HammerSim: A Tool to Model Rowhammer

Kaustav Goswami, Ayaz Akram, Hari Venugopalan, Jason Lowe-Power
{kggoswami,yazakram,hvenugopalan,jlowepower}@ucdavis.edu
University of California, Davis

ABSTRACT

Recently the rowhammer vulnerability has affected modern memory devices, which allows an attacker to cause bitflips without accessing the corresponding cell. The rowhammer effects can exacerbate future memory technologies due to scaling. Hence, we need to invest in studying and mitigating rowhammer attacks. We, therefore, propose a model to simulate RowHammer in gem5 to capture the system-level interaction of RowHammer.

1 INTRODUCTION

Dynamic random access memory, or DRAM, is the *de-facto* choice for main memory design due to its cost-effectiveness. However, due to scaling, researchers have noticed variations in the nominal parameters of these devices [6, 18, 32]. Recently, it has been found that accessing (or `ACT`ivating) a particular cell of a DRAM module repeatedly causes data corruption in the neighborhood. This is called RowHammer [4, 15]. There have been extensive studies done to study and mitigate rowhammer [2, 15, 17, 25, 29]. Most of such variations or data corruptions are unfortunately not captured in simulators [8].

We aim at closing the gap between DRAM DIMMs and its simulated counterpart along the lines of RowHammer. Researchers have characterized bitflips in the past [1, 23, 24, 30]. There exists circuit-level models to simulate RowHammer [13]. FPGAs help in studying RowHammer with software-based memory controllers [11, 21]. However, this approach necessitates specialized setup. Currently we do not posses a comprehensive model of RowHammer at the system level. Developing such a model is crucial as this enables us to study the interaction of workloads and data-corruption alongside estimating the behavior of future DRAM technology.

As a precursor to the simulation model of RowHammer, we performed extensive hammering on different DRAM modules (or DIMMs[1]). The analytical model of RowHammer that people reply on assumes an equally likely probability of a bitflip in a DRAM row when the number of accesses crosses a certain threshold in the neighborhood [2, 15, 17, 25, 29]. We saw that this is not the case on actual DRAM DIMMs. Most unique bitflips saturate over time as not all regions of the memory are vulnerable. This is shown in Figure 1, where we see that the number of unique bitflips saturates over time. This information is not previously captured while modeling RowHammer, which can largely influence the future designs of mitigation techniques.

Modeling the aforementioned reliability would enable researchers to estimate the probability of obtaining the correct output of a program in a non-trusted environment like the cloud. The RowHammer threshold is decreasing as DRAMs are becoming denser [14], which motivates the need for such a model. In this abstract, we propose a system-level RowHammer model called *HammerSim*, which we have integrated with gem5 [19].

[1]A DIMM refers to dual inline memory module.

YArch '23, March 26, 2023, Vancouver, Canada

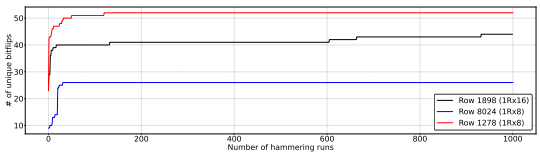


Figure 1: Count of unique bitflips on different DRAM DIMMs' row.

2 METHODOLOGY

Our objective is to model RowHammer closely to its hardware counterpart. Toward this, we have attempted to reproduce bitflips on real DDR4 DIMMs. Based on our findings and previous literature, we conclude that RowHammer is a result of a combination of several probability distributions. These distributions are primarily based on the inherent variations induced during manufacturing time [15, 23, 31]. However, there is no deterministic correlation between process variation and RowHammer [15]. We observed that a weak memory cell (or a capacitor) may or may not exhibit a bitflip under a RowHammer attack. This adds to the randomness of RowHammer. This is a consistent observation across several attack papers, where at each hammering instance, a vulnerable bit does not always flip [7, 12, 23, 27, 28, 31]. Figure 2 shows instances of bitflips on a single row (1278), across 4 different runs. A white dot represents a bitflip in a column. Figure 2(f) plots the histogram of bitflip count across 1000 hammering instances.

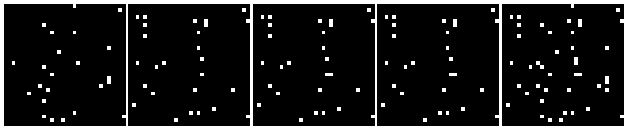
2.1 Simulation Model

Overall, we have the following probabily distributions for modelling rowhammer bitflips:

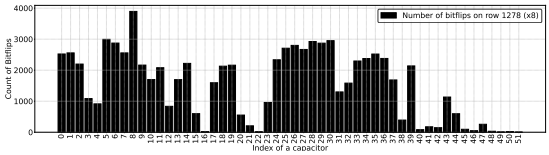
2.1.1 A model for process variaiton. We model a variation map based on VARIUS [26], a statistical model of process variation. It models process variation as a multivariate normal distribution. The outcome is a binary decision: *whether a given cell or a capacitor is weak or a strong cell?* This is represented as `WC`, or weak cell in Equation 1.

2.1.2 Uniform probability for flipping a bit. For a given *weak* cell `WC`, we use a uniform probability function to flip the cell. This is a property of the DIMM. We experimentally determine the probability which ranges from $\frac{1}{5\times10^{10}}$ to $\frac{1}{5\times10^{8}}$ for a bitflip. This can be tuned by the user in the simulation framework. `F_WC` represents a *flippable* bit in Equation 1

2.1.3 Probability escalation of an N-sided rowhammer attack. This probability distribution is correlated with the other aggressor rows in the neighborhood. For this, we use counter-pairs while counting

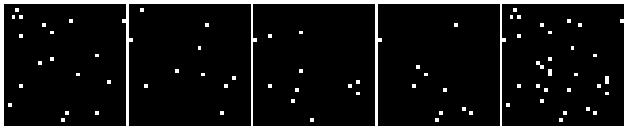


(a) Run 1 (b) Run 2 (c) Run 3 (d) Run 4 (e) Superimposed Image

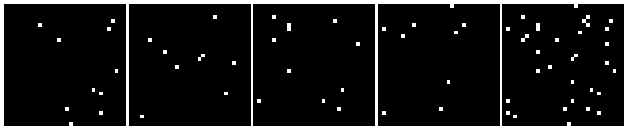


(f) Histogram of bitflip counts on row number 1278 across 1000 runs.

Figure 2: Observed bitflips on a single row on a width 8 DRAM DIMM.



(a) Run 1 (b) Run 2 (c) Run 3 (d) Run 4 (e) Superimposed Image



(f) Run 1 (g) Run 2 (h) Run 3 (i) Run 4 (j) Superimposed Image

Figure 3: Simulated result of bitflips on a single row. The top set of runs used the variation map taken from the actual DRAM DIMMs. The bottom set is generated using a statistical model of process variation.

`ACTs`. This is also a property of the DIMM. We use an experimentally determined probability of $\frac{1}{2.5\times10^3}$ for the same.

2.1.4 Half-double. Half-double [9, 16] can only be modeled via counters of triggering a bitflip. For every `ACT` on a row, we have to maintain counters for each of the rows (4) in the blast radius. This is not trivial to maintain as checking the memory after each `ACT` increase the simulation time exponentially. Therefore, we simply monitor half-double access patterns only during a refresh event.

We combine these aforementioned probability distributions in a conventional manner, which is given by:

$$P_{bitflip} = P(WC|F\_WC) \times \frac{1}{P(n\text{-}sided)} \times \frac{1}{P(half\text{-}double)} \quad (1)$$

3 ANALYSIS

HammerSim is implemented on gem5 [19], a full-system cycle-level simulator. We modeled the probability distributions of RowHammer (referred to in Section 2) within gem5's memory interface. In addition, we have also modeled the mitigation mechanism installed on one of the DRAM vendors using the reverse-engineered understanding of the same depicted in U-TRR [10][2]. The RowHammer bitflip map generated via gem5 is shown in Figure 3. Figures 3(a)–3(e) simulate bitflips in gem5 with a variation map taken from the actual DRAM DIMM. The measured similarity index between the real and the simulated run is 0.31 in terms of JS Divergence [22] value (lower is better). The lower set of images represents bitflips with a statistically generated variation map.

This abstract does a general analysis of RowHammer consequences on HPC workloads taken from GAPBS [5] and NAS-Parallel benchmark suite [3]. We simulate an x86 system with 2 levels of caches (L1: 32KB + 32KB; L2: 256KB). Figure 4 shows the instances of row `ACTs` crossing the LPDDRx (16.8K – 4.8K), DDR4 (45K) and DDR3 (139K) thresholds. We see that there are 20041.16 instances on average, where rows cross the DDR4 threshold of 45K ACTs. The figure is worse for LPDDR [20] DIMMs, where the threshold count is even lower. The threshold is likely to be even lower for DDR5 DRAM DIMMs.

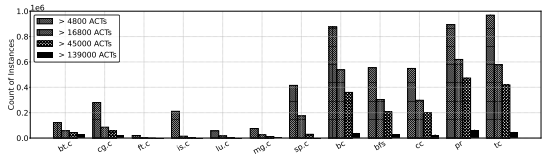


Figure 4: Observed instances of `ACT` count crossing 4.8K, 16.8K (LPDDRx), 45K (DDR4) and 139K (DDR3).

We did a bitflip estimation of RowHammer by probabilistically flipping bits during runtime. We tuned the probability of a bitflip from $\frac{1}{10^9}$ to 1 for each of these benchmarks. Figure 5 plots the same. This shows that even benign applications have a significant probability of flipping bits. Note that for this experiment, we have only considered N-sided probability. This value is predicted to worsen due to increasing density.

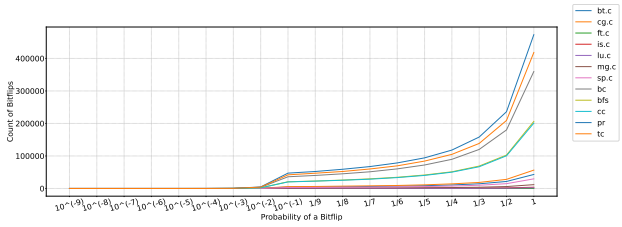


Figure 5: Estimation of bitflips by varying the probability of a bitflip.

4 CONCLUSION

In this abstract, we have proposed a preliminary model of RowHammer. The model is presented as a tool called HammerSim, which is

[2]correspond's to U-TRR's Vendor B.

implemented in a full-system simulator. This allows us to study the interaction between data-corruption and real-world workloads. In the future, we plan on quantitatively evaluating RowHammer and its mitigation as a metric at the system-level using HammerSim. Furthermore, we plan on extending this infrastructure to simulate cloud environments with malicious programs running alongside benign applications to create a more realistic scenario.

REFERENCES

[1] Barbara Aichinger. 2015. DDR memory errors caused by Row Hammer. In *2015 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–5. https://doi.org/10.1109/HPEC.2015.7322462

[2] Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd Austin. 2016. ANVIL: Software-Based Protection Against Next-Generation Rowhammer Attacks. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems* (Atlanta, Georgia, USA) *(ASPLOS '16)*. Association for Computing Machinery, New York, NY, USA, 743–755. https://doi.org/10.1145/2872362.2872390

[3] David H Bailey, Eric Barszcz, John T Barton, David S Browning, Robert L Carter, Leonardo Dagum, Rod A Fatoohi, Paul O Frederickson, Thomas A Lasinski, Rob S Schreiber, et al. 1991. The NAS Parallel Benchmarks. *The International Journal of Supercomputing Applic ations* 5, 3 (1991), 63–73.

[4] Kuljit S. Bains and John B. Halbert. 2012. Distributed row hammer tracking. US Patent US20140095780A1.

[5] Scott Beamer, Krste Asanović, and David Patterson. 2015. The GAP Benchmark Suite. *arXiv preprint arXiv:1508.03619* (2015).

[6] S. Borkar. 2005. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro* 25, 6 (2005), 10–16. https://doi.org/10.1109/MM.2005.110

[7] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2020. TRRespass: Exploiting the Many Sides of Target Row Refresh. arXiv:2004.01807 [cs.CR] https://arxiv.org/abs/2004.01807

[8] Saugata Ghose, Abdullah Giray Yaglikçi, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X. Liu, Hasan Hassan, Kevin K. Chang, Niladrish Chatterjee, Aditya Agrawal, Mike O'Connor, and Onur Mutlu. 2018. What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 3, Article 38 (dec 2018), 41 pages. https://doi.org/10.1145/3224419

[9] Google LLC. 2021. "Half-Double": Next-Row-Over Assisted Rowhammer. , 22 pages. https://raw.githubusercontent.com/google/hammer-kit/main/20210525_half_double.pdf

[10] Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu. 2021. Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications. https://doi.org/10.48550/ARXIV.2110.10603

[11] Hasan Hassan, Nandita Vijaykumar, Samira Khan, Saugata Ghose, Kevin Chang, Gennady Pekhimenko, Donghyuk Lee, Oguz Ergin, and Onur Mutlu. 2017. SoftMC: A flexible and practical open-source infrastructure for enabling experimental DRAM studies. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 241–252.

[12] Patrick Jattke, Victor Van Der Veen, Pietro Frigo, Stijn Gunter, and Kaveh Razavi. 2022. BLACKSMITH: Scalable Rowhammering in the Frequency Domain. In *2022 IEEE Symposium on Security and Privacy (SP)*. 716–734. https://doi.org/10.1109/SP46214.2022.9833772

[13] Yichen Jiang, Huifeng Zhu, Dean Sullivan, Xiaolong Guo, Xuan Zhang, and Yier Jin. 2021. Quantifying Rowhammer Vulnerability for DRAM Security. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 73–78. https://doi.org/10.1109/DAC18074.2021.9586119

[14] Jeremie S Kim, Minesh Patel, A Giray Yağlıkçı, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu. 2020. Revisiting rowhammer: An experimental analysis of modern dram devices and mitigation techniques. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 638–651.

[15] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *Proceeding of the 41st Annual International Symposium on Computer Architecture*. IEEE Press.

[16] Andreas Kogler, Jonas Juffinger, Salman Qazi, Yoongu Kim, Moritz Lipp, Nicolas Boichat, Eric Shiu, Mattias Nissler, and Daniel Gruss. 2022. Half-Double: Hammering From the Next Row Over. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 3807–3824. https://www.usenix.org/conference/usenixsecurity22/presentation/kogler-half-double

[17] Eojin Lee, Ingab Kang, Sukhan Lee, G. Edward Suh, and Jung Ho Ahn. 2019. TWiCe: Preventing Row-Hammering by Exploiting Time Window Counters. In *Proceedings of the 46th International Symposium on Computer Architecture* (Phoenix, Arizona) *(ISCA '19)*. Association for Computing Machinery, New York, NY, USA, 385–396. https://doi.org/10.1145/3307650.3322232

[18] Kevin Loughlin, Stefan Saroiu, Alec Wolman, Yatin A. Manerkar, and Baris Kasikci. 2022. MOESI-Prime: Preventing Coherence-Induced Hammering in Commodity Workloads. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (New York, New York) *(ISCA '22)*. Association for Computing Machinery, New York, NY, USA, 670–684. https://doi.org/10.1145/3470496.3527427

[19] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach, Nils Asmussen, Brad Beckmann, Srikant Bharadwaj, et al. 2020. The gem5 simulator: Version 20.0+. *arXiv preprint arXiv:2007.03152* (2020).

[20] Micron Technology. 2005. *TN-46-12: Mobile DRAM Power-Saving Features and Power Calculations.* Technical Report TN46_12.

[21] Onur Mutlu and Jeremie S. Kim. 2020. RowHammer: A Retrospective. *Trans. Comp.-Aided Des. Integ. Cir. Sys.* 39, 8 (aug 2020), 1555–1571. https://doi.org/10.1109/TCAD.2019.2915318

[22] Notes on AI. [n. d.]. Jensen-Shannon Divergence. https://notesonai.com/Jensen%E2%80%93Shannon+Divergence.

[23] Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu. 2021. A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (Virtual Event, Greece) *(MICRO '21)*. Association for Computing Machinery, New York, NY, USA, 1182–1197. https://doi.org/10.1145/3466752.3480069

[24] Kyungbae Park, Donghyuk Yun, and Sanghyeon Baeg. 2016. Statistical distributions of row-hammering induced failures in DDR3 components. *Microelectronics Reliability* 67 (2016), 143–149. https://doi.org/10.1016/j.microrel.2016.10.014

[25] Yeonhong Park, Woosuk Kwon, Eojin Lee, Tae Jun Ham, Jung Ho Ahn, and Jae W. Lee. 2020. Graphene: Strong yet Lightweight Row Hammer Protection. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–13. https://doi.org/10.1109/MICRO50266.2020.00014

[26] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. 2008. VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects. *IEEE Transactions on Semiconductor Manufacturing* 21, 1 (2008), 3–13. https://doi.org/10.1109/TSM.2007.913186

[27] Andre Schaller, Wenjie Xiong, Nikolaos Athanasios Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, and Jakub Szefer. 2017. Intrinsic Rowhammer PUFs: Leveraging the Rowhammer effect for improved security. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. https://doi.org/10.1109/hst.2017.7951729

[28] Mark Seaborn and Thomas Dullien. 2015. Exploiting the DRAM rowhammer bug to gain kernel privileges. *Black Hat* 15 (2015), 71.

[29] Mungyu Son, Hyunsun Park, Junwhan Ahn, and Sungjoo Yoo. 2017. Making DRAM Stronger Against Row Hammering. In *Proceedings of the 54th Annual Design Automation Conference 2017* (Austin, TX, USA) *(DAC '17)*. Association for Computing Machinery, New York, NY, USA, Article 55, 6 pages. https://doi.org/10.1145/3061639.3062281

[30] Andrei Tatar, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer. In *Research in Attacks, Intrusions, and Defenses*, Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer International Publishing, Cham, 47–66.

[31] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clementine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. 2016. Drammer: Deterministic Rowhammer Attacks on Mobile Platforms. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1675–1689. https://doi.org/10.1145/2976749.2978406

[32] B. Zhao, Y. Du, J. Yang, and Y. Zhang. 2013. Process Variation-Aware Nonuniform Cache Management in a 3D Die-Stacked Multicore Processor. *IEEE Trans. Comput.* 62, 11, 2252–2265. https://doi.org/10.1109/TC.2012.129