. We combine Construction BBZS with the constructions of Abdel-Ghaffar *et al.* [11] and Abdel-Ghaffar [12] to obtain codes with a variety of parameters. The redundancy of the construction is kept relatively small as our horizontal code will find only the location of the error and not its shape. This idea is the first key to all our constructions. The second idea to reduce the redundancy is to use a binary horizontal code instead of a code over GF(2^{b_1}). Finally, the structure of the construction makes it possible to generalize it to any dimension. The generalization is relatively quite simple, with low redundancy, and can be applied to a large range of parameters. One disadvantage is that the construction is defined for a box-error whose volume is an odd integer. To apply the construction to a box-error whose volume is an even integer, we have to increase the box-error artificially such that its volume will be an odd integer and the real box error will be located inside the artificial box error. This will cost us extra unnecessary redundancy. In Section III-A we will solve this problem by giving a novel construction for correction of a box-error whose volume is an even integer.

A. Two-Dimensional Codes

The vertical component code of Construction BBZS finds the rows in which the burst occurred and the shape of the cluster up to a cyclic permutation of the columns. Hence, the work done by the horizontal code to find the shape of the cluster is redundant. Therefore, we want to find a horizontal component code that will determine only the first column of the cluster. More explicitly, a burst $e = (e_0, e_1, \ldots, e_{b_2-1})$, where $e_i \in$ GF(2^{b_1}), for $0 \leq i \leq b_2 - 1$ found by the vertical code, can start at any column $0 \leq i \leq n_2 - b_2$ (see Fig. 2). However, if the first column of the cluster is i, then the cluster that occurred is $e' = (e_{i_0}, e_{i_0+1}, \ldots, e_{i_0+b_2-1})$ where $i_0 \equiv i \pmod{b_2}$, and indices are taken modulo b_2.

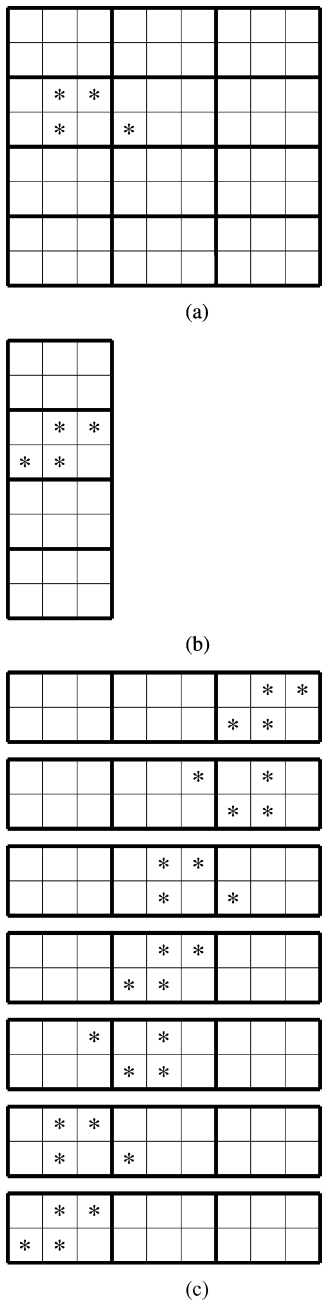

Fig. 2. (a) A cluster-error in a (2×3) burst-correcting code. The cluster found by the vertical code is demonstrated in (b), and (c) shows the possible clusters considered by the horizontal code.

Our new construction, in which the horizontal code only locates the first column of the cluster, is based on two lemmas. The first one is proved here only for the binary case.

Lemma 1: If $e_2(x) = 1 + x + x^2 + \cdots + x^{b-1}$ and b is an odd integer then $e_2(x)$ is a b-polynomial over GF(2).

Proof: Clearly, $e_2(x)$ is not divisible by x. The derivative of $x^b - 1$ over GF(2) is x^{b-1}, and since g.c.d. $(x^b - 1, x^{b-1}) = 1$, it follows that $x^b - 1$ is a square-free polynomial and hence $e_2(x)$ is also square-free. Therefore, by Theorem 1, $e_2(x)$ is a b-polynomial over GF(2). □

There is an alternative more general version of Lemma 1.

Lemma 2: Let $e_2(x)$ be the polynomial $e_2(x) = 1+x+x^2+\cdots+x^{b_2-1}$ over GF(2^{b_1}), where b_1 and b_2 are positive integers. Assume that the following conditions hold:

1) $\gcd(b_2, 2) = 1$,
2) $\gcd(b_2, 2^{b_1} - 1) = 1$,
3) $\gcd(\phi(b_2), 2^{b_1} - 1) = 1$.

Then, $e_2(x)$ is a b_2-polynomial over GF(2^{b_1}).

We omit the proof of the lemma (see [18]) as the construction which uses the lemma, and is described in this subsection, has inferior redundancy than the one described in the next subsection and uses Lemma 1. We will compare these redundancies in the sequel. The code over GF(2^{b_1}) is described since it is a bridging step to understand the one over GF(2).

By Theorem 2, for the b_2-polynomial $e_2(x) = 1 + x + x^2 + \cdots + x^{b_2-1}$, over GF$(2^{b_1})$, there exists an irreducible polynomial $p_2(x)$ of degree $m_2 = r_2 - b_2 + 1$ such that $e_2(x)p_2(x)$ generates an optimum b_2-burst-correcting code $\mathcal{C}^*$ of length $n_2 = \frac{(2^{b_1})^{m_2}-1}{2^{b_1}-1}$ and redundancy r_2. Let

$$\mathcal{C}^* = \{f(x) \in \text{GF}(2^{b_1})[x] : e_2(x)p_2(x) | f(x),\ \deg f(x) < n_2\}$$

and let

$$\mathcal{C}_2 = \{f(x) \in \text{GF}(2^{b_1})[x] : p_2(x) | f(x),\ \deg f(x) < n_2\}.$$

The code $\mathcal{C}_2$ is also of length n_2 and has m_2 redundancy symbols over GF(2^{b_1}). We will show now that $\mathcal{C}_2$ can serve as the horizontal component code, i.e., given that the cluster occurred up to a cyclic permutation is $e = (e_0, e_1, \ldots, e_{b_2-1})$, where $e_i \in \text{GF}(2^{b_1})$, for $0 \le i \le b_2 - 1$, it will be possible to determine the first column i, $0 \le i \le n_2 - b_2$, of the cluster with $\mathcal{C}_2$.

Lemma 3: Let $e = (e_0, e_1, \ldots, e_{b_2-1})$ be a given cluster, up to a cyclic permutation, which occurred in a transmitted codeword and found by the vertical component code. Then, the horizontal component code $\mathcal{C}_2$ can determine the first column of the given burst.

Proof: We have to prove that if the burst e, or a cyclic shift of e, occurred in two different codewords $f_1(x)$, $f_2(x)$ then two different words will be generated. Since $\mathcal{C}_2$ is a linear code, it is sufficient to prove that there is no codeword which is equal to the difference of two clusters which are cyclic shifts of the cluster e. Assume the first column of the cluster e is i, i.e., the cluster is $e' = (e_{i_0}, e_{i_0+1}, \ldots, e_{i_0+b_2-1})$ where $i_0 \equiv i \pmod{b_2}$ and indices are taken modulo b_2. The polynomial representing this cluster is

$$h_i(x) = x^{\ell_i b_2}\Big(e_{i_0}x^{i_0} + e_{i_0+1}x^{i_0+1} + \cdots + e_{b_2-1}x^{b_2-1} + e_0x^{b_2} + e_1x^{b_2+1} + \cdots + e_{i_0-1}x^{b_2+i_0-1}\Big)$$

where $\ell_i = \left\lfloor \frac{i}{b_2} \right\rfloor$. We can write $h_i(x)$ as

$$\begin{aligned} h_i(x) &= x^{\ell_i b_2}\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1} + e_0(x^{b_2} - 1) + e_1(x^{b_2+1} - x) + \cdots + e_{i_0-1}(x^{b_2+i_0-1} - x^{i_0-1})\right) \\ &= x^{\ell_i b_2}\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1} + (x^{b_2} - 1)(e_0 + e_1x + \cdots + e_{i_0-1}x^{i_0-1})\right) \\ &= x^{\ell_i b_2}\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1}\right) + x^{\ell_i b_2}(x^{b_2} - 1)\left(e_0 + e_1x + \cdots + e_{i_0-1}x^{i_0-1}\right). \end{aligned}$$

Assume the contrary, that the difference between two clusters which are cyclic shifts of the cluster e is a codeword. Assume that these two clusters start at columns i and j. Hence, the polynomial $h_i(x) - h_j(x)$ is the codeword

$$\begin{aligned} &h_i(x) - h_j(x) \\ &\quad = x^{\ell_i b_2}\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1}\right) + x^{\ell_i b_2}(x^{b_2} - 1)\left(e_0 + e_1x + \cdots + e_{i_0-1}x^{i_0-1}\right) \\ &\quad - x^{\ell_j b_2}\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1}\right) - x^{\ell_j b_2}(x^{b_2} - 1)\left(e_0 + e_1x + \cdots + e_{j_0-1}x^{j_0-1}\right). \end{aligned}$$

$h_i(x) - h_j(x)$ can be written as

$$\begin{aligned} &h_i(x) - h_j(x) \\ &\quad = (x^{\ell_i b_2} - x^{\ell_j b_2})\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1}\right) + (x^{b_2} - 1)\left(x^{\ell_i b_2}\left(e_0 + e_1x + \cdots + e_{i_0-1}x^{i_0-1}\right)\right. \\ &\qquad \left. - x^{\ell_j b_2}\left(e_0 + e_1x + \cdots + e_{j_0-1}x^{j_0-1}\right)\right) \\ &\quad = x^{\ell_j b_2}(x^{(\ell_i-\ell_j)b_2} - 1)\left(e_0 + e_1x + \cdots + e_{b_2-1}x^{b_2-1}\right) + (x^{b_2} - 1)\left(x^{\ell_i b_2}\left(e_0 + e_1x + \cdots + e_{i_0-1}x^{i_0-1}\right)\right. \\ &\qquad \left. - x^{\ell_j b_2}\left(e_0 + e_1x + \cdots + e_{j_0-1}x^{j_0-1}\right)\right). \end{aligned}$$

This last presentation of $h_i(x) - h_j(x)$ implies that it is dividable by $x^{b_2} - 1$ and hence also by $e_2(x) = \frac{x^{b_2}-1}{x-1}$. Since $h_i(x) - h_j(x)$ is a codeword in $\mathcal{C}_2$ it follows that $p_2(x) | h_i(x) - h_j(x)$. Since also $e_2(x) | h_i(x) - h_j(x)$, $p_2(x)$ is an irreducible polynomial, and its degree is greater than b_2 it follows that $e_2(x)p_2(x) | h_i(x) - h_j(x)$. Therefore, $h_i(x) - h_j(x)$ is also a codeword of $\mathcal{C}^*$, a contradiction since $\mathcal{C}^*$ can correct any burst of length b_2. □

A code $\mathcal{C}$ that can find the first column of a burst $e = (e_0, e_1, \ldots, e_{b-1})$ given up to a cyclic shift will be called a *b-burst-locator code*. Thus, by Lemma 3, $\mathcal{C}_2$ is a b_2-burst-locator code. Based on the constructions of [10], [11], and Lemma 3 we can construct an $n_1 \times n_2$ two-dimensional $(b_1 \times b_2)$ cluster-correcting code with small excess redundancy.

Let $\mathcal{C}_1$ be an optimum b_1-burst-correcting code, over GF(2^{b_2}), of length

$$n_1 = \frac{(2^{b_2})^{r_1-b_1+1} - 1}{2^{b_2} - 1}$$

and redundancy r_1. Let $\mathcal{C}_2$ be a b_2-burst-locator code, over GF(2^{b_1}), of length

$$n_2 = \frac{(2^{b_1})^{m_2} - 1}{2^{b_1} - 1}$$

and redundancy $m_2 = r_2 - b_2 + 1$. We can give a construction in which each codeword of size $n_1 \times n_2$ has $r_1 b_2 + m_2 b_1 + 1 = \lceil \log_2(n_1 n_2) \rceil + b_1 b_2 + b_1$ redundancy bits. The redundancy bits are partitioned into three subsets.

- $r_1 b_2$ redundancy bits are located in the upper right corner of the array and are computed from the complete vertical codeword as done in Construction BBZS.
- $m_2 b_1$ redundancy bits which are computed from the complete horizontal codeword as done in Construction BBZS. These redundancy bits are spread in the array in a way that they will fulfill the following requirement. If a redundancy bit is erroneous it will be possible to determine its row (note, that if the vertical code finds only one row where errors occurred, there are b_1 different sets of b_1 rows in which the burst occurred). Hence, in $2b_1 - 1$ consecutive rows there can be at most one redundancy bit. This requirement implies also that in any $(b_1 \times b_2)$-cluster we have at most one redundancy bit.
- One redundancy bit which is a parity of all redundancy bits of the second subset. Its role is to determine whether this bit or a redundancy bit from the second subset is erroneous.

The encoding is done similarly to Construction BBZS with two exceptions. When we compute the elements of the horizontal component code, the $r_1 b_2$ redundancy bits of the first subset are not assumed to be zeros as in Construction BBZS, but have the values which were computed by the previous steps of the encoding procedure. The second exception is the extra computation of the redundancy bit of the third subset, which is taken as an even parity bit of all the redundancy bits of the second subset.

The decoding is done similarly to Construction BBZS with the following exceptions. In Construction BBZS, if redundancy bits are erroneous then they will be recovered by the corresponding component code. The redundancy bits in the right upper corner are recovered by the vertical component code. They will be recovered by this code also in our construction. The redundancy bits in the left lower corner, of Construction BBZS, are recovered by the horizontal component code in Construction BBZS. Since, in our construction we do not use a burst-correcting code as a horizontal component code we cannot use this code to recover the related redundancy bits of the second subset. Each $b_1 \times b_2$ subcodeword can contain at most one redundancy bit from the second or third subset. By summing all these redundancy bits we will know if one of them is erroneous. Also, these redundancy bits are spaced in a way that if we know in which rows errors occurred then we will know which one of these bits is in error. Once we will find this erroneous bit we will know the shape of the burst and the horizontal burst-locator code will find the column in which the cluster started. If only a redundancy bit from the second or third subset is in error then the vertical code will not find erroneous bits. In this case, the sum of these bits is odd. If a bit from the second subset is in error then the horizontal burst-locator code will correct this error since this code is generated by a primitive polynomial and hence it can also correct a single error. Otherwise, the horizontal code will not find an error, which implies that the redundancy bit of the third subset is erroneous.

B. Binary Burst-Locator Code

The redundancy of the construction is improved if we use as the horizontal burst-locator code a binary $(b_1 b_2)$-burst-locator code $\mathcal{C}$ of length $2^m - 1$, where $m = r - b_1 b_2 + 1$, $b_1 b_2$ odd, and $e_2(x) = 1 + x + \cdots + x^{b_1 b_2 - 1}$. This is done simply by taking the b_1 parity symbols which are computed for each column as b_1 consecutive symbols in $\mathcal{C}_2$ instead of an element in GF(2^{b_1}).

Each codeword $\{c_{ij}\}$ of size $n_1 \times n_2$ has $r_1 b_2 + m + 1$ redundancy bits in the following positions.

- $\{(i,j)\colon 0 \le i \le r_1 - 1, n_2 - b_2 \le j \le n_2 - 1\}$. These redundancy bits are computed from the vertical component code.
- $\{(i(2b_1 - 1) + j(2b_1 - 1)b_1, j)\colon 0 \le i \le b_1 - 1,\ 0 \le j,\ (j+1)b_1 + i + 1 \le m\}$. These redundancy bits are computed from the horizontal component code.
- $\{(n_1 - 1, n_2 - 1)\}$. This redundancy bit is an even parity bit for the redundancy bits of the second subset.

Encoding:

All the redundancy bits in a codeword are set initially to be zeroes. The vertical component code and the first set of redundancy bits are computed as in Construction BBZS, i.e., for each row $i = r_1, \ldots, n_1 - 1$, b_2 parity-check bits are generated. $p_{i\ell}$, $\ell = 0, 1, \ldots, b_2 - 1$ is computed as

$$p_{i\ell} = \sum_{j = \ell, \ell + b_2, \ell + 2b_2, \ldots, j < n_2} c_{ij}. \tag{2}$$

The parity bits $p_{i\ell}$, $\ell = 0, 1, \ldots, b_2 - 1$ generate afterward a symbol $\underline{p}_i = (p_{i,0}, p_{i,1}, \ldots, p_{i,b_2 - 1})$ from the extension field GF(2^{b_2}). The symbols $\underline{p}_i$, $i = r_1, \ldots, n_1 - 1$ are considered as the information symbols of the code $\mathcal{C}_1$. By the encoding procedure for $\mathcal{C}_1$ we obtain r_1 redundancy symbols $\underline{p}_i$, $i = 0, \ldots, r_1 - 1$, and the $r_1 b_2$ upper right corner redundancy bits of the array are computed in a way that (2) holds for $i = 0, 1, \ldots, r_1 - 1$ and $\ell = 0, 1, \ldots, b_2 - 1$. We now turn to the encoding procedure of the horizontal component code. During this process the $r_1 b_2$ redundancy bits of the first subset will have the values which were just computed (as said before, this is different from Construction BBZS, in which they were assumed to be zeros). The second subset of redundancy bits spans over $\left\lceil \frac{m}{b_1} \right\rceil$ consecutive columns. In each column, with a possible exception of the last one, there are b_1 redundancy bits in b_1 positions (rows) which cover all the b_1 distinct residues modulo b_1. We compute $n_2 - m$ information symbols of the horizontal component codeword as in Construction BBZS. The remaining m symbols are the redundancy symbols of the horizontal $(b_1 b_2)$-burst-locator code and they are computed from the $n_2 - m$ information symbols. The only redundancy bit of the third subset is the binary sum of the computed redundancy bits from the second subset.

Decoding:

The decoding is done similarly to the one of Construction BBZS. First, each row generates b_2 parity bits by using (2) (this includes also the first r_1 rows, but the $m + 1$ redundancy bits of the second and the third subsets are assumed to be zeros). These b_2 bits are considered as a symbol in GF(2^{b_2}) and hence a word

of length n_1 over GF(2^{b_2}) is generated. Now, we use the decoding procedure of the vertical b_1-burst-correcting code to correct a burst of length b_1. If such a burst occurred it "almost" determines the rows in which errors occurred and also the shape of the cluster up to horizontal cyclic permutation. We say "almost" since the vertical code does not find erroneous redundancy bits from the second and the third subsets. These bits are spaced in a way that at most one such bit is in error. We sum these $m+1$ bits and if the result is not zero then one of these bits is erroneous. If this is the case then from rows of the cluster-error, discovered by the vertical code, we will know the exact row of this bit. If the vertical code did not find any burst and a redundancy bit from either the second subset or the third subset is erroneous then we have exactly a single error in the array. Now, since the $(b_1 b_2)$-burst-locator code is also a single-error-correcting code (binary Hamming code), this single error can be corrected. If there are more errors in the cluster, then we continue by either correcting the redundancy bit of the last two subsets (and the corresponding bit in the horizontal codeword) or adding this erroneous redundancy bit to the shape of the burst. In either case, the horizontal $(b_1 b_2)$-burst-locator code will discover the first column in which the cluster occurred, and hence the pattern discovered by the vertical component code enables us to correct the errors.

Remark 1: The parity bits of the second set can be chosen in other ways as long as they form a set of redundancy symbols for the burst-locator code, e.g., they do not have to be in consecutive columns. Such choices can result in other array sizes.

Remark 2: A natural question is to ask why not use also a binary vertical component code? The answer is that we can. The main advantage will be that we will have more flexibility in the parameters of our two-dimensional array. The disadvantage is that the excess redundancy will be increased by $\lceil \log_2 b_2 \rceil$.

The consequence of this construction is the following theorem (the computational part of the proof will be given in the next subsection).

Theorem 3: The given construction produces a $(b_1 \times b_2)$-cluster-correcting code of size

$$n_1 \times n_2 = \frac{(2^{b_2})^{r_1 - b_1 + 1} - 1}{2^{b_2} - 1} \times \left\lfloor \frac{2^{r - b_1 b_2 + 1} - 1}{b_1} \right\rfloor$$

with redundancy

$$r_1 b_2 + r - b_1 b_2 + 2 \leq \lceil \log_2 (n_1 n_2) \rceil + b_1 b_2 + \lceil \log_2 b_1 \rceil.$$

The construction can be applied whenever $b_1 b_2$ is odd integer, and there exists

- an optimum b_1-burst-correcting code, over GF(2^{b_2}), with redundancy r_1 and length $\frac{(2^{b_2})^{r_1 - b_1 + 1} - 1}{2^{b_2} - 1}$;
- a binary $(b_1 b_2)$-burst-locator code with redundancy r and length $2^{r - b_1 b_2 + 1} - 1$.

C. Multidimensional Arrays

As said earlier, one of the advantages of our construction is that it can be generalized in relatively simple way to D dimensions, while the excess redundancy remains relatively small. Assume we want to construct a D-dimensional code of size $n_1 \times n_2 \times \cdots \times n_D$ which corrects a box-error of size $b_1 \times b_2 \times \cdots \times b_D$. Let $B = \prod_{i=1}^{D} b_i$, where B is an odd integer. For the first dimension, we use a component code of length n_1 over GF$(2^{\frac{B}{b_1}})$ which corrects a burst error of size b_1. In each of the other $D-1$ dimensions, we use a burst-locator code which locates the position of the burst and its cyclic permutation in the corresponding direction. In dimension i, $2 \leq i \leq D$, we use a b_i-burst-locator code of length n_i over GF$(2^{\frac{B}{b_i}})$. The code of the first dimension finds the position of the error in the first dimension and the shape of the error, with a possible cyclic shift in each of the other $D-1$ dimensions. The code in dimension i, $2 \leq i \leq D$, finds the location of the position where the cluster starts in dimension i. After each code discovers the position where the cluster starts in its dimension (note, that this can be done in parallel), we have the corresponding cyclic shift in each dimension of the box-error found by the first code. Hence, we can now form the actual burst which occurred and correct it. As before, we can use in dimension i, $2 \leq i \leq D$, a binary B-burst-locator code. For the first dimension, we can choose consecutive redundancy bits as the redundancy bits of the first subset. For each other dimension, we will have to choose positions for the redundancy bits, which will fulfill the requirements for the redundancy bits of the second subset (only one redundancy bit will be needed as a parity bit for all the redundancy bits of this form for the burst-locator codes of all dimensions).

Theorem 4: Assume $b_1, b_2, \ldots, b_D$ are odd integers, $B = \prod_{i=1}^{D} b_i$, and the following codes exist.

- An optimum b_1-burst-correcting code, over GF$(2^{\frac{B}{b_1}})$, with redundancy r_1 and length $\frac{(2^{\frac{B}{b_1}})^{r_1 - b_1 + 1} - 1}{2^{\frac{B}{b_1}} - 1}$.
- For each i, $2 \leq i \leq D$, a binary B-burst-locator code with redundancy r_i and length $2^{m_i} - 1 = 2^{r_i - B + 1} - 1$.

Then, there exists a $(b_1 \times b_2 \times \cdots \times b_D)$-burst correcting code of size

$$n_1 \times n_2 \times \cdots \times n_D = \frac{(2^{\frac{B}{b_1}})^{r_1 - b_1 + 1} - 1}{2^{\frac{B}{b_1}} - 1} \times \left\lfloor \frac{2^{m_2} - 1}{b_2} \right\rfloor \times \cdots \times \left\lfloor \frac{2^{m_D} - 1}{b_D} \right\rfloor$$

and redundancy

$$r_1 \frac{B}{b_1} + \sum_{j=2}^{D} m_j + 1 \leq \lceil \log_2 (n_1 \cdots n_D) \rceil + B + \left\lceil \log_2 (b_1 B^{D-2}) \right\rceil + 1.$$

Proof: The existence of the code is implied by the proceeding description and we only have to compute its redundancy. For each j, $2 \leq j \leq D$

$$n_j = \left\lfloor \frac{2^{m_j} - 1}{\frac{B}{b_j}} \right\rfloor \geq \frac{2^{m_j} - \frac{B}{b_j}}{\frac{B}{b_j}}.$$

Therefore

$$n_1 n_2 \cdots n_D \geq \frac{(2^{\frac{B}{b_1}})^{r_1 - b_1 + 1}}{2^{\frac{B}{b_1}}} \prod_{j=2}^{D} \frac{2^{m_j} - \frac{B}{b_j}}{\frac{B}{b_j}}.$$

Now, taking into account that for each j, $2 \leq j \leq D$, $m_j > B$, and without loss of generality (w.l.o.g.) we can assume that for each i, $1 \leq i \leq D$, $b_i > 1$, we have that for each j, $2 \leq j \leq D$, $2^{m_j} b_j > 2(D-1)B$. It follows that

$$\log_2(n_1 n_2 \cdots n_D) \geq \frac{Br_1}{b_1} - B + \left(\sum_{j=2}^{D} m_j - 1\right) - \sum_{i=2}^{D} \log_2 \frac{B}{b_i}$$
$$= \frac{Br_1}{b_1} - B + \sum_{j=2}^{D} m_j - \log_2\left(B^{D-2} b_1\right) - 1.$$

Hence, the redundancy of the code is

$$r_1 \frac{B}{b_1} + \sum_{j=2}^{D} m_j + 1 \leq \lceil \log_2(n_1 \cdots n_D) \rceil + B + \left\lceil \log_2\left(b_1 B^{D-2}\right) \right\rceil + 1. \quad \square$$

When B is even we have to modify our method in order to obtain similar results. The modifications include binary component codes in all dimensions. Each one of the $D-1$ burst-locator codes locates the position of a cyclic burst of size $B+1$. This modification is described in Section IV.

IV. Colorings for Error Correction

The constructions presented in Section III are best applied when the volume of the box error is an odd integer. The reason is that Lemma 1 is true only when b is an odd integer. Hence, if the volume of the box-error is an even integer then the construction of Section III has to be used in a slightly different way. We have to apply the construction for correcting a box-error which has odd volume and contains the "real" box-error. The price will be an increase in the excess redundancy. In this section, we offer a novel method which will be useful to correct a box-error with even volume and also for correcting other types of cluster-errors. The excess redundancy will be similar to the one of the constructions in Section III.

A. The Coloring Method

The constructions with binary component codes use D components codes from which the first one is a burst-correcting code and the other $D-1$ codes are burst-locator codes. Position k in component code s is the binary summation of certain positions in the array, which were defined with correspondence to some modulo value related to s, k, and the D indices which define the position in the array. We generalize this idea to handle more complicated cluster-errors to a method which will be called the *coloring method*. A codeword is a D-dimensional array $\mathcal{A}$ (not necessarily a D-dimensional box). We want to correct any cluster-error with a given shape whose volume is B.

Again, we use D binary component codes to correct the cluster-error. The first code is a $(B+\delta_1)$-burst-correcting code, $\delta_1 \geq 0$. The sth component code, $2 \leq s \leq D$, is a $(B+\delta_s)$-burst-locator code, $\delta_s \geq \delta_1$. We further use D different colorings of the D-dimensional array. To each position of $\mathcal{A}$ we assign a color for each one of the D colorings. Each coloring will be associated with a different binary component code. For a given coloring and the corresponding component code, position k in the component code is the binary sum of all bits which are colored with color k. As we want to correct a cluster-error of a certain shape in the array we want that the colorings will satisfy a few properties:

- **(p.1)** For the sth coloring, for each s, $1 \leq s \leq D$, the colors inside a burst of the given shape are distinct integers and the difference between the largest integer and the smallest one is at most $B + \delta_s - 1$.
- **(p.2)** Given D colorings, a color ν_s for the sth coloring, $1 \leq s \leq D$. There is at most one position in the array which is colored with the colors $(\nu_1, \nu_2, \ldots, \nu_D)$.
- **(p.3)** Any two positions which are colored with the same color by the first coloring, have colors which differ by a multiple of $B+\delta_s$ by the sth coloring, for each s, $2 \leq s \leq D$.

Finally, we have to choose redundancy bits in the array, in a similar way to the method used in Section III.

Theorem 5: Assume that there exists a $(B+\delta_1)$-burst-correcting code of length n_1 and for each s, $2 \leq s \leq D$, there exists a $(B+\delta_s)$-burst-locator code of length n_s. Assume further that there exist D colorings which satisfy properties (p.1), (p.2), and (p.3), such that the sth coloring assigns colors between 1 and n_s to the D-dimensional array. Then the coloring method implies the existence of a B-cluster-correcting code for a D-dimensional array $\mathcal{A}$ and a cluster with a given shape and volume B.

Proof: The proof is straightforward from the description. We just note, that by property (p.2) there is no ambiguity in the erroneous positions. By property (p.1), for each s, $1 \leq s \leq D$, the erroneous positions affect at most $B+\delta_s$ consecutive positions in the codeword of the sth component code. Finally, since two positions in the array which are assigned the same color by the first coloring, have been assigned by the sth coloring colors which differ by a multiple of $B+\delta_s$ (see property (p.3)), it follows that the possible bursts in the sth $(B+\delta_s)$-burst-locator code are cyclic shifts of a burst with length $B+\delta_s$. $\square$

It will be more convenient when each coloring is a linear function of the coordinate indices, i.e., given a position $(i_1, i_2, \ldots, i_D)$, its color for the sth coloring will be defined by

$$\sum_{j=1}^{D} \alpha_j^s i_j$$

where α_j^s is a constant integer which depends on the coloring s and the shape of the D-dimensional cluster. Such a coloring will be called a *linear coloring*. With a linear coloring we associate a *coloring matrix* A_D, where $(A_D)_{s,j} = \alpha_j^s$. It is easy to verify that property (p.2) is fulfilled for a linear coloring if and only if the coloring matrix is an invertible matrix.

One can verify that the coloring method is a generalization of the method described in Section III. To observe this, we should assign a color to position $(i_1, i_2, \ldots, i_D)$ by the sth coloring, $1 \leq s \leq D$, in a slightly different way than the assignment in the next subsection. If position k in component code s is the binary sum of a certain set $\mathcal{S}$ of positions in the array, then all positions of $\mathcal{S}$ are colored with color k by the sth coloring. We leave the exact definitions of the colorings as an exercise for the reader.

B. Multidimensional Box-Errors

To demonstrate how the coloring method works we will first show how it is used to correct multidimensional box-errors, where the volume of the box error is an even integer.

Assume we want to construct an $n_1 \times n_2 \times \cdots \times n_D$ D-dimensional $(b_1 \times b_2 \times \cdots \times b_D)$ cluster-correcting code, where $B = \prod_{i=1}^{D} b_i$ is an even integer. We will use D binary component codes. One component code will be able to correct a burst of length B and $D-1$ component codes will be able to locate the position of a burst, whose length is $B+1$, given by a cyclic shift. Let

$$B_j = b_j B_{j-1}, \quad \text{where } 1 \le j \le D \text{ and } B_0 = 1.$$

For each entry $(i_1, i_2, \ldots, i_D)$ in the array we assign D colors. The sth color $1 \le s \le D$ is defined by

$$a^s_{i_1 i_2 \cdots i_D} = \sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}} i_j + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j.$$

Each coloring corresponds to one component code. The codeword of component code $s, \mathcal{C}_s, 1 \le s \le D$ is defined according to coloring s. Position k in the codeword is the sum modulo 2 of the values in positions colored with color k by the sth coloring. We will prove that these D codes satisfy properties (p.1) through (p.3).

Lemma 4: If the D-dimensional code has a box-error of size $b_1 \times b_2 \times \cdots \times b_D$ then each one of the D component codewords has a burst whose length is at most B.

Proof: A cluster that occurred in the array is located inside a multidimensional box of the form

$$\left\{(i_1^* + i_1, i_2^* + i_2, \ldots, i_D^* + i_D) : 0 \le i_j \le b_j - 1, 1 \le j \le D\right\}$$

for a fixed position $(i_1^*, i_2^*, \ldots, i_D^*)$. The smallest color, $\ell_{\mathcal{C}_s}$, of an erroneous position in $\mathcal{C}_s$, $1 \le s \le D$ is

$$\ell_{\mathcal{C}_s} = \sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}}\left(i_j^* + b_j - 1\right) + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j^*$$

which is the color of position $(i_1^* + b_1 - 1, \ldots, i_{s-1}^* + b_{s-1} - 1, i_s^*, \ldots, i_D^*)$ and the largest color $h_{\mathcal{C}_s}$ of an erroneous position in $\mathcal{C}_s$, $1 \le s \le D$ is

$$h_{\mathcal{C}_s} = \sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}} i_j^* + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}}\left(i_j^* + b_j - 1\right)$$

which is the color of position $(i_1^*, \ldots, i_{s-1}^*, i_s^* + b_s - 1, \ldots, i_D^* + b_D - 1)$. Now, we compute the difference $h_{\mathcal{C}_s} - \ell_{\mathcal{C}_s}$.

$$\begin{aligned} h_{\mathcal{C}_s} - \ell_{\mathcal{C}_s} &= \sum_{j=1}^{s-1} B_{j-1}\frac{B_D}{B_{s-1}}(b_j - 1) + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}}(b_j - 1) \\ &= \frac{B_D}{B_{s-1}}\left(\sum_{j=1}^{s-1} B_{j-1}(b_j - 1)\right) \\ &\quad + \frac{1}{B_{s-1}}\left(\sum_{j=s}^{D} B_{j-1}(b_j - 1)\right) \\ &= \frac{B_D}{B_{s-1}}\left(B_{s-1} - B_0\right) + \frac{1}{B_{s-1}}\left(B_D - B_{s-1}\right) \\ &= B_D - \frac{B_D}{B_{s-1}} + \frac{B_D}{B_{s-1}} - 1 = B - 1. \end{aligned}$$

Therefore, the length of a burst in each component code is at most B. □

Lemma 5: For each one of the D colorings, the B colors in each D-dimensional box of size $b_1 \times b_2 \times \cdots \times b_D$ in the array are all distinct.

Proof: Assume the contrary, that there exist two different positions $(i_1, \ldots, i_D)$, $(t_1, \ldots, t_D)$ located inside a box of size $b_1 \times b_2 \times \cdots \times b_D$ in the array whose sth color is identical. Therefore, by definition

$$\begin{aligned} &\sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}} i_j + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j \\ &\qquad = \sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}} t_j + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} t_j \quad (3) \end{aligned}$$

which implies

$$\begin{aligned} t_s - i_s &= \sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}}(i_j - t_j) + \sum_{j=s+1}^{D} \frac{B_{j-1}}{B_{s-1}}(i_j - t_j) \\ &= b_s\left(\sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_s}(i_j - t_j) + \sum_{j=s+1}^{D} \frac{B_{j-1}}{B_s}(i_j - t_j)\right). \end{aligned} \quad (4)$$

Since these two positions are located inside a box of size $b_1 \times \cdots \times b_D$ it follows that $0 \le |t_s - i_s| \le b_s - 1$ and hence $t_s - i_s = 0$. We continue with (4) and by induction we prove similarly that $t_k - i_k = 0$ for each k, $s+1 \le k \le D$. Therefore, by (3), we have

$$\sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}}(i_j - t_j) = 0$$

which implies

$$\sum_{j=1}^{s-1} -B_{j-1}(i_j - t_j) = 0\,.$$

Now, we will show by induction that for each j, $1 \le j \le s-1$, $t_j - i_j = 0$. For $j = 1$, we have

$$t_1 - i_1 = \sum_{j=2}^{s-1} B_{j-1}(i_j - t_j) = b_1\left(\sum_{j=2}^{s-1} \frac{B_{j-1}}{b_1}(i_j - t_j)\right)$$

and since $0 \le |t_1 - i_1| \le b_1 - 1$ we have $t_1 - i_1 = 0$. We continue in a similar way and obtain for each j, $1 \le j \le s-1$, $t_j - i_j = 0$. Therefore, for each $1 \le s \le D$, we have $t_s = i_s$. Thus, for each one of the D colorings, the B colors in each D-dimensional box of size $b_1 \times b_2 \times \cdots \times b_D$ in the array are all distinct. □

Lemma 6: Any two positions which are colored with the same color by the first coloring, have colors which differ by a multiple of $B+1$ by the sth coloring, for each s, $2 \le s \le D$.

Proof: Assume the kth position in the codeword of $\mathcal{C}_1$ is erroneous. This error results from an array error in position $(i_1, i_2, \ldots, i_D)$ such that $a^1_{i_1,i_2,\ldots,i_D} = \sum_{j=1}^{D} B_{j-1} i_j = k$, and, hence, $i_1 = k - \sum_{j=2}^{D} B_{j-1} i_j$. The possible error locations in $\mathcal{C}_s$, $2 \le s \le D$ are of the form

$$\begin{aligned} a^s_{i_1 i_2 \cdots i_D} &= \sum_{j=1}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}} i_j + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j \\ &= -\frac{B_D}{B_{s-1}} \left(k - \sum_{j=2}^{D} B_{j-1} i_j \right) + \sum_{j=2}^{s-1} -B_{j-1}\frac{B_D}{B_{s-1}} i_j \\ &\quad + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j \\ &= -\frac{B_D}{B_{s-1}} k + \sum_{j=2}^{D} B_{j-1}\frac{B_D}{B_{s-1}} i_j - \sum_{j=2}^{s-1} B_{j-1}\frac{B_D}{B_{s-1}} i_j \\ &\quad + \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j \\ &= -\frac{B_D}{B_{s-1}} k + \sum_{j=s}^{D} \left(B_{j-1}\frac{B_D}{B_{s-1}} + \frac{B_{j-1}}{B_{s-1}} \right) i_j \\ &= -\frac{B_D}{B_{s-1}} k + (B+1) \sum_{j=s}^{D} \frac{B_{j-1}}{B_{s-1}} i_j. \end{aligned}$$

$\frac{B_D}{B_{s-1}} k$ is a constant and therefore, two positions which have the same color by the first coloring, have colors which differ in a multiple of $B+1$ by the sth coloring, $2 \le s \le D$. □

Lemma 7: Given a position c_s in the sth component code, $1 \le s \le D$, the set of equations

$$c_s = a^s_{i_1 i_2 \cdots i_D}, \quad 1 \le s \le D \tag{5}$$

has exactly one solution for $(i_1, \ldots, i_D)$.

Proof: We will prove that the coloring matrix is invertible by proving that its determinant is not equal to *zero*. The (s, j) entry of the coloring matrix A_D is given by

$$(A_D)_{s,j} = \begin{cases} -B_{j-1}\frac{B_D}{B_{s-1}}, & \text{for } j < s \\ \frac{B_{j-1}}{B_{s-1}}, & \text{for } j \ge s. \end{cases}$$

We will prove by induction on D, $D \ge 2$, that $|A_D| = (1 + B)^{D-1}$, where B is the volume of the box-error.

For the basis of the induction, $A_2 = \begin{pmatrix} 1 & b_1 \\ -b_2 & 1 \end{pmatrix}$ and, hence, $|A_2| = 1 + b_1 b_2 = 1 + b_1 b_2$, where the two-dimensional cluster has size $b_1 \times b_2$.

For the induction hypothesis we assume that the determinant of every coloring matrix of size $(D-1) \times (D-1)$ is given by $|A_{D-1}| = (1 + B')^{D-2}$, where B' is the volume of the corresponding $(D-1)$-dimensional box-error.

For the induction step let A_D be a coloring matrix of size $D \times D$. The determinant of A_D is given by

$$|A_D| = \sum_{j=1}^{D} (-1)^{j+1} B_{j-1} |A_D[1, j]|$$

where $A_D[1, j]$ is the matrix obtained from A_D by deleting row 1 and column j. For $2 \le s \le D-1$, the sth row of $A_D[1, s]$ is given by

$$\frac{1}{B_{s-1}} (-B_D B_0, -B_D B_1, \ldots, -B_D B_{s-2}, B_s, \ldots, B_{D-1})$$

and the $(s+1)$th row is given by

$$\frac{1}{B_s} (-B_D B_0, -B_D B_1, \ldots, -B_D B_{s-2}, B_s, \ldots, B_{D-1}).$$

These two rows are linearly dependent and therefore, $|(A_D)_{1s}| = 0$ for $2 \le s \le D-1$ and we have

$$|A_D| = B_0 |(A_D)_{11}| + (-1)^{D+1} B_{D-1} |(A_D)_{1D}|. \tag{6}$$

The matrix $(A_D)_{11}$ is also a coloring matrix with respect to coloring related to the box-error $b_2 \times b_3 \times \cdots \times b_{D-1} \times (b_D b_1)$ and according to the induction assumption its determinant is given by $(A_D)_{11} = (1 + B)^{D-2}$. Let $(A'_D)_{1D}$ be the matrix constructed from $(A_D)_{1D}$ by dividing each element of the last row of $(A_D)_{1D}$ by $-b_D$ and shifting this row cyclically to be the first row. Clearly, $|(A_D)_{1D}| = -b_D (-1)^{D-2} |(A'_D)_{1D}|$. The matrix $(A'_D)_{1D}$ is also a coloring matrix with respect to coloring related to the error box $b_1 \times b_2 \times \cdots \times b_{D-2} \times (b_{D-1} b_D)$. Hence, $|(A_D)_{1D}| = -b_D (-1)^{D-2} (1+B)^{D-2}$ and from (6) we have $|A_D| = (1+B)^{D-2} + (-1)^{D+1} B_{D-1} (-b_D) (-1)^{D-2} (1+B)^{D-2} = (1+B)^{D-1}$. Thus, A_D is invertible and (5) has a unique solution. □

The encoding procedure is quite straightforward. First we have to choose three sets [or $D+1$ sets] of positions for the redundancy bits as in the previous constructions. Position k in the codeword of the sth component code is the binary sum of all positions in the array colored with k by the sth coloring. In the decoding procedure, the first component code provides a list of erroneous positions. If k is the erroneous location of the first code, the difference between equivalent errors in $\mathcal{C}_s$, $2 \le s \le D$ is a multiple of $B+1$ and by Lemma 6 the value $-\frac{B_D}{B_{s-1}} k$ is the residue modulo $B+1$ of the error location in the codeword of $\mathcal{C}_s$. Therefore, for the codeword of $\mathcal{C}_s$, the burst is known up to a cyclic permutation of length $B+1$. $\mathcal{C}_s$ is a burst-locator code which can locate where such burst started. Hence, each burst-locator code locates the position in which the burst has started with respect to the corresponding dimension. Therefore, the location of the erroneous position k in the first code is known in each of the $D-1$ burst-locator codes. Hence, we can partition the positions of the errors in the D component codewords into D-tuples. Each such tuple is of the form $(c_1, c_2, \ldots, c_D)$, where c_s is the error location in the codeword of $\mathcal{C}_s$, $1 \le s \le D$. By Lemma 7 such a D-tuple corresponds to exactly one position in the D-dimensional array, which is an erroneous position.

To summarize, by Lemmas 4 and 5, the D colorings satisfy (p.1); by Lemma 6, they satisfy (p.2); and by Lemma 7 they satisfy (p.3). Hence, by Theorem 5 the code is a $(b_1 \times b_2 \times \cdots \times b_D)$-cluster-correcting code. The redundancy of the code is slightly larger than the one for odd B (see Theorem 4). We omit the tedious proof.

V. Lee Spheres Cluster Errors

An error event at a position $(i_1, i_2, \ldots, i_D)$ can affect other positions around it. If we assume that the error event can be spread up to radius R, then any position $(t_1, t_2, \ldots, t_D)$ such that $\sum_{\ell=1}^{D} |t_\ell - i_\ell| \leq R$ might be erroneous. The set of positions $\{(t_1, t_2, \ldots, t_D) : \sum_{\ell=1}^{D} |t_\ell - i_\ell| \leq R\}$ forms a D-dimensional Lee sphere with radius R[14], [19]. Another important observation is that any arbitrary cluster-error of size b is located inside a Lee sphere with radius $\lfloor \frac{b}{2} \rfloor$. We wish to supply a method to correct all errors which might occur in such a given Lee sphere. The size of such sphere is $\sum_{j=0}^{\min\{D,R\}} 2^j \binom{D}{j}\binom{R}{j}$ [19] and each position can be erroneous. We can think about three different methods to perform the task.

- We can use any method which is able to correct all errors occurred in a
$$\underbrace{(2R+1) \times \cdots \times (2R+1)}_{D \text{ times}}$$
D-dimensional cube since a Lee sphere with radius R is located inside such a cube. The main disadvantage of this method is that the code can correct much more errors than the ones needed as its goal. For example, the size of the D-dimensional cube is $(2R+1)^D$, while the size of the D-dimensional Lee sphere with radius R is $\frac{(2R)^D}{D!} + O(R^{D-1})$ when D is fixed and $R \longrightarrow \infty$.
- We can transform the space into another space in a way that a Lee sphere will be transformed to a shape located inside another error-shape with a small volume which we know how to correct efficiently.
- We can find a coloring and perform correction of errors as explained in Section IV.

In this section we will explain how we apply the last two methods.

A. Space Transformation

The idea we are going to use is to transform the space into another space in such a way that a Lee sphere in one space will be transformed into a shape located inside a box in the second space. After that we will be able to use the encoding and decoding introduced in Section III. We start with the two-dimensional transformation.

Lemma 8: Let M, M^* be infinite two-dimensional arrays and let T be the transformation from M into M^* defined by $T(i_1, i_2) = \left(\lceil \frac{i_1+i_2}{2} \rceil, i_2 - i_1\right)$. Then a Lee sphere with radius R in the array M is located after the transformation T inside a rectangle of size $(R+1) \times (2R+1)$ in M^*.

Proof: A Lee sphere of radius R with center at (i_i, i_2) in the array M includes the set of positions

$$\begin{aligned} B_R(i_1, i_2) &= \{(t_1, t_2) : |t_1 - i_1| + |t_2 - i_2| \leq R\} \\ &= \{(i_1 + R_\ell, i_2 + R_j) : |R_\ell| + |R_j| \leq R\}. \end{aligned}$$

$B_R(i_1, i_2)$ is transformed by T into the set of positions

$$B_R^*(i_1, i_2) = \left\{ \left(\left\lceil \frac{i_1 + R_\ell + i_2 + R_j}{2} \right\rceil, i_2 + R_j - i_1 - R_\ell \right) : |R_\ell| + |R_j| \leq R \right\}$$

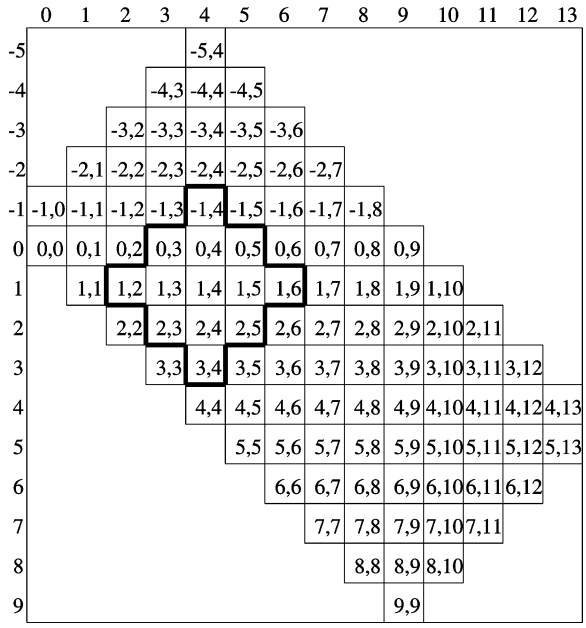

Fig. 3. Positions in the parallelogram.

0,0	-1,0	-1,1	-2,1	-2,2	-3,2	-3,3	-4,3	-4,4	-5,4
1,1	0,1	0,2	-1,2	-1,3	-2,3	-2,4	-3,4	-3,5	-4,5
2,2	1,2	1,3	0,3	0,4	-1,4	-1,5	-2,5	-2,6	-3,6
3,3	2,3	2,4	1,4	1,5	0,5	0,6	-1,6	-1,7	-2,7
4,4	3,4	3,5	2,5	2,6	1,6	1,7	0,7	0,8	-1,8
5,5	4,5	4,6	3,6	3,7	2,7	2,8	1,8	1,9	0,9
6,6	5,6	5,7	4,7	4,8	3,8	3,9	2,9	2,10	1,10
7,7	6,7	6,8	5,8	5,9	4,9	4,10	3,10	3,11	2,11
8,8	7,8	7,9	6,9	6,10	5,10	5,11	4,11	4,12	3,12
9,9	8,9	8,10	7,10	7,11	6,11	6,12	5,12	5,13	4,13

Fig. 4. Transformation to rectangle.

of M^*. Denote $i_1^* = \left\lceil \frac{i_1+i_2-R}{2} \right\rceil$, $i_2^* = i_2 - i_1 - R$, and we have that $B_R^*(i_1, i_2)$ is located inside the rectangle $\{(i_1^* + t_1, i_2^* + t_2) : 0 \leq t_1 \leq R, 0 \leq t_2 \leq 2R\}$.

The transformation T transforms a parallelogram into a rectangle (see Figs. 3 and 4) and hence we will need some adjustment in our encoding and decoding procedures if we want to correct Lee sphere clusters in a rectangular array rather than a parallelogram.

The construction is generalized to D dimensions. The transformation T will work between two D-dimensional arrays. For each entry $(i_1, i_2, \ldots, i_D)$ in the D-dimensional array M

$$T(i_1, i_2, \ldots, i_D) = \left(i_1^T, i_2^T, \ldots, i_D^T\right)$$

where $i_1^T = \left\lceil \frac{i_1+i_2}{2} \right\rceil$, $i_2^T = \left\lceil \frac{-i_1+i_2+i_3}{2} \right\rceil, \ldots, i_j^T = \left\lceil \frac{(-1)^{j+1} i_1 + (-1)^j i_2 + \cdots - i_{j-1} + i_j + i_{j+1}}{2} \right\rceil$, for $1 \leq j \leq D-1$ and $i_D^T = \sum_{j=1}^{D} (-1)^{D-j} i_j$. We invoke first the two-dimensional transformation T on the first two coordinates, then on the second and the third coordinates, and so on. Similarly to the proof of Lemma 8 we can prove the following lemma.

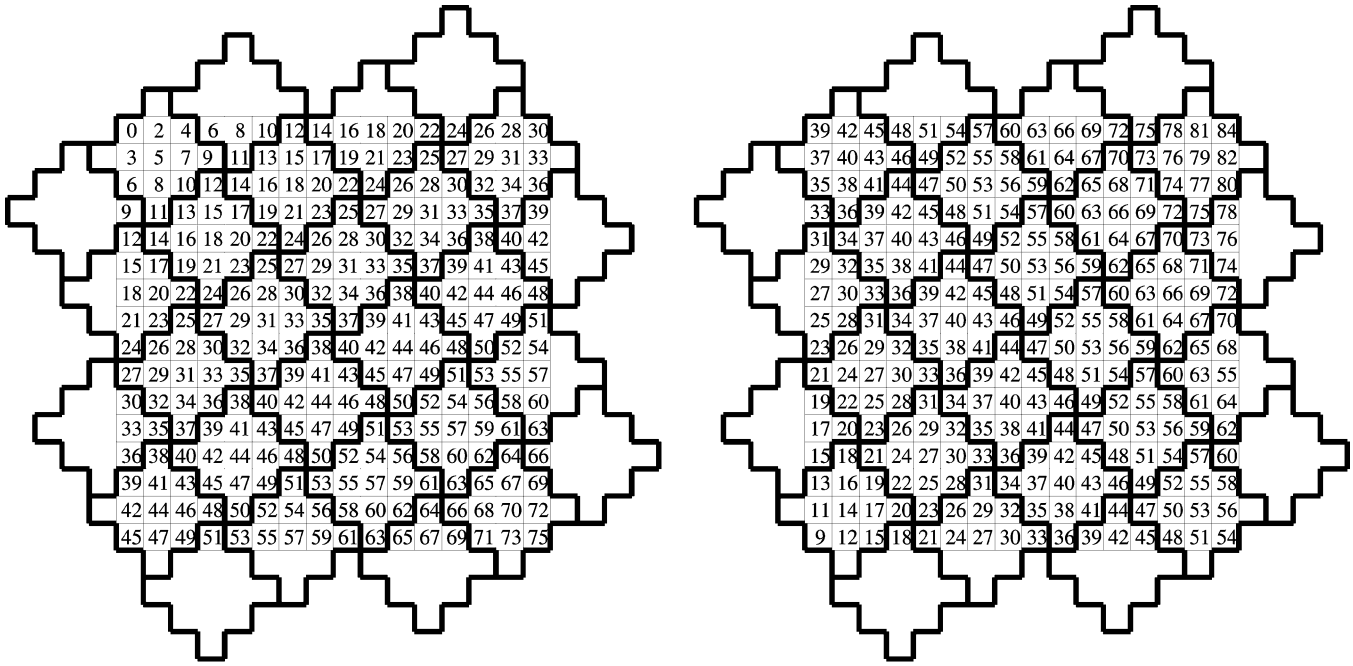

Fig. 5. (a) The coloring Ψ_1, and (b) the coloring Ψ_2, with $R = 2$. The left upper corner coordinate is $(-6, 9)$.

Lemma 9: Let M be an infinite D-dimensional array. Then a Lee sphere with radius R in the array M is located after the transformation T inside a D-dimensional box of size

$$\underbrace{(R+1)\times(R+1)\times\cdots\times(R+1)}_{D-1 \text{ times}}\times(2R+1)$$

in M^*.

As a consequence of this transformation we can use the constructions of Sections III and IV for correction of D-dimensional box-error. If we assume that our codewords are D-dimensional arrays rather than D-dimensional parallelograms we can use an array located inside the parallelogram. The redundancy in this case will be slightly increased.

B. Tiling and Coloring

When the error shape is a two-dimensional Lee sphere with radius R we can provide a code with a better redundancy than the code constructed by using the two-dimensional space transformation. We will use the coloring method of Section IV. Indeed, this construction is a good example for efficient uses of the coloring method. We choose two appropriate colorings Ψ_1 and Ψ_2. For a given position (i_1, i_2) in the $n_1 \times n_2$ two-dimensional array let $\Psi_1(i_1, i_2) = (R+1)i_1 + Ri_2$ and $\Psi_2(i_1, i_2) = -Ri_1 + (R+1)i_2$ (see Fig. 5).

Lemma 10: If $\Psi_1(i_1, i_2) = \Psi_1(t_1, t_2)$ then $\Psi_2(i_1, i_2) - \Psi_2(t_1, t_2)$ is a multiple of $2R^2 + 2R + 1$.

Proof: $\Psi_1(i_1, i_2) = \Psi_1(t_1, t_2)$ implies that $(R+1)i_1 + Ri_2 = (R+1)t_1 + Rt_2$. Hence, $R(i_2 - t_2) = (R+1)(t_1 - i_1)$, i.e., $(R+1)(i_2 - t_2) = \frac{(R+1)^2}{R}(t_1 - i_1)$.

Now

$$\begin{aligned}\Psi_2(i_1, i_2) &- \Psi_2(t_1, t_2)\\ &= -Ri_1 + (R+1)i_2 - (-Rt_1 + (R+1)t_2)\\ &= R(t_1 - i_1) + (R+1)(i_2 - t_2)\\ &= R(t_1 - i_1) + \frac{(R+1)^2}{R}(t_1 - i_1)\\ &= \frac{2R^2 + 2R + 1}{R}(t_1 - i_1).\end{aligned}$$

Since R and $2R^2 + 2R + 1$ are relatively primes it follows that R divides $t_1 - i_1$ and hence $\Psi_2(i_1, i_2) - \Psi_2(t_1, t_2)$ is a multiple of $2R^2 + 2R + 1$. □

The effect of these colorings of the two-dimensional arrays is best seen if we consider a tiling of the two-dimensional space with Lee spheres with radius R. Such tiling is well known and given in [14], [19]. In this tiling, each Lee sphere belongs to two diagonal strips. For the first coloring, all relative positions of the Lee spheres in the same diagonal strip have the same number; and in the other direction they are congruent modulo $b^* = 2R^2 + 2R + 1$ which is the size of a sphere (see Fig. 5).

Lemma 11: For any given Lee sphere with radius R in a two-dimensional array, the color numbers assigned by Ψ_i, $i = 1, 2$, to the positions of the Lee sphere are $2R^2 + 2R + 1$ consecutive integers.

Proof: Given a Lee sphere centered at the point (i_1, i_2) it is readily verified that smallest color number assigned by Ψ_1 to a point in the Lee sphere is $\Psi_1(i_1 - R, i_2)$ and the largest color number is $\Psi_1(i_1 + R, i_2)$. Now, $\Psi_1(i_1 + R, i_2) - \Psi_1(i_1 - R, i_2) = (R+1)(i_1 + R) + Ri_2 - ((R+1)(i_1 - R) + Ri_2) = 2R^2 + 2R$. Hence, to complete the proof we have to show that all the color numbers assigned by Ψ_1 to the positions inside a Lee sphere are distinct.

Let (i_1, i_2) and (t_1, t_2) be a pair of points for which $\Psi_1(i_1, i_2) = \Psi_1(t_1, t_2)$. By the proof of Lemma 10 we have that $R(i_2 - t_2) = (R+1)(t_1 - i_1)$. Hence, $i_2 - t_2 = \frac{R+1}{R}(t_1 - i_1)$ and $t_1 - i_1 = \frac{R}{R+1}(i_2 - t_2)$. Since R and $R+1$ are relatively primes it follows that R divides $t_1 - i_1$ and $R+1$ divides $i_2 - t_2$. This implies that $R \le |t_1 - i_1|$ and $R + 1 \le |i_2 - t_2|$.

Therefore, $2R+1 \leq |t_1-i_1|+|i_2-t_2|$ and the two points (i_1,i_2) and (t_1,t_2) cannot be contained in the same Lee sphere.

Thus, all color numbers assigned by Ψ_1, to the positions contained in a Lee sphere are $2R^2+2R+1$ consecutive integers. The same proof holds for Ψ_2. □

Lemma 12: The coloring matrix defined by Ψ_1 and Ψ_2 is an invertible matrix.

Proof: The coloring matrix defined by Ψ_1 and Ψ_2 is

$$\begin{pmatrix} R+1 & R \\ -R & R+1 \end{pmatrix}$$

which is clearly an invertible matrix. □

By Lemmas 10, 11, and 12, we have that (p.1), (p.2), and (p.3), are satisfied, respectively. Hence, by Theorem 5 the code constructed is capable to correct a Lee-sphere error with radius R and size b^*.

We choose $r+m+1$ appropriate redundancy bits similarly to the constructions in Section III. We use two component codes. One code is a b^*-burst-correcting code of length $2^{r-b^*+1}-1$, where 2^{r-b^*+1} is the least power of 2 greater than $(R+1)(n_1-1)+R(n_2-1)+1$ which is the number of colors needed to color the array with the first coloring. The second component code is a b^*-burst-locator code of length 2^m-1 (such a code exists since $b^*=2R^2+2R+1$ is an odd integer). We take 2^m to be the least power of 2 greater than $(R+1)(n_2-1)+R(n_1-1)+1$ which is the number of colors needed to color the array with the second coloring (for simplicity of computations we will take $2^{r-b^*+1} > (R+1)n_1+Rn_2 \geq 2^{r-b^*}$ and $2^m > (R+1)n_2+Rn_1 \geq 2^{m-1}$). The redundancy of the $n_1 \times n_2$ two-dimensional code is at most $\lceil \log_2(n_1n_2) \rceil + b^* + \lceil 2\log_2(2R+1) \rceil + 2$. If $n_1=n_2=n$, then the redundancy of the two-dimensional code is at most $\lceil \log_2 n^2 \rceil + b^* + \lceil 2\log_2(2R+1) \rceil$ and if each codeword is a rhombus of size n^2 then the redundancy is at most $\lceil \log_2 n^2 \rceil + b^* + \lceil \log_2 b^* \rceil$.

C. Multidimensional Codes

A similar idea can be used for D-dimensional code of size $n_1 \times n_2 \times \cdots \times n_D$ correcting Lee sphere error with radius one. We use D different colorings of the array. For a position $(i_1,i_2,\ldots,i_D)$, $0 \leq i_\ell \leq n-1$, $1 \leq \ell \leq D$, the sth coloring, $1 \leq s \leq D$, assigns the color $\Sigma_{\ell=1}^{D} \ell \cdot i_{s+\ell-1}$, where indices are residues modulo D between 1 and D (note that each color in each coloring forms a perfect code [19] when we consider the coloring in $\mathbb{Z}^D$ and not just in the array). Again, for each coloring, all the color numbers located in a Lee sphere of the D-dimensional array are $2D+1$ consecutive integers. Bit k of the sth component code is the binary sum of all the bits colored with the integer k, by the sth coloring, in the D-dimensional codeword. It is easy to prove that the coloring matrix is invertible. But, property (p.3) does not hold in all dimensions. Therefore, if property (p.3) does not hold for the sth coloring we need to take a $(2D+1)$-burst-correcting code instead of $(2D+1)$-burst-locator code for the sth dimension. The consequence will be a code with larger redundancy. Clearly, there are many appropriate colorings for each dimension. Hence, we can try to replace each coloring by one for which property (p.3) holds. Of course we have to make sure that the coloring matrix will be invertible. The excess redundancy in this case is quadratic in D, compared to exponential in D if we use the transformation T and the code which corrects D-dimensional box-error.

VI. Correction of Arbitrary Bursts

Finally, we want to design a code which corrects an *arbitrary D-dimensional cluster-error* of size b. Such a cluster $\mathcal{S}$ contains b positions and fulfills the following condition. If $\zeta=(i_1,i_2,\ldots,i_D) \in \mathcal{S}$ and $\xi=(j_1,j_2,\ldots,j_D) \in \mathcal{S}$, then there exists a path $\eta_1=\zeta,\eta_2,\ldots,\eta_\tau=\xi$ which satisfies the following two properties: for each $1 \leq i \leq \tau-1$, $\eta_i \in \mathcal{S}$; and the L_1 distance between η_i and η_{i+1} is one, i.e., they differ in exactly one coordinate and their difference in this coordinate is one. If b is odd then the cluster is located inside a Lee sphere with radius $\frac{b-1}{2}$. If b is even then we consider it as a cluster-error of size $b+1$. Modification based on coloring can be also obtained if $b=3$ and $D \geq 3$. The construction in this section will be based on codes which correct an one-dimensional burst of size b and limited weight t.

A. Bursts With Limited Weight

Assume we need to correct errors in a burst of size b, where the number of erroneous positions is at most t. These codes have an obvious application as we can expect that in area that suffers from an event which caused errors, not all positions were affected. The one-dimensional case was discussed in [18]. The parity-check matrix for such code is presented in Section VII.

Theorem 6: [18] There exists a code $\mathcal{C}$ of length $n=2^m-1$ which corrects every burst of length b with at most t erroneous positions. If b is an odd integer then the redundancy of the code is $\lceil \log_2 n \rceil + t\lceil \log_2 b \rceil$, and if b is an even integer then the redundancy of the code is $\lceil \log_2 n \rceil + t\lceil \log_2(b+1) \rceil$.

Now, we want to design a two-dimensional code of size $n_1 \times n_2$ capable of correcting a $(b_1 \times b_2)$-cluster with weight at most t, where b_1b_2 is an odd integer. We use a construction similar to the one used in previous sections. In this construction we have to use two binary component codes. The vertical one is a (b_1b_2)-burst-correcting code, of length n_1b_2 in which the weight of the burst is at most t. The horizontal component code is a (b_1b_2)-burst-locator code of length n_2b_1. They are used in the same manner as the two component codes are used in previous sections to correct a burst of size $b_1 \times b_2$. Since the vertical code can find a burst only if its weight is at most t, it follows that the two-dimensional code can handle bursts of size $b_1 \times b_2$ only if their weight is at most t. If b_1b_2 is an odd integer than by Theorem 6 we can take a vertical code of length n_1b_2, $2^m > n_1b_2 \geq 2^{m-1}$, with at most $m+t\lceil \log_2(b_1b_2) \rceil$ redundancy bits. The horizontal code has length $2^{r-b_1b_2+1}-1$, $2^{r-b_1b_2+1} > n_2b_1 \geq 2^{r-b_1b_2}$, with $r-b_1b_2+1$ redundancy bits. Therefore we have the following theorem.

Theorem 7: The redundancy of the $n_1 \times n_2$ code $\mathcal{C}$ which is capable to correct a $(b_1 \times b_2)$-cluster, b_1b_2 an odd integer, with weight at most t, is at most $\lceil \log_2(n_1n_2) \rceil + (t+1)\lceil \log_2(b_1b_2) \rceil + 3$.

The generalization for multidimensional codes is straightforward. If the size of the cluster is an even integer then we will use an appropriate coloring. Note, that always the first component code is a binary code which corrects an one-dimensional burst with a limited weight.

B. Arbitrary Bursts

We now show how one-dimensional codes which correct a burst with a limited weight help to correct an arbitrary cluster-error of size b. We start again with two-dimensional codes. As said before, a cluster-error of odd size b is located inside a Lee sphere with radius $R = \frac{b-1}{2}$. We will use the colorings Ψ_1 and Ψ_2 given in Section V and two component codes, the first one is a b^*-burst-correcting code, $b^* = 2R^2 + 2R + 1$, in which the weight of the burst is at most b, and the second one is b^*-burst-locator code. Now, we apply the coloring method of Section IV. The redundancy computations are similar to the ones in Section V-B and in Theorem 7 and we have the following result.

Theorem 8: The redundancy of an $n_1 \times n_2$ code capable to correct a cluster of odd size b is at most $\lceil \log_2(n_1 n_2) \rceil + (b+1)\lceil 2\log_2 b \rceil + 3$.

Generalization for D-dimensional code is done by using the transformation T of Section V. Hence, there is some loss of efficiency, but the performance is still better than the performance of a code which corrects a D-dimensional cluster error whose shape is a D-dimensional box-error with limited weight b. The redundancy computation is similar to the ones in previous sections.

The next question of interest is a lower bound on the excess redundancy of a code which corrects an arbitrary cluster of size b. A lower bound on the excess redundancy is $\lceil \log_2 N_D(b) \rceil$, where $N_D(b)$ is the number of distinct patterns considered as D-dimensional clusters of size b. Finding bounds on $N_D(b)$ is an interesting geometrical combinatorial problem in itself. A related question is to find the number of distinct clusters with size b, with no "holes," and exactly b erroneous positions. This problem is the same as finding the number of b-polyominos. For $D = 2$, the known lower bound on their number is 3.981037^b [20] and the known upper bound is 4.649551^b[21]. Therefore, we have the following result.

Theorem 9: The excess redundancy of a two-dimensional code, which is capable to correct an arbitrary cluster of size b, is at least $\lceil b \cdot \log_2 3.981037 \rceil$.

Theorem 9 is a small improvement of the trivial lower bound (which is b) on the excess redundancy. However, the gap between the orders of the lower bound $O(b)$ and the upper bound $O(b \cdot \log_2 b)$ is still large.

VII. Representation With Parity-Check Matrices

All codes which were discussed in the previous sections are using auxiliary linear codes for the computation of the redundancy bits in the codewords and to conduct the proper decoding. It is not difficult to see that all the multidimensional codes are linear by noting that the bit-by-bit addition of two codewords is another codeword. In this section, we will explain how to present similar codes with parity-check matrices. This will be done by considering all the component codes as binary codes.

Remark: When the component code is a linear burst-correcting code of length n over GF(2^b) we can consider it as a binary code of size $n \times b$, as we actually use it. We note that the bit-by-bit addition of two codewords of size $n \times b$ is also a codeword and hence the code is a binary linear code.

In these new codes, which will be constructed, we will not need the redundancy bit of the third subset and hence the overall redundancy will be reduced by one. The idea is to use a parity-check matrix $\mathcal{H}$ similar to the parity-check matrix H used in the construction for codes which correct a one-dimensional burst with a limited weight [18] constructed as follows. Let $\mathcal{C}_1$ be a t-error-correcting code of length b, and redundancy r. Let H_1 be its parity-check matrix of size $r \times b$

$$H_1 = \left[h_0^1, h_1^1, \ldots, h_{b-1}^1\right].$$

Let $\mathcal{C}_2$ be a b-burst-locator code, of length $n = 2^m - 1$ and redundancy m. Its parity-check matrix, H_2, is of size $m \times (2^m - 1)$

$$H_2 = \left[h_0^2, h_1^2, \ldots, h_{n-1}^2\right].$$

Based on these two matrices, we construct a new parity-check matrix H

$$H = \begin{bmatrix} h_0^1 & h_1^1 & \cdots & h_{b-1}^1 & h_0^1 & \cdots & h_{(n-1)(\mathrm{mod}\ b)}^1 \\ h_0^2 & h_1^2 & \cdots & h_{b-1}^2 & h_b^2 & \cdots & h_{n-1}^2 \end{bmatrix}.$$

The jth column, $0 \le j \le n-1$, of the matrix H will be defined as the concatenation of the $j(\mathrm{mod}\ b)$th column of the matrix H_1 and the jth column of the matrix H_2.

This technique can be generalized for two-dimensional and multidimensional codes. We will describe it only for two-dimensional codes. Each of our constructions for two-dimensional codes uses two component codes $\mathcal{C}_1$ and $\mathcal{C}_2$. Assume that $\mathcal{C}_1$ and $\mathcal{C}_2$ are binary codes with $r_1 \times n_1$ parity-check matrix H_1 and $r_2 \times n_2$ parity-check matrix H_2, respectively. We construct a two-dimensional parity-check matrix $\mathcal{H}$ for our two-dimensional code as follows. $\mathcal{H}$ is a two-dimensional matrix whose shape is the shape of the two-dimensional codeword. In each position Υ in this two-dimensional shape, $\mathcal{H}$ has a column vector of length $r_1 + r_2$ which is a concatenation of a column from H_1 and a column from H_2. The column from H_1 is ξ_1 if ξ_1 is the color given to position Υ by the first coloring (as mentioned before all our constructions can be represented by the coloring method). Similarly, the column from H_2 is ξ_2 if ξ_2 is the color given to position Υ by the second coloring (to avoid confusion, in the coloring method all positions are assigned with a color, by each coloring, including the redundancy bits). The proof that $\mathcal{H}$ is a parity-check matrix for the required code is not difficult and it is similar to the proof in [18]. A generalization for multidimensional codes is straightforward.

Finally, note that the same method can be also applied to the three constructions presented in [10]. Hence, we can supply a parity-check matrix for each code constructed by these three constructions.

VIII. Conclusion and Open Problems

As we stated in the abstract, the main results of the paper are summarized as follows.

1) A construction of small redundancy multidimensional codes capable to correct a box-error. These codes and the box-error have considerably more flexible parameters from previously known constructions.
2) A novel method based on D colorings of the D-dimensional space for constructing D-dimensional codes correcting a D-dimensional cluster-error of various shapes.
3) A transformation of the D-dimensional space into another D-dimensional space in a way that a D-dimensional Lee sphere is transformed into a shape located in a D-dimensional box of a relatively small size. This transformation enables us to use the previous constructions to correct a D-dimensional error whose shape is a D-dimensional Lee sphere.
4) Applying the coloring method to correct more efficiently a two-dimensional error whose shape is a Lee sphere.
5) A method to obtain multidimensional codes which correct a burst-error of length b in which the number of erroneous positions is t.
6) A construction for a code which corrects a D-dimensional arbitrary cluster-error with relatively small redundancy. This code is generated by using a combination of two other constructions. The first one is for correction of a Lee sphere error. The second one is a construction for one-dimensional codes which correct a cluster-error with a small number of erroneous positions.

Two new concepts, locator-codes and bursts with limited weight, were also introduced for the first time in this paper. These tools were the keys in obtaining the new results.

All the codes we have constructed are binary. We did not discuss cluster-correcting codes over GF(q), but most of our results can be generalized straightforwardly for codes over GF(q). We have omitted some tedious proofs. The interested reader is referred to [18] to see some of these proofs.

Clearly, our constructions do not cover all possible parameters. Moreover, the redundancy of our codes is close to optimal, but not optimal, so there is lot of ground for possible improvements with possibly new construction methods. In fact, the main disadvantage of our methods is that, for large b, the lengths of the known b-burst-correcting codes and b-burst-locator codes are very large.

Another question we did not discuss in this paper is constructions for cluster-correcting codes which correct a small cluster. This question was considered in [16]. The construction of D-dimensional cluster-correcting code which corrects a D-dimensional Lee sphere error with radius one is important in this connection. As we mentioned in Section V-C, we do not know the maximum number of $(2D+1)$-burst-locator codes, among the D component codes, that we can use.

The next question is how to implement the coloring method for correction of multidimensional Lee sphere errors with dimension greater than two and radius greater than one?

Finally, we still do not know and even do not have any indication what should be the excess redundancy of an optimum code which corrects an arbitrary multidimensional cluster-error. Even the two-dimensional case is far from being resolved. The gap between the lower and upper bounds of Theorem 9 and Theorem 8, respectively, is quite large and we believe that both bounds can be improved.

Acknowledgment

The authors wish to thank Khaled Abdel-Ghaffar for providing [8]. They also thank Khaled Abdel-Ghaffar and Jack Wolf for helpful discussions. We also thank two anonymous reviewers whose comments improved the presentation.

References

[1] M. A. Neifeld and M. McDonald, "Error correction for increasing the usable capacity of photorefractive memories," *Opt. Lett.*, vol. 19, pp. 1483–1483, 1994.
[2] C. Gu, J. Hong, I. McMichael, R. Saxena, and F. Mok, "Cross-talk limited storage capacity of volume holographic memory," *J. Opt. Soc. Amer.*, vol. 9A, pp. 1978–1987, 1992.
[3] J. F. Heanie, M. C. Bashaw, and L. Hesselink, "Volume holographic storage and retrieval of digital data," *Sci. Mag.*, vol. 265, pp. 749–752, 1994.
[4] S. A. Lis and P. D. Henshaw, "Ultra-dense optical mass storage," 1991, Report prepared for USAF, AFSC, AD-A232 767.
[5] M. Bossert and V. Sidorenko, "Singleton-type bound for blot-correcting codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 1021–1023, May 1996.
[6] S. H. Reiger, "Codes for correction of 'clustered' errors," *IRE Trans. Inf. Theory*, vol. IT-6, no. 1, pp. 16–21, Mar. 1960.
[7] I. M. Boyarinov, "Two-dimensional array codes correcting rectangular burst errors," *Probl. Inf. Transm.*, vol. 42, pp. 26–43, Jun. 2006.
[8] K. A. S. Abdel-Ghaffar, "An Information- and Coding-Theoretic Study of Bursty Channels With Applications to Computer Memories," Ph.D. dissertation, Calif. Inst. Technol., Pasadena, CA, 1986.
[9] K. A. S. Abdel-Ghaffar, R. J. McEliece, and H. C. A. van Tilborg, "Two-dimensional burst identification codes and their use in burst correction," *IEEE Trans. Inf. Theory*, vol. 34, no. 3, pp. 494–504, May 1988.
[10] M. Breitbach, M. Bossert, V. Zyablov, and V. Sidorenko, "Array codes correcting a two-dimensional cluster of errors," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 2025–2031, Sep. 1998.
[11] K. A. S. Abdel-Ghaffar, R. J. McEliece, A. M. Odlyzko, and H. C. A. van Tilborg, "On the existence of optimum cyclic burst-correcting codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 6, pp. 768–775, Nov. 1986.
[12] K. A. S. Abdel-Ghaffar, "On the existence of optimum cyclic burst-correcting codes over GF(q)," *IEEE Trans. Inf. Theory*, vol. 34, no. 2, pp. 329–332, Mar. 1988.
[13] H. Imai, "Two-dimensional fire codes," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 6, pp. 796–806, Nov. 1973.
[14] M. Blaum, J. Bruck, and A. Vardy, "Interleaving schemes for multidimensional cluster errors," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 730–743, Mar. 1998.
[15] T. Etzion and A. Vardy, "Two-dimensional interleaving schemes with repetitions: Constructions and bounds," *IEEE Trans. Inf. Theory*, vol. 48, no. 2, pp. 428–457, Feb. 2002.
[16] M. Schwartz and T. Etzion, "Two-dimensional cluser-correcting codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 2121–2132, Jun. 2005.
[17] R. M. Roth and G. Seroussi, "Reduced-redundancy product codes for burst error correction," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1395–1406, Jul. 1998.
[18] E. Yaakobi, "Codes for Correcting Multidimensional Bursts," M.Sc. thesis, Computer Science Dept., Technion–Israel Inst. Technol., Haifa, Israel, 2007.
[19] S. W. Golomb and L. R. Welch, "Perfect codes in the Lee metric and the packing of polyominos," *SIAM J. Appl. Math.*, vol. 18, pp. 302–317, 1970.
[20] G. Barequet, M. Moffiee, A. Ribó, and G. Rote, "Counting polyominos on twisted cylinders," *Electron. J. Comb. Number Theory*, vol. 6, no. A22, 2006.

[21] D. A. Klarner and R. L. Rivest, "A procedure for improving the upper bound for the number of n-ominoes," *Canad. J. Math.*, vol. 25, pp. 585–602, 1973.

Tuvi Etzion (M'89–SM'99–F'04) was born in Tel-Aviv, Israel, in 1956. He received the B.A., M.Sc., and D.Sc. degrees from the Technion–Israel Institute of Technology, Haifa, Israel, in 1980, 1982, and 1984, respectively.

Since 1984, he has held a position in the Department of Computer Science at the Technion, where he now holds a Professor position. During the years 1986–1987, he was Visiting Research Professor with the Department of Electrical Engineering–Systems at the University of Southern California, Los Angeles. During the summers of 1990 and 1991, he was visiting Bellcore in Morristown, NJ. During the years 1994–1996, he was a Visiting Research Fellow in the Computer Science Department at Royal Holloway College, Egham, U.K. He also had several visits to the Coordinated Science Laboratory at University of Illinois in Urbana-Champaign during the years 1995–1998, two visits to HP Bristol during the summers of 1996, 2000, a few visits to the Department of Electrical Engineering, University of California, San Diego during the years 2000–2008, and several visits to the Mathematics Department at Royal Holloway College, Egham, U.K., during the years 2007–2008. His research interests include applications of discrete mathematics to problems in computer science and information theory, coding theory, and combinatorial designs.

Dr. Etzion is currently an Associate Editor for Coding Theory of the IEEE TRANSACTIONS ON INFORMATION THEORY.

Eitan Yaakobi (S'07) was born in Israel in 1980. He received the B.A. and M.Sc. degrees from the Technion–Israel Institute of Technology, Haifa, Israel, in 2005 and 2007 respectively, from the Computer Science Department.

He is currently working toward the Ph.D. degree in electrical engineering at the University of California, San Diego, where he is associated with the Center for Magnetic Recording Research. His research interests include algebraic error-correction coding, coding theory, and their applications for digital data storage.