Fault Clustering Technique for 3D Memory BISR

Tianjian Li[1], Yan Han[1], Xiaoyao Liang[1], Hsien-Hsin S. Lee[2] and Li Jiang[1*]
[1] Department of Computer Science & Engineering, Shanghai Jiao Tong University
[2]Taiwan Semiconductor Manufacturing Company, Ltd.
Email: {ltj2013, hy123321, liang-xy, ljiang_cs}@sjtu.edu.cn, hhleeq@tsmc.com

***Abstract*—Three Dimensional (3D) memory has gained a great momentum because of its large storage capacity, bandwidth and etc. A critical challenge for 3D memory is the significant yield loss due to the disruptive integration process: any memory die that cannot be successfully repaired leads to the failure of the whole stack. The repair ratio of each die must be as high as possible to guarantee the overall yield. Existing memory repair methods, however, follow the traditional way of using redundancies: a redundant row/column replaces a row/column containing few or even one faulty cell. We propose a novel technique specifically in 3D memory that can overcome this limitation. It can cluster faulty cells across layers to the same row/column in the same memory array so that each redundant row/column can repair more "faults". Moreover, it can be applied to the existing repair algorithms. We design the BIST and BISR modules to implement the proposed repair technique. Experimental results show more than 71% enhancement of the repair ratio over the global 3D GESP solution and 80% redundancy-cost reduction, respectively.**

I. Introduction

Three Dimensional (3D) memory has been regarded as a promising alternative to the conventional 2D memory. In a typical 3D stacked memory [1], as shown in Fig. 1, a peripheral die and multiple memory dies are vertically stacked. Through Silicon-Vias (TSVs) are deployed as the address and data lines across the layers. The decoder is partitioned and distributed in the peripheral layer and memory layers; given an address, it will first choose which layer to access and then choose which memory array to access. The separated memory die and peripheral die can be optimized for device density and logic performance, respectively [2]. By using TSVs, 3D memory can provide much higher bandwidth, smaller latency and lower power consumption when compared to conventional 2D memories; it can increase the memory capacity without scaling down the devices. Many 3D stacked DRAM products have been commercialized [3], [4]. In addition, memories with heterogenous technologies can be easily integrated in 3D fashion, such as monolithically 3D integrated RRAMs [5] and STT-RAMs [6], rendering new opportunities in non-volatile 3D memory.

Despite these promising advantages, the relative high cost (low yield) is one of the most critical challenges in the 3D memory technology due to the disruptive integration process [7]: any memory die that can not be successfully repaired leads to the failure of the whole stack. As reported in [8], the yield of 3D memories drops significantly as the number of stacked memory layers increases. Conventionally, two dimensional (2D) hardware redundancy—spare rows and columns used to replace the rows and columns containing faults—is widely adopted to improve the memory yield. Various repair algorithms can obtain the optimal repair ratio for these 2D redundancy memory architecture [9]. These repair algorithms and the one implemented in built-in self-repair (BISR) mechanisms can be properly applied in 3D memories [8].

This work is partly supported by the National Natural Science Foundation of China (Grant No. 61602300, Grant No. 61202026 and No. 61332001), Shanghai Science and Technology Committee (Grant No. 15YF1406000), and Program of China National 1000 Young Talent Plan. *Li Jiang is the corresponding author.

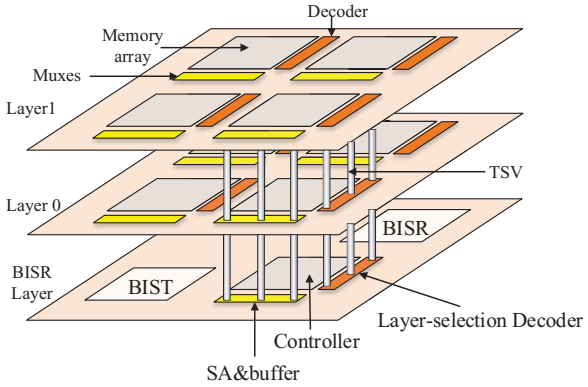

Fig. 1: An example of 3D stacked memory.

Effective utilization of the redundancy is important for 3D memory that is sensitive to cost issue. Simply adding more redundancy to increase the repair ratio may cause unacceptable hardware overhead and complex routing and thereby may not be cost-effective. On the one hand, various methods have been developed to utilize the redundancies in a more effective way [10], [11], [12], [13]. The basic idea is to leverage the die-to-die variation of defect density. The memory layer that cannot repair itself is allowed to borrow redundancy from other memory layer that has unused redundancy. The global 3D GESP algorithm outperforms the existing 2D redundancy sharing methods [13]. In these works, however, the redundancy is not fully utilized because a spare row/column repairs a row/column with few or even single faulty cell. On the other hand, [14], [15] use more flexible pointer/cache-based schemes to replace faulty memory cells using redundant cells. In contrast to replace the whole row/column, these techniques store a word (e.g., 16 bit) in a faulty cache to replace the original word containing the faulty bits, rendering finer-grained granularity of redundancy. However, their efficiency drops if most words in the faulty cache only contain few faulty bits. Not to mention their design complexity and large cache overhead.

Therefore, in this paper, we propose a novel technique to further improve the utilization of redundancies in 3D stacked memory. In contrast to share the spare rows/columns across the memory dies, we share the "faulty cells" across the memory dies. Intuitively, we can cluster the faults from different memory layers to the same row/column; as a result, each spare row/column can repair more faulty cells. The proposed *fault clustering* technique achieve the above objective by exchanging the row/column address among different memory array across die and consequently reallocating the faults associated with the exchanged row/column in the redundancy CAM. We redesign the conventional BIST and BISR modules to accommodate the proposed *fault clustering* technique. This technique, on the one hand, can be applied to any repair algorithms in 1D/2D redundancy architecture. On the other hand, when applied to pointer/cache-based

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

redundancy architecture [14], [15], it can cluster the faulty bits within a cache line/word to reduce the overhead of faulty cache. Due to the page limit, in this paper, we demonstrate the efficiency of the proposed technique by proposing the 3D Cluster-ESP algorithm which is based on the global based GESP algorithm.

The rest of the paper is organized as follows. Section II provides the background and motivates this paper. Section III-A, III-B and III-C propose the *fault clustering* techniques and the 3D Cluster-ESP algorithm. Section III-D optimizes the *fault clustering* technique. The experimental results are shown in section IV. Section V concludes the paper.

II. Related Works and Motivation

In this section, we introduce the related works on 3D memory repair techniques and motivate this paper.

A. Prior works on 3D Memory Repair

Various bonding strategies (e.g., die-to-die bonding or wafer-to-wafer bonding) and stacking flows result in diverse yields for 3D stacked memory [11]. For die-to-die bonding, pre-bond testing can derive the fault information of each memory die; redundancy analysis algorithms (RA), i.e., memory repair algorithms, then determine whether the memory die is self-repairable. Those self-repairable memory dies are stacked while others are discarded. In contrast, post-bond test can detect the faults in all the memory dies and those faults are induced during the period of die stacking. RA algorithms then repair all the memory dies. The stack can be useless if any memory die cannot repair itself.

The memory test and repair are normally accommodated in BIST and BISR modules. Two schemes exist in 3D memory: 3D (Parallel Test and Parallel Repair) (PTPR) and 3D (Serial Test and Serial Repair) (STSR) [8]. In 3D PTPR method, each memory die has individual BIST and BISR modules. Thus, all the memory dies can be tested and repaired in parallel. However, this method must pay large hardware overhead for the short time cost of test and repair. On the contrary, 3D STSR scheme only requires one BIST module and BISR module, normally in the peripheral die, which test and repair all the memory dies serially. It trades the time cost of test and repair for the hardware overhead. Comparing these two schemes, the latter one is more preferable as it provides a chance of more efficient utilization of redundancy for higher yield, as described below.

Various techniques to share the redundancy are proposed in 1D/2D redundancy architecture. Bahl et al. [16] propose to share the redundancy among memory arrays in 2D memory. However, accessing the shared redundancy must go through the long interconnect between blocks, leading to large access latency and routing overhead. The short and densely placed TSVs in 3D memory, on the one hand, can eliminate the above limitation; on the other hand, sharing redundancy across dies leverages the die-to-die variation of defect density. Chou et al. [17] propose to salvage good memory blocks inside of the bad irreparable memory dies by TSVs and Muxes. However, the placements are accomplished at block-level rather than row-level; the utilization of redundancy is less efficient than the following techniques. Chou et al. [11] propose inter-die sharing of spare columns and the optimized stacking flows to enhance the yield of 3D memory. This technique, however, only shares spare columns. In [10], both spare rows and columns are sharable across layers; thereby, sophisticated repair algorithms and a series of die matching algorithms are developed for more efficient utilization of redundancy. This work shows half amount of the redundancy saving to achieve the same yield. But, it can only share spares between two neighboring stacked memory dies. Chi et al. [18] provide a BISR architecture that can shares cross-die spares, associated with pre-bond and post-bond test/repair flow and a die matching heuristic. Instead of sharing redundancy across memory layers, Wang et al. [13] propose a *global sharing* method for the 3D memory, wherein the redundancy is only deployed in the peripheral die. The global redundancy can be used by any memory die at row-level. Because of this flexibility, this work can outperform the above redundancy sharing schemes. Moreover, it can test/repair all the memory die in parallel.

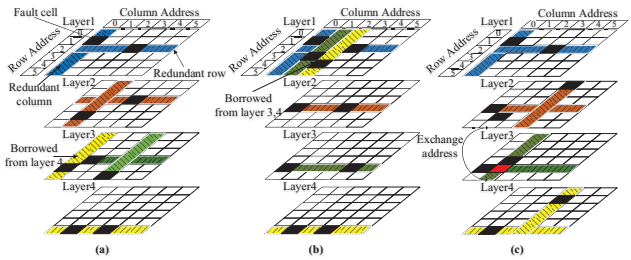

Fig. 2: Example of redundancy share techniques. (a) Inter-die Redundancy Share; (b) Global Redundancy Share; (c) Fault Clustering.

B. Motivation

Despite all that prior works have developed in improving the utilization of redundancy, they all fall into a limitation: a redundant row/column replaces a row/column containing few or even one faulty cell. A lot of redundant resources are wasted when the faulty cells are sparsely or uniformly distributed in the memory arrays. Intuitively, if faulty cells in different rows/columns can be clustered into a single row/column, we can increase the repair ratio to yet another level with the same amount of redundancy. Unfortunately, it is difficult and not cost-effective, if not impossible, to cluster the faulty cells in 2D memory dies. Instead, exchanging memory rows/columns across memory arrays vertically stacked provides a way to cluster fault cells in different layers into one layer.

We use an example to show our basic idea. We assume four memory layers, each of which has one memory array; and each memory array has one redundant row and one redundant column. Obviously, the conventional repair method in 3D memory cannot repair the faults in layer 3, as shown in Fig. 2(a). Inter-die redundancy share scheme, e.g., [10], can repair the faults in layer 3 by borrowing a redundant column from layer 4. But, it cannot repair the faults of layer 1, as shown in Fig. 2(b): layer 1 is irreparable even it borrows a redundancy from layer 2. With global redundancy share scheme [13], we can repair all the faults by borrowing more redundancy from layer 3 and 4. However, the global redundancy share method can do nothing about the faults shown in Fig. 2(c). By exchanging the address, the two rows in layer 1 containing the faulty cells, each requiring a redundant row, are now reallocated as shown in the figure; the faulty cells are now in the same rows with other faulty cells in layer 2 and 3. The redundant row in both layer 2 and 3 can repair more faults, rendering all faulty cells repaired with the same amount of redundancy. The remaining problem is how to achieve the above fault clustering legally in

Therefore, in this work, we explain why above *fault cluttering* is legal and efficient in 3D memory and describe the 3D memory design in detail to embrace the *fault clustering* techniques. Consequently, we apply the *fault clustering* techniques to the conventional 3D GESP algorithm—develop a novel *3D Cluster-ESP algorithm*—as well as the BISR design. Then, we discuss the optimal configuration of the

fault clustering to make trade-offs between the repair ratio and the hardware overhead.

III. Proposed Approach

In this section, we first describe our BISR and the principle of fault clustering. Consequently, we describe the fault clustering algorithm. At last, as a case study, we adapt the fault clustering to 3D Global ESP algorithm, which is called 3D Cluster-ESP algorithm.

A. Proposed 3D BISR with Fault Clustering

The proposed 3D BISR architecture adopts a global share redundancy scheme. As shown in Fig. 3, the conventional BISR consists of fault collection registers, redundancy analysis (RA) block, redundancy Content Addressable Memories (CAM) and control circuits. The fault collection registers store the locations of faulty cells derived from BIST. The RA block fetches the location of faulty cells and executes the redundancy analysis algorithm (3D GESP in this paper). Based on the available redundancy, RA block determines to replace the row/column with fault cells in the array with the redundancy. And the informations of replacing will be transfered to the redundancy CAM. When the input address is the fault cell address, the redundancy CAM will map it to the redundancy address.

Conventionally, the BISR works as follows: when the input address is decoded, both the memory array and the redundancy CAM are accessed concurrently. If the input address is not found in the redundancy CAM, the content derived from the memory array is the final output. In contrast, if the input address is found, the content X in the redundancy CAM is merged with the content Y from memory array, in which the data from faulty cell in Y is replaced by the data from X. Use Fig. 4 (a) as an example. One column in Array 1 c_{13} has four data, A, B, C and D. It has a physical faulty cell and thereby B cannot be fetched out correctly. A redundant row is used to replace the row r_{12}; thus, the fault CAM contains all the data in r_{12}. If we access r_{12}, the fault CAM then provides the data to merge the data of r_{12} and deliver the correct data. Similar behavior can be observed in Array 2. Two redundant rows are required to repair all the faults in this example.

In our proposed BISR architecture, the fault clustering algorithm, carried out in the RA block, analyzes the faulty maps of all the vertically aligned memory arrays in different layers and decides the exchange of rows/columns. The addresses involved in above exchanges are stored into address CAM. RA algorithm, also carried out in the RA block, allocates the redundancy to repair all the faults based on the new address information. The RA algorithm is tricked

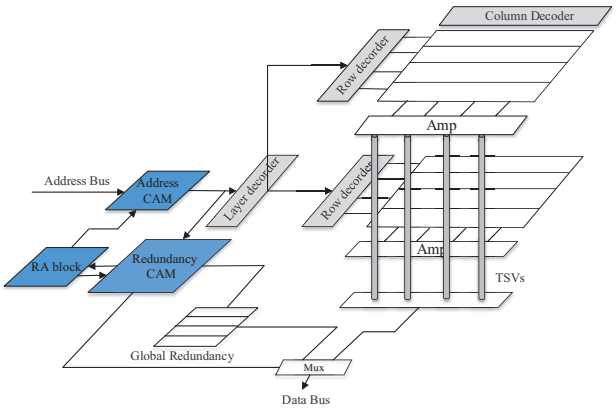

Fig. 3: Overview of 3D global architecture.

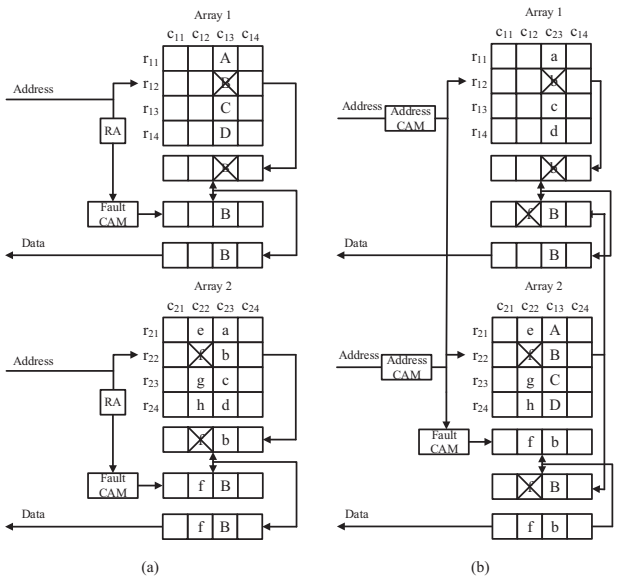

Fig. 4: The principle of fault clustering.

by the faults that disguise themselves with "exchanged" address. Magically, the faults from other layers are clustered in the same row/column for repair.

We disclose the magic in the above process using the example in Fig. 4(b). We exchange c_{13} and c_{23} and the data in Array 1 and 2 are exchanged as shown in the figure. When the input address arrives for memory access, the address CAM is first accessed and searched for the input address. If the address has been exchanged, the decoder will access the new physical location in the memory array. For example, the data of c_{13} is now stored in Array 2 without any faults, while the data of c_{23} is now in Array 1 and the data "b" is damaged. In the faulty CAM, both data "b" and "f" are actually stored in the same redundant row used to replace r_{22}. When r_{22} is accessed, its data is merged with the one—derived from the redundant column —in the redundancy CAM. Compared to the conventional repair solution, in Fig. 4(a), only one redundant row is required. It should be noted that, in the above example, we cannot store "b" and "f" in the same redundant column because it will damage the data integraty.

The fault clustering technique must be implemented in 3D memory. First, it is meaningless to exchange two rows/columns in the same memory array; because no matter how we exchange the address rows/columns, we cannot cluster the faulty cells that are initially in different rows/columns into a single row/column. Moreover, in 2D memory, it is not cost-effective to exchange the rows/columns between two memory arrays due to the area overheads and routing complexity. Fortunately, no such problems exist in 3D memory if the extreme short TSVs are used for routing.

The detail of fault clustering algorithm—how to decide the pairs of rows/columns for exchange—is then described in the next section.

B. Fault Clustering Algorithm

In order to clearly describe the fault clustering algorithm, we define two terms: the *Mapping layer* and the *Mapped layer*. Intuitively, the faulty cells of the memory array in the mapping layer are "mapped" to the memory array in the mapped layer. In the implementation, the faulty rows/columns in the mapping layer have their address manipulated for an exchange of the faultless rows/columns in the

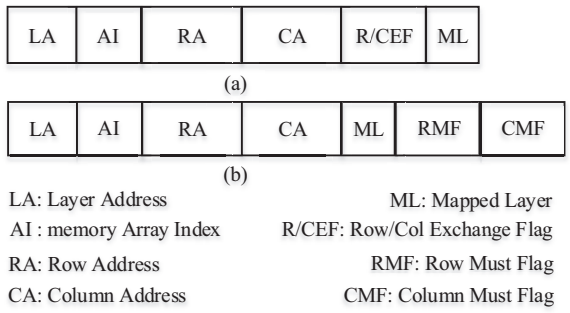

Fig. 5: CAM structure: (a) Address CAM; (b) Redundancy CAM.

mapped layer. For example, in Fig. 2(c), Layer 1 is the mapping layer and layer 2 and 3 are the mapped layer.

The address CAM manipulates the row/column address; its basic structure is shown in Fig. 5 (a). LA field embodies the memory layer index; AI field embodies the memory array index in the memory layer LA, which are used to locate the memory array. The four fields, i.e., row address (RA), column address (CA), LA and AI, belong to the mapping layer. While Mapped layer (ML) field determines the index of the mapped layer. The 1-bit Row/Col exchange flag (R/CEF) indicates the type of address, row or column, to be exchanged. If $R/CEF = 0$, the row indicated by the LA, AI and RA fields is exchanged with the row indicated by ML, AI and RA fields. To be specific, whenever an input address comes, if its first three fields (LA, AI, RA) matches those fields of an entry and the $R/CEF = 0$, the LA field in the input address is changed by the ML field. Similarly, if $R/CEF = 1$, the column indicated by the LA, AI and CA fields can be exchanged with the column indicated by ML, AI and CA fields. Note that, to reduce the hardware cost of address CAM, we restrain the exchange of row/column only happens between the addresses with the same row/column index (RA/CA) in the memory array with the same index (AI) across different layers (i.e., $LA \neq ML$). Consequently, the address CAM only has one set of AI, RA and CA fields; only 1 bit is required for R/CEF; and $\lg n$ bits are required for ML field, wherein n is the total number of memory layers.

The proposed fault clustering algorithm modifies the basic redundancy CAM as shown in Fig. 5 (b). The fields of LA, AI, RA, CA, ML are the same with those in address CAM. The Row Must Flag (RMF) indicates that the fault must be repaired by the redundant row ($1 = enable$). So does the Column Must Flag. It should be noted that, if the faulty cells from the mapping layer are clustered into the mapped layer through row (column) exchange, then the faulty cells in the mapped layer must be repaired by redundant column (row). The address CAM and redundancy CAM is cost effective. Use an 8-layer 4GB 3D memory as an example. Each layer has 64 memory arrays, each of which has 1K columns and 1K rows. Each entry of the address CAM has 3-bit LA, 6-bit AI, 10-bit RA and CA, 3-bit ML and 1-bit R/CEF, totally 33 bits. While each entry of redundancy CAM has 34 bits.

The fault clustering algorithm in RA block analyzes the redundancy CAM and configure the address CAM for address exchange. Given the mapping layer/mapped layer L_m/L_n, we use the row-based address exchange as an example to describe the fault clustering process as follows:

- *Step 1:* Search the entry E_e in the redundancy CAM with $LA = L_m$, find the faulty cell $Cell_m$ whose location is (R_i, C_j). If RMF is set as 1, go to step 4;

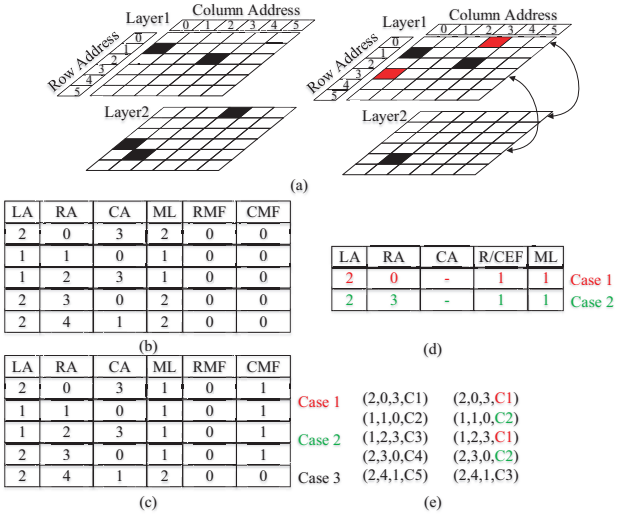

Fig. 6: (a) A conceptional example of fault clustering; (b) Status of the redundancy CAM before fault clustering; (c) Status of the redundancy CAM after after fault clustering; (d) Status of address CAM after fault clustering; (e) Potential spare allocation results: without fault clustering and with fault clustering.

- *Step 2:* Search the entry in the redundancy CAM with $ML = L_n$, find a faulty cell $Cell_n$ whose location is (R_k, C_j), $R_i \neq R_k$. If not found, go to step 4;
- *Step 3:* If $Cell_n$ found and the row R_i in L_n is faultless, the two rows R_i in L_m and L_n is exchanged. Add two entries in the address CAM and modify the entry in redundancy CAM by setting the column-must flag as 1).
- *Step 4:* If not finished, increase the entry index, $e++$, return to step 1; otherwise, stop.

In step 1, if the RMF in the found entry is set as 1, the RA has decided to repair this faulty cell with redundant row—multiple faulty cells exist in this row R_i—it is not legal to exchange this row. In step 2, if we cannot find a faulty cell $Cell_n$ in the mapped layer L_n with the same column index C_j (but different row index) with $Cell_m$, it is not beneficial to exchange: a redundant column in L_n repairs only a single faulty cell. Step 4 is to control the end of fault clustering algorithm: if all the entries in the redundancy CAM has been searched, the algorithm suspends.

Fig.6 represents an example of the fault clustering process. Layer 1 is the mapped layer $ML = 1$, while Layer 2 is the mapping layer $LA = 2$. The locations of faulty cells are stored in the redundancy CAM as shown in Fig. 6(b); the address CAM is empty. As case 1 (in red color), we first search the faulty cell in redundancy CAM with $LA = 2$, and find one with location (0,3). Then, we iteratively search for the faulty cell with $ML = 1$ and with column index equals to 3. A faulty cell in layer 1 with location (2,3) is found. At last, the row 0 in layer 1 is faultless, and thereby we can exchange the two rows with 0 index between these two layers, resulting in two new entries in the address CAM as shown in Fig 6 (d). The involved entries in the redundancy CAM is changed as shown in Fig. 6 (c). The fault clustering algorithm then finds the faulty cell (3,0) in layer 2 and exchange row 3 between these two layers. The changes in the address CAM and redundancy CAM is shown as case 2 in the figure. The fault clustering algorithm continues and finds the faulty cell (4,1) in the mapping layer. As no faulty cell in column 1 of the mapped

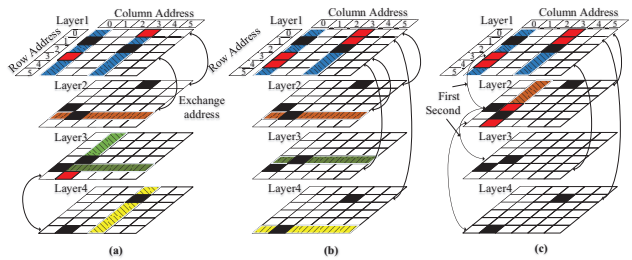

Fig. 7: Example of fault clustering techniques. (a) Pairwise mapping; (b) Multi-to-one mapping; (c) Cyclic mapping.

layer is found, no row is exchanged. Fig.6 (e) represents the spare allocation results for the above cases. Without fault clustering, five redundant columns are required. In contrast, with fault clustering, three spare columns are required to repair all the faults. Faulty cells (0,3) and (2,3) lie in different layer, but can be replaced by a single column (C1).

C. 3D Map-Global ESP

The proposed fault clustering technique can be applied in many repair algorithms. It disguises the input address through the address CAM, without interfering other modules in the BISR and memory controller. While any redundancy analysis algorithms can pretend that no exchange of row/column ever happens in the redundancy CAM and then repair the faults as normal. In this paper, we provide a 3D Map-GESP algorithm, by combining the fault clustering technique and the 3D GESP algorithm [13]. First, the 3D Map-GESP algorithm collects the location of faults derived from BIST module, and stores them in the redundancy CAM. Then, it executes the fault cluster algorithm and fills in the address CAM and changes the redundancy CAM. Consequently, it allocates the redundancy and repairs the faults based on the R(C)MF fields; and it executes the 3D GESP algorithm for the remaining faults in the redundancy CAM based on the remaining redundancy.

D. Optimization for Fault Clustering

Until now, we demonstrate the fault clustering technique between two adjacent layers. We denote it as *pairwise fault clustering*. Next, we optimize the fault clustering technique through judiciously strategies of choosing the mapping layers and mapped layers. We use a 4-layer 3D memory containing faults in Fig. 2 (a) as an example. For clarity, we only show one memory array in each layer. Without fault clustering technique, nine redundant columns are required. If we apply the pairwise fault clustering between layer 1 and 2, and between layer 3 and 4, six redundant columns are required as shown in the Fig. 7 (a). However, we need more efficient methods in the following scenarios: faults in multiple layers can be clustered into one layer for redundancy reduction.

In contrast to the *pairwise fault clustering*, *Multi-to-one fault clustering* strategy allows multiple mapping layers but only a single mapped layer. As shown in Fig. 7 (b), the faults in layer 2-4 can be clustered into layer 1. We then reduce the requirement of redundant columns from six to five. However, a considerable amount of faults are still remaining in layer 2-4 without clustering.

We propose *Cyclic fault clustering* to resolve the above problem. The basic idea is to iteratively execute *Multi-to-one fault clustering* strategy with a different mapped layer in each iteration. For a 3D memory with n layers, the *Multi-to-one fault clustering* process is executed in $n-1$ iterations. In the first iteration, the faults in the top $n-1$ layers are clustered into the top layer. In the second iteration, the faults in the top $n-2$ layers are clustered into the second layer. This process continues until the $n-1$ iterations are finished. Note that some rows or columns of a memory array may be exchanged multiple times. As shown in Fig. 7 (c), row 3 of layer 2 are exchanged twice: it is first exchanged with row 3 of layer 1, and then is exchanged with row 3 of layer 3. *Cyclic fault clustering* can further reduce the requirement of redundant columns to three. We note that the number of layers allowed for fault clustering determines the complexity and overhead of address CAM and redundancy CAM in the proposed 3D memory BISR architecture. We have to pay high cost for high repair ratio. With no doubts, *Cyclic mapping* leads to unacceptable cost when the number of layers in 3D stacked memory is very large.

To obtain a trade-off between repair rate and cost, we propose a *Group-based fault clustering* strategy. As an eclectic strategy, it divides memory layers into groups, each of which adopts the *Cyclic fault clustering* strategy. No fault clustering is allowed across different groups.

IV. Experiments

A. Experimental Setup

In order to evaluate the performance of proposed 3D BISR schemes, we build a 3D memory simulator. The sample 3D memory is composed of eight layers; each layer has 64 memory arrays, each of which contains 1024×1024 memory cells. We generate 1000 3D memory samples for Monte Carlo simulation. The simulator can generate fault maps by randomly injecting faults using Compound Poisson distribution [19], which well models the real distribution of faults. We evaluate the proposed Pairwise, Multi-to-one, Cyclic and Group-based fault clustering strategies, and compare them with the Original 3D GESP repair algorithm for the repair ratio. As the 3D GESP repair algorithm outperforms the remaining 3D repair solutions, no other repair methods are chosen as comparison. The yield of 3D memories and the hardware overhead of the proposed BISR architecture are also analyzed.

B. Results

In Fig. 8 (a), we evaluate the repair ratio of 3D memory with varying number of redundancies. The error rate is set as 1.5%. All the proposed fault clustering techniques obtain much higher repair ratio than that of Original solution. The *Cyclic fault clustering* technique outperforms the others. In average, the *Cyclic fault clustering*, the *Multi-to-one fault clustering* and the *pairwise fault clustering* techniques can approximately improve the repair ratio by 50%, 25% and 15% compared to the original scheme, respectively. More mapped layers involved in the solution, higher repair ratio can be obtained as more faults can be clustered together that improve the efficiency of redundancy. As the number of redundancy increases, the improvement on repair ratio decreases.

Fig. 8 (b) presents the repair ratio with varying error rate of 3D stacked memory. The number of redundancy is set as 40. The proposed fault clustering techniques can significantly improve the repair ratio. Their improvements are similar to what we observe in Fig. 8 (a). As the error rate increases, the repair ratio drops for all the methods. While for *Cyclic fault clustering* technique, the repair ratio drops smoothly. In average, the *Cyclic fault clustering*, the *Multi-to-one fault clustering* and the *pairwise fault clustering* techniques can approximately improve the repair ratio by 50%, 25% and 15% compared to the original scheme, respectively. In the highest error

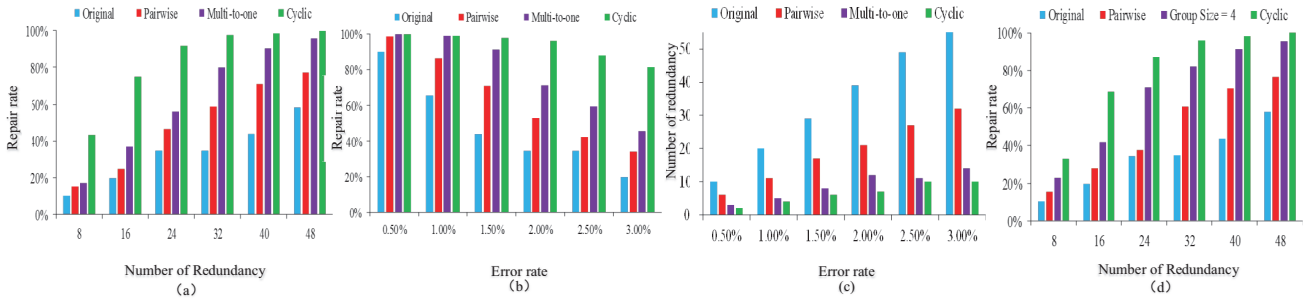

Fig. 8: Simulation results on (a) repair ratio varying the error rate; (b) repair ratio varying the error rate; (c) the number of redundancy; (d) repair ratio of *Group-based fault clustering* with different group size varying the redundancy.

rate, the repair ratio of *Cyclic fault clustering* technique is about 85%, which is about 60% better than the original solution.

Fig. 8 (c) shows the required redundancy with a target 99% yield varying the error rate. With the increasing error rate, the required redundancy increases for all the methods. However, the proposed techniques can efficiently save the requirement of redundancy. Surprisingly, whatever the error rate is, few redundancy is required for *Cyclic fault clustering* technique. In the highest error rate, the original solution needs more than 50 redundant rows/columns; on the contrary, only 10 redundancy is enough for *Cyclic fault clustering*. Five times of cost reduction can be achieved.

Finally, to evaluate the repair ratio of *Group-based fault clustering*, we compare the repair ratio of different group size varying the number of redundancy. The error rate is fixed as 1.5%. When the group size is two and eight, *Group-based fault clustering* reduces to the *Pairwise fault clustering* and *Cyclic fault clustering* techniques, respectively. As shown in Fig. 8 (d), for *Cyclic fault clustering* and *pairwise clustering*, the results are similar to what is shown in Fig. 8 (a). Interestingly, if we compare these two figures, we find that the *Group-based fault clustering* technique outperforms the *Multi-to-one fault clustering*.

C. Overhead analysis

The proposed BISR architecture incurs some timing penalty because additional time will be needed to search the address CAM and manipulate the address. Compared to the 3D BISR with global spare sharing [13], the additional hardware cost is list as follows. We don't need additional TSVs. On average, for each layer, We need to map for 4 times. For a 8-layer chip, 32 entries in the address CAM, which roughly equals to 1056 bit SRAM cells. We need 80 entries in the redundancy CAM; each entry has 2 more bits than that in [13]. The control logic for address manipulation around the address CAM can be ignored and the one around the redundancy CAM is similar to [13]. In total, the hardware overhead in the BISR is 0.53% larger than that in [13].

V. Conclusion

Three Dimensional (3D) memory has gained a great momentum because of its large storage capacity, bandwidth and etc. The yield has become one of the most critical challenges 3D memory design. We propose a novel BISR architecture for 3D memory to dramatically improve the repair ratio with ignorable timing penalty and hardware overhead. The experimental results show 71.82% higher repair rate than what can be achieved with the state-of-art 3D BISR technique in average.

References

[1] Kiran Puttaswamy et al. 3D-integrated SRAM components for high-performance microprocessors. *IEEE Transactions on Computers*, 58(10):1369–1381, 2009.

[2] G H Loh. 3D-Stacked Memory Architectures for Multi-core Processors. *ACM Sigarch Computer Architecture News*, 36(3):453–464, 2008.

[3] Uksong Kang et al. 8Gb 3-D DDR3 DRAM using through-silicon-via technology. *IEEE Journal of Solid-State Circuits*, 45(1):111–119, 2010.

[4] M Kawano et al. A 3D packaging technology for 4 Gbit stacked DRAM with 3 Gbps data transfer. In *2006 International Electron Devices Meeting*, pages 1–4. IEEE, 2006.

[5] Liu T et al. A 130.7-2-layer 32-gb reram memory device in 24-nm technology. *IEEE Journal of Solid-State Circuits*, 49(1):140–153, 2014.

[6] Sabry Aly M M et al. Energy-efficient abundant-data computing: The n3xt 1,000x. *Computer*, 48(12):24–33, 2015.

[7] Meng-Fan Chang et al. Challenges and trends in low-power 3D die-stacked IC designs using RAM, memristor logic, and resistive memory (ReRAM). In *ASIC (ASICON), 2011 IEEE 9th International Conference on*, pages 299–302. IEEE, 2011.

[8] Wooheon Kang et al. A 3 Dimensional Built-In Self-Repair Scheme for Yield Improvement of 3 Dimensional Memories. *IEEE Transactions on Reliability*, 64(2):586–595, 2015.

[9] Woosik Jeong et al. An advanced BIRA for memories with an optimal repair rate and fast analysis speed by using a branch analyzer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(12):2014–2026, 2010.

[10] Li Jiang et al. Yield enhancement for 3D-stacked memory by redundancy sharing across dies. In *Proceedings of the International Conference on Computer-Aided Design*, pages 230–234. IEEE Press, 2010.

[11] Che-Wei Chou et al. Yield-enhancement techniques for 3D random access memories. In *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*, pages 104–107. IEEE, 2010.

[12] Minsu Choi et al. Balanced redundancy utilization in embedded memory cores for dependable systems. In *Defect and Fault Tolerance in VLSI Systems, 2002. DFT 2002. Proceedings. 17th IEEE International Symposium on*, pages 419–427. IEEE, 2002.

[13] Xiaodong Wang et al. Global built-in self-repair for 3D memories with redundancy sharing and parallel testing. In *3D Systems Integration Conference (3DIC), 2011 IEEE International*, pages 1–8. IEEE, 2012.

[14] Y. H. Son et al. Cidra: A cache-inspired dram resilience architecture. In *IEEE International Symposium on High Performance Computer Architecture*, pages 502–513, 2015.

[15] P. J. Nair et al. Archshield: architectural framework for assisting DRAM scaling by tolerating high error rates. In *International Symposium on Computer Architecture*, pages 72–83, 2013.

[16] Swapnil Bahl. A sharable built-in self-repair for semiconductor memories with 2-D redundancy scheme. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 331–339. IEEE, 2007.

[17] Yung-Fa Chou et al. Memory repair by die stacking with through silicon vias. In *2009 IEEE International Workshop on Memory Technology, Design, and Testing*, pages 53–58. IEEE, 2009.

[18] Chun-Chuan Chi et al. 3D-IC BISR for stacked memories using cross-die spares. In *VLSI Design, Automation, and Test (VLSI-DAT), 2012 International Symposium on*, pages 1–4. IEEE, 2012.

[19] Israel Koren et al. Defect tolerance in VLSI circuits: techniques and yield analysis. *Proceedings of the IEEE*, 86(9):1819–1838, 1998.