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
  – Recently-discovered bug: RowHammer

• 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

- CPU
- Memory Bus
- Precharge
- CPU
- DRAM Cell
- DRAM Row
- Sense Amplifier
DRAM Latency

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
To develop new mechanisms improving \textbf{reliability} and \textbf{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
SoftMC API

Writing data to DRAM:

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
- Read Capture

Instruction Dispatcher

DDR PHY

Host Machine

DRAM

Wait (Read Access Latency)

Instructions

Activate

Read

Data
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

Can be implemented with just ~100 lines of code

```java
InstructionSequence iseq;
iseq.insert(genACT(bank, row));
iseq.insert(genWAIT(tRCD));
for(int col = 0; col < COLUMNS; col++){
  iseq.insert(genWR(bank, col, data));
  iseq.insert(genWAIT(tBL));
}
iseq.insert(genWAIT(tCL + tWR));
iseq.insert(genPRE(bank));
iseq.insert(genWAIT(tRP));
iseq.insert(genEND());
iseq.execute(fpga));
```
Retention Time Test: Results

@ ~20°C (room temperature)

Validates the correctness of the SoftMC Infrastructure
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**
- **Read**
- **Precharge**
- **Activate**

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

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 → Change the Wait Interval

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)

Number of Erroneous Bytes

Refresh Interval

We do not observe the expected latency reduction effect in existing DRAM chips.

Real Curves

Expected Curves
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 showing the relationship between charge, time, and access to data.](Image)

- **Ready to Access**
- **Sense Amplifier**
- **Potential Reduction**
- **Fixed Latency**
- **Data 0**
- **Data 1**
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 first prototype

[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
Backup Slides
# Key SoftMC Instructions

<table>
<thead>
<tr>
<th>InstrType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR (4)</td>
<td>unused (3)</td>
</tr>
<tr>
<td>WAIT (4)</td>
<td>cycles (28)</td>
</tr>
<tr>
<td>BUSDIR (4)</td>
<td>unused (27)</td>
</tr>
<tr>
<td>END (4)</td>
<td>unused (28)</td>
</tr>
</tbody>
</table>
SoftMC @ Github

No description, website, or topics provided.

3 commits
1 branch
0 releases
1 contributor

Clone or download

arthasSin fix readme:

- hw/boards/ML605
- prebuilt
- sw
- LICENSE
- README.md

Latest commit a75ef1 an hour ago

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).
Ready-to-Access Latency Test Results

**Module A**

- Latency (cycles) vs. Number of Erroneous Bytes
- Wait Interval (ms) ranges from 8 to 456

**Module B**

- Latency (cycles) vs. Number of Erroneous Bytes
- Wait Interval (ms) ranges from 8 to 488

**Module C**

- Latency (cycles) vs. Number of Erroneous Bytes
- Wait Interval (ms) ranges from 8 to 488
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

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) vs. Wait Interval (ms)

**Module B**
- Number of Erroneous Bytes vs. Wait Interval (ms)
- Latency (cycles) vs. Wait Interval (ms)

**Module C**
- Number of Erroneous Bytes vs. Wait Interval (ms)
- Latency (cycles) vs. Wait Interval (ms)