Received September 10, 2021, accepted September 22, 2021, date of publication September 24, 2021, date of current version October 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3115496

ECC-Aware Fast and Reliable Pattern Matching Redundancy Analysis for Highly Reliable Memory

DONGHYUN HAN[1], HAYOUNG LEE[1], SEUNGTAEK LEE[2], AND SUNGHO KANG[1], (Senior Member, IEEE)
[1]Department of Electrical and Electronic Engineering, Yonsei University, Seodaemun-gu, Seoul 03722, South Korea
[2]SK Hynix Inc., Icheon-si, Gyeonggi-do 17336, South Korea

Corresponding author: Sungho Kang (shkang@yonsei.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant by the Korean Government through the Ministry of Science and ICT (MSIT) under Grant 2019R1A2C3011079.

ABSTRACT Advanced capacity and density of memory have resulted in an increase in the probability of memory faults. The in-memory Error Correction Code (ECC), which solves this problem, is a widely used technology to improve the yield of highly integrated memory. However, the use of in-memory ECC causes problems that have not been considered in memory repair algorithms. Redundancy analysis is effective for repairing memory with redundant memory and in-memory ECC. In this paper, an ECC-aware fast and reliable pattern matching redundancy analysis algorithm for memory using both spare memory and in-memory ECC is proposed. This algorithm simplifies large-scale fault groups using in-memory ECC and includes an early termination method that can determine whether a memory that cannot be repaired with line spares can be repaired considering in-memory ECC. Experimental results show that the proposed pattern matching redundancy analysis algorithm achieves a similar yield but 14.6% less RA time and 8.6 times higher reliability compared to the existing redundancy analysis algorithms.

INDEX TERMS In-memory ECC, memory repair, redundancy analysis, yield, reliability.

I. INTRODUCTION

Advances in memory production and design technology have led to steady improvements in memory cell density, performance, power consumption efficiency, and cost. However, with a decrease in the size of memory cells and increase in the degree of integration, more faults have occurred in the memory production process than before, especially for Dynamic Random-Access Memory (DRAM). There are two types of fault patterns: (1) a case with multiple faults that share a row or column address and (2) a single cell fault that does not share an address with any other fault [1]. Moreover, owing to the improvement in memory design and production technology, soft errors are more frequent than the errors in the case where memory cells are large and placed far apart from each other. These soft errors are present in cosmic rays [2] and random telegraph noise [3], substantially threatening the system reliability. Thus, improving the yield and reliability has been a major challenge for the advancement of DRAM technology.

Redundancy Analysis (RA) [4]–[10] and Error Correction Codes (ECCs) have been studied as technologies for improving memory yield and reliability, respectively. RA uses redundant memories located in the memory layer to repair faulty cells found through memory tests in each DRAM array [9]–[11]. In general, redundant memory is divided into row and column line spares, and the row or column in which the faulty cell is located is routed to this spare resource for repair. In addition, redundant memory of various sizes has been proposed to improve yield [12]. RA has been studied to achieve maximum yield with limited spare resources. Repair Most (RM) [4] and Branch and Bound (B&B) [5] are the cornerstones of RA. RM allocates spare resources on the line where most faults are located. It has a fast analysis speed with a simple analysis algorithm but cannot repair all faulty memories that can be repaired. In contrast, B&B repairs all repairable faulty memories because it builds trees to explore the number of possible repair solutions.

The associate editor coordinating the review of this manuscript and approving it for publication was Norbert Herencsar.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

However, it has a very slow analysis speed because it explores all cases. Fast RA (FAST) [6] is the fastest RA algorithm and uses a fault grouping method. The final solution is determined by combining the row and column line spares required to repair each group of faults; however, it does not achieve high repair rates. For this reason, Very Efficient RA (VERA) [7] has been proposed where the use of the fault group method is the same, but it is possible to repair all faulty memories. Although, its analysis speed is slower than that of FAST, it can find an optimal solution by deriving the final solution with a binary search tree. Fault Group Pattern Matching (FGPM) [8] performs best among algorithms that use the fault group method. By adopting effective early termination and a method that matches fault patterns to solutions, all repairable faults can be repaired at high speeds.

Unlike RA, which repairs faulty cells during production, ECC is used to protect the data in memory against soft errors that occur during operation. ECC is commonly used in rank level DRAM. The dual inline memory module with ECC applies single-error-correction and double-error-detection codes, along with an extra chip to store ECC check bits [13]. As the probability of soft errors increases, there is a need to develop effective fault-tolerant techniques for DRAM [14], [15]. Therefore, DRAM researchers and producers studied the placement of ECCs on DRAM dies, which are called in-memory ECCs [16]–[18]. Despite many difficulties in applying in-memory ECCs to DRAM [17], results have been reported for DRAM chips with in-memory ECCs [19]–[21].

An in-memory ECC has a limitation in that it can correct only one fault within one codeword. However, it can repair single faults more effectively than line spare memory under a spare row or column replacement policy. According to the line-based replacement policy, one line spare must be used to repair a single fault, whereas an in-memory ECC can repair any number of single faults. Because of these features, the in-memory ECC is a technology proposed to correct soft errors occurring during operation; nevertheless, its use for improving yield is also a major goal [13], [16], [19], [20], [22], [23].

However, the use of in-memory ECC has led to new discussions. When it is used to improve yield with redundant memory, very high yields can be achieved compared to the case where only redundant memory is used. Even if the entire processing capability of the in-memory ECC is not used to improve the yield, the yield is clearly improved. However, if a soft error occurs in a codeword in which a hard defect is located, which has not been repaired considering in-memory ECC, the memory fails. Therefore, to increase the yield without compromising the reliability of the memory, the ECC must be efficiently considered in RA. In memory design, repair technology considering in-memory ECC is emerging as an important research topic [7], [8], [13], [24]. For this reason, it is necessary to study RA that can improve the repair rate and cost efficiency of the memory by using in-memory ECC and redundant memory.

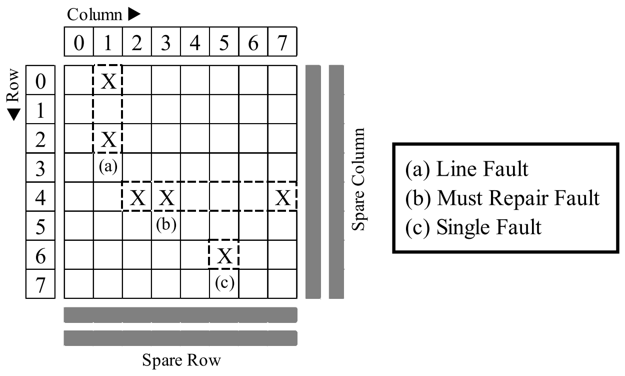

FIGURE 1. Example of a memory using a line replacement policy.

In this paper, an ECC-aware fast and reliable Pattern matching Redundancy Analysis algorithm (EPRA) for high reliability memory is proposed. This algorithm simplifies largescale fault groups, which incur considerable analysis time in RA, using the fault grouping method with in-memory ECC. In addition, a new early termination algorithm is proposed to quickly determine whether repair is possible, by considering in-memory ECC for memory that cannot be repaired with line spares.

II. BACKGROUND
A. REPAIR RATE
The repair rate is an indicator that directly affects the yield and is very important in the RA methodology. A low repair rate reduces the number of memories produced, leading to a lower yield. The repair rate indicates whether the RA can find a solution to be effective. It was introduced in [25] and can be defined as follows:

$$\text{Repair rate} = \frac{\text{No. of repaired chips}}{\text{No. of total tested chips}} \tag{1}$$

However, there may be cases where repair is impossible, regardless of the method used, owing to the limited spare resources of the memory. For an accurate analysis of RA performance, it is necessary to compare the number of repairs with the repairable memory when all solutions are considered. This is indicated by the normalized repair rate, introduced in [26], which can be defined as follows:

$$\text{Normalized repair rate} = \frac{\text{No. of repaired chips}}{\text{No. of repairable chips}} \tag{2}$$

In addition, a case in which the normalized repair rate is 100%, that is, all repairable memories are repaired, is referred to as the optimal repair rate.

B. SPARE ALLOCATION
In general, memory repair using the RA methodology follows a line replacement policy. There are two types of spares: 1) row line spare and 2) column line spare, as shown in Fig. 1. Line spares are used to replace faulty cells in memory. Note that the memory repair requires a replacement of the faulty cell with a line spare, regardless of the number of faults

located in one row or column line. In Fig. 1, (a) and (b) indicate multiple faults sharing the same row or column address. They are called line faults and are a common pattern of faults. The difference between (a) and (b) lies in the spare resources available to repair the faulty line. To repair (a), two row line spares or one column line spare must be used. The memory depicted in Fig. 1 has enough spare resources to repair (a) in either manner. In contrast, repairing (b) requires one row line spare or three column line spares. The memory shown in the example has two row line spares and two column line spares; thus, (b) can only be repaired with one row line spare. Thus, a case in which the number of faults sharing the column (row) address is greater than that of row (column) line spares and can be repaired only with column (row) line spares is called a must repair fault. In Fig. 1, fault (c) does not share a row or column address with other faults. This is called a single cell fault [1], [27], and it requires one line spare, which may be either a row or column. The line replacement policy of replacing several healthy cells to repair a few defective cells may appear inefficient, but it has the strong advantage of simple replacement control and the use of small control logic. However, in the real world, faults are much more complex than those depicted in Fig. 1; hence, an effective RA method is very important in the line replacement policy. RA has been studied to repair as many faults as possible with limited line spare resources. For this purpose, spare resources of a form other than line [12] or in-memory ECCs have been used to improve yield [13], [16], [22].

C. FAULT GROUPING AND PATTERNING

Fault grouping methods have been used in many RA techniques and for the evaluation of RA efficiency [6], [7], [28], [29], [30]. Each fault group comprises a single fault or multiple faults sharing the same row or column address. An important point in this approach is that the must repair line must be repaired prior to forming fault groups. Repair of the must repair line through the allocation of a line spare reduces the number of faults in each fault group, as well as unnecessary RA, by consuming line spares that must be used in advance.

The most significant feature of the fault grouping method is that faults in a group do not share the same row or column address as those in other groups. This allows each fault group to have a repair solution independent of other fault groups, so that solution searches can be performed independently. An example of the fault grouping method is shown in Fig. 2. As shown in Groups 1 and 2 in the figure, the faults included in each group do not share any address with the other groups. In contrast, the faults included in the same group are in the same row or column as those of one or more other faults. For this reason, the faults of each group can be analyzed independently.

Fault group patterning is a method proposed in FGPM [8] where after patterning each configured fault group, the information for each group is collected. The investigated information is used for effective early termination. Moreover, rather than analyzing the group's repair solution, it helps to match

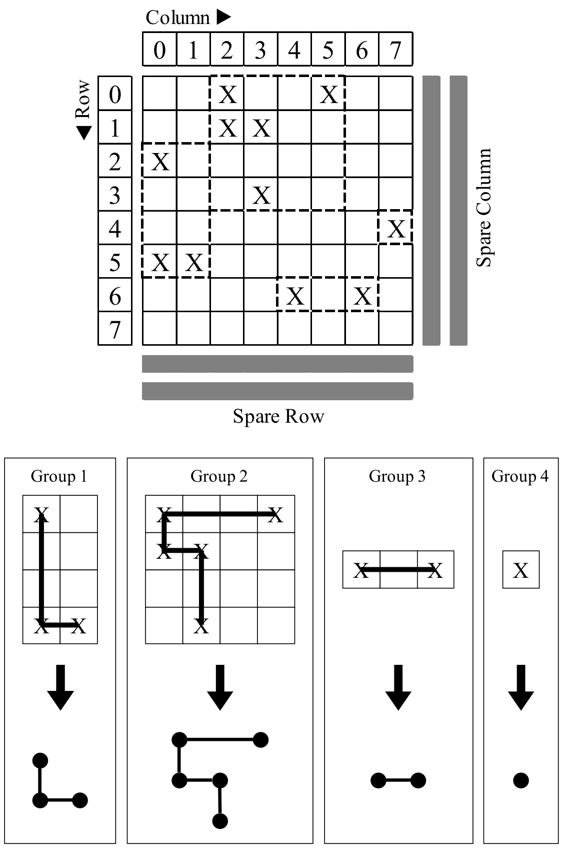

FIGURE 2. Example of a memory with the fault grouping method and pattern generation.

the solution immediately, thereby reducing the analysis time. Fault group patterns are formed by linking faults with the same address in each fault group. The investigated information is obtained from the configured fault group pattern and is listed as follows: 1) the number of faults (N_f); 2) the number of faulty rows (N_{fr}); 3) the number of faulty columns (N_{fc}); 4) the number of multi fault rows (N_{mr}); and 5) the number of multi fault columns (N_{mc}).

Let us consider Group 2 in Fig. 2 as an example, which comprises five faults ($N_f = 5$). Considering the row and column addresses of faults in the fault group, these faults are located in rows 0, 1, and 3, and in columns 2, 3, and 5. That is, because there are three rows and three columns having faults, both N_{fr} and N_{fc} have a value of 3. Considering addresses with multiple faults, there are two faults in each of the rows 0 and 1 and two in columns 2 and 3. That is, the number of rows and columns comprising multiple faults is two; thus, both N_{mr} and N_{mc} have a value of 2. As such, the pattern matching method divides a fault group by the number of faults and the number of faulty rows and columns. That is, how far apart the faults are within the pattern does not affect the patterning. Due to this, the pattern matching method can be applied regardless of the memory size. However, not all fault groups can use the pattern matching method. According to the study, the maximum number of faults in a group for which the pattern matching method can be used (N_p) is five. This is because the fault groups with more than N_p faults have different solution patterns even though they have the same investigated information.

III. PROPOSED BIRA APPROACH

A. OVERVIEW OF THE PROPOSED RA

The proposed ECC-aware RA exhibits the following three characteristics: 1) pattern matching-based solution search, 2) group simplification, and 3) ECC-aware termination. The pattern matching method is one of the most effective fault grouping RAs [8]. When the group pattern is classified based on certain information of the fault group, the repair solution of the group is derived from the list of solutions designated for each pattern. This substantially reduces the analysis time for RA by reducing the number of solutions to be considered. However, pattern matching is impossible for groups comprising more than five faults ($N_p = 5$). ECC-aware RA can reduce the number of faults that need to be repaired by line spares, by correcting faults with ECC. This feature considerably aids the pattern matching method, where the number of faults is important. In addition, ECC can repair only one fault within the same codeword, which implies that the positional relation between the faults should be considered in its use. Pattern matching is a specification of the positional relation between faults with some information and has similar characteristics to those of ECC. For these two reasons, the pattern matching technique can generate good synergy with the ECC-aware RA, which is why the proposed RA adopts the pattern matching method.

Group simplification is a process of changing a large fault group comprising more than N_p faults into a group capable of pattern matching by utilizing the features of ECC. By skip the faults that can be corrected by ECC, the number of faults that must be repaired by line spares is reduced.

Finally, ECC-aware termination is an effective early termination. For improved memory reliability and yield, ECC should be used while allocating line spares as efficiently as possible. ECC-aware termination determines whether repair is possible when considering ECC for a memory that cannot be repaired only with line spares. Using this termination, a quick judgment can be made because the calculation result is determined for each fault pattern, such as a solution list of pattern matching.

There are three parameters that affect memory productivity: the RA time, the yield, and the reliability. Short RA time and high yield allow memory vendors to produce more memory. However, if the produced memory breaks down quickly, even if it has high productivity, it is meaningless. The produced memories must be able to operate without failure for a long time. Since the proposed RA is an ATE-based RA, there is no circuit to be built into the memory, so the hardware overhead is not considered. The proposed idea uses ECC, a structure for reliability, to improve memory productivity. ECC is proposed to correct errors that occur during operation. But in the proposed idea, ECC is also used to correct permanent faults detected that detected during production. The

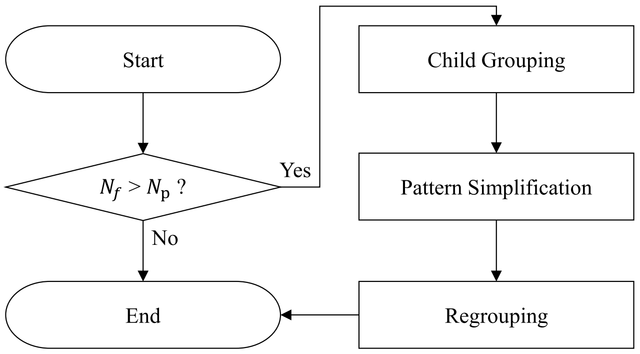

FIGURE 3. Flow chart of the group simplification.

proposed ECC-aware pattern matching algorithm shows fast RA speed and high yield. However, due to the careful use of ECC, high productivity gains are achieved with little loss of reliability. Also, the proposed idea is applicable not only to DRAM, but also to all types of memories using line spare with ECC to improve the reliability of the memory.

B. GROUP SIMPLIFICATION OF PREPROCESSING

In the preprocessing step, RAs using the pattern matching method classify faults into fault groups, generate fault group patterns, and then investigate the fault group information. Here, the investigated information is used to classify the fault group pattern, and the classified pattern has a solution within a specified solution list, without the need to search for a solution. This considerably reduces the RA time of the pattern matching method. However, the pattern matching method, which is one of the most effective methods among fault grouping RAs, also has limitations. The fault group pattern is classified based on the five types of information (N_f, N_{fr}, N_{fc}, N_{mr}, and N_{mc}) collected in the preprocessing stage. A fault group pattern comprising N_p or fewer faults has the same solution list if the fault group pattern information is the same. In contrast, in case of a fault group pattern comprising more than N_p faults, a different solution list is obtained even if they are classified in the same fault group pattern based on the five types of information. That is, the pattern matching method cannot be used for a group in which the number of faults exceeds N_p. Large fault groups comprising more than N_p faults increase the computational load while slowing down the analysis speed [8].

The proposed RA reduces the number of faults in a large fault group and enables pattern matching by finding and removing faults that can be corrected by in-memory ECC. This is called group simplification. In this stage, after collecting fault group information during preprocessing, 1) child grouping, 2) pattern simplification, and 3) regrouping is sequentially performed for large fault groups comprising more than N_p faults. The workflow of group simplification is illustrated in Fig. 3. Group simplification can also be applied to fault groups of 5 or fewer faults, but it can greatly increase the number of faults corrected by ECC. Since it can greatly reduce the reliability of the memory, the proposed RA applies group simplification only to a large fault group that cannot be pattern-matched.

Memory arrays comprise hundreds or thousands of rows (word lines) and columns (bit lines). Rows of in-memory ECC-applied memory comprise dozens of ECC codewords, each of which comprises data bits and check bits. Child grouping classifies the addresses of faults belonging to a large fault group by a codeword, and forms subgroups of the large group belonging to the same codeword. ECC can detect two faults within one codeword and correct one fault [13]. Child grouping is a preparation process to find faults that can be corrected by ECC among faults belonging to a large fault group. In the proposed RA, a large fault group comprising several faults (more than N_p) is called the mother group and the subgroups formed by dividing it into codeword units are called child groups. Faults that can be corrected by ECC in child groups are erased according to the fault skip rule in the next step, simplifying each child group. An example of child grouping is shown in Fig. 4(a), which indicates a part of the memory where a fault group is located. Let us suppose that the memory comprises 8-bit codewords, and the mother group, shown in Fig. 4, comprises eight faults spread across three codewords. When divided into codeword units, there are three subgroups, as shown in the figure, which are called child groups. The faults within a child group can share a row address with other child groups, but not the column address. Within the same codeword, the rules of the existing fault grouping are followed. If the same row or column address is shared, it is assigned to the same child group, and a fault without a shared address is a child group comprising one fault. For the above reasons, multiple child groups may exist within one codeword.

ECC can detect up to two faults in one codeword unit and can correct one fault. This means that in-memory ECC can fix one fault for each row in a child group. However, the fault skip rule of the proposed RA does not skip all faults that can be corrected by ECC. If all faults that can be corrected by in-memory ECC are handled by ECC, and if line spares are used only for faults that cannot be corrected, the repair rate will surely increase. However, reliability in the online operation of the memory may be adversely affected. This is explained in Section 3.5. For this reason, the proposed RA should be able to guarantee both yield and reliability of the memory by using ECC efficiently. For the most efficient use of ECC for RA, the proposed RA establishes a fault skip rule that skips faults that do not share the row address with other faults in the child group.

1) FAULT SKIP RULE

If a fault belonging to a group does not share a row address with other faults belonging to the group, it can be skipped from the group because it can be corrected by in-memory ECC.

There are two reasons for establishing the above rule. First, fault skip aims to simplify the fault group pattern. ECC should be used for faults that can change the pattern. When there are multiple faults in one codeword, skip one fault does not guarantee a pattern change. In contrast, when there is one fault in the codeword, the fault pattern always changes. Second, even if one fault is skipped for a row in which multiple faults exist, more than one line spare must be used anyway. If a line spare is to be used, it is more efficient to fix the line fault than a single fault. However, it is not necessary to use ECC in a situation where a line spare must be used anyway for a fault located in the row. The proposed RA simplifies the large fault group by skip the faults within the child group in the fault simplification step based on the fault skip rule. Fig. 4(b) shows an example of pattern simplification. Here, child group 1 comprises four faults, and faults located in rows 1 and 6 do not share the same row address in the group. These faults are cleared according to the fault skip rule, and the pattern of the child group appears to change through pattern simplification.

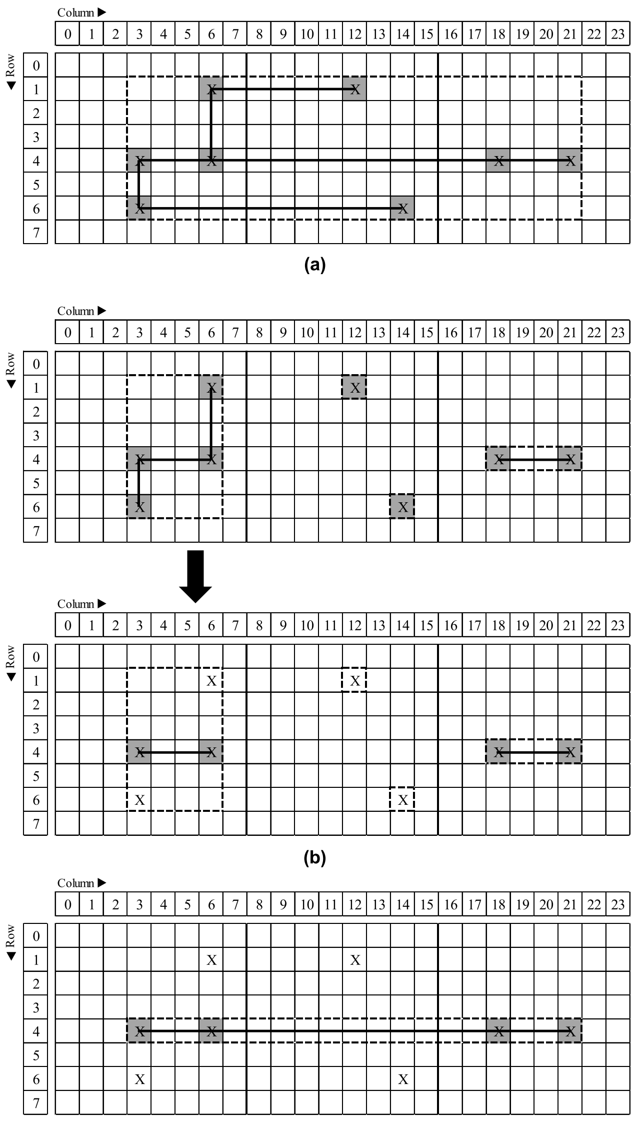

FIGURE 4. Example of child grouping in memory 8-bit codeword memory. (a) fault pattern of mother group. (b) child grouping and fault omission. (c) regrouping child groups after fault omission.

As the child group is codeword-based, if it is directly repaired using line spares, the inefficiency of using the spare occurs. For efficient spare allocation, it is necessary to group child groups back into line-based groups, such as the mother group. In the proposed RA, the process of recombining pattern-simplified child groups into line units is called regrouping. Pattern-simplified child group patterns are already grouped based on the column line. For grouping between child groups, only the row address needs to be checked; thus, it can be done faster than conventional grouping. In addition, as there are faults that have been eliminated, one mother group can be divided into several groups. The group simplification of the proposed RA ends with regrouping. Fig. 4(c) shows the regrouping of child groups after pattern simplification, where the group comprising eight faults has changed into that comprising four faults. As pattern matching is possible only for groups comprising five or fewer faults, pattern matching is not possible before for the group shown in the example but is possible after group simplification.

When the group simplification is completed, the total number of faults in the simplified fault group is checked. If the number of faults in the group is less than five, pattern matching is performed by collecting five types of fault groups, as presented above. If not, a solution search is conducted subsequently. Group simplification of the proposed RA handles faults that can be corrected by in-memory ECC within the fault groups, allowing pattern matching of as many fault groups as possible. This stage allows the solution list to be matched even for groups containing several faults, which can substantially reduce the analysis time.

C. ECC-AWARE TERMINATION

Early termination, which reduces unnecessary RA operations, is very important in improving RA performance. Among the termination methods, FGPM proposes effective early termination based on fault group pattern information [8]. This early termination calculates the minimum number of spares required for repairing each fault group (N_{mrs}) with the five types of information. If the sum of N_{mrs} of each fault group is greater than the number of available spares, the RA is terminated prematurely. However, this early termination does not consider faults that can be corrected by in-memory ECC. The premise of RA considering ECC is the efficient use of line spares. There should not be any inefficiencies, such as using ECC even though there are line spares left. Taking this into account, the proposed RA finally reviews the possibility of repair considering ECC only for the memory that failed to be repaired as a result of early termination. This is called ECC-aware termination. Here, while using the line spares as much as possible, the memory that could not be repaired previously is quickly determined through calculations to check if it can be repaired when considering in-memory ECC.

To simplify the fault group pattern using ECC, the fault skip rule defined above is followed. This rule changes the fault pattern as it eliminates single faults on a row line basis. Fault patterns are divided into fault group pattern information. In other words, this information is modified according to the fault skip rule. The fault group pattern information to which the fault skip rule is applied can be derived through a simple operation.

TABLE 1. Specifications of loads used in the experiment.

# of Faults	# of Patterns	$(N_f, N_{fr}, N_{fc}, N_{mr}, N_{mc})$
1	1	(1, 1, 1, 0, 0)
2	2	(2, 1, 2, 1, 0) (2, 2, 1, 0, 1)
3	6	(3, 1, 3, 1, 0) (3, 3, 1, 0, 1) (3, 2, 2, 1, 1)
4	27	(4, 1, 4, 1, 0) (4, 4, 1, 0, 1) (4, 2, 3, 1, 1) (4, 3, 2, 1, 1) (4, 2, 2, 2, 2) (4, 3, 2, 1, 2) (4, 2, 3, 2, 1)
5	148	(5, 1, 5, 1, 0) (5, 5, 1, 0, 1) (5, 2, 4, 1, 1) (5, 4, 2, 1, 1) (5, 2, 3, 2, 2) (5, 3, 2, 2, 2) (5, 2, 4, 2, 1) (5, 4, 2, 1, 2) (5, 3, 3, 2, 2) (5, 3, 3, 1, 2) (5, 3, 3, 2, 1) (5, 3, 3, 1, 1)

First, for the fault skip rule to be applied, there must be lines with a fault among the row lines. That is, it can be applied only to a fault group pattern where the value of N_{fr} is greater than that of N_{mr}.

$$N_{fr} > N_{mr} \tag{3}$$

When the fault skip rule is applied, the row lines where a fault is in the fault pattern are eliminated. In other words, the number of faulty rows of the modified fault group pattern (N'_{fr}) has the same value as N_{mr}.

$$N'_{fr} = N_{mr} \tag{4}$$

In addition, because the number of rows in which multiple faults are located is not affected by the fault skip rule, it does not change even in the modified fault group (N'_{mr}).

$$N'_{mr} = N_{mr} \tag{5}$$

Because each of the removed row lines contains one fault, the total number of faults in the fault group also changes. The number of faults of the modified fault group pattern (N'_f) is the number of faults in the original fault group (N_f) minus the number of removed row lines ($N_{fr} - N_{mr}$).

$$N'_f = N_f - (N_{fr} - N_{mr}) \tag{6}$$

When considering only the row line address, faults that are removed are single faults. However, as these faults belong to a fault group, if they are not in a group consisting of only one fault, they would have another fault that shares the column address. That is, the number of faulty columns of the modified fault group (N'_{fc}) remains the same as before. However, it is not known whether the number of multi fault rows has changed.

$$N'_{fc} = N_{fc} \tag{7}$$

However, it is not completely impossible to determine the number of multi fault columns of the modified fault

TABLE 2. Simplification result of fault group patterns.

N'_{mrs}	Simplified Pattern	$(N_f, N_{fr}, N_{fc}, N_{mr}, N_{mc})$
0	(0, 0, 0, 0, 0)	(2, 2, 1, 0, 1) (3, 3, 1, 0, 1) (4, 4, 1, 0, 1) (5, 5, 1, 0, 1)
1	(2, 1, 2, 1, 0)	(3, 2, 2, 1, 1) (4, 3, 2, 1, 1) (4, 3, 2, 1, 2) (5, 4, 2, 1, 1) (5, 4, 2, 1, 2)
	(3, 1, 3, 1, 0)	(4, 2, 3, 1, 1) (5, 3, 3, 1, 2) (5, 3, 3, 1, 1)
	(4, 1, 4, 1, 0)	(5, 2, 4, 1, 1)
2	(4, 2, 2, 2, 2)	(5, 3, 2, 2, 2)
	(4, 2, 3, 2, 1)	(5, 3, 3, 2, 2) (5, 3, 3, 2, 1)

group (N'_{mc}). The fault groups handled by the ECC-aware termination stage have patterns. In addition, the types of patterns are limited. The fault group patterns are listed in Table 1. The maximum number of faults in the fault group pattern is five; thus, the number of faults of the modified fault group pattern is up to four. In addition, N'_{fr} and N'_{mr} have the same value. The fault group patterns that satisfy the condition are (0, 0, 0, 0, 0), (2, 1, 2, 1, 0), (3, 1, 3, 1, 0), (4, 1, 4, 1, 0), (4, 2, 2, 2, 2), and (4, 2, 3, 2, 1). Because there is no fault group pattern with the same four types of information that can be calculated among these, all fault group patterns can find a modified fault group pattern under the fault skip rule applied with a simple operation.

The strongest advantage of the pattern matching method is that there is a list of solutions for each pattern. If the pattern can be specified, the RA speed can be substantially improved because the solution can be selected from a list of solutions. ECC-aware termination has the same advantage. It does not have a corresponding modified fault group pattern for each fault group pattern but has the minimum number of spares for the repair modified fault group (N'_{mrs}) required for early termination. The modified group patterns are specified in six types, and their N_{mrs} (same as N'_{mrs}) can be calculated. The original fault group patterns, simplified fault group patterns by the fault skip rule, and N'_{mrs} are summarized in Table 2. Because ECC-aware termination is applicable only to fault group patterns that satisfy Equation (3), not all fault group patterns are shown in Table 1.

Early termination requires N'_{mrs}, not the fault group pattern information. The purpose of ECC-aware termination is to show N'_{mrs}, that is, N_{mr} when ECC is considered for each fault group pattern. This is because if there is a memory that cannot be repaired as a result of early termination, it is possible to determine if it can be repaired considering the ECC by comparing the fault groups of the memory again with N'_{mrs}. Comparing the features of the original fault group patterns and N'_{mrs} in Table 2, it can be seen that N'_{mrs} can be easily derived through several features of the original fault group patterns.

First, the common point of the fault group patterns where $N'_{mrs} = 0$, is $N_{mr} = 0$. When N'_{mrs} is 1, it is difficult to extract the common points, but when N'_{mrs} is 2, the common points can be easily determined. When N_f is 5 and N_{mr} is 2, N'_{mrs}

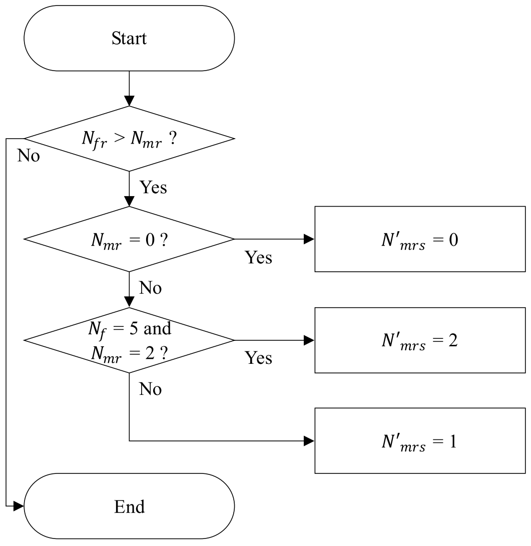

FIGURE 5. Flow chart of the ECC-aware termination.

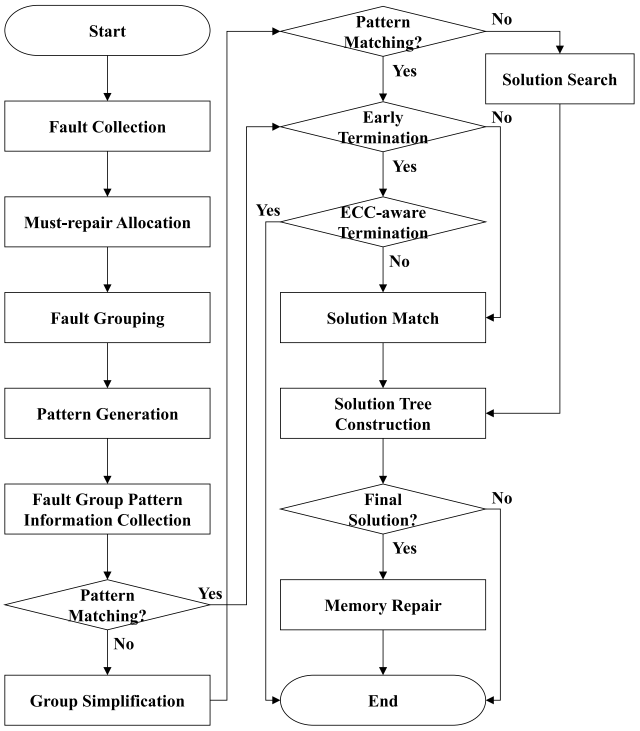

FIGURE 6. Flow chart of the proposed RA.

is always 2. The workflow of the ECC-aware termination is shown in Fig. 5. With the ECC-aware termination, fault groups can easily derive N'_{mrs} (N_{mrs} when simplified by the fault skip rule). This makes enables us to check whether an unrepairable memory can be repaired when in-memory ECC is efficiently considered.

D. WORKFLOW OF EPRA

Fig. 6 shows the overall workflow of the proposed RA. First, the locations of faults found as a result of the test during the fault collection stage are stored in the fault bitmap. Whenever a fault is detected in the test, the proposed RA determines whether the fault should be written to the bitmap. This is because if a fault was previously recorded in the bitmap, there would be no need to rewrite it. The faults thus recorded are divided into fault groups; the N_f of each group is analyzed while forming the fault groups. After all fault groups are determined, the pattern generation stage begins. In this stage, fault groups are converted into patterns suitable for each group and the fault group pattern information to be used in early termination and simplification is recorded. As mentioned earlier, the list of information collected for each group is: 1) N_{fr}, 2) N_{fc}, 3) N_{mr}, and 4) N_{mc}. N_f is not aggregated at this stage because it was previously analyzed while forming the fault groups. Because most fault groups comprise five or fewer faults [8], the pattern can be matched immediately; however, large fault groups need to undergo a group simplification process to determine if they can be converted into groups with five or fewer faults. Fault groups that cannot match the pattern are divided into child groups in units of a codeword, to identify faults that can be skipped. Faults that can be corrected by in-memory ECC are skipped. The simplified child groups are again grouped into a new group. As the number of faults in the group is significantly reduced, fault groups that were previously impossible to be matched can now be matched. If pattern matching is still not possible, all possible solutions are checked to determine an optimized repair solution for the fault group. After pattern matching, the fault groups for which information is recorded are subjected to early termination. If the conditions are met, the list of solutions specified for each pattern is recorded. However, if the conditions are not met, ECC-aware termination is performed. The ECC-aware termination proposed in this paper simplifies the fault group and proceeds with a new termination. If the conditions are met, it moves to the solution match stage, as for the previous fault groups; otherwise, the RA is ended because repair is impossible even if ECC is used. After the solution match and solution search produce a list of all solutions for each fault group, the solution tree construction step is performed to derive the final solution. Solutions for each fault group are selected, which then generate the stem of the solution tree. The proposed RA compares the available spare memories with the solution tree. If it finds a configurable stem with the available spare memories, the proposed RA determines it as the final solution and completes the RA. When an optimal solution is required, the proposed RA will explore the last trunk. If there are no configurable trunks of available spare memories, that memory is considered unrepairable and the RA ends.

The proposed RA has a common point with FGPM in that it quickly derives a repair solution by using a fault group pattern, but other than that, it has significant differences. First, the proposed RA has group simplification stage. The pattern matching stage in FGPM is performed for the fault groups consisting of 5 or fewer faults. However, in the proposed RA, the large faults group consisting of more than 5 faults are divided into codeword units (child grouping) and simplified. The simplified fault groups are merged into one group again (regrouping), and the pattern matching stage is performed. The second point the ECC-aware termination stage. By improving the early termination of FGPM, the proposed ECC-aware termination considers faults that can be corrected by ECC and quickly determines whether to repair. These two improve the RA time by speeding up the analysis of large fault groups, which had a significant impact on the RA time of FGPM. The difference due to these features is analyzed in the next section.

E. RELIABILITY ISSUES USING ECC WITH RA

Certain faults caused by hard defects are corrected by in-memory ECC. A single fault in a codeword is corrected by ECC to skip the faults located in a large group and convert it to a group capable of pattern matching, or to fix a memory that cannot be repaired with only line spares. Therefore, if a fault is added to the same codeword owing to a soft error, it is equivalent to a 2-bit error in the same codeword. Because ECC can only correct a single fault in a codeword, an ECC correction error occurs, and the memory would fail.

The Poisson distribution is used to model the soft error distribution occurring in each bit. The number of hard faults repaired by ECC is denoted as n_e, and the number of soft errors that occurred is denoted as n_s. If $n_e + n_s$ exceeds the number of soft errors that can be corrected by the ECC, this memory will be failed. Since the most widely used code of ECC can correct one error, the reliability of a memory over time is shown in Equation (8) [31], [32].

$$R_s(t) = \left[\left(e^{-\lambda t}\right)^{l-1}\right]^{N_k} \times \left[C_0^l\left(e^{-\lambda t}\right)^l + C_1^l\left(1-e^{-\lambda t}\right)\left(e^{-\lambda t}\right)^{l-1}\right]^{N_s} \quad (8)$$

In the above equation, λ denotes the soft error rate, and l is the size of words that are corrected at once by ECC. N_k is the number of correctable words in incorrect words and is the number of words that have already been repaired one hard error using the ECC. And N_s is the number of words with soft errors.

As can be seen from Equation (8), when using ECC-aware RA, the factor that has the greatest influence on reliability is the number of words that repaired hard defects using ECC. The proposed RA classifies and corrects only the minimum number of faults required to improve RA performance among faults that can be corrected by ECC. Through this, the number of faults corrected by ECC is minimized, thereby minimizing the decrease in reliability pointed out as a problem of ECC-aware RA. The analysis of reliability is covered in the next section.

IV. PERFORMANCE ANALYSIS OF THE PROPOSED RA

To evaluate the performance of the proposed RA, a simulation program is implemented in Python. Here, a 1024 X 1024 memory is used because, although the memory block

TABLE 3. Comparison of preprocessing time between FGPM and EPRA.

# of faults	Preprocessing time difference			RA time difference		
	Cs = Rs = 3	Cs = Rs = 4	Cs = Rs = 5	Cs = Rs = 3	Cs = Rs = 4	Cs = Rs = 5
8	4.82%	1.54%	9.13%	4.89%	-0.74%	5.79%
9	7.90%	-1.14%	3.54%	1.95%	-1.63%	0.06%
10	-3.09%	-0.08%	4.80%	-9.94%	-0.50%	-0.11%
11	1.21%	-0.44%	2.83%	-14.40%	-0.54%	-2.12%
12	5.62%	1.03%	-0.97%	-16.71%	-4.90%	-6.43%
13	4.84%	8.84%	0.37%	-18.12%	-18.62%	-7.77%
14	-9.94%	5.39%	9.10%	-26.07%	-36.78%	-9.17%
15	-3.41%	8.17%	8.98%	-24.02%	-47.95%	-34.09%
16	5.56%	4.91%	5.96%	-29.24%	-59.75%	-53.96%

TABLE 4. RA time comparison.

$Unit = s$

# of faults	Rs = Cs = 3				Rs = Cs = 5			
	FGPM	OA	HA	EPRA	FGPM	OA	HA	EPRA
4	15.8819	17.5600	16.7497	15.8064	15.7799	17.6378	17.0546	15.8393
5	16.9653	18.8506	17.9214	16.7318	16.4702	19.1222	18.3458	16.2625
6	18.1263	20.4324	19.4543	17.8762	17.5423	20.9753	19.6044	17.3395
7	17.1885	21.8735	20.5090	17.7020	18.7908	22.1338	20.8453	19.1568
8	14.7366	22.8908	21.2814	15.4576	19.3201	24.6979	22.0206	20.2897
9	15.3021	25.8077	23.7112	15.5997	20.2768	25.0722	22.4244	20.4396
10	17.6807	29.2558	26.7439	15.9224	22.1767	26.5313	24.4354	22.1527
11	19.5846	29.8432	27.1384	16.7651	23.8053	28.3769	25.6033	23.3017
12	21.2262	30.6331	27.6584	17.6787	28.3522	30.7903	27.9538	26.5280
13	22.6623	32.1271	28.2489	18.5564	32.1167	33.0109	31.3126	29.6228
14	26.5679	34.6405	28.9275	19.6430	37.4136	34.9354	34.3042	33.9823
15	27.8475	36.6162	29.4528	21.1574	54.1276	36.5930	36.0358	35.6736
16	31.1367	39.3344	30.6895	22.0330	79.9979	38.1103	37.8962	36.8318

sizes vary, the trend of the experimental results is nearly identical. For a fair comparison, various fault distributions are considered [4]–[8]. The faults are distributed in an experiment using the Polya–Eggenberger distribution model [33]–[35], which is widely used in memory fault modeling and is known to be one of the closest to the actual fault distribution among the various fault distribution models. For the reliability of the experiment, each experiment per fault distribution is performed 100,000 times, and the average value is calculated. Additionally, all experiments are conducted with various numbers of faults and spares.

In the experiment, the proposed RA is compared with FGPM [8], Optimal repair analysis Algorithm (OA) and Heuristic repair analysis Algorithm (HA) [13]. FGPM is an RA that repairs memory using only spare rows and columns and adopts fault grouping and pattern matching methods to show optimal repair rate and fast analysis speed. OA is one of the most recent ECC-aware RAs, and it shows high repair rate through greedy use of ECC. HA uses in-memory ECC less greedy than OA, allowing faster solution derivation instead of obtaining a slightly lower repair rate. The performance of the proposed RA is compared and analyzed with the representative studies of the fault grouping method RA and ECC-aware RA.

A. RA TIME

The proposed RA adopts fault grouping and pattern matching. This method groups faults by preprocessing during fault collection time and extracts fault group information. However, unlike RAs of the existing fault grouping and pattern matching method, the proposed RA additionally goes through a process of simplifying large fault groups. In this part, first, fault group simplification occurrence rate of the proposed RA is analyzed by changing the number of faults and the number of available spares.

Fig. 7 shows the rate of fault groups that are simplified according to the number of faults. The number of large fault groups comprising six or more faults does not change according to the number of line spares but is only affected by the total number of faults. For this reason, comparisons are made for the same number of faults regardless of the number of line spares. The figure shows the ratio of the number of fault groups that have been simplified to the number of large fault groups. As the number of faults increases, the percentage of fault groups enabling pattern matching decreases. This can be seen at the ratio of the simplified fault groups. It is correct that the ratio of large fault groups consisting of six or more faults should be affected only by the number of faults, but looking at Fig. 7, it can be seen that the ratio of the simplified fault groups also increases as the number of line spares increases. This is because the number of faults satisfying the must repair condition decreases as the number

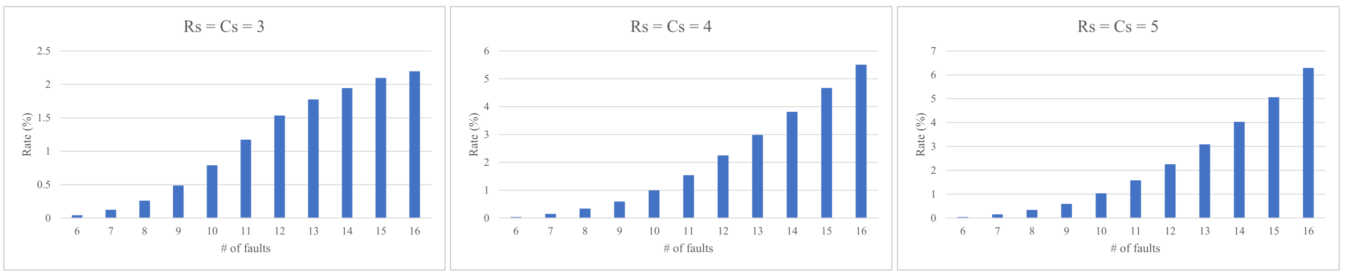

FIGURE 7. Proposed group simplification occurrence rate.

TABLE 5. Repair rare comparison.

# of faults	Rs = Cs = 4				Rs = Cs = 5			
	FGPM	OA	HA	EPRA	FGPM	OA	HA	EPRA
8	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
9	96.01%	100.00%	100.00%	96.01%	100.00%	100.00%	100.00%	100.00%
10	84.87%	100.00%	100.00%	89.23%	100.00%	100.00%	100.00%	100.00%
11	67.40%	100.00%	99.95%	80.78%	99.30%	100.00%	100.00%	99.30%
12	48.28%	100.00%	99.79%	72.57%	96.53%	100.00%	100.00%	97.77%
13	31.30%	100.00%	99.35%	65.74%	89.88%	100.00%	99.98%	95.18%
14	18.08%	100.00%	98.28%	60.83%	79.57%	100.00%	99.92%	92.11%
15	9.79%	100.00%	96.36%	57.94%	66.63%	100.00%	99.67%	88.91%
16	4.84%	100.00%	93.56%	54.96%	52.14%	100.00%	99.27%	86.31%

of line spares increases, resulting in more large fault groups. The pattern matching RA requires solution to search for large fault groups in the analysis process. This occupies a large part of the total analysis time and is one of the factors that hinder the performance of RA. To solve this problem, the proposed RA simplifies large fault groups so that more fault groups can be pattern matched.

However, for pattern simplification, processes of dividing large fault group into codeword units, simplifying and regrouping them are necessary. This is done during the preprocessing process and may affect the total RA time as well as the preprocessing time. To confirm this, FGPM, one of the most effective pattern matching RAs, is compared. Table 3 shows the difference between the preprocessing times of FGPM and EPRA, and difference in the entire repair time. First, looking at the preprocessing time difference, the difference shows a value between -10% and 10%, so there is no big deviation, but overall, the proposed RA takes more time. However, looking at the total RA time, the proposed RA shows a much faster. If the total number of faults is small and the proportion of analysis time out of the RA time is small, it is greatly affected by the difference in preprocessing time. However, as the number of faults increases, the proportion of analysis time increases, and the proposed RA, which simplifies large fault group and generates a minimum solution search, shows a much faster speed than FGPM. This becomes evident as the number of faults increases and the number of available spares increases.

In Table 4, the RA times of FGPM, OA, HA and proposed RA are compared. OA and HA have a different preprocessing process from other two algorithms, so rather than comparing only the preprocessing time separately, the total RA time including this is compared. The RA time of OA is not changed significantly depending on the difference in the number of spares. On the other hand, for FGPM, HA and EPRA, the repair time becomes longer as the number of spares increases. This is because in the case of the three algorithms, the time for the allocation of spares occupies most of the total RA time. Compared to EPRA, the change in the repair time of FGPM is remarkable, and when the number of Rs and Cs is 3 each, it is faster in all sections than OA and HA. However, when the number of spares increases, it is fast in most of the sections, but when the number of faults increases, the repair time also increases rapidly, and eventually it can be confirmed that it is slower than OA and HA. This is due to the increase in repair time as more fault groups need to be searched for solutions. On the other hand, EPRA shows a similar speed to FGPM in the section where the number of faults is small, but there is no significant deviation in repair speed even when the number of faults increases, so it shows the fastest repair speed among four algorithms.

Comparing OA, HA and EPRA in more detail, the data show that EPRA has a faster repair time in all processes compared to OA and HA. As the number of line spares is small and the number of faults increases, the strength of EPRA in repair time is revealed. In the experimental results, there are cases where OA took almost 70% more of the repair time of EPRA. The difference in repair time between OA and HA is not large, but as the number of faults increases, the difference becomes clear. In EPRA, even if the number of faults increases, the change in repair time is not large, but OA and HA increases significantly compared to EPRA. In a situation in which more faults than the suggested number for the experiments, the repair time of OA and EPRA shows a larger difference. This difference in repair time is important in the modern computer industry that produces hundreds of thousands of memories.

B. YIELD

RA aims to achieve high yield by repairing the produced memory as much as possible using limited repair resources. Therefore, the repair rate, which determines how many of faulty memories can be repaired under the same conditions, is one of the factors that directly affect the yield. In this section, the repair rate and yield of the four algorithms (FGPM, OA, HA and EPRA) are compared and analyzed in various conditions.

Looking at the repair rate, the proposed RA uses ECC to improve yield; thus, if it has the same number of line spares, it has a higher repair rate than all RAs using only line spares. In other words, the comparison of the normalized repair rate is meaningless because it shows a higher repair rate than RA, which shows a 100% normalized repair rate based on line spares. For an accurate comparison, this study uses the repair rate instead of the normalized repair rate.

Table 5 summarizes the results of measuring the repair rate of FGPM, OA, HA and EPRA according to the various numbers of line spares and faults. Both OA, HA and EPRA proceed with RA considering in-memory ECC, but there is a big difference in the use of ECC. OA focuses on improving the repair rate by correcting as many faults as possible with ECC. However, such greedy use of ECC can increase the probability of failure during memory operation. On the other hand, EPRA corrects faults with ECC only the minimum number of faults necessary to simplify the fault group. HA is somewhere in between the two algorithms, but it is a little closer to the characteristics of OA. All available spare resources are used, but the use of ECC is given priority.

TABLE 6. Information of fault sets.

No	Standard Deviation	Avergae # of Faults	Maximun # of Faults	No	Standard Deviation	Avergae # of Faults	Maximun # of Faults
1	0.50	0.19	2	11	3.00	4.09	24
2	0.75	0.55	4	12	3.25	4.47	27
3	1.00	0.93	8	13	3.50	4.88	28
4	1.25	1.32	11	14	3.75	5.29	31
5	1.50	1.71	13	15	4.00	5.67	34
6	1.75	2.12	15	16	4.25	6.09	36
7	2.00	2.51	16	17	4.50	6.47	38
8	2.25	2.90	19	18	4.75	6.88	41
9	2.50	3.30	21	19	5.00	7.27	44
10	2.75	3.69	23	20	5.25	7.66	46

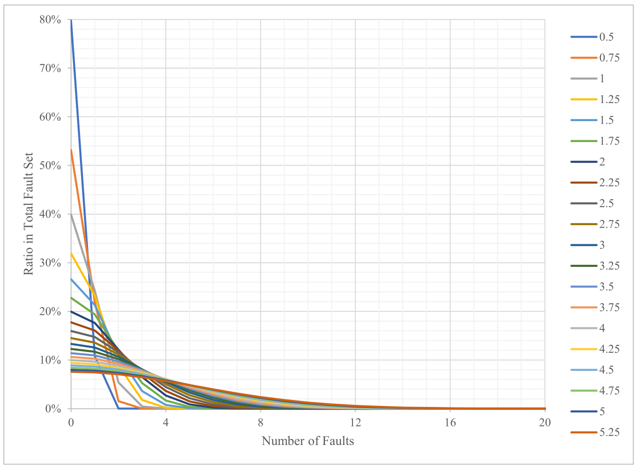

FIGURE 8. Probability of occurrence according to the number of faults for each deviation.

OA shows 100% repair rate over the entire range tested, whereas HA and EPRA do not. This difference is due to the concept of ECC use of the three algorithms. Comparing the repair rates of FGPM and EPRA, the proposed RA does not show a significant difference in repair rates compared to FGPM when the number of faults is small but shows a large difference as the number of faults increases. This is because in the proposed RA, as the number of faults increases, line spares are effectively disposed through fault skip. This feature is evident even when the number of line spares is increased. As the number of line spares increases, the proposed RA has less change in repair rate as the number of faults increases compared to FGPM. To summarize the experiment of comparing the repair rate of the three algorithms, compared to the maximum use of ECC, the repair rate is lower, but the proposed RA is effective as the number of faults increases and the number of usable line spares increases.

OA, HA and EPRA show a big difference in repair rate as the number of faults increases. However, this difference is not very significant in terms of yield. In an actual memory production, various numbers of faults occur in each memory die. Usually, most of dies are faultless or have only a very small number of faults. Therefore, in order to compare the

TABLE 7. RA performance according to fault sets.

Spares	Algorithm	Yield	RA time	ECC Corrected Faults	
				Average	Maximun
Rs = Cs = 3	Fault Set 3				
	FGPM	100.00%	13.2874	-	-
	OA	100.00%	13.6784	0.9264	7
	HA	100.00%	13.2948	0.0000	0
	EPRA	100.00%	13.1984	0.0000	0
	Fault Set 8				
	FGPM	95.90%	14.5235	-	-
	OA	100.00%	16.5701	2.8417	17
	HA	99.96%	15.2049	0.2087	11
	EPRA	96.70%	14.4643	0.0554	6
	Fault Set 10				
	FGPM	90.58%	18.4314	-	-
	OA	100.00%	17.3227	3.6064	20
	HA	99.75%	15.9909	0.5029	14
	EPRA	92.58%	14.8220	0.1461	8
Rs = Cs = 4	Fault Set 4				
	FGPM	100.00%	13.5305	-	-
	OA	100.00%	14.4344	1.2937	10
	HA	100.00%	13.7340	0.0000	1
	EPRA	100.00%	13.5242	0.0000	0
	Fault Set 11				
	FGPM	96.50%	20.8831	-	-
	OA	100.00%	18.3658	3.9963	23
	HA	99.92%	17.2420	0.3174	14
	EPRA	98.02%	16.1487	0.1592	7
	Fault Set 14				
	FGPM	91.01%	497.0473	-	-
	OA	100.00%	19.9072	5.1345	28
	HA	99.47%	17.5520	0.7548	19
	EPRA	95.30%	16.5178	0.4349	9
Rs = Cs = 5	Fault Set 6				
	FGPM	100.00%	14.3915	-	-
	OA	100.00%	15.4776	2.0729	13
	HA	100.00%	14.6464	0.0006	3
	EPRA	100.00%	14.2522	0.0000	0
	Fault Set 16				
	FGPM	95.32%	737.7172	-	-
	OA	100.00%	21.7125	5.8761	31
	HA	99.58%	20.2367	0.6656	18
	EPRA	98.75%	19.0119	0.4435	7
	Fault Set 19				
	FGPM	90.19%	Over	-	-
	OA	100.00%	23.7261	7.0208	37
	HA	98.71%	21.6262	1.0976	21
	EPRA	97.84%	20.2528	0.8138	9

yield of RAs, various fault sets must be used to compare the repair rate.

Memory produced under a mature production process will most likely be fault-free. On the other hand, memories manufactured under an immature process may contain many faults. In order to compare the yield according to each RA in such various cases, the deviation of the normal distribution is increased by 0.25 units from 0.5 to 5.25 and 20 fault sets are generated. Each fault set consists of 100,000 memories, as shown in the repair rate experiment. However, unlike the previous experiment, each memory has various numbers of faults, not the same number of faults. The number of faults in each memory is determined by the probability of occurrence according to the deviation value determined for each fault set. The probability of occurrence according to the number of faults per set is shown in Fig. 8. The deviation, average number of faults, and maximum number of faults of the 20 fault sets created according to the distribution in Fig. 8 are shown in Table 6.

In Table 7, yield, RA time, and the number of hard defects corrected by ECC according to RAs and number of line spares are summarized. For memory producers, unrepaired memory is a cost. Therefore, when the yield falls below a certain level, it is economical to improve the yield by adding spare resources. For this reason, it makes no sense to compare yields below a certain level. In this paper, the yield of FGPM using only line spares for repairing faults is taken as a standard. Among the fault sets made previously for the yield experiment, fault sets showing yields of 100%, 95%, and 90% by the number of line spares when FGPM is applied are selected. The selected fault sets are simulated by applying OA, HA and EPRA, and the results are shown in Table 7.

Looking at the yield, it shows that when the number of faults is large, the repair rate of OA, HA and EPRA shows a large difference, but there is no significant difference in the yield. Even in the case of the largest difference, the yield of OA is only 1.08 times that of EPRA, and the difference in yield decreases as the number of line spares increases. HA also has a similar value to OA, which is 1.07 times higher than that of EPRA. RA times, as analyzed in section 4.1 above, are faster for EPRA than for other algorithms. This becomes evident as the average number of faults in fault set increases. The number of faults corrected by ECC shows the characteristics of each algorithm. Except for FGPM, which does not use ECC in RA, EPRA corrects only a very small number of faults on average compared to OA and HA repairing as many faults as possible with ECC. This is also evident in the maximum value of faults corrected by ECC. A detailed analysis of this will be described in the next session.

In summary, EPRA shows lower repair rate than OA or HA when the number of faults is large. But this does not affect the yield, because this situation does not happen very often. The difference in yield is only as small as 1.08 times at most, but to make this difference, OA must correct so many faults with ECC compared to EPRA. There is no significant difference in yield with HA, and relatively few faults are repaired with

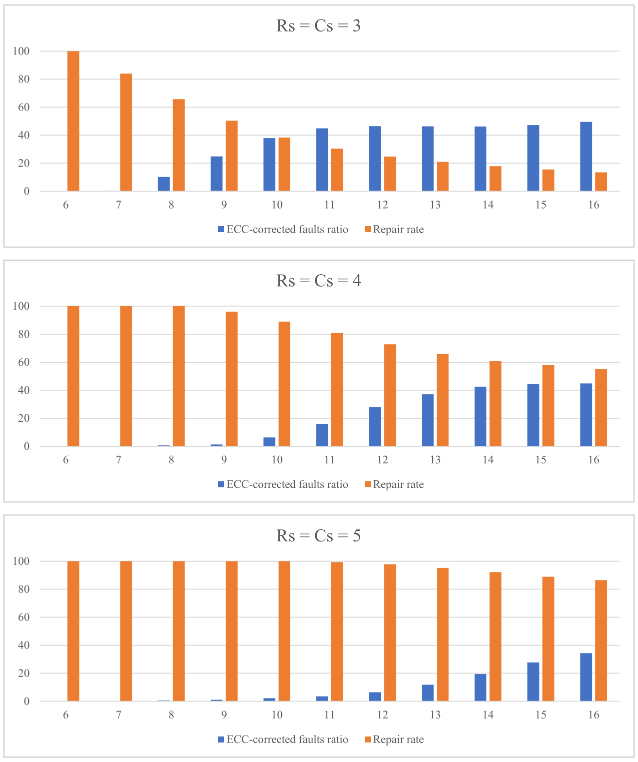

FIGURE 9. Ratio of faults corrected by in-memory ECC.

ECC compared to OA, but HA also uses more ECC compared to EPRA like OA.

C. RELIABILITY

In-memory ECC, which has been proposed to improve memory reliability by correcting soft errors that occur during online operation, has been actively studied to improve yield because it effectively repairs single faults. The proposed RA also uses ECC and line spares to improve the yield. However, an excessive use of ECC increases the probability of future memory failures. To solve this problem, EPRA uses as many line spares as possible and establishes usage rules so that ECC can be used most effectively. In this experiment, the number of faults that the proposed RA can repair through ECC, as well as whether ECC use for repairing hard defects does not impair the memory reliability, is analyzed.

In Fig. 9, the number of faults corrected by in-memory ECC is shown when the proposed RA is applied while changing the numbers of faults and line spares. In each case, the ECC-corrected fault ratio increases with the number of faults, up to a certain level. When it reaches a certain level, the repair rate drops. This shows that EPRA fault skip rule is proposed in consideration of efficient use, rather than indiscriminately increasing the repair rate using ECC. Analyzing the number of faults that single line spare repairs on average, in the proposed RA, this value also increases as the number of failures increases. This shows that ECC is being used to increase the efficiency of using line spares.

TABLE 8. Yield and failure rate of OA and HA against EPRA.

Spare	Fault Set	Yield Per EPM (%)		Failure Rate (‱)					
				Average			Top 10%		
		OA	HA	OA	HA	EPM	OA	HA	EPM
Rs = Cs = 3	Set 3	100.00%	100.00%	1.19	0.00	0.00	5.01	0.00	0.00
	Set 8	103.41%	103.37%	3.65	0.27	0.07	11.62	2.71	0.72
	Set 10	108.01%	107.74%	4.67	0.65	0.19	14.22	5.67	1.91
Rs = Cs = 4	Set 4	100.00%	100.00%	1.69	0.00	0.00	6.38	0.00	0.00
	Set 11	102.02%	101.94%	5.15	0.42	0.20	15.47	4.15	1.42
	Set 14	104.93%	104.38%	6.64	0.98	0.56	19.22	7.78	3.93
Rs = Cs = 5	Set 6	100.00%	100.00%	2.69	0.00	0.00	9.11	0.08	0.01
	Set 16	101.27%	100.84%	7.58	0.87	0.57	21.66	7.56	3.57
	Set 19	102.21%	100.89%	9.02	1.43	1.05	25.17	10.32	6.22
Rs = Cs = 6	Set 8	100.00%	100.00%	4.40	0.00	0.00	14.05	0.24	0.03
	Set 21	100.77%	100.13%	9.92	1.76	1.47	29.14	11.12	7.35
	Set 26	101.17%	100.34%	11.44	2.27	1.95	31.45	13.58	10.44
Rs = Cs = 7	Set 10	100.00%	100.00%	7.19	0.00	0.00	24.57	1.47	0.13
	Set 27	100.21%	100.05%	11.84	3.18	2.75	32.16	15.96	15.12
	Set 33	100.64%	100.12%	13.71	3.75	3.52	40.51	18.30	17.52

In the case of hard defects in the memory discovered in the production stage, even if they are not repaired, they can be corrected through ECC. In this case, it does not significantly affect actual memory operation. However, if a soft error occurs while reading a codeword with a hard defect that has been corrected by ECC, memory failure occurs. Since ECC can only correct one error in one codeword, it fails to read the codeword. This memory failure can greatly affect the overall operation of the system. Memory reliability is one of the most important factors in the stability of the system, and for the above reasons, the use of ECC in RAs that repair hard defects should be cautious. Due to the feature of ECC that cannot correct a plurality of faults, a codeword in which a hard defect to be corrected by ECC is located leads to a memory failure when a soft error occurs. That is, the increase in the probability of memory failure occurring as the RA considers ECC is proportional to the number of hard defects corrected by ECC.

Of the four RAs covered in the experiment in this paper, FGPM does not consider ECC in the RA process, so there is no additional change in the probability of memory failure. On the other hand, since OA, HA and EPRA are ECC-aware RAs, it is necessary to compare the probability of additional memory failure. An additional failure rate experiment is performed to measure memory failure caused by correction of hard defects using ECC in each algorithm. Instead of repairing some of the hard defects found in offline, ECC-aware RAs correct them using ECC in online. Memory failure occurs when an error in online occurs in the same code word as the hard defect to be corrected. That is, the failure rate per error is proportional to the number of codewords where hard defects to be corrected by ECC are located. To check the exact failure rate per error, the fault set for this experiment consist of 10,000 random errors. In this experiment, it is assumed that an error occurs in the requested codeword when the system reads the memory. When an error occurs in the read codeword 10,000 times, the failure rate is derived by checking the percentage of memory failures that an error cannot be corrected by ECC. These errors are online errors and include both soft errors and hard defects. This fault set is assigned to the memories that were determined to be good in the yield experiment. In Table 8, the yield and failure rate of OA, HA, and EPRA are summarized.

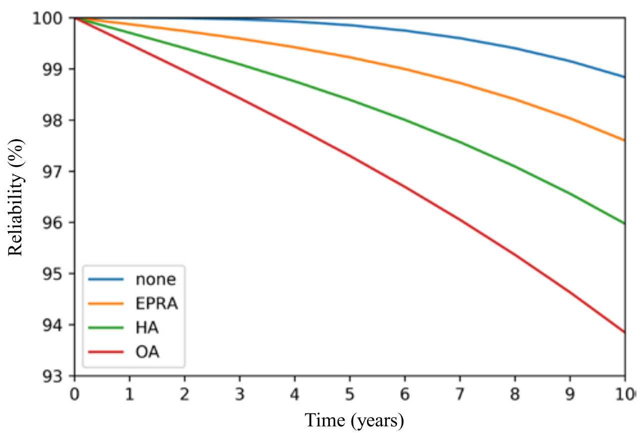

FIGURE 10. The reliability function of a memory in the RAs.

From the experimental results, the memory repaired by the OA that uses ECC greedy shows a very high failure rate in all cases. HA shows a very low failure rate compared to OA. However, the proposed RA shows a lower failure rate than HA, and thus shows the highest reliability among the three algorithms in all cases. As the number of spares increases, the difference in yield between the three algorithms decreases. This proves that the proposed RA focuses on the efficient use of line spares. Nevertheless, although the difference is smaller than when the number of spares is small, compared to OA and HA, EPRA clearly shows a low failure rate in all cases. Especially when looking at the memory with the top 10% failure rate, it shows at least nearly half the failure rate in the same condition.

This is due to the concept of each RA. The failure rate is proportional to the number of faults corrected by ECC. OA shows the highest failure rate because all ECC-correctable faults are corrected by ECC. However, in HA, ECC-uncorrectable faults are repaired by line spares. After repairing ECC-uncorrectable faults, among the remaining faults, ECC-correctable faults are corrected by ECC. On the other hand, the proposed RA corrects ECC-correctable faults by ECC only for faults groups consisting of a large number of faults. As the number of faults increases, the number of fault groups consisting of a large number of faults and ECC-uncorrectable faults increases. Because of this, HA and the proposed RA gradually show a similar failure rate. However, due to the criteria for selecting faults to be corrected by ECC, the proposed RA always corrects fewer faults than HA by ECC.

In actual use, the errors of memory in online occur due to various causes such as the aging of elements [36], the failure of modules in memory [37]. When correcting data of the

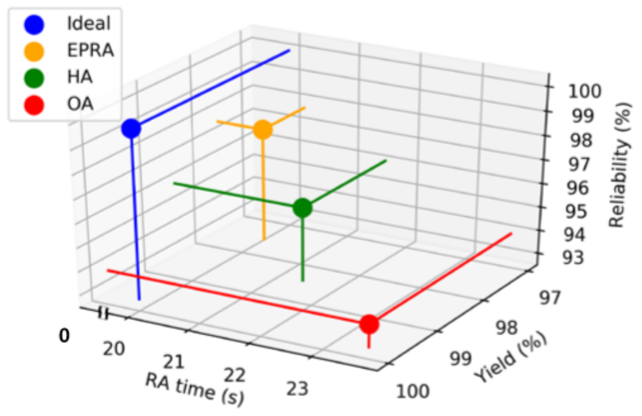

FIGURE 11. Comparison of overall performance of ECC-aware RAs.

memory in online, regardless of the cause of the error, it is recognized as an error in the memory cell and corrected by ECC. As such, from the perspective of correcting the memory in online, the frequency of the memory cell error is considered regardless of the cause of the error. For this reason, the error rate of memory in online [31], [32], [38] is used to analyze the reliability. According to the studies, in this experiment, the soft error rate is set to 10-10 per hour for the memory bits. To compare the reliability of each algorithm, the previous experimental results are substituted into Equation (8).

If ECC is not used to repair the hard defect, the reliability of the memory after 10 years is 98.95%. Looking at the reliability of ECC-aware RAs, EPRA is 97.71%, HA is 96.08%, and OA is 93.96%. In the case of EPRA, which uses less ECC, it shows a 1.25% decrease in reliability compared to when ECC is not used. On the other hand, HA shows 2.9% and OA shows 5.05% decrease in reliability. This is a decline in reliability that is 2.31 times and 4.03 times greater compared to EPRA, respectively. The reliability functions of memory with three ECC-aware RAs and memory without using ECC for repairing hard defects are shown in Fig. 10. The reliability of memory can also be determined as a parameter of Mean Time To Failure (MTTF), which is defined as the total execution time divided by the total number of failures. It is measured for estimating the average lifetime of a chip. The MTTF is obtained by integrating the reliability function with time [32], [37]. Calculating the MTTF, FGPM is 624,946 hours, HA is 514,556 hours, OA is 445,134 hours, and the proposed method is 574,672 hours. The MTTF of the proposed method is 8.04% less than FGPM, but 11.68% higher than HA and 29.1% higher than OA. In summary, EPRA, like OA and HA, is ECC-aware RA, which uses ECC to repair hard defects in offline. However, since ECC is used for efficient use of line spares in EPRA, the indiscriminate use of ECC is reduced, thus showing very high reliability compared to the two algorithms.

Fig. 11 shows a comparison of overall performance for ECC-aware RA, such as with RA time, yield, reliability for EPRA, OA, HA, and ideal RA. To compare RAs clearly, the case of fault set 19 in Table 7 is used. For an ideal BIRA, it is assumed that the RA time is zero and its yield and repair rate are 100%. According to Fig. 11, OA has the highest yield, but the slowest speed and the lowest reliability. HA has a slightly lower yield than OA but has a faster RA time and higher reliability. HA also shows good performance, but EPRA shows the fastest RA time and the highest reliability. Therefore, the proposed ECC-aware RA performs better than OA and HA.

V. CONCLUSION

The RA and ECC methodologies are widely used to improve the yield and reliability of memory, respectively. As ECC is used for improving not only the reliability but also the yield, there is a need to study an effective RA that considers ECC. To address these problems, EPRA has been proposed. This is a fault group pattern matching-based RA and involves a fault group simplification method and a new early termination method considering ECC. This increases the possibility of pattern matching of fault groups and enables a quick determination of whether to repair when considering ECC. The proposed algorithm achieves a similar yield but 14.6% less RA time and 8.6 times higher reliability compared to the previous studies. Therefore, this is an effective ECC-aware RA method for repair.

REFERENCES

[1] W. Jeong, I. Kang, K. Jin, and S. Kang, "A fast built-in redundancy analysis for memories with optimal repair rate using a line-based search tree," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 12, pp. 1665–1678, Dec. 2009.
[2] A. Hwang, I. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice: Understanding the nature of DRAM errors and the implications for system design," in *Proc. Int. Conf. Architectural Support Program. Lang. Oper. Syst. (ASPLOS)*, London, U.K., Mar. 2012, pp. 111–122.
[3] K. K. Hung, P. K. Ko, C. Hu, and Y. C. Cheng, "Random telegraph noise of deep-submicrometer MOSFETs," *IEEE Electron Device Lett.*, vol. 11, no. 2, pp. 90–92, Feb. 1990.
[4] M. Tarr, D. Boundreau, and R. Murphy, "Defect analysis system speeds test and repair of redundant memories," *Electronics*, vol. 57, no. 1, pp. 175–179, Jan. 1984.
[5] S.-Y. Kuo and W. K. Fuchs, "Efficient spare allocation in reconfigurable arrays," in *Proc. Design Autom. Conf.*, Las Vegas, NV, USA, Jun./Jul. 1986, pp. 385–390.
[6] H. Cho, W. Kang, and S. Kang, "A fast redundancy analysis algorithm in ATE for repairing faulty memories," *ETRI J.*, vol. 34, no. 3, pp. 478–481, Jun. 2012.
[7] H. Cho, W. Kang, and S. Kang, "A very efficient redundancy analysis method using fault grouping," *ETRI J.*, vol. 34, no. 3, pp. 478–481, Jun. 2012.
[8] H. Lee, K. Cho, D. Kim, and S. Kang, "Fault group pattern matching with efficient early termination for high-speed redundancy analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1473–1482, Jul. 2018.
[9] C. Stapper and R. Rosner, "Integrated circuit yield management and yield analysis: Development and implementation," *IEEE Trans. Semicond. Manuf.*, vol. 8, no. 2, pp. 95–102, May 1995.
[10] M. Horiguchi *et al.*, "Redundancy," in *Nanoscale Memory Repair*. New York, NY, USA: Springer, 2011, pp. 19–67.
[11] V. R. S. Rao and M. A. Rani, "Self healing memory using 2D spares," in *Proc. 5th Int. Conf. Comput. Methodol. Commun. (ICCMC)*, Erode, India, Apr. 2021, pp. 695–699.
[12] D. Kim, H. Lee, and S. Kang, "An area-efficient BIRA with 1-D spare segments," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 1, pp. 206–210, Jan. 2018.
[13] M. Lv, H. Sun, J. Xin, and N. Zheng, "Efficient repair analysis algorithm exploration for memory with redundancy and in-memory ECC," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 775–788, May 2021.

[14] Y. H. Son, S. Lee, O. Seongil, S. Kwon, N. S. Kim, and J. H. Ahn, "CiDRA: A cache-inspired DRAM resilience architecture," in *Proc. IEEE Symp. High Perform. Comput. Architecture (HPCA)*, Burlingame, CA, USA, Feb. 2015, pp. 502–513.

[15] S.-I. Pae, V. Kozhikkottu, D. Somasekar, W. Wu, S. G. Ramasubramanian, M. Dadual, H. Cho, and K.-W. Kwon, "Minimal aliasing single-error-correction codes for DRAM reliability improvement," *IEEE Access*, vol. 9, pp. 29862–29869, 2021.

[16] U. Kang, H.-S. Yu, C. Park, H. Zheng, J. Halbert, K. Bains, S. Jang, and J. S. Choi, "Co-architecting controllers and DRAM to enhance DRAM process scaling," *Memory Forum*, vol. 14, pp. 1–14, Jun. 2014.

[17] S. Kwon, Y. Hoon Son, and J. Ho Ahn, "Understanding DDR4 in pursuit of in-DRAM ECC," in *Proc. Int. SoC Design Conf. (ISOCC)*, Jeju, South Korea, Nov. 2014, pp. 276–277.

[18] *LPDDR4 SDRAM Specification*, JEDEC Standard JESD209-4, 2014.

[19] T.-Y. Oh *et al.*, "A 3.2 Gbps/pin 8 Gbit 1.0 V LPDDR4 SDRAM with integrated ECC engine for sub-1 V DRAM core operation," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 178–190, Jan. 2015.

[20] K. C. Chun *et al.*, "A 16 Gb LPDDR4X SDRAM with an NBTI-tolerant circuit solution, an SWD PMOS GIDL reduction technique, an adaptive gear-down scheme and a metastable-free DQS aligner in a 10 nm class DRAM process," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2018, pp. 206–208.

[21] K. C. Chun *et al.*, "A 16-GB 640-GB/s HBM2E DRAM with a data-bus window extension technique and a synergetic on-die ECC scheme," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 199–211, Jan. 2021.

[22] H. Kalter, C. H. Stapper, J. E. Barth, J. DiLorenzo, C. E. Drake, J. A. Fifield, G. A. Kelley, S. C. Lewis, W. B. Van Der Hoeven, and J. A. Yankosky, "A 50-ns 16-Mb DRAM with a 10-ns data rate and on-chip ECC," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1118–1128, Oct. 1990.

[23] S. Cha, O. Seongil, H. Shin, S. Hwang, K. Park, S. J. Jang, J. S. Choi, G. Y. Jin, Y. H. Son, H. Cho, J. H. Ahn, and N. S. Kim, "Defect analysis and cost-effective resilience architecture for future DRAM devices," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture (HPCA)*, Austin, TX, USA, Feb. 2017, pp. 61–72.

[24] H. Sun, J. Zhao, F. Wang, N. Zheng, and T. Zhang, "Cost-efficient built-in repair analysis for embedded memories with on-chip ECC," in *Proc. 1st Int. Symp. Access Spaces (ISAS)*, Yokohama, Japan, Jun. 2011, pp. 95–100.

[25] C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," *IEEE Trans. Rel.*, vol. 52, no. 4, pp. 386–399, Dec. 2003.

[26] J.-H. Lee, K.-H. Park, and S.-H. Kang, "High-efficiency BIRA for embedded memories with a high repair rate and low area overhead," *J. Semicond. Technol. Sci.*, vol. 12, no. 3, pp. 266–269, Sep. 2012.

[27] W. Jeong, J. Lee, T. Han, K. Lee, and S. Kang, "An advanced BIRA for memories with an optimal repair rate and fast analysis speed by using a branch analyzer," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 2014–2026, Dec. 2010.

[28] S. Shoukourian, V. Vardanian, and Y. Zorian, "SoC yield optimization via an embedded-memory test and repair infrastructure," *IEEE Des. Test*, vol. 21, no. 3, pp. 200–207, May/Jun. 2004.

[29] S. Shoukourian, V. A. Vardanian, and Y. Zorian, "A methodology for design and evaluation of redundancy allocation algorithms," in *Proc. VLSI Test Symp.*, Napa Valley, CA, USA, Apr. 2004, pp. 249–255.

[30] S. Bahl, "A sharable built-in self-repair for semiconductor memories with 2-D redundancy scheme," in *Proc. 22nd IEEE Int. Symp. Defect Fault-Tolerance VLSI Syst.*, Rome, Italy, Sep. 2007, pp. 331–339.

[31] T.-H. Wu, P.-Y. Chen, M. Lee, B.-Y. Lin, C.-W. Wu, C.-H. Tien, H.-C. Lin, H. Chen, C.-N. Peng, and M.-J. Wang, "A memory yield improvement scheme combining built-in self-repair and error correction codes," in *Proc. IEEE Int. Test Conf.*, Anaheim, CA, USA, Nov. 2012, pp. 1–9.

[32] C.-L. Su, Y.-T. Yeh, and C.-W. Wu, "An integrated ECC and redundancy repair scheme for memory reliability enhancement," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Monterey, CA, USA, Oct. 2005, pp. 81–89.

[33] C. H. Stapper, "On a composite model to the IC yield problem," *IEEE J. Solid-State Circuits*, vol. SSC-10, no. 6, pp. 537–539, Dec. 1975.

[34] C. H. Stapper, A. N. McLaren, and M. Dreckmann, "Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product," *IBM J. Res. Develop.*, vol. 24, no. 3, pp. 398–409, May 1980.

[35] R.-F. Huang, J.-F. Li, J.-C. Yeh, and C.-W. Wu, "A simulator for evaluating redundancy analysis algorithms of repairable embedded memories," in *Proc. IEEE Int. Workshop Memory Technol., Design Testing*, Bendor, France, Jul. 2002, pp. 68–73.

[36] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Machine learning-based approach for hardware faults prediction," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 11, pp. 3880–3892, Nov. 2020.

[37] K. Khalil, O. Eldash, A. Kumar, and M. Bayoumi, "Intelligent fault-prediction assisted self-healing for embryonic hardware," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 4, pp. 852–866, Aug. 2020.

[38] D.-H. Kim and L. Milor, "An ECC-assisted postpackage repair methodology in main memory systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 7, pp. 2045–2058, Jul. 2017.