Making Error Correcting Codes Work for Flash Memory
Part III: New Coding Methods

Anxiao (Andrew) Jiang

Department of Computer Science and Engineering
Texas A&M University

Tutorial at Flash Memory Summit, August 4, 2014
Acknowledgment to: Wenyi Zhu (for artistic illustrations)
Constrained coding
Constrained Coding
Constrained coding for inter-cell interference

Inter-cell interference in flash memory:
The $V_{th}$ shift of middle cell caused by shifting of neighboring cells is

$$
\Delta V_{i,j} = C_x (\Delta V_{i-1,j} + \Delta V_{i+1,j}) + C_y (\Delta V_{i,j-1} + \Delta V_{i,j+1})
+ C_{x,y} (\Delta V_{i-1,j-1} + \Delta V_{i+1,j-1} + \Delta V_{i-1,j+1} + \Delta V_{i+1,j+1})
$$
One constraint to set for $q$-level cells: The difference between adjacent levels cannot be too large. A concrete example: Avoid $(q - 1)0(q - 1)$ pattern for adjacent cell levels.

Minghai Qin, Eitan Yaakobi, and Paul Siegel, “Constrained codes that mitigate intercell interference in read/write cycles for flash memories,” in JSAC Special Issue, May 2014.

K. A. S. Immink, “Coding schemes for multi-level channels with unknown gain and/or offset,” in ISIT 2013.


Rewriting: Change the value of the stored data.

Requirement: The cell levels can only increase, not decrease, in order to avoid block erasures.

Objective: Maximize the number of times the data are rewritten, or maximize the summation of the code rates over the multiple rewrites.

Papers in ISIT 2007:
Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.

Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.
Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.

Cell Levels:
1st write: 10
2nd write: 01

Data:
00
11
01
00
11
01
00
11
01
00
Write Once Memory (WOM)

Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.

Sum rate: \( \frac{2}{3} + \frac{2}{3} = 1.33 \)
For WOM of $q$-level cells and $t$ rewrites, the capacity (maximum achievable sum rate) is

$$\log_2 \binom{t + q - 1}{q - 1}.$$ bits per cell.


Capacity of WOM

Graph showing the capacity over time for WOM and Ordinary with different q values.

- WOM-q=2
- WOM-q=4
- WOM-q=8
- Ordinary-q=2
- Ordinary-q=4
- Ordinary-q=8

Axes:
- Capacity on the y-axis
- Time (t) on the x-axis
For Rewriting to be used in flash memories, it is CRITICAL to combine it with Error-Correcting Codes.
A joint coding scheme for rewriting and error correction, which can correct a substantial number of errors and supports any number of rewrites.

Model of rewriting and noise:

1st write $\rightarrow$ BSC(p) $\rightarrow$ 2nd write $\rightarrow$ BSC(p) $\rightarrow$ 0 0 0 $\rightarrow$ t-th write $\rightarrow$ BSC(p)
Lower bound to achievable sum-rate (for WOM):

![Graph showing lower bound to achievable sum-rate for different error probability p.]

Fig. 6. Lower bound to achievable sum-rates for different error probability $p$. 
Further reading

- A. Shpilka, Capacity achieving multiwrite WOM codes, 2012.
Rank Modulation

- Constrained Coding
- WOM
- Rewrite
- ECC
Rank Modulation
Parallel cell programming for MLC
Muti-level cell (MLC): Parallel programming, common thresholds, heterogeneous cells, random process of charge injection, over-injection of charge, disturbs and inter-cell interference, block erasure, difficulty in adjusting threshold voltages, very careful repeated charge injection and measuring.
Challenges of parallel cell programming for MLC

Dilemma among:

- Capacity
- Speed
- Reliability and endurance

Due to: Inflexibility in adjusting cell levels.
Definition (Rank Modulation)

Use the relative order of cell levels to represent data.

Some advantages of rank modulation:

1. Flexibility in adjusting relative cells levels, even though we can only increase cell levels;
2. Tolerance for charge leakage / cell level drifting;
3. Enable memory scrubbing without block erasure.
2. Extended models of rank modulation
Extension: Rank modulation with multiple permutations

Some advantages: (1) Enable the building of long codes; (2) Cells in different permutations can have very close cell levels.

Extension: Rank modulation with multi-set permutation

Example: A group of $n = 6$ cells

Some advantages: Similar to multiple permutations, but more suitable if cells can be programmed accurately.
Example: Every rank has one cell
Example: Every rank has two cells
Example: Every rank has three cells
• **Extension: Bounded rank modulation**
  

• **Extension: Local rank modulation**
  

• **Extension: Partial rank modulation:**
  

Some advantages: Faster read, and/or enabling long codewords.
Rewrite

Rank Modulation

Constrained Coding

WOM

Rewrite

ECC
Definition (Rewrite)

Change data by changing the permutation – by moving cell levels up.
Virtual levels to help us estimate rewriting cost (increase in cell levels).
Get the permutation right from low to high.
Get the permutation right from low to high.
Get the permutation right from low to high.
Rewriting cost: 1.
Consider: Store data of $k$ values in $n$ cells.
Every subset of permutations represents one value of the data.
Consider one such subset, which represents one particular data value.
Say the red dot is the current state of the $n$ cells. We want to change the data to the value represented by the green subset $\cdots$
Bound the rewriting cost by $r$. 

\[ \leq r \]
The green subset needs to be a *dominating set* of incoming covering radius $r$. 
We show an optimal code as an example. Parameters: $n = 4$ cells, $k = 6$ data values, rewriting cost $r = 1$.

| 1 2 3 4 | 2 3 4 1 | 3 4 1 2 | 4 1 2 3 |
| 1 2 4 3 | 2 4 3 1 | 4 3 1 2 | 3 1 2 4 |
| 1 3 2 4 | 3 2 4 1 | 2 4 1 3 | 4 1 3 2 |
| 1 3 4 2 | 3 4 2 1 | 4 2 1 3 | 2 1 3 4 |
| 1 4 2 3 | 4 2 3 1 | 2 3 1 4 | 3 1 4 2 |
| 1 4 3 2 | 4 3 2 1 | 3 2 1 4 | 2 1 4 3 |

<table>
<thead>
<tr>
<th>1 2 3 4</th>
<th>2 3 4 1</th>
<th>3 4 1 2</th>
<th>4 1 2 3</th>
<th>Subgroup&lt;\langle1,2,3,4\rangle&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 3</td>
<td>2 4 3 1</td>
<td>4 3 1 2</td>
<td>3 1 2 4</td>
<td>Coset</td>
</tr>
<tr>
<td>1 3 2 4</td>
<td>3 2 4 1</td>
<td>2 4 1 3</td>
<td>4 1 3 2</td>
<td>Coset</td>
</tr>
<tr>
<td>1 3 4 2</td>
<td>3 4 2 1</td>
<td>4 2 1 3</td>
<td>2 1 3 4</td>
<td>Coset</td>
</tr>
<tr>
<td>1 4 2 3</td>
<td>4 2 3 1</td>
<td>2 3 1 4</td>
<td>3 1 4 2</td>
<td>Coset</td>
</tr>
<tr>
<td>1 4 3 2</td>
<td>4 3 2 1</td>
<td>3 2 1 4</td>
<td>2 1 4 3</td>
<td>Coset</td>
</tr>
</tbody>
</table>
Every row (subgroup) is a dominating set of radius 1.
Every row (subgroup) is a dominating set of radius 1.

So we can map the 6 cosets to 6 data values. The code has a bounded rewriting cost of 1.
1. Model errors: Noise modeling, and error quantization.

2. Design ECC.
Kendall-\(\tau\) distance
Definition (Kendall-\(\tau\) distance)

The number of adjacent transpositions to change one permutation into another. (The distance is symmetric.)

Example

For permutations \(\alpha = [2, 1, 3, 4]\) and \(\beta = [2, 3, 4, 1]\), the Kendall-\(\tau\) distance \(d_{\tau}(\alpha, \beta) = 2\) because

\([2, 1, 3, 4] \rightarrow [2, 3, 1, 4] \rightarrow [2, 3, 4, 1]\).
We can define an adjacency graph for permutations based on Kendall-$\tau$ distance.

**Example**

Permutations $S_n$ with $n = 4$. 

![Graph of permutations](image)
An technique for ECC construction: Embedding

Other techniques: Interleaving (product of sub-codes), modular (for limited-magnitude errors), etc.
Theorem

The adjacency graph for permutations is a subgraph of an \((n - 1)\)-dimensional array, whose size is \(2 \times 3 \times \cdots \times n\).

Construction (One-Error-Correcting Rank Modulation Code)

Let $C_1, C_2 \subseteq S_n$ denote two rank modulation codes constructed as follows. Let $A \in S_n$ be a general permutation whose inversion vector is $(x_1, x_2, \cdots, x_{n-1})$. Then $A$ is a codeword in $C_1$ iff the following equation is satisfied:

$$\sum_{i=1}^{n-1} ix_i \equiv 0 \pmod{2n-1}$$

$A$ is a codeword in $C_2$ iff the following equation is satisfied:

$$\sum_{i=1}^{n-2} ix_i + (n - 1) \cdot (-x_{n-1}) \equiv 0 \pmod{2n-1}$$

Between $C_1$ and $C_2$, choose the code with more codewords as the final output.
For the above code, it can be proved that:

- The code can correct one Kendall error.
- The size of the code is at least $\frac{(n-1)!}{2}$.
- The size of the code is at least half of optimal.
Codes correcting more Kendall errors are constructed based on embedding.

First, consider codes of the following form:

- Let $m \geq n - 1$ and let $h_1, \cdots, h_{n-1}$ be a set of integers, where $0 < h_i < m$ for $i = 1, \cdots, n-1$. Define the code as follows:

$$
\mathcal{C} = \left\{ (x_1, x_2, \cdots, x_{n-1}) \mid \sum_{i=1}^{n-1} h_i x_i \equiv 0 \pmod{m} \right\}
$$

Let the number of cells $n \to \infty$. Consider capacity.

**Theorem (Capacity of Rank Modulation ECC with $n \to \infty$)**

Let $A(n, d)$ be the maximum number of permutations in $S_n$ with minimum Kendall-tau distance $d$. We call

$$C(d) = \lim_{n \to \infty} \frac{\ln A(n, d)}{\ln n!}$$

the capacity of rank modulation ECC of Kendall-tau distance $d$. Then,

$$C(d) = \begin{cases} 
1 & \text{if } d = O(n) \\
1 - \epsilon & \text{if } d = \Theta(n^{1+\epsilon}), \ 0 < \epsilon < 1 \\
0 & \text{if } d = \Theta(n^2)
\end{cases}$$

Further reading


Further reading

Further reading

Further Reading

- M. Schwartz and I. Tamo, “Optimal permutation anticodes with the infinity norm via permanents of (0, 1)-matrices,” in *Journal of Combinatorial Theory*, 2011.
Performance of SLC, MLC and TLC:

- SLC: 2 levels, endurance of $\sim 10^5$ Program/Erase cycles.
- MLC: 4 levels, endurance of $\sim 10^4$ Program/Erase cycles.
- TLC: 8 levels, endurance of $\sim 10^3$ Program/Erase cycles.

Question: *Is there a way to adaptively choose the number of levels, based on the cells’ quality and random programming performance?*
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.

Existing Technology: Fixed Thresholds and Levels

Cell-level Distribution of TLC

level 0 level 1 level 2 level 3 level 4 level 5 level 6 level 7

T1 T2 T3 T4 T5 T6 T7
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Variable Level Cell (VLC)

Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Main Idea of VLC:

- Set thresholds dynamically.
- Do not fix the number of levels in advance.
Variable Level Cell (VLC)

- VLC is more adaptive compared to current schemes.
- Programming is more robust to
  - Cell quality degradation/variance;
  - Probabilistic charge injection behavior.
- Multiple levels can be programmed in parallel for higher speed.
How to store data? One solution for one-write storage:

Cell-level Distribution of VLC

n cells

level 0
- Level 1 can store $nH(x_1)$ bits.
- Reading these $nH(x_1)$ bits will require two threshold comparisons.

Cell-level Distribution of VLC

![Graph showing cell-level distribution of VLC]

$n(1-x_1)$ cells

$n x_1$ cells

level 0  level 1
Level 2 can store $n(1 - x_1)H(x_2)$ bits.

Reading these $n(1 - x_1)H(x_2)$ bits will require one additional threshold comparison.

Cell-level Distribution of VLC
Assume

- Level 1 can be programmed with probability $p_1$;
- Level 2 can be programmed with probability $p_1p_2$;
- Level 3 can be programmed with probability $p_1p_2p_3$;
- \ldots;
- Level $q$ can be programmed with probability $p_1p_2\cdots p_q$, where $q$ is the maximum possible level number.
Define $A_1, A_2, \cdots, A_{q-1}$ recursively:

- Let $A_{q-1} = 2^{p(q-1)}$;
- For $i = q - 2, q - 3, \cdots, 1$, let $A_i = (1 + A_{i+1})^{p_i}$.

**Theorem**

The capacity (expected value) of VLC is

$$C_{VLC} = \log_2 A_1$$

bits per cell.

For the capacity region of rewriting codes, see:
