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

Systems Software &
Architecture Lab.

Seoul National University

Spring 2020

Solid State Drives (SSDs)

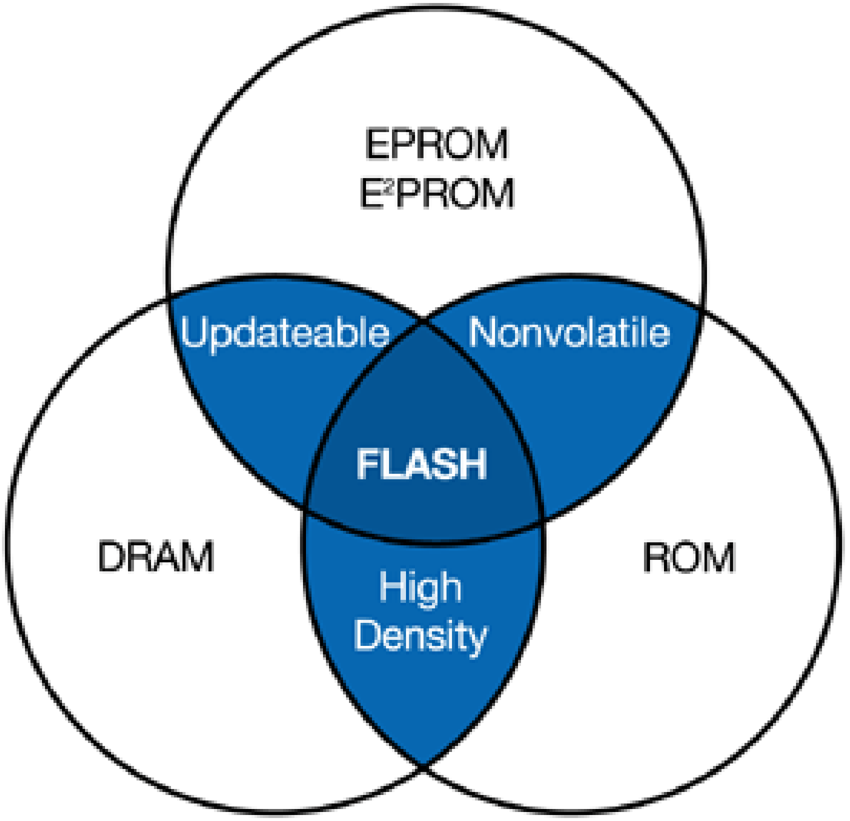
Memory Types

FLASH

- High-density
- Low-cost
- High-speed
- Low-power
- High reliability

DRAM

- High-density
- Low-cost
- High-speed
- High-power



EPROM

- Non-volatile
- High-density
- Ultraviolet light for erasure

EEPROM

- Non-volatile
- Lower reliability
- Higher cost
- Lowest density
- Electrically byte-erasable

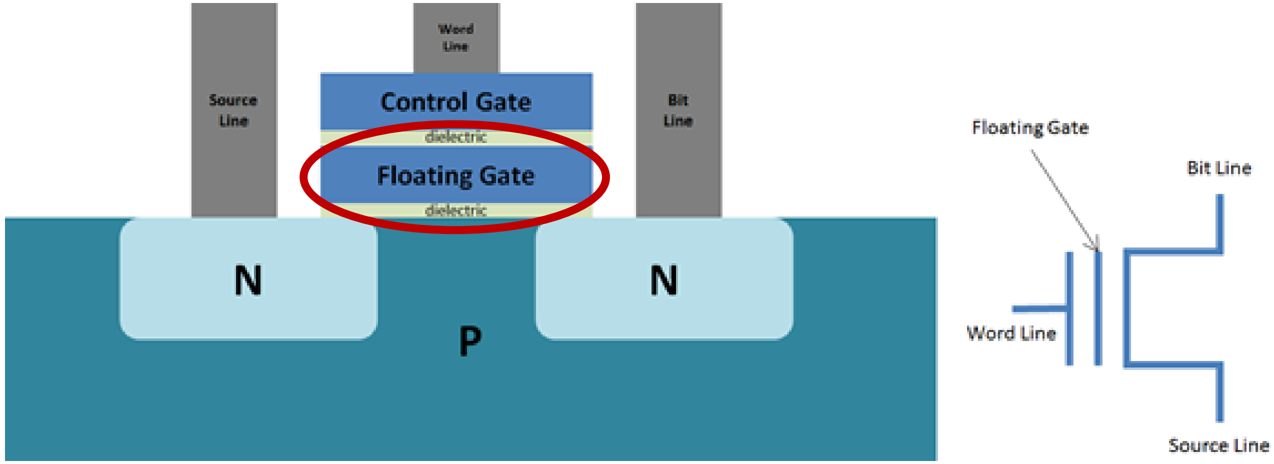
ROM

- High-density
- Reliable
- Low-cost
- Suitable for high production with stable code

Source: Intel Corporation

Flash Memory Cell

- Transistor with floating gate
  - The floating gate is insulated all around with an oxide layer
  - Electrons trapped in the floating gate can remain for up to years

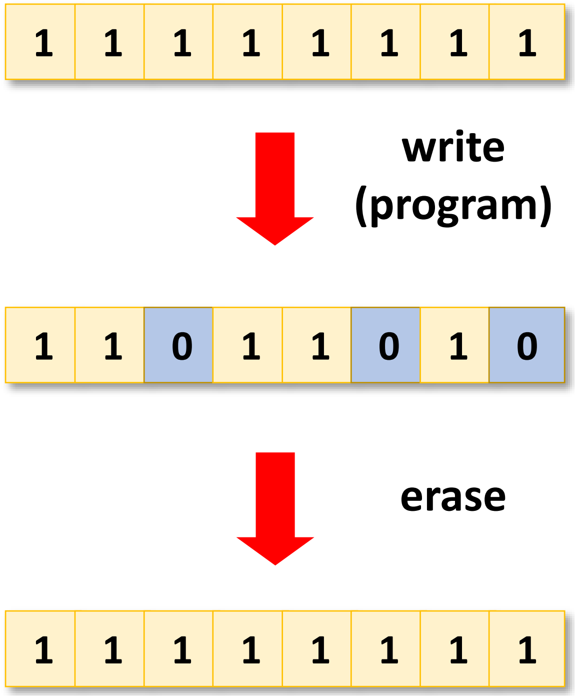


http://www.thenandflash.com

Flash Memory Characteristics

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

- Bulk erase
  - Program unit:
    - NOR: byte or word
    - NAND: page
  - Erase unit: ___________



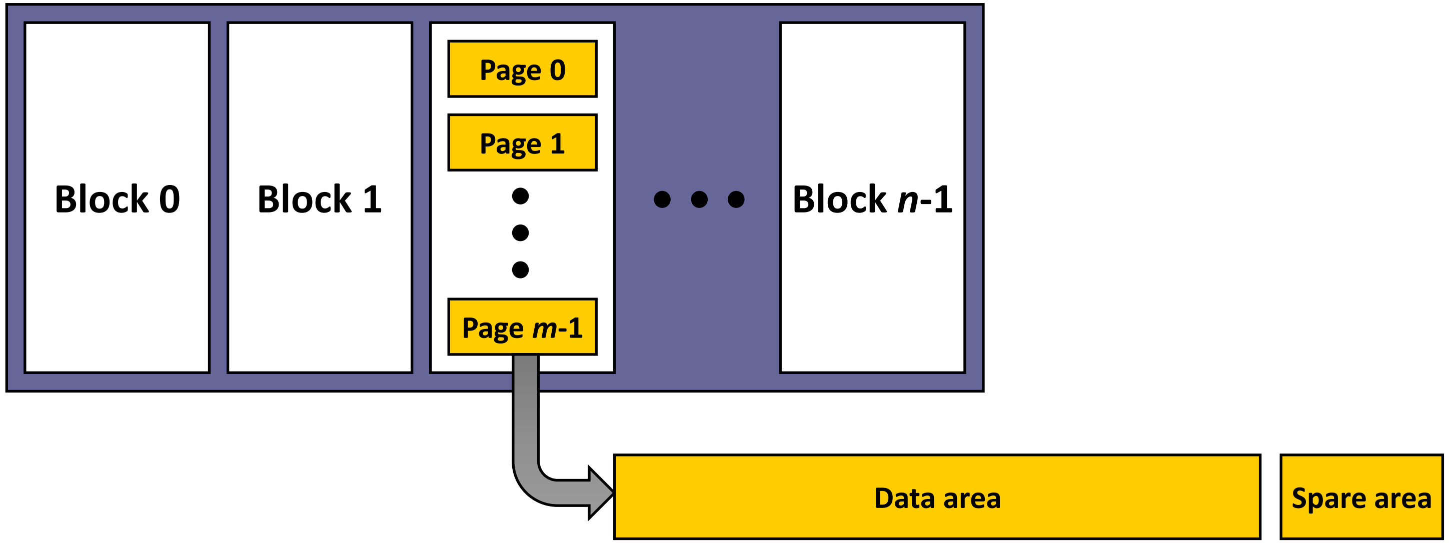
1 1 1 1 1 1 1 1

1 1 0 1 1 0 1 0

1 1 1 1 1 1 1 1

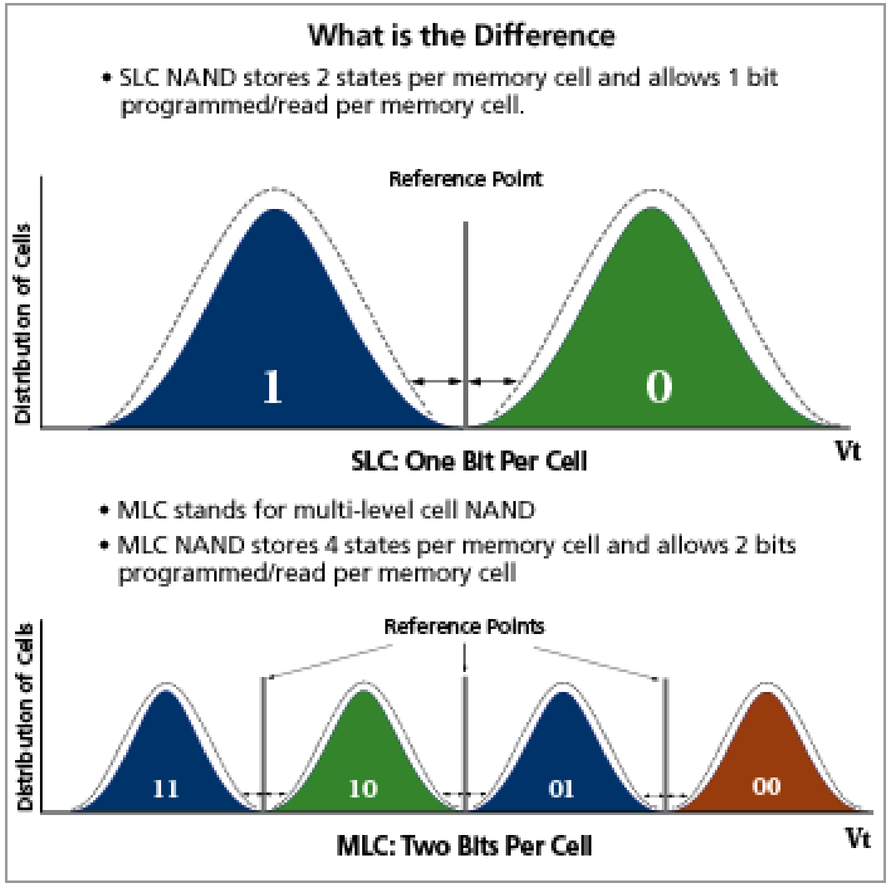
Logical View of NAND Flash

- 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)



NAND Flash Types

- SLC NAND
  - Single Level Cell
  - 1 bit/cell
- MLC NAND
  - Multi Level Cell (misnomer)
  - 2 bits/cell
- TLC NAND
  - Triple Level Cell
  - 3 bits/cell
- 3D NAND



What is the Difference

- SLC NAND stores 2 states per memory cell and allows 1 bit programmed/read per memory cell.

SLC: One Bit Per Cell

- MLC stands for multi-level cell NAND
- MLC NAND stores 4 states per memory cell and allows 2 bits programmed/read per memory cell

MLC: Two Bits Per Cell

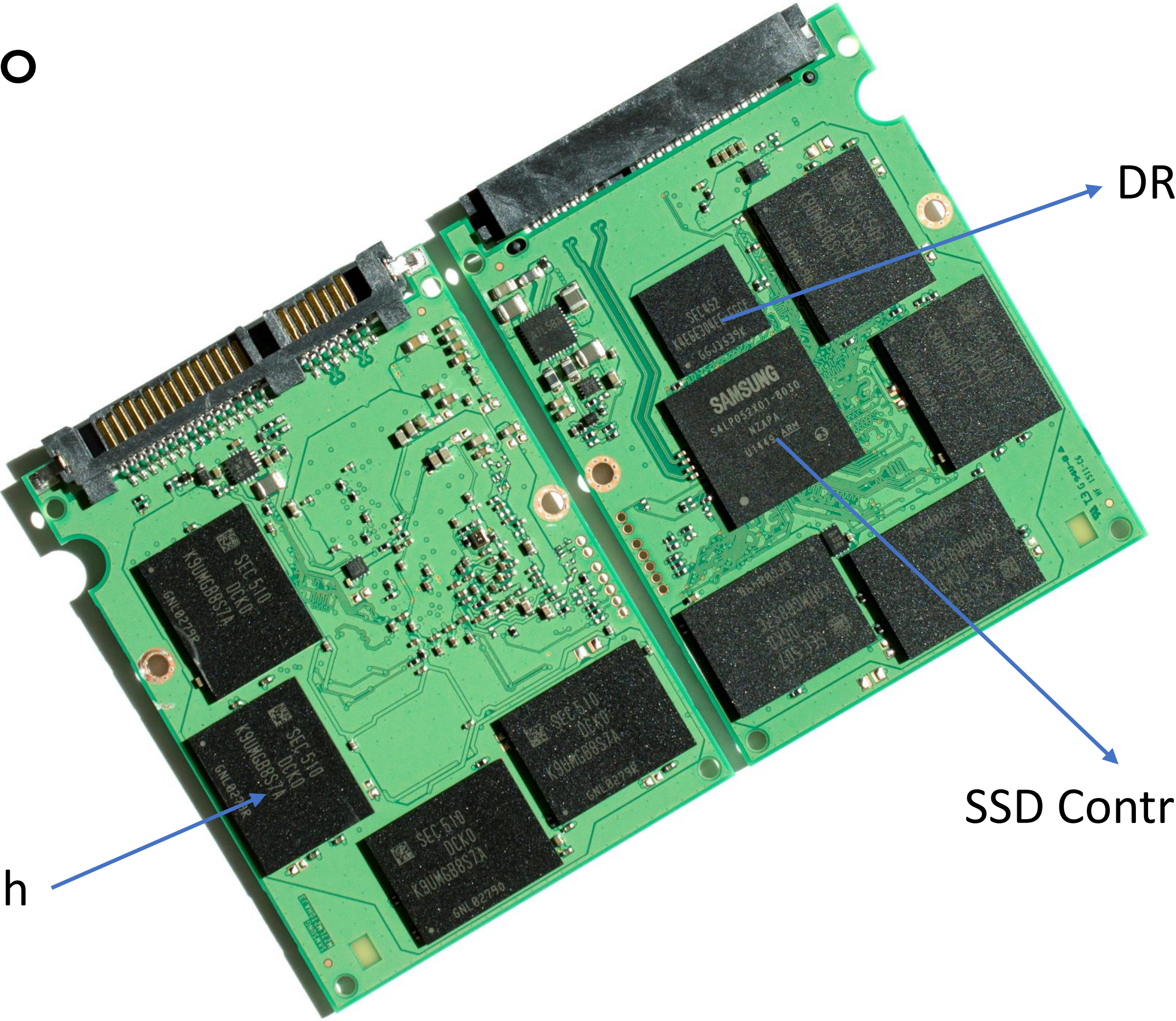
Source: Micron Technology, Inc.

NAND Applications

- Universal Flash Drives (UFDs)
- Flash cards
  - CompactFlash, MMC, SD, Memory stick, …
- Smartphones
  - eMMC (Embedded MMC)
  - UFS (Universal Flash Storage)
- SSDs (Solid State Drives)
- Other embedded devices
  - MP3 players, Digital TVs, Set-top boxes, Car navigators, …

Anatomy of an SSD

- Samsung 850 Evo



DRAM

SSD Controller

NAND Flash

http://www.anandtech.com/show/9451/the-2tb-samsung-850-pro-evo-ssd-review

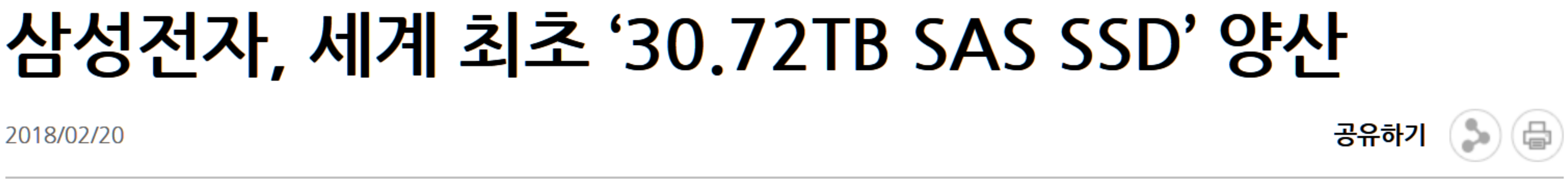
HDDs vs. SSDs

Feature	SSD (Samsung)	HDD (Seagate)
Model	MZ-75E2T0B (850 Evo)	ST2000LM003 (SpinPoint M9T)
Capacity	2TB (128Gb 32-Layer 3D V-NAND TLC x 16 die/channel x 8 channels)	2TB (3 Discs, 6 Heads, 5400 RPM)
Form factor	2.5”, 66g	2.5”, 130g
DRAM	2 GB	32 MB
Host interface	SATA-3 (6.0 Gbps)	SATA-3 (6.0 Gbps)
Power consumption (Active / Idle / Sleep)	3.7, 4.7 W / 0.5 W / 0.05 W	2.3 W / 0.7 W / 0.18 W
Performance **850 Evo**[1]: Sequential: 128KB/QD2 Random: 4KB/QD32 **M9T**[2]: Sequential: 2MB Random: 4KB	Sequential read: 544 MB/s Sequential write: 520 MB/s Random read: 97,687 IOPS Random write: 89,049 IOPS Random read: 11,335 IOPS (QD1) Random write: 38,433 IOPS (QD1)	Sequential read: 124 MB/s Sequential write: 124 MB/s Random read: 56 IOPS Random write: 98 IOPS Power-on to ready: 3.5 sec Average seek: 12/14 ms Average latency: 5.6 ms
Price[3]	940,910 won (470won/GB)	175,900 won (88won/GB)

[1] http://www.tomshardware.com/reviews/samsung-850-evo-850-pro-2tb-ssd,4205.html

[1] http://www.storagereview.com/samsung_spinpoint_m9t_hard_drive_review [3] http://www.enuri.com (As of May. 26, 2020)

State-of-the-Art @ 2018



삼성전자, 세계 최초 '30.72TB SAS SSD' 양산

2018/02/20

- 2.5"
- 1TB V-NAND x32
- 4GB DRAM x 10

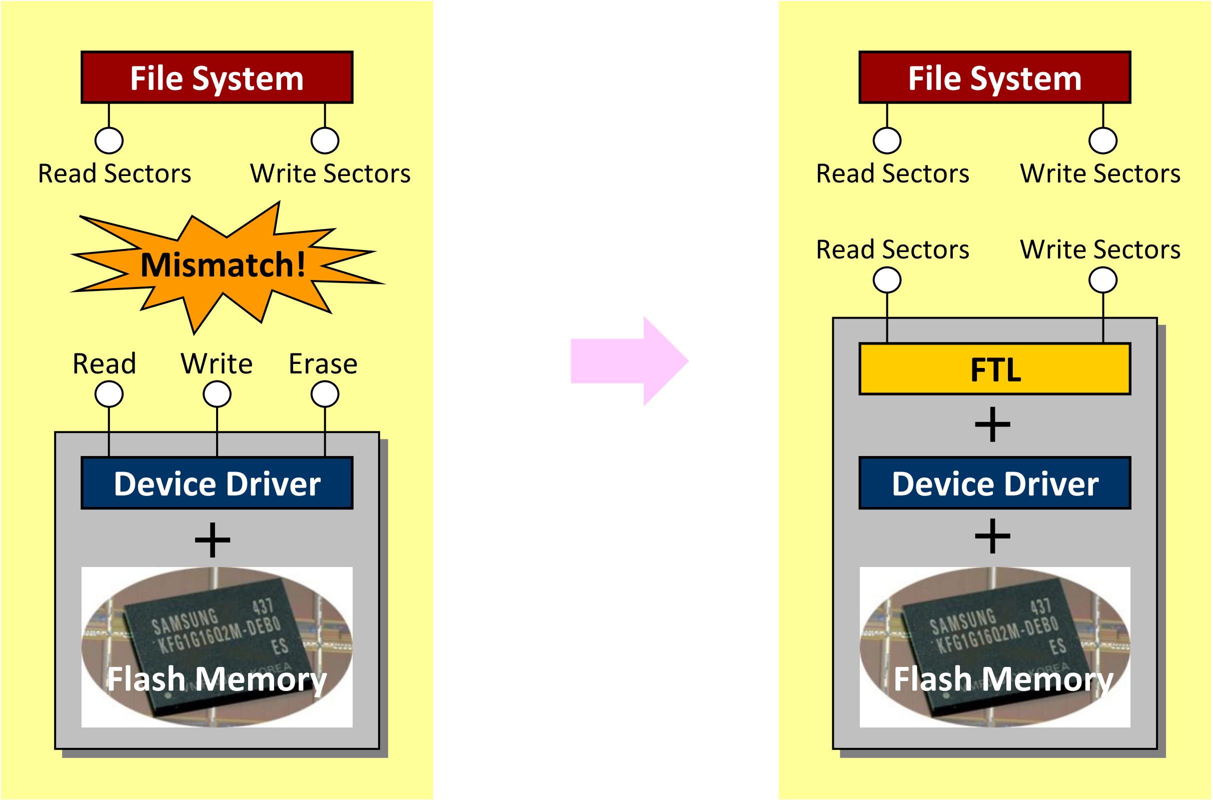
- Sequential read: 2100MB/s
- Sequential write: 1700MB/s

NAND Constraints

- No in-place update
  - Require sector remapping (or address translation)
- Bit errors
  - Require the use of error correction codes (ECCs)
- Bad blocks
  - Factory-marked and run-time bad blocks
  - Require bad block remapping
- Limited program/erase cycles
  - < 100K for SLCs, < 3K for MLCs, < 1K for TLCs
  - Require wear-leveling

Flash Translation Layer (FTL)

- A software layer to make NAND flash fully emulate traditional block devices (e.g., disks)



File System

Read Sectors

Write Sectors

Mismatch!

Read

Write

Erase

Device Driver

+

Flash Memory

File System

Read Sectors

Write Sectors

Read Sectors

Write Sectors

FTL

+

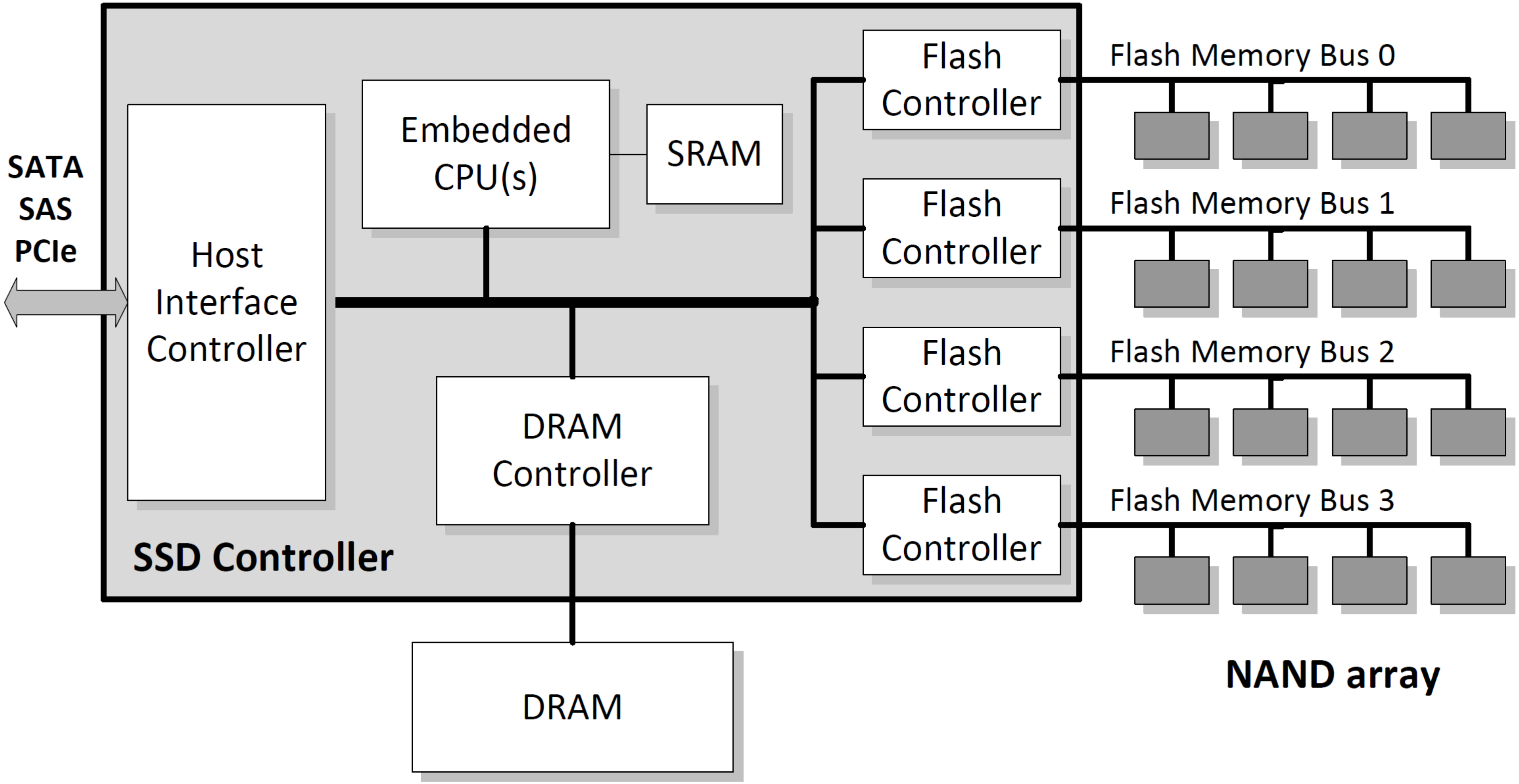
Device Driver

+

Flash Memory

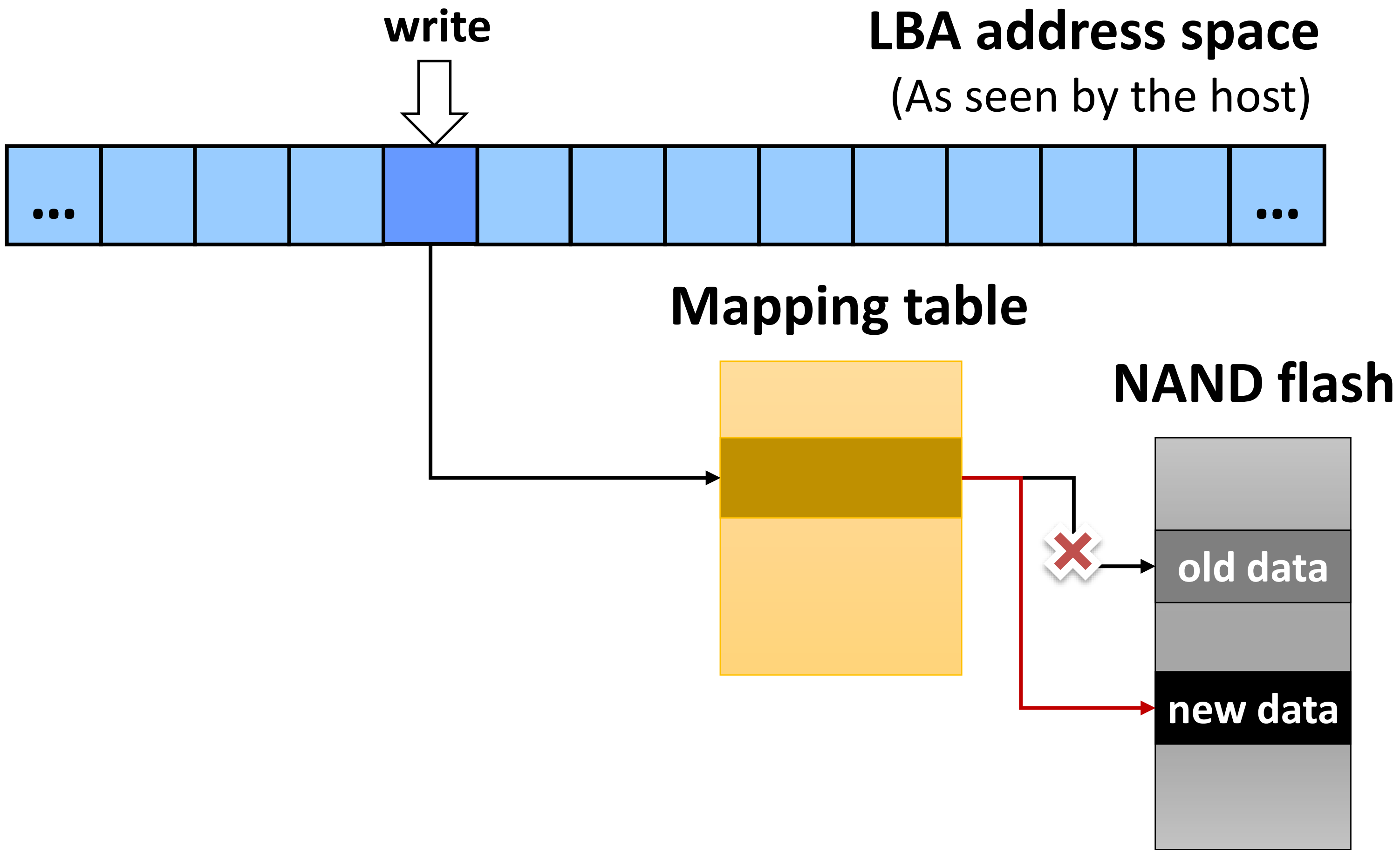
Source: Zeen Info. Tech.

SSD Internals



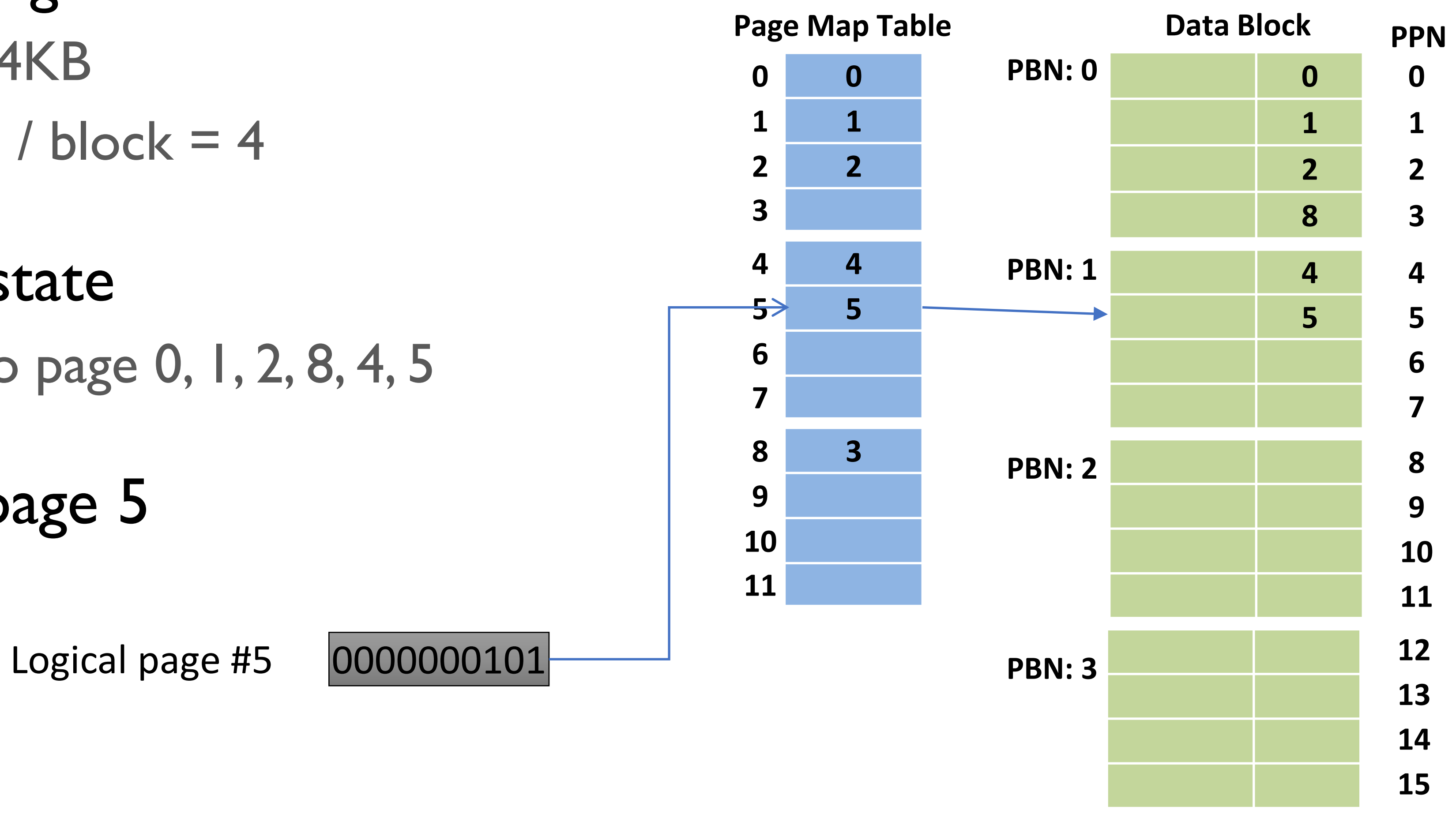
Address Mapping

- Required since flash pages cannot be overwritten



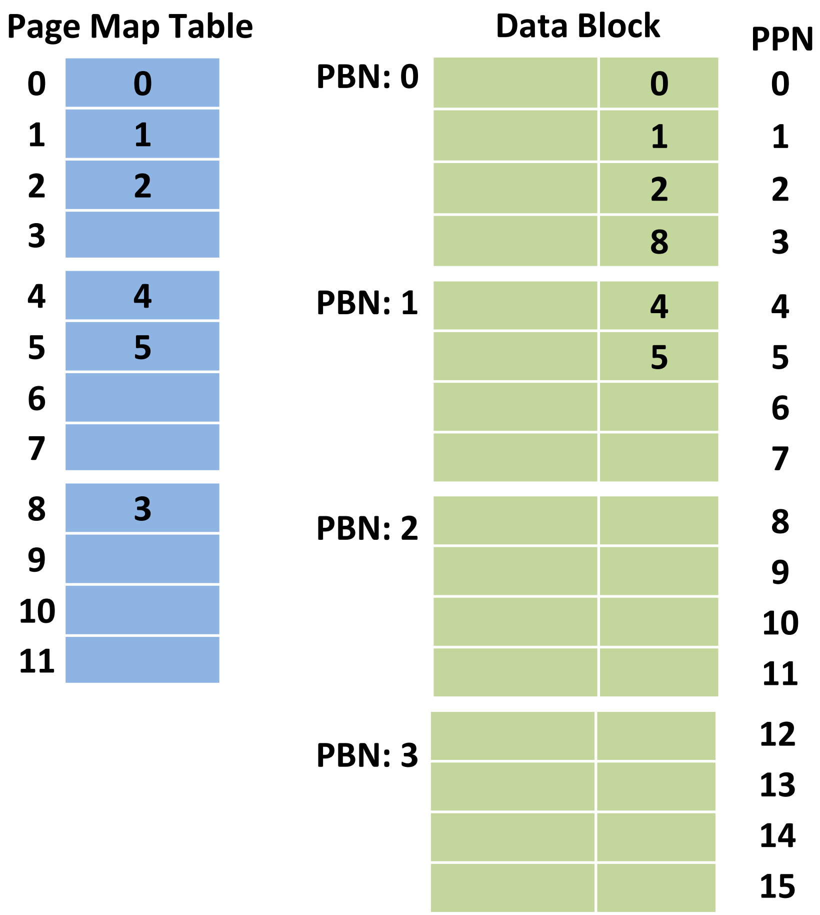
Example: Page Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- Reading page 5



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

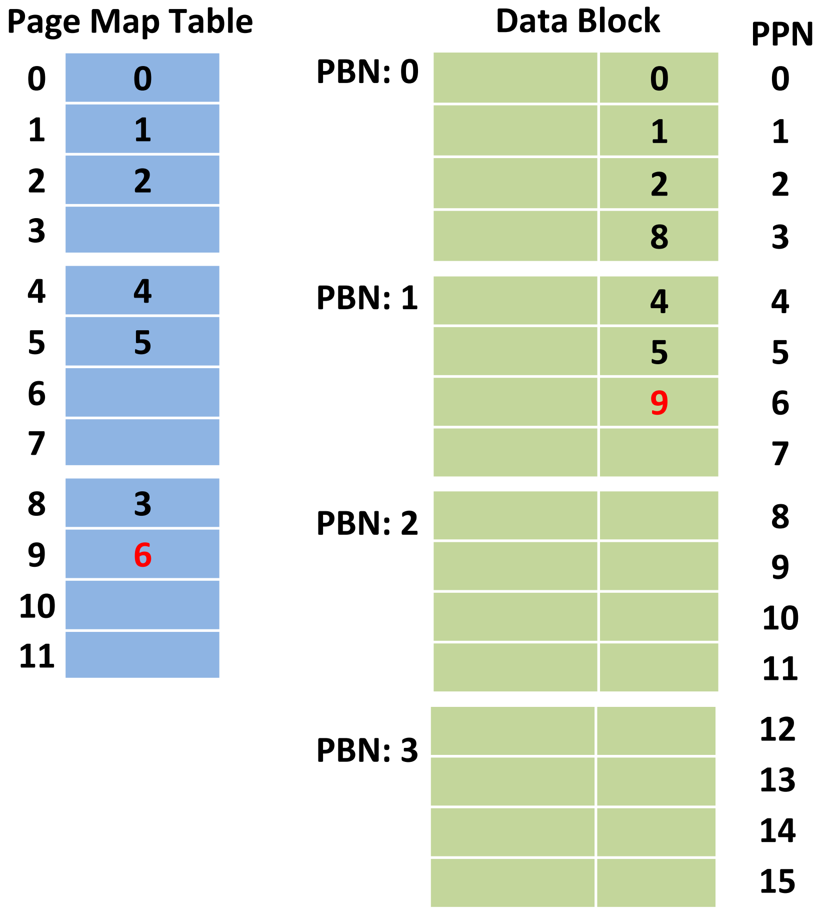
0	0
1	1
2	2
3	
4	4
5	5
6	
7	
8	3
9	
10	
11	

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

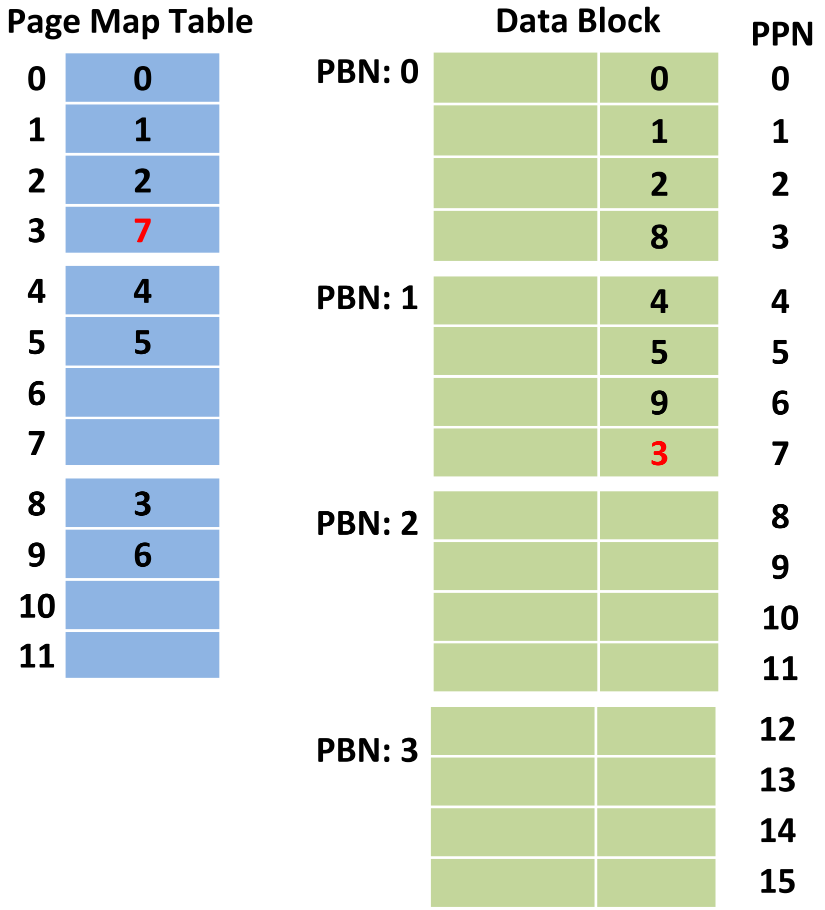
0	0
1	1
2	2
3	
4	4
5	5
6	
7	
8	3
9	6
10	
11	

Data Block

			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

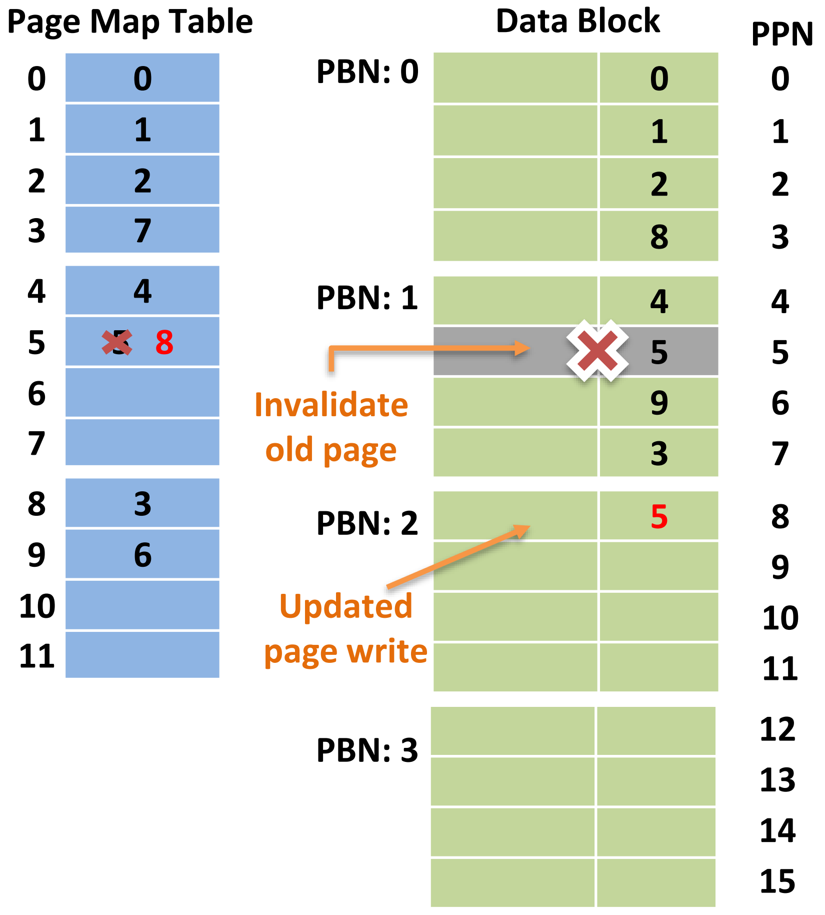
0	0
1	1
2	2
3	7
4	4
5	5
6	
7	
8	3
9	6
10	
11	

Data Block

			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 Map Table

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

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	5	8
		9
		10
		11
PBN: 3		12
		13
		14
		15

Invalidate old page

Updated page write

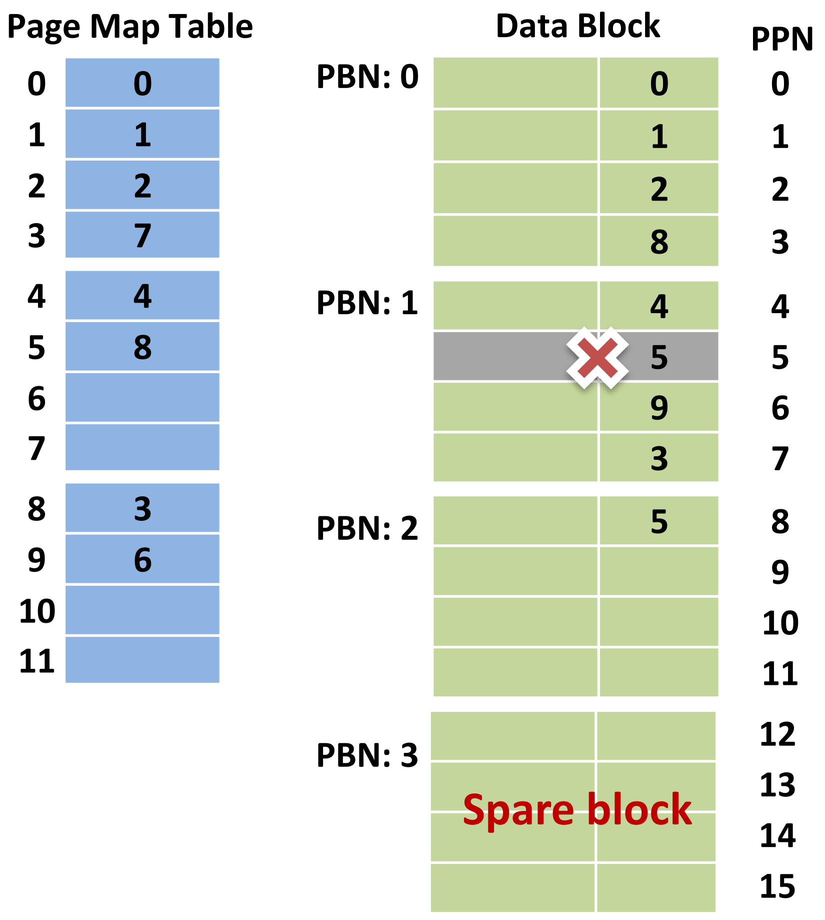
Garbage Collection

- 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

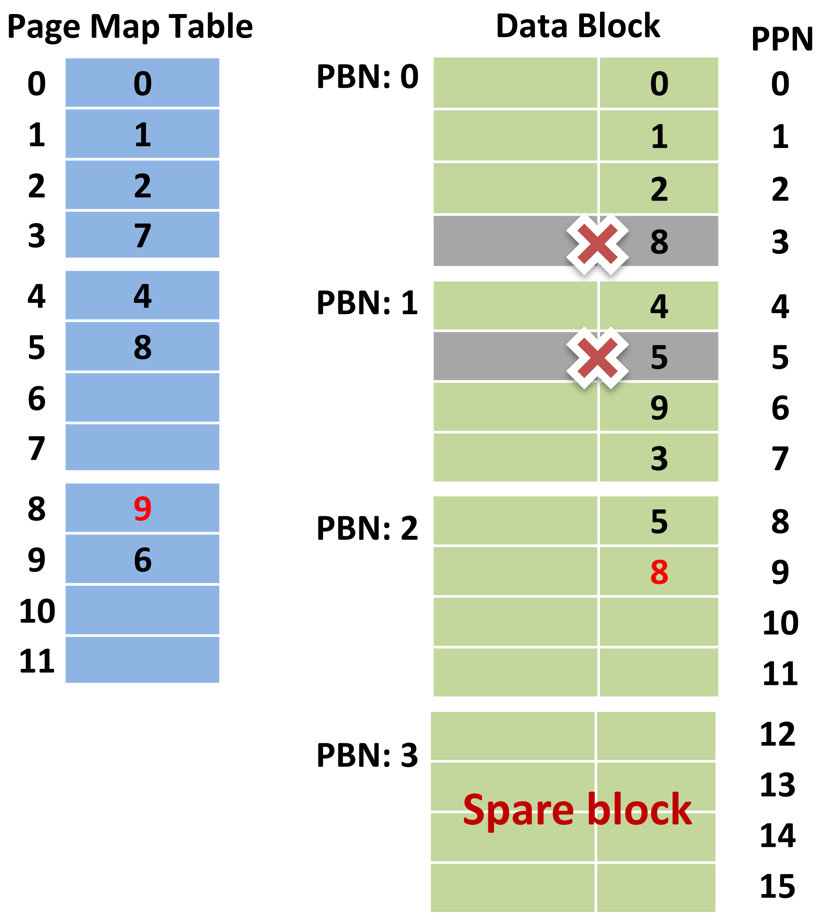
0	0
1	1
2	2
3	7
4	4
5	8
6	
7	
8	3
9	6
10	
11	

	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		5	8
			9
			10
			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



Page Map Table

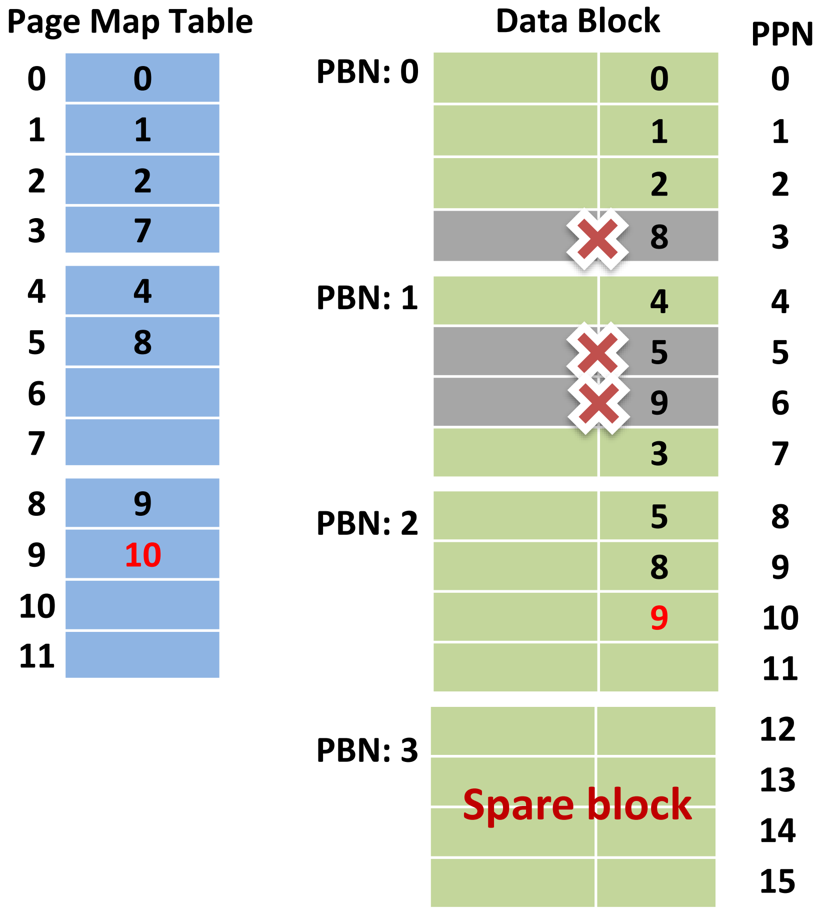
0	0
1	1
2	2
3	7
4	4
5	8
6	
7	
8	9
9	6
10	
11	

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		5	8
		8	9
			10
			11
PBN: 3	Spare block		12
			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



Page Map Table

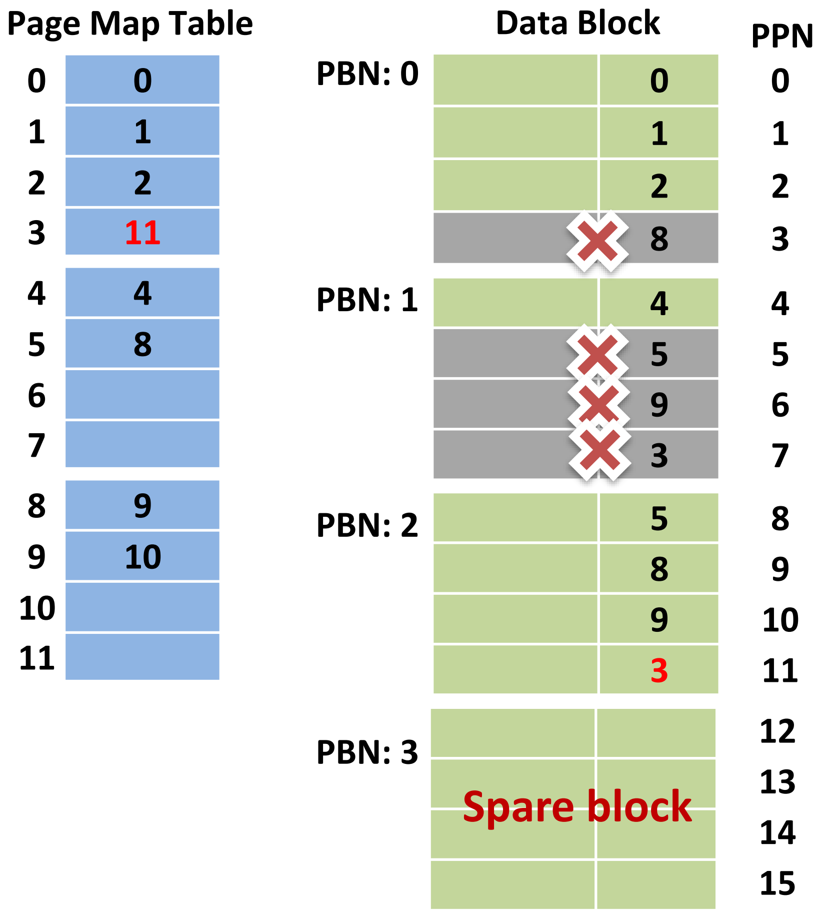
0	0
1	1
2	2
3	7
4	4
5	8
6	
7	
8	9
9	10
10	
11	

Data Block

			PPN
PBN: 0		0	0
		1	1
		2	2
	✕	8	3
PBN: 1		4	4
	✕	5	5
	✕	9	6
		3	7
PBN: 2		5	8
		8	9
		9	10
			11
PBN: 3	Spare block		12
			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



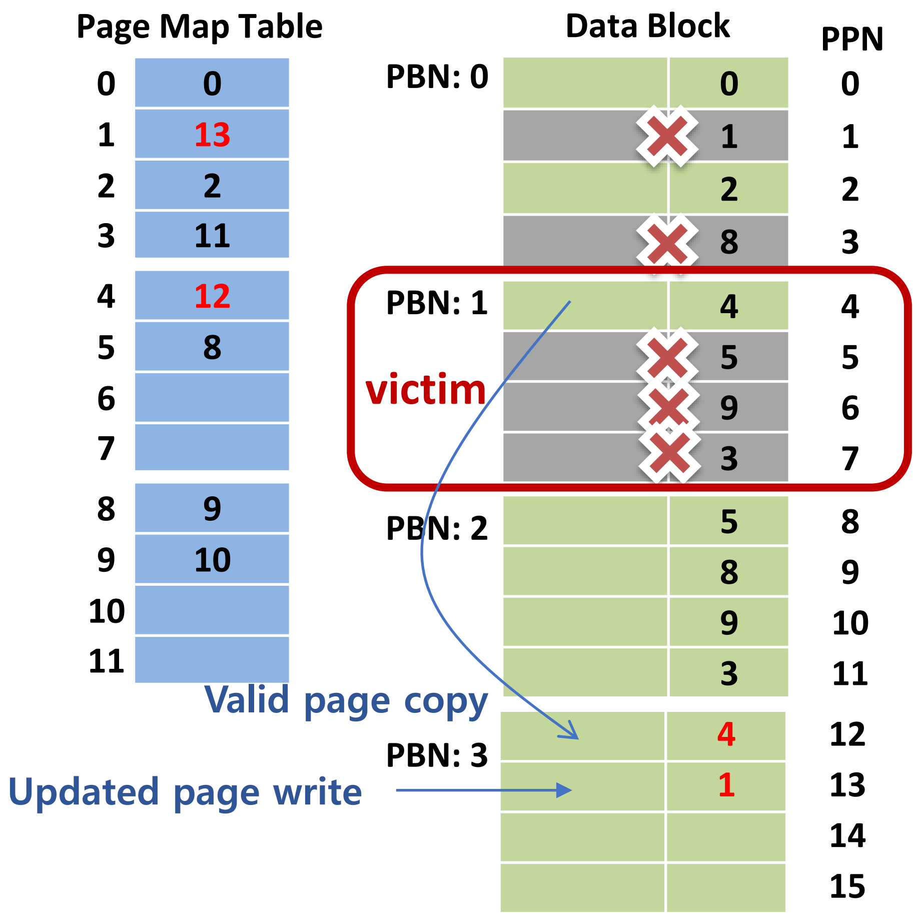
Page Map Table

0	0
1	1
2	2
3	11
4	4
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	4
	✗	5	5
	✗	9	6
	✗	3	7
PBN: 2		5	8
		8	9
		9	10
		3	11
PBN: 3	Spare block		12
			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



Page Map Table

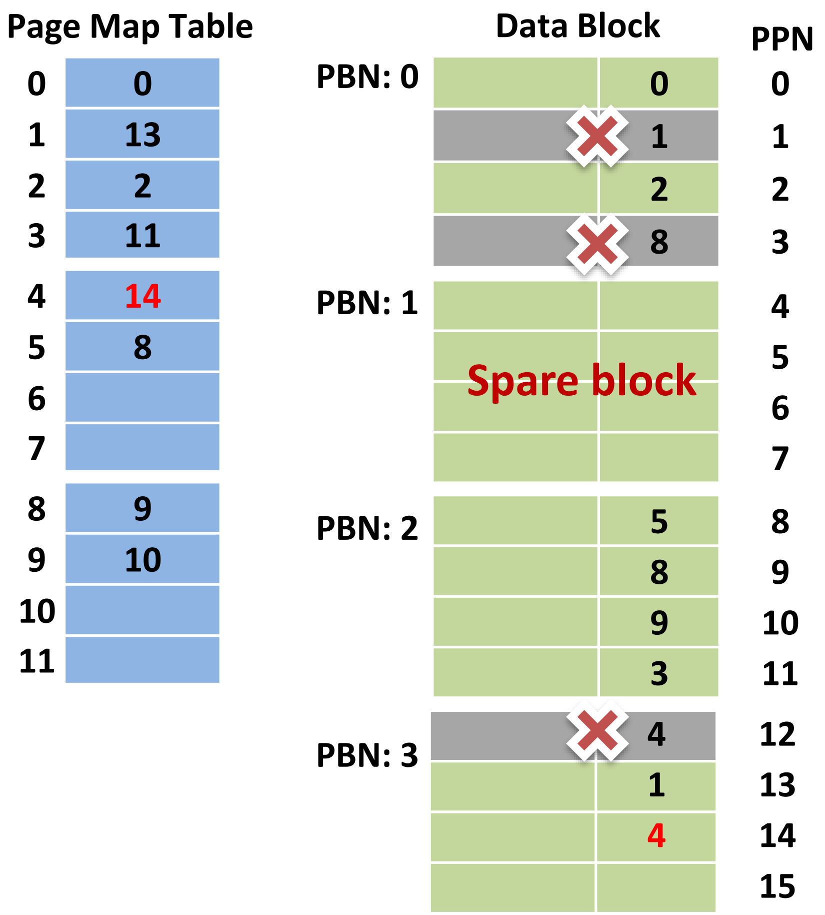
0	0
1	13
2	2
3	11
4	12
5	8
6	
7	
8	9
9	10
10	
11	

Data Block

			PPN
PBN: 0		0	0
	✗	1	1
		2	2
	✗	8	3
PBN: 1		4	4
	✗	5	5
	✗	9	6
	✗	3	7
PBN: 2		5	8
		8	9
		9	10
		3	11
PBN: 3		4	12
		1	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



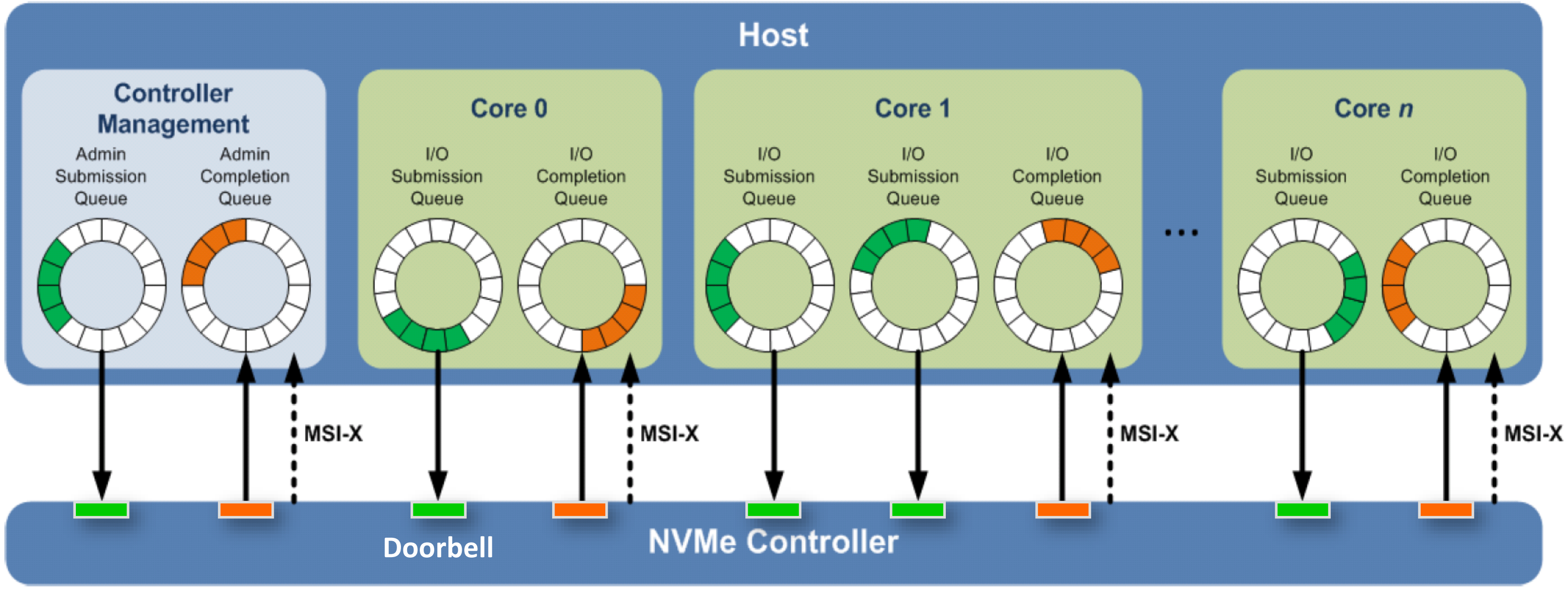
Page Map Table

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	Spare block	4
		5
		6
		7
PBN: 2	5	8
	8	9
	9	10
	3	11
PBN: 3	✗ 4	12
	1	13
	4	14
		15

NVMe SSD

- PCIe-based (PCIe Gen. 3: 1GB/s per lane, up to 32 lanes)
- Deep queue: 64K commands per queue, up to 64K queues
- Streamlined command set: only 13 required commands
- One register write to issue a command (“doorbell”)



Source: Intel & Samsung

OS Implications

- NAND flash has different characteristics compared to disks
  - No seek time
  - Asymmetric read/write access times
  - No in-place-update
  - Good sequential read/write and random read performance, but bad random write performance
  - Wear-leveling
  - …

  - Traditional operating systems have been optimized for disks. What should be changed?

SSD Support in OS

- Turn off "defragmentation" for SSDs
- New "TRIM" command
  - Remove-on-delete
- Simpler I/O scheduler
- Align file system partition with SSD layout
- Flash-aware file systems (e.g., F2FS in Linux)
- Larger block size (4KB)
- New "multi-stream" interface
- …

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
  - No in-place-update
  - Much slower program/erase operations
  - Erase unit > read/write unit
  - Bit errors
  - Limited lifetime etc.
- Software support is essential for performance and reliability!