A New BIST for Diagnosis and Transparent Testing in High Density SRAMs

Myung-Hoon Yang, Jong-Cheol Lee, Yong-Seok Kang and Sungho Kang

Department of Electrical Engineering, Yonsei University
Shinchon-Dong 134, Seodaemun-Gu, Seoul, 120-749 KOREA
Phone: +82-2-361-2775, Fax: +82-2-313-8053, E-MAIL: shkang@bubble.yonsei.ac.kr

Abstract—**In this paper we present a new BIST(Buit-In Self Test) for high density static random-access memories(SRAMs). The proposed approach is implemented based on parallel testing in order to access multiple cells simultaneously. Since transparent testing can preserve initial contesnts of RAMs, it is very efficient for periodic testing. But transparent testing is not suitable for fabrication testing, because it requires longer testing time than non-transparent testing. Thus non-transparent testing is used to decrease test time for fabrication testing, and transparent testing is performed only for periodic testing.**

1 Introduction

With the progress of VLSI(very large scale integrated circuits) fabrication technology the size of RAMs has increased rapidly and, memory testing becomes more crucial in the large capacity memory chips. Goor [1] suggested the relevant march elements for each functional fault model and showed the effectiveness of the march tests. However, each march test is complex and requires a very large test time. As a result, the conventional march tests are no longer adequate for the large size memories.

The limited reliability of large high-density memory chips requires periodic field testing. In some applications the components of a system must be tested periodically without affecting the memory contents. Preservation of the RAM contents after testing can be achived by transparent testing. Several innovative transparent testing algorithms for RAMs have been developed [2, 3]. These algorithms can be used fot both fabrication testing and periodic testing. Since transparent testing requires additional operation to generate signature, it results in an increase of the test cost and time.

In order to reduce the test time memory architecture is modified so that multiple cells can be accessed simultaneously and tested in parallel. In this paper, a new SRAM BIST with transparent and non-transparent testing has been proposed. A detected fault in non-transparent testing can be located and transparent testing is possible in periodic testing.

2 Fault Models

2.1 Faults in the memory cell array

Many different faults can occur in a memory cell array[4,5]. The faults can be categorized as follows.

1. In a *stuck-at fault*(SAF), the logic value of a stuck-at cell or line is always 0 (SA0) or always 1 (SA1).
2. A cell with a *transition fault*(TF) fails to undergo the transitions $0 \rightarrow 1$ or $1 \rightarrow 0$.
3. A cell, say cell i, is said to be *state coupled* to another cell, say j, if cell i is fixed at a certain value x only if cell j is in one defined state y.
4. An *inversion coupling fault*(CFin) is defined as follows : An transition in one cell, say j inverts the contents of a second cell i.
5. An *idempotent fault*(CFid) involves two cells i and j. The fault is sensitized by a transition write operation to a cell j, which forces the contents of another cell i to a fixed value.
6. A *dynamic coupling fault*(CFdyn) is sensitized by a read or a write operation on one cell j, which forces the contents of a second cell i to a certain value.

2.2 Faults in the address decoder

Address decoder faults(AFs) represents faults in the address decoder. A general fault model for the address decoder was presented by Nair, Thatte, and Abraham [6]. The tests which can detect the faults in the memory cell array can detect AFs too, since the AFs looks like the faults in memory cell array.

2.3 Faults in the read/write logic

Some output lines of the sense amplifier logic or write driver logic may be SA0 or SA1. In either case, this fault can be considered as the same as SA0 or SA1 fault in the memory cell array [6,7]. The data lines may have shorts or capacitive coupling between them. These failures can be visualized as coupling between the memory cells that correspond to the coupled data lines [6,7].

3 Test Methodology

In order to test memories in parallel, it is necessary to modify the address decoder. With the modified decoder, a write operation results in parallel writing a logic value to all the locations selected by the decoder. In non-transparent testing mode the parallel comparator, which does not require a priori known reference value, is used to determine whether all the accessed cells have the identical content. And in transparent testing mode the output response verification will be performed by an output compaction block. This block can be an MISR(Multiple Input Signature Register) or any other compaction scheme.

3.1 Modified Decoder

The modified address decoder has two input words; a normal address word and a mask address word. The mask address word allows one to access multiple cells at a time. The exact operation of the decoder is described in [8]. In order to enable these multiple selections, a mask address decoder must be implemented such that a value of 1 at a bit position in the mask word masks the corresponding bit position in the address word and that at this bit position both values of the address bit are valid. Fig. 1 shows the added circuits for modification for address masking. *Mn* and *An* are a mask address input and a normal address input, respectively.

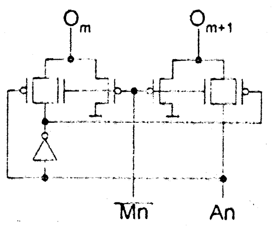

Figure 1: The Added Circuits for Modification

3.2 Parallel Comparator

A parallel comparator is shown in Fig. 2. The parallel comparator monitors the outputs of the sense amplifiers of the selected cells by comparing the output values. When any selected bit line has a different value from the other bit lines in the selected group, the output of the circuit will be 0 indicating the occurrence of a fault.

4 Test Algorithm

Assume that the memory cell array is a $\sqrt{N} \times \sqrt{N}$ two-dimensional array. The memory plane can be

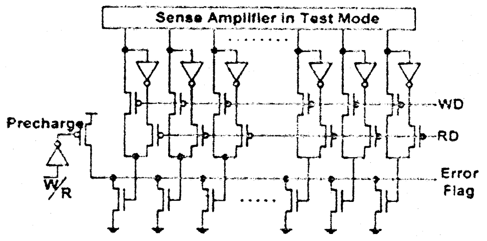

Figure 2: Parallel Comparator

thought of as a set of small blocks. Let the block consist of $\sqrt{k} \times \sqrt{k}$. With the mask address decoder the cells in the same position in each block can be accessed simultaneously. Let this block be a *basic marching block*. *March C−* algorithm shown in Fig. 3 is adopted to the proposed parallel BIST scheme. While *March C−* requires $10n$ operations and detects all SAFs, all AFs, all TFs and all CFs [1,4,9], it requires large test time for testing large memories. With the parallel test scheme, the test time can be reduced.

⇑(w0);	⇑(r0,w1);	⇑(r1,w0);	⇓(r0,w1);	⇓(r1,w0);	⇓(r0)
M_1	M_2	M_3	M_4	M_5	M_6

Figure 3: *March C−* Algorithm

4.1 Non-Transparent Algorithm

In test mode, the write operation can be carried out by increasing the normal address with the mask address fixed; all the mask addresses are 1's. The cells at the same position in each block are written simultaneously with the value 0 or 1. Thus the operation required for one write operation of *MarchC−* is the same as the number of cells in the basic marching block, k. The total number of operations required for the *MarchC−* is $5k$. The read operation requires more accesses to the memory arrays. In the test scheme, only the cells sharing the same word line can be read simultaneously. Thus the row normal addresses are not masked and the number of read operations are $\sqrt{N/k}$ times the number of write operations. $\sqrt{N/k}$ is the number of the row of the block. As a result the total operations required for *MarchC−* become $5\sqrt{k}(\sqrt{k}+\sqrt{N})$. The complexity would be varied by the variable k.

The variable k also affects the fault coverage. In case of the AFs and CFs the fault coverage is slightly lower than that of *March C−*. This problem arises because the coupling between the cells of the same position in each block. Therefore the fault coverage is reduced to 100(1-1/k)%. However if the variable k is large enough then the fault coverage would be above 99.9%.

When the comparator in Fig 2 detects a fault in a read operation, the *Error Flag* becomes low and the BIST control logic performs diagnosis. As mentioned in the last section, only the cells in the same position in each marching block are read at a time. Thus the fault location process is performed for the cells which are read at that time. When a fault is detected, the mask row address word is all 0's and the row address generator holds the corresponding address. Because the row address generator already holds the row address of the faulty cell, in order to determine the exact address of the faulty cell, the BIST circuitry needs to decide which column the faulty cell is. And then the comparator compares the data read and decide whether the faulty cell is selected or not. If the faulty cell is not selected, the generator again changes the address to read the other half and divides the column again. In this way the number of cells which are accessed is become smaller and finally the generator can determine the exact address of the faulty cell.

4.2 Transparent Algorithm

In oder to transform a conventional test algorithm into a transparent one, we will use Nicolaidis' transformation rules[2]. Let ALin be the initial algorithm. If this rule are applied for *March C−* algorithm, we get transparent *March C−* algorithm and Signature Generation Sequence as shown in Fig. 4 and Fig. 5. While the original *March C−* has $10n$ operations, the transformed algorithm has $14n$ operations. Therefore the transformed transparent algorithm is suitable for periodic testing but is not suitable for fabrication testing. In the transparent *March C−* algorithm is equal to *March C−* algorithm except for test data and additional signature generation sequence. In the proposed BIST, to implement BIST architecture for transparent testing and non-transparent testing with small hardware overhead, the same method of parallel test is applied to transparent testing.

$$\underset{M_1}{\Uparrow(r\bar{a_i}, wa_i)};\quad \underset{M_2}{\Uparrow(ra_i, w\bar{a_i})};\quad \underset{M_3}{\Downarrow(r\bar{a_i}, wa_i)};\quad \underset{M_4}{\Downarrow(ra_i, w\bar{a_i})};\quad \underset{M_5}{\Downarrow(r\bar{a_i})}$$

Figure 4: Transparent *March C−* Algorithm

$$\underset{M_1'}{\Uparrow(r\bar{a_i})};\quad \underset{M_2'}{\Uparrow(ra_i)};\quad \underset{M_3'}{\Downarrow(r\bar{a_i})};\quad \underset{M_4'}{\Downarrow(ra_i)};\quad \underset{M_5'}{\Downarrow(r\bar{a_i})}$$

Figure 5: Signature Generation Sequence

5 BIST Architecture

To distinguish the normal address bits($log_2 N$) and the masked address bits $1/2(log_2(N/k))$, two up/down synchronous counters, *BMBAG* and *MdAG*, shown in Fig. 6 and Fig. 7 respectively, are used.

In the test mode(TM), $1/2(log_2(N/k))$ bits of the column address bits are always masked and the same number of bits of the row address are not masked in read operations, so *MdAG* generates these row address

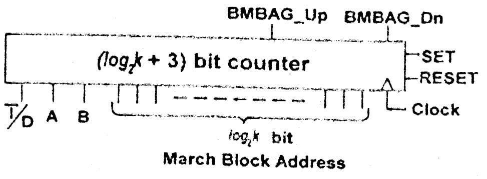

Figure 6: Basic March Block Address Generator

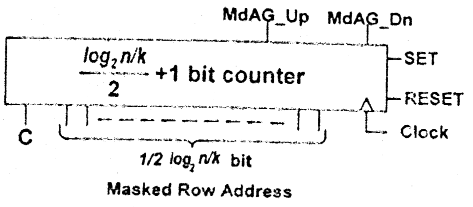

Figure 7: Masked Row Address Generator

bits for read operations in the test mode. This approach using two counters solves the problem of the difference between the number of read and write operations which could result in a complex control logic in case of using only one counter. In Fig. 6 and Fig. 7, the added bits such as $\bar{I}/D$, A, B, and C are used to generate control signals. Only these signals and the generated clock signal are needed with the independence of the memory size. These signals indicate that which march element to be performed, that the current operation should be a read operation or a write operation, and that the next address to be a increased value or a decreased value. All mask column address bits(MC) are always 1's in the test mode to mask $1/2(log_2(N/k))$ bits of the column address.

The operations in each march element of the proposed BIST architecture are as follows. As an example, the procedure to test $1Mbit$ SRAM which consists of $64 \times 64Kbit$ basic march blocks.

As soon as TM is asserted, all BIST circuits are reset. Then, $M1$ march element is performed in parallel. Since *BMBAG_Up* signal is enabled, *BMBAG* would generate log_2k bit address and all bits of the mask address, mask row address bitsMR and MC, are 1's. 64 cells are written to 0 simultaneously in one write operation. This procedure is repeated until signal B of *BMBAG* is not 0. When the signal B is 1, $M2$ march element is applied. Signal *BMBAG_Up* is disabled and *MdAG* starts counting up by signal *MdAG_Up*. If the basic march block is $64Kbit$, 8 read operations are required to read 64 cells because only the bits of MC is value of 1's and the bits of MR are 0's.

After all read operations are finished and signal C is 1, one write operation is applied and then *BMBAG* counts up by 1. These $\sqrt{N/k}$ read operations and one write operation is continued until signal A of *BMBAG* is to be 1. The operation of $M3$ march element is similar to that of the element $M2$. After $M3$ is applied completely, $BMBAG$ and $MdAG$ should count down to satisfy the address sequences of *March C−* algorithm. Therefore a stage to reset the counters is necessary. The output signal of the decoder when the input $\overline{I}/D\ A\ B\ C$ is 0110 is used as the reset signal of two counters, $BMBAG$ and $MdAG$. The march element $M4$ and $M5$ are the same as the $M2$ and $M3$ except the address counting direction. In march element $M6$, only read operations are performed. Therefore, a signal which disable write operations is generated and this signal would add the dummy read operation. This dummy operation does not affect the fault coverage.

In order to perform transparent testing with this BIST, simple modification of the BIST control logic is necessary. BIST control logic can be modified easily. Firstly, in transparent testing column mask addresses are always 0s. In signature generation sequence all write operation is excluded in transparent testing.

6 Results

In the previous sections, the parallel testing methodology, transparent testing and the BIST architecture with two test modes have been presented. By using parallel testing technique, the test time can be reduced to a great extent. Table 1 shows the number of operations required for this parallel test algorithm compared in different block sizes. And the calculated results are compared to the conventional *March C−* algorithm. The proposed parallel test method is much more faster than the conventional *March C−* algorithm. Table 2 shows the hardware overhead for BIST compared in various SRAM size and function of BIST. For any case overhead is not larger than 0.1%.

block Size	The Number of Operations			
	4M	16M	64M	256M
6K	1.39M	2.70M	5.33M	10.57M
64K	2.93M	5.57M	10.81M	21.29M
256K	6.55M	11.79M	22.28M	43.25M
1M	15.73M	26.21M	47.19M	89.13M
March C−	41.94M	167.77M	671.10M	2.68G

Table 1: The Number of Operations

Block Size	Detection	Detection+ Location	Detection+ Transparent	Detection+ Location+ Transparent
256K	0.051%	0.089%	0.067%	0.098%
1M	0.014%	0.025%	0.018%	0.026%
4M	0.004%	0.007%	0.005%	0.008%

Table 2: The Hardware Overhead of BIST

7 Conclusions

The paper has discussed an efficient technique to speed up the SRAM test algorithms with parallel transparent testing. The concept of self-testing with parallel access to multiple cells will be vital for the very large capacity single chip memory to reduce the test time. A march algorithm is performed concurrently in each basic marching block. Transparent testing is much efficient for periodic testing but not for fabrication testing since it requires long test time. Therefore in the proposed BIST non-transparent testing is performed in fabrication testing and transparent testing can be used for periodic testing. Since both transparent and non-transparent testing are performed with parallel testing, test time is very short. The proposed BIST architecture is very efficient for both fabrication testing and periodic testing. The results show that the new BIST requires very small hardware overhead.

References

[1] A. J. Goor, "Using March Tests to Test SRAMs," in *IEEE Design and Test of Computers*, March 1993, pp. 8–14.

[2] M. Nicolaidis, "Theory of Transparent BIST for RAMs," in *IEEE Trans. on Computers*, October 1996, pp. 1141–1155.

[3] M. G. Karpovsky and V. N. Yarmolik, "Transparent Memory Testing for Pattern Sensitive Faults," in *Proc. of Internation Test Conference*, 1994, pp. 860–869.

[4] A. J. Goor, *Testing Semiconductor Memories: Theory and Practice.* Singapore: John Wiley and Sons, 1995.

[5] P. Mazumder and K. Chakraborty, *Testing and Testable Design of High-Density Random-Access Memories.* USA: Kluwer Academic Publishers, 1996.

[6] R. Nair, S. M. Thatte and J. A. Abraham, "Efficient Algorithms for Testing Semiconductor Random-Access Memories," in *IEEE Trans. on Computers*, June 1978, pp. 572–576.

[7] S. Jain and C. Stroud, "Built-in Self Testing of Embedded Memories," in *IEEE Design and Test of Computers*, October 1986, pp. 27–37.

[8] Y. S. Kang, J. C. Lee and S. Kang, "Parallel BIST architecture for CAMs," in *Electronics Letters*, January 1997, pp. 30–31.

[9] R. Dekker, F. Beenker and L. Thijssen, "A Realistic Fault Model and Test Algorithm for Static Random Access Memories," in *IEEE Trans. on CAD*, June 1990, pp. 567–572.