Global Built-In Self-Repair for 3D Memories with Redundancy Sharing and Parallel Testing

Xiaodong Wang[1]
xw285@cornell.edu

Dilip Vasudevan[†]
dv2@cs.ucc.ie

Hsien-Hsin S. Lee
leehs@gatech.edu

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

[†] CEOL, Department of Computer Science
University College Cork
Cork, Ireland

ABSTRACT

3D integration is a promising technology that provides high memory bandwidth, reduced power, shortened latency, and smaller form factor. Among many issues in 3D IC design and production, testing remains one of the major challenges. This paper introduces a new design-for-test technique called 3D-GESP, an efficient Built-In-Self-Repair (BISR) algorithm to fulfill the test and reliability needs for 3D-stacked memories. Instead of the *local* testing and redundancy allocation method as most current BISR techniques employed, we introduce a *global* 3D BISR scheme, which not only enables redundancy sharing, but also parallelizes the BISR procedure among all the stacked layers of a 3D memory. Our simulation results show that our proposed technique will significantly increase the memory repair rate and reduce the test time.

1. INTRODUCTION

3D stacked IC is an emerging integration process. By stacking individual die vertically using face-to-face vias or through silicon vias (TSVs), it promises to provide benefits to improve interconnect latency, power, bandwidth, etc. Additionally, it results in a more compact form for the integrated system. More importantly, it continues to increase device density and their functionality for a given footprint to track Moore's Law without scaling down the devices.

Among many different 3D-stacked architecture alternatives, homogeneous 3D-stacked memory is becoming one of the first commercial 3-D IC products. Recently, Samsung announced to mass-produce stacked 40nm DDR3 DRAM using TSV. Other stacking architectures such as memory-on-logic, have also been studied or prototyped [2, 4, 10, 13, 20] to demonstrate the benefits brought by stacking memory tiers directly atop of the processing elements to improve performance (both latency and bandwidth) and power consumption.

Yielding will gradually become a critical issue for 3D memories as the number of layers stacked grows [12]. Built-in self-repair (BISR), a common technique to boost the yield of traditional 2D memories [7, 11, 16] should be appropriately applied to 3D memories. For 3D memories, techniques such as *through-silicon vias redundancy structure* to replace faulty TSV were recently prototyped [5, 9].

At first glance, it seems very straightforward that the BISR algorithm of 2D memories can be directly applied to 3D memories without any modification, because the way of accessing memory chips remain the same for 2D and 3D memories although the physical structure has been greatly changed. However, after examining the characteristics of 3D memories more carefully, we found the inefficiency in direct application and proposed a real global BISR algorithm and physical structure specifically tailored for 3D memories. We found that on average, the repair rate of our scheme is increased by 27.01% over the traditional local BISR scheme, and 8.26% over another global MESP BISR algorithm. In the meantime, the testing time can be reduced down to $\frac{1}{n}$ (n is the number of layers) of 2D memories given the same memory capacity.

[1]Xiaodong Wang is currently a graduate student at Cornell University. This research was performed when he was an exchange student at Georgia Tech.

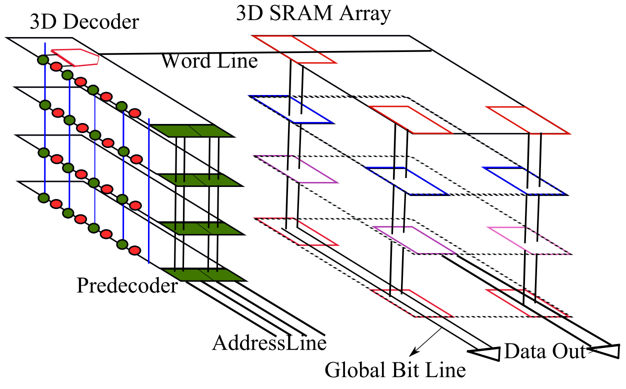


Figure 1: 3D SRAM Array Architecture

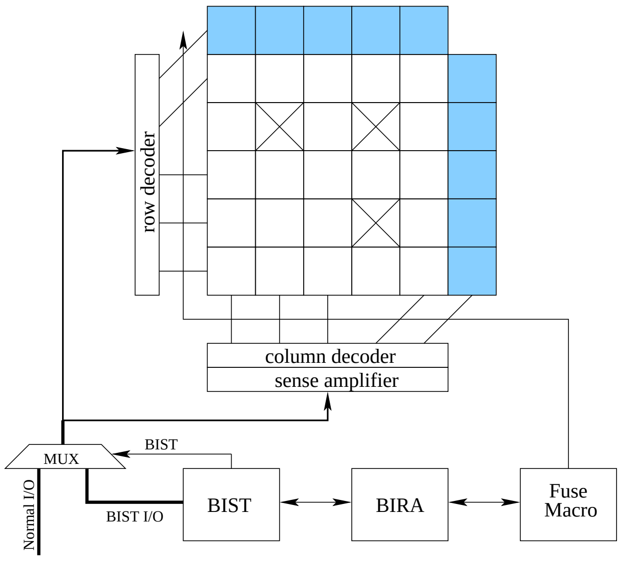
The rest of the paper is organized as follows. Section 2 reviews background. Section 3 motivates and introduces our global BISR for 3D memories. Section 4 proposed our 3D-GESP algorithm. Section 5 analyzes our simulation results. Section 6 concludes.

2. BACKGROUND

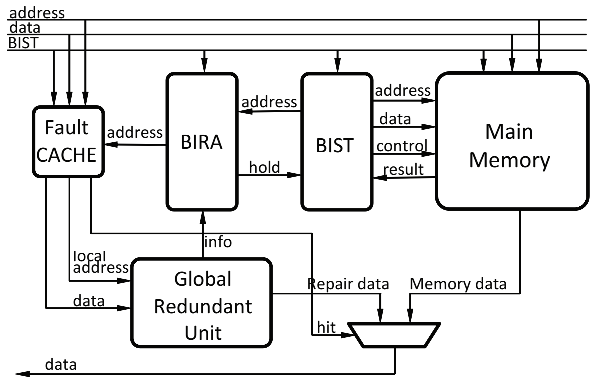
2.1 3D Memory Architecture

3D TSV-based memories are generally designed by stacking 2D planar memory layers with the address and data lines running across them vertically. The vertical connections of the address and data lines through the silicon are accomplished by using TSV. Thus the data storage spans across multiple die layers in contrast to the 2D design, where both the logic and memory are on the same plane.

A typical 3D memory architecture with vertical bitlines and 3D decoders was described in [15]. An abstract view of the architecture is illustrated in Figure 1. In which, several banks of SRAM are distributed vertically across different die layers. The original 2D decoder is also dissected and partitioned across the 3D layers. This example is one variant of possible 3D SRAM implementations. According to [15], their 128-entry multi-ported SRAM array can reduce energy and latency by 36% and 55%, respectively.



(a)



(b)

Figure 2: Two Typical BISR schemes (a) Decoder Redirection BISR, and (b) Fault Cache BISR

2.2 Typical BISR Architecture

The BISR technique requires several spare rows and columns manufactured as a part of the memory cells in order to replace the faulty cells in the array. In general, almost all the BISR design and optimization are based on two basic architectures: Decoder Redirection BISR [19], and Fault Cache BISR [18].

2.2.1 Decoder Redirection BISR

The block diagram of this conventional BISR architecture [19] is depicted in Figure 2 (a). It consists of four major parts.

- **Redundant Row/Column**. As shown in Figure 2(a), the faulty cells (marked as "X") are replaced by the redundant row and column (shaded squares) with necessary modifications on the decoder, in order to guarantee the correctness of memory operation.
- **Built-in Self-Test (BIST) circuit**. It generates test patterns, and then sends it to the memories. The march-like algorithm that gives high coverage, linear testing time, and simple hardware implementation, is the most widely used technique in memory testing.
- **Built-in Redundancy-Analysis (BIRA) circuit**. It collects the fault information from the BIST and then allocates the redundant units for the memory array.
- **Fuse Macro**. It stores the fault information and modifies the decoder to redirect the address from faulty cells to redundant units.

This architecture is easy to implement. However, it is inherently a *local* architecture because it is difficult for a decoder to redirect its local faulty cells to the neighboring redundant resources.

2.2.2 Fault Cache BISR

The block diagram of this BISR architecture [18] is depicted in Figure 2(b). It comprises the BIST, BIRA circuit, Fault Cache, and spare memories called Global Redundant Units (GRUs).

- **Global Redundant Unit**. Rather than manufactured together with the memory blocks, the GRU is fabricated separately. Nonetheless, its function to replacing the faulty cells remains the same.
- **Fault Cache**. The information of the BISR replacement is stored in it. During the normal operation, Fault Cache will determine whether the memory cell the system accesses has been replaced by the GRU. If so, it will generate a "hit" signal along with its local address to the GRU array, and choose the data from GRU for the data bus. When it is a cache miss, the data retrieved from the main memory will be used.

This "Fault Cache" BISR architecture does not reconfigure the decoder, so its redundancy resources can be global. However, this scheme requires extra components and more complicated wiring compared with the Decoder Redirection BISR.

3. GLOBAL BISR SCHEME

3.1 Motivation

Based on the two basic BISR architectures in Section 2, a number of optimizations have been proposed to improve the repair rate and the area overhead. Tseng *et al.* [19] proposed a ReBISR scheme for the RAMs in SOCs . In their scheme, multiple RAMs can share the same global BIRA circuits. However, the redundancy resources are not shareable — the neighboring memory blocks cannot share their redundancy with each other. Such *local* replacement may cause a problem. When the number of redundant units around a single block are insufficient, there will be repair failure, while some redundancy resources remain unused in other blocks, thus wasted. To solve this "local" problem, studies in [1, 21] proposed global replaceable redundancy schemes, allowing the use of the redundancy in one memory block to repair faults in others. However, these techniques not only require large area overheads, but also reduce the routability of the memory, and thus are less practical for traditional 2D memories.

Fortunately, one key feature of 3D IC is that the total length of interconnect can be reduced considerably via TSV. Therefore, for 3D memories, the routability problem of 2D memories with "global" redundancy can be resolved by intelligent 3D design. Toward this, Chou *et al.* [3] proposed a memory repair technique by stitching good parts of bad dies and stacking them together through TSV. However, their replacement is at the block level rather than at the row level as most BISR schemes do, so it will result in huge wastage. More recently, Jiang *et al.* [8] introduced a wise redundancy sharing technique across the dies for 3D memories. However, their scheme is pairwise only and not scalable for multi-layer 3D design. In addition, their proposal did not consider the test time, which is a critical issue for the cost and profitability.

3.2 Real Global 3D BISR Architecture

As discussed above, the current *local* BISR redundancy allocation cannot fully utilize the redundant resources on chip. This situation becomes even worse for 3-D memories. Since each memory

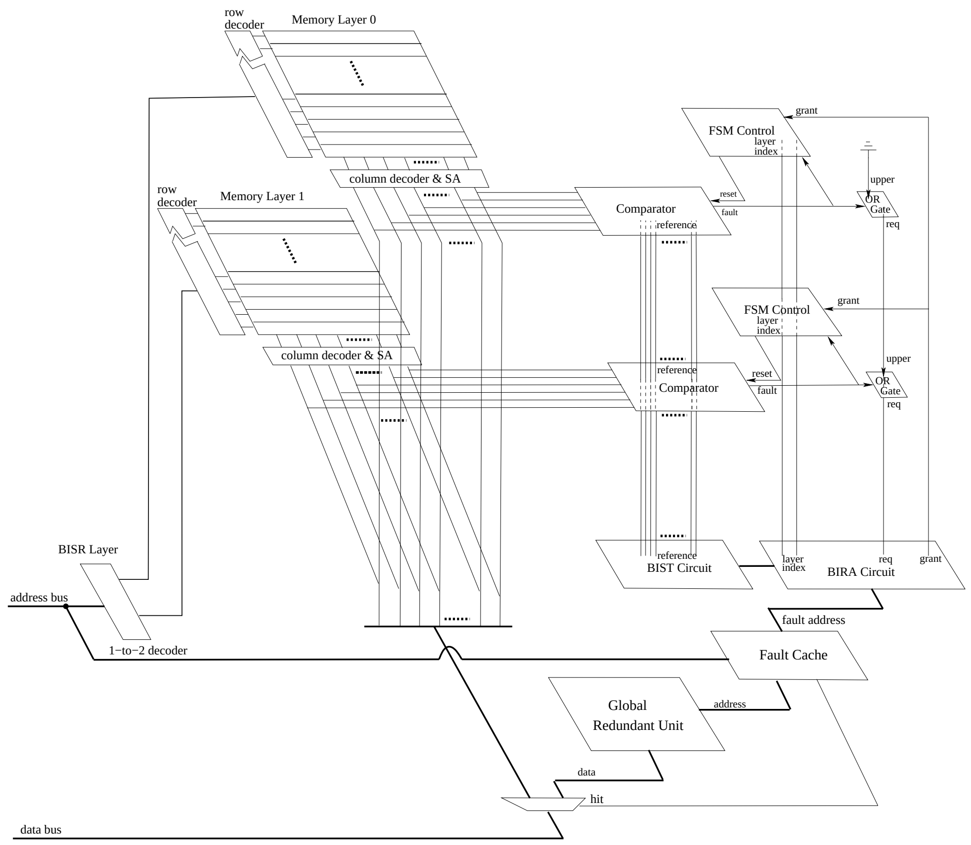


Figure 3: Schematic of our Global BISR Design

layer is produced separately, the number of faulty cells may vary for each layer. Therefore, besides the block-level wastage, the layer-level wastage will occur, where the redundant resources may be insufficient in certain layers while wasted in the other layers.

In addition, it may not be desirable to conduct BISR procedure serially among the memory layers. For 2D memories, it is difficult to parallelize the test among memory blocks, because a new datapath, which directly connects the BISR and every block, needs to be created. This complicates routing and incurs too much area overhead for a planar design. For 3D memories, the wiring constraint will be much alleviated by TSV with modest area overhead.

In this paper, *global* is defined according to the discussion above.

- **Shareable global redundancy**. The redundant resources can be shared by all the memory layers of a single 3D memory chip. In this way, the redundant resources can be fully utilized, and the overall yield will be increased.
- **Parallel testing**. Besides the yield issue, the global BISR should also provide parallel testing among memory blocks. The test time will be significantly reduced, so will the cost.

We choose the "Fault Cache BISR" architecture as the basis of our scheme for it can help realize a real global BISR design. As Figure 2(b) demonstrated, the Global Redundant Unit (GRU) can be used to repair the faulty cells of all the memory blocks, avoiding the restriction set by the decoder. On the other hand, its wiring (*i.e.*, routability of this architecture can be resolved by 3D technology through intelligent design. The detailed hardware and software BISR design will be discussed in Section 4.

4. 3D-GESP ALGORITHM

In this section, we introduce our global 3D BISR hardware design. The architecture is illustrated in Figure 3. To support this hardware layout for realizing the real *global* BISR scheme defined in Section 3, we also introduce a 3D-Global ESP (3D-Global Essential Spare Pivoting) algorithm. This algorithm is a combination of two algorithms we proposed: a Global ESP (GESP) and a 3D-BISR algorithm.

4.1 Global ESP Algorithm Extension

The GESP algorithm is specifically designed for the *shareable global redundancy*, corresponding to the first definition of *global* in Section 3. As Figure 3 shows, the redundancies (GRUs), Fault Cache, BIST, BIRA circuits, and all other auxiliary circuits, are placed at the bottom layer called the "BISR Layer", and are shared by all the memory layers.

The MESP scheme, a widely used algorithm in industry [14], can directly be applied to our hardware of Figure 3. However, the original scheme was specifically designed for traditional 2D memories. For our 3D BISR, we made several improvements mentioned below to further utilize the global characteristics and increase the repair rate.

Because the GRU architecture uses the Fault Cache to determine whether the main memory cell or GRU is the one that should be accessed, there is no need to set a boundary required by the architecture which modifies the decoder. This situation is shown in Figure 2. Thus we propose two more characteristics for achieving an efficient GESP algorithm.

1. We do not differentiate spare row or column as MESP did. One GRU entry can be used either as a spare row or column, according to the preference of the BIRA algorithm. This can avoid the situation where the BIRA needs additional one more spare row but all the spare rows have been used up. In that case, MESP

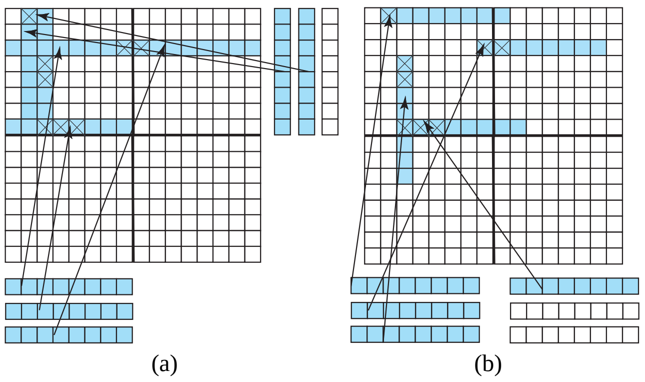


Figure 4: The Comparison between BIRA Algorithms. (a) Repairing of the MESP, and (b) Repairing of our GESP.

may have to use at least one or typically more spare columns to replace that 1-row repair. In our scheme, we can dynamically configure the spares as spare row or column, and thus, as long as there are some spares left, we can always use it as the BIRA's wish. This will further exploit the *global* characteristics of our scheme.

2. Unlike a conventional MESP the replacement must start at an aligned boundary (shown in Figure 4(a)) in a memory row or column, in our architecture, each GRU entry can point to any arbitrary location of a memory row or column for replacement as demonstrated in Figure 4(b).

   According to the figure, after having detected a new fault, which is not covered by any allocated GRUs, that fault will always serve as the "start point" of the new allocated GRU. Assuming this point's location is (x_i, y_i), and the coverage length of GRU is L. When the future faults are detected, BIRA will check whether it resides within the range of the previous GRU coverage, i.e., $[x_i + L, y_i]$ (row repair) or $[x_i, y_i + L]$ (column repair). On the contrary, for MESP, the "start point" of an GRU entry is always the boundary of the memory blocks, no matter where the faults are located.

As shown in Figure 4, for the same fault map, MESP requires five GRUs (three row entries plus two column entries) whereas our GESP requires only four. If clustered faults are present, in particular, crossing the alignment boundary in the memory array (eight squares in the figure), our GESP algorithm will provide more benefits and a higher repair rate. We will show our simulation results in Section 5.

Obviously, these two improvements can be applied to traditional 2D memories. However, it is only applicable to "Fault Cache BISR", which is described in Section 2.2.2. For the "Decoder Redirection BISR" scheme, row decoders will not have the ability to access spare columns, and the column decoders cannot access spare rows.

However, the "Decoder Redirection BISR" is more common in industry, because it is simpler and will incur less routability problem, as Figure 2 demonstrates. Moreover, even if the memory applies the "Fault Cache BISR", implementing our improvements will suffer from additional area overhead, which will be discussed in Section 5. However, this overhead problem can be hidden by our 3D memory structure, which will also be discussed in Section 5.

In the literaure, some other prior efforts were made over the MESP to achieve higher repair rates. For example, Huang *et al.* [6] proposed HYPERA to effectively increase the repair rate. But their design will incur severe timing penalty when accessing memories. Our GESP algorithm, however, does not suffer from the timing penalties because the main memory and the redundancies are accessed simultaneously, according to Figure 2(b). Some additional overhead will be required, but it's comparable with traditional MESP and can be hidden by our 3D memory layout. We'll analyze these overheads in Section 5 in details.

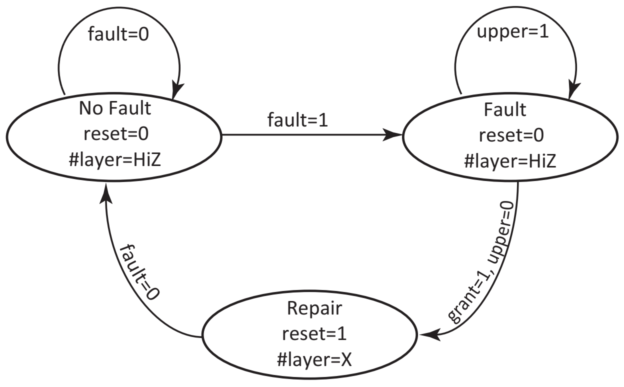


Figure 5: State Transition Diagram of the Control FSM

4.2 3D-BISR Algorithm

Besides the *shareable global redundancy*, our hardware design in Figure 3 can also enable parallel testing, corresponding to the second definition of *global* in Section 3. Combined with our 3D-BISR algorithm, the BISR procedure is parallelized for all memory layers. Therefore, no matter how many layers are stacked, the testing time will remain the same as if testing one-layer memory. The key point of this 3D parallel testing is to test all the memory layers simultaneously. At the system level, the 3D-BISR can be described as follows.

- Step 1: Perform BIST for one cell among all layers. The address allocator (1-to-2 decoder in Figure 3) will ignore the layer address (higher-order bits), sending the data and local address to every memory layer.
- Step 2: All memory layers determine whether any cell is faulty.
- Step 3: From Layer 1 to N, serially report to the bottom layer whether any tested cell of that layer is faulty.
- Step 4: Allocate GRU resources. Return to BIST.

This system-level scheme needs one OR gate, one comparator, and one FSM controller to be added in each layer as shown in Figure 3. Figure 5 shows the state transition diagram. Here we define the variables in the diagram.

- Fault: When the layer comparator detects that the content of the memory differs from the reference value provided by the BISR layer, it indicates a fault. The variable will be set to '1', otherwise '0'. It is one input of the OR gate as depicted in Figure 3.
- Upper: For a certain local location, when there is a fault in the upper layer, the "upper" signal is set to '1', another input of the OR gate.
- Request: The output of the OR gate. For a certain layer, when there are faults in that layer or in its upper layer, this signal will always be set to '1'.
- Reset: Clear the comparator when it is set to '1'. This will set the "Fault" signal to '0', and continue the 3D-BISR procedure.
- Grant: This is the signal sent by the BISR layer, telling the FSM of the layer which is in its "Fault" state to report the layer index if its "Upper" signal is '0'.
- #index: Reporting the layer index to the BISR layer when the FSM enters "Repair" state. For example, layer 1 will report "01", and layer 2 will report "10".

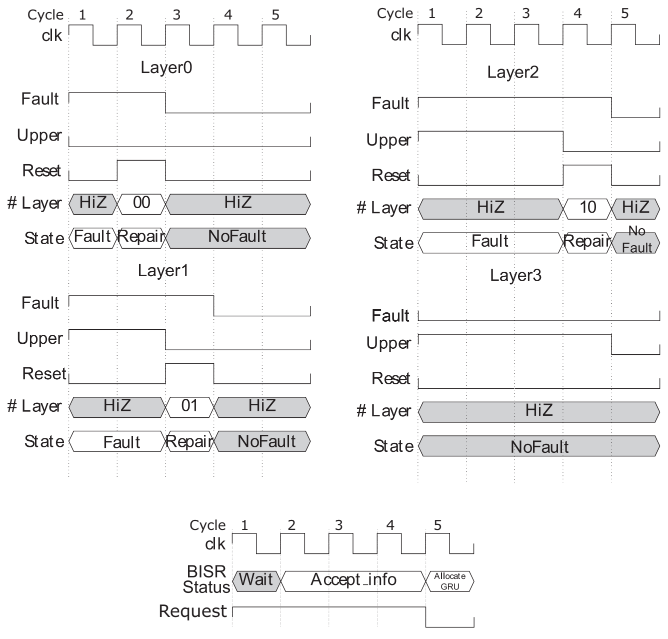


Figure 6: Timing Diagram of an Example for 3D-BISR

Whenever the BIST detects a faulty cell in certain layers, the "request" signal of the BIRA circuit will be set to "1" by the comparator and OR gates as shown in Figure 3. The BIST procedure will be stopped, and restored only after the fault information has been reported. For a specific layer, the 3D-BISR can be described as follows.

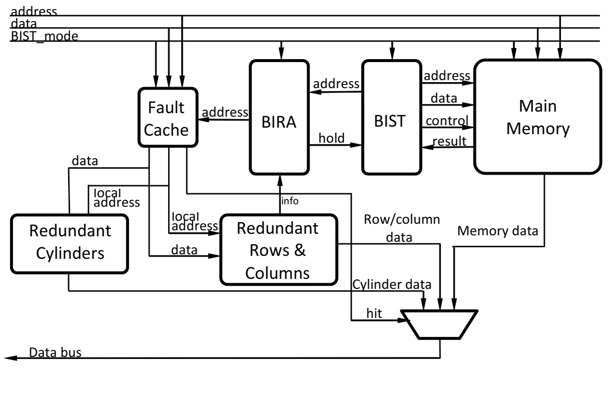
- Step 1: Perform BIST for the cell specified.
- Step 2: Determine whether the cell is faulty. If not, wait for the next cell's BIST. If yes, set "Fault" as '1' and go to step 3. Its FSM will enter "Fault" state.
- Step 3: If the layer's "Upper" equals '0', and "Grant" equals '1', go to step 4. The FSM will enter the "Repair" state. Otherwise, keep step 3.
- Step 4: Report its layer index through "#layer" signal. Set "Fault" signal to '0'. The "FSM" will enter "No Fault" state. Then, wait for the BIST of the following memory cells.

4.2.1 *Example for 3D-BISR algorithm*

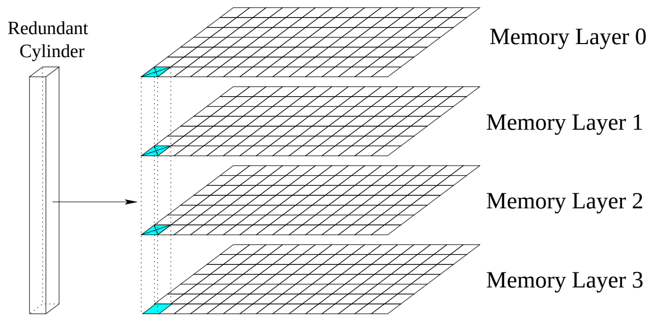
We show an example for detailing our algorithm. Suppose a four-layer memory stack where layer 0, 1, and 2 each contains a faulty cell at one exact location with a local row address 0x00 and local column address 0x00; that is, the memory locations: 0x00000, 0x10000, and 0x20000 are faulty. We assume layer 0 is the top layer. Firstly, the BIST begins testing 0x0000 of all four memory layers. Then, the comparators will determine whether the tested cell is faulty. After that, the status of our 3D-BISR related circuits will function according to the algorithm described in Section 4.2. Figure 6 shows the timing diagram which illustrates how our 3D-BISR scheme works cycle-by-cycle.

4.2.2 *New 3D redundancy structure*

As can be seen from the above example, the memory access and result comparison are done in parallel, while the reporting is done in a serial manner. This is because we want to save the quantity of the TSV and make our design scalable. Our design only needs $\log_2(N)$ reporting TSV shown in Figure 3, while at least N TSV are needed for the parallel reporting scheme. However, it is obvious that in the worst case when all the cells along the vertical axis are faulty, the test and the repair procedure will be completely serial.



(a)



(b)

Figure 7: 3DR Architecture: (a) Modified Block Diagram, (b) Detailed Replacement Scheme

To address this shortcoming, we propose a novel 3D redundant structure. The current 2D BISR algorithm uses the spare rows and columns for 2D replacement. For 3D memories, 3D redundancy structure can be developed intuitively. Besides the spare rows and columns, the spare *cylinder* structure is introduced in this section.

On the additional BISR layer, the 3D redundancy (3DR) unit — Redundant Cylinder is added as shown in Figure 7(a). Basically, our redundant cylinder has the same functionality as the redundant row/column does. However, instead of replacing the faulty row/column in a 2D manner, our redundant cylinder structure replace the faulty cells along the vertical axis, as demonstrated in Figure 7(b).

In order to support this cylinder replacement scheme, the fault cache should store the local row and column address of the faulty cylinder, and ignore the layer address. Whenever there is a address "hit" during normal operation, the system will access the redundant cylinder, rather than the faulty cells for read and write.

Now we'll describe how this Redundant Cylinder structure help to fully parallize the BISR procedure. When executing BIST, the same cells along the vertical axis of all the 3D memory layers will be tested simultaneously. If no more than one fault is detected and reported, the BISR procedure will remain the same as described previously. However, if multiple faults are deteceted, our BISR algorithm will not spend more cycles in accepting the fault information from the second and other faults. To be more specific, after the BISR logic receives the first fault information and send a "grant" signal, it finds out that the "request" signal is still '1', which means some other layer wants to report a fault. In this case, our scheme will not waste time in listening to any more fault information. Instead, it will allocate a *redundant cylinder* immediately, which replaces all the memory cells that have the same local location, just as shown in Figure 7(b). Therefore, the

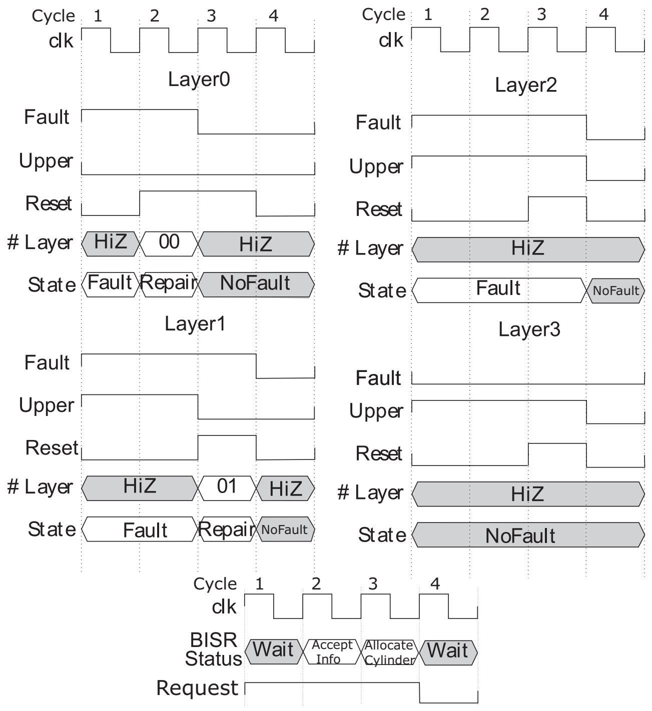


Figure 8: Timing Diagram of an Example for 3DR

maximum test time will be constrained to the upper limit of 2-layer 3D memories no matter how many layers the 3-D memory has.

4.2.3 *Example for 3DR allocation*

We show an example for our 3DR structure. Similar to the assumption in the previous example, layer 0, 1, and 2 each contains a faulty cell on their local address 0x0000. Figure 8 shows the timing diagram. The first two cycles remain the same as in Figure 6 .

Cycle 3: The "Request" input of BIRA is still '1', indicating that there is more than one fault along the vertical axis. Instead of accepting the second fault layer's index, it sends the "Reset" signal to all the memory layers (needs one additional shared TSV to transfer this "Reset" in Figure 3), and allocates a *spare cylinder* to this location as shown in Figure 7.

Cycle 4: All the comparators have been reset, so the "Request" input of BISR will go back to "0". Then the BISR circuit will deassert the "hold" signal to the BIST and continues testing. No more cycles are needed.

5. SIMULATION RESULTS AND ANALYSIS

5.1 Experiments

To evaluate the effect of our proposed algorithm, we use the clustered fault model in [17]. According to [17], the probability of not introducing an error into a given circuit area during a time interval Δt of the manufacturing process is:

$$p(\Delta t \mid k, l_1, l_2, \ldots, l_n) = c(x,y) + b \times k + d \times l(x,y)$$

Where $l(x,y)$ is the number of faulty neighboring memory cells around a memory location (x,y), $c(x,y)$ is the susceptibility parameter specified by the fabrication process, k is the number of faults already on that layer, b and d are the clustered factors.

Δt_0: It is the beginning of the fault generation. $l(x,y) = 0$ and $k = 0$. The probability of the memory cells to be faulty is $c(x,y)$ coherently across the layer. Then we generate an evenly distributed random value (0 to 1) for each memory cell. If the value is larger than its corresponding probability, then we determine that there is a fault on that location.

Δt_1: First we calculate the $l(x,y)$ and k after the first Δt_0. Then according to the probability equation, we still generate an evenly distributed random value (0 to 1) for each memory cell to determine whether there is a fault.

Δt_2—Δt_n: Repeat Δt_1.

There is no need to set an average number of faults for each layer, since this equation implicitly sets that value. Assume $c = 0.85, b = 0.04, d = -0.1$. The maximum $l(x,y)$ will be 8 (*i.e.*, all the neighboring cells are faulty). When the probability equation generates a value greater than 100%, there will be no faults generated. The relationship is shown below.

$$p(\Delta t_n \mid k, l_1, l_2, \ldots, l_n) = c(x,y) + b \times k + d \times l(x,y)$$
$$p = 0.85 + 0.04 \times k - 0.1 \times 8 < 100\%$$
$$k < \frac{1 - 0.85 + 0.1 \times 8}{0.04} = 23.75$$

Therefore, when $k > 23.75$, even if $l(x,y) = 8$, the probability will never be smaller than 100%, indicating that there will be no fault generated. Therefore, the upper limit of faults on one layer in this example is set to 24.

Here we also introduce the concept of *length of training intervals* (LTI). This fault generation procedure presented in [17] is just like training. Given enough number of *training intervals*, the number of faults on one layer will be very close to the *maximum faults*. Therefore, if we set the LTI long enough, the number of faults on one layer will be stable. In this way, we can better simulate the real manufacturing circumstances.

In the following analysis, each memory layer simulated has the size of $1024 \times 1024 \times 8bit$, with an average of 23.5340 faulty cells in each layer (implicitly set as discussed above). The parameters varied in the simulation are the following.

- GRU: The number of global redundancy units available for replacing the faulty row/column.
- Grid: The width of a row/column that a GRU replaces. For example, if $grid = 32$, the single GRU entry has 32×8 bits = 256 bits, which replaces 256-bit horizontally (row) or vertically (column) in the main memory.
- Cylinder: The number of *Cylinder* units are used during the whole BISR process.

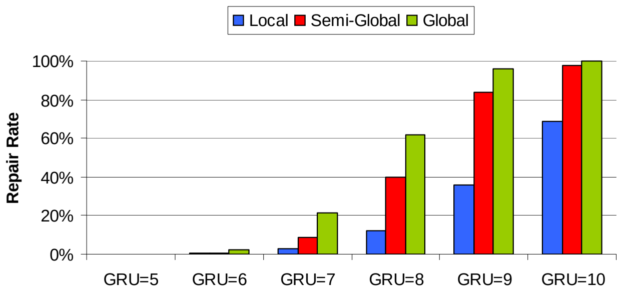


Figure 9: Comparison of Global and Local BISR Schemes

5.2 Performance of Proposed GESP Algorithm

First, we quantify the efficiency of our *shareable global redundancy* structure. This simulation is based on 1,000 samples of an eight-layer 3D memory. The grid size is chosen to be 128. The results are shown in Figure 9 where *Global* means that the eight memory layers share the GRUs; *Semi-global* is that every four memory

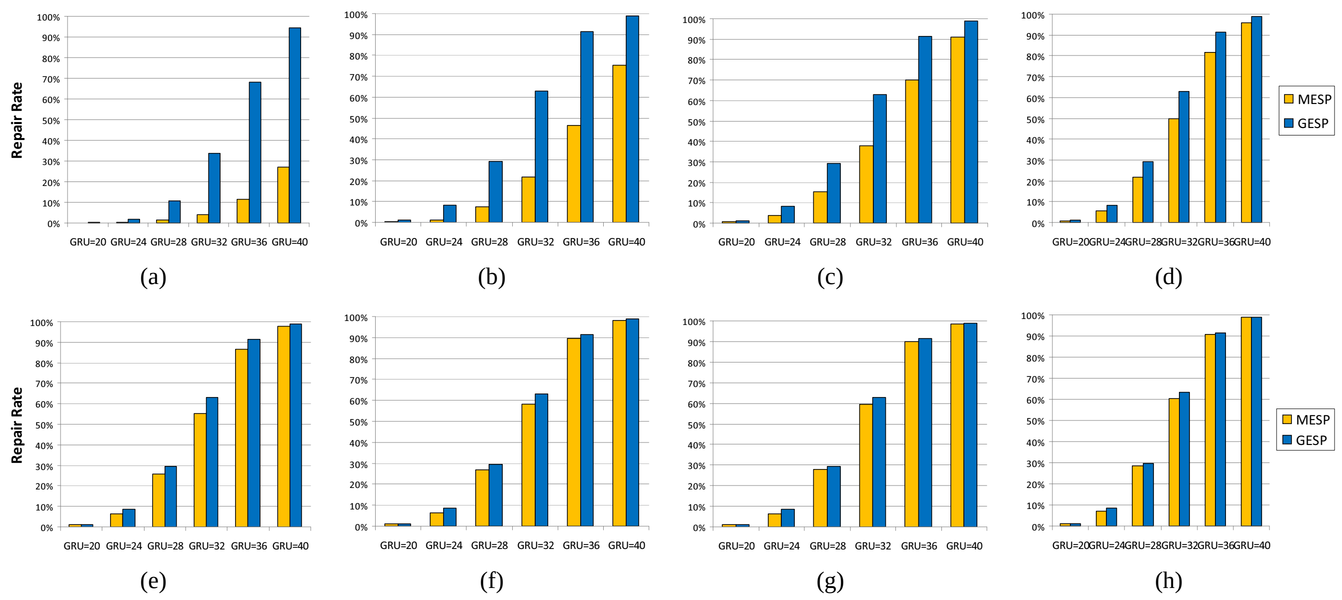


Figure 10: Comparison of BIRA algorithms. (a) grid=4. (b) grid=8. (c) grid=16. (d) grid=32. (e) grid=64. (f) grid=128. (g) grid=256. (h) grid=512.

layers share the GRUs, which are not shareable between groups; *Local* means every layer has its own GRUs, no sharing across layers. The total amount of the GRUs are the same for these three architectures. As shown, we clearly find that the *Global* scheme achieves 27% higher repair rate than the *Local* scheme on average (59.9% maximum improvement for GRU=8 on each layer). Compared with the *Semi-global* scheme, our *Global* scheme has 8.6% improvement for the repair rate, with a maximal improvement of 22.3%.

Secondly, we evaluate the performance gain of the “cross-boundary” technique in our proposed GESP algorithm over the MESP algorithm. Our simulation is based on 1000 samples of a four-layer memory. The selected grid sizes are 4, 8, 16, 32, 64, 128, 256, and 512. Figure 10 shows the results. As shown, we can see that our GESP has a much better performance than that of MESP when the grid is small, and that gap is closing with increased grid size. When the grid reaches half of the entire memory size, the improvement diminishes. This is not difficult to understand as we use the *clustered fault model* to evaluate the algorithms. When the grid is small, it is more likely for MESP to encounter the “cross-boundary” problem, wasting a lot of GRU entries for repair. For our proposed GESP algorithm, there is no such “cross-boundary” issue given our GRU could fix them at any arbitrary location as explained earlier in Figure 4. When the grid becomes larger, the likelihood of such “cross-boundary” problem will dwindle, thus the performance gap of these two algorithms is shrunk. Overall, our GESP algorithm outperforms the MESP algorithm by 8.26% on average (27.60% for maximum), with only a little hardware overhead, which will be described next.

5.3 Hardware overhead analysis

Performance-wise, our 3D-GESP algorithm does not incur timing penalty when accessing the memory because the memory and the redundancy resources are accessed concurrently. For area, however, our 3D-GESP scheme needs space for its BISR module. This overhead has two sources. Firstly, for each memory layer, we dedicate a comparator, an FSM, and an OR gate to support our global 3D BISR scheme. Given these are simple logic structure, the main contributor of the area overhead lie in the TSV. Using data in [13], the pitch of TSV to be 4 to 10 μm, and for 50nm process, the DRAM density is $27.9Mb/mm^2$. Assume a four-layer 3D memory, according to our 3D BISR design in Figure 3, the comparator needs 8 TSVs, the FSM needs $(\log_2 4)$ TSVs, the OR gate need 1 TSV, and the “Cylinder Repari” needs 1 TSV for “request” and 1 TSV for “grant”. In total, each layer requires 13 TSVs, occupying $1300\mu m^2$ with a $10\mu m$ TSV-pitch. For $1MB$ main memory each layer, it takes approximately $285,714\mu m^2$. The area overhead is merely 0.455%.

In a more general case, assuming the 3D memory has a layers, each layer has b cells, each cell contains c bits. The total number of TSVs that require by our scheme is: $\log_2 a + c + 3$. Therefore:

$$total\ area\ overhead = \frac{3488 \times (\log_2 a + c + 3)}{b \times c}$$

According to most 3D memory configurations, the area overhead will be less than 1%.

Secondly, for the BIST and BISR circuits, the area overhead for the MESP scheme was 8.7% as shown in [14]. For our 3D-GESP scheme, which exploits *cross-boundary* repair, may need more area. For MESP, the Fault Cache only needs to store partial address of the faulty cells because of the existence of the *aligned boundary*. For example, only 15 bits is needed in the Fault Cache for a four-layer 3-D memory with a grid size of 128. For our 3D-GESP scheme, all the address bits need to be stored, which is 31.8% more. Except that, no other area overhead is needed for our scheme. In the worst case, our hardware overhead will amount to 11.5%. However, in our 3D BISR design as depicted in Figure 3, we dedicate an entire layer to BISR. In this way, the hardware overhead for BISR modules will not be a problem for implementing 3D-GESP. The real area overhead on the memory layers is 0.39%.

5.4 Implication of 3D Memories

Finally, during simulation, we made an interesting observation. As shown in Figure 11, given 1-, 4-, and 8-layer 3D memories with 4, 5, 6, 7, 8, 9, and 10 GRU per layer to guarantee the hardware overhead ratio is the same, the 8-layer memory has the smallest repair rate among others when the number of the GRU is small; on the contrary the 8-layer memory has the highest repair rate when the number of the GRU is sufficient. Here the overhead ratio is defined as:

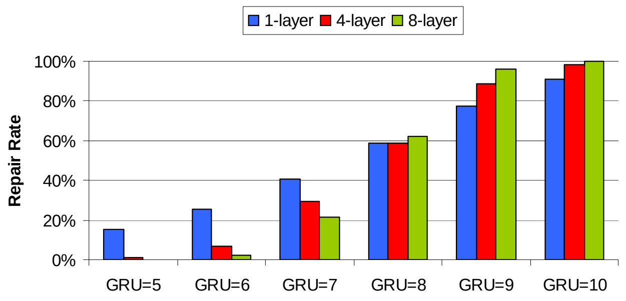


Figure 11: Layer's effect on the repair rate given the same GRU/layer

$$Overhead = \frac{GRU\ resources\ provided}{total\ cells\ of\ 3D\ memory\ product}$$

This is a characteristics of 3D memory. We assume that all the memory layers are produced separately and have no correlation in fault map, the average number of faults for each layer is m, and the standard deviation is σ, same to those of one traditional 2D memory. For an n-layer 3D memory, the average faults for each layer is m and the standard deviation is $\frac{\sigma}{\sqrt{n}}$. Since the standard deviation of a 3D memory is smaller than that of a 2D memory, the average number of faults of each 3D memory layer will be converged into the range of $m \pm p$. To simplify the problem, we use the following example by assuming the number of GRU can only repair no more than 16 faults of each layer, and the average number of faults $m = 20$. Since the standard deviation is smaller for 3D memories, the possibility for a certain 3D memory product to have faults less than 16 for each layer is assumed to be 10%, while that of 2D memory is 20%. Therefore, 2D memory will result in a higher repair rate. On the other hand, when the number of GRU can repair no more than 24 faults of each layer, the possibility for a certain 3D memory product that cannot be repaired is 10%, while that of 2D memory is 20%. The 2D memory will suffer from a lower repair rate under that circumstance.

Since it is always desirable to have more than 90% repair rate, 3D memories will show their advantage over 2D. As shown in Figure 11, with above 90% repair rate, the 3D memories need fewer GRU per layer than their 2D counterpart.

6. CONCLUSION

In this paper, we present a novel hardware and software design for 3-D BISR. Our scheme, 3D-GESP, is a real *Global* BISR technique, which enables the global redundancy sharing and parallel testing. The experimental results showed that our 3D-GESP scheme can achieve 27.01% higher repair rate compared to the local BISR, and 8.26% over another global algorithm MESP. In addition, our scheme only requires $\frac{1}{n}$ testing time compared with the traditional BISR procedure, where n is the number of stacked layers of 3D memories. Therefore, our scheme will significantly improve the manufacturing yield, repair rate, and testing throughput of 3D die-stacked memories.

7. ACKNOWLEDGMENTS

This work was sponsored by an NSF grant CCF-0811738, ITRI Taiwan, and gift from Intel Corp. Xiaodong Wang conducted this work while he was an exchange student at Georgia Tech from Shanghai Jiao Tong University. Dilip Vasudevan was a visiting scholar at Georgia Tech sponsored by the University of College Cork at Ireland.

8. REFERENCES

[1] S. 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*, pages 331–339, 2007.

[2] B. Black and et al. Die Stacking (3D) Microarchitecture. In *Proc. of the International Symposium on Microarchitecture*, 2006.

[3] Y.-F. Chou, D.-M. Kwai, and C.-W. Wu. Memory Repair by Die Stacking with Through Silicon Vias. In *IEEE International Workshop on Memory Technology, Design, and Testing*, pages 53–58, 2009.

[4] M. Healy, K. Athikulwongse, R. Goel, M. Hossain, D. Kim, Y.-J. Lee, D. Lewis, T.-W. Lin, C. Liu, M. Jung, B. Ouellette, M. Pathak, H. Sane, G. Shen, D. H. Woo, X. Zhao, G. Loh, H.-H. S. Lee, and S. K. Lim. Design and Analysis of 3D-MAPS: A Many-Core 3D Processor with Stacked Memory. In *IEEE Custom Integrated Circuits Conference*, 2010.

[5] A.-C. Hsieh, T.-T. Hwang, M.-T. Chang, M.-H. Tsai, C.-M. Tseng, and H.-C. Li. TSV Redundancy: Architecture and Design Issues in 3D IC. In *Proceedings of the Conference on Design Automation and Test in Europe*, pages 166–171, 2010.

[6] T.-C. Huang and et al. HYPERA: High-Yield Performance-Efficient Redundancy Analysis. In *19th IEEE Asian Test Symposium (ATS)*, pages 231–236, 2010.

[7] W. K. Huang, Y.-N. Shen, and F. Lombardi. New approaches for the repairs of memories with redundancy by row/column deletion for yield enhancement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(3), March 1990.

[8] L. Jiang, R. Ye, and Q. Xu. Yield Enhancement for 3D-Stacked Memory by Redundancy Sharing across Dies. In *Proc. of International Conference on Computer-Aided Design*, 2010.

[9] U. Kang and et al. 8 Gb 3-D DDR3 DRAM Using Through-Silicon Via Techndology. *IEEE Journal of Solid-State Circuits*, 45(1):111–119, 2010.

[10] D. H. Kim, K. Athikulwongse, M. B. Healy, M. M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. H. Loh, H.-H. S. Lee, and S. K. Lim. 3D-MAPS: 3D Massively Parallel Processor with Stacked Memory. In *IEEE International Solid-State Circuits Conference (ISSCC)*, 2012.

[11] I. Kim and et al. Built-in self-repair for embedded high density sram. In *IEEE International Test Conference*, pages 1112–1119, 1998.

[12] H.-H. S. Lee and K. Chakrabarty. Test Challenges for 3D Integrated Circuits. *IEEE Design and Test of Computers, Special Issue on 3D IC Design and Test*, 26(5):26–35, Sep/Oct 2009.

[13] G. H. Loh. 3D-Stacked Memory Architectures for Multi-Core Processors. In *Proceedings of the International Symposium on Computer Architecture*, pages 453–464, 2008.

[14] S.-K. Lu and et al. Efficient BISR Techniques for Embedded Memories Considering Cluster Faults. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(2), February 2010.

[15] K. Puttaswamy and G. Loh. 3D-Integrated SRAM Components for High-Performance Microprocessors. *IEEE Transactions on Computers*, 58(10):1369 –1381, 2009.

[16] R. Rajsuman. Design and test of large embedded memories: An overview. *IEEE Design and Test of Computers*, 18(3), May 2001.

[17] C. H. Stapper. Simulation of spatial fault distributions for integrated circuit yield estimations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(12), 1989.

[18] A. Tanabe and et al. A 30-ns 64-Mb DRAM with Built-in Self-Test and Self-Repair Function. *IEEE Journal of Solid-State Circuits*, 27(11):1525–1533, 1992.

[19] T.-W. Tseng, J.-F. Li, and C.-C. Hsu. ReBISR: A Reconfigurable Built-In Self-Repair Scheme for Random Access Memories in SOCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(6), 2010.

[20] D. H. Woo, N. H. Seong, D. L. Lewis, and H.-H. S. Lee. An optimized 3d-stacked memory architecture by exploring excessive, high-density tsv bandwidth. In *Proceedings of the 16th International Symposium on High-Performance Computer Architecture*, pages 429–440, 2010.

[21] T. Yamagata and et al. A Distributed Globally Replaceable Redundancy Scheme for Sub-Half-Micron ULSI Memories and Beyond. *IEEE Journal of Solid-State Circuits*, 31(2):195–201, 1996.