An Effective DRAM Address Remapping for Mitigating Rowhammer Errors

Moonsoo Kim, Jungwoo Choi, Hyun Kim, and Hyuk-Jae Lee

Abstract—A rowhammer error represents a loss of data stored in a DRAM cell caused by electromagnetic interference due to repetitive access to the same and/or adjacent rows. Due to the concentrated occurrence of rowhammer errors in specific rows and columns, these errors cannot be corrected by the conventional error correcting code (ECC) commonly used in DRAM devices. Previous techniques avoid these errors by having additional refresh operations that require additional hardware resources and/or power consumption. This paper proposes a different approach to handle rowhammer errors by distributing them across different DRAM rows and columns so that the attack cells are not concentrated on specific rows and columns. To this end, the distribution of rowhammer errors is observed with experiments using several commercial DRAM devices by employing state-of-the-art rowhammer attack techniques. The observation of the rowhammer errors concentrated in specific rows and columns underlies the proposal of an effective DRAM address remapping scheme for re-distribution of rowhammer errors. By using different address mappings to different chips and arrays in a DIMM, the proposed remapping effectively distributes errors over different rows and columns. As a result, the proposed remapping scheme decreases the possibility of multiple errors in a single word, and consequently, reduces uncorrectable errors under single error or single symbol correcting ECC. Experimental results with commercial DIMMs show that the proposed scheme reduces uncorrectable errors by about 95 percent while incurring a small additional hardware cost.

Index Terms—DRAM, rowhammer error, fault tolerance, reliability

1 Introduction

A rowhammer error represents a loss of data stored in a DRAM cell. This is caused by repetitive access to the same or adjacent rows that creates electromagnetic interference amongst these rows. Rowhammer errors do not occur frequently in DDR2 devices; however, they become serious for DDR3 devices because of a small feature size and high operating speed [1], [2], [3]. As a result, DDR4 and next-generation DDR devices may become even more vulnerable to rowhammer errors because DRAM manufacturers are continuously improving their manufacturing process and operation speed [4]. The difficulty in handling rowhammer errors lies in the fact that these errors cannot be corrected by conventional error correcting codes (ECCs) because they are likely to be concentrated in a specific word. Furthermore, recent developments in [5], [6], [7], [8], [9] makes use of rowhammer errors for malicious attacks on the security of systems, increasing the urgency of developing of a method to avoid or reduce rowhammer errors.

- *M. Kim, J. Choi, and H.-J. Lee are with the Inter-University Semiconductor Research Center, Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea. E-mail: {kimms213, choijw8525, hyuk_jae_lee}@capp.snu.ac.kr.*
- *H. Kim is with the Department of Electrical and Information Engineering, The Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, Korea. E-mail: hyunkim@seoultech.ac.kr.*

Manuscript received 26 June 2018; revised 2 Mar. 2019; accepted 13 Mar. 2019. Date of publication 24 Mar. 2019; date of current version 16 Sept. 2019. (Corresponding author: Hyun Kim.)
Recommended for acceptance by P. Girard.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2019.2907248

Since rowhammer errors in DRAM devices are observed in [1], research activities about rowhammer errors have been undertaken in two directions: one attempting to avoid or reduce rowhammer errors and the other making use of rowhammer errors to attack the security of a system using DRAM devices. The research in the direction toward security attack shows that a double-sided rowhammer attack can increase the error rate by alternately activating the rows above and below the victim (i.e., target) row [8]. Cacheline-based rowhammer attack [7] is another technique to increase the error rate by attacking rows in a cacheline-wise rather than page-wise fashion. Repeatedly applying these techniques to specific regions of a DRAM device may result in intensive rowhammer errors in the regions. In [5], [6], [7], [9] , rowhammer attacks make it possible to generate errors in a page mapping table, which holds the critical information to gain access privileges to a system. Failure to protect this critical information may result in a fatal flaw in system security.

A conventional technique to correct an error is to use an ECC [10]. Because a DRAM device seldom suffers from errors in normal operation, the ECC for a DRAM, in general, is not required to correct multiple errors. On the other hand, rowhammer errors often occur in multiple bits in the same word. Therefore, a conventional ECC may not be capable of correcting rowhammer errors [11]. Instead of relying on ECC, PARA [1], a probabilistic row refresh technique, is proposed to use a pseudo-random number generation and to probabilistically refresh DRAM rows whenever they are activated. This technique is simple to implement but it requires a large number of refresh operations, which may be unnecessary if no attack is attempted. In [12], a probabilistic technique to keep track of possible victim rows is proposed. This

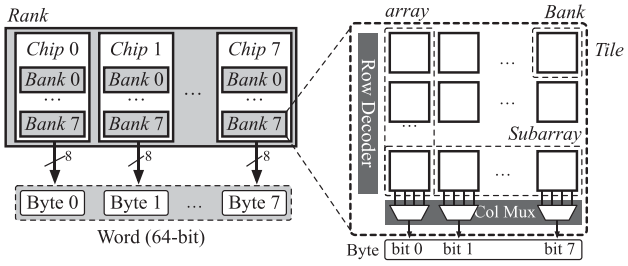

Fig. 1. The diagram of the DRAM rank, chip, bank and subarray.

technique, called PRoHIT, outperforms PARA by utilizing previous access history. Another technique to avoid rowhammer errors is to count the number of activations per row using per-row counters, and to refresh adjacent rows whenever the counter reaches a threshold [13]. The advantage of this technique lies in the reduction of unnecessary refresh operations although the per-row counters may increase the hardware cost. To reduce the hardware cost, a tree-structured counter implementation is proposed in [14]. It dynamically assigns the counters to the rows which are accessed frequently, and thereby reducing the number of counters.

In summary, the previous techniques attempt to avoid rowhammer errors by aggressively refreshing the rows with abnormally high activation. Consequently, these techniques demand additional power consumption for refresh operations and hardware overhead for the storage of activation information.

To mitigate rowhammer errors without additional refresh overhead, this paper proposes a new technique based on DRAM address remapping. Address remapping, or scrambling, is a widely used technique which remaps row and column addresses within a DRAM subarray [15], [16], [17]. It is previously used for two reasons. First, it improves the hardware efficiency of the row decoder and column mux [15] in a DRAM. Second, it hides DRAM address space from users, and thereby strengthening its tolerance against security attack. Recently-proposed methods for security attacks, however, successfully finds the remapped address, and consequently, generates rowhammer errors even for an address-remapped DRAM [18].

This paper presents a novel two-level DRAM address remapping to reduce rowhammer errors that cannot be corrected by an ECC. The most important property of the proposed remapping is to use different remapping strategies for different DRAM chips in a DIMM as well as the arrays even inside a chip. The first level of the two-level remapping remaps an address in the chip level to make the address of each chip to be different. The second level remaps an address inside a chip, making every bit in a word to be originated from different address. As a result, rowhammer errors are distributed evenly over multiple words, and thereby allowing the word-wise ECC to correct the errors. The contributions of this paper are summarized as follows.

- An address remapping is proposed to reduce rowhammer errors by delivering different addresses to different DRAMs in a DIMM. The remapping is made in two levels so that its benefit is maximized.
- For the selection of an effective remapping, two main causes of the rowhammer errors are observed from experimental results. Then, these causes are mathematically formulated as the necessary conditions to reduce rowhammer errors.
- Mathematical analysis shows that the proposed remapping significantly increases the difficulty of rowhammer attack. The reason is because the probability of uncorrectable errors (UEs) within a refresh interval is decreased by 71 times.
- An effective implementation of the proposed remapping is presented with the evaluation of the overhead. Layout simulation shows that the hardware overhead is negligible to implement the row decoder and column mux for the proposed remapping.

The rest of the paper is organized as follows: Section 2 introduces the background of DRAM organization and the characteristics of rowhammer errors. Section 3 presents the experimental observations of the distribution of the rowhammer errors. In Sections 4 and 5, the proposed two-level remapping scheme is presented. Section 6 demonstrates possible attack scenarios. Section 7 shows the experimental results that show a reduction of rowhammer errors by the proposed remapping. Finally, Section 8 concludes the paper.

2 Background

2.1 DRAM Organization and Access Unit

Fig. 1 shows the typical organization of a DRAM module which consists of multiple DRAM chips. In this module, eight 4-Gbit DRAM chips with an 8-bit I/O channel are combined to make a module. As a result, this DRAM module has a 32-Gbit storage capacity with a 64-bit I/O bandwidth. The hierarchy of DRAM chips operating together to offer a wider bandwidth is called a rank. The rank consists of multiple banks distributed over DRAM chips. Each chip inside a bank shares common command, address and data channels so that all data in the same bank are accessed simultaneously although they are stored in different chips. A single bank in a chip consists of matrix of tiles, which are the basic access units. A row of tiles, termed as a subarray, is always accessed together. Typically, each bank in a chip includes 64 subarrays. A column of tiles, termed as *array* hereafter, shares a column multiplexer. This means that the array corresponds to a certain bit location in a byte.

In a rank structure such as that shown in Fig. 1, the read access is initiated by delivering bank, row and column addresses to the rank. As every chip shares the same addresses, the row and column addresses are the same for all DRAM chips. In each chip, a single row is selected by the row decoder and the entire row is read from the subarray and stored in a row buffer. Next, the column address is driven to column multiplexers and the selected column is output from the bank. In the example in Fig. 1, eight columns are selected from each DRAM chip. For eight DRAM chips, a total of 64-bit data is transmitted to/from a DRAM module, as shown in the left of Fig. 1. These 64-bit data correspond to the basic unit of DRAM access, which is called a word.

Normally, the ECC of the DRAM is applied to each word. Parity bits of the ECC are stored in an additional chip or reserved area of the DRAM module. They are accessed and transmitted together with the DRAM data to a memory

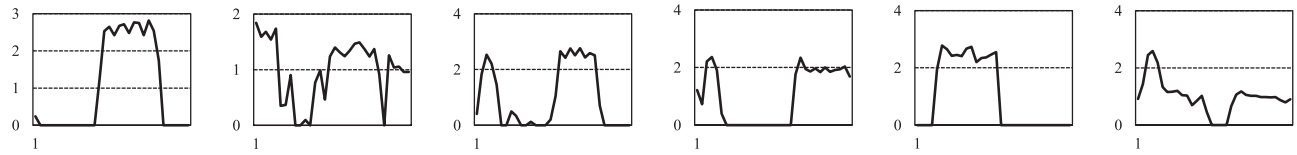

Fig. 2. The relative frequency (y-axis) of rowhammer errors per word, with respect to subarray index (x-axis).

controller. The controller decodes the ECC and detects/corrects an error if it exists.

2.2 Charicteristics of Rowhammer Errors

Rowhammer errors take place when a specific row is activated repeatedly. Repetitive activation causes the fluctuation of the wordline voltage which results in electromagnetic interference affecting the adjacent rows, and thereby causing a data loss. The row experiencing repetitive activation is called an *aggressor row* and the adjacent row experiencing errors is called a *victim row*. Because the electromagnetic interference is larger for the nearby rows, the error rate is larger for the rows close to the aggressor row. In [1], it is reported that most rowhammer errors take place in the rows just above and below the aggressor row.

The current DRAM organization is vulnerable to rowhammer errors because all DRAM chips in a rank share DRAM addresses. For example, assume that row 1 is repeatedly accessed. Row 1 is accessed for all the arrays in eight chips, making rows 0 and 2 are vulnerable in all arrays. When rows 0 or 2 are accessed later, the errors from all arrays are concentrated in a single word, and thereby making it impossible for an ECC to correct the errors.

It is difficult to avoid this rowhammer attack even when a conventional DRAM address remapping is adopted. Even though the addresses are remapped, there still exist physically adjacent rows. For example, there exist rows i, j, and k that are physically adjacent in the remapped addresses although they are not in a logical address space. In this case, rows i and k suffer from rowhammer errors if row j is attacked. Errors are concentrated in rows i and k for all DRAM chips because they are remapped in the same manner in the conventional remapping.

In double-sided rowhammer attack, the possibility of error concentration increases significantly. This attack is a technique to increase the probability of rowhammer errors, made by repetitive activations of two aggressor rows that are next to the victim row. For example, rows 0 and 2 are the aggressor rows to attack row 1 which is the victim row. With the repetitive activations of rows 0 and 2, the probability of rowhammer errors in row 1 is larger than that with the activation of a single row (either row 0 or row 2). For all DRAM chips, the victim row is row 1 in which errors take place. Therefore, the concentrated errors in row 1 make the accessed word having multiple errors when any word from row 1 is accessed. Consequently, the multiple errors in the accessed word make it difficult to correct them even with an ECC.

3 Rowhammer Error Distribution

3.1 Vulnerable Rows and Columns

The first step to reduce the possibility of rowhammer errors is to observe the characteristics of rowhammer errors. It is not easy to create rowhammer errors because they do not occur frequently. Therefore, the dedicated program to generate rowhammer attaks is executed for the creation of rowhammer errors. Experiments are conducted with six commercial DIMMs to execute the dedicate program repeatedly and then to observe the distribution of rowhammer errors. These DIMMs consist of eight DRAM chips with 8-bit width and 2-4 GB capacity and they are installed in a host PC with an x86 CPU. The rowhammer attack program used in the experiment is based on [8], and it is modified to make double-sided, unit based attacks as in [7]. To this end, the conventional address remapping is revealed as explained in [18]. For every unit in each memory module, the number of rowhammer errors is counted, which is used to create the distribution of rowhammer errors.

Fig. 2 shows the *relative frequency* of rowhammer errors in each word, which represents the number of rowhammer errors divided by the average number of errors for all words. Modules A from F represent the six commercial DIMMs which are used in experiments, respectively. In Fig. 2, the horizontal axis represents the subarray index. The relative frequency varies significantly depending on the subarray index. The maximum relative frequency is greater than 2 for all modules, which means that a specific subarray exhibits 2 times more errors than the average. Also, there are many subarrays where rowhammer errors do not occur at all.

In Fig. 3, the horizontal axis represents the column addresses, clustered to 128 bins. The relative frequency varies significantly depending on the column address. For example, module B varies most significantly, ranging from 0.2 to 2. Fig. 4 shows the relative frequency of every unit with respect to a row address. The variation of the relative frequency is even larger than that in Fig. 3 such that the minimum

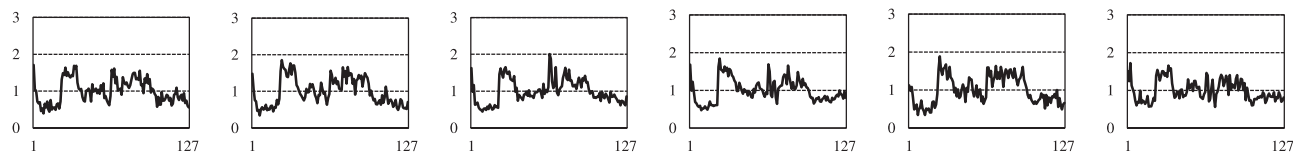

Fig. 3. The relative frequency (y-axis) of rowhammer errors per word, with respect to column address (x-axis).

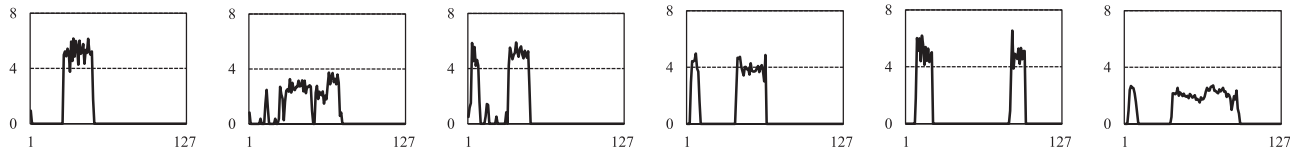

Fig. 4. The relative frequency (y-axis) of rowhammer errors per word, with respect to row address (x-axis).

frequency is 0 whereas the maximum is almost 7. Moreover, in most modules, there is a clear distinction between no-error rows and multiple-error rows.

The experimental results shown in Figs. 2, 3, and 4 indicate that each module includes specific rows or columns which are very vulnerable to rowhammer errors. It has been reported that there exist certain DRAM cells of which retention times are shorter than the other cells. These cells are located close to each other in certain regions of a DRAM layout [19], [20]. The short-retention times may be caused by the variation in manufacturing process creating a higher vulnerability in certain rows/columns than others. Another reason might be defects in the circuit design steps. No matter what the reason is, the addresses of the vulnerable rows or columns may differ from one module to another. Nonetheless, the important fact is that all the modules used in the experiment include vulnerable rows or columns. If these vulnerable rows or columns are attacked repetitively, they may suffer from a data loss due to rowhammer errors which cannot be detected or corrected by a conventional ECC.

3.2 Statistical Analysis for the Redistribution of Rowhammer Errors

This section presents the possibility of the redistribution of rowhammer errors and the reduction of UEs which are the errors that cannot be corrected by an ECC. Table 1 shows the number of words in which UE is generated, for all words in a module. To help understanding, an example is given. Column X represents the number of UEs, that is the number of words that include more than one error among the errors in T (i.e., the errors observed from experiments). On the other hand, column Y represents an estimated number of UEs with the assumption that all the errors in column T are randomly distributed across all the words in a DRAM. The value of Y corresponds to the expected number of UEs assuming the probability of uniform random distribution. A large value of X/Y indicates that errors are concentrated on certain words of a DRAM. Therefore, this also implies that there exists a large opportunity to reduce the UEs by randomly redistributing the errors.

TABLE 1
The Comparison of Real and Expectation Value of UEs

	Number of Errors (T)	Number of UE (X)	Expected Value of UE (Y)	Ratio (X/Y)
A	10790	110	4.25	25.90
B	14977	123	5.57	22.09
C	89272	7385	294.29	25.09
D	136008	12450	532.82	23.37
E	6042	35	0.7	50.00
F	13758	28	2.78	10.09

As in the fourth column of Table 1, the ratio varies from 10.09 to 50.00, which means that the differences between the real values and the expectation values are large. This result indicates that rowhammer errors are concentrated on certain words and re-distribution of rowhammer errors to different rows or columns may significantly reduce UEs. In the next section, an effective way to re-distribute rowhammer errors is to be discussed.

4 DRAM Address Remapping

As described in Sections 2.2 and 3, there are two main causes of the concentration of rowhammer errors in specific rows and columns.

1) *Direct adjacency:* Most rowhammer errors occur in the directly adjacent rows, i.e., the row prior to and after the aggressor row.
2) *Vulnerable row/column:* There exist certain rows or columns that are vulnerable to rowhammer errors.

These two causes seem somewhat obvious, but they are emphasized here because they are to be used in the design of the proposed remapping scheme discussed in the subsequent subsections.

To reduce UEs by rowhammer attack, the section presents a proposal of DRAM address remapping. To this end, conventional DRAM address mapping is modified to re-distribute concentrated rowhammer errors. The essence of the proposed address mapping lies in the application of different address mappings for different chips and arrays in a memory module. In the next subsections, effective DRAM address remapping is analyzed considering the above two causes.

DRAM address remapping hereafter refers to the operation of converting a single input address to a set of remapped addresses. The input address can be both a row and a column address. The remapped addresses are used as an actual physical addresses for all chips and arrays.

4.1 Remapping Matrix

DRAM address remapping is mathematically expressed as a *remapping matrix*, R, as shown in Fig 5. The column of the matrix represents the remapped address in each chip or

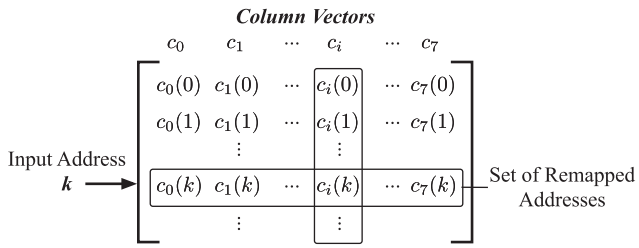

Fig. 5. The diagram of the remapping matrix.

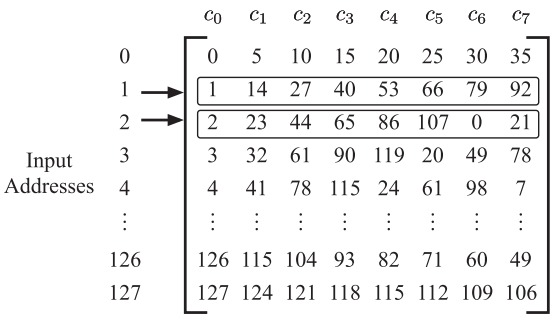

Fig. 6. An example of an appropriate remapping matrix.

subarray. For example, the values in the first column represent the addresses used in chip 0, those in the second column are for chip 1, and so on. When address k is given as the input to chip i, it is remapped as $c_i(k)$ by the remapping matrix.

The kth row of the matrix represents the set of the remapped addresses for all chips when the input address is k. This means that the row index of the matrix corresponds to the input address. The input address can be either a row or column address. To handle both cases in a single matrix without a loss of generality over various modules, the number of rows of the matrix is limited to 2^n. Therefore, the input address must be smaller than 2^n. To ensure this condition, only n least significant bits (LSBs) of the input address are used by the remapping and the remaining most significant bits (MSBs) are used as the original values. In the next, an example in Fig. 6 is used to explain the remapping operation that is represented by the remapping matrix.

Example 1. The remapping matrix shown in Fig. 6 represents the address remapping operation. For simplicity, only 128 input addresses are considered in this example, which means n is 7.

When the input address is 129, the value of the 7-bit LSBs is 1, and thereby making the input address k equal to 1. Therefore, row 1 of the remapping matrix is selected. From chip 0 to 7, (1, 14, 27, 40, 53, 2, 15, 28) addresses are used, respectively.

There are two conditions that the remapping matrix R must satisfy. The first condition is given to guarantee that different addresses are generated for different chips or subarrays. To this end, the remapping matrix must have distinct column vectors. In other words, R must be composed of eight different column vectors c_i, as shown below.

$$i \neq j \rightarrow c_i \neq c_j \quad (0 \leq i, j < 8). \tag{1}$$

The second condition is to ensure that different input addresses are remapped to the different locations in a DRAM chip and subarray. Otherwise, two different addresses can be remapped to the same location, causing a fatal memory access error. In other words, if input addresses k and l are different, then the remapped addresses must be different, which is described as follows.

$$k \neq l \rightarrow c_i(k) \neq c_i(l) \quad (0 \leq k, l < 2^n). \tag{2}$$

The remapping matrix shown in Fig. 6 satisfies the two conditions.

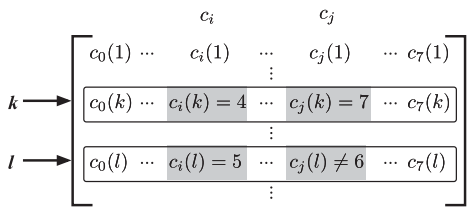

Fig. 7. The directly adjacent cause in the remapping matrix.

Example 1. (Continued) The remapping matrix in Fig. 6 satisfies (1) and (2) because all column vectors c_is are different from each other, and for every column vector, all entries are distinct, too.

In the next subsections, two causes of error concentration are mathematically analyzed and the remapping matrix is derived to resolve the two causes. For the sake of mathematical analysis, column vectors are restricted to the following form.

$$c_i(k) = (a_i \cdot k + b_i) \ \ mod \ \ 2^n. \tag{3}$$

It is obvious that the form of 3 limits the number of possible column vectors. However, there exist a sufficient number of remapping matrices that satisfy 3 and solve the two causes of rowhammer concentration. Therefore, the limitation of Equation 3 does not prevent the derivation of the column vectors to constitute a remapping matrix.

With the restriction of 3, the derivation of the remapping matrix corresponds to the derivations of a_is and b_is.

4.2 Analysis of Direct Adjacency Cause

The first cause of the rowhammer error concentration is that most rowhammer errors occur in the previous and next rows of an aggressor row. In Fig. 7, the input address is given as k and the corresponding remapped addresses are from $c_0(k)$ to $c_7(k)$. In chip i, rowhammer errors occur at adjacent rows $c_i(k) \pm 1$. Similarly, errors occur at the adjacent rows, $c_j(k) \pm 1$ in chip j. To distribute the errors in this case, $c_i(k) \pm 1$ and $c_j(k) \pm 1$ must belong to different rows in the matrix. For example, assuming that $c_i(k)$ is 4 as in Fig. 7, rowhammer errors may occur in row l because $c_i(l)$ is 5. On the other hand, since $c_j(k)$ is 7, $c_j(l)$ should not be 6 or 8 because those rows may contain errors. These conditions are formulated as below.

$$|c_i(k) - c_i(l)| = 1 \rightarrow |c_j(k) - c_j(l)| \neq 1. \tag{4}$$

Now, the remapping matrix which satisfies (4) is derived. Consider 8 different c_is of the Equation (3), where a_is satisfy the following condition:

$$((a_i + a_j) \ mod \ 2^n) \neq 0 \ \ and \ \ 0 < a_i, a_j < 2^n. \tag{5}$$

If the remapping matrix is constructed with 8 c_is which satisfy (5), condition (4) is also satisfied. This is summarized as the following lemma:

Lemma 1. *If c_i, c_j in R satisfy (5) , then (4) is also satisfied.*

Proof. By (3),

$$\begin{aligned} a_i \cdot k + b_i &= c_i(k) + q \cdot 2^n \\ a_i \cdot l + b_i &= c_i(l) + q' \cdot 2^n \end{aligned} \tag{6}$$

are derived. By subtracting both sides from each other and using $|c_i(k) - c_i(l)| = 1$ as in (4),

$$\begin{aligned}(k-l)\cdot a_i &= s\cdot 2^n \pm 1\\ \rightarrow k-l &= (s\cdot 2^n \pm 1)/a_i,\end{aligned} \tag{7}$$

are obtained. Similar processes are also applied to c_j, as below.

$$(k-l)\cdot a_j = s'\cdot 2^n + (c_j(k) - c_j(l)). \tag{8}$$

By substituting $(k-l)$ using (7),

$$(a_i s' - a_j s)\cdot 2^n = a_i\cdot(c_j(k) - c_j(l)) \mp a_j, \tag{9}$$

is obtained. Assume that $|c_j(k) - c_j(l)|$ is 1. The left-hand side of (9) is in the form of $S\cdot 2^n$ whereas the right-hand side is either $a_i + a_j$ or $a_i - a_j$. First, $a_i + a_j$ cannot be the same as the left-hand side because $a_i + a_j \neq s\cdot 2^n$ as explained in (5). Second in the case of $a_i - a_j$, inequality $-2^n < a_i - a_j < 2^n$ stands because $0 < a_i, a_j < 2^n$. Thus, the constraint of $a_i - a_j = 0$ gives the only possibility that $a_i - a_j$ can be the same as the left-hand side. However, a_i and a_j are different, and consequently, $a_i - a_j$ cannot be 0. In both cases, Equation (9) cannot be established. Since this is a contradiction, $|c_j(k) - c_j(l)| \neq 1$. □

In summary, the first cause (the directly adjacent cause) is avoided when a_is of (3) satisfy condition (5).

4.3 Analysis of Vulnerable Rows/Columns Cause

The second cause of rowhammer error concentration comes from the DRAM characteristic that certain rows or columns are more vulnerable to rowhammer errors. The remapping scheme to avoid this cause is explained with Fig. 7. When k is an input address, the remapped addresses are selected from $c_0(k)$ to $c_7(k)$. Suppose that $c_i(k)$ is a vulnerable address. To avoid rowhammer attack to $c_i(k)$ in the other chips, the remapping for address k in the other chips must be different from $c_i(k)$. This condition is formulated as follows.

$$c_i(k) \neq c_j(k) \quad (0 \leq i, j < 8). \tag{10}$$

Note that this condition is directly derived from (1).

Now, consider 8 different c_is of the Equation (3), where a_is and b_is satisfy the following equation.

$$(a_i - a_j)\cdot k + (b_i - b_j) \neq s\cdot 2^n \ (0 \leq i, j < 8). \tag{11}$$

Then, condition (11) is the sufficient condition to satisfy condition (10). In other words, condition (10) is satisfied if R is constructed with 8 c_is which satisfy (11). This is summarized as the following lemma:

Lemma 2. *If c_i, c_j in R satisfy (11), then (10) is also satisfied.*

Proof. By (3),

$$\begin{aligned}a_i\cdot k + b_i &= c_i(k) + q\cdot 2^n\\ a_j\cdot k + b_j &= c_j(k) + q'\cdot 2^n.\end{aligned} \tag{12}$$

are derived. By subtracting both side from each other,

$$(a_i - a_j)\cdot k + (b_i - b_j) = s\cdot 2^n + (c_i(k) - c_j(k)), \tag{13}$$

are obtained. Therefore, (11) must be satisfied to make $c_i(k) - c_j(k) \neq 0$. □

In summary, the second cause by vulnerable rows/columns is resolved when the a_is and b_is of (3) satisfy condition (11).

4.4 Derivation of a Remapping Matrix

Considering the above discussions, the remapping matrix which resolves two causes must satisfy:

- conditions (1) and (2), the basic conditions that the matrix must satisfy, and
- conditions (4) and (10), which are the conditions which resolve two causes.

Conditions (1) and (10) are identical, thus (2), (4), and (10) are the necessary conditions that the matrix must satisfy. By restricting column vectors in the form of (3), conditions (4) and (10) are reduced to (5) and (11), respectively. Therefore, the final conditions are (2), (5), and (11).

The final claim is formulated as the following Equation (14). If a_is and b_is of a column vector satisfy (14), then conditions (2), (5), and (11) are also satisfied.

$$\begin{aligned}a_i &= a\cdot i + a_0 \ (a = 4m,\ a_0\ is\ odd),\\ b_i &= b\cdot i \ \ (b\ is\ odd).\end{aligned} \tag{14}$$

Lemma 3. *If a_i, b_i in R satisfy (14), conditions (2), (5), and (11) are also satisfied.*

Proof. Condition (2) is checked first. For different k and l, $|c_i(k) - c_i(l)|$ is $a_i\cdot(k-l)\ mod\ 2^n$. Because a_i is an odd number as in (14), a_i and 2^n are relatively prime, resulting in $a_i\cdot(k-l)\ mod\ 2^n$ cannot be 0. Therefore, condition (2) is satisfied.

Second, condition (5) is checked. From (14), $a_i + a_j = a\cdot(i+j) + 2\cdot a_0$ is derived. Thus, condition (5) is reduced to $a\cdot(i+j) + 2\cdot a_0 \neq s\cdot 2^n$. By dividing both sides by 2, $a/2\cdot(i+j) + a_0 \neq s\cdot 2^{(n-1)}$ is derived. Since n is greater than 1, the left-hand side is odd while the right-hand side is even. Therefore, condition (5) is satisfied.

Finally, condition (11) is checked. By substituting a_i and b_i using (14), (11) turns into $(a\cdot k + b)\cdot(i-j) \neq s\cdot 2^n$. Because $a = 4m$ and b is an odd number, $(a\cdot k + b)$ is an odd number. Therefore, the left-hand side and the right-hand side cannot be equal, which means condition (11) is satisfied. □

In conclusion, the effective remapping matrix which resolves both causes of the rowhammer error concentration can be obtained by selecting a, a_0, and b that satisfy (14).

Example 1. (Continued) Fig. 6 shows an example remapping matrix when $n = 7$, $a = 8$, $a_0 = 1$, and $b = 5$. Every row of the matrix satisfies both conditions. For example, consider the relationship between input addresses 1 and 2. Since $c_0(1)$ is 1 and $c_0(2)$ is 2, the values should not differ by 1 in other columns according to condition (4). The values in other columns are (14, 23) in column 1, (27, 44) in column 2, and so on. Therefore, the value difference is obviously larger than 1. Condition (10) is also satisfied. In every row, there is no overlap among the eight values in the row.

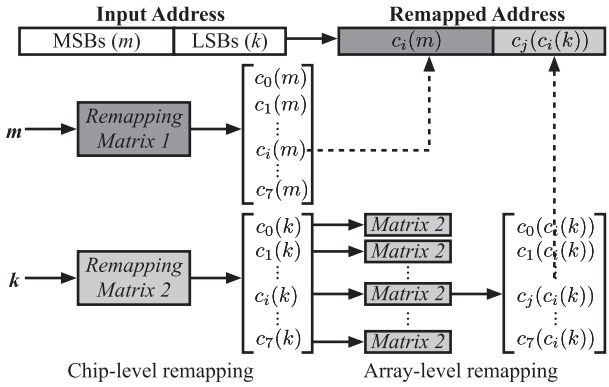

Fig. 8. The diagram of the overall two-level remapping structure.

5 Two-Level DRAM Address Remapping

5.1 Organization of Two-Level Address Remapping

The previous section addresses the question of which remapping matrix should be used to effectively distribute the rowhammer error. Extending the address remapping presented in the previous section, this section presents a two-level address remapping scheme which takes advantage of the DRAM organization.

As explained in Section 2.1, an array in a chip corresponds to a certain bit location. In the proposed organization, DRAM address remapping is performed at two levels to deliver different addresses among all chips, and arrays inside the chips. The first level is at the chip level, which delivers different addresses for different chips. The second level is at the array level, which delivers different addresses for different arrays in the chip. Chip level remapping can only distribute rowhammer errors by byte because each chip delivers a certain byte. On the other hand, array level remapping can distribute rowhammer errors by each bit inside the byte because each array corresponds to a single bit. By adopting two-level remapping, rowhammer errors are distributed not only by a byte, but also by a bit inside a byte.

The experimental results in Figs. 3 and 4 show the necessity of two-level address remapping because rowhammer errors occur in adjacent rows and columns. If only single-level address remapping is used, 8 bits (1 byte) from each chip are derived from a single row/column. If this row/column is vulnerable to a rowhammer error, it is probable to have multiple errors to occur within these 8 bits. On the other hand, two-level address remapping randomly distributes the error because the row/column varies depending on the array accessed within the chip. Experimental results also show the improvement by the two-level remapping. Fig. 15a shows that two-level address remapping reduces the UE by 39.42 percent on average compared to single-level remapping with SECDED ECC.

The proposed two-level address remapping is illustrated in Fig. 8 where the remapping matrix satisfying (14) is used twice. When a row or column address is given as an input address, it is decomposed into LSBs and MSBs parts. The LSB-part k is used for address remapping. First, k is remapped to the row k, which is $(c_0(k), c_1(k), \ldots, c_7(k))$. These values are passed on to each chip and the array-level remapping is performed. For example, in chip i, $c_i(k)$ is passed on and it is remapped to $(c_0(k), c_1(k), \ldots, c_7(k))$ by another remapping matrix. As a result, the remapped result for array j of chip i is $c_j(c_i(k))$.

The MSB-part m is also remapped. As shown in Fig. 9, a typical row decoder consists of two-level hierarchical decoders [21], [22]. The remapping in LSB-part illustrated in the above paragraph is done by applying the remapping matrix to local decoders. On the other hand, the remapping of the MSB-part is applied to the global row decoder, making errors to randomly distribute over subarrays. The remapping result of the MSB-part for chip i is $c_i(m)$. The final remapped address is obtained by concatenating $c_i(m)$ to $c_j(c_i(k))$. These processes are illustrated in the following example.

Example 1. (Continued) The two-level address remapping process is explained using the remapping matrix of Fig. 6, as an example. Suppose the input address is 514. Then m is 1 and k is 2. As the first step, the chip-level remapping of k is performed. Since k is 2, row 2 is selected, which is (2, 23, 44, 65, 86, 107, 0, 21). Each value is passed on to the corresponding chip and the array-level remapping is performed. In chip 6, the remapped value is 0, so row 0 is selected, which is (0, 5, 10, 15, 20, 25, 30, 35). Remapping of m gives (1, 14, 27, 40, 53, 66, 79, 92), which means the remapped MSB-part is 79 in chip 6. The final remapped address is obtained by concatenating the two remapped values.

5.2 Implementation Details

The two-level remapping can be effectively implemented without a significant change of the conventional DRAM architecture. A naive implementation of the structure in Fig. 8 is to design an additional hardware unit for the remapping operation. This hardware unit receives an input address, generates a different remapped address for each chip/array and then delivers the remapped address to each subarray. This implementation leads to significant overhead in the chip area and latency because a new hardware unit is added in the access path.

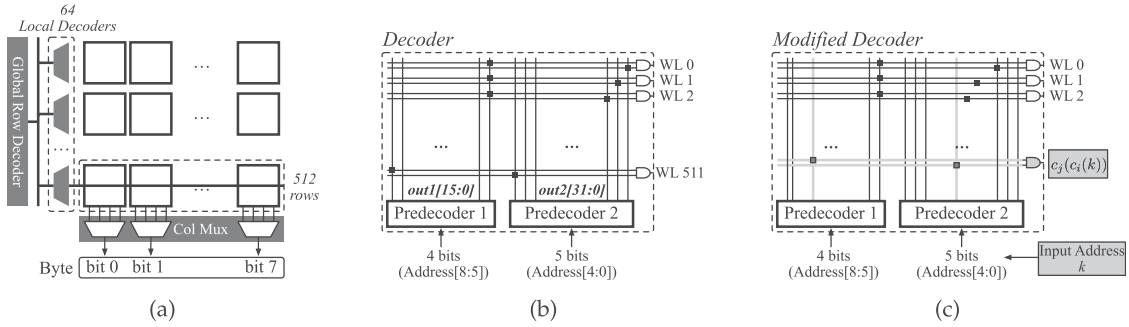

Fig. 9. The diagram of the (a) hierarchical row decoder, (b) a single decoder, (c) a modified decoder.

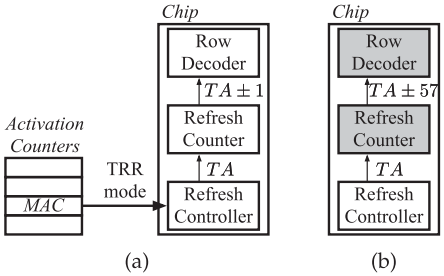

Fig. 10. The diagram of (a) the TRR/PARA refresh controller and counter, and (b) modified refresh controller and counter.

To minimize such overhead, an efficient implementation is presented next. Unlike the naive implementation, where the subarray receives input addresses directly, the proposed implementation directly selects the remapped address by altering the row decoder and the column mux. The conventional implementation of the row decoder is explained first and then the proposed implementation is described next. As depicted in Fig. 9a, the row decoder of a bank consists of two-level hierarchical decoders [21], [22]. The first level is the global decoder which selects a subarray, and the second level is local row decoder which selects a row inside the subarray. Typically, each subarray consists of 512 rows which requires a 9-to-512 decoder. This 9-to-512 local decoder is implemented as shown in Fig. 9b [22]. The input address is decomposed into two parts and passed to two predecoders. Predecoder 1 receives 4 MSBs of the address and generates the corresponding 16 outputs, and Predecoder 2 receives the remaining 5 bits of the address to generates 32 outputs. The final output of the local decoder is obtained by boolean AND operation between the one output from Predecoder 1 and the other from Predecoder 2. For example, row 0 is selected at the condition of $(out1[0]\ \&\ out2[0])$, row 1 at $(out1[0]\ \&\ out2[1])$, and row 511 at $(out1[15]\ \&\ out2[31])$.

The remapping matrix is applied to both the global and local row decoders. Fig. 9c shows the structure of the modified decoders. The important change noteworty in the figure is that the location of the contacts is rearranged according to the remapped address. These contacts are indicated by small boxes in the figure. For example, if address 2 is remapped to 0, then row 0 is obtained by $(out1[0]\ \&\ out2[2])$ as shown in Fig. 9c. In this manner, every predecoder line is connected to boolean AND gate that corresponds to the remapped address, and thereby resulting in input address k to be remapped to row $c_j(c_i(k)$.

Since the remapping matrix for each chip is fixed, the proposed remapping structure can be implemented by simply adapting the above method to each chip. Implementing the remapping matrix in such way does not change the design at the logic level. The predecoder blocks are never changed, and the 512 AND gates remain the same. Therefore, impacts on chip area and latency are negligible.

The detailed synthesis and place and route results are to be presented in Section 7.2.

5.3 Compatibility to Previous Solutions Against Rowhammer Attack

The proposed two-level remapping structure does not completely eliminate the rowhammer attacks. Therefore, it is practical to use the propose structure with previous solutions, such as targeted row refresh (TRR) [13] or PARA [1]. In this section, the compatibility of the proposed and TRR/PARA structure is explained.

TABLE 2
The Number of UEs and Reduction Rates

Module	Baseline	Proposed (Ratio)	TRR (Ratio)	Proposed +TRR
C	7385	228 (0.031)	57 (0.007)	0
D	12450	566 (0.045)	121 (0.010)	0
I	328	0 (0)	3 (0.009)	0
J	540	1 (0.002)	6 (0.011)	0

TRR is a scheme that counts the number of accesses per row and refreshes adjacent rows when a specific row is accessed intensively. As shown in Fig. 10a, TRR mode is triggered when the value of the activation counter exceeds the maximum activation count. When the TRR mode signal is delivered to the chip, the refresh controller triggers the refresh command and delivers the target address (TA) to the refresh counter. Finally, the refresh counter increments or decrements TA to deliver $(TA \pm 1)$ to the row decoder. PARA is a scheme that probabilistically refreshes adjacent rows for each activation. The refresh controller probabilistically triggers the refresh command, and the rest procedures are identical to those of TRR.

Under the conventional address mapping structure where the row decoder is not changed, the row decoder can select the adjacent rows without any problem. However, TRR/PARA do not work properly if our proposed remapping structure is applied. The row decoder is modified to select remapped address, resulting in a selection of completely different rows than the adjacent ones when $TA-1$, $TA+1$ addresses are requested.

To solve this problem, the refresh counter is modified in a way of applying inverse mapping of the remapping. Since the remapping applied to the row decoder is a linear mapping in LSBs, the inverse mapping is also linear in LSBs. For example, if the remapping of the row decoder is $c = (9 \cdot k + 5)\ mod\ 2^7$, its inverse mapping is computed as $k = (57 \cdot c + 99)\ mod\ 2^7$. In this case, the refresh counter is modified to increment/decrement 57 to TA, not 1. As shown in Fig. 10 adjacent rows are refreshed when TA, $TA \pm 57$ are sent to the row decoder. This modification in the refresh counter is very minor, and thus it does not degrade the performance.

Since each chip has a fixed remapping matrix, the inverse mapping can also be specified. If the refresh counter is changed according to the calculated inverse mapping, our proposed structure and TRR/PARA are compatible.

Experiments are conducted to evaluate the effectiveness of our proposed remapping when used with TRR. For the experiment, we implemented and simulated the operation of TRR in the DDR3 modules. When a row is accessed more than a certain threshold value, all operations are halted by 64ms (i.e., refresh interval length). By doing this operation, a victim row is refreshed automatically by DRAM controller. In this experimental environment where TRR operation is implemented, all eleven modules (module A to K of Table 1) are tested for fair comparison. Experimental results show that rowhammer errors still occurred under TRR environment on four modules (module C, D, I, and J) and Table 2

TABLE 3
Parameters Used for Three Remapping Matrices

Matrix Type	R_1					R_2				
n	7	7	7	7	7	7	7	7	7	7
a	2	4	8	8	8	2	4	8	8	8
a_0	1	1	1	3	5	1	1	1	1	1
b	0	0	0	0	0	5	5	3	5	7

shows the number of UEs and reduction rate of these modules. As presented in the third and fourth columns, the proposed remapping shows the reduction ratios of 0 to 0.045 whereas TRR shows the reduction ratios of 0.007 to 0.011. These results mean that the average effectiveness of the proposed remapping scheme is better than that of the prior TRR scheme. In addition, the fifth column of Table 2 shows that the UEs are completely removed when the two structures are used together. In conclusion, the proposed structure and TRR can be used together to effectively prevent rowhammer errors (i.e., the proposed remapping scheme has high compatibility and scalability).

6 Robustness Against Rowhammer Attack of the Two-Level Remapping

This section discusses the robustness of the proposed two-level remapping against rowhammer attack.

Suppose that an attacker selects row 1 as the victim row. Because of address remapping, the aggressor rows for each chip are different. For example, the aggressor rows of chip 0 are 0 and 2, and those of chip 1 are 13 or 15. Because the remapping matrix satisfies condition (4), 0/2 of chip 0 and 13/15 of chip 1 belong to different rows in the remapping matrix. Thanks to the two-level remapping which also remaps in the array-level, every array in a rank is attacked at different aggressor rows. Therefore, a single attack affects only a single array, and consequently, can create an error in only a single bit in a word.

Because a single bit, not a whole word, is attacked by a single attack, the probability of UE occurrence within a refresh interval (RI) is significantly reduced. A typical RI is 64ms whereas the minimum attack time is 8.2ms as reported in [1]. This means that the rowhammer attack cannot be performed more than 8 times within an RI. If a BER is set to 10^{-4} (typical value obtained by experiments), the probability of UE occurrence under conventional structure is derived as $2 \cdot 10^{-5} (= 1 - (1 - BER)^{64} - 64 \cdot BER \cdot (1 - BER)^{63})$. On the other hand, when 8 attacks are performed and thereby 8 bits in a word are attacked, the probability is $2.8 \cdot 10^{-7} (= 1 - (1 - BER)^{8} - 8 \cdot BER \cdot (1 - BER)^{7})$. This value is reduced by 71 times compared to the previous one.

In summary, the proposed remapping is more robust than the conventional mapping against rowhammer attack because the number of attacked bits in a word is reduced. As a result, the probability of UE occurrence within an RI is reduced by 71 times.

7 Experimental Results

7.1 Experimental Environment

This section presents the experimental results for the evaluation of the effectiveness of the proposed two-level address remapping. Rowhammer errors are measured with eleven commercial DDR3 DIMMs, each of which includes eight chips with x8 bit width and 2 or 4 GB capacity. These DIMMs are from various vendors and have manufacturing dates ranging from 11-07 to 16-25 (in form of yy-ww). These DIMMs are installed in a host PC with x86 CPUs and the rowhammer attack programs based on [8] are executed to inject errors for all units in a given memory space. The program in [8] only attacks addresses in the same bank without information about DRAM address mapping. In this case, the double-sided rowhammer attack cannot be performed because it is not possible to know which addresses are consecutive in the DRAM. Therefore, techniques such as [18], [23] are first used to find out the DRAM address mapping. Then, by using the mapping information, the double-sided, unit-based rowhammer attacks detailed in [7] are executed to increase the error rate. For each victim row, the upper and lower rows are activated one million times within the RI (64 ms).

The location of the rowhammer errors caused by the attack codes are recorded for each module. Since the rowhammer attacks are performed on commercial packaged DRAM modules, it is impossible to actually implement the proposed remapping structure. Instead, the remapping processes are implemented by moving the locations of recorded rowhammer errors according to the remapping matrix. To show the effect of the proposed remapping structure, two types of remapping matrices are experimentally tested, as listed in Table 3. The first type, denoted as R_1, is a matrix satisfying only condition (4), and the second type, denoted as R_2, is a matrix satisfying both (4) and (10). The specific parameters of the remapping matrices used for each type are shown in Table 3. For remapping matrix of the global row decoder, a fixed R_2 type matrix is used (n of 5, a of 4, a_0 of 1, and b of 5).

All remapping matrices in Table 3 can be applied to various DRAM organizations. The effects of the number of rows, subarrays, and data bus width to the remapping matrix are explained. The typical number of rows ranges from 16,384 to 65,536. Considering the number of rows per subarray is generally fixed to 512, the number of subarrays ranges from 32 to 128. The value of n for all remapping matrices in Table 3 is 7. Because 2^n value (128) is less than the number of rows per subarray (512), all matrices can be used regardless of the number of rows and subarrays. The data bus width is typically ranges from 4 to 16. This means that the columns per column mux ranges from 4096 (bus width of 4) to 1024 (bus width of 16). This value is a multiple of 2^n (128), and therefore the remapping matrices can also be used.

It should be noted that every row and column address is accessed in the experimental setup carried out to obtain the results in this paper. This means that the entire memory space is attacked, and errors caused by attacks are accumulated. The results given in Section 7.2 are evaluated under this experimental setup.

7.2 Results

The relative frequencies of the errors are given first. Figs. 11, 12 and 13 list the relative frequencies with respect to subarray, column and row addresses, respectively. First, Fig. 11 shows that the relative frequencies of the remapped structure are closer to 1 than the original graph. For Figs. 12 and 13, the original, the remapping with an R_1 type matrix,

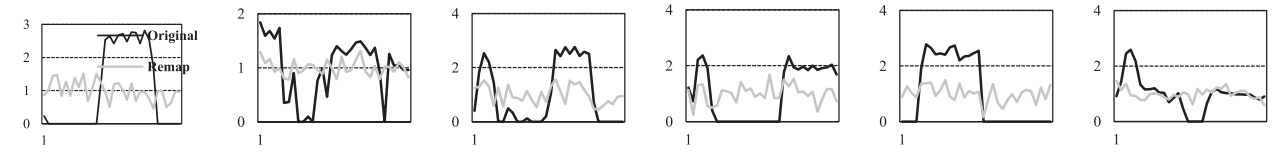

Fig. 11. The relative frequency (y-axis) of rowhammer errors per word, with respect to subarray index (x-axis). The black and grey lines represent the original mapping and R_2 remapping, respectively.

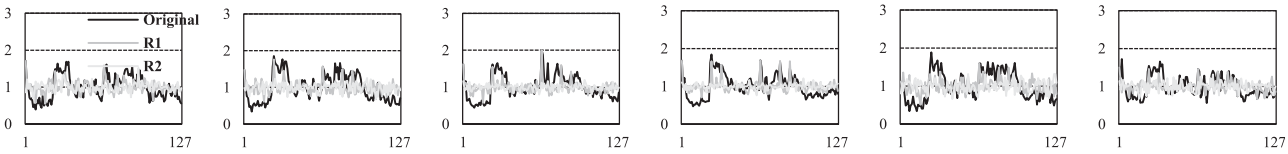

Fig. 12. The relative frequency (y-axis) of rowhammer errors per word, with respect to column address (x-axis). The black, grey, and light grey lines represent the original mapping, R_1 remapping, and R_2 remapping, respectively.

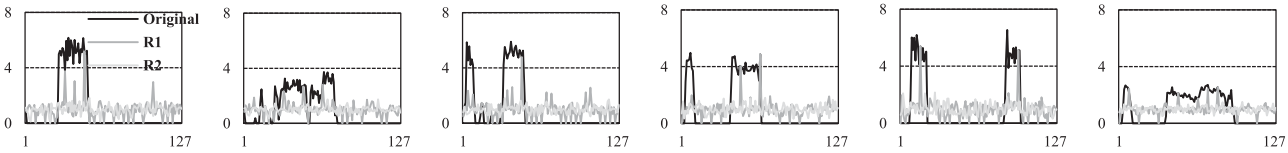

Fig. 13. The relative frequency (y-axis) of rowhammer errors per word, with respect to row address (x-axis). The black, grey, and light grey lines represent the original mapping, R_1 remapping, and R_2 remapping, respectively.

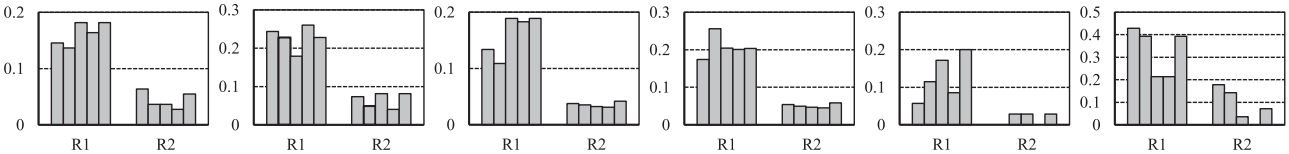

Fig. 14. The reduction ratio of UEs under SECDED, for R_1 and R_2 matrices.

and the remapping with an R_2 type matrix are given. The graphs show that the relative frequencies are close to 1 when the remapping is applied in all tested modules. Moreover, the remapping with R_2 shows more uniform results than that with R_1. Therefore, this result shows that rowhammer errors are efficiently distributed when the remapping is applied with the matrix satisfying both conditions (4) and (10), as analyzed in Section 4.

The next experiments aim to show the reduction of UEs by the proposed two-level address remapping under the error correction by an ECC. To this end, rowhammer errors are assumed to be recovered by two ECCs: SECDED and SSCDSD. SECDED ECC allows a single error in a word to be correctable, which implies that multiple errors in a word become UEs. On the other hand, SSCDSD allows an occurrence of error(s) in a single symbol of 8-bit size to be correctable so that errors in multiple symbols are regarded as UEs. More specifically, a 64-bit word size is assumed for SECDED , and a 64-bit word size with a 8-bit symbol size is assumed for SSCDSD. The ECC schemes assumed above are the basic schemes used in real server-class processors. Low-end servers use SECDED because of its simple implementation while high-end servers use SSCDSD because of its strong error protection capability [24]. The effectiveness of the proposed address remapping is evaluated with the number of UEs.

Fig. 14 shows the UE reduction rates for each matrix in Table 3. Five matrices per remapping type are plotted in order. The graphs show that the reduction rate of the R_2 type remapping is lower than that of R_1 type. When comparing the average value of the five matrices of type, R_2 achieves 3.50-7.33 times better reduction rates compared to R_1. In particular, in modules C and D which show the highest BER, the reduction rate of R_2 is more than 4 times better than that of R1. From these results, the effectiveness of a remapping matrix satisfying both conditions (4) and (10) is verified again.

Figs. 15a and 15b show the reduction rate of UEs under SECDED and SSCDSD ECC, respectively. For each module, the reduction rate is expressed in five steps. The first result denoted by 'original' represents the number of UEs with the original address mapping. This number is normalized to become 1 and used as the reference for comparison with the results of other mappings. The results denoted 'R1 single-level' indicate the chip-level remapping by R_1 matrix, whereas 'R1 two-level' represents the results with the R1

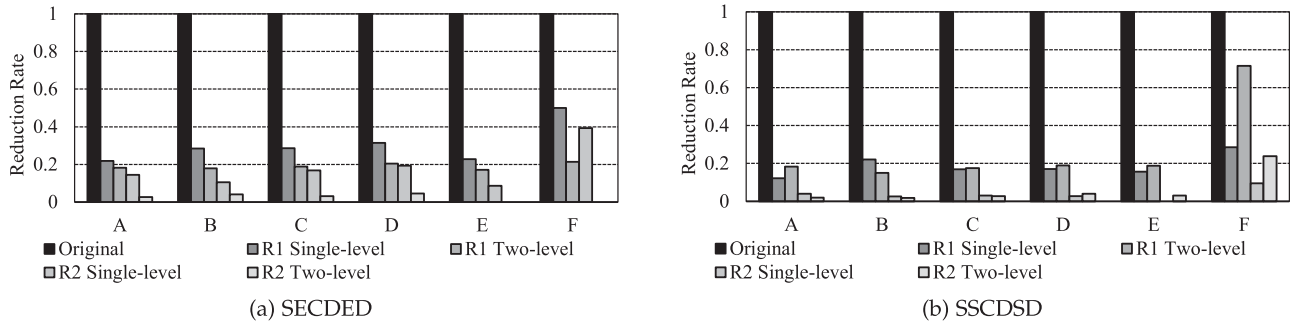

Fig. 15. The reduction ratio of UE under (a) SECDED ECC, (b) SSCDSD ECC.

matrix when both chip-level and array-level mapping are applied. The same experiments are conducted with the R_2 remapping matrix, and the results are also presented in Figs. 15a and 15b. The third row and ninth row of Table 3 are used as representative of R_1 and R_2 matrices, respectively.

The reduction rates between the single-level and the two-level remapping are compared. In Fig. 15a which shows the reduction rates under SECDED ECC, the two-level remapping shows better results in most cases. Conversely, in Fig. 15b which shows the reduction rates under SSCDSD ECC, the single-level remapping shows better results. This is because multiple errors occurring in one symbol are distributed over several symbols through array-level remapping, which results in increasing the number of symbols where errors have occurred. Therefore, in case of SSCDSD ECC, it is better to use the result of single-level remapping as the final remapping result.

Additional experiments with DDR4 modules are also carried out but the results are not reported in the paper because of the following reason. Most DDR4 modules have built-in rowhammer protection schemes such as TRR [25]. Because of the protection scheme, the occurrence of UEs is not sufficiently large when compared to that with DDR3 modules. BER of DDR4 modules is of 10^{-7} to 10^{-9}, which is hundreds times smaller than that of DDR3. The results obtained with an insufficient number of UEs may not be statistically reliable, and also they are not sufficient to verify the effectiveness of the proposed remapping structure. Therefore, only the results with the DDR3 modules are presented in this paper. Nonetheless, the proposed remapping can be used together with existing solutions such as TRR/PARA for the next generation DRAMs because the advance in DRAM technology further increases the probability of rowhammer errors so that TRR/PARA may not efficiently cover all rowhammer attacks.

7.3 Hardware Cost Analysis

This subsection discusses the hardware cost for the implementation of two-level remapping which requires the 8 global decoders and 64 local decoders different from the original without remapping. The areas and latencies of these modified decoders may be different from each other and the modification may result in an increase of the chip area and latency compared to the original decoder without address remapping. For the evaluation of area and latency, all possible decoders are designed, and the area and latency are estimated after synthesis, place and route (PNR) with a 65nm standard cell library using Synopsys Design Compiler and IC Compiler. PNR results of 8 global decoders show that the area ranges from 101.52 to 102.60um^2, and the critical path delay ranges from 0.4233 to 0.4308ns. For 64 local decoders, PNR results show that the area ranges from 1891.08 to 1910.88um^2, and the critical path delay ranges from 0.6058 to 0.6885ns.

Consider the worst case scenario such that the original decoder without remapping is the smallest (101.52 and 1891.08um^2) and the fastest (0.4233 and 0.6058ns) whereas the address remapping requires a decoder with the largest area (102.60 and 1910.88um^2) and latency (0.4308 and 0.6885ns). Then the increase of the global decoder area is at most $1.08um^2$ which is the difference between the largest (102.60um^2) and the smallest (101.52um^2). Because each chip has 16 global decoders, the area overhead is at most 17.28um^2 ($1.08um^2\times$ 16). For local decoders, the increase in area is at most $19.8um^2$ which is the difference between the largest decoder (1910.88um^2) and the smallest decoder (1891.08um^2). Because each chip has 1024 local decoders and 256 local muxes (64 row decoders and 16 column muxes per bank, and 16 banks per rank), the increase of the area by all 1280 decoders and muxes is at most 0.025mm^2 ($19.8um^2\times$ 1280) is increased.

For comparison, the overall rank chip area is obtained from CACTI simulation [26] under 65nm environment to model DDR3 module and the total area is obtained as $11.107mm^2$. This means that the overall increased area ($17.28um^2 + 0.025mm^2$) corresponds to only a small fraction (0.23 percent) of the original chip area.

The critical path delay is increased by 0.0075ns (0.4308-0.4233ns) due to global decoder, and 0.0827ns (0.6885-0.6058ns) due to local decoder. CACTI simulation shows that the overall DRAM delay is 9.78ns. Therefore, the critical path delay is increased by 0.93 percent, which is negligible.

7.4 Summary

Experimental results show that an R_2 type matrix is most effective in the reduction of UEs by address remapping. In addition, two-level remapping in SECDED and single-level remapping in SSCDSD ECC must be used, respectively. The results under these configurations are summarized in Table 4. The reduction rate ranges from 0 to 0.071. In particular, in modules C and D which suffer relatively large UEs, the reduction rate is between 0.023 and 0.045. This means that the proposed remapping effectively reduces UEs even in the vulnerable modules.

When the proposed remapping is applied to a commercial DRAM module, the reduction rate is expected to be even better than the values given in Table 4. This is due to the limitation of the experimental method. As mentioned earlier in this section, the rowhammer errors are distributed

TABLE 4
The Number of UEs and Reduction Rates with Respect to Modules

Memory Module	SECDED				SSCDSD			
	Before Remapping	Chip-level Remapping	Array-level Remapping	Reduction Rate	Before Remapping	Chip-level Remapping	Array-level Remapping	Reduction Rate
A	110	16	3	**0.027**	98	4	2	**0.036**
B	123	13	5	**0.040**	113	3	2	**0.024**
C	7385	1244	228	**0.031**	6494	199	175	**0.027**
D	12450	2416	566	**0.045**	10655	287	432	**0.023**
E	35	3	0	**0**	32	0	1	**0**
F	28	11	0	**0**	21	2	5	**0.071**
G	12	4	0	**0**	8	0	0	**0**
H	13	1	0	**0**	12	0	0	**0**
I	328	45	0	**0**	290	0	0	**0**
J	540	111	1	**0.002**	438	0	0	**0**
K	25	8	0	**0**	21	1	2	**0.048**

only *after* the rowhammer attacks have taken place. However, when the address remapping is applied to a commercial DRAM module, the remapping weakens the double-sided rowhammer attack because adjacent rows of the aggressor row differ by chips and subarrays. Therefore, the number of UEs is expected to be lower than the those listed in Table 4, resulting in even better reduction rates.

Hardware cost analysis shows that the proposed two-level remapping structure only incurs additional 0.23 percent area and 0.85 percent latency compared to conventional DRAM module, which is a negligible increase.

8 Related Work

This section presents an overview of existing techniques for rowhammer attack and defense. Furthermore, the previous work about DRAM address mapping is introduced.

Rowhammer Attack. Since the discovery of rowhammer errors in DRAM devices in [1], research efforts have been made to increase the probability of rowhammer errors. Double-sided rowhammer attack [8] increases the error rate by alternately activating rows above and below the victim (i.e., target) row. Unit-based rowhammer attack [7] is another technique to increase the error rate by attacking rows in a cache line size rather than page-wise fashion. Another technique in [27], one-location hammering, attacks only a single row and therefore, it can bypass many rowhammer protection mechanisms. Repeatedly using these techniques to specific regions of a DRAM device may result in intensive occurrence of rowhammer errors in the regions. In [5], [6], [7], [9], rowhammer attacks make it possible to generate errors in a page table, which holds critical information to gain access privileges to a system. Moreover, techniques in [27], [28] can even penetrate Intel SGX [29], which is a software-based security enhancement solution developed by Intel.

Rowhammer Defense. Instead of relying on a conventional ECC, PARA, [1], a probabilistic row refresh technique, is proposed. A pseudo-random number generator is used to probabilistically refresh a row whenever it is activated. This technique is simple to implement but it requires a large number of refresh operations, most of which may not be necessary. In [12], a probabilistic technique to keep track of possible victim rows is proposed. This technique, called PRoHIT, is efficient in the utilization of previous access history. Another technique aims to avoid rowhammer errors by refreshing the rows that are accessed frequently. To this end, the numbers of activations per row is counted using a per-row counter and the adjacent rows are refreshed whenever the counter reaches a threshold [13]. This kind of approach is supported by chipset manufacturers [30], and thus it is most general defense method in practical use. The advantage of this technique lies in the reduction of unnecessary refresh operations although the per-row counters may increase the hardware cost. To reduce the hardware cost, a tree-structured counter implementation is proposed in [14]. It dynamically assigns counters to the rows which are accessed frequently, thereby reducing the number of counters. Software-based rowhammer protections are also studied. ANVIL [31] protects rowhammer attacks by detecting locality of DRAM access using existing hardware performance counters. While this technique focuses on reducing the errors themselves, [32] prevents errors from occurring in security sensitive areas. It changes the physical memory allocator to a physically isolated kernel and user space.

DRAM Address Remapping. Address remapping, or scrambling, is a widely used technique which remaps row and column addresses within a DRAM subarray [15], [16], [17]. It is previously used for two reasons. First, it improves the hardware efficiency of the row decoder and column mux [15] in a DRAM. Second, it hides DRAM address space from users, thereby strengthening its tolerance against security attacks including rowhammer attacks. Recent work uses different remapped address for each chip to further enhance the spatial locality of DRAM access [33]. The proposed work is similar to [33] in that both uses different remapping structure per chip. However, the goal of the remapping is different, thereby the method of generating remapped address is also different.

Relationship with the Proposed Work. Since the proposed work is about protecting a system against rowhammer errors, it is compared to previous solutions for rowhammer defense. The previous solutions about rowhammer defense mostly aim to reduce the error rate itself. On the other hand, the proposed work does not affect (reduce) the error rate itself. Instead, it attempts to distribute errors under the given error rate to make them correctable with an ECC. This means that the proposed work can be used together with the previous

solutions. For example, a previous rowhammer solution is first applied to reduce the error rate, and then the proposed remapping is applied under the reduced error rate to further reduce UEs. Considering the experimental results that the proposed remapping completely eliminates the UEs when a BER is low, the combination of two solutions can effectively defend the system against rowhammer attack.

9 Conclusion

This paper presents a new DRAM address remapping scheme that distributes rowhammer errors over multiple rows and columns. By exploiting the distribution of the rowhammer errors, the effective remapping matrix is derived mathematically. Using the derived matrix, the overall DRAM address is remapped via a two-step process. Experimental results show that the derived matrix effectively distributes the rowhammer errors, thereby reducing the occurrence of UEs by up to 99 percent. The proposed remapping significantly reduces the probability of UE occurrence by 71 times.

Another advantage of the proposed structure lies in its broad applicability. For example, it can be used together with previous rowhammer solutions that reduce the rowhammer error rate. Moreover, the proposed structure does not require a large hardware cost. This means that the proposed structure can be used as a complement, rather than a substitute solution.

It might be interesting to adopt dynamic address remapping for the future work. The dynamic remapping can address the limitation of this work, that it cannot eliminate the UEs already occurred. Together with ECC protection, the UEs can be eliminated by dynamically changing the remapped address.

Acknowledgments

This research was supported by the MOTIE (Ministry of Trade, Industry & Energy (10080613, DRAM/PRAM heterogeneous memory architecture and controller IC design technology research and development) and the Research fund for a new professor by the SeoulTech (Seoul National University of Science and Technology).

References

[1] 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 *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit.*, 2014, pp. 361–372.

[2] V. Sridharan, N. Debardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems: The good, the bad and the ugly," in *Proc. 20th Int. Conf. Archit. Support Program. Languages Operating Syst.*, 2015, pp. 297–310.

[3] O. Mutlu, "The rowhammer problem and other issues we may face as memory becomes denser," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2017, pp. 1116–1121.

[4] O. Mutlu, "Memory scaling: A systems architecture perspective," in *Proc. 5th IEEE Int. Memory Workshop*, 2013, pp. 21–25.

[5] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip feng shui: Hammering a needle in the software stack," in *Proc. USENIX Security Symp.*, 2016, pp. 1–18.

[6] D. Gruss, C. Maurice, and S. Mangard, "Rowhammer. js: A remote software-induced fault attack in javascript," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Berlin, Germany: Springer, 2016, pp. 300–321.

[7] Y. Xiao, X. Zhang, Y. Zhang, and R. Teodorescu, "One bit flips, one cloud flops: Cross-VM row hammer attacks and privilege escalation," in *Proc. USENIX Security Symp.*, 2016, pp. 19–35.

[8] R. Qiao and M. Seaborn, "A new approach for rowhammer attacks," in *Proc. IEEE Int. Symp. Hardware Oriented Security Trust*, 2016, pp. 161–166.

[9] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 1675–1689.

[10] S. Khan, D. Lee, Y. Kim, A. R. Alameldeen, C. Wilkerson, and O. Mutlu, "The efficacy of error mitigation techniques for DRAM retention failures: A comparative experimental study," in *Proc. ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, 2014, pp. 519–532.

[11] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ECC memory against rowhammer attacks," presented at 40th IEEE Symposium on Security and Privacy, San Francisco, U.S., 2019.

[12] M. Son, H. Park, J. Ahn, and S. Yoo, "Making DRAM stronger against row hammering," in *Proc. 54th Annu. Des. Autom. Conf.*, 2017, Art. no. 55.

[13] D.-H. Kim, P. J. Nair, and M. K. Qureshi, "Architectural support for mitigating row hammering in DRAM memories," *IEEE Comput. Archit. Lett.*, vol. 14, no. 1, pp. 9–12, Jan.-Jun. 2015.

[14] S. M. Seyedzadeh, A. K. Jones, and R. Melhem, "Counter-based tree structure for row hammering mitigation in DRAM," *IEEE Comput. Archit. Lett.*, vol. 16, no. 1, pp. 18–21, Jan.-Jun. 2017.

[15] A. J. Van De Goor and I. Schanstra, "Address and data scrambling: Causes and impact on memory tests," in *Proc. 1st IEEE Int. Workshop Electron. Des. Test Appl.*, 2002, pp. 128–136.

[16] S. Khan, D. Lee, and O. Mutlu, "Parbor: An efficient system-level technique to detect data-dependent failures in DRAM," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2016, pp. 239–250.

[17] B. Akin, F. Franchetti, and J. C. Hoe, "Data reorganization in memory using 3d-stacked DRAM," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit.*, 2015, pp. 131–143.

[18] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "Drama: Exploiting DRAM addressing for cross-cpu attacks," in *Proc. USENIX Security Symp.*, 2016, pp. 565–581.

[19] M. K. Qureshi, D.-H. Kim, S. Khan, P. J. Nair, and O. Mutlu, "Avatar: A variable-retention-time (VRT) aware refresh for DRAM systems," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2015, pp. 427–437.

[20] M. Patel, J. S. Kim, and O. Mutlu, "The reach profiler (REAPER): Enabling the mitigation of DRAM retention failures via profiling at aggressive conditions," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 255–268.

[21] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (SALP) in DRAM," in *Proc. 39th Annu. Int. Symp. Comput. Archit.*, 2012, pp. 368–379.

[22] B. Jacob, S. Ng, and D. Wang, *Memory Systems: Cache, DRAM, Disk*. Burlington, MA, USA: Morgan Kaufmann, 2010.

[23] A. Barenghi, L. Breveglieri, N. Izzo, and G. Pelosi, "Software-only reverse engineering of physical DRAM mappings for rowhammer attacks," in *Proc. IEEE 3rd Int. Verif. Security Workshop*, 2018, pp. 19–24.

[24] S. Mittal and M. S. Inukonda, "A survey of techniques for improving error-resilience of DRAM," *J. Syst. Archit.*, vol. 91, pp. 11–40, 2018.

[25] K. Kasamsetty, "DRAM scaling challenges and solutions in lpddr4 context," presented at MemCon, Santa Clara, U.S., 2014.

[26] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP Laboratories*, Tech. Rep. HPL-2009 -85, pp. 22–31, 2009.

[27] D. Gruss, M. Lipp, M. Schwarz, D. Genkin, J. Juffinger, S. O'Connell, W. Schoechl, and Y. Yarom, "Another flip in the wall of rowhammer defenses," in *Proc. IEEE Symp. Security Privacy*, 2018, pp. 245–261.

[28] Y. Jang, J. Lee, S. Lee, and T. Kim, "SGX-bomb: Locking down the processor via rowhammer attack," in *Proc. 2nd Workshop Syst. Softw. Trusted Execution*, 2017, Art. no. 5.

[29] V. CostanandS. Devadas,"Intel SGX explained," *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[30] M. Kaczmarski, "Thoughts on Intel Xeon e5–2600 v2 product family performance optimisation–component selection guidelines," Aug., 2014. [Online]. Available: http://infobazy.gda.pl/2014/pliki/prezentacje/d2s2e4-Kaczmarski-Optymalna.pdf

[31] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "Anvil: Software-based protection against next-generation rowhammer attacks," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 743–755, 2016.
[32] F. Brasser, L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi, "Can't touch this: Practical and generic software-only defenses against rowhammer attacks," in *Proc. 26th USENIX Security Symp.*, 2017, pp. 117–130.
[33] V. Seshadri, T. Mullins, A. Boroumand, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Gather-scatter DRAM: In-DRAM address translation to improve the spatial locality of non-unit strided accesses," in *Proc. 48th ACM Int. Symp. Microarchitecture*, 2015, pp. 267–280.

Moonsoo Kim received the BS degree in electrical and computer engineering from Seoul National University, Seoul, Korea, in 2014. He is currently working toward the integrated MS and PhD degrees in electrical and computer engineering at Seoul National University, Seoul, Korea. His current research interests include SoC design of video/images, and low-power, reliable design of memory hierarchy.

Jungwoo Choi received BS degree in electrical and computer engineering from Seoul National University, Seoul, Korea, in 2016, and is currently working toward the integrated MS and PhD degrees in electrical and computer engineering at Seoul National University. His research interests include image processing, energy-efficient computer architecture, and memory systems.

Hyun Kim received the BS, MS and PhD degrees in electrical engineering and computer science from Seoul National University, Seoul, Korea, in 2009, 2011 and 2015, respectively. From 2015 to 2018, he was with the BK21 Creative Research Engineer Development for IT, Seoul National University, Seoul, Korea, as a research professor. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, Korea, where he is currently working as an assistant professor. His research interests include areas of algorithm, computer architecture, and SoC design for low-complexity multimedia applications.

Hyuk-Jae Lee received BS and MS degrees in electronics engineering from Seoul National University, Korea, in 1987 and 1989, respectively, and the PhD degree in electrical and computer engineering from Purdue University at West Lafayette, Indiana, in 1996. From 1998 to 2001, he worked at the Server and Workstation Chipset Division of Intel Corporation in Hillsboro, Oregon as a senior component design engineer. From 1996 to 1998, he was on the faculty of the Department of Computer Science of Louisiana Tech University at Ruston, Louisiana. In 2001, he joined the School of Electrical Engineering and Computer Science at Seoul National University, Korea, where he is currently working as a Professor. He is a founder of Mamurian Design, Inc., a fabless SoC design house for multimedia applications. His research interests are in the areas of computer architecture and SoC design for multimedia applications.