

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.
Seoul National University

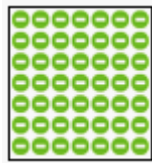
Fall 2021

Flash Memory



Flash Memory Basics

- Two states based on the presence of electrons



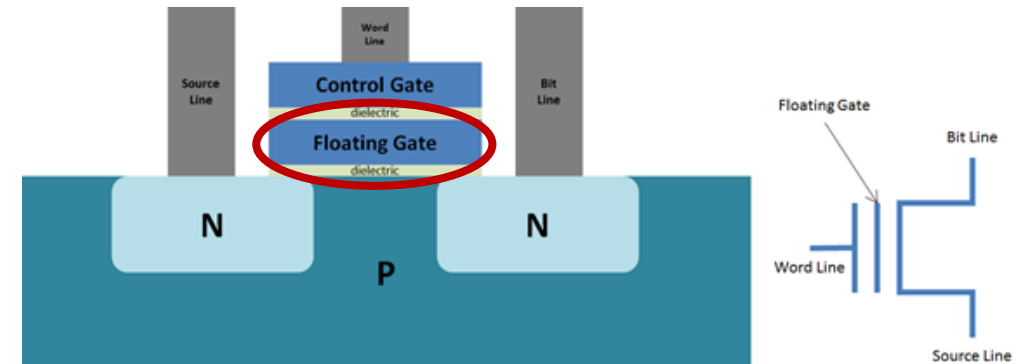
0 = Electrons present



1 = No electrons

- Challenges

- How to attract or expel electrons?
- How to find out whether there are electrons or not?
- How to keep electrons without any power?



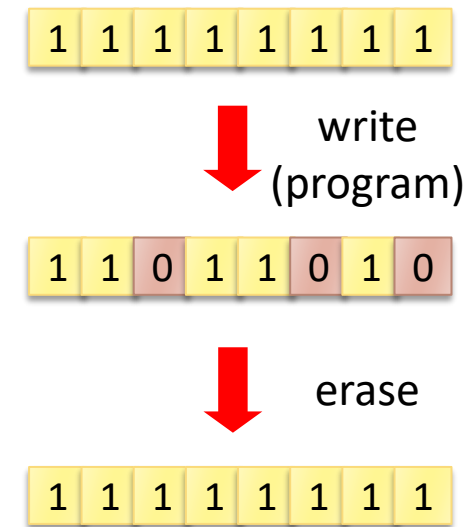
Flash Memory Characteristics

- Erase-before-write

- Read
- Write or Program: 1 → 0
- Erase: 0 → 1

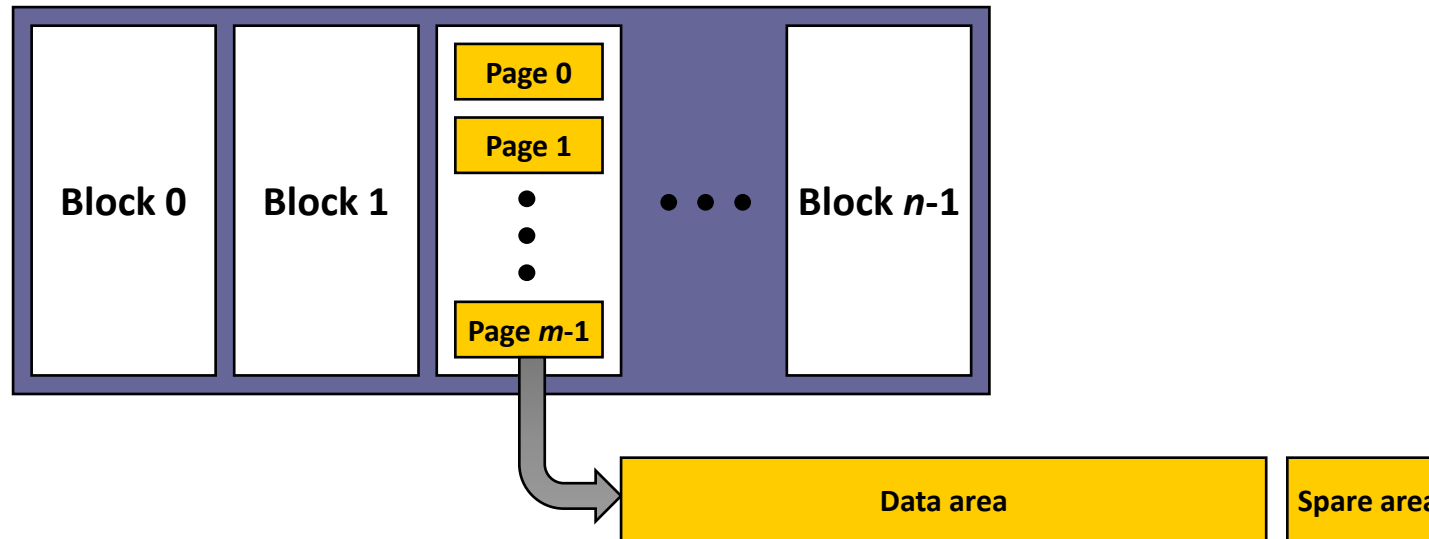
- Bulk erase

- Read/program unit
 - NOR: byte or word
 - NAND: page
- Erase unit: block



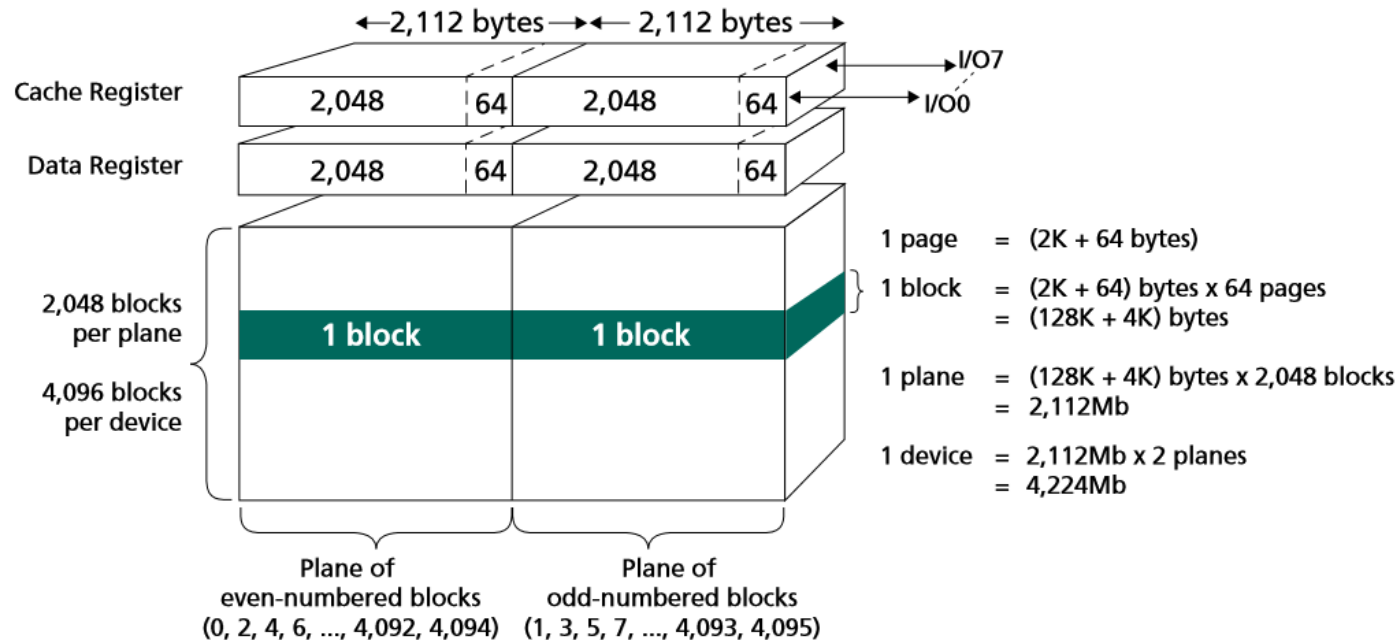
Logical View of NAND Flash

- A collection of **blocks**
- Each block has a number of **pages**
- The size of a block or a page depends on the technology (but, it's getting larger)



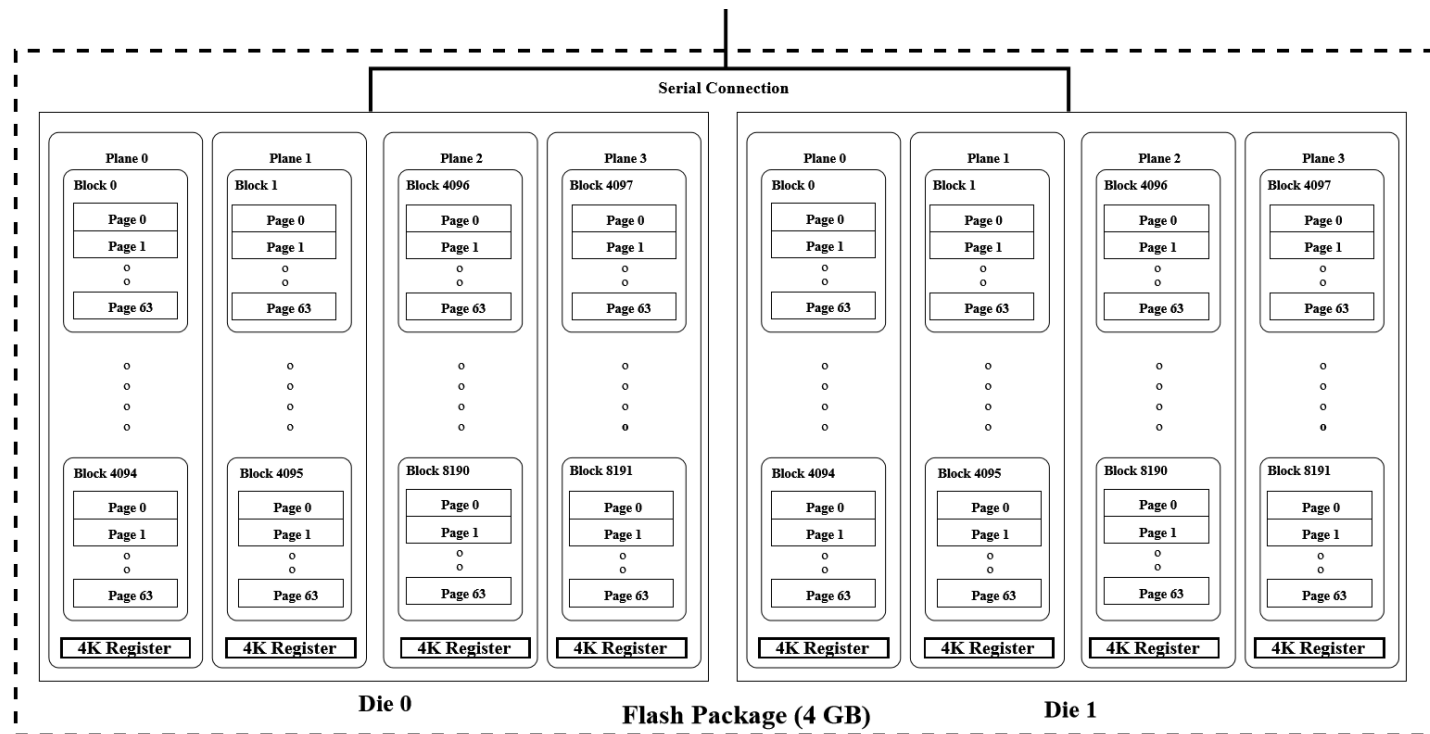
Plane

- Each plane has its own page register and cache register
- Pages can be programmed or read at once
- Optional feature: 1, 2, 4, 8, ... planes



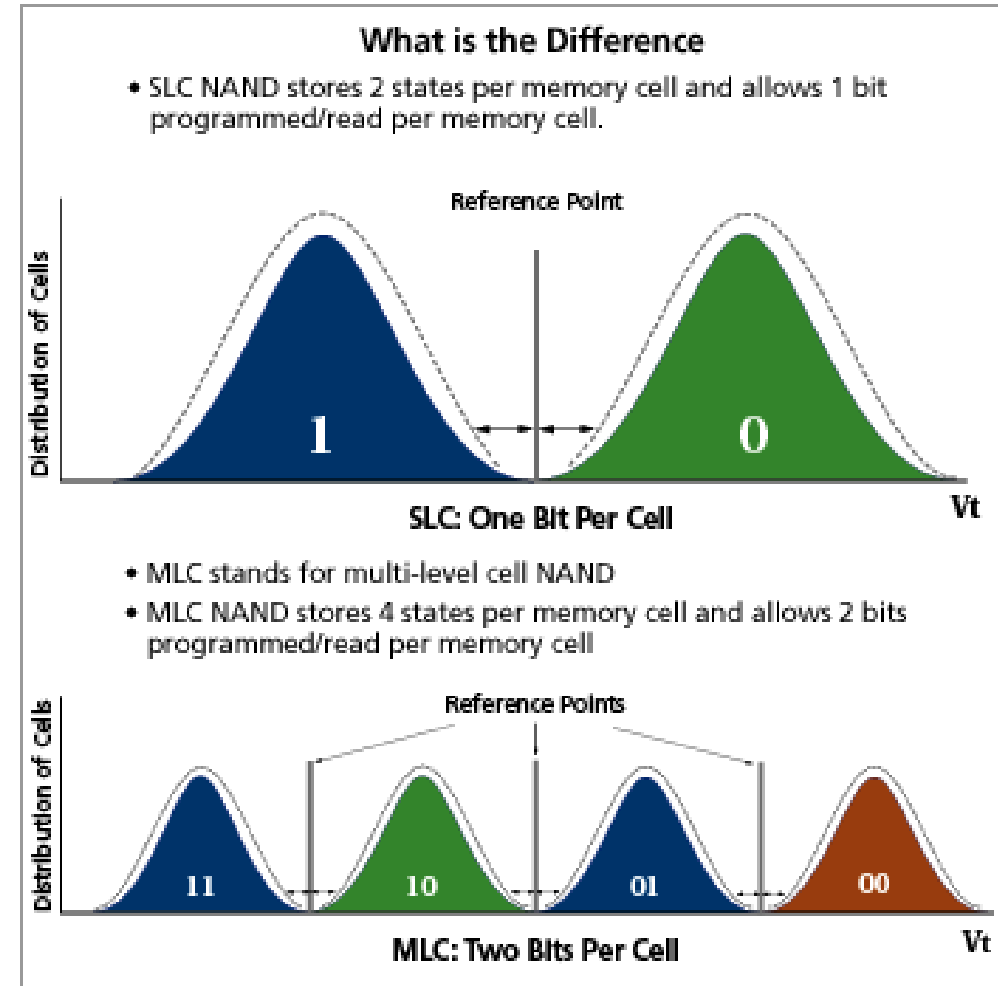
Die / Chip

- Each chip has multiple dies (can be stacked)
- + extra circuits, chip enable signal, ready/busy signal



NAND Flash Types

- SLC NAND
 - Single Level Cell (1 bit/cell)
- MLC NAND
 - Multi Level Cell (2 bits/cell)
- TLC NAND
 - Triple Level Cell (3 bits/cell)
- QLC NAND
 - Quad Level Cell (4 bits/cell)
- 3D NAND (or V-NAND)



Characteristics of NAND Flash

Erase-Before-Write

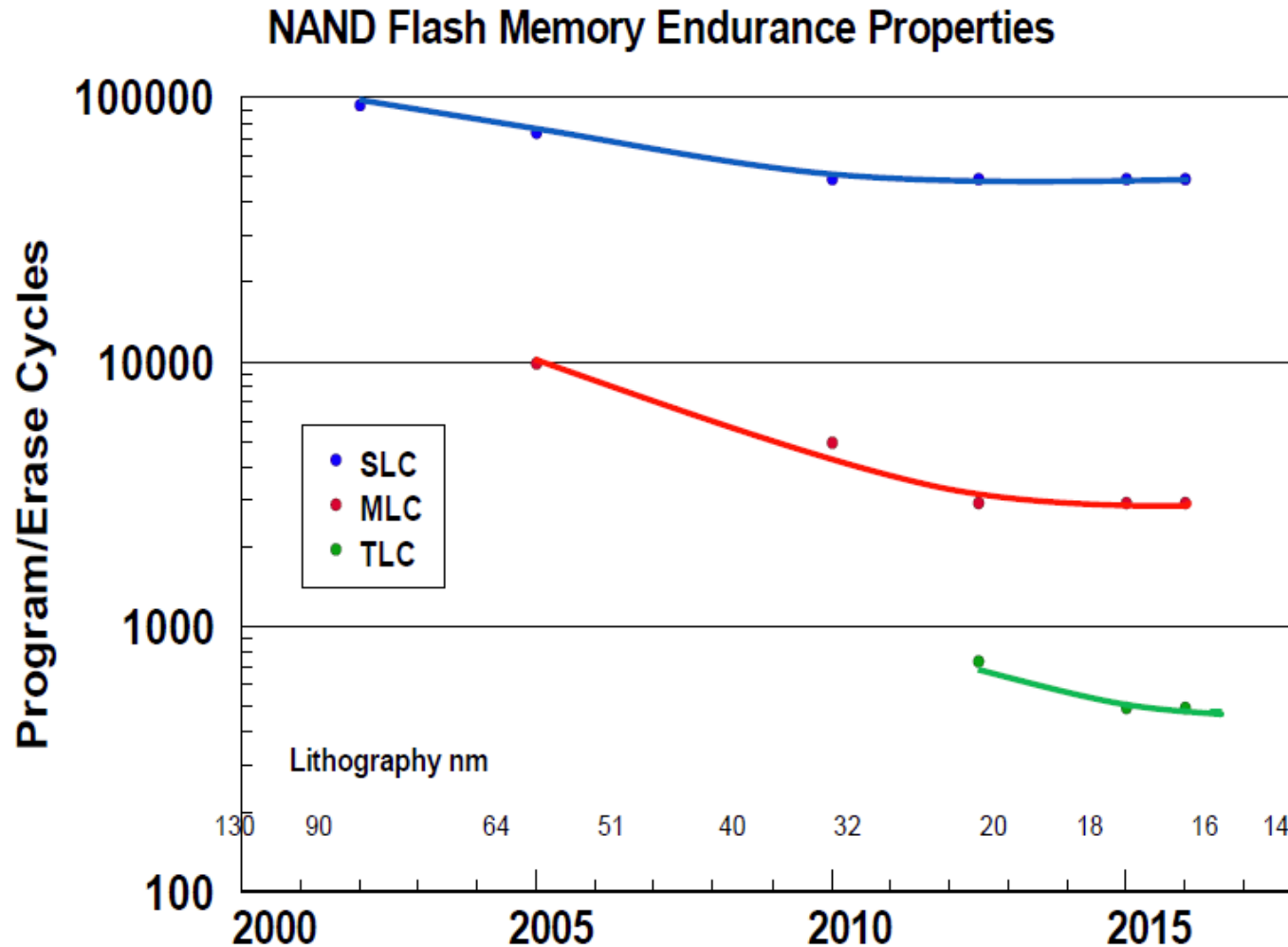
- In-place update (overwrite) is not allowed
- Pages must be erased before new data is programmed
- The erase unit is much larger than the read/write unit
 - Read/write unit: page (4KB, 8KB, 16KB, ...)
 - Erase unit: block (64-512 pages)
- What if there are live pages in the block we wish to erase?

Limited Lifetime

- The number of times NAND flash blocks can reliably be programmed and erased (P/E cycle) is limited
 - SLCs: 50,000 ~ 100,000
 - MLCs: 1,500 ~ 5,000
 - eMLCs (Enterprise MLCs): 10,000 ~ 30,000
 - TLCs: < 1,000
 - QLCs: ???
- High voltage applied to cell degrades oxide
 - Electrons are trapped in oxide
 - Break down of the oxide structure
- Requires _____



Flash Endurance



Old Data

Asymmetric Read/Write Latency

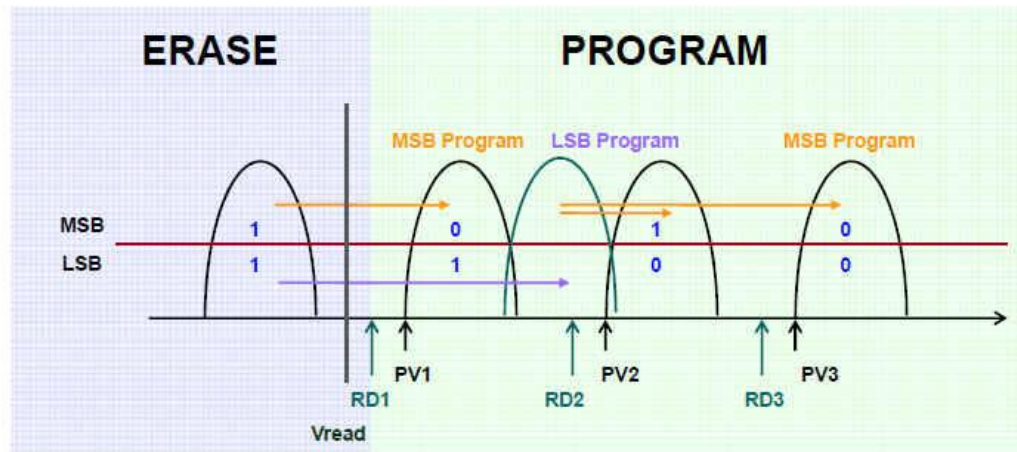
- Reading a page is faster than programming it
- Usually more than 10x
 - e.g., 1ynm MLC¹: Read 45 μ s, Program 1350 μ s, Erase 4ms
- Programming a page should go through multiple steps of Program & Verify phases

- As the technology shrinks, read/write latency tends to increase
- MLC and TLC make it even worse

¹D. Sharma, *System Design for Mainstream TLC SSD, FMS, 2014.*

MLC Programming

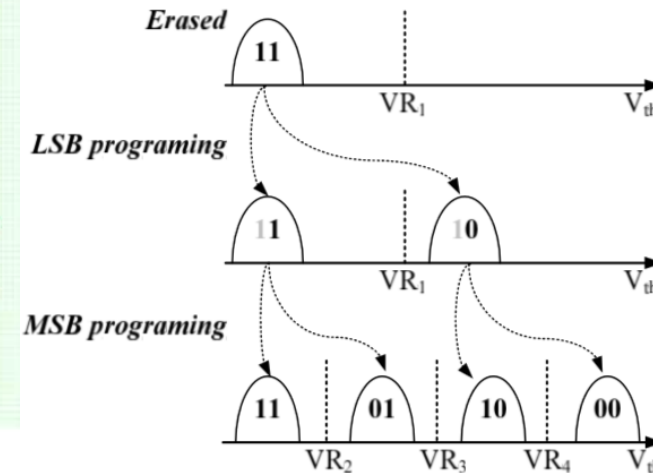
- LSB programmed first
 - Cell cannot move to the lower voltage before erase



Program : "1"(Erase) → "0"(Program)

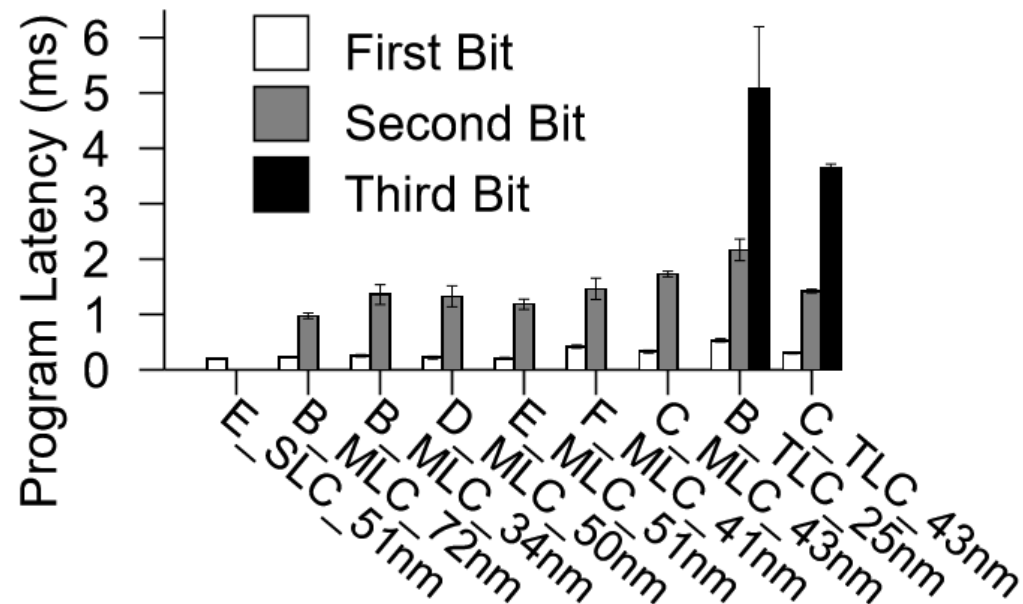
LSB Program : 1) Erase → Erase, 2) Erase → LSB

MSB Program: 1) Erase → Erase, 2) Erase → PV1, 3) LSB → PV2, 4) LSB → PV3



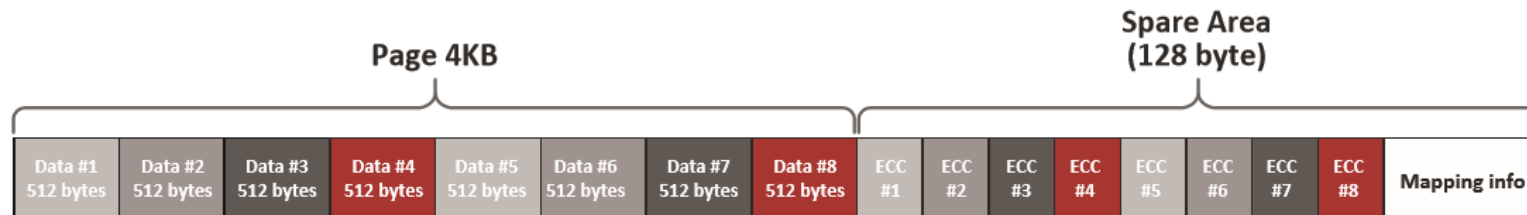
Paired Pages in MLC

- One cell represents two or three bits in paired pages
 - LSB: low voltage, fast program, less error
 - MSB: high voltage, slow program, more error
- Performance difference
- LSB page can be corrupted when MSB page programming is interrupted



Bit Errors

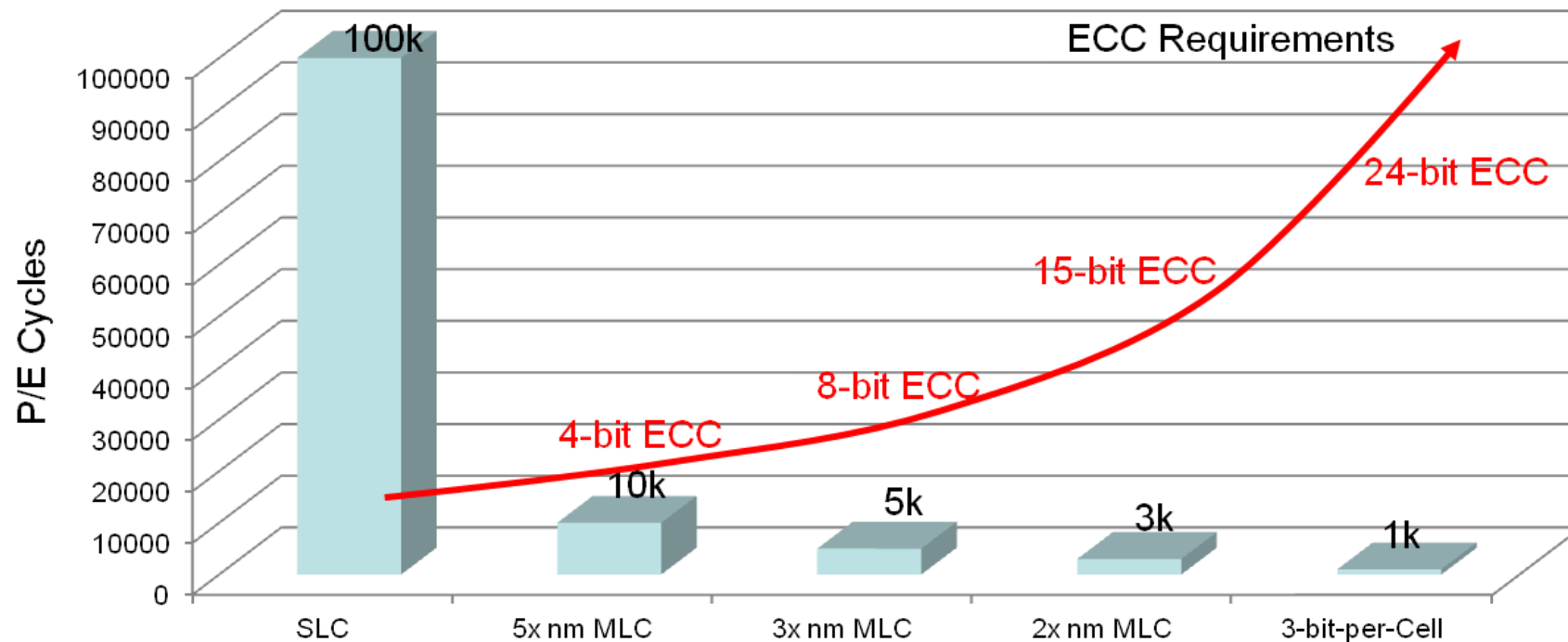
- Bits are flipping frequently
- Error Correction Code (ECC) in spare area



| Error Correction Level | Bits Required in the NAND Flash Spare Area | | |
|------------------------|--|--------------|-----|
| | Hamming | Reed-Solomon | BCH |
| 1 | 13 | 18 | 13 |
| 2 | N/A | 36 | 26 |
| 3 | N/A | 54 | 39 |
| 4 | N/A | 72 | 52 |
| 5 | N/A | 90 | 65 |
| 6 | N/A | 108 | 78 |
| 7 | N/A | 126 | 91 |
| 8 | N/A | 144 | 104 |
| 9 | N/A | 162 | 117 |
| 10 | N/A | 180 | 130 |

ECC Requirements

- Endurance continues to deteriorate
- Stronger ECCs are required: RS, BCH, LDPC



Reliability

■ Write disturbance

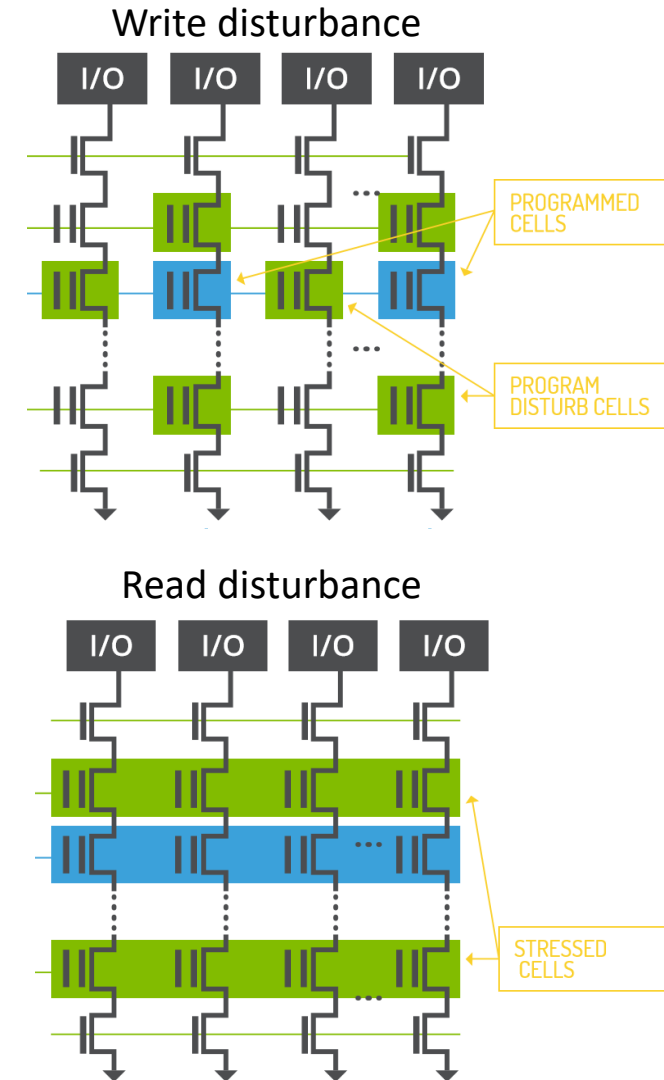
- When a page is programmed, adjacent cells receive elevated voltage stress

■ Read disturbance

- Repeated reading from one page can alter the values stored in other unread pages

■ error

- Threshold voltage shifts down due to charge leakage from the floating gate



Bad Blocks

- **Initial bad blocks**
 - Due to production yield constraints and the pressure to keep costs low
 - SLCs: up to 2%
 - MLCs: up to 5%
- **Run-time bad blocks**
 - Read, write, or erase failure
 - Permanent shift in the voltage levels of the cells due to trapped electrons
- **Requires run-time bad block management**

Page Programming Constraints

- **NOP**
 - The number of partial-page programming is limited
 - 1 / sector for most SLCs (4 for 2KB page)
 - 1 / page for most MLCs and TLCs
- **Sequential page programming**
 - Pages should be programmed sequentially inside a block
 - For large block SLCs, MLCs, and TLCs
- **SLC mode**
 - Possible to use only LSB pages in MLCs and TLCs
 - Faster and more reliable, higher P/E cycles

Beauty and the Beast

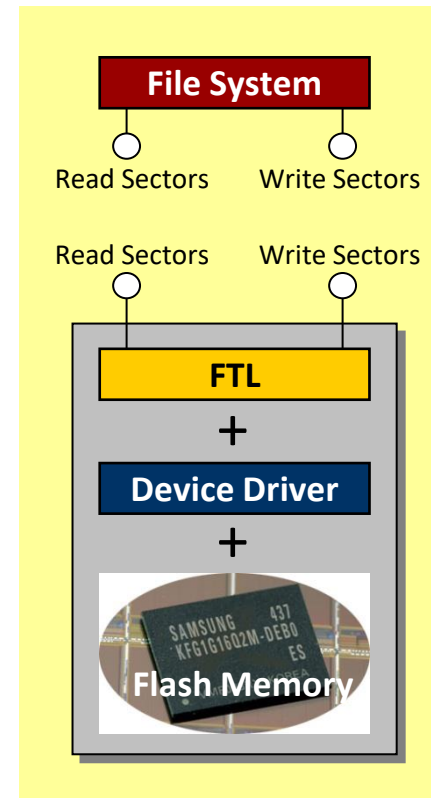
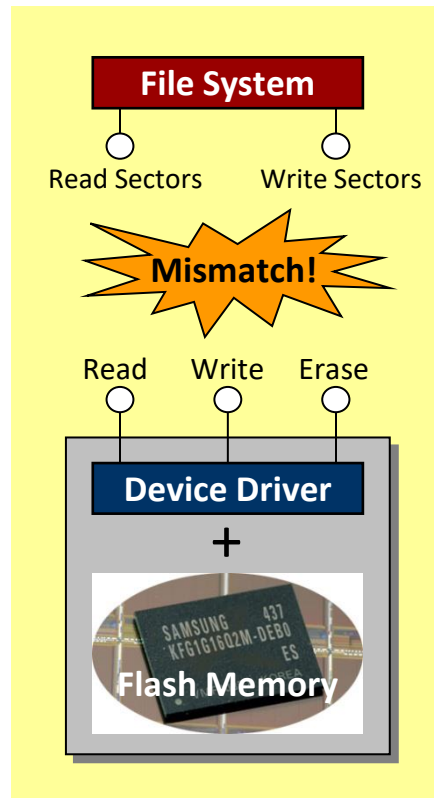
- **NAND Flash memory is a beauty**
 - Small, light-weight, robust, low-cost, low-power non-volatile device
- **NAND Flash memory is a beast**
 - Much slower program/erase operations
 - No in-place-update
 - Erase unit > write unit
 - Limited lifetime
 - Bit errors, bad blocks, ...
- **Software support is essential for performance and reliability!**



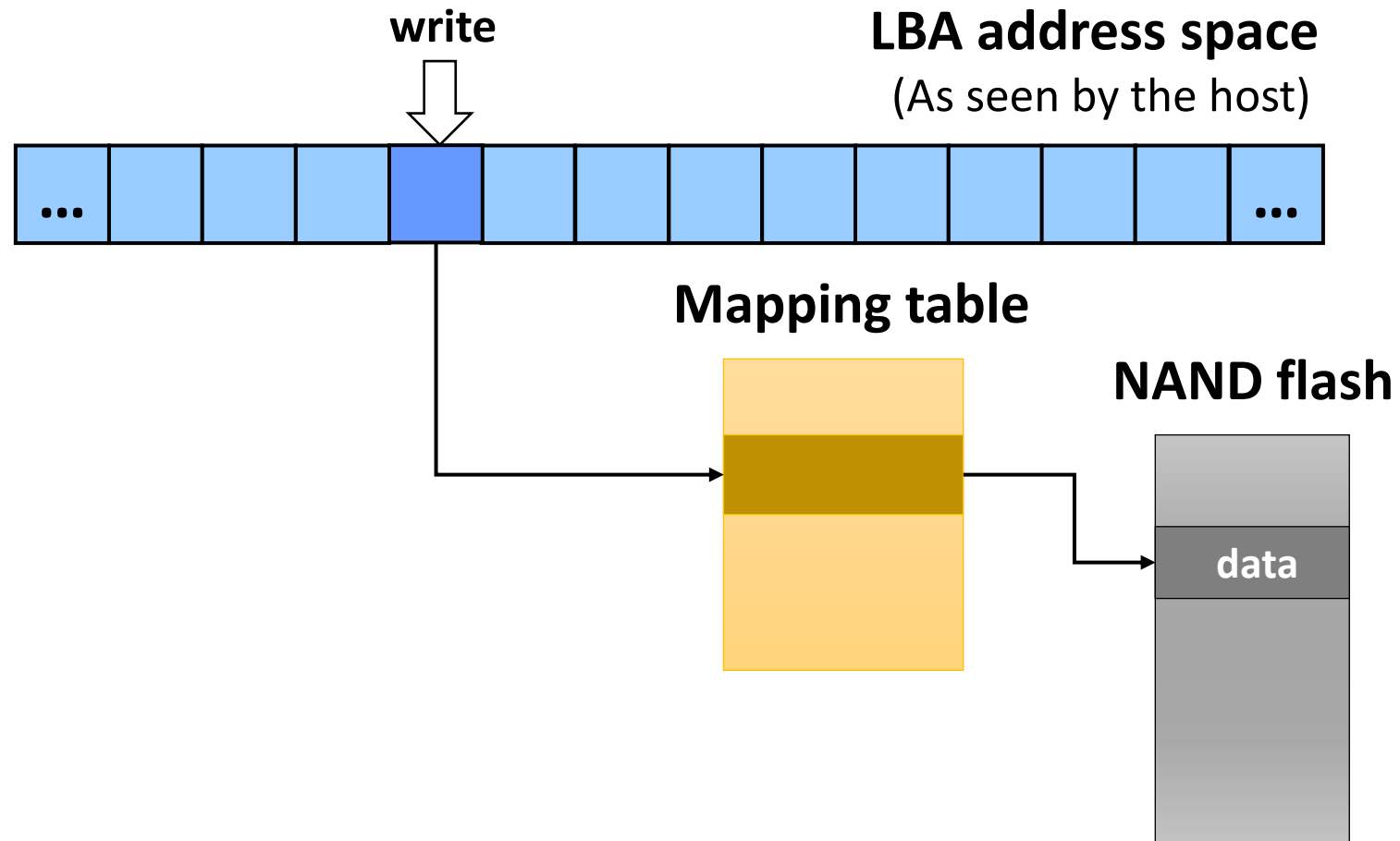
Page Mapping FTL

What is FTL?

- A software layer to make NAND flash fully emulate traditional block devices (or disks)

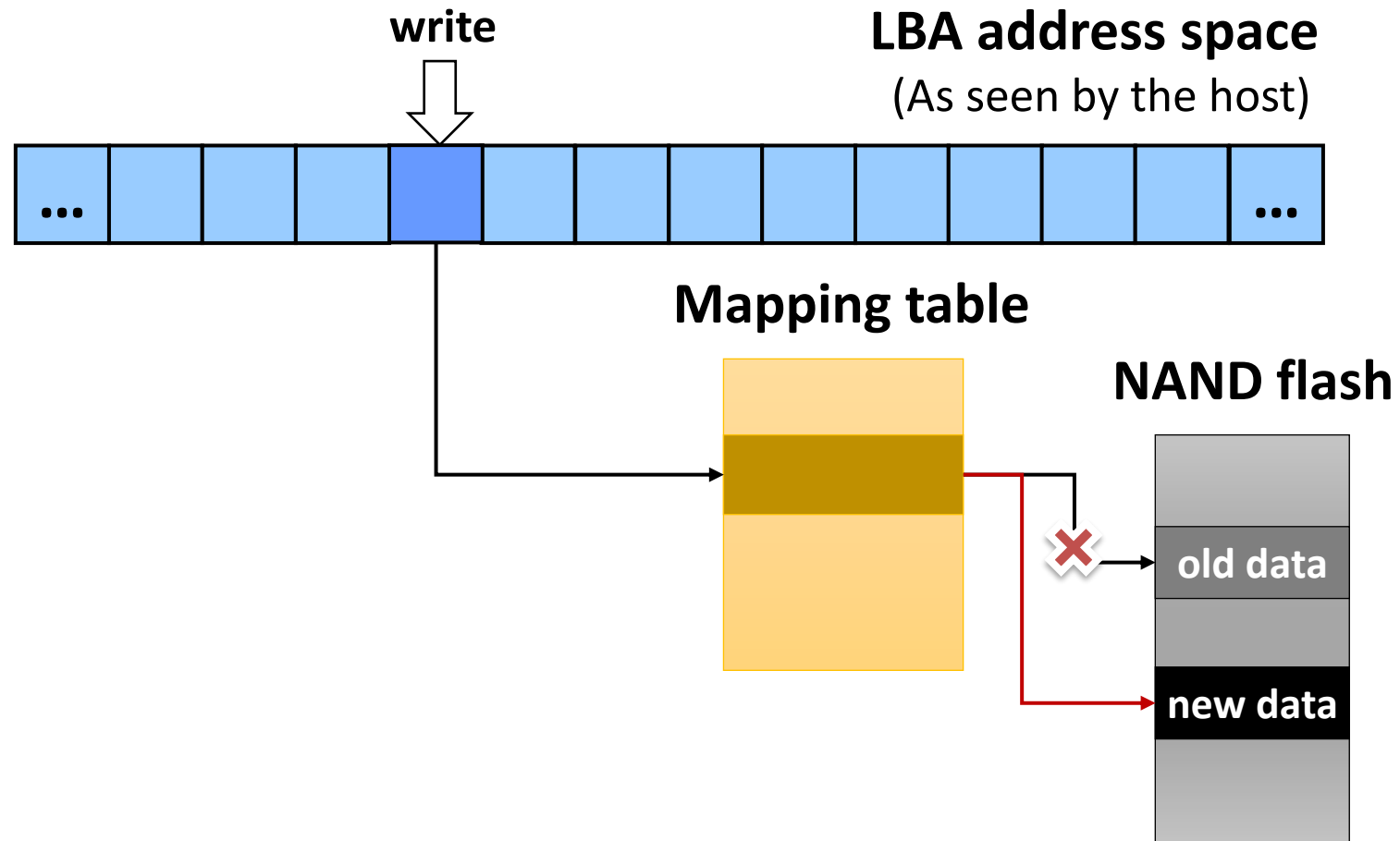


Address Mapping



Address Mapping

- Required due to “no overwrite” characteristic



Plethora of FTLs

SAST HFTL
SFTL MS FTL BPLRU

BFTL AFTL FAST LazyFTL KAST

Chameleon CNFTL DFTL LAST MNFTL

super-block scheme CFTL Log block scheme

GFTL μ -FTL JFTL zFTL

Hydra FTL Vanilla FTL Replacement block scheme

Reconfigurable FTL YanusFTL

WAFTL UFTL and so on



Mapping Schemes

- **Page mapping**
 - Fine-granularity page-level map table
 - High amount of memory space required for the map table
- **Block mapping**
 - Coarse-granularity block-level map table
 - Small amount of memory space required for the map table
- **_____ mapping**
 - Use both page-level and block-level map tables
 - Higher algorithm complexity

Page Mapping

- Mapping in page-level
 - Logical page number \rightarrow physical page number
 - Page mapping table (PMT) required
 - # entries in PMT == # pages visible to OS
- Translation
 - Step 1: logical sector number (LSN) \rightarrow logical page number (LPN)
 - Step 2: LPN \rightarrow physical page number (PPN) via PMT

Example: Page Mapping

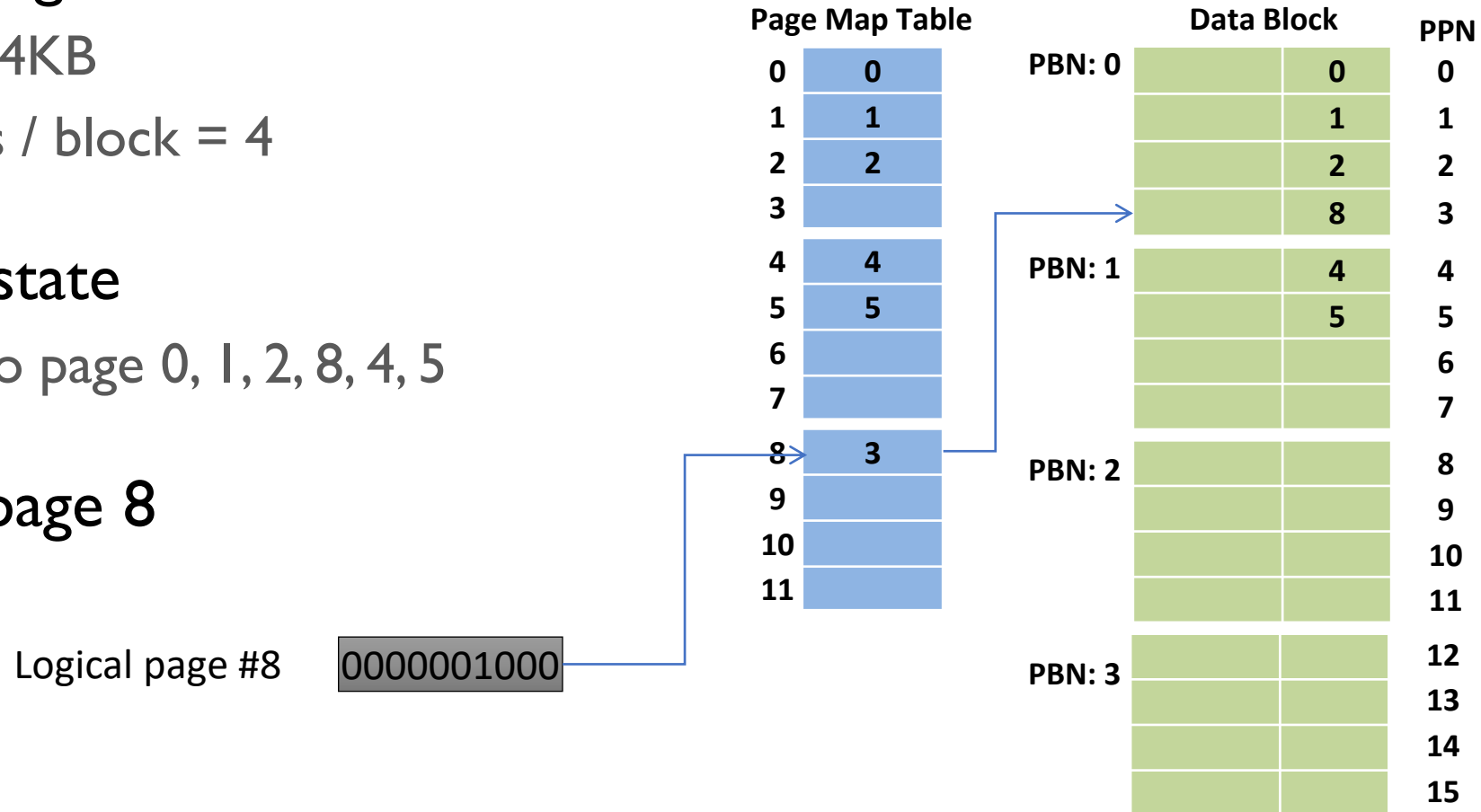
- Flash configuration

- Page size: 4KB
- # of pages / block = 4

- Current state

- Written to page 0, 1, 2, 8, 4, 5

- Reading page 8



Example: Page Mapping

- Flash configuration
 - Page size: 4KB
 - # of pages / block = 4
- Current state
 - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
 - Write to page 9
 - Write to page 3
 - Write to page 5

Page Map Table

| | |
|----|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | |
| 4 | 4 |
| 5 | 5 |
| 6 | |
| 7 | |
| 8 | 3 |
| 9 | |
| 10 | |
| 11 | |

Data Block

| | Data Block | PPN |
|--------|------------|-----|
| PBN: 0 | 0 | 0 |
| | 1 | 1 |
| | 2 | 2 |
| | 8 | 3 |
| PBN: 1 | 4 | 4 |
| | 5 | 5 |
| | | 6 |
| | | 7 |
| PBN: 2 | | 8 |
| | | 9 |
| | | 10 |
| | | 11 |
| PBN: 3 | | 12 |
| | | 13 |
| | | 14 |
| | | 15 |

Example: Page Mapping

- Flash configuration
 - Page size: 4KB
 - # of pages / block = 4
- Current state
 - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
 - Write to page 9
 - Write to page 3
 - Write to page 5

Page Map Table

| | |
|----|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | |
| 4 | 4 |
| 5 | 5 |
| 6 | |
| 7 | |
| 8 | 3 |
| 9 | 6 |
| 10 | |
| 11 | |

Data Block

| | Data Block | PPN |
|--------|------------|-----|
| PBN: 0 | 0 | 0 |
| | 1 | 1 |
| | 2 | 2 |
| | 8 | 3 |
| PBN: 1 | 4 | 4 |
| | 5 | 5 |
| | 9 | 6 |
| | | 7 |
| PBN: 2 | | 8 |
| | | 9 |
| | | 10 |
| | | 11 |
| PBN: 3 | | 12 |
| | | 13 |
| | | 14 |
| | | 15 |

Example: Page Mapping

- Flash configuration
 - Page size: 4KB
 - # of pages / block = 4
- Current state
 - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
 - Write to page 9
 - Write to page 3**
 - Write to page 5

Page Map Table

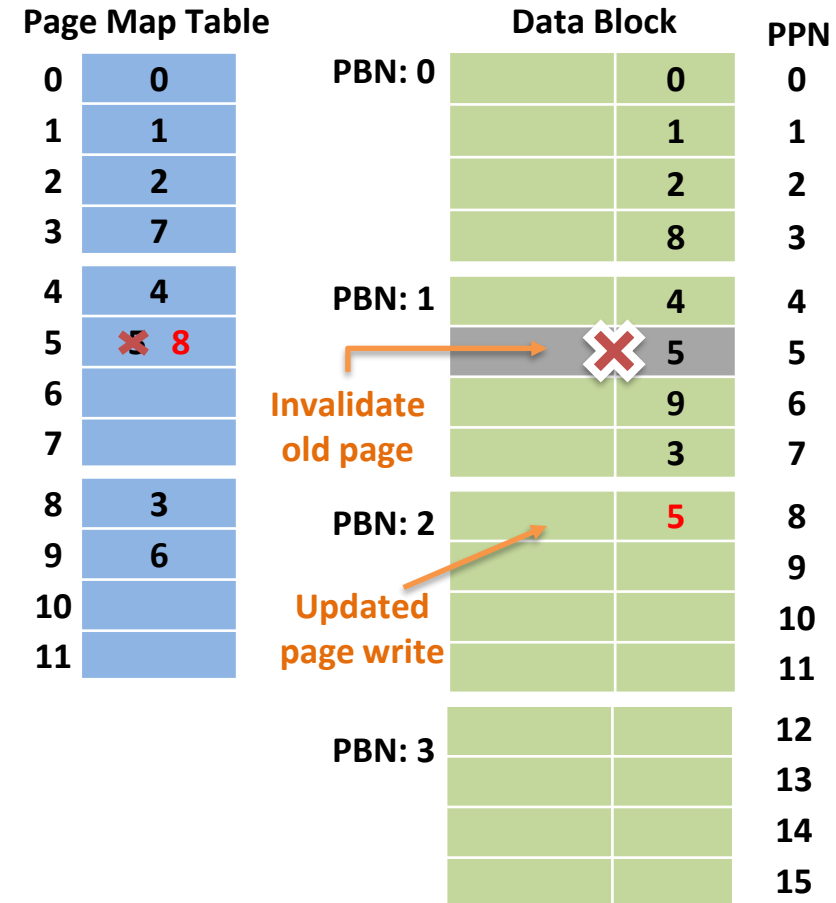
| | |
|----|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 7 |
| 4 | 4 |
| 5 | 5 |
| 6 | |
| 7 | |
| 8 | 3 |
| 9 | 6 |
| 10 | |
| 11 | |

Data Block

| | Data Block | PPN |
|--------|------------|-----|
| PBN: 0 | 0 | 0 |
| | 1 | 1 |
| | 2 | 2 |
| | 8 | 3 |
| PBN: 1 | 4 | 4 |
| | 5 | 5 |
| | 9 | 6 |
| | 3 | 7 |
| PBN: 2 | | 8 |
| | | 9 |
| | | 10 |
| | | 11 |
| PBN: 3 | | 12 |
| | | 13 |
| | | 14 |
| | | 15 |

Example: Page Mapping

- Flash configuration
 - Page size: 4KB
 - # of pages / block = 4
- Current state
 - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
 - Write to page 9
 - Write to page 3
 - Write to page 5



Page Mapping

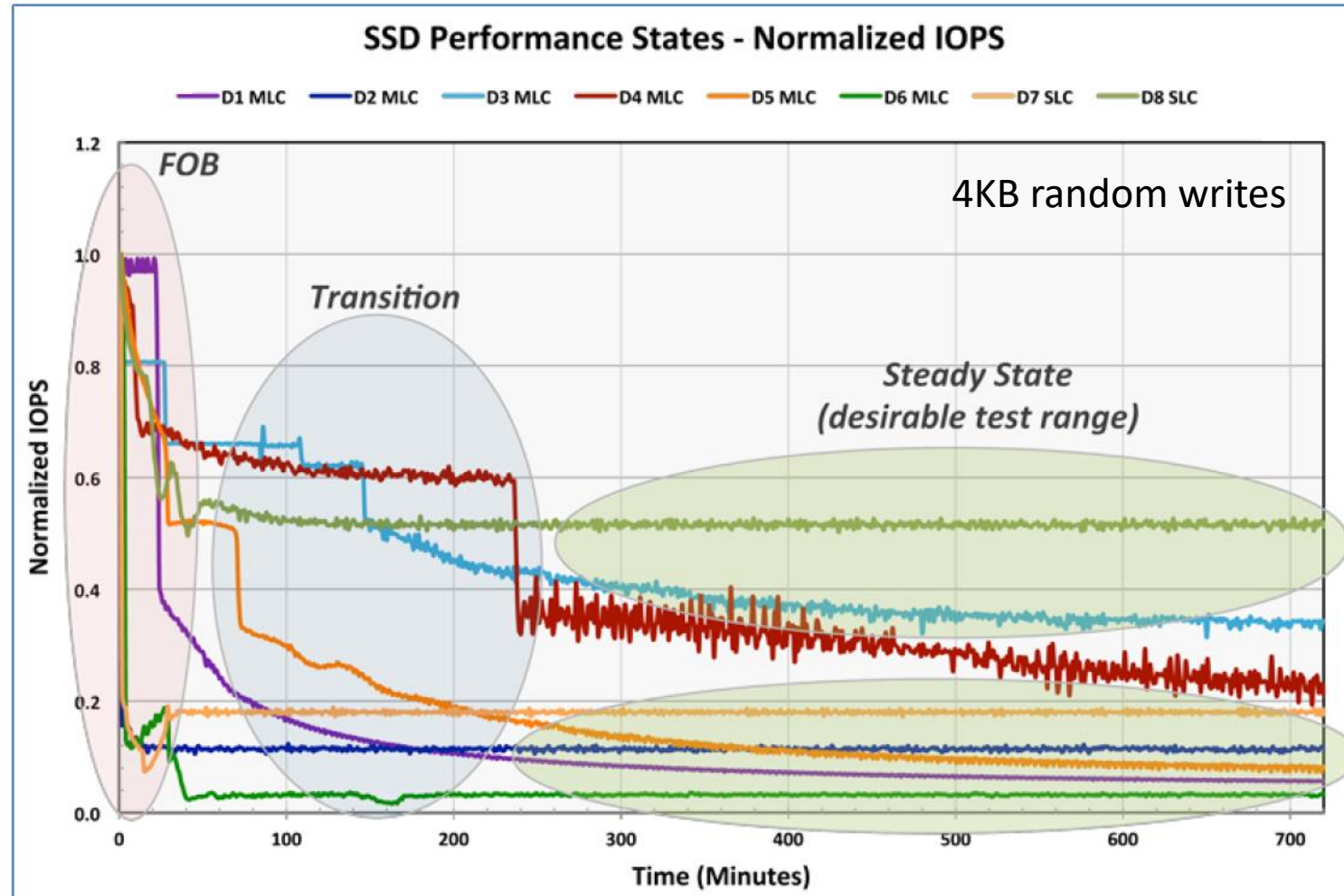
■ Pros

- Most flexible
- Efficient handling of small random writes
 - A logical page can be located anywhere within the flash storage
 - Updated page can be written to any free page

■ Cons

- Large memory footprint
 - One page mapping entry per page
 - 32MB for 32GB (4KB page)
- Sensitive to the amount of reserved blocks
- Performance affected as the system ages

Why?



Garbage Collection

- **Garbage collection (GC)**
 - Eventually, FTL will run out of blocks to write to
 - GC must be performed to reclaim free space
 - Actual GC procedure depends on the mapping scheme

- **GC in page-mapping FTL**
 - Select victim block(s)
 - Copy all valid pages of victim block(s) to free block
 - Erase victim block(s)
 - Note: At least one free block should be reserved for GC

Example: GC in Page Mapping

■ Current state

- Written to page 0, 1, 2, 8, 4, 5
- Written to page 9, 3, 5

■ New requests (in order)

- Write to page 8
- Write to page 9
- Write to page 3
- Write to page 1
- Write to page 4

| Page Map Table | | Data Block | | PPN |
|----------------|---|------------|-------------|-----|
| 0 | 0 | PBN: 0 | 0 | 0 |
| 1 | 1 | | 1 | 1 |
| 2 | 2 | | 2 | 2 |
| 3 | 7 | | 8 | 3 |
| 4 | 4 | PBN: 1 | 4 | 4 |
| 5 | 8 | | 5 | 5 |
| 6 | | | 9 | 6 |
| 7 | | | 3 | 7 |
| 8 | 3 | PBN: 2 | 5 | 8 |
| 9 | 6 | | | 9 |
| 10 | | | | 10 |
| 11 | | | | 11 |
| | | PBN: 3 | | 12 |
| | | | | 13 |
| | | | Spare block | 14 |
| | | | | 15 |

Example: GC in Page Mapping

■ Current state

- Written to page 0, 1, 2, 8, 4, 5
- Written to page 9, 3, 5

■ New requests (in order)

- Write to page 8
- Write to page 9
- Write to page 3
- Write to page 1
- Write to page 4

| Page Map Table | | Data Block | | PPN |
|----------------|---|------------|-------------|-----|
| 0 | 0 | PBN: 0 | 0 | 0 |
| 1 | 1 | | 1 | 1 |
| 2 | 2 | | 2 | 2 |
| 3 | 7 | | X 8 | 3 |
| 4 | 4 | PBN: 1 | 4 | 4 |
| 5 | 8 | | X 5 | 5 |
| 6 | | | 9 | 6 |
| 7 | | | 3 | 7 |
| 8 | 9 | PBN: 2 | 5 | 8 |
| 9 | 6 | | 8 | 9 |
| 10 | | | | 10 |
| 11 | | | | 11 |
| | | PBN: 3 | | 12 |
| | | | Spare block | 13 |
| | | | | 14 |
| | | | | 15 |

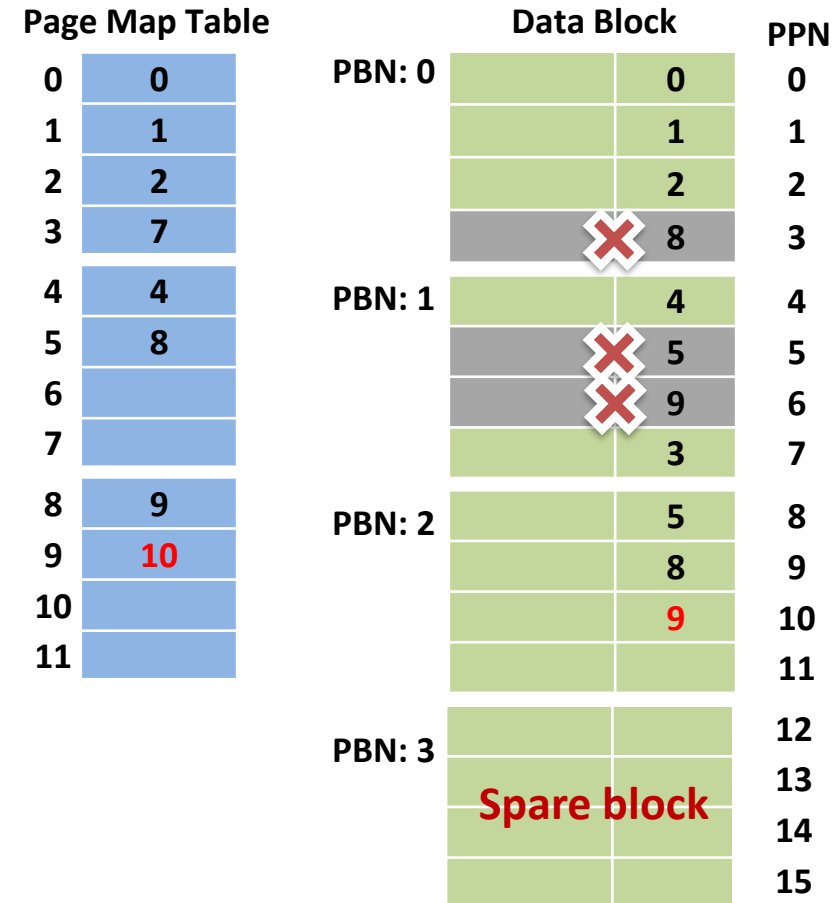
Example: GC in Page Mapping

■ Current state

- Written to page 0, 1, 2, 8, 4, 5
- Written to page 9, 3, 5

■ New requests (in order)

- Write to page 8
- **Write to page 9**
- Write to page 3
- Write to page 1
- Write to page 4



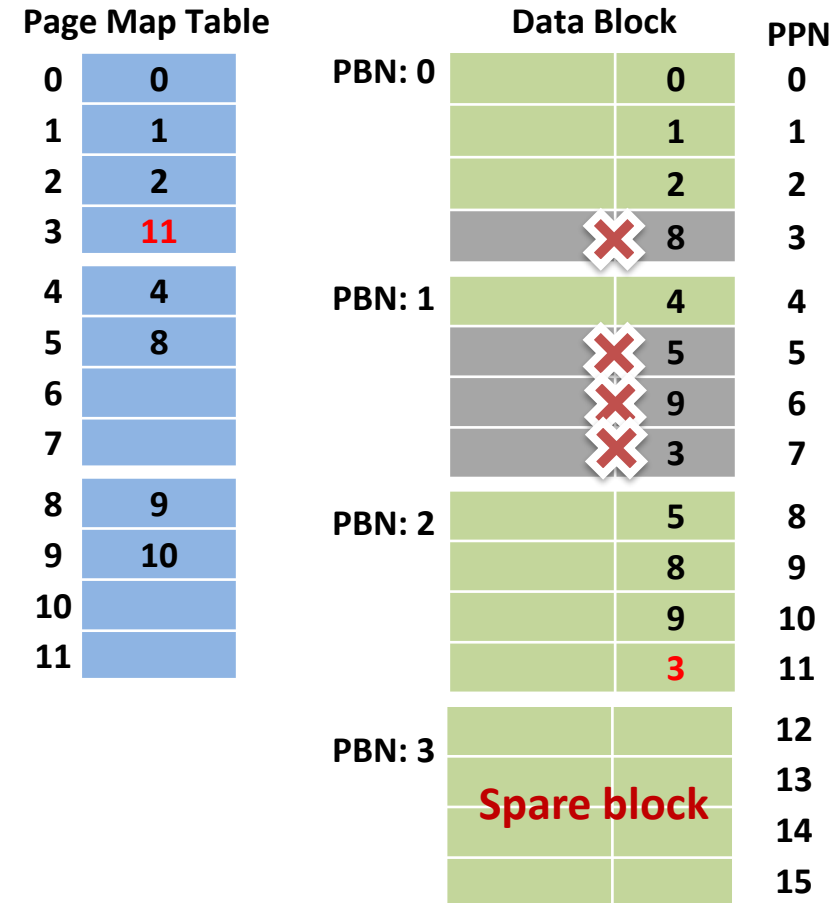
Example: GC in Page Mapping

■ Current state

- Written to page 0, 1, 2, 8, 4, 5
- Written to page 9, 3, 5

■ New requests (in order)

- Write to page 8
- Write to page 9
- **Write to page 3**
- Write to page 1
- Write to page 4



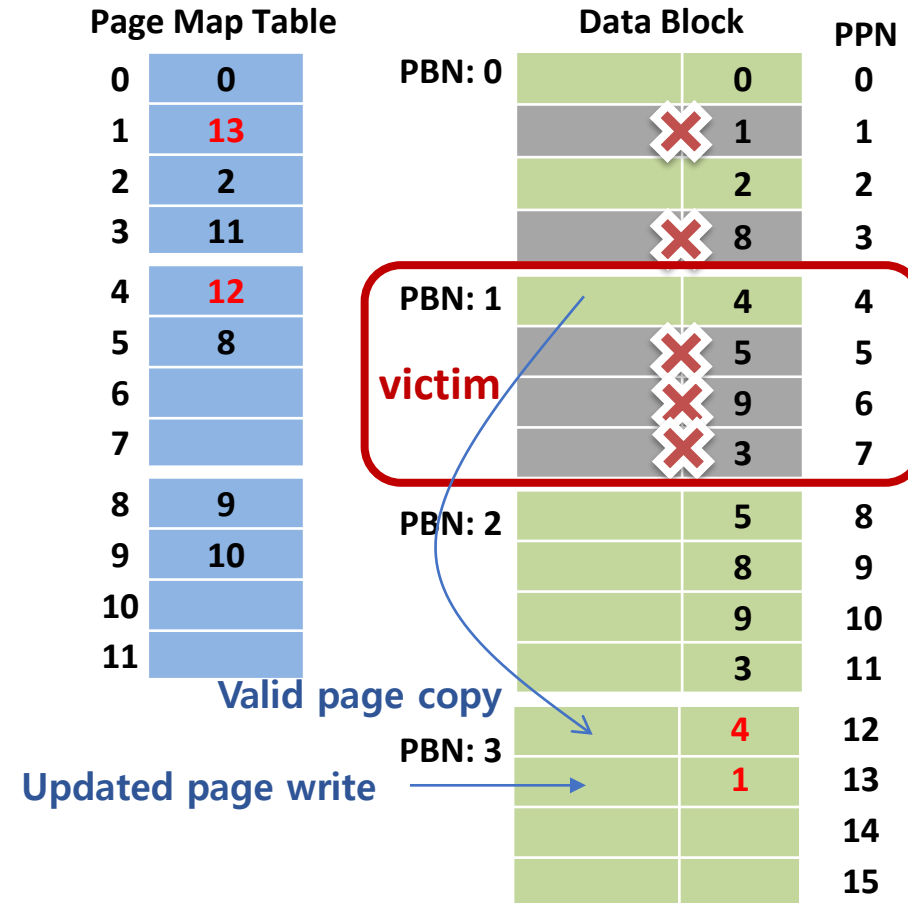
Example: GC in Page Mapping

■ Current state

- Written to page 0, 1, 2, 8, 4, 5
- Written to page 9, 3, 5

■ New requests (in order)

- Write to page 8
- Write to page 9
- Write to page 3
- **Write to page 1**
- Write to page 4



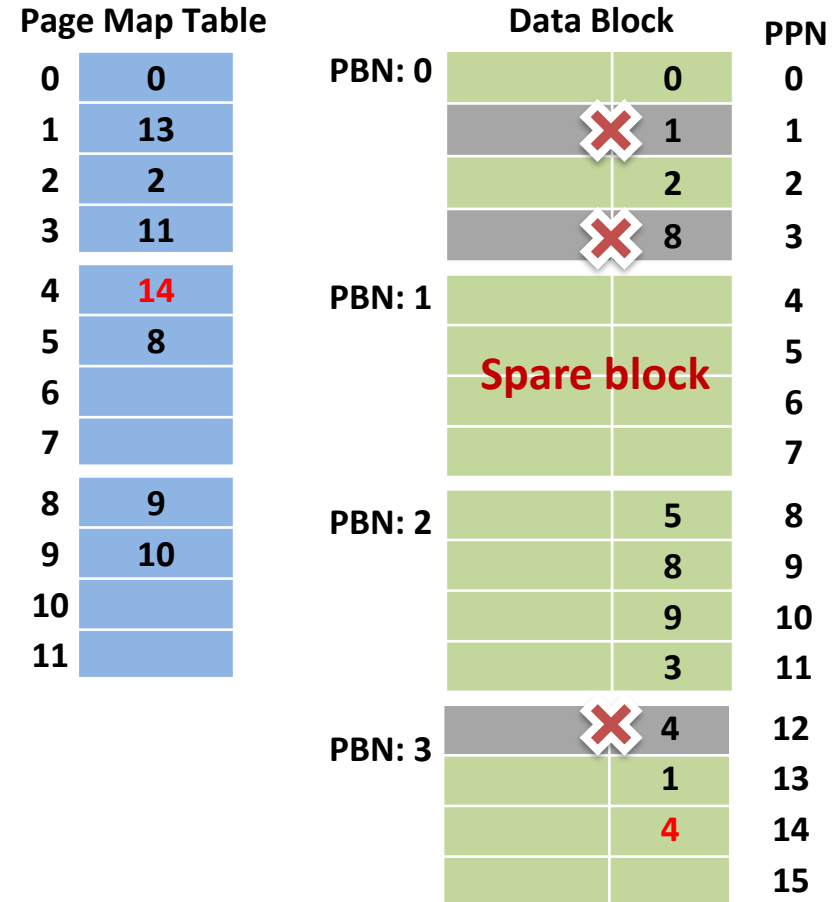
Example: GC in Page Mapping

■ Current state

- Written to page 0, 1, 2, 8, 4, 5
- Written to page 9, 3, 5

■ New requests (in order)

- Write to page 8
- Write to page 9
- Write to page 3
- Write to page 1
- **Write to page 4**



Write Amplification

- Ratio of data written to flash to data written from host

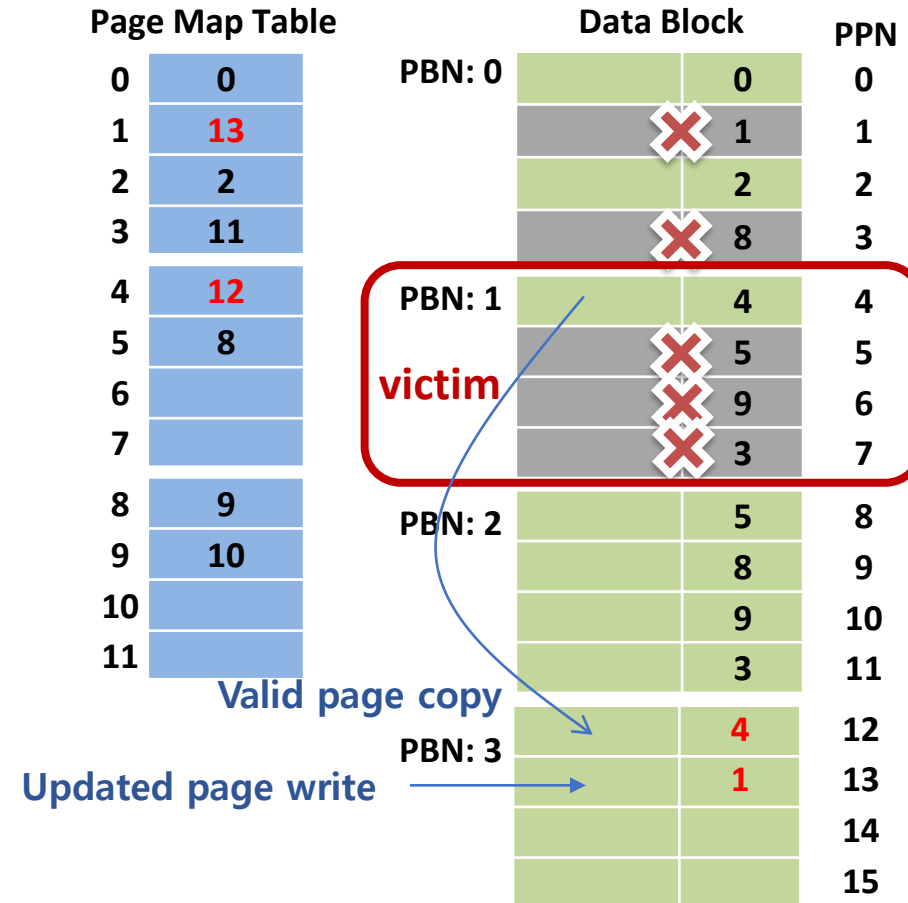
- Write Amplification Factor (WAF)

$$= \frac{\text{Bytes written to *Flash*}}{\text{Bytes written from *Host*}} = \frac{\text{Bytes written from *Host* + Bytes written during *GC*}}{\text{Bytes written from *Host*}}$$

- Generally, WAF is greater than one in flash storage
 - Due to valid page copies made from victim block to free block during GC
 - WAF is one of metrics that shows the efficiency of GC

Example: Write Amplification

- Current state
 - Written to page 0, 1, 2, 8, 4, 5
 - Written to page 9, 3, 5
- New requests (in order)
 - Write to page 8, 9, 3, 1
- WAF = 1.08
 - Total host writes: 13
 - Total flash writes: 14



Victim Selection Policy: Greedy

- Selects a block with the largest amount of invalid data
- A block with the _____ utilization u

$$u = \frac{\text{Number of valid pages in a block}}{\text{Number of Pages in a block}}$$

- **Pros?**

- **Cons?**

Victim Selection Policy: Cost-Benefit

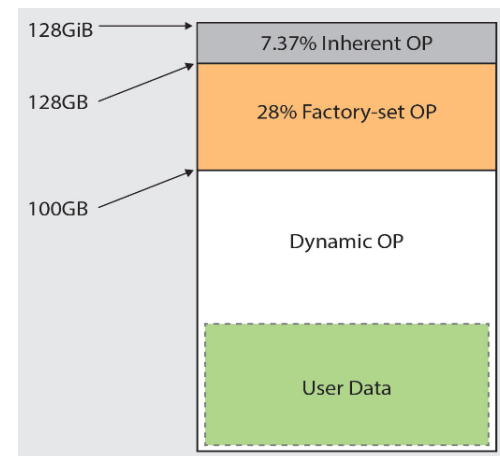
- Selects a block with the _____

$$\frac{Benefit}{Cost} = \frac{(1 - u)}{2u} \times age$$

- u : utilization
 - age : the time since the last modification
- **Pros?**
 - **Cons?**

Over-Provisioning

- OP (Over-Provisioning) = $\frac{\text{Physical Capacity}}{\text{Logical Capacity}} - 1$
 - Extra media space on an SSD that does not contain user data
- Typical SSDs have more space than is advertised
 - Consumer SSDs: ~ 7%
 - 1 Gigabyte (GB) = 10^9 bytes = 1,000,000,000 bytes
 - 1 Gibibyte (GiB) = 2^{30} bytes = 1,073,741,824 bytes
 - Enterprise SSDs: > 25%
 - 100GB user space on 128GiB SSD: 37.4%



Example: Over-Provisioning

- OP = 33%
- Logical capacity: 3 blocks
- Physical capacity: 4 blocks

Page Map Table

| | |
|----|----|
| 0 | 0 |
| 1 | 13 |
| 2 | 2 |
| 3 | 11 |
| 4 | 14 |
| 5 | 8 |
| 6 | |
| 7 | |
| 8 | 9 |
| 9 | 10 |
| 10 | |
| 11 | |

| | Data Block | PPN |
|--------|--------------|-----|
| PBN: 0 | 0 | 0 |
| | 1 | 1 |
| | 2 | 2 |
| | 8 | 3 |
| PBN: 1 | | 4 |
| | Spare block | 5 |
| | | 6 |
| | | 7 |
| PBN: 2 | 5 | 8 |
| | 8 | 9 |
| | 9 | 10 |
| | 3 | 11 |
| PBN: 3 | 4 | 12 |
| | 1 | 13 |
| | 4 | 14 |
| | | 15 |

Why Over-Provisioning?

- Over-Provisioning Space (OPS) is used for
 - Write buffers
 - _____
 - _____
 - _____
- Garbage collection cost
 - Affected by utilization of SSD space and Over-Provisioning
 - _____ utilization → Better performance
 - _____ OP → Better performance

Over-Provisioning and GC

- IOPS for random write workloads
 - What about for sequential write workloads?

