Tackling Intracell Variability in TLC Flash Through Tensor Product Codes

Ryan Gabrys*, Eitan Yaakobi†‡, Laura Grupp†, Steven Swanson†, Lara Dolecek*
*University of California, Los Angeles †University of California, San Diego ‡California Institute of Technology

***Abstract*—Flash memory is a promising new storage technology. To fully utilize future multi-level cell Flash memories, it is necessary to develop error correction coding schemes attuned to the underlying physical characteristics of Flash. Based on a careful inspection of fine-grained, experimentally-collected error patterns of TLC (three bits per cell) Flash, we propose a mathematical model that captures the intracell variability, which is manifested by certain patterns of bit-errors. Error correction codes are constructed for this model based upon generalized tensor product codes. For fixed levels of redundancy, these codes are shown to exhibit substantially lower bit error rates than existing error correction schemes.**

I. Introduction

Flash memory devices can be found almost everywhere today. They are lighter, faster and more shock resistant than traditional magnetic hard drives. As this technology scales and the storage density increases, data errors become more prevalent, making error correction coding critical for maintaining data integrity.

The storage density of a Flash memory device is dependent on the number of discrete voltage levels the floating gate cell is capable of representing. In early generations, every memory cell could represent two voltage levels and thus store a single bit (SLC). The demand for increased storage capacity has created the need to store more than a single bit per cell by simply representing more than two voltage levels. In this work, we follow the commonly adopted nomenclature and assume that multiple level cell (MLC) chips store two bits per cell, and that triple level cell (TLC) chips store three bits per cell.

Recently, the subject of error-correction coding for Flash memory has received significant attention. In [5], trellis coded modulation techniques were applied to Flash memory. In [8], the use of non-binary LDPC codes was investigated. In [6], algebraic error-correction codes were used for rewriting as well as correcting errors. In [1], [4], codes that correct limited magnitude asymmetric errors were constructed. In [12], this model was extended to correct graded error patterns.

The error model in this work is motivated by data collected from a TLC Flash device. As observed in [13], if the information from each Flash cell is interpreted as a triple-bit word, then the errors largely cause only a single bit in each word to change. From this observation, we suggest the use of a new class of codes derived from tensor product codes [7], [11] in the context of Flash memory. This work generalizes the result of [13] to correct errors that mostly have only a small number of bits in error for each cell-error. The technique used to address the problem is based on the generalized tensor product (GTP) scheme proposed in [7].

Tensor product codes were first introduced in [11] and were generalized to produce efficient binary codes in [7]. More recently, tensor product codes were used in the context of magnetic recording [2], [3]. In a concatenated coding scheme, the use of a tensor product parity code as the inner code was shown to offer the performance advantages of a short length parity code but without the associated rate penalty. In this work, it is shown that generalized tensor product codes can be used to efficiently correct the errors that occur within a TLC Flash device, and in turn extend the lifetime of a memory system. The main contributions are construction methods for codes that correct up to t_1 symbol errors with up to l_1 bit errors and t_2 symbol errors with up to l_2 bit errors.

In Section II, the data collected from a TLC Flash chip is summarized. In Section III, the error model, motivated by the experimental data, is proposed. In Section IV, code constructions for this model are given. In Section V, these constructions are shown through simulation to be superior to commonly used storage codes. Section VI concludes the paper.

II. Structure and Error Characterization of TLC Flash

In this section, we report on the observed errors measured from a TLC chip provided by an anonymous vendor. A TLC chip is divided into multiple planes. Each plane is divided into a set of blocks and these blocks are further decomposed into pages. For the particular TLC chip measured, there are 384 pages within a block and 8 kilobytes (KB) within a page. The eight discrete voltage levels from the cell are represented as a triple-bit word. We refer to the first bit in the word as the most significant bit (MSB), the second bit in the word as the center significant bit (CSB), and the third bit in the word as the least significant bit (LSB). For more details on the structure of a TLC chip, see [13].

The errors were measured from sixteen blocks evenly divided across two planes. The following testing procedure was repeatedly performed. On the first cycle of every 100 program/erase (P/E) cycles, a block was erased, and random data was then written and finally read back for errors. On the other 99 cycles, the block was simply erased and the memory was programmed to simulate the aging of the device.

In Figure 1, the Bit Error Rate (BER) is illustrated for the TLC chip tested over the course of its lifetime. It can be seen that over time, the BER increases dramatically but at different rates depending on which bit is programmed. The 'Symbol Error Rate' plot refers to the symbol error rate when each cell is represented as a symbol over $GF(8)$.

The dominant trend from Figure 1 is that the 'Symbol Error Rate' appears to be roughly the sum of the individual BERs

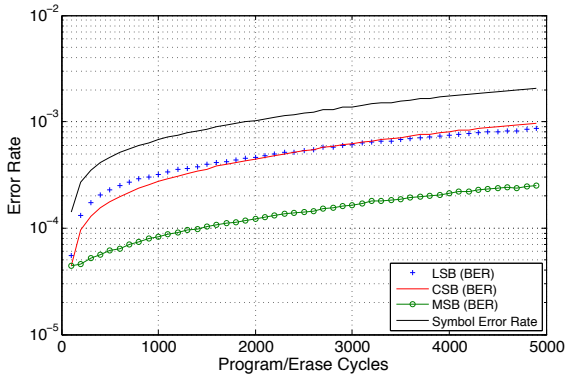

Fig. 1. Error Rates Measured from a TLC Flash Device.

of the MSB, CSB, and LSB. This suggests that whenever a cell-error occurs, with high probability only one of the three bits in the cell errs. More specifically, 96.17% of cell-errors only had a single bit in error. This is a result of the special programming property of the bits where the three bits are not programmed all at once. More details on this phenomena are reported in [13]. Note that this error model is considerably different than the one of asymmetric limited magnitude errors, studied in many previous works, e.g., [1] and [4].

The new codes introduced in this paper correct errors that mostly affect a single bit within each cell-error. In addition, these new codes also have the special property that they can correct the remaining few cell-errors with two or three bit-errors.

III. Model and Definitions

In this section, the relevant error models as well as code definitions are given.

Definition 1. *A linear code $\mathcal{C}$ of length n and dimension k over an alphabet of size q that can correct t errors is referred to as an $[n,k,t]_q$ code.*

All codes considered in this work have alphabets whose cardinality is $q=2^m$ where m is some positive integer. Each cell can take on 2^m possible values and is displayed as an m-bit vector. Thus, a word of length n is represented as a length-nm binary vector where bits $mi,\ldots,m(i+1)-1$ represent the ith cell for $0\leq i\leq n-1$.

Accordingly, every cell-error is represented as a length-m vector $\boldsymbol{e}_i$. For a fixed ℓ, if $wt(\boldsymbol{e}_i)\leq\ell$ then such an error is called an *ℓ-bit-cell-error*, where the Hamming weight of a vector $\boldsymbol{x}$ is denoted by $wt(\boldsymbol{x})$. Motivated by the nature of the errors observed, it is useful to define the following class of error-vectors and codes.

Definition 2. *Given the parameters t and ℓ, an error-vector $\boldsymbol{e}=(\boldsymbol{e}_0,\boldsymbol{e}_1,\ldots,\boldsymbol{e}_{n-1})$ over $(GF(2)^m)^n$ is called a $[t;\ell]_{2^m}$-***bit-error-vector*** if the following holds:*

1) $wt(\boldsymbol{e})=|\{i:\boldsymbol{e}_i\neq\boldsymbol{0}\}|\leq t$
2) $\forall i,\ wt(\boldsymbol{e}_i)\leq\ell$.

Definition 3. *A 2^m-ary linear code $\mathcal{C}$ that can correct any $[t;\ell]_{2^m}$-bit-error-vector is called a $[t;\ell]_{2^m}$-***bit-error-correcting code***.*

From the data collected from the TLC flash device, it was observed that while most cell-errors suffered a single bit-error, only a small number of cells had double or triple bit-errors. Therefore, to correct all observed errors, it is useful to define the following refined error-vectors and corresponding codes.

Definition 4. *Let $0<\ell_1<\ell_2\leq m$, $t_1,t_2>0$. Then a vector $\boldsymbol{e}=(\boldsymbol{e}_0,\boldsymbol{e}_1,\ldots,\boldsymbol{e}_{n-1})$ over $(GF(2)^m)^n$ is called a $[t_1,t_2;\ell_1,\ell_2]_{2^m}$-***bit-error-vector*** if the following holds:*

1) $wt(\boldsymbol{e})=|\{i:\boldsymbol{e}_i\neq\boldsymbol{0}\}|\leq t_1+t_2$.
2) $\forall i,\ wt(\boldsymbol{e}_i)\leq\ell_2$.
3) $|\{i:wt(\boldsymbol{e}_i)>\ell_1\}|\leq t_2$

Definition 5. *Let $0<\ell_1<\ell_2\leq m$, $t_1,t_2>0$. Then a 2^m-ary code $\mathcal{C}$ is said to be a $[t_1,t_2;\ell_1,\ell_2]_{2^m}$-***bit-error-correcting code*** if it can correct any $[t_1,t_2;\ell_1,\ell_2]_{2^m}$-bit-error-vector.*

The next definition is useful in determining the parity-check matrices of bit-error-correcting codes.

Definition 6. *Let $A\in GF(q)^{m\times n}$, $B\in GF(q)^{p\times r}$. Then the* tensor product *of A and B is defined as the matrix*

$$A\otimes B=\begin{pmatrix}a_{0,0}B & \ldots & a_{0,n-1}B\\ \vdots & \ddots & \vdots\\ a_{m-1,0}B & \ldots & a_{m-1,n-1}B\end{pmatrix}\in GF(q)^{mp\times nr}.$$

Furthermore,

$$rank(A\otimes B)=rank(A)\cdot rank(B).$$

IV. Code Constructions

In this section, code constructions are given for bit-error-correcting codes. The section begins by revisiting a result from [11] that can be used to create $[t;\ell]_{2^m}$-bit-error-correcting codes. This idea is extended to create $[t_1,t_2;\ell_1,\ell_2]_{2^m}$-bit-error-correcting codes. Any error pattern a $[t_1,t_2;\ell_1,\ell_2]_{2^m}$ code can correct is also correctable by a $[t_1+t_2;\ell_2]_{2^m}$ code. It is shown that the former code has better redundancy whenever $\frac{t_1}{t_2}(\ell_2-\ell_1)$ is large.

In [11], it was shown that the tensor product of two parity check matrices results in a code that can correct a prescribed number of errors of a pre-defined type. For example, suppose a code with a parity check matrix $H_1\in GF(2)^{r_1\times m}$ corrects all burst errors of length 2 and a code with a parity check matrix $H_2\in(GF(2)^{r_1})^{r_2\times n}$ corrects any 3 symbol errors. Then $H_2\otimes H_1$ is a parity check matrix of a code of length nm bits, partitioned into n m-bit blocks. This code corrects any 3 block-errors assuming each block-error is a burst of length 2. In Construction A, this result is stated more formally.

A. Construction A

We start by presenting a construction of $[t;\ell]_{2^m}$-bit-error-correcting codes.

Construction A. *(see first [11]) Let $\mathcal{C}_1$ be an $[m,k_1,\ell]_2$ code with a parity check matrix H_1. Let $\mathcal{C}_2$ be an $[n,k_2,t]_{2^{m-k_1}}$ code with a parity check matrix H_2. Then, the code $\mathcal{C}_A$ with the parity matrix*

$$H_A=(H_2\otimes H_1)\,, \tag{1}$$

is a $[t;\ell]_{2^m}$-bit-error-correcting code of length n.

The correctness of the error-correction capability was proved in [11]. Furthermore, since the parity check matrix of the code $\mathcal{C}_A$ is the tensor product of the matrices H_1 and H_2, and $rank(H_2\otimes H_1)=rank(H_2)\cdot rank(H_1)$, we get that the redundancy of the code $\mathcal{C}_A$ is r_1r_2, where $r_1=m-k_1$ and $r_2=n-k_2$. An example of the encoding

of such codes was given in [10]. Suppose $\boldsymbol{c} \in \mathcal{C}_A$, where $\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{n-1}) \in (GF(2)^m)^n$. Then,

$$H_A \cdot \boldsymbol{c}^T = (H_2 \otimes H_1) \cdot \boldsymbol{c}^T$$
$$= \begin{pmatrix} h_{0,0}H_1 & \ldots & h_{0,n-1}H_1 \\ \vdots & \ddots & \vdots \\ h_{m-1,0}H_1 & \ldots & h_{m-1,n-1}H_1 \end{pmatrix} \cdot \boldsymbol{c}^T$$
$$= \begin{pmatrix} h_{0,0} & \ldots & h_{0,n-1} \\ \vdots & \ddots & \vdots \\ h_{m-1,0} & \ldots & h_{m-1,n-1} \end{pmatrix} \cdot \begin{pmatrix} H_1 \cdot {\boldsymbol{c}_0}^T \\ \vdots \\ H_1 \cdot {\boldsymbol{c}_{n-1}}^T \end{pmatrix},$$

where $h_{i,j}$ represents the symbol in position row i, column j of H_2. Thus, $\boldsymbol{c} \in \mathcal{C}_A$ if and only if $(H_1 \cdot {\boldsymbol{c}_0}^T, \ldots, H_1 \cdot {\boldsymbol{c}_{n-1}}^T) \in \mathcal{C}_2$ and the code $\mathcal{C}_A$ can be expressed as follows:

$$\mathcal{C}_A = \{\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{n-1}) \in (GF(2)^m)^n \ : \ (H_1 \cdot \boldsymbol{c}_0^T, \ldots, H_1 \cdot \boldsymbol{c}_{n-1}^T) \in \mathcal{C}_2\}.$$

For completeness and for the subsequent discussion, let us describe here a decoder for the code $\mathcal{C}_A$. Let

$$\mathcal{D}_1 : \{0,1\}^{r_1} \to \{0,1\}^m, \ \mathcal{D}_2 : (\{0,1\}^{r_1})^{r_2} \to (\{0,1\}^{r_1})^n$$

be the decoder of the code $\mathcal{C}_1, \mathcal{C}_2$, respectively. Here, and henceforth we assume that the input to the decoders of the constituent codes is the syndrome of the received vector and the output is the detected error vector. We also assume that if the code can correct t errors, then the weight of the output error vector is at most t. If the decoder finds an error vector of weight greater than t then the all-zero vector is returned as an output.

The decoder $\mathcal{D}_A : (\{0,1\}^m)^n \to (\{0,1\}^m)^n$ of the code $\mathcal{C}_A$ gets as an input a word of the form $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{c} \in \mathcal{C}_A$ and $\boldsymbol{e} \in (GF(2)^m)^n$ is a $[t; \ell]_{2^m}$-bit-error-vector. The output of the decoder is the estimate of the error vector:

1) $\mathcal{D}_2(H_2 \cdot (H_1 \cdot \boldsymbol{y}_0^T, \ldots, H_1 \cdot \boldsymbol{y}_{n-1}^T)^T) = (\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{n-1})$.
2) $\widehat{\boldsymbol{e}} = (\mathcal{D}_1(\boldsymbol{s}_0), \ldots, \mathcal{D}_1(\boldsymbol{s}_{n-1}))$.

Lemma 1. *The decoder output satisfies $\mathcal{D}_A(\boldsymbol{y}) = \widehat{\boldsymbol{e}} = \boldsymbol{e}$*

Proof: According to the definition of the code $\mathcal{C}_A$ we have $(H_1 \cdot \boldsymbol{c}_0^T, \ldots, H_1 \cdot \boldsymbol{c}_{n-1}^T) \in \mathcal{C}_2$ and we can write

$$(H_1 \cdot \boldsymbol{y}_0^T, \ldots, H_1 \cdot \boldsymbol{y}_{n-1}^T)$$
$$= (H_1 \cdot \boldsymbol{c}_0^T, \ldots, H_1 \cdot \boldsymbol{c}_{n-1}^T) + (H_1 \cdot \boldsymbol{e}_0^T, \ldots, H_1 \cdot \boldsymbol{e}_{n-1}^T).$$

The vector $(H_1 \cdot \boldsymbol{e}_0^T, \ldots, H_1 \cdot \boldsymbol{e}_{n-1}^T)$ has weight at most t and since $\mathcal{C}_2$ can correct t errors we get that

$$(\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{n-1}) = (H_1 \cdot \boldsymbol{e}_0^T, \ldots, H_1 \cdot \boldsymbol{e}_{n-1}^T).$$

Next, for every $0 \le i \le n-1$

$$H_1 \cdot \boldsymbol{y}_i^T = H_1 \cdot (\boldsymbol{c}_i^T + \boldsymbol{e}_i^T) = H_1 \cdot \boldsymbol{c}_i^T + H_1 \cdot \boldsymbol{e}_i^T$$

and since $\boldsymbol{s}_i = H_1 \cdot \boldsymbol{e}_i^T$ and the weight of $\boldsymbol{e}_i$ is at most ℓ, we get that $\mathcal{D}_1(\boldsymbol{s}_i) = \boldsymbol{e}_i$, that is, $\boldsymbol{e} = \widehat{\boldsymbol{e}}$. ∎

B. Construction B

The codes given in Construction A correct error patterns according to the maximum number of bit-errors in every cell (or m-bit symbol). Construction B extends this idea so that, while most cells suffer a small number of bit-errors, relatively few cell-errors may occur with a larger number of bit-errors. We capture this property in the following construction of $[t_1, t_2; \ell_1, \ell_2]_{2^m}$-bit-error-correcting codes.

Construction B. *Let $\mathcal{C}_1$ be an $[m, k, \ell_2]_2$ code with a parity check matrix H_1, and let $r = m - k$ be such that the following holds:*

1) *There exists $0 \le r' < r$ such that the matrix H_1' comprised of the first r' rows of H_1 is a parity check matrix for an $[m, m-r', \ell_1]_2$ code $\mathcal{C}_1'$.*
2) *H_1'' is an $r'' \times m$ matrix consisting of the last r'' rows of H_1, where $r'' = r - r'$.*
3) *H_2 is a parity check matrix for an $[n, k_2, t_1 + t_2]_{2^{r'}}$ code $\mathcal{C}_2$, and $r_2 = n - k_2$.*
4) *H_3 is a parity check matrix for an $[n, k_3, t_2]_{2^{r''}}$ code $\mathcal{C}_3$, and $r_3 = n - k_3$*

Then, a parity check matrix for a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$-bit-error-correcting code $\mathcal{C}_B$ of length n is

$$H_B = \begin{pmatrix} H_2 \otimes H_1' \\ H_3 \otimes H_1'' \end{pmatrix}. \tag{2}$$

Remark 1. *The parity check matrix H_1 of the ℓ_2-error-correcting code $\mathcal{C}_1$ needs to satisfy the property that it can be decomposed into two matrices, where the first matrix is a parity check matrix of an ℓ_1-error-correcting code $\mathcal{C}_1'$. We note this requirement is not hard to satisfy as many codes can follow this structure, and in particular BCH codes.*

We first present an example followed by the decoder for $\mathcal{C}_B$ before the error-correction ability is proven.

Example 1. *Suppose $\mathcal{C}_1$ is a triple error-correcting $[3, 0, 3]_2$ code with a parity check matrix $H_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$. Let $r' = 2$ so that $H_1' = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ where H_1' is a parity check matrix for a $[3, 1, 1]_2$ Hamming code and $H_1'' = (\ 0\ 0\ 1\ )$. Let $\mathcal{C}_2$ be a $[15, 9, 2]_4$ code with a parity check matrix H_2. Furthermore, let $\mathcal{C}_3$ be a $[15, 11, 1]_2$ Hamming code with a parity check matrix H_3. Then using Construction B, the code $\mathcal{C}_B$ has a parity check matrix*

$$H_B = \begin{pmatrix} H_2 \otimes H_1' \\ H_3 \otimes H_1'' \end{pmatrix}.$$

H_B is the parity check matrix for a $[1, 1; 1, 3]_{2^3}$-bit-error-correcting code. The particular choice of $\mathcal{C}_1$ in this example results in the same code that was proposed in [13].

Note that $\boldsymbol{c} \in \mathcal{C}_B$ if and only if

$$\boldsymbol{0} = H_B \cdot \boldsymbol{c}^T = \begin{pmatrix} H_2 \otimes H_1' \\ H_3 \otimes H_1'' \end{pmatrix} \cdot \boldsymbol{c}^T$$
$$= \begin{pmatrix} H_2 \cdot \left(\ H_1' \cdot {\boldsymbol{c}_0}^T, \ldots, H_1' \cdot {\boldsymbol{c}_{n-1}}^T\ \right)^T \\ H_3 \cdot \left(\ H_1'' \cdot {\boldsymbol{c}_0}^T, \ldots, H_1'' \cdot {\boldsymbol{c}_{n-1}}^T\ \right)^T \end{pmatrix}.$$

Hence, the code $\mathcal{C}_B$ can be expressed as

$$\mathcal{C}_B = \{\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{n-1}) \in (GF(2)^m)^n \ : \ (H_1' \cdot \boldsymbol{c}_0^T, \ldots, H_1' \cdot \boldsymbol{c}_{n-1}^T) \in \mathcal{C}_2, \ (H_1'' \cdot \boldsymbol{c}_0^T, \ldots, H_1'' \cdot \boldsymbol{c}_{n-1}^T) \in \mathcal{C}_3\}.$$

and its redundancy is at most $r'r_2 + r''r_3$.

Let us denote

$$\mathcal{D}_1 : \{0,1\}^r \to \{0,1\}^m, \ \mathcal{D}_1' : \{0,1\}^{r'} \to \{0,1\}^m,$$
$$\mathcal{D}_2 : (\{0,1\}^{r'})^{r_2} \to (\{0,1\}^{r'})^n,$$
$$\mathcal{D}_3 : (\{0,1\}^{r''})^{r_3} \to (\{0,1\}^{r''})^n,$$

to be the decoder of the code $\mathcal{C}_1, \mathcal{C}_1', \mathcal{C}_2, \mathcal{C}_3$, respectively. As before, the input to all these encoders is the syndrome and the output is the error vector whose weight is no greater than the guaranteed error-correction capability of the corresponding code.

Before presenting the decoder's steps, let us explain the idea behind this construction and its decoding procedure. We start in a similar fashion to the decoder in Construction A, where at most t_1+t_2 cell-errors each of weight at most ℓ_1 are found. Clearly, it may not possible to correct all cell-errors this way. If a cell-error has weight at most ℓ_1 then it is corrected. Otherwise, it is miscorrected to a cell-error vector, with weight at most $\ell_1+\ell_2$ since the weight of each miscorrection has been restricted to be ℓ_1. This, in turn, guarantees that the new cell-error vector is not a codeword in $\mathcal{C}_1$, since its minimum distance is at least $2\ell_2+1$. Thus, the next step is to detect these cells which were miscorrected. For cell-errors with more than ℓ_1 bits in error, the remaining part of the syndrome according to the code $\mathcal{C}_1$ is recovered. The decoder $\mathcal{D}_1$ is then used to recover the remaining errors.

The decoder $\mathcal{D}_B : (\{0,1\}^m)^n \to (\{0,1\}^m)^n$ of the code $\mathcal{C}_B$ gets as an input a word of the form $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{c} \in \mathcal{C}_B$ and $\boldsymbol{e} \in (GF(2)^m)^n$ is a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$-bit-error-vector. Its output is an estimate of the error vector $\mathcal{D}_B(\boldsymbol{y}) = \widehat{\boldsymbol{e}}$. The decoder $\mathcal{D}_B$ operates as follows:

1) $\mathcal{D}_2(H_2 \cdot (H_1' \cdot \boldsymbol{y}_0^T, \ldots, H_1' \cdot \boldsymbol{y}_{n-1}^T)^T) = (\boldsymbol{s}_0^0, \ldots, \boldsymbol{s}_{n-1}^0)$.
2) $\widehat{\boldsymbol{e}}^* = (\mathcal{D}_1'(\boldsymbol{s}_0^0), \ldots, \mathcal{D}_1'(\boldsymbol{s}_{n-1}^0))$.
3) $\boldsymbol{y}' = \boldsymbol{y} + \widehat{\boldsymbol{e}}^*$.
4) $\mathcal{D}_2(H_2 \cdot (H_1' \cdot \boldsymbol{y}_0'^T, \ldots, H_1' \cdot \boldsymbol{y}_{n-1}'^T)^T) = (\boldsymbol{s}_0', \ldots, \boldsymbol{s}_{n-1}')$.
5) $\mathcal{D}_3(H_3 \cdot (H_1'' \cdot \boldsymbol{y}_0'^T, \ldots, H_1'' \cdot \boldsymbol{y}_{n-1}'^T)^T) = (\boldsymbol{s}_0'', \ldots, \boldsymbol{s}_{n-1}'')$.
6) Let $I = \{i : (\boldsymbol{s}_i', \boldsymbol{s}_i'') \neq (\boldsymbol{0}, \boldsymbol{0})\}$.
7) Let $\boldsymbol{y}''$ satisfy: $\boldsymbol{y}_i'' = \boldsymbol{y}_i$ if $i \in I$ and $\boldsymbol{y}_i'' = \boldsymbol{y}_i'$ if $i \notin I$.
8) $\mathcal{D}_3(H_3 \cdot (H_1'' \cdot \boldsymbol{y}_0''^T, \ldots, H_1'' \cdot \boldsymbol{y}_{n-1}''^T)^T) = (\boldsymbol{s}_0^1, \ldots, \boldsymbol{s}_{n-1}^1)$.
9) $\widehat{\boldsymbol{e}} = (\widehat{\boldsymbol{e}}_0, \ldots, \widehat{\boldsymbol{e}}_{n-1})$ where $\widehat{\boldsymbol{e}}_i = \widehat{\boldsymbol{e}}_i^*$ if $i \notin I$ and otherwise $\widehat{\boldsymbol{e}}_i = \mathcal{D}_1(\boldsymbol{s}_i^0, \boldsymbol{s}_i^1)$.

Theorem 1. *The decoder output satisfies $\mathcal{D}_B(\boldsymbol{y}) = \widehat{\boldsymbol{e}} = \boldsymbol{e}$.*

Proof: Let $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ be the received word to the decoder $\mathcal{D}_B$ where $\boldsymbol{c} \in \mathcal{C}_B$ and $\boldsymbol{e} \in (GF(2)^m)^n$ is a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$-bit-error-vector. Then according to the definition of the code $\mathcal{C}_B$, $(H_1' \cdot \boldsymbol{c}_0^T, \ldots, H_1' \cdot \boldsymbol{c}_{n-1}^T) \in \mathcal{C}_2$ and

$$(H_1' \cdot \boldsymbol{y}_0^T, \ldots, H_1' \cdot \boldsymbol{y}_{n-1}^T) = (H_1' \cdot \boldsymbol{c}_0^T, \ldots, H_1' \cdot \boldsymbol{c}_{n-1}^T) + (H_1' \cdot \boldsymbol{e}_0^T, \ldots, H_1' \cdot \boldsymbol{e}_{n-1}^T).$$

The vector $(H_1' \cdot \boldsymbol{e}_0^T, \ldots, H_1' \cdot \boldsymbol{e}_{n-1}^T)$ now has weight at most t_1+t_2 and since the code $\mathcal{C}_2$ can correct this number of errors we get that $(\boldsymbol{s}_0^0, \ldots, \boldsymbol{s}_{n-1}^0) = (H_1' \cdot \boldsymbol{e}_0^T, \ldots, H_1' \cdot \boldsymbol{e}_{n-1}^T)$ after step 1.

At step 2 since for every $0 \leq i \leq n-1$, $H_1' \cdot \boldsymbol{y}_i^T = H_1' \cdot \boldsymbol{c}_i^T + H_1' \cdot \boldsymbol{e}_i^T$, if $wt(\boldsymbol{e}_i) \leq \ell_1$,

$$\widehat{\boldsymbol{e}}_i^* = \mathcal{D}_1'(\boldsymbol{s}_i^0) = \boldsymbol{e}_i,$$

as $\mathcal{C}_1'$ corrects ℓ_1 errors. However, if the weight of $\boldsymbol{e}_i$ is between ℓ_1+1 and ℓ_2 then $\widehat{\boldsymbol{e}}_i^* = \mathcal{D}_1(\boldsymbol{s}_i^0) \neq \boldsymbol{e}_i$. This observation results from the fact that the decoder $\mathcal{C}_1'$ can only return a cell-error vector of weight at most ℓ_1. In particular, we get that $wt(\widehat{\boldsymbol{e}}^*) \leq t_2$ and for all $0 \leq i \leq n-1$, $wt(\widehat{\boldsymbol{e}}_i^*) \leq \ell_1 + \ell_2$. Thus, at the end of step 3, $\boldsymbol{y}'$ contains no cell errors of weight less than ℓ_1 and all the remaining (at most t_2) cell-errors have weight at most $\ell_1+\ell_2$.

Steps 4 and 5 compute the syndrome using $\boldsymbol{y}'$ as input. Since the minimum distance of the code $\mathcal{C}_1$ is at least $2\ell_2+1 > \ell_1+\ell_2$, we get that for all $0 \leq i \leq n-1$, if a miscorrection occurred, then $\widehat{\boldsymbol{e}}_i^*$ is not a codeword in $\mathcal{C}_1$. Therefore, $(\boldsymbol{s}_i', \boldsymbol{s}_i'') \neq (\boldsymbol{0}, \boldsymbol{0})$ and in step 6 the set I is the set of all $0 \leq i \leq n-1$ such that $\ell_1 < wt(\boldsymbol{e}_i) \leq \ell_2$. In step 7, the word $\boldsymbol{y}''$ is the word of $\boldsymbol{y}$ after removing all cell-errors of weight at most ℓ_1.

In step 8 the remaining portion of the syndrome is recovered for all cell-errors with more than ℓ_1 bits in error. Lastly in step 9 for every cell-error at position i, if ℓ_1 or less bit-errors occurred then $\widehat{\boldsymbol{e}}_i^*$ is its corresponding cell-error vector and if more bit-errors occurred then the decoder $\mathcal{D}_1$ is used. Since $\mathcal{C}_1$ can correct ℓ_2 errors and the syndrome $H_1 \cdot \boldsymbol{e}_i{}^T = (\boldsymbol{s}_i^0, \boldsymbol{s}_i^1)$ is known for all cell-errors with more than ℓ_1 bits in error, these errors are corrected as well. ∎

It can be shown that $\mathcal{C}_B$ requires less redundancy than $\mathcal{C}_A$ approximately whenever $\frac{\log n}{\log m} < \frac{t_1}{t_2}(\ell_2 - \ell_1)$. The next construction reduces the required redundancy in certain cases when the ratio $\frac{t_1}{t_2}$ is small.

C. Construction C

Construction C extends Construction B by using a combination of codes whose abilities are to correct errors, correct erasures, and detect errors. In particular, the code $\mathcal{C}_1'$ in Construction B is modified such that it corrects ℓ_1 errors and detects when there are between ℓ_1+1 and ℓ_2 errors. Accordingly, the code $\mathcal{C}_3$ in Construction B need only correct t_2 erasures instead of t_2 errors.

Construction C. *Let $\mathcal{C}_C$ be a code with the following modifications with respect to the code construction of $\mathcal{C}_B$:*

1) *The code $\mathcal{C}_1$ remains an $[m, k, \ell_2]_2$ code as in Construction B where $r = m - k$.*
2) *The matrix H_1' now consists of the first r' rows of H_1 where*
   a) *H_1' is a parity check matrix for an $[m, m-r', \ell_1]_2$-bit-error-correcting code $\mathcal{C}_1'$,*
   b) *The minimum distance of $\mathcal{C}_1'$ is at least $\ell_1+\ell_2+1$ so is can detect an error vector of weight between ℓ_1+1 and ℓ_2.*
3) *H_1'' is an $r'' \times m$ matrix consisting of the last r'' rows of H_1, where $r'' = r - r'$.*
4) *H_2 is a parity check matrix for an $[n, k_2, t_1+t_2]_{2^{r'}}$ code $\mathcal{C}_2$, and $r_2 = n - k_2$.*
5) *H_3 is a parity check matrix of an $[n, k_3, \lceil \frac{t_2}{2} \rceil]_{2^{r''}}$ code $\mathcal{C}_3$ that can correct at least t_2 erasures, and $r_3 = n - k_3$.*

A parity check matrix for $\mathcal{C}_C$ is $H_C = \begin{pmatrix} H_2 \otimes H_1' \\ H_3 \otimes H_1'' \end{pmatrix}$. The decoders of the codes $\mathcal{C}_1'$ and $\mathcal{C}_3$ are also changed while the decoders for $\mathcal{C}_1$ and $\mathcal{C}_2$ remain the same as in Construction B. The decoder $\mathcal{D}_1'$, in addition to correcting ℓ_1 errors, also detects if the number of errors is between ℓ_1+1 and ℓ_2. The decoder is defined

$$\mathcal{D}_1' : \{0,1\}^{r'} \to \{0,1\}^m \cup \{E\},$$

where the symbol E indicates a detected error of weight between ℓ_1+1 and ℓ_2. Note that now the decoder $\mathcal{D}_1'$ never miscorrects when the number of errors in each cell is at most ℓ_2. The input to the decoder $\mathcal{D}_3$ is no longer a syndrome but a vector $\boldsymbol{x} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1})$ with at most t_2 erasures

$$\mathcal{D}_3 : (\{0,1\}^{r''} \cup ?)^n \rightarrow (\{0,1\}^{r''})^n,$$

where ? is the erasure symbol. The output of the decoder $\mathcal{D}_C$ is the 'erasure' vector $\boldsymbol{e} = (\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{n-1})$, where for every $\boldsymbol{e}_i \neq \boldsymbol{0}$, $\boldsymbol{e}_i$ is the correct value for $\boldsymbol{x}_i$. The decoder $\mathcal{D}_C : (\{0,1\}^m)^n \rightarrow (\{0,1\}^m)^n$ is summarized below. Recall the input is $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.

1) $\mathcal{D}_2(H_2 \cdot (H_1' \cdot \boldsymbol{y}_0^T, \ldots, H_1' \cdot \boldsymbol{y}_{n-1}^T)^T) = (\boldsymbol{s}_0^0, \ldots, \boldsymbol{s}_{n-1}^0)$.
2) $\widehat{\boldsymbol{e}}^* = (\mathcal{D}_1'(\boldsymbol{s}_0^0), \ldots, \mathcal{D}_1'(\boldsymbol{s}_{n-1}^0))$.
3) Let $I = \{i \ : \ \widehat{\boldsymbol{e}}_i^* = \ E\}$.
4) Let $\widehat{\boldsymbol{e}}'$ satisfy: $\widehat{\boldsymbol{e}}_i' = \boldsymbol{0}$ if $i \in I$ and $\widehat{\boldsymbol{e}}_i' = \widehat{\boldsymbol{e}}_i^*$ if $i \notin I$.
5) $\boldsymbol{y}' = \boldsymbol{y} + \widehat{\boldsymbol{e}}'$.
6) Let $\boldsymbol{x}$ satisfy: $\boldsymbol{x}_i = ?$ if $i \in I$ and $\boldsymbol{x}_i = H_1'' \cdot \boldsymbol{y}_i'^T$ if $i \notin I$.
7) $\mathcal{D}_3(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1}) = (\boldsymbol{s}_0^1, \ldots, \boldsymbol{s}_{n-1}^1)$.
8) $\widehat{\boldsymbol{e}} = (\widehat{\boldsymbol{e}}_0, \ldots, \widehat{\boldsymbol{e}}_{n-1})$ where $\widehat{\boldsymbol{e}}_i = \widehat{\boldsymbol{e}}_i^*$ if $i \notin I$ and otherwise $\widehat{\boldsymbol{e}}_i = \mathcal{D}_1(\boldsymbol{s}_i^0, \boldsymbol{s}_i^1 + H_1'' \cdot \boldsymbol{y}_i'^T)$.

The proof of the decoder correctness of Construction C is omitted due to a lack of space. It can be shown that Construction C requires less redundancy than Construction A approximately when $\frac{\log n}{\log m} < (\frac{t_1}{t_2} + \frac{1}{2})(\ell_2 - \ell_1)$. Furthermore, it requires less redundancy than Construction B roughly when $\frac{\log n}{\log m} > (\frac{t_1}{t_2} - \frac{1}{2})(\ell_2 - \ell_1)$.

V. Performance and Results

In this section, the performance of various linear error-correcting codes with guaranteed error-correction capability is evaluated for the TLC Flash device. The results of these simulations are shown in Figure 2. All the known codes used were the best known linear codes according to [9] of the longest block length.

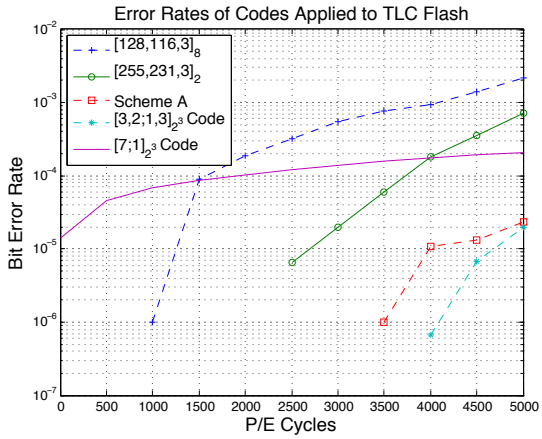

Fig. 2. Bit Error Rates of Codes Applied to TLC Flash

The first code shown in the legend of Fig. 2 is a non-binary $[128, 116, 3]_8$ code with rate 0.906 where each 8-ary symbol corresponds to an 8-ary cell. The second code in the figure is the result of applying a $[255, 232, 3]_2$ BCH code three times to each of the 3 bits in each cell.

The rate 0.904 code labeled 'Scheme A' is comprised of a non-binary $[256, 227, 5]_4$ code applied to the LSB and CSB bits for each Flash memory cell. Next, an independent binary $[256, 240, 2]_2$ code was used to protect the remaining bit of information from each cell. This scheme was designed to target the property observed in Section II where the CSB and LSB were more likely to err than the MSB.

The $[3, 2; 1, 3]_8$-bit-error-correcting code of length 256 with rate 0.904 was generated according to Construction B. In this case, $\mathcal{C}_1{}'$ is the $[3, 1, 1]_2$ binary repetition code. The code $\mathcal{C}_2$ is the same $[256, 227, 5]_4$ code used in Scheme A and $\mathcal{C}_3$ is the same $[256, 240, 2]_2$ code used in Scheme A as well. For reference, we included a $[7; 1]_{2^3}$ bit-error code of length 256 and rate 0.83 (constructed using one tensor product operation). From Fig. 2 this particular tensor product code has the ability to delay the appearance of any errors in the system by a factor of 4 over the naive $GF(8)$ code. In addition, the proposed tensor product code offers a 1.6x lifetime improvement over the popular BCH codes.

VI. Conclusion

In this work, data from a TLC Flash device demonstrated that when errors occur within a Flash cell, the vast majority of such errors only affect one of the 3 bits of information. This observation was used to motivate a new error-correction model for Flash memory. Error-correcting code constructions based upon generalized tensor product codes were provided that were analytically and empirically shown to offer a potentially valuable component for future coding schemes in the context of Flash memory.

Acknowledgement

Research supported in part by SMART scholarship, and NSF grants CCF-1029030 and CCF-1150212.

References

[1] Y. Cassuto *et al.*, "Codes for asymmetric limited-magnitude errors with application to multi-level flash memories, *IEEE Trans. on Inform. Theory*, vol. 56, no. 4, pp. 1582-1595, Apr. 2010.

[2] P. Chaichanavong and P. H. Siegel, "A tensor-product parity code for magnetic recording," *IEEE Trans. on Magnetics*, vol. 42, no. 2, pp. 350-352, Feb. 2006.

[3] P. Chaichanavong and P. H. Siegel, "Tensor-product parity codes: combination with constrained codes and application to perpendicular recording," *IEEE Trans. on Magnetics*, vol. 42, no. 2, pp. 214-219, Feb. 2006.

[4] 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, Mar. 2010.

[5] S. Fei *et al.*, "Multilevel Flash memory on-chip error correction based on trellis coded modulation," *IEEE ISCS*, Island of Kos, Greece, Sep. 2006.

[6] Q. Huang, S. Lin, and K. A. S. Abdel-Ghaffar, "Error-correcting codes for flash coding," *IEEE Trans. Inform. Theory*, vol. 57, no. 9, pp. 6097-6108, Apr. 2011.

[7] H. Imai and H. Fujiya, "Generalized tensor product codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 2, pp. 181-187, Mar. 1981.

[8] Y. Maeda and H. Kaneko, "Error control coding for multilevel cell flash memories using nonbinary low-density parity-check codes," *IEEE ISDFTVS*, Chicago, IL, Oct. 2009.

[9] University of Sydney, "Magma Computational Algebra System," *http://magma.maths.usyd.edu.au/magma/*, 2011.

[10] J. K. Wolf, "An introduction to tensor product codes and applications to digital storage systems," *IEEE ITW*, Punta del Este, Uruguay, Oct. 2006.

[11] J. K. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Trans. Inform. Theory*, vol. 11, no. 2, pp. 281-284, Apr. 1965.

[12] E. Yaakobi *et al.*, "On codes that correct asymmetric errors with graded magnitude distribution," *IEEE ISIT*, St. Petersburgh, Russia, Aug. 2011.

[13] E. Yaakobi *et al.*, "Characterization and error-correcting codes for TLC flash memories," *IEEE ICNC*, Maui, HI, Jan.-Feb. 2012.