# A new class of array codes for memory storage

L. A. Lastras-Montaño[1], P. J. Meaney[2], E. Stephens[2], B. M. Trager[1], J. O'Connor[2], L. C. Alves[2]

[1]IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY, 10598

[2]IBM Systems and Technology Group
2455 South Road
Poughkeepsie, NY, 12601

***Abstract*— In this article we describe a class of error control codes called "diff-MDS" codes that are custom designed for highly resilient computer memory storage. The error scenarios of concern range from simple single bit errors, to memory chip failures and catastrophic memory module failures. Our approach to building codes for this setting relies on the concept of expurgating a parity code that is easy to decode for memory module failures so that a few additional small errors can be handled as well, thus preserving most of the decoding complexity advantages of the original code while extending its original intent. The manner in which we expurgate is carefully crafted so that the strength of the resulting code is comparable to that of a Reed-Solomon code when used for this particular setting. An instance of this class of algorithms has been incorporated in IBM's zEnterprise mainframe offering, setting a new industry standard for memory resiliency.**

## I. Introduction

The main memory of the majority of modern computer servers is structured so any given read or write request from a processor is serviced through a parallel access to multiple Dynamic Random Access Memory (DRAM) chips. Error control codes whose codewords span these chips are used routinely to correct single bit errors caused, for example, by alpha particles and cosmic rays [1], [2], [3]. Failures affecting multiple bits have also been observed, ranging from a small section of a chip to entire chip failures. For this type of occurrence, servers employ symbol (as opposed to bit) oriented codes capable of correcting bursts of bit errors [4], [5].

Our goal is to present coding techniques that further protect a server against another type of failure that arises from the manner in which chips are organized within a memory system. In systems architecture, a *memory channel* is an independent physical construct that a processor uses to communicate to a group of memory chips that are often physically placed in a memory module; typically a processor will have multiple memory channels and in some instances distribute an ECC word over more than one channel. Sometimes it is the case that the design of a memory channel has single points of failure. For example, the chips in a memory channel may share a *memory buffer* in common which may fail, and/or may share a common clock signal that can also result in correlated errors. The goal is then to design a memory system that can survive entire memory channel errors, in addition to the kinds of errors described earlier.

IBM has introduced a mainframe server (zEnterprise) that possesses this kind of memory resilience. From an error control code design perspective, the technical challenges that needed to be overcome relate to the requirement of obtaining a reliability and redundancy optimum design while keeping the complexity of the design to a point in which very aggressive latency, bandwidth and chip circuit area requirements set by a main memory application are met.

Our investigation on how to attain these goals has led to a new class of error control codes which we call diff-MDS codes for reasons that will be clear later. In this article, we will systematically develop the foundations for how these codes are designed and analyzed. The basic concept we employ is that of *code expurgation* - that is, the process of removing codewords from an error correcting code in order to improve its error correcting capabilities. The code that we will expurgate will be designed as the simplest possible code that can correct the largest failure mode we will face, in this particular case, memory channel failures. The expurgation has the goal of providing additional redundancy to handle individual chip errors, to locate memory channel failures and to verify the accuracy of any corrections. The expurgating method is a key technical contribution of this article.

The article is organized as follows. In Section II we will define mathematically the problem to be solved, while motivating the problem with characteristics of the memory system application. In Section III we will motivate and introduce the concept of diff-MDS codes. In Section IV we will give a result on the error correcting and detecting capabilities of what we call expurgated simple parity check codes, which contain diff-MDS codes as a subclass. In Section V we give a methodology for building diff-MDS codes. Finally in Section VI we give a brief comparison of our code construction to Reed-Solomon codes. An Appendix with ancillary material is included at the end. The memory system that incorporates elements discussed in this article is discussed in a separate publication [6].

## II. Preliminaries

The error control codes that we will be discussing are defined over an $N \times M$ array of symbols, each of them comprised of $b$ bits. We will regard each symbol to be an element of the Galois Field with $2^b$ elements; we denote this Galois Field as $GF(2^b)$. In Table I we describe how these symbols are labeled and organized. In our application, a column is a memory channel, and any element of a column is

| channel 0 | channel 1 | channel 2 | $\cdots$ | channel M-1 |
|---|---|---|---|---|
| $d_0^{(0)}$ | $d_0^{(1)}$ | $d_0^{(2)}$ | $\cdots$ | $d_0^{(M-1)}$ |
| $d_1^{(0)}$ | $d_1^{(1)}$ | $d_1^{(2)}$ | $\cdots$ | $d_1^{(M-1)}$ |
| $d_2^{(0)}$ | $d_2^{(1)}$ | $d_2^{(2)}$ | $\cdots$ | $d_2^{(M-1)}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $d_{N-1}^{(0)}$ | $d_{N-1}^{(1)}$ | $d_{N-1}^{(2)}$ | $\cdots$ | $d_{N-1}^{(M-1)}$ |

TABLE I

ARRAY OF SYMBOLS IN THE ERROR CONTROL CODE. THERE ARE $M$ COLUMNS AND $N$ ROWS IN THE ARRAY.

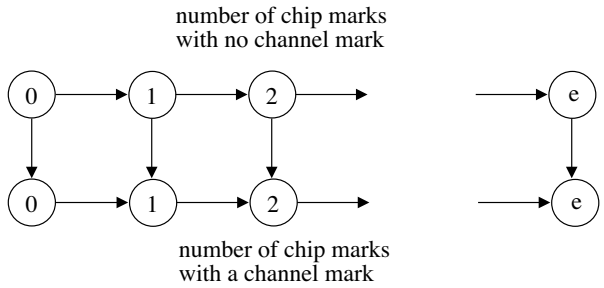


Fig. 1. Allowed transitions for array mark states. In addition to the state information of this diagram, the overall mark state includes a pointer to which chips and/or channel have been marked as bad.

a memory chip. The upper index in parenthesis in $d_i^{(j)}$ denotes the channel index. The subscript indicates the chip within the channel. From these $N \times M$ symbols, $k$ will be devoted to storing data and $r = N \times M - k$ will be used as redundant symbols.

Typically, a machine runs error free with occasional temporary errors affecting one symbol from Table I. Nonetheless, a hard error -whether single bit or multibit- may take place, and from that point onwards there is a significantly higher likelihood of an error being perceived at the associated symbol. Information about persistent faults is stored in the *marking state* of the memory (see Figure 1) - for a description of the techniques for gathering and maintaining this information see the associated publication [6]. Whenever something is *marked* (as potentially with errors) the decoder treats the associated symbol or symbols as erasures, in the standard sense of information and coding theory. Doing so is expected to allow the decoder to correct all marked symbols and potentially new additional errors as well. We assume that a maximum of $e$ chip marks may be stored for use in the decoding of a codeword of the error control code. In the most common use, these marks are applied sequentially as the system discovers hard chip errors, each time causing the marking state in Figure 1 to transition to an adjacent state on the right until no more chip marks are available.

In addition to single symbol errors, a memory channel may fail, resulting in a new multi-symbol error that is contained in a column in Table I. The decoder is expected to deduce entirely on its own the memory channel containing the multi-symbol error, even in the presence of chip marks in any place of the memory array, although in some instances there may be independent indications that a channel is failing [6]. Using variants of well known arguments (see the Reiger [7] and Singleton [8] bounds) it is possible to deduce that it is *mathematically impossible* to design any code that accomplishes the task of locating and correcting (autonomously) the failing channel with 100% certainty unless $2N + e \leq r$. Nonetheless, as we shall see, it is still feasible to find a failing channel with overwhelmingly high probability. Furthermore *once the bad channel is located* (by any available means, this is either by the decoder or by another agent), then a system that achieves 100% correction of this channel need only satisfy the relation $N + e \leq r$, which is a large improvement since typically $N$ is large. This leads to the notion of a *channel mark*, which is similar to the idea of a chip mark, although in this case applied to an entire channel. Whenever a channel mark is in place, the decoder no longer is required to find the location of a bad channel and may assume instead that the bad channel is the one pointed to by the channel mark. The judicious use of channel marks by a memory system and the use of alternate means for locating failing channels [6] results in the elimination, for all practical purposes, of the problem caused by the finite probability of not being able to locate failing channel.

The reader familiar with the theory of Reed-Solomon (RS) error correcting codes will notice that as long as $NM \leq 2^b - 1$, RS codes appear to be an ideal candidate for this problem, since they are Maximum-Distance-Separable (MDS) [9] and hence have strong optimality properties for settings with erasures and/or unknown symbol errors. It can be indeed shown that this is the case from the standpoint of the degree of reliability that these codes offer. Nonetheless the large burst of length $N$ symbols that needs to be corrected in potential combination with other errors can introduce implementation complexity and performance problems, particularly in the high performance setting of our problem (see Section VI).

The question that this article explores is whether there is a class of error control codes with favorable implementation complexity characteristics that offer comparable reliability as Reed-Solomon codes *when applied to the memory system reliability problem just described*. Note that if the only problem we had was to correct a memory channel in error, and we always knew which channel was in error, then we could have one of the $M$ channels be a simple parity of the other $M - 1$ channels as it is done in RAID systems, with decoding being a straightforward operation. For this simple abstract problem, clearly Reed-Solomon codes are unnecessary. One the other hand, a RAID scheme where one channel is a parity check of the other channels code does not provide means for locating autonomously a failing channel nor for correcting chip failures within the channel.

Our solution to this problem is to carefully augment the RAID scheme described above with a relatively small amount of additional redundancy in order to handle these additional

complications. Our emphasis on implementation complexity parallels earlier developments in the theory of array codes [10], [11] which have found widespread use in RAID systems. The mathematical problem setup follows next.

### A. Mathematical problem definition

For any positive integer $n$, let $GF(2^b)^n$ denote the space of vectors of length $n$ with elements in $GF(2^b)$. Let $\mathbf{d} \in GF(2^b)^{NM}$ denote a vector that is stored in any given memory location. The memory has $M$ channels and thus we write

$$\mathbf{d} = \left( \begin{array}{cccc} (\mathbf{d}^{(0)})^T & (\mathbf{d}^{(1)})^T & \cdots & (\mathbf{d}^{(M-1)})^T \end{array} \right)^T,$$

where $T$ denotes vector/matrix transpose, and where

$$\mathbf{d}^{(i)} = \left( \begin{array}{cccc} d_0^{(i)} & d_1^{(i)} & \cdots & d_{N-1}^{(i)} \end{array} \right)^T$$

for $i \in \{0, \cdots, M-1\}$. Thus for each $j \in \{0, \cdots, M-1\}$, $\mathbf{d}^{(j)} \in GF(q)^N$ and for $i \in \{0, \cdots, N-1\}$, $\mathbf{d}_i^{(j)} = \mathbf{d}_{jN+i}$. We shall assume that $\mathbf{d}$ is a codeword of the code defined by a parity check matrix $H$ that has dimensions $(N+\Delta) \times NM$ and entries in $GF(2^b)$. The parameter $\Delta > 0$ controls the *excess redundancy* one possesses to provide protection against individual symbol errors, for locating channel errors and for ensuring that any corrections that are applied are sound.

Any codeword of the code by definition must satisfy $H\mathbf{d} = 0$. The check matrices of interest can be written as

$$H = \left( \begin{array}{cccc} I_N & I_N & \cdots & I_N \\ \hat{H}^{(0)} & \hat{H}^{(1)} & \ldots & \hat{H}^{(M-1)} \end{array} \right) \tag{1}$$

where $I_N$ denotes the $N \times N$ identity matrix and

$$\hat{H} \triangleq \left( \begin{array}{cccc} \hat{H}^{(0)} & \hat{H}^{(1)} & \ldots & \hat{H}^{(M-1)} \end{array} \right) \tag{2}$$

where $\hat{H}$ is a $\Delta \times MN$ matrix. The essential constraint in (1) is that $\sum_{j=0}^{M-1} \mathbf{d}^{(j)} = 0$. We call a code with check matrix $\left( \begin{array}{cccc} I_N & I_N & \cdots & I_N \end{array} \right)$ a *simple parity check* code. Because the codes (1) are obtained by expurgating codewords from a simple parity check code, we call any code whose parity check matrix can be written as (1) an *expurgated simple parity check* code, and we call $\hat{H}$ the *expurgating matrix*.

The *marking state* of the memory describes

1) whether a channel is marked and the location of the possible channel mark if applicable.
2) how many chips are marked (up to $e$ chips), including the information of which chips are marked.

If there is no channel mark and $u$ chip marks, we will say we are in the marking state $(0, u)$. If there is a channel mark and $u$ chip marks, we will say in are in the marking state $(1, u)$. This notation is in reference to Figure 1, where the first index refers to a row and the second index refers to a column. Note that to be accurate, the marking state also includes the identity of the entities to which marks point to; this shorthand will be often useful though and should cause no confusion.

In order to compare the strength of various choices of $H$ (not necessarily expurgated simple parity check codes), we need to settle on a precise set of metrics for evaluating codes. We will now give a description of these metrics.

For all marking states of the form $(0, u)$, we assume that in every marked chip -if any- there can be any kind of error, or no error at all. In addition to an assumption that any errors on marked chips can be corrected, the code's strength is characterized by

- The maximum number $t_c^{(0,u)}$ of simultaneous chip errors it can correct.
- The maximum number $t_d^{(0,u)}$ of simultaneous chip errors it can detect or correct.
- The probability $p_c^{(0,u)}$ of correcting a channel error, under the *random channel error* model (see below).

A *random channel error* is a channel error which can be modeled as 1) choosing the channel in error from the set $\{0, 1, \cdots, M-1\}$ uniformly at random and 2) choosing for the error in the chosen channel a vector uniformly from $GF(2^b)^N$. Note that this includes the possibility that no error at all is added, nonetheless this is of no consequence to the analysis for sufficiently large (but still practical) $b$ and $N$. Note also that this error model allows for channel errors on channels where there may be chip marks.

For all marking states of the form $(1, u)$ we assume that in the marked channel and any marked chip there can be any kind of error or no error at all. All marked errors are assumed to be correctable. In addition, the code's strength is characterized by

- The maximum number $t_c^{(1,u)}$ of simultaneous chip errors it can correct.
- The maximum number $t_d^{(1,u)}$ of simultaneous chip errors it can correct or detect.

Given $(N, M, k)$ and $e$, the number of marked chips one can have, we are interested in a characterization of which values of $\{t_c^{(0,u)}, t_d^{(0,u)}, p_c^{(0,u)}, t_c^{(1,u)}, t_d^{(1,u)}\}_{u=0}^{e-1}$ can be attained practically. An answer to this is developed in Sections III, IV and V.

## III. DIFF-MDS CODES

For $j \in \{0, \cdots, M-1\}$, we define the matrix $\hat{H}^{(-j)}$ by deleting the $j$th component matrix from $\hat{H}$ and subtracting it from each of the remaining matrices. For example, the case when $j$ is neither 0 nor $M-1$ is as follows:

$$\begin{aligned} \hat{H}^{(-j)} = \Big( & \hat{H}^{(0)} - \hat{H}^{(j)} \quad \cdots \quad \hat{H}^{(j-1)} - \hat{H}^{(j)} \\ & \hat{H}^{(j+1)} - \hat{H}^{(j)} \quad \ldots \quad \hat{H}^{(M-1)} - \hat{H}^{(j)} \Big). \end{aligned}$$

The definition of $\hat{H}^{(-0)}$ and $\hat{H}^{(-(M-1))}$ follow directly from our explanation and example above. Note that in $GF(2^b)$, subtraction is identical to addition, so the use of the $-$ sign is superfluous. Nonetheless, all of the results in this article can be extended to finite fields with characteristic other than 2, and hence our use of the subtraction sign. For $j \in \{0, \cdots, M-1\}$ and for an arbitrary vector $v \in GF(2^b)^{NM}$, let $v^{\{-j\}}$ the vector obtained by deleting from $v$ the $j$th subcomponent. We illustrate this with an example when $j$ is neither 0 nor $M-1$:

$$\mathbf{v}^{\{-j\}} = (\mathbf{v}^{(0)}, \cdots, \mathbf{v}^{(j-1)}, \mathbf{v}^{(j+1)}, \cdots, \mathbf{v}^{(M-1)}).$$

Note that $\hat{H}^{(-j)}$ is a $\Delta \times N(M-1)$ matrix with elements in $GF(2^b)$. Let $A$ be any matrix with entries in $GF(2^b)$. Let $d(A)$ denote the minimum symbol distance of the code defined by parity check matrix $A$. This is, $d(A)$ is the largest integer such that any choice of $d(A)-1$ columns of $A$ are linearly independent. A code with parity check matrix is said to be Maximum-Distance-Separable (MDS) if $d(A)-1$ is equal to the number of rows of $A$.

We say that $\hat{H}$ is diff-MDS if $\hat{H}^{(-j)}$ is Maximum Distance Separable (MDS) for every $j \in \{0,1,\cdots,M-1\}$. We will motivate the definition of diff-MDS codes with an example of how one might decode a code whose parity check matrix has the form (1). In our example, channel 0 is affected with a "serious" error $\delta^{(0)} \in GF(2^b)^N$ which is any nonzero vector. Channels 1 through $M-1$ are affected with errors $\xi^{(1)}$ through $\xi^{(M-1)}$, which collectively, have exactly $t$ nonzero entries (and hence $t$ errors). Thus an encoded vector $\mathbf{d}$ is corrupted so that

$$\mathbf{v} = \begin{pmatrix} \mathbf{d}^{(0)} + \delta^{(0)} \\ \mathbf{d}^{(1)} + \xi^{(1)} \\ \vdots \\ \mathbf{d}^{(M-1)} + \xi^{(M-1)} \end{pmatrix}$$

is what is retrieved from the memory. Note that

$$\sum_{j=0}^{M-1} \mathbf{v}^{(i)} = \delta^{(0)} + \sum_{j=1}^{M-1} \xi^{(j)} + \sum_{j=0}^{M-1} \mathbf{d}^{(i)} = \delta^{(0)} + \sum_{j=1}^{M-1} \xi^{(j)}.$$

Suppose that we somehow knew that the "serious" channel error $\delta^{(0)}$ was in channel 0. Subtracting $\sum_{j=0}^{M-1} \mathbf{v}^{(i)}$ from channel 0 in $\mathbf{v}$ we obtain

$$\mathbf{w} = \begin{pmatrix} \mathbf{d}^{(0)} - \sum_{j=1}^{M-1} \xi^{(j)} \\ \mathbf{d}^{(1)} + \xi^{(1)} \\ \vdots \\ \mathbf{d}^{(M-1)} + \xi^{(M-1)} \end{pmatrix}.$$

Note that we have succeeded in removing the large error $\delta^{(0)}$, but the smaller errors have now "propagated" from channels 1 through $M-1$ to channel 0. Now let us compute the $\hat{H}$-syndrome of $\mathbf{w}$:

$$\begin{aligned} \hat{H}\mathbf{w} &= -\hat{H}^{(0)} \left( \sum_{j=1}^{M-1} \xi^{(j)} \right) + \sum_{j=1}^{M-1} \hat{H}^{(j)} \xi^{(j)} \\ &= \sum_{j=1}^{M-1} (\hat{H}^{(j)} - \hat{H}^{(0)}) \xi^{(j)} = \hat{H}^{(-0)} \mathbf{w}^{\{-0\}} \end{aligned}$$

where the last equality follows from the definitions made at the beginning of this section.

The key observation to make here is that a decoder can actually compute the $\hat{H}^{(-0)}$-syndrome of $w^{\{-0\}}$, which contains every other error *not* in channel 0. Thus if the linear code with parity check matrix $\hat{H}^{(-0)}$ can correct $t$ errors, we will be able to correct the error in channel 0 and the additional $t$ errors in the other channels.

From this discussion it becomes clear that if one must use the restriction (1), then one would like for the linear codes with parity check matrices $\hat{H}^{(-0)}, \cdots, \hat{H}^{(-(M-1))}$ to be good codes in the sense of minimum distance as well as in the sense of their ease of decoding.

The reader now should understand why the diff-MDS property is important in this particular setting. If $\hat{H}$ is diff-MDS, then the linear codes defined by $\hat{H}^{(-0)}, \cdots, \hat{H}^{(-(M-1))}$ are precisely MDS and hence optimum from the standpoint of minimum distance.

The present discussion will be elaborated upon in the following section, where we will characterize various properties of expurgated simple parity check codes as well as the particular class of diff-MDS codes.

## IV. The performance of expurgated simple parity check codes

Throughout this and the subsequent sections, we assume that $H$ satisfies (1); as usual $\hat{H}$ denotes the bottom $\Delta$ rows of $H$. Define the "diff-minimum distance" of $\hat{H}$ as

$$d_{\text{diff}}(\hat{H}) = \min_{0 \le i \le M-1} d(\hat{H}^{(-i)}).$$

This distance notion determines the majority of the key properties expurgated simple parity check codes. A first observation is that it is linked to the conventional minimum distance of the code defined by $H$ as follows:

*Lemma 1:* The matrix $H$ of an expurgated simple parity check matrix code together with its expurgating matrix $\hat{H}$ (see Equations (1, 2)) satisfy:

$$\left\lceil \frac{d_{\text{diff}}(\hat{H})}{1-1/M} \right\rceil \le d(H) \le \left\lceil \frac{\Delta+1}{1-1/M} \right\rceil$$

where $\Delta$ is the number of rows in $\hat{H}$.

For a proof of this result, see the Appendix. The bounds in this lemma are tight when $\hat{H}$ is a diff-MDS matrix since in this case we have $d_{\text{diff}}(\hat{H}) = \Delta + 1$. This lemma makes it clear that our codes are far from being MDS. Nonetheless because the error patterns we target are fairly specific, this is not of concern; our goals are different from, for example, those of [12] although low density parity check matrices and the MDS concept also play an important role in our work.

The following result summarizes the capability of a particular decoding architecture for expurgated simple parity check codes. The decoder architecture is given in the proof.

*Theorem 1:* Assume $u \ge 0$ chips are marked. Further assume that

$$u + t_c^{(0,u)} + t_d^{(0,u)} < d(H) \quad (3)$$

$$u + t_d^{(0,u)} < d_{\text{diff}}(\hat{H}) \quad (4)$$

$$u + t_c^{(1,u)} + t_d^{(1,u)} < d_{\text{diff}}(\hat{H}) \quad (5)$$

$$N + 1 \ge d_{\text{diff}}(\hat{H}). \quad (6)$$

Then an expurgated simple parity check code admits a decoder that can

1) in the absence of a channel mark, in addition to correcting the errors in the marked chips,

- detect (but not necessarily correct) up to $t_d^{(0,u)}$ chips in error,
- correct up to $t_c^{(0,u)}$ chips in error,
- detect any channel error, and correct it with probability at least

$$p_c^{(0,u)} \geq 1 - \frac{M-1}{2^{b(d_{\text{diff}}(\hat{H})-1-u)}} \tag{7}$$

under the random channel error model,

2) in the presence of a channel mark, in addition to correcting the errors in the marked channel and the marked chips,
   - detect (but not necessarily correct) up to $t_d^{(1,u)}$ chips in error,
   - correct up to $t_c^{(1,u)}$ chips in error.

*Remark:* The assumption (6) is a technical condition that we use to more easily obtain the estimate (7); it will be weakened in a subsequent publication.

### A. Proof preliminaries

The decoder that will be given in the proof will be based on processing the syndrome of the vector retrieved from the memory, as it is common in decoders for linear codes. Suppose that we have a vector $\mathbf{v} = \mathbf{x} + \mathbf{e} \in GF(2^b)^{NM}$ that we wish to decode, where $\mathbf{x}$ is the vector originally written to memory. Define the syndromes

$$\mathbf{s} = \hat{H}\mathbf{v}, \qquad \mathbf{z} = \sum_{j=0}^{M-1} \mathbf{v}^{(j)}. \tag{8}$$

The *channel modified* syndromes are defined as:

$$\mathbf{s}^{(-j)} \triangleq \hat{H}^{(-j)}\mathbf{v}^{\{-j\}} = \hat{H}^{(-j)}\mathbf{e}^{\{-j\}}. \tag{9}$$

Let $\{m_0, \cdots, m_{u-1}\} \subset \mathcal{M} \triangleq \{0, \cdots, NM-1\}$ denote the *chip marks* passed to the decoder.

If $\mathcal{A}_1$ and $\mathcal{A}_2$ are any two sets containing sequences from $GF(2^b)$ of identical length, then the set $\mathcal{A}_1 + \mathcal{A}_2$ is defined by the following:

$$\mathcal{A}_1 + \mathcal{A}_2 = \{\xi : \exists a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2 \text{ such that } \xi = a_1 + a_2\}.$$

Also note that in a field of characteristic 2, such as $GF(2^b)$, the the addition and subtraction operators are identical.

Let us now define basic classes of error patterns. First we have the patterns where the only errors are in channel $j$:

$$A_j = \{\xi \in GF(2^b)^{NM} : \xi^{\{-j\}} = 0\}.$$

We also define $A = \cup_{j=0}^{M-1} A_j$. Next we have the "up to $t$ random errors" case, for $t > 0$:

$$B_t = \{\xi \in GF(2^b)^{NM} : w(\xi) \leq t\}$$

where $w(\cdot)$ denotes the number of nonzero entries in the vector argument. Finally, define the set of errors patterns that only affect chips that are marked:

$$C = \{\xi \in GF(2^b)^{NM} : \text{if } \xi_j \neq 0 \text{ for some } j \in \mathcal{M} \text{ then } j \in \{m_0, \cdots, m_{u-1}\}\}.$$

Note that each of these sets contain the zero vector and in general can intersect in other ways. Finally note that by definition $t_d^{(0,u)} > t_c^{(0,u)}$, $t_d^{(1,u)} > t_c^{(1,u)}$.

### B. Proof in the case there is a channel marked

Suppose that channel $j^*$ is marked. In this case we assume that the error pattern satisfies $\mathbf{e} \in A_{j^*} + B_{t_d^{(1,u)}} + C$, since $t_d^{(1,u)} > t_c^{(1,u)}$. We assume, without loss of generality, that all of the $u$ chip marks are not located in channel $j^*$ since the following argument can be applied by removing any such marks, thereby reducing the value of $u$. Note that from (9), the syndrome $\mathbf{s}^{(-j^*)}$ depends only on $\mathbf{e}^{\{-j^*\}}$, which excludes any errors in channel $j^*$. We can regard the code with parity check matrix $\hat{H}^{(-j^*)}$, which has minimum distance at least $d_{\text{diff}}(\hat{H})$, as a $u$-erasure, $t_c^{(1,u)}$ random error correct and $t_d^{(1,u)}$ random error detect code as long as the condition (5) of the theorem is satisfied. Using any available decoder for this code results in recovering the error pattern $\mathbf{e}^{\{-j^*\}} \in GF(2^b)^{N(M-1)}$, unless an uncorrectable error has been detected. One can then recover the error in the channel marked through the relation $\mathbf{e}^{(j^*)} = \mathbf{z} - \sum_{i \neq j^*} \mathbf{e}^{(j)}$. At this point all errors have been recovered (if the errors were correctable) and the decoder can add the $\mathbf{e}$ vector to $\mathbf{v}$ to retrieve $\mathbf{x}$.

### C. Proof in the case where there is no channel marked

The decoder for this case operates in two main steps. In the first step, the decoder searches for a unique $\mathbf{e}' \in B_{t_c^{(0,u)}} + C$ such that $H\mathbf{e}' = [\ \mathbf{z}^T \quad \mathbf{s}^T\ ]^T$. If it succeeds, then the error is claimed correctable, $\mathbf{e}'$ is the error pattern to be added to the data that was read from the memory and the decoding finishes. If it fails, then the decoder next searches for a unique $\mathbf{e}' \in A + C$ such that $H\mathbf{e}' = [\ \mathbf{z}^T \quad \mathbf{s}^T\ ]^T$. As before if it succeeds then the decoder applies $\mathbf{e}'$ to the data, otherwise an uncorrectable error is declared. For the first step, we use a $u$-erasure, $t_c^{(1,u)}$ random error correct and $t_d^{(1,u)}$ random error detect decoder which by classic coding theory exists as long as condition (3) is satisfied. We will assume that this decoder will also ensure that the final correction $\mathbf{e}'$, if a correctable error has been found, has the same syndromes as the retrieved data, so that all information provided by the $\mathbf{s}, \mathbf{z}$ syndromes has been exhausted.

Now suppose that $\mathbf{e} \in (A + C) \setminus (B_{t_c^{(0,u)}} + C)$. First we claim that for any $\mathbf{a} \in B_{t_c^{(0,u)}} + C$, $H\mathbf{e} \neq H\mathbf{a}$. This is an important conclusion that ensures that error patterns in $(A + C) \setminus (B_{t_c^{(0,u)}} + C)$ are not mistakenly corrected in the first step of the decoder, which is not equipped to deal with them. If $\sum_j \mathbf{e}^{(j)} - \mathbf{a}^{(j)} \neq 0$ obviously $H\mathbf{e} \neq H\mathbf{a}$, so we can assume $\sum_j \mathbf{e}^{(j)} - \mathbf{a}^{(j)} = 0$. Note that $\mathbf{e} - \mathbf{a} \in A_j + B_{t_c^{(0,u)}} + C$ for some $j \in \{0, \cdots, M-1\}$. This implies $\hat{H}(\mathbf{e} - \mathbf{a}) = \hat{H}^{(-j)}(\mathbf{e}^{\{-j\}} - \mathbf{a}^{\{-j\}})$. Therefore, there are not more than $u + t_c^{(0,u)}$ nonzero elements in the vector $(\mathbf{e}^{\{-j\}} - \mathbf{a}^{\{-j\}})$ and thus by (4) and the fact that $t_c^{(0,u)} < t_d^{(0,u)}$, we have that $\hat{H}(\mathbf{e} - \mathbf{a}) \neq 0$, proving the claim.

Recall that we guarantee detection (but not necessarily correction) in the case $\mathbf{e} \in B_{t_d^{(0,u)}} + C$. As a matter of fact, if

$\mathbf{e} \in (B_{t_d^{(0,u)}} + C) \setminus (B_{t_c^{(0,u)}} + C)$ then by construction step 1 declares an uncorrectable error. What we would like to show is that under this condition for $\mathbf{e}$, the subsequent decoder in step 2 either correctly decodes the pattern or declares an uncorrectable event; stated differently we want to guarantee that in this case the decoder never miscorrects. Although we have not yet defined how the decoder in the second step operates, it will suffice to state that whatever correction it computes, it will have exactly the same syndromes as the retrieved data (as the decoder for the first step does) or otherwise declare an uncorrectable error. Under this assumption, we can show that miscorrection never happens by demonstrating that if $\mathbf{a} \in A+C$, and $\mathbf{a} \neq \mathbf{e}$, then $H\mathbf{a} \neq H\mathbf{e}$. The proof for this uses same arguments employed in the previous paragraph, with the exception that the assumption (4) is used directly.

The decoder next processes all channel modified syndromes (9). We note that the syndrome $\mathbf{s}^{(-j)}$ does not contain any contribution from errors in channel $j$, including any potential error in a chip marked if that chip happens to be in channel $j$. The decoder interprets, for each $j \in \{0, \cdots, M-1\}$ the code with parity check matrix $\hat{H}^{(-j)}$ to be a code that can correct up to $u$ erasures. The corresponding decoder is the one cited by the following Lemma:

*Lemma 2:* Let $A$ be a $r \times n$ parity check matrix of a linear code with entries in $GF(2^b)$. Let $u < d(A) - 1$. Then there exists a decoder $g : \{0, \cdots, n-1\}^u \times GF(2^b)^n \rightarrow \{\text{correctable}, \text{uncorrectable}\} \times GF(2^b)^n$ for the linear code with this parity check matrix $A$ with the property that it can correct any $u$ erasures. Now let $\{i_0, \cdots, i_{t-1}\} \subset \{0, \cdots, n-1\}$ be distinct but otherwise arbitrary indices, with $t \geq d(A) - 1$. Let $\mathbf{x} \in GF(2^b)^n$ be a codeword of this code. Let $\mathbf{e} \in GF(2^b)^n$ be such that $\mathbf{e}_i$ is chosen independently and uniformly at random from $GF(2^b)$ if $i \in \{i_0, \cdots, i_{t-1}\}$; no assumption is made about the remaining entries of $\mathbf{e}$. Whenever the decoding $\mathbf{x} + \mathbf{e}$, this decoder will mistakenly declare "correctable" with probability at most $1/2^{b(d(A)-u-1)}$ regardless of the erasure locations.

A proof of this Lemma can be found in the Appendix. For each $j \in \{0, \cdots, M-1\}$, the vector $\mathbf{x}^{\{-j\}} + \mathbf{e}^{\{-j\}}$ is passed to a decoder for the parity matrix $H^{(-j)}$, and the outputs from these $M$ decodings are collected. If one and only one of these decodings results in a "correctable" outcome, then the corresponding channel is claimed by the decoder to be the channel that failed. Decoding is then finalized in a manner similar to Subsection IV-B, since at this point we can consider the channel that failed to be marked (in this case by the decoder itself). If two or more decodings result in a correctable outcome, then the decoder claims an uncorrectable error.

If $\mathbf{e} \in (A_{j^*} + C) \setminus (B_{t_c^{(0,u)}} + C)$ for some $j^*$, then it is easy to see that during the decoding of $\mathbf{x}^{\{-j^*\}} + \mathbf{e}^{\{-j^*\}}$ assuming a code with check matrix $\hat{H}^{(-j^*)}$ we will obtain a correctable outcome as there are no more than $u$ erasures to be solved for. Thus for the pattern to be correctable by the main decoder, none of the other $M-1$ decoders must claim a correctable event. We now compute the probability of an uncorrectable error under the assumption that channel $j^*$ has been corrupted with a vector $\mathbf{e}^{(j^*)}$ that is drawn uniformly at random from $GF(2^b)^N$. In Lemma 2 substitute $t \leftarrow N$, $n \leftarrow N(M-1)$, $r \leftarrow \Delta$, $A \leftarrow \hat{H}^{(-j)}$ and use (6) to obtain the assertion that if $j \neq j^*$, then when decoding $\mathbf{x}^{\{-j\}} + \mathbf{e}^{\{-j\}}$ for the parity check matrix $\hat{H}^{(-j)}$ will result in a miscorrection with probability at most $1/2^{b(d_{\text{diff}}(\hat{H})-u-1)}$. Since there are $M-1$ different decodings one will perform (in addition to the one at channel $j^*$) using the probability union bound we find that the probability of two or more correctable decodings is at most $(M-1)/2^{b(d_{\text{diff}}(\hat{H})-u-1)}$ finalizing the proof of the Theorem. We stress to the reader that the main decoder *never* miscorrects when any channel fail happens in any state of the form $(0, u)$; instead it always detects, and with very high probability, corrects. The miscorrection events alluded to are purely a useful logical device that is internal to the decoder.

## V. A family of practical diff-MDS codes

Recall that the codes under consideration have a parity check matrix that can be written as in (1, 2):

$$H = \begin{pmatrix} I_N & I_N & \cdots & I_N \\ \hat{H}^{(0)} & \hat{H}^{(1)} & \ldots & \hat{H}^{(M-1)} \end{pmatrix}$$

where $I_N$ denotes the $N \times N$ identity matrix and

$$\hat{H} = \begin{pmatrix} \hat{H}^{(0)} & \hat{H}^{(1)} & \ldots & \hat{H}^{(M-1)} \end{pmatrix}$$

is a $\Delta \times MN$ matrix. The family of practical codes we propose chooses, for $k \in \{0, \cdots, M-1\}$, $i \in \{0, \cdots, \Delta-1\}$, $j \in \{0, \cdots, N-1\}$,

$$\hat{H}^{(k)}_{i,j} = X^{2^i}_{j,k} \tag{10}$$

where the $\{X_{j,k}\}$ are all distinct elements of $GF(2^b)$. The primary reason for choosing this construction is because in a field with characteristic 2 (such as our field $GF(2^b)$), for any $a, b$ elements of such field we have $a^2 + b^2 = (a+b)^2$. As a consequence, the matrices $\hat{H}^{(-k)}$, defined at the beginning of Section III, can be written in a manner similar to (10), creating algebraic structure that we can then exploit. Specifically, the $M-1$ components of $\hat{H}^{(-k)}$ have the form, for $l \neq k$,

$$(\hat{H}^{(l)} - \hat{H}^{(k)})_{i,j} = X^{2^i}_{j,l} - X^{2^i}_{j,k} = (X_{j,l} - X_{j,k})^{2^i}. \tag{11}$$

This enables a systematic analysis of the properties of the matrices $\hat{H}^{(-j)}$which will connect the theory of memory storage array codes with the form (10) with the theory of binary codes. Not any choice for $\{X_{i,j}\}$ will be suitable for our goals; to find good choices we will first establish the binary codes connection and then give one design technique.

### A. Connection to binary codes

In this and the following subsection, we will make use of the fact that an element of $GF(2^b)$ can be described using $b$ elements of $GF(2)$ using any given basis for $GF(2^b)$ over $GF(2)$. If $a \in GF(2^b)$, we will denote by $[a]$ the binary

column vector containing the $b$ coefficients of the expansion of $a$ using the given basis, indexed $[a]_0$ through $[a]_{b-1}$:

$$\begin{aligned} a \in GF(2^b) \quad &\longleftrightarrow \quad [a] \in GF(2)^b \\ [a] \quad &= \quad \left( \begin{array}{cccc} [a]_0 & [a]_1 & \cdots & [a]_{b-1} \end{array} \right)^T. \end{aligned}$$

The central tool for this section is the following result, which is a direct consequence of Lemma 1 of [14]; see also the preceding [15]:

*Lemma 3:* Let $A$ be a $r \times n$ matrix with elements in $GF(2^b)$ with the property that for $i \in \{2, \cdots, r\}$, $A_{i,j} = A_{i-1,j}^2$. Furthermore, let $B$ be the $b \times n$ *binary* matrix given by

$$B = \left( \begin{array}{cccc} [A_{0,0}] & [A_{0,1}] & \cdots & [A_{0,n-1}] \end{array} \right).$$

Then $d(A) = \min(r+1, d(B))$ where $d(A)$ is the minimum Hamming distance measured in symbols from $GF(2^b)$ and where $d(B)$ is the minimum Hamming distance measured in bits.

In order to connect this lemma with the goal of computing $d_{\text{diff}}(\hat{H})$ for a code of the form (10), we refer the reader to Figure 2, where we illustrate a portion of the process of computing $d(H^{(-0)})$. At the top, we have the $\Delta \times NM$ matrix $\hat{H}$. The bottom $\Delta - 1$ rows of this matrix are gray because the minimum distance of the $\{\hat{H}^{(-j)}\}_j$ codes depends only the first row, as per Lemma 3. In the subsequent step, we show $\hat{H}^{(-0)}$ restated to incorporate (11). In the last step, we take the first row of the previous step, which is comprised of $N(M-1)$ elements of $GF(2^b)$ and substitute each element with a column vector comprised of $b$ bits. This column vector contains the coefficients of the expansion of the corresponding $GF(2^b)$ element using the given basis for $GF(2^b)$ over $GF(2)$. The resulting $b \times NM$ binary matrix is denoted by $B^{(-0)}$. We take advantage of this example to similarly define, by an omitted extension, $B^{(-j)}$ for $j \in \{1, \cdots, M-1\}$.

Given Lemma 3, we have that $d(\hat{H}^{(-j)}) = \min(\Delta + 1, d(B^{(-j)}))$ and thus

$$\begin{aligned} d_{\text{diff}}(\hat{H}) \quad &= \quad \min_{j \in \{0, \cdots, M-1\}} \min(\Delta + 1, d(B^{(-j)})) \\ &= \quad \min\left(\Delta + 1, \min_{j \in \{0, \cdots, M-1\}} d(B^{(-j)})\right). \end{aligned}$$

Thus if in particular $d(B^{(-j)}) = \Delta + 1$ for every $j \in \{0, \cdots, M-1\}$, then $d_{\text{diff}}(\hat{H}) = \Delta + 1$ and $\hat{H}$ is a diff-MDS code.

### *B. Selection method for the* $\{X_{i,j}\}$

A family of codes leading to a parity check matrix $\hat{H}$ with the property that $d_{\text{diff}}(\hat{H}) = \Delta + 1$ (and hence diff-MDS) can be constructed by choosing for $i \in \{0, \cdots, N-1\}$, $j \in \{0, \cdots, M-1\}$,

$$X_{i,j} = \gamma_i \beta_j \tag{12}$$

where $b$ is assumed to be a multiple of $\Delta$, $\beta_j \in GF(2^{b/\Delta})$, $\gamma_i \in GF(2^b)$, and where the following holds:

1) If one chooses any subset from $\{\beta_0, \cdots, \beta_{M-1}\}$ with cardinality $\Delta'$, the elements of this subset are linearly independent over $GF(2)$, where $\Delta' = \Delta$ if $\Delta$ is even, otherwise $\Delta' = \Delta + 1$.
2) If one chooses any subset from $\{\gamma_0, \cdots, \gamma_{N-1}\}$ with cardinality $\Delta$, the elements of this subset are linearly independent over $GF(2^{b/\Delta})$.

In general, one may have to make $b$ sufficiently large to satisfy these requirements. In an alternate method, $\beta_j \in GF(2^b)$, $\gamma_i \in GF(2^{b/\Delta})$, and the following holds:

1) If one chooses any subset from $\{\gamma_0, \cdots, \gamma_{N-1}\}$ with cardinality $\Delta$, the elements of this subset are linearly independent over $GF(2)$.
2) If one chooses any subset from $\{\beta_0, \cdots, \beta_{M-1}\}$ with cardinality $\Delta+1$, the elements of this subset are linearly independent over $GF(2^{b/\Delta})$.

The proof of correctness of the latter is similar to the former option and hence it is omitted.

We now prove that $d_{\text{diff}}(\hat{H}) = \Delta + 1$ if the conditions for the case $\beta_j \in GF(2^{b/\Delta})$, $\gamma_i \in GF(2^b)$ are satisfied. Clearly, $d_{\text{diff}}(\hat{H}) \leq \Delta + 1$. We wish to show that for all $j \in \{0, \cdots, M-1\}$, $\min_{j \in \{0, \cdots, M-1\}} d(B^{(-j)}) \geq \Delta + 1$ where we emphasize that $d(B^{(-j)})$ denotes a *binary* minimum Hamming distance. We will demonstrate that the multiplication of the matrix $B^{(-j)}$ times any nonzero binary vector with length $N(M-1)$ and weight no larger than $\Delta$ results in a *nonzero* vector. This will imply that $\min_{j \in \{0, \cdots, M-1\}} d(B^{(-j)}) \geq \Delta + 1$.

Let $\mathbf{y}^{(i)} \in GF(2)^N$ for $i \in \{0, \cdots, M-1\}$. We will focus on the computation

$$B^{(-j)} \left( \begin{array}{cccc} \mathbf{y}^{(0)} \cdots \mathbf{y}^{(j-1)} & \mathbf{y}^{(j+1)} \cdots \mathbf{y}^{(M-1)} \end{array} \right)^T = \tag{13}$$

$$\sum_{i \neq j} \sum_{l=0}^{N-1} \left[ (X_{l,i} - X_{l,j}) \mathbf{y}_l^{(i)} \right] \tag{14}$$

where in the above $\mathbf{y}_l^{(i)}$ is regarded as an element of $GF(2^b)$ for the purposes of multiplication. The vector we are premultiplying with $B^{(-j)}$ has weight at least 1 and at most $\Delta$. This matrix/vector product can be rewritten as

$$\begin{aligned} \sum_{i \neq j} \sum_{l=0}^{N-1} \left[ X_{l,i} \mathbf{y}_l^{(i)} \right] + \sum_{l=0}^{N-1} \left[ X_{l,j} \left\{ - \sum_{i \neq j} \mathbf{y}_l^{(i)} \right\} \right] \\ = \left[ \sum_{i \in \{0, \cdots, M-1\}} \sum_{l=0}^{N-1} X_{l,i} \mathbf{y}_l^{(i)} \right] \end{aligned}$$

where similarly the quantity in curly brackets is regarded as an element of $GF(2^b)$ and where we have defined

$$\mathbf{y}_l^{(j)} \triangleq - \sum_{i \neq j} \mathbf{y}_l^{(i)}. \tag{15}$$

Now we write

$$\sum_{i=0}^{M-1} \sum_{l=0}^{N-1} X_{l,i} \mathbf{y}_l^{(i)} = \sum_{i=0}^{M-1} \sum_{l=0}^{N-1} \gamma_l \beta_i \mathbf{y}_l^{(i)} \tag{16}$$

$$= \quad \sum_{l=0}^{N-1} \gamma_l \left( \sum_{i=0}^{M-1} \beta_i \mathbf{y}_l^{(i)} \right) \triangleq \sum_{l=0}^{N-1} \gamma_l \eta_l \tag{17}$$

$$\hat{H} = \begin{pmatrix} X_{0,0}\cdots X_{N-1,0} & X_{0,1}\cdots X_{N-1,1} & \cdots & X_{0,M-1}\cdots X_{N-1,M-1} \\ X_{0,0}^2\cdots X_{N-1,0}^2 & X_{0,1}^2\cdots X_{N-1,1}^2 & \cdots & X_{0,M-1}^2\cdots X_{N-1,M-1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ X_{0,0}^{2^{\Delta-1}}\cdots X_{N-1,0}^{2^{\Delta-1}} & X_{0,1}^{2^{\Delta-1}}\cdots X_{N-1,1}^{2^{\Delta-1}} & \cdots & X_{0,M-1}^{2^{\Delta-1}}\cdots X_{N-1,M-1}^{2^{\Delta-1}} \end{pmatrix}$$

Obtain $H^{(-0)}$ (associated with channel 0 failure)

$$\hat{H}^{(-0)} = \begin{pmatrix} X_{0,1}-X_{0,0}\cdots X_{N-1,1}-X_{N-1,0} & \cdots & X_{0,M-1}-X_{0,0}\cdots X_{N-1,M-1}-X_{N-1,0} \\ (X_{0,1}-X_{0,1})^2\cdots(X_{N-1,1}-X_{N-1,0})^2 & \cdots & (X_{0,M-1}-X_{0,0})^2\cdots(X_{N-1,M-1}-X_{N-1,0})^2 \\ \vdots & \vdots & \vdots \\ (X_{0,1}-X_{0,1})^{2^{\Delta-1}}\cdots(X_{N-1,1}-X_{N-1,0})^{2^{\Delta-1}} & \cdots & (X_{0,M-1}-X_{0,0})^{2^{\Delta-1}}\cdots(X_{N-1,M-1}-X_{N-1,0})^{2^{\Delta-1}} \end{pmatrix}$$

Perform binary expansion of first row of $\hat{H}^{(-0)}$

$$B^{(-0)} = \begin{pmatrix} [X_{0,1}-X_{0,0}]_0\cdots[X_{N-1,1}-X_{N-1,0}]_0 & \cdots & [X_{0,M-1}-X_{0,0}]_0\cdots[X_{N-1,M-1}-X_{N-1,0}]_0 \\ [X_{0,1}-X_{0,0}]_1\cdots[X_{N-1,1}-X_{N-1,0}]_1 & \cdots & [X_{0,M-1}-X_{0,0}]_1\cdots[X_{N-1,M-1}-X_{N-1,0}]_1 \\ \vdots & \ddots & \vdots \\ [X_{0,1}-X_{0,0}]_{b-1}\cdots[X_{N-1,1}-X_{N-1,0}]_{b-1} & \cdots & [X_{0,M-1}-X_{0,0}]_{b-1}\cdots[X_{N-1,M-1}-X_{N-1,0}]_{b-1} \end{pmatrix}$$

Fig. 2. Example of how a binary code is derived from the original parity check matrix $\hat{H}$, which is assumed to be of the form (10). The (binary) minimum distance of the binary code shown at the bottom determines the $GF(2^b)$ minimum distance of $\hat{H}^{(-0)}$.

Let $l$ be fixed. If $\mathbf{y}_l^{(i)} = 1$ for any $i \in \{0,\cdots,M-1\}$, we claim that $\eta_l \neq 0$ and otherwise $\eta_l = 0$. The latter is obvious. To see the former note that there are at most $\Delta+1$ elements of $\{\mathbf{y}_l^{(0)},\cdots,\mathbf{y}_l^{(M-1)}\}$ that are nonzero. To be more precise, if $\Delta$ is even, then the maximum number of nonzero elements in this set is actually $\Delta$, because in (15), whenever there is an even number of nonzero summands in the right hand side, we have $\mathbf{y}_l^{(j)} = 0$. On the other hand, if $\Delta$ is odd, then the maximum number of nonzero elements is $\Delta+1$, for a similar reason.

Since $\eta_l$ is a linear combination of the $\{\beta_i\}$ using coefficients from $GF(2)$, due to the linear independence property that we assume of the $\{\beta_i\}$ we conclude that $\eta_l \neq 0$ if $\mathbf{y}_l^{(i)} = 1$ for any $i \in \{0,\cdots,M-1\}$, as desired. Finally note that $|\{l \in \{0,\cdots,N-1\} : \eta_l \neq 0\}| \leq \Delta$ since there are at most $\Delta$ nonzero values in the $\{\mathbf{y}^{(i)}\}_{i\neq j}$ collectively. Note also that $\eta_l \in GF(2^{b/\Delta})$. Thus in (17), we are mixing the $\{\gamma_l\}$ using at most $\Delta$ elements of $GF(2^{b/\Delta})$ and by the assumptions on the $\{\gamma_l\}$, the result of the combination must be nonzero, since at least one of the $\eta_l$ is nonzero. This proves that $d(B^{(-j)}) \geq \Delta+1$ as desired.

## VI. A comparison to Reed-Solomon codes

As discussed in the Preliminaries, Reed-Solomon codes may be considered for the memory array storage problem considered in this article; in here $H$ would no longer have the form (1) and instead would be the parity check matrix of a Reed-Solomon code. The main difficulty that arises in their application to the main memory of a server relates to the extraordinarily high bandwidth, low decoding latencies and small chip area footprint that this setting demands. A complete discussion of this topic is well beyond the scope of this paper and thus will be presented in a subsequent publication, nonetheless we will offer here some of the basic arguments.

First, we point out that if $NM < 2^b$, then a (generally shortened) Reed-Solomon code exists that is at least as strong as a diff-MDS code (in here of course we are also assuming the existence of the latter). For example, if one employs a general decoder organization similar to that in the proof of Theorem 1, then one can prove an analogous result in which we substitute (3,4,5,6,7) with

$$u + t_c^{(0,u)} + t_d^{(0,u)} < N+\Delta+1 \quad (18)$$

$$u + t_d^{(0,u)} < \Delta+1 \quad (19)$$

$$u + t_c^{(1,u)} + t_d^{(1,u)} < \Delta+1 \quad (20)$$

$$N \geq \Delta \quad (21)$$

$$p_c^{(0,u)} \geq 1 - \frac{M-1}{2^{b(\Delta-u)}}. \quad (22)$$

We now turn to decoding complexity; in what follows all operations discussed are in $GF(2^b)$ unless stated otherwise. In both diff-MDS and Reed-Solomon codes we start by computing the syndromes of the retrieved vector. It is easy to see that the $\mathbf{z}$ syndrome can be computed using $N(M-1)$ addition operations and that the $\mathbf{s}$ syndrome can be computed using approximately (and at most) $\Delta NM$ additions and multiplications, where the multiplications have one of its operands known at code design time. Syndrome

computation in Reed-Solomon codes can be accomplished using the obvious technique using approximately $(N+\Delta)NM$ additions and multiplications again with one of the operands of the latter being known at design time. Nonetheless, in some instances we can exploit techniques originally developed for the field of complex numbers [16] and argue that this operation can be accomplished using $O(NM\log^2 NM)$ operations; see [17] and its references for an early example of this idea. We note that these techniques rely on the ability to compute efficiently an interpolating polynomial and on the existence of a Fast Fourier Transform. Efficient methods for solving these in finite fields can be derived from algorithms in complex fields, nonetheless the finite field setting places restrictions on the block lengths and fields for which such efficient operations are known to exist; in particular, $2^b-1$ should preferably have many factors. Note that in order to obtain a conservative estimate of the relative computational complexity advantages of diff-MDS codes over Reed-Solomon codes we are not considering any benefits that our proposed diff-MDS codes can derive from exploiting the algebraic structure given by (10) and/or (12).

From this discussion it is clear that if $NM$ is large with respect to $\Delta$, then the complexity of computing syndromes for expurgated simple parity check codes can be much lower than that of Reed-Solomon codes. This is not very surprising, since $\Delta$ relatively "small" means that most of the errors that one will be solving for are "large" column errors and Reed-Solomon codes are not specifically designed for this setting while expurgated simple parity check codes are. Moreover, in specific problem instances, efficient "super-fast" algorithms cited above may not be known, since the associated complexity estimates are asymptotic, further magnifying the computational advantage of diff-MDS codes.

Assuming that the problem is simply to find the error magnitude of a channel that has been marked and no chip marked and no additional error, then it is easy to see that in the case of an expurgated simple parity check code, the syndrome $\mathbf{z}$ is the channel error magnitude. To achieve the corresponding with a Reed-Solomon code, one needs to decode $N$ erasures. A standard way of doing this is by solving an $N\times N$ system of $V\mathbf{e}=\mathbf{s}$ where $V$ is a Vandermonde matrix and $\mathbf{s}$ is a vector with $N$ syndromes of the Reed-Solomon code. It is well known that this system of equations can be using $O(N^2)$ multiplications and additions in $GF(2^b)$. As before, by using more advanced techniques [16] the complexity of these computations may be improved, in some instances, to $O(N\log^2 N)$, but it can certainly not be reduced to no computation at all which is what it is competing against.

The setting that we have presented in this paper goes well beyond the simple example given above, and therefore a complete assessment of the relative computational complexity advantages that we are suggesting requires a painstaking methodology in which optimized, complete decoders for both settings are presented. As stated earlier, this is beyond the scope of our article.

## VII. Appendix

**Proof of Lemma 1**. The upper bound in the lemma is proved by finding a vector $\xi$ with $w(\xi)=\lceil(\Delta+1)/(1-M^{-1})\rceil$ such that $H\xi=0$. Arrange the symbols of $\xi^{(0)},\cdots,\xi^{(M-2)}$, each an element of $GF(2^b)$, in a grid as follows:

$$\begin{pmatrix} \xi_0^{(0)} & \xi_0^{(1)} & \cdots & \xi_0^{(M-2)} \\ \xi_1^{(0)} & \xi_1^{(1)} & \cdots & \xi_1^{(M-2)} \\ \vdots & \vdots & \vdots & \vdots \\ \xi_{N-1}^{(0)} & \xi_{N-1}^{(1)} & \cdots & \xi_{N-1}^{(M-2)} \end{pmatrix}.$$

This arrangement is in agreement with the the first $M-1$ columns of Table I; the last column will be added shortly. We select $\Delta+1$ entries from the grid above to be allowed to contain nonzero values; the rest of the entries in the grid will have zero values. The entries are chosen so as to include as many whole rows as possible. One selection mechanism starts from the top, left corner, and progresses to the right, selecting entries towards the maximum count of $\Delta+1$. If the maximum is not reached, then entries from the second row are chosen starting from the leftmost, etc.

The total number of rows containing at least one chosen entry is $\lceil(\Delta+1)/(M-1)\rceil$. Since $\hat{H}^{(-(M-1))}$ has $\Delta$ rows, one can always find an assignment for the $\Delta+1$ entries selected so that $\hat{H}^{(-(M-1))}\xi^{\{-(M-1)\}}=0$. Now define

$$\xi^{(M-1)}=-\sum_{j=0}^{M-2}\xi^{(j)}. \tag{23}$$

Due to our selection mechanism, it s clear that $w(\xi)\le\Delta+1+\lceil(\Delta+1)/(M-1)\rceil=\lceil(\Delta+1)/(1-1/M)\rceil$. Note that

$$0=\sum_{j=0}^{M-2}\left(\hat{H}^{(j)}-\hat{H}^{(-(M-1))}\right)\xi^{(j)}=\hat{H}\xi.$$

This together with (23), implies that $H\xi=0$, finalizing the proof of the upper bound. We now turn our attention to the lower found. If $\xi\neq 0$ and $H\xi=0$ then for every $i\in\{0,\cdots,M-1\}$,

$$\hat{H}^{(-i)}\xi^{\{-i\}}=0, \qquad \xi^{\{-i\}}\neq 0.$$

The latter holds because if there is a unique $j$ such that $\xi^{(j)}\neq 0$ then necessarily $H\xi\neq 0$, contradicting the original assumption. This immediately implies that $w(\xi^{\{-i\}})\ge d(\hat{H}^{(-i)})$. Since $w(\xi)=w(\xi^{(i)})+w(\xi^{\{-i\}})$ then we obtain

$$\begin{aligned} d(H) &\ge \min_{\xi\neq 0:H\xi=0} w(\xi) \\ &\ge \min_{\xi\neq 0:H\xi=0}\max_{0\le i\le M-1}\left(w(\xi^{(i)})+d(\hat{H}^{(-i)})\right) \\ &\ge \min_{0\le i\le M-1} d(\hat{H}^{(-i)})+\min_{\xi\neq 0:H\xi=0}\max_{0\le i\le M-1} w(\xi^{(i)}) \\ &\ge \min_{0\le i\le M-1} d(\hat{H}^{(-i)})+\lceil d(H)/M\rceil \\ &\ge \min_{0\le i\le M-1} d(\hat{H}^{(-i)})+d(H)/M. \end{aligned}$$

From the above, $d(H) \geq d_{\text{diff}}(\hat{H})/(1 - M^{-1})$. Since the minimum distance is integer valued, we can take the ceiling, obtaining the lower bound in the lemma.

**Proof of Lemma 2**. Let $\{j_0, \cdots, j_{u-1}\}$ be the erasure locations passed to the decoder $g$. As it is well known, in order to solve for $u$ erasures, the decoder $g$ computes the syndrome $\mathbf{s}$ by multiplying the matrix $A$ times the retrieved vector, extracts the $u$ columns of $A$ corresponding to the $u$ erasures and solves the linear system $[A_{j_0} A_{j_1} \cdots A_{j_{u-1}}]\mathbf{v} = \mathbf{s}$ for the vector $\mathbf{v}$, which will contain the error magnitudes. This can be accomplished because by assumption, the $u$ columns of the matrix above are linearly independent. Note that for any given choice of $\{j_0, \cdots, j_{u-1}\}$, there are exactly $2^{ub}$ distinct possible values for $\mathbf{s}$ that must be mapped to distinct error magnitudes. Let these "correctable" syndromes be denoted by $\mathcal{C}_{j_0,\cdots,j_{u-1}}$. The decoder will claim an uncorrectable error whenever the calculated syndrome $\mathbf{s}$ is not in $\mathcal{C}_{j_0,\cdots,j_{u-1}}$, and otherwise will claim a correctable error.

For the remainder of the proof, let $\mathbf{s} = A(\mathbf{x} + \mathbf{e})$ with $\mathbf{x}$ and $\mathbf{e}$ defined as in the lemma statement. Since any $d(A) - 1$ columns of the matrix $A$ are linearly independent, there must be $d(A) - 1$ rows of the matrix $[A_{i_0} A_{i_1} \cdots A_{i_{d(A)-2}}]$ that are linearly independent. Let the indices of these rows be $\{h_0, \cdots, h_{d(A)-2}\}$. Extracting these rows results in an invertible square matrix $M$. Note that the vector

$$\xi = M \left( \begin{array}{cccc} \mathbf{e}_{i_0} & \mathbf{e}_{i_1} & \cdots & \mathbf{e}_{i_{d(A)-2}} \end{array} \right)^T \tag{24}$$

is, statistically speaking, a vector chosen uniformly at random from $GF(2^b)^{NM}$. The reason for this is that the right hand side has this property, and $M$ is an invertible matrix. The syndrome $\mathbf{s}$, when subsampled at the same rows, can be written as: $\mathbf{s}_{\{h_0,\cdots,h_{d(A)-2}\}} = \xi + \chi$ for some vector $\chi$. The vector $\chi$, which in general can have both random and deterministic entries, is statistically independent from $\xi$. The reason is that any random component of $\chi$ depends on elements of $\mathbf{e}$ not in the list $\{\mathbf{e}_{i_0}, \cdots, \mathbf{e}_{i_{d(A)-2}}\}$, and all the entries of $\mathbf{e}$ are independent. As a result, $\mathbf{s}_{\{h_0,\cdots,h_{d(A)-2}\}}$ is also a vector chosen uniformly at random from $GF(2^b)^{NM}$. Through a similar reasoning, we can see that the vector $\mathbf{s}_{\{h_0,\cdots,h_{d(A)-2}\}}$ is statistically independent of the vector $\mathbf{s}_{\{h_0,\cdots,h_{d(A)-2}\}^c}$ where the complement is taken with respect to the index set $\{0, \cdots, r-1\}$. As a consequence of this, if $\mathbf{z} \in GF(2^b)^{NM}$ is any given deterministic vector, we have that

$$P(\mathbf{s} = \mathbf{z}) \leq \frac{1}{2^{b(d(A)-1)}}.$$

Finally note that for any given erasure locations $\{j_0, \cdots, j_{u-1}\}$ the set of correctable syndromes $\mathcal{C}_{j_0,\cdots,j_{u-1}}$ has cardinality exactly $2^{bu}$. Our decoder miscorrects whenever $A\mathbf{e} \in \mathcal{C}_{j_0,\cdots,j_{u-1}}$ and thus the miscorrection probability must satisfy

$$P\left(\mathbf{s} \in \mathcal{C}_{j_0,\cdots,j_{u-1}}\right) \leq \frac{1}{2^{b(d(A)-u-1)}}.$$

This finishes the proof of the lemma.

## VIII. Acknowledgements

We thank Shmuel Winograd for numerous insightful discussions on error control coding and computational complexity. We also acknowledge the support that IBM's Systems and Technology Group and IBM Research gave to this research and to the larger system project to which it belongs. We give a special acknowledgement to the team that was responsible for the RAIM ECC design testing and formal verification, in particular to the excellent work of Alia Shah, Gero Dittmann and Raimund Breil.

## References

[1] M.Y. Hsiao C.L. Chen. "Error-correcting codes for semiconductor memory applications: A state-of-the-art review." IBM Journal of Research and Development, 28(2):124134, March 1984.

[2] T. C. May and M. H. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic Memories, *IEEE Trans. Electron Devices* 26, No. 1, 29 (1979).

[3] J. F. Ziegler and W. A. Lanford, "Effect of Cosmic Rays on Computer Memories, *Science* 206, No. 4420, 776788 (1979).

[4] T. J. Dell, "A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory, IBM Corporation, white paper; see http://www.ece.umd.edu/courses/enee759h.S2003/references/chipkill_white_paper.pdf.

[5] T. J. Dell, "System RAS implications of DRAM soft errors", *IBM Journal of Research and Development*, Volume 52, No. 3, May 2008

[6] P.J. Meaney, L.A. Lastras-Montaño, V. Papazova, E. Stephens, J. Johnson, L.C. Alves, J. O'Connor, W. Clarke, "zEnterprise RAIM Memory Subsystem", to appear in the *IBM Journal of Research and Development, zEnterprise Special Issue (2011)*.

[7] S.H. Reiger, "Codes for the correction of clustered errors", *IRE Trans. Inf. Theory*, Vol IT-6, No 1, pp 16-21 (1960)

[8] R.C. Singleton, "Maximum distance q-nary codes", *IEEE Trans. Inf. Theory* 10: 116118.(1964)

[9] R.E., Blahut, "Algebraic Codes for Data Transmission" *Cambridge University Press*.

[10] M. Blaum, P. G. Farrell and H. van Tilborg, "Array Codes", *Handbook of Coding Theory*, edited by V. S. Pless and W. C. Huffman, Elsevier Science B. V., Chapter 22, pp. 1855-1909, 1998.

[11] M. Blaum, P.G. Farrell, H. van Tilborg, "A class of burst error-correcting array codes" *IEEE Transactions on Information Theory 32(6): 836-839 (1986)*.

[12] M. Blaum, R.M. Roth, "On Lowest Density MDS codes". *IEEE Transactions on Information Theory 45(1): 46-59 (1999)*.

[13] P. Fire, "A class of Multiple-Error Correcting Binary Codes for Non-Independent Errors", *Sylvania Report RSL-E-2, Sylvania Reconnaissance Systems Lab, Mountain View, CA, 1959*.

[14] R.M. Roth, "Maximum-rank array codes and their application to crisscross error correction", *IEEE Transactions on Information Theory*, March 1991, Vol. 37, Issue 2

[15] E. M. Gabidulin "Theory of codes with maximum rank distance", *Probl. Inform. Transm.*, pp. 1-12, July 1985. Translated from Russian, vol. 21, no. 1, pp 3-16, Jan.-Mar., 1985

[16] I. Gohberg, V. Olshevsky, "Fast algorithms with preprocessing for matrix-vector multiplication problems", *Journal of Complexity*, 1994

[17] Jørn Justesen, "On the complexity of decoding Reed-Solomon Codes", *IEEE Transactions on Information Theory*, pp 237-238, Vol. 22, Issue 2, March 1976