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 TOBB ETÜ University of Economics & Technology 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** → **QU**adruple **AC**tivation **(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

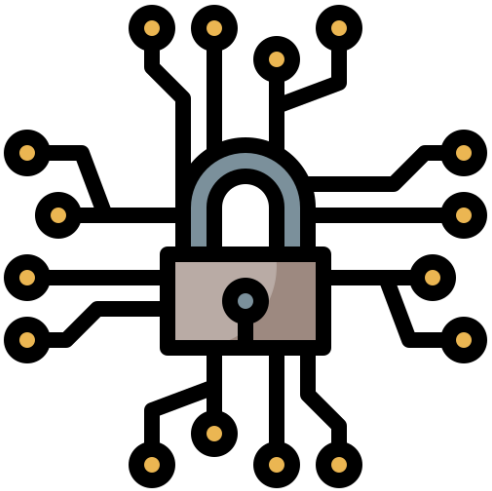
QUadruple ACtivation (QUAC)

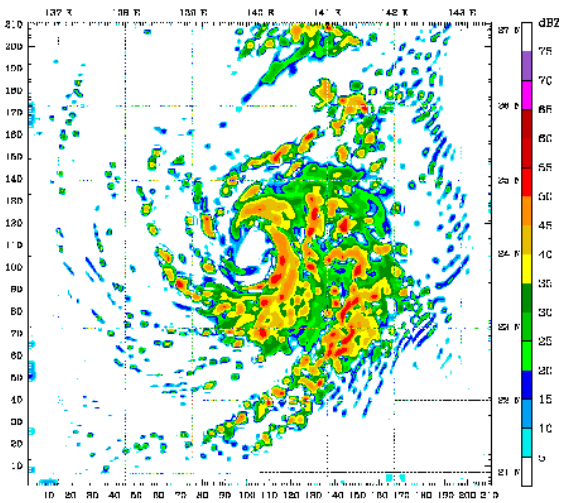
QUAC-TRNG

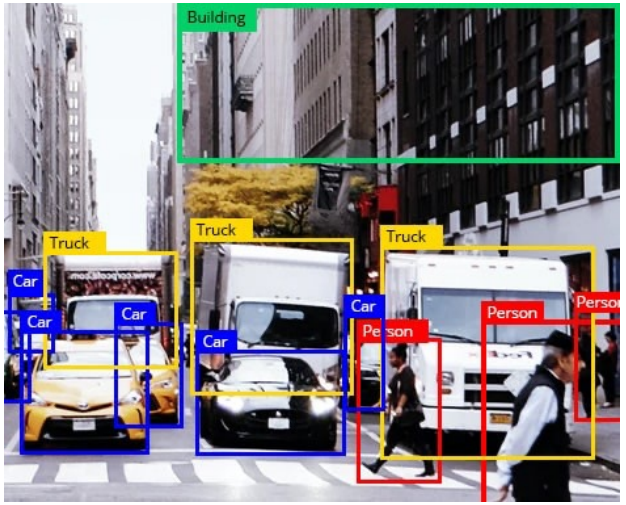
Evaluation

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

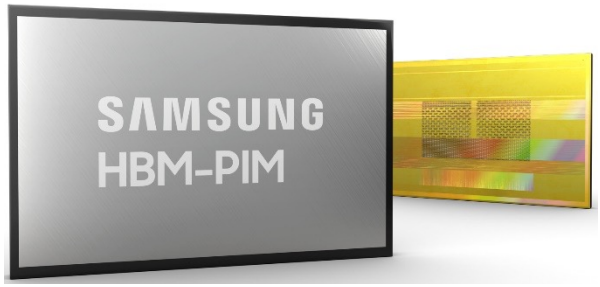
High-throughput: > Gb/s throughput

- Open application space that require high-throughput TRNG

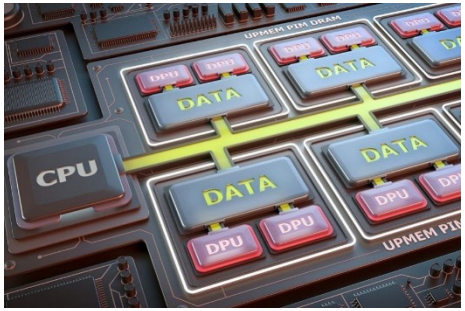
Synergy with Processing-in-Memory

Processing-in-Memory (PIM) Systems

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



[Samsung]



[UPMEM]

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

Outline

True Random Numbers in DRAM

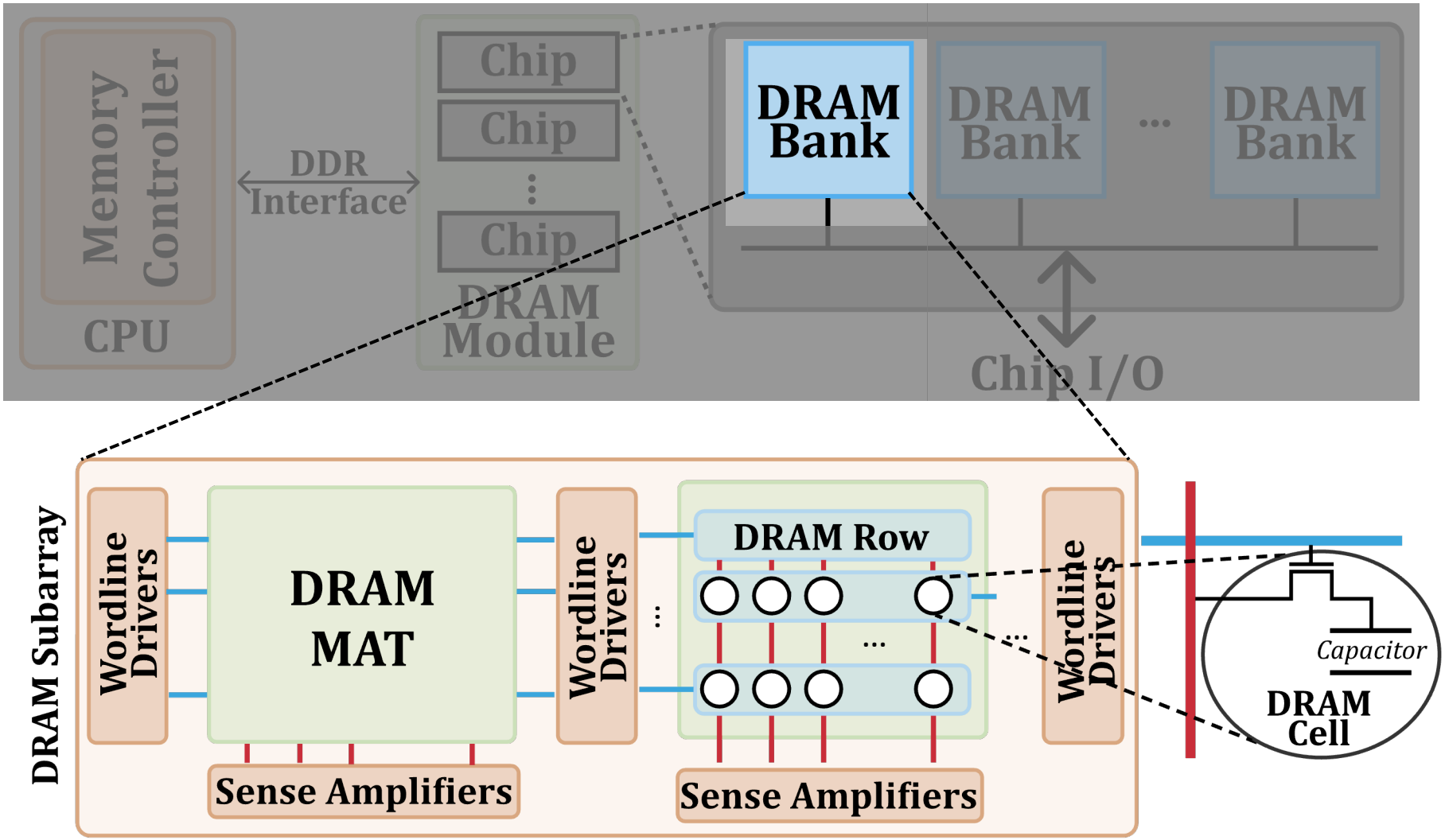
DRAM Organization and Operation

QUadruple ACtivation (QUAC)

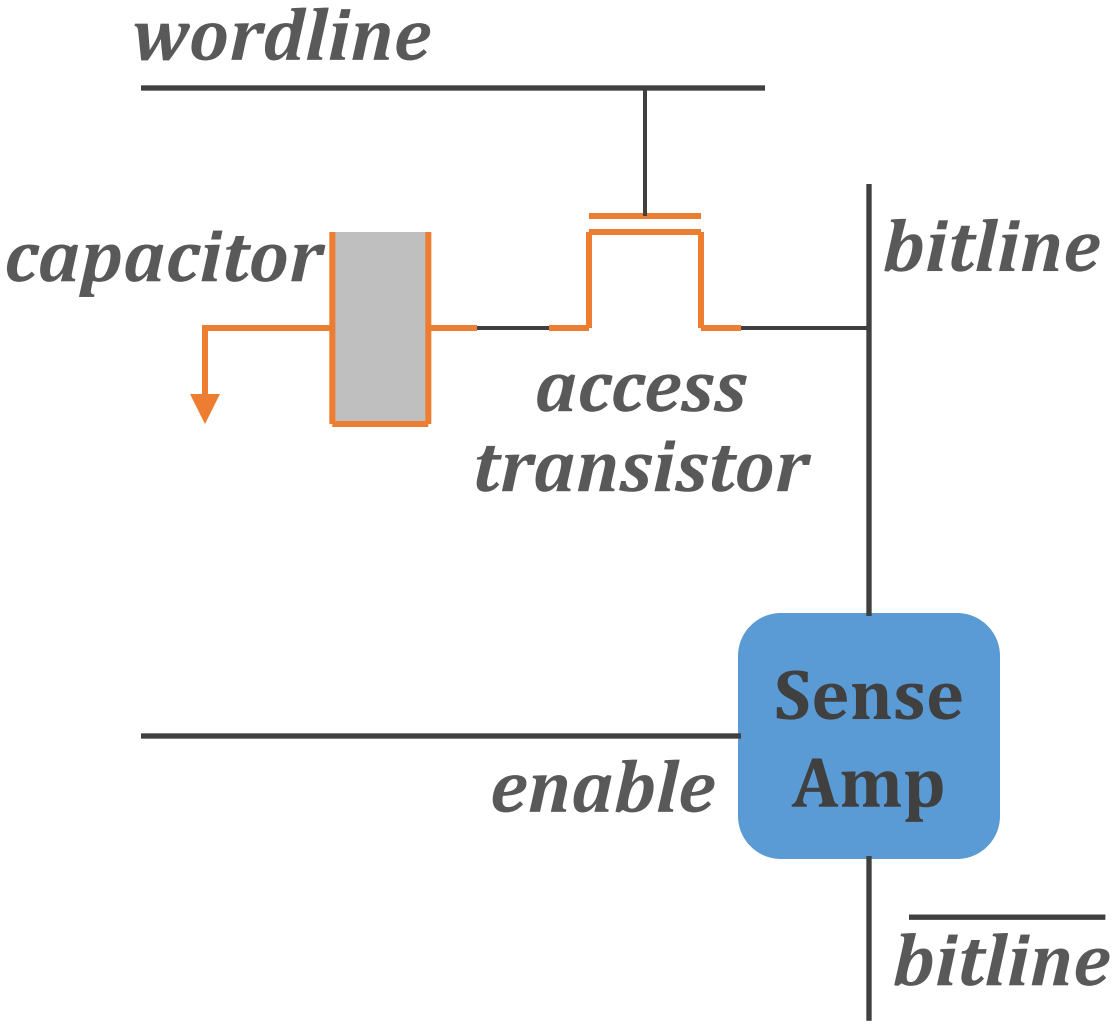
QUAC-TRNG

Evaluation

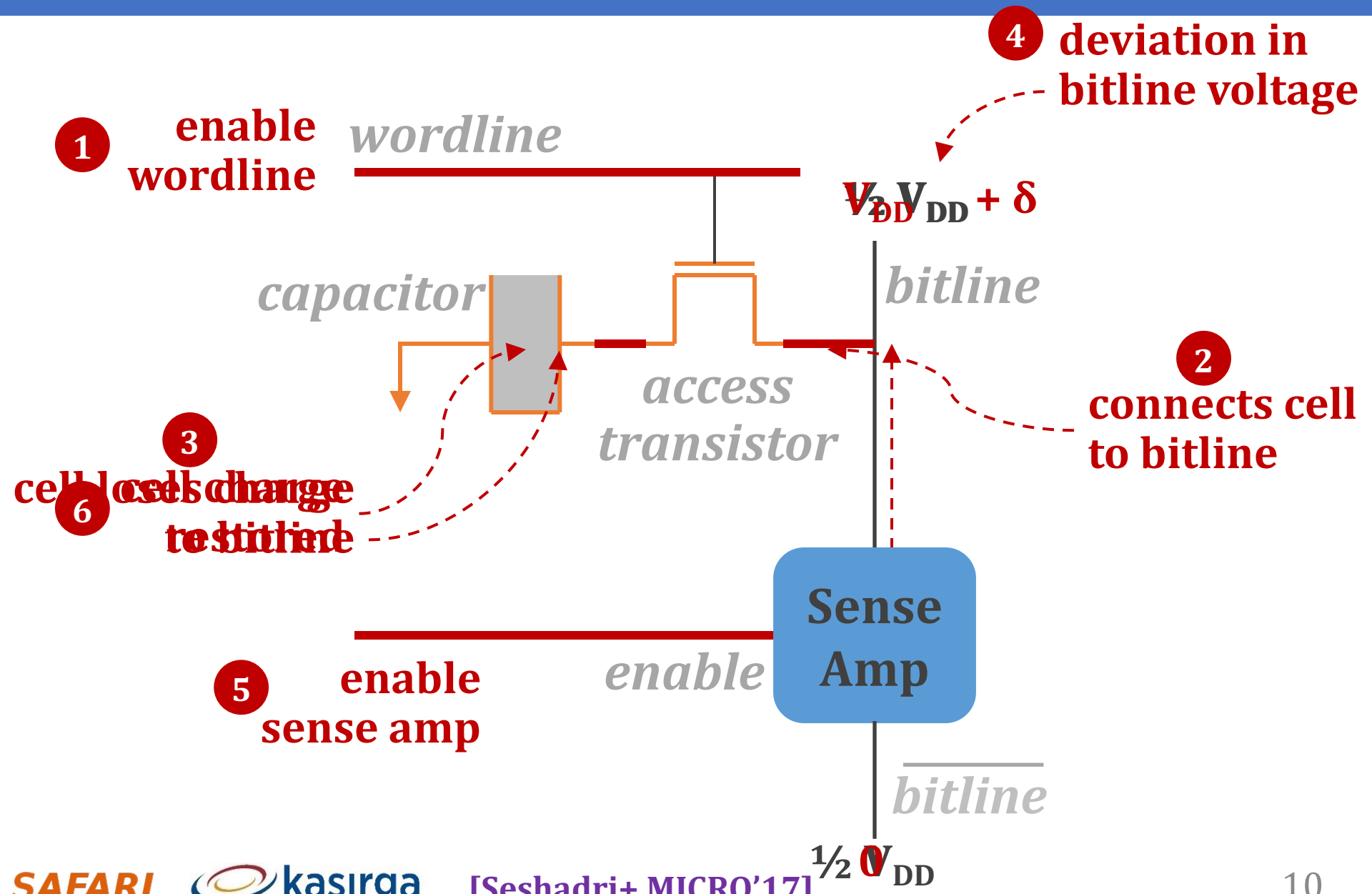
DRAM Organization



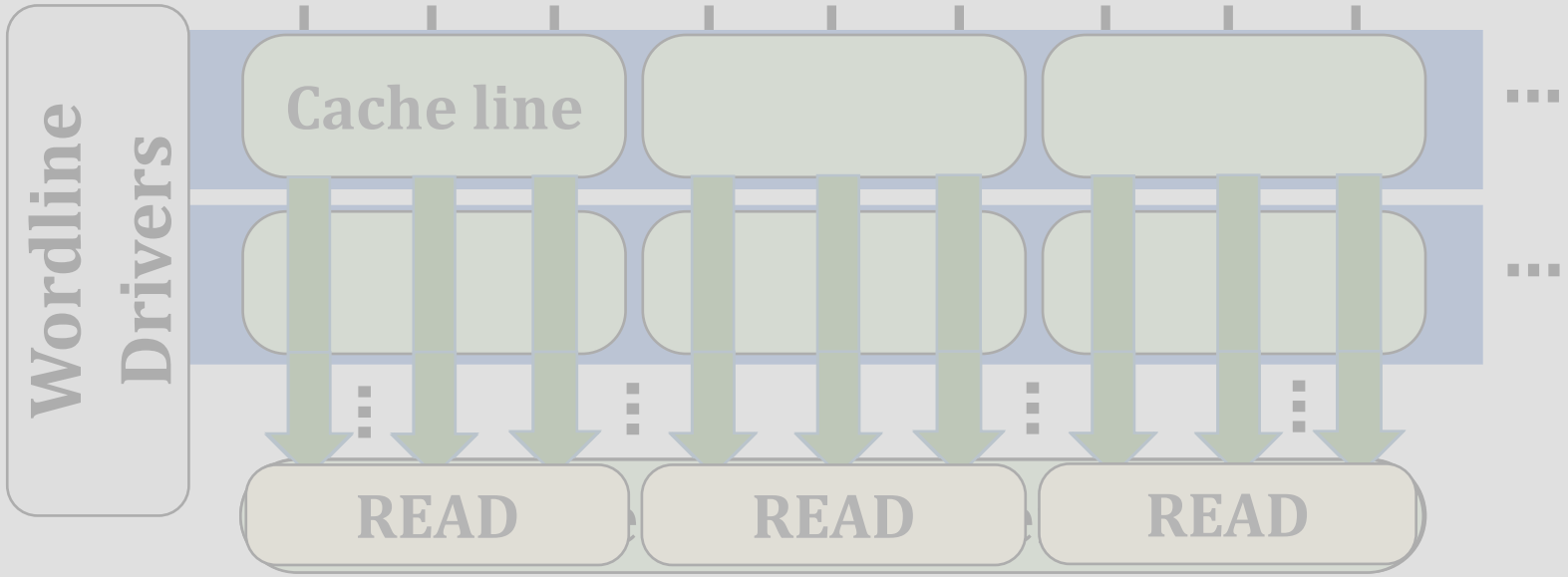
Accessing a DRAM Cell

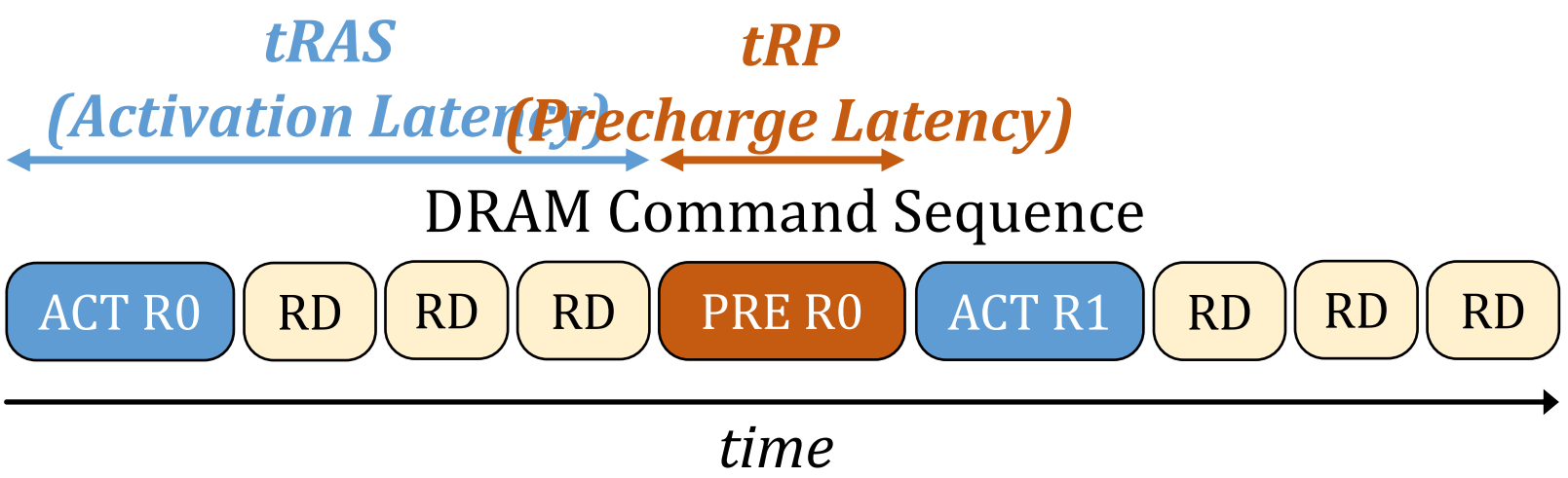


Accessing a DRAM Cell



DRAM Operation





Outline

True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACtivation (QUAC)

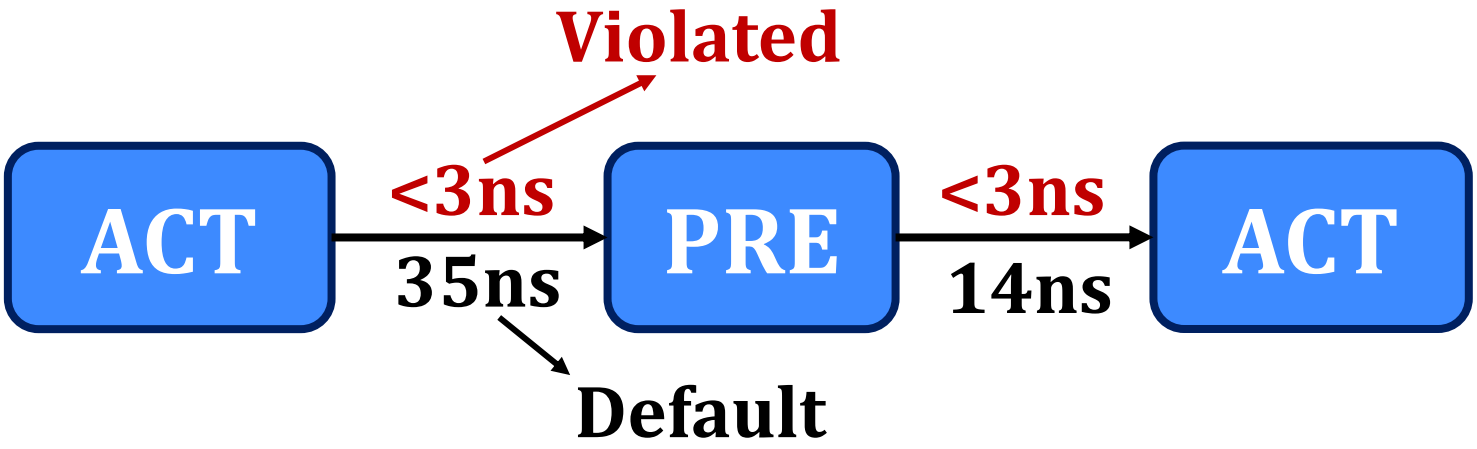
QUAC-TRNG

Evaluation

Quadruple Activation (QUAC)

New Observation

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

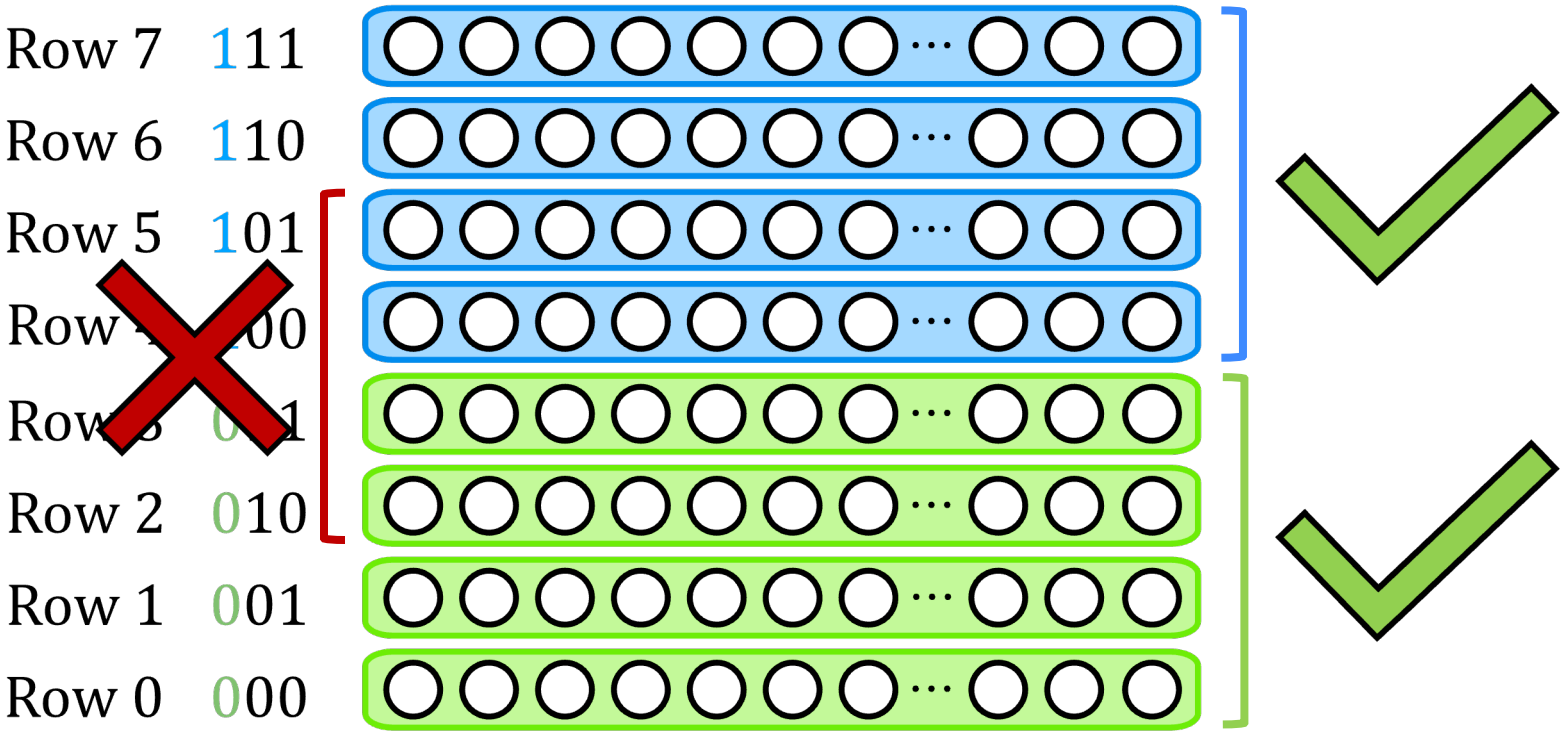


Activate four rows with two ACT commands

Quadruple Activation (QUAC)

Characteristic 1

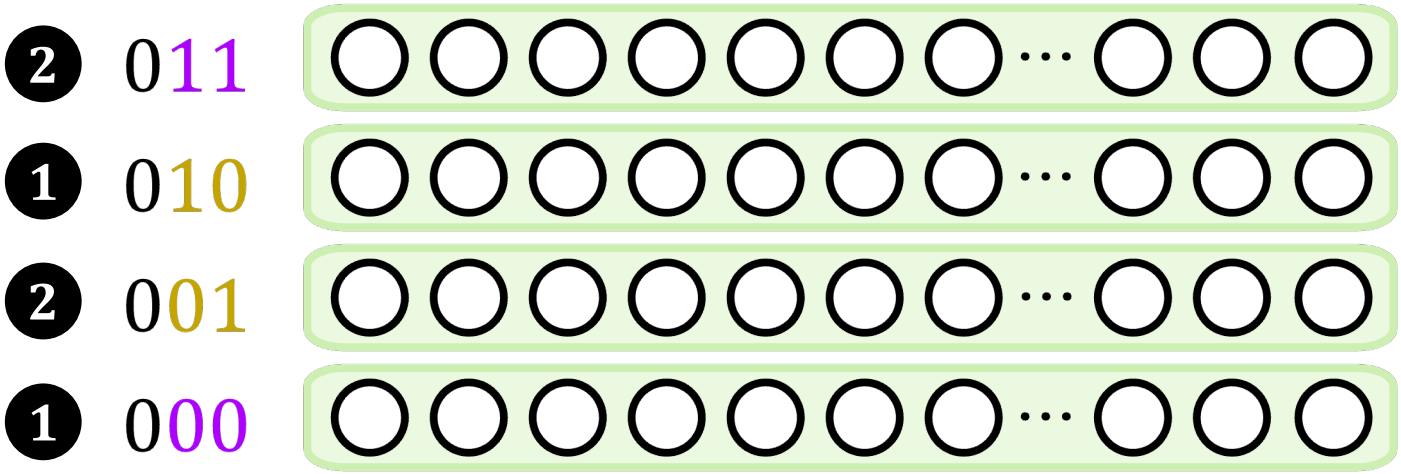
Activates a set of four DRAM rows whose addresses differ **only** in their two LSBs

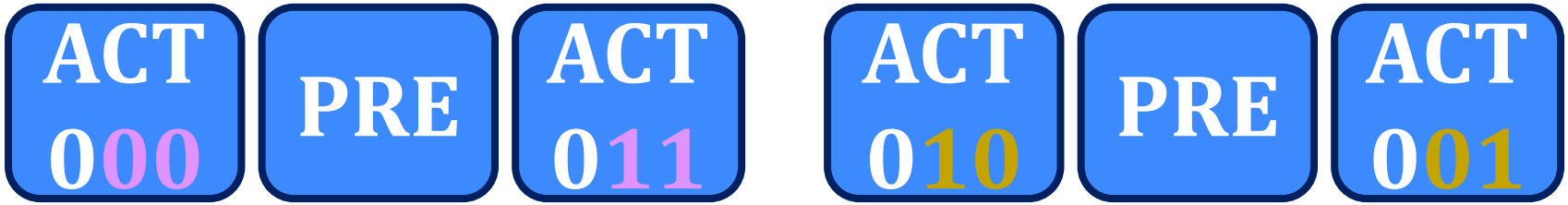


Quadruple Activation (QUAC)

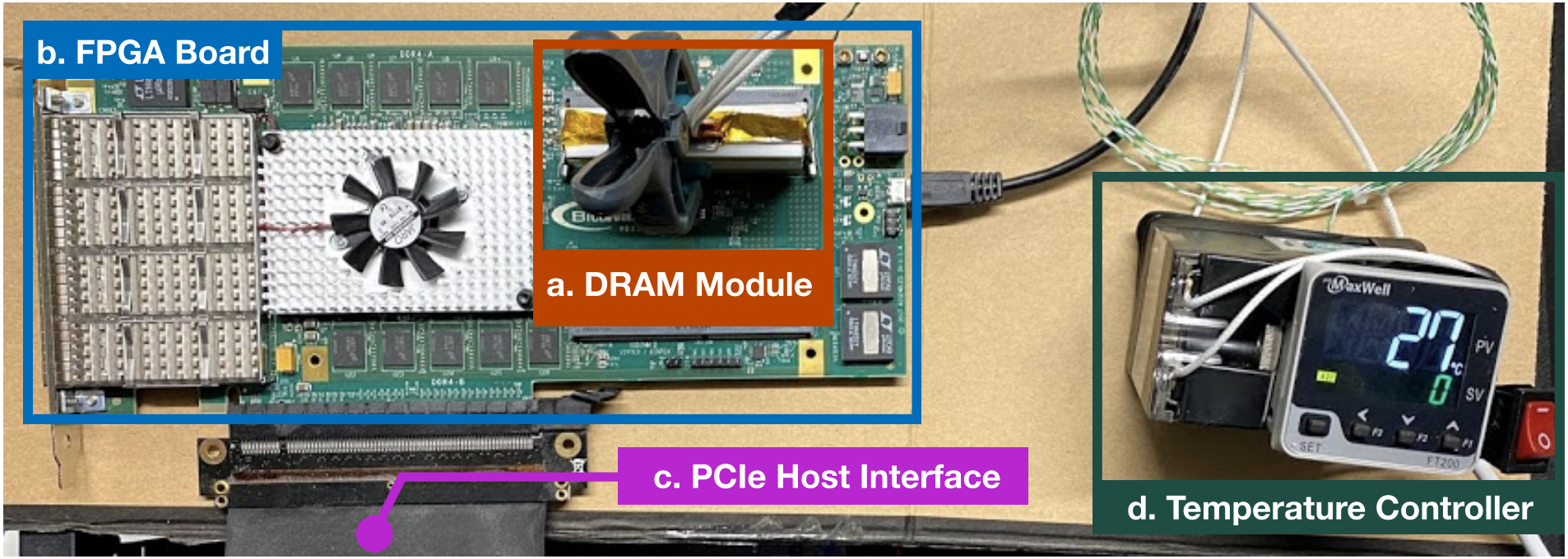
Characteristic 2

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





QUAC on Real DRAM Chips



Valid QUAC behavior on 136 DDR4 chips

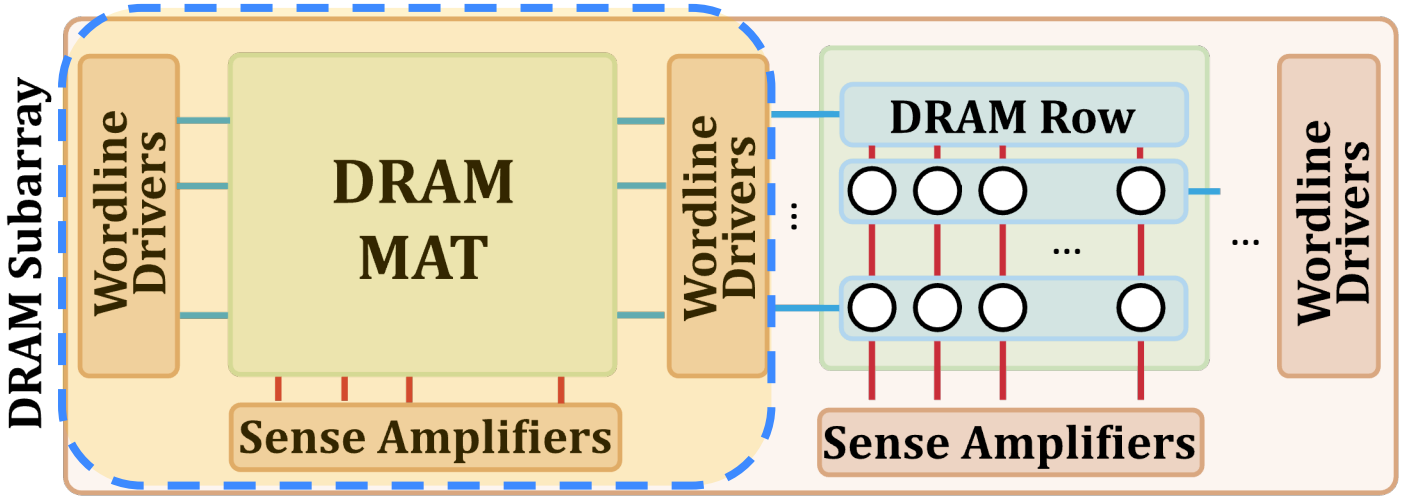
Why Does QUAC Work?

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

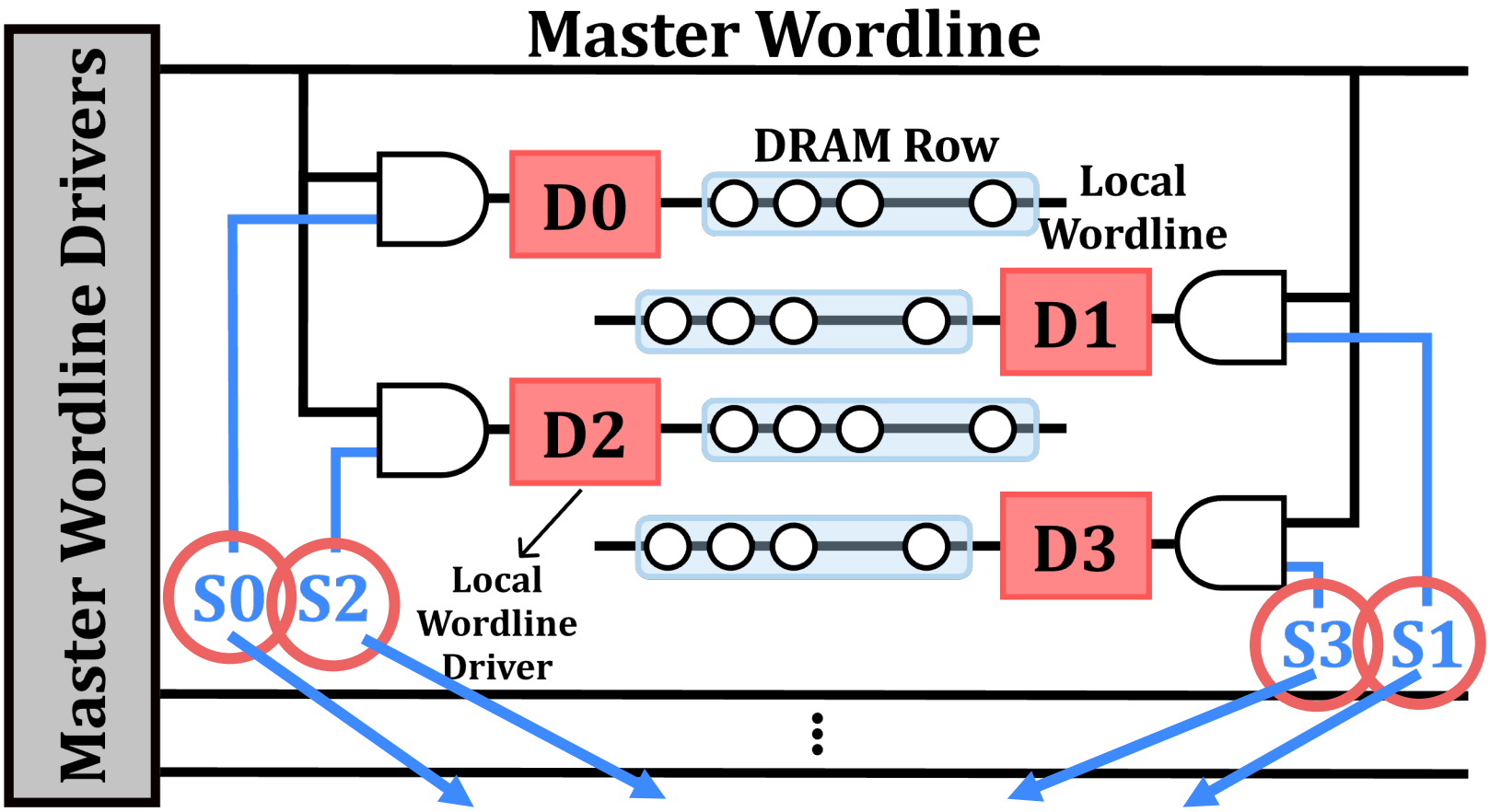
Hierarchical Wordlines

A master wordline drives multiple local wordlines

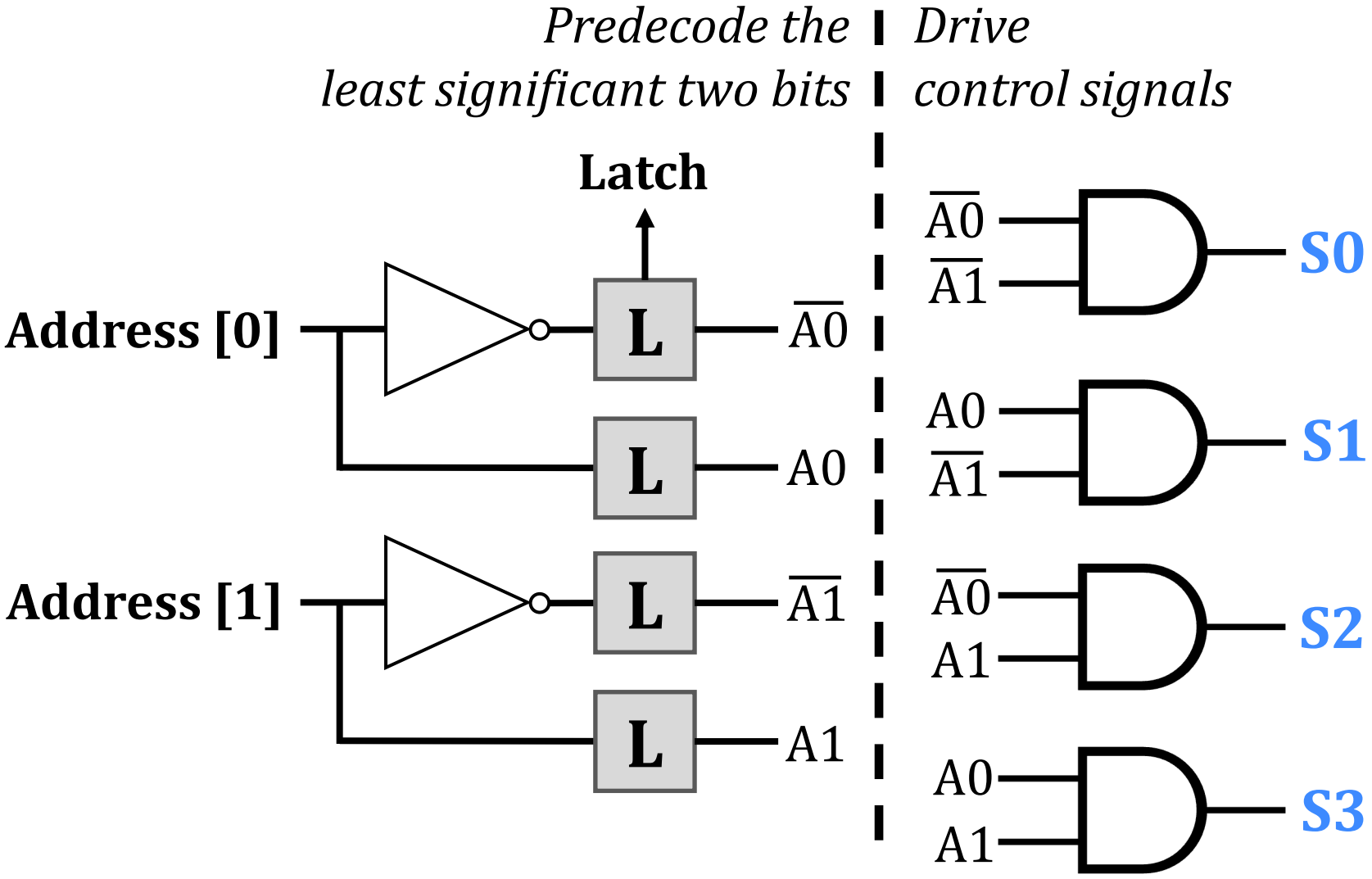


Hierarchical Wordlines

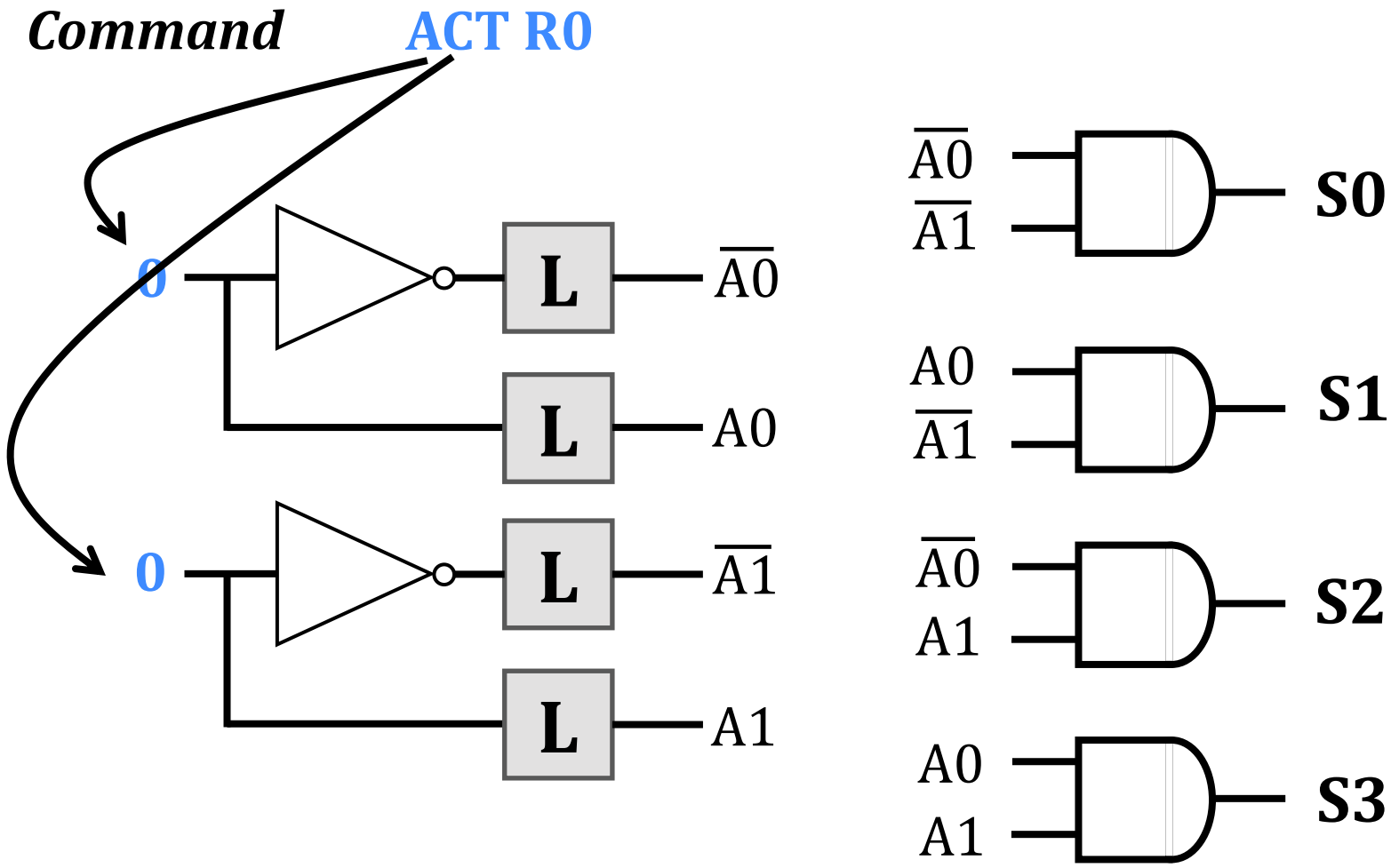
A master wordline drives multiple local wordlines



Hypothetical Row Decoder



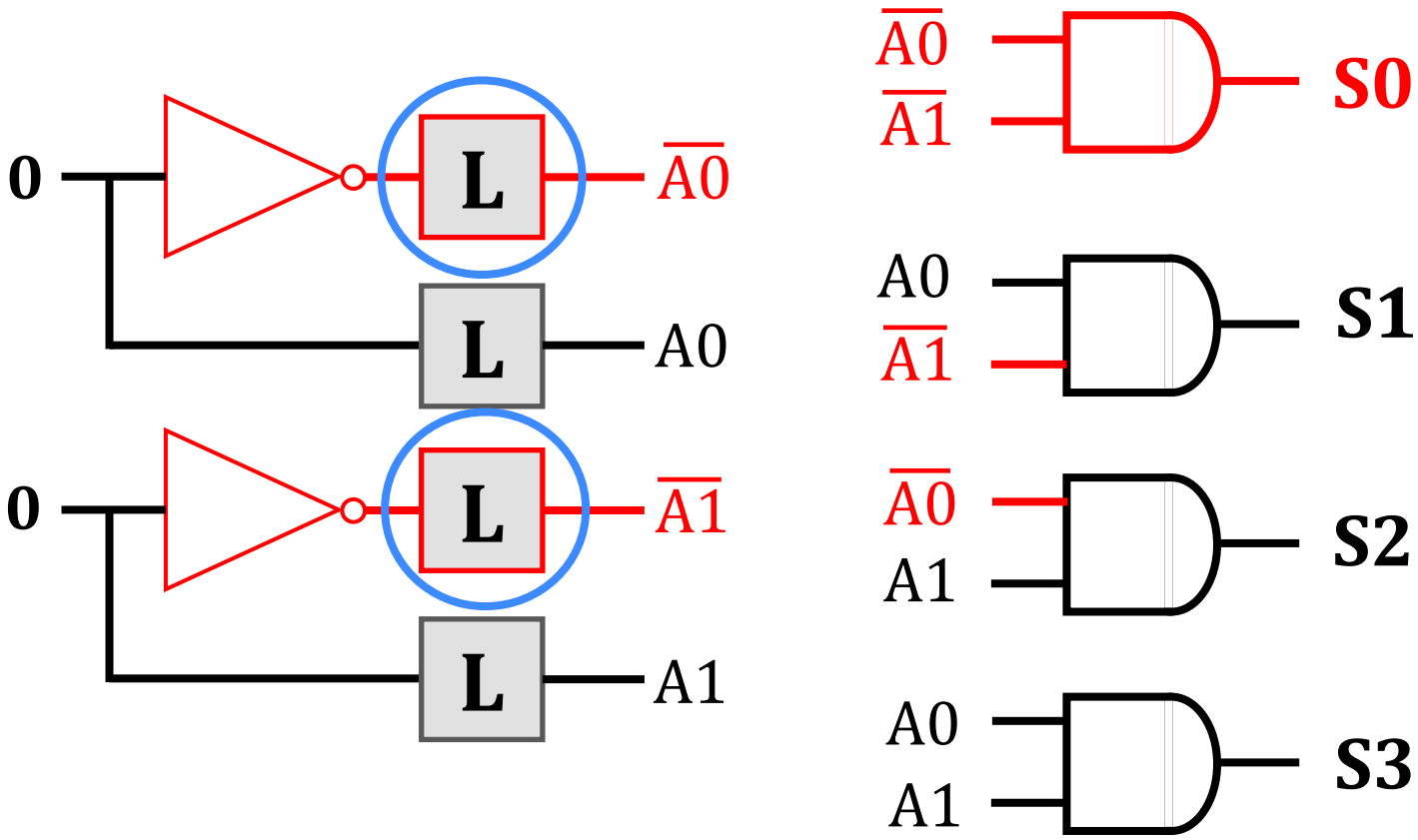
Hypothetical Row Decoder



First ACT command drives a single wordline

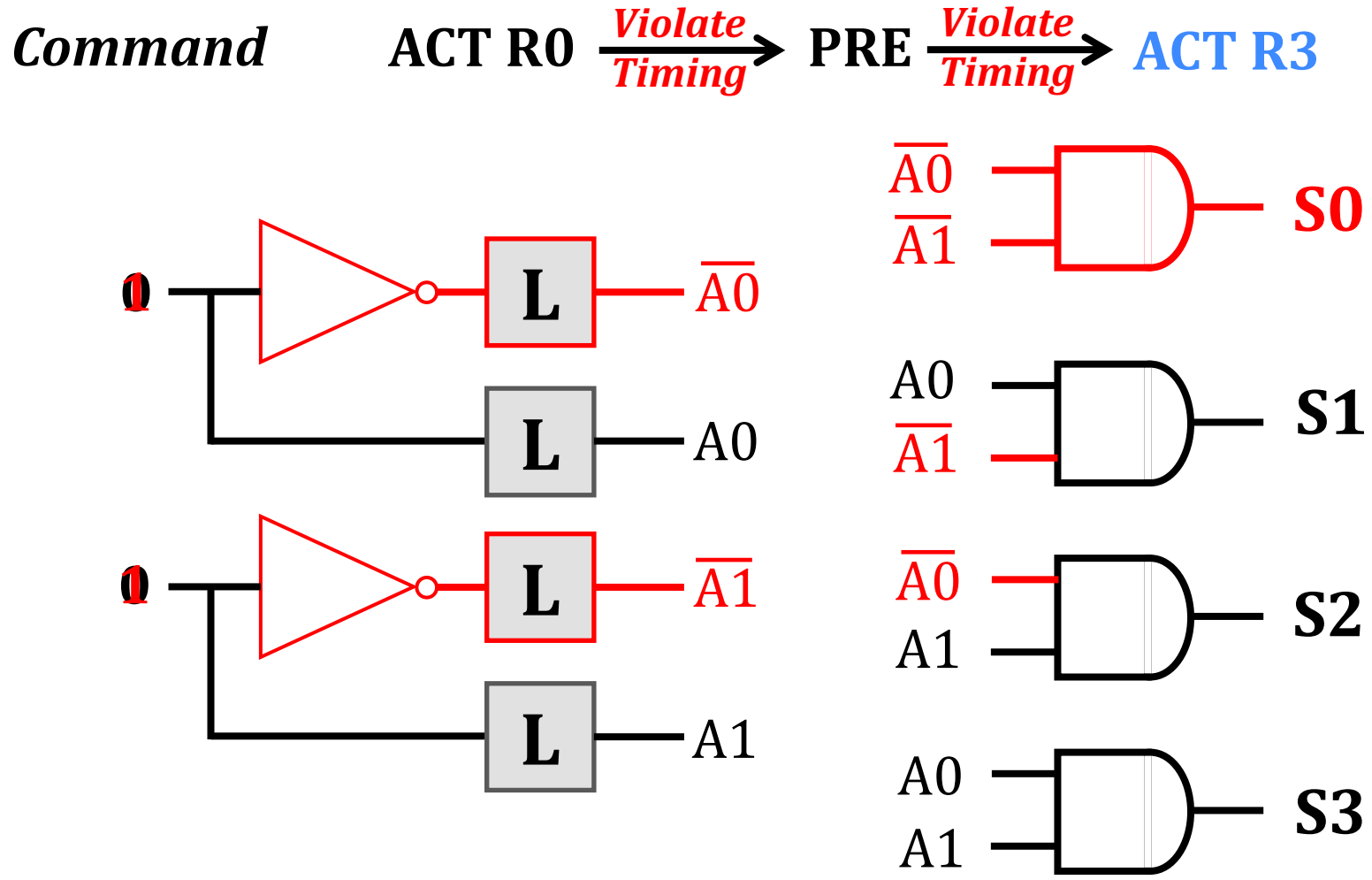
Hypothetical Row Decoder

Command **ACT R0** —*Violate Timing*→ **PRE**



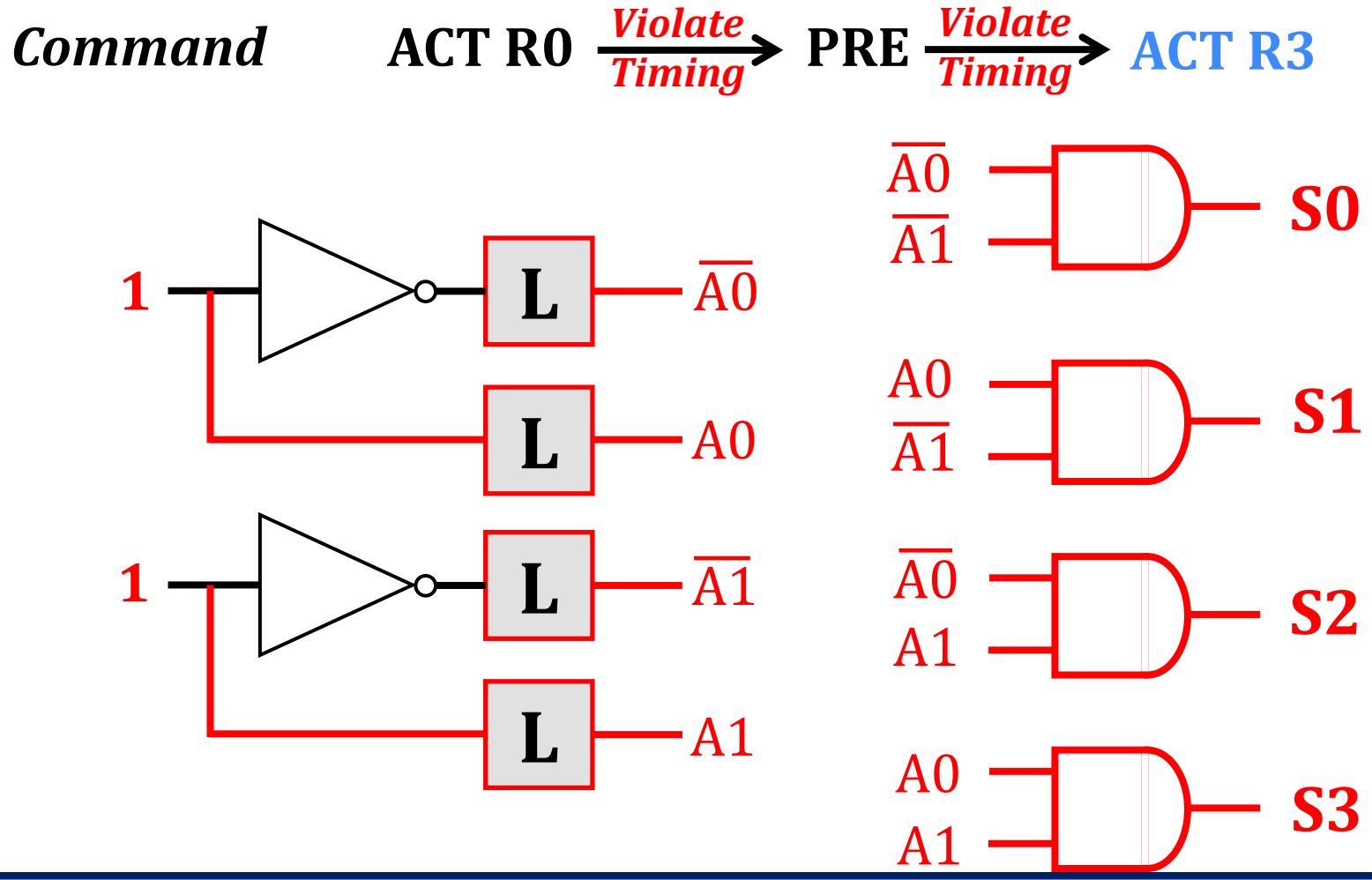
PRE command **cannot** disable latches

Hypothetical Row Decoder



Second ACT drives the **remaining three** wordlines

Hypothetical Row Decoder



All four wordlines are enabled

Quadruple Activation

Outline

True Random Numbers in DRAM

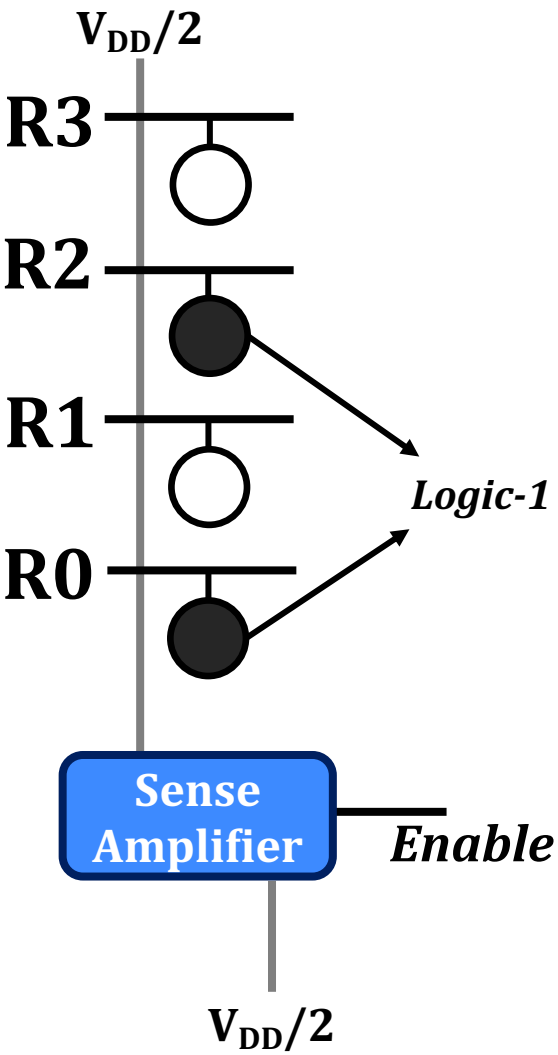
DRAM Organization and Operation

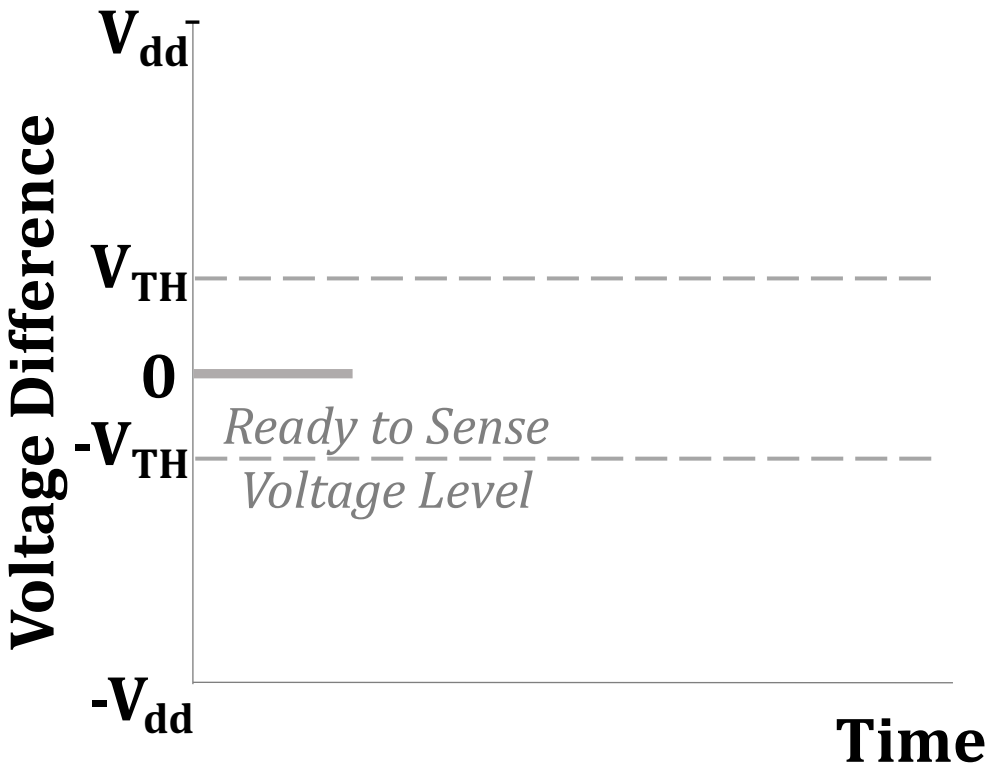
QUadruple ACtivation (QUAC)

QUAC-TRNG

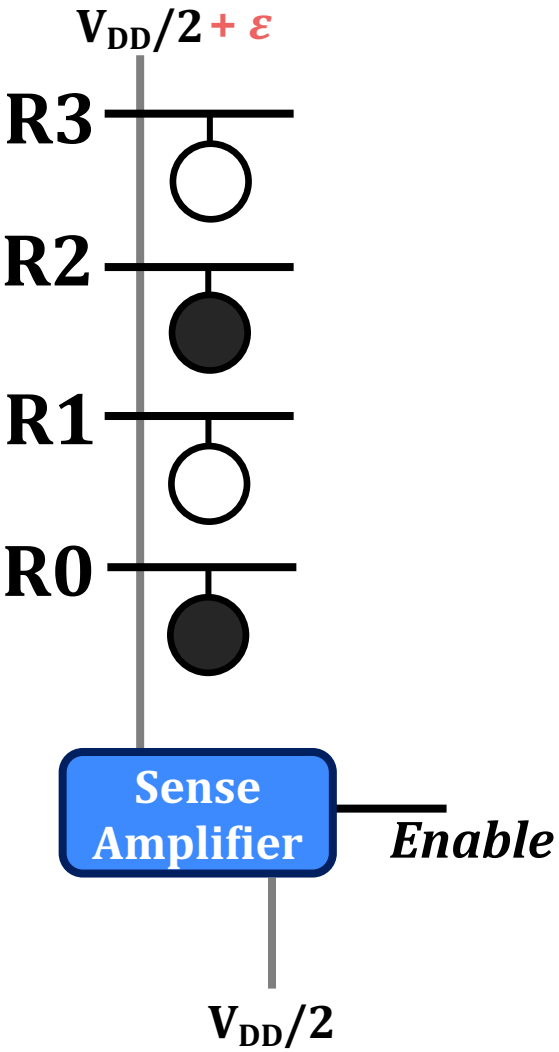
Evaluation

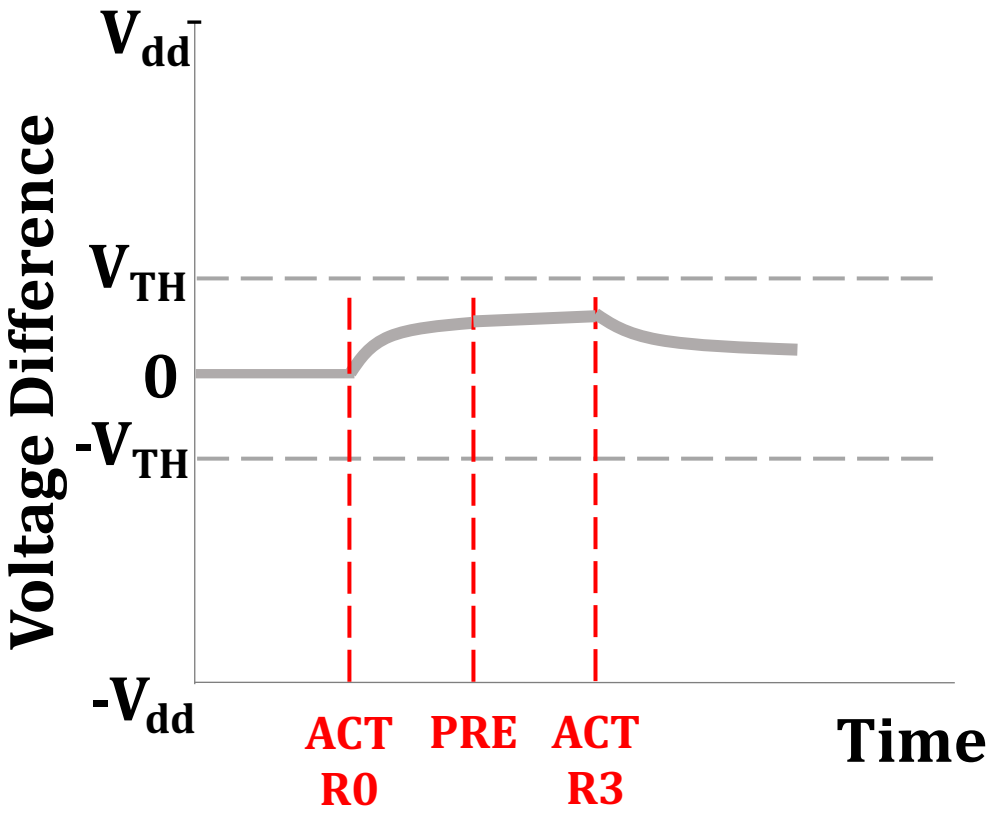
Generating Random Values via QUAC



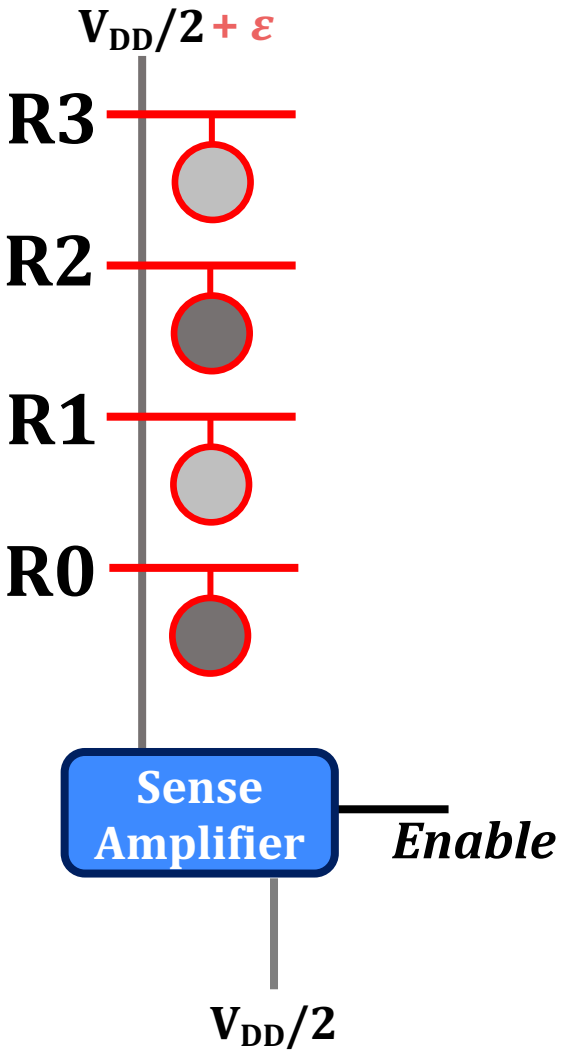


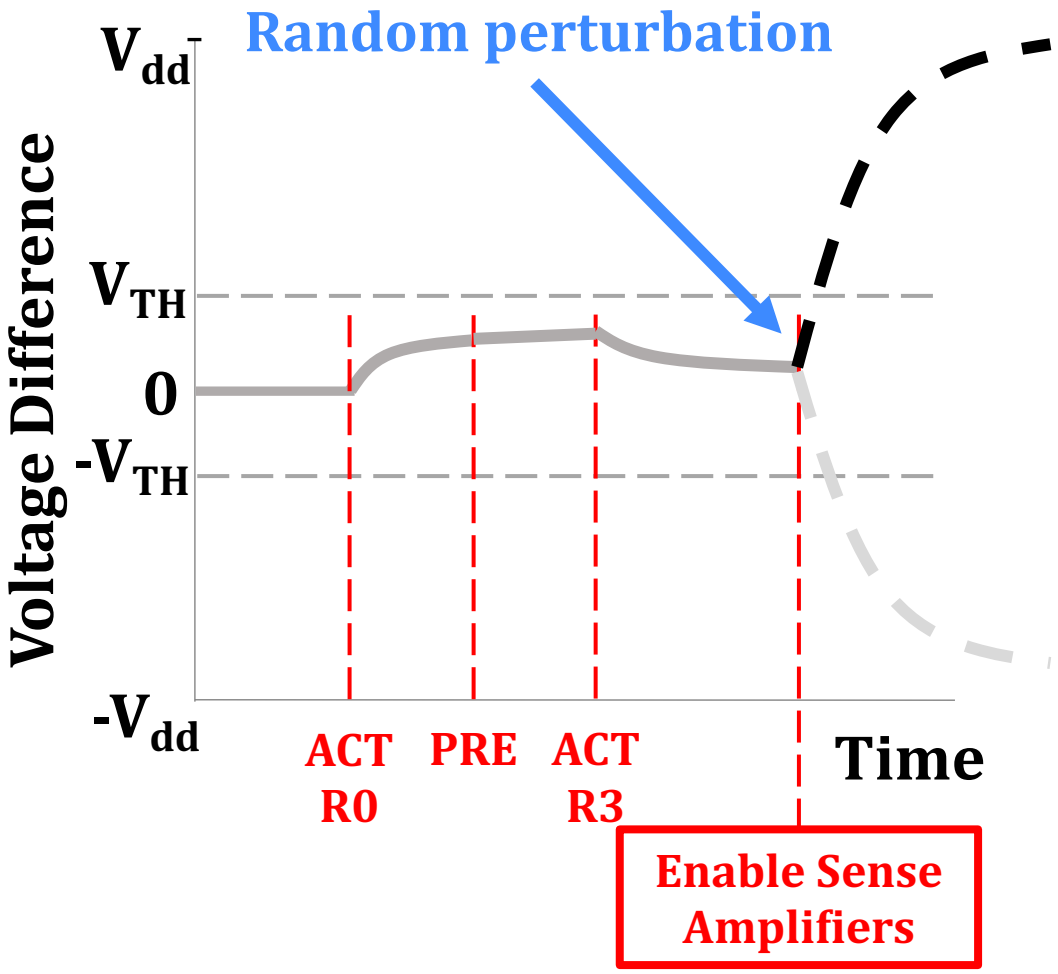
Generating Random Values via QUAC





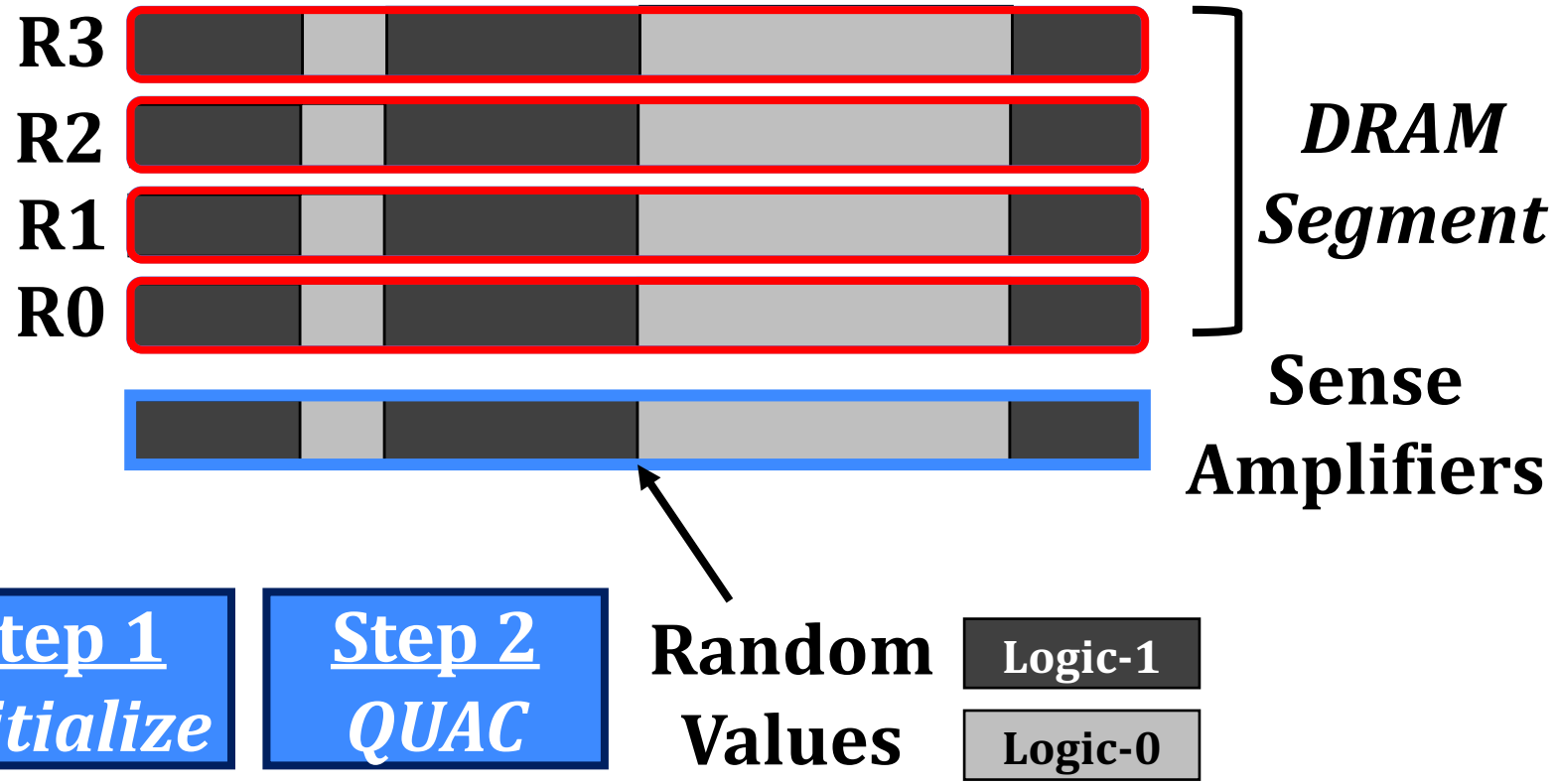
Generating Random Values via QUAC





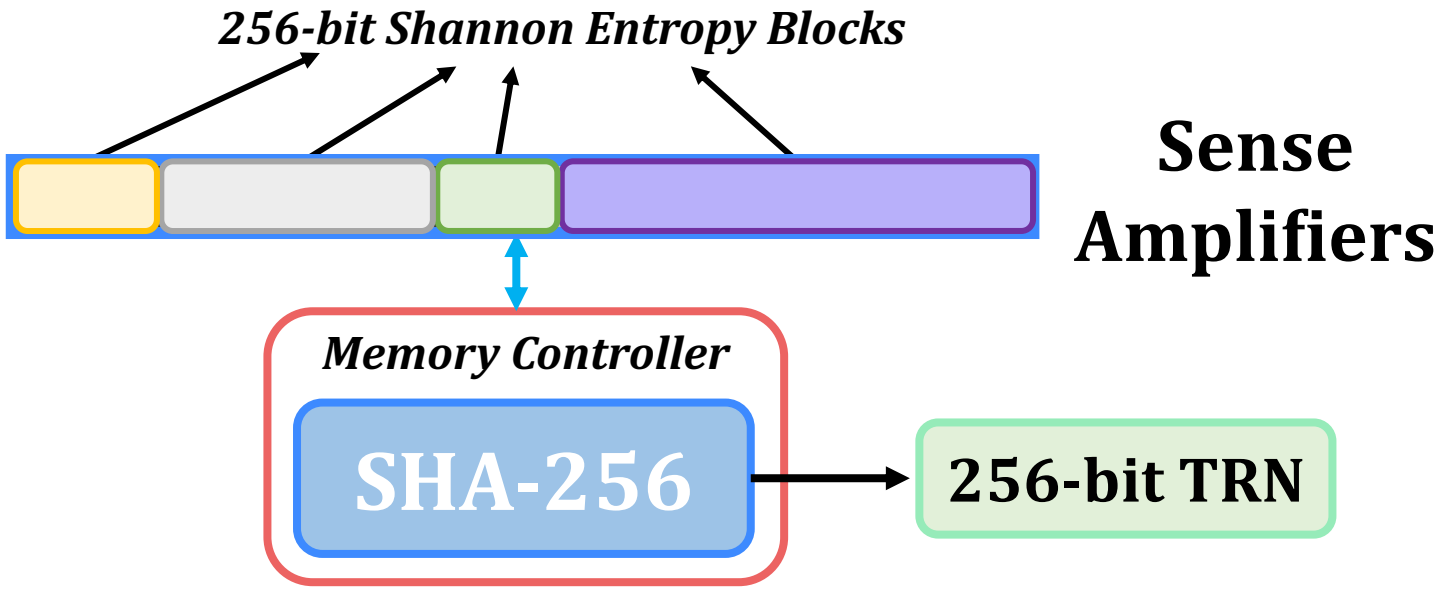
QUAC-TRNG

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



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

Outline

True Random Numbers in DRAM

DRAM Organization and Operation

QUadruple ACtivation (QUAC)

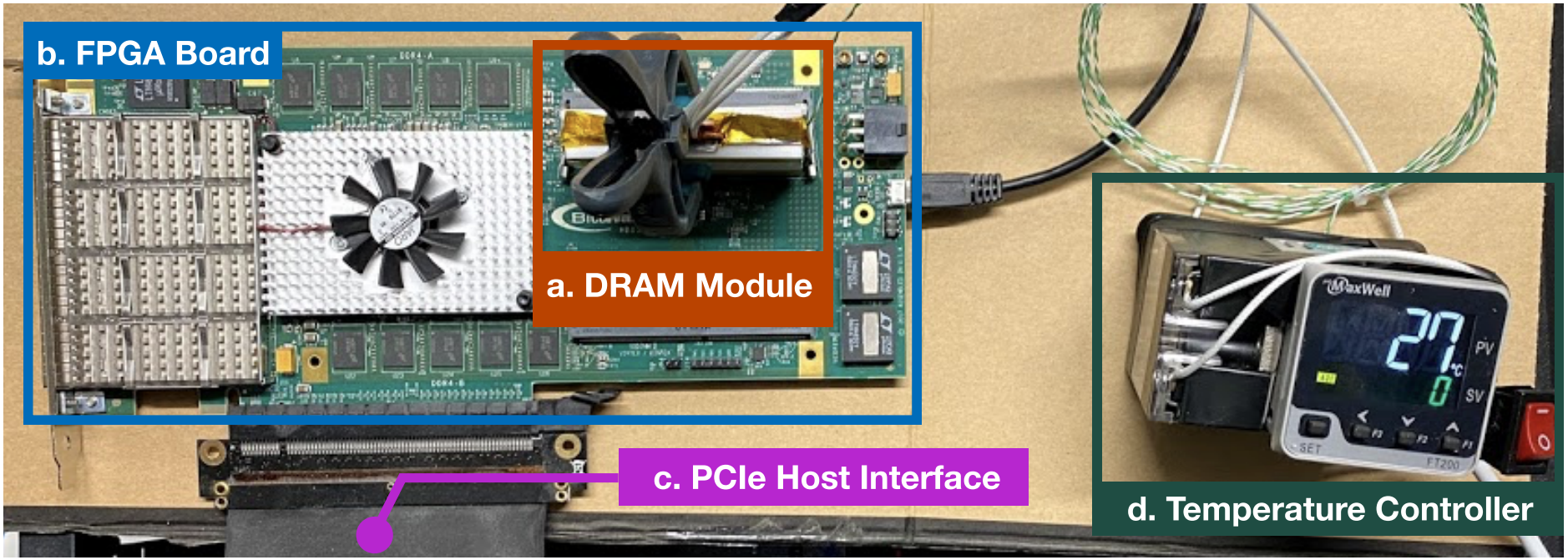
QUAC-TRNG

Evaluation

Real Chip Characterization

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

DDR4 DRAM Bender → DRAM Testing Infrastructure



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 *logic-1* and *logic-0* values in the random bitstream

Sample each bitline following QUAC *1000 times* and calculate the bitline's Shannon Entropy

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

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

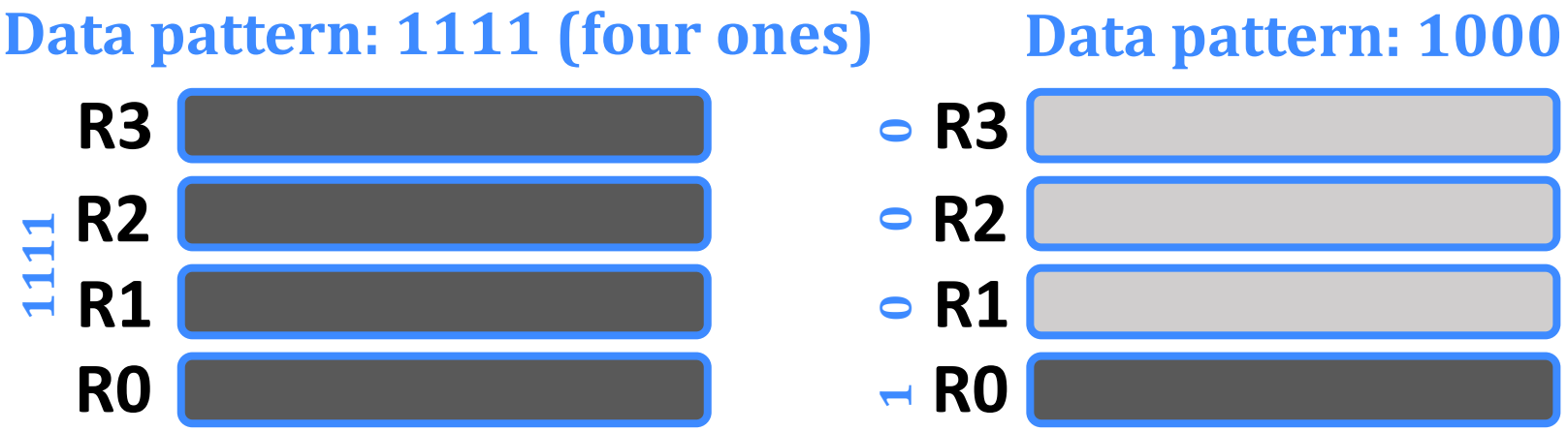
Real Chip Characterization

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 Dependence

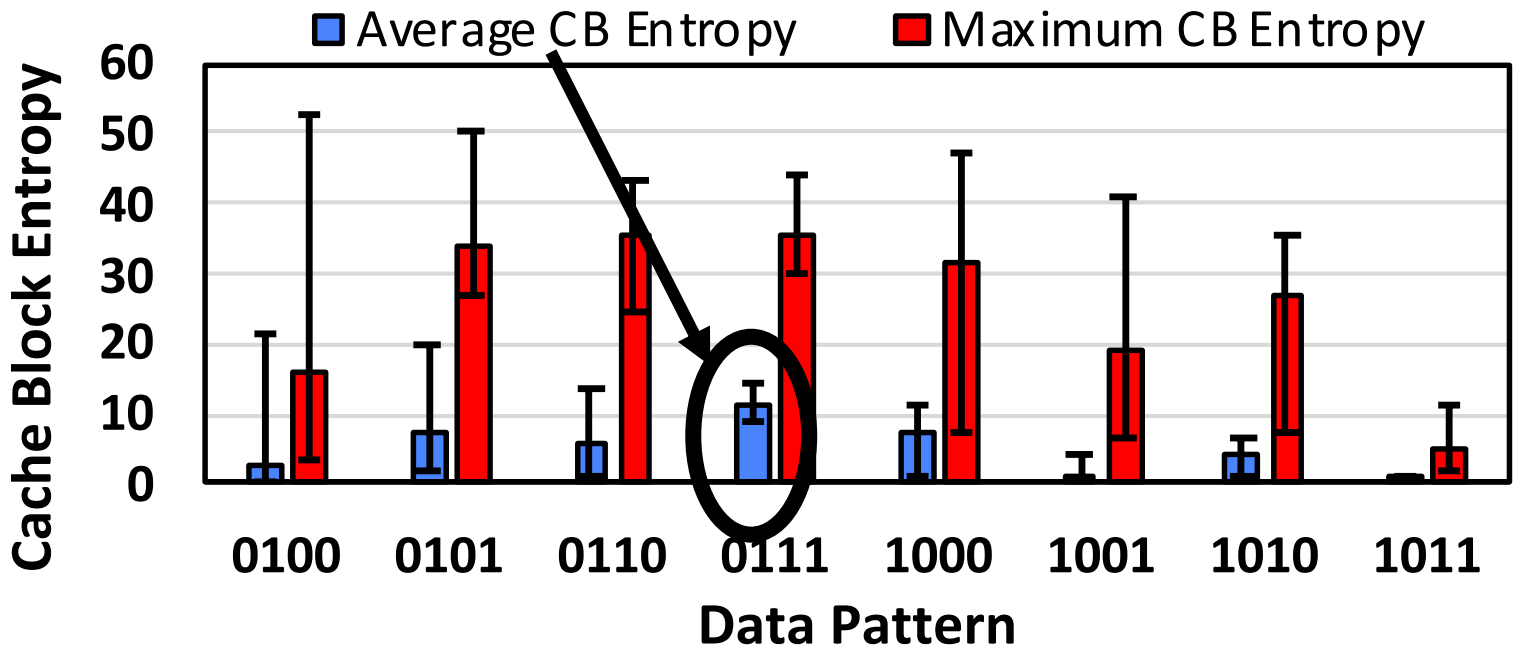
Calculate *cache block entropy (CBE)*

$$\sum \textit{all bitline entropies in the cache block}$$

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

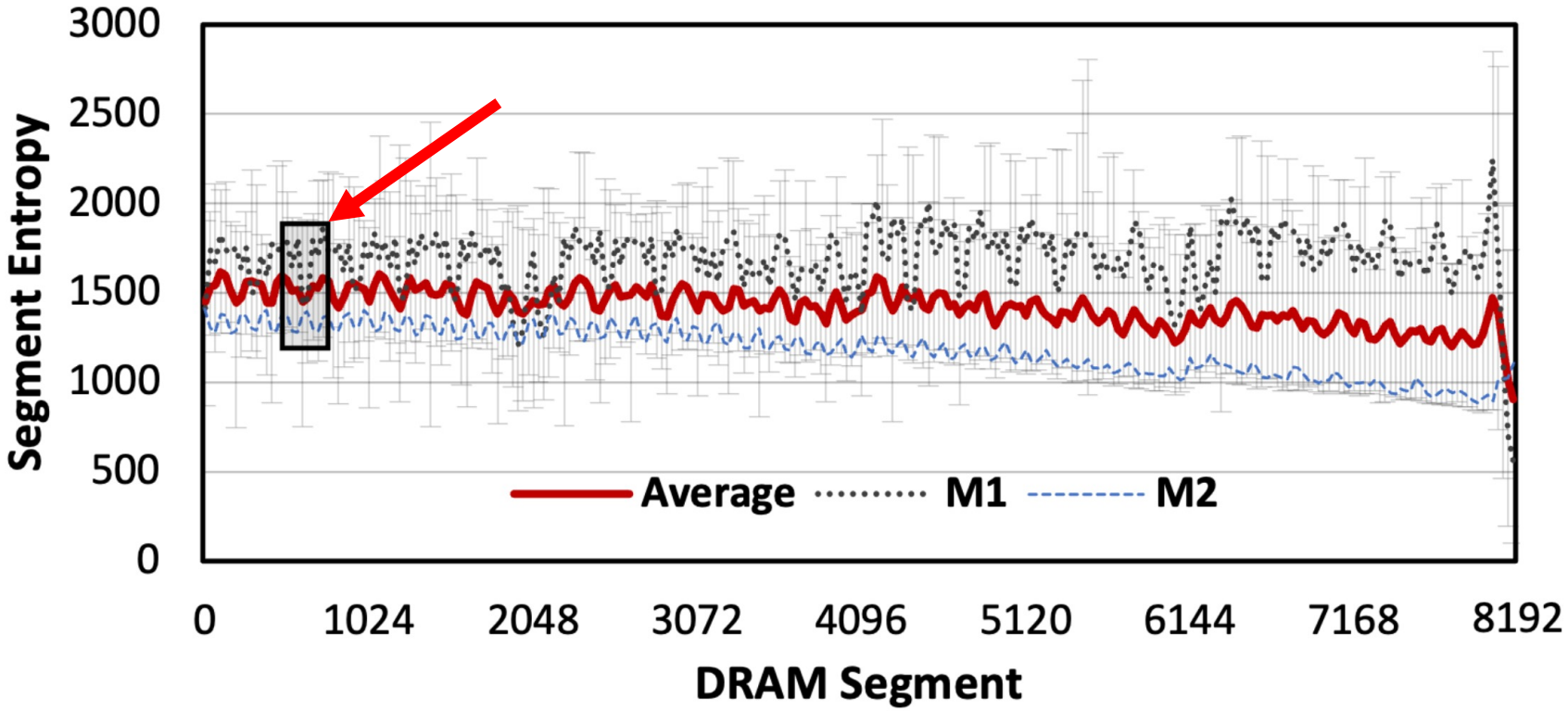


Entropy varies with data pattern

Highest average entropy with pattern "0111"

Spatial Distribution

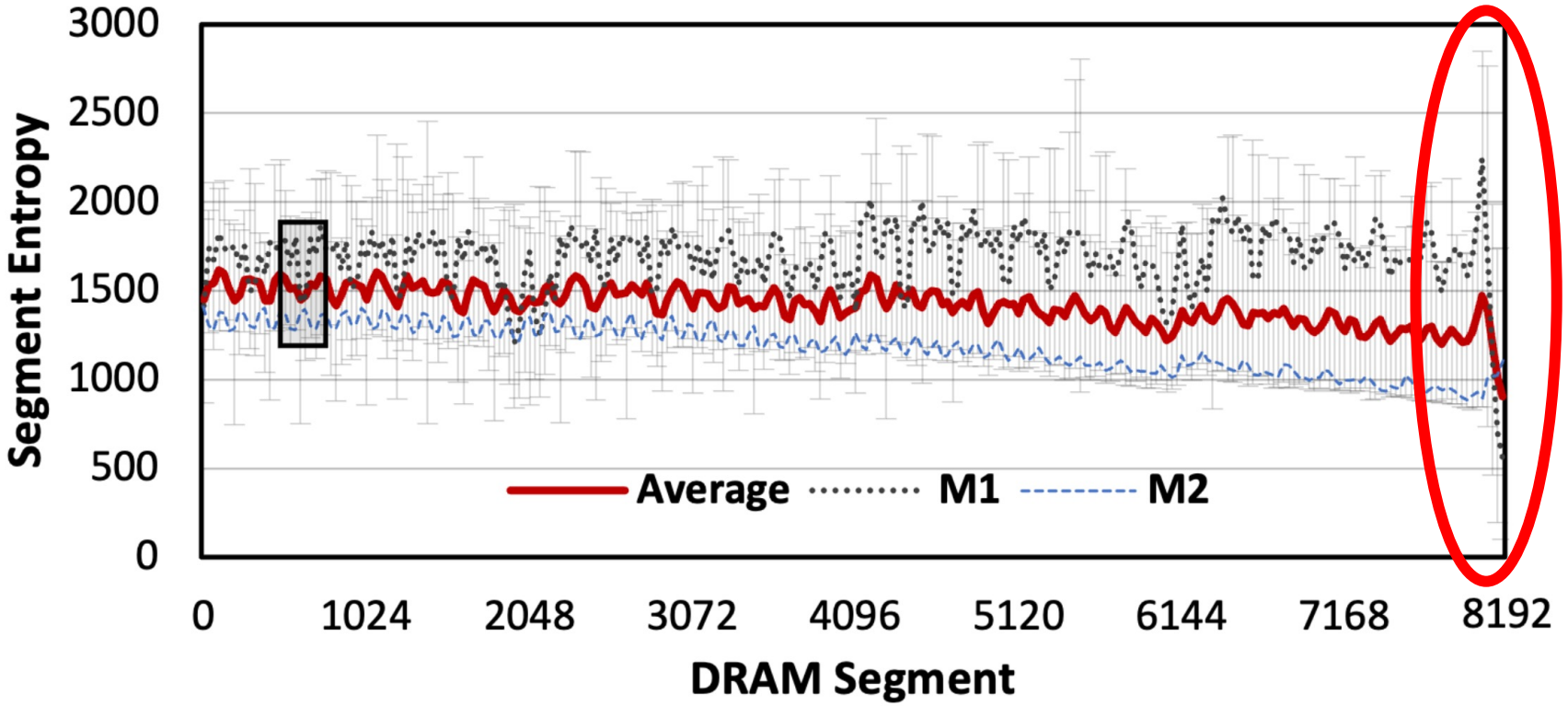
$$\text{Segment entropy} = \sum \textit{all bitline entropies in the segment}$$



Segment entropy behavior is different for different modules

Spatial Distribution

$$\text{Segment entropy} = \sum \textit{all bitline entropies in the segment}$$



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

QUAC-TRNG's Quality

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

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 Results

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
random_excursion	0.164	0.466
random_excursion_variant	0.116	0.510

*VNC: Von Neumann Corrector

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

QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG's throughput according to:

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

SIB: # of **S**HA **I**nput **B**locks in the highest-entropy segment

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

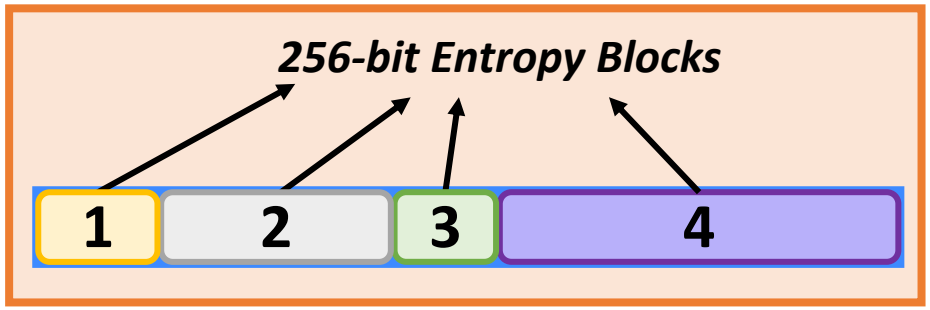
QUAC-TRNG Throughput Estimation

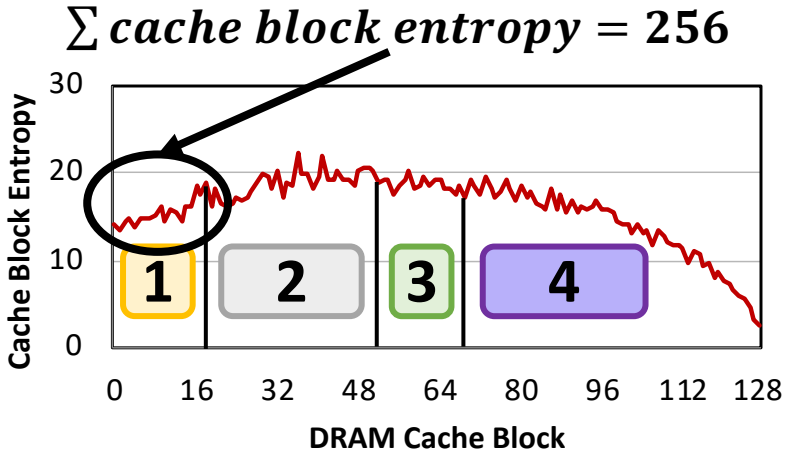
Estimate QUAC-TRNG's throughput according to:

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

SIB: # of **S**HA **I**nput **B**locks in the highest-entropy segment

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





QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG's throughput according to:

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

SIB: # of **S**HA **I**nput **B**locks in the highest-entropy segment

L: Latency of one QUAC operation in nanoseconds

QUAC-TRNG Throughput Estimation

Estimate QUAC-TRNG's throughput according to:

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

SIB: # of **S**HA **I**nput **B**locks in the highest-entropy segment

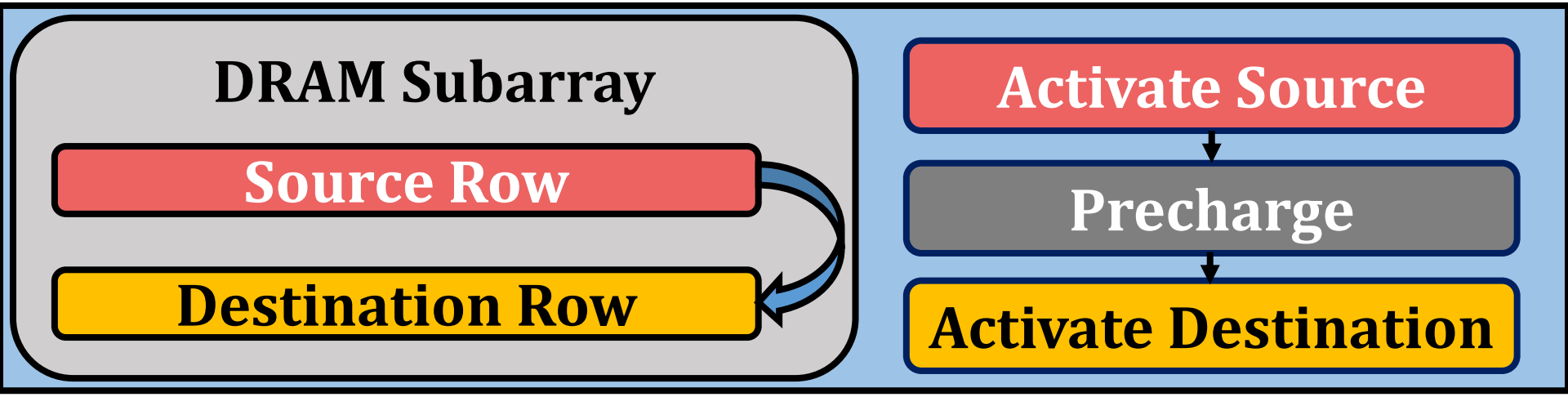
L: **L**atency of one QUAC operation in nanoseconds

QUAC-TRNG Configurations

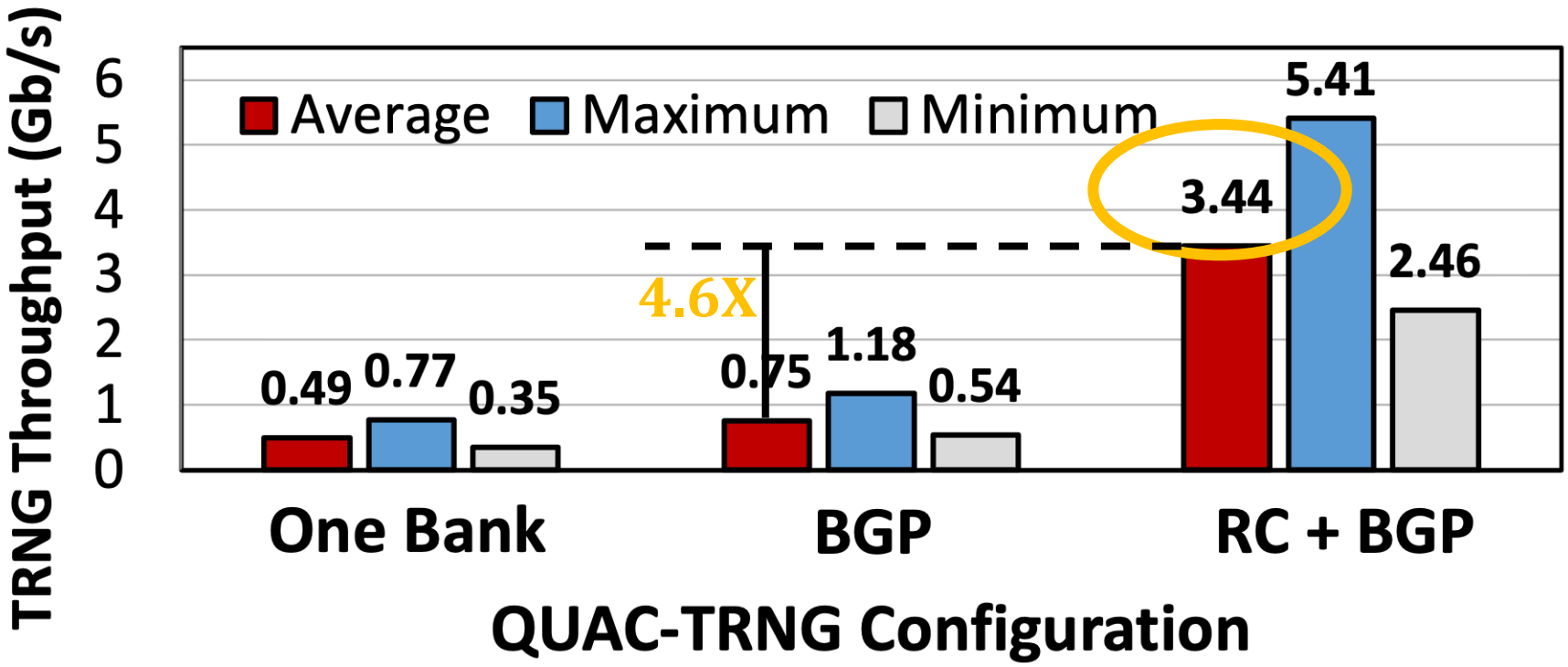
1. ***One Bank*** — Use a single DRAM bank

2. ***BGP*** — **B**ank **G**roup-Level **P**arallelism
Use four banks from different bank groups

3. ***RC + BGP*** — **R**ow**C**lone **+ BGP**
Use in-DRAM copy to initialize DRAM rows *and* use four banks from different bank groups



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

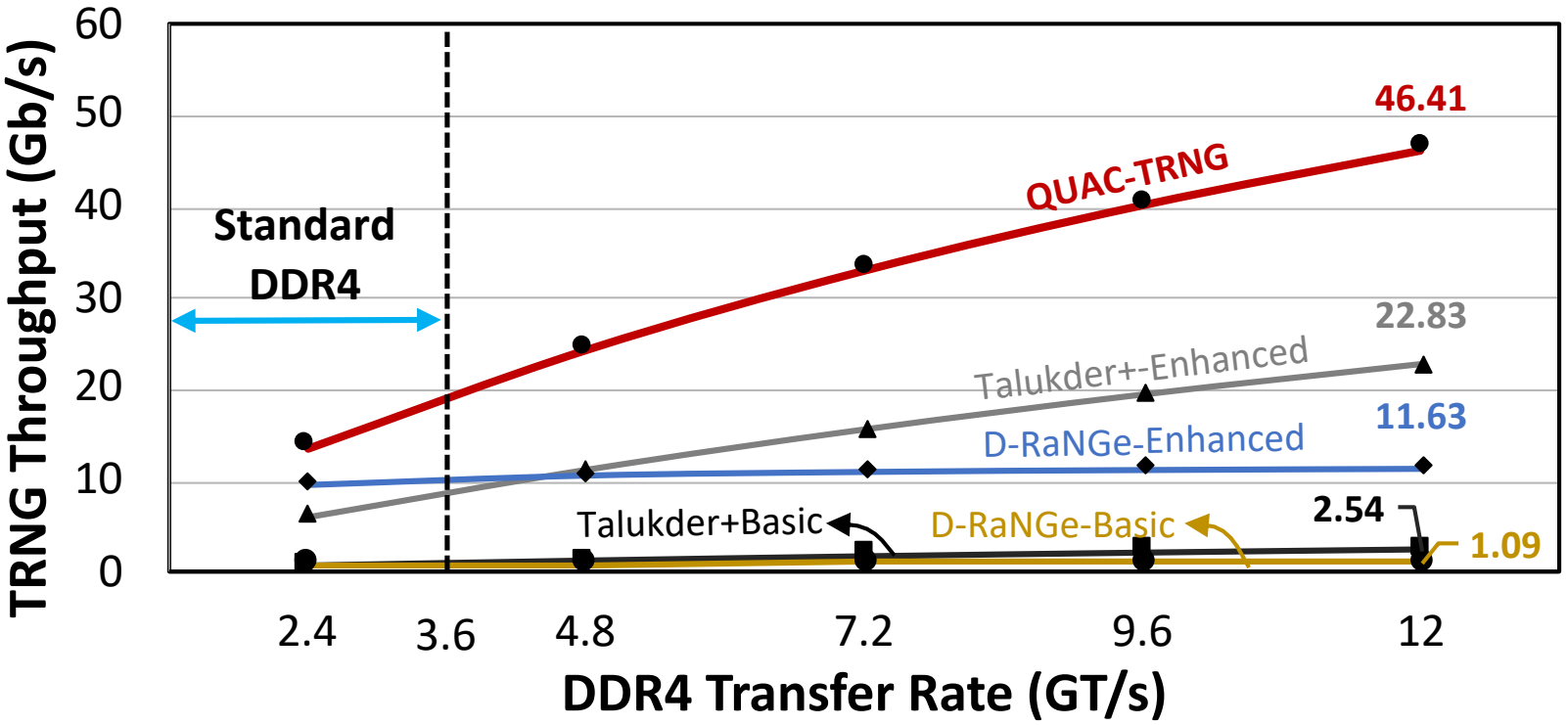
Calculate throughput by tightly scheduling the DDR4 commands required to induce failures

Evaluate two versions of these past two works:

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

Assume four-channel DDR4 memory

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 → QU**adruple **AC**tivation **(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

SAFARI kasırga

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

A Case for Self-Managing DRAM Chips:
Enabling Efficient in-DRAM Maintenance Operations

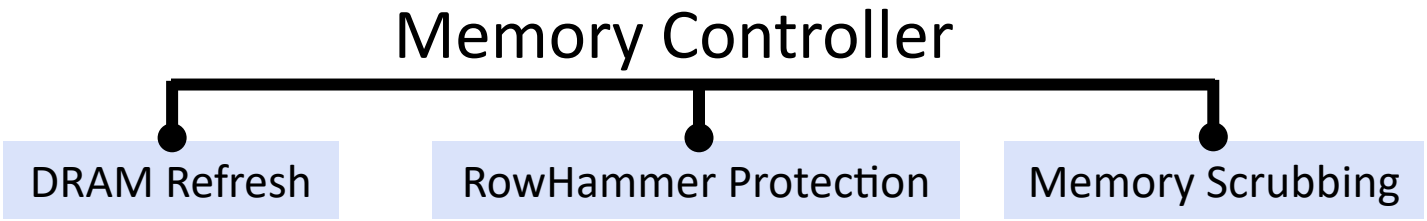
Hasan Hassan, Ataberk Olgun, A. Giray Yaglikci, Haocong Luo, Onur Mutlu

PhD Thesis: “Improving DRAM Performance, Reliability, and Security by Rigorously Understanding Intrinsic DRAM Operation”

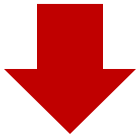
2023 EDAA Outstanding Dissertation Award

Self-Managing DRAM: Problem

The **Memory Controller** manages
DRAM maintenance operations



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

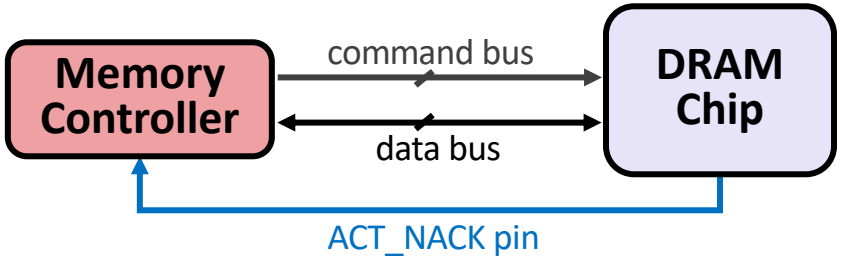
SMD: Overview

Self-Managing DRAM (SMD)

enables autonomous in-DRAM maintenance operations

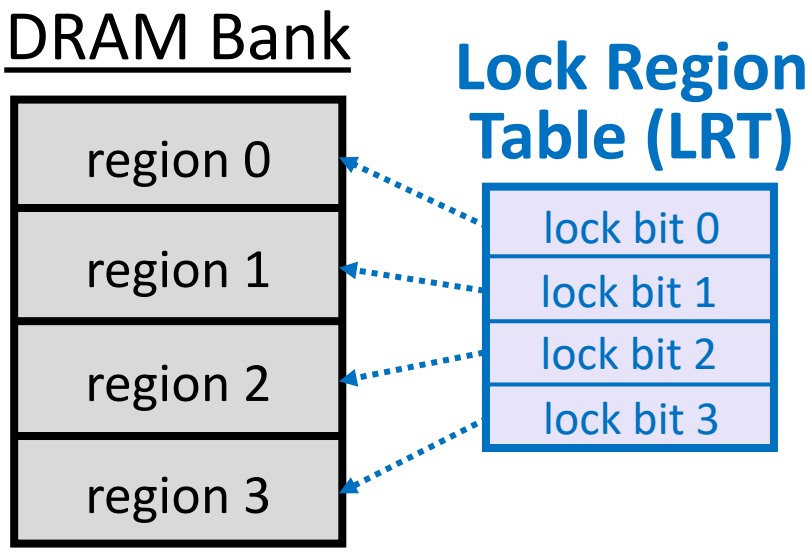
Key Idea:

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



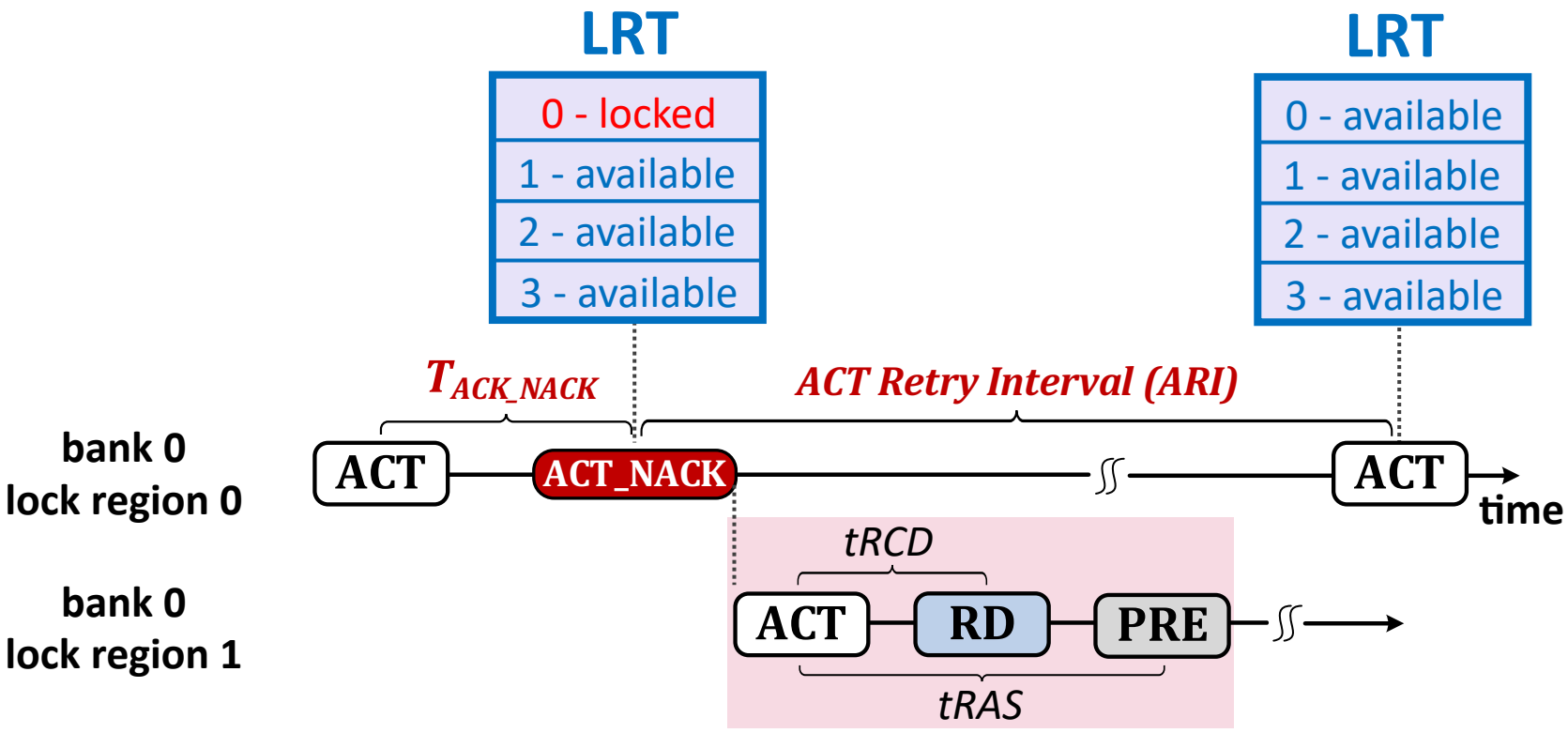
Leveraging the ability to *reject an ACT*, a maintenance operation can be implemented *completely* within a DRAM chip

SMD: DRAM Bank Architecture



- 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



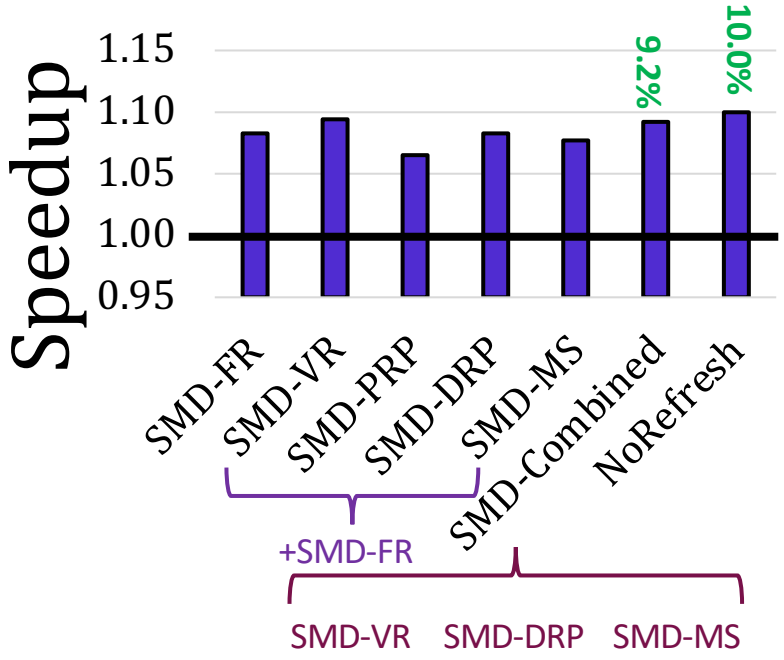
SMD-Based Maintenance Mechanisms

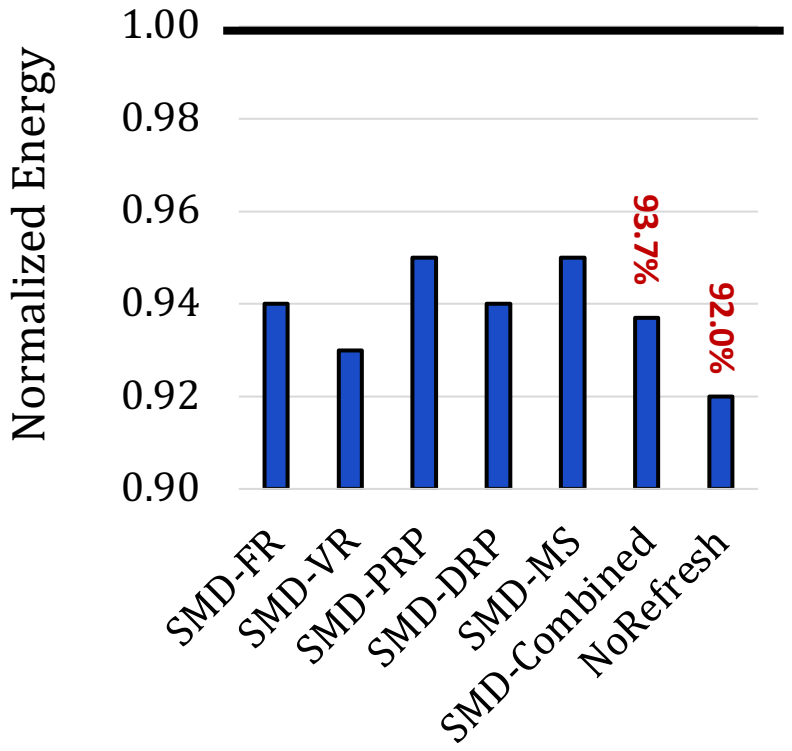
DRAM Refresh	**Fixed Rate (SMD-FR)** *__uniformly__ refreshes __all__ DRAM rows with a __fixed__ refresh period*	**Variable Rate (SMD-VR)** *__skips__ refreshing rows that can __retain their data for longer__ than the default refresh period*
RowHammer Protection	**Probabilistic (SMD-PRP)** *Performs __neighbor row refresh__ with __a small probability__ on every row activation*	**Deterministic (SMD-DRP)** *__keeps track__ of most __frequently activated__ rows and performs __neighbor__ row refresh when activation count threshold is exceeded*
Memory Scrubbing	**Periodic Scrubbing (SMD-MS)** *periodically __scans__ the __entire__ DRAM for errors and corrects them*	

Methodology

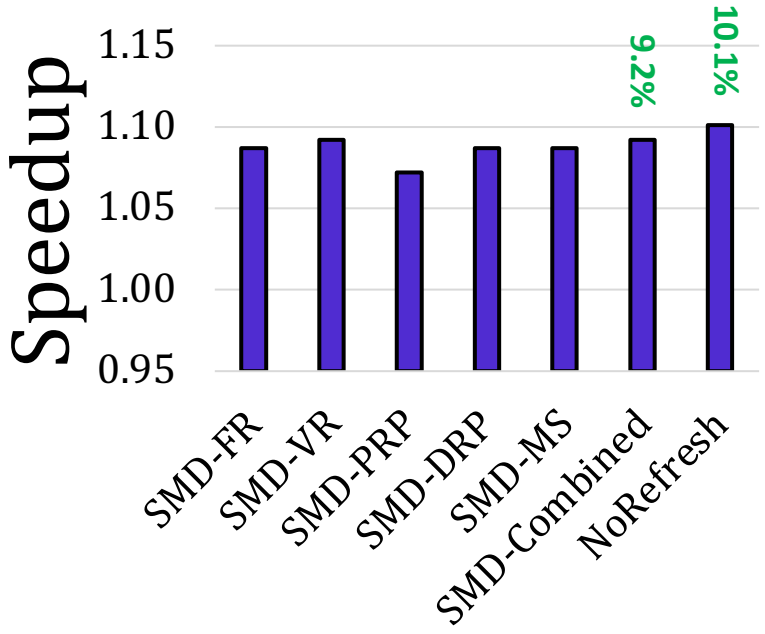
- **Simulators**
  - Ramulator [Kim+, CAL'15]
    https://github/CMU-SAFARI/ramulator
  - DRAMPower [Chandrasekar+, DSD'11]
    https://github.com/tukl-msd/DRAMPower
- **Workloads**
  - 44 single-core workloads
    SPEC CPU2006, TPC, STREAM, MediaBench
  - 60 multi-programmed four-core workloads
    By randomly choosing from single-core workloads
- **System Parameters**
  - 4-channel dual-rank DDR4 DRAM
  - 32ms default refresh period

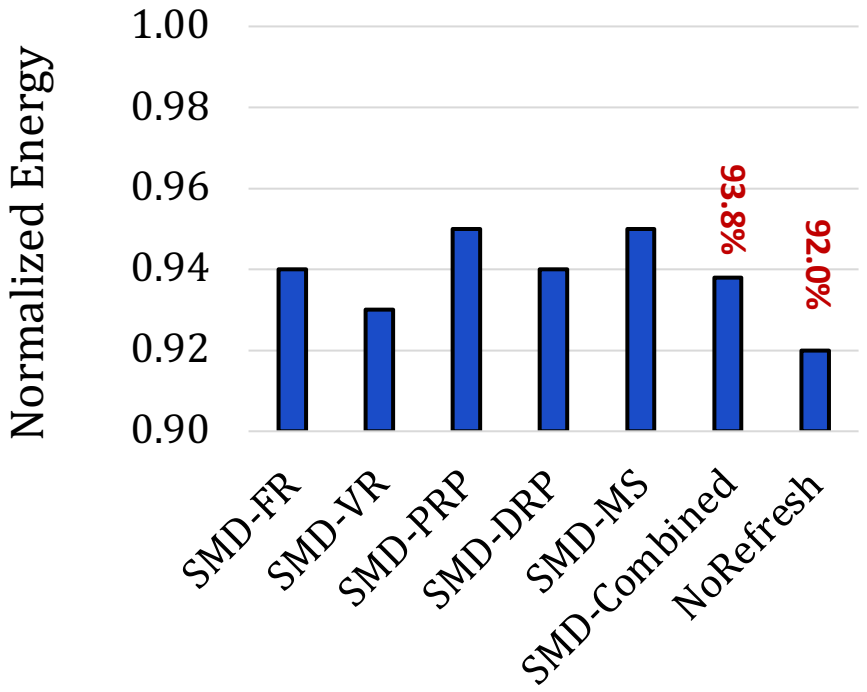
Single-Core Results





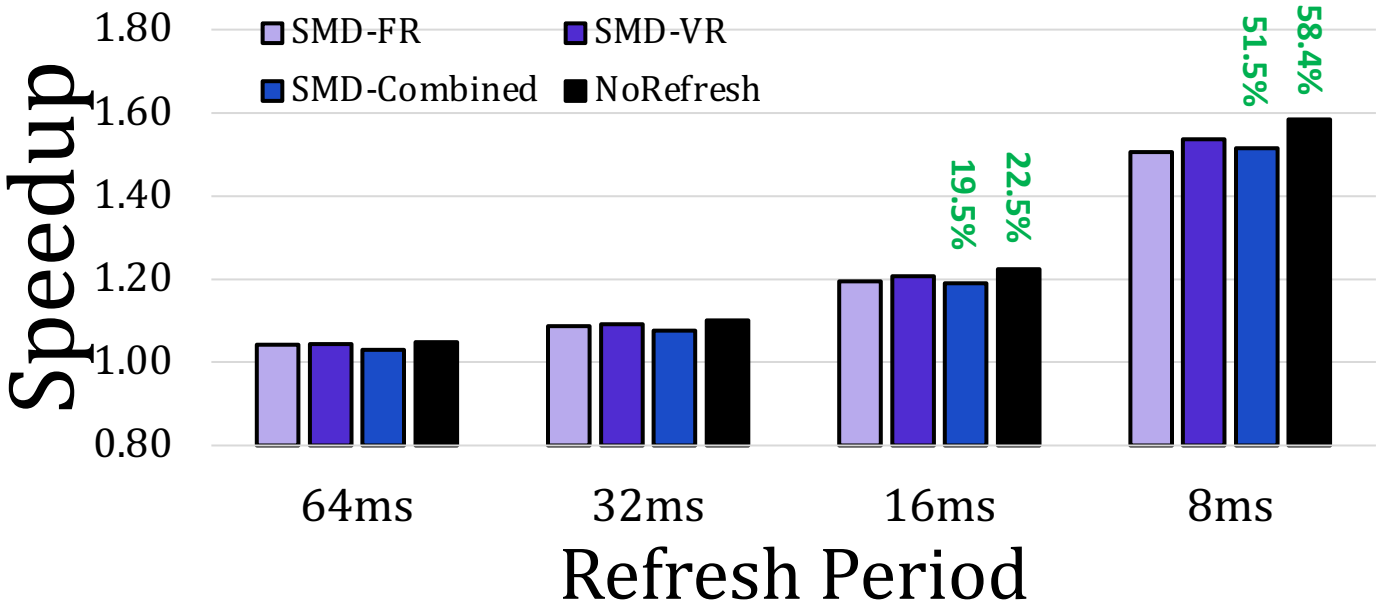
Four-Core Results





SMD-based maintenance mechanisms have **significant** *performance* and *energy efficiency* **benefits**

Sensitivity to Refresh Period



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

The three major DRAM maintenance operations:

- Refresh
- RowHammer Protection
- Memory Scrubbing

Source code is available:

github.com/CMU-SAFARI/SelfManagingDRAM

Implementing new **maintenance mechanisms** often requires difficult-to-realize changes

Our Goal

1. Ease the process of enabling new DRAM maintenance operations
2. Enable more efficient in-DRAM maintenance operations

Self-Managing DRAM (SMD)

Enables implementing new **in-DRAM** maintenance mechanisms
with **no further changes** in the *DRAM interface* and *memory controller*

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

A Case for Self-Managing DRAM Chips:
Enabling Efficient in-DRAM Maintenance Operations

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

PhD Thesis: “Improving DRAM Performance, Reliability, and Security by Rigorously Understanding Intrinsic DRAM Operation”

2023 EDAA Outstanding Dissertation Award

Backup Slides

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.

Random Numbers

Sequence of bits without any discernible pattern

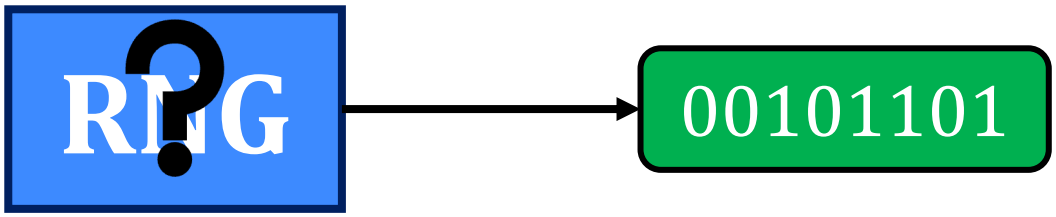
Random Number Candidates

00001111	**Pattern exists**
01010101	**Pattern exists**
00101101	**Pattern does not seem to exist**

Producing Random Numbers

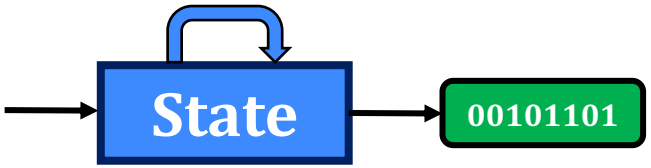
Random Number Generator (RNG)

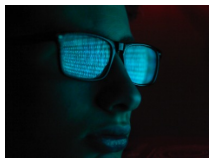
A device/program that generates random numbers



Pseudo-Random Number Generator (PRNG)

Arithmetic Transform





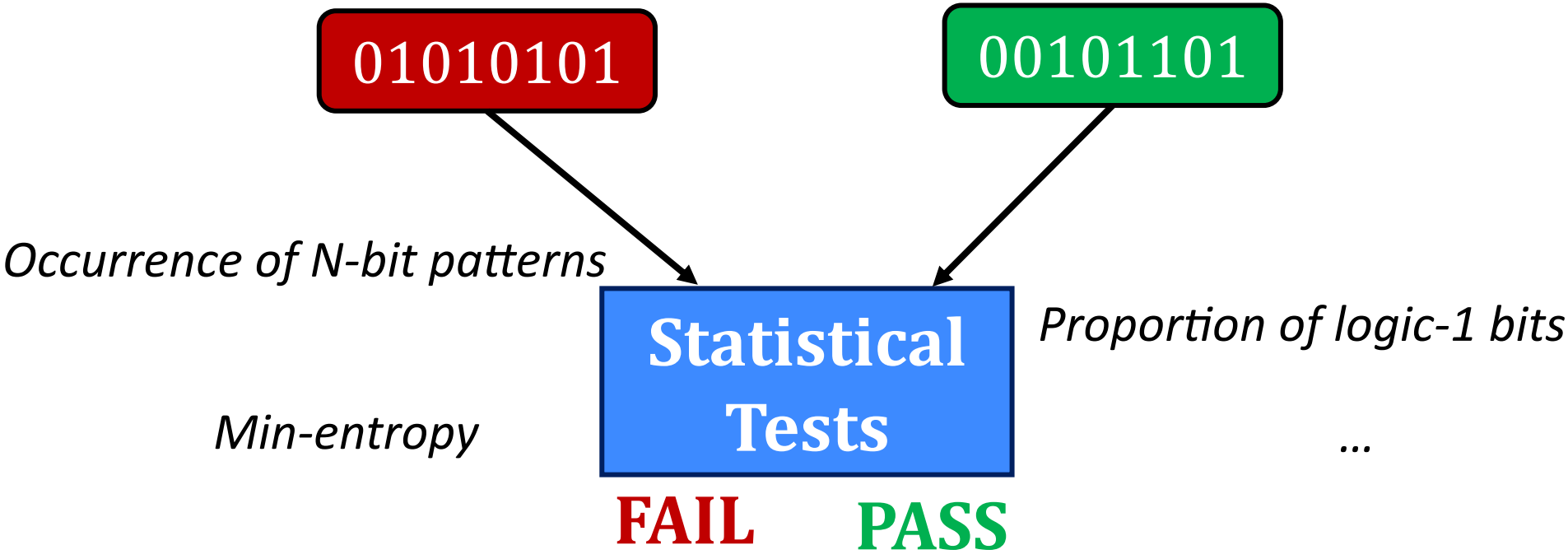
If seed is compromised
random number sequence can be regenerated

True Random Number Generator (TRNG)

TRNG output cannot be regenerated
from the physical process

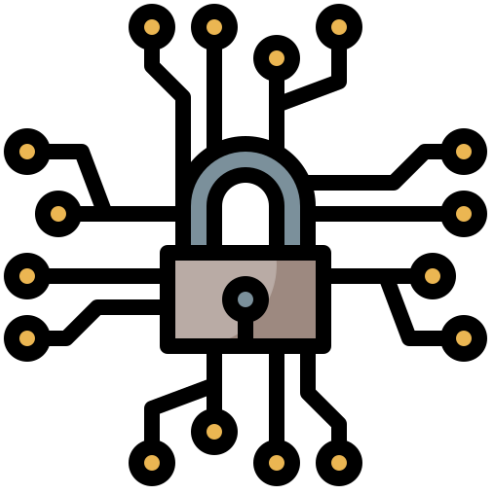
Assesing the Quality of a TRNG

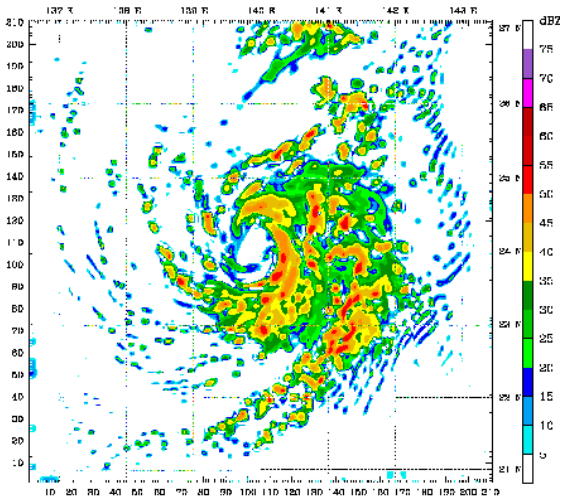
Systematically distinguish between
a low-quality and a high-quality TRNG

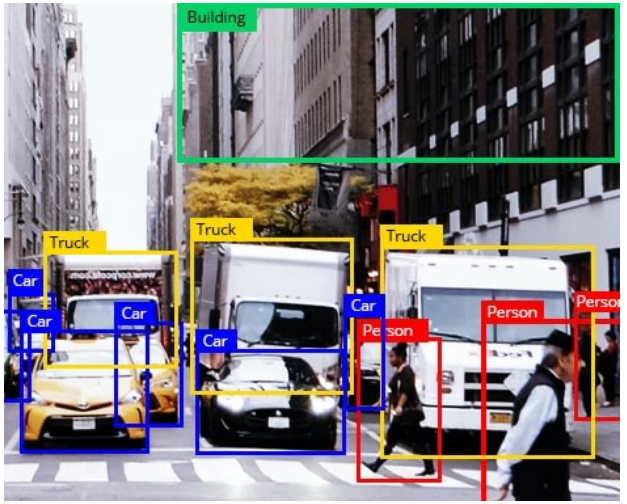


Use Cases of True Random Numbers

High-quality true random numbers are critical to many applications







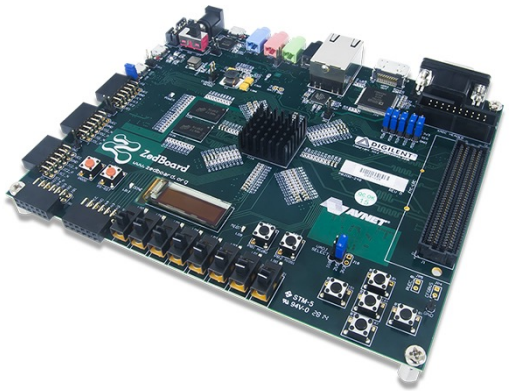
- Cryptographic Key Generation
- Signature Generation
- Monte Carlo Simulations
- Randomized Training
- Genetic Algorithms

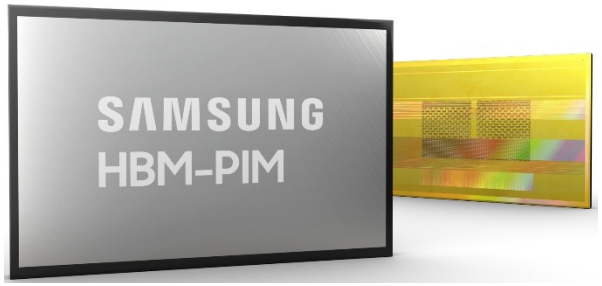
SAFARI kasirga

Use Cases of True Random Numbers

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

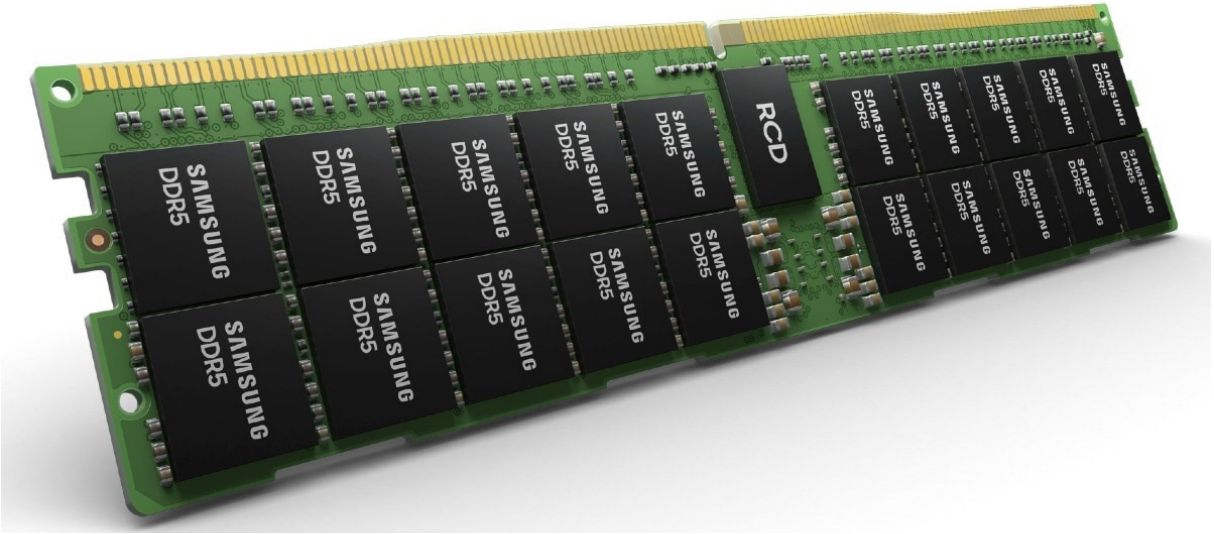
Unfortunately, not all computing systems are equipped with TRNG hardware (e.g., dedicated circuitry)





DRAM-Based TRNGs

DRAM chips are ubiquitous in modern computing platforms

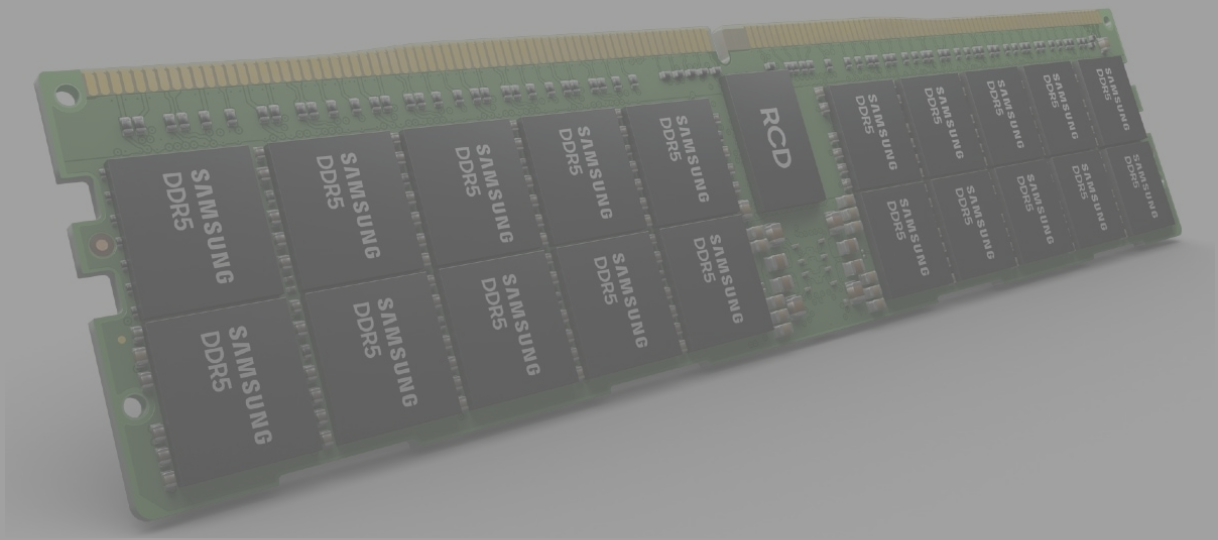


DRAM-based TRNGs enable true random number generation within DRAM chips

DRAM-Based TRNGs

Low-cost: No specialized circuitry for RNG

Beneficial for constrained systems



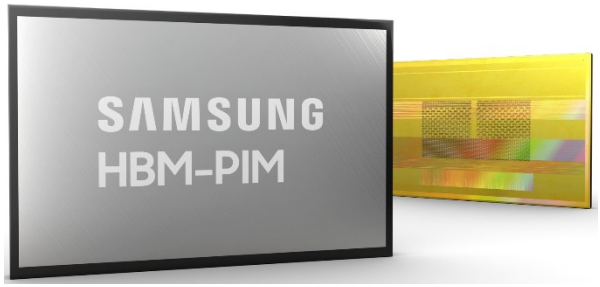
High-throughput: *> Gb/s throughput*

Enable applications that require high-throughput TRNG

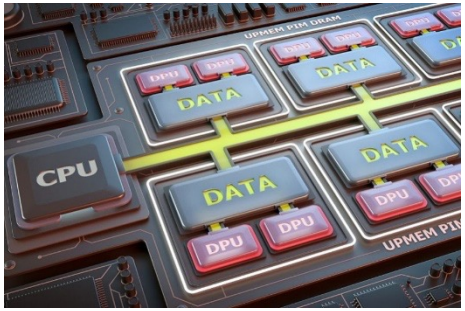
Synergy with Processing-in-Memory

Processing-in-Memory (PIM) Systems

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



[Samsung]



[UPMEM]

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

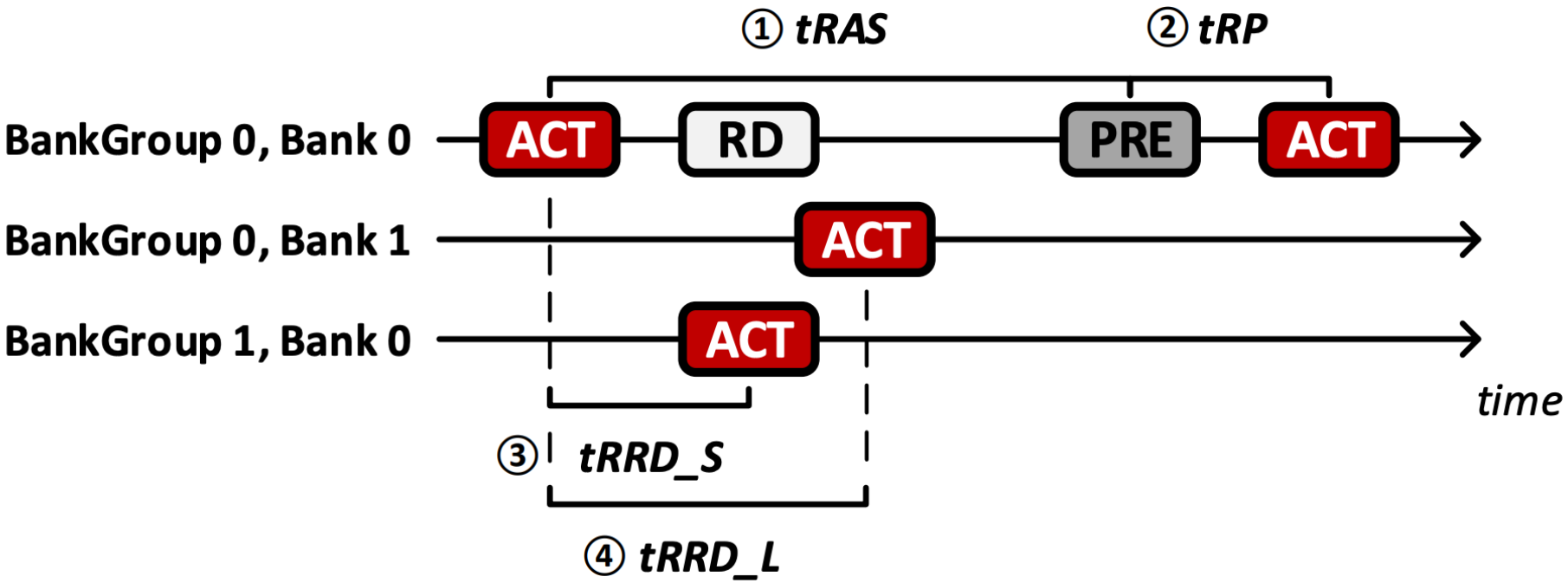


Figure 2: Timeline of key DDR4 commands.

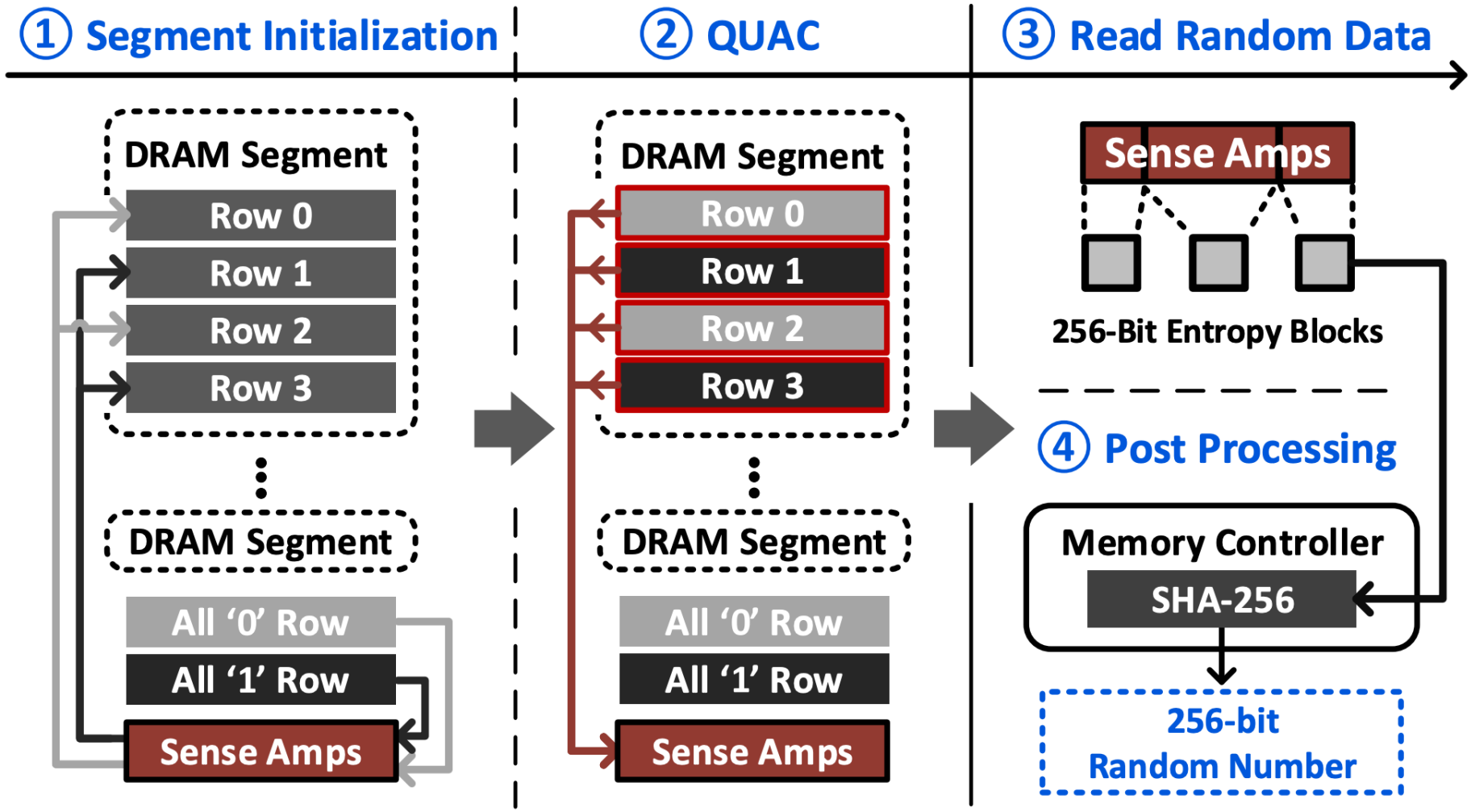
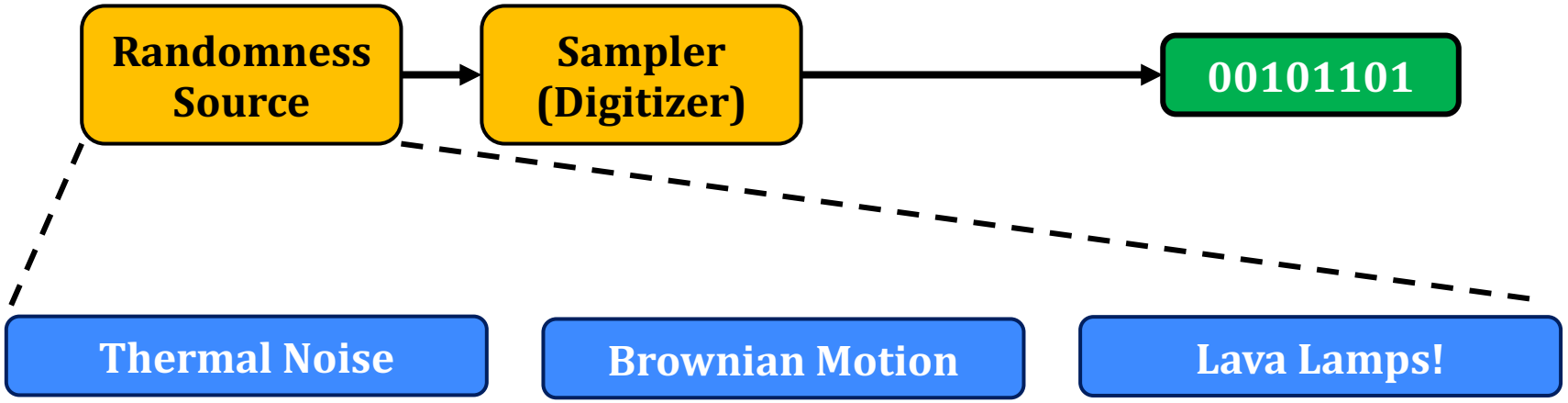
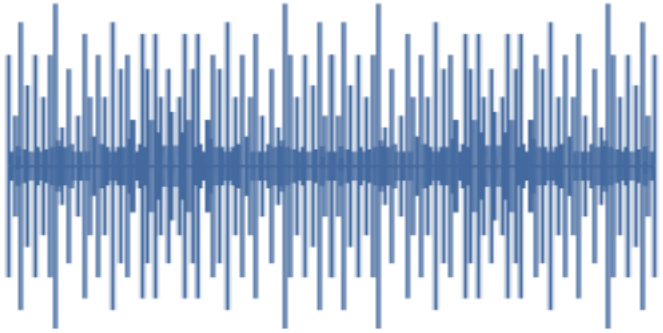


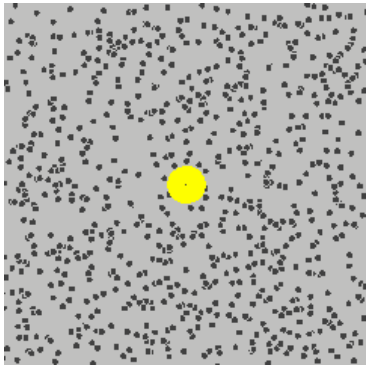
Figure 6: QUAC-TRNG mechanism.

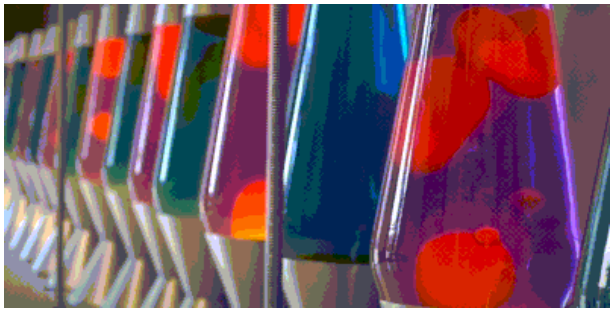
True Random Number Generators

Sample random physical phenomena

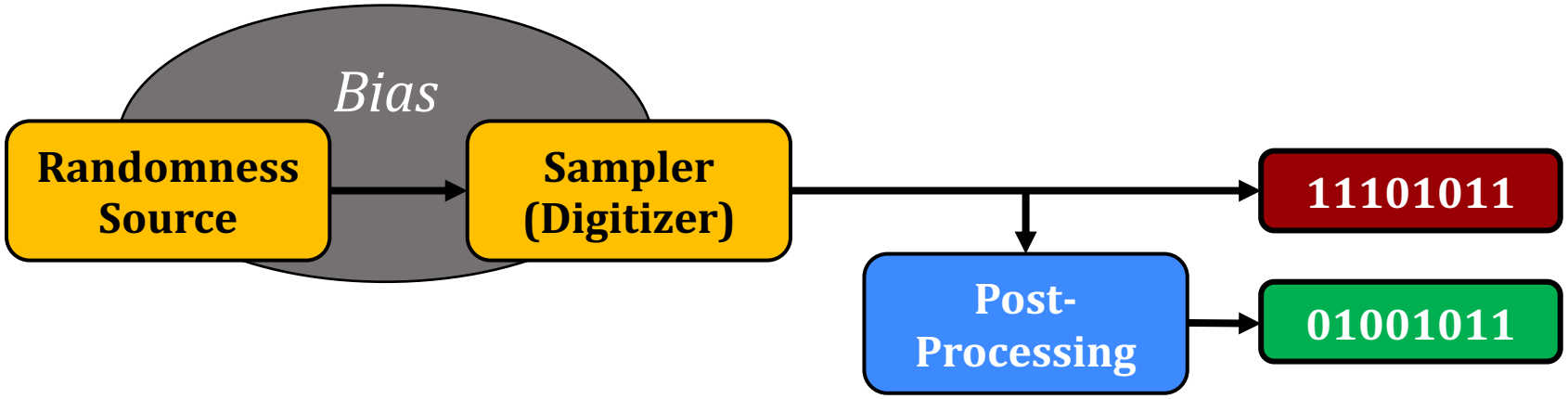








Post-processing for TRNGs



Bias in entropy source → bias in TRNG output

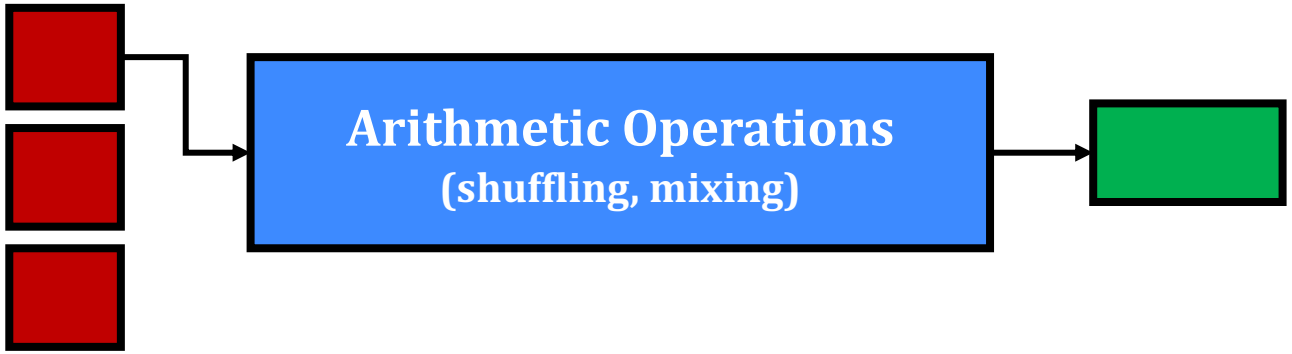
- Low quality (statistically) random numbers

Post-processing to remove bias

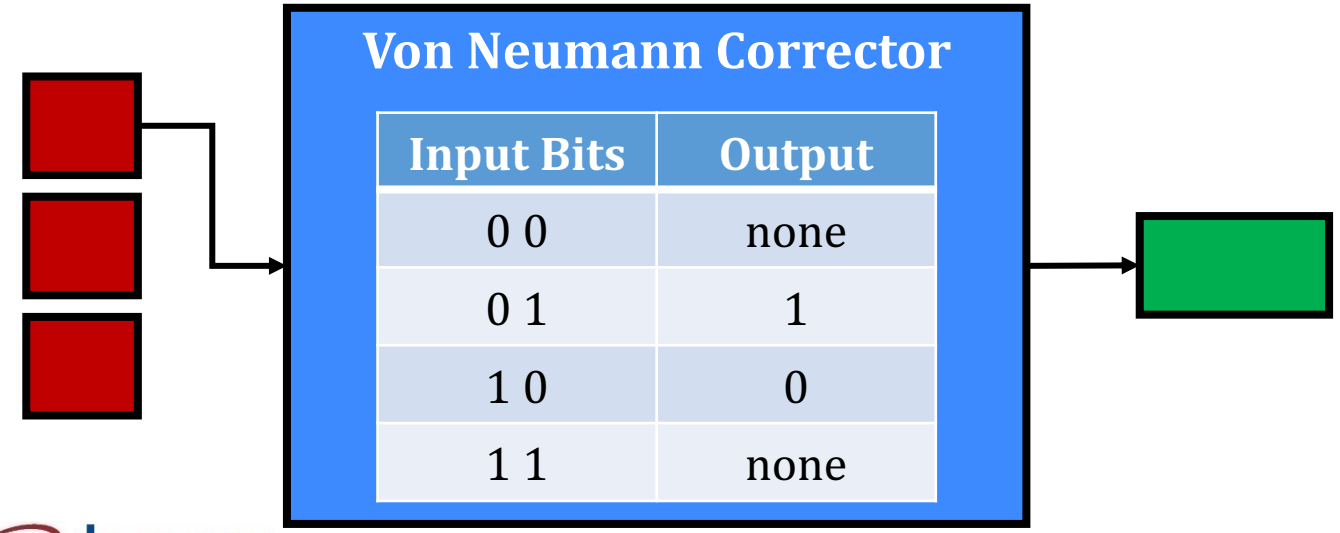
- Improve TRNG quality

SHA-256 and Von Neumann Corrector

Cryptographic Hash Functions



Von Neumann Corrector

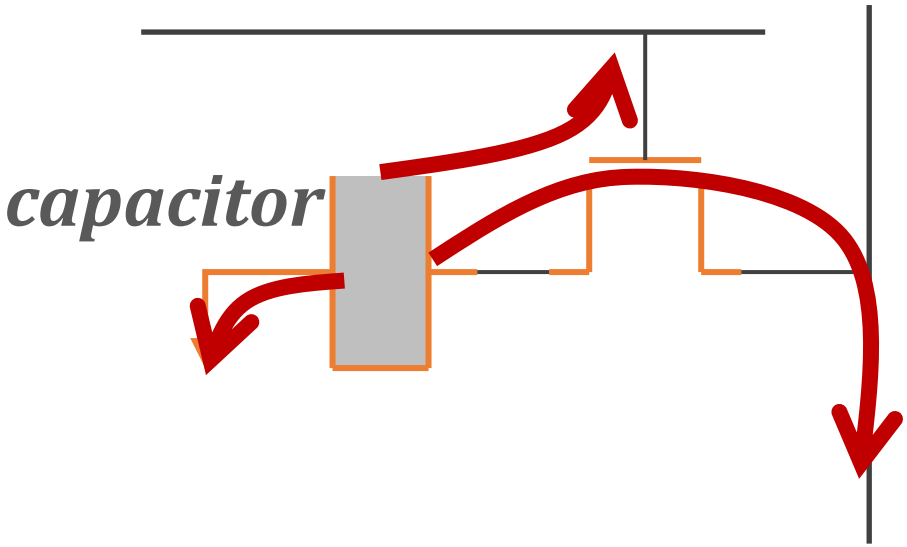


Entropy Sources in DRAM

1

Retention Failures

Fundamentally Slow: cells leak charge slowly



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

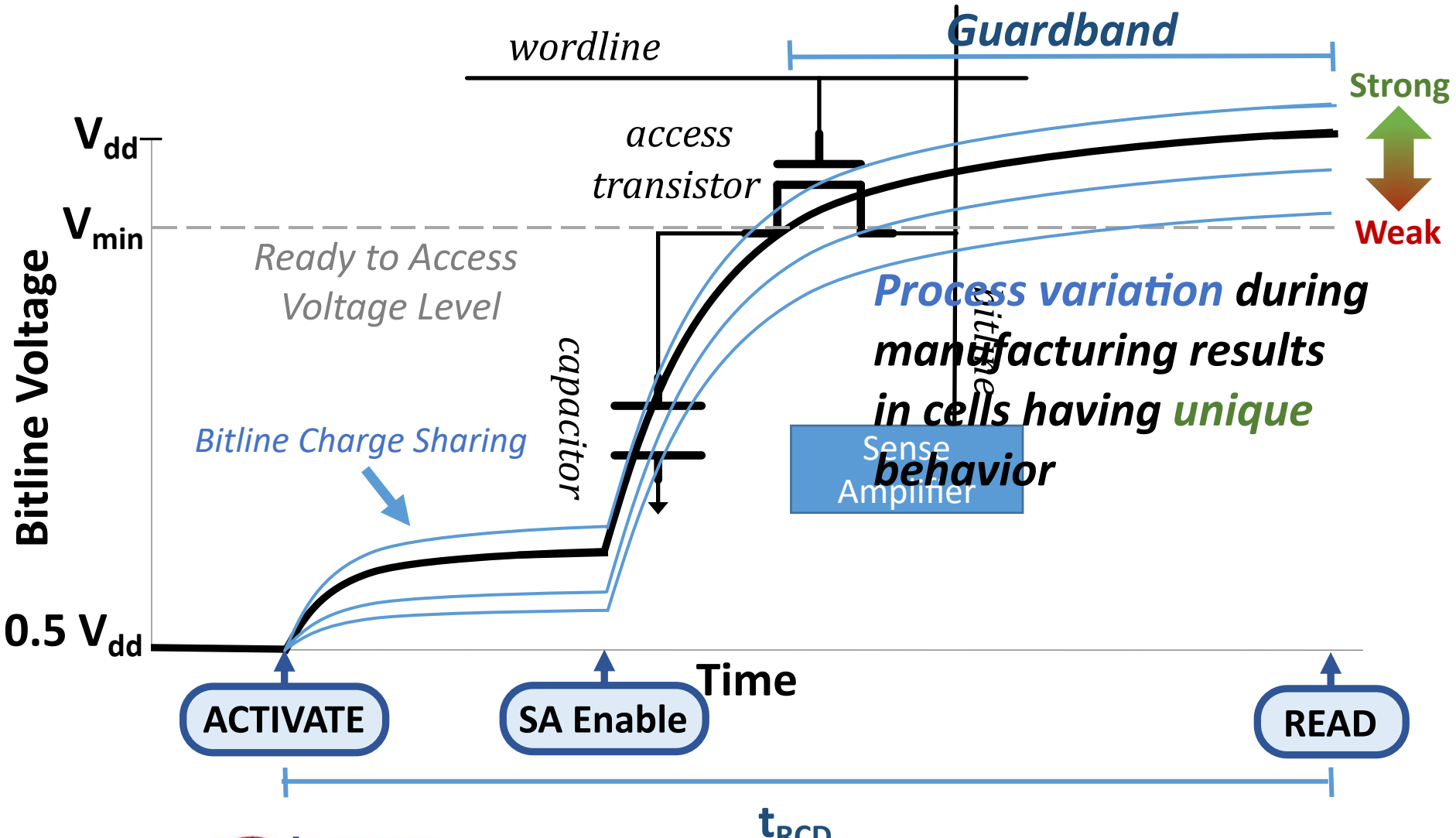
2

Talukder et al.

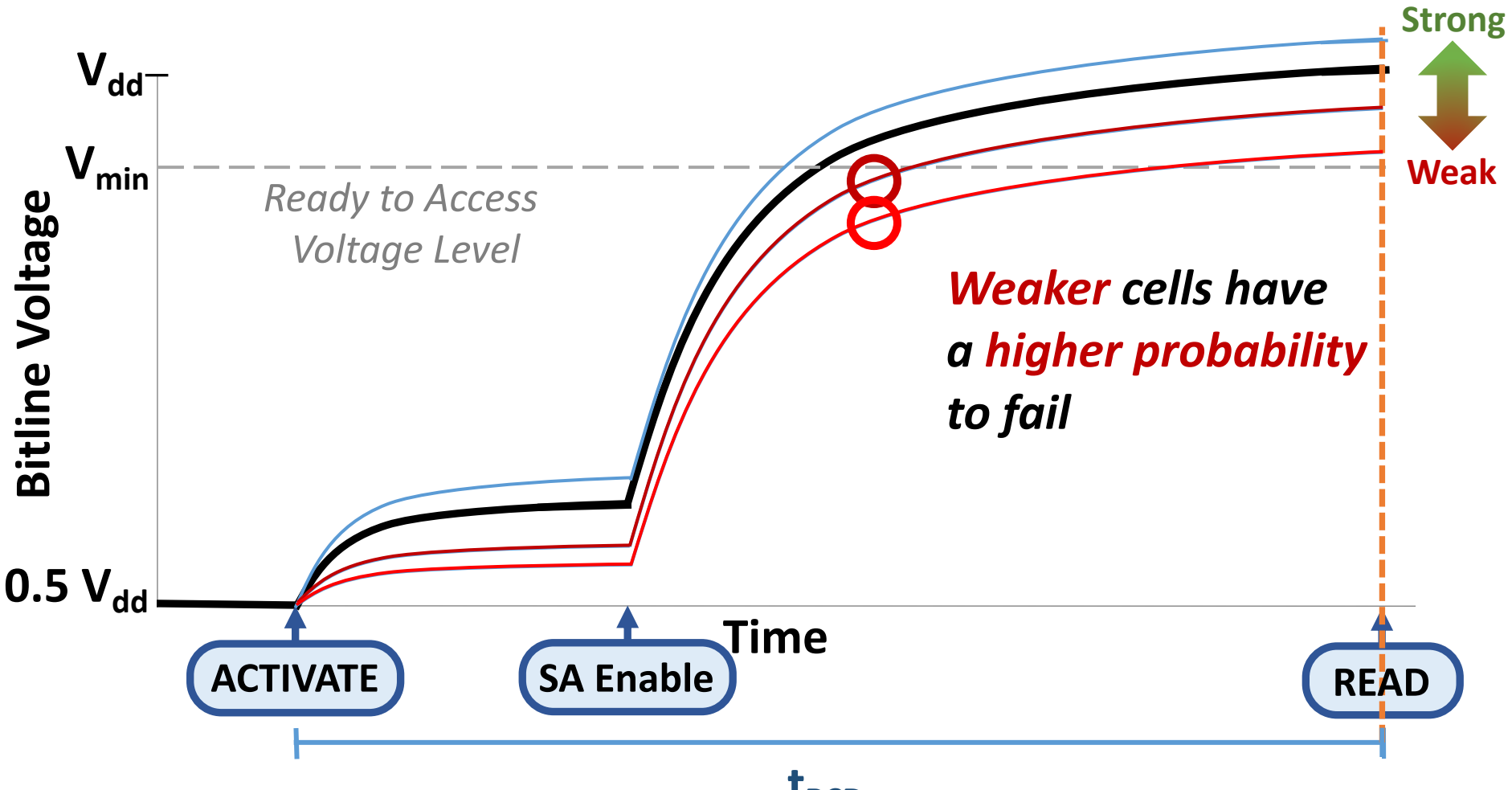
Precharge latency (tRP) failures

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

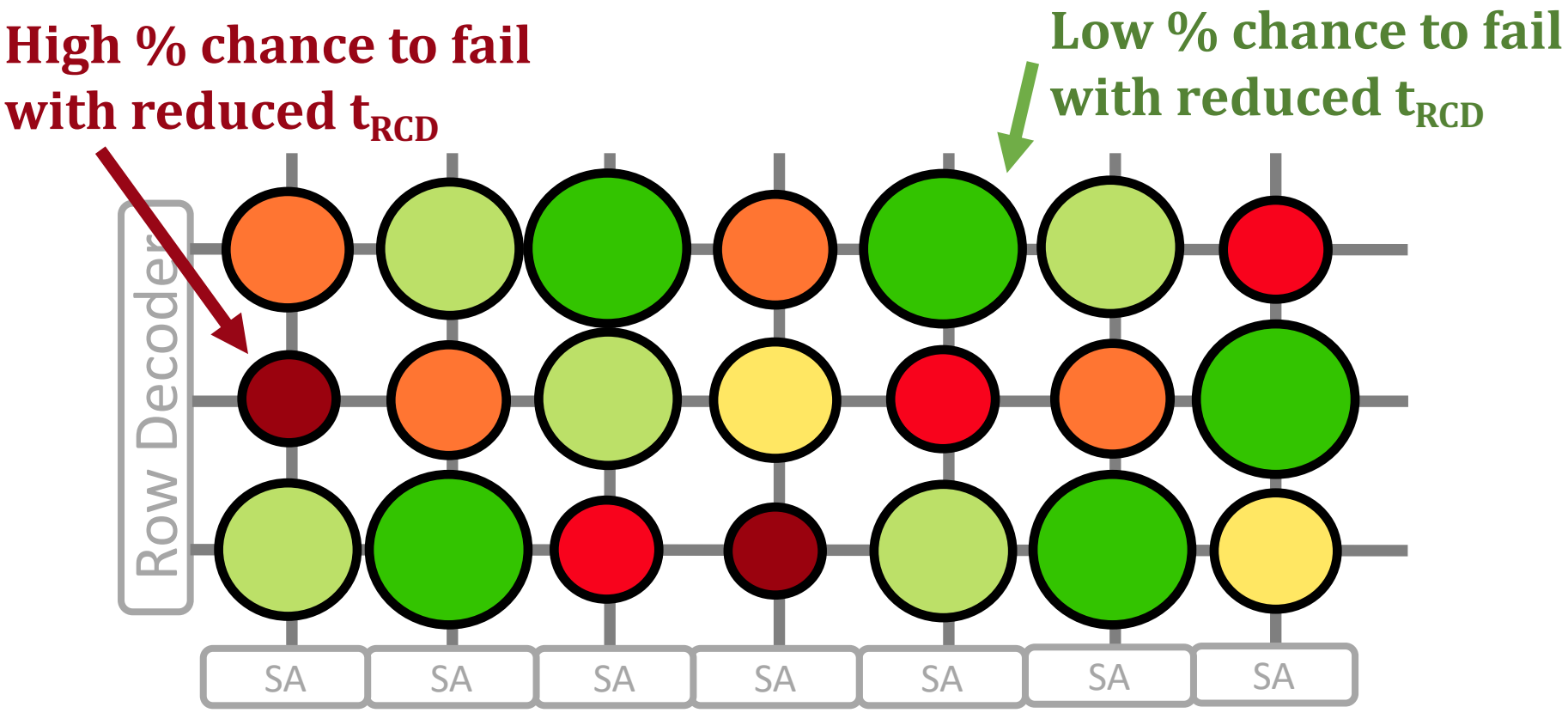
D-RaNGe: Accessing a Cell



D-RaNGe: Accessing a Cell



D-RaNGe Key Idea

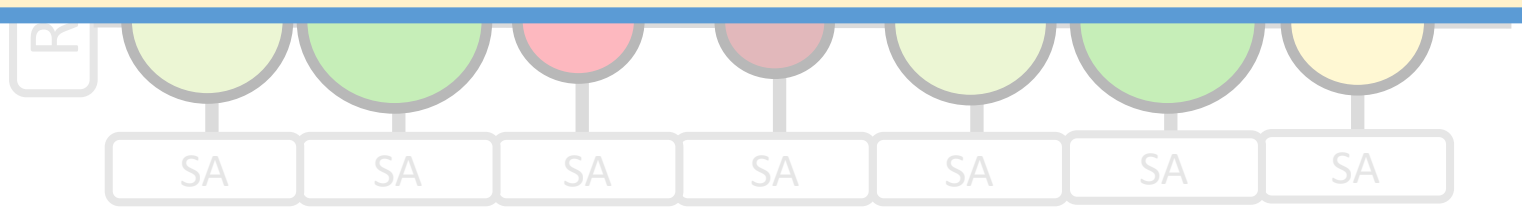


D-RaNGe Key Idea

High % chance to fail

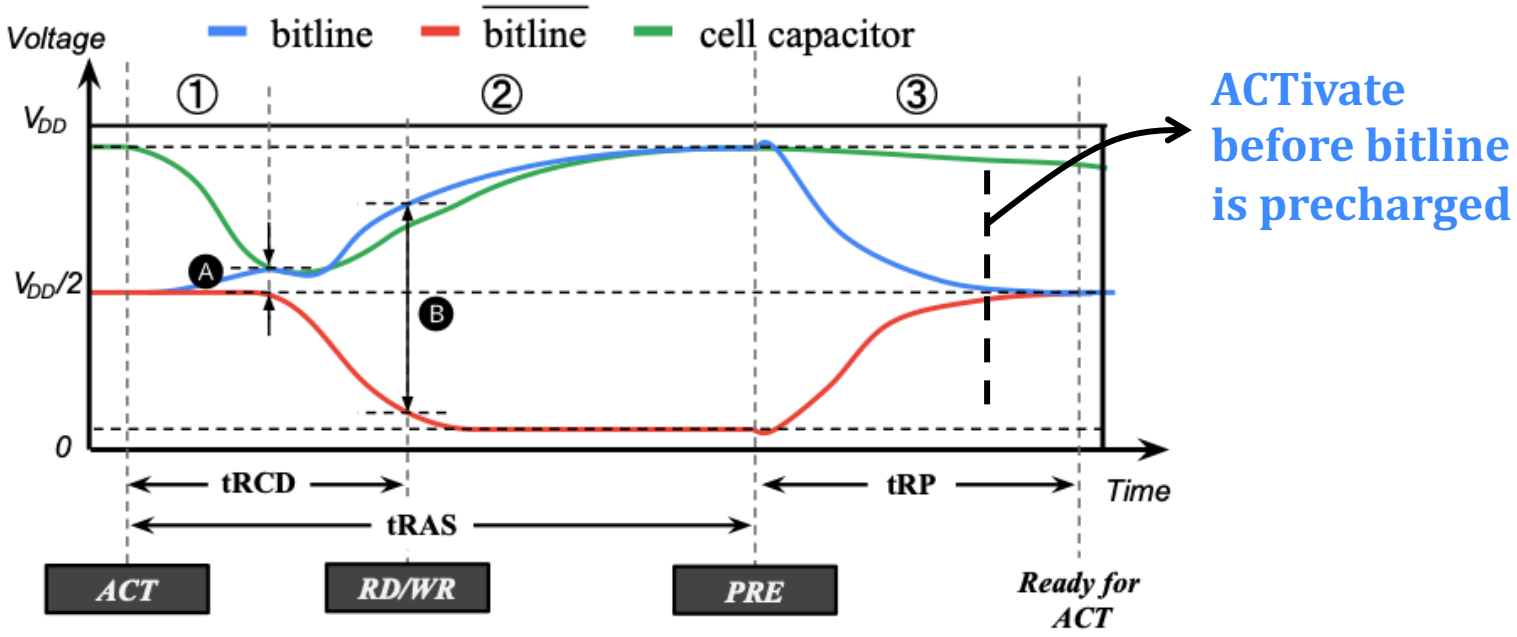
Low % chance to fail

The key idea is to extract random values by sampling DRAM cells that fail truly randomly



Talukder et al.

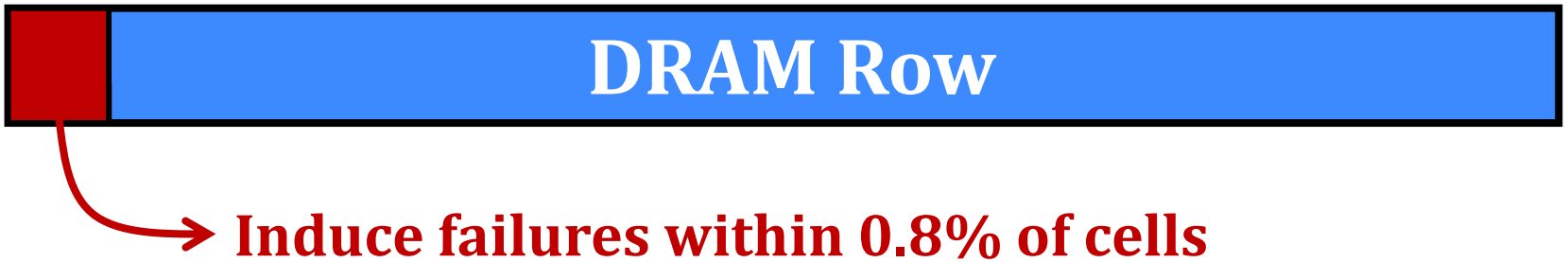
Key Idea: Sample DRAM cells that fail randomly, but with reduced tRP



Limitations of State-of-the-art

D-RaNGe

- Poorly utilizes activation granularity

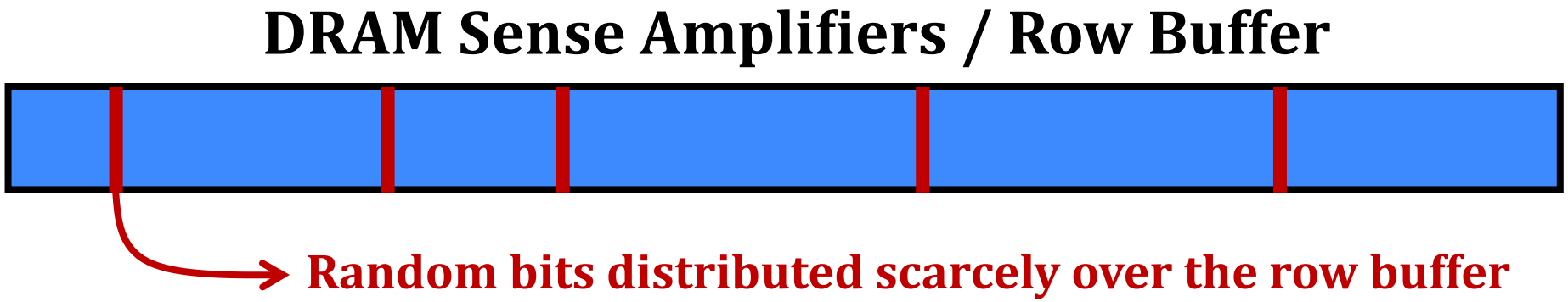


- Throughput limited by access latency

Limitations of State-of-the-art

Talukder et al.

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



- 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

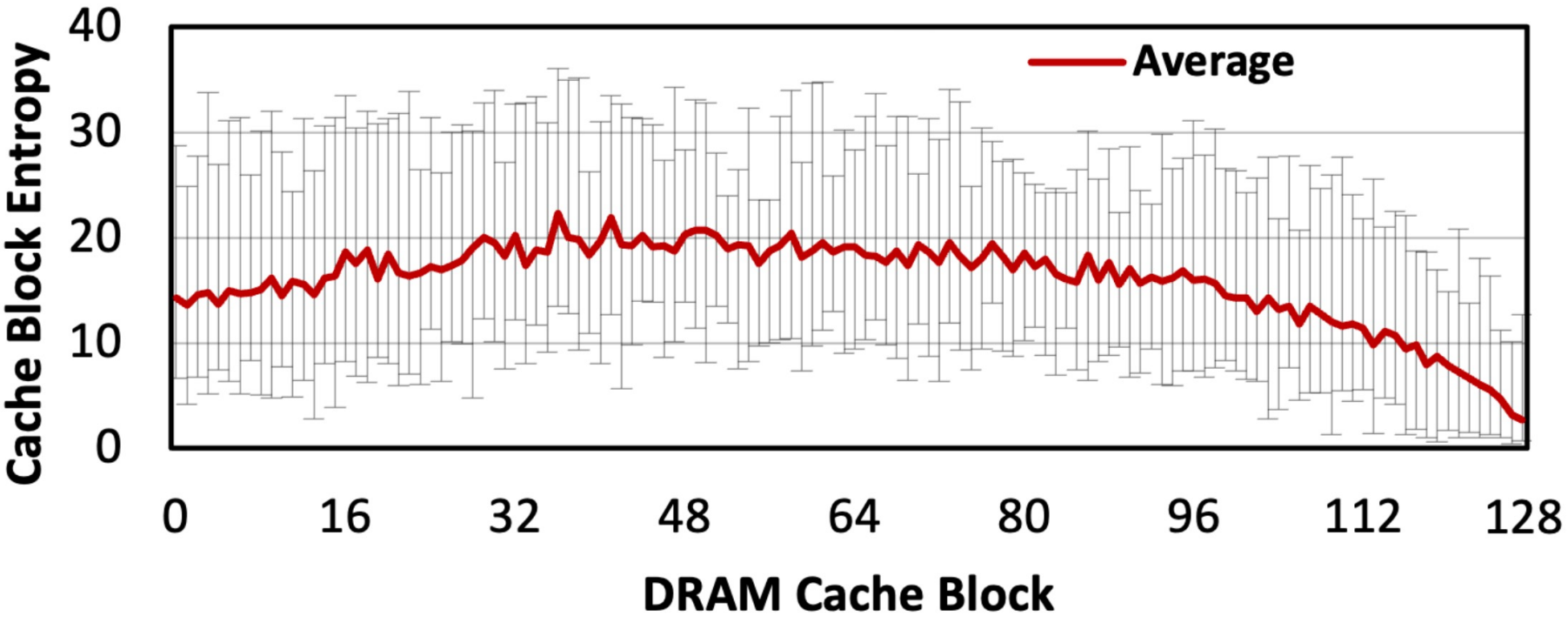
A **high-throughput** DRAM TRNG needs to:

(i) well utilize **activation granularity**

(ii) induce metastability on **many sense amplifiers**

- Throughput limited by poor-quality randomness source

Spatial Distribution

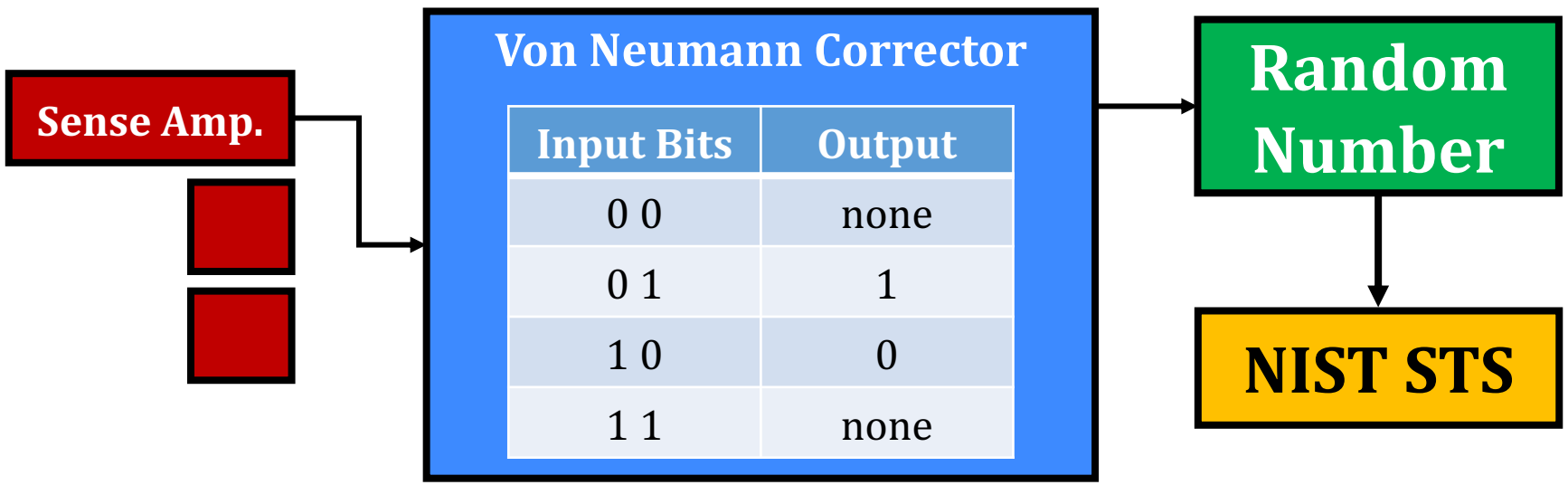


Cache block entropy is the highest around the middle of the DRAM segment

QUAC-TRNG's Quality

Collect bitstreams by repeatedly sampling bitlines after QUAC operations

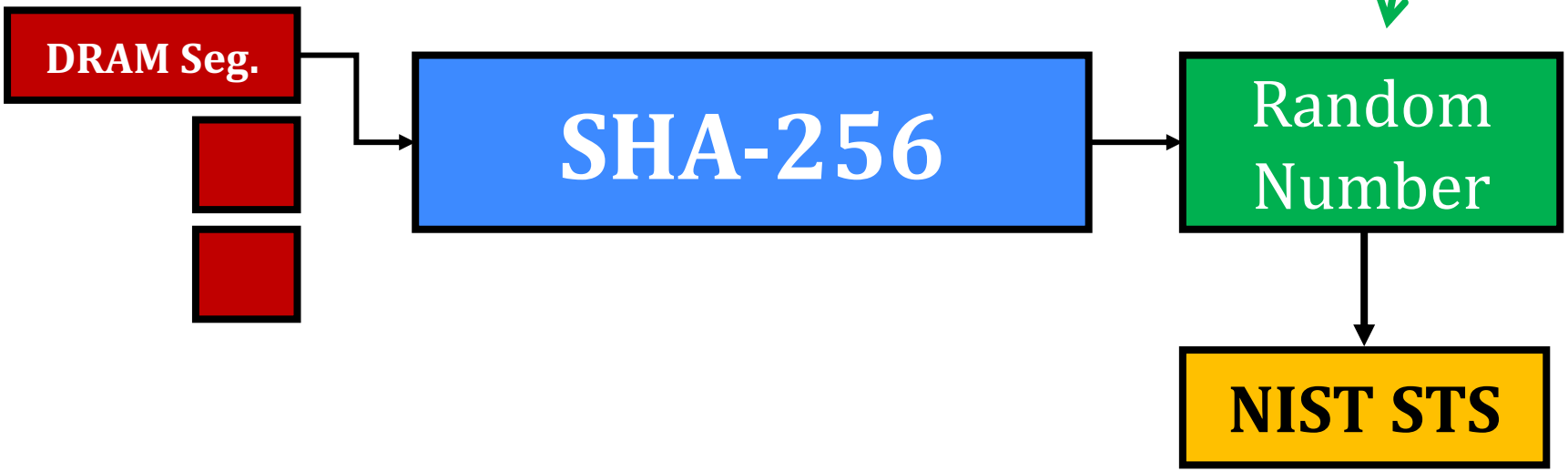
- 1 Mb random bitstreams
- Post-processing: Von Neumann Corrector



QUAC-TRNG's Quality

Collect bitstreams using QUAC-TRNG

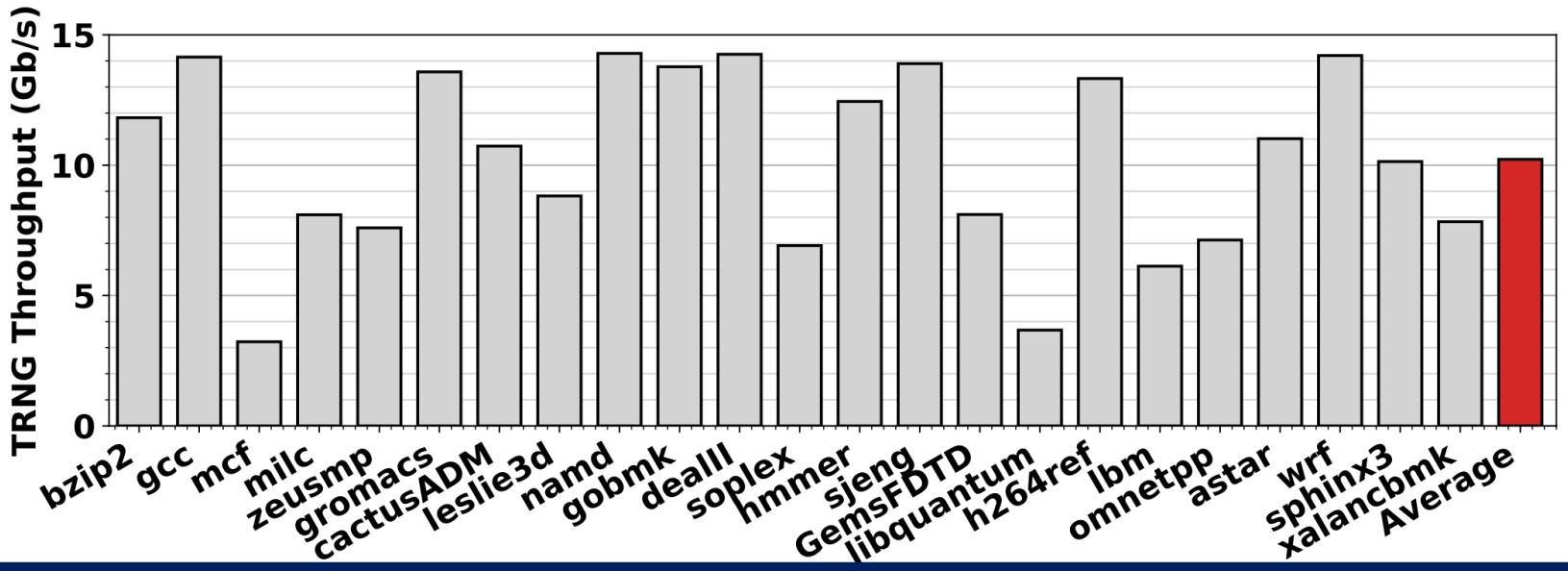
- 1 Gb bitstreams
- Post-processing: SHA-256



System Performance Study

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

Throughput & Latency Comparison

Table 2: Summary of prior DRAM-TRNGs vs QUAC-TRNG

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

Based on fundamentally slow processes

Memory Overhead

The memory allocated for QUAC-TRNG is only 192 KiBs 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

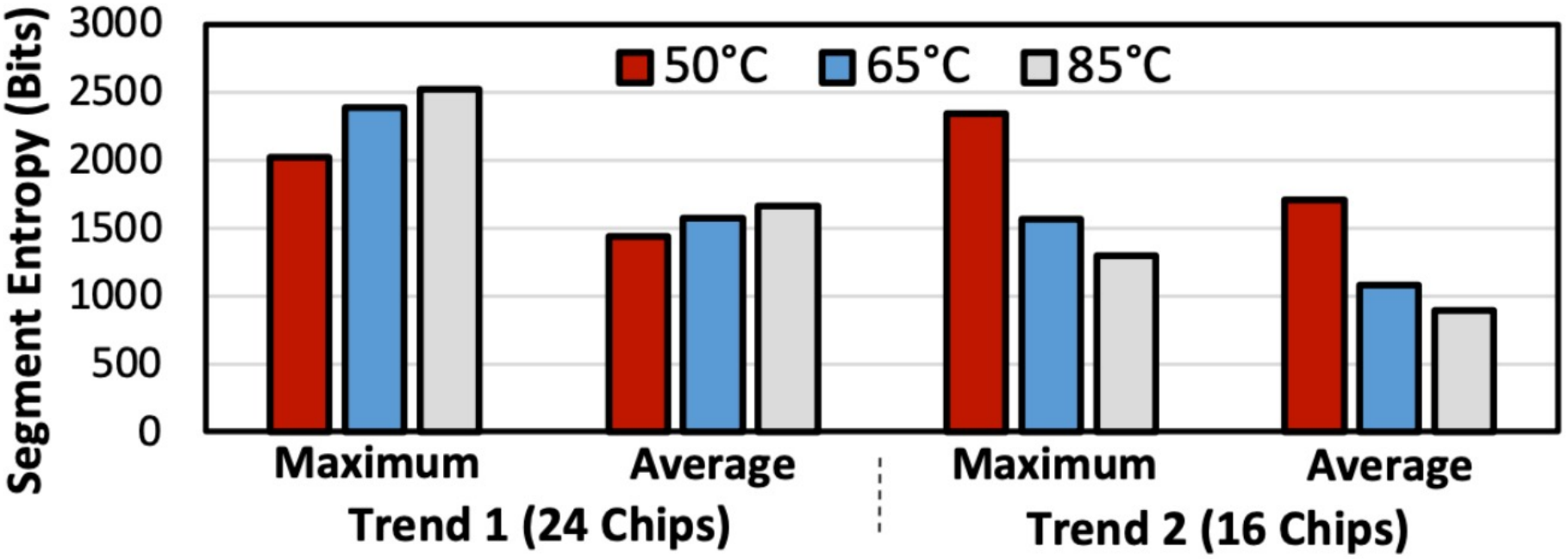
Record bitline entropy at 50, 65 and 85°C

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

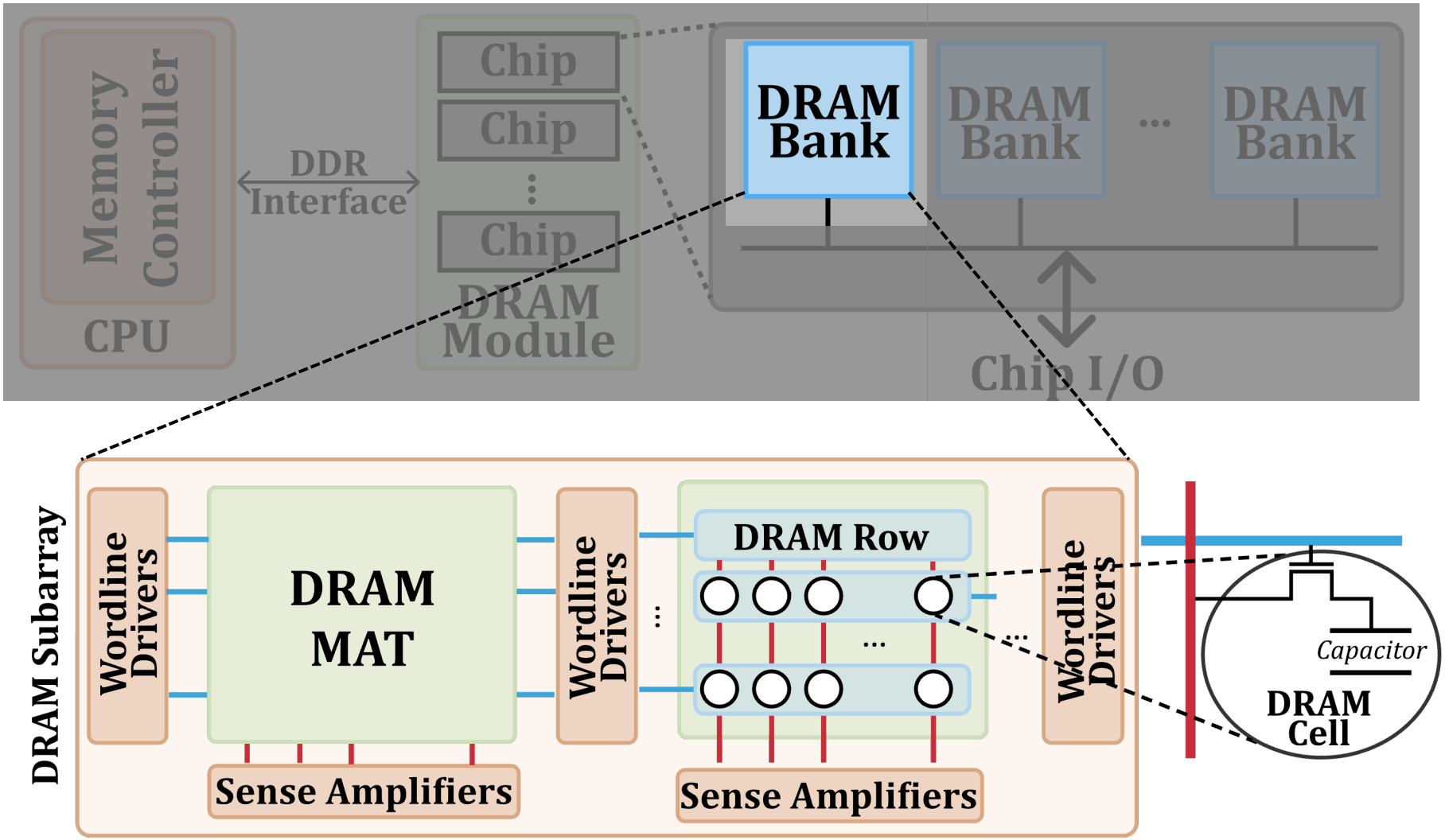
Time Dependence

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

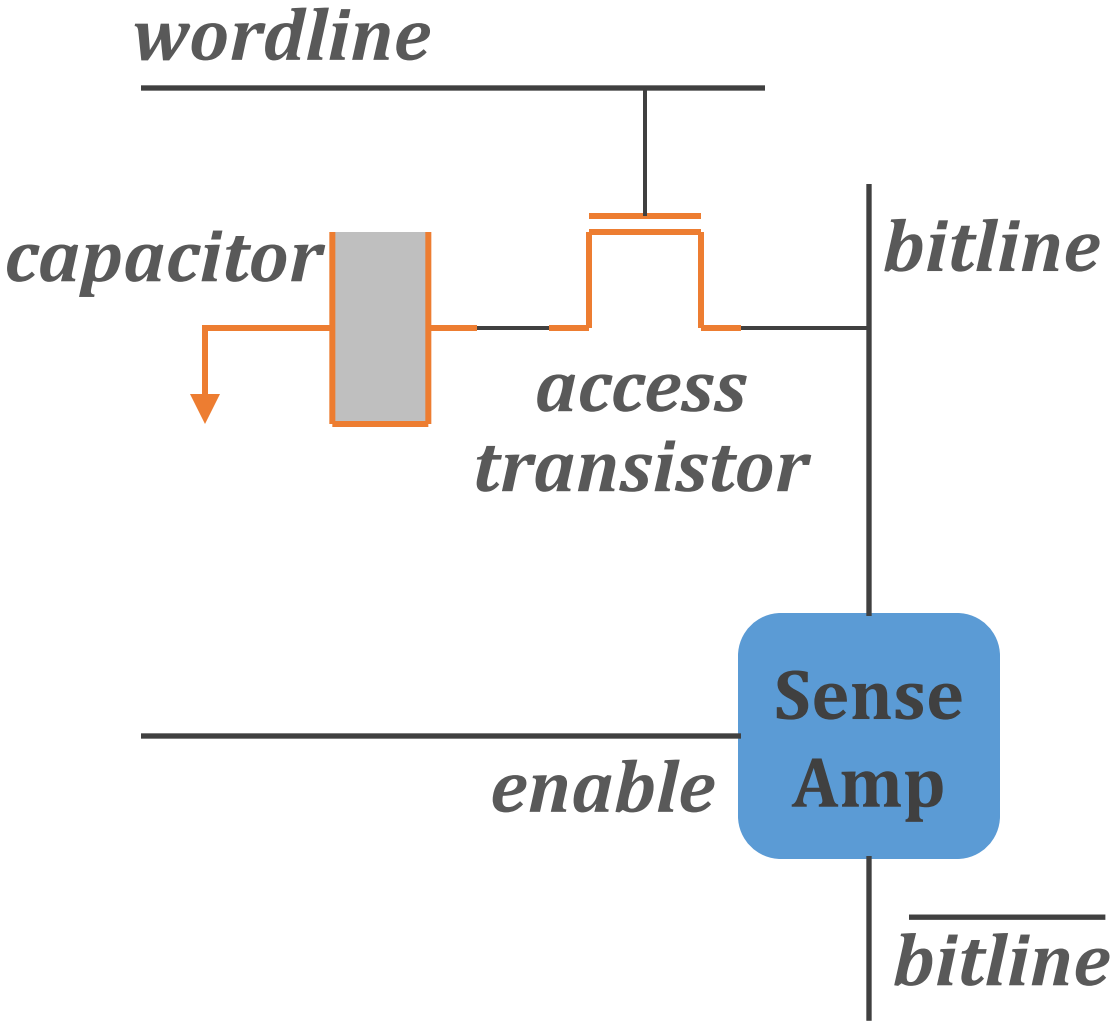
- 2.4% average across all chips
- Maximum difference is 5.2% (minimum is 0.9%)

Characterized entropy is valid for at least 30 days

DRAM Organization



Accessing a DRAM Cell



Accessing a DRAM Cell