Efficient Built-In Self-Test Algorithm for Memory

Syng-Jyan Wang and Chen-Jung Wei
Institute of Computer Science
National Chung-Hsing University
Taichung 402, Taiwan ROC
{sjwang,cjwei}@cs.nchu.edu.tw

Abstract

We present a new pseudorandom testing algorithm for the Built-In Self-Test (BIST) of DRAM. In this algorithm, test patterns are complemented to generate state-transitions that are needed for the detection of coulping faults. As a result, the number of test patterns required is less than half of the traditional method, while the extra hardware is negligible.

1. Introduction

Semiconductor memories are widely used in digital systems. Memory devices are designed with regular structures, which makes them the most densily packed devices among all types of integrated circuits (IC). Developments in VLSI technology result in a continuously increasing density of memory chips, and the number of components per chip has quadrupled every to four years. The exponential increase in density creates great challenge for memory testing. As the feature size of components shrinks, the sensitivity to faults also increases while the faults become more complex. Furthermore, test time grows at least linearly as the number of storage elements per chip increases. However, test cost can not grow at such a pace since the price per storage element drops dramatically as the density increases. Recent development in system-on-chip technology makes it possible to incorporate large embeded memory into a chip; whoever, it also complicate the test process as usually there is no direct control to the embedded memory from the outside envoriment.

Built-In Self-Test (BIST) can solve the above memory testing problems. Test patterns generated by a BIST controller can be either deterministic or pseudorandom. Deterministic algorithms (e.g., march algorithms) can achieve higher fault coverage with fewer number of test patterns; however, the circuit implementation is more complicated. On the other hand, pseudorandom testing requires more test patterns while the controller circuits are much simpler. Moreover, for some complicated fault models, pseudorandom testing is the only feasible solution. In this paper we present a new testing algorithm based on pseudorandom patterns. This method greatly reduces required number of test patterns while the area overhead is minimal. We shall also present an analytical to show the efficiency of this method.

This paper will be organized as follows. In the next section we give some preliminary information regarding memory fault models and memory testing. Our testing algorithm is presented in Section 3, and the performance of this algorithm is also analyzed in this section. We conclude the paper in Section 4.

2. Preliminary

In this section we give some preliminary information about memory testing.

2.1. Fault Models

Physical failures in dynamic random access memory (DRAM) chips are usually represented by reduced functional faults. The most widely used reduced functional fault models include stuck-at faults (SAF), transitition fault (TF), coupling fault (CF), and pattern sensitive fault (PSF) [1]. For functional testing these faults are usually considered sufficient [2],[3]. A memory cell with stuck-at-0 (s-a-0) fault can not hold logic 1, while a stuck-at-1 (s-a-1) cell can not be modified with logic 0. If a cell fails to make a 0→1 transition when it is written, it is said to contain an up transition fault. On the other hand, a cell with down transition fault can not make a 1→0 transition. With address decoder faults, either no cell is accessed for a

Acknowledgement: This work was supported in part by National Science Council under contract Number NSC-87-2215-005-001

given address, or cells that do not correspond to the given address are also accessed as well.

Coupling faults involves at least two cells. Read operations do not activate CFs, while a write operations which generates a 0→1 or a 1→0 transition in one cell changes the contents of a second cell. Such a fault is referred to as a 2-coupling fault, since only two cells are involved. In general, a k-coupling fault involves the same two cells as the 2-coupling fault and, in addition, only allows the fault to occur when another k–2 cells are in a certain state. Two special types of CFs exist: the inversion CF (CFin) and the idempotent CF (CFid). A CFin occurs when a 0→1 (or 1→0) transition in one cell inverts the contents of a second cell. Such a fault will be noted as $\langle\uparrow;\updownarrow\rangle$ (or $\langle\downarrow;\updownarrow\rangle$) since a 0→1, denoted as $\uparrow$, in the first cell (or a 1→0 transition, $\downarrow$) inverts the contents the second cell (denoted as $\updownarrow$). On the other hand, a CFid is definded as follows: An $\uparrow$ (or $\downarrow$) transition in one cell forces the contents of a second to a certain value, 0 or 1. For example, a $\uparrow$ transition in the first cell forces the second cell to state 1 is denoted as $\langle\uparrow;1\rangle$. Pattern sensitive faults occur when the contents of a cell, or the ability to change the contents, is influenced by the contents of all other cells in the memory. Therefore, k-coupling fault is a special case of PSF.

2.2. Random Testing of DRAM

Random testing of DRAM can be classified into two types. In the RARWRD (R stands for random) algorithms, all signal lines (including Address, Data, and read/write control) are applied with random patterns. On the other hand, the DADWRD (D for deterministic) algorithms apply random data only to the data lines, while the sequence of generated address and read/write control are known *a priori*. In real applications, random patterns are approximated by pseudorandom patterns that are generated by linear feedback shift registers (LFSRs). LFSRs can generate deterministic sequences that have many characteristics of random patterns. Therefore, the generated patterns are often referred to as the pseudorandom patterns. In the DADWRD algorithms, the address can be generated by either a counter or a modified LFSR. In all these algorithms, the applied patterns can be either bit-oreiented or word-oriented. In general, DADWRD algorithms required fewer test patterns than RARWRD algorithms while the required hardware costs are comarable. Therefore, we shall focus on DADWRD algorithms in this paper.

Random testing algorithms are usually analysed with Markov chain models. According to the model, one can calculate the *test length* (that is, the number of required test patterns) for a target *escape probability*. The escape probability is defined as the probability that a fault in a cell ketp undetected. The analytical results for the DADWRD algorithm are given Table I [2]. Shown in the Table are the test lengths required to achieve an escape probability is 0.001. In the Table, P_d is the probability that the generated data is 1. Therefore, $1-P_d$ is the probability that the generated data is 0. From the Table, it is easy to see that the test length for a 2-coupling fault is much larger than that of a stuck-at fault. The reason of this longer test length is that the activation condition is hard to satisfy. To detect a stuck-at-0 fault, one must write a '0' into a cell and then read back the stored value. On the other hand, to detect a CFid in between two cells, the coupled cell must be set to a given state while the coupling cell should have a state transition. We shall describe such a algorithm in next section.

If both stuck-at and coupling faults are in the fault model under consideration, then random test is certainly not a good methodology since it is biased toward one of the target fault models. Therefore, the overall test time can be reduced if we can develop an algorithm that effectively reduces the test lengths for coupling faults.

Random testing is not used as often as march algorithms in memory BIST circuits since it requires more test patterns for the same set of faults. On the other hand, pseudorandom testing has the advantage that the test pattern generator is simper and required far less silicon area. Besides, for more complicated fault models (e.g., k-coupling fault with larger k), pseudorandom testing is the only feasible method.

3. Test Algorithm and Analysis

The detection of coupling fault can be made easier if we can increase the probability of state transitions during the process of test generation. This can be achieved by complementing the test data during pattern generation. The algorithm can be written as follows for a memory of size n.

Step 1 initializes the memory with random patterns. In each iteration of step 2, each cell performs a read operation followed by three write operations. In the three write operations, the first pattern x, $x \in \{0,1\}$, is a pseudorandom pattern, while the following two operation write the complement of previous pattern into the selected cell. In this way, we force two state transitions into the selected cell (i.e., both $\uparrow$ and $\downarrow$), which increases the possibility of the detection of coupling faults.

Test Algorithm

```
1: for (i = 0; i < n−1; i++)        // initialisation
1.1    A[i] = ?;
// A[i] is the location of address i, ? ∈ {0,1}
2: for (j = 1; j <= t; j++) {       // repeat t times
       for (i = 0; i < n−1; i++) {
2.1      read A[i];
2.1      A[i] = ?;   // write a random pattern
2.3      A[i] = A[i]';
         // complement the data in address i
2.4      A[i] = A[i]';
         // complement again the data in address i
       }
   }
3: for (i = 0; i < n−1; i++)
3.1    read A[i];   // final read
```

Consider a $\langle\uparrow;1\rangle$ fault which makes cell i coupled to cell j. The Markov chain model for the detection of this fault is shown in Fig 1. In this figure, state S_U is the state in which the coupling fault has not yet been detected, while state S_D is the final state in which the coupling fault is detected. P_d is the probability that a '1' is generated and written into the coupled cell, while P_G is the probability that the required pattern appears. For a 2-coupling fault, we will have $P_G = 1$. Since cell j is applied with both $\uparrow$ and $\downarrow$ transitions in any iteration, the fault will surely be detected whenever the random pattern written into the cell i is 1.

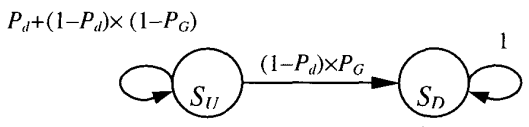

Fig. 1. Markov chain for detecting a $\langle\uparrow;1\rangle$ fault

Suppose that setp 2 of the test algorithm is conducted t times, the probability that the fault keeps undetected is equal to the probability that the chain is in state S_U:

$$P_{S_U}(t) = [P_d + (1-P_d)\times(1-P_G)]^t \qquad (1)$$

For a given escape probability E, the number of iterations (i.e., step 2) required is:

$$T(E) = \ln E \Big/ \ln[P_d + (1-P_d)\times(1-P_G)] \qquad (2)$$

Since there are four operations in each iteration (one read operation and three write operations), in all there will be $4\times T(E)+2$ operations for each cell, including the initialisation and final real operation.

The Markov chain models for the detection other idempotent CFs (i.e., $\langle\uparrow;0\rangle$, $\langle\downarrow;1\rangle$, and $\langle\downarrow;0\rangle$) can simarly obtained and are not shown here.

The detection of stuck-at faults is conceptually easy. The read operation in step 2.1 reads only the data written into a cell performed in step 2.4 in previous iteration. Since steps 2.3 and 2.4 only complement the data stored in a memory cell, the data written in step 2.4 are the same as those written in step 2.2, which are generated random patterns. Therefore, a s-a-1 (s-a-0) fault in a cell is detected when a '0' ('1') is written into it. The Markov chain model for the detection of a s-a-1 fault is shown in Fig. 2.

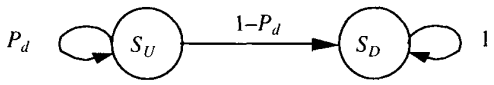

Fig. 2. Markov chain for detecting a s-a-1 fault

The Markov chain model in Fig. 2 is similar to that in Fig. 1 if we make $P_G = 1$, which is the case for 2-coupling faults. However, the data used to initialise the memory (i.e., written in step 1) can also be used to detect stuck-at faults. The probability that the model stays in state S_U after step 2 having been conducted t times is:

$$P_{S_U}(t) = P_d^{t+1} \qquad (3)$$

For a given escape probability E, the number of iterations required is:

$$T(E) = \frac{\ln E}{\ln P_d} - 1 \qquad (4)$$

The detection of an inversion CF is somewhat different. Consider a $\langle\uparrow;2\rangle$ fault which complements the content of cell i whenever there is a $\uparrow$ transition in cell j. In each iteration of step 2, at least there is one $\uparrow$ transition and one $\downarrow$ transition in cell j, which are greated by the two complement operations (steps 2.3 and 2.4). Because of the two complement operations, the data stored in a cell at the end of an iteration (step 2.3) is the same as the random pattern generated in step 2.1. If the random pattern generated is different from the one generated in previous iteration, there will be one more bit transition (either $\uparrow$ or $\downarrow$).

The Markov chain model for the detection of a $\langle\uparrow;2\rangle$ fault is shown in Fig 3. State S_0 is the state in which the random pattern written into the coupling cell is a '0', and S_1 represents the state in which the random pattern written into the coupling cell is a '1'. State S_D is the final state in which the coupling fault is detected.

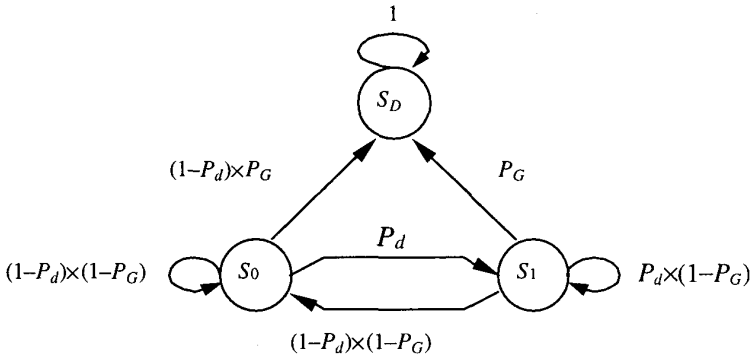

Fig. 3. Markov chain model for the detection of a ⟨↑;2⟩ fault

Given that the required pattern for a k-coupling fault appears, there is only one case in which the fault is undetectable: the current state is S_0 while the next written pattern is a '1'. In this case there are two ↑ transitions occurring in the coupling cell, which creates two inversions in the coupled cell before the content of the coupled cell is read. This model can be represented by the following equations:

$$P_{S_0}(t) = (1-P_d)\times(1-P_G)\times P_{S_0}(t-1) + (1-P_d)\times(1-P_G)\times P_{S_1}(t-1) \quad (5a)$$

$$P_{S_1}(t) = P_d \times P_{S_0}(t-1) + P_d \times (1-P_G)\times P_{S_1}(t-1) \quad (5b)$$

$$P_{S_1}(t) = P_d \times P_{S_0}(t-1) + P_d \times (1-P_G)\times P_{S_1}(t-1) \quad (5c)$$

With the boundary condition:

$$P_{S_0}(0) + P_{S_1}(0) = 1$$

For a given escape probability E, we can find the required number of iterations t by finding the minimum t such that the probability of state S_D is greater than $1-E$.

4. Conclusion

The analytical results of the proposed method are given in Table II. In Table we give the minimum number of operations required on any cell in order to detect faults with an escape probability no more than 0.001. Compared with Table I, there are two things of particular interest. (1) The test lengths for SAF and 2-coupling CFid are roughly the same in our scheme, while in DADWRD algorithm the test length of SAF is much smaller than that of 2-coupling CFid. (2) For all k-coupling CFid faults, the test length of our method is about half of the length of DADWRD algorithm. The reason is obvious. In our method we include complement operations which create bit transitions that are required for the detection of coupling faults. However, these complement operations do not detect any new SAFs.

If both SAF and CFid may appear, DADWRD algorithm does not perform well since the overall test length is the larger test length for the two fault types. On the other hand, out method reduces the overall test length although it may not be as good for one of the fault model (i.e., SAF). Besides, it is easy to implement our method with insignificant extra hardware, which makes this method very attractive for memory BIST design.

REFERENCES

[1] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, John Wiley and Sons Ltd., 1991.

[2] S. M. Thatte and J. A. Abraham, "Testing semiconductor random-access memories," in *Proc. 7th Annual Intl. Conf. on Fault-Tolerant Computers*, pp. 81-87, 1977.

[3] R. Nair, et al., "Efficient algorithms for testing semiconductor random-access memories," *IEEE Trans. on Computers*, vol. 27, no. 6, pp. 572-576, June 1978.

[4] J. Savir et al., "Testing for coupled cells in random-access memories," in *Proc. Intl. Test Conf.*, pp. 439-451, 1989.

Table I. Test length coefficients for DADWRD and RARWRD algorithms

$P_a = P_w = P_d = 0.5$, $e = 0.001$

Fault	Test length	
	DADWRD	RARWRD
SAF	$20 \cdot n$	$48 \cdot n$
CFid, k=2, P_G=1	$90 \cdot n$	$228 \cdot n$
CFid, k=3, P_G=P_d	$202 \cdot n$	$449 \cdot n$
CFid, k=4, P_G=P_d^2	$424 \cdot n$	$891 \cdot n$
CFid, k=5, P_G=P_d^3	$866 \cdot n$	$1775 \cdot n$

Table II. Test length coefficients for the proposed method

$P_d = 0.5$, $e = 0.001$, $P_{S0} = P_{S1} = 0.5$

Fault	Test length
SAF	$38 \cdot n$
CFid, k=2, P_G=1	$42 \cdot n$
CFid, k=3, P_G=P_d	$102 \cdot n$
CFid, k=4, P_G=P_d^2	$210 \cdot n$
CFid, k=5, P_G=P_d^3	$434 \cdot n$
CFin, k=2, P_G=1	$10 \cdot n$
CFin, k=3, P_G= P_d	$58 \cdot n$
CFin, k=4, P_G= P_d^2	$134 \cdot n$
CFin, k=5, P_G= P_d^3	$278 \cdot n$