3. SSD Controller

Special Topics in Computer Systems:

Modern Storage Systems
(IC820-01)

Instructor:

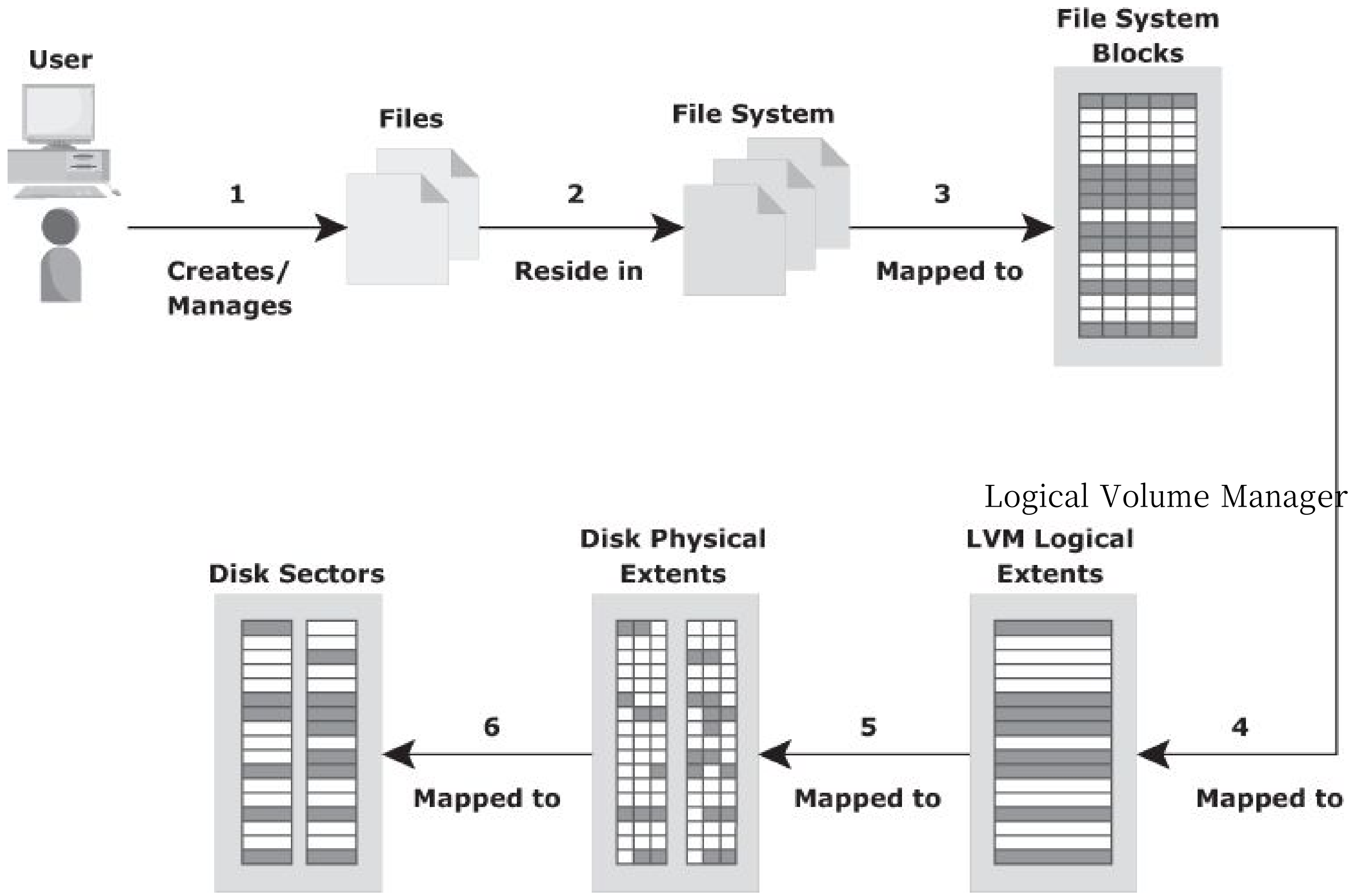
Prof. Sungjin Lee (sungjin.lee@dgist.ac.kr)

Outline

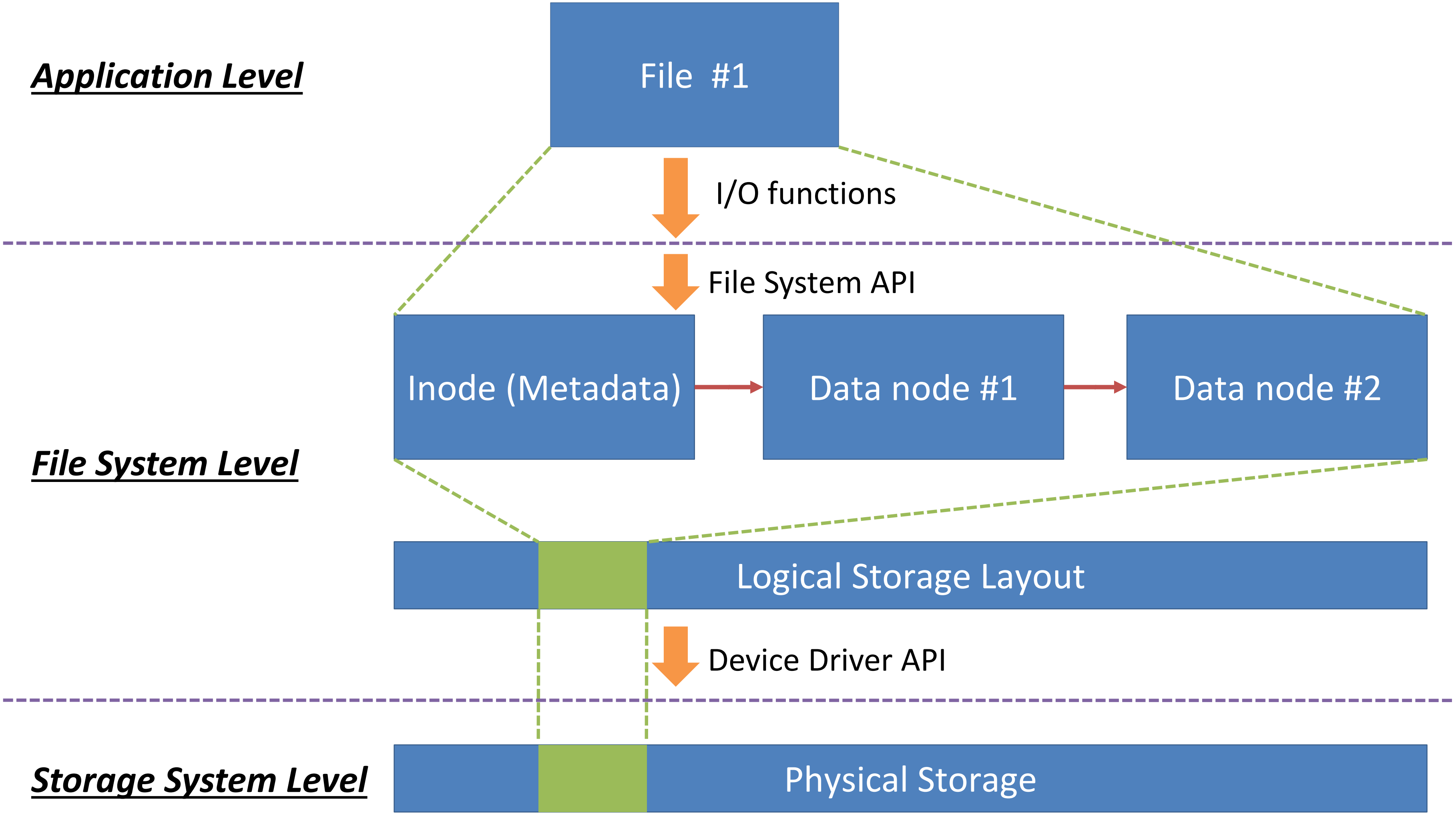
- **Storage Abstraction & Protocols**
- Flash Packages
- SSD Controller
- Flash Array & I/O Parallelism

File to Storage

- In most cases, users manage their data in the form of files



File to Storage (Cont.)

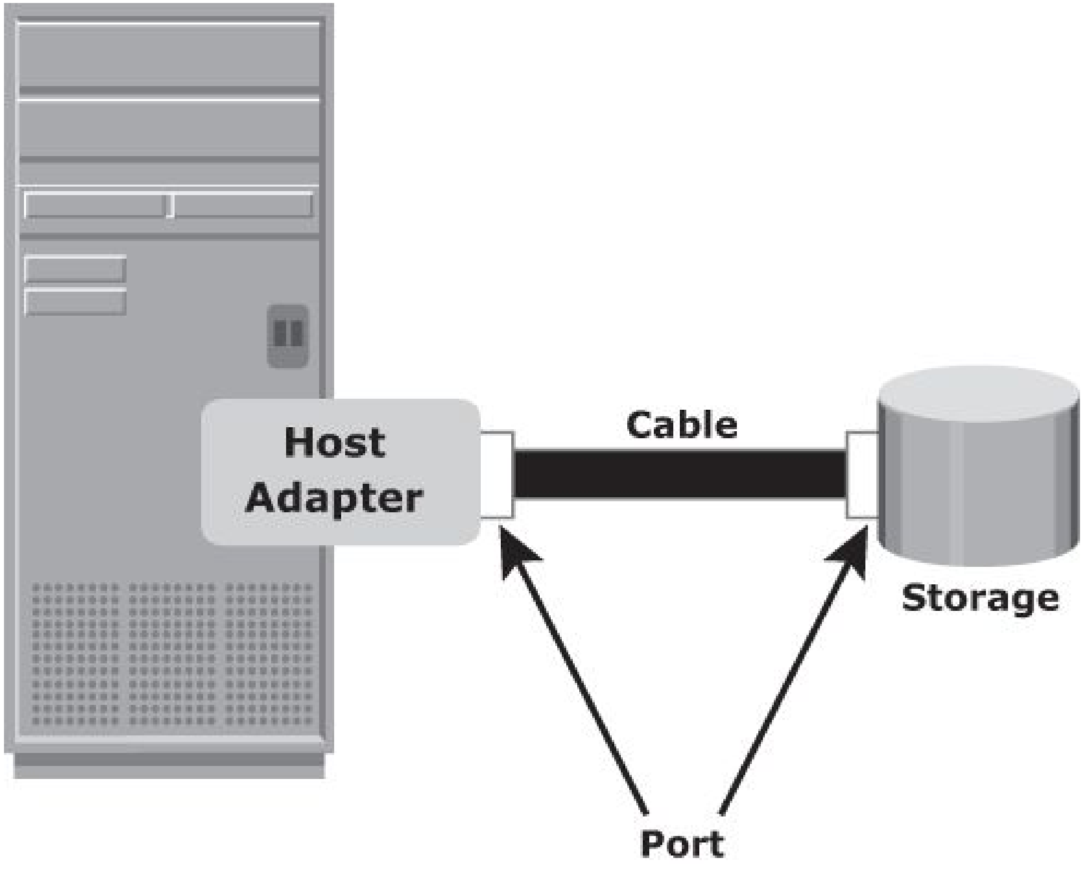


Application Level

File System Level

Storage System Level

Storage Interface & Protocols

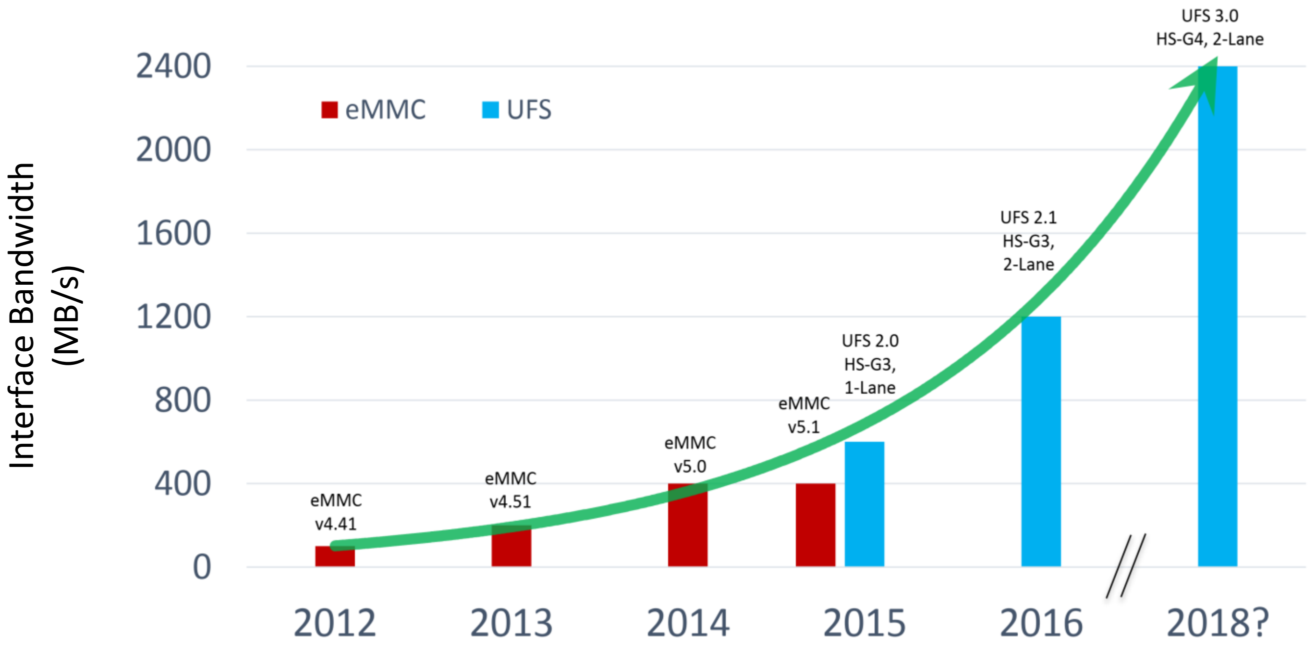


More detailed introduction to storage interface & protocols will be given as a separate lecture

Storage Interface & Protocols (Cont.)

- **There are various types of storage interfaces depending on target applications' requirements**

- **Case #1: mobile storage interface**
  - High throughput and low latency (everybody wants high-speed storage)
  - ***Low active power*** (e.g., Max 1.62W) and ***limited form factor***



Storage Interface & Protocols (Cont.)

- **There are various types of storage interfaces depending on target applications' requirements**

- **Case #2: desktop/server storage interface**
  - ***High performance*** is the primary goal
  - Servers require ***many*** storage devices and high ***reliability***

Interface	Mnemonic Meaning	Transfer Speed	Characteristics
SATA	Serial ATA	Up to 2GB/s	Low cost
SAS	Serial Attached SCSI	Up to 3GB/s HDD 150개도 가능	Supports multiple ports Error detection/correction
FC	Fibre Channel	Up to 16GB/s	Predominately SCSI commands and features
NVMe	Nonvolatile memory express over Pcie	4 GB/s per lane	Up to 32 lanes High command queue support
NVDIMM	Nonvolatile memory on memory channel	Up to 1 GB/s over 64-bit bus	Very slow latency No interrupt Deterministic

* T. Coughlin et.al., "Digital Storage and Memory Technology (Part1)," IEEE Technical Trend Paper, Nov. 2017

Outline

- Storage Abstraction & Protocols
- **Flash Packages**
- SSD Controller
- Flash Array & I/O Parallelism

Review:
NAND Flash Chip

- **A group of blocks compose a NAND flash die**
- **Two or four dies are then packed as a single NAND chips**

<NAND flash chip specification>

Page Read to SRAM	50 μs
Page Program (Write) from SRAM	500 μs
Block Erasure	4 *ms*
Serial Access to SRAM	100 μs (per 4 KB)
Page Size	16 KB
Block Size	4 MB
Die Size	8 Gb
Dies per Package	1, 2, or 4

- **Key properties**
  - Read is 10x faster than write: 50 μs vs 500 μs
  - Erasure is slowest: 4 *ms*
  - Throughput is not so high: 36.4 MB/s for reads (Note: recent SSDs offer 5 GB/s)

Review:

How to Improve Performance?

Read throughput (36.4 MB/s) = 16 KB / (50 + 400) μs

- **Strategy 1: Reduce the latency of three I/O primitives**
  - The latency of three I/O primitives get longer as NAND cells scale down

	SLC (2D)	MLC (2D)	TLC (2D)	MLC (3D)
Page Program	25 μs	50 μs	100 μs	50 μs
Page Read	250 μs	500 μs	3 ms	500 μs

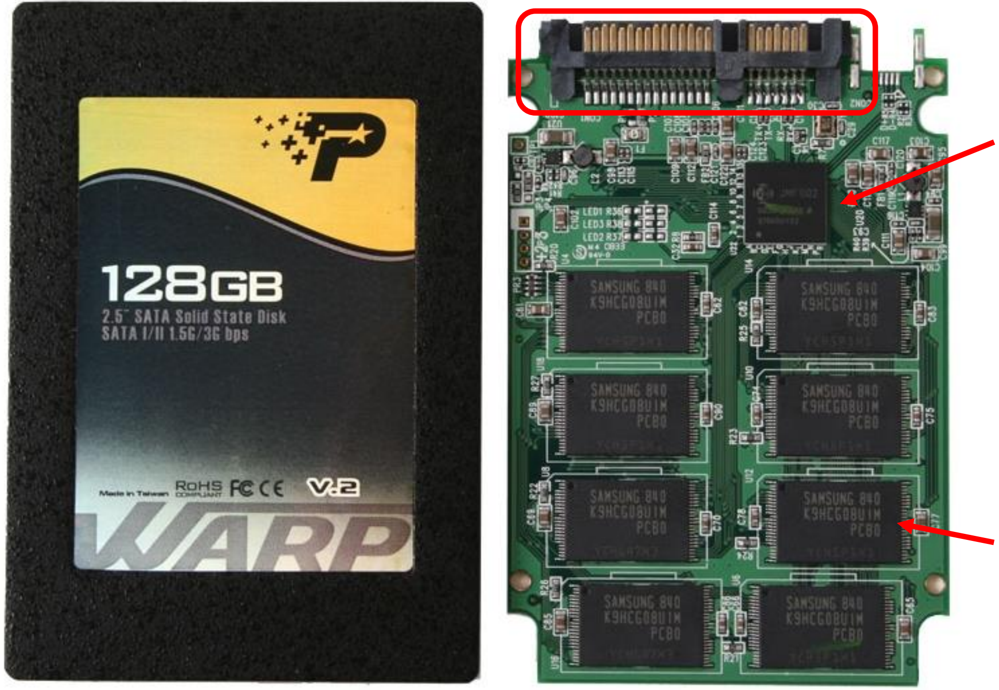
    - 3D NAND improves I/O latency with larger cells

- **Strategy 2: Improve the bandwidth of I/O interface**

	Interface	Throughput (MB/s)
Before 2006	No Standard (SDR)	40 MB/s
2006	NV-SDR	50 MB/s
2008	NV-DDR	200 MB/s
2011	NV-DDR2	533 MB/s
2014	NV-DDR3	800 MB/s

Review:
How to Improve Performance? (Cont.)

- **Strategy 3: Aggregate many NAND chips and access them in parallel**
  - Example: 36.4 MB/s x 64-128 NAND dies = **2.3-4.7 GB/s**
  - This is what an SSD controller does!
    - To achieve optimal performance, sophisticated algorithms are needed!



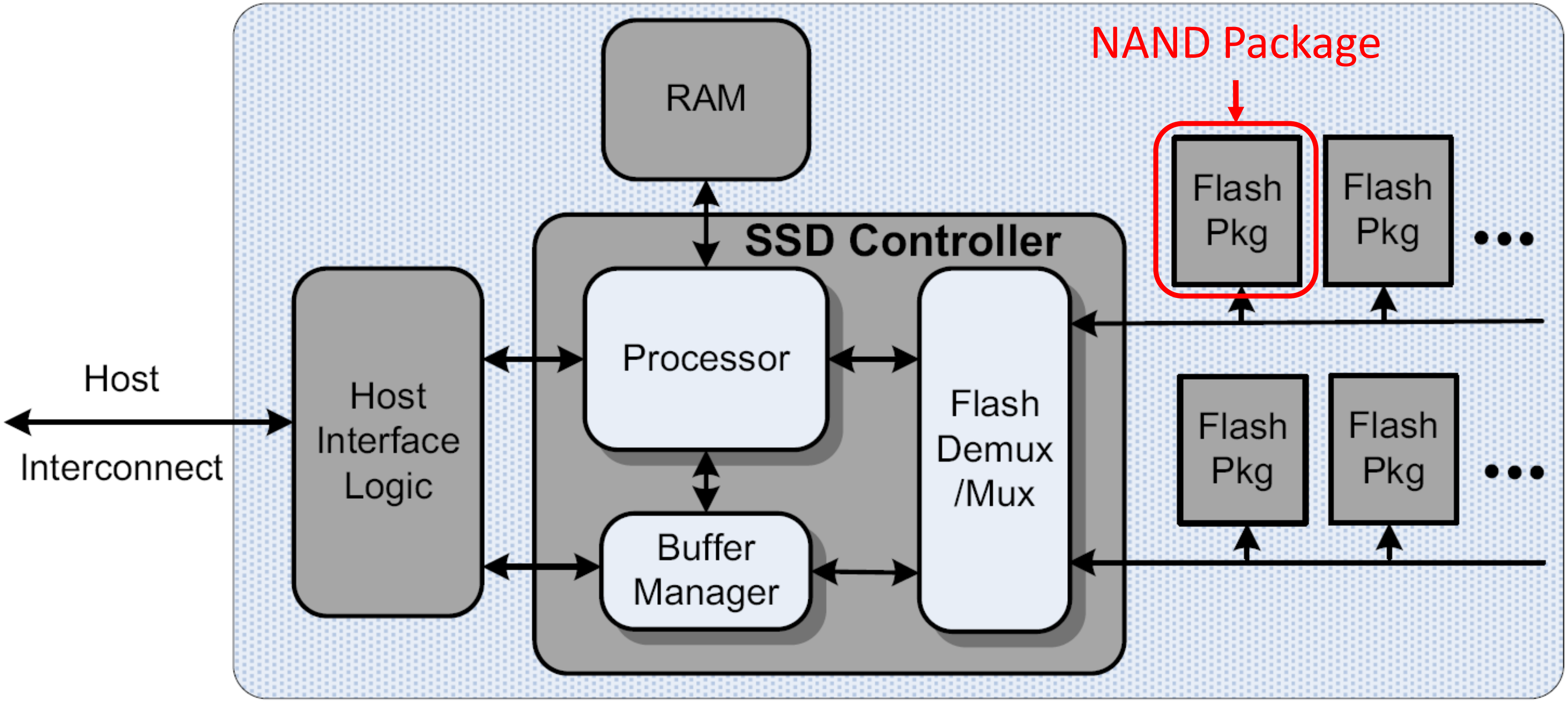
SATA Interface

SSD Controller

NAND chip

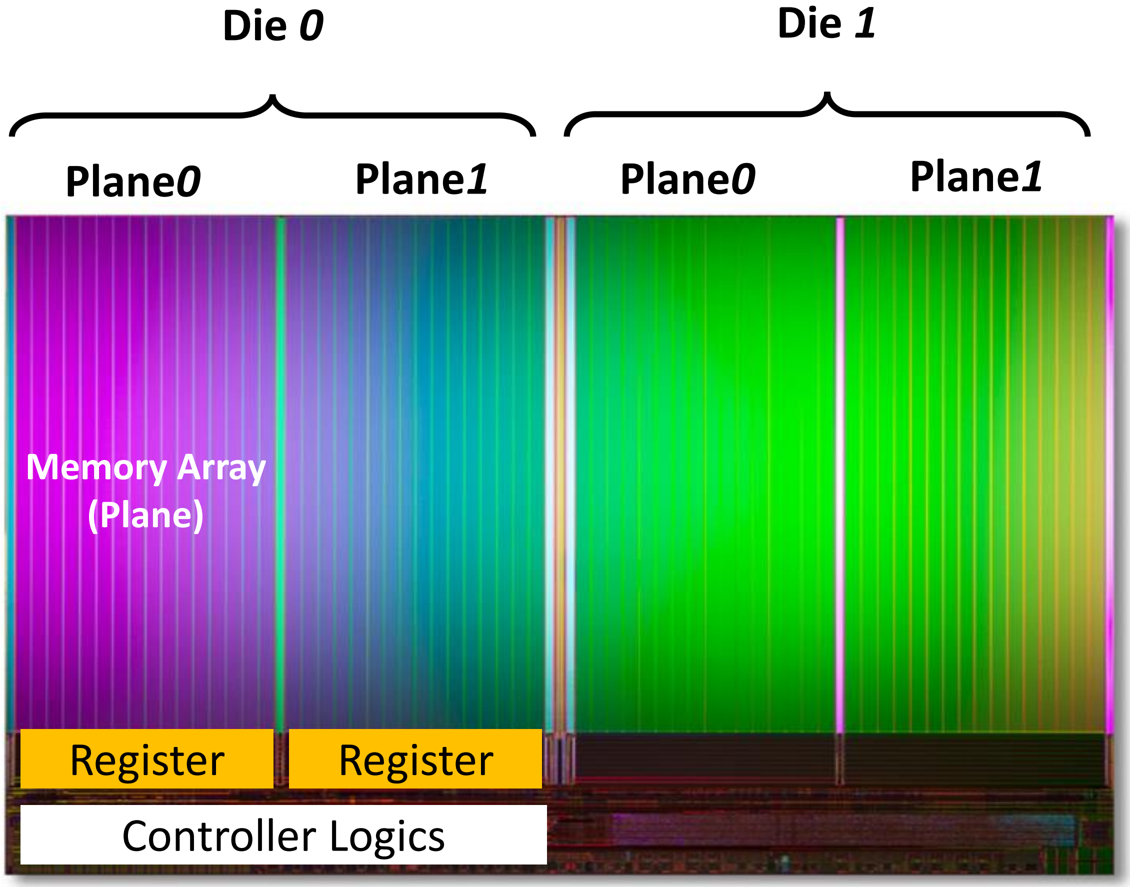
SSD Overview

- **Composed of three main modules**
  - (1) A host interface, (2) an SSD Controller, and (3) an array of NAND packages (or chips)



NAND Package

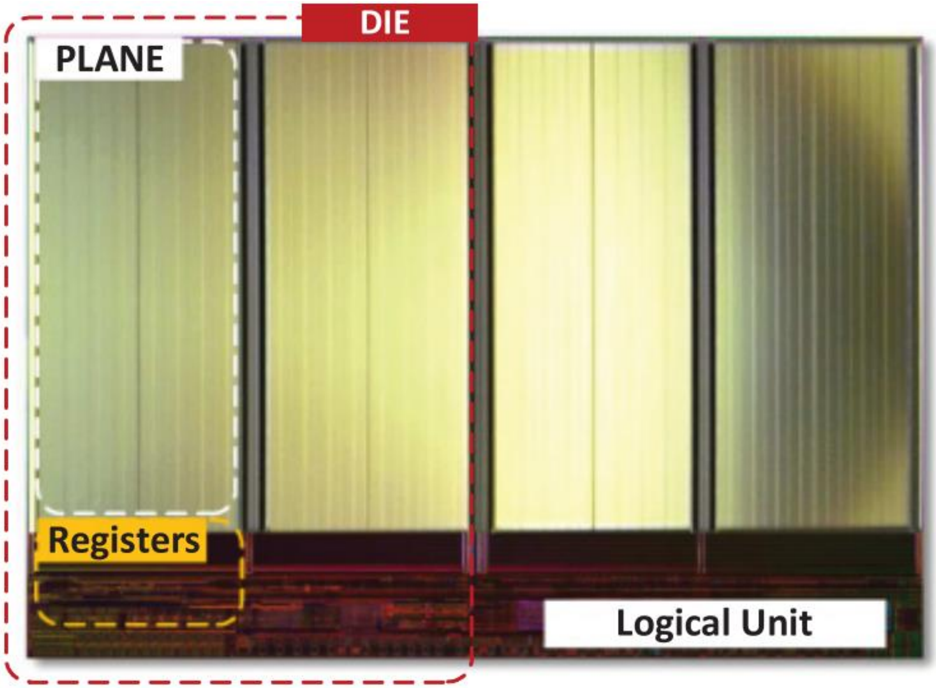
- Employing cache and data registers
- Multiple planes
- Multiple dies



NAND Package (Cont.)

- **Flash package employs multiple dies, each consisting of one or more planes**
  - Planes *share* multiple peripherals, which enable all the target pages connected to their wordlines/bitlines
- **Each plane employs a set of registers to cache/buffer the data brought by flash interface**

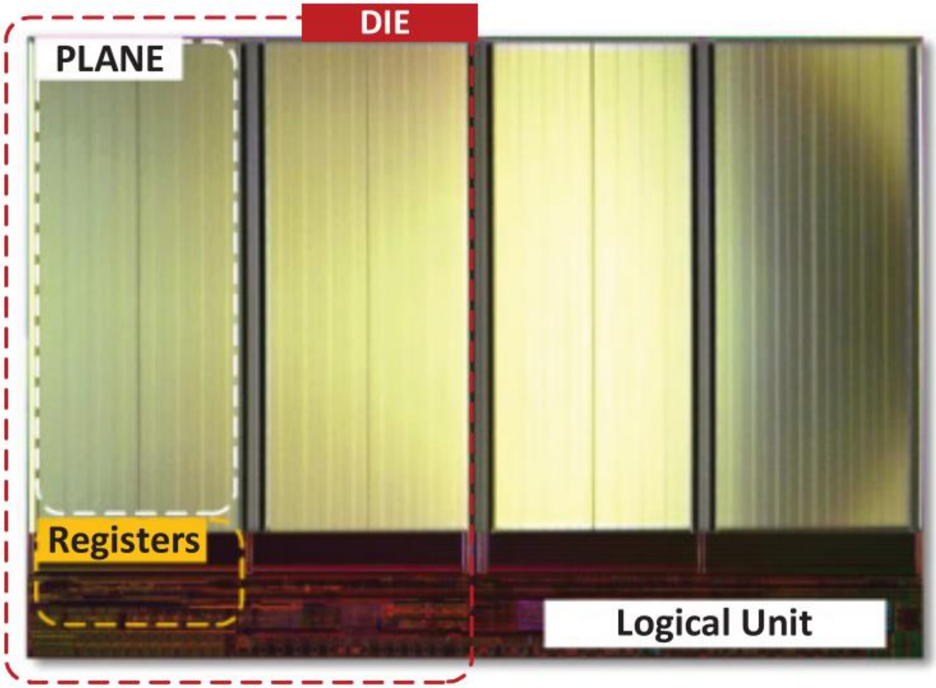
die는 각자가 control logic을 가지고 있는데
Plane은 공유한다.
공간 때문에.
read/write/erase 등을 handle하는 로직을 넣기 위한 공간이 부족해서 die 내의 plane끼리는 공유하는 것.



Physical Plane and Die

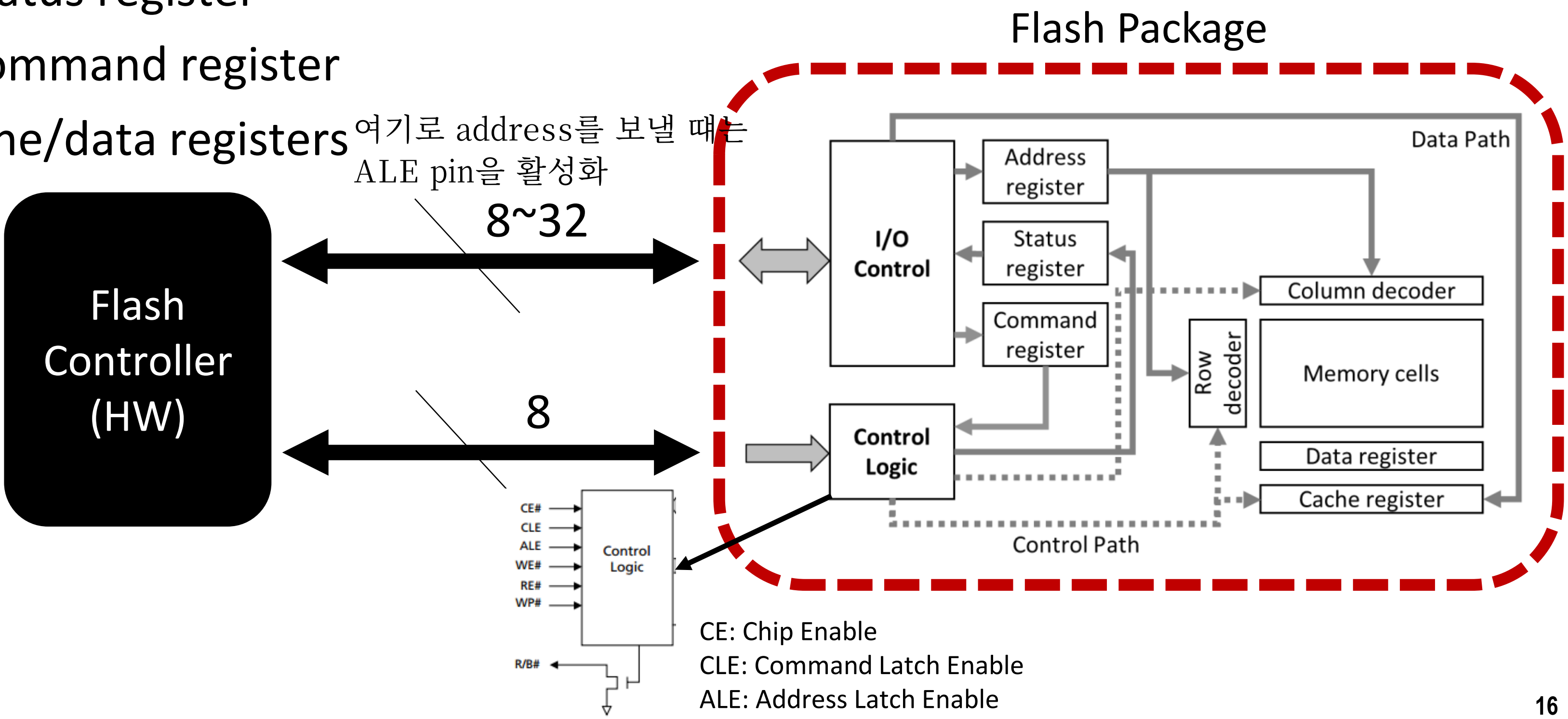
- **Each plane defines a different set of block addresses, and the pages/blocks on different planes operate in parallel**
  - The operation type for the requests across different plane should be same
  - The page and plane addresses should be identical
- **All individual dies can be simultaneously activated, but share data-path (described later)**

모든 die 각각은 동시에 작동하지만
외부와 연결되는 data - path(bus)는 공유한다.



Flash Microarchitecture

- **All data, commands and addresses are multiplexed onto same I/O pins and received by I/O control circuit**
- **Each component of flash transactions is latched by**
  - An address register
  - A status register
  - A command register
  - Cache/data registers

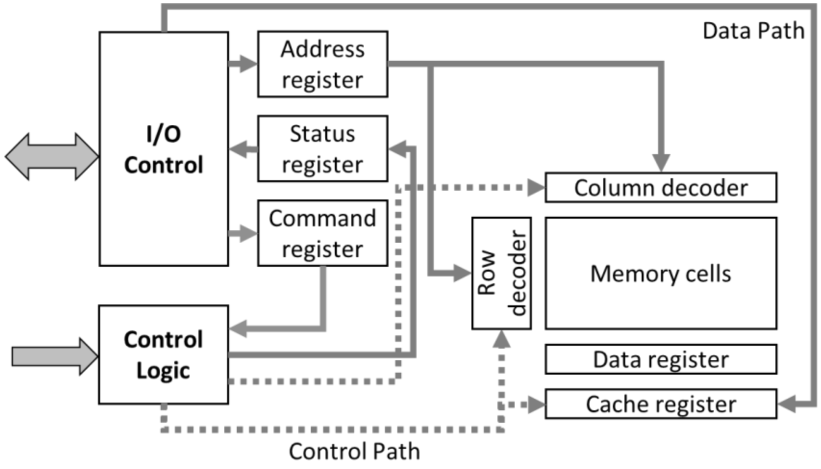


여기로 address를 보낼 때는 ALE pin을 활성화

CE: Chip Enable
CLE: Command Latch Enable
ALE: Address Latch Enable

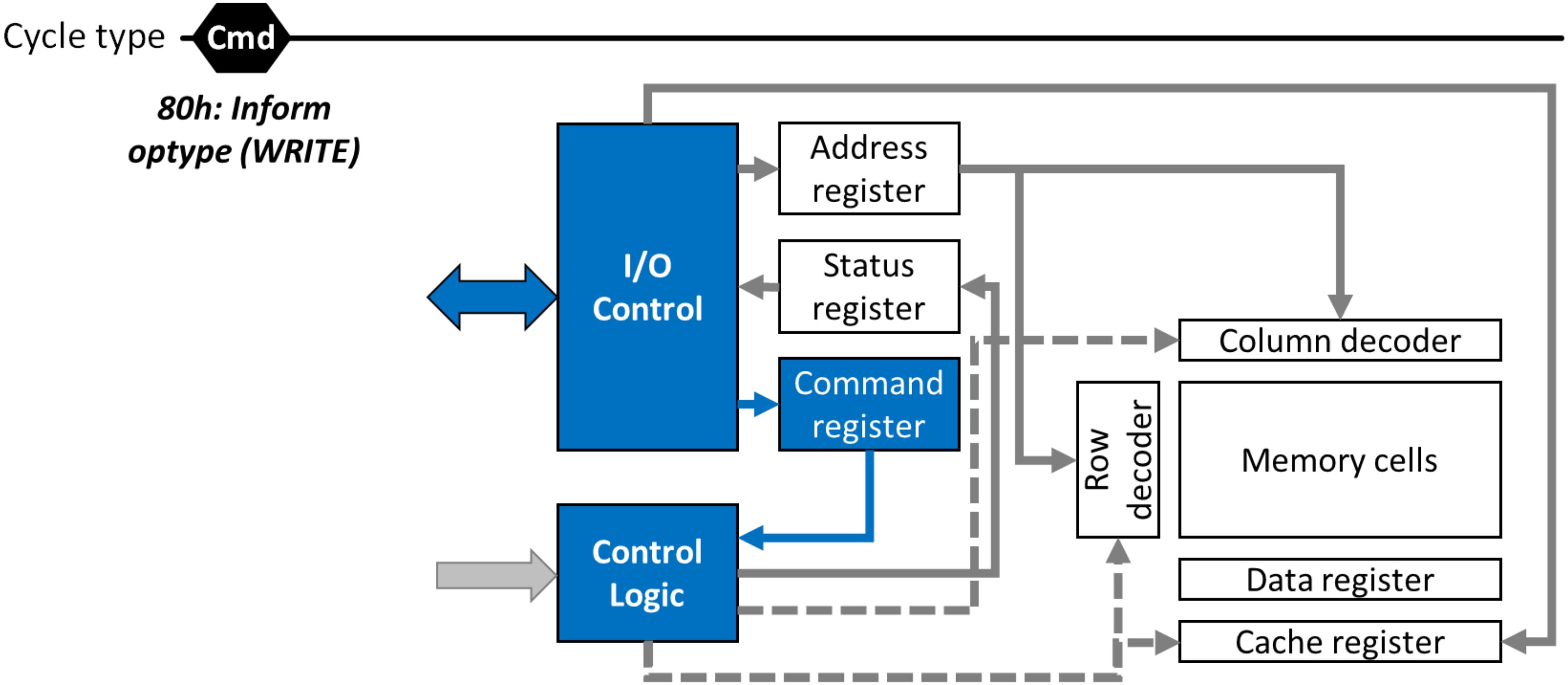
Flash Interface and Protocol

- **To issue a read/write request through the multiplexed I/O pins, a flash controller should obey the protocol that flash interface defines for all memory transactions**
  - Basic operations (e.g., reads and writes)
  - Cache mode operations
  - Multi-plane mode operations
  - Copyback, etc.



Page Write

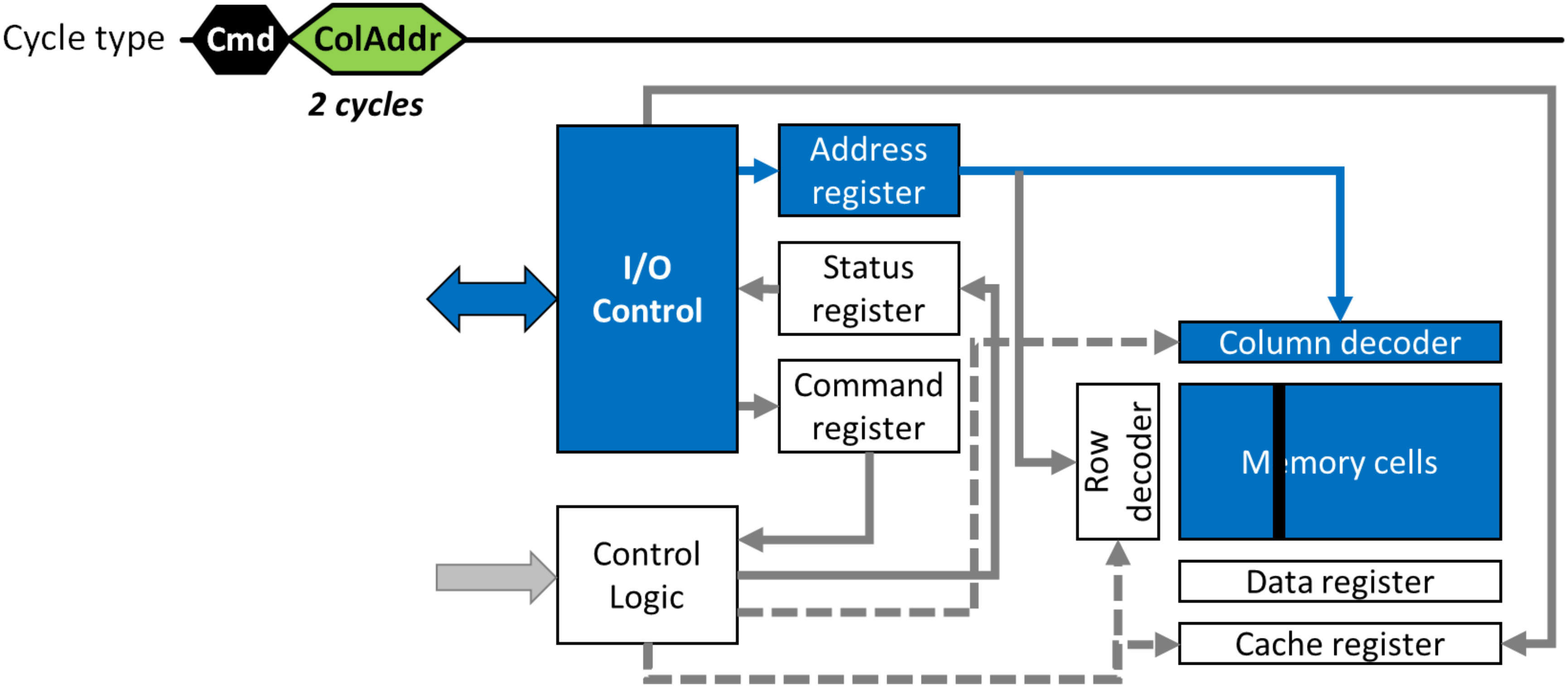
- The first command is needed to indicate that this is a page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



80h: Inform optype (WRITE)

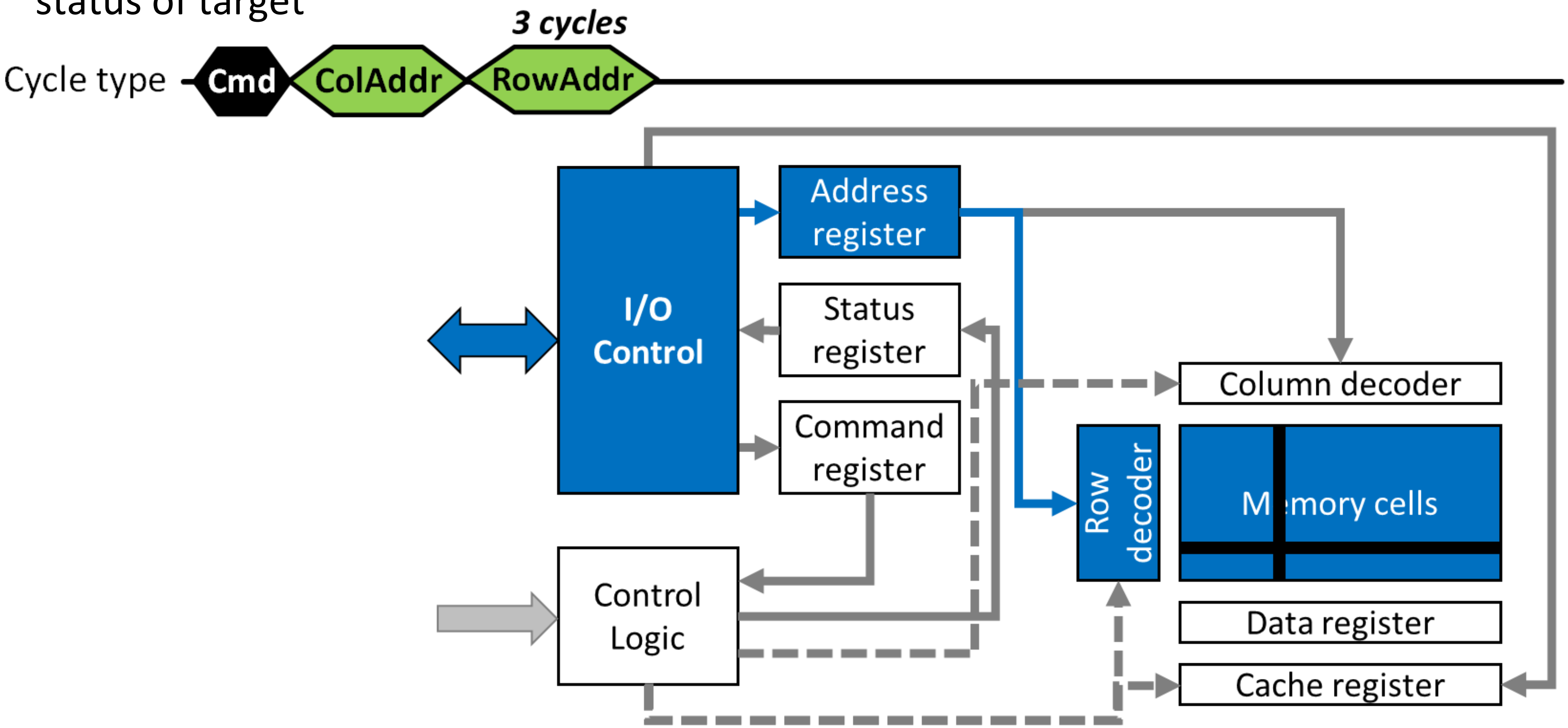
Page Write

- The first command is needed to indicate that this is a page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



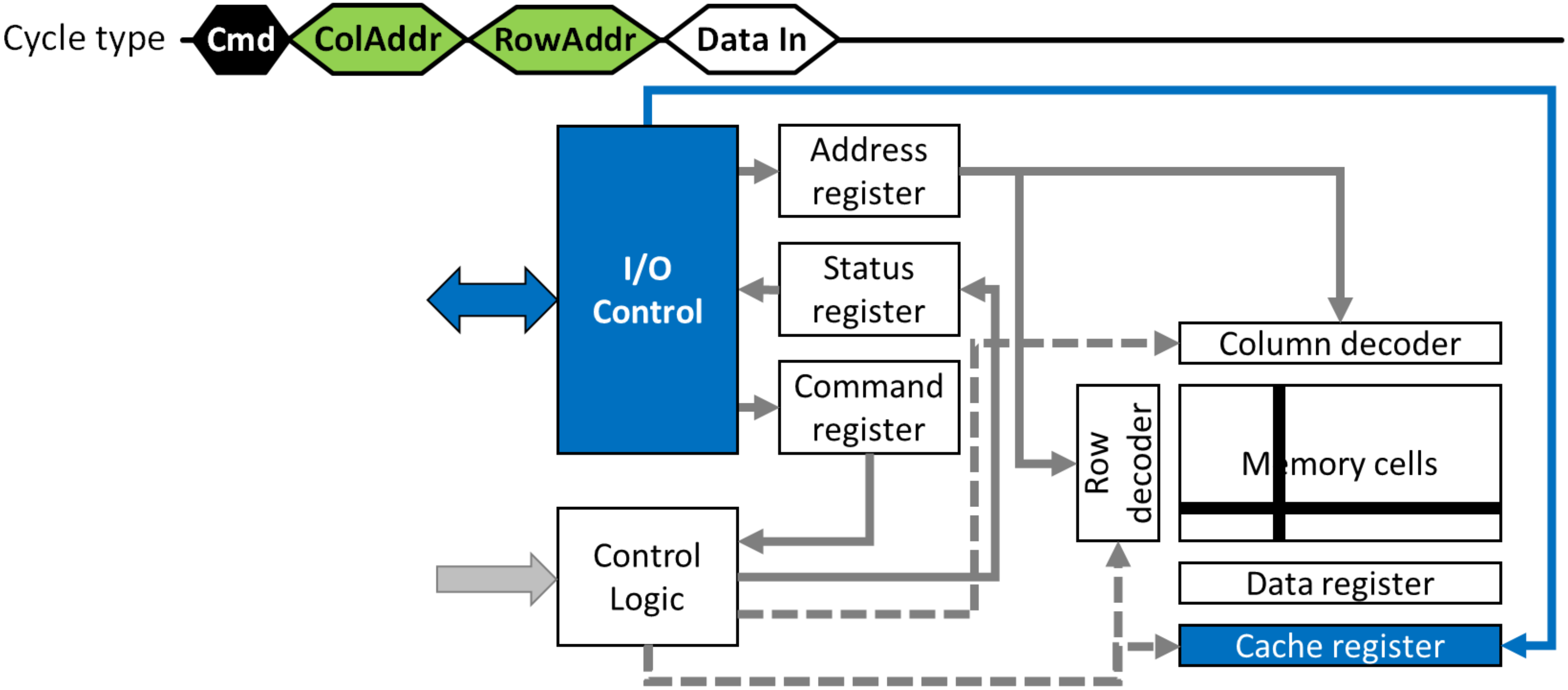
Page Write

- The first command is needed to indicate that this is a page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



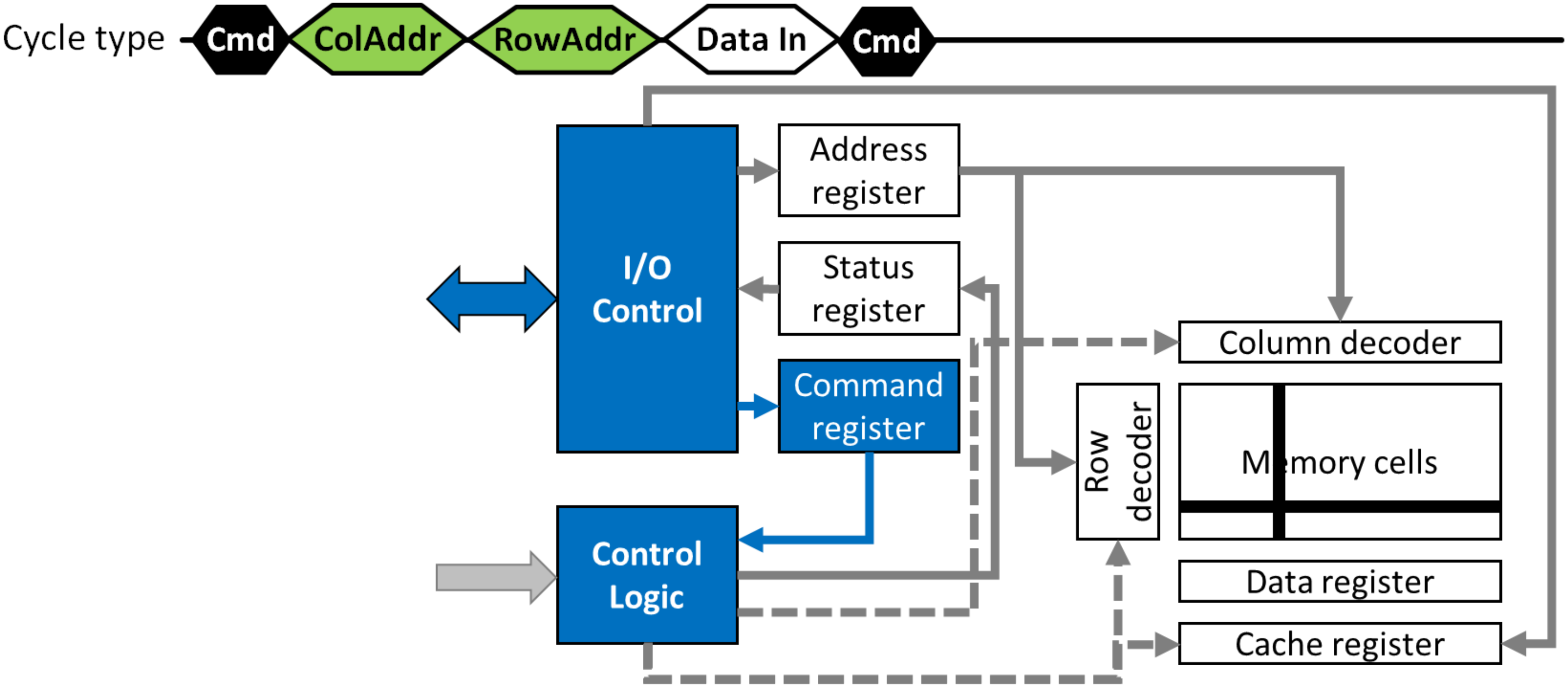
Page Write

- The first command is needed to indicate that this is a page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



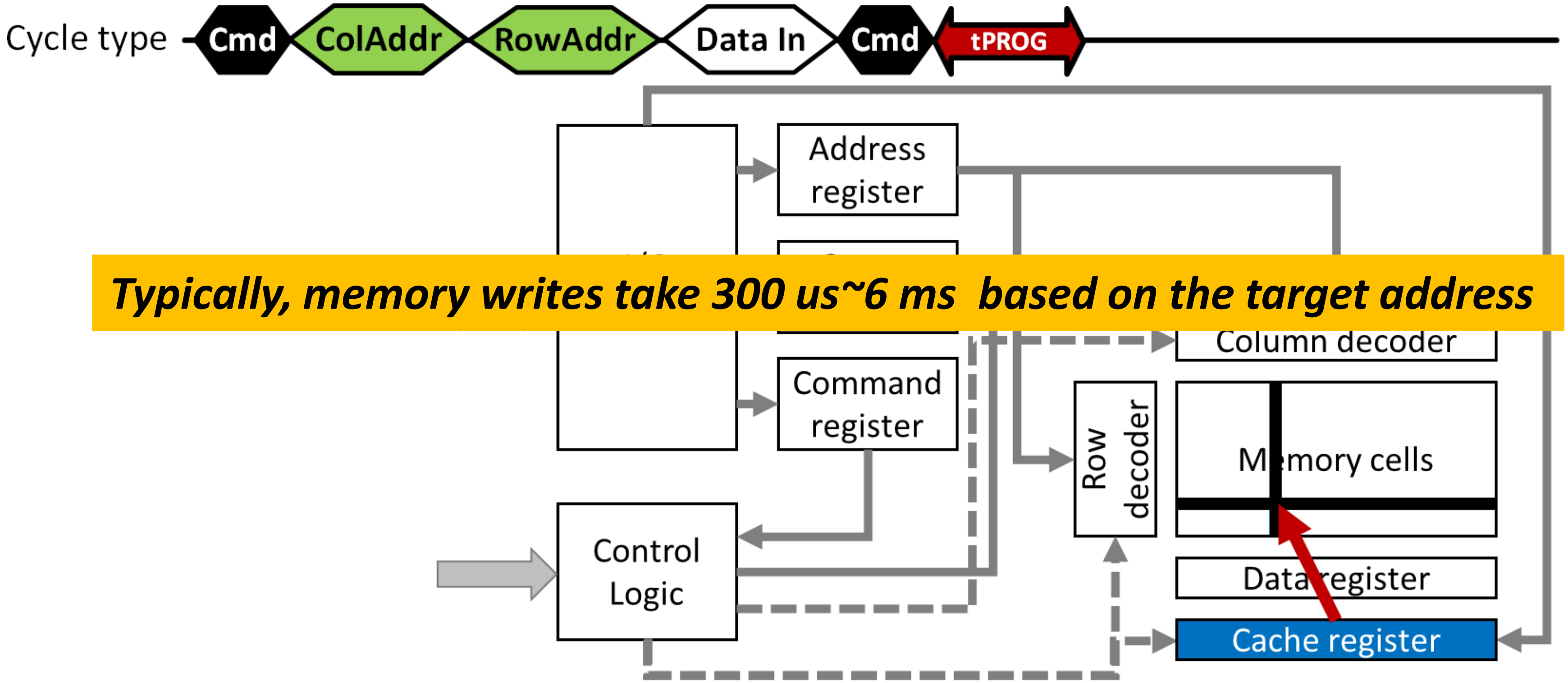
Page Write

- The first command is needed to indicate that this is a page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



Page Write

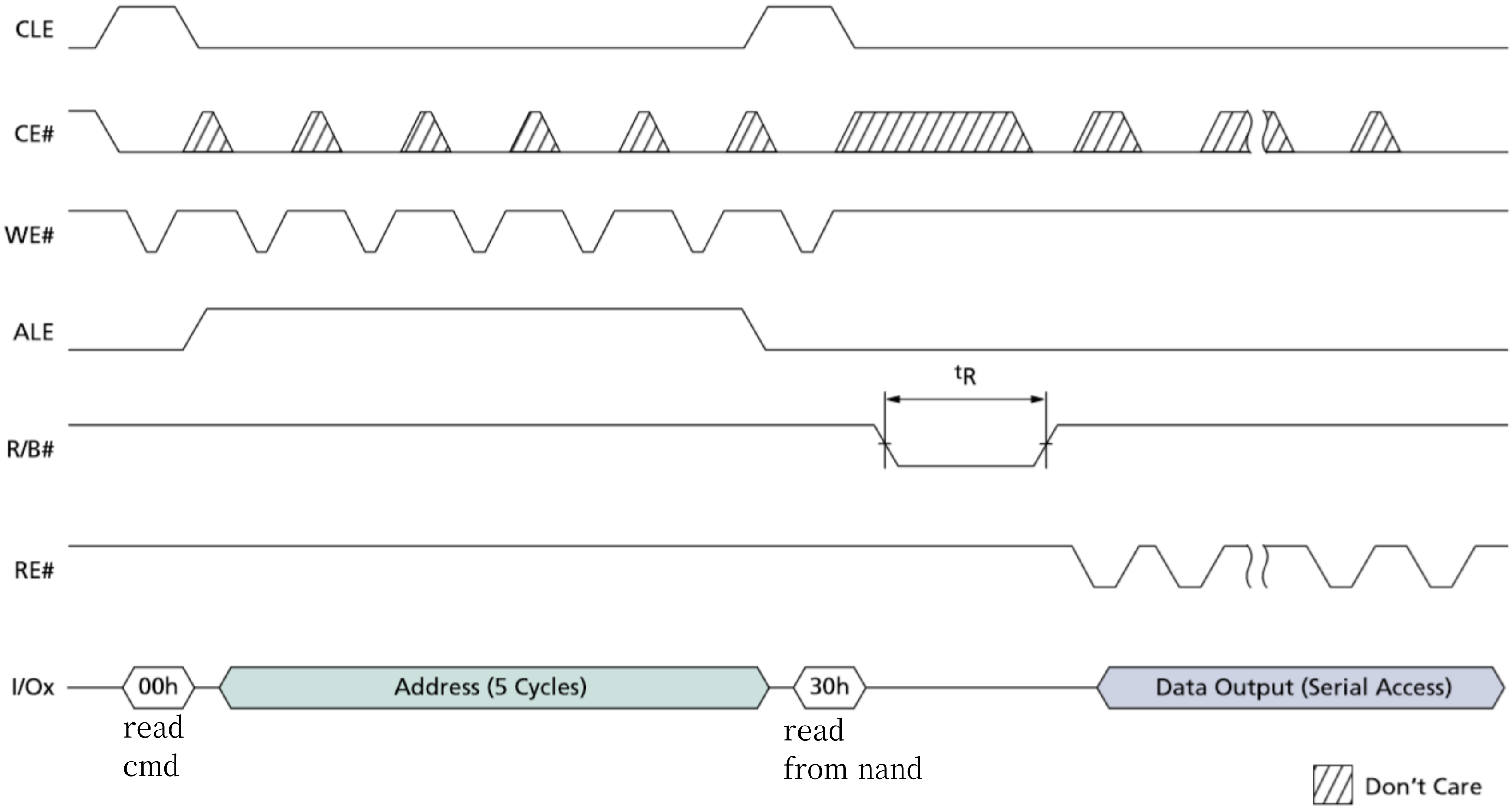
- The first command is needed to indicate that this is a page write
- The column address (C-ADDR) and row address (R-ADDR) are shipped in a serial order
- Data should be then transferred, and the second command initiates a write
- Once the write is performed, users need to send the last command to check up the status of target



Typically, memory writes take 300 us~6 ms based on the target address

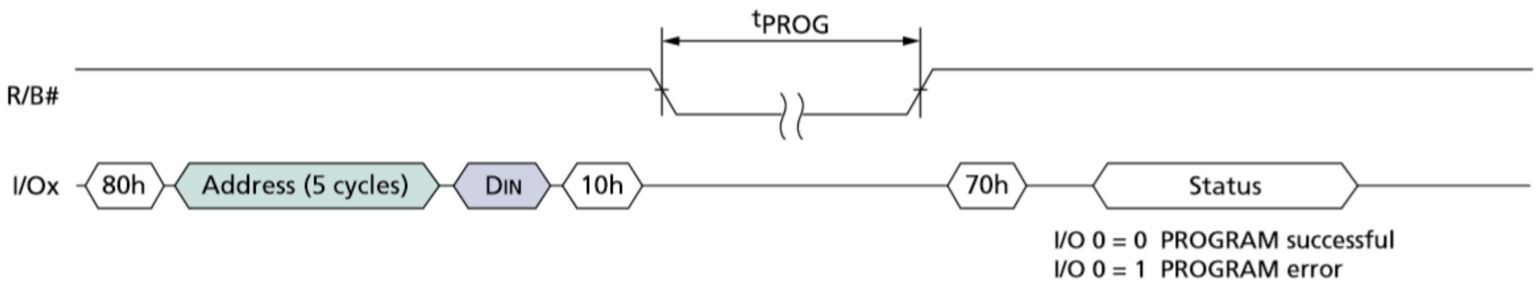
Waveform: Page Read

Figure 18: PAGE READ Operation



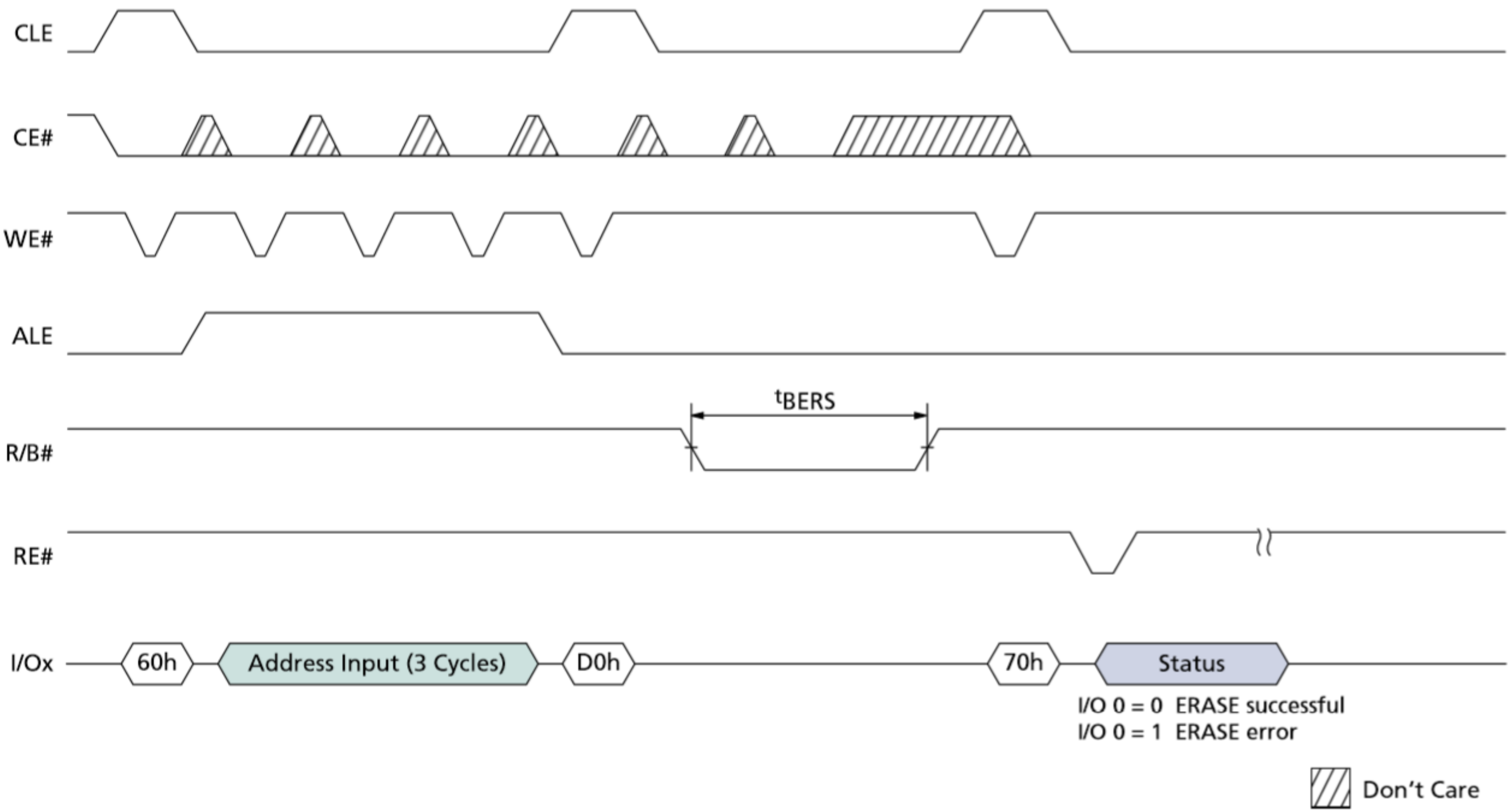
Waveform: Page Write

Figure 23: PROGRAM and READ STATUS Operation



Waveform: Block Erasure

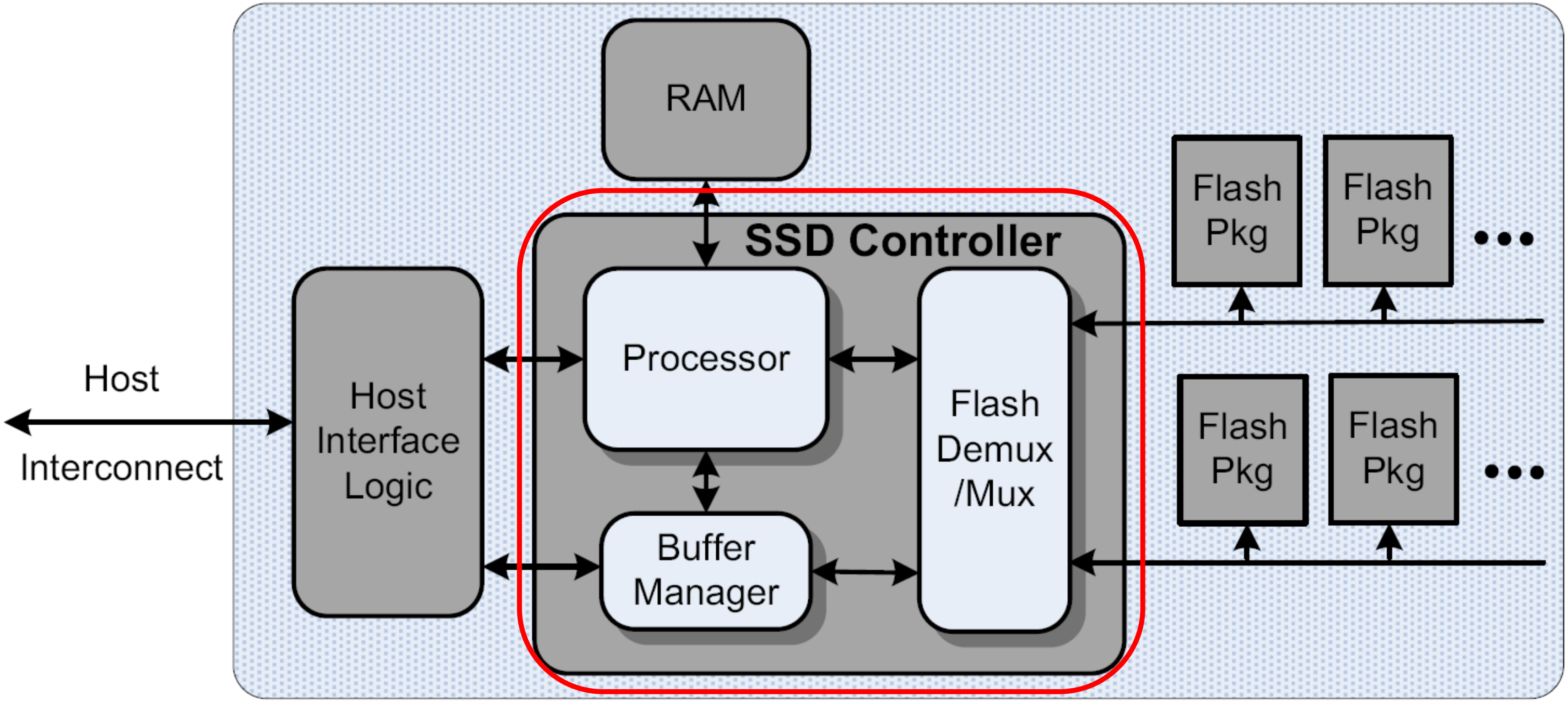
Figure 28: BLOCK ERASE Operation



Outline

- Storage Abstraction & Protocols
- Flash Packages
- SSD Controller
- Flash Array & I/O Parallelism

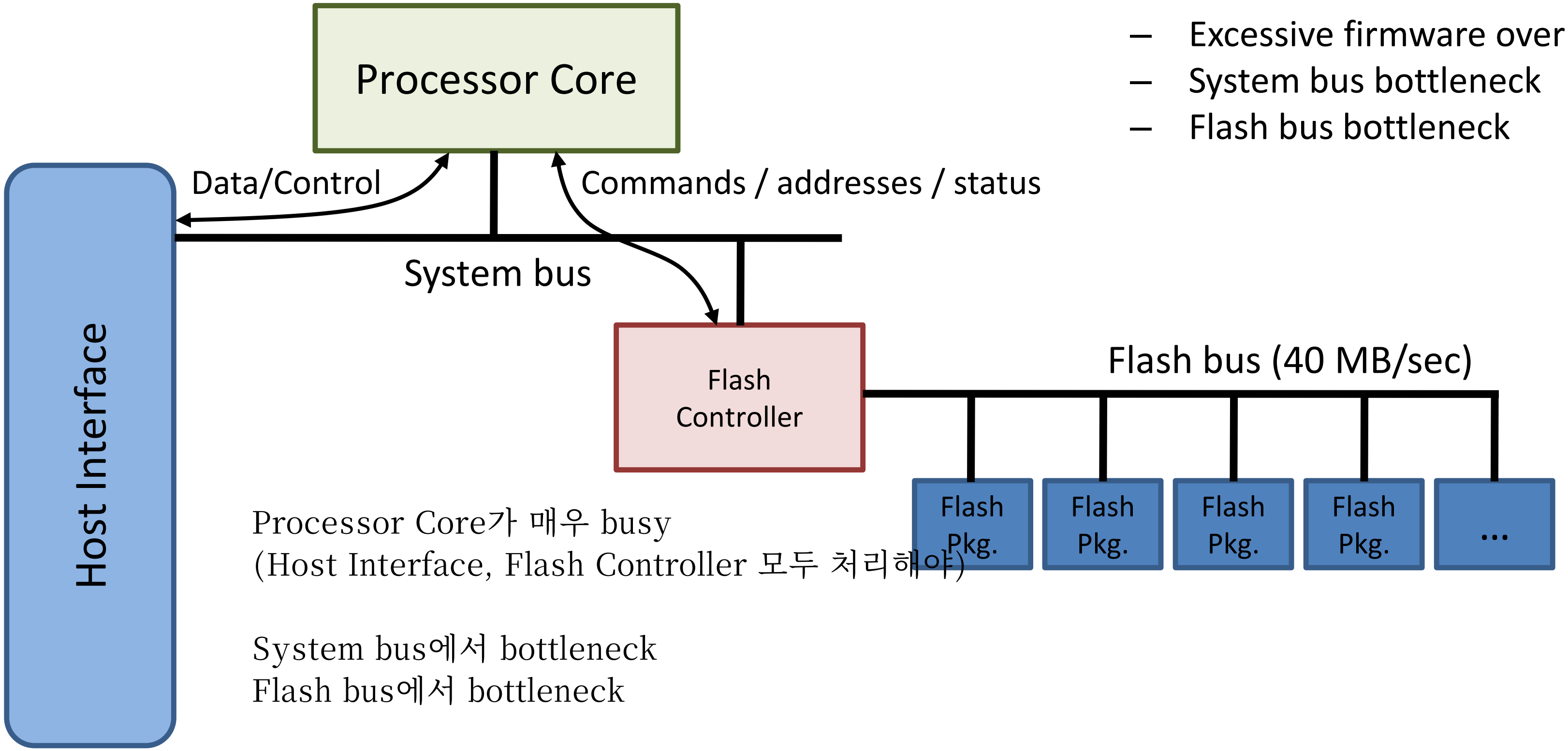
SSD Overview



Simplest Controller Architecture

- ARM CPU deals with host commands, managing NAND packages directly

- Problems
  - Excessive firmware overhead
  - System bus bottleneck
  - Flash bus bottleneck



Processor Core가 매우 busy
(Host Interface, Flash Controller 모두 처리해야)

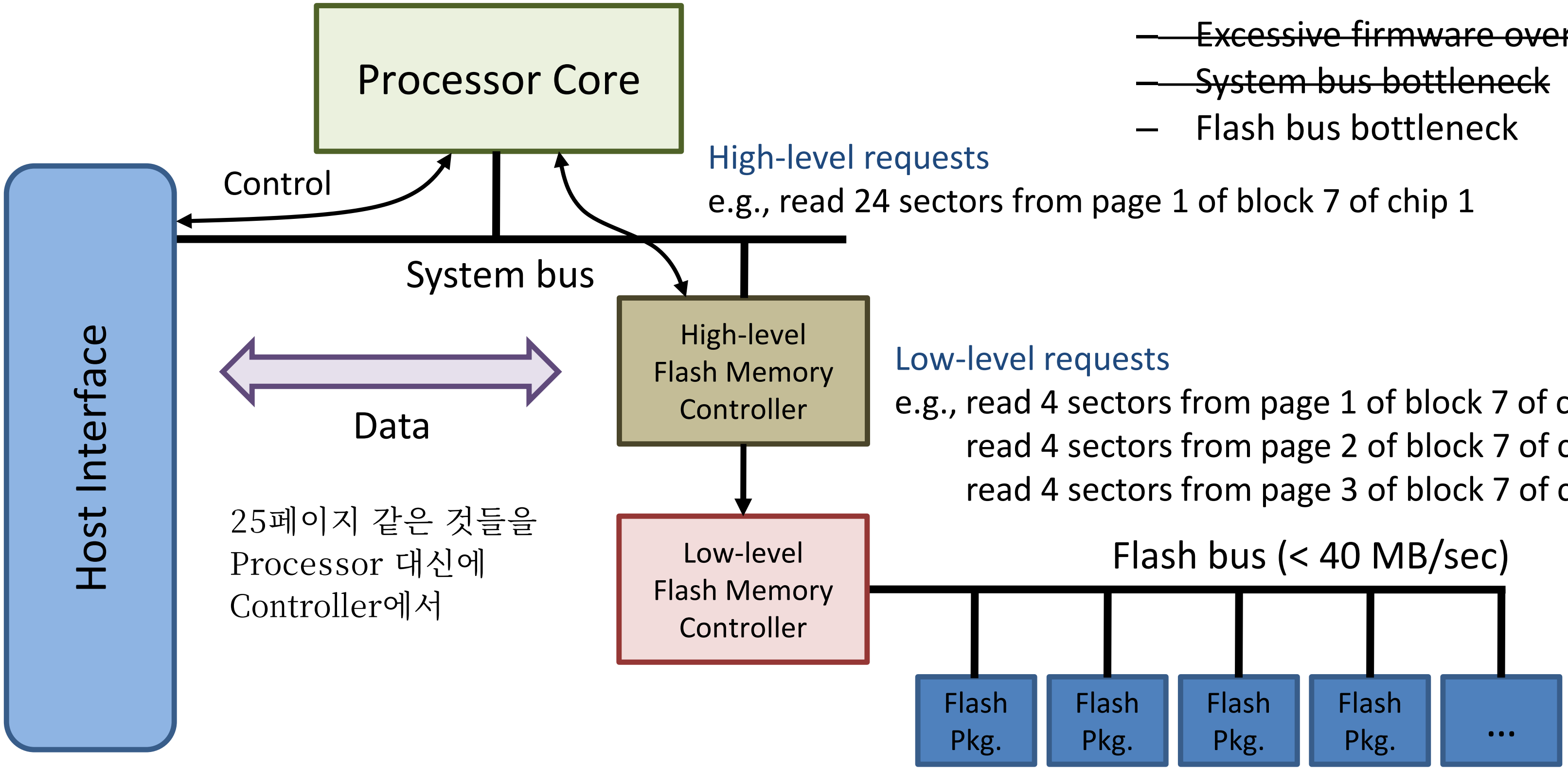
System bus에서 bottleneck
Flash bus에서 bottleneck

Bandwidth = 500 MB/sec
(SATA 3.0)

High-level Flash Memory Controller

- Offload some management duty to high-level controller
- Use high-bandwidth bus for fast data transfer

- Problems
  - ~~Excessive firmware overhead~~
  - ~~System bus bottleneck~~
  - Flash bus bottleneck



High-level requests
e.g., read 24 sectors from page 1 of block 7 of chip 1

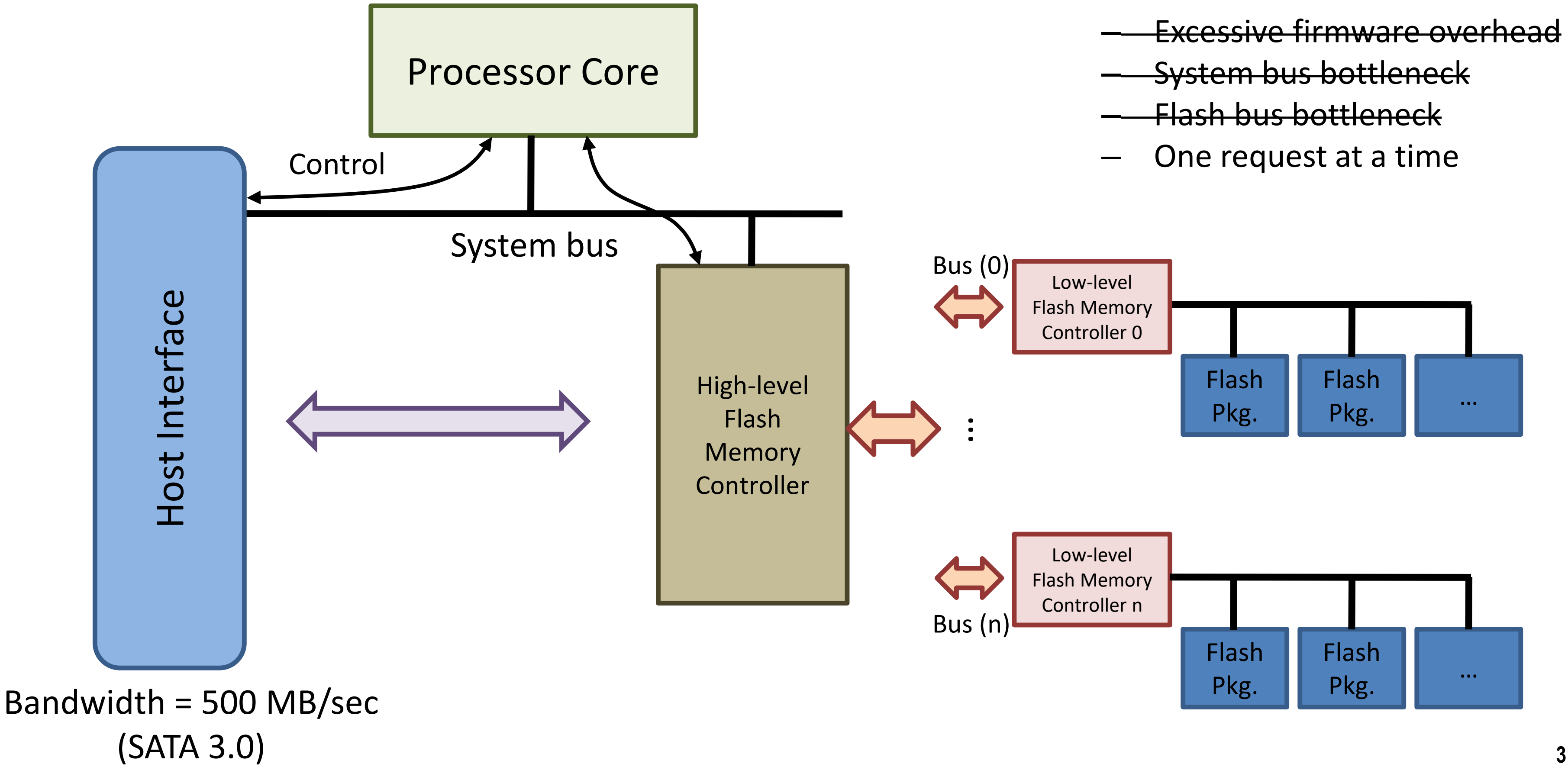
Low-level requests
e.g., read 4 sectors from page 1 of block 7 of chip1
read 4 sectors from page 2 of block 7 of chip1
read 4 sectors from page 3 of block 7 of chip1

25페이지 같은 것들을
Processor 대신에
Controller에서

Bandwidth = 500 MB/sec
(SATA 3.0)

Bus-level Interleaving

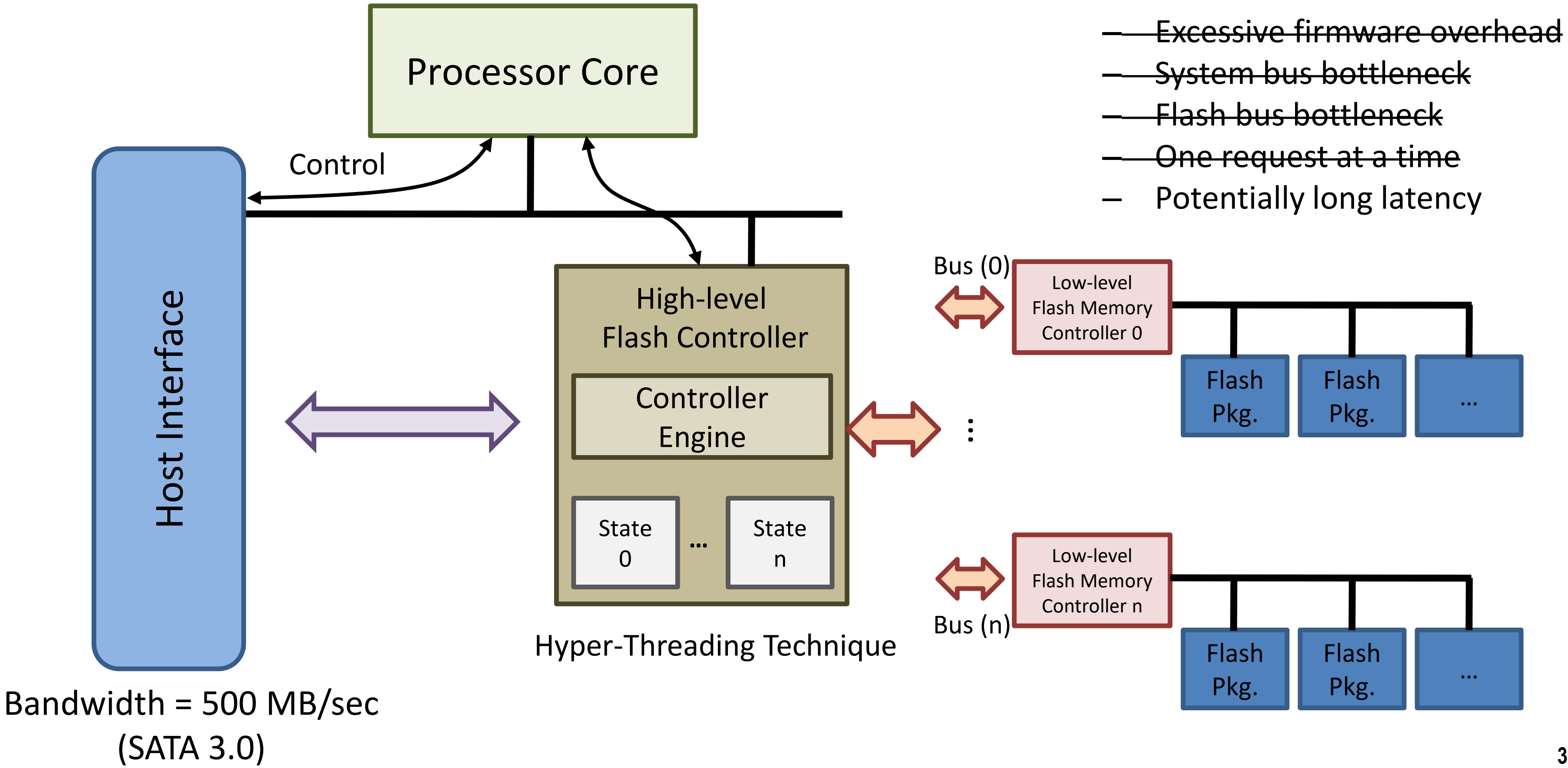
- Add more buses for a group of flash packages and send commands in the interleaved manner


Processor Core
Control
System bus
Host Interface
High-level Flash Memory Controller
Bus (0)
Low-level Flash Memory Controller 0
Flash Pkg.
Flash Pkg.
…
⋮
Low-level Flash Memory Controller n
Bus (n)
Flash Pkg.
Flash Pkg.
…
Bandwidth = 500 MB/sec (SATA 3.0)

- Problems
  - ~~Excessive firmware overhead~~
  - ~~System bus bottleneck~~
  - ~~Flash bus bottleneck~~
  - One request at a time

Multiple Controller Units

- Enhance the high-level controller so that it is able to cope with multiple buses at the same time



Bandwidth = 500 MB/sec
(SATA 3.0)

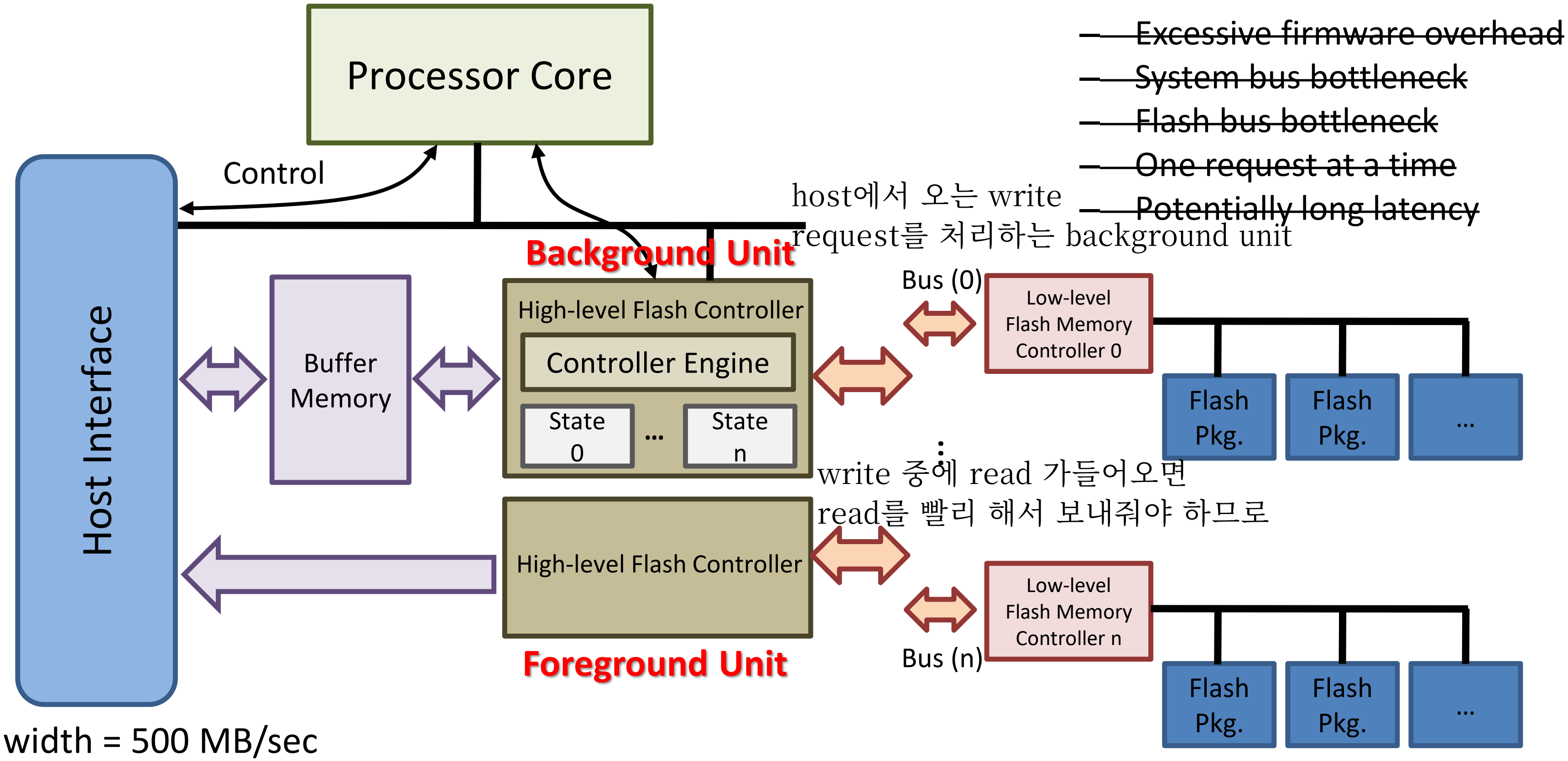
Hyper-Threading Technique

- Problems
  - ~~Excessive firmware overhead~~
  - ~~System bus bottleneck~~
  - ~~Flash bus bottleneck~~
  - ~~One request at a time~~
  - Potentially long latency

Buffering and Prioritized Handling

- Manage reads and writes with different priority
- Buffer writes in DRAM temporary and flush them later

- Problems
  - ~~Excessive firmware overhead~~
  - ~~System bus bottleneck~~
  - ~~Flash bus bottleneck~~
  - ~~One request at a time~~
  - ~~Potentially long latency~~



host에서 오는 write request를 처리하는 background unit

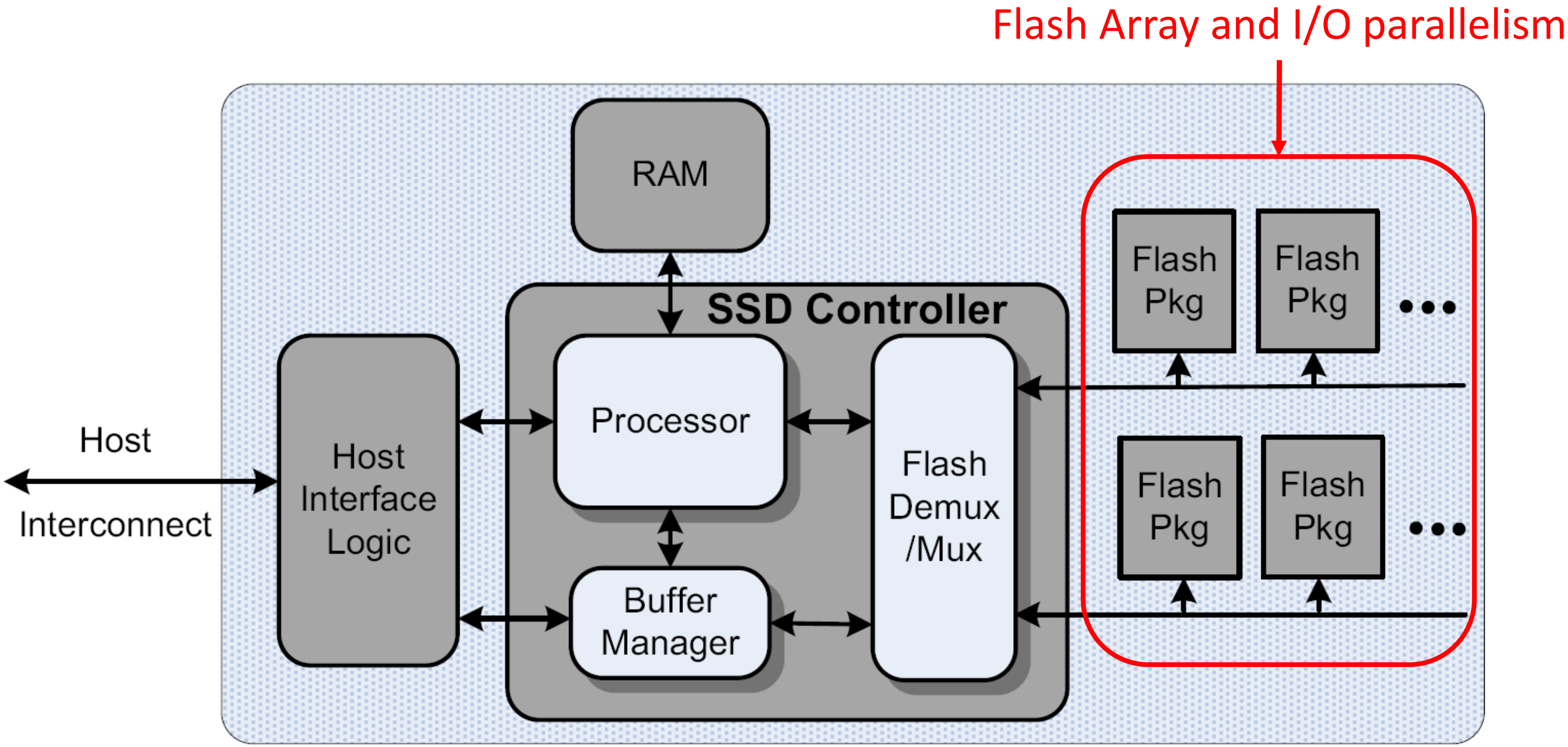
write 중에 read 가들어오면 read를 빨리 해서 보내줘야 하므로

Bandwidth = 500 MB/sec
(SATA 3.0)

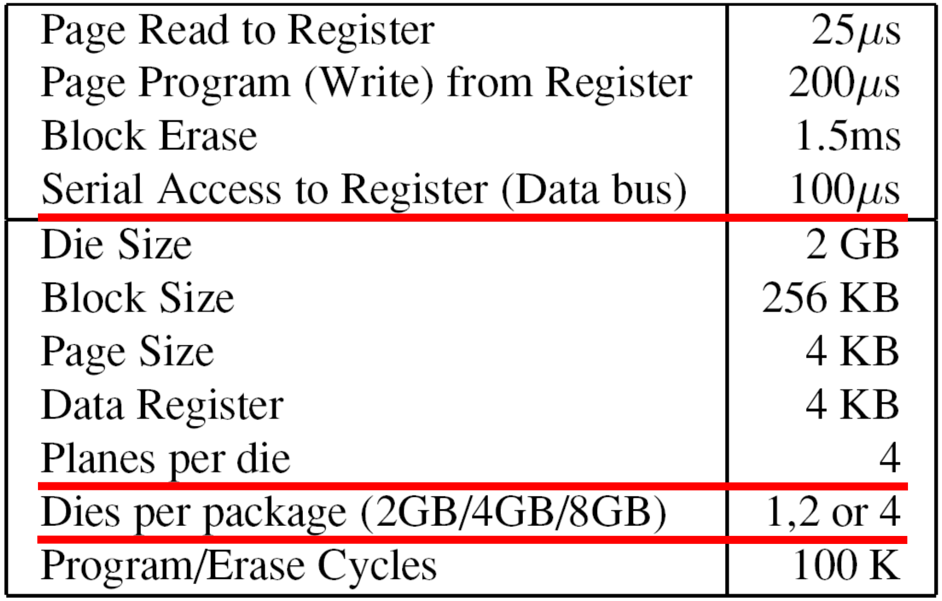
Outline

- Storage Abstraction & Protocols
- Flash Packages
- SSD Controller
- **Flash Array & I/O Parallelism**

SSD Overview



NAND Chip Parameters



Page Read to Register	$25\mu s$
Page Program (Write) from Register	$200\mu s$
Block Erase	1.5ms
Serial Access to Register (Data bus)	$100\mu s$
Die Size	2 GB
Block Size	256 KB
Page Size	4 KB
Data Register	4 KB
Planes per die	4
Dies per package (2GB/4GB/8GB)	1,2 or 4
Program/Erase Cycles	100 K

2007

(*Note: very old parameters)

- **One page read**
  - Read a page from the media into the data register (25 us)
  - Shift 4 KB data out over the data bus (100 us)
- **One page write**
  - Shift 4 KB data into the data register (100 us)
  - Write a page from the data register to a flash page (200 us)

The Serial Bus is a Primary Bottleneck

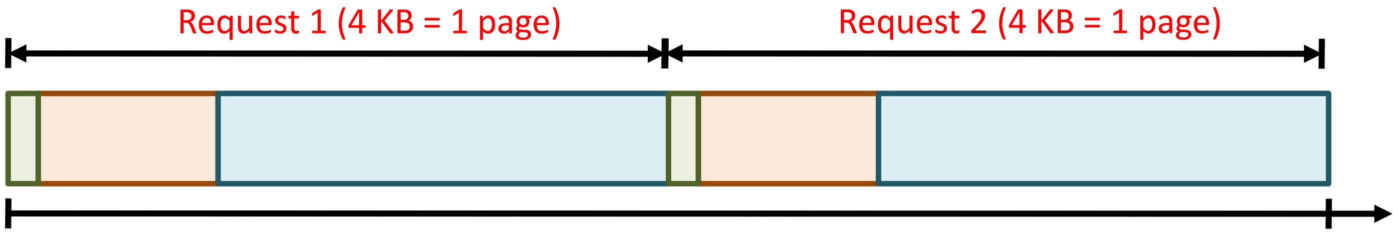
- **Two page reads**

🟩 Time to send control signals (< 1 us)

🟧 Time to move 4 KB page data to on-chip register (25 us)

🟦 Time to move 4 KB on-chip data to flash page (200 us)

⬜ Time to move 4 KB page data to the controller through the serial bus (100 us)

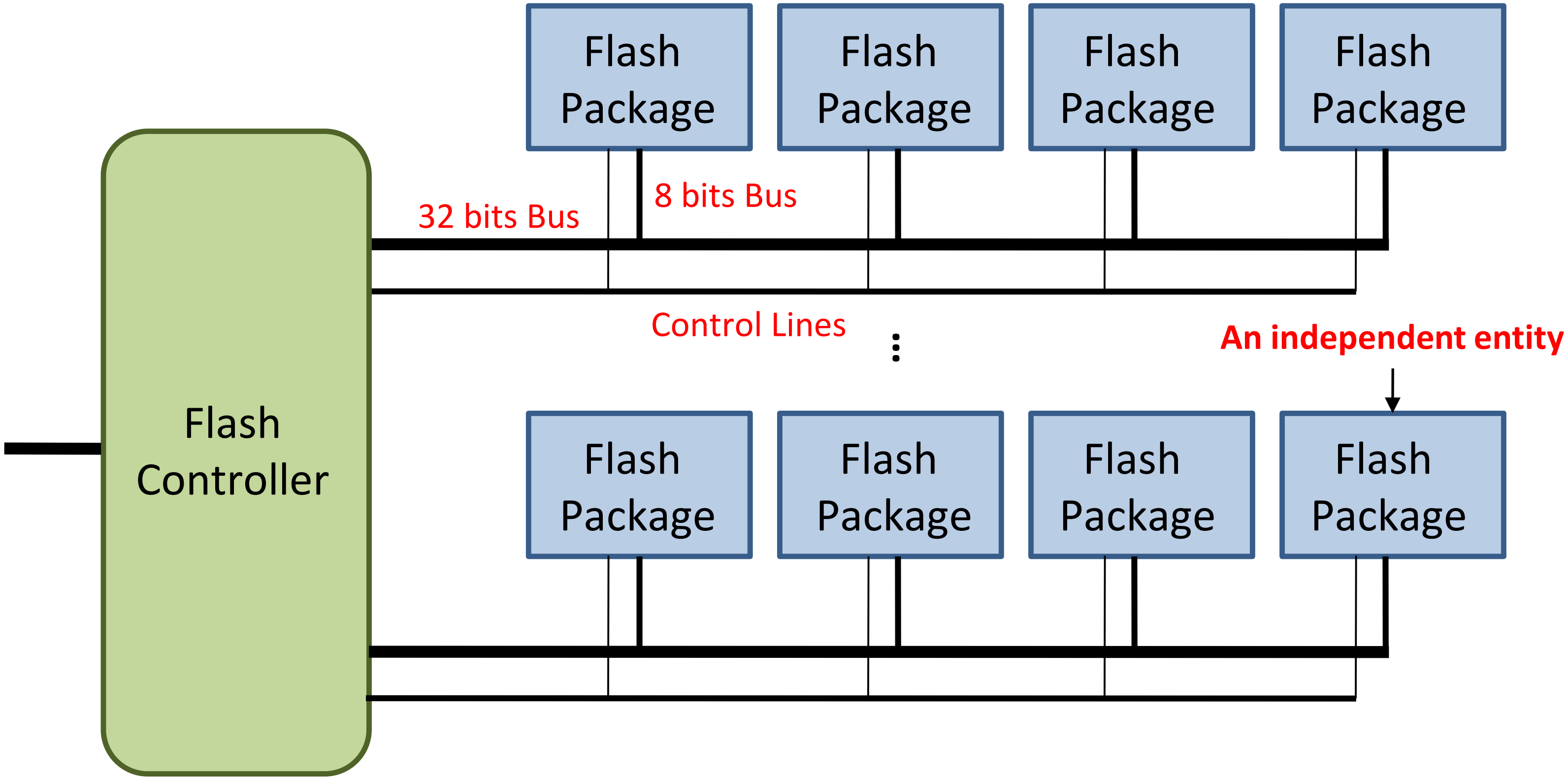


Total time to read 8 KB page = 250 us

- **Solution – Use parallelism of a flash array**
  - (1) ***Plane-level Pipelining***, (2) ***Interleaving***, and (3) ***Stripping***

Flash Array

- **SSD handles requests on flash packages in parallel**
  - To achieve bandwidths or I/O rates greater than a single-chip



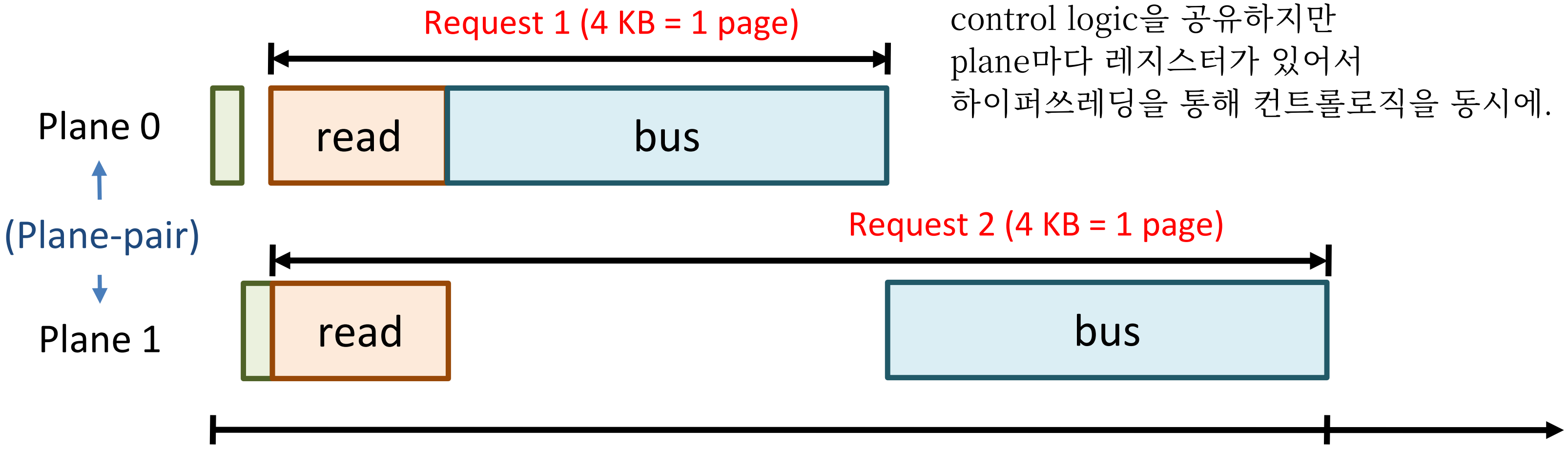
〈Fully connected flash array architecture〉

Plane-level Pipelining: Read

- **Two page reads with pipelining**

bus는 하나니까 동시에 못한다.

control logic을 공유하지만
plane마다 레지스터가 있어서
하이퍼쓰레딩을 통해 컨트롤로직을 동시에.

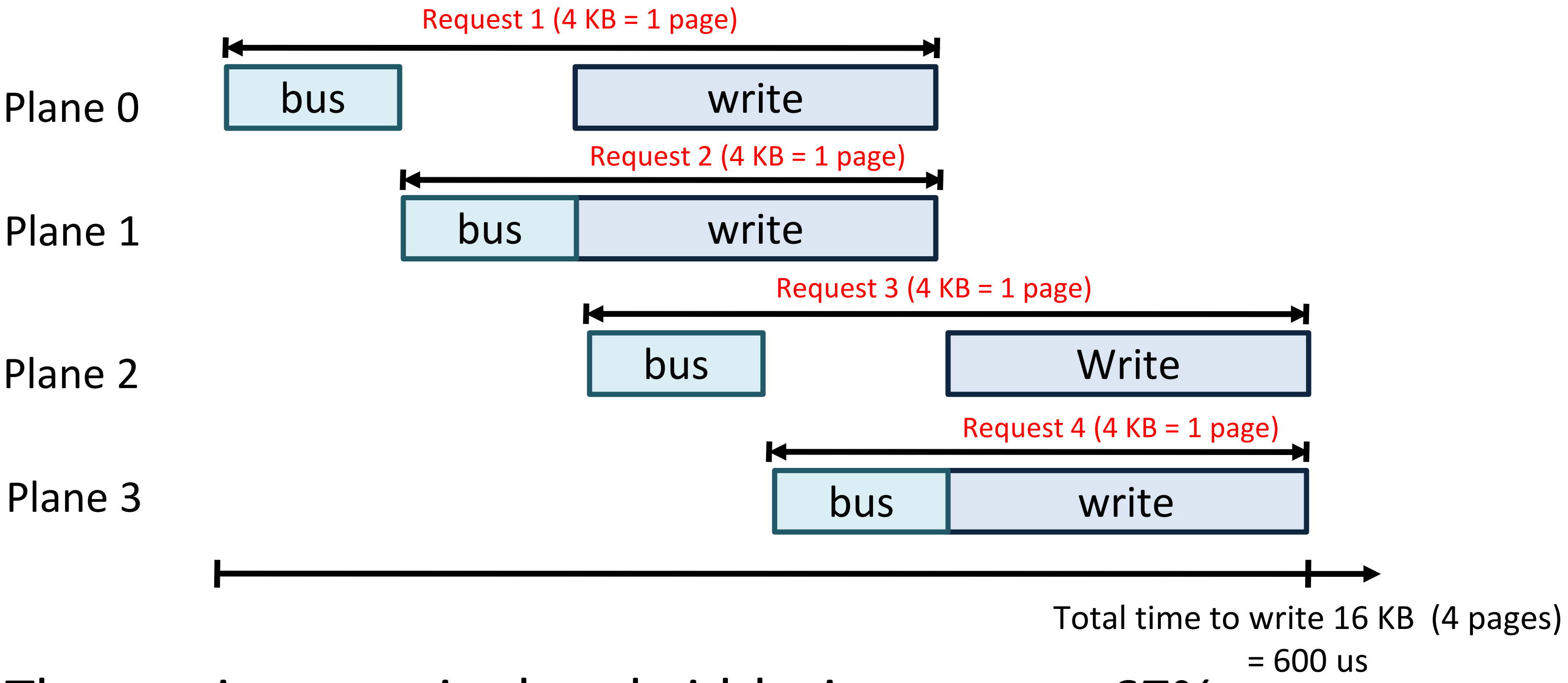


Total time to read 8 KB (2 pages)
= 225 us

- The maximum read bandwidth improves to 20%
  - One page read requires about 100 usec

Plane-level Pipelining: Write

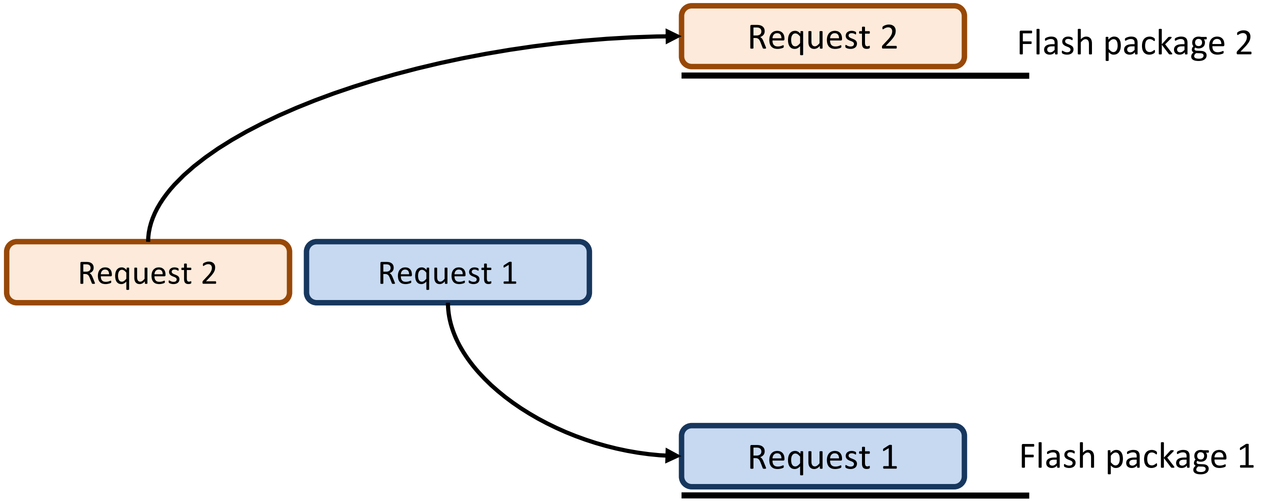
- **Four page writes with pipelining**


Request 1 (4 KB = 1 page)
Plane 0
bus
write
Request 2 (4 KB = 1 page)
Plane 1
bus
write
Request 3 (4 KB = 1 page)
Plane 2
bus
Write
Request 4 (4 KB = 1 page)
Plane 3
bus
write
Total time to write 16 KB (4 pages)
= 600 us

- The maximum write bandwidths improve to 67%
  - One page write requires about 100 usec

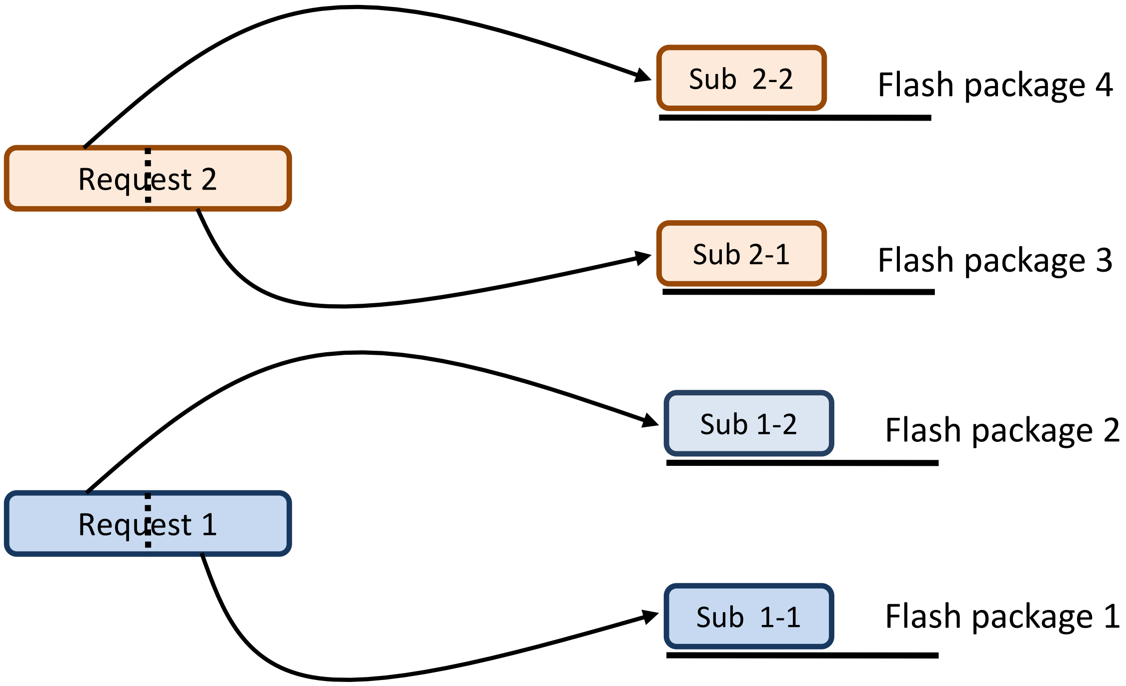
Interleaving Technique

- Exploiting inter-request parallelism using multiple flash packages



Stripping Technique

- **Exploiting intra-request parallelism by spreading out a request across multiple packages**

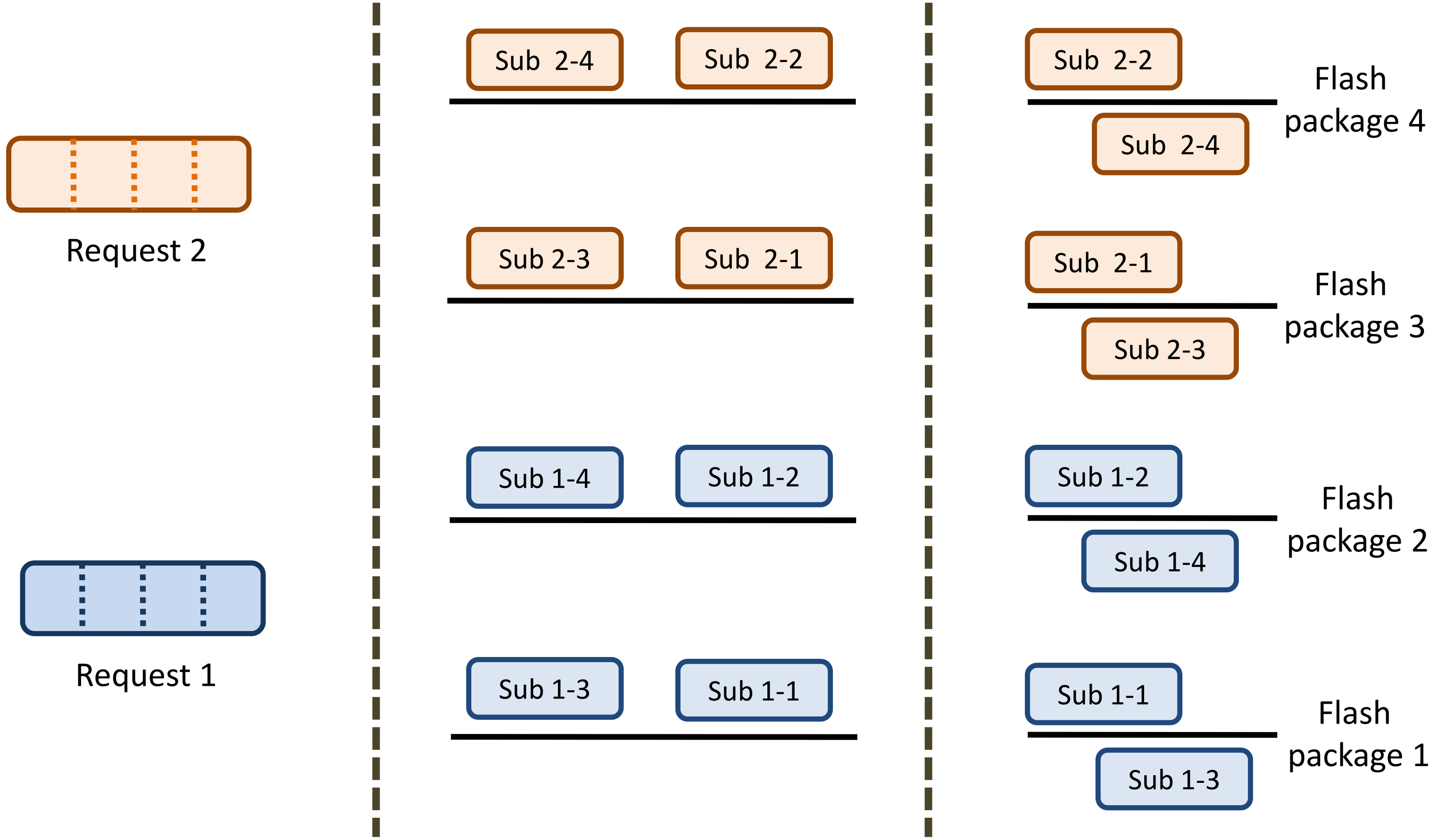


Putting All Together

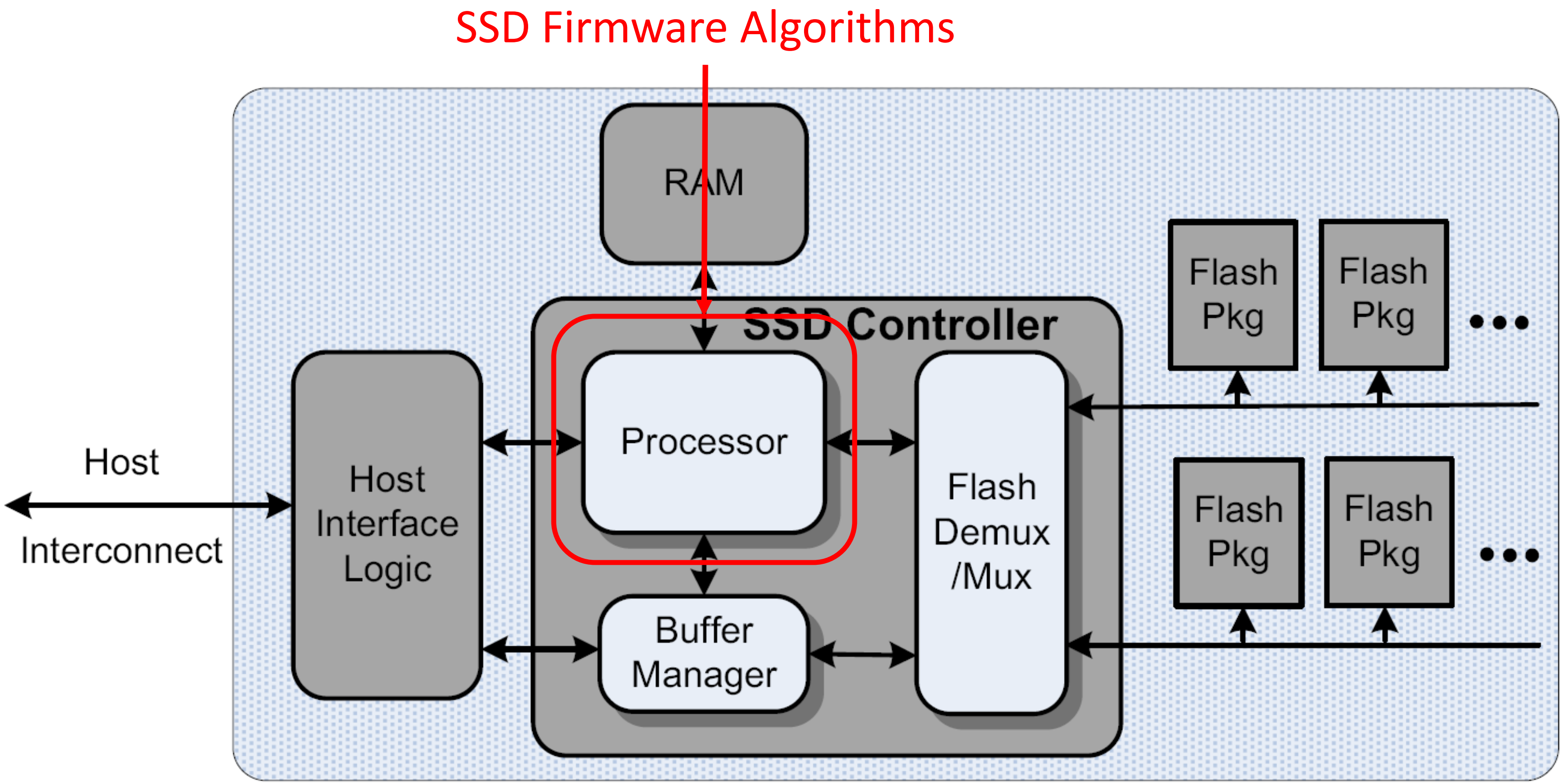
Interleaving technique

Stripping technique

Plane-level Pipelining



SSD Firmware Algorithms



SSD Firmware Algorithms (Cont.)

사실 저런 파이프라이닝이나 다른 것들도
다 알고리즘이 핵심

- **Some form of mapping between logical address and physical flash location is required in NAND flash**
  - The SSD controller receives a list of LBAs with commands, which must be mapped to specific pages of blocks in flash packages

- **Major concerns in designing a logical block map (address mapping) algorithm**
  - Mapping table size
  - Garbage collection
  - Load balancing
  - Parallel access

Decide SSD performance & reliability
(See in detail next week)

End of Chapter 3