Bit-fixing Codes for Multi-level Cells

Anxiao (Andrew) Jiang
Computer Science and Eng. Dept.
Texas A&M University
College Station, TX 77843
ajiang@cse.tamu.edu

Yue Li
Computer Science and Eng. Dept.
Texas A&M University
College Station, TX 77843
yli@cse.tamu.edu

Jehoshua Bruck
Electrical Engineering Dept.
California Institute of Technology
Pasadena, CA 91125
bruck@caltech.edu

***Abstract*—Codes that correct limited-magnitude errors for multi-level cell nonvolatile memories, such as flash memories and phase-change memories, have received interest in recent years. This work proposes a new coding scheme that generalizes a known result [2] and works for arbitrary error distributions. In this scheme, every cell's discrete level ℓ is mapped to its binary representation $(b_{m-1}, \cdots, b_1, b_0)$, where the m bits belong to m different error-correcting codes. The error ε in a cell is mapped to its binary representation $(e_{m-1}, \cdots, e_1, e_0)$, and the codes are designed such that every error bit e_i only affects the codeword containing the data bit b_i. The m codewords are decoded sequentially to correct the bit-errors $e_0, e_1, \cdots, e_{m-1}$ in order. The scheme can be generalized to many more numeral systems for cell levels and errors, optimized cell-level labelings, and any number of cell levels. It can be applied not only to storage but also to amplitude-modulation communication systems.**

I. Introduction

Multi-level cell (MLC) nonvolatile memories have become increasing important for storage in recent years. A well-known example is flash memory, where cells with $q = 4$ and 8 levels are already widely used. Emerging nonvolatile memories, such as phase-change memory, are also embracing the MLC technology. Generally speaking, every cell has an analog value that represents the cell's state. For flash memory, the value is the threshold voltage of the cell; and for phase-change memory, it is the electrical resistance of the cell. The values of cells are quantized into q discrete levels, which can store $\log_2 q$ bits. MLCs with more and more levels are actively developed for higher storage capacity. For example, flash memories of $q = 16$ levels have been demonstrated.

It is important to design error-correcting codes (ECCs) that consider the many properties of MLC memories. Errors often have limited magnitudes and non-symmetric distributions, due to the memories' unique disturb and noise mechanisms. Also, current MLCs are restricted by q, the number of levels, being a power of 2. Coding schemes that can map cells to binary codes conveniently for an arbitrary number of levels are worth studying. All these will be addressed in this paper.

There are different approaches to map cell levels to binary codes when q is a power of 2, including binary representation and Gray codes. Consider n cells; and for $i = 1, \cdots, n$, let $\ell_i \in \{0, 1, \cdots, q-1\}$ be the level of the ith cell. Let $m = \log_2 q$. And let $\mathcal{B}_m(\ell_i) \triangleq (b_{i,m-1}, \cdots, b_{i,1}, b_{i,0}) \in \{0,1\}^m$ be the binary representation of ℓ_i, namely, $\ell_i = \sum_{j=0}^{m-1} b_{i,j} \cdot 2^j$. Since the m bits in a cell have different error probabilities, in a basic binary-representation approach, m ECCs of different rates are used. Specifically, for $j = 0, 1, \cdots, m-1$, we let $(b_{1,j}, \cdots, b_{n,j})$ be a separate ECC. To further reduce error probabilities, a more common approach is to represent the bits in a cell using Gray codes, and then apply m ECCs.

In addition to the above approaches, ECCs for limited-magnitude errors have received interest recently. In [2], a new coding scheme with applications to flash memories, etc., was presented. To correct t asymmetric errors of maximum magnitude τ, it takes the modulo $\tau + 1$ of the cell levels $(\ell_1, \cdots, \ell_n)$ and applies to it an ECC of alphabet size $\tau + 1$ that corrects t errors. The scheme substantially extends the interesting study in [1] and can be generalized to bidirectional errors. In [5], the study was extended to errors of graded magnitude, and nice results were presented. The work focused on a generalized error model in which at most t_1 asymmetric errors of maximum magnitude τ_1 and at most t_2 asymmetric errors of maximum magnitude τ_2, with $\tau_1 < \tau_2$, may occur. More codes for limited-magnitude errors have been presented recently. In particular, optimal systematic ECCs for both asymmetric and symmetric errors were presented in [3], [4].

In this paper, we propose an alternative coding scheme named *bit-fixing code*. Its main idea is to sequentially correct the bits in the binary representation of errors. And it can be generalized to more numeral systems. When $q = 2^m$, let $\varepsilon_i \in \{-\ell_i, \cdots, 0, \cdots, q-1-\ell_i\}$ denote the additive error in the ith cell's level ℓ_i, and let $\mathcal{B}_m(\varepsilon_i \mod q) \triangleq (e_{i,m-1}, \cdots, e_{i,0}) \in \{0,1\}^m$ be the binary representation of $\varepsilon_i \mod q$. For $j = 0, \cdots, m-1$, let $(b_{1,j}, \cdots, b_{n,j})$ be a binary ECC $\mathcal{C}j$. The scheme has the nice property that the error bits $(e_{1,j}, \cdots, e_{n,j})$ only affect the code $\mathcal{C}_j$. (Note that this property does not hold for the binary-representation scheme introduced above.) That enables us to allocate redundancy appropriately and decode $\mathcal{C}_0, \cdots, \mathcal{C}_{m-1}$ sequentially.

The bit-fixing coding scheme can be applied to arbitrary error distributions, including both asymmetric and bidirectional errors. It can be generalized from the binary representation to many more numeral representations, including k-ary numbers (for any integer $k \geq 2$) and mixed-radix numeral systems such as factoradic systems. It can also be extended to an arbitrary number of cell levels, which means q can be any integer instead of a power of 2 and binary codes can still be used. The coding scheme in fact contains the ECC for asymmetric errors of limited magnitude in [2] as a special case. It is also related to the codes in [5], but is more specific in its construction and more general in various ways. It can be applied not only to storage but also to amplitude-modulation communication systems. (Due to space limitation, we skip some details.)

II. Bit-fixing Coding Scheme

A. *Numeral Systems for Cell Levels and Errors*

Consider n memory cells of q levels. For $i = 1, \cdots, n$, let $\ell_i \in \{0, 1 \cdots, q-1\}$ be the written level of the ith cell, and let $\varepsilon_i \in \{-\ell_i, \cdots, 0, \cdots, q-1-\ell_i\}$ be the additive error in the ith cell. Then the noisy cell level – the level we read – of the ith cell is $\ell'_i = \ell_i + \varepsilon_i \in \{0, 1, \cdots, q-1\}$.

Given a non-negative integer x, let $\mathcal{B}_i(x)$ denote the last i bits in the binary representation of x. That is, if $\mathcal{B}_i(x) = (y_{i-1}, \cdots, y_1, y_0) \in \{0,1\}^i$, then $\left(x \mod 2^i\right) = \sum_{j=0}^{i-1} y_j \cdot 2^j$. For example, $\mathcal{B}_3(5) = (1,0,1)$ and $\mathcal{B}_2(5) = (0,1)$. Let $m = \lceil \log_2 q \rceil$. Let $(b_{i,m-1}, \cdots, b_{i,1}, b_{i,0}) \triangleq \mathcal{B}_m(\ell_i)$ be the last m bits in the binary representation of the cell level ℓ_i, and let $(e_{i,m-1}, \cdots, e_{i,1}, e_{i,0}) \triangleq \mathcal{B}_m(\varepsilon_i \mod 2^m)$ be the last m bits in the binary representation of $\varepsilon_i \mod 2^m$. Note that if errors are asymmetric, – say they are upward errors, namely $\forall\ i,\ \varepsilon_i \geq 0$, – then $(e_{i,m-1}, \cdots, e_{i,1}, e_{i,0}) = \mathcal{B}_m(\varepsilon_i)$.

We can extend the representation of cell levels and errors from binary representation to more general numeral systems. Let $c_1, c_2, \cdots, c_{m'}$ be positive integers such that $\prod_{i=1}^{m'} c_i \geq q$. Then in a mixed-radix numeral system with bases $(c_1, c_2, \cdots, c_{m'})$, every integer $x \in \{0, 1, \cdots, q-1\}$ has a unique representation $\mathcal{R}(x) = (y_{m'-1}, \cdots, y_1, y_0)$ – where for $i = 0, \cdots, m'-1$, the digit $y_i \in \{0, 1, \cdots, c_{i+1}-1\}$ – that satisfies the condition $x = y_0 + y_1 c_1 + y_2 c_1 c_2 + \cdots + y_{m'-1} c_1 c_2 \cdots c_{m'-1}$. Note that the mixed-radix numeral system contains several common numeral systems as special cases: (1) If $c_1 = c_2 = \cdots = c_{m'} = 2$, it becomes the binary representation; (2) More generally, if $c_1 = c_2 = \cdots = c_{m'} = k$ for some integer $k \geq 2$, it becomes the k-ary representation; (3) If $c_i = i+1$, it becomes the factorial number system. As we will see, for the bit-fixing coding scheme, given q, it is beneficial to choose the bases c_i (for $i = 1, \cdots, m'$) with two properties: First, good ECCs for alphabet size c_i can be designed; Second, the representations of errors (modulo $c_1 c_2 \cdots c_{m'}$) contain as few non-zero digits as possible.

B. *Bit-fixing Coding Scheme for Binary Representation*

We first present the bit-fixing coding scheme for the case where q is a power of 2 and binary representations are used. It will be extended to general cases later.

Construction 1. Encoding of Bit-fixing Scheme

For $j = 0, 1, \cdots, m-1$, let $\mathcal{C}_j$ be an (n, k_j) binary ECC that can correct t_j errors. We store $k_0 + k_1 + \cdots + k_{m-1}$ information bits in n cells of $q = 2^m$ levels as follows. First, we partition the information bits into m chunks, where for $j = 0, \cdots, m-1$, the jth chunk has k_j information bits: $\mathbf{d}_j = (d_{j,1}, d_{j,2}, \cdots, d_{j,k_j})$. Next, for $j = 0, \cdots, m-1$, we use $\mathcal{C}_j$ to encode $\mathbf{d}_j$ into a codeword $\mathbf{b}_j = (b_{1,j}, b_{2,j}, \cdots, b_{n,j})$. Then, for $i = 1, \cdots, n$, let $\ell_i = \sum_{j=0}^{m-1} b_{i,j} \cdot 2^j$, and we write the ith cell's level as ℓ_i.

After cells are written, additive errors $\varepsilon_1, \cdots, \varepsilon_n$ will appear and change cell levels to $\ell'_1 = \ell_1 + \varepsilon_1, \cdots, \ell'_n = \ell_n + \varepsilon_n$.

Construction 2. Decoding of Bit-fixing Scheme

Let $\ell'_1, \cdots, \ell'_n$ be the noisy cell levels we read. As the initialization step, for $i = 1, \cdots, n$, let $\hat{\ell}_i = \ell'_i$.

For $j = 0, 1, \cdots, m-1$, carry out these three steps:

1) *For $i = 1, \cdots, n$, let $(\hat{b}_{i,m-1}, \cdots, \hat{b}_{i,1}, \hat{b}_{i,0}) = \mathcal{B}_m(\hat{\ell}_i)$ be the binary representation of the estimated cell level $\hat{\ell}_i$.*
2) *Use code $\mathcal{C}_j$ to decode the codeword $(\hat{b}_{1,j}, \cdots, \hat{b}_{n,j})$, and let $(\hat{e}_{1,j}, \cdots, \hat{e}_{n,j})$ be the discovered error vector. (That is, the recovered codeword is $(\hat{b}_{1,j} \oplus \hat{e}_{1,j}, \cdots, \hat{b}_{n,j} \oplus \hat{e}_{n,j})$, where "$\oplus$" is the exclusive-OR operation.)*
3) *For $i = 1, \cdots, n$, update the estimated cell level $\hat{\ell}_i$ as follows: $\hat{\ell}_i \leftarrow \left(\hat{\ell}_i - \hat{e}_{i,j} \cdot 2^j \mod q\right)$.*

Now $\hat{\ell}_1, \cdots, \hat{\ell}_n$ are our recovered cell levels. From them, the information bits can be readily obtained.

In the above decoding algorithm, m ECCs $\mathcal{C}_0, \mathcal{C}_1, \cdots, \mathcal{C}_{m-1}$ are decoded sequentially. There is a nice mapping: The codeword of each $\mathcal{C}_i$ is $(b_{1,i}, \cdots, b_{n,i})$, which are the bits at position i in the binary representations of cell levels $(\ell_1, \cdots, \ell_n)$; and as will be shown later, the binary errors in that codeword are $(e_{1,i}, \cdots, e_{n,i})$, which are the bits at position i in the binary representations of (the modulo q of) the errors $(\varepsilon_1, \cdots, \varepsilon_n)$. However, note that the error vectors $(e_{1,i}, \cdots, e_{n,i})$ cannot be obtained directly from the binary representations of the noisy cell levels $(\ell'_1 = \ell_1 + \varepsilon_1, \cdots, \ell'_n = \ell_n + \varepsilon_n)$ (except for $i = 0$). Instead, they are obtained gradually as more and more ECCs are decoded and the estimations of cell levels are made closer and closer to their true values.

Example 3. *Consider n cells of $q = 8$ levels. Then $m = \log_2 q = 3$. Assume $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2$ can correct no less than 3, 1, and 2 errors, respectively. Without loss of generality (WLOG), suppose that after cells are written, errors appear in cells 1, 2 and 3, respectively. Let $\ell_1 = 3, \ell_2 = 1, \ell_3 = 2$ be their original levels, and let $\varepsilon_1 = 1, \varepsilon_2 = 5, \varepsilon_3 = -1$ be their errors. Then their noisy levels are $\ell'_1 = 4, \ell'_2 = 6, \ell'_3 = 1$, respectively. (See Fig.* 1 *and the following table for an illustration.)*

	Cell 1	*Cell 2*	*Cell 3*
Original level	*3: $(\underline{0},\underline{1},\underline{1})$*	*1: $(\underline{0},\underline{0},\underline{1})$*	*2: $(\underline{0},\underline{1},\underline{0})$*
Error	*1: (0,0,1)*	*5: (1,0,1)*	*-1: (1,1,1)*
Noisy level	*4: (1,0,0)*	*6: (1,1,0)*	*1: (0,0,1)*
Level after decoding $\mathcal{C}_0$	*3: $(0,1,\underline{1})$*	*5: $(1,0,\underline{1})$*	*0: $(0,0,\underline{0})$*
Level after decoding $\mathcal{C}_1$	*3: $(0,\underline{1},1)$*	*5: $(1,\underline{0},1)$*	*6: $(1,\underline{1},0)$*
Level after decoding $\mathcal{C}_2$	*3: $(\underline{0},1,1)$*	*1: $(\underline{0},0,1)$*	*2: $(\underline{0},1,0)$*

In the decoding process, we first decode $\mathcal{C}_0$, where the noisy codeword is $(0, 0, 1, \cdots)$. (It is because the least-significant bits (LSB) of $(\mathcal{B}_m(\ell'_1), \mathcal{B}_m(\ell'_2), \mathcal{B}_m(\ell'_3), \cdots)$ are $(0, 0, 1, \cdots)$.) By decoding it, we find its error vector $(e_{1,0}, e_{2,0}, e_{3,0}, \cdots) = (1, 1, 1, \cdots)$. So we change the cell levels to $(4 - e_{1,0} \mod 8, 6 - e_{2,0} \mod 8, 1 - e_{3,0} \mod 8) = (3, 5, 0)$.

Next, we decode $\mathcal{C}_1$, where the noisy codeword is $(1, 0, 0, \cdots)$. (It is because the middle bits of $(\mathcal{B}_m(3), \mathcal{B}_m(5), \mathcal{B}_m(0), \cdots)$ are $(1, 0, 0, \cdots)$.) By decoding it, we find its error vector $(e_{1,1}, e_{2,1}, e_{3,1}, \cdots) = (0, 0, 1, \cdots)$. So we change the cell levels to $(3 - e_{1,1} \cdot 2 \mod 8, 5 - e_{2,1} \cdot 2 \mod 8, 0 - e_{3,1} \cdot 2 \mod 8) = (3, 5, 6)$.

We then decode $\mathcal{C}_2$, where the noisy codeword is $(0,1,1,\cdots)$. *(It is because the most-significant bits (MSB) of* $(\mathcal{B}_m(3),\mathcal{B}_m(5),\mathcal{B}_m(6),\cdots)$ *are* $(0,1,1,\cdots)$.*) By decoding it, we find its error vector* $(e_{1,2},e_{2,2},e_{3,2},\cdots)=(0,1,1,\cdots)$. *So we change the cell levels to* $(3-e_{1,2}\cdot 2^2 \mod 8, 5-e_{2,2}\cdot 2^2 \mod 8, 6-e_{3,2}\cdot 2^2 \mod 8)=(3,1,2)$. *They are the original cell levels, from which we can recover information bits.* □

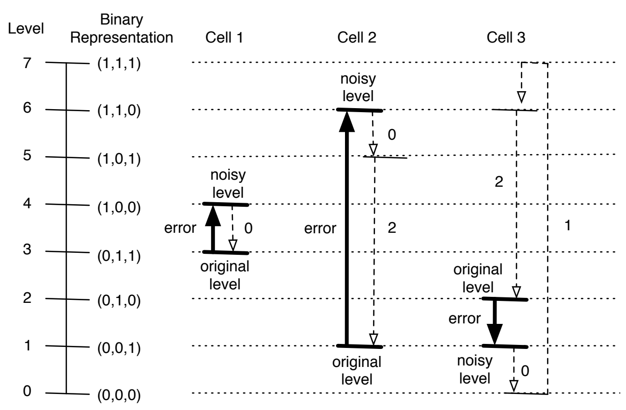


Fig. 1. Decoding process of bit-fixing scheme. Here thick solid arrows show how errors change cell levels to noisy levels. And thin dotted arrows show how decoding changes the noisy levels back to the original levels. For $i=0,1,2$, the thin dotted arrows labeled by i correspond to the decoding of $\mathcal{C}_i$.

We now prove the number of errors of variable magnitudes the bit-fixing coding scheme can correct. Given a vector $\mathbf{v}=(v_{k-1},\cdots,v_1,v_0)\in\{0,1\}^k$, define its *support* as $support(\mathbf{v})\triangleq\{i|i\in\{0,1,\cdots,k-1\},v_i=1\}$. Given $i\in\{0,1,\cdots,m-1\}$, we define the *cross of i* as $cross_m(i)\triangleq$

$$\{j|j\in\{0,1,\cdots,2^m-1\},i\in support(\mathcal{B}_m(j))\}.$$

Namely, $cross_m(i)$ is the set of integers in $\{0,1,\cdots,2^m-1\}$ whose binary representations have 1 in the ith position.

Example 4. *Let* $m=3$. *Since* $\mathcal{B}_m(j)=(0,0,0)$, $(0,0,1)$, $(0,1,0)$, $(0,1,1)$, $(1,0,0)$, $(1,0,1)$, $(1,1,0)$, $(1,1,1)$ *for* $j=0,\cdots,7$ *respectively, we have* $cross_m(0)=\{1,3,5,7\}$, $cross_m(1)=\{2,3,6,7\}$, $cross_m(2)=\{4,5,6,7\}$. □

For $i=0,1,\cdots,q-1$, define $\gamma_i\triangleq|\{j|j\in\{1,\cdots,n\},\varepsilon_j\equiv i \mod q\}|$. That is, there are γ_i cells with errors of magnitude exactly i (mod q).

Theorem 5. *The bit-fixing coding scheme can recover all information bits if for* $j=0,1,\cdots,m-1$, *the binary error-correcting code* $\mathcal{C}_j$ *can correct* $\sum_{k\in cross_m(j)}\gamma_k$ *binary errors.*

Proof: The decoding algorithm in Construction 2 decodes $\mathcal{C}_0,\cdots,\mathcal{C}_{m-1}$ sequentially and updates the cell-level estimation $\hat{\ell}_i$ (for $i=1,\cdots,n$) along the way. We prove this claim by induction: For $j=0,1,\cdots,m-1$, after $\mathcal{C}_j$ is decoded, each $\hat{\ell}_i$ is updated as $\ell_i+\sum_{k=j+1}^{m-1}e_{i,k}\cdot 2^k \mod q$.

First consider the base case $j=0$. The noisy codeword for $\mathcal{C}_0$ is $(\ell'_1 \mod 2,\cdots,\ell'_n \mod 2)$. Since for $i=1,\cdots,n$, $\ell'_i\equiv\ell_i+\varepsilon_i\equiv\sum_{k=0}^{m-1}b_{i,k}\cdot 2^k+\sum_{k=0}^{m-1}e_{i,k}\cdot 2^k \mod q$, the noisy codeword is $(b_{1,0}\oplus e_{1,0},\cdots,b_{n,0}\oplus e_{n,0})$. The Hamming weight of the error vector $(e_{1,0},\cdots,e_{n,0})$ is $\sum_{k\in cross_m(0)}\gamma_k$, so $\mathcal{C}_0$ can correct all those errors. Then $\hat{\ell}_i$ is updated as $(\ell'_i-e_{i,0} \mod q)=(\ell_i+\sum_{k=1}^{m-1}e_{i,k}\cdot 2^k \mod q)$.

Now suppose the claim holds for $j=0,1,\cdots,j'<m-1$. Consider $j=j'+1$. The noisy codeword for $\mathcal{C}_j$ is $(\hat{b}_{1,j},\cdots,\hat{b}_{n,j})$, where $\hat{b}_{i,j}$ is the $(j+1)$th LSB in the binary representation of $\hat{\ell}_i$ and equals $\left(\frac{\hat{\ell}_i-(\hat{\ell}_i \mod 2^j)}{2^j} \mod 2\right)$. By induction assumption, $\hat{\ell}_i=(\ell_i+\sum_{k=j}^{m-1}e_{i,k}\cdot 2^k \mod q)=(\sum_{k=0}^{m-1}b_{i,k}\cdot 2^k+\sum_{k=j}^{m-1}e_{i,k}\cdot 2^k \mod q)$. So $\hat{b}_{i,j}=b_{i,j}\oplus e_{i,j}$. So the noisy codeword is $(b_{1,j}\oplus e_{1,j},\cdots,b_{n,j}\oplus e_{n,j})$, which has $\sum_{k\in cross_m(j)}\gamma_k$ errors. So $\mathcal{C}_j$ can correct all those errors. Then $\hat{\ell}_i$ is updated as $((\ell_i+\sum_{k=j}^{m-1}e_{i,k}\cdot 2^k)-e_{i,j}\cdot 2^j \mod q)=(\ell_i+\sum_{k=j+1}^{m-1}e_{i,k}\cdot 2^k \mod q)$.

So after $\mathcal{C}_0,\cdots,\mathcal{C}_{m-1}$ are all decoded, each estimated cell level $\hat{\ell}_i$ equals the original level ℓ_i, from which information bits can be recovered. ■

C. Bit-fixing Coding Scheme for General Numeral Systems

The above bit-fixing coding scheme can be generalized to mixed-radix numeral systems. Let m and $c_1,c_2,\cdots,c_m$ be positive integers, and let $q=c_1c_2\cdots c_m$. Given n cells of q levels, for $i=1,\cdots,n$, we can represent the cell level ℓ_i as $\mathcal{R}(\ell_i)=(b_{i,m-1},\cdots,b_{i,1},b_{i,0})\in\{0,\cdots,c_m-1\}\times\cdots\times\{0,\cdots,c_2-1\}\times\{0,\cdots,c_1-1\}$ using the mixed-radix numeral system with bases $(c_1,c_2,\cdots,c_m)$. Similarly, we can represent the error ε_i as $\mathcal{R}(\varepsilon_i \mod q)=(e_{i,m-1},\cdots,e_{i,1},e_{i,0})$. We can then encode data in the same way as Construction 1, except that each $\mathcal{C}_j$ (for $j=0,1,\cdots,m-1$) is an ECC of alphabet size c_{j+1}. We can decode data in the same way as Construction 2, except that the "$\oplus$" (exclusive-OR) operation is replaced by the "mod c_{j+1}" operation and after $\mathcal{C}_j$ is decoded (for $j=0,1,\cdots,m-1$), the estimated cell level $\hat{\ell}_i$ is updated as $\hat{\ell}_i-\hat{e}_{i,j}\prod_{k=1}^{j}c_k \mod q$.

Given a vector $\mathbf{v}=(v_{m-1},\cdots,v_1,v_0)\in\{0,\cdots,c_m-1\}\times\cdots\times\{0,\cdots,c_2-1\}\times\{0,\cdots,c_1-1\}$, define its *support* as $support(\mathbf{v})\triangleq\{i|i\in\{0,1,\cdots,m-1\},v_i\neq 0\}$. Given $i\in\{0,1,\cdots,m-1\}$, we define the *cross of i* as $cross(i)\triangleq$

$$\{j|j\in\{0,1,\cdots,q-1\},i\in support(\mathcal{R}(j))\}.$$

Theorem 6. *The bit-fixing coding scheme (for the general mixed-radix numeral system) can recover all information bits if for* $j=0,1,\cdots,m-1$, *the* c_{j+1}*-ary error-correcting code* $\mathcal{C}_j$ *can correct* $\sum_{k\in cross(j)}\gamma_k$ *Hamming errors.*

Note that if C_0 is an ECC and $C_1,\cdots,C_{m-1}$ contain no redundancy, the scheme here is reduced to the main code construction (Construction 1) in [2] for asymmetric errors of maximum magnitude c_1-1.

D. Achievable Rate of Bit-fixing Coding Scheme

We now analyze the achievable rates of the bit-fixing coding scheme. For simplicity, we present the case in which q is a power of 2 and binary representations are used. The analysis can be extended naturally to more general cases.

Consider a cell of level $\ell \in \{0, 1, \cdots, q-1\}$. Let $\ell' = \ell + \varepsilon \in \{0, 1 \cdots, q-1\}$ denote the noisy cell level, where $\varepsilon \in \{-\ell, \cdots, 0, \cdots, q-1-\ell\}$ is the error. We assume the errors in different cells are i.i.d. Due to the complex mechanisms for errors (e.g., disturbs and charge leakage, cell-level drifting and different memory manufacturing processes), it is hard to model errors with a universal model. So in this paper, we consider a general model: "$\forall$ $i, j \in \{0, 1, \cdots, q-1\}$, let $p_{i,j} \triangleq Pr\{\varepsilon \equiv j \mod q | \ell = i\}$ be a known distribution." Let $\mathcal{B}_m(\ell) = (b_{m-1}, \cdots, b_1, b_0)$ and $\mathcal{B}_m(\varepsilon \mod q) = (e_{m-1}, \cdots, e_1, e_0)$. Here $b_0, \cdots, b_{m-1}$ are m bits that belong to m different codes $\mathcal{C}_0, \cdots, \mathcal{C}_{m-1}$; and we let them be independent. For $i = 0, 1, \cdots, m-1$, let $\beta_i \triangleq Pr\{b_i = 0\}$; let $\alpha_{i,0} \triangleq Pr\{e_i = 1 | b_i = 0\}$ and $\alpha_{i,1} \triangleq Pr\{e_i = 1 | b_i = 1\}$ denote the cross-over probabilities in the binary channel corresponding to b_i; and define $\overline{cross}_m(i) \triangleq \{0, 1, \cdots, q-1\} - cross_m(i)$.

We can derive the cross-over probabilities: $Pr\{e_i = 1 | b_i = 0\} = \sum_{(d_{m-1}, \cdots, d_1, d_0) : \mathcal{B}_m^{-1}((d_{m-1}, \cdots, d_1, d_0)) \in \overline{cross}_m(i)} \sum_{k \in cross_m(i)} \left(\prod_{x \in \{0, 1, \cdots, m-1\} - \{i\}} \beta_x^{1-d_x} (1-\beta_x)^{d_x} \right) \cdot p_{\mathcal{B}_m^{-1}((d_{m-1}, \cdots, d_1, d_0)), k}$, and $Pr\{e_i = 1 | b_i = 1\} = \sum_{(d_{m-1}, \cdots, d_1, d_0) : \mathcal{B}_m^{-1}((d_{m-1}, \cdots, d_1, d_0)) \in cross_m(i)} \sum_{k \in cross_m(i)} \left(\prod_{x \in \{0, 1, \cdots, m-1\} - \{i\}} \beta_x^{1-d_x} (1-\beta_x)^{d_x} \right) \cdot p_{\mathcal{B}_m^{-1}((d_{m-1}, \cdots, d_1, d_0)), k}$. When $n \to \infty$, for $i = 0, \cdots, m-1$, the code $\mathcal{C}_i$ can achieve rate $I(b_i; b_i \oplus e_i) = H(\beta_i(1-\alpha_{i,0}) + (1-\beta_i)\alpha_{i,1}) - \beta_i H(\alpha_{i,0}) - (1-\beta_i) H(\alpha_{i,1})$, where H is the entropy function. The bit-fixing scheme can achieve rate $\max_{\beta_0, \beta_1, \cdots, \beta_{m-1} \in [0,1]} \sum_{i=0}^{m-1} H(\beta_i(1-\alpha_{i,0}) + (1-\beta_i)\alpha_{i,1}) - \beta_i H(\alpha_{i,0}) - (1-\beta_i) H(\alpha_{i,1})$ bits per cell.

III. Optimal Labeling of Cell-levels

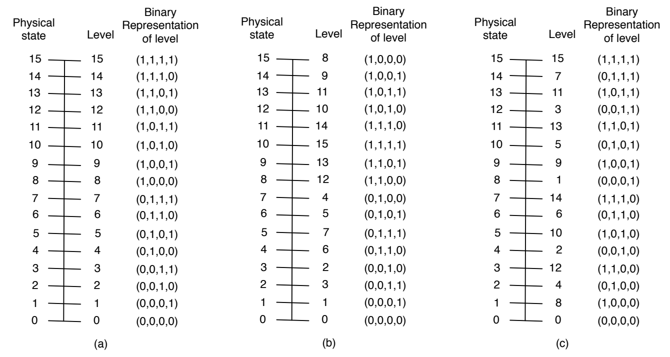
In this section, we present a new technique, *labeling of cell levels*, for better performance. So far, we have not yet differentiated the *physical state* of a cell from the *labeled level* of the cell. We have used ℓ to denote both, and the greater ℓ is, the higher the "physical state" of the cell (e.g., threshold voltage for a flash memory cell) is. However, the bit-fixing scheme presented earlier works for any labeling of cell levels. And this freedom in labeling enables the further optimization of performance. So in this section, we differentiate the physical state $s \in \{0, 1, \cdots, q-1\}$ from the labeled level $\ell \in \{0, 1, \cdots, q-1\}$ of a cell.

Let $\pi : \{0, 1, \cdots, q-1\} \to \{0, 1, \cdots, q-1\}$ be a permutation function that maps every physical state s to its corresponding level $\pi(s)$. Let $s \in \{0, 1, \cdots, q-1\}$ denote the original physical state of a cell, let $\delta \in \{-s, \cdots, 0, \cdots, q-1-s\}$ denote the physical error in it, and let $s' = s + \delta$ denote its noisy physical state. Correspondingly, let $\ell = \pi(s)$ denote its original level, let $\ell' = \pi(s')$ denote its noisy level, and let $\varepsilon = \ell' - \ell$ denote the *error in the cell level*.

The objective of a good labeling is to decrease the number of bit-errors in $\mathcal{C}_0, \cdots, \mathcal{C}_{m-1}$ caused by physical errors, and maximize the overall code rate. In the following, for simplicity, assume $q = 2^m$ and binary representations are used.

Example 7. *Let $q = 16$. Three labelings are shown in Fig. 2, which perform differently. For example, consider an error δ that changes the physical cell state from $s = 5$ to $s' = 4$, namely $\delta = -1$. In Fig. 2 (a) (or (b)), with the straightforward (or Gray-code) labeling, the level is changed from $\ell = 5$ to $\ell' = 4$ (or from $\ell = 7$ to $\ell' = 6$), and the error in level is $\varepsilon = \ell' - \ell = -1$. Since $\mathcal{B}_m(-1 \mod 16) = (1, 1, 1, 1)$, 4 bit-errors are caused. In Fig. 2 (c), however, the same physical error changes the level from $\ell = 10$ to $\ell' = 2$, so $\varepsilon = \ell' - \ell = -8$. Since $\mathcal{B}_m(-8 \mod 16) = (1, 0, 0, 0)$, only 1 bit-error is caused.*

Consider physical errors of magnitude one, which are very common for bidirectional errors. Since $q = 16$, there are 30 such errors. It can be shown that on average, *for such a physical error, the simple labeling in Fig. 2 (a) introduces 2.5 bit-errors, the Gray-code labeling in Fig. 2 (b) introduces 2.13 bit-errors, and the labeling in Fig. 2 (c) introduces only 1.37 bit-errors.* □



Physical state	Level	Binary Representation of level	Physical state	Level	Binary Representation of level	Physical state	Level	Binary Representation of level
15	15	(1,1,1,1)	15	8	(1,0,0,0)	15	15	(1,1,1,1)
14	14	(1,1,1,0)	14	9	(1,0,0,1)	14	7	(0,1,1,1)
13	13	(1,1,0,1)	13	11	(1,0,1,1)	13	11	(1,0,1,1)
12	12	(1,1,0,0)	12	10	(1,0,1,0)	12	3	(0,0,1,1)
11	11	(1,0,1,1)	11	14	(1,1,1,0)	11	13	(1,1,0,1)
10	10	(1,0,1,0)	10	15	(1,1,1,1)	10	5	(0,1,0,1)
9	9	(1,0,0,1)	9	13	(1,1,0,1)	9	9	(1,0,0,1)
8	8	(1,0,0,0)	8	12	(1,1,0,0)	8	1	(0,0,0,1)
7	7	(0,1,1,1)	7	4	(0,1,0,0)	7	14	(1,1,1,0)
6	6	(0,1,1,0)	6	5	(0,1,0,1)	6	6	(0,1,1,0)
5	5	(0,1,0,1)	5	7	(0,1,1,1)	5	10	(1,0,1,0)
4	4	(0,1,0,0)	4	6	(0,1,1,0)	4	2	(0,0,1,0)
3	3	(0,0,1,1)	3	2	(0,0,1,0)	3	12	(1,1,0,0)
2	2	(0,0,1,0)	2	3	(0,0,1,1)	2	4	(0,1,0,0)
1	1	(0,0,0,1)	1	1	(0,0,0,1)	1	8	(1,0,0,0)
0	0	(0,0,0,0)	0	0	(0,0,0,0)	0	0	(0,0,0,0)

Fig. 2. Labeling physical cell states with levels. (a) A straightforward labeling, where every physical state $s \in \{0, 1, \cdots, 15\}$ is labeled by level $\ell = s$. (b) A Gray-code labeling. (c) A new labeling.

In practice, physical errors of smaller magnitudes are usually more likely than larger ones. Let us focus on physical errors of magnitude one now, which are often the most probable errors. Given a vector $\mathbf{v} = (v_1, \cdots, v_k) \in \{0, 1\}^k$, its Hamming weight is defined as $w_H(\mathbf{v}) \triangleq \{i | i \in \{1, \cdots, k\}, v_i = 1\} = |support(\mathbf{v})|$. A physical error δ changes the physical cell state from s to $s' = s + \delta$, and the number of bit-errors it introduces is $W(s, \delta) \triangleq w_H(\mathcal{B}_m(\varepsilon \mod 2^m)) = w_H(\mathcal{B}_m(\ell' - \ell \mod 2^m)) = w_H(\mathcal{B}_m(\pi(s + \delta) - \pi(s) \mod 2^m))$. Let us call a labeling π *Order-one Optimal* if it minimizes the total number of bit-errors introduced by magnitude-one (including +1 and -1) physical errors. That is, it minimizes $W_{total} \triangleq \sum_{i=0}^{q-2} W(i, 1) + W(i+1, -1)$. In the following, we present such an optimal labeling.

Construction 8. A Method for cell-level labeling
Let $\pi(0) = 0$. For $i = 1, 2, \cdots, m$ and $j = 2^{i-1}, 2^{i-1} + 1, \cdots, 2^i - 1$, let $\pi(j) = \pi(j - 2^{i-1}) + 2^{m-i}$. □

Construction 8 generalizes Fig. 2 (c). (Actually, it can be further generalized to the case where for $i = 0, 1, \cdots, m$ and

$j = 0, 1, \cdots, 2^{m-i} - 1$, $\{\pi(j \cdot 2^i + k) | k \in \{0, 1, \cdots, 2^i - 1\}\} = \{k \cdot 2^{m-i} + z \mid k \in \{0, 1, \cdots, 2^i - 1\}\}$ for some $z \in \{0, 1, \cdots, 2^{m-i} - 1\}$.) We now prove that it is order-one optimal. For any $i \in \{-2^m + 1, \cdots, -2, -1, 1, 2, \cdots, 2^m - 1\}$, define $\chi(i) \triangleq \max\{j | j \in \{0, 1, \cdots, m-1\}, (i \mod 2^m)$ is a multiple of $2^j\}$. Note that in the binary representation of $(i \mod 2^m)$, – namely $\mathcal{B}_m(i \mod 2^m)$, – the $\chi(i)$ least significant bits are all 0s, and the $(\chi(i) + 1)$th least significant bit is 1. For convenience, let us define $W(2^m - 1, 1) \triangleq w_H(\mathcal{B}_m(\pi(0) - \pi(2^m - 1) \mod 2^m))$, and define $W(0, -1) \triangleq w_H(\mathcal{B}_m(\pi(2^m - 1) - \pi(0) \mod 2^m))$.

Lemma 9. *Given a labeling* $\pi : \{0, 1, \cdots, 2^m - 1\} \to \{0, 1, \cdots, 2^m - 1\}$, *for any* $i \in \{0, 1, \cdots, 2^m - 1\}$, *we have* $\chi(\pi(i+1 \mod 2^m) - \pi(i)) = \chi(\pi(i) - \pi(i+1 \mod 2^m))$ *and* $W(i, 1) + W(i+1 \mod 2^m, -1) = m - \chi(\pi(i+1 \mod 2^m) - \pi(i)) + 1$.

Corollary 10. *For* $i = 0, 1, \cdots, 2^m - 1$, *we have* $2 \le W(i, 1) + W(i+1 \mod 2^m, -1) \le m + 1$.

Lemma 11. *For* $j = 2, 3, \cdots, m$,

$$|\{i | 0 \le i \le 2^m - 1, W(i, 1) + W(i+1 \mod 2^m, -1) \le j\}| \le \sum_{k=m-1, m-2, \cdots, m-j+1} 2^k = 2^{m-j+1}(2^{j-1} - 1).$$

Theorem 12. *Construction* 8 *is an order-one optimal labeling.*

Sketch of proof: For $j = 2, 3, \cdots, m+1$, define $count(j) \triangleq |i | 0 \le i \le 2^m - 1, W(i, 1) + W(i+1 \mod 2^m, -1) = j|$. In Construction 8, $count(k) = 2^{m-k+1}$ for $k = 2, \cdots, m$, and $count(m+1) = 2$. It minimizes $\sum_{j=2}^{m+1} j \cdot count(j) - [W(2^m - 1, 1) + W(0, -1))] = W_{total}$. ■

IV. Coding for Any Number of Cell Levels

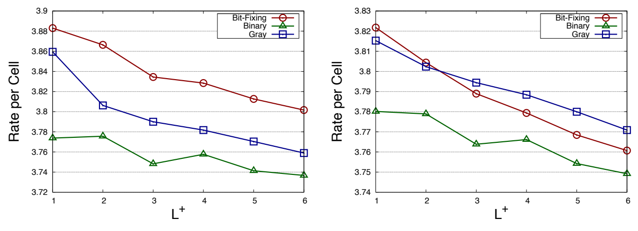
The bit-fixing scheme can be generalized to any q levels. For simplicity, we show the case where binary representations are used but q is not a power of 2. The idea is to first store data in n "virtual cells" of 2^m levels – where $m = \lceil \log_2 q \rceil$ – in the same way as before. Let $\ell^* \in \{0, 1, \cdots, 2^m - 1\}$ denote a virtual cell's level, and let $\ell \in \{0, 1, \cdots, q-1\}$ denote the corresponding real cell's level. Then if $\ell^* \ge q$, we let $\ell = \ell^* - 2^{m-1}$; otherwise, we let $\ell = \ell^*$. (This *linear folding* has the property that if $\ell \neq \ell^*$, their binary representations differ only in the MSB.) It can be shown that $\mathcal{C}_0, \cdots, \mathcal{C}_{m-2}$ can be decoded the same way as before. And for $\mathcal{C}_{m-1}$, for its codeword bits, the channel model is a combination of a partial erasure channel (corresponding to the linear folding) and noise. And it can be designed and decoded accordingly. Due to the space limitation, we skip the details.

V. Performance Evaluation

We have evaluated the performance of the bit-fixing scheme, and compared it to the commonly used basic binary-representation scheme and the Gray-code based scheme (which were introduced in Section I). It usually performs better than the former and is comparable to (and sometimes better than) the latter. In the following, we introduce one such comparison on achievable rates given the space limitation.

We consider errors of magnitude range $[-L^-, L^+]$, modeled as follows. Let there be $n \to \infty$ cells of $q = 2^m$ levels, whose errors are i.i.d. Let $p \in [0, 1]$ be a parameter, let L^+ and L^- be non-negative integers (with $L^+ + L^- > 0$), and let $\tilde{\delta} \in \{-L^-, \cdots, 0, \cdots, L^+\}$ be a random variable with this distribution: $Pr\{\tilde{\delta} = 0\} = 1 - p$; and $\forall\ i \in \{-L^-, \cdots, -1, 1, \cdots, L^+\}$, $Pr\{\tilde{\delta} = i\} = p/(L^- + L^+)$. For a cell of original physical state $s \in \{0, 1, \cdots, q-1\}$, the noise $\tilde{\delta}$ is added to it. If $\tilde{\delta} > 0$, the noisy physical level s' becomes $\min\{s + \tilde{\delta}, q-1\}$; otherwise, s' becomes $\max\{s + \tilde{\delta}, 0\}$. (It is modeled this way because a cell's state must be in $\{0, 1, \cdots, q-1\}$. And given a labeling π, the error changes the level from $\ell = \pi(s)$ to $\ell' = \pi(s')$.)

We consider the practical case where $Pr\{b_i = 0\} = 1/2$ for $i = 0, 1, \cdots, m-1$, for the reason that in most practical ECCs, codeword bits are (nearly) equally likely to be 0s and 1s. (This constraint can reduce achievable rates, however.) Some results on achievable rates are shown in Fig. 3, with $q = 16$, $p = 0.01$, and L^+ changing from 1 to 6. Fig. 3 (a) shows a case for asymmetric errors, where $L^- = 0$ and the bit-fixing scheme uses the simple labeling $\ell = s$. Fig. 3 (b) shows a case for bidirectional errors, where $L^- = 3$ and the bit-fixing scheme uses the labeling in Construction 8. It can be seen that the bit-fixing coding scheme compares favorably with the basic binary-representation scheme, and is comparable to the Gray-code based scheme.



(a) (b)

Fig. 3. Comparison of achievable rates (number of stored bits per cell). Here $q = 16$, $p = 0.01$, and L^+ increases from 1 to 6. (a) Asymmetric errors, where $L^- = 0$. (b) Bidirectional errors, where $L^- = 3$.

Acknowledgment

This work was supported in part by the NSF CAREER Award CCF-0747415 and an NSF-NRI award.

References

[1] R. Ahlswede, H. Aydinian and L. Khachatrian, "Unidirectional error control codes and related combinatorial problems," in *Proc. Eighth Int. Workshop Algebr. Combin. Coding Theory*, pp. 6-9, 2002.

[2] Y. Cassuto, M. Schwartz, V. Bohossian and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," in *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582–95, 2010.

[3] N. Elarief and B. Bose, "Optimal, systematic, q-ary codes correcting all asymmetric and symmetric errors of limited magnitude," *IEEE Trans. Inform. Theory*, vol. 56, no. 3, pp. 979–983, March 2010.

[4] T. Klove, B. Bose and N. Elarief, "Systematic single limited magnitude asymmetric error correcting codes," *Proc. ITW*, Cairo, Egypt, 2010.

[5] E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "On codes that correct asymmetric errors with graded magnitude distribution," in *Proc. ISIT*. pp. 1021–1025, St. Petersburg, Russia, August 2011.