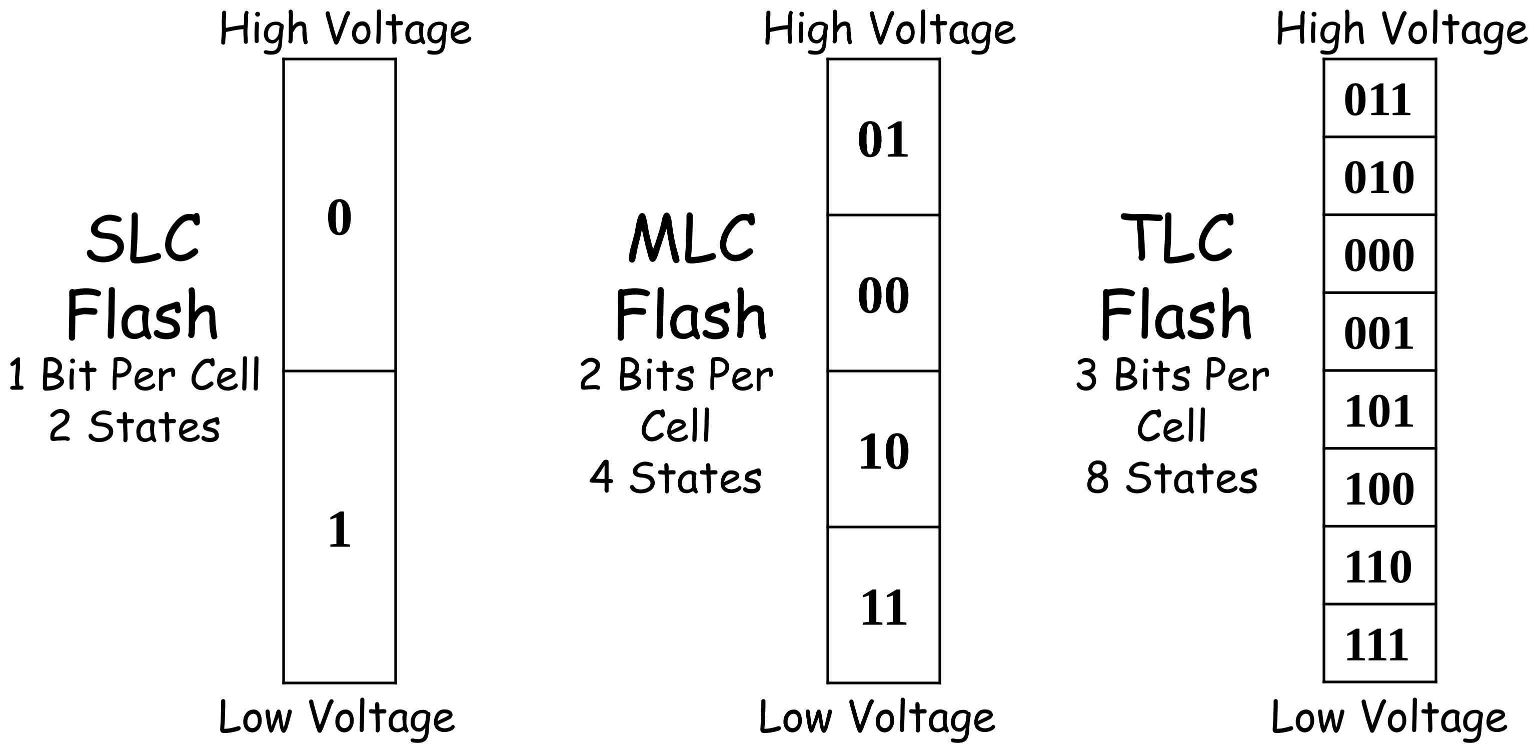
048704/236803
Seminar on Coding for Non-Volatile Memories

SLC, MLC and TLC Flash

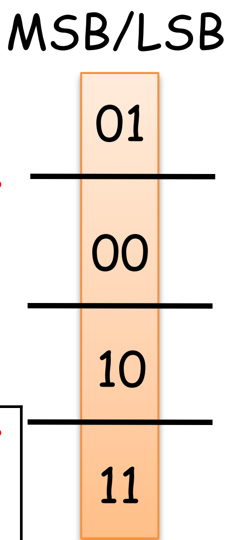


Flash Memory Structure

- A group of cells constitute a page
- A group of pages constitute a block
  - In SLC flash, a typical block layout is as follows

page 0	page 1
page 2	page 3
page 4	page 5
. . .	. . .
page 62	page 63

Flash Memory Structure



- In MLC flash the two bits within a cell **DO NOT** belong to the same page – **MSB page** and **LSB page**
- Given a group of cells, all the MSB's constitute one page and all the LSB's constitute another page

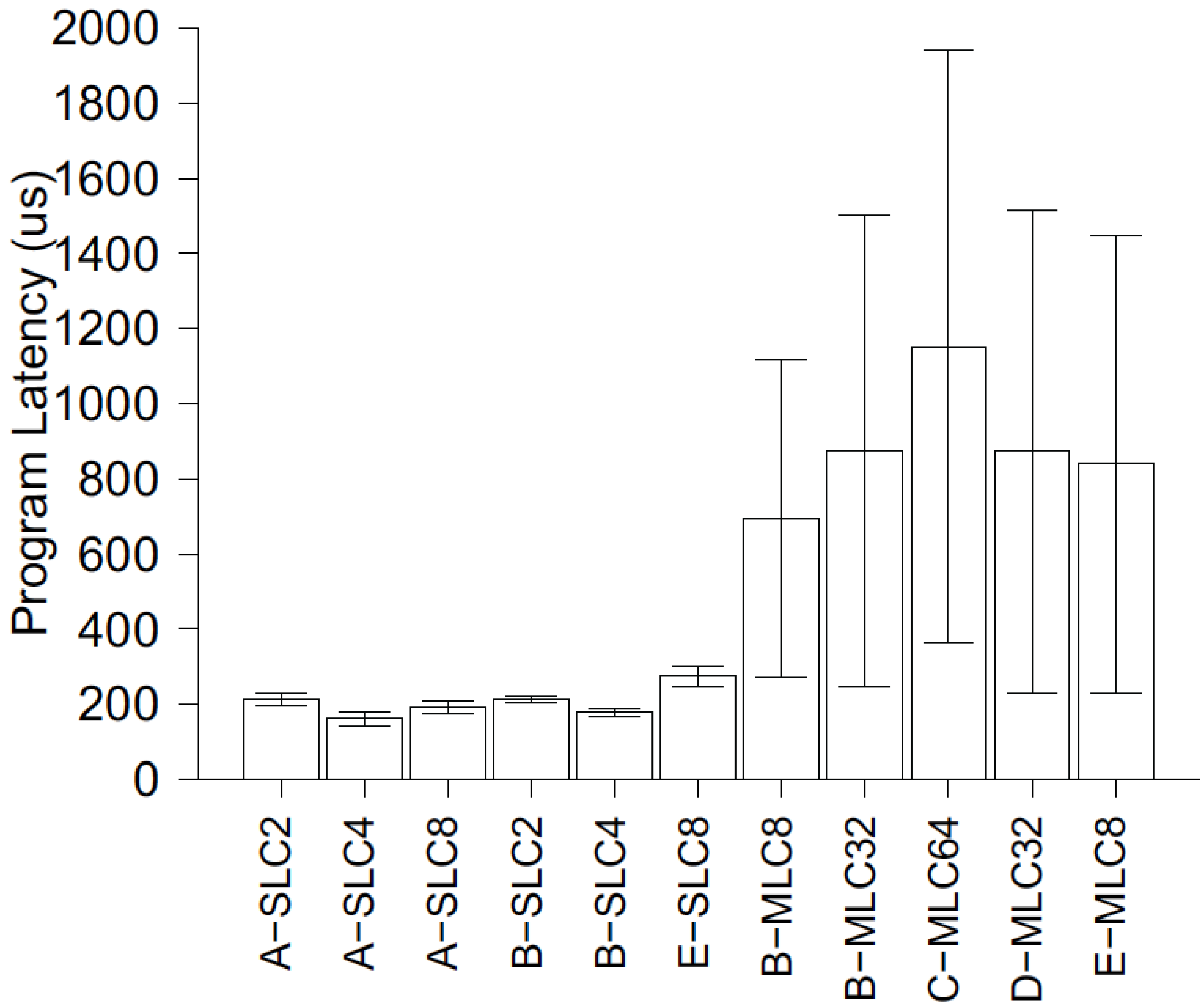
Row index	**MSB** of **first** 2^{14} cells	**LSB** of **first** 2^{14} cells	**MSB** of **last** 2^{14} cells	**LSB** of **last** 2^{14} cells
0	page 0	page 4	page 1	page 5
1	page 2	page 8	page 3	page 9
2	page 6	page 12	page 7	page 13
3	page 10	page 16	page 11	page 17
⋮	⋮	⋮	⋮	⋮
30	page 118	page 124	page 119	page 125
31	page 122	page 126	page 123	page 127

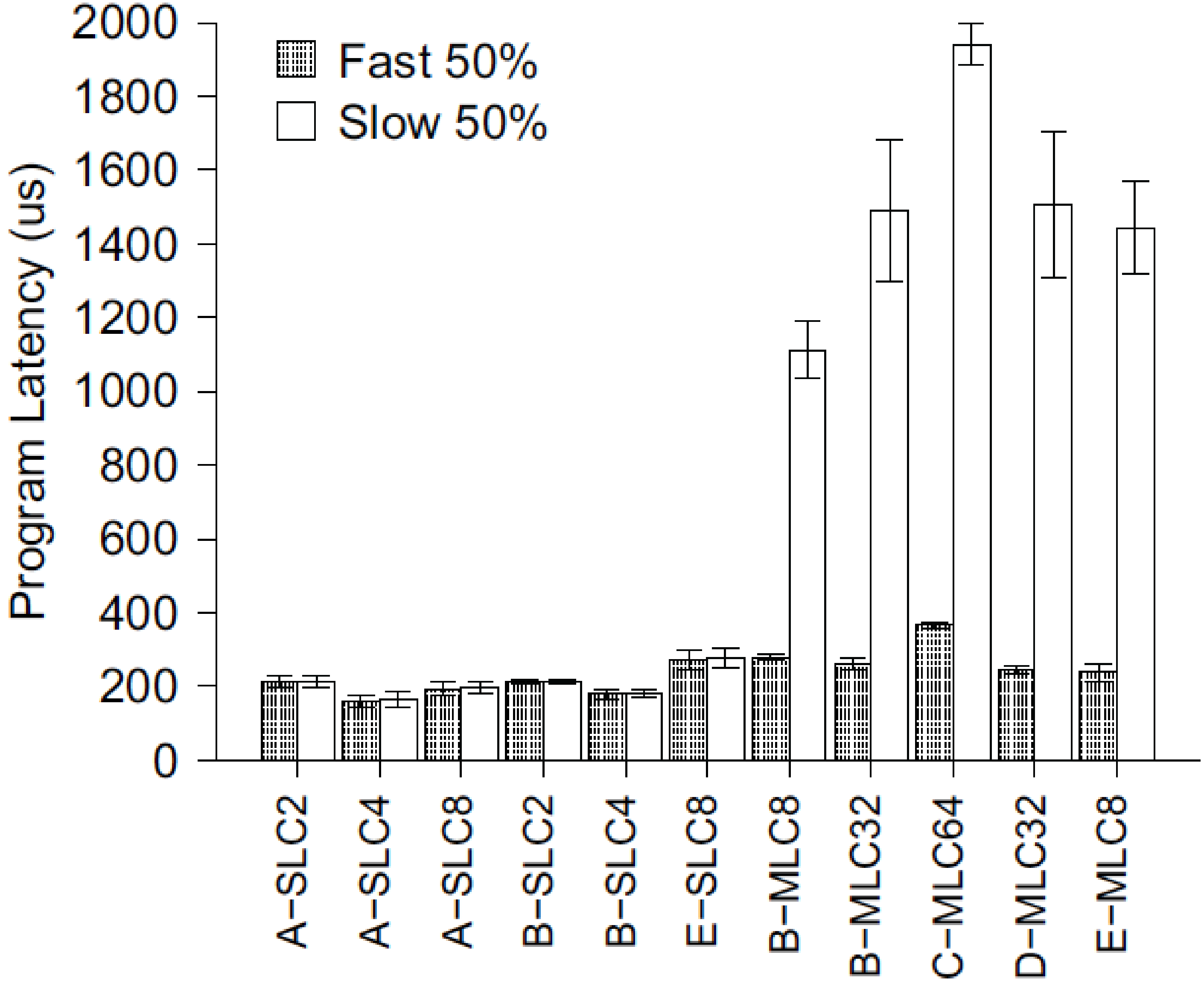
Flash Memory Structure

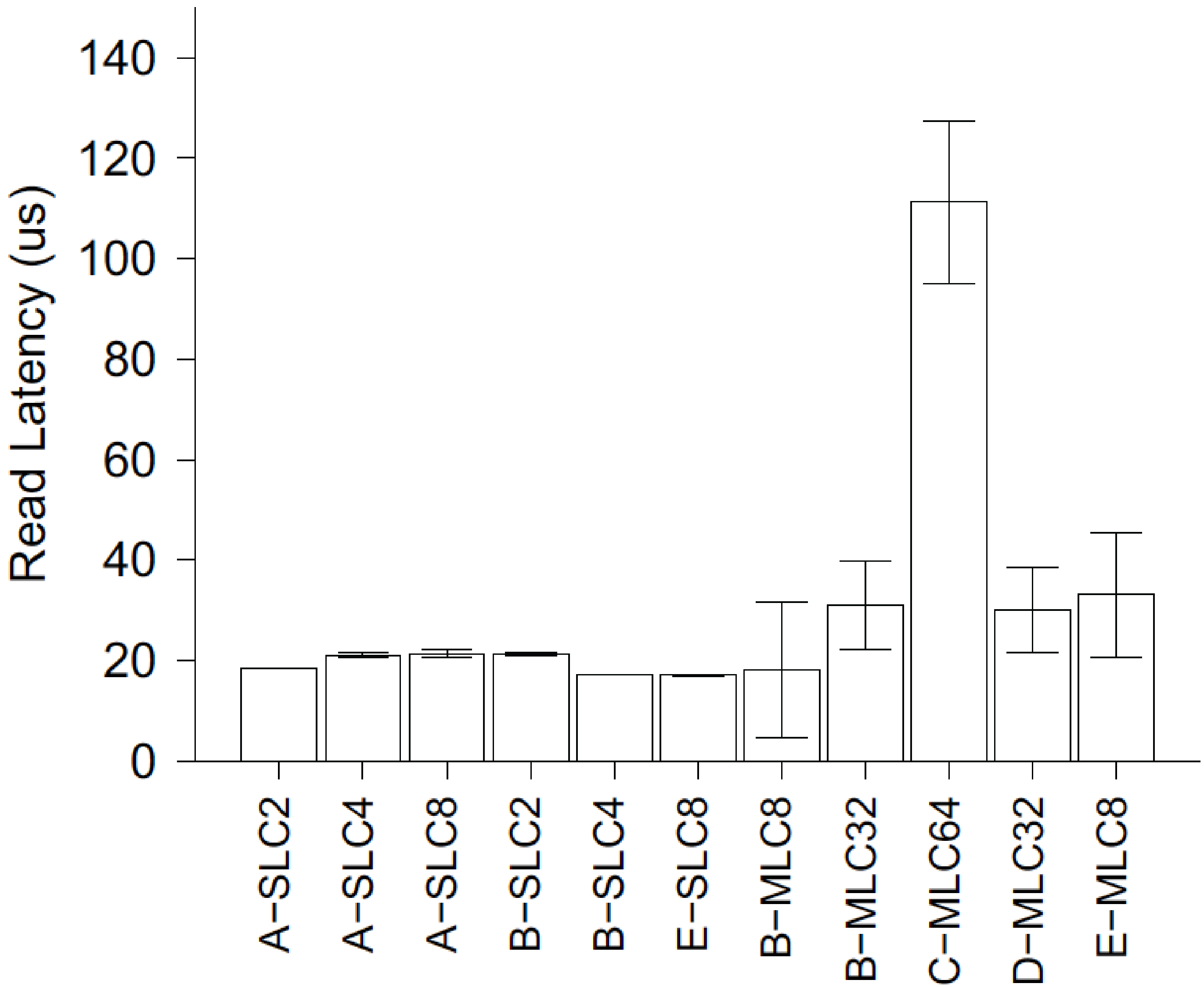
	MSB Page	CSB Page	LSB Page	MSB Page	CSB Page	LSB Page
Row index	**MSB** of **first** 2^{16} cells	**CSB** of **first** 2^{16} cells	**LSB** of **first** 2^{16} cells	**MSB** of **last** 2^{16} cells	**CSB** of **last** 2^{16} cells	**LSB** of **last** 2^{16} cells
0	page 0			page 1		
1	page 2	page 6	page 12	page 3	page 7	page 13
2	page 4	page 10	page 18	page 5	page 11	page 19
3	page 8	page 16	page 24	page 9	page 17	page 25
4	page 14	page 22	page 30	page 15	page 23	page 31
⋮	⋮		⋮	⋮		⋮
62	page 362	page 370	page 378	page 363	page 371	page 379
63	page 368	page 376		page 369	page 377	
64	page 374	page 382		page 375	page 383	
65	page 380			page 381		

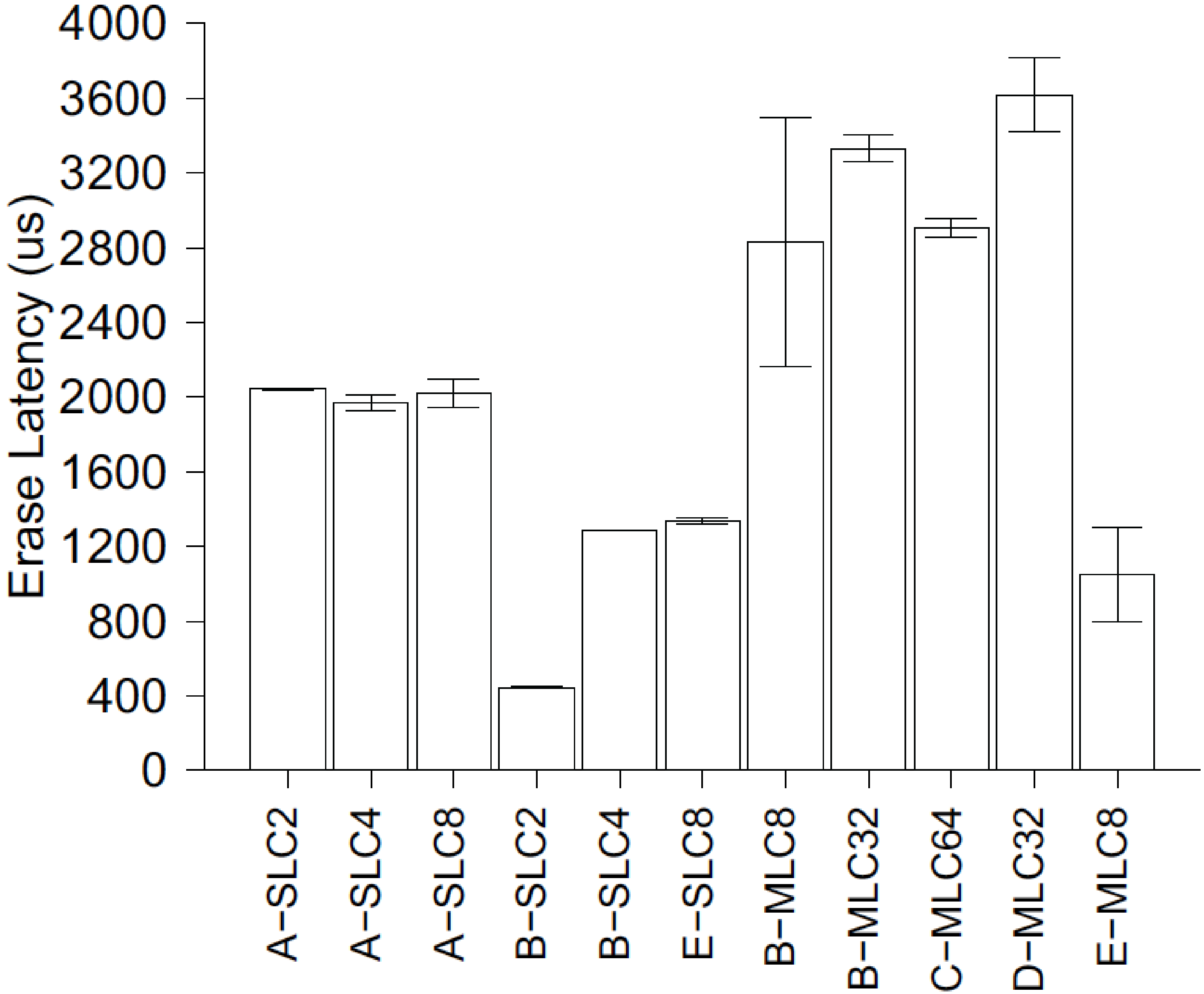
Flash Memory Structure

- Why to split cell bits to different pages?
  - Fast writing
  - Fast reading
  - Reduces the BER
  - Reduces the inter-cell interference (ICI)
- Side effects
  - Different BER to different pages
  - Different writing and reading times for different pages
  - Pages can affect other ones even if they don't share related information


Program Latency (us)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
A−SLC2
A−SLC4
A−SLC8
B−SLC2
B−SLC4
E−SLC8
B−MLC8
B−MLC32
C−MLC64
D−MLC32
E−MLC8

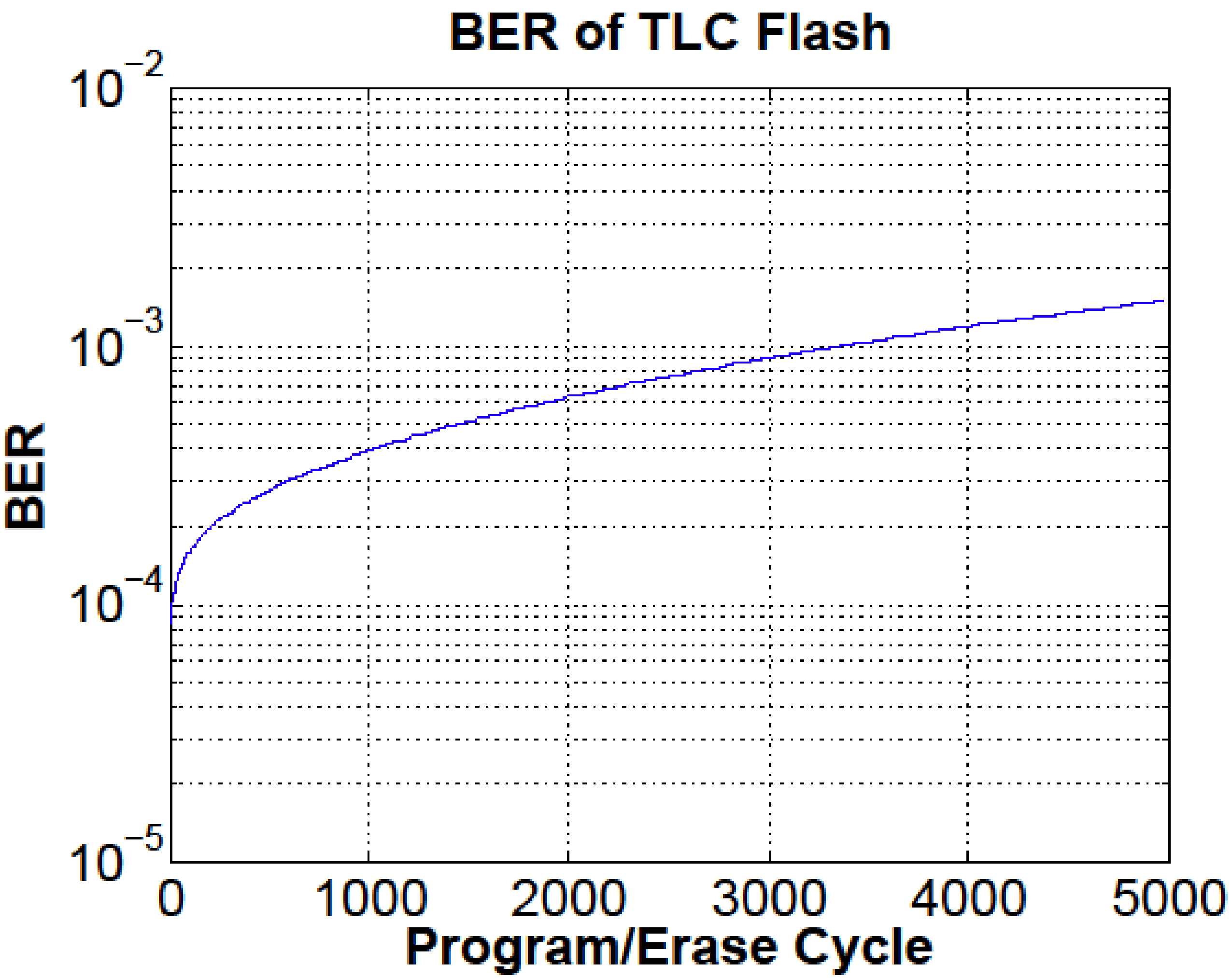

Program Latency (us)
2000
1800
1600
1400
1200
1000
800
600
400
200
0
Fast 50%
Slow 50%
A−SLC2
A−SLC4
A−SLC8
B−SLC2
B−SLC4
E−SLC8
B−MLC8
B−MLC32
C−MLC64
D−MLC32
E−MLC8


Read Latency (us)
140
120
100
80
60
40
20
0
A−SLC2
A−SLC4
A−SLC8
B−SLC2
B−SLC4
E−SLC8
B−MLC8
B−MLC32
C−MLC64
D−MLC32
E−MLC8

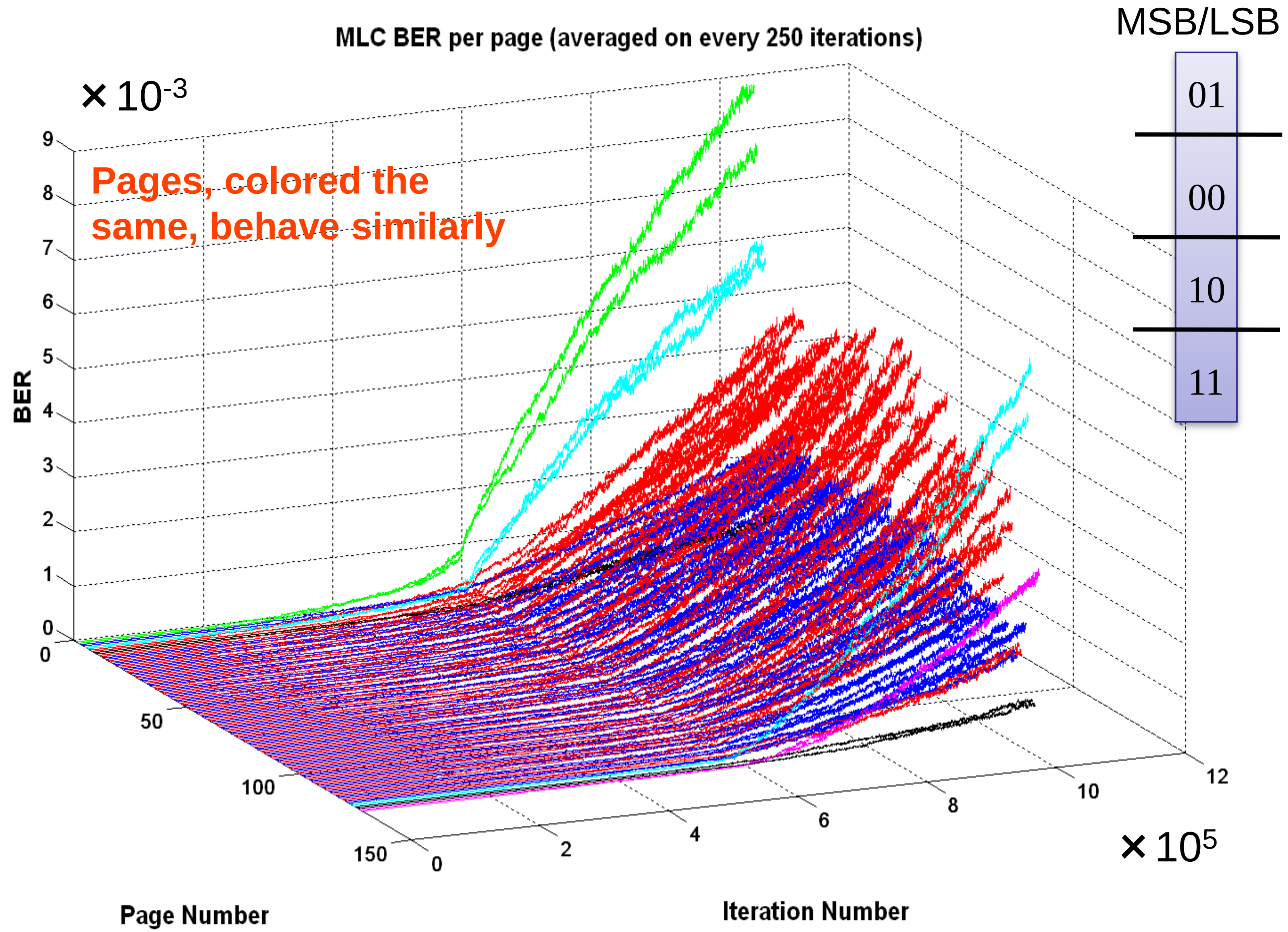


Erase Latency (us)
4000
3600
3200
2800
2400
2000
1600
1200
800
400
0
A−SLC2
A−SLC4
A−SLC8
B−SLC2
B−SLC4
E−SLC8
B−MLC8
B−MLC32
C−MLC64
D−MLC32
E−MLC8

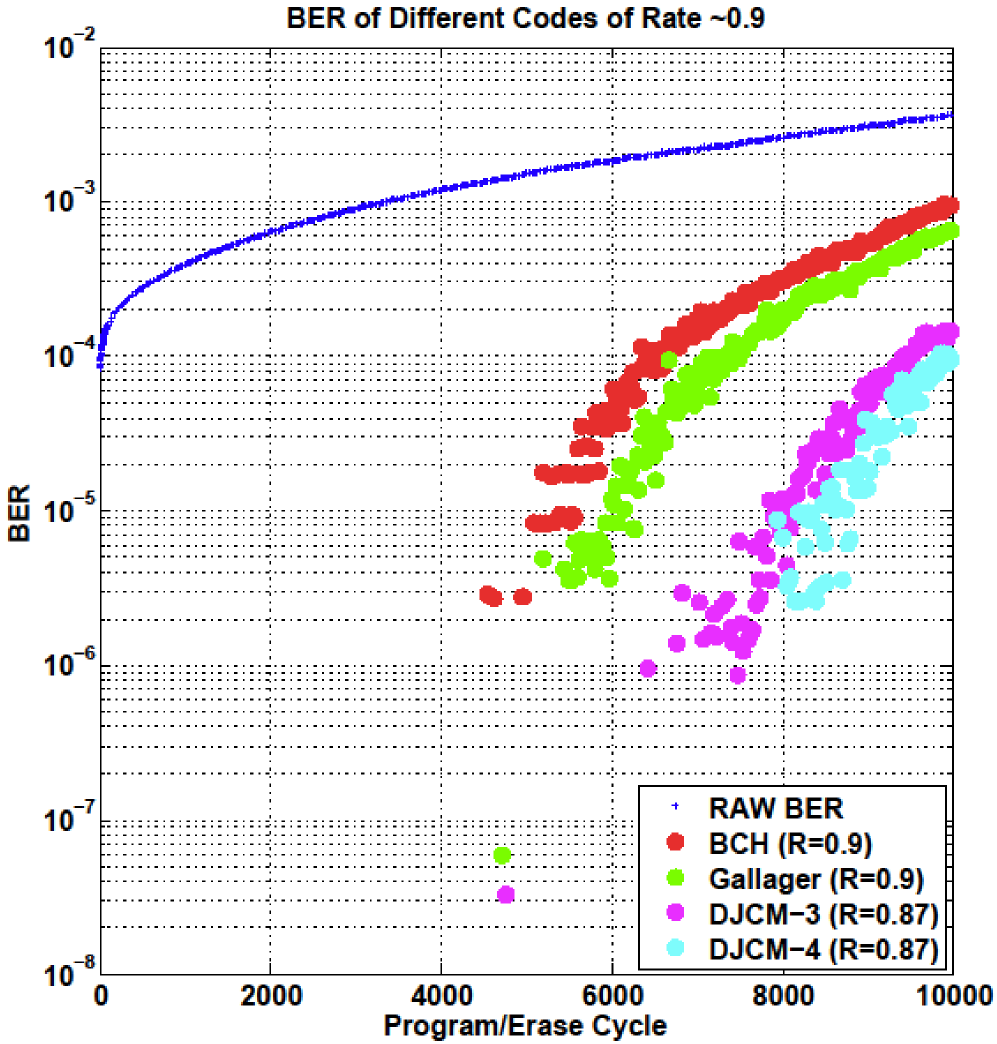
Raw BER Results



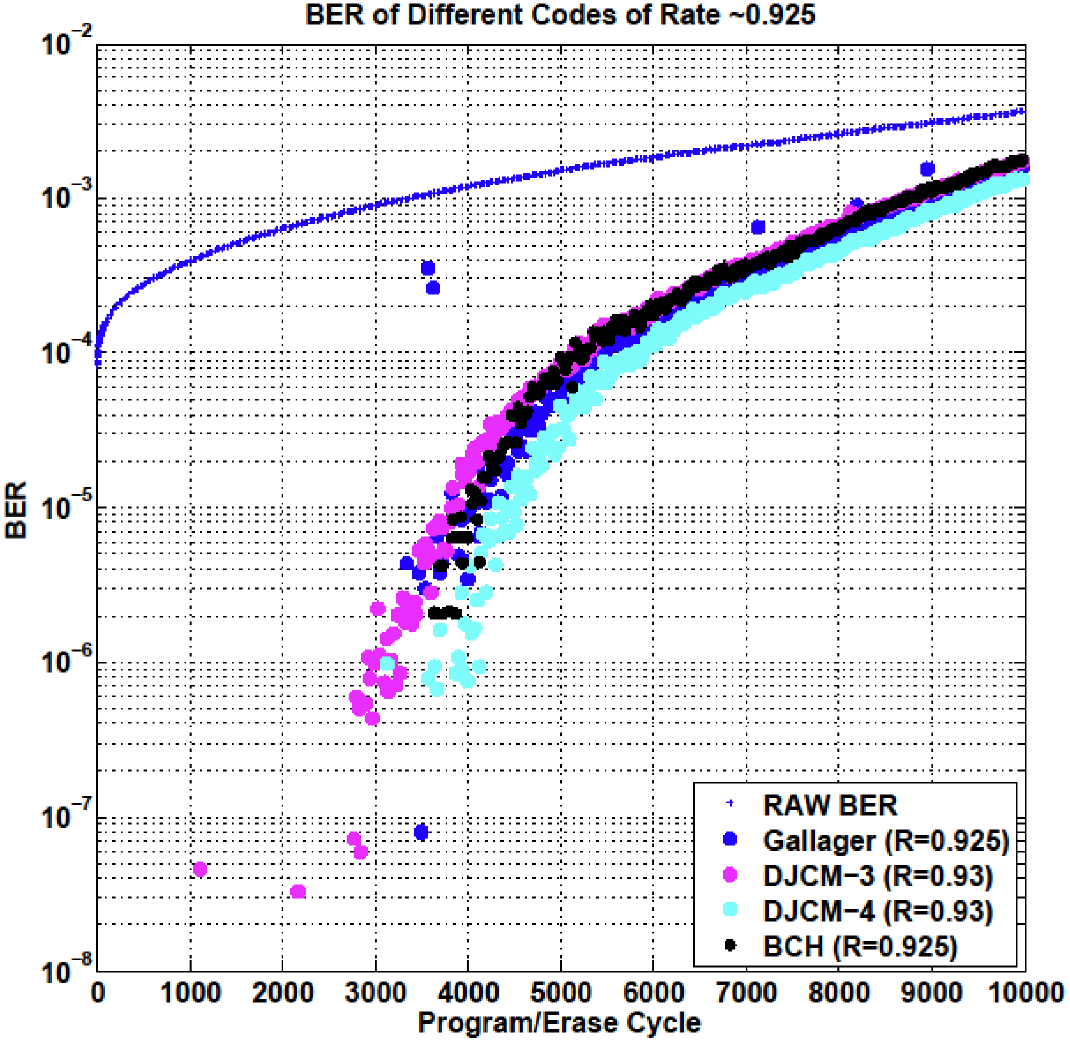
BER per page for MLC block



ECC Comparison $R \approx 0.9$

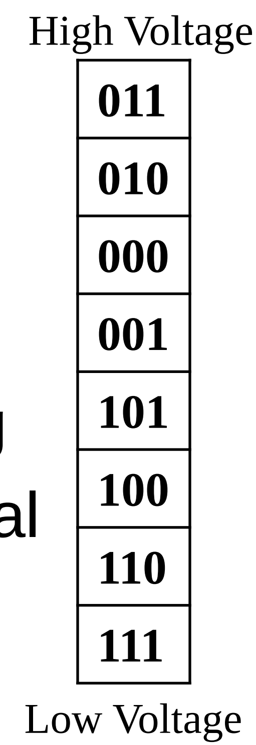


ECC Comparison $R \approx 0.925$

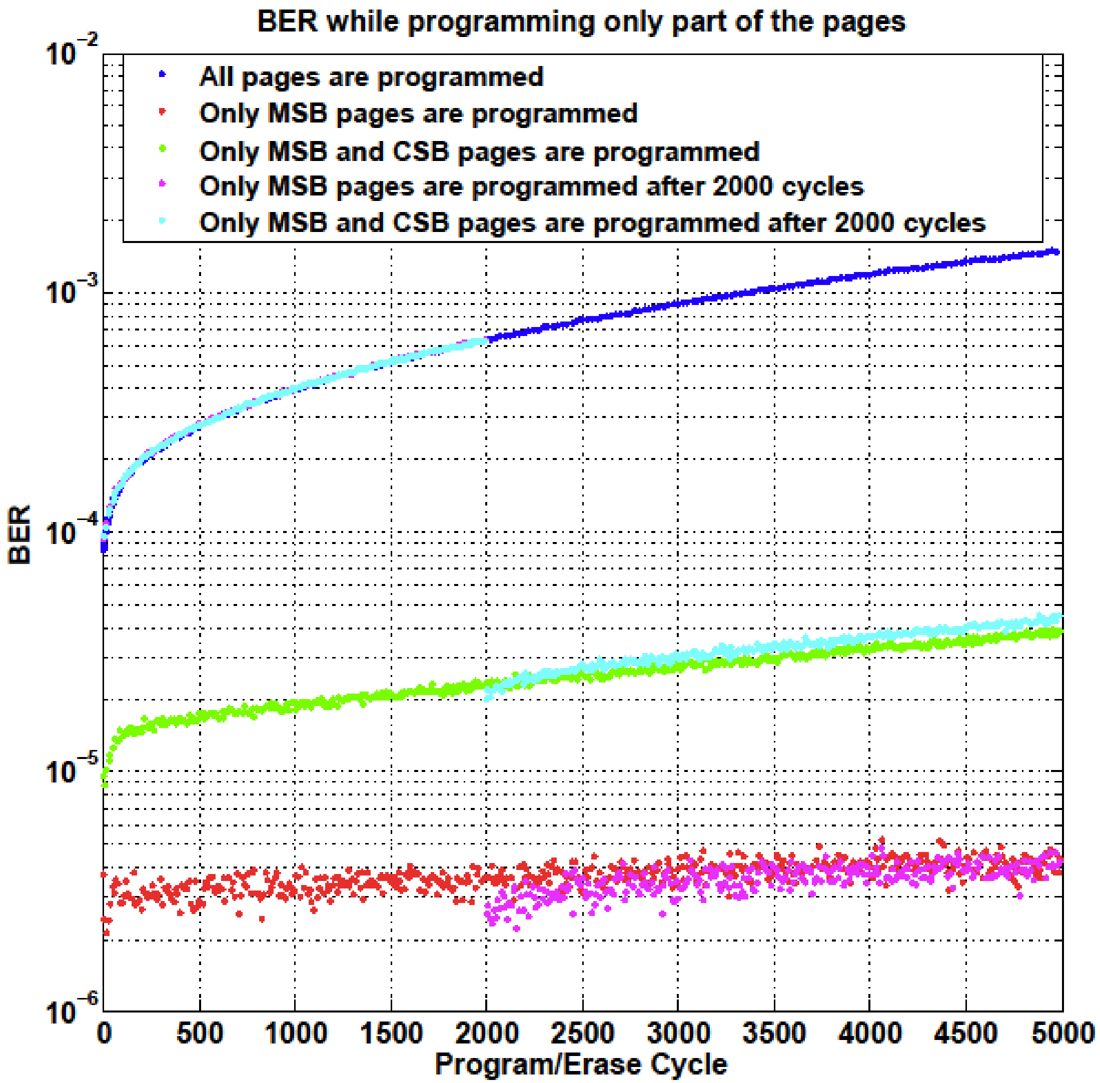


Partial Cell State Usage

- Store either one or two bits in every cell
  - For one bit, only the MSB pages
  - For two bits, only the MSB and CSB pages
- Two cases:
  - The partial storage is introduced at the beginning
  - The partial storage is introduced after 2000 normal program/erase cycles

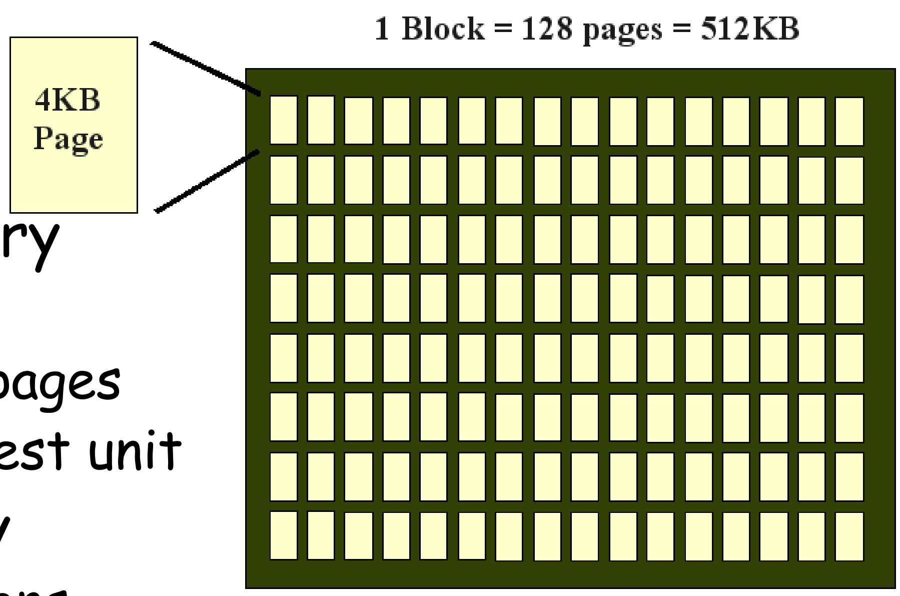


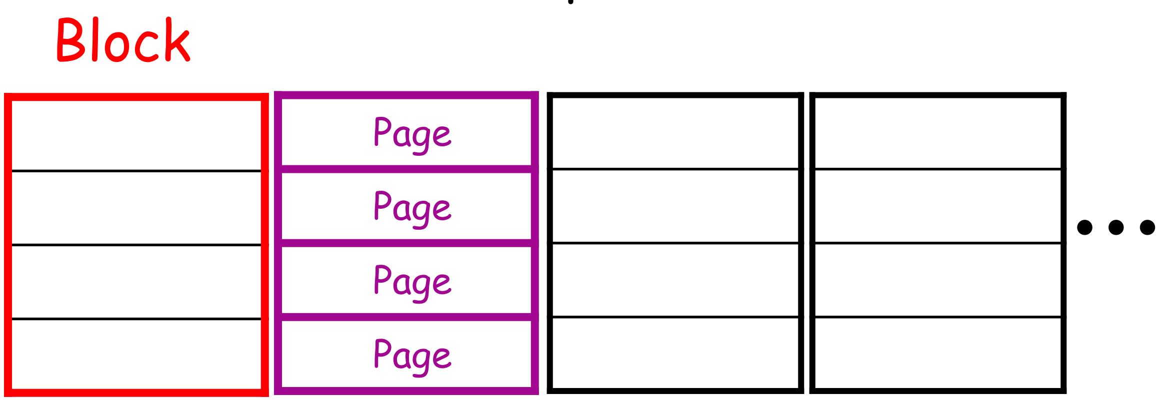
Partial Cell State Usage - BER



Organization of flash memory

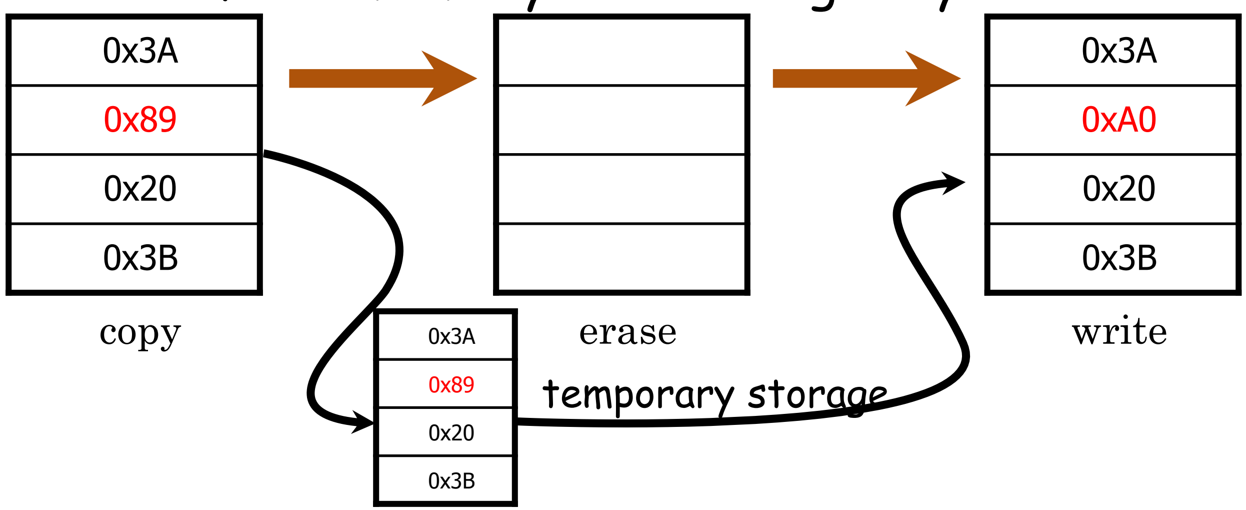
- Flash is a re-writable semiconductor memory
- Organization of flash memory
  - Contains thousands of blocks
  - A block contains typically 64 pages
  - A page is typically 4 KB, smallest unit
- Operations on flash memory
  - Page-level read/write operations
  - Block-level erase operations





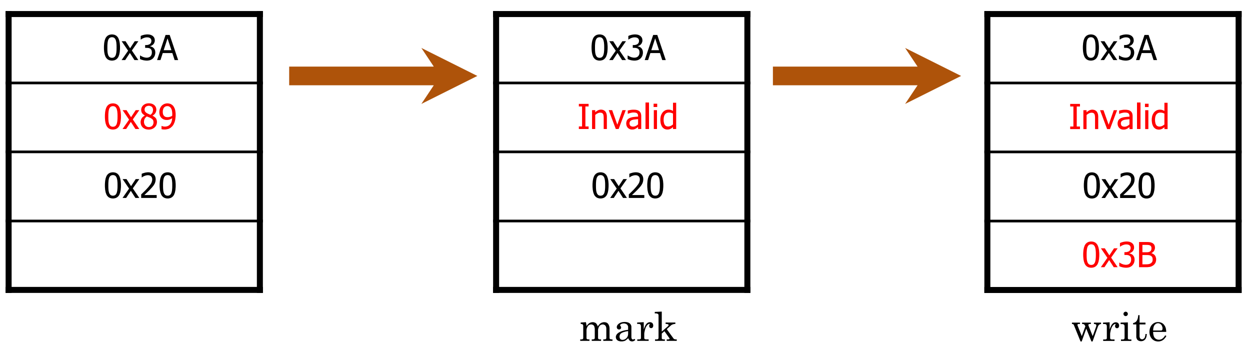
Flash memory: Two limitations

- Limitation 1: block erase
- Limitation 2: non-support of overwrite
- To change one page, must copy-erase-write
- "Write amplification" Changing one page requires 64 page writes!
- Undesirable:
  - reduces system performance
  - reduces flash memory device longevity



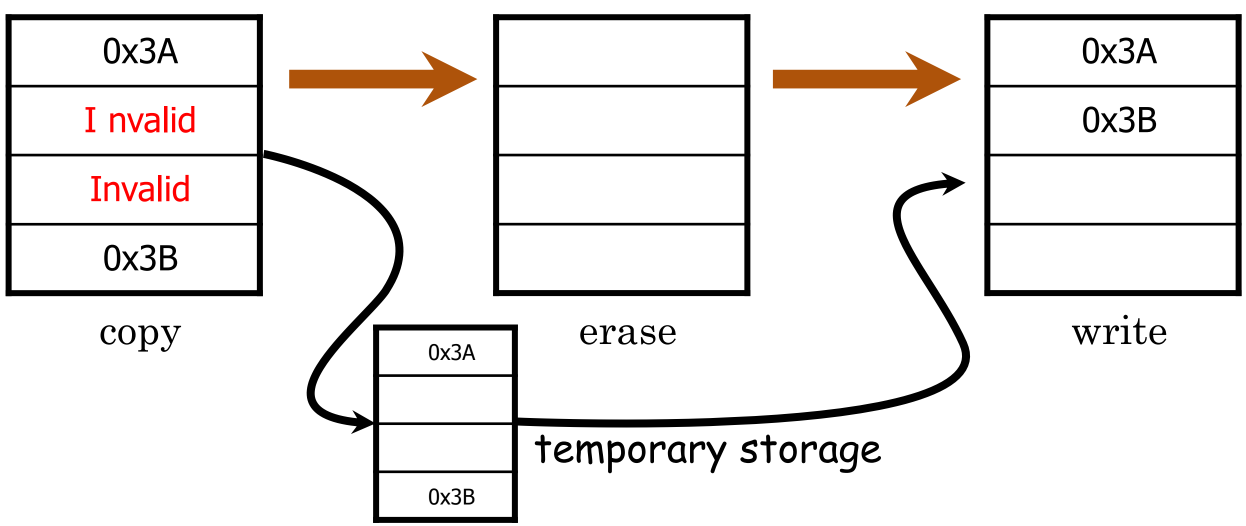
Flash memory: Out-of-place write

- Limitation 1: block erase
- Limitation 2: non-support of overwrite
- To change one page,
  - Mark the old page as invalid
  - Write the new data into a free page
- Invalid pages must be reclaimed



Flash memory: Garbage collection

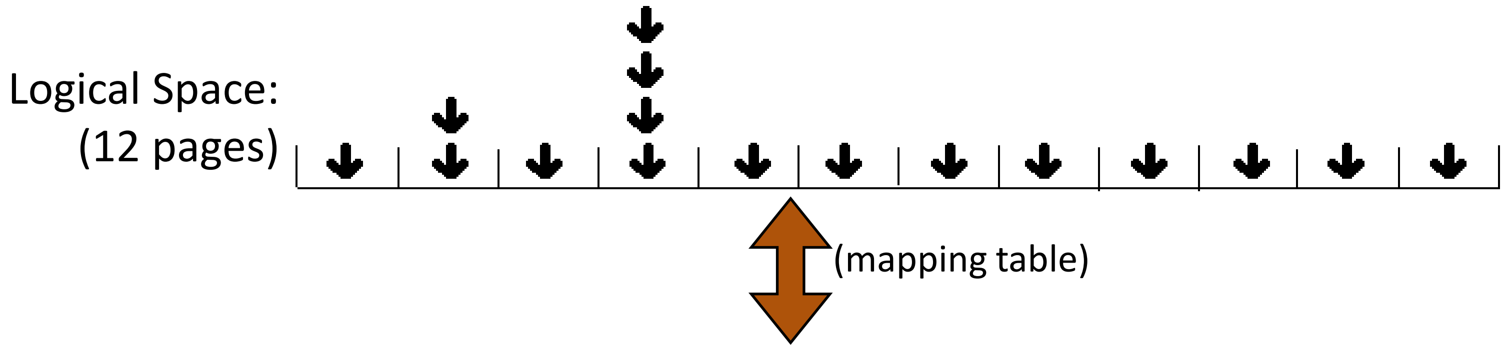
- Reclaim the invalid pages into free pages
- Steps:
  - Choose a block for garbage collection
  - Copy all valid pages out
  - Erase the block
  - Copy the valid pages back
- Causes undesired physical writes (write amplification)



Flash Transition Table (FTL)

- A table mapping the physical page of each logical page
- Table Size = TS = # logical pages*log(#physical pages)
  Z= # pages in a block
- Example:
  - Flash storage of 32GB with 64 pages of 2KB per block
  - #logical pages = 32GB/2KB = 2^{24}
  - If #logical pages = #physical pages, then
    TS = 2^{24}*log(2^{24})b = 2^{24}*3B = 48MB
- The table needs to be saved in the flash when power is down and rebuilt again when power is on

System
Example of Writing Flash Memory



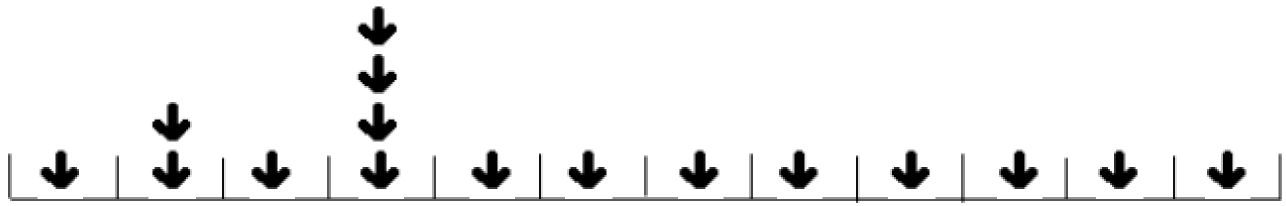
Physical Space:
(16 pages in 4 blocks)

Valid	Invalid	Valid	Valid
Invalid	Valid	Valid	Valid
Valid	Valid	Invalid	Valid
Valid	Invalid	Valid	Valid

Initial condition: Start with an empty memory
User writes uniformly and randomly distributed on user space
stationary condition: Logical memory is always full (worst case)

Garbage Collection

Logical Space:
(12 pages)



Physical Space:
(16 pages in 4 blocks)

Valid	Invalid	Valid	Valid
Invalid	Valid	Valid	Valid
Valid	Valid	Invalid	Valid
Valid	Invalid	Valid	Valid

Time to erase

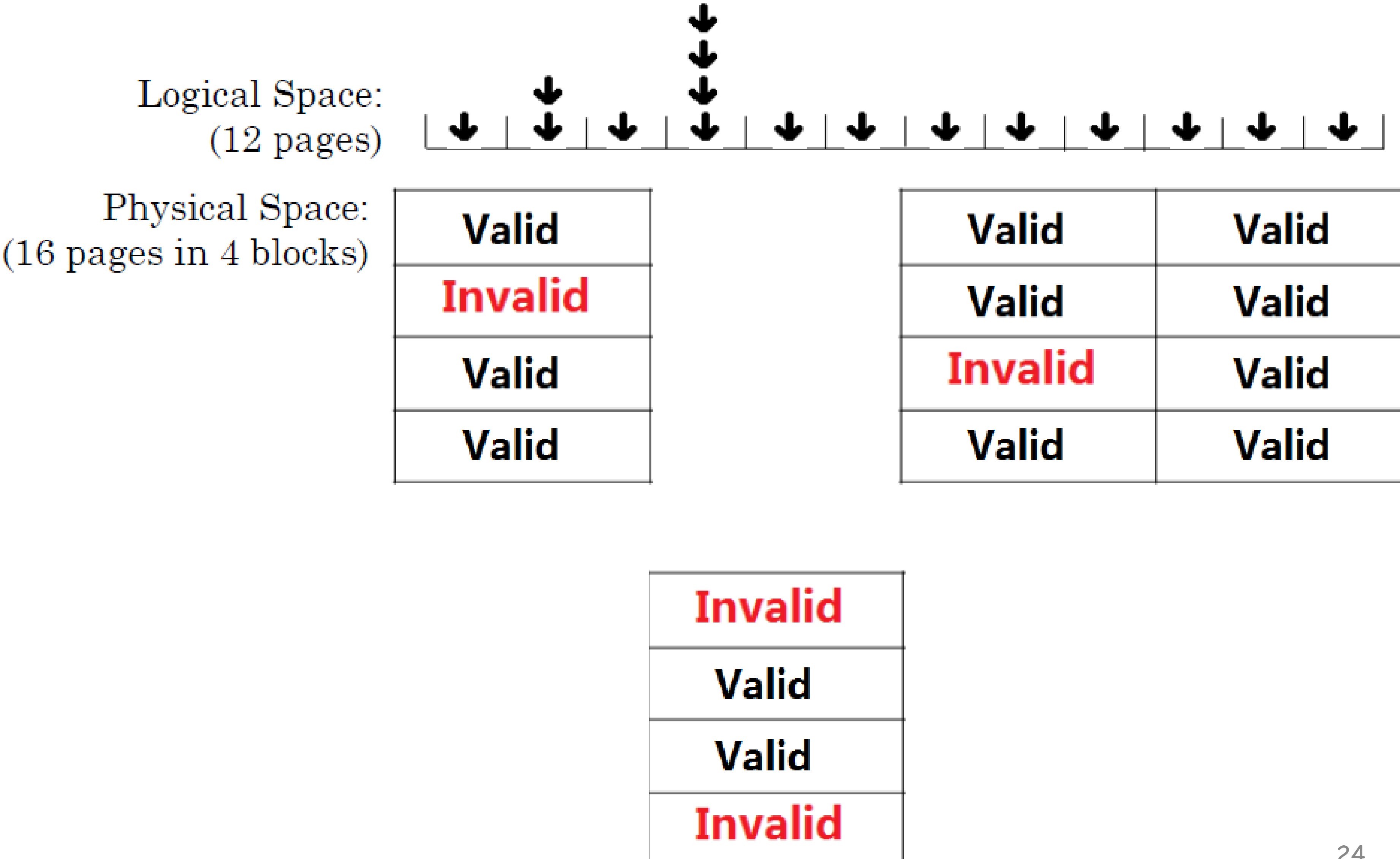
Greedy Garbage collection:

- Block with most invalid pages

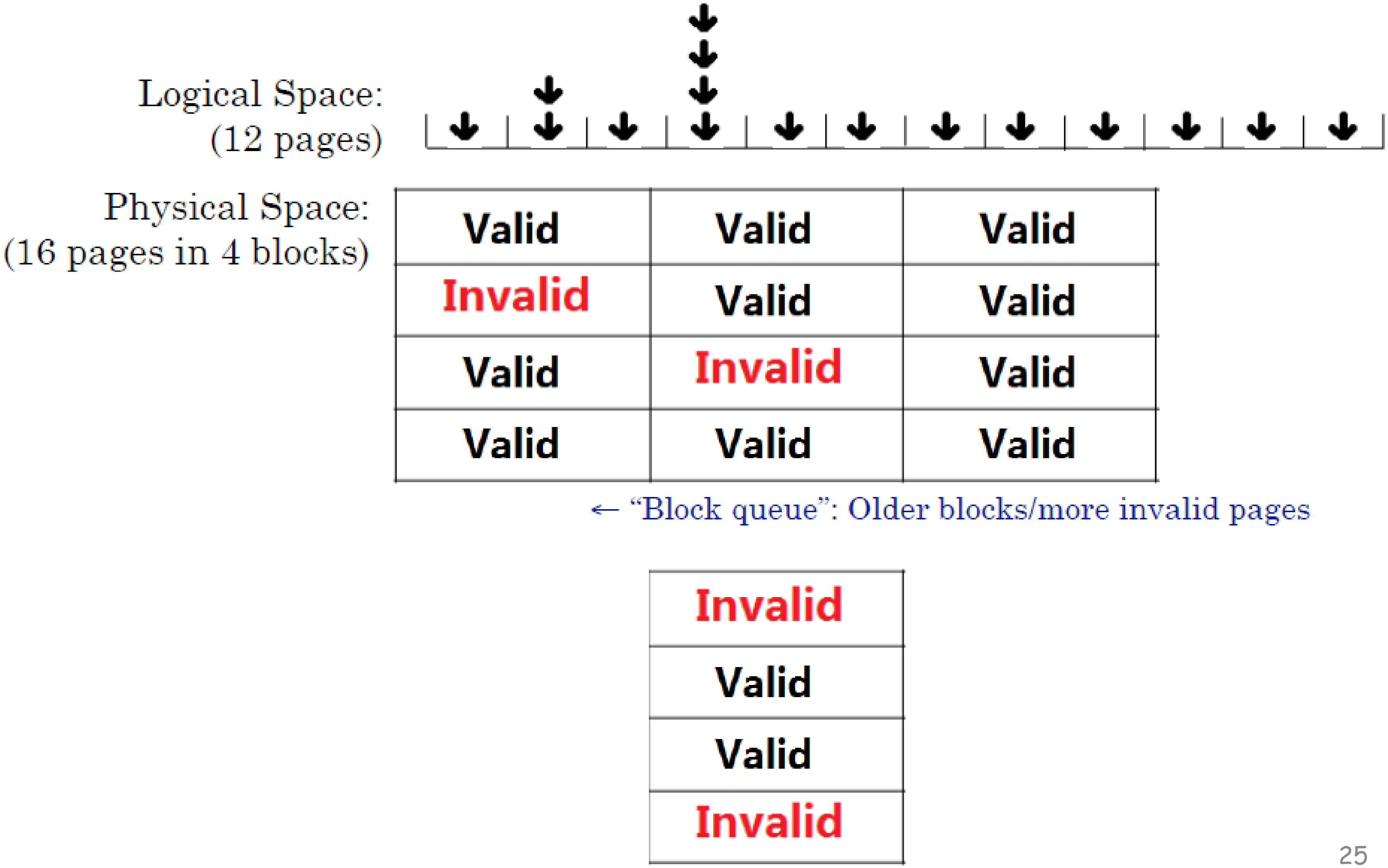
Only two writes needed

System

Garbage Collection

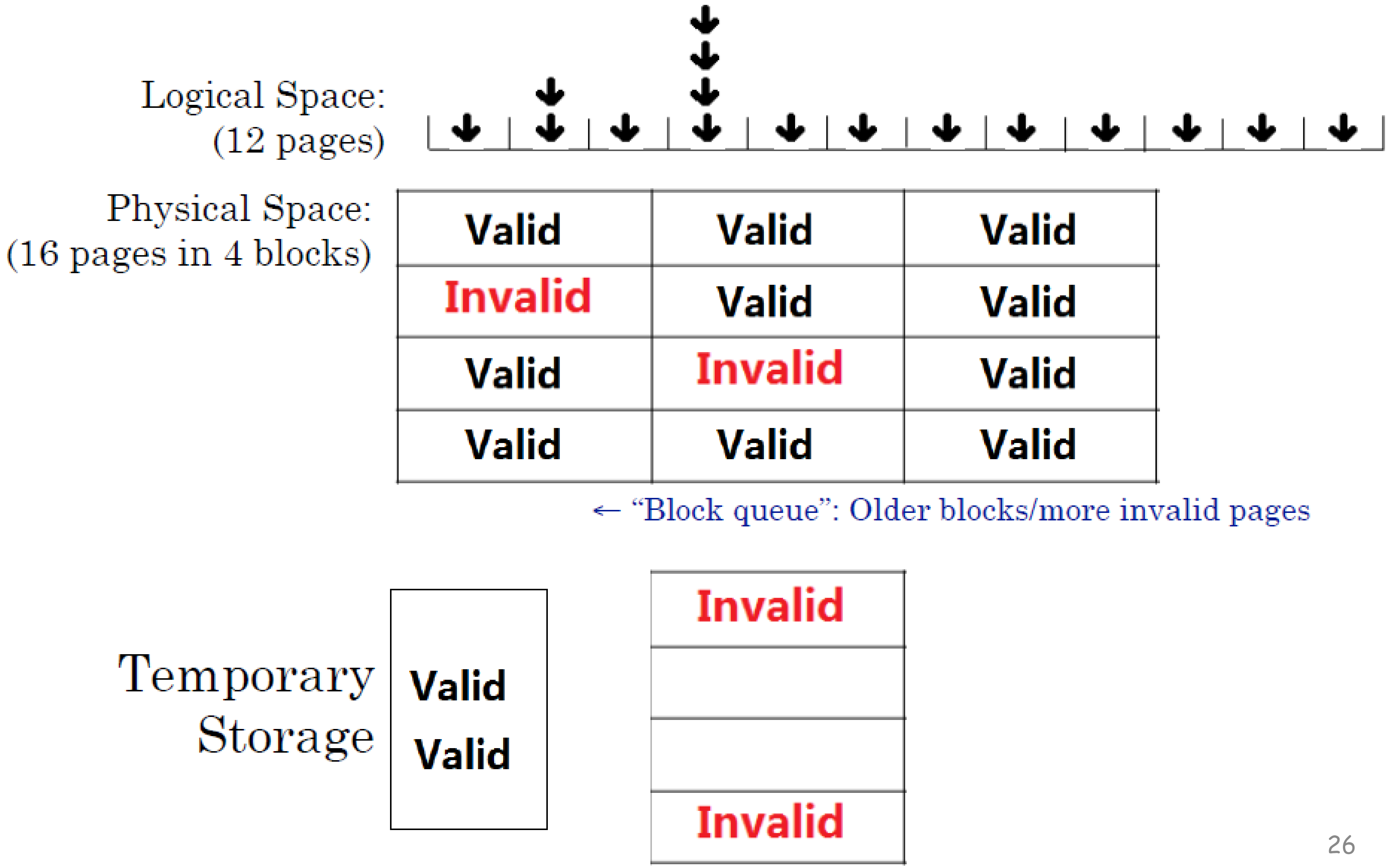


Garbage Collection



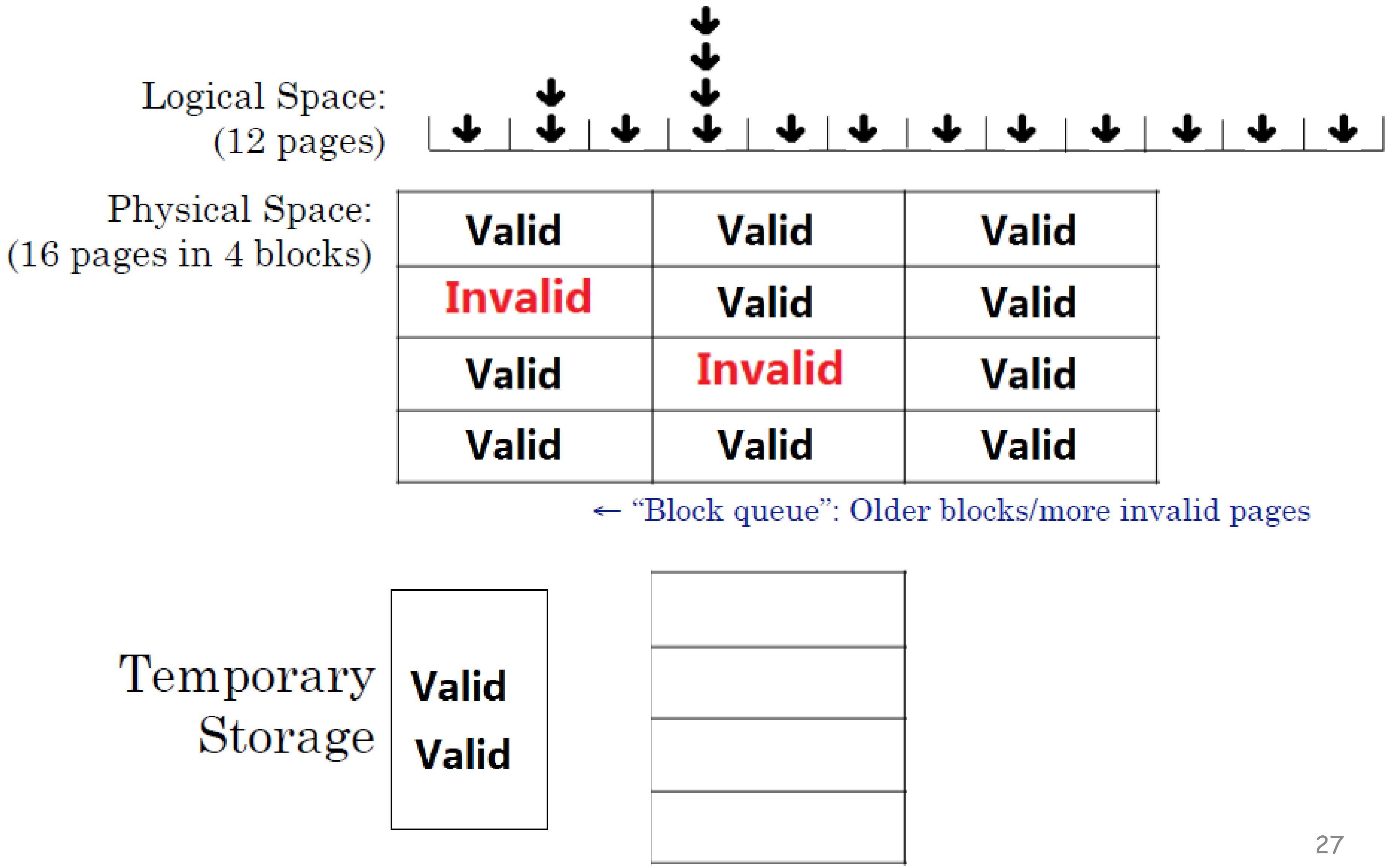
System

Garbage Collection



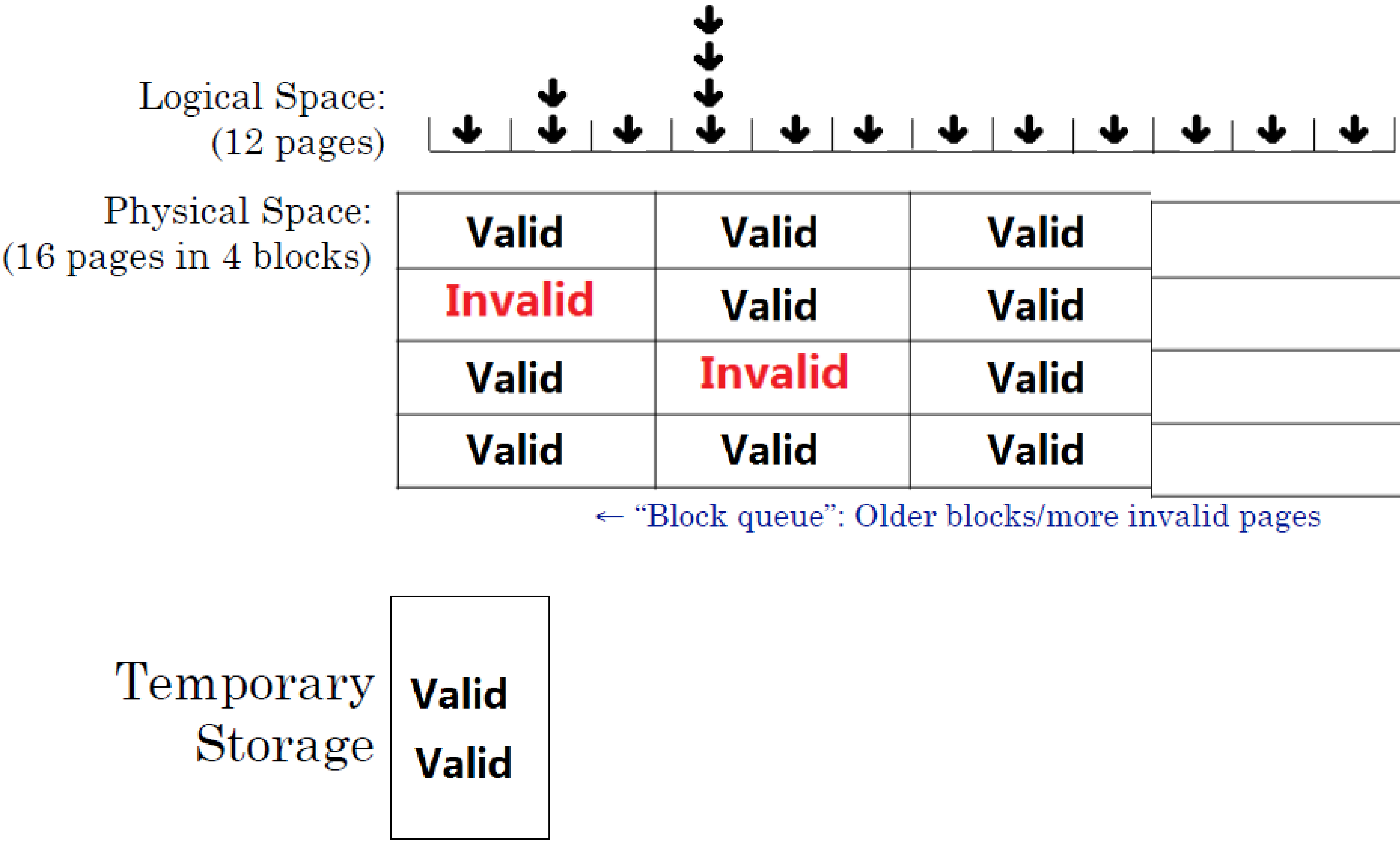
System

Garbage Collection



System

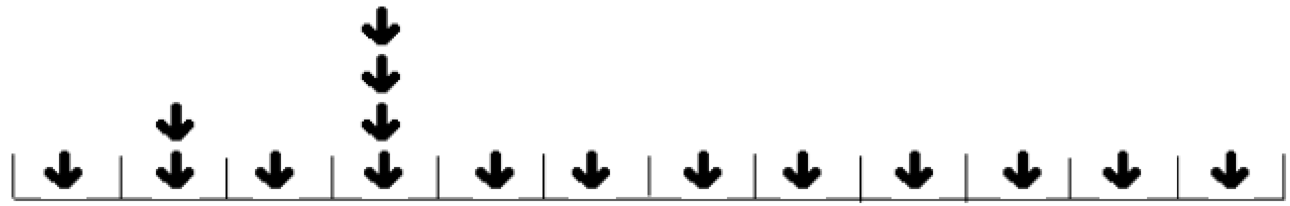
Garbage Collection



System

Garbage Collection

Logical Space:
(12 pages)



Physical Space:
(16 pages in 4 blocks)

Valid	Valid	Valid	
Invalid	Valid	Valid	Valid
Valid	Invalid	Valid	Valid
Valid	Valid	Valid	

Temporary Storage

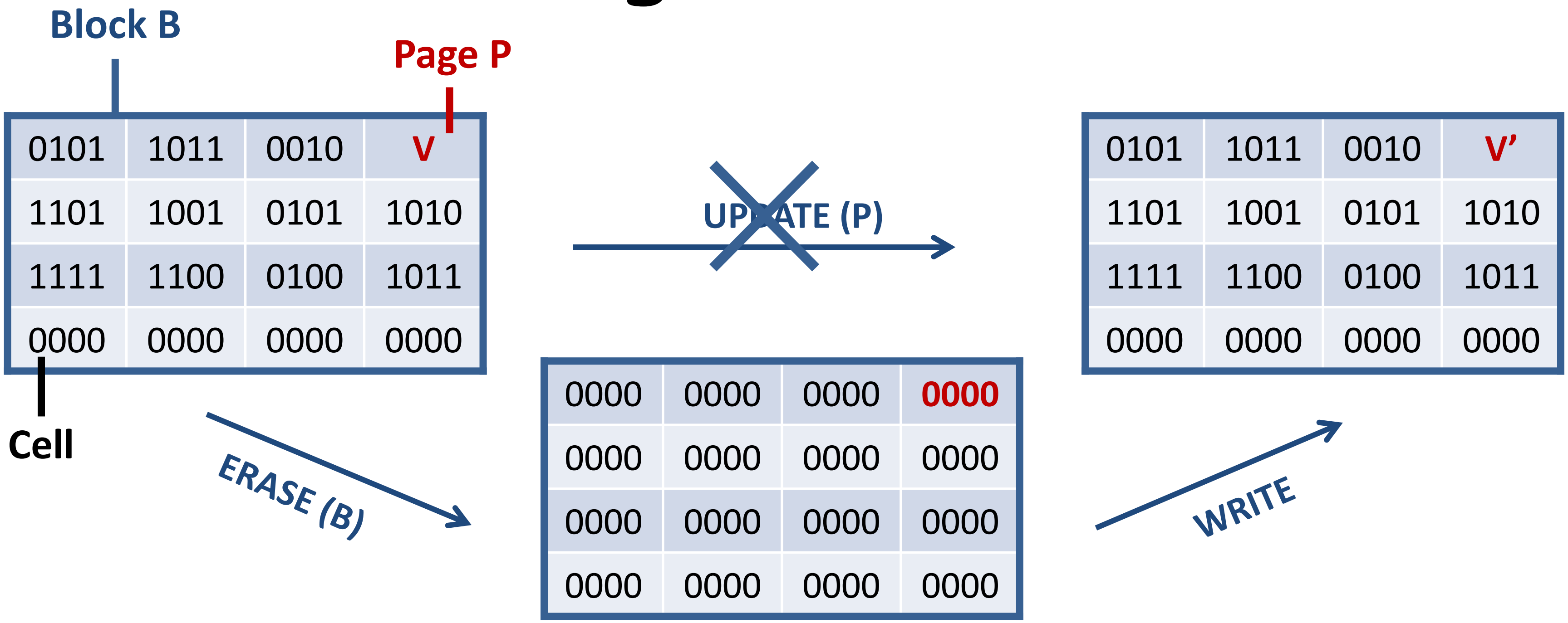
Time to erase

Greedy Garbage collection:

- Block with most invalid pages

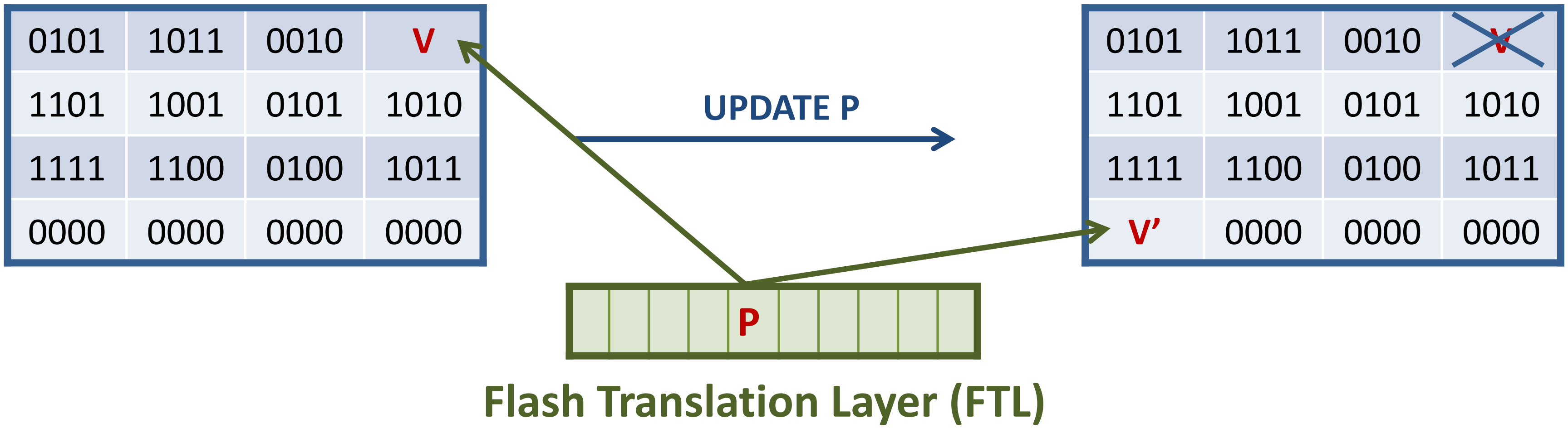
Only two writes needed

Garbage Collection

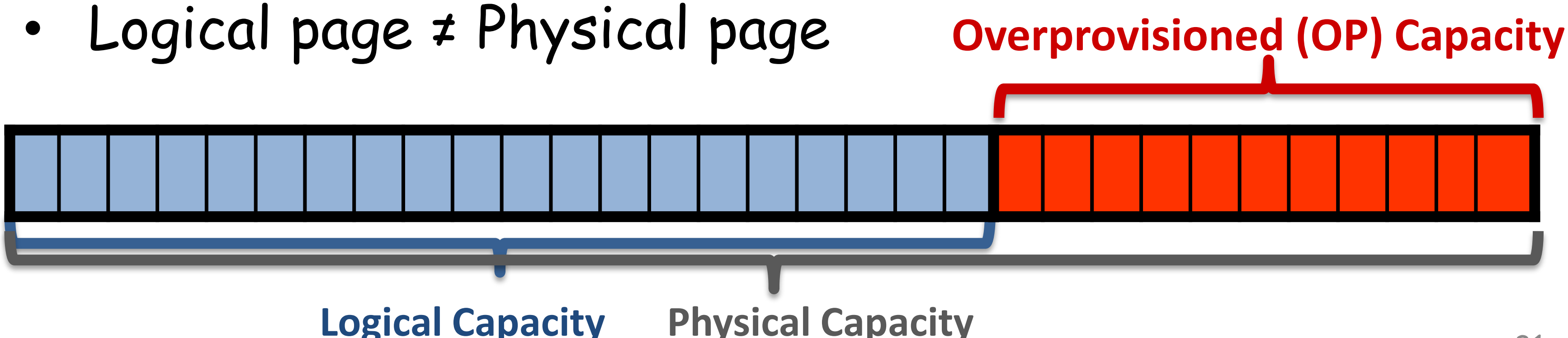


- Read/write pages quickly
- Erase blocks slowly
- Erase before write (no updates)

Garbage Collection



- **Out of place writes** replace updates
- Logical page ≠ Physical page

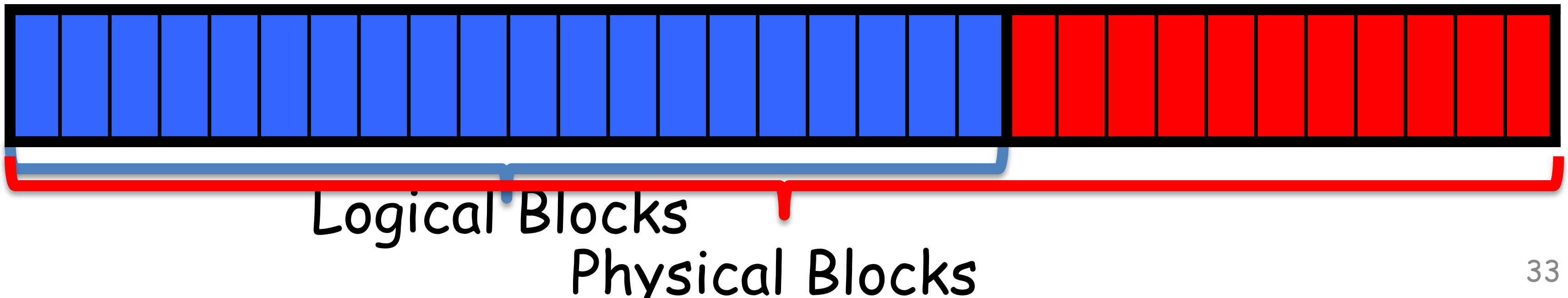


Garbage Collection

- **Garbage Collection (GC)** generates extra writes
- Write Amplification (WA) = (user writes + GC writes)/user writes
- Larger OP → Lower WA → Less erasures
- How does it all work together?

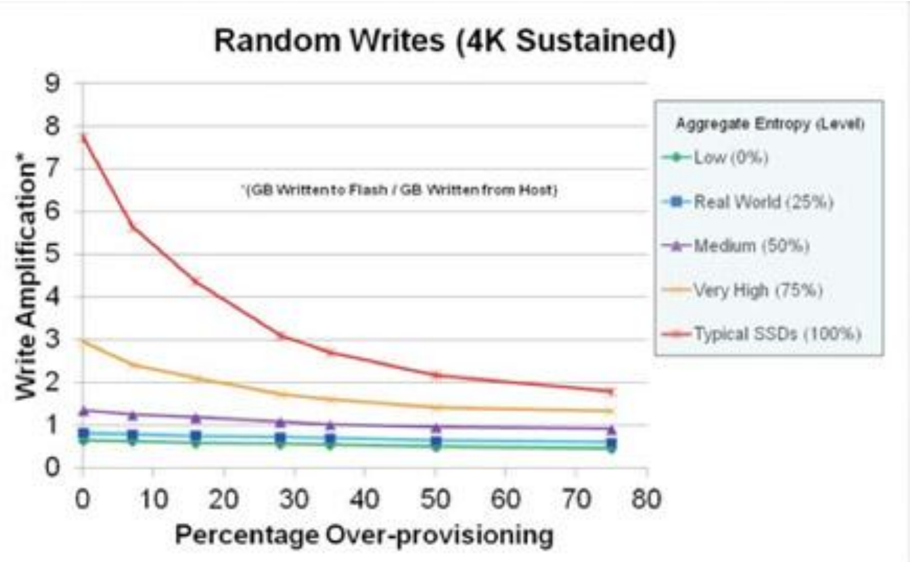
Greedy Garbage Collection

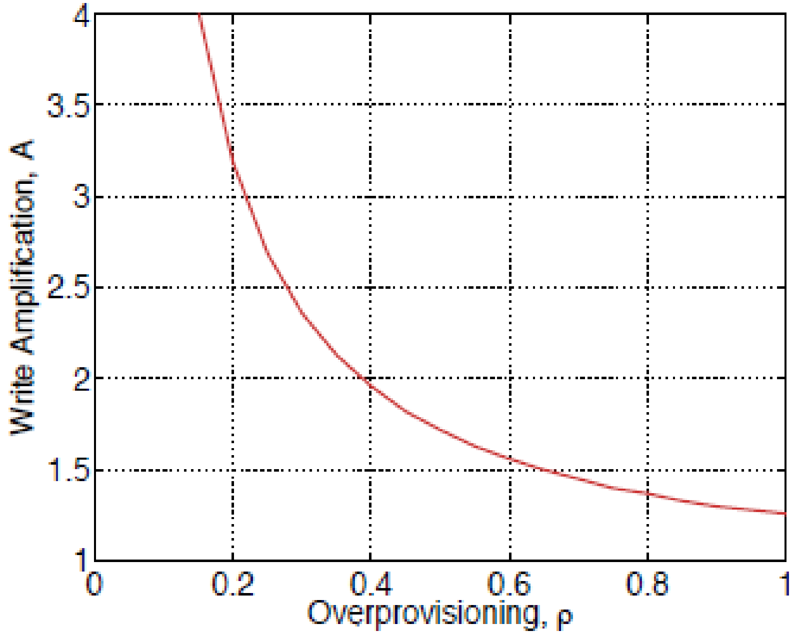
- **Write amplification** = $\frac{\text{\# Physical writes}}{\text{\# Logical writes}}$
- **Overprovisioning** = (T-U)/U ;
  T = #physical blocks, U = #logical blocks
- **Question**: How are the overprovisioning factor and write amplification related?
- **Theorem [Hu & Haas '10]**: Greedy garbage collection is optimal in order to reduce the write amplification (for uniform writing)



Analysis

- **Write amplification** $= \frac{\text{\# Physical writes}}{\text{\# Logical writes}}$
- **Overprovisioning** = (T-U)/U ;
  T = #physical blocks, U = #logical blocks
- **Question**: How are the overprovisioning factor and write amplification related?





Analysis

- **Write amplification** = $\frac{\text{\# Physical writes}}{\text{\# Logical writes}}$
- **Overprovisioning** = (T-U)/U ;
  T = #physical blocks, U = #logical blocks
- **Question'**: How are the overprovisioning factor and write amplification related, ***under random uniform writing***?
  - **N** = #logical page writes ; **M** = # physical page writes
  - **E** = #block erasures = M/Z , **Z** = # pages in a block
- On average: **Y = α'Z** valid pages in an erased block
  - **M = N + EY**
  - E = M/Z = (N+EY)/Z; (Z-Y)E = N; E=N/(Z-Y);

$$E=N/Z(1-\alpha')$$

- **Question**: What is the connection b/w **α=U/T** and **α'=Z/Y**?
- **Answer**: $\alpha = (\alpha'-1)/\ln(\alpha')$ (Menon '95, Desnoyers '12)

Other Variations of GC

- Wear Leveling algorithms to balance the number of times each block is erased
- Mapping in the block level: mapping between logical and physical blocks
  - Reduce the FTL size
- Other variations that take into account hot and cold data
- Usually performance is analyzed for random distribution but practical purposes take the Zipf distribution and benchmarks