CODIC

A Low-cost Substrate for Enabling Custom in-DRAM Functionalities and Optimizations

Lois Orosa

Yaohua Wang Mohammad Sadrosadati Jeremie Kim

Mines Patel Ivan Puddu Haocong Luo Kaveh Razavi

Juan Gómez-Luna Hasan Hassan Nika Mansouri Ghiasi

Saugata Ghose Onur Mutlu

SAFARI

ISCA 2021

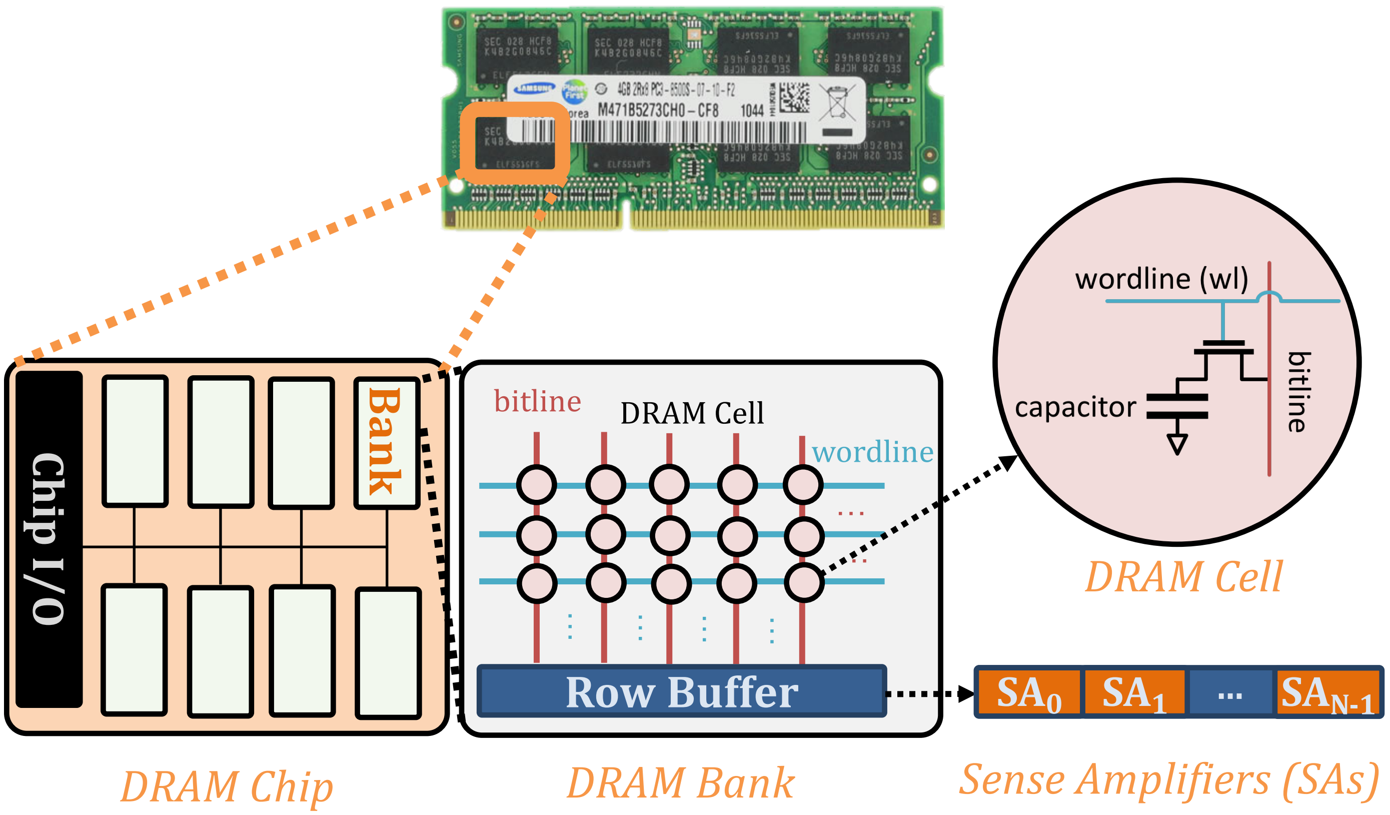
Executive Summary

- **Problem:** The timing of internal DRAM operations is fixed, which hinders the potential of DRAM for implementing new functionalities and optimizations
- **Goal:** Provide **control** over DRAM internal circuit **timings to enable new functionalities and optimizations**
- **CODIC substrate:**
  - **Key idea:** Enable fine-grained control of fundamental DRAM internal circuit timings that control key basic components in the DRAM array (e.g., wordline, sense amplifier, precharge logic)
  - We propose and evaluate two CODIC variants: 1) **CODIC-sig** generates signature values, and 2) **CODIC-det** generates deterministic values
  - CODIC variants have low latency (i.e, ~25ns)
- **CODIC-based Physical Unclonable Functions (PUFs):**
  - A **PUF** generates **signatures unique to a device** due to the **unique physical variations** of the device
  - **Key idea:** Use CODIC-sig to generate unique signatures that can uniquely identify a DRAM device
  - **2x faster** than the best **state-of-the-art** DRAM PUF
- **CODIC-based Cold Boot Attack Prevention Mechanism:**
  - **Cold Boot Attack:** The attacker physically removes the DRAM module from the victim system and places it in a system under their control to extract secret information
  - **Key idea:** Destroy all data at power-on using CODIC
  - Does **not incur any latency or energy at runtime**, and it is **2.0x lower latency and 1.7x lower energy** than the best state-of-the-art mechanisms during DRAM power-on
- **Conclusion:**
  - CODIC can be used for implementing very **efficient security applications**
  - CODIC can enable **new DRAM functionalities, and reliability, performance, and energy optimizations**

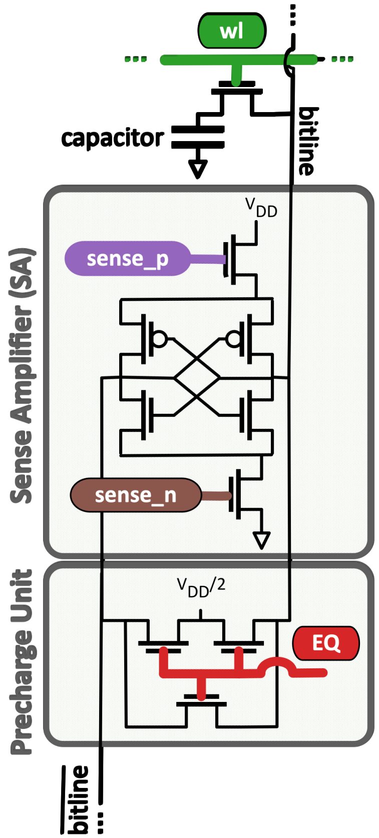
Outline

Outline

DRAM Organization



Internal DRAM Signals



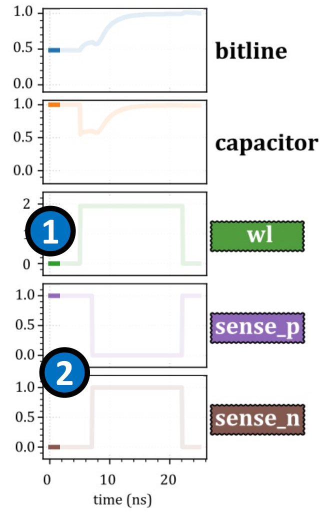
- **wl** controls the access transistor that connects the cell capacitor to the bitline
- **sense_p** controls the PMOS amplifier in the SA
- **sense_n** controls the NMOS amplifier in the SA
- **EQ** controls the precharge unit that sets the bitline to $V_{dd}/2$

SAFARI

DRAM Operation - Activate

DRAM controller can issue **4** main **memory commands**

1) **ACTIVATE**: Activates the DRAM row containing the data



1 wl is raised to share the charge between the cell and the bitline

2 sense_n and sense_p are raised to sense and amplify the small charge in the cell

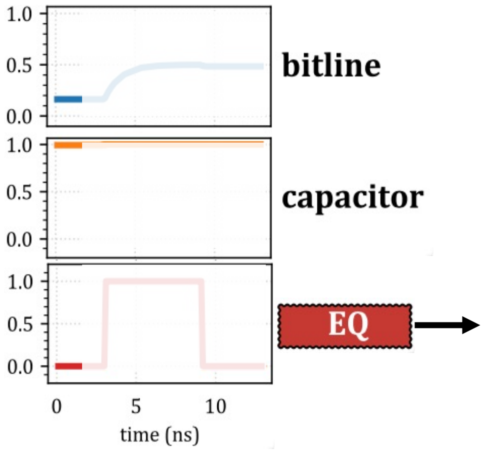
2) **READ**: Reads a column of data from the RB

3) **WRITE**: Writes a column of data into DRAM

DRAM Operation - Precharge

DRAM controller can issue **4** main **memory commands**

4) PRECHARGE: Prepares all bitlines for a subsequent ACTIVATE command to a different row



is raised for setting the bitline to the precharge voltage level ($V_{DD}/2$)

Outline

Limitations of Recent Works

- Many recent works change timings between DRAM commands
  - Have fixed DRAM internal circuit timings

- **Limitations** of fixed DRAM internal circuit timings

  1) Chosen at design time and cannot be modified

  2) Conservative internal circuit timings to ensure reliable operation

  3) Memory controller does not have any knowledge or control over the internal implementations of DRAM commands

Controlling DRAM Circuits

- Our work explores the **potential** of controlling internal DRAM circuit timings

1) Enables more aggressive performance, reliability, and energy optimizations

2) Enables new functionalities

3) May open new areas of research

Goals

Enable new and enhance existing DRAM commands and optimizations

by providing a low-cost substrate that enables fine-grained control over DRAM internal circuit timings

Design new security mechanisms that have strong security and high performance

by using the CODIC substrate

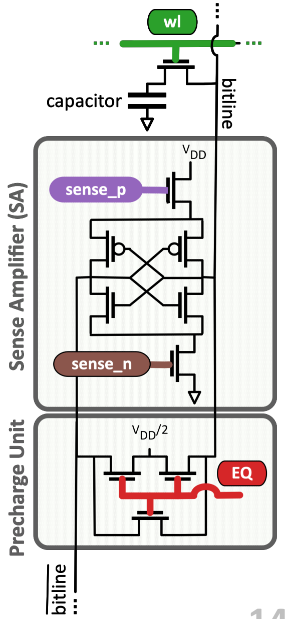
Outline

Overview

- **CODIC** substrate enables greater control over DRAM internal circuit timings

- **CODIC** is an efficient and low-cost way to enable new functionalities and optimizations in DRAM

- **CODIC** controls *four* key signals that orchestrate DRAM internal circuit timings
  - **wordline (wl)**: Connects DRAM cells to bitlines
  - **sense_p** and **sense_n**: Trigger sense amplifiers
  - **EQ**: Triggers the logic that prepares a DRAM bank for the next access



CODIC Variants

- CODIC can trigger and disable internal DRAM signals in a fine-grained manner
  - Time steps of 1ns, within a time window of 25ns

- Large number of possible timing combinations that allow 300^4 different CODIC variants

- We demonstrate and evaluate two CODIC variants
  - **CODIC-sig:** Generates digital signatures that depend on process variation
  - **CODIC-det:** Generates deterministic values

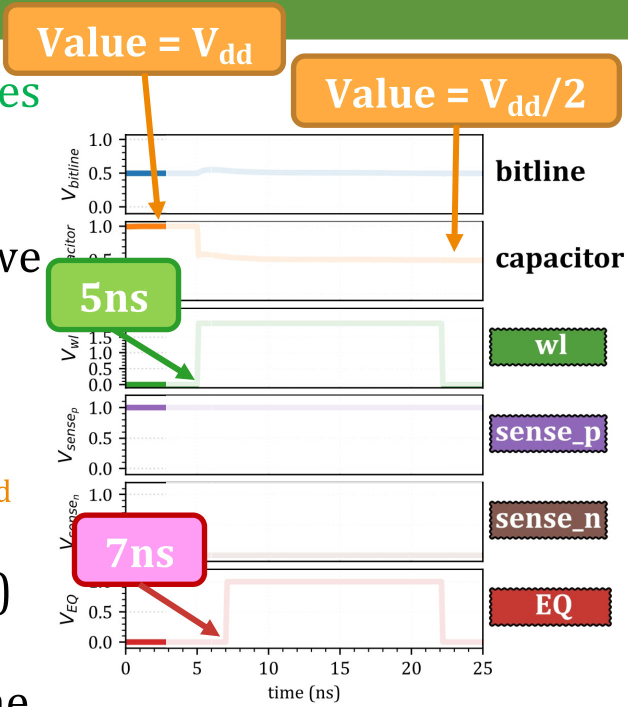
CODIC-sig

- CODIC-sig generates digital signatures that depend on process variation

- **Key idea:** Amplify a DRAM cell that we set to the precharge voltage ($V_{dd}/2$)

- **Mechanism:**
  1) Capacitor is initially set to 0 or V_{dd}
  2) Raise the wordline signal (wl)
  3) Raise the precharge signal (EQ)

- **Result:** the final value of the cell is the precharge voltage ($V_{dd}/2$)



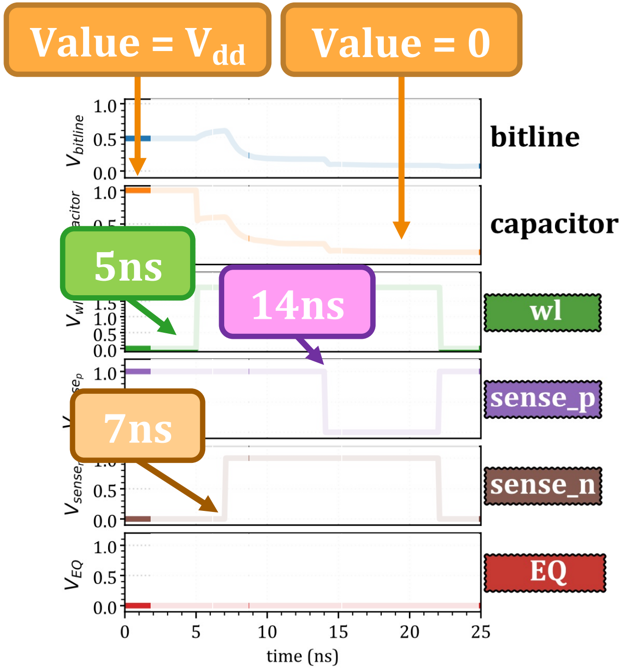
- In the **next activate command,** the SA generates a **signature value** that depends on truly-random process variation

CODIC-det

- **CODIC-det** generates deterministic values

- **Key idea:** Activate **sense_n** and **sense_p** with a delay between them

- **Mechanism:**
  - Capacitor is initially set to 0 or V_{dd}
  - Raise the wordline **wl**
  - **To generate 0**, raise:
    1) **sense_n**
    2) **sense_p**
  - **To generate 1**, raise:
    1) **sense_p**
    2) **sense_n**



- **Result:** CODIC-det generates a 0 or 1 value deterministically

SAFARI

Also in the Paper…

- An optimization of CODIC-sig to achieve lower latency [1,2]

- A new CODIC variant (CODIC-sigsa) that generates digital signatures without destroying the content of DRAM [2]

- Details about the hardware implementation and the low area overhead (~1% per mat) [1,2]

- Details about the minimal changes to the DDRx interface [1,2]

[1] Orosa+, "CODIC: Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations", ISCA 2021
[2] Orosa+, "CODIC: Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations", arXiv 2021

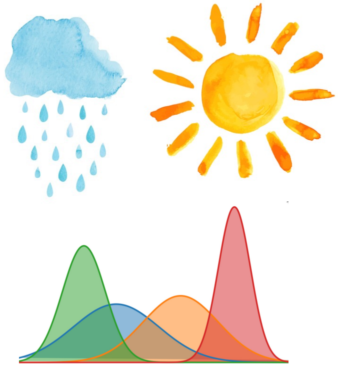
Outline

Applications of CODIC

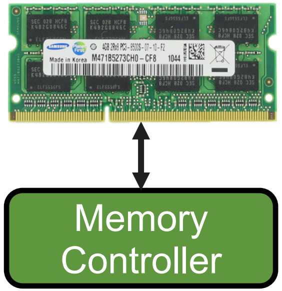
CODIC can be used in **many applications**

Security

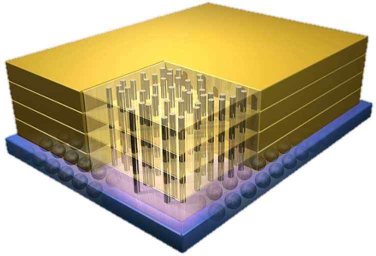
Custom optimizations



Accurate DRAM Characterization



Processing-In-Memory (PIM)



We use **CODIC** to design two new applications in the security domain

1) A CODIC-based Physical Unclonable Function (PUF)
2) A CODIC-based Cold Boot Attack Prevention Mechanism

Application 1: PUF

- A Physical Unclonable Function (**PUF)** generates signatures unique to a device due to the unique physical variations of the device
- PUFs are typically used to authenticate or identify a device
- A PUF maps a unique input (i.e., challenge) to a unique output (i.e., response)

- **Limitations** of state-of-the-art DRAM PUFs:
    1) Long evaluation times
    2) Require heavy filtering mechanisms to deal with inherently noisy DRAM PUF responses
    3) Responses to the same challenge exhibit high variation with temperature changes
    4) Data dependency on data stored in the evaluated DRAM region

CODIC-sig PUF

- **CODIC-sig** can generate signature values in DRAM that can be used as a PUF
  - By amplifying a DRAM cell that we set to the precharge voltage

- **Characteristics** of **CODIC-sig PUF:**
  1) CODIC-sig has short evaluation latency
  2) Repeatable PUF responses without relying on a filtering mechanism
  3) CODIC-sig is resilient to temperature changes, i.e., changing the temperature does not influence much the repeatability of the PUF responses
  4) CODIG-sig responses do not depend on the content of DRAM

Application 2: Cold Boot Attacks

- A cold boot attack is a physical attack on DRAM that involves hot-swapping a DRAM chip and reading out the contents of the DRAM chip on another system

- The attacker first disables the power and then transfers the DRAM to another system

- Cold boot attacks are possible because the data stored in DRAM is not immediately lost when the chip is powered-off
  - Data in DRAM is stored in capacitors, and the data can remain in the cells long enough for it to be stolen

- This data retention effect can be even more significant at low temperatures

CODIC Self-Destruction

Observation

It is possible to protect from cold boot attacks by deleting the entire memory content during DRAM power-on

Key Idea

Self-destruction: A low cost in-DRAM mechanism based on CODIC that destroys all DRAM content during DRAM power-on

How Does It Work?

- Self-destruction is implemented completely within DRAM
  - It does not require the intervention of the memory controller
- Can use CODIC-sig or CODIC-det variants to destroy data
- Uses a dedicated circuit within DRAM that issues CODIC commands back-to-back to all DRAM rows
  - Parallelizes commands across banks and enforces JEDEC* standard timings
- Destroys the entire memory content at power-on
- During self-destruction, the DRAM chip does not accept any memory commands to ensure the atomicity of the process
- Does not introduce any performance or energy overheads at runtime

* JEDEC Solid State Technology Assn., *"JESD79-3F: DDR3 SDRAM Standard"*, July 2012

Outline

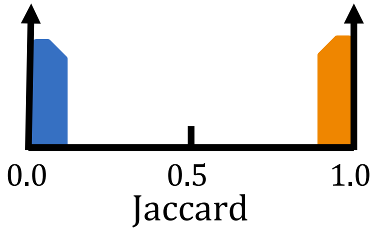
Experimental Setup

- We evaluate **quality** and **performance** of CODIC-sig PUF
- Evaluation of **72 real DDR3L (low power)** DRAM chips and **64 real DDR3** DRAM chips
- Customized memory controller built with SoftMC [Hasan+, HPCA 2017][https://github.com/CMU-SAFARI/SoftMC]
- We cannot implement CODIC-sig in commodity DRAM chips
  - Requires changing the timings of **EQ** and **wordline** signals to set the cell to $V_{dd}/2$
- We emulate CODIC-sig behavior in real DRAM chips
  1) We set the cell to $V_{dd}/2$ via not refreshing DRAM rows for 48h
  2) We activate the cell to generate the signature value

Metrics

- We use **Jaccard** indices to measure
  - **Similarity** of two responses from the same memory segment (**Intra-Jaccard**)
  - **Uniqueness** of two responses from different memory segments (**Inter-Jaccard**)

- An **ideal PUF** should have
  1) Intra-Jaccard indices close to 1
  2) Inter-Jaccard indices close to 0



Comparison Points

- **CODIC-sig PUF**
  - No filter: Robust responses in most cases (99.72%)
  - Filter: Always robust responses by using 5 PUF responses to compose the final response

- **DRAM Latency PUF** generates responses by reducing the activation latency (tRCD = 2.5ns)
  - No filter: *Non-robust* responses
  - Filter: 100 PUF responses to compose the final response

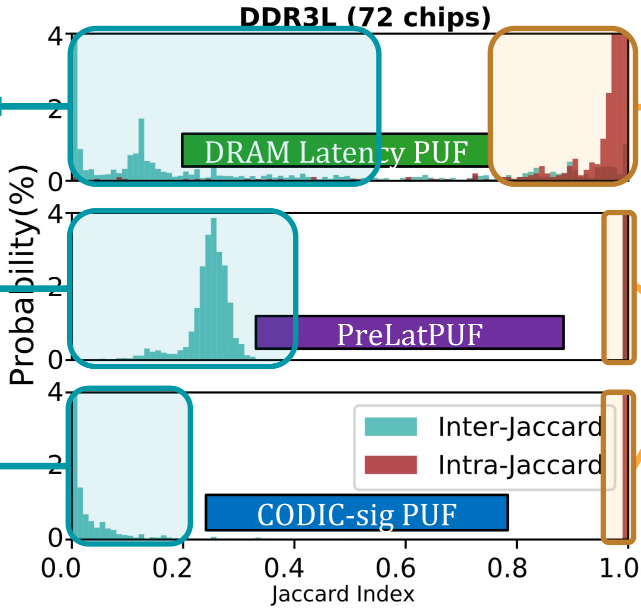
- **PreLatPUF** generates responses by reducing the precharge latency (tRP = 2.5ns)
  - No filter: Robust responses in most cases (96.92%)
  - Filter: Always robust responses by using 5 PUF responses to compose the final response

Results - Jaccard Index

Disperse Inter-Jaccard index ✗

Inter-Jaccard index far from 0 ✗

Inter-Jaccard index close to 0 ✓



Intra-Jaccard index close to 1 ✓

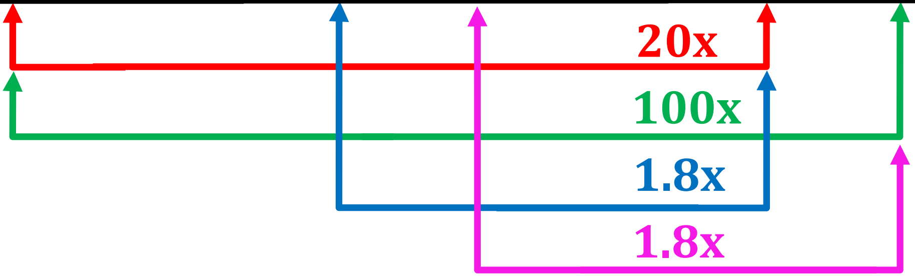
Intra-Jaccard index is 1.0 with more than 99% probability ✓

CODIC-sig PUF is very effective at providing very similar responses to the same challenge

CODIC-sig maintains uniqueness across responses to different challenges

Results - PUF Evaluation Time

DRAM Latency PUF	PreLatPUF w/(w/o) filter	CODIC-sig PUF w/(w/o) filter
88.2 ms	7.95 (1.59) ms	4.41 (0.88) ms



CODIC-sig PUF is 1.8x faster than the best state-of-the-art DRAM PUF

SAFARI

Also in the Paper...

- We evaluate 64 DDR3 chips that show similar quality results as DDR3L (low power) chips
- CODIC-sig PUF is resilient to temperature changes (more so than other PUFs)
- CODIC-sig PUF passes all 15 NIST randomness tests
- CODIC-sig PUF is very robust to aging

Outline

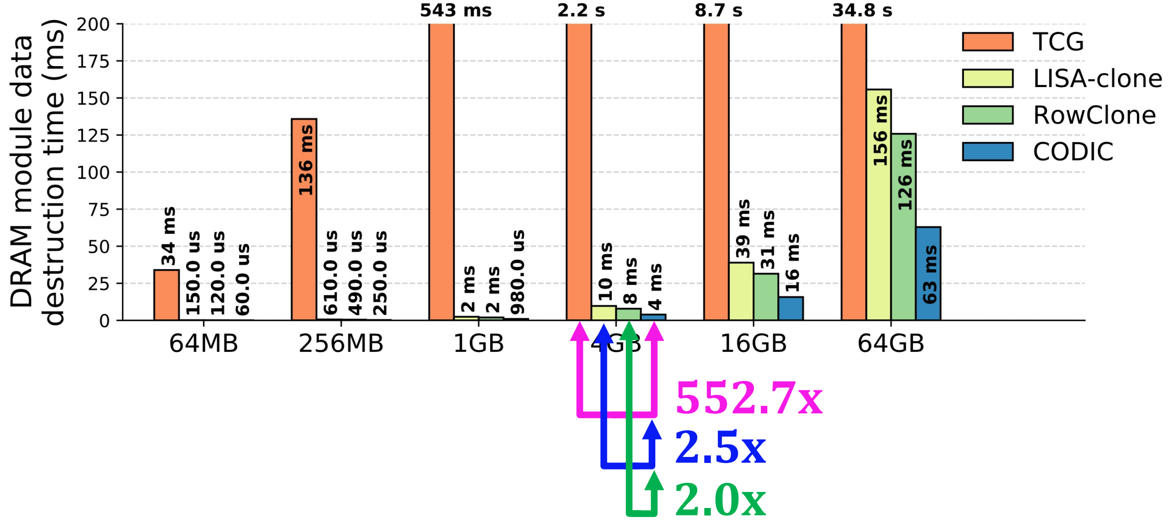
Experimental Setup

- We evaluate latency overheads of **self-destruction** at power-on, implemented with
  - **CODIC:** CODIC-det or CODIC-sig variants
  - **TCG:** Issues regular write commands from the memory controller, as proposed by the Trusted Computing Group (TCG)
  - **LISA-clone:** Copies bulk data in-DRAM [Chang+, HPCA'16]
  - **RowClone:** Copies bulk data in-DRAM [Seshadri, MICRO'13]

- We evaluate power, performance, and area overhead of **CODIC self-destruction** compared to **full main memory encryption**
  - **ChaCha-8**
  - **AES-128**

- Ramulator DRAM simulator [Kim+, CAL'15] [https://github.com/CMU-SAFARI/ramulator]
- DRAMPower [Chandrasekar+][http://www.drampower.info]

DRAM Destruction Time



CODIC destroys the entire content of DRAM at least 2x faster than the state-of-the-art

CODIC vs. Encryption

- Full main memory encryption provides strong security guarantees at the cost of additional energy consumption and complexity
- Comparison points: **ChaCha-8** and **AES-128** in a Intel Atom N280

Overhead	CODIC Self-d	ChaCha-8	AES-128
Performance	~0%	~0%	~0%
Power	~0%	~17%	~12%
Processor Area	~0.0%	~0.9%	~1.3%
DRAM Area	~1.1%	~0.0%	~0.0%

CODIC has no performance and power overhead at runtime

CODIC has no processor area overhead, and 1.1% DRAM area overhead

Also in the Paper…

- We propose and evaluate another security application that accelerates secure deallocation with CODIC [2]

[2] Orosa+, "CODIC: Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations", arXiv 2021

Outline

Executive Summary

- **Problem:** The timing of internal DRAM operations is fixed, which hinders the potential of DRAM for implementing new functionalities and optimizations
- **Goal:** Provide **control** over DRAM internal circuit **timings to enable new functionalities and optimizations**
- **CODIC substrate:**
  - **Key idea:** Enable fine-grained control of fundamental DRAM internal circuit timings that control key basic components in the DRAM array (e.g., wordline, sense amplifier, precharge logic)
  - We propose and evaluate two CODIC variants: 1) **CODIC-sig** generates signature values, and 2) **CODIC-det** generates deterministic values
  - CODIC variants have low latency (i.e, ~25ns)
- **CODIC-based Physical Unclonable Functions (PUFs):**
  - A **PUF** generates **signatures unique to a device** due to the **unique physical variations** of the device
  - **Key idea:** Use CODIC-sig to generate unique signatures that can uniquely identify a DRAM device
  - **2x faster** than the best **state-of-the-art** DRAM PUF
- **CODIC-based Cold Boot Attack Prevention Mechanism:**
  - **Cold Boot Attack:** The attacker physically removes the DRAM module from the victim system and places it in a system under their control to extract secret information
  - **Key idea:** Destroy all data at power-on using CODIC
  - Does **not incur any latency or energy at runtime**, and it is **2.0x lower latency and 1.7x lower energy** than the best state-of-the-art mechanisms during DRAM power-on
- **Conclusion:**
  - CODIC can be used for implementing very **efficient security applications**
  - CODIC can enable **new DRAM functionalities, and reliability, performance, and energy optimizations**

CODIC

A Low-cost Substrate for Enabling Custom in-DRAM Functionalities and Optimizations

Lois Orosa

Yaohua Wang Mohammad Sadrosadati Jeremie Kim

Mines Patel Ivan Puddu Haocong Luo Kaveh Razavi

Juan Gómez-Luna Hasan Hassan Nika Mansouri Ghiasi

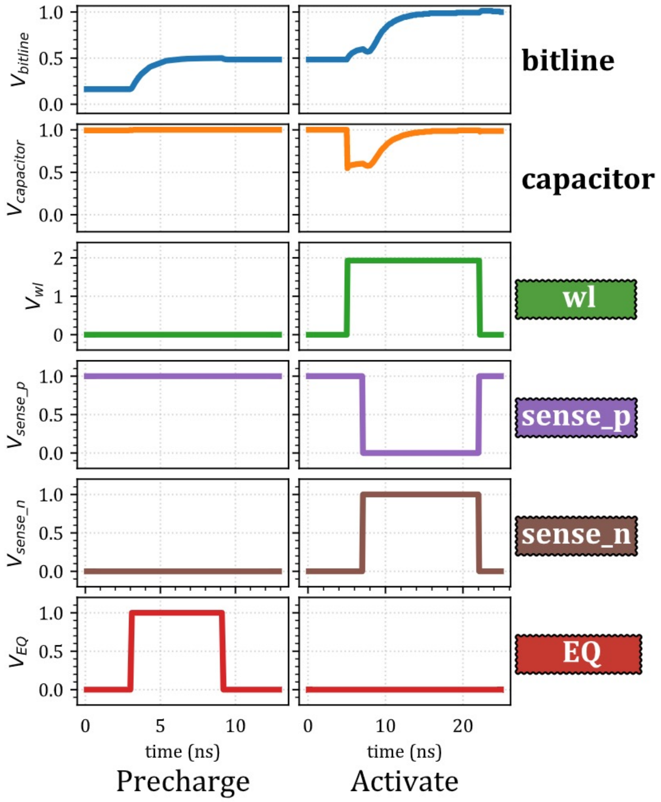
Saugata Ghose Onur Mutlu

SAFARI

ISCA 2021

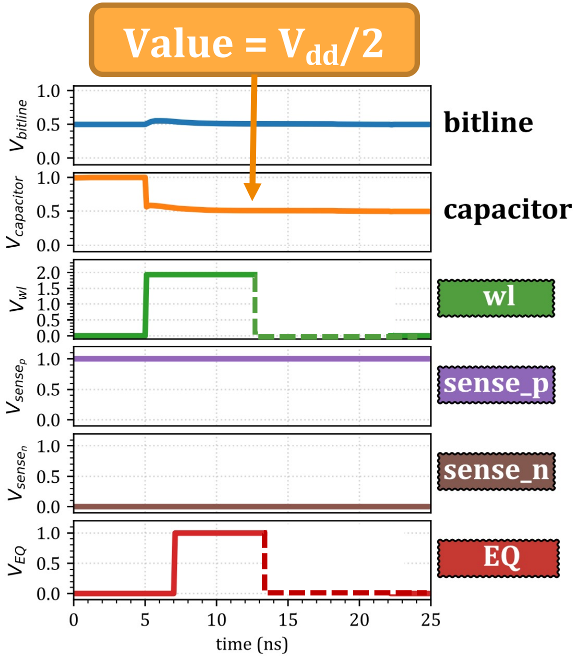
Backup

Precharge & Activate



Optimization of CODIC-sig

- **Observation:** that CODIC-sig can set the voltage of the DRAM capacitor to $V_{dd}/2$ very quickly
- **Key idea:** terminate the **wl** and **EQ** signals earlier to reduce the latency of the command, without sacrificing reliability
- **Result:** same result as CODIC-sig, with lower latency



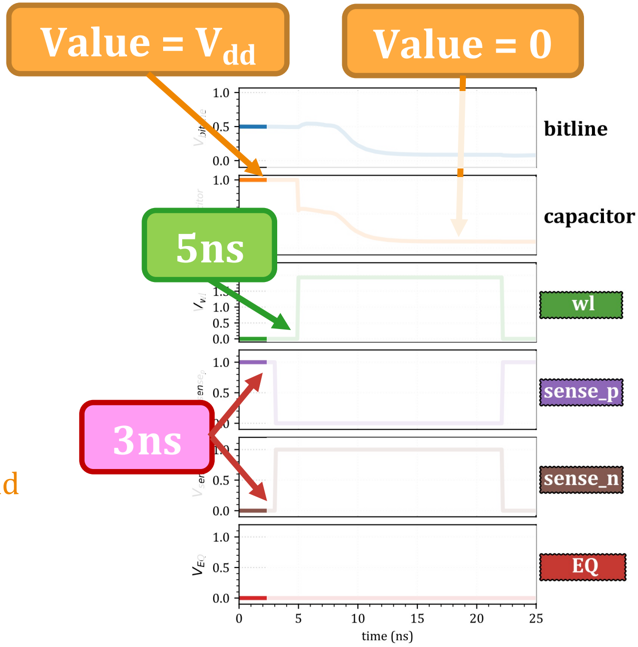
CODIC-sigsa

- **CODIC-sigsa** generates digital signatures that depend on process variation

- **Key idea:** Sense and amplify a precharge bitline ($V_{dd}/2$) that is not connected to the DRAM cell

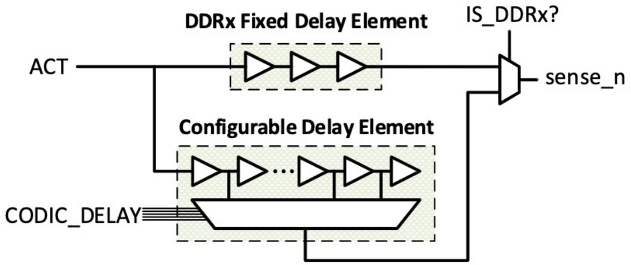
- **Mechanism:**
  1) Capacitor is initially set to 0 or V_{dd}
  2) Raise **sense_n** and **sense_p**
  3) Raise **wl**

- **Result:** the final value a **signature value** that depends on truly-random process variation



Hardware Implementation

- Configurable delay To implement the fine-grained configurable control of the timing interval between DRAM internal circuit control signals



- Delay element using the 22nm PTM transistor model in SPICE
- Each buffer stage has a propagation delay of approximately 1ns at the output of the multiplexer
- Total DRAM hardware area overhead is 1.1%

Changes to DDRx Interface

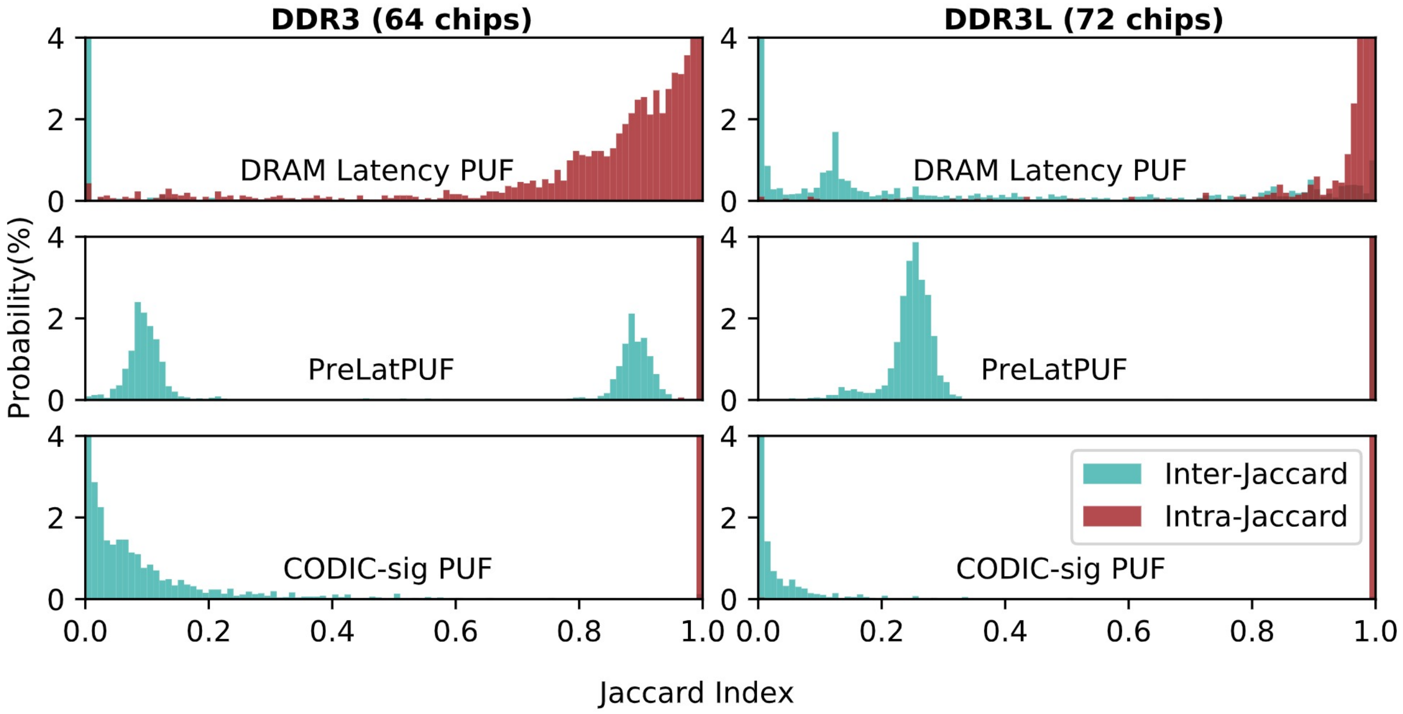
- New DRAM command to the DDRx interface to leverage the CODIC substrate

- We can integrate the new command in the JEDEC standard specification **without extra cost**
  - There is reserved space in the standard for including new commands

- 4 dedicated10-bit **mode registers (MRs)** in DRAM
  - Used to store the timings of the 4 internal timing signals that CODIC can modify

- CODIC uses the existing **mode register set (MRS) command** defined in the DDRx specification to change the contents of its MR

CODIC-sig PUF – DDR3 Jaccard



CODIC-sig PUF – Temperature

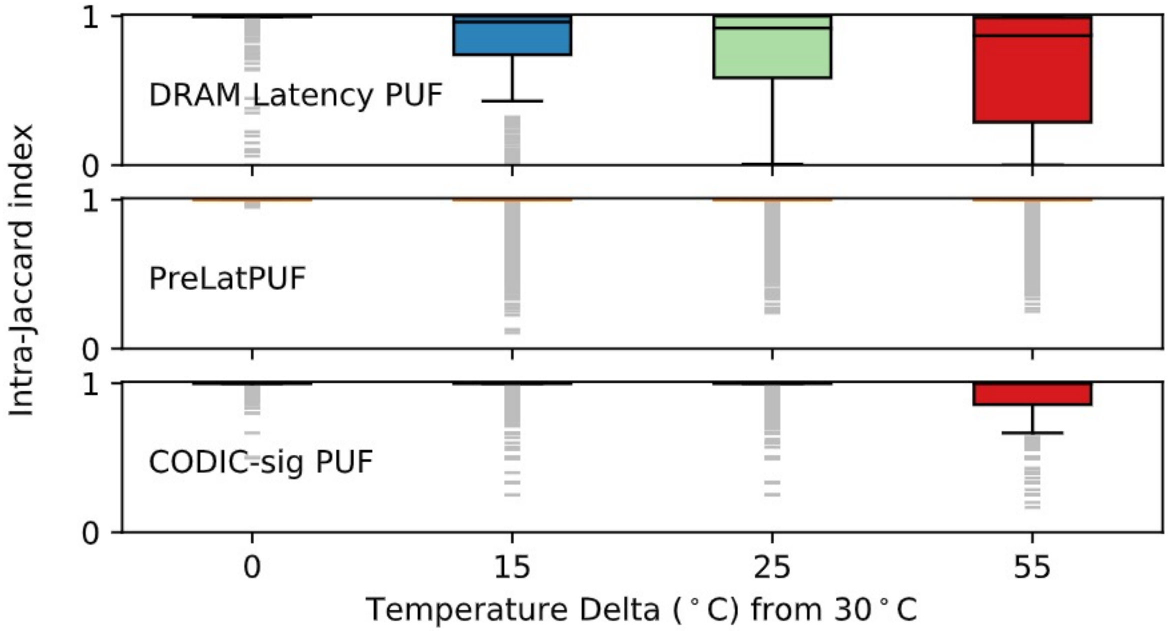


Figure 6: Intra-Jaccard indices vs. temperature.

SAFARI

CODIC-sig PUF – Aging

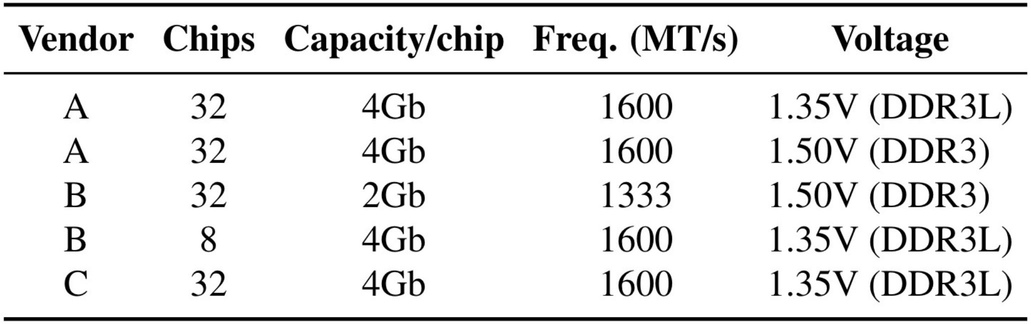
- We use **accelerated aging** techniques to artificially age our DRAM chips
- We artificially age the DRAM chips by operating them at **125C** degrees running **stress tests for 8 hours**.
- We observe from our experiments that CODIC-sig PUF is robust to aging
  - Most PUF responses are the same before and after aging

CODIC-sig PUF – NIST Tests

NIST Test	*p*-value	*Result*
monobit	0.681	PASS
frequency_within_block	1.000	PASS
runs	0.298	PASS
longest_run_ones_in_a_block	0.287	PASS
binary_matrix_rank	0.536	PASS
dft	0.165	PASS
non_overlapping_template_matching	0.808	PASS
overlapping_template_matching	0.210	PASS
maurers_universal	0.987	PASS
linear_complexity	0.0185	PASS
serial	0.988	PASS
approximate_entropy	0.194	PASS
cumulative_sums	0.940	PASS
random_excursion	0.951	PASS
random_excursion_variant	0.693	PASS

Evaluated DDR3 DRAM Chips

Table 3: Characteristics of the 136 evaluated DDR3 DRAM chips.



Vendor	Chips	Capacity/chip	Freq. (MT/s)	Voltage
A	32	4Gb	1600	1.35V (DDR3L)
A	32	4Gb	1600	1.50V (DDR3)
B	32	2Gb	1333	1.50V (DDR3)
B	8	4Gb	1600	1.35V (DDR3L)
C	32	4Gb	1600	1.35V (DDR3L)

App. 3: Secure Deallocation

- Modern applications do not immediately erase data from memory when it is no longer needed

- The Operating System (OS) physically erases the data in a memory region when the program deallocates the memory region
  - Only when the physical memory is required for allocating new data

- Sensitive data could remain in memory for an indefinite amount of time

- **Secure deallocation** is a technique that resets data in memory to zero at the moment of deallocation

- Reduces the time that critical data is vulnerable to attacks

- **CODIC** enables the implementation of secure deallocation with very low latency, energy, and area overhead

Methodology (Sec. Deallocation)

- We evaluate latency and energy overheads of **secure deallocation**, implemented with
  - **Baseline:** fills memory with zero values using common memory write instructions
  - **CODIC:** CODIC-det or CODIC-sig variants
  - **LISA-clone:** Copies bulk data in-DRAM [Chang+, HPCA'16]
  - **RowClone:** Copies bulk data in-DRAM [Seshadri, MICRO'13]

- Ramulator DRAM simulator [Kim+, CAL'15] [https://github.com/CMU-SAFARI/ramulator]
- Pin Instrumentation tool for generating traces [Luk+, PLDI'05]
- Bochs full-system emulator to generate memory traces that include Linux kernel page allocations and deallocations [http://bochs.sourceforge.net/]
- DRAMPower [Chandrasekar+][http://www.drampower.info]

Benchmarks (Sec. Deallocation)

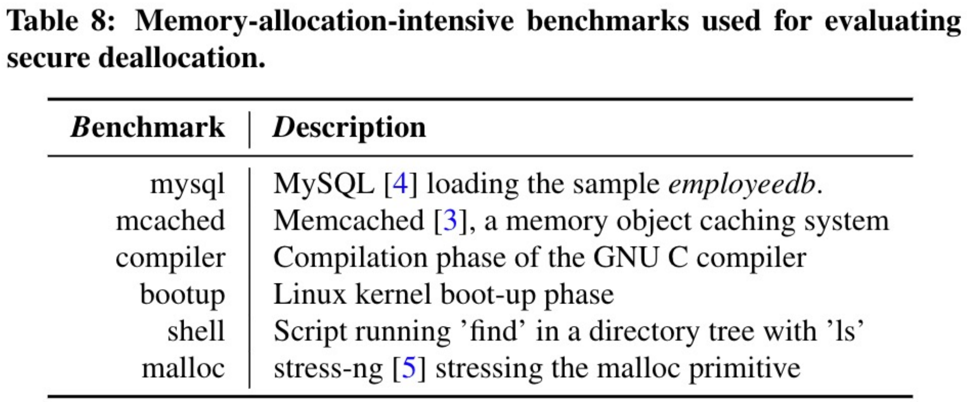


Table 8: Memory-allocation-intensive benchmarks used for evaluating secure deallocation.

Benchmark	*Description*
mysql	MySQL [4] loading the sample *employeedb*.
mcached	Memcached [3], a memory object caching system
compiler	Compilation phase of the GNU C compiler
bootup	Linux kernel boot-up phase
shell	Script running 'find' in a directory tree with 'ls'
malloc	stress-ng [5] stressing the malloc primitive

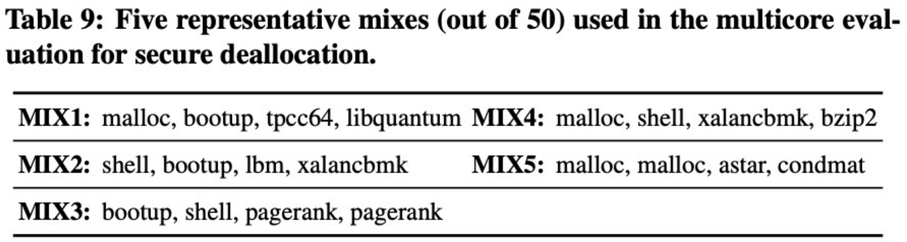
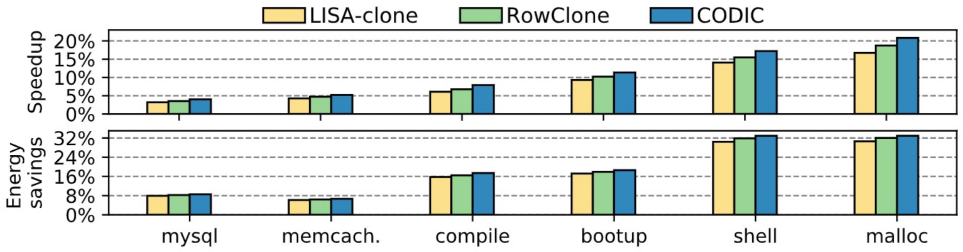


Table 9: Five representative mixes (out of 50) used in the multicore evaluation for secure deallocation.

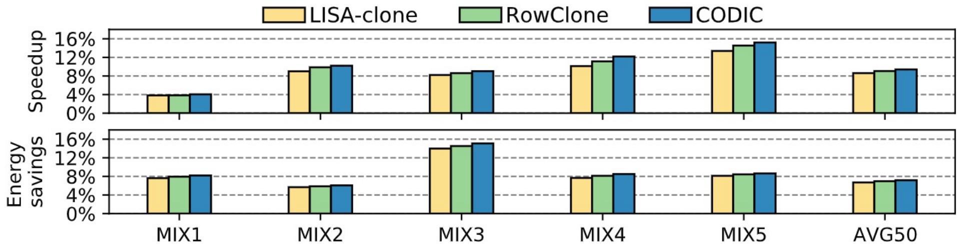
MIX1:	malloc, bootup, tpcc64, libquantum	**MIX4:**	malloc, shell, xalancbmk, bzip2
MIX2:	shell, bootup, lbm, xalancbmk	**MIX5:**	malloc, malloc, astar, condmat
MIX3:	bootup, shell, pagerank, pagerank		

Sec. Dealloc. Latency & Energy

Single core



4-core



SAFARI