Channel Coding Methods for Emerging Data Storage and Memory Systems

Opportunities to Innovate Beyond the Hamming Metric

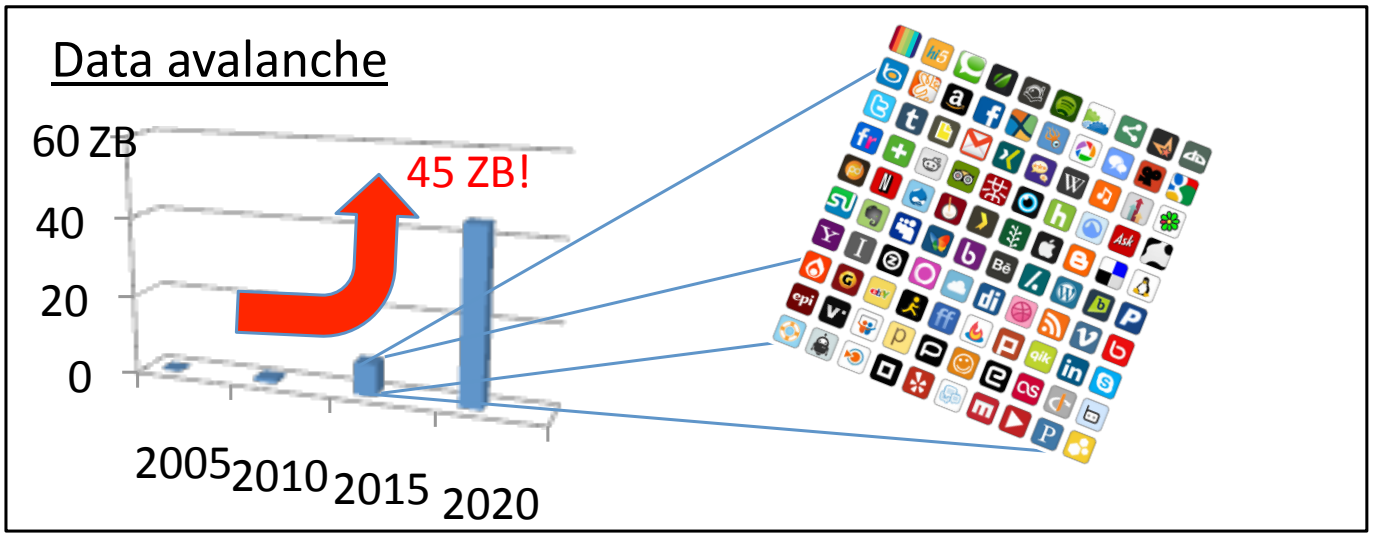
L. Dolecek[1] A. Jiang[2]

[1]Department of Electrical Engineering, UCLA

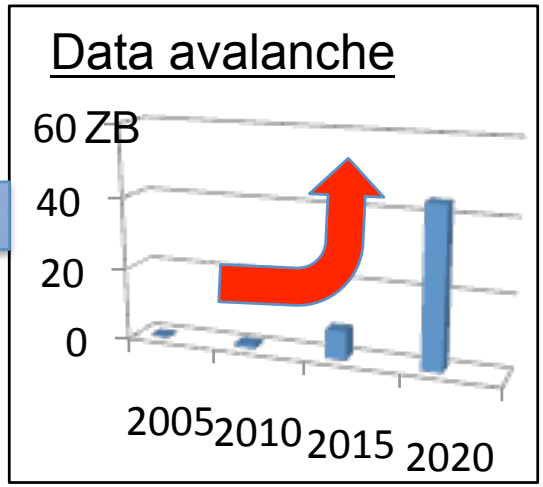
[2]Department of Computer Science, Texas A&M

ISIT, 2014

Welcome to the zettabyte age



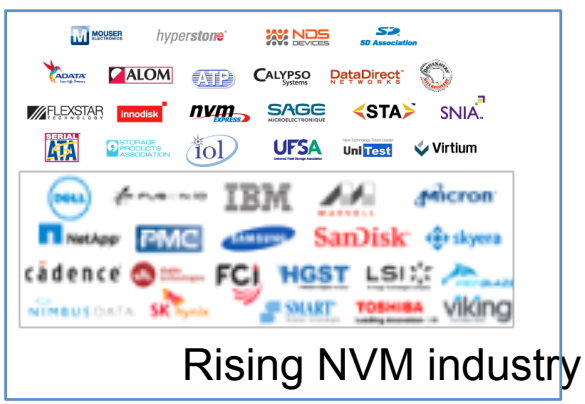
Exciting times for data storage



Innovations in data storage are the key enabler of the information revolution

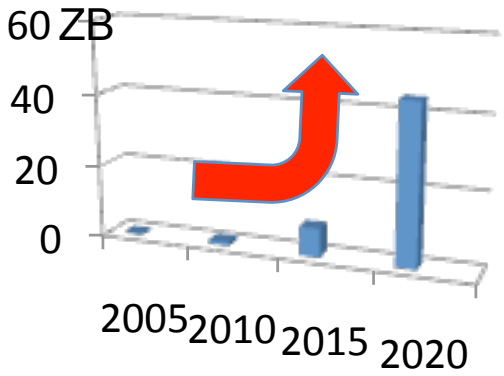
Data storage

Data center farms



Exciting times for data storage

Data avalanche

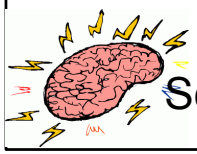


New technologies

Personalized healthcare

Augmented reality

Security



Smart cities

Scientific discoveries

Innovations in data storage are the key enabler of the information revolution

Data storage

Data center farms

Rising NVM industry

What is this tutorial about ?

Today we will learn about

- Physical characteristics of emerging non-volatile memories,

What is this tutorial about ?

Today we will learn about

- Physical characteristics of emerging non-volatile memories,
- Channel models associated with these technologies,

What is this tutorial about ?

Today we will learn about

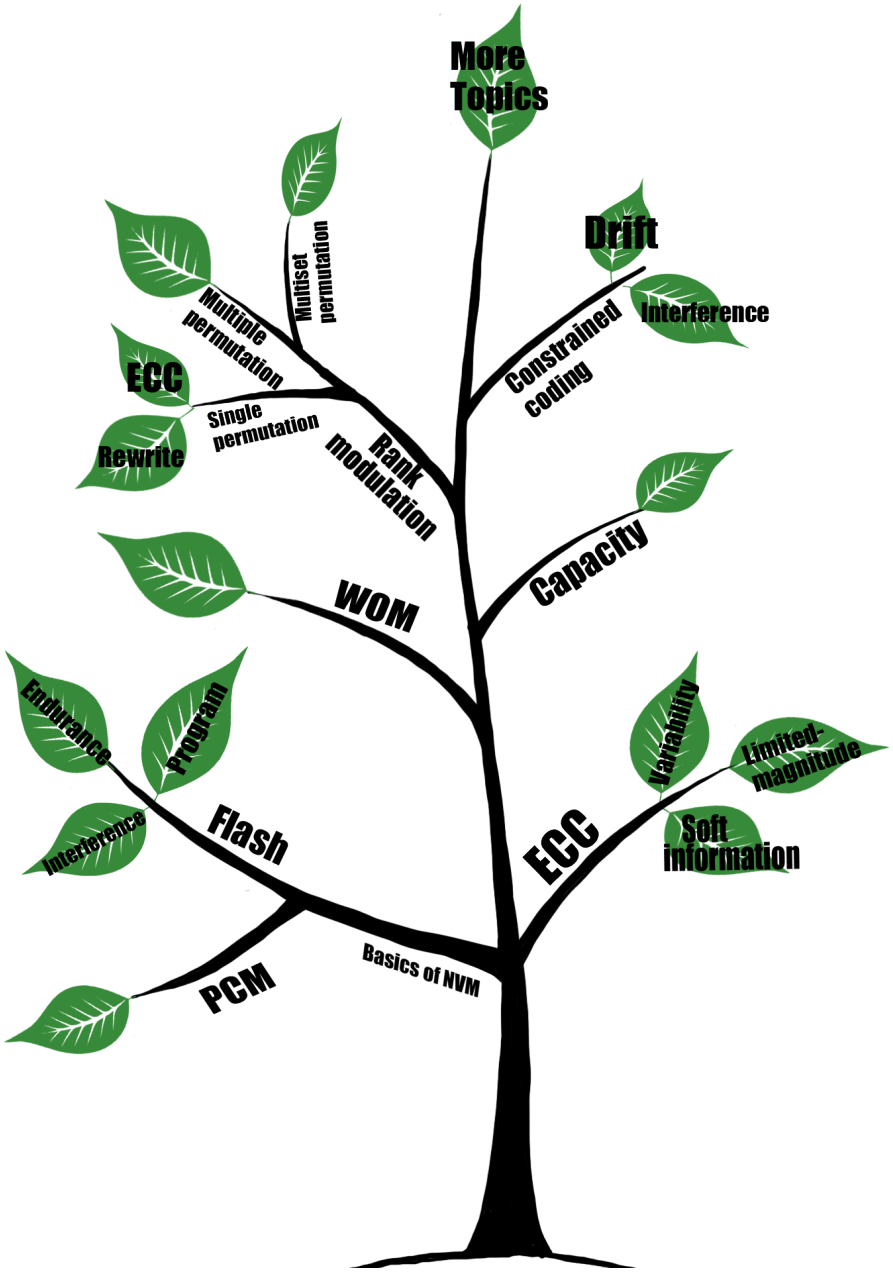
- Physical characteristics of emerging non-volatile memories,
- Channel models associated with these technologies,
- Various recent developments in coding for memories, and

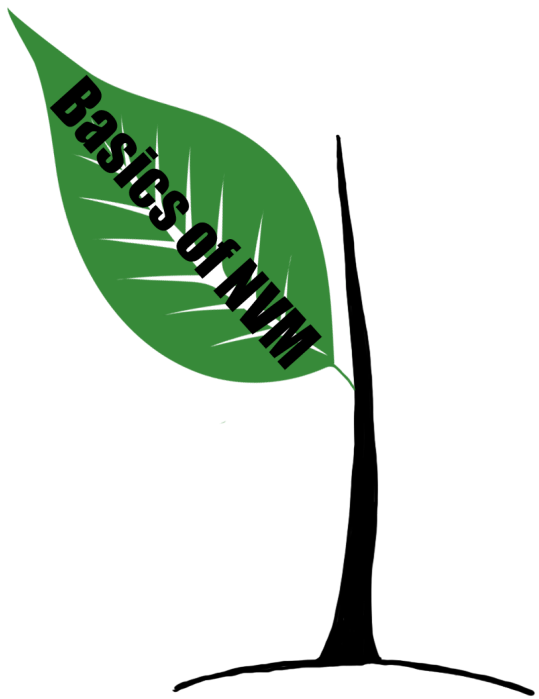
What is this tutorial about ?

Today we will learn about

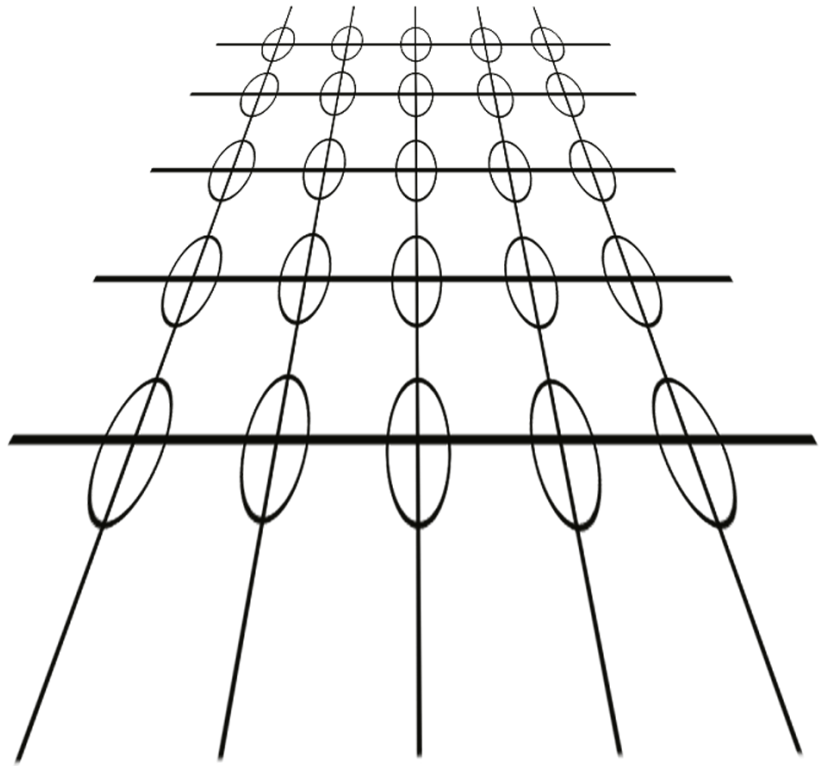
- Physical characteristics of emerging non-volatile memories,
- Channel models associated with these technologies,
- Various recent developments in coding for memories, and
- Open problems and future directions in communication techniques for memories.

Outline


More Topics
Drift
Interference
Constrained coding
Capacity
Rank modulation
Multiset permutation
Multiple permutation
ECC
Single permutation
Rewrite
WOM
Flash
Endurance
Program
Interference
PCM
Basics of NVM
ECC
Variability
Limited-magnitude
Soft information

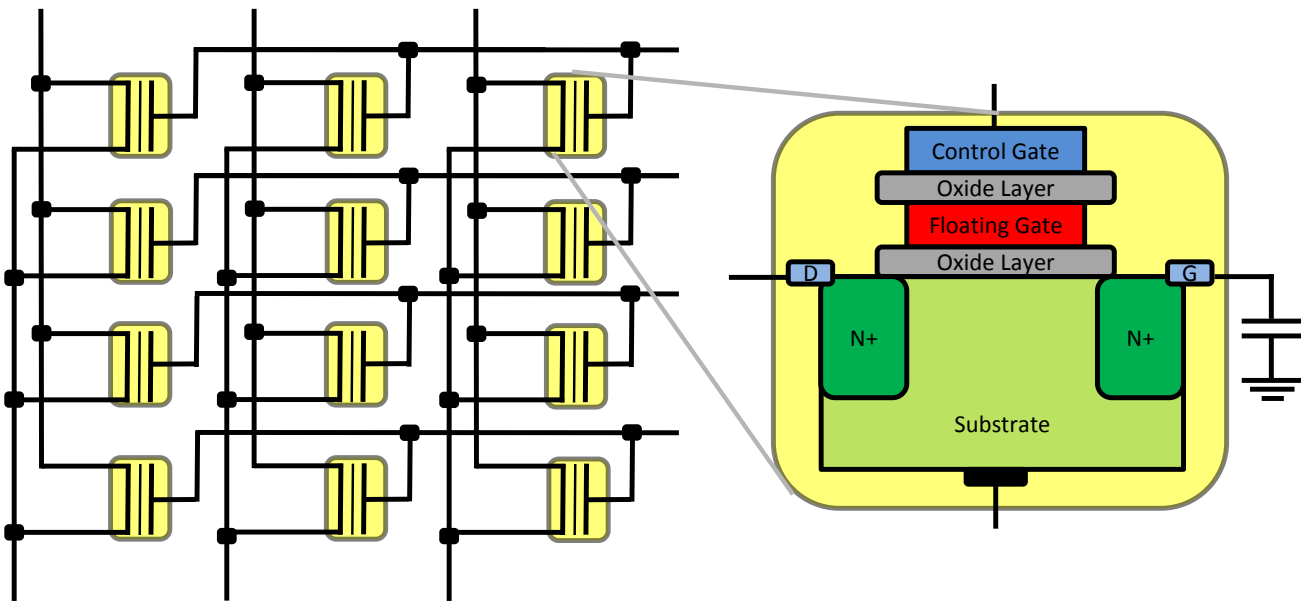

Basics of NVM

Basics of NVM operations



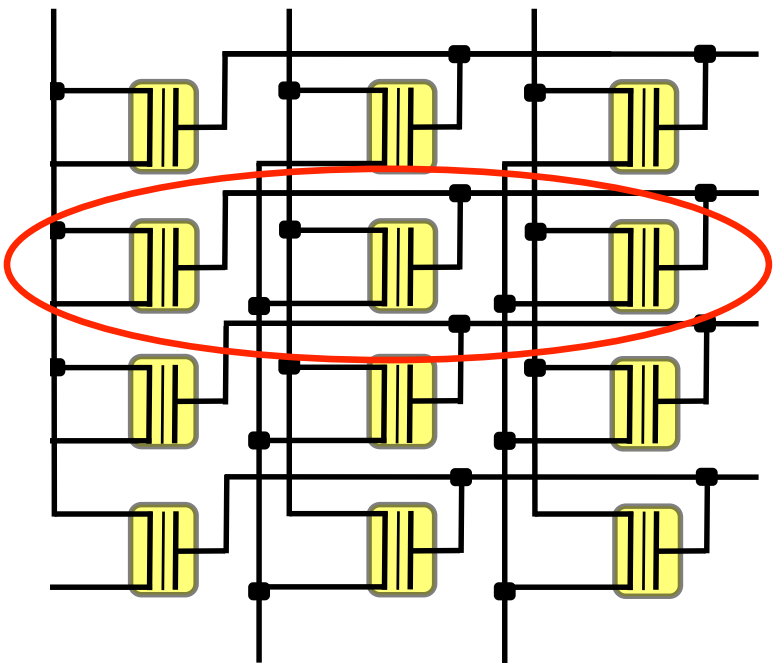
Flash technology

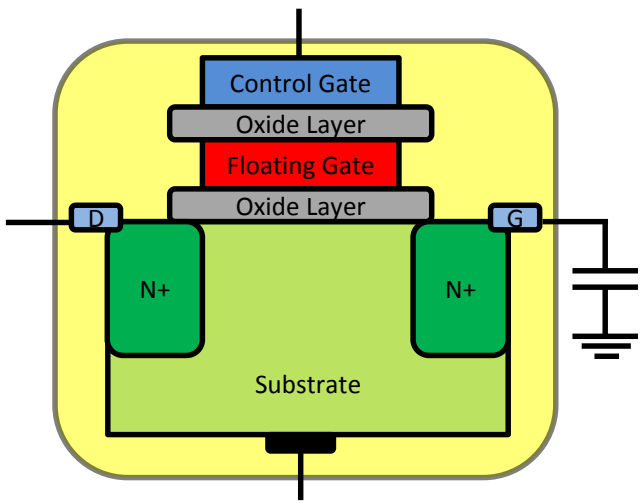
- A cell is a floating-gate transistor.



Flash technology

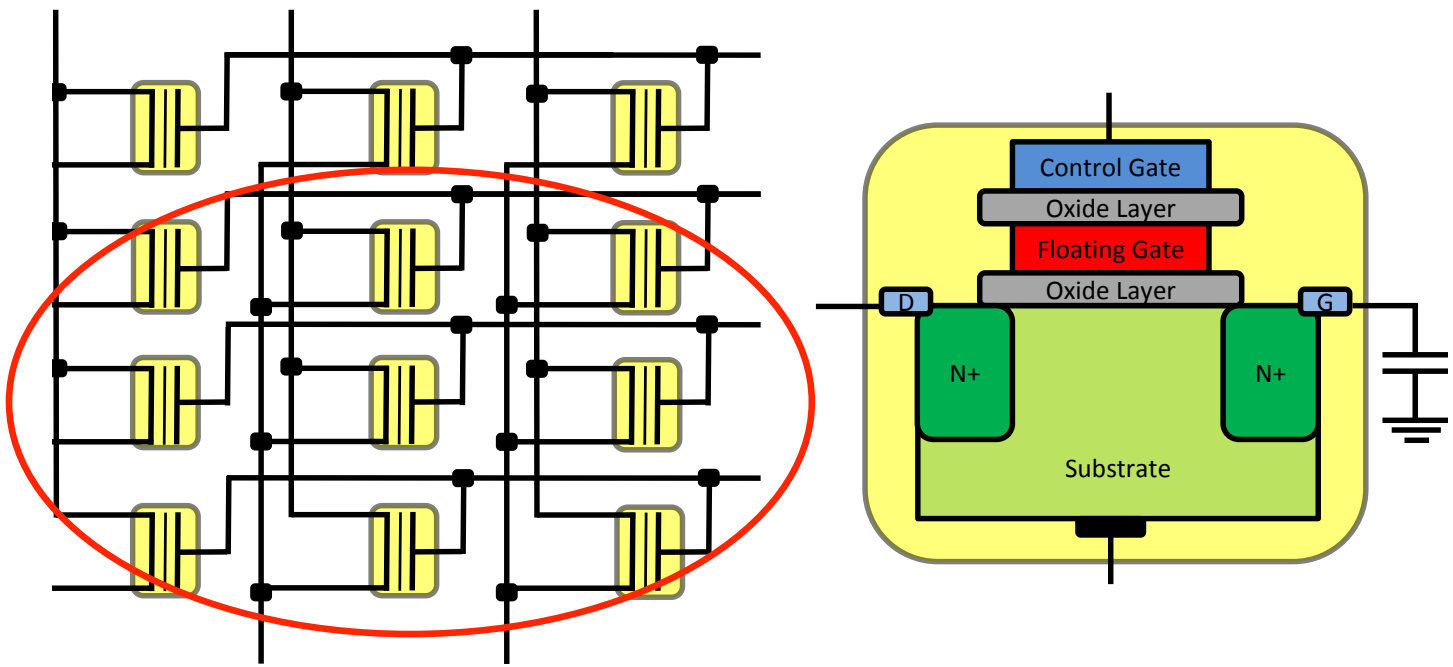
- A cell is a floating-gate transistor.
- Row of cells is a page.





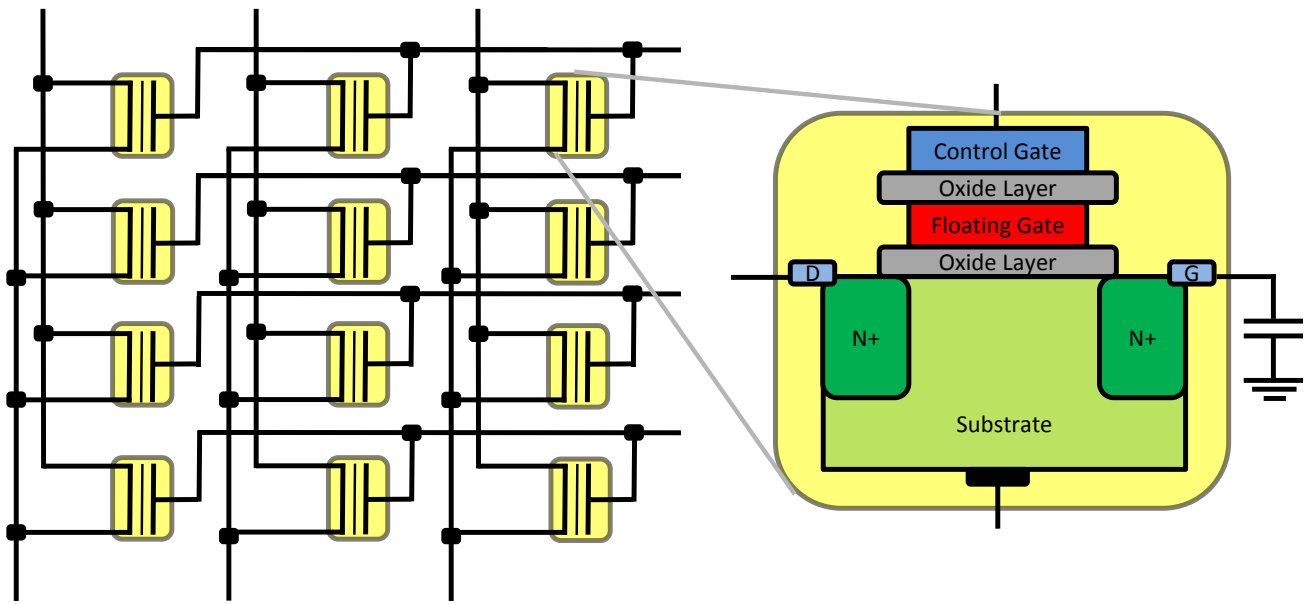
Flash technology

- A cell is a floating-gate transistor.
- Row of cells is a page.
- Group of pages is a block.



Flash technology

- A cell is a floating-gate transistor.
- Row of cells is a page.
- Group of pages is a block.
- A flash memory block is an array of $2^{20} \sim$ million cells.

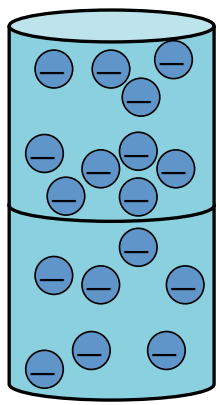


Flash technology

- Cell value corresponds to the voltage induced by the number of electrons stored on the gate.

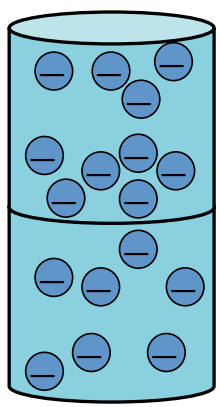
Flash technology

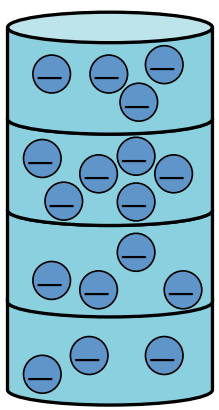
- Cell value corresponds to the voltage induced by the number of electrons stored on the gate.
- Single level cell (SLC)
  - stores 1 bit per cell; one separating level



Flash technology

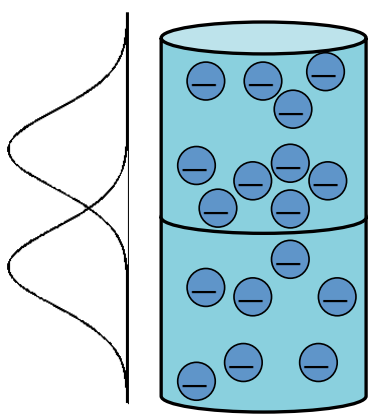
- Cell value corresponds to the voltage induced by the number of electrons stored on the gate.
- Single level cell (SLC)
  - stores 1 bit per cell; one separating level
- Multiple level cell (MLC)
  - stores 2 or more bits per cell; multiple separating levels

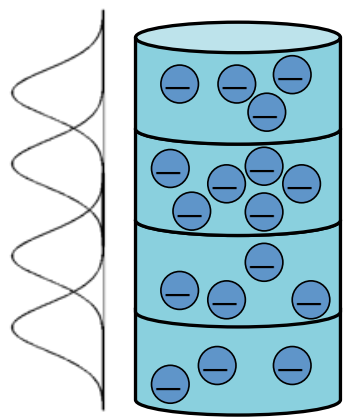




Flash technology

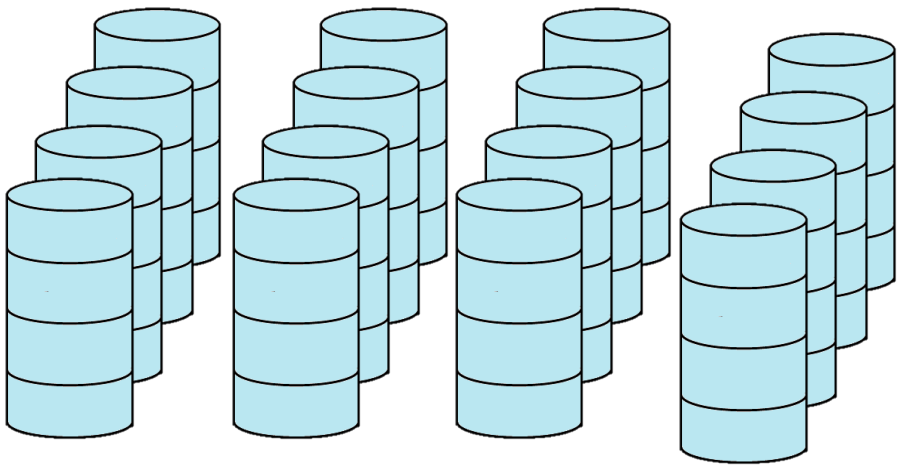
- Cell value corresponds to the voltage induced by the number of electrons stored on the gate.
- Single level cell (SLC)
  - stores 1 bit per cell; one separating level
- Multiple level cell (MLC)
  - stores 2 or more bits per cell; multiple separating levels





Flash programming – example 1

Flash block

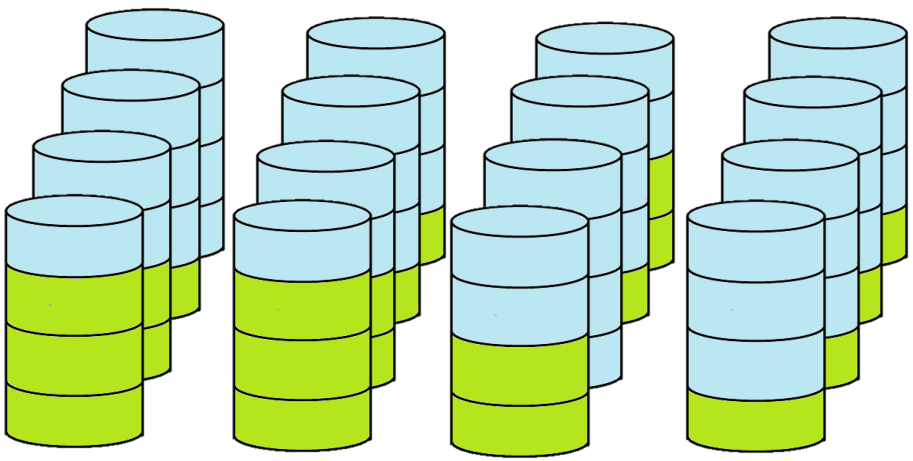


Initial state

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Flash programming – example 1

Flash block

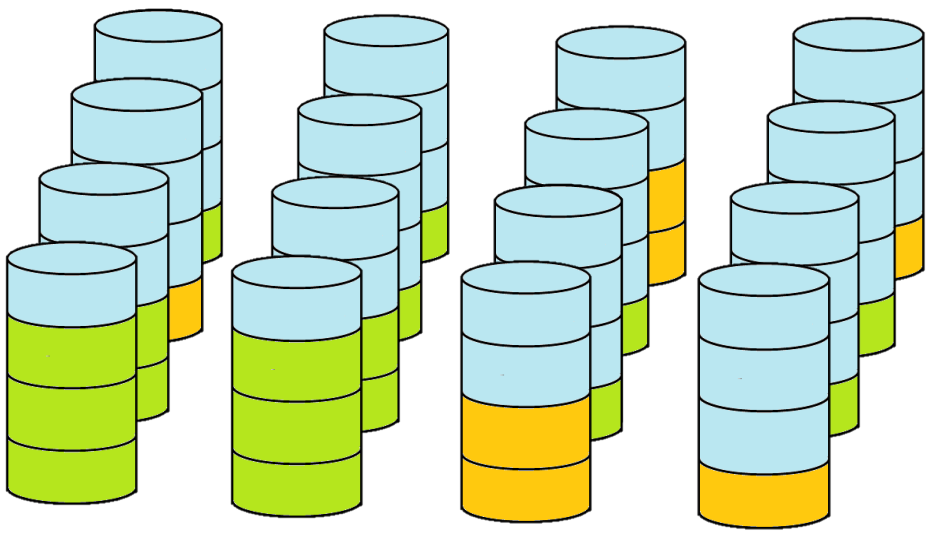


Final state

1	1	2	1
1	1	1	1
2	2	1	1
3	3	2	1

Flash programming – example 2

Flash block

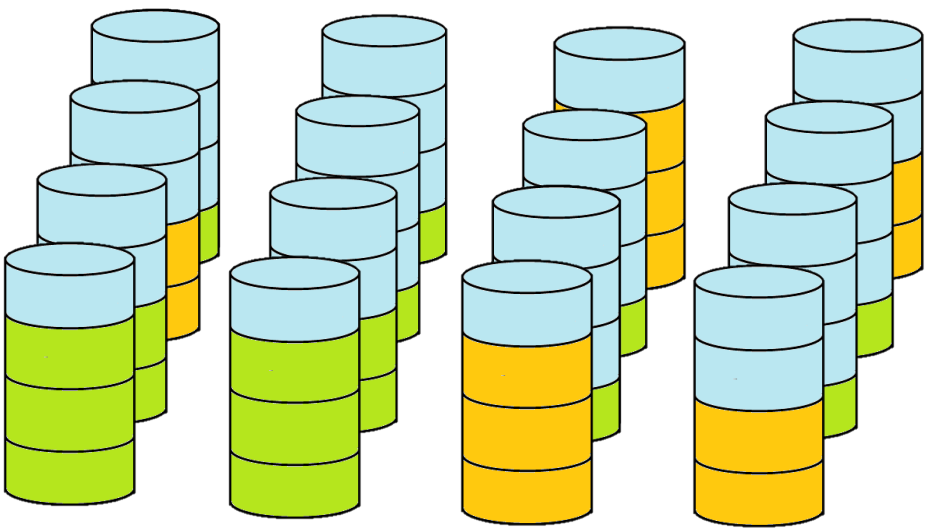


Initial state

1	1	2	1
1	1	1	1
2	2	1	1
3	3	2	1

Flash programming – example 2

Flash block

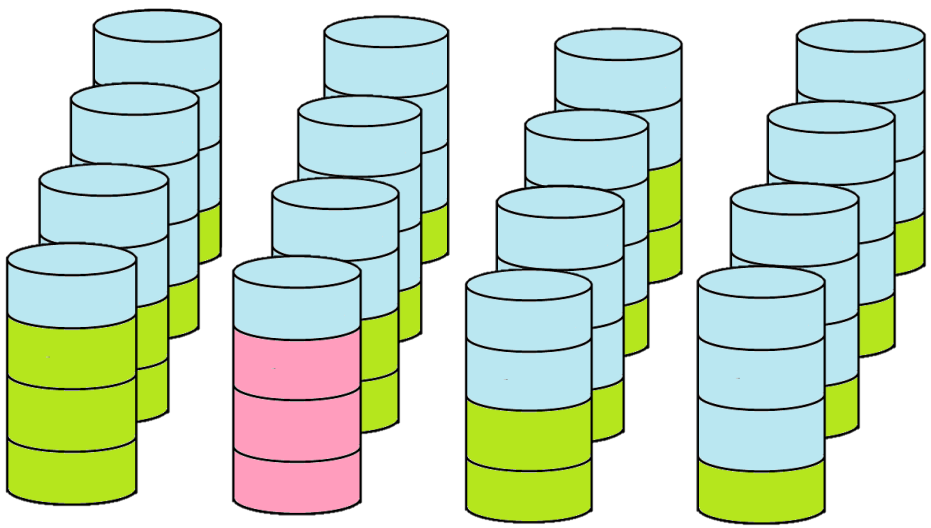


Final state

1	1	3	2
2	1	1	1
2	2	1	1
3	3	3	2

Flash programming – example 3

Flash block

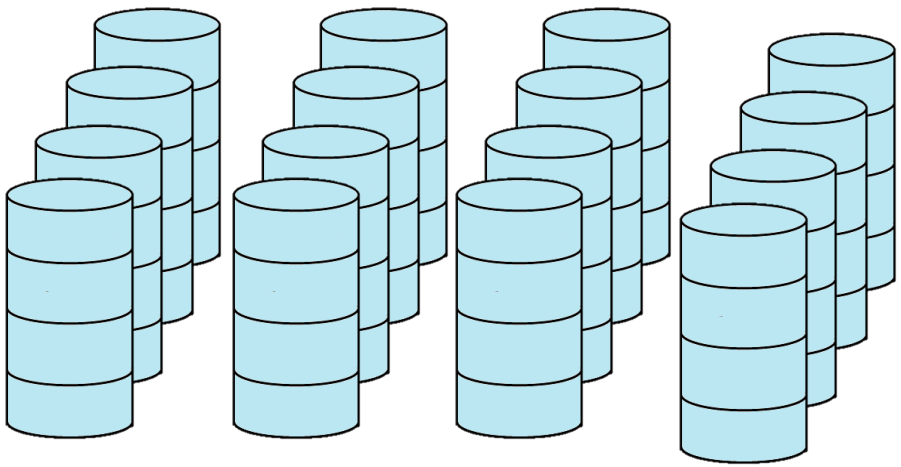


Initial state

1	1	2	1
1	1	1	1
2	2	1	1
3	3	2	1

Flash programming – example 3

Flash block

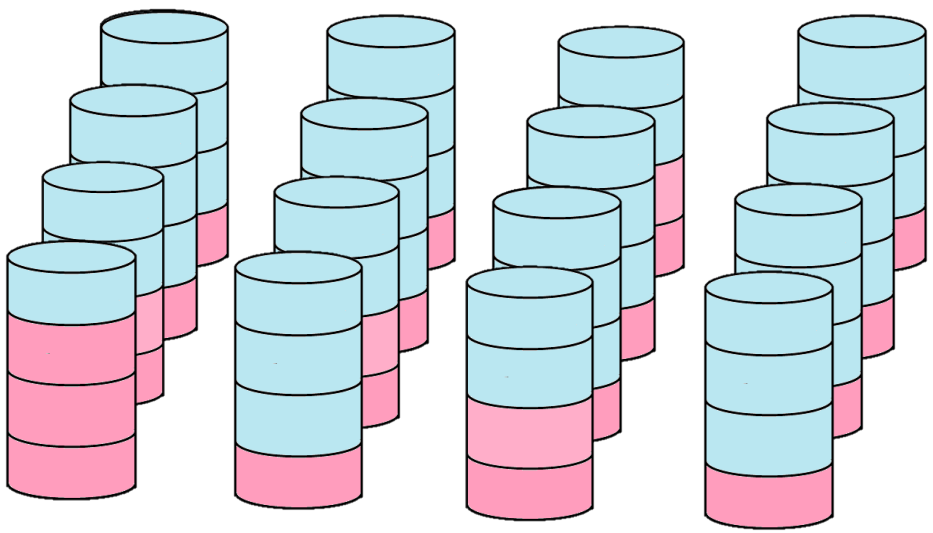


Intermediate state

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Flash programming – example 3

Flash block

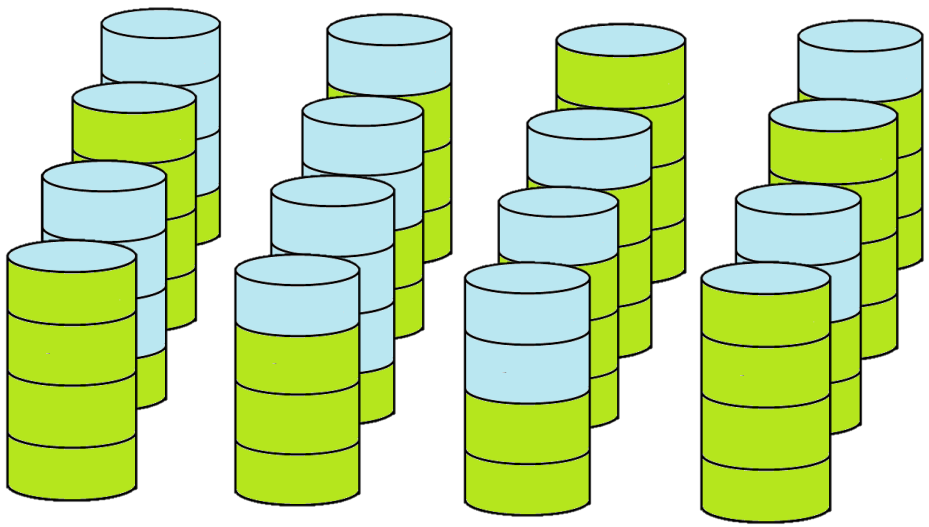


Final state

1	1	2	1
1	1	1	1
2	2	1	1
3	1	2	1

Flash endurance

Flash block

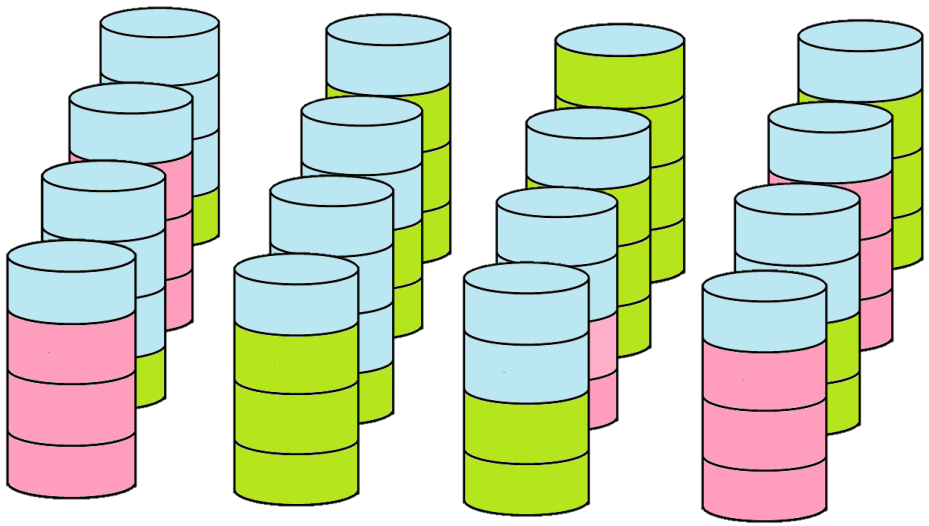


Programmed state

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

Flash endurance

Flash block



Retrieved state

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

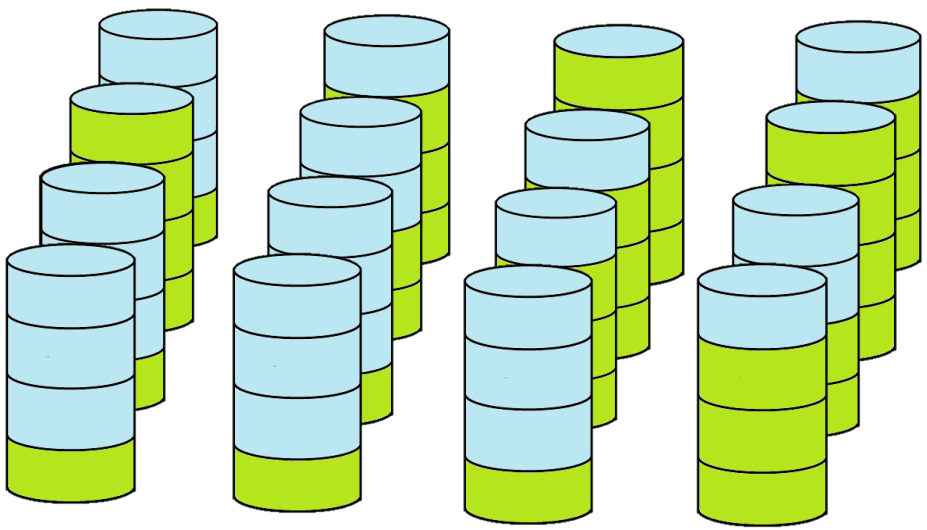
- Charge leakage over time results in level drop!

Flash endurance

- Frequent block erases cause so-called "wearout" due to charge loss on the gates.
- Flash memory lifetime is commonly expressed in terms of the number of program and erase (P/E) cycles allowed before memory is deemed unusable.
- Lifetime:
  - SLC: $\approx 10^6$ P/E cycles
  - MLC: $\approx 10^3 - 10^5$ P/E cycles
- With frequent writes 2GB TLC lasts less than 3 months!

Intercell coupling

Flash block

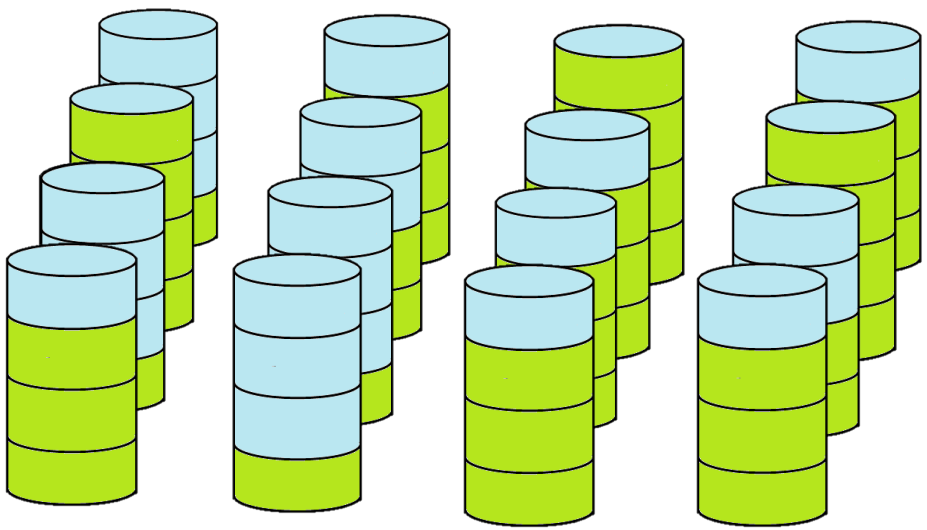


Original state

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

Intercell coupling

Flash block

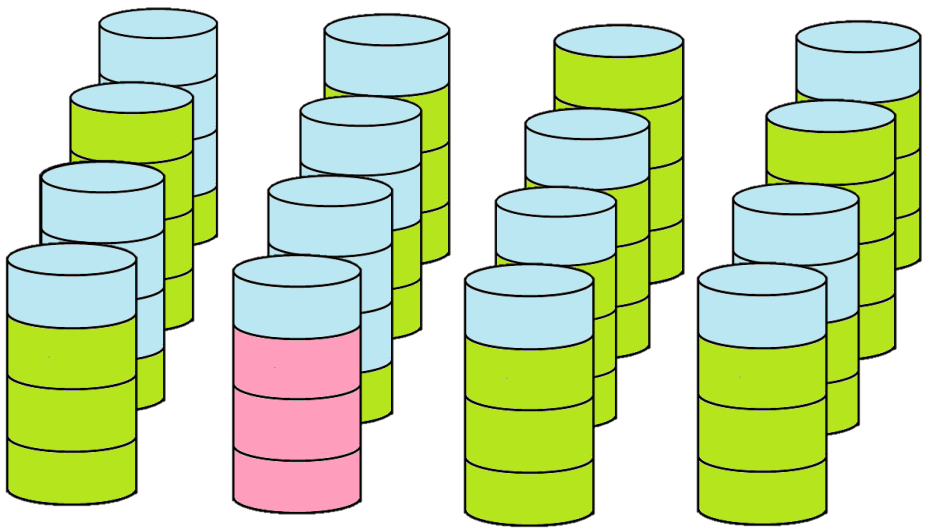


Programmed state

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

Intercell coupling

Flash block



Retrieved state

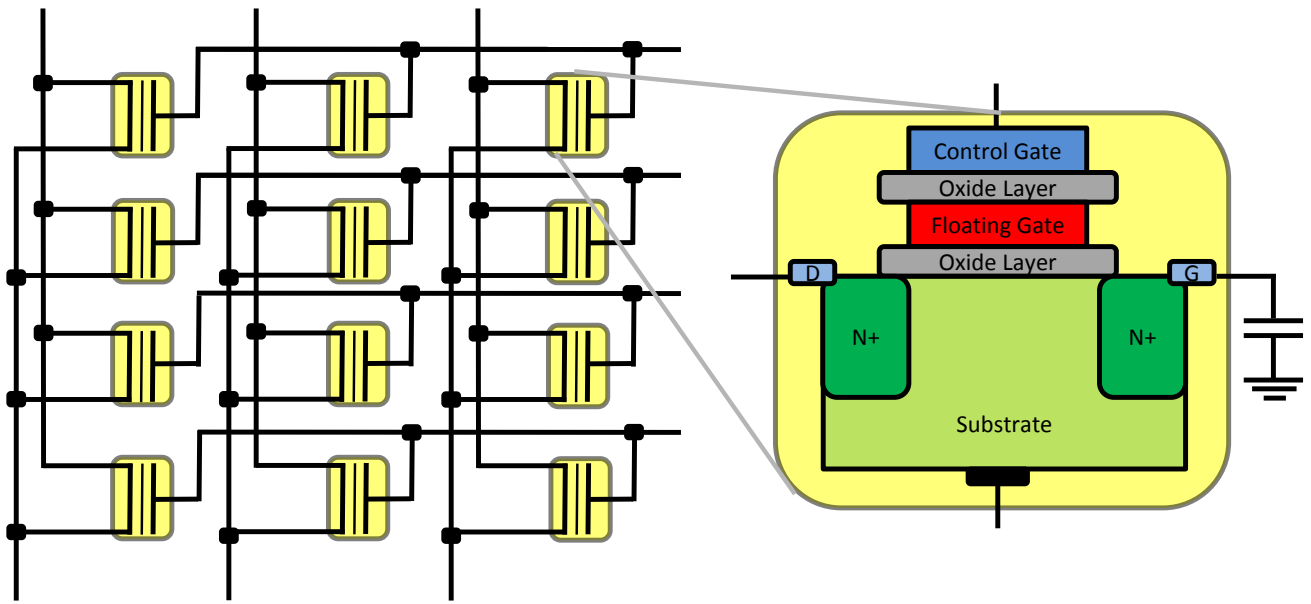
1	3	4	3
4	2	3	4
1	1	3	2
3	3	3	3

- Change in charge in one cell affects voltage threshold of a neighboring cell.
- During write/read, stressed (victim) cell appears weakly programmed.

To summarize...

Key flash features

- Write on the page level, erase on the block level
- Wearout limits usable lifetime
- Inter-symbol interference




Endurance
Program
Interference
Flash
Basics of NVM
PCM

Phase change memories (PCM)

Each cell is in two fundamental states (with in-between states)

- Amorphous – high resistance
  ↓ (intermediate)
  ↓ (intermediate)
- Polycrystalline – low resistance

Phase change memories (PCM)

Pros

- Erases on the cell level
- Faster access
- More P/E cycles, longer lifetime

Each cell is in two fundamental states (with in-between states)

- Amorphous – high resistance
  ↓ (intermediate)
  ↓ (intermediate)
- Polycrystalline – low resistance

Phase change memories (PCM)

Pros

- Erases on the cell level
- Faster access
- More P/E cycles, longer lifetime

Cons

- Not as dense
- Thermal accumulation
- Thermal cross talk
- Higher processing cost

Each cell is in two fundamental states (with in-between states)

- Amorphous – high resistance
  ↓ (intermediate)
  ↓ (intermediate)
- Polycrystalline – low resistance

References

- J. Cooke, "The inconvenient truths about NAND flash memories", FS, 2007.
- L. M. Grupp *et al.*, "Beyond the datasheet: using test beds to probe non-volatile memories' dark secrets", GC, 2010.
- L. M. Grupp *et al.*, "Characterizing flash memory: anomalies, observations, and applications", Micro, 2009.
- J. Burr *et al.*, "Phase change memory technology", JVST, 2010.
- X. Huang *et al.*, "Optimization of achievable information rates and number of levels in multilevel flash memories", ICN, 2013.


Endurance
Program
Interference
Flash
PCM
Basics of NVM
ECC

Part II: Error Correction Coding

ECC is a must for new NVMs

- Micronix, “NAND Flash Technical Note”, 2014.
- Spansion, “What Types of ECC Should Be Used on Flash Memory?”, 2011.
- Micron, “The Inconvenient Truth About NAND Flash”, 2007.

ECC is a must for new NVMs

- Micronix, "NAND Flash Technical Note", 2014.
- Spansion, "What Types of ECC Should Be Used on Flash Memory?", 2011.
- Micron, "The Inconvenient Truth About NAND Flash", 2007.

ECC is a must for new NVMs

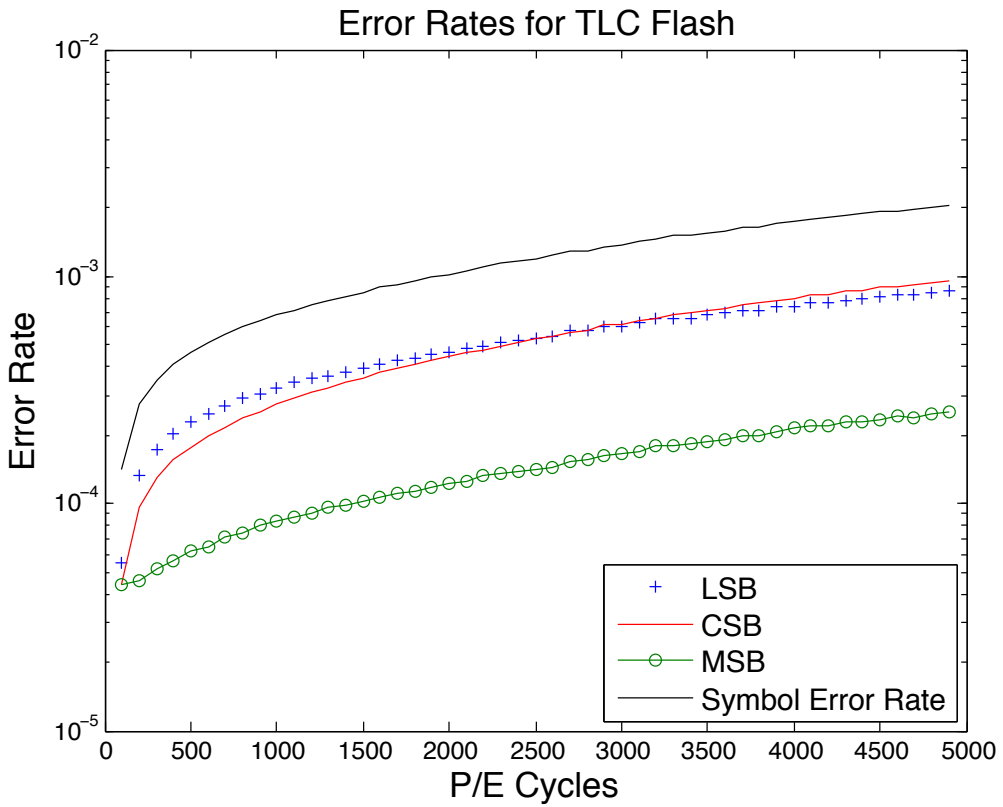
- Early designs:
  Hamming code, Reed-Solomon code
- Widespread industry standard:
  BCH code

Micronix, "NAND Flash Technical Note", 2014.

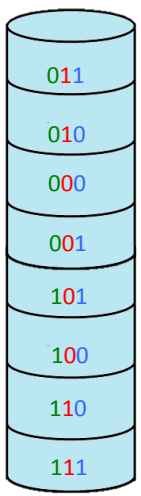
Spansion, "What Types of ECC Should Be Used on Flash Memory?", 2011.

Micron, "The Inconvenient Truth About NAND Flash", 2007.

Flash chip data collection: a closer look



LSB: least significant bit
CSB: center significant bit
MSB: most significant bit



- Data collected in Swanson Lab, UCSD.

Error patterns in TLC and the need for new codes

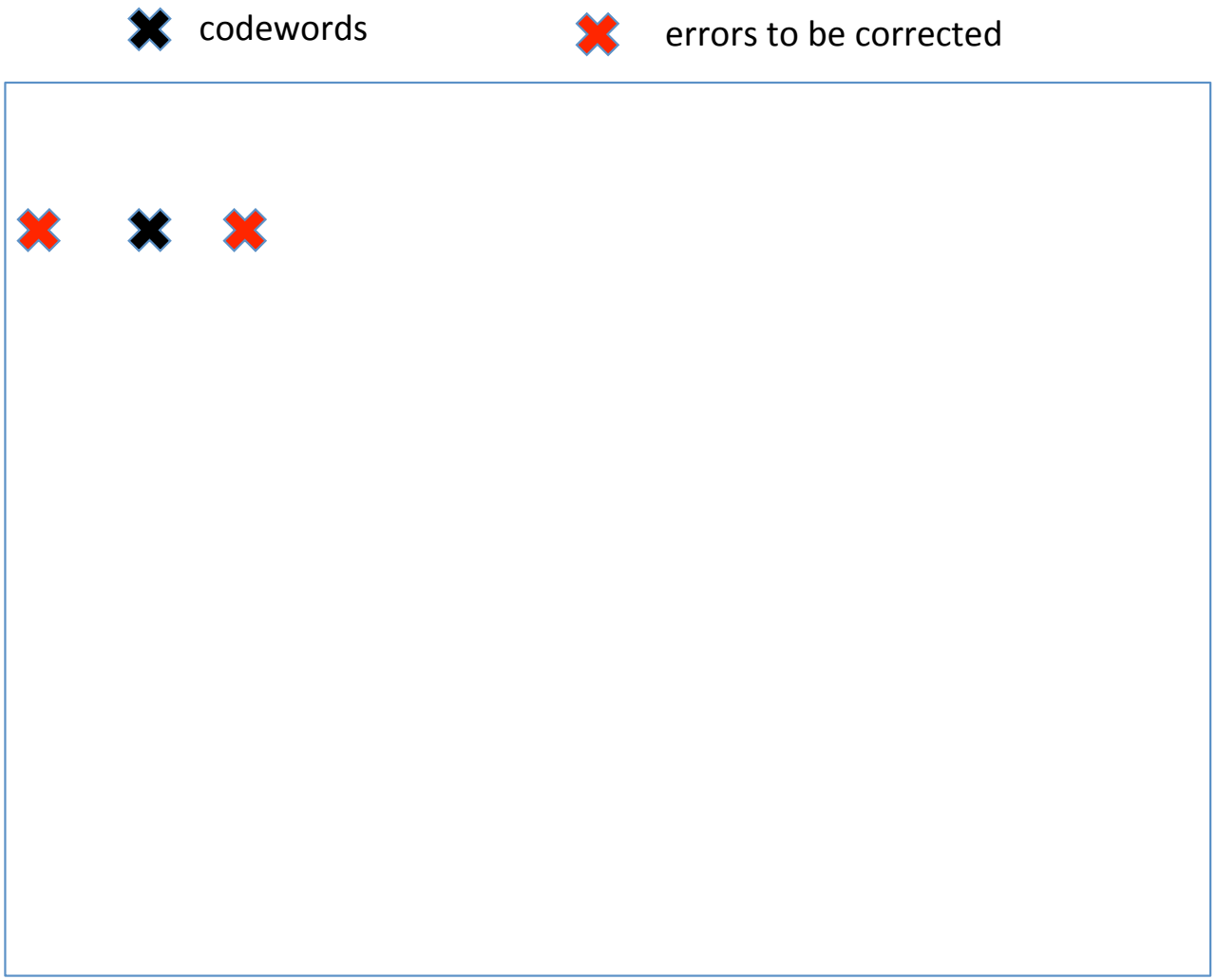
Number of wrong bits per wrong symbol	Percentage of errors
1	0.9617
2	0.0314
3	0.0069

Error patterns in TLC and the need for new codes

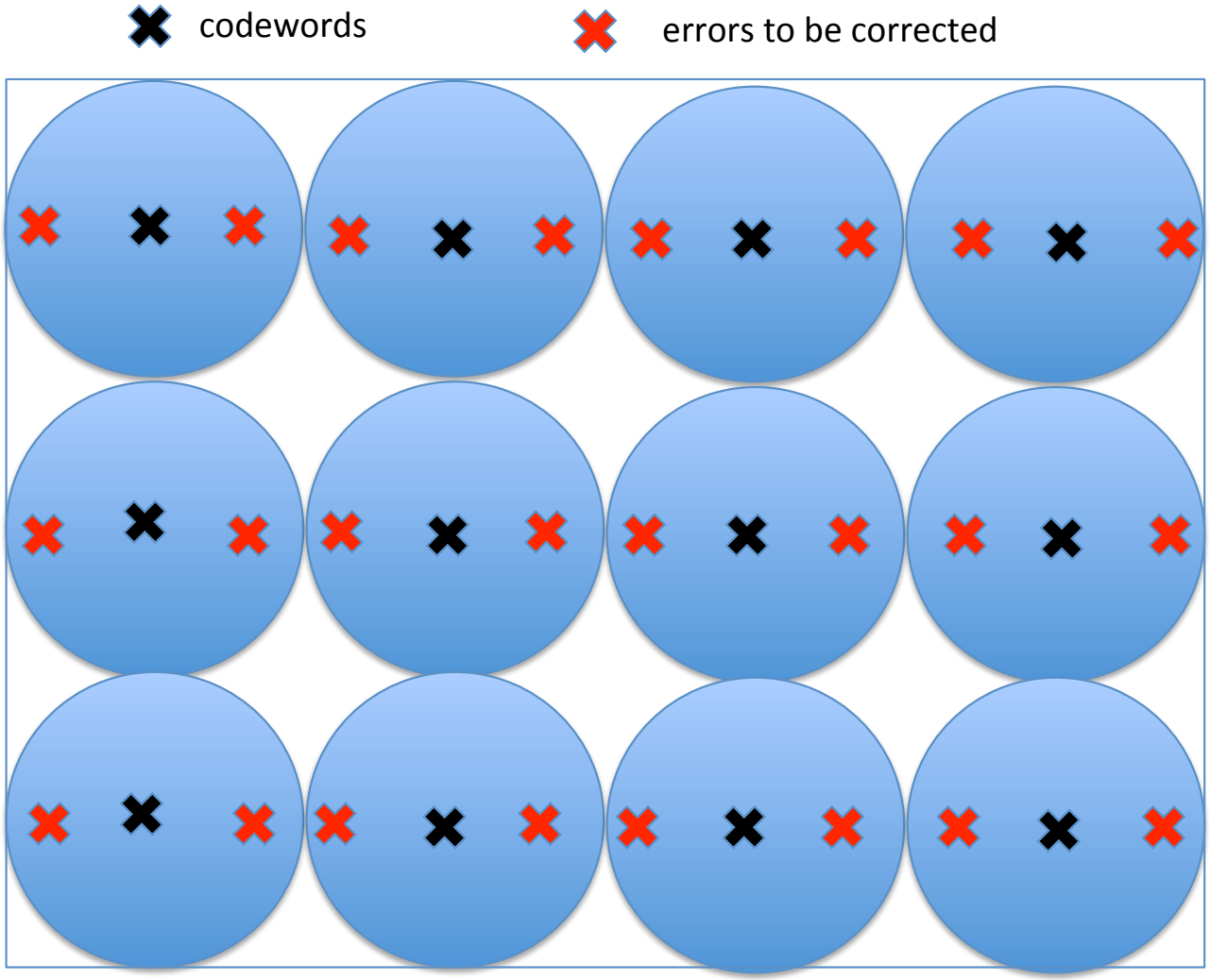
Number of wrong bits per wrong symbol	Percentage of errors
1	0.9617
2	0.0314
3	0.0069

Why not use standard codes ?

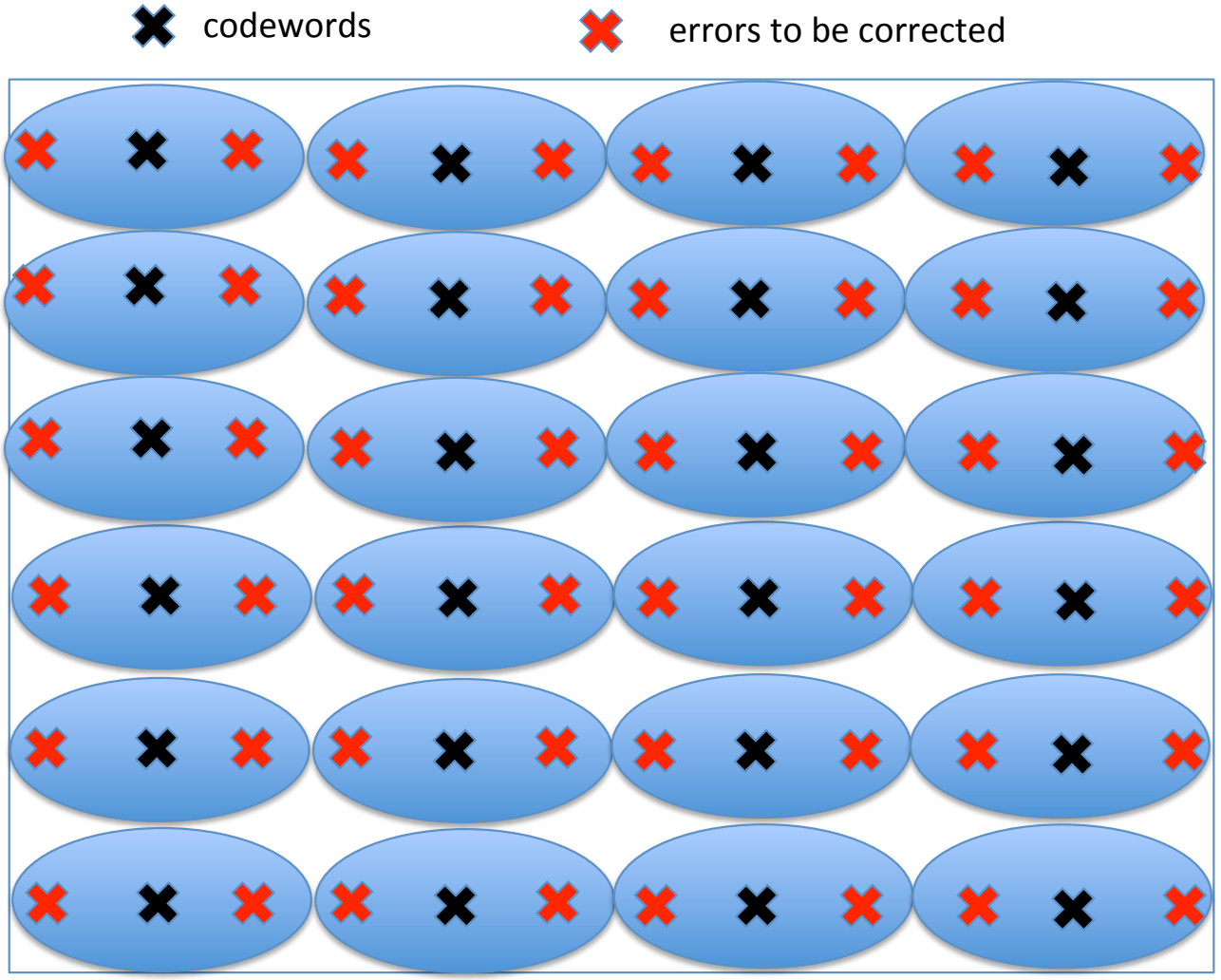
Symmetric vs. non-symmetric error spheres



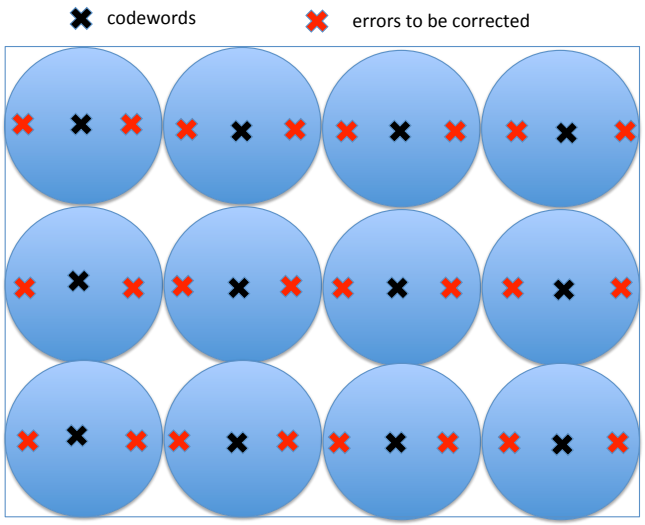
Symmetric vs. non-symmetric error spheres

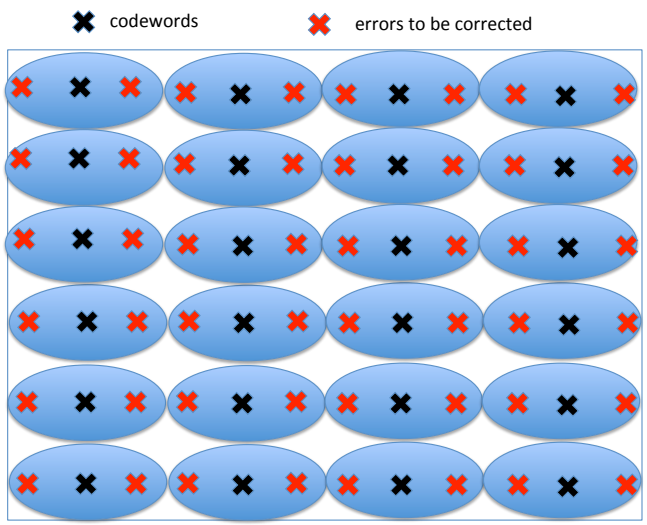


Symmetric vs. non-symmetric error spheres

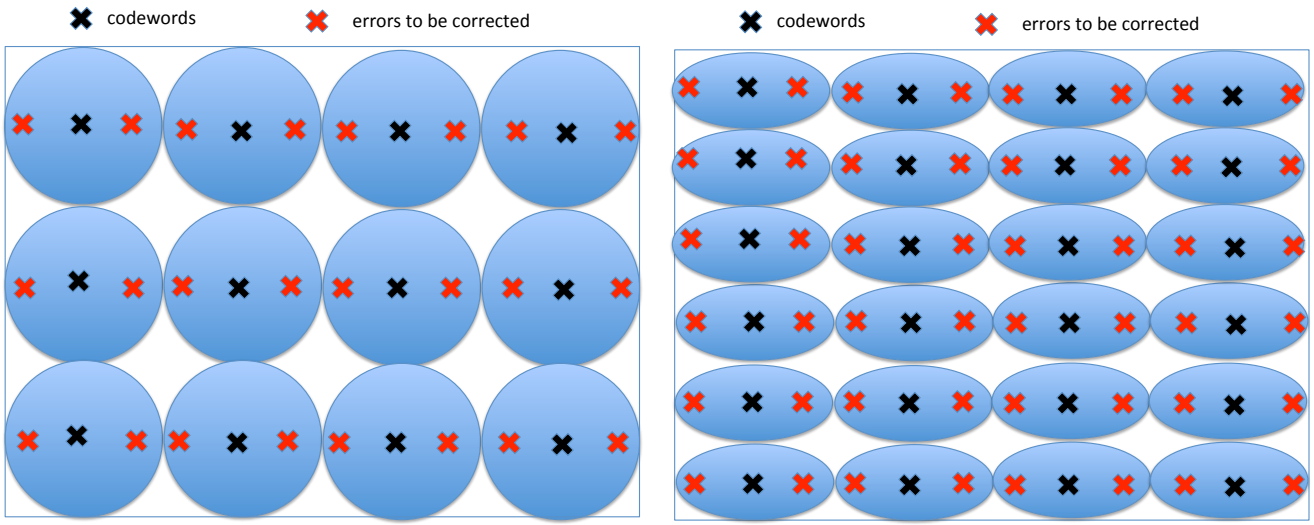


Symmetric vs. non-symmetric error spheres



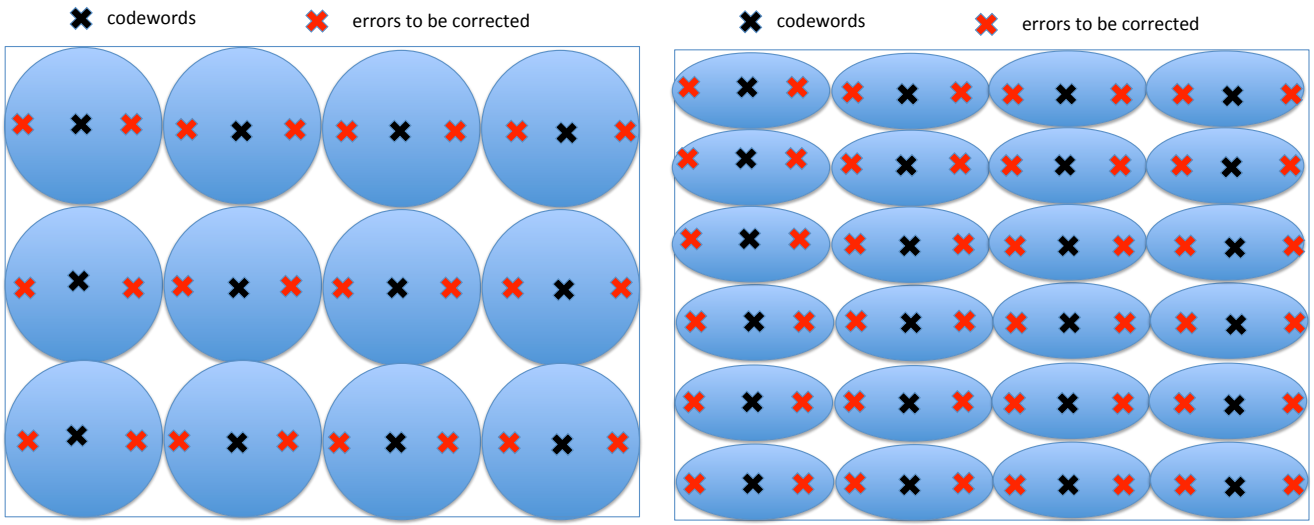


Symmetric vs. non-symmetric error spheres



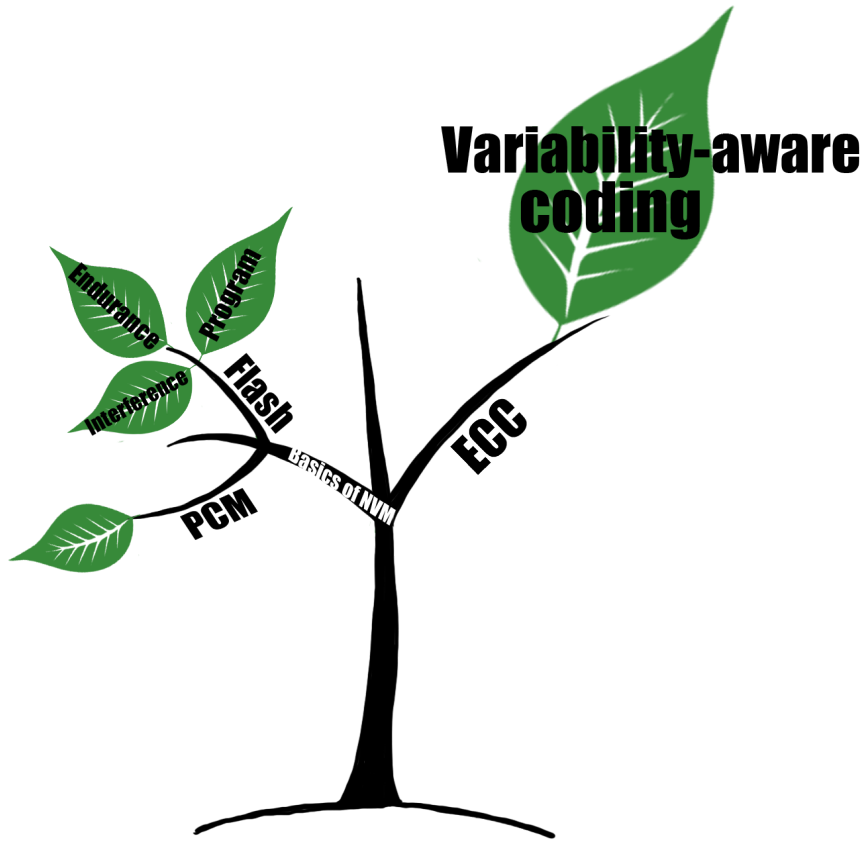
Conventional codes are optimized for the Hamming metric.

Symmetric vs. non-symmetric error spheres



Conventional codes are optimized for the Hamming metric.

1. Do not differentiate among different types of symbol errors
2. Result in unnecessary overhead (rate loss)


Variability-aware coding
Endurance
Program
Interference
Flash
PCM
Basics of NVM
ECC

Variability-aware coding

Key idea

Control for both the number of cells to be corrected as well as the number of bits per erroneous cell to be corrected.

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

1. There are 3 symbols in error

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

1. There are 3 symbols in error

2. There are 3 symbols in error, with at most 2 erroneous bits each

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

1. There are 3 symbols in error
2. There are 3 symbols in error, with at most 2 erroneous bits each
3. There are 3 symbols in error, with at most 2 erroneous bits each, and there is 1 symbol with more than 1 bit in error

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

1. There are 3 symbols in error
   $\rightarrow$ Code that corrects t symbols
2. There are 3 symbols in error, with at most 2 erroneous bits each
3. There are 3 symbols in error, with at most 2 erroneous bits each, and there is 1 symbol with more than 1 bit in error

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

1. There are 3 symbols in error
   $\rightarrow$ Code that corrects t symbols
2. There are 3 symbols in error, with at most 2 erroneous bits each
   $\rightarrow$ $[t, \ell]$ code that corrects t symbols, each with at most ℓ bits in error
3. There are 3 symbols in error, with at most 2 erroneous bits each, and there is 1 symbol with more than 1 bit in error

Variability-aware coding

An example:

- Suppose (000 110 010 101 000 111) is stored
- Suppose (101 110 000 101 000 011) is retrieved

Error pattern characterization:

1. There are 3 symbols in error
   $\rightarrow$ Code that corrects t symbols
2. There are 3 symbols in error, with at most 2 erroneous bits each
   $\rightarrow$ $[t, \ell]$ code that corrects t symbols, each with at most ℓ bits in error
3. There are 3 symbols in error, with at most 2 erroneous bits each, and there is 1 symbol with more than 1 bit in error
   $\rightarrow$ $[t_1, t_2; \ell_1, \ell_2]$ code that corrects $t_1 + t_2$ symbols, each with at most ℓ_1 bits in error, and at most t_2 symbols with more than ℓ_2 bits in error

New code constructions

Theorem (Construction 2)

- *Let H_1 be the parity check matrix of a $[m, k_1, \ell]_2$ code $\mathcal{C}_1$ (standard $[n, k, e]$ notation).*

- k_1 denotes the number of message bits, m denotes the number of coded bits and ℓ is the number of bit errors code $\mathcal{C}_1$ can correct.

J. K. Wolf, "On codes derivable from the tensor product of check matrices", T-IT, 1965.

New code constructions

Theorem (Construction 2)

- *Let* H_1 *be the parity check matrix of a* $[m, k_1, \ell]_2$ *code* $\mathcal{C}_1$ *(standard* $[n, k, e]$ *notation).*
- *Let* H_2 *be the parity check matrix of a* $[n, k_2, t]_{2^{m-k_1}}$ *code* $\mathcal{C}_2$ *defined over the alphabet of size* $GF(2)^{m-k_1}$.

- k_2 denotes the number of message symbols, n denotes the number of coded symbols and t is the number of errors code $\mathcal{C}_2$ can correct.

J. K. Wolf, "On codes derivable from the tensor product of check matrices", T-IT, 1965.

New code constructions

Theorem (Construction 2)

- *Let H_1 be the parity check matrix of a $[m, k_1, \ell]_2$ code $\mathcal{C}_1$ (standard $[n, k, e]$ notation).*
- *Let H_2 be the parity check matrix of a $[n, k_2, t]_{2^{m-k_1}}$ code $\mathcal{C}_2$ defined over the alphabet of size $GF(2)^{m-k_1}$.*
- *Then, $H_A = H_2 \otimes H_1$ is the parity check matrix of a $[t; \ell]_{2^m}$ code of length n.*

J. K. Wolf, "On codes derivable from the tensor product of check matrices", T-IT, 1965.

New code constructions

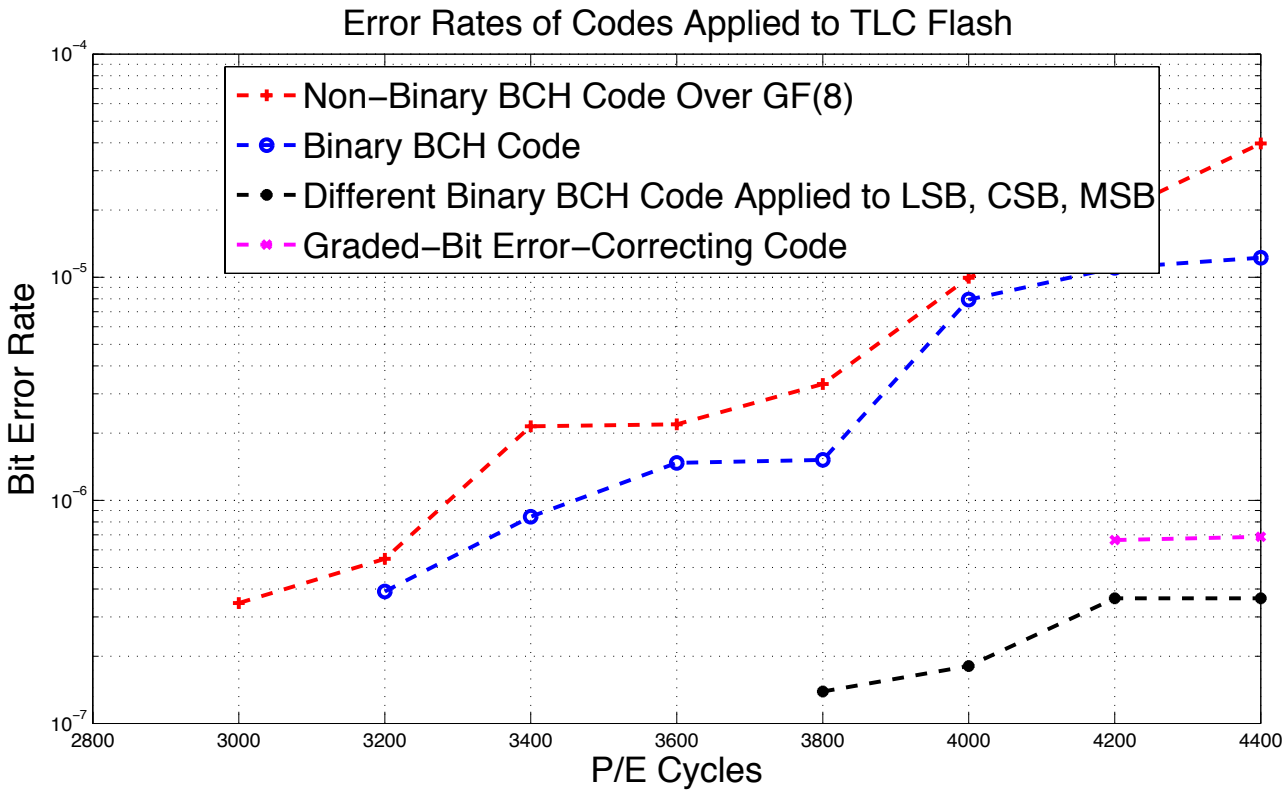
- Choose $H = \begin{bmatrix} H_1 \\ H_3 \end{bmatrix}$ as a parity check matrix of a $[m, k_1, \ell_2]_2$ code and H_1 as a parity check matrix of a $[m, m-r, \ell_1]_2$ code
  - H_1 is an $r \times m$ matrix, H_3 is an $s \times m$ matrix.
- Suppose H_2 is a parity check matrix of a $[n, k_2, t_1 + t_2]_{2^r}$ code.
- Suppose H_4 is a parity check matrix of a $[n, k_3, t_2]_{2^s}$ code.

Theorem (Construction 3)

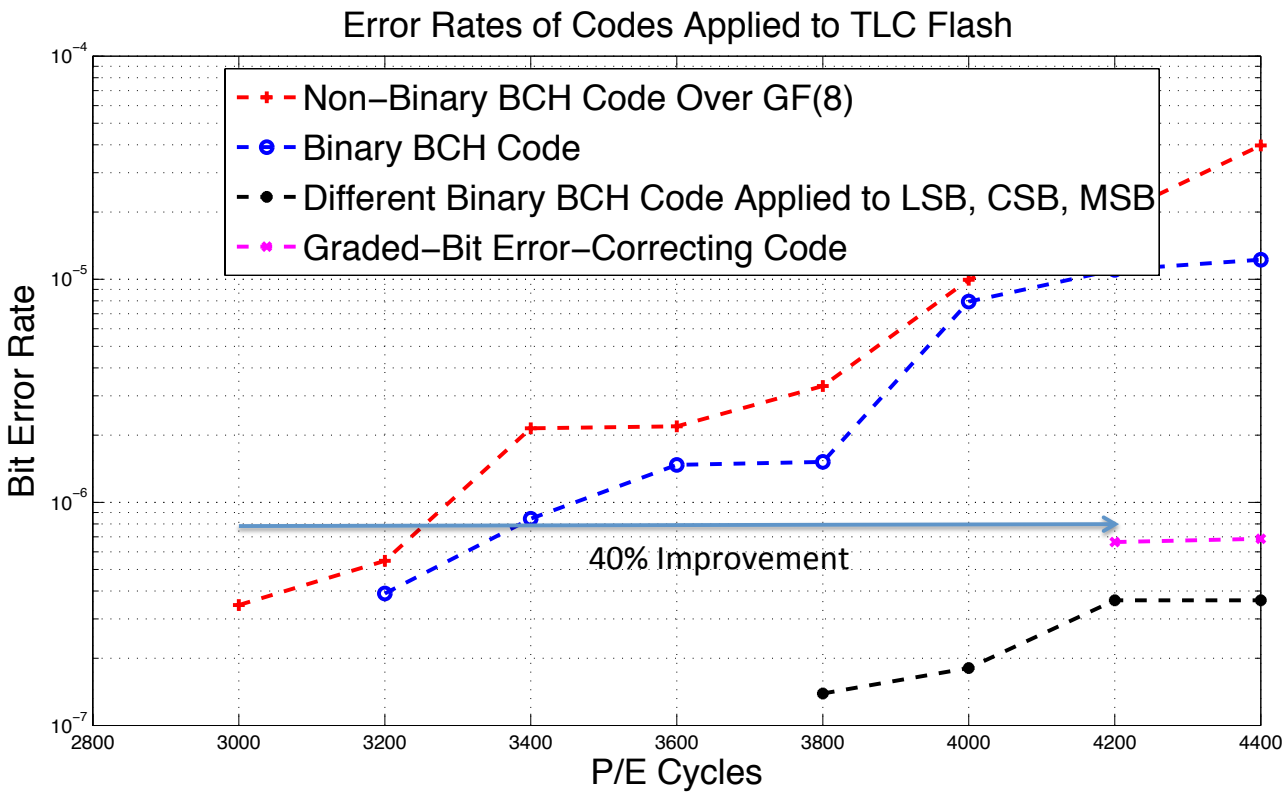
Then, H_B is the parity check matrix of a $[t_1, t_2; \ell_1, \ell_2]_{2^m}$ code, where

$$H_B = \begin{pmatrix} H_2 \otimes H_1 \\ H_4 \otimes H_3 \end{pmatrix}.$$

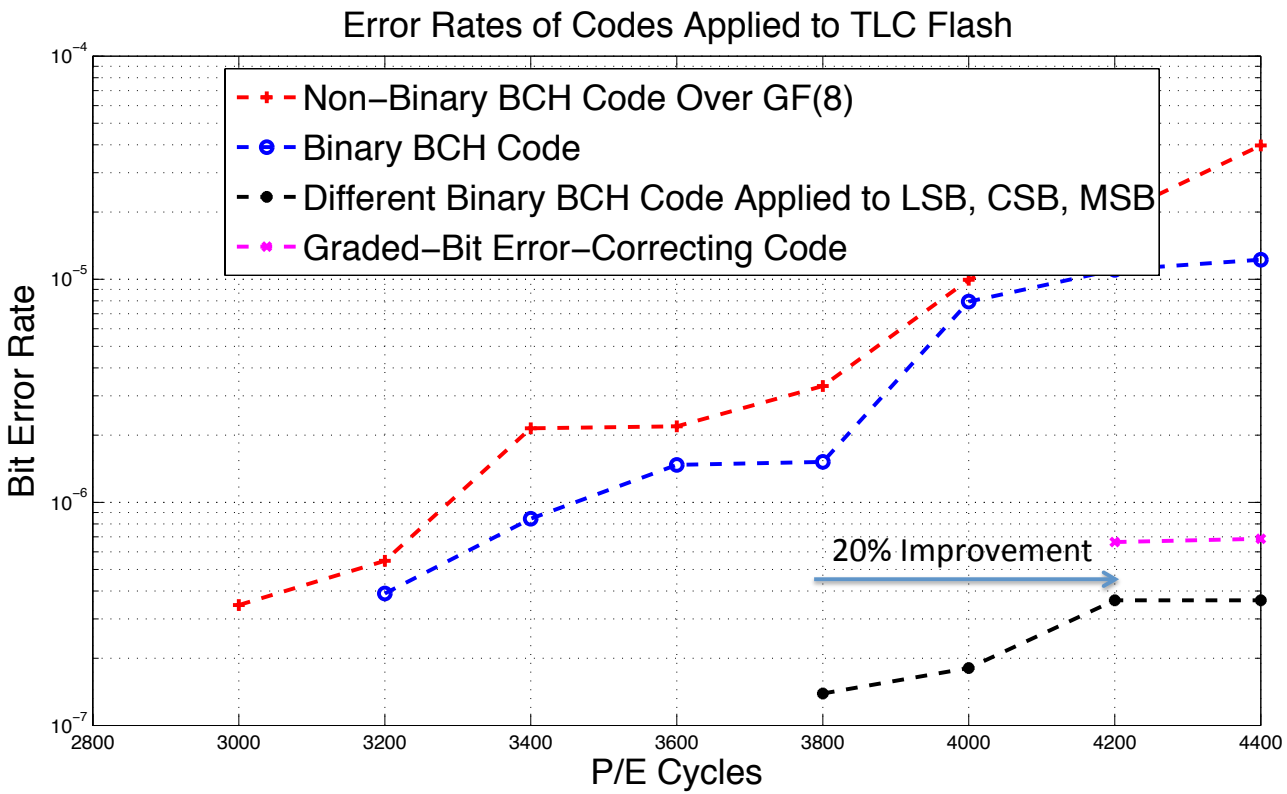
Performance evaluation for codes of length 4000 bits and rate 0.86

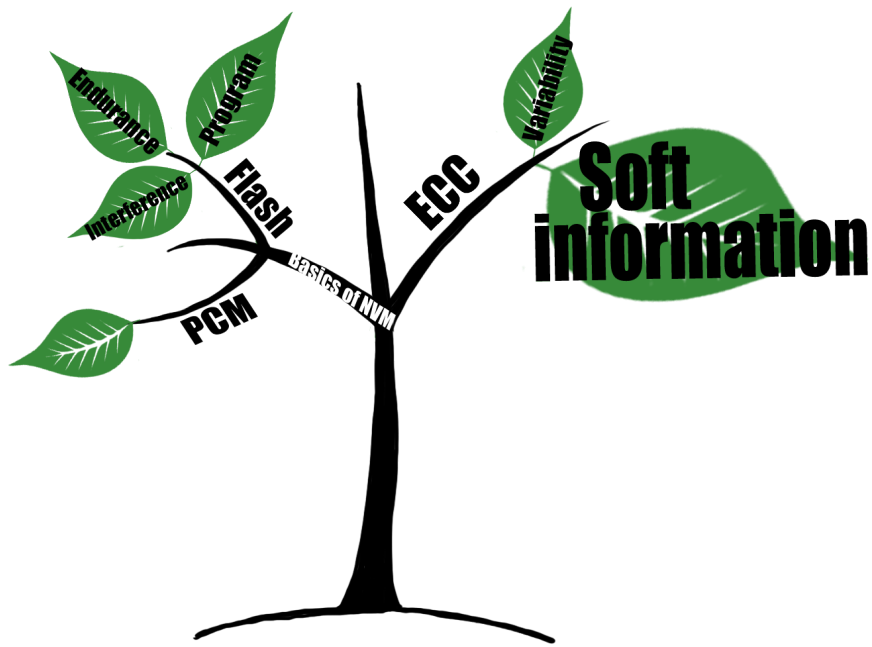


Performance evaluation for codes of length 4000 bits and rate 0.86



Performance evaluation for codes of length 4000 bits and rate 0.86




Endurance
Program
Interference
Flash
PCM
Basics of NVM
ECC
Variability
Soft information

Exploiting level distributions

Exploiting level distributions

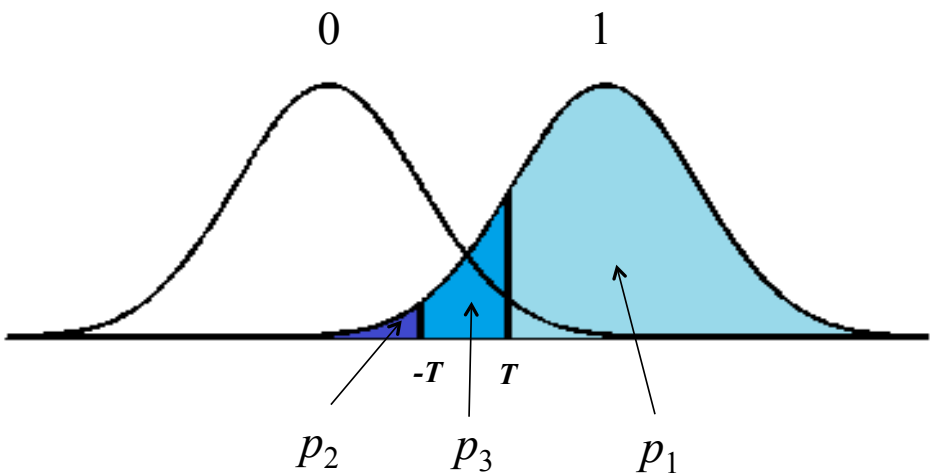
- What high-performance codes are amenable for soft decoding ?

LDPC for Flash: Extracting soft information

- Idea: multiple word line reads

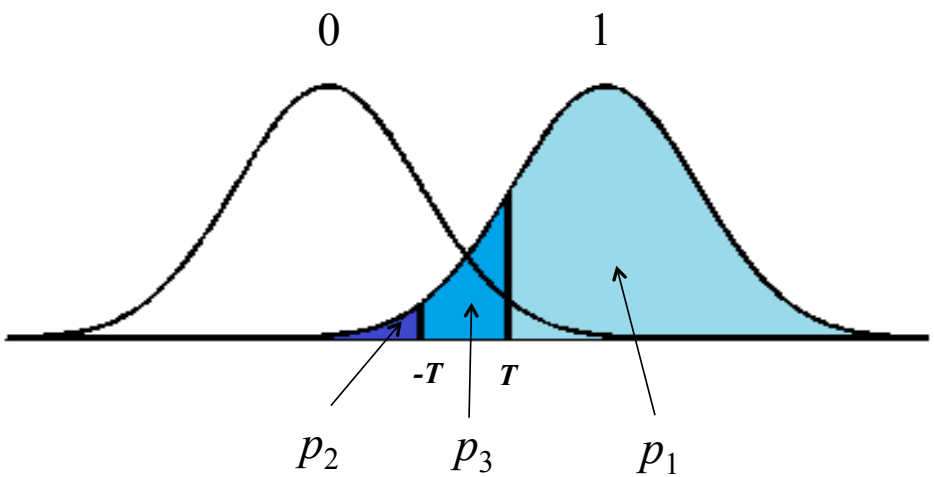
LDPC for Flash: Extracting soft information

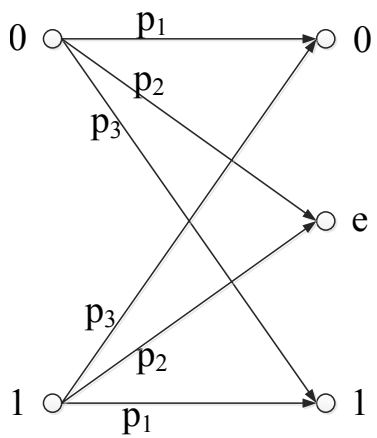
- Idea: multiple word line reads
- 2 reads compare against two thresholds



LDPC for Flash: Extracting soft information

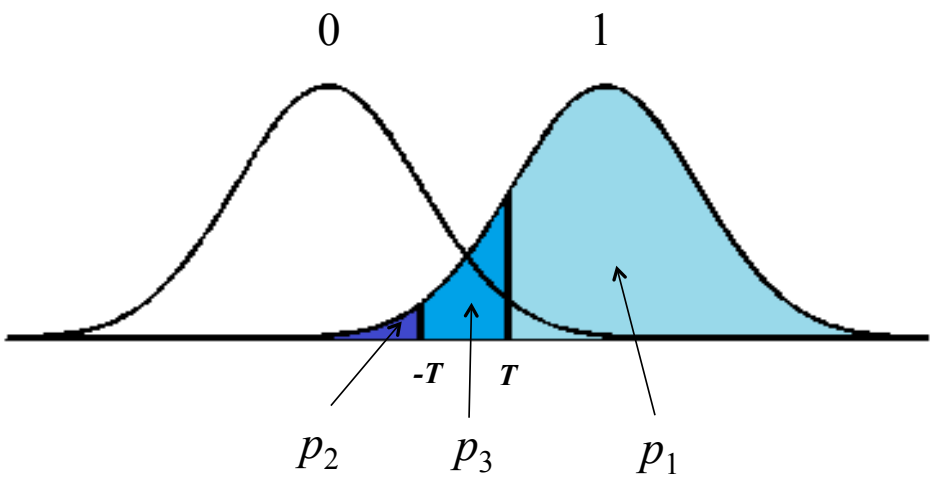
- Idea: multiple word line reads
- 2 reads compare against two thresholds

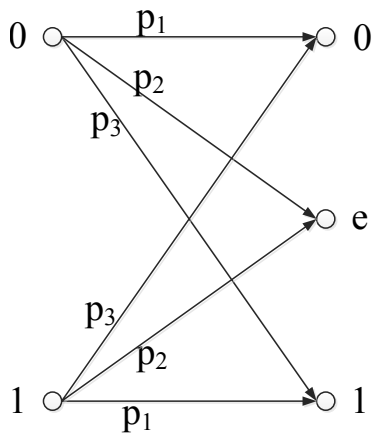




LDPC for Flash: Extracting soft information

- Idea: multiple word line reads
- 2 reads compare against two thresholds





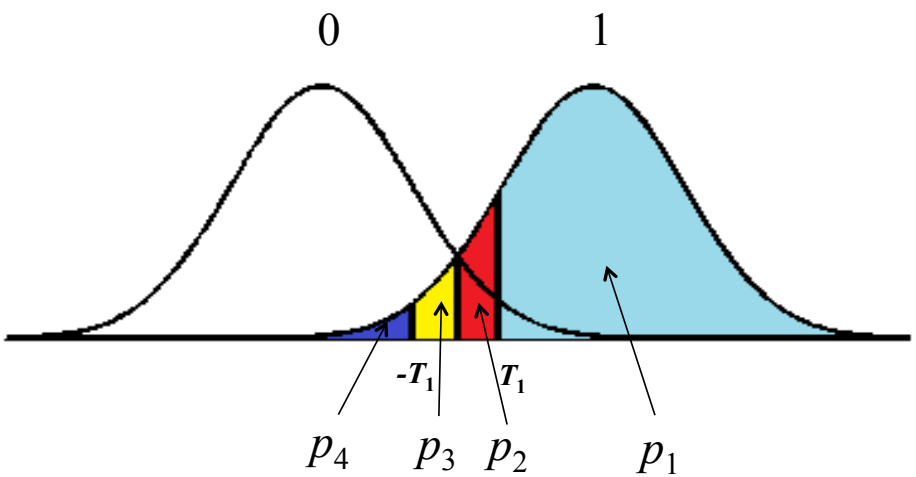
- Maximize mutual information of the induced channel to determine the best thresholds (here T and $-T$)

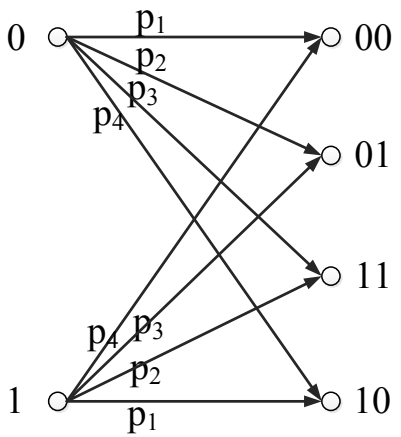
LDPC for Flash: Extracting soft information

- Idea: multiple word line reads

LDPC for Flash: Extracting soft information

- Idea: multiple word line reads
- 3 reads compare against three thresholds

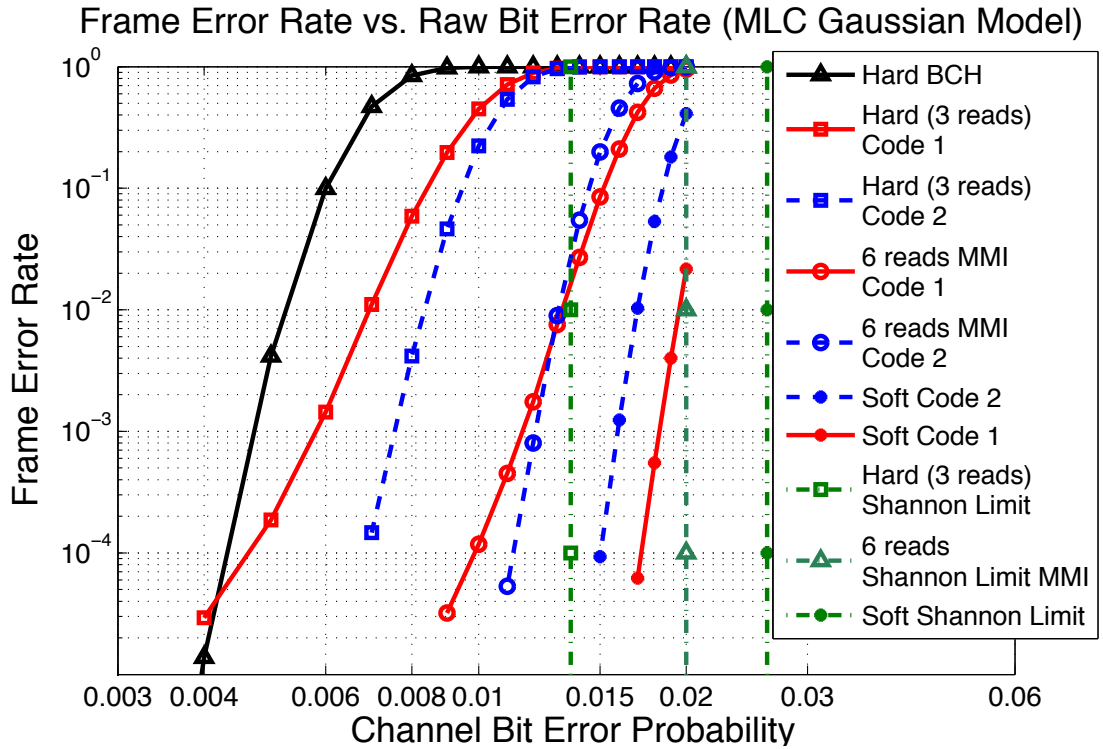




- Maximize mutual information of the induced channel to determine the best thresholds (here T_1, $-T_1$ and 0)

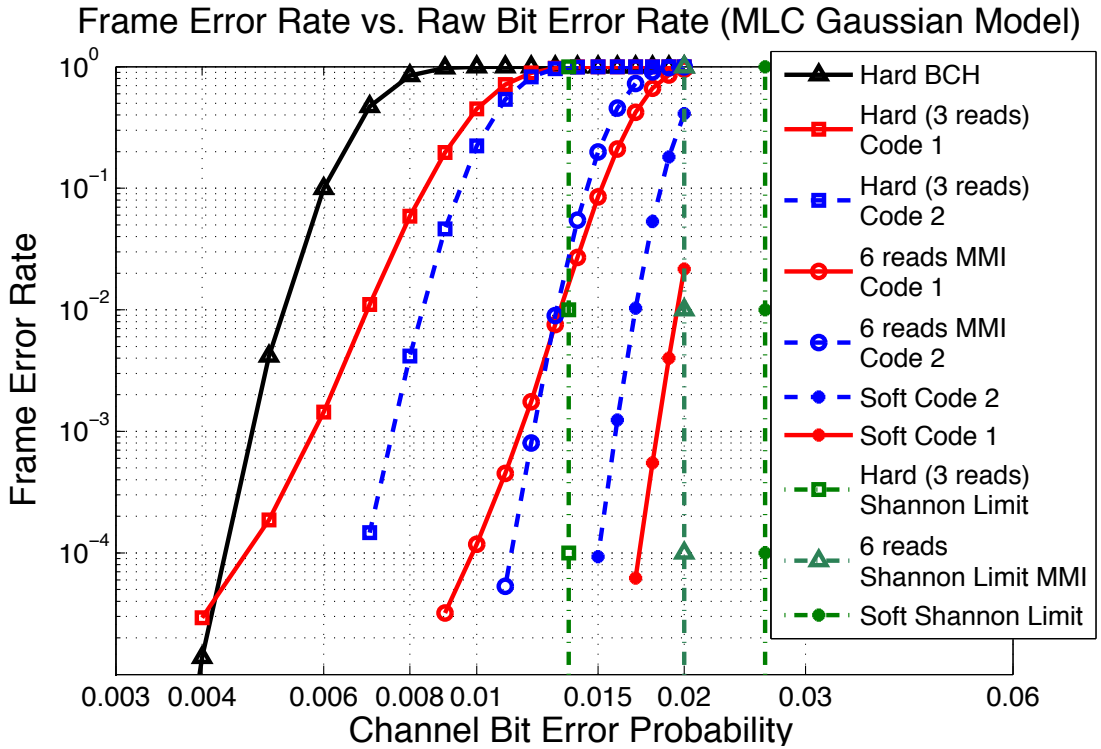
Performance with multi read

Figure: Performance comparison for 0.9-rate LDPC and BCH codes of length $n = 9100$.



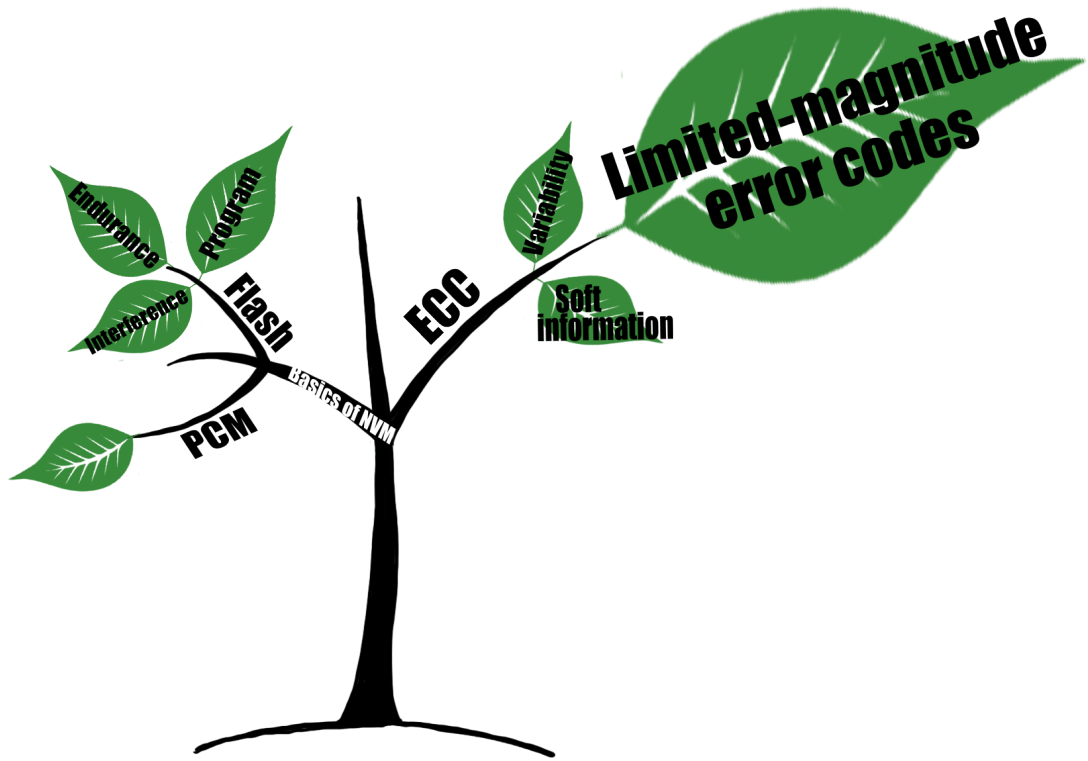
Performance with multi read

Figure: Performance comparison for 0.9-rate LDPC and BCH codes of length $n = 9100$.



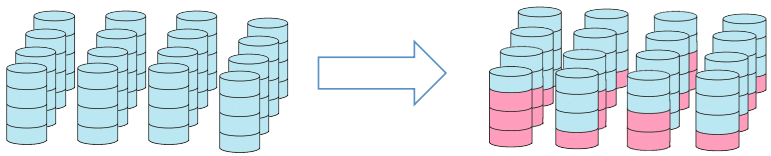
Caution:

- Optimal code design in the error floor region depends on the chosen quantization.
- AWGN-optimized LDPC codes may not be the best for the quantized (and asymmetric) Flash channel !


Limited-magnitude error codes
Variability
ECC
Soft information
Endurance
Program
Interference
Flash
Basics of NVM
PCM

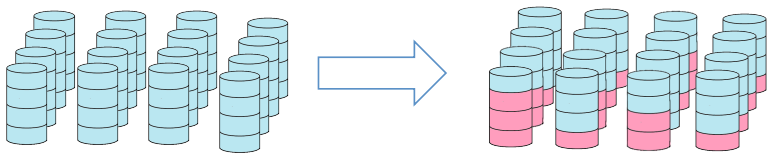
Motivation: Write latency

- Recall Flash programming

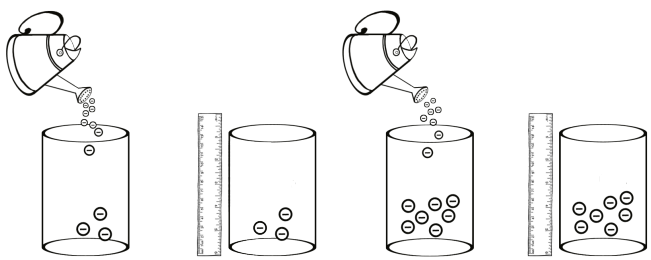


Motivation: Write latency

- Recall Flash programming

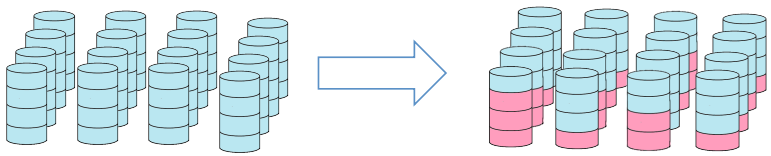


- In practice incremental step pulse programming is used, a.k.a. guess and verify.
- Latency increases with number of levels.

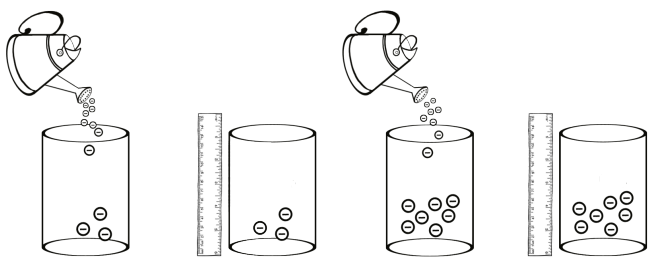


Motivation: Write latency

- Recall Flash programming



- In practice incremental step pulse programming is used, a.k.a. guess and verify.
- Latency increases with number of levels.



- If one allows for "dirty writes", it suffices to correct errors in only one direction.

Coding for asymmetric, limited-magnitude errors

- If one allows for "dirty writes", it suffices to correct errors in only one direction and that are of limited magnitude.

Definition (ALM Code)

Let $\mathcal{C}_1$ be a code over the alphabet Q_1. The code $\mathcal{C}$ over the alphabet Q (with $|Q| > |Q_1| = q_1 = \ell + 1$) is defined as
$\mathcal{C} = \{\mathbf{x} = (x_1, x_2, \ldots, x_n) \in Q^n \mid \mathbf{x} \mod q_1 \in \mathcal{C}_1\}$.

Coding for asymmetric, limited-magnitude errors

- If one allows for "dirty writes", it suffices to correct errors in only one direction and that are of limited magnitude.

Definition (ALM Code)

Let $\mathcal{C}_1$ be a code over the alphabet Q_1. The code $\mathcal{C}$ over the alphabet Q (with $|Q| > |Q_1| = q_1 = \ell + 1$) is defined as
$\mathcal{C} = \{\mathbf{x} = (x_1, x_2, \ldots, x_n) \in Q^n \mid \mathbf{x} \mod q_1 \in \mathcal{C}_1\}$.

Theorem

Code $\mathcal{C}$ corrects t asymmetric errors of limited magnitude ℓ if code $\mathcal{C}_1$ corrects t symmetric errors.

- New construction inherits encoding/decoding complexity of the underlying (symmetric) ECC.
- Connections with additive number theory and asymmetric ECC (Varshamov 1970s).

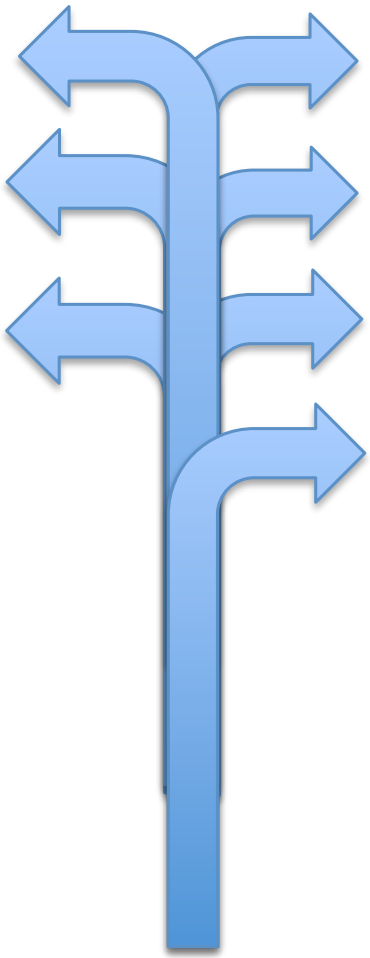
Further Reading (1/2)

- Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited magnitude errors with application to multilevel flash memories", T-IT, 2010.
- S. Yari, T. Kløve, and B. Bose, "Some codes correcting unbalanced errors of limited magnitude for flash memories", T-IT, 2013.
- S. Bizaglo and T. Etzion, "Tilings with n-dimensional chairs and their applications to asymmetric codes", T-IT,2013.
- H. Zhou, A. Jiang, and J. Bruck, "Nonuniform codes for correcting asymmetric errors in data storage", T-IT, 2013.
- M. Blaum, "Codes for detecting and correcting unidirectional errors", 1993.

Further Reading (2/2)

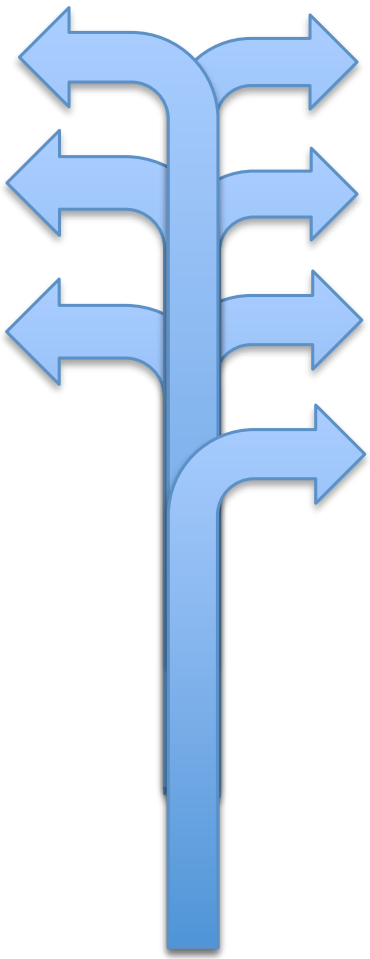
- R. Gabrys, E. Yaakobi, and L. Dolecek, "Graded bit error correcting codes with applications to flash memory", T-IT, 2013.
- P. Vontobel and R. Roth, "Coding for combined block-symbol error correction", T-IT, 2014.
- J. Wang *et al.*, "Enhanced precision through multiple reads for LDPC decoding in flash memories", JSAC, 2014.
- J. Wang, L. Dolecek, and R. D. Wesel, "The cycle consistency matrix approach to LDPC absorbing sets in separable circulant-based codes", T-IT, 2013.
- K. Haymaker and C. Kelley, "Structured bit-interleaved LDPC codes for MLC flash memory", JSAC, 2014.
- A. Jiang, H. Li, and J. Bruck, "On the capacity and programming of flash memories", T-IT, 2012.

Research problems on coding for spatio-temporal variability in NVMs



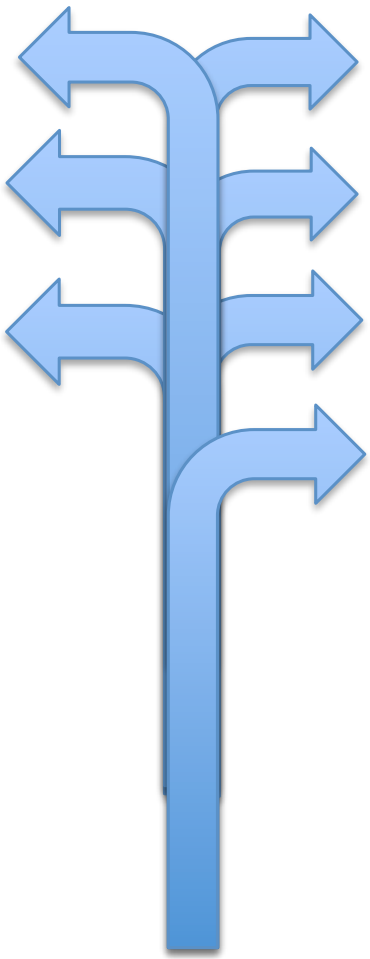
1. Investigation of algebraic codes for both transient and permanent errors

Research problems on coding for spatio-temporal variability in NVMs



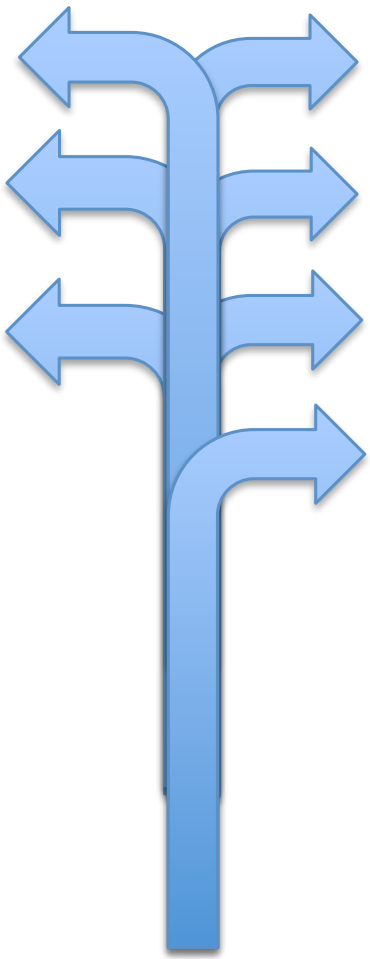
1. Investigation of algebraic codes for both transient and permanent errors
2. Investigation of graph-based binary and non-binary codes and their decoders for flash (LDPC, spatially coupled codes etc.)

Research problems on coding for spatio-temporal variability in NVMs



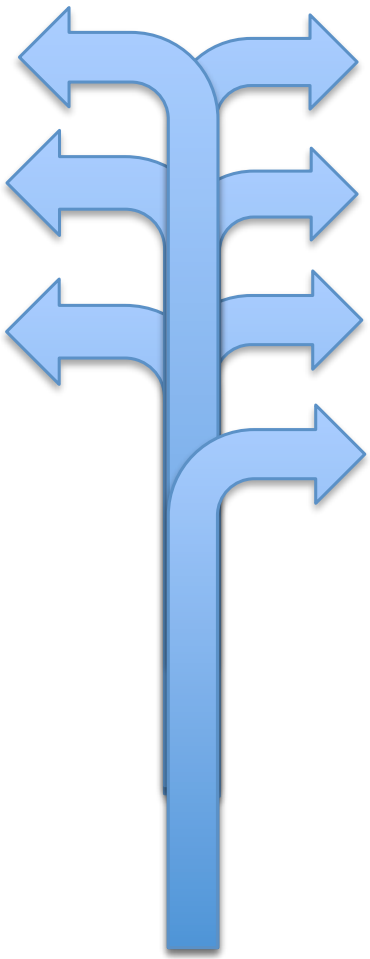
1. Investigation of algebraic codes for both transient and permanent errors
2. Investigation of graph-based binary and non-binary codes and their decoders for flash (LDPC, spatially coupled codes etc.)
3. Constructions with tighter bounds, i.e., beyond asymptotic optimality

Research problems on coding for spatio-temporal variability in NVMs



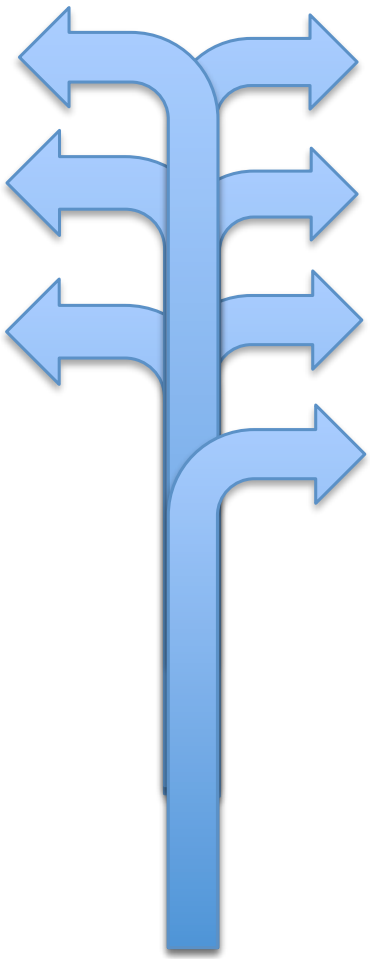
1. Investigation of algebraic codes for both transient and permanent errors
2. Investigation of graph-based binary and non-binary codes and their decoders for flash (LDPC, spatially coupled codes etc.)
3. Constructions with tighter bounds, i.e., beyond asymptotic optimality
4. Noise-adaptive coding

Research problems on coding for spatio-temporal variability in NVMs



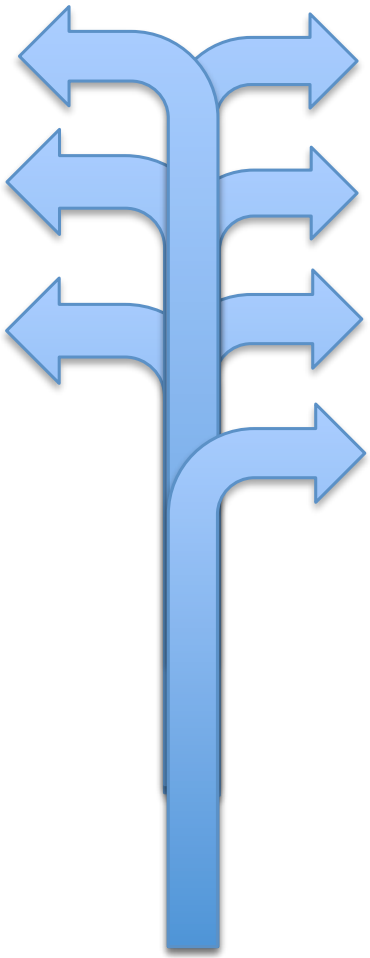
1. Investigation of algebraic codes for both transient and permanent errors
2. Investigation of graph-based binary and non-binary codes and their decoders for flash (LDPC, spatially coupled codes etc.)
3. Constructions with tighter bounds, i.e., beyond asymptotic optimality
4. Noise-adaptive coding
5. Coding for the target average performance

Research problems on coding for spatio-temporal variability in NVMs

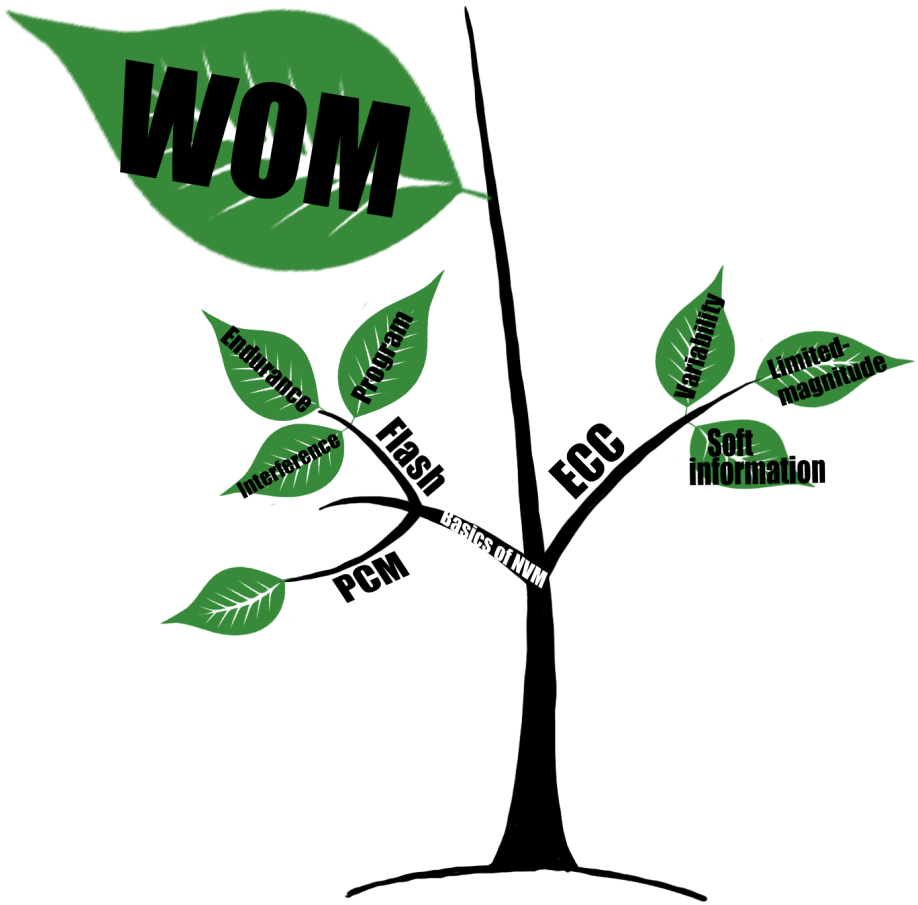


1. Investigation of algebraic codes for both transient and permanent errors
2. Investigation of graph-based binary and non-binary codes and their decoders for flash (LDPC, spatially coupled codes etc.)
3. Constructions with tighter bounds, i.e., beyond asymptotic optimality
4. Noise-adaptive coding
5. Coding for the target average performance
6. Deeper connections with number theoretic and combinatorial methods

Research problems on coding for spatio-temporal variability in NVMs

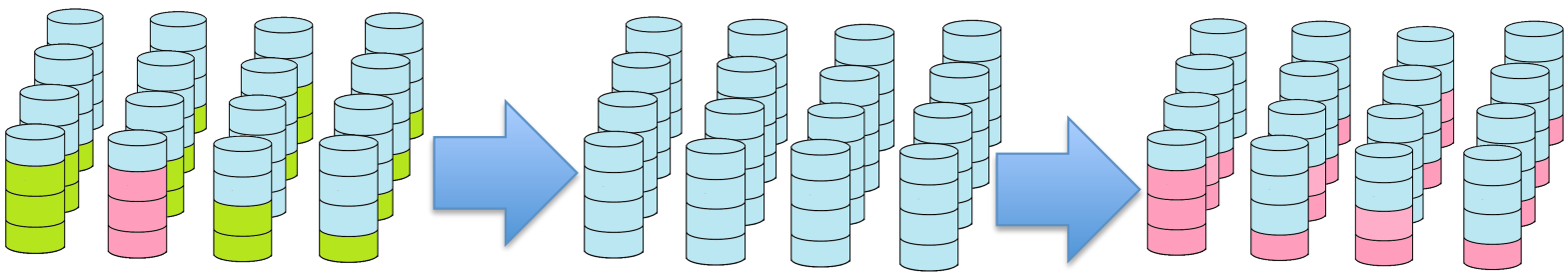


1. Investigation of algebraic codes for both transient and permanent errors
2. Investigation of graph-based binary and non-binary codes and their decoders for flash (LDPC, spatially coupled codes etc.)
3. Constructions with tighter bounds, i.e., beyond asymptotic optimality
4. Noise-adaptive coding
5. Coding for the target average performance
6. Deeper connections with number theoretic and combinatorial methods
7. Performance – complexity tradeoff and evaluations for more realistic channels

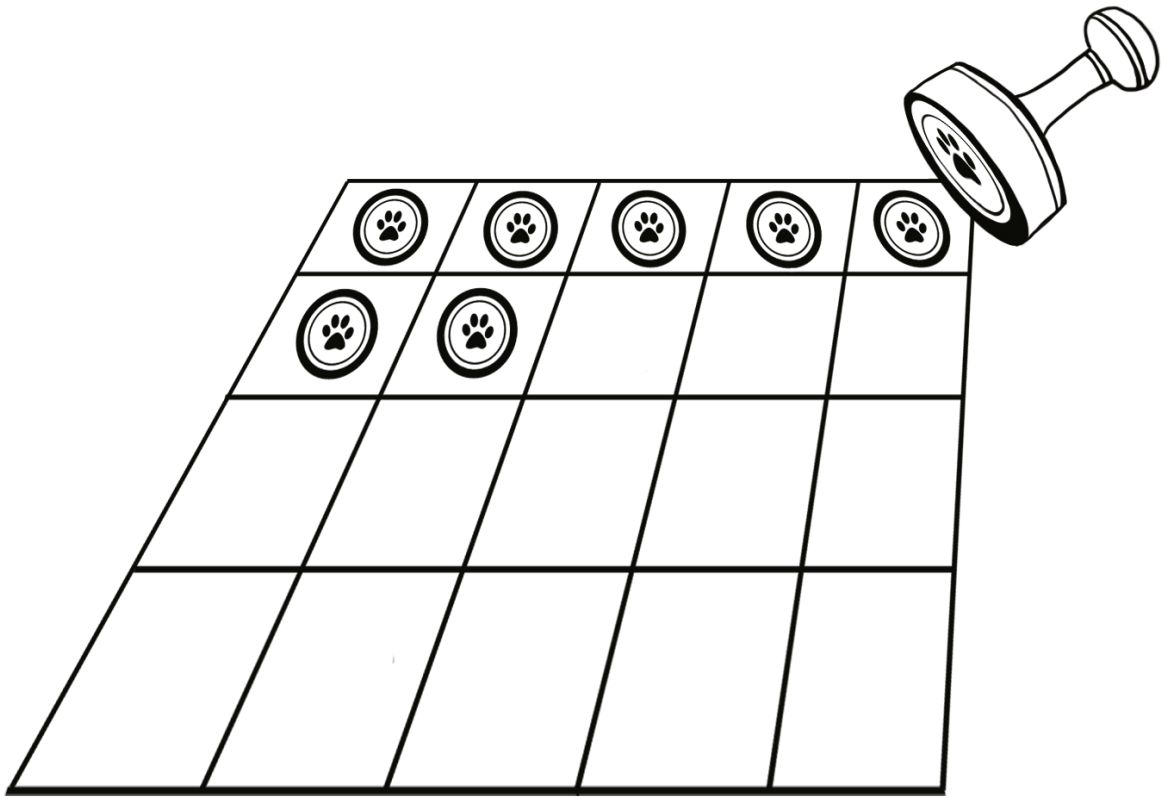

WOM
Endurance
Program
Interference
Flash
PCM
Basics of NVM
ECC
Variability
Limited-
magnitude
Soft
information

Motivation: Coding can defer costly erases

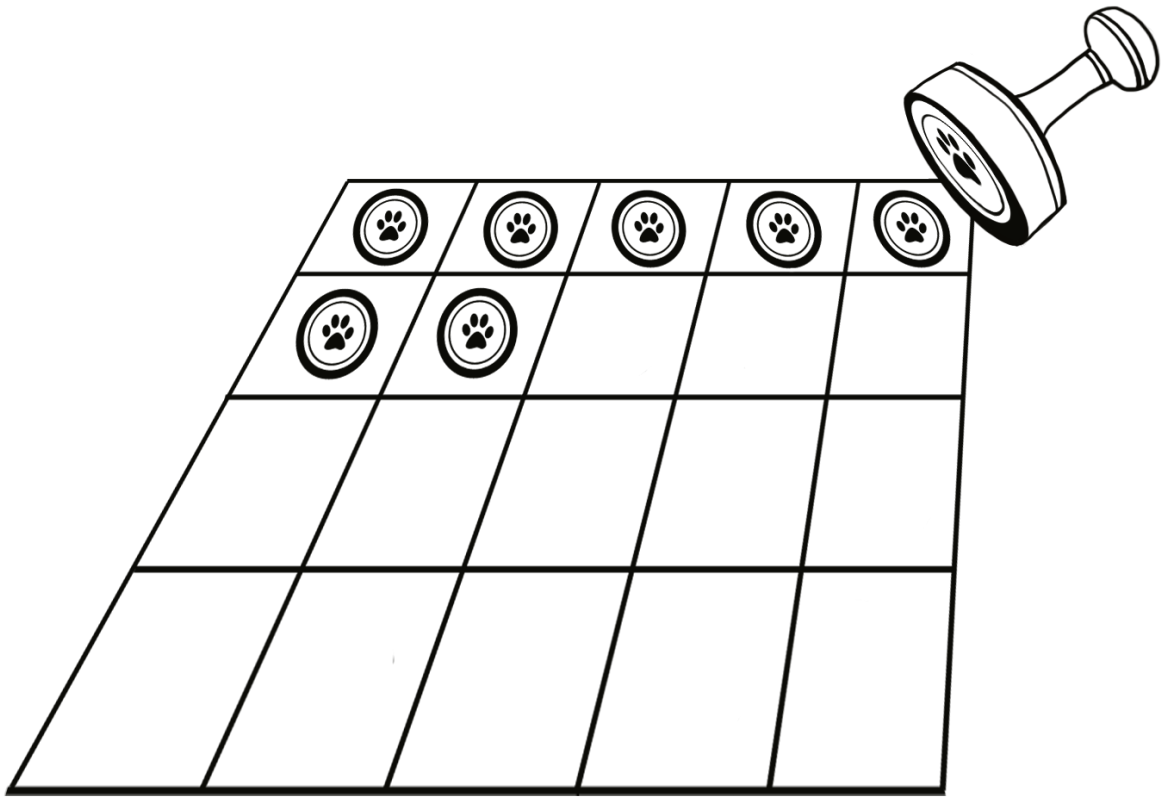
- Recall write and block-erase operations



- What if we can maximize the amount of data written before the block erase is necessary?

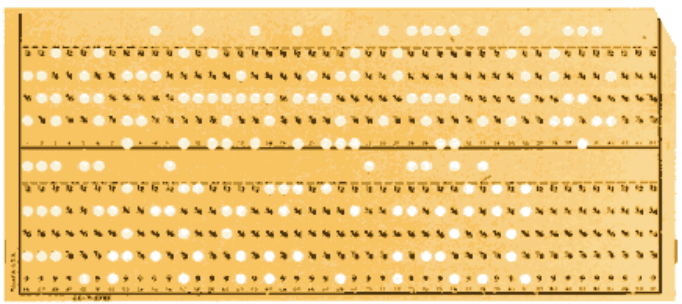


WOM Coding



Write-once memory (WOM)

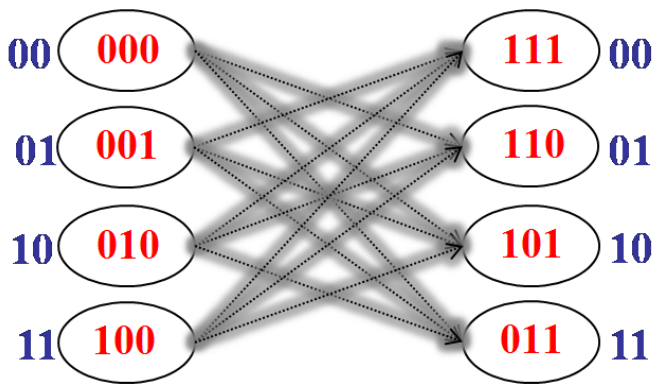
- Cells are irreversibly programmed from “0” to “1”
- Original applications of WOM were punch cards and optical disks



- WOM coding allows for increased storage utilization

The mother of all WOM codes

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011



Encoding example 1

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

Encoding example 1

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

- Write-1: 01 $\rightarrow$ Encode: 001.

Encoding example 1

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

- Write-1: 01 $\rightarrow$ Encode: 001.
- Write-2: 10 $\rightarrow$ Encode: 101.

Encoding example 1

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

- Write-1: 01 $\rightarrow$ Encode: 001.
- Write-2: 10 $\rightarrow$ Encode: 101.

Encoding example 2

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

Encoding example 2

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

- Write-1: 01 $\rightarrow$ Encode: 001.

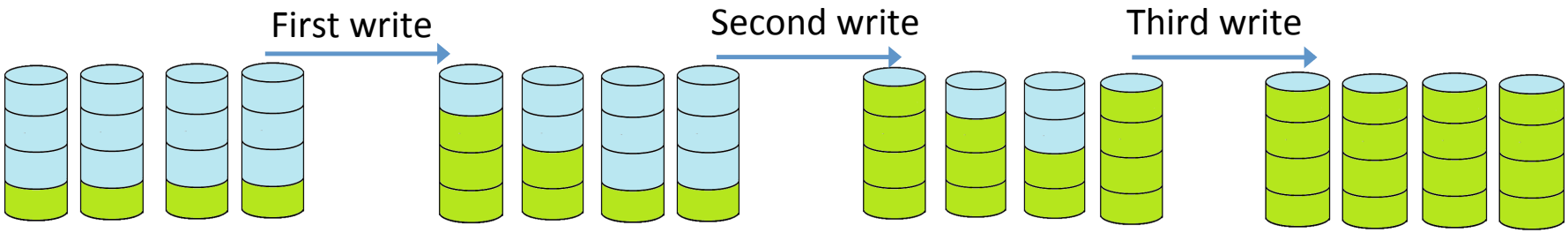
Encoding example 2

Rivest-Shamir WOM code:

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

- Write-1: 01 $\rightarrow$ Encode: 001.
- Write-2: 01 $\rightarrow$ Encode: 001. No change.

WOM model



Definition (WOM constraint)

The memory state is modeled as a vector $\mathbf{y}^j$ of length n where j is the current write (or generation). Each element y_i^j, $1 \leq i \leq n$, takes values in the set $\{0, 1, \ldots q-1\}$. On write j, the encoder writes one of M_j messages to the memory by updating $\mathbf{y}^{j-1}$ to $\mathbf{y}^j$ while satisfying the **WOM-constraint** $\mathbf{y}^j \geq \mathbf{y}^{j-1}$.

WOM model

Definition (Sum rate)

If M_j codewords can be represented at generation j, then generation j has rate $\frac{1}{n}\log(M_j)$. The **sum rate** is the sum of rates across generations.

WOM model

Definition (Sum rate)

If M_j codewords can be represented at generation j, then generation j has rate $\frac{1}{n}\log(M_j)$. The **sum rate** is the sum of rates across generations.

- Rivest-Shamir code has rate $\log(M_1 + M_2)/n = \log(4+4)/3 = 1.33$.

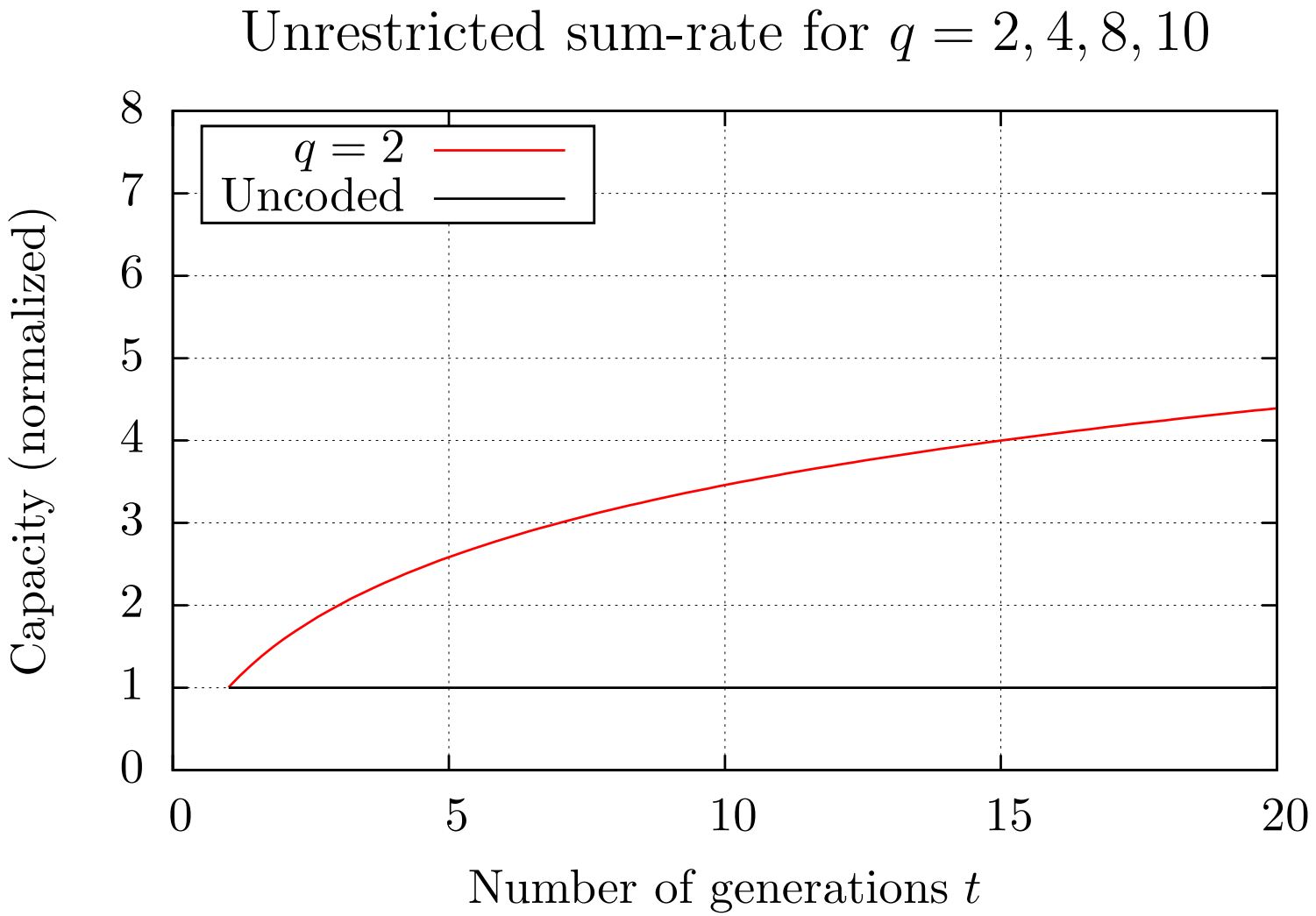
WOM model

Definition (Sum rate)

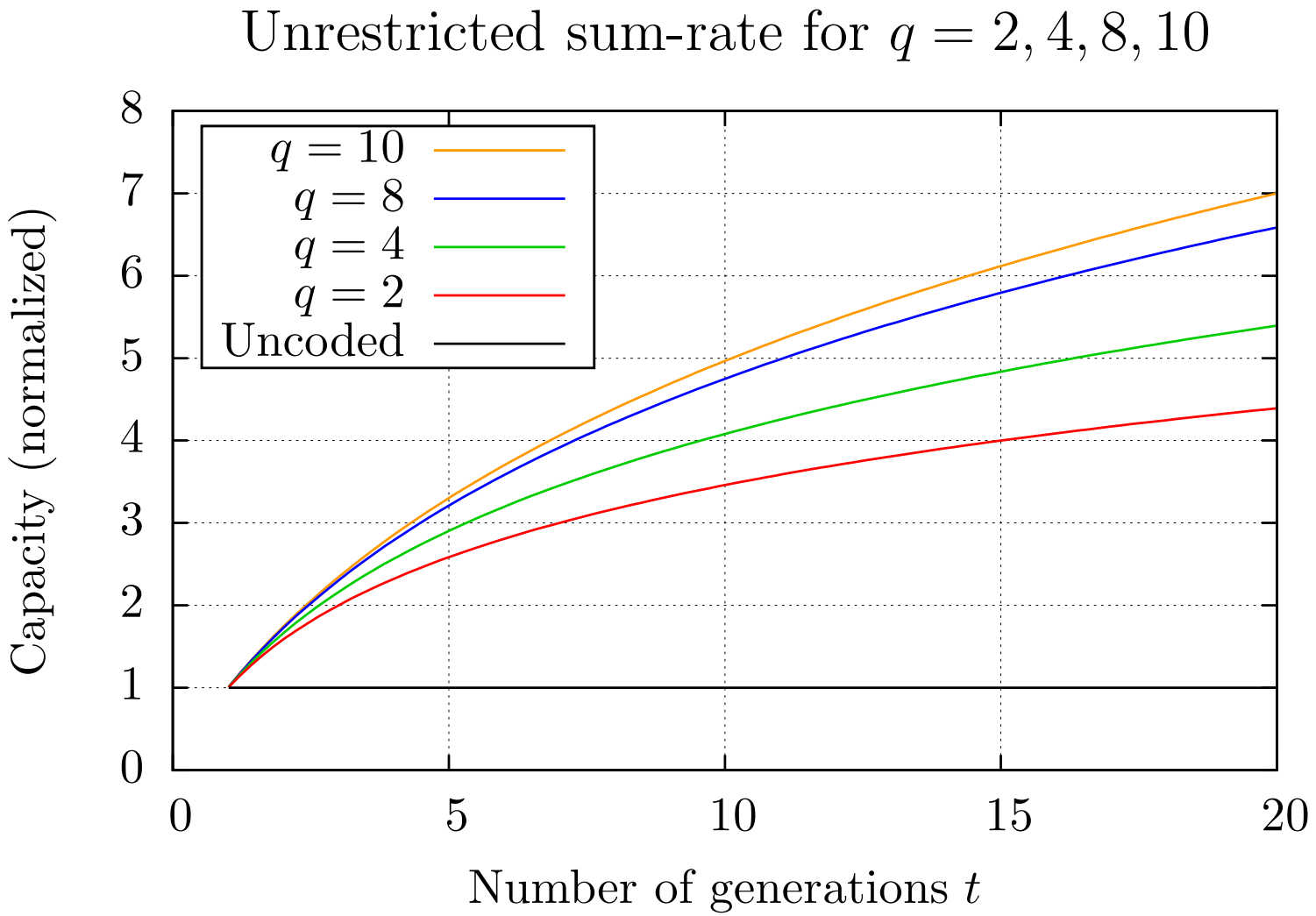
If M_j codewords can be represented at generation j, then generation j has rate $\frac{1}{n}\log(M_j)$. The **sum rate** is the sum of rates across generations.

- Rivest-Shamir code has rate $\log(M_1 + M_2)/n = \log(4+4)/3 = 1.33$.
- Capacity is the maximum achievable sum rate.

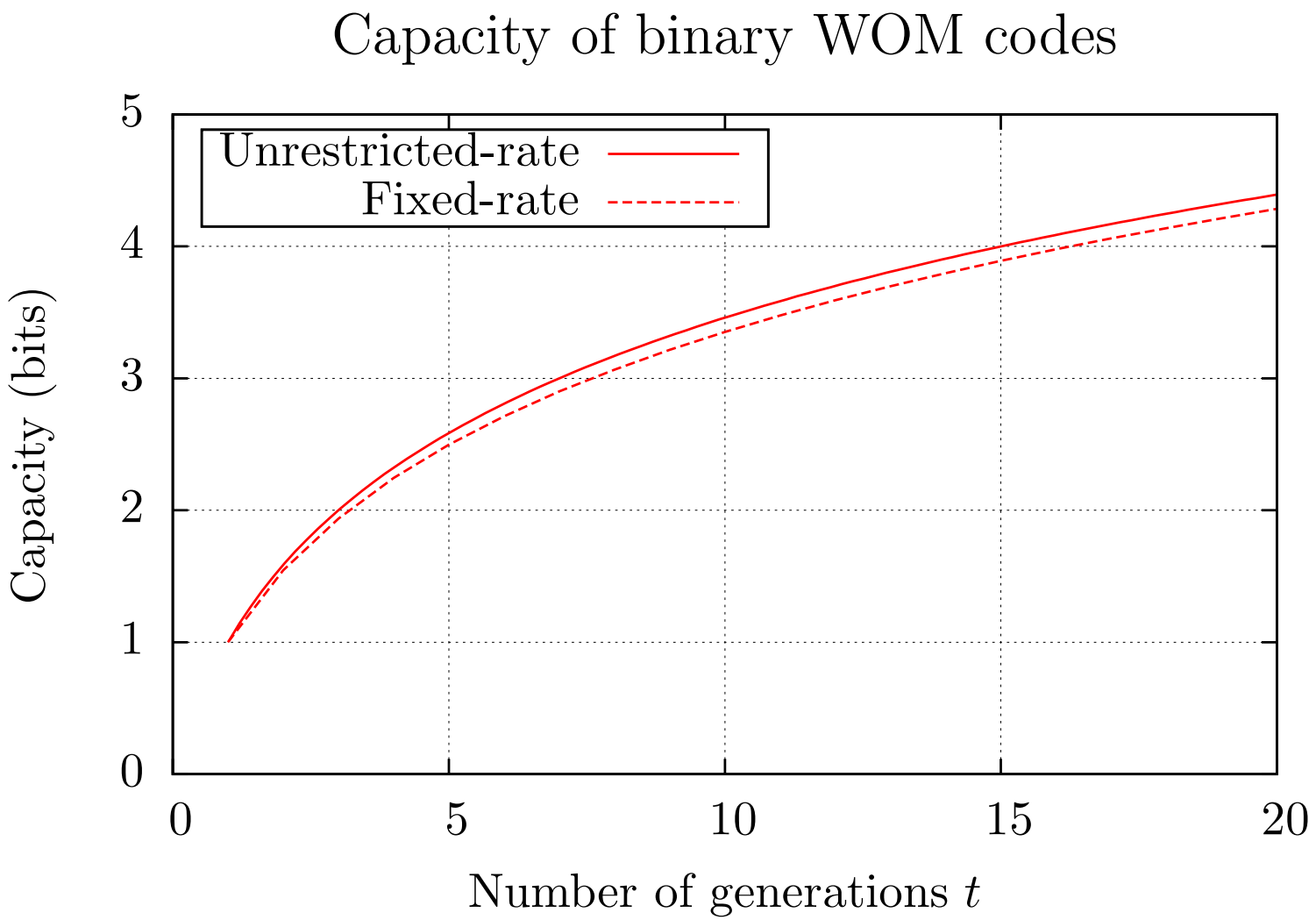
Capacity as a function of field order size



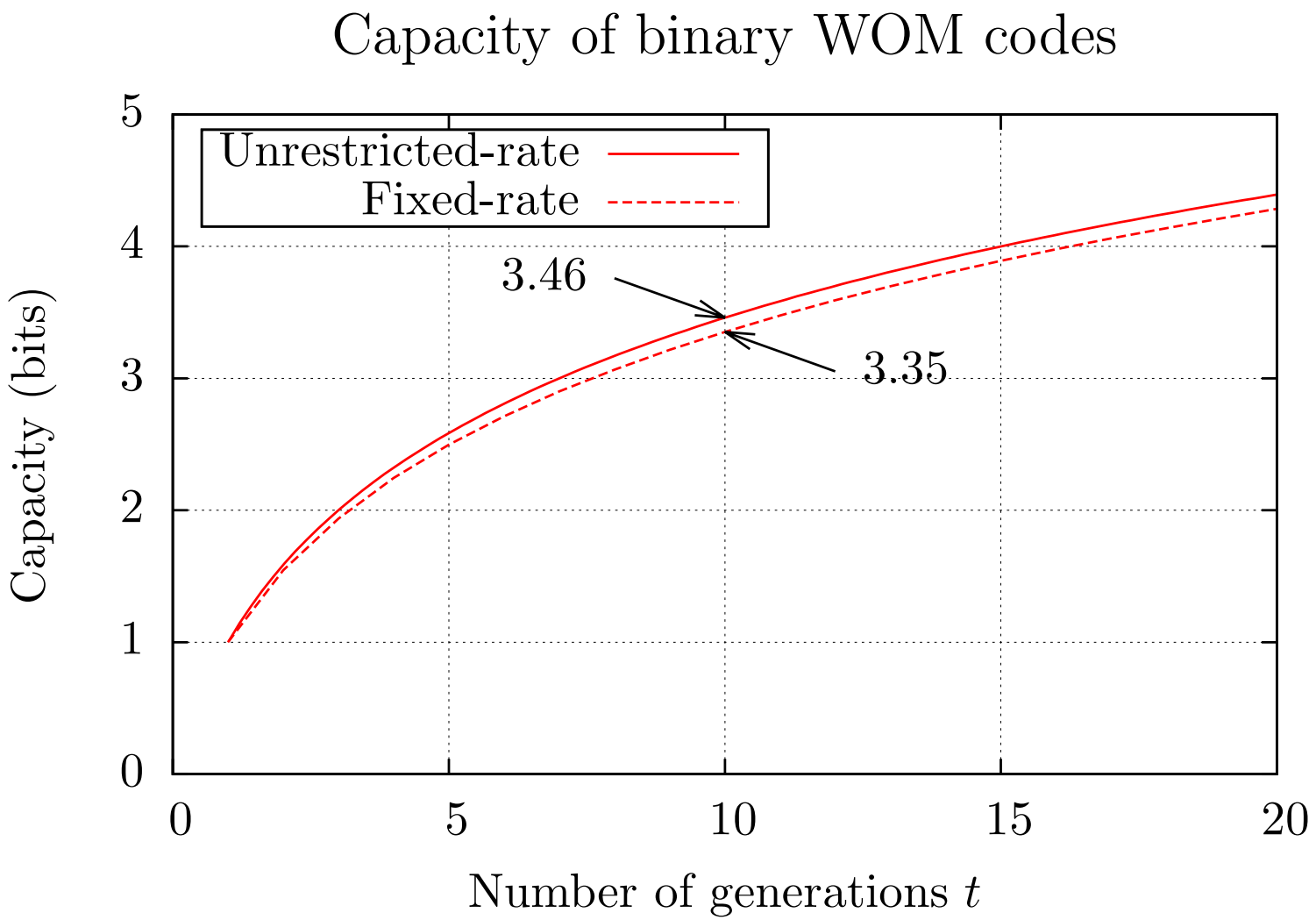
Capacity as a function of field order size



Equal rate vs. unequal rate per write



Equal rate vs. unequal rate per write



A construction of binary WOM codes

- We want to construct a binary three-write WOM over $2n$ cells.

A construction of binary WOM codes

- We want to construct a binary three-write WOM over $2n$ cells.
- Let $\mathcal{C}_3$ be a ternary two-write WOM of length n.

A construction of binary WOM codes

- We want to construct a binary three-write WOM over $2n$ cells.
- Let $\mathcal{C}_3$ be a ternary two-write WOM of length n.
- First write
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_3$.
  - Map length-n $\mathbf{u}$ into length-$2n$ binary string using $\Phi(0) = 00$, $\Phi(1) = 10$, $\Phi(2) = 01$.

A construction of binary WOM codes

- We want to construct a binary three-write WOM over $2n$ cells.
- Let $\mathcal{C}_3$ be a ternary two-write WOM of length n.
- First write
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_3$.
  - Map length-n $\mathbf{u}$ into length-$2n$ binary string using $\Phi(0) = 00$, $\Phi(1) = 10$, $\Phi(2) = 01$.
- Second write
  - Pick a second-generation codeword $\mathbf{v}$ in $\mathcal{C}_3$.
  - Map length-n $\mathbf{v}$ into length-$2n$ binary string using $\Phi(0) = 00$, $\Phi(1) = 10$, $\Phi(2) = 01$.
  - Program each pair only once.

A construction of binary WOM codes

- We want to construct a binary three-write WOM over $2n$ cells.
- Let $\mathcal{C}_3$ be a ternary two-write WOM of length n.
- First write
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_3$.
  - Map length-n $\mathbf{u}$ into length-$2n$ binary string using $\Phi(0) = 00$, $\Phi(1) = 10$, $\Phi(2) = 01$.
- Second write
  - Pick a second-generation codeword $\mathbf{v}$ in $\mathcal{C}_3$.
  - Map length-n $\mathbf{v}$ into length-$2n$ binary string using $\Phi(0) = 00$, $\Phi(1) = 10$, $\Phi(2) = 01$.
  - Program each pair only once.
- Third write
  - Write length-n binary string in the unused cells (one per pair).

A construction of non-binary WOM codes

- We want to construct a non-binary two-write WOM with q levels, with $3|(q+1)$.

A construction of non-binary WOM codes

- We want to construct a non-binary two-write WOM with q levels, with $3|(q+1)$.
- Partition levels in groups of $L = (q+1)/3$

  first partition is $\{0, 1, \ldots, L-1\}$
  second partition is $\{L-1, L, \ldots, 2L-1\}$
  third partition is $\{2L-1, 2L, \ldots, 3L-1\}$

A construction of non-binary WOM codes

- We want to construct a non-binary two-write WOM with q levels, with $3|(q+1)$.
- Partition levels in groups of $L = (q+1)/3$

  first partition is $\{0, 1, \ldots, L-1\}$
  second partition is $\{L-1, L, \ldots, 2L-1\}$
  third partition is $\{2L-1, 2L, \ldots, 3L-1\}$
- Let $\mathcal{C}_2$ be a binary two-write WOM code of length n.
- First write:
  - Pick a message $\mathbf{m}$ in Z_3^n.
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $u_i = 0$ write m_i. If $u_i = 1$ write $m_i + L$.

A construction of non-binary WOM codes

- We want to construct a non-binary two-write WOM with q levels, with $3|(q+1)$.
- Partition levels in groups of $L = (q+1)/3$

  first partition is $\{0, 1, \ldots, L-1\}$
  second partition is $\{L-1, L, \ldots, 2L-1\}$
  third partition is $\{2L-1, 2L, \ldots, 3L-1\}$
- Let $\mathcal{C}_2$ be a binary two-write WOM code of length n.
- First write:
  - Pick a message $\mathbf{m}$ in Z_3^n.
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $u_i = 0$ write m_i. If $u_i = 1$ write $m_i + L$.
- Second write:
  - Pick a message $\mathbf{s}$ in Z_3^n.
  - Pick a second-generation codeword $\mathbf{v}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $v_i = 0$ write $s_i + (L-1)$. If $v_i = 1$ write $s_i + (2L-1)$.

A construction of non-binary WOM codes

- We want to construct a non-binary two-write WOM with q levels, with $3|(q+1)$.
- Partition levels in groups of $L = (q+1)/3$

  first partition is $\{0, 1, \ldots, L-1\}$
  second partition is $\{L-1, L, \ldots, 2L-1\}$
  third partition is $\{2L-1, 2L, \ldots, 3L-1\}$
- Let $\mathcal{C}_2$ be a binary two-write WOM code of length n.
- First write: first and second partition only
  - Pick a message $\mathbf{m}$ in Z_3^n.
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $u_i = 0$ write m_i. If $u_i = 1$ write $m_i + L$.
- Second write:
  - Pick a message $\mathbf{s}$ in Z_3^n.
  - Pick a second-generation codeword $\mathbf{v}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $v_i = 0$ write $s_i + (L-1)$. If $v_i = 1$ write $s_i + (2L-1)$.

A construction of non-binary WOM codes

- We want to construct a non-binary two-write WOM with q levels, with $3|(q+1)$.
- Partition levels in groups of $L = (q+1)/3$

  first partition is $\{0, 1, \ldots, L-1\}$
  second partition is $\{L-1, L, \ldots, 2L-1\}$
  third partition is $\{2L-1, 2L, \ldots, 3L-1\}$
- Let $\mathcal{C}_2$ be a binary two-write WOM code of length n.
- First write: first and second partition only
  - Pick a message $\mathbf{m}$ in Z_3^n.
  - Pick a first-generation codeword $\mathbf{u}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $u_i = 0$ write m_i. If $u_i = 1$ write $m_i + L$.
- Second write: second and third partition only
  - Pick a message $\mathbf{s}$ in Z_3^n.
  - Pick a second-generation codeword $\mathbf{v}$ in $\mathcal{C}_2$.
  - For each i, $1 \leq i \leq n$: If $v_i = 0$ write $s_i + (L-1)$. If $v_i = 1$ write $s_i + (2L-1)$.

An example using Rivest and Shamir write-twice code for $q = 2$ and $n = 8$

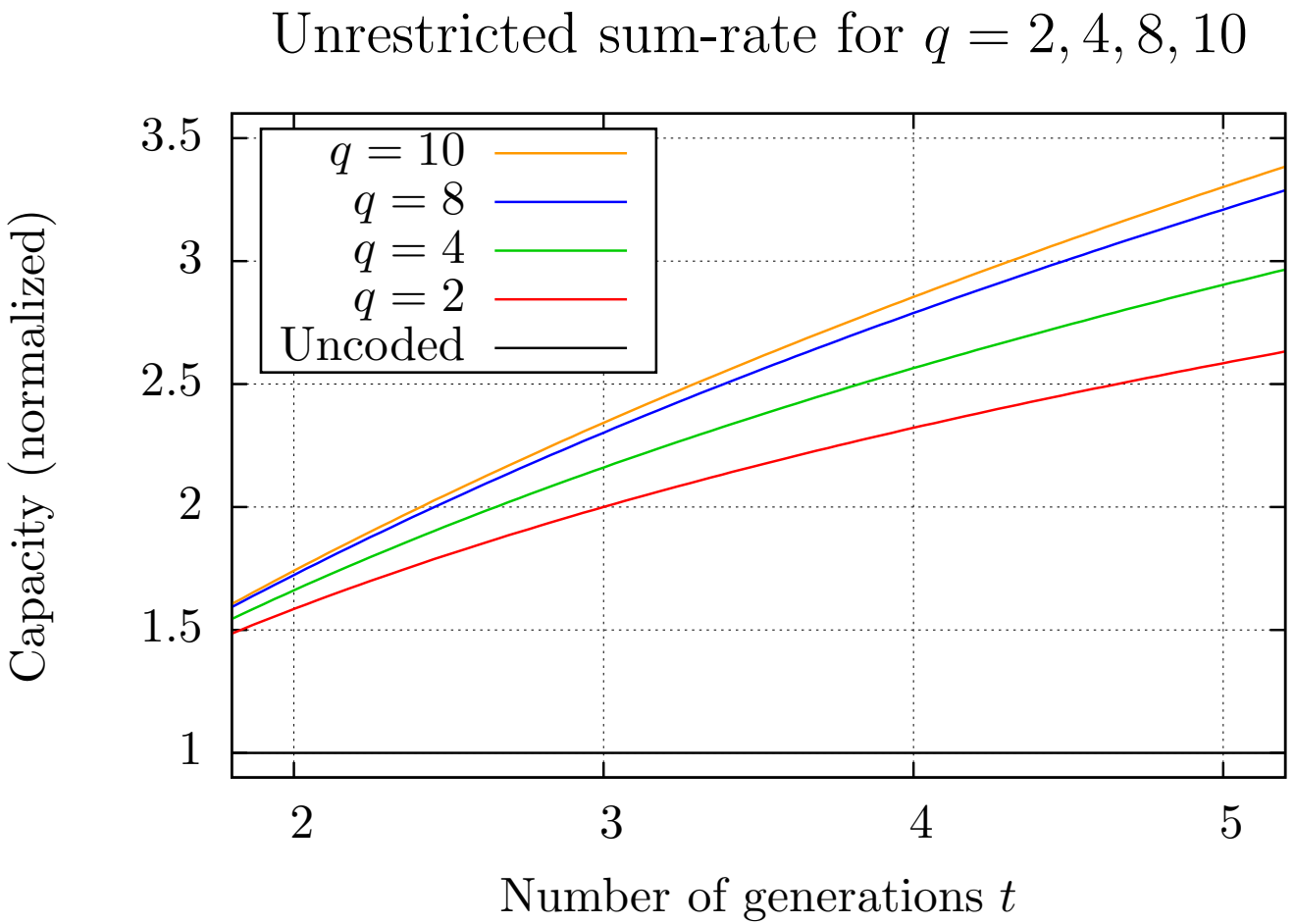
Rivest-Shamir WOM code

Information	First Generation	Second Generation
00	000	111
01	001	110
10	010	101
11	100	011

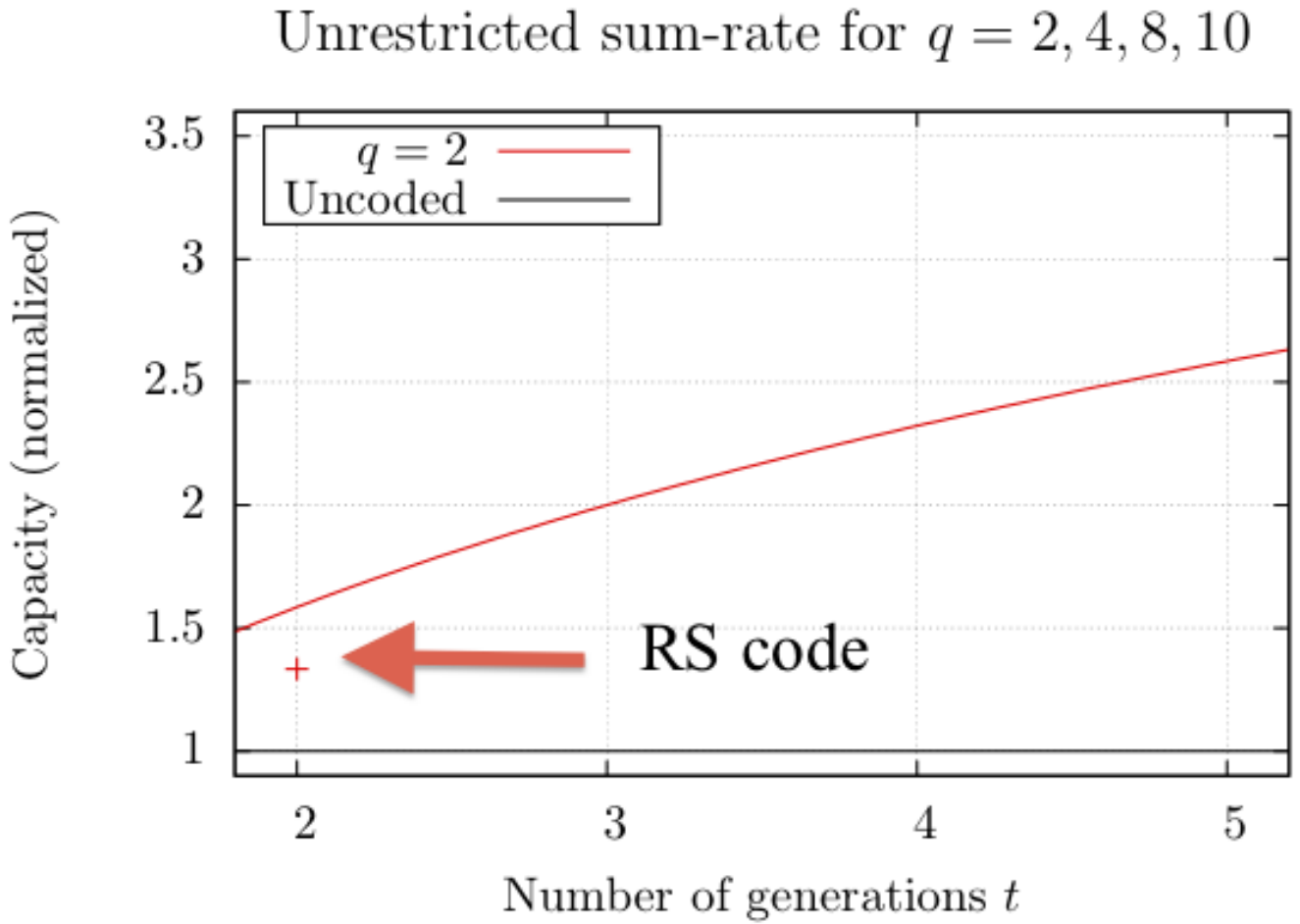
Non-binary WOM code

Write no.	Information	RS code + info	Encoded values
1	(0,1),(0,1,2)	(001),(012)	(0,1,5)
2	(0,0),(2,1,2)	(111),(212)	(7,6,7)

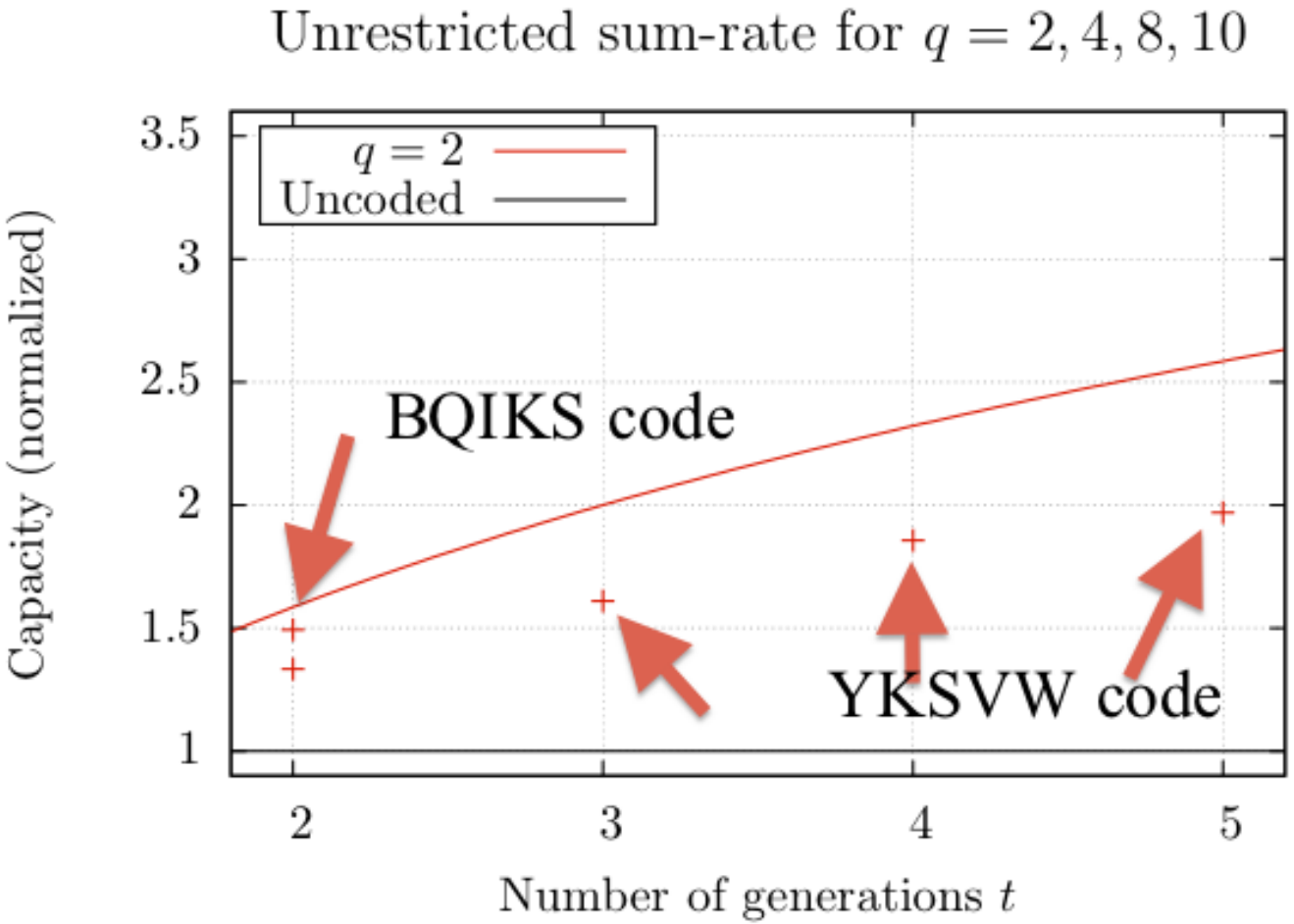
Sum rate of short WOM codes



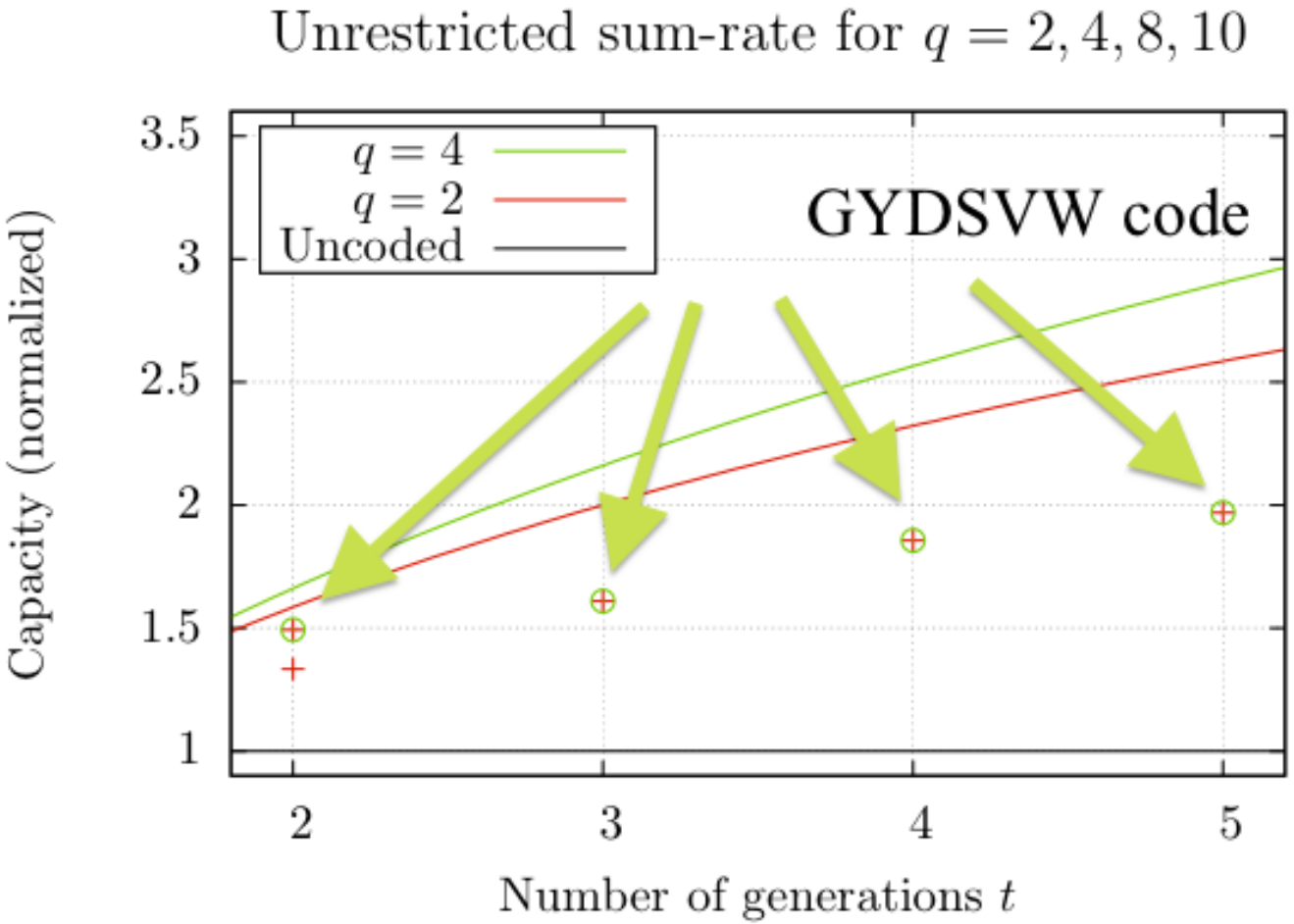
Sum rate of short WOM codes



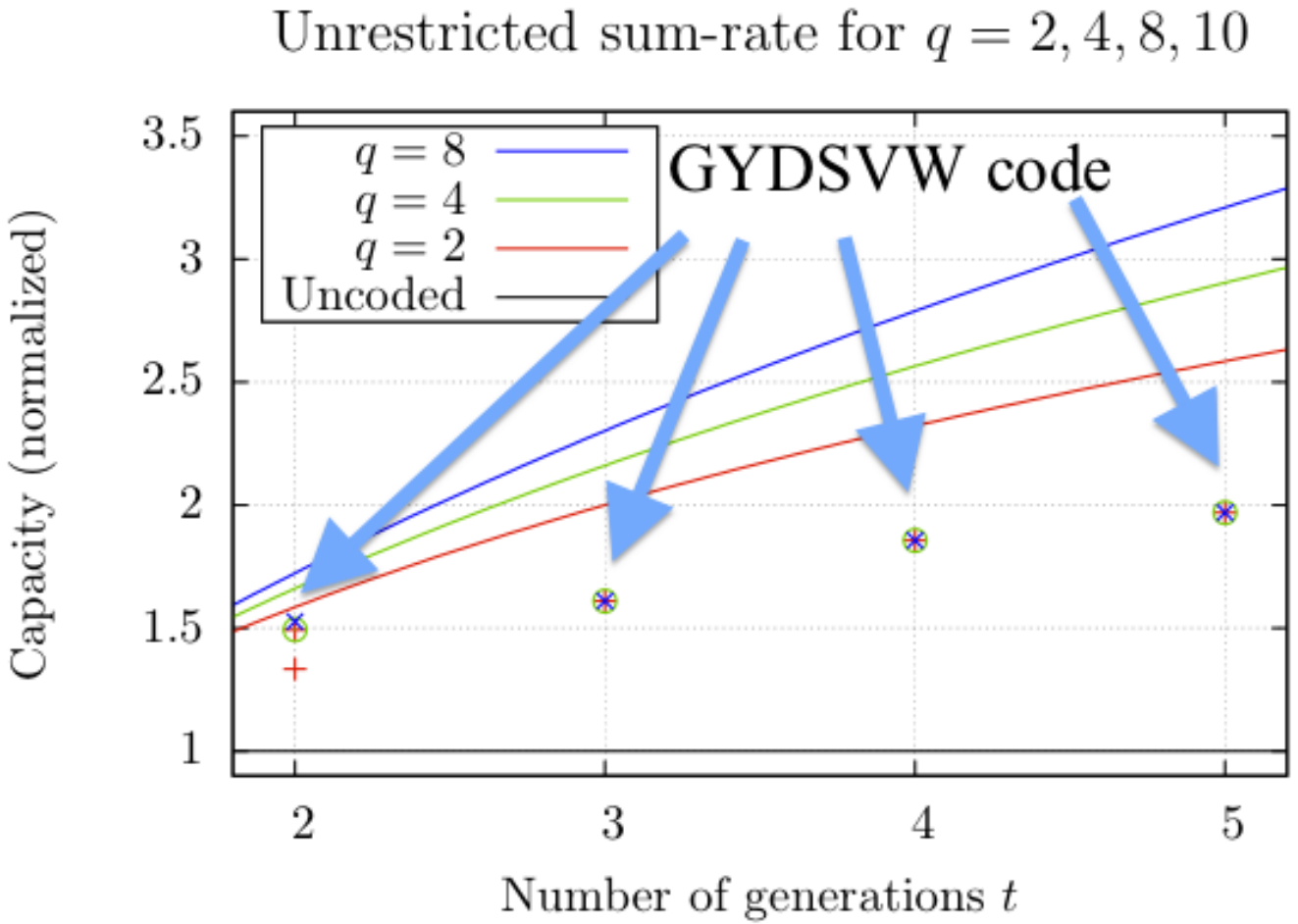
Sum rate of short WOM codes



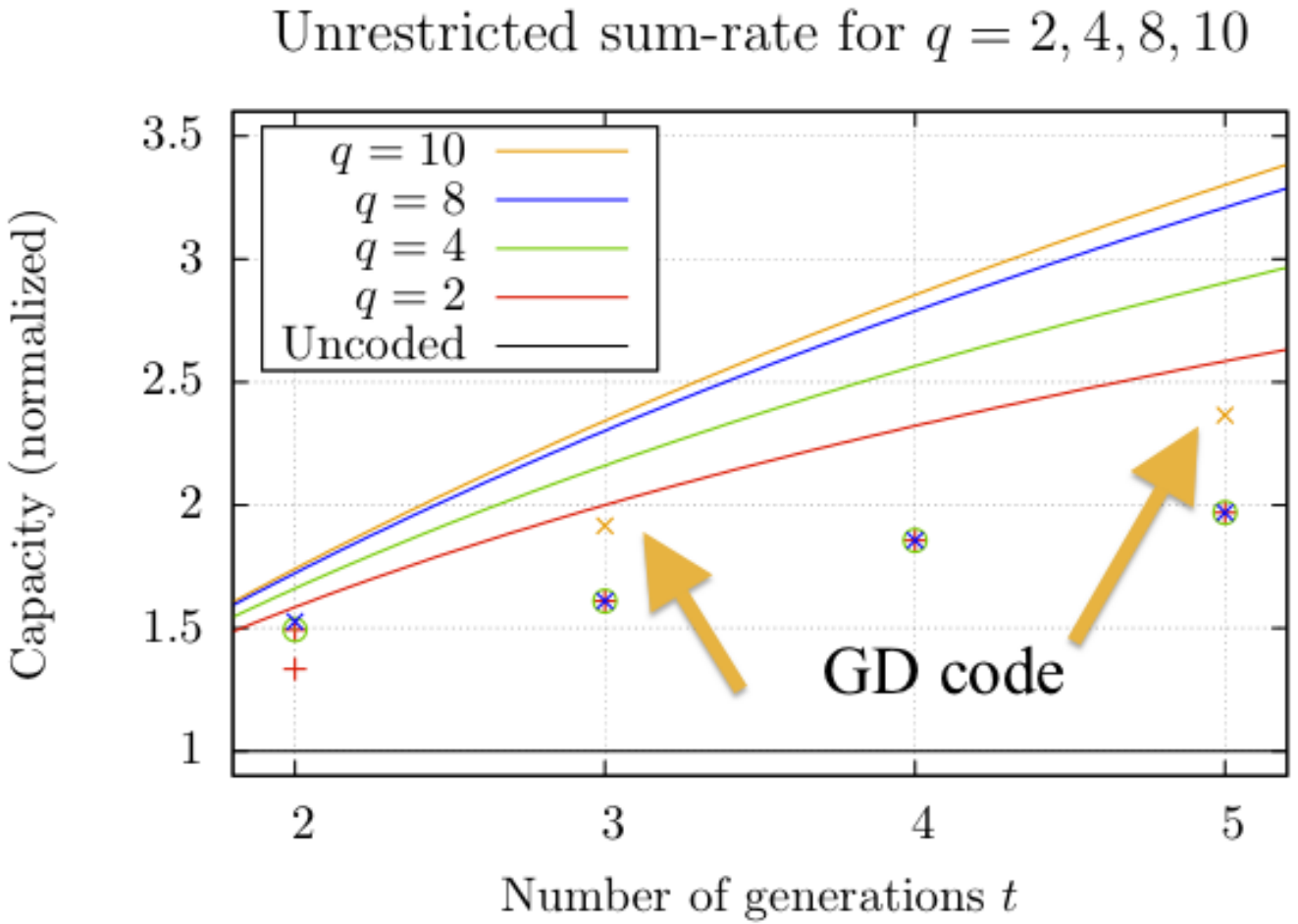
Sum rate of short WOM codes



Sum rate of short WOM codes

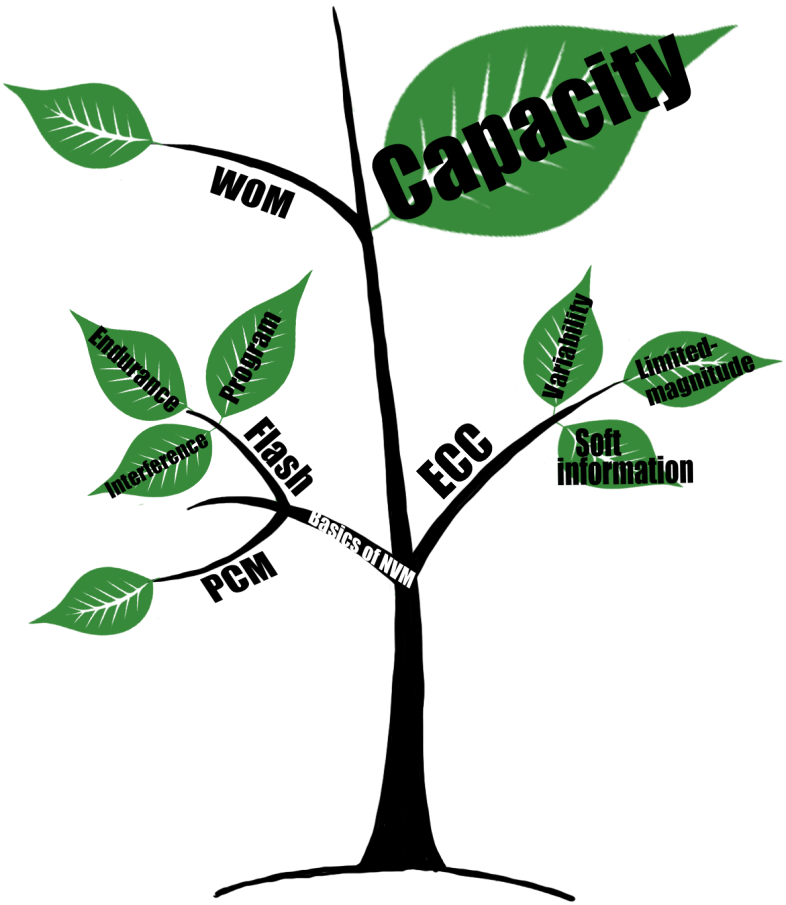


Sum rate of short WOM codes



Facets of WOM coding

- Capacity approaching constructions
  - A. Shpilka, "Capacity achieving multiwrite WOM codes", T-IT, 2014.
  - D. Burshtein and A. Strugatski, "Polar write once memory codes", T-IT 2013.
- WOM with error correction/detection capabilities
  - Q. Huang, S. Lin, and K. A. S. Abdel Ghaffar, "Error correcting codes for flash coding", T-IT, 2011.
  - E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple error correcting WOM codes", T-IT, 2012.
  - A. Jiang, Y. Li, E. En Gad, M. Langberg and J. Bruck, "Joint rewriting and error correction in write-once memories", ISIT, 2013.
- Extensions
  - N. Bitouze, A. Graell i Amat, and E. Rosnes, "Using short synchronous WOM codes to make WOM codes decodable", TCOM, 2014.
  - L. Wang, M. Qin, E. Yaakobi, Y. H. Kim, and P. H. Siegel, "WOM with retained messages", ISIT, 2012.
  - Y. Cassuto and E. Yaakobi, "Short q-ary WOM codes with hot/cold write differentiation", ISIT, 2012.


Capacity
WOM
Flash
Endurance
Program
Interference
PCM
Basics of NVM
ECC
Variability
Limited-
magnitude
Soft
information

Motivation: Noisy writes in PCM

- The write process in PCM is inherently inexact

Motivation: Noisy writes in PCM

- The write process in PCM is inherently inexact
- The data can be written iteratively, by checking the content between successive writes

Motivation: Noisy writes in PCM

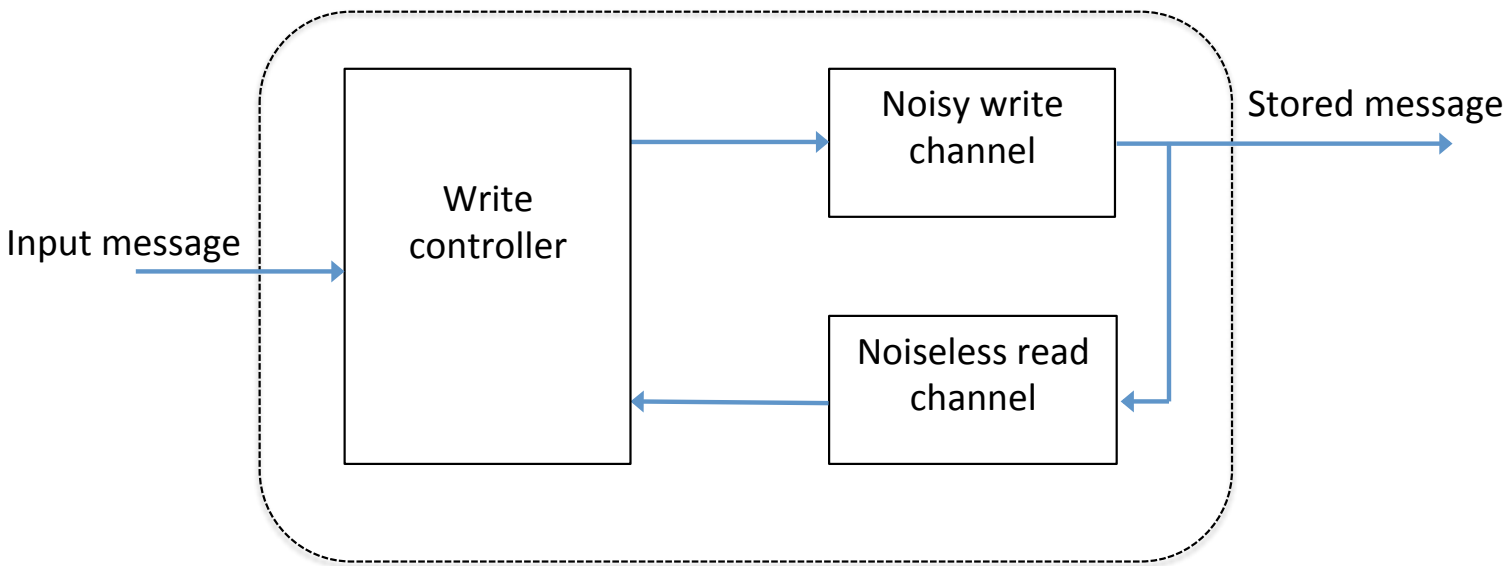
- The write process in PCM is inherently inexact
- The data can be written iteratively, by checking the content between successive writes
- Correctly written cells need not be further updated (cells in PCM can be accessed individually)

Motivation: Noisy writes in PCM

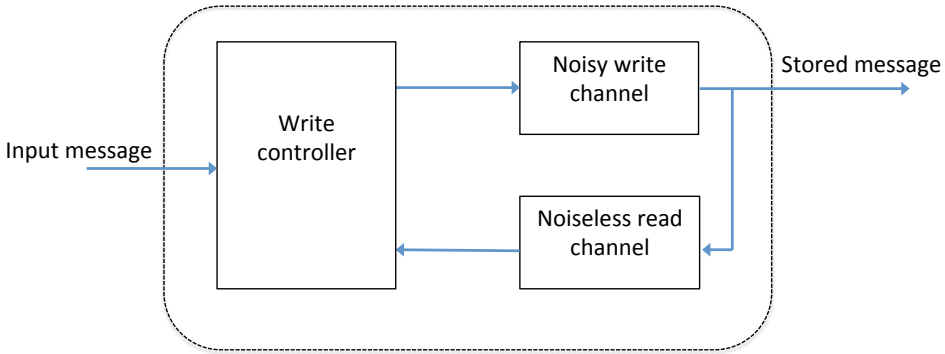
- The write process in PCM is inherently inexact
- The data can be written iteratively, by checking the content between successive writes
- Correctly written cells need not be further updated (cells in PCM can be accessed individually)
- The write process terminates when all cells are sufficiently well programmed

Motivation: Noisy writes in PCM

- The write process in PCM is inherently inexact
- The data can be written iteratively, by checking the content between successive writes
- Correctly written cells need not be further updated (cells in PCM can be accessed individually)
- The write process terminates when all cells are sufficiently well programmed



Rewritable channel for PCM



- Average rewriting cost is κ

Theorem

Capacity is upper-bounded by $\log(\Gamma\kappa)$

- Parameter Γ depends on noise characteristics.
- Exact expressions known in certain cases
- Several extensions available

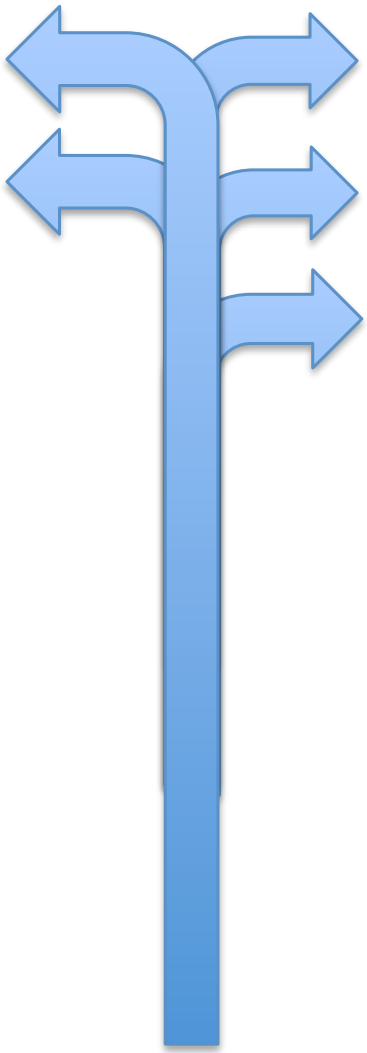
Further Reading – Early Results

- R. L. Rivest and A. Shamir, "How to reuse a "write-once" memory", I&C, 1982.
- J. K. Wolf, A. D. Wyner, J. Ziv, and J. Körner, "Coding for write-once memory", AT&T, 1984.
- C. Heegard, "On the capacity of permanent memory", T-IT, 1985.
- G. D. Cohen, P. Godlewski, and F. Merkx, "Linear binary code for write-once memories", T-IT, 1986.
- 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", T-IT, 1999.

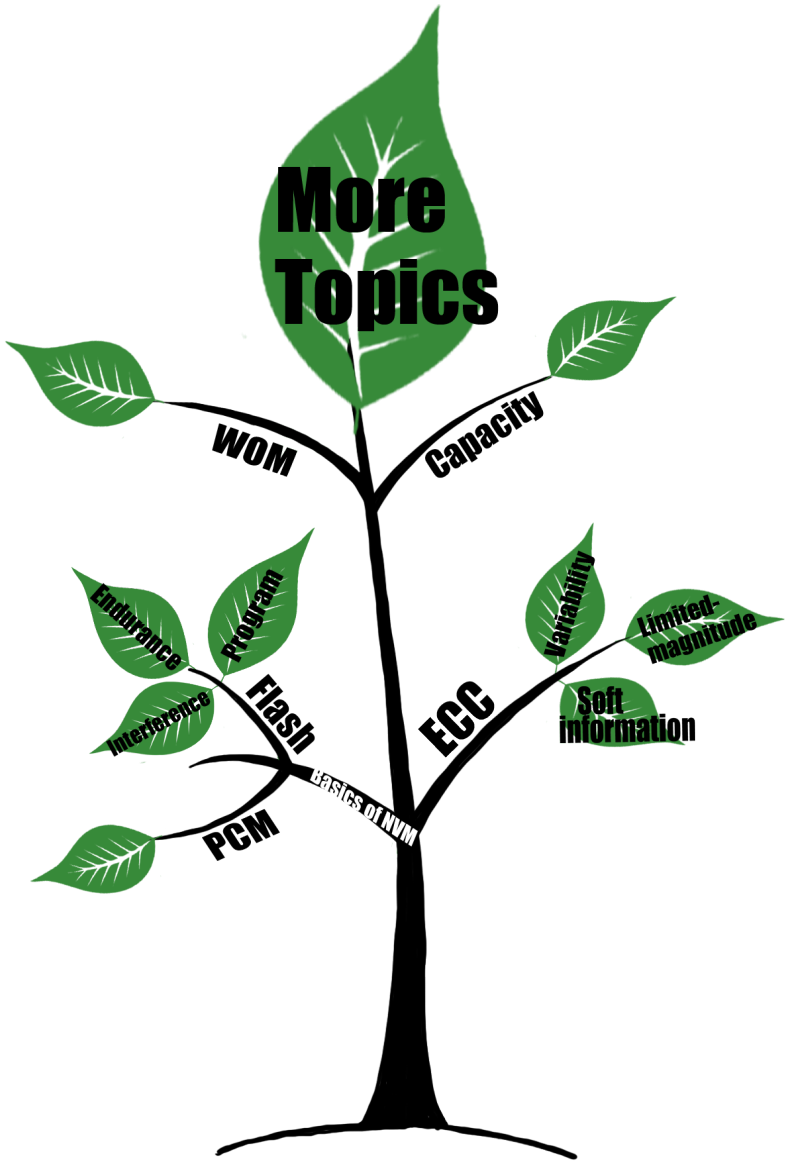
Further Reading – Recent Results

- E. Yaakobi, S. Kayser, P. Siegel, A. Vardy, and J. K. Wolf, "Codes for write-once memories", T-IT, 2012.
- A. Bhatia, M. Qin, A. Iyengar, B. Kurkoski, and P. Siegel, "Lattice-based WOM codes for multilevel flash memories", JSAC, 2014.
- R. Gabrys and L. Dolecek, "Constructions of nonbinary WOM codes for multilevel flash memories", preprint, 2014.
- R. Gabrys, E. Yaakobi, L. Dolecek, P. Siegel, A. Vardy, and J. K. Wolf, "Non-binary WOM codes for multilevel flash memories", ITW, 2011.
- L. A. Lastras-Montaño, M. Franceschini, T. Mittelholzer, and M. Sharma, "Rewritable storage channels", ISITA, 2008.
- R. Venkataramanan, S. Tatikonda, L. Lastras-Montaño, M. Franceschini, "Rewritable storage channels with hidden state", JSAC, 2014.

Research problems on WOM and rewriting



1. Coding for rewritable channels with read feedback
2. Extensions to other related models (WAM, floating codes etc).
3. Connections with interference channels (e.g., dirty paper coding)
4. Establishment of the complete capacity region/beyond zero-error capacity.
5. Development of new tools for high-rate codes spanning more than a couple of cells (e.g., lattices, posets)


More Topics
WOM
Capacity
Endurance
Program
Interference
Flash
PCM
Basics of NVM
ECC
Variability
Limited-magnitude
Soft information

Further topics – beyond coding

- Statistical and signal processing methods for voltage threshold modeling and characterization
  - Parameter estimation and hypothesis testing
  - Study of time-varying stochastic processes

Further topics – beyond coding

- Statistical and signal processing methods for voltage threshold modeling and characterization
  - Parameter estimation and hypothesis testing
  - Study of time-varying stochastic processes
- Communication methods for Flash channels
  - Design of optimal detectors under ICI
  - Use of equalization and signal shaping

Further topics – beyond coding

- Statistical and signal processing methods for voltage threshold modeling and characterization
  - Parameter estimation and hypothesis testing
  - Study of time-varying stochastic processes
- Communication methods for Flash channels
  - Design of optimal detectors under ICI
  - Use of equalization and signal shaping
- Capacity calculations
  - Analysis of quantized channels
  - Non asymptotic performance analysis

Further topics – beyond coding

- Statistical and signal processing methods for voltage threshold modeling and characterization
  - Parameter estimation and hypothesis testing
  - Study of time-varying stochastic processes
- Communication methods for Flash channels
  - Design of optimal detectors under ICI
  - Use of equalization and signal shaping
- Capacity calculations
  - Analysis of quantized channels
  - Non asymptotic performance analysis
- Probabilistic methods and complexity
  - Implementation complexity evaluation and prototyping
  - Average vs. worst case performance analysis and design

Further topics – beyond coding

- Statistical and signal processing methods for voltage threshold modeling and characterization
  - Parameter estimation and hypothesis testing
  - Study of time-varying stochastic processes
- Communication methods for Flash channels
  - Design of optimal detectors under ICI
  - Use of equalization and signal shaping
- Capacity calculations
  - Analysis of quantized channels
  - Non asymptotic performance analysis
- Probabilistic methods and complexity
  - Implementation complexity evaluation and prototyping
  - Average vs. worst case performance analysis and design
- Applications of interference alignment and interference channels

TIME FOR A BREAK!