A Constraint Scheme for Correcting Massive Asymmetric Magnitude-1 Errors in Multi-Level NVMs

Evyatar Hemo and **Yuval Cassuto**
Department of Electrical Engineering, Technion – Israel Institute of Technology
evyatar@tx.technion.ac.il, ycassuto@ee.technion.ac.il

Abstract— **We present a constraint-coding scheme to correct large numbers of asymmetric magnitude-1 errors in multi-level non-volatile memories. The scheme is shown to deliver better correction capability compared to known alternatives, while admitting low-complexity of decoding. Our results include an algebraic formulation of the constraint, necessary and sufficient conditions for correctability, a maximum-likelihood decoder running in complexity linear in the alphabet size, and lower bounds on the probability to correct t errors. Besides the superior rate-correction tradeoff, another advantage of this scheme over standard error-correcting codes is the flexibility to vary the code parameters without significant modifications.**

I. Introduction

Multi-level non-volatile memories (NVMs) hold great promise to sustain the continued scaling of mass-storage device capacities. In flash technology, a memory level is determined by the amount of electrical charge trapped inside a floating gate transistor. Equivalently, each memory level is represented by a certain value of voltage or current. However, due to a variety of factors such as noise, manufacturing variability, and inter-cell interference, it becomes very challenging to scale the number of levels while keeping the voltage dynamic range fixed. As a result, the gaps between memory levels become narrower, and a growing error rate compromises the data fidelity.

Errors resulting from narrowing margins between levels have been an active subject of research in recent years. In particular, errors with *asymmetry* and *magnitude limit* have emerged as useful coding models to tackle this problem. Asymmetry of the errors stems from the inherent asymmetry between the write and erase operations in flash, which often renders one error direction much more dominant than the other. The limited magnitudes of the errors come from "incremental" error sources such as retention errors (slow decay of charge level) or disturbs (gradual level increments from writes to adjacent cells) [1],[2].

As the density scaling progresses, we expect that *asymmetric, magnitude*-1 errors will be the first to become the dominant reliability issue of multi-level NVMs. Hence our objective is to develop a coding scheme that will deal with a large number of asymmetric magnitude-1 errors within a code block. The coding scheme we propose herein will be evaluated in comparison to known coding techniques for asymmetric magnitude-1 errors, which we now briefly survey. In the extreme scenario, we can use an *all asymmetric magnitude*-1 error correcting code, which is equivalent to using only the even cell levels out of the q supported levels [3]. A slightly less redundant code, correcting asymmetric magnitude-1 errors in up to *half* of the code coordinates, can be obtained by the construction of [4] when used with the binary repetition code. This option is equivalent to using only even levels or only odd levels within each codeword – hence termed *even/odd* in the sequel. When the code block length is not too large, we may also use the construction of [4] with lower-redundancy binary codes, such as *BCH codes*, to achieve a better tradeoff between rate and correctability. Finally, another alternative is to use codes coming from the recently developed theory for *L1 distance* codes reported in a sequence of papers by Bose, Tallini and others. See for example [5] and citations thereof. In the sequel we show that for moderate to high error rates, our coding scheme gives a better correctability-rate tradeoff compared to existing alternatives.

The proposed scheme, based on constraint coding, encodes data in a way that asymmetric magnitude-1 errors will be easily detected and corrected, but with milder restrictions than the all-errors correcting all-even code or the half-errors correcting even/odd code. The scheme is called *Non-Consecutive Constraint (NCC)*, attesting to the enforced constraint of disallowing the codeword to contain two adjacent levels from $\{0, \ldots, q-1\}$. The NCC scheme has three main advantages:

1) Performance: the NCC outperforms alternative coding schemes for random errors, when the error rate is moderate to high.
2) Complexity: the decoder of the NCC has low complexity, and achieves the performance of the maximum likelihood decoder.
3) Flexibility: the tradeoff between the redundancy of the code and its error correction capabilities can be easily adjusted without re-designing the code or altering the decoder and encoder.

Section II describes common coding schemes for asymmetric errors, Section III presents the NCC coding scheme and its algebraic formulation. A maximum-likelihood, low complexity decoding algorithm for the NCC is presented in Section IV, and detailed performance analysis appears in Section V.

II. Asymmetric Errors with Magnitude 1

As described in the Introduction, we assume throughout the paper that all errors are asymmetric with magnitude 1, in the *downward* direction. A general recipe for converting a standard error-correcting code to a higher-rate version of the code designed for asymmetric errors with magnitude-1 can be found in [4]. The idea behind [4] in a nutshell is that in order to protect an n-cell q-ary memory block from asymmetric errors with magnitude-1, we can convert the block to its binary form, encode only the LSBs with a standard binary error correcting code, and map back to the q-ary form. By using this method, it is possible to use well-known codes and achieve very high-rate coding schemes. When using this recipe with the binary repetition code we get a q-ary code where in each codeword either all levels are even or all are odd. This scheme allows the correction of $\left\lfloor \frac{n-1}{2} \right\rfloor$ asymmetric magnitude-1 errors. Later in the paper we refer to this coding scheme as the **even/odd** code.

Example 1. *The following $q = 8$, $n = 4$ words $(2,2,4,0)$ and $(1,7,3,1)$ are legitimate even/odd codewords. However, $(2,2,4,1)$ is not a legitimate codeword since it contains both odd and even values. Notice that if an asymmetric error with magnitude-1 occurs in the first coordinate of the first codeword we get $(1,2,4,0)$. After reading such a word we can definitely deduce that an error occurred. There are three even values and a single odd value, therefore, by taking the majority it is clear that the error occurred in the first coordinate and the original codeword can be successfully restored.*

Another simple coding scheme is the **all-even** code [3], which is obtained by using only the even levels from the alphabet, and which can correct n asymmetric magnitude-1 errors.

III. The Non-Consecutive Constraint (NCC)

We now present the **Non-Consecutive Constraint** (NCC) coding scheme, which is specifically designed to correct large numbers of asymmetric errors with magnitude-1. Let the state of the memory cell be represented as a discrete cell level c, taken from the integer set $\{0,\ldots,q-1\}$.

Definition 1. *The NCC(n,q) is a constraint in which every q-level, n-cell memory block does not contain cells with consecutive memory levels.*

Example 2. *The following $q = 8$, $n = 8$ codeword $(2,5,7,0,2,0,4,4)$ is not an NCC codeword (because $4,5$ are consecutive levels and both appear in the codeword). In contrast, $(2,4,4,0,2,0,4,7)$ is an NCC codeword.*

A. Information rate

Definition 2. *The* information rate *$\mathcal{R}$ of a code for n-cell, q-level memory array is defined as*

$$\mathcal{R} = log_q(M)/n,$$

where M is the number of legal combinations according to the code specification. $\mathcal{R}$ represents the number of information q-ary symbols stored per physical cell.

Theorem 1. *The information rate of the NCC(n,q) is given by*

$$\mathcal{R}_{NCC}(n,q) = \log_q\left[\sum_{k=1}^{\frac{q}{2}} k!\cdot S(n,k)\cdot\binom{q-k+1}{k}\right]/n, \quad (1)$$

where $S(n,k)$ is the Stirling number of the second kind [6].

Proof: The number of combinations to occupy k non-consecutive levels out of q levels is $\binom{q-k+1}{k}$ [6](Ch.13). By multiplying it by the number of surjections from n-cell set to k-level set we obtain the desired combinations count. The number of surjections from an n-set to a k-set equals $k!\cdot S(n,k)$ [7]. Summing all possible values of k gives exactly the expression inside the log in (1). ■

In order to compare between the NCC and the even/odd and all-even coding schemes, we now mention the information rates for the other methods. Due to the fact that for an even/odd codeword all n cells are either all even or all odd, the number of possible combinations is given by $M = 2\cdot(q/2)^n$. Therefore, the information rate of the n-cell, q-level even/odd code is given by

$$\mathcal{R}_{even/odd}(n,q) = 1 - \frac{n-1}{n}\log_q 2. \quad (2)$$

The all-even code has an information rate that is independent of n, and is equal to $1 - log_q 2$. As a result, unlike the NCC and even-odd codes above, it is not possible to change the rate-correction tradeoff of all-even by varying its block size.

B. Algebraic formulation of the NCC

In order to better understand and analyze the NCC coding scheme, we turn to a more formal algebraic definition of the constraint. In the sequel, inequalities involving vectors are interpreted element-wise.

Definition 3. *Given a vector of cell levels $\boldsymbol{c} = (c_1,\ldots,c_n)$, with $c_i\in\{0,\ldots,q-1\}$, define the* **histogram vector** *as the length q vector $\boldsymbol{h}$ in which the i-th element $i\in\{0,\ldots,q-1\}$ is the number of cells that occupy level i. That is, $h_i = |\{s\in\{1,\ldots,n\}|c_s = i\}|$.*

Example 3. *For the following $q = 8$, $n = 8$ cell-level vector $\boldsymbol{c} = (2,2,4,5,2,2,4,0)$, we have $\boldsymbol{h} = [1,0,4,0,2,1,0,0]$.*

In the remainder of the section we discuss only histogram vectors and not cell-level vectors, but keep in mind that there is a many-to-one relation between cell-level vectors and histogram vectors.

Proposition 2. *Let $\boldsymbol{h}$ be a histogram vector of some memory word. $\boldsymbol{h}$ is a histogram of a legal NCC codeword if and only if it satisfies*

$$diag(\boldsymbol{h})\,M\boldsymbol{h} = 0, \quad (3)$$

where $diag(\boldsymbol{h})$ stands for a square diagonal matrix with the elements of vector $\boldsymbol{h}$ on its main diagonal, and

$$M = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (4)$$

Proof: $M\boldsymbol{h}$ is the shifted histogram $[h_1,h_2,\ldots,h_{q-1},0]$, and $diag(\boldsymbol{h})\,M\boldsymbol{h}$ is the scalar multiplication of the vectors $\boldsymbol{h}$ and $M\boldsymbol{h}$. Therefore, we get

$$diag(\boldsymbol{h})\,M\boldsymbol{h} = \left[h_0\cdot h_1, h_1\cdot h_2,\ldots,h_{q-2}\cdot h_{q-1},0\right], \quad (5)$$

and requiring this vector to be the all 0 vector is equivalent to having no two consecutive occupied levels. ■

Let $\boldsymbol{h}$ be a histogram vector of some memory word, and let $\boldsymbol{e}$ be an *error* histogram vector with all non-negative elements, whose i-th element specifies how many memory cells moved from level i to level $i-1$ due to asymmetric magnitude-1 errors. The effect of each such error is decrementing h_i and incrementing h_{i-1}. As a result, we get that the received histogram $\boldsymbol{w}$ is

$$\boldsymbol{w} = \boldsymbol{h} + T\boldsymbol{e}, \quad (6)$$

where

$$T = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & \cdots & -1 \end{pmatrix}_{q\times q}, T^{\dagger} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & -1 & -1 & \cdots & -1 \\ 0 & 0 & -1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}_{q\times q}, \quad (7)$$

and the matrix $T^{\dagger}$ represents the inverse action of T because $T^{\dagger}T$ gives the identity mapping on length q vectors of the form $(0, u_1, \ldots, u_{q-1})$, which is exactly the form of error histogram vectors $\boldsymbol{e}$. Note that by definition $\boldsymbol{e}$ should satisfy $\boldsymbol{e} \leqslant \boldsymbol{h}$.

Definition 4. *Given two histogram vectors $\boldsymbol{h_1}, \boldsymbol{h_2}$ we define their* **downward difference vector** $\boldsymbol{d}$ *and their* **upward difference vector** $\hat{\boldsymbol{d}}$ *as follows*

$$\boldsymbol{d}(\boldsymbol{h_1}, \boldsymbol{h_2}) = \frac{T^{\dagger}(\boldsymbol{h_2} - \boldsymbol{h_1}) + |T^{\dagger}(\boldsymbol{h_2} - \boldsymbol{h_1})|}{2},$$
$$\hat{\boldsymbol{d}}(\boldsymbol{h_1}, \boldsymbol{h_2}) = -\frac{T^{\dagger}(\boldsymbol{h_2} - \boldsymbol{h_1}) - |T^{\dagger}(\boldsymbol{h_2} - \boldsymbol{h_1})|}{2}. \quad (8)$$

$|\cdot|$ represents element-wise absolute value. Notice that the elements of $\boldsymbol{d}$ and $\hat{\boldsymbol{d}}$ are non-negative. In addition, $\boldsymbol{d} - \hat{\boldsymbol{d}} = T^{\dagger}(\boldsymbol{h_2} - \boldsymbol{h_1})$. We now move to define a distance function for histograms.

Definition 5. *Given two histogram vectors $\boldsymbol{h_1}, \boldsymbol{h_2}$ we define their* **downward distance** $\mathcal{D}^{\downarrow}$ *and their* **upward distance** $\mathcal{D}^{\uparrow}$ *as follows*

$$\mathcal{D}^{\downarrow}(\boldsymbol{h_1}, \boldsymbol{h_2}) = ||\boldsymbol{d}(\boldsymbol{h_1}, \boldsymbol{h_2})||,$$
$$\mathcal{D}^{\uparrow}(\boldsymbol{h_1}, \boldsymbol{h_2}) = ||\hat{\boldsymbol{d}}(\boldsymbol{h_1}, \boldsymbol{h_2})||. \quad (9)$$

$||\cdot||$ stands for the ℓ_1 vector norm.

A tool to analyze and design codes is the measure of t-confusability between histograms.

Definition 6. *The histogram $\boldsymbol{h_1}$ is* ***t*-confusable** *as $\boldsymbol{h_2}$ if after suffering t errors (or less) a maximum-likelihood decoder returns $\boldsymbol{h_2}$.*

Note that the definition of t-confusability does not mean that *every* pair of cell-level vectors with the corresponding histograms is t-confusable. But it does mean that there exists at least one such pair of cell-level vectors that can confuse the decoder in the case of t errors or less.

Proposition 3. *$\boldsymbol{h_1}$ is t-confusable as $\boldsymbol{h_2}$ if and only if the following two conditions hold*

1) $\boldsymbol{d}(\boldsymbol{h_1}, \boldsymbol{h_2}) \leqslant \boldsymbol{h_1}$, $\hat{\boldsymbol{d}}(\boldsymbol{h_1}, \boldsymbol{h_2}) \leqslant \boldsymbol{h_2}$,
2) $\mathcal{D}^{\uparrow}(\boldsymbol{h_1}, \boldsymbol{h_2}) \leqslant \mathcal{D}^{\downarrow}(\boldsymbol{h_1}, \boldsymbol{h_2}) \leqslant t$.

Proof: If $\boldsymbol{h_1}$ can be confused as $\boldsymbol{h_2}$ after suffering t errors, then there exist two error histogram vectors $\boldsymbol{e} \leqslant \boldsymbol{h_1}$ and $\boldsymbol{f} \leqslant \boldsymbol{h_2}$ such that $||\boldsymbol{f}|| \leqslant ||\boldsymbol{e}|| \leqslant t$ that satisfy $\boldsymbol{h_1} + T\boldsymbol{e} = \boldsymbol{h_2} + T\boldsymbol{f}$. By reordering and multiplying by $T^{\dagger}$ we get that $T^{\dagger}(\boldsymbol{h_2} - \boldsymbol{h_1}) = \boldsymbol{e} - \boldsymbol{f}$. It follows from Definition 4 that $\boldsymbol{d}(\boldsymbol{h_1}, \boldsymbol{h_2}) + \boldsymbol{c} = \boldsymbol{e}$, $\hat{\boldsymbol{d}}(\boldsymbol{h_1}, \boldsymbol{h_2}) + \boldsymbol{c} = \boldsymbol{f}$ for some non-negative vector $\boldsymbol{c}$, and conditions 1 and 2 hold by the properties of $\boldsymbol{e}, \boldsymbol{f}$.
For the other direction we assume that the conditions are met and examine the vectors $\boldsymbol{w} = \boldsymbol{h_1} + T\boldsymbol{d}(\boldsymbol{h_1}, \boldsymbol{h_2})$ and $\boldsymbol{w'} = \boldsymbol{h_2} + T\hat{\boldsymbol{d}}(\boldsymbol{h_1}, \boldsymbol{h_2})$. By subtracting the equations and multiplying by $T^{\dagger}$ we get that $T^{\dagger}(\boldsymbol{w'} - \boldsymbol{w}) = \boldsymbol{0}$. We can conclude that $\boldsymbol{w'} = \boldsymbol{w}$ because the only solutions to the linear system $T^{\dagger}\boldsymbol{x} = \boldsymbol{0}$ are of the form $[x_0, 0, \ldots, 0]$, which is a contradiction because $\boldsymbol{w}$ and $\boldsymbol{w'}$ sum to the same value n. Hence the received histogram $\boldsymbol{w} = \boldsymbol{w'}$ will confuse the decoder when $\boldsymbol{h_1}$ is stored and the error is $\boldsymbol{d}(\boldsymbol{h_1}, \boldsymbol{h_2})$. ■

IV. Decoding Algorithm for the NCC

In this section we describe a decoding algorithm for the NCC. The objective is a maximum-likelihood decoding algorithm that finds the nearest codeword to the received memory word, in the sense that a minimal number of +1 cell transitions map between the memory word and the output codeword. The strengths of the proposed decoding algorithm are that it guarantees to return the nearest codeword, is invariant to the code block length n, and has very low $O(q)$ complexity. To present the algorithm, we first give several formal definitions.

Definition 7. *For a histogram vector $\boldsymbol{h}$, we define a* **section** *$\mathcal{S}(\boldsymbol{h})$ as a minimal length sub-vector of $\boldsymbol{h}$ that is separated from other occupied levels by at least two empty levels. $N_{\mathcal{S}}(\boldsymbol{h})$ is the number of sections in $\boldsymbol{h}$. This definition implies that the decoding of each section can be done independently of other sections.*

Definition 8. *A* ***burst*** *$b(\boldsymbol{h})$ is defined as a maximal length sub-vector in a histogram $\boldsymbol{h}$ with all its entries greater than zero. $N_b(\boldsymbol{h})$ is the number of bursts in $\boldsymbol{h}$.*

$\mathcal{S}_l(\boldsymbol{h})$ represents the l-th lowest section of a certain histogram vector $\boldsymbol{h}$. We drop the $\boldsymbol{h}$ notation when it is clear to which histogram vector the section refers to. $b_j(\mathcal{S}_l)$ represents the j-th lowest burst in section $\mathcal{S}_l$, and for simplicity of notation, we denote $b_j(\mathcal{S}_l) \equiv b_j^l$.

Example 4. *For the following $q = 12$, $n = 15$ histogram vector $\boldsymbol{h} = (2, 0, 0, 1, 3, 1, 0, 0, 1, 2, 0, 5)$, there are three sections: $\mathcal{S}_1(\boldsymbol{h}) = [2]$, $\mathcal{S}_2(\boldsymbol{h}) = [1, 3, 1]$ and $\mathcal{S}_3(\boldsymbol{h}) = [1, 2, 0, 5]$. The sections $\mathcal{S}_1$ and $\mathcal{S}_2$ contain only one burst each, and $\mathcal{S}_3$ has two bursts: $b_1^3 = b_1(\mathcal{S}_3) = [1, 2]$ and $b_2^3 = b_2(\mathcal{S}_3) = [5]$.*

From Definition 8, it is obvious that any burst of two levels or more must violate the NCC. Therefore, the purpose of the decoding algorithm is to resolve all the NCC violations expressed as a set of bursts. As a consequence, a decoded codeword must not contain a burst of two levels or more. An NCC violation in a burst can be resolved by up to two options: either all the burst occupations of odd levels must be moved up by one to the even ones, or all the burst occupations of even levels must be moved up to the odd ones. As will be explained further on, for some scenarios, only one of the movement options is possible.

Definition 9. *We define the* **movement** *that resolves the NCC violation induced by burst b as either $\sigma(b)$, when keeping the highest level of b intact, or $\bar{\sigma}(b)$, when the occupation of the highest level of b is moved up.*

Definition 10. *For a movement of levels $\pi(b)$, the* **cost** *of this movement $C(\pi(b))$ is defined as the number of cells whose level is moved in this action. When $\pi(b)$ includes the movement of cells occupying the level $q-1$ (which is impossible), we define $C(\pi(b)) = \infty$.*

During the decoding process, the different combinations for resolving bursts are determined in order to achieve minimal cost, hence, finding the nearest NCC codeword.

Example 5. *For $q = 10$, $n = 12$, let us assume we read a codeword with the histogram $\boldsymbol{h} = (0, 4, 2, 0, 0, 1, 0, 0, 3, 2)$. From the existence of two-level bursts, it is clear that this memory word is corrupt by some errors. All three sections contain only one burst each; the burst of the first section is*

$b_1^1 = [4,2]$. *So in order to resolve this violation, it is possible to perform a* $\bar{\sigma}\left(b_1^1\right)$ *movement (which means we move the* 2 *cells at level* 2 *upward) and obtain* $[4,0,2]$. *Alternatively, we can perform a* $\sigma\left(b_1^1\right)$ *movement (moving the* 4 *cells at level* 1 *upward), and obtain* $[0,6,0]$. *The costs of these two movement options are* 2 *and* 4, *respectively, so in order to minimize the cost, it is obvious that* $\bar{\sigma}\left(b_1^1\right)$ *is preferred. The burst of the second section is* $b_1^2 = [1]$. *In case this cell endured an error, it cannot be decoded successfully. The burst of the last section* $b_1^3 = [3,2]$ *has only one correction option* $\sigma\left(b_1^3\right)$, *because level* $q-1$ *cannot move upward. The decoded histogram is therefore* $\boldsymbol{h'} = (0,4,0,2,0,1,0,0,0,5)$.

The decoding processes can work on each section independently, but multiple bursts within a section need to be decoded jointly. The decoder is thus specified below through the main iteration on sections in Algorithm 1, and with the decoding of an individual section as a dynamic-programming (Viterbi-like) subroutine in Algorithm 2. The algorithm starts with the lowest burst, and for every burst j and a movement $\pi \in \{\sigma, \bar{\sigma}\}$, it builds a concatenated cost table $\mathcal{W}_j(\pi)$, which holds the minimal cost of movements for resolving all bursts up to burst j. The algorithm also stores the movement path Π_j which holds the sequence of movements with the optimal cost. During the burst iteration we distinguish between two cases: in odd-length bursts not moving the high level of burst j necessitates not moving the high level of burst $j-1$, and in even-length bursts *moving* the high level of burst j necessitates not moving the high level of burst $j-1$. The decoding process ends with the highest burst that determines which movement is possible for the previous burst (which was already solved). It is clear that the decoding algorithm has $O(q)$ complexity and uses two tables of size $2\sum_{l=1}^{N_S} N_b(\mathcal{S}_l)$ which is smaller than $2q$.

Algorithm 1: DecodeHistogram

```
input  : $\boldsymbol{h}$
output: $\boldsymbol{h'}$
for $l = 1$ to $N_S(\boldsymbol{h})$
    $actionVec$ =DecodeSection($\mathcal{S}_l(\boldsymbol{h})$)
    fix $\boldsymbol{h} \Rightarrow \boldsymbol{h'}$ by applying $actionVec$ on all $b_j \in \mathcal{S}_l(\boldsymbol{h})$
end
```

Algorithm 2: DecodeSection

```
input  : $\mathcal{S}$
output: $actionVec$
//initialization:
$\mathcal{W}_1(\sigma) = C(\sigma(b_1))$, $\mathcal{W}_1(\bar{\sigma}) = C(\bar{\sigma}(b_1))$
$\Pi_1(\sigma) = \Pi_1(\bar{\sigma}) = \emptyset$
//tables fill
for $j = 2$ to $N_b(\mathcal{S})$
    $\alpha = argmin_{\{\sigma,\bar{\sigma}\}}\left(\mathcal{W}_{j-1}(\sigma), \mathcal{W}_{j-1}(\bar{\sigma})\right)$
//for odd-length bursts:
    $\mathcal{W}_j(\sigma) = C\left(\sigma\left(b_j\right)\right) + \mathcal{W}_{j-1}(\sigma)$
    $\Pi_j(\sigma) = \left[\Pi_{j-1}(\sigma), \sigma\right]$
    $\mathcal{W}_j(\bar{\sigma}) = C\left(\bar{\sigma}\left(b_j\right)\right) + \min\left(\mathcal{W}_{j-1}(\sigma), \mathcal{W}_{j-1}(\bar{\sigma})\right)$
    $\Pi_j(\bar{\sigma}) = \left[\Pi_{j-1}(\alpha), \alpha\right]$
//for even-length bursts:
    $\mathcal{W}_j(\sigma) = C\left(\sigma\left(b_j\right)\right) + \min\left(\mathcal{W}_{j-1}(\sigma), \mathcal{W}_{j-1}(\bar{\sigma})\right)$
    $\Pi_j(\sigma) = \left[\Pi_{j-1}(\alpha), \alpha\right]$
    $\mathcal{W}_j(\bar{\sigma}) = C\left(\bar{\sigma}\left(b_j\right)\right) + \mathcal{W}_{j-1}(\sigma)$
    $\Pi_j(\bar{\sigma}) = \left[\Pi_{j-1}(\sigma), \sigma\right]$
end
//making final decision
$\alpha = argmin_{\{\sigma,\bar{\sigma}\}}\left(\mathcal{W}_{N_b(\mathcal{S})}(\sigma), \mathcal{W}_{N_b(\mathcal{S})}(\bar{\sigma})\right)$
$actionVec = [\Pi_{N_b}(\alpha), \alpha]$
```

V. Performance Analysis

In this section we wish to evaluate the error-correcting performance of the NCC code. An exact calculation of the correction probability is challenging in general, so we derive lower bounds. We also show experimentally that the NCC code gives superior performance to known coding alternatives for the same error model.

A. Experimental results

Table I presents the correction performance of the NCC code for $q = 8$. The two left columns include the code length n and its corresponding information rate. The rest of the table presents the probability to fully-correct a specified number of uniformly distributed asymmetric magnitude-1 errors. As can be seen in Table I, the correction capability improves as n increases (but the rate decreases). Moving between rows in the table to different rate-correction tradeoffs is done flexibly without need to change the decoder. Note that there is no guarantee that given a certain number of errors it can always be fully corrected. Whether a certain error can be corrected depends on both the codeword and the error pattern. What does not show in the table is that even in instances that did not fully correct, many of the errors were indeed corrected. In fact, this property will be next shown to result in superior performance of NCC. The way we compare the codes' performance is by measuring the symbol-error rate (SER) at the decoder output for a given rate of asymmetric magnitude-1 errors at the input. In Fig. 1 we plot the output SER for three codes: even/odd, $BCH(15,5,7)$ and length 7 NCC (plus a no-coding option), with equal rate of 0.777. The quoted BCH code is a binary code that is used in the construction of [4] to get asymmetric magnitude-1 correction. The x-axis is the input SER, which is the probability that a symbol exhibits an asymmetric magnitude-1 error. From Fig. 1 one can notice that the NCC code outperforms all the other codes with equal rate, for input SER values around 0.2 and higher. The SER remaining at the decoder output will be taken care of by a standard ECC, which in the case of NCC will require less redundancy because of the lower residual error rates. Note that the short lengths of the NCC code blocks do not limit the size of the flash word lines. Long word-lines can supported

TABLE I

FULL-CORRECTION PROBABILITY FOR THE n-CELL 8-LEVEL NCC

$q = 8$		Total number of errors $\|\|\boldsymbol{e}\|\|$					
n	$\mathcal{R}_{NCC}$	1	2	3	4	5	6
5	0.816	0.801	0.478	0.170	0.043	0.007	0
9	0.752	0.967	0.908	0.805	0.635	0.384	0.193
13	0.726	0.993	0.981	0.960	0.927	0.869	0.777
17	0.712	0.998	0.995	0.990	0.983	0.971	0.952

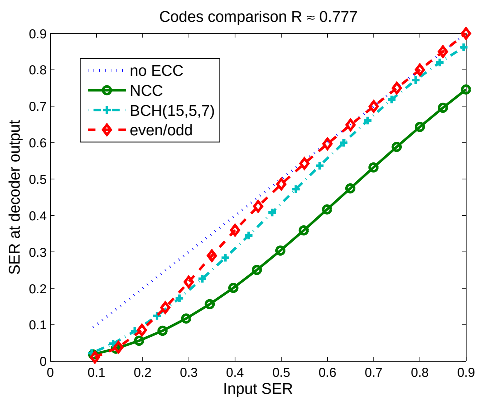

Figure 1. Correcting capabilities as a function of media SER for information rate of 0.777 and $q = 8$. No code (dotted), NCC (solid & circles), BCH(15,5,7) (dashed & diamonds), even-odd (semi-dashed & pluses)

by concatenating multiple NCC codewords

B. Bounds on performance

Theorem 4. *Given that $t \leqslant \left\lfloor \frac{n}{2} \right\rfloor - 2$ uniformly selected errors occurred in a uniformly selected NCC(n,q) codeword, the probability $\mathcal{P}_t$ to successfully correct **all** t errors is bounded from below by*

$$1 - \frac{\sum_{k=1}^{q/2} k! F_{2t+1} \cdot \left[\binom{q-k+1}{k} - 1 \right] \cdot \frac{2k}{n-2} \cdot \frac{t^t (1-2/n)^t}{(t-1)!}}{M} \leqslant \mathcal{P}_t, \qquad (10)$$

where $M = q^{n\mathcal{R}_{NCC}(n,q)}$ is the number of n-cell, q-level NCC codewords, and

$$F_{2t+1}(n,k) = S(n,k) - S_{2t+1}(n,k), \qquad (11)$$

where $S_r(n,k)$ are the r-associated Stirling numbers of the second kind [8]. (The combinatorial interpretation of $S_r(n,k)$ is the number of partitions of an n-set to k subsets, all of which are of cardinality at least r).

Proof: In order to derive this bound we first need to bound from above the number of codewords that cannot always be successfully decoded after enduring t errors ($t \leqslant \left\lfloor \frac{n}{2} \right\rfloor - 2$). It is clear that if a certain memory level is occupied by less than $2t + 1$ cells, it cannot be always corrected since it is possible that all t errors occur in this specific level, and we will not be able to choose which of two adjacent levels we should move. In the worst case when this level is a single-level section, decision by majority would lead to errors. $F_{2t+1}(n,k)$ is the number of partitions of an n-set to k subsets in which at least one subset is of cardinality less than $2t + 1$. Recall the expression for counting NCC codewords (1), exchanging $S(n,k)$ with $F_{2t+1}(n,k)$ and subtracting a single combination of levels from $\binom{q-k+1}{k}$ (since the level-set $\{0, 2, 4, \ldots, 2k-2\}$ can always correct t errors regardless of the assignment of n cells to the k levels) give

$$\sum_{k=1}^{q/2} k! F_{2t+1} \cdot \left[\binom{q-k+1}{k} - 1 \right], \qquad (12)$$

which is an upper bound on the number of codewords that cannot always be successfully decoded after enduring t errors. Next, we multiply the expression inside the summation in (12) with an upper bound on the probability over error vectors that the error cannot be corrected. When k levels are occupied, there are at most k levels that can endure uncorrectable errors. Each such potentially uncorrectable level can contain at most $h_i = 2t$ cells, and can cause an unsuccessful decoding when enduring $h_i/2$ errors or more. So, in order to calculate this error probability, we need to count all the error patterns which can cause uncorrectable errors in and divide it with the number of all possible error patterns. Then, given that $t \leqslant \left\lfloor \frac{n}{2} \right\rfloor - 2$, we bound this probability (proof omitted) in order to get an expression that does not depend on h_i. Altogether we have

$$k \cdot \sum_{j=0}^{h_i/2} \binom{h_i}{h_i/2 + j} \binom{n - h_i}{t - h_i/2 - j} \leqslant \frac{2kt^t}{(t-1)!} \cdot \left(1 - \frac{2}{n}\right)^t \cdot \frac{1}{n-2}, \qquad (13)$$

Combining equations (12) and (13), dividing by M and subtracting from 1 give us the expression for the lower bound in (10). ■

For the case in which a single error occurs, it is possible to derive the asymptotic behavior for the error probability when n is large.

Theorem 5. *For n-cell, q-level NCC, when q is fixed and $n \to \infty$, the probability $\mathcal{P}_{t=1}$ to correct a single error is bounded by*

$$\mathcal{P}_1 \geqslant 1 - \frac{q^2}{q+2} \left(1 - \frac{2}{q}\right)^{n-1}. \qquad (14)$$

VI. Conclusion

The NCC flexible coding scheme for asymmetric magnitude-1 errors was presented. An optimal, low complexity decoding algorithm was introduced. The NCC was also analyzed and compared to BCH and even/odd codes, outperforming these codes for moderate to high values of SER.

VII. Acknowledgment

This work was supported by a Marie Curie CIG grant, by the Israel Ministry of Science and Technology, and by a GIF Young Investigator grant.

References

[1] N. Mielke, et al., “Bit error rate in NAND flash memories,” Reliability Physics Symposium, IRPS 2008, IEEE International, pp.9-19, 2008.
[2] Y. Cai, et al., “Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis,” Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012. IEEE, 2012.
[3] R. Ahlswede, H. Aydinian, and L. Khachatrian, “Unidirectional error control codes and related combinatorial problems,” in Proc. of the Eighth International Workshop on Algebraic and Combinatorial Coding Theory (ACCT-8), St. Petersburg, Russia ,pp.6-9, 2002.
[4] Y. Cassuto , M. Schwartz , V. Bohossian and J. Bruck, “Codes for asymmetric limited-magnitude errors with application to multi-level flash memories,” IEEE Trans. Inf. Theory, vol. 56, no. 4, pp.1582 -1595, 2010.
[5] L.G. Tallini and B. Bose, “On L 1-distance error control codes,” in Proc. of IEEE Int. Symp. on Information Theory, ISIT, pp.1061-1065, July 2013.
[6] J. van Lint and R. Wilson, *A Course in Combinatorics, second edition*. Cambridge UK: Cambridge University Press, 2001.
[7] A. Mohr and T.D. Porter, “Applications of Chromatic Polynomials Involving Stirling Numbers, ” Department of Mathematics Southern Illinois University, 2008.
[8] L. Comtet, *Advanced Combinatorics*. Reidel, Dordrecht, 1974.
[9] L. Howard, “Associated Stirling numbers, ” The Fibonacci Quarterly 18, pp.303-315, 1980.