Counterpoint: One-Hot Counting for PRAC-Based RowHammer Mitigation

Shih-Lien Lu[1], Jeonghyun Woo[2], and Prashant J. Nair[2]

[1]School of EECS, Washington State University
[2]Dept. of ECE, The University of British Columbia

Dynamic Random-Access Memory (DRAM) continues to scale to smaller nodes to increase memory capacity. However, at sub-20nm scales, repeated activations of adjacent rows can cause unintended charge leakage in neighboring cells, resulting in the well-known security vulnerability known as RowHammer. To mitigate this, the JEDEC DDR5 standard introduces Per-Row Activation Counting (PRAC), which enables precise tracking of row activations by adding a dedicated counter to each DRAM row. PRAC allows DRAM to request mitigation from the memory controller when any counter exceeds a critical threshold.

While effective, the current PRAC implementation incurs significant performance overhead. Each row activation requires a Read-Modify-Write (RMW) operation to update the corresponding counter, leading to substantial timing delays and slowdowns of up to 18%. Previous work has attempted to alleviate this by decoupling counter updates and leveraging subarray-level parallelism to hide the overhead. However, these designs suffer from high area costs due to the added latches, wires, and control logic. To address this, we propose a high-performance PRAC implementation based on One-Hot Encoding, which eliminates RMW operations, ensures precise activation counting, and reduces hardware complexity, thereby significantly lowering both performance and implementation overheads compared to existing solutions.

1. Introduction

As DRAM technology continues to scale down, it faces increasing challenges related to security and reliability. Reduced cell sizes exacerbate charge leakage and electrical interference, making modern DRAM more susceptible to data integrity issues [1]. One of the most notable vulnerabilities is RowHammer, a read-disturbance phenomenon in which repeatedly activating a DRAM row can induce bit flips in physically adjacent rows, even without directly accessing them [2, 3].

The Per Row Activation Counting (PRAC) mechanism in the JESD79-5C DDR5 SDRAM standard enhances RowHammer defense by tracking row activations at wordline granularity [4]. When activation thresholds are exceeded, PRAC enables DRAM to alert the system, pause memory traffic, and initiate mitigation measures [5, 6]. This improves tracking accuracy and coordination between DRAM and the system. Additionally, PRAC employs a back-off protocol via the ALERTn signal to delay new commands and prevent bit flips.

Implementing PRAC introduces timing changes primarily related to row activation tracking and refresh mechanisms. PRAC requires additional logic within the DRAM to monitor row activations, which can impose timing overhead on each access. When a row's activation count exceeds a threshold, PRAC pauses command issuance from the memory controller, triggering a back-off protocol via the ALERTn signal. This delays incoming commands to enable refreshing vulnerable rows. While effective in mitigating RowHammer, this approach incurs significant performance and energy overhead [7, 8].

Ideally, one would achieve the security benefits of PRAC without incurring its performance overhead. Several strategies have been explored to accomplish this. One such approach is multi-level counting, where frequently accessed rows are tracked with fine granularity, while less active rows are monitored with coarser, aggregated counters. Although not yet applied to PRAC, the hybrid design in Hydra [7] uses this method within memory controllers. A second approach reduces the frequency of activation counter updates by exploiting locality in the memory access pattern. For example, Perfect RowHammer Tracking (PRHT) [9] proposes using a small storage in the memory controller to accumulate activations to recently activated rows. This approach updates the corresponding activation counters only when they are evicted from memory. A third approach, Chronus [10], seeks to eliminate counter updates from the critical path by decoupling counter updates to minimize latency. Unfortunately, these techniques either encounter pathological scenarios [11], lack precision, or require significant changes to the DRAM interface or micro-architecture.

We observe that a more direct approach to PRAC-based RowHammer mitigation in DRAM is to avoid imposing an *explicit* read-modify-write operation while updating the per-row counts. To this end, we propose a novel one-hot encoding scheme for per-row PRAC counters, paired with an innovative DRAM sub-array architecture that requires minimal modifications. This new design allows counter updates to occur directly within the local sense amplifier of the sub-array.

2. Background

2.1. DRAM Read Cycle Timing

The DRAM local sense amplifier plays a crucial role in reading and amplifying the small voltage differences that result from charge sharing between the memory cell and the bitline. A short overview of the phases involved in the DRAM read cycle, pictorially in Figure 2, is as follows:

1. **Precharge Phase**: Before data access, the bit-line and its complementary bit-line (bit-line#) are pre-charged and equalized to a reference voltage, typically half of the core array supply voltage ($\frac{V_{cca}}{2}$), ensuring a stable starting point.
2. **Activation Phase**: When a row is accessed, the word-line is activated (driven high to the word-line voltage), allowing charge from the storage capacitor to affect the bit-line.
3. **Sensing and Amplification**: The sense amplifier detects the slight voltage difference between the bit-line and bit-

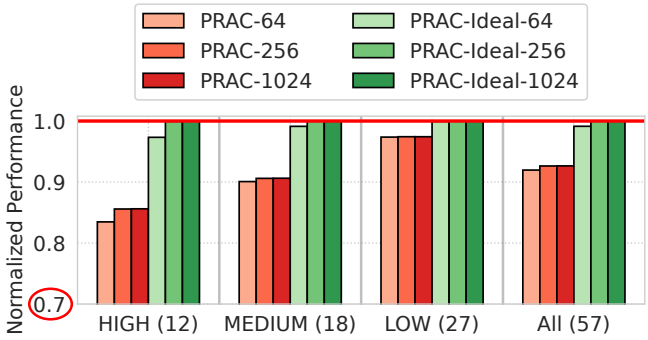

Figure 1: Normalized performance of the current PRAC design and an ideal variant (PRAC-Ideal) that avoids Read-Modify-Write (RMW) operations, evaluated at N_{RH} values of 64, 256, and 1024 (see Section 5 for details). PRAC incurs up to 16.7% slowdown due to RMW-induced timing overhead, while PRAC-Ideal shows only 2.7% overhead at N_{RH} of 64 and almost no overhead at higher thresholds. These results demonstrate that RMW operations are the primary source of performance slowdowns. This paper aims to eliminate this bottleneck by proposing a high-performance PRAC design with an alternative counting mechanism that avoids RMWs.

line#, amplifying them to either the core supply voltage (V_{cca}) or ground (V_{ssa}) levels, corresponding to '1' and '0'.

4. **Data Readout**: The amplified signal is then sent to the output circuitry for further processing and delivery to the requesting unit.
5. **Restoration**: Since DRAM uses capacitive storage and performs destructive reads, the sensed data is rewritten back to the memory cell to maintain data integrity.
6. **Precharge Reset**: After the operation, the bit-lines are reset to prepare for the next memory access cycle.

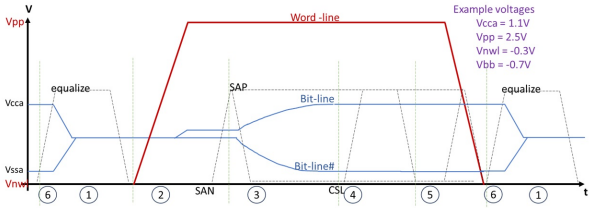

Figure 2: DRAM Access Signals and Timings.

2.2. Restrictions and Overheads of PRAC

The DRAM local sense amplifier detects and amplifies the slight voltage differences resulting from charge sharing between the cell's capacitor and the bit-line capacitance. This charge sharing disrupts the original content of the memory cell, requiring restoration of the charge in each activated cell. As a result, these cells cannot share a sense amplifier, and the sense amplifier layout must align precisely with the bit-line pitch, limiting potential modifications to the area.

In the case of current PRAC designs, the counter bits are stored within the subarray and typically need to be read out to global logic for incrementing, which introduces additional latency. After incrementing, the PRAC counter values must be updated within the subarray, which causes additional write delays and increases activation latency. This Read-Modify-Write operation is the primary source of PRAC's performance overhead. Moreover, integrating the global incrementing logic directly into individual subarrays is challenging due to strict area and bitline placement constraints. Any design changes must take these limitations into account.

This paper develops a new counting algorithm and design that can be integrated with the DRAM subarray sense amplifier while respecting these placement restrictions. The additional circuitry required for incrementing the count bits introduces minimal overhead and can be accommodated *within* the sense amplifier area. This approach eliminates the need to read, increment the count externally, and then write it back, simplifying the process and reducing latency.

3. Count Without Explicit Read-Modify-Writes

3.1. Leveraging the Simplicity of One-Hot Encoding

Binary One-Hot Encoding is a method for representing information using a bit vector in which only one bit is set to '1', while all others are '0'. In a one-hot encoded counter, counting up corresponds to a right shift, and counting down corresponds to a left shift. For example, Figure 3 shows a 4-bit one-hot counter initialized to all '0's. On each cycle (or access), a constant '1' is shifted from the leftmost position through the counter bits. The counter saturates when all bits are set to '1'. A 4-bit counter can count up to 5, and in general, an N-bit one-hot counter can count up to N+1. These counter bits are reset to '0' during each auto-refresh. In the context of PRAC, if this counting method is applied within the sense amplifiers for counter bits, the counter is "incremented" (shifted) each time a row is activated. The updated counter is then written back during the restoration phase. The changes required to implement this in the system will be discussed in Section 4.1.

initial value	0	0	0	0	*0*	
shift 1 from L	1	0	0	0	*1*	
	1	1	0	0	*2*	
	1	1	1	0	*3*	
	1	**1**	**1**	**1**	*4*	count saturated
						count to 5 (N+1)

Figure 3: 4-bit One-Hot Binary Counter

3.2. Extending Counts with One-Hot Encoding

Basic binary one-hot encoding with an N-bit counter can count up to N+1. To count to 500, 499 counting bits are needed per row. In a typical DRAM array with 8K columns per row, this results in approximately 6% overhead. To reduce this, we modify the one-hot counter to extend the count range to 2N using an N-bit counter. From a sense amplifier perspective, this modification requires adding only one extra transistor per bit-line near the sense amplifier. The enhanced encoding supports both left and right shifts. Figure 4 shows this approach with a 4-bit counter that counts up to 8.

4. The Proposed Design

Our design leverages the fact that the primary function of one-hot counting is shifting. This operation is achieved using pass transistors in conjunction with sense amplifiers. More

initial value	0	0	0	0	*0*	
shiftR 1 from L once	1	0	0	0	*1*	
shift 0	0	1	0	0	*2*	
	0	0	1	0	*3*	
	0	0	0	1	*4*	**see 1 @ R most bit, shiftL 1 from R**
	0	0	1	1	*5*	
	0	1	1	1	*6*	
	1	1	1	1	*7*	**count saturated**
						count 2N

Figure 4: Modified 4-bit One-Hot Binary Counter

importantly, we eliminate the need to integrate a conventional 'increment' block within the sense amplifiers, enabling a practical implementation. The shifting process redirects data from the sense amplifier to adjacent bit-lines by controlling pass transistors, which act as multiplexers. Instead of writing the amplified value back to the cell on the existing bit line, the pass transistors transfer the current count bit to the cell on the adjacent bit line.

4.1. Circuit Design for One-Hot Counting

To implement counting using basic one-hot encoding, we propose adding two pass transistors per bit-line. Figure 5 shows the abstract circuit diagram with four bit-lines and their corresponding complementary bit-line-bars. In this diagram, the blue circle with a cross represents a pass transistor. These pass-transistor pairs are controlled by complementary "Control" signals at the bottom. The control signals either restore the current bit-line (for refresh) or shift the data to the adjacent bit-line. For example, with the control signals configured as shown, a '1' is shifted into the counter from the bottom edge, and the remaining counter bits shift to the next position.

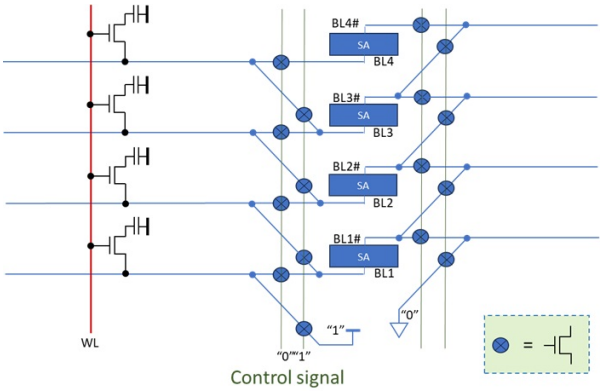

Figure 5: Circuit to Perform Basic One-Hot Counting.

4.2. Timing for the Proposed Counting Mechanism

The proposed design efficiently moves data across neighboring bit-lines without requiring additional complex circuitry. This makes it particularly beneficial for tightly packed memory architectures. Furthermore, shifting (counting) occurs naturally during the restoration phase, with new control signals activated as shown in Figure 6 during phase 5. Since all cells in the activated row are equalized first, only a small delta voltage is generated during phase 3. Modern sense amplifiers also employ isolation techniques to decouple the bit-line from the sensing process, so overwriting the existing value does not present a significant challenge for the sense amplifier.

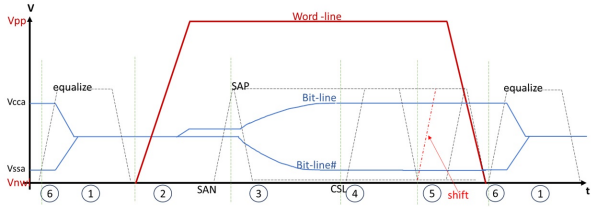

Figure 6: New DRAM Access Signals and Timings.

4.3. Circuit Design for Extended One-Hot Counting

As discussed in section 3.2, we propose a modification to reduce the overhead of adding counting bits to the DRAM array, allowing counting to 2N with an N-bit counter. This requires adding a pass-transistor per bit-line to enable shifting in the opposite direction. Figure 7 illustrates a 4-bit counter example, where control signals are set to shift a '0' from the top into the counter bits. The other bits shift down (left in our example), with each bit moving to position $i - 1$.

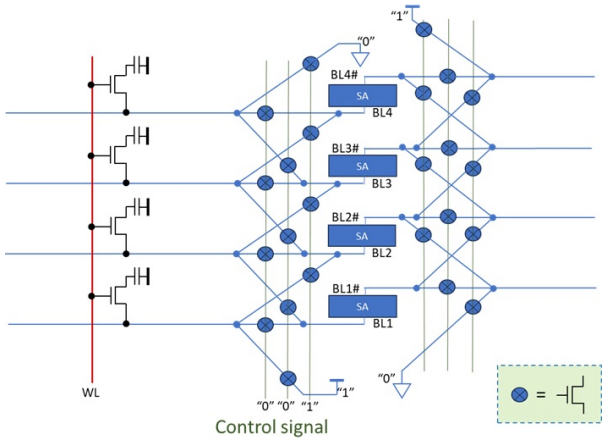

Figure 7: Modified Circuit to Allow Shifting in Both Directions.

5. Evaluation Methodology

Simulation Framework: We evaluate the performance overhead of the proposed One-Hot Counting mechanism and other PRAC designs using the cycle-accurate, trace-based DRAM simulator Ramulator2 [12, 13]. The simulation employs an internal Out-of-Order core model, consistent with prior RowHammer research [5, 10, 11, 14–18]. The system configuration is shown in Table 1. We simulate a 4-core processor with an 8MB shared Last-Level Cache (LLC) and 128GB of DRAM. The memory controller uses Minimalist Open-Page (MOP) address mapping [19] and a First Ready First Come First Served (FR-FCFS) scheduler [20], with a cap of 4 [21], as used in prior work [5, 10, 11]. The memory system consists of a single-channel, quad-rank 32Gb DDR5-8000B chip, with timing parameters from the latest JEDEC DDR5 specification (JESD79-5C, April 2024) [4], including all PRAC-specific timing modifications.

Evaluated Designs: We evaluate the performance of One-Hot Counting alongside three PRAC variants against a baseline

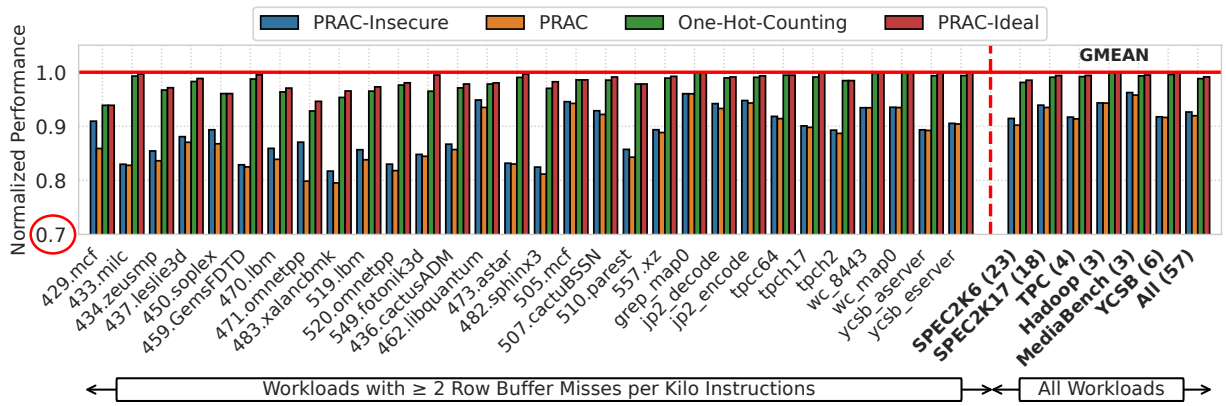

Figure 8: Normalized performance of One-Hot Counting and three PRAC designs compared to a baseline DDR5 system without PRAC at a RowHammer Threshold (NRH) of 64. PRAC-Insecure and PRAC incur significant average slowdowns of 7.4% and 8%, respectively, primarily due to the timing overhead of Read-Modify-Write (RMW) operations required to update activation counters. In contrast, One-Hot Counting incurs only a 1.1% slowdown, closely matching PRAC-Ideal's 0.9% slowdown, despite a conservative 20% increase in tRAS, by effectively eliminating RMW overhead through shift-based counting.

Table 1: System Configuration

Out-of-Order Cores	4 Cores, 4GHz, 4-wide issue/retire, 352-entry ROB
Last Level Cache (Shared)	8MB, 8-way, 64B lines
Address Mapping Scheduling Policy	Minimalist Open-Page (MOP) [19] FR-FCFS [20, 22] with a cap of 4 [21]
Memory Type DRAM Organization Rows Per Bank, Size	DDR5-8000, 32Gb chip, x8 device 4 Bank x 8 Groups x 4 Ranks x 1 Channel 128K, 8KB

Table 2: Workload Categorization Based on RBMPKI

RBMPKI	Workloads
High [10+)	429.mcf, 433.milc, 434.zeusmp, 437.leslie3d, 450.soplex, 459.GemsFDTD, 470.lbm, 471.omnetpp, 483.xalancbmk, 519.lbm, 520.omnetpp, 549.fotonik3d
Medium [1, 10)	436.cactusADM, 462.libquantum, 473.astar, 482.sphinx3, 505.mcf, 507.cactuBSSN, 510.parest, 557.xz, grep_map0, jp2_decode, jp2_encode, tpcc64, tpch17, tpch2 wc_8443, wc_map0, ycsb_aserver, ycsb_eserver
Low [0, 1)	401.bzip2, 403.gcc, 435.gromacs, 444.namd, 445.gobmk, 447.dealII, 456.hmmer, 458.sjeng, 464.h264ref, 481.wrf, 500.perlbench, 502.gcc, 508.namd, 531.deepsjeng, 511.povray, 523.xalancbmk, 525.x264, 526.blender, 538.imagick, 541.leela, 544.nab, h264_encode, tpch6, ycsb_abgsave, ycsb_bserver, ycsb_cserver, ycsb_dserver

DDR5 system without PRAC. To do this, we extend Ramulator2 to model PRAC components, including per-row activation counters and the Alert Back-Off (ABO) protocol. The evaluated designs are: 1) **PRAC-Insecure**: A PRAC design that excludes the ABO protocol, isolating and quantifying the performance overhead due solely to Read-Modify-Write (RMW) operations for updating activation counters on each row activation. 2) **PRAC**: A secure PRAC design with both per-row activation counters and the ABO protocol. We use QPRAC [5] as our secure PRAC design, as it is a proven solution for secure PRAC [5, 6]. 3) **One-Hot Counting**: Our proposed method eliminates RMW operations by using a shifting-based counting scheme during the restoration phase. This avoids additional RMW overhead but slightly extends the restoration time (`tRAS`) to accommodate the shift operations. 4) **PRAC-Ideal**: A secure PRAC design with no RMW timing overhead during activations, representing idealized PRAC performance.

Workloads and Configurations: Our evaluation uses 57 diverse applications from benchmark suites including SPEC2006 [23], SPEC2017 [24], TPC [25], Hadoop [26], MediaBench [27], and YCSB [28], available in the Ramulator2 repository [29]. We categorize workloads into three memory-intensity groups based on row buffer misses per kilo instructions (RBMPKI), as shown in Table 2. For each simulation, we run four homogeneous workloads, with each core executing 500 million instructions, totaling 2 billion instructions.

We conservatively assume that the shifting operation in One-Hot Counting increases the restoration time (tRAS) from 32 ns to 38 ns, representing a roughly 20% increase. To assess the performance impact, we conduct a sensitivity analysis, varying tRAS from 32 ns to 42 ns [1]. By default, we set the RowHammer threshold (N_{RH}) to 64 for all evaluated designs. For secure PRAC designs (PRAC, One-Hot Counting, and PRAC-Ideal), we configure one RFM per Alert, termed PRAC-1. We further explore sensitivity by varying N_{RH} between 64 and 4096, using Back-Off Thresholds (N_{BO}) derived from the QPRAC security analysis [5]. The evaluated (N_{RH}, N_{BO}) pairs are: (64, 23), (128, 80), (256, 222), (512, 481), (1024, 995), (2048, 2022), and (4096, 4072). We use weighted speedup as our performance metric to quantify the overhead introduced by each design.

6. Results and Analysis

6.1. Performance Overhead

Figure 8 shows the normalized performance of One-Hot-Counting and three PRAC designs compared to a baseline DDR5 system without PRAC at a RowHammer Threshold (N_{RH}) of 64. PRAC-Insecure and PRAC incur significant average slowdowns of 7.4% and 8%, while One-Hot-Counting and PRAC-Ideal show only a 1.1% and 0.9% performance loss.

[1] A detailed circuit-level analysis of this modification is left for future work.

The substantial overhead of PRAC-Insecure and PRAC primarily stems from the required Read-Modify-Write (RMW) operation on every activation to update per-row activation counters. This operation delays memory requests during row buffer conflicts, resulting in significant performance degradation, especially for memory-intensive workloads. For example, PRAC-Insecure causes over 10% slowdown in many memory-intensive workloads, such as `433.milc, 459.GemsFDTD`, and `482.sphinx3`, and reaches up to 18.3% in `483.xalancbmk`, as shown in Figure 8. PRAC introduces even higher overhead in several workloads due to additional mitigation overhead from Alert Back-Off (ABO), which stalls all DRAM banks for 350ns. For instance, in `471.omnetpp`, PRAC incurs a 20.1% slowdown, compared to 13.4% for PRAC-Insecure.

In contrast, our proposed One-Hot Counting avoids RMW operations and achieves near-ideal performance, incurring only a 1.1% average slowdown compared to 0.9% for PRAC-Ideal, even with a conservative 20% increase in tRAS to accommodate shift operations for counting. Even in the worst case, it causes only a 3% additional slowdown (`549.fotonik3d`) due to the increased tRAS. Overall, One-Hot-Counting effectively bridges the gap between PRAC and PRAC-Ideal while maintaining accurate activation counting.

6.2. Performance Sensitivity to Increased tRAS

Figure 9 shows the performance impact of One-Hot Counting as tRAS increases from the baseline 32ns to 42ns. As expected, performance degradation grows with longer tRAS, with average slowdown rising from 0.9% at 32ns to 1.2% at 42ns. The effect is more pronounced for memory-intensive workloads, where slowdown increases from 2.3% to 3.9%. Despite this, the overall overhead remains modest, even under conservative timing assumptions, demonstrating the practicality and efficiency of One-Hot Counting.

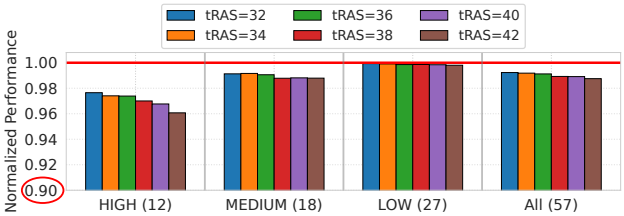

Figure 9: Normalized performance of One-Hot Counting as tRAS varies from 32ns to 42ns. Performance slightly degrades with increasing tRAS, with the average slowdown increasing from 0.9% at 32ns to 1.2% at 42ns.

6.3. Performance Sensitivity to RH Thresholds

Figure 10 presents the performance of One-Hot-Counting and three PRAC variants across N_{RH} from 64 to 4096. PRAC-Insecure and PRAC incur a significant average slowdown of 7.4% even at N_{RH} of 4096 due to Read-Modify-Write (RMW) operations for activation counter updates. In contrast, One-Hot-Counting incurs a 1.1% slowdown at N_{RH} of 64 and less than 0.5% at higher thresholds, even with a conservative 20% increase in tRAS to account for the cost of shift operations for counting. This closely matches the slowdowns of PRAC-Ideal,

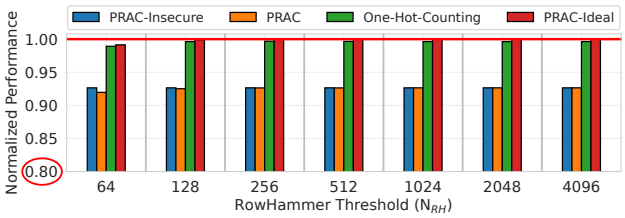

Figure 10: Normalized performance of One-Hot-Counting and three PRAC designs as N_{RH} varies from 64 to 4096. PRAC-Insecure and PRAC incur a significant average slowdown of 7.4% even at RowHammer Threshold (N_{RH}) of 4096 due to Read-Modify-Write operations for activation counter updates. In contrast, PRAC-Ideal incurs just a 0.9% slowdown at N_{RH} of 64 and almost no overhead at higher thresholds. One-Hot-Counting closely matches this performance, incurring only a 1.1% slowdown at N_{RH} of 64 and less than 0.5% at higher thresholds, even with a conservative 20% increase in tRAS to support shift operations.

which incurs a 0.9% slowdown at N_{RH} of 64 and almost no overhead at higher thresholds. This shows that One-Hot-Counting effectively eliminates the timing overhead of RMW operations while maintaining secure and accurate activation tracking.

7. Discussion and Future Work

The one–hot–encoded PRAC architecture demonstrates that row-activation counters can be integrated directly into the sense-amplifier region with minimal hardware, using two or three pass transistors per bit-line, thereby eliminating high–latency read–modify–write operations. Going into the future, our next steps will focus on (i) transistor-level layout, (ii) timing characterization, and (iii) adaptive control logic. We will co-optimize the placement of pass transistors within the bit-line pitch, evaluate parasitic loading on the sense amplifiers, and tune the restore-phase timing to ensure reliable shifts under typical process-voltage-temperature (PVT) corners. We are already working with a DRAM vendor to develop an industry-grade, SPICE-validated netlist. This will allow us to extract precise tRAS extensions, confirming (or tightening) the 32–42 ns range that we used as an upper bound in Section 4.2.

8. Conclusions

We propose a PRAC counting mechanism that eliminates the explicit read-modify-write (RMW) operations required by the current DDR5 PRAC specification. Our design employs one-hot encoding, allowing counter increments through simple shift operations. These shifts are efficiently implemented using pass transistors to redirect data during the restoration phase. To mitigate the higher area overhead associated with basic one-hot counting, we introduce a modified encoding scheme that can count up to twice as many states with minimal hardware additions. This modification significantly reduces overhead, limiting it to just a few percent for counts extending into the hundreds. The primary advantage of our design is the removal of performance-intensive RMW operations, substantially alleviating the timing and complexity challenges present in current PRAC implementations.

Acknowledgment

The authors thank Winbond Technology (DRAM vendor) for discussions on the feasibility of circuit modifications.

References

[1] A. Fakhrzadehgan, Y. N. Patt, P. J. Nair, and M. K. Qureshi, "Safeguard: Reducing the security risk from row-hammer via low-cost integrity protection," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 373–386.

[2] D.-H. Kim, P. J. Nair, and M. K. Qureshi, "Architectural support for mitigating row hammering in dram memories," *IEEE Computer Architecture Letters*, vol. 14, no. 1, pp. 9–12, 2015.

[3] 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 *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ser. ISCA '14. IEEE Press, 2014, p. 361–372.

[4] JEDEC, "Ddr5 sdram," https://www.jedec.org/standards-documents/docs/jesd79-5c01, 2024, accessed: 2025-4-18.

[5] 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 *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Los Alamitos, CA, USA: IEEE Computer Society, Mar. 2025, pp. 1021–1037. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/HPCA61900.2025.00080

[6] M. Qureshi and S. Qazi, "MOAT: Securely mitigating rowhammer with Per-Row activation counters," in *Proceedings of the 30th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, vol. 1, 2025.

[7] M. Qureshi, A. Rohan, S. Gururaj, and P. J. Nair, "Hydra: Enabling Low-Overhead mitigation of Row-Hammer at Ultra-Low thresholds via hybrid tracking," in *International Symposium on Computer Architecture (ISCA)*, 2022.

[8] O. Canpolat, G. A. Yağlıkçı, G. F. Oliveira, A. Olgun, O. Ergin, and O. Mutlu, "Understanding the security benefits and overheads of emerging industry solutions to DRAM read disturbance," in *4th Workshop on DRAM Security (DRAMSec))*, 2024.

[9] B. Nale and K. S. Bains, "Perfect row hammer tracking with multiple count increments," Apr. 21 2022, uS Patent App. 17/561,598.

[10] O. Canpolat, A. G. Yaglikci, G. F. Oliveira, A. Olgun, N. Bostanci, I. E. Yuksel, H. Luo, O. Ergin, and O. Mutlu, " Chronus: Understanding and Securing the Cutting-Edge Industry Solutions to DRAM Read Disturbance ," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. Los Alamitos, CA, USA: IEEE Computer Society, Mar. 2025, pp. 887–905. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/HPCA61900.2025.00071

[11] J. Woo and P. J. Nair, "Dapper: A performance-attack-resilient tracker for rowhammer defense," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 1005–1020.

[12] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible dram simulator," *IEEE Computer architecture letters*, vol. 15, no. 1, pp. 45–49, 2015.

[13] H. Luo, Y. C. Tuğrul, F. N. Bostancı, A. Olgun, A. G. Yağlıkçı, and O. Mutlu, "Ramulator 2.0: A modern, modular, and extensible dram simulator," *IEEE Computer Architecture Letters*, vol. 23, no. 1, pp. 112–116, 2024.

[14] A. G. Yağlikçi, M. Patel, J. S. Kim, R. Azizi, A. Olgun, L. Orosa, H. Hassan, J. Park, K. Kanellopoulos, T. Shahroodi, S. Ghose, and O. Mutlu, "Blockhammer: Preventing rowhammer at low cost by blacklisting rapidly-accessed dram rows," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 345–358.

[15] 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 *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 593–612.

[16] 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 *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1579–1596. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/olgun

[17] A. G. Yağlikçi, A. Olgun, M. Patel, H. Luo, H. Hassan, L. Orosa, O. Ergin, and O. Mutlu, "Hira: Hidden row activation for reducing refresh latency of off-the-shelf dram chips," in *MICRO*, 2022.

[18] 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 *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579371.3589063

[19] D. Kaseridis, J. Stuecheli, and L. K. John, "Minimalist open-page: a dram page-mode scheduling policy for the many-core era," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. New York, NY, USA: Association for Computing Machinery, 2011, p. 24–35. [Online]. Available: https://doi.org/10.1145/2155620.2155624

[20] S. Rixner, W. Dally, U. Kapasi, P. Mattson, and J. Owens, "Memory access scheduling," in *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, 2000, pp. 128–138.

[21] O. Mutlu and T. Moscibroda, "Stall-time fair memory access scheduling for chip multiprocessors," in *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, 2007, pp. 146–160.

[22] W. K. Zuravleff and T. Robinson, "Controller for a synchronous dram that maximizes throughput by allowing memory requests and commands to be issued out of order," May 13 1997, uS Patent 5,630,096.

[23] S. P. E. Corporation, "Spec cpu2006 benchmark suite," 2006. [Online]. Available: http://www.spec.org/cpu2006/

[24] "SPEC CPU2017 Benchmark Suite," Standard Performance Evaluation Corporation. [Online]. Available: http://www.spec.org/cpu2017/

[25] Transaction Processing Performance Council, "TPC Benchmarks." [Online]. Available: http://tpc.org/

[26] A. Foundation, "Apache hadoop." [Online]. Available: http://hadoop.apache.org/

[27] J. E. Fritts, F. W. Steiling, J. A. Tucek, and W. Wolf, "Mediabench ii video: Expediting the next generation of video systems research," *Microprocessors and Microsystems*, vol. 33, no. 4, pp. 301–318, 2009.

[28] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp. 143–154.

[29] SAFARI Research Group, "ABACuS — GitHub Repository," 2023. [Online]. Available: https://github.com/CMU-SAFARI/ABACuS