Overview of Coding Methods for Flash Memories

Brian M. Kurkoski

kurkoski@ice.uec.ac.jp

Dept. of Information and Communications Engineering
University of Electro-Communications

Tokyo, Japan

フラッシュメモリ符号化に関するワークショップ

名古屋工業大学

http://flashworkshop.org/

2010 April 3

Outline

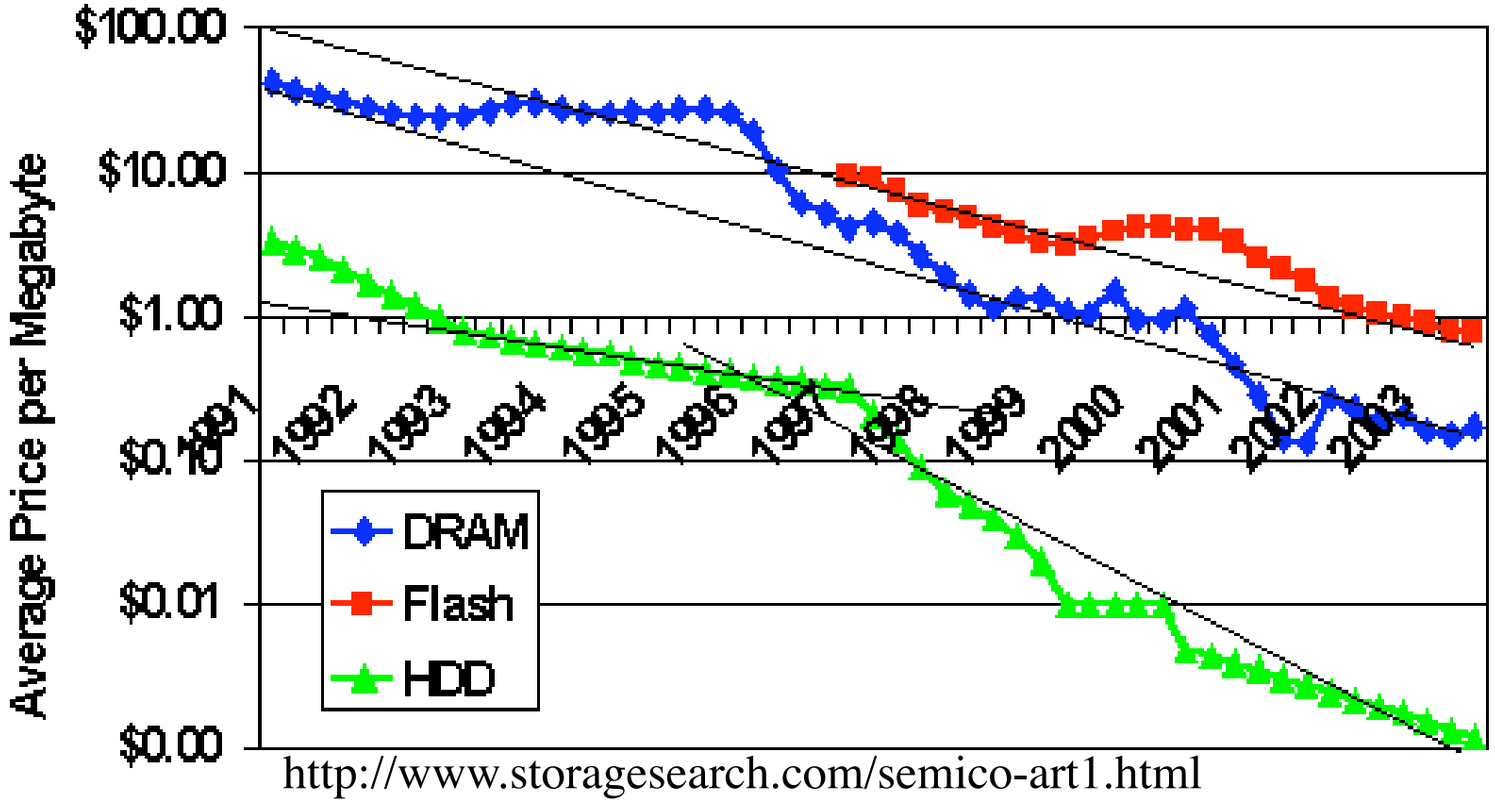
- This talk is on coding for flash memories.
- Concentrate on codes for re-writing memories

1. Overview of flash memory: benefits and problems
2. Codes for rewriting memories
   - Codes for binary q=2 memories
   - Codes for non-binary q>2 memories
3. Traditional error-correction for flash

Flash Memory

Flash is a semiconductor memory whose price has been rapidly decreasing. History:

- 1980 NOR Flash invented by Fujio Masuoka at Toshiba
- 1986 NAND Flash also invented by Masuoka
- 1988 Intel introduces commercial NOR Flash (PC BIOS, etc.)
- 1998 Early MP3 Player (32 MB, Korean SaeHan Information Systems)
- 2000 First USB Flash drive (8MB IBM)



http://www.storagesearch.com/semico-art1.html

Flash vs. Hard Disks as a Storage Medium

Compared to hard disks, flash has advantages:

- Mechanically durable
- very fast random reading and writing (~100 times hard disks)
- fast sequential reads
- small memories are feasible (2 GB for ¥900)
- lower power

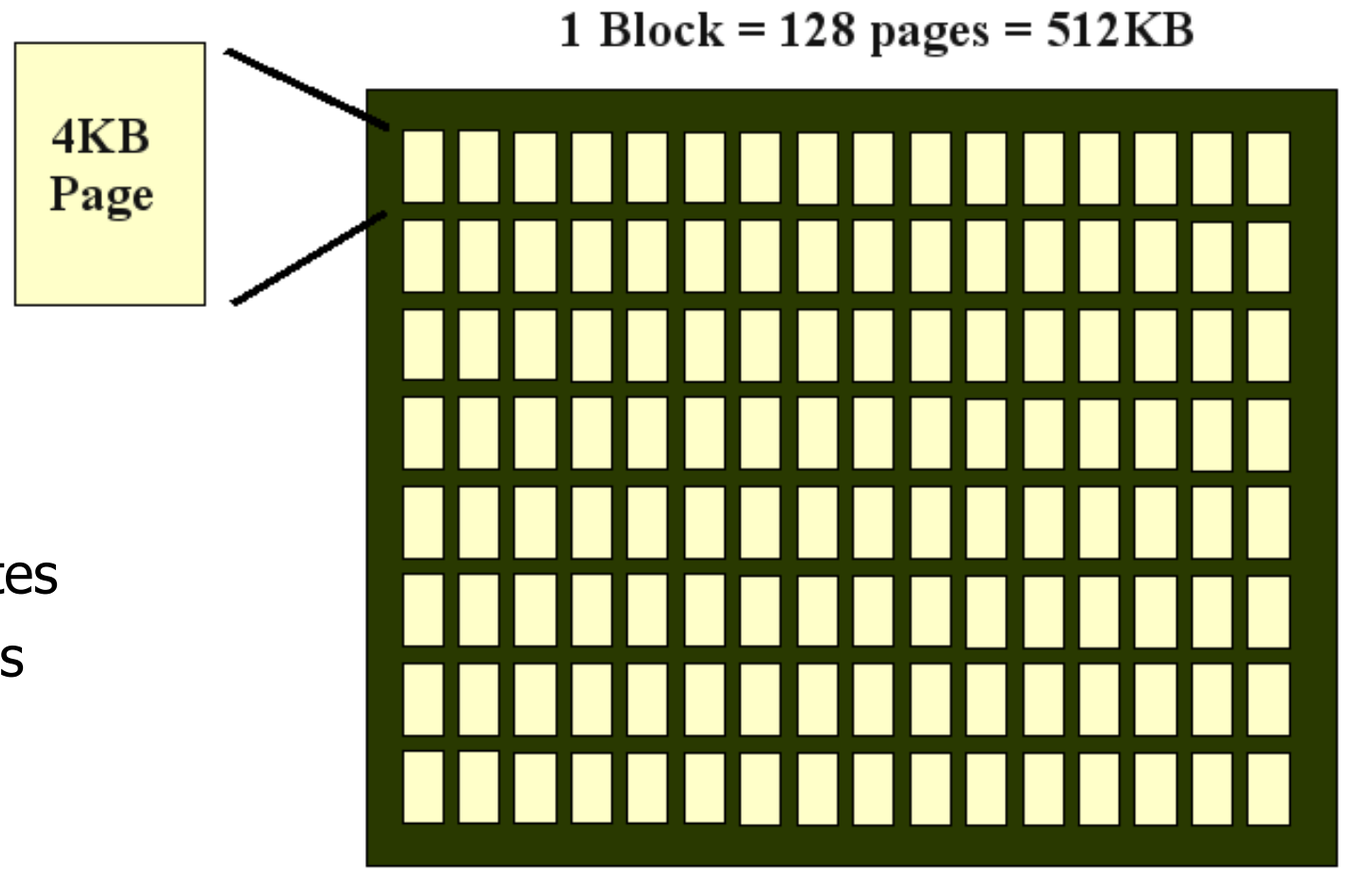
Flash disadvantages:

- performance decreases as SSD is used
- high cost per megabyte

But the cost of flash SSDs are almost reasonable

About Flash

- NOR Flash
  - Relatively low density
  - Small sizes blocks — "random" access is possible
  - Used to storing computer programs
- NAND Flash
  - Higher density
  - Has very large blocks
  - "Sequential" access
  - Widely used: cameras to SSDs



http://www.linux-mag.com/cache/7590/1.html

- NAND Flash is arranged:
  - one **page** consists of 512-4096 bytes
  - one **block** consists of 32-128 pages
  - one **plane** consists of 1024 blocks

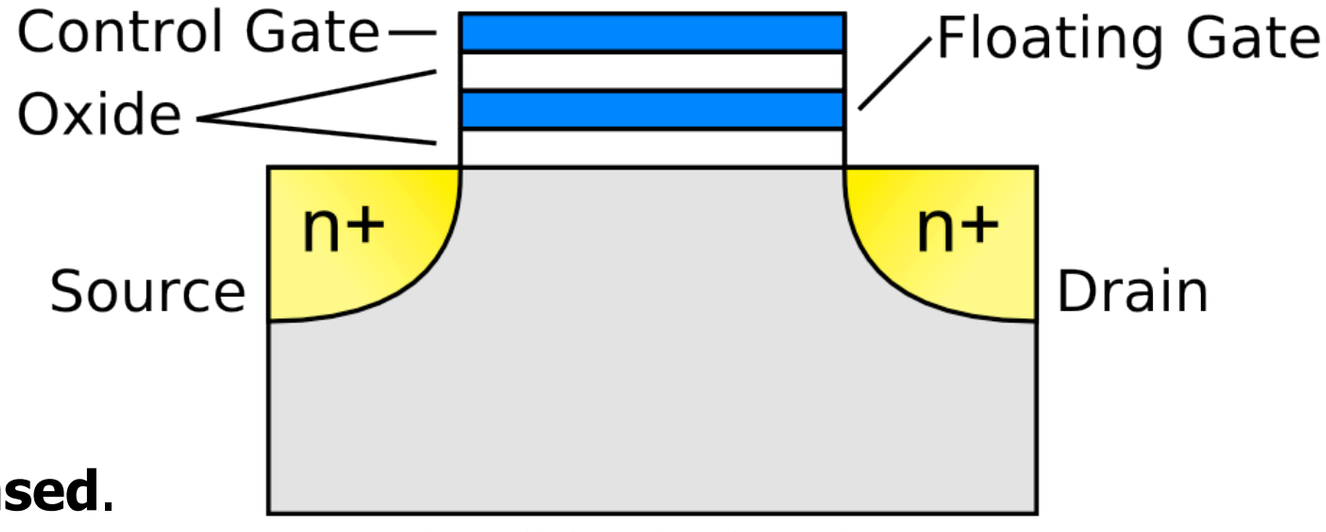
About Flash: Charge is easily added

- In flash memories, charge is stored on a "floating gate", and read as a voltage.
- Two very important things about flash:

  1. Charge can easily be increased, but can only be decreased by an erasure operation. **Only whole blocks of ~512 KB can be erased**.

  2. Each block has a limited number of erase cycles it can handle. After 10,000 - 100,000 erasures, the block cannot be reliably be used. Also, erasures are slow.

- Thus, erasures should be avoided.



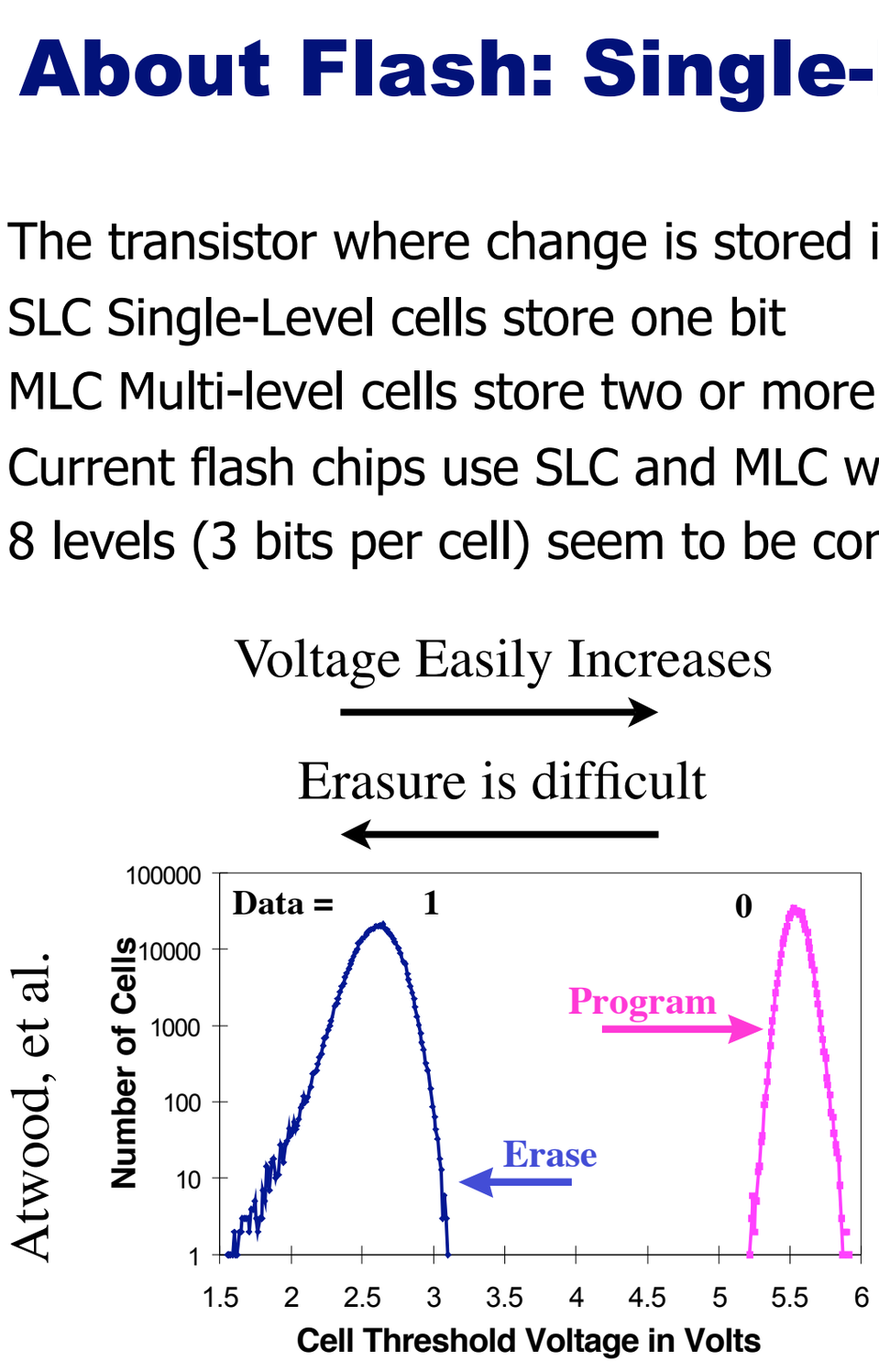
About Flash: Single-Level and Multi-Level

- The transistor where change is stored is called a "cell"
- SLC Single-Level cells store one bit
- MLC Multi-level cells store two or more bits
- Current flash chips use SLC and MLC with 4 levels (2 bits per cell)
- 8 levels (3 bits per cell) seem to be coming. Proposals as high as 256 levels.

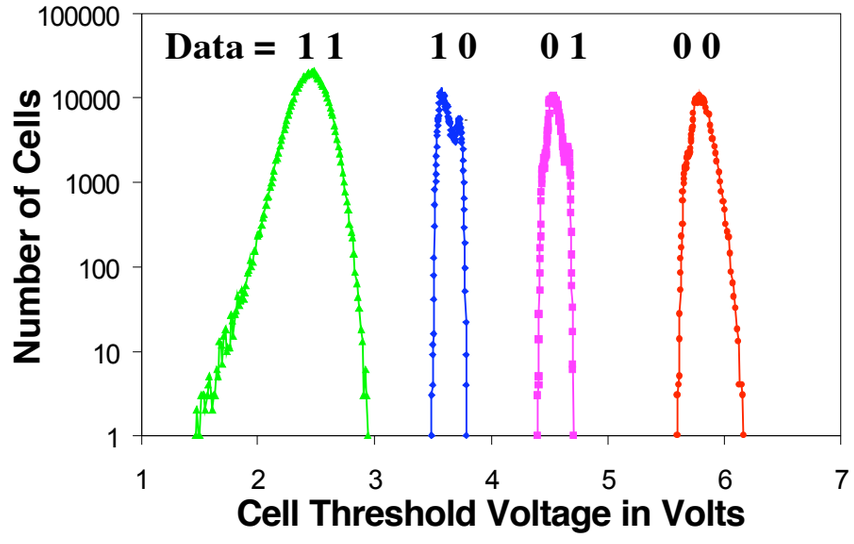
Voltage Easily Increases →

← Erasure is difficult

"Write Asymmetry"



SLC



MLC (2 bits)

Current Approaches

Because erasures shorten the longevity (長寿命化) of flash memory:

- Current solution: flash translation layer (FTL) and wear leveling
- Computer science research: "log file systems," garbage collection, etc.

Big Question

"Can coding theory improve the longevity and performance of flash memories?"

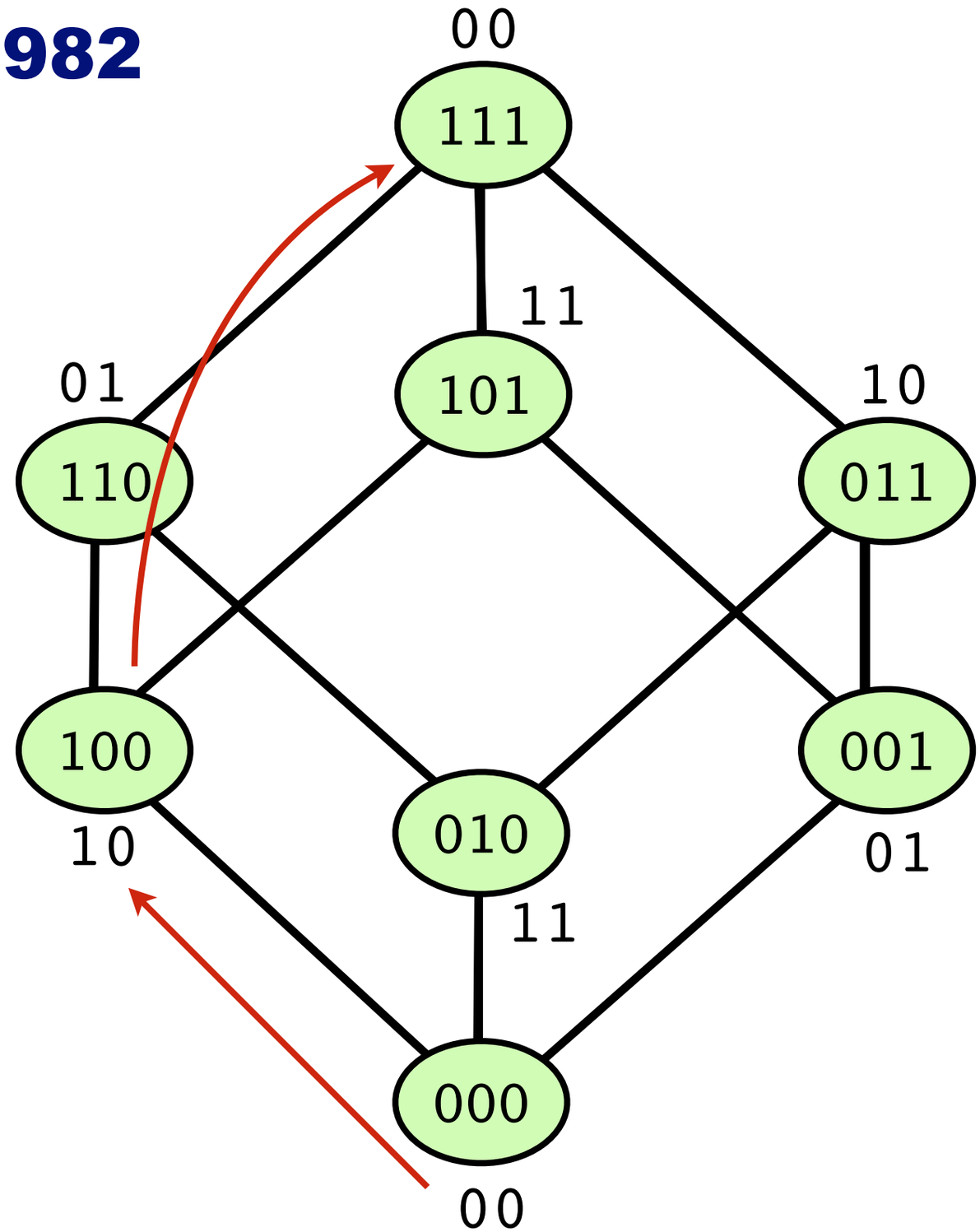
A Few More Problems

- SLC -> MLC Errors are possible, ECC is needed
- Write-verify cycle: programming is imprecise, and must avoid overshoot
- Read disturb
- Write disturb

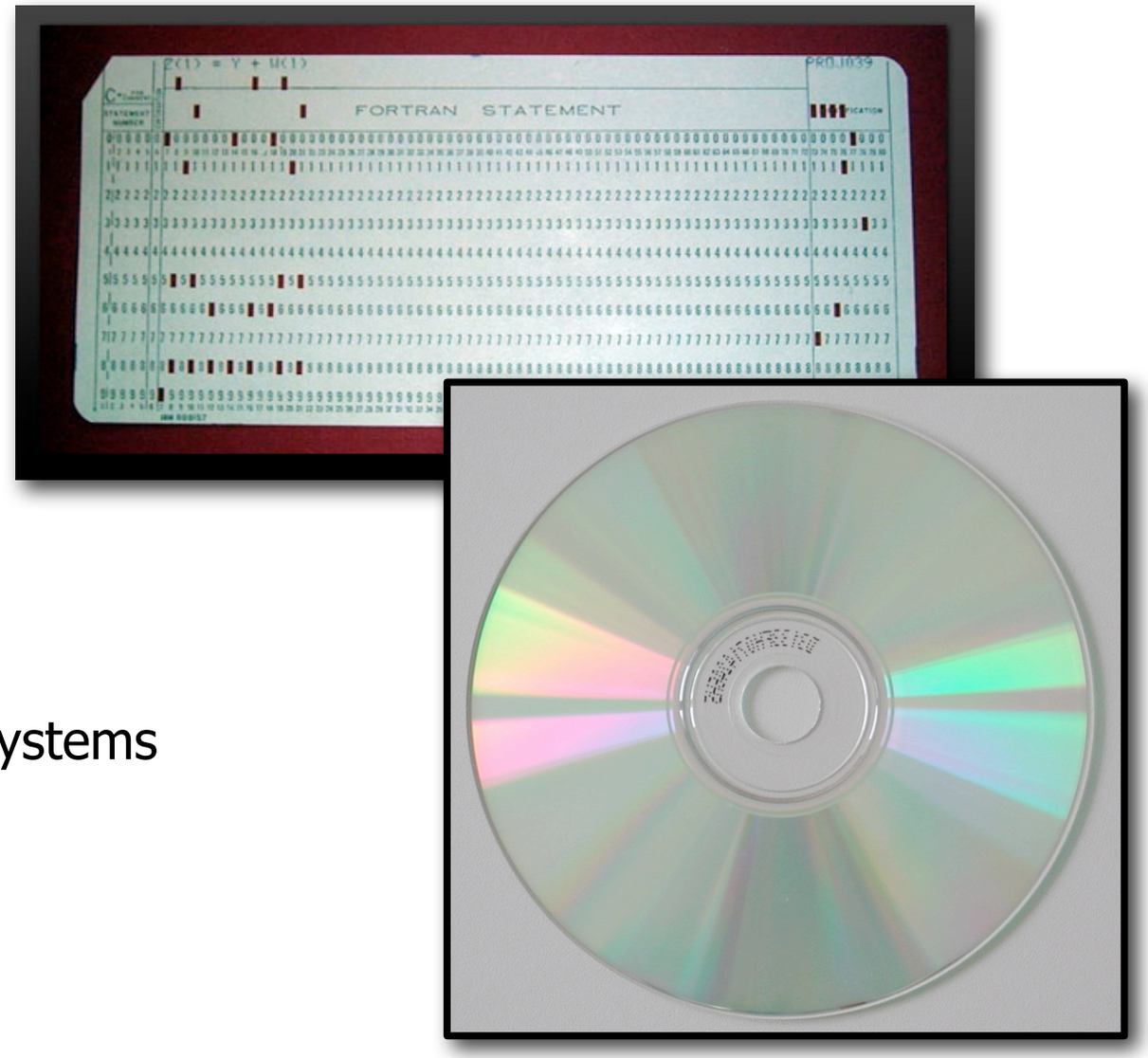
Rivest and Shamir: “How to Reuse a ‘Write Once’ Memory” 1982

Toy example:

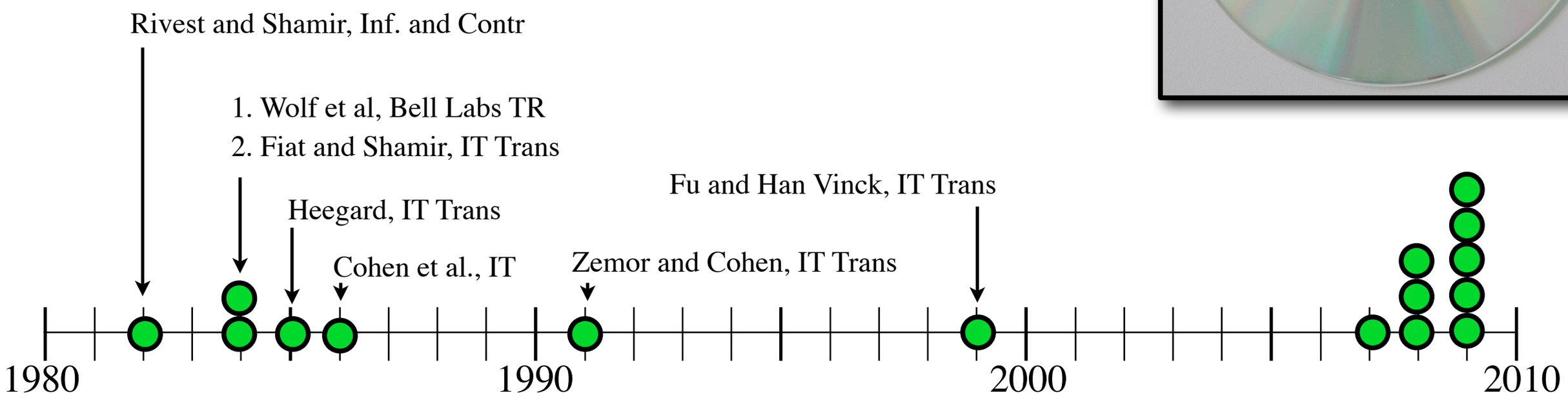
- 3 storage “cells”
  - Initially (0,0,0) state
  - 0 → 1 is allowed
  - 1 → 0 is not allowed
- Store 2 bits of information.
- Can write data 2 times:
  - first write can be any two bits
  - second write can be any two bits
- Example:
  - store 01, then
  - store 00
- Code rate is 2/3
- Guaranteed minimum of two writes



Literature Overview



- The problem of "asymmetrical writing" storage systems is not new!
  - write-once optical media: CD-R
  - PROM
  - punch cards
- Prior work mostly considered binary storage systems



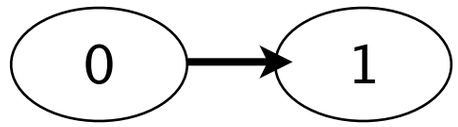
- This talk is about coding. However, there are also information theoretic results.

Outline of Rewriting Codes

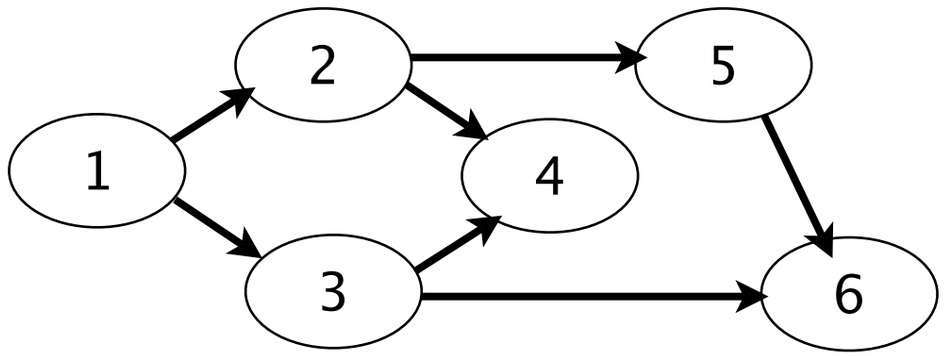
- Channel models, definitions and assumptions
- Codes for binary cells:
  - Rivest and Shamir bounds
  - linear code (based upon linear error-correcting codes) [Cohen et al, 1986]
- Codes for q-ary cells
  - JBB07 bounds
  - A low rate code

Channel Models for One "Cell"

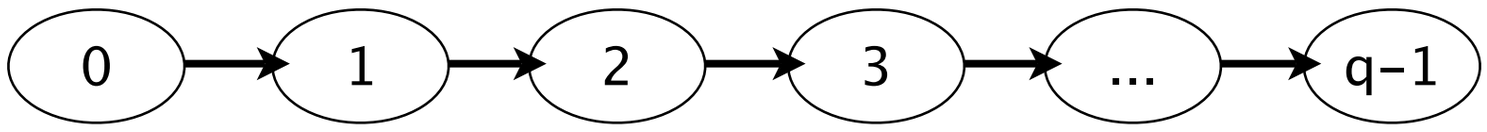
- "Write Once Memory" (Rivest and Shamir, 1982)



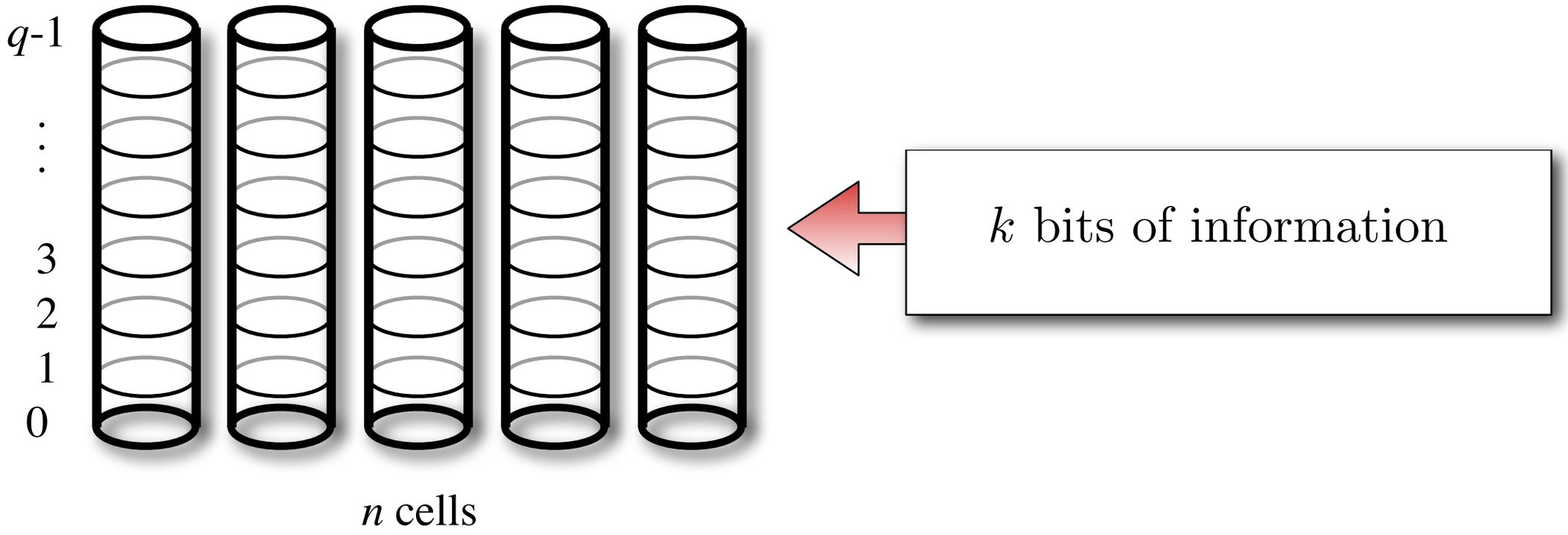
- Directed Acyclic Graph (DAG) Fiat and Shamir, 1984. Other capacity results.



- q-ary "Write Asymmetric Memory", Jiang et al 2007



Definitions and Assumptions



Floating code (or flash code)

- Encoding function:

$$\{\text{Information: One Variable}\} \times \{\text{Current Memory State}\} \longrightarrow \{\text{New Memory State}\} \cup \mathsf{E}$$

- Decoding function:

$$\{\text{Current Memory State}\} \longrightarrow \{k \text{ Information bits}\}$$

- Let t or T denote the *minimum* number of times information can be written, before erasure.

Definitions and Assumptions: Point 1, Writes

1. More than T or t writes may be possible.
2. Full memories must be erased before next write.

There are *two* perspectives on the number of writes:

- Word Writing (Rivest-Shamir).
  - k bits are written simultaneously.
  - A code allows *at least* T word writes, before the memory is "full".
- Bit Writing (Jiang et al).
  - Only 1 bit written at a time.
  - A code allows *at least* t bit writes, before the memory is "full".
  - 1 word write performed by k bit writes,

$$T = \frac{t}{k}$$

To make a fair comparison, choose *word writes* T as the metric.

- Consistent with block-oriented nature of storage devices

Definitions and Assumptions: Point 2, Rate

Natural questions:

- For a given n, k, T, q, does a floating code exist?
- What is the relationship between n block length, k info bits, T writes and q cell levels?

Fiat-Shamir (1984):

- for arbitrary DAG: NP Hard
- for a tree (including flash memory): polynomial time

Recent work has ignored the code rate (very low rates).

Define code rate as:

$$R \quad = \quad \frac{k}{n} \text{ bits per cell}$$

($R > 1$ is possible, for example: $q = 16 \Rightarrow R \leq 4$)

Previous papers:

$$R = \frac{n}{k}, \qquad R = \frac{kT}{n}$$

$$R = \frac{k}{n \log_2 q}$$

Will concentrate on this question:

- What is the relationship between T and R, for various n?

Codes for Binary Memory, q=2

Asymptotic bounds

Rivest & Shamir use $w\big(\langle 2^k \rangle^T\big)$ to mean "the length of an optimal code".

For $T = 2$, the rate as $k \to \infty$ is:

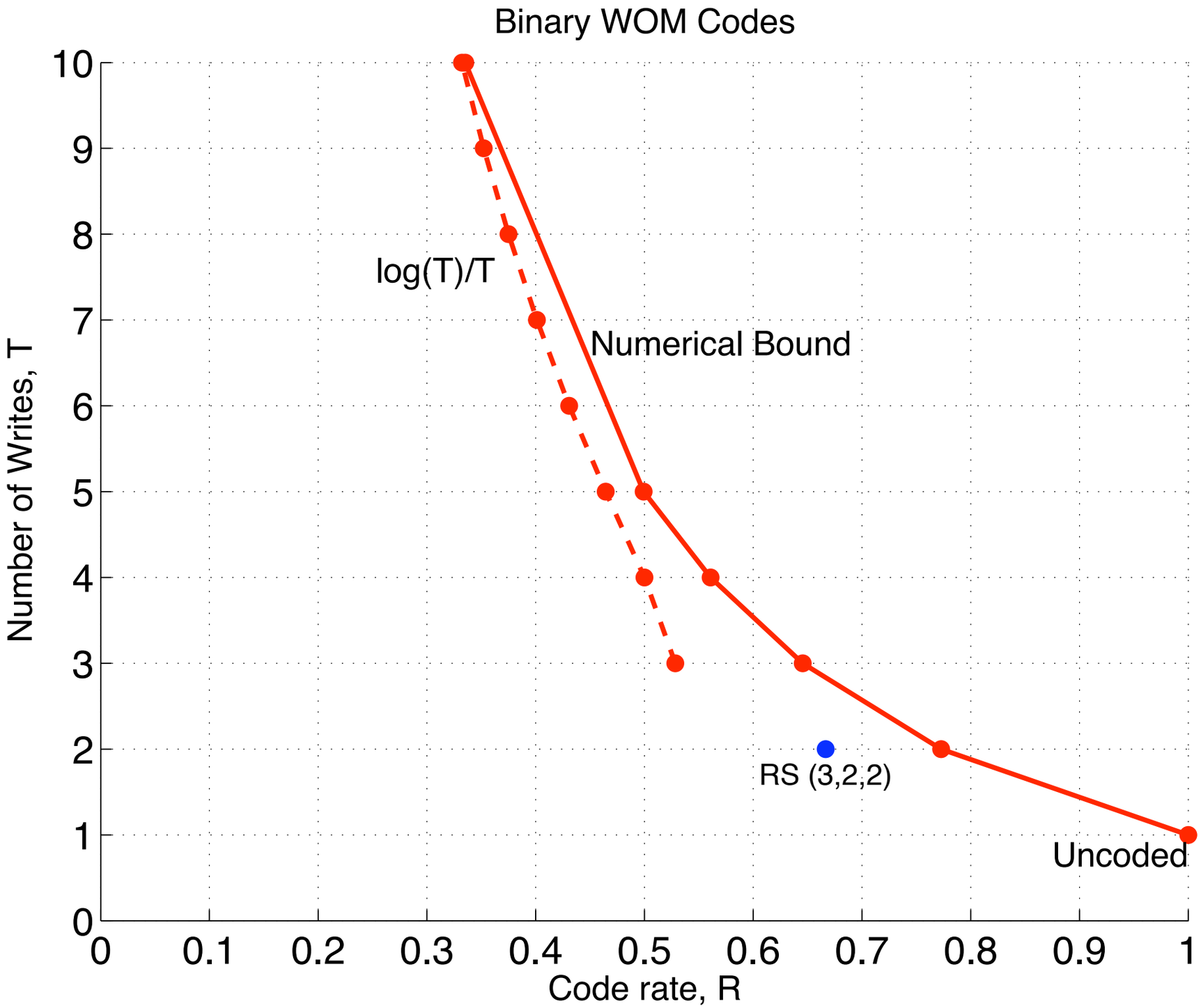
$$\text{"capacity" (or achievable rate)} \quad = \quad \lim_{k\to\infty} \frac{k}{w\big(\langle 2^k \rangle^2\big)} = 0.7729$$

Interestingly, 0.7729 is the solution to the equation $H(1-p) = p$.

For other values of T, estimate are given

T	"Capacity" Estimate	$\log(T)/T$
3	0.6456	0.5283
4	0.5609	0.5
5	0.4993	0.4644
10	0.3352	0.3322
20	0.2142	0.2161
50	0.1116	0.1129
100	0.0658	0.0664
200	0.0380	0.0382

Binary WOM Codes
Code Rate vs. Number of Writes



Simple Scheme to Write 1 Bit in n cells

As a "sanity check", consider a simple scheme for encoding $k = 1$ info bit.
Example $n = 5$. Stored sequence is:

$$(0,0,0,0,0) \to (1,0,0,0,0) \to (1,1,0,0,0) \to (1,1,1,0,0) \to (1,1,1,1,0) \to (1,1,1,1,1)$$

$$\downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow$$

$$0 \qquad 1 \qquad 0 \qquad 1 \qquad 0 \qquad 1$$

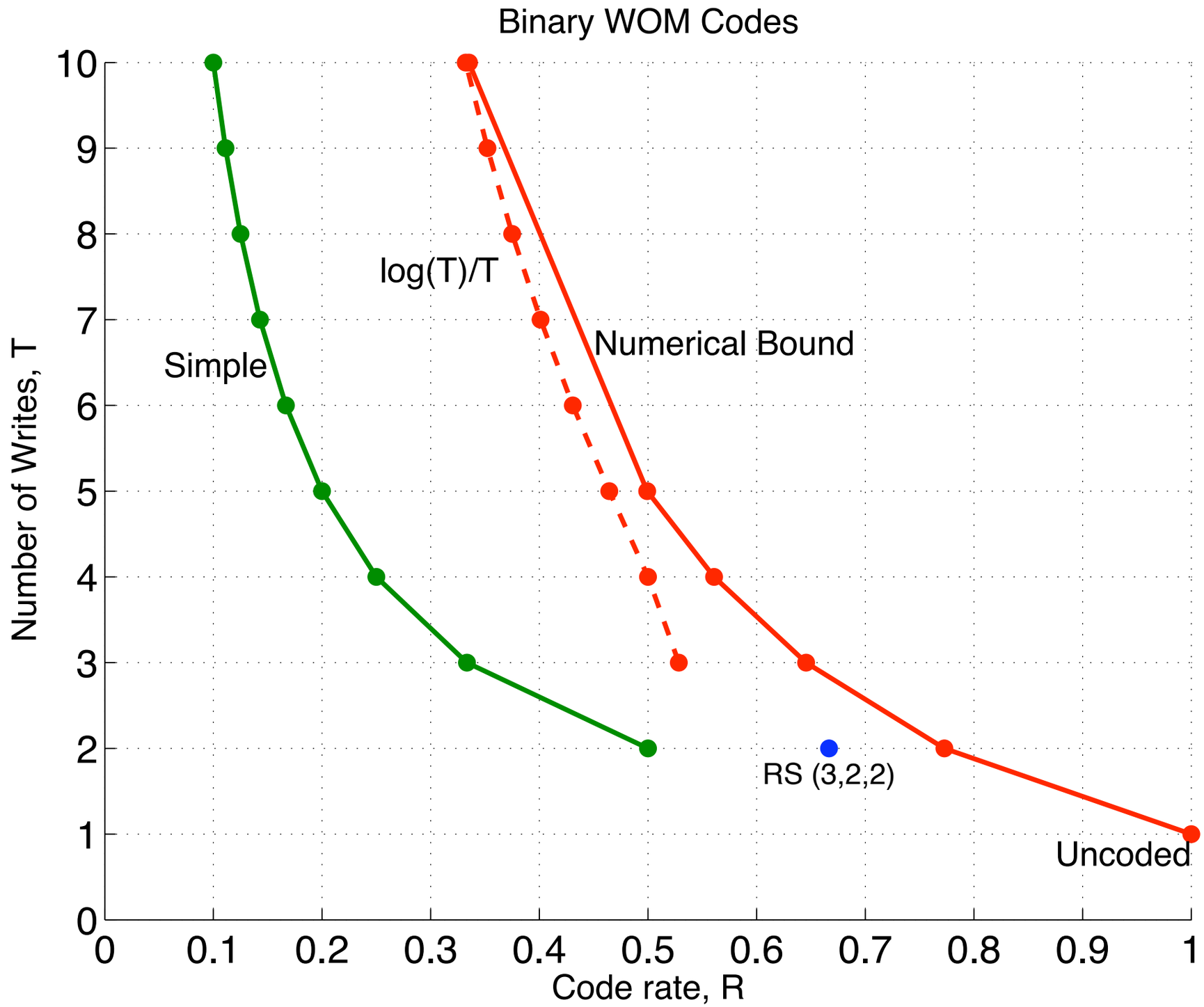
The information in each stage is the mod-2 sum of the stored sequence.
For any $n \geq 1$, this simple scheme has rate:

$$R = \frac{1}{T}$$

and allows for n writes:

$$T = n$$

Binary WOM Codes
Code Rate vs. Number of Writes



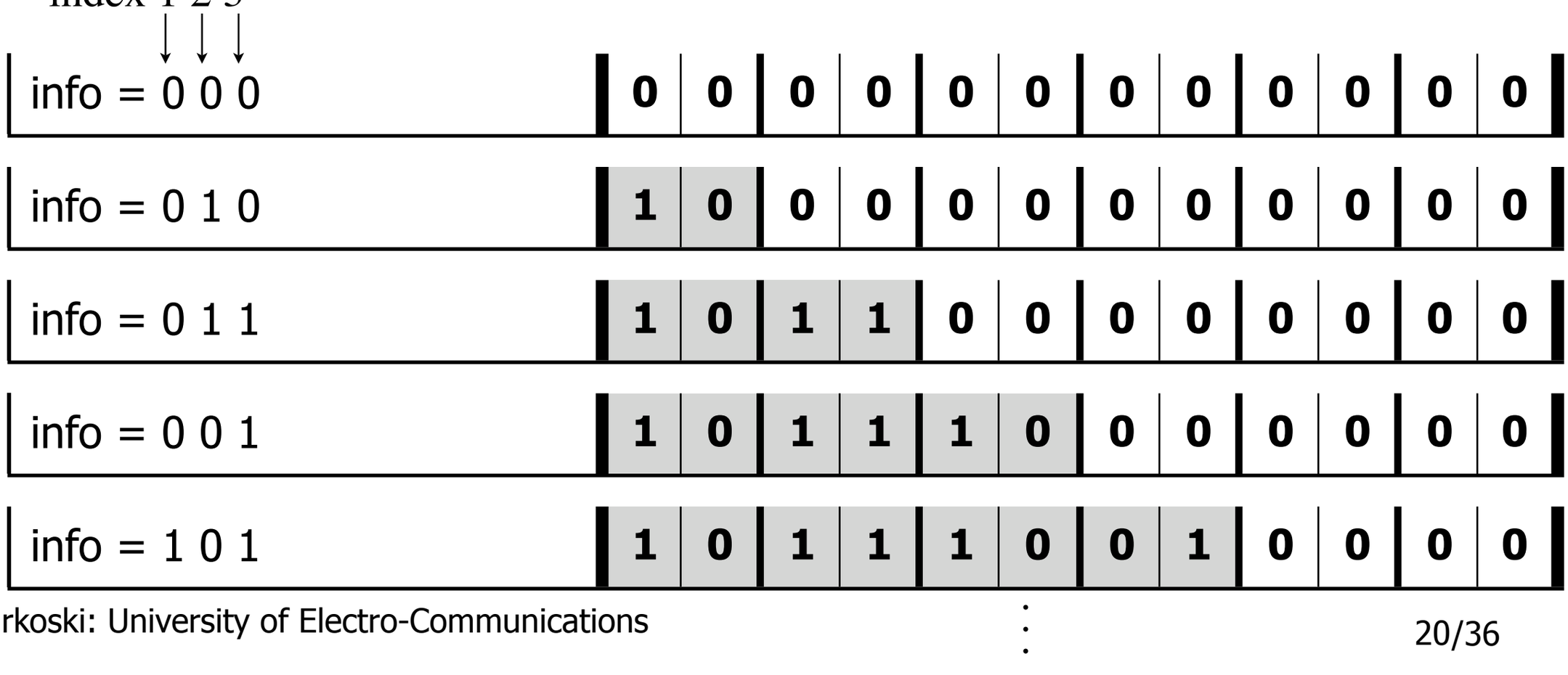
A Binary Index-Type Scheme

The following codes was given by Mahdavifar, et al [MSVWY] at ISIT 2009 to illustrate a more complicated code.

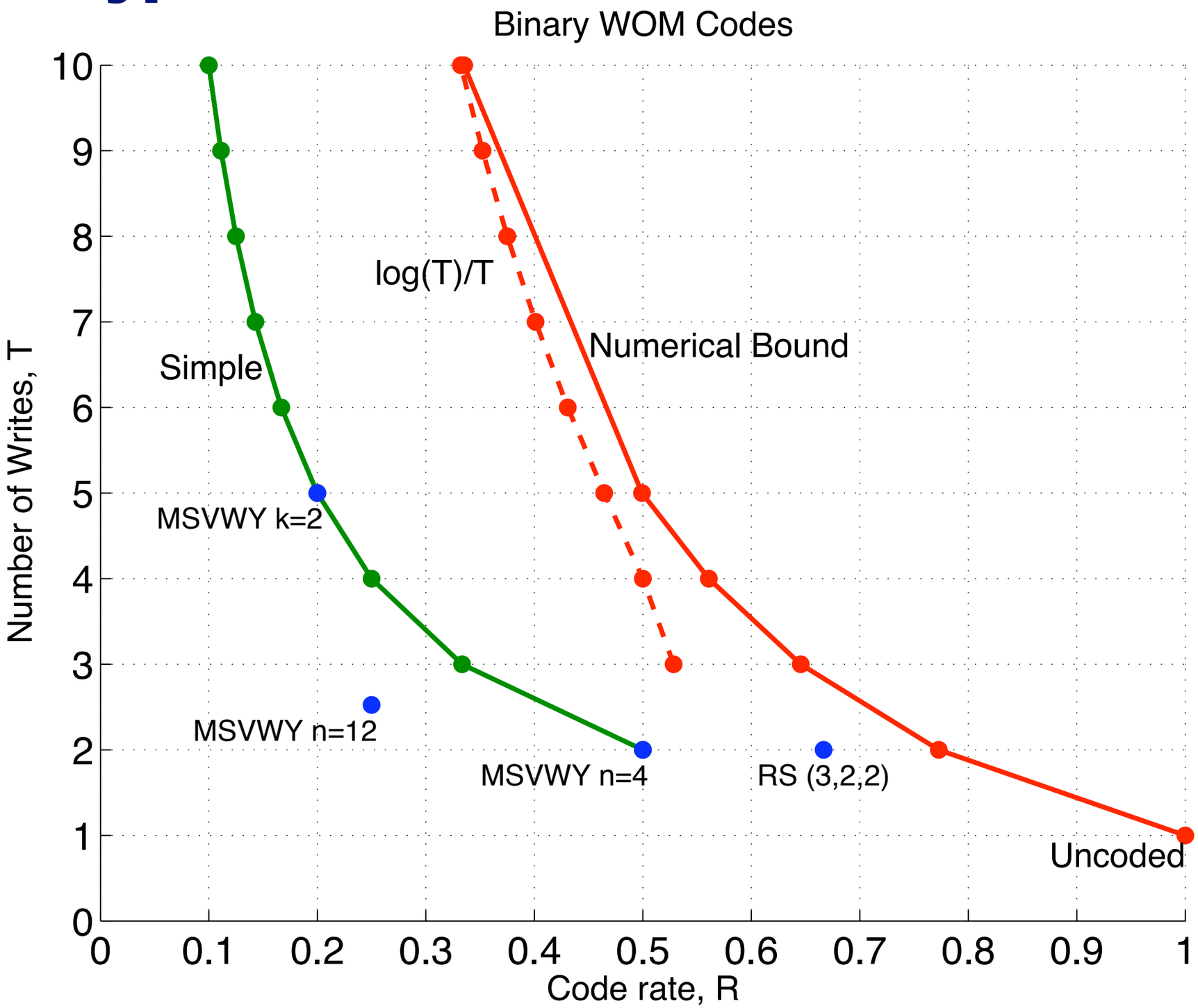
Has poor rate, but explains an index-type scheme.

Encoding: partition n cells into blocks of size $\log_2 k$. When an information bit changes, record its index in the next available block.

Example: $n = 12, R = \frac{1}{4}, k = 3$. Results in $T = \frac{1}{R \log_2 k}$ word writes.



Binary WOM Codes
Index-Type Scheme



d

A Linear Scheme

Cohen, Godlewski and Merkx, "Linear Binary Code for Write-Once Memories," IT Trans., 1986.

Use coset coding to encode information. Pick a linear code. Encoding:

1. Information is encoded as the syndrome of a sequence
2. From the coset of that syndrome, select the coset codeword with the minimum weight.
3. Write that coset codeword to memory.

Decoding:

1. Compute the syndrome of the recorded sequence.

Example: Use Hamming (7,4) code to encode information with $T = 3$ writes:

$$(0,0,0) \underset{①}{\rightarrow} (1,0,1) \underset{②}{\rightarrow} (1,1,0) \underset{③}{\rightarrow} (0,0,0)$$

Hamming Codewords = stored data

Coset leaders

0000000	0000001	0000010	0000100	0001000	0010000	0100000	1000000
0001111				0000111		0101111	1001111
0010011				0011011		0110011	1010011
0011100				0010100		0111100	1011100
0100101				0101101		0000101	1100101
0101010				0100010		0001010	1101010
0110110				0111110		0010110	1110110
0111001				0110001		0011001	1111001
1000110				1001110		1100110	0000110
1001001				1000001		1101001	0001001
1010101				1011101		1110101	0010101
1011010				1010010		1111010	0011010
1100011				1101011		1000011	0100011
1101100				1100100		1001100	0101100
1110000				1110000		1010000	0110000
1111111				1110111		1011111	0111111

Syndromes = Information

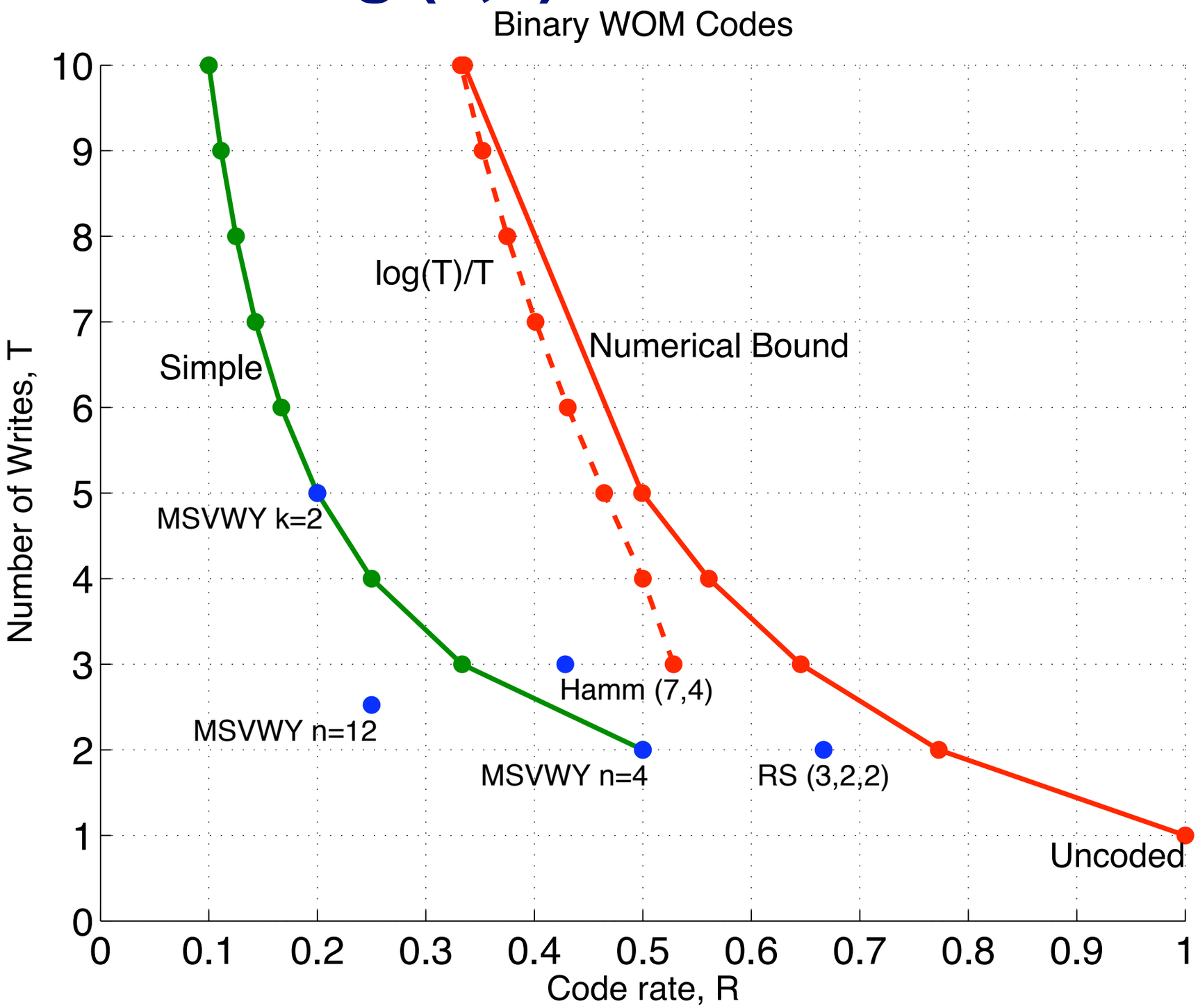
000	001	010	100	111	011	101	110

①

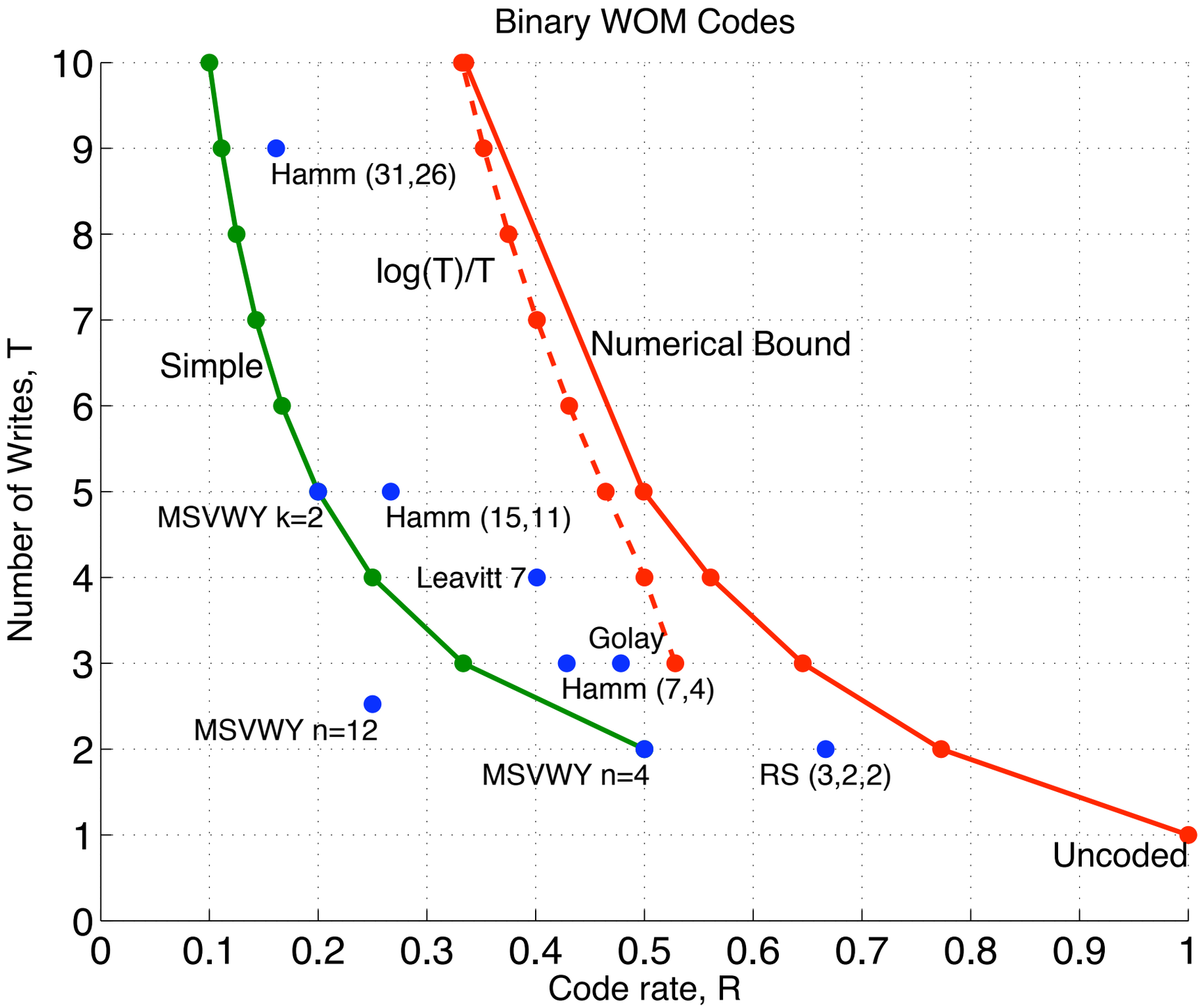
②

③

Binary WOM Codes
Linear Hamming (7,4) Code

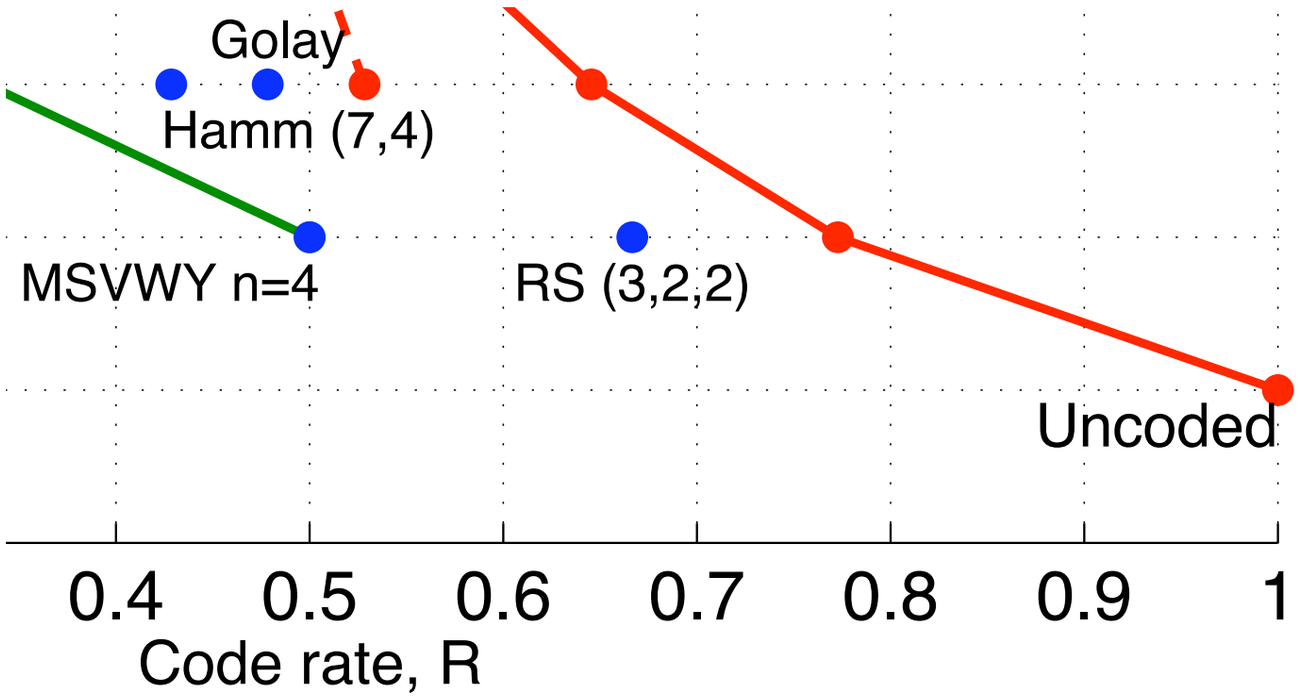


Binary WOM Codes
More Linear Codes

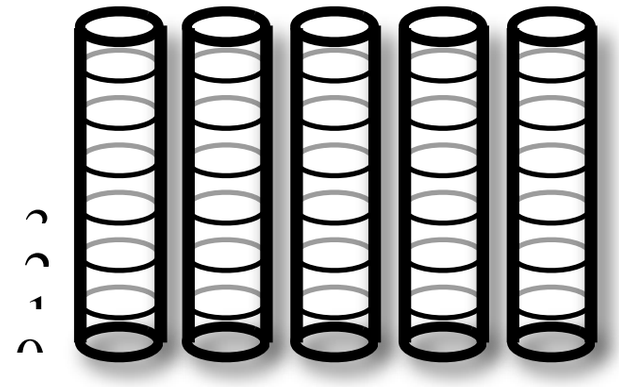


Summary of Binary Codes

- Simple or "naive" coding: $R = \frac{1}{T}$.
- Rivest and Shamir showed that $R = \frac{\log T}{T}$ is possible.
- Clearly, there is a tradeoff in number of writes and rate. But Rivest and Shamir showed you can do better than naive.
- For $T = 2$, the "toy example" n=3, k=2 code has rate 2/3.
- Optimal rate at $T = 2$ is 0.77. This is fairly low rate. Practical?



Codes for Multilevel Flash: q > 2



By increasing q, can we get better codes?

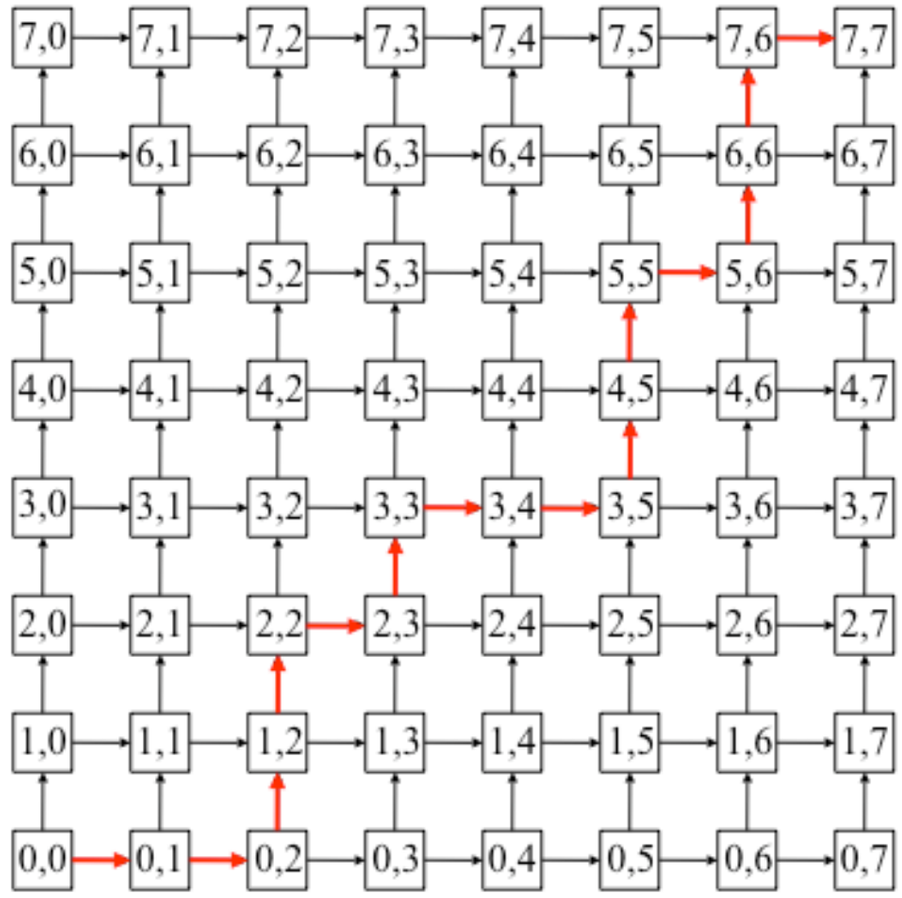
This is recent work, since 2007.

Trivial upper bound:

$$\begin{aligned} t &\leq n(q-1) \quad \text{(bit writes)} \\ T &\leq \frac{(q-1)}{R} \quad \text{(word writes)} \end{aligned}$$

Tighter upper bound (approximate) Jiang, Bohossian, Bruck, ISIT 2007 (JBB07):

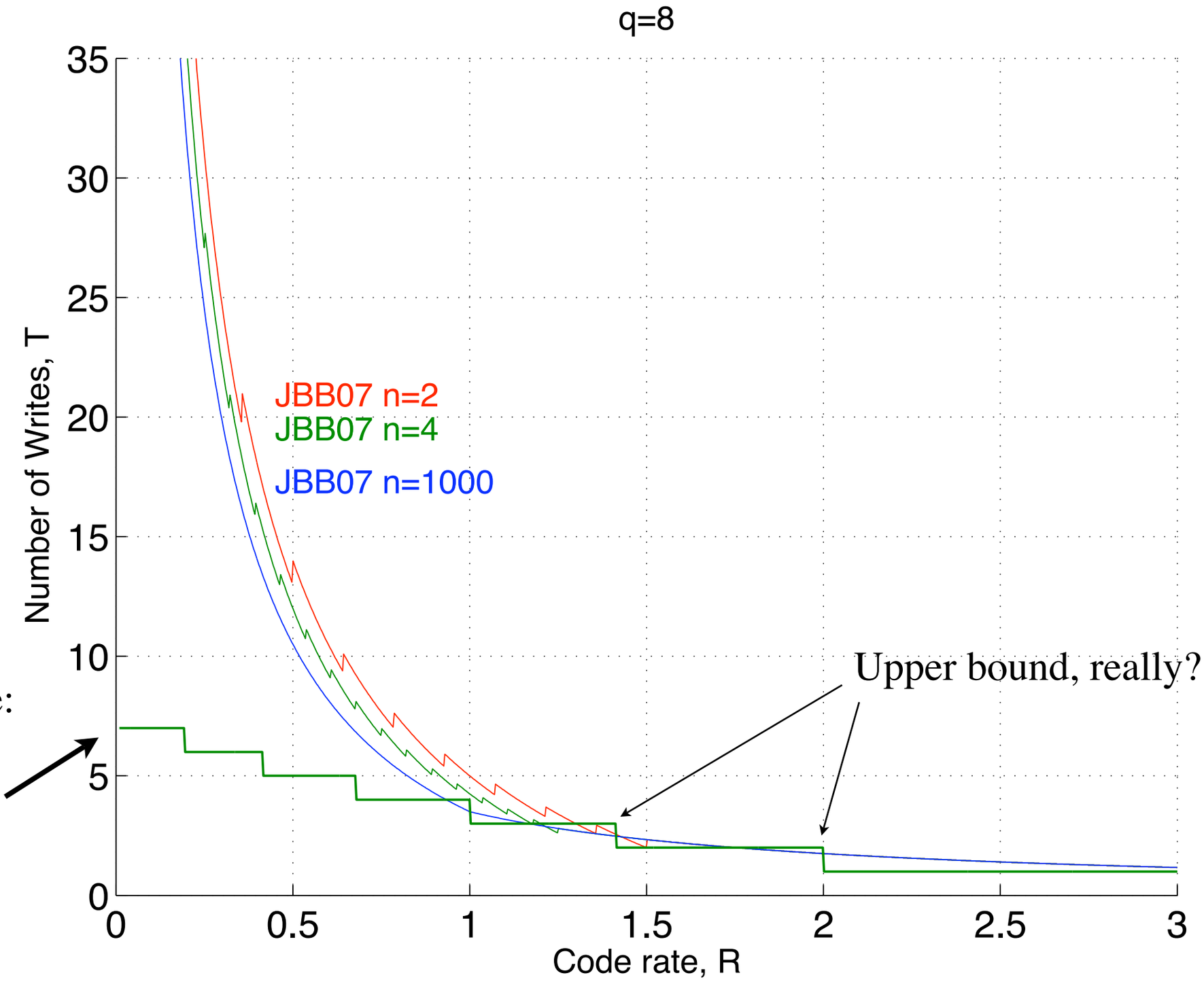
$$\begin{aligned} t &\leq n(q-1) - \frac{1}{2}(q-1)\min\left(n, k-1\right) \\ T &\leq \frac{(q-1)}{R} - \frac{1}{2}(q-1)\min\left(\frac{1}{R}, 1\right) \end{aligned}$$



Trivial bound: n=2, q=8

Image: Eitan Yaakobi

Rewriting Codes for q=8: Bounds



Simple Code:

$$T \leq \left\lfloor \frac{q}{2^k} \right\rfloor$$

Codes for q>2

- Jiang, Bohossian and Bruck [ISIT 2007] also proposed a re-writing code for k=2 bits
  - It complicated and hard to understand.
  - It is a low rate code
  - It achieves:

$$t = (n-1)(q-1) + \lfloor \frac{q-1}{2} \rfloor$$

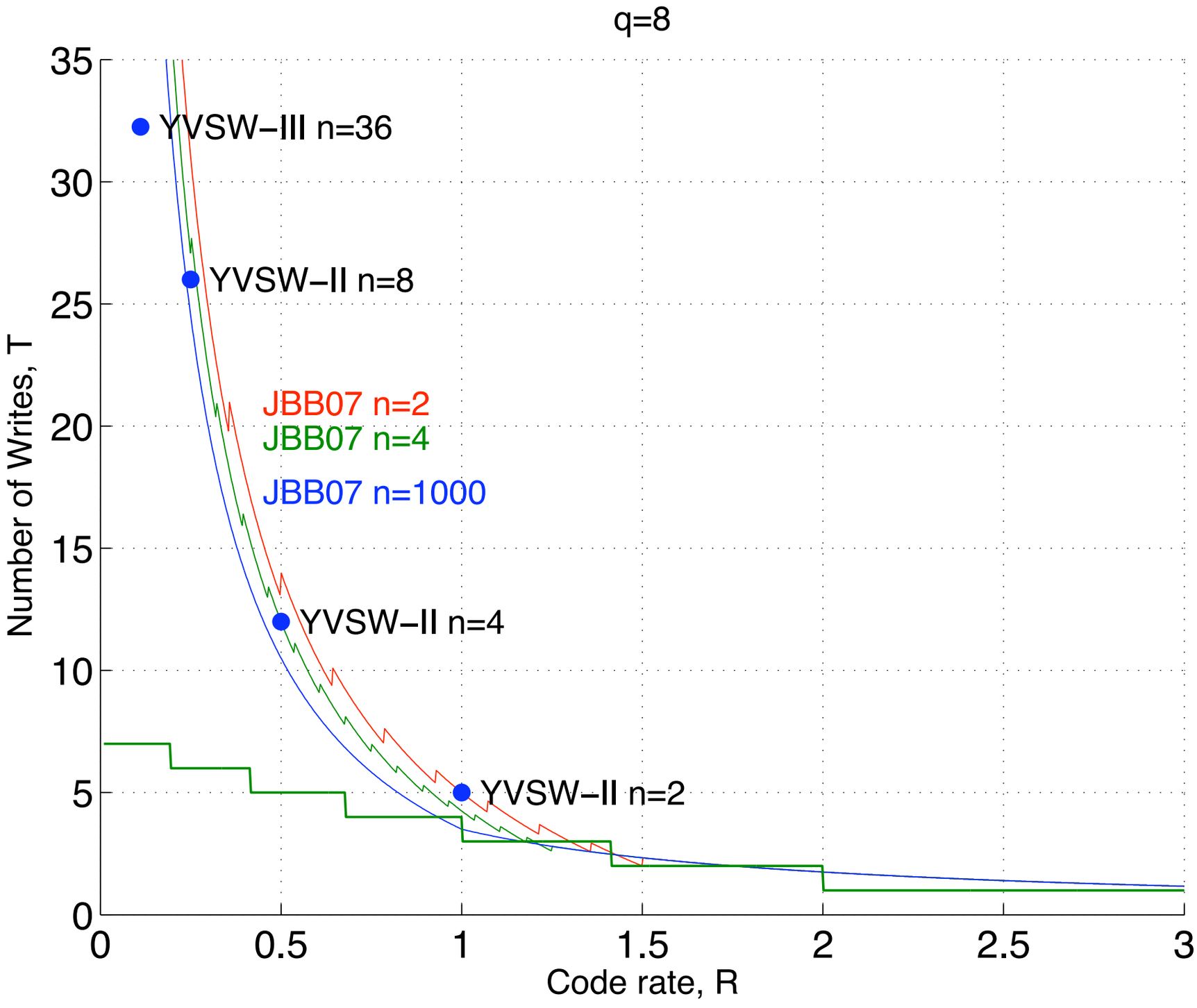
- Yaakobi, Vardy, Siegel and Wolf [Allerton 2008] proposed "multidimensional codes"
  - Achieves the same re-writing rate.
  - Easier to understand the construction

- Jiang, et al. [ISIT 2009] "Trajectory Code" :
- Mahdavifar, et al. [ISIT 2009]

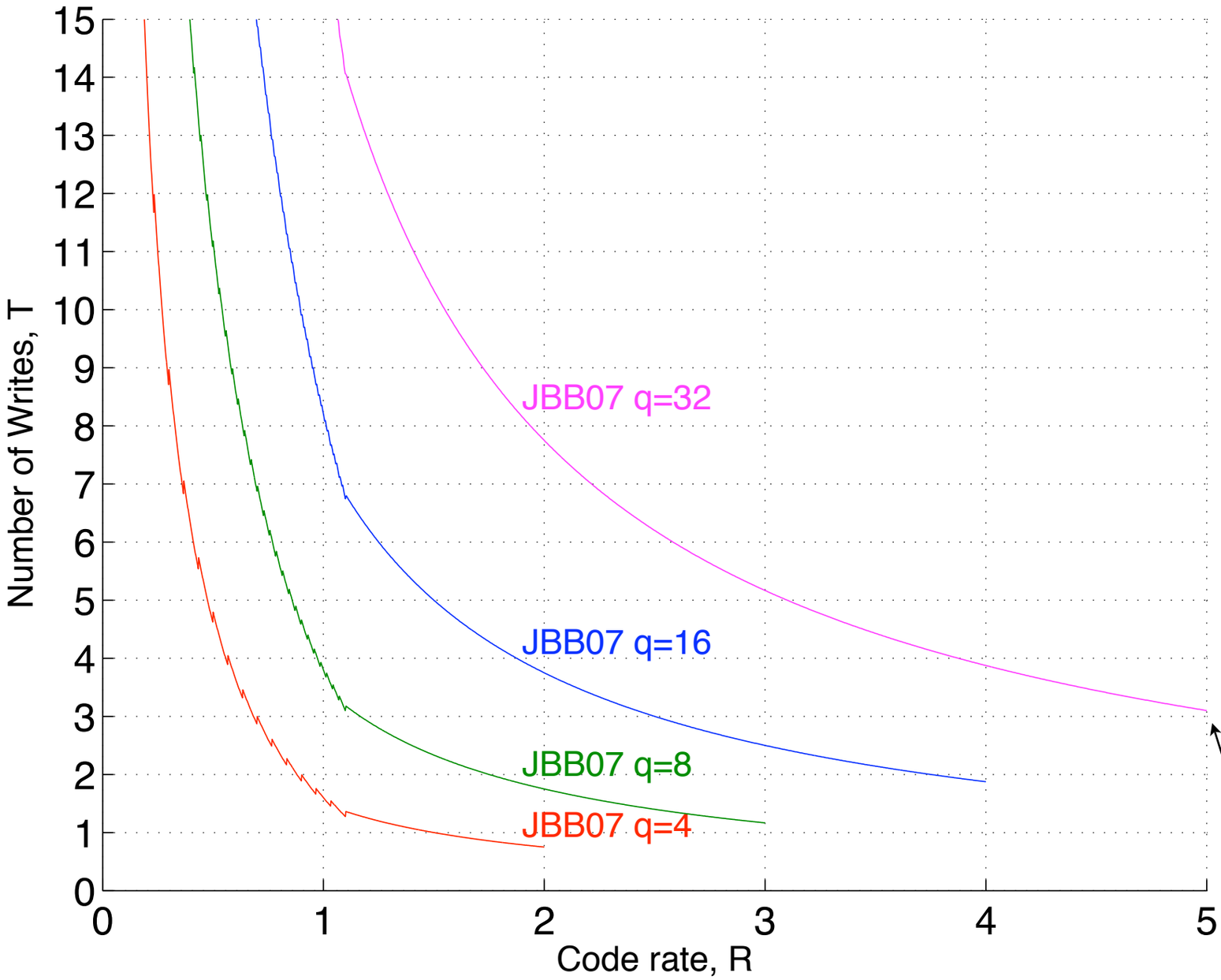
$$2^k \leq 2^{\sqrt{n}} \quad \Rightarrow \quad R \leq \frac{1}{k}$$

- Most constructions appear to be low rate!

Rewriting Codes for q=8: YVSW Code



Number of Writes increases in q!



DAG is directed acyclic graph, the memory model.

"The significant improvement in memory capability is linear with the DAG depth. For a fixed number of states a 'deep and narrow' DAG cell is always preferable to a 'shallow and wide' DAG cell."

-Fiat and Shamir, 1984

Tight bound, really?

Summary of q>2 Codes and Open Problems

- In traditional coding theory, dmin increases for increasing block length
  - But for rewriting codes, does T increase for increasing block length? (no?)
  - However, seems like T does increase for increasing levels q

- High rate coding:
  - system designers use high rate codes, but there are few/no high rate codes
  - perhaps I'm too excited about high rate codes
  - Tighter bounds at high rate?

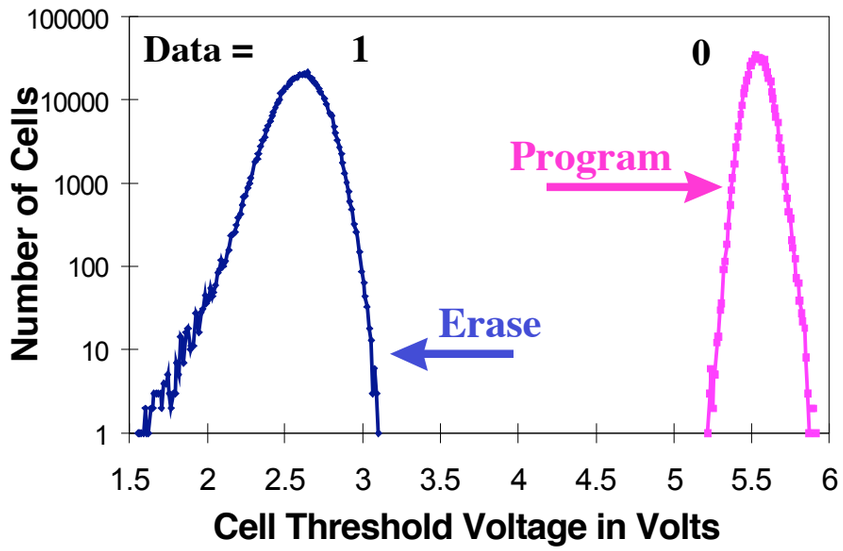
- Average vs. Minimum number of writes
  - t and T was defined as the minimum number of writes
  - Average number of writes is always greater
  - Does average number of writes have better properties (improves with block length)?

- I did not mention other rewriting codes developed by Jiang, et al:
  - Buffer coding
  - Rank modulation

Error-Correction for Flash Memories

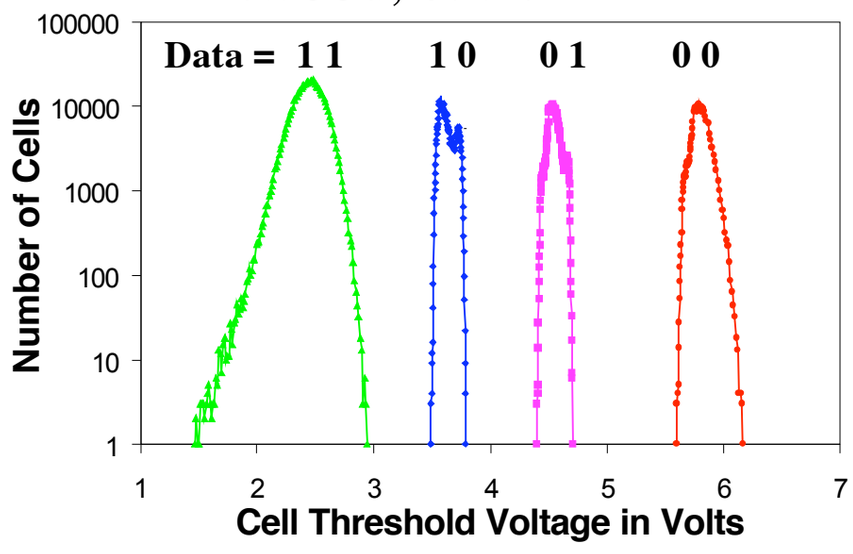
- Flash memories, particularly NAND flash are noisy.

Use Gray mapping



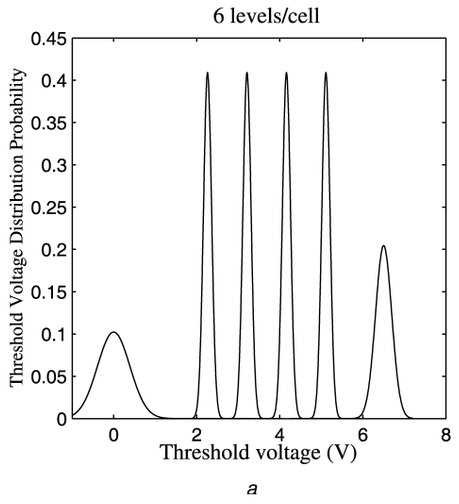
SLC

Atwood, et al.



MLC (2 bits)

Sun, et al.



Model:

- level 0: $\sim N(0, 4\sigma^2)$
- level 1 to $q-2$: $\sim N(0, \sigma^2)$
- level $q-1$: $\sim N(0, 2\sigma^2)$

Error-Correction for Flash Memories

- Most MLC flash uses error correction
  - Early chips: proposal to use Hamming codes to correct single bit errors

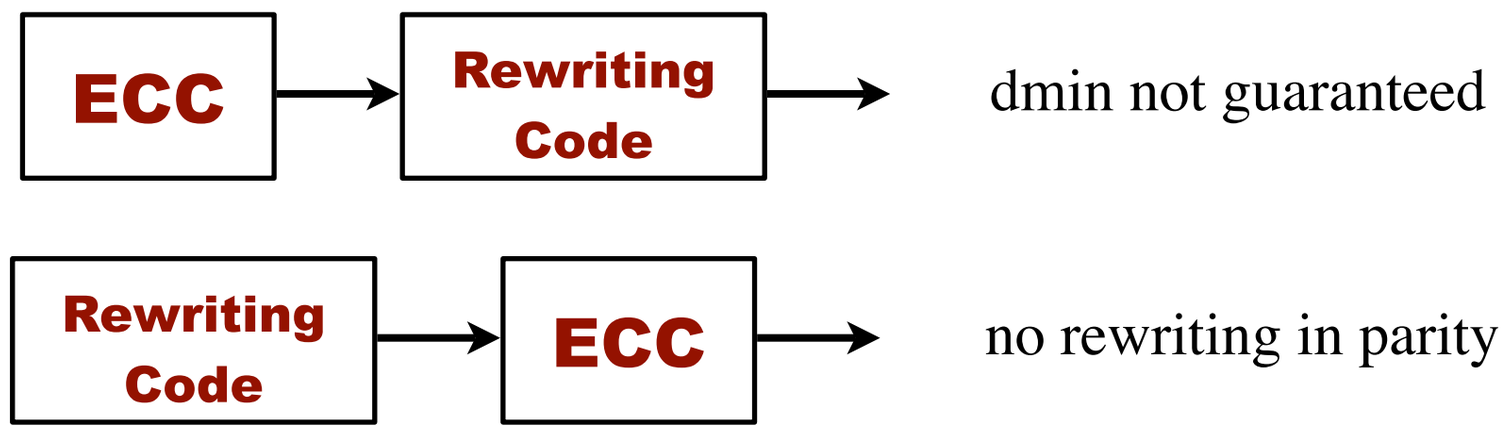
- MLC errors appear random:
  - Reed-Solomon codes correct burst errors well
  - Reed-Solomon codes, widely used in hard drives, DVDs, CD, etc, are not needed
  - However, Reed-Solomon has more efficient decder [Chen et al., 2008]

- BCH codes can correct random errors well (R > 0.98)
  - Liu, Rho and Sung (2006): BCH (4148,4096) to correct 4 bit errors with 52 parity bits
  - Micheloni, et al. (2006): VLSI using BCH (32767,32692) to correct 5 errors

- LDPC Codes
  - Maeda and Kaneko (2009): Use non-binary LDPC codes of field size q
    - q=8, 16. R=1/2, 5/8. Found slight improvement in BER by using average column weight of 2.5

More Open Problems

- Rewriting codes plus ECC
  - Only a few papers on this topic. But, a serious problem (think RLL in hard drives)



- Intersymbol interference (ISI)
  - Errors often appear independent , so BCH codes are used
  - However, densities increase → errors become correlated, ISI occurs
  - Need ISI models!

- Asymmetric Noise
  - read disturb and retention problem: charge leaks from the cell → voltage decrease
  - Errors are asymmetric

Conclusion

- Flash memories are rapidly increasing in density, and should become widespread in the future.
- Flash memories have a limited number of write cycles. Avoid erasures by using coding
  - Binary codes are suitable for SLC, but SLC is being replaced by MLC
  - There appear to be few codes of sufficiently high rate for MLC
- Flash memories also have errors like a traditional communication system
  - Hamming codes, BCH codes, Reed-Solomon, LDPC codes appear to be effective