Coding for Unreliable Flash Memory Cells

Ryan Gabrys, Lara Dolecek

Laboratory for Robust Information Systems (LORIS)
Department of Electrical Engineering, UCLA

Flash Memory Basics

- Flash memory is comprised of a set of floating gate cells.

Flash Memory Basics

- 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 Basics

- 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

Flash Memory Basics

- 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.

Flash Memory Basics

- 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.

Flash Memory Basics

- 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.

Flash Memory Basics

- 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.

Mapping Information to Voltage Levels in TLC

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

Data Collection

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

Data Collection

- On the first of every 100 P/E cycles the following was performed:
  1. Erase the block. (block= 2^{20} cells)

Data Collection

- 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

- 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.

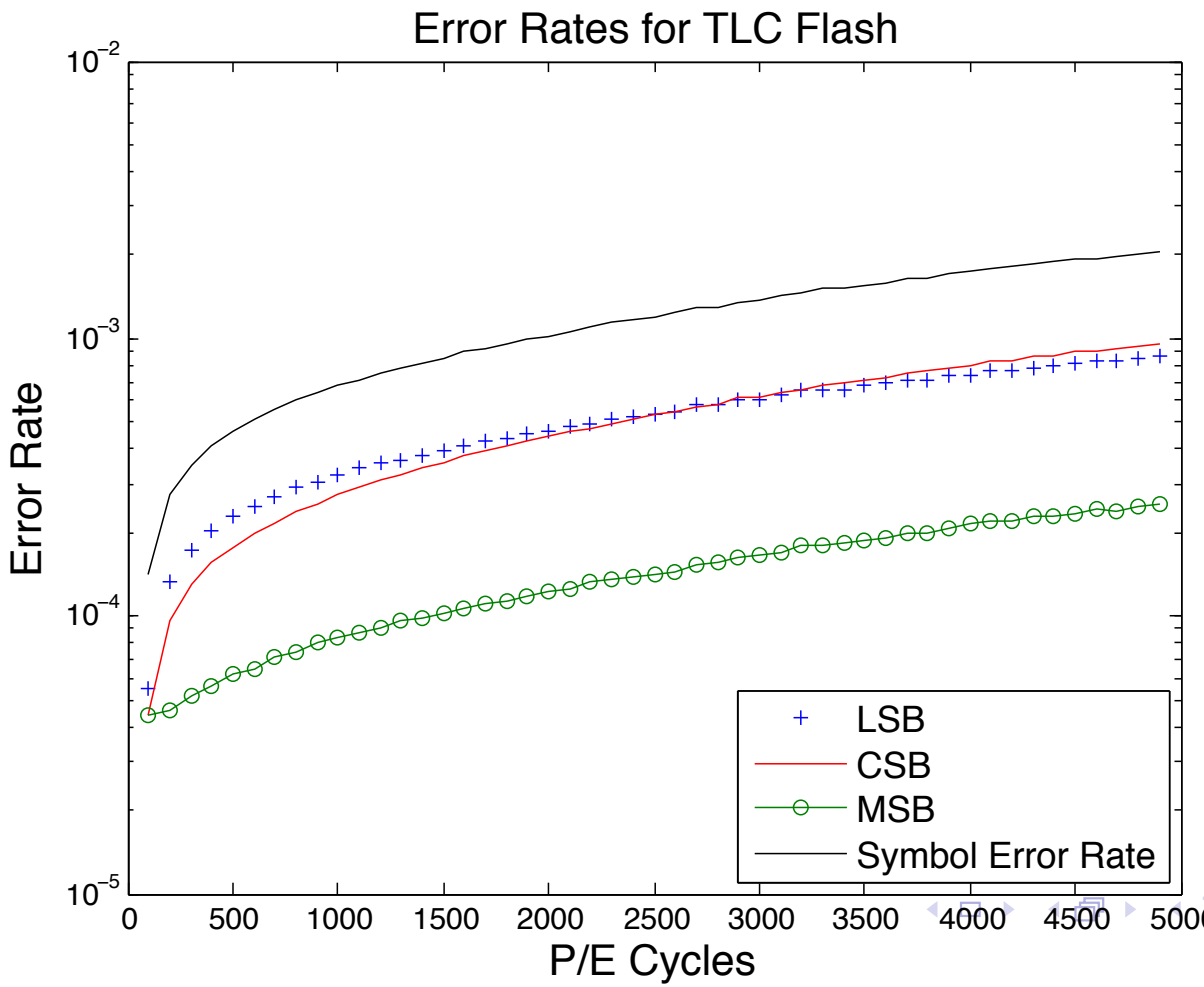
Data Collection

- 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

- 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



Observation 1: Error Patterns Within a Symbol

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

Observation 2: Unreliable Flash Cells

- We organized the Flash cells into two categories:

Observation 2: Unreliable Flash Cells

- We organized the Flash cells into two categories:
  1. reliable Flash cells,

Observation 2: Unreliable Flash Cells

- We organized the Flash cells into two categories:
  1. reliable Flash cells,
  2. unreliable Flash cells.

Observation 2: Unreliable Flash Cells

- We organized the Flash cells into two categories:
  1. reliable Flash cells,
  2. unreliable Flash cells.
- An unreliable Flash cell is a cell that experienced ≥ 50 errors across the lifetime of the device.

Observation 2: Unreliable Flash Cells

- We organized the Flash cells into two categories:
  1. reliable Flash cells,
  2. unreliable Flash cells.
- An unreliable Flash cell is a cell that experienced ≥ 50 errors across the lifetime of the device.
- We identified 62659 (of the 134217728 total cells tested) unreliable Flash cells.

Observation 2: Unreliable Flash Cells

- We organized the Flash cells into two categories:
  1. reliable Flash cells,
  2. unreliable Flash cells.
- An unreliable Flash cell is a cell that experienced ≥ 50 errors across the lifetime of the device.
- We identified 62659 (of the 134217728 total cells tested) unreliable Flash cells.
- These cells accounted for over 10% of the total errors measured.

Observation 3: Unreliable Flash Cells Have Prominent Error Patterns

Programmed state	Percentage of Errors
000	0.4745
000	0.1630
010	0.0711
000	0.0676
001	0.0558
001	0.0525
011	0.0186

High Voltage

011
010
000
001
101
100
110
111

Low Voltage

Main Idea

- Design error-correction scheme that takes into account Observations 1, 2, and 3.

Main Idea

- Design error-correction scheme that takes into account Observations 1, 2, and 3.
  1. Code corrects errors where most of the errors affect only a single bit within each Flash cell (Observation 1).

Main Idea

- Design error-correction scheme that takes into account Observations 1, 2, and 3.
  1. Code corrects errors where most of the errors affect only a single bit within each Flash cell (Observation 1).
  2. Code allows unreliable Flash cells to be programmed at low voltage levels (Observations 2,3).

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 (Graded Bit-Error Vector)

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

Error Vectors

Definition (Graded Bit-Error Vector)

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

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

Error Vectors

Definition (Graded Bit-Error Vector)

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

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

Error Vectors

Definition (Graded Bit-Error Vector)

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

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

Graded Bit-Error Example

- Suppose the vector $\boldsymbol{x}$ of length 6 with 3-bit symbols shown below was transmitted.

  $\boldsymbol{x} = (000\ 110\ 010\ 101\ 000\ 111)$

Graded Bit-Error Example

- Suppose the vector $\boldsymbol{x}$ of length 6 with 3-bit symbols shown below was transmitted.

  $\boldsymbol{x} = (000\ 110\ 010\ 101\ 000\ 111)$

- The vector $\boldsymbol{y}$ was received:

  $\boldsymbol{y} = (101\ 110\ 000\ 101\ 000\ 011)$.

Graded Bit-Error Example

- Suppose the vector $\boldsymbol{x}$ of length 6 with 3-bit symbols shown below was transmitted.

  $\boldsymbol{x} = (000\ 110\ 010\ 101\ 000\ 111)$

- The vector $\boldsymbol{y}$ was received:

  $\boldsymbol{y} = (101\ 110\ 000\ 101\ 000\ 011)$.

- Then $[2, 1; 1, 2]$-bit-errors occurred.

Graded Bit-Error Example

- Suppose the vector $\boldsymbol{x}$ of length 6 with 3-bit symbols shown below was transmitted.

  $\boldsymbol{x} = (000\ 110\ 010\ 101\ 000\ 111)$

- The vector $\boldsymbol{y}$ was received:

  $\boldsymbol{y} = (101\ 110\ 000\ 101\ 000\ 011)$.

- Then $[2, 1; 1, 2]$-bit-errors occurred.
  - There are $2 + 1$ **symbols in error.**

Graded Bit-Error Example

- Suppose the vector $\boldsymbol{x}$ of length 6 with 3-bit symbols shown below was transmitted.

  $\boldsymbol{x} = (000\ 110\ 010\ 101\ 000\ 111)$

- The vector $\boldsymbol{y}$ was received:

  $\boldsymbol{y} = (101\ 110\ 000\ 101\ 000\ 011)$.

- Then $[2, 1; 1, 2]$-bit-errors occurred.
  - There are $2 + 1$ **symbols in error.**
  - **At most** 2 **bits** are in error for each symbol.

Graded Bit-Error Example

- Suppose the vector $\boldsymbol{x}$ of length 6 with 3-bit symbols shown below was transmitted.

  $\boldsymbol{x} = (000\ 110\ 010\ 101\ 000\ 111)$

- The vector $\boldsymbol{y}$ was received:

  $\boldsymbol{y} = (101\ 110\ 000\ 101\ 000\ 011)$.

- Then $[2, 1; 1, 2]$-bit-errors occurred.
  - There are $2 + 1$ **symbols in error.**
  - **At most** 2 **bits** are in error for each symbol.
  - There is 1 **symbol that has more than** 1 **bit** in error.

Dynamic Bit-Error-Correcting Codes

- Let $\mathcal{C}$ be a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$ code.

Dynamic Bit-Error-Correcting Codes

- Let $\mathcal{C}$ be a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$ code.
- The encoder chooses a message $w \in \mathbb{Z}_M$ for some positive integer M.

Dynamic Bit-Error-Correcting Codes

- Let $\mathcal{C}$ be a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$ code.
- The encoder chooses a message $w \in \mathbb{Z}_M$ for some positive integer M.
- For any collection of indices $\mathcal{S}$ where $|\mathcal{S}| \leq s_1$, there exists a codeword $\boldsymbol{c}$ that can be used to represent w such that:

Dynamic Bit-Error-Correcting Codes

- Let $\mathcal{C}$ be a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$ code.
- The encoder chooses a message $w \in \mathbb{Z}_M$ for some positive integer M.
- For any collection of indices $\mathcal{S}$ where $|\mathcal{S}| \leq s_1$, there exists a codeword $\boldsymbol{c}$ that can be used to represent w such that:
  - For any $j \in \mathcal{S}$, the voltage level of cell $\boldsymbol{c}_j$ is at most s_2.

Dynamic Bit-Error-Correcting Codes

- Let $\mathcal{C}$ be a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$ code.
- The encoder chooses a message $w \in \mathbb{Z}_M$ for some positive integer M.
- For any collection of indices $\mathcal{S}$ where $|\mathcal{S}| \leq s_1$, there exists a codeword $\boldsymbol{c}$ that can be used to represent w such that:
  - For any $j \in \mathcal{S}$, the voltage level of cell $\boldsymbol{c}_j$ is at most s_2.
- The code $\mathcal{C}$ is referred to as an $[t_1, t_2; \ell_1, \ell_2; s_1, s_2]_{2^m}$ **dynamic-bit-error-correcting code**.

Evaluation

- We compared a dynamic graded-bit-error-correcting code against the following classes of codes:

Evaluation

- We compared a dynamic graded-bit-error-correcting code against the following classes of codes:
  1. A non-binary code over $GF(8)$.

Evaluation

- We compared a dynamic graded-bit-error-correcting code against the following classes of codes:
  1. A non-binary code over $GF(8)$.
  2. A binary BCH code applied to MSB/CSB/LSB in parallel.

Evaluation

- We compared a dynamic graded-bit-error-correcting code against the following classes of codes:
  1. A non-binary code over $GF(8)$.
  2. A binary BCH code applied to MSB/CSB/LSB in parallel.
  3. Three different binary BCH codes each one applied to one of the MSB/CSB/LSB.

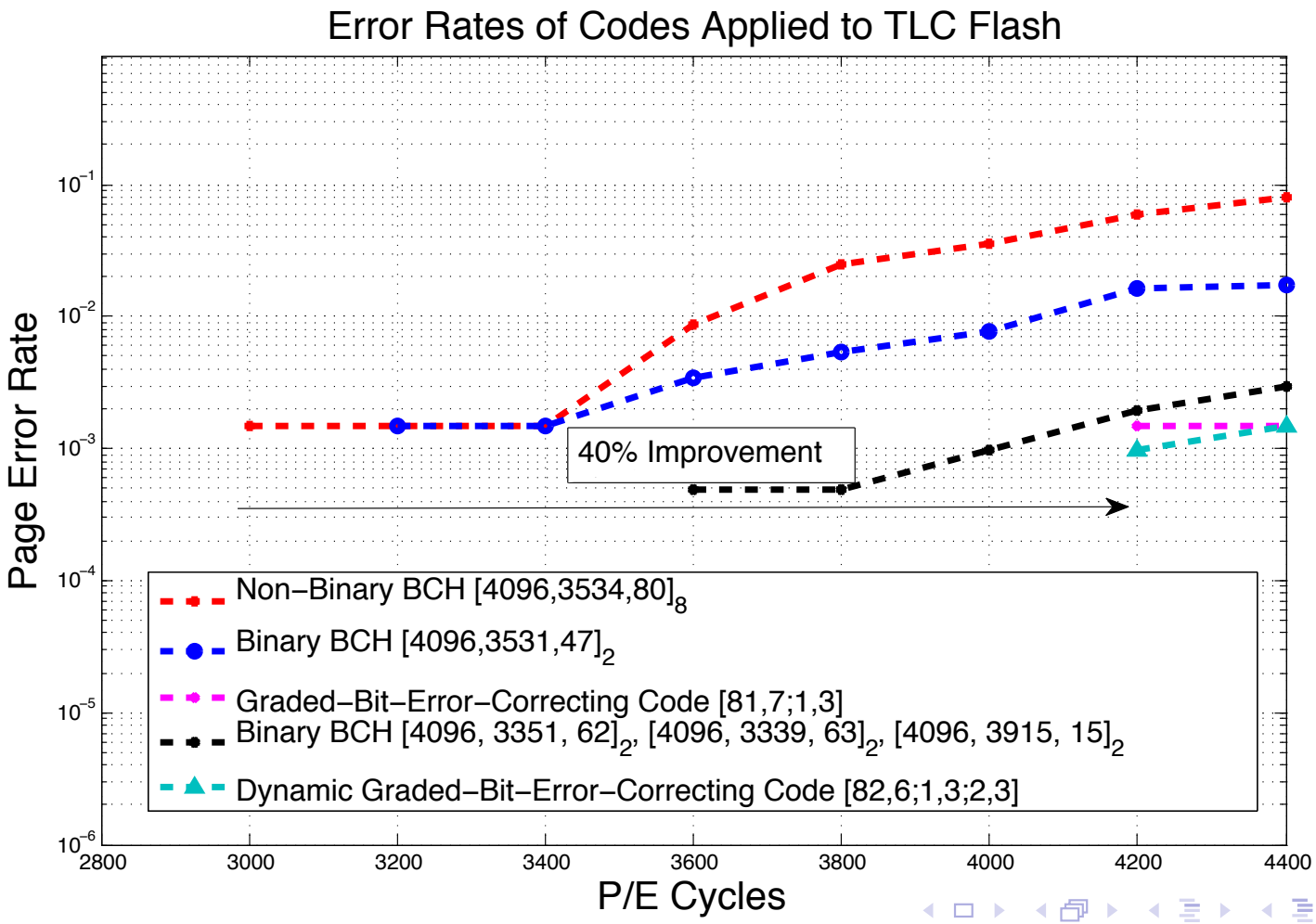
Evaluation

- We compared a dynamic graded-bit-error-correcting code against the following classes of codes:
  1. A non-binary code over $GF(8)$.
  2. A binary BCH code applied to MSB/CSB/LSB in parallel.
  3. Three different binary BCH codes each one applied to one of the MSB/CSB/LSB.
  4. A $[t_1, t_2; \ell_1, \ell_2]_{2^3}$-bit-error-correcting code.

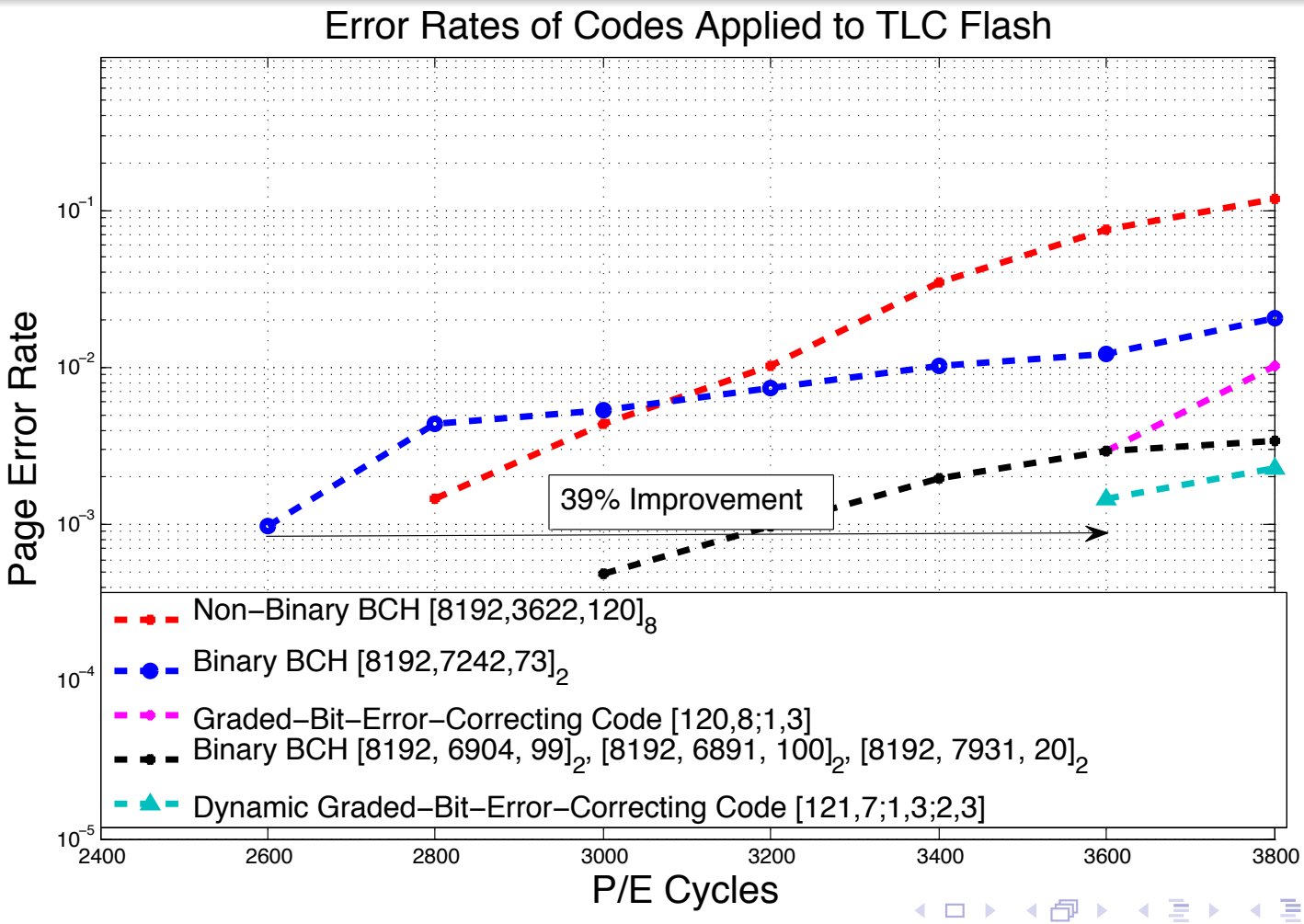
Evaluation

- We compared a dynamic graded-bit-error-correcting code against the following classes of codes:
  1. A non-binary code over $GF(8)$.
  2. A binary BCH code applied to MSB/CSB/LSB in parallel.
  3. Three different binary BCH codes each one applied to one of the MSB/CSB/LSB.
  4. A $[t_1, t_2; \ell_1, \ell_2]_{2^3}$-bit-error-correcting code.
- For each graph, the codes compared have the same length and the same rate.

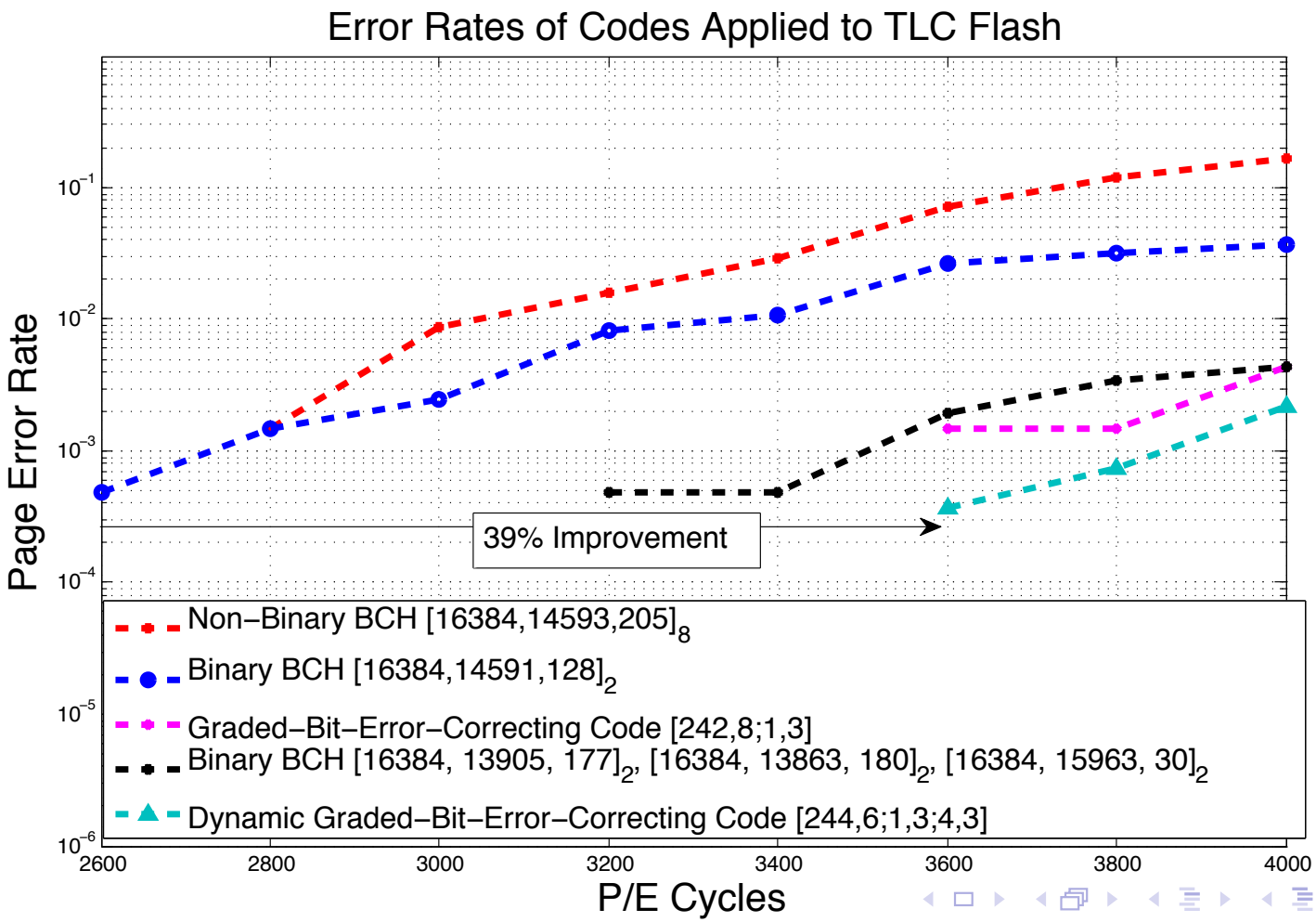
Results for block length 4096



Results for block length 8192



Results for block length 16384



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.
- Errors that occur within these newer Flash devices tend to follow certain patterns.

Conclusion

- Newer generations of Flash memory continue to demand more efficient error-correction schemes.
- Errors that occur within these newer Flash devices tend to follow certain patterns.
- By taking into account the dominant error patterns observed on a TLC Flash cell, we designed more efficient error correction codes.

Thank You

New center on Coding for Storage at UCLA:
http://www.loris.ee.ucla.edu/codess

Kick-off day on 9/19/2013!

Registration is free.
Register early, space is limited.