Today

- SRAM vs. DRAM
- Interleaving/Banking
- DRAM Microarchitecture
  - Memory controller
  - Memory buses
  - Banks, ranks, channels, DIMMs
  - Address mapping: software vs. hardware
  - DRAM refresh
- Memory scheduling policies
- Memory power/energy management
- Multi-core issues
  - Fairness, interference
  - Large DRAM capacity
Readings

- **Recommended:**
Main Memory in the System
Memory Bank Organization

- Read access sequence:
  1. Decode row address & drive word-lines
  2. Selected bits drive bit-lines
     - Entire row read
  3. Amplify row data
  4. Decode column address & select subset of row
     - Send to output
  5. Precharge bit-lines
     - For next access
SRAM (Static Random Access Memory)

Read Sequence
1. address decode
2. drive row select
3. selected bit-cells drive bitlines
   (entire row is read together)
4. diff. sensing and col. select
   (data is ready)
5. precharge all bitlines
   (for next read or write)

Access latency dominated by steps 2 and 3
Cycling time dominated by steps 2, 3 and 5
- step 2 proportional to $2^m$
- step 3 and 5 proportional to $2^n$
**DRAM (Dynamic Random Access Memory)**

Bits stored as charges on node capacitance (non-restorative)
- bit cell loses charge when read
- bit cell loses charge over time

**Read Sequence**
1~3 same as SRAM
4. a “flip-flopping” sense amp amplifies and regenerates the bitline, data bit is mux’ed out
5. precharge all bitlines

**Refresh**: A DRAM controller must periodically read all rows within the allowed refresh time (10s of ms) such that charge is restored in cells
SRAM vs. DRAM

- SRAM is preferable for register files and L1/L2 caches
  - Fast access
  - No refreshes
  - Simpler manufacturing (compatible with logic process)
  - Lower density (6 transistors per cell)
  - Higher cost

- DRAM is preferable for stand-alone memory chips
  - Much higher capacity
  - Higher density
  - Lower cost
Memory subsystem organization

- Memory subsystem organization
  - Channel
  - DIMM
  - Rank
  - Chip
  - Bank
  - Row/Column
Breaking down a DIMM

DIMM (Dual in-line memory module)

Side view

Front of DIMM

Back of DIMM

SIDE

4.00
Breaking down a DIMM

DIMM (Dual in-line memory module)

Side view

Rank 0: collection of 8 chips

Rank 1
## DIMM & Rank (from JEDEC)

### 5 Unbuffered DIMM Details

#### Table 8 — SDRAM Module Configurations (Reference Designs)

<table>
<thead>
<tr>
<th>Raw Card Version</th>
<th>DIMM Capacity</th>
<th>DIMM Organization</th>
<th>SDRAM Density</th>
<th>SDRAM Organization</th>
<th>Number of SDRAMs</th>
<th>Number of Physical Ranks</th>
<th>Number of Banks in SDRAM</th>
<th>Number of Address Bits Row/Column</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>512MB</td>
<td>64 Meg x 64</td>
<td>512 Megabit</td>
<td>64 Meg x 8</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>13/10</td>
</tr>
<tr>
<td></td>
<td>1GB</td>
<td>128 Meg x 64</td>
<td>1 Gigabit</td>
<td>128 Meg x 8</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>14/10</td>
</tr>
<tr>
<td></td>
<td>2GB</td>
<td>256 Meg x 64</td>
<td>2 Gigabit</td>
<td>256 Meg x 8</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>15/10</td>
</tr>
<tr>
<td></td>
<td>4GB</td>
<td>512 Meg x 64</td>
<td>4 Gigabit</td>
<td>512 Meg x 8</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>16/10</td>
</tr>
<tr>
<td></td>
<td>8GB</td>
<td>1 Gig x 64</td>
<td>8 Gigabit</td>
<td>1 Gig x 8</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>16/11</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>1GB</td>
<td>128 Meg x 64</td>
<td>512 Megabit</td>
<td>64 Meg x 8</td>
<td>16</td>
<td>2</td>
<td>8</td>
<td>13/10</td>
</tr>
<tr>
<td></td>
<td>2GB</td>
<td>256 Meg x 64</td>
<td>1 Gigabit</td>
<td>128 Meg x 8</td>
<td>16</td>
<td>2</td>
<td>8</td>
<td>14/10</td>
</tr>
<tr>
<td></td>
<td>4GB</td>
<td>512 Meg x 64</td>
<td>2 Gigabit</td>
<td>256 Meg x 8</td>
<td>16</td>
<td>2</td>
<td>8</td>
<td>15/10</td>
</tr>
<tr>
<td></td>
<td>8GB</td>
<td>1 Gig x 64</td>
<td>4 Gigabit</td>
<td>512 Meg x 8</td>
<td>16</td>
<td>2</td>
<td>8</td>
<td>16/10</td>
</tr>
<tr>
<td></td>
<td>16GB</td>
<td>2 Gig x 64</td>
<td>8 Gigabit</td>
<td>1 Gig x 8</td>
<td>16</td>
<td>2</td>
<td>8</td>
<td>16/11</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>256MB</td>
<td>32 Meg x 64</td>
<td>512 Megabit</td>
<td>32 Meg x 16</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>12/10</td>
</tr>
<tr>
<td></td>
<td>512MB</td>
<td>64 Meg x 64</td>
<td>1 Gigabit</td>
<td>64 Meg x 16</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>13/10</td>
</tr>
<tr>
<td></td>
<td>1GB</td>
<td>128 Meg x 64</td>
<td>2 Gigabit</td>
<td>128 Meg x 16</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>14/10</td>
</tr>
<tr>
<td></td>
<td>2GB</td>
<td>256 Meg x 64</td>
<td>4 Gigabit</td>
<td>256 Meg x 16</td>
<td>4</td>
<td>1</td>
<td>8</td>
<td>15/10</td>
</tr>
</tbody>
</table>
Breaking down a Rank

Rank 0

Chip 0
Chip 1

... Chip 7

Data <0:63>

<0:7>  <8:15>  <56:63>
Breaking down a Chip
Breaking down a Bank
Memory subsystem organization

- Channel
- DIMM
- Rank
- Chip
- Bank
- Row/Column
Example: Transferring a cache block

Physical memory space

0xFFFF…F

0x40

0x00

64B cache block

Mapped to

Channel 0

DIMM 0

Rank 0

Mapped to

Core™ i7
Example: Transferring a cache block

Physical memory space

0xFFFF...F

... 0x40 0x00

64B cache block

Chip 0

Chip 1

Rank 0

Chip 7

<0:7> <8:15> <56:63>

Data <0:63>
Example: Transferring a cache block

Physical memory space

Chip 0

Row 0

Col 0

Chip 1

Rank 0

Chip 7

64B cache block

Data <0:63>

<0:7>

<8:15>

<56:63>
Example: Transferring a cache block

Physical memory space

64B cache block
Example: Transferring a cache block

Physical memory space

- 0xFFFF...F
- 0x40
- 0x00
- 0x00

64B cache block

- Row 0
- Col 1

Chip 0

Chip 1

Chip 7

Rank 0

Data <0:63>

<0:7>

<8:15>

<56:63>
Example: Transferring a cache block

Physical memory space

- 0xFFFF...F
- 0x40
- 0x00

64B cache block

- Row 0
- Col 1

Chip 0

Chip 1

Rank 0

Chip 7

Data <0:63>

<0:7>

<8:15>

<56:63>
Example: Transferring a cache block

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

During the process, 8 columns are read sequentially.
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)

Row address 0 → Row decoder → Columns

Row 1

Row Buffer

CONFLICT!

Column address 85 → Column mux → Data
Latency Components: Basic DRAM Operation

- CPU $\rightarrow$ controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller $\rightarrow$ DRAM transfer time
- DRAM bank latency
  - Simple CAS is row is “open” OR
  - RAS + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)
- DRAM $\rightarrow$ CPU transfer time (through controller)
A DRAM Chip and DIMM

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

- **Multiple chips are put together to form a wide interface**
  - Called a module
  - **DIMM:** Dual Inline Memory Module
  - All chips in one side of a DIMM are operated the same way (rank)
    - Respond to a single command
    - Share address and command buses, but provide different data

- **If we have chips with 8-bit interface, to read 8 bytes in a single access, use 8 chips in a DIMM**
128M x 8-bit DRAM Chip
A 64-bit Wide DIMM
A 64-bit Wide DIMM

**Advantages:**
- Acts like a high-capacity DRAM chip with a wide interface
- Flexibility: memory controller does not need to deal with individual chips

**Disadvantages:**
- Granularity: Accesses cannot be smaller than the interface width
Multiple DIMMs

- **Advantages:**
  - Enables even higher capacity

- **Disadvantages:**
  - Interconnect complexity and energy consumption can be high

![Diagram showing "Mesh Topology" with connections between Single Channel SDRAM Controller and multiple DIMMs.]
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
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)
How Multiple Banks/Channels Help

Before: No Overlapping
Assuming accesses to different DRAM rows

After: Overlapped Accesses
Assuming no bank conflicts
Multiple Channels

- **Advantages**
  - Increased bandwidth
  - Multiple concurrent accesses (if independent channels)

- **Disadvantages**
  - Higher cost than a single channel
    - More board wires
    - More pins (if on-chip memory controller)
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

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

- Accesses to consecutive cache blocks can be serviced in parallel
- How about random accesses? Strided accesses?
Bank Mapping Randomization

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

```plaintext
Column (11 bits) 3 bits ▼ XOR ▼ Bank index (3 bits) ▼ Byte in bus (3 bits)
```
Address Mapping (Multiple Channels)

<table>
<thead>
<tr>
<th>C</th>
<th>Row (14 bits)</th>
<th>Bank (3 bits)</th>
<th>Column (11 bits)</th>
<th>Byte in bus (3 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>C</td>
<td>Bank (3 bits)</td>
<td>Column (11 bits)</td>
</tr>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>Bank (3 bits)</td>
<td>C</td>
<td>Column (11 bits)</td>
</tr>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>Bank (3 bits)</td>
<td>Column (11 bits)</td>
<td>C</td>
</tr>
</tbody>
</table>

- **Where are consecutive cache blocks?**

<table>
<thead>
<tr>
<th>C</th>
<th>Row (14 bits)</th>
<th>High Column</th>
<th>Bank (3 bits)</th>
<th>Low Col.</th>
<th>Byte in bus (3 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>C</td>
<td>High Column</td>
<td>Bank (3 bits)</td>
<td>Low Col.</td>
</tr>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>High Column</td>
<td>C</td>
<td>Bank (3 bits)</td>
<td>Low Col.</td>
</tr>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>High Column</td>
<td>Bank (3 bits)</td>
<td>C</td>
<td>Low Col.</td>
</tr>
<tr>
<td></td>
<td>Row (14 bits)</td>
<td>High Column</td>
<td>Bank (3 bits)</td>
<td>Low Col.</td>
<td>C</td>
</tr>
</tbody>
</table>
Interaction with Virtual → Physical Mapping

- Operating System influences where an address maps to in DRAM

<table>
<thead>
<tr>
<th>Virtual Page number (52 bits)</th>
<th>Page offset (12 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Frame number (19 bits)</td>
<td>Page offset (12 bits)</td>
</tr>
<tr>
<td>Row (14 bits)</td>
<td>Bank (3 bits)</td>
</tr>
</tbody>
</table>

- Operating system can control which bank a virtual page is mapped to. It can randomize Page → <Bank,Channel> mappings

- Application cannot know/determine which bank it is accessing
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?
Purpose and functions

- Ensure correct operation of DRAM (refresh)
- 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 and row-buffer management

- Manage power consumption and thermals in DRAM
  - Turn on/off DRAM chips, manage power modes
DRAM Controller Issues

- 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)
DRAM Controller (II)
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
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?