DRAM Parameters

- DRAM parameters
  - Large capacity: e.g., 1-4Gb
    - Arranged as square
      + Minimizes wire length
      + Maximizes refresh efficiency
  - Narrow data interface: 1–16 bit
    - Cheap packages → few bus pins
    - Pins are expensive
  - Narrow address interface: N/2 bits
    - 16Mb DRAM had a 12-bit address bus
    - How does that work?
Access Time and Cycle Time

- DRAM access much slower than SRAM
  - More bits → longer wires
  - Buffered access with two-level addressing
  - SRAM access latency: 2–3ns
  - DRAM access latency: 20-35ns

- 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
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)
    • Core more closely resembles today’s disks than memory
    • “Core dump” is legacy terminology
  • 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
A Few Flavors of DRAM

• DRAM comes in several different varieties
  • Go to Dell.com and see what kinds you can get for your laptop

• SDRAM = synchronous DRAM
  • Fast, clocked DRAM technology
  • Very common now
  • Several flavors: DDR, DDR2, DDR3

• RDRAM = Rambus DRAM
  • Very fast, expensive DRAM
DRAM Packaging

• DIMM = dual inline memory module
  • E.g., 8 DRAM chips, each chip is 4 or 8 bits wide
DRAM: A Vast Topic

- Many flavors of DRAMs
  - DDR3 SDRAM, RDRAM, etc.
- Many ways to package them
  - SIMM, DIMM, FB-DIMM, etc.
- Many different parameters to characterize their timing
  - $t_{RC}$, $t_{RAC}$, $t_{RCD}$, $t_{RAS}$, etc.
- Many ways of using row buffer for “caching”
- Etc.
- There’s at least one whole textbook on this topic!
  - And it has ~1K pages
- We could, but won’t, spend rest of semester on DRAM
This Unit: Main Memory

- Memory hierarchy review
- DRAM technology
  - A few more transistors
  - Organization: two level addressing
- Building a memory system
  - Bandwidth matching
  - Error correction
- Organizing a memory system
- Virtual memory
  - Address translation and page tables
  - A virtual memory hierarchy

Application

OS

Compiler

Firmware

CPU

I/O

Memory

Digital Circuits

Gates & Transistors

© 2009 Daniel J. Sorin from Roth
Building a Memory System

- How do we build an efficient main memory out of standard DRAM chips?
  - How many DRAM chips?
  - What width/speed (data) bus to use?
    - Assume separate address bus
An Example Memory System

- Parameters
  - 32-bit machine
  - L2 with 32B blocks (must pull 32B out of memory at a time)
  - 4Mx16b DRAMs, 20ns access time, 40ns cycle time
    - Each chip is 4Mx2B = 8 MB
  - 100MHz (10ns period) data bus
  - 100MHz, 32-bit address bus

- How many DRAM chips?
- How wide to make the data bus?
First Memory System Design

- 1 DRAM + 16b (=2B) bus
  - Access time: 630ns
    • Not including address
  - Cycle time: 640ns
    • DRAM ready to handle another miss
  - Observation: data bus idle 75% of time!
    • We have over-designed bus
    • Can we use a cheaper bus?
Second Memory System Design

- 1 DRAM + 4b bus
  - One DRAM chip, don’t need 16b bus
  - DRAM: 2B / 40ns → 4b / 10ns
  - Balanced system → match bandwidths

- Access time: 660ns (30ns longer=+4%)
- Cycle time: 640ns (same as before)
  + Much cheaper!

<table>
<thead>
<tr>
<th>T (ns)</th>
<th>DRAM</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>[31:30]</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>[31:30]</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>refresh</td>
<td>[31H]</td>
</tr>
<tr>
<td>40</td>
<td>refresh</td>
<td>[31L]</td>
</tr>
<tr>
<td>50</td>
<td>[29:28]</td>
<td>[30H]</td>
</tr>
<tr>
<td>60</td>
<td>[29:28]</td>
<td>[30L]</td>
</tr>
<tr>
<td>70</td>
<td>refresh</td>
<td>[29H]</td>
</tr>
<tr>
<td>80</td>
<td>refresh</td>
<td>[29L]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>600</td>
<td>[1:0]</td>
<td>[2H]</td>
</tr>
<tr>
<td>610</td>
<td>[1:0]</td>
<td>[2L]</td>
</tr>
<tr>
<td>620</td>
<td>refresh</td>
<td>[1H]</td>
</tr>
<tr>
<td>640</td>
<td>refresh</td>
<td>[1L]</td>
</tr>
<tr>
<td>650</td>
<td>[0H]</td>
<td></td>
</tr>
<tr>
<td>660</td>
<td>[0L]</td>
<td></td>
</tr>
</tbody>
</table>
Third Memory System Design

- How fast can we go?
- 16 DRAM chips + 32B bus
  - **Stripe data across chips**
  - Byte M in chip \((M/2)\%16\) (e.g., byte 38 is in chip 3)
  - Access time: 30ns
  - Cycle time: 40ns
    - 32B bus is very expensive

<table>
<thead>
<tr>
<th>T (ns)</th>
<th>DRAM0</th>
<th>DRAM1</th>
<th>DRAM15</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>[31:30]</td>
<td>[29:28]</td>
<td>[1:0]</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>[31:30]</td>
<td>[29:28]</td>
<td>[1:0]</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[31:0]</td>
</tr>
<tr>
<td>40</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td></td>
</tr>
</tbody>
</table>
Latency and Bandwidth

• In general, given bus parameters...
  • Find smallest number of chips that minimizes cycle time
  • Approach: match bandwidths between DRAMs and data bus
    • If they don’t match, you’re paying too much for the one with more bandwidth
**Fourth Memory System Design**

- 2B bus
  - Bus b/w: 2B/10ns
  - DRAM b/w: 2B/40ns
  - 4 DRAM chips
  - Access time: 180ns
  - Cycle time: 160ns

<table>
<thead>
<tr>
<th>T (ns)</th>
<th>DRAM0</th>
<th>DRAM1</th>
<th>DRAM2</th>
<th>DRAM3</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>[31:30]</td>
<td>[29:28]</td>
<td>[27:26]</td>
<td>[25:24]</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>[31:30]</td>
<td>[29:28]</td>
<td>[27:26]</td>
<td>[25:24]</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[31:30]</td>
</tr>
<tr>
<td>40</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[29:28]</td>
</tr>
<tr>
<td>50</td>
<td>[23:22]</td>
<td>[21:20]</td>
<td>[19:18]</td>
<td>[17:16]</td>
<td>[27:26]</td>
</tr>
<tr>
<td>60</td>
<td>[23:22]</td>
<td>[21:20]</td>
<td>[19:18]</td>
<td>[17:16]</td>
<td>[25:24]</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>110</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[15:14]</td>
</tr>
<tr>
<td>120</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[13:12]</td>
</tr>
<tr>
<td>130</td>
<td>[7:6]</td>
<td>[5:4]</td>
<td>[3:2]</td>
<td>[1:0]</td>
<td>[11:10]</td>
</tr>
<tr>
<td>140</td>
<td>[7:6]</td>
<td>[5:4]</td>
<td>[3:2]</td>
<td>[1:0]</td>
<td>[9:8]</td>
</tr>
<tr>
<td>150</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[7:6]</td>
</tr>
<tr>
<td>160</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>refresh</td>
<td>[5:4]</td>
</tr>
<tr>
<td>170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3:2]</td>
</tr>
<tr>
<td>180</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1:0]</td>
</tr>
</tbody>
</table>
Memory Access and Clock Frequency

• Nominal **clock frequency** applies to CPU and caches
  • Memory bus has its own clock, typically much slower
  • SDRAM operates on bus clock

• Another reason why processor clock frequency isn’t perfect performance metric
  • Clock frequency increases don’t reduce memory or bus latency
  • May make misses come out faster
    • At some point memory bandwidth may become a **bottleneck**
    • Further increases in (core) clock speed won’t help at all
Error Detection and Correction

- One last thing about DRAM technology: errors
  - DRAM fails at a higher rate than SRAM or CPU logic
    - Capacitor wear
    - Bit flips from energetic $\alpha$-particle strikes
    - Many more bits
  - Modern DRAM systems: built-in error detection/correction

- 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
Error Detection and Correction

- Error detection/correction schemes distinguished by...
  - How many (simultaneous) errors they can detect
  - How many (simultaneous) errors they can correct
Error Detection Example: Parity

- **Parity**: simplest scheme
  - \( f(\text{data}_{N-1 \ldots 0}) = \text{XOR}(\text{data}_{N-1}, \ldots, \text{data}_1, \text{data}_0) \)
  - Single-error detect: detects a single bit flip (common case)
    - Will miss two simultaneous bit flips...
    - But what are the odds of that happening?
    - Zero-error correct: no way to tell which bit flipped

- Many other schemes exist for detecting/correcting errors
  - Take ECE 254 (Fault Tolerant Computing) for more info
Memory Organization

• So data is striped across DRAM chips
• But how is it organized?
  • Block size?
  • Associativity?
  • Replacement policy?
  • Write-back vs. write-thru?
  • Write-allocate vs. write-non-allocate?
  • Write buffer?
  • Optimizations: victim buffer, prefetching, anything else?
**Low $\%_{\text{miss}}$ At All Costs**

- For a memory component: $t_{\text{hit}}$ vs. $\%_{\text{miss}}$ tradeoff

- Upper components (I$, D$) emphasize low $t_{\text{hit}}$
  - Frequent access $\rightarrow$ minimal $t_{\text{hit}}$ important
  - $t_{\text{miss}}$ is not bad $\rightarrow$ minimal $\%_{\text{miss}}$ less important
  - Low capacity/associativity/block-size, write-back or write-through

- Moving down (L2) emphasis turns to $\%_{\text{miss}}$
  - Infrequent access $\rightarrow$ minimal $t_{\text{hit}}$ less important
  - $t_{\text{miss}}$ is bad $\rightarrow$ minimal $\%_{\text{miss}}$ important
  - High capacity/associativity/block size, write-back

- For memory, emphasis entirely on $\%_{\text{miss}}$
  - $t_{\text{miss}}$ is disk access time (measured in ms, not ns)
## Typical Memory Organization Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>I$$/D$</th>
<th>L2</th>
<th>Main Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{hit}$</td>
<td>1-2ns</td>
<td>10ns</td>
<td>30ns</td>
</tr>
<tr>
<td>$t_{miss}$</td>
<td>10ns</td>
<td>30ns</td>
<td>10ms (10M ns)</td>
</tr>
<tr>
<td>Capacity</td>
<td>8–64KB</td>
<td>128KB–2MB</td>
<td>512MB–8GB</td>
</tr>
<tr>
<td>Block size</td>
<td>16–32B</td>
<td>32–256B</td>
<td>8–64KB pages</td>
</tr>
<tr>
<td>Associativity</td>
<td>1–4</td>
<td>4–16</td>
<td>Full</td>
</tr>
<tr>
<td>Replacement Policy</td>
<td>NMRU</td>
<td>NMRU</td>
<td>working set</td>
</tr>
<tr>
<td>Write-through?</td>
<td>Sometimes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Write-allocate?</td>
<td>Sometimes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Write buffer?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Victim buffer?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Prefetching?</td>
<td>Sometimes</td>
<td>Yes</td>
<td>Sometimes</td>
</tr>
</tbody>
</table>

© 2009 Daniel J. Sorin from Roth
One Last Gotcha

- On a 32-bit architecture, there are $2^{32}$ byte addresses
  - Requires 4 GB of memory
  - But not everyone buys machines with 4 GB of memory
  - And what about 64-bit architectures?

- Let’s take a step back...