A Built-In Self-Repair Scheme for Semiconductor Memories with 2-D Redundancy

Jin-Fu Li
Department of Electrical Engineering
National Central University
Jungli, Taiwan 320

Jen-Chieh Yeh, Rei-Fu Huang, and Cheng-Wen Wu
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 300

Peir-Yuan Tsai, Archer Hsu, and Eugene Chow
ADMTek Incorporated
Hsinchu, Taiwan 300

Abstract

Embedded memories are among the most widely used cores in current system-on-chip (SOC) implementations. Memory cores usually occupy a significant portion of the chip area, and dominate the manufacturing yield of the chip. Efficient yield-enhancement techniques for embedded memories thus are important for SOC. In this paper we present a built-in self-repair (BISR) scheme for semiconductor memories with 2-D redundancy structures. The BISR design is composed of a built-in self-test (BIST) module and a built-in redundancy analysis (BIRA) module. Our BIST circuit supports three test modes: the 1) main memory testing, 2) spare memory testing, and 3) repair modes. The BIRA module executes the proposed redundancy analysis (RA) algorithm for RAM with a 2-D redundancy structure, i.e., spare rows and spare columns. The BIRA module also serves as the reconfiguration (address remapping) unit in the normal mode. Experimental results show that a high repair rate (i.e., the ratio of the number of repaired memories to the number of defective memories) is achieved with the proposed RA algorithm and BISR scheme. The BISR circuit has a low area overhead—about 4.6% for an 8K×64 SRAM.

Keywords: built-in self-test, built-in self-repair, built-in redundancy-analysis, memory testing, semiconductor memory.

1. Introduction

Memories are key components of a typical system-on-chip (SOC). They normally are dense and covers a large portion of the chip area, thus dominate the yield of the chip. Keeping the memory cores at a reasonable yield level is thus vital for SOC products. For such purpose, memory designers usually employ redundancy repair—using, e.g., spare rows and/or spare columns of cells—to improve the yield [1–4]. However, redundancy increases silicon area and thus has a negative impact on yield. To maximize the yield with a reasonable cost, redundancy analysis (RA) is necessary. Conventionally, RA is performed on the host computer of the automatic test equipment (ATE) if it is an on-line process, or on a separate computer if it is an off-line process. Either way it is time consuming since RA algorithms are complicated and the memories that implement redundancies are usually large. Moreover, embedded memories are harder to deal with using ATE, and few believe that any known ATE architecture can accurately test tomorrow's system chips for the demanded yield and reliability [5]. Therefore, built-in self-test (BIST) has become a standard solution for embedded memories [6–10], and built-in redundancy analysis (BIRA) and built-in self-repair (BISR) also are gaining popularity. Different BISR approaches have been shown to improve the memory yield from 5 to 20%, such that the net SOC yield increase can range from 2 to 10% [11]. Also, memory BIST and BISR designs have been developed as infrastructure IPs for SOC yield enhancement [12, 13].

Recently, a DRAM installed BISR function is presented in [14]. The defective cells detected by the BIST circuit are replaced by the cells of the spare SRAM. The built-in self-diagnosis method presented in [15] for repairable SRAMs uses a reduced-instruction-set processor to determine a repair solution. In addition to the complicated processor, it requires a large fault-free RAM blocks to store the fail bitmap. In [16] a BISR scheme for ultra-large capacity memory chips is proposed. The proposed memory architecture has a hierarchical organization that optimizes memory access time and increases the efficiency of test and repair. In [4] the authors describe a power-on (soft) BISR design for embedded high density SRAMs with only spare columns. The design can detect defective memory cells and map spare

memory cells to functionally replace the defective cells. Because the 1-D redundancy is implemented, the RA algorithm is simple and straightforward. If the RAM has a 2-D redundancy (with spare rows and columns), the optimal redundancy allocation problem becomes NP-complete [17]. A heuristic RA algorithm and its realization are reported in [18], where the RA algorithm provides about 83.3% repair rate (the ratio of the number of repaired memories to the number of defective memories) for a $512 \times 16 \times 72$-bit RAM with 2×2 redundancies (i.e., 2 spare rows and 2 spare columns). In Alpha 21264, a redundancy analyzer for a RAM core with 1×1 redundancies per memory block was implemented [19]. In another case [20], a BISR design with comprehensive real-time exhaustive search test and analysis (CRESTA) scheme was proposed. It guarantees 100% correction of the repairable chips. However, the hardware cost of CRESTA increases drastically with the number of redundancies. A two-phase RA algorithm and its implementation were later proposed [21]. The algorithm first executes the *must-repair* procedure [22], and then the remaining errors are repaired in sequence by the unused spare lines. The area of the BISR circuit is about 1.7 mm^2 for an 8K$\times$16$\times$256-bit embedded DRAM. Recently, a BISR design consisting of the BIST circuit, redundancy wrapper logic, and fuse boxes that store the failing addresses for word-oriented memories is proposed [23]. By the approach one can use RAMs generated from a memory compiler, without custom designed-in redundancies as those used in conventional RAMs. An on-line BISR design with a transparent BIST algorithm for SRAMs is proposed in [24]. The transparent BIST design allows the memory to be tested and repaired periodically when it is in normal operation.

In this paper we present a BISR design for RAM with 2-D redundancy. In addition to the spare rows and columns that replace the defective cells of the RAM, a software-based address remapping scheme is also used to avoid the defective cells that cannot be repaired. Such a RAM finds applications in any system that allows a variable RAM size, e.g., the packet buffer of a network processing chip, the file buffer of a printer controller chip, etc. The proposed BISR design is composed of the BIST and BIRA modules, in addition to the spare memory blocks and the repair circuitry. The BISR scheme has the main/spare memory testing mode and repair mode. An efficient RA algorithm is also proposed and realized by the BIRA module, which also serves as the reconfiguration circuit in the normal mode—the addresses of the faulty cells are remapped to the fault-free spare addresses. Experimental results show that the repair rate of the proposed redundancy algorithm is good. The BISR hardware overhead is about 4.6% for an 8K$\times$64 SRAM.

The rest of this paper is organized as follows. Section 2 introduces the redundancy organization and the proposed BISR scheme and BIRA procedure. Section 3 presents the BIST and BIRA implementations. In Sec. 4 the experimental results are shown. Finally, Sec. 5 concludes the paper.

2. Proposed BISR Scheme

2.1. Redundancy Organization

The redundancy organization of a RAM affects not only the repair rate but also the area cost of the BIRA circuitry. Figure 1 shows a RAM cell array with redundancy rows and columns. In the figure, a 512-bit RAM has two spare rows at the bottom and four spare columns on the right. If there is a faulty row, any of the spare rows (SR0 and SR1) can be used to replace it. However, the spare columns are used differently: we partition them into several *spare column groups* (SCGs). In the figure, two spare columns are grouped into an SCG, i.e., the *group size* is 2—groups SCG0 (resp. SCG1) contains columns SC0 and SC1 (resp. SC2 and SC3). Moreover, each SCG is logically divided into segments for better utilization, i.e., the segments of a spare column are not physically divided by local sense amplifiers. The switching from the main memory to the spare column groups is controlled by the BISR circuit, so only the multiplexers induce additional access time and area cost. In Fig. 1, one SCG has four segments (SEG0, SEG1, SEG2, and SEG3). The segments are identified by the first two most significant bits (MSBs) of the row address. Let the row addresses be $a_3a_2a_1a_0$, then a_3a_2 specifies the segment where the addressed row sits. Different segments of an SCG can be used to repair defective cells in different columns of the main memory. For example, if the cells C0, C1, C2, and C3 (see Fig. 1) are faulty, then SEG0 and SEG1 of SCG0 can be used to replace them. Note that the spare rows also can be logically divided into segments, though this is not shown in the example. We have developed a simulator for evaluating the RA algorithms of redundancy repairable memories [25].

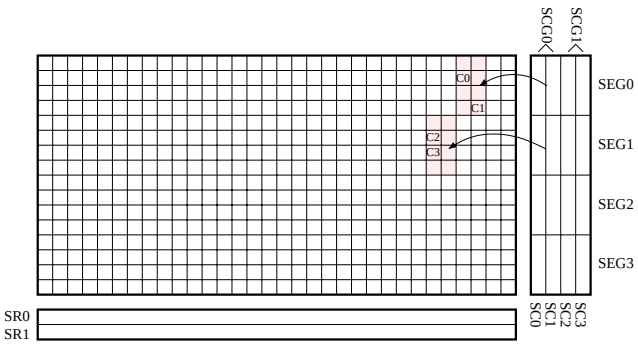

Figure 1. An example RAM with two spare rows and 2 spare column groups.

2.2. BISR Architecture and Procedure

Figure 2 depicts the block diagram of the proposed BISR scheme, including the BIST module, BIRA module, and test

wrapper for the memory. The BIST circuit detects the faults in the main memory and spare memory. It is programmable at the March element level [9]. The BIRA circuit performs redundancy allocation using the proposed RA algorithm (to be discussed later). The test wrapper switches the memory between the test/repair mode and normal mode. In the test/repair mode the memory is accessed by the BIST module, while in the normal mode the wrapper selects the data outputs either from the main memory or the spare memory (replacing the faulty memory cells) depending on the control signals from the BIRA module.

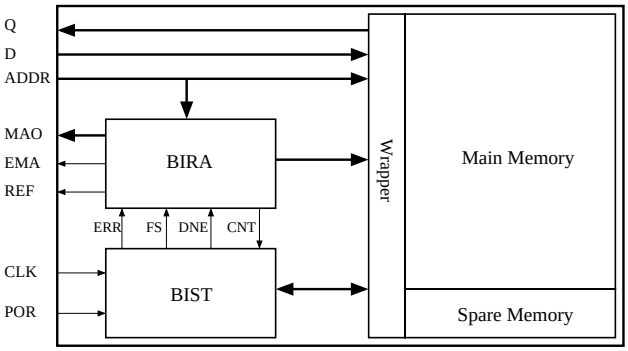

Figure 2. Block diagram of the proposed BISR scheme.

The BISR procedure is shown in Fig. 3. Upon turning on the power, the BIST module starts to test the spare memory. Once a fault is detected, it informs the BIRA module to mark the defective spare row or column as faulty through the error (ERR) and fault syndrome (FS) signals. After finishing the spare memory test, it tests the main memory. If a fault is detected (ERR outputs a pulse), the test process pauses and the BIST module exports the FS to the BIRA module, which then performs the RA procedure. When the procedure is completed and the memory testing is not finished yet, the BIRA module issues a continue signal (CNT) to resume the test process. During the RA procedure, if a spare row is requested but there is no more spare row, the BIRA module exports the faulty row address through the EMA (export mask address) and MAO (mask address output) signals. The memory will then be operated at a down-graded mode (i.e., with a smaller usable capacity) by software-based address remapping. For example, assume that a memory with multiple blocks is used for buffering, and the blocks are chained by pointers. If some block is faulty and should be masked, then the pointers are updated to invalidate the block. The size of the memory is reduced, as one block is removed. The system still works if a smaller buffer is allowed, though the performance may be affected. This approach effectively increases the yield of the products. The number of blocks that can be invalidated normally depends on the performance penalty that can be tolerated. If the down-grade mode is not allowed, the MAO is removed and the EMA indicates whether the memory is repairable.

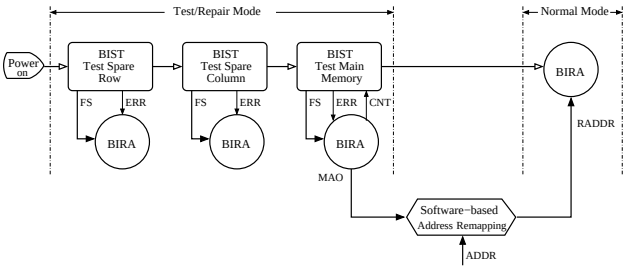

Figure 3. Power-on BISR procedure.

When the main memory test and RA are finished, the REF (repair end flag) signal goes high and the BIRA module switches to the normal mode. The BIRA module then serves as the address remapper, and the memory can be accessed using the original address (ADDR). When the memory is accessed, ADDR is compared with the fault addresses stored in the BIRA module. If ADDR is the same as any of the fault address, the BIRA module controls the wrapper to remap the access to the spare memory.

Let a *subword* be consecutive bits of a word, whose length is the same as the group size. For example, in Fig.1 we assume there are 32 bits in each row, and 2 bits in each subword, so a row has 16 two-bit subwords. To reduce complexity, we use two row-repair rules. One row-repair rule is that if a row has multiple faulty subwords, we repair the faulty row by a spare row if it is available. Take Fig. 4(a) as an example. If a word has two faulty subwords (marked 'X' in the figure), we actually can repair them using two spare column group segments. However, for multiple spare column group segments, there can be many possible ways to repair the faulty subwords, resulting in complex output multiplexing and RA, and thus the memory performance is degraded and the cost of the BIRA circuit increases. After the *must-repair* phase, the spare allocation for remaining faulty elements is performed according to the *repair-most* rules [22].

The other row-repair rule is illustrated in Fig. 4(b). Suppose the faulty cell C0 is repaired by a spare column group segment, and some time later the faulty cell C1 is repaired by a spare row. In this case, when we access the cells in the overlapped region, the address remapper gives the priority to the spare row. This effectively reduces the complexity of the BIRA circuit.

The BIRA procedure consists of the following major steps.

1: Run BIST; pause and jump to Step 2 when it detects a fault.
Stop when BIST is done.
2: Check whether both row-repair rules can be applied. If so, go to Step 4.
3: Allocate a spare row or spare column group segment to repair corresponding faulty cells according to the repair-most rules.
Resume Step 1.
4: Check if there are available spare rows.
If so, repair by a spare row and resume Step 1.
5: Export the corresponding faulty row address;
Resume Step 1.

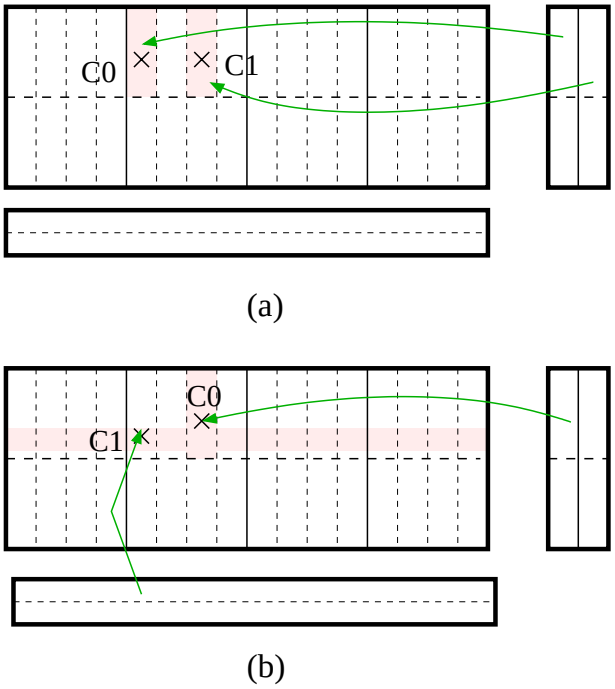

Figure 4. Faulty memory examples.

3. BIST and BIRA Implementation

3.1. BIST Module

The BIST module block diagram is shown in Fig. 5, which consists of a controller (CTR) and a test pattern generator (TPG) for handling test operations and generating test stimulus, respectively. In addition to the clock (CLK), the BIST only needs the power-on reset (POR) signal to initiate the test procedure. The BDN (BIST done), ERR (error indicator), FS (fault syndrome), and CNT (continue) are signals between the BIST and BIRA modules. The TPG output signals are connected to the memory under test. The BNS (BIST normal selection) signal is used to switch the memory between test/repair mode and normal mode.

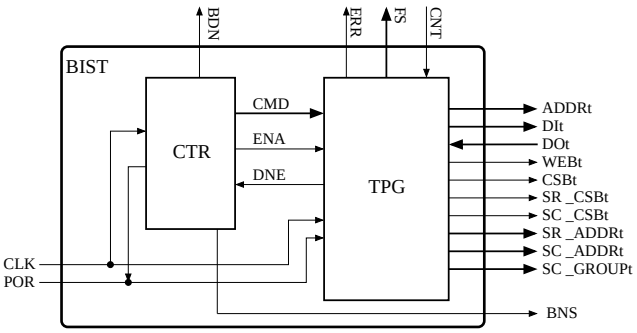

Figure 5. BIST module block diagram.

The CTR is a typical finite state machine (see, e.g., [9, 26]). The TPG executes the test command (CMD) provided by the CTR. When a fault is detected, it pauses and sends the ERR and FS signals to inform the BIRA module to perform RA. When the RA finishes, the BIRA module sends the CNT signal to resume the TPG process. The BIST implementation is typical, similar to our previous design [9].

3.2. BIRA Module

The BIRA module has three components—the multiple faulty subwords detector (MFSD), process element (PE), and address remapping unit (ARU)—as shown in Fig. 6. When the power is on, all flip-flops are reset to the initial state. The signal normal = 0 and FS is connected to the ARU. The input address (ADDR) is sent to the ARU when it is in normal mode (normal = 1). Initially, the signals solid_flag, faulty_flag, repaired_flag, row_match, and col_match are all reset to 0. The PE evaluates the status of these signals and issues the control signals solid_en, repair_en, update, and export_mask_addr to the ARU, which then updates the status of its registers. The signals REF (repair end flag), EMA (export mask address), and MAO (mask address output) are connected to the ATE.

The MFSD detects whether the number of faulty subwords on a row is larger than one. The PE is implemented

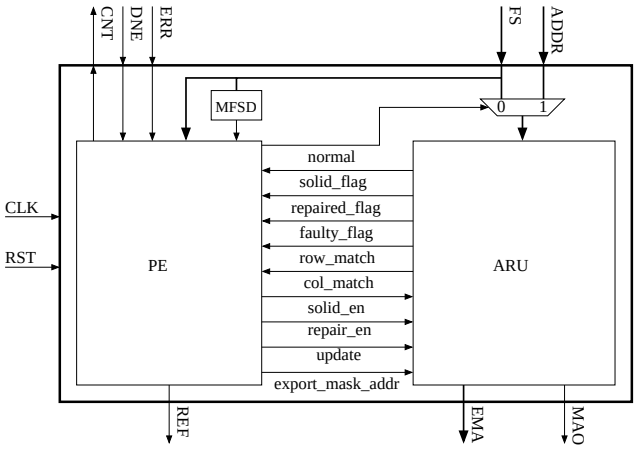

Figure 6. BIRA module block diagram.

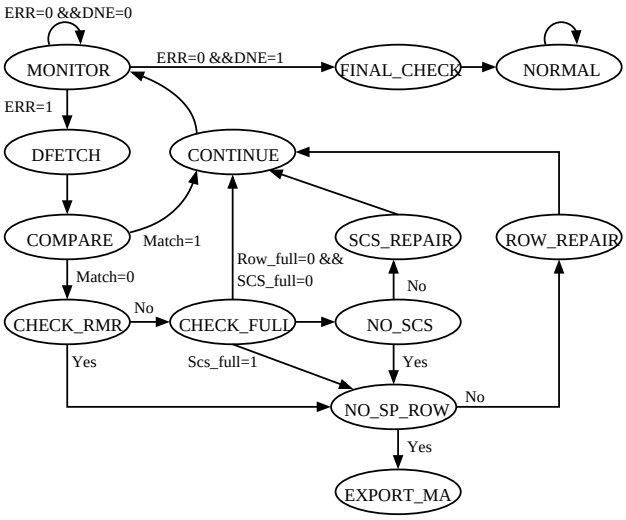

Figure 7. State diagram of the PE.

by a finite state machine (FSM), whose state transition diagram is shown in Fig. 7. The initial state is MONITOR, which monitors the ERR signal from the BIST circuit. If a fault is detected, the PE goes to the DFETCH state to load the status data into the flip-flops. In the COMPARE state, the PE compares the faulty address with the previously stored addresses. If there is a match, the PE goes back to the MONITOR state through the CONTINUE state. Otherwise, it goes to the CHECK_RMR state. If the status is must-repair (by row), then it goes to the NO_SP_ROW state and checks whether there are available spare rows. If no spare row is available, it sends a signal to the ARU that will then export the faulty row address for software-based repair later in the down-graded operation mode. If, on the other hand, a spare row is available, then the faulty row is replaced by the spare row in the ROW_REPAIR state, and the PE goes back to the MONITOR state, through the CONTINUE state where a continue signal is issued to the BIST circuit. If in the CHECK_RMR state the must-repair conditions are not satisfied, the PE will go to the CHECK_FULL state to see if the solid flags (explained below) in the ARU are on—if all the spare rows (resp. column segments) are full, the spare column segments (resp. rows) are used for repair (unless both are full). It then goes to either the NO_SP_SCS state or NO_SP_ROW state. Finally, in the MONITOR state, if the BIST module issues a done signal to the PE, the FSM will go to the NORMAL state through the FINAL_CHECK state, where it checks and sets the flags of all the repaired storage elements.

In the test/repair mode, the ARU stores the addresses of the faulty cells detected so far, and compares the current faulty-cell address with the stored ones. Figure 8 shows the ARU block diagram, which mainly contains the storage elements (registers), comparators, and a signal generator. Each storage element stores a faulty cell information. Assume that there are M spare rows and N spare column group segments. Then M row storage elements and N column storage elements are implemented. Each storage element has three status flags: 1) the fault flag (FF) denotes whether the corresponding spare element is defective (FF = 1) or fault-free (FF = 0); 2) the repair flag (RF) indicates whether the spare element is used to repair the defective main memory (RF = 1) or not (RF = 0); and 3) the solid flag (SF) shows whether the storage element has been loaded the faulty cell information (SF = 1) or not (SF = 0). Each column storage element has an identification (ID) field to store its segment number.

When the system is operated in the test/repair mode, the Row_addr_in (row address input), Col_addr_in (column address input), and Faulty_subword_in (faulty subword input) signals come from the BIST circuit, and the Row_r_en (row repair enable), Row_s_en (row solid enable), Col_r_en (column repair enable), and Col_solid_en (column solid enable) signals are from the PE. The spare rows and columns are first tested. If a fault is detected, the FF of its corresponding storage element is set to 1. The main memory is tested next. If a fault is detected, the row address, column address, and faulty subwords are compared with the data stored in the storage elements with SF = 1 and FF = 0. The results are exported from the Row_match and Col_match terminals to the PE. If there is no match, the fault information is written into an empty storage element, and the PE sets its SF (SF = 1) through the Solid_r_en or Solid_c_en input.

The Row_addr_in and Col_addr_in inputs are the address inputs in the normal mode. When the memory is accessed, the address also is compared with those stored in the storage elements. If it is the same as one of the stored addresses, the signal generator triggers the control signals to reconfigure the I/Os between the main memory and spare memory. The delay time of the accessing path through ARU usually is shorter than that through the main memory, i.e., it does not induce performance penalty. A slight performance degrada-

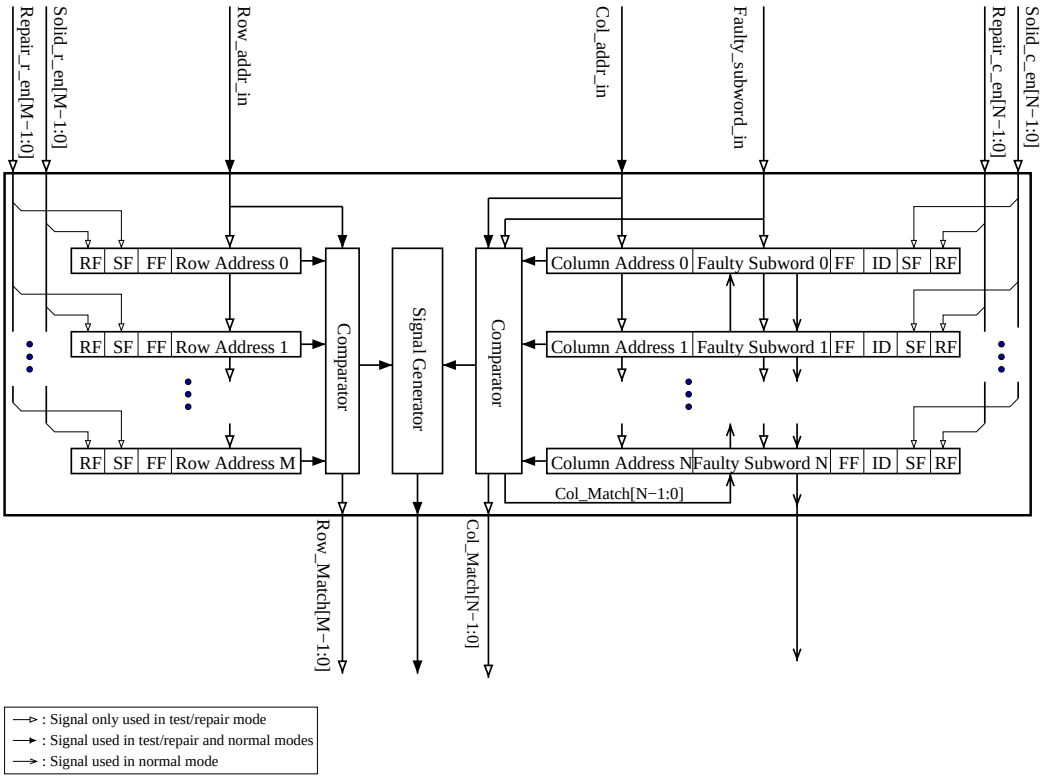

Figure 8. Block diagram of the ARU.

tion results from the wrapper, however.

4. Experimental Results

4.1. An Industrial Case

We have implemented a repairable SRAM with the proposed BISR methodology on an industrial network chip. Figure 9 shows the repairable SRAM, which is composed of two blocks with 8K$\times$32 bits each. Four spare rows and two spare column groups—with a group size of four—are implemented using two separate memory blocks. In the figure, the data input DI is broadcast to the main memory and spare memory. The chip select signals determine which memory the data should be written into. When the memory is in the normal mode, the data output may come from the main memory or spare memory, depending on the control signals sc_group and sr_csb.

The gate count of the BISR (including BIST) design is about 5.6K using a typical synthesis procedure and a standard 0.25μm CMOS cell library. Figure 10 shows the layout of the 8K$\times$64 repairable SRAM. The areas of the SRAM (BANK 1 and BANK 2), BISR module, and spare elements (SPARE ROW and SPARE COL) are 6538362μm^2, 301040μm^2, and 298856μm^2, respectively. The hardware

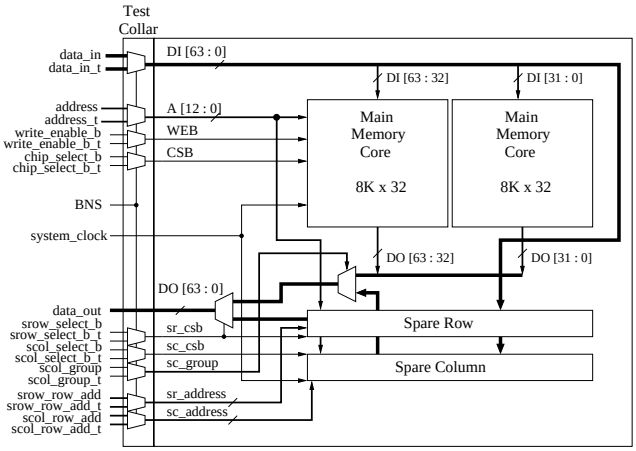

Figure 9. An 8K$\times$64 repairable SRAM with 4 spare rows and 2 spare column groups.

overhead of the spare elements (HO_{spare}) and that of the BISR module (HO_{bisr}) are calculated and shown below.

$$HO_{spare} = \frac{298856}{6538362} \times 100\% = 4.57\% \quad (1)$$

$$HO_{bisr} = \frac{301040}{6538362} \times 100\% = 4.6\% \quad (2)$$

The total hardware overhead for this repairable SRAM is about 9.17%. We guarantee 100% repair rate if the number of random faults is no more than 10 (it will be analyzed in the next section).

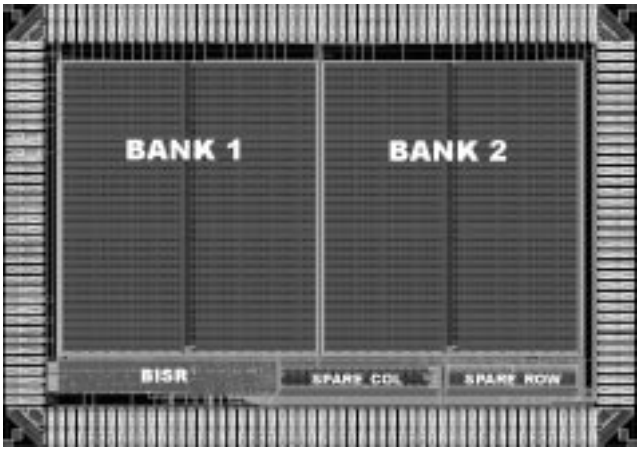

Figure 10. Layout of the 8K×64 repairable SRAM.

Figure 11 shows part of the timing diagram for MAO and EMA from post-layout simulation. It shows that if the spare rows are exhausted but a spare row is still required to repair the defective memory, the address of the defective row is exported to the ATE through the MAO output. When the BIRA circuit wants to export a mask address, the EMA signal becomes 1 such that the ATE can correctly receive the valid mask address. Figure 12 shows a waveform sample of the data inputs/outputs and some control signals of the spare memories during the normal-mode memory access. If sc_csb = 0, the 4-bit data out is from the Spare Column, controlled by sc_group. In this example, sc_group = f7ff, i.e., the 12th four-bit data output is from Qc[3:0] (data outputs of the Spare Column). If both sr_csb and sc_csb are 0, the data output is from the Spare Row, controlled by sr_csb, i.e., the data output is from Qr[63:0] (data output of the Spare Row). This avoids data access conflict.

We have also implemented a repairable SRAM with the proposed BISR scheme in an industrial home-gateway chip. The size of the SRAM is 2,048×27, and four spare rows and 2 spare column groups are implemented. Among the 72 chips tested there is only one chip with a defective SRAM, and the defective SRAM has been repaired with the proposed BISR scheme.

4.2. Repair Rate Analysis

We define the *repair rate* (RR) as the ratio of the number of repaired memories to the number of defective memories. To estimate the repair rate of the proposed BISR scheme, a simulator for evaluating RA algorithms has been implemented [25]. The simulator can evaluate the repair rate and area cost of the redundancies according to the given specifications. Especially, the simulator can simulate the sequence of the faults in the order they are detected by the BIST circuit. It improves the accuracy of the analysis results.

Table 1 summarizes the RRs for different redundancy configurations based on the proposed and exhaustive redundancy analysis algorithms. The number of injected random faults is from 1 to 10, and the number of memory samples is 534. The defect distribution assumed is pessimistic, which is used for evaluating the proposed scheme. Mature products have a far lower defect density. Note that the exhaustive RA algorithm is simulated based on the assumption that a single spare row and spare column can be used to repair any defective row and column, respectively. It guarantees 100% RR under such type of redundancy organization [20]. In the table, N_{SR}, N_{SC}, and N_{SCG} denote the numbers of spare rows, spare columns, and spare column groups. The RR column reports the RR of the proposed approach. The results show that the RR difference between the proposed approach and the best (exhaustive search) is very small for most of the redundancy configurations. In the xMA columns of Table 1, the values represent the numbers of unrepairable memories for the respective spare configurations that can still be used in the down-graded mode if we mask out x faulty-cell row addresses. For example, the 1MA column show the numbers of unrepairable memories which can still be used if 1 masked address is allowed—the memory thus has one less usable address. According to the table, if $N_{SR} = 2$ and $N_{SCG} = 2$, the number of unrepairable memories is 4. Among them, 3 can work in the down-graded mode if 1 masked address is allowed, and 1 memory can work in the down-graded mode if 2 masked addresses are allowed. In the industrial chip design, $N_{SR} = 4$ and $N_{SCG} = 2$. Therefore, the BISR design can achieve 100% RR with low area cost. However, if the CRESTA in [20] is implemented, it requires $C_2^{12} = 66$ sub-analyzers to try 66 possible solutions, resulting in very high hardware cost.

We now discuss the relation between the column group size and RR. Figure 13 plots the RR for different spare configurations with a group size of two. The number of injected random faults is from 1 to 10 and the total number of memory samples is 534. Note that the conventional algorithm being compared here is the one without spare row or column grouping or segmentation, but an exhaustive search is assumed. The figure shows that the RR of the proposed approach is even better than the conventional one in many instances. Figure 14 shows a similar comparison, but now the group size is four. The result of our approach is this

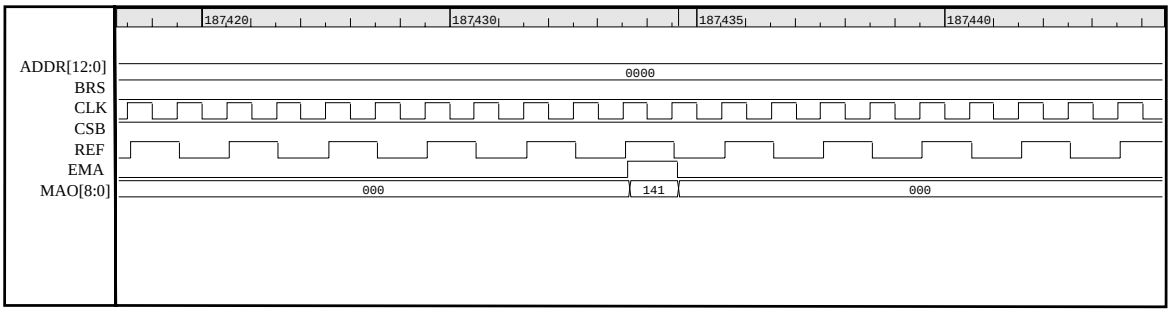

Figure 11. A waveform sample of the EMA and MAO signals.

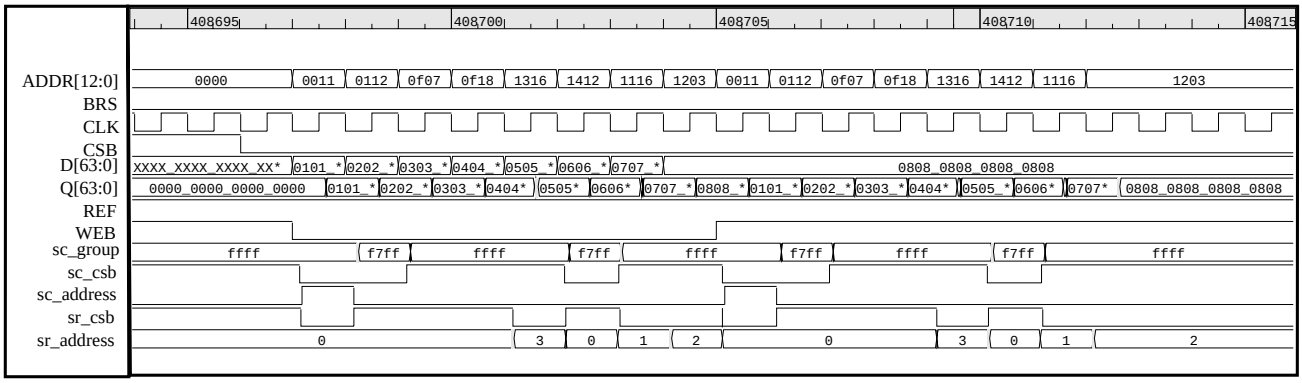

Figure 12. A waveform sample showing normal-mode memory access.

Table 1. Simulation results.

N_{SR}	N_{SC}	N_{SCG}	RR	1MA	2MA	3MA	4MA	5MA	>5MA	RR (Best)
1	0	0	18.37%	99	191	4	69	45	32	18.54%
1	4	1	73.10%	38	40	35	16	9	7	86.14%
1	8	2	94.43%	5	7	12	1	3	2	99.81%
1	12	3	99.26%	1	1	1	1	0	0	100 %
2	0	0	36.55%	192	2	71	46	18	13	37.08%
2	4	1	86.09%	36	16	12	3	8	0	94.01%
2	8	2	99.26%	3	1	0	0	0	0	100%
2	12	3	100%	0	0	0	0	0	0	100%
3	0	0	72.17%	0	75	43	18	7	7	55.06%
3	4	1	96.10%	7	5	4	3	2	0	97.38%
3	8	2	99.81%	1	0	0	0	0	0	100%
3	12	3	100%	0	0	0	0	0	0	100%
4	0	0	72.36%	73	44	18	8	5	1	71.91%
4	4	1	98.52%	4	3	0	0	0	0	98.69%
4	8	2	100%	0	0	0	0	0	0	100%
4	12	3	100%	0	0	0	0	0	0	100%
5	0	0	85.90%	44	18	7	6	1	0	85.77%
5	4	1	99.81%	1	0	0	0	0	0	99.81%
5	8	2	100%	0	0	0	0	0	0	100%
5	12	3	100%	0	0	0	0	0	0	100%
6	0	0	94.06%	18	7	6	1	0	0	94.01%
6	4	1	100%	0	0	0	0	0	0	100%
6	8	2	100%	0	0	0	0	0	0	100%
6	12	3	100%	0	0	0	0	0	0	100%
7	0	0	97.40%	8	5	1	0	0	0	97.57%
7	4	1	100%	0	0	0	0	0	0	100%
7	8	2	100%	0	0	0	0	0	0	100%
7	12	3	100%	0	0	0	0	0	0	100%
8	0	0	98.70%	6	1	0	0	0	0	98.69%
8	4	1	100%	0	0	0	0	0	0	100%
8	8	2	100%	0	0	0	0	0	0	100%
8	12	3	100%	0	0	0	0	0	0	100%

case is not as good as the previous one. However, the area cost of the BIRA circuit is lower. For example, if there are four spare columns, then the number of required column storage elements for the case where group size is two (two spare column groups) is larger than the one whose group size is four (only one spare column group). Note also that the RR is in fact related to how the defects are distributed. An analysis of the fail bit patterns of the target memory design and process technology is required to achieve the most cost-effective solution.

5. Conclusions

We have proposed a BISR scheme for RAM. The BISR circuit is composed of a BIST module and a BIRA module. The BIST circuit supports three operation modes—main memory testing, spare memory testing, and repair. The BIRA circuit executes a proposed RA algorithm for 2-D redundancy—spare rows and spare columns. The spare columns are grouped and segmented. A software-based address remapping (masking) is performed in the downgraded operation mode, where certain amount of unrepairable faulty rows can be tolerated. The experimental results show that high repair rate can be obtained. Compared with the conventional approach (without grouping and segmentation) using exhaustive search, the proposed

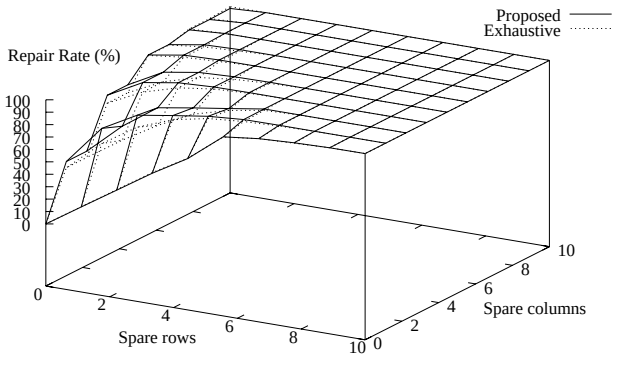

Figure 13. RR comparison when the group size is two.

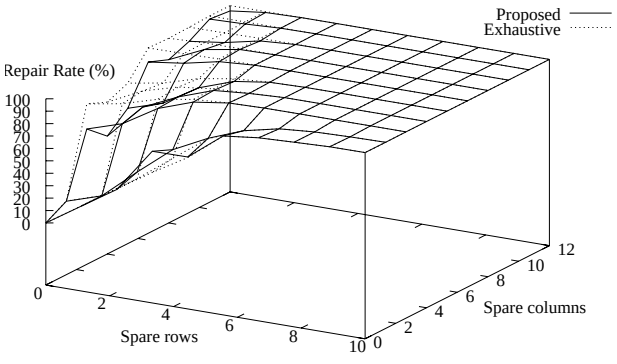

Figure 14. RR comparison when the group size is four.

scheme outperforms in many instances and can be implemented with low area cost. A BISR design for an industrial 8K×64 SRAM with 4 spare rows and 2 spare column groups has also been implemented. In this case, full repair can be achieved if the number of random faults is no more than 10. Moreover, the area overhead of the BISR design is low—about only 4.6% for the 8K×64 SRAM.

References

[1] S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories", *IEEE Journal of Solid-State Circuits*, vol. 13, no. 5, pp. 698–703, Oct. 1978.

[2] M. Horiguchi, J. Etoh, M. Masakazu, K. Itoh, and T. Matsumoto, "A flexible redundancy technique for high-density DRAM's", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 1, pp. 12–17, Jan. 1991.

[3] T. Yamagata, H. Sato, K. Fujita, Y. Nishimura, and K. Anami, "A distributed globally replaceable redundancy scheme for sub-half-micron ULSI memories and beyond", *IEEE Journal of Solid-State Circuits*, vol. 31, no. 2, pp. 195–201, Feb. 1996.

[4] 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.

[5] S. Runyon, "Testing big chips becomes an internal affair", *IEEE Spectrum*, pp. 49–55, Apr. 1999.

[6] P. Camurati, P. Prinetto, M. S. Reorda, S. Barbagallo, A. Burri, and D. Medina, "Industrial BIST of embedded RAMs", *IEEE Design & Test of Computers*, vol. 12, no. 3, pp. 86–95, Fall 1995.

[7] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, ComTex Publishing, Gouda, The Netherlands, 1998.

[8] J. Dreibelbis, J. Barth, H. Kalter, and R. Kho, "Processor-based built-in self-test for embedded DRAM", *IEEE Journal of Solid-State Circuits*, pp. 1731–1740, Nov. 1998.

[9] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu, and T.-Y. Chang, "A programmable BIST core for embedded DRAM", *IEEE Design & Test of Computers*, vol. 16, no. 1, pp. 59–70, Jan.-Mar. 1999.

[10] K. Zarrineh and S. J. Upadhyaya, "On programmable memory built-in self test architecutres", in *Proc. Design, Automation and Test in Europe (DATE)*, Paris, Mar. 1999, pp. 708–713.

[11] R. Rajsuman, "Design and test of large embedded memories: an overview", *IEEE Design & Test of Computers*, vol. 18, no. 3, pp. 16–27, May 2001.

[12] Y. Zorian, “Embedded memory test & repair: infrastructure IP for SOC yield”, in *Proc. Int. Test Conf. (ITC)*, Baltmore, Oct. 2002, pp. 340–349.

[13] Y. Zorian, “Embedded infrastructure IP for SOC yield improvement”, in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, New Orleans, June 2002, pp. 709–712.

[14] Akira Tanabe, Toshio Takeshima, Hiroki Koike, Yoshiharu Aimoto, Masahide Takada, Toshiyuki Ishijima, Naoki Kasai, Hiromitsu Hada, Kentaro Shibahara, Tahemitsu Kunio, Takaho Tanigawa, Takanori Saeki, Masato Sakao, Hidenobu Miyamoto, Hiroshi Nozue, Shuichi Ohya, Tatsunori Murotani, Kuniaki Koyama, and Takashi Okuda, “A 30-ns 64-Mb DRAM with built-in self-test and self-repair function”, *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1525–1533, Nov. 1992.

[15] R. P. Treuer and V. K. Agarwal, “Built-in self-diagnosis for repairable embedded RAMs”, *IEEE Design & Test of Computers*, vol. 10, no. 2, pp. 24–33, June 1993.

[16] T. Chen and G. Sunada, “Design of a self-testing and self-repairing structure for highly hierarchical ultra-large capacity memory chips”, *IEEE Trans. VLSI Systems*, vol. 1, no. 1, pp. 88–97, June 1993.

[17] S.-Y. Kuo and W. K. Fuchs, “Efficient spare allocation in reconfigurable arrays”, *IEEE Design & Test of Computers*, vol. 4, no. 1, pp. 24–31, Feb. 1987.

[18] 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.

[19] 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)*, 1999, pp. 311–318.

[20] 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.

[21] Y. Nagura, M. Mullins, A. Sauvageau, Y. Fujiwara, K. Furue, R. Ohmura, T. Komoike, T. Okitaka, T. Tanizaki, K. Dosaka, K. Arimito, Y. Koda, and T. Tada, “Test cost reduction by at-speed BISR for embedded DRAMs”, in *Proc. Int. Test Conf. (ITC)*, Baltimore, Oct. 2001, pp. 182–187.

[22] M. Tarr, D. Boudreau, and R. Murphy, “Defect analysis system speeds test and repair of redundant memories”, *Electronics*, pp. 175–179, Jan. 12 1984.

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

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

[25] R.-F. Huang, J.-F. Li, J.-C. Yeh, and C.-W. Wu, “A simulator for evaluating redundancy analysis algorithms of repairable embedded memories”, in *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, Isle of Bendor, France, July 2002, pp. 68–73.

[26] C. Cheng, C.-T. Huang, J.-R. Huang, C.-W. Wu, C.-J. Wey, and M.-C. Tsai, “BRAINS: A BIST complier for embedded memories”, in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Yamanashi, Oct. 2000, pp. 299–307.