Reducing Refresh Overhead with In-DRAM Error Correction Codes

Hanbyeol Kwon
Dept. of Nanoscale Semiconductor Engineering
Hanyang University
Seoul, Republic of Korea
gksquf808@hanyang.ac.kr

Kwangrae Kim
Dept. of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
kksilver91@hanyang.ac.kr

Dongsuk Jeon
Graduate School of Convergence Science and Technology
Seoul National University
Seoul, Republic of Korea
djeon1@snu.ac.kr

Ki-Seok Chung
Dept. of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
kchung@hanyang.ac.kr

2021 18th International SoC Design Conference (ISOCC) | 978-1-6654-0174-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISOCC53507.2021.9613990

Abstract— **DRAM technology scaling has continuously improved memory density, but the limited cell capacitance makes more susceptible to reliability issues. Hence, it has become inevitable to employ in-DRAM ECC. Also, the performance and power consumption overhead due to refresh operations have become a critical issue as the DRAM density increases. Therefore, it is very important to reduce the refresh overhead without sacrificing the reliability of DRAM. In this paper, we propose a retention-aware refresh scheme with in-DRAM ECC. The key idea of our proposed method is that the in-DRAM ECC can correct a single-bit error, and this will effectively reduce the number of weak rows that have to be refreshed every 64ms. Also, a runtime profiler is proposed to keep up-to-date information of weak rows to solve the variable retention time problem. Our experiments with SPEC benchmarks show up to 6.8% performance improvement of performance, and up to 15.4% reduction of power consumption compared with the conventional refresh schemes.**

Keywords; Retention-Aware Refresh; In-DRAM ECC

I. INTRODUCTION

As the DRAM cell size becomes smaller and the memory capacity increases, the overheads due to refresh operations often lead to significant power consumption and performance degradation [3]. As defined by the JEDEC standard, DRAM cells must be refreshed every 64ms to ensure data integrity [1]. However, most cells may retain data for much longer than 64ms (*strong cells*), while only a few *weak cells* require the 64ms refresh interval. Therefore, many studies [3,4,5,6,7] have tried to minimize the refresh overhead by reducing the refresh frequency of the strong cells. This approach is typically called *retention-aware refresh*. However, as the DRAM technology scales down, issues such as growing number of weak cells and variable retention time (VRT) characteristics have emerged to limit the benefits of the retention-aware refresh [5, 6].

On the other hand, the JEDEC DDR5 standard specifies the Error Check and Scrub (ECS) function by using in-DRAM error correction codes (IECC) because the reliability of a DRAM cell has gradually deteriorated as the cell capacity scales down [2]. Therefore, DDR5 will be equipped with on-die error-correction codes to correct at least a single-bit error for each codeword. The ECS operation periodically reads DRAM data with IECC and corrects an error if it occurs. We claim that the proportion of the weak rows can be significantly reduced, and the VRT problem, which addresses a problem of randomly changing retention times of some cells at runtime, can be solved by smartly utilizing the ECS operation.

In this paper, we propose a novel retention-aware refresh scheme. The previously proposed refresh schemes without IECC defined a weak row as the row that would include at least one weak cell because a refresh operation is performed on one row at a time. Instead, our proposed retention-aware refresh with IECC defines a weak row as the row that contains two or more weak cells. Also, we suggest a runtime profiling method to solve the VRT problem by updating the weak row information with periodic ECS operations. Our experiments with the SPEC benchmarks show 6.8%/2.9% improvement of performance, and 15.4%/8.1% reduction of power consumption at 1.28e-05 weak cell probability compared with commonly used *auto-refresh* [1] and *Elaborate refresh* [7] schemes, respectively.

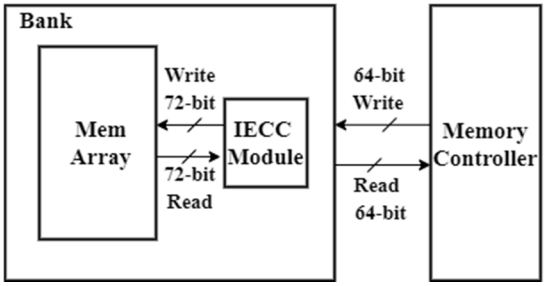

Figure 1. DRAM bank with IECC module

II. BACKGROUND & MOTIVATION

A. Error Check and Scrub with In-DRAM ECC

Fig. 1 shows an architecture of a DRAM bank with a (72, 64) In-DRAM ECC (IECC) module. When the memory controller issues a write command of a 64-bit data, the IECC module writes a 72-bit data by adding a generated 8-bit parity to the memory array. This data plus parity bit is called *codeword*. In case of a read operation, the IECC module reads a 72-bit data, and checks and corrects the error before sending

a 64-bit data to the memory controller [10]. There are two types of Hamming codes that are commonly used as IECC: a 128-bit data with an 8-bit parity code (6.25% overhead) to deal with single error correction (SEC), and a 64-bit data with an 8-bit parity code (12.5% overhead) to enable SEC with double error detection (SECDED). These two codes are popular because they can be implemented with low area and latency overheads in DRAM [4].

The Error Check and Scrub (ECS) operation internally checks the occurrence of a single-bit error with IECC module after reading a memory array and writes back the corrected data if an error occurs. In a 32Gb DRAM chip, the ECS operation is periodically conducted with a 0.322ms interval to scrub the whole chip every 24 hours. During the ECS mode, the information such as the error count per row and the corresponding address is updated in real-time. Therefore, it is possible to solve the VRT problem, which is regarded as one of the most challenging issues to solve in the previous retention-aware refresh schemes without IECC.

B. Retention-Aware Refresh

The major source of DRAM errors are retention errors because of randomly scattered weak cells, which have short retention times [4]. To avoid bit errors due to weak cells, the DRAM cells must be refreshed periodically. A widely used refresh scheme called *auto-refresh* refreshes all rows by issuing 8192 refresh commands within 64ms (tRET). However, most rows do not require such a short refresh interval [3]. To mitigate the inefficiency of the conventional refresh scheme, many prior works have suggested multi-rate refresh methods to reduce the refresh overhead by exploiting the information on the weak row that requires a refresh interval less than 256ms. Unfortunately, these methods still conduct lots of refreshes when the weak cell probability is high because rows containing at least one weak cell are regarded as weak rows. Moreover, it is practically impossible to profile on weak rows with the VRT characteristic, at the manufacturing stage. Commonly, a weak row probability is calculated as follows:

$$P_{weak\_row} = 1 - (1 - P_{weak\_cell})^N \quad (1)$$

where N is the number of cells per row and P_{weak_cell} is the weak cell probability. The weak row probability depends on the refresh granularity.

Table I. Weak row probability with 1KB page size and 8 chips

Weak Cell Prob.	Weak Row Probability			
	Rank-level Refresh	Chip-level Refresh	Chip-level Refresh (≥2 errors)	Chip-level Refresh (≥3 errors)
3.2e-06	18.92%	2.59%	0.034%	0.00029%
6.4e-06	34.26%	5.11%	0.13%	0.0023%
1.28e-05	56.78%	9.95%	0.51%	0.018%
2.56e-05	81.32%	18.92%	1.91%	0.13%

C. Motivation

Term *row group* is defined as a group of rows that is refreshed by one refresh command, and it is determined by two factors, refresh granularity and the number of weak cells in each row group. The refresh granularity means how many rows are refreshed at a time. Table I summarizes the weak row probability with respect to various granularities and weak cell probabilities. The conventional rank-level refresh simultaneously refreshes the rows with the same address in each chip. Table I shows that 81.32% of the rows are regarded as weak rows when the weak cell probability is 2.56e-05. Another refresh scheme called *Elaborate refresh* [6] proposed a chip-level refresh method, which stores the weak row address in each chip and simultaneously refreshes weak rows in each chip. In this method, the weak row granularity is reduced to 1/8 in a system with eight chips, and the weak row probability is 18.92%, much smaller than 81.32%. However, it is still a high probability that will require frequent refresh operations. This frequent refresh is caused by the fact that only one weak cell in a row group will require a refresh. On the other hand, an IECC-equipped DRAM system can correct a single bit error due to a weak cell. Therefore, unless two or more errors occur in each 74-bit codeword, the system reliability can be maintained without having to carry out additional refreshes. We observe that the probability that a row contains two or more weak cells is 1.91%, significantly lower than 18.92%. Therefore, in this paper, we propose a scheme that reduces the number of weak rows by relaxing the criterion for weak rows from a single weak cell per row to two or more weak cells per row. Also, we propose a real-time weak row profiling method using the ECS operation to solve the VRT problem.

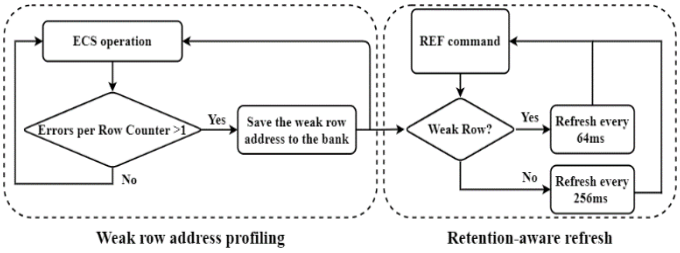

Figure 2. The proposed method

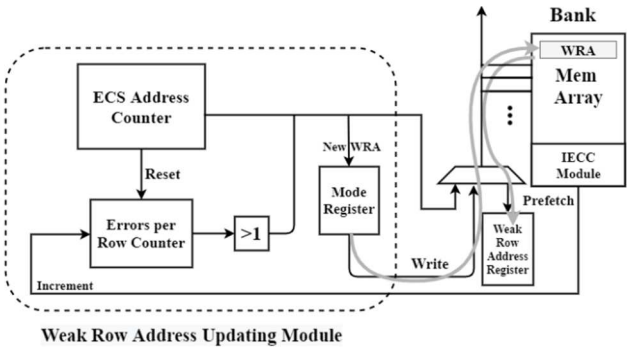

Figure 3. Module for updating Weak Row Address (WRA)

III. PROPOSED METHOD

A. Overall operation

We propose a chip-level retention-aware refresh scheme for DRAM systems with in-DRAM ECC. Fig. 2 shows the overall flow of our proposed scheme: 1) Weak row address profiling and 2) Retention-aware refresh. The proposed scheme enables runtime profiling of weak row addresses with the ECS operation. An ECS operation is performed in each codeword, and a counter called *Errors per Row Counter* (EpRC) is incremented if an error is detected. If the EpRC value is more than one in one row after finishing the ECS operation, a module for weak row address profiling saves the weak row address to the designated area within a bank as shown in Fig. 3.

The proposed method refreshes strong rows every 256ms and refreshes weak rows whose address information is saved in each bank every 64ms. Compared to other existing retention-aware refresh schemes, the number of weak rows is significantly reduced because the proposed method redefines a weak row as the row containing two or more weak cells because IECC can correct a single-bit error while reading the codeword. Meanwhile, the VRT problem may cause even a strong row to contain two or more errors. Our proposed method solves the VRT problem by profiling the entire set of rows once every 24 hours to keep an up-to-date information on strong rows and weak ones.

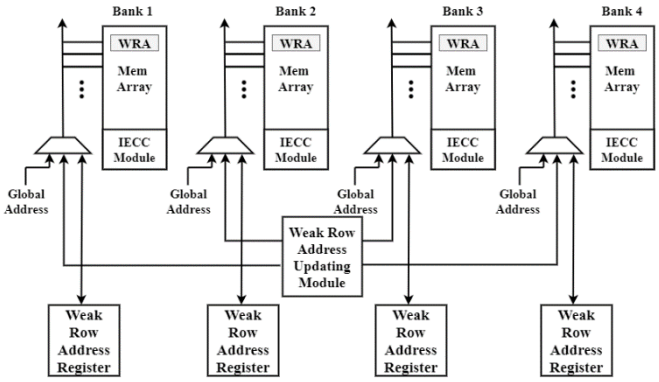

Figure 4. Address bus architecture

B. Implementation

Fig. 4 shows the implementation of the address bus architecture. During the system initialization, the address information on weak rows is assumed to be obtained from the profiling during the manufacturing process and stored in a designated area within each DRAM bank. Storing the weak row address in each bank makes it possible for the DRAM device to refresh weak rows in each bank in parallel. Each *Weak Row Address Register* (WRAR) can store up to eight *weak row addresses* (WRA) that are copied from the designated area within each bank. When a strong row group is refreshed, the corresponding row address is provided via the global bus line. On the other hand, when a weak row group is refreshed, the address is provided by WRAR. A module for updating WRA is used to transfer the ECS operation results and update the WRA information correspondingly.

Fig. 3 shows the operation of the module for updating WRA for the proposed runtime profiling. (72,64) SECDED Hamming codes are used as IECC in our proposed method. During an ECS operation, the IECC module reads a codeword (a 64-bit data with an 8-bit parity code) whose address is provided by a block called *ECS Address Counter* (EAC) and writes the corrected data back if a single-bit error is detected. The IECC module increments *Errors per Row Counter* (EpRC) to count the number of the corrected errors. EAC resets EpRC when finishing the ECS operation for each row. If a new weak row is found, the module for updating WRA in Fig. 3 stores the corresponding address in a register called *Mode Register* (MR) and writes the new WRA to the designated area of the DRAM bank. This operation must be completed before the end of the next row's ECS operation because the MR can store only one address. The overhead of writing a new WRA to the DRAM is negligible because the VRT problem does not occur frequently. The weak rows in the updated WRA will be refreshed every 64ms.

IV. EXPERIMENTAL ENVIRONMENT AND EVAULTION

A DRAM simulator called DRAMSim2 [8] and an instruction set simulator called Gem5 [9] were integrated with some additions to implement the proposed method. Table II shows the set of simulation parameters of a DRAM device and a multi-core processor that have been used for performance evaluation. We simulated an out-of-order 8-core processor by running the Alpha benchmark binaries selected from the SPEC CPU 2006 benchmark suite. The weighted sum metric, which is sum of the ratios of the benchmark execution time, is used to evaluate the performance of our proposed method. We compared our proposed method with the conventional auto-refresh and Elaborate refresh methods. As aforementioned, (72,64) SECDED Hamming codes were used in the IECC module.

Figures 5 and 6 show the normalized weighted speed-up and energy per access when the weak cell probability is 1.28e-5. Each result is normalized to the JEDEC's standard auto-

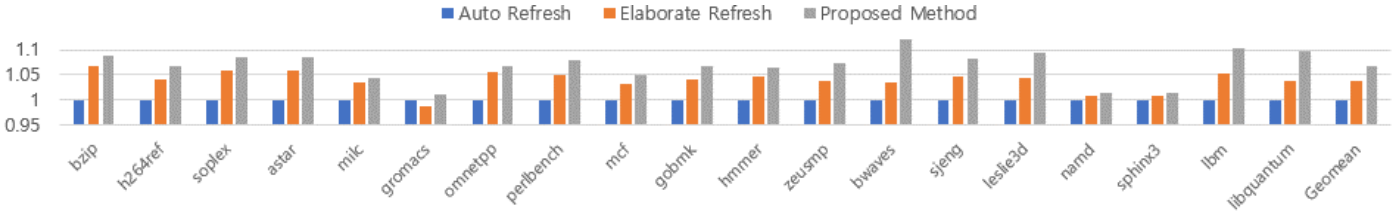

Figure 5. Normalized weighted speed-up when the weak cell probability is 1.28e-5

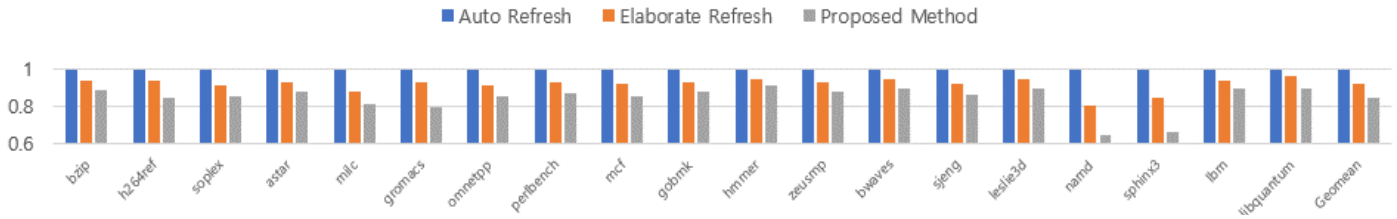

Figure 6. Normalized energy per access when the weak cell probability is 1.28e-5

refresh scheme, and the geometric mean is used to compare results. Our proposed method showed 6.8%/2.9% of performance improvement, and 15.4%/8.1% of energy reduction per access over the auto-refresh and Elaborate refresh schemes, respectively.

Table II. Simulation parameters

Processor	8 cores, 2GHz, 8-wide issue, 8 MSHRs/core, out-of-order 192-entry instruction window
Last-Level Cache	4MB shared, 64B cache line, 8-way associative
DRAM Controller	FR-FCFS, 64-entry request queue
DRAM	32Gb Device, x8 DDR4-3200, 1 channel, 1 rank, 8 bank groups/rank, 4 banks/bank group 128K rows/bank, 1KB page size
IECC	(72, 64) SECDED Hamming code
Weak Cell Probability	1.28e-5, 2.56e-5

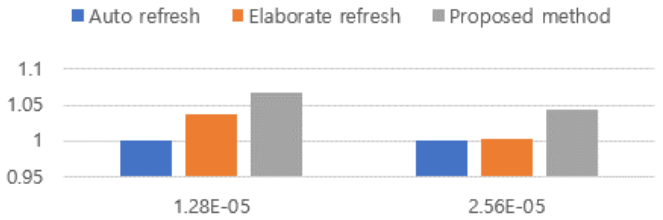

Figure 7. Normalized geometric mean of performance when the weak cell probability is 1.28e-5 and 2.56e-5

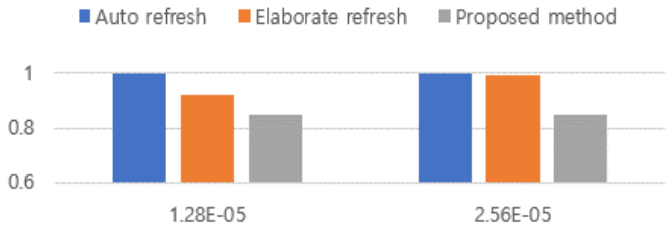

Figure 8. Normalized geometric mean of energy per access when the weak cell probability is 1.28e-5 and 2.56e-5

Figures 7 and 8 show the comparison results of performance and energy per access when the weak cell probability is 1.28e-5 and 2.56e-5, respectively. At a higher weak cell probability, the differences on both the performance and the energy consumption between the auto-refresh scheme and the Elaborate refresh scheme are negligibly small. However, our proposed method showed 4.4% of performance improvement and 14.9% of energy reduction per access. Therefore, we can conclude that the proposed method is very effective to reduce the refresh overhead to improve the performance and the energy efficiency without sacrificing the reliability of DRAM systems.

V. CONCLUSION

In this paper, we propose a retention-aware refresh scheme for a DRAM system with in-DRAM ECC. Our proposed method can effectively reduce the number of weak rows by utilizing the in-DRAM ECC module. Also, a runtime profiler to dynamically update the weak row information is proposed. Thereby, the DRAM refresh overhead was significantly reduced and the VRT problem was effectively solved. Our experimental results show that the proposed method have 6.8%/2.9% of performance improvement and 15.4%/8.1% of energy reduction at 1.28e-05 as the weak cell probability compared with the conventional Auto-refresh and Elaborate refresh schemes, respectively. We expect that our proposed method will be more valuable when the weak cell probability is getting higher in the future.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A4A4079177).

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00198, Development of in-building 28GHz OTA digital repeater by using uplink noise cancellation)

REFERENCES

[1] JEDEC. “DDR4 SDRAM specification JESD79-4B.”2017.

[2] JEDEC. “DDR5 SDRAM specification JESD79-5.”2020

[3] Liu, Jamie, et al. "RAIDR: Retention-aware intelligent DRAM refresh." ACM SIGARCH Computer Architecture News 40.3 (2012): 1-12.

[4] Cha, Sanguhn, et al. "Defect analysis and cost-effective resilience architecture for future DRAM devices." 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2017. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[5] Choi, Haerang, et al. "Reducing DRAM refresh power consumption by runtime profiling of retention time and dual-row activation." Microprocessors and Microsystems 72 (2020): 102942.

[6] Qureshi, Moinuddin K., et al. "AVATAR: A variable-retention-time (VRT) aware refresh for DRAM systems." 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE, 2015.

[7] Seol, Hoseok, et al. "Elaborate refresh: a fine granularity retention management for deep submicron DRAMs." IEEE Transactions on Computers 67.10 (2018): 1403-1415. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[8] Rosenfeld, Paul, Elliott Cooper-Balis, and Bruce Jacob. "DRAMSim2: A cycle accurate memory system simulator." IEEE computer architecture letters 10.1 (2011): 16-19. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[9] Binkert, Nathan, et al. "The gem5 simulator." ACM SIGARCH computer architecture news 39.2 (2011): 1-7.

[10] Jeong, Sangmok, SeungYup Kang, and Joon-Sung Yang. "PAIR: Pin-aligned In-DRAM ECC architecture using expandability of Reed-Solomon code." *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020