Volatile Memories In System On Chip (SOC) With Built-In-Self -Modification Scheme

P. Kumar, H. Devanna and K. Sudhakar

Abstract— Built-in self- modification (BISM) technique has been widely used to modify embedded random access memories (RAMs). This paper presents a built-in-self -modification (BISM) scheme for repairing RAMs with different sizes and redundancy organizations An efficient redundancy analysis algorithm is proposed to allocate redundancies of defective RAMs. In the BISM, are configurable built-in redundancy analysis (BIRA) circuit is designed to perform the redundancy algorithm for various RAMs. Also, an adaptively reconfigurable fusing methodology is proposed to reduce their pair setup time when the RAMs are operated in normal mode. Experimental results show that the BISM scheme can achieve high modified rate (i.e., the ratio of the number of repaired RAMs to the number of defective RAMs). The area cost of the BISM is very small, which is only about 2.7% for four RAMs (one 4 Kbit RAM, one 16 Kbit RAM, one 128 Kbit RAM, and one 512 Kbit RAM). Moreover, the time overhead of redundancy analysis is very small. For example, the ratio of the redundancy analysis time to the test time for a 512 Kbit RAM tested by a March-14 N test with solid data backgrounds is only about 0.25%. On the other hand, the proposed fusing scheme can achieve about 86.94% reduction of repair setup time in comparison with a typical fusing scheme for 20 512 X 16 X 64-bit RAMs of which each RAM has one spare row and one spare column. The contrast with a microcontroller is one of degree. Microcontrollers typically have under 100 kB of RAM (often just a few kilobytes) and often really are single-chip-systems, whereas the term SoC is typically used with more powerful processors, capable of running software.

Keywords: System on chip (SOC), Built-in redundancy analysis (BIRA), built-in self test (BIST), built-in self-modification (BISM), March test, volatile memory (RAM), yield improvement.

I. INTRODUCTION

EMBEDDED random access memory (RAM) is one key component in modern complex system-on-chip (SOC) designs. Typically, many RAMs with various sizes are included in an SOC, and they occupy a significant portion of the chip area. Furthermore, RAMs are subject to aggressive design rules, such that they are more prone to manufacturing defects. That is, RAMs have more serious problems of yield and reliability than any other embedded cores in an SOC.

P.Kumar, electronics and communication engineering (M.Tech, VLSI System Design), H. Devanna, Assoc. Professor, M.Tech (PhD), AMIE, SJCET, Yemmiganur and K. Sudhakar, Assoc. Professor, M.Tech (PhD), AMIE, SJCET, Yemmiganur, prabhusampathkumarvarma@gmail.com, contact *no:9951895151*

Keeping the RAM cores at a reasonable yield level is thus vital for SOC products. Built-in self-repair (BISR) technique has been shown to improve the RAM yield efficiently. For example, the work in [1] shows that the BISR circuit can improve the RAM yield from 5% to 20%, such that the net SOC yield increase can range from 2% to 10%. Therefore, the BISR technique is a promising and popular solution for RAM yield improvement [2]. Many BISR schemes for RAMs have been proposed in[2]–[20]. Built-in redundancy-analysis (BIRA) algorithm is one key component of a BISR scheme, and it is responsible for allocating redundancies of RAMs under test. Thus, the BIRA circuit has heavily influence on the repair efficiency of the BISR scheme. If a RAM has only spare rows or spare columns, i.e., 1-D redundancy, then the redundancy allocation is simple and straightforward. For example, BISR schemes proposed in [3], [4], [10], and [11] are used for repairing RAMs with 1-D redundancy.

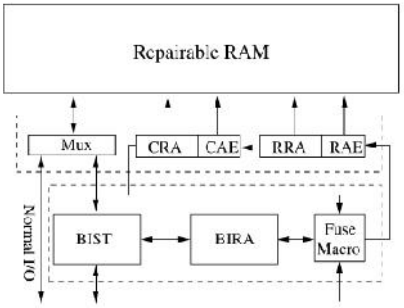
On the other hand, if a RAM has spare rows/spare columns or spare rows/spare IOs, i.e., 2-D redundancy, the redundancy allocation becomes an NP-complete problem [21]. Most of existing BISR schemes deal with the repair of RAMs with 2-D redundancy, e.g., [2], [5]–[9], [12], [13], [15]–[19]. To achieve the best repair efficiency, BISR schemes reported in [7], [12],and [17] use exhaustive search algorithms to allocate 2-D redundancy. But, either the area cost or the test/repair time of these schemes is very high. In [7] and [12], the authors use parallel BIRA modules to allocate redundancy optimally. However, the number of BIRA modules is drastically increased with respect to the number of redundancies of a RAM under test. This result in very high area cost. In [17], Holler *et al.* use a single BIRA module to allocate redundancy optimally. The BIRA module is programmable and one repair strategy is programmed to allocate redundancies each time. Therefore, if the programmed repair strategy cannot repair the memory under test, then another repair strategy is programmed in the BIRA module and the memory under test is retested. This process is repeatedly until a successful repair strategy is found or all possible repair strategies are tried. Although the area cost of the BIRA circuit is reduced, this results in very high time cost of test and repair. Therefore, some of existing BISR schemes use heuristic redundancy analysis algorithms to allocate the redundancy of RAMs under test. These heuristic analysis algorithms attempt to optimize the repair efficiency and minimize the area cost and the time of test/repair. For example, the BIRA module of the BISR schemes reported in [5], [6], [13], [18], and [19] performs heuristic redundancy analysis algorithms to allocate the redundancy. Due to the importance of redundancy analysis algorithms, furthermore, some research works focus on the development of efficient heuristic redundancy analysis algorithms[15], [22]–[24]. To boost the reliability of a RAM

in life time, some BISR schemes integrating online repair functions have been proposed in [14] and [20].However, RAMs in an SOC usually have various sizes, different numbers of redundancies, and even different types of redundancy organizations. If each repairable RAM uses one self-contained BISM circuit, then the area cost of BISR circuits in an SOC becomes high. This results in converse effect in the yield of RAMs. To reduce the area cost, several processor-based BISM schemes have been reported in [8], [9], [18], and [25]. In these BISR schemes, a BISM circuit can repair multiple RAMs. However, a processor-based BISR scheme uses specific instructions to construct the redundancy analysis algorithm [8], [18],[25]. This results in long redundancy analysis time, since a statement of the redundancy analysis algorithm may need multiple instructions to realize it. Therefore, a time-efficient and area-efficient BISM scheme is needed to improve the yield of RAMs in SOCs economically. After the BISR circuit completes the test/repair process of a defective RAM, an electrically programmable fuse macro enables the on-chip self-reconfiguration process for skipping defective elements of the RAM operated in normal mode. Various implementations of fuse macros have been proposed in [4],[26]–[28]. Typically, multiple RAMs of an SOC share a fuse macro to store the repair signatures of the RAMs. Once the power of the SOC is turned on, the repair signatures are serially shifted to the individual repair register of each RAM. The loading time of the repair signatures is called the repair setup time of the RAMs [15]. The repair setup time determines how long the RAMs in an SOC can be switched to the normal operation Mode after the power of the SOC is turned on. Thus, reducing the repair setup time can shorten the boost time of the system. This paper presents a reconfigurable BISM (ReBISM) scheme. The ReBISM can be shared by multiple RAMs with different sizes and redundancy organizations. This can reduce the area cost of the BISR circuits in an SOC. Also, an efficient reconfigurable BIRA (ReBIRA) scheme is used to allocate 2-D redundancies of multiple RAMs [15]. The ReBIRA provides very good repair rate (the ratio of the number of repaired RAMs to the number of defective RAMs [22]). Moreover, a reconfigurable fusing methodology is proposed to reduce the repair setup time of RAMs in an SOC. In comparison with a typical fusing methodology (repair signatures in the fuse macro are shifted into the repair registers in which a repair register is connected to another repair register in daisy chain), the proposed scheme can drastically reduce the repair setup time. The rest of this paper is organized as follows. Section II reviews typical BISR architecture for RAMs. Section III introduces the proposed ReBISM scheme, redundancy-analysis algorithm, and ReBIRA design. Section IV describes the proposed adaptively reconfigurable fusing methodology. Section V summarizes experimental results. Section VI concludes this paper.

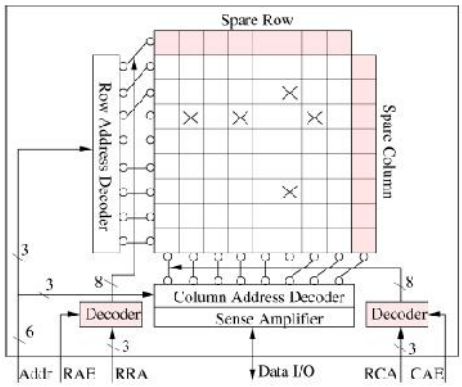
II. TYPICAL MEMORY BISM ARCHITECTURE

Fig. 1 shows the block diagram of a typical BISM scheme for a RAM, which consists of four major components.1) *Repairable RAM*. A RAM with redundancies and reconfiguration circuit.

Fig. 2 depicts an example of an 8 8bit-oriented RAM with 1 spare row and 1 spare column. If a spare row is allocated to replace a defective row, then
the row address of the defective row is called row repair address(RRA). Then a decoder decodes the RRA into control signals for switching row multiplexers to skip the defective row if the row address enable (RAE) signal is asserted. The reconfiguration of the defective column and the spare column is performed in a similar way, i.e., give a column repair addresses (CRA) and assert the column address enable signal to repair the defective column using the spare column.

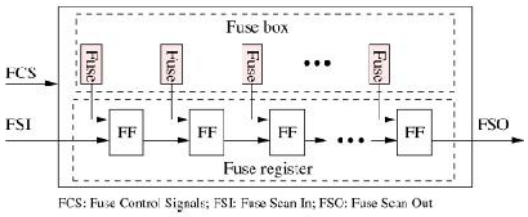


. Typical BISR scheme for embedded RAMs.



. Conceptual diagram of an 8 × 8 bit-oriented repairable RAM with one row and one spare column.

2) *Built-in Self-Test (BIST) Circuit*. It can generate test patterns for RAMs under test. While a fault in a defective RAM is detected by the BIST circuit, the faulty information is sent to the BIRA circuit.
3) *BIRA Circuit*. It collects the faulty information sent from the BIST circuit and allocates redundancies according to the collected faulty information using the implemented redundancy analysis algorithm.
4) *Fuse Macro*. It stores repair signatures of RAMs under test. Fig. 3 shows the conceptual block diagram of a typical implementation of fuse macro [4]. The fuses of the fuse box can be implemented in different technologies, e.g., laser blown fuses, electronic-programmable fuses, etc. The fuse register is the transportation interface between the fuse box and the repair register in the repairable RAM. As Fig. 1 shows, the overall RAM BISR flow is roughly described as follows. Firstly, the BIST tests the repairable RAM.



FCS: Fuse Control Signals; FSI: Fuse Scan In; FSO: Fuse Scan Out

g. 3. Conceptual block diagram of a fuse macro [4].

If a fault is detected, then the fault information is stored in the BIRA circuit. Then, the BIRA circuit allocates redundancies to replace effective elements. As soon as the repair process is completed, the repair signatures are blown in the fuse box. Subsequently, the repair signatures are loaded into the fuse register first and then are shifted to the repair registers (i.e., registers in
the wrappers for storing RRA, RAE, CRA, and CAE data) in normal operation mode. Finally, the repairable RAM can be operated correctly.

III. PROPOSED MEMORY REBISM SCHEME

As Fig. 1 shows, if each RAM in an SOC has individual BISR circuit, then the total area of BISR circuits for the RAMs in the SOC is large, since an SOC usually has many RAMs. To reduce the area cost of BISM circuits, we propose a ReBISM scheme for RAMs in an SOC. A ReBISM circuit can be shared by multiple RAMs such that the total area cost of BISR circuits in an SOC can be drastically reduced.

A. Architecture of the ReBISM Scheme

Fig. 4 shows the simplified block diagram of the proposed ReBISM scheme for repairing multiple repairable RAMs in an SOC, where detail control signals for the ReBISM circuit is not shown. The Wrapper of a RAM under test consists of multiplexers, a test pattern generator (TPG), and repair registers. The multiplexers switch the RAM between test/repair mode and normal mode. The TPG generates desired test patterns according to the given command from the test controller (CTR). The repair registers store the repair signatures. The CTR coordinates the operations of the TPG and the ReBIRA circuit. The Fuse Macro consists of the fuse and the fuse register. The number of bits of the fuse, the fuse registers, and the repair register is the same. Different technologies can be used to implement the fuse, e.g., the laser-blown fuse, the programmable electronic fuse, etc. If the laser-blown fuse is used, then the ReBISM only can repair the target RAMs one time. If the programmable electronic fuse is used, then the Re- BISR can perform RAM repair multiple times. In our ReBISM scheme, moreover, the repairable RAMs can be equipped with one of the following two redundancy organizations: 1) spare rows and spare columns and 2) spare rows and spare IOs. As Fig. 4 shows, the CTR and ReBIRA circuits are shared by multiple RAMs. Therefore, the area cost of the ReBISM circuit is reduced. Fig. 5(a) shows the repair process of the proposed ReBISM scheme during test and repair phase. If the BIST detects a fault,

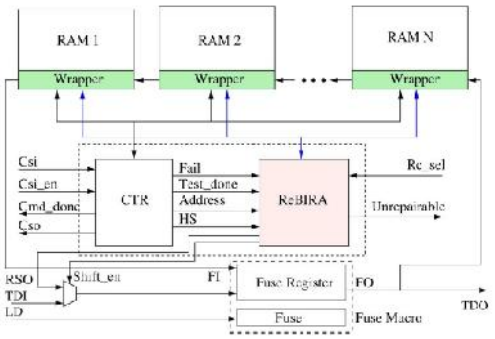
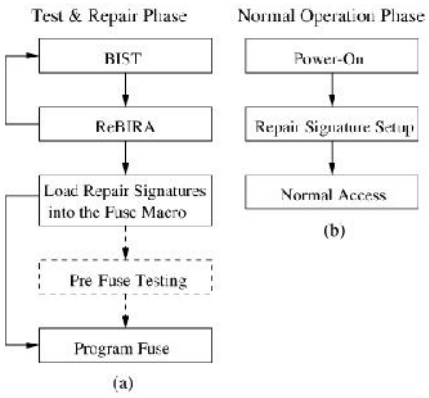


Fig. 4. Simplified block diagram of the proposed ReBISR scheme for multiple RAMs.

then the fault information is exported to the ReBIRA circuitry, and then the ReBIRA performs redundancy allocation on the fly using the rules of the implemented redundancy algorithm (the proposed redundancy algorithm will be described in the next subsection). The ReBIRA allocating redundancy on the fly means that the redundancy allocation process and the BIST process are performed concurrently. The proposed ReBIRA scheme uses a local bitmap (i.e., a small bitmap) to store fault information of the faults detected by the BIST circuit. Once the local bitmap is full, the BIST is paused and the ReBIRA allocates redundancies according the the fault information. After the ReBIRA allocates a redundancy to repair a corresponding faulty row or column, the local bitmap is updated and the BIST is resumed. This process is iterated until the test and repair process is completed. Once one spare element is allocated, the ReBIRA module stores the corresponding repair signature in its Repair Signature Register. If the BIST is not completed, then the BIST continues to test the RAMs. When the BIST and BIRA are completed, the repair signatures stored in the Repair Signature Register are shifted into the Fuse Register of Fuse Macro through the RSO (repair signature output). Before programming the fuses, the user can first load repair signatures into the repair registers in the Wrappers. Then the BIST is used to test the repaired RAMs again. This is called *prefuse testing*. Subsequently, if the perfuse testing is completed and no fault is detected, the repair signatures in the Fuse Macro are exported to the fuse-programming equipment or circuit through TDO. Then the repair signatures can be programmed into the fuses. Note that the prefuse testing can be optional. The user can directly program the fuse without executing the perfuse testing if programmable fuses are used. Fig. 5(b) shows the repair process during the normal operation phase. Note that if a soft repair strategy (only registers are used to store repair signatures [4]) is used in the ReBISM scheme, then this phase is not needed. In the life time of the repaired RAMs, once the power of the RAMs is turned on, the repair signatures stored in the fuses are loaded into the Fuse Register of the Fuse Macro by asserting the signal LD. Thenthe repair signatures in the Fuse Register are shifted into the repair registers in

Wrappers through the Fuse input(FI)and



5. (a) Repair process during test and repair phase. (b) Repair process ig normal operation phase.

Fuse output (FO). The time for setting the repair signatures is called *repair setup time*. Once the repair signatures have been loaded into the repair registers, the RAMs can be accessed normally.

In an SOC, multiple ReBISM circuits are allowed. Thus, RAMs in the SOC can be divided into groups and each group is served by one ReBISM circuit. The grouping should consider

the issues of power, repair time, area cost, etc. Then the IEEE 1500 test wrappers [29] can be used to control the ReBISM circuits. Also, all Fuse Macros can be connected through theTDI and TDO. Therefore, the chip-level IEEE 1149.1 [30] can communicate with the Fuse Macros through TDI and TDO. An example is illustrated to explain the chip-level controlling

scheme further. Fig. 6 shows a chip-level controlling scheme for multiple ReBISM circuits. Assume that RAMs in a chip are divided into three groups and each group has a ReBISR circuit. As Fig. 6 shows, the CTR and ReBIRA of the ReBISM circuit can be controlled by a 1500-compatible interface [31]. Then all the 1500 interfaces are handled by the 1149.1 interface. The 1149.1 can be extended to support the controlling of 1500interfaces [32]. If the test and repair phase is performed, then the required control signals and instructions can be imported

to CTR and ReBIRA circuits of the ReBISM circuits through the TAP (test access port) of the 1149.1 interface and the 1500 interfaces. When the test and repair phase is completed, the ReBIRA in each group exports the repair signature into the Fuse Register of the individual Fuse Marco. If prefuse testing is considered, then the repair signature in each Fuse Register is Shifted into repair registers of the Wrappers of the memories in the corresponding group. Otherwise, the repair signatures of all groups can be exported to the fuse-programming equipment or circuit through the TDI and TDO and the fuses can be programmed. When the chip is used in field, the fuse control signal (FCS) is asserted and the repair signature stored in the Fuse of each Fuse Macro is updated to the corresponding Fuse Register. Then the repair signature in each Fuse Register is shifted into the repair registers of the Wrappers of memories in corresponding group through the FI and FO. Thus the memory repair is completed.

B. Redundancy Analysis Algorithm

In this section, we describe the proposed redundancy analysis algorithm—*range-checking first algorithm* (RCFA). The RCFA can support two types of redundancy configurations—spare row/spare column and spare row/spare IO. Also, the RCFA can be used for a RAM with local redundancies (i.e., each RAM block has individual redundancies) or for a RAM with global redundancies and local redundancies. For brevity, we only present the RCFA algorithm for a RAM with more complicated redundancies, i.e., a RAM with global redundancies and local redundancies. But, the RCFA can easily be modified to fit a RAM with local redundancies. Consider that a RAM consists of two sub arrays. Also, each sub array of the RAM has individual spare column and the two sub arrays has global spare rows. Fig. 7 depicts an example of a RAM with one global spare row (GSR) and two local spare columns.

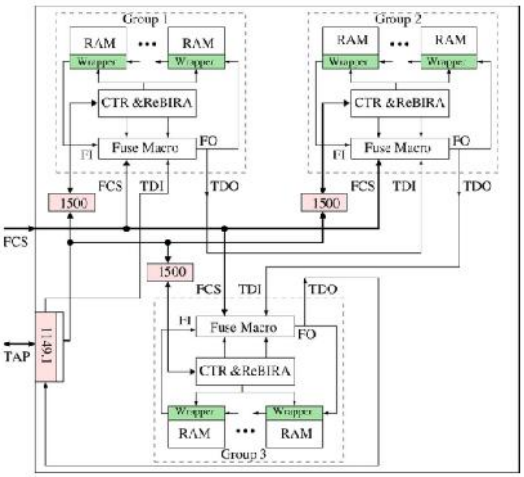


Fig. 6. Example of chip-level controlling scheme for multiple ReBISRs in an SOC.

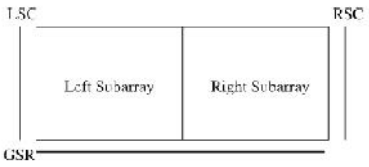
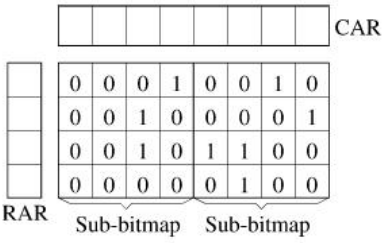


Fig. 7. Example of a RAM with one global spare row and two local spare columns.

The right local spare column (RSC) and the left spare column (LSC) only can replace the corresponding defective column at the right sub array and left sub array, respectively. If a RAM is equipped with global spare rows and local spare IOs, the use of spare IOs is similar to that of spare columns. Therefore, we only describe the RCFA based on a RAM with global spare rows and local spare columns in this paper.



Example of a local bitmap.

The RCFA requires a local bitmap for storing fault information from the BIST. Once the local bitmap is full, the

RCFA performs the redundancy analysis based on the stored fault information. Fig. 8 shows an example of the local bitmap for a RAM. The local bitmap is divided into two parts—left sub-bitmap and right sub-bitmap with respect to the left subarray and the right subarray of the targeted RAM. Each row and column of the local bitmap has one individual row address register (RAR) and column address register (CAR) for storing the faulty addresses. One major feature of the RCFA is that the algorithm first checks the number of row entries and the number of column entries with fault information in the local bitmap. Then if the number of row entries with fault information is larger than the number of column entries with fault information, the algorithm allocates a spare column to replace the corresponding faulty column with the largest number of faulty bits. Otherwise, the algorithmallocates a spare rowto replace the corresponding faulty row with the largest number of faulty bits. As Fig. 8 shows, the number of row entries with fault information is 4. Also, the number of column entries with fault information in the left sub-bitmap and in the right sub-bitmap is 2 and 4, i.e., and , where and denote the number of column entries with fault information in the left sub-bitmap and the right sub-bitmap, respectively. Subsequently, more detail redundancy allocation procedure of the RCFA is described. First, more notations used in the description of RCFA is defined: 1) —number of row entries in the local bitmap; 2) —number of column entries in each sub-bitmap; 3) —the row with maximal number of faulty bits in the local bitmap; 4) —the column with maximal number of faulty bits in the sub-bitmap. Major steps of the redundancy allocation procedure of RCFA are shown in Algorithm 1. An example is illustrated to further explain the RCFA. Assumethat a repairable RAM has two spare rows and two spare local columns. Also, the corresponding local bitmap consists of two 4 4 sub-bitmaps as shown in Fig. 9, where the RAR and CAR are not shown for brevity. The data 0 or 1 in the local bitmap denotes the position of row address and column address stored in the corresponding RAR and CAR is fault-free or faulty.

Two different cases are illustrated in this example. Case 1: Assume that a defective RAM is tested by the BIST circuit and the status of the corresponding local bitmap is shown in Fig. 9(a).

Since the local bitmap is full, the RCFA first checks the , and . In this case, , and . Since is larger than and , an available spare column is used to replace . In this case, the is at the right sub-bitmap. Thus, the local spare column in the right subarray is used to replace the corresponding faulty column. Assume that no any other fault is detected and the BIST is done after the local spare column is allocated. Then the , and the number of faulty bits in and is 1. However, the number of available spare rows and local spare columns is 2 and 1, respectively. Therefore, an available spare row is first used to replace the corresponding faulty row. Finally, the remaining faulty bit can be repaired by the available spare row or local spare column. Fig. 9(b) shows the repaired status when redundancy analysis is done.

Algorithm 1 RCFA: A Range-Checking First Algorithm for Allocating 2D Redundancy

1: Run BIST; pause and jump to Step 2 when it detects a fault.

2: Check whether the detected fault has been repaired. If so, go to Step 1. Otherwise, go to Step 3.

3: Check whether the Hamming syndrome has multiple 1s (The Hamming syndrome is defined as the modulo-2 sum of the expected (fault-free) data output vector and the output vector from the RAM under test, for word-oriented RAMs [33].) If so, repair the faulty word with an available spare row. Otherwise, store the corresponding Hamming syndrome in the local bitmap.4: Check whether the local bitmap is full. If so, go to the next step. Otherwise, go to Step 1.5: If , allocate an available spare column to replace the (if available spare column is exhausted, allocate an available spare row to replace the); else if , allocate an available spare row to replace the (if available spare row is exhausted, allocate an available spare column to replace the); else , replace the faulty element with the largest number of faulty bits with a corresponding available spare element. If spare elements are exhausted, the RAM is unrepeatable. 6: Check whether the BIST is done. If so, go to the next step. Otherwise, go to Step 1 when the local bitmap is not full; go to Step 5 when the local bitmap is still full. 7: Check whether the local bitmap is empty. If so, export the repaired addresses and then stop. Otherwise, go to Step 5. Case 2: if the local bitmap is full and the status is shown as Fig. 9(c). In this case, , and . Since is larger than , RCFA allocates an available spare row to replace (i.e., the row address with respect to the second row entry of the local bitmap). Again, assume that no more fault is detected and the BIST is done after the first spare row is allocated. Then , and the number of faulty bits in and is 1. Therefore, either an available spare row is first used to replace the corresponding faulty row or an available local spare column in the left sub array is first used to replace the corresponding faulty column. Assume that an available spare column is used to replace one faulty column.Finally, the remaining faulty bit is repaired by an available spare row. Fig. 9(d) shows the repaired status when redundancy analysis is done.

C. Design of the ReBIRA

Fig. 10 shows the simplified block diagram of the proposed ReBIRA, which mainly consists of a multi-fault detector (MFD), a hamming syndrome (HS) encoder, a local bitmap, an FSM, an address masker (AM), and a configurable repair signature register (CRSR). The MFD can detect whether the Hamming syndrome of the detected fault has multiple 1s or not. If so, the corresponding faulty row is repaired by an available spare row. Otherwise, the HS only has one 1 and the HS is encoded into a bit address by the HS encoder. Then the bit address is stored in the local bitmap. If the local bitmap is full, then the redundancy allocation is executed. The FSM major realizes the redundancy allocation procedure of RCFA.

The identification register (ID Reg) and Parameters are configuration data for the AM, local bitmap, and CRSR for each RAM, since different RAMs have different lengths of addresses and different number of redundancies. The AM can set the corresponding registers of the local bitmap to fit the configuration of the RAM under test according to the parameters. The length of CRSR for a RAM is associated to the number of redundancies and the number of

bits of row address and column address. Therefore, the CRSR also should be configured as different lengths to fit the configuration of the RAM under repair.

To support RAMs with different configurations, each design parameter of the ReBIRA for multiple RAMs is defined as the largest one among all the RAMs under test. For example, consider four RAMs sharing a ReBIRA and the configuration of these four RAMs are shown in Table I, where (Row Column Word) denotes the row address width, column address width, and word width; denotes the numbers of spare rows and columns, respectively. Therefore, design parameters of the ReBIRA for these four RAMs as shown in Table I are and . The proposed ReBISM scheme serially tests and repairs each

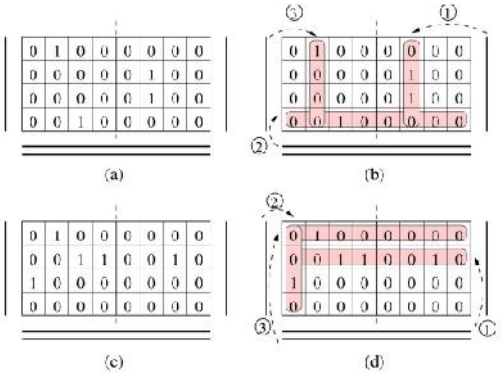


Fig. 9. Example for two different defective RAMs.

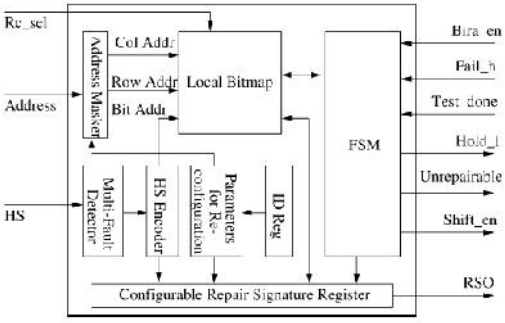
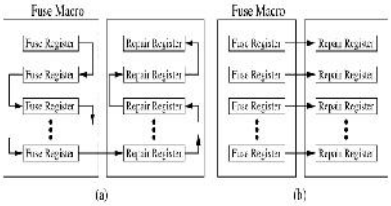


Fig. 10. Simplified block diagram of the proposed ReBIRA.

TABLE I
EXAMPLE OF RAM CONFIGURATIONS

	RAM Size (Row×Column×Word)	Redundancy Configuration (r, c)
RAM 1	9×6×32	(1,2)
RAM 2	7×7×61	(0,2)
RAM 3	9×6×128	(2,2)
RAM 4	8×8×16	(0,2)

RAM. Therefore, the local bitmap and CRSR must be configured as the configuration of the RAM under test and repair. Subsequently, we briefly explain the operation of the Re- BIRA. If the signal Bira en is asserted, the ReBIRA monitors the signal Fail h which is from the BIST circuitry. If the BISTdetects a fault, it asserts the corresponding bit of Fail h signal to 1 and exports the fault information (Address and HS) to the ReBIRA. Also, the ReBIRA sets the Hold l to 1 and the BIST is paused simultaneously. If the redundancy analysis result is unrepairable, then the ReBIRA sets the Un repairable signal to 1. If the ReBIRA completes the redundancy analysis for the fault information stored in the local bitmap, it resets the Hold l signal to 0, which resumes the BIST circuitry to continue the testing process. Once the redundancy analysis of one defective RAM is completed and the first fault of the next defective RAM is detected, the Shift en is asserted to 1 and the corresponding repaired addresses in the CRSR of the repaired RAM is exported to the Fuse Macro through the output RSO (repair signature output). The redundancy configuration selection signal Rc sel is used to configure the AM for different redundancy configurations. If the RAM under test is equipped with spare rows and spare columns (IOs), then the Rc sel is set to 0 (1). When , the AM sets the corresponding column address registers of the local bitmap to all-0 state. The AM also is configured according to the corresponding parameter of the RAM, such that the local bitmap can be configured as the configuration of the RAM under test and repair. If the testing of all the RAMs is completed, the BIST asserts the Test done signal to high. Then the ReBIRA first checks whether the local bitmap is empty or not. If the local bitmap is not empty, the ReBIRA allocates redundancy to repair the faulty elements stored in the local bitmap. Then the repair signatures in the CRSR are exported to the Fuse Macro. Otherwise, the ReBIRA directly exports the repair signatures to the Fuse Macro.



1. (a) Serial access of the fuse information. (b) Parallel access of the fuse information [1]

IV. EXPERIMENTAL RESULTS

A. Analysis of the RCFA Algorithm

We estimate the repair efficiency of the proposed BIRA algorithm in terms of repair rate which is defined as the ratio of the number of repaired RAMs to the number of defective RAMs [22]. A simulator is used to evaluate the repair efficiency of BIRA algorithms. The simulator is implemented according to the methodology reported in [34] and [35]. The simulator can Table II summarizes the repair rates of an exhaustive search algorithm (i.e., the algorithm can get the best repair rate) and our BIRA algorithm for Case 1. In the table, the and denote the number of spare rows and spare columns. Also, Best denotes the exhaustive search algorithm. Note that the spare columns are evenly divided in left and right sub arrays. That is, if , one is for the left sub array and the other is for the right sub array. As Table II shows, the repair rate of the proposed BIRA algorithm can achieve very good repair rate. For most the simulated redundancy configurations, the repair rates of our algorithm approximate to those of the exhaustive search algorithm. For Case 2, 50% single-cell faults, 20% faulty rows, 20% faulty columns, and 10% column twin-bit faults (two adjacent bits in a column are faulty) are injected

TABLE II
REPAIR RATES OF THE EXHAUSTIVE SEARCH (BEST) AND RCFA ALGORITHMS FOR CASE 1

(r,c)	Best	RCFA	(r,c)	Best	RCFA
(0,2)	29.8%	29.8%	(2,4)	97.0%	93.2%
(0,4)	67.4%	67.4%	(3,0)	62.4%	62.4%
(1,0)	12.6%	12.6%	(3,2)	93.4%	93.2%
(1,2)	57.4%	57.2%	(3,4)	99.4%	97.2%
(1,4)	87.6%	81.2%	(4,0)	87.0%	87.0%
(2,0)	44.0%	44.0%	(4,2)	96.6%	96.6%
(2,2)	83.4%	83.2%	(4,4)	99.8%	99.6%

into each RAM core at random locations. Note that these fault types are known based on empirical evidence but the probabilities of occurrence of these fault types are not for an empirical evidence. Apparently, the fault distribution of Case 2 is a very pessimistic assumption. The purpose for simulating this case is to show that the RCFA algorithm can achieve good repair rate even the fault distribution is very worse. Table III summarizes the repair rates of the exhaustive search algorithm and our BIRA algorithm for Case 2. As the table shows, we see that the repair rate of the RCFA is still very good for a RAM with 40% faulty rows and faulty columns. Thus, the repair efficiency of the proposed BIRA algorithm is very high. Note that for some redundancy configurations, the difference of repair rate between the proposed algorithm and the exhaustive algorithm
is larger for a RAM with more spare columns and fewer spare rows, e.g., . The reason is that in the proposed algorithm, if a faulty word has multiple faulty bits, then the faulty word should be repaired with an available spare row.

This causes that if available spare rows are exhausted, then a faulty word with multiple faulty bits cannot be repaired even the number of available spare columns is larger than the number of faulty bits of the faulty word.

TABLE III
REPAIR RATES FOR THE EXHAUSTIVE SEARCH (BEST) AND RCFA ALGORITHMS FOR CASE 2

(r,c)	Best	RCFA	(r,c)	Best	RCFA
(0,2)	28.2%	28.2%	(2,4)	93.8%	92.4%
(0,4)	67.0%	67.0%	(3,0)	61.8%	61.8%
(1,0)	12.4%	12.4%	(3,2)	92.8%	89.8%
(1,2)	57.8%	57.4%	(3,4)	98.2%	96.6%
(1,4)	85.8%	81.2%	(4,0)	86.6%	86.6%
(2,0)	44.0%	44.0%	(4,2)	96.4%	94.8%
(2,2)	82.2%	82.0%	(4,4)	99.8%	98.8%

TABLE IV
RESULTS OF REDUNDANCY ANALYSIS CYCLES

Memory Configurations	Repair Rate(%)	ReBIRA Cycles	BIST Cycles	Ratio (%)
512×16×64	83.6	29952	47939584	0.06
512×8×128	82.3	30698	23605658	0.13
512×4×256	83.8	30404	12013568	0.25

TABLE V
CONFIGURATIONS OF RAMS FOR THREE SIMULATION CASES

	Case 1	Case 2	Case 3
RAM 0	64×2×8	64×2×16	64×2×32
RAM 1	128×4×16	128×4×32	128×2×64
RAM 2	256×8×32	256×8×64	256×4×128
RAM 3	512×16×64	512×8×128	512×4×256

Subsequently, we estimate the analysis time overhead of the RCFA. The required redundancy analysis time of the RCFA is also simulated by the implemented simulator. Three RAMs with different configurations are simulated. For each configuration, 500 samples are simulated and only defects resulting in single-cell faults are injected. Poisson defect distribution is considered and the maximum number of defects is 10. Also, assume that the BIST executes a March-14 test with solid data backgrounds. Table IV summarizes the results of repair rate, Re- BIRA cycles, BIST cycles, and the ratio of ReBIRA cycles to BIST cycles, where each RAM with 2 spare rows and 2 local spare columns is assumed. Note that the needed BIST cycles for the un repairable RAMs are not included in the BIST cycles.

As Table IV shows, the ReBIRA needs 29 952 cycles, 30 698 cycles, and 30 404 cycles to complete the redundancy analysisfor the defective and repairable RAMs with 512 16 64 bits,512 8 128 bits, and 512 4 256 bits, respectively. Also, the ratio of the ReBIRA analysis time to the BIST test time for the three RAMs is about 0.06%, 0.13%, and 0.25%, respectively. Thus the time overhead caused by the proposed ReBIRA scheme is very small.

B. Area Overhead of the ReBISR

We implemented the ReBISR scheme for three cases. The RAM configurations in each case are summarized in Table V, where each RAM equipped with two spare rows and two spare columns are assumed. Also, FTC (Faraday Technology Cooperation) 0.13- m standard cell library is used to realize the Re-BISR circuit. According to Table V shows, we see that design parameters of the ReBISR for Case 1, Case 2, and Case 3 shouldbe 512 16 64, 512 8 128, and 512 4 256.Table VI summarizes simulation and comparison results of the ReBISM scheme and the dedicated BISR (DeBISR) scheme, i.e., each RAM has a self-contained BISR circuit. The secondrow shows the area of RAMs for each case. The third row shows the area of ReBISR circuit for each case; and the

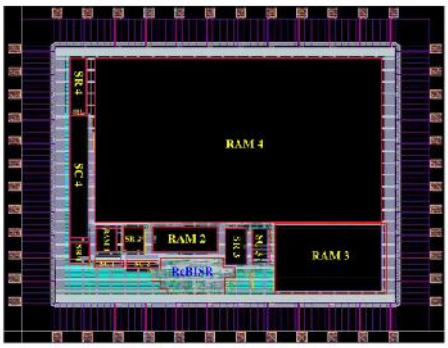


Fig. 15. Layout view of a ReBISR circuit for four RAMs.

TABLE VI
RESULTS OF AREA AND TIMING FOR THE SIMULATED THREE CASES

	Case 1	Case 2	Case 3
Area of RAMs (A_{RAM})	1496258 um^2	1497561 um^2	1528848 um^2
Area of ReBISR (A_{ReBISR})	$25270um^2$	$30618um^2$	$41295um^2$
Area of DeBISRs (A_{DeBISR})	$51628um^2$	$58374um^2$	$69527um^2$
A_{ReBISR}/A_{RAM}%	1.69%	2.04%	2.70%
A_{DeBISR}/A_{RAM}%	3.45%	3.90%	4.55%
ARR	51.04%	47.55%	40.61%
Clock cycle	2.5ns	2.5ns	2.5ns

TABLE VII
COMPARISON RESULTS OF THE REBIRA SCHEME AND BIRA SCHEMES WITH EXHAUSTIVE SEARCH ALGORITHMS

BIRA Schemes	RAM Type	BIRA Arch.	Repair Rate	Area Cost	Time Cost
BIRA1 [7]	BOM	Dedicated	Best	High	Short
BIRA2 [12]	WOM	Dedicated	Best	High	Short
BIRA3 [17]	WOM	Dedicated	Best	Medium	Long
ReBIRA	WOM	Shared	Near Best	Low	Short

fourth row reports the area of DeBISR circuit .As the fifth row shows, the area overhead of the ReBISM circuit for the third

case is only about 2.70%. But, the area overhead of the De-BISR circuit is about 4.55% for the same case.Apparently, the area cost of the ReBISM scheme is much lower than that of the DeBISR scheme. As the 6th row depicts, the area reduction ratio(ARR) which is defined as % is larger than 40% for the three cases. That is, the ReBISM
scheme can save much more area cost in comparison with the DeBISR scheme. The last row of Table VI shows the delay of the ReBISR circuit, which is about 2.5 ns. Fig. 15 shows the layout view of the proposed ReBISM scheme for the RAMs in Case 1.Subsequently, comparison results of the proposed ReBIRA scheme and existing BIRA schemes which provide the best repair rate are shown. Table VII summarizes the comparison results. The BIRA1 [7] is developed for bit-oriented memories (BOMs) only. However, most of embedded memories are word oriented memories (WOMs). To cope with this problem, theBIRA2 [12] extends the BIRA1 [7] to support word-oriented memories. However, both BIRA1 and BIRA2 require multiple

TABLE VIII
COMPARISON OF DIFFERENT E2R SCHEMES WITH HEURISTIC REDUNDANCY ANALYSIS ALGORITHMS

BISR Scheme	Type	Flexibility	Area Cost	Application
[2]	Dedicated	Low	4.68% (one 256×64-bit SRAM)	General
[8]	Shared	High	2.34% (Swapper area for one 32K×32-bit SRAM)	Specific
[15]	Shared	High	4.68% (two 8K×16-bit, one 4K×16-bit, and one 2K×32-bit SRAMs)	General
Ours	Shared	High	2.7% (one 128×32-bit, one 256×64-bit, one 1K×128-bit, and one 2K×256-bit SRAMs)	General

REFERENCES

[1] R. Rajsuman, "Design and test of large embedded memories: An overview," *IEEE Des. Test Comput.*, vol. 18, no. 3, pp. 16–27, May 2001.

[2] Y. Zorian, "Embedded memory test&repair: Infrastructure IP for SOC yield," in *Proc. Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2002, pp. 340–349.

[3] I. Kim, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L. Lweandowski, "Built in self repair for embedded high density SRAM," in *Proc. Int. Test Conf. (ITC)*, Oct. 1998, pp. 1112–1119.

[4] V. Schober, S. Paul, and O. Picot, "Memory built-in self-repair using redundant words," in *Proc. Int. Test Conf. (ITC)*, Baltimore, MD, Oct. 2001, pp. 995–1001.

[5] S. Nakahara, K. Higeta, M. Kohno, T. Kawamura, and K. Kakitani, "Built-in self-test for GHz embedded SRAMs using flexible pattern generator and new repair algorithm," in *Proc. Int. Test Conf. (ITC)*, 1999, pp. 301–310.

[6] D. K. Bhavsar, "An algorithm for row-column self-repair of RAMs and its implementation in the Alpha 21264," in *Proc. Int. Test Conf. (ITC)*, Atlantic City, NJ, Sep. 1999, pp. 311–318.

[7] T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada, and H. Hidaka, "A built-in self-repair analyzer (CRESTA) for embedded DRAMs," in *Proc. Int. Test Conf. (ITC)*, 2000, pp. 567–574.

[8] C.-L. Su, R.-F. Huang, and C.-W.Wu, "A processor-based built-in selfrepair design for embedded memories," in *Proc. 12th IEEE Asian Test Symp. (ATS)*, Xian, Nov. 2003, pp. 366–371.

[9] Y. Zorian and S. Shoukourian, "Embedded-memory test and repair: Infrastructure IP for SoC yield," *IEEE Des. Test Comput.*, vol. 20, pp. 58–66, May–Jun. 2003.

[10] M. Nicolaidis, N. Achouri, and S. Boutobza, "Optimal reconfiguration functions for column or data-bit built-in self-repair," in *Proc. Conf. Des., Autom., Test Eur. (DATE)*, Munich, Germany, Mar. 2003, pp. 590–595.

[11] M. Nicolaidis, N. Achouri, and S. Boutobza, "Dynamic data-bit memory built-in self-repair," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, San Jose, CA, Nov. 2003, pp. 588–594.

[12] D. Xiaogang, S. M. Reddy, W.-T. Cheng, J. Rayhawk, and N. Mukherjee, "At-speed built-in self-repair analyzer for embedded word-oriented memories," in *Proc. Int. Conf. VLSI Des.*, 2004, pp. 895–900.

[13] J.-F. Li, J.-C. Yeh, R.-F. Huang, and C.-W. Wu, "A built-in self-repair design for RAMs with 2-D redundancies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 6, pp. 742–745, Jun. 2005.

[14] 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 Toler. VLSI Syst. (DFT)*, Monterey, CA, Oct. 2005, pp. 81–89.

[15] T.-W. Tseng, J.-F. Li, C.-C. Hsu, A. Pao, K. Chiu, and E. Chen, "A reconfigurable built-in self-repair scheme for multiple repairable RAMs in SOCs," in *Proc. Int. Test Conf. (ITC)*, Santa Clara, CA, Oct. 2006, pp. 1–8, Paper 30.2.

[16] S. K. Thakur, R. A. Parekhji, and A. N. Chandorkar, "On-chip test and repair of memories for static and dynamic faults," in *Proc. Int. Test Conf. (ITC)*, Santa Clara, CA, Oct. 2006, pp. 1–10, Paper 30.1.

[17] P. Ohler, S. Hellebrand, and H.-J. Wunderlich, "An integrated built-in self-test and repair approach for memories with 2D redundancy," in *Proc. IEEE Eur. Test Symp. (ETS)*, Freiburg, May 2007, pp. 91–99.

[18] C.-D. Huang, J.-F. Li, and T.-W. Tseng, "ProTaR: An infrastructure IP for repairing RAMs in SOCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 10, pp. 1135–1143, Oct. 2007.

[19] T.-W. Tseng, C.-H. Wu, Y.-J. Huang, J.-F. Li, A. Pao, K. Chiu, and E. Chen, "A built-in self-repair scheme for multiport RAMs," in *Proc. IEEE VLSI Test Symp. (VTS)*, Berkeley, CA, May 2007, pp. 355–360.

[20] A. Benso, S. Chiusano, G. D. Natale, and P. Prinetto, "An on-line BIST RAM architecture with self-repair capabilities," *IEEE Trans. Reliab.*, vol. 51, no. 1, pp. 123–128, Mar. 2002.

[21] S.-Y. Kuo and W. K. Fuchs, "Efficient spare allocation in reconfigurable arrays," *IEEE Des. Test Comput.*, vol. 4, no. 1, pp. 24–31, Feb. 1987.

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

[23] T.-W. Tseng, J.-F. Li, and D.-M. Chang, "A built-in redundancy-analysis scheme for RAMs with 2D redundancy using 1D local bitmap," in *Proc. Conf. Des., Autom., Test Eur. (DATE)*, Munich, Germany, Mar. 2006, pp. 53–58.

[24] S.-K. Lu, Y.-C. Tsai, C.-H. Hsu, K.-H.Wang, and C.-W.Wu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 1, pp. 34–42, Jan. 2006.

[25] C.-D. Huang, T.-W. Tseng, and J.-F. Li, "An infrastructure IP for repairing multiple RAMs in SOCs," in *Proc. IEEE Int. Symp. VLSI Des.,Autom., Test (VLSI-DAT)*, Hsinchu, Apr. 2006, pp. 163–166.

[26] M. R. Ouellette, D. L. Anand, and P. Jakobsen, "Shared fuse macro for multiple embedded memory device with redundancies," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Diego, CA, May 2001, pp. 191–194.