Parallelizing Bank-level Fine Granularity Refresh with Column Access Operation using Split Row Buffer

Minkyu Lee and Ki-Seok Chung*
Department of Electronic and Computer Engineering, Hanyang University
Seoul, Republic of Korea
{hanlovelan, *kchung}@hanyang.ac.kr

***Abstract*—DRAM requires refresh operations to maintain data integrity. As the density of DRAM increases, system performance degradation caused by refresh is getting more serious because column access commands are blocked during the refresh operation. In this paper, we propose a novel refresh method called Bank-Level Fine Granularity Refresh (BFGR) that minimizes the performance degradation due to refresh. In BFGR, the refresh granularity is finer by applying Fine Granularity Refresh (FGR) of DDR4 at the bank level to reduce the stall time that column access commands are blocked by a refresh operation. If a special row buffer called split row buffer is used, a column access operation and a refresh operation can be simultaneously carried out, and BFGR employs this split row buffer to parallelize column access operations with a refresh operation. Our evaluation for DRAM devices with a 64ms retention time shows that the performance improvement of the proposed scheme over the conventional DRAM system on memory-intensive applications is 13.6% for 16Gb devices and 17.1% for 32Gb devices on a multi-core system, respectively.**

Keywords—Memory System;DRAM; Refrseh

I. INTRODUCTION

Each memory cell of a dynamic random-access memory (DRAM) is composed of a capacitor and an access transistor to store one-bit data, and it is mainly used for the main memory of computer systems. Because the charge in a DRAM cell leaks over time, DRAM is required to periodically carry out a refresh operation that recharges the capacitor to maintain data integrity.

The minimum time that a logical value of the weakest memory cell in DRAM is retained, is defined as the retention time in the JEDEC-DDR standard [1-4]. To keep the retention time of all memory cells, refresh commands are issued periodically. Because the memory controller cannot issue any memory request during a refresh operation, the refresh operation degrades system performance by increasing the response time of memory requests because requests are pended in the memory controller.

A basic refresh scheme is *all-bank refresh* that all banks in a rank are refreshed. Because the memory controller cannot issue a memory request command to any bank during the all-bank refresh operation, the system performance degradation by all-bank refresh gets worse as the DRAM device density increases. To reduce the overhead of all-bank refresh, JEDEC standards provide options for LPDDR DRAM and DDR4 DRAM; *per-bank refresh* (*bank-level refresh*) for LPDDR DRAM and *fine granularity refresh* (FGR) for DDR4 DRAM [2-4]. The two refresh schemes reduce the refresh granularity and increase the refresh frequency. Since the amount of bits to be refreshed by a single refresh command decreases in the two refresh schemes, latencies of the two refresh operations are also lower than all-bank refresh. However, although the refresh latency is reduced by the bank-level refresh and FGR, the reduced refresh latency is still high. As a result, the blocked time that any memory request cannot be issued to the DRAM device will be high and performance degradation due to refresh will be quite considerable.

In this paper, to mitigate the refresh overhead, we propose a refresh scheme called bank-level fine granularity refresh (BFGR). The proposed scheme employs a special row buffer called *split row buffer* where a conventional row buffer is split into a row access buffer and a column access buffer to enable a column access command to be serviced during a refresh operation [5]. In the BFGR scheme, the refresh latency is lower than conventional refresh schemes by applying FGR per bank to reduce the refresh granularity further. Our evaluation for 16Gb and 32Gb DRAM devices with a 64ms retention time shows that the performance improvement of the proposed scheme over the conventional DRAM system for memory intensive applications is 13.6% and 17.1% on a multi-core system, respectively.

II. MOTIVATION

Split Row buffer DRAM (SRDRAM) is proposed to hide the latency of activate and precharge operations [5]. Fig. 1 shows the structure of SRDRAM. SRDRAM separates the row buffer into row access buffer (RAB) and column access buffer (CAB). RAB operates row-level operations such as activate and precharge operations, and CAB operates column-level operations such as read and write operations. SRDRAM reduces the activate and precharge overhead by activating a new page or precharging in RAB while executing a column access command in CAB. Also, SRDRAM can serve a column

access command in CAB while refreshing in RAB. However, because the refresh latency of SRDRAM increases as the DRAM device density increases, the system performance degradation gets worse for a bigger device density. SRDRAM can execute a refresh operation and column access commands that have the same row address with the page in CAB in parallel, but it is not possible to activate a new page while refreshing, as shown in Fig. 2. Therefore, the number of issuable column access commands in parallel with refresh is limited. After all issuable commands have been issued during a refresh in SRDRAM, the remained commands are blocked until the refresh operation is completed. The stall time increases as the refresh latency increases in proportion to the DRAM device density. To reduce the stall time, we should reduce the refresh latency to reduce the performance loss.

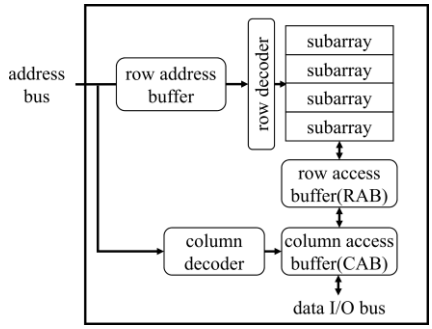

Fig. 1. Bank Structure of Split Row buffer DRAM (SRDRAM)

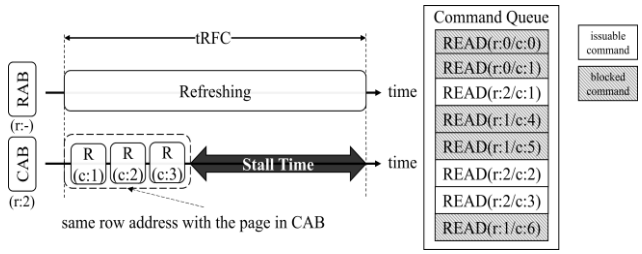

Fig. 2. Service timeline of column access command along with refresh command in SRDRAM

III. Proposed Refresh Sememe

To reduce the refresh latency, we propose a new refresh scheme called bank-level fine granularity refresh (BFGR). BFGR is a refresh scheme that applies FGR at the bank level. BFGR refreshes cells at the bank level like the bank-level refresh but the refresh granularity is finer than the conventional bank-level refresh because FGR is supported per bank. BFGR supports various modes which are similar to modes in FGR; BFGR 1x, BFGR 2x, and BFGR 4x. BFGR 1x has the same granularity as the existing bank-level refresh, and the refresh units of BFGR 2x and BFGR 4x are one half and one quarter of the bank-level refresh, respectively. In BFGR, the reduced refresh latency reduces the stall time and thus alleviates the performance loss due to the refresh operation

A. BFGR timing parameters

In order to accurately estimate the performance of the proposed method, timing parameters for BFGR are determined based on timing parameters of the bank-level refresh and those of FGR. Table I shows the refresh latency of the bank-level refresh in LPDDR4 [4] and FGR in DDR4 [2]. The bank-level refresh latency (*tREFpb*) is approximately 50% of all-bank refresh latency (*tREFab*). On the other hand, the latency for REF 2x of the 8Gb DRAM device is the same as that for REF 1x of the 4Gb DRAM device. The latency for REF 4x of the 8Gb DRAM device is the same as that for FGR 2x of the 4Gb DRAM device and that for REF 1x of the 2Gb DRAM device. This similarity is resulted from the fact that FGR latencies are proportional to the number of rows to be refresh. We calculate the refresh latencies of BFGR for 16Gb and 32Gb DRAM devices [6] are determined based on the latency ratios of the all-bank refresh and the bank-level refresh in LPDDR, and the FGR latency in DDR4. Table II shows the refresh latency of BFGR. For example, the latency for BFGR 1x of a 32Gb DRAM device is half of that of all-bank refresh of a 32Gb DRAM device, and the latency for BFGR 2x of a 32Gb DRAM is the same as that for BFGR 1x of a 16Gb DRAM.

TABLE I. Refresh latency for conevnional DRAM

DDR Type	Refresh Mode	DRAM device density		
		2Gb	4Gb	8Gb
DDR4	REF x1	160 ns	260 ns	350 ns
DDR4	REF x2	110 ns	160 ns	260 ns
DDR4	REF x4	90 ns	110 ns	160 ns
LPDDR4	REFab	160 ns	180 ns	280 ns
LPDDR4	REFpb	60 ns	90 ns	140 ns

TABLE II. Latency for BFGR

Refresh Mode	DRAM device density			
	4Gb	8Gb	16Gb	32Gb
REFab	260 ns	350 ns	530 ns	890 ns
BFGR x1	160 ns	175 ns	265 ns	445 ns
BFGR x2	-	-	175 ns	265 ns
BFGR x4	-	-	130 ns	175 ns

B. BFGR Power Consumption

Table 3 shows the amount of current of the activate and refresh operations. The amounts of current of the activate operation and all-bank refresh are obtained by referring to the values given in the Micron's 4Gb DDR3 DRAM datasheet [7]. According to Micron's DDR4 DRAM [8], the amounts of current for REF 2x and REF 4x are less than that for REF x1 by -14% and -33%, respectively. According to the LPDDR2 datasheet [9], the current of bank-level refresh is approximately 30% of that of all-bank refresh. By taking operational similarities into account, the current of BFGR 1x is assumed to be approximately 30% of all-bank refresh, and BFGR 2x and 4x are assumed to be 86% and 66% of BFGR 1x, respectively.

The JEDEC-DDR standard specifies several AC parameters to limit the power consumption of the DRAM. One of them,

defined as four-active window (tFAW), limits the number of activate operations to at most four within a window. Unlike conventional DRAMs, SRDRAM performs both the activation and refresh operations simultaneously. Therefore, to prevent the power consumption of SRDRAM from exceeding the limit of the power supply network, the activate and the BFGR operations are limited based on tFAW. Table III shows that the activate current and the BFGR current are similar. Therefore, we limit the maximum number of activation operations to three, not four, during the refresh operation.

TABLE III. POWER PARAMETERS FOR BFGR

Operation	Current (mA)	Operation	Current (mA)
ACT	65	BFGR	210
BFGR x1	63	BFGR x2	54
BFGR x4	42		

C. Implementation Cost of BFGR

BFGR is a per-bank refresh based on FGR. At the bank level, the refresh row address decoder is the same as DDR4 because BFGR is equivalent to FGR in terms of the number of refreshed rows per bank. To implement BFGR, only the logic circuit to select banks to be refreshed should be added. With this circuit, the bank address is sent through a refresh command just like per-bank refresh of LPDDR and the selected bank is refreshed. Therefore, the implementation overhead of BRGR is minimal.

TABLE IV. SIMULATION PARAMETERS

Processors	4 cores, 2GHz, Out-of-order, 32-enty inst. windows
L1 Cache	64B cache line, 32KB inst./data cache per core, 2-way, LRU, 2 cycles
L2 Cache	64B cache line, shared 512KB, 8-way, LRU, 20 cycles
DRAM Controller	FR-FCRS scheduling policy, 32-enty command queue per bank, 1 channel
DRAM device	DDR3-1600 [7],1 rank, 8 banks per rank, latency: 13-13-13ns (tRP-tRCD-tCL)
Refresh setting	tRFCab=530/890ns for 16/32Gb DRAM [6] device, tREFIab = 7.8μs

TABLE V. WORKLOAD MPKI

MPKI	Workloads
> 4	mcf, lbm, bwaves, zeusmp, bzip2, leslie3d, sjeng, libaunantum, milic, aster
<4	soplex, omnetpp, gobmk, hmmer, perlbench, gromacs, calculix, h264ref, sphinx3, povray, named, gcc, GemsFDTD

IV. EXPERIMENTAL ENVIRONMENT

To conduct performance evaluation of the proposed scheme, we have combined two of widely used architecture simulators, gem5 [10] and DRAMsim2 [11] with some modifications to implement the proposed methods. Table IV shows simulation parameters for the target processor and the DRAM. We use a 16Gb and a 32Gb future DRAM devices to evaluate the performance [6]. Used DDR3 parameters [7] and BFGR parameters are summarized in Table II. We evaluated our system with a 64ms retention time, which is a very common value for commercial DDR DRAMs. Benchmarks were selected from the SPEC CPU 2006 benchmark suite [12], and they were executed with one billion instructions. Table V shows that an application is classified into memory intensive and memory non-intensive based on miss per kilo instruction (MPKI). A metric called Weighted Speed-up (WS) [13], was used to evaluate the performance of the proposed refresh scheme.

V. EVALUATION RESULTS

The proposed scheme is compared with the following memory devices: 1) a conventional DRAM, 2) SRDRAM that does not issue column access commands during all-bank refresh ($NOISSUE_{SRDRAM}$), 3) SRDRAM with all bank refresh that issues column access commands during all-bank refresh ($REFab_{SRDRAM}$), 4) SRDRAM with BFGR ($BFGR_{SRDRAM}$).

Fig. 3 shows the normalized weighted speedup of the proposed schemes over the conventional DRAM. The performance improvements of $NOISSUE_{SRDRAM}$ for the 16Gb density on memory-intensive applications (MPKI > 4) is 9.9%. This performance improvement is mainly due to SRDRAM that can perform row access operations such as activate and precharge operations and column access operations such as read and write operations in parallel. The performance improvements of $REFab_{SRDRAM}$ for the 16Gb device is 11.4%. Since $REFab_{SRDRAM}$ can serve column access commands in parallel with a refresh operation, the system performance degradation due to refresh is reduced. The performance gain of $BFGR_{SRDRAM}$ is larger than $REFab_{SRDRAM}$ because the refresh latency of $BFGR_{SRDRAM}$ is shorter than $REFab_{SRDRAM}$, and therefore the stall time is further reduced. The performance improvements of $NOISSUE_{SRDRAM}$, $REFab_{SRDRAM}$ and $BFGR_{SRDRAM}$ for the 32Gb DRAM device are 10.9%, 11.7% and 17.1%, respectively. The performance improvements of SRDRAM with the technique to reduce the refresh overhead are larger for the 32Gb DRAM device because the system performance degradation due to refresh of the conventional DRAM for the 32Gb DRAM device is larger than that for the 16Gb DRAM device.

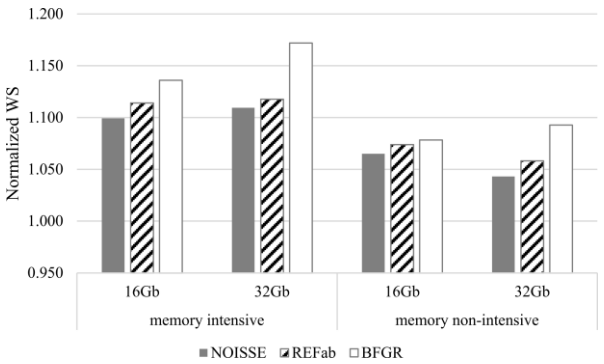

Fig. 3. Normalized Weighted Speed-up (WS) of a multi-core system for a various DRAM device densities

In the case of memory non-intensive applications (MPKI < 4), the performance improvements of $NOISSUE_{SRDRAM}$, $REFab_{SRDRAM}$, and $BFGR_{SRDRAM}$ with the 16Gb device are 6.5%, 7.4% and 7.8%, respectively, Those for the 32Gb device are 4.3%, 5.8%, and 9.3%, respectively. System performance improvements in memory non-intensive applications are lower than those in memory-intensive applications because SRDRAM needs sufficient memory requests to serve column access commands in parallel with row access commands.

VI. Conclusion

To reduce the refresh overhead, we propose a new scheme called *Bank-Level Fine Granularity Refresh* (BFGR) in this paper. This scheme employs a special row buffer called *split row buffer* Split Row-buffer DRAM (SRDRAM) has a unique feature that it can simultaneously carry out a column access command and a row access command. In addition, BFGR reduces the refresh latency by reducing the refresh granularity in order to reduce the stall time that there is no remaining issuable column access commands during a refresh operation in SRDRAM. Our evaluations for 16Gb and 32Gb DRAM densities with a 64ms retention time show that the proposed scheme on memory-intensive applications provides 13.6% and 17.1% performance improvement on average over the conventional DRAM system, respectively.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R7119-16-1009, Development of Intelligent Semiconductor Core Technologies for IoT Devices based on Harvest Energy).

References

[1] JEDEC Standard, DDR3 SDRAM STANDARD, JESD79-3F, July 2012

[2] JEDEC Standard, DDR4 SDRAM, JESD79-4A, November 2013.

[3] JEDEC Standard, Low Power Double Data Rate 3 (LPDDR3), JESD209-3C, August 2015.

[4] JEDEC Standard, Low Power Double Data Rate 4 (LPDDR4), JESD209-4A, November 2015.

[5] M. Lee and K. Chung, “High performance DRAM architecture with split row buffer,” *Electron. Lett*., vol. 52, pp. 1844-1845 , 2016.

[6] I. Bhati, M. T. Chang and Z. Chishti, “DRAM refresh mechanisms, penalties, and trade-offs,” *IEEE Trans. Comput*. vol. 39, pp. 108-121. 2016

[7] Micron Technology, 4Gb: x4, x8, x16 DDR3 SDRAM, December 2014.

[8] Micron Technology, 8Gb: x4, x8, x16 DDR4 SDRAM, January 2017.

[9] Micron Technology, 168-Ball, Single-channel Mobile LPDDR2 SDRAM, July 2014.

[10] N. Binkert et al., “The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, pp. 1-7, 2011.

[11] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, “DRAMSim2: A cycle accurate memory system simulator," *IEEE Computer Architecture Letters*, vol. 10, pp. 16--19, 2011.

[12] SPEC CPU 2006. https://www.spec.org/cpu2006.

[13] S. Eyerman and L. Eeckhout, “System-level performance metrics for multiprogram workloads," *IEEE MICRO*, vol. 28, pp. 42-53, 2008.