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

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
  
- **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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,0]</td>
<td>[0,1]</td>
<td>[0,2]</td>
<td>[0,3]</td>
</tr>
<tr>
<td>[1,0]</td>
<td>[1,1]</td>
<td>[1,2]</td>
<td>[1,3]</td>
</tr>
<tr>
<td>[2,0]</td>
<td>[2,1]</td>
<td>[2,2]</td>
<td>[2,3]</td>
</tr>
<tr>
<td>[3,0]</td>
<td>[3,1]</td>
<td>[3,2]</td>
<td>[3,3]</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th></th>
<th>M0</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>1,2</td>
<td>1,3</td>
<td>1,4</td>
<td>0,1</td>
</tr>
<tr>
<td>2,1</td>
<td>2,2</td>
<td>2,3</td>
<td>2,4</td>
<td>1,2</td>
</tr>
<tr>
<td>3,1</td>
<td>3,2</td>
<td>3,3</td>
<td>3,4</td>
<td>2,3</td>
</tr>
<tr>
<td>4,1</td>
<td>4,2</td>
<td>4,3</td>
<td>4,4</td>
<td>3,4</td>
</tr>
</tbody>
</table>

- 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, $k$, 2$k$, 3$k$, 4$k$, ...)

- 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:
    \[ \text{foo}(\text{SIZE}, \text{SIZE}) \]
  - After:
    \[ \text{foo}(\text{SIZE}, \text{SIZE}+1) \]
  - Row and column accesses have no conflicts
  - Diagonal access uses only 2 banks of 4

\[
\begin{array}{cccc}
1,1 & 1,2 & 1,3 & 1,4 & 1,5 \\
2,1 & 2,2 & 2,3 & 2,4 & 2,5 \\
3,1 & 3,2 & 3,3 & 3,4 & 3,5 \\
4,1 & 4,2 & 4,3 & 4,4 & 4,5 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1,1 & 2,1 & 3,1 & 4,1 \\
1 & 5,1 & 1,2 & 2,2 & 3,2 \\
2 & 4,2 & 5,2 & 1,3 & 2,3 \\
3 & 3,3 & 4,3 & 5,3 & 1,4 \\
4 & 2,4 & 3,4 & 4,4 & 5,4 \\
\end{array}
\]
Skewed Addressing

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

<table>
<thead>
<tr>
<th>Module</th>
<th>M0</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>1,1</td>
<td>2,1</td>
<td>3,1</td>
<td>4,1</td>
</tr>
<tr>
<td>2,1</td>
<td>4,2</td>
<td>1,2</td>
<td>2,2</td>
<td>3,2</td>
</tr>
<tr>
<td>3,1</td>
<td>3,3</td>
<td>4,3</td>
<td>1,3</td>
<td>2,3</td>
</tr>
<tr>
<td>4,1</td>
<td>2,4</td>
<td>3,4</td>
<td>4,4</td>
<td>1,4</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EXAMPLE: 3 BANKS OF 8 WORDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
</tr>
<tr>
<td>ELEMENT</td>
</tr>
<tr>
<td>(1,1)</td>
</tr>
<tr>
<td>(2,1)</td>
</tr>
<tr>
<td>(3,1)</td>
</tr>
<tr>
<td>(4,1)</td>
</tr>
<tr>
<td>(1,2)</td>
</tr>
<tr>
<td>(2,2)</td>
</tr>
<tr>
<td>(3,2)</td>
</tr>
<tr>
<td>(1,3)</td>
</tr>
</tbody>
</table>


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 $n \leq m$**
  - 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
**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)

<table>
<thead>
<tr>
<th>READ – FULL-PAGE BURST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COMMAND</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>ROW</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>COL</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>DATA</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- Full-page burst does not self-terminate.
- Can use BURST TERMINATE command.
Synchronous DRAM -- Alternating Banks

(Micron MT48LC4M4A1 S)  

ALTERNATING BANK READ ACCESES

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

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

(Siewiorek & Koopman Plate 1)
Titan Interleaved Memory

- Two data buses accessing 8-way interleaved memory

Wide-Word Interleaving via Page Mode

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

- 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

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
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