Memory Systems
and Memory-Centric Computing Systems

Lecture 5: Low-Latency Memory I

Prof. Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

18 June 2019
TU Wien Fast Course 2019
Four Key Directions

- Fundamentally Secure/Reliable/Safe Architectures
- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures
- Fundamentally Low-Latency Architectures
- Architectures for Genomics, Medicine, Health
Maslow’s Hierarchy of Needs, A Third Time


Source: https://www.simplypsychology.org/maslow.html
Challenge and Opportunity for Future

Fundamentally Low-Latency Computing Architectures
Memory Latency: Fundamental Tradeoffs
Review: Memory Latency Lags Behind

Memory latency remains almost constant
DRAM Latency Is Critical for Performance

In-memory Databases
[Mao+, EuroSys’12; Clapp+ (Intel), IISWC’15]

Graph/Tree Processing
[Xu+, IISWC’12; Umuroglu+, FPL’15]

In-Memory Data Analytics
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
DRAM Latency Is Critical for Performance

In-memory Databases

Graph/Tree Processing

Long memory latency $\rightarrow$ performance bottleneck

In-Memory Data Analytics
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
The Memory Latency Problem

- High memory latency is a significant **limiter of system performance and energy-efficiency**

- It is becoming increasingly so with **higher memory contention** in multi-core and heterogeneous architectures
  - Exacerbating the bandwidth need
  - Exacerbating the QoS problem

- It increases **processor design complexity** due to the mechanisms incorporated to tolerate memory latency
Retrospective: Conventional Latency Tolerance Techniques

- Caching [initially by Wilkes, 1965]
  - Widely used, simple, effective, but inefficient, passive
  - Not all applications/phases exhibit temporal or spatial locality

- Prefetching [initially in IBM 360/91, 1967]
  - Works well for regular memory access patterns
  - Prefetching irregular access patterns is difficult, inaccurate, and hardware-intensive

- Multithreading [initially in CDC 6600, 1964]
  - Works well if there are multiple threads
  - Improving single thread performance using multithreading hardware is an ongoing research effort

- Out-of-order execution [initially by Tomasulo, 1967]
  - Tolerates cache misses that cannot be prefetched
  - Requires extensive hardware resources for tolerating long latencies

None of These
Fundamentally Reduce
Memory Latency

ongoing research effort
Runahead Execution
Perfect Caches:
Load 1 Hit
Load 2 Hit

Small OoO Instruction Window:
Load 1 Miss
Load 2 Miss

Runahead:
Load 1 Miss
Load 2 Miss
Load 1 Hit
Load 2 Hit

Runahead Execution Example
Saved Cycles
Effect of Runahead in Sun ROCK

- Shailender Chaudhry talk, Aug 2008.
More on Runahead Execution

- Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"


---

Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors

Onur Mutlu §  Jared Stark †  Chris Wilkerson ‡  Yale N. Patt §

§ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

†Microprocessor Research
Intel Labs
jared.w.stark@intel.com

‡Desktop Platforms Group
Intel Corporation
chris.wilkerson@intel.com
More on Runahead Execution (Short)

Runahead Readings

- **Required**

- **Recommended**
Retrospective: Conventional Latency Tolerance Techniques

- Caching [initially by Wilkes, 1965]
  - Widely used, simple, effective, but inefficient, passive
  - Not all applications/phases exhibit temporal or spatial locality

- Prefetching [initially in IBM 360/91, 1967]
  - Works well for regular memory access patterns
  - Prefetching irregular access patterns is difficult, inaccurate, and hardware-intensive

- Multithreading [initially in CDC 6600, 1964]
  - Works well if there are multiple threads
  - Improving single thread performance using multithreading hardware is an ongoing research effort

- Out-of-order execution [initially by Tomasulo, 1967]
  - Tolerates cache misses that cannot be prefetched
  - Requires extensive hardware resources for tolerating long latencies

None of These Fundamentally Reduce Memory Latency
Two Major Sources of Latency Inefficiency

- Modern DRAM is not designed for low latency
  - Main focus is cost-per-bit (capacity)

- Modern DRAM latency is determined by worst case conditions and worst case devices
  - Much of memory latency is unnecessary

**Our Goal:** Reduce Memory Latency at the Source of the Problem
What Causes the Long Memory Latency?
Why the Long Memory Latency?

- **Reason 1: Design of DRAM Micro-architecture**
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2: “One size fits all” approach to latency specification**
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips
  - Same latency parameters for all parts of a DRAM chip
  - Same latency parameters for all supply voltage levels
  - Same latency parameters for all application data
  - ...
Brief Review:
Inside A DRAM Chip
DRAM Module and Chip
Goals

• Cost
• Latency
• Bandwidth
• Parallelism
• Power
• Energy
• Reliability
• ...

...
DRAM Chip
Sense Amplifier

**Enable**

**Top**

**Bottom**

**Inverter**
Sense Amplifier – Two Stable States

Logical “1”

Logical “0”
Sense Amplifier Operation

\[ V_T > V_B \]
DRAM Cell – Capacitor

Empty State

Logical “0”

1. Small – Cannot drive circuits

2. Reading destroys the state

Fully Charged State

Logical “1”
Capacitor to Sense Amplifier
DRAM Subarray – Building Block for DRAM Chip

Row Decoder

Cell Array

Array of Sense Amplifiers (Row Buffer) 8Kb

Cell Array
DRAM Bank

Row Decoder
- Array of Sense Amplifiers (8Kb)
- Cell Array

Row Decoder
- Cell Array
- Array of Sense Amplifiers
- Cell Array

Bank I/O (64b)

Address

Data
DRAM Chip

Shared internal bus

Memory channel - 8bits
DRAM Operation

1. ACTIVATE Row
2. READ/WRITE Column
3. PRECHARGE

Row Address

Row Decoder

Array of Sense Amplifiers

Cell Array

Bank I/O

Column Address

Data
More on DRAM Operation: Section 2


---

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri  
Microsoft Research India  
visesha@microsoft.com

Onur Mutlu  
ETH Zürich  
onur.mutlu@inf.ethz.ch
Why the Long Memory Latency?

- **Reason 1: Design of DRAM Micro-architecture**
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2: “One size fits all” approach to latency specification**
  - Same latency parameters for all *temperatures*
  - Same latency parameters for all *DRAM chips*
  - Same latency parameters for all *parts of a DRAM chip*
  - Same latency parameters for all *supply voltage levels*
  - Same latency parameters for all *application data*
  - ...
Tiered Latency DRAM
What Causes the Long Latency?

**DRAM Chip**

![Diagram showing subarray, I/O, and channel with latency calculation](image)

\[
\text{DRAM Latency} = \text{Subarray Latency} + \text{I/O Latency}
\]

Dominant

[38]
Why is the Subarray So Slow?

- Long bitline
  - Amortizes sense amplifier cost $\rightarrow$ Small area
  - Large bitline capacitance $\rightarrow$ High latency & power
Trade-Off: Area (Die Size) vs. Latency

Long Bitline

Short Bitline

Faster

Smaller

Trade-Off: Area vs. Latency
Trade-Off: Area (Die Size) vs. Latency

- Commodity DRAM
  - Long Bitline
- Fancy DRAM
  - Short Bitline

GOAL

- 512 cells/bitline

Normalized DRAM Area vs. Latency (ns)

Cheaper vs. Faster
Approximating the Best of Both Worlds

**Long Bitline**
- Small Area
- High Latency

**Our Proposal**
- Add Isolation Transistors
- Short Bitline

**Short Bitline**
- Large Area
- Low Latency

**Need Isolation**
Approximating the Best of Both Worlds

<table>
<thead>
<tr>
<th>Long Bitline</th>
<th>Tiered-Latency DRAM</th>
<th>Short Bitline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Area</td>
<td>Small Area</td>
<td>Large Area</td>
</tr>
<tr>
<td>High Latency</td>
<td>Low Latency</td>
<td>Low Latency</td>
</tr>
</tbody>
</table>

Small area using long bitline

Low Latency
Commodity DRAM vs. TL-DRAM [HPCA 2013]

- DRAM Latency ($t_{RC}$) • DRAM Power

- DRAM Area Overhead

$\sim 3\%$: mainly due to the isolation transistors
Trade-Off: Area (Die-Area) vs. Latency

- Cheaper
- normalized DRAM area
- cheaper vs. latency
- goal
- near segment
- far segment
- 512 cells/bitline

- 32
- 64
- 128
- 256

GOAL

Faster
Leveraging Tiered-Latency DRAM

- TL-DRAM is a **substrate** that can be leveraged by the hardware and/or software.

- Many potential uses:
  1. Use near segment as hardware-managed *inclusive* cache to far segment
  2. Use near segment as hardware-managed *exclusive* cache to far segment
  3. Profile-based page mapping by operating system
  4. Simply replace DRAM with TL-DRAM

Near Segment as Hardware-Managed Cache

- **Challenge 1:** How to efficiently migrate a row between segments?
- **Challenge 2:** How to efficiently manage the cache?
Inter-Segment Migration

• **Goal:** Migrate source row into destination row
• **Naïve way:** Memory controller reads the source row byte by byte and writes to destination row byte by byte → High latency
Inter-Segment Migration

• Our way:
  – Source and destination cells *share bitlines*
  – Transfer data from source to destination across *shared bitlines* concurrently

![Diagram of inter-segment migration]

- Far Segment
- Isolation Transistor
- Near Segment
- Sense Amplifier
Inter-Segment Migration

- Our way:
  - Source and destination cells *share bitlines*
  - Transfer data from source to destination
    - *shared bitlines* concurrently

Migration is overlapped with source row access
Additional ~4ns over row access latency

**Step 1:** Activate source row

**Step 2:** Activate destination row to connect cell and bitline

Near Segment

Sense Amplifier
Near Segment as Hardware-Managed Cache

- **Challenge 1**: How to efficiently migrate a row between segments?
- **Challenge 2**: How to efficiently manage the cache?
Using near segment as a cache improves performance and reduces power consumption

Single-Core: Varying Near Segment Length

By adjusting the near segment length, we can trade off cache capacity for cache latency.
More on TL-DRAM

LISA: Low-Cost Inter-Linked Subarrays
[HPCA 2016]
Problem: Inefficient Bulk Data Movement

Bulk data movement is a key operation in many applications

– `memmove` & `memcpy`: 5% cycles in Google's datacenter [Kanev+ ISCA'15]

---

**Channel**

64 bits

**Long latency and high energy**
Goal: Provide a new substrate to enable wide connectivity between subarrays.
Key Idea and Applications

• Low-cost Inter-linked subarrays (LISA)
  – Fast bulk data movement between subarrays
  – Wide datapath via isolation transistors: 0.8% DRAM chip area
  
  ![Diagram of Subarray 1 and Subarray 2 with arrows indicating data movement]

• LISA is a versatile substrate → new applications
  Fast bulk data copy: Copy latency 1.363ms→0.148ms (9.2x)
  → 66% speedup, -55% DRAM energy

  In-DRAM caching: Hot data access latency 48.7ns→21.5ns (2.2x)
  → 5% speedup

  Fast precharge: Precharge latency 13.1ns→5.0ns (2.6x)
  → 8% speedup
New DRAM Command to Use LISA

Row Buffer Movement (RBM): Move a row of data in an activated row buffer to a precharged one

RBM transfers an entire row b/w subarrays
RBM Analysis

• The range of RBM depends on the DRAM design
  – Multiple RBMs to move data across > 3 subarrays

![Diagram of subarrays](image)

• Validated with SPICE using worst-case cells
  – NCSU FreePDK 45nm library

• 4KB data in 8ns (w/ 60% guardband)
  → 500 GB/s, 26x bandwidth of a DDR4-2400 channel

• 0.8% DRAM chip area overhead [O+ ISCA’14]
1. Rapid Inter-Subarray Copying (RISC)

- **Goal:** Efficiently copy a row across subarrays
- **Key idea:** Use RBM to form a new command sequence

1. **Activate** src row
2. **RBM** SA1 → SA2

Reduces row-copy latency by 9.2x, DRAM energy by 48.1x
2. Variable Latency DRAM (VILLA)

- **Goal**: Reduce DRAM latency with low area overhead
- **Motivation**: Trade-off between area and latency

**Long Bitline (DDRx)**

**Short Bitline (RLDRAM)**

Shorter bitlines $\rightarrow$ faster activate and precharge time

High area overhead: >40%
2. Variable Latency DRAM (VILLA)

- **Key idea:** Reduce access latency of hot data via a heterogeneous DRAM design [Lee+ HPCA’13, Son+ ISCA’13]
- **VILLA:** Add fast subarrays as a cache in each bank

**Challenge:** VILLA cache requires frequent movement of data rows

Reduces hot data access latency by 2.2x at only 1.6% area overhead
3. Linked Precharge (LIP)

- **Problem**: The precharge time is limited by the strength of one precharge unit
- **Linked Precharge (LIP)**: LISA precharges a subarray using multiple precharge units

Reduces precharge latency by 2.6x (43% guardband)
More on LISA

Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"


[Slides (pptx) (pdf)]
[Source Code]
CROW: The Copy Row Substrate
[ISCA 2019]
Challenges of DRAM Scaling

1. access latency
2. refresh overhead
3. exposure to vulnerabilities
Conventional DRAM
Copy Row DRAM (CROW)

Multiple row activation
Use Cases of CROW

- **CROW-cache**
  - reduces *access latency*

- **CROW-ref**
  - reduces DRAM *refresh overhead*

- A mechanism for protecting against *RowHammer*
Key Results

**CROW-cache + CROW-ref**
- 20% speedup
- 22% less DRAM energy

**Hardware Overhead**
- 0.5% DRAM chip area
- 1.6% DRAM capacity
- 11.3 KiB memory controller storage
More on CROW

Hasan Hassan, Minesh Patel, Jeremie S. Kim, A. Giray Yaglikci, Nandita Vijaykumar, Nika Mansourighiasi, Saugata Ghose, and Onur Mutlu,

"CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability"
SALP: Reducing DRAM Bank Conflict Impact

Kim, Seshadri, Lee, Liu, Mutlu
A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM
ISCA 2012.
Problem: Bank conflicts are costly for performance and energy
- serialized requests, wasted energy (thrashing of row buffer, busy wait)

Goal: Reduce bank conflicts without adding more banks (low cost)

Observation 1: A DRAM bank is divided into subarrays and each subarray has its own local row buffer
SALP: Key Ideas

- Observation 2: **Subarrays are mostly independent**
  - Except when sharing global structures to reduce cost

---

**Key Idea of SALP:** Minimally reduce sharing of global structures

- Reduce the sharing of ...
  - Global decoder \(\rightarrow\) Enables almost parallel access to subarrays
  - Global row buffer \(\rightarrow\) Utilizes multiple local row buffers
Instead of a global latch, have *per-subarray latches*.
SALP: Reduce Sharing of Global Row-Buffer

Selectively connect local row-buffers to global row-buffer using a **Designated** single-bit latch.
SALP: Baseline Bank Organization

Global Decoder

Latch

Global bitlines

Local row-buffer

Global row-buffer
SALP: Proposed Bank Organization

Global Decoder

Overhead of SALP in DRAM chip: 0.15%
1. Global latch → per-subarray local latches
2. Designated bit latches and wire to selectively enable a subarray
SALP: Results

- Wide variety of systems with different #channels, banks, ranks, subarrays
- Server, streaming, random-access, SPEC workloads

- **Dynamic DRAM energy reduction:** 19%
  - DRAM row hit rate improvement: 13%

- **System performance improvement:** 17%
  - Within 3% of ideal (all independent banks)

- **DRAM die area overhead:** 0.15%
  - vs. 36% overhead of independent banks
More on SALP

- Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu,
  "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM"
  Proceedings of the 39th International Symposium on Computer Architecture (ISCA), Portland, OR, June 2012. Slides (pptx)
More on SALP

**DRAM Process Scaling Challenges**

- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng, **John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

*Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel*
More on SALP

Sub-array Level Parallelism with tWR Relaxation

- **tWR relaxation**
  - Relaxing tWR results in DRAM yield improvement but can degrade performance requiring new compensating features
  - By increasing tWR 5X (from 15ns to 75ns), fail bit counts are expected to reduce by 1 to 2 orders of magnitudes

- **Sub-array level parallelism (SALP)**
  - Allows a page in another sub-array in the same bank to be opened in parallel with the currently activated sub-array
  - Results in performance gain by increasing the row access parallelism within a bank
    ⇒ Used to compensate for the performance loss caused by tWR relaxation

![Diagram showing sub-array level parallelism with tWR relaxation](http://www.cs.utah.edu/thememoryforum/kang_slides.pdf)
More on SALP

Performance Impact of SALP and tWR relaxation

- Performance simulations run for various workloads when tWR is relaxed by 2X and 3X, and when SALP is applied with 2 sub-banks

- Results show that performance is reduced by ~5% and ~2% in average if tWR is relaxed by 3X and 2X, respectively

- Results also show that performance is compensated, and even improved to up to ~3% in average when SALP is applied, even with tWR relaxed by 3X

Why the Long Memory Latency?

- **Reason 1:** Design of DRAM Micro-architecture
  - Goal: Maximize capacity/area, not minimize latency

- **Reason 2:** “One size fits all” approach to latency specification
  - Same latency parameters for all temperatures
  - Same latency parameters for all DRAM chips
  - Same latency parameters for all parts of a DRAM chip
  - Same latency parameters for all supply voltage levels
  - Same latency parameters for all application data
  - ...

SAFARI
Tackling the Fixed Latency Mindset

- Reliable operation latency is actually very heterogeneous
  - Across temperatures, chips, parts of a chip, voltage levels, ...

- Idea: **Dynamically find out and use the lowest latency one can reliably access a memory location with**
  - Adaptive-Latency DRAM [HPCA 2015]
  - Flexible-Latency DRAM [SIGMETRICS 2016]
  - Design-Induced Variation-Aware DRAM [SIGMETRICS 2017]
  - Voltron [SIGMETRICS 2017]
  - DRAM Latency PUF [HPCA 2018]
  - Solar DRAM [ICCD 2018]
  - DRAM Latency True Random Number Generator [HPCA 2019]
  - ...

- We would like to find sources of latency heterogeneity and exploit them to minimize latency (or create other benefits)
Latency Variation in Memory Chips

Heterogeneous manufacturing & operating conditions → latency variation in timing parameters
Why is Latency High?

- **DRAM latency**: Delay as specified in DRAM standards
  - Doesn’t reflect true DRAM device latency
- Imperfect manufacturing process → latency variation
- **High standard latency** chosen to increase yield

![Diagram showing manufacturing variation and DRAM latency](image)
What Causes the Long Memory Latency?

- **Conservative timing margins!**

- **DRAM timing parameters are set to cover the worst case**

- **Worst-case temperatures**
  - 85 degrees vs. common-case
  - to enable a wide range of operating conditions

- **Worst-case devices**
  - DRAM cell with smallest charge across any acceptable device
  - to tolerate process variation at acceptable yield

- **This leads to large timing margins for the common case**
Understanding and Exploiting Variation in DRAM Latency
DRAM Stores Data as Charge

Three steps of charge movement

1. Sensing
2. Restore
3. Precharge
Why does DRAM need the extra timing margin?
Two Reasons for Timing Margin

1. Process Variation
   - DRAM cells are not equal
   - Leads to extra timing margin for a cell that can store a large amount of charge

2. Temperature Dependence
DRAM Cells are Not Equal

**Ideal**

- Same Size
- Same Charge
- Same Latency

**Real**

- Different Size
- Different Charge
- Different Latency

Large variation in cell size
Large variation in charge
Large variation in access latency

Largest Cell
Smallest Cell
Process Variation

DRAM Cell
- Capacitor
- Bitline
- Contact
- Access Transistor
- ACCESS

Small cell can store small charge
- Small cell capacitance
- High contact resistance
- Slow access transistor

High access latency
Two Reasons for Timing Margin

1. Process Variation
   - DRAM cells are not equal
   - Leads to extra timing margin for a cell that can store a large amount of charge

2. Temperature Dependence
   - DRAM leaks more charge at higher temperature
   - Leads to extra timing margin for cells that operate at low temperature
Charge Leakage vs. Temperature

Cells store small charge at high temperature and large charge at low temperature → Large variation in access latency
DRAM Timing Parameters

• **DRAM timing parameters are dictated by the worst-case**
  – The smallest cell with the smallest charge **in all DRAM products**
  – Operating at **the highest temperature**

• **Large timing margin for the common-case**
Adaptive-Latency DRAM [HPCA 2015]

- Idea: Optimize DRAM timing for the common case
  - Current temperature
  - Current DRAM module

- Why would this reduce latency?
  - A DRAM cell can store much more charge in the common case (low temperature, strong cell) than in the worst case
  - More charge in a DRAM cell
    - Faster sensing, charge restoration, precharging
    - Faster access (read, write, refresh, ...)

Extra Charge $\rightarrow$ Reduced Latency

1. Sensing
   Sense cells with extra charge faster
   $\rightarrow$ Lower sensing latency

2. Restore
   No need to fully restore cells with extra charge
   $\rightarrow$ Lower restoration latency

3. Precharge
   No need to fully precharge bitlines for cells with extra charge
   $\rightarrow$ Lower precharge latency

- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

github.com/CMU-SAFARI/SoftMC
SoftMC: Open Source DRAM Infrastructure

- https://github.com/CMU-SAFARI/SoftMC

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan\textsuperscript{1,2,3} Nandita Vijaykumar\textsuperscript{3} Samira Khan\textsuperscript{4,3} Saugata Ghose\textsuperscript{3} Kevin Chang\textsuperscript{3} Gennady Pekhimenko\textsuperscript{5,3} Donghyuk Lee\textsuperscript{6,3} Oguz Ergin\textsuperscript{2} Onur Mutlu\textsuperscript{1,3}

\textsuperscript{1}ETH Zürich \quad \textsuperscript{2}TOBB University of Economics & Technology \quad \textsuperscript{3}Carnegie Mellon University
\textsuperscript{4}University of Virginia \quad \textsuperscript{5}Microsoft Research \quad \textsuperscript{6}NVIDIA Research
Observation 1. Faster Sensing

**Typical DIMM at Low Temperature**

- More Charge
- Strong Charge Flow
- Faster Sensing

**115 DIMM Characterization**

**Timing**

- \((t_{RCD})\)
- 17% ↓
- No Errors

**Typical DIMM at Low Temperature**

→ More charge  ➔ Faster sensing
Observation 2. Reducing Restore Time

Typical DIMM at Low Temperature

- Less Leakage ➔
- Extra Charge
- No Need to Fully Restore Charge

Read (t_{RAS})
37% ↓

Write (t_{WR})
54% ↓
No Errors

Typical DIMM at lower temperature

➔ More charge ➔ Restore time reduction
AL-DRAM

• *Key idea*
  – Optimize DRAM timing parameters online

• *Two components*
  – DRAM manufacturer provides multiple sets of reliable DRAM timing parameters at different temperatures for each DIMM
  – System monitors DRAM temperature & uses appropriate DRAM timing parameters

DRAM Temperature

- **DRAM temperature measurement**
  - Server cluster: Operates at under 34°C
  - Desktop: Operates at under 50°C
  - **DRAM standard optimized for 85 °C**

**DRAM operates at low temperatures in the common-case**

- **Previous works – Maintain low DRAM temperature**
  - David+ ICAC 2011
  - Liu+ ISCA 2007
  - Zhu+ I ThERM 2008
Latency Reduction Summary of 115 DIMMs

• *Latency reduction for read & write (55°C)*
  – Read Latency: 32.7%
  – Write Latency: 55.1%

• *Latency reduction for each timing parameter (55°C)*
  – Sensing: 17.3%
  – Restore: 37.3% (read), 54.8% (write)
  – Precharge: 35.2%

AL-DRAM: Real System Evaluation

- **System**
  - **CPU**: AMD 4386 (8 Cores, 3.1GHz, 8MB LLC)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved.</td>
</tr>
<tr>
<td>29:24</td>
<td><strong>Tras</strong>: row active strobe. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from an activate command to a precharge command, both to the same chip select bank.</td>
</tr>
<tr>
<td></td>
<td>Bits</td>
</tr>
<tr>
<td>07h-00h</td>
<td>Reserved</td>
</tr>
<tr>
<td>2Ah-08h</td>
<td>&lt;Tras&gt; clocks</td>
</tr>
<tr>
<td>3Fh-2Bh</td>
<td>Reserved</td>
</tr>
<tr>
<td>23:21</td>
<td>Reserved.</td>
</tr>
<tr>
<td>20:16</td>
<td><strong>Trp</strong>: row precharge time. Read-write. BIOS: See 2.9.7.5 [SPD ROM-Based Configuration]. Specifies the minimum time in memory clock cycles from a precharge command to an activate command or auto refresh refresh command, both to the same bank.</td>
</tr>
</tbody>
</table>
AL-DRAM: Single-Core Evaluation

- **Performance Improvement**
  - soplex: 0%
  - mcf: 5%
  - milc: 10%
  - libq: 15%
  - lbm: 20%
  - gems: 25%
  - copy: 20%
  - s.cluster: 5%
  - gups: 6.7%
  - non-intensive: 1.4%
  - intensive: 5.0%
  - all-workload: 35%

**Average Improvement**

- AL-DRAM improves performance on a real system.
AL-DRAM provides higher performance for multi-programmed & multi-threaded workloads
Reducing Latency Also Reduces Energy

- AL-DRAM reduces DRAM power consumption by 5.8%
- Major reason: reduction in row activation time
AL-DRAM: Advantages & Disadvantages

- **Advantages**
  + Simple mechanism to reduce latency
  + Significant system performance and energy benefits
    + Benefits higher at low temperature
  + Low cost, low complexity

- **Disadvantages**
  - Need to determine reliable operating latencies for different temperatures and different DIMMs $\rightarrow$ higher testing cost
    (might not be that difficult for low temperatures)
More on AL-DRAM

- Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case"
  [Slides (pptx) (pdf)] [Full data sets]
Different Types of Latency Variation

- AL-DRAM exploits latency variation
  - Across time (different temperatures)
  - Across chips

- Is there also latency variation within a chip?
  - Across different parts of a chip
Modern DRAM chips exhibit significant variation in activation latency
Spatial Locality of Activation Errors

Activation errors are concentrated at certain columns of cells
Mechanism to Reduce DRAM Latency

• **Observation:** DRAM timing errors (slow DRAM cells) are concentrated on certain regions

• **Flexible-Latency (FLY) DRAM**
  – A software-transparent design that reduces latency

• **Key idea:**
  1) Divide memory into regions of different latencies
  2) *Memory controller:* Use lower latency for regions without slow cells; higher latency for other regions

Chang+, “**Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization**,” SIGMETRICS 2016.
FLY-DRAM improves performance by exploiting spatial latency variation in DRAM

FLY-DRAM: Advantages & Disadvantages

- **Advantages**
  - Reduces latency significantly
  - Exploits significant within-chip latency variation

- **Disadvantages**
  - Need to determine reliable operating latencies for different parts of a chip → higher testing cost
  - Slightly more complicated controller
Analysis of Latency Variation in DRAM Chips

- Kevin Chang, Abhijith Kashyap, Hasan Hassan, Samira Khan, Kevin Hsieh, Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Tianshi Li, and Onur Mutlu,

"Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"


[Slides (pptx) (pdf)]
[Source Code]

Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization

Kevin K. Chang¹ Abhijith Kashyap¹ Hasan Hassan¹,²
Saugata Ghose¹ Kevin Hsieh¹ Donghyuk Lee¹ Tianshi Li¹,³
Gennady Pekhimenko¹ Samira Khan⁴ Onur Mutlu⁵,¹

¹Carnegie Mellon University ²TOBB ETÜ ³Peking University ⁴University of Virginia ⁵ETH Zürich
Spatial Distribution of Failures

How are activation failures spatially distributed in DRAM?

Activation failures are **highly constrained** to local bitlines.
Short-term Variation

Does a bitline’s probability of failure change over time?

A weak bitline is likely to remain weak and a strong bitline is likely to remain strong over time.
Short-term Variation

Does a bitline’s probability of failure change over time?

We can rely on a **static profile** of weak bitlines to determine whether an access will cause failures.

A **weak bitline** is likely to remain **weak** and a **strong bitline** is likely to remain **strong** over time.
We can reliably issue write operations with significantly reduced $t_{RCD}$ (e.g., by 77%)
Solar-DRAM

Uses a **static profile of weak subarray columns**
- Identifies subarray columns as weak or strong
- Obtained in a one-time profiling step

**Three Components**

1. Variable-latency cache lines (VLC)
2. Reordered subarray columns (RSC)
3. Reduced latency for writes (RLW)
Solar-DRAM

Uses a **static profile of weak subarray columns**
- Identifies subarray columns as weak or strong
- Obtained in a one-time profiling step

**Three Components**

1. Variable-latency cache lines (VLC)
2. Reordered subarray columns (RSC)
3. Reduced latency for writes (RLW)
Solar-DRAM: VLC (I)

Identify cache lines comprised of **strong bitlines**
Access such cache lines with a **reduced** $t_{RCD}$
Solar-DRAM

Uses a **static profile of weak subarray columns**
  - Identifies subarray columns as weak or strong
  - Obtained in a one-time profiling step

**Three Components**

1. Variable-latency cache lines (VLC)
2. Reordered subarray columns (RSC)
3. Reduced latency for writes (RLW)
Remap cache lines across DRAM at the memory controller level so cache line 0 will likely map to a strong cache line.
Solar-DRAM

Uses a static profile of weak subarray columns
• Identifies subarray columns as weak or strong
• Obtained in a one-time profiling step

Three Components
1. Variable-latency cache lines (VLC)
2. Reordered subarray columns (RSC)
3. Reduced latency for writes (RLW)
Solar-DRAM: RLW (III)

All bitlines are strong when issuing writes

Write to all locations in DRAM with a significantly reduced $t_{RCD}$ (e.g., by 77%)
More on Solar-DRAM

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines"
  [Slides (pptx) (pdf)]
  [Talk Video (16 minutes)]

Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines

Jeremie S. Kim‡§ Minesh Patel§ Hasan Hassan§ Onur Mutlu§†
‡Carnegie Mellon University §ETH Zürich

SAFARI
Memory Systems
and Memory-Centric Computing Systems

Lecture 5: Low-Latency Memory I

Prof. Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
18 June 2019
TU Wien Fast Course 2019
Backup Slides
Why Is There Spatial Latency Variation Within a Chip?
What Is Design-Induced Variation?

Systematic variation in cell access times caused by the physical organization of DRAM

Inherently fast

Inherently slow

across column
distance from wordline driver

fast
slow

cross row
distance from sense amplifier

wordline drivers

sense amplifiers
DIVA Online Profiling

Design-Induced-Variation-Aware

Profile only slow regions to determine min. latency → Dynamic & low cost latency optimization
DIVA Online Profiling

Design-Induced-Variation-Aware

- Slow cells
- Process variation
- Random error

Inherently slow
Design-induced variation
Localized error

Combine error-correcting codes & online profiling
→ Reliably reduce DRAM latency
DIVA-DRAM reduces latency more aggressively and uses ECC to correct random slow cells.
DIVA-DRAM: Advantages & Disadvantages

- **Advantages**
  
  ++ Automatically finds the lowest reliable operating latency at system runtime (lower production-time testing cost)
  
  + Reduces latency more than prior methods (w/ ECC)
  
  + Reduces latency at high temperatures as well

- **Disadvantages**
  
  - Requires knowledge of inherently-slow regions
  
  - Requires ECC (Error Correcting Codes)
  
  - Imposes overhead during runtime profiling
Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee, NVIDIA and Carnegie Mellon University
Samira Khan, University of Virginia
Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Carnegie Mellon University
Gennady Pekhimenko, Vivek Seshadri, Microsoft Research
Onur Mutlu, ETH Zürich and Carnegie Mellon University
Understanding & Exploiting the Voltage-Latency-Reliability Relationship
High DRAM Power Consumption

- **Problem:** High DRAM (memory) power in today’s systems

>40% in POWER7 \(^{(\text{Ware+}, \text{HPCA’10})}\)  \>40% in GPU \(^{(\text{Paul+}, \text{ISCA’15})}\)
Low-Voltage Memory

• Existing DRAM designs to help reduce DRAM power by lowering supply voltage conservatively
  – Power $\propto$ Voltage$^2$

• DDR3L (low-voltage) reduces voltage from 1.5V to 1.35V (-10%)

• LPDDR4 (low-power) employs low-power I/O interface with 1.2V (lower bandwidth)

Can we reduce DRAM power and energy by further reducing supply voltage?
Goals

1. Understand and characterize the various characteristics of DRAM under reduced voltage

2. Develop a mechanism that reduces DRAM energy by lowering voltage while keeping performance loss within a target
Key Questions

• How does reducing voltage affect *reliability* (errors)?

• How does reducing voltage affect *DRAM latency*?

• How do we design a new DRAM energy reduction mechanism?
Supply Voltage Control on DRAM

Adjust the supply voltage to every chip on the same module.
Custom Testing Platform

**SoftMC** [Hassan+, HPCA’17]: FPGA testing platform to

1) Adjust supply voltage to DRAM modules
2) Schedule DRAM commands to DRAM modules

Existing systems: DRAM commands not exposed to users

[Diagram of FPGA testing platform]

https://github.com/CMU-SAFA/DRAM-Voltage-Study
Tested DRAM Modules

• 124 DDR3L (low-voltage) DRAM chips
  – 31 SO-DIMMs
  – 1.35V (DDR3 uses 1.5V)
  – Density: 4Gb per chip
  – Three major vendors/manufacturers

• Iteratively read every bit in each 4Gb chip under a wide range of supply voltage levels: 1.35V to 1.0V (-26%)
Reliability Worsens with Lower Voltage

Errors induced by reduced-voltage operation

Reducing voltage below $V_{\text{min}}$ causes an increasing number of errors
Source of Errors

Detailed circuit simulations (SPICE) of a DRAM cell array to model the behavior of DRAM operations

https://github.com/CMU-SAFARI/DRAM-Voltage-Study

Circuit model

Reliable low-voltage operation requires higher latency
DIMMs Operating at Higher Latency

Measured minimum latency that does not cause errors in DRAM modules

Distribution of latency in the total population

DRAM requires longer latency to access data without errors at lower voltage
Spatial Locality of Errors

A module under 1.175V (12% voltage reduction)

Errors concentrate in certain regions
Summary of Key Experimental Observations

- Voltage-induced errors increase as voltage reduces further below $V_{\text{min}}$

- Errors exhibit spatial locality

- Increasing the latency of DRAM operations mitigates voltage-induced errors
**DRAM Voltage Adjustment to Reduce Energy**

- **Goal:** Exploit the trade-off between voltage and latency to reduce energy consumption.

- **Approach:** Reduce DRAM voltage **reliably**
  - Performance loss due to increased latency at lower voltage.

---

![Graph showing DRAM performance and power savings vs. supply voltage]

- **Performance**
- **High Power Savings**
- **Low Power Savings**
- **Bad Performance**
- **Good Performance**

**Supply Voltage (V)**

<table>
<thead>
<tr>
<th>Performance Over Nominal Voltage (%)</th>
<th>0.9</th>
<th>1.0</th>
<th>1.1</th>
<th>1.2</th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High Power Savings</strong></td>
<td>+30</td>
<td>+20</td>
<td>+10</td>
<td>+5</td>
<td>+3</td>
</tr>
<tr>
<td><strong>Low Power Savings</strong></td>
<td>-7</td>
<td>-8</td>
<td>-12</td>
<td>-15</td>
<td>-18</td>
</tr>
<tr>
<td><strong>Bad Performance</strong></td>
<td>-20</td>
<td>-30</td>
<td>-40</td>
<td>-50</td>
<td>-60</td>
</tr>
<tr>
<td><strong>Good Performance</strong></td>
<td>+15</td>
<td>+30</td>
<td>+45</td>
<td>+60</td>
<td>+75</td>
</tr>
</tbody>
</table>
How do we predict performance loss due to increased latency under low DRAM voltage?

User specifies the **performance loss target**

Voltron

Select the **minimum** DRAM voltage without violating the target
Linear Model to Predict Performance

User specifies the performance loss target

Select the minimum DRAM voltage without violating the target

Application’s characteristics

[1.3V, 1.25V, ...] DRAM Voltage

Linear regression model

[-1%, -3%, ...] Predicted performance loss

Min. Voltage

Final Voltage

Target
Regression Model to Predict Performance

- **Application’s characteristics for the model:**
  - *Memory intensity:* Frequency of last-level cache misses
  - *Memory stall time:* Amount of time memory requests stall commit inside CPU

- **Handling multiple applications:**
  - Predict a performance loss for each application
  - Select the minimum voltage that satisfies the performance target for all applications
Comparison to Prior Work

- **Prior work**: Dynamically scale *frequency and voltage* of the entire DRAM based on bandwidth demand [David+, ICAC’11]
  - **Problem**: Lowering voltage on the peripheral circuitry decreases channel frequency (memory data throughput)

- **Voltron**: Reduce voltage to only *DRAM array* without changing the voltage to peripheral circuitry
Exploiting Spatial Locality of Errors

Key idea: Increase the latency only for DRAM banks that observe errors under low voltage

- Benefit: Higher performance
Voltron Evaluation Methodology

• **Cycle-level simulator:** Ramulator [CAL’15]
  – McPAT and DRAMPower for energy measurement
    [https://github.com/CMU-SAFAI/ramulator](https://github.com/CMU-SAFAI/ramulator)

• **4-core** system with DDR3L memory

• **Benchmarks:** SPEC2006, YCSB

• **Comparison to prior work:** MemDVFS [David+, ICAC’11]
  – Dynamic DRAM frequency and voltage scaling
  – Scaling based on the *memory bandwidth consumption*
Energy Savings with Bounded Performance

- More savings for high bandwidth applications
- Meets performance target

- 3.2% energy savings for low memory intensity
- 7.3% energy savings for high memory intensity

- -1.6% performance loss for low memory intensity
- -1.8% performance loss for high memory intensity

[David+, ICAC’11]
Voltron: Advantages & Disadvantages

- **Advantages**
  - Can trade-off between voltage and latency to improve energy or performance
  - Can exploit the high voltage margin present in DRAM

- **Disadvantages**
  - Requires finding the reliable operating voltage for each chip → higher testing cost
Analysis of Latency-Voltage in DRAM Chips

Kevin Chang, A. Giray Yaglikci, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu,
"Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms"
And, What If …

- … we can sacrifice reliability of some data to access it with even lower latency?
Reducing Memory Latency to Support Security Primitives
Using Memory for Security

- Generating True Random Numbers (using DRAM)
  - Kim et al., HPCA 2019

- Evaluating Physically Unclonable Functions (using DRAM)
  - Kim et al., HPCA 2018

- Quickly Destroying In-Memory Data (using DRAM)
  - Orosa et al., arxiv 2019
D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, Onur Mutlu

SAFARI

ETH Zürich, Carnegie Mellon

HPCA 2019
D-RaNGe Executive Summary

• **Motivation**: High-throughput true random numbers enable system security and various randomized algorithms.
  • Many systems (e.g., IoT, mobile, embedded) do not have dedicated **True Random Number Generator (TRNG)** hardware but have DRAM devices

• **Problem**: Current DRAM-based TRNGs either
  1. do **not** sample a fundamentally non-deterministic entropy source
  2. are **too slow** for continuous high-throughput operation

• **Goal**: A novel and effective TRNG that uses **existing** commodity DRAM to provide random values with 1) **high-throughput**, 2) **low latency** and 3) no adverse effect on concurrently running applications

• **D-RaNGe**: Reduce DRAM access latency **below reliable values** and exploit DRAM cells’ failure probabilities to generate random values

• **Evaluation**:
  1. Experimentally characterize **282 real LPDDR4 DRAM devices**
  2. **D-RaNGe (717.4 Mb/s)** has significantly higher throughput (**211x**)
  3. **D-RaNGe (100ns)** has significantly lower latency (**180x**)

SAFARI
DRAM Latency Characterization of 282 LPDDR4 DRAM Devices

• Latency failures come from accessing DRAM with reduced timing parameters.

• Key Observations:
  1. A cell’s latency failure probability is determined by random process variation
  2. Some cells fail randomly
DRAM Accesses and Failures

Process variation during manufacturing results in cells having unique behavior.

Guardband

Sense Amplifier

Bitline Charge Sharing

0.5 Vdd

Vdd

Vmin

Time

tRCD

ACTIVATE

SA Enable

READ

Strong

Weak

Ready to Access Voltage Level

Bitline Voltage

wordline

access transistor

Guardband
DRAM Accesses and Failures

Weaker cells have a higher probability to fail.
D-RaNGe Key Idea

High % chance to fail with reduced $t_{RCD}$

Low % chance to fail with reduced $t_{RCD}$

Fails randomly with reduced $t_{RCD}$
D-RaNGe Key Idea

We refer to cells that fail randomly when accessed with a reduced $t_{RCD}$ as RNG cells.
Our D-RaNGe Evaluation

• We generate random values by repeatedly accessing RNG cells and aggregating the data read.

• The random data satisfies the NIST statistical test suite for randomness.

• The D-RaNGe generates random numbers:
  - Throughput: 717.4 Mb/s
  - Latency: 64 bits in <1us
  - Power: 4.4 nJ/bit
D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim  Minesh Patel
Hasan Hassan    Lois Orosa    Onur Mutlu

SAFARI

HPCA 2019

ETH Zürich  Carnegie Mellon
More on D-RaNGe


D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim‡§ Minesh Patel§ Hasan Hassan§ Lois Orosa§ Onur Mutlu§‡
‡Carnegie Mellon University §ETH Zürich
The DRAM Latency PUF:
Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim  Minesh Patel
Hasan Hassan    Onur Mutlu

SAFARI
Systems@ETH Zürich
ETH Zürich
Carnegie Mellon
DL-PUF: Executive Summary

- **Motivation:**
  - We can authenticate a system via *unique signatures* if we can evaluate a **Physical Unclonable Function (PUF)** on it
  - Signatures (**PUF response**) reflect inherent properties of a device
  - DRAM is a promising substrate for PUFs because it is **widely** used

- **Problem:** Current DRAM PUFs are 1) very slow, 2) require a DRAM reboot, or 3) require additional custom hardware

- **Goal:** To develop a novel and effective PUF for **existing** commodity DRAM devices with **low-latency evaluation time** and **low system interference** across all operating temperatures

- **DRAM Latency PUF:** Reduce DRAM access latency below **reliable values** and exploit the resulting error patterns as **unique identifiers**

- **Evaluation:**
  1. Experimentally characterize 223 real LPDDR4 DRAM devices
  2. **DRAM latency PUF** (88.2 ms) achieves a speedup of 102x/860x at 70°C/55°C over prior DRAM PUF evaluation mechanisms
Motivation

We want a way to ensure that a system’s components are not **compromised**

- **Physical Unclonable Function (PUF):** a function we **evaluate** on a device to **generate** a **signature unique** to the device
- We refer to the unique signature as a **PUF response**
- Often used in a **Challenge-Response Protocol (CRP)**

---

**Input:**

- **Trusted Device**
  - Checking PUF response...

- **Output:**
  - PUF Response

**Authenticated Device**

- **Evaluating PUF...**
Motivation

1. We want a **runtime-accessible** PUF
   - Should be evaluated **quickly** with **minimal** impact on concurrent applications
   - Can protect against **attacks that swap system components with malicious parts**

2. DRAM is a **promising substrate** for evaluating PUFs because it is **ubiquitous** in modern systems
   - Unfortunately, current DRAM PUFs are **slow** and get **exponentially slower** at lower temperatures
DRAM Latency Characterization of 223 LPDDR4 DRAM Devices

- Latency failures come from accessing DRAM with reduced timing parameters.

- Key Observations:
  1. A cell’s latency failure probability is determined by random process variation
  2. Latency failure patterns are repeatable and unique to a device
DRAM Latency PUF Key Idea

- A cell’s latency failure probability is inherently related to **random process variation** from manufacturing.
- We can provide **repeatable and unique device signatures** using latency error patterns.

High % chance to fail with reduced $t_{RCD}$

Low % chance to fail with reduced $t_{RCD}$
The **key idea** is to compose a PUF response using the DRAM cells that fail with **high probability**
The DRAM Latency PUF Evaluation

• We generate PUF responses using *latency errors* in a region of DRAM

• The latency error patterns *satisfy PUF requirements*

• The DRAM Latency PUF *generates PUF responses in 88.2ms*
Results – PUF Evaluation Latency

DRAM latency PUF is

1. Fast and constant latency (88.2ms)
Results – PUF Evaluation Latency

DRAM latency PUF is

1. Fast and constant latency (88.2ms)
Results – PUF Evaluation Latency

<table>
<thead>
<tr>
<th>Temperature (°C)</th>
<th>DRAM Retention PUF</th>
<th>DRAM Latency PUF</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>Manufacturer A</td>
<td>All Manufacturers</td>
</tr>
<tr>
<td>58</td>
<td>Manufacturer B</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Manufacturer C</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Evaluation Time (s)**

- 8KiB memory segment: 11.5x
- 64KiB memory segment: 17.3x
- 64MiB memory segment: 100x
- 8KiB memory segment: 1000x

**DRAM latency PUF is**

1. Fast and constant latency (88.2ms)
Results – PUF Evaluation Latency

DRAM latency PUF is

1. Fast and constant latency (88.2ms)
2. On average, 102x/860x faster than the previous DRAM PUF with the same DRAM capacity overhead (64KiB)
Other Results in the Paper

• How the DRAM latency PUF meets the basic requirements for an effective PUF

• A detailed analysis on:
  - Devices of the three major DRAM manufacturers
  - The evaluation time of a PUF

• Further discussion on:
  - Optimizing retention PUFs
  - System interference of DRAM retention and latency PUFs
  - Algorithm to quickly and reliably evaluate DRAM latency PUF
  - Design considerations for a DRAM latency PUF
  - The DRAM Latency PUF overhead analysis
The DRAM Latency PUF:
Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim    Minesh Patel
Hasan Hassan      Onur Mutlu

DRAM Latency PUFs

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"


[Lightning Talk Video]
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
Reducing Refresh Latency
On Reducing Refresh Latency

Anup Das, Hasan Hassan, and Onur Mutlu, "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency"
Proceedings of the 55th Design Automation Conference (DAC), San Francisco, CA, USA, June 2018.
[Slides (pdf)] [Poster (pdf)]

VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency

Anup Das  
Drexel University  
Philadelphia, PA, USA  
anup.das@drexel.edu

Hasan Hassan  
ETH Zürich  
Zürich, Switzerland  
hhasan@ethz.ch

Onur Mutlu  
ETH Zürich  
Zürich, Switzerland  
omutlu@gmail.com
Reducing Memory Latency by Exploiting Memory Access Patterns
ChargeCache: Executive Summary

- **Goal**: Reduce average DRAM access latency with no modification to the existing DRAM chips

- **Observations**:
  1) A highly-charged DRAM row can be accessed with low latency
  2) A row’s charge is restored when the row is accessed
  3) A recently-accessed row is likely to be accessed again:
     **Row Level Temporal Locality (RLTL)**

- **Key Idea**: Track recently-accessed DRAM rows and use lower timing parameters if such rows are accessed again

- **ChargeCache**:
  - Low cost & no modifications to the DRAM
  - Higher performance (**8.6-10.6%** on average for 8-core)
  - Lower DRAM energy (**7.9%** on average)
DRAM Charge over Time

Ready to Access
Ready to Precharge
Ready to Access Charge Level

Cell
Sense-Amplifier

Sensing
Restore
Precharge
tRCD
tRAS

ACT
R/W
PRE

Data 0
Data 1

SAFARI
Accessing Highly-charged Rows

- Ready to Access
- Ready to Precharge

Cell
Sense-Amplifier

Data 0
Data 1

Charge: Sensing, Restore, Precharge

-tRCD
- tRAS

SAFARI
Observation 1

A highly-charged DRAM row can be accessed with low latency

- tRCD: 44%
- tRAS: 37%

How does a row become highly-charged?
How Does a Row Become Highly-Charged?

DRAM cells lose charge over time

Two ways of restoring a row’s charge:

- Refresh Operation
- Access
Observation 2

A row’s charge is restored when the row is accessed

How likely is a recently-accessed row to be accessed again?
Row Level Temporal Locality (RLTL)

A recently-accessed DRAM row is likely to be accessed again.

- **t-RLTL**: Fraction of rows that are accessed within time *t* after their previous access.

![Bar Chart](image)

- **88%**—RLTL for single-core workloads
- **97%**—RLTL for eight-core workloads
Key Idea

Track recently-accessed DRAM rows and use lower timing parameters if such rows are accessed again.
ChargeCache Overview

Requests: A D A

ChargeCache Hits: Use Default Timings
Area and Power Overhead

• Modeled with CACTI

• Area
  – ~5KB for 128-entry ChargeCache
  – 0.24% of a 4MB Last Level Cache (LLC) area

• Power Consumption
  – 0.15 mW on average (static + dynamic)
  – 0.23% of the 4MB LLC power consumption
Methodology

• Simulator
  – DRAM Simulator (Ramulator [Kim+, CAL’15])
    https://github.com/CMU-SAFARI/ramulator

• Workloads
  – 22 single-core workloads
    • SPEC CPU2006, TPC, STREAM
  – 20 multi-programmed 8-core workloads
    • By randomly choosing from single-core workloads
  – Execute at least 1 billion representative instructions per core (Pinpoints)

• System Parameters
  – 1/8 core system with 4MB LLC
  – Default tRCD/tRAS of 11/28 cycles
ChargeCacheCache improves single-core performance
Eight-core Performance

- **NUAT** 2.5%
- **ChargeCache** 9%
- **ChargeCache + NUAT**
- **LL-DRAM (Upperbound)** 13%

ChargeCache significantly improves multi-core performance
DRAM Energy Savings

ChargeCache reduces DRAM energy
More on ChargeCache

Hasan Hassan, Gennady Pekhimenko, Nandita Vijaykumar, Vivek Seshadri, Donghyuk Lee, Oguz Ergin, and Onur Mutlu,
"ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality"
[Slides (pptx) (pdf)]
[Source Code]
A Very Recent Work

- Yaohua Wang, Arash Tavakkol, Lois Orosa, Saugata Ghose, Nika Mansouri Ghiasi, Minesh Patel, Jeremie S. Kim, Hasan Hassan, Mohammad Sadrosadati, and Onur Mutlu,

"Reducing DRAM Latency via Charge-Level-Aware Look-Ahead Partial Restoration"

Summary: Low-Latency Memory
Reason 1: Design of DRAM Micro-architecture
- Goal: Maximize capacity/area, not minimize latency

Reason 2: “One size fits all” approach to latency specification
- Same latency parameters for all temperatures
- Same latency parameters for all DRAM chips (e.g., rows)
- Same latency parameters for all parts of a DRAM chip
- Same latency parameters for all supply voltage levels
- Same latency parameters for all application data
- ...

Summary: Tackling Long Memory Latency
Challenge and Opportunity for Future

Fundamentally Low Latency Computing Architectures
On DRAM Power Consumption
Power Measurement Platform

Keysight 34134A DC Current Probe

DDR3L SO-DIMM

Virtex 6 FPGA

JET-5467A Riser Board
Power Measurement Methodology

- **SoftMC: an FPGA-based memory controller** [Hassan+ HPCA ’17]
  - Modified to repeatedly loop commands
  - Open-source: https://github.com/CMU-SAFARI/SoftMC

- Measure current consumed by a module during a SoftMC test

- **Tested 50 DDR3L DRAM modules** (200 DRAM chips)
  - Supply voltage: 1.35 V
  - **Three major vendors: A, B, C**
  - Manufactured between 2014 and 2016

- **For each experimental test that we perform**
  - 10 runs of each test per module
  - At least 10 current samples per run
1. Real DRAM Power Varies Widely from IDD Values

- Different vendors have very different margins (i.e., guardbands)
- Low variance among different modules from same vendor

Current consumed by real DRAM modules varies significantly for all IDD values that we measure
2. DRAM Power is Dependent on Data Values

- Some variation due to infrastructure – can be subtracted
- Without infrastructure variation: up to 230 mA of change
- Toggle affects power consumption, but < 0.15 mA per bit

DRAM power consumption depends strongly on the data value
3. Structural Variation Affects DRAM Power Usage

- Vendor C: variation in idle current across banks
- All vendors: variation in read current across banks
- All vendors: variation in activation based on row address

Significant structural variation: DRAM power varies systematically by bank and row
4. Generational Savings Are Smaller Than Expected

- Similar trends for idle and read currents

Actual power savings of newer DRAM is much lower than the savings indicated in the datasheets.
Summary of New Observations on DRAM Power

1. Real DRAM modules often **consume less power** than vendor-provided IDD values state

2. DRAM power consumption is **dependent on the data value** that is read/written

3. Across banks and rows, **structural variation affects power consumption of DRAM**

4. **Newer DRAM modules save less power** than indicated in datasheets by vendors

*Detailed observations and analyses in the paper*
A New Variation-Aware DRAM Power Model

- **VAMPIRE**: Variation-Aware model of Memory Power Informed by Real Experiments

**Inputs**
- Trace of DRAM commands, timing
- Data that is being written

**VAMPIRE**
- Read/Write and Data-Dependent Power Modeling
- Idle/Activate/Precharge Power Modeling
- Structural Variation Aware Power Modeling

**Outputs**
- Per-vendor power consumption
- Range for each vendor (optional)

- **VAMPIRE** and raw characterization data will be open-source: [https://github.com/CMU-SAFARI/VAMPIRE](https://github.com/CMU-SAFARI/VAMPIRE)
VAMPIRE Has Lower Error Than Existing Models

- Validated using new power measurements: details in the paper

VAMPIRE has very low error for all vendors: 6.8%
Much more accurate than prior models
VAMPIRE Enables Several New Studies

- Taking advantage of structural variation to perform variation-aware physical page allocation to reduce power

- Smarter DRAM power-down scheduling

- Reducing DRAM energy with data-dependency-aware cache line encodings
  - 23 applications from the SPEC 2006 benchmark suite
  - Traces collected using Pin and Ramulator

- We expect there to be many other new studies in the future
What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study

Saugata Ghose†, Abdullah Giray Yağlıkçı‡†, Raghav Gupta†, Donghyuk Lee§, Kais Kudrolli†, William X. Liu†, Hasan Hassan‡, Kevin K. Chang†, Niladrish Chatterjee§, Aditya Agrawal§, Mike O’Connor§‖, and Onur Mutlu‡†.


[Abstract]
Conclusion
Four Key Directions

- Fundamentally Secure/Reliable/Safe Architectures

- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally Low-Latency Architectures

- Architectures for Genomics, Medicine, Health
Some Solution Principles (So Far)

- Data-centric system design & intelligence spread around
  - Do not center everything around traditional computation units

- Better cooperation across layers of the system
  - Careful co-design of components and layers: system/arch/device
  - Better, richer, more expressive and flexible interfaces

- Better-than-worst-case design
  - Do not optimize for the worst case
  - Worst case should not determine the common case

- Heterogeneity in design (specialization, asymmetry)
  - Enables a more efficient design (No one size fits all)
Some Solution Principles (More Compact)

- Data-centric design
- All components intelligent
- Better cross-layer communication, better interfaces
- Better-than-worst-case design
- Heterogeneity
- Flexibility, adaptability

Open minds
Data-Aware Architectures

- A data-aware architecture understands what it can do with and to each piece of data.

- It makes use of different properties of data to improve performance, efficiency and other metrics:
  - Compressibility
  - Approximability
  - Locality
  - Sparsity
  - Criticality for Computation X
  - Access Semantics
  - ...

One Problem: Limited Interfaces

Higher-level information is not visible to HW

Software

Hardware

- Data Structures
- Code Optimizations
- Access Patterns
- Integer
- Float
- Char

100011111...
101010011...

Instructions
Memory Addresses
A Solution: More Expressive Interfaces
Expressive (Memory) Interfaces

- Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons and Onur Mutlu,

"A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory"

[Slides (pptx) (pdf)] [Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video]
Expressive (Memory) Interfaces for GPUs

- Nandita Vijaykumar, Eiman Ebrahimi, Kevin Hsieh, Phillip B. Gibbons and Onur Mutlu, "The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs"


[Slides (pptx) (pdf)] [Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video]

The Locality Descriptor:
A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

Nandita Vijaykumar†§ Eiman Ebrahimi‡ Phillip B. Gibbons†
Kevin Hsieh† Onur Mutlu§†

†Carnegie Mellon University ‡NVIDIA §ETH Zürich
Architectures for Intelligent Machines

Data-centric

Data-driven

Data-aware
It Is Time to ...

- ... design **principled system architectures** to solve the memory problem

- ... design complete systems to be balanced, high-performance, and energy-efficient, i.e., data-centric (or memory-centric)

- ... make memory a **key priority** in system design and optimize it & integrate it better into the system

This can

- Lead to **orders-of-magnitude** improvements
- Enable new applications & computing platforms
- Enable better understanding of nature
- ...

240
We Need to Revisit the Entire Stack

We can get there step by step