Array Codes for Clustered Column Erasures

Yuval Cassuto
Hitachi Global Storage Technologies
3403 Yerba Buena Rd.
San Jose, CA 95135, U.S.A.
yuval.cassuto@hitachigst.com

Jehoshua Bruck
California Institute of Technology
1200 E California Blvd., Mail Code 136-93
Pasadena, CA 91125, U.S.A.
bruck@paradise.caltech.edu

***Abstract*— A new error model is proposed for codes over channels with memory. According to this error model, both the number of symbol errors and the number of error clusters are used to characterize permissible errors. Considering this model as a generalization of random erasures in array codes naturally captures the properties of high-order failure events in disk arrays. A new family of codes tailored to such a model is shown to provide significant complexity improvements compared to known array codes.**

I. Introduction

Most results in Coding Theory implicitly assume operation over a Memoryless channel. Constructs like *t error-correcting codes* and concepts like the *Hamming distance* only consider the number of errors within a code block and not their relative locations. Very useful exceptions to that, on the other extreme – channels with strong memory, are codes for *burst* errors [8, Ch.9], and *multiple burst* errors [2]. In this paper a new family of error models that capture channels with memory is considered. Under these error models, codes are required to correct errors in up to a given number (ρ) of symbols, as long as the errors occupy at most a given number ($< \rho$) of *clusters*. Therefore, correctable error patterns are characterized by the number of symbol errors *and* the number of error clusters. An example of this characterization of error patterns is given in Figure 1 for patterns with $\rho = 4$ errors.

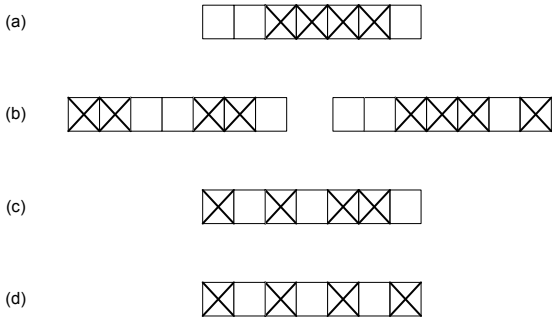

Figure 1. Classification of symbol combinations by their respective numbers of clusters. Four symbols (marked with X) that fall into (a) One cluster (b) Two clusters (c) Three clusters (d) Four clusters (non-clustered)

The present error characterization is different (and stronger) from the multiple-burst error model, as it does not predefine the cluster *sizes*, only their *number*. For example, an error model that requires correcting $\rho = 4$ errors in up to 2 clusters covers the single error burst in (a), and the *two possible patterns* of two error clusters in (b) of Figure 1.

Our current motivation to study this error characterization comes from multiple-failure-resilient disk arrays. In disk arrays, failure occurrences that affect many (ρ) devices, often originate from a smaller number ($< \rho$) of "independent events". Since adjacent devices in the array share cabling, cooling, power supply and mechanical stress, common failure events may affect multiple adjacent devices. Hence the number of clusters captures the number of "independent" failure events, each possibly affecting multiple devices in a single cluster.

The particular instance of the cluster-based error characterization in this paper is the case where the handled errors are *erasures* (errors in known locations), and more specifically for $\rho = 4$ symbol erasures. The motivation to address this case here is to allow for high-order failure-resilient disk arrays with improvement of various implementation complexity measures, compared to known array codes for random erasures. Some of the complexity properties of known array codes prevent their practical implementation in disk arrays, and the new code construction here provides a more efficient implementation alternative that fits well the nature of high-order failure events.

Besides the new error characterization model, the main contribution of this paper is the construction of an array-code family, called RC codes (for random/clustered) that correct essentially all clustered, and 7 out of 8 non-clustered, 4-erasure combinations. The RC codes are better than the (best) known family of array codes called Generalized EVENODD($r = 4$) [3] in all implementation complexity parameters. They improve the encoding and decoding complexities by 25%, the small-write update complexity by 28.57% and the full-column update complexity by 25%. They also support twice as many devices compared to EVENODD codes, for the same column size.

II. Definitions and Notations

A. Array Codes

A *length* n array code consists of n columns. In the codes discussed herein, there are k columns that store uncoded information bits and r columns that store redundant parity bits (thus $n = k + r$). A *column erasure* occurs when, for some physical reason, the contents of a particular column cannot be used by the decoder. We say that an array with given column erasures is *correctable* by the array code if there exists a decoding algorithm that, independent of the specific array contents, can reconstruct the original array from unerased columns only. An array code is called MDS (*Maximum Distance Separable*) if it has r redundant columns and it can correct all possible combinations of r column erasures. Beyond the redundancy of the code, two other complexity measures are considered in the context of array codes. The *small-write update complexity* (often simply called update complexity) is defined as the number of parity-bit updates required for a single information bit update, averaged over all information bits. The *full-column update complexity* is the number of parity columns that have to be modified per a single full-column

update. Another crucial performance criterion of an array code is its *erasure-decoding complexity*, defined as the number of bit operations (additions, shifts) required to recover the erased columns from the surviving ones.

Unless noted otherwise, p will refer to a prime number such that 2 is a primitive in the Galois field GF(p).

B. Random vs. Clustered erasure correction

To describe column erasure combinations whose only restriction is the number of erased columns, we refer to them as *random* [8] erasures.

Definition 1. *An array is said to recover from ρ* ***random erasures*** *if it can correct all combinations of ρ erased columns.*

All array-code constructions discussed in the literature aim at correcting random erasures. Refinement of the erasure model is possible by adding restrictions on the relative locations of the erased columns. This paper considers *clustered* erasures where the ρ erasures fall into a limited ($< \rho$) number of clusters. We now turn to some definitions related to the clustered erasure model.

In words, a *cluster* is a contiguous block of columns. More precisely,

Definition 2. *In an array code with columns numbered $\{0, 1, 2, \ldots, n-1\}$, a* ***cluster*** *is a set of σ columns such that the difference between the highest numbered column and the lowest numbered one is exactly $\sigma - 1$.*

For example, $\{2, 3, 4, 5\}$ is a cluster with $\sigma = 4$.

Definition 3. *A set of ρ columns is called* ***clustered*** *if the number of clusters that it occupies is* strictly *less than ρ.*

Random erasures have no restriction on their respective numbers of clusters and therefore they include both clustered and non-clustered erasures. The other extreme is the well-known *column burst* model where all erased columns need to fall into a single cluster. The thrust of the present paper is, for the first time, to address intermediate cases of clustered erasures between those two extremes as shown in (b) and (c) of Figure 1.

III. Preliminaries and Relevant Known Results

The purpose of this section is to discuss relevant known results in sufficient detail to prepare for the presentation of the new RC codes in the next section.

A. Codes for random erasures

The best known array codes that correct ρ random erasures, for a general ρ, are the Generalized EVENODD codes [3]. These codes are MDS codes and thus satisfy $r = \rho$. Discussing Generalized EVENODD codes in depth is beyond the scope of this paper. We only mention the following facts that are relevant to our presentation.

- The asymptotic small-write update-complexity of the general EVENODD code family is $2r - 1 - o(1)$. $o(1)$ refers to terms that tend to zero as the code length goes to infinity. Their full-column update-complexity is r.
- For $r > 3$, generalized r random erasure correcting EVENODD codes are only guaranteed to exist for p that belong to a subset of the primes: those that satisfy that 2 is a primitive element in the Galois field GF(p).
- The best known way to decode general EVENODD codes is using the algorithm of [4], for which the decoding complexity is dominated by the term rkp, the number of operations required to compute the syndrome of the surviving columns.

The properties of Generalized EVENODD codes will serve as the known state-of-the-art in multiple erasure correction, for comparison with the new RC codes proposed in this paper.

B. Algebraic View of Array Codes

To prove the correction capabilities of array codes, an algebraic view of the codes is developed to transform their simple encoding rules to parity-check matrices over some algebra. According to the algebraic framework first introduced in [4], columns of the code array of size $p-1$ are viewed as polynomials of degree $\leqslant p-2$ over $\mathbb{F}_2$ modulo the polynomial $M_p(x)$, where $M_p(x) = (x^p+1)/(x+1) = x^{p-1} + x^{p-2} + \cdots + x + 1$. When 2 is primitive in GF(p), the polynomial $M_p(x)$ is irreducible, and the encoding of parity columns can be represented by arithmetic operations over the Galois field[1] GF(2^{p-1}).

IV. Definition of the RC Codes

A. Geometric Description

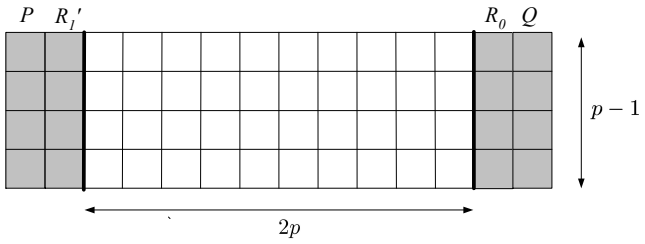

Figure 2. The RC-code array. RC codes have $2p$ information columns and 4 parity columns. The column size is $p-1$.

Referring to Figure 2, the RC code has $2p$ information columns (white) of $p-1$ bits each and 4 parity columns (shaded) with the same number of bits. The information columns are numbered in ascending order from left to right using the integers $\{0, 1, 2, \ldots, 2p-1\}$. Parity columns are not numbered and we use letter labels for them. The code is defined using its encoding rules shown in Figure 3, for the case $p = 5$. An imaginary row is added to the array to show the structure of the encoding function. Each icon represents a group of bits that are constrained by the code to have predetermined parities. Parity column P, located in the left most column of the left parity section, is simply the bit-wise even parity of the $2p$ information columns. Parity column R_1', located second from left, is the slope -1 diagonal parity of the *odd* numbered information columns $\{1, 3, \ldots, 2p-1\}$. Similarly to the EVENODD code family, the bit groups of R_1' are set to have even parity if the bits marked EO have even parity, and odd parity otherwise. Parity column R_0, located in the left most column of the right parity section, is the slope 2 diagonal parity of the *even* numbered information columns $\{0, 2, \ldots, 2p-2\}$. Parity column Q, located in the right most column of the right parity section, is the XOR of the slope 1 diagonal parities of both the even numbered columns and the odd numbered columns. The parity groups of Q and R_0, similarly to those of R_1', are set to be even/odd, based on the parity of the corresponding EO groups. Note that parity

[1] In [4] a more general case is considered where $M_p(x)$ is not necessarily irreducible and columns are represented as elements in a *ring*.

columns P and Q can be decomposed into $P = P0 \oplus P1$ and $Q = Q0 \oplus Q1$, where $P0, Q0$ depend only on even information columns and $P1, Q1$ only on odd ones.

For a formal definition of the code we write the encoding functions explicitly. Denote by $c_{i,j}$ the bit in location i in information column j. For an integer l, define $\langle l \rangle$ to be $l \pmod p$. Now we write the explicit expression of the parity bits.

$$P_i = \bigoplus_{j=0}^{2p-1} c_{i,j}$$

$$R'_{1_i} = S_1 \oplus \bigoplus_{j=0}^{p-1} c_{\langle i+j \rangle, 2j+1} \text{ , where } \quad S_1 = \bigoplus_{j=0}^{p-1} c_{\langle p-1+j \rangle, 2j+1}$$

$$R_{0_i} = S_0 \oplus \bigoplus_{j=0}^{p-1} c_{\langle i-2j \rangle, 2j} \text{ , where } \quad S_0 = \bigoplus_{j=0}^{p-1} c_{\langle p-1-2j \rangle, 2j}$$

$$Q_i = S_Q \oplus \left(\bigoplus_{j=0}^{p-1} c_{\langle i-j \rangle, 2j}\right) \oplus \left(\bigoplus_{j=0}^{p-1} c_{\langle i-j \rangle, 2j+1}\right) ,$$

$$\text{where} \qquad S_Q = \left(\bigoplus_{j=0}^{p-1} c_{\langle p-1-j \rangle, 2j}\right) \oplus \left(\bigoplus_{j=0}^{p-1} c_{\langle p-1-j \rangle, 2j+1}\right)$$

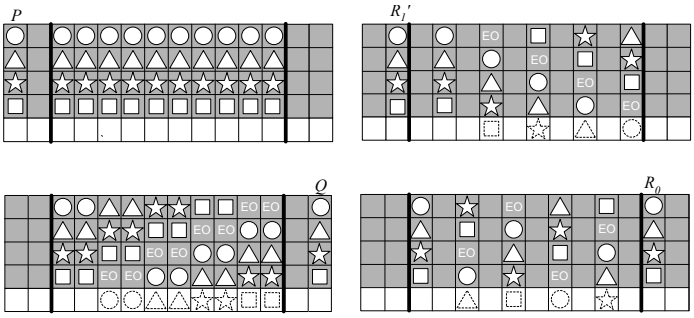

Figure 3. Encoding of the RC code. From top to bottom: the parity groups of parity columns P (slope 0), R'_1 (slope -1), Q (slope 1) and R_0 (slope 2). Parity columns R_0 and R'_1 each depends on only half of the columns, contributing to the low implementation complexity.

B. Algebraic Description

Over the field GF(2^{p-1}), the parity check matrix H of the RC code for $p = 5$ is given by

$$\left[\begin{array}{cc|cccccccccc|cc} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & \alpha^4 & 0 & \alpha^3 & 0 & \alpha^2 & 0 & \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 & \alpha^2 & 0 & \alpha^4 & 0 & \alpha & 0 & \alpha^3 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 & \alpha^3 & \alpha^3 & \alpha^4 & \alpha^4 & 0 & 1 \end{array}\right]$$

The correspondence between the encoding function in Figure 3 and the parity check matrix above is straightforward. The columns of the parity check matrix correspond to columns of the code array. The two left most columns are for parity columns P and R'_1 and the two right most columns are for R_0 and Q. Columns in between correspond to information columns in the array. In the parity check matrix, row 1 represents the constraints enforced by parity column P, rows 2, 3, 4 similarly represent the parity constraints of R'_1, R_0, Q, respectively. In any row j, the difference of exponents of α in two different columns is exactly the relative vertical shift of the two columns in the icon layout of the appropriate parity in Figure 3. For example, in the top row, all information columns have the same element, $1 (= \alpha^0)$, to account for the identical vertical alignment of the icons in the encoding rule of parity P. For general p the parity check matrix H has the following form.

$$H = \left[\begin{array}{cc|cccccccc|cc} 1 & 0 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & \cdots & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & \cdots & 0 & \alpha^{-i} & 0 & \alpha^{-(i+1)} & \cdots & \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & \alpha^{2i} & 0 & \alpha^{2(i+1)} & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & \cdots & \alpha^{i} & \alpha^{i} & \alpha^{i+1} & \alpha^{i+1} & \cdots & \alpha^{p-1} & 0 & 1 \end{array}\right] \quad (1)$$

After presenting the RC code family, we proceed in the next section to prove its erasure correction capabilities.

V. Erasure Correctability of RC Codes

In this section we prove that essentially all clustered combinations of 4 erasures are correctable by RC codes. Moreover, considering random erasure correctability, RC codes correct a $7/8$ portion of all combinations of 4 erasures. Discussion of the latter property is omitted in this presentation.

RC codes' excellent correction capability of clustered erasures is established in Theorems 4 and 5 below that follow a sequence of lemmas. The lemmas prove the key property that RC codes correct 4-erasure patterns, as long as they are not all at even locations or all at odd locations. Since all clustered erasures satisfy this condition, the theorems follow. Recall that the $2p + 4$ columns of the RC codes are labeled $\{P, R'_1, 0, 1, \ldots, 2p-2, 2p-1, R_0, Q\}$.

Lemma 1. *For a combination of* 4 *erasures, if* 3 *columns are either even numbered information columns or parity columns in* $\{R_0, P, Q\}$*, and* 1 *column is an odd numbered information column or the parity column* R'_1*, then it is a correctable* 4*-erasure. The complement case,* 3 *odd (or* R'_1 *or* P *or* Q*) and* 1 *even (or* R_0*), is correctable as well.*

Proof: The RC code can correct the erasure patterns under consideration using a two-step procedure. The first step is to recover the erased odd information column. Since only one odd column is erased, parity column R'_1 can be used to easily recover all of its bits. Then, when all odd columns are available, $P1$ and $Q1$ are computed, and used to find $P0$ and $Q0$ from P and Q (if not erased) by

$$P0 = P1 \oplus P \quad , \quad Q0 = Q1 \oplus Q$$

After that step, between even information columns, R_0, $P0$ and $Q0$, only 3 columns are missing. Since even columns, R_0, $P0$ and $Q0$ constitute an EVENODD code with $r = 3$, the 3 missing columns can be recovered. The complement case of 3 odd and 1 even column erasures is settled by observing that odd columns, R'_1, $P1$ and $Q1$ constitute an $r = 3$ MDS code [7]. ■

The next lemma presents the key property that, together with Lemma 1, gives RC codes their favorable clustered (and also random) erasure correctability.

Lemma 2. *For* p *such that* $p > 5$*, for a combination of* 4 *erasures, if* 2 *columns are even numbered information columns and* 2 *columns are odd numbered information columns, then it is a correctable* 4*-erasure.*

Proof: For the case of 2 even and 2 odd information-column erasures we consider two erasure patterns. All possible erasure

combinations of that kind are either covered by these patterns directly or are equivalent to them in a way discussed below. The discussion of each erasure pattern will commence with its representing diagram. In these diagrams, a column marked 0 represents an even column and a column marked 1 represents an odd column. Between each pair of columns, an expression for the number of columns that separate them is given .

a) Erasures of the form

0	$\leftarrow 2j \rightarrow$	1	$\leftarrow 2(k-1) \rightarrow$	0	$\leftarrow 2l \rightarrow$	1	

The variables j,k,l satisfy the following conditions: $1 \leqslant k$, $1 \leqslant j+k+l \leqslant p-1$.

The location of the first even erased column, together with j,k,l determine the locations of the 4 erasures. Any even cyclic shift of the diagram above does not change the correctability of the erasure pattern since this shift gives the same sub-matrix of the parity-check matrix, up to a non-zero multiplicative constant. Hence, we can assume, without loss of generality, that the first even erased column is located in the leftmost information column. To prove the correctability of this erasure pattern we examine the determinant (over GF(2^{p-1})) of the square sub-matrix of H that corresponds to the erased columns. If this determinant is non-zero for all values of j,k,l, then correctability is proved for all erasure combinations of this form.

The sub-matrix that corresponds to the erasure pattern above is

$$\mathcal{M}_a^{(j,k,l)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \alpha^{-j} & 0 & \alpha^{-j-k-l} \\ 1 & 0 & \alpha^{2(j+k)} & 0 \\ 1 & \alpha^{j} & \alpha^{j+k} & \alpha^{j+k+l} \end{bmatrix}.$$

Evaluating the determinant of this matrix gives

$$\begin{aligned} \left|\mathcal{M}_a^{(j,k,l)}\right| &= \alpha^{-l}(\alpha^{j+k}+1)(\alpha^{k+l}+1)\cdot \\ &\quad \cdot(\alpha^{j+k+l}+\alpha^{j}+\alpha^{l}+1+\alpha^{-k}) \end{aligned}$$

The first three factors in the product are clearly non-zero for any legal j,k,l. The last factor is non-zero for all j,k,l when $p > 5$, since the number of monomials is odd (and thus they cannot cancel each other to equal the zero polynomial). Furthermore, for $p = 5$, checking possible assignments of j,k,l and verifying that the last factor does not equal $M_5(x)$, we conclude that this pattern is correctable for $p = 5$ as well.

b) Erasures of the form

0	$\leftarrow 2j-1 \rightarrow$	0	$\leftarrow 2k \rightarrow$	1	$\leftarrow 2l-1 \rightarrow$	1	

The variables j,k,l satisfy the following conditions: $1 \leqslant j$, $1 \leqslant l$, $1 \leqslant j+k+l \leqslant p-1$.

Here, like in the previous pattern, we assume, without loss of generality, that the first even erased column is located in the leftmost information column.

The sub-matrix that corresponds to the erasure pattern above is

$$\mathcal{M}_b^{(j,k,l)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & \alpha^{-j-k} & \alpha^{-j-k-l} \\ 1 & \alpha^{2j} & 0 & 0 \\ 1 & \alpha^{j} & \alpha^{j+k} & \alpha^{j+k+l} \end{bmatrix}.$$

Evaluating the determinant of this matrix gives

$$\left|\mathcal{M}_b^{(j,k,l)}\right| = (\alpha^{j}+1)(\alpha^{l}+1)(\alpha^{j}+\alpha^{j-l}+1+\alpha^{-l}+\alpha^{-k-l})$$

In a similar manner to case a), this determinant is non-zero for $p > 5$. ■

The next Lemma treats (without proof) additional erasure combinations that include parity columns and that are not covered by Lemma 1.

Lemma 3. *When $p > 3$, the following 4-erasure combinations are correctable:*

1) R_1', 1 *odd information column and* 2 *even information columns*
2) R_0, 1 *even information column and* 2 *odd information columns*
3) R_0, R_1', 1 *even information column and* 1 *odd information column, except pairs of information columns numbered* $2i, 2i+1$, *respectively, for* $1 \leqslant i \leqslant p$.

Finally, we are ready to present the main result of this section. RC codes are next shown to correct all 4-erasures in up to two clusters, and almost all 4-erasures in three clusters. Given the lemmas above, establishing these results is rather straightforward by elementary counting, and will be given without proof.

Theorem 4. *When $p > 3$, RC codes correct all 4-erasures that fall into at most two clusters.*

Theorem 5. *When $p > 5$, the ratio between the number of RC-correctable 4-erasures that fall into three clusters and the total number of 4-erasures with three clusters is greater than 0.9696. As p goes to infinity, this ratio tends to 1.*

VI. Efficient Decoding of Erasures

In the previous section, the decodability of erasures was proved by algebraic reasoning. In this section we take a more constructive path and study simple and efficient ways to decode erasures. The purpose of this analysis is to prove that decoding the RC code can be done using $3kp + o(p^2)$ bit operations, while the best known algorithm for a 4-erasure correcting MDS code is $4kp + o(p^2)$ [4]. Since k is taken to be in the order of p, saving about kp bit operations gives an additive quadratic (in p) saving in computations that is very significant in practice for large p.

For the sake of the analysis, we only consider erasure of 4 *information* columns since these are the most common and most challenging cases. We moreover only consider erasures of two even columns and two odd columns, since as the proof of Lemma 1 shows, the three even and one odd case (or three odd and one even), reduces to a correction of three even (or odd) erasures, preceded by a simple parity completion for the single odd (or even) column erasure. Throughout the section we will assume that one of the erased columns is the leftmost even information column, as all other cases are cyclically equivalent.

A. Description of 4-erasure decoding algorithm

Decoding 4 erased columns represented by the GF(2^{p-1}) symbols $\{e_1, o_1, e_2, o_2\}$ is performed by solving the equation

$$\begin{bmatrix} e_1 & o_1 & e_2 & o_2 \end{bmatrix}^T = E^{-1}\boldsymbol{s} \tag{2}$$

where $\boldsymbol{s}$ is the syndrome of the non-erased columns. e_1, e_2 have even locations and o_1, o_2 have odd locations and so E has the following form.

$$E = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \alpha^{-u} & 0 & \alpha^{-w} \\ 1 & 0 & \alpha^{2v} & 0 \\ 1 & \alpha^{u} & \alpha^{v} & \alpha^{w} \end{bmatrix}.$$

The inverse of E, which is used in (2) to decode erasures, is now given in a closed form

$$E^{-1} = \left(\alpha^u + \alpha^v + \alpha^w + \alpha^{u+v} + \alpha^{v+w}\right)^{-1}$$

$$\cdot \begin{bmatrix} 1+\alpha^v & 0 & 0 & 0 \\ 0 & \alpha^u+\alpha^w & 0 & 0 \\ 0 & 0 & 1+\alpha^v & 0 \\ 0 & 0 & 0 & \alpha^u+\alpha^w \end{bmatrix}^{-1}$$

$$\cdot \begin{bmatrix} \alpha^{2v}(\alpha^u+\alpha^w) & \alpha^{u+2v+w} & \alpha^u+\alpha^v+\alpha^w & \alpha^{2v} \\ \alpha^{u+v} & \alpha^{u+w}(\alpha^v+\alpha^w+\alpha^{v+w}) & \alpha^u & \alpha^u(1+\alpha^v) \\ \alpha^u+\alpha^w & \alpha^{u+w} & 1+\alpha^u+\alpha^w & 1 \\ \alpha^{v+w} & \alpha^{u+w}(\alpha^u+\alpha^v+\alpha^{u+v}) & \alpha^w & \alpha^w(1+\alpha^v) \end{bmatrix}$$

From (2) and the closed-form expression above, the erased symbol e_1 can be recovered by the following product

$$e_1 = \left[\left(\alpha^u + \alpha^v + \alpha^w + \alpha^{u+v} + \alpha^{v+w}\right) \cdot (1+\alpha^v)\right]^{-1} \cdot$$
$$\cdot \left[\ \alpha^{2v}(\alpha^u+\alpha^w),\quad \alpha^{u+2v+w},\quad \alpha^u+\alpha^v+\alpha^w,\quad \alpha^{2v}\ \right] \cdot \boldsymbol{s}$$

Once e_1 is known, e_2 can be recovered using a simple parity completion with the aid of parity column R_0. The bits of the odd columns are then recovered by a chain of XOR operations with the aid of parity columns P, Q, that can now be adjusted to $P1, Q1$ when all even columns are known.

Calculating e_1 then reduces to the following chain of calculations

1) Finding the inverse of $\left(\alpha^u + \alpha^v + \alpha^w + \alpha^{u+v} + \alpha^{v+w}\right)(1+\alpha^v)$ over $\mathrm{GF}(2^{p-1})$.
2) A constant number of multiplications of two $\mathrm{GF}(2^{p-1})$ elements.

B. Analysis of 4-erasure decoding algorithm

We now analyze the number of bit operations required for each decoding task. In particular, it is shown that each task requires a number of bit operations that is $o(p^2)$.

1) **Finding inverses of $\mathbf{GF}(2^{p-1})$ elements**: An efficient algorithm for polynomial greatest common divisors that can be used to find inverses in $\mathrm{GF}(2^{p-1})$ is given in [1, Ch.8]. The algorithm requires $O(p \log^4 p)$ bit operations ($O(\log p)$ polynomial multiplications, each taking $O(p \log^3 p)$ bit operations, as shown in item 2 below).
2) **Multiplication of two $\mathbf{GF}(2^{p-1})$ elements** can be done in $O(p \log^3 p)$ bit operations using Fourier domain polynomial multiplication. The details of this standard operation for our special case of polynomial coefficients over $\mathrm{GF}(2)$ are omitted.

VII. Code Evaluation and Comparison with Known Codes

We compare RC codes to Generalized EVENODD ($r = 4$) codes using various performance criteria. The erasure-correction properties of both are given in Table I.

Table I concludes that RC codes outperform EVENODD codes in all complexity measures. The improvement of 28.57% in update complexity is the most crucial as a high update complexity slows down the storage array even when no failures occur. RC codes' reduced correction capability of random erasures has been analytically shown [5, Ch.3] to introduce only an insignificant decrease in reliability when moderately frequent clustered erasures are exhibited.

	RC Codes	4-EVENODD
Code Length (up to)	$2p$	p
Redundancy	4	4
Encoding Complexity	$3kp$	$4kp$
Decoding Complexity	$3kp$	$4kp$
Update Complexity	5	7
Clustered Failures	$\sim$ All	All
Random Failures	7/8	All

TABLE I
Comparison of RC Codes and EVENODD Codes

VIII. Conclusion

The key idea in the construction of the family of RC codes is to find a "good" "cooperating interleaving" of two codes. By "cooperating interleaving" we mean that some of the code parity bits are computed from only one constituent code, but other parity bits are computed from both codes. By "good" we mean that all 4-erasure combinations, except those that fall exclusively on one constituent code, are correctable by the code. For the particular case addressed by RC codes, the challenge was to simultaneously correct both combinations of (3 even/odd,1 odd/even) column erasures *and* combinations of (2 even,2 odd) column erasures. Pyramid codes [6] also use "cooperating interleaving" in their construction, but are only able to correct the (3 even/odd,1 odd/even) case and thus have no clustered erasure correctability.

It is an interesting research direction to pursue the more general framework of clustered errors proposed here from a coding theoretic perspective. Finding general combinatorial tools to address clustered error models, and use them to find code families and upper bounds is a promising direction with both theoretical and practical importance.

References

[1] A. Aho, J. Hopcroft, and J. Ullman, *The design and analysis of computer algorithms*. Reading, MA USA: Addison-Wesley, 1974.
[2] L. Bahl and R. Chien, "Multiple-burst-error correction by threshold decoding," *Information and Control*, vol. 15, no. 5, pp. 397–406, 1969.
[3] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.
[4] M. Blaum and R. Roth, "New array codes for multiple phased burst correction," *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp. 66–77, 1993.
[5] Y. Cassuto, "Coding techniques for data-storage systems," Ph. D. Dissertation, California Institute of Technology, Pasadena, CA, 2007.
[6] C. Huang, M. Chen, and J. Li, "Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proceedings of the Sixth IEEE International Symposium on Network Computing and Applications*, Cambridge, MA USA, 2007.
[7] C. Huang and L. Xu, "Star: An efficient coding scheme for correcting triple storage node failures," in *Proceedings of the 4th USENIX Conference on File and Storage Technologies*, San-Francisco CA, 2005.
[8] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1983.