P&S Modern SSDs

Address Translation & Garbage Collection

Dr. Jisung Park
Prof. Onur Mutlu
ETH Zürich
Spring 2021
19 May 2021

Recap: What We Have Discussed So Far

- SSD Organization
  - SSD controller
    - Multicore CPU running firmware (Flash Translation Layer, FTL)
    - Per-channel flash controller for ECC and data randomization
  - DRAM: Metadata store (0.1% of SSD capacity)
  - NAND flash packages (channel – die – plane – block – page)

- NAND flash organization & operation
  - Cell → NAND string & Page → Block → Plane → Die
  - Unique characteristics: Erase-before-write, limited endurance, performance & operation-unit asymmetries, …
  - Basic operations: Page Read/Page Program/Block Erasure
  - Advanced operations: Sub-page read, cache read, multi-plane, and program/erase suspend/resume commands

Flash Translation Layer: Overview

- SSD firmware (often referred to as SSD controller)
  - Provides backward compatibility with traditional HDDs
  - By hiding unique characteristics of NAND flash memory

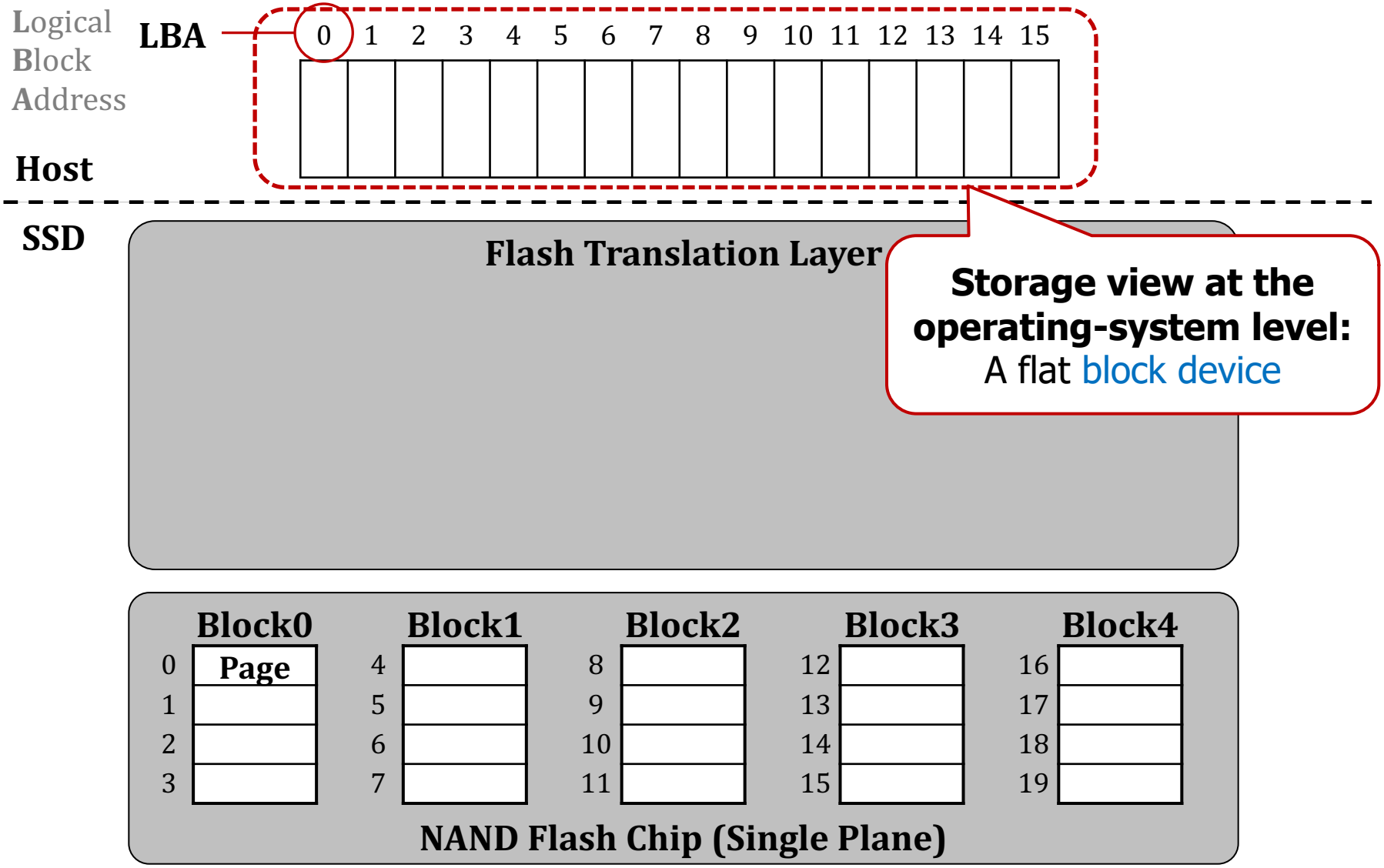
- Responsible for many important SSD-management tasks
  - Address translation + garbage collection
    - Performs out-of-place writes due to erase-before-write property
  - Wear leveling
    - To prolong SSD lifetime by evenly distributing P/E cycles
  - Data refresh
    - Resets transient errors by copying data to a new page(s)
  - I/O scheduling
    - To take full advantage of SSD internal parallelism

Flash Translation Layer: Overview

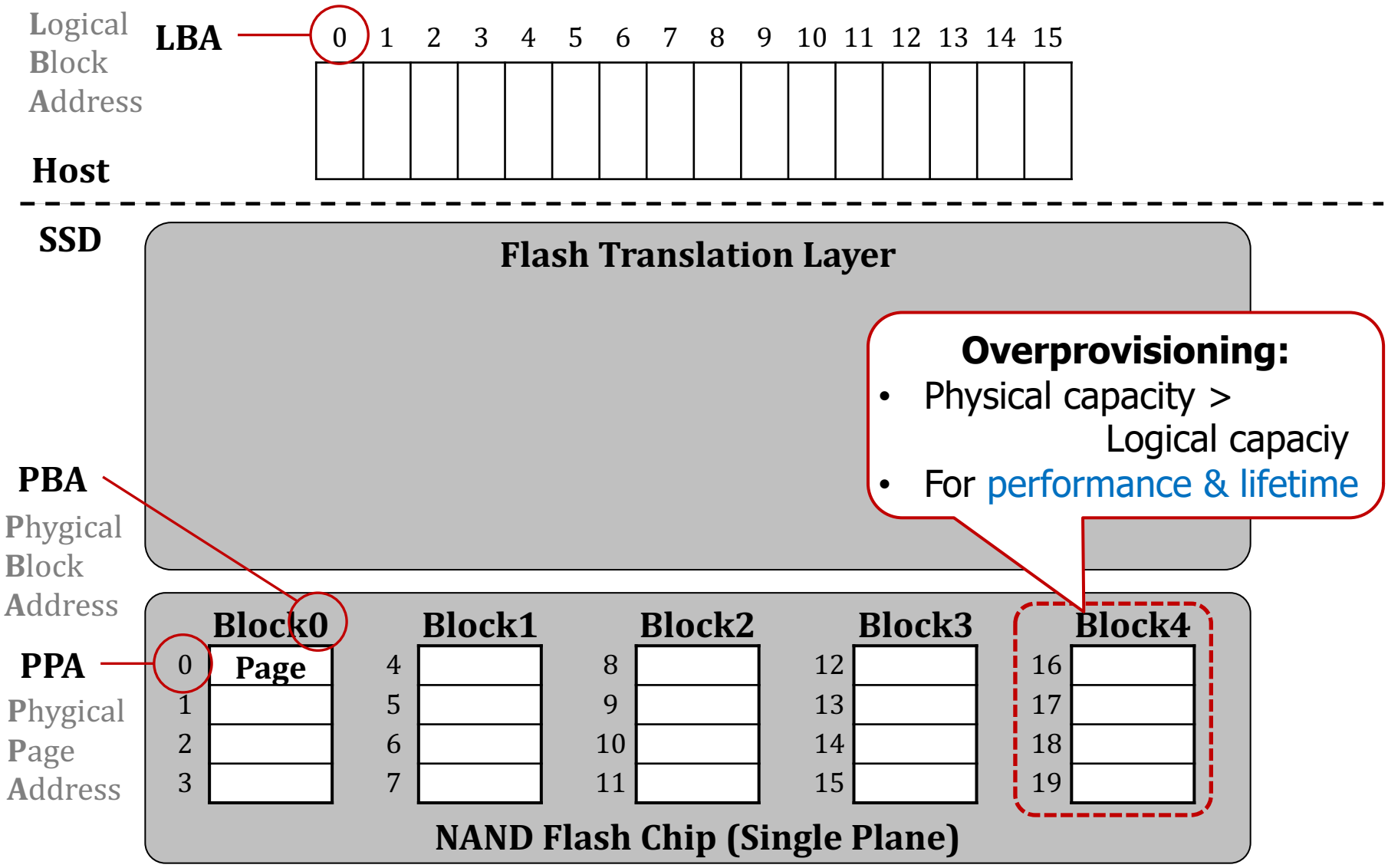
- SSD firmware (often referred to as SSD controller)
  - Provides backward compatibility with traditional HDDs
  - By hiding unique characteristics of NAND flash memory

- Responsible for many important SSD-management tasks
  - Address translation + garbage collection
    - Performs out-of-place writes due to erase-before-write property
  - Wear leveling
    - To prolong SSD lifetime by evenly distributing P/E cycles
  - Data refresh
    - Resets transient errors by copying data to a new page(s)
  - I/O scheduling
    - To take full advantage of SSD internal parallelism

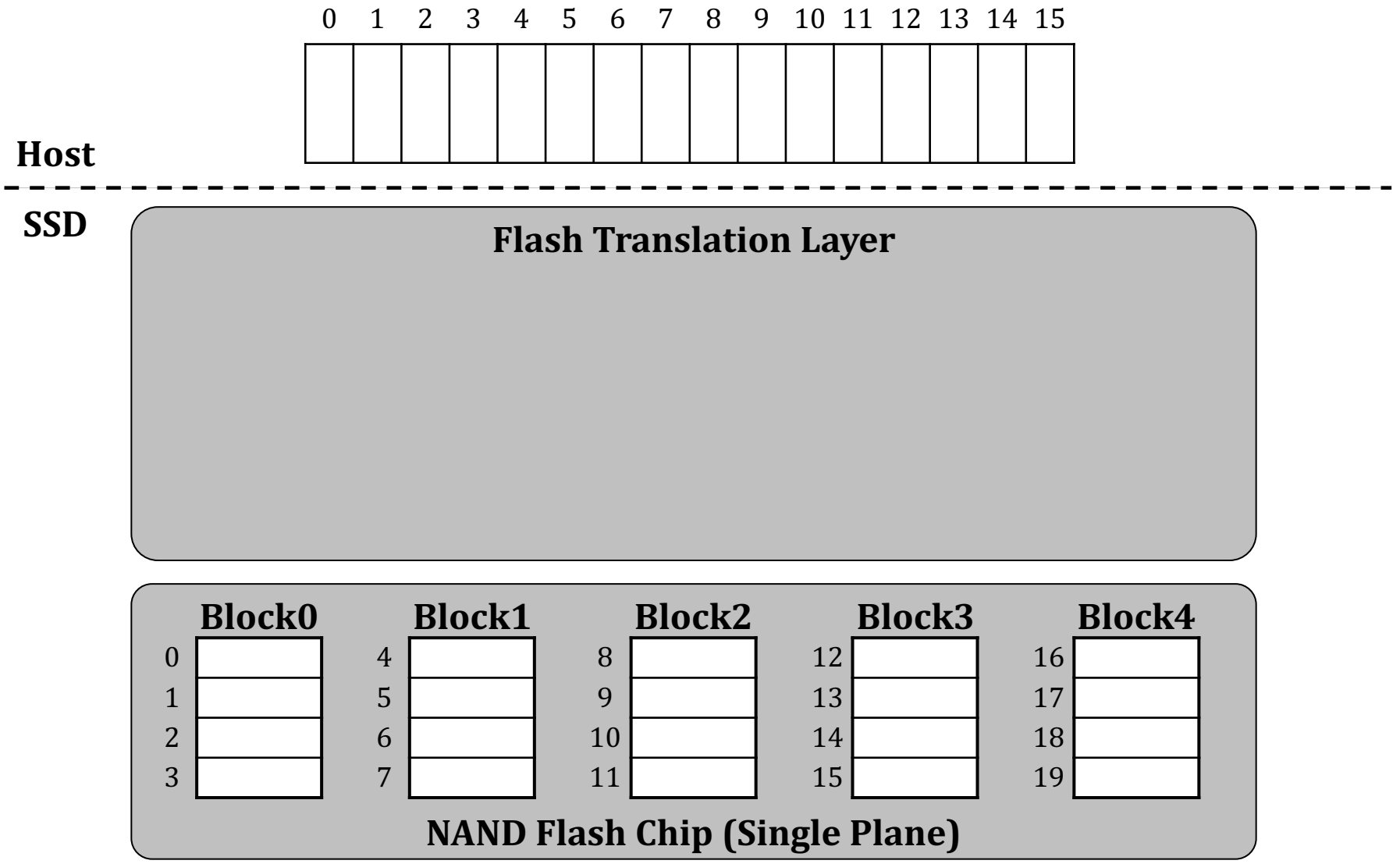
Simple SSD Architecture



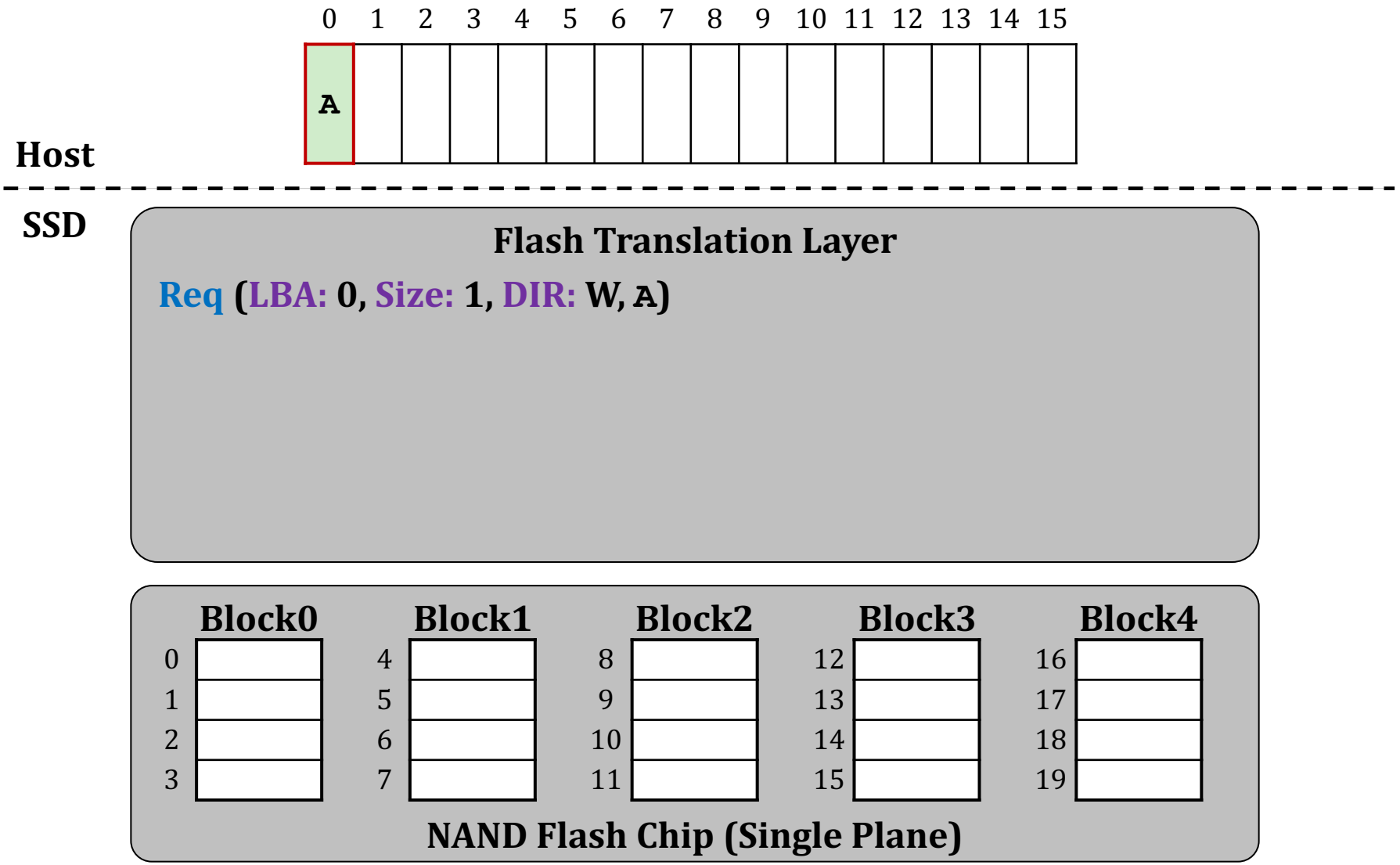
Simple SSD Architecture



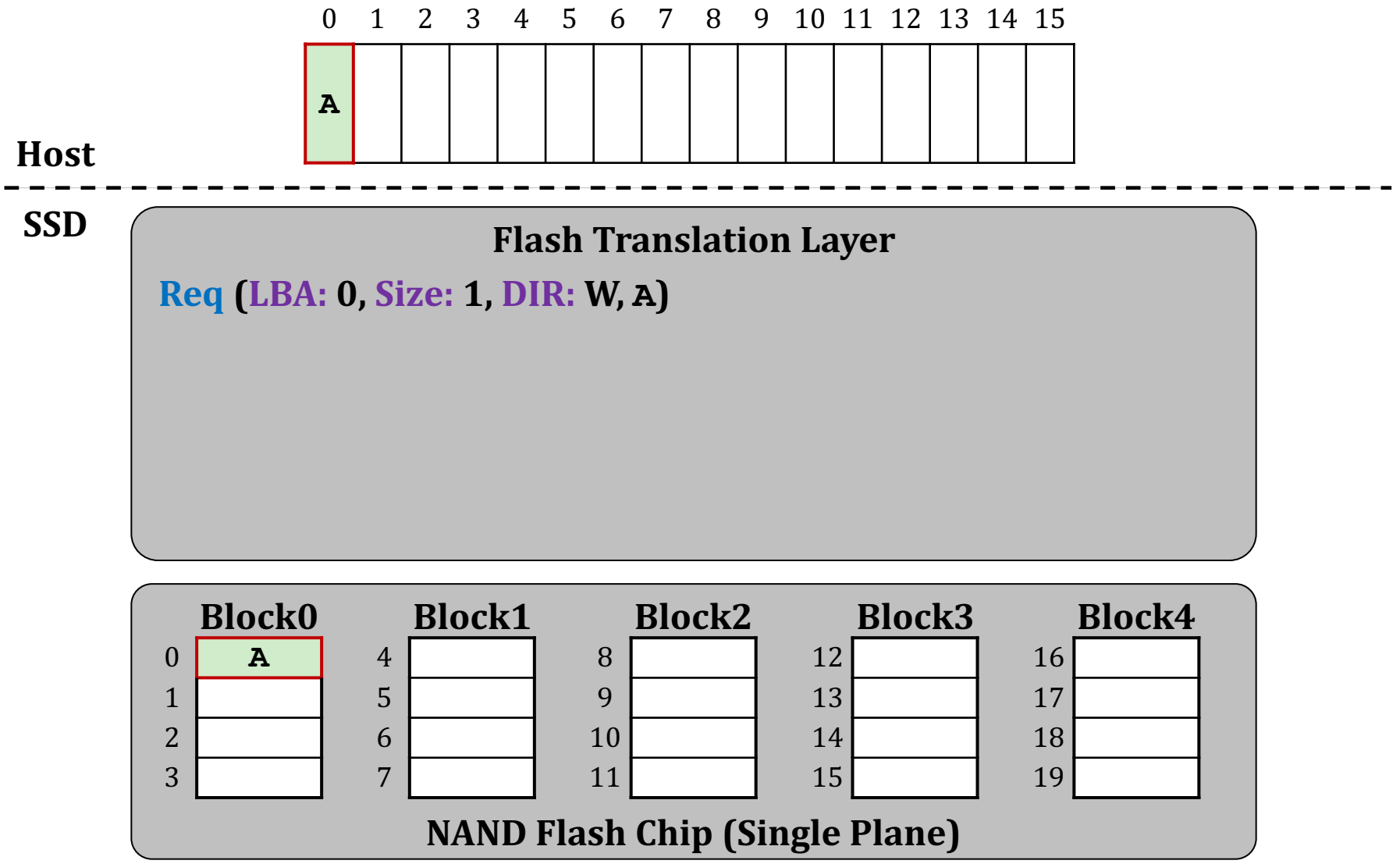
Write Request Handling: Page Write



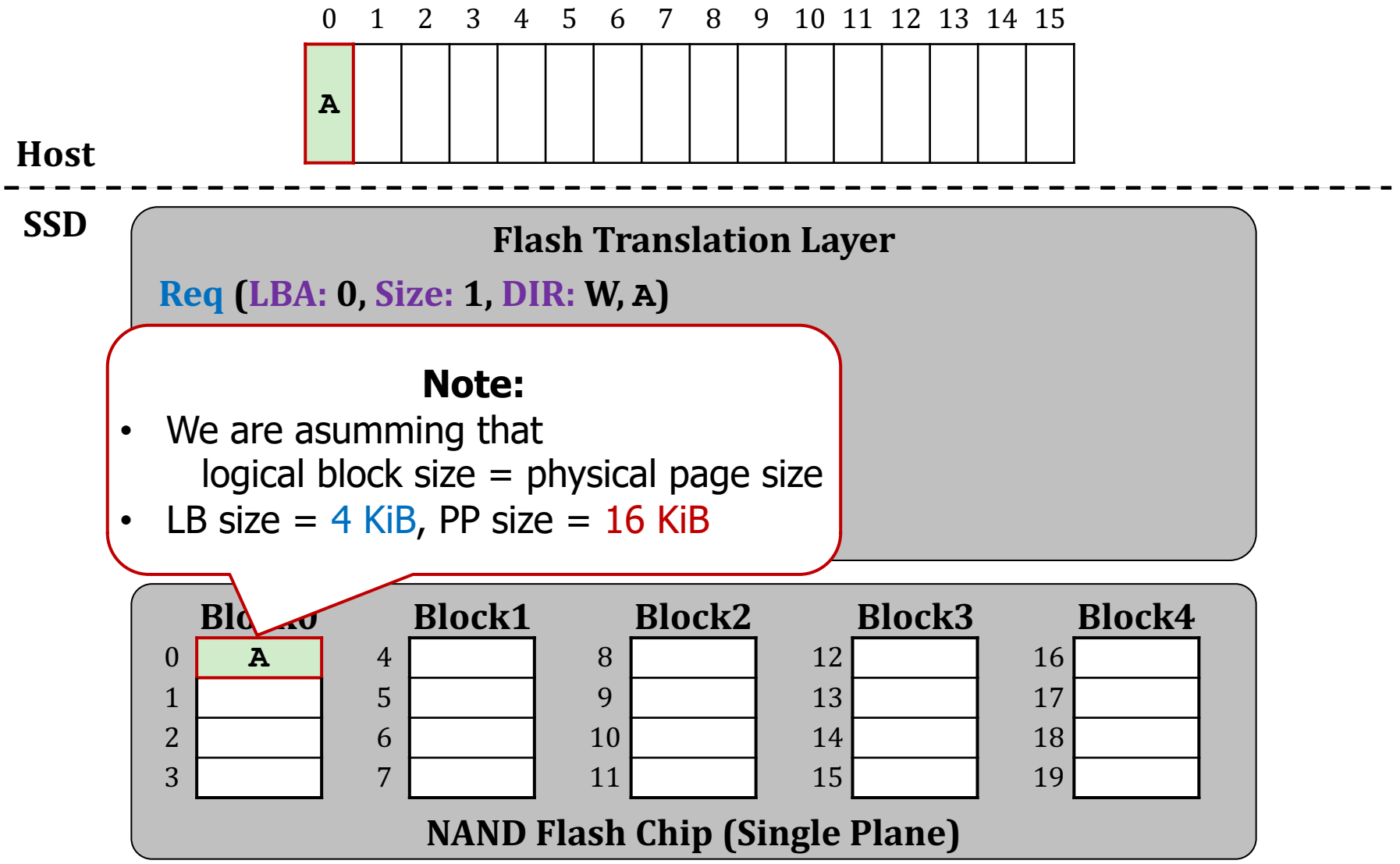
Write Request Handling: Page Write



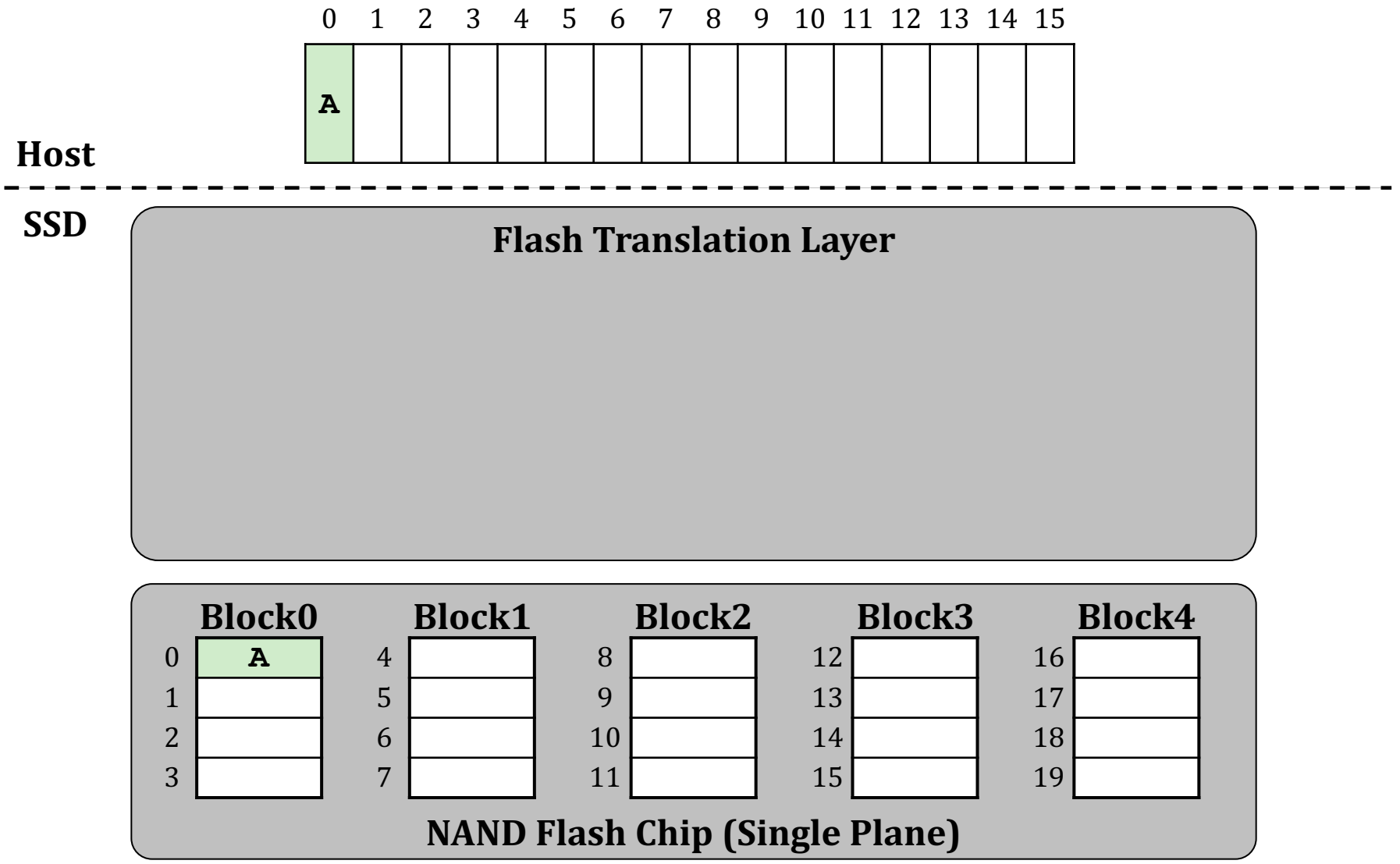
Write Request Handling: Page Write



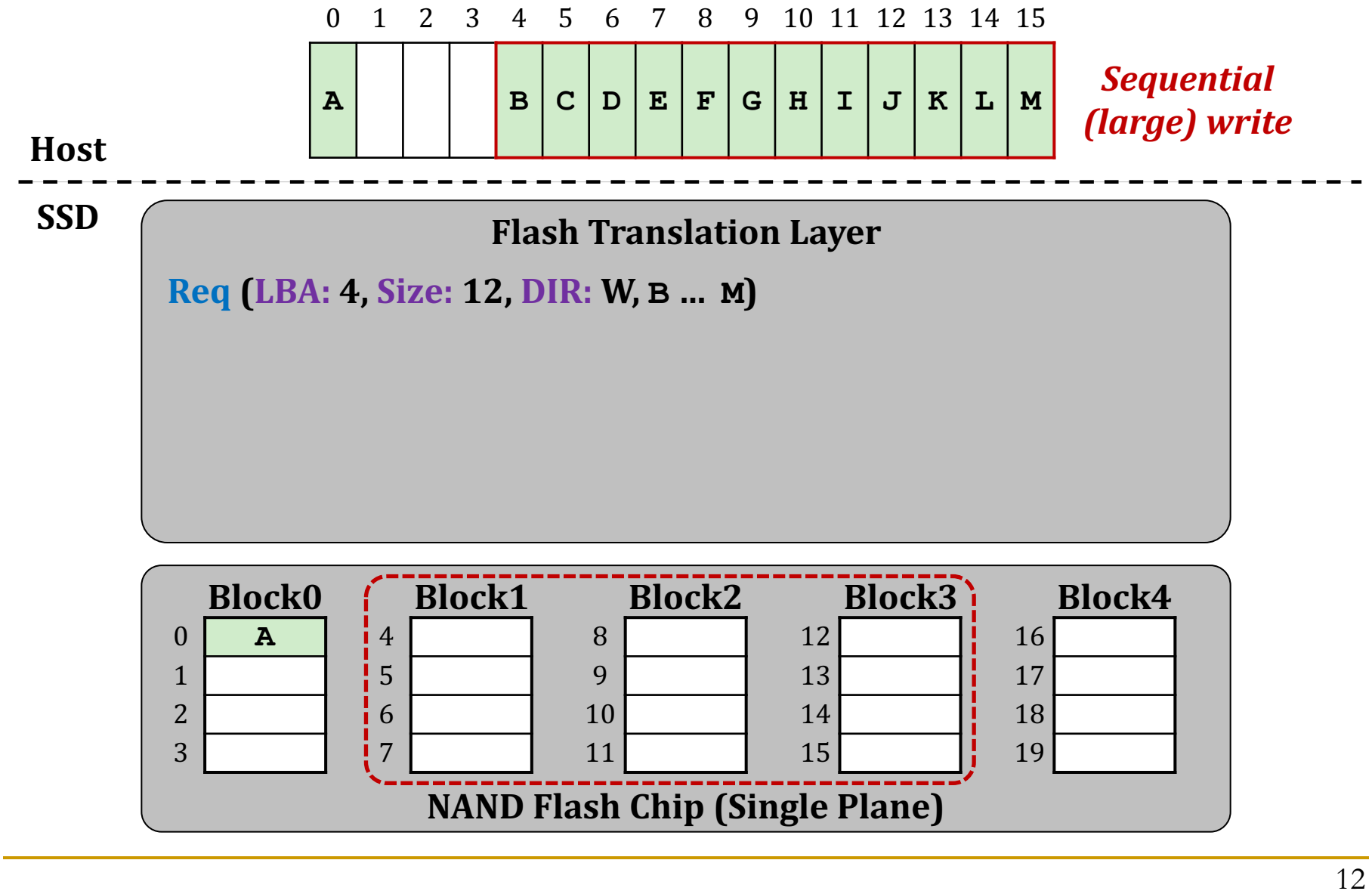
Write Request Handling: Page Write



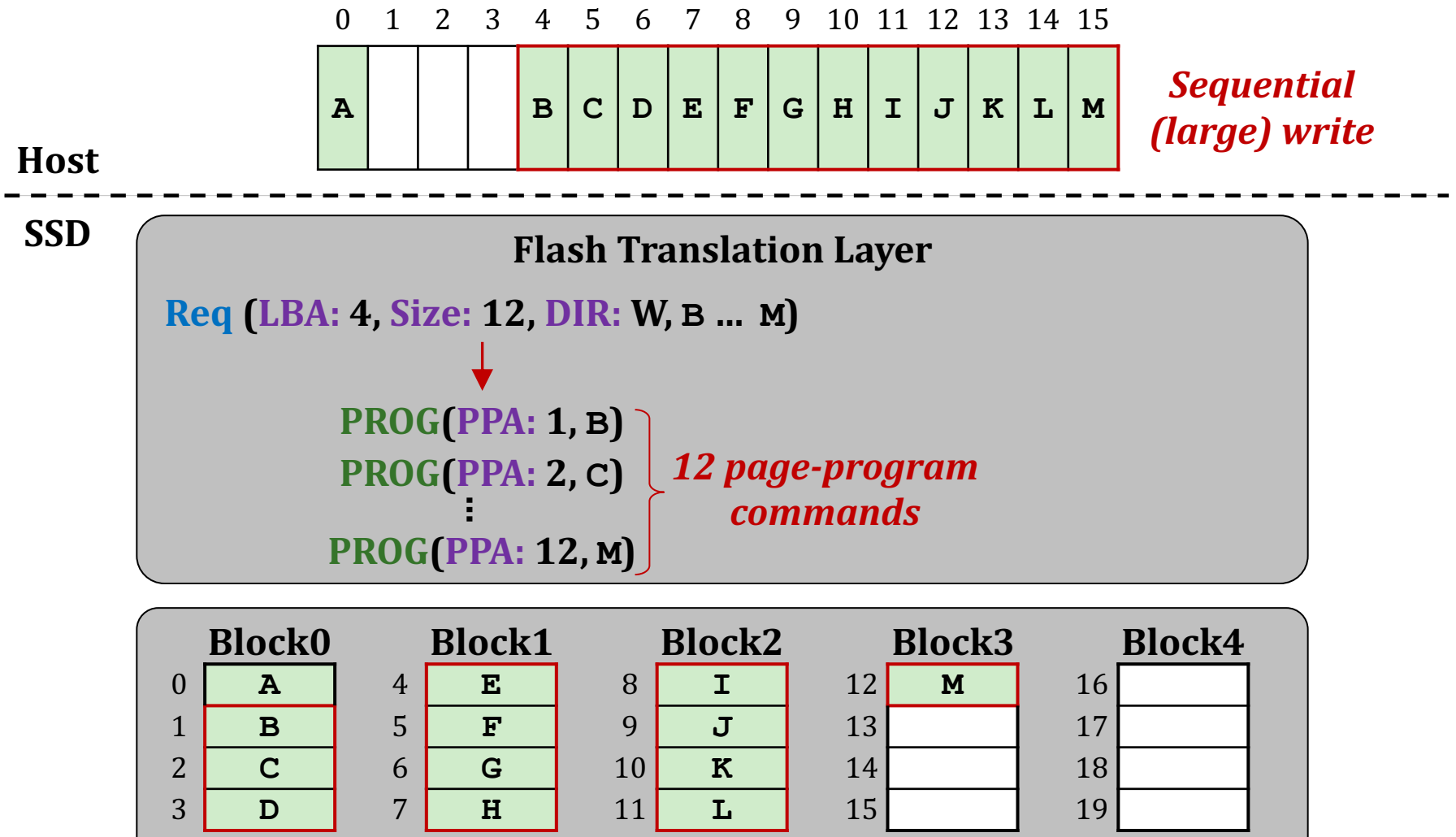
Write Request Handling: Sequential Write



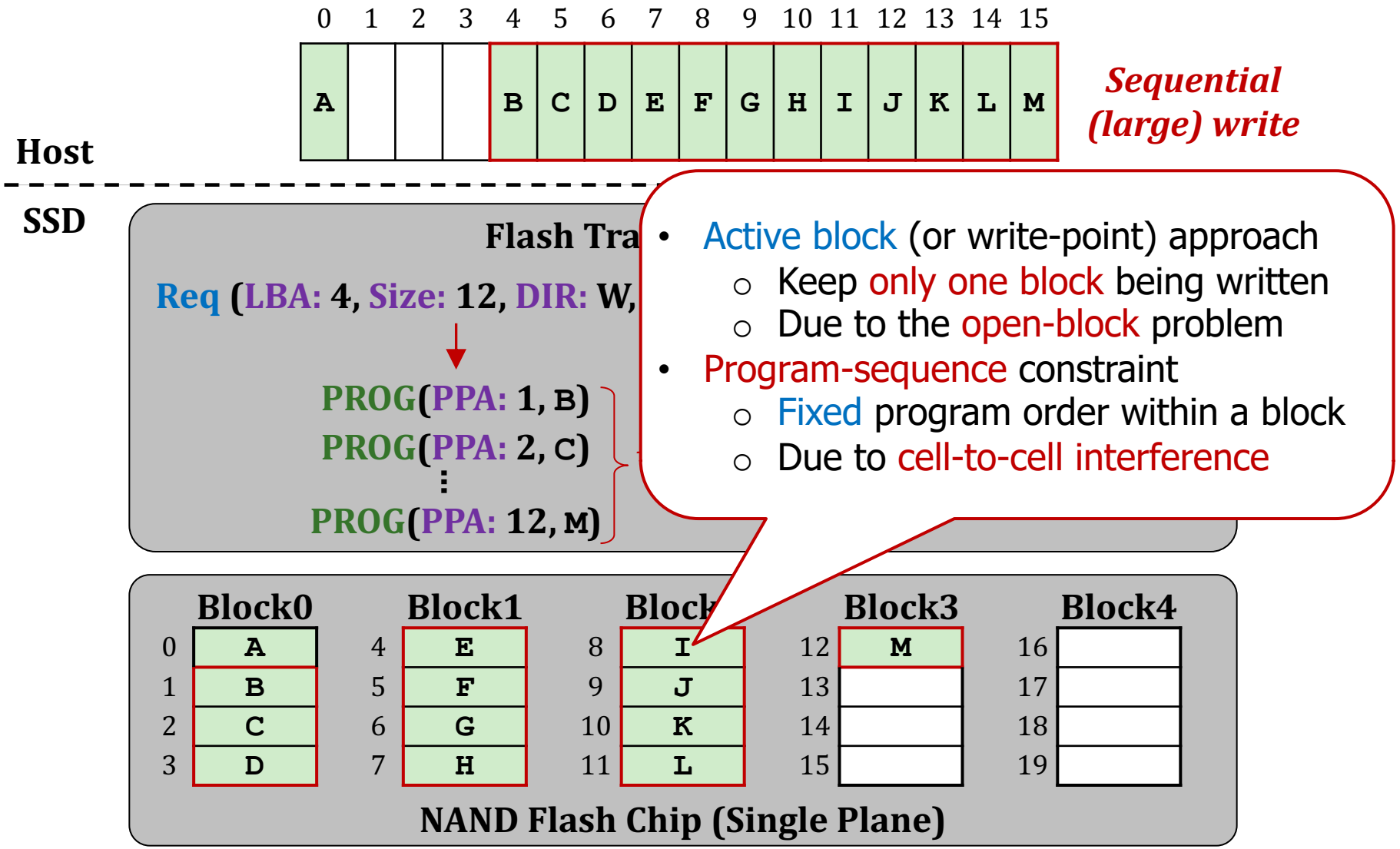
Write Request Handling: Sequential Write



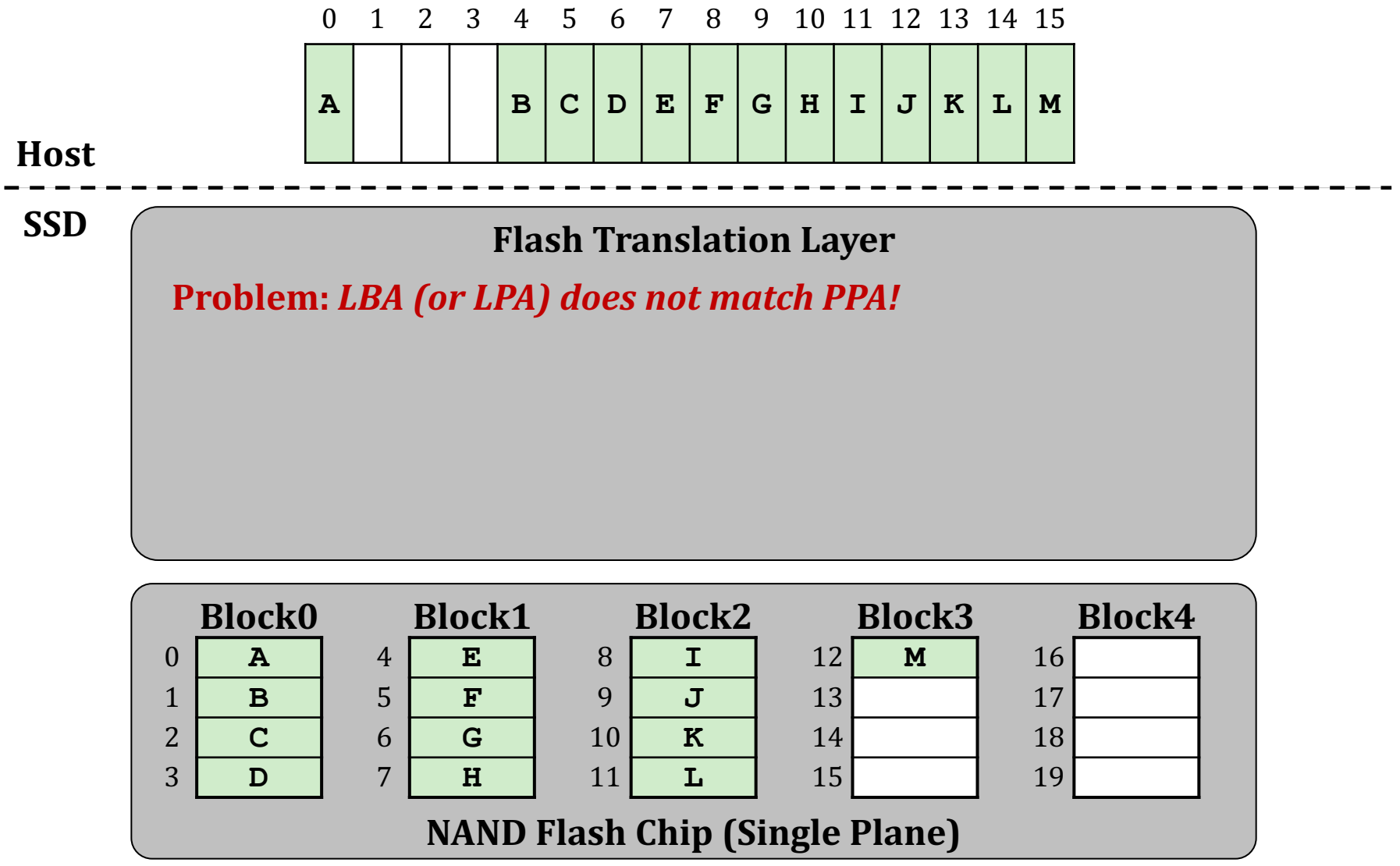
Write Request Handling: Sequential Write



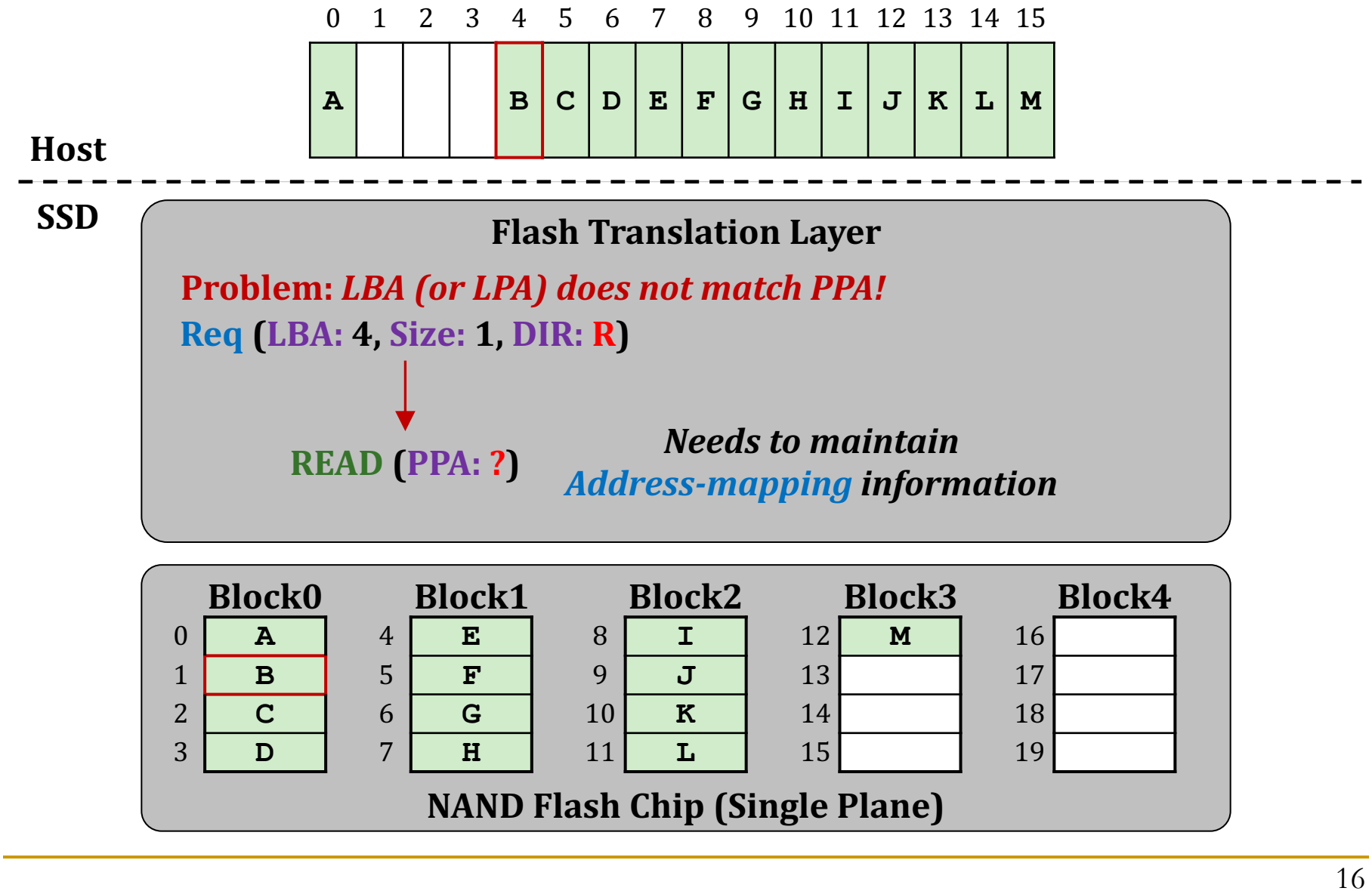
Write Request Handling: Sequential Write



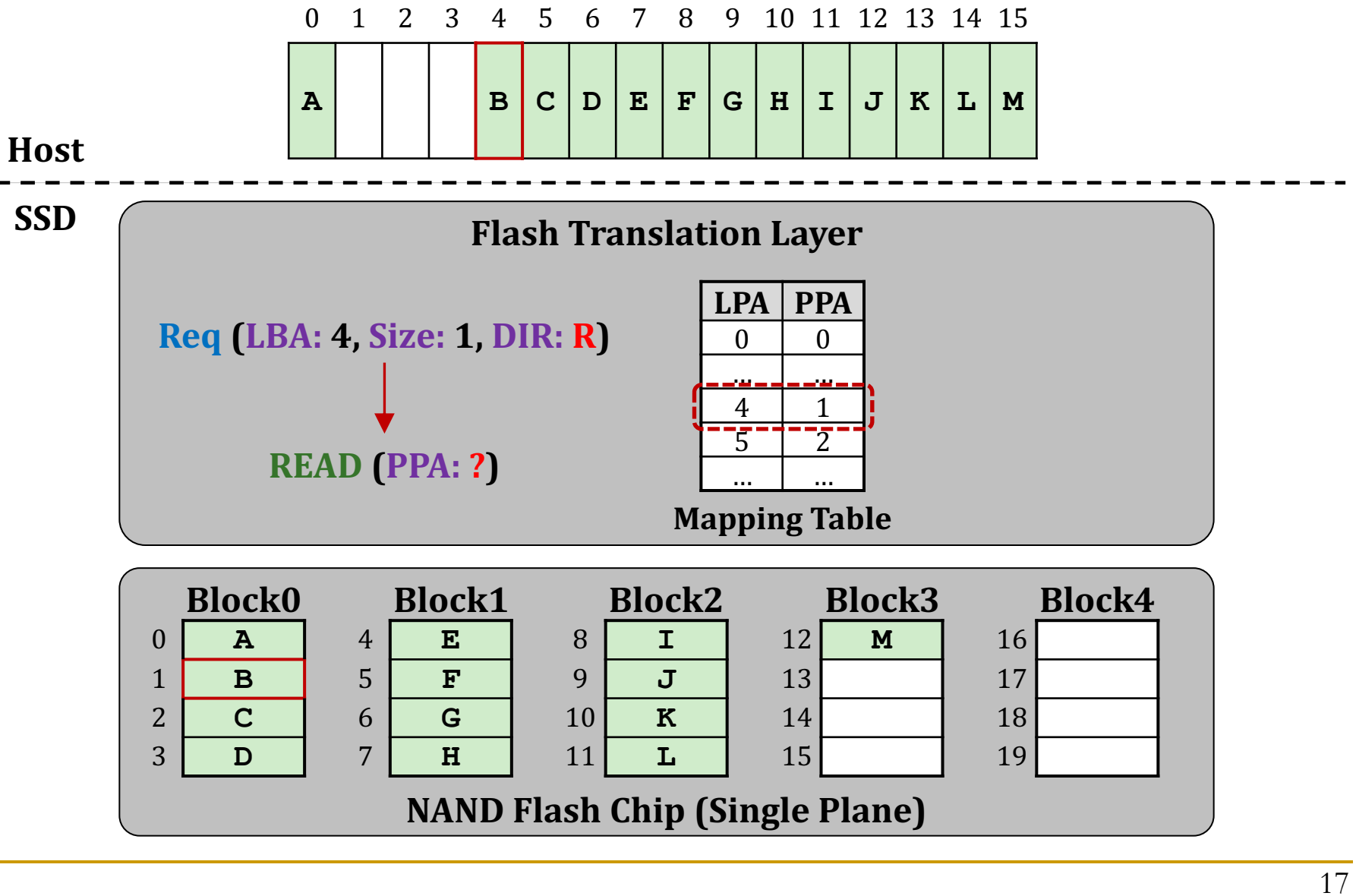
Write Request Handling: Address Mapping



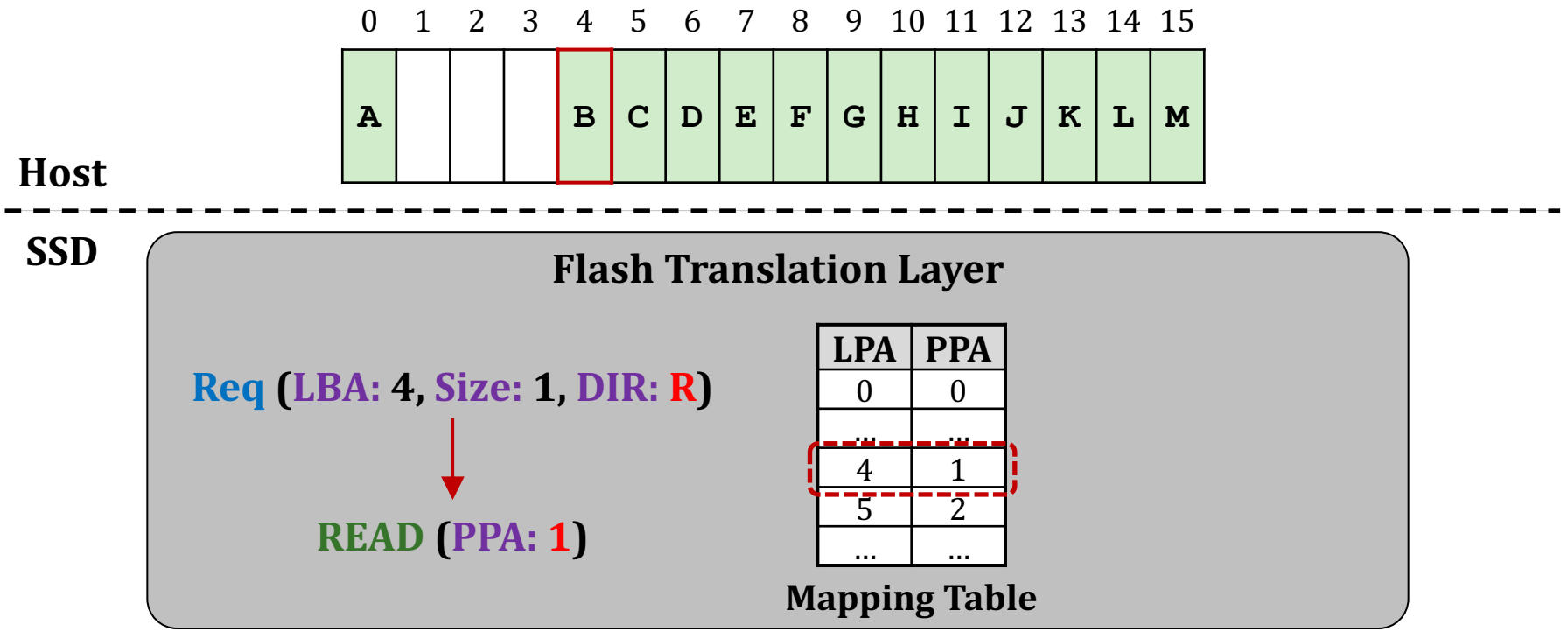
Write Request Handling: Address Mapping



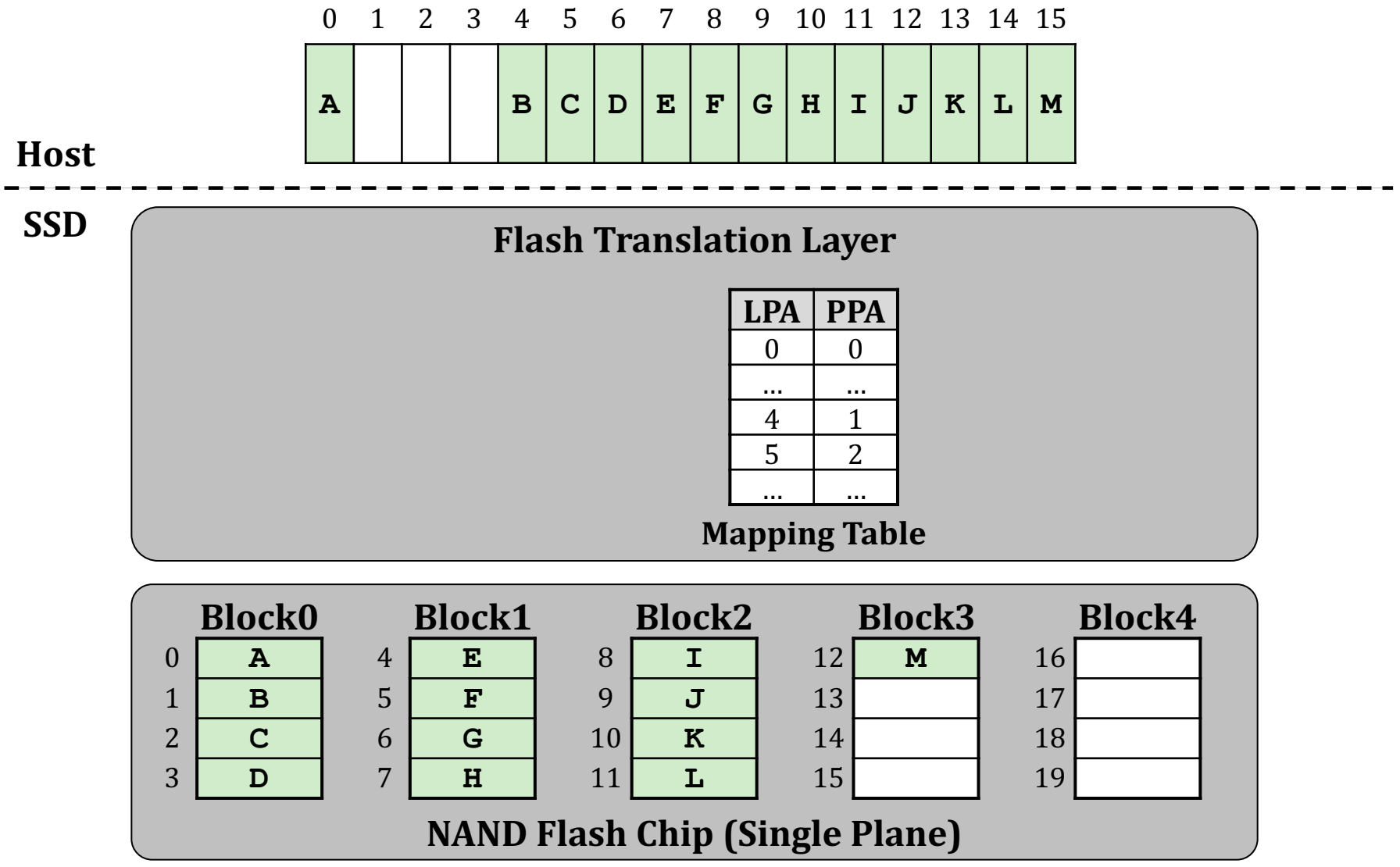
Write Request Handling: Address Mapping



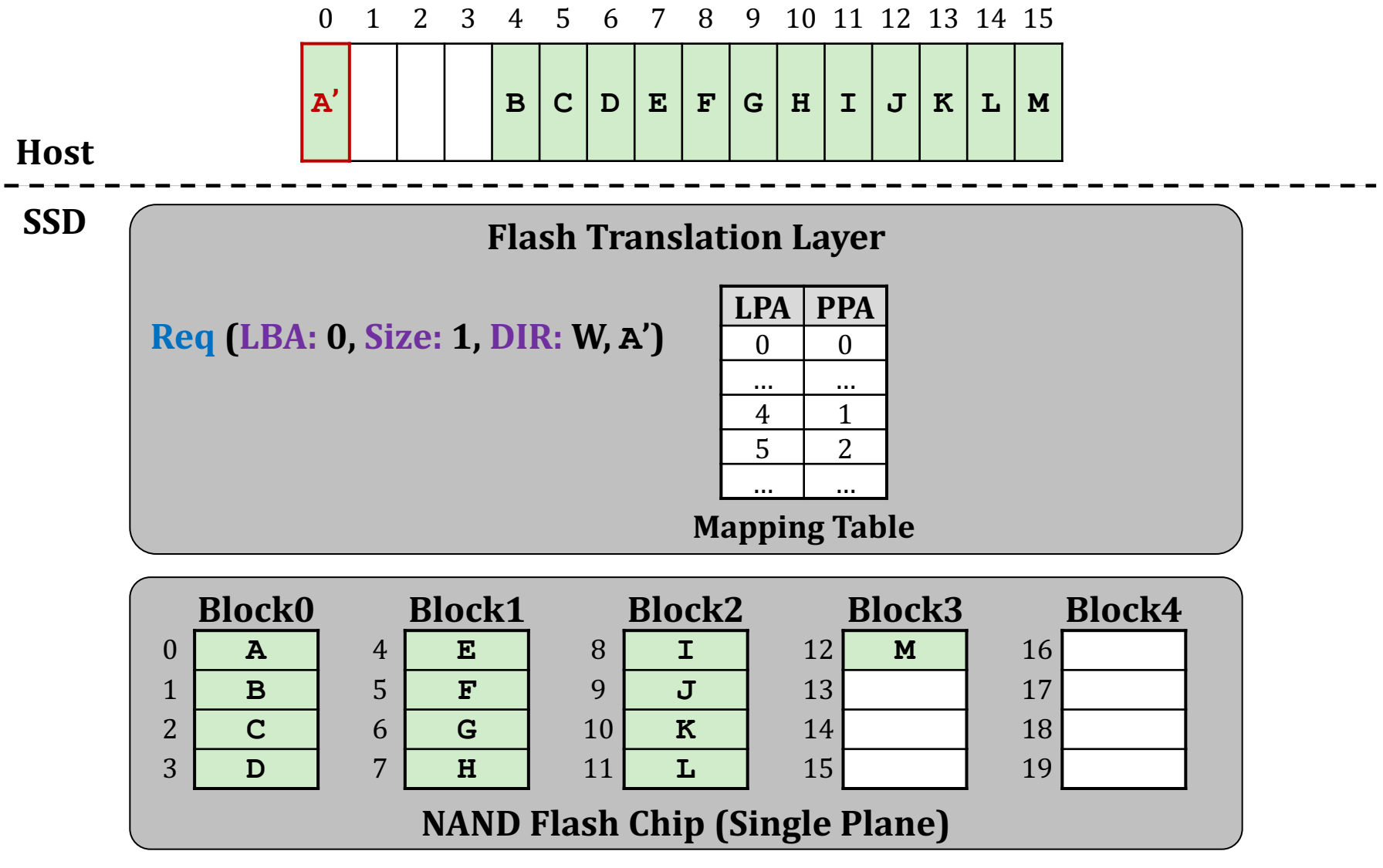
Write Request Handling: Address Mapping



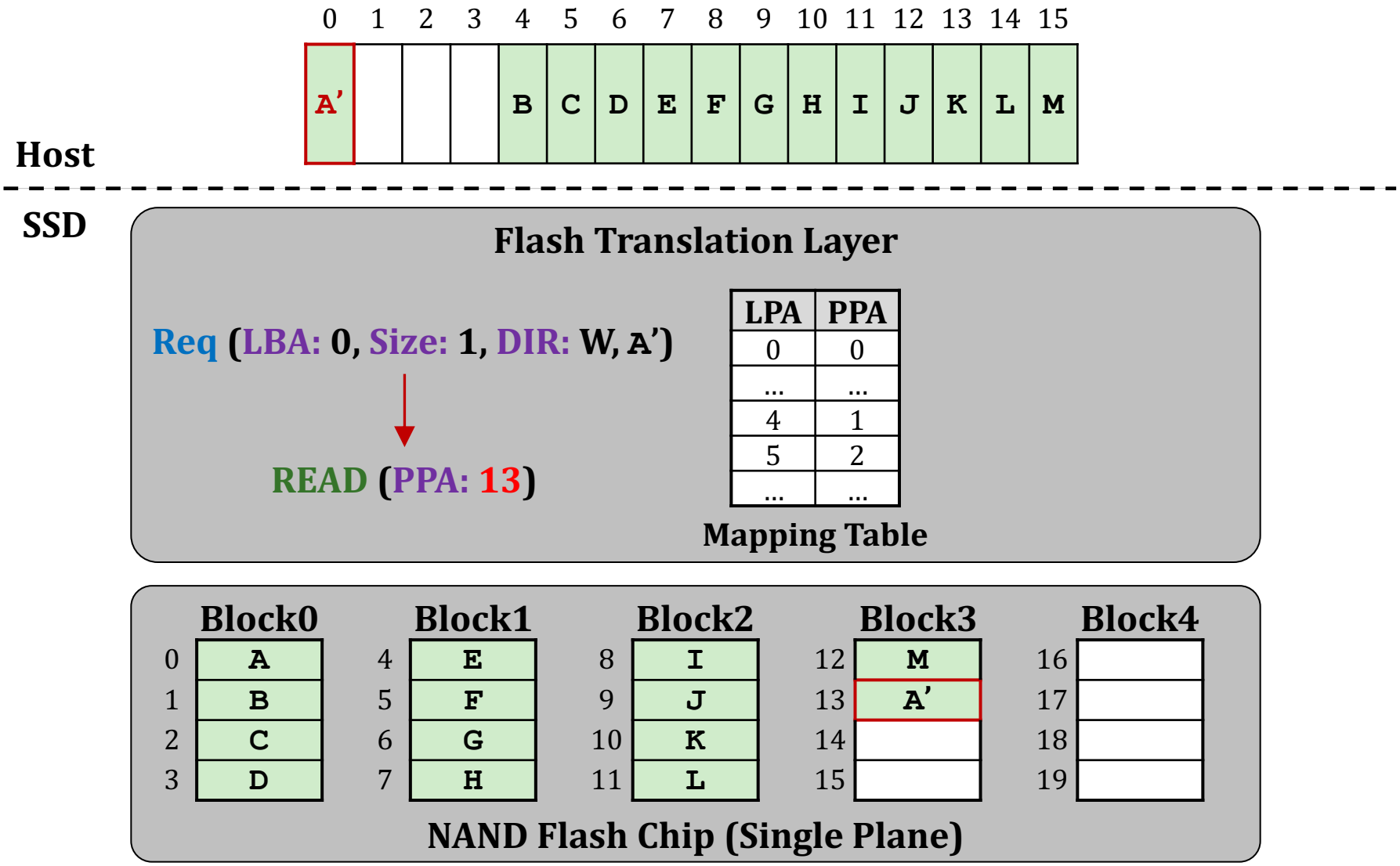
Write Request Handling: Update



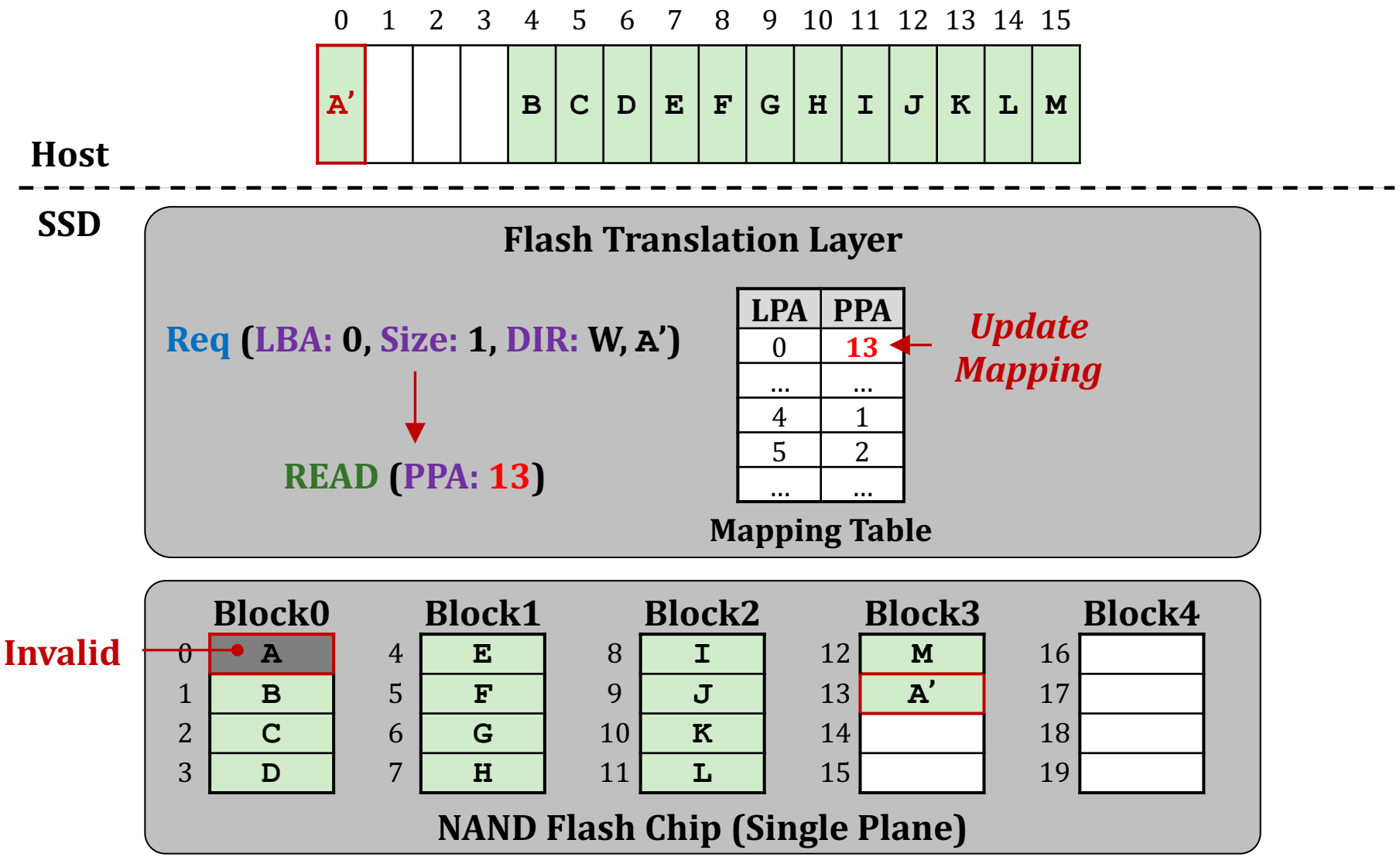
Write Request Handling: Update



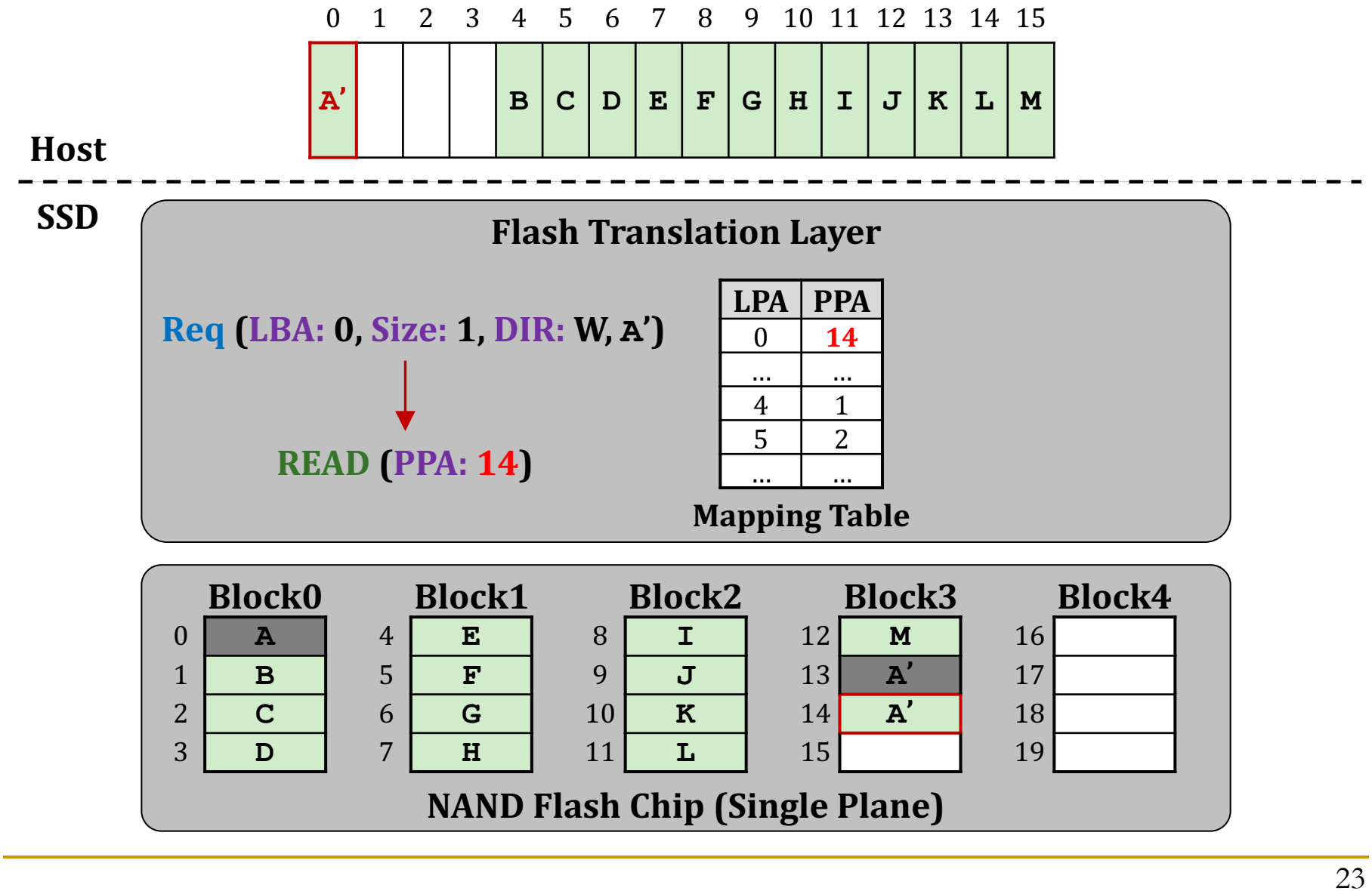
Write Request Handling: Update



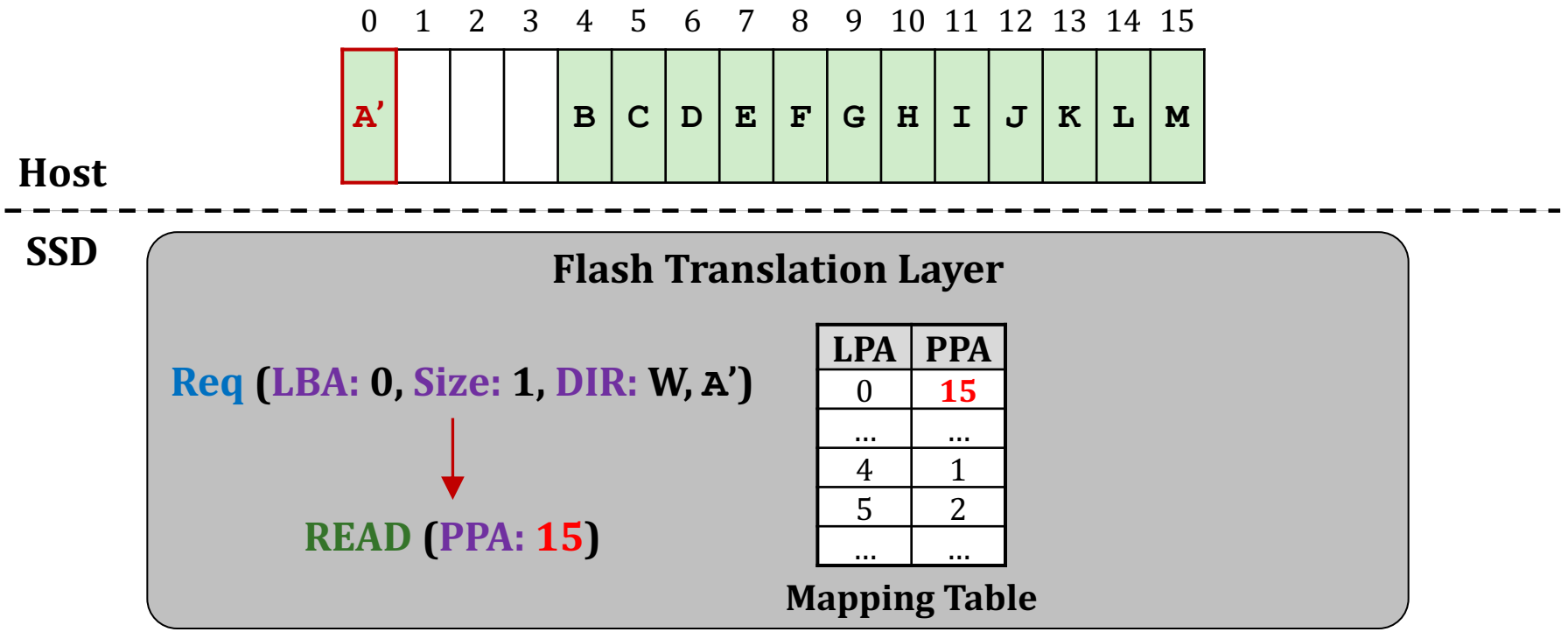
Write Request Handling: Update



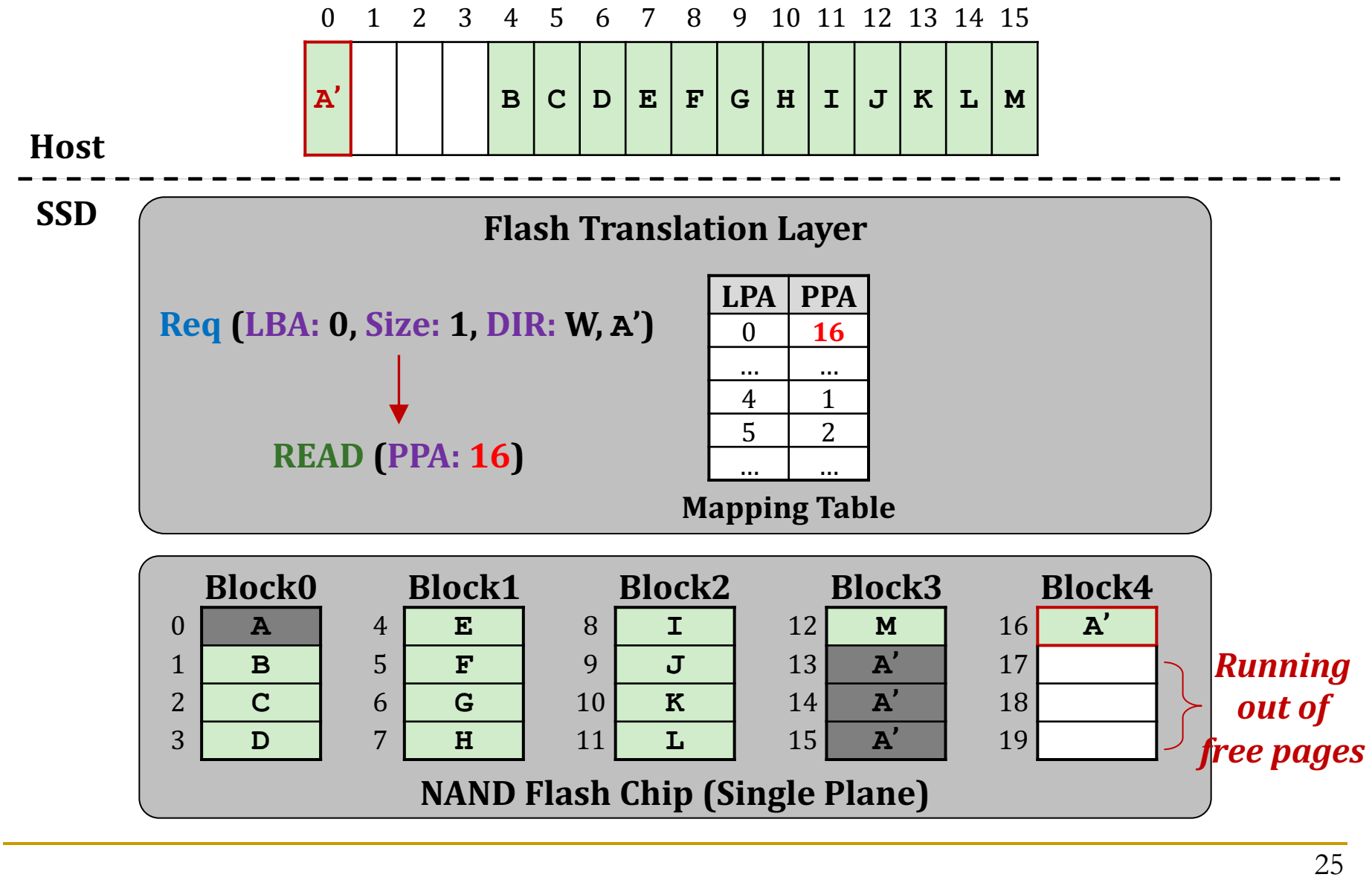
Write Request Handling: Update



Write Request Handling: Update



Write Request Handling: Update

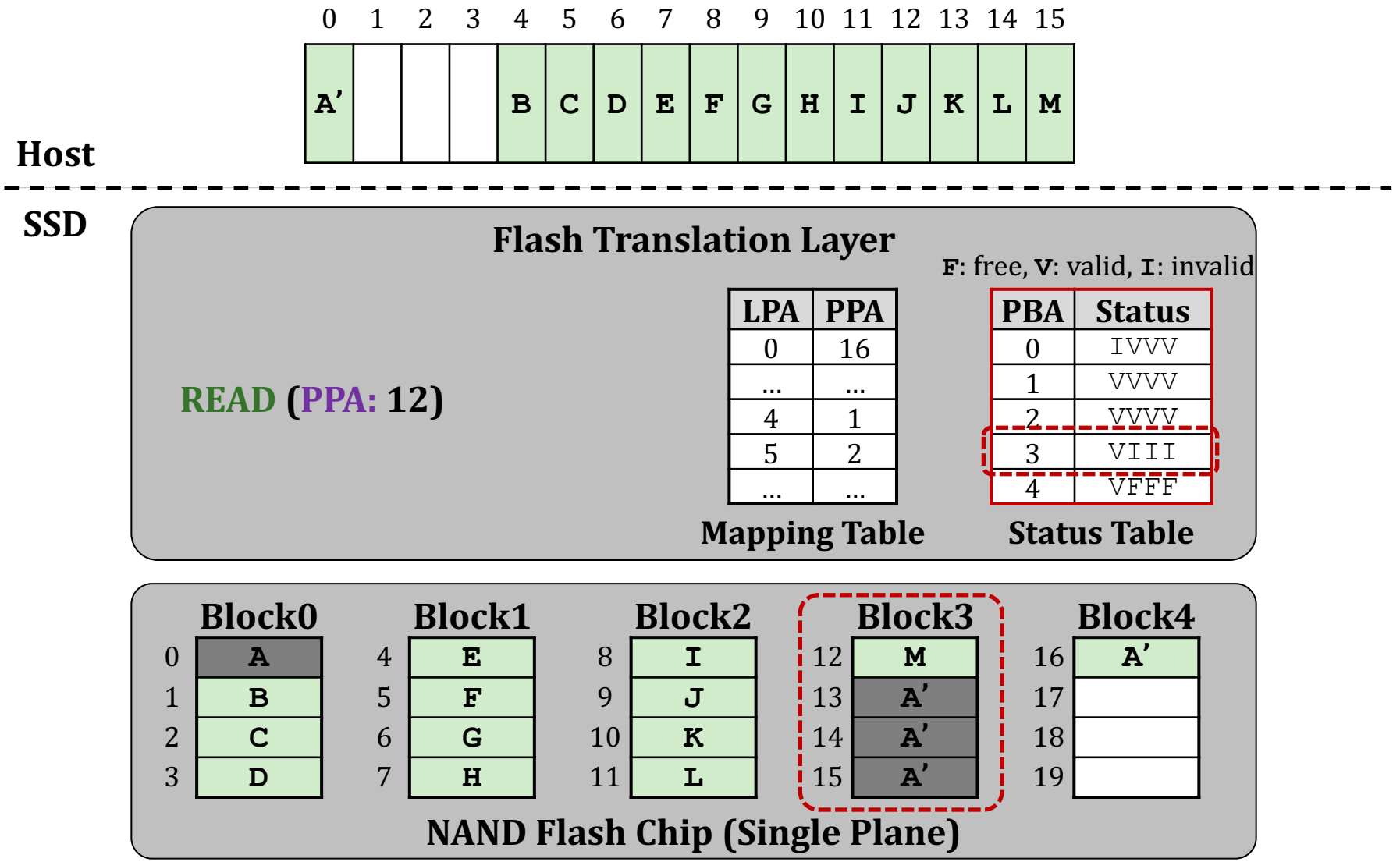


Garbage Collection

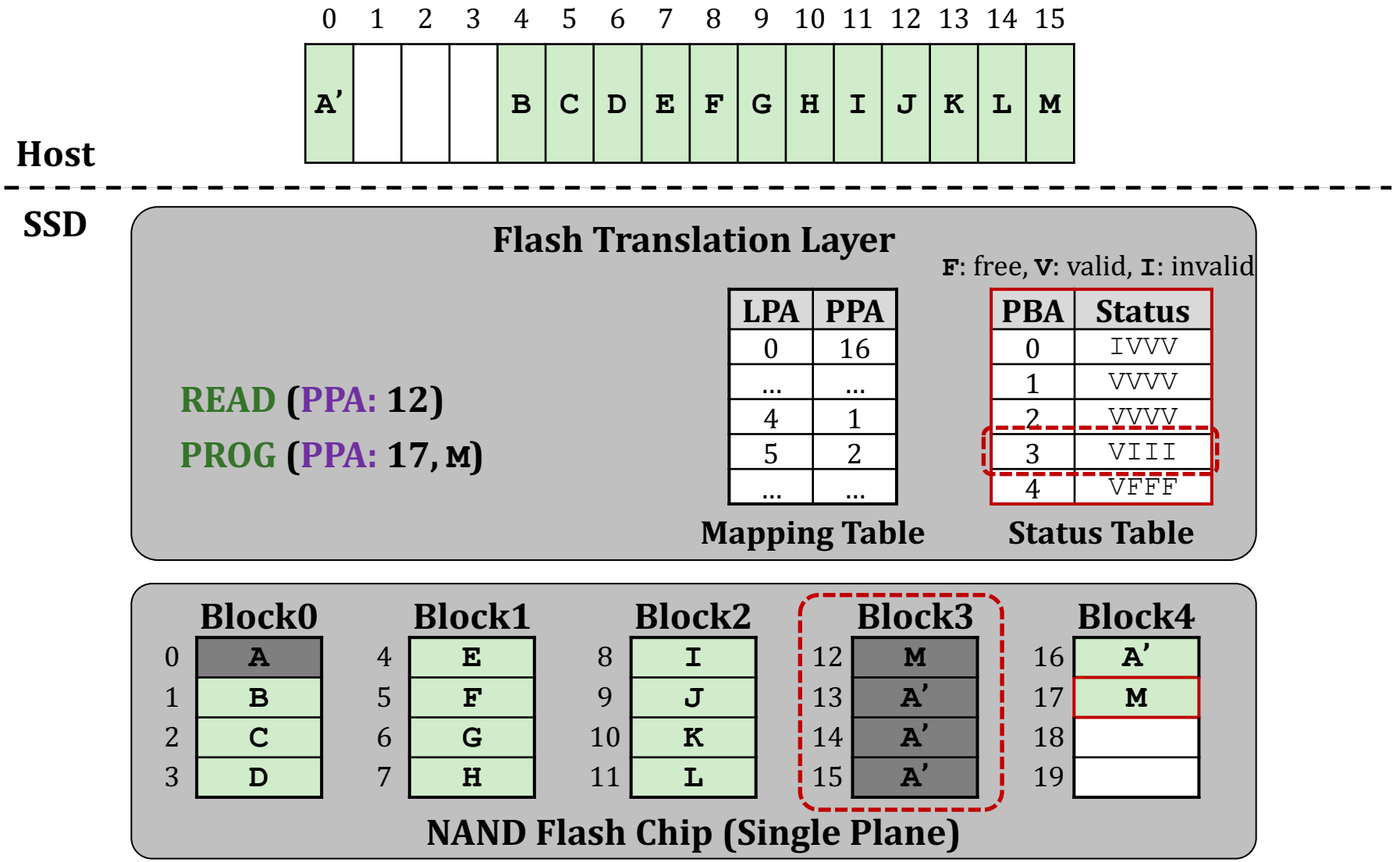
- Reclaims free pages by erasing invalid pages
  - Erase unit: block
  - If a victim block (to erase) has valid pages,
    all the valid pages need to be copied to other free pages
    - Performance overhead: (tREAD + tPROG) × # of valid pages
    - Lifetime overhead: additional writes → P/E-cycle increase

- Greedy victim-selection policy:
  Erases the block with the largest number of invalid pages
  - Needs to maintain # of invalid (or valid) pages for each block

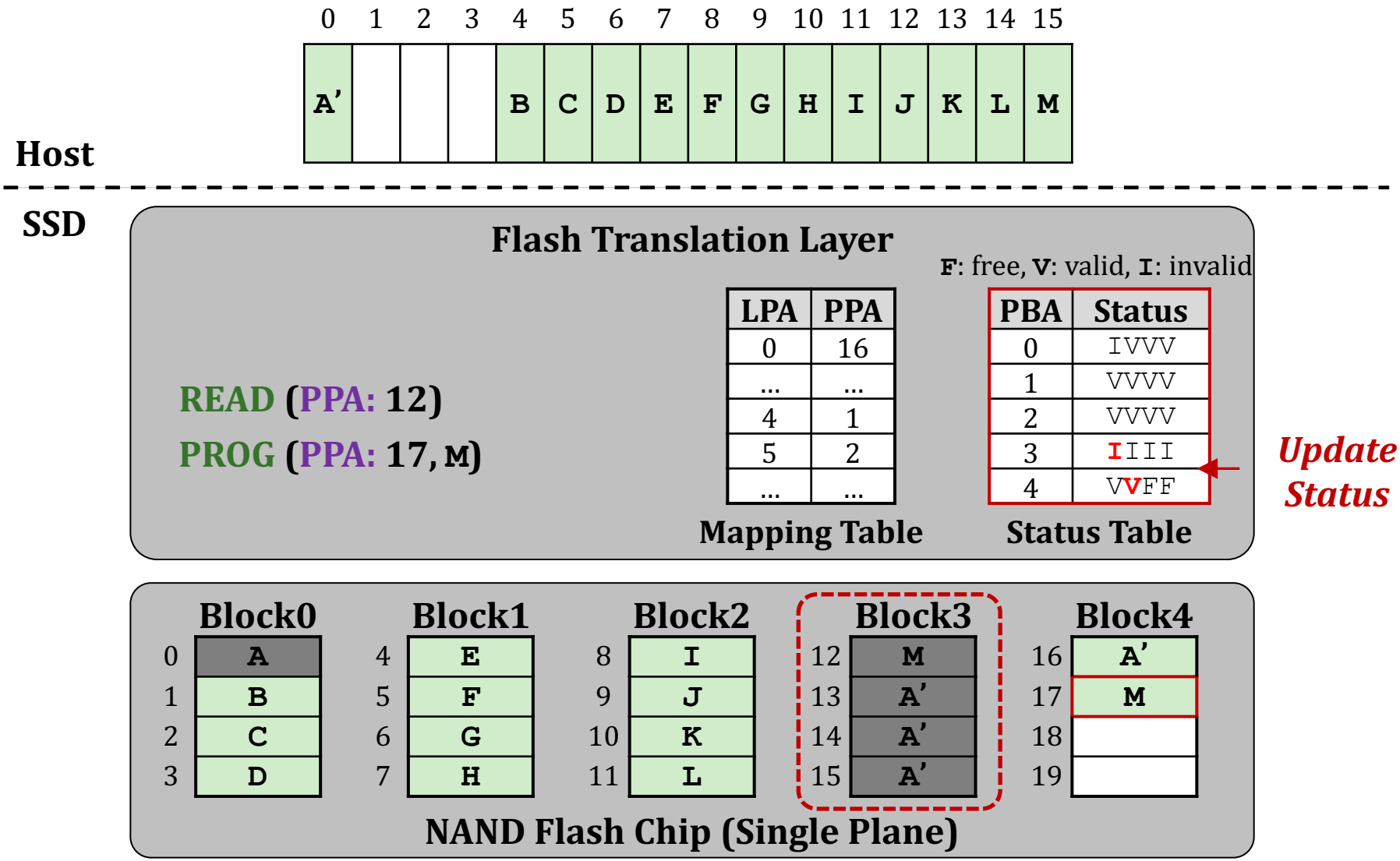
Write Request Handling: Garbage Collection



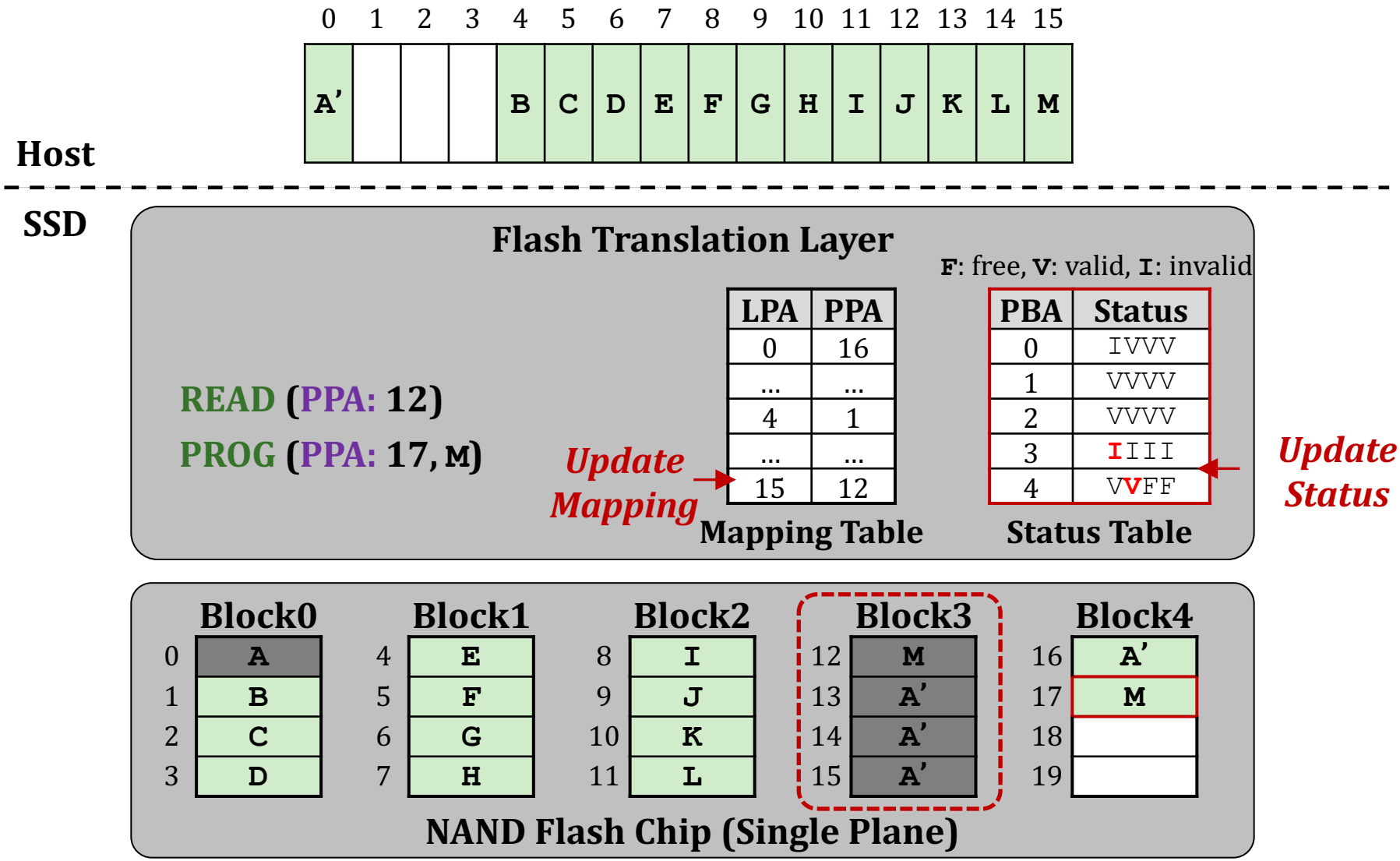
Write Request Handling: Garbage Collection



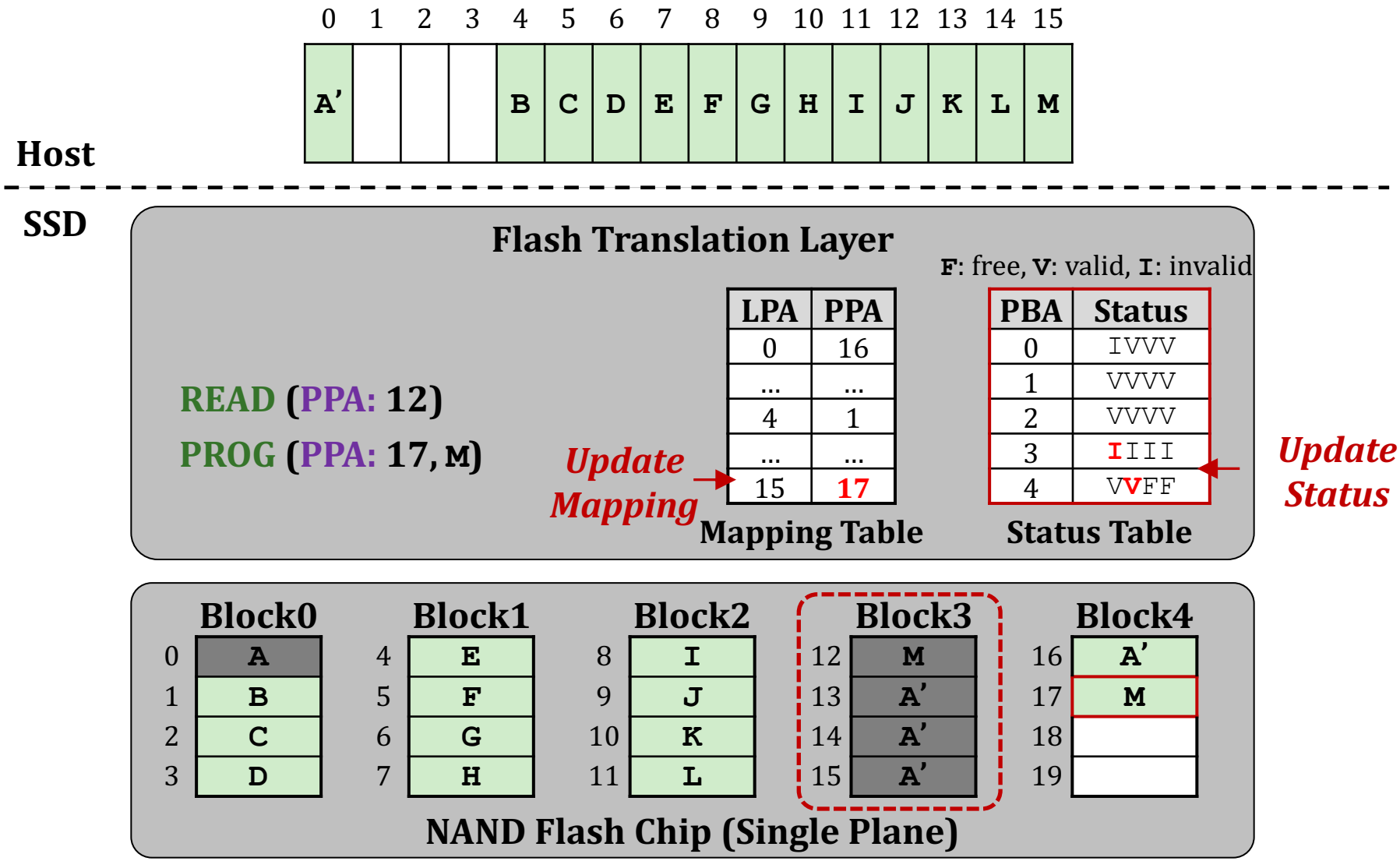
Write Request Handling: Garbage Collection



Write Request Handling: Garbage Collection

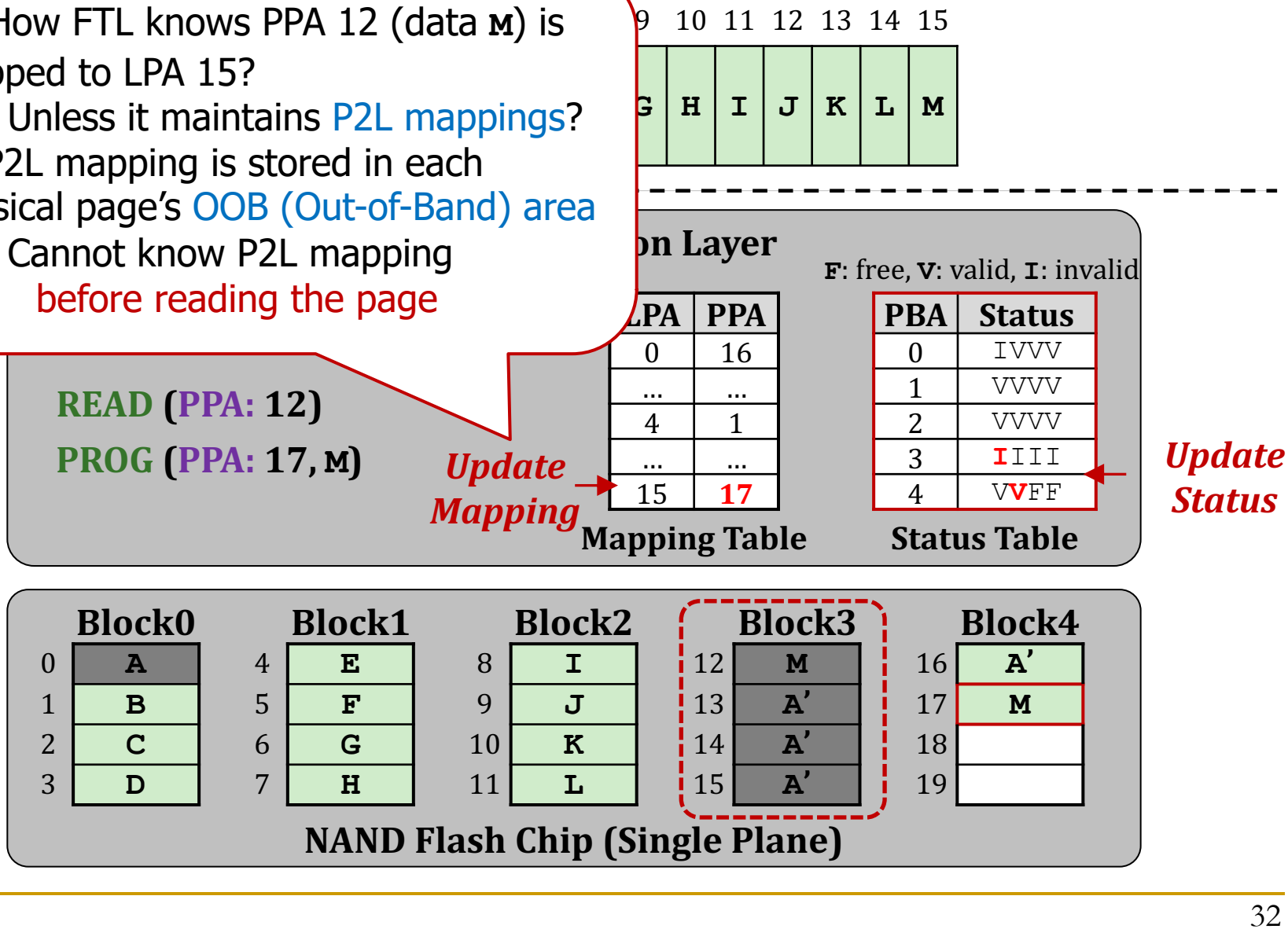


Write Request Handling: Garbage Collection

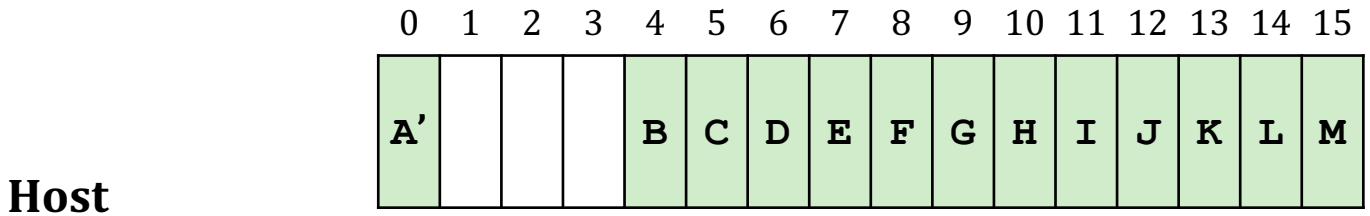


Write Request Handling: Garbage Collection

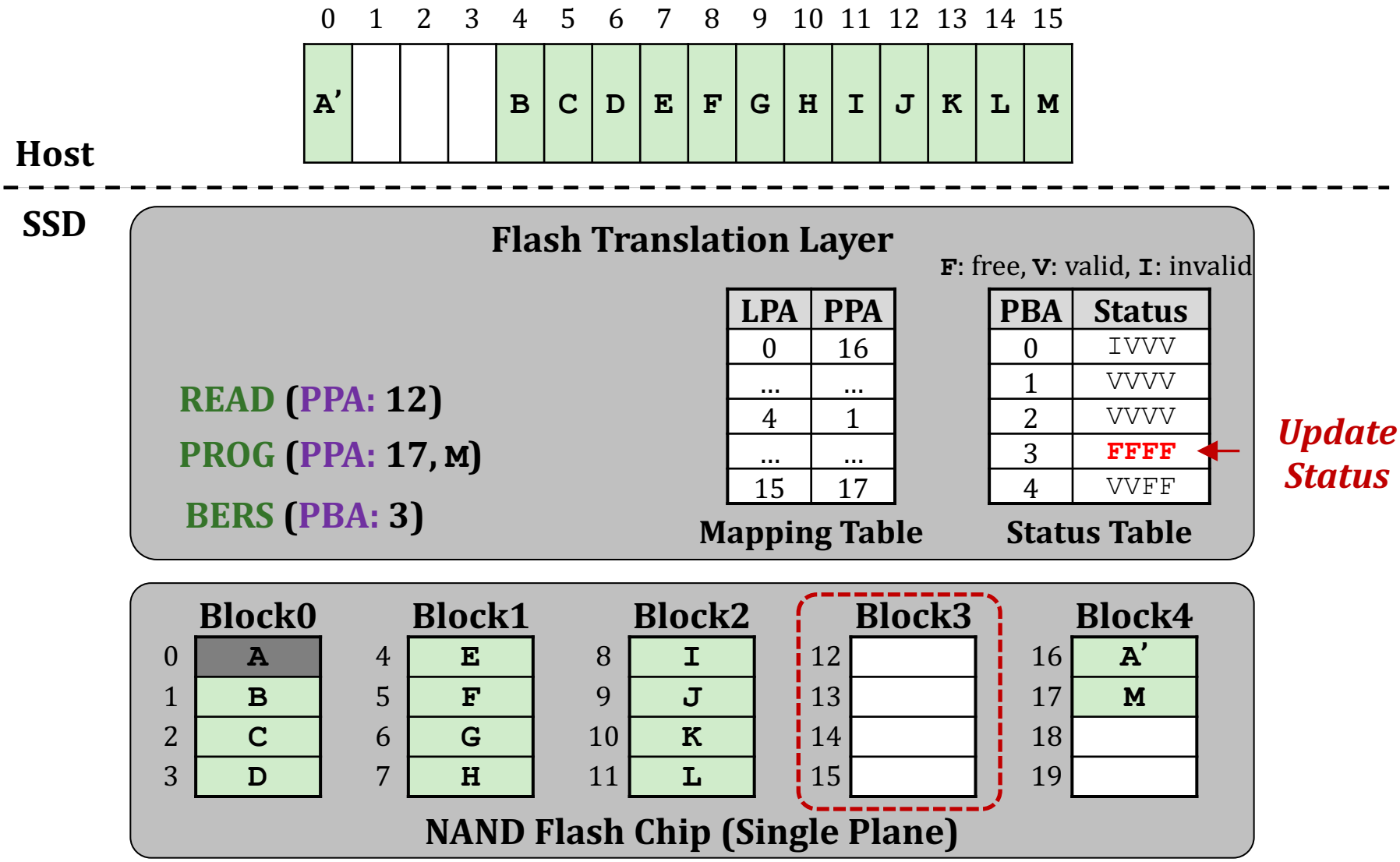
- **Q:** How FTL knows PPA 12 (data **M**) is mapped to LPA 15?
  - Unless it maintains P2L mappings?
- **A:** P2L mapping is stored in each physical page's OOB (Out-of-Band) area
  - Cannot know P2L mapping before reading the page



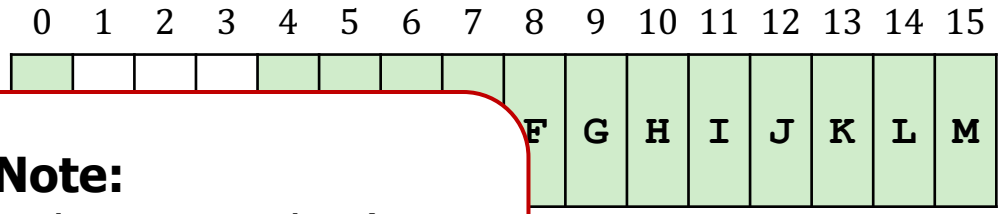
Write Request Handling: Garbage Collection



Write Request Handling: Garbage Collection



Write Request Handling: Garbage Collection



Note:

- Block erasure (and status update) is done just before programming a new page to the block (i.e., lazy erase)
  - Due to the open-block problem

ation Layer

F: free, **V**: valid, **I**: invalid

LPA	PPA
0	16
...	...
4	1
...	...
15	17

Mapping Table

PBA	Status
0	IVVV
1	VVVV
2	VVVV
3	FFFF
4	VVFF

Status Table

Update Status

(PPA: 12)

G (PPA: 17, M)

BERS (PBA: 3)

Block0		Block1		Block2		Block3		Block4	
0	A	4	E	8	I	12		16	A'
1	B	5	F	9	J	13		17	M
2	C	6	G	10	K	14		18	
3	D	7	H	11	L	15		19	

NAND Flash Chip (Single Plane)

Performance Issues

- Garbage collection significantly affects SSD performance
  - High latency: Large block size of modern NAND flash memory
    - Assume 1) a block contains 576 pages,
      2) only 5% of the pages in the victim block are valid
      3) tR = 100 us, tPROG = 700 us, tBERS = 5 ms
      - # of pages to copy = $576 \times 0.05 = 28.8 \rightarrow 28$ pages
      - GC latency > $28 \times (tR + tPROG) + tBERS = 27{,}400$ ms
    - Order(s) of magnitude larger latency than tR and tPROG
    - Copy operations are the major contributor (rather than tBERS)
  - If FTL performs GC in an atomic manner,
    it delays user requests for a significantly long time
    - Long tail latency (performance fluctuation)
    - Noisy neighbor: a read-dominant workload's performance would be significantly affected when running with a write-intensive workload (+ performance fairness problem)

Performance Issues: Mitigation

- TRIM (or discard) command
  - Informs FTL of deletion/deallocation of a logical block
  - Allows FTL to skip copy of obsolete (i.e., invalid) data

- Background GC: Exploits SSD idle time
  - Challenge: how to accurately predict SSD idle time
  - Premature GC: copied pages could have been invalidated by the host system

- Progressive GC: Divide GC process into subtasks
  - e.g., copying 28 pages → (copying 1 page + servicing user request) × 28
  - Effective at decreasing tail latency

P&S Modern SSDs

Address Translation & Garbage Collection

Dr. Jisung Park
Prof. Onur Mutlu
ETH Zürich
Spring 2021
19 May 2021