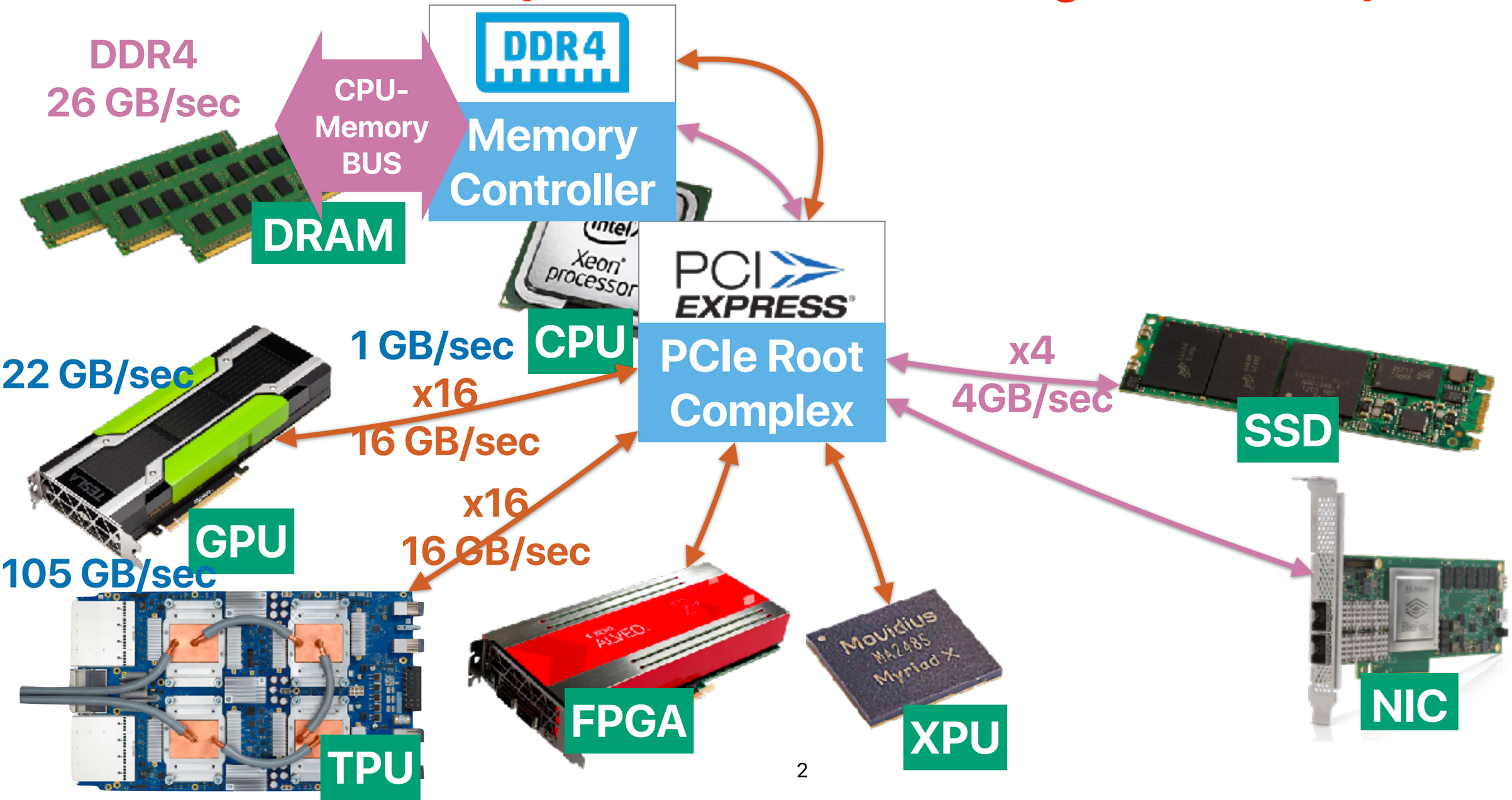
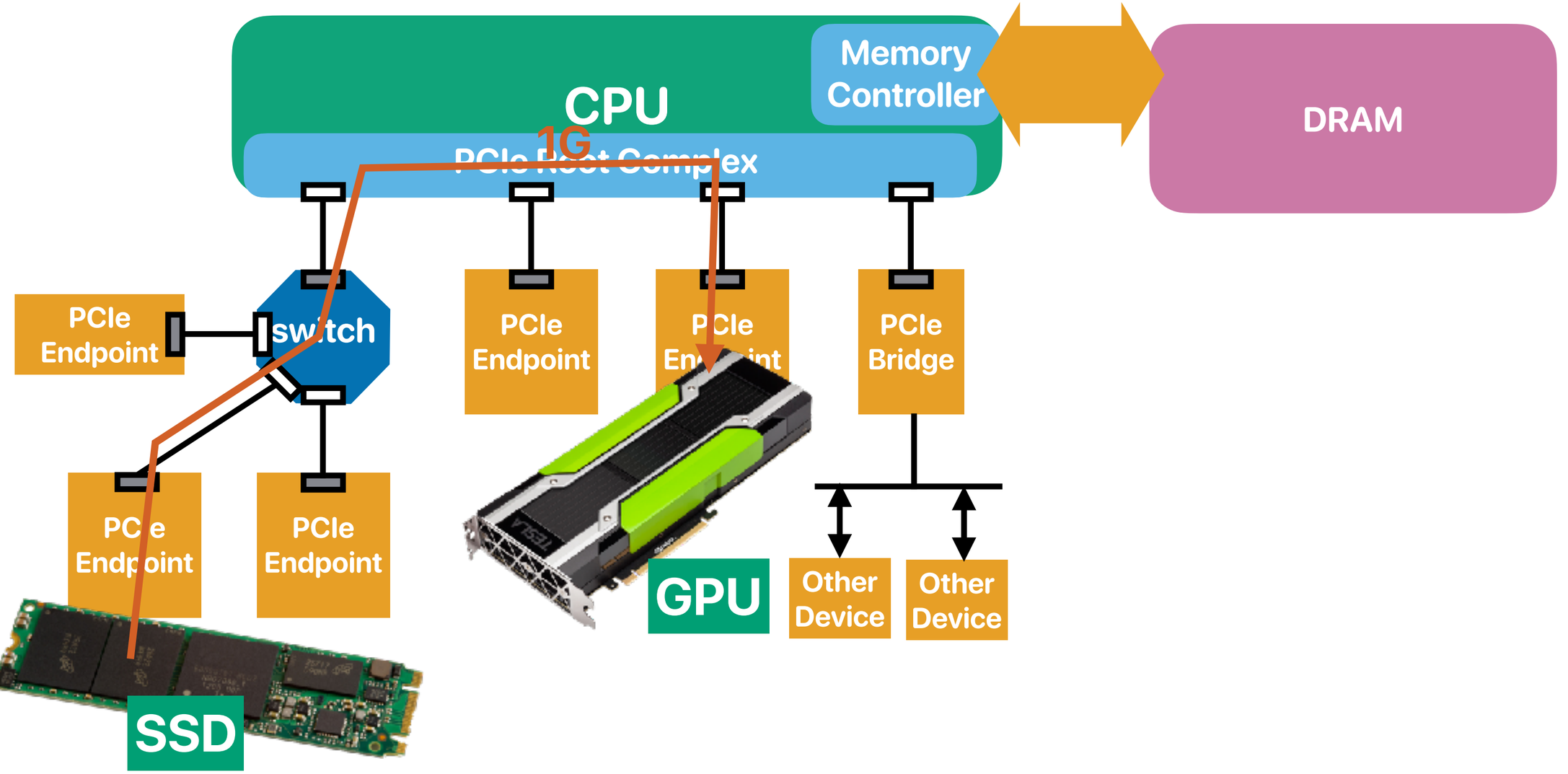
The Concept of Near Data Processing & In-Storage Processing

Hung-Wei Tseng

Review: The "data path" in modern heterogeneous computers



PCIe supports peer-to-peer communication!



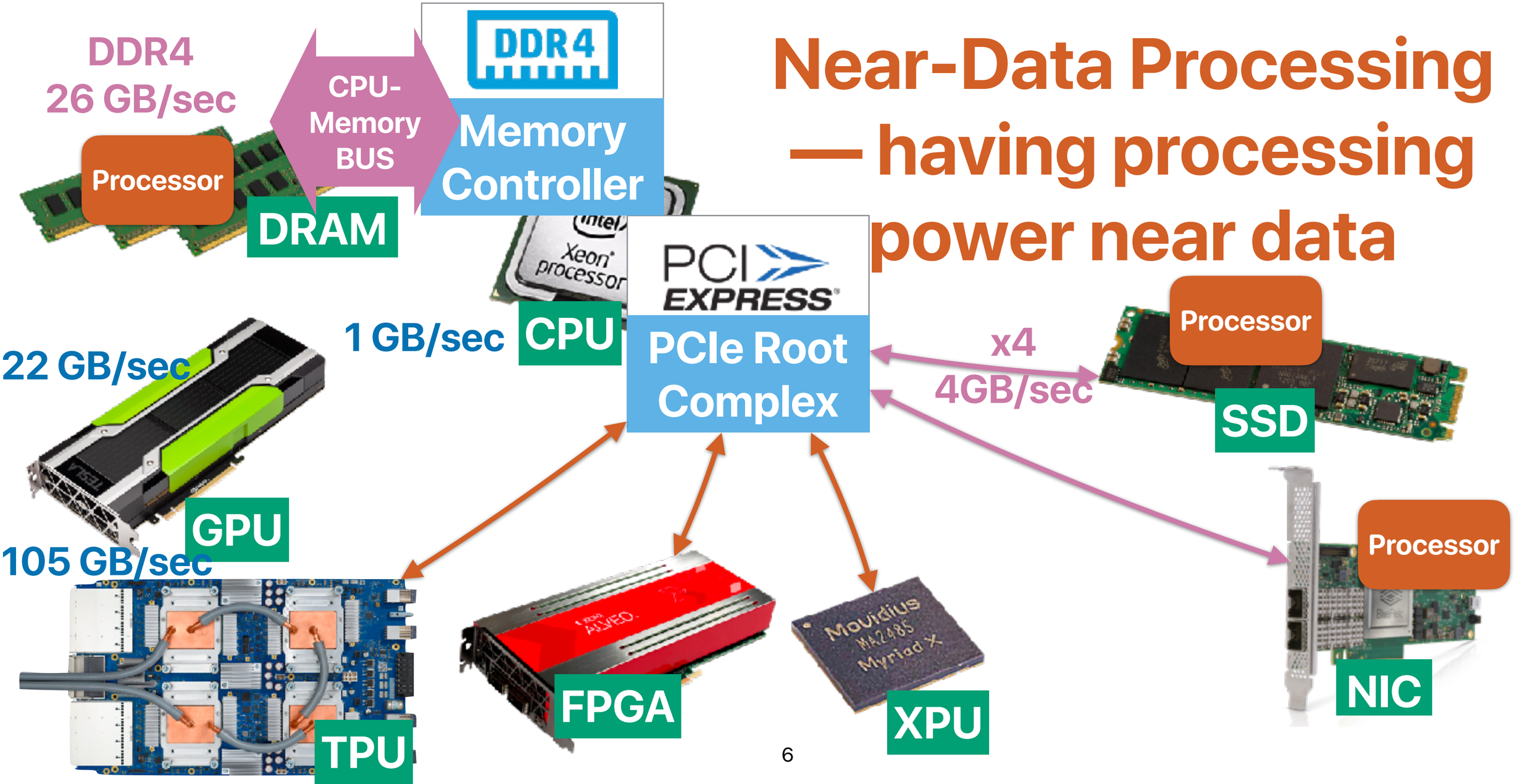
Recap: simply P2P is not enough

- If your data resides in SSD — the total out-going bandwidth from SSD is still way lower than the throughput of your computing resources
- If your dataset is large, you will need to partition your computation — DRAM is a better buffer than an SSD

Think different — do we really need to separate data and computation?

Think different

Near-Data Processing — having processing power near data



3:00

Under what criteria do you think NDP can improve performance?

Criteria where NDP can help

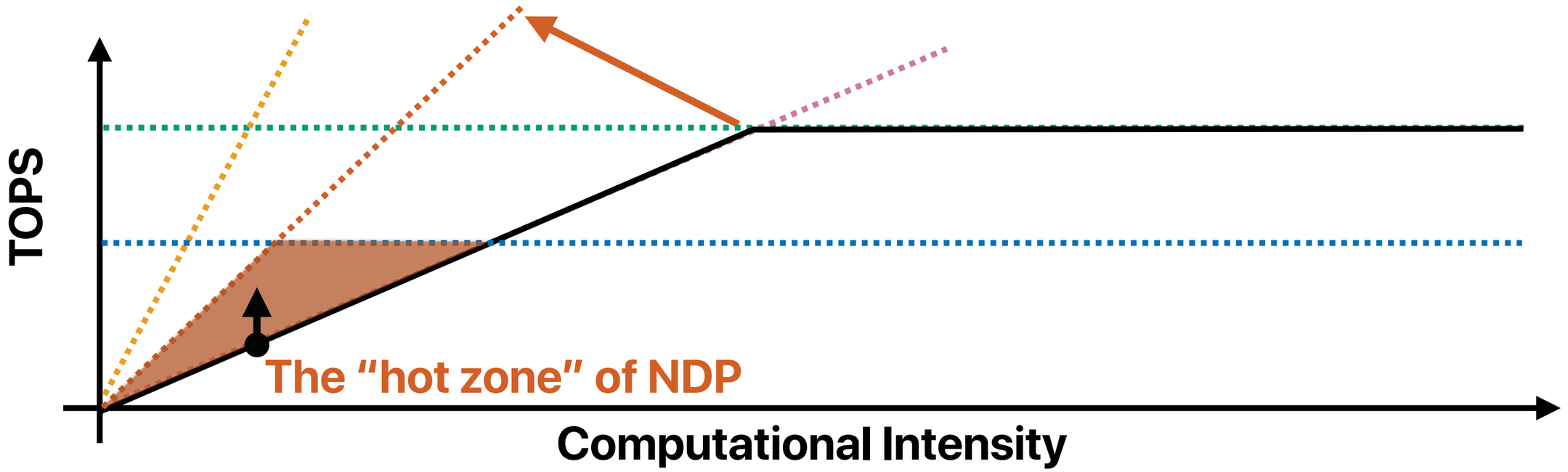
Reviewing the roofline model

min(Peak OPS of target computing resource)

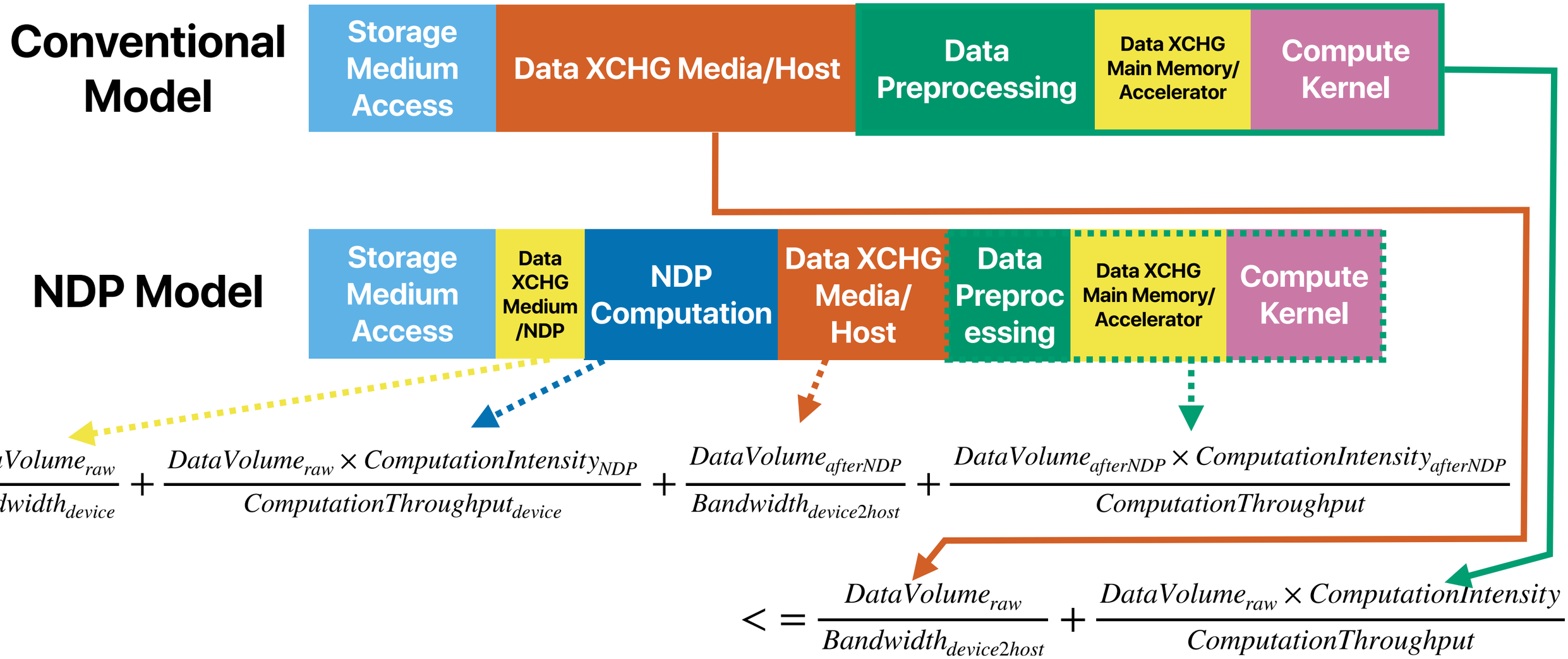
Peak memory bandwidth × computational intensity ÷ reduction of data volume

min(Peak OPS of target NDP device)

Peak device internal bandwidth × computational intensity of NDP program



The "Winning Formula" of Near-Data Processing



$$\frac{DataVolume_{raw}}{Bandwidth_{device}} + \frac{DataVolume_{raw} \times ComputationIntensity_{NDP}}{ComputationThroughput_{device}} + \frac{DataVolume_{afterNDP}}{Bandwidth_{device2host}} + \frac{DataVolume_{afterNDP} \times ComputationIntensity_{afterNDP}}{ComputationThroughput}$$

$$<= \frac{DataVolume_{raw}}{Bandwidth_{device2host}} + \frac{DataVolume_{raw} \times ComputationIntensity}{ComputationThroughput}$$

The "Winning Formula" of Near-Data Processing

Conventional Model

NDP Model

- The NDP computation can help offload computation
- The NDP computation can help reduce data volume
- The NDP computation can facilitate data preprocessing
- The NDP computation itself cannot be too slow

Not the cases of Near-Data Processing

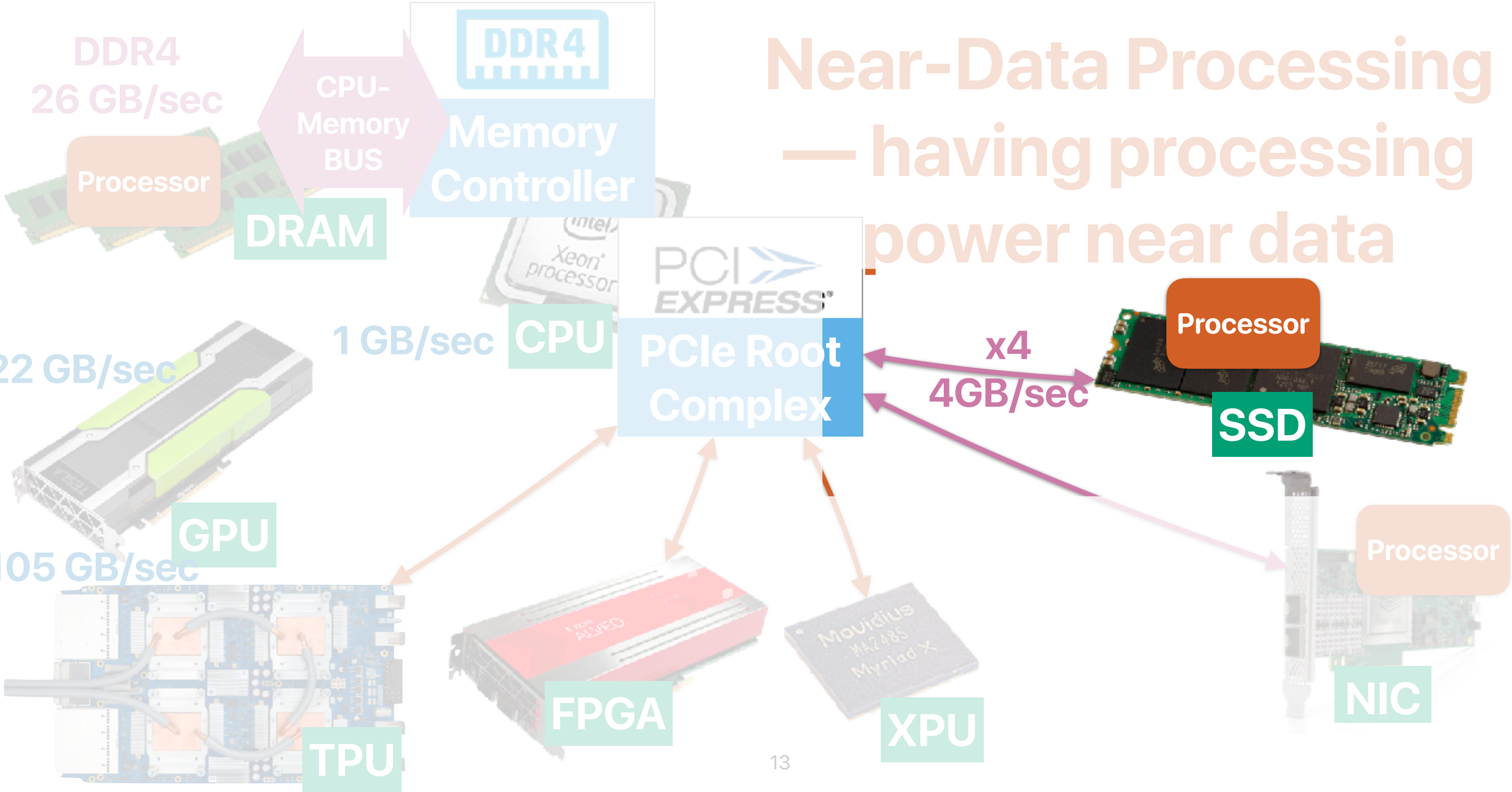
Conventional Model

NDP Model

- The NDP processing exceeds NDP computing resources' capabilities
- The NDP processing does not help reduce data volume
- The NDP device itself does not offer rich internal bandwidth to the computing resource near-by

Think different

Near-Data Processing — having processing power near data



3:00

What’s inside an SSD and why do we need them?

What an SSD looks like

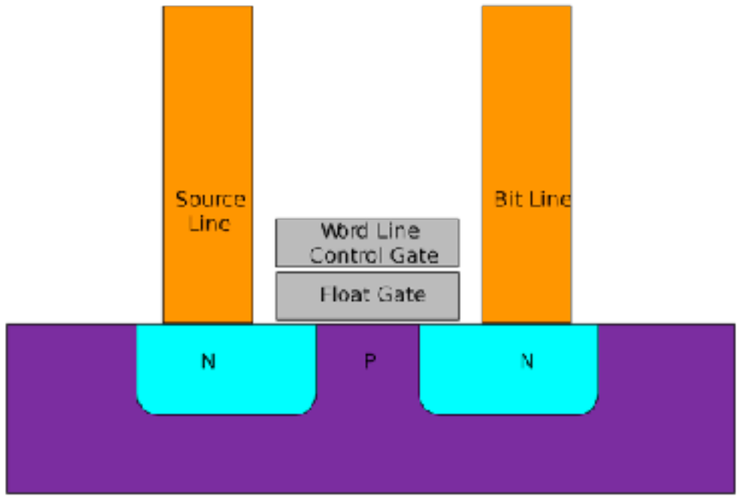
Controller + Registers

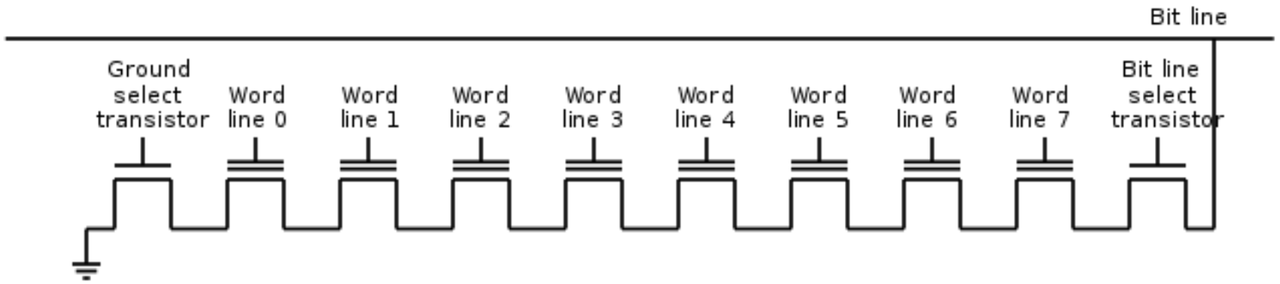
Why do we need controllers in SSDs?

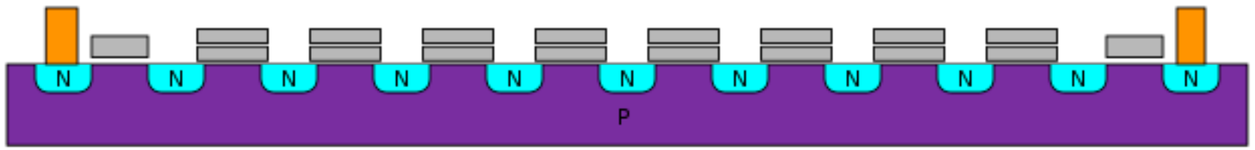
- Interfacing with the interconnect
- Maintaining the block device abstraction
- Dealing with the “weird” device characteristics

Flash memory

- Floating gate made by polycrystalline silicon trap electrons
- The voltage level within the floating gate determines the value of the cell
- The floating gates will wear out eventually



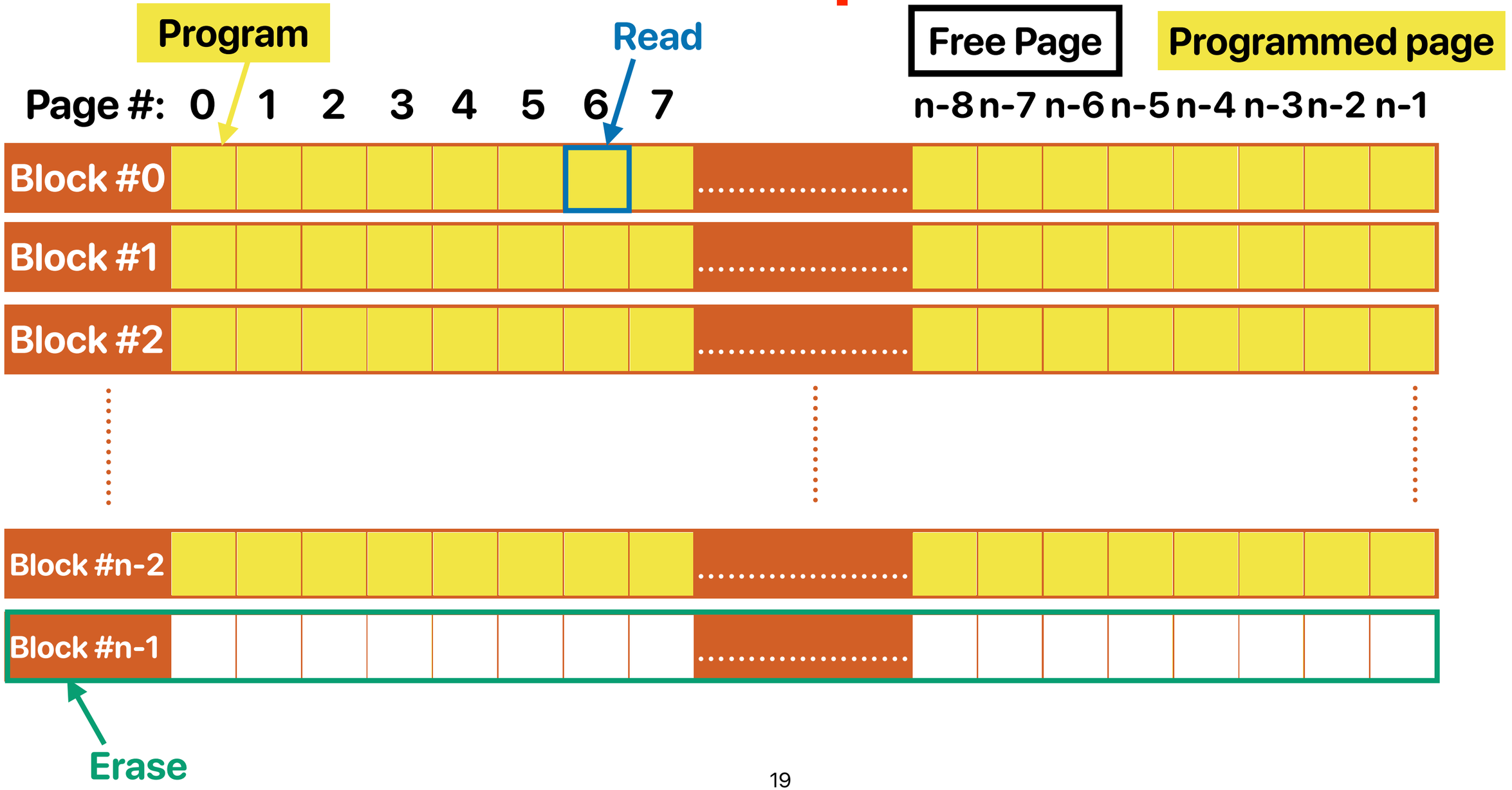




NAND flash is just odd!

- Modern SSDs are based on NAND flash memory
- Different operation granularities
  - Read/Program in pages
  - Erase in blocks (64-384 pages)
- Performance of operation varies
  - Read — tens of us
  - Program — hundreds of us
  - Erase — ms
- Limited erase cycles
  - Only 1000 times for “QLC”

Basic flash operations



Why we need a processor in SSDs?

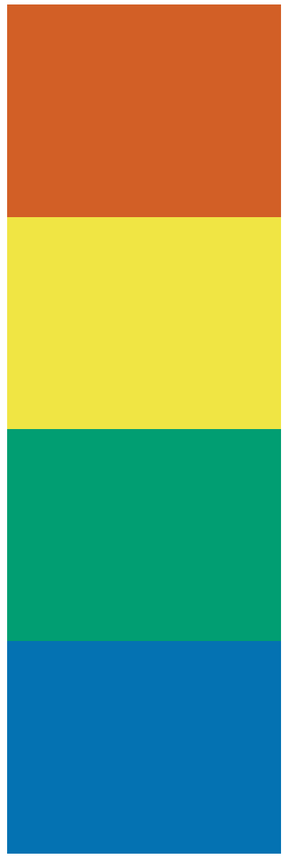
- Modern SSDs are based on NAND flash memory
- Different operation granularities
  - Read/Program in pages
  - Erase in blocks (64-384 pages)
- Performance of operation varies
  - Read — tens of us
  - Program — hundreds of us
  - Erase — ms
- Limited erase cycles
  - Only 1000 times for “QLC”

Types of Flash Chips

2 voltage levels, 1-bit

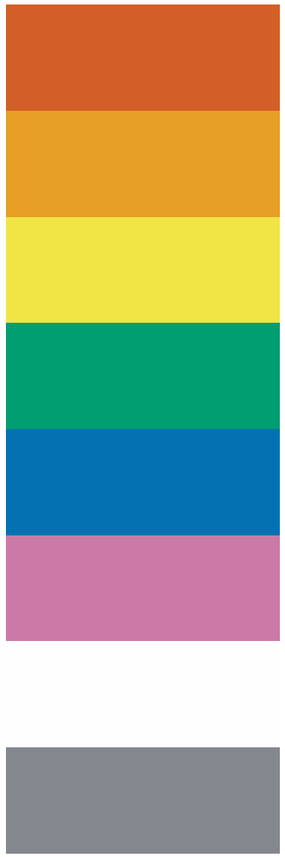
Single-Level Cell (SLC)

4 voltage levels, 2-bit



Multi-Level Cell (MLC)

8 voltage levels, 3-bit



Triple-Level Cell (TLC)

16 voltage levels, 4-bit

Quad-Level Cell (QLC)

Programming in MLC

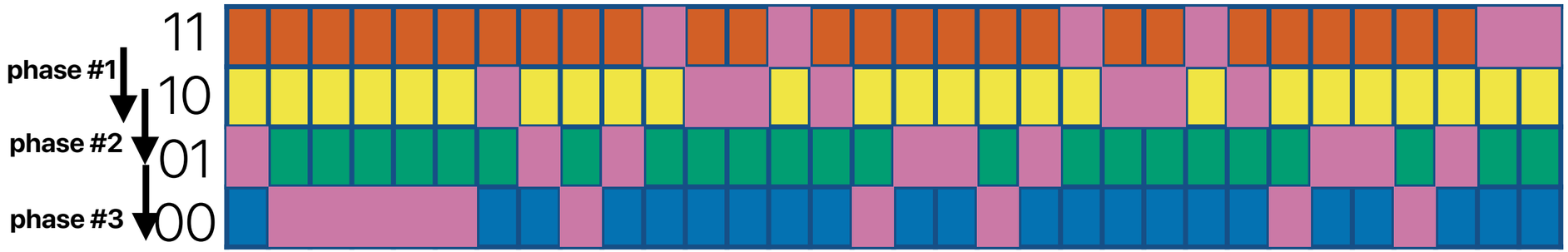
4 voltage levels,
2-bit

Multi-Level Cell (MLC)

3.14000000000000012434497875800

= 0x40091EB851EB851F

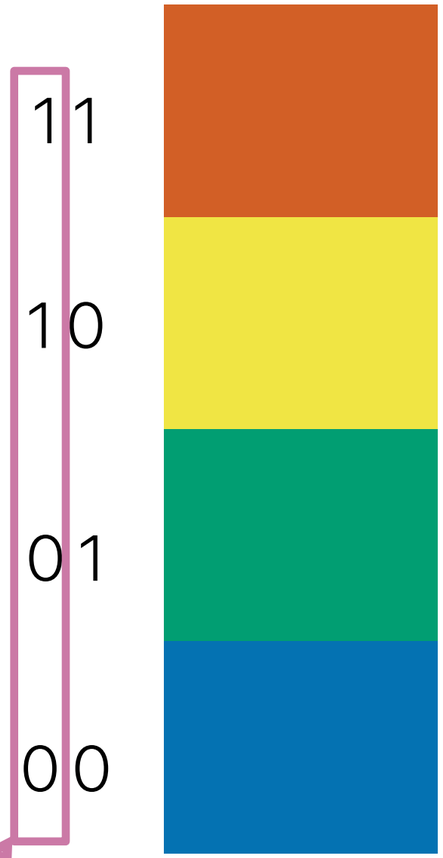
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



3 Cycles/Phases to finish programming

Programming in MLC

4 voltage levels,
2-bit



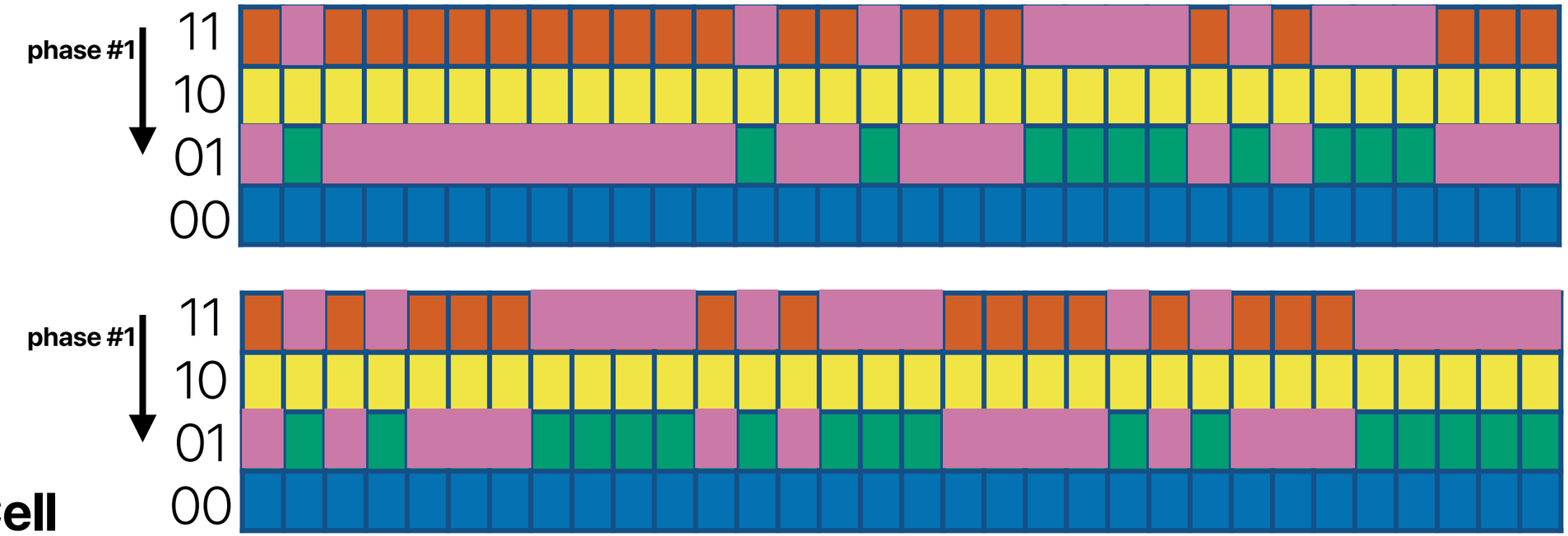
1st page

Multi-Level Cell (MLC)

3.14000000000000012434497875800

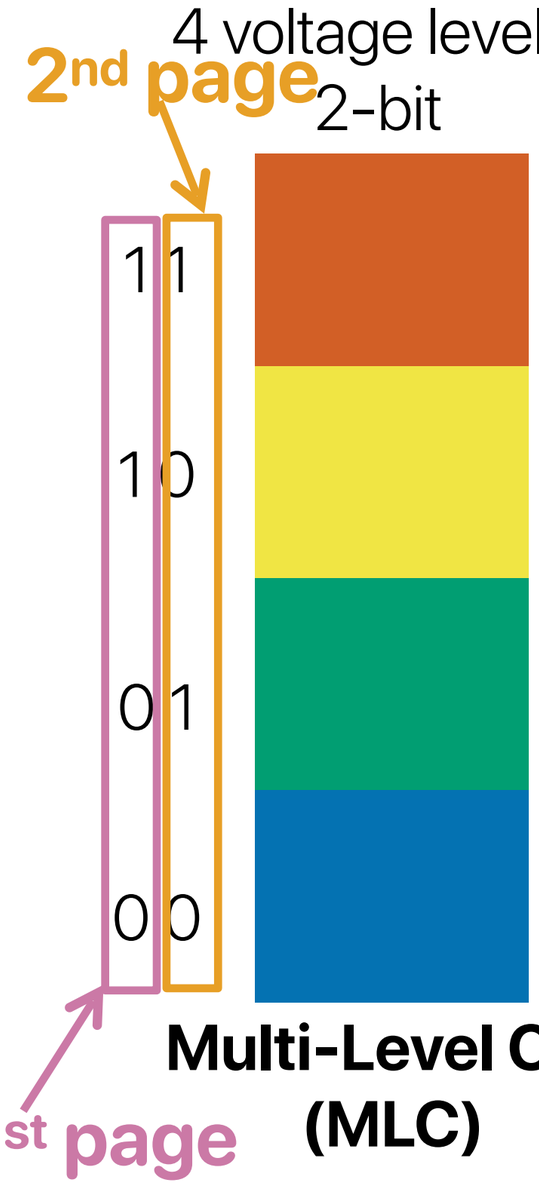
= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



1 Phase to finish programming the first page!

Programming the 2nd page in MLC

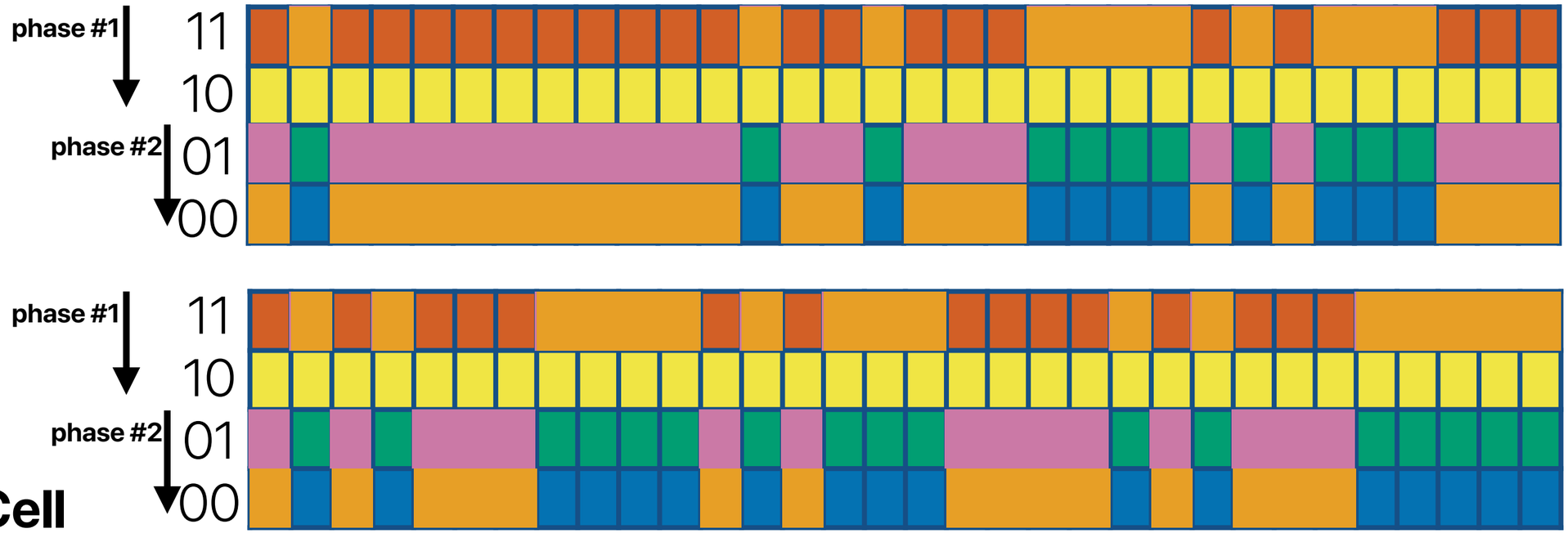


3.14000000000000012434497875800

= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111

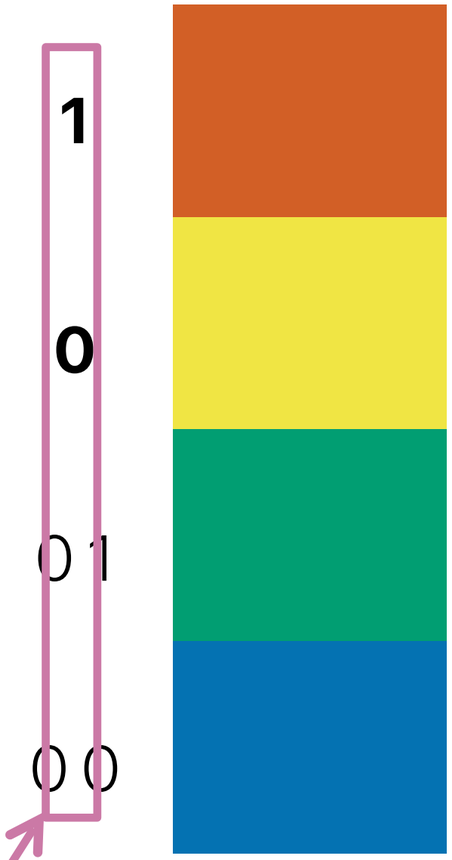
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



2 Phase to finish programming the second page!

Optimizing 1st Page Programming in MLC

4 voltage levels,
2-bit



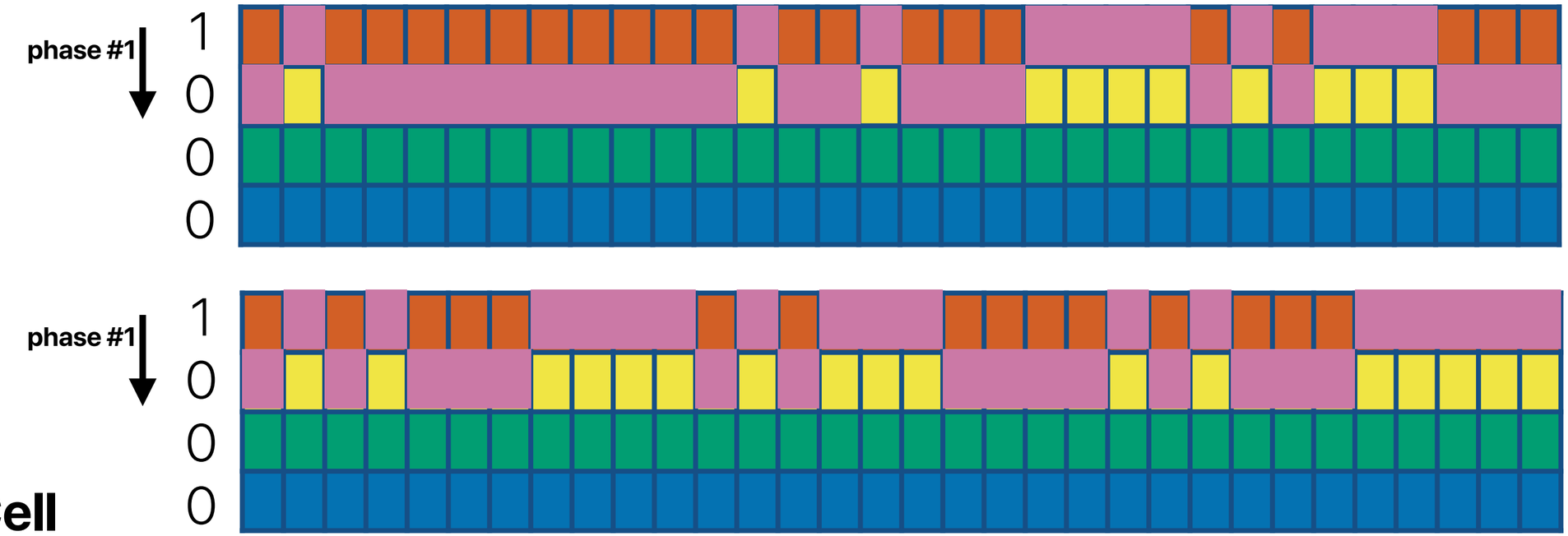
Multi-Level Cell (MLC)

1st page

3.14000000000000012434497875800

= 0x40091EB851EB851F

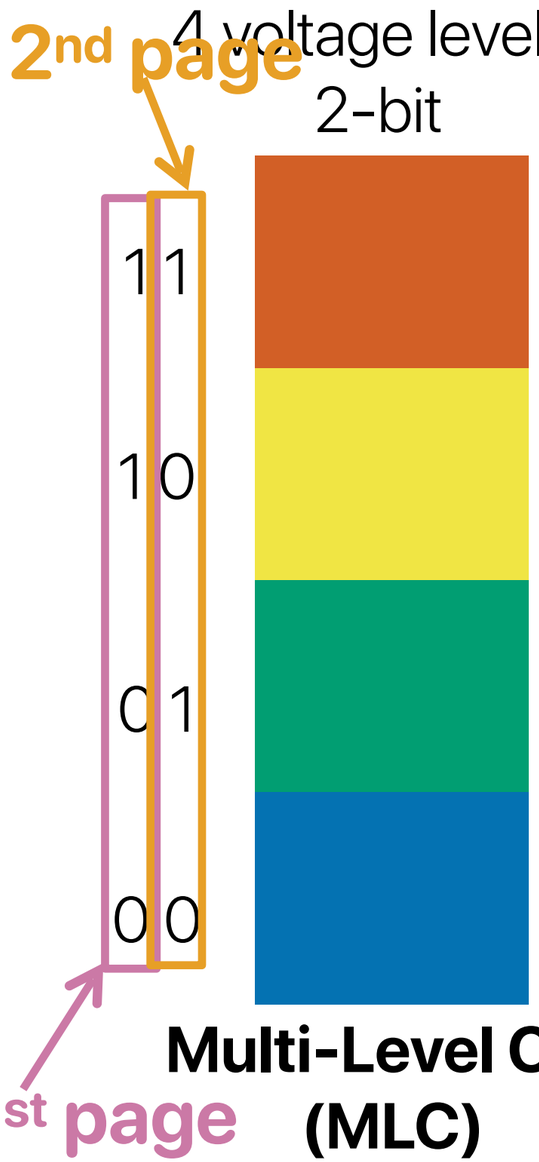
= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



1 Phase to finish programming the first page!

— the phase is shorter now

2nd Page Programming in MLC

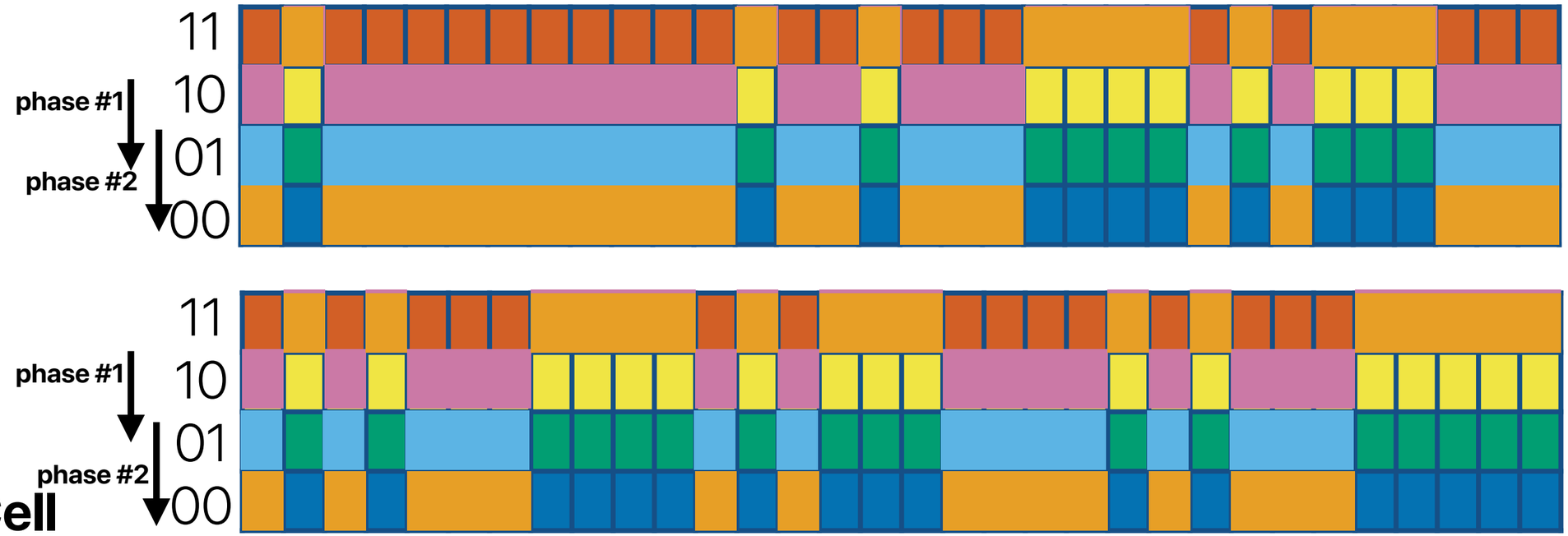


3.14000000000000012434497875800

= 0x40091EB851EB851F

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111

= 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111



2 Phase to finish programming the second page!

QLC = More Density Per NAND Cell

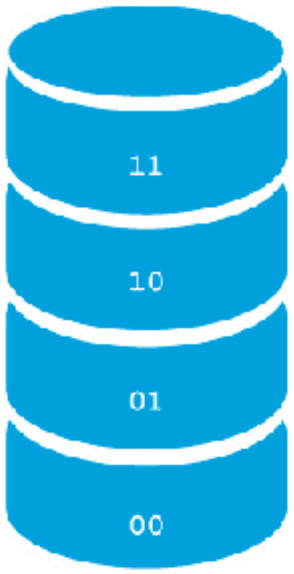
Lower $ per GB

SLC

1 Bit Per Cell

First SSD NAND technology

100K P/E Cycles
(at technology introduction)

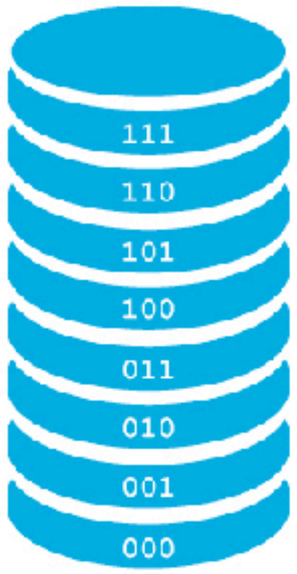


MLC

2 Bits Per Cell

100% increase

10K P/E Cycles

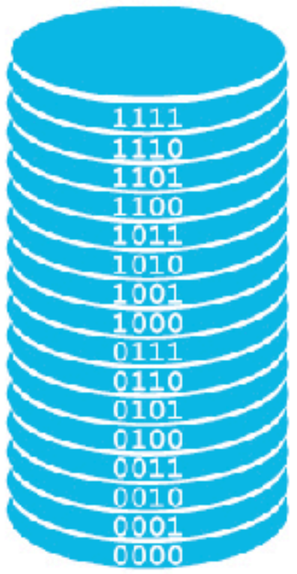


TLC

3 Bits Per Cell

50% increase

3K P/E Cycles



QLC

4 Bits Per Cell

33% increase

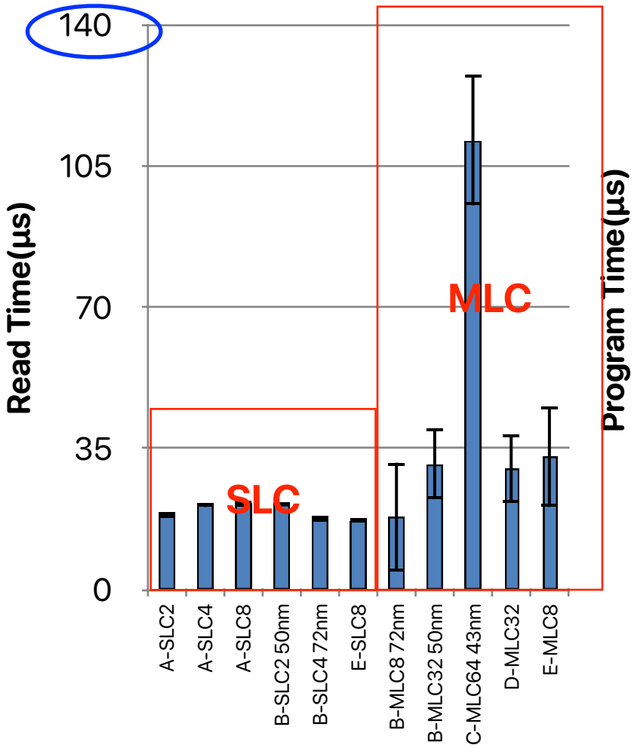
1K P/E Cycles

Fewer writes per cell

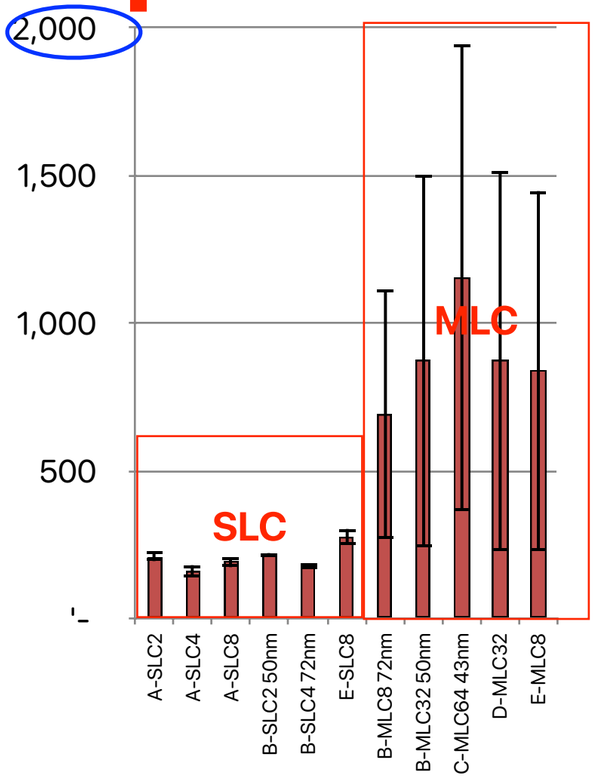
Micron

Flash performance

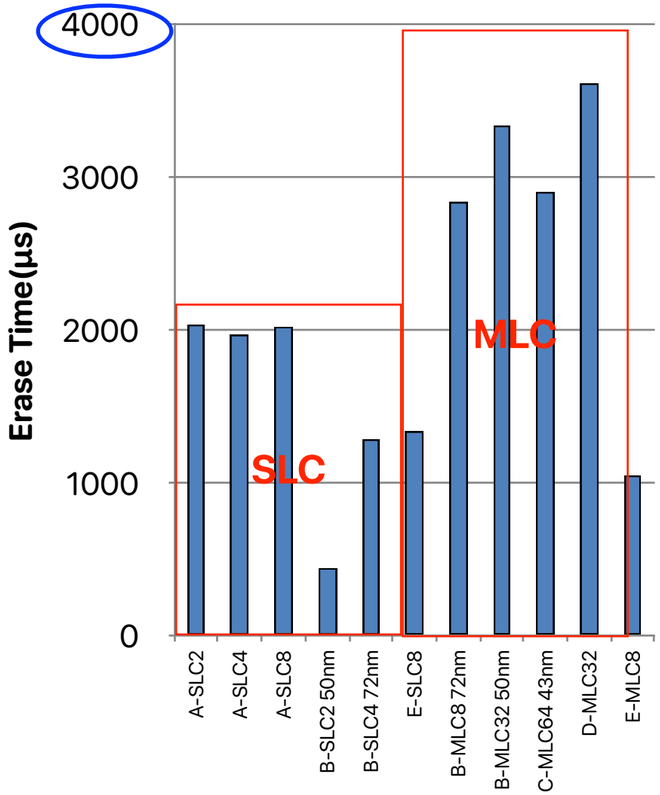
Not a good practice



Reads: less than 150us



Program/write: less than 2ms

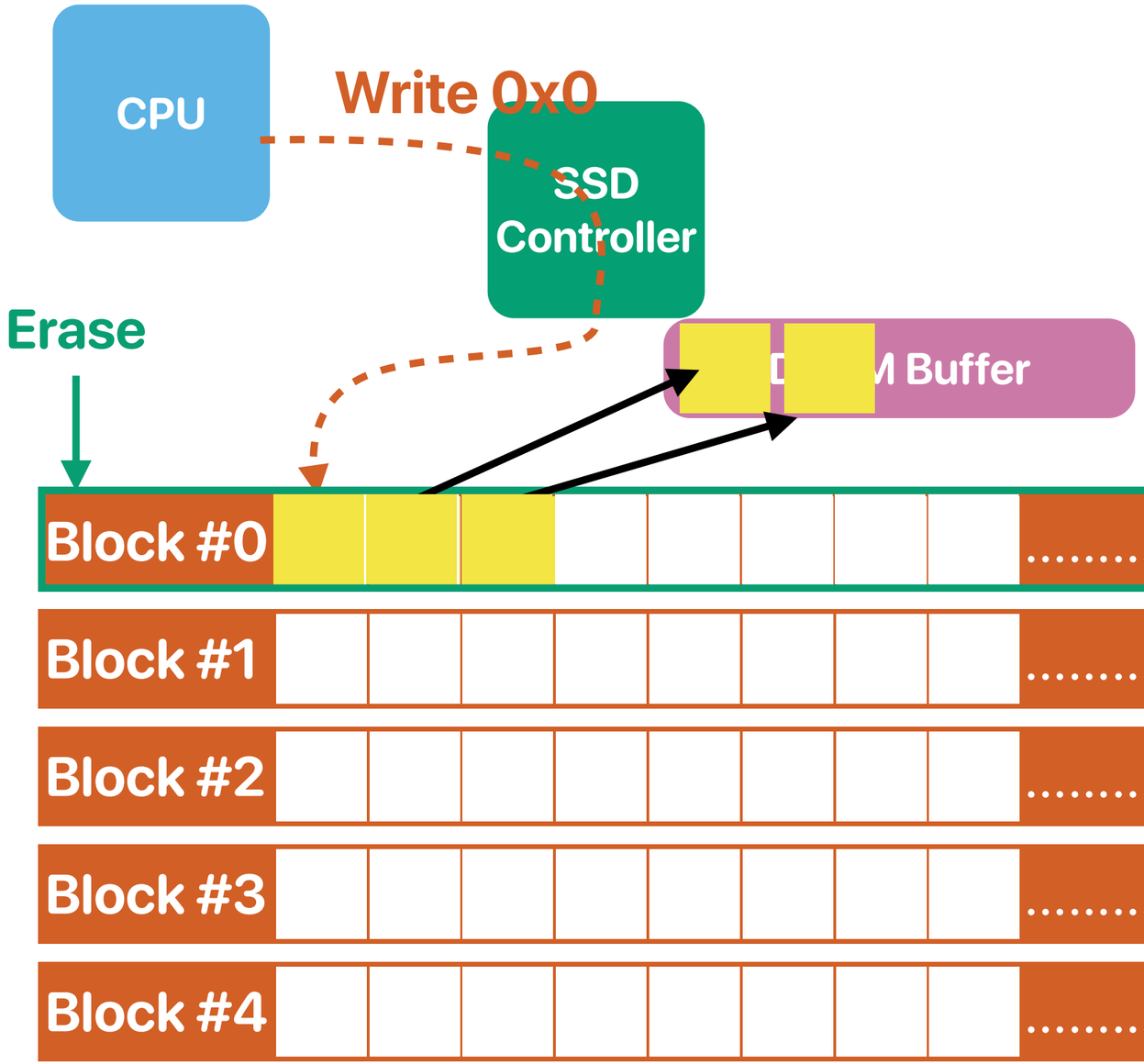


Erase: less than 3.6ms

Similar relative performance for reads, writes and erases

Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. Characterizing flash memory: anomalies, observations, and applications. In MICRO 2009.

What happens on a write if we use the same abstractions as H.D.D.



Can we write to page #0 directly? No.

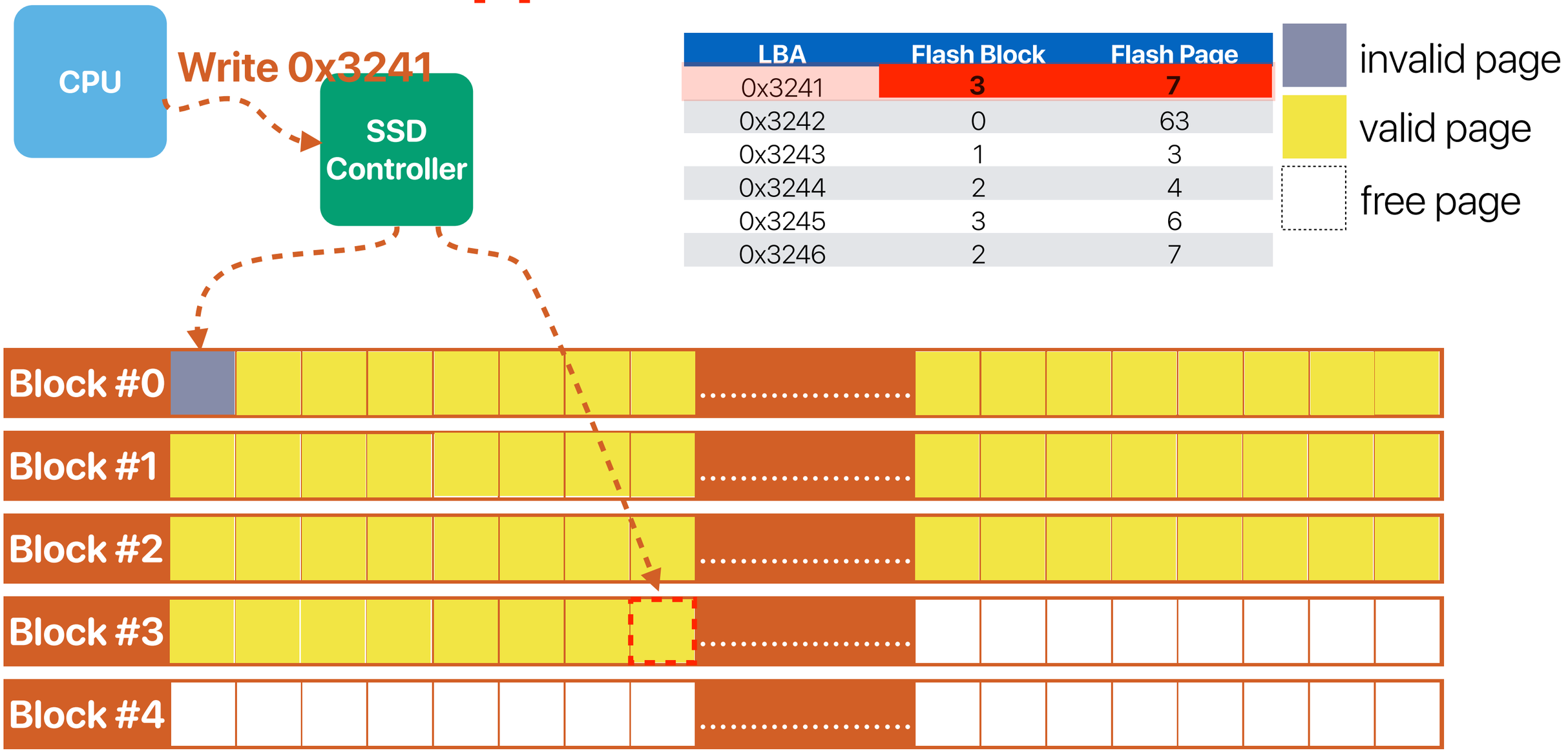
We have to copy page #1, page #2 in block #0 to somewhere (e.g. RAM buffer) and then erase the block

Write this the new 0 and the old 15 back to block #0 again!

Read: 6*30us + Writing: 2ms*3 + Erasing 3ms ~ 9 ms

Not much faster than the H.D.D. — also hurts the lifetime

What happens on a write with FTL



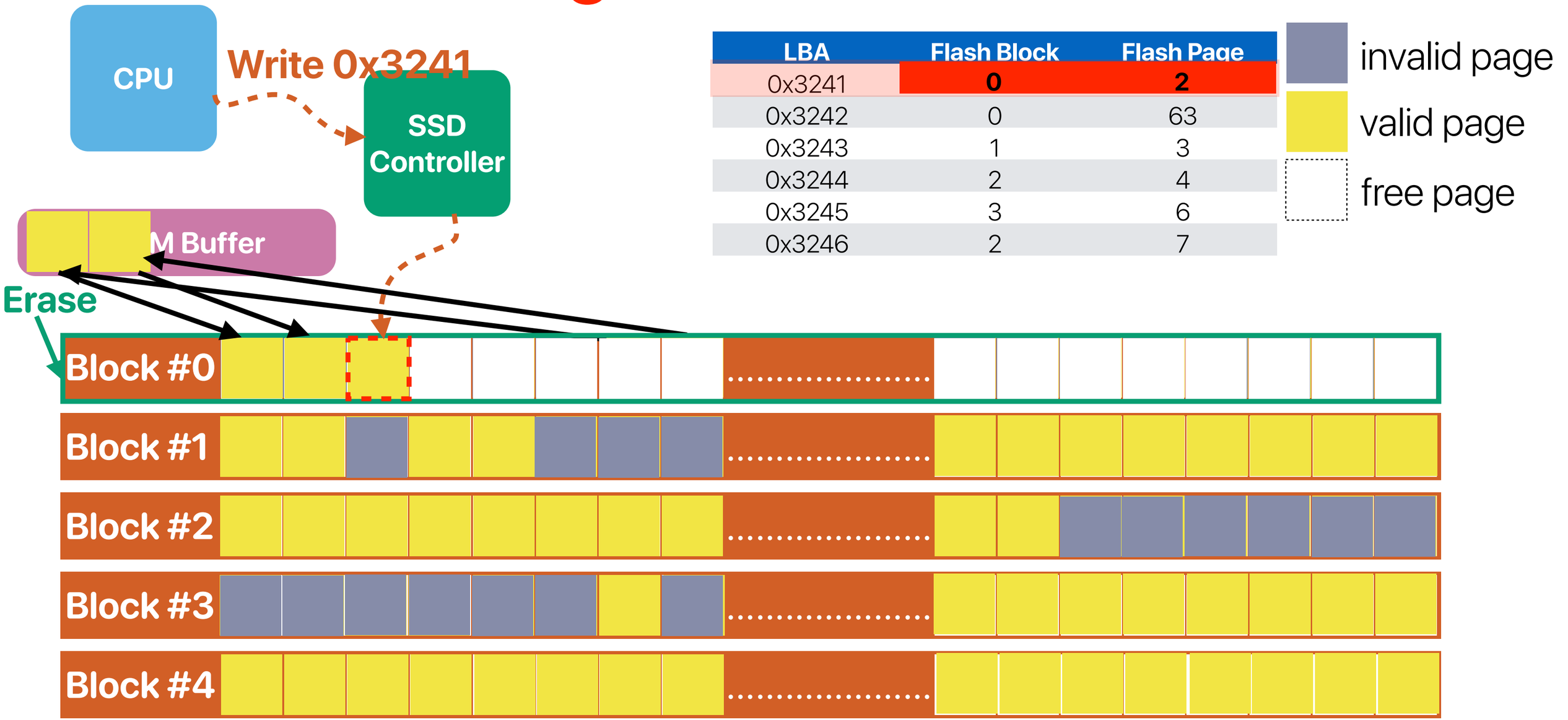
LBA	Flash Block	Flash Page
0x3241	3	7
0x3242	0	63
0x3243	1	3
0x3244	2	4
0x3245	3	6
0x3246	2	7

invalid page

valid page

free page

Garbage Collection in FTL



LBA	Flash Block	Flash Page
0x3241	**0**	**2**
0x3242	0	63
0x3243	1	3
0x3244	2	4
0x3245	3	6
0x3246	2	7

invalid page

valid page

free page

Multiple channels to improve bandwidth

Flashtec™ NVMe2032 and NVMe2016 Controllers

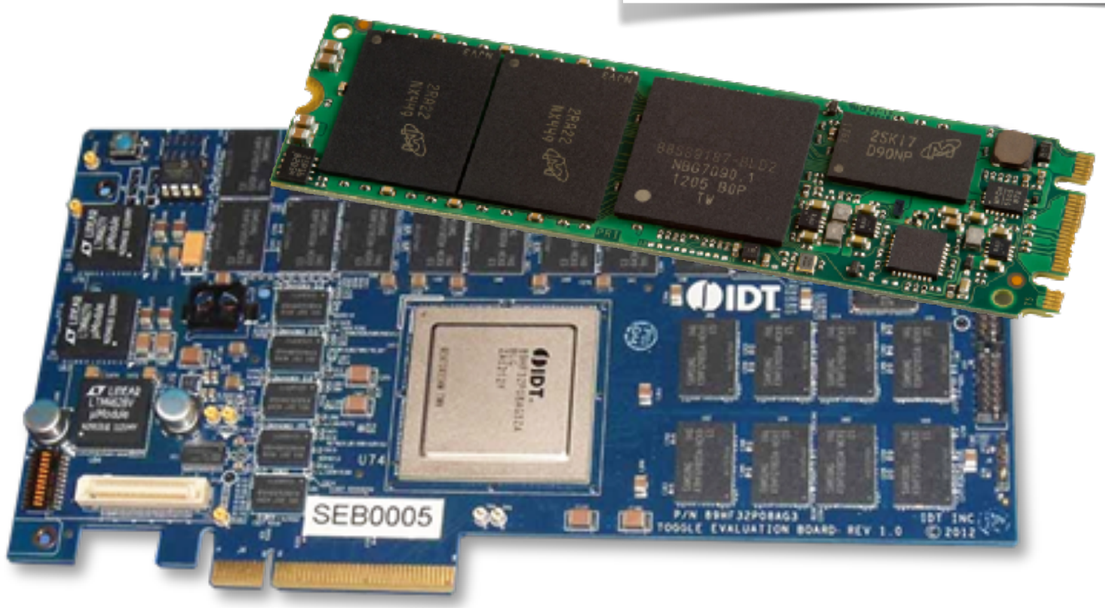
32- and 16-Channel PCIe Flash Controller Pr

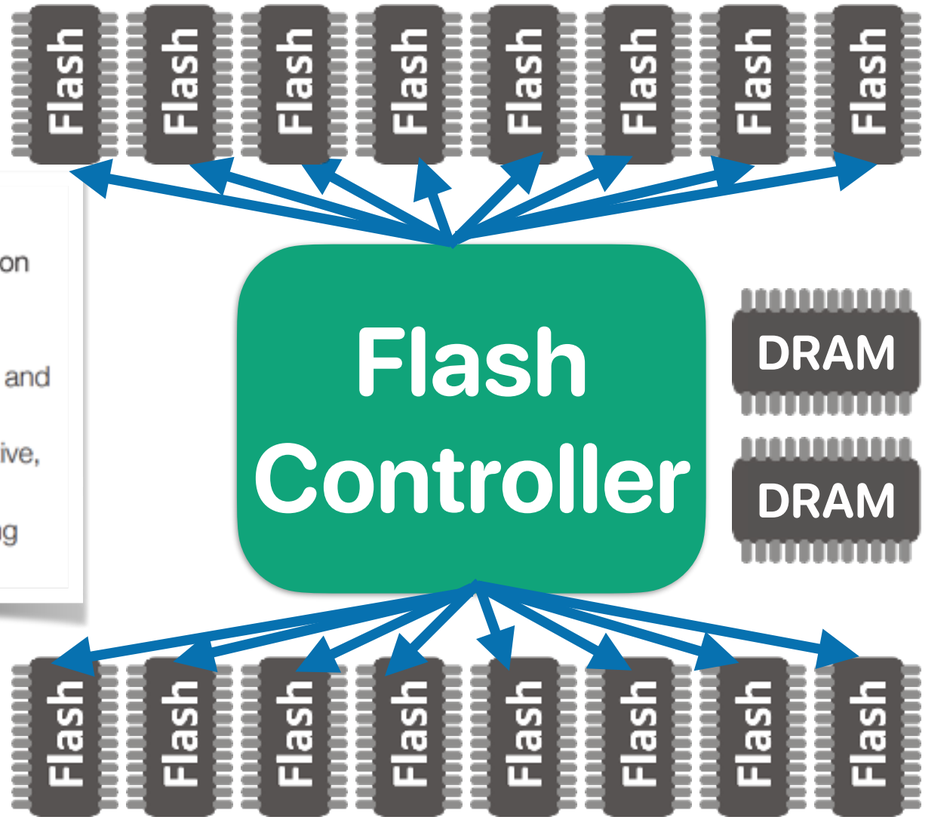
Summary

The Flashtec™ 2nd generation NVMe Controller Family enables the
and data centers to realize the highest performance SSDs utilizing n
technologies. Combining world-class capacity and flexibility, the Flas
reliable choice. The Flashtec NVMe2032 and NVMe2016 controllers
Express (NVMe) host interface and are optimized for high-performan
operations, performing all flash management operations on-chip an
processing and memory resources.

Features

- Flashtec NVMe2032 controller can achieve up to 1 million random read IOPS on 4 KB operations
- Up to 20 TB Flash capacity using 256 GB Flash
- SLC, MLC, Enterprise MLC, and TLC Flash with toggle and ONFI interface
- PCIe Gen 3 x8 or dual independent PCIe Gen 3 x4 (active, active/standby) host interface
- 16 and 32 independent Flash channels, each supporting up to 8 CE





Electrical
Computer Science
Engineering

277

つづく