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

Ataberk Olgun[§†] Minesh Patel[§] A. Giray Yağlıkçı[§] 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

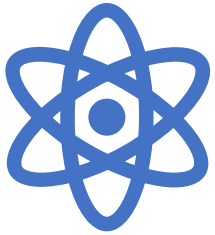
- Motivation: True random numbers are used across a wide range of workloads
- Problem:
  - High throughput TRNGs use specialized hardware
  - Not all computing systems have designated TRNG hardware
  - Limited ability to run TRN-needing applications
- Goal: high-throughput and low-latency TRNG in commodity DRAM chips
- Key Idea: Use Quadruple Activation to generate metastability in DRAM Sense Amplifiers

Presentation Overview

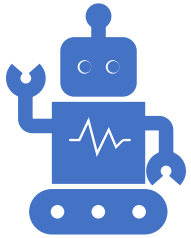
- Random Number Generation
- Challenges and Solution
- Background
- QUAC-TRNG
- Experimental Results and Conclusion
- Paper Analysis:
  - Strengths
  - Weaknesses
- Audience Questions and Discussion

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

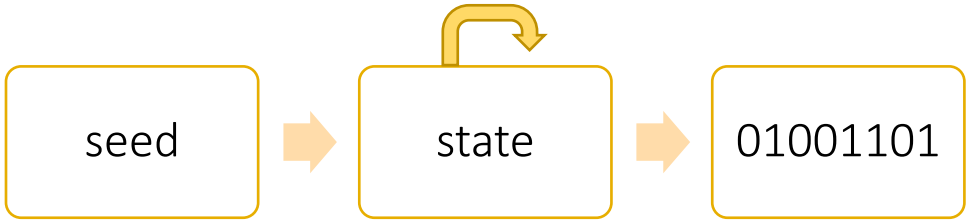
Random Number Generators

- Pseudo-Random Number Generator (PRNG)

- True Random Number Generator (TRNG)

Random Number Generators

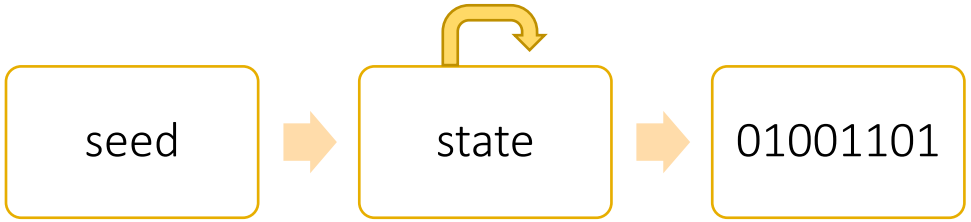
- Pseudo-Random Number Generator (PRNG)
  - Arithmetic transformation on seed



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

Random Number Generators

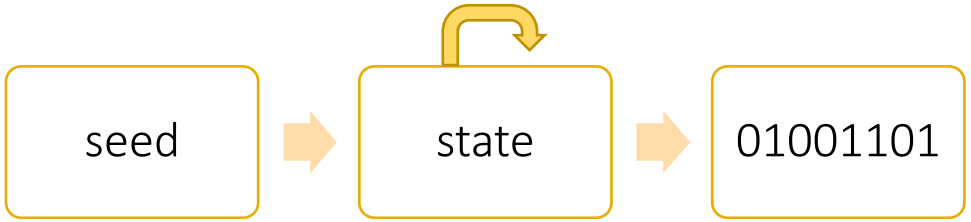
- Pseudo-Random Number Generator (PRNG)
  - Arithmetic transformation on seed



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

Random Number Generators

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

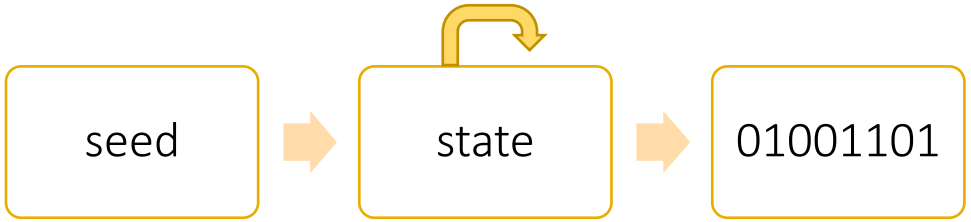


If seed is compromised the RN-sequence can be regenerated.

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

Random Number Generators

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

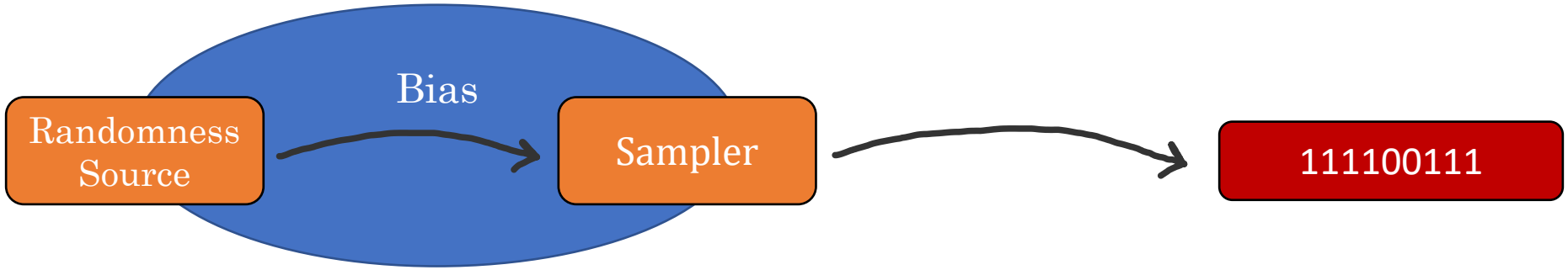


If seed is compromised the RN-sequence can be regenerated.

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

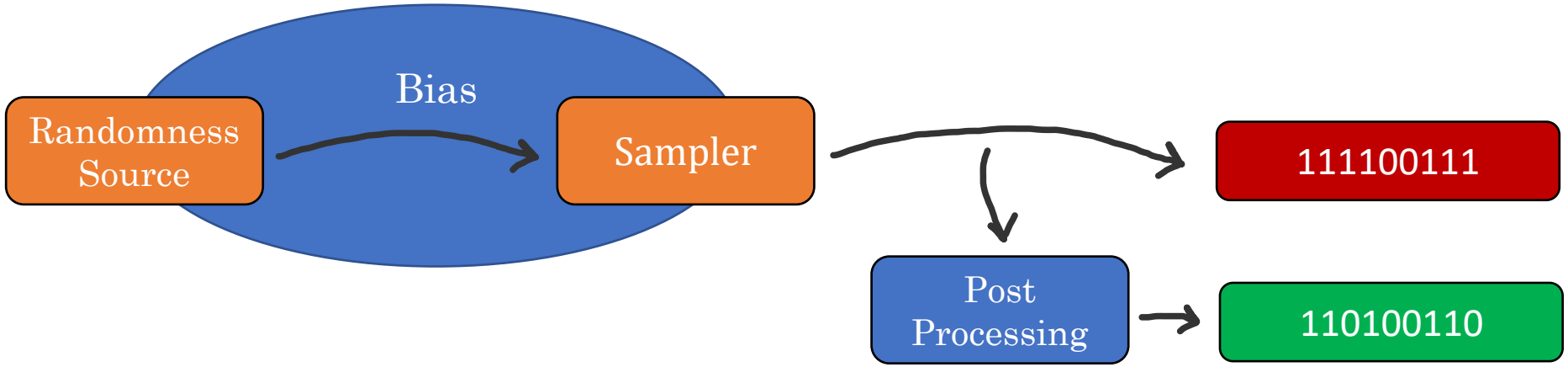
Output cannot be regenerated by observing physical process.

Post Processing



[Image: Ataberk Olgun ISCA 21-Talk]

Post Processing



Post Processing:

- Remove bias
- Improve TRN quality

[Image: Ataberk Olgun ISCA 21-Talk]

Cryptographic Hash Functions

Cryptographic Hash Functions

- Scramble and randomize input

[Image: Ataberk Olgun ISCA 21-Talk]

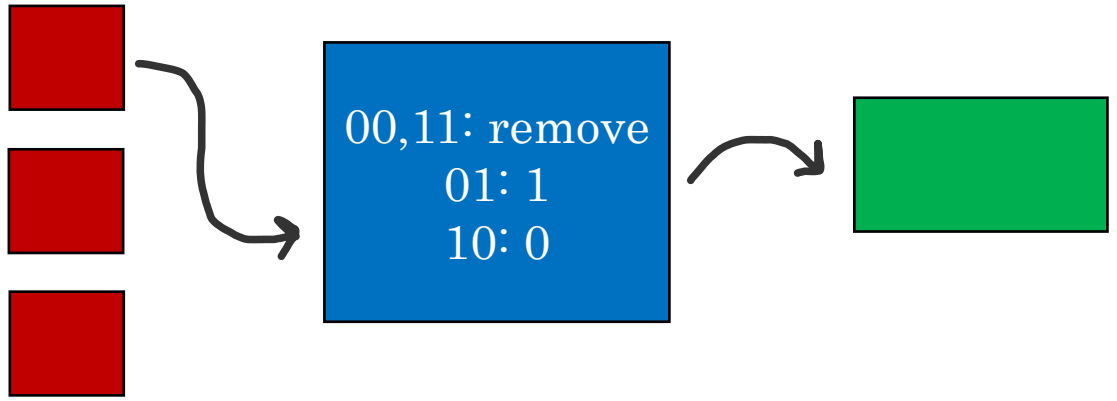
Von Neuman Corrector (VNC)

Von Neuman Corrector (VNC)

- Split all bits into groups of 2

Von Neuman Corrector (VNC)

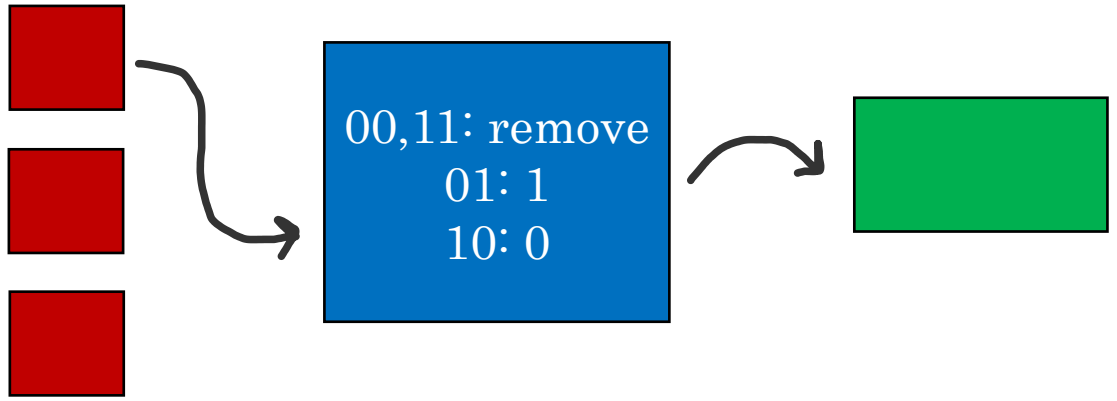
- Split all bits into groups of 2
  - Remove group if same value
  - Replace with "1" if the bits are "01"
  - Replace with "0" otherwise



[Image: Ataberk Olgun ISCA 21-Talk]

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

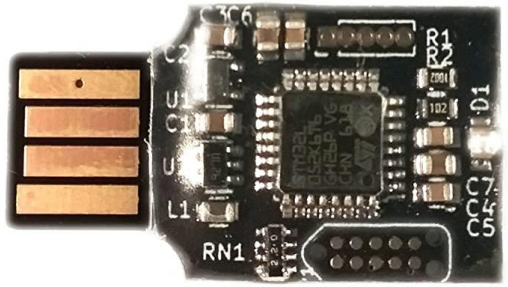


[Image: Ataberk Olgun ISCA 21-Talk]

Presentation Overview

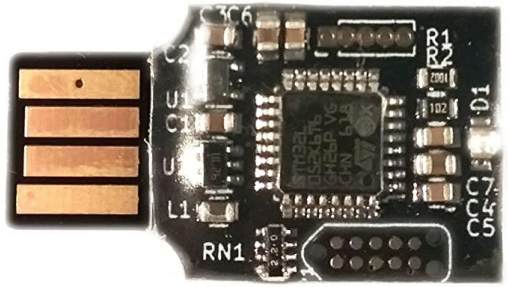
- Random Number Generation
- Challenges and Solution
- Background
- QUAC-TRNG
- Experimental Results and Conclusion
- Paper Analysis:
  - Strengths
  - Weaknesses
- Audience Questions and Discussion

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

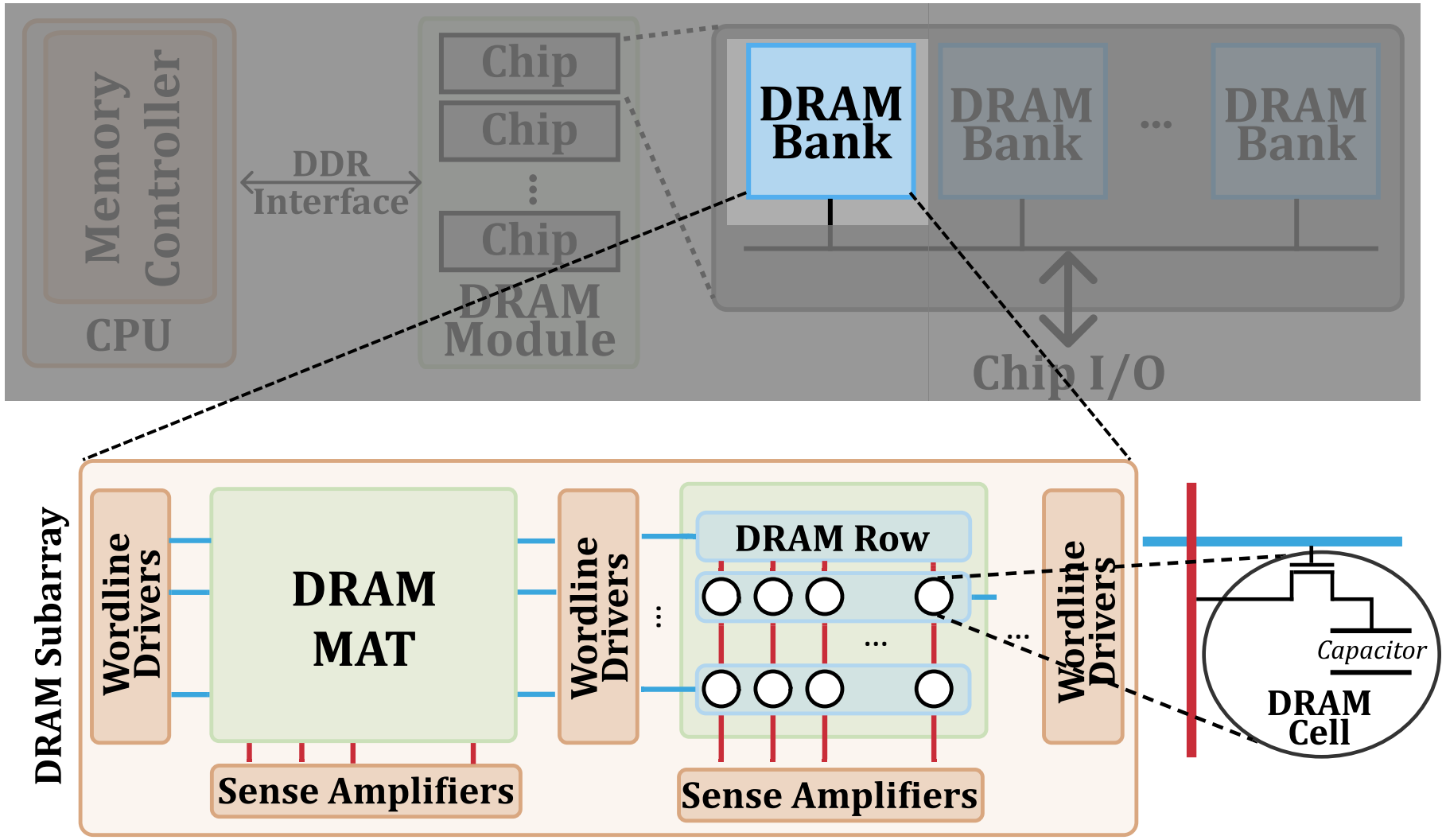
DRAM-based TRNGS

- 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

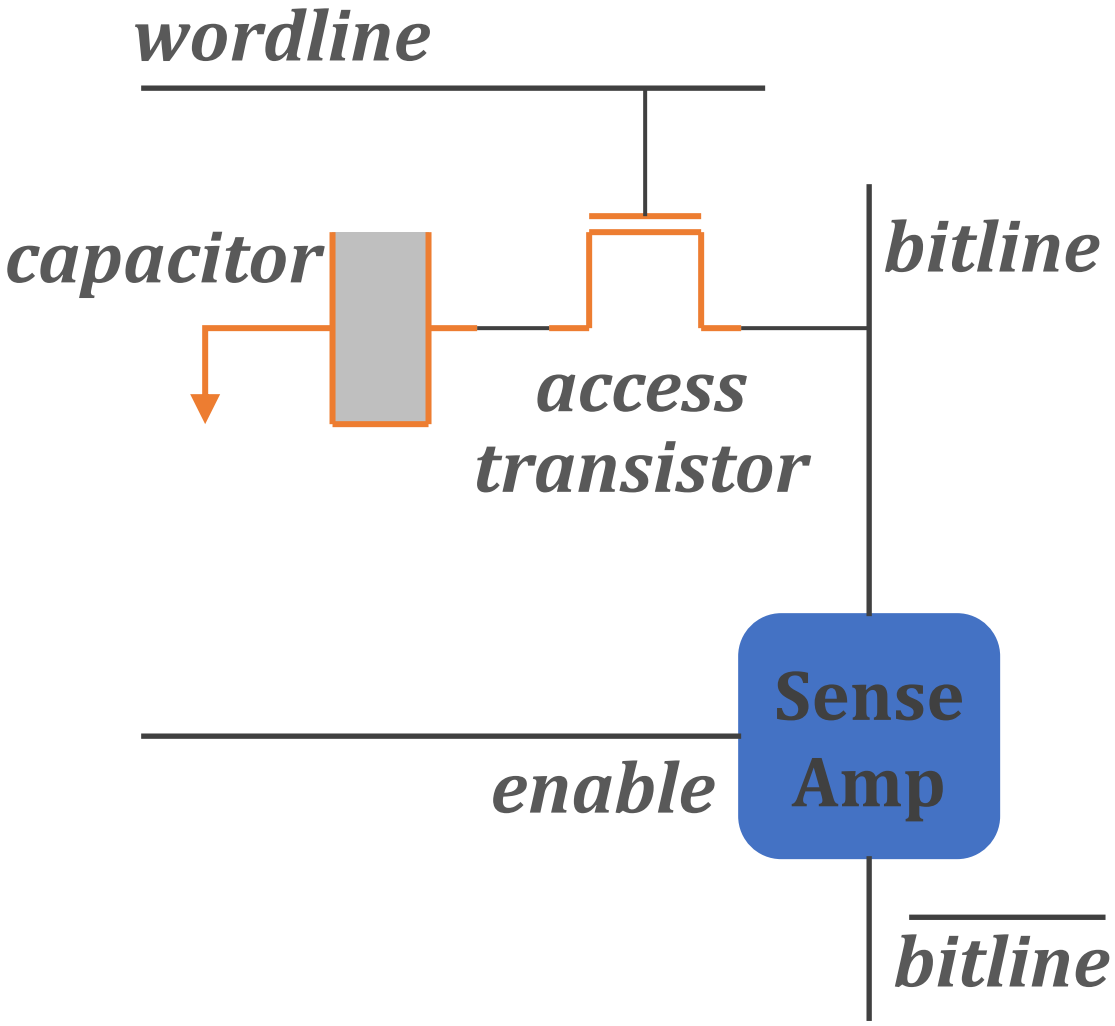
Presentation Overview

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

DRAM Organization

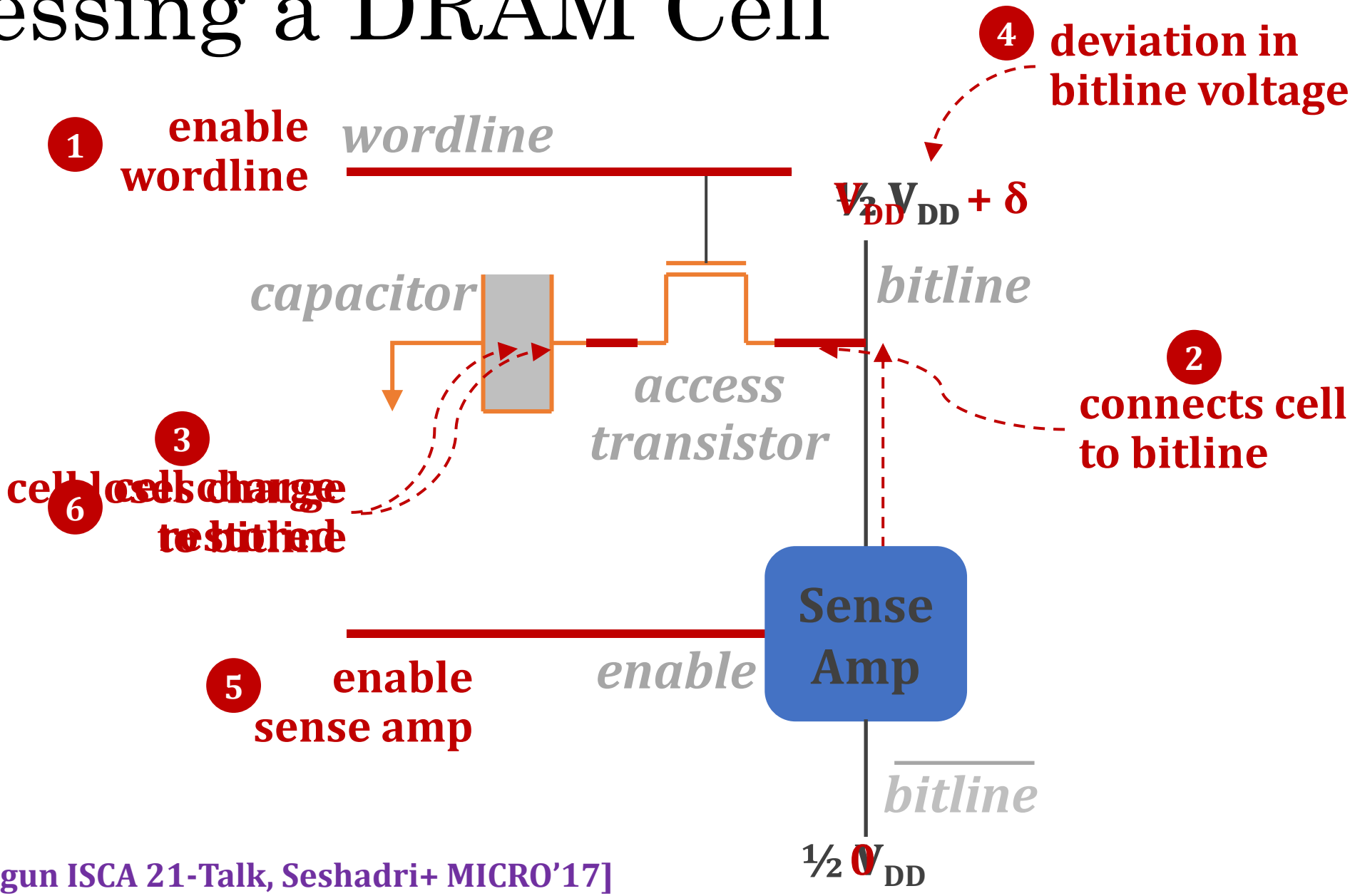


Accessing a DRAM Cell



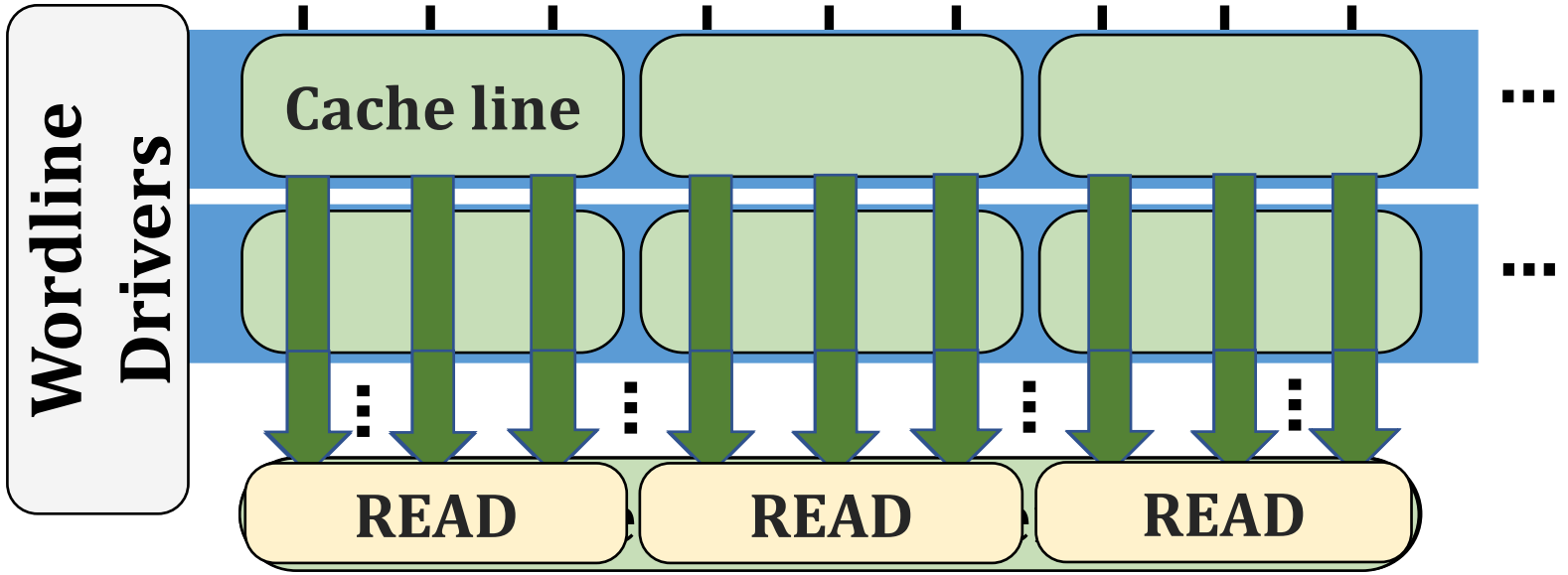
[Ataberk Olgun ISCA 21-Talk, Seshadri+ MICRO'17]

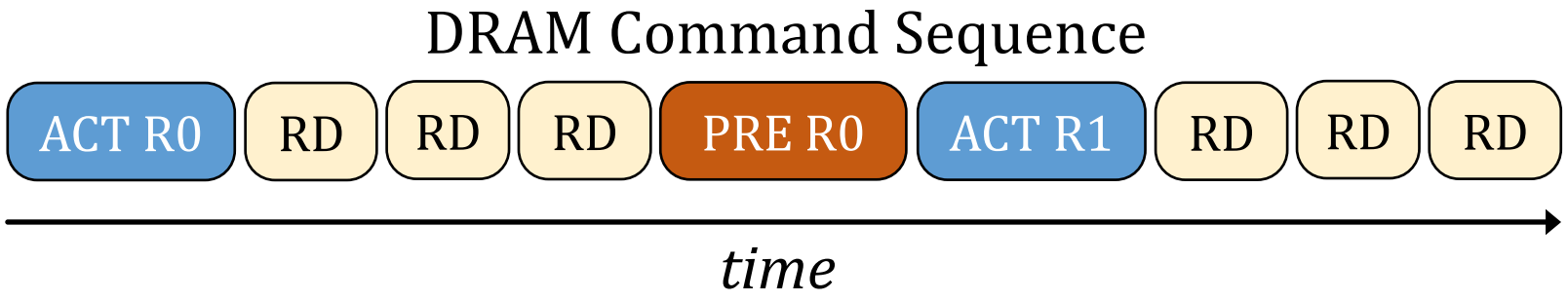
Accessing a DRAM Cell



[Ataberk Olgun ISCA 21-Talk, Seshadri+ MICRO'17]

DRAM Operation





[Ataberk Olgun ISCA 21-Talk, Kim+ HPCA'19]

DRAM Timing Parameters

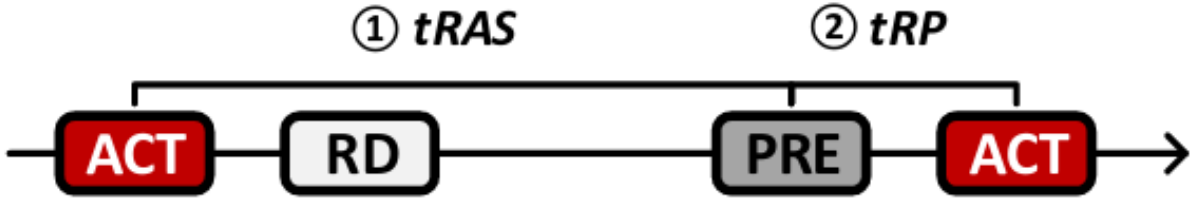
- DRAM controller must obey timing parameters when scheduling commands

DRAM Timing Parameters

- DRAM controller must obey timing parameters when scheduling commands
- **ACT and PRE** commands interleaved by t_{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 ***tRAS***
  - Allow cells to fully restore charge
- **PRE and ACT** interleaved by ***tRP***
  - 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:

- Activates 4 consecutive rows in succession

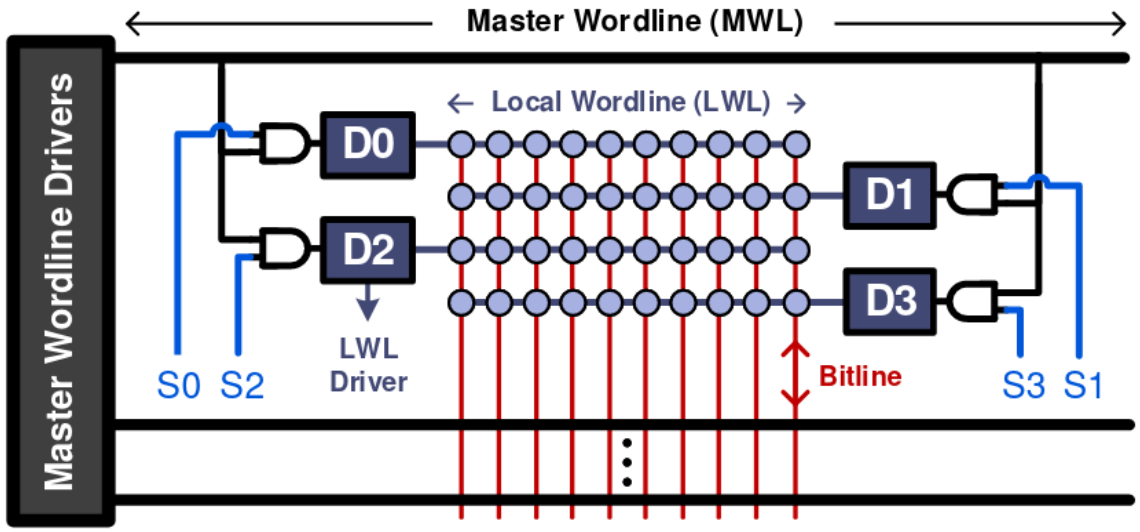
Quadruple Activation

- 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

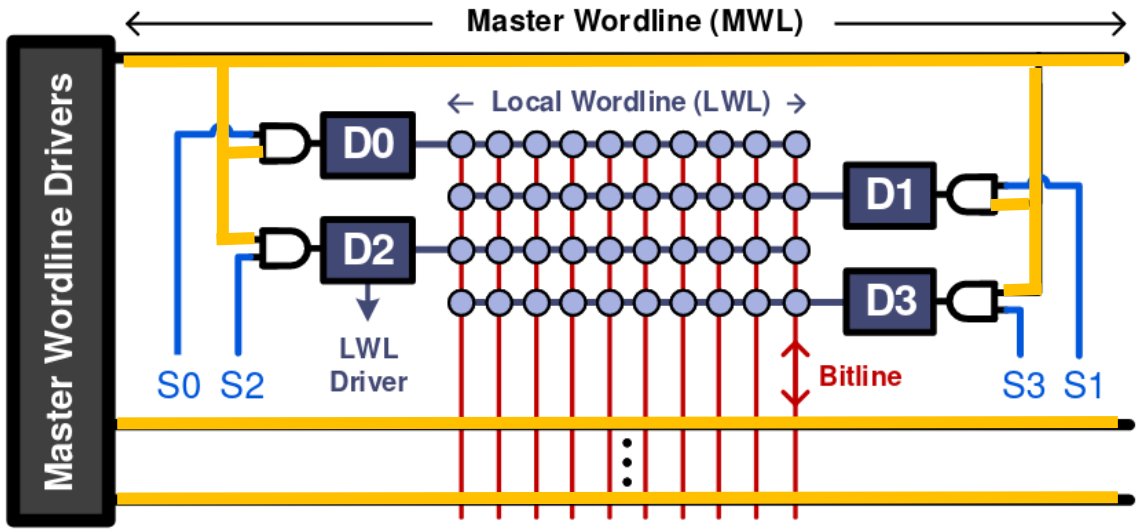
Hierarchical DRAM Organization

- **Hierarchical wordlines**
  2 step DRAM row access



Hierarchical DRAM Organization

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

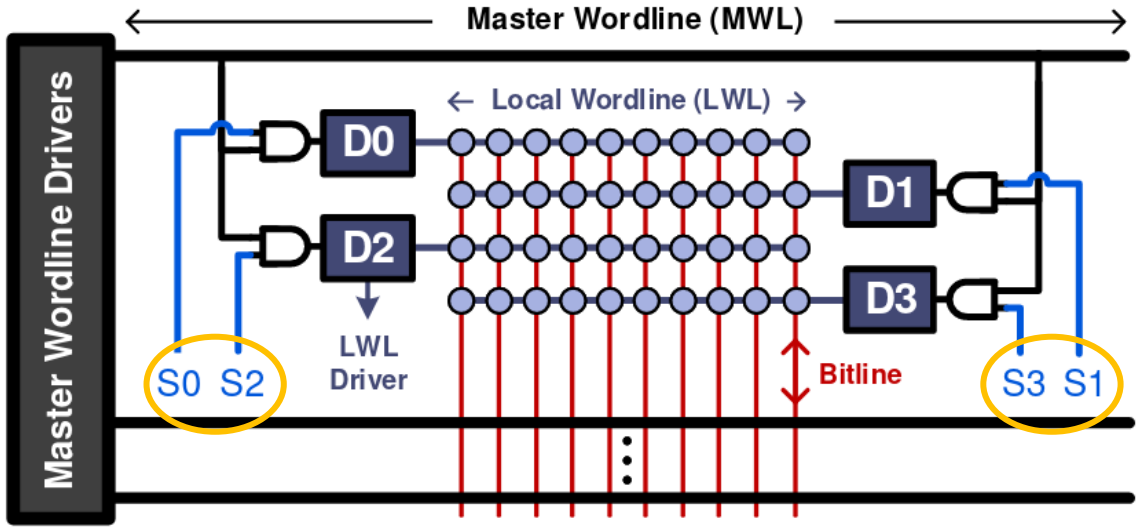


Hierarchical DRAM Organization

- **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

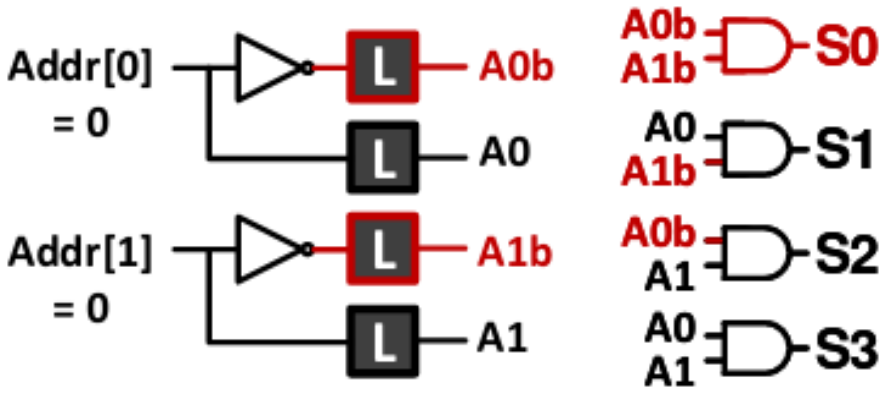


Hypothetical Row Decoder

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

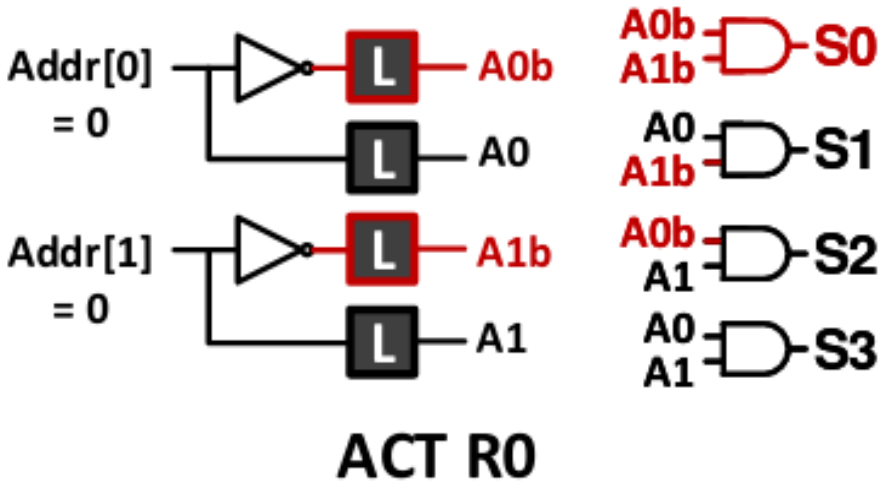
Hypothetical Row Decoder

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



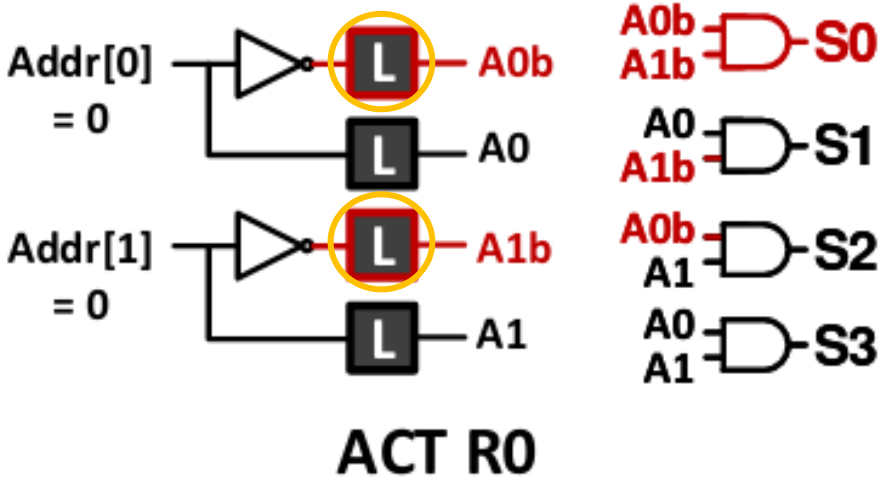
Hypothetical Row Decoder

- Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands
- ACT(R0)



Hypothetical Row Decoder

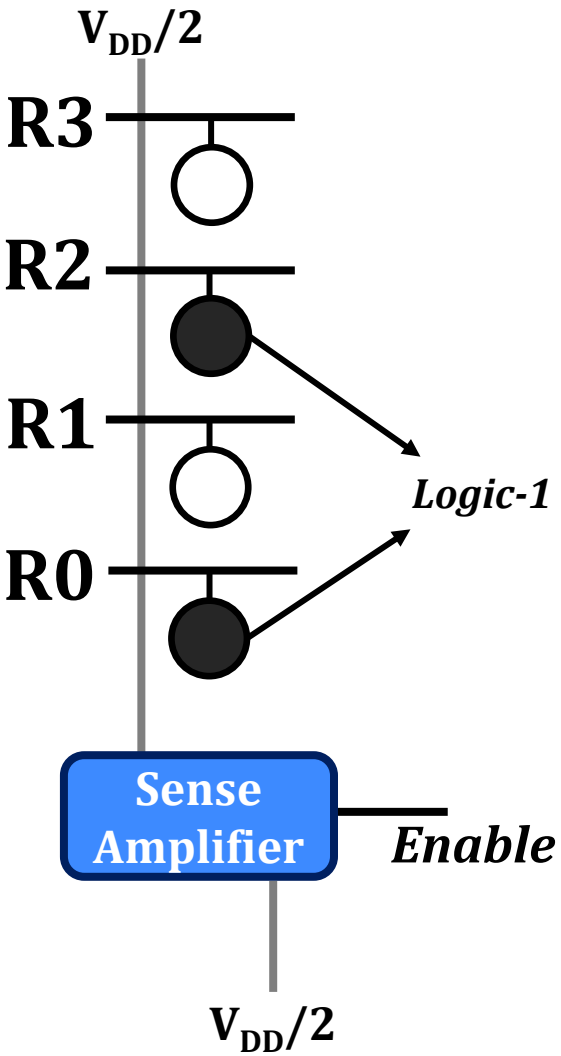
- Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands
- ACT(R0)-PRE

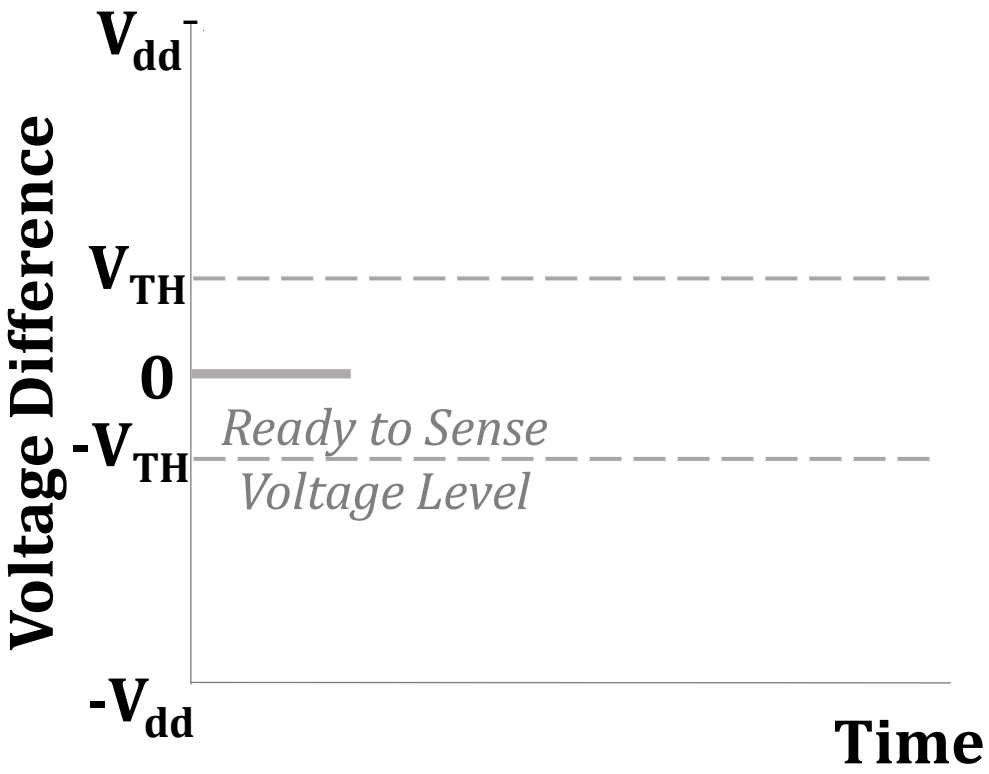


Hypothetical Row Decoder

- Goal: Simultaneously activates 4 rows when it receives a series of ACT-PRE-ACT commands
- ACT(R0)-PRE-ACT(R3)

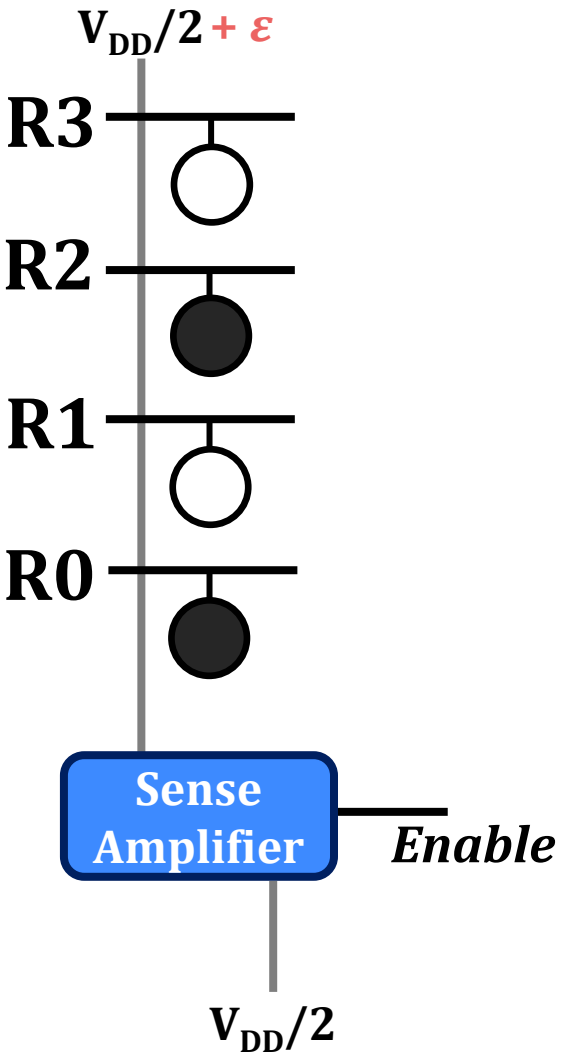
Generating Random Values via QUAC

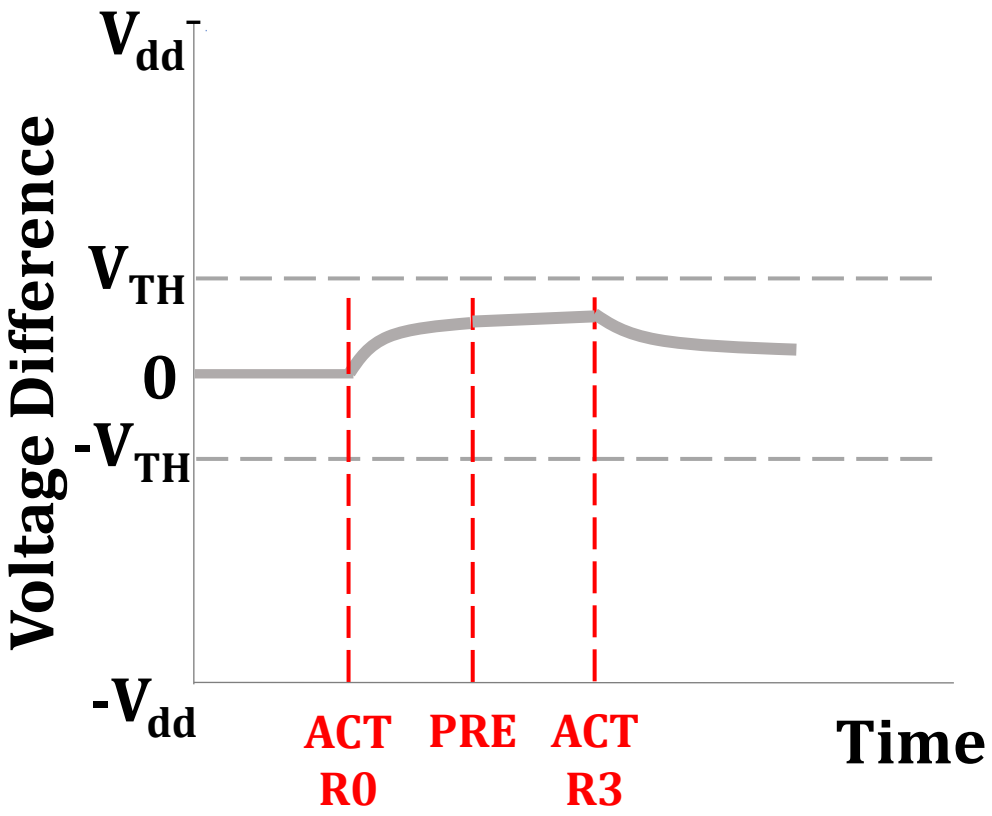




[Ataberk Olgun ISCA 21-Talk]

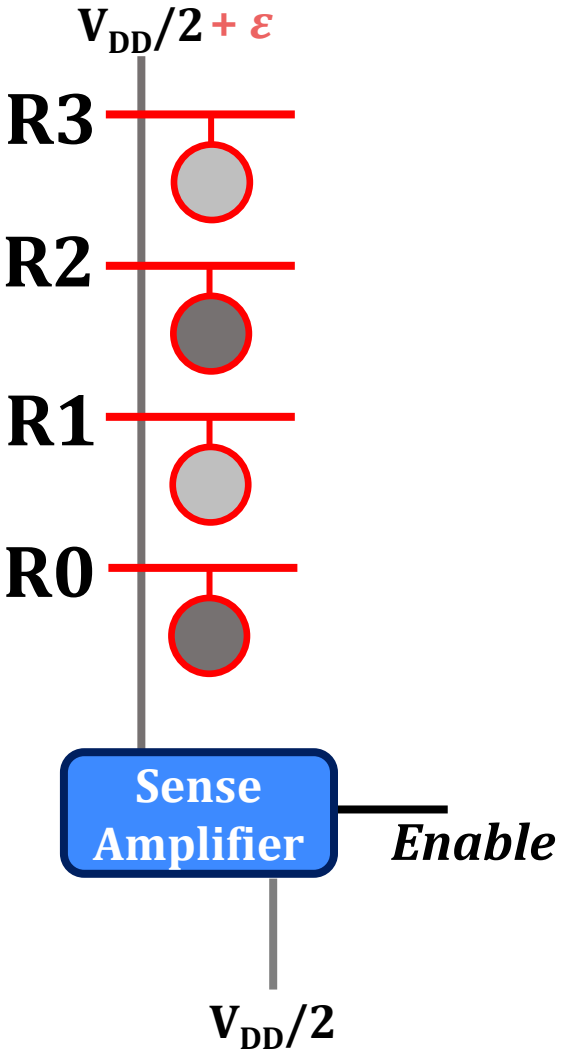
Generating Random Values via QUAC

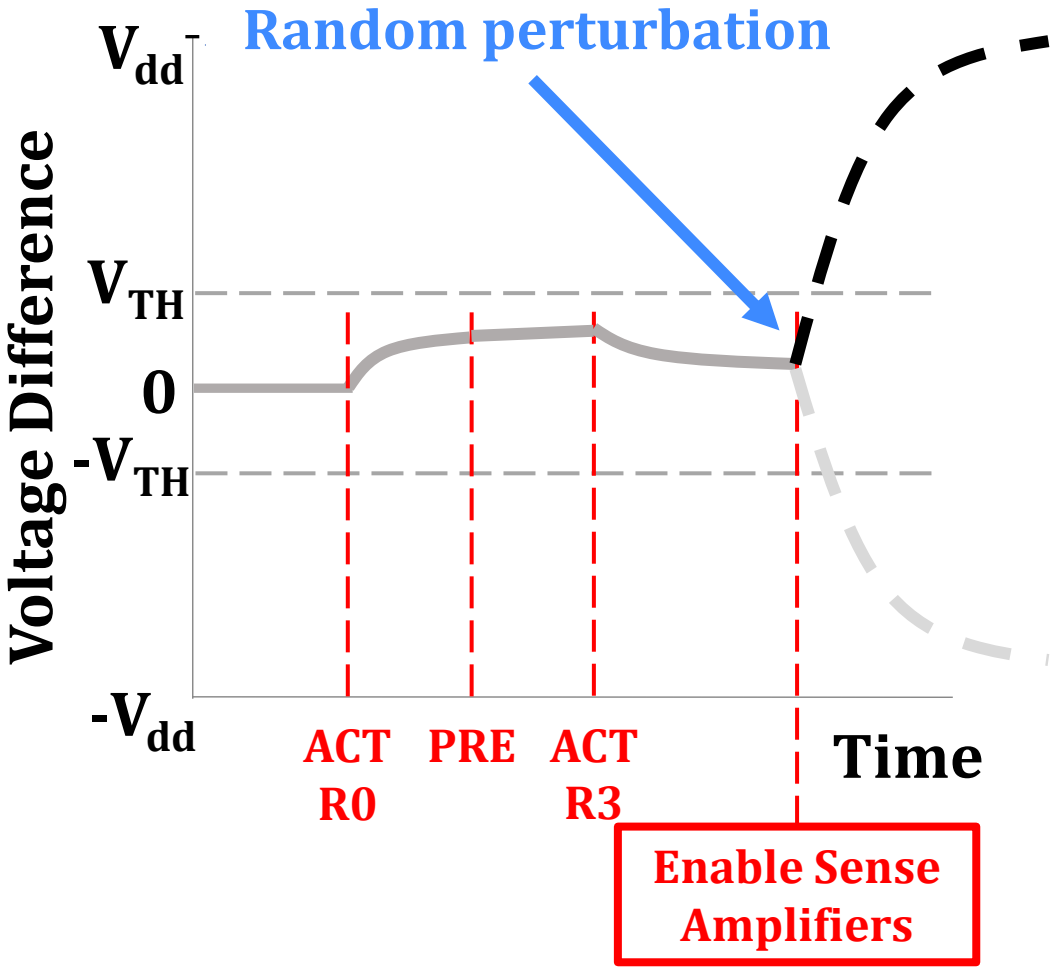




[Ataberk Olgun ISCA 21-Talk]

Generating Random Values via QUAC

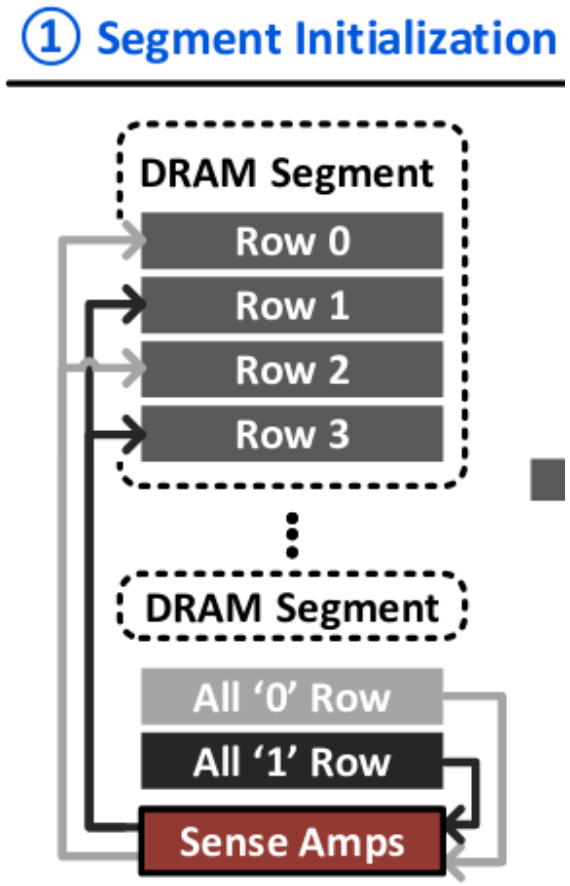




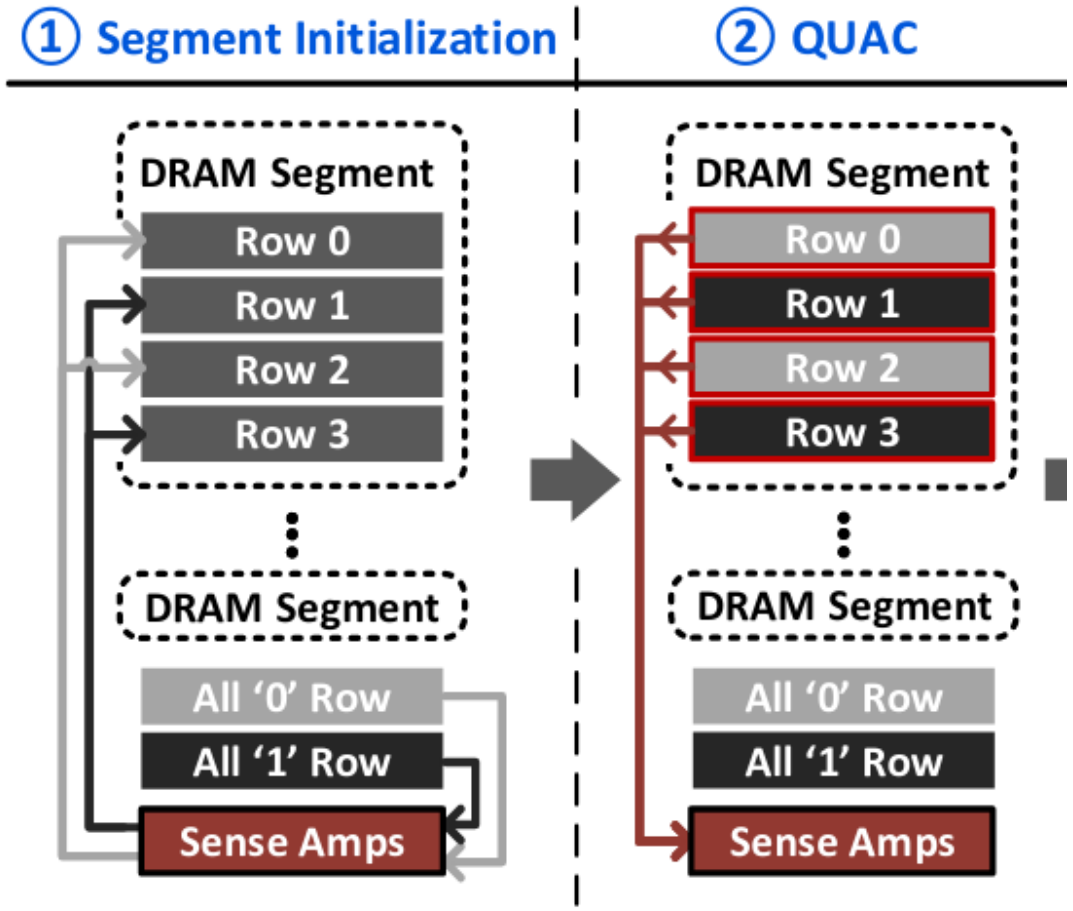
[Ataberk Olgun ISCA 21-Talk]

Mechanism

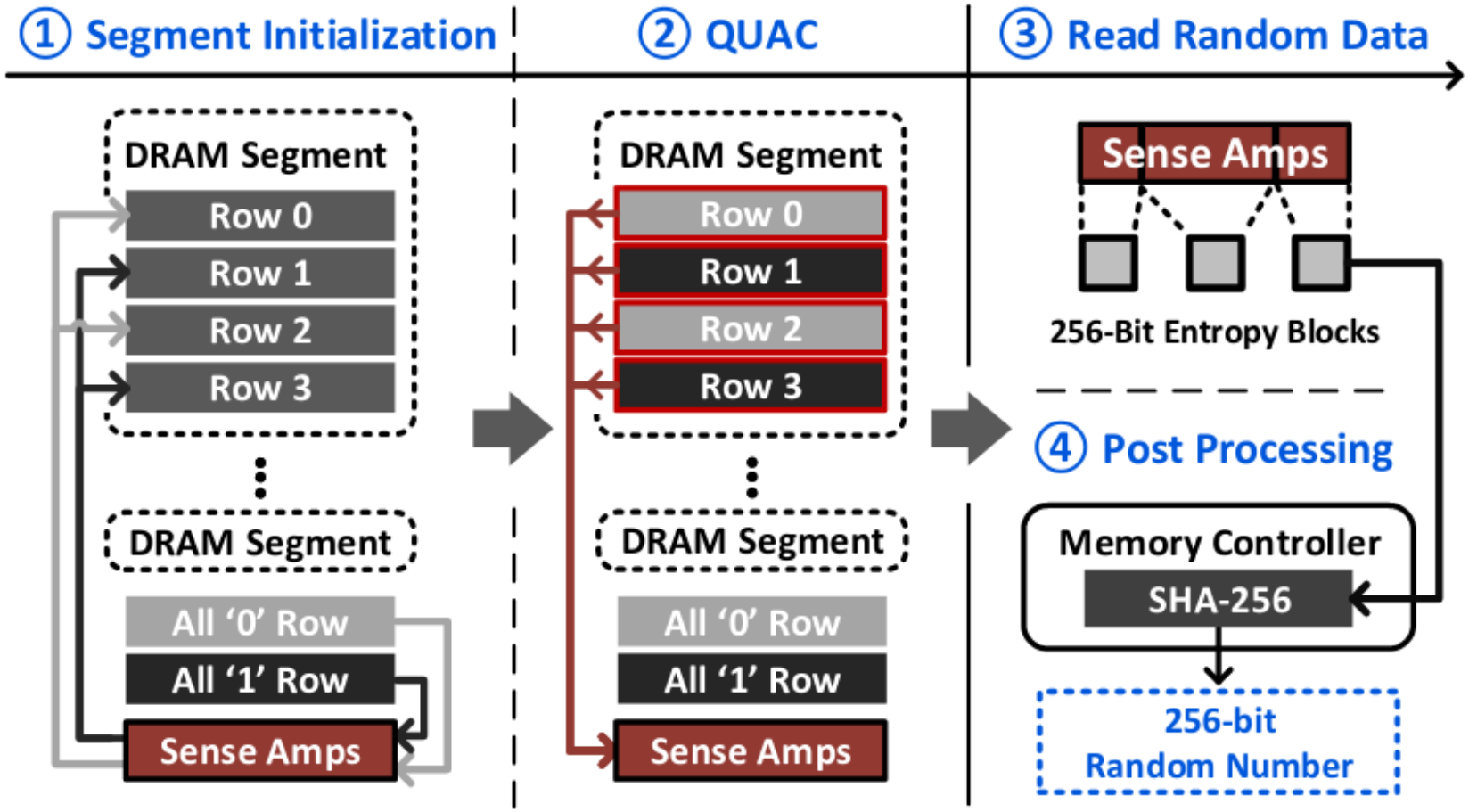
Mechanism


① Segment Initialization
DRAM Segment
Row 0
Row 1
Row 2
Row 3
DRAM Segment
All '0' Row
All '1' Row
Sense Amps

Mechanism


① Segment Initialization
② QUAC
DRAM Segment
Row 0
Row 1
Row 2
Row 3
DRAM Segment
All '0' Row
All '1' Row
Sense Amps
DRAM Segment
Row 0
Row 1
Row 2
Row 3
DRAM Segment
All '0' Row
All '1' Row
Sense Amps

Mechanism

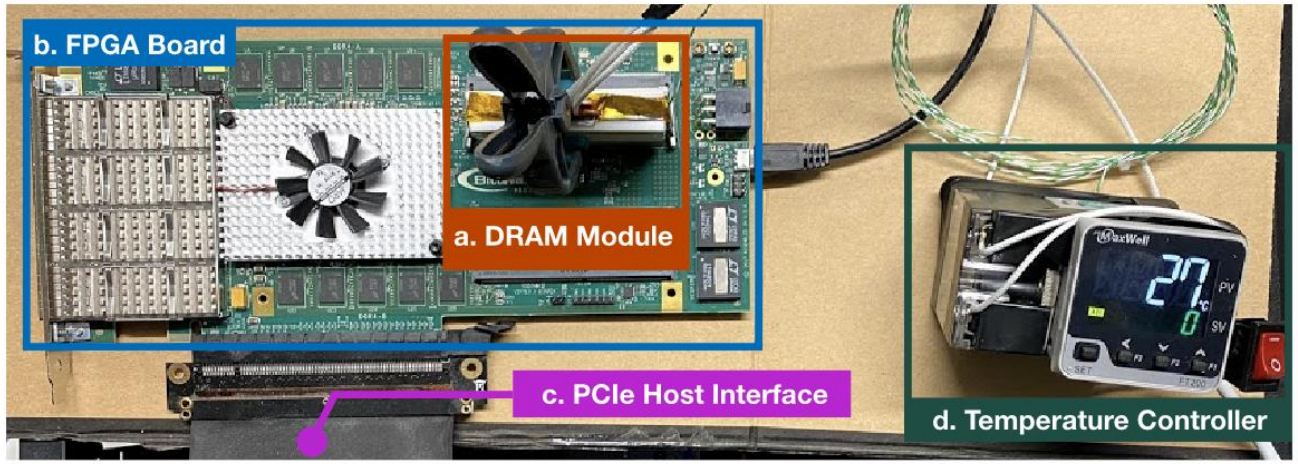

① Segment Initialization
② QUAC
③ Read Random Data
DRAM Segment
Row 0
Row 1
Row 2
Row 3
DRAM Segment
All '0' Row
All '1' Row
Sense Amps
DRAM Segment
Row 0
Row 1
Row 2
Row 3
DRAM Segment
All '0' Row
All '1' Row
Sense Amps
Sense Amps
256-Bit Entropy Blocks
④ Post Processing
Memory Controller
SHA-256
256-bit
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

SE(*1111…111*) = 0

0 < **SE(*1001…010*)** < 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

$$\mathbf{SE}(\mathit{1111...111}) = 0$$

$$0 < \mathbf{SE}(\mathit{1001...010}) < 1$$

- Perform QUAC **1000 times**
- Measure entropy **for each SAs 1000-bit bitstream**

Data Pattern Dependence

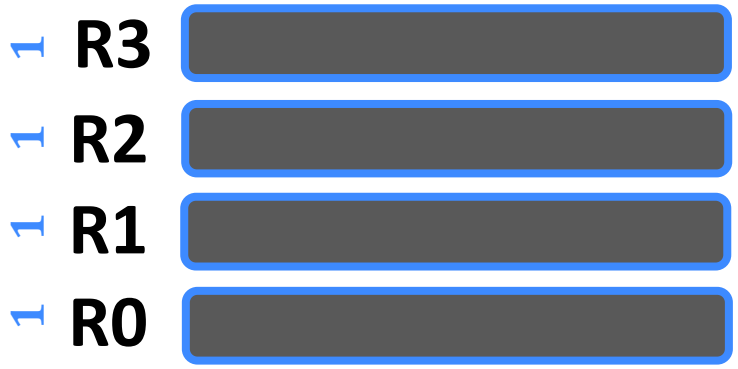
Data Pattern Dependence

- 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)



[Image: Ataberk Olgun ISCA 21-Talk]

Data Pattern Dependence

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

[Image: Ataberk Olgun ISCA 21-Talk]

Data Pattern Dependence

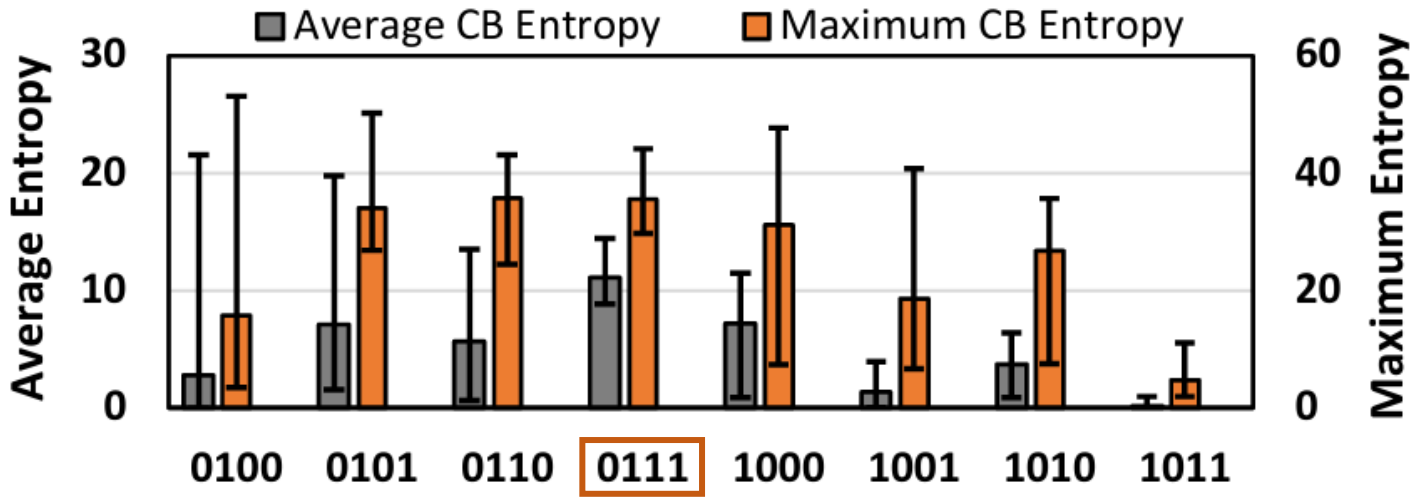
- **Cache block entropy (CBE)**: sum of entropy of all bitlines in that cache block (max: 512)

Data Pattern Dependence

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

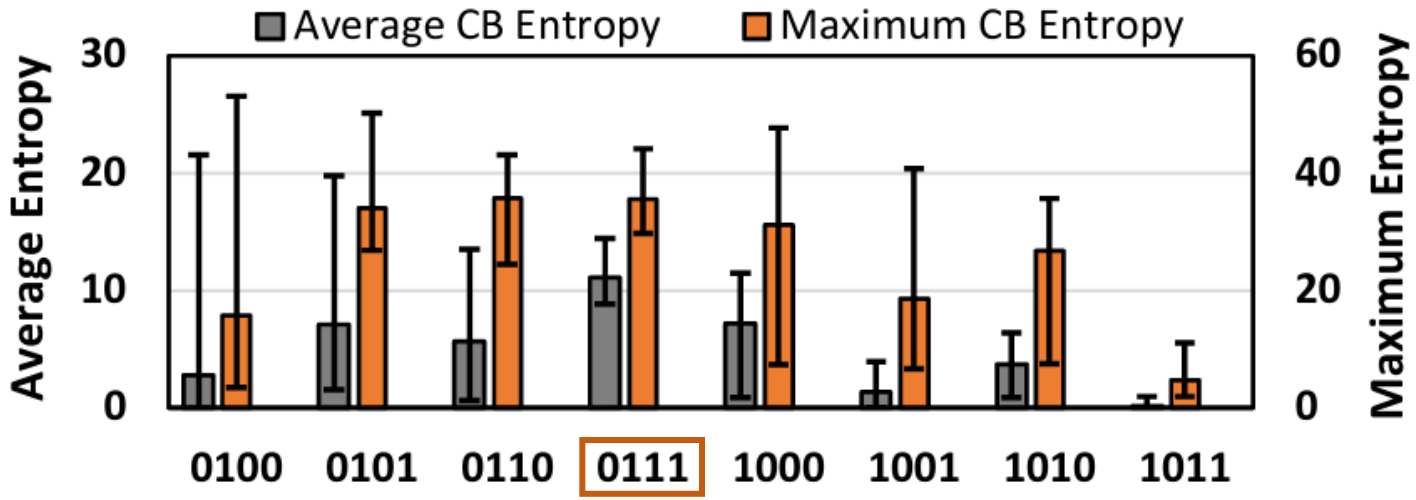
- Average CBE: average across all cache blocks in a module
- Maximum CBE: greatest of all CBEs in a module

Data Pattern Dependence



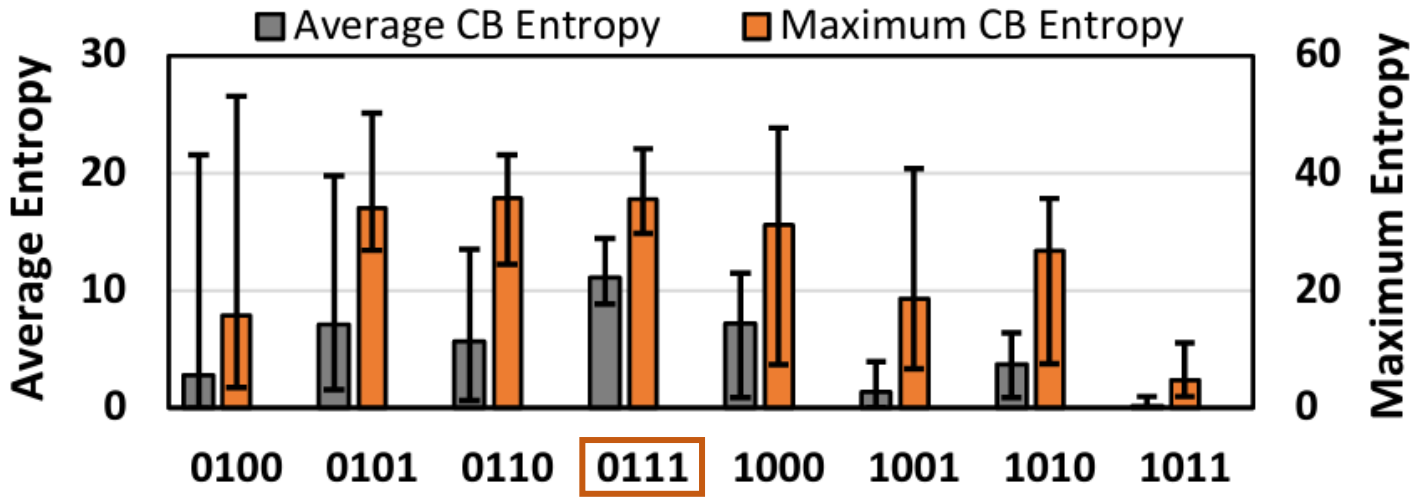
Data Pattern Dependence

- Average entropy varies with data pattern



Data Pattern Dependence

- Average entropy varies with data pattern
- More randomness when **R0 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

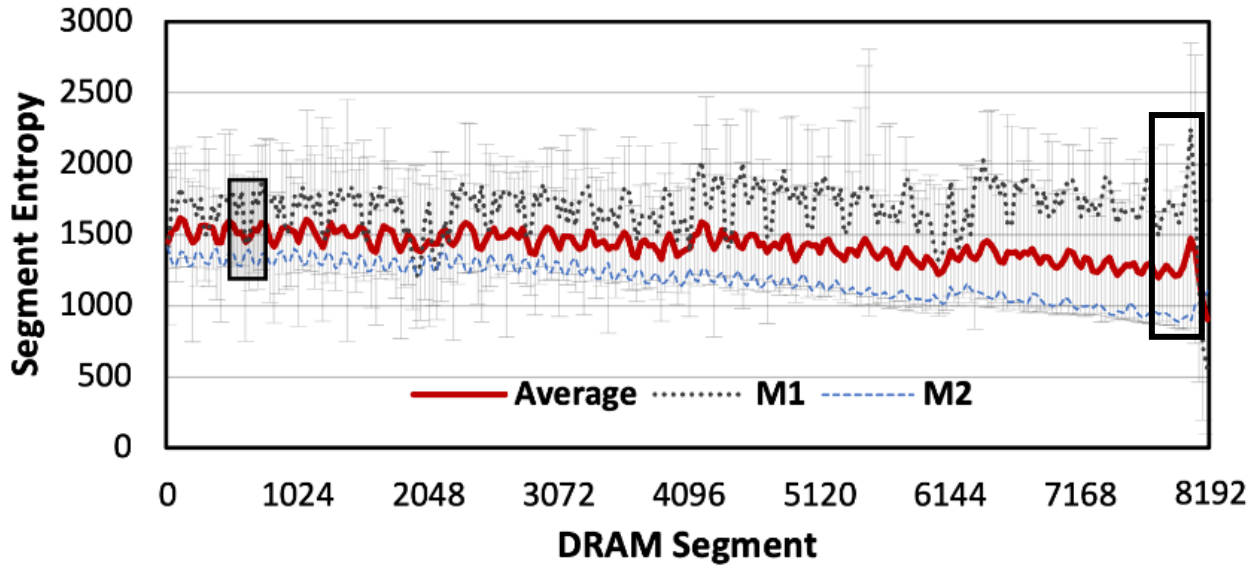
Spatial Distribution of Entropy

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)

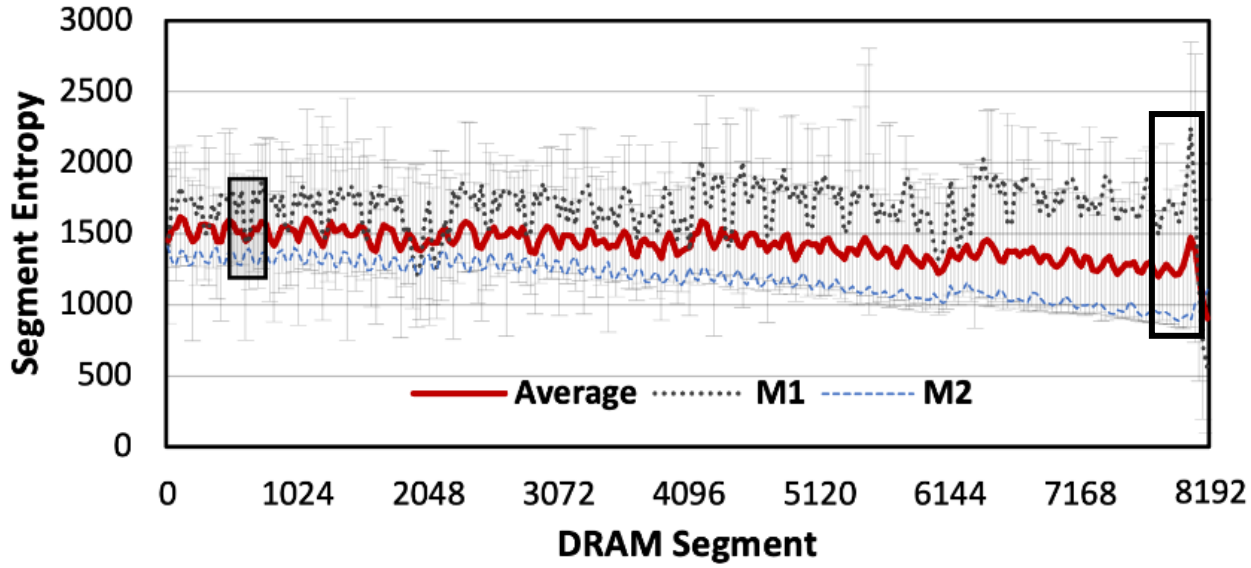
Data pattern: "0111"

Spatial Distribution of Entropy



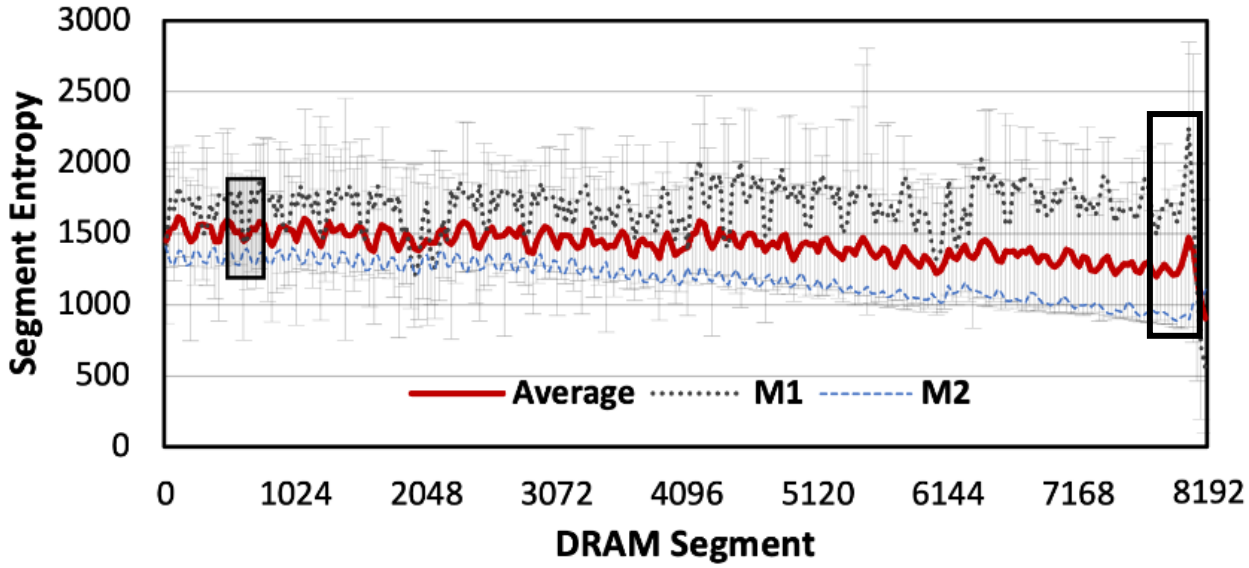
Spatial Distribution of Entropy

- Wave like pattern: Systematic process variations



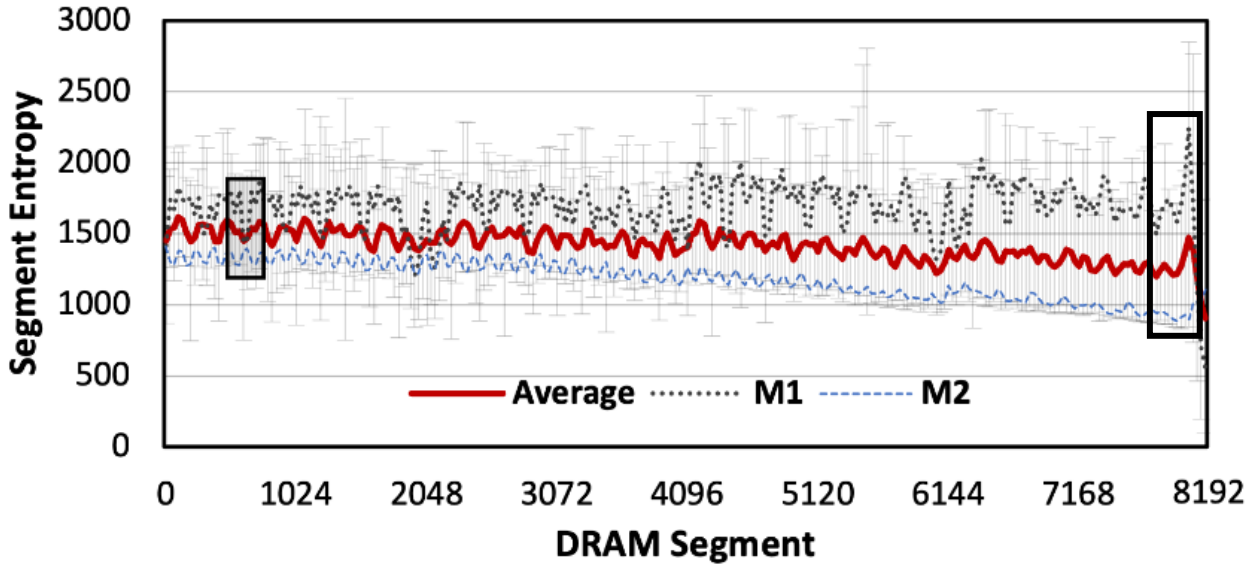
Spatial Distribution of Entropy

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



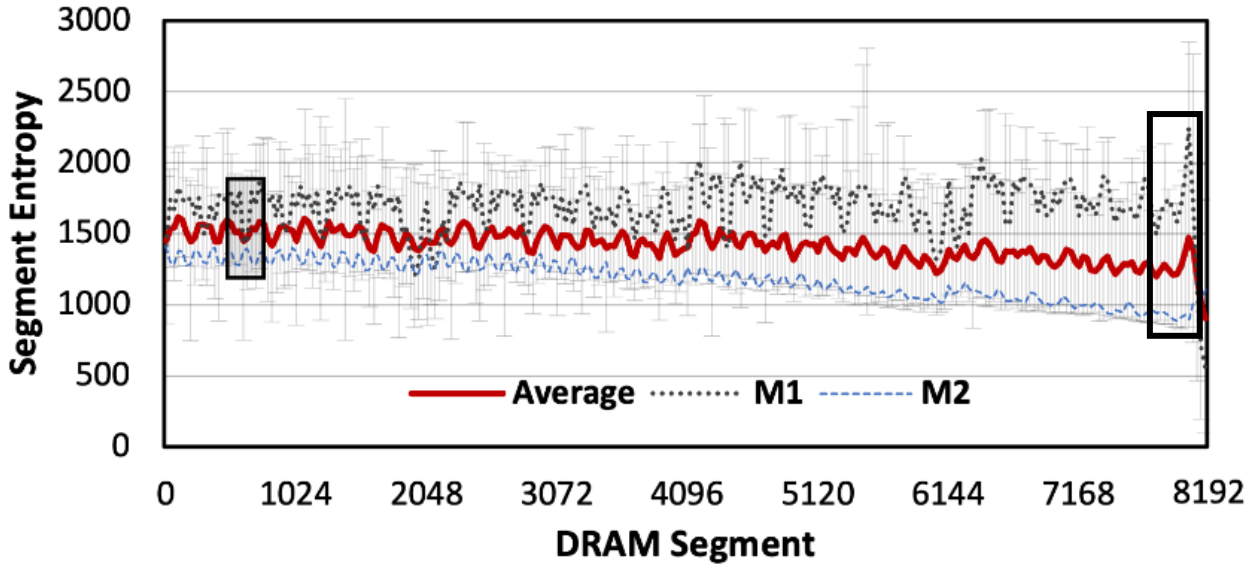
Spatial Distribution of Entropy

- 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

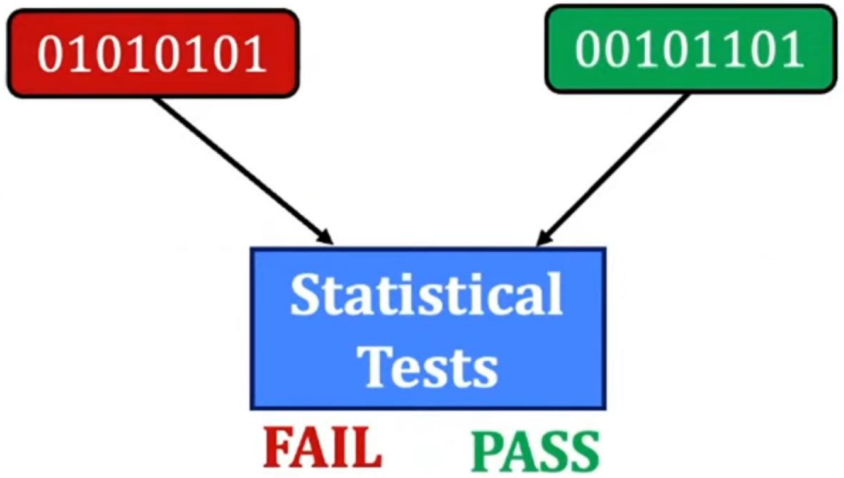
1. Initialize **highest entropy** DRAM segments with data pattern "0111"
2. Perform QUAC
3. Read out the values generated in the SAs, split into 256-bit blocks

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

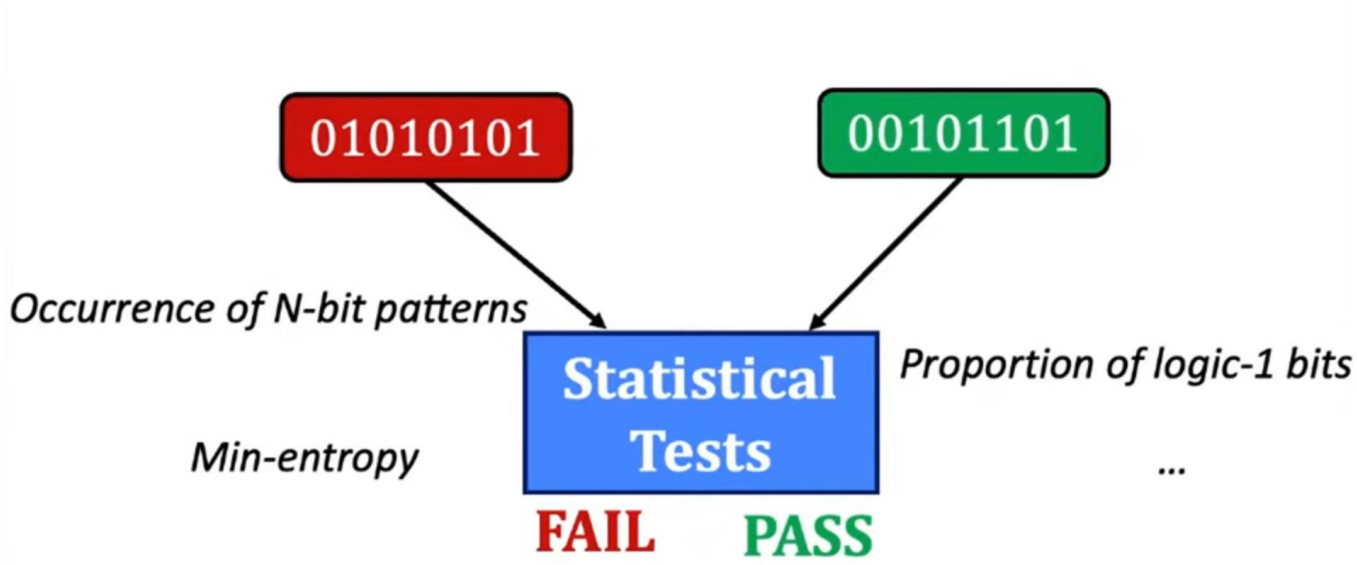
- Measure **quality of TRNG**



[Image: Ataberk Olgun ISCA 21-Talk]

NIST Statistical Test Suite

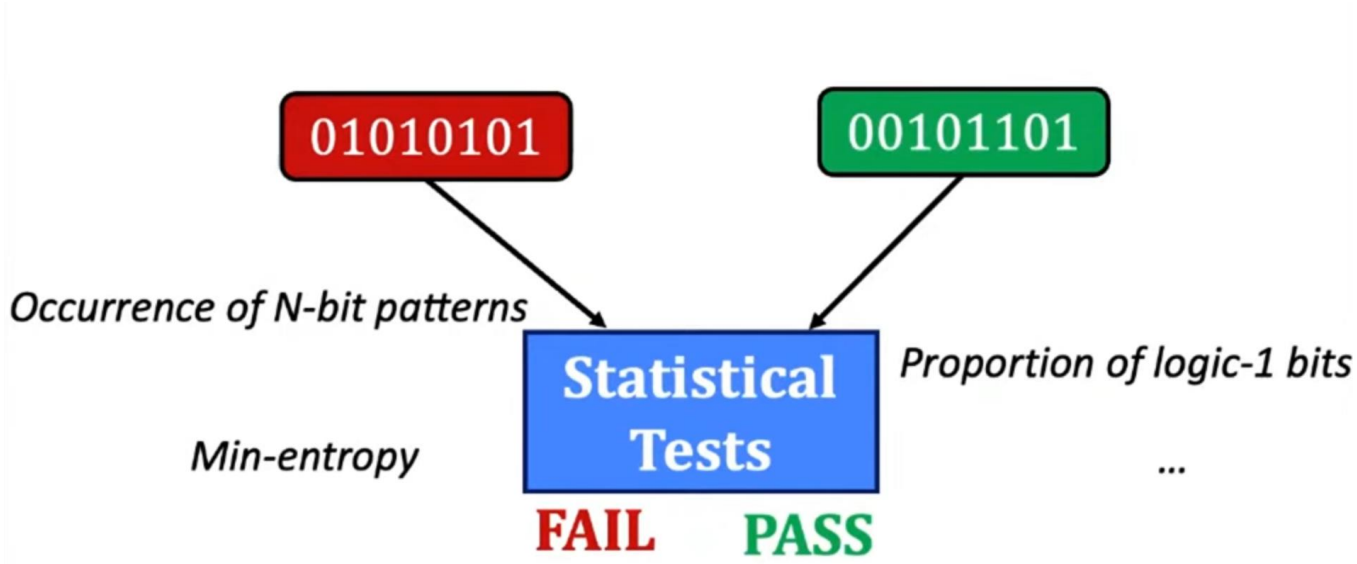
- Measure **quality of TRNG**
- Runs multiple tests, evaluates statistical properties to find patterns



[Image: Ataberk Olgun ISCA 21-Talk]

NIST Statistical Test Suite

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



[Image: Ataberk Olgun ISCA 21-Talk]

NIST STS Results

Average p-value for each test

Table 1: NIST STS Randomness Test Results

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

*VNC: Von Neumann Corrector

NIST STS Results

Desired: **p-value > 0.001**

- Both bitstreams pass all tests
- QUAC-TRNG outputs high quality random bitstreams

Average p-value for each test

Table 1: NIST STS Randomness Test Results

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

*VNC: Von Neumann Corrector

QUAC-TRNG Throughput

QUAC-TRNG Throughput

Throughput of

$$(256 \cdot SIB)/(L \cdot 10^{-9})$$

bits per second

QUAC-TRNG Throughput

Throughput of

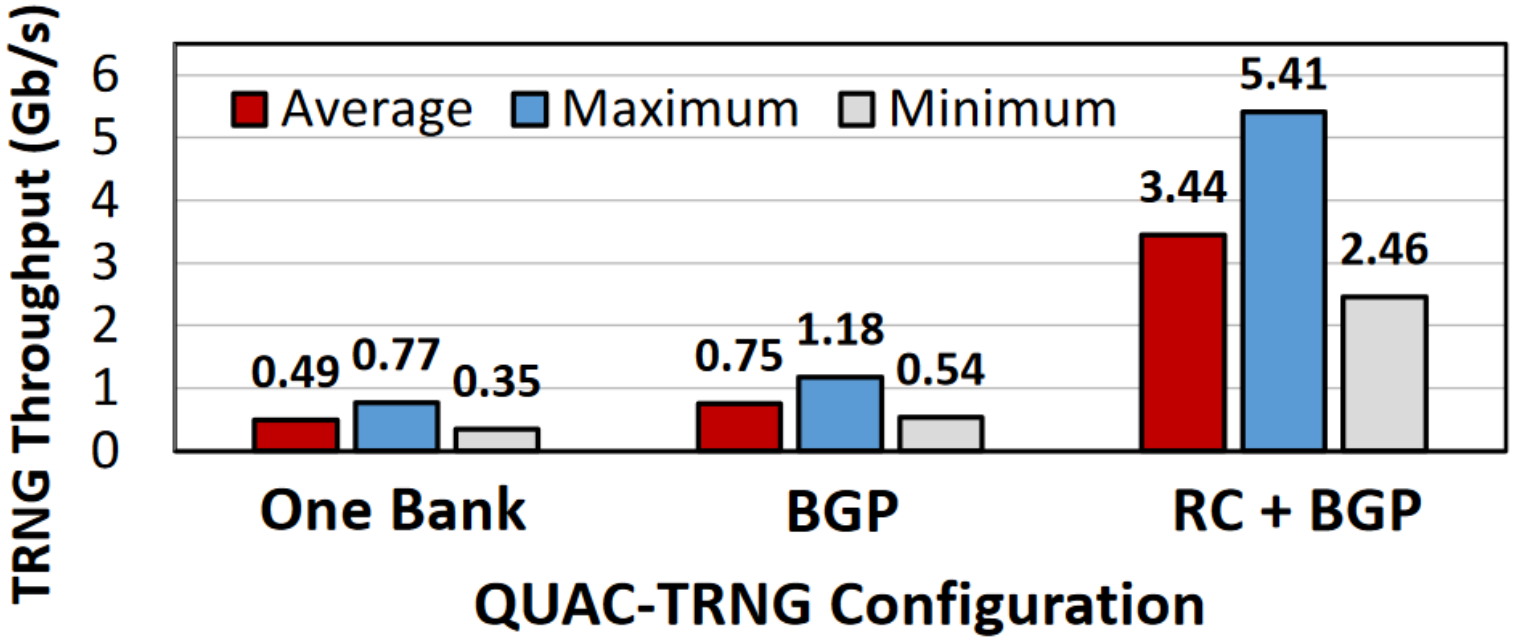
$$(256 \cdot SIB)/(L \cdot 10^{-9})$$

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

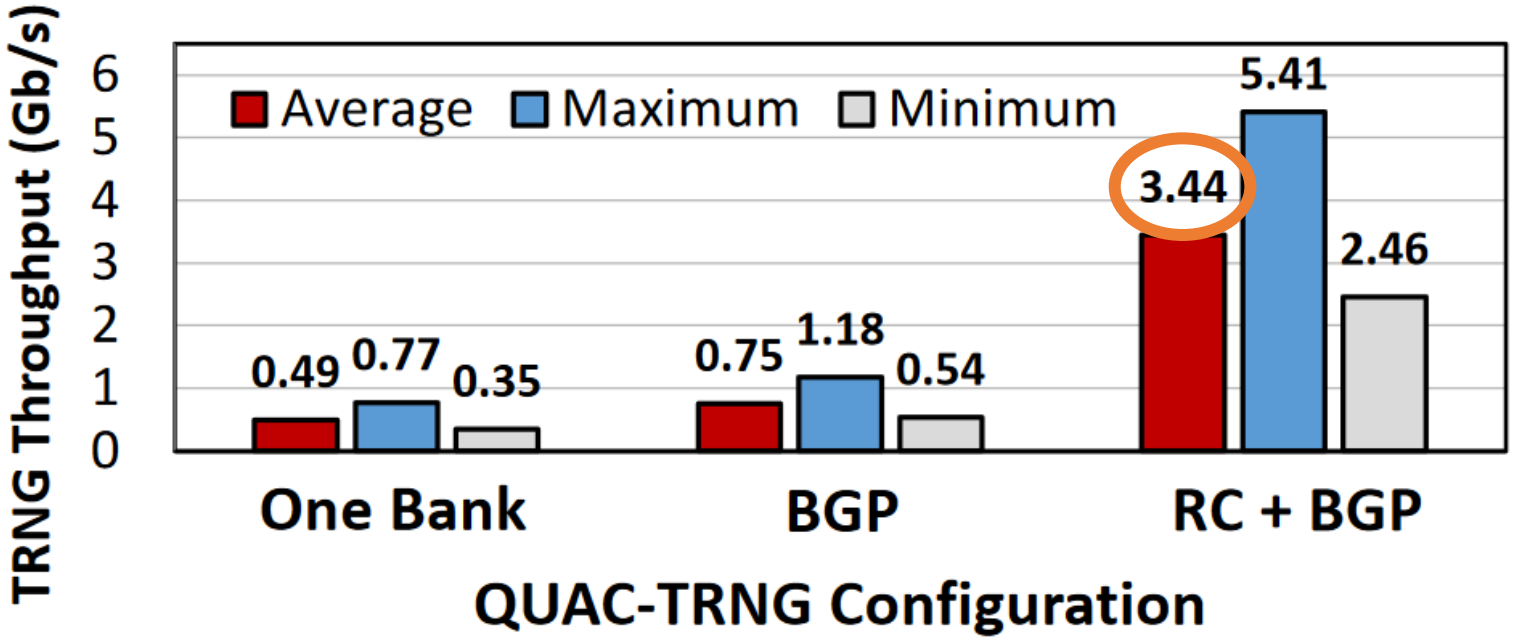


BGP: 4 banks from different groups to overlap DRAM command latencies

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

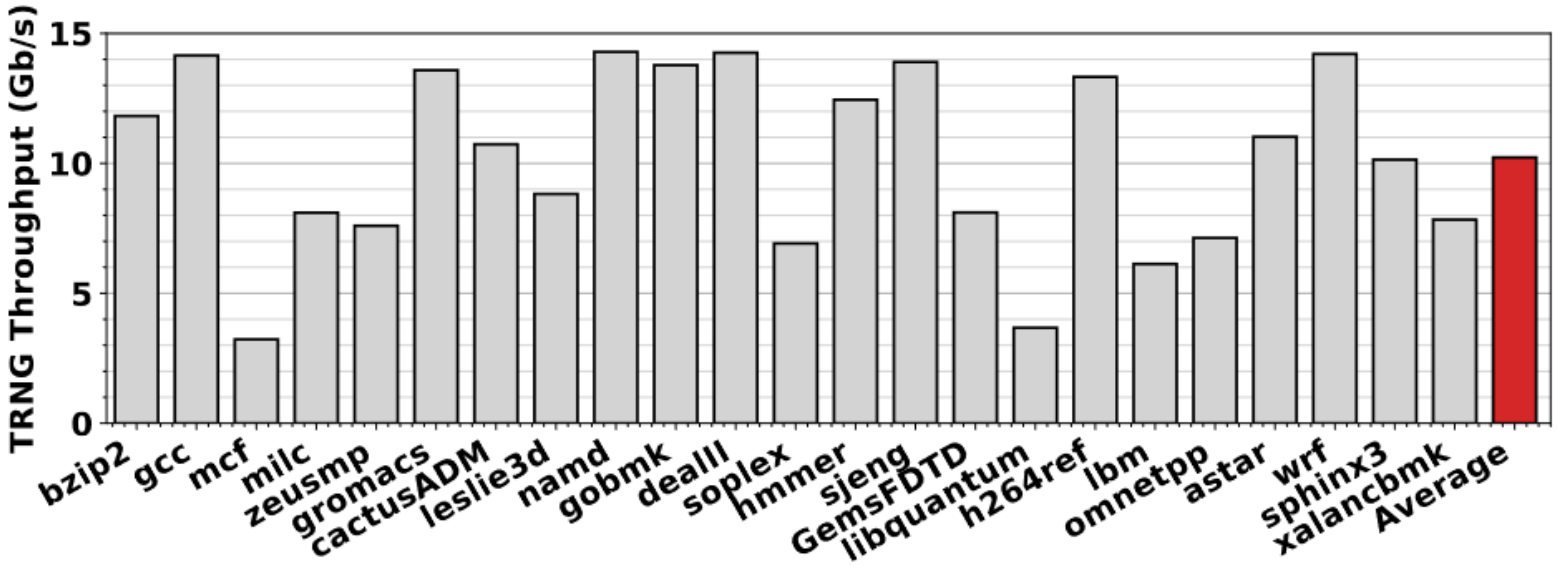
System Performance

- SPEC2006 benchmark suite
- Simulate 3.2 GHz core, 4 DRAM channels

System Performance

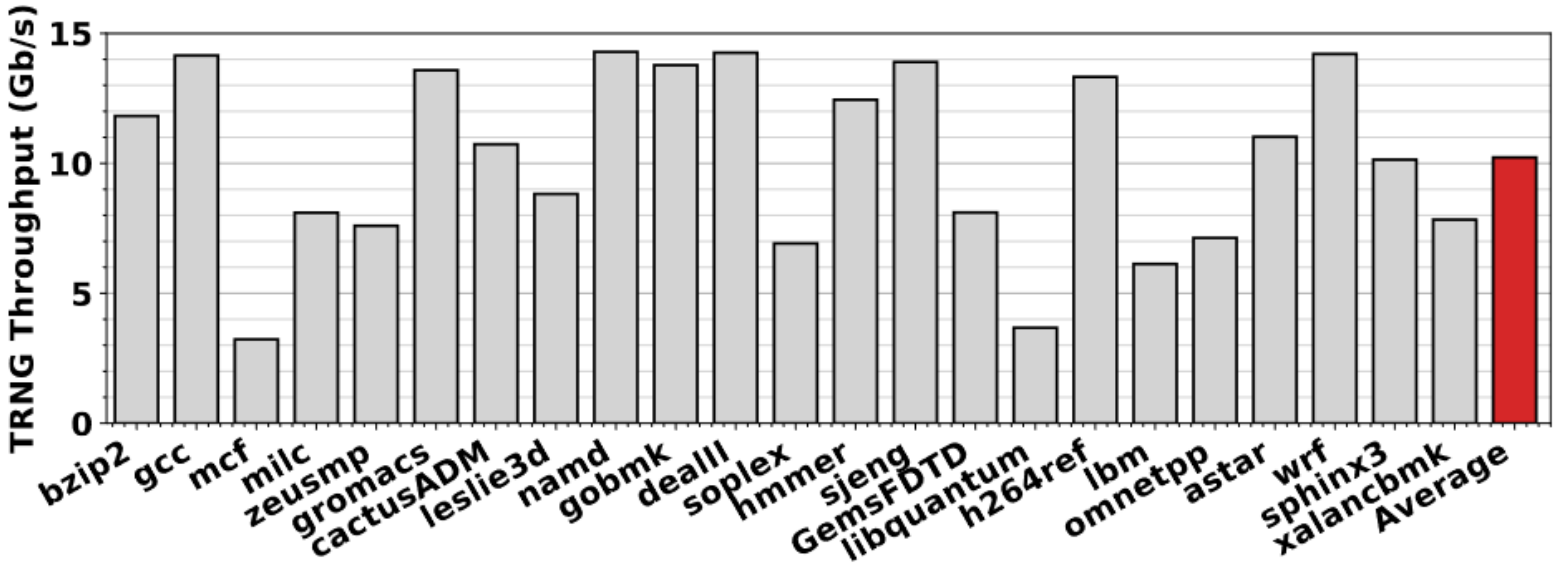
- SPEC2006 benchmark suite
- Simulate 3.2 GHz core, 4 DRAM channels
- Calculate when channel idle
  - Issue QUAC-TRNG commands in idle periods

Results


TRNG Throughput (Gb/s)
0
5
10
15
bzip2
gcc
mcf
milc
zeusmp
gromacs
cactusADM
leslie3d
namd
gobmk
dealII
soplex
hmmer
sjeng
GemsFDTD
libquantum
h264ref
lbm
omnetpp
astar
wrf
sphinx3
xalancbmk
Average

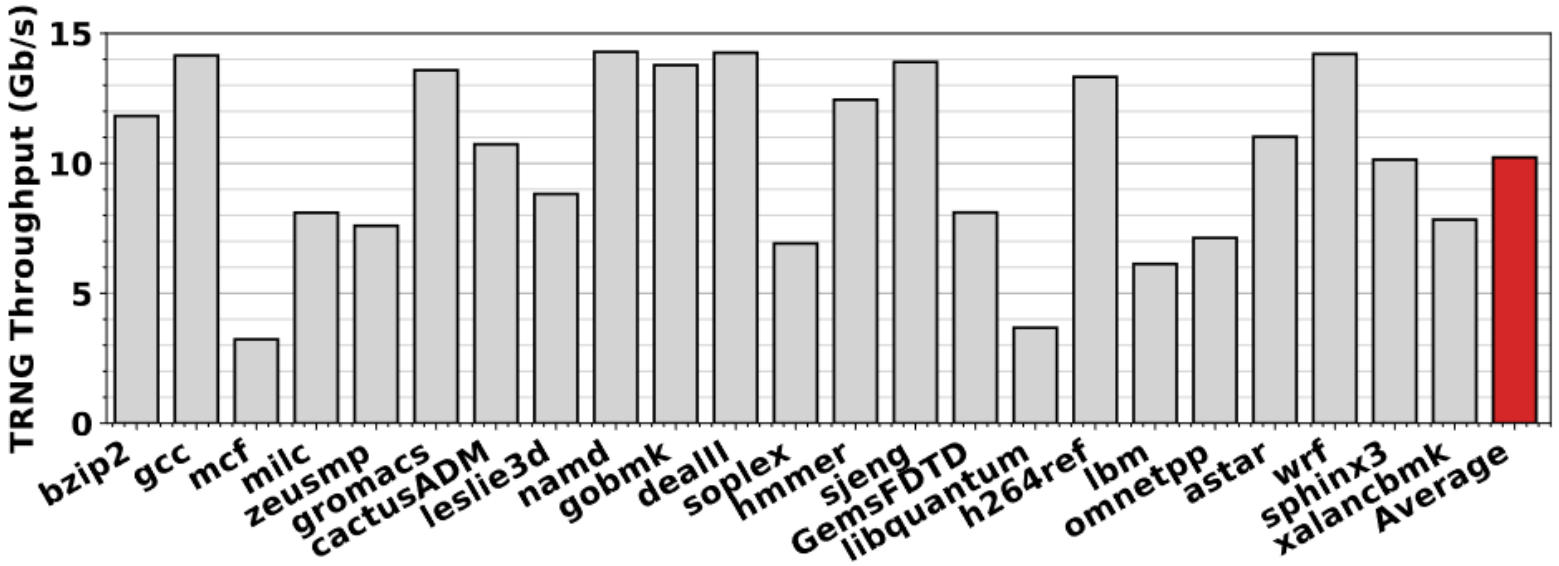
Results

- 10.2 Gb/s average throughput



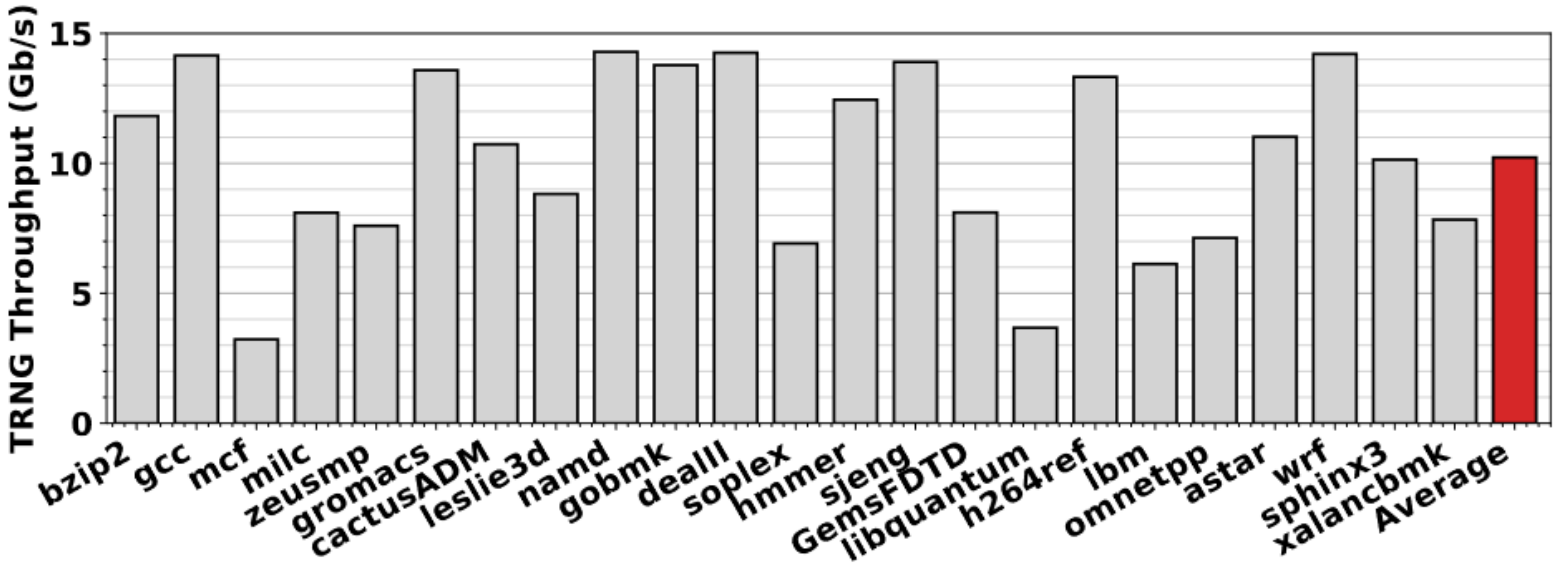
Results

- **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

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

Comparison With Prior Work

- Throughput and latency on 4 DRAM channels

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

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[†§] Ataberk Olgun[†§] Lois Orosa[§] A. Giray Yağlıkçı[§]
Jeremie S. Kim[§] Hasan Hassan[§] Oğuz Ergin[†] Onur Mutlu[§]

[†]*TOBB University of Economics and Technology* [§]*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)

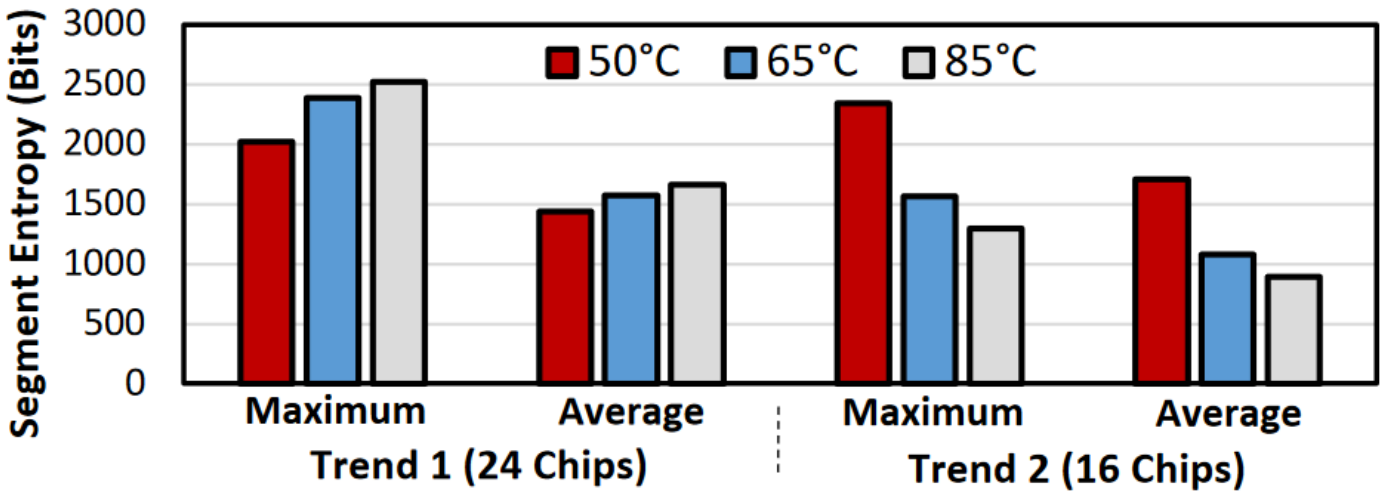
DRAM Chip Requirements

- 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

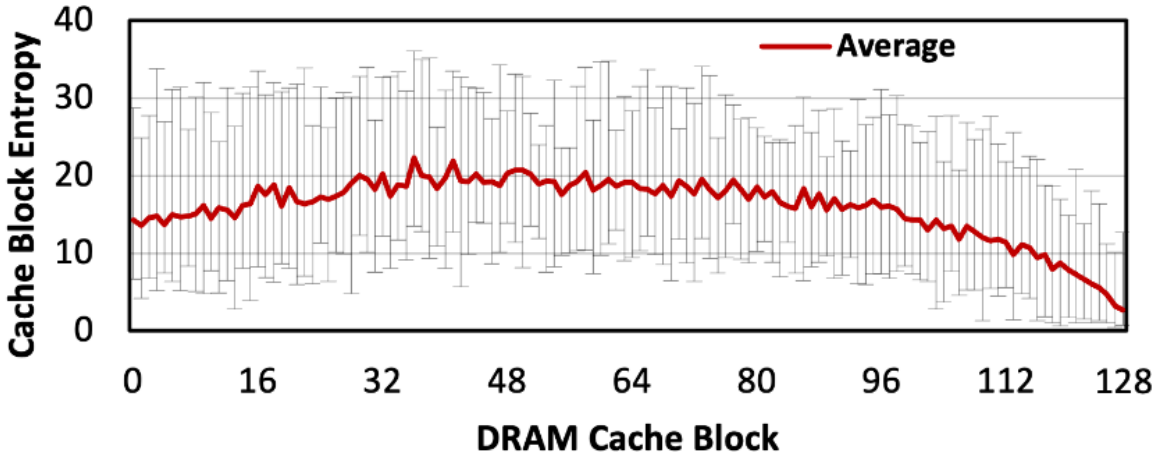
Spatial Distribution of Entropy

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

Spatial Distribution of Entropy

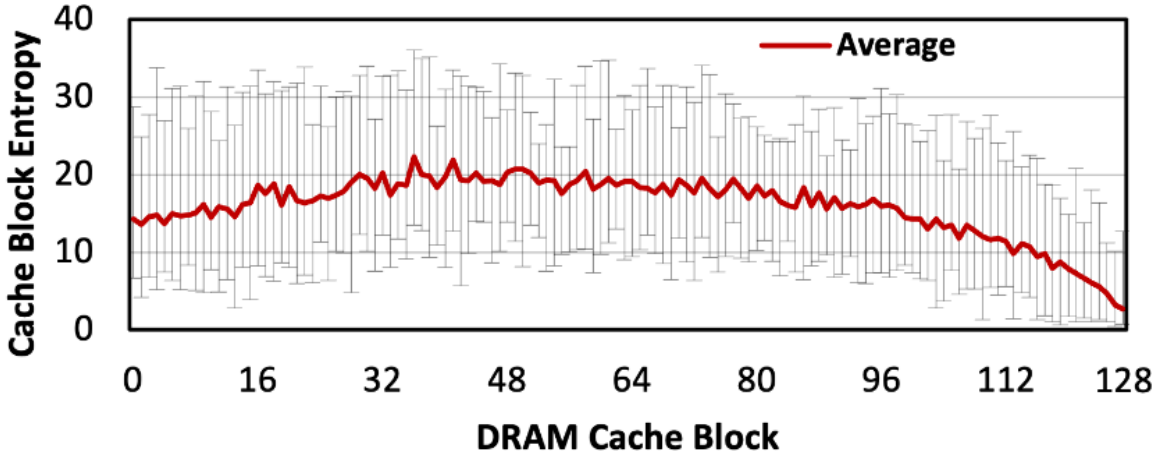
- **Cache block entropy:** sum of entropy of all bitlines in that cache block
- Cache block in **highest entropy segment** (in each module)
- Data pattern: "0111"

Spatial Distribution of Entropy



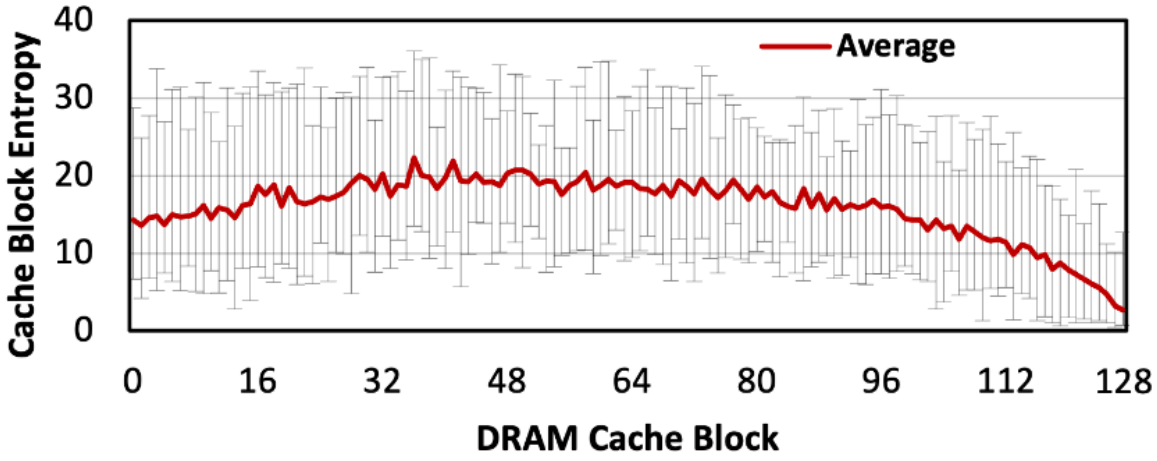
Spatial Distribution of Entropy

- **Peaks** around the **middle**



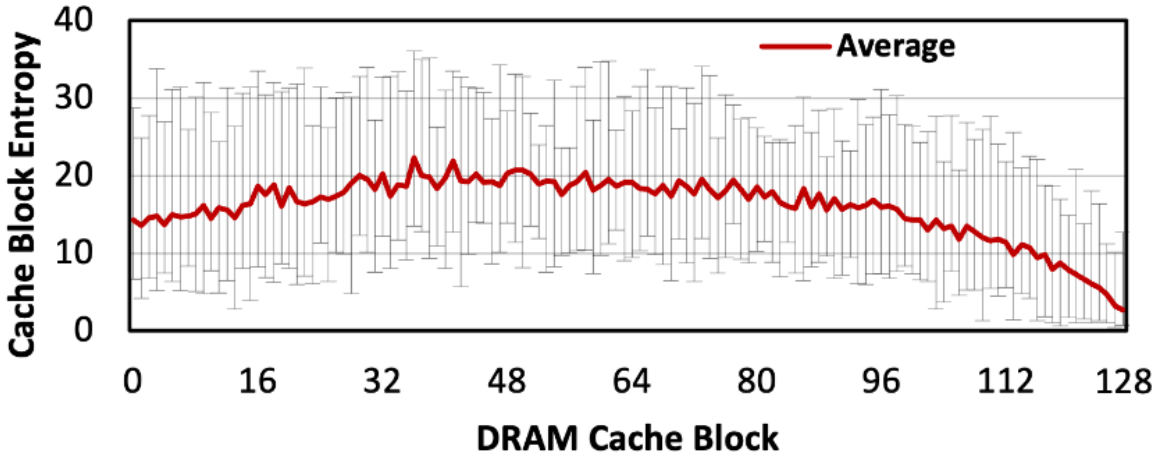
Spatial Distribution of Entropy

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



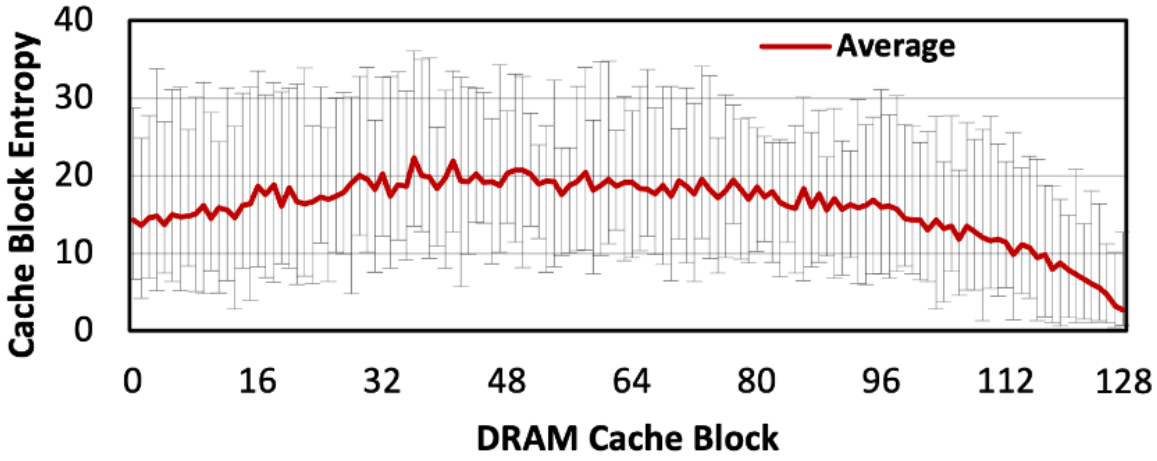
Spatial Distribution of Entropy

- **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 >= *Period Threshold*
  - Short: # of cycles < *Period Threshold*
  - *Period Threshold e*mpirically 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-α) Q(s,a) + α ·r* (*α= 0,05;* 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

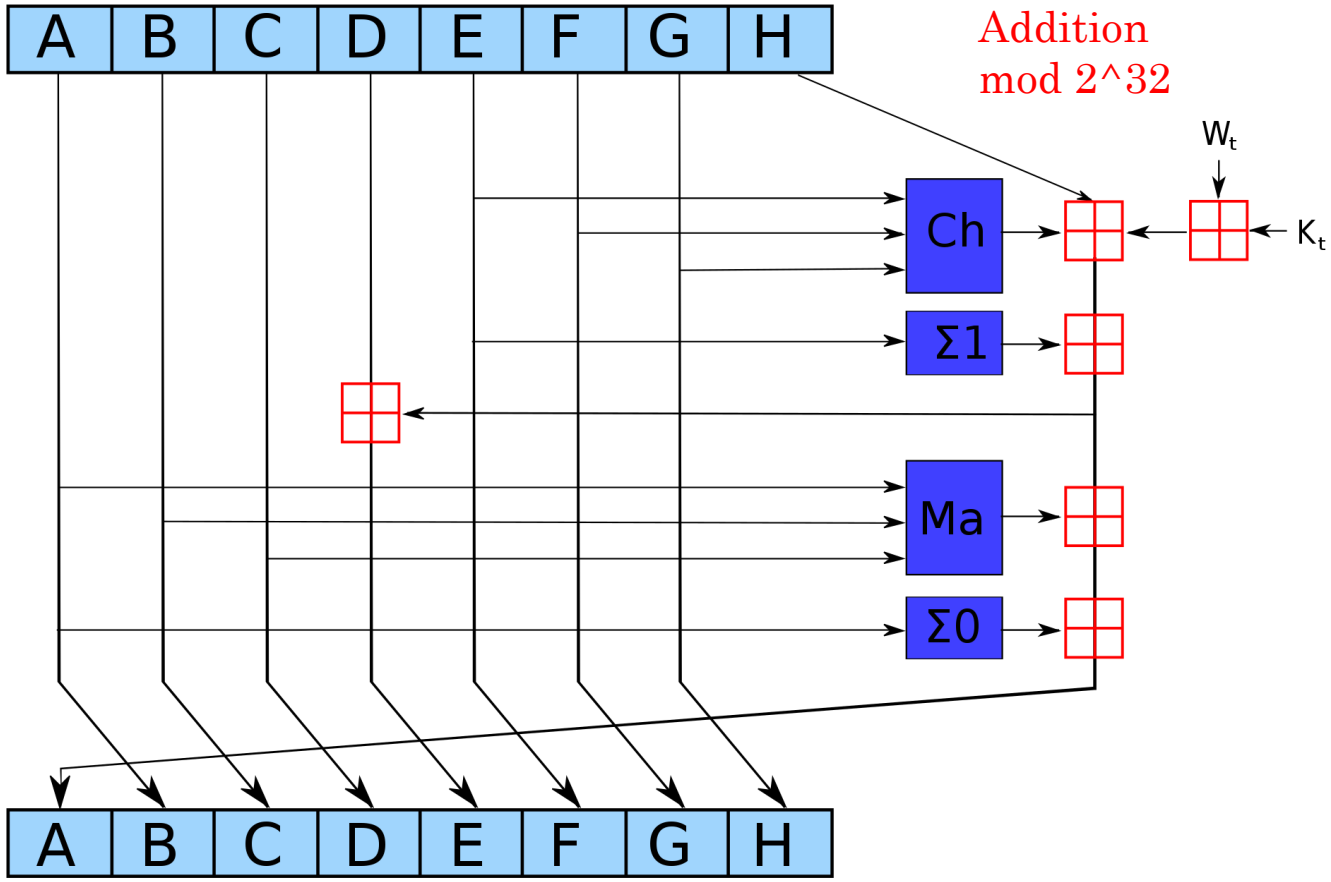
- Secure Hashing Algorithm
- Input padded to 512 bits
- Divide input into 32-bit words: *M0…Mn*
- Process the input for each *Mi*
- 8 buffered A, B, C, D, E, F, G, H of 32 bits each are used
  - Values are fixed at the beginning

SHA-256

Process each 16 words for 64 rounds

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

- Wt: different depending on the repetition, for the first 16 rounds it's the input message
- Kt: has a new fixed value for each round



Source: https://en.wikipedia.org/wiki/SHA-2

DDR4 Modules

Module	Module Identifier	Chip Identifier	Freq. (MT/s)	Organization			Segment Entropy		
				Size (GB)	Chips	Pins	Avg.	Max.†	Avg. (after 30 days)
$M1$	*Unknown*	H5AN4G8NAFR-TFC	2133	4	8	x8	1688.1	2247.4	–
$M2$	*Unknown*	*Unknown*	2133	4	8	x8	1180.4	1406.1	–
$M3$	*Unknown*	H5AN4G8NAFR-TFC	2133	4	8	x8	1205.0	1858.3	1192.9
$M4$	76TT21NUS1R8-4G	H5AN4G8NAFR-TFC	2133	4	8	x8	1608.1	2406.5	1588.0
$M5$	*Unknown*	T4D5128HT-21	2133	4	8	x8	1618.2	2121.6	–
$M6$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1211.5	1444.6	–
$M7$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1177.7	1404.4	–
$M8$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1332.9	1600.9	1407.0
$M9$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1137.1	1370.9	–
$M10$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1208.5	1473.2	1251.8
$M11$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1176.0	1382.9	1165.1
$M12$	TLRD44G2666HC18F-SBK	H5AN4G8NMFR-VKC	2666	4	8	x8	1485.0	1740.6	–
$M13$	KSM32RD8/16HDR	H5AN4G8NAFA-UHC	2400	4	8	x8	1853.5	2849.6	–
$M14$	F4-2400C17S-8GNT	H5AN4G8NMFR-UHC	2400	8	8	x8	1369.3	1942.2	–
$M15$	F4-2400C17S-8GNT	H5AN4G8NMFR-UHC	3200	8	8	x8	1545.8	2147.2	–
$M16$	KSM32RD8/16HDR	H5AN8G8NDJR-XNC	3200	16	8	x8	1634.4	1944.6	–
$M17$	KSM32RD8/16HDR	H5AN8G8NDJR-XNC	3200	16	8	x8	1664.7	2016.6	–

†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* α
- Here: α=0.001

NIST STS Results

Desired: **p-value > 0.001**

- Both bitstreams pass all tests
- QUAC-TRNG outputs 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%)

Average p-value for each test

Table 1: NIST STS Randomness Test Results

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

*VNC: Von Neumann Corrector