An Efficient Row Buffer Policy Based on Per-Row Activation Counts

Wonjae Choi[O], Gunjae Koo

Korea University

{wonjaechoi, gunjaekoo} @korea.ac.kr

Abstract

An important role of the memory controller is deciding whether to keep a DRAM row open or close it after each access. Conventional open-page and close-page policies are simple but cannot adapt to dynamic memory access patterns which leaves room for performance optimization. Hybrid policies improve adaptability but their need to maintain multiple statistics tracking counters introduce high hardware overhead, especially as DRAM capacity grows. To address this limitation, we propose PRACOpen, a lightweight row buffer policy that uses preexisting per-row activation count (PRAC) data in DDR5 DRAMs as a heuristic to inform row buffer management decisions. We evaluate the performance of PRACOpen by comparing it against the two conventional row buffer policies using a simulated computer system with benchmarks from the SPEC CPU2017 suite. Our evaluation shows that PRACOpen achieves performance gains while keeping hardware overhead low.

1. Introduction[1]

Row buffer management is a critical factor in DRAM performance. Conventional policies such as open-page which keeps a row open after access and close-page which immediately closes it are simple but lack adaptability. Adaptive or hybrid policies address this limitation by monitoring access patterns and switching between open and close modes. Although effective, these approaches require additional per-row and per-bank counters to keep track of statistics which leads to significant hardware and power overhead.

Meanwhile, modern DDR5 DRAMs include Per-Row Activation Counting (PRAC) to mitigate read disturbance errors (known as RowHammer-induced errors) by tracking how often each row is activated and signaling the controller when a threshold is reached. While designed for reliability, this mechanism maintains row access count data which can be used as a heuristic to determine memory access patterns.

We propose a PRAC-based Open-Page Policy (PRACOpen), a lightweight and adaptive row buffer management policy that exploits per-row activation count data in DDR5 DRAM devices. This design offers several advantages. First, it achieves better performance than conventional row buffer policies without having to maintain extensive statistics-monitoring counters. Second, it requires only minor extensions to the controller logic to implement and does not require any changes to existing DDR5 DRAM interface.

[1] This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (RS-2025-02304483, Chiplet-based hub SoC development optimized for on-device AI)

2. Background and Motivation

2.1 DRAM Organization and Operation

DRAM is hierarchically organized into channels, ranks, banks, rows, and columns. Each bank is a 2D array of cells accessed through the following commands: PRE (precharge) closes the active row, ACT (activate) opens a new row, and RD/WR (read/write) accesses data. The target row must be open to access data. In turn, accesses are classified as row buffer hits (same as previous row, latency tCL), misses (no active row, latency tRCD + tCL), or conflicts (different row, latency tRP + tRCD + tCL). The open-page policy keeps a row open after access, aiming to maximize row hits but at the cost of more row conflicts. The open-page policy performs well for workloads with high row buffer locality where consecutive accesses often target the same row but performs poorly for workloads with low locality where accesses frequently switch between rows. In contrast, the close-page policy minimizes conflicts by immediately closing rows after access which makes it more effective for low locality or random access patterns but it cannot exploit row hits in high-locality workloads.

2.2 Per-Row Activation Counting (PRAC)

In order to address the read disturbance errors, modern DDR5 DRAMs introduce Per-Row Activation Counting (PRAC), as specified in the updated JEDEC DDR5 standard [1]. PRAC implements a hardware activation counter for every DRAM row, incremented each time the row is activated. When a row's activation count reaches a defined back-off threshold (N_{BO}), the DRAM chip asserts a back-off signal to the memory controller. Figure 1 shows the basic operation of PRAC and hardware counters located next to individual DRAM rows. Upon

receiving this signal, the controller pauses normal operations and issues Refresh Management (RFM) commands, allowing DRAM to refresh potential victim rows and avoid bit flips. [2].

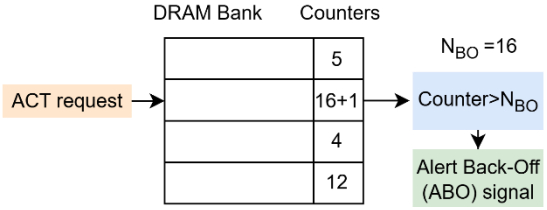

Figure 1. Basic PRAC operation

2.3. Current Challenges and Motivation

Existing adaptive page policies achieve performance improvements by monitoring row-access behavior using additional counters and heuristics inside the memory controller. However, as DRAM capacity grows the number of counters required scales with the number of banks or rows, introducing significant hardware cost. At the same time, DDR5 DRAM devices already maintain per-row activation counts for RowHammer protection. PRACOpen overcomes the scaling challenges faced by conventional adaptive page policy by repurposing the activation count data already maintained by PRAC to inform row buffer management decisions. Then, PRACOpen can effectively combine the strengths of both open-page and close-page policies while avoiding their weaknesses. In usual cases where the workload exhibits good row buffer locality, PRACOpen behaves like open-page policy and keeps frequently accessed rows open to maximize row hits and reduce access latency. However, when PRACOpen sees that the memory access pattern has become more random (low row buffer locality), PRACOpen temporarily switches to close-page mode for that bank. This preemptive closing can mitigate latency penalties caused by row buffer conflicts and thus improves performance.

3. Proposed Idea

3.1 Overview and design of PRACOpen policy

Figure 2 shows an overview and design of PRACOpen policy. We make the observation that per-row activation counter data kept in the DRAM device itself is exposed to the memory controller whenever the DRAM device triggers an alert signal. When DRAM triggers the alert signal, the memory controller can pinpoint which row triggered it by keeping track of the last issued activation command address for each bank. PRACOpen checks if two alert signals have been triggered by the same row within a set window. Under the open-page policy, a row remains open until a different row in the same bank is requested. ACT

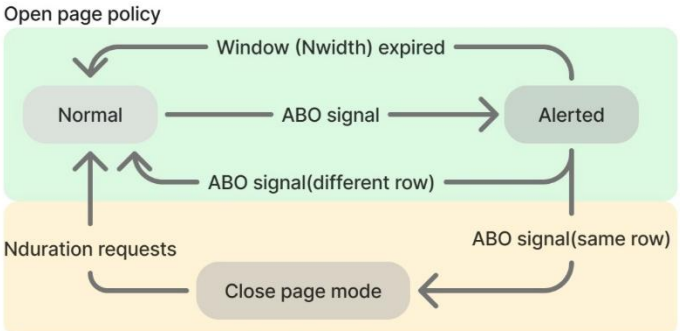

Figure 2. Overview of PRACOpen

requests are only generated in the case of row buffer conflict. Therefore, 2 alert signals from the same row in quick succession indicate row buffer contention where row conflicts are frequent. When such contention is detected, PRACOpen temporarily switches the affected bank to the close-page policy which minimizes latency under low locality conditions. After a fixed number of requests, it reverts to open-page mode assuming the access pattern has shifted back to one that allows row hits.

3.2 Operation of PRACOpen Policy

PRACOpen is implemented in the memory controller such that it (1) monitors PRAC alert (ABO) signals, (2) tests whether two ABOs come from the same row within a short request-count window (N_{width}), and (3) temporarily switches the target bank from open-page behavior to close-page behavior when condition indicates row-buffer contention. The duration for which close-page policy is applied after row buffer contention is detected is also defined by a fixed number of memory requests ($N_{duration}$). When a bank enters close-page policy, ABO and request-count window counting must be suspended. This is because under close-page policy, activations are forced essentially for every access which means repeated ABOs become expected and no longer indicate contention. PRACOpen does not change the correctness obligations imposed by PRAC. Whenever the DRAM asserts ABO, the controller still must issue the prescribed RFM(s).

4. Evaluation

We evaluate the effectiveness of PRACOpen using the cycle-accurate Gem5 simulator, where the memory subsystem is modeled with Ramulator2. [3]. Our proposed policy is implemented by extending the source code of Ramulator2. We simulate one copy of each workload from the SPECINT2017 benchmark suite for 10 million instructions[2] on each core. Table 1 describes the configuration of the simulated system.

[2] Due to simulator limitations, only 5 million instructions are simulated for the benchmark “xalancbmk”.

Table 1. System configuration

Processor	4 OOO cores, 4.2GHz
Cache	16KB L1I and L1D cache, 512KB L2 cache
Memory	DDR5 x8, 1-channel, 2-rank, 16GB, DDR5_3200AN speed bin
Memory address mapping	RoBaRaCoCh
PRAC	N_{BO}=16, PRAC-4
PRACOpen	N_{width}=64, $N_{duration}$ = 8

4.1. Performance Evaluation

We evaluate the performance benefit of PRACOpen over open and close-page policies by analyzing average per-core IPC values for each benchmark.

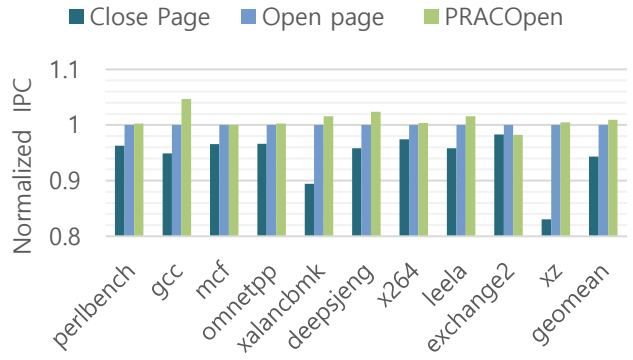

Figure 3. Performance of row buffer policies normalized to open-page policy

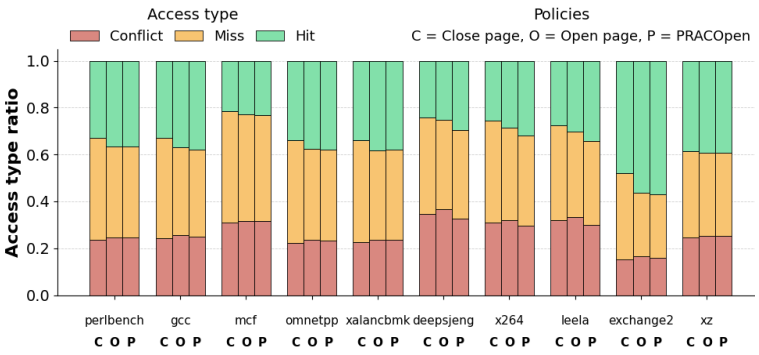

Figure 4. Row buffer access type breakdown

Figure 3 shows PRACOpen outperformed the open-page policy by 0.98% and the close-page policy by 7.21% on average (geometric mean). The only benchmark where PRACOpen performed worse than its peers was in "exchange2". Figure 4[3] shows that PRACOpen reduced the ratio of row buffer conflicts by 1.16 pp while increasing the ratio of row buffer hits by 1.37 pp compared to open-page policy. This indicates that PRACOpen successfully adapted to varying memory access patterns by selectively applying close page behavior when contention is detected, which resulted in the performance gain shown in Figure 3.

4.2 Hardware Evaluation

We focus on the additional storage overhead incurred by using PRACOpen. Each bank must maintain two address registers to keep track of the last activation address and the previous alert signal address, and 2 counters to track the duration remaining to use close-page policy as well as the number of requests between alert signals. Table 2 compares the storage overhead for PRACOpen with a traditional time-based adaptive page which utilizes 2-bit saturating counters per row [4]. As there can be thousands of rows per bank, PRACOpen incurs much less hardware overhead than the traditional adaptive policy, especially as DRAM size grows.

Table 2. Additional storage requirements for PRACOpen and a traditional adaptive page policy

PRACOpen	Traditional adaptive
$N_{banks} * (Address\ witdh + \log(N_{duration}) + \log(N_{width}))$	$N_{banks} * N_{rows\ per\ bank} * 2$

5. Conclusion and Future Work

We proposed PRACOpen, a novel buffer management policy that exploits activation alert (ABO) signals from DDR5 DRAMs to infer row access behavior. Our experiments show a modest yet consistent improvement in performance compared to conventional open and close-page policies, while significantly reducing hardware overhead compared to traditional adaptive approaches. This work demonstrates that reliability focused mechanisms such as PRAC can also be repurposed for performance optimization with minimal design effort. In future work, our idea may be extended to improve scheduling algorithms or refresh management policies.

6. References

[1] JEDEC Solid State Technology Association. *DDR5 SDRAM (JESD79-5C)*, 2024. https://www.jedec.org/standards-documents/docs/jesd79-5c01

[2] Canpolat, O., Yağlıkçı, A. G., Oliveira, G. F., Olgun, A., Ergin, O., and O. Mutlu, "Understanding the security benefits and overheads of emerging industry solutions to DRAM read disturbance", in DRAMSec, p.14 ,2024.

[3] 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", in IEEE Computer Architecture Letters, vol. 23, no. 1, pp. 112–116, 2024.

[4] Ghasempour, M., Jaleel, A., Garside, J. D., & Luján, M. "HAPPY: Hybrid Address-based Page Policy in DRAMs", in Proceedings of the Second International Symposium on Memory Systems, p.320, 2016.

[3] Instead of immediately closing a row, Ramulator2's implementation of close-page policy allows a fixed number of row accesses before closing it which allows row buffer hits and conflicts to occur.