QUAC-TRNG

High-Throughput True Random Number Generation Using Quadruple Row Activation in Real DRAM Chips

Ataberk Olgun
Minesh Patel  A. Giray Yağlıkçı  Haocong Luo
Jeremie S. Kim  F. Nisa Bostancı  Nandita Vijaykumar
Oğuz Ergin  Onur Mutlu

SAFARI  kasırga

ETH zürich  University of Economics & Technology
TOBB ETÜ  UNIVERSITY OF TORONTO
Executive Summary

- **Motivation**: DRAM-based true random number generators (TRNGs) provide true random numbers at low cost on a wide range of computing systems.

- **Problem**: Prior DRAM-based TRNGs are slow:
  1. Based on fundamentally slow processes → high latency
  2. Cannot effectively harness entropy from DRAM rows → low throughput

- **Goal**: Develop a high-throughput and low-latency TRNG that uses commodity DRAM devices.

- **Key Observation**: Carefully engineered sequence of DRAM commands can activate four DRAM rows → QUadruple ACtivation (QUAC).

- **Key Idea**: Use QUAC to activate DRAM rows that are initialized with conflicting data (e.g., two ‘1’s and two ‘0’s) to generate random values.

- **QUAC-TRNG**: DRAM-based TRNG that generates true random numbers at high-throughput and low-latency by repeatedly performing QUAC operations.

- **Results**: We evaluate QUAC-TRNG using 136 real DDR4 chips:
  1. 5.4 Gb/s maximum (3.4 Gb/s average) TRNG throughput per DRAM channel
  2. Outperforms existing DRAM-based TRNGs by 15.08x (base), and 1.41x (enhanced)
  3. Low TRNG latency: 256-bit RN in 274 ns
  4. Passes all 15 NIST randomness tests.
Outline

True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACtivation (QUAC)

QUAC-TRNG

Evaluation
Use Cases of True Random Numbers

High-quality true random numbers are critical to many applications.

True random numbers can only be obtained by sampling random physical processes.

Unfortunately, not all computing systems are equipped with TRNG hardware (e.g., dedicated circuitry).
DRAM-Based TRNGs

**DRAM chips are ubiquitous** in modern computing platforms

DRAM-based TRNGs enable true random number generation **within DRAM chips**

*Low-cost:* No specialized circuitry for RNG
  - Beneficial for constrained systems

*High-throughput:* > Gb/s throughput
  - Open application space that require high-throughput TRNG
Synergy with Processing-in-Memory

Processing-in-Memory (PIM) Systems

• Perform computation directly within a memory chip
• Improve system performance by avoiding off-chip data movement

True random number generation within DRAM

• Enables PIM workloads to sample true random numbers directly within the memory chip
• Avoids inefficient communication to other possible off-chip TRNG sources, enhances security & privacy
True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACtivation (QUAC)

QUAC-TRNG

Evaluation
DRAM Organization
Accessing a DRAM Cell

- **wordline**
- **capacitor**
- **access transistor**
- **enable**
- **Sense Amp**
- **bitline**
Accessing a DRAM Cell

1. Enable wordline
2. Connects cell to bitline
3. Cell loses charge to bitline
4. Deviation in bitline voltage
5. Enable sense amp
6. Cell loses charge to bitline

$V_{DD}$ $V_{DD} + \delta$

$\frac{1}{2}V_{DD}$
DRAM Operation

DRAM Command Sequence

ACT R0  RD  RD  RD  PRE R0  ACT R1  RD  RD  RD  RD

$tRAS$ (Activation Latency)
$tRP$ (Precharge Latency)

[Kim+ HPCA'19]
Outline

True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACTivation (QUAC)

QUAC-TRNG

Evaluation
Quadruple Activation (QUAC)

New Observation

Carefully-engineered DRAM commands can activate four rows in real DRAM chips

Activate four rows with two ACT commands
Quadruple Activation (QUAC)

**Characteristic 1**

Activates a set of four DRAM rows whose addresses **differ only** in their **two LSBs**

<table>
<thead>
<tr>
<th>Row</th>
<th>Address</th>
<th>DRAM Rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>111</td>
<td>![Row 7 DRAM Rows]</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>![Row 6 DRAM Rows]</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>![Row 5 DRAM Rows]</td>
</tr>
<tr>
<td>3</td>
<td>001</td>
<td>![Row 3 DRAM Rows]</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>![Row 2 DRAM Rows]</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>![Row 1 DRAM Rows]</td>
</tr>
<tr>
<td>0</td>
<td>000</td>
<td>![Row 0 DRAM Rows]</td>
</tr>
</tbody>
</table>
**Quadruple Activation (QUAC)**

**Characteristic 2**

First and second ACT's addresses must have their two LSBs inverted.

<table>
<thead>
<tr>
<th>ACT</th>
<th>PRE</th>
<th>ACT</th>
<th>PRE</th>
<th>ACT</th>
<th>PRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>011</td>
<td>010</td>
<td>001</td>
<td>000</td>
<td>001</td>
</tr>
</tbody>
</table>
QUAC on Real DRAM Chips

Valid QUAC behavior on 136 DDR4 chips
Why Does QUAC Work?

Hypothetical circuit to explain QUAC

• High *density* and *performance* requirements push for *hierarchical* organization

• Hierarchical organization of DRAM wordlines enable *high-density* and *low-latency* DRAM operation
Hierarchical Wordlines

A master wordline drives multiple local wordlines
Hierarchical Wordlines

A master wordline drives multiple local wordlines

Select signals for rows 0-3
Hypothetical Row Decoder

Predecode the least significant two bits

Drive control signals

Address [0]

Address [1]

Latch

A0

A0

A1

A1

S0

S1

S2

S3
First ACT command drives a single wordline
Hypothetical Row Decoder

Command | ACT R0 => PRE

PRE command cannot disable latches
Hypothetical Row Decoder

Command | ACT R0 | PRE | ACT R3

0

Second ACT drives the **remaining three** wordlines
Hypothetical Row Decoder

Command | ACT R0 | Violate Timing | PRE | Violate Timing | ACT R3

1

1

All four wordlines are enabled
Quadruple Activation
Outline

True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACtivation (QUAC)

QUAC-TRNG

Evaluation
Generating Random Values via QUAC

Voltage Difference

- \( V_{TH} \)
- 0
- \(-V_{TH} \)

Ready to Sense Voltage Level

Time

Sense Amplifier

Enable

Logic-1

\( V_{DD}/2 \)

\( V_{DD}/2 \)

\( V_{DD}/2 \)
Generating Random Values via QUAC

\[ V_{\text{DD}}/2 + \varepsilon \]

R3

R2

R1

R0

Sense Amplifier

Enable

Voltage Difference

\[ V_{\text{dd}}^- \]

\[ V_{\text{TH}} \]

\[ 0 \]

\[ -V_{\text{TH}} \]

\[ -V_{\text{dd}} \]

ACT R0

PRE

ACT R3

Time
Generating Random Values via QUAC

Voltage Difference

$V_{dd}^-$

$V_{TH}$

$0$

$-V_{TH}$

$-V_{dd}$

ACT

R0

PRE

ACT

R3

Enable

Sense Amplifier

Enable Sense Amplifiers

Random perturbation

$V_{dd}/2 + \varepsilon$

R3

R2

R1

R0

$V_{dd}/2$

$V_{DD}$

$V_{DD}/2$

$V_{DD}$
**Key Idea:** Leverage random values on sense amplifiers generated by QUAC operations as source of entropy.
**QUAC-TRNG**

**Key Idea:** Leverage random values on sense amplifiers generated by QUAC operations as source of entropy

1. **Step 1** Initialize
2. **Step 2** QUAC
3. **Step 3** Read
4. **Step 4** Post-process

![Diagram of QUAC-TRNG process](image)
Outline

True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACtivation (QUAC)

QUAC-TRNG

Evaluation
Real Chip Characterization

Experimentally study QUAC and QUAC-TRNG using 136 real DDR4 chips from SK Hynix

**DDR4 DRAM Bender** → DRAM Testing Infrastructure

---

**b. FPGA Board**

**a. DRAM Module**

**c. PCIe Host Interface**

**d. Temperature Controller**

[SAFARI Kasirga [Olgun+ TCAD’23] https://github.com/CMU-SAFARI/DRAM-Bender]
Real Chip Characterization

Measure randomness of bitstreams using Shannon Entropy

\[ H(x) = - \sum_{i=1}^{2} p(x_i) \log_2 p(x_i) \]

Calculating probabilities:
Proportion of \textit{logic-1} and \textit{logic-0} values in the random bitstream

Sample each bitline following QUAC \textit{1000 times}
and calculate the bitline’s Shannon Entropy

\[
\text{SE}(1111...111) = 0 \\
0 < \text{SE}(1001...010) < 1
\]
Real Chip Characterization

At 50°C and nominal voltage:

Repeatedly perform QUAC 1000 times and measure the Shannon Entropy of each bitline in 8K DRAM Segments (32K DRAM Rows), using all 16 different four-bit data patterns.

Data pattern: 1111 (four ones)

R3
R2
R1
R0

Data pattern: 1000

0 R3
0 R2
0 R1
1 R0
Data Pattern Dependence

Calculate *cache block entropy (CBE)*

\[ \sum \text{all bitline entropies in the cache block} \]

Metrics based on CBE:

1. **Average CBE**: Average entropy across all cache blocks in a module

2. **Maximum CBE**: Maximum of the cache block entropies in a module
Data Pattern Dependence

Cache Block Entropy

Average CB Entropy
Maximum CB Entropy

Entropy varies with data pattern

Highest average entropy with pattern “0111”
Segment entropy behavior is different for different modules

Segment entropy = \[ \sum \text{all bitline entropies in the segment} \]
Segment entropy = \sum \text{all bitline entropies in the segment}

Entropy significantly increases towards the end of the DRAM bank
Takeaways: QUAC Entropy

We observe that **entropy resulting from QUAC operations changes** according to the

- **data pattern** used in initialization
- **physical location** of DRAM segments

attributed to:

- **systematic manufacturing process variation**
- **design-induced variation**
Two experiments to measure quality

1. QUAC + lightweight post-processing
   To see if QUACs produce high-quality random values on sense amplifiers

2. QUAC-TRNG using SHA-256
   To evaluate QUAC-TRNG’s quality

Use NIST STS to evaluate quality
# NIST Results

## Table 1: NIST STS Randomness Test Results

<table>
<thead>
<tr>
<th>NIST STS Test</th>
<th>VNC* (p-value)</th>
<th>SHA-256 (p-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>monobit</td>
<td>0.430</td>
<td>0.500</td>
</tr>
<tr>
<td>frequency_within_block</td>
<td>0.408</td>
<td>0.528</td>
</tr>
<tr>
<td>runs</td>
<td>0.335</td>
<td>0.558</td>
</tr>
<tr>
<td>longest_run_ones_in_a_block</td>
<td>0.564</td>
<td>0.533</td>
</tr>
<tr>
<td>binary_matrix_rank</td>
<td>0.554</td>
<td>0.548</td>
</tr>
<tr>
<td>dft</td>
<td>0.538</td>
<td>0.364</td>
</tr>
<tr>
<td>non_overlapping_template_matching</td>
<td>&gt;0.999</td>
<td>0.488</td>
</tr>
<tr>
<td>overlapping_template_matching</td>
<td>0.513</td>
<td>0.410</td>
</tr>
<tr>
<td>maurers_universal</td>
<td>0.493</td>
<td>0.387</td>
</tr>
<tr>
<td>linear_complexity</td>
<td>0.483</td>
<td>0.559</td>
</tr>
<tr>
<td>serial</td>
<td>0.355</td>
<td>0.510</td>
</tr>
<tr>
<td>approximate_entropy</td>
<td>0.448</td>
<td>0.539</td>
</tr>
<tr>
<td>cumulative_sums</td>
<td>0.356</td>
<td>0.381</td>
</tr>
<tr>
<td>random_excursion</td>
<td>0.164</td>
<td>0.466</td>
</tr>
<tr>
<td>random_excursion_variant</td>
<td>0.116</td>
<td>0.510</td>
</tr>
</tbody>
</table>

*VNC: Von Neumann Corrector*
# NIST Results

## Table 1: NIST STS Randomness Test Results

<table>
<thead>
<tr>
<th>NIST STS Test</th>
<th>VNC* (p-value)</th>
<th>SHA-256 (p-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>monobit</td>
<td>0.430</td>
<td>0.500</td>
</tr>
<tr>
<td>frequency_within_block</td>
<td>0.408</td>
<td>0.528</td>
</tr>
<tr>
<td>runs</td>
<td>0.335</td>
<td>0.558</td>
</tr>
<tr>
<td>longest_run_ones_in_a_block</td>
<td>0.564</td>
<td>0.533</td>
</tr>
<tr>
<td>binary_matrix_rank</td>
<td>0.554</td>
<td>0.548</td>
</tr>
<tr>
<td>dft</td>
<td>0.538</td>
<td>0.364</td>
</tr>
<tr>
<td>non_overlapping_template_matching</td>
<td>&gt;0.999</td>
<td>0.488</td>
</tr>
<tr>
<td>overlapping_template_matching</td>
<td>0.513</td>
<td>0.410</td>
</tr>
</tbody>
</table>

QUAC and QUAC-TRNG bitstreams pass all 15 NIST randomness tests

| random_excursion                                   | 0.164          | 0.466             |
| random_excursion_variant                           | 0.116          | 0.510             |

*VNC: Von Neumann Corrector
QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG’s throughput according to:

\[
(256 \times \text{SIB})/(L \times 10^{-9}) \text{ bps}
\]

**SIB:** # of SHA Input Blocks in the highest-entropy segment

**L:** Latency of one QUAC operation in nanoseconds
QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG’s throughput according to:

\[
(256 \times SIB) / (L \times 10^{-9}) \text{ bps}
\]

**SIB:** # of SHA Input Blocks in the highest-entropy segment

**L:** Latency of one QUAC operation in nanoseconds

\[\sum \text{cache block entropy} = 256\]
QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG’s throughput according to:

\[
(256 \times SIB) / (L \times 10^{-9}) \text{ bps}
\]

**SIB**: # of SHA Input Blocks in the highest-entropy segment

**L**: Latency of one QUAC operation in nanoseconds
QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG’s throughput according to:

\[
\frac{256 \times SIB}{L \times 10^{-9}} \text{ bps}
\]

**SIB**: # of SHA Input Blocks in the highest-entropy segment

**L**: Latency of one QUAC operation in nanoseconds
QUAC-TRNG Configurations

1. **One Bank**
   - Use a single DRAM bank

2. **BGP**
   - Bank Group-Level Parallelism
   - Use four banks from different bank groups

3. **RC + BGP**
   - RowClone + BGP
   - Use in-DRAM copy to initialize DRAM rows and use four banks from different bank groups

[Diagram showing DRAM subarray, source row, destination row, activate source, precharge, activate destination]

[Seshadri+ MICRO’13]  [Gao+ MICRO’19]
QUAC-TRNG Throughput

Achieves 3.44 Gb/s throughput per DRAM channel on average across all modules.

In-DRAM initialization greatly improves throughput.
QUAC-TRNG vs State-Of-The-Art

High-throughput DRAM-based TRNGs:

- **D-RaNGe**: Activation latency failures
- **Talukder et. al**: Precharge latency failures

Calculate throughput by *tightly scheduling the DDR4 commands* required to induce failures

Evaluate two versions of these past two works:

- **Base**: As proposed
- **Enhanced (Fair)**: Throughput-optimized (SHA-256)

Assume four-channel DDR4 memory

[Kim+, HPCA'19] [Talukder+, IEEE Access 2019]
QUAC-TRNG vs State-Of-The-Art

Outperforms best prior DRAM-based TRNG
(i) “base” by 15.08x at 2.4 GT/s
(ii) “enhanced” by 2.03x at 12 GT/s
More in the Paper

• NIST randomness tests results

• Throughput & latency comparison against four other DRAM-based TRNGs

• System Integration
  - How QUAC-TRNG can be implemented in real systems
  - System performance study
    • QUAC-TRNG’s throughput with concurrently running applications
  - Area overhead: 0.04% of a contemporary CPU (7 nm)
  - Memory overhead: 0.002% of an 8 GiB DRAM module

• Sensitivity Analysis
  - Effect of temperature on QUAC’s entropy
    • Entropy changes with temperature
  - Time dependence study
    • Entropy remains stable for at least up to a month
Executive Summary

- **Motivation**: DRAM-based true random number generators (TRNGs) provide true random numbers at low cost on a wide range of computing systems.

- **Problem**: Prior DRAM-based TRNGs are slow:
  1. Based on fundamentally slow processes → high latency
  2. Cannot effectively harness entropy from DRAM rows → low throughput

- **Goal**: Develop a high-throughput and low-latency TRNG that uses commodity DRAM devices.

- **Key Observation**: Carefully engineered sequence of DRAM commands can activate four DRAM rows → QUadruple ACTivation (QUAC).

- **Key Idea**: Use QUAC to activate DRAM rows that are initialized with conflicting data (e.g., two ‘1’s and two ‘0’s) to generate random values.

- **QUAC-TRNG**: DRAM-based TRNG that generates true random numbers at high-throughput and low-latency by repeatedly performing QUAC operations.

- **Results**: We evaluate QUAC-TRNG using 136 real DDR4 chips:
  1. 5.4 Gb/s maximum (3.4 Gb/s average) TRNG throughput per DRAM channel
  2. QUAC-TRNG has low TRNG latency: 256-bit RN in 274 ns
  3. Outperforms existing DRAM-based TRNGs by 15.08x (base), and 1.41x (enhanced)
  4. QUAC-TRNG passes all 15 NIST randomness tests.
QUAC-TRNG

High-Throughput True Random Number Generation Using Quadruple Row Activation in Real DRAM Chips

Ataberk Olgun
Minesh Patel A. Giray Yağlıkçı Haocong Luo
Jeremie S. Kim F. Nisa Bostancı Nandita Vijaykumar
Oğuz Ergin Onur Mutlu

Hasan Hassan, Ataberk Olgun, A. Giray Yaglikci, Haocong Luo, Onur Mutlu


2023 EDAA Outstanding Dissertation Award
Self-Managing DRAM: Problem

The Memory Controller manages DRAM maintenance operations

Memory Controller

- DRAM Refresh
- RowHammer Protection
- Memory Scrubbing

Changes to maintenance operations are often reflected to the memory controller design, DRAM interface, and other system components

Implementing new maintenance operations (or modifying the existing ones) is difficult-to-realize
A Prime Example: New Features of DDR5

**DRAM Refresh**

**Same Bank Refresh** - simultaneously refreshes one bank in each bank group

The new **REFsb** command requires changes in DRAM interface and memory controller.

**RowHammer Protection**

**Refresh Management (RFM)** – memory controller issues the new RFM command to allow DRAM chips more time to perform victim row refresh

The new **RFM** command requires changes in DRAM interface and memory controller.

**Memory Scrubbing**

**In-DRAM Scrubbing** – DDR5 uses the on-die ECC to perform periodic scrubbing

The new **scrub** command requires changes in DRAM interface and memory controller.

**DDR5** changes are **difficult-to-implement** as they were **only** possible after **multiple years** required to develop a new DRAM standard.
Self-Managing DRAM (SMD) enables autonomous in-DRAM maintenance operations.

**Key Idea:**

Prevent the memory controller from accessing DRAM regions that are *under maintenance* by *rejecting* row activation (ACT) commands.

Leveraging the ability to *reject an ACT*, a *maintenance operation* can be implemented *completely* within a DRAM chip.
A DRAM bank is divided into configurable-size Lock Regions.
SMD marks regions locked for maintenance in the Lock Region Table (LRT).
SMD rejects any ACT command that targets a locked region by sending the memory controller an ACT_NACK signal.
SMD: High-Level Operation

LRT

0 - locked
1 - available
2 - available
3 - available

ACT

ACT_NACK

T_{ACK_NACK}

ACT Retry Interval (ARI)

bank 0
lock region 0

bank 0
lock region 1

tRCD

tRAS

ACT

RD

PRE

ACT

LRT

0 - available
1 - available
2 - available
3 - available

time
# SMD-Based Maintenance Mechanisms

<table>
<thead>
<tr>
<th>DRAM Refresh</th>
<th>Fixed Rate (SMD-FR)</th>
<th>Variable Rate (SMD-VR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform refreshes all DRAM rows with a fixed refresh period</td>
<td>skips refreshing rows that can retain their data for longer than the default refresh period</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RowHammer Protection</th>
<th>Probabilistic (SMD-PRP)</th>
<th>Deterministic (SMD-DRP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performs neighbor row refresh with a small probability on every row activation</td>
<td>keeps track of most frequently activated rows and performs neighbor row refresh when activation count threshold is exceeded</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory Scrubbing</th>
<th>Periodic Scrubbing (SMD-MS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>periodically scans the entire DRAM for errors and corrects them</td>
<td></td>
</tr>
</tbody>
</table>
Methodology

• Simulators
  • Ramulator [Kim+, CAL’15]
    https://github/CMU-SAFAI/ramulator
  • DRAMPower [Chandrasekar+, DSD’11]
    https://github.com/tukl-msd/DRAMPower

• Workloads
  • 44 single-core workloads
    SPEC CPU2006, TPC, STREAM, MediaBench
  • 60 multi-programmed four-core workloads
    By randomly choosing from single-core workloads

• System Parameters
  • 4-channel dual-rank DDR4 DRAM
  • 32ms default refresh period
Single-Core Results

![Speedup Chart]

- SMD-FR
- SMD-VR
- SMD-PRP
- SMD-DRP
- SMD-MS
- SMD-Combined
- NoRefresh

![Normalized Energy Chart]

- SMD-FR
- SMD-VR
- SMD-PRP
- SMD-DRP
- SMD-MS
- SMD-Combined
- NoRefresh

Speedup and Normalized Energy comparisons for different configurations.
SMD-based maintenance mechanisms have significant performance and energy efficiency benefits.
Sensitivity to Refresh Period

<table>
<thead>
<tr>
<th>Refresh Period</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>64ms</td>
<td>SMD-FR 19.5%</td>
</tr>
<tr>
<td></td>
<td>SMD-VR 51.5%</td>
</tr>
<tr>
<td></td>
<td>SMD-Combined 58.4%</td>
</tr>
<tr>
<td></td>
<td>NoRefresh</td>
</tr>
<tr>
<td>32ms</td>
<td>SMD-FR 195%</td>
</tr>
<tr>
<td></td>
<td>SMD-VR 225%</td>
</tr>
<tr>
<td></td>
<td>SMD-Combined</td>
</tr>
<tr>
<td></td>
<td>NoRefresh</td>
</tr>
<tr>
<td>16ms</td>
<td>SMD-FR 195%</td>
</tr>
<tr>
<td></td>
<td>SMD-VR 225%</td>
</tr>
<tr>
<td></td>
<td>SMD-Combined</td>
</tr>
<tr>
<td></td>
<td>NoRefresh</td>
</tr>
<tr>
<td>8ms</td>
<td>SMD-FR 195%</td>
</tr>
<tr>
<td></td>
<td>SMD-VR 225%</td>
</tr>
<tr>
<td></td>
<td>SMD-Combined 51.5%</td>
</tr>
<tr>
<td></td>
<td>NoRefresh 58.4%</td>
</tr>
</tbody>
</table>
Hardware Overhead

Interface Modifications
• A single ACT_NACK pin per DRAM chip or repurpose the existing alert_n signal

DRAM Chip Modifications
• Lock Region Table incurs only:
  • 32um2 area overhead (0.001% of a 45.4mm2 DRAM chip)
  • 0.053ns access latency overhead

Memory Controller Modifications
• Changes in request scheduling to handle ACT_NACK signals

• No further changes needed for new maintenance operations
• The memory controller no longer manages DRAM maintenance
Other Results in the Paper

- Lock region size sensitivity
- Comparison to memory controller-based RH protection
- Comparison to memory controller-based scrubbing
- SMD-DRP maximum activation threshold sensitivity
- Victim row window sensitivity
Summary

实施新的维护机制通常要求难以实现的改变。

三大部分的DRAM维护操作如下：
- 刷新 (Refresh)
- 行锤保护 (RowHammer Protection)
- 内存刮洗 (Memory Scrubbing)

Our Goal

1. 使使能新DRAM维护操作的过程更简单
2. 使能更高效的DRAM维护操作

Self-Managing DRAM (SMD)

SMD使能实现新的在DRAM中的维护机制，无需进一步的更改

SMD-based refresh, RowHammer protection, and scrubbing achieve 9.2% speedup and 6.2% lower DRAM energy vs. conventional DRAM

Hasan Hassan, Ataberk Olgun, A. Giray Yaglikci, Haocong Luo, Onur Mutlu


2023 EDAA Outstanding Dissertation Award
Backup Slides
## DDR4 Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Module Identifier</th>
<th>Chip Identifier</th>
<th>Freq. (MT/s)</th>
<th>Organization</th>
<th>Segment Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Size (GB)</td>
<td>Chips</td>
</tr>
<tr>
<td>M1</td>
<td>Unknown</td>
<td>H5AN4G8NAFR-TFC</td>
<td>2133</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M2</td>
<td>Unknown</td>
<td>Unknown</td>
<td>2133</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M3</td>
<td>Unknown</td>
<td>H5AN4G8NAFR-TFC</td>
<td>2133</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M4</td>
<td>76TT21NUS1R8-4G</td>
<td>H5AN4G8NAFR-TFC</td>
<td>2133</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M5</td>
<td>Unknown</td>
<td>T4D5128HT-21</td>
<td>2133</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M6</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M7</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M8</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M9</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M10</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M11</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M12</td>
<td>TLRD44G2666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M13</td>
<td>KSM32RD8/16HDR</td>
<td>H5AN4G8NAFA-UHC</td>
<td>2400</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M14</td>
<td>F4-2400C17S-8GNT</td>
<td>H5AN4G8NMFR-UHC</td>
<td>2400</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>M15</td>
<td>F4-2400C17S-8GNT</td>
<td>H5AN4G8NMFR-UHC</td>
<td>3200</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>M16</td>
<td>KSM32RD8/16HDR</td>
<td>H5AN8G8NDJR-XNC</td>
<td>3200</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>M17</td>
<td>KSM32RD8/16HDR</td>
<td>H5AN8G8NDJR-XNC</td>
<td>3200</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

†The maximum possible entropy in a DRAM segment is 64K (65,536) bits.
Random Numbers

Sequence of bits without any discernible pattern

Random Number Candidates

00001111 Pattern exists

01010101 Pattern exists

00101101 Pattern does not seem to exist
Producing Random Numbers

Random Number Generator (RNG)
A device/program that generates random numbers

Pseudo-Random Number Generator (PRNG)
- Arithmetic Transform

True Random Number Generator (TRNG)
- Physical Process

Seed → State → 00101101
If seed is compromised, random number sequence can be regenerated

TRNG output cannot be regenerated from the physical process
Assessing the Quality of a TRNG

Systematically distinguish between a low-quality and a high-quality TRNG

Statistical Tests

01010101
Occurrence of N-bit patterns
Min-entropy
FAIL

00101101
Proportion of logic-1 bits
PASS

...
Use Cases of True Random Numbers

High-quality true random numbers are critical to many applications

- Cryptographic Key Generation
- Signature Generation
- Monte Carlo Simulations
- Randomized Training
- Genetic Algorithms
True random numbers can **only** be obtained by sampling random physical processes.

Unfortunately, **not all computing systems** are equipped with **TRNG hardware** (e.g., dedicated circuitry).
DRAM-Based TRNGs

DRAM chips are ubiquitous in modern computing platforms

DRAM-based TRNGs enable true random number generation within DRAM chips
**DRAM-Based TRNGs**

*Low-cost:* No specialized circuitry for RNG
Beneficial for constrained systems

*High-throughput:* > Gb/s throughput
Enable applications that require high-throughput TRNG
Synergy with Processing-in-Memory

Processing-in-Memory (PIM) Systems

• Perform computation directly within a memory chip
• Improve system performance by avoiding off-chip data movement

True random number generation within DRAM

• Enables PIM workloads to sample true random numbers directly within the memory chip
• Avoids inefficient communication to other possible off-chip TRNG sources, enhances security & privacy
Figure 2: Timeline of key DDR4 commands.
Figure 6: QUAC-TRNG mechanism.
Sample random physical phenomena
Bias in entropy source $\Rightarrow$ bias in TRNG output

- Low quality (statistically) random numbers

Post-processing to remove bias

- Improve TRNG quality
Cryptographic Hash Functions

Arithmetic Operations (shuffling, mixing)

Von Neumann Corrector

Von Neumann Corrector

<table>
<thead>
<tr>
<th>Input Bits</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>none</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>none</td>
</tr>
</tbody>
</table>
Entropy Sources in DRAM

Retention Failures

Fundamentally Slow: cells leak charge slowly

capacitor
Entropy Sources in DRAM

1. **Retention Failures**
   - Fundamentally Slow: cells leak charge slowly

2. **Start-up Values**
   - Fundamentally Slow: requires power-cycle

3. **Timing Failures**
   - Fast: limited by DRAM command latencies
Entropy Sources in DRAM

1. **Retention Failures**
   - Fundamentally Slow: cells leak charge slowly

2. **Start-up Values**
   - Fundamentally Slow: requires power-cycle

3. **Timing Failures**
   - Fast: limited by DRAM command latencies
Timing Failure-based TRNGs

1. D-RaNGe
   Activation latency (tRCD) failures
   
   J. S. Kim, M. Patel, H. Hassan, L. Orosa and O. Mutlu,
   "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput," HPCA, 2019

2. Talukder et al.
   Precharge latency (tRP) failures
   
   B. M. S. Bahar Talukder, J. Kerns, B. Ray, T. Morris and M. T. Rahman,
   "Exploiting DRAM Latency Variations for Generating True Random Numbers," ICCE, 2019
D-RaNGe: Accessing a Cell

Bitline Voltage

\[ V_{dd} \]

\[ V_{min} \]

\[ 0.5 V_{dd} \]

Process variation during manufacturing results in cells having unique behavior

Guardband

Guardband

Ready to Access Voltage Level

Bitline Charge Sharing

wordline

access transistor

Sense Amplifier

Strong

Weak

Kim+, HPCA 2019
D-RaNGe: Accessing a Cell

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

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

Low % chance to fail with reduced $t_{\text{RCD}}$
The key idea is to extract random values by sampling DRAM cells that fail truly randomly.

Kim+, HPCA 2019
Key Idea: Sample DRAM cells that fail randomly, but with reduced tRP
Limitations of State-of-the-art

D-RaNGe

- Poorly utilizes activation granularity

• Induce failures within 0.8% of cells

• Throughput limited by access latency
Limitations of State-of-the-art

Talukder et al.

• Well utilizes activation granularity
• *Cannot* induce metastability on many sense amplifiers

**DRAM Sense Amplifiers / Row Buffer**

Random bits distributed scarcely over the row buffer

• Throughput *limited by poor-quality randomness source*
Limitations of State-of-the-art

Talukder et al.

• Well utilizes activation granularity
• Cannot induce metastability on many sense amplifiers

A **high-throughput** DRAM TRNG needs to:

(i) well utilize **activation granularity**
(ii) induce metastability on **many sense amplifiers**

• Throughput **limited by poor-quality randomness source**
Spatial Distribution

Cache block entropy is the highest around the middle of the DRAM segment
QUAC-TRNG’s Quality

Collect bitstreams by repeatedly sampling bitlines after QUAC operations

- 1 Mb random bitstreams
- Post-processing: Von Neumann Corrector

Von Neumann Corrector

<table>
<thead>
<tr>
<th>Input Bits</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>none</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>none</td>
</tr>
</tbody>
</table>

Random Number

NIST STS
QUAC-TRNG’s Quality

Collect bitstreams using QUAC-TRNG

• 1 Gb bitstreams
• Post-processing: SHA-256

Diagram:
- DRAM Seg.
- SHA-256
- Random Number
- NIST STS
The maximum throughput QUAC-TRNG provides without reducing the total off-chip memory bandwidth

**Ramulator:** 3.2 GHz core, *four-channel* DDR4 memory

**QUAC-TRNG achieves 74.1% of the empirical average throughput**
### Table 2: Summary of prior DRAM-TRNGs vs QUAC-TRNG

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Entropy Source</th>
<th>TRNG Throughput</th>
<th>256-bit TRNG Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUAC-TRNG</td>
<td>Quadruple ACT</td>
<td>13.76 Gb/s</td>
<td>274 ns</td>
</tr>
<tr>
<td>Talukder+ [15]</td>
<td>Precharge Failure</td>
<td>0.68 - 6.13 Gb/s</td>
<td>249 ns - 201 ns</td>
</tr>
<tr>
<td>D-RaNGe [88]</td>
<td>Activation Failure</td>
<td>0.92 - 9.73 Gb/s</td>
<td>260 ns - 36 ns</td>
</tr>
<tr>
<td>D-PUF [150]</td>
<td>Retention Failure</td>
<td>0.20 Mb/s</td>
<td>40 s</td>
</tr>
<tr>
<td>DRNG [47]</td>
<td>DRAM Start-up</td>
<td>N/A</td>
<td>700 µs</td>
</tr>
<tr>
<td>Keller+ [81]</td>
<td>Retention Failure</td>
<td>0.025 Mb/s</td>
<td>40 s</td>
</tr>
<tr>
<td>Pyo+ [126]</td>
<td>DRAM Cmd Schedule</td>
<td>2.17 Mb/s</td>
<td>112.5 µs</td>
</tr>
</tbody>
</table>

Based on fundamentally slow processes.
Memory Overhead

The memory allocated for QUAC-TRNG is only **192 KiB**s per DRAM module

- **Four segments** for QUAC (128 KiB)
- **Eight DRAM rows** for bulk initialization (64 KiB)
- **0.002% of the capacity of an 8 GB DDR4 DIMM.**
Area Overhead

Require **326 bits of storage per DRAM channel**

- Four DRAM addresses to point to segments (72 bits)
- Eight addresses to point to initialization operands (144 bits)
- 11 column addresses → 256-bit total entropy cache block ranges (110 bits)

Model storage using **CACTI**:

- **0.0003 mm$^2$** for storage
- **0.0011 mm$^2$** for the SHA-256 engine
- **0.04%** of a Zen 2 CPU (7 nm)
Sensitivity Study

Temperature Dependence
Record bitline entropy at 50, 65 and 85°C

Time Dependence
Record entropy at the beginning and the end of a 30-day period

Both studies use 5 DRAM modules and the “0111” data pattern
Temperature Dependence

Observe two temperature dependence trends

Entropy changes with temperature
Entropy difference between the beginning and the end of the 30-day period:

• 2.4% average across all chips
• Maximum difference is 5.2% (minimum is 0.9%)

Characterized entropy is valid for at least 30 days
Accessing a DRAM Cell

- wordline
- capacitor
- access transistor
- enable
- Sense Amp
- bitline

[Seshadri+ MICRO’17]
Accessing a DRAM Cell

1. Enable wordline
2. Connects cell to bitline
3. Cell loses charge to bitline
4. Deviation in bitline voltage
5. Enable sense amp
6. Cell charge restored

Seshadri+ MICRO’17