DRAM CONTROLLER

Mahdi Nazm Bojnordi

Assistant Professor
School of Computing
University of Utah
Overview

- Announcement
  - Homework 5 submission deadline: Apr. 10th

- This lecture
  - DRAM control
  - DRAM timing
  - DRAM hierarchy
    - Channel, bank
Recall: DRAM System

- DRAM chips can perform basic operations
Recall: DRAM Operations

- Main DRAM operations are
  - **Precharge** bitlines to prepare subarray for activating a wordline
  - **Activate** a row by connecting DRAM cells to the bitlines and start sensing
  - **Read** the contents of a data block from the row buffer
  - **Write** new contents for data block into the row buffer
  - **Refresh** DRAM cells
    - can be done through a precharge followed by an activate
DRAM Row Buffer

- All reads and writes are performed through RB

DRAM Cell

DRAM Sense Amp.

Row Access Strobe (RAS)

Data Array

Column Access Strobe (CAS)

Row Buffer (RB)
DRAM Row Buffer

- Row buffer holds a single row of the array
  - A typical DRAM row (page) size is 8KB
- The entire row is moved to row buffer; but only a block is accessed each time

Row buffer access possibilities

- **Row buffer hit:** no need for a precharge or activate
  - ~20ns only for moving data between pins and RB
- **Row buffer miss:** activate (and precharge) are needed
  - ~40ns for an empty row
  - ~60ns for on a row conflict
DRAM Control

- DRAM chips have no intelligence
  - An external controller dictates operations
  - Modern controllers are integrated on CPU
- Basic DRAM timings are
  - $t_{\text{CAS}}$: column access strobe (RD $\rightarrow$ DATA)
  - $t_{\text{RAS}}$: row active strobe (ACT $\rightarrow$ PRE)
  - $t_{\text{RP}}$: row precharge (PRE $\rightarrow$ ACT)
  - $t_{\text{RC}}$: row cycle (ACT $\rightarrow$ PRE $\rightarrow$ ACT)
  - $t_{\text{RCD}}$: row to column delay (ACT $\rightarrow$ RD/WT)
DRAM Timing Example

- **Access time**
  - **Row hit:** \( t_{\text{CAS}} \)
  - **Row empty:** \( t_{\text{RCD}} + t_{\text{CAS}} \)
  - **Row conflict:** \( t_{\text{RP}} + t_{\text{RCD}} + t_{\text{CAS}} \)
Memory Channels

- Memory channels provide fully parallel accesses
  - Separate data, control, and address buses

Not scalable due to pin overhead
Memory Banks

- Memory banks provide parallel operations
  - Shared data, control, and address buses
- The goal is to keep the data bus fully utilized

[Diagram showing data transfer between banks]

Shorter data transfer time to reduce bus conflicts
Double data rate vs. single rate
DRAM Organization

- DRAM channels are independently accessed through dedicated data, address, and command buses
  - Physically broken down into DIMMs (dual in-line memory modules)
  - Logically divided into ranks, which are a collection of DRAM chips responding to the same memory request
Memory Controller

- Memory controller connects CPU and DRAM
- Receives requests after cache misses in LLC
  - Possibly originating from multiple cores
- Complicated piece of hardware, handles:
  - DRAM refresh management
  - Command scheduling
  - Row-buffer management policies
  - Address mapping schemes
DRAM Refresh Management

- DRAM requires the cells’ contents to be read and written periodically
  - **Burst refresh**: refresh all of the cells each time
    - Simple control mechanism
  - **Distributed refresh**: a group of cells are refreshed
    - Avoid blocking memory for a long time
- Recently accessed rows need not to be refreshed
  - **Smart refresh**
Command Scheduling

- Write buffering
  - Writes can wait until reads are done

- Controller queues DRAM commands
  - Usually into per-bank queues
  - Allows easily reordering ops. meant for same bank

- Common policies:
  - First-Come-First-Served (FCFS)
  - First-Ready First-Come-First-Served (FR-FCFS)
Command Scheduling

- First-Come-First-Served
  - Oldest request first

- First-Ready First-Come-First-Served
  - Prioritize column changes over row changes
  - Skip over older conflicting requests
  - Find row hits (on queued requests)
    - Find oldest
    - If no conflicts with in-progress request → good
    - Otherwise (if conflicts), try next oldest
FCFS vs. FR-FCFS

- READ(B0,R0,C0)  READ(B0,R1,C0)  READ(B0,R0,C1)

- FCFS

```
<table>
<thead>
<tr>
<th>Cmd</th>
<th>Addr</th>
<th>ACT</th>
<th>READ</th>
<th>PRE</th>
<th>ACT</th>
<th>READ</th>
<th>PRE</th>
<th>ACT</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>C0</td>
<td></td>
<td></td>
<td></td>
<td>B0</td>
<td>R1</td>
<td></td>
<td>C0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B1</td>
<td>R0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>READ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C1</td>
<td></td>
</tr>
</tbody>
</table>
```
FCFS vs. FR-FCFS

- READ(B0,R0,C0) READ(B0,R1,C0) READ(B0,R0,C1)
  - FCFS
  - FR-FCFS
  - Savings
Row-Buffer Management Policies

- Open-page Policy
  - After access, keep page in DRAM row buffer
  - If access to different page, must close old one first
    - Good if lots of locality

- Close-page Policy
  - After access, immediately close page in DRAM row buffer
  - If access to different page, old one already closed
    - Good if no locality (random access)