# Modulation Codes for Flash Memory Based on Load-Balancing Theory

Fan Zhang and Henry D. Pfister

Texas A&M University
College Station

Allerton Conference 2009
University of Illinois at Urbana-Champaign
September 30, 2009

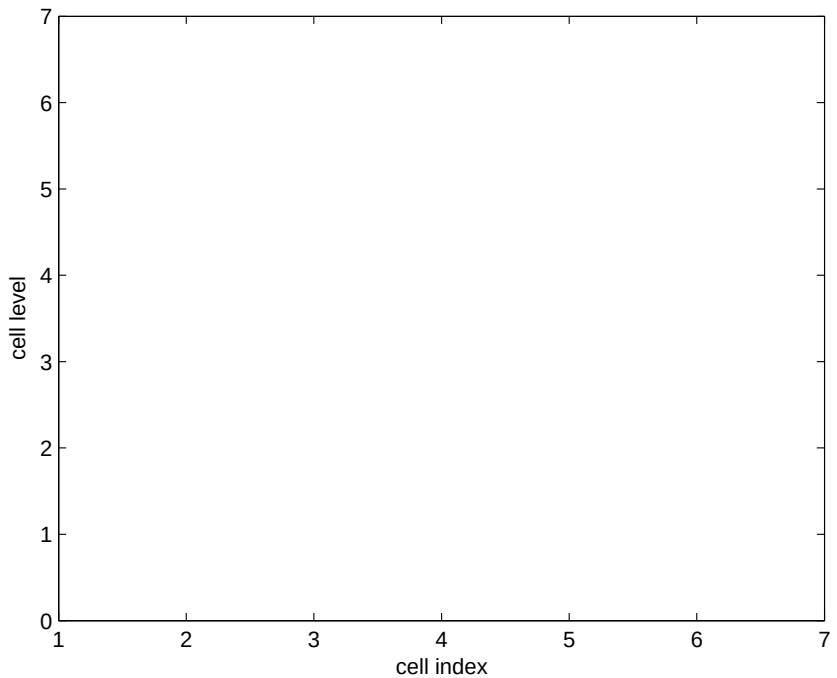
# Outline

# Flash Memory

- Flash memory: high reliability, high storage density, relatively low cost, low power consumption and high read/write speed.
- Some properties of flash memory
  - ($10^5$~$10^{20}$) cells are organized as a block
  - Adding charge to a cell is easy, but block erasure is needed if we need to reduce the charge level
  - Maximum number of erasure ~$10^6$
  - Multi-level cell technology increases the density of storage

# A Simple Modulation Code: Step 1

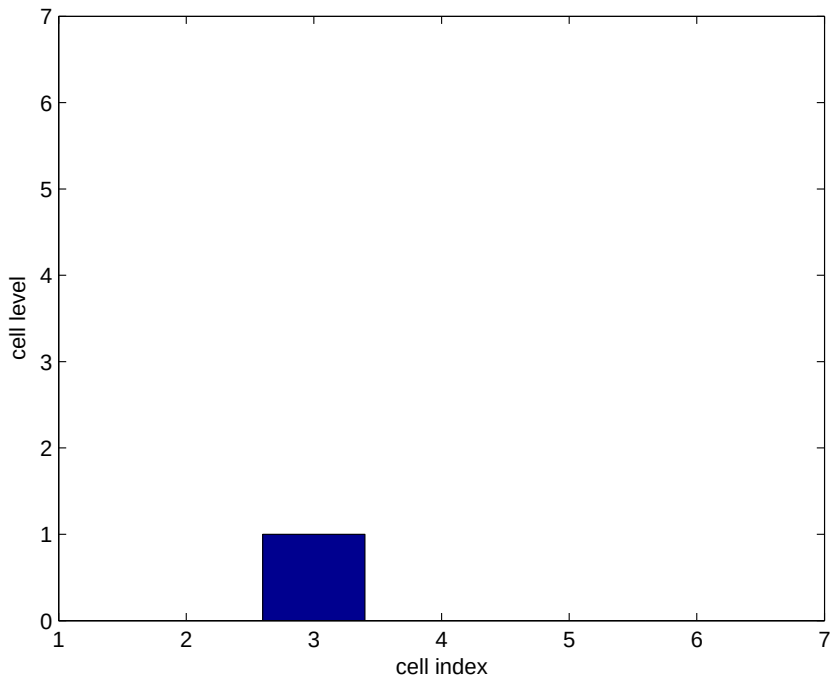
Example: Storing 3 bits in 7 cells, 3-bit data value can change arbitrarily (unconstrained data model)




- State time $t$: $s_t$
  - $s_0 = [0,0,0,0,0,0,0]$
- Decoder: $\hat{x}_t = g(s_t)$
  - $\sum_{j=1}^{7} s_t(j)\, j \bmod 8$
  - $\hat{x}_0 = g(s_0) = 0$

# A Simple Modulation Code: Step 1

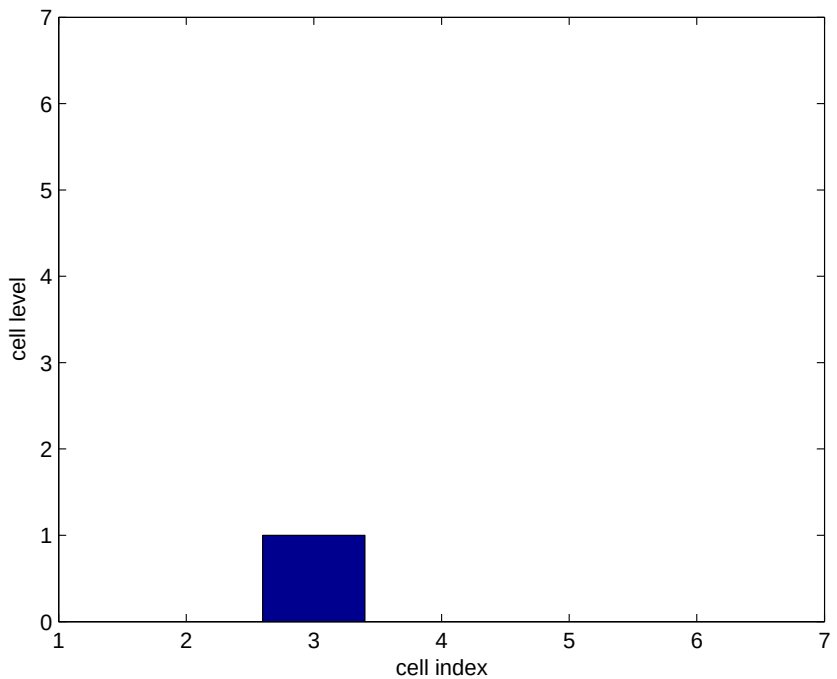
Example: Storing 3 bits in 7 cells, 3-bit data value can change arbitrarily (unconstrained data model)




- State time $t$: $s_t$
  - $s_0 = [0,0,0,0,0,0,0]$
- Decoder: $\hat{x}_t = g(s_t)$
  - $\sum_{j=1}^{7} s_t(j)\, j \bmod 8$
  - $\hat{x}_0 = g(s_0) = 0$
- Encoder: $x_1 = 3$
  - $\Delta x_1 = x_1 - \hat{x}_0 \equiv 3$
  - Increase 3rd cell by 1
  - $s_1 = [0,0,1,0,0,0,0]$

# A Simple Modulation Code: Step 2

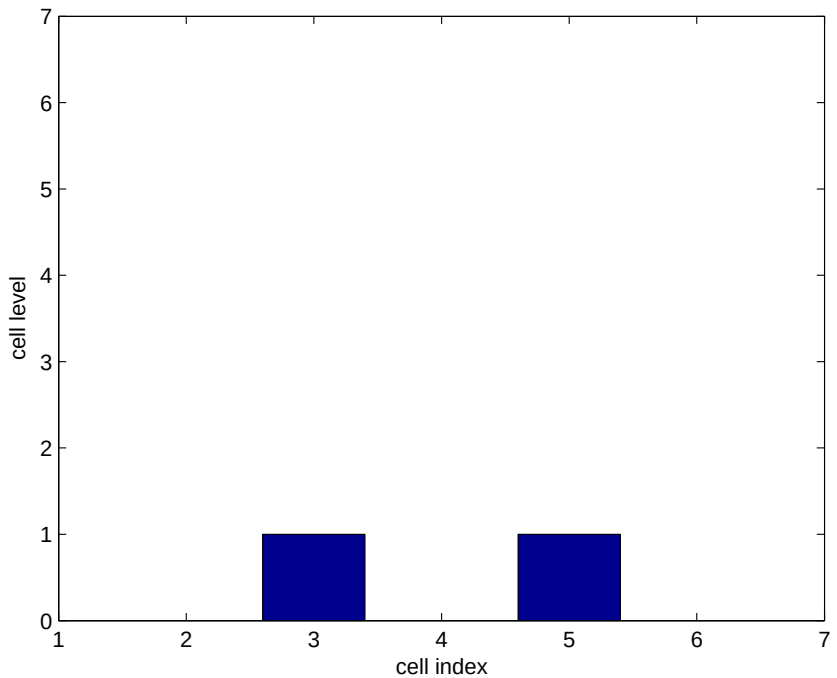
Example: Storing 3 bits in 7 cells, 3-bit data value can change arbitrarily (unconstrained data model)




- $s_1 = [0,0,1,0,0,0,0]$
- Decoder: $\hat{x}_t = g(s_t)$
  - $\sum_{j=1}^{7} s_t(j)\, j \bmod 8$
  - $\hat{x}_1 = g(s_1) = 3$

# A Simple Modulation Code: Step 2

Example: Storing 3 bits in 7 cells, 3-bit data value can change arbitrarily (unconstrained data model)




- $s_1 = [0,0,1,0,0,0,0]$
- Decoder: $\hat{x}_t = g(s_t)$
  - $\sum_{j=1}^{7} s_t(j)\, j \bmod 8$
  - $\hat{x}_1 = g(s_1) = 3$
- Encoder: $x_2 = 0$
  - $\Delta x_2 = x_2 - \hat{x}_1 \equiv 5$
  - Increase 5th cell by 1
  - $s_2 = [0,0,1,0,1,0,0]$
  - $g(s_2) = 3+5 \equiv 0$

# A Simple Modulation Code: Properties

- Uses $2^3 - 1$ cells to store $3$ binary variables
- When the input r.v. is not uniform, some cells may reach maximum before others
- Can we have a code which increases all cell-levels with equal probability for arbitrary i.i.d. input r.v.?

# Problem Setup

- Use $n$ $q$-level cells (called an $n$-cell) to jointly store $k$ $l$-ary variables (called a $k$-variable)
- Denote input at time $t$ as $x_t \in \mathbb{Z}_l^k$ and cell state as $s_t \in \mathbb{Z}_q^n$
  - Encoder maps $x_t$ and $s_{t-1}$ to $s_t$: $f : \mathbb{Z}_l^k \times \mathbb{Z}_q^n \to \mathbb{Z}_q^n$
  - Decoder only knows $s_t$ and decodes to $\hat{x}_t$: $g : \mathbb{Z}_q^n \to \mathbb{Z}_l^k$
  - Charge levels can only be increased or stay the same
- Design $f$ and $g$ such that the system is optimized
- If max # of rewrites, symptotically optimum codes:

$$\lim_{n \text{ or } q\to\infty} \frac{\text{number of rewrites}}{n(q-1)} = 1.$$

# Data Model and Performance Metric

- In previous works, the authors consider data models and optimality measures as follows.

| teams | input data model | | max # of rewrites |
|---|---|---|---|
| [Jiang et. al] | any sequence | 1 var changed | worst case |
| [Finucane et. al] | Markov chain | 1 var changed | average case |
| [Yaakobi et. al] | any sequence | 1 var changed | worst case |

(note: we call it as constrained data model when only 1 variable changed each rewrite)

# Another Performance Metric

- Consider another metric: storage efficiency

$$\gamma \triangleq E\left(\frac{\sum_{i=1}^{R} I_i}{n(q-1)}\right),$$

  - $I_i$ : amount of information stored at the $i$-th rewrite
  - $R$: the number of rewrites between two erasures.
- Let $N$ be the number of $n$-cells in a block
  - $\max_{f,g}$ #rewrites in worst case = $\max_{f,g}\ \gamma$ when $N \to \infty$
  - $\max_{f,g}$ #rewrites on average = $\max_{f,g}\ \gamma$ when $N = 1$

# Data Models and Upper Bounds on $\gamma$

- Constrained data model ($n = kl$ )
  - Only one of the $k$ variables changes at a time
  - Implies $\gamma < \log_2 kl$.
- Unconstrained data model ( $n = l^k$ )
  - Arbitrary data changes allowed
  - Implies $\gamma < k \log_2 l$
- Large $k$ (or $l$) needed to achieve high storage efficiency $\gamma$
- Assume all cells in block used as a single $n$-cell, i.e., $N = 1$

# A Self-randomized Modulation Code (SRMC)

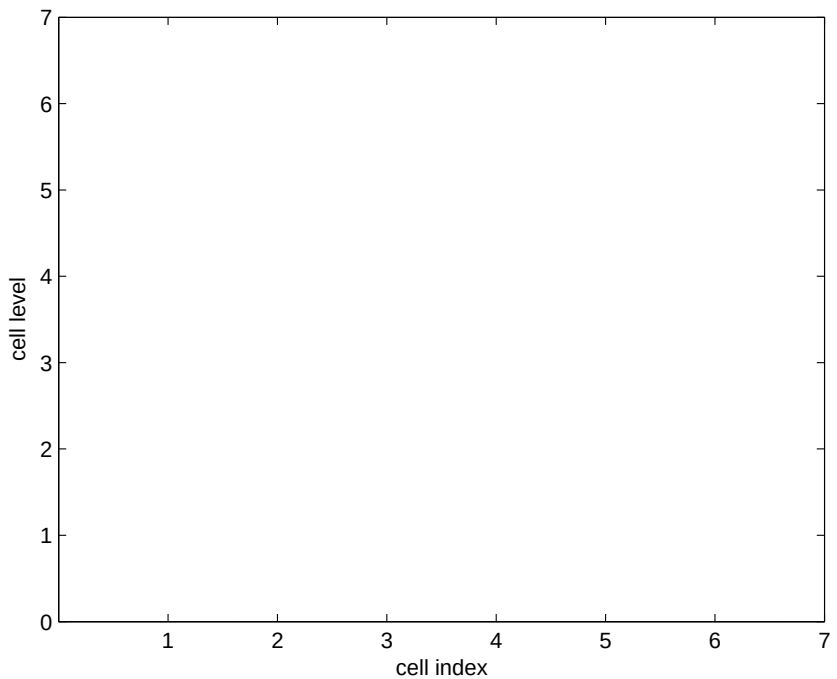
- Asymptotically optimal: $\lim_{q\to\infty} \frac{\text{average \#of rewrites}}{n(q-1)} = 1$.
  - For arbitrary $k$ and $l$ and arbitrary i.i.d. input distributed r.v.
  - Optimality identical to *weakly robust codes* in [Jiang ISIT09]
- The code in [Finucane el. al] is asymptotically optimal for arbitrary i.i.d. input distributed r.v. only when $k = 2$.
- For arbitrary $k$ and $l$, SRMC uses $n = l^k$ cells.

# A Self-randomized Modulation Code (SRMC)

- Main idea behind SRMC:
  - Use deterministic scrambling to randomize cell index
- Encoder
  - First decode $s_{t-1}$ to the value $\hat{x}_{t-1}$ stored in the $n$-cell, then calculate difference $x_t - x_{t-1} \bmod l^k$
  - Randomize difference $\Delta x_t = x_t - \hat{x}_{t-1} + \|s_{t-1}\|_1 \bmod l^k$ and increase the cell-level by 1
  - Randomizing the mappings over time induces a uniform distribution over cell indices regardless of input distribution

# SRMC Example: Initialization

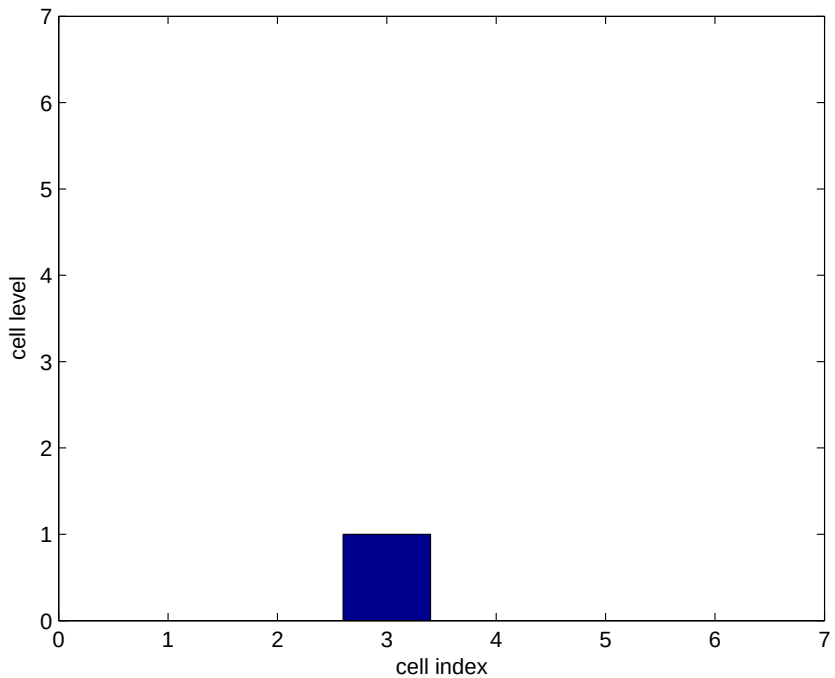
Example: Storing 3 bits in 8 cells, 3-bit data value can change arbitrarily (unconstrained data model)




- $s_0 = [0,0,0,0,0,0,0,0]$, $\hat{x}_0 = 0$

# SRMC Example: Step 1

Example: Storing 3 bits in 8 cells, 3-bit data value can change arbitrarily (unconstrained data model)

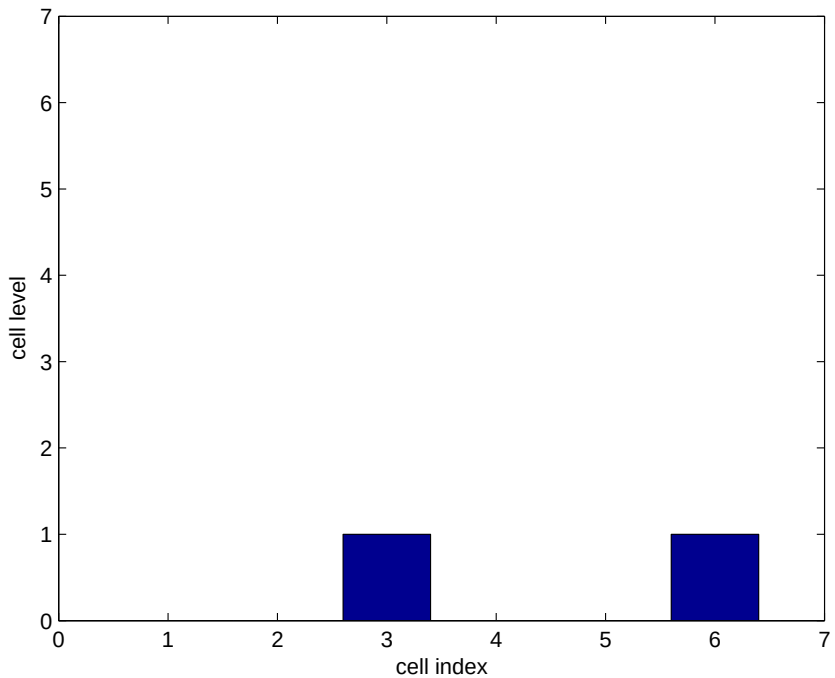



- $s_0 = [0,0,0,0,0,0,0,0]$, $\hat{x}_0 = 0$
- Encode: $x_1 = 3$
  - $\Delta x_1 = x_1 - \hat{x}_0 + \|s_0\| \equiv 3$
  - Increase 3rd cell by 1
- Decode:

$$\hat{x}_1 = 3 - \frac{\|s_0\|\,(\|s_0\|+1)}{2} \equiv 3$$

# SRMC Example: Step 2

Example: Storing 3 bits in 8 cells, 3-bit data value can change arbitrarily (unconstrained data model)

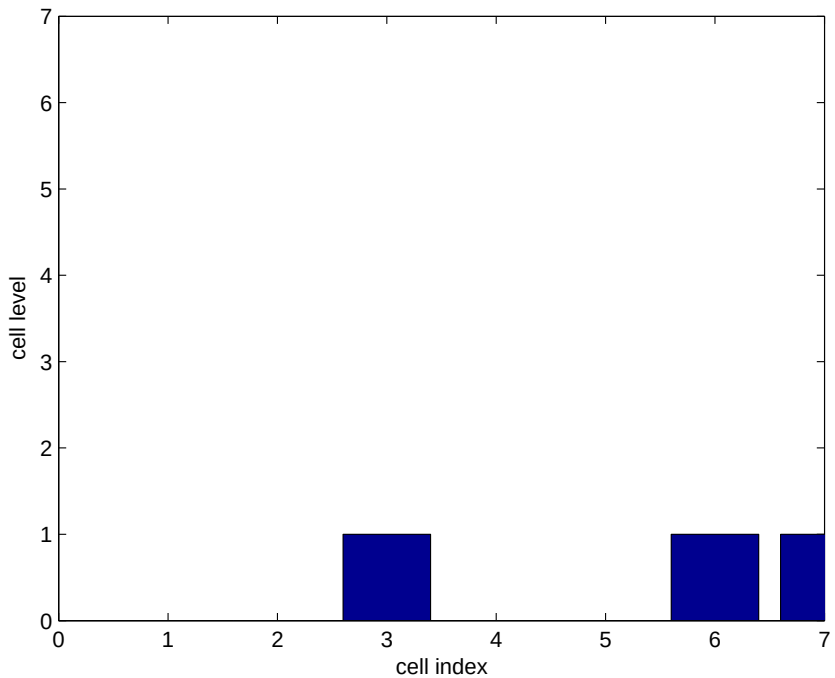



- $s_1 = [0,0,0,1,0,0,0,0]$, $\hat{x}_1 = 3$
- Encode: $x_2 = 0$
  - $\Delta x_2 = x_2 - \hat{x}_1 + \|s_1\| \equiv 6$
  - Increase 6th cell by 1
- Decode:

$$\hat{x}_2 = 9 - \frac{\|s_1\|\,(\|s_1\|+1)}{2} \equiv 0$$

# SRMC Example: Step 3

Example: Storing 3 bits in 8 cells, 3-bit data value can change arbitrarily (unconstrained data model)




- $s_2 = [0,0,0,1,0,0,1,0]$, $\hat{x}_2 = 0$
- Encode: $x_3 = 5$
  - $\Delta x_3 = x_3 - \hat{x}_2 + \|s_2\| \equiv 7$
  - Increase 7th cell by 1
- Decode:

$$\hat{x}_3 = 16 - \frac{\|s_2\|\,(\|s_2\|+1)}{2} \equiv 5$$

# Questions on SRMC

- There are totally $l^k - 1$ possible values for the new message. But we use $l^k$ cells to store one of those $l^k - 1$ values. Is it necessary?
  - A group-theoretic analysis shows that we need at least 1 extra cell to let the code be robust against arbitrary i.i.d. r.v.
- The asymptotic optimality requires $q \to \infty$ which is not true in practice (MLC with 16 levels is still very cutting-edge). How can we analyze/improve $\gamma$ for moderately large $q$?
  - Tools from load-balancing theory

# Load Balancing and Modulation Codes

- Load-balancing is a technique to distribute objects (e.g., workloads) evenly across two or more resources (e.g., computers, CPU's and hard drives)
- Classical load-balancing: the balls-and-bins problem
  - $n$ balls are thrown into $n$ bins independently and uniformly
  - What is the maximum load (w.h.p.) as $n \to \infty$? ($\approx \frac{\ln n}{\ln \ln n}$)
- Connections between balls-and-bins problem and modulation code design problem
  - Think of balls as charge levels and bins as cells
  - Don't worry about decodability for now

# Results from Load-Balancing Theory

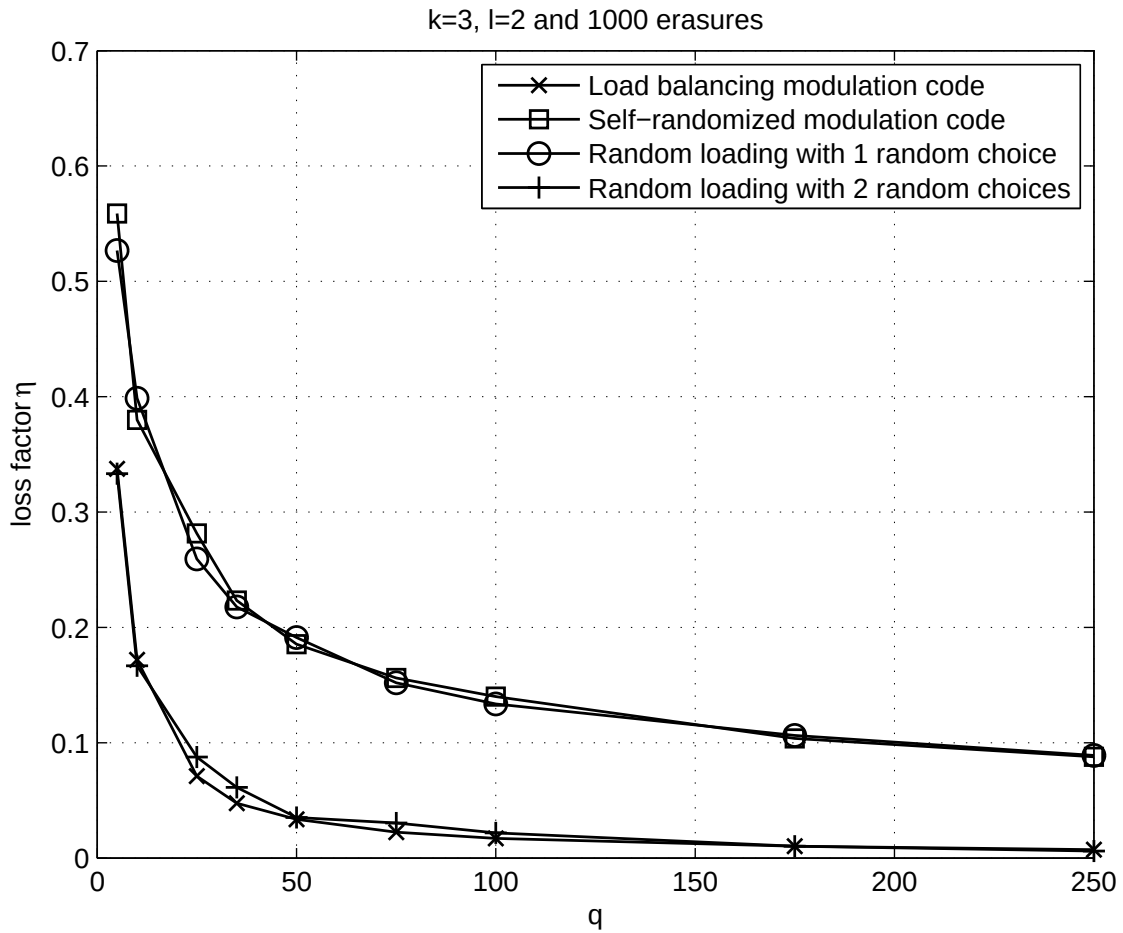
- $n(q-1)$ balls thrown into $n$ bins
  - Each ball placed into least loaded bin of $d$ random choices
  - $M$ = number of balls in most loaded bin
- Scaling as $n \to \infty$ with $q$ constant
  - $M \approx O\left(\frac{\ln n}{\ln \ln n}\right)$ when $d = 1$ (RL1C)
  - $M \approx (q-1) + O\left(\frac{\ln \ln n}{\ln d}\right)$ when $d \geq 2$ (RL$d$C)
- More general scaling in [Raab et. al.] [Karlinz et. al.]
- Can we achieve decodability without losing any load-balancing performance?
  - SRMC has the same l.b. performance with RL1C
  - LBMC has the same l.b. performance with RL$d$C

# Load-balancing Modulation Codes (LBMC)

- The idea of LBMC:
  - Each value can be stored by increasing any of $d$ cells
  - Encoding adds one to the cell-level of least charged cell
  - By making these $d$ choices independent with each other and uniform over time, LBMC performs like RL$d$C

# Load Balancing Performance of SRMC and LBMC

To study the load-balancing capability, define $\eta \triangleq 1 - \frac{E[R]}{n(q-1)}$.




(note: the code in [Finucane et. al. ] also matches RL1C)

# Storage Efficiency of SRMC and LBMC (small $n$)

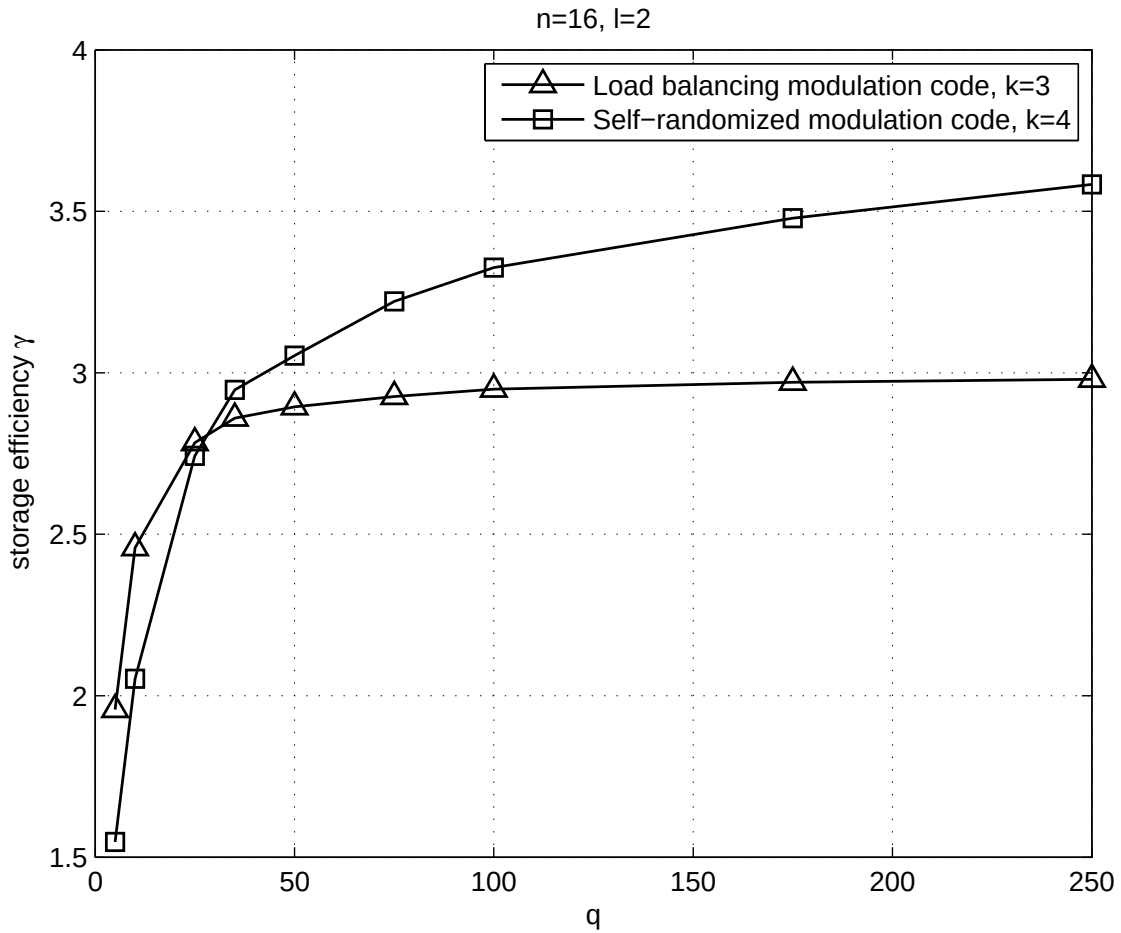



Figure: Storage Efficiency of SRMC and LBMC with $n = 16$.

# Storage Efficiency of SRMC and LBMC (large $n$)

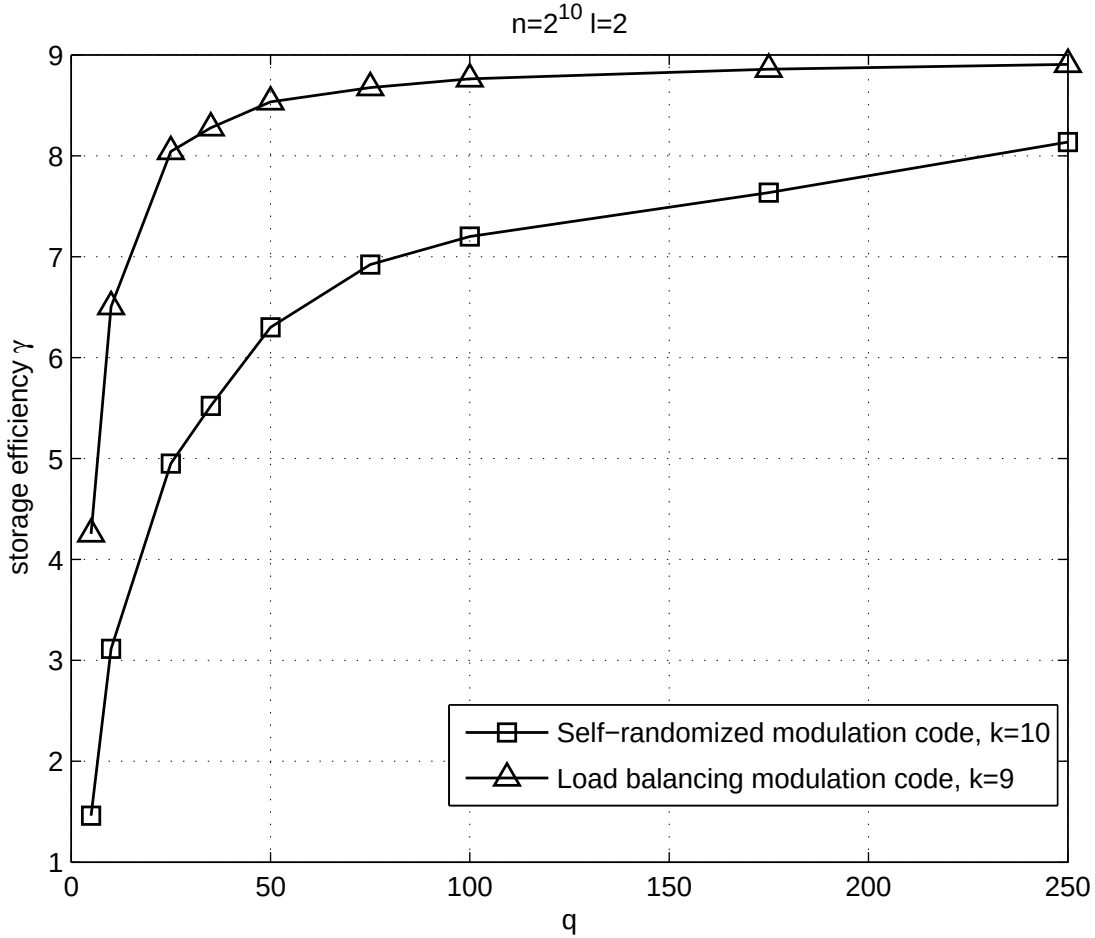



Figure: Storage Efficiency of SRMC and LBMC with $n = 2^{10}$.

# Conclusions

- Proposed a Self-randomized Modulation Code (SRMC)
  - Asymptotically optimal for arbitrary $k, l$ and i.i.d. input
  - Analysis for finite $q$ exposes a load-balancing issue
- Proposed a Load-balancing Modulation Code (LBMC)
  - Analysis implies significant improvement when $q$ is small
- Simulation results verify the analytical conclusions

# Thank You

Thank you