Error-Correcting Codes for Flash Coding

Qin Huang, Shu Lin and Khaled Abdel-Ghaffar
Electrical and Computer Engineering Department
University of California, Davis
Email: \{qinhuang, shulin, ghaffar\}@ece.ucdavis.edu

Abstract

Flash memory is a non-volatile computer storage device which consists of blocks of cells. While increasing the voltage level of a single cell is fast and simple, reducing the level of a cell requires the erasing of the entire block containing the cell. Since block-erasures are costly, traditional flash coding schemes have been developed to maximize the number of writes before a block-erasure is needed. A novel coding scheme based on error-correcting codes allows the cell levels to increase as evenly as possible and as a result, increases the number of writes before a block-erasure. The scheme is based on the premise that cells whose levels are higher than others need not be increased. This introduces errors in the recorded data which can be corrected by the error-correcting code provided that the number of erroneous cells are within the error-correcting capability of the code. The scheme is also capable of combating noise, causing additional errors and erasures, in flash memories in order to enhance data reliability. For added flexibility, the scheme can be combined with other flash codes to yield concatenated schemes of high rates.

Index Terms

Block-write, block-erasure, code rate, concatenated code, controllable errors, error-correcting code, flash code, WOM code

This research was supported by NSF under the Grants CCF-0727478 and CCF-1015548 and gift grants from Northrop Grumman Space Technology, Intel and Denali Software Inc.. Part of this paper is presented at GlobeCom 2010.
I. Introduction

Flash memory is a non-volatile computer storage device, i.e., it can retain the stored information even if the power is turned off [25], [4]. It can be considered as a specific type of EEPROM (Electrically Erasable Programmable Read-Only Memory) that is erased and programmed in large blocks. However, they are cheaper and faster than other EEPROM devices. Actually, the name “flash” is associated with the speed at which large data blocks can be erased which is similar to that of the flash of a camera. In contrast, old-style EEPROM allows only the simultaneous erasure of bytes. For these reasons, flash memories are becoming dominant as secondary memory devices in digital cameras, digital audio players, mobile phones, and PC basic input/output system (BIOS) chips. A flash memory is an array of cells that consist of floating gate transistors. Information is stored as an electric charge in each cell. In multi-level cell (MLC) devices [8], [14], which we consider in this paper, a cell can assume $q \geq 2$ possible voltage levels.

However, there remain many technical challenges in flash memories. Adding charge to a single cell is easy, but removing charge from a cell requires erasing the entire block containing that cell and reprogramming all cells in that block. These block-erasures are time-consuming and can also cause physical degradation and shorten memory life. Therefore, it is important to reduce the frequency of block-erasures [11]. Coding techniques have been introduced to accomplish this. A variety of coding schemes for flash memories were introduced such as floating codes [17], [18], [20], [28], [19], buffer codes [3], [19], trajectory codes [21], multidimensional flash codes [35], and rank modulation codes [23], [32].

Another challenge in flash memories is data loss due to charge leakage, read/write disturbance and over-programming. These result in errors and erasures in flash memories. Therefore, error-correcting codes are necessary for error correction in flash memories [13], [5], [22]. Some flash coding schemes based on rank modulation [24], [31], [1], have been proposed. However, most flash coding schemes do not accommodate any error-correcting capability. Thus, an error-correcting code needs to be imposed on top of the flash code to correct the errors.

What is more, flash memories can be thought of as a generalization of write once memory (WOM) to multilevel cells. WOMs were first proposed by Rivest and Shamir in [30] to model memories like punch cards and optical disks where a memory element can change from a zero-state to a one-state but not vice versa. Coding schemes for WOMs have been investigated in [30], [7], [12], [29], [33], [26]. Furthermore, WOM codes that can also correct errors are proposed in [36], [34]. Generalized WOMs can have more
than two states [9]. The capacities of WOMs and generalized WOMs were determined in [15], [10].

In this paper, we propose a new flash coding scheme based on error-correcting codes which aims to minimize the frequency of block-erasures by inducing some controllable errors. Messages are recorded on the flash memory as vectors of lengths equal to the block size. These vectors, called cell-state vectors, represent the voltage levels of the individual cells in the block. Contrary to floating codes, in which a single cell changes its level with each write, we consider schemes with block-writes in which the levels of several cells can increase to represent a message. This is identical to the case of generalized WOM codes where the levels of some cells are increased to produce a cell-state vector which identifies the message. However, our key idea is not to increase the levels of some cells in a block when their levels exceed the levels of other cells. This introduces controllable errors. Then in reading operations, error-correcting codes help to recover the correct message. Thus, it is much less likely that a block-erasure is incurred by few cells which cannot be re-written. In other words, the frequency of block-erasures can be significantly reduced. Since error-correcting systems are already used to combat errors and erasures in flash memories, the proposed scheme, unlike other existing schemes, does not require additional circuitry. The newly proposed scheme allows for a flexible trade-off between the scheme’s capability to correct errors and erasures in flash memories and its ability to reduce the frequency of block-erasures. It can also be used in concatenation with other flash memory coding schemes. With appropriate choice of the error-correcting codes, the resulting flash codes have higher rates than other flash codes.

In this work, we use \( \mathbb{A}_q \), where \( q \) is a positive integer, to denote the set \{0, 1, \ldots, q - 1\} of \( q \) integers. We also use \((a)_2\), where \( a \) is an integer, to denote \( a \) reduced modulo 2, i.e., \((a)_2 = 0\) if \( a \) is even and \((a)_2 = 1\) if \( a \) is odd. If \( \mathbf{a} = (a_1, a_2, \ldots, a_n) \) is an integer vector, then \((\mathbf{a})_2 = ((a_1)_2, (a_2)_2, \ldots, (a_n)_2)\) is a binary vector. The vector \((\mathbf{a})_2\) is called the reduced binary vector of \( \mathbf{a} \).

The rest of the paper consists of three sections. In Section II, we propose the flash coding scheme based on error-correcting codes and discuss several of its variations. Analysis and simulations are presented in Section III. The paper is concluded in Section IV.

II. Flash Coding Schemes Based on Error-Correcting Codes

In this section, some background on error-correcting codes is introduced. Then, a binary flash coding scheme and a non-binary flash coding scheme are proposed to reduce block-erasures. Moreover, simplifications and enhancements are discussed.
A. Error-Correcting Codes

A $Q$-ary $(N, M, d)$ code $C$ consists of $M$ codewords, each is a sequence of $N$ symbols from a set, $Q$, composed of $Q$ symbols such that the minimum Hamming distance between two distinct codewords is $d$. In this paper, $Q$ is always a power of 2, i.e., $Q = 2^s$ for some integer $s$. If $Q = 2$, then $C$ is a binary code. If the code is linear, then $M = Q^K$ where $K$ is the dimension of the code. In this case, we say that $C$ is an $[N, K, d]$ code over $\text{GF}(2^s)$. The $M$ codewords in $C$ can be used to represent messages in a set $\mathcal{M}$ of $M = |\mathcal{M}|$ messages. This is done by associating to the code $C$ an encoding function $E_C : \mathcal{M} \rightarrow C$, which is a one-to-one correspondence that maps a message $m$ to a codeword $E_C(m)$. Due to noise, a symbol in a codeword representing a message may change its value resulting in an erasure or an error. Erasures correspond to erroneous symbols denoted by “?” in known positions while errors correspond to erroneous symbols in unknown positions. A code of minimum Hamming distance $d$ is capable of correcting any pattern of $t$ errors and $e$ erasures \cite{27}, \cite{16} provided that

$$d \geq 2t + e + 1. \quad (1)$$

This ability is realized by a decoding function $D_C : (Q \cup ?)^N \rightarrow \mathcal{M} \cup E_C$ that maps a sequence $v$ of length $N$ with $e$ erasures into a message $m$ such that its codeword $E_C(m)$ differs from $v$ in at most $t$ unerased positions if such message exists, and to $E_C$, a symbol representing a decoding failure if otherwise.

B. Flash Memory Codes Based on Binary Error-Correcting Codes

Consider a $q$-ary flash memory composed of $n$ cells that stores a message from a set $\mathcal{M}$ composed of $M = |\mathcal{M}|$ messages. Each message $m \in \mathcal{M}$ is represented by values stored in the $n$ cells, where each value belongs to the set $\mathcal{A}_q = \{0, 1, \ldots, q-1\}$. The sequence of values is denoted by $a = (a_1, a_2, \ldots, a_n)$ with $a_i \in \mathcal{A}_q$, which is called the cell-state vector. The cell-state vector representing the message $m$ depends on $m$ and the cell-state vector representing the previous message. We propose a scheme to accomplish this by using a binary $(N, M, d)$ error-correcting code $C$ of length $N = n$, with encoding and decoding functions $E_C$ and $D_C$, respectively. The block-write flash memory code $F$ is characterized by encoding and decoding functions denoted by $E_F$ and $D_F$, respectively. The decoding function, $D_F : \mathcal{A}_q^n \rightarrow \mathcal{M} \cup E_C$ is given by $D_F(a) = D_C((a)_2)$. Thus, the decoding function $D_F$ maps a vector of length $n$ over $\mathcal{A}_q$ into the message such that its codeword is at Hamming distance at most $\lfloor (d-1)/2 \rfloor$ from its reduced binary vector $(a)_2$ if such message exists. Otherwise, it maps the vector over $\mathcal{A}_q$ to $E_C$.

The encoding function is characterized by a nonnegative integer $t_F \leq \lfloor (d-1)/2 \rfloor$. Let $a = (a_1, a_2, \ldots, a_n)$ be the current cell-state vector and $c = E_C(m) = (c_1, c_2, \ldots, c_n)$ be the codeword in $C$ corresponding to
the message \(m\) which is to be recorded on the flash memory. Let \(I_0(c,a) = \{i : i = 1, 2, \ldots, n, c_i \neq (a_i)_2\}.\) To faithfully represent the message \(m\), i.e., for the reduced binary vector of the new cell-state vector to be equal to the codeword \(\mathbf{c} = E_C(m)\), all cells with indices in \(I_0(c,a)\) should increase their levels by 1. However, our main idea is not to increase the levels of those cells in which the levels are already the highest among all cells with indices in \(I_0(c,a)\). By doing this, we introduce errors. Provided that the number of errors is within the error-correcting capability of the error-correcting code \(C\), the decoder can correctly retrieve the message \(m\). For this purpose, let \(I(c,a) \subseteq I_0(c,a)\) be a subset of size \(\min\{|I_0(c,a)|, t_F\}\) such that for all \(i \in I(c,a)\) and \(i' \in I_0(c,a) \setminus I(c,a)\), we have \(a_i \geq a_{i'}\). The subset \(I(c,a)\), which is not necessarily unique, contains indices of cells in which the levels are highest among all cells with indices belonging to \(I_0(c,a)\). The levels of cells with indices in \(I(c,a)\) will not be increased. Thus, we define \(c \odot I a = (a'_1, a'_2, \ldots, a'_n)\), where

\[
a'_i = \begin{cases} 
a_i + 1, & \text{if } i \in I_0(c,a) \setminus I(c,a) \\
a_i, & \text{otherwise.}
\end{cases}
\]

and the sum \(a_i + 1\) is over the integers. The encoding function of the flash code is then \(E_F : \mathcal{M} \times \mathcal{A}_q^n \rightarrow \mathcal{A}_q^n \cup E_F\) which encodes the message \(m \in \mathcal{M}\) by updating the cell-state vector \(a\) to \(E_C(m) \odot I a\) if all its components are less than \(q\) and to \(E_F\) otherwise, where \(E_F\) denotes an encoding failure for the flash memory code. If the encoding does not result in \(E_F\), then we say that it is successful. In this case, the reduced binary vector of the cell-state vector differs from the codeword representing the message \(m\) in \(|I(c,a)|\) positions. Since \(|I(c,a)| \leq t_F \leq [(d - 1)/2]\), the flash memory decoder can correctly retrieve the message. Notice that the cells with indices belonging to \(I(c,a)\) are not updated deliberately to avoid increasing their levels. The errors in these cells are called \textit{controllable errors} (CE). Introducing controllable errors is the main feature of our scheme that allows for more successful block-writes before requiring a block-erasure.

\textbf{Example 1.} We give an example where \(n = 7\) and \(q = 3\). For the code \(C\), we use the binary \([7, 4, 3]\) cyclic Hamming code. We set \(t_F = 1\). There are 16 messages and each message is represented by a vector of length four over \(GF(2)\). Encoding is accomplished by multiplying the message polynomial by the generator polynomial \(1 + x + x^3\) [27]. In this example, starting with the all-zero cell-state vector, we encode the messages \((1100), (0001), (1011), (1010), (0100), (0011)\). Encoding this sequence of messages results in a sequence of cell-state vectors and a sequence of controllable errors as displayed in Table I.

As an example, the fifth message \((0100)\) is encoded using the code \(C\) into the codeword \(c = (0110100)\).
TABLE I
MESSAGE RECORDING AND STATE TRANSITION PROCESS OF EXAMPLE 1.

<table>
<thead>
<tr>
<th>Current state</th>
<th>m</th>
<th>c = EC(m)</th>
<th>I_0(c, a)</th>
<th>I(c, a)</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0000000)</td>
<td>(1100)</td>
<td>(1011100)</td>
<td>{1, 3, 4, 5}</td>
<td>{1}</td>
<td>(0011100)</td>
</tr>
<tr>
<td>(0011100)</td>
<td>(0001)</td>
<td>(0001101)</td>
<td>{3, 7}</td>
<td>{3}</td>
<td>(0111101)</td>
</tr>
<tr>
<td>(0011101)</td>
<td>(1011)</td>
<td>(1111111)</td>
<td>{1, 2, 6}</td>
<td>{6}</td>
<td>(1111101)</td>
</tr>
<tr>
<td>(1111101)</td>
<td>(1010)</td>
<td>(1110010)</td>
<td>{4, 5, 6, 7}</td>
<td>{7}</td>
<td>(1112211)</td>
</tr>
<tr>
<td>(1112211)</td>
<td>(0100)</td>
<td>(0110100)</td>
<td>{1, 5, 6, 7}</td>
<td>{5}</td>
<td>(2112222)</td>
</tr>
<tr>
<td>(2112222)</td>
<td>(0011)</td>
<td>(0010111)</td>
<td>{2, 5, 6, 7}</td>
<td>{6}</td>
<td>E_F</td>
</tr>
</tbody>
</table>

Since the cell-state vector is (1112211), then, if no controllable errors are introduced, the new cell-vector representing (0100) should be (2112322). This leads to an encoding failure as one of the levels is 3 which exceeds the maximum level q − 1 = 2. However, since t_F = 1, we are allowed one controllable error and we record the message (0100) using the cell-state vector (2112222). The reduced binary vector of this cell-state vector is (0110000) with a single error at the fifth position. Since the error-correcting code can correct one error, the correct message can be correctly recovered from the recorded cell-state vector. Notice that this flash code accommodated five block-writes. The sixth block-write results in a block-erasure. On the other hand, if no coding is used, i.e., the message of four bits is recorded directly on a flash memory composed of four cells, then, for the same sequence of messages, only two block-writes can be accommodated since, starting from the all-zero cell-state vector of length four, the first message (1100) will be recorded as (1100), the second message (0001) will be recorded as (2201), and the third message (1011) will result in a block-erasure. This simple scheme falls within our framework by using the trivial binary error-correcting code [4, 4, 1] and setting EC(m) = m and t_F = 0.

C. Flash Memory Codes Based on Q-ary Error-Correcting Codes

The scheme described above uses binary error-correcting codes. In practice, non-binary codes, and in particular Reed-Solomon (RS) codes [2], [27], are widely used due to their ease of implementation and large minimum Hamming distances. By representing each symbol from an alphabet of size Q = 2^s as a binary vector of length s, a Q-ary (N, M, d) code, C, can be viewed as a binary (sN, M, d') code, C', where d' ≥ d. The code C' is called the binary image of C. It can then be employed as the error-correcting code component in a scheme with n = sN memory cells as explained in the previous subsection. The parameter t_F is chosen to satisfy t_F ≤ ⌊(d − 1)/2⌋. However, since multiple errors affecting bits falling in a vector representing the same Q-ary symbol give rise to only one symbol error, more controllable errors can be allowed as long as the number of affected symbols, rather than affected bits, does not
exceed $t_F$. This improves the capability of flash memories to avoid block-erasures. To accomplish this, some modifications to the encoding and decoding functions have to be made as explained below.

Consider a flash memory composed of $n = sN$ cells that stores a message from a set $M$ of $M = |M|$ messages. We use a $Q$-ary $(N, M, d)$ code, $C$, with symbols in a set $Q$ of size $Q = 2^s$ for some positive integer $s$. To the code $C$, we associate the encoding function $E_C$ and the decoding function $D_C$.

The block-write flash memory code $F$ is characterized by encoding and decoding functions denoted by $E_F$ and $D_F$, respectively. Let $a = (a_1, a_2, \ldots, a_n)$, a $q$-ary sequence of length $n = sN$ over $A_q$, be the cell-state vector. Then, $(a)_2$ is a binary vector of length $sN$ which can be considered as a vector of length $N$ over GF($2^s$) using a one-to-one correspondence between binary vectors of length $s$ and symbols in GF($2^s$). (Such a one-to-one correspondence can be established, for example, by representing an element in the finite field GF($2^s$) as a polynomial of degree less than $s$, with binary coefficients, in a fixed primitive element in the field. The binary vector representing the element is the ordered $s$-tuple listing the coefficients, see, e.g., Section 2.4 in [27] for the details.) With this correspondence, the decoding function, $D_F : A_q^n \rightarrow M \cup E_C$ is given by $D_F(a) = D_C((a)_2)$.

The encoding function is characterized by an integer $t_F \leq \lfloor (d-1)/2 \rfloor$. Let $a = (a_1, a_2, \ldots, a_n)$ be the current cell-state vector and $c = E_C(m) = (c_1, c_2, \ldots, c_n)$ be the codeword in $C$, viewed as a binary sequence of length $n = sN$, corresponding to the message $m$ which is to be recorded on the flash memory. For each $i = 1, 2, \ldots, N$, we define the following set of indices:

$$Z_i = \{ j : j = 1, 2, \ldots, s, c(i-1)s+j \neq (a(i-1)s+j)_2 \},$$

and for each $i$ such that $Z_i$ is nonempty we define the integer

$$a_{i,\text{max}} = \max\{a(i-1)s+j : j \in Z_i \}.$$

To faithfully represent the message $m$, i.e., for the reduced binary vector of the new cell-state vector to be equal to the codeword $c = E_C(m)$, viewed as a binary vector of length $n = sN$, all cells with indices $(i-1)s + j$, $j \in Z_i$, for every $i = 1, 2, \ldots, N$, should increase their levels by 1. However, for some values of $i$, we do not alter the levels of any of these cells. These values of $i$ are selected such that the maximum level, $a_{i,\text{max}}$, of cells with indices $(i-1)s + j$, $j \in Z_i$, is maximum. This introduces controllable symbol errors rather than controllable bits errors as in the previous subsection. Let $I_0(c, a) = \{ i : i = 1, \ldots, N, Z_i \neq \emptyset \}$ and $I(c, a) \subseteq I_0(c, a)$ be a subset of size $\min\{|I_0(c, a)|, t_F\}$
such that for all $i \in \mathcal{I}(c, a)$ and $i' \in \mathcal{I}_0(c, a) \setminus \mathcal{I}(c, a)$, we have

$$a_{i, \text{max}} \geq a_{i', \text{max}}.$$  

(2)

Define $c \otimes \mathcal{I} a = (a'_1, a'_2, \ldots, a'_n)$, where

$$
a'_{(i-1)s+j} = \begin{cases} 
a_{(i-1)s+j} + 1, & \text{if } i \in \mathcal{I}_0(c, a) \setminus \mathcal{I}(c, a) \text{ and } j \in \mathbb{Z}, \\
a_{(i-1)s+j}, & \text{otherwise.} \end{cases}
$$

The encoding function of the flash code $E_F$ encodes the message $m \in M$ by updating the cell-state vector $a$ to $c \otimes \mathcal{I} a$ if all its components are less than $q$ and to $E_F$ otherwise. There are $t_F$ or less controllable symbol errors in the sequence $c \otimes \mathcal{I} a$ which can be corrected by the error-correcting code since $t_F \leq \lfloor (d - 1)/2 \rfloor$.

Note that the subset $\mathcal{I}(c, a)$ is not necessarily unique. In case of more than one option, we select the subset $\mathcal{I}(c, a)$ for which the sum of $|Z_i|$ over all $i \in \mathcal{I}(c, a)$ is maximum. This minimizes the number of cells in which the levels need to be incremented.

Notice that the scheme of Subsection II-B can be considered as a special case of the scheme in this subsection where $a_{i, \text{max}} = a_i$.

D. Selection of Controllable Errors by Threshold Method

Selecting controllable errors for the scheme in Subsection II-C based on (2) requires sorting operations. This is also the case for the scheme of Subsection II-B where $a_{i, \text{max}} = a_i$. Therefore, we propose a simpler and parallelizable method to choose the cells with controllable errors. Let $a_{\text{max}} = \max\{a_1, a_2, \ldots, a_n\}$ be the maximum level in the cell-state vector $a$. If the number of controllable errors allowed, $t_F$, is small compared to $N$, then most values of $a_{i, \text{max}}$ for $i \in \mathcal{I}(c, a)$ are equal to $a_{\text{max}}$ or $a_{\text{max}} - 1$. Thus, it is natural to set thresholds on $a_{i, \text{max}}$ to select controllable errors.

Consider the maximum level $a_{\text{max}}$ as the first threshold and define

$$T_1 = \{i : i = 1, 2, \ldots, N, a_{i, \text{max}} = a_{\text{max}}\}.$$

If $|T_1| \geq t_F$, then $t_F$ entries of $T_1$ are randomly chosen to form $\mathcal{I}(c, a)$. If not, we select all entries of $T_1$ and use $a_{\text{max}} - 1$ as the second threshold to select more symbols in

$$T_2 = \{i : i = 1, 2, \ldots, N, a_{i, \text{max}} = a_{\text{max}} - 1\}.$$

Then $\min\{|T_2|, t_F - |T_1|\}$ entries of $T_2$ are added to $\mathcal{I}(c, a)$. In particular, $\mathcal{I}(c, a)$ is either a subset of
$T_1$ or the union of $T_1$ and a subset of $T_2$. Selecting controllable errors based on thresholds is in general suboptimal since only values of $a_{i,\text{max}}$ which equal $a_{\text{max}}$ or $a_{\text{max}} - 1$ are considered for controllable errors and the number of these values may be less than $t_F$. However, using thresholds to determine the set $I(c,a)$ is computationally simple and causes only slight performance degradation as shown by simulations presented later in Section III.

**E. Correction of Errors and Erasures in Flash Memories**

The scheme we have developed based on error-correcting codes naturally allows for the correction of the recorded data if the values stored in the memory cells suffer from errors or erasures. Since the flash memory encoder introduces at most $t_F$ controllable errors and the minimum Hamming distance of the error-correcting code is $d$, then, from (1), the code can correct $t$ additional errors and $e$ erasures provided that

$$d - 2t_F \geq 2t + e + 1.$$  \hfill (3)

Under this condition, the decoding function $D_F$ retrieves the recorded message correctly.

Furthermore, by recording the locations of the controllable errors, the controllable errors become *controllable erasures* (CEr). We would like to determine the correction capability of the scheme in this case and compare it with (3). Notice that the cells recording the locations of the controllable erasures may suffer from errors leading them to point to locations not having controllable erasures or erasures leading them to lose the locations they are storing. If a cell recording the location of a controllable erasure is erased, then effectively it turns the controllable erasure into a controllable error. On the other hand, if the same cell suffers from an error, then it turns the controllable erasure into a controllable error and it may cause an additional controllable erasure. Assume that in total there are $t$ errors and $e$ erasures, of which $t'$ errors and $e'$ erasures, respectively, affect cells recording the controllable erasures. Then, the decoder of the error-correcting code may have to decode a word suffering from $t + t' + e'$ errors and $t_F + e - e'$ erasures. According to (1), this is feasible provided that

$$d - t_F - 2t' - e' \geq 2t + e + 1.$$  \hfill (4)

Comparing (4) with (3), it follows that recording the locations of the controllable errors, i.e., turning them into controllable erasures, improves the correcting capability of the scheme provided that $t_F > 2t' + e'$. In particular, if neither errors nor erasures occur, then definitely there is an improvement if $t_F \geq 1$.

However, there is a cost incurred by recording the locations of the controllable erasures since a number
of cells, which is logarithmic in \( n \), need to be reserved to record each location. This results in a reduction of the code rate. Furthermore, the levels of the cells reserved to record the locations may increase drastically causing a block-erasure even if the scheme is capable of controlling the levels of the other cells. This problem can be alleviated by dedicating more cells to record the locations or by using different cells to store the locations for each write. In this case, the write number needs to be also recorded in order to determine the identity of cells storing the locations. It is argued in [33] that it is possible by using replicas of the flash coding scheme to make the loss in rate due to the storage of the write numbers as small as we want.

**F. Concatenation with Traditional Flash Coding Schemes**

Concatenation is a well-known coding technique which combines two or more simpler codes in order to achieve good performance with reasonable complexity [27]. The proposed scheme can also be applied in concatenation with other flash memory schemes, not only discrete multilevel flash codes, but also rank modulation. As shown in Fig. 1, the concatenated system consists of the proposed scheme as the *outer code* and a traditional flash coding scheme as the *inner code*.

We use as an outer code a \( Q \)-ary \((N, M, d)\) error-correcting code, \( C \), with symbols in an alphabet \( Q \) of size \( Q = 2^s \) for some positive integer \( s \). The encoding function of \( C \) is \( E_C : \mathcal{M} \rightarrow C \), which is a one-to-one correspondence that maps a message \( m \) from a set \( \mathcal{M} \) of \( M \) messages into a codeword \( E_C(m) \). The decoding function of \( C \) is \( D_C : (Q \cup \varnothing)^N \rightarrow \mathcal{M} \cup E_C \) which maps a sequence of length \( N \) into a message \( m \in \mathcal{M} \) or to \( E_C \), a symbol representing a decoding failure.

As an inner code, we use a traditional flash code, \( F' \), that records messages from \( Q \) on \( n' \) cells. The inner flash code, \( F' \), is characterized by an encoding function \( E_{F'} \) and a decoding function \( D_{F'} \). The encoding function \( E_{F'} : Q \times A_q^{n'} \rightarrow A_q^{n'} \cup E_{F'} \) maps a message from \( Q \) into a cell-state vector based on the current cell-state vector unless an encoding failure occurs which is denoted by \( E_{F'} \). The decoding function \( D_F : A_q^{n'} \rightarrow Q \) retrieves the message from the cell-state vector.

The concatenated flash coding scheme, denoted by \( F \), records messages from the set \( \mathcal{M} \) onto \( n = n'N \) cells. The scheme is characterized by encoding and decoding functions denoted by \( E_F \) and \( D_F \), respectively. Let \( a = (a_1, a_2, \ldots, a_N) \) be the cell-state vector, where \( a_i \) is a \( q \)-ary sequence of length \( n' \). Then \( D_F = D_C(D_{F'}(a_1), D_{F'}(a_2), \ldots, D_{F'}(a_N)) \) defines the decoding function of \( F \).

The encoding function is characterized by an integer \( t_F \leq \lfloor (d - 1)/2 \rfloor \). Let \( a = (a_1, a_2, \ldots, a_N) \) be the current cell-state vector and \( c = E_C(m) = (c_1, c_2, \ldots, c_N) \), where \( c_i \in Q \) for \( i = 1, 2, \ldots, N \), be the codeword in \( C \) corresponding to the message \( m \) which is to be recorded on the flash memory.
We define the set of indices $I_0(c,a) = \{i : i = 1, 2, \ldots, N, c_i \neq D_F'(a_i)\}$ and for each integer $i = 1, 2, \ldots, N$, we define $a_{i, \text{max}}$ to be the maximum level in the cell-state vector $a_i$. To faithfully represent the message $m$, for each $i \in I_0(c,a)$, the cell-state vector should be updated such that decoding it with $D_F'$ yields the symbol $c_i$. However, for a number not exceeding $t_F$ of values of $i$, we do not alter the cell-state vector $a_i$. These values of $i$ are selected such that the maximum level, $a_{i, \text{max}}$, in $a_i$ is maximum. This introduces controllable symbol errors which can be corrected by the outer code $C$ since $t_F \leq \lfloor (d-1)/2 \rfloor$. Let $I(c,a) \subseteq I_0(c,a)$ be a subset of size $\min\{|I_0(c,a)|, t_F\}$ such that for all $i \in I(c,a)$ and $i' \in I_0(c,a) \setminus I(c,a)$, we have $a_{i, \text{max}} \geq a_{i', \text{max}}$. Define $c \otimes_I a = (a'_1, a'_2, \ldots, a'_n)$, where

$$a'_i = \begin{cases} E_{F'}(c_i), & \text{if } i \in I_0(c,a) \setminus I(c,a) \\ a_i, & \text{otherwise.} \end{cases}$$

The encoding function of the concatenated flash code $E_F$ encodes the message $m \in M$ by updating the cell-state vector $a$ to $c \otimes_I a$ if $E_{F'}(c_i) \neq E_F$ for all $i \in I_0(c,a) \setminus I(c,a)$ and to $E_F$ otherwise. There are $t_F$ or less controllable symbol errors in the sequence $c \otimes_I a$ which can be corrected by the error-correcting code since $t_F \leq \lfloor (d-1)/2 \rfloor$. Actually, if (3) holds, then the scheme can correct $t$ additional errors and $e$ erasures.

If the inner flash memory code allows for $W'$ block-writes before block-erasures, then each inner encoder, the cells that record the output of the encoder, and the inner decoder can be combined into a single recording device that allows for $W'$ writes of $Q$-ary elements. This device can be viewed as a “generalized” cell with $q' = W' + 1$ “generalized levels”, since $W'$ symbol writes are guaranteed starting from the zeroth level. Each one of these generalized cells is composed of $n'$ flash memory cells.
Example 2. In this example, we use a $[5, 3, 3]$ extended RS code over $GF(4)$ as the outer code. For the inner code, we use the WOM code presented in Table 1 of [30]. This code allows for two writes where in each write an arbitrary message from four possible messages can be recorded using $n' = 3$ two-level cells, i.e., $q = 2$. Each symbol in the Reed-Solomon code over $GF(4)$ can then be written using 3 two-level cells. Thus, the concatenated scheme can write a message from a set of $4^3 = 64$ messages on 15 two-level cells. We can view each inner encoder, the 3 two-level cells recording the output of the encoder, and the corresponding inner decoder as a generalized cell that records symbols in $GF(4)$ twice, i.e., $W' = 2$. Each generalized cell has three levels, i.e., $q' = 3$. Since the minimum Hamming distance of the code is three, we can choose $t_F = 1$ and thus allow for one controllable error.

In the above description of the concatenated scheme, we assumed that each $Q$-ary symbol is recorded using an inner flash code. This need not be the case as several symbols can be recorded using the same inner code. In particular, if $N$ is divisible by $S$, then the symbols $c_1, c_2, \ldots, c_N$ can be partitioned into $S$ subblocks and each subblock of $N/S$ symbols can be recorded using an inner flash code. Controllable errors are introduced by not recording at most $t_F$ symbols as specified above. Fig. 1 shows a block diagram of the concatenated scheme composed of $S$ subblocks. In this scheme, the outer code is an error-correcting code and the inner code is any traditional flash memory code.

In the decoding procedure, the inner decoders read the stored sequence from the cells and then the outer decoder corrects all errors in the stored sequence and outputs the original message.

III. Analyses and Simulations

In this section, we perform a worst-case analysis to determine the number of guaranteed successful block-writes, starting from the all-zero cell-state vector, before incurring a block-erasure. The result shows how this number increases with increasing the parameter $t_F$. Then, we determine the rate of a class of flash codes constructed based on our scheme and compare it with the rate of other classes of flash codes. Finally, we present simulation results comparing different variations of our scheme with the uncoded scheme.

A. Worst Case Analysis

Consider a $q$-ary flash-memory code $\mathcal{F}$ based on a $Q$-ary $(N, M, d)$ error-correcting code $\mathcal{C}$. Let $t_F \leq \lfloor (d - 1)/2 \rfloor$ be the selected maximum number of controllable errors allowed in the scheme. We
define $W(q, C, t_F)$ to be the number of guaranteed successful block-writes starting from the all-zero cell-state vector before the encoder outputs $E_F$, minimized over all recorded messages. The following result, whose proof is given in the appendix, gives an explicit expression for $W(q, C, t_F)$.

**Theorem 1.** Let $l$ be the maximum Hamming distance between a pair of codewords in $C$. Then, for $l > 2t_F$ and $q ≥ 2$,

$$W(q, C, t_F) = 2\left\lfloor \frac{(q-2)t_F}{l-2t_F} \right\rfloor + q - 1.$$ 

Theorem 1 shows that allowing controllable errors increases the capability of the flash memory to accommodate block-writes without incurring block-erasures. If controllable errors are not allowed, i.e., $t_F = 0$, then the flash memory is guaranteed to accommodate only $q-1$ successful block-writes before block-erasure. As an example, let $C$ be the binary $[9, 1, 9]$ repetition code. Then, $l = d = 9$. For $t_F = 4$ and $q = 4$, the number of guaranteed successful block-writes is 19 but only 3 if $t_F = 0$. Fig. 2 is a bar graph that shows the increase in levels after each block-write. The increases due to the $i$-th block-write are identified by the bar segments having the entry $i$. Notice that the levels increase evenly and this contrasts with the case where no coding is used in which the level of a single cell may increase with each write while the levels of all other cells are equal to zero.

From the theorem, we notice that for our scheme to improve upon the uncoded scheme in the worst case, $q$ should be greater than 2. If $q = 2$, i.e., the memory cells have two levels only, then we can employ a concatenated scheme that uses a flash code with two-level cells as an inner code. Each $Q$-ary symbol is written using an inner flash code with $n'$ cells. Here the number of subblocks, $S$, in Fig. 1 equals the length of the outer code $N$. If $W'$ is the number of guaranteed consecutive writes in the inner
flash code, then as mentioned in Subsection II-F, the concatenated scheme can be viewed as using the code $C$ to produce $Q$-ary symbols which are recorded on “generalized cells” with $q' = W' + 1$ levels. As shown in the appendix, replacing the cells of $q$ levels with generalized cells of $q'$ levels, it follows that, starting with the all-zero cell-state vector, the concatenated scheme accommodates $W(q', C, t_F)$ block-writes. (Actually, it may accommodate more depending on the encoding function of the inner flash code, see the proof in the appendix.)

In summary, Theorem 1 holds in case the flash coding scheme is as presented in Subsection II-B, II-C, or II-F. The error-correcting code, $C$, used is binary in the first case and $Q$-ary in the other two cases. In all cases, the number of controllable errors, $t_F$, and the maximum Hamming distance between a pair of codewords, $l$, are measured in code symbols, which are bits in Subsection II-B and $Q$-ary symbols in the other two subsections. The number $q$ in the theorem denotes the number of levels of cells in the first two cases and in the third case equals $W' + 1$, where $W'$ is the guaranteed number of successful writes using the inner flash code.

B. Rates of Flash Codes

For a flash memory code that encodes a total of $M_{tot}$ messages on $n$ cells in $W$ successive writes, the rate of the code is defined as $R = \log_2(M_{tot})/n$. This rate is upper bounded by the capacity $\text{Cap}_{q,W}$ of the memory which is given by [10], [15]

$$\text{Cap}_{q,W} = \log_2 \left( \frac{W + q - 1}{q - 1} \right).$$

(5)

This is the maximum total number of information bits stored in one cell during the $W$ successive block-writes. In order to achieve the capacity, the flash memory codes should be allowed to store different number of messages in each write. In any case, from (5), it follows that

$$\text{Cap}_{q,W} = \log_2 \prod_{i=0}^{q-2} \frac{(W + q - 1 - i)}{(q - 1)!}$$

$$= (q - 1) \log_2 W + \log_2 \prod_{i=0}^{q-2} \left( 1 + \frac{q - 1 - i}{W} \right) - \log_2((q - 1)!).$$

Hence, as a function of $W$, $\text{Cap}_{q,W}$ tends to grow logarithmically (rather than linearly which is the case if a cell can change from any given level to any higher or lower level) with the number of writes $W$. If we define the asymptotic normalized capacity, $\text{cap}_q$ to be the limit of $\text{Cap}_{q,W}/\log_2 W$ as $W$ goes to infinity, then

$$\text{cap}_q = q - 1.$$
Given a sequence of block-write flash memory codes that write on \( q \)-level cells, where the \( i \)-th code, for \( i = 1, 2, \ldots \), has rate \( R_i \) and allows for \( W_i \) successive block-writes, \( W_1 < W_2 < \cdots \), we define the asymptotic normalized rate of the codes to be

\[
r_q = \lim_{i \to \infty} \frac{R_i}{\log_2 W_i},
\]

provided that the limit exists. In particular, \( r_q \leq \text{cap}_q = q - 1 \).

In our construction of flash codes using a \( Q \)-ary \((N, M, d)\) error-correcting code, \( C \), \( M \) messages can be encoded in each write. In particular, \( M_{\text{tot}} = M^W \) and the rate of the code is

\[
R = \frac{W(q, C, t_F) \log_2 M}{n},
\]

where \( W(q, C, t_F) \) is specified in Theorem 1. If a concatenated flash coding scheme is used for which the inner flash code writes each \( Q \)-ary symbol produced by the outer code on \( n' \) cells, and \( W' \) successful writes are guaranteed by the inner flash code, then, from (8), the rate of the concatenated flash coding scheme is given by

\[
R = \frac{W(q', C, t_F) \log_2 M}{n'N},
\]

where \( q' = W' + 1 \). If \( C \) is an \([N, K, d]\) linear code over \( GF(2^s) \), then by substituting \( M = Q^K \) and \( Q = 2^s \) in (9), the rate of the concatenated code can be written as

\[
R = \frac{W(q', C, t_F)sK}{n'N}.
\]

To get concrete results, we consider, as an outer code, the \( Q \)-ary \([(Q^K - 1)/(Q - 1), K, Q^{K-1}] \) error-correcting code, \( C_K \), known as the simplex code [16], which is the dual of the \( Q \)-ary \([(Q^K - 1)/(Q - 1), (Q^K - 1)/(Q - 1) - 3, 3] \) Hamming code. All the non-zero codewords in \( C_K \) have weight \( Q^{K-1} \). For this code, the maximum Hamming distance between a pair of codewords, \( l \), is \( Q^{K-1} \), which is the same as the minimum Hamming distance of the code. With the choice of \( t_F = \frac{1}{2}Q^{K-1} - t - 1 \), where \( t \) is the number of symbol errors to be corrected in addition to the controllable errors, it follows from Theorem 1 that the number of guaranteed successful block-writes is

\[
W(q', C_K, t_F) = 2 \left\lfloor \frac{(q' - 2)(\frac{1}{2}Q^{K-1} - t - 1)}{2(t + 1)} \right\rfloor + (q' - 1).
\]
With $Q = 2^s$, it follows by combining (10) and (11) that the rate of the concatenated flash code is

$$R_K = \left( 2 \left( \frac{(q' - 2)(\frac{1}{2}Q^{K-1} - t - 1)}{2(t+1)} \right) + (q' - 1) \right) \frac{sK}{n'(Q^K - 1)/(Q - 1)}. \quad (12)$$

We notice that as $K \to \infty$, $(\log_2 W(q', C_K, t_F))/K \to \log_2 Q = s$ and the asymptotic normalized rate of the codes, defined in (7), can be readily obtained from (11) and (12) as

$$r_q = \frac{1 - 2^{-s} q' - 2}{2(t+1) n'}. \quad (13)$$

In [30], Rivest and Shamir gave an explicit construction of a sequence of WOM codes, called tabular codes, that can be used to write a message composed of $s$ bits $W'$ times on $n'$ two-level cells for any fixed $s$ such that the limit of the ratio $n'/W'$ tends to 1. Using these codes as inner flash memory codes in our concatenated scheme, and recalling that $q' = W' + 1$, it follows from (13) that the asymptotic normalized rate of our concatenated flash memory coding scheme for two-level cells is

$$r_2 = \frac{1 - 2^{-s} }{2(t+1)}, \quad (14)$$

which can be made arbitrarily close to $1/(2(t+1))$ by choosing $s$ sufficiently large. For $t = 0$, this ratio, $1/2$, is half the asymptotic normalized capacity from (6) with $q = 2$. However, to achieve this asymptotic normalized capacity, as mentioned earlier, the flash memory codes may have to encode different number of messages in each write, which is not practical in most applications.

Next, we compare our concatenated coding scheme with other block-write coding schemes given in the literature. In [7], WOM codes, with no error-correcting capability, are constructed that can write messages composed of $K \geq 4$ bits $2^{K-2} + 2$ times on $2^K - 1$ two-level cells. This result is improved in [12] by allowing the $K$ bits to be written $2^{K-2} + 2^{K-4} + 1$ times on the same number of two-level cells. The asymptotic normalized rates for these codes is $1/4$, which is half the value of our scheme in case $t = 0$ as $s$ is allowed to increase arbitrarily. For codes that can correct a single error, [36] gives two constructions, based on double error correcting and triple error correcting BCH codes, which have asymptotic normalized rates $1/15.42$ and $1/13.45$, respectively. More efficient single error correcting WOM codes are introduced in [34] with asymptotic normalized rate of $1/6.4$. For $t = 1$, our concatenated scheme has asymptotic normalized rate of $(1 - 2^{-s})/4$, which can be made arbitrarily close to $1/4$ by choosing $s$ sufficiently large.
C. Simulation Results

Theorem 1 considers only the worst case scenario for block-writes. Its conclusion may not hold for the average scenario. Therefore, we present simulation results to illustrate how the proposed scheme based on the widely applied RS codes, performs on average.

In simulations, we use an \([N, K, d]\) RS codes over \(\mathbb{GF}(2^s)\), where \(d = N - K + 1\) and \(s\) is the least positive integer not less than \(\log_2 N\). These codes can be efficiently decoded using the Berlekamp-Massey (BM) algorithm [2], [27]. The simulations compare the scheme using RS codes, as explained in Subsection II-C, and the uncoded scheme consisting of \(sN\) cells. The number of cells used in both schemes is \(sN\) and each cell has \(q\) levels. We define \(W_{av}\) to be the *average number of block-writes per block-erasure* and \(R_{av}\) to be the *average rate* of the flash coding scheme, i.e., \(R_{av} = W_{av}K/N\) for the scheme using the RS code and \(R_{av} = W_{av}\) for the uncoded scheme. The values of \(W_{av}\) are estimated based on simulations using randomly generated bits for re-writes. We also included the number of writes, \(W\), and the code rate \(R\) based on worst case scenarios as defined in Theorem 1 and (8).

For the coded scheme, we consider two variations: one denoted by CE, referring to controllable errors as explained in Subsection II-C with \(t_F = \lfloor (d - 1)/2 \rfloor = \lfloor (N - K)/2 \rfloor\), and the other denoted by CEr, referring to controllable erasures as explained in Subsection II-E with \(t_F = d - 1 = N - K\), in which the locations of the controllable errors are recorded. However, in our simulations, the locations are not stored in the simulated flash memory, rather we assume that the scheme supports an additional memory to store these locations and this memory is symmetric, i.e., its contents can increase or decrease unlike flash memories. Therefore, the results associated with CEr are intended to show the potential of turning controllable errors into controllable erasures under this assumption. Storing the locations in the flash memory itself is discussed in Subsection II-E. We also consider using the threshold method, denoted by CE* and CEr*, as explained in Subsection II-D. We do not present values for \(R\) and \(W\) in the worst case since Theorem 1 does not necessarily hold when the threshold method is used. It suffices to note that these values are at most equal to the corresponding values denoted by CE and CEr in the table.

Using the average code rate as a performance measure, we notice that the coded scheme outperforms the uncoded scheme and the improvement depends on both the error-correcting code used and the number of levels \(q\). We also notice that using thresholds causes a slight degradation in performance. Finally, we notice that, for the coded scheme, the average case scenario, measured in terms of \(W_{av}\) and \(R_{av}\), can be significantly different from the worst case scenario, measured in terms of \(W\) and \(R\). However, for the uncoded scheme, the difference between the two scenarios is much smaller. Actually, for large number
TABLE II
SIMULATION RESULTS SHOWING THE AVERAGE NUMBER OF BLOCK-WRITES AND THE AVERAGE CODE RATE FOR RS CODED AND UNCODED SCHEMES. CE DENOTES CONTROLLABLE ERRORS, CER DENOTES CONTROLLABLE ERASURES, AND * DENOTES THE THRESHOLD METHOD.

<table>
<thead>
<tr>
<th>Code</th>
<th>( q )</th>
<th>( W_{av} )</th>
<th>( R_{av} )</th>
<th>( W )</th>
<th>( R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncoded ( 8 \times 255 ) cells</td>
<td>10</td>
<td>9.26</td>
<td>9.26</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>RS ([255,223,33]) CE</td>
<td>10</td>
<td>12.52</td>
<td>10.95</td>
<td>9</td>
<td>7.87</td>
</tr>
<tr>
<td>RS ([255,223,33]) CER</td>
<td>10</td>
<td>14.68</td>
<td>12.84</td>
<td>11</td>
<td>9.62</td>
</tr>
<tr>
<td>Uncoded ( 10 \times 1023 ) cells</td>
<td>10</td>
<td>9.00</td>
<td>9.00</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>RS ([1023,781,243]) CE</td>
<td>10</td>
<td>14.00</td>
<td>10.69</td>
<td>11</td>
<td>8.40</td>
</tr>
<tr>
<td>RS ([1023,781,243]) CER</td>
<td>10</td>
<td>18.00</td>
<td>13.74</td>
<td>15</td>
<td>11.45</td>
</tr>
<tr>
<td>Uncoded ( 10 \times 1023 ) cells</td>
<td>20</td>
<td>21.49</td>
<td>21.49</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>RS ([1023,781,243]) CE</td>
<td>20</td>
<td>34.28</td>
<td>26.17</td>
<td>23</td>
<td>17.56</td>
</tr>
<tr>
<td>RS ([1023,781,243]) CER</td>
<td>20</td>
<td>43.04</td>
<td>32.86</td>
<td>35</td>
<td>26.72</td>
</tr>
<tr>
<td>RS ([1023,781,243]) CE*</td>
<td>20</td>
<td>34.00</td>
<td>25.96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS ([1023,781,243]) CER*</td>
<td>20</td>
<td>41.40</td>
<td>31.61</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

of cells and small number of levels \( q \), as shown in case the number of cells is \( 10230 \) and \( q = 10 \), the difference is not noticeable. A statistical analysis based on Markov chains that confirms the discrepancy between worst case scenario and average case scenario for coded schemes is presented in [6].

IV. DISCUSSION AND CONCLUSION

In this paper, we propose a novel flash coding scheme based on error correcting codes. The proposed scheme can simultaneously reduce the frequency of block-erasures and provide flexible error-correcting capability. It avoids increasing the levels of cells with high levels by introducing controllable errors that can be corrected using an error-correcting code. By explicitly specifying error-correcting codes and inner codes, we have shown that concatenated coding schemes can be constructed with higher rates than those of other classes of codes presented in the literature. However, the rates we obtained are lower than capacity. To achieve capacity, different numbers of messages should be allowed in different writes.

It is interesting to see if our scheme can be modified to accommodate different numbers of messages, probably by using codes of different sizes, and whether this can lead to rates closer to capacity.

We also discuss a simplification by setting thresholds to select controllable errors and investigate the concatenation of our scheme with other flash coding schemes. Moreover, the problem of over-programming (injecting more charge than desired) can be alleviated by error-correcting codes. When this happens, the error-correcting system can be used to treat the content of a corrupted cell as an error
or an erasure, and correct it.

An issue that has important practical implications is the computational complexity required for implementing the encoder and the decoder of the flash code. For our scheme, this complexity is essentially the complexity of the encoder and the decoder of the error-correcting code in addition to the complexity involved in identifying the controllable errors. In case of a concatenated coding scheme, we need also to add the complexity of the encoder and the decoder of the inner flash code. In practical applications, error-correcting codes with fast decoding algorithms are definitely preferable. However, we notice that since error-correcting schemes have to be incorporated to combat noise in any flash memory, then essentially the extra complexity is due to the computations needed to identify those cells corresponding to the controllable errors and the correction of these controllable errors in addition to the errors caused by noise.

APPENDIX

Proof of Theorem 1

First, we give the proof in case the code $C$ is binary as in Subsection II-B. The cases in which $C$ is $Q$-ary as in Subsection II-C or used as an outer code in a concatenated scheme as in Subsection II-F are considered later.

**Binary Case:** Let $c_1$ and $c_2$ be two codewords in the code $C$ that are at maximum Hamming distance, $l$, apart. The minimum number of block-writes leading to $E_F$ is achieved by a sequence of messages encoded using the code $C$ into the sequence of codewords $c_1, c_2, c_1, c_2, \ldots$. To simplify notation, we may assume using a simple transformation that does not affect the result that $c_2 = 0$, where 0 denotes the all-zero codeword. We simply denote $c_1$, a vector of weight $l$, by $c$. Hence, to determine the minimum number of block-writes leading to $E_F$, we consider writing the sequence $c, 0, c, 0, \ldots$. Let $a^{(i)} = (a_1^{(i)}, a_2^{(i)}, \ldots, a_n^{(i)})$ be the cell-state vector that records the $i$-th message for $i \geq 1$. For convenience, we define $a^{(0)}$ to be the all-zero vector. Let $a_{\text{max}}^{(i)} = \max\{a_1^{(i)}, a_2^{(i)}, \ldots, a_n^{(i)}\}$ for $i \geq 1$ and $i_{\text{a,min}} = \min\{i : a_j^{(i)} = a \text{ for some } j = 1, 2, \ldots, n\}$ for $a \geq 1$. Then, $W(q, C, t_F) = i_{\text{q,min}} - 1$ (where, to simplify notation, we assume that the cell-state vector $a^{(i_{\text{q,min}})}$ is formed even though it has at least one entry equal to $q$). In the following, we determine $i_{\text{a,min}}$ using a recursion process.

Let $c = (c_1, c_2, \ldots, c_n)$ and $\mathcal{J} = \{j : c_j \neq 0\}$. For $i \geq 0$, define the sequence $f^{(i)} = f_0^{(i)}, f_1^{(i)}, \ldots$, where $f_a^{(i)}$ for $a \geq 0$ is the number of values of $j \in \mathcal{J}$ such that $a_j^{(i)} = a$, i.e., the number of cells whose levels equal $a$ after recording the $i$-th message. Notice that the components of $f^{(i)}$ sum to $|\mathcal{J}| = l$. 
Furthermore, the number of values of \( j \in \mathcal{J} \) such that \( a_j^{(i)} \) is odd gives the Hamming distance between \((a^{(i)})_2\) and 0, which is the number of controllable errors in case \( i \) is even. Similarly, the number of values of \( j \in \mathcal{J} \) such that \( a_j^{(i)} \) is even gives the Hamming distance between \((a^{(i)})_2\) and \( c \), which is the number of controllable errors in case \( i \) is odd. Hence, for even \( i \geq 2 \), the sum of \( f_a^{(i)} \) over odd \( a \) should be \( t_F \) and, for odd \( i \geq 1 \), the sum of \( f_a^{(i)} \) over even \( a \) should be also \( t_F \). Starting with \( f^{(0)} = l, 0, \ldots \), we have \( f^{(1)} = t_F, l - t_F, 0, \ldots \). For even \( i \geq 2 \), to record the \( i \)-th message, which is encoded as 0 in \( \mathcal{C} \), \( l - 2t_F \) cells of smallest odd levels and whose indices belong to \( \mathcal{J} \) increase their levels by 1. Similarly, for odd \( i \geq 3 \), to record the \( i \)-th message, which is encoded as \( c \), \( l - 2t_F \) cells of smallest even levels and whose indices belong to \( \mathcal{J} \) increase their levels by 1.

Based on these observations, the sequences \( f^{(i)} \), for \( i \geq 2 \) are uniquely determined. For example, \( f^{(2)} = t_F, t_F, l - 2t_F, 0, \ldots \). It also follows that the non-zero entries in \( f^{(i)} \) for \( i \geq 1 \), are confined to at most three consecutive entries, i.e., \( f^{(i)} = 0, \ldots, 0, f^{(i)}_{\max - 2}, f^{(i)}_{\max - 1}, f^{(i)}_\max, 0, \ldots \), where either \( f^{(i)}_{\max - 2} + f^{(i)}_{\max - 1} = l - t_F \) and \( f^{(i)}_\max = t_F \), in which case we say that \( f^{(i)} \) is of type I, or \( f^{(i)}_{\max - 2} + f^{(i)}_{\max} = t_F \) and \( f^{(i)}_{\max - 1} = l - t_F \) in which case we say that \( f^{(i)} \) is of type II. Suppose \( i \geq 2 \) and \( \max \) have the same parity, i.e., both are even or both are odd, and \( f^{(i)} \) is of type I. If \( f^{(i)}_{\max - 2} \geq l - 2t_F \), then

\[
\begin{align*}
f^{(i+1)}_{\max - 2} &= f^{(i)}_{\max - 2} - (l - 2t_F), \\
f^{(i+1)}_{\max - 1} &= f^{(i)}_{\max - 1} + (l - 2t_F) = l - t_F, \\
f^{(i+1)}_{\max} &= f^{(i)}_{\max}, \\
f^{(i+2)}_{\max - 2} &= f^{(i+1)}_{\max - 2} = f^{(i)}_{\max - 2} - (l - 2t_F), \\
f^{(i+2)}_{\max - 1} &= f^{(i+1)}_{\max - 1} - (l - 2t_F) = t_F, \\
f^{(i+2)}_{\max} &= f^{(i+1)}_{\max} + (l - 2t_F) = f^{(i)}_{\max} + (l - 2t_F).
\end{align*}
\]

In this case, \( a^{(i+2)}_{\max} = a^{(i)}_{\max} \), both \( i + 2 \) and \( a^{(i+2)}_{\max} \) have the same parity, and \( f^{(i+2)} \) is of type I. On the other hand, if \( f^{(i)}_{a^{(i)}_{\max - 2}} < l - 2t_F \), then

\[
\begin{align*}
f^{(i+1)}_{\max - 2} &= 0, \\
f^{(i+1)}_{\max - 1} &= t_F + f^{(i)}_{a^{(i)}_{\max - 2}}, \\
f^{(i+1)}_{\max} &= f^{(i)}_{\max} + f^{(i)}_{a^{(i)}_{\max - 2}} - (l - 2t_F) = t_F, \\
f^{(i+1)}_{\max + 1} &= (l - 2t_F) - f^{(i)}_{a^{(i)}_{\max - 2}} > 0.
\end{align*}
\]

In particular, \( a^{(i+1)}_{\max} = a^{(i)}_{\max} + 1 \), both \( i + 1 \) and \( a^{(i+1)}_{\max} \) have the same parity and \( f^{(i+1)} \) is of type I. We conclude that if \( i \) and \( a^{(i)}_{\max} \) have the same parity, \( f^{(i)}_{a^{(i)}_{\max - 2}} \geq l - 2t_F \), and \( f^{(i)} \) is of type I, then
two block-writes can be accommodated without raising the maximum level of cells, i.e., \( a_{\text{max}}^{(i+2)} = a_{\text{max}}^{(i)} \).

Furthermore, after these two block-writes, \( i + 2 \) and \( a_{\text{max}}^{(i+2)} \) have the same parity, \( f^{(i+2)} \) is also of type I, and \( f_{a_{\text{max}}^{(i+2)}-2}^{(i+2)} = f_{a_{\text{max}}^{(i)}-2}^{(i)} - (l - 2t_F) \). However, if \( i \) and \( a_{\text{max}}^{(i)} \) have the same parity, \( f_{a_{\text{max}}^{(i)}-2}^{(i)} < l - 2t_F \), and \( f^{(i)} \) is of type I, then the next block-write raises the maximum level of cells by 1. After this block-write, \( i + 1 \) and \( a_{\text{max}}^{(i+1)} \) have the same parity, \( f^{(i+1)} \) is also of type I, and \( f_{a_{\text{max}}^{(i+1)}-2}^{(i+1)} = t_F + f_{a_{\text{max}}^{(i)}-2}^{(i)} \). Hence, \( 2F_{a_{\text{max}}^{(i)}-2}^{t_F}/(l - 2t_F) \) is the maximum number of block-writes following the \( i \)-th block-write such that no cell has level greater than \( a_{\text{max}}^{(i)} \). Based on the definition of \( i_{a,\text{min}} \), it follows that \( a \) and \( i_{a,\text{min}} \) have the same parity and that \( f^{(i_{a,\text{min}})} \) is of type I for \( a \geq 2 \). Let \( f_a^{*} = f_{a_{\text{min}}-2}^{i_{a,\text{min}}} \). We have

\[
i_{a+1,\text{min}} - i_{a,\text{min}} = 2 \left\lfloor \frac{f_a^*}{l - 2t_F} \right\rfloor + 1
\]

and

\[
f_{a+1}^* = t_F + f_a^* - \left\lfloor \frac{f_a^*}{l - 2t_F} \right\rfloor (l - 2t_F).
\]

With \( f_{2}^* = t_F \), the second recursion gives

\[
f_a^* = (a - 1)t_F - \left\lfloor \frac{(a - 2)t_F}{l - 2t_F} \right\rfloor (l - 2t_F).
\]

Substituting in the first recursion, we get

\[
i_{a+1,\text{min}} - i_{a,\text{min}} = 2 \left\lfloor \frac{(a - 1)t_F}{l - 2t_F} \right\rfloor - 2 \left\lfloor \frac{(a - 2)t_F}{l - 2t_F} \right\rfloor + 1.
\]

With \( i_{2,\text{min}} = 2 \), this recursion gives

\[
i_{a,\text{min}} = 2 \left\lfloor \frac{(a - 2)t_F}{l - 2t_F} \right\rfloor + a.
\]

We conclude that \( W(q, C, t_F) = i_{q,\text{min}} - 1 \) is as given in the statement of the theorem.

**Q-ary Case:** In the above proof, we assumed the code \( C \) to be binary. The same proof holds if \( C \) is a \( Q \)-ary code, where \( Q = 2^s \). In this case, \( t_F \) denotes the number of controllable symbol, rather than bit, errors and \( l \) denotes the maximum weight, in symbols, of a codeword in \( C \). We also need to redefine \( a_j^{(i)} \) to be the maximum level of the \( s \) cells representing the \( j \)-th symbol in the \( i \)-th message.

**Concatenation Case:** The same proof holds in case of a concatenated scheme where \( a_j^{(i)} \) denotes the generalized level of a generalized cell which is assigned a numerical value in the set \( \{0, 1, \ldots, q - 1\} \). Starting with the generalized level \( a_j^{(0)} = 0 \) denoting the symbol 0, for the \( i \)-th write \( a_j^{(i)} \), \( j \in J \), is either increased by 1 or remains the same if the \( j \)-th generalized cell is selected for a controllable error. In particular, the generalized level is even if it represents the symbol 0 and odd if it represents the symbol \( c_j \).
However, it should be noted that the expression stated in Theorem 1 in case of a concatenation scheme is a lower bound on the number of guaranteed successful block-writes since the inner flash code guarantees \( q - 1 \) consecutive writes in the worst case. Depending on the encoding function of the inner flash code, it may be able to accommodate more alternating writes of the symbols \( c_i \) and 0 for each \( i = 1, 2, \ldots, N \).

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their helpful comments.

REFERENCES


