QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

Ataberk Olgun§†, Minesh Patel§, A. Giray Yaşlıkońçu§, Haocong Luo§, Jeremie S. Kim§, F. Nisa Bostancı§†, Nandita Vijaykumar§○, Oğuz Ergin†, Onur Mutlu§

§ETH Zürich, †TOBB University of Economics and Technology, ○University of Toronto

ISCA 2021

Maria Makeenkova
Seminar in Computer Architecture
02/06/22
Executive Summary

• Motivation: True random numbers are used across a wide range of workloads
Executive Summary

• Motivation: True random numbers are used across a wide range of workloads

• Problem:
  • High throughput TRNGs use specialized hardware
  • Not all computing systems have designated TRNG hardware
  • Limited ability to run TRN-needing applications
Executive Summary

• **Motivation:** True random numbers are used across a wide range of workloads

• **Problem:**
  - High throughput TRNGs use specialized hardware
  - Not all computing systems have designated TRNG hardware
  - Limited ability to run TRN-needing applications

• **Goal:** high-throughput and low-latency TRNG in commodity DRAM chips
Executive Summary

• Motivation: True random numbers are used across a wide range of workloads

• Problem:
  • High throughput TRNGs use specialized hardware
  • Not all computing systems have designated TRNG hardware
  • Limited ability to run TRN-needing applications

• Goal: high-throughput and low-latency TRNG in commodity DRAM chips

• Key Idea: Use Quadruple Activation to generate metastability in DRAM Sense Amplifiers
Presentation Overview

- Random Number Generation
- Challenges and Solution
- Background
- QUAC-TRNG
- Experimental Results and Conclusion
- Paper Analysis:
  - Strengths
  - Weaknesses
- Audience Questions and Discussion
Use Cases for Random Numbers

- **Cryptography** (e.g. signature generation)
- **Scientific Simulations**
- **Machine Learning** (e.g. Randomized Training)
Random Number Generators

**Random Number Generator (RNG):** device or program that produces random numbers
Random Number Generators

• Pseudo-Random Number Generator (PRNG)

• True Random Number Generator (TRNG)
Random Number Generators

- Pseudo-Random Number Generator (PRNG)
  - Arithmetic transformation on seed

- True Random Number Generator (TRNG)
  - Sample random physical processes
Random Number Generators

• Pseudo-Random Number Generator (PRNG)
  • Arithmetic transformation on seed

• True Random Number Generator (TRNG)
  • Sample random physical processes
Random Number Generators

• Pseudo-Random Number Generator (PRNG)
  • Arithmetic transformation on seed

seed → state → 01001101

If seed is compromised the RN-sequence can be regenerated.

• True Random Number Generator (TRNG)
  • Sample random physical processes

Physical process → sample → 01001101
Random Number Generators

• Pseudo-Random Number Generator (PRNG)
  • Arithmetic transformation on seed

![Diagram of seed, state, and 01001101]

If seed is compromised the RN-sequence can be regenerated.

• True Random Number Generator (TRNG)
  • Sample random physical processes

![Diagram of physical process, sample, and 01001101]

Output cannot be regenerated by observing physical process.
Post Processing

Randomness Source → Bias → Sampler → 111100111

[Image: Ataberk Olgun ISCA 21-Talk]
Post Processing

- Remove bias
- Improve TRN quality
Cryptographic Hash Functions
Cryptographic Hash Functions

• Scramble and randomize input
Von Neuman Corrector (VNC)
Von Neuman Corrector (VNC)

- Split all bits into groups of 2
Von Neuman Corrector (VNC)

- Split all bits into groups of 2
  - Remove group if same value
  - Replace with "1" if the bits are "01"
  - Replace with "0" otherwise
Von Neuman Corrector (VNC)

- Split all bits into groups of 2
  - Remove group if same value
  - Replace with "1" if the bits are "01"
  - Replace with "0" otherwise

- E.g. "0010" becomes "0" after VNC
Presentation Overview

• Random Number Generation
• Challenges and Solution
• Background
• QUAC-TRNG
• Experimental Results and Conclusion
• Paper Analysis:
  • Strengths
  • Weaknesses
• Audience Questions and Discussion
Challenges

• High throughput TRNGs use specialized hardware
• Not all computing systems have designated TRNG hardware
• Limited ability to run TRN-needing applications
Challenges

• High throughput TRNGs use **specialized hardware**
• Not all computing systems have designated TRNG hardware
• Limited ability to run TRN-needing applications

**Goal:** TRNG that uses commodity DRAM devices to generate random numbers with high throughput and low latency.
DRAM-based TRNGS
DRAM-based TRNGS

- DRAM is used in most computing systems
DRAM-based TRNGS

• DRAM is used in most computing systems
• Low hardware cost to implement
DRAM-based TRNGS

- DRAM is used in most computing systems
- **Low hardware cost** to implement
- **In-memory generation**: less data movement
  - Good for PIM workloads
  - Avoids communication with designated TRNG hardware
DRAM-based TRNGs

- DRAM is used in most computing systems
- **Low hardware cost** to implement
- **In-memory generation**: less data movement
  - Good for PIM workloads
  - Avoids communication with designated TRNG hardware
- **High throughput**: more applications can use TRNs
Presentation Overview

• Random Number Generation
• Challenges and Solution
• Background
• QUAC-TRNG
• Experimental Results and Conclusion
• Paper Analysis:
  • Strengths
  • Weaknesses
• Audience Questions and Discussion
DRAM Organization

[Diagram of DRAM organization showing the memory controller, DDR interface, and various DRAM components including chips, banks, and subarrays.]

[Diagram of DRAM subarray showing wordline drivers, DRAM MAT, and sense amplifiers.]
Accessing a DRAM Cell

Ataberk Olgun ISCA 21-Talk, 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

Sense Amp

-deviation in bitline voltage

[ataberk olgun isca 21-talk, seshadri+ micro’17]
DRAM Operation

DRAM Command Sequence

[Ataberk Olgun ISCA 21-Talk, Kim+ HPCA’19]
DRAM Timing Parameters

• DRAM controller must obey timing parameters when scheduling commands
DRAM Timing Parameters

• DRAM controller must obey timing parameters when scheduling commands

• **ACT** and **PRE** commands interleaved by $t_{\text{RAS}}$
  • Allow cells to fully restore charge
DRAM Timing Parameters

• DRAM controller must obey timing parameters when scheduling commands
  • ACT and PRE commands interleaved by $t_{RAS}$
    • Allow cells to fully restore charge
  • PRE and ACT interleaved by $t_{RP}$
    • Settle the bitline voltage, disable activated wordline
Presentation Overview

• Random Number Generation
• Challenges and Solution
• Background
• QUAC-TRNG
• Experimental Results and Conclusion
• Paper Analysis:
  • Strengths
  • Weaknesses
• Audience Questions and Discussion
Quadruple Activation

- We can induce entropy in DRAM by violating the timing parameters for the following command sequence:
Quadruple Activation

• We can induce entropy in DRAM by **violating the timing parameters** for the following command sequence:

  ![Command Sequence Diagram]

  - Activates 4 consecutive rows in succession
Quadruple Activation

• We can induce entropy in DRAM by violating the timing parameters for the following command sequence:

• Activates 4 consecutive rows in succession
• Works in commodity DRAM chips by SK Hynix
Hierarchical DRAM Organization

• Hierarchical wordlines
  2 step DRAM row access
Hierarchical DRAM Organization

- Hierarchical wordlines
  - 2 step DRAM row access
  - 1. Select and activate master wordline (MWL)
Hierarchical DRAM Organization

- **Hierarchical wordlines**
  2 step DRAM row access:
  1. Select and activate master wordline (MWL)
  2. Drive local wordlines with control signals to activate DRAM cells
Hypothetical Row Decoder

• Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands
Hypothetical Row Decoder

• Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands
Hypothetical Row Decoder

• Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands

• ACT(R0)
Hypothetical Row Decoder

- Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands

- **ACT(R0)-PRE**
Hypothetical Row Decoder

• Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands

• ACT(R0)-PRE-ACT(R3)
Generating Random Values via QUAC

- Voltage Difference
  - $V_{TH}$
  - $V_{DD}/2$
  - $-V_{TH}$
  - $-V_{DD}$

- Ready to Sense Voltage Level

- Logic-1

- Sense Amplifier
  - Enable

- $V_{DD}/2$

[Ataberk Olgun ISCA 21-Talk]
Generating Random Values via QUAC

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

R3

R2

R1

R0

\[ V_{DD}/2 \]

Sense Amplifier

Enable

\[ V_{dd} \]

Voltage Difference

\[ V_{TH} \]

0

\[ -V_{TH} \]

\[ -V_{dd} \]

ACT R0

PRE

ACT R3

Time

[Ataberk Olgun ISCA 21-Talk]
Generating Random Values via QUAC

Voltage Difference: $V_{DD}/2 \pm \varepsilon$

Enable Sense Amplifiers

Random perturbation

Enable

[Ataberk Olgun ISCA 21-Talk]
Mechanism
Mechanism

① Segment Initialization

DRAM Segment

Row 0
Row 1
Row 2
Row 3

DRAM Segment

All ‘0’ Row
All ‘1’ Row

Sense Amps
Mechanism
Mechanism

1. Segment Initialization
   - DRAM Segment
   - Row 0
   - Row 1
   - Row 2
   - Row 3

2. QUAC
   - DRAM Segment
   - Row 0
   - Row 1
   - Row 2
   - Row 3

3. Read Random Data
   - Sense Amps
   - 256-Bit Entropy Blocks

4. Post Processing
   - Memory Controller
   - SHA-256
   - 256-bit Random Number
Presentation Overview

• Random Number Generation
• Challenges and Solution
• Background
• QUAC-TRNG
• Experimental Results and Conclusion
• Paper Analysis:
  • Strengths
  • Weaknesses
• Audience Questions and Discussion
QUAC-TRNG Testing Setup

- 136 DRAM chips from 17 off-the-shelf DDR4 modules by SK Hynix
- Modified SoftMC
- DDR4 commands sent to FPGA board
- Control DRAM temperatures (50°C)
Shannon Entropy

Measure Randomness in Bitstream
Shannon Entropy

Measure Randomness in Bitstream

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

Probabilities: proportion of logic-1 and logic-0 values in bitstream
Shannon Entropy

Measure Randomness in Bitstream

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

**Probabilities:** proportion of logic-1 and logic-0 values in bitstream

\[ \text{SE}(111\ldots11) = 0 \]
\[ 0 < \text{SE}(1001\ldots010) < 1 \]
Shannon Entropy

Measure Randomness in Bitstream

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

**Probabilities:** proportion of *logic-1* and *logic-0* values in bitstream

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

- Perform QUAC 1000 times
- Measure entropy for each SAs 1000-bit bitstream
Data Pattern Dependence
Data Pattern Dependence

- 50 °C, 8K DRAM segments, **16 data patterns**, across 17 DRAM modules
Data Pattern Dependence

- 50 °C, 8K DRAM segments, **16 data patterns**, across 17 DRAM modules

**Data pattern: 1111 (four ones)**

<table>
<thead>
<tr>
<th>R0</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Data Pattern Dependence

- 50 °C, 8K DRAM segments, 16 data patterns, across 17 DRAM modules
Data Pattern Dependence

• **Cache block entropy (CBE):** sum of entropy of all bitlines in that cache block (max: 512)
Data Pattern Dependence

- **Cache block entropy (CBE):** sum of entropy of all bitlines in that cache block (max: 512)

- **Average CBE:** average across all cache blocks in a module
- **Maximum CBE:** greatest of all CBES in a module
Data Pattern Dependence
Data Pattern Dependence

- Average entropy varies with data pattern
Data Pattern Dependence

• Average entropy varies with data pattern
• More randomness when \( R_0 \) initialized to inverted value of other 3 (more time to share charge)
Spatial Distribution of Entropy

Distribution of Entropy based on physical location of the segment on the DRAM chip
Spatial Distribution of Entropy

Distribution of Entropy based on physical location of the segment on the DRAM chip

**Segment entropy**: sum of all bitline entropies **in DRAM segment** (max: 64K)

Data pattern: "0111"
Spatial Distribution of Entropy
Spatial Distribution of Entropy

- Wave like pattern: Systematic process variations
Spatial Distribution of Entropy

- Wave like pattern: Systematic process variations
- Post manufacturing repair (rows being remapped)
Spatial Distribution of Entropy

- Wave like pattern: Systematic process variations
- Post manufacturing repair (rows being remapped)
- Segments distance from SAs
Spatial Distribution of Entropy

- Wave like pattern: Systematic process variations
- Post manufacturing repair (rows being remapped)
- Segments distance from SAs
- Significant increase towards the end and drop: differently sized subarrays at the end of bank?
Randomness Test
Randomness Test

1. Initialize highest entropy DRAM segments with data pattern "0111"
Randomness Test

1. Initialize **highest entropy** DRAM segments with data pattern "0111"
2. Perform QUAC
Randomness Test

1. Initialize **highest entropy** DRAM segments with data pattern "0111"
2. Perform QUAC
3. Read out the values generated in the SAs, split into 256-bit blocks
Randomness Test

1. Initialize **highest entropy** DRAM segments with data pattern "0111"
2. Perform QUAC
3. Read out the values generated in the SAs, split into 256-bit blocks
4. Post process with **Von Neumann Corrector** and **SHA-256**
NIST Statistical Test Suite

• Measure quality of TRNG
NIST Statistical Test Suite

- Measure quality of TRNG
- Runs multiple tests, evaluates statistical properties to find patterns
NIST Statistical Test Suite

- Measure **quality of TRNG**
- Runs multiple tests, evaluates statistical properties to find patterns
- Inputs can either **pass** or **fail** test
# NIST STS Results

Average p-value for each test

<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 STS Results

Desired: \textbf{p-value > 0.001}
- Both bitstreams \textbf{pass} all tests
- QUAC-TRNG outputs \textbf{high quality} random bitstreams

<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
QUAC-TRNG Throughput
QUAC-TRNG Throughput

Throughput of

\[ \frac{256 \cdot \text{SIB}}{L \cdot 10^{-9}} \] bits per second
QUAC-TRNG Throughput

Throughput of

\( \frac{256 \cdot \text{SIB}}{L \cdot 10^{-9}} \) bits per second

- SIB: SHA Input Blocks, number of input blocks with 256 bits of entropy in highest entropy DRAM segment
- L: Latency of QUAC operation [ns]
Results

Throughput per DRAM channel
Average, max and min measured across all DRAM modules

BGP: 4 banks from different groups to overlap DRAM command latencies

RC: initialize DRAM segment using in-DRAM copy
Results

Throughput per DRAM channel
Average, max and min measured across all DRAM modules

BGP: 4 banks from different groups to overlap DRAM command latencies

RC: initialize DRAM segment using in-DRAM copy
System Performance
System Performance

• SPEC2006 benchmark suite
System Performance

• SPEC2006 benchmark suite
• Simulate 3.2 GHz core, 4 DRAM channels
System Performance

- SPEC2006 benchmark suite
- Simulate 3.2 GHz core, 4 DRAM channels
- Calculate when channel idle
  - Issue QUAC-TRNG commands in idle periods
Results
Results

• 10.2 Gb/s average throughput
Results

- 10.2 Gb/s average throughput
- 3.22 Gb/s min, 14.3 Gb/s max
Results

• **10.2 Gb/s average throughput**
• **3.22 Gb/s min, 14.3 Gb/s max**
• **74.13% of imperical average** (3.44•4=13.76 Gb/s for 4 channels)
Comparison With Prior Work

• Throughput and latency on 4 DRAM channels

<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>
Comparison With Prior Work

- Throughput and latency on 4 DRAM channels

<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>

High latency can be alleviated with random number buffer
System Integration I

- SHA-256 can be implemented in hardware at low area and latency cost
  - Suitable for implementation in memory controller
  - 0.001mm² area
System Integration II

QUAC-TRNG:
• Memory Overhead: 192 KB reserved (0.002% of 8 GB DDR4)
System Integration II

QUAC-TRNG:
• Memory Overhead: 192 KB reserved (0.002% of 8 GB DDR4)
• Area Overhead: 0.0003mm²
System Integration II

QUAC-TRNG:
• Memory Overhead: 192 KB reserved (0.002% of 8 GB DDR4)
• Area Overhead: 0.0003mm²

• Total (including SHA-256): 0.0014mm² (0.04% of chip area)
Executive Summary

• Motivation: True random numbers are used across a wide range of workloads

• Problem:
  • High throughput TRNGs use specialized hardware
  • Not all computing systems have designated TRNG hardware
  • Limited ability to run TRN-needing applications

• Goal: high-throughput and low-latency TRNG in commodity DRAM chips

• Key Idea: Use Quadruple Activation to generate metastability in DRAM Sense Amplifiers
Presentation Overview

- Random Number Generation
- Challenges and Solution
- Background
- QUAC-TRNG
- Experimental Results and Conclusion
- Paper Analysis:
  - Strengths
  - Weaknesses
- Audience Questions and Discussion
Strengths of the Paper
Strengths of the Paper

• Explores a new and interesting idea which provides significant improvement
Strengths of the Paper

• Explores a new and interesting idea which provides significant improvement

• Many aspects of implementation are analyzed
Strengths of the Paper

• Explores a new and interesting idea which provides significant improvement
• Many aspects of implementation are analyzed
• Provides a lot of background
Strengths of the Paper

• Explores a new and interesting idea which provides significant improvement
• Many aspects of implementation are analyzed
• Provides a lot of background
• The concept is thoroughly researched and the results are presented comprehensively
Strengths of the Paper

• Explores a new and interesting idea which provides significant improvement
• Many aspects of implementation are analyzed
• Provides a lot of background
• The concept is thoroughly researched and the results are presented comprehensively
• PIM solution
Weaknesses of the Paper
Weaknesses of the Paper

• Not explored how QUAC-TRNG would interact with other processes on the system
Weaknesses of the Paper

• Not explored how QUAC-TRNG would interact with other processes on the system
• Only capable of using DRAM by one manufacturer to generate random numbers
Weaknesses of the Paper

• Not explored how QUAC-TRNG would interact with other processes on the system
• Only capable of using DRAM by one manufacturer to generate random numbers
• Repetitive writing style
DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators

F. Nisa Bostancı\textsuperscript{†\$} Ataberk Olgun\textsuperscript{†\$} Lois Orosa\textsuperscript{§} A. Giray Yağlıkçı\textsuperscript{§}
Jeremie S. Kim\textsuperscript{§} Hasan Hassan\textsuperscript{§} Oğuz Ergin\textsuperscript{†} Onur Mutlu\textsuperscript{§}

\textsuperscript{†}TOBB University of Economics and Technology \hspace{1cm} \textsuperscript{§}ETH Zürich
DR-STRaNGe

- Random number buffering mechanism
  - Hide high latency
DR-STRaNGe

- Random number buffering mechanism
  - Hide high latency
- Predict and use idle DRAM cycles to generate RN
  - Less interference in system
DR-STRaNGe

• Random number buffering mechanism
  • Hide high latency

• Predict and use idle DRAM cycles to generate RN
  • Less interference in system

• RNG-aware scheduler
  • Reduces interference
  • Separate RNG and non-RNG request queues
  • Schedule based on priority levels of running processes
DR-STRaNGe

- Random number buffering mechanism
  - Hide high latency
- Predict and use idle DRAM cycles to generate RN
  - Less interference in system
- RNG-aware scheduler
  - Reduces interference
  - Separate RNG and non-RNG request queues
  - Schedule based on priority levels of running processes
- Application Interface
Presentation Overview

• Random Number Generation
• Challenges and Solution
• Background
• QUAC-TRNG
• Experimental Results and Conclusion
• Paper Analysis:
  • Strengths
  • Weaknesses
• Audience Questions and Discussion
Discussion I

• Are there potential unseen dangers in violating DRAM timing parameters?
  • Induce Rowhammer?
  • Reduce quality of SAs? Accelerating aging?
  • Affect data stored nearby?
  • How can we potentially avoid those issues?
Discussion II

• Currently there are not many specific workloads that would require such a high TRN throughput.
  • Future oriented thinking?
  • Can this development lead to TRN intense workloads becoming more common?
  • Can this lead to increased security in commodity devices?
Discussion III

• Where do you most see QUAC-TRNG implemented?
  • What kinds of workloads?
  • What kinds of computing systems?
A Thank You to my Mentors

Ataberk Olgun, Hasan Hassan, Konstantinos Kanellopoulos
Backup Slides

SHA-256, More on DR-STRaNGE
Shortcomings of Past Works
Shortcomings of Past Works

• **High latencies** because they rely on **fundamentally slow processes**
  • e.g. DRAM retention values or startup values
Shortcomings of Past Works

- **High latencies** because they rely on fundamentally slow processes
  - e.g. DRAM retention values or startup values

- **Low throughput**
  - Use only small portions of selected DRAM
  - Or fail to induce metastability in all sense amplifiers (SAs)
DRAM Chip Requirements

• Row addresses only differ in their two LSBs
  • e.g. rows 111, 110, 101, 100
  • But not 110, 101, 100, 001

• The address of the two ACT commands must have LSBs inverted
  • e.g. rows 111, 110, 101, 100
  • Or rows 111, 110, 101, 100
  • The order of the activate commands does not matter
Temperature Dependence

• Test bitline entropies at 50°C, 65°C and 85°C
• On real DRAM chips from 5 modules, with "0111" data pattern

Conclusion: Implementation needs to account for changes in temperature.
Maintaining Entropy with varying Temperature

- Goal: SHA-256 input blocks always have 256 bits of entropy despite different temperatures

- Memory controller stores list of column addresses for temperature ranges
- List initialized during one-time characterization step
- Depending on temperature QUAC-TRNG gets optimal address from list
Spatial Distribution of Entropy

- **Cache block entropy**: sum of entropy of all bitlines in that cache block
Spatial Distribution of Entropy

• **Cache block entropy**: sum of entropy of all bitlines in that cache block
• Cache block in *highest entropy segment* (in each module)
• Data pattern: "0111"
Spatial Distribution of Entropy
Spatial Distribution of Entropy

- **Peaks** around the **middle**
Spatial Distribution of Entropy

- **Peaks** around the **middle**
- **Drops** towards the **end**
  - Higher numbered cache blocks are less random
Spatial Distribution of Entropy

- **Peaks** around the middle
- **Drops** towards the end
  - Higher numbered cache blocks are less random
- Non uniform distribution
Spatial Distribution of Entropy

- **Peaks** around the **middle**
- **Drops** towards the **end**
  - Higher numbered cache blocks are less random
- **Non uniform distribution**
  - Systematic variation in manufacturing
Optimizing Latency & Throughput

• Latency is dominated by initialization of 4 DRAM rows

• Use in-DRAM copy operations to initialize segments at row granularity (Row-Clone based)

• Concurrently execute QUAC across multiple DRAM banks (bank-level parallelism)
System Integration

• Memory Overhead:
  • Simultaneously use 8 DRAM rows
  • 4 segments
  • Across 4 banks in different bank groups
  • Total: 192 KB reserved (0.002% of 8 BG DDR4)
DR-STRaNGe

End-to-End System Design for DRAM-based TRNGs
DR-STRaNGe: 3 Key Challenges

1. Can cause significant slowdown of running applications
2. Doesn't differentiate between RNG and non-RNG memory requests
   - Overhead from modifying timing parameters
   - Unfair scheduling
3. High latency
DR-STRaNGe: Solutions

- Random number buffering mechanism
  - Hide high latency
- Predict and use idle DRAM cycles to generate RN
  - Less interference in system
- RNG-aware scheduler
  - Reduces interference
  - Separate RNG and non-RNG request queues
  - Schedule based on priority levels of running processes
- Application Interface
DR-STRaNGe: Performance

- Improves performance for both RNG and non-RNG tasks
- Reduces execution time compared to RNG-oblivious system:
  - Dual core:
    - By 17.9% for non-RNG
    - By 25.1% for RNG
  - Multi core (average over 4-, 8-, 16-core workloads):
    - 7.6% for non-RNG
    - 17.8% for RNG
- Improves system fairness by 32.1%
- 16 entry random number buffer achieves an average serve rate of 0.55
- Idleness predictor: 19.3% and 23.9% improvement
DR-STRaNGe: Area and Energy Consumption

- Area Overhead: 0.0022mm²
  - 0.00048% of Intel Cascade Lake CPU Core (at 22nm process technology node)
- Reduced energy consumption and total memory cycles by 21% for RNG and 15.8% for non-RNG
Simple DRAM Idleness Predictor

- **Goal:** identify long idle periods in DRAM
- Uses last accessed memory address to predict period length
- Table stored for each channel:
  - 2-bit saturating counters
  - Register for *last accessed address value*
  - Counter for *idle period length* (initialized at 0)
Simple DRAM Idleness Predictor

- A channels predictor table is accessed when request queue is empty
- 2 kinds of idle periods:
  - Long: # of cycles $\geq$ \textit{Period Threshold}
  - Short: # of cycles $< \textit{Period Threshold}$
    - \textit{Period Threshold} empirically determined at 40 cycles
- Predictor table updated during idle periods
Reinforcement Learning Agent for DRAM Idleness Predictor

- Define DRAM idleness problem as a reinforcement learning (RL) problem
- State machine
- Performing action \( a \) at state \( s \) generates Q-value \( Q(s,a) \)
- 2 possible actions:
  - Initiate random number generation
  - Wait
- After action taken: update \( Q(s,a) \) determine reward \( r \)
  - Idle period length determines correctness of prediction
  - \( Q(s,a) = (1-a) Q(s,a) + a \cdot r \) \((a = 0.05; \text{learning rate})\)
RNG-Aware Memory Scheduler

• Goal: improve system fairness, don't stall any request for too long
• 2 modes for memory controller: RNG and non-RNG
• Separate queues for RNG and non-RNG memory requests
• Use OS provided priority levels for applications to prioritize one of the queues
• Schedule all the requests in a queue at a time
System Integration

• SHA-256 can be implemented in hardware at low area and latency cost
  • Suitable for implementation in memory controller
  • 65 clock cycle latency, 19.7 GB/s throughput, 0.001mm² area
SHA-256

- Secure Hashing Algorithm
- Input padded to 512 bits
- Divide input into 32-bit words: $M_0...M_n$
- Process the input for each $M_i$
- 8 buffered A, B, C, D, E, F, G, H of 32 bits each are used
  - Values are fixed at the beginning
SHA-256

Process each 16 words for 64 rounds

\[
\begin{align*}
\text{Ch}(E, F, G) &= (E \land F) \oplus (\neg E \land G) \\
\text{Ma}(A, B, C) &= (A \land B) \oplus (A \land C) \oplus (B \land C) \\
\Sigma_0(A) &= (A \gg 2) \oplus (A \gg 13) \oplus (A \gg 22) \\
\Sigma_1(E) &= (E \gg 6) \oplus (E \gg 11) \oplus (E \gg 25)
\end{align*}
\]

- \text{W}_t$: different depending on the repetition, for the first 16 rounds it's the input message
- \text{K}_t$: has a new fixed value for each round

# 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>76TT21NU8I8-4G</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>TLRD4G2G666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M7</td>
<td>TLRD4G2G666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M8</td>
<td>TLRD4G2G666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M9</td>
<td>TLRD4G2G666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M10</td>
<td>TLRD4G2G666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M11</td>
<td>TLRD4G2G666HC18F-SBK</td>
<td>H5AN4G8NMFR-VKC</td>
<td>2666</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>M12</td>
<td>TLRD4G2G666HC18F-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.
SHA-256

• In the end the initial value of A, B, C, D, E, F, G, H is added to the computed values
• Total 256 bits of output
NIST Statistical Test Suite

• Validate randomness
• Null hypothesis H0: input sequence is random
• Outputs \( p \)-value for all statistical test used
• H0 holds if \( p \)-value larger than level of significance \( \alpha \)
• Here: \( \alpha=0.001 \)
NIST STS Results

Desired: \textbf{p-value} > 0.001

- Both bitstreams pass all tests
- QUAC-TRNG outputs \textbf{high quality} random bitstreams

SHA-256:

- DRAM segment produces 1Mb sequences
- Test 1024 sequences per segment
- 99.28\% pass NIST STS (over acceptable rate of 98.84\%)

\textbf{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>

\textit{Average p-value for each test}

\textit{VNC: Von Neumann Corrector}