Per-Bank Refresh with Adaptive Early Termination for High Density DRAM

Hyun-Woong Yang
Dept. of Nanoscale Semiconductor Engineering, Hanyang University
222, Wangsimni-ro, Seongdong-gu, Seoul, 04763, Republic of Korea
+82-10-9483-9143
yhw0011@naver.com

Min-Kyu Lee
Dept. of Electronic and Computer Engineering, Hanyang University
222, Wangsimni-ro, Seongdong-gu, Seoul, 04763, Republic of Korea
+82-10-9188-8034
hanloveland@naver.com

Ki-Seok Chung
Dept. of Electronic and Computer Engineering, Hanyang University
222, Wangsimni-ro, Seongdong-gu, Seoul, 04763, Republic of Korea
+82-10-9056-7400
kchung@hanyang.ac.kr

ABSTRACT
DRAM, which is mainly used as main memory, requires a refresh operation to maintain the integrity of stored data. Since memory read and write operations to a bank are not allowed while the bank is being refreshed, a lot of memory accesses may be blocked due to refresh, which may lead to significant performance degradation. Therefore, a lot of active studies to minimize this negative performance impact of refresh have been conducted. In a refresh scheme called per-bank refresh, the refresh unit will be one bank rather than all banks in a rank, allowing memory access to other banks while a certain bank in the same rank is refreshed. However, the per-bank refresh consumes more power than all-bank refresh. In this paper, we propose a per-bank refresh method with adaptive early termination, which allows both the refresh period and the size of each row group to be non-uniformly determined, to increase the efficiency of per-bank refresh and reduce energy consumption. By using this method, compared to the basic all-bank refresh model, the average weighted speed increases by about 6.4% and the energy consumption is reduced by about 51%.

CCS Concepts
• **Hardware→Dynamic memory.**

Keywords
DRAM refresh; Low power DRAM; Memory architecture

1. INTRODUCTION
Dynamic random access memory (DRAM) is mainly used as main memory in modern computing devices. DRAM is composed of a huge array of memory cells in which data is stored. One memory cell consists of a transistor and a capacitor, and 1-bit information is stored by charging the capacitor. However, the stored charge may leak away over time from the capacitor. If the amount of leakage is significant, the stored information will be lost. Therefore, a refresh operation, which periodically recharges the capacitor, is required to prevent the data loss.

ICCIP 2018, November 2–4, 2018, Qingdao, China

ACM ISBN 978-1-4503-6534-5/18/11…$15.00
http://doi.org/10.1145/3290420.3290442

Refresh is indispensable in the DRAM operation, but it may cause DRAM performance degradation. The criterion that determines the refresh period for a particular row is the minimum time duration that the logical value of the weakest memory cell in the row is maintained. This time is defined as retention time in the JEDEC-DDR standard. Thousands of refresh commands are invoked to refresh all rows in the memory array. A set of rows that is refreshed by one refresh command is called as row group. Conventionally, all banks in a rank are refreshed by one refresh command, and this method is called as all-bank refresh (ABR). In ABR, the memory controller cannot issue a memory request to any bank in the same rank during the refresh operation time. Therefore, performance degradation due to blocked memory requests during the refresh will be significant. In order to reduce the negative performance impact caused by ABR, JEDEC proposes an additional refresh method called per-bank refresh (PBR) for a low power DRAM memory called LPDDR. The row group size of PBR is the same as that of ABR divided by the bank size. In other words, in PBR, only the rows in one bank are refreshed at one refresh command. Therefore, the memory request can be serviced for the banks that are not being refreshed, so that parallelizing refreshes and accesses to banks in the same rank becomes possible. However, since only a few rows are refreshed in one refresh command, more refresh commands need to be generated within the retention time. As a result, the total refresh time increases and the power consumption due to refresh also increases.

Commodity DRAM performs the refresh operation with the same refresh period in all rows in accordance with the retention time of the weakest cell. Weak cells are not controllable because they are resulted from process-voltage-temperature (PVT) variation [1]. However, the probability of the occurrence of weak cells is extremely small, and the retention time of most cells is longer than 64 ms [2]. Therefore, the refresh period of all rows does not have to be determined based on the retention time of the weak cell. If an early termination refresh method using the above retention time characteristic is used, it is possible to perform refresh operations with different retention times for each row group [3]. Therefore, it is possible to avoid power dissipation and time waste due to unnecessary refresh. However, it is impossible to access a bank and refresh another bank in parallel because the existing early termination refresh method is based on ABR. In this paper, we propose a refresh method that applies the early termination refresh method to PBR and improves the refresh power efficiency by adaptively changing the granularity of skipping the refresh for each row group.

The remaining of this paper is organized as follows. In Section 2, we explain background information that helps to understand this paper. In Section 3, we propose a method for adaptive early termination refresh to improve performance and power efficiency. In Section 4, we explain experimental results. Section 5 will conclude this paper.

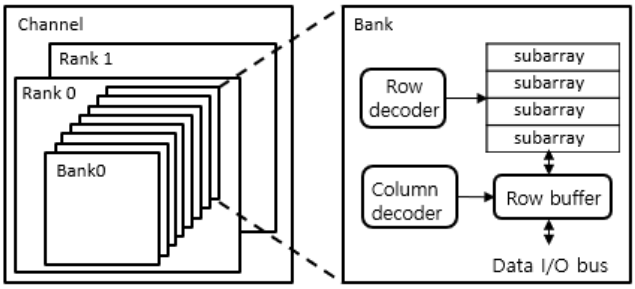

Figure 1. DRAM and bank organization.

2. BACKGROUND

2.1 Dram System Organization

Figure 1 shows the general structure of a DRAM system and a bank. The channel consists of several ranks that share a memory bus between the memory controller and the DRAM. Each rank is made up of several banks that operate independently. In DRAM, data is read and written by three commands: activate, read/write, and precharge. Each bank consists of subarrays, and a subarray is made up of a two-dimensional array of memory cells with rows and columns. The DRAM sends a row address to the row decoder using the activate command to select a row. Then, the data of the selected row is copied from the memory array to a row buffer. After the activated row is stored in the row buffer, the column decoder executes a column access command such as read or write. To execute a read or write command on another row that is not in the row buffer, the precharge command must be executed in order for the row buffer to prepare for the new activate command.

2.2 Dram Refresh

2.2.1 All-bank refresh

Since there are tens of thousands of rows in the DRAM, the refresh latency increases when all rows are refreshed at once. Therefore, the memory controller periodically sends several refresh commands within the retention time. The refresh command in a typical DRAM acts across all the banks and this refresh method is called as all-bank refresh (ABR). In general, ABR refreshes as many rows as the total number of rows divided by 8192. Therefore, the refresh command generation period is 7.8 s. A refresh scheme where a set of rows is grouped and then the set is refreshed together is called auto refresh (AR). In ABR, the refresh command refreshes all banks in a rank, so the rank cannot service memory requests during the refresh time.

2.2.2 Per-bank refresh

LPDDR DRAMs support an additional refresh method called *per-bank refresh* (PBR) to allow concurrent memory accesses to the DRAM during refresh [4]. The number of refresh commands to be issued in PBR is equal to the number of refresh commands in ABR multiplied by the number of banks. That is, in PBR, one row group in one bank is refreshed with each refresh command. With the support of PBR, LPDDR DRAM can process refresh operations of specific banks and memory accesses of the other banks in the same rank in parallel. Therefore, performance degradation due to refresh operation is reduced in PBR. However, the frequently issuing refresh commands will increase the total time required for the refresh operation compared to ABR. When a row group is refreshed, additional timing parameters such as tRCD and tRP are required to prepare for the refresh operation. Therefore, in the case of PBR, it takes more time to refresh all rows in the DRAM device. As a result, the power consumption due to refresh also increases accordingly.

2.3 Early Termination Refresh

Early termination refresh (ETR) is a refresh method that has an advantage of reduced refresh overhead because it selectively skips refresh operations for a row group with a long retention time. To apply ETR, flag bits need to be added to the memory array. Profiling on the DRAM retention time will identify which cell in each row group has the shortest retention time. Based on the cell that has the shortest retention time in each row group, each row group will be classified into one of the four retention time types [2], and the type will be marked by a 2-bit flag in the first row of each row group. The overhead caused by the flag bits is extremely small [3]. The added flag bits indicate the retention time of the row group and the flag bits are refreshed every 64 ms. Every time the flag bits are refreshed, the memory controller checks whether the refresh of the remaining rows in the row group is necessary or not. Since the percentage of having a weak cell in a row group is very low, the percentage of row groups with a retention time of 64 ms is extremely small. Therefore, skipping refresh operations will be possible for a large number of row groups to minimize refresh overhead [2]. However, since this ETR is based on ABR, servicing memory accesses concurrently with refresh operations is impossible.

3. PROPOSED REFRESH SCHEME

Since ETR based on ABR cannot carry out memory accesses and refresh operations in parallel, an ETR scheme based on PBR is proposed in this paper. Especially, we propose a new refresh method called adaptive early termination refresh (AETR) that can change the refresh granularity adaptively on top of the existing ETR method which changes the refresh period according to the retention time of the row group. In the proposed AETR method, the parallelism of PBR is ensured, while the efficiency of the refresh is improved by skipping refresh operations on more rows, leading to performance improvement and energy saving.

As the DRAM density increases, the number of rows included in one row group increases. Correspondingly, the probability that a weak cell is included in the row group also increases. Therefore, to apply AETR, the base row group size is set to 1/4 of the row group size used in general per-bank refresh. On the other hand, since the row group size is reduced to reduce the number of row groups having a weak cell, the refresh command is issued 32 times more often than the conventional ETR in the case of that there are 8 banks in a rank. Because of the increased frequency of refresh command issues, the time required to test whether refresh will be skipped causes a significant overhead. Therefore, in AETR, consecutive row groups that have the same retention time will be merged into a new row group. Then, the size of row groups will be different, and therefore, additional flag bits to indicate the size of a row group need to be added.

To minimize the additional overhead due to the additional flag for the row group size, the total number of bits to be used for the two different flags is fixed to 4 bits. Then there are two possible combinations: 2 bits for the retention time type and 2 bits for the row group size ((2:2) scheme) or 1 bit for the time and 3 bits for

the size ((1:3) scheme). We have conducted simulations to determine which is better. In case of the (2:2) scheme, the retention time is divided into 4 types of 64ms, 128ms, 256ms, and 512ms, and the size is divided into 4 types consisting of 1, 2, 4, and 8 times of the base row group size. For the other (1:3) scheme, the retention time is divided into 2 types of 64ms and 256ms, and the size is divided into 8 types consisting of 1, 2, 3, 4, 8, 16, 24 and 32 times of the base row group size.

Table 1. Performance comparison based on retention time and group size

Retention time	Group size	Average REF Cycle	Issued REF cmd #	Total Refresh Cycle (In 512ms)
1	3	118.612	280774	68815836
2	2	63.1658	1677226	105943272

Table 1 shows the simulation result when we measure the number of refresh cycles and that of refresh command issues during 512 ms. The number of cycles used for refresh for the (1:3) scheme is about 65% of that used for refresh for the (2:2) scheme. Also, during the same time period, the number of issued refresh commands for the (1:3) scheme is 86% less than that of the (2:2) scheme. This simulation result strongly implies that it is much better to have more row group size options than more retention time options. By having large row groups, it is possible to skip refresh on a large number of rows at once. Therefore, in the proposed method, the 4-bit flag is composed of 1-bit retention time flag and 3-bit row group size flag.

Figure 2 is an illustration of the operation of AETR. When the retention time bit is '1', the row group will be refreshed every 64 ms, and when the bit is '0', the row group will be refreshed every 256 ms. The 3-bit row group size flag indicates 8 different allowed row group sizes: 1, 2, 3, 4, 8, 16, 24 and 32 times of the base row group size denoted by 000 to 111, respectively.

Merging two adjacent row groups to form a bigger row group is conducted recursively in a bottom-up fashion. Adjacent row groups are merged only if the retention time bits of the two groups are the same and the size of the merged row group is one of the allowed group sizes. So in Figure 2, row groups 0~7 are merged into a merged row group in a bottom-up fashion. Also, row groups 8~10 are merged into a merged group similarly. However these two adjacent row groups will not be merged further because the size of the merged row group will become 11 which is not one of the allowed group sizes.

The first row of each row group contains the 4-bit flag, and it is read every 64ms. A refresh controller in the DRAM device decides whether the refresh operation will be conducted or not according to the retention time bit. The remaining 3-bit group size flag will be used to locate the first row of the next row group. In the proposed design, a counter called Refresh Counter is incremented by the size to find the first row of the next row group.

Since the occurrence probability of weak cells is very low in practice, the size of the row group of which retention time bit is '1' is very likely to be kept to be the minimum which is the size of the basic row group . Therefore, the number of rows to be refreshed every 64ms should be minimized.

In summary, in ATER, the size of the row group is not uniform, and since the occurrence of weak cells is not frequent, the average size of the row group without any weak cell will be much larger than that of ETR. Therefore, the total number of row groups in ATER is less than ETR. Correspondingly, the overhead to handle refresh operations, and the number of rows to be refreshed often will be minimized in practice.

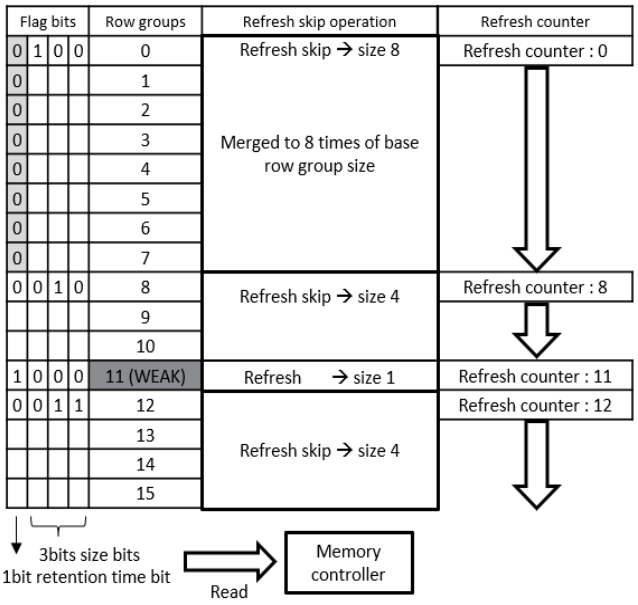

Figure 2. Overview of Adaptive early termination refresh.

Table 2. Simulation parameter

Processor	4 cores, 2.0GHz, out-of-order, 32-entry inst window
L1 caches	64B cache-line, 32 KB inst/data, 2-way, LRU, 2 cycles
L2 caches	64B cache-line, 512 KB, 8-way, LRU, 20 cycles
DRAM controller	FR-FCFS scheduling policies, 64-entry request queue, 32-entry command queue per bank
DRAM parameter	667 MHz bus cycle, DDR3-1866 [7], 8 B-data bus, 8 DRAM banks, latency: 13-13-13 ns (tRP-tRCD-CL)

4. EXPERIMENTAL RESULTS

4.1 Environment for experiments

Gem5 [4] and DRAMSim2 [5], widely used simulators, were used to evaluate the performance of AETR. Also, refresh energy consumption is measured by using retention time variation [2] and power consumption ratio of ABR and PBR of Micron's LPDDR [4]. Table 2 shows simulation parameters of the DRAM device and the processor used in the simulation. We simulated an out-of-order processor by running Alpha benchmark binaries which were selected from the SPEC CPU 2006 benchmark suite. A metric called "weighted speedup", which is the sum of the ratios of the execution time when a benchmark program runs under multi-program execution over the execution time when the benchmark program runs under single-program execution [6], was used to evaluate the performance of the proposed method. The baseline refresh scheme is the conventional ABR scheme.

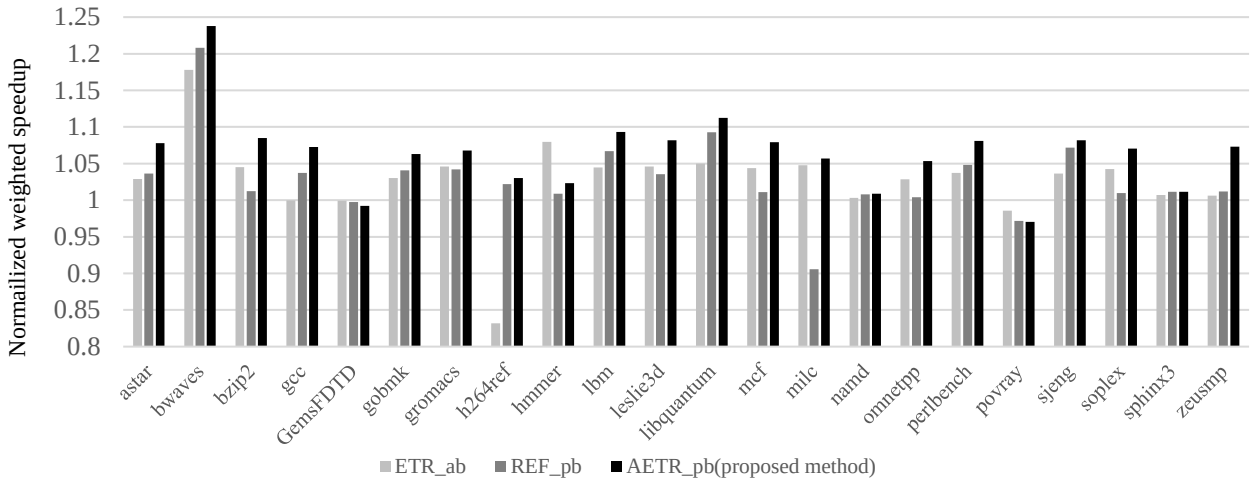

Figure 3. Normalized weighted speedup.

4.2 Experiment results

Figure 3 shows the normalized weighted speedup of each refresh method for each benchmark. Each result is normalized to the baseline ABR scheme (REF_ab). Three different refresh methods are compared with REF_ab. REF_pb represents the conventional per-bank refresh scheme [4] and ETR_ab represents the early termination refresh [3]. AETR represents the proposed Adaptive Early Termination Refresh. The average weighted speedup for all the benchmark programs is 1.064 for AETR_pb, 1.029 for REF_pb, and 1.028 for ETR_ab. The proposed method achieves the highest average weighted speedup. Specifically, the weighted speedup of AETR is 6.4%, 3.5% and 3.6% better than REF_ab, REF_pb and ETR_ab, respectively.

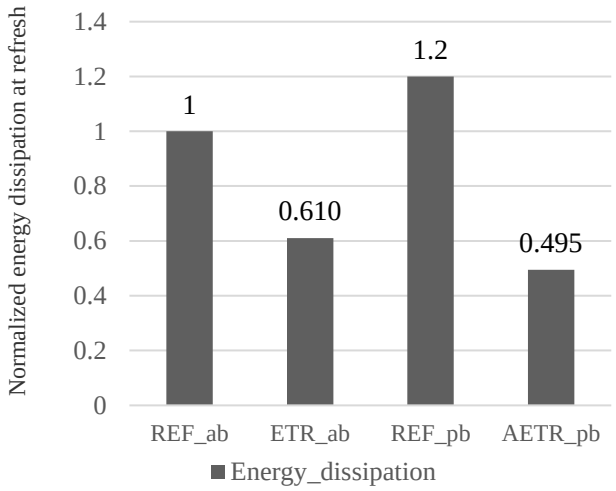

Figure 4: Refresh energy dissipation

Figure 4 shows the energy dissipation when the DRAM device operates refresh commands. In order to measure energy consumption, we used a method of multiplying the current value by the time that all rows are refreshed. It is assumed that the supplied voltages are the same. Compared with REF_ab, energy consumption improvement of about 61% is achieved with ETR_ab. In particular, AETR_pb improves energy consumption by about 51% when compared to REF_ab.

From the results shown in Figure 3 and Figure 4, it is confirmed that the proposed method not only improves performance, but also achieves considerable reduction on energy consumption.

5. CONCLUSION

As the DRAM device density grows rapidly, time and energy consumption due to refresh operations increase correspondingly. Since memory read and write operations to a bank are not allowed while the bank is being refreshed, it is very important to minimize such performance degradation due to refresh. In this paper, we propose a per-bank refresh method with adaptive early termination called Adaptive Early Termination Refresh (AETR). AETR allows both the refresh period and the size of each row group to be non-uniformly determined in order to increase the efficiency of per-bank refresh and reduce energy consumption. By using the proposed AETR, the average weighted speedup is improved by about 6.4% and the energy consumption is reduced by about 51% compared to the conventional all-bank refresh scheme.

6. ACKNOWLEDGMENTS

This work was supported by the Technology Innovation Program (10076583, Development of free-running speech recognition technologies for embedded robot system) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

7. REFERENCES

[1] Kim, K., and Lee, J. 2006. A New Investigation of Data Retention Time in Truly Nanoscaled DRAMs. *IEEE Electron Device Letter*, vol. 30 (July. 2009), 846-848. DOI=https://doi.org/10.1109/LED.2009.2023248

[2] Agrawal, A., O'connor, M., Bolotin, E., Chatterjee, N., Emer, J., and Keckler, S. 2016. CLARA: Circular Linked-List Auto and Self Refresh Architecture. *MEMSYS'16* (Oct. 2016), 338-349. DOI= https://doi.org/10.1145/2989081.2989084

[3] Lee, M. K., and Chung, K. S. 2018. Early termination refresh to reduce refresh overhead. *IET Electronics Letter*, vol. 54 (Feb. 2018), 142-144. DOI= http://dx.doi.org/10.1049/el.2017.3843

[4] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D., and Wood, D. A. 2011. The Gem5 simulator. *ACM*

SIGARCH Computer Architecture News, vol. 39 (May. 2011), 1–7. DOI=https://doi.org/10.1145/2024716.2024718

[5] Rosenfeld, P., Balis, E. C., and Jacob, B. 2011. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Computer Architecture Letter*, vol. 10 (Jan. 2011), 16-19. DOI=https://doi.org/10.1109/L-CA.2011.4

[6] Eyerman, S., and Eeckhout, L. 2008. System-Level Performance Metrics for Multiprogram Workloads. *IEEE Micro*, vol. 28 (May. 2008), 42-53. DOI=https://doi.org/10.1109/MM.2008.44

[7] '8Gb: x4, x8, x16 DDR3 SDRAM'. Available at www.micron.com/products/datasheets, accessed 16 September 2017.