CS647 Modern Storage Systems

Hard Disk Drives (HDDs)

NAND Flash & Solid-State Drives (SSDs)

Non-Volatile Memory (NVM)

CS647 Modern Storage Systems, Fall 2022
(Modern Key-Value Store Design and Implementation)
Angelos Bilas, bilas@csd.uoc.gr
Computer Science Department, University of Crete

Outline

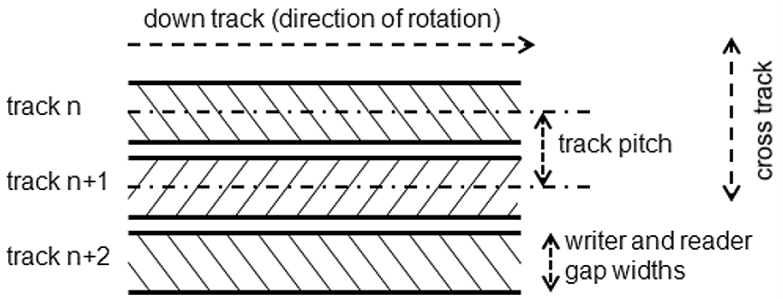
- HDDs
  - Capacity, cost
  - Shingled Magnetic Recording
- NAND Flash
  - Flash cells and arrays: How it works, basic operations, errors
  - SSDs: Structure, concurrency
  - FTL: Garbage collection, wear leveling
- Non-volatile memories (NVM)
- Remarks

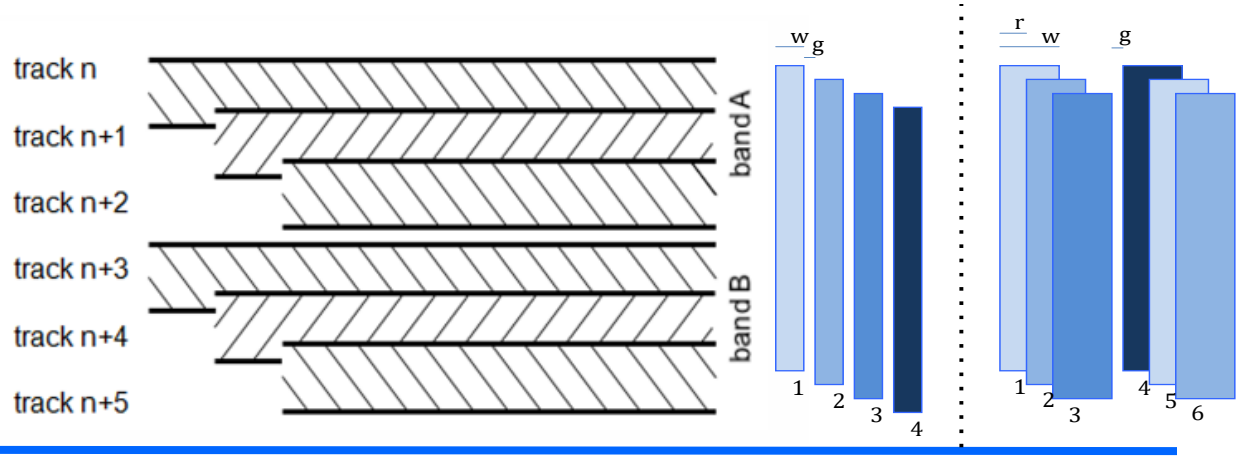
Hard Disk Drives (HDDs)

- We know what they are…
  - From: https://www.enterprisestorageforum.com/hardware/hdd-industry-trends
  - (data) IDC reported high volumes of created data in 2020, partly due to people staying home more during the pandemic. And in a 2020 report done in partnership with IDC, Seagate Technologies predicts that by 2025, data will increase exponentially to hit 175 created zettabytes.
  - (HDDs) "Hard disk drives to remain dominant storage media in 2022." "Global hard drive unit sales are decreasing, the overall storage capacity of HDDs is increasing, according to Horizon Technology. The HDD market crossed an impressive threshold in 2020, hitting one zettabyte (ZB) of shipped capacity."
  - (vs. SSDs) Overall, standard enterprise hard drives have greater capacity than solid-state drives. And though plenty of data center SSDs range into the multiple-terabyte size, where HDDs win is their cost per capacity. Hard drives are still cheaper overall. While SSDs are popular, hard drive capacity continues to increase, IDC expects that HDDs will remain the most common installed storage medium until at least 2024.
- Since the 1960s, year after year, capacity has improved
  - 30-40% per year on average, step function
  - Every 10 years or so an important technique appears and refinements follow for a number of years
- Latest technique: Shingled Magnetic Recording (SRM)
  - And variants: BPMR, HAMR, MAMR, SWR, TDMR, etc.

Shingled Magnetic Recording (SRM)

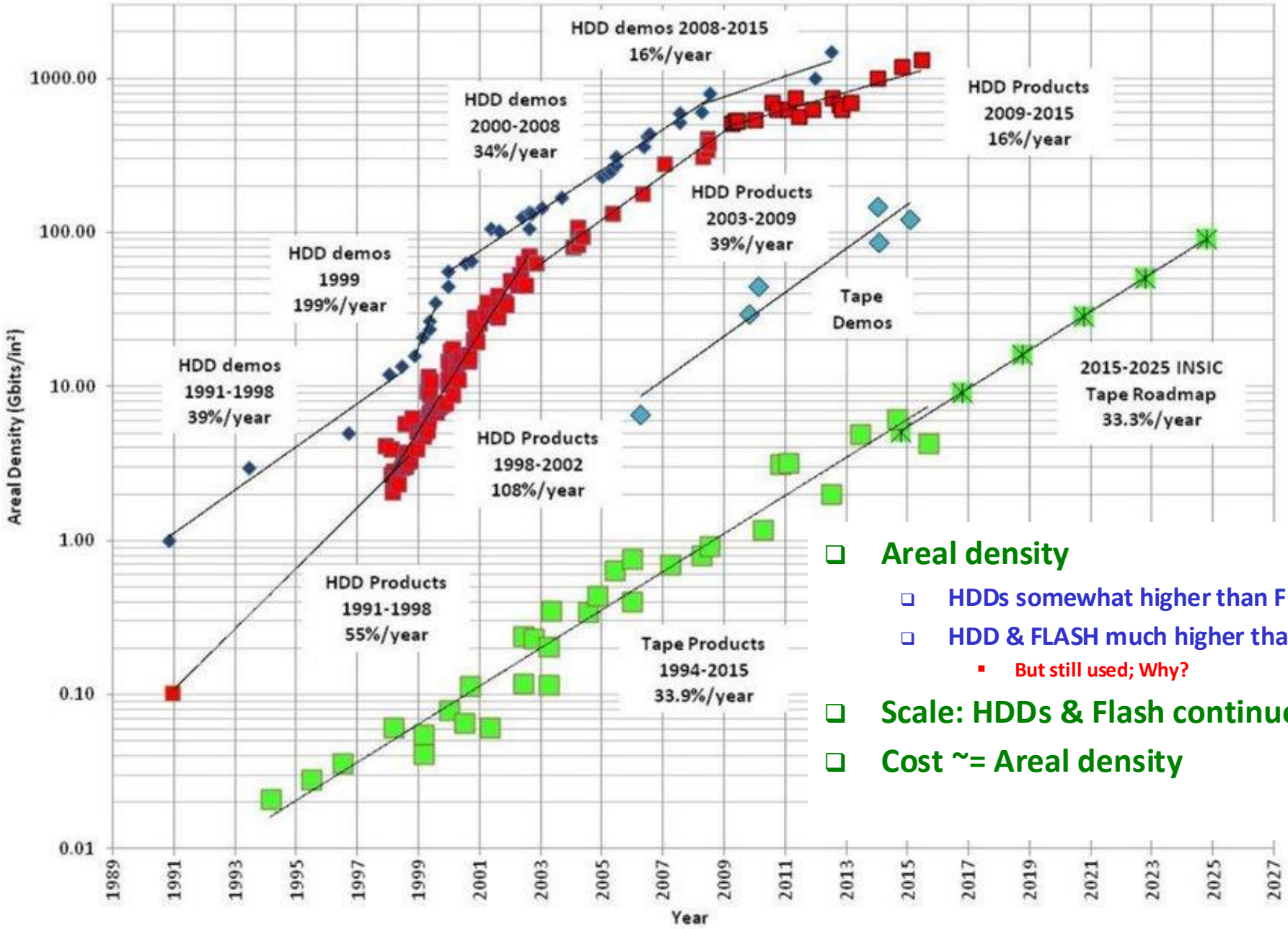
- [Shingled Magnetic Recording Areal Density Increase Requires New Data Management. T. FELDMAN AND G. GIBSON. Usenix ;login: JUNE 2013, VOL. 38 NO. 3]
  [G. Gibson and G. Ganger, "Principles of Operation for Shingled Disk Devices," Technical Report CMU-PDL- 11-107, Parallel Data Lab, Carnegie Mellon University, Pittsburgh, PA, 2011.]
- Disk head has a reader and a writer component
- Until recently both reader and writer became smaller (narrower) (figure below/left+right)
  - This allowed tracks to become narrower and more => higher density
- Reader continues to scale down, but not writer (figure below/right)
  - Narrower tracks cannot be written
- SMR (figure below/center)
  - 1. Narrow reader, wider writer
  - 2. Overlap tracks in bands, let's say 3-4-5 tracks per band
    - Bands have space in between
  - 3. When writing you destroy the track on one direction, leaving the other direction in readable state
    - Important: you do not destroy both directions (let's say above/downstream and below/upstream) of the overlap
    - This is similar to layering shingles in roofs
- Writing a sector in a track, destroys downstream sectors/tracks (above) in the same bank
  - You need to write those again (but need to read them first => read-modify-write)
  - Essentially, you can read a sector but you need to write a block (variable size, depending on location)
  - **Issue: Appropriate logical to physical block mappings so that read-modify-write operations "make sense" for the application for optimization/efficiency purposes**
  - Solutions: HDD firmware or host-level





Areal Density Trends

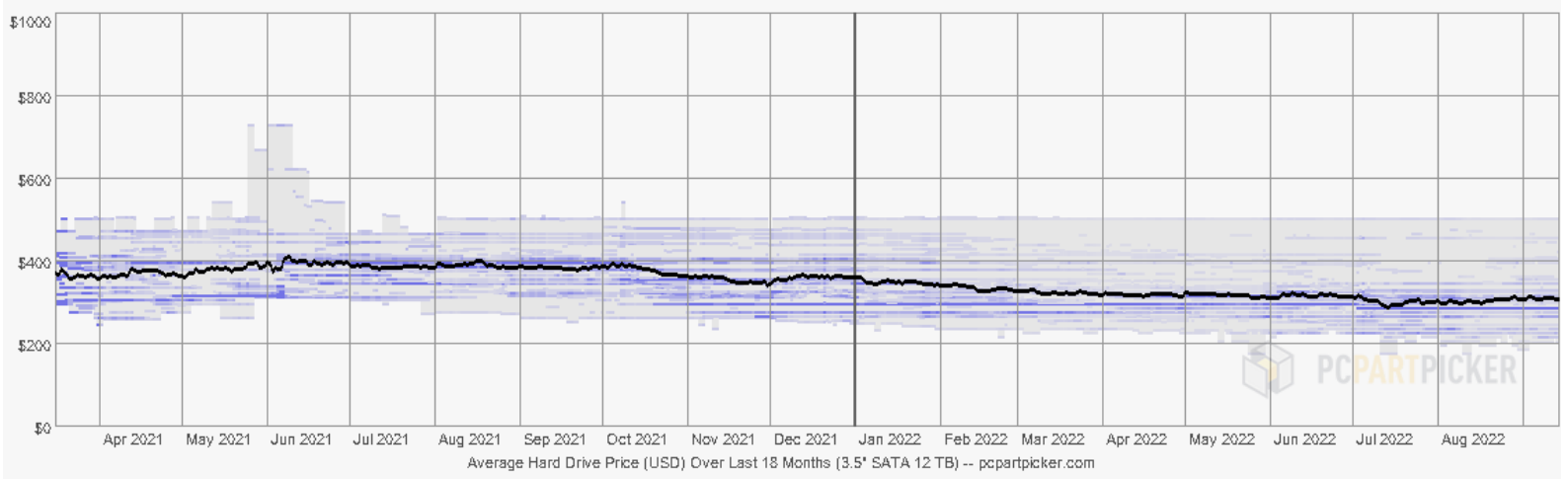
[Li, Hai (2018). Storage Physics and Noise Mechanism in Heat-Assisted Magnetic Recording. Carnegie Mellon University. Thesis. https://doi.org/10.1184/R1/6723209.v1]



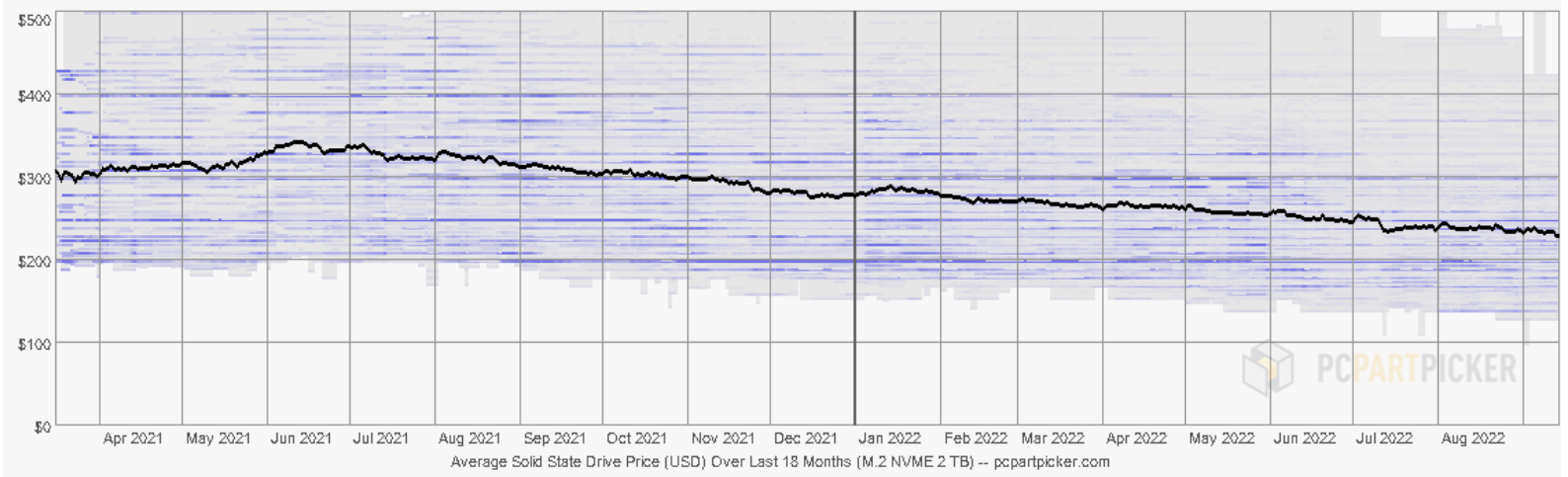
- **Areal density**
  - HDDs somewhat higher than FLASH (say 2x)
  - HDD & FLASH much higher than Tape
    - But still used; Why?
- **Scale: HDDs & Flash continue to scale**
- **Cost ~= Areal density**

HDDs vs SSDs Cost

- HDDs are most cost effective
  - https://pcpartpicker.com/trends/price/internal-hard-drive/
  - Other parameters matter besides cost
- Hard Drive - 3.5" SATA 12 TB (Average price in USD over last 18 months)



- Solid State Drive - M.2 NVME 2 TB (Average price in USD over last 18 months)



Outline

- HDDs
  - Capacity, cost
  - Shingled Magnetic Recording
- NAND Flash (Dominates interest today)
  - NAND Flash: From cells to SSDs
  - FTL: Garbage collection, wear leveling
- Non-volatile memories (NVM)
- Remarks

Flash Memory Summit – Flash Memory Timeline 2022

https://flashmemorysummit.com/English/PDFs/FMS2022-Timeline.pdf

20+ Years Flash Floating Gate Technology



1986/1.5µm

1988/1.0µm

1991/0.8µm

1993/0.6µm

1996/0.4µm

1998/0.25µm

2000/0.18µm

2002/0.13µm

2004/90nm

2006/72nm

2007/50nm

2008/34nm
32Gbit

3+ decades of floating-gate technology scaling starting on EPROM → Flash

Technology designed for high-volume manufacturing

Source: Intel

1987 View of NVM in Compute



A Full Range of NVM in Compute



Intel® X25-E Extreme SATA Solid-State Drive



Intel® X25-M and X18-M Mainstream SATA Solid-State Drive



Intel® Turbo Memory



Intel® Z-P230 PATA Solid-State Drive



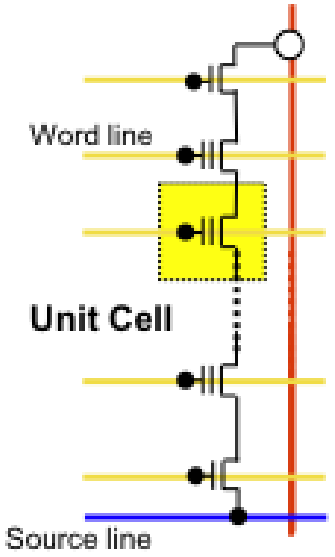
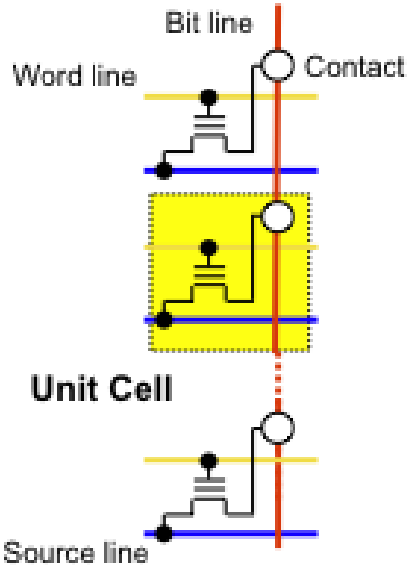
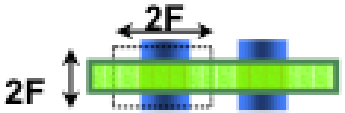
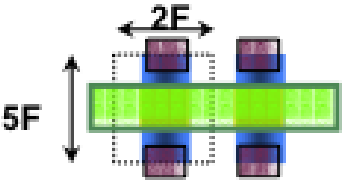
Intel® Z-P140 PATA Solid-State Drive



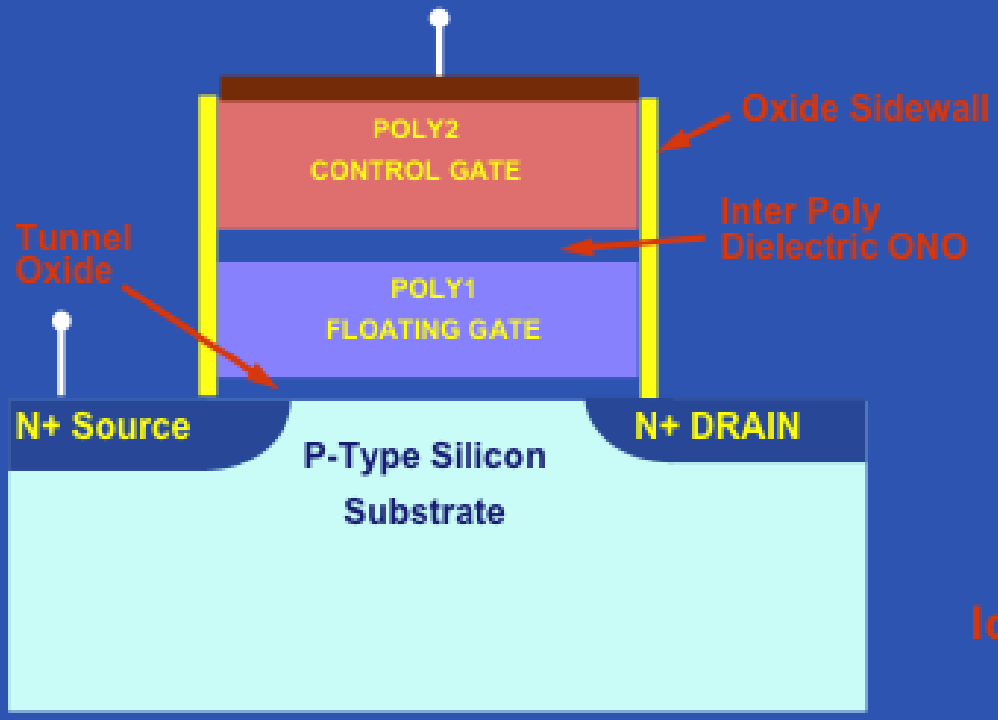
Intel® Z-U130 USB Solid-State Drive

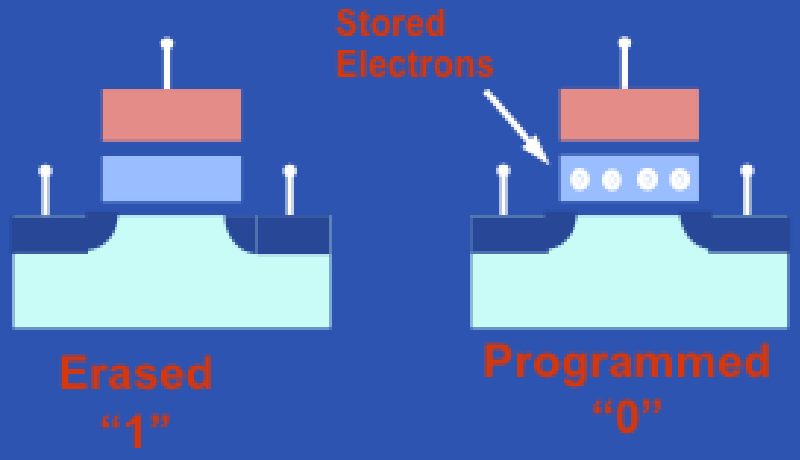
NAND FLASH: From Cells to SSDs

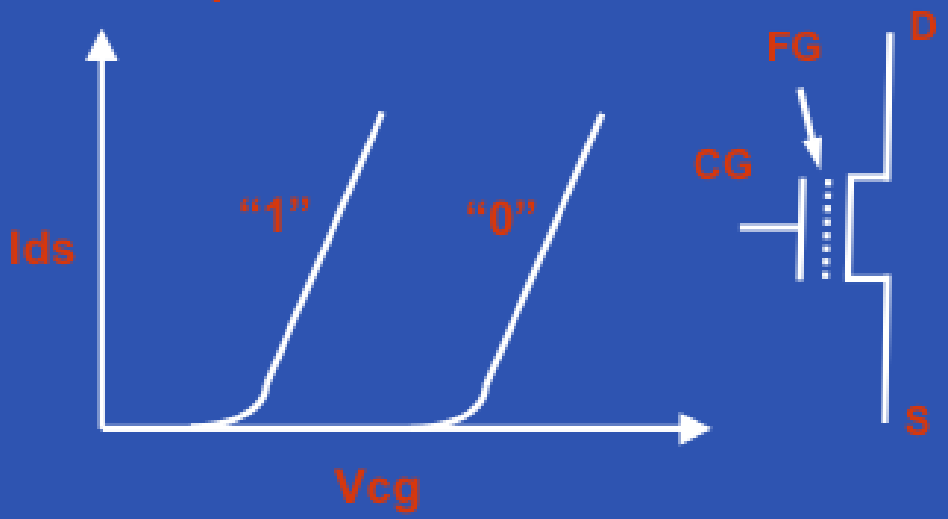
NAND vs. NOR - Cell Structure

	NAND	NOR
Cell Array		
Layout		
Cross-section		
Cell size	$4F^2$	$10F^2$

Flash Cell Layout and Cross-Section



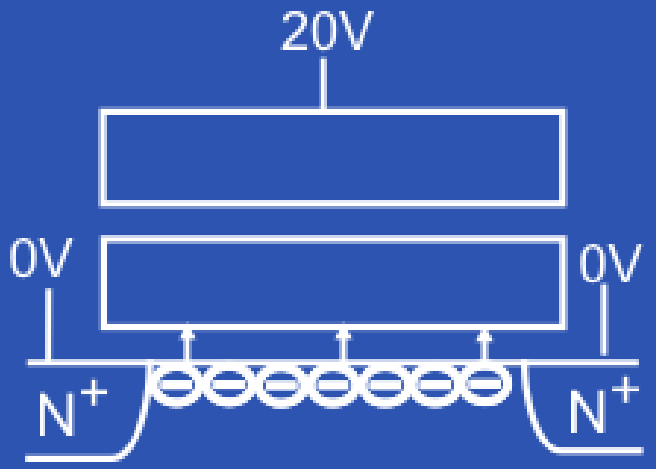




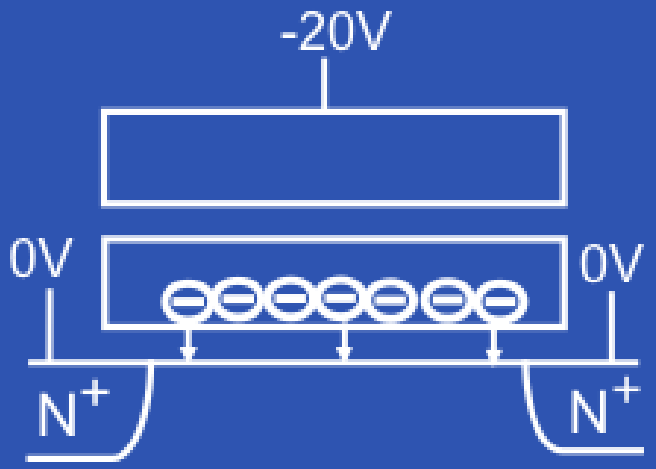
N-Channel MOSFET with a few distinguishing features:

- Isolated floating gate
- Charge storage on Floating gate modulates threshold voltage of underlying MOSFET

Charge Storage: Program and Erase



Programming: NAND



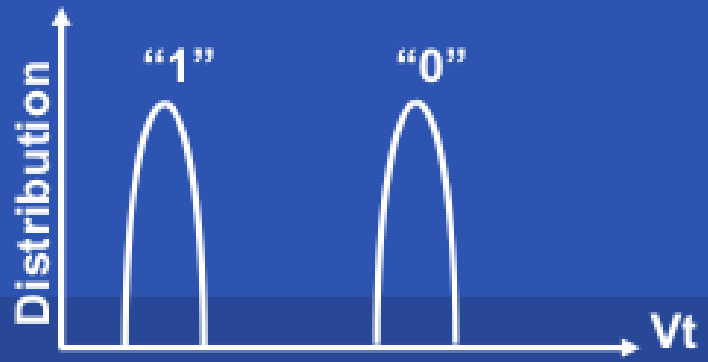
Erase

Programming means injecting electrons to the FG

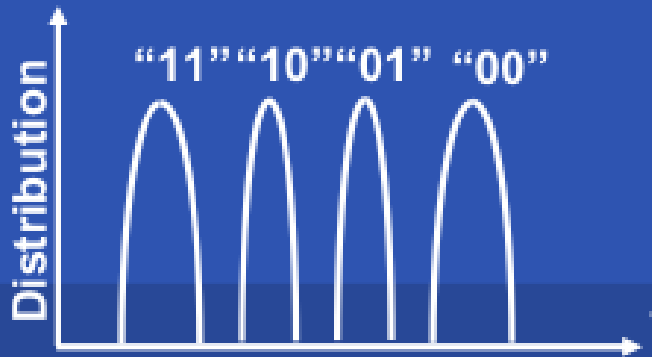
- Fowler-Nordheim Tunneling

Erase: Fowler-Nordheim Tunneling in reverse direction

2 Levels => 1 bit/cell

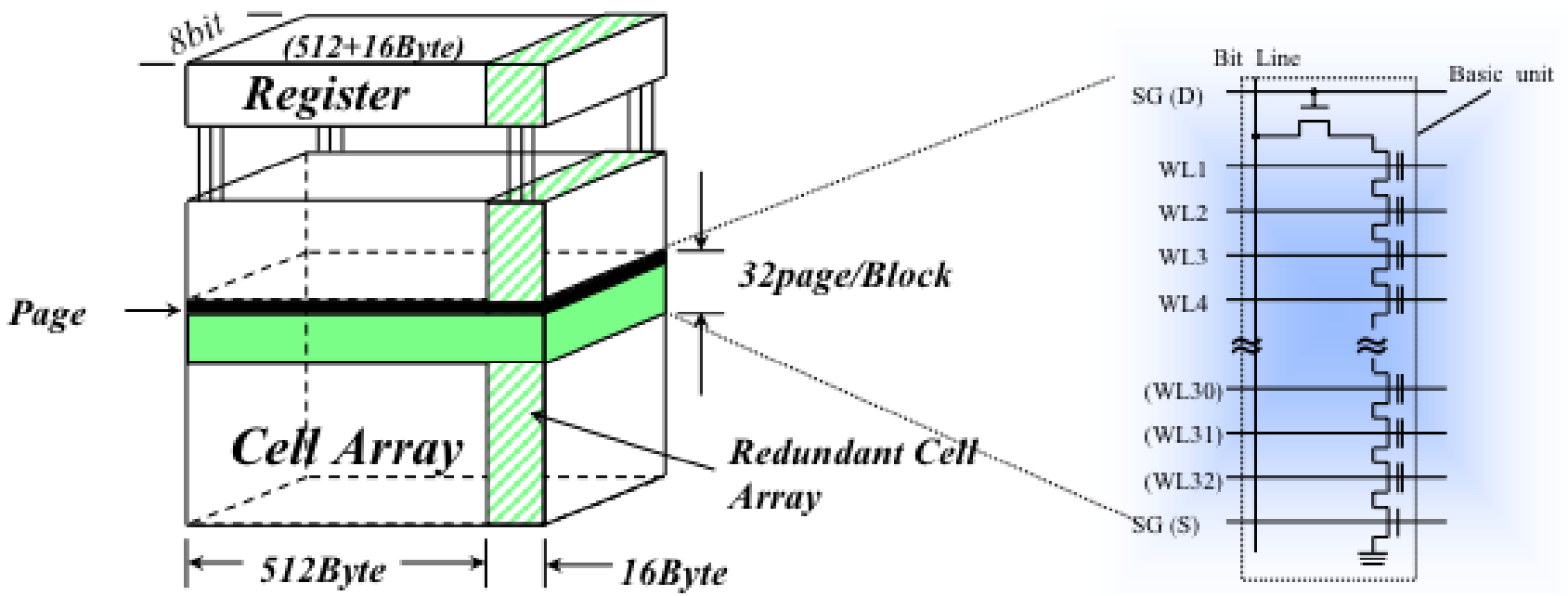


4 Levels => 2 bit/cell



NAND Flash Memory Block Diagram

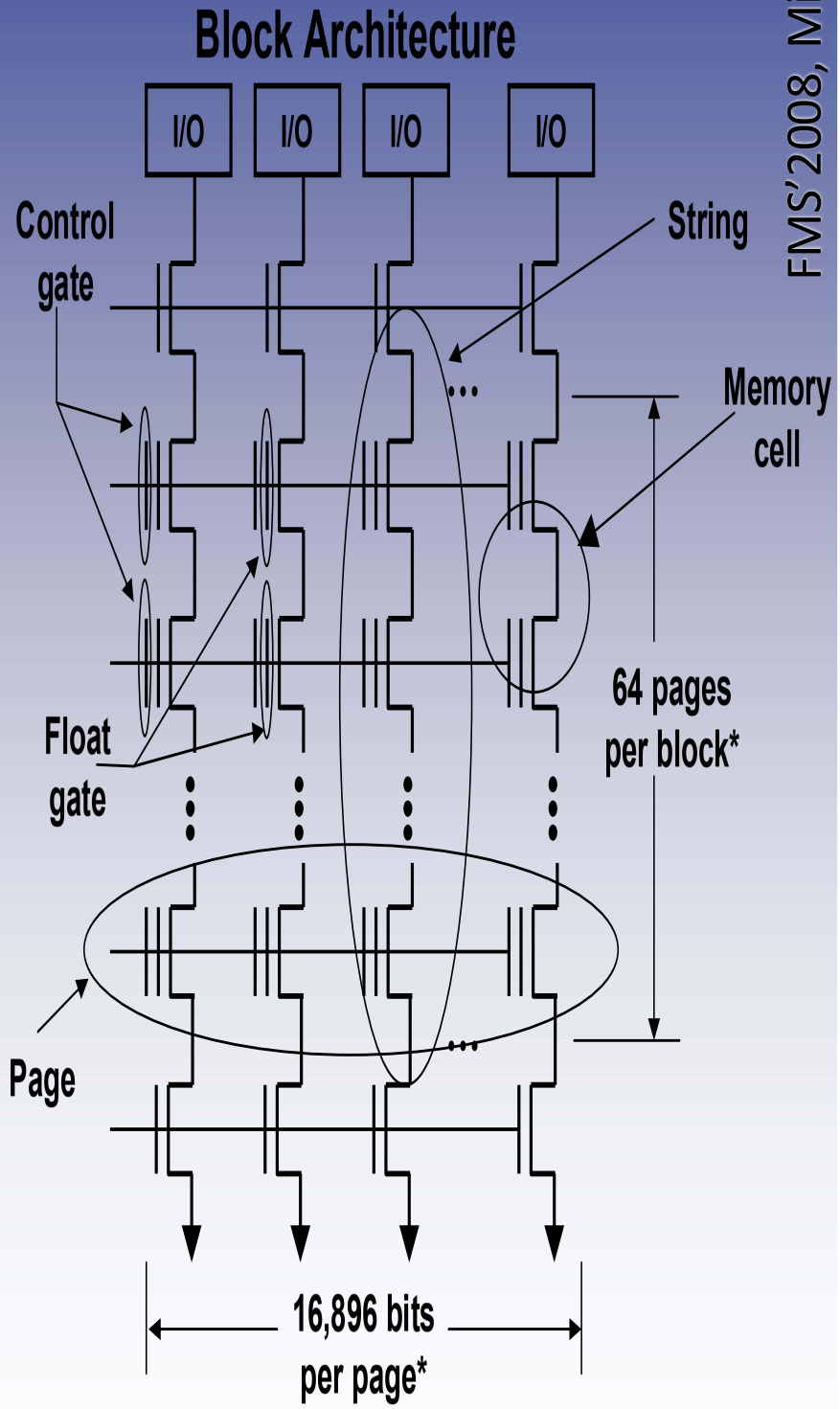
ex.256Mb NAND Flash Memory



256Mb NAND Flash
- Page Size : 512+16 Bytes
- Block Size : 16KBytes
- # of Blocks : 2048 Blocks

Let's Get Orientated

- NAND architecture is based on independent *blocks*
- Blocks are the smallest erasable units
- Pages are the smallest programmable units
  - Partial pages can be programmed in some devices



* Typical for 4Gb SLC

NAND Operations

- Read
  - Setup time to “reach” a page within block (sequential)
  - Read a full page at a time
  - NOR can be read “truly” randomly
- Program (1 -> 0)
  - Setup time to “reach” a page within block (sequential)
  - Program a full page at a time, set independent cell bits to 0
  - NOR can be programmed randomly but each word separate (therefore a page takes longer to program)
- Erase (0 ->1)
  - Can only erase a full block (physical organization and method)
  - Different methods to erase NAND/NOR

TOSHIBA

NAND / NOR Characteristics

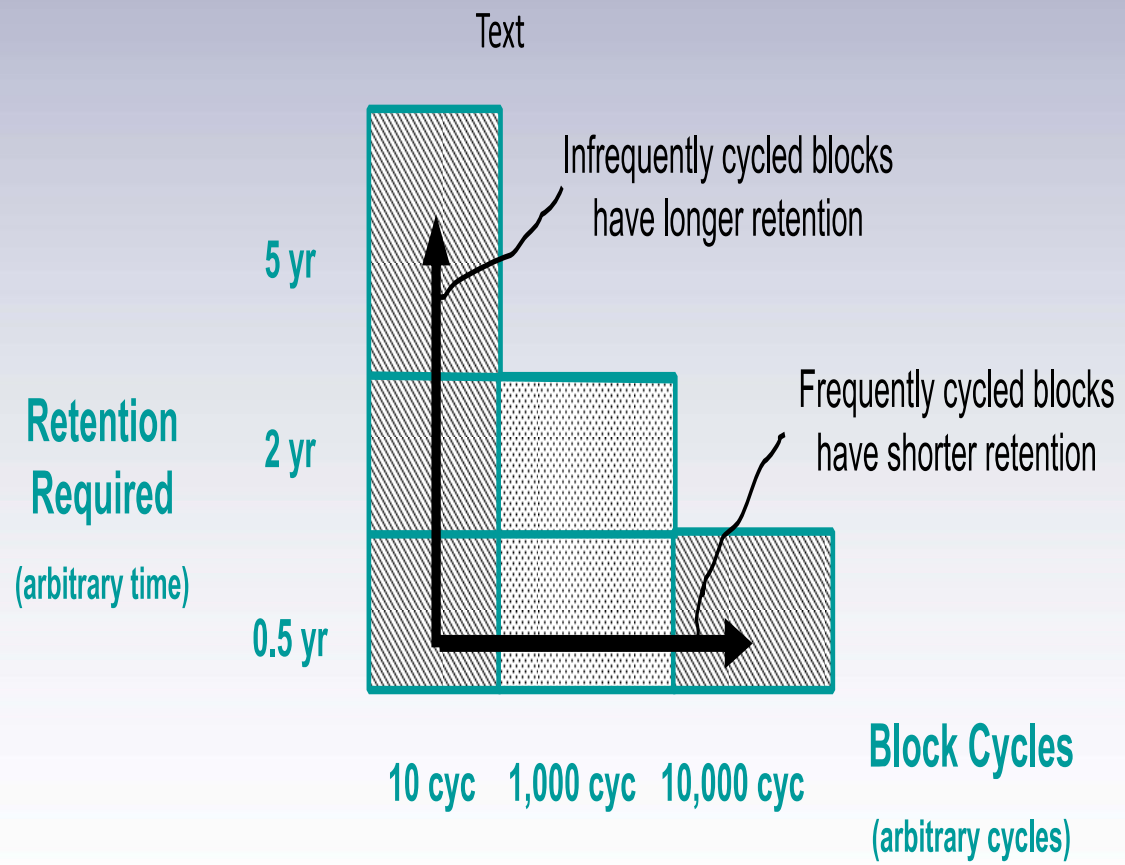
	NAND	NOR
Capacity	~ 1Gbit (2chips/pkg)	~ 128Mbit
Power Supply	2.7-3.6V	2.3-3.6V
I/O	x8	x8/x16
Access Time	50ns(serial access cycle) 25μs(random access)	70ns(30pF, 2.3V) 65ns(30pF, 2.7V)
Program Speed (typ.)	———	8μs/Byte
	200μs/512Byte	4.1ms/512Byte
Erase Speed(typ.)	2ms/Block (16KB)	700ms/Block
Prog+Erase(typ.)	33.6ms / 64KB	1.23s/Block (main:64KB)

Reliability - Data retention

- Three main types of problems
- 1. Oxide degradation over time (data retention)
  - No accesses – FLASH on the shelf
  - Oxide degradation leads to bit errors
  - Unused FLASH, typical retention time 10 years
  - BUT depends on how much the material has been used (erase/prog cycles)
- 2. Oxide degradation due to erase/write
  - Main problem so far
  - Between 10,000 and 100,000 erase cycles
  - Wear leveling used to compensate (will discuss)
- 3. Bit errors due to reading neighboring cells
  - Read disturbance
  - Until now smaller problem, but starting to be relevant
  - Depends on how much the material has been used (erase/prog cycles)
- Read/program/erase process is electrically complex
  - Includes various mechanisms
  - Each mechanism with its own sources of error
- (NOR does not suffer from these problems)

Improving Data Retention

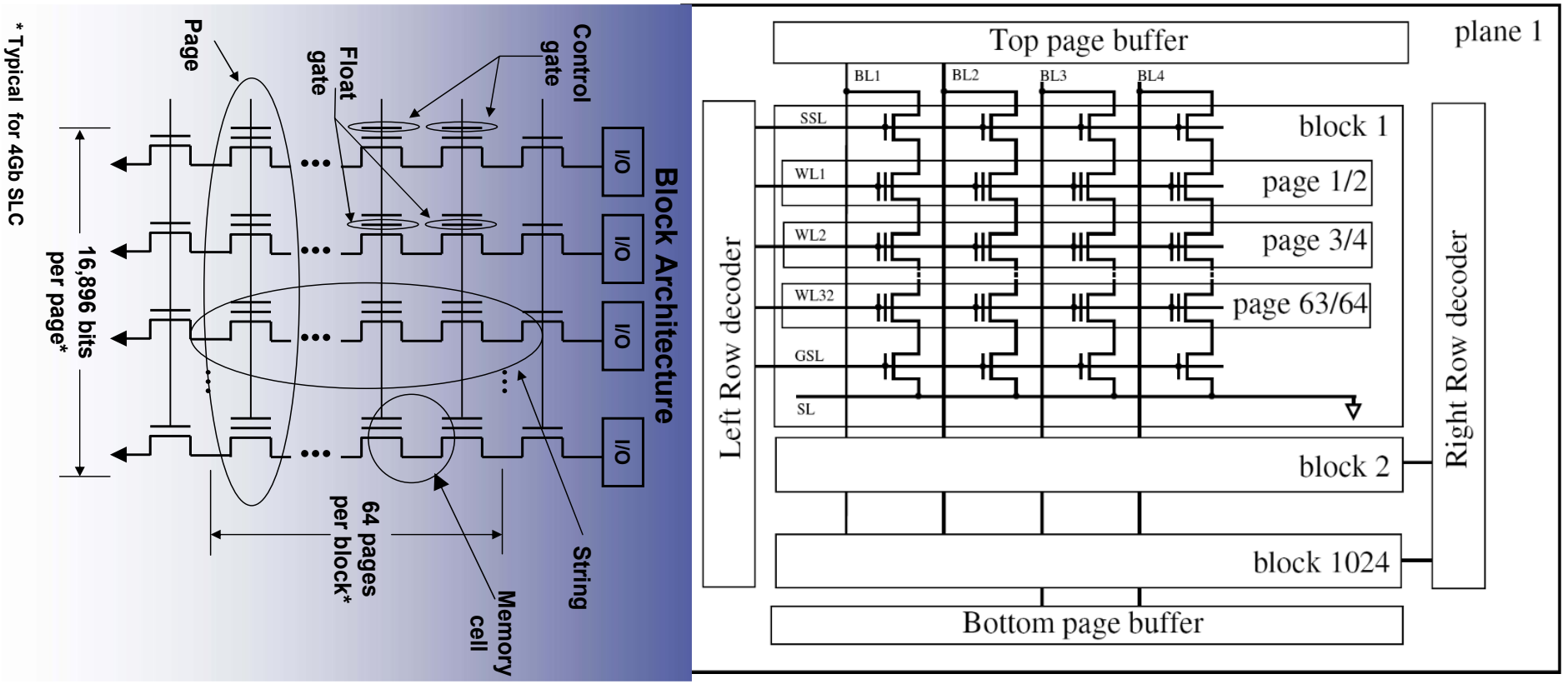
- Limit PROGRAM/ERASE cycles in blocks that require long retention
- Limit READs to reduce read disturb
- Review JEDEC (JESD47) standard



FMS'2008, Micron Inc.

NAND Physical Organization (2D)

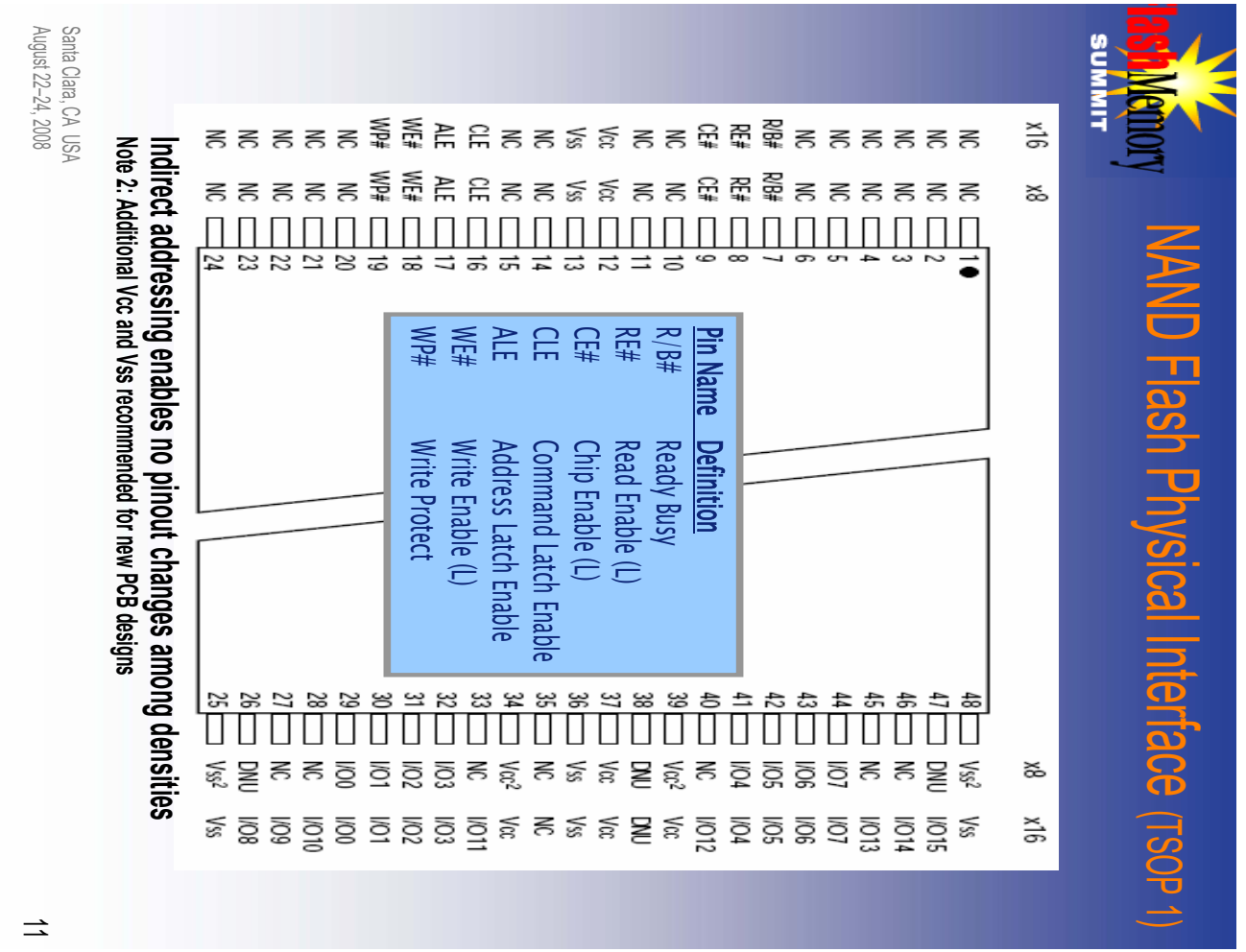
[Micron TN-29-28: Memory Management in NAND Flash Arrays]



- Send a command, address, (data)
- Each command takes time
  - µs for read/program, ms for erase
- Receive data or other result back
- Interface similar to memory (simple, parallel)

Flash Array Pins (Micron)

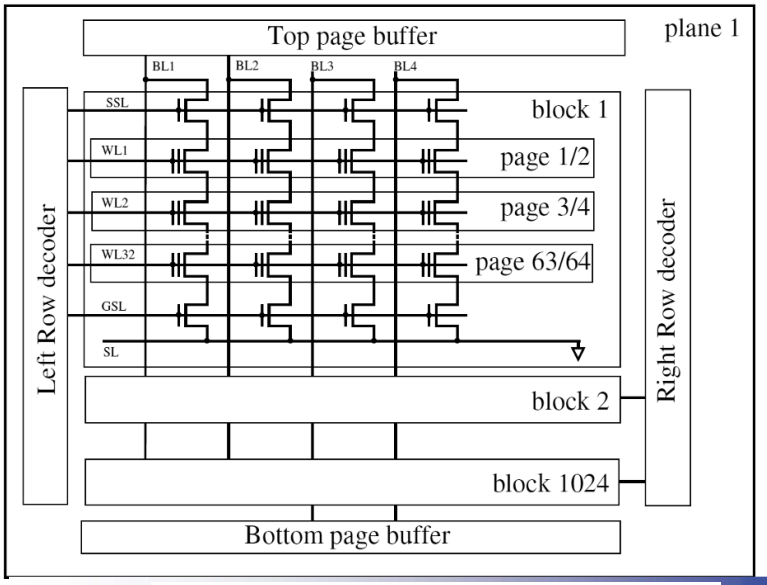
[Introduction to Flash Memory (T1A). Jim Cooke, Micron Technology, Inc. Flash Memory Summit. Santa Clara, CA USA. August 22 –24, 2008]

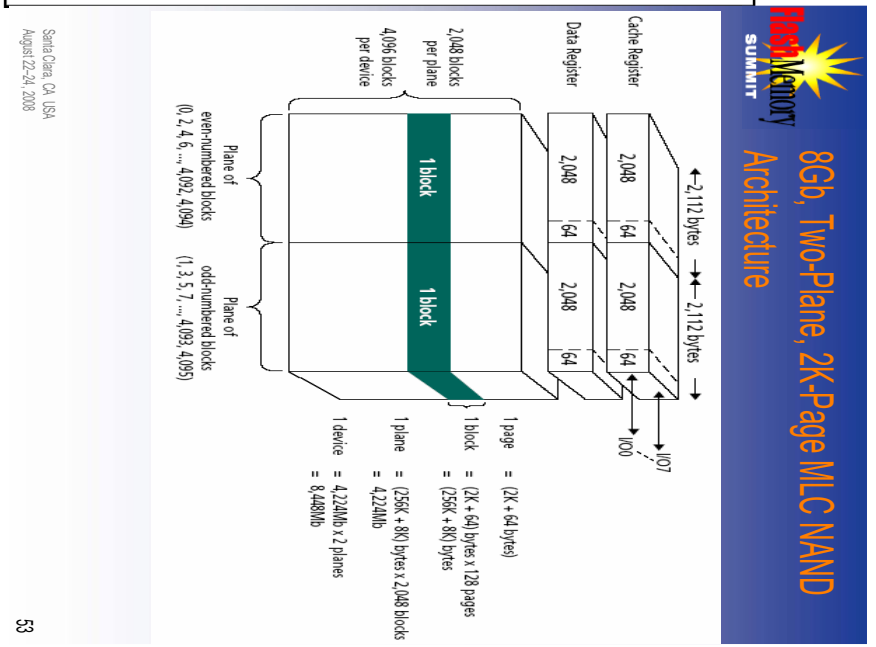


- Number of pins affects cost and connecting many chips to the FLASH controller
  - 7 pins for control/command
  - 8 or 16 IO pins for address + data (multiplexed)
  - 15 or 23 pins for 8-bit (16-bit) wide arrays (7+8 or 7+16)
- Operations take time (µs-ms) → Can we use available pins better? → "Planes"

Flash Planes: Internal parallelism

- Consider a single flash array
  - Externally it has a single set of pins
- (a) Larger pages, larger blocks
  - Some form of parallelism (but requires locality)
- (b) Operation parallelism (planes)
  - Create two sets of blocks (left and right)
    - Can have more than two planes per array
  - Two decoders
  - Ability to handle (latch) two addresses & commands
- What does this achieve?
  - Perform same command, on multiple pages/blocks
  - BUT transfer time still proportional to data because of shared I/O pins
  - Overlap execution time of the command on each block/page but not data transfer
  - Increases parallelism without increasing # pins





Overlapping Operations in Planes

- E.g. 4KB pages: Read(R)=-, Write(W)= ----, Transfer(T)=--
- RRRR
  - Without planes
    - RTTRTTRTTRTT
    - 12 time units
  - With planes
    - RTT
    - R TT
    - R TT
    - R TT
    - 9 time units (30% improvement in read throughput e.g. 30 MB/s -> 40 MB/s)
- WWWW
  - Without planes
    - WWWWTTWWWWTTWWWWTTWWWWTT
    - 24 time units
  - With planes
    - WWWWTT
    - WWWW TT
    - WWWW TT
    - WWWW TT
    - 12 time units (100% improvement in throughput, e.g. 10 MB/s -> 20 MB/s)
- Benefits higher for slower ops to flash array (program, erase) compared to transfer
- Assuming enough concurrent ops, limit is I/O throughput from flash array to controller

Flash Package

[Design Tradeoffs for SSD Performance Agrawal et al. ATC'08]

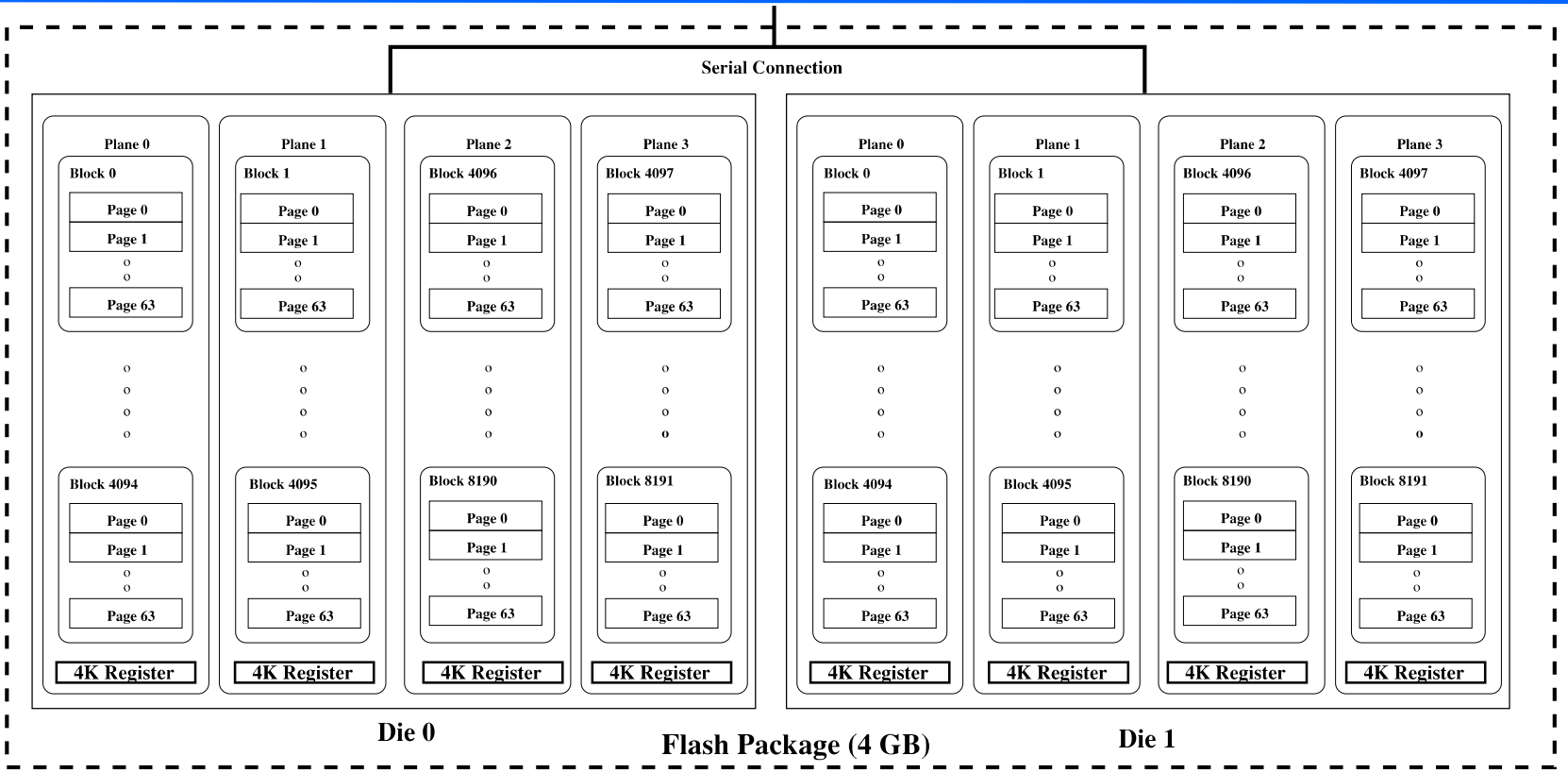
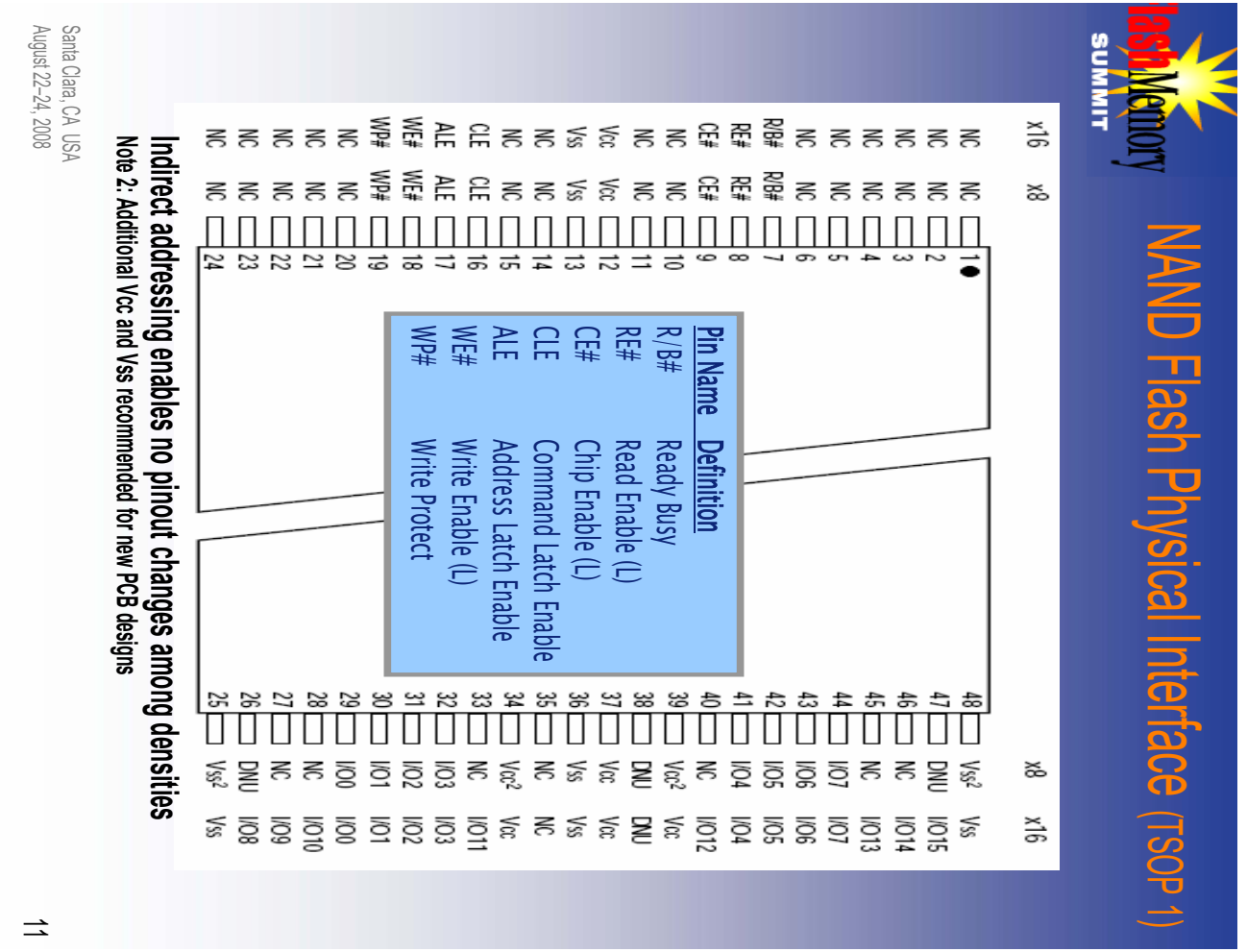


Figure 1: Samsung 4GB flash internals

- Multiple plane pairs (flash arrays) within each die, multiple dies per flash package
  - Increases capacity + may increase parallelism
- Throughput eventually limited by serial interface (channel) to controller
  - 100 MHz serial interface => 1KB/10µs or 4KB/40µs
  - Flash delay for **read** is typically similar/lower (program and erase have higher flash array delays)
  - Synchronous I/O interfaces to increase speed and parallelism (ONFi Open NAND FLASH Interface)
- Serial connection from controller to package matters, for sending different commands to different dies/arrays

We still have the same interface: Flash Array Pins (Micron)

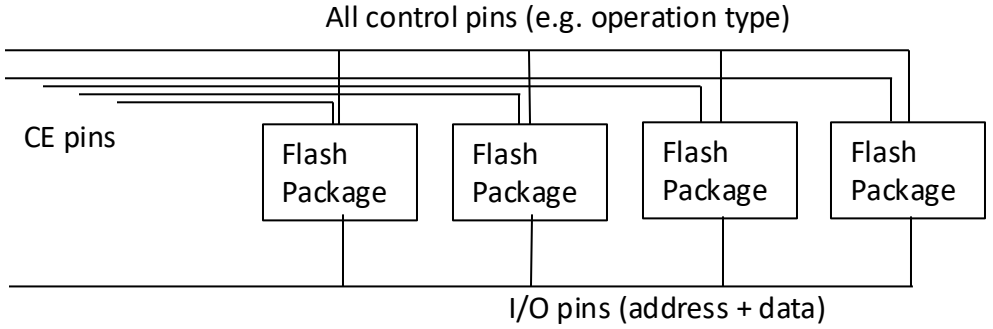
[Introduction to Flash Memory (T1A). Jim Cooke, Micron Technology, Inc. Flash Memory Summit. Santa Clara, CA USA. August 22 –24, 2008]

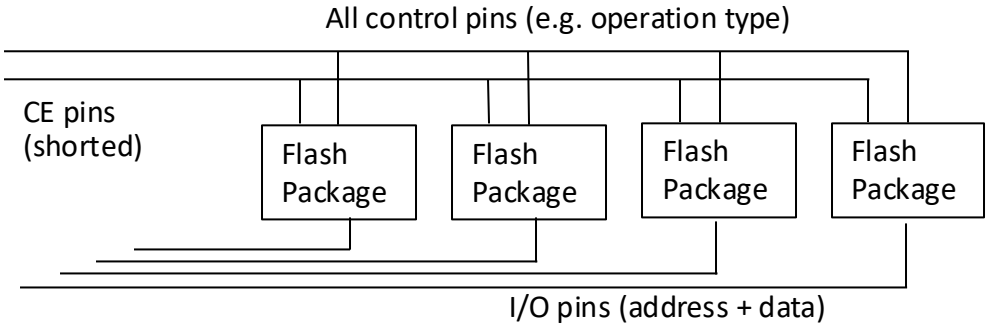


- Send on control lines: type of operation, e.g. read, latch addr, timing signals
- Send on address lines: addr, addr, … (to different planes)
- Wait for results
- Read on data lines: data, data, …

Multiple chips in a device

- How do we connect multiple chips in an SSD, e.g. to increase capacity
- Need wires for each chip, or can they share? How?



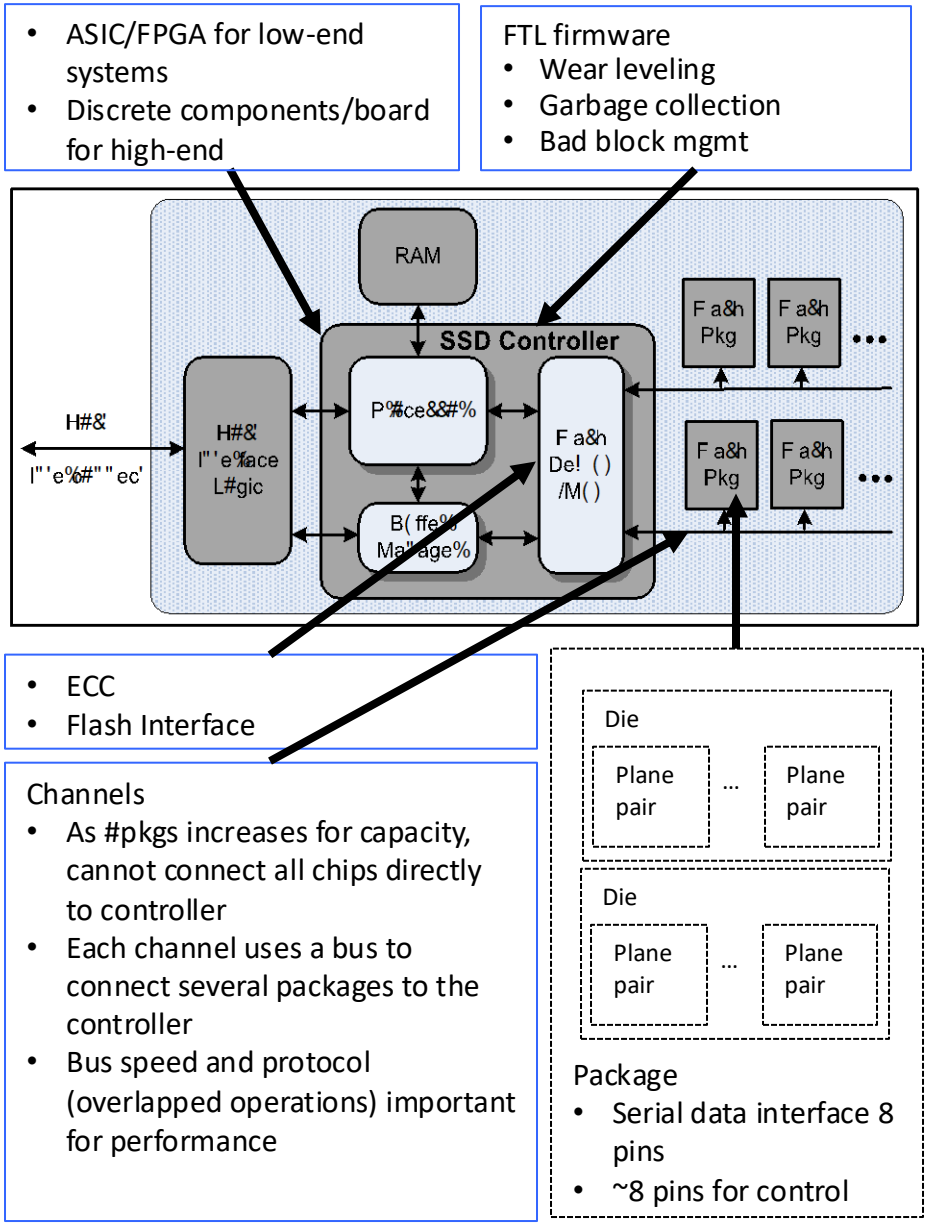


- Too expensive to have separate wires per chip and does not scale with the number of chips
  - Wires need to connect to a controller that has limited pins too
- What makes sense to share? Control pins, address/data pins, both?
- Shared everything: control, address/data pins
  - Execute the same commands to the same addresses
  - Transfers are serialized too
  - Limits our options for parallelism
- Shared address/data – but not control
  - Execute different command to different address(es)
  - Transfers are still serialized

SSD Organization

[Design Tradeoffs for SSD Performance. Agrawal et al. ATC'08]

- Concurrency and capacity (2021-22)
  - Multiple outstanding commands per channel, package, dir, plane pair
- Each 3D-NAND die
  - 512 Gb die, TLC (3b/c): 176 layers, 10 Gbits/mm2 , 48 mm2/die
  - 1 Tbit die, QLC (4b/c): 144 layers, 14 Gbits/mm2, 74 mm2/die
- Several plane pairs per die (~1 Tbit)
- Several dies per package (~1 TBy)
- Several packages per channel (2-4 TBy)
- Several channels per device (8-16 TBy)



Today

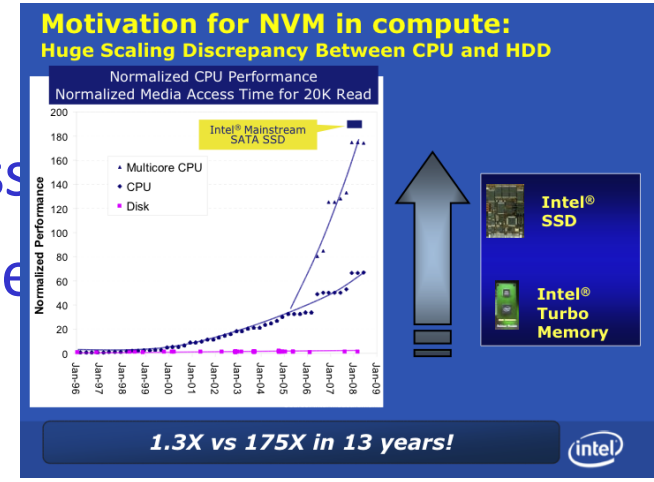
- A lot of spatial parallelism and concurrency internally
  - Placement of operations/data to improve I/O performance
- Unclear how to optimize concurrent internal ops
  - E.g. spread out mutually blocking operations
- Use lower-cost operations when possible
  - Eg some types of copying/moving faster than others

We went from Cells to SSDs: Done?

- We would like SSDs to replace HDDs and be done!
  - Generally, in I/O we consider
    - sequential vs. random I/Os
      - Typically, large I/Os are considered sequential (why?)
    - metrics: throughput vs. IOPS
- Easy use for SSDs – workloads with
  - Read requests only
  - Large writes only
  - Mobile (media) is a good match: few updates, large files
- Can we use SSDs for general-purpose servers?
  - Not straight-forward
  - Small, random writes are important
  - Reliability is also important

Potential

- Today: interest in I/O
  - (a) Most computation today is data process
  - (b) Computation continues to increase (doe
- HDD is 100 IOPS and 100 MBy/s
- SSD is 100,000 IOPS and 1000 MBy/s
- Does it matter?
  - Could always use many disks – parallelism
- How many HDDs to feed a server?
  - x1000/server?
  - Profound role of locality, reuse, working sets, etc…

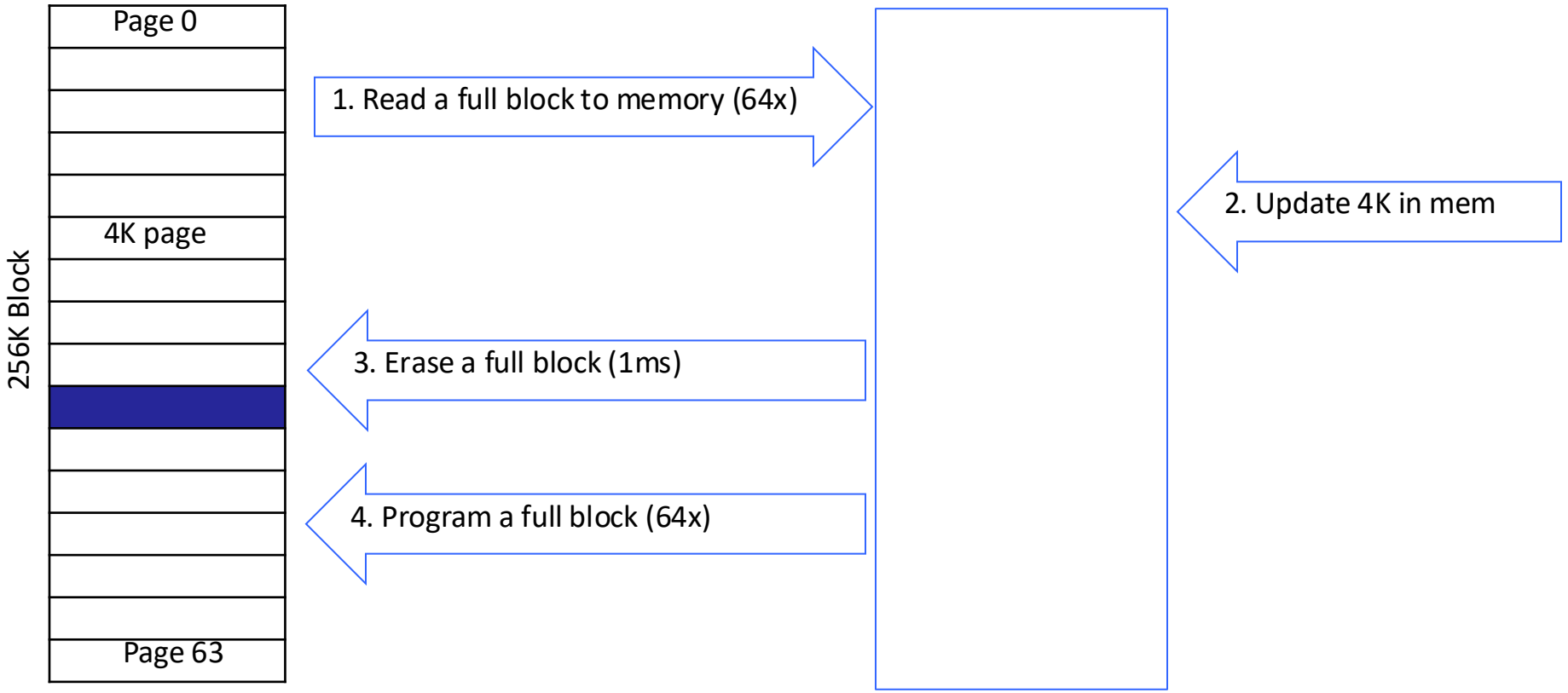


Flash Translation Layer (FTL)

I/O writes in SSDs

- Let's try to do a small write – where small = 4K
- Let's say this is an update (not first write)
- What could we do?
- Find physical 4K page containing block
- Cannot write page directly – need to erase block first
- Read block to memory (read)
- Update page (modify)
- Erase block and program full block (write)
- This is a lot of overhead!

I/O writes in SSDs



- Traffic 128x more
- Erase on critical path
- Read-modify-write does not work (write amplification)

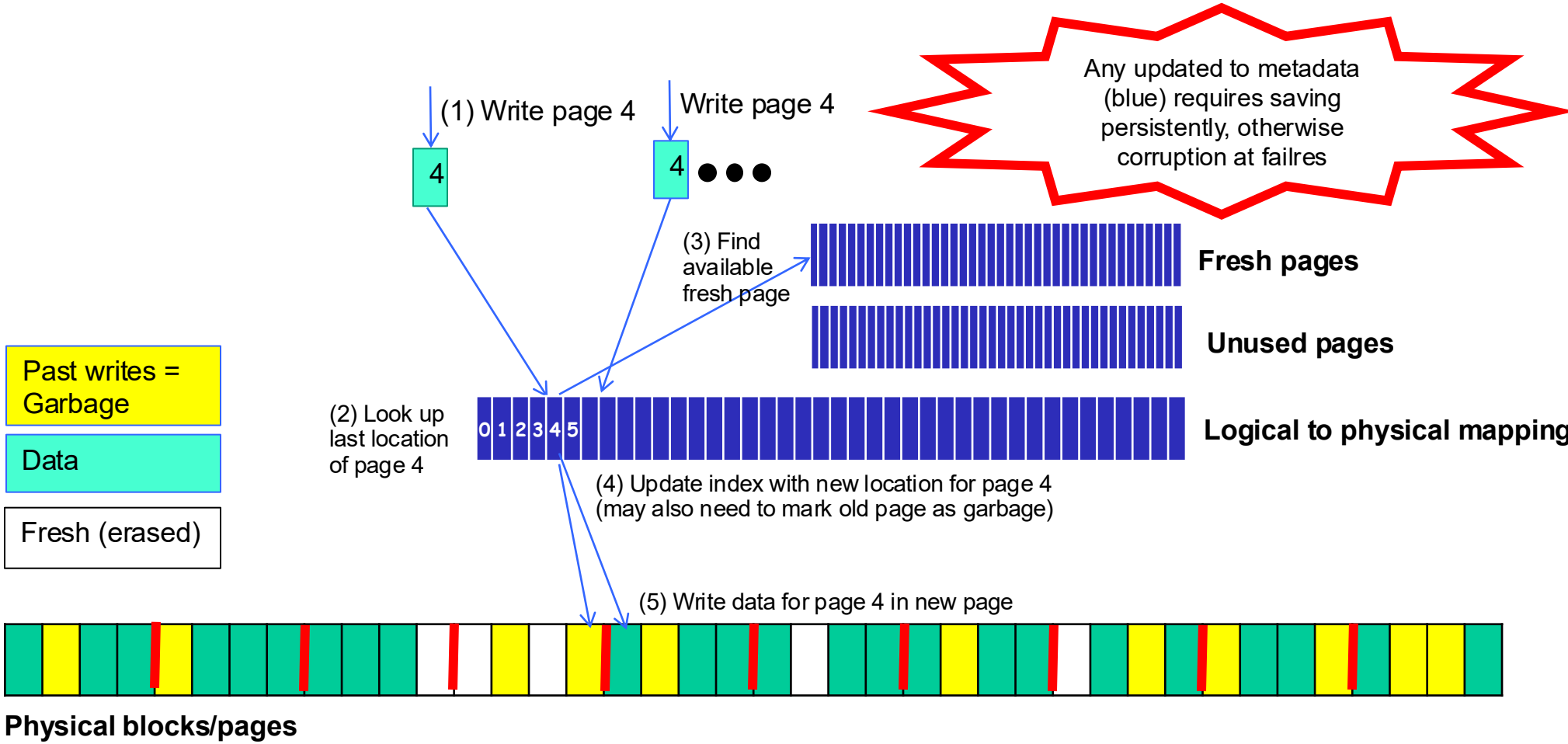
The problem

- Different erase block vs. page sizes
  - E.g. erase block size 256K - erases
  - E.g. page size 4K – updates
- Why? Large blocks => higher density for NAND

Let's rethink

- We cannot overwrite the same page, unless we erase
- Fine, let's find another page in a block that has been fresh erased
- Write 4K to this page
- Remap page to new physical page
- Free old page
- Essentially, introduce a logical-physical page mapping
  - First time ever in a storage device for all blocks
  - Requires a lot and non-trivial metadata – like a file system

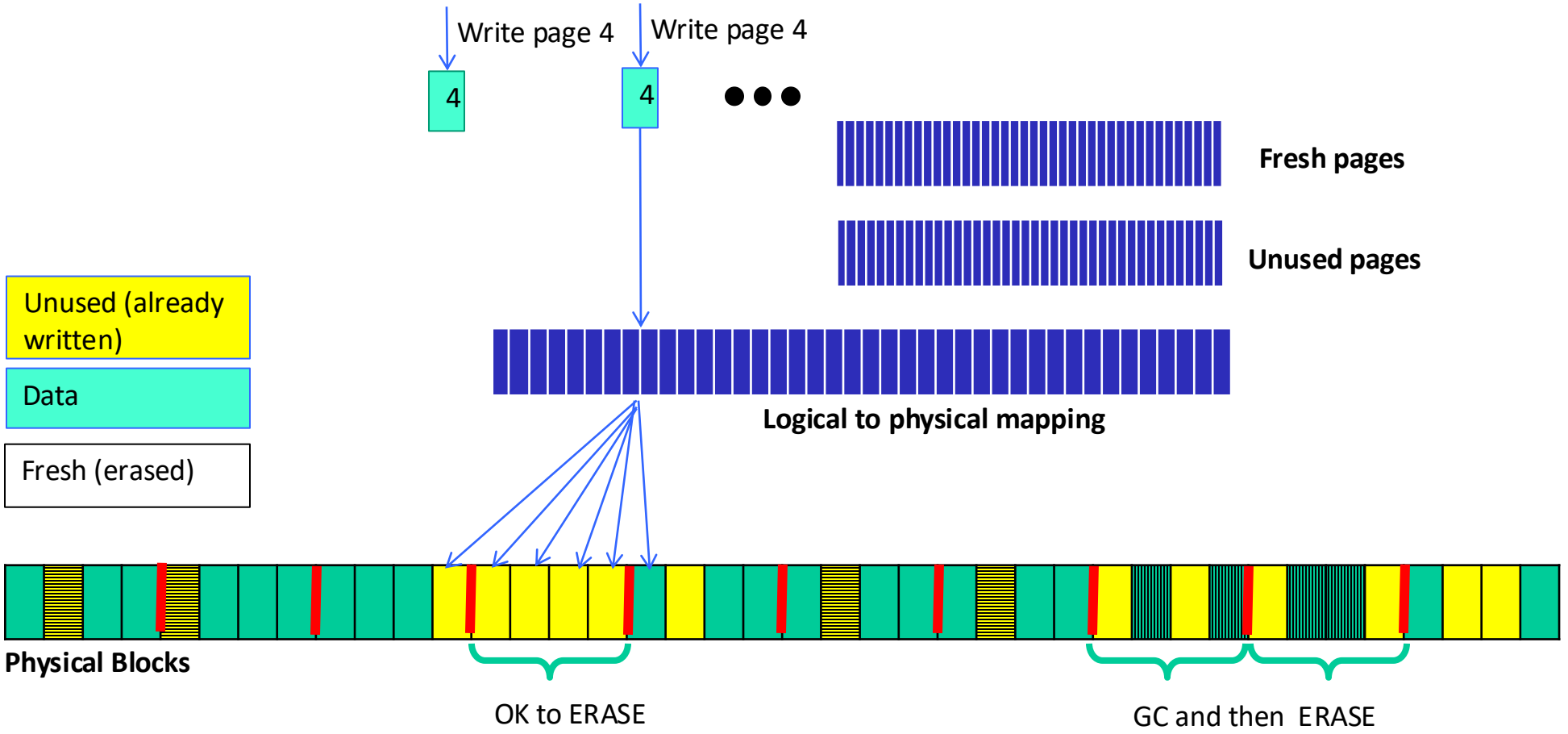
Logical to physical mapping



After a while…

- We end up with
- Erased blocks that were empty will fill up
- Pages that were re-written will be free leaving holes in various blocks here and there
- At a next write, there will be no location to put the data
  - Cannot write a “hole”, need to erase first
  - Cannot erase a “hole”, need to erase the full block

Create “fresh” pages

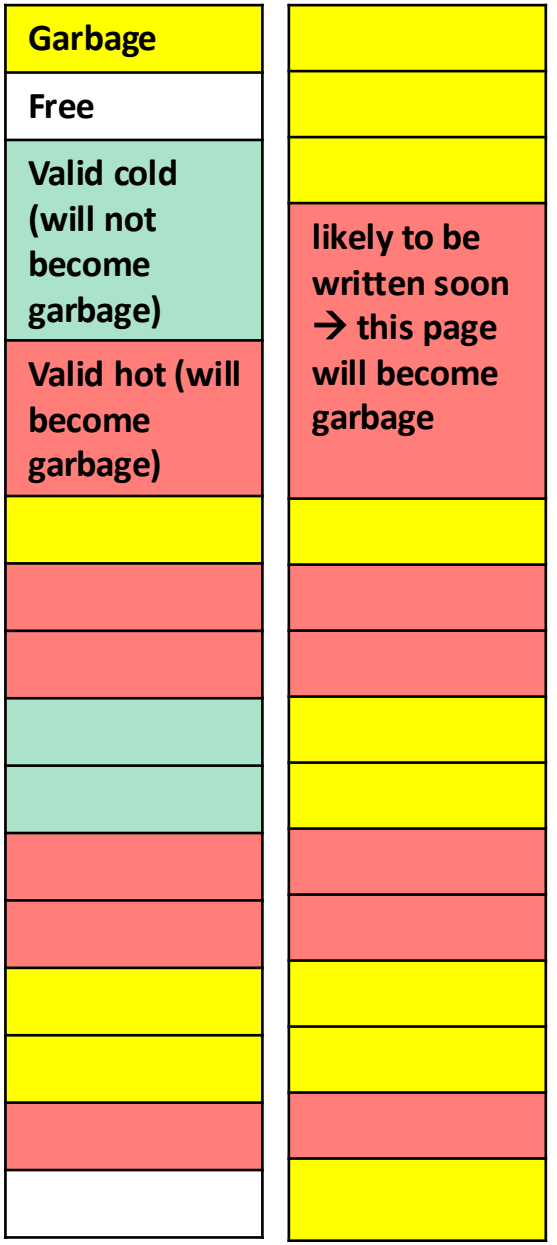


Garbage Collection

- So at a next write, need to erase a block – how?
  - Then place pages in fresh list
- A) If block fully garbage, easy, just erase
- B) If block data + garbage
  - Move data in other block → how?
  - **Pick a two (or more) blocks that have garbage**
  - **Copy their data in a spare block → need one (a few) spare blocks**
  - **Update physical to logical page mapping**
  - Both (or more) blocks become garbage → put back one spare block
  - Erase the rest (at least one) and place in fresh pages
- (B) is garbage collection in FLASH – (A) is just a simple case
- Requires
  - Read several pages from various erase blocks
  - Write these pages elsewhere (some form of compaction)
  - Update metadata (mappings) and worry about failures
  - Replenish fresh pages (and spare blocks)
- All this while performing I/O writes (and reads)
  - High overhead
  - Unpredictable

Garbage Collection – Block selection

- To reduce GC overhead
- Generally, a block contains pages with
  - Garbage: This page has been updated and is elsewhere
  - Valid data: Contains data and may or may not be updated
  - Free space: Pages that have not been programmed after erase
- When selecting a block to erase
  - Choose blocks with garbage only – simple, but how often?
  - Avoid block with free pages -> wasted potential
  - Prefer blocks with a lot of garbage and little valid data **-> Do they exist?**
- How do you create blocks with a lot of garbage?
  - Important: "hot" vs. "cold" pages/data (many vs. few updates)
    - Hot pages (data) create garbage in blocks – this is good for GC
    - A block with only hot pages will become garbage soon
    - If all blocks contain few hot pages and many cold then all blocks will have a lot of valid pages to move during GC
  - Approach: categorize hot/cold blocks and use two block pools
    - When a page is written
    - Determine if this page is becoming hot or cold
    - Then place in a block from the respective pool
    - It helps if hot pool has few blocks, then less spread during writes (but less parallelism during write operation to update same block)
  - Various techniques to categorize pages as hot-cold
    - Bitmap based (bloom filters)
    - Queue based



Today

- Logical to physical translation with various table-based techniques at block, page, hybrid level
- Perform garbage collection asynchronously
- Reduce overhead of GC with hot-cold techniques
- Generally
  - A complex process
  - High overhead
  - Still unpredictable behavior
- FTL metadata for GC & WL

[Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. 2009. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. SIGPLAN Not. 44, 3 (March 2009), 229–240. https://doi.org/10.1145/1508284.1508271]

Wear Leveling

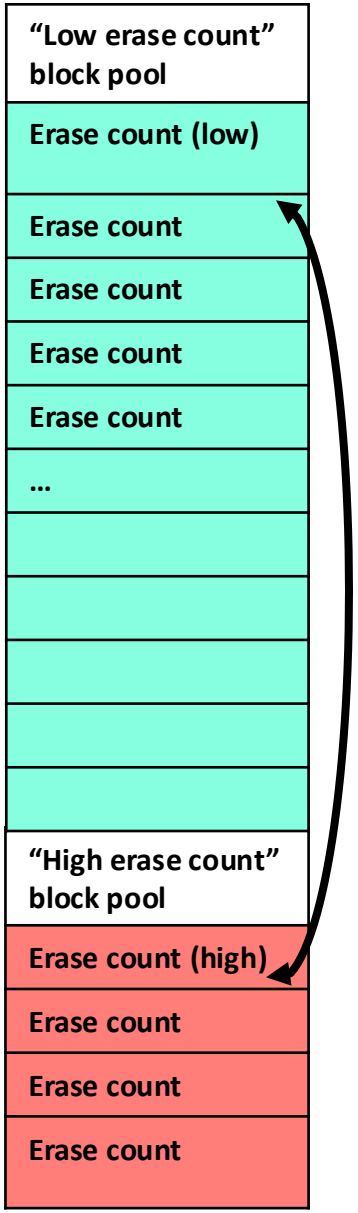
- Each block can be erased a number of times only
  - After that not possible to update/read
- If we are unlucky (and naive)
  - One block gets picked all the time to erase
  - After a number of erase cycles it goes bad
  - The SSD needs to reduce size
  - The filesystem (usually) does not handle this
  - Device is not usable
- What can we do
  - (1) Replace blocks
  - (2) Spread erases across blocks

Replace blocks

- No matter what, some blocks will go bad
- SSDs have some spare blocks internally (non-negligible percentage)
- When a block goes bad, the SSD detects this
- It remaps logical block to a new physical block – SSD size remains the same
- This block-remapping is present in disks as well
  - Not a common-path operation

Spread erase operations

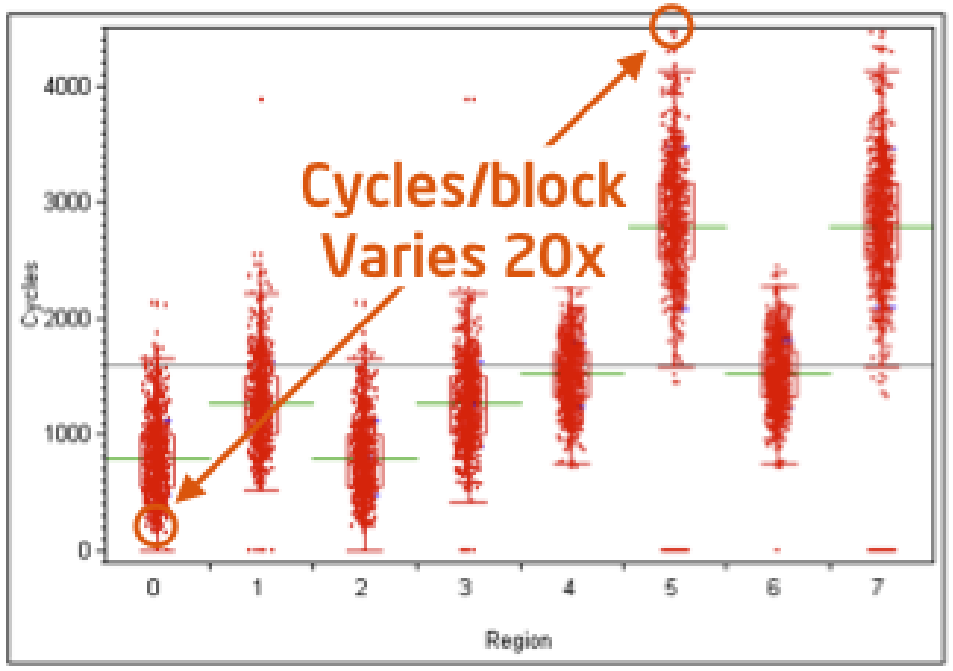
- Ensure that each block is erased about the same number of times
  - "Wear-leveling"
- Each block has metadata on how many times it has been erased
- Over time
  - Check erase counters of blocks
  - When they become un-even
    - E.g. when a cold block counter is lower than a hot block counter
  - **Exchange the roles of blocks → expensive**
    - Requires moving/copying pages
  - Next few erases will happen to the block with low erase count, while the block with the higher erase count will not see erase ops
- When need to GC and erase, pick blocks from low-count pool



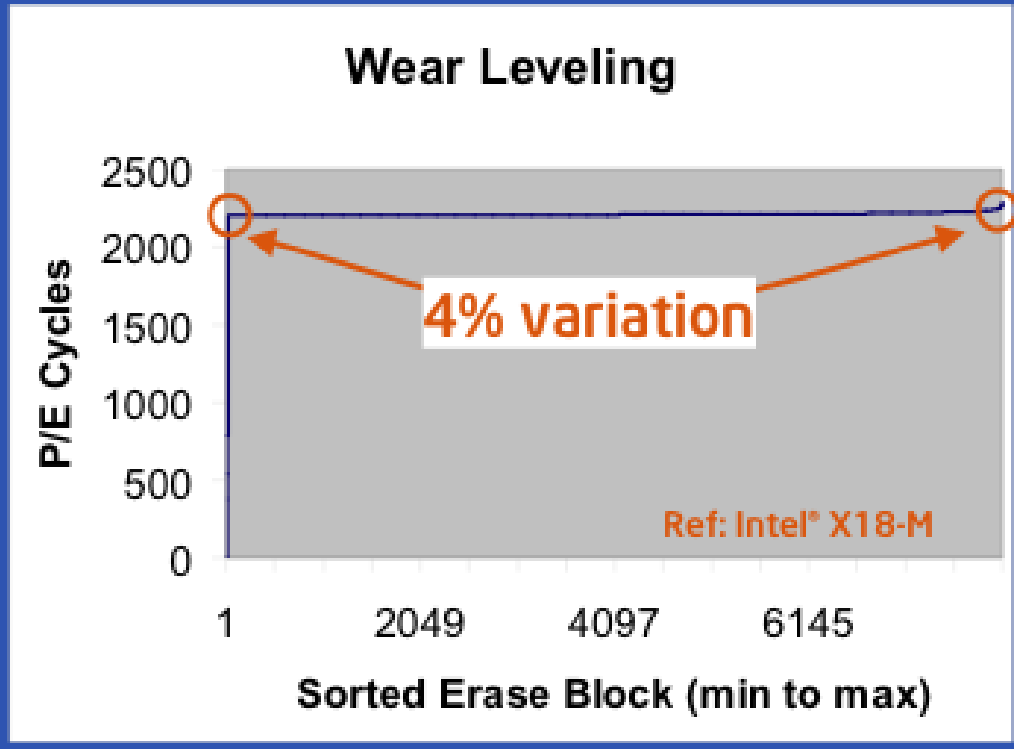
Example Tradeoffs

- Hot-cold data
  - => reduced overhead for GC
  - Separate hot from cold data in different blocks
  - Good for GC – select hot blocks for GC (more garbage)
  - (Less spread of hot data to hot blocks is even better for GC)
- Most writes will go to (few) blocks with hot data
- Erase counts will increase for these blocks (problem for wear)
  - Need to switch wear-leveling pools for these blocks
  - => high overhead for wear leveling
- Need to balance size of pools and how much writes are spread or concentrated

Variability in Wear Leveling



Unsophisticated regioned scheme



More sophisticated scheme

Controllers vary in in wear-leveling effectiveness

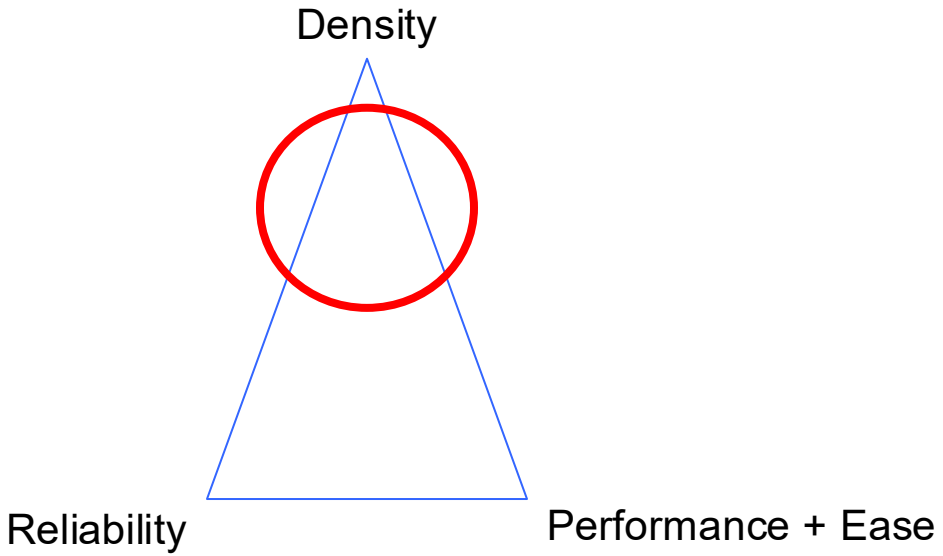
Poor wear leveling can have high impact

20x in cycles can be 10x or more in RBER

10x in RBER is 10^{ECC+1} in ECC failure rate: 100,000x for 4-bit ECC

Go back one step

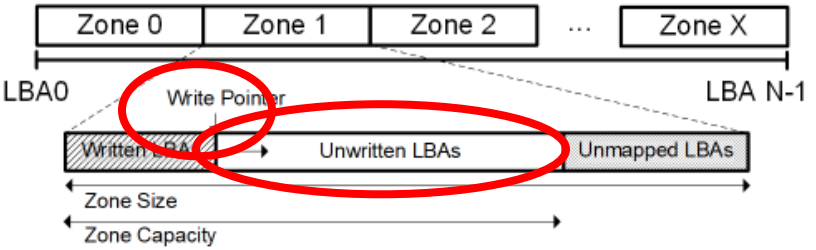
- Can we avoid this problem?
- (1) pages / erase block (# metal contacts) – hurts performance
  - E.g. Erase block size = Page size
  - Erase block size depends on technology and affects density
  - Density critical – probably THE driving force behind market size
- (2) bits / cell (multi-level) – hurts reliability
- Maybe there is room for different configurations



Zoned Namespaces (ZNS) SSDs

[Matias Bjørling et al. 2021. ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. USENIX ATC '21. https://www.usenix.org/system/files/atc21-bjorling.pdf]

- Previous work on Stream SSDs and Open Channel SSDs as well
- Provides "zones" of fixed size (each several blocks)
  - Hundreds of MBs – lower is better, but has to be feasible/practical
- Zones support only
  - READ
  - **PROGRAM_NEXT (APPEND)**,
  - RESET (ERASE)
  - **Cannot** PROGRAM any page
  - Several threads writing a zone, write one after the other in append-only manner (but failures can create holes)
  - A limited number of zones can be open (~8-32) for programming (constraint: per-zone resources in SSD for prorgam/erase)
  - Closed zones can still be read
- Zones fill up with writes/program operations
  - As page updates arrive
  - New page is appended to same or new zone, old page becomes garbage
  - Host/Application needs to remember pages that are garbage → Need for host-side FTL (but simpler)
- **Garbage collection**
  - **If app knows that all pages in zone are garbage, then RESET**
  - **If some pages are not garbage then zone cannot be erased**
  - **If all zones have valid data and no room, then app needs to GC zones into new ones and create new empty zones**
  - **Eliminates the need for garbage collection by the FTL**
- LSM
  - Assume SST = zone
  - At compaction create new zones and reset old ones -> no GC at device level
    - There is a question here for half full SSTs/zones
  - In reality SST < zone which means that SSTs need to be selected carefully for colocation in each zone
    - If successful, then zone will become garbage as a whole and quickly
    - Otherwise, zone will contain some valid data for much longer



”Sustainability”

[The promise of NVMe FDP in Data Center Sustainability. Roshan R. Nair and Arun George. SNIA Developer Conference, Sep. 16-18, 2024, Santa Clara, CA, USA.]

- SSDs contribute to carbon footprint a) during their fabrication and b) during their operation (energy consumption)
- In a datacenter you want to:
  - Minimize the number of SSDs (for given capacity of data)
    - Do not overprovision
  - Minimize the energy during operation
    - Do not garbage collect (GC is excess work)
  - Interestingly, these are conflicting goals
    - With overprovisioning we can reduce GC
- Techniques like zones (and FDP) can help reduce GC without overprovisioning
  - Fewer SSDs and less energy during operation

Putting it together: SSD Reliability Metrics

SSD UBER values can be << 10-15

UBER ∝ usage: program/erase/read & subsequent retention

Intel® X18-M and X25-M Mainstream SATA SSD (80GB)

- 10 Channels Architecture with 50nm MLC ONFI 1.0 NAND
- 5 years usage, 1000G, 1.2million hrs MTBF
- GB/day client workload @ 1e-15 UBER → >>100GB/day, 5 years

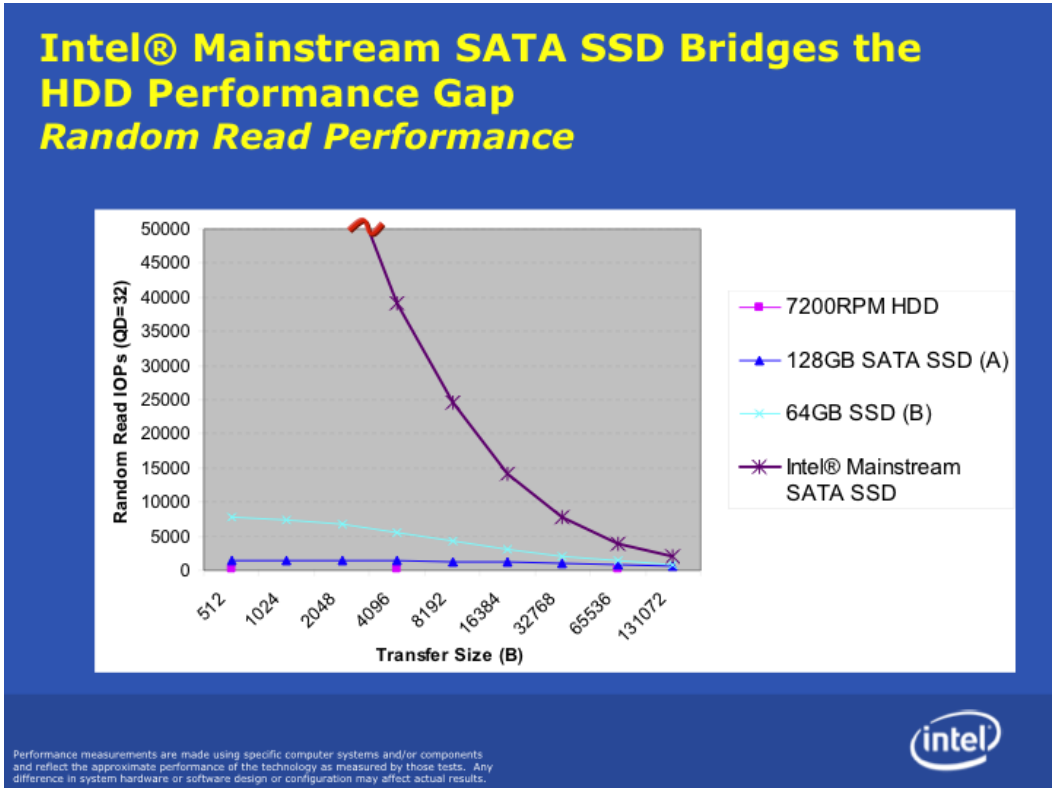
Intel® X25-M and X18-M Mainstream SATA SSDs deliver >5X accepted requirement for clients (20GB/day)

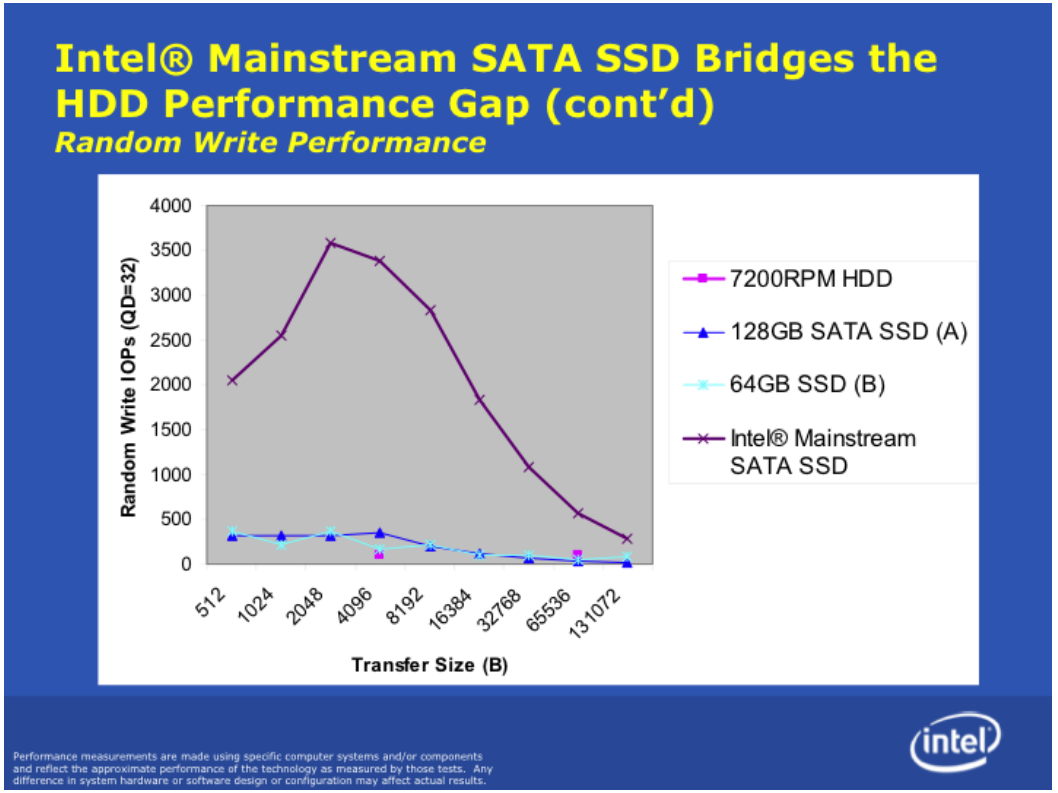
Intel® X25-E Extreme SATA SSD (32GB)

- 10 Channels Architecture with 50nm SLC ONFI 1.0 NAND
- 1000G, 2Million hrs MTBF
- Intel SLC SSD support > 7000 8K 2:1 R/W Random IOPs 24/7, 5 years

Intel X25-E SLC SSDs support the endurance required to replace many 15K RPM HDDs for IOPS applications

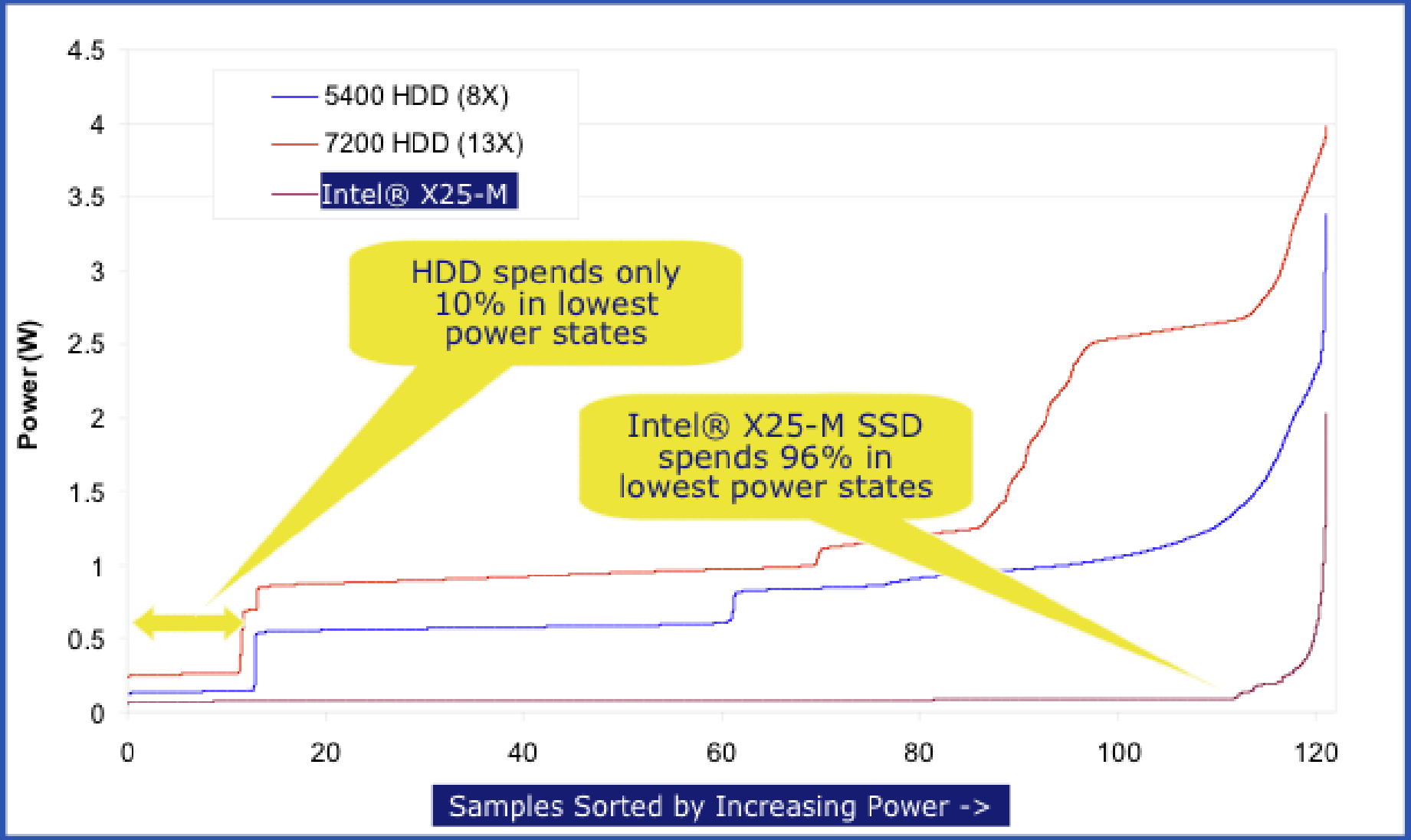
SSD Performance





- Performance variation a big problem, even today
- SSD controllers and host side become more complex
- SSD performance is not a "closed" issue

Intel® Mainstream SATA SSDs Save Power: SATA Power Rails With 2 Hour Mobile Workload

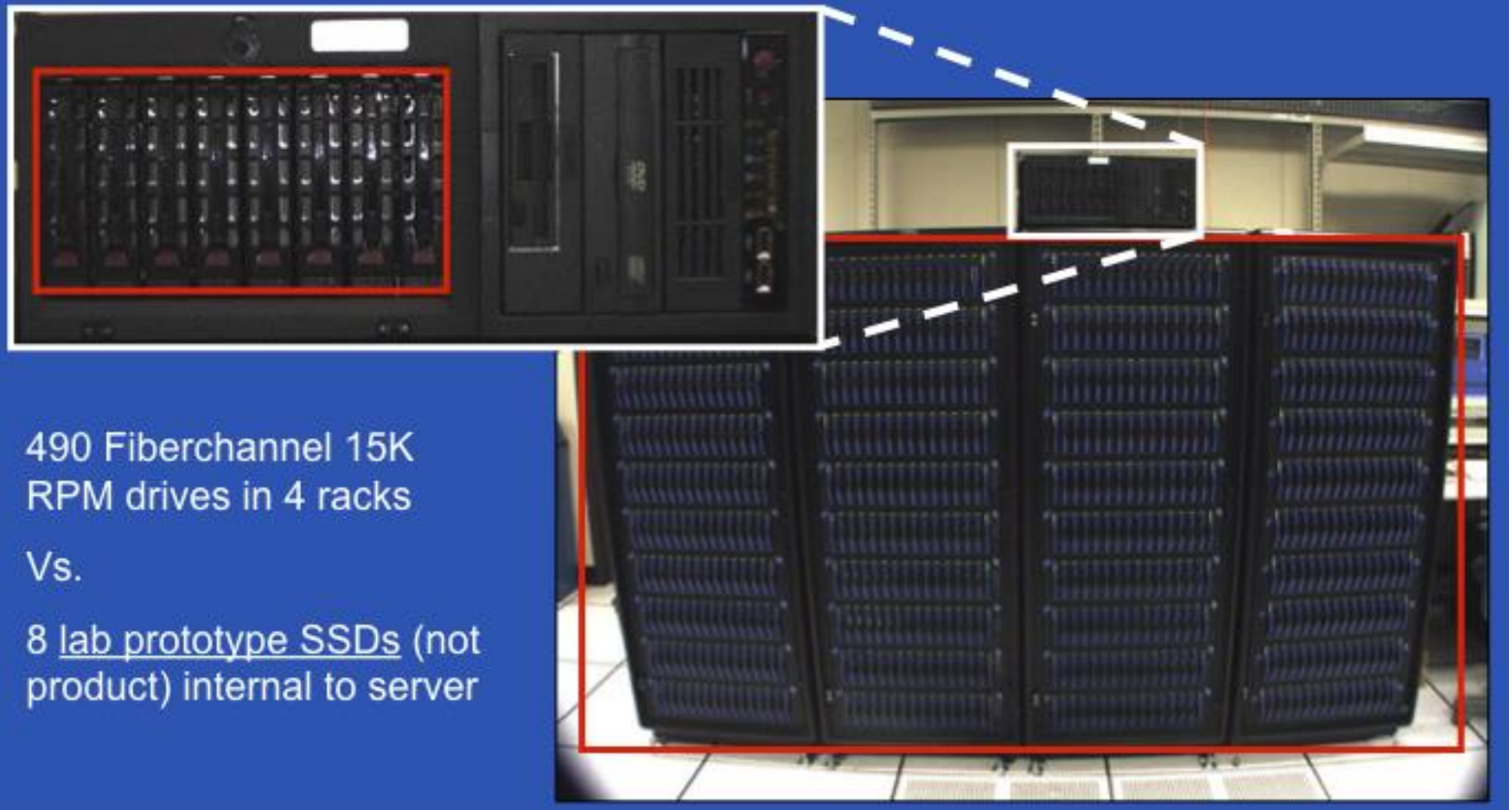


Performance measurements are made using specific computer systems and/or components and reflect the approximate performance of the technology as measured by those tests. Any difference in system hardware or software design or configuration may affect actual results.

What makes a good SSD

- SSD performance not always superior to hard disks (HDDs)
  - Request size matters
  - Request type matters
  - Access pattern matters
  - An HDD raid can have better price/performance than an SSD
    - Certainly cost/bit (Question: will this be the case in a few years?)
- A good SSD
  - I/O concurrency
  - Garbage Collection
  - Wear leveling
- Still room for improvement

Similar I/O Performance For IOPS Intensive Workload



490 Fiberchannel 15K RPM drives in 4 racks

Vs.

8 lab prototype SSDs (not product) internal to server

Unlikely to happen…

- Flash will not replace disks soon
  - Disk lowest cost/Gbyte
  - Trends not clear – not necessarily in favor of Flash
  - Manufacturing capacity for disks high
  - Would require replacing disk manufacturing capacity with Flash manufacturing capacity – not viable economically
- More likely
  - Integrate both SSDs and HDDs in storage system
- "Deeper" storage hierarchy
  - DRAM → NVMe → SSD → HDD
  - Not straight-forward at all to organize, operate

Outline

- HDDs
  - Capacity, cost
  - Shingled Magnetic Recording
- NAND Flash
  - Flash cells and arrays: How it works, basic operations, errors
  - SSDs: Structure, concurrency
  - FTL: Garbage collection, wear leveling
- Non-volatile memories (NVM)
- Remarks

Non-volatile memory (NVM)

- What do we mean with "NVM"
- Terminology
  - Block-addressable NVM → NAND Flash (SSDs, NVMe devices)
  - Byte-addressable NVM → PCM, STT
- NAND Flash is a type of block-addressable NVM
  - Not accessible at word granularity (otherwise low density)
  - Accessible via the same interfaces as HDDs
  - NVMe is Flash on PCI – NVMe is a protocol, not a type of medium
  - NVMe controllers many differences to support high IOPs over PCI
- Byte-addressable NVM is emerging
  - Phase change memory (PCM)
  - Magnetic RAM, variant Spin Transfer Torque (STT-RAM)
  - Today, path to scaling density and who will win not clear

NVM Technologies

- Different technologies for NVM exist
  - C. Xue et al. 2011. Emerging non-volatile memories: opportunities and challenges. 7th IEEE/ACM/IFIP International conference on Hardware/software codesign and system synthesis (CODES+ISSS '11). ACM, New York, NY, USA, 325–334. https://doi.org/10.1145/2039370.2039420
- Phase Change Memory (PCM)
  - Better scale-down (5nm?) compared to DRAM (22nm?) ➔ Higher density/capacity
  - Available in market
  - Concerns: write cost, endurance, energy
- Spin-transfer Torque Magnetic RAM (STT-RAM)
- Memristors

Attributes	DRAM	PCM	NAND
Non-Volatile	No	Yes	Yes
Erase Required	Bit	Bit	Block
Software	Simple	Simple	Complex
Power	~W/GB	100→500mW/die	~100mW/die
Write Bandwidth	~GB/s	1→100 + MB/s/die	10→100MB/s/die
Write Latency	~20-50ns	~1μs	~100μs
Write Energy	~0.1nJ/b	<1nJ/b	0.1-1nJ/b
Read Latency	50ns	50-100ns	10-25μs
Read Energy	~0.1nJ/b	<<1nJ/b	<<1nJ/b
Idle Power	~W/GB	<<0.1W	<<0.1W
Endurance	∞	10^8	10^5→10^4
Data Retention	ms	Not *f* (cycles)	*f* (cycles)

- NVM use
  - 1: More (but slower) memory
  - 2: Faster (but smaller) storage
  - 3: Persistent heaps
  - All interesting and being explored today
- Observe/keep in mind that today we treat memory (heap) and storage (files/kvs) as two separate functions/roles
- How to achieve better performance using NVDIMMs transparently
  - Q1: as regular memory
  - Q2: as (persistent) storage
  - Q3: as persistent memory

The future memory/storage hierarchy

- One memory to rule them all - Unlikely
- NVM to replace DRAM
  - DRAM is faster for writes
  - All NVM memories need to change state in a “permanent” manner for writes and this costs
- NVM/SSD to replace magnetic disks
  - Cost: Disk has lower cost/bit which makes it attractive
  - Errors: HDDs are better understood than SSDs (tapes are even better)
  - Manufacturing capacity: Not enough NVM/SSDs to replace disks
- Eventually, complex memory “hierarchy”
  - DRAM ⇔ NVM ⇔ Flash ⇔ HDDs
  - Multiple variations of each technology for different configs
  - Network too in the picture
- How do you organize such a hierarchy?
  - Nobody knows yet ☺
  - Not an easy problem – very complex interactions
  - There is potential for significant impact

Reading

- Review
  - Slides + pointers, You class notes
- Read
  - SMR hard drives
    - Shingled Magnetic Recording Areal Density Increase Requires New Data Management. T. FELDMAN AND G. GIBSON. Usenix ;login: JUNE 2013, VOL. 38 NO. 3 [https://www.usenix.org/system/files/login/articles/05_feldman_022-030_final.pdf]
    - G. Gibson and G. Ganger, "Principles of Operation for Shingled Disk Devices," Technical Report CMU-PDL- 11-107, Parallel Data Lab, Carnegie Mellon University, Pittsburgh, PA, 2011. [https://www.pdl.cmu.edu/PDL-FTP/Storage/CMU-PDL-11-107.pdf]
  - Flash basics
    - Micron Technical Note, NAND Flash 101: An Introduction to NAND Flash and How to Design It In to Your Next Product. 2006. [https://opencourses.uoc.gr/courses/pluginfile.php/13933/mod_resource/content/1/Micron%20Technical%20Note.pdf]
    - Introduction to Flash Memory (T1A). Jim Cooke, Micron Technology, Inc. Flash Memory Summit. Santa Clara, CA USA. August 22–24, 2008. [https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2008/20080813_T1A_Cooke.pdf]
    - Micron TN-29-28: Memory Management in NAND Flash Arrays Overview. Technical Note. 2005. Memory Management in NAND Flash Arrays. [https://www.micron.com/media/client/global/documents/products/technical-note/nand-flash/tn2928.pdf]
    - Section 2 of: Luo et al., Data Retention in MLC NAND Flash Memory: Characterization, Optimization, and Recovery, HPCA15 [https://www.cs.cmu.edu/~yixinluo/index_files/flash-memory-data-retention_hpca15.pdf]
  - SSD structure
    - Sections 1,2,3,5 of: Nitin Agrawal et al. Design tradeoffs for SSD performance. In USENIX 2008 Annual Technical Conference (ATC'08). [https://www.usenix.org/legacy/events/usenix08/tech/full_papers/agrawal/agrawal.pdf]
  - FTL address translation + GC
    - Section 2 of: DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings. Gupta et al. ASPLOS'09. [https://www.cse.psu.edu/~buu1/papers/ps/dftl-asplos09.pdf]
  - SSD DRAM cache
    - Section 3 of: BPLRU: A buffer management scheme for improving random writes in Flash storage. Kim et al. FAST'08. [https://www.usenix.org/legacy/event/fast08/tech/full_papers/kim/kim.pdf]
- Next class
  - Paper discussion

Intel Optane Persistent Memory Modules

[Intel® Optane™ DC Persistent. Memory. Quick Start Guide. June 2020. Revision 1.1. https://www.intel.com/content/dam/support/us/en/documents/memory-and-storage/data-center-persistent-mem/Intel-Optane-DC-Persistent-Memory-Quick-Start-Guide.pdf]

- DIMMs + NVDIMMs (technically DCPMMs)
- Rules / limitations **(need to verify)**
  - Each NVDIMM has to be configured at boot time for Memory Mode and/or AppDirect Mode
    - Each NVDIMM needs to be partitioned (at boot time / bios)
    - Any % (at 32GB grain) – 0% or 100% means use fully in Memory or AppDirect mode
  - Channels, NVDIMMs, and DIMMs
    - Each CPU socket has 2x memory controllers (MCs)
    - Each MC has 3x channels
    - Each channel has 2x slots
    - A channel can have max 2x DIMMs or 1x NVDIMM
  - An NVDIMM in Memory Mode can use as cache only the full DIMM in the same channel
- Q1: How do you place/partition NVDIMMs between modes?
  - Do you want more memory or faster storage in a server?
  - Static configuration significant limitation, apps/workloads are dynamic
  - Some sort of symmetric configuration is probably most reasonable (similar to DIMMs)
- Q2: What performance can you expect?
  - Highly dependent on configuration and workload
  - A lot of difficulty here, far from NVDIMMs being a drop-in replacement, e.g. placing more DIMMs in a socket can be great for workloads that need fast heaps and really poor for workloads that need large heaps, while heterogeneous setups usually incur low resource utilization (high cost)
- **Today: How to achieve better performance using NVDIMMs transparently**
  - **Q1: as regular memory**
  - **Q2: as (persistent) storage**
  - **Q3: as persistent memory**