2027 IEEE International Symposium on High-Performance Computer Architecture (HPCA)

ECClipse: Eclipsing On-Die ECC for Rowhammer Characterization on DDR5

Patrick Jattke†, Diego Meyer†, Salman Qazi‡, Daniel Moghimi‡ and Kaveh Razavi†
ETH Zurich†, Google‡
pjattke@ethz.ch, dimeyer@ethz.ch, sqazi@google.com, danielmm@google.com, kaveh@ethz.ch

Abstract*—Rowhammer has been threatening system security for more than a decade. Designing secure mitigations requires understanding how susceptible the underlying DRAM substrate is to Rowhammer. However, the susceptibility of the latest DDR5 devices remains unknown because on-die ECC masks the cells most vulnerable to Rowhammer. To address this gap, we introduce ECClipse, a new technique that can identify how quickly the weakest DRAM cells flip due to Rowhammer *after* the corrections that are performed by on-die ECC. The key idea is to induce errors using failure mechanisms different from Rowhammer, such as retention failures and Rowpress, to create visibility for the most vulnerable cells that are otherwise eclipsed by on-die ECC. ECClipse enable us to measure the Rowhammer threshold of 192 DDR5 DRAM chips (80 from SK Hynix, 64 from Micron, 48 from Samsung) from 20 DDR5 RDIMM devices in our test pool for the first time. We find that the minimum number of accesses required to trigger Rowhammer, also known as minimum hammer count, has decreased to 5 100, 6 500, and 6 900 for Samsung, SK Hynix, and Micron devices, respectively. Although Rowhammer thresholds in DDR5 have declined, these results show that the situation is not as severe as previously assumed: the thresholds of the latest DDR5 devices remain more than one order of magnitude higher than the 10-51x drop anticipated by proposed Rowhammer mitigations. Consequently, existing mitigations report up to 12.8x, 5.2x, and 5x higher storage, energy, and performance overheads, respectively, than necessary to protect the most vulnerable DDR5 devices.

I. Introduction

A plethora of studies over the past decade have shown the dangers of Rowhammer attacks [6], [12], [15], [19], [30], [40], [62], [63], [67], [73], [74], [79]. In response, academia has come up with a vast number of mitigation proposals [7], [8], [20], [23], [33], [46], [48], [55], [58]–[60], [64]–[66], [70], [77], [81] while the DRAM industry has deployed the Target Row Refresh (TRR) mitigation inside DDR4 and DDR5 devices [16], [17], [21], [24], [26] and specified new mitigations such as Refresh Management (RFM) [27] and Per Row Activating Counting (PRAC) [28]. Crucially, these mitigations rely on knowing how susceptible the protected device is to Rowhammer—information that is currently missing for DDR5 devices. This paper designs novel techniques for uncovering the device vulnerability on commodity DDR5 devices to enable the correct configuration of Rowhammer mitigations.

HCmin. A certain number of activations to some *aggressor rows* are necessary to trigger bit flips in a particular *victim row* [32], [34]. While DRAM cells in different rows require a different number of activations to their aggressor rows, the Rowhammer-weakest DRAM cells require a *minimum number* of such activations to their aggressor rows. This number, known as HCmin, is crucial to know for the correct configuration and evaluation of Rowhammer mitigations that try to prevent Rowhammer bit flips by preemptively refreshing the victim rows. Measuring HCmin in DDR4 devices is straightforward by simply checking a sufficient number of victim rows against increasingly smaller HC values, until no more bit flips appear [16], [21], [32]. The introduction of On-Die Error Correction Code (ODECC) in DDR5 [28], however, makes the measurement of HCmin challenging as only bit flips that overwhelm the ODECC protection become visible outside the device. As such, recent work [17], [49] only reports HCmin *after ODECC*, which is not the true HCmin. While DRAM vendors can disable ODECC for internal testing, such custom features are undocumented and unavailable for indepedent evaluation which is crucial for devising verifiable and efficient mitigations. BEER [57] is the only existing work that studies ODECC by observing multiple retention-based faults that ODECC cannot correct. While this technique can recover information the ECC code word layout, it cannot measure HCmin since at least two Rowhammer faults will be necessary to avoid correction, masking the the lowest Rowhammer threshold (i.e., the potential HCmin). We hence need a new approach to measure HCmin on DRAM devices that are equipped with ODECC.

Circumventing ODECC. We make the key observation that since the errors caused by different fault types (e.g., Rowhammer or retention) are distinct, as reported in previous work [32], [34], [39], [44], [84], we can induce visibility over what happens before ODECC using one fault type while studying another. We rely on this observation in the constructions of two new techniques, collectively referred to as *ECClipse*, that enable measuring the HCmin of commodity DDR5 devices before ODECC. *ECCfade* uses retention errors and *ECCstress* uses Rowpress [44] to induce bit flips into the ODECC code word. Because retention errors and Rowpress do not correlate strongly with Rowhammer [34], [44], ECCfade and ECCstress can trigger bit flips in ODECC codewords outside of the weakest Rowhammer cells (i.e., those that determine HCmin). Consequently, any additional bit flip caused by Rowhammer becomes immediately visible after creating visibility with our ECClipse methods. We perform a comparative evaluation of ECCfade and ECCstress to find the most suitable method for our large-scale evaluation of HCmin on DDR5 devices.

Characterizing HCmin in DDR5. We experimentally characterize HCmin on 20 RDIMMs from the three largest DRAM vendors for the first time. Our results show that HCmin in DDR5 has worsened compared to DDR4 devices for two of three vendors. The lowest HCmin we observed for Samsung devices has decreased from 11 K (accesses to each aggressor in a double-sided pair [21]) to 5.1 K and for SK Hynix devices from 11 K to 6.5 K. These results show that not only existing estimations are pessimistic about the worsening of the Rowhammer vulnerability [48], [58], [65], [66], but in some devices the situation remains roughly the same as in the previous generation of devices.

Contributions. We make the following contributions:

1) We propose two new techniques for measuring HCmin on DDR5 devices exploiting retention errors (ECCfade) and Rowpress (ECCstress) while circumventing on-die ECC.
2) We comprehensively evaluate the two techniques to find the most suitable method for a large-scale characterization of HCmin on DDR5 devices.
3) We characterize the HCmin of 192 DDR5 DRAM chips from the three main DRAM vendors for the first time.
4) We evaluate the impact of these new estimations on the energy, storage, and performance overheads of previous mitigations.

II. Background and Motivation

We discuss the background on DRAM in § II-A, introduce Rowhammer in § II-B, and motivate this paper in § II-C.

A. Dynamic Random-Access Memory

DRAM modules are hierarchically structured to balance performance, density, and parallelism [1], [31], [35], [47], [52]. Each DRAM cell stores a single bit of data in a capacitor. Over time, the cell will gradually lose charge, resulting in *retention failures*. To prevent this, DRAM cells are periodically refreshed, once every tens of milliseconds (e.g., 64 ms for most DDR4 devices [21]). The cells are arranged into rows and columns, forming a bank. Each bank includes a row decoder to activate a specific row and load its contents into the row buffer, and a column decoder to access specific columns. Multiple banks are grouped into bank groups within a single DRAM chip, enabling greater parallelism and improving memory access throughput. Dual In-line Memory Modules (DIMMs) connect multiple DRAM chips to the CPU's memory controller via dedicated memory channels, which coordinate data transfers and refresh operations.

DRAM commands. An activate (ACT) command brings a row into the row buffer. Once a row is activated, it can be read/written within the row buffer using read/write (RD/WR) commands. A precharge (PRE) command prepares the bank for activating a different row. A refresh (REF) command, as discussed earlier, refreshes DRAM cells to maintain their charge. The memory controller has to issue 8192 REFs in one refresh window (tREFW; e.g., 32 ms on DDR5 [28]). Internally, the DRAM cycles through all the rows, refreshing each one. This cycle varies from part to part, but must have refreshed all rows before any retention failures can occur.

On-Die ECC. To enhance reliability, DDR5 DRAM integrates On-Die Error Correction Code (ODECC) [28] within each chip, allowing detection and correction of single bit errors. This correction happens without the knowledge of the memory controller. Additionally, there is no mechanism provided to detect errors beyond a single bit. ODECC increases the robustness of DRAM cells against both accidental faults and deliberate disturbance-based attacks such as Rowhammer [34]. According to the JEDEC DDR5 specification [28], every 128 data bits are protected by 8 parity bits.

B. Rowhammer

Rowhammer attacks apply to modern DRAM where repeatedly accessing ("hammering") specific memory rows can induce unintended bit flips in neighboring rows [34]. This issue stems from the proximity of DRAM cells and the resulting electrical interference in densely-packed chips [3], [10], [45], [51], [76], [82]. As manufacturing processes have advanced and cell sizes shrunk, capacitors retain less charge, making them even more susceptible to Rowhammer [16], [32]. By strategically issuing memory accesses, attackers can modify data structures they normally do not have access to, achieving privilege escalation [6], [67], [74], compromising browsers [13], [19], [30], [63], [63], clouds [62], [79] and even systems over the network [40], [73].

Mitigations. To counter Rowhammer, hardware vendors have introduced Target Row Refresh (TRR) to refresh adjacent rows preemptively [16]. Despite these efforts, Rowhammer is a moving target [16], [25]. The DRAM industry has standardized mitigations termed Refresh Management (RFM), but they have yet to see adoption by CPU vendors [24]. Per-Row Activation Counting (PRAC)—a recent mitigation standardized by JEDEC—offers a more deterministic approach to Rowhammer prevention [28]. However, PRAC has yet to see widespread deployment, and research is ongoing to develop more effective and low-overhead solutions for future DRAM generations [5], [8], [43], [59], [64], [72], [75], [78].

Attack patterns. Classic Rowhammer patterns include single-sided [34], double-sided [67], and one-location hammering [18]. Half-Double [36], [42] allows an attacker to focus most of their accesses at a distance of two rows from the intended victim with a small minority targeting the immediate neighbor. To bypass TRR, research found new attack patterns such as many-sided [16] and non-uniform hammering [25]. New attack methods continue to evolve [4], [9], [26], [63], highlighting gaps in current mitigation techniques. There are also new effects, such as Rowpress [44], showing that by keeping rows open for longer, it is possible to trigger disturbance errors that have little overlap with Rowhammer errors.

C. Motivation

The difficulty of producing a cost-effective Rowhammer defense depends on the smallest number of accesses necessary to induce a Rowhammer bit flip which is also known as the

minimum Hammer Count (HCmin). To bypass mitigations successfully, the attacker has to "hide" HCmin accesses to an aggressor row from the mitigation. The DRAM substrate in DDR4 devices requires a significantly lower HCmin compared to DDR3 [16], [32], which is the main reason why the more advanced patterns can bypass the deployed mitigations [16], [25]. To address this drop of HCmin, existing mitigation proposals try to estimate the HCmin for the more recent DDR5 and future devices [48], [58], [65], [66]. This raises the question: *are existing HCmin estimations realistic when considering real DDR5 devices?*

Recent work [17], [49] experimentally evaluates the minimum number of accesses required to trigger bit flips on DDR5 devices. These results show HCmin values as high as 52 K, comparable to DDR4 [32], it is significantly higher than previous estimations [48], [58], [65], [66]. This could be explained by the fact that the flips are only observed after correction by ODECC—hiding the *true HCmin* of the DDR5 substrate in these devices. Secure Rowhammer mitigations, however, cannot rely on the weak single-bit guarantees provided by ODECC as the probability that two bit flips end in the same ECC block (i.e., 128 bits) is not negligible. Knowing the true HCmin is therefore fundamental to the secure design and the correct configuration of Rowhammer defenses, such as the number of required counters or the frequency of TRR refreshes [7], [8], [20], [23], [33], [46], [48], [55], [58]–[60], [64]–[66], [70], [77], [81]. While DRAM vendors can disable ODECC and measure the vulnerability of their devices [38], these custom features are neither public nor accessible for independent evaluation. To the best of our knowledge, the only work evaluating ODECC is BEER [57]. The main goal of BEER is to reverse engineer ODECC, including the code word layout and parity matrix. While this is useful to understand the location of errors, BEER does not allow measuring the HCmin of a device because single-bit errors are corrected by ODECC and therefore not observable. On the other hand, multi-bit errors that are required for observability, mask the true HCmin of the device. Hence, we need new techniques to be able to measure the pre-ODECC HCmin, i.e., the HCmin of DDR5 devices before ODECC.

As we summarize in Tbl. I and discuss in § VI, ***ECCclipse*** provides the first comprehensive characterization of Rowhammer on DDR5 DRAM chips of all three major DRAM manufacturers. Crucially, we uncover the true HCmin by accounting for the masking effects of ODECC. Our results show that the median HCmin across all devices is 16.1 K, which is a decrease of 8.6x compared to DDR4, less severe than the overly pessimistic drop of 10.2–51x in prior work. Finally, we demonstrate that the exaggerated HCmin estimates from previous studies impose prohibitive costs on existing mitigations [46], [60], resulting in overheads of up to 12.8x in storage, 5.2x in energy consumption, and 5x in performance.

III. Exposing Bit Flips in DDR5 DRAM

Modern DDR5 DRAM devices employ ODECC to improve data integrity and reliability [28]. ODECC corrects errors in-flight on the read path, i.e., the corrected data is returned to the memory controller on a memory read request, but the uncorrected data remains in DRAM. Once the ODECC's capacity to correct errors is exhausted, *uncorrectable* errors (and potentially miscorrected bits) become apparent to the memory controller. This behavior complicates the experimental characterization of memory disturbance effects such as Rowhammer, as it masks the earliest physical bit flips. We present two new methods that allow characterizing Rowhammer on DDR5 DRAM devices despite ODECC.

TABLE I: **Comparison of related characterization work.** We present the DRAM technology of the devices characterized in each work, the number of DRAM chips characterized per studied technology (i.e., DDR3/4/5), the lowest minimum hammer count (HCmin) reported, and whether the work measures HCmin before on-die ECC (pre-ODECC). We use *n/a* to denote non-applicability due to lack of ODECC. †Result for a single-sided Rowhammer attack.

	Year	Technology DDR 3 / 4 / 5	#DRAM Chips	Lowest HCmin	pre-ODECC
ECCclipse	2026	✘ / ✘ / ✔	192	5.1 K	✔
Phoenix [49]	2026	✘ / ✘ / ✔	80	51.6 K	✘
REFault [17]	2025	✘ / ✘ / ✔	16	32 K	✘
WhistleBlower [22]	2023	✔ / ✔ / ✘	16 / 276	30 K	*n/a*
RH Sens. [54]	2021	✔ / ✔ / ✘	24 / 248	33 K	*n/a*
U-TRR [21]	2021	✘ / ✔ / ✘	292	6 K	*n/a*
Revisiting RH [32]	2020	✔ / ✔ / ✘	408 / 652	10 K	*n/a*
Kim et al. [34]	2014	✔ / ✘ / ✘	972	139 K†	*n/a*

Assumption. Based on prior work [32], [34], [39], [44], [84] correlating data retention errors and disturbance effects such as Rowhammer and Rowpress, we assume that the majority of the errors produced by these two mechanisms are distinct. This means, there is only a small overlap of bit flips based on these mechanisms and therefore, the correlation between errors from different mechanisms is weak. We will further validate this assumption in § V-A.

Approach. We enable error *visibility* by deliberately exhausting ECC's error correction capability such that any additional bit error becomes immediately observable. For this, we induce errors in all ECC codewords in the row of interest before performing our *measurement* of HCmin. We aim to avoid introducing bit flips via the visibility method that overlap with the bit flips from the studied mechanism (i.e., Rowhammer), as these *blind* the measurement.

Overview. We developed two new methods that enable visibility of physical bit flips, *ECCfade* and *ECCstress*. These methods are based on two distinct error-inducing sources: (i) ECCfade relies on retention-induced errors (§ III-A), (ii) ECCstress uses Rowpress bit flips (§III-B). As the ODECC implementation is proprietary, our methods do not make any assumptions about the ECC scheme or codeword layout.

A. ECCfade: Creating Visibility Using Retention Errors

Our first method, ECCfade, exploits the charge leakage of DRAM cells by disabling periodic refreshes for a certain time to induce retention errors in DRAM cells. The retention time

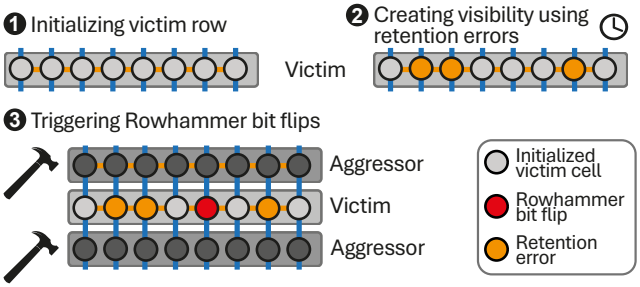


Fig. 1: **Enabling visibility using retention errors.** ❶ The victim row is initialized with a data pattern. ❷ Retention errors accumulate over time. ❸ Once visibility is achieved by sufficient bit flips, we run our measurement; e.g., determining HCmin.

of a DRAM cell is the minimum time during which a cell must be able to hold its charge before a refresh is required such that, when read, it returns the correct value. If we disable periodic refreshes long enough, to cause a sufficient number of retention errors, we can create visibility for the measurement.

Steps. Next, we explain more in detail the four steps involved in ECCfade, as illustrated in Fig. 1.

❶ **Initializing the victim.** We write a known data pattern to the victim row. To not make any assumptions about the data encoding (i.e., true-/anti-cell) of DRAM cells, we use a random data pattern or repeat the same experiment twice, with an arbitrary data pattern and its inverse.

❷ **Inducing retention errors.** We temporarily disable periodic DRAM refreshes to create retention errors. After a predetermined interval, we re-enable periodic refreshes to stop any more retention errors from emerging while preserving the induced errors. In § IV-B, we determine the optimal waiting time to induce a sufficient number of bit flips in every ECC codeword.

❸ **Capturing the error pattern.** We read the victim row to capture the error pattern induced by the retention errors. We note that the captured error pattern may differ from the data actually stored in DRAM because either some ECC codewords may remain intact such that errors in these will be corrected when read or the ECC scheme may have introduced errors through miscorrection [56], [57]. However, any additional errors observed *after* the measurement phase (e.g., performing Rowhammer for measuring HCmin) can be attributed to the measurement.

❹ **Running the measurement.** The experiment is executed, e.g., a double-sided Rowhammer attack with different hammer counts to determine the HCmin pre-ODECC. Then, the victim row is read and its data compared to the original error pattern (❸) to identify any new bit flips.

B. ECCstress: Creating Visibility Using Rowpress Bit Flips

In the ECCstress method, we exploit Rowpress to induce bit flips that create visibility beyond ODECC. Rowpress is a more recently (2023) discovered read-disturb phenomenon in DRAM chips [44], distinct from Rowhammer [34]. Using Rowpress to induce thousands of errors in a DRAM row is much faster than retention, as we show in § IV-B.

Steps. The approach follows the same four steps as the retention-based method in § III-A. In step ❶, we need to take into account that Rowpress is a data pattern-dependent effect [44]. As we do not need to find the most effective data pattern, we take a random pattern for the victim and its inverse for the aggressor rows. In step ❷, we induce Rowpress errors by disabling periodic refreshes while repeatedly pressing one of the physically adjacent rows of the victim row for a specific time, referred to as tAggON [44]. Afterwards, we re-enable periodic refreshes to avoid retention errors and to preserve the induced Rowpress errors, before continuing with steps ❸ and ❹. We determine the optimal values for the total attack time and the pressing duration (tAggON) in § IV.

TABLE II: **Comparison** of our two proposed methods.

	ECCfade (§ III-A)	**ECCstress** (§ III-B)
Underlying effect	retention errors	Rowpress
Assumptions	*none*	RowPress-vulnerable row
Requirements	stable temp, ref. ctrl.	refresh control
Parameters	wait time	tAggON, # reps.
Bit flip direction	charge → discharge	mostly 1-to-0
Affected rows	all rows	adjacent rows

C. Comparative Summary

We summarize and compare the two presented methods, ECCfade and ECCstress in Tbl. II. ECCfade's **underlying effect** are retention errors, which has the advantage that this method will always work with DRAM by design. ECCstress relies on the Rowpress disturbance mechanism. This affects the **assumptions** of the methods: ECCstress requires Rowpress-vulnerable rows, while ECCfade relies only on the default retention behavior of DRAM cells. A **requirement** of all methods is control over periodic refreshes, for example, as provided by an FPGA-based memory controller. ECCfade additionally needs a stable temperature to precisely induce a predetermined number of retention errors. The **parameters** of the methods control how many errors are to be induced. This can be controlled by the time without periodic refreshes (ECCfade); the pressing duration (tAggON) and the number of repetitions (ECCstress). The **bit flip direction** of ECCfade is charge-to-discharge. For ECCstress, 1-to-0 bit flips are dominant. The **affected rows** are either all rows at once (ECCfade) or only the physically adjacent rows (ECCstress). Affecting all rows is beneficial as it allows creating visibility for all rows in a single run.

IV. Evaluating the Visibility Mechanisms

In this section, we evaluate the two proposed methods on the DDR5 DIMMs to better understand how they perform in practice. The goal of this evaluation is to determine the optimal method and parameters to be used in our Rowhammer characterization in § V.

TABLE III: **DDR5 DIMMs Testpool.** We assign a vendor specific abbreviation to each DIMM (H for SK Hynix, M for Micron, and S for Samsung), and report the manufacturing date (Mf. Date) as yy-mm; DIMM size; the I/O width (I/O Wd.) in bits; the geometry, i.e., number of ranks (R), bankgroups (G), and banks (B); and if they are used in the evaluation (E) of the two presented methods.

ID	Model	Mf. Date	Size [GB]	I/O Wd.	Geom. (R,G,B)	E
H_1	9965784-026.A00G	22-12	32	x4	1, 2, 4	✘
H_2	9965793-020.A00G	23-05	16	x8	1, 2, 4	✘
H_3	9965788-018.A00G	23-04	64	x4	2, 2, 4	✔
H_4	KF560R32-16	24-02	16	x8	1, 2, 4	✔
H_5	KF564R32-16	23-11	16	x8	1, 2, 4	✘
H_6	KF548R36-16	24-04	16	x8	1, 2, 4	✘
H_7	HMCG78AHBRA472N	24-03	16	x8	1, 2, 4	✔
H_8	HMCG78AHBRA472N	24-03	16	x8	1, 2, 4	✘
M_1	MTC20F1045S1RC48BA2	22-09	32	x4	1, 2, 4	✘
M_2	MTC20F2085S1RC48BA1	23-08	32	x8	2, 2, 4	✘
M_3	MTC40F2046S1RC48BA1	24-05	64	x4	2, 2, 4	✘
M_4	MTC20F2085S1RC48BA1	24-03	32	x8	2, 2, 4	✔
M_5	MTC10F1084S1RC48BA1	24-07	16	x8	1, 2, 4	✔
M_6	MTC20F2085S1RC48BA1	22-09	32	x8	2, 2, 4	✔
S_1	M321R4GA3BB6-CQKVS	23-07	32	x8	2, 2, 4	✔
S_2	M321R4GA3BB6-CQKVS	23-07	32	x8	2, 2, 4	✔
S_3	M321R2GA3BB6-CQKET	22-12	16	x8	1, 2, 4	✔
S_4	M321R2GA3BB6-CQKDG	22-06	16	x8	1, 2, 4	✔
S_5	M321R2GA3EB2-CWMJJ	25-03	16	x8	1, 2, 4	✔
S_6	M321R2GA3EB2-CWMJJ	25-03	16	x8	1, 2, 4	✘

Qualities. Our evaluation considers three qualities:

(i) the parameters (§IV-A), i.e., the combination of method-specific parameters that leads to the smallest pre-ODECC HCmin estimate for all DIMMs;

(ii) the number of bit flips (§ IV-B), i.e., how does the number of induced bit flips change depending on the method-specific parameters; and

(iii) the estimation quality (§ IV-C), i.e., by how much lower the pre-ODECC HCmin is compared to the post-ODECC HCmin.

Setup. We conduct all experiments on the Antmicro RDIMM DDR5 platform [2], providing full control over DRAM commands, including refreshes. We use a temperature-controlled infrastructure with two heating pads around the DIMM to ensure a constant temperature of 50 °C during all experiments. As our prototypical platform does not support higher temperatures, our characterization is limited to this temperature.

DRAM DIMMs. Tbl. III presents the 20 DDR5 DIMMs used in this work, covering 192 DDR5 DRAM chips (80x SK Hynix, 64x Micron, and 48x Samsung). Due to the long experiment time, we only use the subset of DIMMs marked in the "E" column to evaluate the two different methods. Later, in § V, we characterize Rowhammer on all DIMMs.

A. Parameters

The presented methods use different parameters to control how many errors to induce when creating visibility. We want to know how many errors are optimal to induce in each DRAM row to ensure that the next bit flip exceeds the ECC's correction capability and immediately becomes observable by the memory controller. We use the HCmin estimation (i.e., HCmin pre-ODECC) as a metric to quantify the degree of visibility achieved by each method.

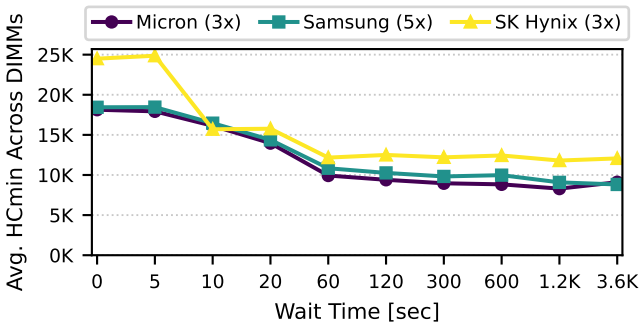


Fig. 2: **ECCfade parameters.** We show the pre-ODECC HCmin (y-axis) over all tested rows when inducing retention errors with different wait times (x-axis) while keeping periodic refreshes disabled. The HCmin across all DIMMs is the lowest with a retention time of 20 min for DIMMs from Micron (8.3 K) and SK Hynix (11.8 K); and with a retention time of 60 min for DIMMs from Samsung (8.8 K).

In the ideal case, we would have exactly one bit flip in each ECC codeword. This minimizes the bit flips that can cause blinding with the measurement while making sure that any further bit flip, in any of the ECC codewords, becomes observable. However, we have limited control over the location of induced bit flips with our two methods and do not want to make any assumption about the exact ECC codeword layout of a DRAM row. Therefore, we empirically determine the optimal number of bit flips to induce in each ECC codeword to get the lowest pre-ODECC HCmin estimate.

Experiment. We sweep the method-specific parameters over 64 rows while determining the number of induced errors after each run. The sweeped parameters are, the wait time for ECCfade (10 steps, from 0 s to 60 m); and the pressing duration (8 steps, from 36 ns to 32 ms) and number of pressing repetitions (5 steps, from 100 to 150 tREFWs) for ECCstress.

After creating visibility, we use a similar methodology as in previous work [32] to determine the pre-ODECC HCmin. We perform a double-sided Rowhammer attack starting from an initial hammer count of 1 000 and increase it by 100, up to 100 000, until we observe the first bit flips in the victim row. These bit flips happen on top of the bit flips already induced by the visibility mechanism. We report the resulting hammer count as the pre-ODECC HCmin.

Results. We present the results for the ECCfade method in Fig. 2. The plot shows for each DRAM vendor, the estimated pre-ODECC HCmin (y-axis) for different wait times (x-axis) with periodic refreshes disabled. Each line shows the average HCmin across all tested DIMMs of one vendor.

The plot shows that the HCmin strongly decreases up to a wait time of 60 s for all three vendors. This can be explained by the increasing number of retention errors, leading to more visibility, while not proportionally increasing the blinding effect as the overlap between retention errors and Rowhammer bit flips is negligible, as we show in § V-A. We can also see

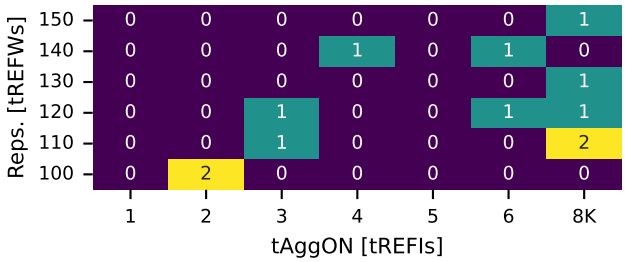


Fig. 3: **ECCstress parameters.** We show for different no. of repetitions (y-axis) and pressing durations (x-axis), the no. of DIMMs achieving their lowest estimated pre-ODECC HCmin.

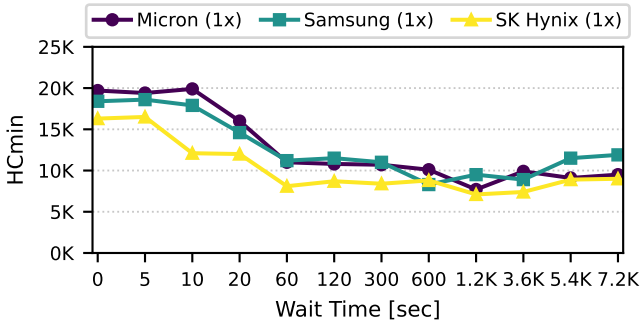


Fig. 4: **ECCfade extended wait period.** We show for three representative DIMMs (S_1, M_5, and H_7) that wait periods of 90 min and 2 h increase the lowest measured HCmin.

that a wait time of 20 min leads overall to the lowest estimated pre-ODECC HCmin for all three vendors, i.e., Micron (8.3 K), Samsung (9.1 K), and SK Hynix (11.8 K). On only two of five DIMMs (Samsung), a longer wait time of 60 min leads to a HCmin that is between 10.1% (S_1) and 15% (S_2) lower. Therefore, a wait time of 20 min optimally balances retention errors to create visibility without causing excessive blinding while keeping the overall experiment time reasonable. This is our first **observation (O)**:

O1. ECCfade's optimal wait time across DIMMs is 20 min.

We demonstrate how ECCfade performs with a longer wait period. For this, we repeat the same ECCfade experiment with extended wait periods of 90 and 120 minutes on one representative DIMM of each vendor (S_1, M_5, and H_7). In Fig. 4, we present the results. On all three tested DIMMs, the 6x longer wait period of 120 min increases the HCmin from 9 500 to 11 900 (20.17%, S_1), from 7 700 to 9 500 (18.95%, M_5), and from 7 100 to 9 000 (21.11%, H_7). This is likely because inducing an excessive number of retention errors may increase the chance of overlapping with the weakest cell. We will show later (§ IV-B) that a wait time of 20 min already induces a large amount of bit flips, on average 2 602 bit flips per row. As a wait time of 20 min better trades-off experiment time and accuracy while reducing the risk of overlapping with weak cells, we chose this wait time for all future experiments.

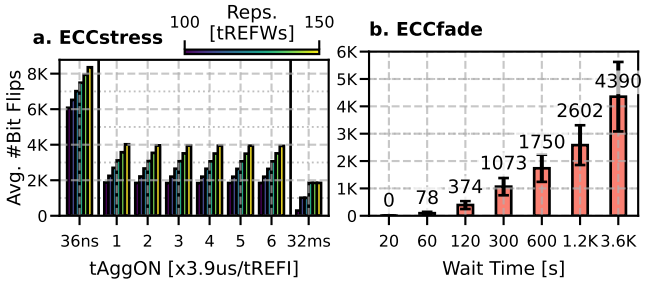


Fig. 5: **Number of bit flips.** We show the average number of bit flips over all DIMMs induced by (a) ECCstress, and (b) ECCfade over time, expressed as hammer count, row-open time (tAggON), and wait time, respectively, when creating visibility using these methods.

O2. A higher wait time in ECCfade not always improves visibility but increases the risk of covering weak cells.

In Fig. 3, we show the results for ECCstress. The plot shows for different combinations of tAggON, which is equal to the row-open time, and pressing repetitions, how many DIMMs achieve their lowest estimated pre-ODECC HCmin. We can see that there is no common optimal parameter setting across DIMMs, making this method difficult to use on a large scale of devices without prior sweeping over all possible parameter combinations on each device.

O3. ECCstress does not have common optimal parameters.

B. Number of Bit Flips for Creating Visibility

The underlying physical effect of the two error-inducing mechanisms is fundamentally different and therefore, they induce a different number of bit flips. To understand their parameters, we determine the number of bit flips when measuring the pre-ODECC HCmin with each method.

Experiment. We use the same experiment as for the parameter evaluation in § IV-A. We determine the pre-ODECC HCmin while varying the method-specific parameters for ECCstress (tAggON and repetitions) and ECCfade (retention time).

Results. We present the results for (a) ECCstress and (b) ECCfade in Fig. 5. For each method, we report the average number of bit flips across all tested rows and DIMMs.

For ECCstress in Fig. 5a, we observe that for a tAggON between 3.9 µs (i.e., 1x tREFI) and 32.4 µs (i.e., 6x tREFI), the number of induced bit flips for the same number of repetitions only varies slightly, between 38 (for 110 reps.) and 70 bit flips (for 120 reps.). To contrast, Rowhammer (tAggON of 36 ns, equal to tRFC), induces roughly two to three times more bit flips as ECCstress. Changing the number of pressing repetitions, however, results in up to ≈ 2 K more bit flips, for example, when changing the number of repetitions from 100 to 150 with a tAggON of 3.9 µs. If we keep the row open for longer time (32 ms), the number of induced bit flips drops to between 411 (for 100 reps.) and 2 352 bit flips (for 130 reps.).

O4. ECCstress shows little effect when varying the tAggON in the range of 1x tREFI to 6x tREFI.

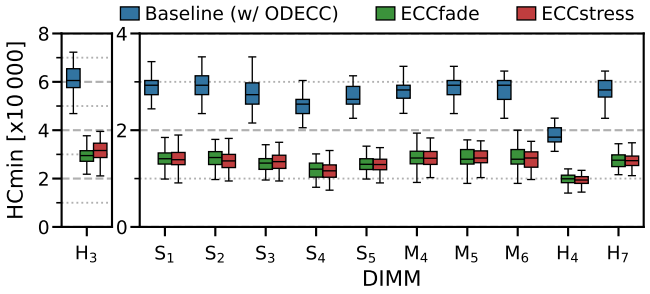


Fig. 6: **Estimation quality.** We show the post-ODECC HCmin (baseline) over all rows relative to the pre-ODECC HCmin as determined by the two different methods, ECCstress and ECCfade. We generally consider an estimate leading to a lower HCmin as better.

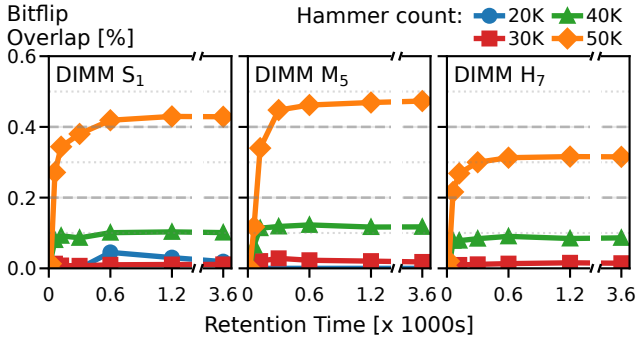


Fig. 7: **Bit flip overlap of retention and Rowhammer** for different retention times (between 5 s and 60 min) and hammer counts (between 20 K and 50 K) for one representative DIMM of each vendor, Samsung (S_1), Micron (M_5), and SK Hynix (H_7). The results show that the overlap generally increases with higher hammer counts but is never exceeding 0.5% for hammer counts up to 50 K.

ECCfade in Fig. 5b shows that we cannot induce any bit flips for retention times lower-or-equal to 20 s. Further, we can see that this method very flexibly allows to induce a specific number of bit flips. With the retention time of 20 min, which we determined to be optimal choice across vendors in § IV-A, we can induce around 2 602 bit flips on average per row. If we assume that retention errors are uniformly random distributed as assumed by existing work [41], this would already induce significantly more than the required single bit flip per codeword. For example, considering a x8 DDR5 DIMM, then a 8 KiB row consisting of 512 ECC codewords of each 128 bit contains roughly 5.1 retention errors per code word, five times more than needed to create visibility.

O5. We need to disable periodic DRAM refreshes for at least 20 s to induce any retention errors.

C. Estimation Quality (HCmin)

We evaluate the quality of the HCmin estimate relative to the real physical HCmin. Our comparison is *relative* between the two methods, as we cannot determine the actual physical HCmin; however, we assume that a method is better if it leads to a lower HCmin. We further compare the HCmin estimate to the post-ODECC HCmin to quantify by how much the method reduces the measured hammer count.

Experiment. We run the same experiment as in § IV-A while recording the pre-ODECC HCmin for each method.

Results. Fig. 6 presents the results for the baseline (i.e., post-ODECC HCmin) and the two methods for the DIMMs of our test pool (Tbl. III). The ECCfade and ECCstress methods achieve similar results, roughly halving the HCmin of the baseline (by 47.76–54.32%). While ECCstress slightly outperforms ECCfade, the difference of the four DIMMs where the HCmin by ECCstress is lower, is negligible, and at most 3.47% lower. We conclude from this that both methods are suitable for measuring the pre-ODECC HCmin.

O6. The ECCfade and ECCstress methods perform similarly well in estimating the pre-ODECC HCmin.

D. Conclusion

We conclude from our evaluation that ECCfade is the more suitable method for characterizing Rowhammer on DDR5 DRAM devices. Unlike ECCstress, the method does not require any complex DIMM-specific parameters, which facilitates testing many devices. It allows inducing a flexible number of bit flips by choosing an appropriate wait time. Further, it enables creating visibility in all rows of a DIMM at once, while ECCstress requires pressing each row individually. Lastly, the pre-ODECC estimate by ECCfade is in most cases the lowest or very close (max. 3.47% higher) to the lowest estimate. Next, we use ECCfade to characterize the HCmin of our DIMMs.

V. Characterizing HCmin of DDR5 DIMMs

In this section, we characterize Rowhammer using ECCfade. First, we show that retention- and Rowhammer-induced bit flips exhibit negligible overlap (§ V-A). Next, we determine the number of rows required for a reliable HCmin estimate (§ V-B) and characterize Rowhammer susceptibility (i.e., pre-ODECC HCmin) (§ V-C). Finally, we analyze the distribution of bit flips across byte offsets, which is important for mounting Rowhammer attacks (§ V-F).

A. Bit Flip Overlap: Retention vs. Rowhammer

The assumption of the ECCfade method is that rows with weak data retention and vulnerable to Rowhammer are mostly distinct. This is important because bits flipped while creating visibility would otherwise occlude the weakest bits (i.e., flipping at HCmin) that we want to measure. Existing work on DDR4 showed that the overlap between retention errors and Rowhammer bit flips is negligibly small [44], [50]. To show that this assumption still holds for DDR5, and ECCfade produces reliable results, we quantify the overlap.

Experiment. We perform two experiments on three representative DIMMs, one per vendor: S_1 (Samsung), M_5 (Micron), and H_7 (SK Hynix), and perform our experiments over 64 rows per DIMM. In the first experiment, we write in every row a (different) random data pattern. Thereafter, we induce

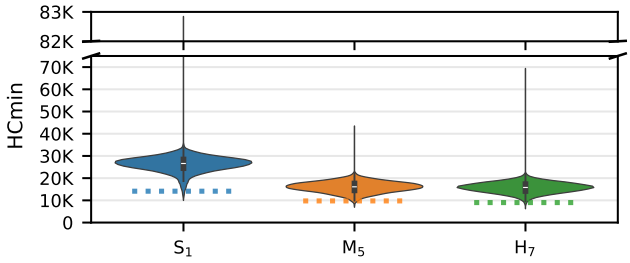


Fig. 8: **Number of profiled rows to find HCmin.** We show the HCmin distribution across 4 K rows for one representative DIMMs from each vendor. We mark the 25th percentile of the HCmin values of each DIMM with a dashed line.

retention errors by waiting between 5 s and 60 min while keeping automatic refreshes disabled. We then record the error retention pattern including the exact locations of the bit flips. In the second experiment, we write in every row a (different) random data pattern and in its immediate neighbor rows the inverted data pattern. We then perform double-sided Rowhammer with hammer counts between 10 K and 50 K, in steps of 10 K, while recording the locations of the induced bit flips. At the end, we compare the bit flip locations of the two experiments to determine the overlap.

Results. The result in Fig. 7 shows that the overlap between retention errors and Rowhammer-induced bit flips increases with the hammer count. While the overlap is relatively stable between 0–0.1% for hammer counts of up to 40 K, independent of the retention time, the overlap increases more notably for a hammer count of 50 K; however, it never exceeds 0.5%.

O7. The bit flip overlap between retention and Rowhammer errors is small ($< 0.5\%$) for HCs of up to 50 K.

B. Number of Rows for HCmin Estimation

An important parameter for the characterization of the Rowhammer vulnerability (HCmin), is the number of rows to profile. On the one side, if too few rows are tested, the measured HCmin might not truthfully represent the actual vulnerability of the device. On the other side, testing all the rows of a device is impractical due to the long experiment time, easily taking several days to weeks per device. Therefore, we design a preliminary experiment to determine how many rows we need to profile to get a reliable estimate of HCmin.

Experiment. We run our experiment on three representative DIMMs, S_1 (Samsung), M_5 (Micron), and H_7 (SK Hynix). We initialize each row with a random data pattern and use ECCfade with 20 min wait time to create visibility. Thereafter, we determine the HCmin of 4 K rows of the first bank of each DIMM by following the same procedure as in § IV-A.

Results. We present the result for the three tested DIMMs in Fig. 8. The HCmin follows a similar trend as reported previously [17], [32]: there are very few rows with a HCmin close to the device's minimum, while most of the rows have a HCmin that is roughly 2.06–2.36x higher than the minimum.

As there are very few rows with a HCmin equal to the device's minimum, we focus on the lower 25th percentile, which is visually marked in Fig. 8 with a dashed line for each DIMM. We find that 0.3–0.6% of the 4 K rows that we tested fall in this HCmin range. If we assume that we want to find at least one row in this range with a probability of 95%, we need to profile between 498 and 976 rows, depending on the DIMM's HCmin distribution. Therefore, we decide to take 1 000 rows per DIMM for our large-scale characterization in § V-C to get a reliable estimate of the device's HCmin.

O8. Testing 1 000 rows is sufficient to get a reliable estimate of a device's vulnerability (HCmin).

C. Minimum Hammercount (HCmin)

We now characterize the Rowhammer susceptibility of the 20x DDR5 DIMMs listed in Tbl. III, covering all three major DRAM vendors (6x Samsung, 6x Micron, and 8x SK Hynix).

We use the ECCfade method with a wait time of 20 min to create visibility. This means, we induce retention errors such that any additional bit flip triggered by Rowhammer immediately becomes observable (i.e., uncorrectable by ECC). Thereafter, we determine the minimum hammer count (HCmin) for each row following the same procedure as in § IV-A. This gives us the HCmin of each row with an accuracy of ± 100.

We profile 1 000 rows per DIMM over the first bank. As we showed (§ V-B), this is sufficient to reliably estimate a device's vulnerability level, i.e., the lowest HCmin across all rows.

Data patterns. Similar to previous work [32], we consider five data patterns: 0x00, 0xFF, 0x55, 0xAA, and a fully random (per row) pattern. We write the same data pattern to both aggressor and victim rows as the victim's data pattern will change after retention errors have been induced.

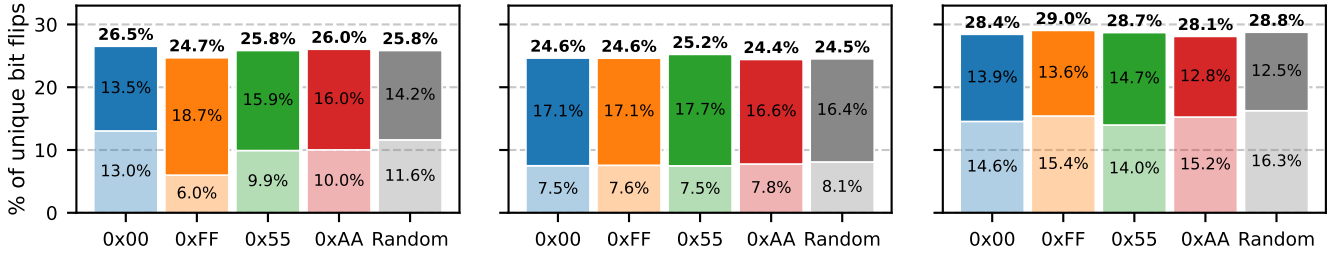
Results. Fig. 10 presents the HCmin distribution across the rows of all tested DIMMs. The median HCmin over all tested DIMMs is 16 100. The median HCmin per vendor is 14 600 for Micron, 16 000 for SK Hynix, and 21 800 for Samsung.

O9. The median HCmin across all tested devices is 16.1 K.

We can see that devices from Samsung (S) and SK Hynix (H) can each be divided into two groups with similar HCmin values. By comparing the manufacturing dates of the devices (Tbl. III), we can see that the Samsung devices with a roughly 1.5x lower HCmin (S_5 and S_6) were produced more recently (2025) than the others (2022-2023). Similarly, the SK Hynix devices with a roughly 2x lower HCmin (H_4–H_8) were produced later (end of 2023 and 2024) than the more vulnerable devices (end of 2022, early 2023).

O10. The HCmin of DDR5 devices produced between 2022 and 2025 has decreased by up to 2x.

From a mitigation point of view, the absolute HCmin for each device is more relevant. The lowest HCmin we ever observed across all devices is 5 100, belonging to S_5 from Samsung. The lowest HCmin for devices from Samsung, Micron, and SK Hynix are 5 100, 6 900, and 6 500, respectively. We will discuss the implications of these findings in § V-H.



(a) SK Hynix DIMM (H_2). (b) Micron DIMM (M_4). (c) Samsung DIMM (S_4).

Fig. 9: **Bit flips across data patterns.** We show for each tested data pattern (x-axis) of three representative DIMMs, how many of the total unique bit flips can be attributed to the data pattern (y-axis). We distinguish between bit flips triggered by at least one other data pattern (bottom bar) and exclusive to this data pattern (top bar).

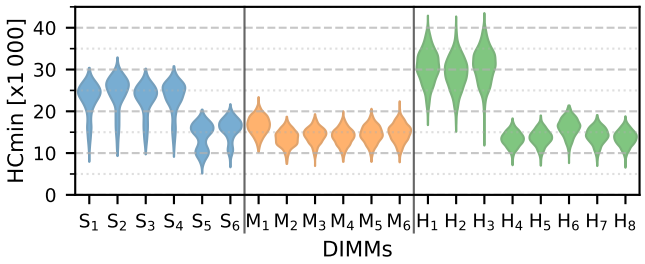


Fig. 10: **HCmin** distribution over 1 000 rows and 20 DIMMs.

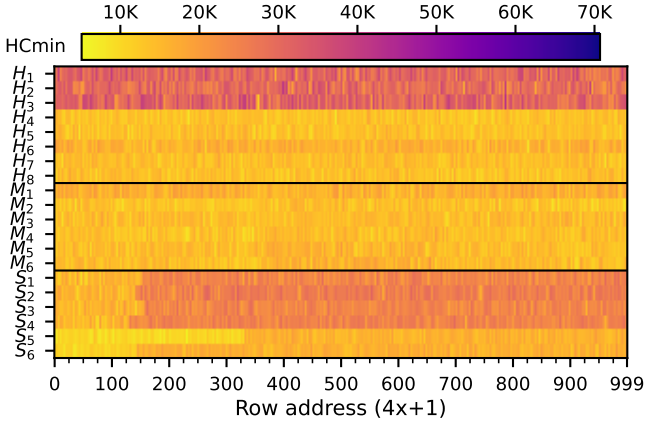


Fig. 11: **Spatial HCmin distribution.** We show for all tested DDR5 DIMMs (y-axis), the spatial distribution of the HCmin (color) across the 1 000 tested rows (x-axis). A darker color indicates a higher HCmin, while a lighter color indicates a lower HCmin. As we skip three rows between two consecutively tested rows, the absolute row address can be calculated by $4x + 1$ where x is the x-axis value.

There is a large variation in the HCmin values across the tested rows of each device. This observation is particularly pronounced on devices from Samsung where the weakest row and the strongest row can differ by up to 4x in terms of their HCmin (S_5). More general, we can see that the differences of all DIMMs range between 3.11–4.0x for Samsung, 2.27–2.87x for Micron, and 2.57–3.69x for SK Hynix.

O11. The HCmin difference between the weakest and strongest row of a device can be up to 4.0x.

D. Bit Flip Distribution Across Data Patterns

Knowing which data pattern is most effective is useful for Rowhammer attacks as it can help to reduce the *templating* phase of an attack where the attacker looks for exploitable bit flips. Therefore, it reduces the overall attack time.

Fig. 9 shows the distribution of bit flips across the five tested data patterns for one representative DIMM per vendor. The difference between the most and least effective pattern on all tested DIMMs is between 0.3–3.0%. Across all tested DIMMs and data patterns, 12.2–23.3% of all bit flips are exclusively triggered by one data pattern only. This suggests that testing multiple data patterns is required to find the HCmin.

O12. We must test different data patterns to find the HCmin.

This is also confirmed by our HCmin data which shows that finding the HCmin of our DIMMs required different data patterns: 0x55 ($S_{3,6}$, $M_{3,4}$, $H_{3,5,6}$), 0x00 ($S_{1,4,5}$, $M_{2,6}$), random ($M_{1,5}$, $H_{7,8}$), 0xFF (S_2, H_2), and 0xAA ($H_{1,4}$)

E. Bit Flip Distribution Across Rows

The distribution of HCmin values across tested DIMMs (see § V-C) show that there is a large variation across rows. This raises the question: *is there any spatial correlation between the HCmin values of a device, for example, created by the physical layout of the DRAM cells?*

In Fig. 11, we present a color-encoded heatmap of the HCmin values (per row) for all tested DIMMs. We can see that the SK Hynix devices H_1, H_2, and H_3 have higher HCmin values than the other SK Hynix devices. Generally, the HCmin values appear randomly distributed across rows without any clear pattern except for Samsung devices: the first 125-150 tested rows (i.e., row addresses 501-601) have a notably lower HCmin than the remaining tested rows. This could be explained by variations that are either design-induced or caused by the manufacturing process.

O13. The HCmin values are mostly randomly distributed across rows, except for Samsung devices which exhibit lower HCmin values in the first tested rows in the bank.

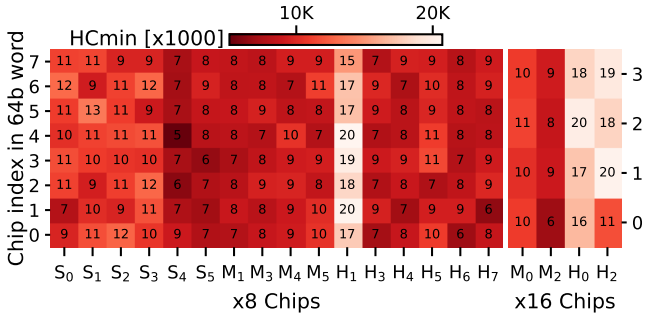


Fig. 12: **HCmin distribution of DRAM chips** for the DIMMs with x8 and x16 chips in our test pool (thousands of accesses).

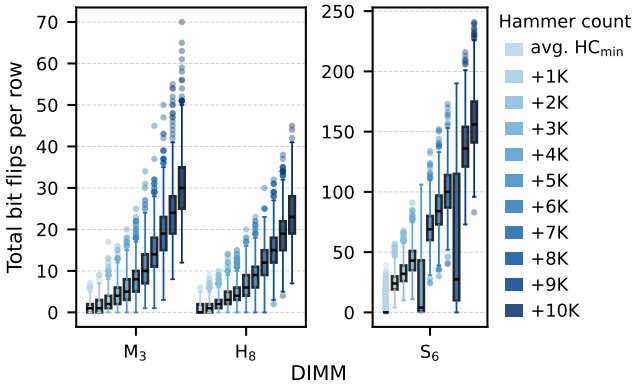


Fig. 13: **Bit flip growth over hammer counts.** For three representative DIMMs, we show the total number of bit flips for different hammer counts over 1 000 rows by starting from the DIMM's average HCmin and incrementing it in steps of 1 K up to 10 K.

F. Bit Flip Distribution Across Chips

We want to find out if different chips in a DDR5 device have different HCmin values. If true, this opens up interesting opportunities for configuring Rowhammer mitigations differently per chip, depending on their vulnerability.

Fig. 12 reports the HCmin for each DRAM chip of each device. These results show that the HCmin differences across chips of the same DIMM can be drastic in some cases. For example, on H_2, where this effect is strongest, the weakest chip is 1.82x more vulnerable than the least vulnerable one. Similar trends also exist for other devices (e.g., S0 and S4).

O14. The HCmin difference across same-DIMM chips of DDR5 devices can vary by up to 1.8x.

Overall, the difference between the weakest and strongest chip is 1.10–1.82x over all devices, averaging at 1.43x. These results show that there is an opportunity for configuring mitigations differently on chips of the same device to save on unnecessary mitigation operations that consume power.

G. Bit Flip Rate for Different Hammer Counts

We aim to understand how the number of bit flips grows with increasing hammer counts. Understanding this relation is important as it affects the exploitability of devices. To study this, we determine the average HCmin over 1 000 rows using ECCfade. Then, we again perform ECCfade followed by a double-sided Rowhammer attack with hammer counts starting from the average HCmin and increasing in ten steps of 1 K. We repeat this on all 1 000 rows and count the total number of bit flips for each hammer count.

The results in Fig. 13 show that the number of bit flips for devices M_3 and H_8 grows quadratically with increasing hammer counts. On S_6, however, this trend is not as strong as for two hammer counts (+4 K and +8 K), the median number of bit flips significantly drops, while the overall maximum number of bit flips per row always increases. For every 1 K additional hammers, the median number of bit flips increases by 1–2x (M_3), 1.15–1.34x (S_6), and 1.21–2.0x (H_8).

O15. Each additional 1 K hammers results in an average 1.4x increase in the median bit-flip count per victim row.

This means that attackers can significantly increase the number of bit flips by increasing the hammer count, which can overwhelm additional protections such as rank-level ECC as we show in § V-H.

H. Discussion

Existing mitigation proposals estimate the degree of vulnerability in future DRAM devices [48], [58], [65], [66]. Some of these proposals [58]) suggest that HCmin will be much lower than 4.8 K for devices produced after 2020. Some others use the worsening of the Rowhammer vulnerability to motivate the need for mitigations that can handle ultra-low HCmin values, expected to fall below 100 by the end of this decade [66].

HCmin trend. To better understand the worsening Rowhammer susceptibility, we put earlier characterization results of DDR3 [32], [34] and DDR4 [32], [45], [80] into perspective with our DDR5 results. We take from each existing work, the lowest HCmin reported among DIMMs of the same manufacturing year. We fit a trend line using polynomial interpolation through these minimum HCmin values to show how the HCmin could evolve. We also take the pessimistic predictions from existing Rowhammer mitigations [53], [58], [61], [65], [66], [68], [75] to show their predicted trend. Fig. 14 shows that, while HCmin has worsened since DDR4 was introduced (2016), the picture is significantly less dramatic than painted by existing mitigation proposals. Comparing the HCmin of the first DDR4 DIMMs (2016) from the newer technology node [32] with the most recent DDR5 DIMMs (2025), we can see that the HCmin has decreased by only 8.6x (from 43.8 K to 5.1 K) over nearly a decade.

O16. While Rowhammer has worsened in DDR5, the HCmin has only decreased by 8.6x since DDR4 (2016).

Predictions from existing work suggest that HCmin would drop to between 500 and 100 in future devices. Our lowest HCmin value (5.1 K) for DDR5 devices produced in 2025, is still 10.2–51x higher than predicted, more than an order of magnitude difference. If this trend continues, the HCmin is not expected to drop below 1 K until roughly 2032.

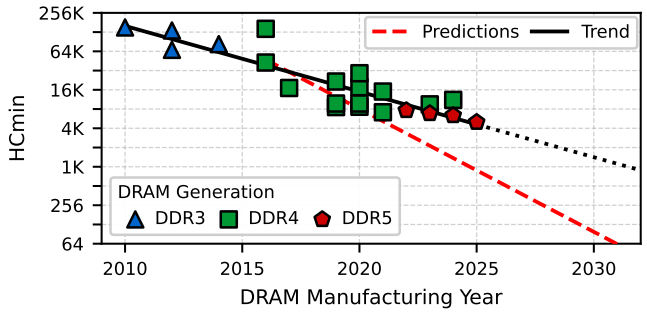


Fig. 14: **HCmin trend** over the DRAM manufacturing years, comparing our results on DDR5 with previous work on DDR3 [32], [34] and DDR4 [32], [45], [80]. The dashed line shows the pessimistically painted HCmin trend derived from existing work [53], [58], [61], [65], [66], [68], [75].

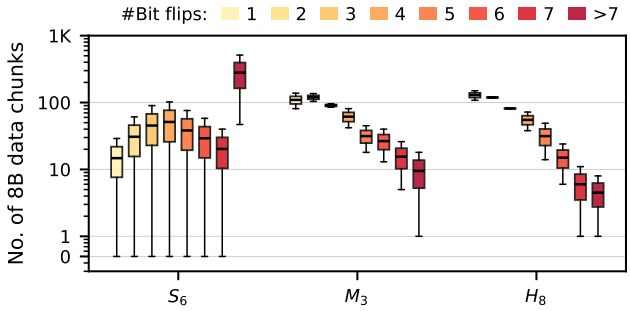


Fig. 15: **Bit flip frequency per 8-byte data chunk (log y-axis)** by varying bit flip counts (bars) across three repr. DIMMs (x-axis). Results are based on hammering for HCmin and HCmin + 1000.

O17. The HCmin predictions from existing work are overestimating the HCmin drop by 10.2–51x.

Mitigations. For properly configuring Rowhammer mitigations, knowledge of the Rowhammer susceptibility level (i.e., HCmin) is crucial to set the mitigation parameters (e.g., number of counters, frequency of mitigative refreshes) according to the device's vulnerability such that it can effectively prevent Rowhammer attacks. Overly pessimistic HCmin predictions, however, can lead to severe overprovisioning of resources, thus causing unnecessary area, performance, and energy overheads.

To illustrate the impact of poor HCmin predictions, we compare the mitigation overhead of the prediction (900) to the value following the actual trend (4 700) from 2025 (see Fig. 14) for three recent mitigations, ProTRR [46], MINT [60], and ABACuS [53]. We extract the results for the ProTRR overheads from the plots and inter-/extrapolate both the MINT [71] (energy and performance) and ABACuS overheads (area and energy). The results show that the storage overhead for ProTRR is 12.8x higher for the predicted HCmin (235 KB) than for the actual HCmin (18 KB). Similarly, MINT's energy and performance overheads are 5.2x (9.11% instead of 1.74%) and 5x (6.25% instead of 1.25%) higher than necessary, respectively. For ABACuS, the area overhead is 5.2x higher (21.04 KB vs 4.03 KB), and equally higher are the static (3.2x) and dynamic (1.3x) energy overheads.

O18. Inaccurate HCmin estimates lead to large area, energy, and performance overheads in Rowhammer mitigations.

On-Die ECC. While error correction provided by ODECC appears to increase the hammer count as single bit flips remain invisible without ECCfade, ODECC is primarily a reliability mechanism targeted at increasing manufacturing yields, and not supposed to fix Rowhammer. Because Rowhammer can produce multiple flips that are spatially close, they can easily bypass the ECC protection for a single codeword. Our data shows that 92.97% of all profiled rows suffer by two to six bit flips per row at their pre-ODECC HCmin. Considering this, results in a probability of 0.274% that two bit flips fall into the exact same 128-bit codeword. This means that testing 1 679 rows suffices to finding two bit flips in the same codeword with a 99% probability.

O19. The majority of rows (92.97%) suffer by more than one bit flip per row at their HCmin.

Rank-level ECC. We analyze whether existing rank-level ECC would be effective in preventing Rowhammer bit flips on our tested devices. Fig. 15 shows for three representative DIMMs, the number of 8-byte data chunks with a certain number of bit flips. A common linear correction scheme is SECDED ECC [69] which can correct up to one bit flip per 8-byte chunk. Given our data, this translates to 5.2% (S_6), 6.2% (M_3), and 15.1% (H_8) of all 8-byte data chunks with bit flips being uncorrectable (i.e., 2-bit errors). Symbol-based codes are an alternative to linear codes and provide stronger protection against DRAM chip failures. For example, Chipkill [14] can correct all errors in one symbol, which would correspond to a failed DRAM chip. In our case, however, we can see that with Chipkill the percentage of uncorrectable errors increases on two of the three DIMMs to 9.4% (M_3) and 21.4% (S_6), while it decreases to 3.8% (H_8) on the other DIMM. This means that deploying rank-level ECC on top of the existing on-die ECC is not effective to completely protect against Rowhammer on our DDR5 DRAM devices given the high number of bit flips.

O20. The protection provided by rank-level ECC against Rowhammer is insufficient, leaving 5.2–15.1% (SECDED) and 3.8–21.4% (Chipkill) of all bit flips uncorrectable.

VI. Related Work

The related work can be categorized into: (i) characterization of bit flips induced by Rowhammer(-like) effects under diverse operating conditions; (ii) bypass of memory controller-based ECC mechanisms; and (iii) Rowhammer characterization studies targeting DDR5 memories.

Bit flip characterization. Previous extensively studied Rowhammer bit flips in DDR3 and DDR4 DRAM devices, as summarized in Tbl. I (p. 3). Kim et al. [34] demonstrated disturbance errors in DDR3 modules through repeated row activations, thereby establishing the Rowhammer phenomenon. TRRespass [16] exposed weaknesses in Target Row Refresh

(TRR), the in-DRAM mitigation. Building on this, Kim et al. [32] conducted a large-scale analysis of 1 500 chips across DDR3, DDR4, and LPDDR4, revealing that many DRAM devices remain vulnerable despite TRR. Hassan et al. [21] proposed the U-TRR method to uncover how DRAM chips internally protect against Rowhammer by exploiting the natural charge loss (retention) of memory cells. They used custom Rowhammer patterns combined with variations in refresh timing to reveal which rows receive extra protection based on their retention characteristics. He et al. [22] conducted a system-level study regarding the factors affecting Rowhammer bit flips, such as data patterns, instruction sequence, and multi-threading. REFault [17] used fault injection to temporarily suppress refreshes and enable measuring the post-ODECC HCmin of two DDR5 devices. In Phoenix [49], besides reverse engineering TRR, they also characterized the post-ODECC HCmin of 10 DDR5 SK Hynix DIMMs using an FPGA platform. Other works have studied related memory disturbance effects [44], [83] and bit flips under environmental influences such as temperature [54] and wordline voltage [80].

Bypassing ECC. Although ECC memory is commonly assumed to protect against soft errors, it does not defend against bit flips intentionally induced by adversaries [11], [29], [37]. ECCploit [11] showed that standard SECDED (Single Error Correction, Double Error Detection) schemes can fail to detect or correct certain multi-bit errors crafted by attackers, demonstrating that ECC alone is insufficient against deliberate fault injection. RAMBleed [37] revealed that Rowhammer-induced bit flips can be used as a side channel to leak adjacent memory contents, such as RSA keys. These prior studies focus on (rank-level) ECC as implemented in the memory controller. In contrast, the DDR5 devices in our study are equipped with on-die ECC that is implemented in the DRAM chips themselves.

Rowhammer on DDR5 DRAM. Despite DDR5's enhanced mitigations—including TRR, on-die ECC, and higher refresh rates—recent work demonstrates its continued vulnerability to Rowhammer attacks [26], [49]. Zenhammer [26] provides the first empirical evidence of Rowhammer-induced bit flips on DDR5, though only one device exhibited bit flips using fuzzing, suggesting that these mitigations make attacks significantly harder but not impossible. Phoenix [49] advances this work, demonstrating successful bit flips on all 15 tested SK Hynix DDR5 DIMMs and achieving the first privilege escalation exploit on DDR5 systems using self-correcting refresh synchronization—a technique that automatically realigns the attack pattern upon detecting missed refresh operations. Together, these works demonstrate that DDR5's protective mechanisms, while raising the bar for attacks, fail to eliminate the fundamental Rowhammer threat.

VII. Conclusion

Measuring the minimum number of accesses for triggering Rowhammer bit flips (i.e., HCmin) is crucial for a correct configuration of Rowhammer mitigations. On-Die ECC is commonplace in DDR5 devices, however, it mask the bit flips in cells that are the most vulnerable to Rowhammer, making the measurement of HCmin challenging on these devices. We described two new techniques circumventing ODECC with the aim of measuring HCmin in commodity DDR5 devices. These techniques rely on compromising ODECC in ways that do not mask bit flips in cells that are the weakest with respect to Rowhammer. Our results show that due to signifcant errors in estimating HCmin, existing Rowhammer mitigations report up to 12.8x, 5.2x, and 5x higher storage, energy, and performance overheads, respectively, than necessary to protect the most vulnerable DDR5 devices.

References

[1] T. A. Alawneh, A. A. M. Sharadqh, A. Alsharah, E. Awada, J. S. Alkasassbeh, A. Y. Al-Rawashdeh, and A. Al-Qaisi, "A Highly Parallel DRAM Architecture to Mitigate Large Access Latency and Improve Energy Efficiency of Modern DRAM Systems," *IEEE access*, pp. 1–1, 2024.

[2] Antmicro AB, "Data Center RDIMM DDR5 Tester · Antmicro/DDR5-Tester@rev.1.0.1-Production," Jul. 2022.

[3] S. Baeg, D. Yun, M. Chun, and S.-J. Wen, "Estimation of the Trap Energy Characteristics of Row Hammer-Affected Cells in Gamma-Irradiated DDR4 DRAM," *IEEE Transactions on Nuclear Science*, vol. 69, no. 3, pp. 558–566, Mar. 2022.

[4] S. Baek, M. Wi, S. Park, H. Nam, M. J. Kim, N. S. Kim, and J. H. Ahn, "Marionette: A RowHammer Attack via Row Coupling," in *ASPLOS '25*. Rotterdam Netherlands: ACM, Mar. 2025, pp. 637–652.

[5] T. Bennett, S. Saroiu, A. Wolman, L. Cojocar, and A.-G. Llc, "Panopticon: A Complete In-DRAM Rowhammer Mitigation," in *DRAMSec '21*, 2021, p. 7.

[6] M. Bölcskei, P. Jattke, J. Wikner, and K. Razavi, "Rubicon: Precise microarchitectural attacks with page-granular massaging," in *EuroS&P '25*. IEEE, 2025.

[7] F. N. Bostanci, I. E. Yüksel, A. Olgun, K. Kanellopoulos, Y. C. Tuğrul, A. G. Yağliçi, M. Sadrosadati, and O. Mutlu, "CoMeT: Count-min-sketch-based row tracking to mitigate RowHammer at low cost," in *HPCA '24*. IEEE, 2024, pp. 593–612.

[8] O. Canpolat, A. G. Yağlıkçı, G. F. Oliveira, A. Olgun, N. Bostancı, İ. E. Yüksel, H. Luo, O. Ergin, and O. Mutlu, "Chronus: Understanding and Securing the Cutting-Edge Industry Solutions to DRAM Read Disturbance," Feb. 2025.

[9] W. Chen, S. Tang, Y. Tang, X. Luo, Y. Zhang, and W. Qiang, "ρHammer: Reviving RowHammer Attacks on New Architectures via Prefetching," in *MICRO '25*. Seoul Republic of Korea: IEEE/ACM, Oct. 2025, pp. 1433–1447.

[10] Y. Cohen, K. S. Tharayil, A. Haenel, D. Genkin, A. D. Keromytis, Y. Oren, and Y. Yarom, "HammerScope: Observing DRAM Power Consumption Using Rowhammer," in *CCS '22*. Los Angeles CA USA: ACM, Nov. 2022, pp. 547–561.

[11] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting Correcting Codes: On the Effectiveness of Ecc Memory Against Rowhammer Attacks," in *S&P '19*. IEEE, 2019, pp. 55–71.

[12] F. de Ridder, P. Frigo, E. Vannacci, H. Bos, C. Giuffrida, and K. Razavi, "SMASH: Synchronized Many-Sided Rowhammer Attacks from JavaScript," in *USENIX Security '21*. USENIX Association, 2021.

[13] F. de Ridder, P. Jattke, and K. Razavi, "Softhammer: Exploiting Rowhammer Bit Flips without Crashing," in *DRAMSec '25*, 2025.

[14] T. J. Dell, "A white paper on the benefits of chipkill-correct ecc for pc server main memory," *IBM Microelectronics division*, vol. 11, no. 1-23, pp. 5–7, 1997.

[15] P. Frigo, C. Giuffrida, H. Bos, and K. Razavi, "Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU," in *S&P '18*. IEEE, 2018.

[16] P. Frigo, E. Vannacc, H. Hassan, V. Van Der Veen, O. Mutlu, C. Giuffrida, H. Bos, and K. Razavi, "TRRespass: Exploiting the Many Sides of Target Row Refresh," in *S&P '20*. IEEE, 2020, pp. 747–762.

[17] S. Gloor, P. Jattke, and K. Razavi, "REFault: A Fault Injection Platform for Rowhammer Research on DDR5 Memory," in *uASC '25*, Bochum, Germany, 2025.

[18] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another Flip in the Wall of Rowhammer Defenses," in *S&P '18*. IEEE, May 2018, pp. 245–261.

[19] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript," in *DIMVA '16*. Springer, 2016.

[20] A. Hajiabadi, M. Marazzi, and K. Razavi, "CHaRM: Checkpointed and hashed counters for flexible and efficient rowhammer mitigation," in *CCS '25*. ACM, 2025.

[21] H. Hassan, Y. C. Tugrul, J. S. Kim, V. Van der Veen, K. Razavi, and O. Mutlu, "Uncovering in-DRAM Rowhammer Protection Mechanisms: A New Methodology, Custom Rowhammer Patterns, and Implications," in *MICRO '21*. IEEE/ACM, 2021, pp. 1198–1213.

[22] W. He, Z. Zhang, Y. Cheng, W. Wang, W. Song, Y. Gao, Q. Zhang, K. Li, D. Liu, and S. Nepal, "WhistleBlower: A System-level Empirical Study on RowHammer," *IEEE Transactions on Computers*, pp. 1–15, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10014649/

[23] A. Jaleel, G. Saileshwar, S. W. Keckler, and M. Qureshi, "PrIDE: Achieving Secure Rowhammer Mitigation with Low-Cost in-DRAM Trackers," in *ISCA '24*. ACM/IEEE, 2024.

[24] P. Jattke, F. Solt, M. Marazzi, M. Wipfli, S. Gloor, and K. Razavi, "McSee: Evaluating Advanced Rowhammer Attacks and Defenses via Automated DRAM Traffic Analysis," in *USENIX Security '25*. USENIX Association, 2025.

[25] P. Jattke, V. van der Veen, P. Frigo, S. Gunter, and K. Razavi, "BLACKSMITH: Scalable Rowhammering in the Frequency Domain," in *S&P '22*. IEEE, May 2022.

[26] P. Jattke, M. Wipfli, F. Solt, M. Marazzi, M. Bölcskei, and K. Razavi, "ZenHammer: Rowhammer Attacks on AMD Zen-based Platforms," in *USENIX Security '24*. USENIX Association, Aug. 2024.

[27] JEDEC Solid State Technology Association, "JESD79-5A: Double Data Rate 5 (DDR5) SDRAM," 2020.

[28] JEDEC Solid State Technology Association, "JESD79-5C: Double Data Rate 5 (DDR5) SDRAM," 2024.

[29] N. Kamadan, W. Wang, S. van Schaik, C. Garman, D. Genkin, Y. Yarom, and R. Model, "Ecc.fail: Mounting Rowhammer Attacks on DDR4 Servers with ECC Memory," in *USENIX Security '25*. USENIX Association, 2025.

[30] I. Kang, W. Wang, J. Kim, S. van Schaik, Y. Tobah, D. Genkin, A. Kwong, and Y. Yarom, "SledgeHammer: Amplifying Rowhammer Via Bank-Level Parallelism," in *USENIX Security '24*. USA: USENIX Association, 2024.

[31] D. Kaseridis, J. Stuecheli, and L. K. John, "Minimalist Open-Page: A DRAM Page-Mode Scheduling Policy for the Many-Core Era," in *MICRO '11*. Porto Alegre, Brazil: ACM/IEEE, 2011, p. 24.

[32] J. S. Kim, M. Patel, A. G. Yaglikci, H. Hassan, R. Azizi, L. Orosa, and O. Mutlu, "Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques," in *ISCA '20*. Valencia, Spain: ACM/IEEE, May 2020, pp. 638–651.

[33] M. J. Kim, J. Park, Y. Park, W. Doh, N. Kim, T. J. Ham, J. W. Lee, and J. H. Ahn, "Mithril: Cooperative Row Hammer Protection on Commodity DRAM Leveraging Managed Refresh," in *HPCA '22*. IEEE, 2022, pp. 1156–1169.

[34] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," in *ISCA '14*. Minneapolis, MN, USA: ACM/IEEE, Jun. 2014, pp. 361–372.

[35] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA '12*. ACM/IEEE, 2012, p. 12.

[36] A. Kogler, J. Juffinger, S. Qazi, Y. Kim, M. Lipp, N. Boichat, E. Shiu, M. Nissler, and D. Gruss, "Half-Double: Hammering From the Next Row Over," in *USENIX Security '22*. USENIX Association, Aug. 2022, pp. 3807–3824.

[37] A. Kwong, D. Genkin, D. Gruss, and Y. Yarom, "Rambleed: Reading Bits in Memory Without Accessing Them," in *S&P '20*. IEEE, 2020, pp. 695–711.

[38] S. Lee, N.-H. Lee, KW. Lee, JH. Kim, JH. Jin, YS. Lee, YC. Hwang, HS. Kim, and S. Pae, "Development and Product Reliability Characterization of Advanced High Speed 14nm DDR5 DRAM with On-die ECC," in *2023 IEEE International Reliability Physics Symposium (IRPS)*, Mar. 2023, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10117889

[39] C. Lim, K. Park, G. Bak, D. Yun, M. Park, S. Baeg, S.-J. Wen, and R. Wong, "Study of Proton Radiation Effect to Row Hammer Fault in DDR4 SDRAMs," *Elsevier Microelectronics Reliability*, vol. 80, pp. 85–90, Jan. 2018.

[40] M. Lipp, M. Schwarz, L. Raab, L. Lamster, M. T. Aga, C. Maurice, and D. Gruss, "Nethammer: Inducing Rowhammer Faults through Network Requests," in *EuroS&P Workshops '20*. IEEE, Sep. 2020, pp. 710–719.

[41] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms," *SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 60–71, Jun. 2013.

[42] G. LLC, "Half-Double: Next-Row-Over Assisted Rowhammer," Google LLC, Tech. Rep., May 2021.

[43] S.-L. Lu, J. Woo, and P. J. Nair, "Counterpoint: One-Hot Counting for PRAC-Based RowHammer Mitigation," in *DRAMSec '25*, 2025.

[44] H. Luo, A. Olgun, A. G. Yağlıkçı, Y. C. Tuğrul, S. Rhyner, M. B. Cavlak, J. Lindegger, M. Sadrosadati, and O. Mutlu, "RowPress: Amplifying read disturbance in modern DRAM chips," in *ISCA '23*. Orlando, FL, USA: ACM/IEEE, Jun. 2023, pp. 1–18.

[45] H. Luo, İ. E. Yüksel, A. Olgun, A. G. Yağlıkçı, and O. Mutlu, "Revisiting DRAM Read Disturbance: Identifying Inconsistencies Between Experimental Characterization and Device-Level Studies," in *VTS '25*. IEEE, 2025, pp. 1–8.

[46] M. Marazzi, P. Jattke, F. Solt, and K. Razavi, "ProTRR: Principled yet Optimal In-DRAM Target Row Refresh," in *S&P '22*. IEEE, 2022.

[47] M. Marazzi, T. Sachsenweger, F. Solt, P. Zeng, K. Takashi, M. Yarema, and K. Razavi, "HiFi-DRAM: Enabling High-Fidelity DRAM Research by Uncovering Sense Amplifiers with IC Imaging," in *ISCA '24*. ACM/IEEE, Jul. 2024.

[48] M. Marazzi, F. Solt, P. Jattke, K. Takashi, and K. Razavi, "REGA: Scalable Rowhammer Mitigation with Refresh-Generating Activations," in *S&P '23*. IEEE, May 2023, pp. 1684–1701.

[49] D. Meyer, P. Jattke, M. Marazzi, S. Qazi, D. Moghimi, and K. Razavi, "Phoenix: Rowhammer Attacks on DDR5 with Self-Correcting Synchronization," in *S&P*, May 2026.

[50] O. Mutlu and J. S. Kim, "RowHammer: A Retrospective," *IEEE TCAD '20*, vol. 39, no. 8, pp. 1555–1571, Aug. 2020.

[51] H. Nam, S. Baek, M. Wi, M. J. Kim, J. Park, C. Song, N. S. Kim, and J. H. Ahn, "DRAMScope: Uncovering DRAM microarchitecture and characteristics by issuing memory commands," in *ISCA '24*. ACM/IEEE, Jun. 2024, pp. 1097–1111.

[52] A. Olgun, F. N. Bostanci, G. F. Oliveira, Y. C. Tugrul, R. Bera, A. G. Yaglikci, H. Hassan, O. Ergin, and O. Mutlu, "Sectored DRAM: An Energy-Efficient High-Throughput and Practical Fine-Grained DRAM Architecture," Aug. 2022.

[53] A. Olgun, Y. C. Tugrul, N. Bostanci, I. E. Yuksel, H. Luo, S. Rhyner, A. G. Yaglikci, G. F. Oliveira, and O. Mutlu, "ABACuS: All-Bank Activation Counters for Scalable and Low Overhead RowHammer Mitigation," in *USENIX Security '24*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1579–1596.

[54] L. Orosa, A. G. Yaglikci, H. Luo, A. Olgun, J. Park, H. Hassan, M. Patel, J. S. Kim, and O. Mutlu, "A deeper look into RowHammer's sensitivities: Experimental analysis of real DRAM chips and implications on future attacks and defenses," in *MICRO '21*. IEEE/ACM, 2021, pp. 1182–1197.

[55] Y. Park, W. Kwon, E. Lee, T. J. Ham, J. H. Ahn, and J. W. Lee, "Graphene: Strong yet Lightweight Row Hammer Protection," in *MICRO '20*. IEEE/ACM, 2020, pp. 1–13.

[56] M. Patel, J. S. Kim, H. Hassan, and O. Mutlu, "Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices," in *DSN '19*. Portland, OR, USA: IEEE/IFIP, Jun. 2019, pp. 13–25.

[57] M. Patel, J. S. Kim, T. Shahroodi, H. Hassan, and O. Mutlu, "Bit-Exact ECC Recovery (BEER): Determining DRAM on-Die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *MICRO '20*. Athens, Greece: IEEE/ACM, Oct. 2020, pp. 282–297.

[58] M. Qureshi, "AutoRFM: Scaling Low-Cost in-DRAM Trackers to Ultra-Low Rowhammer Thresholds," in *HPCA '25*. IEEE, 2025.

[59] M. Qureshi and S. Qazi, "MOAT: Securely Mitigating Rowhammer with Per-Row Activation Counters," in *ASPLOS '25*, ser. ASPLOS '25. New York, NY, USA: ACM, Mar. 2025, pp. 698–714.

[60] M. Qureshi, S. Qazi, and A. Jaleel, "MINT: Securely Mitigating Rowhammer with a Minimalist in-DRAM Tracker," in *MICRO '24*. IEEE/ACM, 2024.

[61] M. Qureshi, A. Rohan, G. Saileshwar, and P. J. Nair, "Hydra: Enabling low-overhead mitigation of row-hammer at ultra-low thresholds via hybrid tracking," in *ISCA '22*. New York New York: ACM/IEEE, Jun. 2022, pp. 699–710.

[62] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip Feng Shui: Hammering a Needle in the Software Stack," in *USENIX Security '16*. USENIX Association, 2016.

[63] F. Ridder, P. Jattke, and K. Razavi, "Posthammer: Pervasive Browser-Based Rowhammer Attacks with Postponed Refresh Commands," in *USENIX Security '25*. Seattle, WA, USA: USENIX Association, Aug. 2025.

[64] A. Saxena, A. Jaleel, and M. Qureshi, "ImPress: Securing DRAM Against Data-Disturbance Errors via Implicit Row-Press Mitigation," in *MICRO '24*. IEEE/ACM, Nov. 2024, pp. 935–948.

[65] A. Saxena, S. Mathur, and M. Qureshi, "Rubix: Reducing the Overhead of Secure Rowhammer Mitigations via Randomized Line-to-Row Mapping," in *ASPLOS '24*. La Jolla CA USA: ACM, Apr. 2024, pp. 1014–1028.

[66] A. Saxena and M. Qureshi, "START: Scalable Tracking for any Rowhammer Threshold," in *HPCA '24*. Edinburgh, United Kingdom: IEEE, Mar. 2024, pp. 578–592.

[67] M. Seaborn and T. Dullien, "Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges," Mar. 2015.

[68] A. Singh and B. Panda, "Rowhammer Cache: A Last-Level Cache for Low-Overhead Rowhammer Tracking," in *HOST '24*. Tysons Corner, VA, USA: IEEE, May 2024, pp. 349–360.

[69] C. W. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 397–404, 2005.

[70] M. Son, H. Park, J. Ahn, and S. Yoo, "Making DRAM Stronger Against Row Hammering," in *DAC '17*. ACM, 2017, pp. 1–6.

[71] H. Taneja, A. Hajiabadi, M. Marazzi, K. Razavi, and M. Qureshi, "Mirza: Efficiently mitigating rowhammer with randomization and alert," in *IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE*, 2026.

[72] H. Taneja and M. Qureshi, "DREAM: Enabling Low-Overhead Rowhammer Mitigation via Directed Refresh Management," in *ISCA '25*. Tokyo Japan: ACM/IEEE, Jun. 2025, pp. 776–792.

[73] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "Throwhammer: Rowhammer Attacks over the Network and Defenses," in *ATC '18*. USENIX Association, 2018.

[74] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *CCS '16*. ACM, Oct. 2016.

[75] S. Vittal, S. Qazi, P. Das, and M. Qureshi, "MoPAC: Efficiently Mitigating Rowhammer with Probabilistic Activation Counting," in *ISCA '25*. Tokyo Japan: ACM/IEEE, Jun. 2025, pp. 723–738.

[76] A. J. Walker, S. Lee, and D. Beery, "On DRAM Rowhammer and the Physics of Insecurity," *IEEE Transactions on Electron Devices*, pp. 1–11, 2021.

[77] J. Woo, S. C. Lin, P. J. Nair, A. Jaleel, and G. Saileshwar, "QPRAC: Towards Secure and Practical Prac-Based Rowhammer Mitigation Using Priority Queues," in *HPCA '25*. IEEE, 2025, pp. 1021–1037.

[78] J. Woo, J. Qu, G. Saileshwar, and P. J. Nair, "When Mitigations Backfire: Timing Channel Attacks and Defense for PRAC-Based RowHammer Mitigations," in *ISCA '25*. Tokyo Japan: ACM/IEEE, Jun. 2025, pp. 739–756.

[79] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation," in *USENIX Security '16*. USENIX Association, 2016.

[80] A. G. Yağlıkçı, H. Luo, G. F. De Oliviera, A. Olgun, M. Patel, J. Park, H. Hassan, J. S. Kim, L. Orosa, and O. Mutlu, "Understanding rowhammer under reduced wordline voltage: An experimental study using real dram devices," in *DSN '22*. IEEE/IFIP, 2022, pp. 475–487.

[81] A. G. Yağlikçi, M. Patel, J. S. Kim, R. Azizi, A. Olgun, L. Orosa, H. Hassan, J. Park, K. Kanellopoulos, T. Shahroodi *et al.*, "BlockHammer: Preventing RowHammer at low cost by blacklisting rapidly-accessed DRAM rows," in *HPCA '21*. IEEE, 2021.

[82] T. Yang and X.-W. Lin, "Trap-Assisted DRAM Row Hammer Effect," *IEEE Electron Device Letters*, vol. 40, no. 3, pp. 391–394, Mar. 2019.

[83] I. E. Yuksel, A. Olgun, N. Bostanci, H. Luo, A. G. Yaglikci, and O. Mutlu, "Columndisturb: Understanding column-based read disturbance in real dram chips and implications for future systems," in *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture*, 2025, pp. 975–994.

[84] L. Zhou, S. Ye, R. Wang, and Z. Ji, "Unveiling RowPress in Sub-20 nm DRAM Through Comparative Analysis With Row Hammer: From Leakage Mechanisms to Key Features," *IEEE Transactions on Electron Devices*, pp. 1–8, 2024.