Tackling Intracell Variability in TLC Flash Through Error Correction Coding

Ryan Gabrys, Lara Dolecek

Department of Electrical Engineering UCLA
1. Background
2. Empirical Data
3. Error-Correction Model
4. Error-Correcting Codes
5. Performance Results
6. Conclusion
Flash memory is comprised of a set of floating gate cells.
Technical constraint

- Flash memory is comprised of a set of floating gate cells.
- Information is stored by controlling the number of electrons stored within each cell.
Flash memory is comprised of a set of floating gate cells.
Information is stored by controlling the number of electrons stored within each cell.
Density Per Cell
Technical constraint

- Flash memory is comprised of a set of floating gate cells.
- Information is stored by controlling the number of electrons stored within each cell.
- Density Per Cell
  - Single-Level-Cell (SLC) 1 bit per cell.
Technical constraint

- Flash memory is comprised of a set of floating gate cells.
- Information is stored by controlling the number of electrons stored within each cell.
- Density Per Cell
  - Single-Level-Cell (SLC) 1 bit per cell.
  - Multiple-Level-Cell (MLC) 2 bits per cell.
Technical constraint

- Flash memory is comprised of a set of floating gate cells.
- Information is stored by controlling the number of electrons stored within each cell.
- Density Per Cell
  - Single-Level-Cell (SLC) 1 bit per cell.
  - Multiple-Level-Cell (MLC) 2 bits per cell.
  - Triple-Level-Cell (TLC) 3 bits of information per cell.
Technical constraint

- Flash memory is comprised of a set of floating gate cells.
- Information is stored by controlling the number of electrons stored within each cell.
- Density Per Cell
  - Single-Level-Cell (SLC) 1 bit per cell.
  - Multiple-Level-Cell (MLC) 2 bits per cell.
  - Triple-Level-Cell (TLC) 3 bits of information per cell.
Previous work

- Recent error-correcting codes for Flash memory
Previous work

- Recent error-correcting codes for Flash memory

- Tensor product codes
Voltage Levels for TLC

- Most Significant Bit **MSB**
- Center Significant Bit **CSB**
- Least Significant Bit **LSB**
Data Collection

- Below is an image of the custom board from UCSD used to collect the data.
Data Collection

- Below is an image of the custom board from UCSD used to collect the data.
- On the first of every 100 P/E cycles the following was performed:

1. Erase the block. (block = 20 cells)
2. Read back the errors.
3. Write random data.
4. Read back the errors.

On the other 99 cycles, the block was erased and all-zeros were written.
Data Collection

- Below is an image of the custom board from UCSD used to collect the data.
- On the first of every 100 P/E cycles the following was performed:
  1. Erase the block. (block = $2^{20}$ cells)
Below is an image of the custom board from UCSD used to collect the data.

On the first of every 100 P/E cycles the following was performed:

1. Erase the block. (block = $2^{20}$ cells)
2. Read back the errors.
Data Collection

- Below is an image of the custom board from UCSD used to collect the data.
- On the first of every 100 P/E cycles the following was performed:
  1. Erase the block. (block = $2^{20}$ cells)
  2. Read back the errors.
  3. Write random data.
Below is an image of the custom board from UCSD used to collect the data.

On the first of every 100 P/E cycles the following was performed:

1. Erase the block. (block = 2^{20} cells)
2. Read back the errors.
3. Write random data.
4. Read back the errors.
Below is an image of the custom board from UCSD used to collect the data.

On the first of every 100 P/E cycles the following was performed:

1. Erase the block. (block= $2^{20}$ cells)
2. Read back the errors.
3. Write random data.
4. Read back the errors.

On the other 99 cycles, the block was erased and all-zeros were written.
Raw Error Rate

Error Rates for TLC Flash

- LSB
- CSB
- MSB
- Symbol Error Rate
### Error Patterns Within a Symbol

<table>
<thead>
<tr>
<th>Number of bits in symbol that err</th>
<th>Percentage of errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9617</td>
</tr>
<tr>
<td>2</td>
<td>0.0314</td>
</tr>
<tr>
<td>3</td>
<td>0.0069</td>
</tr>
</tbody>
</table>
Error Patterns Within a Symbol

<table>
<thead>
<tr>
<th>Number of bits in symbol that err</th>
<th>Percentage of errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9617</td>
</tr>
<tr>
<td>2</td>
<td>0.0314</td>
</tr>
<tr>
<td>3</td>
<td>0.0069</td>
</tr>
</tbody>
</table>

Idea: Design a code for observed intracell variability.
Code Properties

- Codes are over alphabet of size $q = 2^m$, where $m$ is some positive integer and each symbol represents a Flash cell.
Code Properties

- Codes are over alphabet of size $q = 2^m$, where $m$ is some positive integer and each symbol represents a Flash cell.
- A symbol is a binary length-$m$ vector.
Code Properties

- Codes are over alphabet of size $q = 2^m$, where $m$ is some positive integer and each symbol represents a Flash cell.
- A symbol is a binary length-$m$ vector.
- A codeword is $n$ binary length-$m$ vectors so the result is a length-$nm$ vector.
Code Properties

- Codes are over alphabet of size \( q = 2^m \), where \( m \) is some positive integer and each symbol represents a Flash cell.
- A symbol is a binary length-\( m \) vector.
- A codeword is \( n \) binary length-\( m \) vectors so the result is a length-\( nm \) vector.
- Example over alphabet of size 8:
  \((45702) \rightarrow (100\ 101\ 111\ 000\ 010)\)
Error Vectors

**Definition (Bit-Error Vector)**

The length-$nm$ vector $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t; \ell]$-bit-error-vector if

$|\{i : e_i \neq 0\}| \leq t \quad \forall i, \quad \text{wt}(e_i) \leq \ell$.
Error Vectors

Definition (Bit-Error Vector)

The length-$nm$ vector $e = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t; \ell]$-bit-error-vector if

$$|\{i : e_i \neq 0\}| \leq t.$$
Error Vectors

Definition (Bit-Error Vector)

The length-$nm$ vector $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t; \ell]$-bit-error-vector if

1. $|\{i : e_i \neq 0\}| \leq t$.
2. $\forall i, \text{wt}(e_i) \leq \ell$. 
Error Vectors

**Definition (Bit-Error Vector)**

The length-$nm$ vector $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t; \ell]$-bit-error-vector if

1. $|\{i : e_i \neq 0\}| \leq t$.
2. $\forall i, \text{wt}(e_i) \leq \ell$.

**Definition (Bit-Error-Correcting Code)**

A code $\mathcal{C}$ is a $[t; \ell]$-bit-error-correcting code if it can correct any $[t; \ell]$-bit-error-vector.
Definition (Graded Bit-Error Vector)

The length-$nm$ vector $e = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector if

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

Example of a $[5, 2; 1, 3]$-bit-error-vector: $(100, 100, 000, 010, 111, 001, 000, 111, 010)$. 
Error Vectors (ctd.)

Definition (Graded Bit-Error Vector)

The length-$nm$ vector $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{n-1})$, where each $m$-bit vector $\mathbf{e}_i$ represents a symbol of size $2^m$, is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector if

1. $\left| \{ i : \mathbf{e}_i \neq \mathbf{0} \} \right| \leq t_1 + t_2$. 

<table>
<thead>
<tr>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Vectors (ctd.)</td>
</tr>
<tr>
<td>Definition (Graded Bit-Error Vector)</td>
</tr>
</tbody>
</table>

The length-$nm$ vector $\mathbf{e} = (\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{n-1})$, where each $m$-bit vector $\mathbf{e}_i$ represents a symbol of size $2^m$, is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector if

1. $\left| \{ i : \mathbf{e}_i \neq \mathbf{0} \} \right| \leq t_1 + t_2$. 

Example of a $[5, 2; 1, 3]$-bit-error-vector: $(100 100 000 010 111 001 000 111 010)$. 

11 / 19
Definition (Graded Bit-Error Vector)

The length-$nm$ vector $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector if

1. $|\{i : e_i \neq 0\}| \leq t_1 + t_2$.
2. $\forall i$, $wt(e_i) \leq \ell_2$. 

Example of a $[5, 2; 1, 3]$-bit-error-vector: $(100 100 000 010 111 001 000 111 010)$. 
Definition (Graded Bit-Error Vector)

The length-$nm$ vector $e = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector if

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

Example of a $[5, 2; 1, 3]$-bit-error-vector:

$(100 100 000 010 111 001 000 111 010)$. 

11 / 19
Error Vectors (ctd.)

Definition (Graded Bit-Error Vector)

The length-$nm$ vector $e = (e_0, e_1, \ldots, e_{n-1})$, where each $m$-bit vector $e_i$ represents a symbol of size $2^m$, is a $[t_1, t_2; l_1, l_2]$-bit-error-vector if

1. $|\{i : e_i \neq 0\}| \leq t_1 + t_2$.
2. $\forall i, \text{wt}(e_i) \leq l_2$.
3. $|\{i : \text{wt}(e_i) > l_1\}| \leq t_2$.

Example of a $[5, 2; 1, 3]$-bit-error-vector:

( 100 100 000 010 111 001 000 111 010 ).
Correcting Weighted Error Patterns

Definition (Graded Bit-Error-Correcting Code)

A code $C$ is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-correcting code if it can correct any $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector.
Definition (Graded Bit-Error-Correcting Code)

A code $C$ is a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-correcting code if it can correct any $[t_1, t_2; \ell_1, \ell_2]$-bit-error-vector.

Goal is to construct a $[t_1, t_2; \ell_1, \ell_2]$-bit-error-correcting code and apply to Flash to mitigate the observed intracell variability.
Theorem

Let $H_1$ be a parity check matrix for the $[m, k_1, 2\ell + 1]_2$ code $C_1$ \textit{(standard $[n, k, d]$ notation)}.

Tensor Product Codes [1]

Theorem

- Let $H_1$ be a parity check matrix for the $[m, k_1, 2\ell + 1]_2$ code $C^1$ (standard $[n, k, d]$ notation).
- Let $H_2$ be a parity check matrix for the $[n, k_2, 2t + 1]_{2^{m-k_1}}$ code $C^2$ defined over the alphabet of size $GF(2)^{m-k_1}$.

Tensor Product Codes [1]

Theorem

Let $H_1$ be a parity check matrix for the $[m, k_1, 2\ell + 1]_2$ code $C^1$ (standard $[n, k, d]$ notation).

Let $H_2$ be a parity check matrix for the $[n, k_2, 2t + 1]_{2^{m-k_1}}$ code $C^2$ defined over the alphabet of size $GF(2)^{m-k_1}$.

Then, $H_2 \otimes H_1$ is a parity check matrix for a $[t, \ell]$-bit-error-correcting code.

Construction of a \([t_1, t_2; \ell_1, \ell_2]\) graded-bit-error-correcting code

- Suppose \(H_1\) is an \(r \times m\) parity check matrix of a \([m, k_1, \ell_2]_2\) code \(C_1\) where \(H_1\) is \[
\begin{bmatrix}
H'_1 \\
H''_1
\end{bmatrix}
\] such that the following holds:
Construction of a \([t_1, t_2; \ell_1, \ell_2]\) graded-bit-error-correcting code

- Suppose \(H_1\) is an \(r \times m\) parity check matrix of a \([m, k_1, \ell_2]_2\) code \(C_1\) where \(H_1\) is \[
\begin{bmatrix}
H'_1 \\
H_1^\dagger
\end{bmatrix}
\]
such that the following holds:
  - \(H'_1\) is the parity check matrix of a \([m, m - r', \ell_1]_2\) code and
Construction of a $[t_1, t_2; \ell_1, \ell_2]$ graded-bit-error-correcting code

Suppose $H_1$ is an $r \times m$ parity check matrix of a $[m, k_1, \ell_2]_2$ code $C_1$ where $H_1$ is

$$
\begin{bmatrix}
H'_1 \\
H''_1
\end{bmatrix}
$$

such that the following holds:

1. $H'_1$ is the parity check matrix of a $[m, m - r', \ell_1]_2$ code and
2. $H''_1$ is a $r''$ by $m$ matrix for $r'' = r - r'$. 
Construction of a \([t_1, t_2; ℓ_1, ℓ_2]\)-graded-bit-error-correcting code

- Suppose \(H_2\) is the parity check matrix of a \([n, k_2, t_1 + t_2]_{2r'}\) code.
Construction of a $[t_1, t_2; \ell_1, \ell_2]$-graded-bit-error-correcting code

- Suppose $H_2$ is the parity check matrix of a $[n, k_2, t_1 + t_2]_{2^{r'}}$ code.
- Suppose $H_3$ is the parity check matrix of a $[n, k_3, t_2]_{2^{r''}}$ code.
Construction of a \([t_1, t_2; \ell_1, \ell_2]\)-graded-bit-error-correcting code

- Suppose \(H_2\) is the parity check matrix of a \([n, k_2, t_1 + t_2]_{2r'}\) code.
- Suppose \(H_3\) is the parity check matrix of a \([n, k_3, t_2]_{2r''}\) code.

**Theorem (Construction 2)**

Then \(H_B\) is the parity check matrix of a \([t_1, t_2; \ell_1, \ell_2]_{2^m}\)-graded bit error correcting code, where

\[
H_B = \begin{pmatrix}
H_2 \otimes H_1' \\
H_3 \otimes H_1''
\end{pmatrix}.
\]
Using sphere-packing bound argument, it follows that the excess redundancy of $C_B$ is about $t_2 \log(n)$. 
Using sphere-packing bound argument, it follows that the excess redundancy of $C_B$ is about $t_2 \log(n)$.

Construction 1 is also a graded-bit-error correcting code. Construction 2 offers better redundancy than Construction 1. when $(\ell_2 - \ell_1)t_1/t_2 > \log(n)/\log(m)$. 

Using sphere-packing bound argument, it follows that the excess redundancy of $C_B$ is about $t_2 \log(n)$.

Construction 1 is also a graded-bit-error correcting code. Construction 2 offers better redundancy than Construction 1 when $(\ell_2 - \ell_1)t_1/t_2 > \log(n)/\log(m)$.

Further simplifications are possible for special cases of the code parameters.
For TLC Flash, we compared a [3, 2; 1, 3]_8-graded-bit-error-correcting code \( C \) of length 256 with rate 0.904 against the following codes:

1. A non-binary [128, 116, 3]_8 code with rate 0.906.
2. A binary [255, 231, 2]_2 BCH code with rate 0.906, applied to MSB/CSB/LSB in parallel.
3. 'Scheme A' - Comprised of a non-binary [256, 227, 5]_4 code \( C_2 \) applied to the LSB and the CSB for each Flash memory cell. An independent binary [256, 240, 5]_2 code \( C_3 \) was applied to the MSB for each Flash memory cell. The overall rate is 0.904. Constituents of \( C \) are \( C_1 \) as [3, 0, 3]_2 (with \( C'_1 \) as repetition code), and \( C_2 \) and \( C_3 \) from Scheme A.
For TLC Flash, we compared a $[3, 2; 1, 3]_8$-graded-bit-error-correcting code $C$ of length 256 with rate 0.904 against the following codes:

1. A non-binary $[128, 116, 3]_8$ code with rate 0.906.
For TLC Flash, we compared a $[3, 2; 1, 3]_8$-graded-bit-error-correcting code $C$ of length 256 with rate 0.904 against the following codes:

1. A non-binary $[128, 116, 3]_8$ code with rate 0.906.
2. A binary $[255, 231, 3]_2$ BCH code with rate 0.906, applied to MSB/CSB/LSB in parallel.
For TLC Flash, we compared a $[3, 2; 1, 3]_8$-graded-bit-error-correcting code $C$ of length 256 with rate 0.904 against the following codes:

1. A non-binary $[128, 116, 3]_8$ code with rate 0.906.
2. A binary $[255, 231, 3]_2$ BCH code with rate 0.906, applied to MSB/CSB/LSB in parallel.
3. 'Scheme A' - Comprised of a non-binary $[256, 227, 5]_4$ code $C_2^2$ applied to the LSB and the CSB for each Flash memory cell. An independent binary $[256, 240, 5]_2$ code $C_3^3$ was applied to the MSB for each Flash memory cell. The overall rate is 0.904.
Evaluation

- For TLC Flash, we compared a $[3, 2; 1, 3]_8$-graded-bit-error-correcting code $C$ of length 256 with rate 0.904 against the following codes:
  1. A non-binary $[128, 116, 3]_8$ code with rate 0.906.
  2. A binary $[255, 231, 3]_2$ BCH code with rate 0.906, applied to MSB/CSB/LSB in parallel.
  3. 'Scheme A' - Comprised of a non-binary $[256, 227, 5]_4$ code $C^2$ applied to the LSB and the CSB for each Flash memory cell. An independent binary $[256, 240, 5]_2$ code $C^3$ was applied to the MSB for each Flash memory cell. The overall rate is 0.904.

- Constituents of $C$ are $C^1$ as $[3, 0, 3]_2$ (with $C'_1$ as repetition code), and $C^2$ and $C^3$ from Scheme A.
Results

Error Rates of Codes Applied to TLC Flash

- [128,116,3]_8
- [255,231,3]_2
- Scheme A
- [3,2;1,3]_2^3 Code
- [7;1]_2^3 Code

P/E Cycles

Bit Error Rate
Newer generations of Flash memory continue to demand more efficient error-correction schemes.
Newer generations of Flash memory continue to demand more efficient error-correction schemes.

Codes based upon Tensor Product Codes offer an efficient alternative to binary and non-binary linear codes.