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

SAFARI

ETH zürich

kasırga

TOBB ETÜ
University of Economics & Technology

DR-STRaNGe Summary

Motivation:

- Random numbers are important for many applications
- DRAM-based True Random Number Generators (TRNGs) can provide **true random numbers at low cost** on **a wide range** of systems

Problem: There is no end-to-end system design for DRAM-based TRNGs

1. Interference between regular memory requests and RNG requests **significantly slows down** concurrently running applications
2. Unfair prioritization of RNG applications **degrades system fairness**
3. High latency of DRAM-based TRNGs **degrades the RNG applications' performance**

Goal: A **low-cost** and **high-performance** end-to-end system design for DRAM-based TRNGs

DR-STRaNGe: An end-to-end system design for DRAM-based TRNGs that

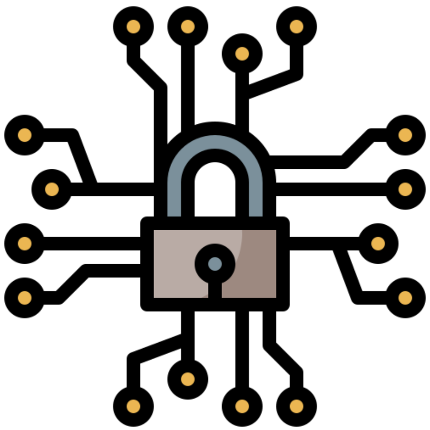
- **Reduces the interference between regular memory requests and RNG requests** by separating them in the memory controller
- **Improves fairness across applications** with an RNG-aware memory request scheduler
- **Hides the large TRNG latencies** using a random number buffering mechanism combined with a new DRAM idleness predictor

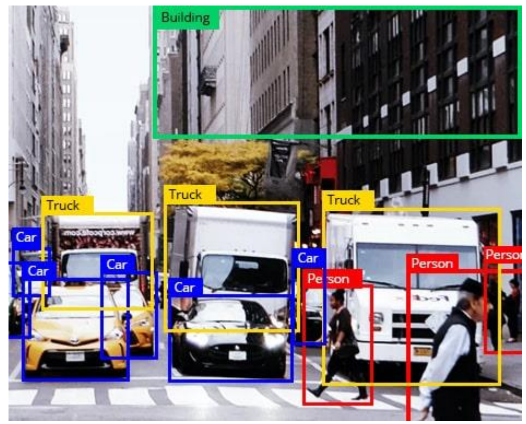
Results: DR-STRaNGe

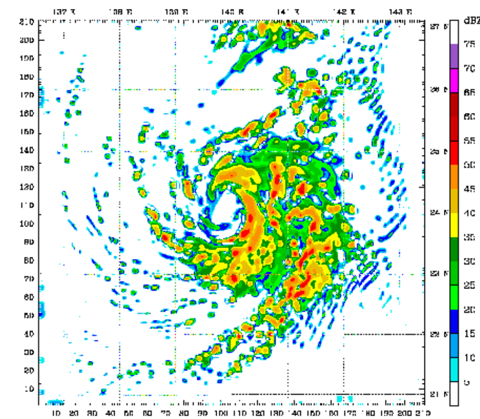
- Improves the average performance of non-RNG (**17.9%**) and RNG (**25.1%**) applications
- Improves the average system fairness (**32.1%**) when generating random numbers at a 5 Gb/s throughput
- Reduces the average energy consumption (**21%**)

True Random Numbers (TRN)

True random numbers are **critical** for many real-world applications





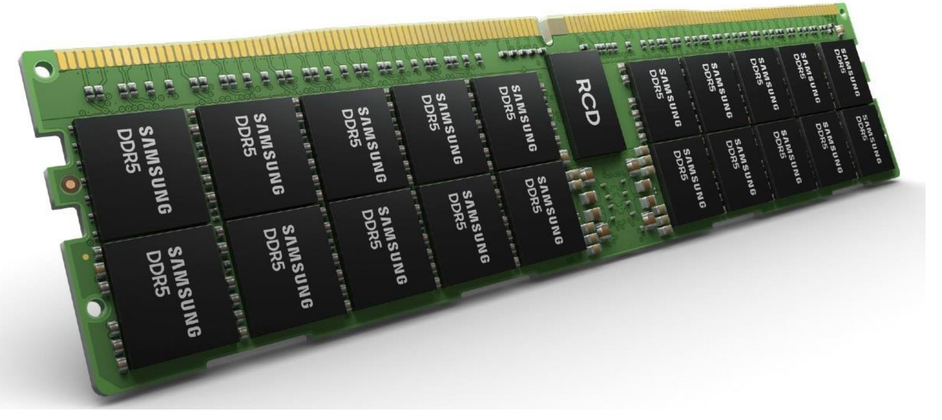


True random numbers are generated by harnessing entropy resulting from **random physical processes**

Dedicated hardware true random number generators (TRNGs) cannot be easily used in all systems

Why DRAM-based TRNGs?

DRAM is **widely available** in most computer systems and can be integrated into mobile and IoT devices as main memory



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

Integration of DRAM-based TRNGs into Real Systems

No prior work provides
an **end-to-end system design**
to enable DRAM-based TRNGs
in **real systems**

Three Key Challenges

1. **RNG Interference**
significantly slows down concurrently-running applications

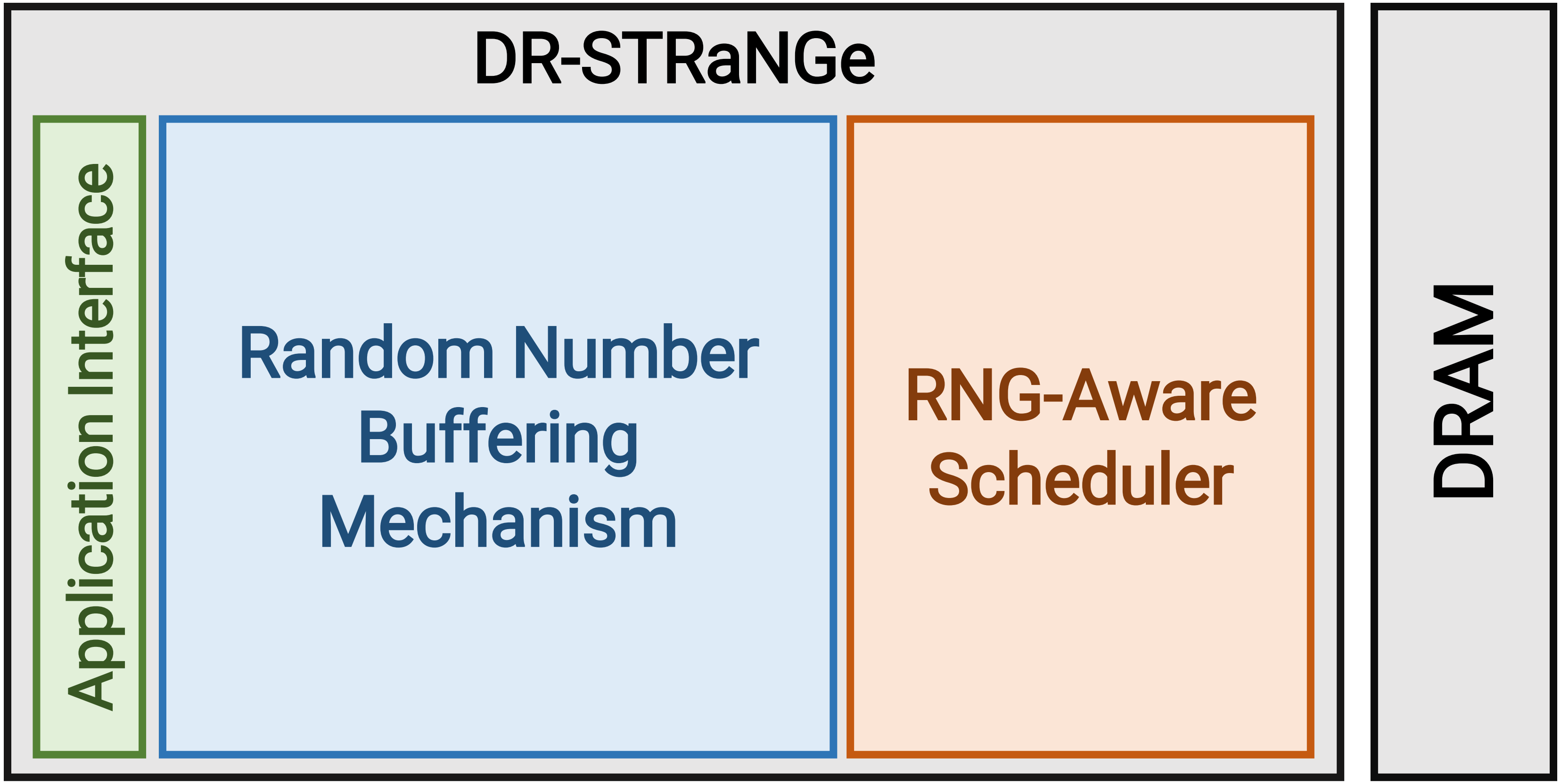
2. **Unfair Prioritization**
degrades overall system fairness

3. **High TRNG Latency**
degrades RNG applications' performance

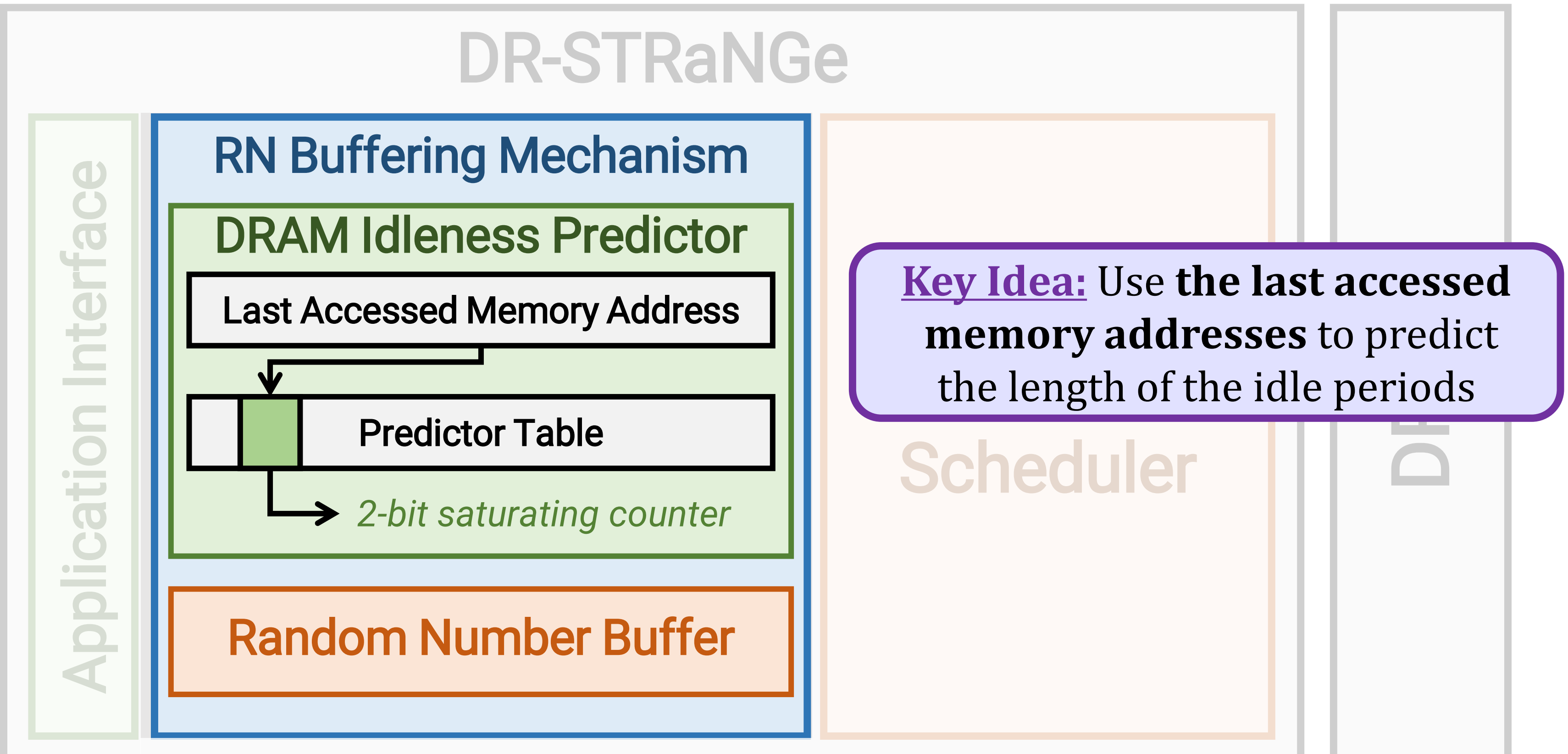
Our Goal

To develop a low-cost and high-performance
end-to-end system design for DRAM-based TRNGs

DR-STRaNGe: Overview



DR-STRaNGe: Overview

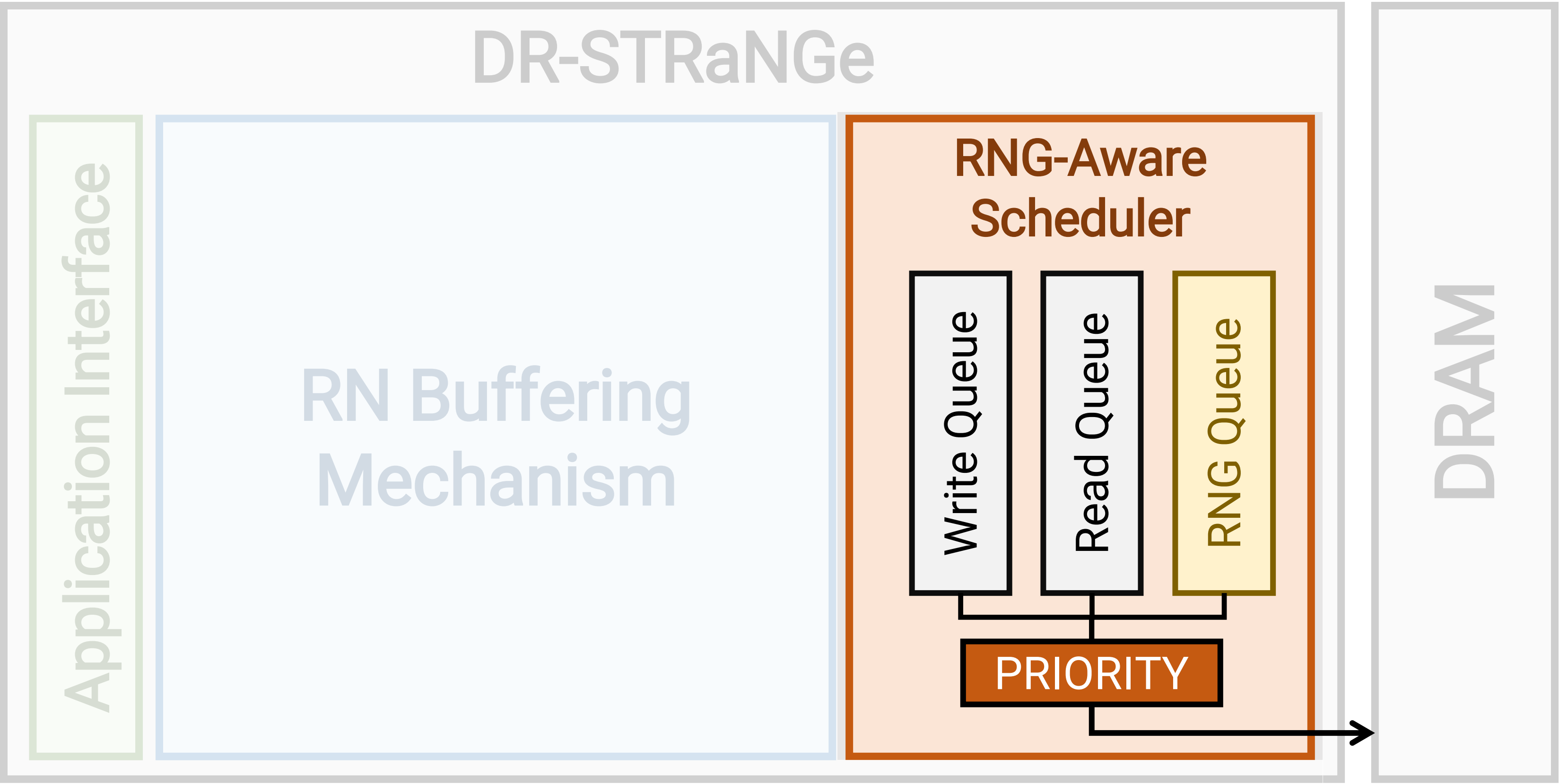


Predicts and utilizes **idle DRAM channels** to generate random numbers

Stores the generated random numbers in a buffer to be served to upcoming RNG requests

Serves RNG requests with low latency

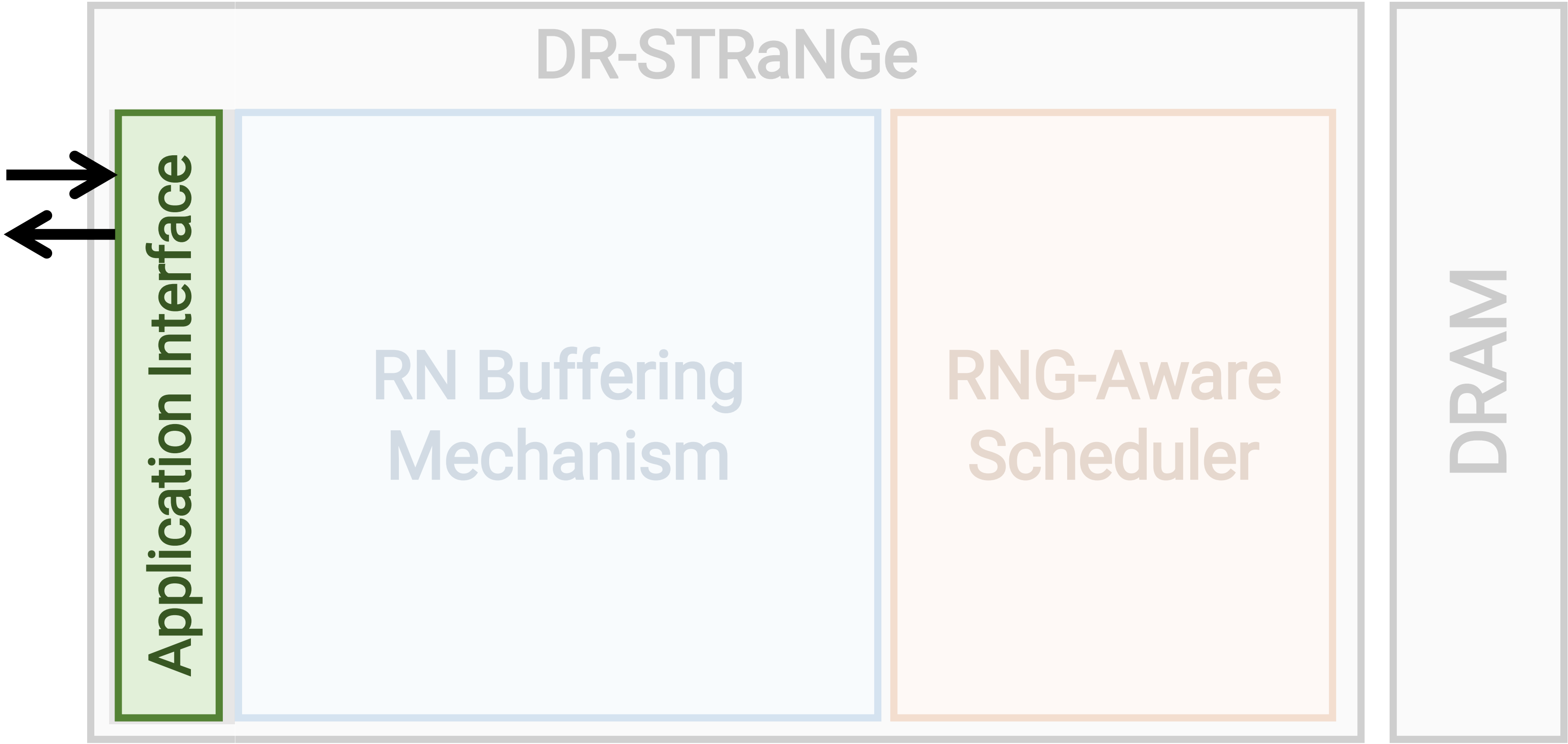
DR-STRaNGe: Overview



Accumulates RNG and regular memory requests in **separate queues**
Schedules requests based on **the priority levels** set **by the operating system**

Reduces the RNG interference and improves system fairness

DR-STRaNGe: Overview



Exposes a secure interface to applications that use random numbers

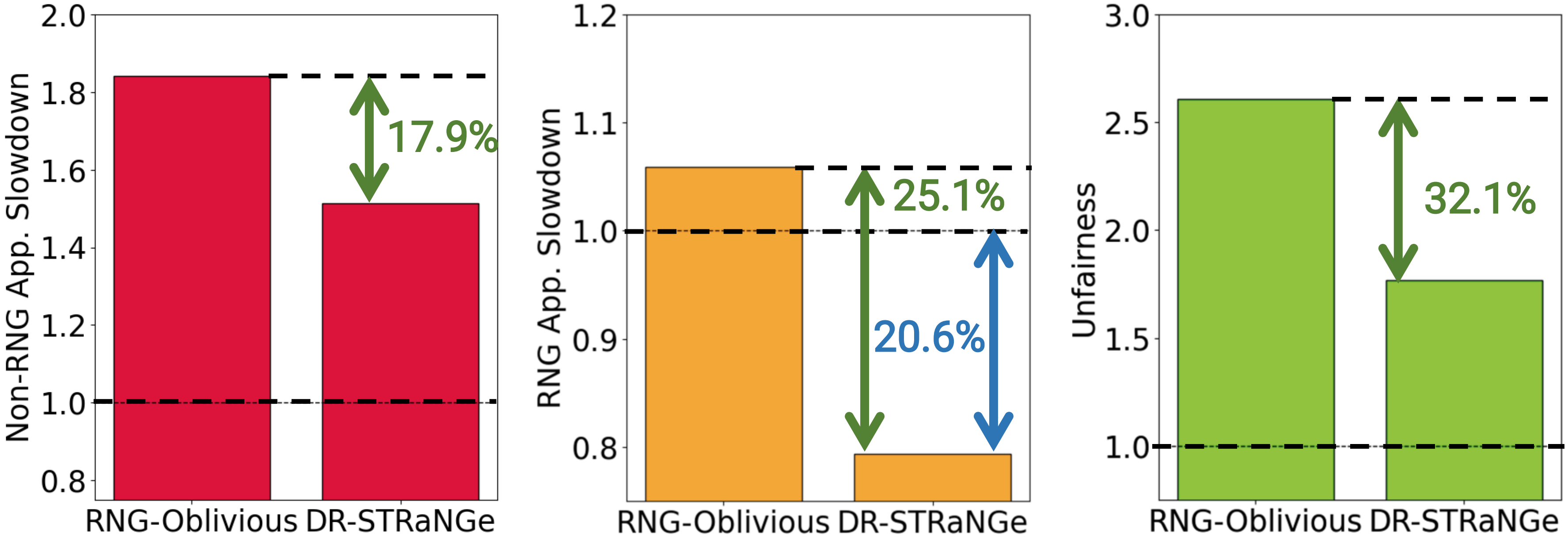
Completes the end-to-end system design and ensures security

Evaluation

- Performance, fairness, energy efficiency, and area overhead
- Cycle-level simulations using **Ramulator** [Kim+, CAL'16] and **DRAMPower** [Chandrasekar+]
- **System configuration:**

Processor	1-,2-,4-,8-,16-core, 4 GHz clock frequency, 3-wide issue, 128-entry instruction window
DRAM	DDR3-1600, 800Mhz bus frequency, 4 channels, 1 rank/channel, 8 banks/rank, 64K rows/bank
Memory Controller	32-entry read/write queues, FR-FCFS with a column cap of 16
DR-STRaNGe	32-entry random read queue, RNG-aware scheduler, 256-entry predictor table/channel, 16-entry random number buffer

Key Results: Performance and Fairness



Improves the performance of both non-RNG (17.9%) and RNG (25.1%) applications compared to the RNG-oblivious baseline design

Improves the performance of RNG applications (20.6%) over the RNG application's single-core performance

Improves the system fairness (32.1%)

Key Results: Scalability, Area, Energy

Performance improvement increases with
the number of memory-intensive applications in the workload mix

Incurs minor area overhead
(0.0022mm^2, 0.00048% of an Intel Cascade Lake CPU Core)

Reduces the average energy consumption (21%)

More in the Paper

- **Security Analysis of DR-STRaNGe**
  - Security of Random Numbers
  - Timing Side-Channel Attacks
  - Covert Channel Attacks
  - Denial of Service Attacks
- **More Results**
  - Impact of DRAM Idleness Predictor
    - Comparison to a Q-learning-based RL agent
  - Impact of the Random Number Buffer
  - Impact of RNG-Aware Scheduling
  - Impact of the Low Utilization Prediction
  - Experiments using QUAC-TRNG [Olgun+, ISCA'21]
  - Results of RNG Applications with Low RNG Demand

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

SAFARI

ETH zürich

kasırga

TOBB ETÜ
University of Economics & Technology