HammerSim: A System-Level Tool to model RowHammer

Kaustav Goswami, Ayaz Akram, Hari Venugopalan and Jason Lowe-Power

UC DAVIS
UNIVERSITY OF CALIFORNIA

Background

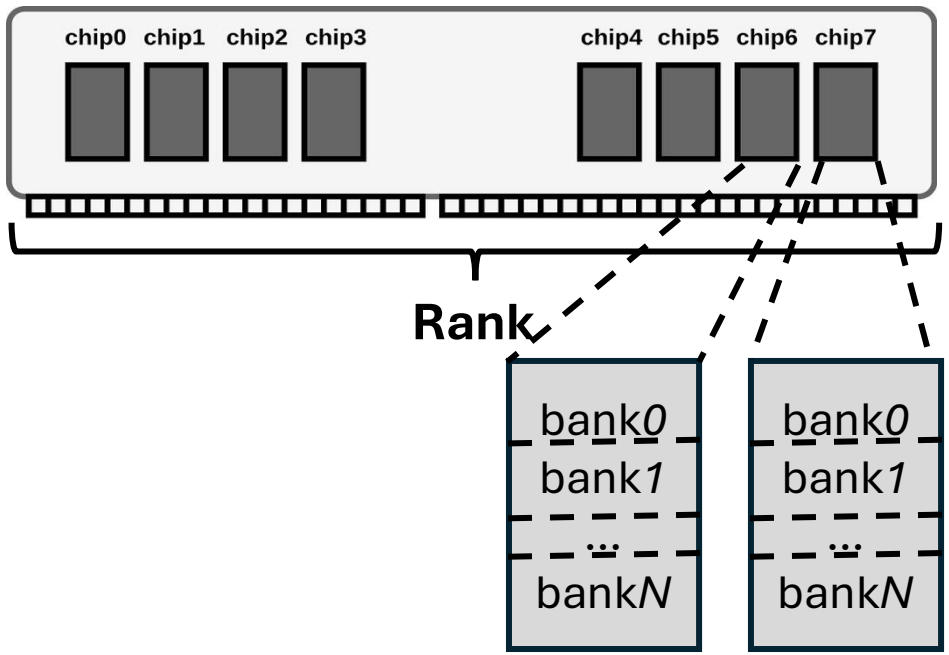
RowHammer

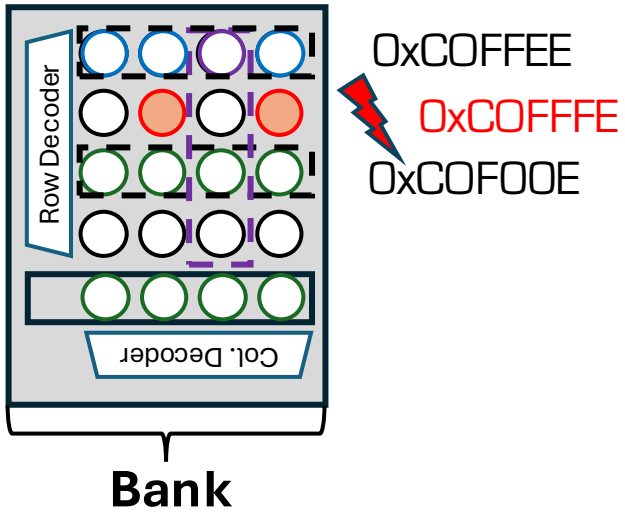
"RowHammer is a security vulnerability in which memory cells interact electrically between themselves by leaking their charges, possibly changing the contents of nearby memory rows that were not addressed in the original memory access."

K. S. Bains *et al.*, Distributed row hammer tracking, US Patent 2012
Y. Kim *et al.*, Flipping bits in memory without accessing them: An experimental study of dram disturbance errors, ISCA 2014

RowHammer

RowHammer





RowHammer Bitflips!

How to study RowHammer in Full-System?

How to study RowHammer in Full-System?

There are three existing ways.

How to study RowHammer in Full-System?

FPGAs

- Like Soft-MC and DRAM Bender
- Highly accurate
- Uncovers TRR
- Cannot model OS abstractions
- Cannot model server-class systems

H. Hassan *et al.,* Softmc: A flexible and practical open-source infrastructure for enabling experimental dram studies, HPCA 2017
A. Olgun *et al.*, DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-theart DRAM Chips, IEEE TCAD 2023

How to study RowHammer in Full-System?

Software exploits

- Like TRRespass and Blacksmith
- Bypasses OS-level protections
- Explore kernel-level vulnerabilities
- Zero hardware visibility
- Hardware-software mitigations

P. Frigo *et al.*, Trrespass: Exploiting the many sides of target row refresh, ArXiv 2020
P. Jattke *et al.*, Blacksmith: Scalable rowhammering in the frequency domain, SP 2022

How to study RowHammer in Full-System?

Trace-based simulators

- Like DRAMsim3 and Ramulator
- Fast
- No hardware-software mitigations
- Uniform probability of bitflips

P. Frigo *et al.*, Trrespass: Exploiting the many sides of target row refresh, ArXiv 2020
P. Jattke *et al.*, Blacksmith: Scalable rowhammering in the frequency domain, SP 2022

Motivation

Predicting RowHammer in new hardware iterations

Tightly coupled RowHammer mitigations

Enter HammerSim

Enter HammerSim

Hardware fidelity

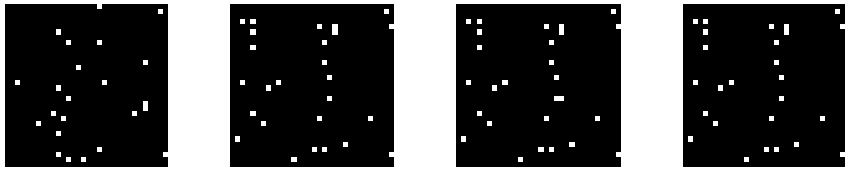
- Cycle-level accuracy
- Non-uniform distribution

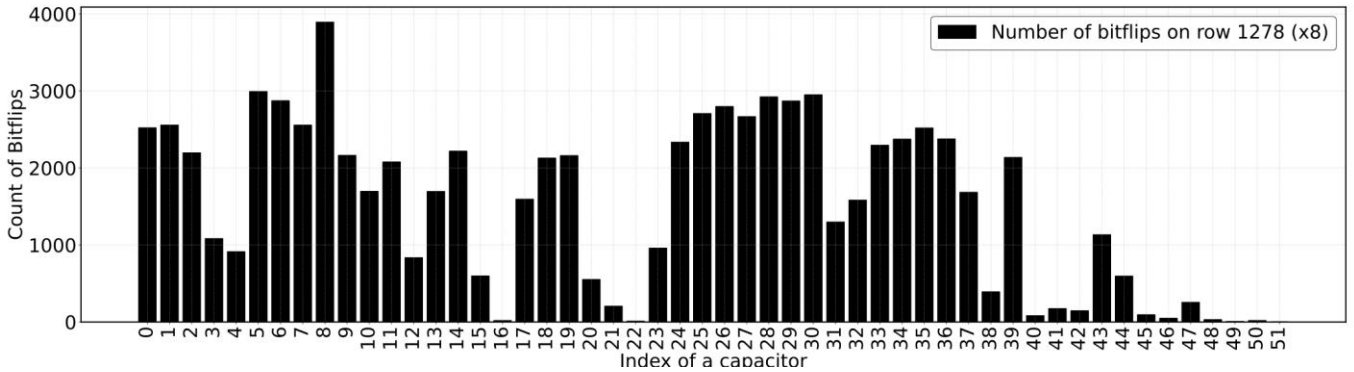
Full-System simulation

- Hardware-software co-design

RowHammer Properties

Spatial localization of bitflips

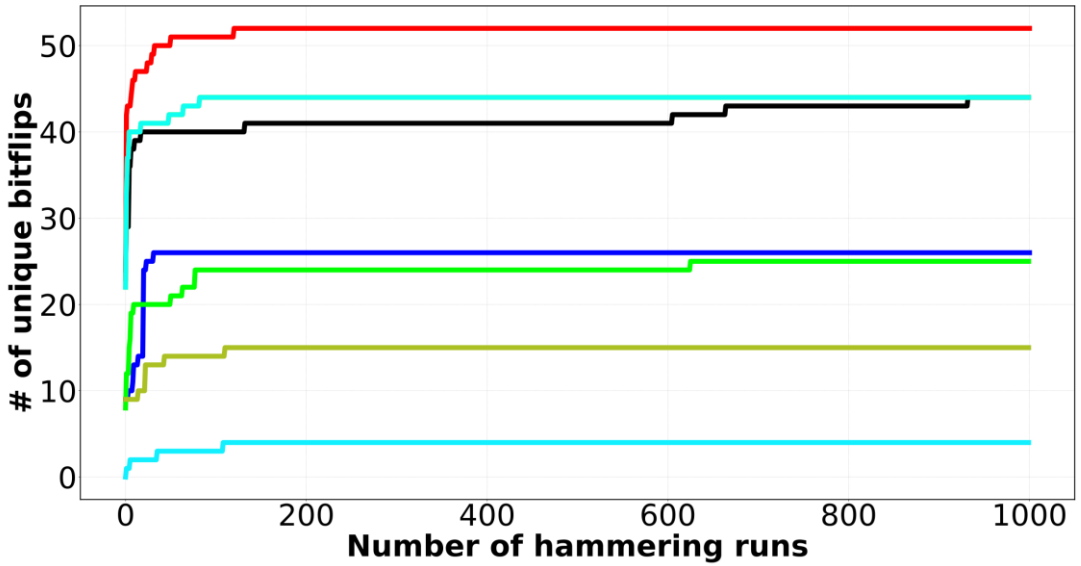




RowHammer Properties

Spatial localization of bitflips

Temporal saturation



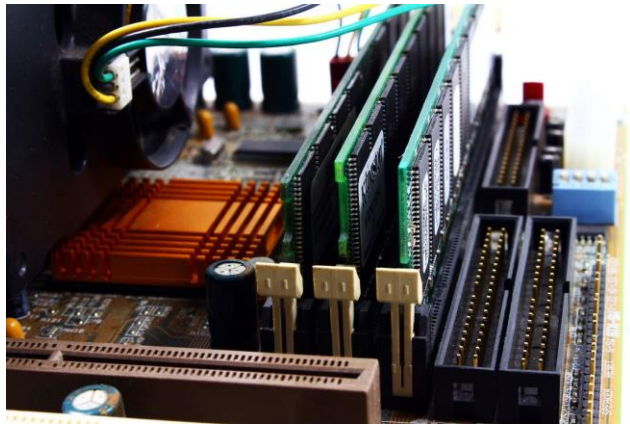
RowHammer Properties

Spatial localization of bitflips

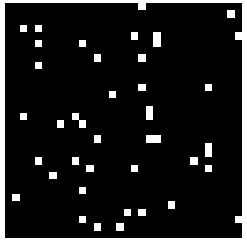
Temporal saturation

Non-Linear aggressor rows

HammerSim in a Nutshell



Row



{ JSON }

gem5

HammerSim

RowHammer stats

J. Lowe-Power et al. 2020. The gem5 simulator: Version 20.0+. arXiv preprint arXiv:2007.03152 (2020).

HammerSim Features

Counter pairs

- Tracks victim and aggressors for bitflips

Online mode

- Live fault injection

Offline mode

- Correcting over and underestimations

Research Enabled

New RowHammer attacks

- Captures system-level effects

New RowHammer mitigations

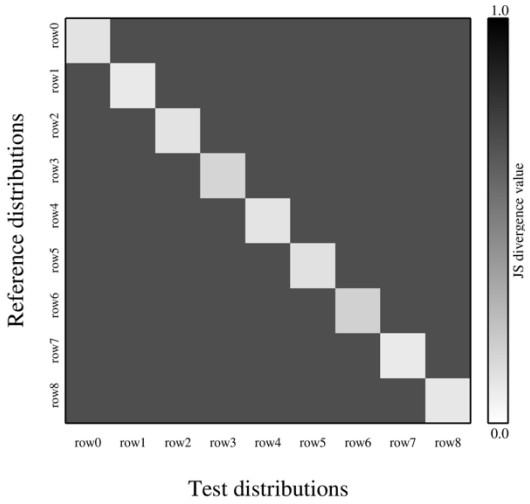
- Hardware-software co-design and ECC

RowHammer PUFs and fingerprints

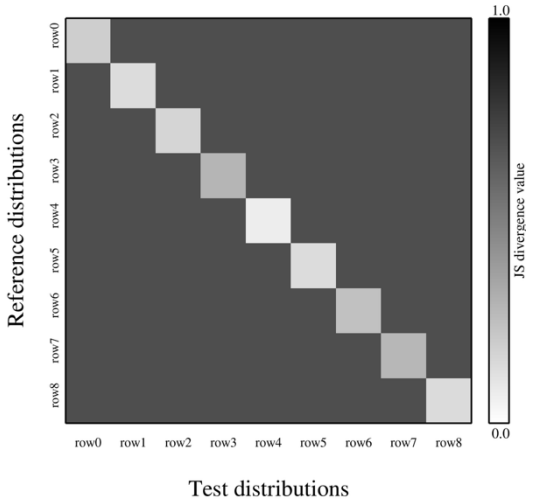
M. Qureshi, Rethinking ecc in the era of row-hammer, DRAMSec 2021
A. Schaller *et al.,* Intrinsic rowhammer PUFs: Leveraging the rowhammer effect for improved security, HOST 2018
H. Venugopalan *et al.*, FP-Rowhammer: DRAM-Based Device Fingerprinting, AsiaCCS 2025

Results

Validating hardware vs simulated bitflips

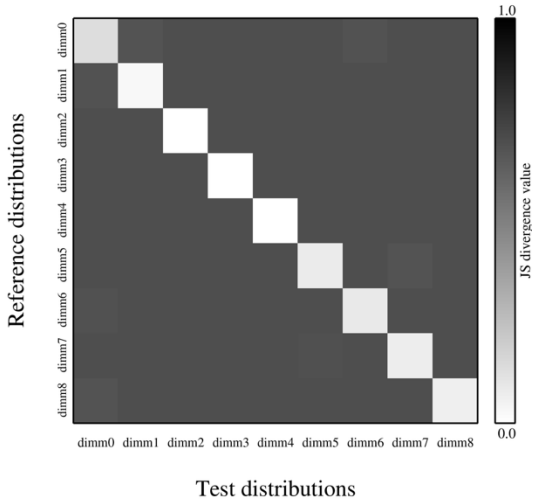


DIMM A



DIMM B

Across Rows



Across DIMMs

Results

Bitflips in the guest OS using Google's rowhammer-test

```
Starting gem5 init... trying to read run script file via readfile.
Running script from gem5-bridge stored in /tmp/script
rowhammer_test
[sudo] password for gem5: clear
cloud-init[1290]: Cloud-init v. 23.1.2-0ubuntu0~22.04.1 finished at Tue,
ource DataSourceNone.  Up 43.50 seconds
cloud-init[1290]: 2025-12-02 18:47:57,632 - cc_final_message.py[WARNING]:
Iteration 0 (after 0.01s)
  Took 338.2 ms per address set
  Took 3.38249 sec in total for 10 address sets
  Took 78.298 nanosec per memory access (for 43200000 memory accesses)
  This gives 102173 accesses per address per 64 ms refresh period
error at 0x7fae5ab6d1f0: got 0xffffffffffffffef
error at 0x7fae5bdbf150: got 0xfeffffffffffffff
error at 0x7fae5bdc31d0: got 0xffffffff7fffffff
error at 0x7fae5bdc3200: got 0xffffffffdfffffff
error at 0x7fae5bdc32a8: got 0xffdfffffffffffff
error at 0x7fae5bdc33c8: got 0xfeffffffffffffff
error at 0x7fae5e2cf008: got 0xffffffff7fffffff
```

M. Seaborn *et al.*, Exploiting the dram rowhammer bug to gain kernel privileges, Black Hat 2015

Results

Functional SECDED ECC

- Corrected, detected and uncorrectable errors

Data corruption in benign applications

- NPB and GAPBS
- Silent errors, kernel panic and segmentation faults

14.7% slower and needs 60% more memory

D. H. Bailey *et al.*, The NAS Parallel Benchmarks, J. of Supercomputing Applications 1991
S. Beamer *et al.*, The gap benchmark suite, ArXiv 2017

Using HammerSim

RowHammer Attack

Using HammerSim

RowHammer Mitigation Sampler

```
// the sampler/counter mechanism is defined here
    .
void
DRAMInterface::activateBank(Rank& rank_ref, Bank
    & bank_ref, Tick act_tick, uint32_t row) {
    ...
    switch (trrVariant) {
        ...
        case N: {
            // write a new mitigation mechanism
            // here.
        }
        ...
    }
    ...
}
```

P. Frigo *et al.*, Trrespass: Exploiting the many sides of target row refresh, ArXiv 2020

Using HammerSim

RowHammer Mitigation

Sampler

```
// the sampler/counter mechanism is defined here
    .
void
DRAMInterface::activateBank(Rank& rank_ref, Bank
    & bank_ref, Tick act_tick, uint32_t row) {
    ...
    switch (trrVariant) {
        ...
        case N: {
            // write a new mitigation mechanism
            // here.
        }
        ...
    }
    ...
}
```

Inhibitor

```
// the inhibitor mechanism is implemented here.
// this is because the inhibitor mechanism is
// triggered when the DRAM device is locked for
// refreshing.
void
DRAMInterface::Rank::processRefreshEvent() {
    ...
    switch(dram.trrVariant) {
        ...
        case N: {
            // write the inhibitor mechanism
            // here to keep DRAM timing
            // consistent.
        }
        ...
    }
    ...
}
```

P. Frigo *et al.*, Trrespass: Exploiting the many sides of target row refresh, ArXiv 2020

Conclusion

In this work, we presented HammerSim

- A full-system cycle-level simulator to model RowHammer
- Allows the study of different aspects of RowHammer
- Supports a dual analysis mode to correct over and underestimations.

HammerSim enables researchers to explore new RowHammer resilient hardware architectures.

The code is released with an open-source license and is available at: https://github.com/darchr/gem5-rowhammer

Thank You.

Full paper and source available at: