Coding for Flash Memories

Anxiao (Andrew) Jiang

IAP, Texas A&M University, 03/03/2009
Storage Technologies

- Magnetic recording
- Optical recording
- Electronic memories
Applications of Flash Memories

- Flash memory is the major type of NVM (more than 90% of NVM market)
  - mobile devices
  - embedded systems
  - flash disks
Physics of Flash Memory

Cell array in a flash memory

A flash memory cell (Floating gate)

- A flash memory cell can store charge. And the charge level represents data.
How does a cell store a bit?
How does a cell store a bit?

- **Inject electrons:** Hot electron injection mechanism, or Fowler-Nordheim tunneling mechanism
- **Remove electrons:** Fowler-Nordheim tunneling mechanism
Single-level cell and Multi-level cell

- **Single-level cell:** Two levels $\rightarrow$ One bit
  - 0
  - 1

- **Multi-level cell:** $q$ levels $\rightarrow \log_2 q$ bits
  - 0
  - 1
  - 2
  - $\cdots$
  - $q-1$
Block Erasure is Costly

- Flash cells are organized as blocks. A block has about 100,000 cells.
- To reduce any cell’s level, the whole block has to be erased, and then reprogrammed.
- Block erasure is very costly:
  1. *Longevity problem*: Every block can tolerate only 100,000 erasures.
  2. *Block erasures reduce speed* and consumes power.
Rewriting Data: Consider discrete cell levels

- The cell level can only increase (from 0 to $q-1$).
- Traditional method: Changing one bit can lead to block erasure. It is very costly.
- To write and rewrite two bits, how many cells are needed?
Floating codes

[Jiang et al. 07] [Jiang et al. 08] [Jiang et al. 09]

- Definition: K variables of alphabet size L are stored in n cells of q levels. Each rewrite changes the value of one variable. Design a code. The objective is to maximize the number of rewrites guaranteed in the worst case.

- Performance: n is large $\Rightarrow$ O(n) rewrites

- Performance: k, L are large $\Rightarrow$ roughly $\frac{nq \lg n}{\lg K + \lg L}$ rewrites.

Buffer codes: for stream data [Bohossian, Jiang, Bruck 07]

General codes: [Jiang et al. 09]
Cell Programming

- Noisy, monotonic


- Trend: more levels, smaller cells
- Question: How to write data reliably when cells cannot be programmed reliably?
- Challenges: overshoot, worst-case constraint.

- Approach: adaptive cell-ensemble programming.
- Rank modulation is such an approach.
Rank Modulation

- Analog cell levels induce permutations. 
  
  Example: 3 cells can induce $3! = 6$ permutations

- Permutations represent data.

- Method of programming: from low to high.

- Advantage: no overshoot, adaptive coding.
Rewrite, error correction, and mimicking MLC

- **Rewrite**: How to rewrite data in the rank modulation scheme?

- **Error correction**: How to design error-correcting codes? What does error mean?

- How to use rank modulation to mimic MLC, so that previous results (e.g., floating codes) can be used by rank modulation?
Rewrite data using rank modulation

- An example code: 4 cells, data alphabet size = 9

  Mapping: 1xxx : 1  2xxx : 2  31xx : 3
  32xx : 4  34xx : 5  41xx : 6
  42xx : 7  431x : 8  432x : 9

  If the current permutation is 1234, to change the data from 1 to 4, we need to push cell 2 and cell 3 to the top, so that the permutation becomes 3214.

- Cost measurement: The number of “push-to-top” operations.

- Theorem: Fixed-length prefix-free code is worst-case optimal.

- Theorem: Variable-length prefix-free code is within a constant ratio from being average-case optimal.

Define the distance between permutations as the minimum number of adjacent transpositions it takes to change one permutation into the other. (Kendall Tau distance.)

Example: The distance between 2134 and 2341 is 2, because 2134 $\leftrightarrow$ 2314 $\leftrightarrow$ 2341.

Example: ECC with 3 cells.
How to design ECC for more cells?

Lehmer code

<table>
<thead>
<tr>
<th>Permutation</th>
<th>Coordinates</th>
<th>Permutation</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4</td>
<td>(0,0,0)</td>
<td>3 1 2 4</td>
<td>(0,2,0)</td>
</tr>
<tr>
<td>1 2 4 3</td>
<td>(0,0,1)</td>
<td>3 1 4 2</td>
<td>(0,2,1)</td>
</tr>
<tr>
<td>1 3 2 4</td>
<td>(0,1,0)</td>
<td>3 2 1 4</td>
<td>(1,2,0)</td>
</tr>
<tr>
<td>1 3 4 2</td>
<td>(0,1,1)</td>
<td>3 2 4 1</td>
<td>(1,2,1)</td>
</tr>
<tr>
<td>1 4 2 3</td>
<td>(0,0,2)</td>
<td>3 4 1 2</td>
<td>(0,2,2)</td>
</tr>
<tr>
<td>1 4 3 2</td>
<td>(0,1,2)</td>
<td>3 4 2 1</td>
<td>(1,2,2)</td>
</tr>
<tr>
<td>2 1 3 4</td>
<td>(1,0,0)</td>
<td>4 1 2 3</td>
<td>(0,0,3)</td>
</tr>
<tr>
<td>2 1 4 3</td>
<td>(1,0,1)</td>
<td>4 1 3 2</td>
<td>(0,1,3)</td>
</tr>
<tr>
<td>2 3 1 4</td>
<td>(1,1,0)</td>
<td>4 2 1 3</td>
<td>(1,0,3)</td>
</tr>
<tr>
<td>2 3 4 1</td>
<td>(1,1,1)</td>
<td>4 2 3 1</td>
<td>(1,1,3)</td>
</tr>
<tr>
<td>2 4 1 3</td>
<td>(1,0,2)</td>
<td>4 3 1 2</td>
<td>(0,2,3)</td>
</tr>
<tr>
<td>2 4 3 1</td>
<td>(1,1,2)</td>
<td>4 3 2 1</td>
<td>(1,2,3)</td>
</tr>
</tbody>
</table>

Embedding

Theorem. The adjacency graph of permutations is a subgraph of the $2 \times 3 \times \cdots \times n$ linear array.
Use Permutations to Mimic MLC

- Examples of 3 and 4 cells

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

|   | 0 | 1 | 2 | 3 | 4 | 5 |

- They are Gray codes.
- Theorem: Gray code always exists for permutations.
How to move data efficiently?

- Coding for wear leveling [Jiang et al. 09]
Summary

- System level: Wear leveling
- Data level: Rewriting, error correction
- Cell level: Programming, data representation

Funding:

NSF CAREER Award, NSF IHCS Program