Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee1,2 Samira Khan3 Lavanya Subramanian2 Saugata Ghose2 Rachata Ausavarungnirun2 Gennady Pekhimenko4 Vivek Seshadri4 Onur Mutlu5,2

1NVIDIA 2Carnegie Mellon University 3University of Virginia 4Microsoft Research 5ETH Zürich

June 7, 2017
What Is Design-Induced Variation?

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

SAFARI
Executive Summary

• **Design-Induced Variation**
  – *Inherently slow regions* due to DRAM cell array organization
  – Goal: Use design-induced variation to reduce DRAM latency

• Analysis: Characterization of **96 real DRAM modules**
  – Three types of *systematic* variation due to design
  – Great *potential* to *reduce DRAM latency* at low cost

• Our Approach: **DIVA-DRAM**
  – **DIVA Profiling**: Reliably reduce DRAM latency
    • Profile *only* cells in *inherently slow regions*
    • Use error correction (ECC) for slow cells that are not profiled
  – **DIVA Shuffling**: Exploit variation to improve ECC reliability

• **15.1%/14.2% higher performance** for 2-/8-core workloads

SAFARI
## Presentation Outline

- **DRAM Background**

- **Experimental Study of Design-Induced Variation**
  - **Goal**: Understand, identify inherently slower regions
  - **Methodology**: Profile 96 real DRAM modules by using FPGA-based DRAM test infrastructure

- **Exploiting Design-Induced Variation**
  - **DIVA Profiling**: Using low-cost slow region profiling to reliably and dynamically reduce DRAM latency
  - **DIVA Shuffling**: Using variation across data bursts to reduce uncorrectable errors
High-Level DRAM Organization

memory channel: 64-bit data bus

8-bit data bus per chip

Processor

DIMM
dual in-line
memory module

DRAM chip
High-Level DRAM Organization

memory channel: 64-bit data bus
- 8-bit data bus per chip

data burst

Processor  Read request  DIMM
dual in-line memory module
High-Level DRAM Organization

memory channel: 64-bit data bus

8 chips X 8 bits X 8 bursts = 64 bytes

Processor

Read request

DIMM
dual in-line memory module
Organization Inside a DRAM Chip

- DRAM mat
- DRAM cell
- Wordline drivers
- Sense amplifiers
Organization Inside a DRAM Chip
DRAM Operations

• *Memory controller* sends commands to DRAM
DRAM Operations

- **Memory controller** sends commands to DRAM

- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells
DRAM Operations

• **Memory controller** sends commands to DRAM

  ![Diagram of DRAM operations]

  • **Activation**: Open one row in each mat
    – Row decoder, wordline drivers select a row of cells
    – Each cell in the row shares charge with a sense amplifier
DRAM Operations

• **Memory controller** sends commands to DRAM

• **Activation**: Open one row in each mat
  – Row decoder, wordline drivers select a row of cells
  – Each cell in the row shares charge with a sense amplifier

• **Read**: Send one column of data from sense amplifiers to CPU
DRAM Operations

• Memory controller sends commands to DRAM

• Activation: Open one row in each mat
  – Row decoder, wordline drivers select a row of cells
  – Each cell in the row shares charge with a sense amplifier

• Read: Send one column of data from sense amplifiers to CPU

• Precharge: Prepare mats for next row
DRAM Operations

- **Memory controller** sends commands to DRAM

- **Activation**: Open one row in each mat
  - Row decoder, wordline drivers select a row of cells
  - Each cell in the row shares charge with a sense amplifier

- **Read**: Send one column of data from sense amplifiers to CPU

- **Precharge**: Prepare mats for next row

- **Timing parameters**: how long to wait for each step to finish
DRAM Timing Parameters

• Standard timing parameters are dictated by the worst case

• Must ensure reliable operation for:
  – The smallest cell with the smallest charge in all DRAM modules (process variation)
  – The highest operating temperature allowed (charge leakage)

• Large timing margin for the common case

→ Goal: Lower common-case latency
DRAM Timing Parameters

• Standard timing parameters are dictated by the worst case
• Must ensure reliable operation for:
  – The smallest cell with the smallest charge in all DRAM modules (process variation)
  – The highest operating temperature allowed (charge leakage)

Can we use design-induced variation to find and use common-case latency at low cost?
Presentation Outline

• DRAM Background

• Experimental Study of Design-Induced Variation
  – **Goal:** Understand, identify inherently slower regions
  – **Methodology:** Profile 96 real DRAM modules by using FPGA-based DRAM test infrastructure

• Exploiting Design-Induced Variation
  – **DIVA Profiling:** Using low-cost slow region profiling to reliably and dynamically reduce DRAM latency
  – **DIVA Shuffling:** Using variation across data bursts to reduce uncorrectable errors
Expected DRAM Characteristics

• Variation
  – Some regions are slower than others
  – If we reduce DRAM latency, slower regions are more vulnerable to errors

• Repeatability
  – Latency (error) characteristics repeat periodically, if the same component (e.g., mat) is duplicated

• Similarity
  – Characteristics repeat across different organizations (e.g., chip/DIMM) that share same design
Characterization Methodology

- We use **error behavior when we reduce latency** to infer DRAM organization and characteristics
  - FPGA-based memory controller infrastructure
DRAM Testing Infrastructure

Temperature Controller

FPGAs

Heater

FPGAs

PC

Tested 96 DIMMs from three vendors
Characterization Methodology

- We use **error behavior when we reduce latency** to infer DRAM organization and characteristics
  - FPGA-based memory controller infrastructure
  - We reverse engineer row address mapping
  - Lower tRP (precharge timing parameter) to 7.5 ns

- We characterize three types of variation
  - Variation across columns
  - Variation across rows
  - Variation across data bursts

- Data and circuit model will be available on GitHub: [https://github.com/CMU-SAFARI/DIVA-DRA](https://github.com/CMU-SAFARI/DIVA-DRA)
1. Variation Across Rows

Latency characteristics **vary** across 512 rows
1. Variation Across Rows

Latency characteristics **vary** across 512 rows.

Same organization repeats every 512 rows.
1. Variation Across Rows

Latency characteristics **vary** across 512 rows
Latency characteristics **repeat** every 512 rows
1.2. Periodic Row Variation Behavior

Row error (latency) characteristics periodically repeat every 512 rows
2. Variation Across Columns

Column latency depends on distance from row decoder, wordline driver

*darker cells are faster*
2. Variation Across Columns

Column latency depends on distance from row decoder, wordline driver
2. Variation Across Columns

Column error (latency) characteristics have specific patterns that repeat across row groups.
3. Variation Across Data Bursts

64-bit data bus in memory channel

Processor

Read request

DIMM

Chip 1
Chip 2
Chip 3
Chip 4
Chip 5
Chip 6
Chip 7
Chip 8

64-bit data from different locations in the same row in the same chip
3. Variation Across Data Bursts

Specific bits in a request *induce more errors*
Summary: Design-Induced Variation

• **Systematic variation across rows**
  – Slow cells further from sense amplifier

• **Systematic variation across columns**
  – Slow cells further from row decoder
  – Slow cells further from wordline driver

• **Systematic variation across data bursts**
  – Slow cells at certain bits in a burst
  – Clustering of errors

Can we use *design-induced variation* to **find and use common-case latency at low cost**?
Presentation Outline

• DRAM Background

• Experimental Study of Design-Induced Variation
  – **Goal**: Understand, identify inherently slower regions
  – **Methodology**: Profile 96 real DRAM modules by using FPGA-based DRAM test infrastructure

• Exploiting Design-Induced Variation
  – **DIVA Profiling**: Using low-cost slow region profiling to reliably and dynamically reduce DRAM latency
  – **DIVA Shuffling**: Using variation across data bursts to reduce uncorrectable errors
Challenges of Lowering Latency

- **Static DRAM latency** (e.g., AL-DRAM [HPCA 2015])
  - DRAM vendors need to provide **fixed** timings, increasing testing costs
  - Doesn’t account for **latency changes** over time (e.g., aging and wear out)

- **Conventional online profiling**
  - Takes long time (**high cost**) to profile all DRAM cells

**Our Goal:**
Use design-induced variation to minimize profiling
1. **DIVA Profiling**

Design-Induced-Variation-Aware

Profile *only slow regions* to determine latency

Profile inherently slow
What About Process Variation?

Design-Induced-Variation-Aware

slow cells
process variation
random error

inherently slow
architectural variation
localized error
What About Process Variation?

Design-Induced-Variation-Aware

slow cells
process variation
random error

inherently slow
architectural variation
localized error

error-correcting codes (ECC)
online profiling

Combine ECC & online profiling
→ Reliably reduce DRAM latency at low cost
Correction with Conventional ECC

Process: 8-bit data bus per chip
Memory: 64-bit data bus in memory channel

Read request

Error-Correcting Code (ECC)
Challenge of Conventional ECC

Processor

DIMM

Uncorrectable by ECC

Uncorrectable by ECC

error
Challenge of Conventional ECC

Clusters of slow cells due to design-induced variation lead to more uncorrectable errors
2. DIVA Shuffling

Design-Induced-Variation-Aware Processor

Shuffle data bursts

DIMM
2. **DIVA Shuffling**

**Design-Induced-Variation-Aware Processor**

![Diagram showing shuffle data bursts and reduce uncorrectable errors]

*Shuffle data bursts → *Reduce uncorrectable errors*
2. **DIVA Shuffling**

**Design-Induced-Variation-Aware**

Processor  

DIMM

How do DIVA Profiling and DIVA Shuffling perform?
DIVA Shuffling Improves ECC

DIVA Shuffling uses architectural variation to improve error correction using the same codeword.
DIVA-DRAM Reduces Latency

**Read**

<table>
<thead>
<tr>
<th>Temperature</th>
<th>AL-DRAM</th>
<th>DIVA Profiling</th>
<th>DIVA Profiling + Shuffling</th>
</tr>
</thead>
<tbody>
<tr>
<td>55°C</td>
<td>31.2%</td>
<td>35.1%</td>
<td>36.6%</td>
</tr>
<tr>
<td>85°C</td>
<td>25.5%</td>
<td>34.6%</td>
<td>35.8%</td>
</tr>
</tbody>
</table>

**Write**

<table>
<thead>
<tr>
<th>Temperature</th>
<th>AL-DRAM</th>
<th>DIVA Profiling</th>
<th>DIVA Profiling + Shuffling</th>
</tr>
</thead>
<tbody>
<tr>
<td>55°C</td>
<td>36.6%</td>
<td>39.4%</td>
<td>41.3%</td>
</tr>
<tr>
<td>85°C</td>
<td>27.5%</td>
<td>38.7%</td>
<td>40.3%</td>
</tr>
</tbody>
</table>
DIVA-DRAM reduces latency more aggressively and uses ECC to correct random slow cells.

How do DRAM latency reductions translate to system performance?
DIVA-DRAM Improves Performance

- 32 single-core benchmarks: SPEC, Stream, TPC, GUPS
- 96 multicore workloads constructed with benchmarks

**System Performance Improvement**

- 1-core: 7.0% (AL-DRAM), 9.2% (DIVA Profiling), 9.5% (DIVA Profiling + Shuffling)
- 2-core: 11.7% (AL-DRAM), 14.7% (DIVA Profiling), 15.1% (DIVA Profiling + Shuffling)
- 4-core: 11.0% (AL-DRAM), 13.7% (DIVA Profiling), 14.2% (DIVA Profiling + Shuffling)
- 8-core: 11.5% (AL-DRAM), 13.8% (DIVA Profiling), 14.1% (DIVA Profiling + Shuffling)
DIVA-DRAM Improves Performance

- 32 single-core benchmarks: SPEC, Stream, TPC, GUPS
- 96 multicore workloads constructed with benchmarks

DIVA-DRAM outperforms the best prior work and can adapt to dynamic latency changes
Conclusion

- **Design-Induced Variation**: Inherently slow regions due to DRAM cell array organization

- Analysis: Characterization of 96 real DRAM modules
  - Three types of systematic variation due to design
  - Great potential to reduce DRAM latency at low cost

- Our Approach: **DIVA-DRAM**
  - **DIVA Profiling**: Reliably reduce DRAM latency
    - Profile *only* cells in inherently slow regions
    - Use error correction (ECC) for slow cells that are not profiled
  - **DIVA Shuffling**: Exploit variation to improve ECC reliability

- **15.1%/14.2% higher performance** for 2-/8-core workloads
Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee\textsuperscript{1,2} Samira Khan\textsuperscript{3} Lavanya Subramanian\textsuperscript{2} Saugata Ghose\textsuperscript{2} Rachata Ausavarungnirun\textsuperscript{2} Gennady Pekhimenko\textsuperscript{4} Vivek Seshadri\textsuperscript{4} Onur Mutlu\textsuperscript{5,2}

\textsuperscript{1}NVIDIA \hspace{1cm} \textsuperscript{2}Carnegie Mellon University \hspace{1cm} \textsuperscript{3}University of Virginia \hspace{1cm} \textsuperscript{4}Microsoft Research \hspace{1cm} \textsuperscript{5}ETH Zürich

Data, Circuit Model Will Be Available at https://github.com/CMU-SAFARI/DIVA-DRAM
Backup Slides
Hierarchical Organization of DRAM

(a) Chip (8 banks)

(b) Bank

(c) Mat (cell array)
Sending Data From a DRAM Chip

Data in a request transferred as \textbf{multiple data bursts}
DRAM Stores Data as Charge

Three steps of charge movement

DRAM cell

Sense amplifier
DRAM Stores Data as Charge

Three steps of charge movement

1. Sensing
DRAM Stores Data as Charge

Three steps of charge movement

1. Sensing
2. Restore
DRAM Stores Data as Charge

Three steps of charge movement

1. Sensing
2. Restore
3. Precharge
DRAM Charge over Time

- Cell
- Sense amplifier
- Charge
- Data 0
- Data 1
- Time
DRAM Charge over Time

- Cell
- Sense amplifier

Charge over time graph with Data 0 and Data 1.
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 cells that can store 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

Real

---

SAFARI
DRAM Cells are Not Equal

Large variation in cell size ➔
Large variation in charge ➔
Large variation in access latency
Two Reasons for Timing Margin

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

2. Temperature Dependence
   - DRAM leaks more charge at higher temperature
   - Leads to extra timing margin when operating at low temperature
Charge Leakage $\propto$ Temperature

Room Temp.  Small leakage

Hot Temp. (85°C)  Large leakage
Cells store small charge at high temperature and large charge at low temperature.

\[ \Rightarrow \text{Large variation in access latency} \]
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
  ➔ Can lower latency for the common case
DRAM Timing Parameters

Command: ACTIVATE

Data

Duration
DRAM Timing Parameters

Activation latency: tRCD
(13ns / 50 cycles)

Command

Data

Duration

Cache line (64B)
DRAM Timing Parameters

1. Activation latency: tRCD
   (13ns / 50 cycles)

2. Precharge latency: tRP
   (13ns / 50 cycles)
Design-Induced Variation in Open Bitline DRAM
Challenge: External ≠ Internal

External address → IO interface → Address mapping → Internal address

External address ≠ Internal address
DRAM-Internal vs. DRAM-External

Estimated Mapping (External → Internal)
Based on Error Counts for the External Address
Row Address Mapping Confidence

Confidence

IntMSB

IntLSB
1.1. Measuring Row Variation

Lower t\textsubscript{RP} (precharge timing parameter) to 7.5 ns

![Graph showing periodic errors]

- **Errors**: Periodic Errors
- **x-axis**: Row address (mod. 512)
- **y-axis**: Erroneous Request Count

*SAFARI*
1.1. Measuring Row Variation

Lower \( t_{RP} \) (precharge timing parameter) to 7.5 ns

Need to reverse engineer row address mapping details in our paper
1.2. Periodic Row Variation Behavior

Sorting with Discovered Row Mapping

Row error (latency) characteristics periodically repeat every 512 rows
1.1. Variation in Rows

$t_{RP}$ (precharge timing parameter)

**Random Errors**

**Periodic Errors**

**Mostly Errors**
1.2. Periodic Row Variation Behavior

Error (latency) characteristics periodically repeat every 512 rows
2. Variation Across Columns
2. Variation Across Columns

Different columns \(\rightarrow\) data from different locations
\(\rightarrow\) different characteristics
3. Variation in Data Bits

Data in a request transferred as multiple data bursts
DIVA-DRAM Evaluation Methodology

Modified version of Ramulator
(cycle accurate DRAM simulator)

<table>
<thead>
<tr>
<th>Component</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>8 cores, 3.2GHz, 3-wide issue,</td>
</tr>
<tr>
<td></td>
<td>8 MSHRs/core, 128-entry inst. window</td>
</tr>
<tr>
<td>Last-level cache</td>
<td>64B cache-line, 16-way associative,</td>
</tr>
<tr>
<td></td>
<td>512KB private cache-slice per core</td>
</tr>
<tr>
<td>Mem. Controller</td>
<td>64/64-entry read/write queues, FR-FCFS [77, 104]</td>
</tr>
<tr>
<td>Memory system</td>
<td>DDR3-1600 [31], 2 channels, 2 ranks-per-channel</td>
</tr>
</tbody>
</table>
local sense amplifier

parasitic resistance & capacitance

cell capacitor

access transistor

local wordline driver

wordline

512 rows (cells per bitline)

512 columns (cells per wordline)

global wordline driver

mat

mat

local sense amplifier

bitline

512 columns (cells per wordline)
charge sharing

sense amplification

ready to access level (0.9 V)

activation

precharge

Voltage (V)

0 0.3 0.6 0.9 1.2

Time (ns)

0 10 20 30 40

D

faster

E

faster

voltage of bitline on cell ②: near wordline driver

voltage of bitline on cell ③: far from wordline driver