Prepare For the Midterm Exam

- March 2nd, in-class, in-person, close-note exam.
- 75 minutes in total exam time.
- Bring your pen. Extra papers will be provided
- No calculator as it is not needed.
- The exam will cover all the contents we have discussed before the exam.

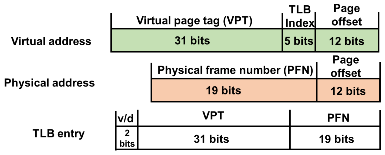
Advanced TLB optimizations

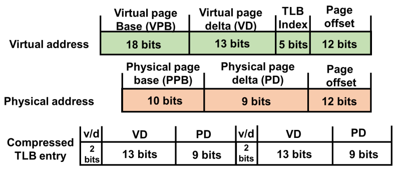
1. Large pages

- 4KB vs 2MB vs 1GB
  - + good TLB performance (less memory accesses!)
  - + good for regular access pattern (streaming, sequential)
  - - Internal fragmentation
  - - Expensive data movement and longer accessing time
- Dynamic/multiple page sizes?
  - Rachata Ausavarungnirun, Joshua Landgraf, Vance Miller, Saugata Ghose, Jayneel Gandhi, Christopher J. Rossbach, and Onur Mutlu. 2017. Mosaic: a GPU memory manager with application-transparent support for multiple page sizes. In Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50 '17).
  - Guilherme Cox and Abhishek Bhattacharjee. 2017. Efficient Address Translation for Architectures with Multiple Page Sizes. In Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ‘17)
  - Complicated design and software/hardware overheads

2. Compression

- Regular stride page accesses (including consecutive ones)
- Use fewer bits in the VA # and PA #. → more translations in TLB
- Binh Pham, Viswanathan Vaidyanathan, Aamer Jaleel, and Abhishek Bhattacharjee. 2012. CoLT: Coalesced Large-Reach TLBs. In Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)
- B. Pham, A. Bhattacharjee, Y. Eckert, and G. H. Loh. 2014. Increasing TLB reach by exploiting clustering in page translations. In 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)
- Xulong Tang, Ziyu Zhang, Weizheng Xu, Mahmut Taylan Kandemir, Rami Melhem, and Jun Yang. 2020. Enhancing Address Translations in Throughput Processors via Compression. In Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques (PACT '20).





3. Speculative TLB

- Similar to cache block prediction and prefetching.
  - Heuristic
  - Machine learning
- Depends on the accuracy and page access pattern.
- T. W. Barr, A. L. Cox, and S. Rixner. 2011. SpecTLB: A mechanism for speculative address translation. In 2011 38th Annual International Symposium on Computer Architecture (ISCA)

4. Range TLB

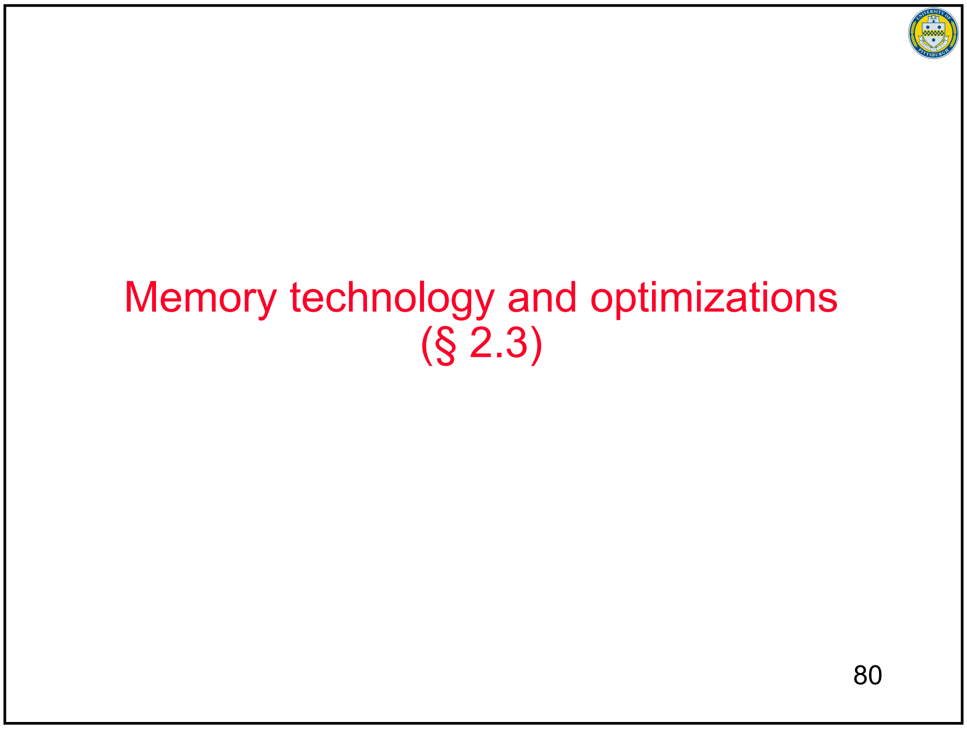
- Combine consecutive translations to occupy single entry in TLB
- Managed by the OS to determine and memorize the range of both virtual and physical addresses.
- Yan, Zi, et al. "Translation ranger: operating system support for contiguity-aware TLBs." *Proceedings of the 46th International Symposium on Computer Architecture*. (ISCA) 2019.

5. Least TLB

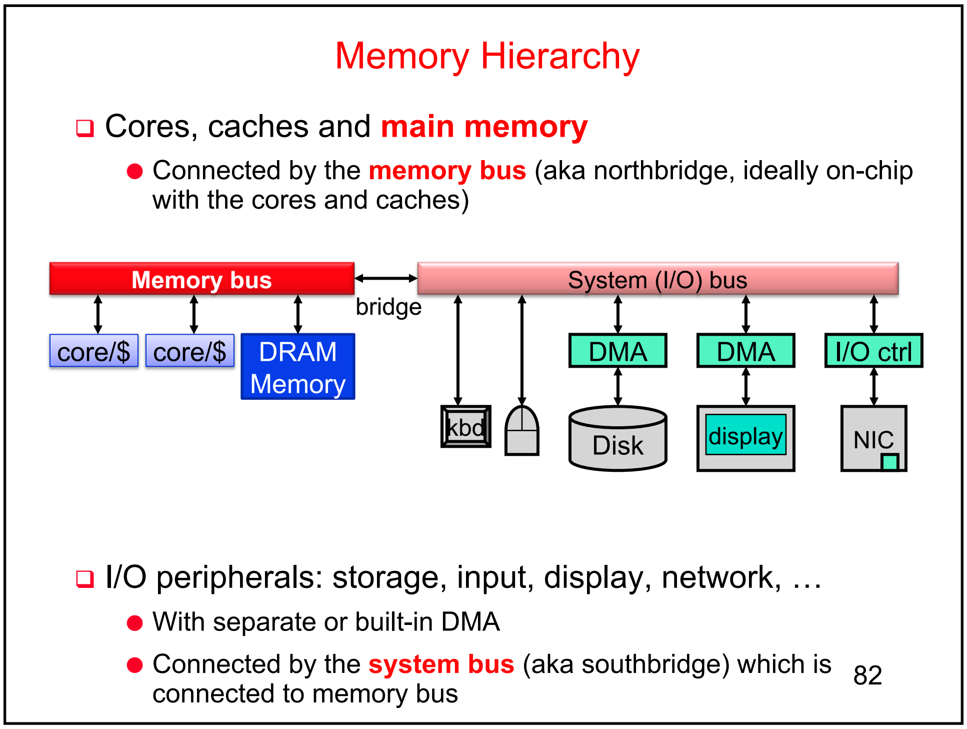
- For multi-GPU infrastructure, addresses are redundantly stored in GPU's local TLBs and IOMMU centralized TLB due to mostly-inclusive policy
- Proposed least-inclusive TLB to use the IOMMU TLB and other GPUs' TLBs as "victim" buffers for translation spilling.
- Bingyao Li, Jieming Yin, Youtao Zhang, Xulong Tang "Improving Address Translation in Multi-GPUs via Sharing and Spilling Aware TLB Design." In Proceedings of the 54th IEEE/ACM International Symposium on Microarchitecture (MICRO 2021)

6. Eager Paging

- Expose range information at allocation, instead of waiting to access (demand paging)
- Allow OS to aggressively optimize the range.
- Vasileios Karakostas, Jayneel Gandhi, Furkan Ayar, Adrián Cristal, Mark D. Hill, Kathryn S. McKinley, Mario Nemirovsky, Michael M. Swift, and Osman Ünsal. 2015. Redundant memory mappings for fast access to large memories. In Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA '15)
- Arkaprava Basu, Jayneel Gandhi, Jichuan Chang, Mark D. Hill, and Michael M. Swift. 2013. Efficient Virtual Memory for Big Memory Servers. In Proceedings of the 40th Annual International Symposium on Computer Architecture (Tel-Aviv, Israel) (ISCA '13)
- Swapnil Haria, Mark D. Hill, and Michael M. Swift. 2018. Devirtualizing Memory in Heterogeneous Systems. In Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (Williamsburg, VA, USA) (ASPLOS '18)



Memory technology and optimizations (§ 2.3)

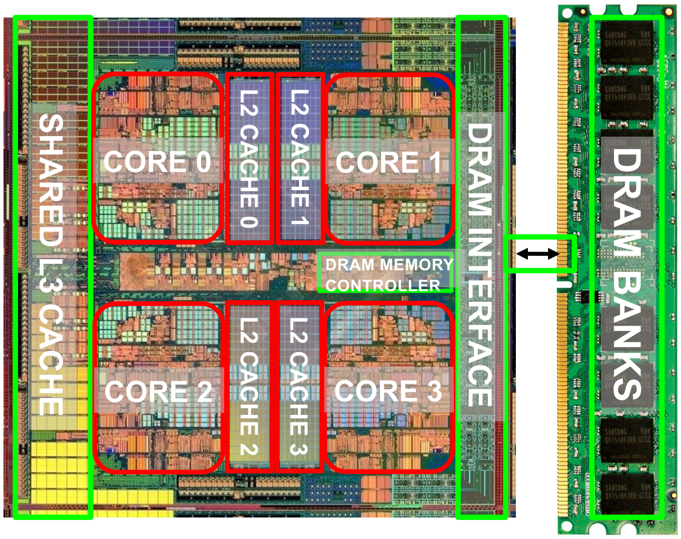


Memory Hierarchy

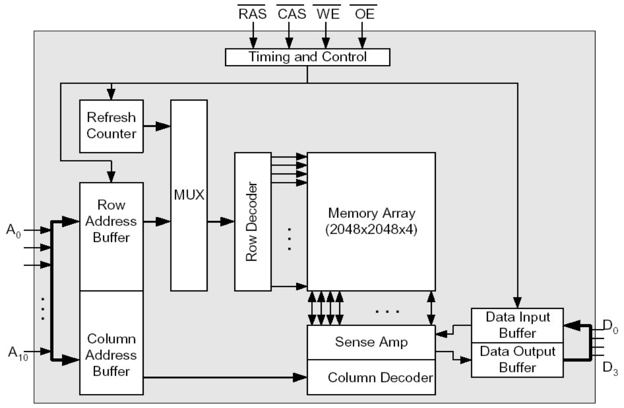
- ❑ Cores, caches and **main memory**
  - Connected by the **memory bus** (aka northbridge, ideally on-chip with the cores and caches)

- ❑ I/O peripherals: storage, input, display, network, …
  - With separate or built-in DMA
  - Connected by the **system bus** (aka southbridge) which is connected to memory bus

Main Memory in the System



DRAM

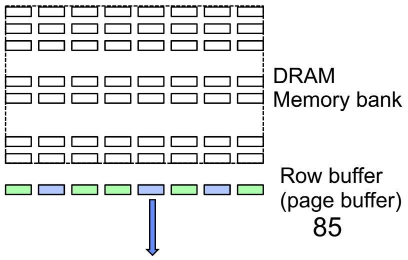


- SDRAM = DRAM with a clocked interface
- DDR SDRAM = double data rate, transfer data at both clock edges
  - DDR2 (1.8 V, 266-400 MHz)
  - DDR3 (1.5 V, 800 MHz)
  - DDR4 (1-1.2 V, 1600 MHz)

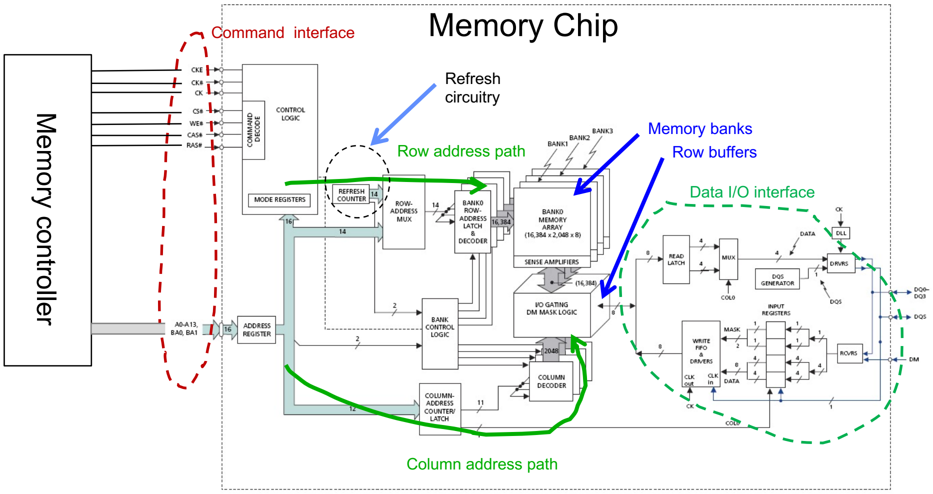
DRAM operation

- Each memory bank has a “row buffer”, which is non-volatile (SRAM registers)
- To read a byte (a similar process applies for writing):
  - The memory controller sends the row address of the byte
  - The entire row is read into the row buffer (the row is opened)
  - The memory controller sends the column address of the byte
  - The memory returns the byte to the controller (from the row buffer)
  - The memory controller sends a Pre-charge signal (close the open row)

- **Closed row/page policy**: close the row after you read the data
- **Open row/page policy**: close the row only when you want to open a new row – useful if you will read again from the same row.



DRAM internals

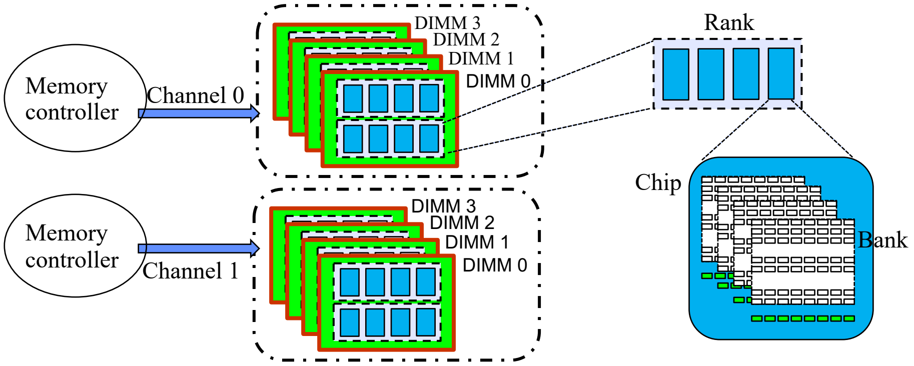


EX: a 128 Mbytes chip (2^{27}) → 4 banks, each having 32 Mbytes organized as 2^{14}=16K rows and 2^{11}=2K columns

Bank address

2 bits	14 row address bits	11 column address bits

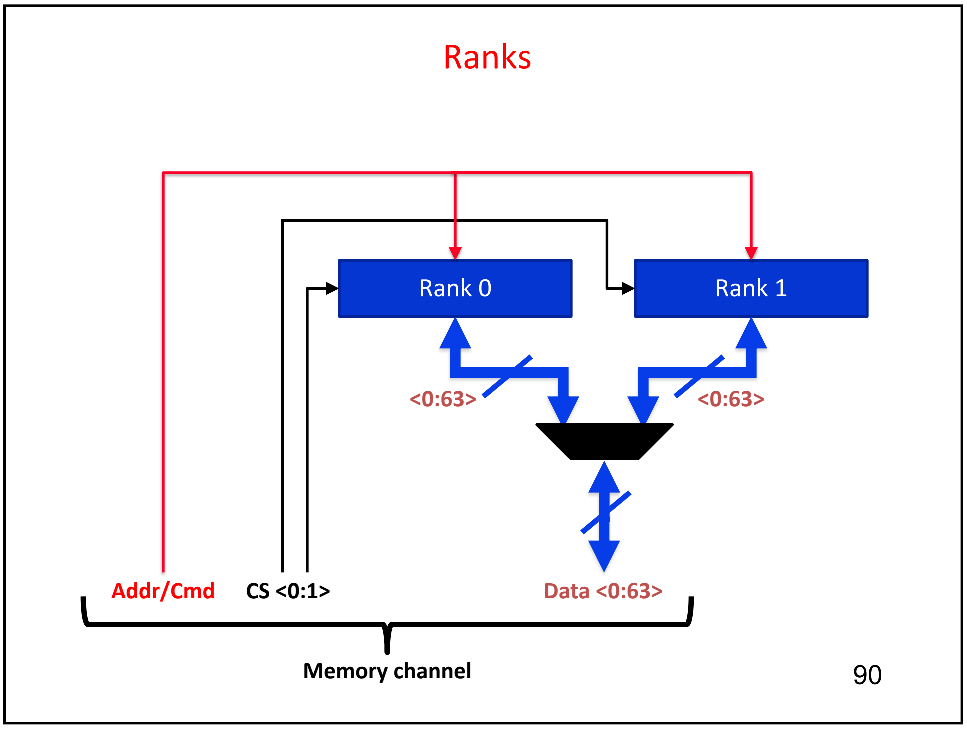
DRAM Organization



- A memory chip is organized internally as a number of banks (1-8 usually).
- Multiple banks can execute different commands at the same time
- A Rank consists of multiple (parallel) chips contributing to the same transaction. For example, each of 4 chips can provide a byte for a total of 32 data bits (read or written).
- A DIMM (Dual Inline Memory Module) consists of 1 – 4 ranks (2 in the figure) mounted on a single printed-circuit board.
- A Channel supports multiple DIMMS (4 in the above figure)

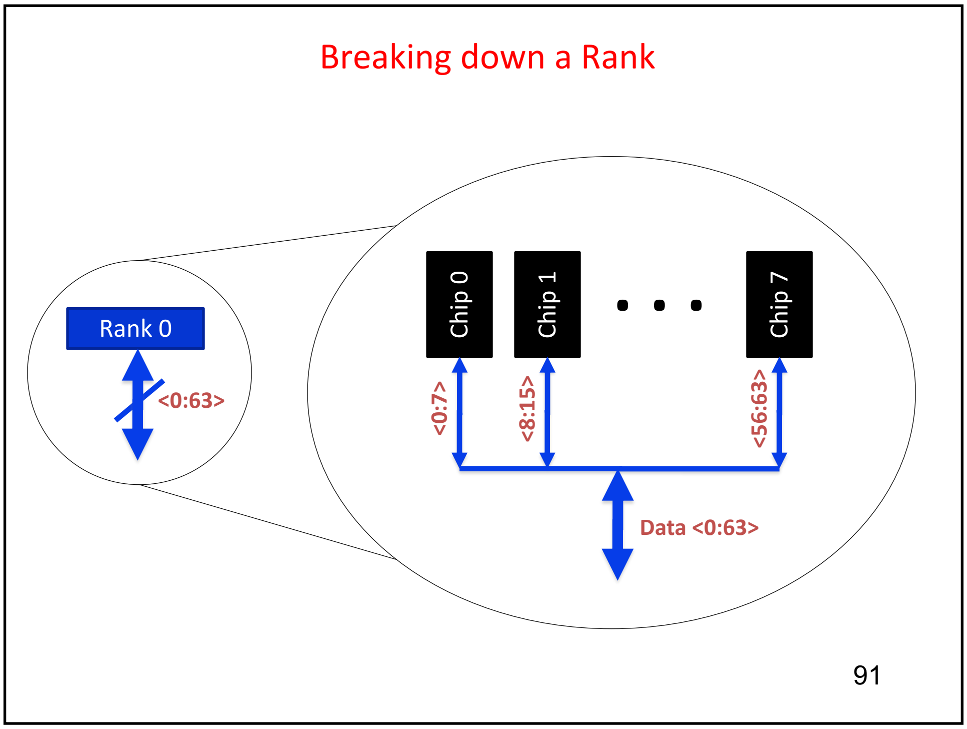
Channels, Memory Controllers, DIMMs, DRAM Chips

- To maximize memory bandwidth, there are multiple memory channels in the system. Each channel is managed by one memory controller (MC), and is connected to one or more DIMMs, each containing multiple DRAM chips.
- These DRAM chips are logically arranged into multiple ranks. Internally, each DRAM chip is partitioned into banks that are accessed in parallel.
- In each memory controller, there is one read queue and one write queue to buffer the outstanding memory requests. A scheduler (e.g., FCFS) determines which request should be *serviced* next. For each channel, only one request can be sent to the memory through the bus at each clock cycle.



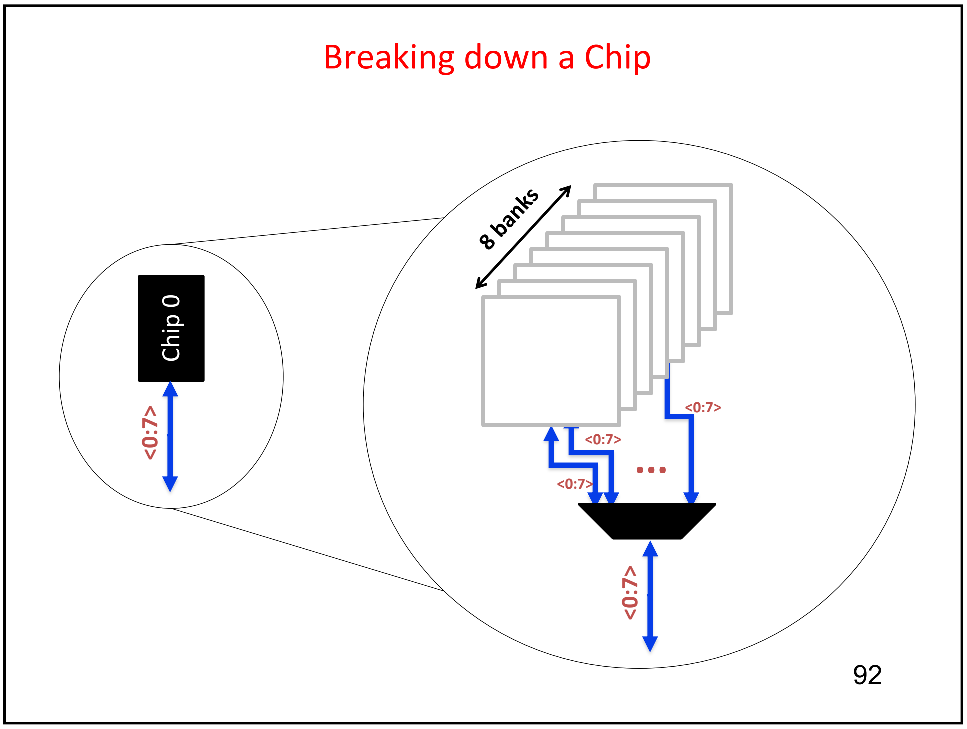
Ranks

90

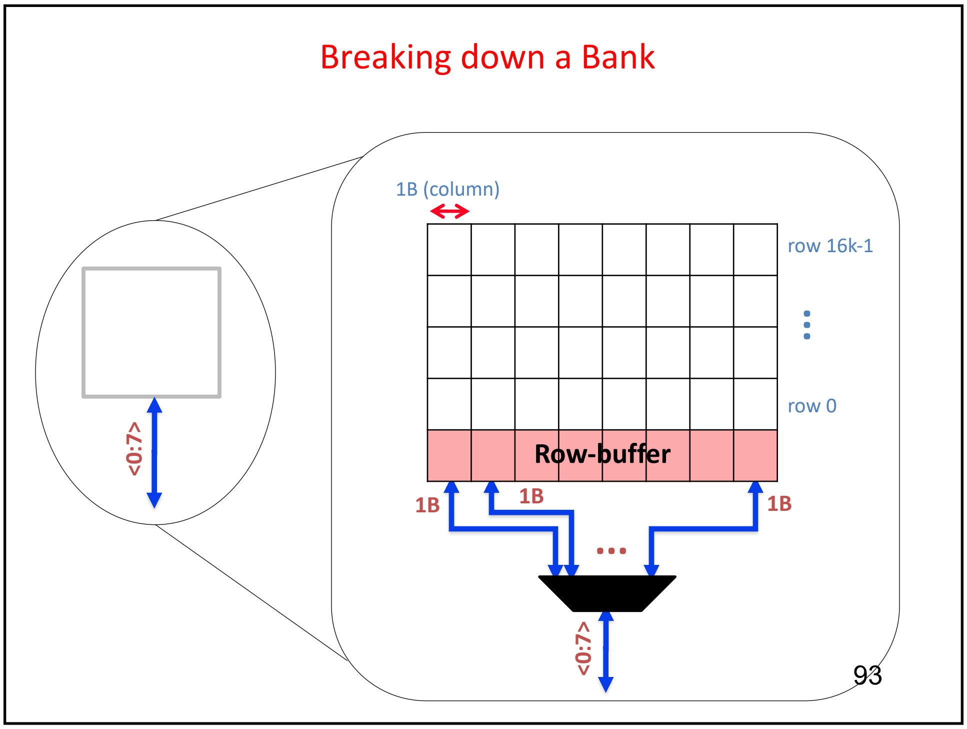


Breaking down a Rank

91



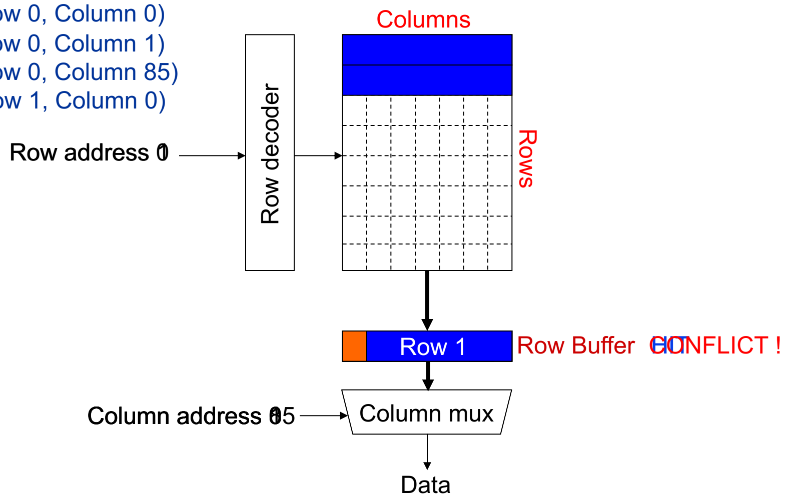
Breaking down a Chip



Breaking down a Bank

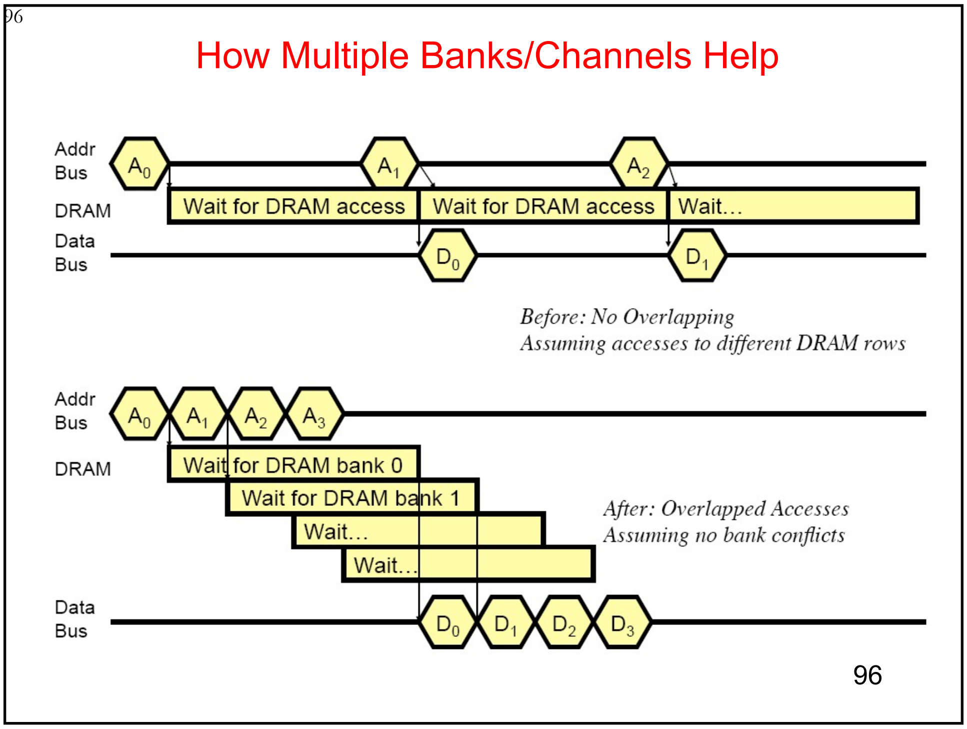
DRAM Bank Operation

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



Multiple Banks (Interleaving) and Channels

- Multiple banks
  - Enable concurrent (pipelined) DRAM accesses
  - Data from different banks are NOT obtained in the same cycle
  - Bits in address determine which bank an address resides in
- Multiple independent channels serve the same purpose
  - But they are even better because they have separate data buses
  - Increased bus bandwidth
- Enabling more concurrency requires reducing
  - Bank conflicts
  - Channel conflicts



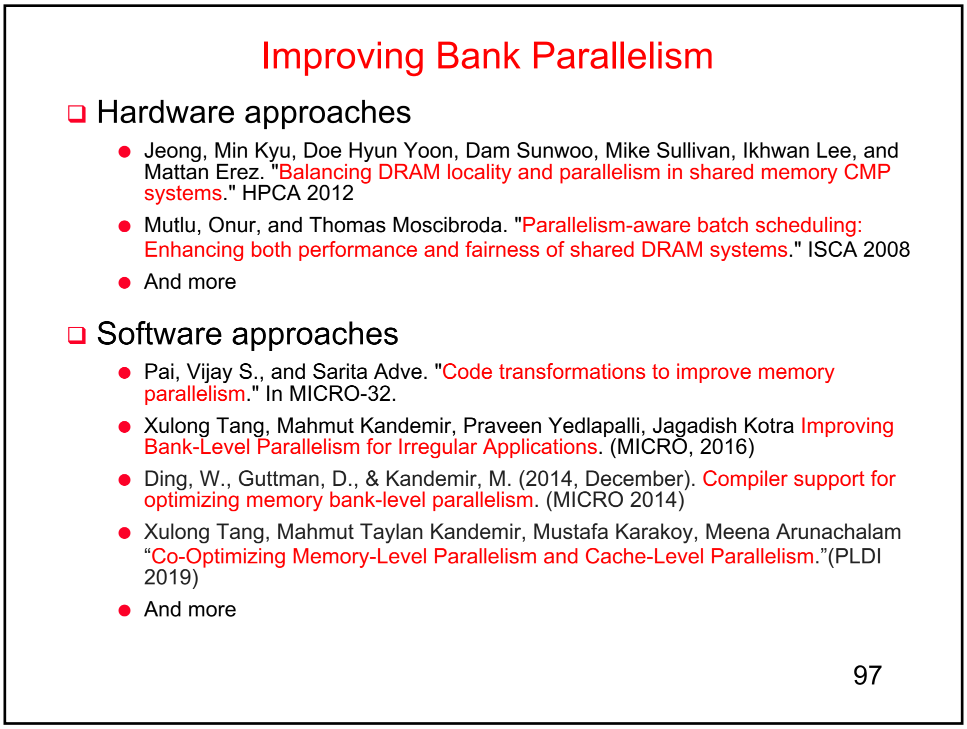
96

How Multiple Banks/Channels Help

Before: No Overlapping
Assuming accesses to different DRAM rows

After: Overlapped Accesses
Assuming no bank conflicts

96

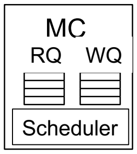


Improving Bank Parallelism

- Hardware approaches
  - Jeong, Min Kyu, Doe Hyun Yoon, Dam Sunwoo, Mike Sullivan, Ikhwan Lee, and Mattan Erez. "Balancing DRAM locality and parallelism in shared memory CMP systems." HPCA 2012
  - Mutlu, Onur, and Thomas Moscibroda. "Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared DRAM systems." ISCA 2008
  - And more
- Software approaches
  - Pai, Vijay S., and Sarita Adve. "Code transformations to improve memory parallelism." In MICRO-32.
  - Xulong Tang, Mahmut Kandemir, Praveen Yedlapalli, Jagadish Kotra Improving Bank-Level Parallelism for Irregular Applications. (MICRO, 2016)
  - Ding, W., Guttman, D., & Kandemir, M. (2014, December). Compiler support for optimizing memory bank-level parallelism. (MICRO 2014)
  - Xulong Tang, Mahmut Taylan Kandemir, Mustafa Karakoy, Meena Arunachalam "Co-Optimizing Memory-Level Parallelism and Cache-Level Parallelism."(PLDI 2019)
  - And more

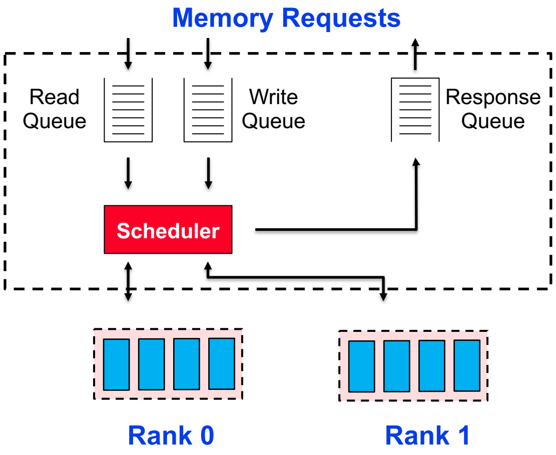
97

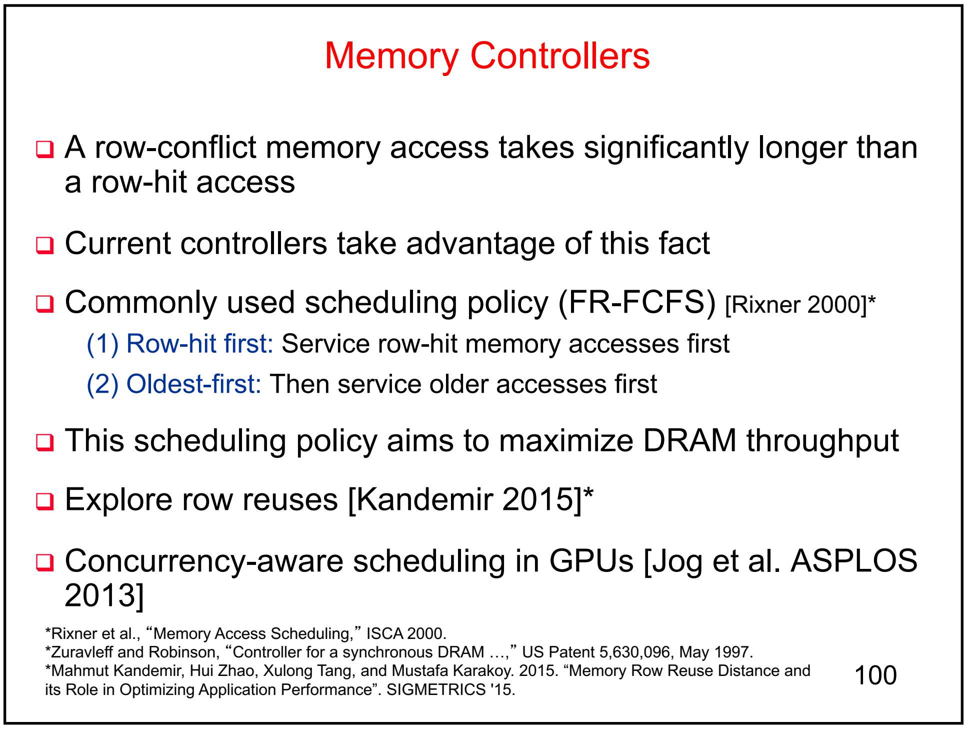
On-Chip Memory Controller



- ❑ A front-end buffers requests to the DRAM from the LLC (is independent of DRAM type)
  - Decodes the address into bank, row, column
  - Schedules requests to DRAM for max bandwidth
    1. Reorders bursts to minimize bank conflicts
    2. Prioritizes reads over writes (a core is waiting for the read data)
  - Sends responses from the DRAM back to the LLC
- ❑ Back-end interface to the target DRAM memory (and is depending on DRAM type)

Memory Controller



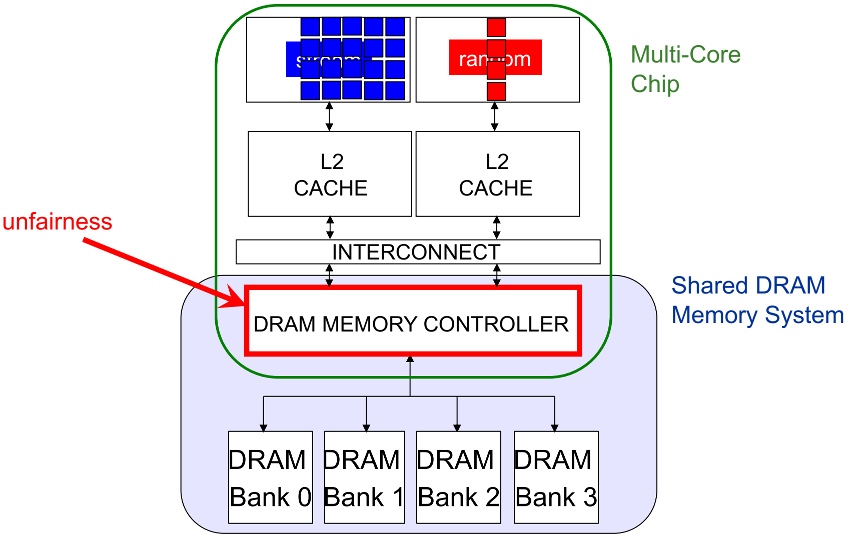


Memory Controllers

- ❑ A row-conflict memory access takes significantly longer than a row-hit access
- ❑ Current controllers take advantage of this fact
- ❑ Commonly used scheduling policy (FR-FCFS) [Rixner 2000]*
  - (1) Row-hit first: Service row-hit memory accesses first
  - (2) Oldest-first: Then service older accesses first
- ❑ This scheduling policy aims to maximize DRAM throughput
- ❑ Explore row reuses [Kandemir 2015]*
- ❑ Concurrency-aware scheduling in GPUs [Jog et al. ASPLOS 2013]

*Rixner et al., "Memory Access Scheduling," ISCA 2000.
*Zuravleff and Robinson, "Controller for a synchronous DRAM …," US Patent 5,630,096, May 1997.
*Mahmut Kandemir, Hui Zhao, Xulong Tang, and Mustafa Karakoy. 2015. "Memory Row Reuse Distance and its Role in Optimizing Application Performance". SIGMETRICS '15.

Uncontrolled Interference: An Example



Multi-Core Chip

Shared DRAM Memory System

unfairness

A Memory Performance Hog

```
// initialize large arrays A, B

for (j=0; j<N; j++) {
    index = j*linesize;   streaming
    A[index] = B[index];
    …
}
```

```
// initialize large arrays A, B

for (j=0; j<N; j++) {
    index = rand();   random
    A[index] = B[index];
    …
}
```

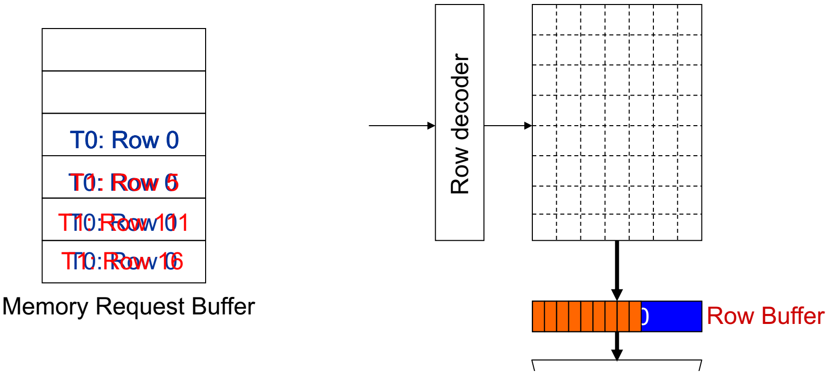
STREAM

- Sequential memory access
- Very high row buffer locality (96% hit rate)
- Memory intensive

RANDOM

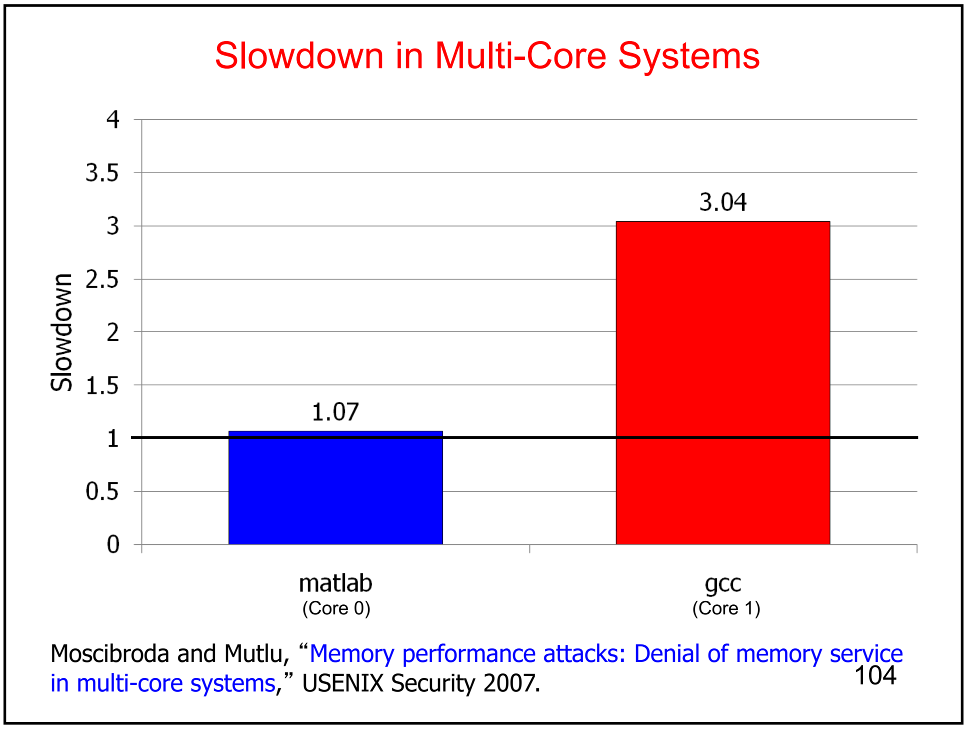
- Random memory access
- Very low row buffer locality (3% hit rate)
- Similarly memory intensive

What Does the Memory Hog Do?



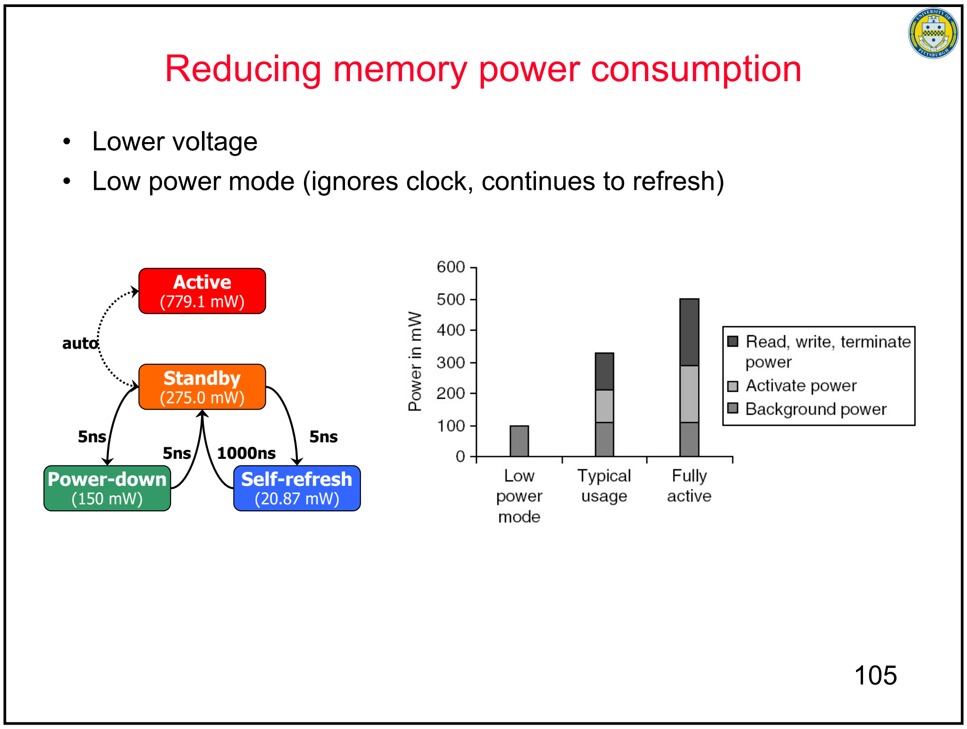
Row size: 8KB, cache block size: 64B

128 (8KB/64B) requests of T0 serviced before T1



Slowdown in Multi-Core Systems

Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

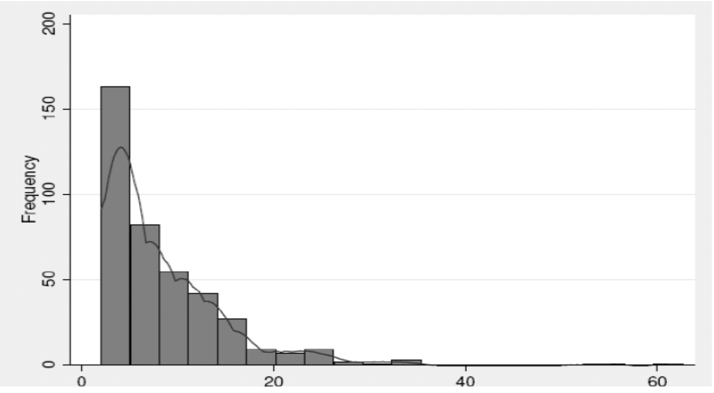


Reducing memory power consumption

- Lower voltage
- Low power mode (ignores clock, continues to refresh)

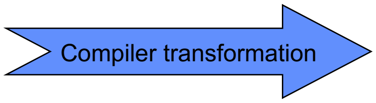
OS assisted Memory Power Management?

- keep a histogram for patterns of bank accesses and idle time distributions.
- Use machine learning techniques to select the optimal “threashold” to switch to a lower power mode.



Example of compiler assisted Memory Power Management?

….
Load x
….
Store x
….
Load z
….
Load y
….
Store z
….
Store y
….
….

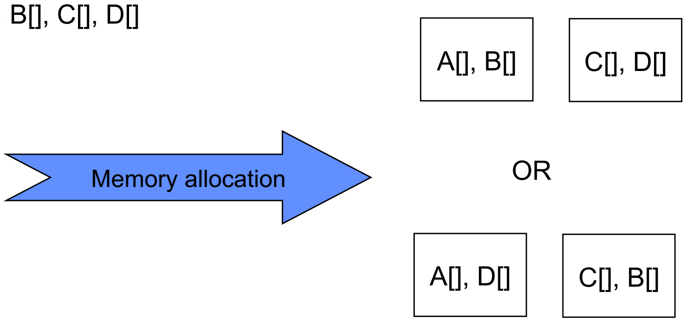


Load x
Load y
Load z
….
….
….
….
….
….
Store y
….
Store z

Code transformations to increase the memory idle time (the time between memory accesses).

Example of compiler assisted Memory Power Management?

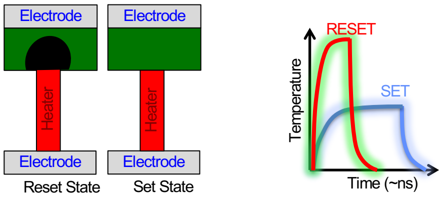
Declare A[], B[], C[], D[]
….
Access A
….
Access D
….
Access B
….
Access C
….
Access B
….
….



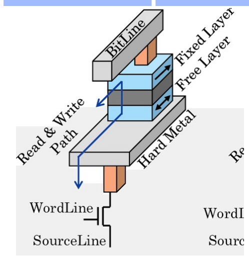
Algorithms that use the access pattern to allocate memory to banks in a way that maximizes bank idle times

Alternative memory technology

	Read Speed	Write Speed	Cell Area	Endurance	Addressability
DRAM	20~50ns	20~50ns	$6F^2$	10^{15}	Yes
SRAM	~2ns	~2ns	$146F^2$	10^{15}~10^{16}	Yes
NAND Flash	25us	500us	$5F^2$	10^4~10^5	No
STT-RAM	2ns	10ns	37~$40F^2$	10^{12}	Yes
PCM	30~50ns	~1us	5~$8F^2$	10^7~10^8	Yes
Domain wall memory					



PCM cells



STT_RAM cells

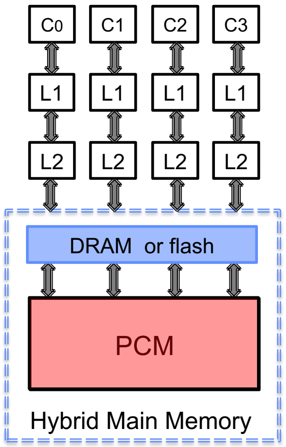
Memory dependability

- Memory is susceptible to cosmic rays
- *Soft errors*: dynamic/transient errors
  - Detected and fixed by error correcting codes (ECC)
- *Hard errors*: permanent errors
  - Use sparse rows to replace defective rows
- Chip-level errors – (Chipkill: a RAID-like error recovery technique)
- *Stuck-at errors*
  - May use data-dependent sparing
- Endurance problems
- Cross-talk (bit-line and word-line)
- Row hammer attack
- Safely reducing refresh rate of DRAM

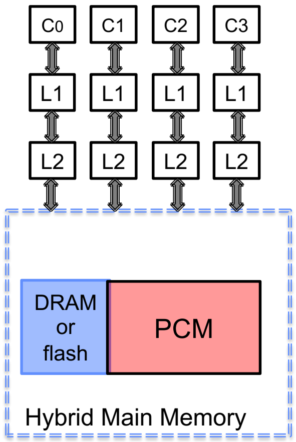
  (Example: Refresh-now-and-then)

Hybrid main memory

- To deal with asymmetric read and write
  – Write is slower, more time consuming and causes wear

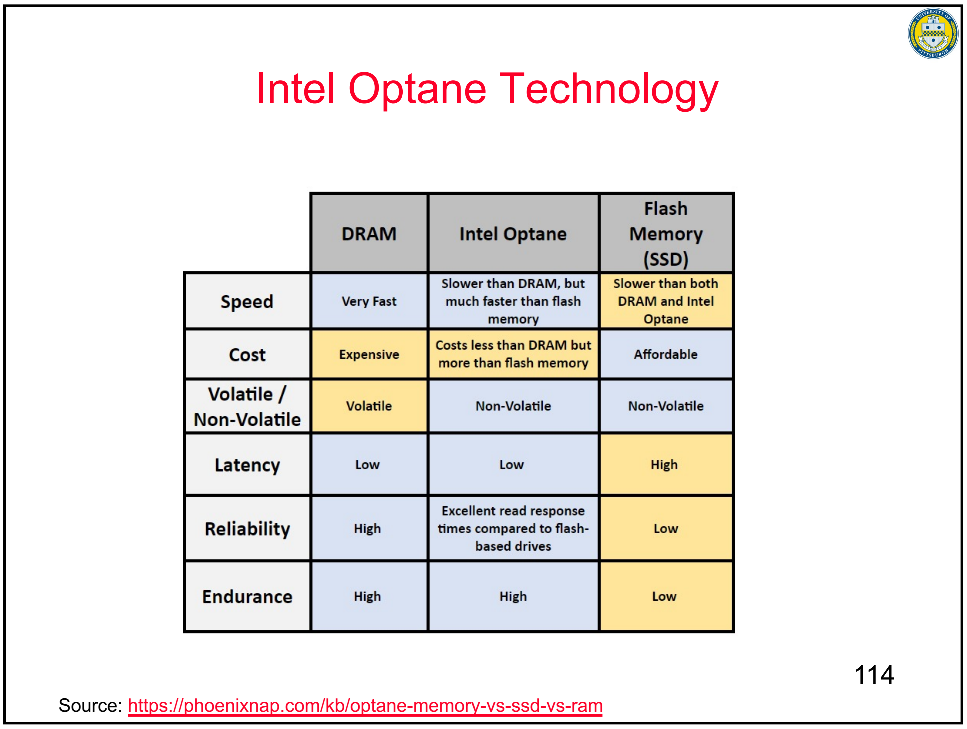


Vertical organization



Horizontal organization

Intel Optane. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-optane-technology.html



Intel Optane Technology

	DRAM	Intel Optane	Flash Memory (SSD)
Speed	Very Fast	Slower than DRAM, but much faster than flash memory	Slower than both DRAM and Intel Optane
Cost	Expensive	Costs less than DRAM but more than flash memory	Affordable
Volatile / Non-Volatile	Volatile	Non-Volatile	Non-Volatile
Latency	Low	Low	High
Reliability	High	Excellent read response times compared to flash-based drives	Low
Endurance	High	High	Low

Source: https://phoenixnap.com/kb/optane-memory-vs-ssd-vs-ram