CS/ECE 752:
Advanced Computer Architecture I

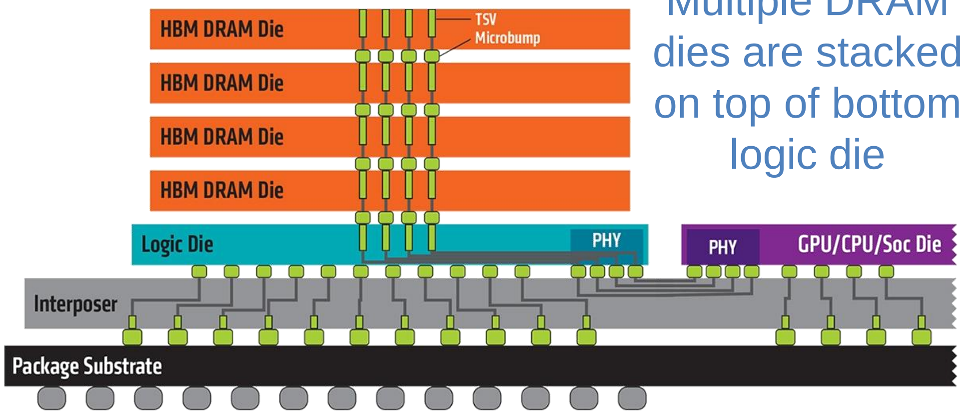
Prof. Matthew D. Sinclair

DRAM Basics

Slide History/Attribution Diagram:

Die-stacked DRAM (3D-DRAM)

- Die-stacked DRAM:
  - Top layers store data
  - Bottom logic layer stores the various control, access, and interface circuits
- Magic: Stacked means high density, so high b/w interposer integration not so expensive.

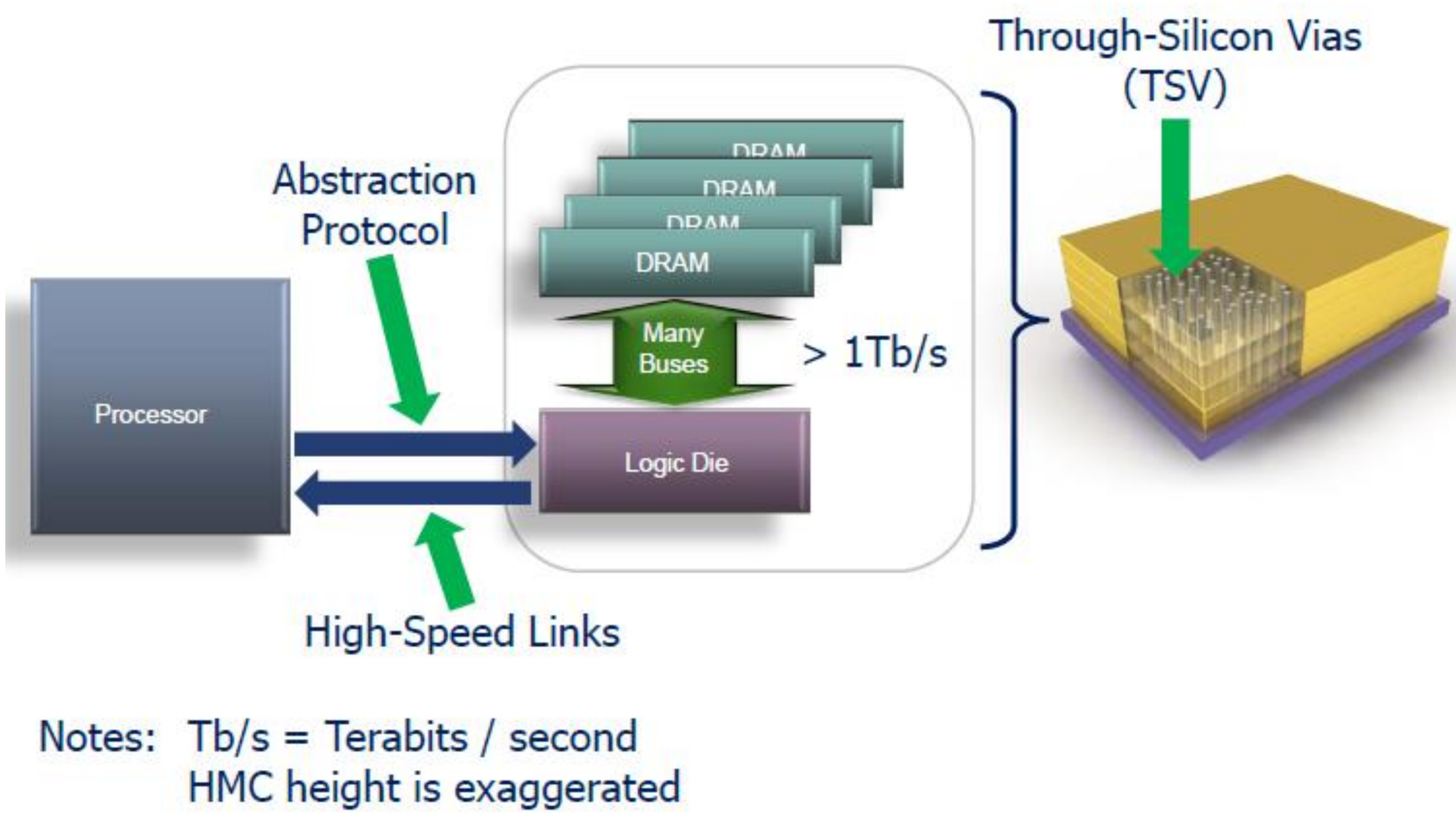


Multiple DRAM dies are stacked on top of bottom logic die

Source: AMD.com

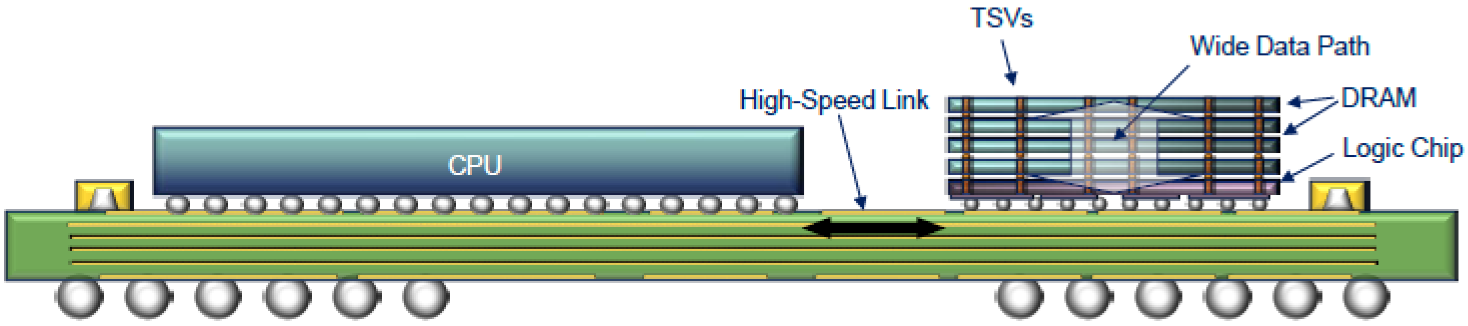
- Current Products:
  - Hybrid Memory Cube (Micron)
  - High Bandwidth Memory (Samsung, AMD, and Hynix)
- Tradeoffs:
  - Basically the same latency as DRAM, but much higher bandwidth
  - More expensive, so we can't have as much memory…
- In GPU: Need high bandwidth **all the time,** but don't need *that* much memory, so it can serve as the main memory.
- What about **CPU:** Need huge memory, so cost is critical…

Emerging: Hybrid Memory Cube



- Micron proposal [Pawlowski, Hot Chips 11]
  - www.hybridmemorycube.org (Now a dead link :( )

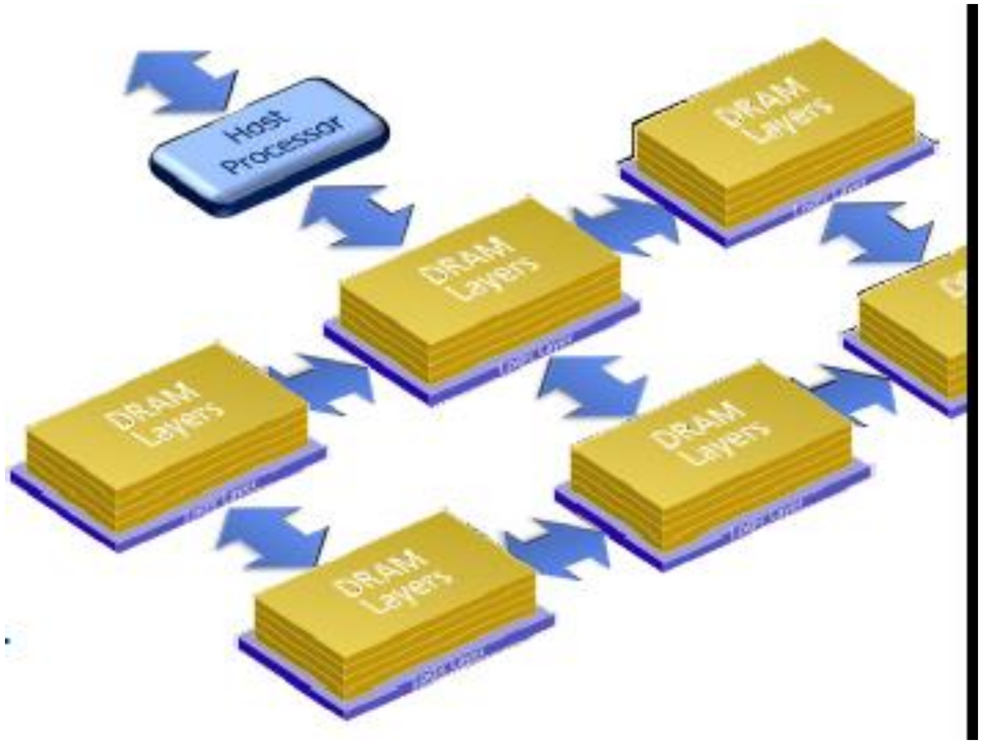
Hybrid Memory Cube MCM



Notes: MCM = multi-chip module
Illustrative purposes only; height is exaggerated

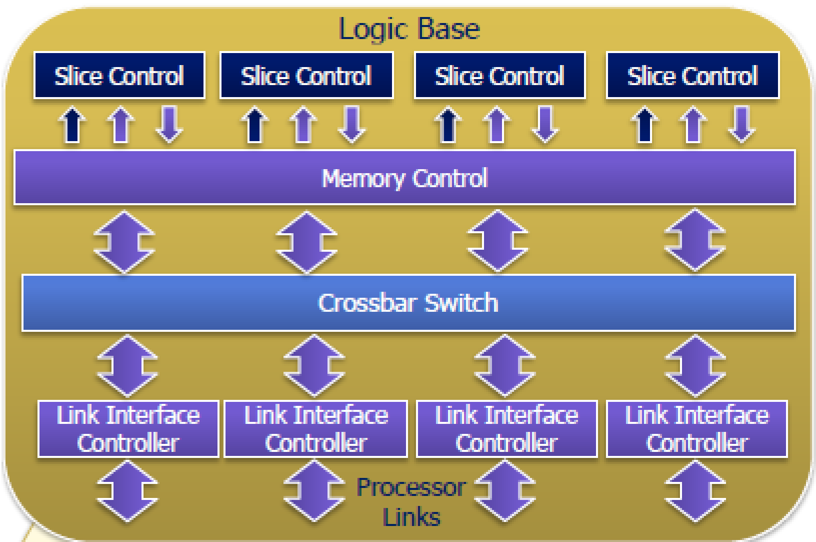
- Micron proposal [Pawlowski, Hot Chips 11]
  - www.hybridmemorycube.org (Now a dead link :( )

Network of DRAM



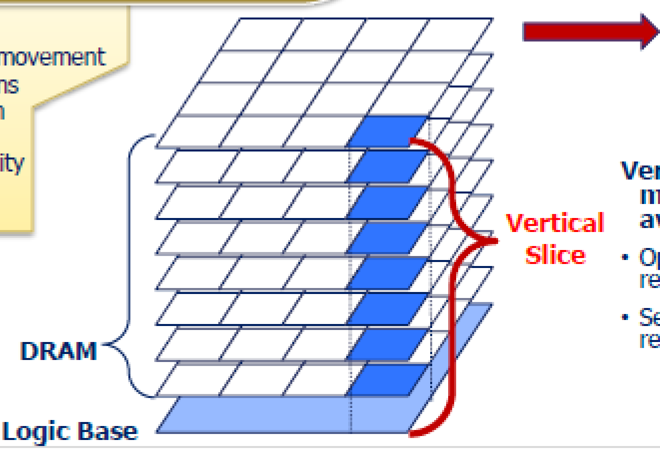
- Traditional DRAM: star topology
- HMC: mesh, etc. are feasible

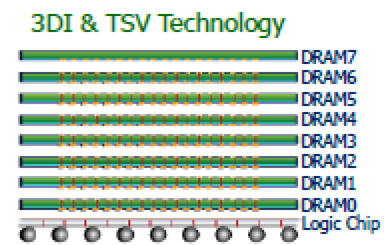
Hybrid Memory Cube



Logic Base

- Wide, high-speed local bus for data movement
- Advanced memory controller functions
- DRAM control at memory rather than distant host controller
- Reduced memory controller complexity and increased efficiency





Add sophisticated switching and optimized memory control…

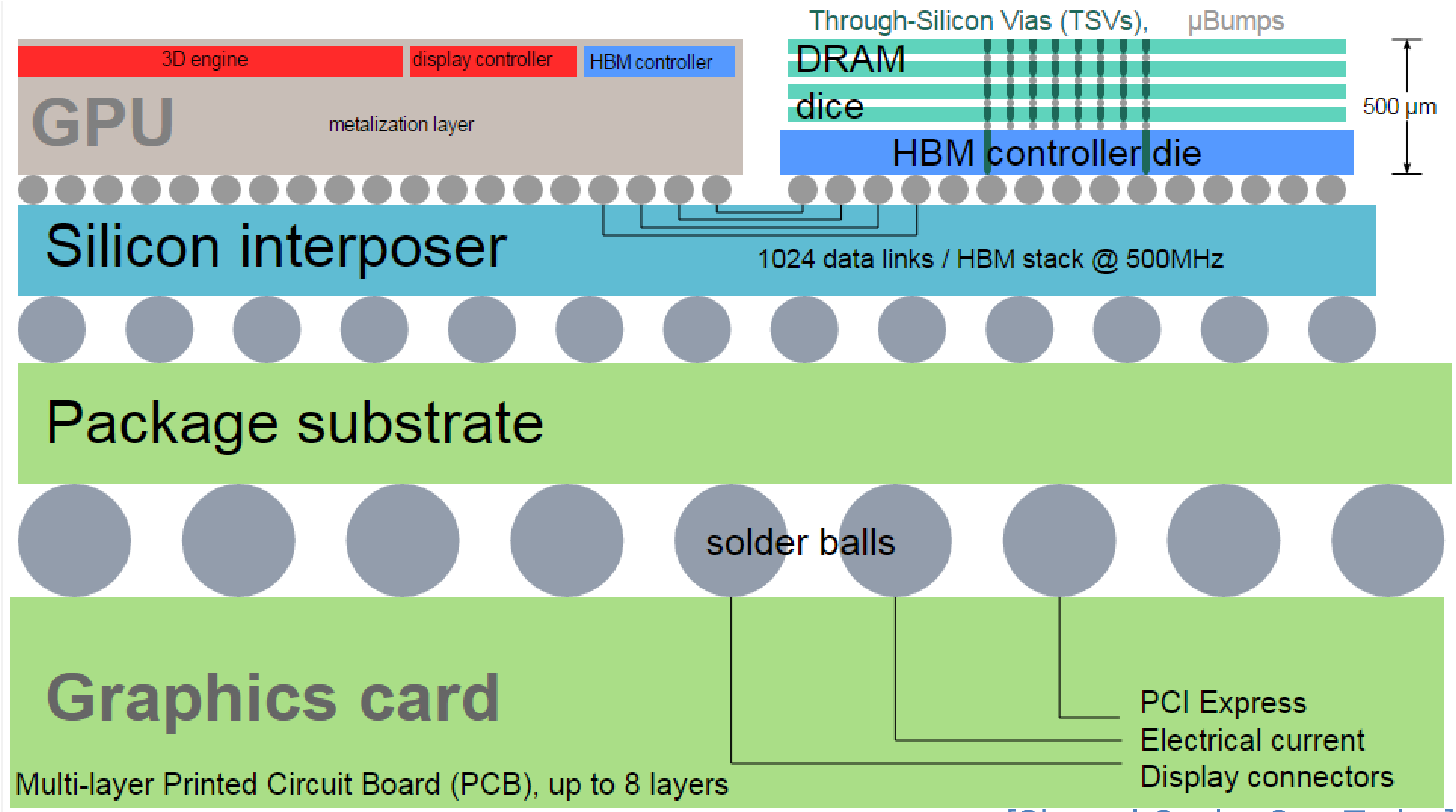
And now we have a whole new set of capabilities

Vertical Slices are managed to maximize overall device availability

- Optimized management of energy and refresh
- Self test, error detection, correction, and repair in the logic base layer

- High-speed logic segregated in chip stack
- 3D TSV for bandwidth

High Bandwidth Memory (HBM)



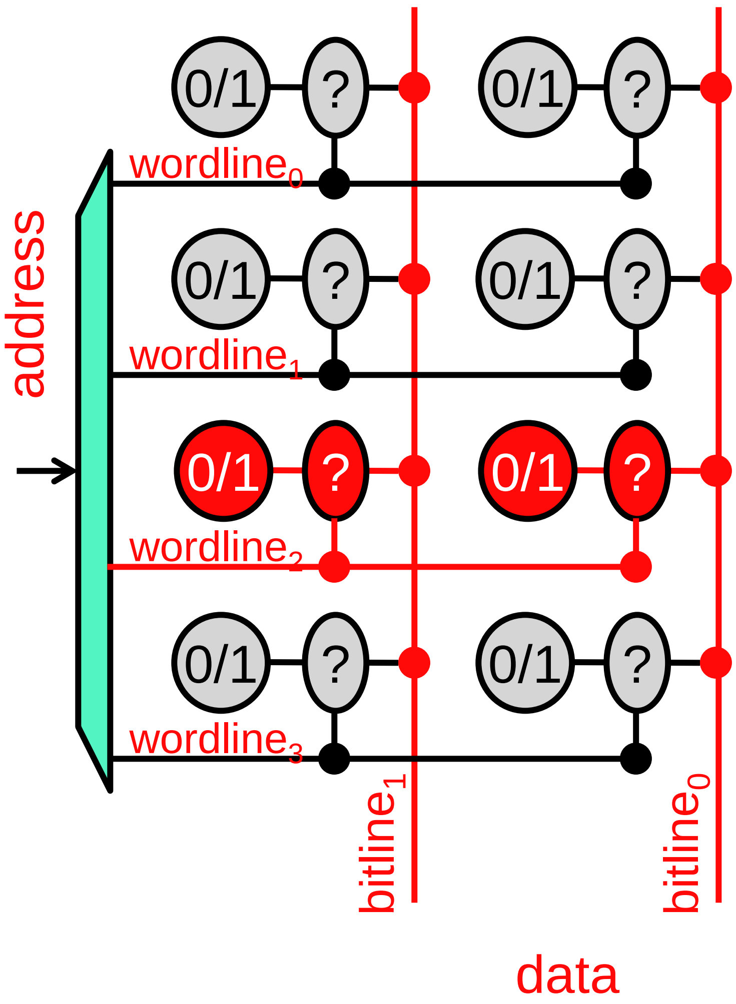
[Shmuel Csaba Otto Traian]

- High-speed serial links vs. 2.5D silicon interposer
- Commercialized, HBM2/HBM3 …

Future: Resistive memory

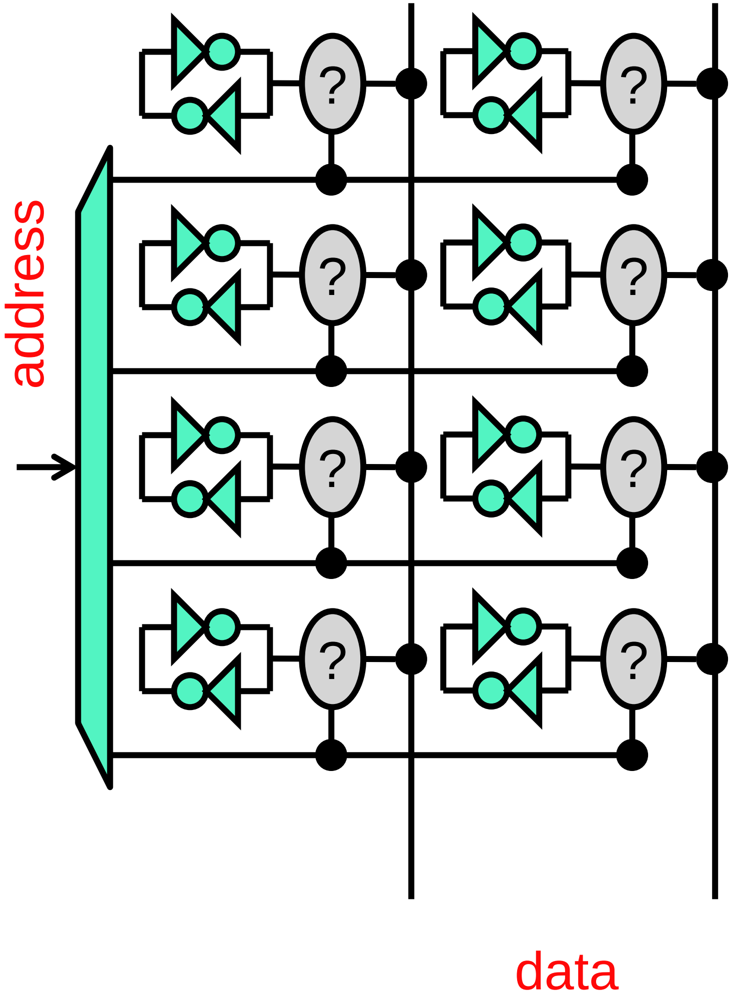
- PCM: store bit in phase state of material
- Alternatives:
  - Memristor, STT-MRAM
- Nonvolatile
- Dense: cross-point architecture (no access device)
- Relatively fast for read
- Very slow for write (also high power)
- Write endurance often limited
  - Write leveling (also done for flash)
  - Avoid redundant writes (read, cmp, write)
  - Fix individual bit errors (write, read, cmp, fix)
- Lots of work on using this to augment/replace main memory

RAM



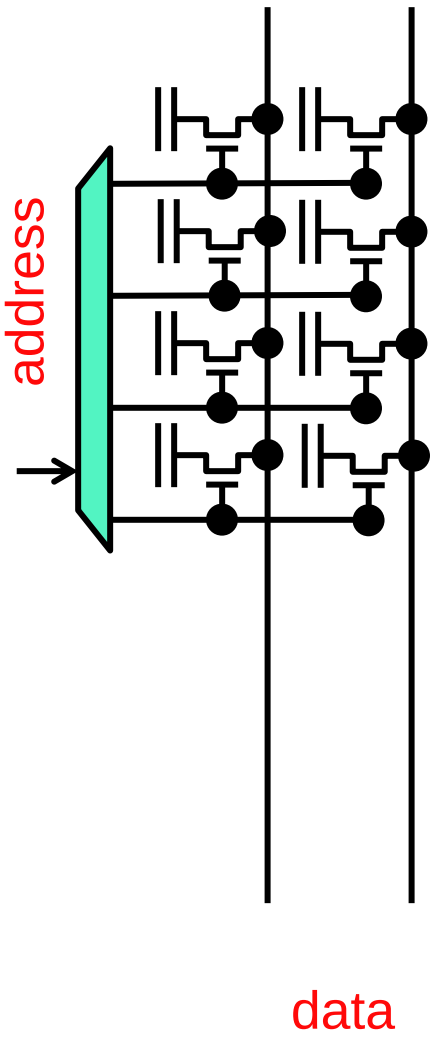
- RAM: large storage arrays
- Basic structure
  - MxN array of bits (M N-bit words)
    - This one is 4x2
  - Bits in word connected by **wordline**
  - Bits in position connected by **bitline**
- Operation
  - Address decodes into M wordlines
  - High wordline → word on bitlines
  - Bit/bitline connection → read/write
- Access latency
  - #ports * √#bits

SRAM



- **SRAM**: static RAM
  - Bits as cross-coupled inverters (CCI)
  - – Four transistors per bit
  - – More transistors for ports
- **"Static"** means
  - Inverters connected to pwr/gnd
  - + Bits naturally/continuously "refreshed"
- Designed for speed

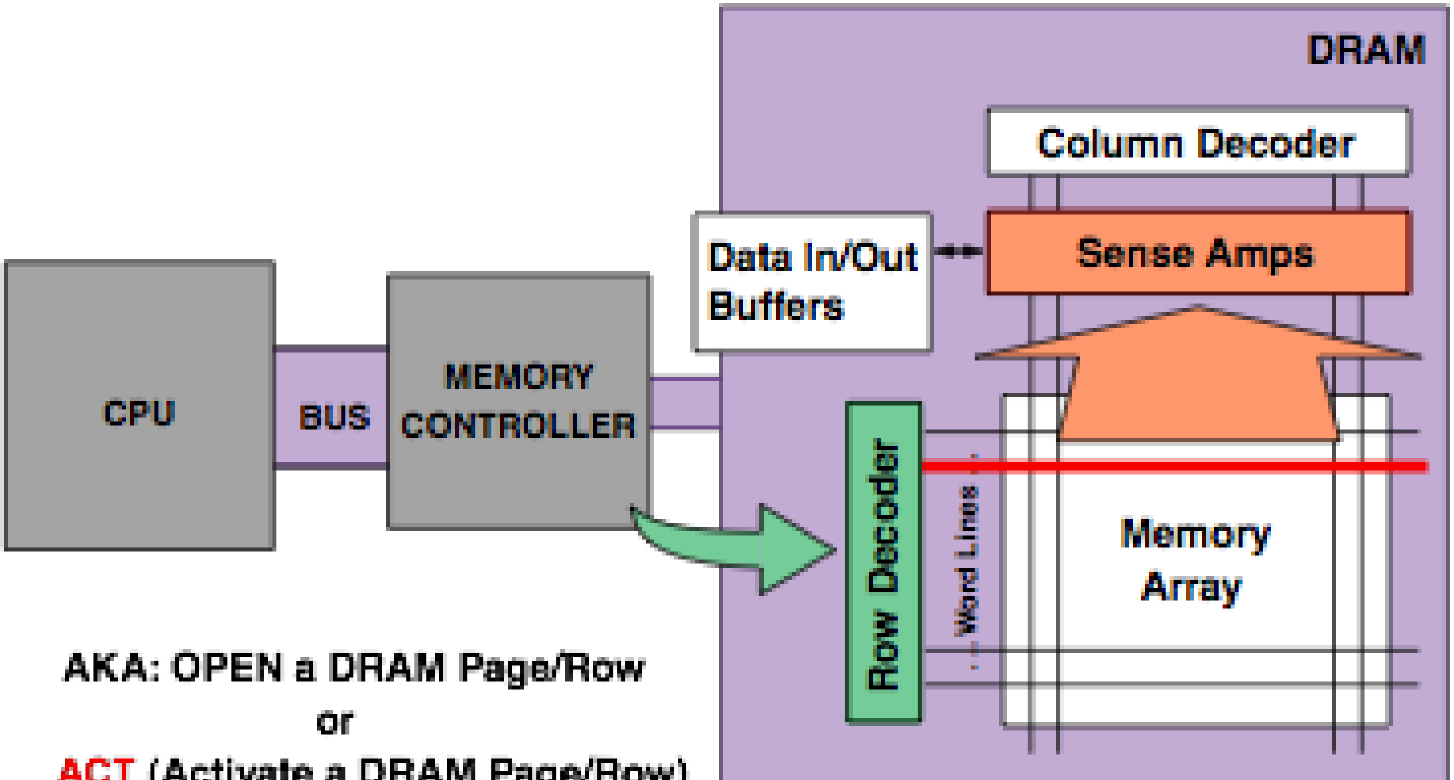
DRAM



- **DRAM**: dynamic RAM
  - Bits as capacitors
  - \+ Single transistors as ports
  - \+ One transistor per bit/port
- **"Dynamic"** means
  - Capacitors not connected to pwr/gnd
  - – Stored charge decays over time
  - – Must be explicitly refreshed
- Designed for density

DRAM Basics [Jacob and Wang]

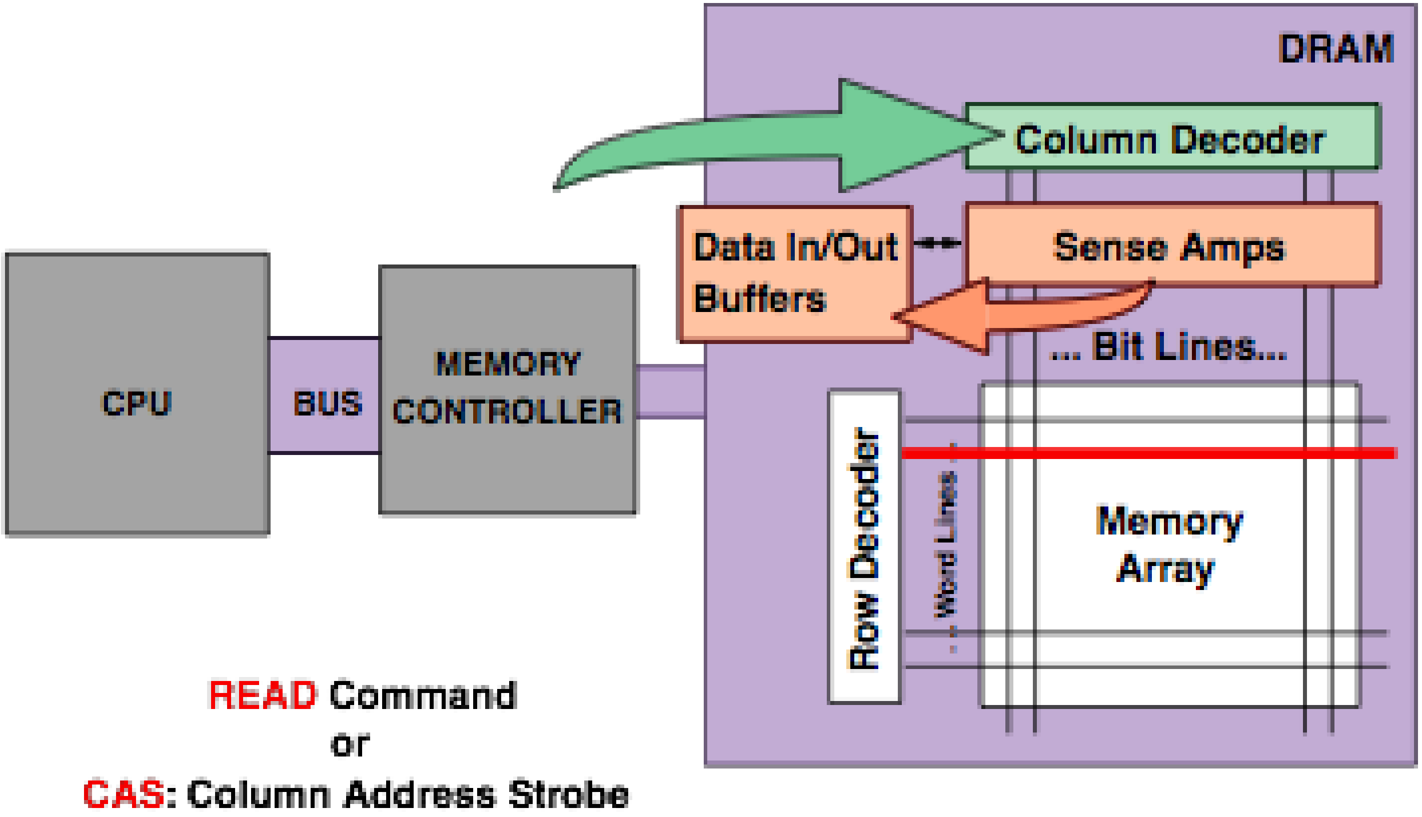
- Precharge and Row Access



AKA: OPEN a DRAM Page/Row
or
ACT (Activate a DRAM Page/Row)
or
RAS (Row Address Strobe)

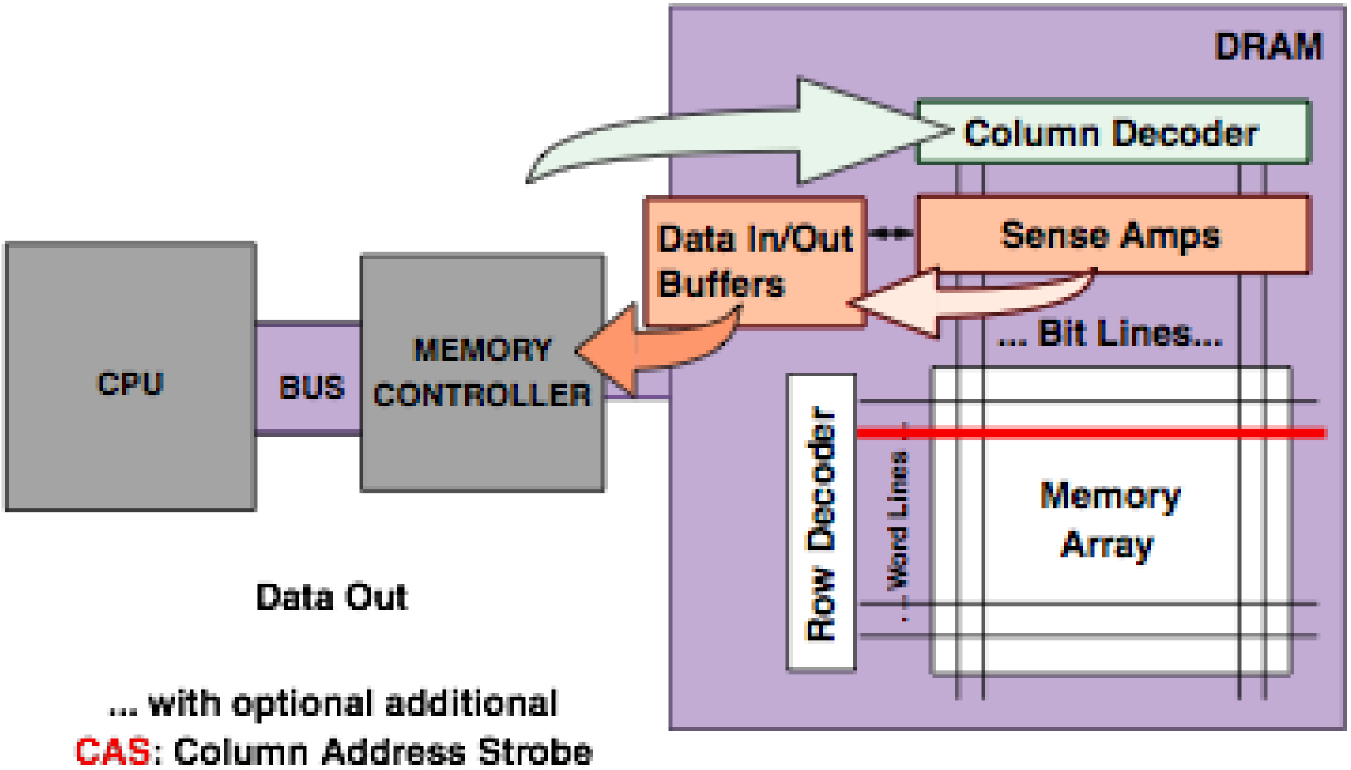
DRAM Basics, cont.

- Column Access

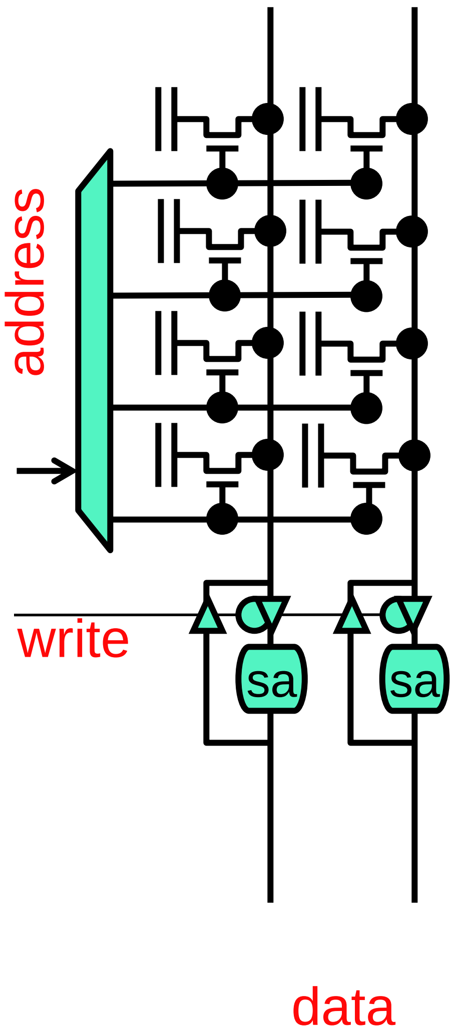


DRAM Basics, cont.

- Data Transfer



DRAM Operation I

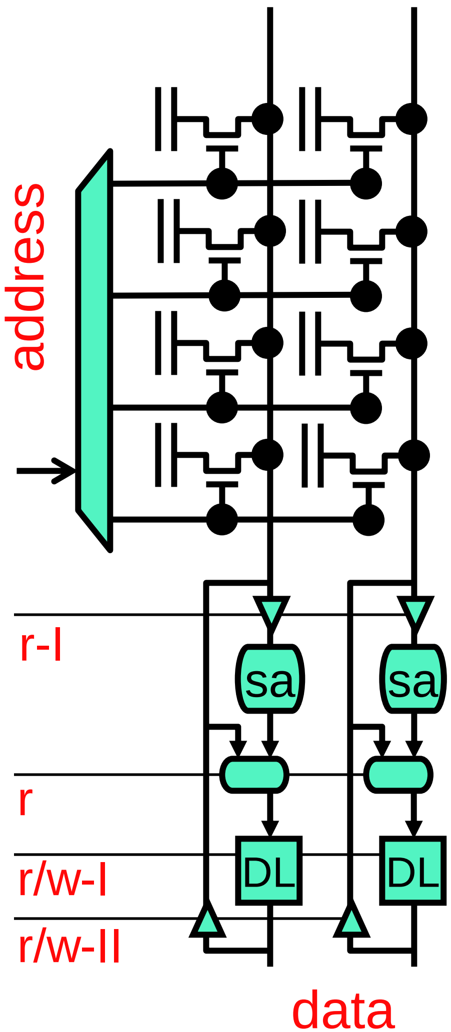


- Read: similar to cache read
  - Phase I: pre-charge bitlines to 0.5V
  - Phase II: decode address, enable wordline
    - Capacitor swings bitline voltage up(down)
    - Sense-amplifier interprets swing as 1(0)
  - **Destructive read**: word bits now discharged

- Write: similar to cache write
  - Phase I: decode address, enable wordline
  - Phase II: enable bitlines
    - High bitlines charge corresponding capacitors

– What about **leakage over time**?

DRAM Operation II



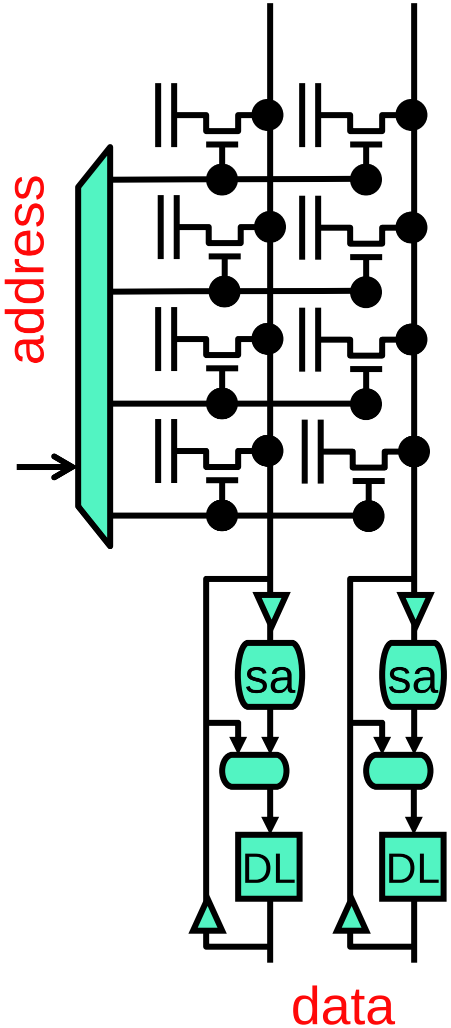
- Solution: add set of D-latches (**row buffer**)

- Read: two steps
  - Step I: read selected word into row buffer
  - Step IIA: read row buffer out to pins
  - Step IIB: write row buffer back to selected word

  \+ Solves "destructive read" problem

- Write: two steps
  - Step IA: read selected word into row buffer
  - Step IB: write data into row buffer
  - Step II: write row buffer back to selected word

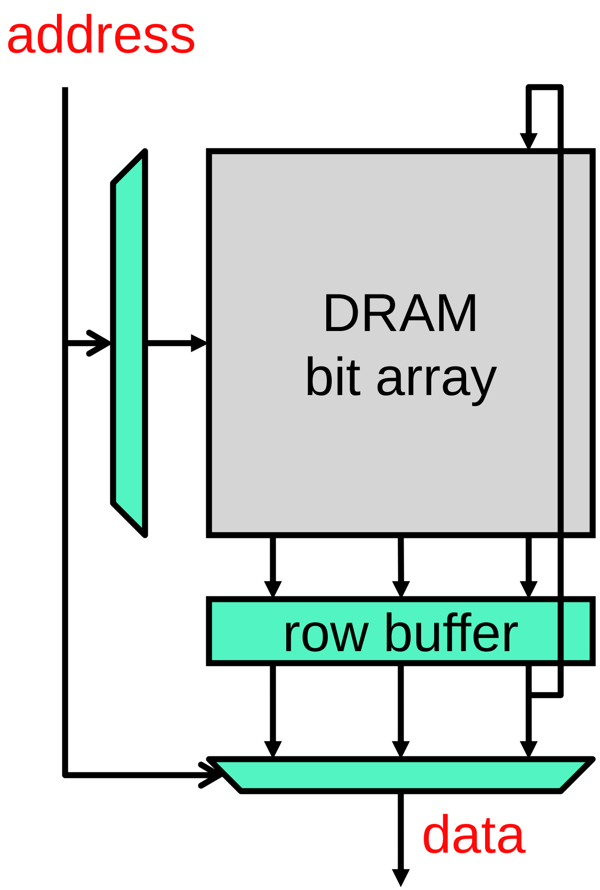
DRAM Refresh



- DRAM periodically refreshes all contents
  - Loops through all words
    - Reads word into row buffer
    - Writes row buffer back into DRAM array

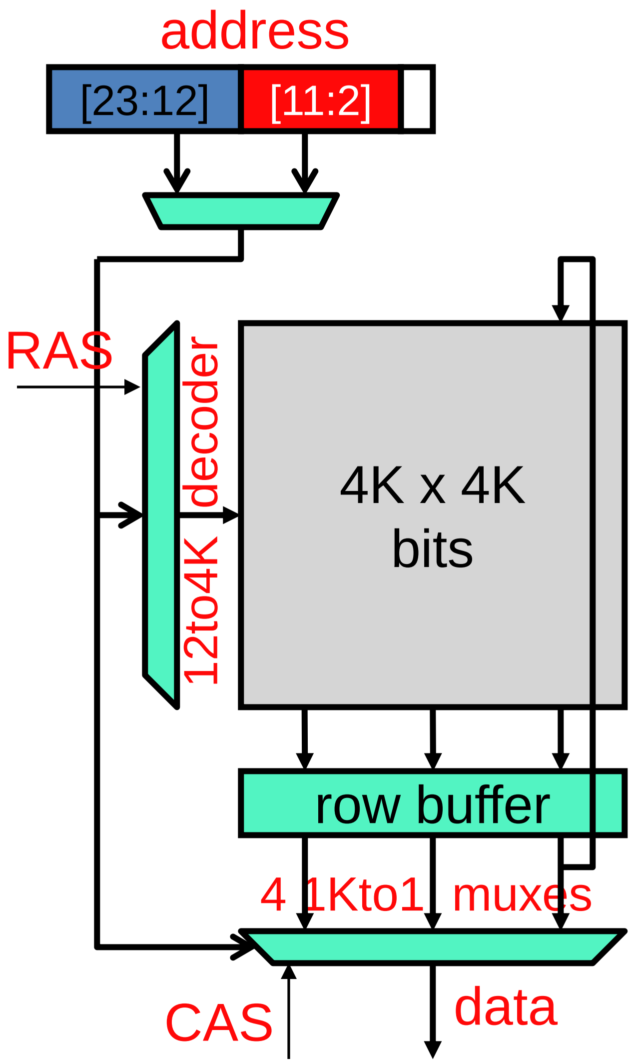
  - 1–2% of DRAM time occupied by refresh

DRAM Parameters



- DRAM parameters
  - Large capacity: e.g., 64–256Mb
    - Arranged as square
    - + Minimizes wire length
    - + Maximizes refresh efficiency
  - Narrow data interface: 1–16 bit
    - Cheap packages → few bus pins
  - Narrow address interface: N/2 bits
    - 16Mb DRAM has a 12-bit address bus
    - How does that work?

Two-Level Addressing



- **Two-level addressing**
  - Row decoder/column muxes share address lines
  - Two strobes (RAS, CAS) signal which part of address currently on bus

Access Latency and Cycle Time

- DRAM access much slower than SRAM
  - More bits $\rightarrow$ longer wires
  - Buffered access with two-level addressing
  - SRAM access latency: <1ns
  - DRAM access latency: 30–50ns

- DRAM cycle time also longer than access time
  - **Cycle time**: time between start of consecutive accesses
  - SRAM: cycle time = access time
    - Begin second access as soon as first access finishes
  - DRAM: cycle time = 2 * access time
    - Why? Can't begin new access while DRAM is refreshing row

Open v. Closed Pages

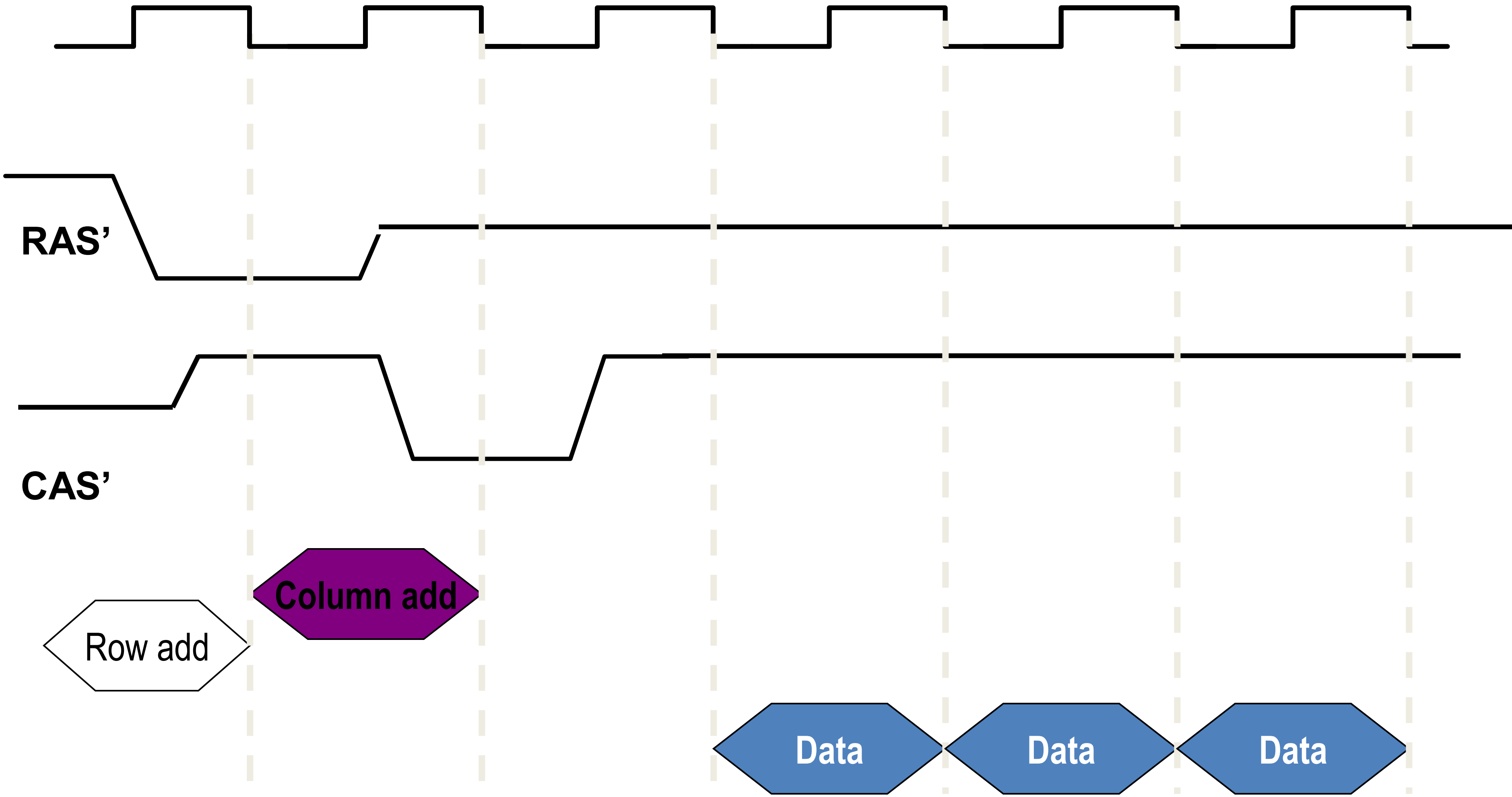
- Open Page
  - Row stays active until another row needs to be accessed
  - Acts as memory-level cache to reduce latency
  - Variable access latency complicates memory controller
  - Higher power dissipation (sense amps remain active)
- Closed Page
  - Immediately deactivate row after access
  - All accesses become Activate Row, Read/Write, Precharge

- Complex power v. performance trade off

DRAM Bandwidth

- Use multiple DRAM chips to increase bandwidth
  - Recall, access are the same size as second-level cache
  - Example, 16 2-byte wide chips for 32B access
- DRAM density increasing faster than demand
  - Result: number of memory chips per system decreasing
- Need to increase the **bandwidth per chip**
  - Especially important in game consoles
  - SDRAM ➔ DDR ➔ DDR2 ➔ FBDIMM (➔ DDR3)
  - Rambus - high-bandwidth memory
    - Used by several game consoles

Synchronous DRAM (SDRAM)



- Add Clock and Wider data!
- Also multiple transfers per RAS/CAS

Enhanced SDRAM & DDR

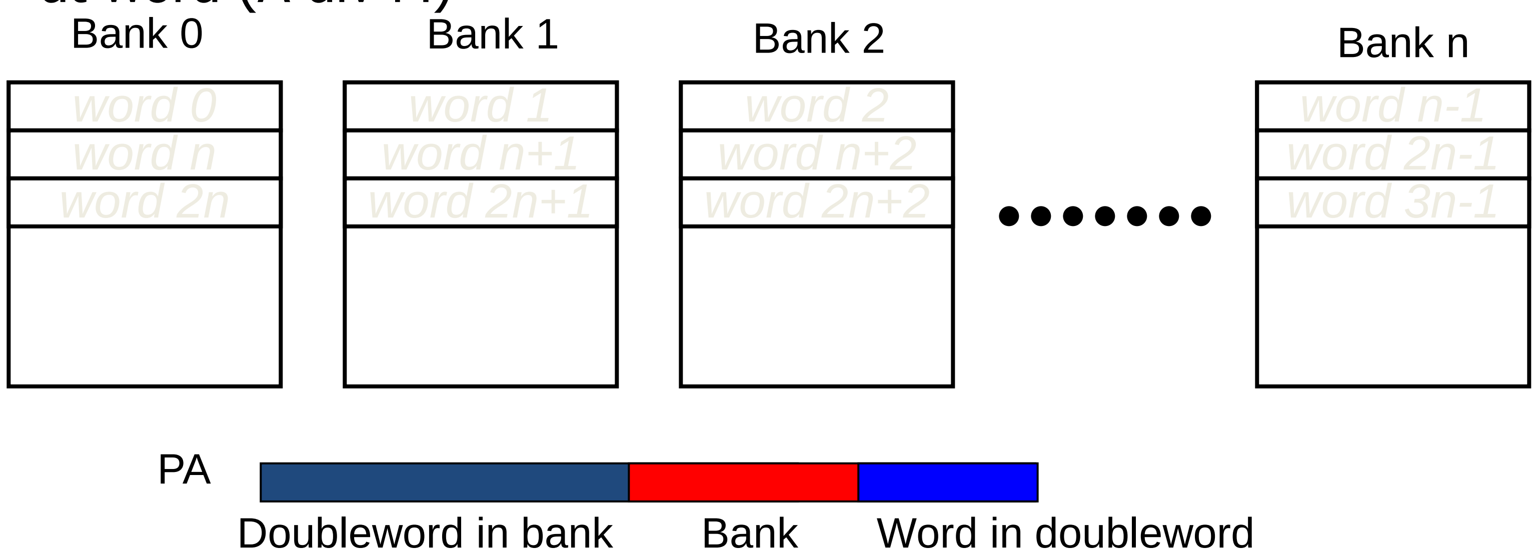
- Evolutionary Enhancements on SDRAM:

1. ESDRAM (Enhanced): Overlap row buffer access with refresh

2. DDR (Double Data Rate): Transfer on both clock edges
3. DDR2's small improvements
   lower voltage, on-chip termination, driver calibration
   prefetching, conflict buffering
4. DDR3, more small improvements
   lower voltage, 2X speed, 2X prefetching,
   2X banks, "fly-by topology", automatic calibration

Interleaved Main Memory

- Divide memory into M banks and “interleave” addresses across them, so word A is
  - in bank (A mod M)
  - at word (A div M)

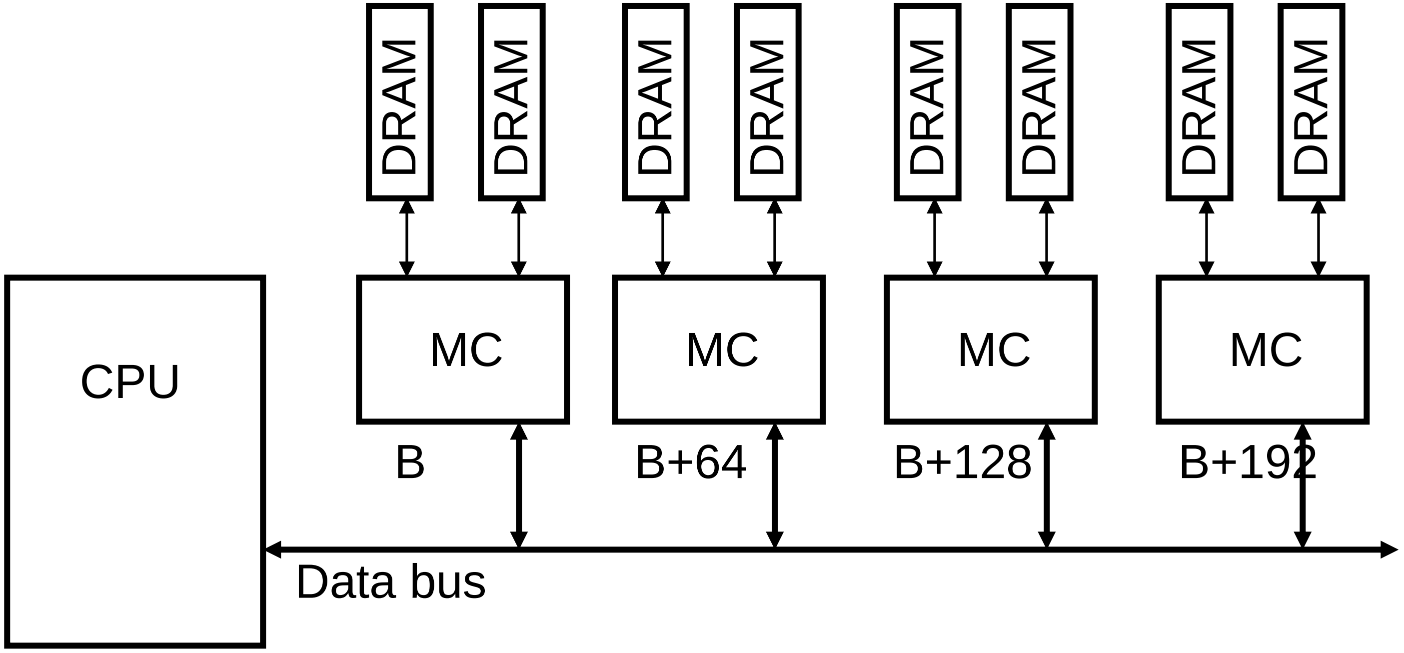


Interleaved memory increases memory BW without wider bus

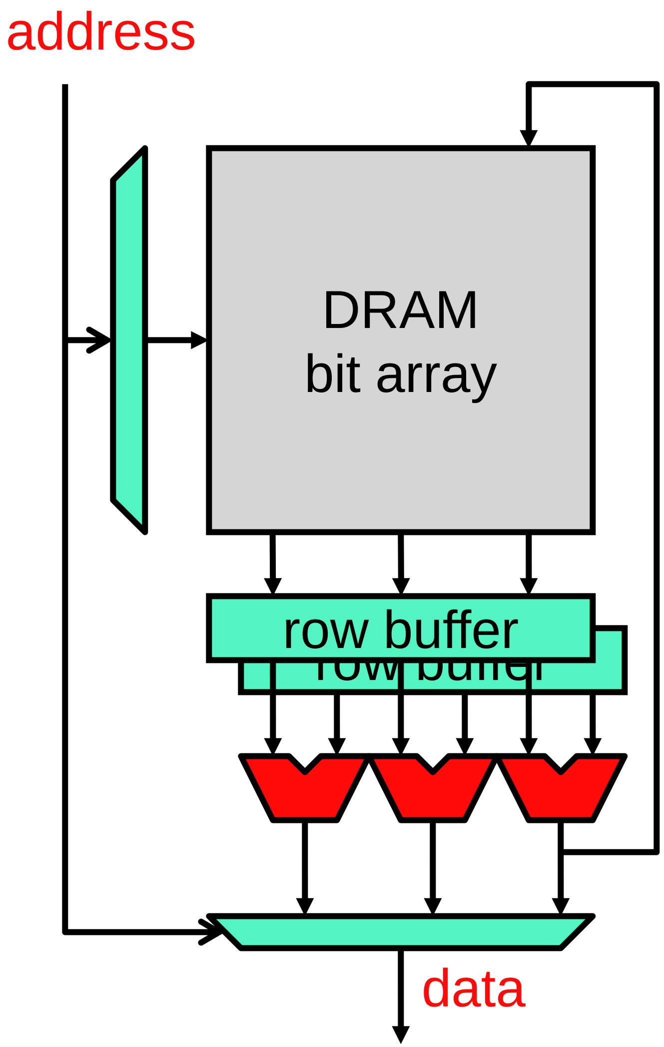
- *Use parallelism in memory banks to hide memory latency*

Block interleaved memory systems

- Cache blocks map to separate memory controllers
  - Interleave across DRAMs w/i a MC
  - Interleave across intra-DRAM banks w/i a DRAM



Research: Processing in Memory



- **Processing in memory**
  - Embed some ALUs in DRAM
    - Picture is logical, not physical
  - Do computation in DRAM rather than…
    - Move data to from DRAM to CPU
    - Compute on CPU
    - Move data from CPU to DRAM
  - Will come back to this in "vectors" unit

  - E.g.,: IRAM: intelligent RAM
    - Berkeley research project
    - [Patterson+,ISCA'97]
    - **Very hot again**

Memory Hierarchy Review

- Storage: registers, **memory**, disk
  - Memory is the fundamental element
- Memory component performance
  - $t_{avg} = t_{hit} + \%_{miss} * t_{miss}$
  - Can't get both low t_{hit} and $\%_{miss}$ in a single structure
- Memory hierarchy
  - Upper components: small, fast, expensive
  - Lower components: big, slow, cheap
  - t_{avg} of hierarchy is close to t_{hit} of upper (fastest) component
    - 10/90 rule: 90% of stuff found in fastest component
  - **Temporal/spatial locality**: automatic up-down data movement

Bonus

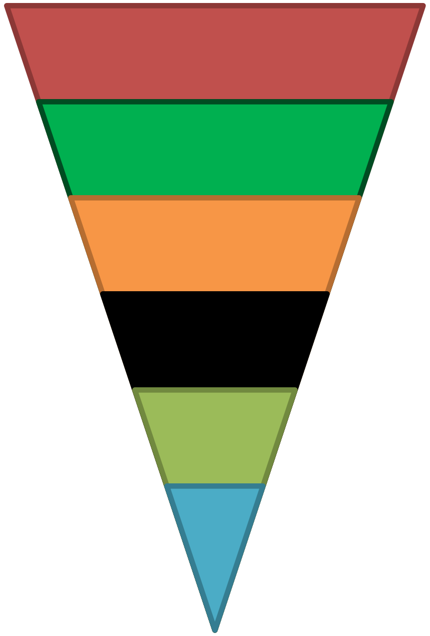
The DRAM Subsystem

Onur Mutlu….

Slide History/Attribution Diagram:

DRAM Subsystem Organization

- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column



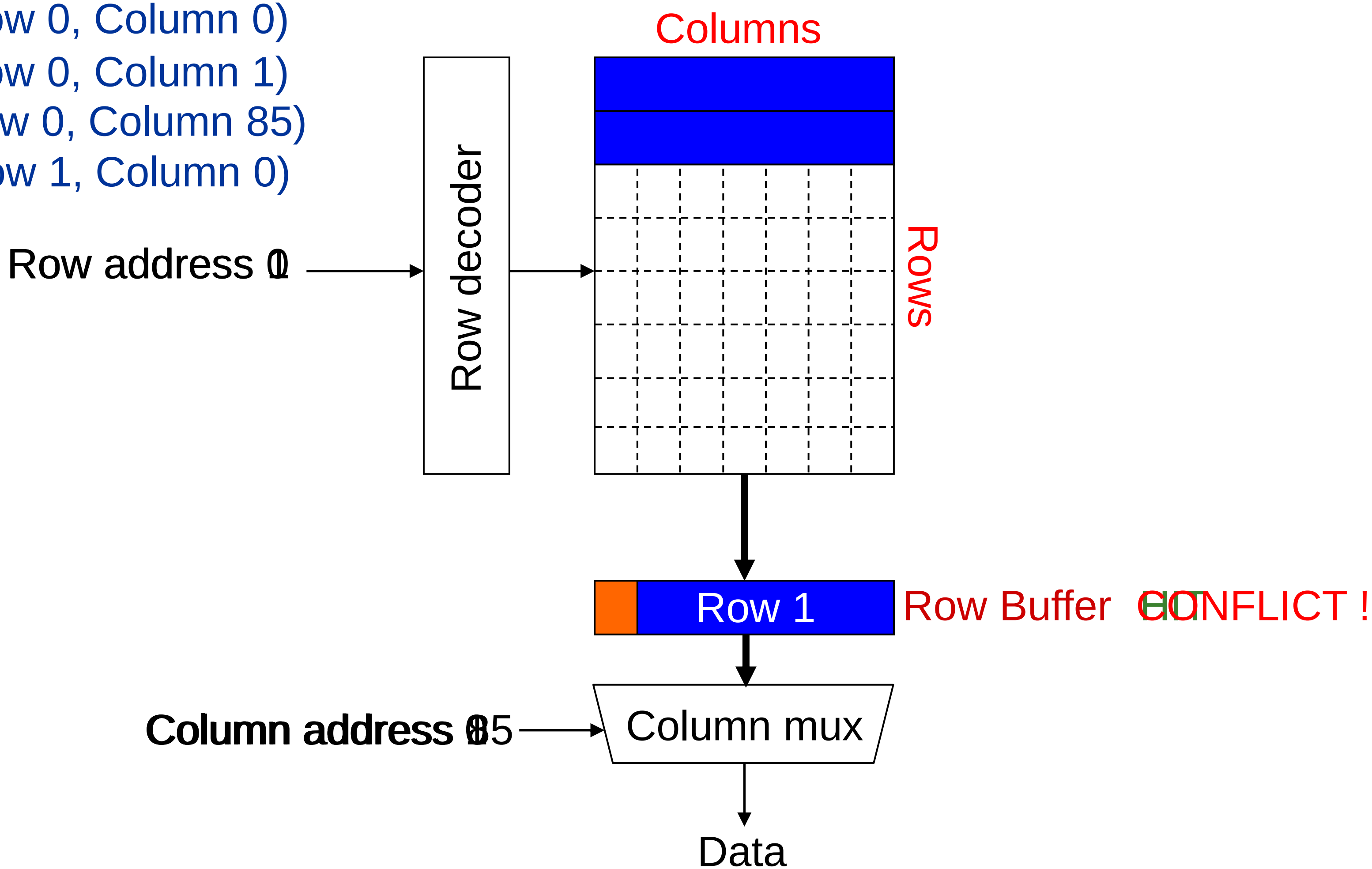
Page Mode DRAM

- A DRAM bank is a 2D array of cells: rows x columns
- A "DRAM row" is also called a "DRAM page"
- "Sense amplifiers" also called "row buffer"

- Each address is a <row,column> pair
- Access to a "closed row"
  - Activate command opens row (placed into row buffer)
  - Read/write command reads/writes column in the row buffer
  - Precharge command closes the row and prepares the bank for next access
- Access to an "open row"
  - No need for activate command

DRAM Bank Operation

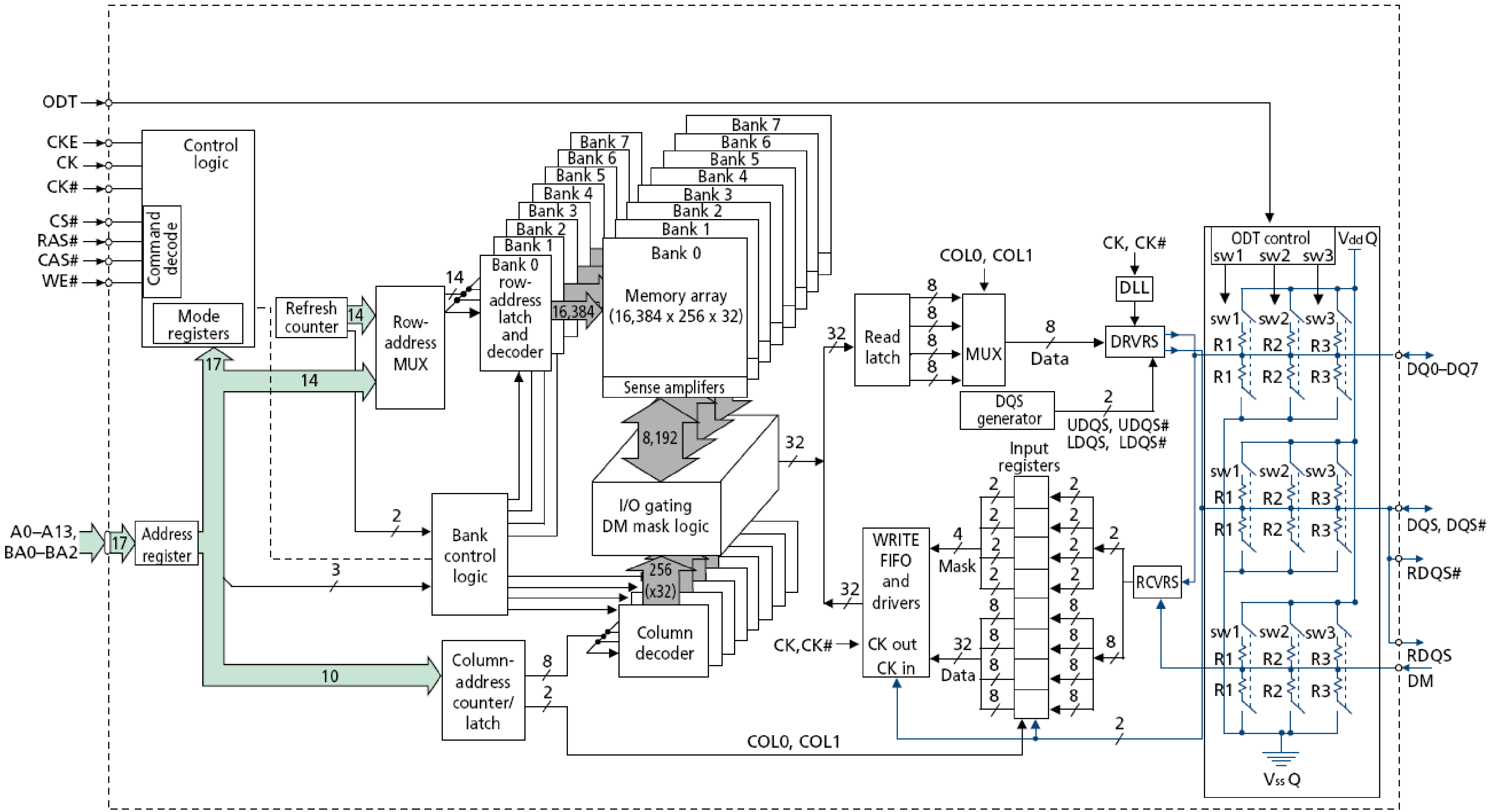
Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



The DRAM Chip

- Consists of multiple banks (2-16)
- Banks share command/address/data buses
- The chip itself has a narrow interface (4-16 bits per read)

128M x 8-bit DRAM Chip



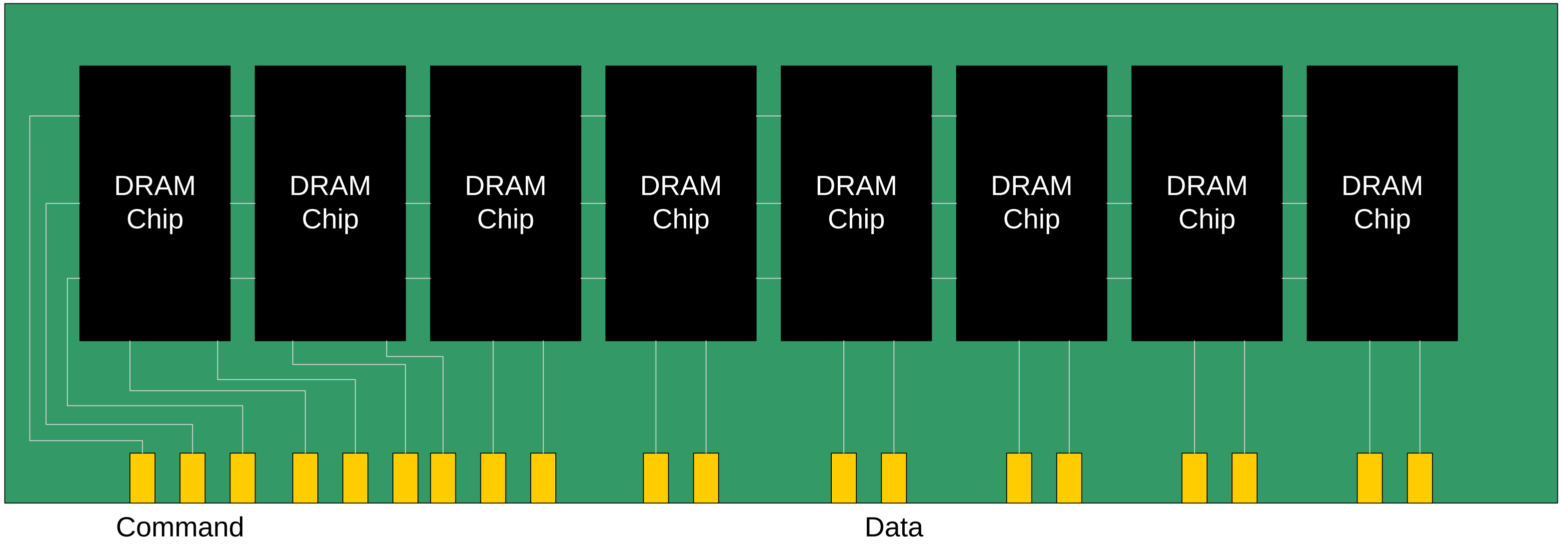
DRAM Rank and Module

- Rank: Multiple chips operated together to form a wide interface
- All chips comprising a rank are controlled at the same time
  - Respond to a single command
  - Share address and command buses, but provide different data

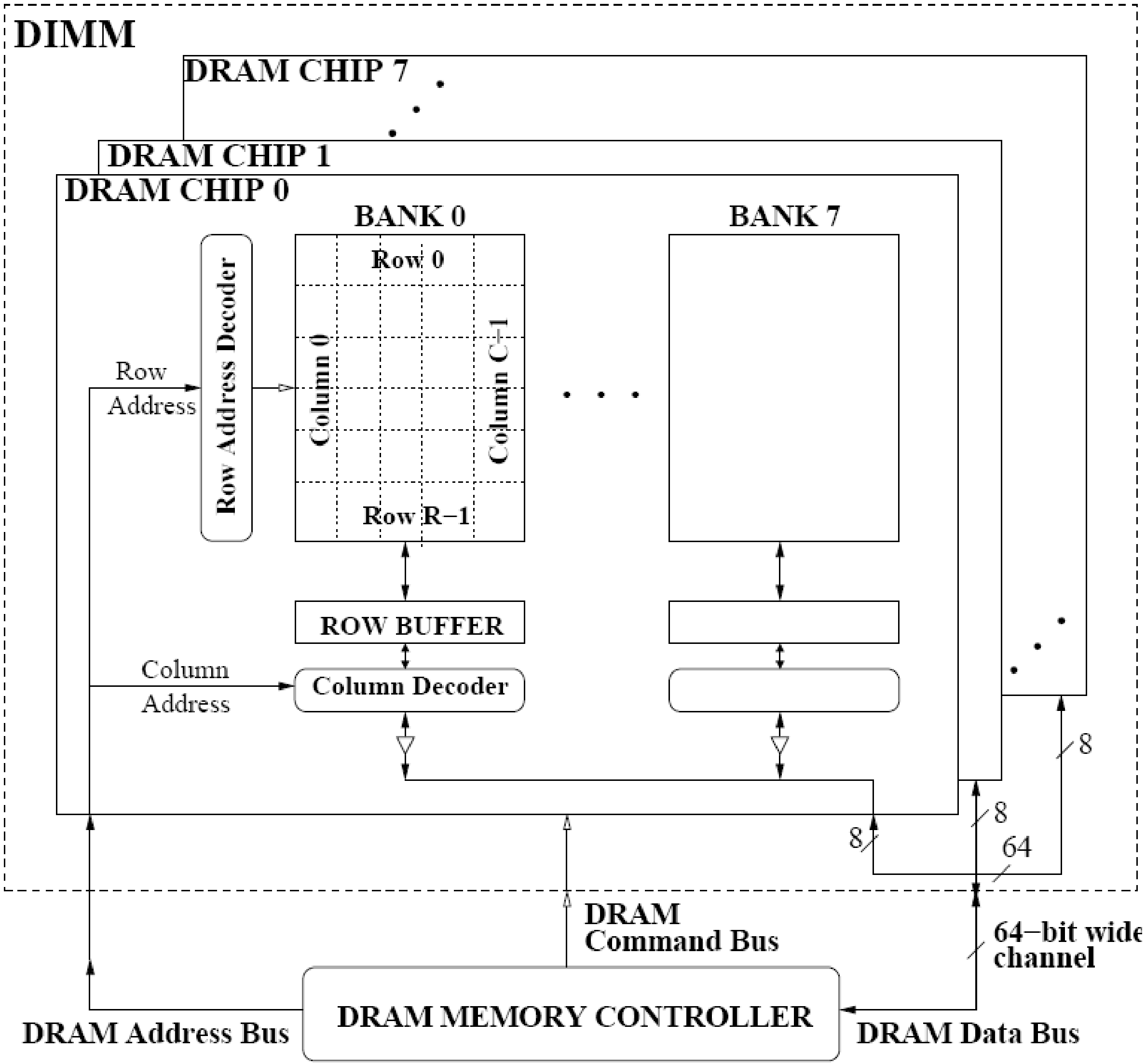
- A DRAM module consists of one or more ranks
  - E.g., DIMM (dual inline memory module)
  - This is what you plug into your motherboard

- If we have chips with 8-bit interface, to read 8 bytes in a single access, use 8 chips in a DIMM

A 64-bit Wide DIMM (One Rank)

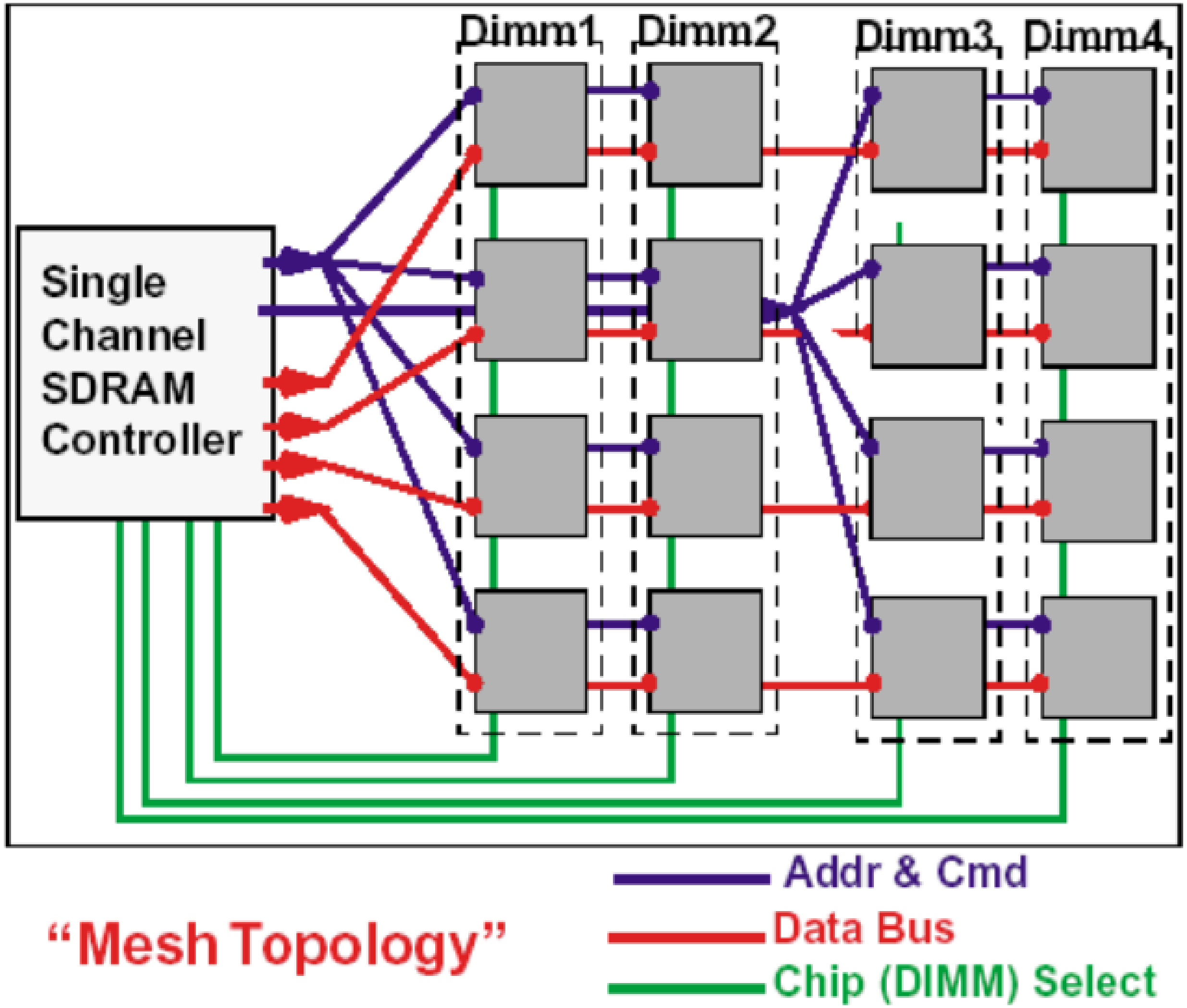


A 64-bit Wide DIMM (One Rank)



- Advantages:
  - Acts like a high-capacity DRAM chip with a wide interface
  - Simplicity: memory controller does not need to deal with individual chips
- Disadvantages:
  - Granularity: Accesses cannot be smaller than the interface width

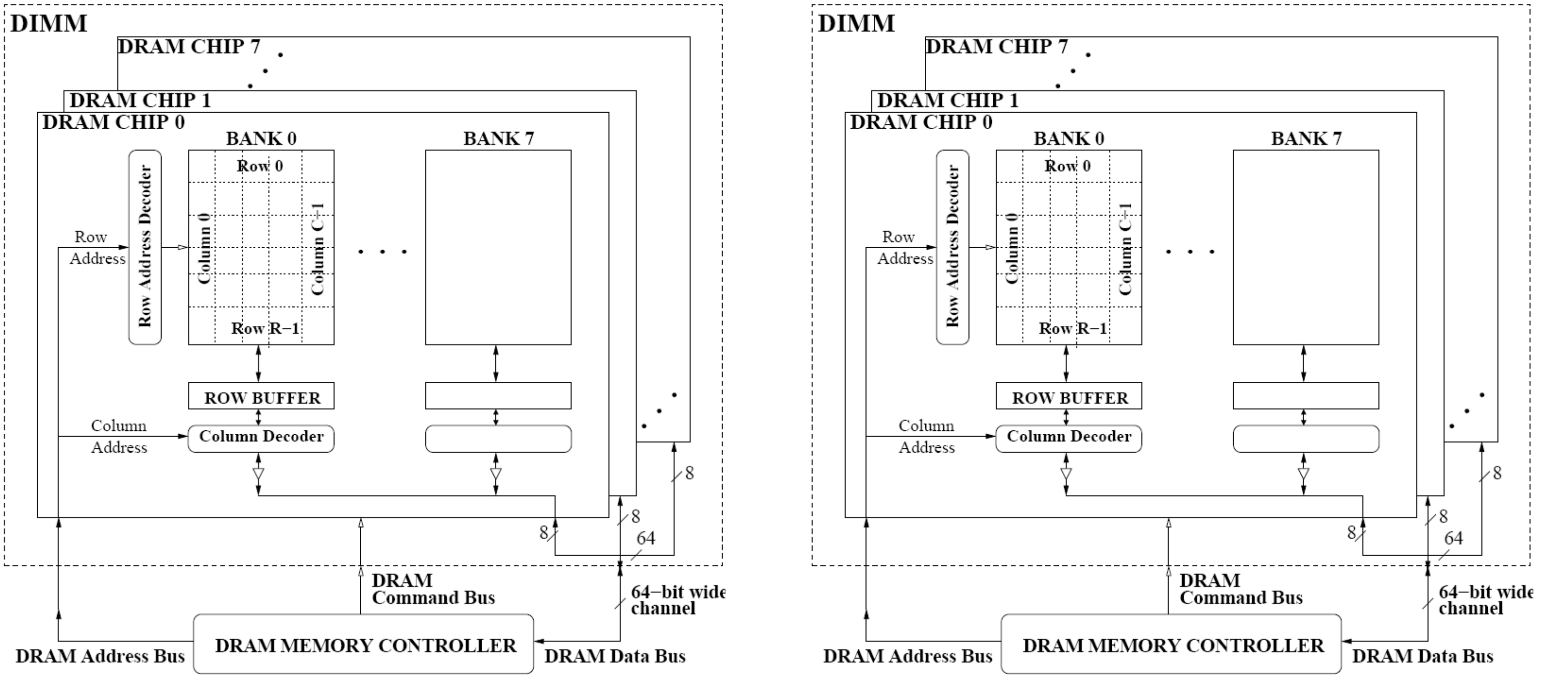
Multiple DIMMs



"Mesh Topology"

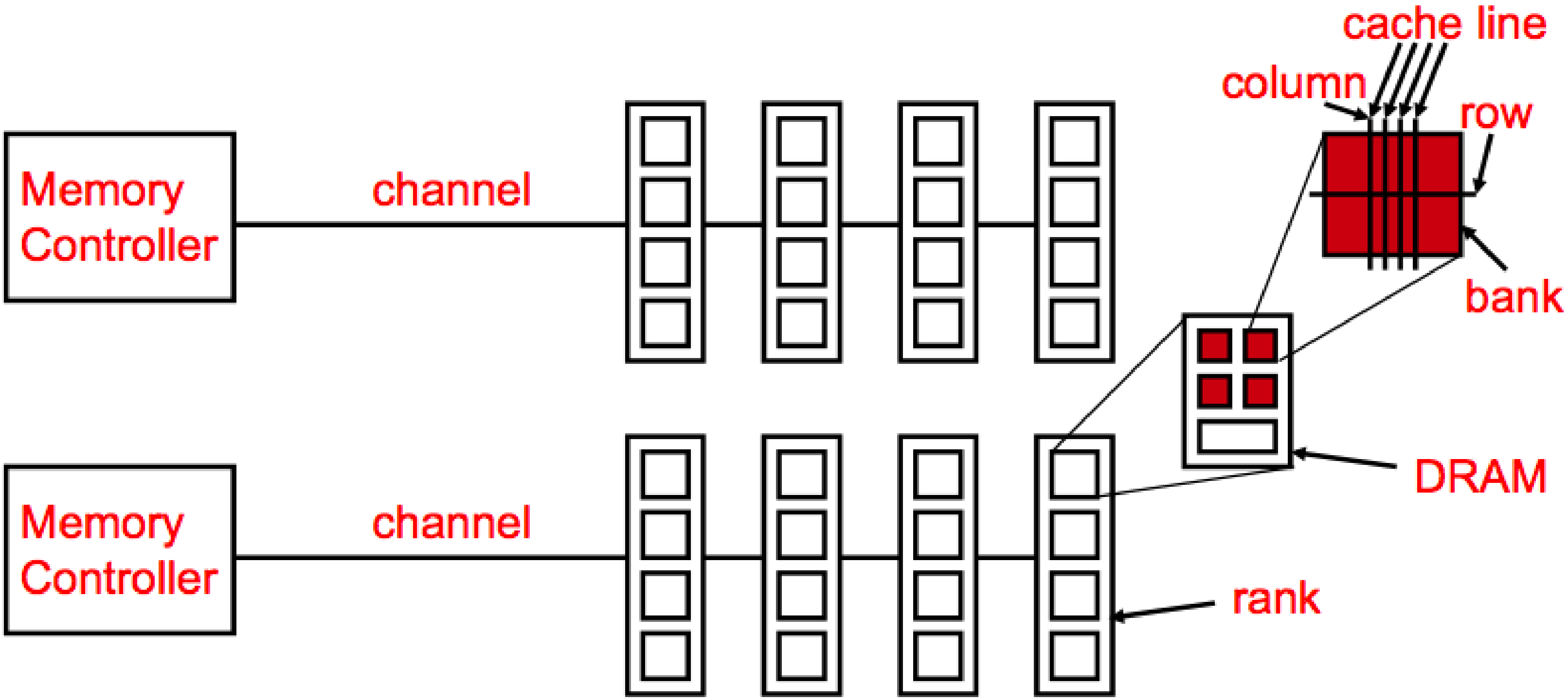
- Advantages:
  - Enables even higher capacity
- Disadvantages:
  - Interconnect complexity and energy consumption can be high

DRAM Channels



- 2 Independent Channels: 2 Memory Controllers (Above)
- 2 Dependent/Lockstep Channels: 1 Memory Controller with wide interface (Not shown above)

Generalized Memory Structure



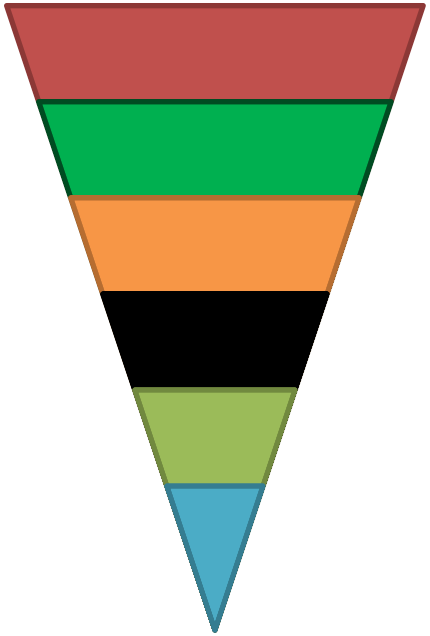
The DRAM Subsystem
The Top Down View

Onur Mutlu….

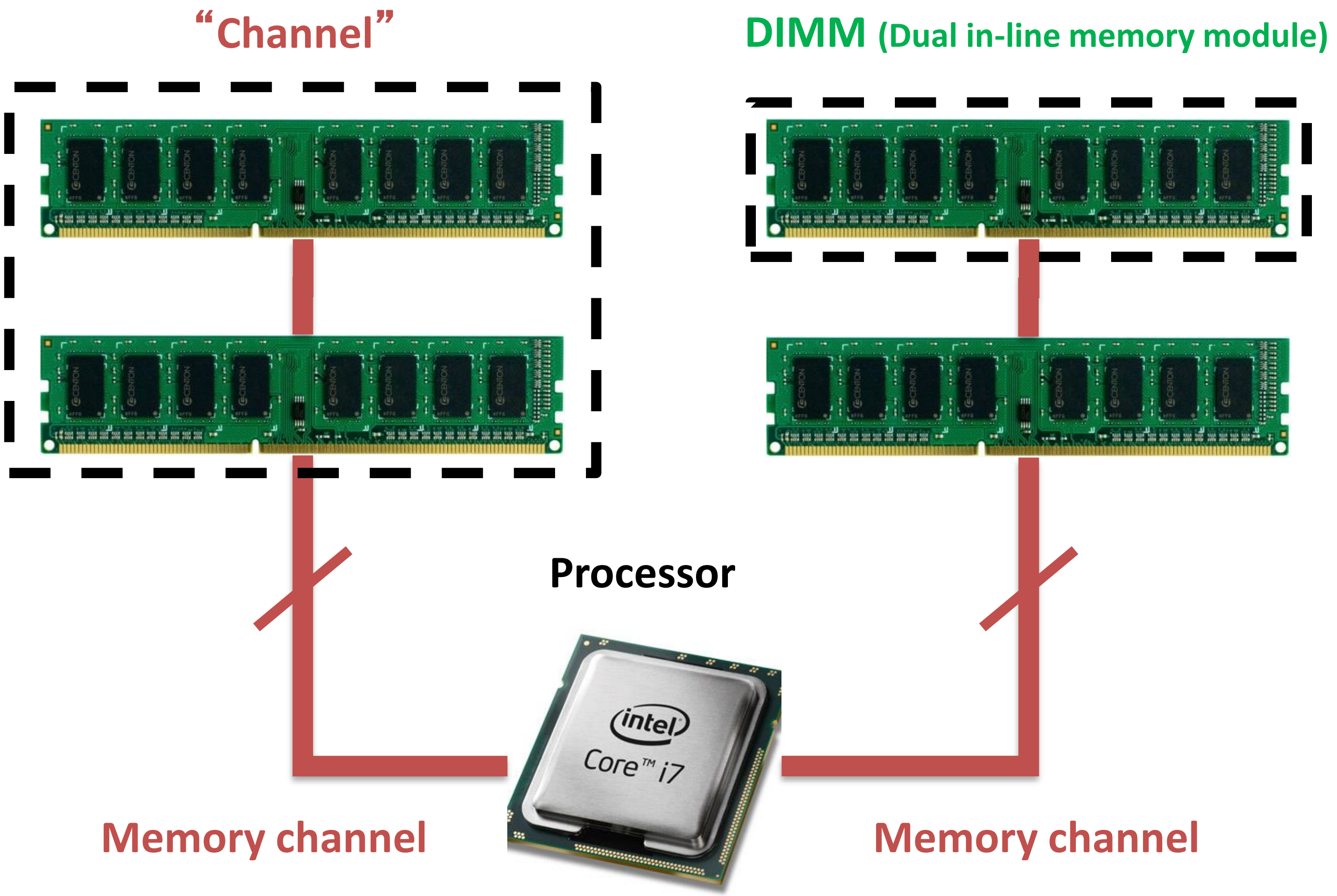
Slide History/Attribution Diagram:

DRAM Subsystem Organization

- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column

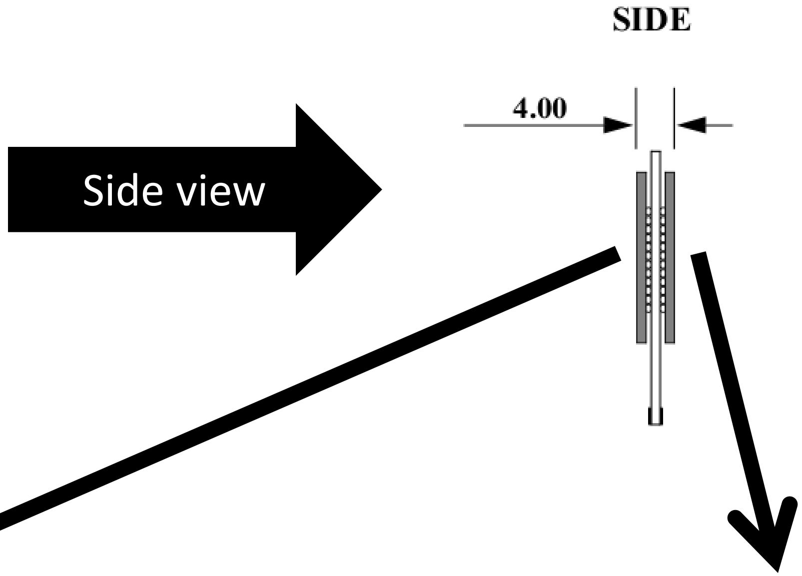


The DRAM subsystem

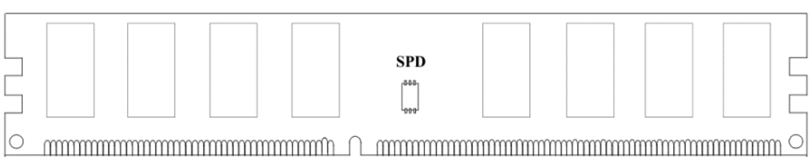


Breaking down a DIMM

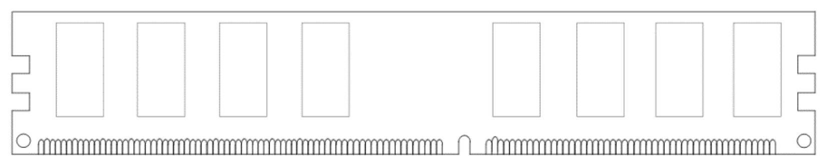
DIMM (Dual in-line memory module)



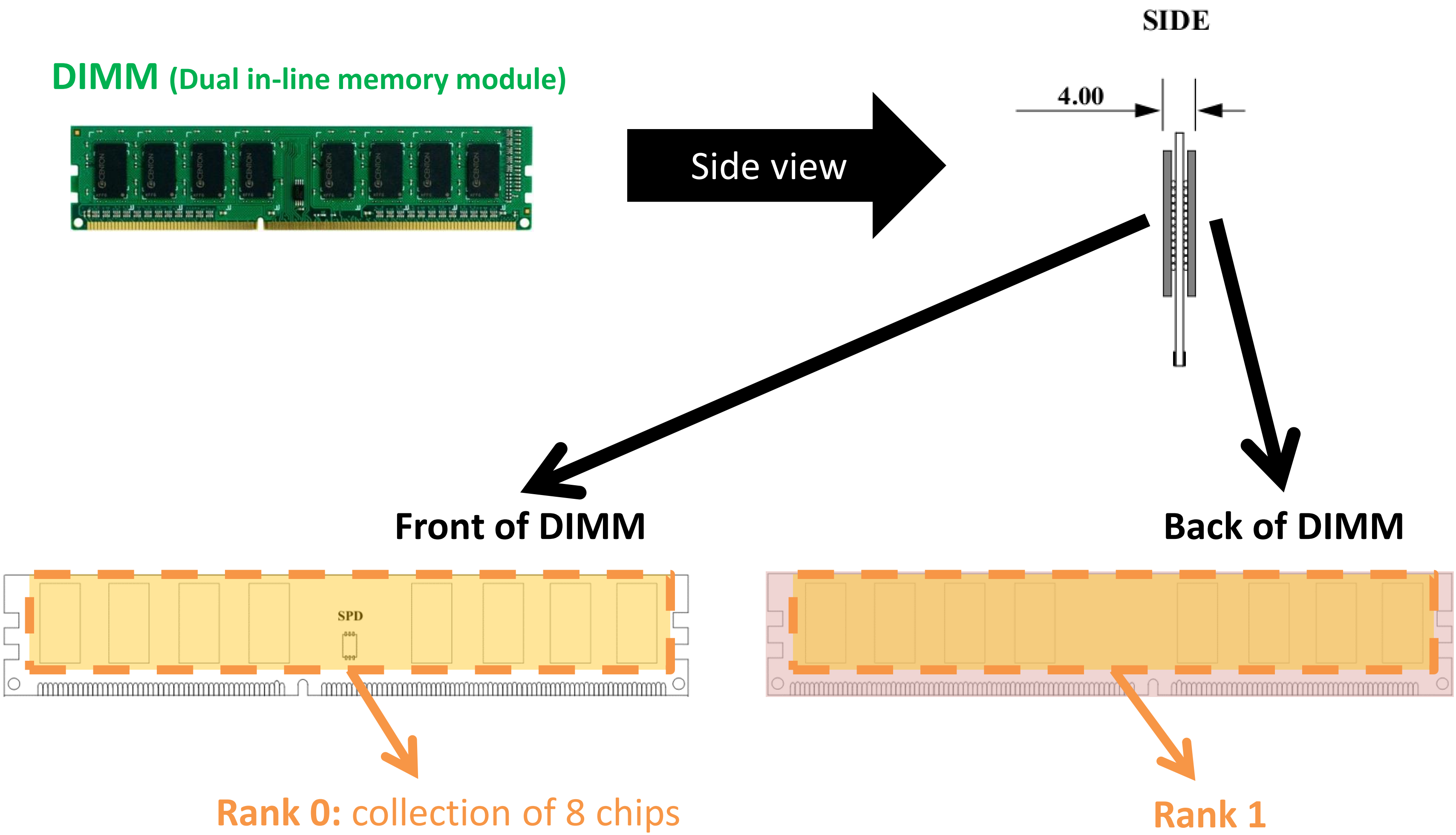
Front of DIMM



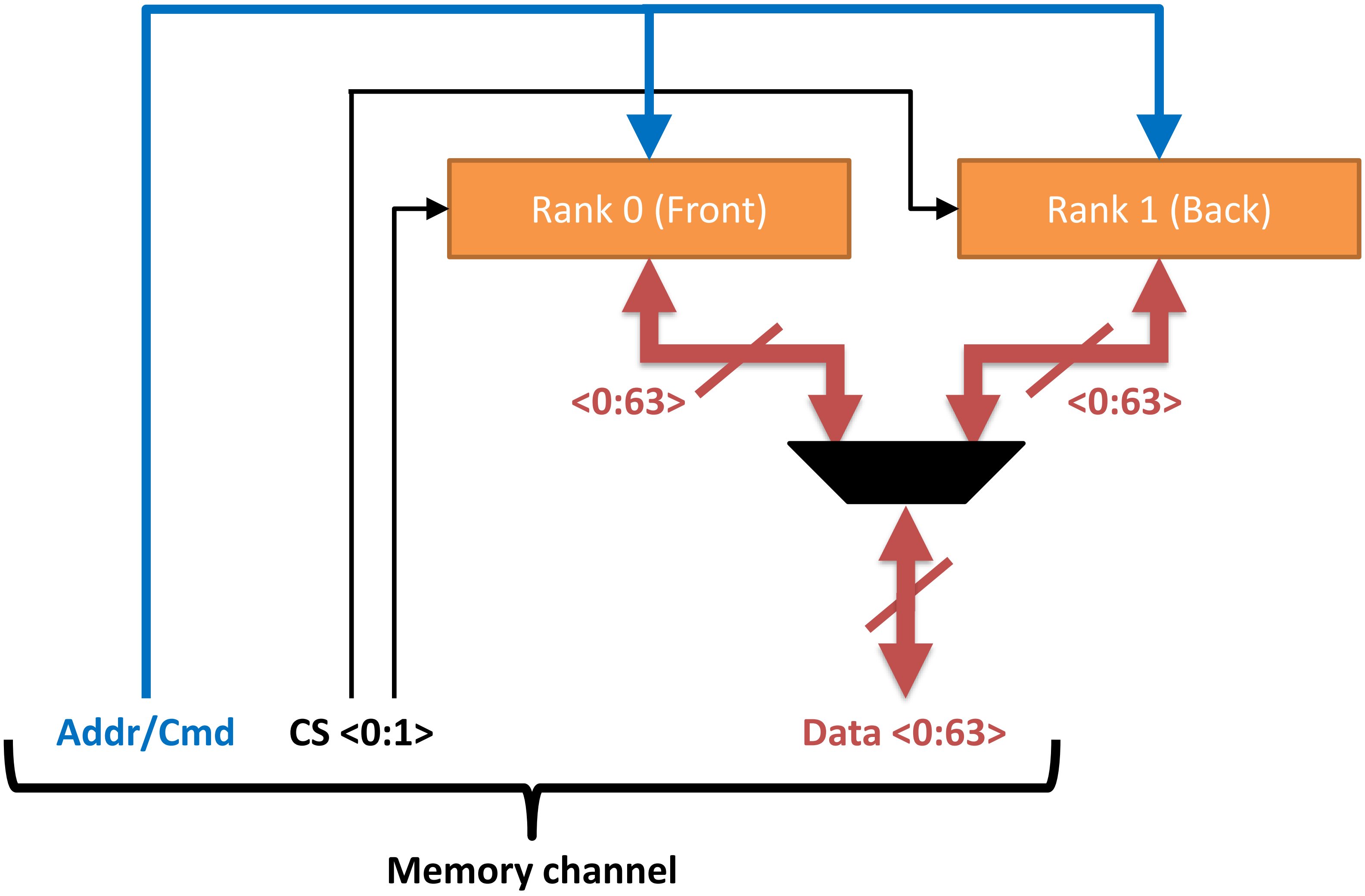
Back of DIMM



Breaking down a DIMM

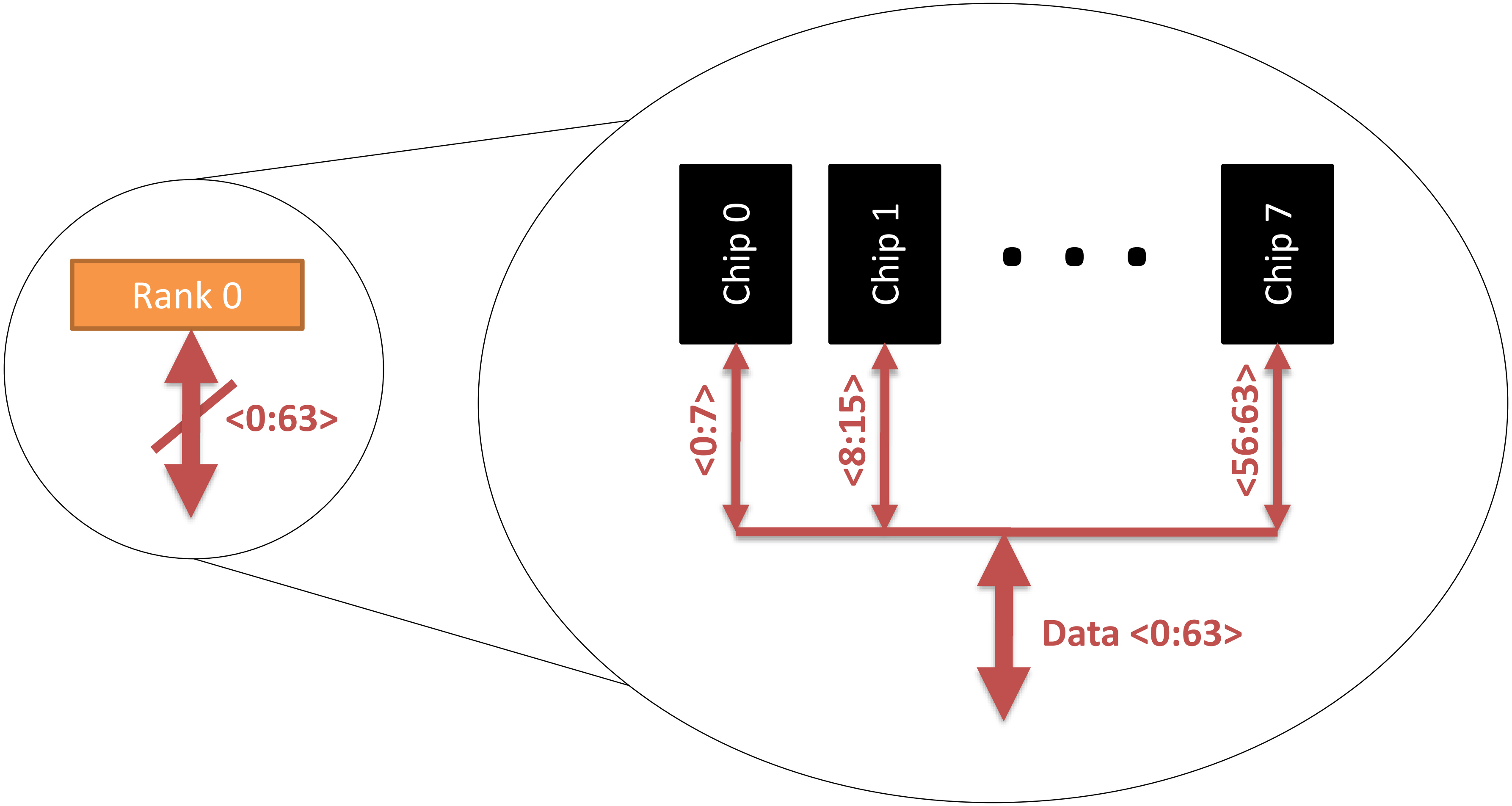


Rank

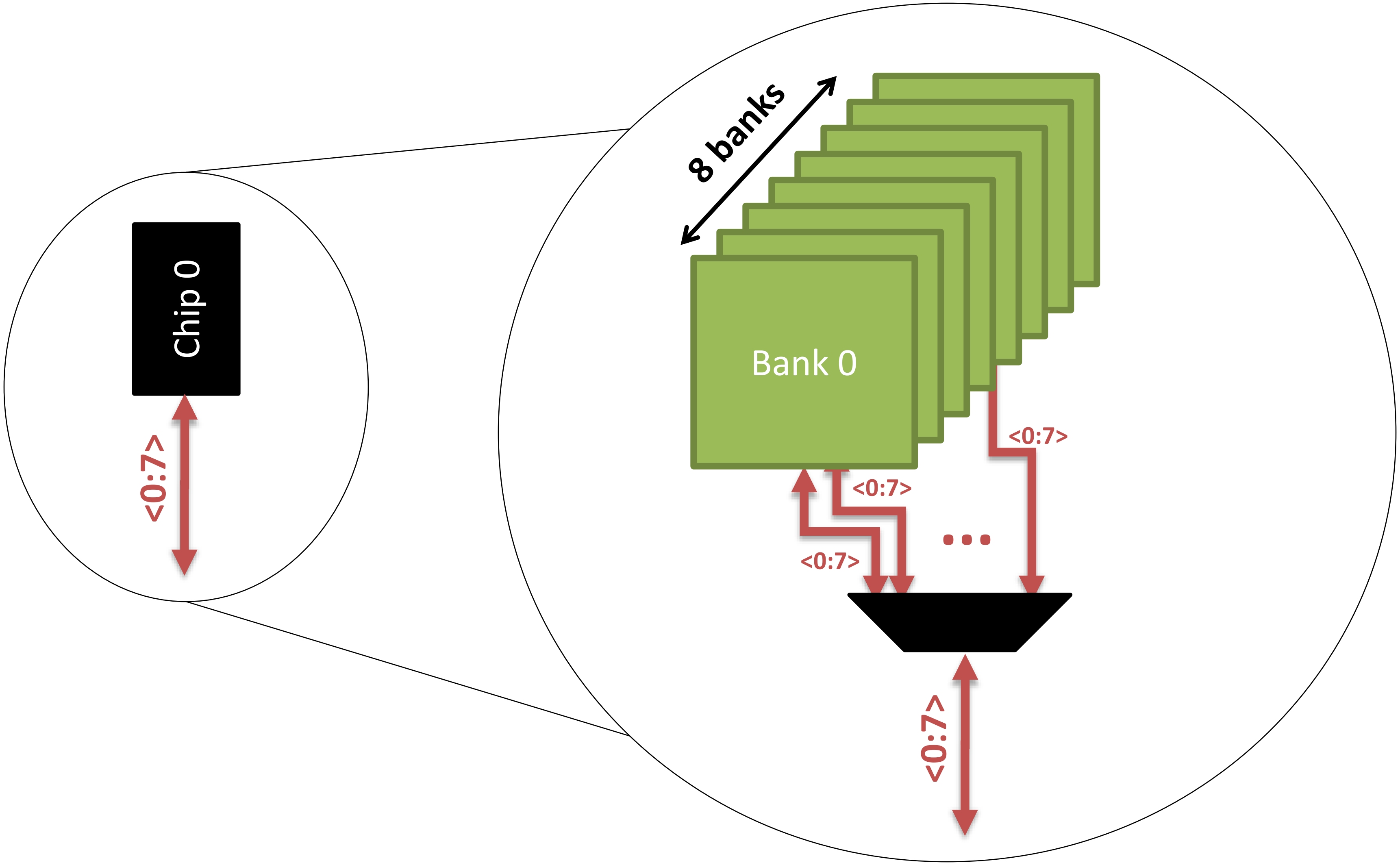


Rank 0 (Front)
Rank 1 (Back)
<0:63>
<0:63>
Addr/Cmd
CS <0:1>
Data <0:63>
Memory channel

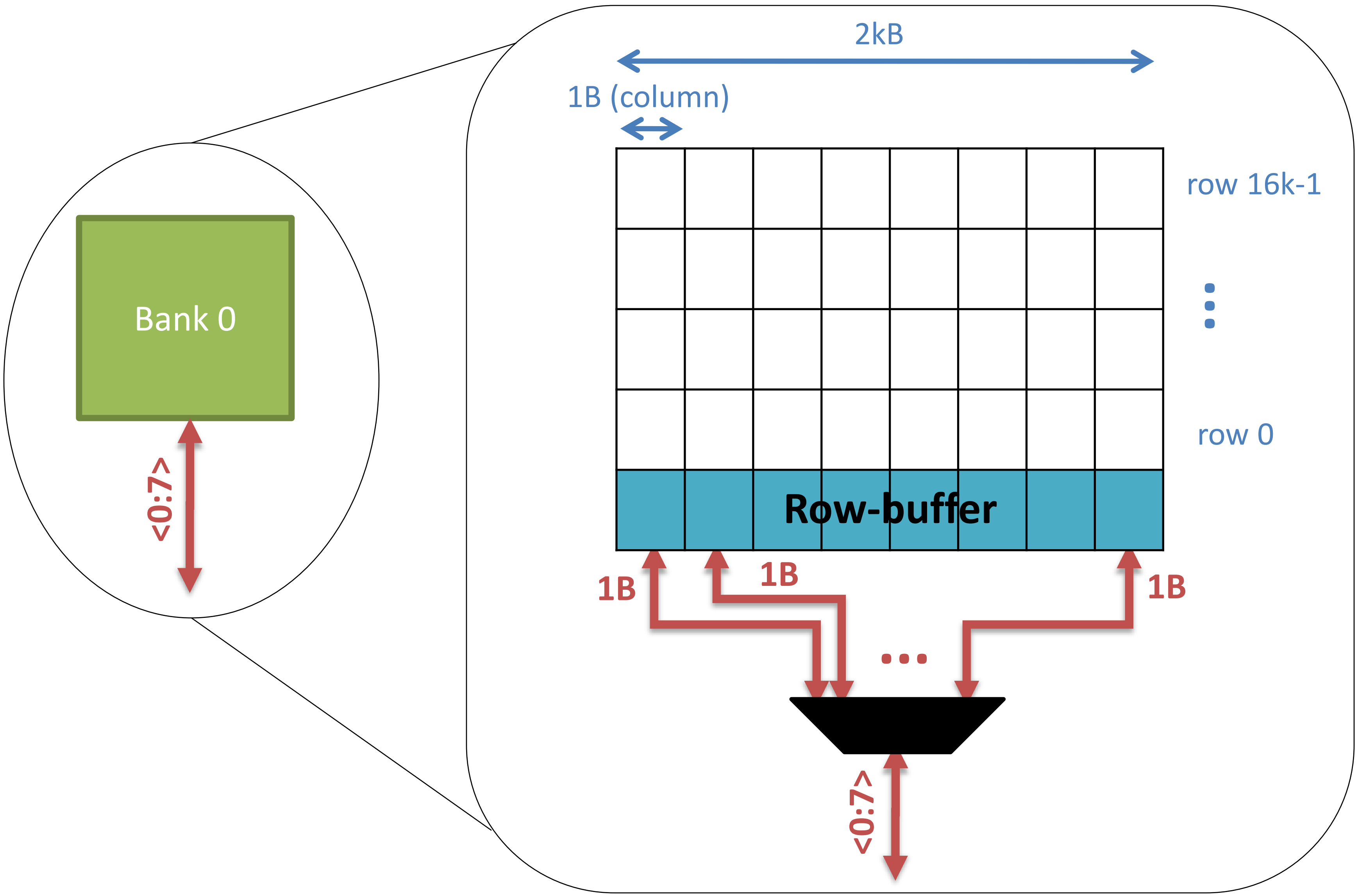
Breaking down a Rank



Breaking down a Chip

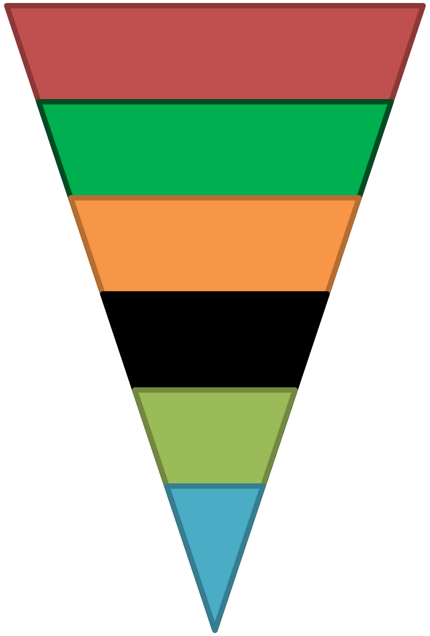


Breaking down a Bank

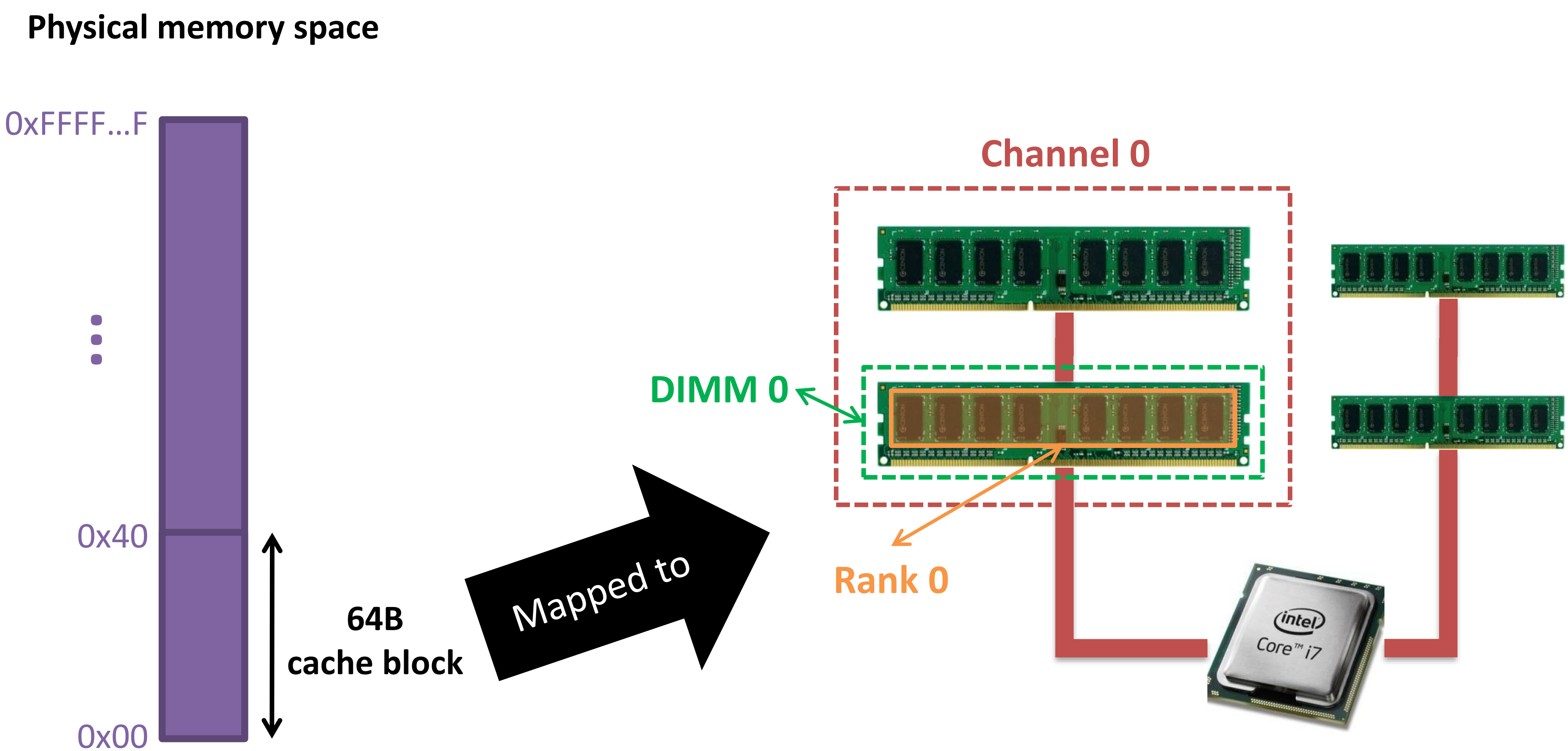


DRAM Subsystem Organization

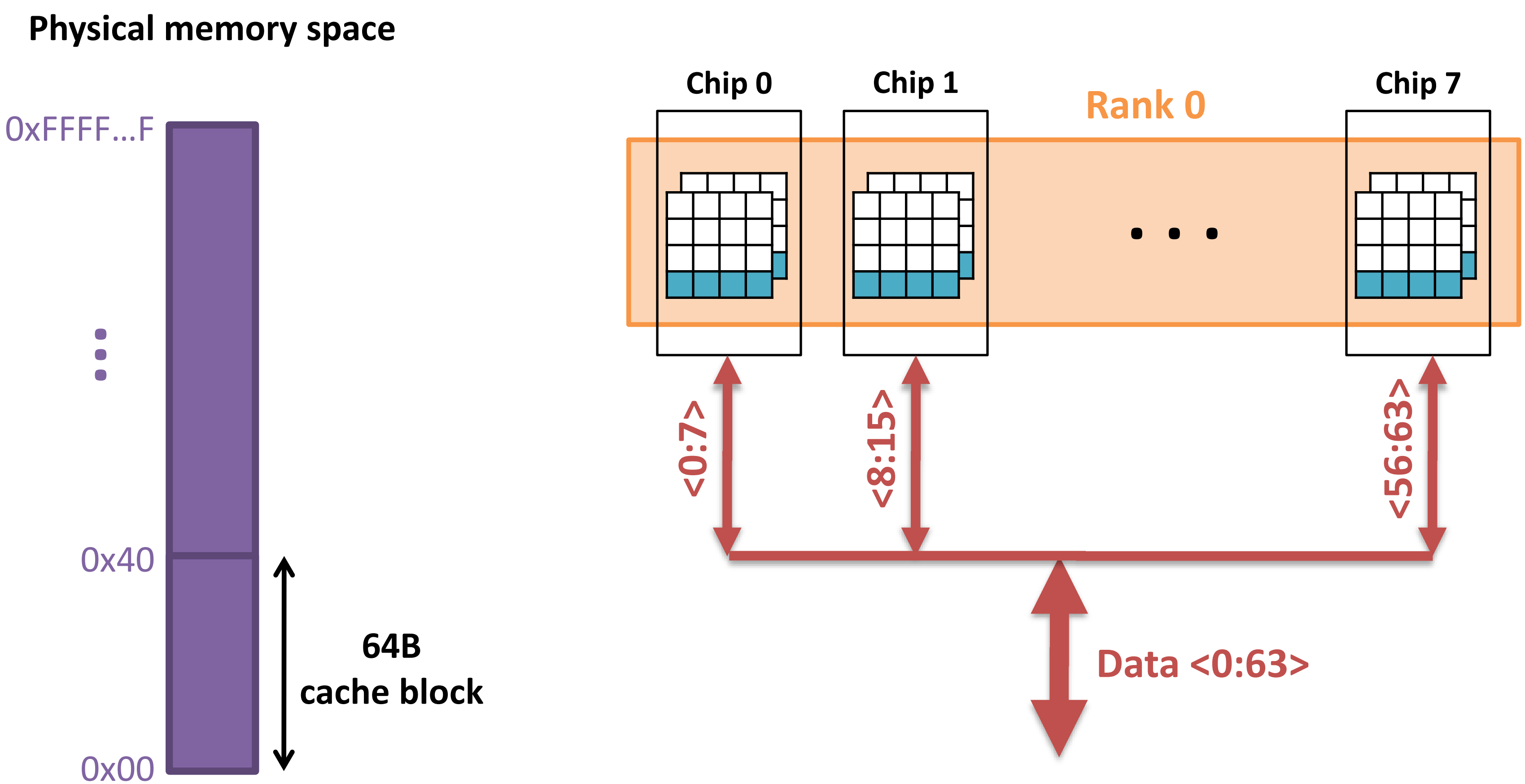
- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column



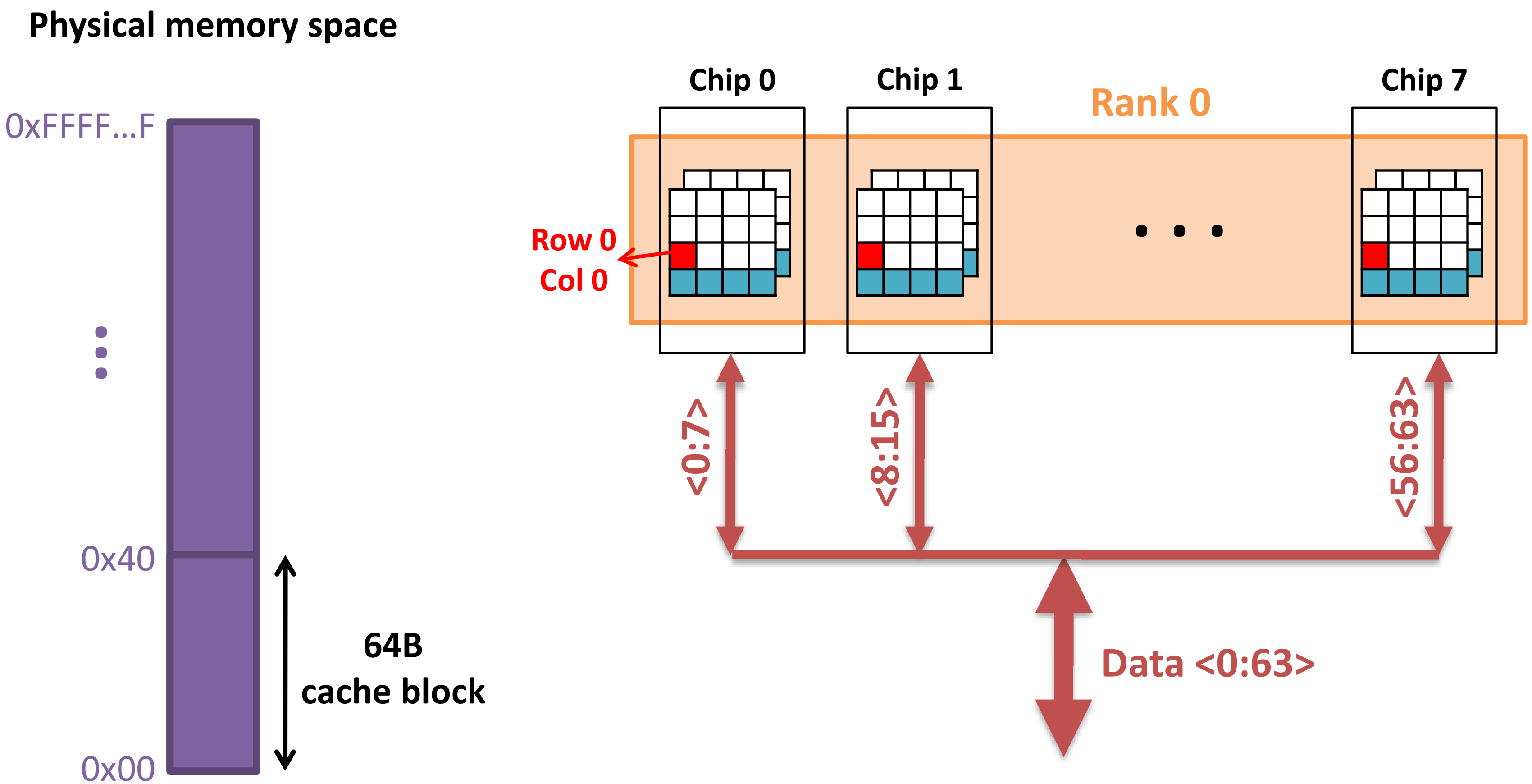
Example: Transferring a cache block



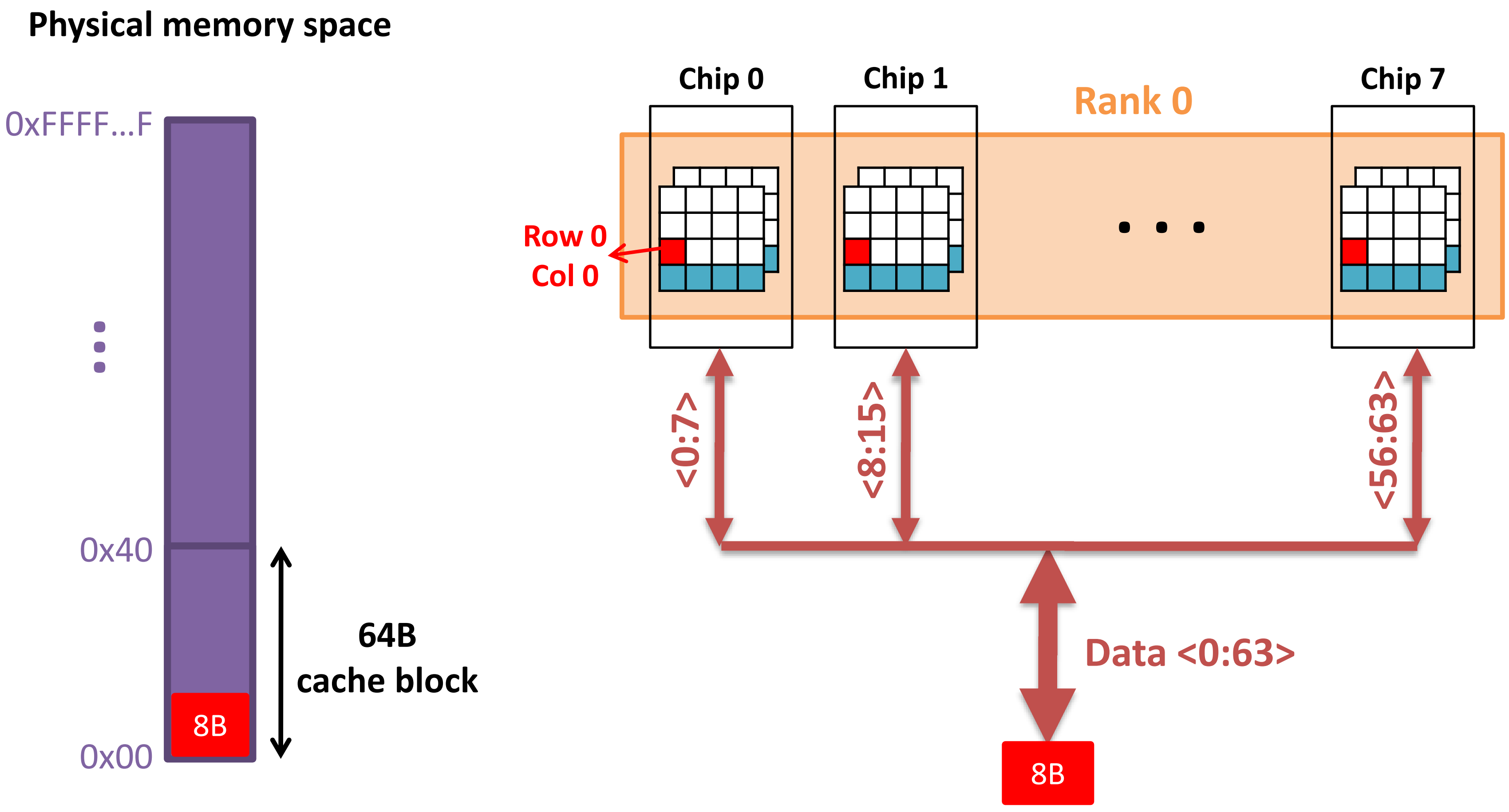
Example: Transferring a cache block



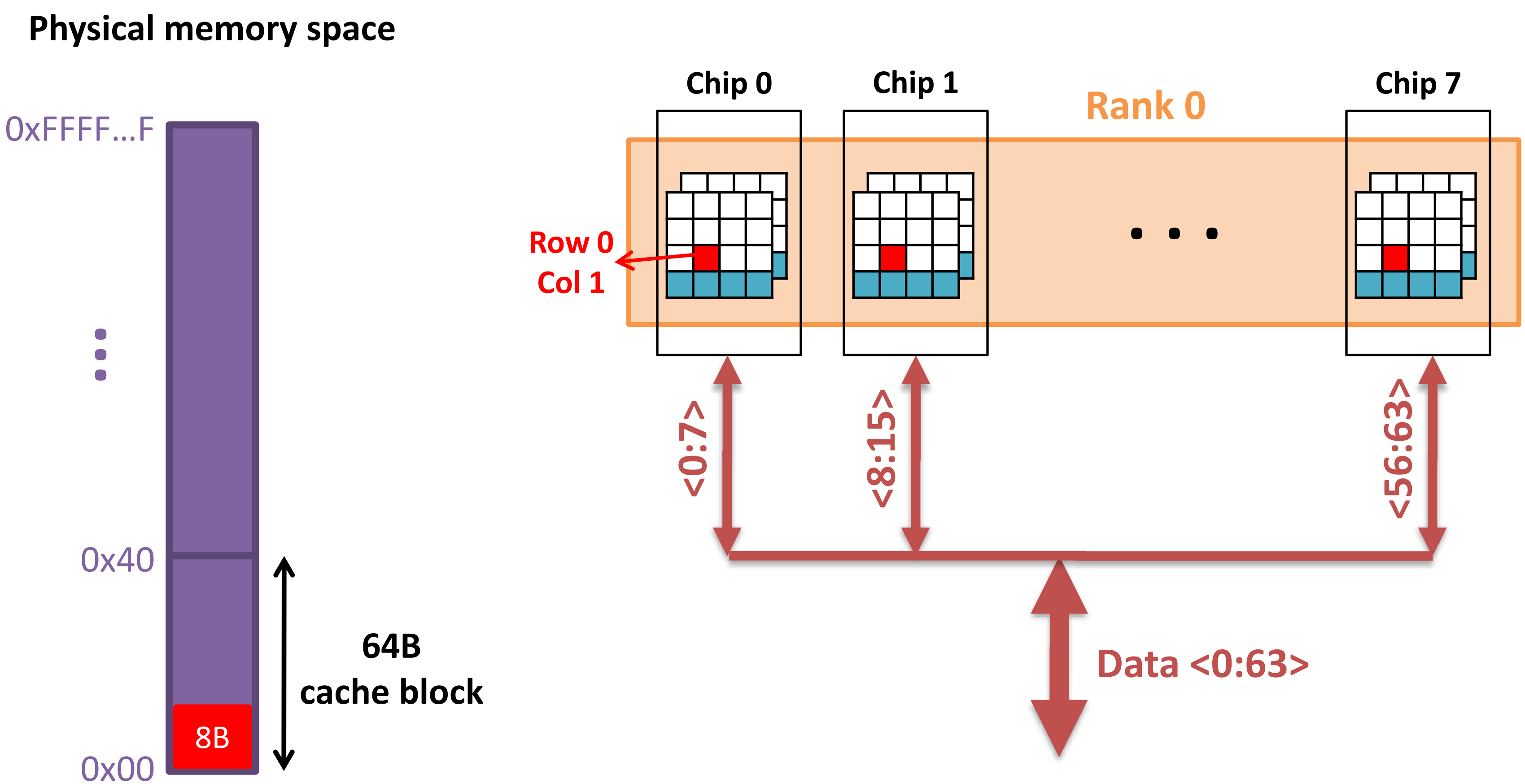
Example: Transferring a cache block



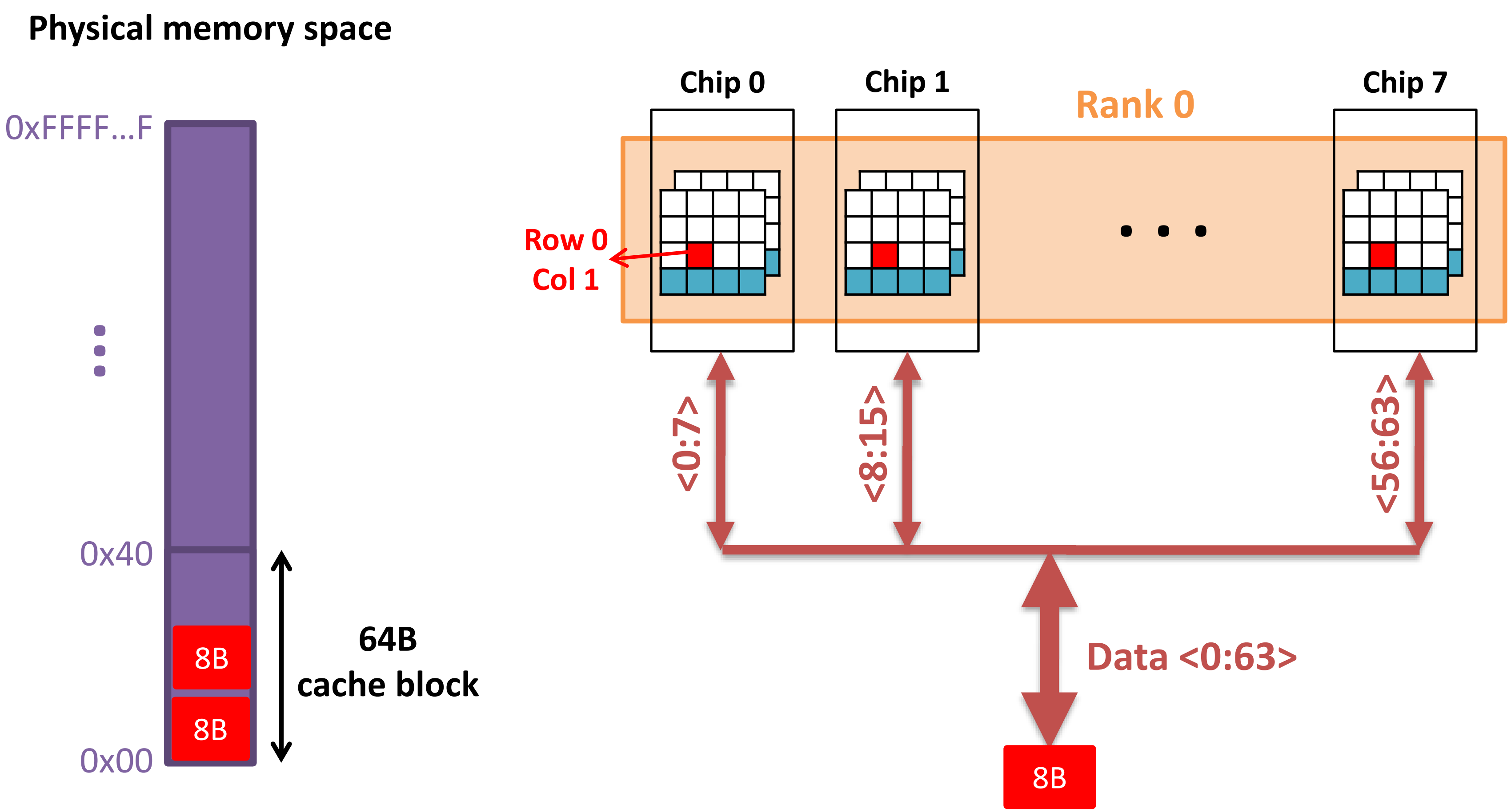
Example: Transferring a cache block



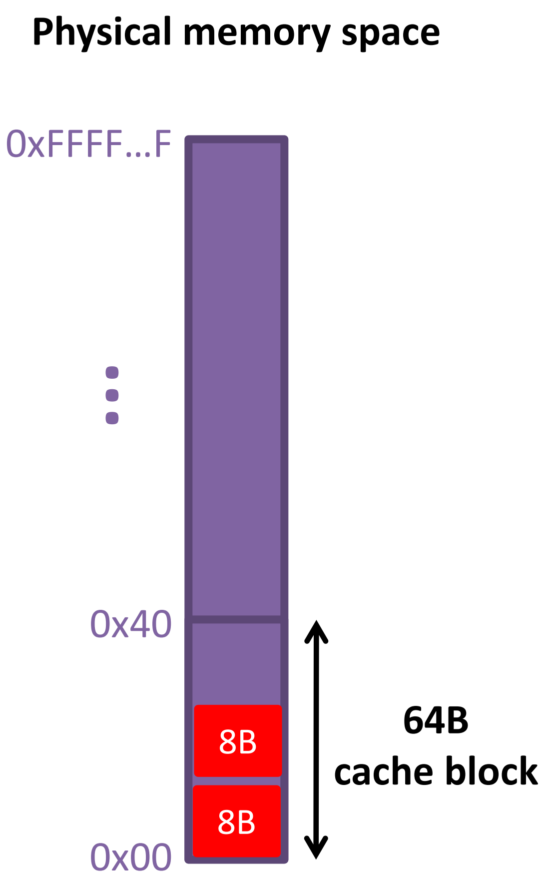
Example: Transferring a cache block

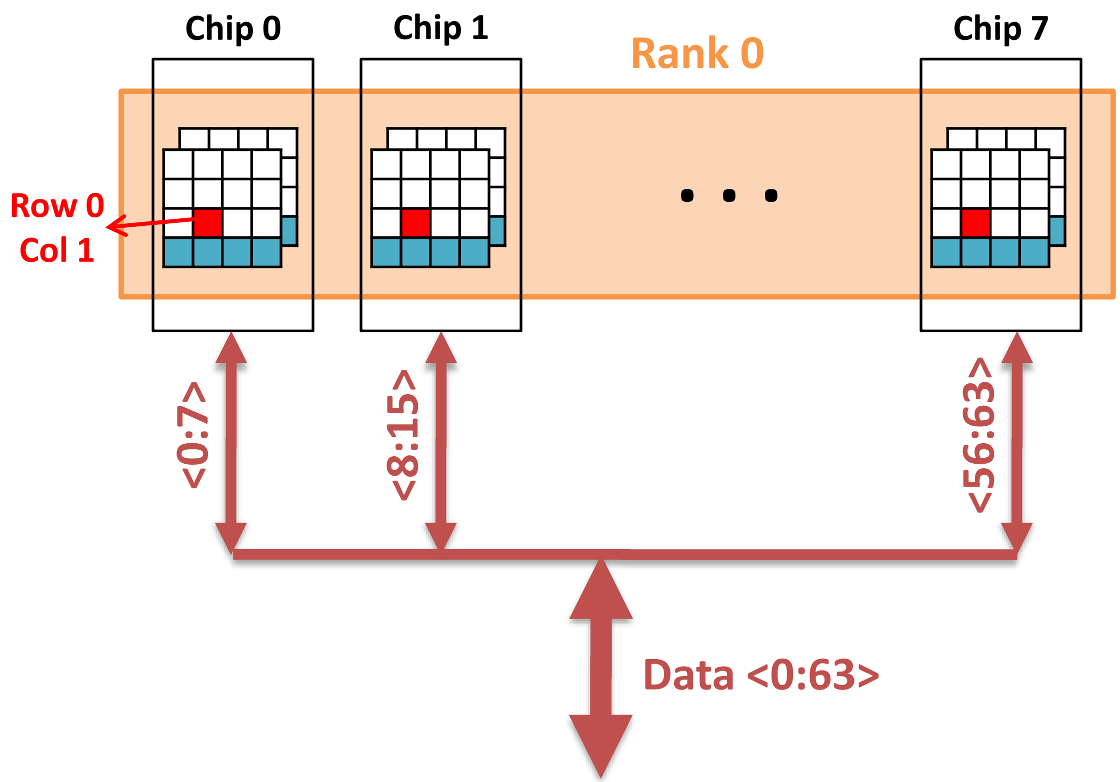


Example: Transferring a cache block



Example: Transferring a cache block





A 64B cache block takes 8 I/O cycles to transfer.

During the process, 8 columns are read sequentially.

Latency Components: Basic DRAM Operation

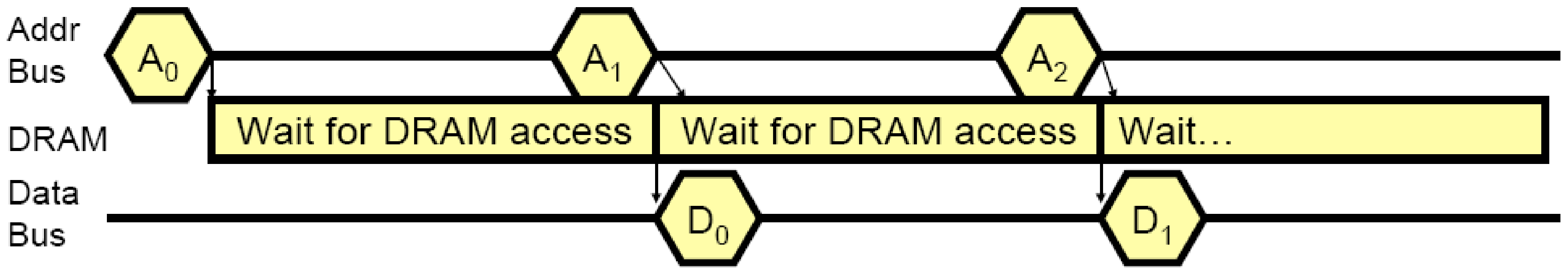
- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller → DRAM transfer time
- DRAM bank latency
  - Simple CAS (column address strobe) if row is "open" OR
  - RAS (row address strobe) + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)
- DRAM → Controller transfer time
  - Bus latency (BL)
- Controller to CPU transfer time

Multiple Banks (Interleaving) and Channels

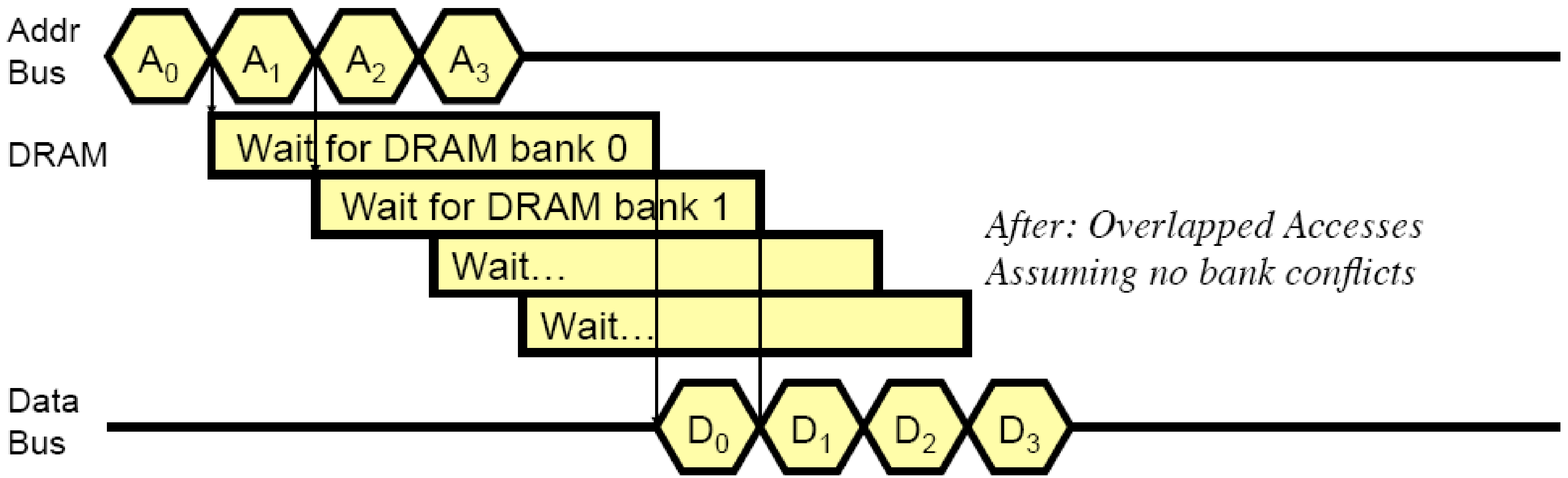
- Multiple banks
  - Enable concurrent DRAM accesses
  - Bits in address determine which bank an address resides in
- Multiple independent channels serve the same purpose
  - But they are even better because they have separate data buses
  - Increased bus bandwidth

- Enabling more concurrency requires reducing
  - Bank conflicts
  - Channel conflicts
- How to select/randomize bank/channel indices in address?
  - Lower order bits have more entropy
  - Randomizing hash functions (XOR of different address bits)
  - Pathological cases (strided at long length)

How Multiple Banks Help



Before: No Overlapping
Assuming accesses to different DRAM rows



After: Overlapped Accesses
Assuming no bank conflicts

Address Mapping (Single Channel)

- Single-channel system with 8-byte memory bus
  - 2GB memory, 8 banks, 16K rows & 2K columns per bank

- Row interleaving
  - Consecutive rows of memory in consecutive banks

Row (14 bits)	Bank (3 bits)	Column (11 bits)	Byte in bus (3 bits)

  - Accesses to consecutive cache blocks serviced in a pipelined manner

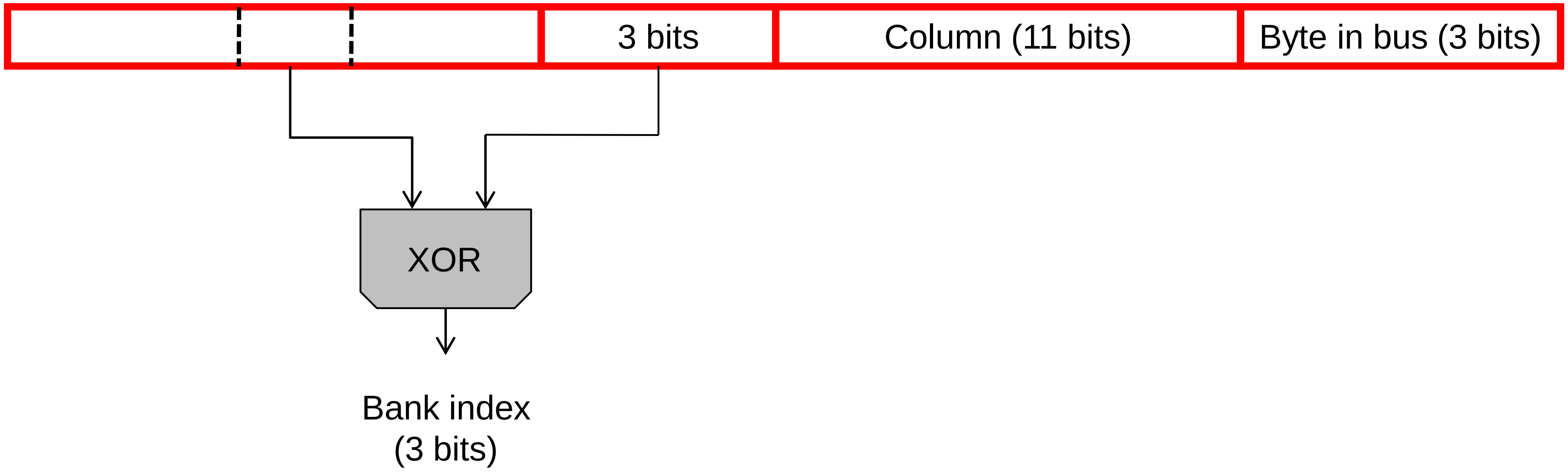
- Cache block interleaving
  - Consecutive cache block addresses in consecutive banks
  - 64 byte cache blocks

Row (14 bits)	High Column	Bank (3 bits)	Low Col.	Byte in bus (3 bits)
	8 bits		3 bits	

  - Accesses to consecutive cache blocks can be serviced in parallel

Bank Mapping Randomization

- DRAM controller can randomize the address mapping to banks so that bank conflicts are less likely

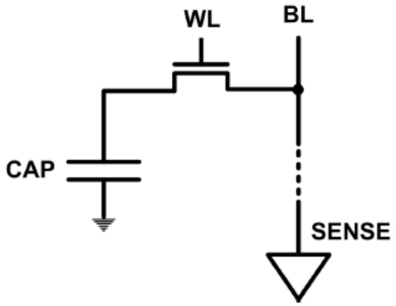


DRAM Refresh (I)

- DRAM capacitor charge leaks over time
- The memory controller needs to read each row periodically to restore the charge
  - Activate + precharge each row every N ms
  - Typical N = 64 ms
- Implications on performance?

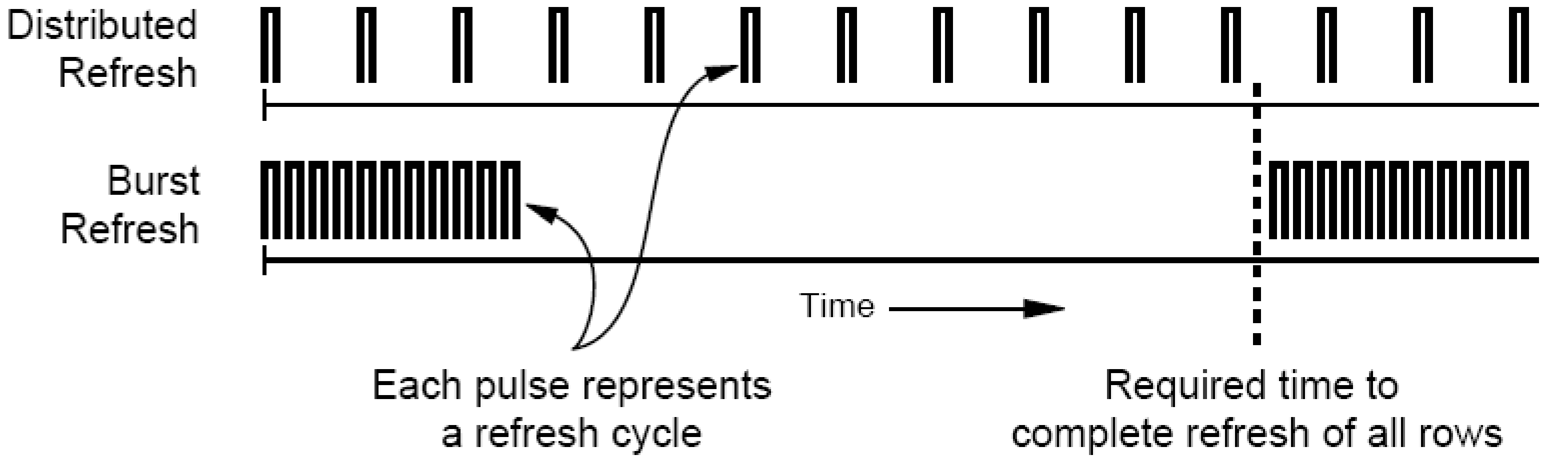
  -- DRAM bank unavailable while refreshed

  -- Long pause times: If we refresh all rows in burst, every 64ms the DRAM will be unavailable until refresh ends



- Burst refresh: All rows refreshed immediately after one another
- Distributed refresh: Each row refreshed at a different time, at regular intervals

DRAM Refresh (II)



- Distributed refresh eliminates long pause times
- How else we can reduce the effect of refresh on performance?
  - Can we reduce the number of refreshes?

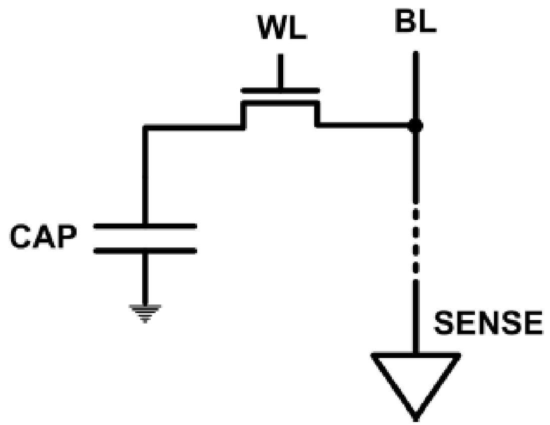
Downsides of DRAM Refresh

-- Energy consumption: Each refresh consumes energy

-- Performance degradation: DRAM rank/bank unavailable while refreshed

-- QoS/predictability impact: (Long) pause times during refresh

-- Refresh rate limits DRAM density scaling



Liu et al., "RAIDR: Retention-aware Intelligent DRAM Refresh," ISCA 2012.

Memory Controllers

Onur Mutlu….

Slide History/Attribution Diagram:

DRAM versus Other Types of Memories

- Long latency memories have similar characteristics that need to be controlled.

- The following discussion will use DRAM as an example, but many issues are similar in the design of controllers for other types of memories
  - Flash memory
  - Other emerging memory technologies
    - Phase Change Memory
    - Spin-Transfer Torque Magnetic Memory

DRAM Types

- DRAM has different types with different interfaces optimized for different purposes
  - Commodity: DDR, DDR2, DDR3, DDR4
  - Low power (for mobile): LPDDR[1-5]
  - High bandwidth (for graphics): GDDR[1-5]
  - Low latency: eDRAM, RLDRAM, …
  - 3D stacked: HBM, HMC,…
- Underlying microarchitecture is fundamentally the same
- A flexible memory controller can support various DRAM types, but…
- This complicates the memory controller
  - Difficult to support all types (and upgrades)

DRAM Controller: Functions

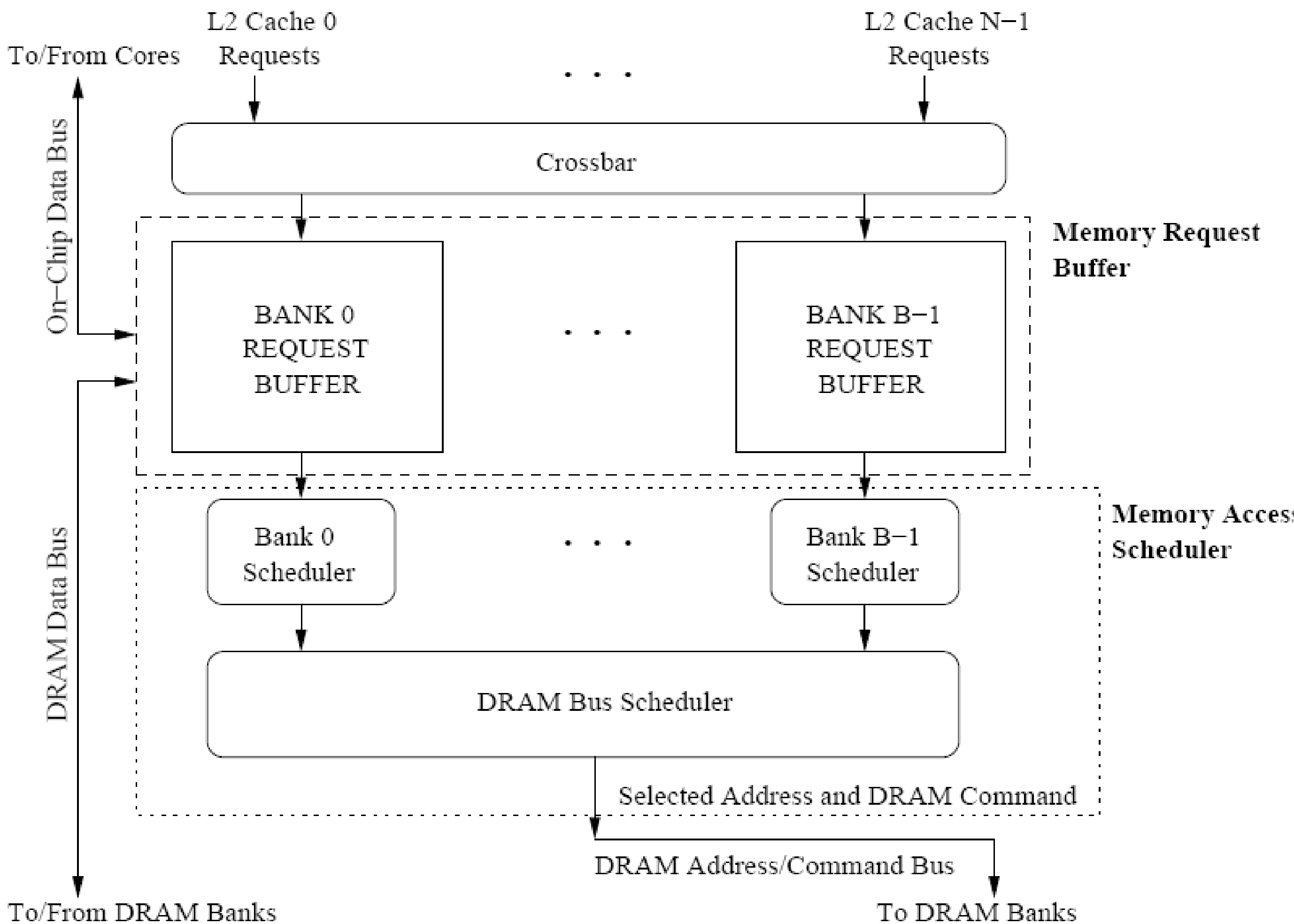
- Ensure correct operation of DRAM (refresh and timing)
- Service DRAM requests while obeying timing constraints of DRAM chips
  - Constraints: resource conflicts (bank, bus, channel), minimum write-to-read delays
  - Translate requests to DRAM command sequences
- Buffer and schedule requests to improve performance
  - Reordering, row-buffer, bank, rank, bus management
- Manage power consumption and thermals in DRAM
  - Turn on/off DRAM chips, manage power modes

DRAM Controller: Where to Place

- In chipset
  + More flexibility to plug different DRAM types into the system
  + Less power density in the CPU chip

- On CPU chip
  + Reduced latency for main memory access
  + Higher bandwidth between cores and controller
    - More information can be communicated (e.g. request’s importance in the processing core)

A Modern DRAM Controller



DRAM Scheduling Policies (I)

- FCFS (first come first served)
  - Oldest request first

- FR-FCFS (first ready, first come first served)
  1. Row-hit first
  2. Oldest first

  Goal: Maximize row buffer hit rate → maximize DRAM throughput

  - Actually, scheduling is done at the command level
    - Column commands (read/write) prioritized over row commands (activate/precharge)
    - Within each group, older commands prioritized over younger ones

DRAM Scheduling Policies (II)

- A scheduling policy is essentially a prioritization order

- Prioritization can be based on
  - Request age
  - Row buffer hit/miss status
  - Request type (prefetch, read, write)
  - Requestor type (load miss or store miss)
  - Request criticality
    - Oldest miss in the core?
    - How many instructions in core are dependent on it?

Row Buffer Management Policies

- Open row
  - Keep the row open after an access

  \+ Next access might need the same row → row hit

  -- Next access might need a different row → row conflict, wasted energy

- Closed row
  - Close the row after an access (if no other requests already in the request buffer need the same row)

  \+ Next access might need a different row → avoid a row conflict

  -- Next access might need the same row → extra activate latency

- Adaptive policies
  - Predict whether or not the next access to the bank will be to the same row

Open vs. Closed Row Policies

Policy	First access	Next access	Commands needed for next access
Open row	Row 0	Row 0 (row hit)	Read
Open row	Row 0	Row 1 (row conflict)	Precharge + Activate Row 1 + Read
Closed row	Row 0	Row 0 – access in request buffer (row hit)	Read
Closed row	Row 0	Row 0 – access not in request buffer (row closed)	Activate Row 0 + Read + Precharge
Closed row	Row 0	Row 1 (row closed)	Activate Row 1 + Read + Precharge

Why are DRAM Controllers Difficult to Design?

- Need to obey DRAM timing constraints for correctness
  - There are many (50+) timing constraints in DRAM
  - tWTR: Minimum number of cycles to wait before issuing a read command after a write command is issued (rank level constraint – change sense of bus)
  - tRC: Minimum number of cycles between the issuing of two consecutive activate commands to the same bank
  - …
- Need to keep track of many resources to prevent conflicts
  - Channels, banks, ranks, data bus, address bus, row buffers
- Need to handle DRAM refresh
- Need to optimize for performance & QoS (in the presence of constraints)
  - Reordering is not simple
  - Fairness and QoS needs complicate the scheduling problem

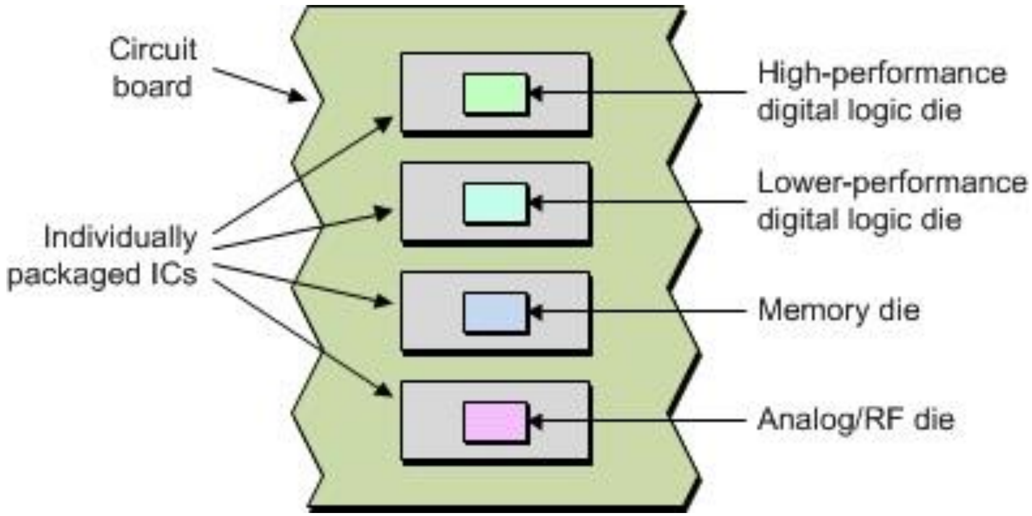
Many DRAM Timing Constraints

Latency	Symbol	DRAM cycles	Latency	Symbol	DRAM cycles
Precharge	${}^tRP$	11	Activate to read/write	${}^tRCD$	11
Read column address strobe	CL	11	Write column address strobe	CWL	8
Additive	AL	0	Activate to activate	${}^tRC$	39
Activate to precharge	${}^tRAS$	28	Read to precharge	${}^tRTP$	6
Burst length	${}^tBL$	4	Column address strobe to column address strobe	${}^tCCD$	4
Activate to activate (different bank)	${}^tRRD$	6	Four activate windows	${}^tFAW$	24
Write to read	${}^tWTR$	6	Write recovery	${}^tWR$	12

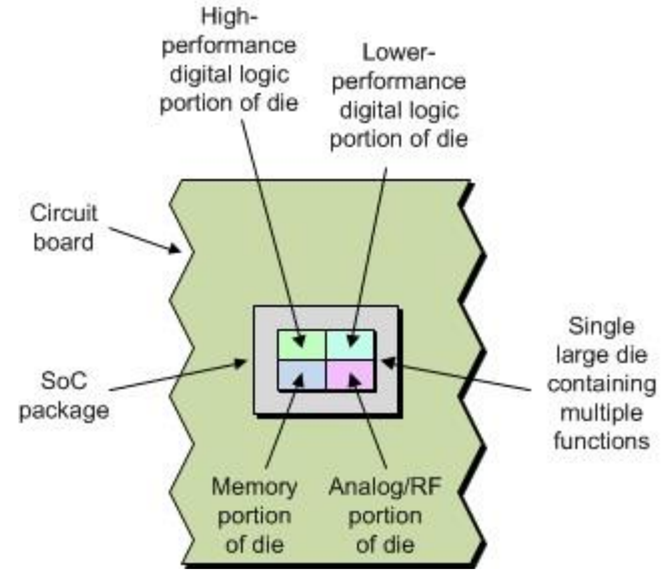
Table 4. DDR3 1600 DRAM timing specifications

- From Lee et al., "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems," HPS Technical Report, April 2010.

2D Packaging



Board level

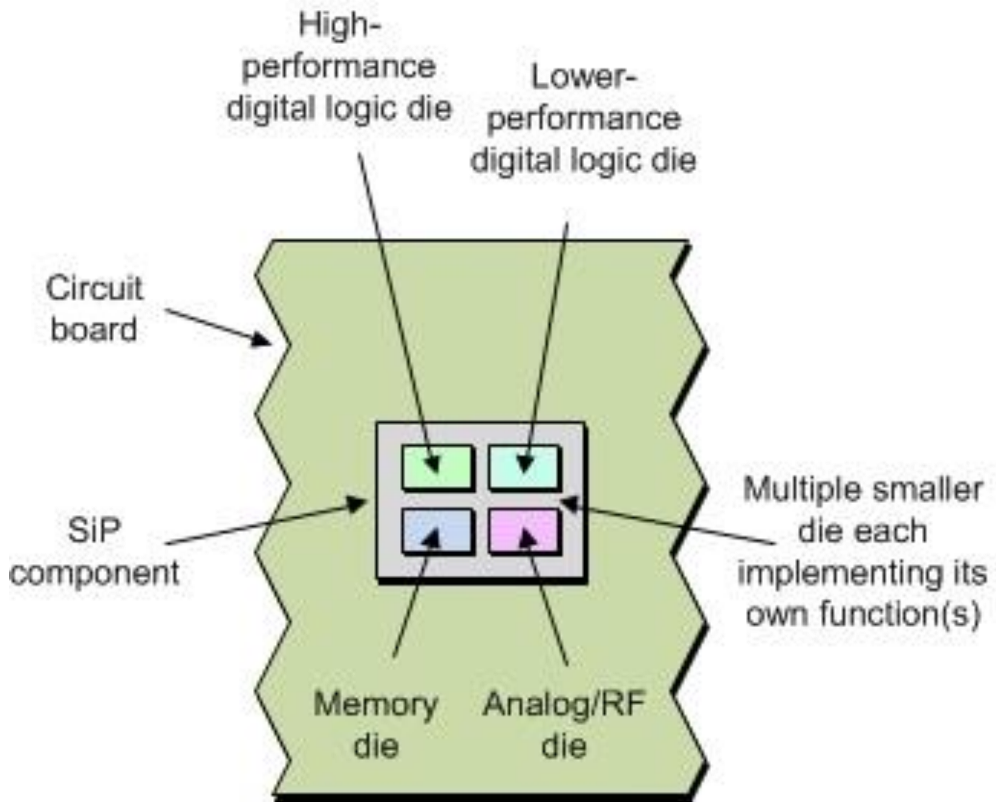


System on Chip

[M. Maxfield, "2D vs. 2.5D vs. 3D ICs 101," EE Times, April 2012]

Conventional packaging approaches

2D Packaging

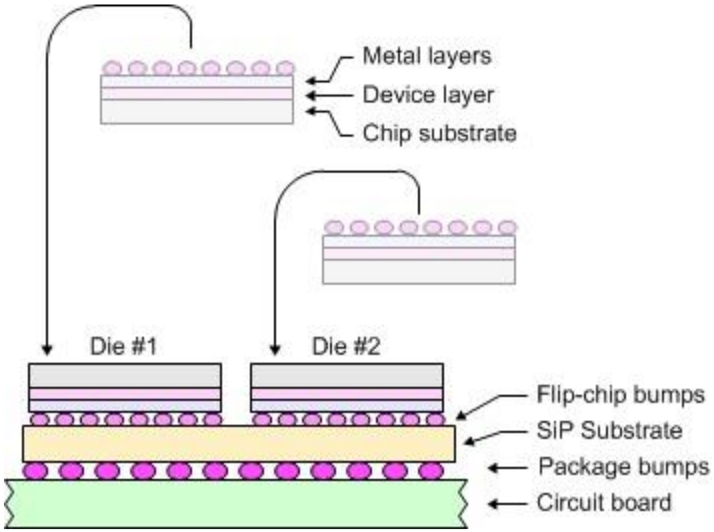


[M. Maxfield, "2D vs. 2.5D vs. 3D ICs 101," EE Times, April 2012]

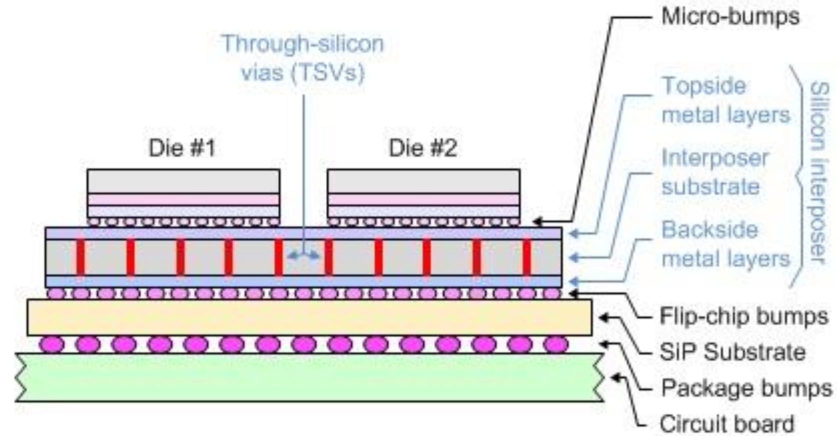
Move toward System in Package (SIP)

- PCB, ceramic, semiconductor substrates

2.5D Packaging



2D Packaging

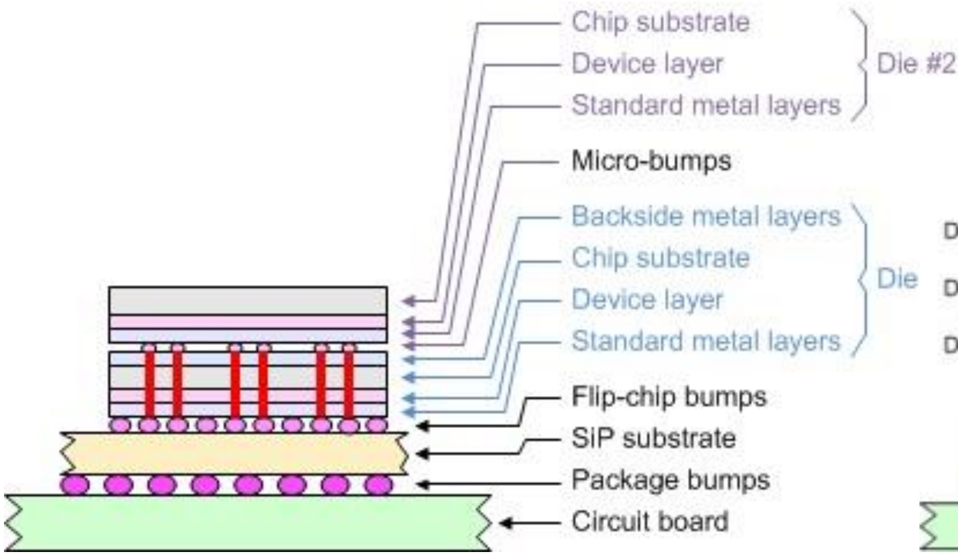


2.5D Packaging

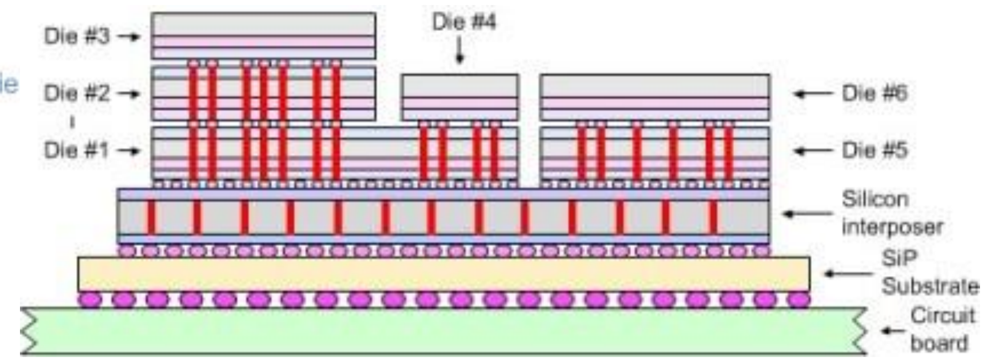
[M. Maxfield, "2D vs. 2.5D vs. 3D ICs 101," EE Times, April 2012]

2.5D uses silicon interposer, through-silicon vias (TSV)

3D Packaging



3D Homogeneous



3D Heterogeneous

[M. Maxfield, “2D vs. 2.5D vs. 3D ICs 101,” EE Times, April 2012]

3D uses through-silicon vias (TSV) and/or interposer

Packaging Discussion

- Heterogeneous integration
  - RF, analog (PHY), FG/PCM/ReRAM, photonics
- Cost
- Silicon yield
- Bandwidth, esp. interposer
- Thermals
- It’s real!
  - DRAM: HMC, HBM
  - FPGAs
  - GPUs: AMD, NVIDIA
  - CPUs: AMD Zen, EPYC

Brief History of DRAM

- DRAM (memory): a major force behind computer industry
  - Modern DRAM came with introduction of IC (1970)
  - Preceded by magnetic "core" memory (1950s)
    - Each cell was a small magnetic "donut"
  - And by mercury delay lines before that (ENIAC)
    - Re-circulating vibrations in mercury tubes

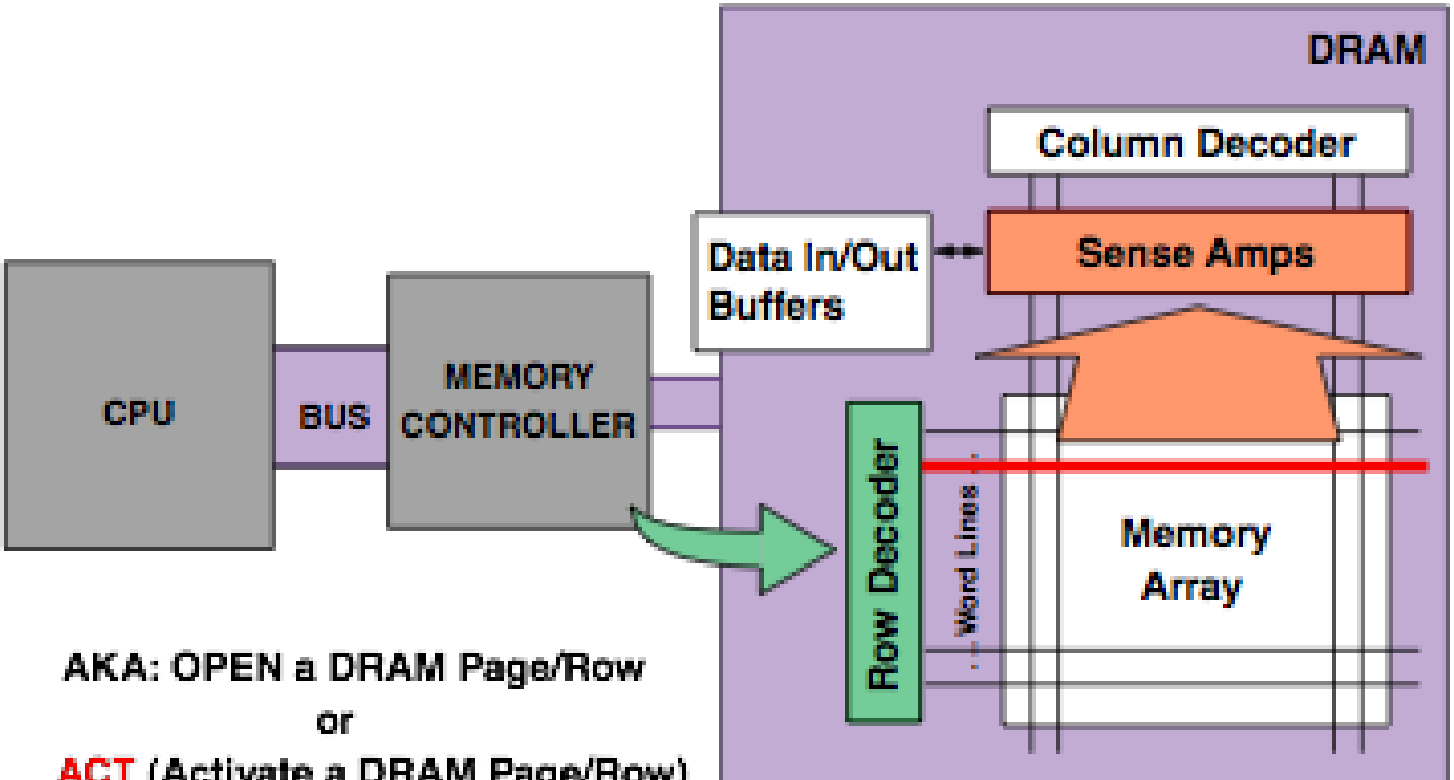
"the one single development that put computers on their feet was the invention of a reliable form of memory, namely the core memory… It's cost was reasonable, it was reliable, and because it was reliable it could in due course be made large"

Maurice Wilkes

Memoirs of a Computer Programmer, 1985

DRAM Basics [Jacob and Wang]

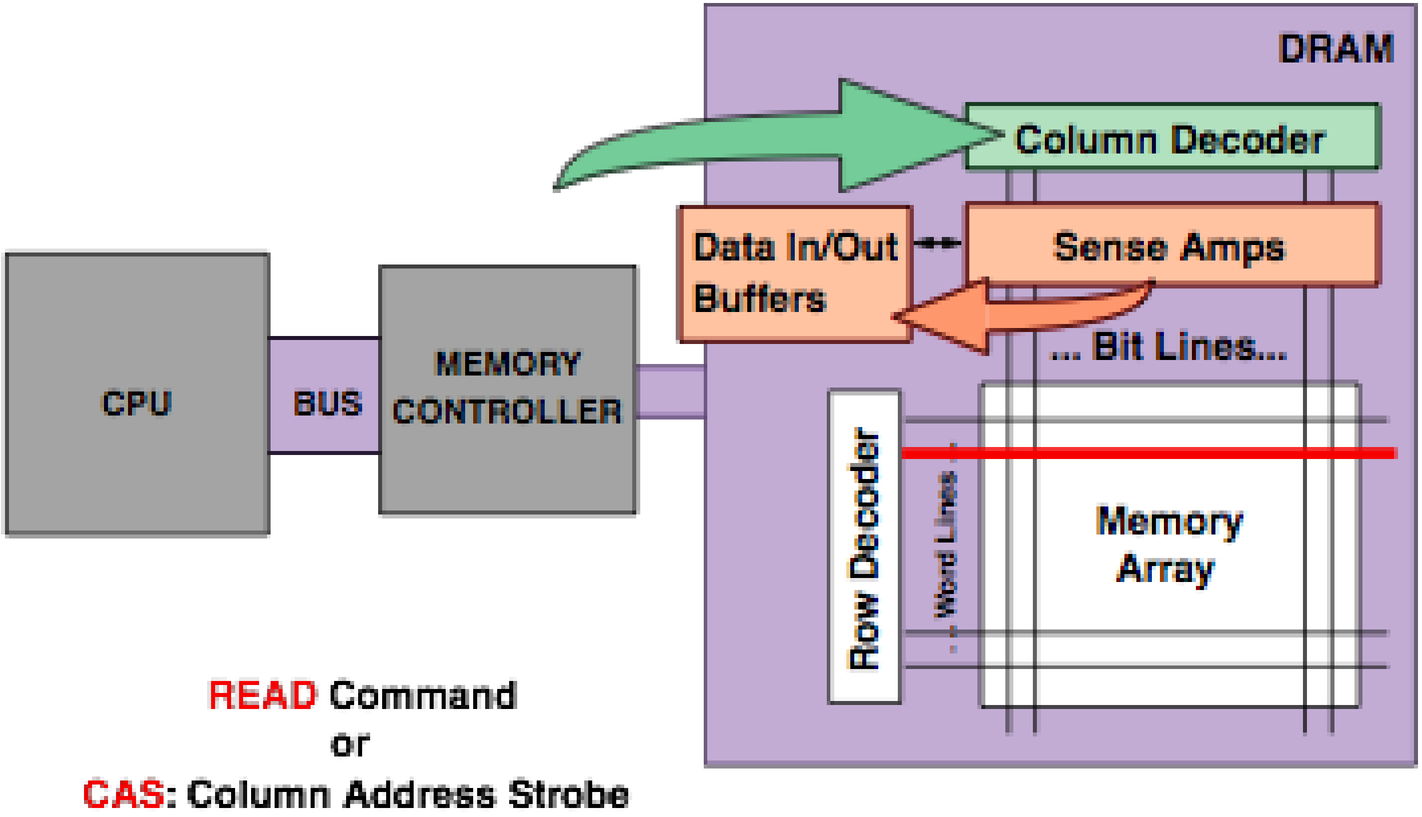
- Precharge and Row Access



AKA: OPEN a DRAM Page/Row
or
ACT (Activate a DRAM Page/Row)
or
RAS (Row Address Strobe)

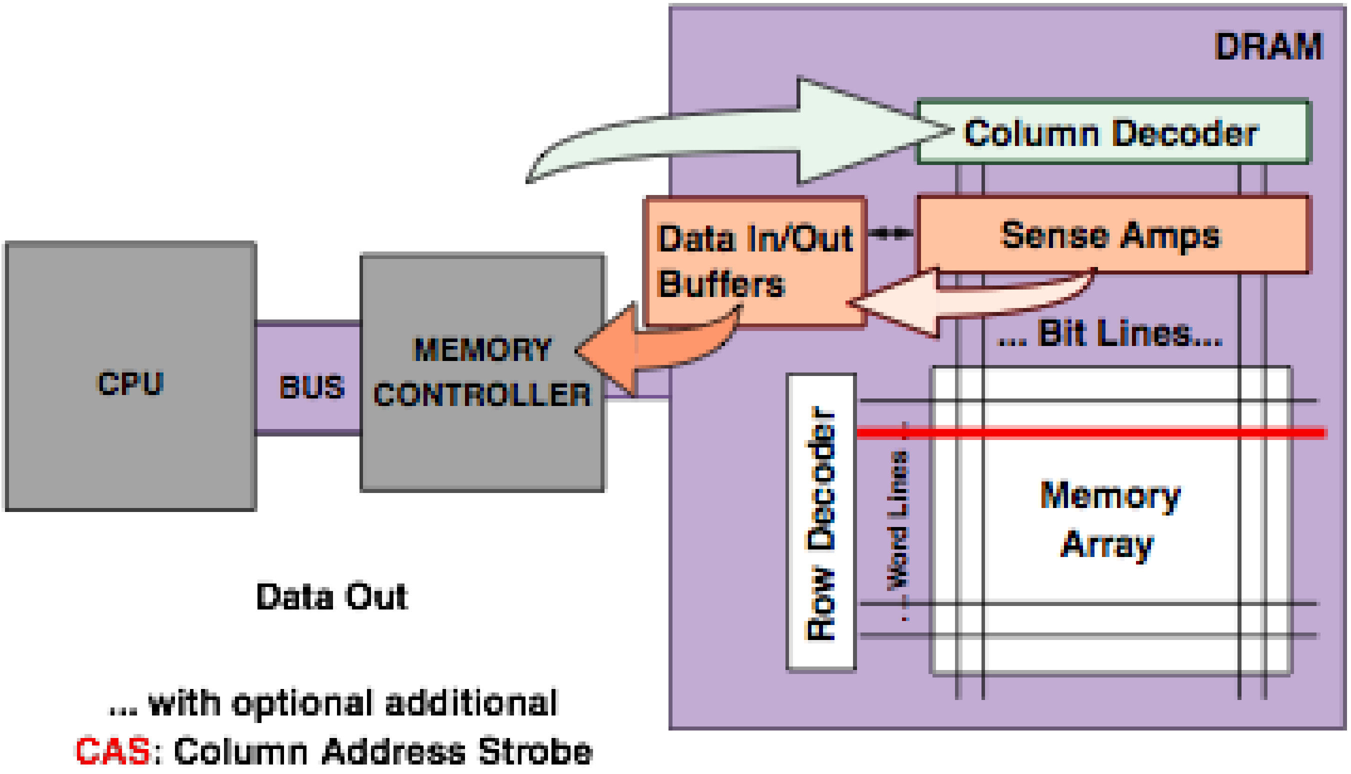
DRAM Basics, cont.

- Column Access



DRAM Basics, cont.

- Data Transfer



Open v. Closed Pages

- Open Page
  - Row stays active until another row needs to be accessed
  - Acts as memory-level cache to reduce latency
  - Variable access latency complicates memory controller
  - Higher power dissipation (sense amps remain active)
- Closed Page
  - Immediately deactivate row after access
  - All accesses become Activate Row, Read/Write, Precharge

- Complex power v. performance trade off

DRAM Bandwidth

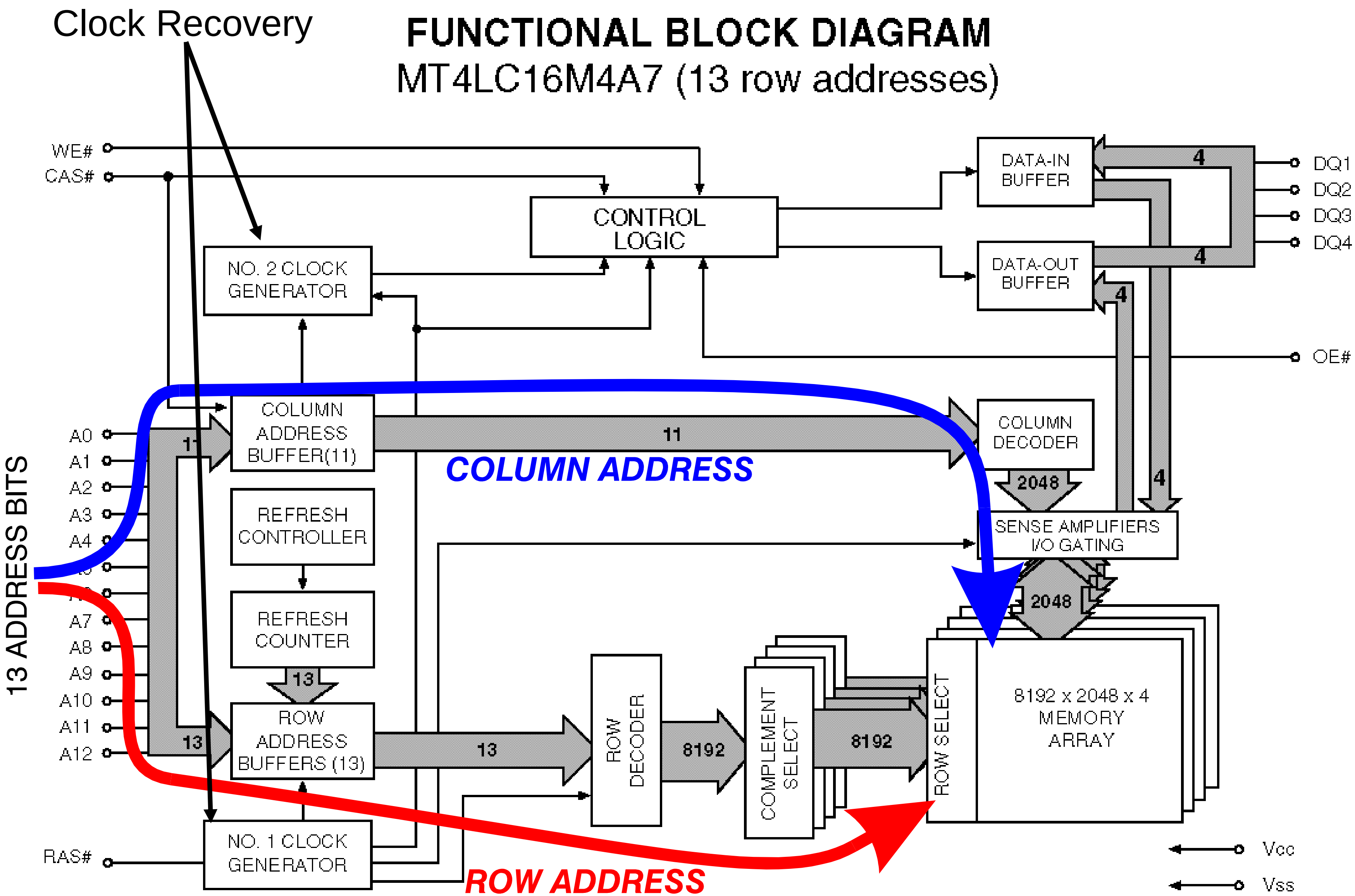
- Use multiple DRAM chips to increase bandwidth
  - Recall, access are the same size as second-level cache
  - Example, 16 2-byte wide chips for 32B access
- DRAM density increasing faster than demand
  - Result: number of memory chips per system decreasing
- Need to increase the **bandwidth per chip**
  - Especially important in game consoles
  - SDRAM ➔ DDR ➔ DDR2 ➔ FBDIMM (➔ DDR3)
  - Rambus - high-bandwidth memory
    - Used by several game consoles

DRAM Evolution

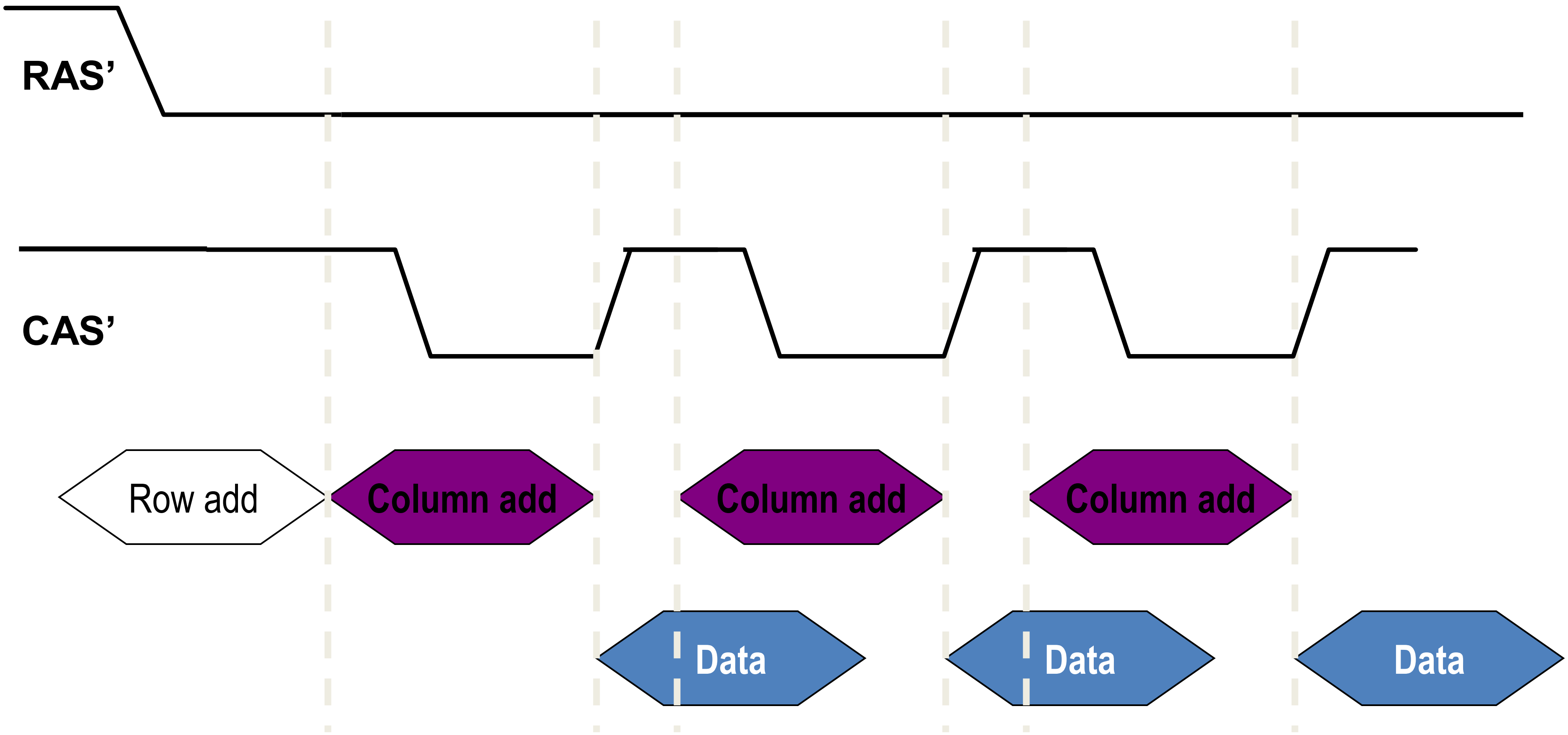
- Survey by Cuppu et al.

1. Early Asynchronous Interface
2. Fast Page Mode/Nibble Mode/Static Column (skip)
3. Extended Data Out
4. Synchronous DRAM & Double Data Rate
5. Rambus & Direct Rambus
6. FB-DIMM

Old 64MbitDRAM Example from Micron

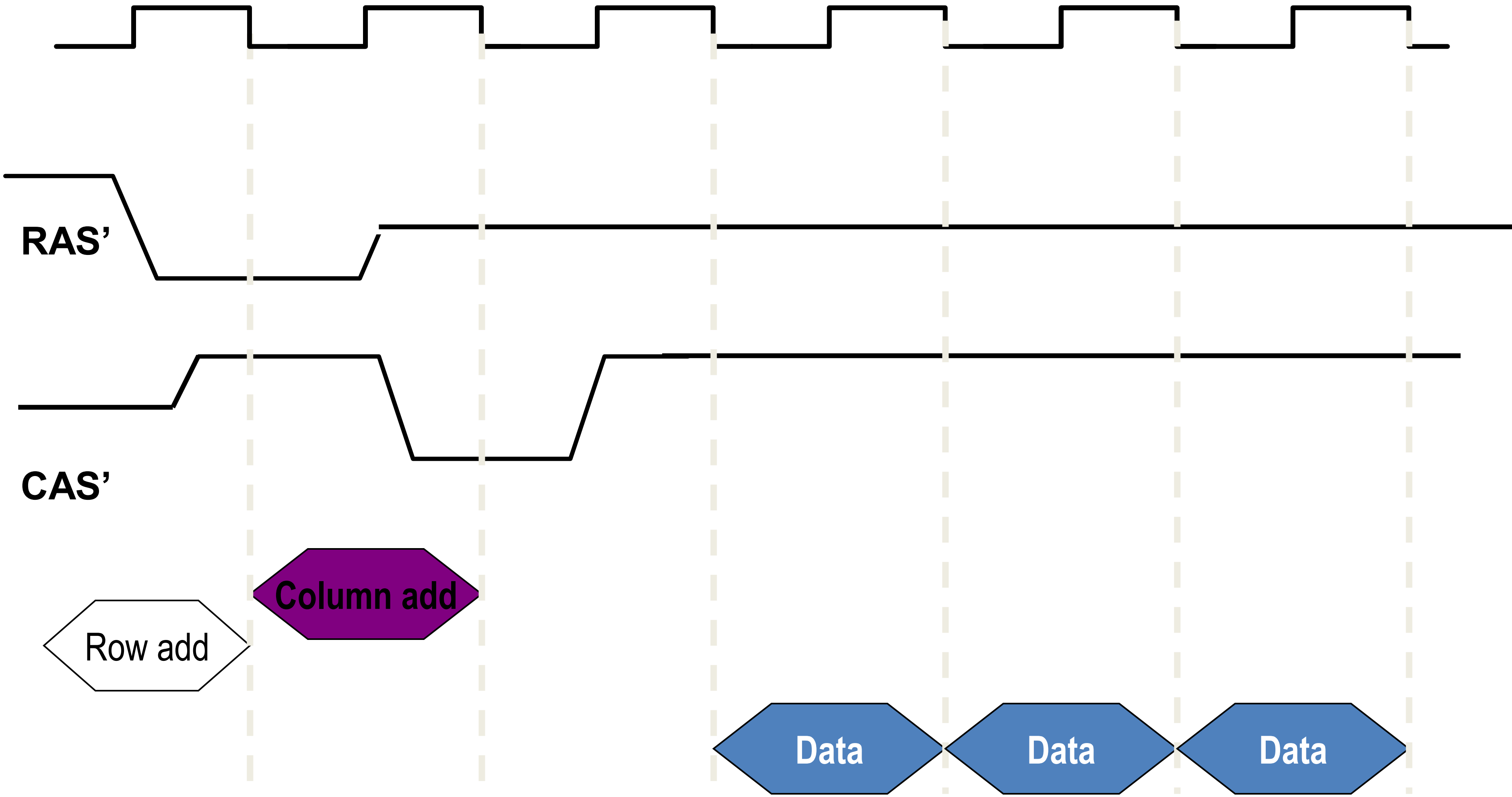


Extended Data Out (EDO)



- Similar to Fast Page Mode
- But overlapped Column Address assert with Data Out

Synchronous DRAM (SDRAM)



- Add Clock and Wider data!
- Also multiple transfers per RAS/CAS

Enhanced SDRAM & DDR

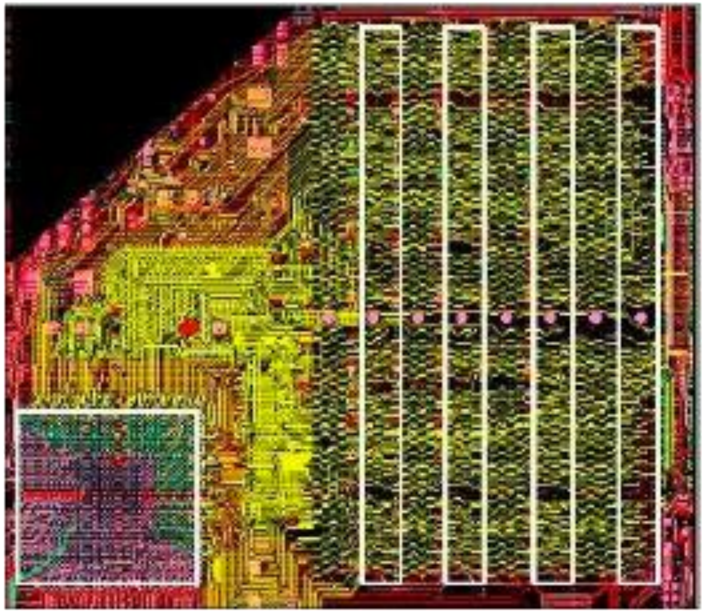
- Evolutionary Enhancements on SDRAM:

1. ESDRAM (Enhanced): Overlap row buffer access with refresh

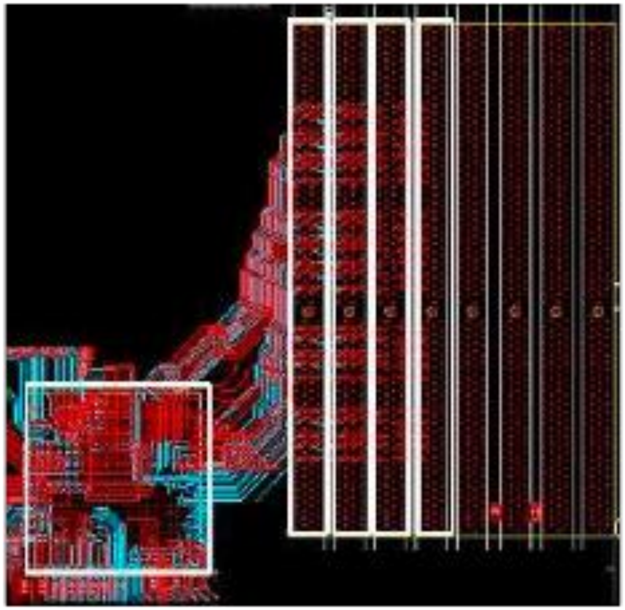
2. DDR (Double Data Rate): Transfer on both clock edges
3. DDR2's small improvements
   lower voltage, on-chip termination, driver calibration
   prefetching, conflict buffering
4. DDR3, more small improvements
   lower voltage, 2X speed, 2X prefetching,
   2X banks, "fly-by topology", automatic calibration

Wide v. Narrow Interfaces

- High frequency ➔ short wavelength ➔ data skew issues
  - Balance wire lengths



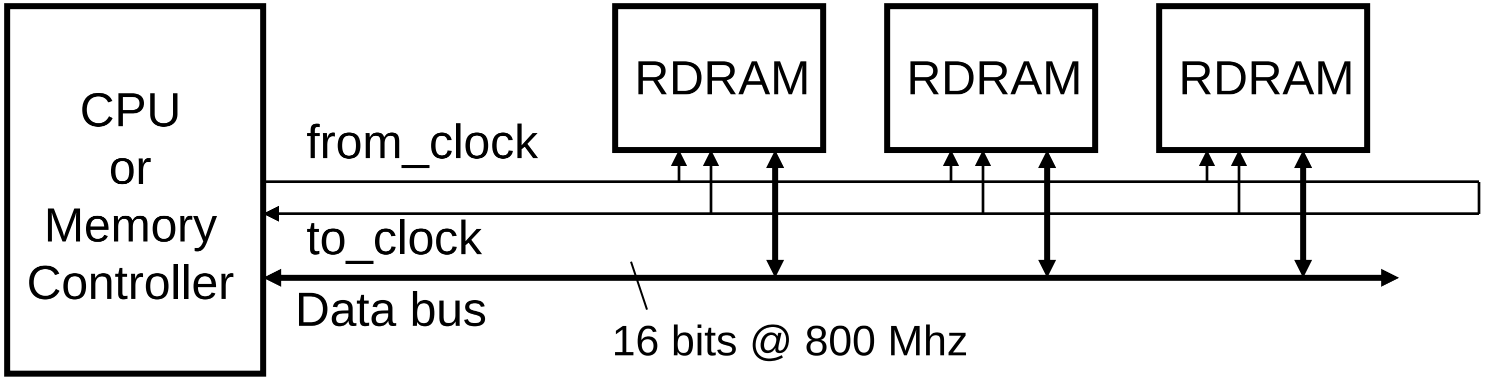
DDR-2 serpentine board routing



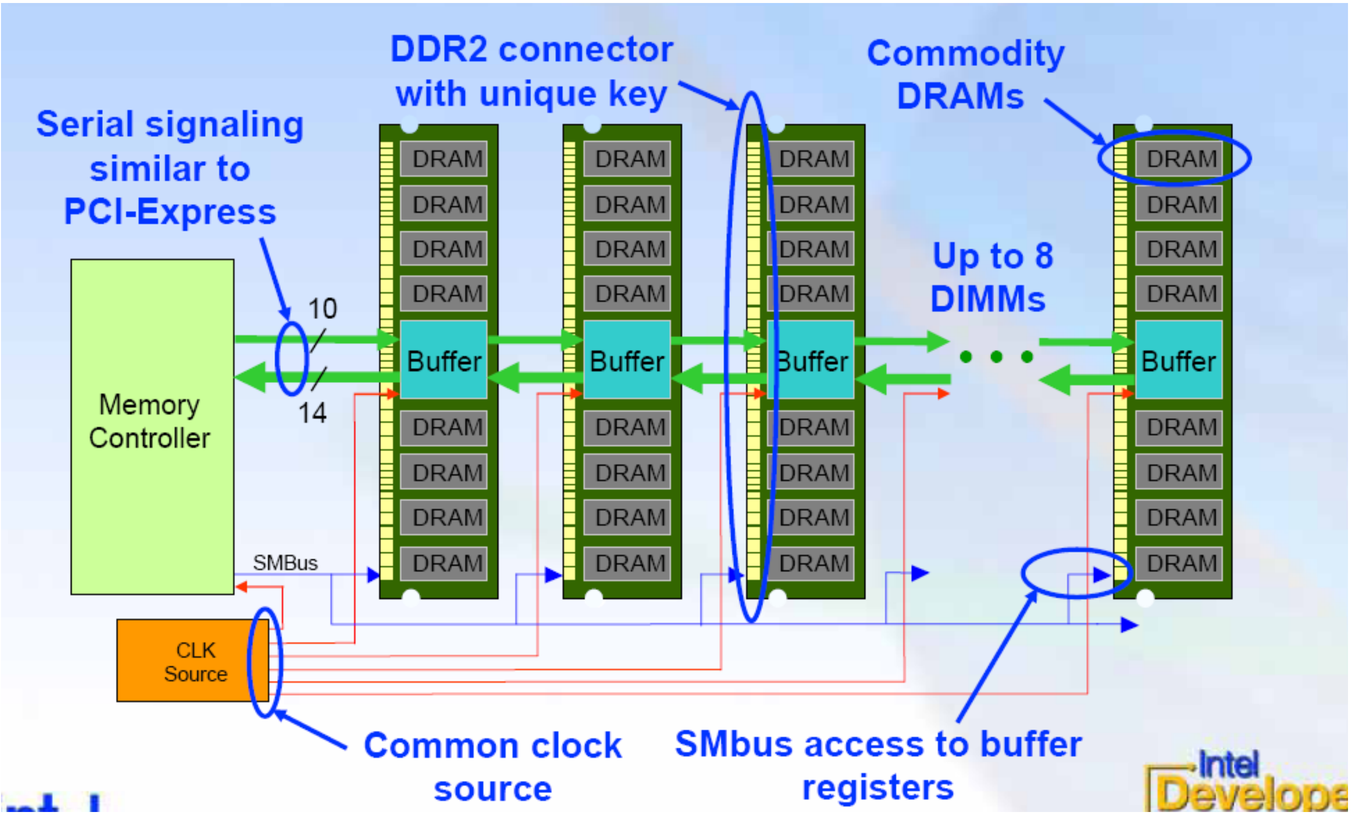
FB-DIMM board routing

Rambus RDRAM

- High-frequency, narrow channel
  - Time multiplexed “bus” ➔ dynamic point-to-point channels
  - ~40 pins ➔ 1.6GB/s
- Proprietary solution
  - Never gained industry-wide acceptance (cost and power)
  - Used in some game consoles (e.g., PS2)



FB-DIMM

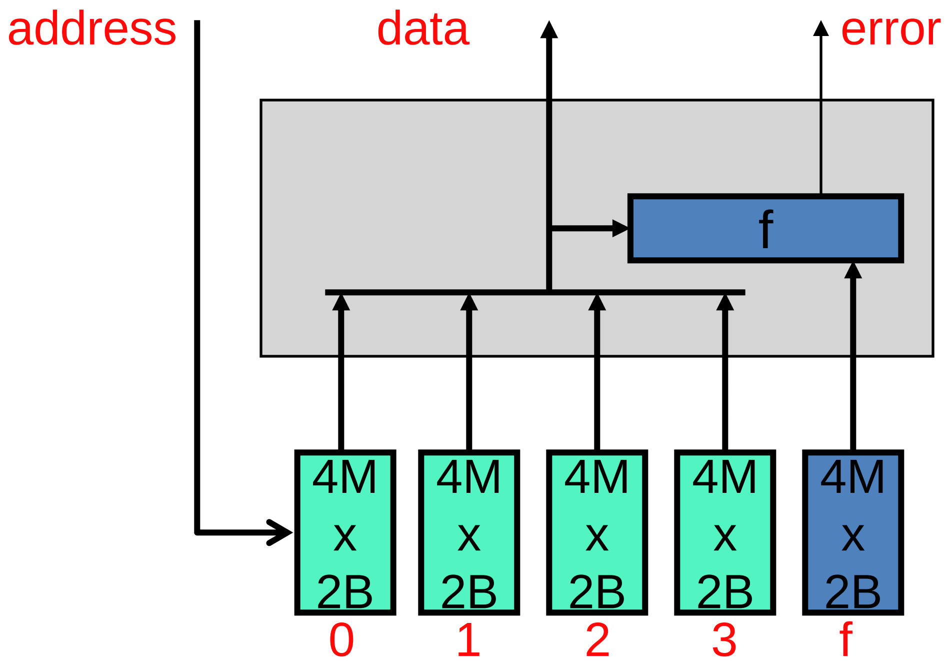


DDR2 connector with unique key
Commodity DRAMs
Serial signaling similar to PCI-Express
Up to 8 DIMMs
Memory Controller
10
14
DRAM
Buffer
SMBus
CLK Source
Common clock source
SMbus access to buffer registers
Intel Developer

DRAM Reliability

- One last thing about DRAM technology… **errors**
  - DRAM bits can flip from 0➔1 or 1➔0
    - Small charge stored per bit
    - Energetic α-particle strikes disrupt stored charge
    - Many more bits
  - Modern DRAM systems: built-in error detection/correction
    - Today all servers; most new desktop and laptops
- **Key idea: checksum-style redundancy**
  - Main DRAM chips store data, additional chips store f(data)
    - |f(data)| < |data|
  - On read: re-compute f(data), compare with stored f(data)
    - Different ? Error…
  - Option I (**detect**): kill program
  - Option II (**correct**): enough information to fix error? fix and go on

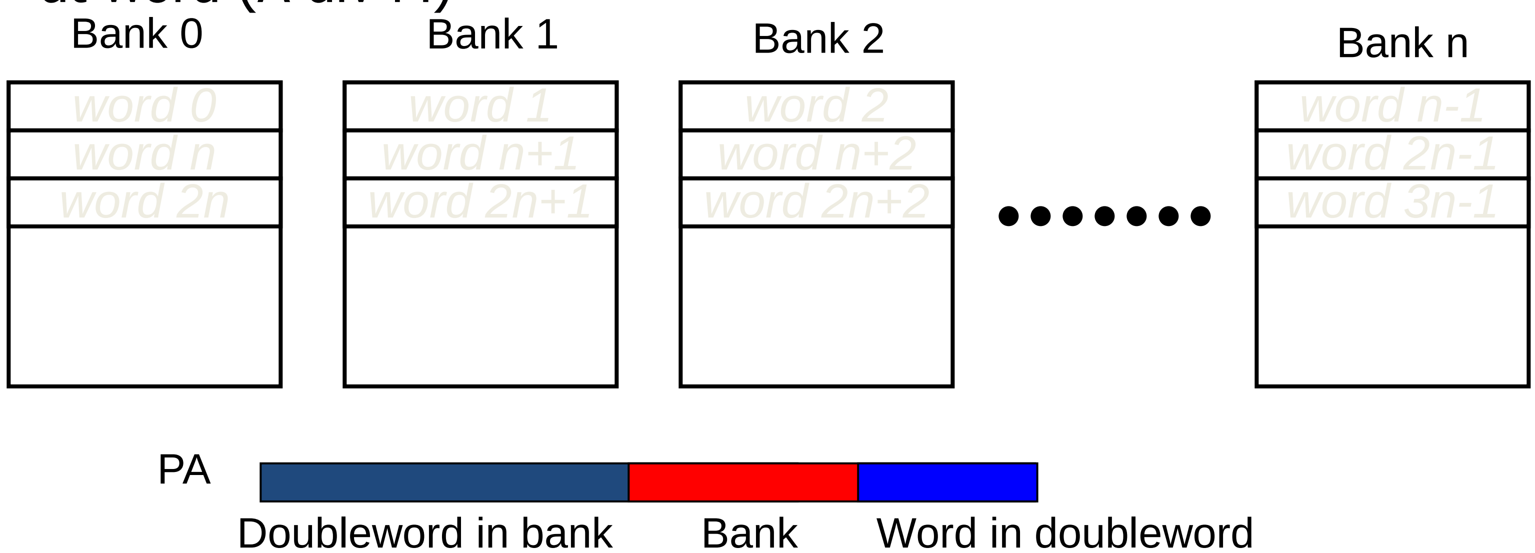
DRAM Error Detection and Correction



- Performed by memory controller (not the DRAM chip)
- Error detection/correction schemes distinguished by…
    - How many (simultaneous) errors they can detect
    - How many (simultaneous) errors they can correct

Interleaved Main Memory

- Divide memory into M banks and "interleave" addresses across them, so word A is
  - in bank (A mod M)
  - at word (A div M)

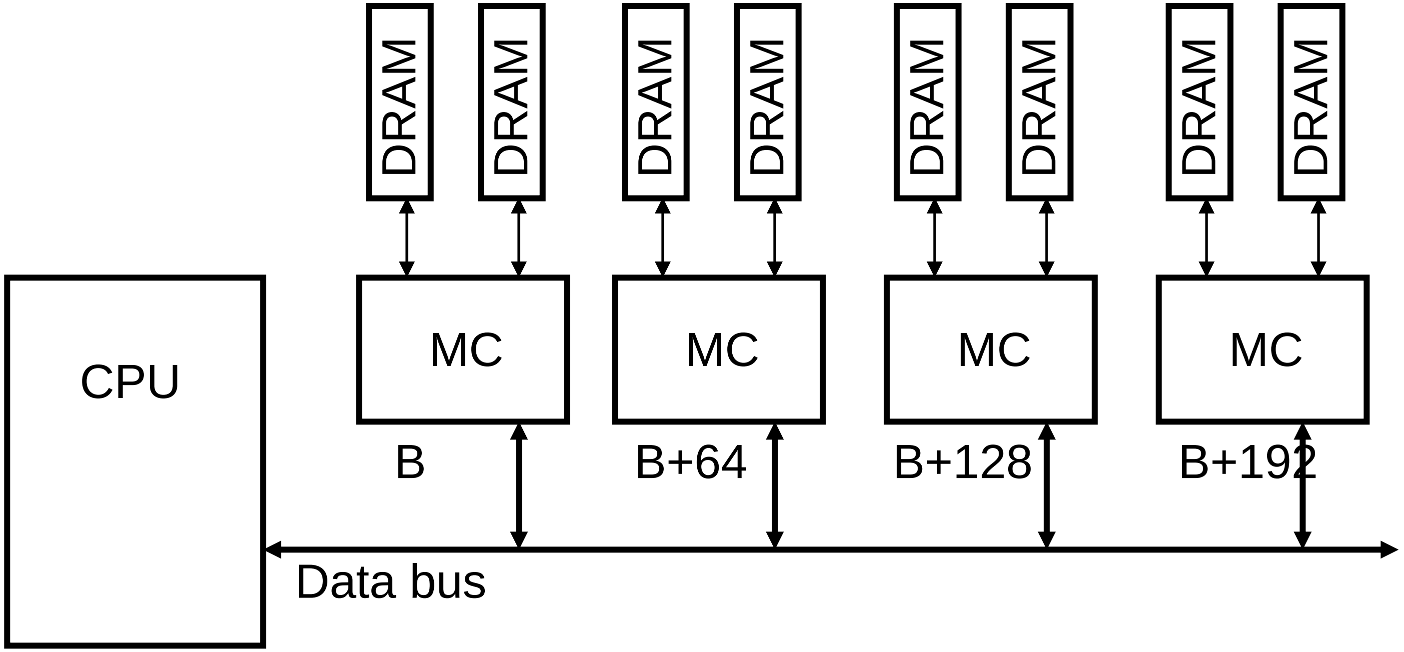


Interleaved memory increases memory BW without wider bus

- *Use parallelism in memory banks to hide memory latency*

Block interleaved memory systems

- Cache blocks map to separate memory controllers
  - Interleave across DRAMs w/i a MC
  - Interleave across intra-DRAM banks w/i a DRAM



Memory Hierarchy Review

- Storage: registers, **memory**, disk
  - Memory is the fundamental element
- Memory component performance
  - $\mathbf{t_{avg} = t_{hit} + \%_{miss} * t_{miss}}$
  - Can't get both low t_{hit} and $\%_{miss}$ in a single structure
- Memory hierarchy
  - Upper components: small, fast, expensive
  - Lower components: big, slow, cheap
  - t_{avg} of hierarchy is close to t_{hit} of upper (fastest) component
    - 10/90 rule: 90% of stuff found in fastest component
  - **Temporal/spatial locality**: automatic up-down data movement

Software Managed Memory

- Isn't full associativity difficult to implement?
  - Yes … in hardware
  - Implement fully associative memory in software

- Let's take a step back…