13
Main Memory Architecture

18-548/15-548 Memory System Architecture
Philip Koopman
October 19, 1998

Required Reading: **Cragon 5.1 - 5.1.5**

Supplemental Reading: Hennessy & Patterson 5.6
IBM App. Note: Understanding DRAM

Carnegie Mellon

Assignments

- **By next class read about Main Memory Performance:**
  - Cragon 5.1.6 - 5.1.7
  - Fast DRAM article from EDN (Feb. 1997)
  - Supplemental Reading:
    - Siewiorek & Koopman 5.2.2
- **Homework 7 due October 21**
- **Lab 4 due October 23**
- **Test #2 Wednesday October 28**
  - Emphasizes material since Test #1
  - In-class review Monday October 26
  - Closed book, closed notes; bring erasers, sharpened pencils, calculator

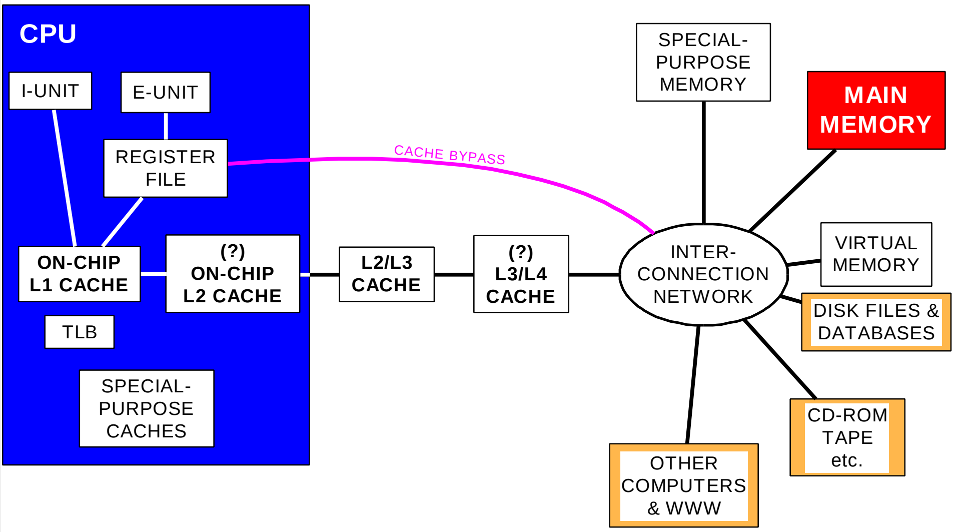
Where Are We Now?

- **Where we've been:**
  - Cache memory
  - Tuning for performance
- **Where we're going for two classes:**
  - Main memory architecture & performance
- **Where we're going next:**
  - Vector computing
  - Buses

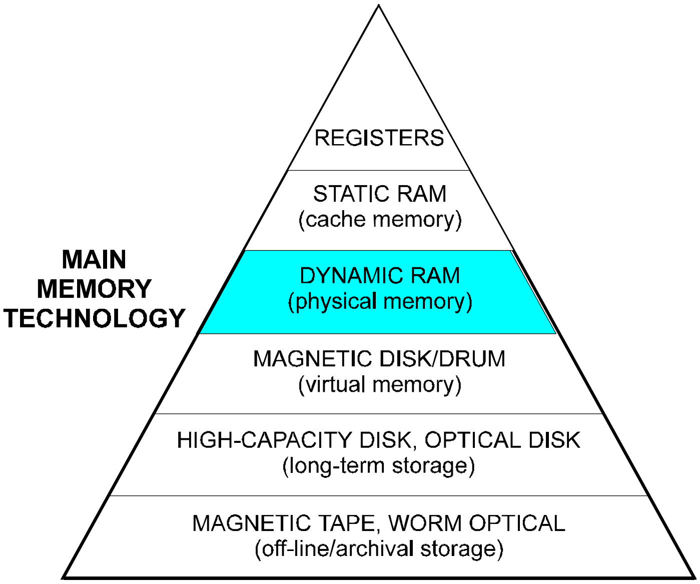
Preview

- **DRAM chip operation**
  - Constraints on minimum memory size vs. performance improvement techniques
- **Increasing memory bandwidth**
  - Burst transfers
  - Interleaved access
  - Some of these techniques help with latency as well

Main Memory



Usually Implemented in DRAM


MAIN MEMORY TECHNOLOGY
REGISTERS
STATIC RAM (cache memory)
DYNAMIC RAM (physical memory)
MAGNETIC DISK/DRUM (virtual memory)
HIGH-CAPACITY DISK, OPTICAL DISK (long-term storage)
MAGNETIC TAPE, WORM OPTICAL (off-line/archival storage)

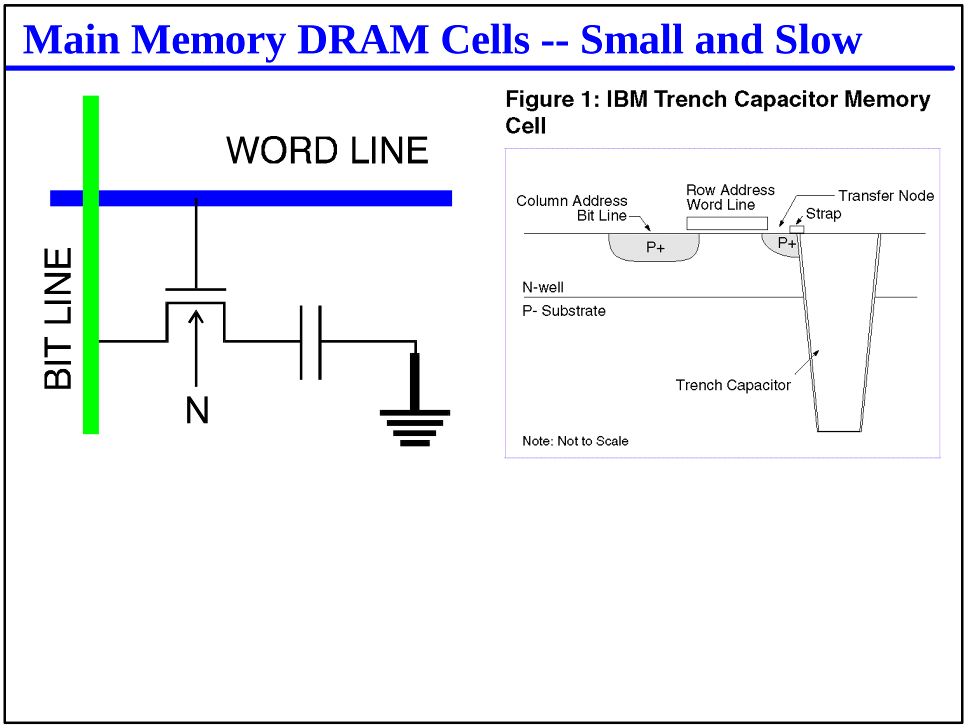
Main Memory

- **Main Memory is what programmers (think they) manipulate**
  - Program space
  - Data space
  - Commonly referred to as “physical memory” (as opposed to “virtual memory”)
- **Typically constructed from DRAM chips**
  - Multiple clock cycles to access data, but may operate in a “burst” mode once data access is started
  - Optimized for capacity, not necessarily speed
- **Latency -- determined by DRAM construction**
  - Shared pins for high & low half of address to save on packaging costs
  - Typically 2 or 3 bus cycles to begin accessing data
  - Once access initiated can return multiple data at rate of datum per bus clock

Main Memory Capacities

- **Main memory capacity is determined by DRAM chip**
  - At least 1 “bank” of DRAM chips is required for minimum memory size (*e.g.*, 4 Mbit chips arranged as 4 bits wide (1 Mbitx4) require 16 chips for a 64-bit bus --- 8 Mbyte minimum memory size)
  - Multiple banks (or bigger chips) used to increase memory capacity
- **Bandwidth -- determined by memory word width**
  - Memory words typically same width as bus
  - Peak memory bandwidth is usually one word per bus cycle
  - Sustained memory bandwidth varies with the complexity of the design
    - Sometimes multiple banks can be activated concurrently, exploiting “interleaved” memory
  - Representative main memory bandwidth is 500 MB/sec peak; 125 MB/sec sustained

DRAM OPERATION



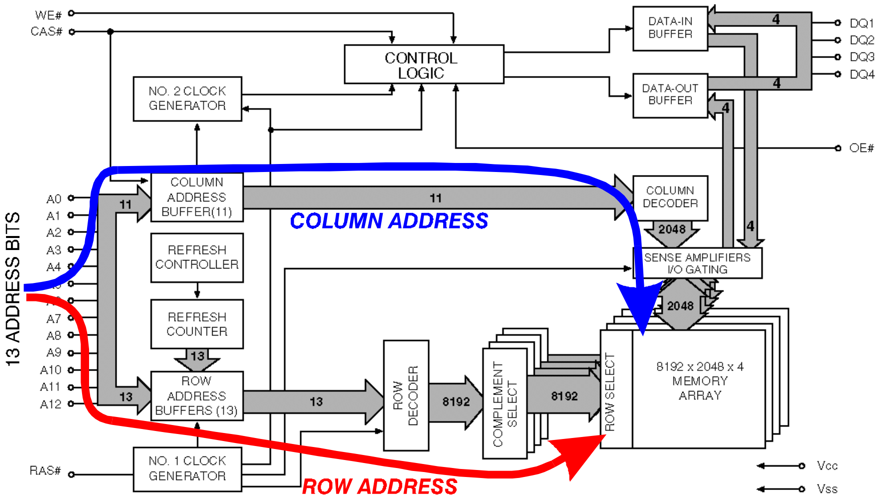
Main Memory DRAM Cells -- Small and Slow

Figure 1: IBM Trench Capacitor Memory Cell

Micron 64 Mbit DRAM

FUNCTIONAL BLOCK DIAGRAM

MT4LC16M4A7 (13 row addresses)

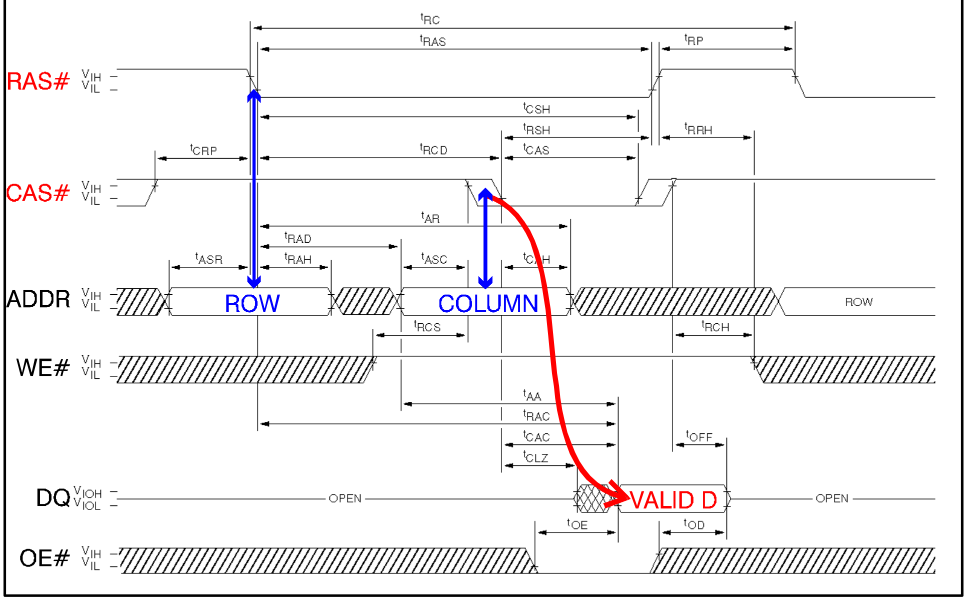


DRAM chip operation

- **Row Address Select (RAS)** RAS# RAS.L
  - Present first half of address to DRAM chip
  - Use to read row from memory array
- **Column Address Select (CAS)** CAS# CAS.L
  - Present second half of address to DRAM chip
  - Use to select bits from row for read/write
- **Cycle time**
  - RAS + CAS + rewriting data back to array
- **Refresh cycle**
  - Access to refresh capacitors
  - Needed every few milliseconds (say, 64 msec); varies with chip

DRAM Read Cycle

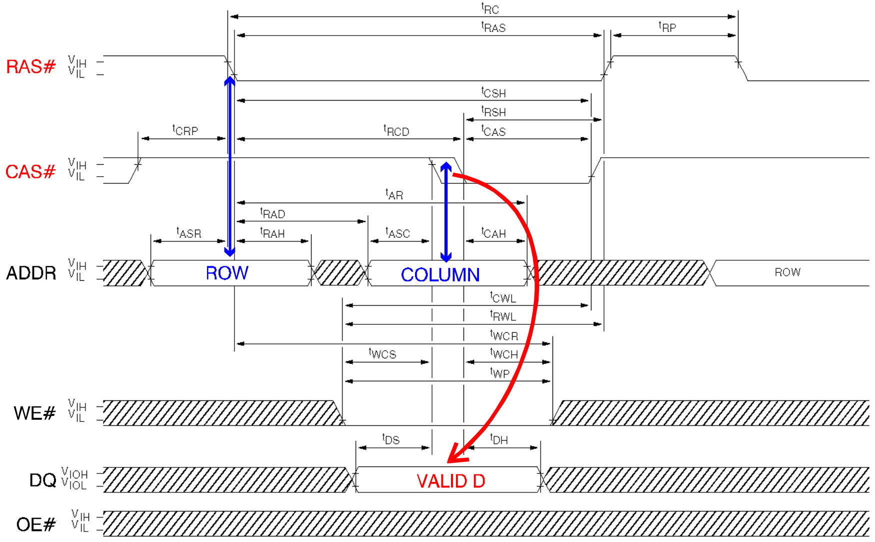
READ CYCLE



DRAM Write Cycle

EARLY WRITE CYCLE

(Micron MT4LC16M4A7)



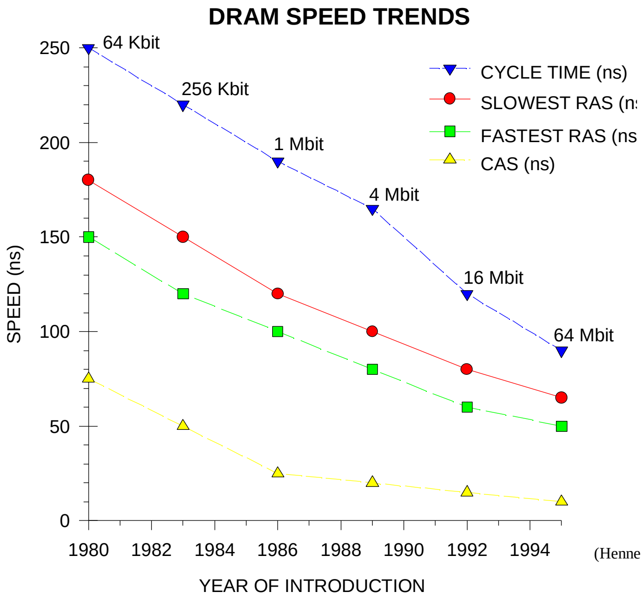
Main Memory vs. Cache Memory

- **Cache is optimized for speed**
  - On-chip when possible; usually SRAM design
  - If off-chip, single bank of SRAM chips for simplicity & speed
- **Main memory is optimized for capacity & cost**
  - Off-chip; DRAM design
  - Multiple banks of DRAM for capacity, introduces issues of:
    - Delays for buffers, chip select, address multiplexing
    - Delays for backplane if separate memory card is used
    - Delays for bus arbitration if memory is shared with I/O or multiple CPUs
- **High capacity machines have longer main memory latency**
  - Alpha L3 cache is 8 MB ...
    ... which has lower latency than accessing the 512 MB main memory DRAM
  - Embedded system with exactly 1 bank of DRAM can get rid of memory system overhead and run faster (but only for small programs)

INCREASING DRAM BANDWIDTH

Part 1 of 2 --
Exotic DRAM components
are in next lecture

Exponential Size Growth; Linear Speedups



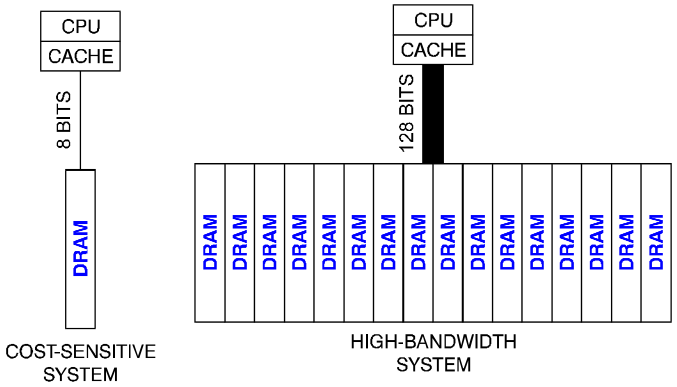
(Hennessy & Patterson Figure 5.30)

Concurrency To Speed Up DRAM Accesses

- **Parallelism: wider paths from cache to DRAM**
  - Provides high bandwidth and low latency
  - Increases cost
- **Pipelining: pipelined access to DRAM**
  - Can provide higher bandwidth with modest latency penalty
  - Often a cost-effective tradeoff, since cache is already helping with latency on most accesses
- **Replication: more than one bank of DRAM**
  - Can start accessing second DRAM bank while first DRAM bank is refreshing row
  - Can initiate accesses to many DRAM banks, then read results later

Wide Paths to DRAM

- **Low cost systems reduce board space & package count**
  - 32-byte cache block might need 32 DRAM cycles to service a miss
- **Wide path to DRAM reduces latency and increases bandwidth**
  - Only 1 DRAM cycle to provide full cache line
  - Minimum DRAM configuration might be very large (and expensive)



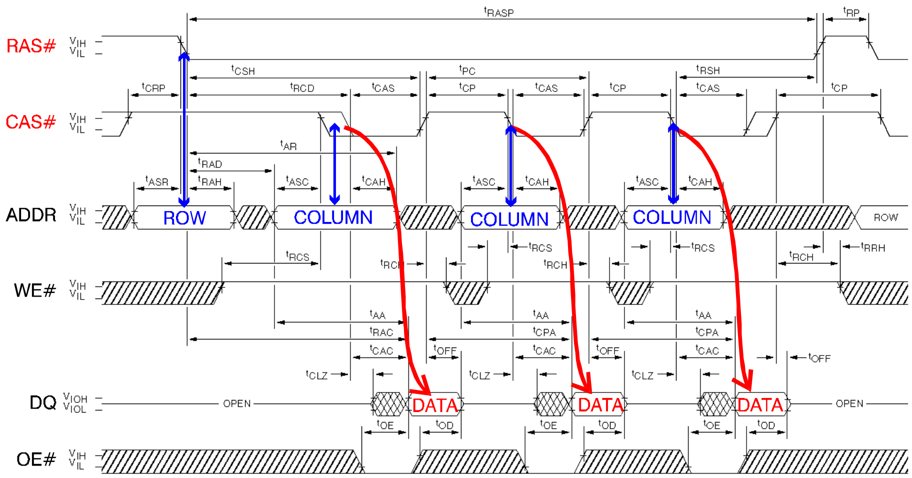
Exploiting DRAM Spatial Locality

- **Multiple CAS cycles for single RAS**
  - Can access multiple bits from same row without refreshing cells, since all bits are latched at sense amps
  - Permits slow access to initial word in DRAM, followed by fast access to subsequent words
  - A good match to servicing cache misses with block size > transfer size
- **Various modes:**
  - Nibble mode: DRAM provides several bits sequentially for every RAS
  - Fast Page mode: DRAM row can be randomly addressed with several CAS cycles
  - Static column: Same as page mode, but asynchronous CAS access

Fast Page Mode

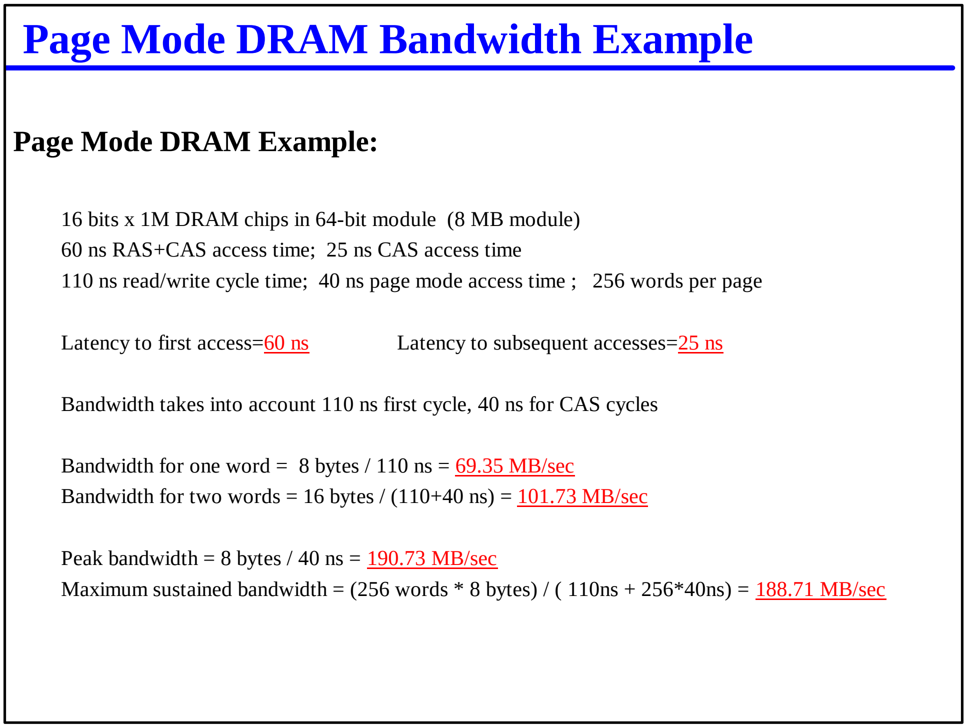
(Micron MT4LC16M4A7)

FAST-PAGE-MODE READ CYCLE



Burst Transfers from DRAM

- **Use fast page mode, *etc.*, to read several words over a modest width DRAM bank**
  - Can provide higher bandwidth with modest latency penalty
  - Often a cost-effective tradeoff, since cache is already helping with latency on most accesses



Page Mode DRAM Bandwidth Example

Page Mode DRAM Example:

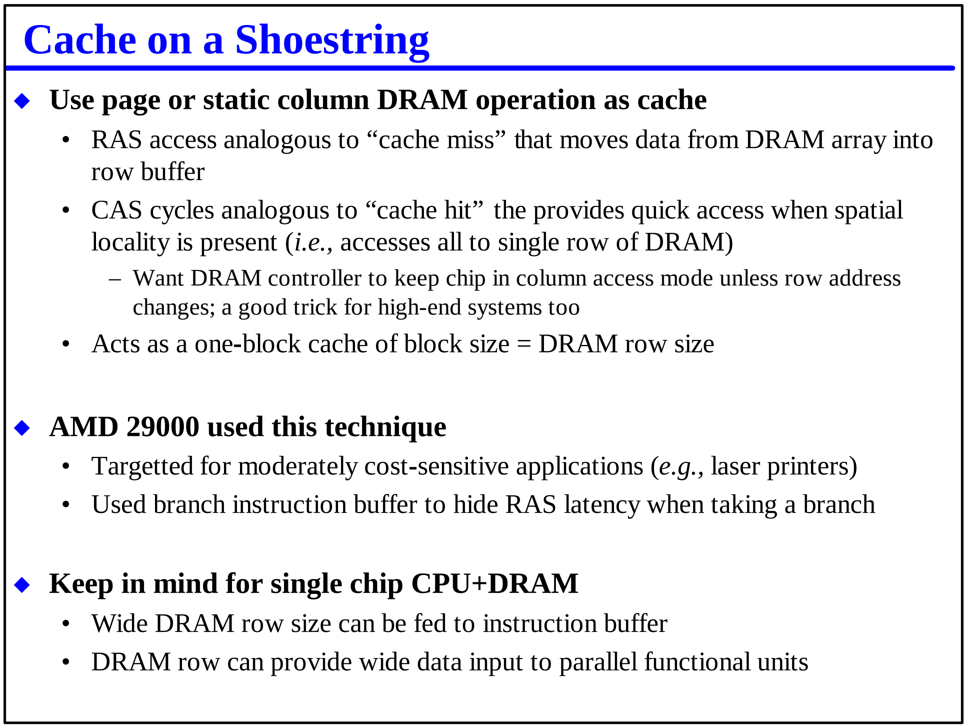
16 bits x 1M DRAM chips in 64-bit module (8 MB module)
60 ns RAS+CAS access time; 25 ns CAS access time
110 ns read/write cycle time; 40 ns page mode access time ; 256 words per page

Latency to first access=60 ns Latency to subsequent accesses=25 ns

Bandwidth takes into account 110 ns first cycle, 40 ns for CAS cycles

Bandwidth for one word = 8 bytes / 110 ns = 69.35 MB/sec
Bandwidth for two words = 16 bytes / (110+40 ns) = 101.73 MB/sec

Peak bandwidth = 8 bytes / 40 ns = 190.73 MB/sec
Maximum sustained bandwidth = (256 words * 8 bytes) / (110ns + 256*40ns) = 188.71 MB/sec



Cache on a Shoestring

- **Use page or static column DRAM operation as cache**
  - RAS access analogous to “cache miss” that moves data from DRAM array into row buffer
  - CAS cycles analogous to “cache hit” the provides quick access when spatial locality is present (*i.e.*, accesses all to single row of DRAM)
    - Want DRAM controller to keep chip in column access mode unless row address changes; a good trick for high-end systems too
  - Acts as a one-block cache of block size = DRAM row size

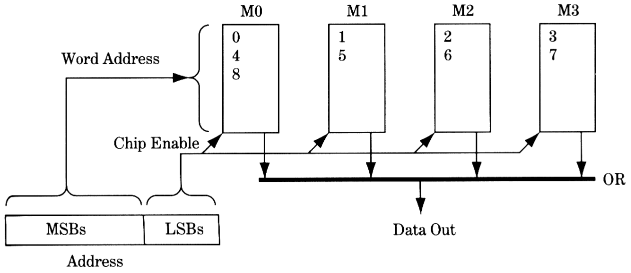
- **AMD 29000 used this technique**
  - Targetted for moderately cost-sensitive applications (*e.g.*, laser printers)
  - Used branch instruction buffer to hide RAS latency when taking a branch

- **Keep in mind for single chip CPU+DRAM**
  - Wide DRAM row size can be fed to instruction buffer
  - DRAM row can provide wide data input to parallel functional units

INTERLEAVED MEMORY

Multiple Memory Banks

- **Can increase available bandwidth**
  - Multiple memory banks take turns supplying data -- *interleaved* access
  - Data can be streamed from memory faster than DRAM cycle time
- **Can reduce latency when multiple memory banks are active**
  - Multiple banks can be used to hide cycle time
  - Multiple memory references can be serviced concurrently
- **Typically number of banks is a power of 2 for addressing ease**
  - Use lowest bits of memory address to select bank
  - Up to 128 banks on supercomputers (Cray 2 and NEC SX/3)



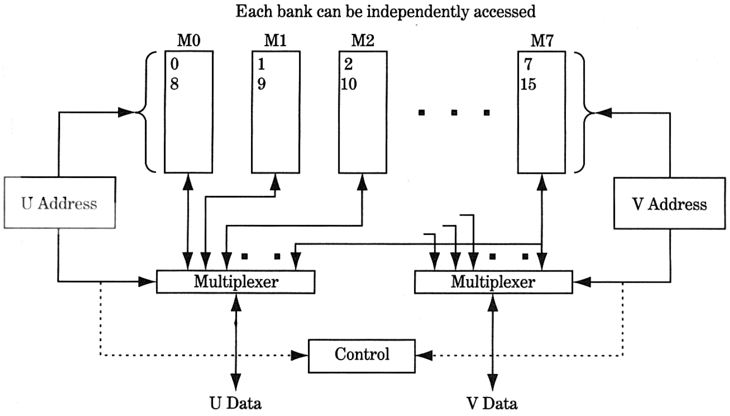
(Cragon Figure 5.1)

Interleaved Bandwidth Increase

- **Banks take turns supplying data**
  - Permits pipelining address and data
  - Equivalent performance to page mode access of DRAM
  - Address *x* is in bank *x* MOD *b* where *b* is number of banks

Interleaved Memory As Dual-Port Alternative

- **Multiple independent memory banks have latent bandwidth available**
  - Can access *m* of *n* single-ported banks simultaneously
  - If *m* << *n* , chances for bank conflict are reduced
  - If bank conflict occurs, stall one access port (can also simplify data dependency handling)
- **Example: Pentium interleaved data cache**



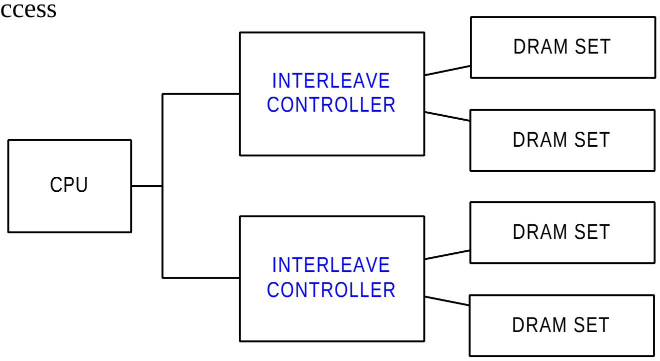
(Cragon Figure 5.6)

Interleaved Latency Decrease -- cycle time

- **Multiple banks hide refresh portion of cycle time**
  - For example, ping-ponging between two banks hides end of cycle time

Interleaved Latency Decrease -- concurrency

- **Multiple banks service multiple pending memory requests**
  - If bus runs faster than DRAM cycles, can have multiple pending memory requests
    - Assumes non-blocking cache or uncached memory accesses
  - Multiple requests can be to arbitrary location; no reliance on spatial locality
    - Memory requests must be to different banks to prevent conflicts
  - BUT, time to go through interleaving process costs time for a single, isolated memory access



Why Interleaved Memory?

- **Historically important on machines that:**
  - Didn’t have cache memory
  - Used magnetic core memory instead of DRAM
  - Had multiple CPUs sharing common memory

- **But, becoming prevalent because of generality of the above situations**
  - Large gap between CPU-type memory technology and Main Memory-type techology
    - Magnetic core instead of transistors
    - DRAM chips instead of SRAM chips
    - Off-chip cache instead of on-chip cache
  - Need for high bandwidth in successive accesses having poor spatial locality
    - Superscalar access to multiple data locations
    - Multiprocessing accessing shared main memory

Practical Limits -- Minimum Memory Size

- **Minimum memory size can be a cost constraint on all but the biggest systems**
  - Assume 64-bit data bus; 64 Mbit DRAMs in 1-bit wide configuration
    - 64-bit DRAM module will have 64 chips with 512 MB of DRAM
  - Assume 8-way interleaving
    - Minimum system size is 64 x 8 = 512 DRAMs = 4 GB of DRAM
    - Memory can only be added in 4 GB chunks
- **More importantly (and independent of current technology):**

  minimum # DRAM chips = (# memory banks * access width) / DRAM chip width

- **Cost-effective interleaving solutions**
  - Use wide DRAM chips (8-bit wide means $^1/_8$ as many chips as 1-bit wide)
  - Permit smaller interleave factors on low-end machines (expanded memory gives larger interleave factors
  - Use multiple cycles in page mode to retrieve data (Titan trick discussed later)

REVIEW

Review

- **Main memory tradeoffs**
  - DRAM optimized for capacity more than for speed
  - Exploit DRAM operation to provide bandwidth (*e.g.*, fast page mode)
  - Minimum possible DRAM size can be a cost constraint
- **Interleaved memory access**
  - Helps with latency by hiding refresh/rewrite time & reducing access conflicts
  - Multiple banks can provide multiple concurrent accesses