USENIX Security '24 Artifact Appendix: ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation

Ataberk Olgun Yahya Can Tugrul Nisa Bostanci Ismail Emir Yuksel
Haocong Luo Steve Rhyner Abdullah Giray Yaglikci Geraldo F. Oliveira Onur Mutlu

ETH Zurich

A Artifact Appendix

A.1 Abstract

Our artifact contains the data, source code, and scripts needed to reproduce our results. We provide: 1) the source code of our simulation infrastructure based on Ramulator and CACTI, and 2) all evaluated workloads' memory access traces and all major evaluation results. We provide Python scripts and Jupyter Notebooks to analyze and plot the results.

A.2 Description & Requirements

To facilitate the artifact evaluation process, we provide SSH access to our internal Slurm-based infrastructure with all the software dependencies already installed. *We strongly recommend using Slurm for running experiments in bulk.* We describe hardware/software dependencies as well as installation instructions for two artifact evaluation scenarios: 1) Remote access, where the artifact evaluator remotely accesses our infrastructure to evaluate the artifact, and 2) personal computer, where the artifact evaluator evaluates the artifact using their own personal computer. Moreover, we will happily help artifact evaluators who want to use their own Slurm-based infrastructure by porting our scripts to their environment. Please contact us through HotCRP and/or the AE committee for details.

A.2.1 Security, privacy, and ethical concerns

None.

A.2.2 How to access

The source code and analysis scripts are provided in our open source repository at the following stable reference `https://github.com/CMU-SAFARI/ABACuS/tree/7491a667fd1a667b556ef81a8eaa035f69461644`. We provide all evaluated workloads' memory access traces and all evaluation results at `https://zenodo.org/doi/10.5281/zenodo.10575682`.

A.2.3 Hardware dependencies

Remote access. None.
Personal computer. We recommend using a PC with 32 GiB of main memory. Approximately 50 GiB of disk space is needed to store intermediate and final experimental results.

A.2.4 Software dependencies

Remote access. None.
Personal computer. `Podman` (we have tested Podman version 3.4.4 on Ubuntu 22.04.1) and `git`.

A.2.5 Benchmarks

We use workload memory traces collected from SPEC2006, SPEC2017, TPC, MediaBench, and YCSB benchmark suites. The used memory traces are available at `https://zenodo.org/doi/10.5281/zenodo.10575682`.
Remote access. The traces are already in the project directory.
Personal computer. A provided script will download and extract the traces.

A.3 Set-up

A.3.1 Installation

Remote access. None.
Personal computer. Clone the git repository using `$ git clone -b usenix24-ae git@github.com:CMU-SAFARI/ABACuS.git`.

A.3.2 Basic Test

Remote access. Run `./simple_test.sh` in the project directory. A successful run takes around 30 seconds and will output `REF CMD energy: 8483.076 nJ` to standard output (as the last line) and put various simulation results (execution statistics) in a file named `ddr4DDR4stats.stats`.
Personal computer. Run `./simple_test_podman.sh`. This will download and extract all workload execution traces as

an intermediate step, which might take around 20 minutes. Expected results are the same as above.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Modern memory-intensive workloads and existing RowHammer attacks activate DRAM rows with the same row address in multiple DRAM banks (i.e., sibling rows) at around the same time. A single shared activation counter, which stores the highest activation count among the activation counts of all sibling rows, can reasonably accurately represent the activation count of all sibling rows. This property of the shared activation counter becomes stronger as the RowHammer threshold reduces. This is proven by the experiment (E1) described in Section 3.1 whose results are illustrated in Figures 2 and 3.

(C2): ABACuS induces small system performance and DRAM energy overheads on average across all tested single-core and multi-core workloads for RowHammer threshold values of 1000, 500, 250, and 125. ABACuS's performance and DRAM energy overheads are closer to the most-performance-efficient state-of-the-art mechanism. ABACuS outperforms and consumes less DRAM energy than the most-area-efficient state-of-the-art (counter-based) mechanism. This is proven by the experiment (E2) described in Sections 8 and 9 whose results are illustrated in Figures 7, 8, 9, 10, 11, and 12.

(C3): At the RowHammer threshold (nRH) of 125, ABACuS performs very similarly to the best prior performance- and energy-efficient RowHammer mitigation mechanism while requiring 22.72× smaller chip area. This is proven by the experiment (E3) described in Section 7.1 whose results are illustrated in Table 1.

A.4.2 Experiments

(E1 and E2): *[Ramulator simulations] [15 human-minutes + 77 compute-hours (assuming approx. 1280 cores available for running tasks in parallel) + 50GB disk]: Execute Ramulator simulations to generate data supporting C1 and C2. Plot all figures that prove C1 and C2.*

Execution (remote access): 1) Execute `$ ./run_artifact_without_podman.sh`. This will launch all Ramulator simulation jobs using Slurm. Wait for simulations to end. The slowest multi-core simulation takes approximately 72 hours. 2) Navigate to `scripts/` and run `$ python3 create_figures.py`. The script takes approximately 5 hours to execute (a large fraction of the execution time is spent on one-time preprocessing of the data used for Figures 2 and 3).

Execution (personal computer): 1) Execute `$ ./run_artifact_with_podman.sh -personalcomputer`. This will launch all Ramulator simulation jobs. Wait for simulations to end. 2) run `$ ./create_figures_with_podman.sh`. The script takes approximately 5 hours to execute (a large fraction of the execution time is spent on one-time preprocessing of the data used for Figures 2 and 3). Use `$ squeue -u aevaluator` to monitor the status of simulations.

Results: A PDF of every figure proving C1 and C2 is created in `scripts/ae_scripts/`.

(E3): *[CACTI simulations] [5 human-minutes + 1 compute-minute + 10MB disk]: Run CACTI simulations to generate chip area estimation results.*

Execution (remote access): Navigate to `abacus_cacti/` and run `$ python3 results.py`.

Execution (personal computer): Run `$ ./area_results_with_podman.sh`.

Results: The area cost of ABACuS, Graphene, and Hydra is printed (along with all data used to fill the cells of Table 1). The ratio of Graphene's "Total Area" at "nRH:125" (5.68mm^2) to ABACuS's "Total Area" at "nRH:125" (0.25mm^2) is 22.72×.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2024/`.