Tackling Intracell Variability in TLC Flash Through Error Correction Coding

Ryan Gabrys,Lara Dolecek

Department of Electrical Engineering UCLA

Technical constraint

- 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.

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

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

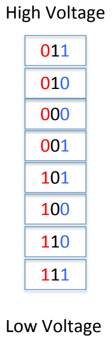
- Recent error-correcting codes for Flash memory
  - T. Kløve, B. Bose, N. Elarief, "Systematic Single Limited Magnitude Error Correcting Codes for Flash Memories," 2011.
  - Y. Cassuto et al., "Codes for Multi-Level Flash Memories: Correcting Asymmetric Limited-Magnitude Errors," 2010.
  - Y. Maeda and H. Kaneko, "Error Control Coding for Multilevel Cell Flash Memories Using Nonbinary Low-Density Parity-Check Codes," 2009.

Previous work

- Recent error-correcting codes for Flash memory
  - T. Kløve, B. Bose, N. Elarief, "Systematic Single Limited Magnitude Error Correcting Codes for Flash Memories," 2011.
  - Y. Cassuto et al., "Codes for Multi-Level Flash Memories: Correcting Asymmetric Limited-Magnitude Errors," 2010.
  - Y. Maeda and H. Kaneko, "Error Control Coding for Multilevel Cell Flash Memories Using Nonbinary Low-Density Parity-Check Codes," 2009.
- Tensor product codes
  - P. Chaichanavong and P.H. Siegel, "A Tensor-Product Parity Code for Magnetic Recording," 2006.
  - J.K. Wolf, "On Codes Derivable from the Tensor Product of Check Matrices," 1965.

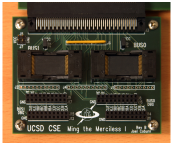
Voltage Levels for TLC

- Most Significant Bit MSB
- Center Significant Bit CSB
- Least Significant Bit LSB



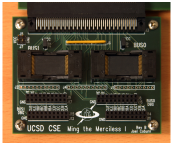
Data Collection

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



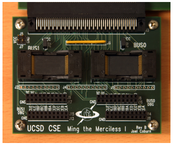
Data Collection

- 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:



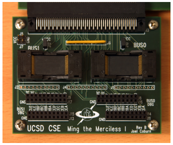
Data Collection

- 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)



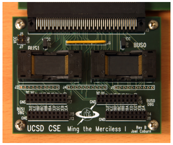
Data Collection

- 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.



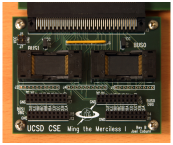
Data Collection

- 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.



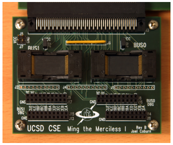
Data Collection

- 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.
  4. Read back the errors.

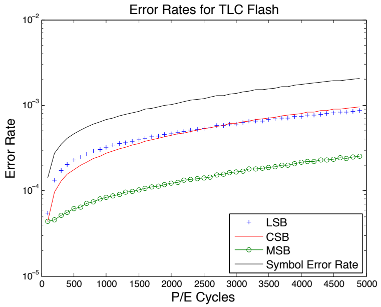


Data Collection

- 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.
  4. Read back the errors.
- On the other 99 cycles, the block was erased and all-zeros were written.



Raw Error Rate



Error Patterns Within a Symbol

Number of bits in symbol that err	Percentage of errors
1	0.9617
2	0.0314
3	0.0069

Error Patterns Within a Symbol

Number of bits in symbol that err	Percentage of errors
1	0.9617
2	0.0314
3	0.0069

$\boxed{\text{Idea: Design a code for observed intracell variability.}}$

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:
  $(45702) - > (100\ 101\ 111\ 000\ 010)$

Error Vectors

Definition (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 $[t; \ell]$-bit-error-vector if

Error Vectors

Definition (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 $[t; \ell]$-bit-error-vector if

1. $|\{i : \mathbf{e}_i \neq \mathbf{0}\}| \leq t$.

Error Vectors

Definition (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 $[t; \ell]$-bit-error-vector if

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

Error Vectors

Definition (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 $[t; \ell]$-bit-error-vector if

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

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.

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

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

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

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

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

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

Correcting Weighted Error Patterns

Definition (Graded Bit-Error-Correcting Code)

A code $\mathcal{C}$ 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.

Correcting Weighted Error Patterns

Definition (Graded Bit-Error-Correcting Code)

A code $\mathcal{C}$ 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.

Tensor Product Codes [1]

Theorem

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

[1] J.K. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Trans. On Information Theory*, vol. 8, pp. 163-169, April 1965.

Tensor Product Codes [1]

Theorem

- *Let* H_1 *be a parity check matrix for the* $[m, k_1, 2\ell + 1]_2$ *code* $\mathcal{C}^1$ *(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* $\mathcal{C}^2$ *defined over the alphabet of size* $GF(2)^{m-k_1}$*.*

[1] J.K. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Trans. On Information Theory*, vol. 8, pp. 163-169, April 1965.

Tensor Product Codes [1]

Theorem

- *Let* H_1 *be a parity check matrix for the* $[m, k_1, 2\ell + 1]_2$ *code* $\mathcal{C}^1$ *(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* $\mathcal{C}^2$ *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.*

[1] J.K. Wolf, "On codes derivable from the tensor product of check matrices," *IEEE Trans. On Information Theory*, vol. 8, pp. 163-169, April 1965.

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 $\mathcal{C}_1$ 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 $\mathcal{C}_1$ 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

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 $\mathcal{C}_1$ 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 $[t_1, t_2; \ell_1, \ell_2]$-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.

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

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

Discussion

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

Discussion

- Using sphere-packing bound argument, it follows that the excess redundancy of $\mathcal{C}_B$ 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 $(\ell_2 - \ell_1)t_1/t_2 > \log(n)/\log(m)$.

Discussion

- Using sphere-packing bound argument, it follows that the excess redundancy of $\mathcal{C}_B$ 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.

Evaluation

- For TLC Flash, we compared a $[3, 2; 1, 3]_8$-graded-bit-error-correcting code $\mathcal{C}$ of length 256 with rate 0.904 against the following codes:

Evaluation

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

Evaluation

- For TLC Flash, we compared a $[3,2;1,3]_8$-graded-bit-error-correcting code $\mathcal{C}$ 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.

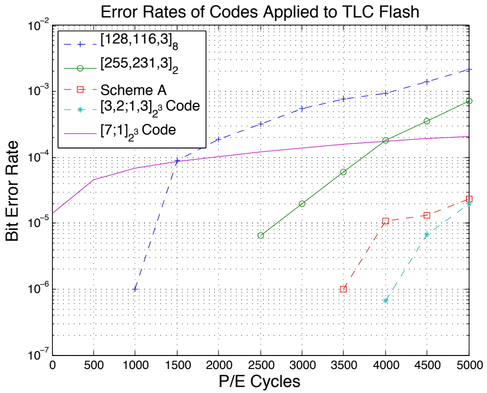
Evaluation

- For TLC Flash, we compared a $[3, 2; 1, 3]_8$-graded-bit-error-correcting code $\mathcal{C}$ 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 $\mathcal{C}^2$ applied to the LSB and the CSB for each Flash memory cell. An independent binary $[256, 240, 5]_2$ code $\mathcal{C}^3$ 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 $\mathcal{C}$ 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 $\mathcal{C}^2$ applied to the LSB and the CSB for each Flash memory cell. An independent binary $[256, 240, 5]_2$ code $\mathcal{C}^3$ was applied to the MSB for each Flash memory cell. The overall rate is 0.904.
- Constituents of $\mathcal{C}$ are $\mathcal{C}^1$ as $[3, 0, 3]_2$ (with $\mathcal{C}'_1$ as repetition code), and $\mathcal{C}^2$ and $\mathcal{C}^3$ from Scheme A.

Results


Error Rates of Codes Applied to TLC Flash
Bit Error Rate
P/E Cycles
$[128,116,3]_8$
$[255,231,3]_2$
Scheme A
$[3,2;1,3]_{2^3}$ Code
$[7;1]_{2^3}$ Code
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Conclusion

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

Conclusion

- 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.