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 12, 2013

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,
- Rank modulation scheme and its error correction,

Outline of this talk

We will learn about

- Joint rewriting and error correction scheme,
- Rank modulation scheme and its error correction,
- Summary and future directions.

$\boxed{\text{Joint rewriting and error correction scheme}}$

Concept of Rewriting

TLC: 8 Levels

No rewrite	One rewrite	Six rewrites
011	01	0
010	00	1
000	10	0
001	11	1
101	01	0
100	00	1
110	10	0
111	11	1

Concept of Rewriting

Advantage of rewriting: Longevity of memory.

Why?

- Delay block erasures.
- Trade instantaneous capacity for sum-capacity over the memory's lifetime.

Rewriting can be applied to any number of levels, including SLC.

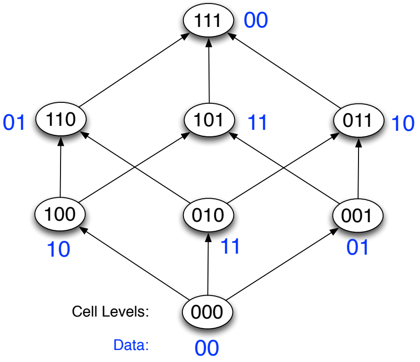
Review: Basic Problem for Write-Once Memory

Let us recall the basic question for Write-Once Memory (WOM):

- Suppose you have n binary cells. Every cell can change its value only from 0 to 1, not from 1 to 0.
  How can you write data, and then rewrite, rewrite, rewrite $\cdots$ the data?

Review: Write Once Memory (WOM) [1]

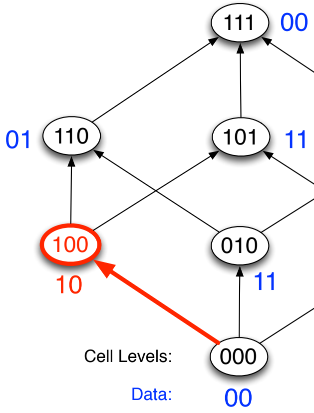
Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.



[1] R. L. Rivest and A. Shamir, "How to reuse a 'write-once' memory," in Information and Control, vol. 55, pp. 1-19, 1982.

Review: Write Once Memory (WOM)

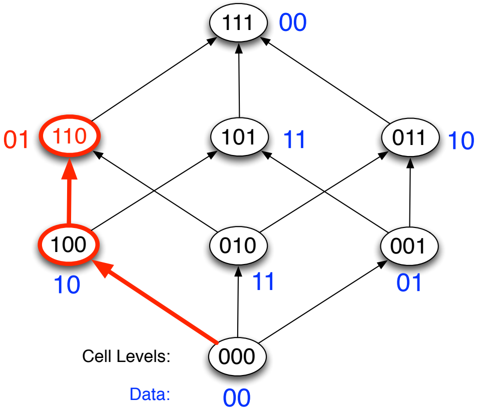
Example: Store 2 bits in 3 SLCs. Write the 2-bit data twice.



1st write: 10

Review: Write Once Memory (WOM)

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

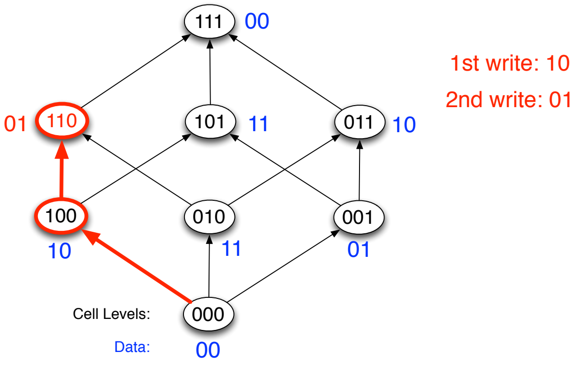


1st write: 10

2nd write: 01

Review: Write Once Memory (WOM)

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



Sum rate: $\frac{2}{3} + \frac{2}{3} = 1.33$

Review: Write-Once Memory Code

This kind of code is called Write-Once Memory (WOM) code.

It is potentially a powerful technology for Flash Memories.

Review: Capacity of WOM [1][2]

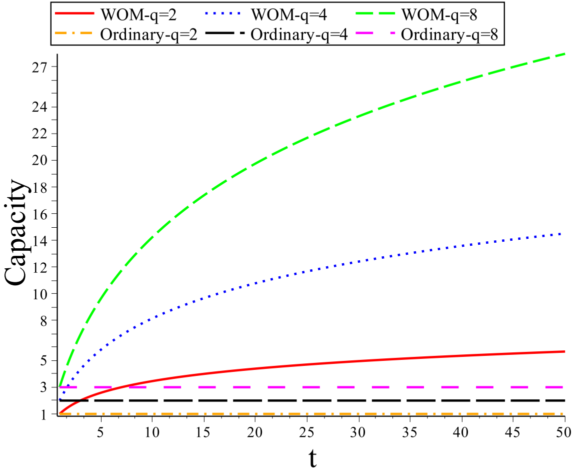
For WOM of q-level cells and t rewrites, the capacity (maximum achievable sum rate) is

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

bits per cell.

[1] C. Heegard, On the capacity of permanent memory, in *IEEE Trans. Information Theory*, vol. IT-31, pp. 34-42, 1985.
[2] F. Fu and A. J. Han Vinck, On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph, in *IEEE Trans. Information Theory*, vol. 45, no. 1, pp. 308-313, 1999.

Review: Capacity of WOM



Recent Developments

How to design good WOM codes?

Two capacity-achieving codes were published in 2012 – the same year!:

- A. Shpilka, Capacity achieving multiwrite WOM codes, 2012.
- D. Burshtein and A. Strugatski, **Polar write once memory codes**, 2012.

Two Parameters: α and ϵ

For a t-write WOM code, consider one of its t writes.

There are two important parameters for this write:

- α: The fraction of cells that are 0 before this write.
- ϵ: For the cells of level 0 before this write, ϵ is the fraction of them that are changed to 1 in this write.

For t-write WOM codes, the optimal values of α and ϵ are known for each of the t writes.

Polar WOM Code [1]

Idea of Burshtein and Strugatski: See a write as the decoding of a polar code:

- See the cells' state **BEFORE** the write as a noisy Polar codeword.
- See the cells' state **AFTER** the write as the correct (i.e., error-free) Polar codeword.

More precisely, they see the write as lossy data compression, using the method presented by Korada and Urbanke [2].

[1] D. Burshtein and A. Strugatski, Polar Write Once Memory Codes, in *Proc. ISIT*, 2012.

[2] S. Korada and R. Urbanke, Polar Codes Are Optimal For Lossy Source Coding, in *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1751–1768, 2010.

Polar WOM Code

Smart Idea by Burshtein and Strugatski:

1. Add dither to cell:
   - Let $s \in \{0, 1\}$ be the level of a cell.
   - Let $g \in \{0, 1\}$ be a pseudo-random number known to the encoder and decoder.
   - Let $v = s \oplus g$ be called the **value** of the cell.
2. Build a test channel for the write, which we shall call the WOM channel:

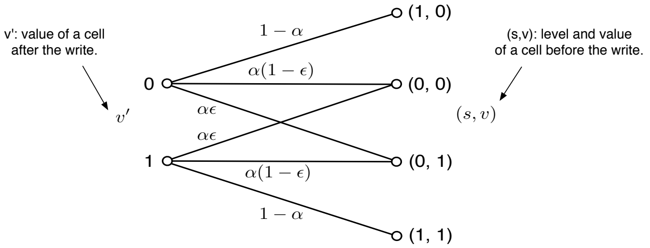
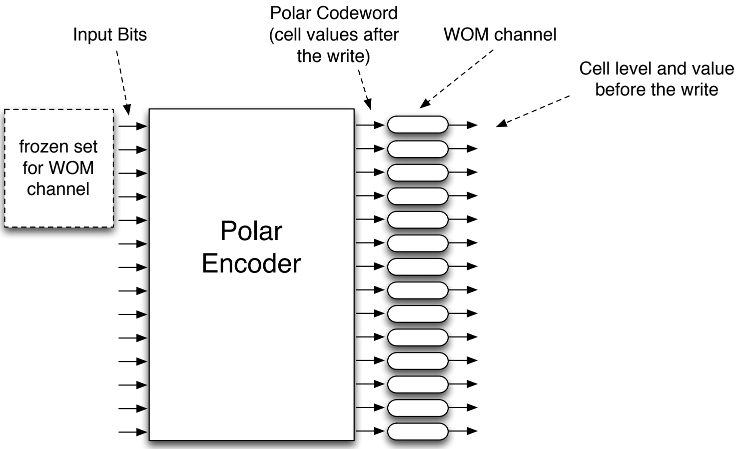
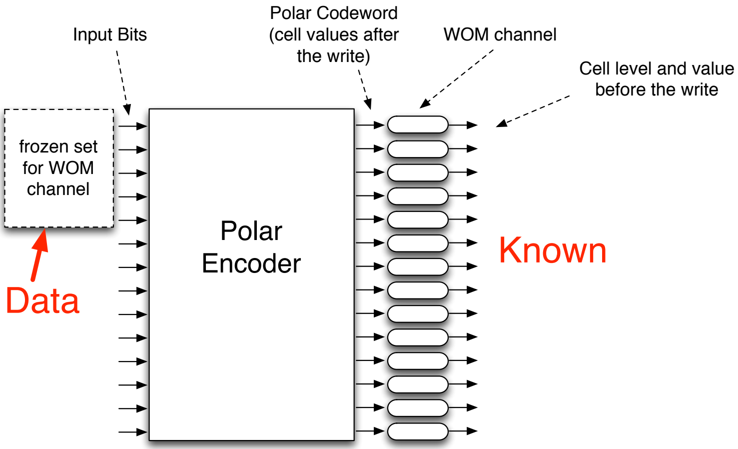


Fig. 1. The WOM channel $WOM(\alpha, \epsilon)$.

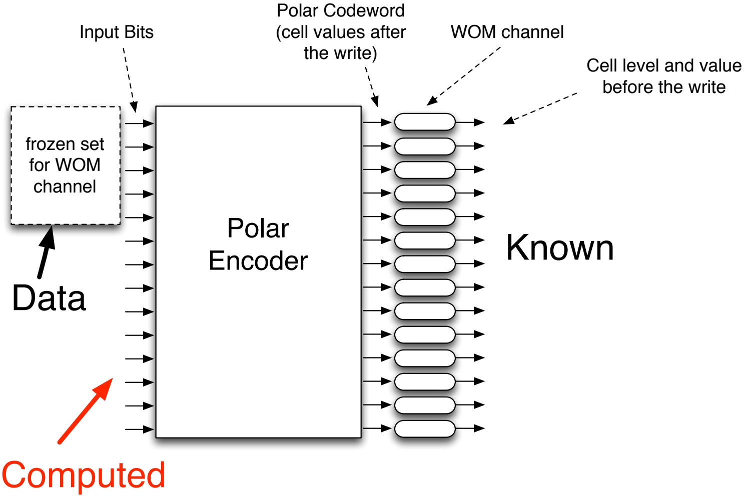
Polar WOM Code: Process of A Write: Encode



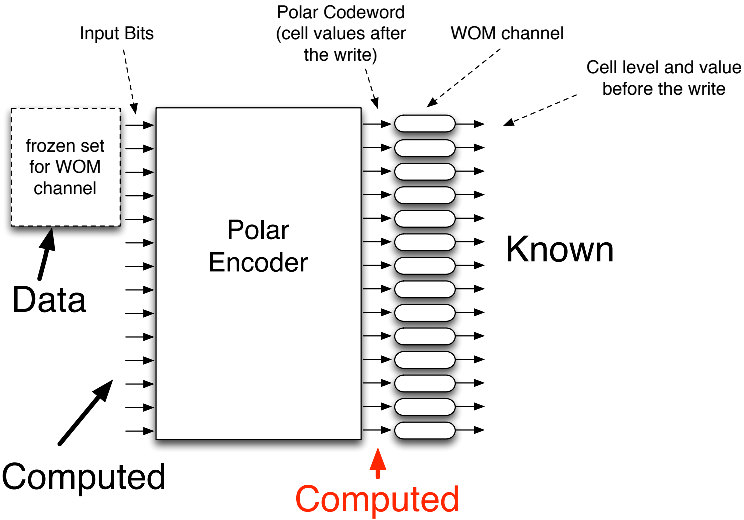
Polar WOM Code: Process of A Write: Encode



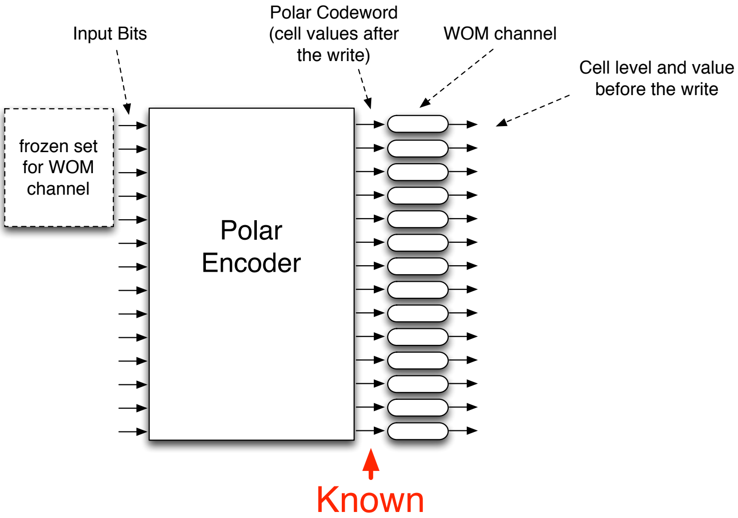
Polar WOM Code: Process of A Write: Encode



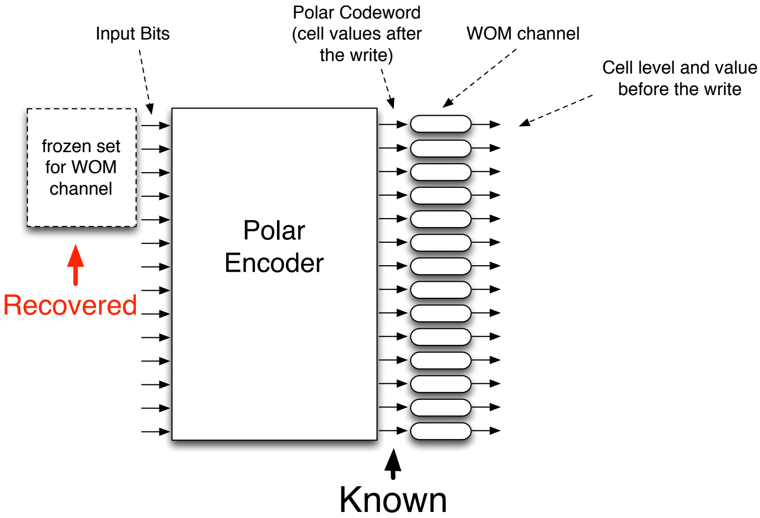
Polar WOM Code: Process of A Write: Encode



Polar WOM Code: Process of A Write: Decode



Polar WOM Code: Process of A Write: Decode



For **Rewriting** to be used in flash memories, it is **CRITICAL** to combine it with **Error-Correcting Codes**.

Some Codes for Joint Rewriting and Error Correction

Previous results are for correcting a few (up to 3) errors:

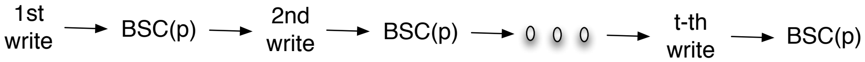
- G. Zemor and G. D. Cohen, Error-Correcting WOM-Codes, in *IEEE Transactions on Information Theory*, vol. 37, no. 3, pp. 730–734, 1991.
- E. Yaakobi, P. Siegel, A. Vardy, and J. Wolf, Multiple Error-Correcting WOM-Codes, in *IEEE Transactions on Information Theory*, vol. 58, no. 4, pp. 2220–2230, 2012.

New Code for Joint Rewriting and Error Correction

We now present a joint coding scheme for rewriting and error correction, which can correct a substantial number of errors and supports any number of rewrites.

- A. Jiang, Y. Li, E. En Gad, M. Langberg, and J. Bruck, Joint Rewriting and Error Correction in Write-Once Memories, 2013.

Model of Rewriting and Noise



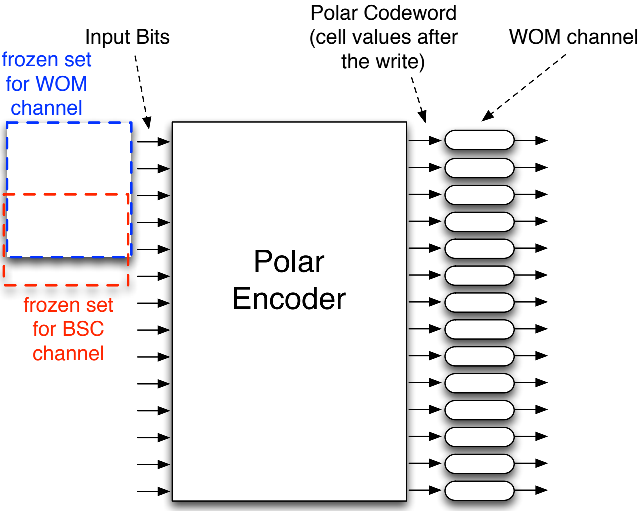
Two Channels

Consider one write.

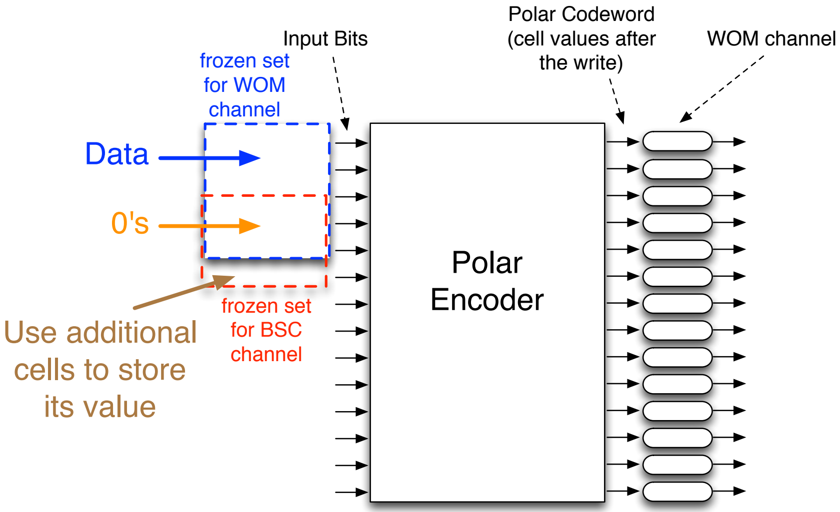
Consider two channels:

1. **WOM channel**. Let its frozen set be $F_{WOM(\alpha,\epsilon)}$.
2. **BSC channel**. Let its frozen set be $F_{BSC(p)}$.

General Coding Scheme



General Coding Scheme



Rate of the Code

Analyze the rate of a single write step:

- Let $N \to \infty$ be the size of the polar code.
- The size of $F_{WOM(\alpha,\epsilon)}$ (the frozen set for the WOM channel) is $\alpha H(\epsilon)N$.
- The size of $F_{BSC(p)}$ (the frozen set for the BSC) is $H(p)N$.
- The number of bits in the written data is $|F_{WOM(\alpha,\epsilon)} - F_{BSC(p)}|$.
- The number of additional cells we use to store the value in $F_{BSC(p)} - F_{WOM(\alpha,\epsilon)}$ is $\frac{|F_{BSC(p)} - F_{WOM(\alpha,\epsilon)}|}{1-H(p)}$.
- For $i = 1, 2, \cdots, t$, let M_i be the number of bits written in the ith write, and let $N_{additional,i}$ be the number of additional cells we use to store the value in $F_{BSC(p)} - F_{WOM(\alpha,\epsilon)}$ in the ith write. Then the sum-rate is

$$R_{sum} = \frac{\sum_{i=1}^{t} M_i}{N + \sum_{i=1}^{t} N_{additional,i}}.$$

When is $F_{BSC(p)}$ a subset of $F_{WOM(\alpha,\epsilon)}$?

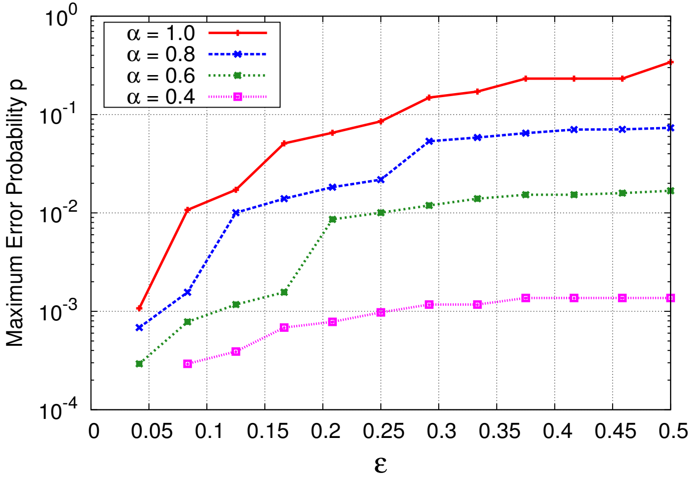


Fig. 8. The maximum value of p found for which $F_{\mathrm{BSC}(p)} \subseteq F_{\mathrm{WOM}(\alpha,\epsilon)}$.

Theoretical Analysis

It is interesting to know how much $F_{WOM(\alpha,\epsilon)}$ and $F_{BSC(p)}$ intersects.

Degrading WOM Channel to BSC

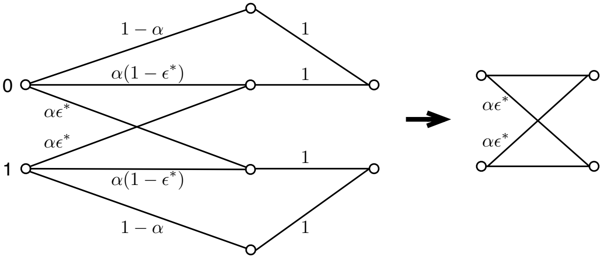


Fig. 3. Degrading the channel $WOM(\alpha, \epsilon^*)$ to $BSC(\alpha\epsilon^*)$. The two channels on the left and on the right are equivalent.

Degrading WOM Channel to Another WOM Channel

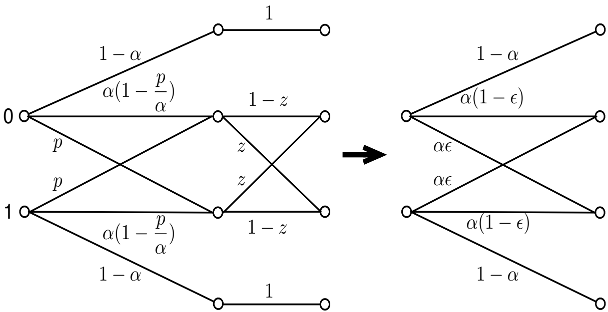


Fig. 4. Degrading channel $WOM(\alpha, \frac{p}{\alpha})$ to $WOM(\alpha, \epsilon)$. Here $z = \frac{\alpha\epsilon - p}{\alpha - 2p}$. The two channels on the left and on the right are equivalent.

Common Upgrading/Degrading of WOM-channel and BSC

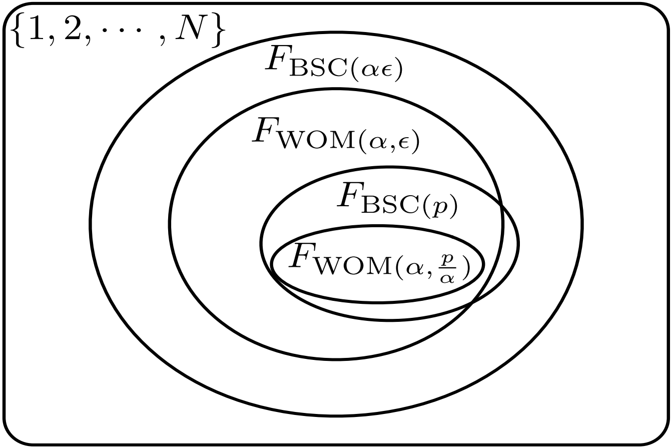
Lemma 2. *When* $p \leq \alpha\epsilon$,

$$F_{\mathrm{WOM}(\alpha,\frac{p}{\alpha})} \subseteq \left(F_{\mathrm{BSC}(p)} \cap F_{\mathrm{WOM}(\alpha,\epsilon)}\right),$$

and

$$\left(F_{\mathrm{WOM}(\alpha,\epsilon)} \cup F_{\mathrm{BSC}(p)}\right) \subseteq F_{\mathrm{BSC}(\alpha\epsilon)}.$$

Common Upgrading/Degrading of WOM-channel and BSC



Lower Bound to Achievable Sum-Rate

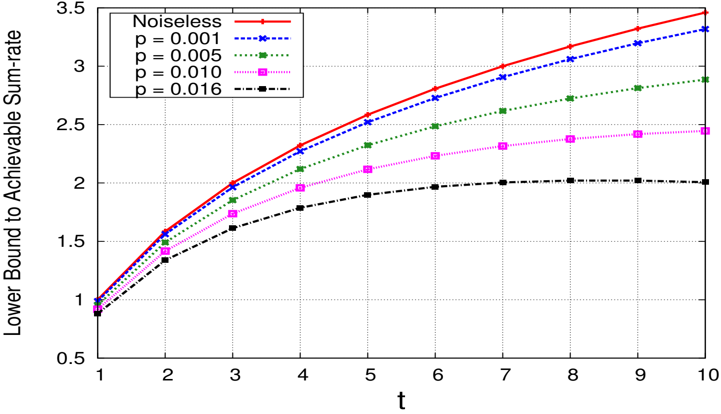


Fig. 6. Lower bound to achievable sum-rates for different error probability p.

$\boxed{\text{Rank Modulation}}$

Definition of Rank Modulation [1-2]

Rank Modulation:

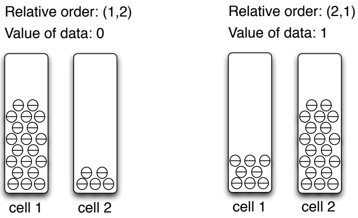
We use the relative order of cell levels (instead of their absolute values) to represent data.

[1] A. Jiang, R. Mateescu, M. Schwartz and J. Bruck, "Rank Modulation for Flash Memories," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1731–1735, July 2008.

[2] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1736–1740, July 2008.

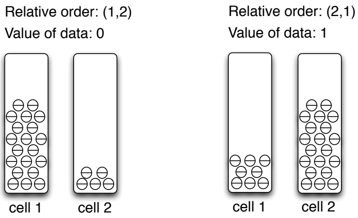
Examples and Extensions of Rank Modulation

- Example: Use 2 cells to store 1 bit.



Examples and Extensions of Rank Modulation

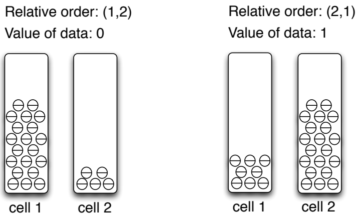
- Example: Use 2 cells to store 1 bit.



- Example: Use 3 cells to store $\log_2 6$ bits. The relative orders $(1,2,3),(1,3,2),\cdots,(3,2,1)$ are mapped to data $0,1,\cdots,5$.

Examples and Extensions of Rank Modulation

- Example: Use 2 cells to store 1 bit.



- Example: Use 3 cells to store $\log_2 6$ bits. The relative orders $(1,2,3),(1,3,2),\cdots,(3,2,1)$ are mapped to data $0,1,\cdots,5$.
- In general, k cells can represent $\log_2(k!)$ bits.

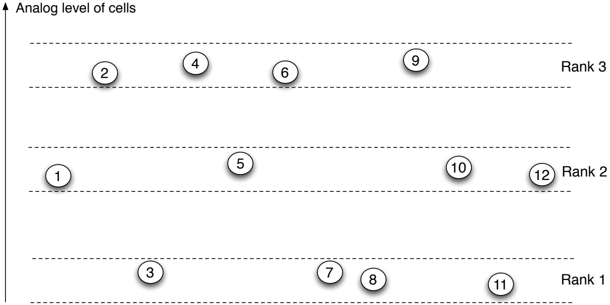
Rank Modulation using Multi-set Permutation

Extension: Let each rank have m cells.

Example

Let $m = 4$. The following is a multi-set permutation

$$(\{2,4,6,9\},\{1,5,10,12\},\{3,7,8,11\}).$$



Advantages of Rank Modulation

Easy Memory Scrubbing:

- Long-term data reliability.
- Easier cell programming.

Error-Correcting Codes for Rank Modulation

$\boxed{\text{Error Correcting Codes for Rank Modulation}}$

Error Models and Distance between Permutations

Based on the error model, there are various reasonable choices for the distance between permutations:

- Kendall-tau distance. (To be introduced in detail.)
- L_∞ distance.
- Gaussian noise based distance.
- Distance defined based on asymmetric errors or inter-cell interference.

We should choose the distance appropriately based on the type and magnitude of errors.

Kendall-tau Distance for Rank Modulation ECC [1]

When errors happen, the smallest change in a permutation is the local exchange of two adjacent numbers in the permutation. That is,

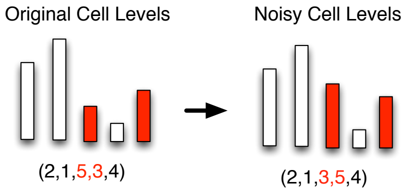
$$(a_1, \cdots, a_{i-1}, \underbrace{a_i, a_{i+1}}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, \underbrace{a_{i+1}, a_i}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n)$$

Kendall-tau Distance for Rank Modulation ECC [1]

When errors happen, the smallest change in a permutation is the local exchange of two adjacent numbers in the permutation. That is,

$$(a_1, \cdots, a_{i-1}, \underbrace{a_i, a_{i+1}}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, \underbrace{a_{i+1}, a_i}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n)$$

Example:

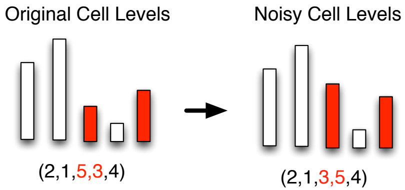


Kendall-tau Distance for Rank Modulation ECC [1]

When errors happen, the smallest change in a permutation is the local exchange of two adjacent numbers in the permutation. That is,

$$(a_1, \cdots, a_{i-1}, \underbrace{a_i, a_{i+1}}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, \underbrace{a_{i+1}, a_i}_{\text{adjacent pair}}, a_{i+2}, \cdots, a_n)$$

Example:



We can extend the concept to multiple such "local exchanges" (for larger errors).

[1] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE* International Symposium on Information Theory (ISIT), pp. 1736–1740, July 2008.

Kendall-tau Distance for Rank Modulation ECC

Definition (Adjacent Transposition)

An adjacent transposition is the local exchange of two neighboring numbers in a permutation, namely,

$$(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \cdots, a_n)$$

Kendall-tau Distance for Rank Modulation ECC

Definition (Adjacent Transposition)

An adjacent transposition is the local exchange of two neighboring numbers in a permutation, namely,

$$(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \cdots, a_n) \rightarrow (a_1, \cdots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \cdots, a_n)$$

Definition (Kendall-tau Distance)

Given two permutations A and B, the Kendall-tau distance between them, $d_\tau(A, B)$, is the minimum number of adjacent transpositions needed to change A into B. (Note that $d_\tau(A, B) = d_\tau(B, A)$.)

Kendall-tau Distance for Rank Modulation ECC

Definition (Adjacent Transposition)

An adjacent transposition is the local exchange of two neighboring numbers in a permutation, namely,

$$(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, a_{i+2}, \cdots, a_n) \to (a_1, \cdots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \cdots, a_n)$$

Definition (Kendall-tau Distance)

Given two permutations A and B, the Kendall-tau distance between them, $d_\tau(A, B)$, is the minimum number of adjacent transpositions needed to change A into B. (Note that $d_\tau(A, B) = d_\tau(B, A)$.)

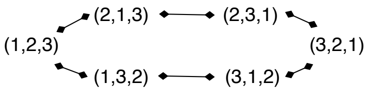
If the minimum Kendall-tau distance of a code is 2t+1, then it can correct t adjacent transposition errors.

Kendall-tau Distance for Rank Modulation ECC

Definition (State Diagram)

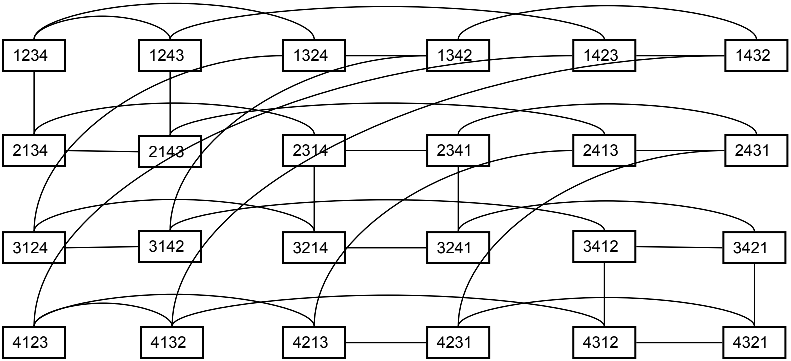
Vertices are permutations. There is an undirected edge between two permutations $A, B \in S_n$ iff $d_\tau(A, B) = 1$.

Example: The state diagram for $n = 3$ cells is



Kendall-tau Distance for Rank Modulation ECC

Example: The state diagram for $n = 4$ cells is



One-Error-Correcting Code

We introduce an error-correcting code of minimum Kendall-tau distance 3, which corrects one Kendall (i.e., adjacent transposition) error.

One-Error-Correcting Code

We introduce an error-correcting code of minimum Kendall-tau distance 3, which corrects one Kendall (i.e., adjacent transposition) error.

Definition (Inversion Vector)

Given a permutation $(a_1, a_2, \cdots, a_n)$, its inversion vector $(x_1, x_2, \cdots, x_{n-1}) \in \{0, 1\} \times \{0, 1, 2\} \times \cdots \times \{0, 1, \cdots, n-1\}$ is determined as follows:

- For $i = 1, 2, \cdots, n-1$, x_i is the number of elements in $\{1, 2, \cdots, i\}$ that are behind $i+1$ in the permutation $(a_1, \cdots, a_n)$.

One-Error-Correcting Code

We introduce an error-correcting code of minimum Kendall-tau distance 3, which corrects one Kendall (i.e., adjacent transposition) error.

Definition (Inversion Vector)

Given a permutation $(a_1, a_2, \cdots, a_n)$, its inversion vector $(x_1, x_2, \cdots, x_{n-1}) \in \{0, 1\} \times \{0, 1, 2\} \times \cdots \times \{0, 1, \cdots, n-1\}$ is determined as follows:

- For $i = 1, 2, \cdots, n-1$, x_i is the number of elements in $\{1, 2, \cdots, i\}$ that are behind $i+1$ in the permutation $(a_1, \cdots, a_n)$.

Example: The inversion vector for $(1, 2, 3, 4)$ is $(0, 0, 0)$. The inversion for $(4, 3, 2, 1)$ is $(1, 2, 3)$. The inversion vector for $(2, 4, 3, 1)$ is $(1, 1, 2)$.

One-Error-Correcting Code [1]

By viewing the inversion vector as coordinates, we embed permutations in an $(n-1)$-dimensional space.

One-Error-Correcting Code [1]

By viewing the inversion vector as coordinates, we embed permutations in an $(n-1)$-dimensional space.

Fact: For any two permutations $A, B \in S_n$, $d_\tau(A, B)$ is no less than their L_1 distance in the $(n-1)$-dimensional space.

One-Error-Correcting Code [1]

By viewing the inversion vector as coordinates, we embed permutations in an $(n-1)$-dimensional space.

Fact: For any two permutations $A, B \in S_n$, $d_\tau(A, B)$ is no less than their L_1 distance in the $(n-1)$-dimensional space.

Idea: We can construct a code of minimum L_1 distance D in the $(n-1)$-dimensional array of size $2 \times 3 \times \cdots \times n$. Then it is a code of Kendall-tau distance at least D for the permutations.

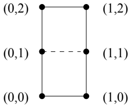
[1] A. Jiang, M. Schwartz and J. Bruck, "Error-Correcting Codes for Rank Modulation," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, pp. 1736–1740, July 2008.

One-Error-Correcting Code

Example: When $n = 3$ or $n = 4$, the embedding is as follows. (Only the solid edges are the edges in the state graph of permutations.)

Permutation		Coordinates
1 2 3	→	(0,0)
1 3 2	→	(0,1)
2 1 3	→	(1,0)
2 3 1	→	(1,1)
3 1 2	→	(0,2)
3 2 1	→	(1,2)

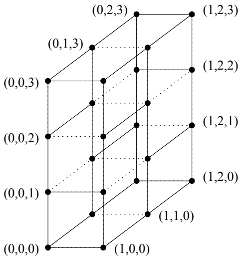
(a)



(b)

Permutation		Coordinates	Permutation		Coordinates
1 2 3 4	→	(0,0,0)	3 1 2 4	→	(0,2,0)
1 2 4 3	→	(0,0,1)	3 1 4 2	→	(0,2,1)
1 3 2 4	→	(0,1,0)	3 2 1 4	→	(1,2,0)
1 3 4 2	→	(0,1,1)	3 2 4 1	→	(1,2,1)
1 4 2 3	→	(0,0,2)	3 4 1 2	→	(0,2,2)
1 4 3 2	→	(0,1,2)	3 4 2 1	→	(1,2,2)
2 1 3 4	→	(1,0,0)	4 1 2 3	→	(0,0,3)
2 1 4 3	→	(1,0,1)	4 1 3 2	→	(0,1,3)
2 3 1 4	→	(1,1,0)	4 2 1 3	→	(1,0,3)
2 3 4 1	→	(1,1,1)	4 2 3 1	→	(1,1,3)
2 4 1 3	→	(1,0,2)	4 3 1 2	→	(0,2,3)
2 4 3 1	→	(1,1,2)	4 3 2 1	→	(1,2,3)

(c)



(d)

One-Error-Correcting Code

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.

One-Error-Correcting Code

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 Errors [1]

- The above code can be generalized to correct more errors.

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

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

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

capacity of rank modulation ECC of Kendall-tau distance d.

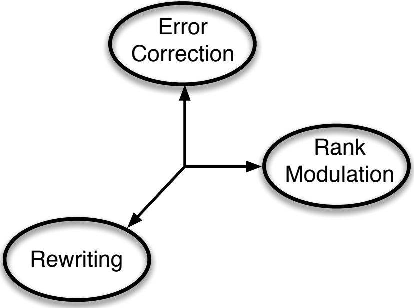
$$C(d) = \begin{cases} 1 & \text{if } d = O(n) \\ 1-\epsilon & \text{if } d = \Theta(n^{1+\epsilon}),\ 0 < \epsilon < 1 \\ 0 & \text{if } d = \Theta(n^2) \end{cases}$$

[1] A. Barg and A. Mazumdar, "Codes in Permutations and Error Correction for Rank Modulation," *ISIT'10*.

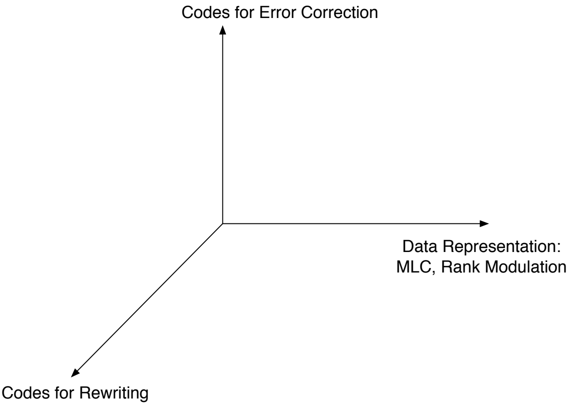
More Aspects of Rank Modulation

Rank Modulation wit Multi-set Permutation: A bridge to existing ECC.

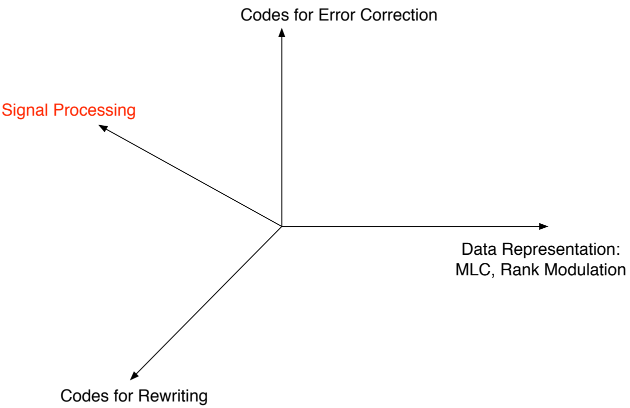
Efficient rewriting.


Error Correction
Rank Modulation
Rewriting

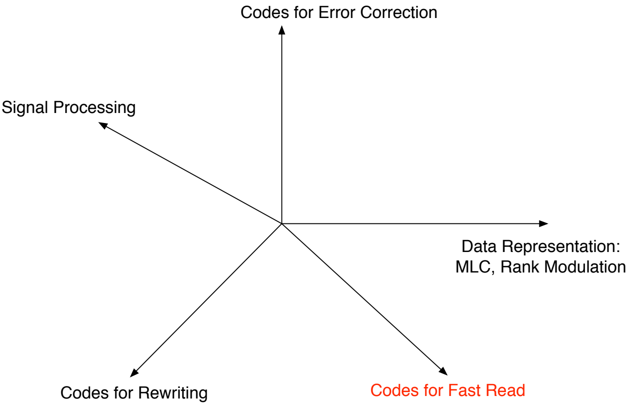
Open Problems on Coding for Flash Memory



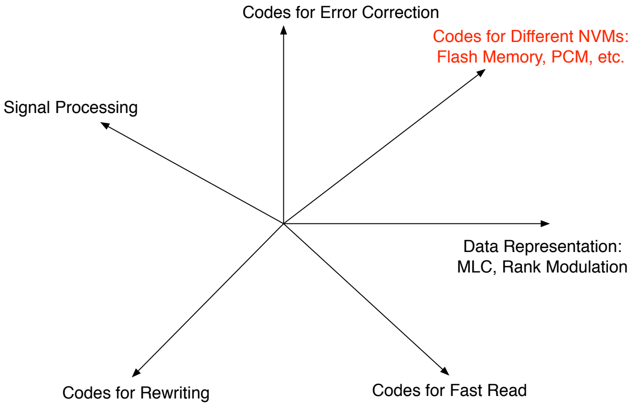
Open Problems on Coding for Flash Memory



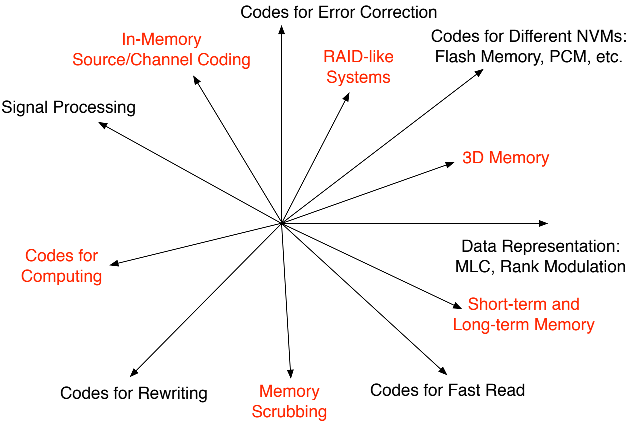
Open Problems on Coding for Flash Memory



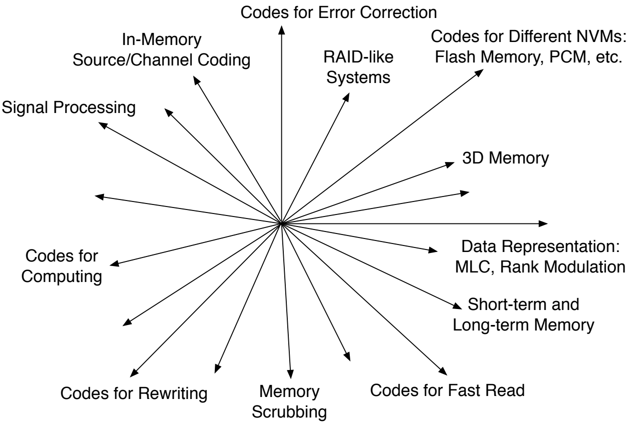
Open Problems on Coding for Flash Memory



Open Problems on Coding for Flash Memory



Open Problems on Coding for Flash Memory



Open Problems on Coding for Flash Memory