Test/Repair Area Overhead Reduction for Small Embedded SRAMs

Baosheng Wang† and Qiang Xu‡

† *ATI Technologies Inc., 1 Commerce Valley Drive East, Markham, ON, Canada L3T 7X6, bawang@ati.com*

‡ *Dept. of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, qxu@cse.cuhk.edu.hk*

Abstract

For current highly-integrated and memory-dominant System-on-a-Chips (SoCs), especially for graphics and networking SoCs, the test/repair area overhead of embedded SRAMs (e-SRAMs) is a big concern. This paper presents various approaches to tackle this problem from a practical point of view. Without sacrificing at-speed testability, diagnosis capability and repairability, the proposed approaches consider partly sharing wrapper for identical memories, sharing memory BIST controllers for e-SRAMs embedded in different functional blocks, test responses compression for wide memories, and various repair strategies for e-SRAMs with different configurations. By combining the above approaches, the test/repair area overhead for e-SRAMs can be significantly reduced. For example, for one benchmark SoC used in our experiments, it can be reduced as much as 10% of the entire memory array.[1]

***Keywords**: Embedded Small SRAMs, Area Overhead, BIST, Single-Element Repair*

1. Introduction

One of the trends associated with the System-on-a-Chip (SoC) paradigm is that an increasingly large number of small SRAMs (up to several thousand currently) are embedded on chips, especially for graphics and networking SoCs [1, 4]. Because of their extremely high density, these embedded SRAMs (e-SRAMs) are more prone to manufacturing defects than other types of on-chip circuitries and it is important to test them thoroughly to certify the quality of the product shipped to customers. Since e-SRAMs are usually deeply embedded on-chip and at the same time need to be tested at-speed, Built-in Self-Test (BIST) has become the only practical solution for testing e-SRAMs. In addition, with the increasing total size of small e-SRAMs, it has been shown in [4] that providing repairability for every small e-SRAM can significantly improve the SoC yield and result in substantial cost savings when a high volume of chips are fabricated.

Various e-SRAM test and repair strategies have been proposed in the literature. A programmable memory BIST architecture was introduced in [18] to increase flexibility in applying test patterns targeting various memory faults. In [2], the authors proposed a multi-level test architecture, which enables scalability, in-system programmability and flexibility in test scheduling. Low-power memory wrapper, which uses gray-code-based address generators and hardware-efficient background pattern generators to save test power, was presented in [3]. The authors in [7-8] focused on designing efficient test architecture for diagnosis purpose. E-SRAMs in test mode can consume significant test power, [5] studied the memory test scheduling problem given a power constraint. In [6], the authors combined memory BIST wrapper and processor-based software test strategies for heterogeneous memories.

The test and repair overhead for small e-SRAMs is relatively large because of their small sizes, for instance, the area overhead for an unrepaired e-SRAM with a 16×143 configuration is 38% when tested with March C- [8]. Therefore, effective e-SRAM test and repair architecture needs to minimize the associated overhead as much as possible, in terms of both silicon area and routing. The serial interfacing technique utilized in [9-12] effectively reduced the routing overhead for e-SRAM testing. However, the test/diagnosis time is increased dramatically, which is not acceptable for production test [13-14]. [4] proposed an intelligent test wrapper for small and wide memories using the single-bit repair strategy. The key idea of this work is to embed intelligent repair analysis algorithms into the memory wrappers under a reasonable repair area overhead.

This paper presents various techniques to reduce e-SRAM test and repair overhead *from a practical point of view*. Without sacrificing at-speed testability, diagnosis capability and repairability, the improved architecture considers sharing a wrapper for multiple identical memories, sharing memory BIST controllers for e-SRAMs embedded in different functional blocks, test responses compression for wide memories, and various repair strategies for e-SRAMs with different configurations. Experimental results on two benchmark SoC chips show that the proposed strategy can significantly reduce e-SRAM test and repair overhead.

The remainder of this paper is organized as follows. In Sec. 2, we briefly review the test architecture and repair strategies proposed for SoCs containing many small e-SRAMs. Sec. 3 describes the various techniques that we propose to reduce the e-SRAM test/repair area overhead without sacrificing test quality. Next, we present the experimental results for two large industrial chips in Sec. 4. Finally, Sec. 5 concludes this paper.

2. Motivation

2.1 SoC e-SRAM Test Architecture

Today, the widely-used industrial e-SRAM test architectures (e.g., the ones in [2, 8]) are multi-level designs, as shown in

[1] This work was supported in part by the Hong Kong SAR UGC Direct Grant 2050366.

Figure 1. All the memories are wrapped for test and repair purposes and the e-SRAMs within the same functional block are clustered to be tested in parallel. The number of BIST controllers within each functional block depends on the different memory port types. For example, two BIST controllers are required if the functional block contains both 1-port memories and 2-port register files. The BIST controller and the memory wrappers communicate with each other through IEEE Std. 1500 serial links [15]. At the system level, these BIST controllers receive test primitives through a JTAG/1500 bridge from JTAG bus, where they decode and execute test algorithms and control the memory wrappers to generate appropriate test patterns (both address and data).

With each memory equipped with a dedicated wrapper and each functional block provided with one or more BIST controllers, the design for test (DFT) area overhead in this general architecture can be quite large when the SoC contains many small e-SRAMs. In practice, however, we are able to improve the general architecture without sacrificing test quality from various aspects. First of all, a major portion of the BIST area is the dedicated wrapper for each e-SRAM. Many SoCs, especially graphics and networking chips, contain lots of identical e-SRAMs (same type, width and size). Sharing a single wrapper for these memories can significantly reduce BIST overhead (e.g., [2, 3]). At the same time, however, we need to share the wrapper intelligently in order not to sacrifice at-speed testability, which was not addressed in previous work. Secondly, for small and wide e-SRAMs, directly storing the entire test responses all at once for diagnosis purpose consumes large silicon area. The bit test results can be compressed in the temporal dimension without losing diagnosis capability. Finally, it is also possible to share BIST controllers among multiple functional blocks when they are physically close to each other to further reduce BIST area overhead.

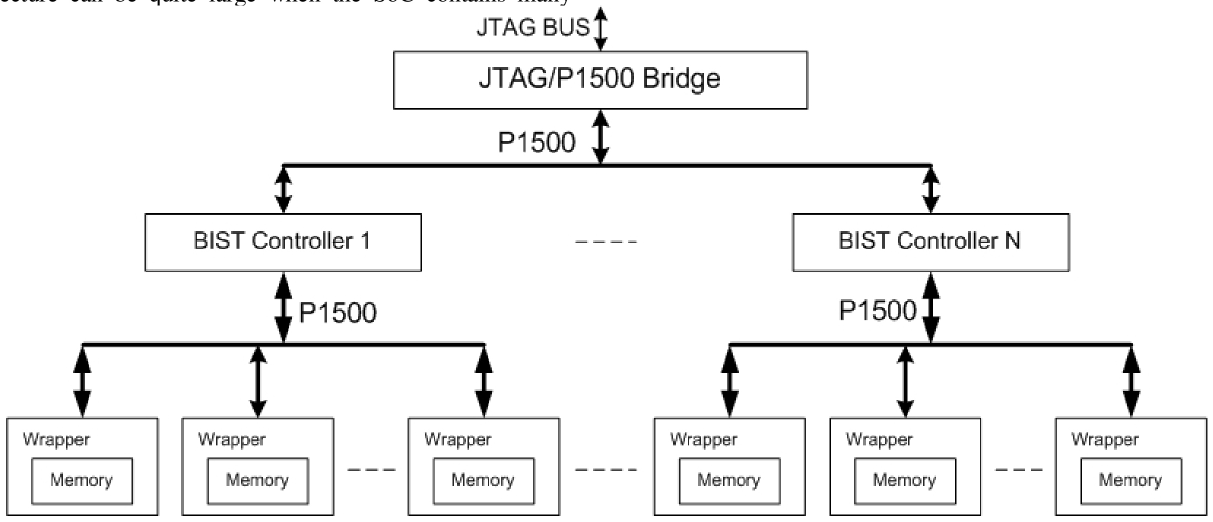


Figure 1. The General Test Architecture

2.2 Small e-SRAM Repair Strategies

Since the repair overhead is quite high for small e-SRAMs and the possibility of a particular small e-SRAM being defective is usually low, traditionally Built-in Self-Repair (BISR) circuitry are not utilized for them. However, as shown in [4], it is necessary to provide repairability for these small e-SRAMs when their total sizes amount to the Mbits range. In [16], a specific repair method is selected for all e-SRAMs independently of their sizes. This technique is especially not economical for small e-SRAMs due to its complexity. In [4], a "smart" wrapper is proposed for small and wide memories with single bit or single input/output (I/O) repair schemes, which allows at-speed testing with an acceptable BISR overhead. For e-SRAMs with medium or small number of I/Os, however, single row or single column repair scheme would be more appropriate in terms of BISR area. This is not considered in [4].

It is the above observations in traditional e-SRAM BIST architecture and repair strategies that motivate us to take various practical issues into consideration in designing our proposed e-SRAM BIST /R architecture with reduced area overhead without sacrificing their test/diagnosis capability, as shown in the following section.

3. e-SRAM Test/Repair Area Reduction

3.1 Simplifying e-SRAM Test Architecture Considering Physical Information

3.1.1 Partly Share Wrappers for Identical Memories

It is quite common to have many identical memories on-chip for today's SoC designs, especially for graphical and networking chips. Several previous works have proposed to share test wrapper for these identical memories to reduce test area overhead (e.g., [2, 3]), however, blindly sharing wrapper for all identical memories may lead to physical design difficulty and/or sacrifice for e-SRAMs' at-speed testability. Therefore, we need to consider the following two important rules in practice. The first one is that the layout flexibility for each functional block cannot be suffered. In [3], a single wrapper is designed to cover all the identical e-SRAMs, even though they might be in different functional blocks. Although this method results in minimum DFT area overhead, it significantly increases the layout difficulty of the design. For the at-speed testability of the e-SRAMs, let us take a close look at a typical e-SRAM wrapper (shown in Figure 2) that supports the application of the widely-used March Algorithms [17]. The critical blocks inside the wrapper that need to operate at-speed are the address generator and the comparator, because the test patterns and test responses need to be applied/captured at-speed. For most of existing March algorithms where data background is not changing every cycle, however, there are enough time for the preparation of data patterns and the associated commands before a new March element starts. As a result, the data generator and the command generator do not need to work at-speed. In [2], the timing-critical address generator block is also shared for identical memories and hence it is not applicable for testing high-speed e-SRAMs today [19].

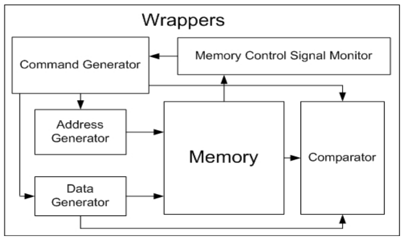


Figure 2. A Wrapper Enabling At-Speed BIST

Based on the above analysis, we propose to share the data generator and the command generator only for identical e-SRAMs within a functional block. The memory control signal monitor block is used to diagnose the read or write enable signals for each memory in case of failures. Therefore, this block is specific for each memory and cannot be shared. Fortunately, it is very small (several combinational gates and one flip-flop), and hence does not affect our simplification. In summary, within a functional block, each identical memory will have its own address generator, response comparator and memory control signal monitor while the data generator and the command generator will be shared among all identical memories in our BIST architecture.

The proposed test architecture with partly shared wrapper is shown in Figure 3. As can be seen, for those non-identical memories within each functional block and identical memories for different blocks, their wrappers are maintained. Experimental results for the benchmark chips show each wrapper area of those identical memories can be reduced by around 50%.

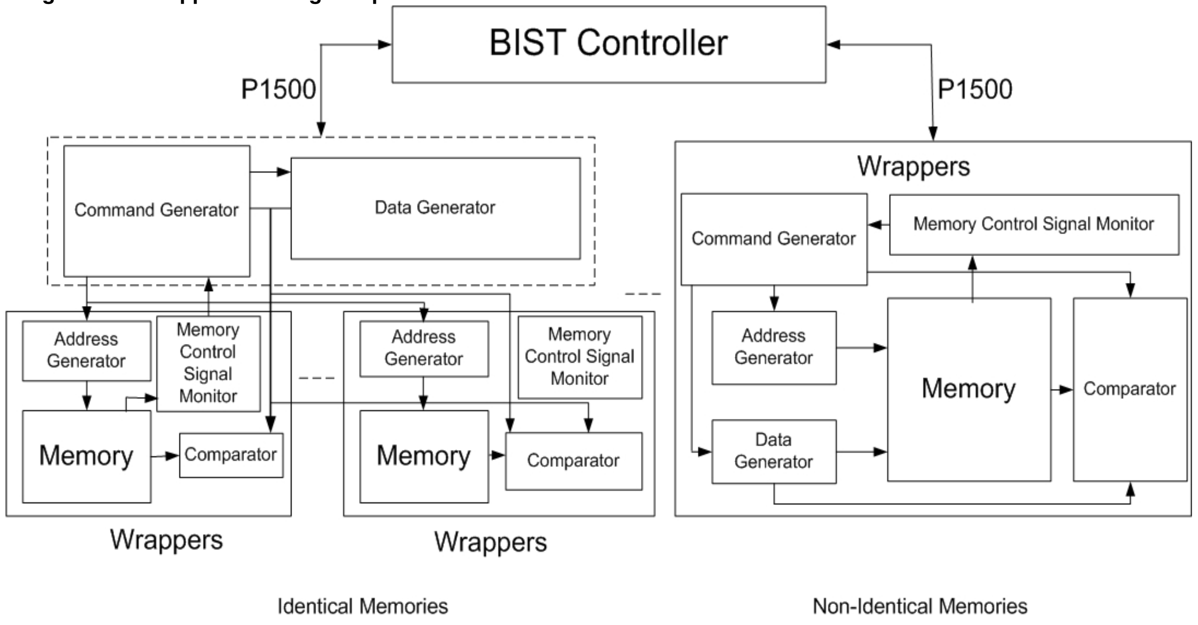


Figure 3. Proposed Wrappers for Identical Memories

3.1.2 Minimize the Number of BIST Controllers

Since the memory BIST controllers are generally implemented at the register transfer level (RTL), every functional block with e-SRAMs contains one or more BIST controllers. After entering the gate level design stage, the physical information of those functional blocks is known and can be utilized to reduce DFT area overhead. That is, for the functional blocks that are close to each other in floorplan, the memories with the same port type (e.g., single-port memories or two-port register files) can share a single BIST controller. It is important to note, however, reducing the number of BIST controllers by this way might cause routing congestion. Therefore, a try-and-error check needs to be done after grouping BIST controllers. The proposed flow for sharing BIST controllers is shown in Figure 4.

From Figure 4, the main difference between the general method and the proposed method is that we utilize the physical information of the functional blocks (including their memories) when designing BIST controllers. The procedure is as follows, we start by grouping some of the nearby functional blocks and re-create a single BIST controller for this group at RTL if their memories belong to the same type. Next, we check whether routing congestion occurs and the timing of the original design is violated. If not, we try to further grouping them in the same way.

After this round of groupings, the BIST controllers' number usually can be reduced significantly. Experimental results on our benchmark chips show that the DFT area overhead after reducing the BIST controllers' number from 24 to 16 is up to 0.9%.

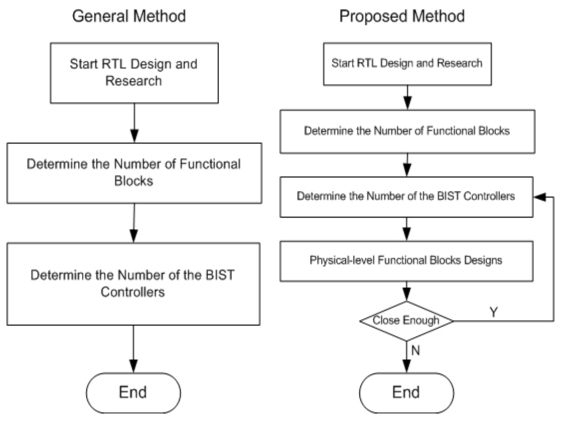
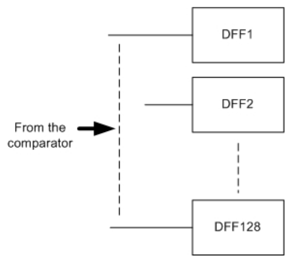

General Method
Start RTL Design and Research
Determine the Number of Functional Blocks
Determine the Number of the BIST Controllers
End
Proposed Method
Start RTL Design and Research
Determine the Number of Functional Blocks
Determine the Number of the BIST Controllers
Physical-level Functional Blocks Designs
Close Enough
Y
N
End

Figure 4. BIST Controller Number Minimization Flow

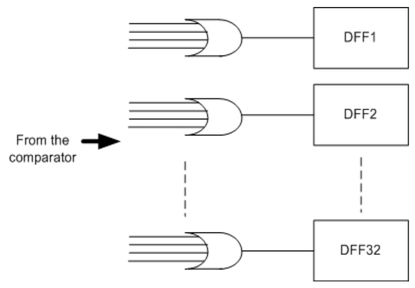
It may be argued that this minimization flow will increase design cycle due to multiple times of synthesis, floorplaning and routing. However, the design portion which requires several rounds of try-and-error checks are only limited to those controllers. This is because that all e-SRAM wrappers will be maintained during this flow.

3.2 Test Results Compression w/o Losing Diagnosis Capability

For small wide memories, storing every bit comparison results consumes quite a large amount of silicon area (as shown in Figure 5(a) for a 128×128 e-SRAM). Compressing them by ORing each bit comparison result into a single-bit test signature or multiple-bit test signature is proved to be a cost-effective approach [20], e.g., compressing each bit comparison result with a 4 to 1 OR gate is shown in Figure 5(b). However, this approach fails to provide full diagnosis resolution of each e-SRAM bit.



(a). Storing Each Bit Information with Diagnosis Capability



(b) Compressing Bit Information w/o Diagnosis Capability

Figure 5. Bit Comparison Results Storing Techniques

To tackle the limitations of both above techniques, we propose to compress the test signature in a temporal dimension. That is, we use the same design as in Figure 5(b), but at the same time we provide more data patterns for diagnosis purpose. For the example of a 128×128 configuration with 4:1 bit compression ratio, a total of 16 kinds of data patterns at most are required in diagnosing each bit status. Since the general data pattern register is 2-bit to cover solid, checkerboard data patterns and their reversed ones, the proposed method only requires increasing the data pattern register from 2 bits to 4 bits and increasing the command register with two extra bits in the wrapper. This 4 bit increase of the data pattern register and the command register for a 128×128 e-SRAM is negligible. The only drawback of this method is the diagnosis time of this memory has been increased by 4 times. Fortunately, not all memories are required to be diagnosed.

With the proposed method, the bit comparison results storing units can be reduced up to 65% for the example in Figure 5.

3.3 Single-Element Repair Strategies

Numerous redundancy repair strategies have been proposed in the literature, ranging from simple to highly complex. For small e-SRAMs, single redundant element repair schemes are considered the best approach due to their small area overhead, where an element can be a single bit/IO, a single row or a single column. In order to select different repair approaches for small e-SRAMs with different physical architectures, the following terminology is defined.

Single Row Repair: this approach is similar to the general row repair method, i.e., using a spare row to replace a faulty row. However, if more than one rows are faulty, the memory is unrepairable. It should be pointed out that a single spare row is capable of repairing multiple faulty cells as long as all the faulty cells are on the same row.

Single Column Repair: similar to the general column repair method, one but only one spare column is designed to repair a faulty column. If more than one column are faulty, the memory is unrepairable.

Single Bit Repair: this repair scheme can tolerate up to a single faulty bit. If there exist more than one faulty bit, the memory cannot be repaired.

Based on the single element repair method, an important factor for determining the repair approach for an e-SRAM is the spare element overhead. For a small e-SRAM organized as $A(R,C)\times B$ bits, where *A, B, R, and C* represent the address space, the number of I/Os, the number of rows and the number of columns of the e-SRAM, respectively, the BISR overhead when using the three approaches above is shown in Table 1.

Table 1. Repair Area Overhead Calculations

Single Element Repair	Repair Area Overhead
Single Row Repair	*1/R*
Single Column Repair	*1/C*
Single Bit Repair	1/*B*

From Table 1, it can be derived that the largest number of a small e-SRAM rows, columns and bits/IOs determines the optimal redundancy method.

According to the repair overhead calculated above, a general repair approach selection flow for small e-SRAMs is shown in Figure 6.

Since the Single Bit repair approach has been presented and demonstrated in [4], this paper only discusses the implementation of the Single Row repair and the Single Column Repair. Here, we select the Single Row repair approach as the example to explain our proposal.

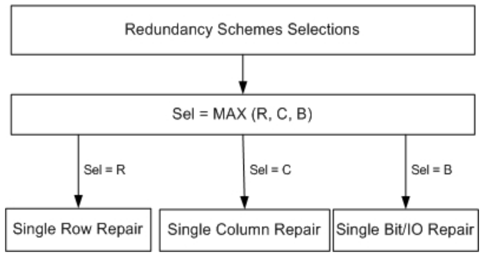
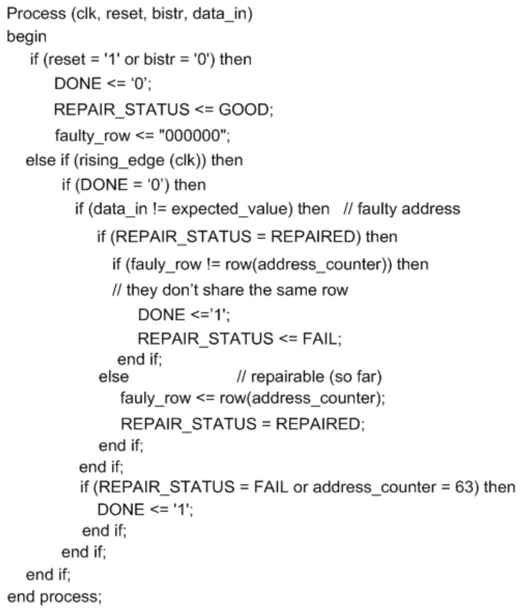


Figure 6. Redundancy Approach Selections



```
Process (clk, reset, bistr, data_in)
begin
  if (reset = '1' or bistr = '0') then
    DONE <= '0';
    REPAIR_STATUS <= GOOD;
    faulty_row <= "000000";
  else if (rising_edge (clk)) then
     if (DONE = '0') then
       if (data_in != expected_value) then   // faulty address
         if (REPAIR_STATUS = REPAIRED) then
           if (fauly_row != row(address_counter)) then
           // they don't share the same row
             DONE <='1';
             REPAIR_STATUS <= FAIL;
           end if;
         else                  // repairable (so far)
           fauly_row <= row(address_counter);
           REPAIR_STATUS = REPAIRED;
         end if;
       end if;
       if (REPAIR_STATUS = FAIL or address_counter = 63) then
         DONE <= '1';
       end if;
     end if;
  end if;
end process;
```

Figure 7. Repair analysis algorithm for e-SRAMs with single-row repair and 64 addresses space

When a memory with single row or column repair fails, we need to know two pieces of information in order to determine whether the memory is repairable and how to repair it, i.e., the row and column numbers. For example, for an e-SRAM with single row repair, if all failures share the same row address, we declare memory as repairable. Otherwise, it is considered as unrepairable. If repairable, the test/repair analyzer will provide the bad row or column address.

The RTL implementation for a memory with 64 row addresses space using single row repair is shown in Figure 7. The RTL code for single-column repair is similar.

Compared to the single bit test/repair analyzer in [4], the hardware costs for the single row repair analyzer are as follows. Assuming R and B to be the number of rows and bits in the memory, respectively, we estimate the hardware cost in terms of flip-flops (FFs), 2-input XOR gates and 2-input OR gates by considering the signals used in Figure 7. The detailed analysis results are shown in Table 2, where the signal address_counter is an overflow flag from the address generator block.

Table 2. Gates counts of the single row repair analyzer

Related signals	FFs	XORs	ORs
DONE	1	1	0
Data_in	0	B	B-1
REPAIR_STATUS	2	4	2
faulty_row	$\mathrm{Log}_2 R$	$\mathrm{Log}_2 R\text{-}1$	$\mathrm{Log}_2 R\text{-}1$
address_counter	0	1	0
Total gate counts	$\mathrm{Log}_2 R + 3$	$\mathrm{Log}_2 R + B + 5$	$\mathrm{Log}_2 R + B$

For e-SRAMs with the single row repair approach, B tends to be small but R is relatively large. Therefore, the total gates count for the single row repair analyzer is low. For example, a wrapper for an e-SRAM with R = 64 and B = 16 with a single spare row requires only 58 standard cells.

In summary, while the single-element repair strategies proposed in [4] is effectively for small wide e-SRAMs only, our proposed repair analysis circuitry embedded in the wrappers is area-efficient for all kinds of small e-SRAMs (wide, medium and narrow) while still enabling good repair quality.

4. Experimental Results

To quantify the benefits of our proposed diverse approaches for reducing e-SRAM BIST/R area overhead, we choose two benchmark chips [21] for demonstration, where chip 1 has 256 unrepaired e-SRAMs and chip 2 has 2574 repaired e-SRAMs with soft repair configuration. The configurations of these two chips are shown in Table 3 and Table 4, where the test/repair area overhead is calculated by comparing the total area of the DFT logic with the total area of the memories.

In order to demonstrate the contributions in terms of test area reduction for each approach presented in Section 3, we record the result after applying each technique. The total area savings which are calculated by applying all the proposed approaches are also shown in the table. The results for chip 1 are shown in Table 5. From this table, we can observe that sharing wrapper technique has the most significant impact on test area overhead reduction. This is expected because the graphical chips utilized in this experiment contain lots of identical memories inside.

For chip 2, all memories were previously repaired without detailed repair analysis. Therefore, the area overhead is rather large. However, according to [4], the yield for those small e-SRAMs when not using the single-element repair strategies is not improved significantly. Therefore, besides the low area overhead, the single-element repair strategies also provide comparable yield level when compared with the general method, e.g., the one in [16]. In this experiment, we replace their general repair processing method with our single-element repair strategies. The test/repair area overhead reduction due to this replacement is shown in Table 6.

Table 3. A Summary of a Benchmark Chip 1

Total e-SRAM amount	Total 1-port e-SRAM amount	Total 2-port e-SRAM amount
256	30	226
Maximum 1-port e-SRAM density	Maximum 2-port e-SRAM density	Total e-SRAM density
170Kbits	32Kbits	1.735Mbits
Total e-SRAM Test Area Overhead	Total 1-port e-SRAM test area overhead	Total 2-port e-SRAM test area overhead
9.98%	11.41%	8.81%

Table 4. A Summary of a Benchmark Chip 2

Total e-SRAM amount	Total 1-port e-SRAM amount	Total 2-port e-SRAM amount
2574	110	2464
Maximum 1-port e-SRAM density	Maixmum 2-port e-SRAM density	Total e-SRAM density
120Kbits	16Kbits	3.0Mbits
Total e-SRAM Test Area Overhead	Total 1-port e-SRAM test area overhead	Total 2-port e-SRAM test area overhead
19.98%	25.23%	16.44%

Table 5. Test Area Overhead Reduction Summary for Chip 1

Approaches	Reduction %
Designing partly shared wrappers	2.1%
Minimizing the number of BIST controllers	0.9%
Test signature compression without losing diagnosis capability	0.85%
All combined	3.85%

Table 6. Test/Repair Area Overhead Reduction Summary for Chip 2

Approaches	Reduction %
Designing partly shared wrappers	5.1%
Minimizing the number of BIST controllers	2.2%
Compressing test signature without losing diagnosis capability	0.8%
Applying single-element repair strategies	2.2%
All combined	10.3%

5. Conclusion

Today's SoCs contain an increasing number of small e-SRAMs. Reducing the test and repair overhead for them is a challenging task. This paper presented diverse approaches to tackle this problem from a practical point of view, including partly sharing wrappers for identical e-SRAMs within a single functional block, minimizing the number of BIST controllers for nearby functional blocks, compressing wide memory bit test results without losing diagnosis capability and various single-element repair strategies. Experimental results show that the proposed techniques significantly reduced e-SRAM test/repair area overhead.

6. Acknowledgements

We wish to thank anonymous reviewers at ATI Technologies Inc, for their insightful comments and suggestions on this work.

7. References

[1] A. Bommireddy, et al., "Test and debug of networking SoCs – a case study", *Proc. VTS*, pp. 121-126, 2000
[2] A. Benso et al., "Programmable built-in self-testing of embedded RAM clusters in a system-on-chip architectures", *IEEE Communications Magazine*, Vol. 41, No. 9 , 2003, pp. 90-97.
[3] B. H Fang and N. Nicolici, "Power-constrained embedded memory BIST architecture", *Proc. DFT*, pp. 451-458, Nov. 2003
[4] R. C. Aitken, "A modular wrapper enabling high speed BIST and repair for small wide memories", *Proc. ITC*, pp. 997-1005, 2004
[5] W. L. Wang, "March based memory core test scheduling for SoC", *Proc. ATS*, pp. 248-253, 2004
[6] B. H. Fang, Q. Xu and N. Nicolici, "Hardware/software co-testing of embedded memories in complex SoCs", *Proc. ICCAD*, pp. 599-605, 2003
[7] M. Lobetti Bodoni et al., "An effective distributed BIST architecture for RAMs", *Proc. ETS*, pp. 119-124, 2000
[8] C. W. Wang et al., "A built-in self-test and self-diagnosis scheme for heterogeneous SRAM clusters", *Proc. ATS*, pp. 103-108, 2001
[9] B. Nadeau-Dostie, et al., "A serial interfacing technique for built-in and external testing of embedded memories", *Proc. CICC*, pp. 22.2/1- 22.2/5, 1989
[10] B. Nadeau-Dostie, et al., "Serial interfacing for embedded-memory testing", *IEEE Design&Test*, Vol. 7, No. 2, April 1990, pp. 52 -63.
[11] W. B. Jone, D. C. Huang and S. R. Das, "An efficient BIST method for non-traditional faults of embedded memory arrays", *IEEE TIM*, Vol. 52, No. 5, Oct. 2003, pp. 1381-1390.
[12] D. C. Huang and W. B. Jone, "A parallel built-in self-diagnostic method for embedded memory arrays", *IEEE TCAD*, Vol. 21, Issue 4, 2002, pp. 449-465.
[13] B. Wang, Y. Wu and A. Ivanov, "Designs for Reducing Test Time of Distributed Small Embedded SRAMs", *Proc. DFT*, pp. 120-128, 2004
[14] B. Wang, Y. Wu and A. Ivanov, "A Fast Diagnosis Scheme for Distributed Small Embedded SRAMs", *Proc. DATE,* pp. 852-857, 2005
[15] IEEE Std. 1500, IEEE Standard for Embedded Core Test – IEEE Std. 1500-2004. IEEE, New York, 2004
[16] C. L. Su, et al., "A processor-based built-in self-repair design for embedded memories", *Proc. ATS*, pp. 366-371, 2003
[17] A. J. Van De Goor, "Using march tests to test SRAMs", *IEEE Design&Test*, Vol. 10, No. 1, March 1993, pp. 8-14
[18] K. Zarrineh and S. J. Upadhyaya, "On Programmable Memory Built-in Self Test Architectures", *Proc. DATE*, pp. 708-713, 1999
[19] T. J. Powell et al., "BIST for deep submicron asic memories with high performance application", *Proc. ITC*, pp. 386-392, 2003
[20] J. T. Chen et al., "Test response compression and bitmap encoding for embedded memories in manufacturing process monitoring", *Proc. ITC*, pp. 258-267, 2001
[21] N. Murthy, F. Hering and J. Rom, "Embedded memory test & repair drives higher yield in nanometer technologies", *Proc. VTS*, 2005