Improving DRAM Performance via Variable Refresh Latency

VRL-DRAM

Anup Das
Hasan Hassan and Onur Mutlu

Drexel University

ETH Zürich
Executive Summary

• Observations
  - During refresh, *almost half* of the refresh time is spent in injecting the *last 5% of charge* of a fully charged cell
  - Once *fully* restored, a DRAM cell can sustain multiple *partial* refreshes *without* sacrificing data integrity

• Idea: Variable Refresh Latency DRAM
  - Fully refresh a DRAM cell *only when necessary*, and otherwise issue partial refresh → reduces refresh latency in common case

• Characterization: new detailed circuit-level analytical model
  - Great potential to *lower DRAM refresh timing parameters*

• Performance Evaluation
  - Significant *refresh overhead reduction* (23% for PARSEC-3.0 workloads without errors)
DRAM Refresh Overhead

- DRAM cell loses charge over time
- Needs periodic refresh: once every 64ms
- Power overhead and performance loss due to DRAM refresh

J. Liu et al., RAIDR: Retention-Aware Intelligent DRAM Refresh in ISCA’12
Mitigating DRAM Refresh Overhead

- Not all DRAM cells require the default 64ms refresh rate

- DRAM refresh overhead can be minimized by skipping refresh for cells that can retain data longer than 64ms

  - Liu et al., RAIDR, ISCA’12
During refresh, **almost half** of the refresh time is spent injecting the last 5% of charge of a fully charged cell.
DRAM Refresh Related Key Observations (I)

- During refresh, **almost half** of the refresh time is spent in injecting the last 5% of charge of a fully charged cell

**Idea:** Lower refresh timing parameters to truncate refresh at **95%** of a cell’s capacity (i.e., **partial refresh**).
Once fully restored, a DRAM cell can sustain *multiple partial refreshes* without sacrificing data integrity.
DRAM Refresh Related Key Observations (II)

- Once **fully** restored, a DRAM cell can sustain **multiple partial refreshes** without sacrificing data integrity.
Variable Refresh Latency DRAM

• Key Idea
  - Use **two timing parameters** for DRAM refresh
    - Full (slow) refresh: Issue **only when necessary**
    - Partial (fast) refresh: Issue **when no problem with correctness**

• Two components
  - Characterization
    - **Accurately estimate** the number of partial refreshes that a DRAM cell can reliably sustain
  - Scheduling
    - **Whenever possible**, issue partial refreshes to reduce the refresh overhead
Outline

• Introduction

• DRAM refresh characterization

• Partial and full refresh scheduling

• Evaluation

• Conclusion
DRAM Refresh Characterization (I)

- Existing DRAM circuit simulators do not take into account
  - Data pattern stored in DRAM
  - Sneak paths
  - Bitline/wordline parasitics

- SPICE simulation is time consuming
DRAM Refresh Characterization (I)

- Estimate number of partial refreshes that can be reliably sustained
  - A new detailed circuit-level analytical model for DRAM
    - Details in paper, also available as an open-source tool [https://github.com/anupkdas-nus/VRL-DRAM](https://github.com/anupkdas-nus/VRL-DRAM)

• End result:

<table>
<thead>
<tr>
<th>Row ID</th>
<th>Number of partial refreshes that can be reliably sustained</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>127</td>
<td>5</td>
</tr>
</tbody>
</table>
DRAM Refresh Characterization (II)

- Characterize retention time of DRAM rows using Liu et al., RAIDR, ISCA’12
- End result:

<table>
<thead>
<tr>
<th>Row ID</th>
<th>Retention time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>127</td>
<td>128</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• DRAM refresh characterization
• Partial and full refresh scheduling
• Evaluation
• Conclusion
Partial Refresh Scheduling (I)

- Key Idea (I): **VRL**
  - Row 0
    - Partial Refresh Interval = 3
    - Retention time = 256
Partial Refresh Scheduling (I)

- Key Idea (1): 
  - VRL
    - Row 0
      - Partial Refresh Interval = 3
      - Retention time = 256

Refresh Overhead Reduced

<table>
<thead>
<tr>
<th>VRL</th>
<th>RAIDR</th>
<th>time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>full</td>
<td>0.</td>
</tr>
<tr>
<td>partial</td>
<td>full</td>
<td>256.</td>
</tr>
<tr>
<td>partial</td>
<td>full</td>
<td>512.</td>
</tr>
<tr>
<td>partial</td>
<td>full</td>
<td>768.</td>
</tr>
<tr>
<td>full</td>
<td>full</td>
<td>1024.</td>
</tr>
<tr>
<td>partial</td>
<td>full</td>
<td>1280.</td>
</tr>
</tbody>
</table>
Partial Refresh Scheduling (II)

- Key Idea (II): **VRL-Access**
  - Use partial refresh in place of full refresh if there is a memory read/write access
    - DRAM activation caused by a read or a write access **fully restores** the charge in the DRAM row
Partial Refresh Scheduling (II)

- Key Idea (II): **VRL-Access**
  - Use partial refresh in place of full refresh if there is a memory
    read/write access
  - DRAM activation caused by a read or a write access fully
    restores the charge in the DRAM row

### Graph

- **Memory access**
  - **VRL-Access**
    - full
    - partial
    - partial
    - partial
    - partial
    - partial
  - **VRL**
    - full
    - partial
    - partial
    - partial
    - partial
    - full
  - **RAIDR**
    - full
    - full
    - full
    - full
    - full
    - full

### Time (ms)

- 0.
- 256.
- 512.
- 768.
- 1024.
- 1280.

**Refresh Overhead Reduced**
Outline

• Introduction
• DRAM refresh characterization
• Partial and full refresh scheduling
• Evaluation
• Conclusion
Evaluation Methodology

• PARSEC-3.0 workloads and a server workload \textit{bgsave}

• 90nm technology node for the refresh parameters

• Refresh overhead computed as the fraction of time in every refresh interval that a DRAM is \textit{unavailable} to service any access request

• Simulate workloads to compute the \textit{average refresh overhead} for all memory requests
VRL and VRL-Access
Reduce Refresh Overhead

RAIDR  VRL  VRL-Access

Normalized Refresh Overhead

gcc  h264-decode  ip2-decode  ip2-encode  mcf  soplex  sphinx3  stream-copy  bzip2  bg save  AVG
Refresh overhead of VRL is on average 23% lower and VRL-Access is on average 34% lower than RAIDR.
VRL-DRAM Area Overhead

- 90nm technology node from Microwind3

<table>
<thead>
<tr>
<th>nbits</th>
<th>Logic area ($\mu m^2$)</th>
<th>% DRAM bank area ($\mu m^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>105</td>
<td>0.97%</td>
</tr>
<tr>
<td>3</td>
<td>152</td>
<td>1.4%</td>
</tr>
<tr>
<td>4</td>
<td>200</td>
<td>1.85%</td>
</tr>
</tbody>
</table>

- nbits is the number of bits required to store the number of partial refreshes for a row

- Area overhead < 2% of DRAM bank area
Accuracy of VRL-DRAM tool

- Detailed circuit-level analytical model
- Tool: https://github.com/anupkedas-nus/VRL-DRAM
Accuracy of VRL-DRAM tool

- Detailed circuit-level analytical model
- Tool: [https://github.com/anupkdas-nus/VRL-DRAM](https://github.com/anupkdas-nus/VRL-DRAM)

**Accuracy** of the detailed analytical model of VRL-DRAM is very close to SPICE simulation
Outline

- Introduction
- DRAM refresh characterization
- Partial and full refresh scheduling
- Evaluation
- Conclusion
Conclusion

• Observations
  - During refresh, almost half of the refresh time is spent in injecting the last 5% of charge of a fully charged cell
  - Once fully restored, a DRAM cell can sustain multiple partial refreshes without sacrificing data integrity

• Idea: Variable Refresh Latency DRAM
  - Fully refresh a DRAM cell only when necessary, and otherwise issue partial refresh reduces refresh latency in common case

• Characterization: new detailed circuit-level analytical model
  - Great potential to lower DRAM refresh timing parameters

• Performance Evaluation
  - Significant refresh overhead reduction (23% for PARSEC-3.0 workloads without errors)
Improving DRAM Performance via Variable Refresh Latency

VRL-DRAM

Anup Das
Hasan Hassan and Onur Mutlu