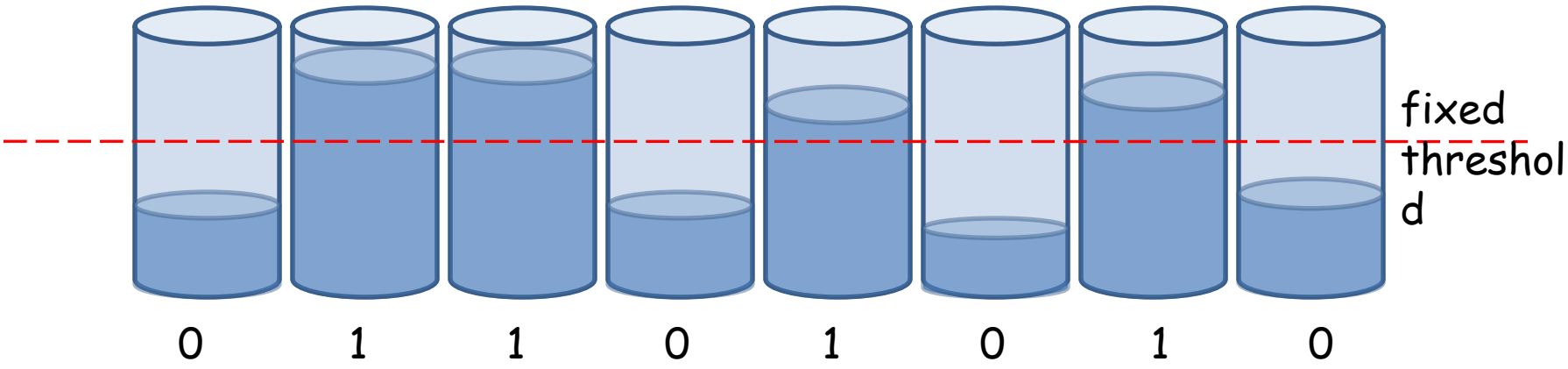
048704/236803
Seminar on Coding for Non-Volatile Memories

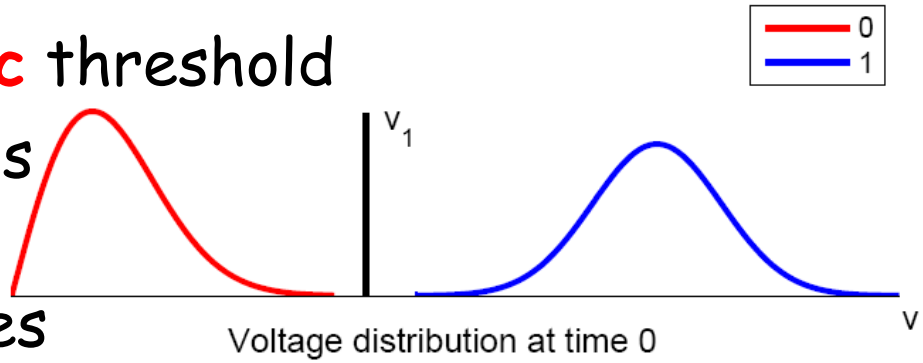
Read Cycle of Flash Memories

- Compare cell levels with a threshold (or a sequence of thresholds)

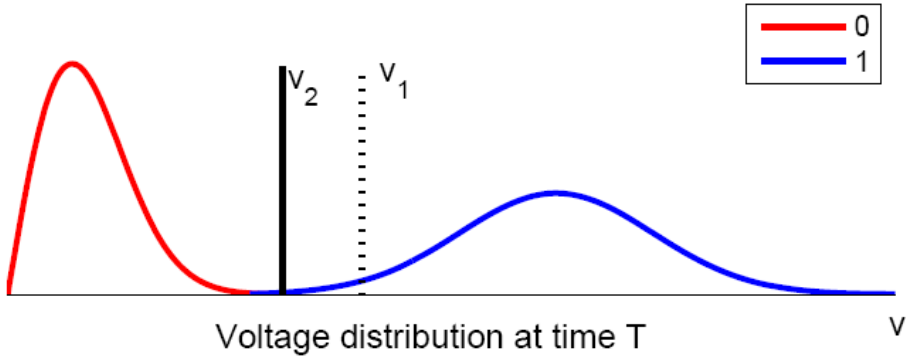


Balanced Codes: Motivation

- Charge Leakage → voltage drift **in one direction**
- **Fixed** threshold vs **dynamic** threshold
- Dynamic reading thresholds reduces the BER
- A **balanced vector** satisfies
  #0's = #1's



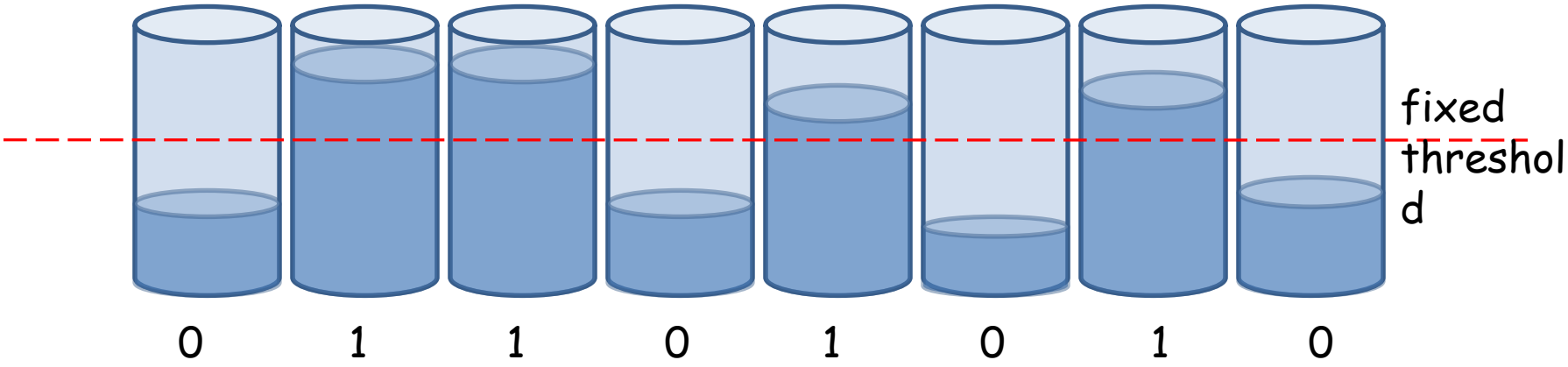
Voltage distribution at time 0



Voltage distribution at time T

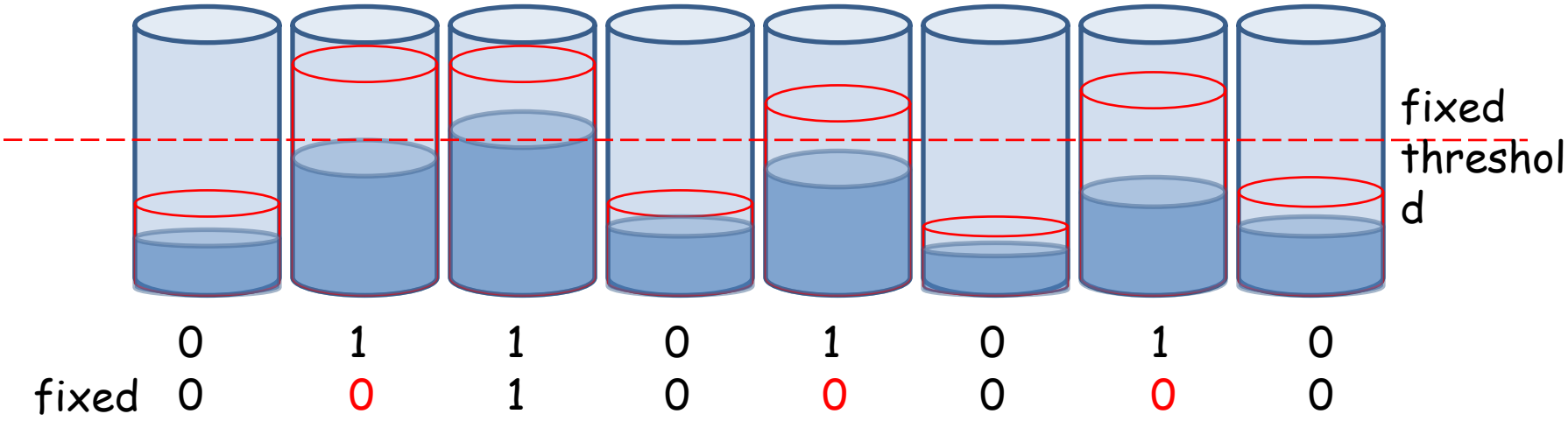
Balanced Codes: Motivation

- In writing, half of cells store **0** and the other half store **1**
- In reading, the **n/2** cells with lower voltages are read as **0**
  The other **n/2** cells with higher voltages are read as **1**
- **Relative ranking** is most likely preserved



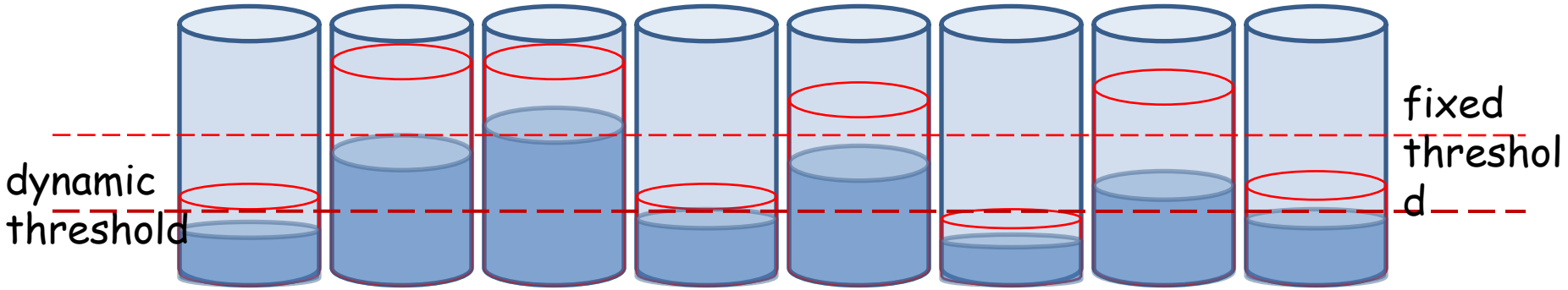
Balanced Codes: Motivation

- In writing, half of cells store **0** and the other half store **1**
- In reading, the **n/2** cells with lower voltages are read as **0**
  The other **n/2** cells with higher voltages are read as **1**
- **Relative ranking** is most likely preserved



Balanced Codes: Motivation

- In writing, half of cells store **0** and the other half store **1**
- In reading, the **n/2** cells with lower voltages are read as **0**
  The other **n/2** cells with higher voltages are read as **1**
- **Relative ranking** is most likely preserved



	0	1	1	0	1	0	1	0
fixed	0	**0**	1	0	**0**	0	**0**	0
dynamic	0	1	1	0	1	0	1	0

Balanced Codes

- Problems:
  1. How to guarantee that **at most** half of the cells have value 1?
  2. How to guarantee that **exactly** half of the cells have value

Balanced Codes

- How many balanced length-n (even) vectors are there?
  - $M(n) = \binom{n}{n/2} = n!/((n/2)!\bullet(n/2)!)$
  - $\log_2 M(n) = n - \frac{1}{2}\bullet\log_2 n - \frac{1}{2}\bullet\log_2(\pi/2) - \varepsilon(n)/n$, $0 \le \varepsilon(n) \le 1.81$
  - $\log_2 M(n) \approx n - \frac{1}{2}\bullet\log_2 n$
- How to construct balanced codes?
  - The redundancy should be close to $\frac{1}{2}\bullet\log_2 n$
  - Lookup table
    - Optimal ✔
    - Not efficient ✗

Knuth's Balancing Algorithm

- Given a length-n vector $\mathbf{v}=(v_0,\ldots,v_{n-1})$
  - If the vector is balanced, no need to do anything
  - Otherwise, find $0\le i\le n-1$ such that the vector

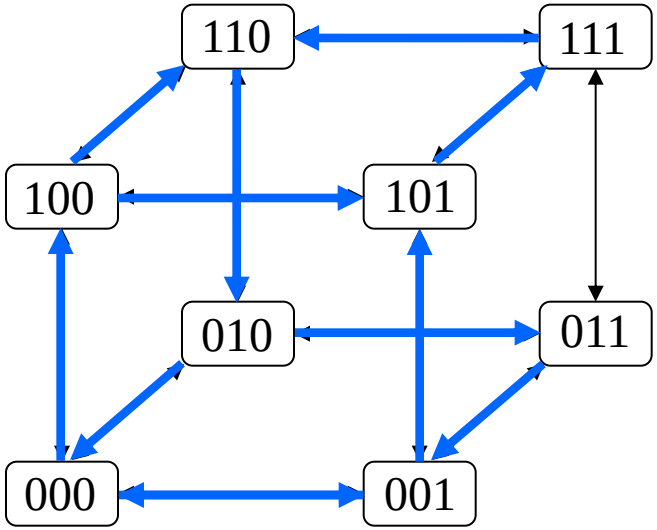
    $\mathbf{v}_i=(1-v_0,\ldots,1-v_i,v_{i+1}\ldots,v_{n-1})$ is balanced
  - Write the index i in $\lceil \log(n) \rceil$ more cells (recursively or using a table so it is balanced)
  - The redundancy is roughly $\lceil \log(n) \rceil$
- Why is it possible to find such an i?
  - $w_H(\mathbf{v}_0) = n - w_H(\mathbf{v}_{n-1})$
  - If $w_H(\mathbf{v}_0) < n/2$ then $n - w_H(\mathbf{v}_{n-1}) > n/2$
  - There exists i such that $w_H(\mathbf{v}_i) = n/2$

Flash/Floating Codes

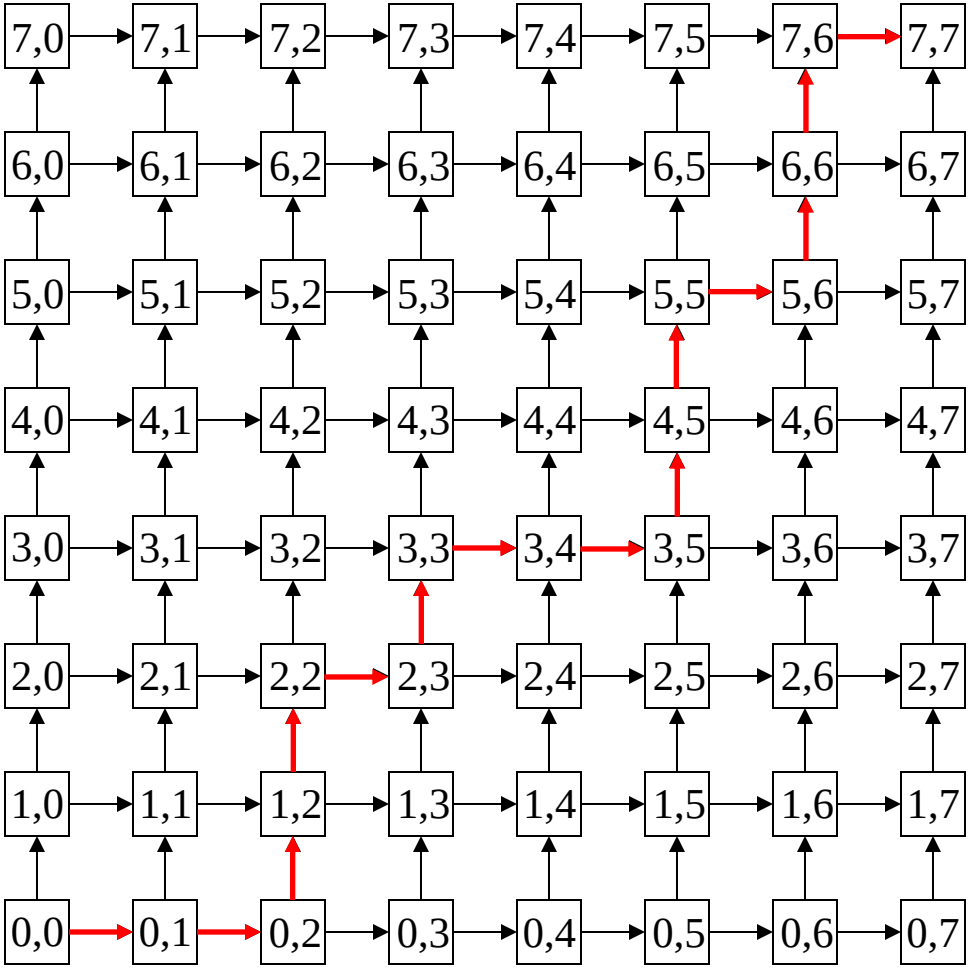
- **k** bits are stored using **n** cells
- A write is a change **0→1** or **1→0** of **one** of the k bits
- **Definition – Flash Codes**: An $(n,k,t)_q$ **Flash/Floating Code** is a coding scheme that accommodates **any sequence of up to t writes** of **k** bits, using **n q**-level cells, in such a way that a block erasure is never required
- **Goal**: Given **k**, **n**, **q** maximize the number of writes **t**

Flash Codes

Example: Storing **three bits** using **two 8-level cells**



Bits Diagram



Cells Diagram

Write Deficiency

- A trivial upper bound on the number of writes:

$$t \leq n(q-1)$$

- **Write Deficiency:** The difference between the trivial upper bound, $n(q-1)$, and the guaranteed number of writes t

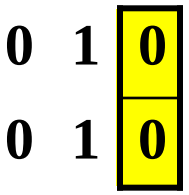
$$\delta = n(q-1) - t$$

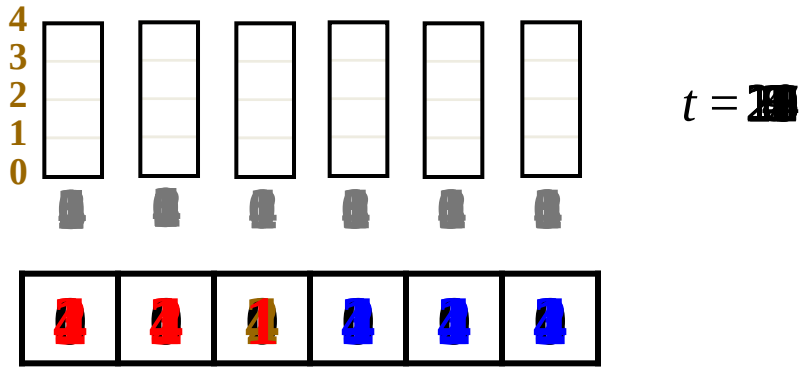
The write deficiency shows how close a given flash code is to the trivial upper bound

Example – Two Bits Construction

- Every cell is filled to the top before moving to the next one
- When the cells coincide, the last cell represents two bits. The cell's residue modulo 4 sets the bits value:

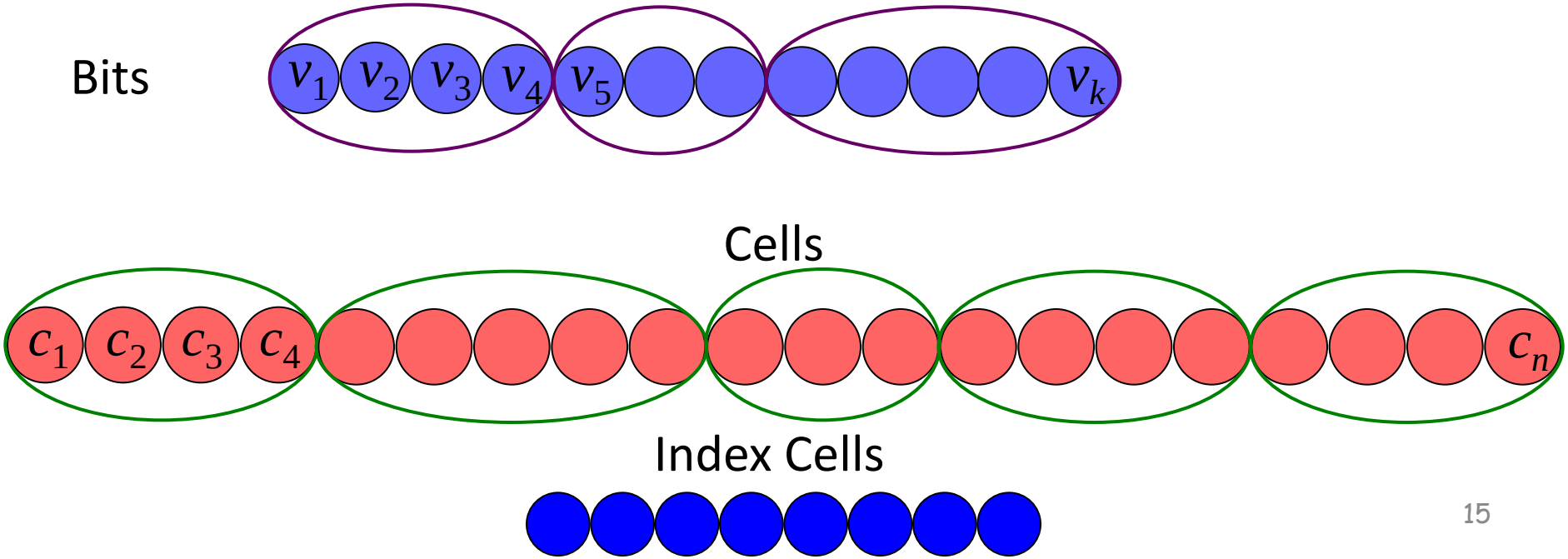
  0 – (0,0) 1 – (0,1) 2 – (1,0) 3 – (1,1)
- The maximum number of writes (worst case) is $n(q-1) - [(q-1)/2]$ (optimal) before erasing is required.





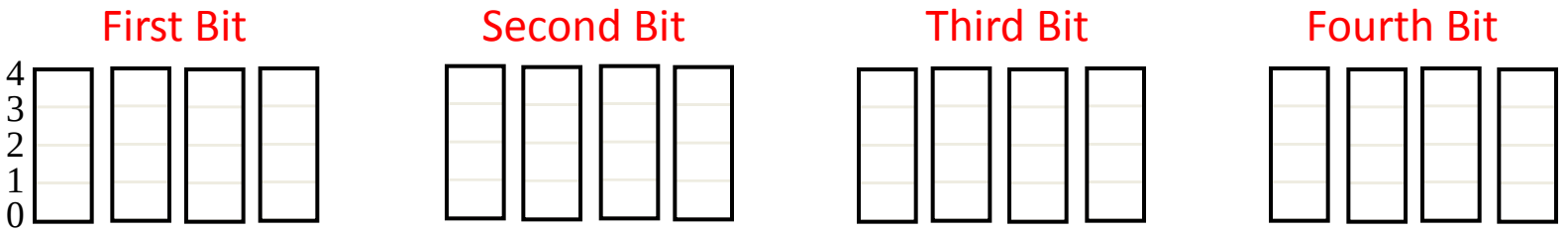
Indexed Flash Codes

- Partition the k bits into groups of k' bits
- Partition the cells into blocks of n' < n cells
- Each group of bits is represented by one of the blocks
- When a block gets full, start using the next empty block
- Indexing is needed to indicate for each block which bit-group it represents
- Leads to overhead due to index cells



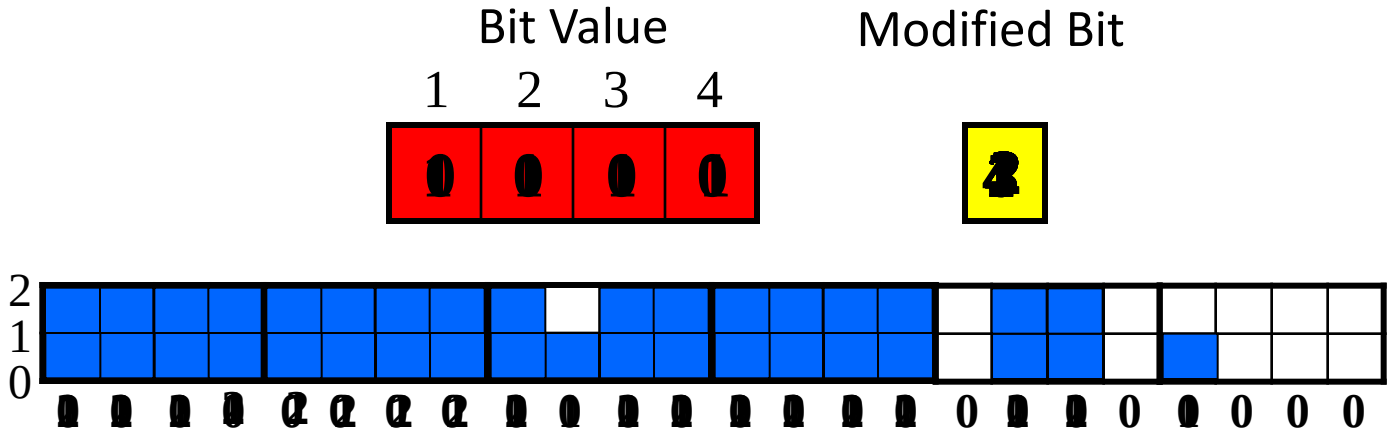
Index-Less Indexed Flash Codes

- Use the same idea but without the index cells
- How?
  - Each block consists of **k cells** and stores **one bit**
  - The blocks are used differently in such a way that it is possible to decode which bit each block stores
  - The bit value is the **parity** of its block
- **Example:** Storing 4 bits
  - The memory consists of **4-cell blocks**
  - Assume each cell has 5 levels: 0,1,2,3,4



Index-Less Indexed Flash Codes

Example: Writing 4 bits using 24 cells of 3 levels

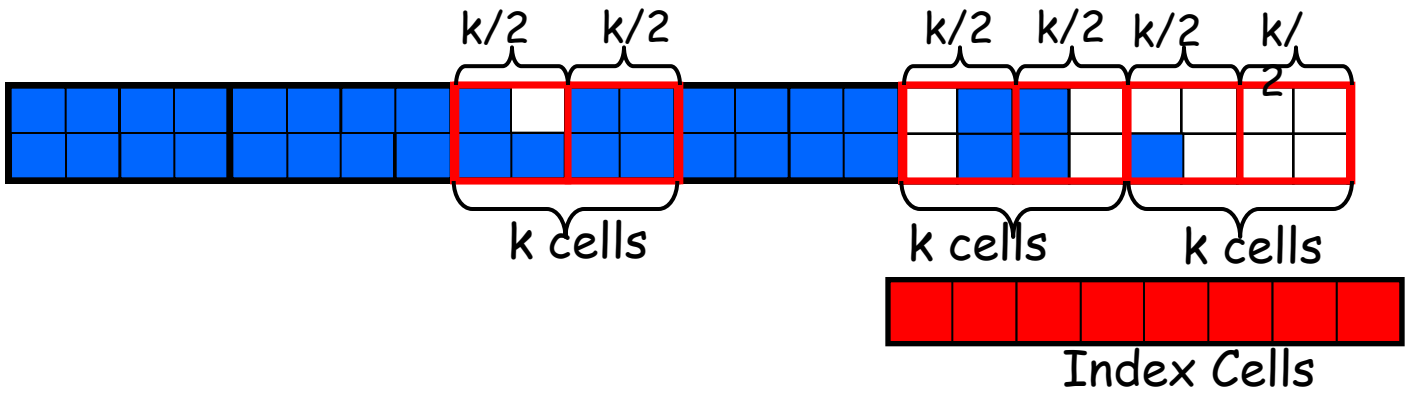


When writing stops, at most $(k-1)(k(q-1)-1)$ levels are not used

The write deficiency order is $\delta = O(k^2 q)$

Nearly Optimal Construction

- How to continue when writing stops?
  - Every block is divided into **two sub-blocks of k/2 cells**
  - Every sub-block stores one bit
  - It is not possible to write the bits as before
  - Now… use the **index cells!**



Nearly Optimal Construction

- How to continue when writing stops?
  - Every block is divided into **two sub-blocks of k/2 cells**
  - Every sub-block stores one bit
  - It is not possible to write the bits as before
  - Now… use the **index cells!**
  - For each sub-block, its bit index is stored in the index cells
  - Repeat the process recursively $\log_2 k$ steps
  - At each step, at most **2k – 2** sub-blocks need to be indexed
- **Theorem**: The write deficiency order is

$$\delta = O(qk\log^2 k / \log q)$$