KING'S College LONDON

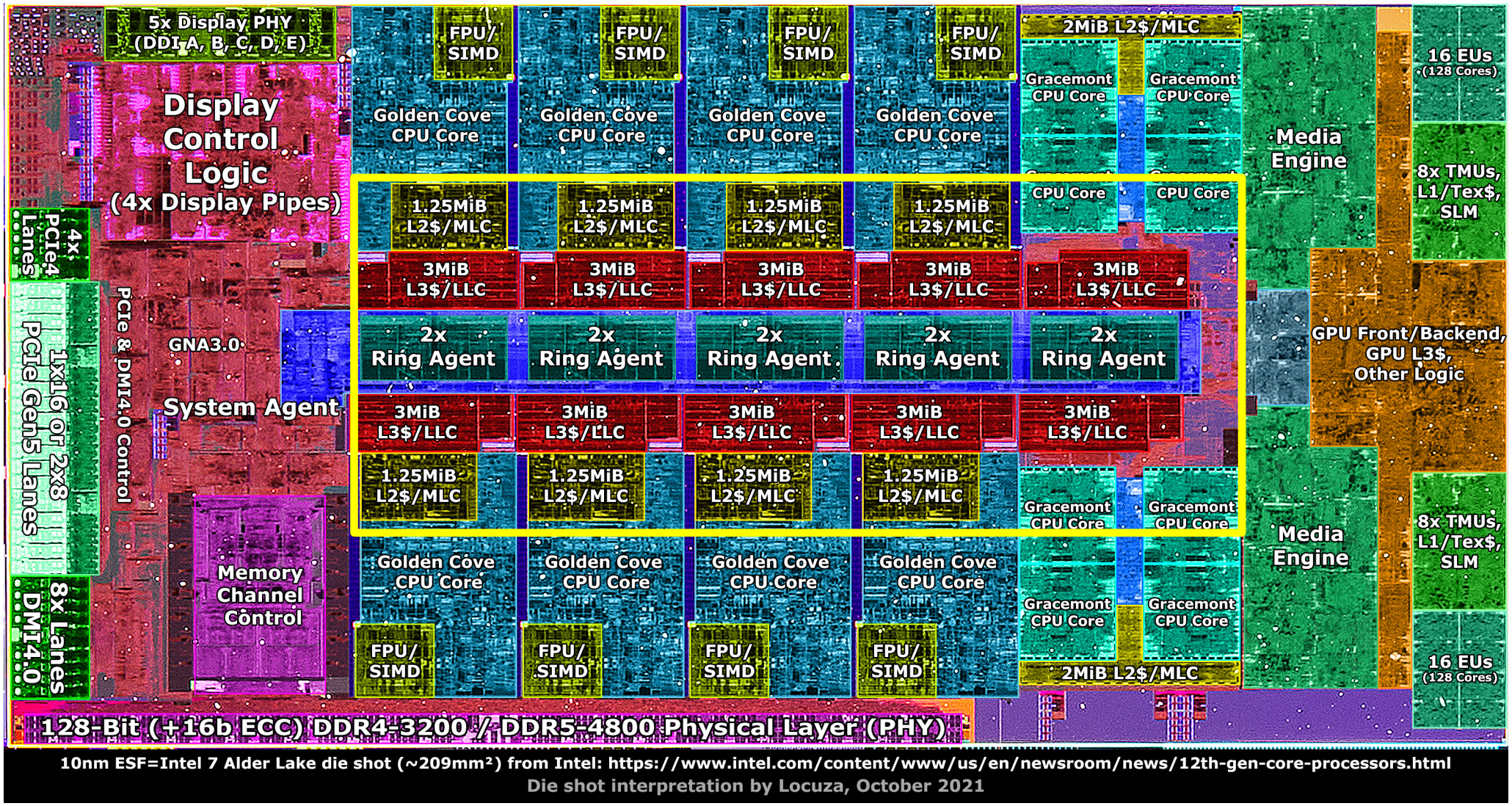
Hardware Design

Lecture 9: Cache (II) and Storage

Dr. Haiyu Mao

26.03.2026

Recall

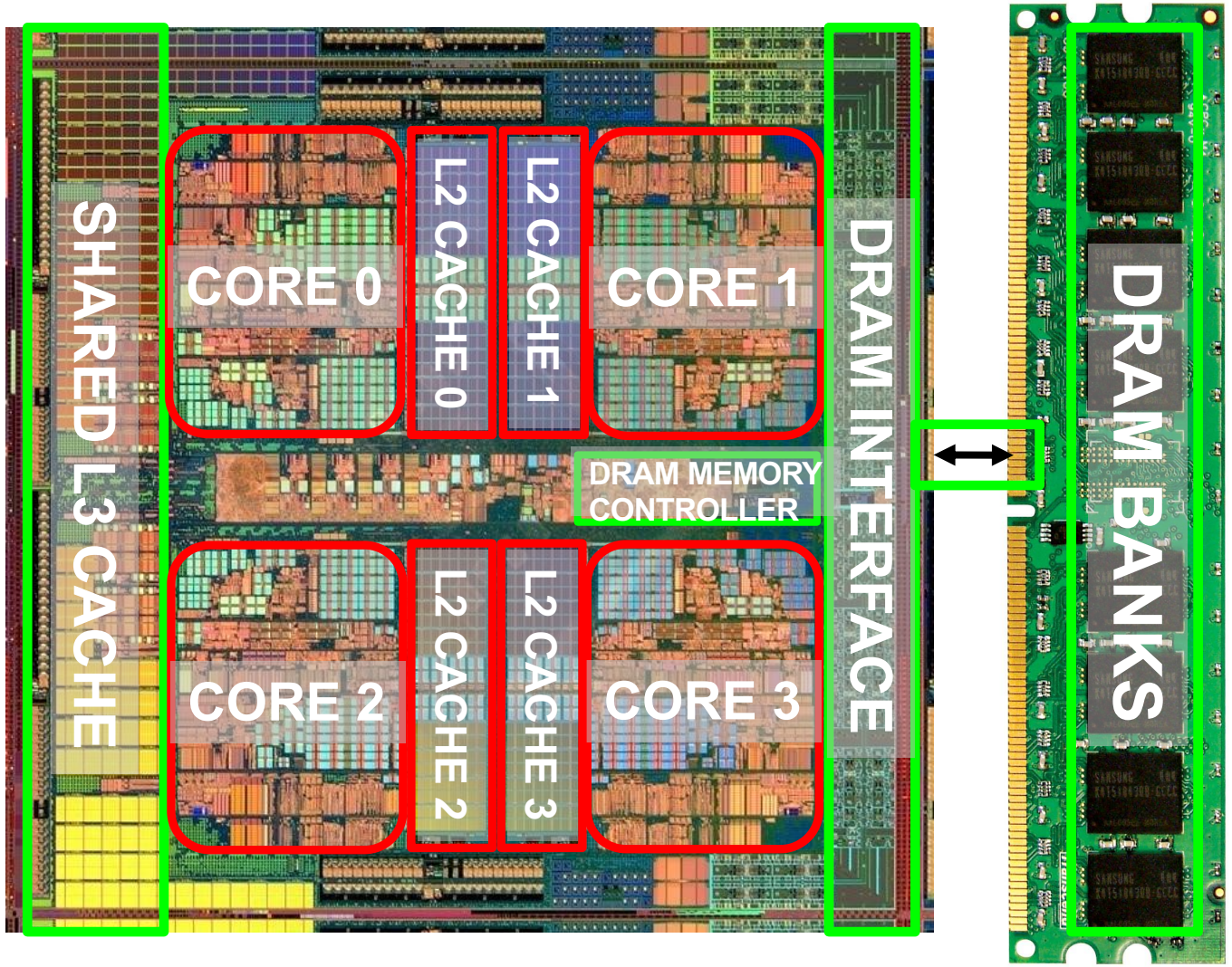


10nm ESF=Intel 7 Alder Lake die shot (~209mm²) from Intel: https://www.intel.com/content/www/us/en/newsroom/news/12th-gen-core-processors.html
Die shot interpretation by Locuza, October 2021

Intel Alder Lake, 2021

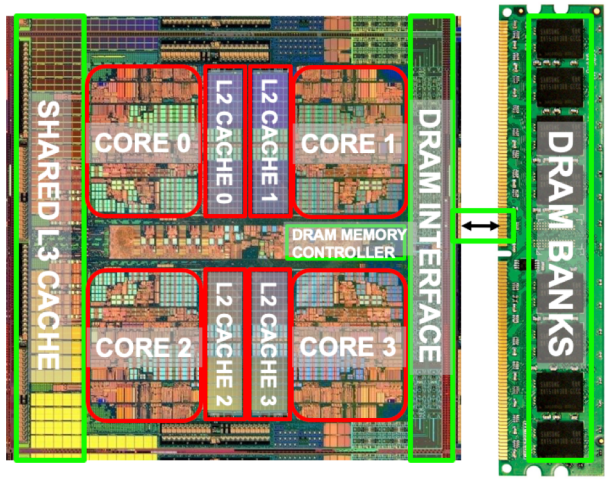
Source: https://twitter.com/Locuza_/status/1454152714930331652

Cache in a Multi-Core System



Hardware Design

Cache in a Multi-Core System



Caches in Multi-Core Systems

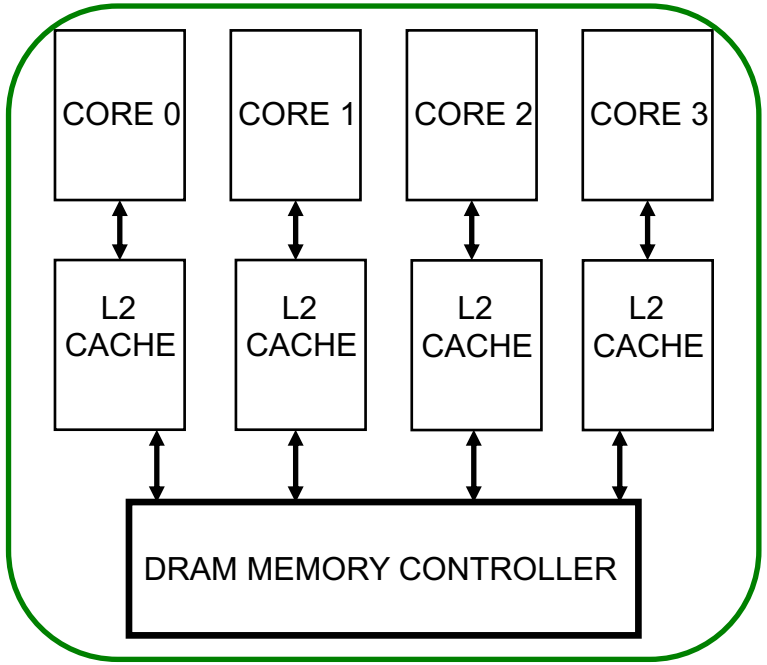
- Cache efficiency becomes even more important in a multi-core/multi-threaded system
  - Memory bandwidth is at a premium
  - Cache space is a limited resource across cores/threads

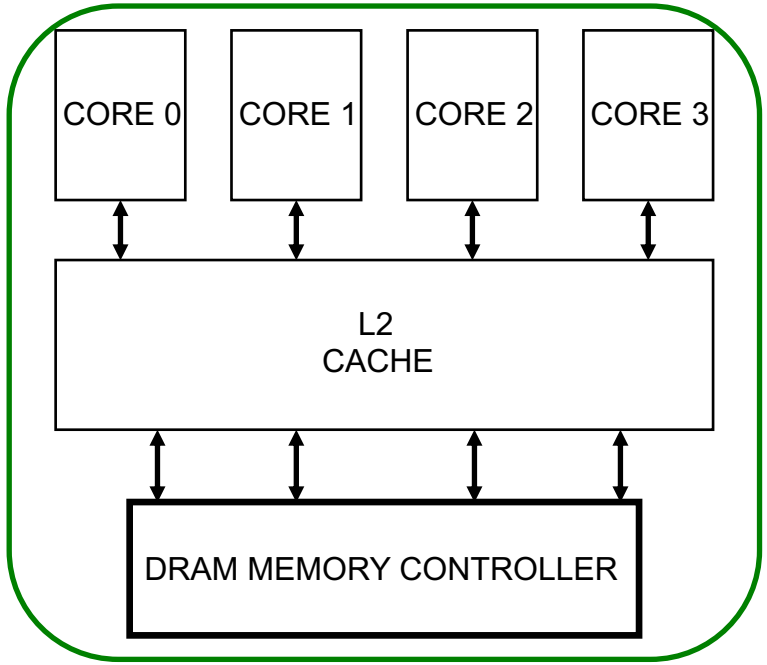
- How do we design the caches in a multi-core system?

- Many decisions and questions
  - Shared vs. private caches
  - How to maximize the performance of the entire system?
  - How to provide QoS & predictable perf. to different threads in a shared cache?
  - Should cache management algorithms be aware of threads?
  - How should space be allocated to threads in a shared cache?
  - Should we store data in compressed format in some caches?
  - How do we do better reuse prediction & management in caches?

Private vs. Shared Caches

- Private cache: Cache belongs to one core (a shared block can be in multiple caches)
- Shared cache: Cache is shared by multiple cores





Resource Sharing Concept and Advantages

- Idea: Instead of dedicating a hardware resource to a hardware context, allow multiple contexts to use it
  - Example resources: functional units, pipeline, caches, buses, memory, interconnects, storage
- Why?

+ Resource sharing improves utilization/efficiency → throughput
  - When a resource is left idle by one thread, another thread can use it; no need to replicate shared data
+ Reduces communication latency
  - For example, data shared between multiple threads can be kept in the same cache in multithreaded processors
+ Compatible with the shared memory programming model

Resource Sharing Disadvantages

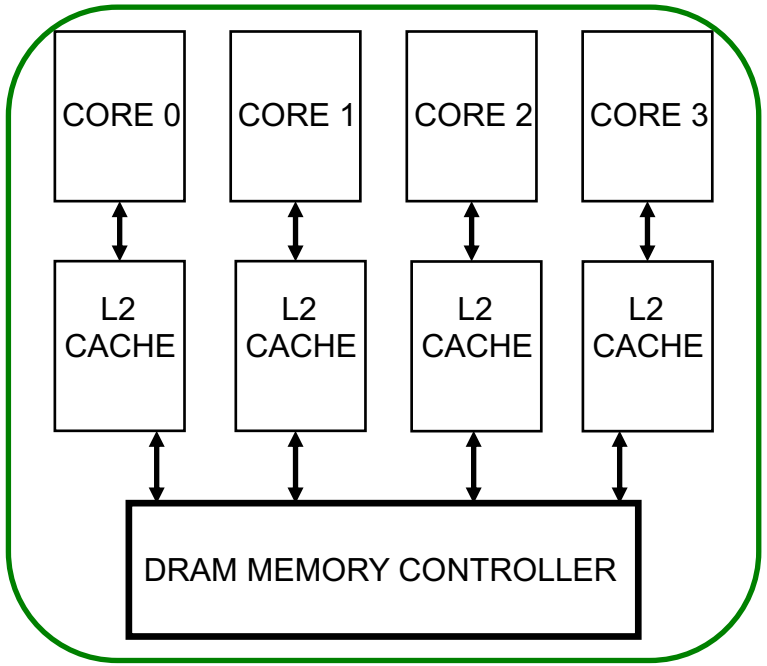
- Resource sharing results in contention for resources
  - When the resource is not idle, another thread cannot use it
  - If space is occupied by one thread, another thread needs to reoccupy it

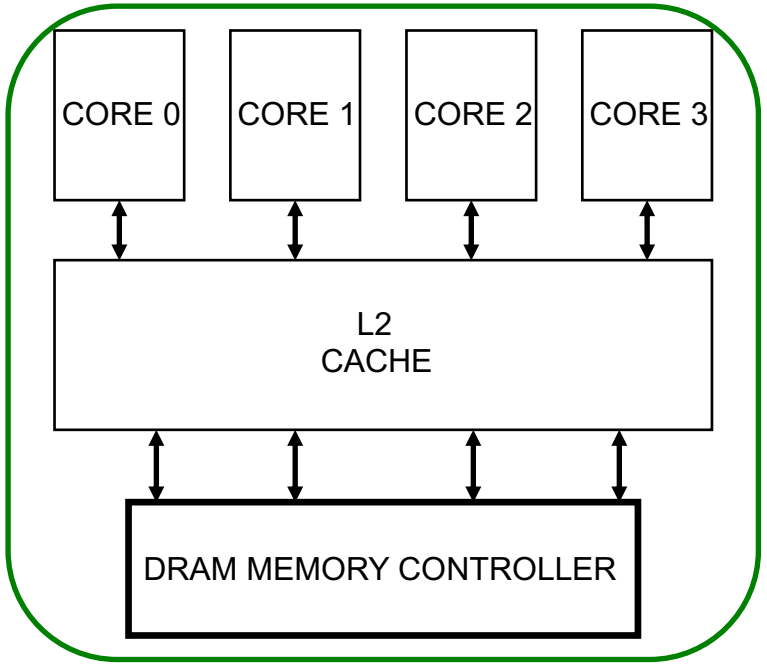
- Sometimes reduces each or some thread's performance
  - Thread performance can be worse than when it is run alone
- Eliminates performance isolation → inconsistent performance across runs
  - Thread performance depends on co-executing threads
- Uncontrolled (free-for-all) sharing degrades the quality of service
  - Causes unfairness, starvation

Need to efficiently and fairly utilize shared resources

Private vs. Shared Caches

- Private cache: Cache belongs to one core (a shared block can be in multiple caches)
- Shared cache: Cache is shared by multiple cores



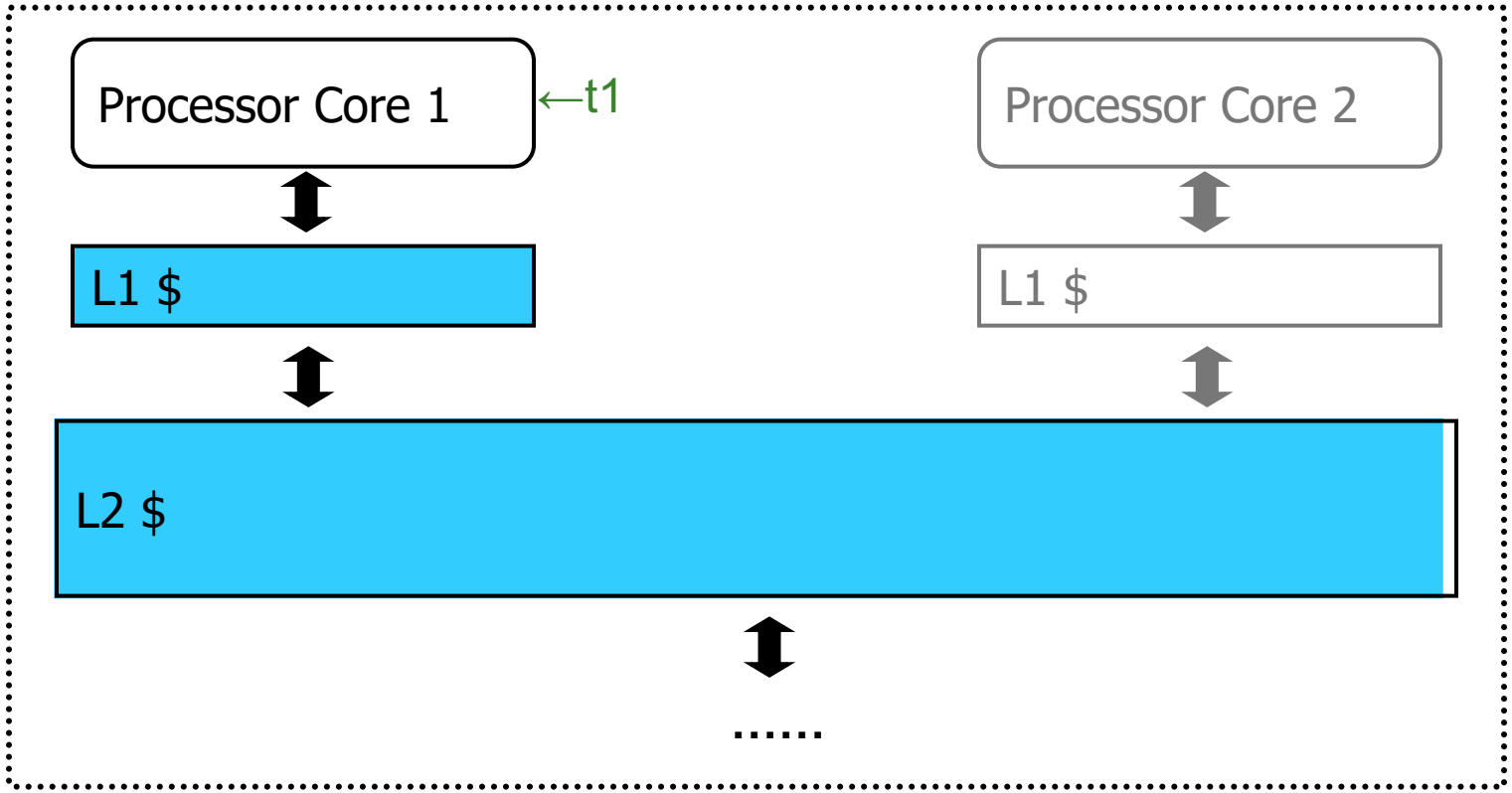


Shared Caches Between Cores

- Advantages:
  - High effective capacity
  - Dynamic partitioning of available cache space
    - No fragmentation due to static partitioning
    - If one core does not utilize some space, another core can
  - Easier to maintain coherence (a cache block is in a single location)

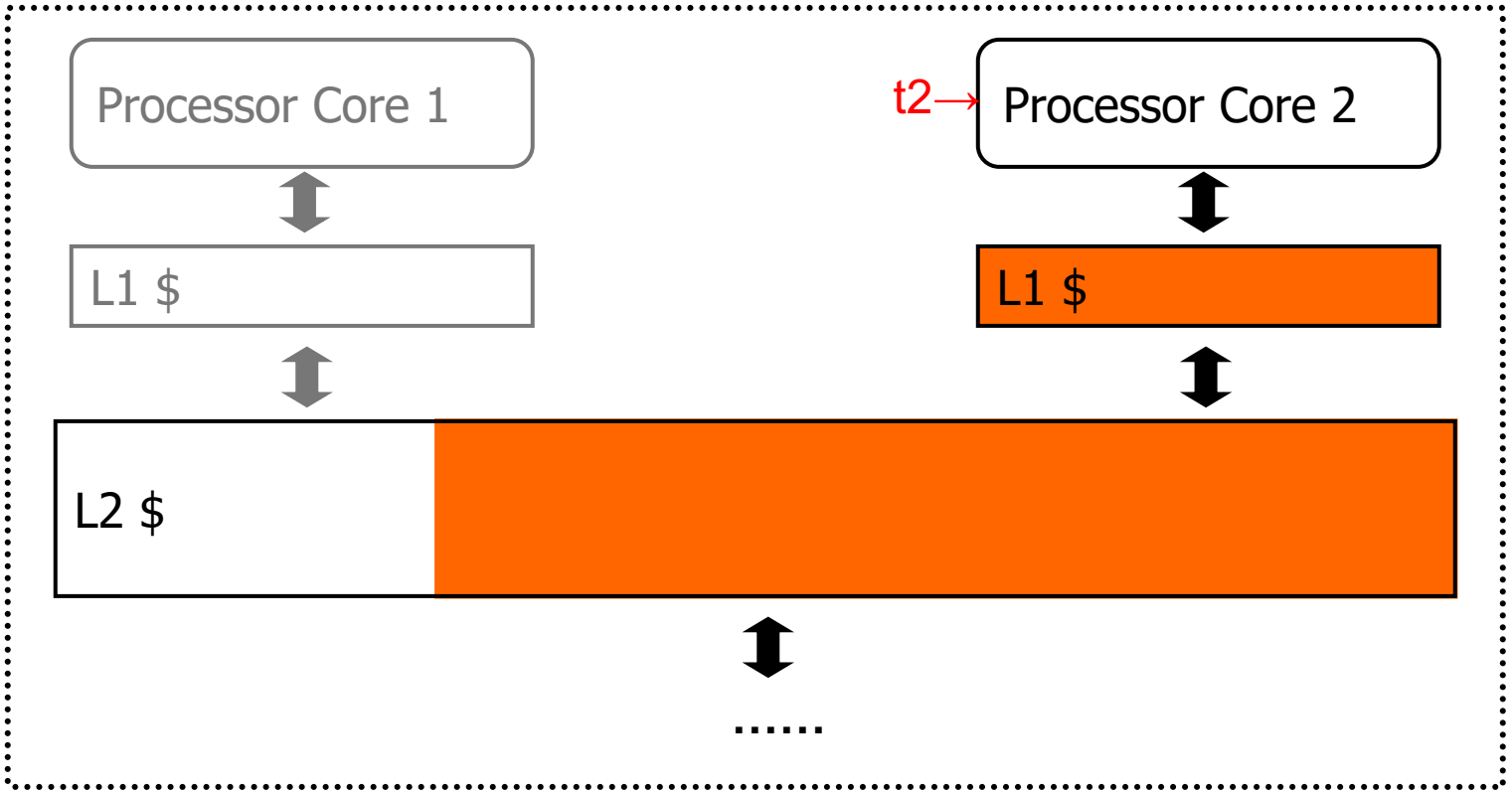
- Disadvantages
  - Slower access (cache is not tightly coupled with the core)
  - Cores incur conflict misses due to other cores' accesses
    - Misses due to inter-core interference
    - Some cores can destroy the hit rate of other cores
  - Guaranteeing a minimum level of service (or fairness) to each core is harder (how much space, how much bandwidth?)

Example: One Problem with Shared Caches

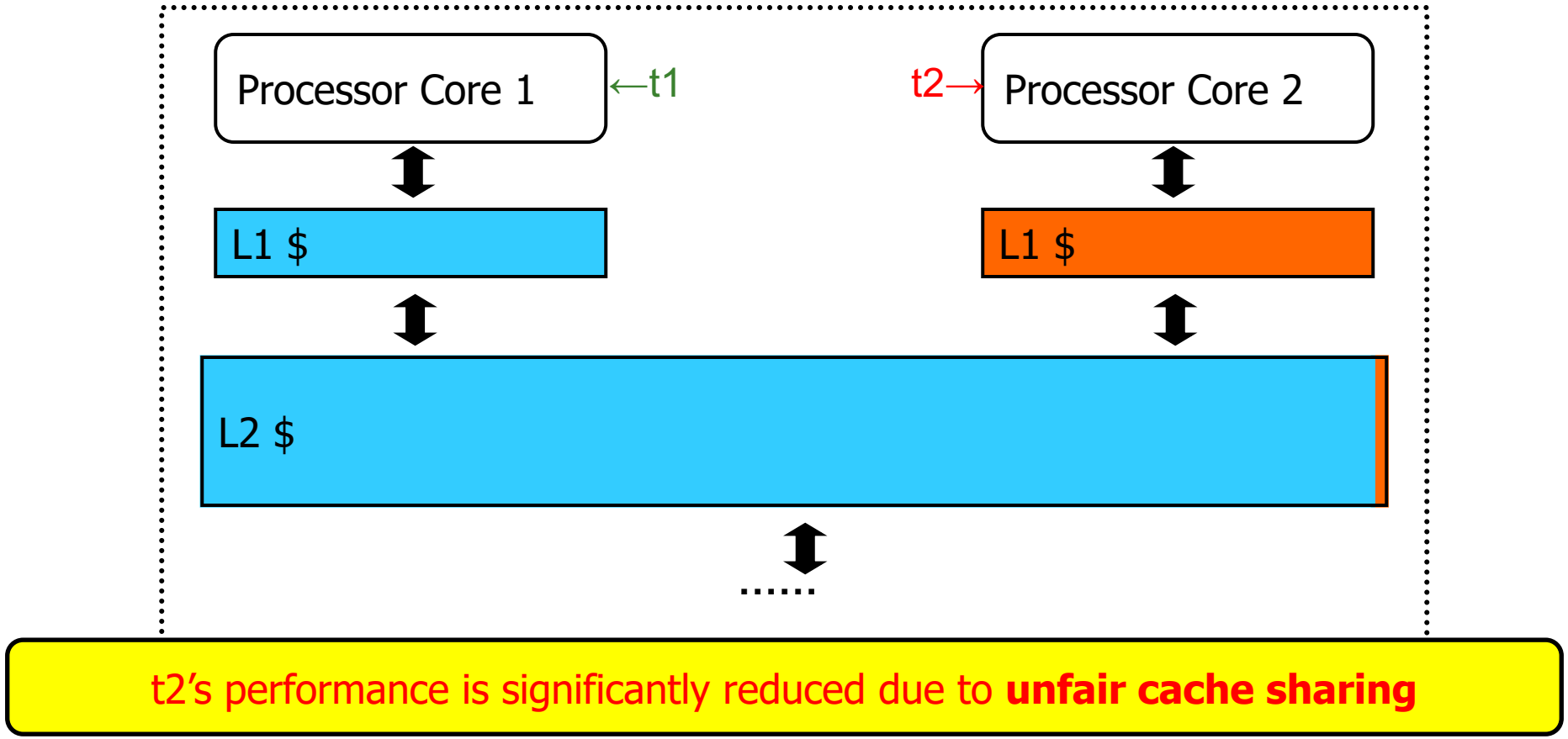


Kim et al., "Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture," PACT 2004.

Example: One Problem with Shared Caches



Example: One Problem with Shared Caches

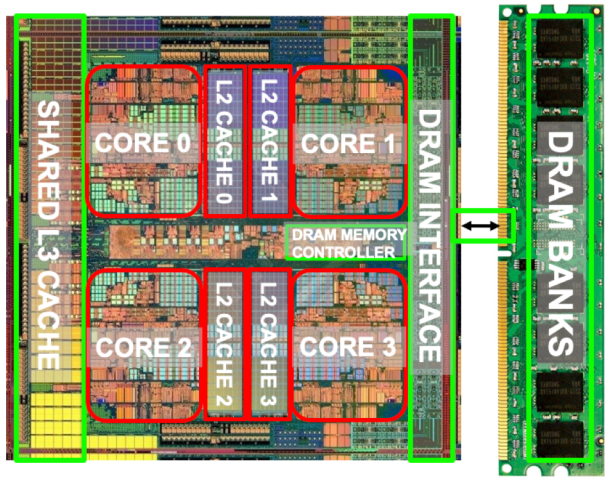


Resource Sharing vs. Partitioning

- Sharing improves resource utilization
  - Better utilization of space
- Partitioning provides performance isolation (predictable performance)
  - Dedicated space
- Can we get the benefits of both?
- Idea: Design shared resources such that they are efficiently utilized, controllable and partitionable
  - No wasted resource + QoS mechanisms for threads

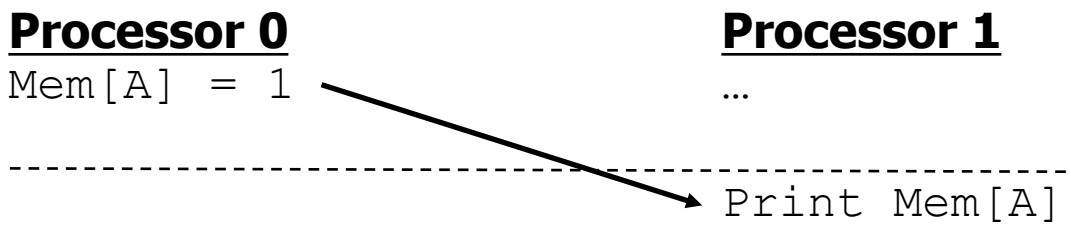
Hardware Design

Cache Coherence



Shared Memory Model

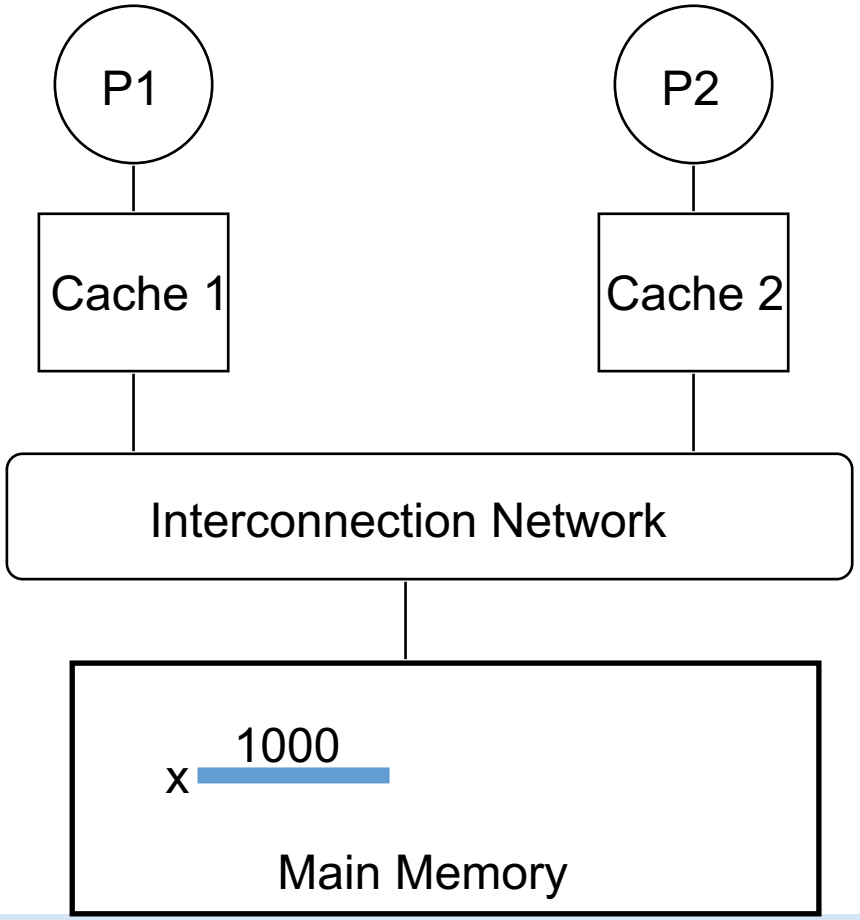
- Threads in parallel programs communicate through *shared memory*
- Thread 0 writes to an address, followed by Thread 1 reading
  - This implies communication between the two



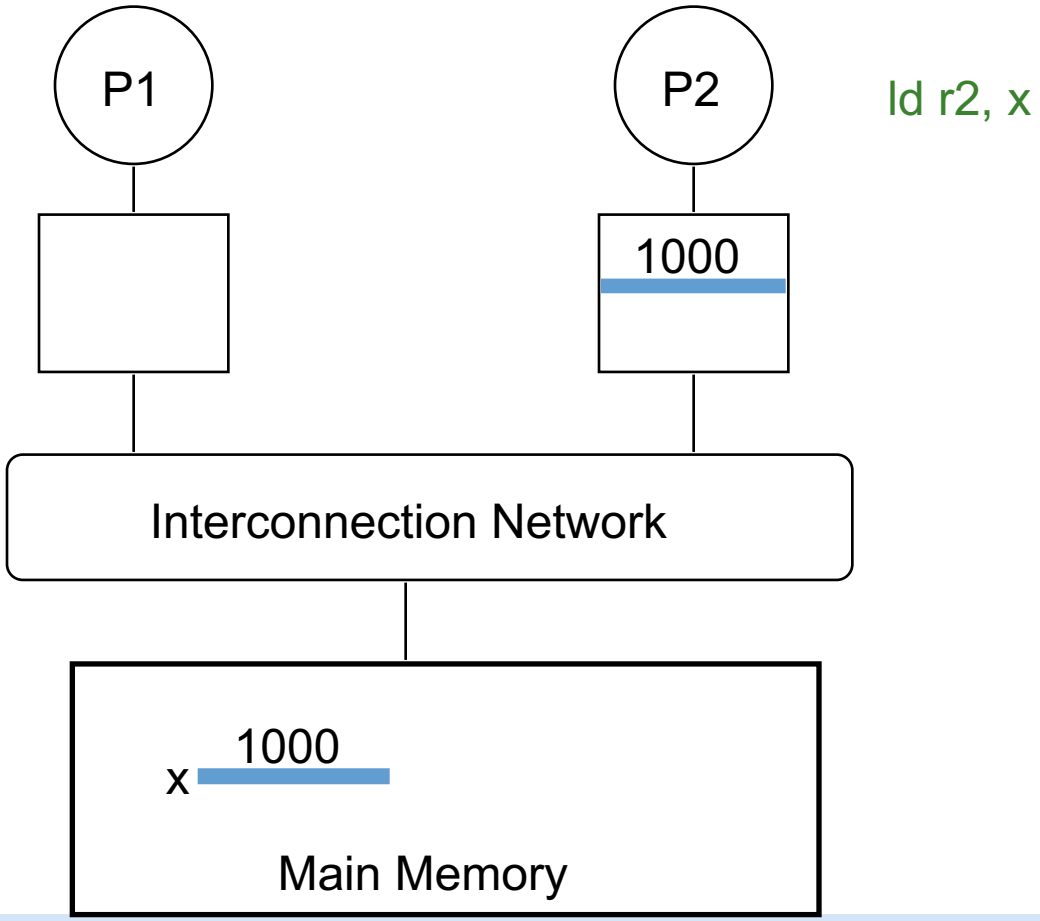
- Each read should receive the value last written by any processor
  - This requires synchronization between threads (i.e., what does last written mean?)
- What if Mem[A] is cached (at either processor)?

Cache Coherence

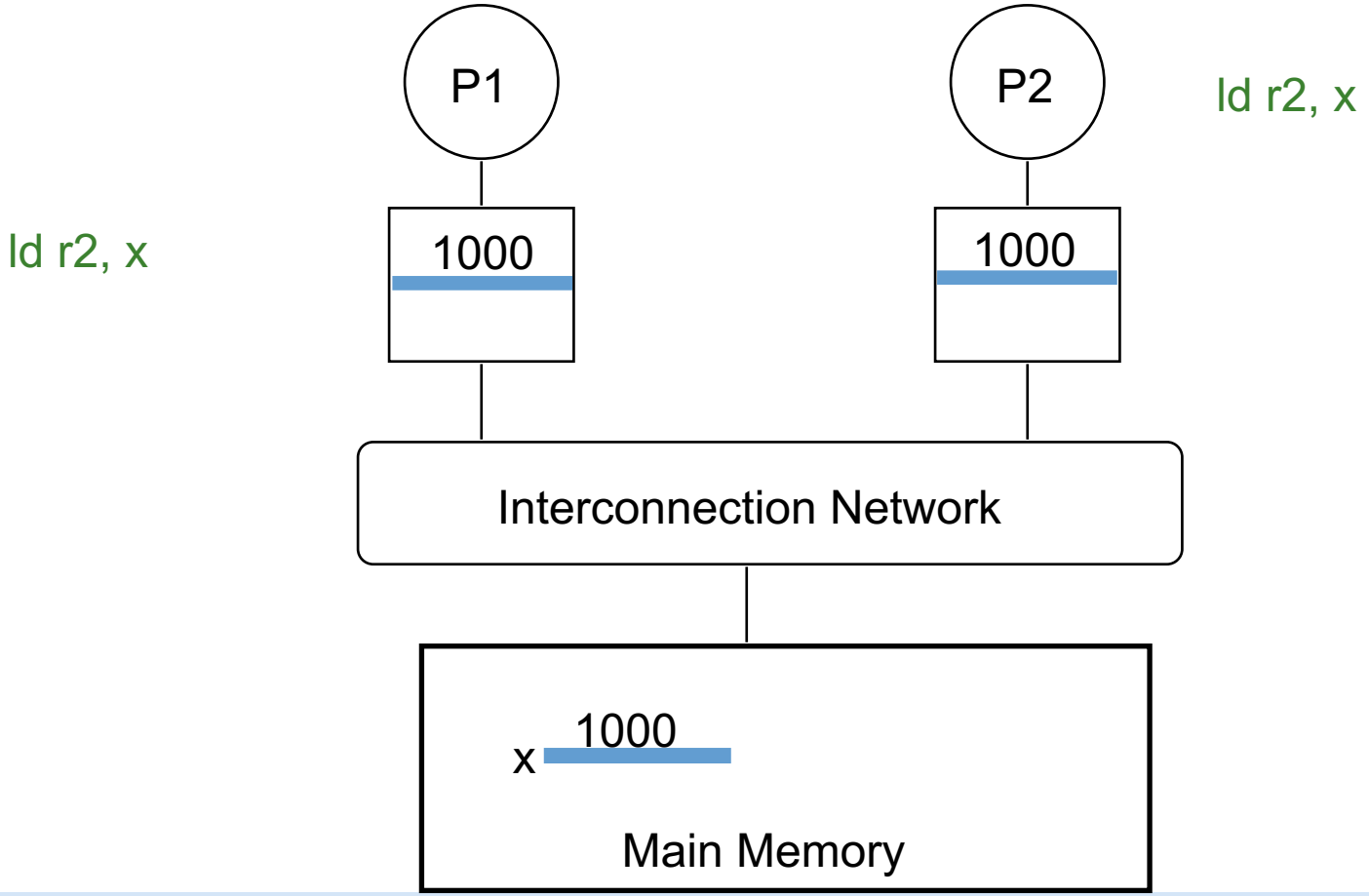
- Basic question: If multiple processors cache the same block, how do they ensure they all see a consistent state?



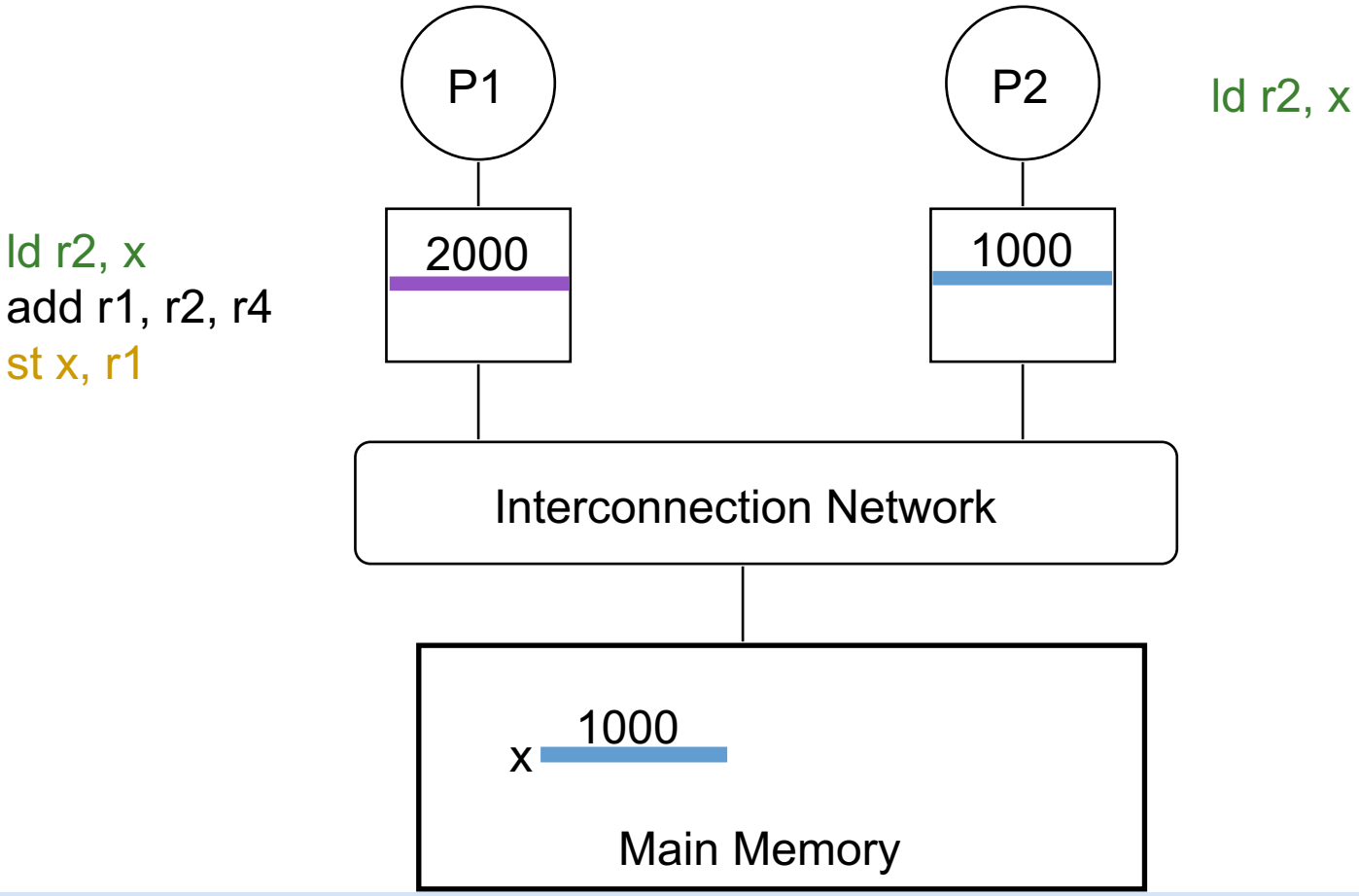
The Cache Coherence Problem



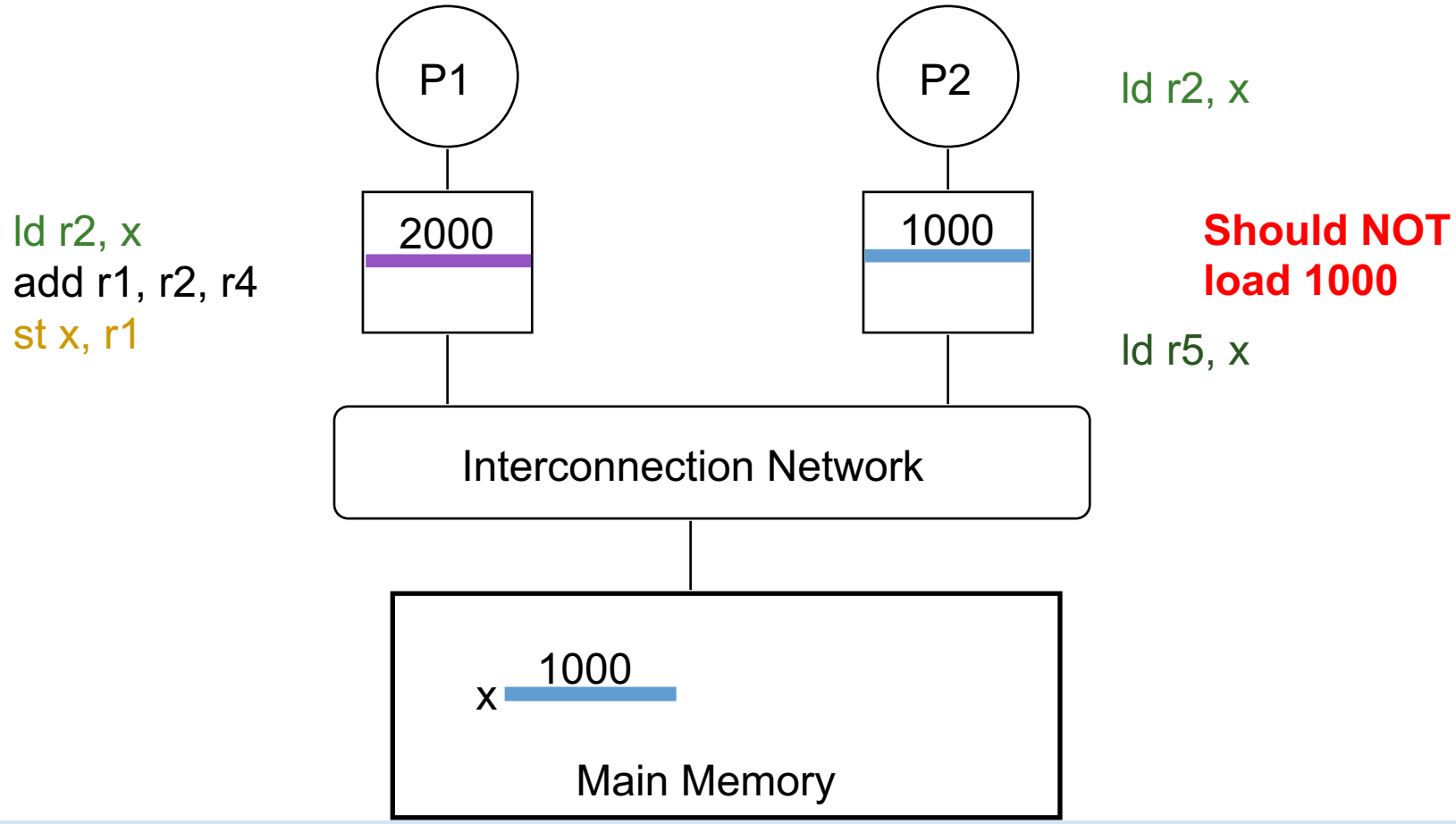
The Cache Coherence Problem



The Cache Coherence Problem

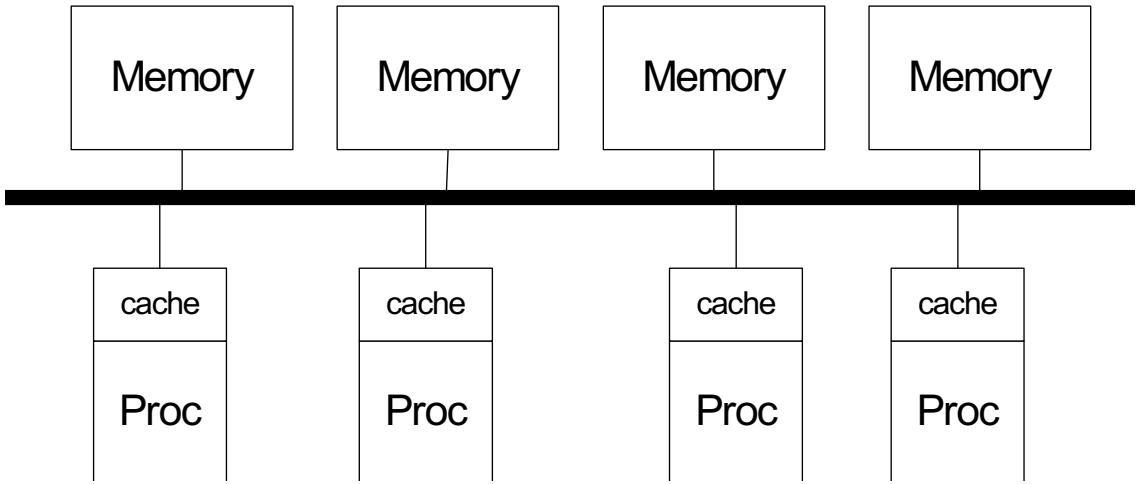


The Cache Coherence Problem



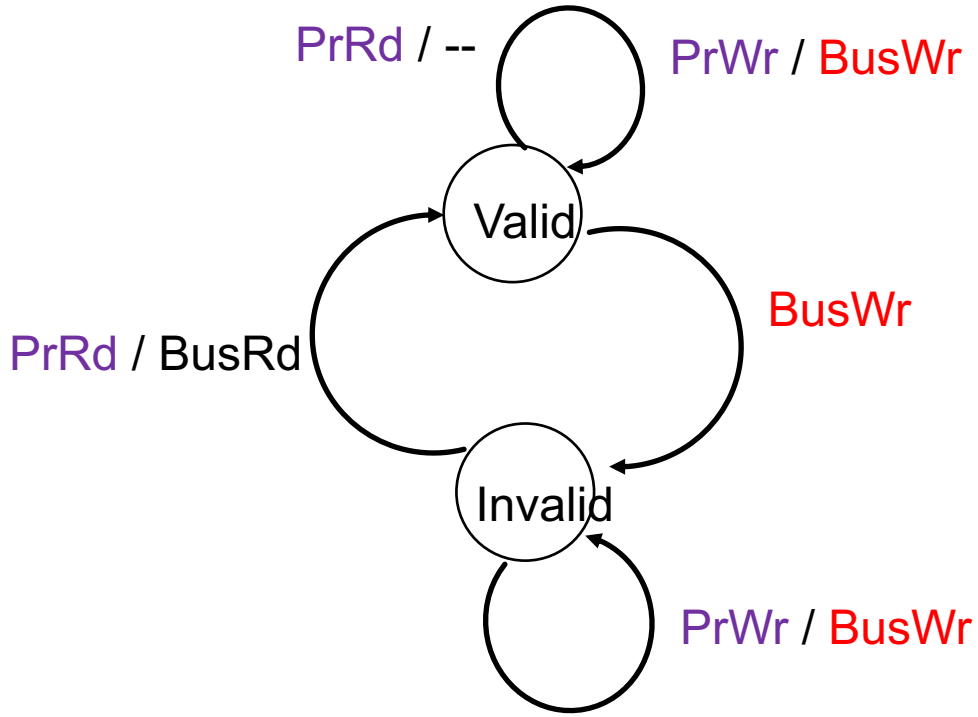
Broadcast-Based Hardware Cache Coherence

- Basic idea:
  - A processor/cache broadcasts its write/update to a cache block to all other processors
  - Another processor/cache that has the block either invalidates or updates its local copy of the block



A Very Simple Coherence Scheme

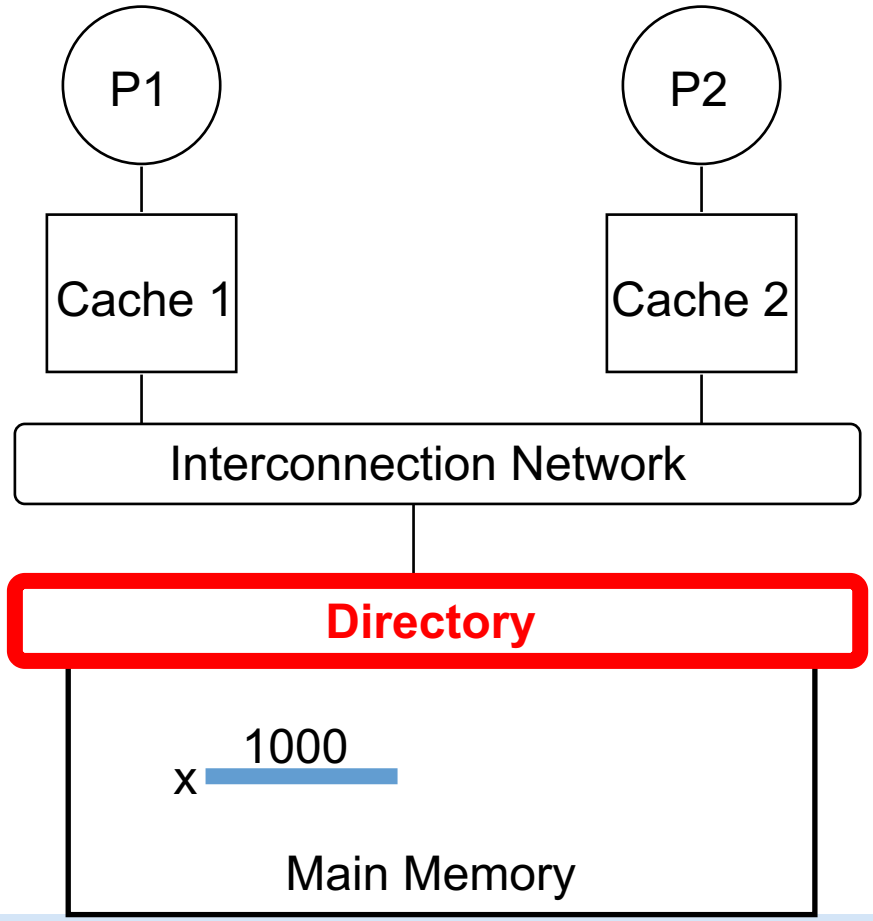
- Idea: All caches "snoop" (observe) each other's write/read operations. If a processor writes to a block, all others invalidate the block.
- A simple protocol:



- Assumption: Write-through, no-write-allocate cache
- Actions of the local processor on the cache block: PrRd, PrWr
- Actions that are broadcast on the bus for the block: BusRd, BusWr

Directory-Based Cache Coherence

- Idea: A central directory keeps track of which caches contain every possible cache block. Every cache consults this directory to ensure data is kept coherent.



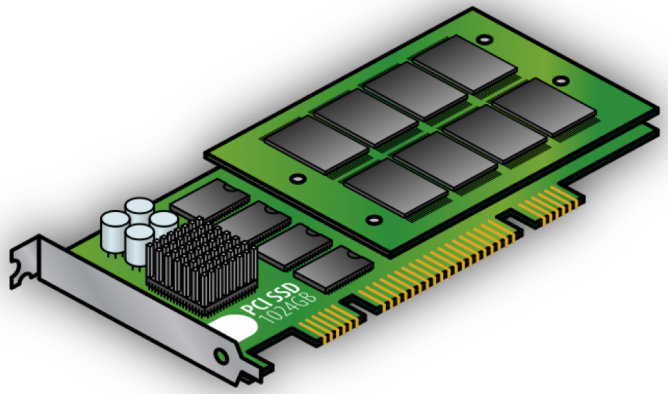
Directory-Based Cache Coherence

- Idea: A central directory keeps track of which caches contain every possible cache block. Every cache consults this directory to ensure data is kept coherent.

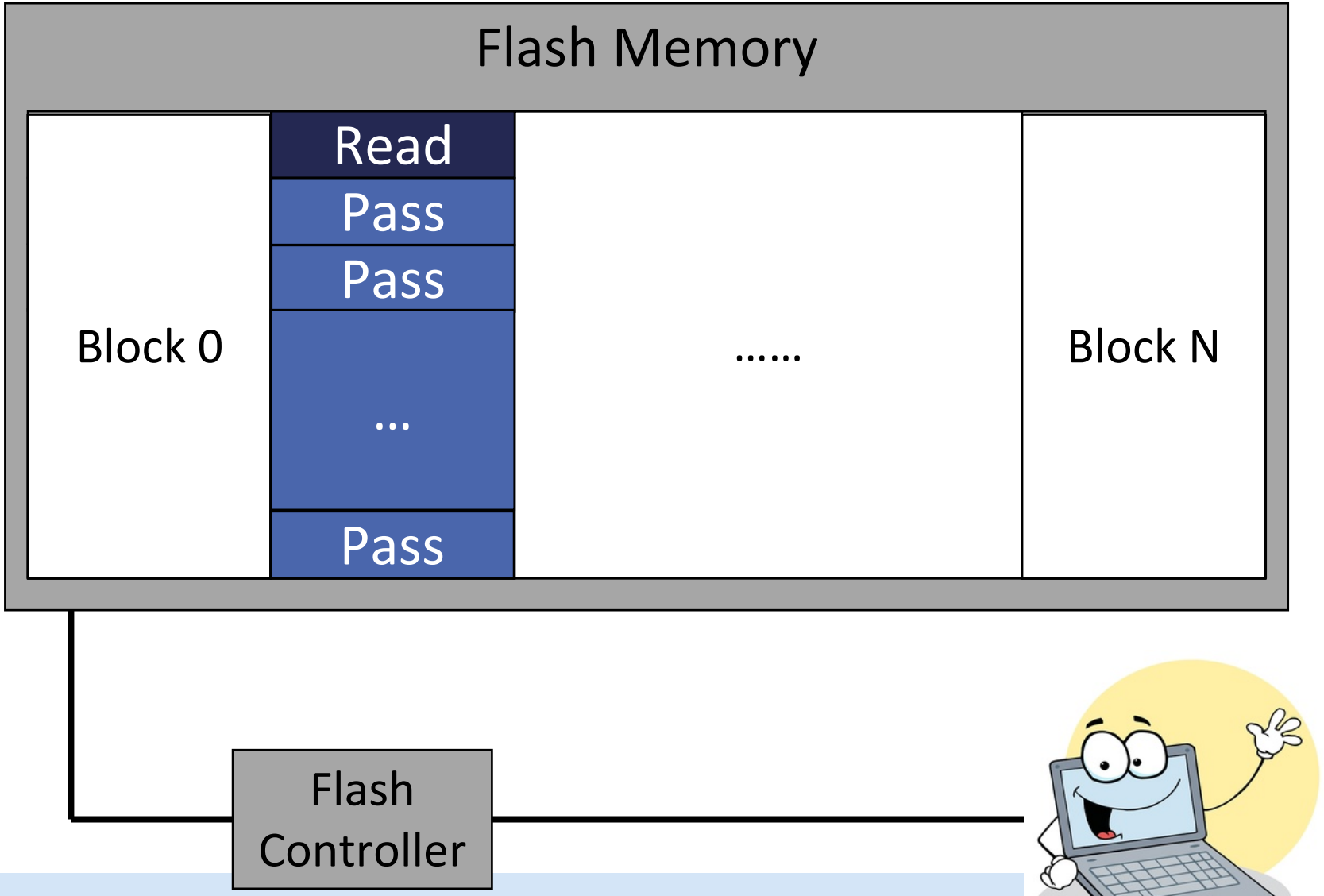
- An example mechanism:
  - For each cache block in memory, store P+1 bits in directory
    - One bit for each cache, indicating whether the block is in cache
    - Exclusive bit: indicates that a cache has the only copy of the block and can update it without notifying others
  - On a read: set the cache's bit in the directory entry for the block and arrange the supply of the block into the cache
  - On a write: invalidate the block in all caches that have the block and reset their bits in the directory entry for the block
  - Have an "exclusive bit" associated with each block in each cache (so that the cache can update the exclusive block silently)

Hardware Design

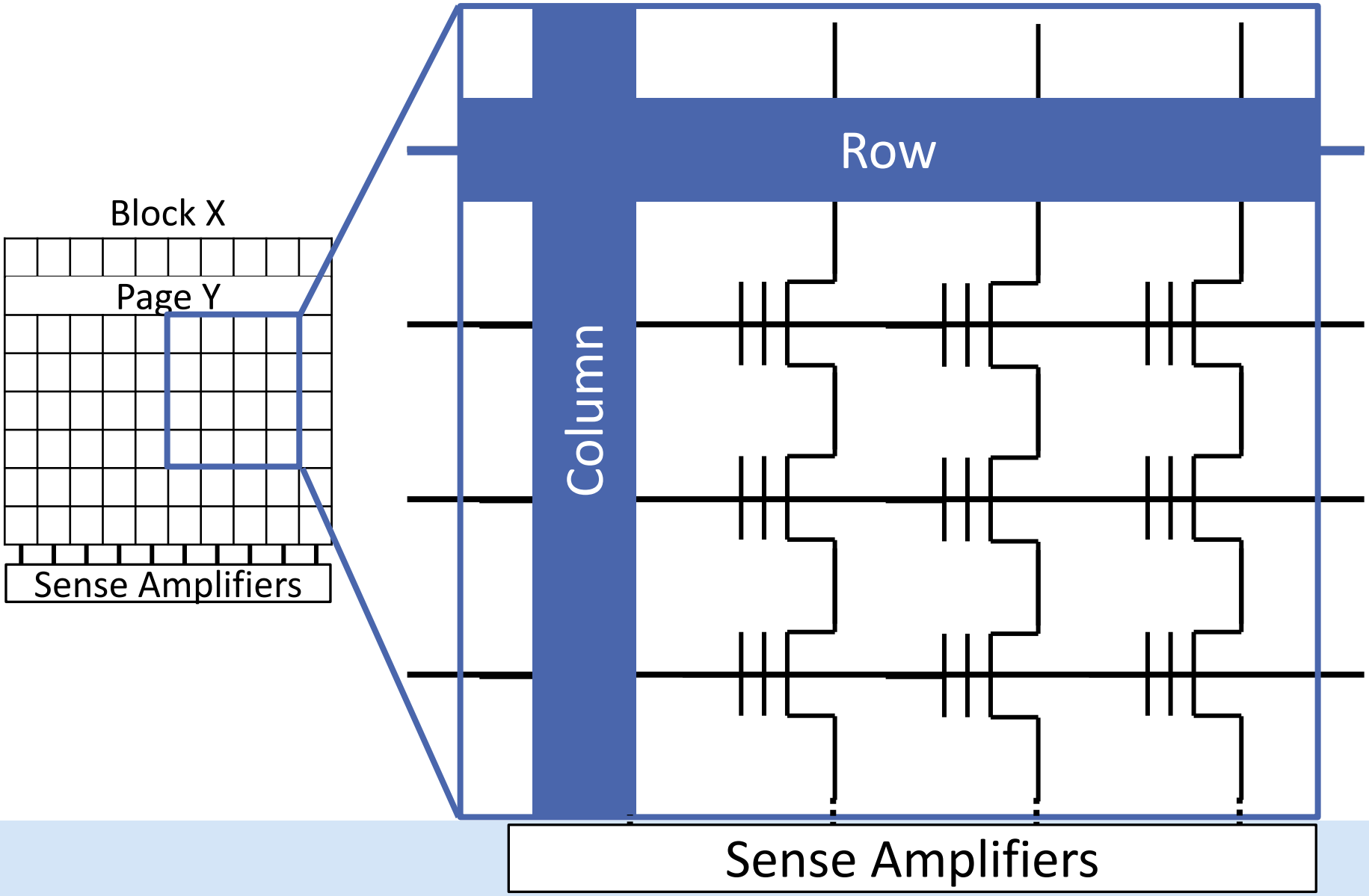
Flash Memory and Solid-State Drives



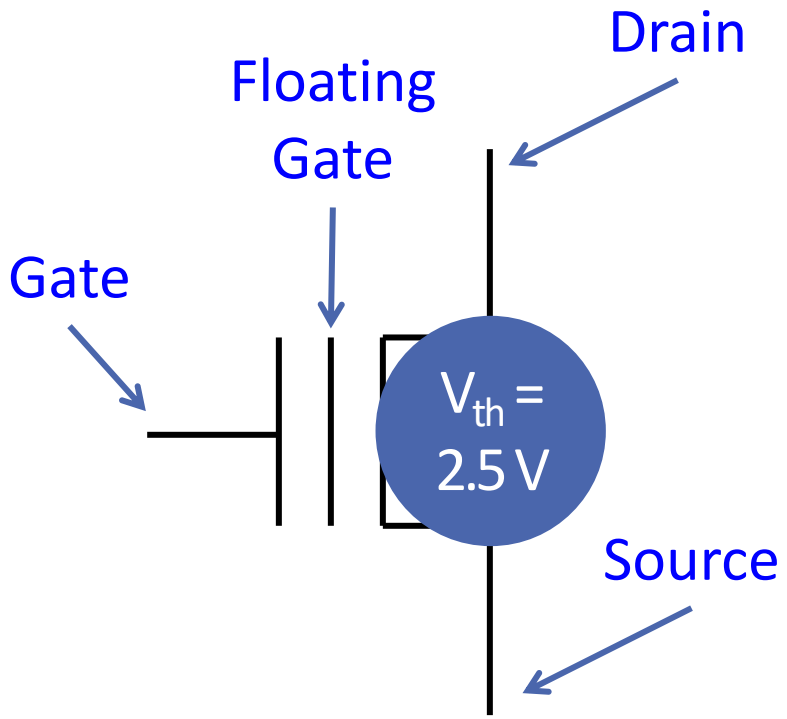
Background: NAND Flash Memory Operation



Flash Cell Array

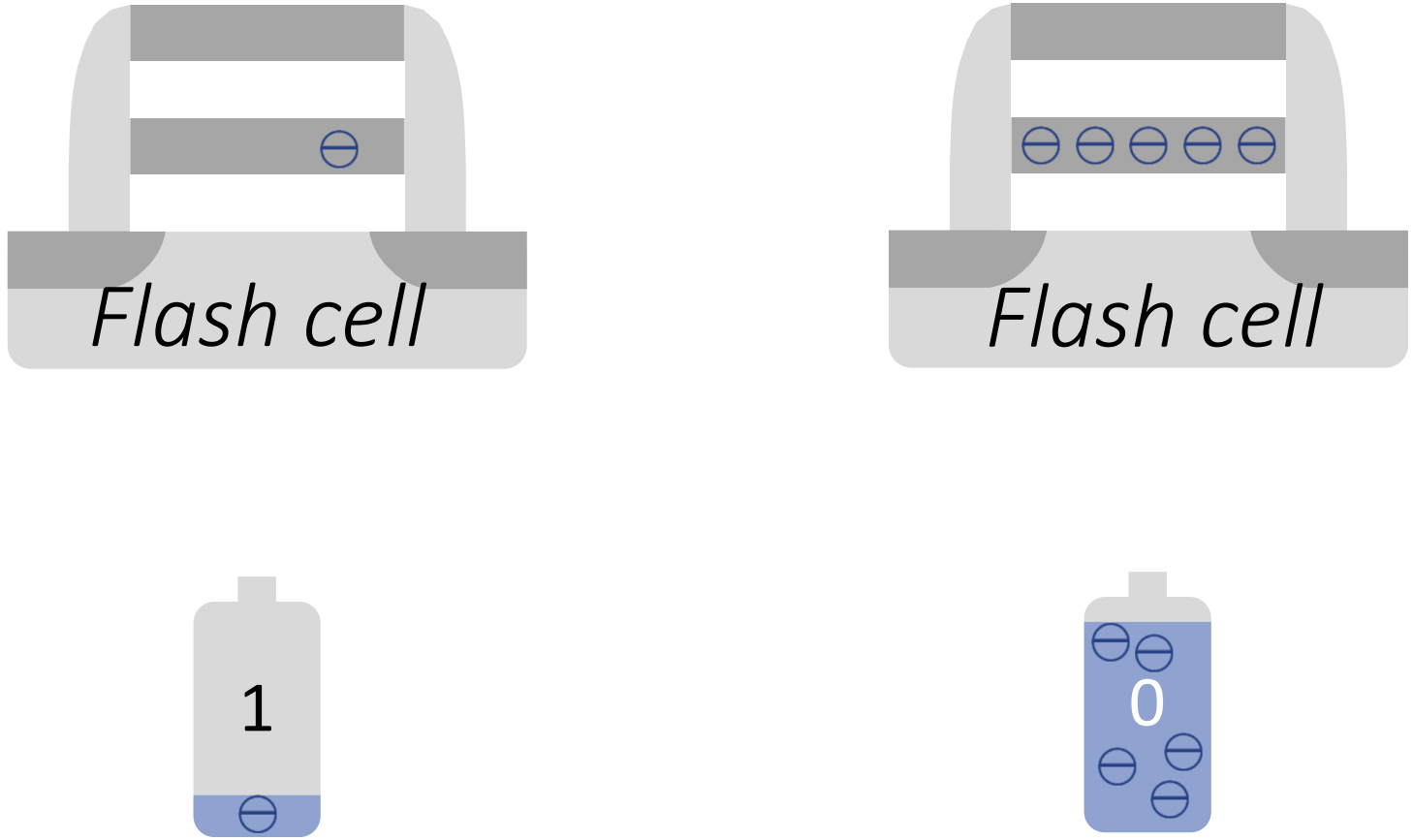


Flash Cell

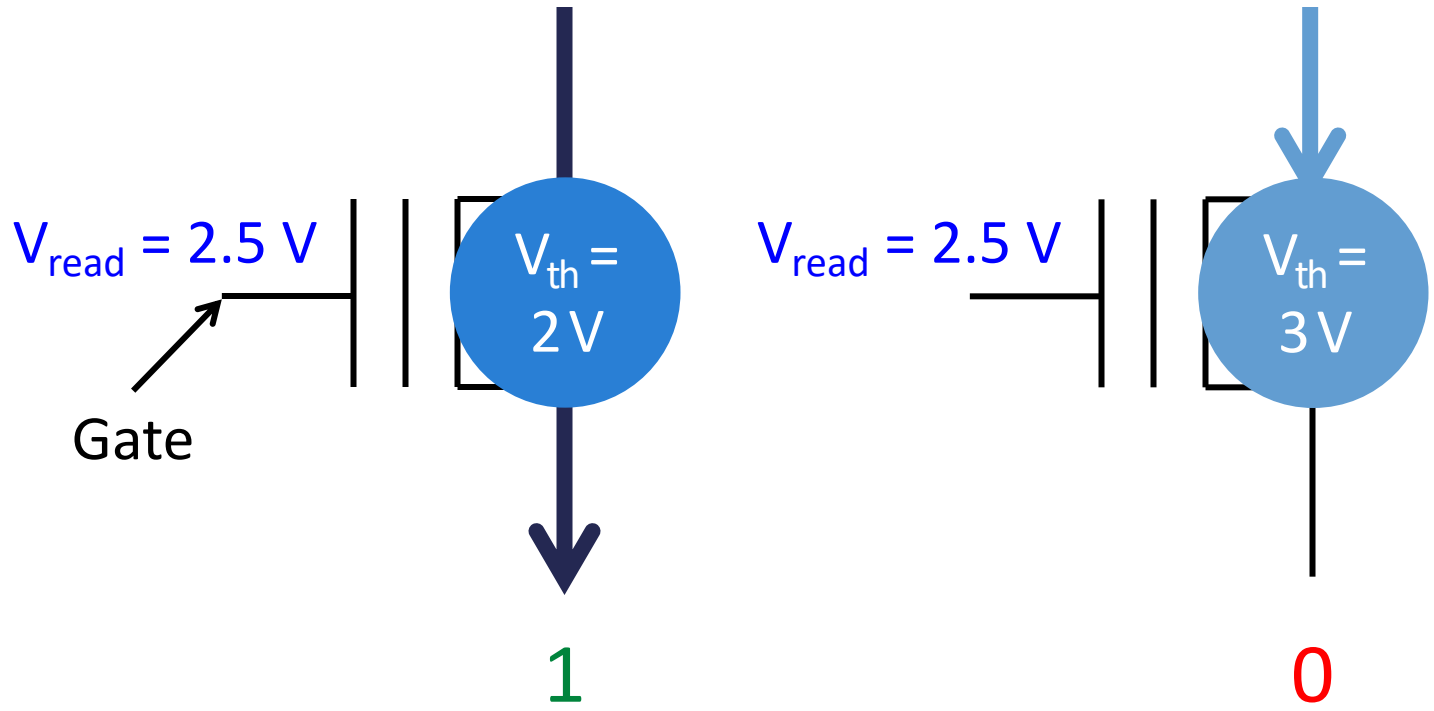


Floating Gate Transistor
(Flash Cell)

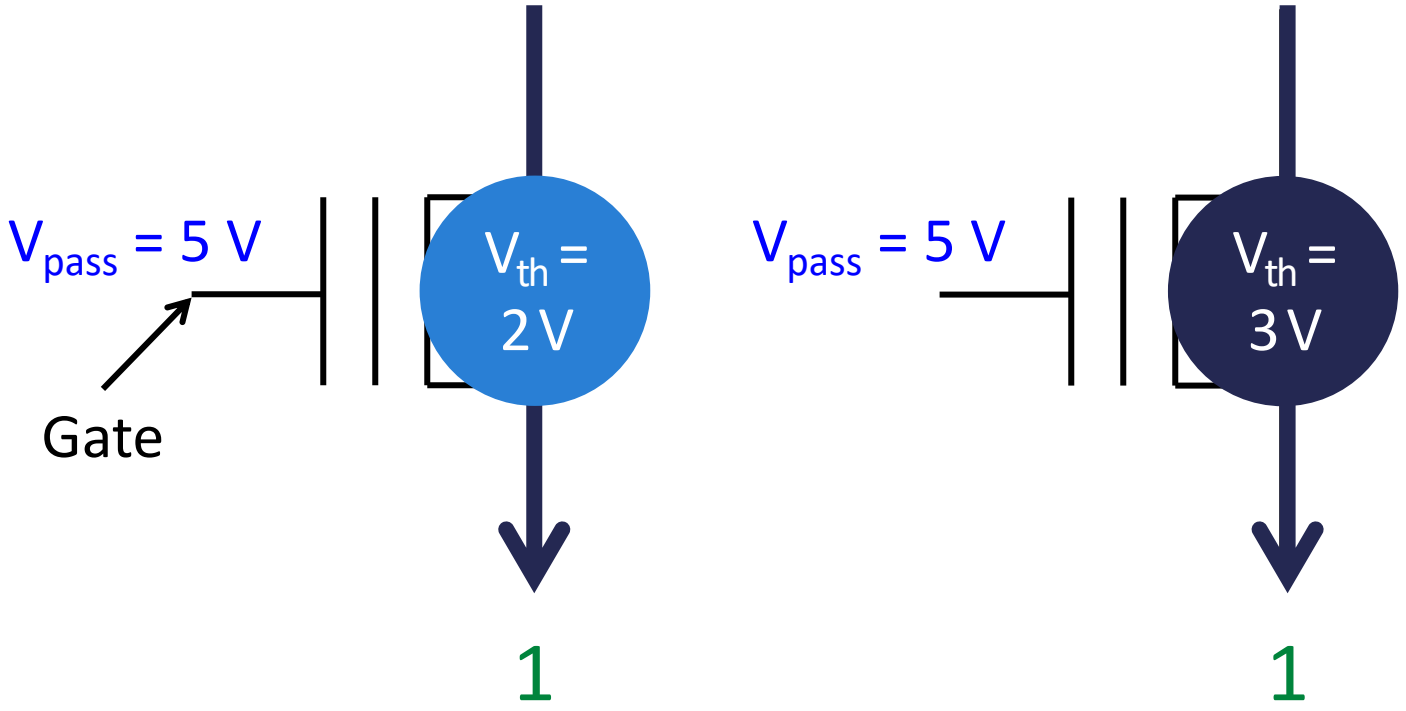
Threshold Voltage (V_{th})



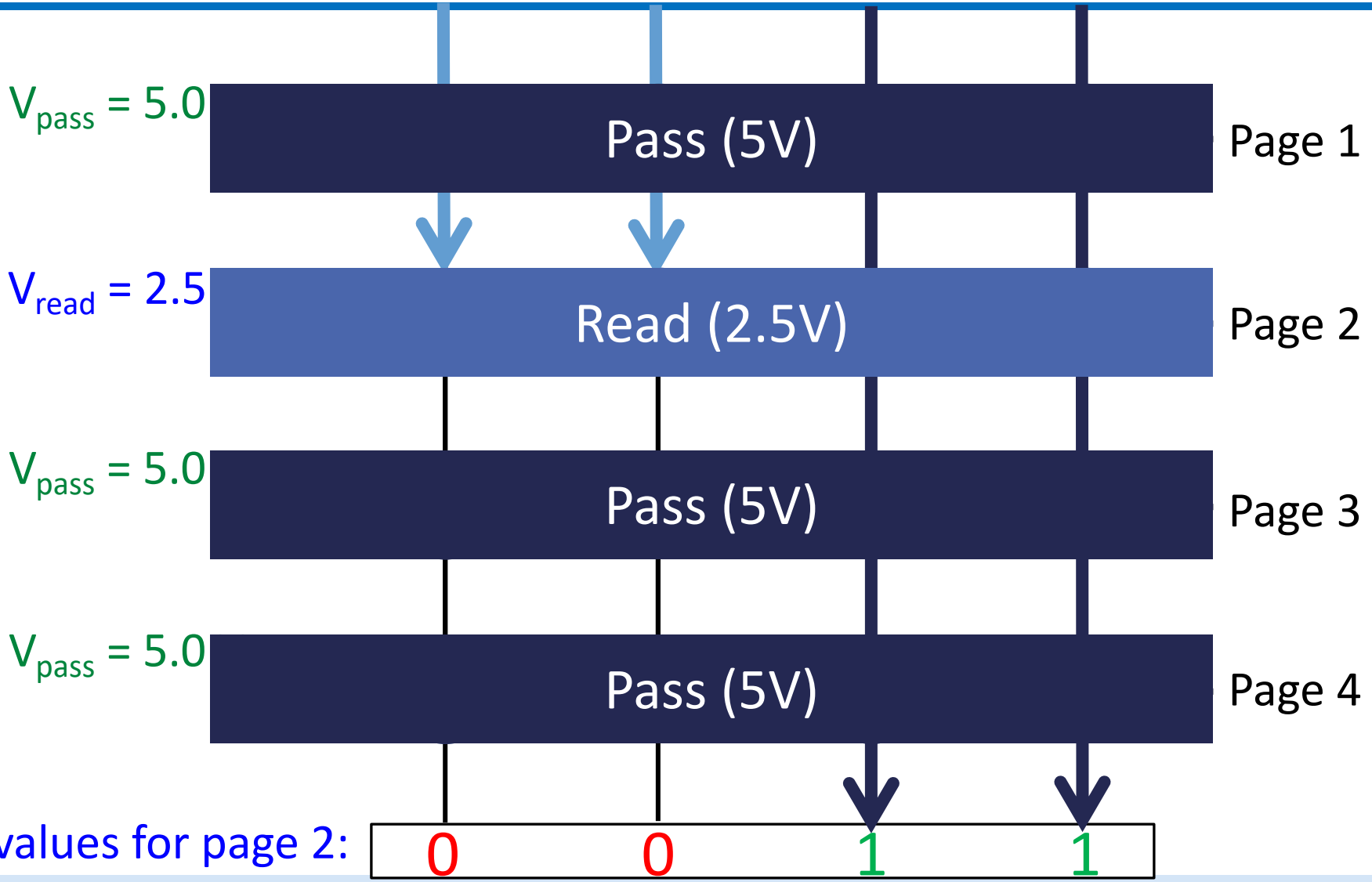
Flash Read



Flash Pass-Through

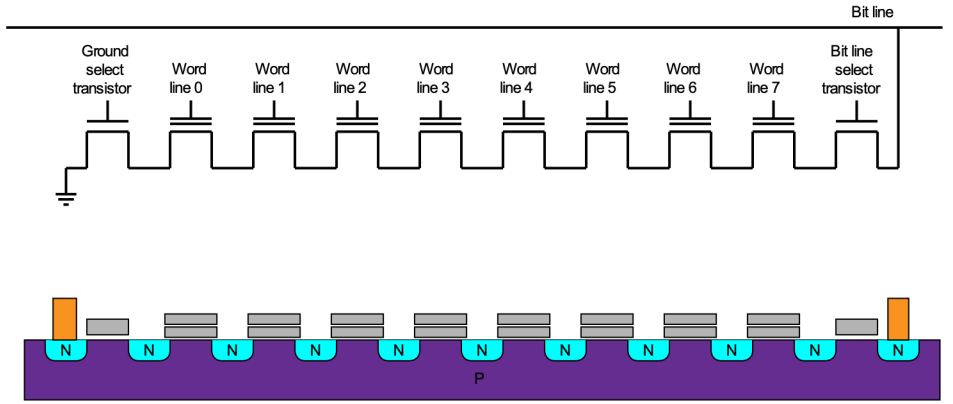


Read from Flash Cell Array

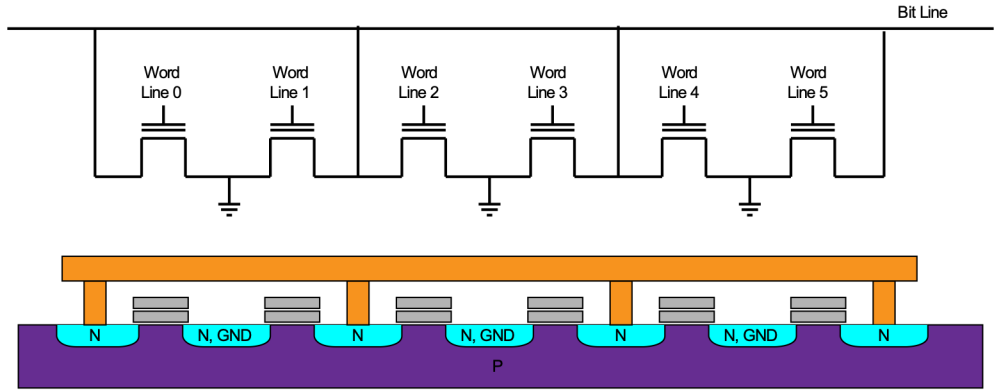


Aside: NAND vs. NOR Flash Memory

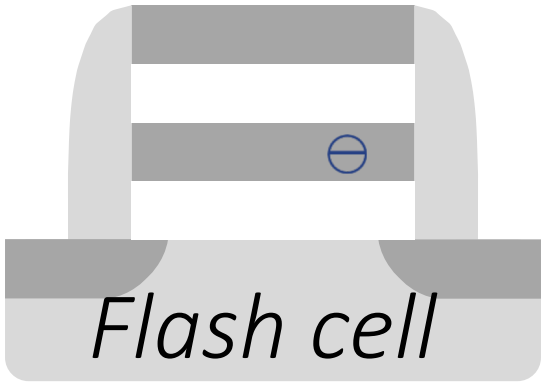
NAND

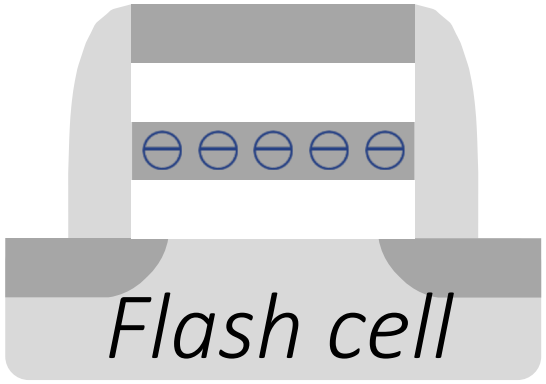


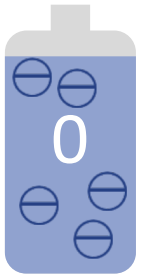
NOR



Threshold Voltage (V_{th})

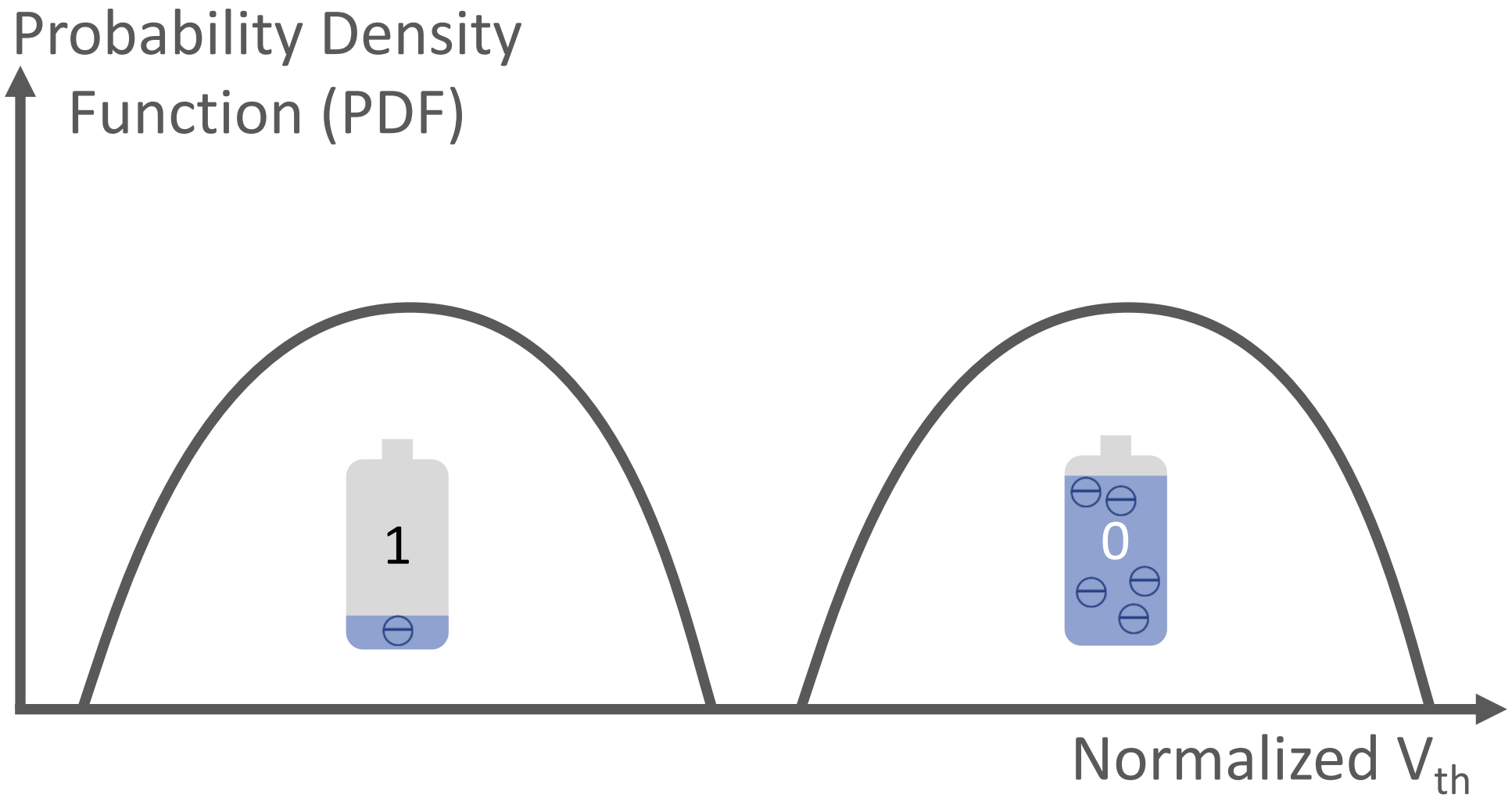




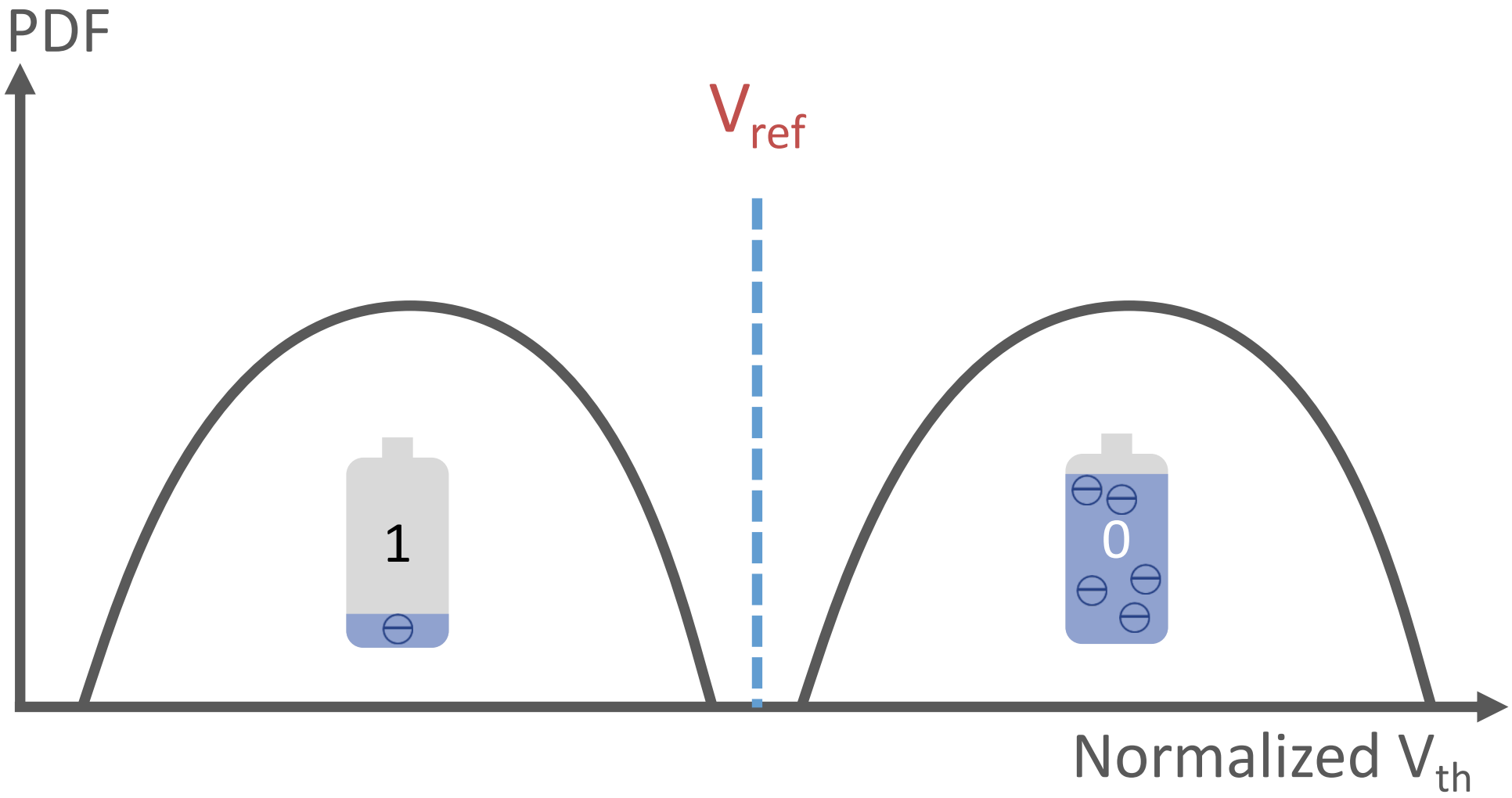


Normalized V_{th}

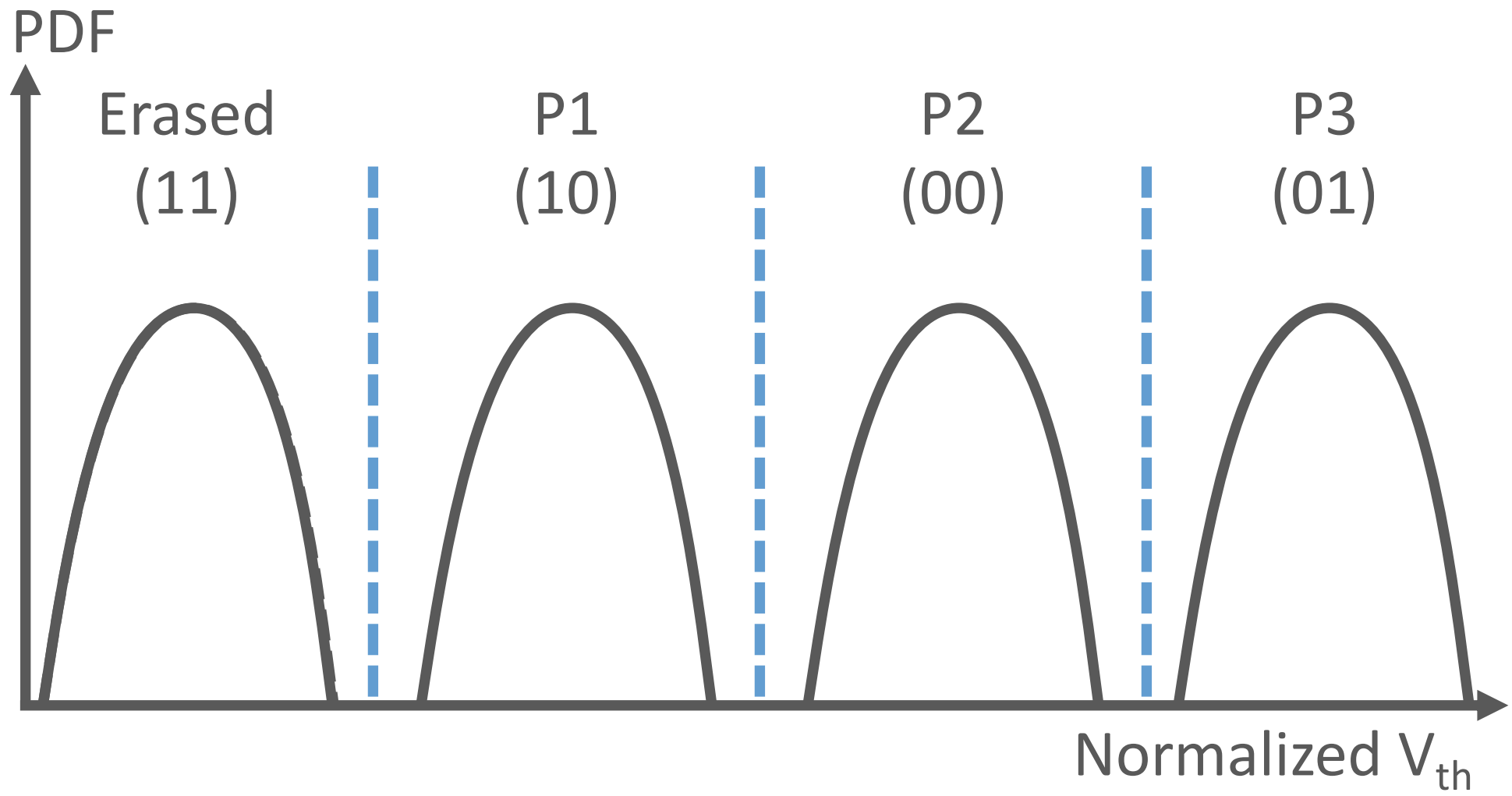
Threshold Voltage (V_{th}) Distribution



Read Reference Voltage (V_{ref})

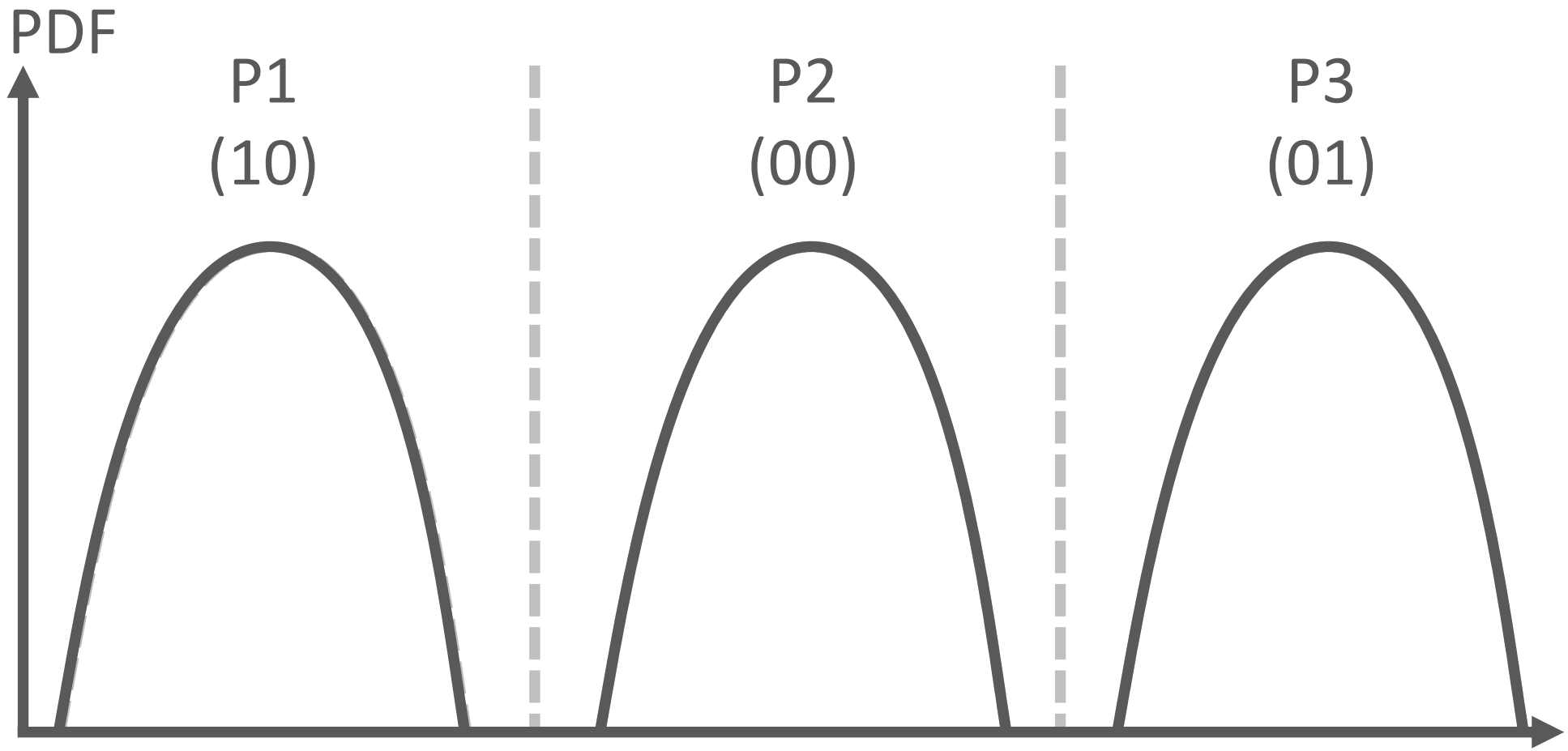


Multi-Level Cell (MLC)



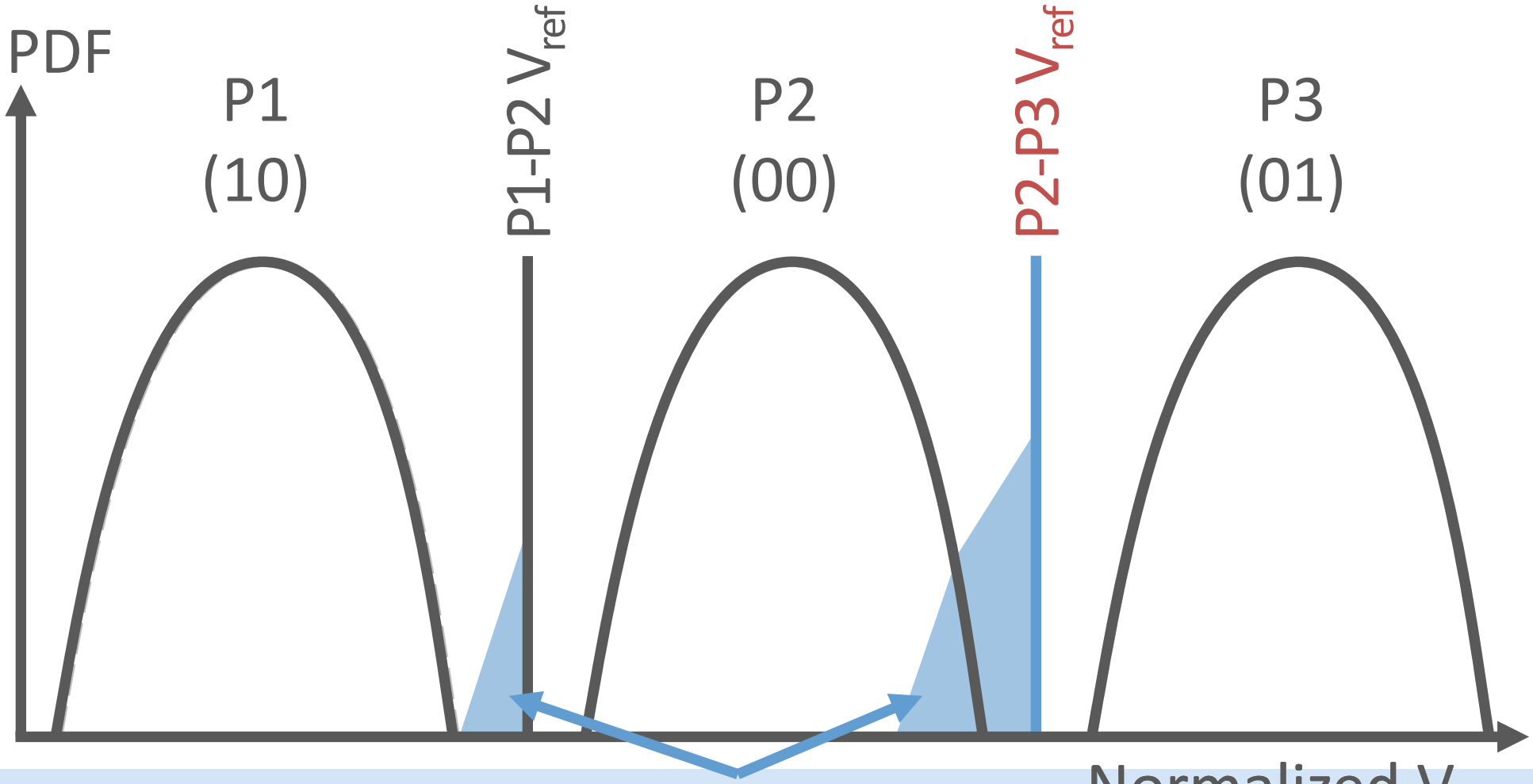
Threshold Voltage Reduces Over Time

After some retention loss:



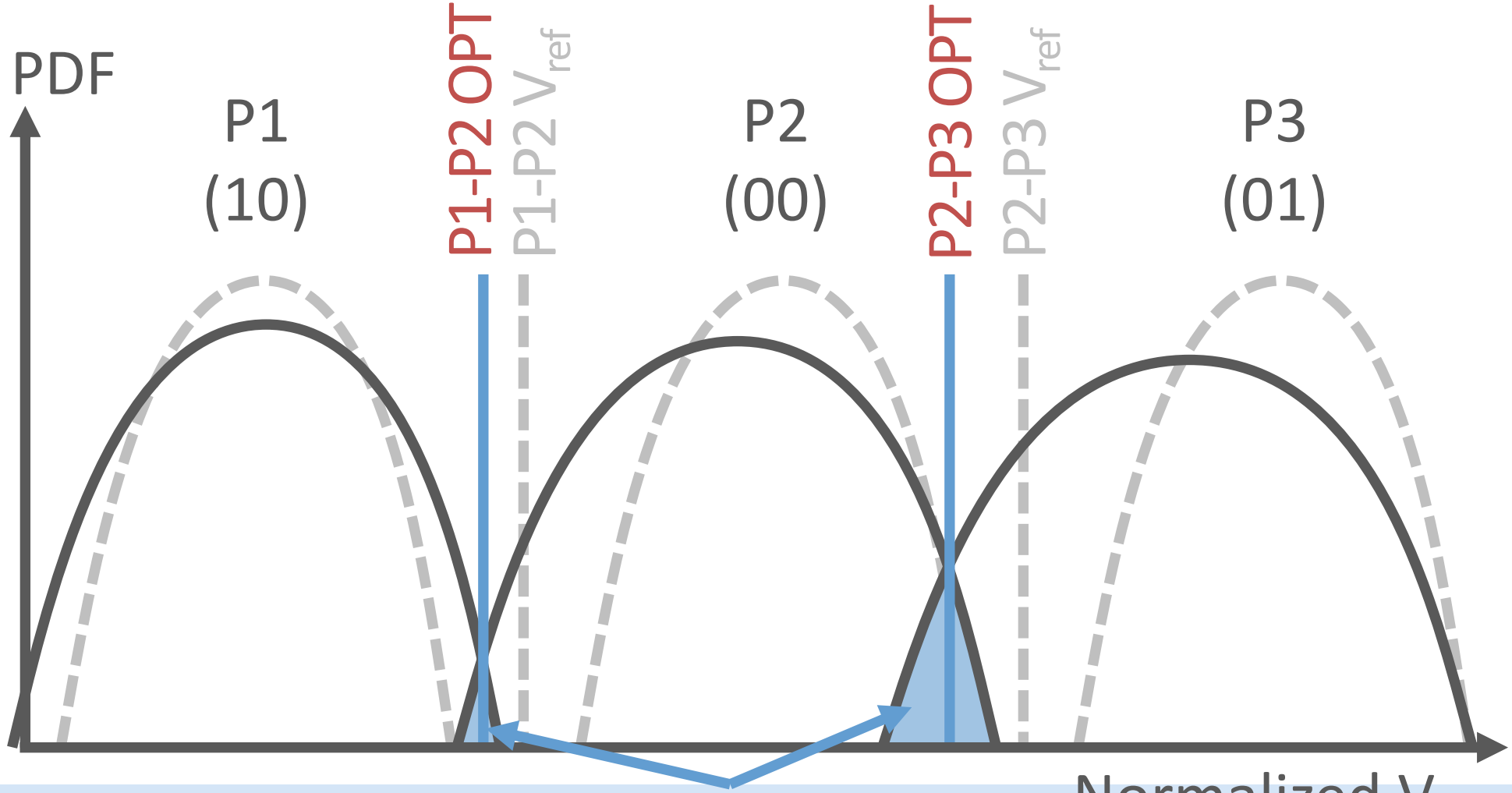
Fixed Read Reference Voltage Becomes Suboptimal

After some retention loss:



Optimal Read Reference Voltage (OPT)

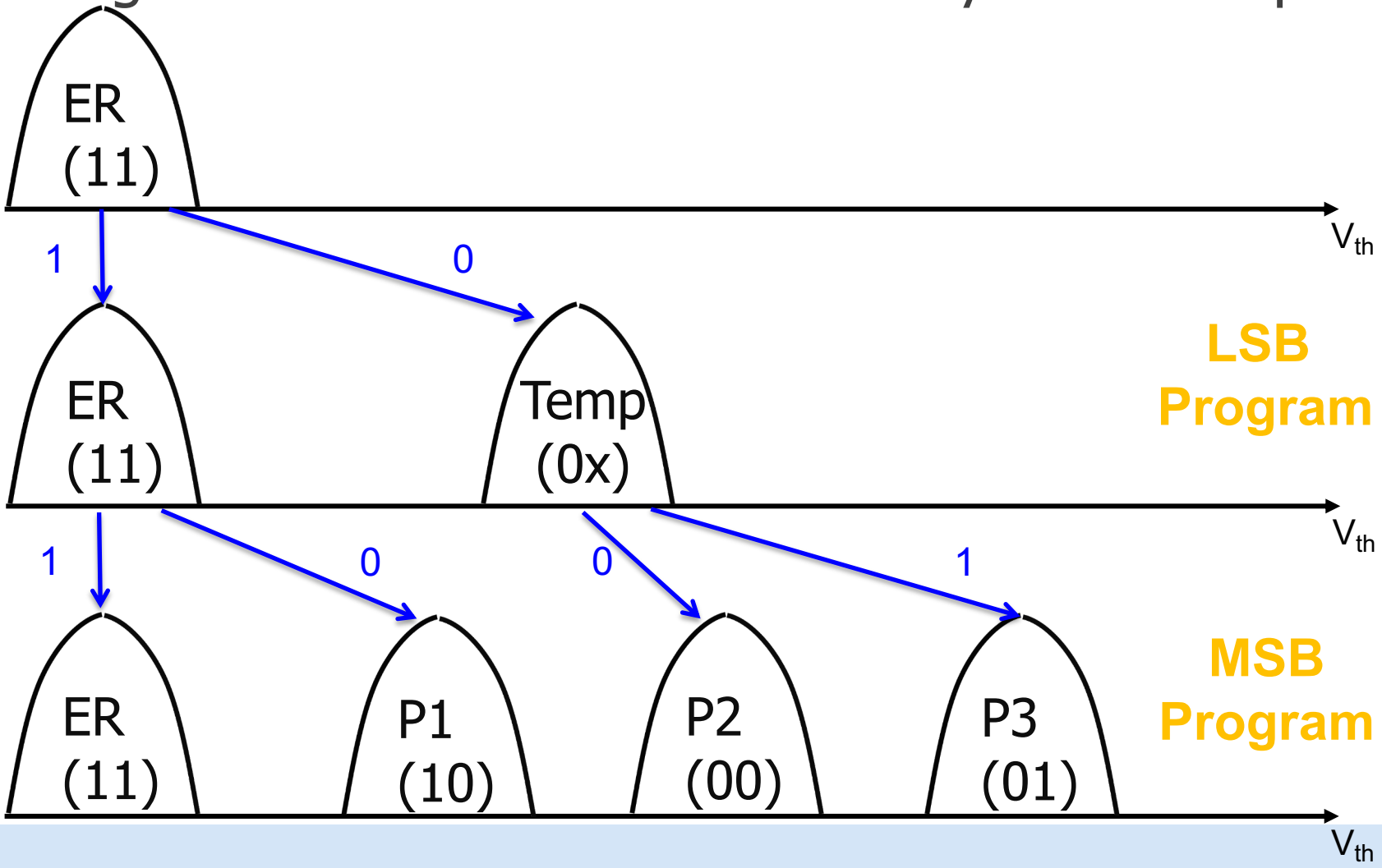
After some retention loss:



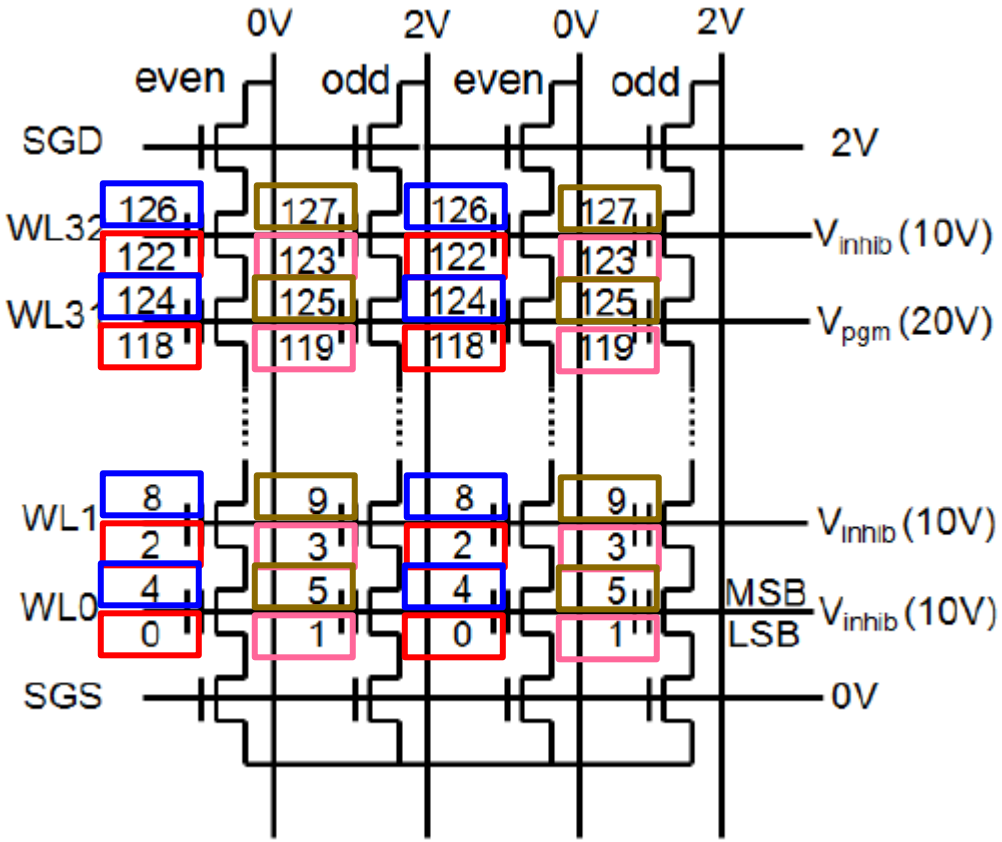
Minimal raw bit errors

How Current Flash Cells are Programmed

- Programming 2-bit MLC NAND flash memory in two steps



MLC Architecture



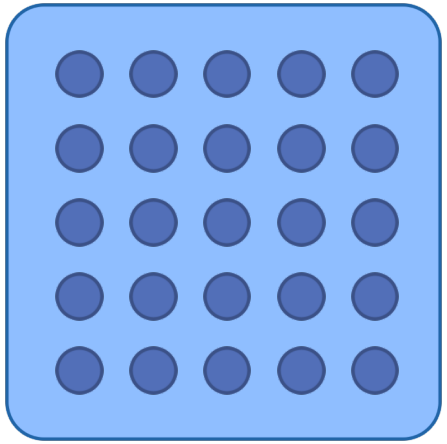
LSB-Even Page Sets

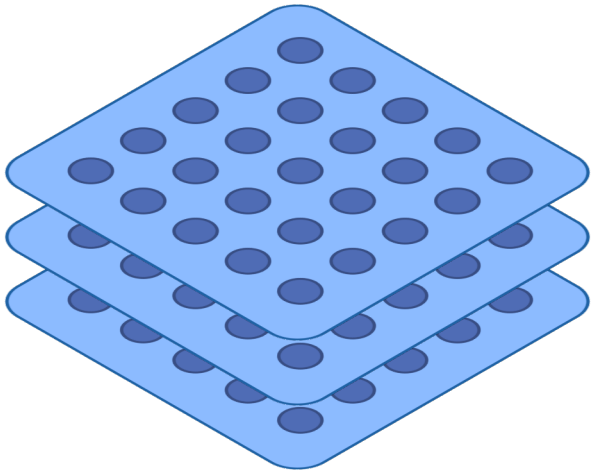
LSB-Odd Page Sets

MSB-Even Page Sets

MSB-Odd Page Sets

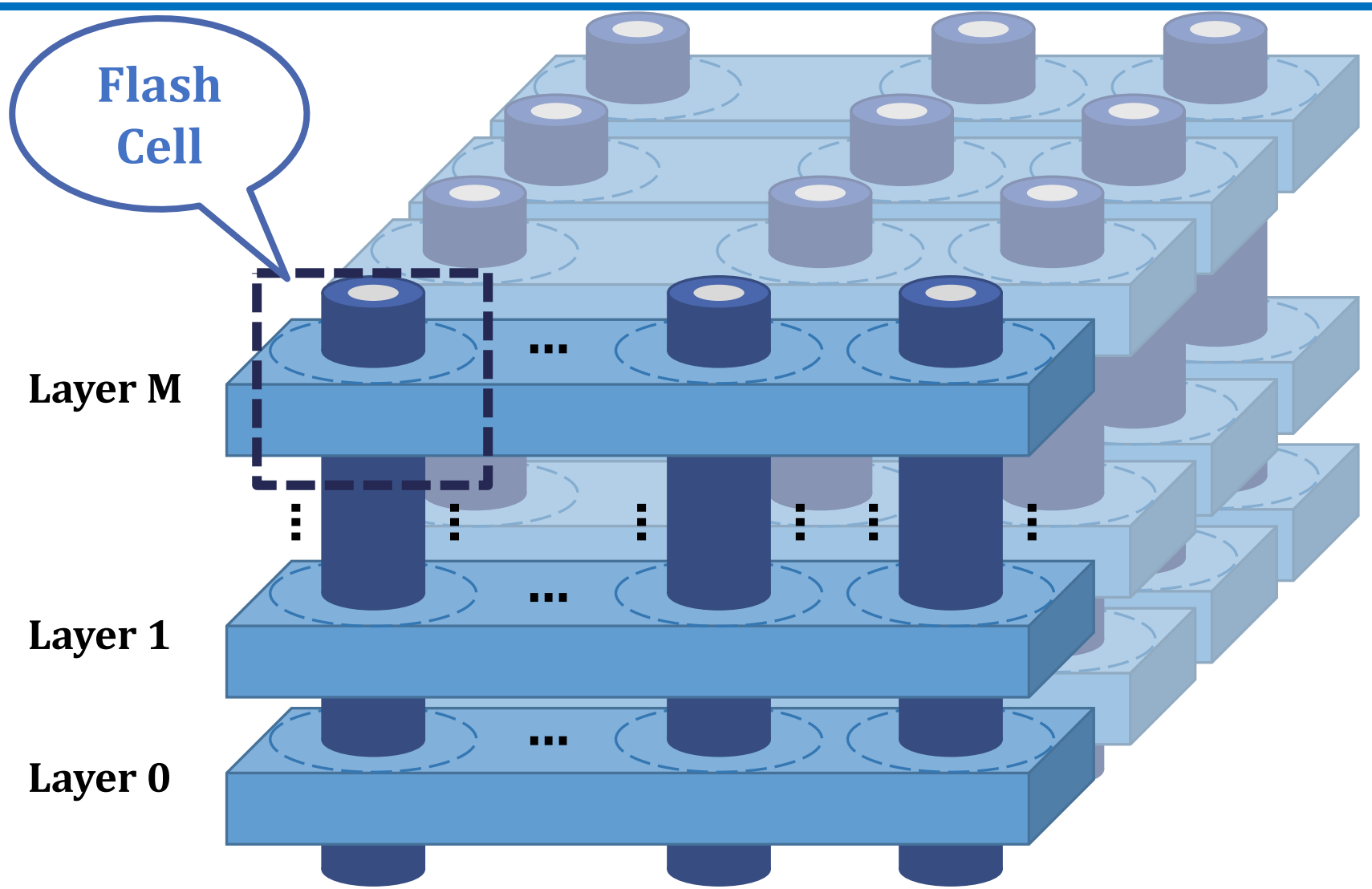
Planar vs. 3D NAND Flash Memory





	Planar NAND Flash Memory	**3D NAND Flash Memory**
Scaling	Reduce flash cell size, Reduce distance b/w cells	Increase # of layers

3D NAND Flash Memory Structure

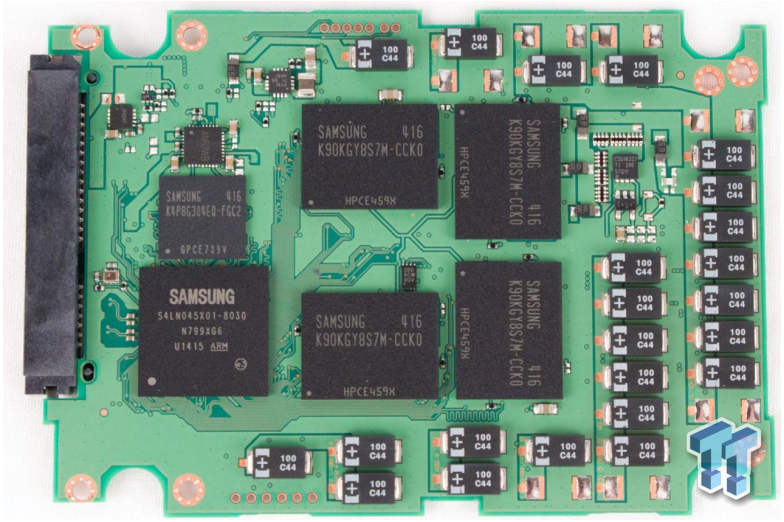


Error Analysis and Management

- Main Characterization Results
- Retention-Aware Error Management
- Threshold Voltage and Program Interference Analysis
- Read Reference Voltage Prediction
- Neighbor-Assisted Error Correction
- Read Disturb Error Handling
- Retention Error Handling
- 3D NAND Flash Memory Reliability

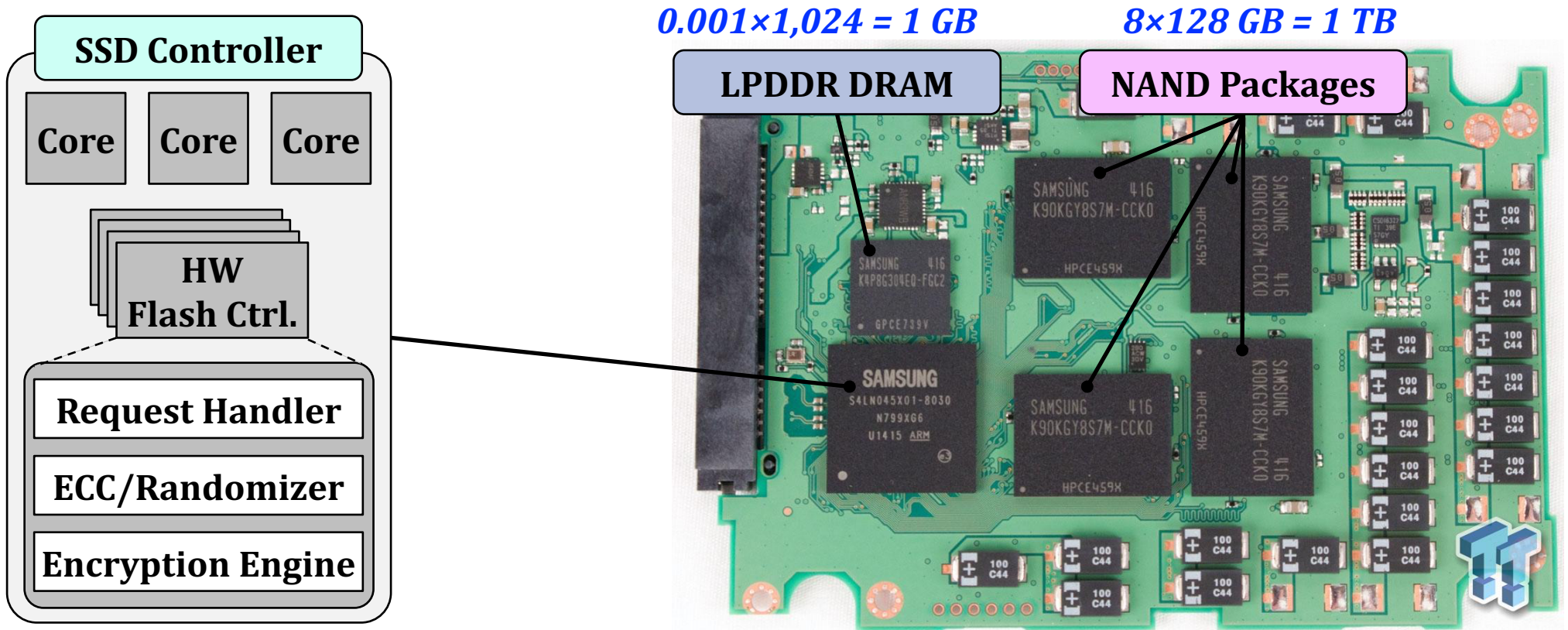
Hardware Design

Modern SSD Architecture



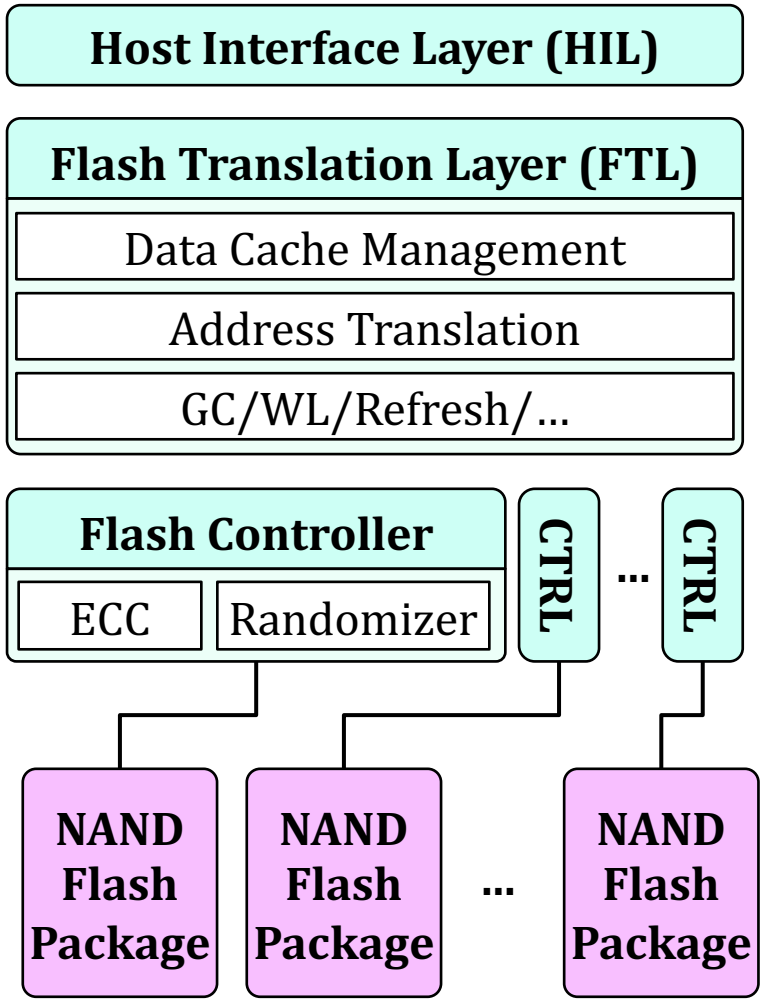
Modern SSD Architecture

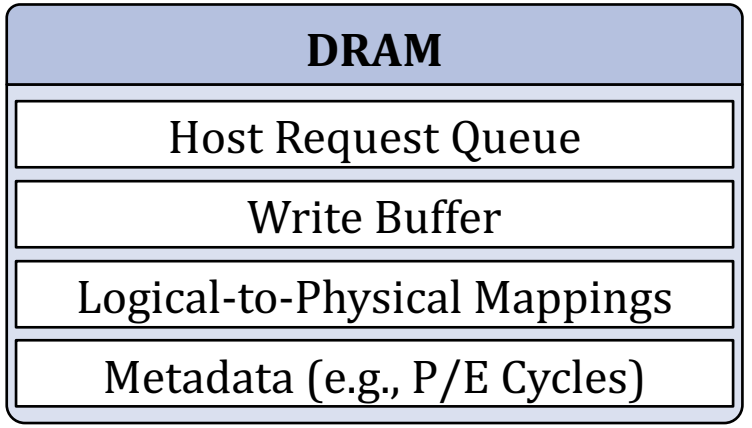
- A modern SSD is a complicated system that consists of multiple cores, HW controllers, DRAM, and NAND flash memory packages

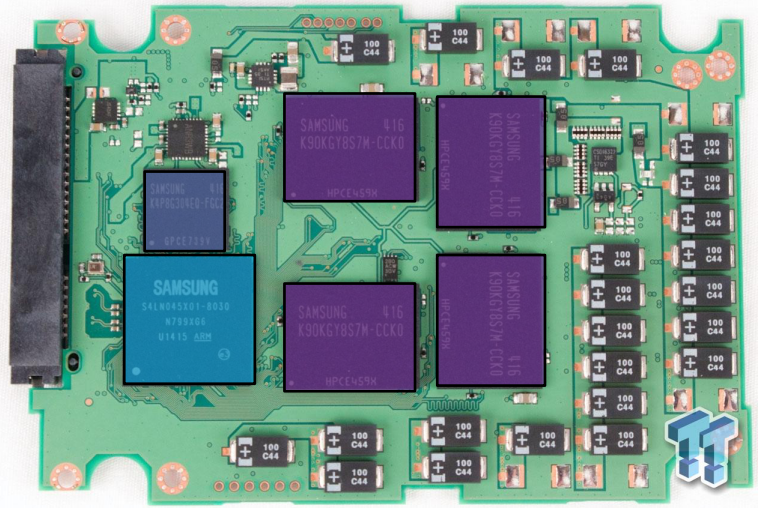


Samsung PM853T 960GB Enterprise SSD (from https://www.tweaktown.com/reviews/6695/samsung-pm853t-960gb-enterprise-ssd-review/index.html)

Another Overview







KING'S College LONDON

Hardware Design

Lecture 9: Cache (II) and Storage

Dr. Haiyu Mao

26.03.2026