Computer Architecture
Lecture 6b: SoftMC

Hasan Ibrahim Hasan
ETH Zürich
Fall 2018
4 October 2018
SoftMC
A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan,
Nandita Vijaykumar, Samira Khan,
Saugata Ghose, Kevin Chang,
Gennady Pekhimenko, Donghyuk Lee,
Oguz Ergin, Onur Mutlu
Executive Summary

• Two critical problems of DRAM: Reliability and Performance
• Characterize, analyze, and understand DRAM cell behavior
• We design and implement SoftMC, an FPGA-based DRAM testing infrastructure
  – Flexible and Easy to Use (C++ API)
  – Open-source (github.com/CMU-SAFARI/SoftMC)
• We implement two use cases
  – A retention time distribution test
  – An experiment to validate two latency reduction mechanisms
• SoftMC enables a wide range of studies
Outline

1. DRAM Basics & Motivation
2. SoftMC
3. Use Cases
   - Retention Time Distribution Study
   - Evaluating Recently-Proposed Ideas
4. Future Research Directions
5. Conclusion
DRAM Operations

Memory Bus

Precharge

CPU

DRAM Row

DRAM Cell

Sense Amplifier
DRAM Latency

- **Activate**
- **Read**
- **Precharge**
- **Activate**

Retention Time: The interval during which the data is retained correctly in the DRAM cell without accessing it
Latency vs. Reliability

Violating latencies negatively affects DRAM reliability
Other Factors Affecting Reliability and Latency

• Temperature
• Voltage
• Inter-cell Interference

To develop new mechanisms improving **reliability** and **latency**, we need to better understand the effects of these factors.
Characterizing DRAM

Many of the factors affecting DRAM reliability and latency cannot be properly modeled.

We need to perform experimental studies of real DRAM chips.
Outline
1. DRAM Basics & Motivation
2. SoftMC
3. Use Cases
   – Retention Time Distribution Study
   – Evaluating Recently-Proposed Ideas
4. Future Research Directions
5. Conclusion
Goals of a DRAM Testing Infrastructure

● Flexibility
  - Ability to test *any* DRAM operation
  - Ability to test *any combination* of DRAM operations and *custom* timing parameters

● Ease of use
  - *Simple* programming interface (C++)
  - *Minimal* programming effort and time
  - *Accessible* to a wide range of users
    - *who may lack experience in hardware design*
SoftMC: High-level View

FPGA-based memory characterization infrastructure

Prototype using *Xilinx* ML605

Easily programmable using the C++ API
SoftMC: Key Components

1. SoftMC API
2. PCIe Driver
3. SoftMC Hardware
InstructionSequence iseq;

iseq.insert(genACT(bank, row));
iseq.insert(genWAIT(tRCD));
iseq.insert(genWR(bank, col, data));
iseq.insert(genWAIT(tCL + tBL + tWR));
iseq.insert(genPRE(bank));
iseq.insert(genWAIT(tRP));
iseq.insert(genEND());
iseq.execute(fpga);
SoftMC: Key Components

1. SoftMC API

2. PCIe Driver*
   Communicates raw data with the FPGA

3. SoftMC Hardware

SoftMC Hardware

PCIe Controller

Instruction Receiver
- Instruction Queue
- Auto-refresh Controller
- Calibration Controller

Instruction Dispatch

Read Capture

SoftMC Hardware (FPGA)

Wait (Ready to Access Latency)

Host Machine

DRAM

Data

Instructions

SAFARI
Outline

1. DRAM Basics & Motivation
2. SoftMC
3. Use Cases
   - Retention Time Distribution Study
   - Evaluating Recently-Proposed Ideas
4. Future Research Directions
5. Conclusion
Retention Time Distribution Study

```java
1 InstructionSequence iseq;
2 iseq.insert(genACT(bank, row));
3 iseq.insert(genWAIT(tRCD));
4 for(int col = 0; col < COLUMNS; col++){
5   iseq.insert(genWR(bank, col, data));
6   iseq.insert(genWAIT(tBL));
7 }
8 iseq.insert(genWAIT(tCL + tWR));
9 iseq.insert(genPRE(bank));
10 iseq.insert(genWAIT(tRP));
11 iseq.insert(genEND());
12 iseq.execute(fpga));
```

Can be implemented with just ~100 lines of code
Retention Time Test: Results

Validates the correctness of the SoftMC Infrastructure

@ ~20°C (room temperature)
Outline

1. DRAM Basics & Motivation

2. SoftMC

3. Use Cases
   - Retention Time Distribution Study
   - Evaluating Recently-Proposed Ideas

4. Future Research Directions

5. Conclusion
Accessing Highly-charged Cells Faster

**NUAT**  
*(Shin+, HPCA 2014)*

**ChargeCache**  
*(Hassan+, HPCA 2016)*

A *highly-charged* cell can be accessed with *low latency*
How a Highly-Charged Cell Is Accessed Faster?

- **Activate DRAM Cell**
- **Sense Amplifier**

**Latency**
- **Ready-to-access Latency**
- **Activation Latency**

**Time**
- 0 (refresh) 64 ms
Ready-to-access Latency Test

- Longer wait → Lower cell charge
- Shorter wait → Higher cell charge

Write Reference Data → Wait for the Wait Interval → Read Back → Observe Errors

With custom ready-to-access latency parameter

Can be implemented with just ~150 lines of code
Ready-to-access Latency: Results

Expected Curves

Latency (cycles)

- 6
- 5
- 4
- 3

Number of Erroneous Bytes

Refresh Interval

We do not observe the expected latency reduction effect in existing DRAM chips.
Why Don’t We See the Latency Reduction Effect?

- The memory controller cannot externally control when a sense amplifier gets enabled in existing DRAM chips.

---

**Diagram Details:**
- **Cell**
- **Sense Amp**
- **Charge Level**
- **Ready to Access**
- **ACT**
- **R/W**
- **Data 0**
- **Data 1**
- **Potential Reduction**
- **Fixed Latency!**
- **Enabling the Sense Amplifier**
- **Ready to Access Charge Level**
Outline

1. DRAM Basics & Motivation
2. SoftMC
3. Use Cases
   - Retention Time Distribution Study
   - Evaluating Recently-Proposed Ideas
4. Future Research Directions
5. Conclusion
Future Research Directions

• More Characterization of DRAM
  – How are the cell characteristics changing with different generations of technology nodes?
  – What types of usage accelerate aging?

• Characterization of Non-volatile Memory

• Extensions
  – Memory Scheduling
  – Workload Analysis
  – Testbed for in-memory Computation
Outline

1. DRAM Basics & Motivation
2. SoftMC
3. Use Cases
   - Retention Time Distribution Study
   - Evaluating Recently-Proposed Ideas
4. Future Research Directions
5. Conclusion
Conclusion

• **SoftMC**: First publicly-available FPGA-based DRAM testing infrastructure

• **Flexible and Easy to Use**

• Implemented two use cases
  – Retention Time Distribution Study
  – Evaluation of two recently-proposed latency reduction mechanisms

• SoftMC can enable many other studies, ideas, and methodologies in the design of future memory systems

• **Download** our prototype  
  [github.com/CMU-SAFARI/SoftMC](https://github.com/CMU-SAFARI/SoftMC)
SoftMC
A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan,
Nandita Vijaykumar, Samira Khan,
Saugata Ghose, Kevin Chang,
Gennady Pekhimenko, Donghyuk Lee,
Oguz Ergin, Onur Mutlu

ETH Zürich
Carnegie Mellon

Safari
Backup Slides
# Key SoftMC Instructions

<table>
<thead>
<tr>
<th>InstrType</th>
<th>DDR (4)</th>
<th>UNUSED (3)</th>
<th>CKE, CS (2), RAS, CAS, WE (6)</th>
<th>Bank (3)</th>
<th>Addr (16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAIT (4)</td>
<td></td>
<td></td>
<td>cycles (28)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUSDIR (4)</td>
<td></td>
<td>UNUSED (27)</td>
<td></td>
<td></td>
<td>dir (1)</td>
</tr>
<tr>
<td>END (4)</td>
<td></td>
<td>UNUSED (28)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SoftMC @ Github

No description, website, or topics provided.

Edit

3 commits
1 branch
0 releases
1 contributor

Branch: master
New pull request

Latest commit 87ef1 an hour ago

<table>
<thead>
<tr>
<th>File</th>
<th>Latest commit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw/boards/ML605</td>
<td>initial commit;</td>
<td></td>
</tr>
<tr>
<td>prebuilt</td>
<td>initial commit;</td>
<td></td>
</tr>
<tr>
<td>sw</td>
<td>initial commit;</td>
<td></td>
</tr>
<tr>
<td>LICENSE</td>
<td>initial commit;</td>
<td></td>
</tr>
<tr>
<td>README.md</td>
<td>fix readme;</td>
<td></td>
</tr>
</tbody>
</table>

SoftMC v1.0

SoftMC is an experimental FPGA-based memory controller design that could be used to develop tests for DDR3 SODIMMs. SoftMC currently supports only the Xilinx ML605 board. Soon, we will port SoftMC on other popularly used boards (e.g., Xilinx VC709).
Activation Latency Test

With **low** activation latency parameter

- **Write Reference Data**
- **Wait for the Wait Interval**
- **ACT-PRE**
- **Change the wait interval**
- **Observe Errors**
- **Read Back**
- **Wait for the Wait Interval**
Activation Latency Test Results

**Module A**
- Number of Erroneous Bytes vs. Wait Interval (ms)
- Latency (cycles)
- Data lines for 14, 11, 8, 5, and 2

**Module B**
- Number of Erroneous Bytes vs. Wait Interval (ms)
- Latency (cycles)
- Data lines for 14, 11, 8, 5, and 2

**Module C**
- Number of Erroneous Bytes vs. Wait Interval (ms)
- Latency (cycles)
- Data lines for 14, 11, 8, 5, and 2