14
Main Memory Performance

18-548/15-548 Memory System Architecture
Philip Koopman
October 21, 1998

Required Reading: **Cragon 5.1.6 - 5.1.7**
Fast DRAM paper from EDN
Supplemental Reading: Siewiorek & Koopman 5.2.2

Carnegie Mellon

Assignments

- **By next class read about Vector Architecture:**
  - Cragon 11.0-11.2.2, 11.4-11.6.3
  - Supplemental Reading:
    - Hennessy & Patterson B.1-B.5 (but some of the terminology is confusing)
    - Siewiorek & Koopman 5.0-5.9
    - Siewiorek, Bell & Newell Chapter 44 (Cray 1) -- photocopy in library folder
- **Homework 8 due *Monday* October 26**
- **Lab 4 due October 23 (on Friday)**

Where Are We Now?

- **Where we've been:**
  - Main memory architecture
- **Where we're going today:**
  - Getting performance out of main memory
- **Where we're going next:**
  - Vector computing

Preview

- **Strided access & performance**
  - Techniques to reduce bank conflicts on interleaved memory
- **"Exotic" DRAM technology**
  - EDO DRAM
  - SDRAM
  - Cached DRAM
  - Rambus
- **Titan memory subsystem example**
  - Mini-supercomputer memory subsystem

Footnote: C vs. Fortran Array Organization

- **C is row-major, 0-based for arrays**
  - Memory layout of 4x4 array ---
    - Access all elements in ROW sequentially (row-by-row storage)
  - Hennessy & Patterson are C-based; workstation tradition

```
[0,0] [0,1] [0,2] [0,3]
...[1,0] [1,1] [1,2] [1,3]
...[2,0] [2,1] [2,2] [2,3]
...[3,0] [3,1] [3,2] [3,3]
```

- **Fortran is column-major, 1-based for arrays**
  - Memory layout of 4x4 array ---
    - Access all elements in COLUMN sequentially (column-by-column storage)
  - Cragon is Fortran-based; mainframe & supercomputer tradition

```
(1,1) (2,1) (3,1) (4,1)
...(1,2) (2,2) (3,2) (4,2)
...(1,3) (2,3) (3,3) (4,3)
...(1,4) (2,4) (3,4) (4,4)
```

INTERLEAVED MEMORY PERFORMANCE

Matrix in Interleaved Memory

- **Example 4x4 array stored in 4-way interleaved memory (column-major/Fortran style):**
  - Sequential elements are stored in consecutive memory banks

$$\begin{bmatrix} 1,1 & 1,2 & 1,3 & 1,4 \\ 2,1 & 2,2 & 2,3 & 2,4 \\ 3,1 & 3,2 & 3,3 & 3,4 \\ 4,1 & 4,2 & 4,3 & 4,4 \end{bmatrix}$$

	M0	M1	M2	M3
0	1,1	2,1	3,1	4,1
1	1,2	2,2	3,2	4,2
2	1,3	2,3	3,3	4,3
3	1,4	2,4	3,4	4,4

- **Sequential accesses to array columns access sequential banks**
  - Diagonals are also conflict-free
- **Sequential accesses to array rows conflict on same bank**
  - True any time array size is evenly divisible by number of banks
  - Similar to problems with cache conflicts (number of banks is analogous to number of sets in cache)

Strided Access

- **Strided access with stride *k* means touching every *k*th memory element**
  - Stride = 1 is sequential access (0, 1, 2, 3, 4, 5, 6, ...)
  - Stride = 2 is (0, 2, 4, 6, 8, ...)
  - Stride = k is (0, k, 2k, 3k, 4k, ...)
- **Strides > 1 commonly found in multidimensional data**
  - Row accesses (stride=N) & diagonal accesses (stride=N+1)
  - Scientific computing (e.g., matrix multiplication)
  - Image processing (image rows and columns)
  - Radar/Sonar processing (angle vs. elevation)
  - In many cases arrays are a power of 2 size, promoting bank conflicts

Conflict Reduction Techniques

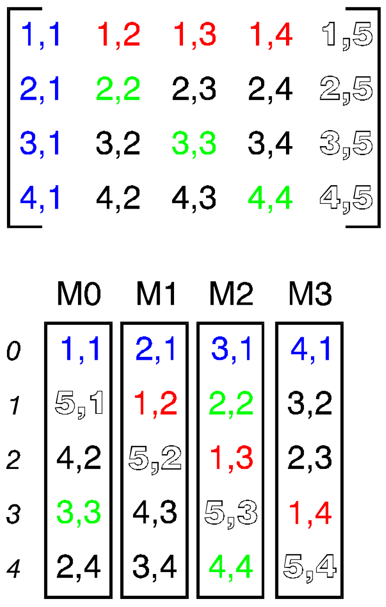
- **Software techniques**
  - Array size change
- **Address modification**
  - Skewed addressing
- **Interleave hardware techniques**
  - Prime number interleaving
  - Superinterleaving

Note: the above techniques can also be used to reduce cache conflicts

- *Interleaved memory bank conflicts are analogous to cache set conflicts (i.e., interleaving works on caches too...)*

SW Technique: Array Size Change

- **Software/compiler solution**
  - Allocate array size relatively prime to number of memory banks
  - Before:
    `foo(SIZE, SIZE)`
  - After:
    `foo(SIZE, SIZE+1)`
  - Row and column accesses have no conflicts
  - Diagonal access uses only 2 banks of 4



$$\begin{bmatrix} 1,1 & 1,2 & 1,3 & 1,4 & 1,5 \\ 2,1 & 2,2 & 2,3 & 2,4 & 2,5 \\ 3,1 & 3,2 & 3,3 & 3,4 & 3,5 \\ 4,1 & 4,2 & 4,3 & 4,4 & 4,5 \end{bmatrix}$$

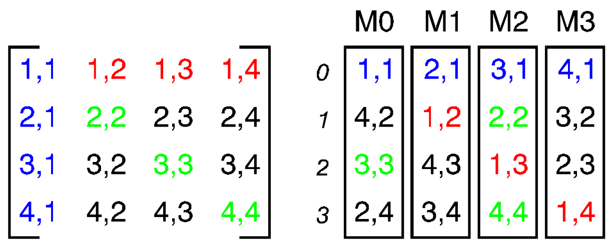
	M0	M1	M2	M3
0	1,1	2,1	3,1	4,1
1	5,1	1,2	2,2	3,2
2	4,2	5,2	1,3	2,3
3	3,3	4,3	5,3	1,4
4	2,4	3,4	4,4	5,4

Skewed Addressing

- **Hardware maps addresses into different modules**
  - Example: skew = 1
  - Same effectiveness as array size change example

EXAMPLE: SKEW FACTOR = 1

ARRAY ELEMENT	NORMAL ADDRESS	MODULE OFFSET	MODULE NUMBER
(1,1)	0	0	0
(2,1)	1	0	1
(3,1)	2	0	2
(4,1)	3	0	3
(1,2)	4	1	1
(2,2)	5	1	2
(3,2)	6	1	3
(4,2)	7	1	0
(1,3)	8	2	2
(2,3)	9	2	3
(3,3)	10	2	0
(4,3)	11	2	1
(1,4)	12	3	3
(2,4)	13	3	0
(3,4)	14	3	1
(4,4)	15	3	2



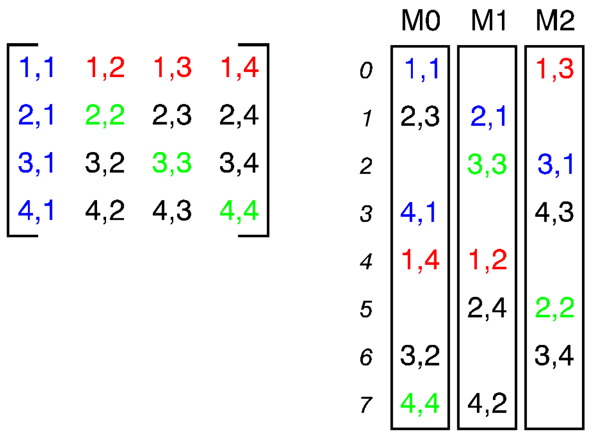
- **Example skewed address computation:**
  - Word address within module = address / #banks
  - Module # = ((address/#banks) + (address * skew)) MOD #banks

Prime Number Interleaving

- **Conflicts happen when stride is an even multiple (or divisor) of interleave factor**
  - Power of 2 bank size is easy to build -- uses low order bits for bank number
  - Power of 2 is a common array size
  - 2 divides into all even numbers (and people naturally use even numbers)
- **Prime number interleave reduces possibility for conflicts**
  - Good tricks available for 2^n+1 and 2^n-1 banks
  - Burroughs Scientific Processor (BSP) used interleaving with m=17

Prime Number Interleaving: 2^N+1 Banks

- **Bank_# = address MOD num_banks**
  - Fast mod discussed in Hennessy & Patterson Exercise 5.10
- **Bank_Address = address MOD words_in_bank**
  - If words_in_bank is a power of 2, then this is simply a shift operation



$$\begin{bmatrix} 1,1 & 1,2 & 1,3 & 1,4 \\ 2,1 & 2,2 & 2,3 & 2,4 \\ 3,1 & 3,2 & 3,3 & 3,4 \\ 4,1 & 4,2 & 4,3 & 4,4 \end{bmatrix}$$

	M0	M1	M2
0	1,1		1,3
1	2,3	2,1	
2		3,3	3,1
3	4,1		4,3
4	1,4	1,2	
5		2,4	2,2
6	3,2		3,4
7	4,4	4,2	

EXAMPLE: 3 BANKS OF 8 WORDS

ARRAY ELEMENT	NORMAL ADDRESS	MODULE OFFSET	MODULE NUMBER
(1,1)	0	0	0
(2,1)	1	1	1
(3,1)	2	2	2
(4,1)	3	3	0
(1,2)	4	4	1
(2,2)	5	5	2
(3,2)	6	6	0
(4,2)	7	7	1
(1,3)	8	0	2
(2,3)	9	1	0
(3,3)	10	2	1
(4,3)	11	3	2
(1,4)	12	4	0
(2,4)	13	5	1
(3,4)	14	6	2
(4,4)	15	7	0
	16	0	1
	17	1	2
	18	2	0
	19	3	1
	20	4	2
	21	5	0
	22	6	1
	23	7	2

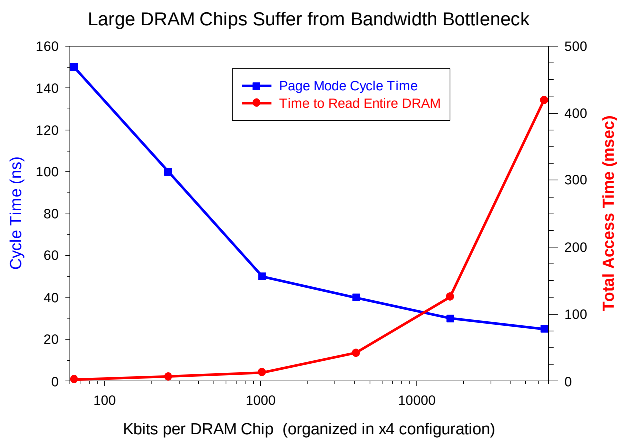
Superinterleaving

- **Assume that there are *m* memory bus cycles per module cycle time**
  - *e.g.*, if memory cycle time is 4 bus clocks, *m*=4
- **"Normal" interleaving has *n* banks, with *n* £ *m***
  - In best case, such as sequential access, all banks can be busy (n=m)
- **Superinterleaving has *n* > *m* banks**
  - In other words, there are more banks than can possibly be active at once; extra banks don't help with raw bandwidth ability
  - Used to reduce chance of conflict (may be less likely that stride will be a multiple of *n* than a multiple of *m*, since n is larger)
  - Example: 8 memory banks on a bus that can only keep 4 banks busy

EXOTIC DRAM ARCHITECTURES

The DRAM Bandwidth Gap

- **Bandwidth is growing more slowly than DRAM capacity**
  - Takes longer and longer to read entire contents of DRAM chip

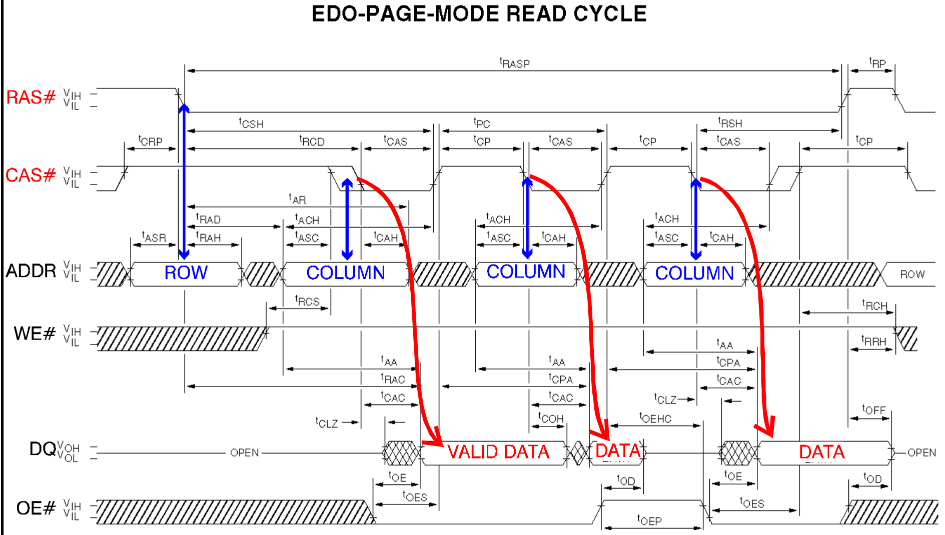


EDO DRAM

- **EDO = Extended Data Out**
- **Stretches data output time**
  - Traditionally data goes invalid on rising CAS#
  - EDO DRAM ties this to OE# instead
  - A minor hack, but eases timing for repeated accesses to a single DRAM chip

Extended Data Out (EDO) DRAM

(Micron MT4LC16M4G3)

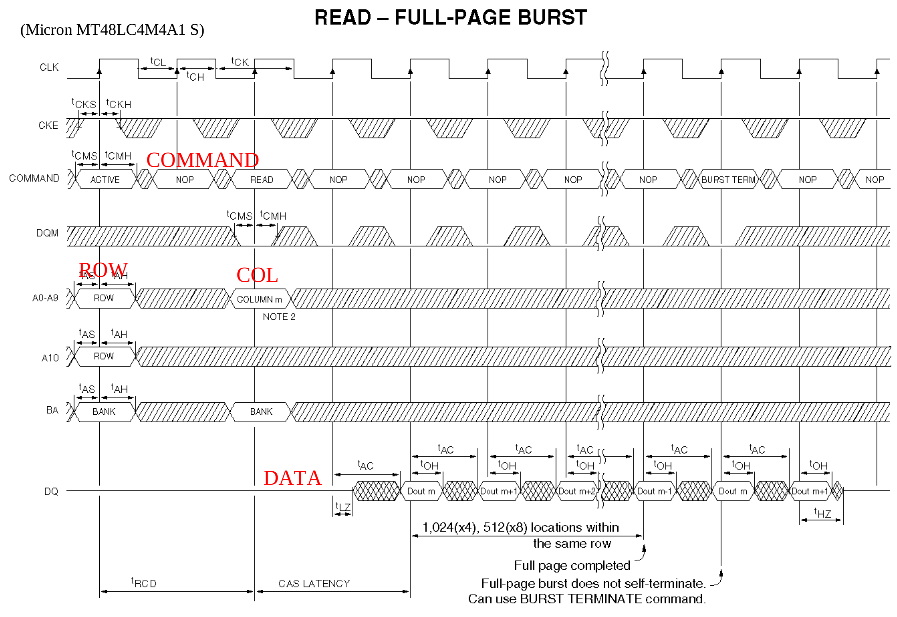


Synchronous DRAM

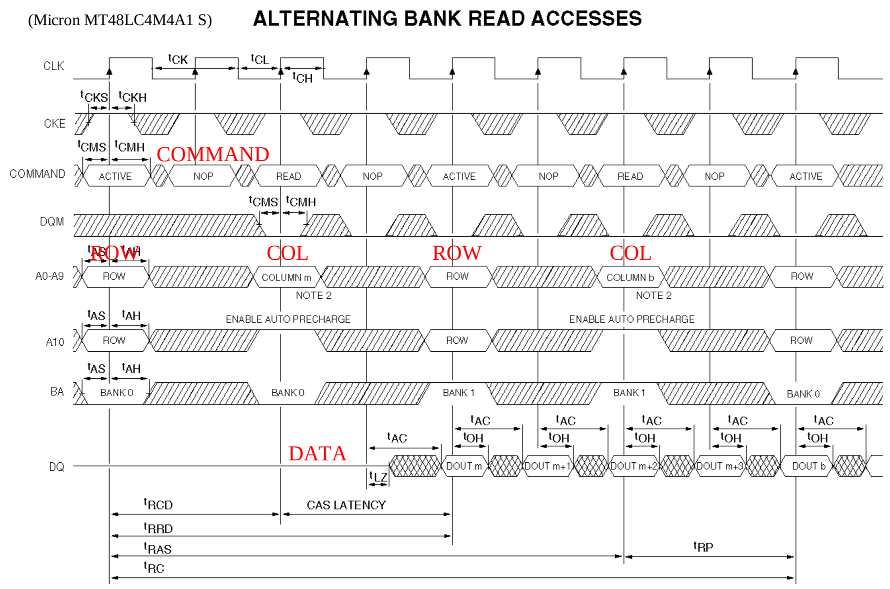
- **Adopted for Pentium use**
- **Optimized for burst accesses**
  - Especially good for cache block refill > bus width
  - Two simultaneous rows accessed at one time (can access both in fast page mode)
    - *Interleaving on-chip*
    - 2 or 4 interleaved memories on a single chip
  - Clocked interface rather than RAS/CAS control signals used for timing
    - Beginning of a switch to a communication-network interface to DRAM chips
  - Supports 5-1-1-1 timing for Pentium L2 cache miss
  - Self-contained refresh

Synchronous DRAM -- Burst Read

(Micron MT48LC4M4A1 S)



Synchronous DRAM -- Alternating Banks

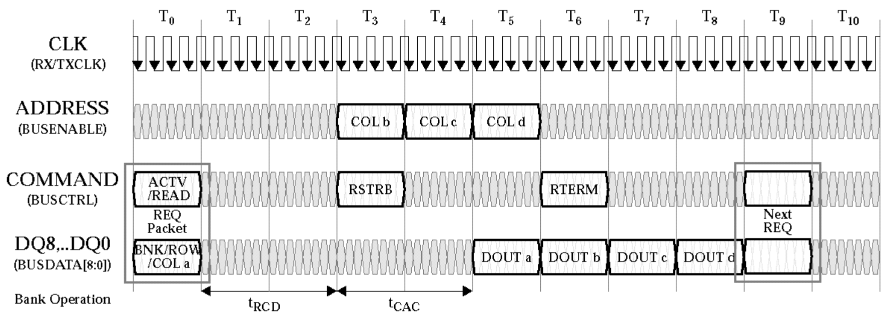


Rambus DRAM

- **Proprietary byte-wide bi-directional bus**
  - Reduced voltage swing (600 mV swing centered at 2.2 V) for speed
  - 500-600 MB/sec per channel; high bandwidth with small number of chips
  - Two or four 1KByte or 2KByte sense amplifiers used as high speed caches
  - Interleaving among the banks
- **Very high bandwidth**
  - But, high latency -- minimum size 8 byte transfer

Rambus

- 8-byte octets are smallest transferable unit



(a) BANK ACTIVATE AND RANDOM READ CYCLES WITHIN A PAGE

(Rambus generic 16 Mbit RDRAM)

Other Special DRAM architectures

- **Video DRAM**
  - Has shift-out register to feed bits to video display
- **Cached DRAM**
  - Traditional cache structure on DRAM in addition to row-oriented buffering
  - “Smart” prefetching logic on DRAM chip to anticipate accesses (DRAM-based stream buffer mechanism?)
- **IRAM -- Intelligent RAM**
  - Parallel processing with a processor on each DRAM chip
- **Real-world stumbling blocks**
  - Process technology differences between DRAM and logic/SRAM fabrication techniques
  - Processor+memory on-chip is limited in memory size, no matter how big DRAM chips get

EXAMPLE:
TITAN MEMORY SUBSYSTEM

Titan Mini-Super Computer

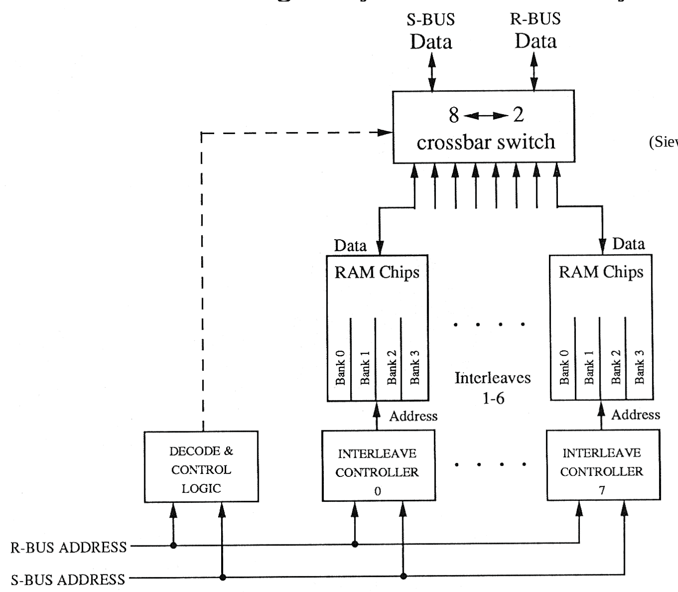
- **"Single-User supercomputer" -- significant fraction of supercomputer performance at a high-end workstation price**
  - Design started 1986. Company name: Dana -> Ardent -> Stardent
  - Up to 4 processors (integer/vector pairs)
  - 16 MFLOPS (single processor) peak for ~$100,000 in 1988
- **Design based on traditional supercomputer approach**
  - Gate arrays used for vector floating point unit
  - MIPS R2000 used for integer control processor
  - Hardware support for 3-D graphics

FIGURE 1. The Stardent 1500 "Titan" series graphics supercomputer: system overview. Copyright © 1988, Stardent Computer Inc.

(Siewiorek & Koopman Plate 1)

Titan Interleaved Memory

- **Two data buses accessing 8-way interleaved memory**



(Siewiorek & Koopman Figure 5.2)

Wide-Word Interleaving via Page Mode

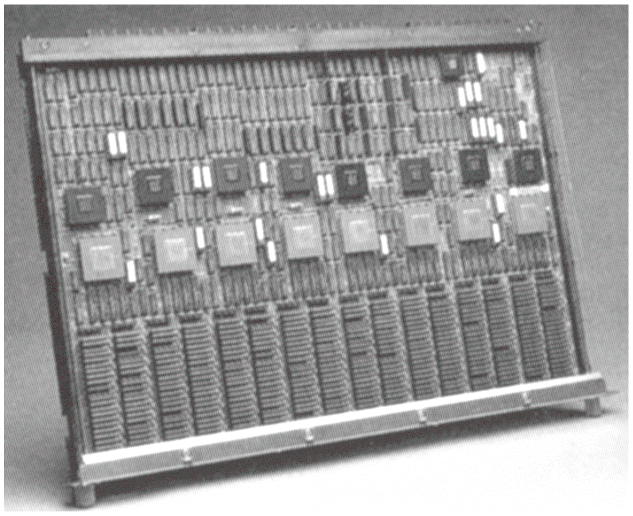
- **Each memory bank 32 bits wide**
  - 64-bit words read using 2-clock page mode access

1st:	ADDRESS	CONTROL	ACCESS	DATA LO	DATA HI	PRECHARGE		
2nd:		ADDRESS	CONTROL	ACCESS	DATA LO	DATA HI	PRECHARGE	
3rd:			ADDRESS	CONTROL	ACCESS	DATA LO	DATA HI	PRECHARGE

Time ®

  - Cuts cost for interleaving in half with small performance hit
    - Data paths through cross bar half as wide
    - Number of minimum chips required in system half as big
    - Still provides 64 bits of data per bus clock (low & high from different interleaves)

Titan Memory Board



8 Interleave Controllers & Crossbar Switches

SIP DRAM chips

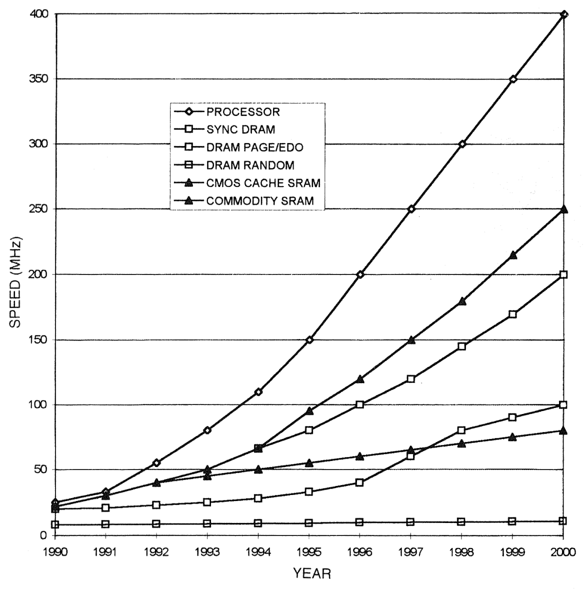
(Siewiorek & Koopman Plate 4.b)

Titan Memory Tradeoffs

- **Wide-word interleaving for cost savings**
  - Interleave data paths only 32 bits, but supports 64-bit accesses
  - 1 clock access latency penalty
  - Doubles number of interleaves available at comparable cost
- **Interleave expansion with memory expansion**
  - Adding second memory board supports 16-way interleaving
- **Dual bus access to main memory**
  - Required for balanced memory bandwidth (discussed later)
  - Interleaved memory use to (usually) support dual access with single-ported DRAM
  - Interleaved memory used to permit streaming results to processors
  - 256K x 4 DRAMs used to reduce minimum memory size over 1M x 1 DRAMs
- **Provides atomic access primitive in memory subsystem**
  - Fetch and increment-if-negative

REVIEW

An Optimistic Look at Memory Bandwidth



High Performance Memories,
Prince, Figure 1.1

Review

- **Interleaved memory access**
  - Helps with latency by hiding refresh time & reducing access conflicts
  - Multiple banks can provide multiple concurrent accesses
- **Strided memory access**
  - Strided array accesses might not be evenly distributed among banks
    - Software solution -- rearrange access patterns
    - Hardware solution -- make common strides access different banks
- **“Exotic” DRAM technology**
  - Application of general architecture techniques within a single DRAM chip
- **Titan as an example memory subsystem**
  - High bandwidth, but with some cost-cutting tricks