D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim[‡§] Minesh Patel[§] Hasan Hassan[§]
Lois Orosa[§] Onur Mutlu[§‡]

[‡]Carnegie Mellon University [§]ETH Zürich

Presented by Fredrik Strupe

ETH Zürich
2 May 2019

Executive Summary

Executive Summary

- **Motivation**
  - True random number generation enables security applications like cryptography and simulations
  - Many systems lack TRNG hardware devices, but got DRAM
- **Problem**
  - Existing DRAM-based RNG solution are either not fundamentally non-deterministic or are too slow
- **Goal**
  - A low-latency, high-throughput TRNG based on DRAM
- **Solution**
  - Reduce timing constraints when reading values from DRAM and extract randomness from failing DRAM cells
- **Evaluation**
  - Tested on 282 LPDDR4 DRAM devices
  - Achieves *100 ns* latency and *717.4 Mb/s* throughput

Problem & Goal

Problem

- True random number generators (**TRNGs**) generate TRNs by extracting randomness from some physical entropy source

- This can be *slow* (e.g. through human input) or require *extra hardware*

- Existing DRAM-based solutions are too slow for high-throughput applications

Goal

- A *high-throughput, low latency* DRAM-based TRNG
  - Can we do this by exploiting some DRAM characteristic?

Background

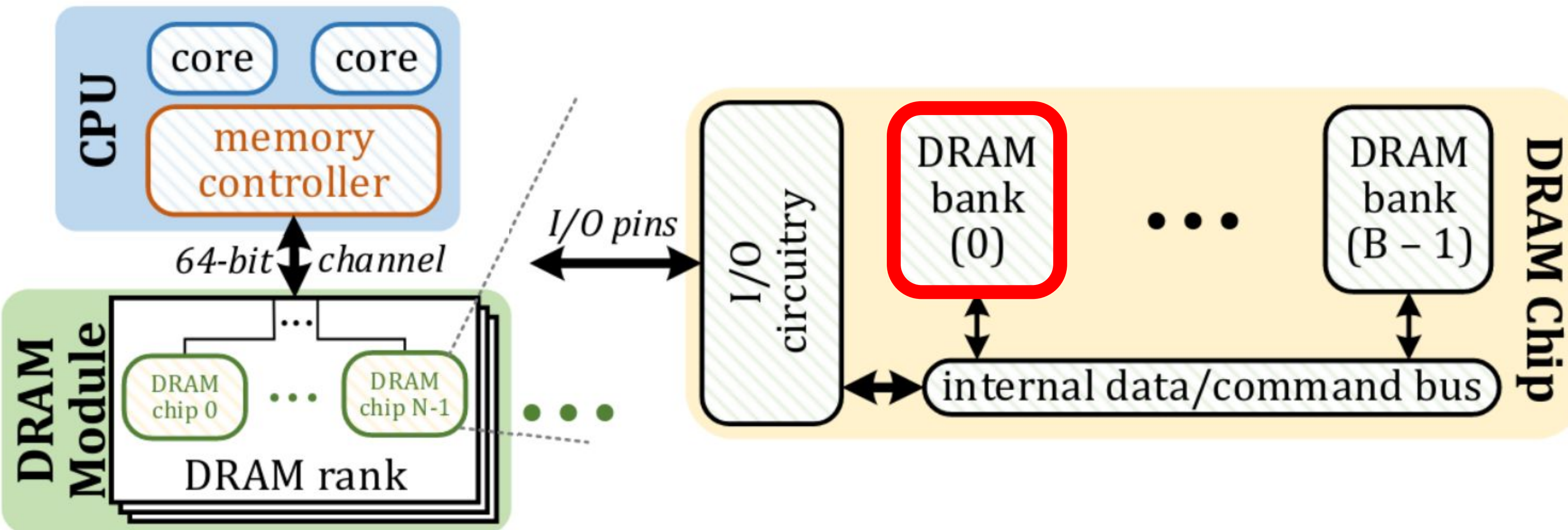
True Random Number Generators

- Numbers from a TRNG *only* depend on some random noise obtained from a physical process, and *not* any previously generated numbers

- An *effective* TRNG must satisfy six key properties:
  - Low implementation cost
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead

DRAM Organization

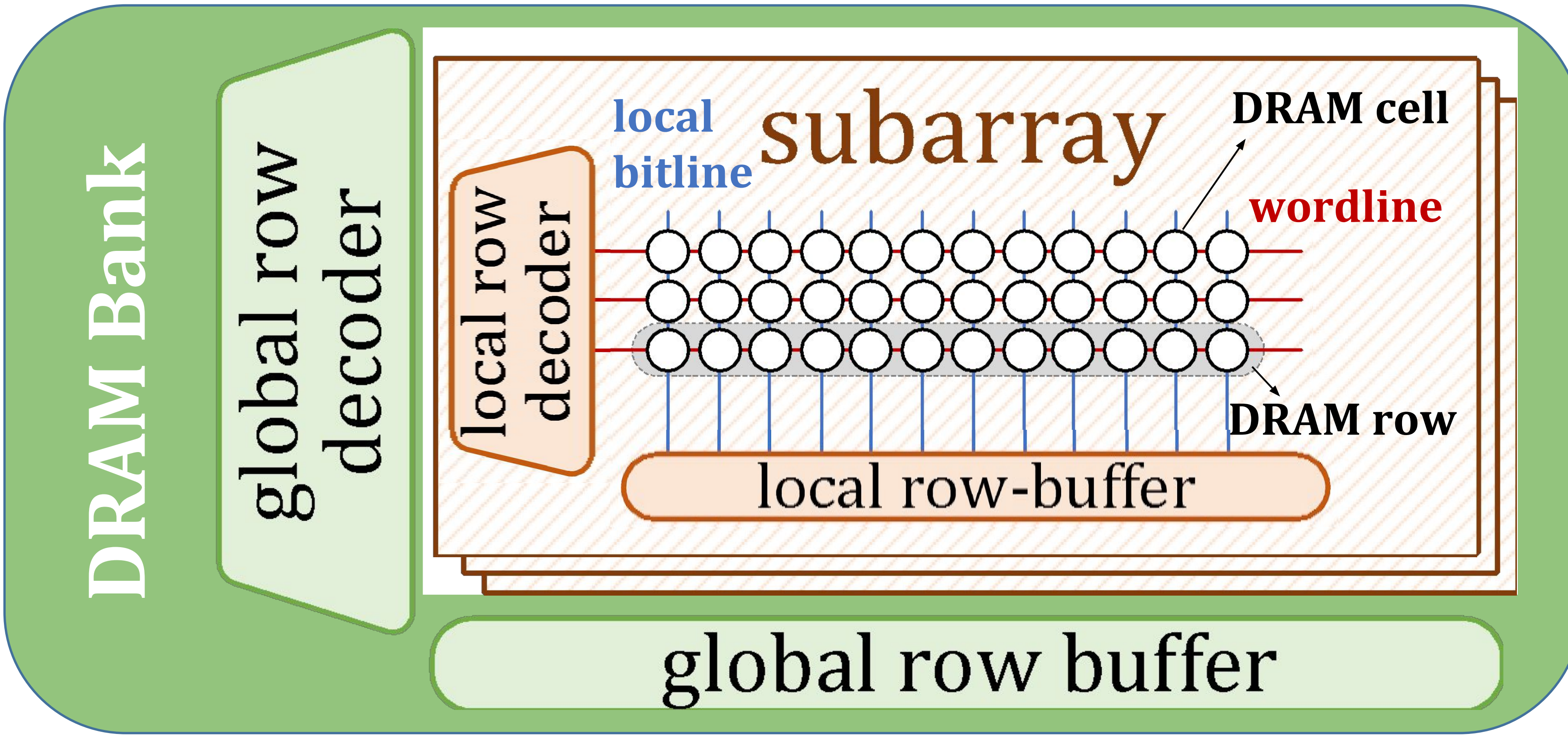
- DRAM is structured hierarchically



- Module → Rank → Chip → Bank

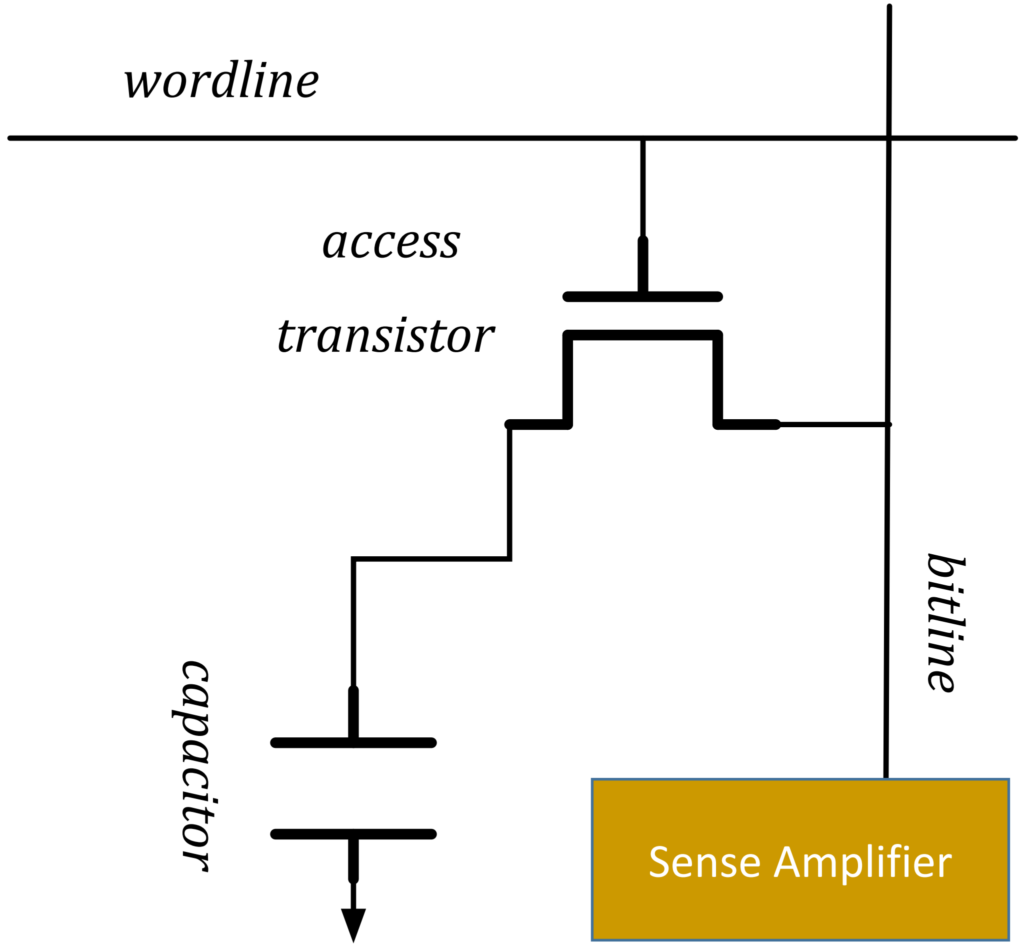
DRAM Organization

- A bank contains an array, further divided into subarrays



Source: https://people.inf.ethz.ch/omutlu/pub/drange-dram-latency-based-true-random-number-generator_hpca19-talk.pdf

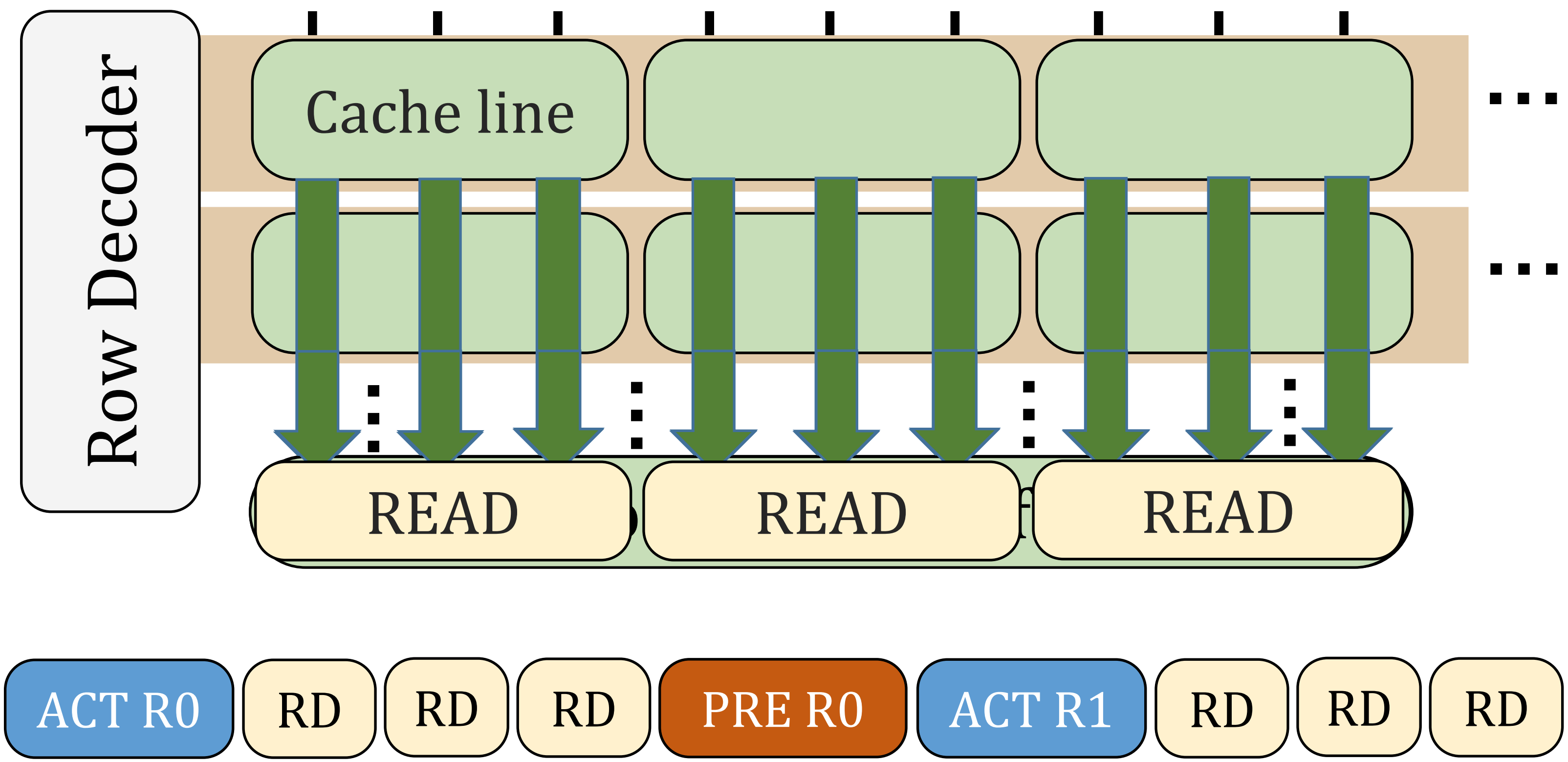
DRAM Cell



Source: https://people.inf.ethz.ch/omutlu/pub/drange-dram-latency-based-true-random-number-generator_hpca19-talk.pdf

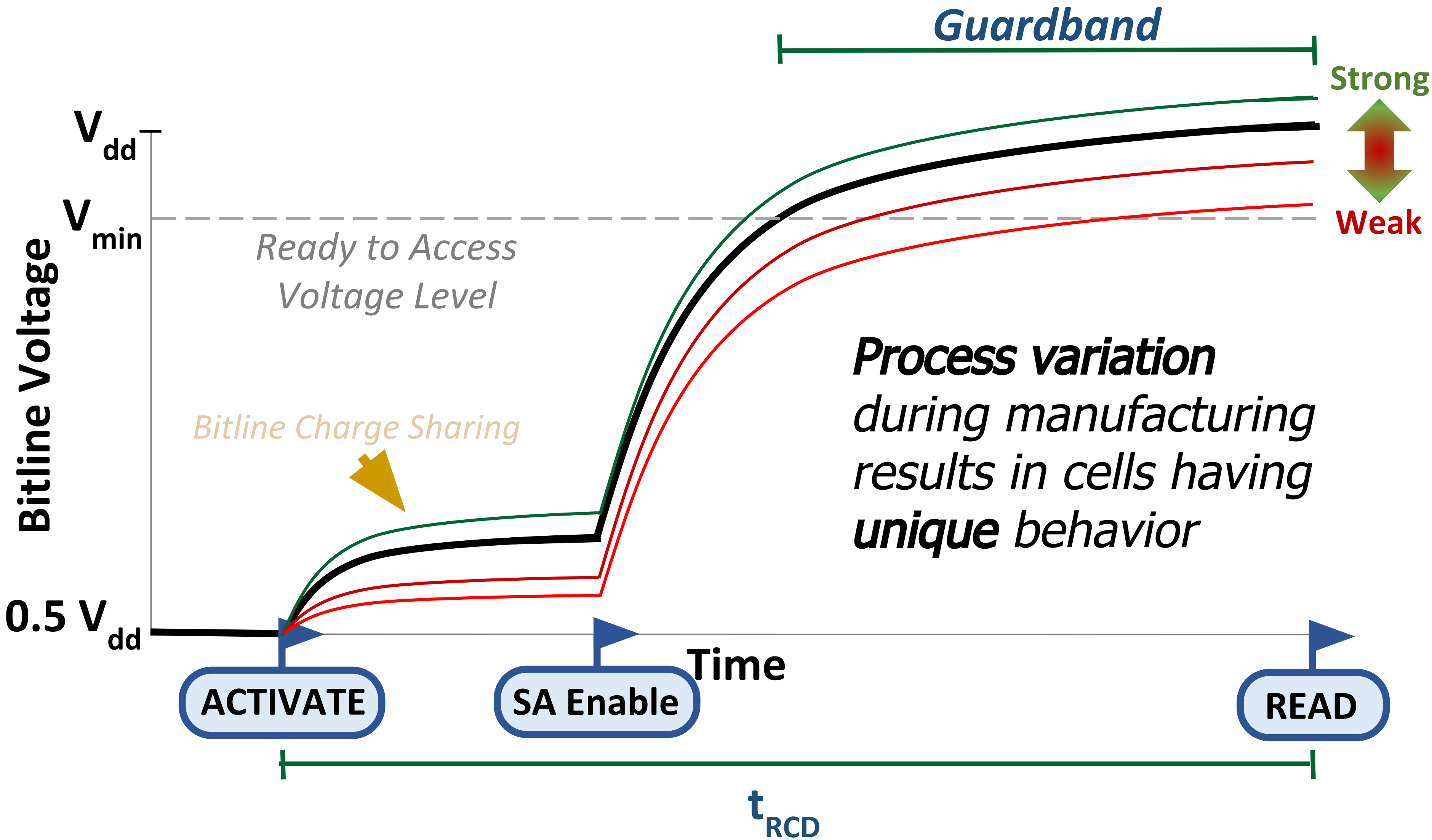
DRAM Operation

- Three main commands for reading:
  ACTIVATE, READ and PRECHARGE



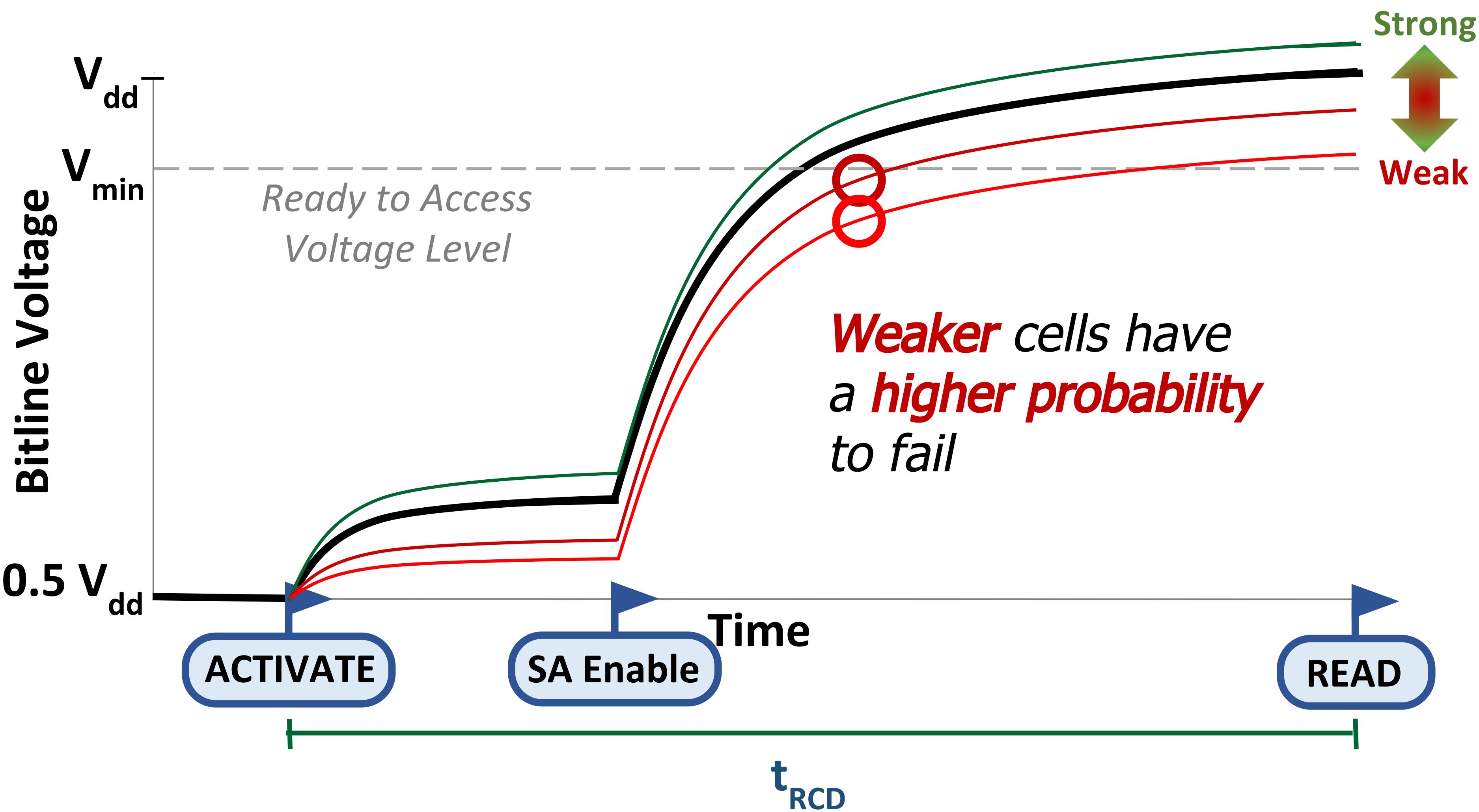
Source: https://people.inf.ethz.ch/omutlu/pub/drange-dram-latency-based-true-random-number-generator_hpca19-talk.pdf

DRAM Accesses and Failures



Source: https://people.inf.ethz.ch/omutlu/pub/drange-dram-latency-based-true-random-number-generator_hpca19-talk.pdf

DRAM Accesses and Failures



Source: https://people.inf.ethz.ch/omutlu/pub/drange-dram-latency-based-true-random-number-generator_hpca19-talk.pdf

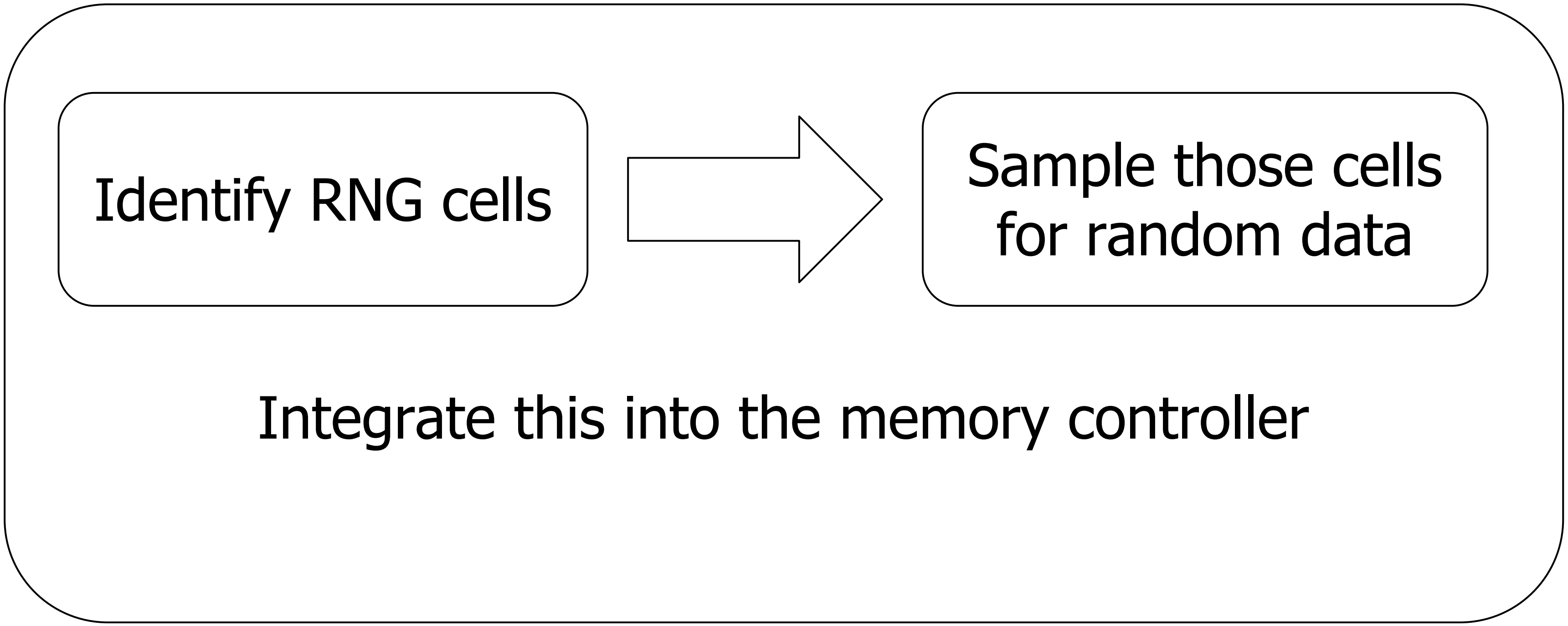
Novelty, Key Approach & Ideas

Novelty

- With a reduced t_{RCD}, some cells fail with a probability close to 50%

- Use these cells as an entropy source for random number generation!

Key Approach



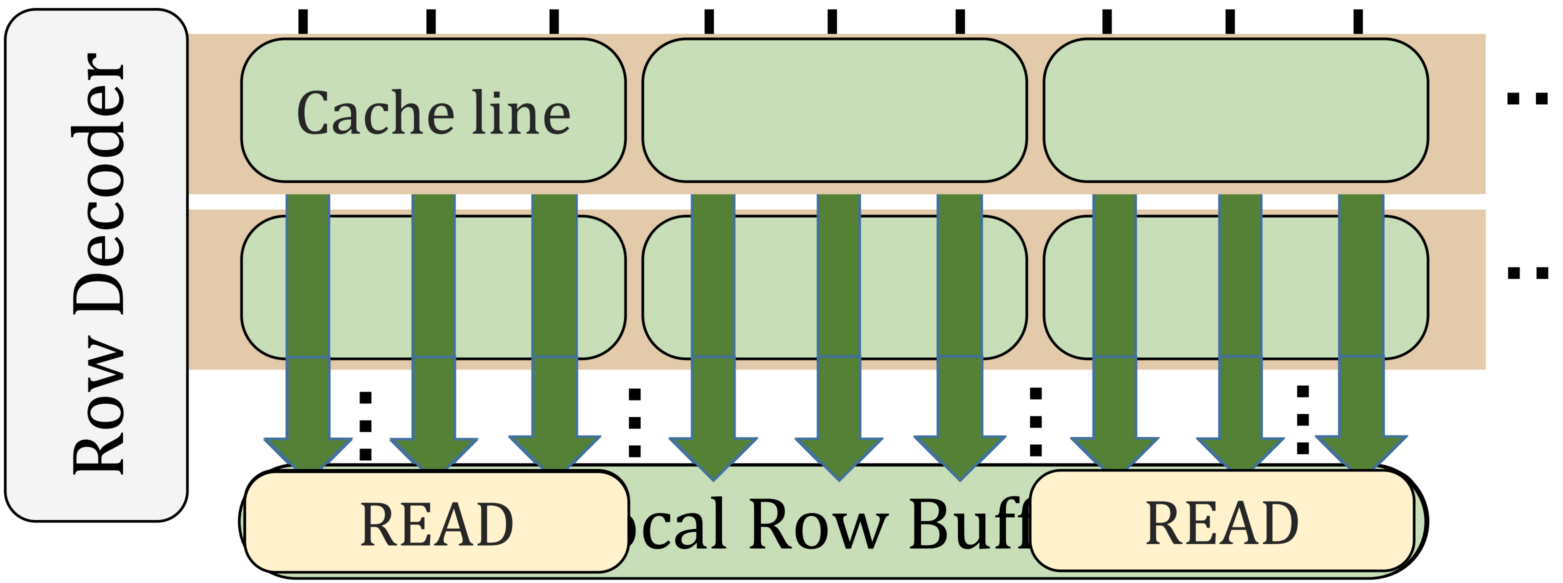
Mechanisms

RNG Cell Identification

- Write some initial data pattern into DRAM
- Read every cell 1000 times with a reduced t_{RCD} (each time with a fresh ACTIVATE)
- Calculate the Shannon (information theoretic) entropy of each cell's generated bitstream

RNG Cell Sampling

- For maximum throughput, alternate between reading two separate rows with the highest number of RNG cells



Source: https://people.inf.ethz.ch/omutlu/pub/drange-dram-latency-based-true-random-number-generator_hpca19-talk.pdf

Full System Integration

- Ideally, all of this should be done automatically by the memory controller
  - Implement identification and sampling in firmware
  - Expose some application interface for data retrieval

- For high availability, store unused data in a cache

- Possible interfaces:
  - Memory-mapped configuration status registers
  - I/O instructions in x86 like `IN`, `OUT`
  - New ISA instruction, like Intel's `RDRAND`

Key Results:

Methodology and Evaluation

Testing Environment

- 282 2y-nm LPDDR4 DRAM chips tested with custom infrastructure
  - From "3 major DRAM manufacturers"
- Also tested with 4 DDR3 chips in SoftMC

Evaluation Criteria

- Can RNG cells be found across different DRAM modules?
- Are the sampled values truly random?
- Are the six TRNG properties satisfied?

RNG Cell Distribution

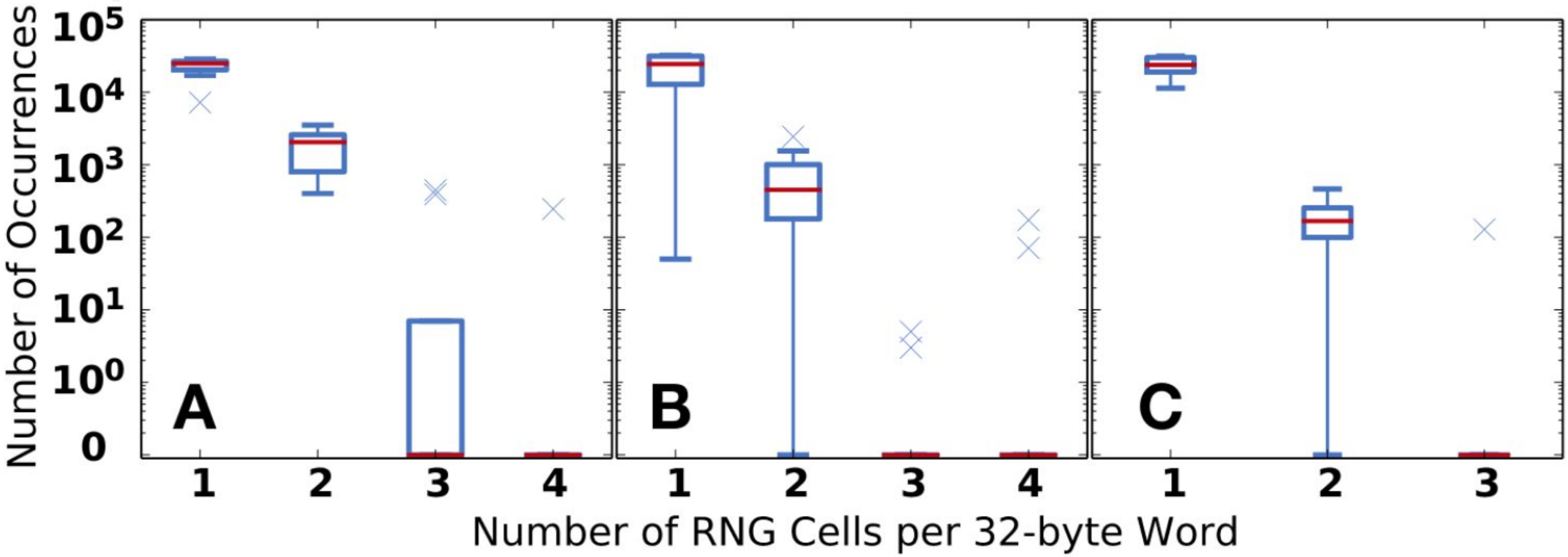


Figure 7: Density of RNG cells in DRAM words per bank.

- RNG cells are widely available ✅

NIST Tests

- Test suite by the US National Institute of Standards and Technology

- Tests for 15 different randomness properties
  - Bit frequencies, longest run etc…

- Result: 15/15 PASSED ✓

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

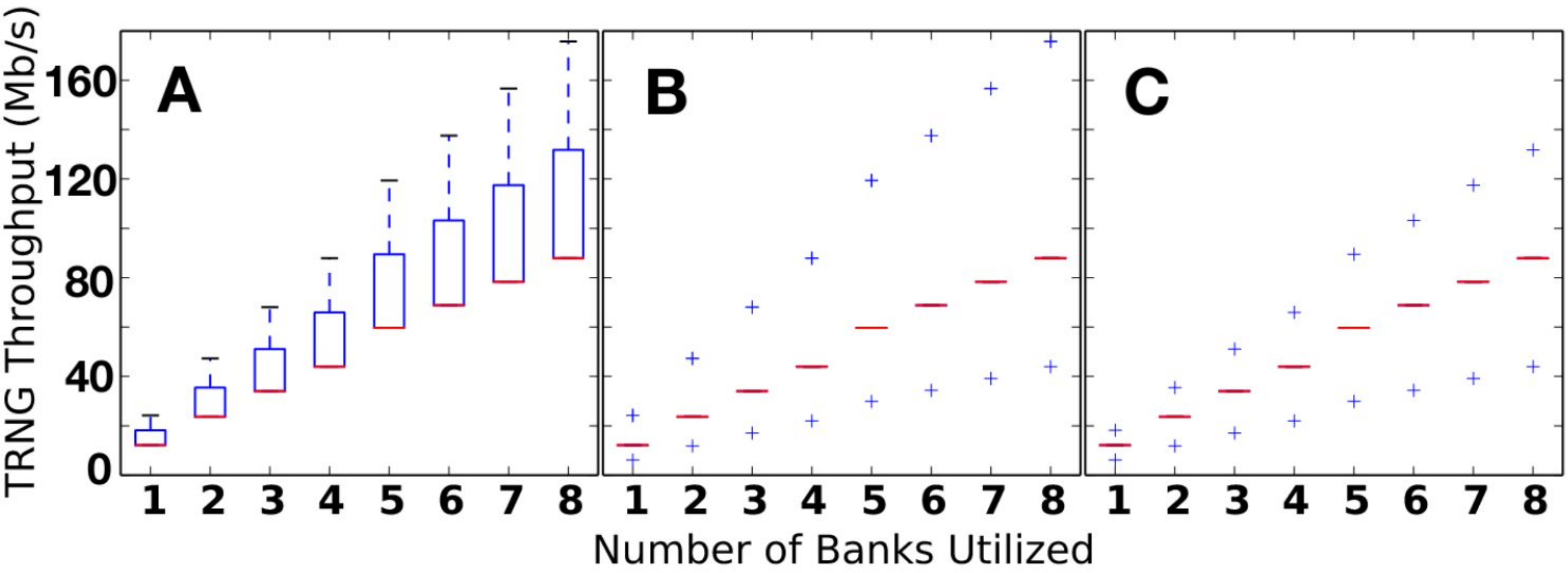
TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
    - Shown by NIST tests
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

Throughput



- Avg. 108.9 Mb/s *per channel*
- With 4 channels: avg 435.7 Mb/s, max 717.4 Mb/s !

Related Works

Proposal	**Year**	**Entropy Source**	**True Random**	**Streaming Capable**	**64-bit TRNG Latency**	**Energy Consumption**	**Peak Throughput**
Pyo+ [116]	2009	Command Schedule	✗	✓	$18\mu s$	N/A	$3.40Mb/s$
Keller+ [65]	2014	Data Retention	✓	✓	$40s$	$6.8mJ/bit$	$0.05Mb/s$
Tehranipoor+ [144]	2016	Startup Values	✓	✗	$> 60ns$ (optimistic)	$> 245.9pJ/bit$ (optimistic)	N/A
Sutar+ [141]	2018	Data Retention	✓	✓	$40s$	$6.8mJ/bit$	$0.05Mb/s$
D-RaNGe	2018	Activation Failures	✓	✓	$100ns < x < 960ns$	$4.4nJ/bit$	$717.4Mb/s$

Table 2: Comparison to previous DRAM-based TRNG proposals.

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

Latency

- Worst case for 64 bits of data: 960 ns
  - 1 bit per word, 1 bank, 1 channel
- With 8 banks and 4 channels: 220 ns
- 4 bits per word: 100ns

Related Works

Proposal	Year	Entropy Source	True Random	Streaming Capable	64-bit TRNG Latency	Energy Consumption	Peak Throughput
Pyo+ [116]	2009	Command Schedule	✗	✓	$18\mu s$	N/A	$3.40 Mb/s$
Keller+ [65]	2014	Data Retention	✓	✓	$40s$	$6.8 mJ/bit$	$0.05 Mb/s$
Tehranipoor+ [144]	2016	Startup Values	✓	✗	$> 60ns$ (optimistic)	$> 245.9 pJ/bit$ (optimistic)	N/A
Sutar+ [141]	2018	Data Retention	✓	✓	$40s$	$6.8 mJ/bit$	$0.05 Mb/s$
D-RaNGe	2018	Activation Failures	✓	✓	$100ns < x < 960ns$	$4.4 nJ/bit$	$717.4 Mb/s$

Table 2: Comparison to previous DRAM-based TRNG proposals.

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

System Interference

- Need to reserve some rows for RNG
  - Only six rows needed per bank
  - Amounts to 0.018% of total storage (2GB)

- Need to occasionally reduce t_{RCD}
  - No significant impact when tested while running SPEC CPU2006 benchmarks

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

Energy Overhead

- Output traces from *Ramulator* analyzed with *DRAMPower*
- Result: 4.4 nJ/bit

Proposal	Year	Entropy Source	True Random	Streaming Capable	64-bit TRNG Latency	Energy Consumption	Peak Throughput
Pyo+ [116]	2009	Command Schedule	✗	✓	$18\mu s$	N/A	$3.40Mb/s$
Keller+ [65]	2014	Data Retention	✓	✓	$40s$	$6.8mJ/bit$	$0.05Mb/s$
Tehranipoor+ [144]	2016	Startup Values	✓	✗	$> 60ns$ (optimistic)	$> 245.9pJ/bit$ (optimistic)	N/A
Sutar+ [141]	2018	Data Retention	✓	✓	$40s$	$6.8mJ/bit$	$0.05Mb/s$
D-RaNGe	2018	Activation Failures	✓	✓	$100ns < x < 960ns$	$4.4nJ/bit$	$717.4Mb/s$

Table 2: Comparison to previous DRAM-based TRNG proposals.

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

Implementation Cost

- Requirements:
  - Adjustable t_{RCD}
    - Possible with some AMD processors
  - Custom memory controller firmware
    - With exposed API

TRNG Key Characteristics

- Recall the six properties for an *effective* TRNG:
  - Fully non-deterministic
  - High throughput
  - Low latency
  - Low system interference
  - Low energy overhead
  - Low implementation cost

Evaluation Criteria

- Can RNG cells be found across different DRAM modules?
  - ✅ Yes, and in fairly high numbers

- Are the sampled values truly random?
  - ✅ Yes, as shown with NIST tests

- Are the six TRNG properties satisfied?
  - ✅ Yes, within reason

Summary

Executive Summary

- **Motivation**
  - True random number generation enables security applications like cryptography
  - Many systems lack TRNG hardware devices, but got DRAM
- **Problem**
  - Existing DRAM-based RNG solution are either not fundamentally non-deterministic or are too slow
- **Goal**
  - A low-latency, high-throughput TRNG based on DRAM
- **Solution**
  - Reduce timing constraints when reading values from DRAM and extract randomness from failing DRAM cells
- **Evaluation**
  - Tested on 282 LPDDR4 DRAM devices
  - Achieves *100 ns* latency and *717.4 Mb/s* throughput

D-RaNGe Summary

- Reducing the time limit between DRAM activate and read (t_{RCD}) can result in incorrect values being read from DRAM cells
- The resulting bitstream of some these cells can be shown to exhibit **true randomness**
- We can exploit these errors to use DRAM as a *high-throughput* (435.7 Mb/s), *low-latency* (100 ns) **True Random Number Generator**

Strengths

Strengths

- Novel idea with good results
  - Much better than related works (best latency/throughput ratio)
- Includes recommendations on how to implement in practice
- Can be useful for real-world applications
- Thoroughly tested with PoC
- Paper well structured and easy to read

Weaknesses

Weaknesses

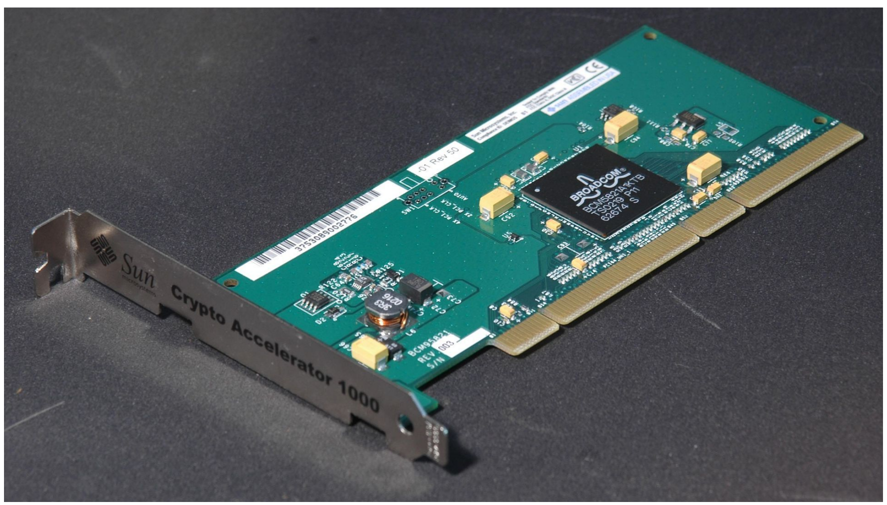
- Not much detail about why randomness occurs
  - If caused by production imperfections, what if production methods improve?
- Underestimation of implementation cost
  - Will it really be that simple to implement?
  - Increased complexity
  - What if the memory controller has no firmware?
- Are 1000 iterations enough for RNG cell identification?
  - The NIST tests were run 1M times
- The possibility of "temperature attacks" is not given much consideration

Thoughts & Ideas

Thoughts & Ideas

- Does it work for SRAM too?
  - Paper only addresses methods based on startup values

- What about a dedicated hardware device based on D-RaNGe?



By Retro-Computing Society of Rhode Island - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7372673

Source:
https://ubld.it/products/truerng-hardware-random-number-generator/

Takeaways

Key Takeaways

- Novel method for extracting randomness from DRAM
- Works in practice
- Pushing limits can have unforeseen consequences

Open Discussion

Discussion Starters

- What constitutes "high-throughput"?
  - 1 Mb/s for flash memory? [1]
- Is it really useful for IoT?
  - Most microcontrollers use flash memory and/or SRAM, not DRAM
- Are attacks like the temperature attack reasonable?
  - What are other possible attacks?
- Will improved production methods make D-RaNGe obsolete?
- Is using DRAM as a TRNG kind of hacky?

[1] Ray, B., & Milenković, A. (2018). True random number generation using read noise of flash memory cells. *IEEE Transactions on Electron Devices*, *65*(3), 963-969.

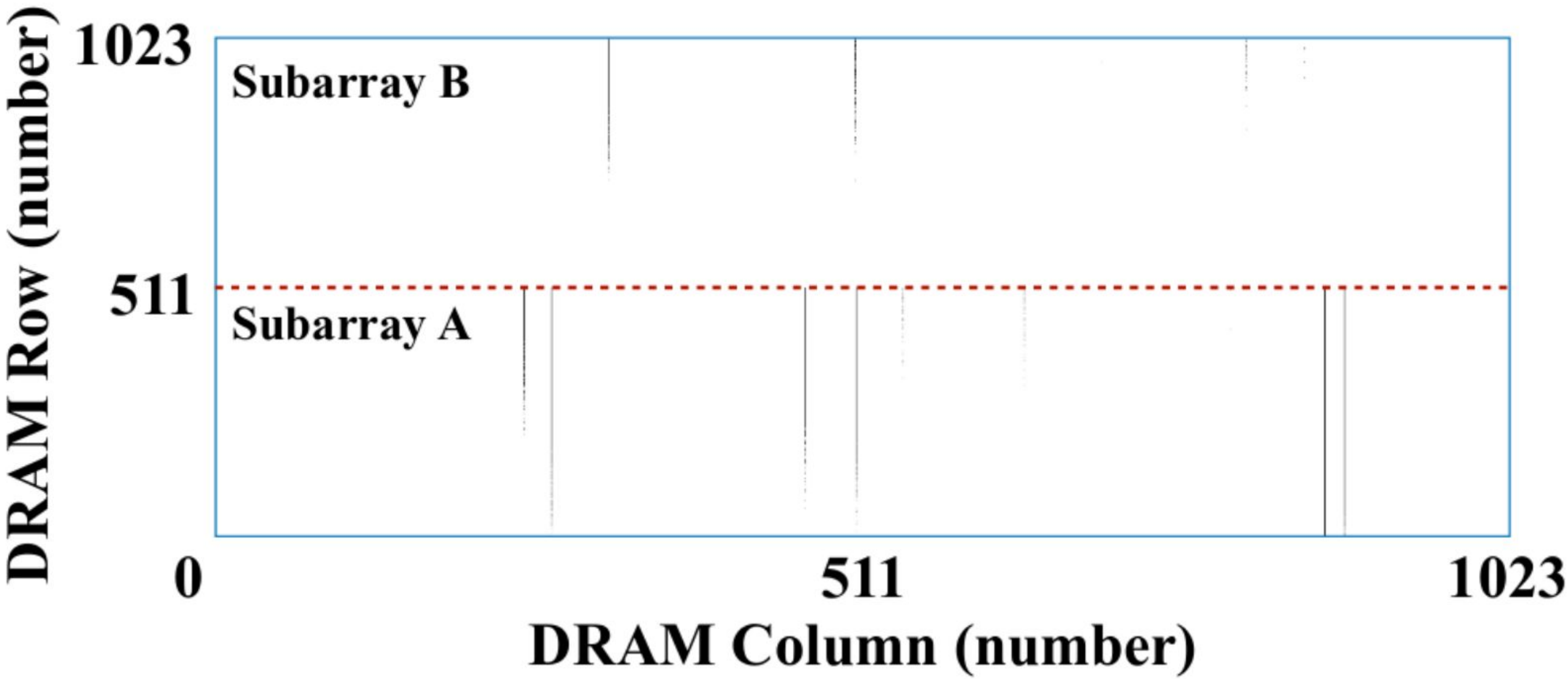
Appendix

Activation Failure Characterization

- What affects the number of activation failures?

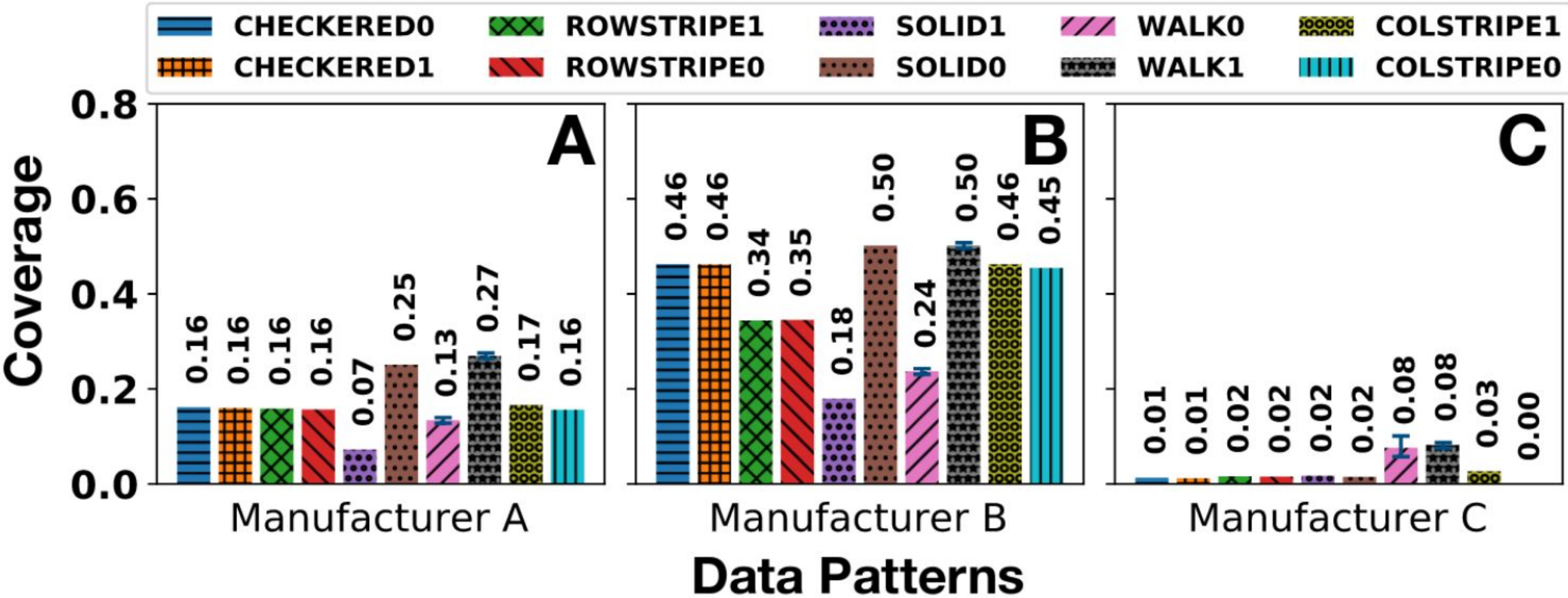
- Aspects to consider:
  - Spatial distribution of failures
  - Data pattern dependence
  - Temperature effects
  - Entropy variation over time

Spatial Distribution of Failures



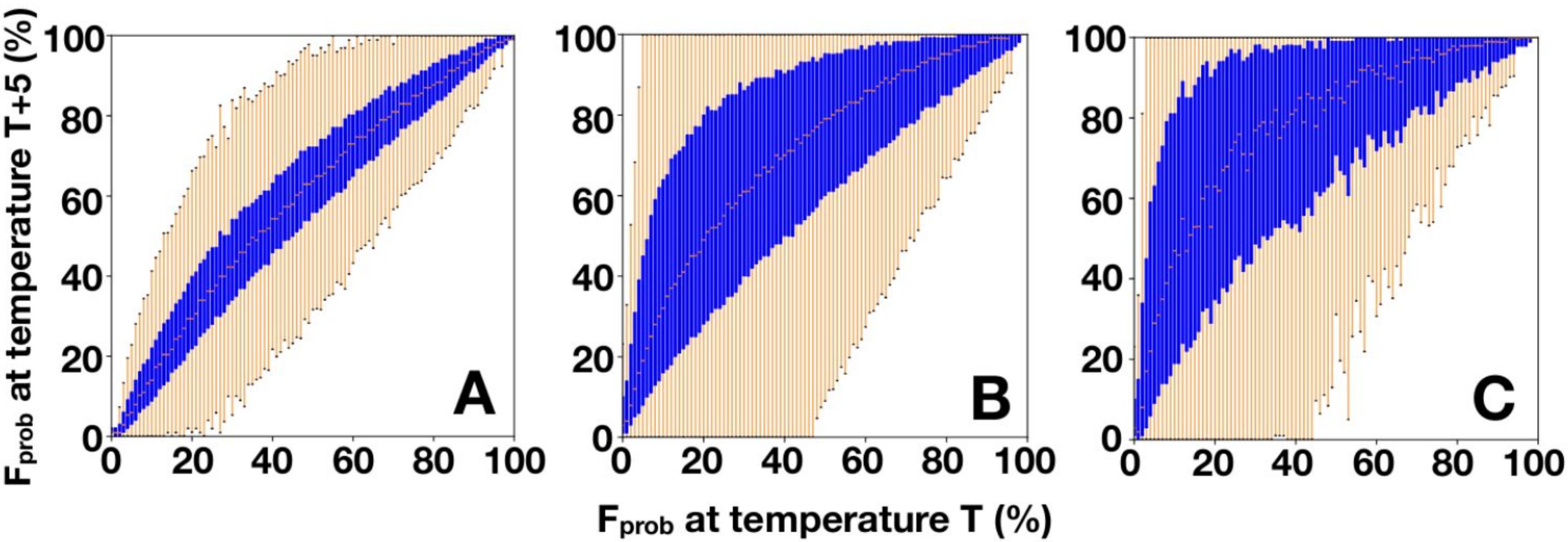
- Observations:
  - Region *and* bitline affects failure rate
  - Differing amounts of failures across subarrays *and* local bitlines

Data Pattern Dependence



- Observations:
  - Data pattern affects entropy extraction
  - Some patterns provides higher coverage

Temperature



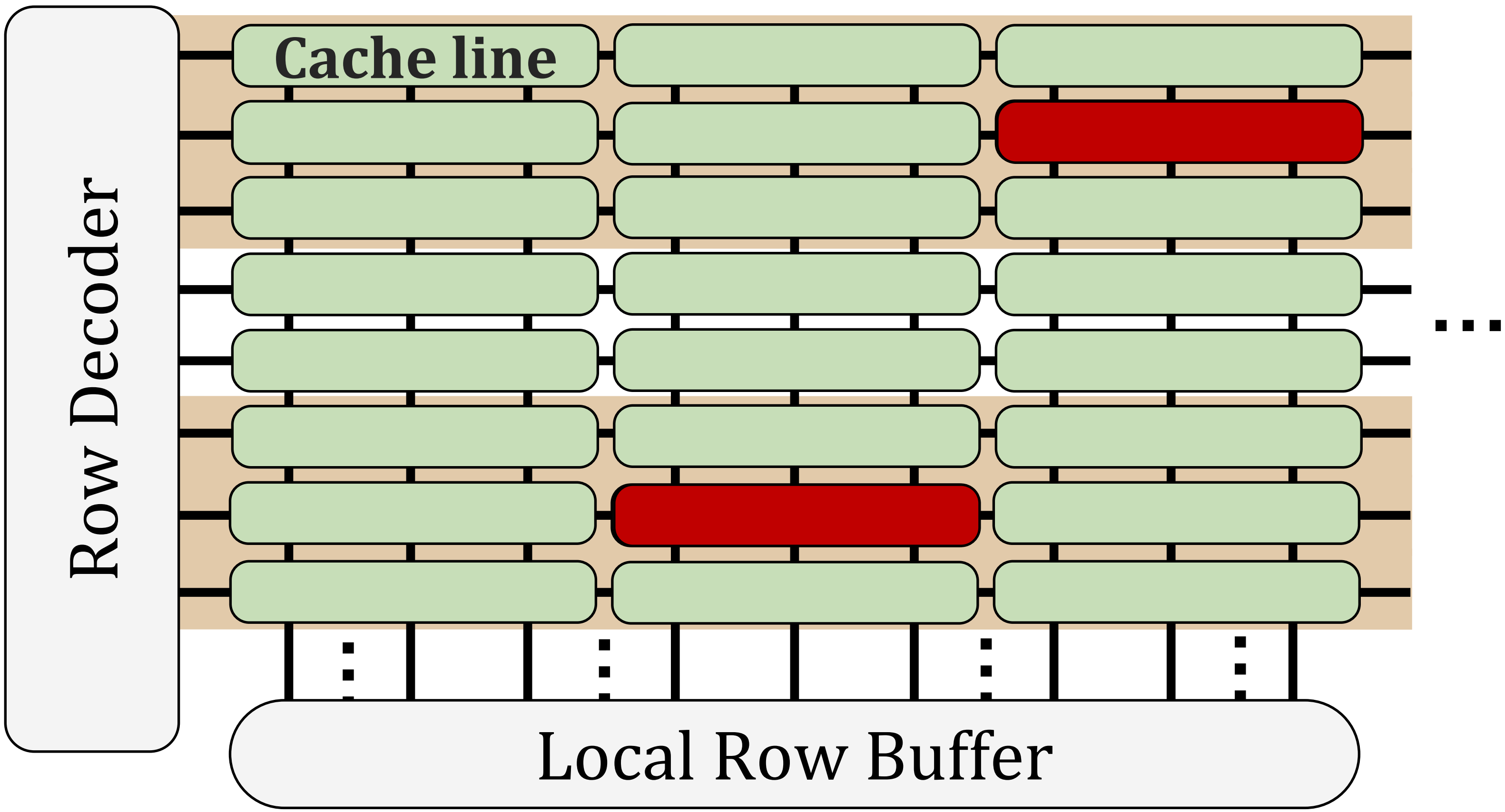
- Temperature affects probability of failure to varying degrees

Entropy Variation over Time

- Stable over a time period of 15 days

Exclusive Access

- We also want exclusive access to these rows to reduce system interference



NIST Tests

NIST Test Name	P-value	Status
monobit	0.675	PASS
frequency_within_block	0.096	PASS
runs	0.501	PASS
longest_run_ones_in_a_block	0.256	PASS
binary_matrix_rank	0.914	PASS
dft	0.424	PASS
non_overlapping_template_matching	>0.999	PASS
overlapping_template_matching	0.624	PASS
maurers_universal	0.999	PASS
linear_complexity	0.663	PASS
serial	0.405	PASS
approximate_entropy	0.735	PASS
cumulative_sums	0.588	PASS
random_excursion	0.200	PASS
random_excursion_variant	0.066	PASS

Table 1: D-RaNGe results with NIST randomness test suite.