Rewritable Coset Coding for Flash Memories

Yeow Meng Chee*, Han Mao Kiah†, and Punarbasu Purkayastha*
*School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
†Coordinated Science Lab, University of Illinois at Urbana-Champaign, USA
Emails: ymchee@ntu.edu.sg, hmkiah@illinois.edu, punarbasu@ntu.edu.sg

***Abstract*—Flash memory is a nonvolatile memory technology that suffers from errors due to charge leakage, can tolerate limited erasures, and where erasures have to be performed in large blocks. We show that using cosets of a linear code can provide correction against uniform charge leakage, and can enhance the rewritability of flash memory which leads to fewer erasures. We introduce two coset coding schemes that are generalizations of the scheme in Jacobvitz *et al.* (2013). For the same worst case rewrite cost, we show that coset codes can encode more information than rank modulation codes. The average case performance of coset codes is demonstrated via numerical simulations.**

1. Introduction

Flash memory is a nonvolatile memory technology that has become a dominant medium of storage over the past decade in both consumer and enterprise applications. In this memory technology charge is injected iteratively into a cell to bring the charge to a desired level, and the level of the charge encodes the bits that are to be recorded. Multilevel flash memory is used to increase the density of information stored, and also to improve the speed of reading and writing data to the memory. Despite being a fast medium, multilevel flash memory technology suffers from a couple of deficiencies that we describe below briefly (see [1] for details).

(i) Resetting the charge level of a cell to the lowest level corresponds to an *erase* operation. This erasure operation can not be performed on individual cells, and instead has to be performed on a block of about a million cells. Therefore, it is a slow operation. Additionally, the number of erasure operations that can be performed is limited to about 10^5 erasures in the lifetime of the device.

(ii) Aging effects in flash memory may result in a uniform drift of charge levels, which can lead to programming and read errors because of the shift in threshold levels.

(iii) Random errors can occur during reading or writing because of the device characteristics.

(iv) Overshooting problem can arise in writing to a multilevel flash memory, where the final programmed charge level exceeds the desired charge level, if the process of charge injection is not carefully controlled.

To overcome these limitations of flash memory, many different encoding techniques have been proposed over the past several decades. In particular, the problem of erasure is handled by modeling the flash memory as a write once memory (WOM) if each cell can represent only a single bit, and a write asymmetric memory (WAM) if each cell can represent more than one bit of information. There is long precedent for codes that have been studied for WOM and WAM type memories, starting with the work of Rivest and Shamir [2], Cohen *et al.* [3], and the more recent works by Jiang *et al.* [4], Yaakobi *et al.* [5], and Jacobvitz *et al.* [6], [7]. The objective of all these works is to maximize the number of rewrites that can be made; or equivalently maximize the amount of information that can be written if the number of times rewrites that can be performed is bounded.

The problem of uniform charge leakage due to aging can be addressed by using error correcting codes. In particular, the study of rank modulation codes was initiated in Jiang *et al.* [8], [9], and error scrubbing codes were studied in Jiang *et al.* [10] to address this problem. The problem of large nonuniform charge leakage was studied in Farnoud *et al.* [11].

Relatively fewer works are present which study codes that can correct both random errors and also ensure rewritability. We note a couple of works in this direction. The work of Cohen *et al.* [3] studied binary error correcting codes for WOM, Yaakobi *et al.* [5] studied WOM codes and their generalizations to WAM, Jiang *et al.* [4] uses nested polar codes, Haymaker [12] uses geometric constructions for WOM, Kurkoski [13] studies codes arising from lattice structures, and Jacobvitz *et al.* [6], [7] uses binary coset codes for error correction and rewrites. A specific construction by Jiang *et al.* [8] also uses rank modulation codes to address the problem of uniform charge leakage, errors due to overshooting, and rewritability.

In this work, we use cosets of linear codes for handling uniform charge leakage, for ensuring rewritability, and for error correction in flash memory. We do not address the problem of overshooting in this work. We assume that the process of careful charge injection can mitigate the overshooting problem. We assume that the flash memory has discrete charge levels. This is a reasonable assumption since the charge is injected in discrete quantities and the detection of different charge levels requires a minimum separation between consecutive charge levels. Our initial construction uses cosets formed from the subspace generated by the all-one vector to capture the phenomenon of uniform charge leakage due to aging. Further errors due to random charge leakage, or programming errors in individual cells can be corrected by the linear code. The second coding scheme builds up on this construction by dividing the total number of levels into parts of size q each, and optimizing the charge level of each cell individually. This scheme can be considered as a generalization of the method in Jacobvitz *et al.* [7] to q-ary codes. As noted in [7], the use of cosets implies that the same information can be represented by a set of codewords. Hence, we can optimize over this set of codewords so that the "cost" of a rewrite is minimized. On the other hand, if we choose the coset from a linear error-correcting code, the error-correcting capability of the coset code follows from that of the linear code. Both these constructions differs from WOM and WAM codes studied in [4], [5] in that the number of levels is not restricted to the alphabet size of the code. Since our code is designed to correct uniform charge leakage, we compare the average number of rewrites with the average number of rewrites in rank modulation. It is observed that using coset codes results in larger number of rewrites on average, as compared to rank modulation codes.

The rest of the paper is organized as follows. The next section introduces some basic notations and definitions. Section 3 gives the two coset coding schemes. Section 4 compares the two constructions from Section 3. In this section we also compare the properties of the code with that of a rank modulation code. To compare the average number of rewrites we simulate the performance of the rank modulation code and the coset codes.

2. Preliminaries and Notations

Throughout this paper, we let L, n, k and q denote positive integers. In particular, q is assumed to be a prime or a power of a prime. The set $\{1,2,\ldots,n\}$ is denoted by $[n]$ while the integers modulo q and the finite field of order q are denoted by $\mathbb{Z}_q$ and $\mathbb{F}_q$ respectively. In this paper, we sometimes map integers to elements in $\mathbb{F}_q$. When q is prime, taking integers modulo q clearly suffices. When q is not prime, we can consider any bijective mapping $\phi:\mathbb{Z}_q\to\mathbb{F}_q$ and abuse our notation by writing $s \bmod q$ to mean $\phi(s \bmod q)$ for all integers s. Extend this to vectors $\mathsf{s}\in\mathbb{Z}^n$ and we have $\mathsf{s} \bmod q \triangleq (\mathsf{s}_i \bmod q)_{i\in[n]}$ to belong to $\mathbb{F}_q^n$. We denote the span of vectors $\mathsf{v}_1,\ldots,\mathsf{v}_M$ by the notation $\langle \mathsf{v}_1,\ldots,\mathsf{v}_M\rangle$.

We consider the *Write Asymmetric Memory* (WAM) model for storage where we consider that the charge levels can only increase. The WAM consists of a block of n cells, where each cell has L discrete levels, viz. states $0,\ldots,L-1$. These levels in a cell may correspond to the charge levels that can be distinguished. In particular, we assume that two consecutive charge levels l,l' in a cell are separated by a *safety margin*, say Δ such that if $|l-l'|>\Delta$, then we can distinguish between the two levels (see [6], [14]). A *state vector* is an element in $\{0,\ldots,L-1\}^n$. We define a *partial ordering* on $\{0,\ldots,L-1\}^n$ via the relation $\mathsf{s}\le\mathsf{s}'$, for $\mathsf{s},\mathsf{s}'\in\{0,\ldots,L-1\}^n$, if $\mathsf{s}_j\le\mathsf{s}'_j$ for all $j\in[n]$. Hence, a transition from s to s' is *valid* if and only if $\mathsf{s}\le\mathsf{s}'$.

Let C be a finite set of messages and let $S\subseteq\{0,\ldots,L-1\}^n$ be a set of *encoded states*. The quadruple (C,S,α,β) is a *coding scheme* for WAM if for all $\mathsf{c}'\in C$ and $\mathsf{s}\in S$, the following hold.

(i) $\alpha: S\times C\to S$ is an *encoding* function such that $\alpha(\mathsf{s},\mathsf{c}')\ge\mathsf{s}$,
(ii) $\beta: S\to C$ is a *decoding* function, where $\beta(\alpha(\mathsf{s},\mathsf{c}'))=\mathsf{c}'$.

In other words, given the current state s, the function α encodes a new codeword c' to a state s' such that the transition from s to s' is valid. On the hand, the function β decodes a state vector s' back to its original codeword c'.

Suppose we transition from state s to state s'. The *cost of rewrite* $\gamma(\mathsf{s}\to\mathsf{s}')$ is defined as the difference of the maximum of the two states, i.e.,

$$\gamma(\mathsf{s}\to\mathsf{s}')\triangleq\max_{i\in[n]}\mathsf{s}'_i-\max_{j\in[n]}\mathsf{s}_j.$$

This notion of the cost is used in [8], [9], [14], and as we discuss below, it provides an analysis of the work in [6], [7].

3. Coset Coding for WAM

In this section we describe our coset coding scheme for WAM that enables us to provide error correction, rewritability, and address the problem of uniform charge leakage in flash memory. To compare our schemes with existing schemes we briefly discuss the schemes in Jacobvitz *et al.* [6], [7] and Jiang *et al.* [8]. Consider codes of block length n. We formally define the encoding function, provided the decoding function is $\beta(\mathsf{s})$ for s in the set of encoded states S. Assume that the current state is s and that we want to add a codeword c'. Then the encoding function $\alpha(\mathsf{s},\mathsf{c}')$ outputs a state s' with the minimum possible cost $\gamma(\mathsf{s}\to\mathsf{s}')$, and minimum cell changes.

$$\begin{aligned}\mathcal{A}(\mathsf{s},\mathsf{c}')&=\operatorname*{arg\,min}_{\mathsf{s}'\in S}\left(\max_{i\in[n]}\mathsf{s}'_i\right)\quad\text{s.t. }\beta(\mathsf{s}')=\mathsf{c}',\mathsf{s}'\ge\mathsf{s},\\ \alpha(\mathsf{s},\mathsf{c}')&\in\operatorname*{arg\,min}_{\mathsf{s}'\in\mathcal{A}(\mathsf{s},\mathsf{c}')}\sum_{i\in[n]}\mathsf{s}'_i-\mathsf{s}_i.\end{aligned}\tag{1}$$

The problems of rewritability and error correction was addressed in [6], [7] by using *binary* linear coset codes to represent the data. We rephrase their construction in our own words. One possible realization of their scheme, called **FlipMin**, is by considering a linear code $\mathcal{C}\subset\mathbb{F}_2^n$ containing the all-one vector j, and by defining

$$\begin{aligned}\beta(\mathsf{s})&=\mathsf{s}\bmod 2+\mathcal{D},\\ \alpha(\mathsf{s},\mathsf{c}')&\in\operatorname*{arg\,min}_{\mathsf{s}'\in S}|\{\mathsf{s}'_i\ne\mathsf{s}_i:i\in[n]\}|\quad\text{s.t. }\beta(\mathsf{s}')=\mathsf{c}',\end{aligned}$$

where $\mathcal{D}$ is a subspace of $\mathcal{C}$ containing j. Larger subspace $\mathcal{D}$ potentially increases the average number of rewrites. Given a new codeword c' that is a representative of the *coset* $\mathsf{c}'+\mathcal{D}$, the encoded word is determined by the following steps. Let c be the previously written codeword. First, we determine the *translate set* $\mathcal{T}=\{\mathsf{c}+y:y\in\mathsf{c}'+\mathcal{D}\}$. Next, the translate that is used to add the coset representative c' into memory is determined by picking any coset leader (of minimum weight) in the translate set $\mathcal{T}$. Hence, in this scheme, the set of possible codewords is given by the set of the coset representatives, or equivalently, the *quotient space* $\mathcal{C}/\mathcal{D}$.

Jacobvitz *et al.* showed that this procedure minimizes the number of bit flips that occur in writing [7]. In the following example, we illustrate the difference with the minimization objective defined in (1).

Example 3.1. Let the current state be $\mathsf{s}=(2,3,3,2)$ and the codeword corresponding to it is $\mathsf{c}=0110$. The cosets

$$S_1=\{0000,0101,1010,1111\},S_2=\{0001,0100,1011,1110\},$$
$$S_3=\{0010,0111,1000,1101\},S_4=\{0011,0110,1001,1100\}.$$

partition the linear space $\mathbb{F}_2^4$ into four equal parts. In other words, the sets $\{S_1, S_2, S_3, S_4\}$ form the quotient space $\mathbb{F}_2^4/S_1$. Suppose that the new codeword belongs to S_2. Then the translate set is $\mathsf{c} + S_2 = \{0111, 0010, 1101, 1000\}$. There are two representatives of minimum weight, namely, $0010, 1000$, both of which minimize the number of flipped bits. However, as illustrated by Fig. 1, using 0010 increases the maximum charge level by one to the new state $\mathsf{s}' = (2, 3, 4, 2)$, or to $\mathsf{s}' = (4, 3, 4, 4)$. On the other hand, using the translate coset leader 1000 implies that the maximum charge level stays the same to give the new level $\mathsf{s}' = (3, 3, 3, 2)$.

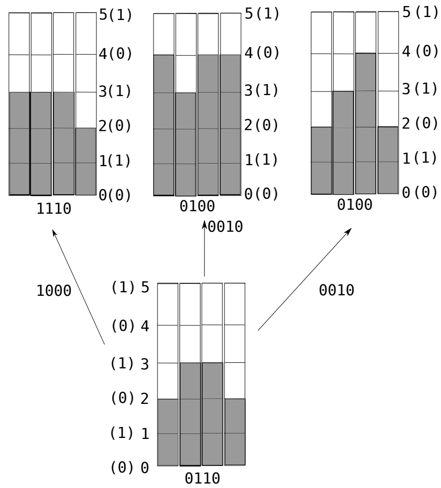

Fig. 1. Levels when encoding using (1) in contrast to minimum bit flip

In *Scheme A*, described below, we generalize the method to q-ary linear coset codes that are encoded using (1).

Scheme A: Let j be the all-one vector of length n. Consider a linear code $\mathcal{C}[n, k+1, d]$ of length n, dimension $k+1$ and mimimum distance d in $\mathbb{F}_q^n$, containing the all-one vector j. We encode vectors from the coset code $C = \mathcal{C}/\langle \mathsf{j} \rangle$ of size q^k by using (1). Let $s_{\min} = \min_{i \in [n]} \mathsf{s}_i$. Then the decoding function is given by

$$\beta(\mathsf{s}) = \phi\big((\mathsf{s}_i - s_{\min})_{i \in [n]}\big) + \langle \mathsf{j} \rangle.$$

Many well-known families of linear codes contain j, including the primitive narrow-sense Bose-Chaudhuri-Hocquenghem codes, the extended Golay code, the Reed-Müller codes, and the Reed-Solomon codes [15]. Uniform charge leakage in flash memory translates all the elements of the coset by a constant value and hence the codeword that is encoded is unchanged, and efficient decoding and error correction is immediate because of the use of linear codes. The vector space $\langle \mathsf{j} \rangle$ contains only q elements and so the complexity of determining the coset representative grows as $\Theta(n)$ for a fixed q. Efficient decoding and error correction is immediate because of the use of linear codes. One may also use a larger coset $\mathcal{D}$ containing j, at the cost of increase in encoding complexity.

The rewritability of the scheme can be determined from the worst-case and average case analysis of the encoding operation. It can be readily seen that the cost of increase is bounded by $q-1$, i.e., $\gamma(\mathsf{s} \to \mathsf{s}') \leq q - 1$. This determines the least number of times we can rewrite a single cell.

Lemma 3.1. Using Scheme A, the number of rewrites is lower bounded by $\left\lfloor \frac{L-1}{q-1} \right\rfloor$.

Proof: The number of levels used on first write is q; subsequent writes use only $q-1$ additional levels at the most. Therefore, the total number of times writes can be performed is at least $\left\lfloor \frac{L-1}{q-1} \right\rfloor$. ■

We determine an upper bound to the average cost that can be numerically computed for small alphabet sizes. To determine this, we first establish a sequence of lemmas. Define

$$\Phi(\mathsf{c}) \triangleq \{\phi^{-1}(\mathsf{c}_i) : i \in [n]\}.$$

Lemma 3.2. Suppose we add a vector c' that has the smallest cost out of all the possible vectors $\mathsf{c}' + \langle \mathsf{j} \rangle$, and the previous state was s. Let 0_n be the all-zero vector. Then,

$$\gamma(\mathsf{s} \to \alpha(\mathsf{s}, c')) \leq \gamma(0_n \to \alpha(0_n, \mathsf{c}')) = \max\{i : i \in \Phi(\mathsf{c}')\}.$$

Proof: If $\mathsf{s} = 0_n$, then we need at least $\max \Phi(\mathsf{c}')$ levels to distinguish between the different alphabets in c'. This determines the equality. If $\mathsf{s} \neq 0_n$, increase all the levels to $\max\{s_i : i \in [n]\}$ and use $\max \Phi(\mathsf{c}')$ levels to distinguish between the alphabets in c'. This establishes the inequality. ■

Define the average cost as the average over all possible states s and cosets $\mathsf{c}' + \langle \mathsf{j} \rangle$. By Lemma 3.2, this is upper bounded by the average over all possible cosets $\mathsf{c}' + \langle \mathsf{j} \rangle$, i.e.,

$$\frac{1}{L^n q^{n-1}} \sum_{\mathsf{s}, \mathsf{c}'} \gamma(\mathsf{s} \to \alpha(\mathsf{s}, \mathsf{c}')) \leq \frac{1}{q^{n-1}} \sum_{\mathsf{c}' \in \mathbb{F}_q^n / \langle \mathsf{j} \rangle} \gamma(0_n \to \alpha(0_n, \mathsf{c}')).$$

The cost $\gamma(0_n \to \alpha(0_n, \mathsf{c}'))$ is determined by the alphabets that occur in $\widetilde{\mathsf{c}}' \in \mathsf{c}' + \mathcal{D}$, as described in the lemma below.

Lemma 3.3. If $\gamma(0_n \to \alpha(0_n, \mathsf{c}')) = q - 1 - \ell$, then a vector $\widetilde{\mathsf{c}}' \in \mathsf{c}' + \langle \mathsf{j} \rangle$ with minimum cost satisfies

$$\Phi(\widetilde{\mathsf{c}}') = \{i_0 = 0, i_1, i_2, \ldots, i_p, i_{p+1} = q - 1 - \ell\}, \qquad (2)$$

where $0 < i_j - i_{j-1} \leq \ell + 1$, $\forall j = 1, \ldots, p+1$.

Conversely, if $\mathsf{c}' \in \mathbb{F}_q^n$ satisfies (2) then the minimum cost is $\gamma(0_n \to \alpha(0_n, \mathsf{c}')) = q - 1 - \ell$.

Proof: To prove the forward direction, suppose that $\widetilde{\mathsf{c}}'$ is the codeword with the minimum cost. First we show that 0 and $q - 1 - \ell$ must occur in $\widetilde{\mathsf{c}}'$. If the cost is $q - 1 - \ell$ then the alphabet elements $q - \ell, q - \ell + 1, \ldots, q - 1$ are not present in $\widetilde{\mathsf{c}}'$. The vector $\widetilde{\mathsf{c}}'$ must contain $q - 1 - \ell$; otherwise by Lemma 3.2, the cost is strictly less than $q - 1 - \ell$. Similarly, if 0 does not occur in $\widetilde{\mathsf{c}}'$ then the word $\widetilde{\mathsf{c}}' - \mathsf{j}$ has a cost strictly less than $q - 1 - \ell$, which contradicts the fact that $\widetilde{\mathsf{c}}'$ was the word with the lowest cost.

To complete the proof of the forward part, we now show that the difference $i_j - i_{j-1}$ is at most $\ell + 1$. Suppose not. Then, there exists an element i such that i is present in $\widetilde{\mathsf{c}}'$ and the elements $i+1, i+2, \ldots, i+\ell+1$ are not present in $\widetilde{\mathsf{c}}'$. Then the vector $\widetilde{\mathsf{c}}' + (q-i-\ell-2)\mathsf{j}$ does not have the elements $q-\ell-1, q-\ell, \ldots, q-1$ in it. Therefore, this word has a cost strictly less than that of $\widetilde{\mathsf{c}}'$. This is a contradiction.

To prove the converse, suppose c' satisfies (2). Then the cost of writing c' is $q-1-\ell$. Since the difference $i_j - i_{j-1}$ is at most $\ell+1$, no shift of c' by a multiple of j will have more than ℓ elements $q-\ell, \ldots, q-1$ absent from the resulting word. ∎

Given $\ell \in \{0, \ldots, q-1\}$, we next determine the distribution of the alphabet elements in any vector c' which satisfy the above lemma. The answer stems from the count of vectors which are *run-length limited* [16]. Let $\mathcal{N}_\ell(m)$ be the set of ℓ-sequences of length m which do not have more than ℓ zeros between two consecutive ones, and let $N_\ell(m) = |\mathcal{N}_\ell(m)|$ Then, $N_\ell(m)$ satisfies a generalized Fibonacci sequence,

$$N_\ell(m) = \begin{cases} 2^m, & \text{for } 0 < m \leq \ell, \\ \sum_{i=1}^{\ell+1} N_\ell(m-i), & \text{for } \ell < m, \end{cases} \tag{3}$$

and an explicit construction can also be determined from a recursive construction of the so-called *cross-bifix-free codes* (see [16], [17]). Consider indicator vectors $v = (v_0, \ldots, v_{q-1})$ of length q where $v_0 = 1 = v_{q-1-\ell}$, and $v_i = 0, i = q-\ell, \ldots, q-1$, and $(v_1, \ldots, v_{q-2-\ell})$ is an ℓ-sequence. Then, $\mathcal{N}_\ell(q-2-\ell)$ is the set of such ℓ-sequences, with the cardinality given by (3). We first illustrate the lemmas by an example.

Example 3.2. Consider the vector $\mathsf{u} = 1147 \in \mathbb{Z}_8^4$. We get $\Phi(\mathsf{u}) = \{1, 4, 7\}$, and so it requires $\max \Phi(\mathsf{u}) = 7$ levels when writing. The coset $\mathsf{u} + \langle \mathsf{j} \rangle$ contains $\mathsf{u}' = 2250$ and $\mathsf{u}'' = 5503$. The vectors u' and u'' require only five levels since $\max \Phi(\mathsf{u}') = 5 = \max \Phi(\mathsf{u}'')$. It can be verified that this is the minimum possible cost of rewrite. The indicator vectors of the alphabets that occur in $\mathsf{u}, \mathsf{u}', \mathsf{u}''$ are respectively, $01001001, 10100100, 10010100$. Here, $\ell = 2$.

Using the count of the number of vectors whose alphabet elements satisfy that their indicator vectors are ℓ-sequences in the first $q-1-\ell$ coordinates (with the first and last coordinates fixed to 1), we can derive an upper bound on the average cost. This is shown by the next Proposition. For a fixed vector, let n_j denote the number of times the alphabet element j occurs in the vector.

Proposition 3.1. Let $\mathbf{n}_\ell = (n_1, \ldots, n_{q-2-\ell})$, $1(x)$ be an indicator function, and $\mathbf{1}(\mathbf{n}_\ell) = (1(n_i > 0))_{i=1,\ldots,q-2-\ell}$. Let

$$B(\ell) \triangleq \sum_{\substack{(n_0, \mathbf{n}_\ell, n_{q-1-\ell}) \\ n_0, n_{q-1-\ell} > 0, \, \mathbf{1}(\mathbf{n}_\ell) \in \mathcal{N}_\ell(q-2-\ell)}} \binom{n}{n_0, \mathbf{n}_\ell, n_{q-1-\ell}}.$$

Then, for $\ell \geq \left\lfloor \frac{q}{2} \right\rfloor$, $B(\ell) = (q-\ell)^n - 2(q-\ell-1)^n + (q-\ell-2)^n$, and

$$\frac{1}{q^{n-1}} \sum_{\mathsf{c}'} \gamma(0_n \to \alpha(0_n, \mathsf{c}')) \leq \frac{(q-1)B(0)}{q^{n-1}q} + \sum_{\ell=1}^{q-2} \frac{q-1-\ell}{q^{n-1}} B(\ell).$$

Proof: The multinomial coefficient $\binom{n}{n_0, \mathbf{n}_\ell, n_{q-\ell-1}}$ gives the total number of vectors $\widetilde{\mathsf{c}}'$ of length n which necessarily have the alphabet elements $0, q-\ell-1$ and such that the other alphabets are present in $\Phi(\widetilde{\mathsf{c}}')$ with gaps of at most ℓ between successive alphabet elements. The cost of writing such vectors to the state 0_n is $q-1-\ell$.

For $\ell = 0$, the quantity $B(0)$ counts the number of vectors which have all the alphabet elements in it, i.e.,

$$B(0) = \sum_{\substack{(n_0, \mathbf{n}_0, n_{q-1}), \\ n_j > 0, \forall j}} \binom{n}{n_0, \mathbf{n}_0, n_{q-1}}.$$

For each vector in this count, any shift by a multiple of j will still have all the alphabet elements occuring in it. Therefore, to count only the cosets, we use the quantity $\frac{1}{q}B(0)$. The cost of writing any vector in such a coset is exactly $q-1$.

For $q-2 \geq \ell \geq \lfloor q/2 \rfloor$, the number of such vectors can be exactly computed since all possible sequences $\mathbf{1}(\mathbf{n}_\ell) \in \{0,1\}^{q-2-\ell}$ are allowed. Thus, we get

$$\begin{aligned} &\sum_{\substack{(n_0, \mathbf{n}_\ell, n_{q-1-\ell}) \\ n_0, n_{q-1-\ell} > 0}} \binom{n}{n_0, \mathbf{n}_\ell, n_{q-1-\ell}} \\ &= \sum_{(n_0, \mathbf{n}_\ell, n_{q-1-\ell})} \binom{n}{n_0, \mathbf{n}_\ell, n_{q-1-\ell}} \\ &- \sum_{\substack{(\mathbf{n}_\ell, n_{q-1-\ell}) \\ n_{q-1-\ell} > 0}} \binom{n}{\mathbf{n}_\ell, n_{q-1-\ell}} - \sum_{\substack{(n_0, \mathbf{n}_\ell) \\ n_0 > 0}} \binom{n}{n_0, \mathbf{n}_\ell} - \sum_{\mathbf{n}_\ell} \binom{n}{\mathbf{n}_\ell} \\ &= (q-\ell)^n - 2(q-1-\ell)^n + (q-2-\ell)^n, \end{aligned}$$

where the last step is obtained by using

$$\sum_{n_0 > 0, \mathbf{n}_\ell} \binom{n}{n_0, \mathbf{n}_\ell} = \sum_{n_0, \mathbf{n}_\ell} \binom{n}{n_0, \mathbf{n}_\ell} - \sum_{\mathbf{n}_\ell} \binom{n}{\mathbf{n}_\ell}.$$

For $\ell = q-1$, the cost is zero.

Finally, for $\left\lfloor \frac{q}{2} \right\rfloor > \ell \geq 1$, there can be multiple vectors in the same coset with the same cost $q-1-\ell$. We simply use $B(\ell)$ as an upper bound since this count of the vectors with the same cost within the same coset could be anywhere between 1 and $q/(\ell+1)$. ∎

For $q = 3$, we get the following expression for the cost

$$\begin{aligned} &\frac{2}{3^{n-1}} \frac{1}{3} \sum_{n_0, n_1, n_2 > 0} \binom{n}{n_0, n_1, n_2} + \frac{1}{3^{n-1}}(2^n - 2) \\ &= \frac{1}{3^{n-1}} \left(\frac{2}{3} \cdot 3^n - 5(2^n - 2) - 6 \right). \end{aligned}$$

This is an upper bound to the cost for $q = 3$. Not surprisingly, for large n, the cost tends to 2.

Lemma 3.4. For sufficiently large n, the dominant term in the upper estimate of the average cost is $\frac{(q-1)B(0)}{q^n}$.

Proof: First we show that the term $f(\ell, n) \triangleq (q-\ell)^n - 2(q-1-\ell)^n + (q-2-\ell)^n$ is strictly decreasing with increasing ℓ. To show this, consider it as a continuous function of ℓ. The first derivative is $f'(\ell, n) = -nf(\ell, n-1)$ which is always strictly negative.

Next, observe that if we relax the run-length-limited condition on $\mathbf{n}_\ell$ then the count $B(\ell)$ can be upper bounded as $B(\ell) \leq f(\ell, n)$ for all $\ell > 0$.

Finally, we estimate the first term $B(0)$ using the inclusion-exclusion principle and show tight asymptotic behavior of this term. In particular, our objective is to show that it dominates the other terms in the asymptotics of large n. First, we note the following upper bound,

$$\begin{aligned}\sum_{\ell=1}^{q-2}(q-1-\ell)B(\ell) &\leq \sum_{\ell=1}^{q-2}(q-1-\ell)f(\ell,n) \\ &\leq (q-2)^2 \times \\ &\quad \big((q-1)^n - 2(q-2)^n + (q-3)^n\big),\end{aligned}$$

where the last inequality is obtained by using the fact that $f(\ell, n)$ is strictly decreasing with increasing ℓ. We can express $B(0)$ as follows,

$$B(0) = \sum_{i=0}^{q-2}(-1)^i\binom{q}{i}(q-i)^n. \tag{4}$$

This term can be clearly upper bounded as

$$B(0) \leq q^n.$$

To get a lower bound on $B(0)$, we expand it and pair up the terms, starting from the third term. For q even, we get

$$\begin{aligned}B(0) &= (q^n - q(q-1)^n) + \\ &\quad \sum_{i=1}^{\frac{q}{2}-2}\left(\binom{q}{2i}(q-2i)^n - \binom{q}{2i+1}(q-2i-1)^n\right) \\ &\quad +\binom{q}{2}2^n,\end{aligned}$$

and for q odd we get

$$\begin{aligned}B(0) &= (q^n - q(q-1)^n) + \\ &\quad \sum_{i=1}^{\lfloor\frac{q}{2}\rfloor-1}\left(\binom{q}{2i}(q-2i)^n - \binom{q}{2i+1}(q-2i-1)^n\right).\end{aligned}$$

Note that all the pairs of terms are positive for sufficiently large n, and hence we can lower bound $B(0)$ as

$$B(0) \geq q^n - q(q-1)^n.$$

Therefore, we can get an upper and lower estimate for the upper bound on the average cost, and both these upper and lower estimates converge to $q-1$. We show this next.

$$\begin{aligned}&\frac{(q-1)B(0)}{q^n} + \frac{(q-2)^2}{q^{n-1}}\big((q-1)^n - 2(q-2)^n + (q-3)^n\big) \\ &\leq (q-1)\bigg(1 + (q-2)^2(q-1)\Big[\Big(\frac{q-1}{q}\Big)^n \\ &\quad - 2\frac{q-2}{q-1}\Big(\frac{q-2}{q}\Big)^{n-1} + \frac{q-3}{q-1}\Big(\frac{q-3}{q-1}\Big)^{n-1}\Big]\bigg) \\ &\to (q-1),\ \text{ for } n\to\infty.\end{aligned}$$

Also,

$$\begin{aligned}&\frac{(q-1)B(0)}{q^n} + \frac{(q-2)^2}{q^{n-1}}\big((q-1)^n - 2(q-2)^n + (q-3)^n\big) \\ &\geq (q-1)\Big[1 - q\Big(\frac{q-1}{q}\Big)^n\Big] + (q-2)^2(q-1)\Big[\Big(\frac{q-1}{q}\Big)^n \\ &\quad - 2\frac{q-2}{q-1}\Big(\frac{q-2}{q}\Big)^{n-1} + \frac{q-3}{q-1}\Big(\frac{q-3}{q-1}\Big)^{n-1}\Big] \\ &\to (q-1),\ \text{ for } n\to\infty.\end{aligned}$$

We note that the first term involving $B(0)$ dominates in the asymptotics of large n. ■

For large n, the upper bound converges to the worst case cost $q-1$. Intuitively, this is not surprising since the probability that all the symbols from the alphabet appear in a codeword of length n tends to one for large blocklengths n.

Scheme A can be generalized to a different scheme that we call Scheme B. This new scheme is based on the observation that given a codeword we can change the charge level of any individual cell independently of the other cells; thus we can potentially increase the average number of rewrites that can be performed. This construction can also be viewed as a generalization of the construction in [7] from binary to q-ary.

Scheme B: Consider $\mathcal{C}[n, k+\delta, d]$ as a subspace of $\mathbb{F}_q^n$ containing the all-one vector j, and let $\mathcal{D}$ be a subcode of $\mathcal{C}$ of dimension δ such that $\mathsf{j} \in \mathcal{D}$. We encode the coset code $C = \mathcal{C}/\mathcal{D}$ of size q^k by using the encoding function in (1) and the decoding function is given by

$$\beta(\mathsf{s}) = \mathsf{s} \bmod q + \mathcal{D}.$$

Intuitively, Scheme B divides the total number of discrete charge levels L into L/q parts, each part representing q distinct levels, and each individual cell is increased to the next higher part independently of the other cells. In particular, for $q = 2$ all the even levels in Fig. 1 represent bit 0, and all the odd levels represent bit 1. If $q = 3$, then the levels 0 and 3 represent 0, levels 1 and 4 represent 1, and levels 2 and 5 correspond to 2. This is a generalization of the method in [7] where the number of levels is divided into distinct sets of size two. It differs from the same work in the encoding because we do not minimize only bit flips. The maximum increase in the level happens when the cell transitions from level $0 \bmod q$ to $(q-1) \bmod q$, or $i \bmod q$ to $(i-1) \bmod q$, $i = 1, \dots, q-1$, which incurs a cost of $q-1$. Thus, Lemma 3.1 holds for Scheme B. On the other hand, the average number of rewrites of Scheme B is potentially better than Scheme A. This is illustrated in Section 4-B.

4. Comparison of Coding Schemes

In this section we compare the coding schemes Scheme A and Scheme B against the rank modulation scheme of [8, Construction 18]. In particular, we analyze the information rate for the worst case cost, and the average number of rewrites between the different schemes.

A. Comparison of Worst Case Behavior

To compare the worst case behavior of Scheme A and Scheme B with the previous work that addresses the problem of uniform discharge, we first briefly introduce the rank modulation scheme. The rank modulation coding scheme in [8] considers permutation vectors as the codewords. Every permutation word of length n corresponds to n distinct charge levels. In [8, Construction 18] the following scheme is proposed for ensuring that the code is optimal in minimizing the *worst case rewrite cost.*

Construction 18: (see [8]) Let S_n denote the set of all permutations of the set $[n]$, and let ${}^{[n]}P_m$ denote the set of all m-permutations of the set $[n]$. If $\gamma(\mathsf{s} \to \mathsf{s}') \leq m$, then ${}^{n}P_m = n!/(n-m)!$ words are uniquely represented by all the words in ${}^{[n]}P_m$. Let $a = (a_1, \dots, a_m) \in {}^{[n]}P_m$. Define the *prefix set* $P_m(a)$ as the set of all permutations in S_n which have a as a prefix. Then a vector $a \in {}^{[n]}P_m$ is encoded to a permutation vector in the prefix set $P_m(a)$ by choosing the permutation which minimizes the maximum level in the new state vector. Thus the rank modulation scheme encodes $\log_2 {}^{n}P_m \leq m \log_2 n$ bits of information as state vectors.

In comparison, for $q - 1 = m$, and for m a constant, a coset code $C = \mathbb{F}_q^n/\mathcal{D}$, where $\mathcal{D}$ has constant dimension δ (independent of n) has the same worst case rewrite cost. However the size of the coset code is $q^{n-\delta}$, and so it encodes $(n-\delta)\log_2 q$ bits of information on every write. The ratio of the number of encoded bits is lower bounded as

$$\frac{\log_2 |C|}{\log_2 {}^{n}P_m} \geq \frac{(n-\delta)\log_2 q}{m \log_2 n} = \frac{(n-\delta)\log_2 q}{(q-1)\log_2 n} \to \infty,$$

when $n \to \infty$. Thus, the coset coding scheme can encode asymptotically more information than the rank modulation scheme for same maximum rewrite cost. Note that this scheme uses the "entire space" in both the linear space and the permutation space and so provides no correction of random errors. Both the schemes can provide error correction in case of uniform charge leakage.

B. Comparison of Average Behavior

To compare the coset code constructions in Scheme B with Construction 18, we first ensure that the total number of discrete levels L are the same. Next, we ensure that the worst case cost of both the schemes are the same. Therefore, we fix $m = q - 1$ and consider the optimal rank modulation code with this worst case cost. Finally, we consider linear codes and permutation codes of the same block length n. Given these constraints, we determine the average number of times we are able to rewrite by randomly choosing the next vector from the respective spaces. In the case of rank modulation code, this random vector is chosen by picking the best vector from the set $P_m(a)$ where $a \in {}^{[n]}P_m$ is randomly selected. For the linear code, a random coset in $\mathbb{F}_q^n/\mathcal{D}$ is chosen, where $\mathsf{j} \in \mathcal{D}$.

Fig. 2 shows the performance of the different schemes for $L = 16$, $n = 8$, $q = 3$, $m = 2$. Table I compares the parameters of the codes that are being simulated. The number of rewrites is averaged over 1000 trials. In each trial, we select words at random and we count the number of times we can rewrite until the level exceeds L. The horizontal axis shows the number of rewrites and the vertical axis shows the frequency of that number. The average number of rewrites of the rank modulation codes is 6, which is significantly lower than the average number of rewrites of the coset codes, viz. 22 for Scheme B with cosets of $\mathcal{D} = \langle 11110000, 00001111 \rangle$ and 38 for Scheme B with cosets of $\mathcal{D} = \langle 11000000, 00110000, 00001100, 00000011 \rangle$, even though the codes have the same worst case cost of rewrites.

To compare the performance of Scheme A with Scheme B, we consider the same coset code $C = \mathbb{F}_3^8/\mathcal{D}$, with $\mathcal{D} = \langle \mathsf{j}_8 \rangle$. The average number of rewrites in Scheme B is 18 which is larger than the average of 14 in Scheme A, and 12 for $\mathbb{F}_3^8$; see Fig. 3. We also compare the performance of Scheme B with the FlipMin scheme in Fig. 3, for $q = 2, \mathcal{D} = \langle \mathsf{j}_8, (\mathsf{j}_4, 0_4) \rangle$. FlipMin achieves an average of 38.2 rewrites, while Scheme B achieves 39.6 rewrites on average. Note that we increase the charge level in each cell individually in FlipMin. The figure also shows the effect of using different alphabets and cosets on the rewrite performance of Scheme A and Scheme B.

TABLE I
TABLE COMPARING THE SIZES AND INFORMATION BITS OF THE CODES USED IN THE DIFFERENT SCHEMES FOR $n = 8$

Scheme	Coset dim./Prefix len.	q	Size	Bits encoded
Construction 18	2	3	56	< 6
Scheme B	2	3	729	> 9
Scheme B	4	3	81	> 6
Scheme A/B	1	3	2187	> 11
Scheme B	0	3	6561	> 12
FlipMin/Scheme B	2	2	64	6

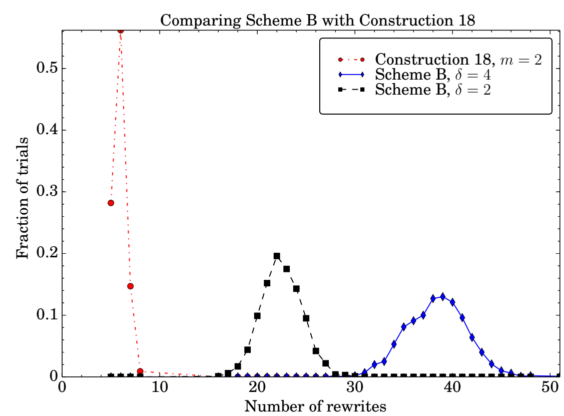

Fig. 2. Comparing Rank Modulation with Scheme B, $L = 16, n = 8, q = 3$

5. Conclusion

We introduce coset codes of linear codes to correct uniform charge leakage, improve rewritability, and to correct random errors in flash memory storage. It will be interesting to combine and study coset codes which can also address the problem of overshooting if we relax the requirement of careful charge injection during writing in flash memory.

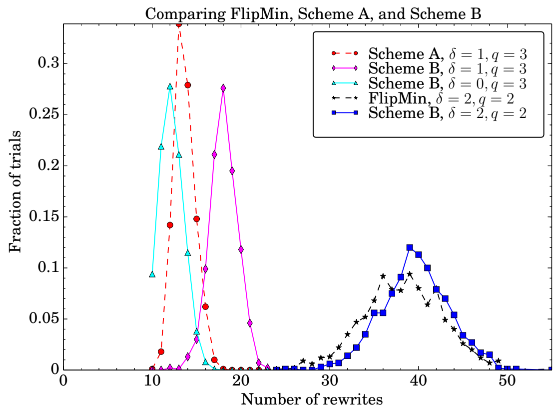

Fig. 3. Comparing Schemes A, B and FlipMin for $L = 16, n = 8$

Acknowledgement

This work was done while H. M. Kiah was a graduate student at Nanyang Technological University. The authors thank the anonymous reviewers for their constructive comments.

References

[1] P. Cappelletti and C. Golla, *Flash memories*, Kluwer Academic Publishers, 1999.

[2] R. L. Rivest and A. Shamir, "How to reuse a write-once memory," *Information and control*, vol. 55, no. 1, pp. 1–19, 1982.

[3] G. Cohen, P. Godlewski, and F. Merkx, "Linear binary code for write-once memories," *IEEE Trans. Inform. Th.*, vol. 32, no. 5, pp. 697–700, May 1986.

[4] A. A. Jiang, Y. Li, E. En Gad, M. Langberg, and J. Bruck, "Joint rewriting and error correction in write-once memories," in *IEEE Int. Symp. Inform. Th. Proc.*, 2013, pp. 1067–1071.

[5] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories," *IEEE Trans. Inform. Th.*, vol. 58, no. 9, pp. 5985–5999, September 2012.

[6] A. N. Jacobvitz, R. Calderbank, and D. J. Sorin, "Writing cosets of a convolutional code to increase the lifetime of flash memory," in *50th Annual Allerton Conf. Commun., Control, Comput.*, 2012, pp. 308–318.

[7] A. N. Jacobvitz, A. R. Calderbank, and D. J. Sorin, "Coset coding to extend the lifetime of memory," in *HPCA*, 2013, pp. 222–233.

[8] A. A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck, "Rank modulation for flash memories," *IEEE Trans. Inform. Th.*, vol. 55, no. 6, pp. 2659–2673, 2009.

[9] E. En Gad, A. A. Jiang, and J. Bruck, "Compressed encoding for rank modulation," in *IEEE Int. Symp. Inform. Th. Proc.*, 2011, pp. 884–888.

[10] A. A. Jiang, H. Li, and Y. Wang, "Error scrubbing codes for flash memories," in *11th Canadian Workshop Inform. Th.*, 2009, pp. 32–35.

[11] F. Farnoud, V. Skachek, and O. Milenkovic, "Error-correction in flash memories via codes in the ulam metric," *IEEE Trans. Inform. Theory*, vol. 59, pp. 3003–3020, 2013.

[12] K. Haymaker and C. Kelley, "Geometric WOM codes and coding strategies for multilevel flash memories," *Designs, Codes and Cryptography*, vol. 70, no. 1-2, pp. 91–104, 2014.

[13] B. M. Kurkoski, "Rewriting codes for flash memories based upon lattices, and an example using the e8 lattice," in *IEEE GLOBECOM Workshops*, 2010, pp. 1861–1865.

[14] A. A. Jiang and Y. Wang, "Rank modulation with multiplicity," in *2010 IEEE GLOBECOM Workshops*. IEEE, 2010, pp. 1866–1870.

[15] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland Publishing Co., 1977.

[16] D. Tang and L. Bahl, "Block codes for a class of constrained noiseless channels," *Inform. and Control*, vol. 17, no. 5, pp. 436 – 461, 1970.

[17] Y. M. Chee, H. M. Kiah, P. Purkayastha, and C. Wang, "Cross-bifix-free codes within a constant factor of optimality," *IEEE Trans. Inform. Th.*, vol. 59, no. 7, pp. 4668–4674, July 2013.