MEMORY TESTING BASED ON TWO-DIMENSIONAL PARITY

E. Busłowska

Technical University of Bialystok, Wiejska str. 45A, 15-333 Bialystok, POLAND
ebus@ii.pb.bialystok.pl

ABSTRACT

This paper presents the new algorithms contents independent testing approaches embedded RAM's. In this suggested approach the memory in the form of bits matrix is tested. New method based on memory testing in two-dimensional space. Algorithms calculating the memory characteristic and comparing it to the initially computed *reference memory characteristic* during the process of memory refreshing and allows to get a high level of fault coverage with low hardware overhead.

1. INTRODUCTION

High-density memories constitute an integral part in most present day micro-electronic systems [5]. In order to ensure reliability of data stored m RAMs and ROMs it is very important to be sure that there are no errors, or at least know that error has appeared. However, regardless of the chosen method of increased reliability, errors practically always occur. Therefore, it is essential to examine correct functioning of a device – this process is called testing. The major aim of testing is to achieve the highest level of reliability in the series period of time. Besides production testing, periodic maintenance testing is especially important for memories to deal with the problem of data retention, e.g. checking for data losses after power-off condition or during run-time. Moreover, frequent operations on memory contents require on-line or periodic consistency checks [5]. When memories are embedded in larger integrated circuits, such as microprocessors or digital communication devices, conventional techniques for external memory testing can no longer be applied due to the limited accessibility of the memories [5]. To solve this problem, and also to reduce the long testing, a number of theoretical and practical built-in self-test (BIST) solutions have been proposed in the past.

100% of error detection is the aim but in practice a test designed to detect all probable errors is impossible to create both from the theoretical and practical point of view. Therefore, the problem of defining the optimal testing sequence is to find a set of tests, which would guarantee possibly the highest parameter of error detection with the minimum of costs.

One of the main testing problems is the difficulty in acquiring the minimal set of testing vectors covering all, or almost all, types of possible errors. As a result, the time of calculations exponentially increases along with the increasing size of the problem. There is also a dilemma of the choice between the precise problem solving and the costs generated by the solving method. Every computer user expects his hardware and stored data to work reliably and failure-free regardless of external factors. The basis of automatically generated test algorithms is to find a set of tests allowing detecting assumed defects.

Currently used RAM and ROM testing methods require additional memory for temporary storage of data existing in the tested area of memory. There is a group of systems, in which the solution of this kind is impossible to apply. In such cases the methods that do not change the contents of the memory should be used. In contrast to the methods used so far, the reference characteristic in the suggested algorithm is calculated by dynamic adjusting, and not during periodical scanning of the entire memory. It gives an enormous gain in time and is achieved at only slightly higher hardware costs.

2. DESCRIPTION OF THE ALGORITHM

In this suggested approach the memory in the form of 0-1 matrix is tested. Its size is $r \times r$ (where $r=2^n-1$; n – is the number of address bits). Every line and column has their own defined address. Parities P_X, P_Y are defined on the base of cell values for particular columns and rows. (tabl. 1.)

Table 1. An example of memory in two-dimensional space.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	0	1	0	1	1	1	1
100	0	0	0	1	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	0	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	0	0	0	

For the given example on tabl. 1 the value of the characteristics is as follows:

$$\begin{array}{lr} & 011 \\ & \underline{100} \\ S_X= & 111 \end{array} \qquad \begin{array}{lr} & 001 \\ & 010 \\ & 101 \\ & \underline{111} \\ S_Y= & 001 \end{array} \qquad (1)$$

The memory characteristic is the concatenation of both characteristics and its form is:

$$S=S_X S_Y=111001 \qquad (2)$$

During cyclical memory refreshing, the characteristic S_T is calculated and compared to the initially computed reference memory characteristic S with the aim to pick out eventual inefficiencies.

During write operations the reference characteristic S is automatically adjusted to the new correct value.

3. Error Detection Analysis

Error detection depend upon multiplication errors.

3.1. Single Errors

Single errors are detectable and correctable. The expression $S \oplus S_T$ gives the address of joined addresses of a line and column of a faulty memory cell.

Table 2. An example of memory in two-dimensional space with single error.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	0	1	0	0	1	1	0
100	0	0	0	1	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	0	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	1	0	0	

For the given example on tabl. 1 the value of the characteristics is as follows:

$$\begin{array}{lr} & 011 \\ & 100 \\ & \underline{101} \\ S_{XT}= & 010 \end{array} \qquad \begin{array}{lr} & 001 \\ & 010 \\ & \underline{111} \\ S_{YT}= & 100 \end{array} \qquad (3)$$

$$S_T=S_{XT} S_{YT}=010100 \qquad (4)$$

$$S \oplus S_T=101101 \qquad (5)$$

3.2. Double Errors

Double errors are always detectable, because $S_T \neq S$.

Table 3. An example of memory in two-dimensional space with double error.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	0	1	0	0	1	1	0
100	0	0	0	1	0	1	0	0
011	1	1	1	1	1	1	1	1
010	0	1	0	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	1	0	1	1	1	0	0	

$$\begin{array}{lr} & 001 \\ & 011 \\ & 100 \\ & \underline{101} \\ S_{XT}= & 011 \end{array} \qquad \begin{array}{lr} & 001 \\ & 010 \\ & 011 \\ & \underline{111} \\ S_{YT}= & 111 \end{array} \qquad (6)$$

$$S_T=S_{XT} S_{YT}=011111 \qquad (7)$$

$$S \neq S_T \qquad (8)$$

3.3. Triple Errors

In the case of triple errors (and higher), there is nonzero probability of their detection. The number of undetectable triple errors for two-dimensional memory $r \times r$ is s follows:

$$N_3=3([r(r-1)]/6)^2 \qquad (9)$$

3.4. Quadruple Errors

There are six cathegories of quadruple errors according to their position in two-dimensional space

- two errors every line and column in the number of:

$$N_4^1 = C_r^2 C_r^2 \qquad (10)$$

- error occurence in one column

To define the number of these errors, it is indispensable to define variables $N_0{}'$, which describe the quantity of combinations of modulo-2 addresses leading to 0. The dependence between them is as follows:

$$N_0^i = [N^{i-1} - N_0^{i-1} - N_0^{i-2}(r-i)]/i \qquad (11)$$

$$N_4^2 = rN_0^4 \qquad (12)$$

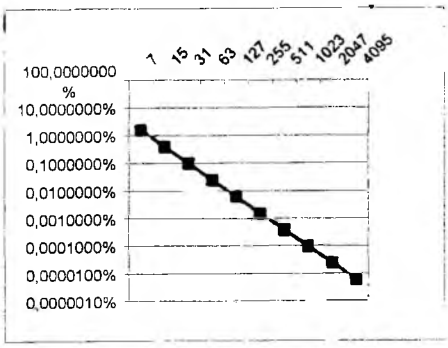

Figure 1 Probability of the occurrence of an undetectable triple error

Table 4. An example of memory in two-dimensional space with quadruple error.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	1	1	0	1	0	1	1
100	0	0	0	1	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	0	0	1	1	0	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	0	0	0	

Table 5. An example of memory in two-dimensional space with quadruple error.

	001	010	011	100	101	110	111	Py
111	0	0	1	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	0	0	0	1	1	1	1
100	0	0	1	1	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	1	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	0	0	0	

- errors in one line

$$N_4^3 = rN_0^4 \tag{13}$$

Table 6. An example of memory in two-dimensional space with quadruple error.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	0	1	1	1	1	1	0	1
100	0	0	0	1	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	0	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	0	0	0	

- every two errors in two columns

$$N_4^4 = N_0^4 C_r^2 C_4^2 \tag{14}$$

Table 7. An example of memory in two-dimensional space with quadruple error.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	0	1	0	1	1	1	1
100	0	0	0	0	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	0	0	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	0	1	0	0	0	

- every two errors in two lines

$$N_4^5 = N_0^4 C_r^2 C_4^2 \tag{15}$$

Table 8. An example of memory in two-dimensional space with quadruple error.

	001	010	011	100	101	110	111	Py
111	0	1	0	1	1	1	1	1
110	1	1	0	1	1	0	0	0
101	1	0	1	0	1	1	1	1
100	0	0	0	1	0	1	0	0
011	0	1	1	0	0	1	1	0
010	0	1	0	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	0	0	0	

- four errors at the intersection of a line and column

$$N_4^6 = 4!(N_0^4)^2 \tag{16}$$

Table 9. An example of memory in two-dimensional space with quadruple error.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	1	0
101	1	0	1	0	1	1	1	1
100	0	0	0	1	1	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	1	1	1	1	1	1
001	0	0	1	0	1	1	1	1
Px	0	0	1	1	0	0.	0	

The sum of all undetectable quadruple errors is as follows:

$$N_4 = N_4^1 + N_4^2 + N_4^3 + N_4^4 + N_4^5 + N_4^6 = ([r(r-1)]/6)^2 3! + C_r^2 C_r^2 + rN_0^4 + rN_0^4 + N_0^4 C_r^2 C_4^2 + N_0^4 C_r^2 C_4^2 + 4!(N_0^4)^2 \tag{17}$$

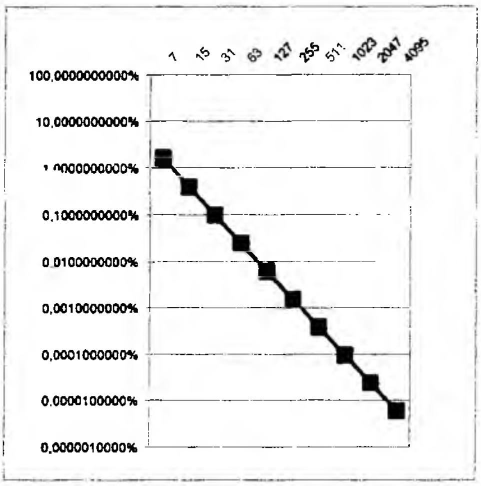

Figure 2 Probability of the occurrence of an undetectable quadruple error

3.5. Quintuple Errors

The number of undetectable quintuple errors is as follows:

$$N_5=[A^4-N^4-N^3(r-5)]/5 \qquad (18)$$

4. New Modificafion of Algorithm

Propose algorithm one can a modification to better quality.

For better efficiency let parities P_y, P_x be defined on the base of cell values for particular rows and columns, and P_{xno} be a quantities of ones in columns (tabl. 10).

Table 10. An example of memory in two-dimensional space.

	001	010	011	100	101	110	111	Py
111	0	0	0	1	1	1	0	1
110	1	1	0	1	1	0	0	0
101	1	0	1	0	1	1	1	1
100	0	0	0	1	0	1	0	0
011	0	1	1	1	1	1	1	0
010	0	1	0	1	1	1	1	1
001	0	1	1	0	1	1	1	1
Px	0	0	1	1	0	0	0	
P_{xno}	010	100	011	101	110	110	100	

Reference characteristic be define as follows:

$$S_R=S_{RX}S_{RY}S_{RX.NO,} \qquad (19)$$

where S_{RX} is the sum of modulo-2 addresses of columns, which parity equals 1 and S_{RY} is the sum of modulo-2 addresses of lines, which parity equals 1. S_{RXNO} is modulo-2 sum all P_{XNO}.

Add S_{RXNO} to previous characteristic induce detection all odd errors and decrease probability of even errors.

Calculation probabilities these errors will subject area next investigation.

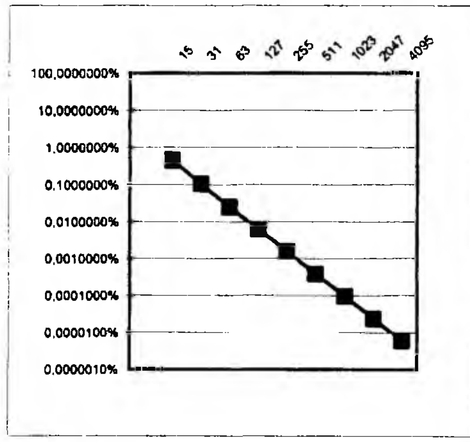

Figure 3 Probability of the occurrence of an undetectable quintuple error.

Conclusions

On the base of the conducted research, a decreased number of undetectable errors in comparison with all errors can be observed. Along with a greater number of multiplication factor the number of undetectable errors does not increase. The most significant advantage of the suggested solution is testing directly during the work of a device. It will surely be applicable in the future.

References

[1]. Goor A. J., Testing Semiconduktor memories, Theory and Practice, *John Whilley&Sons Ltd.* Chichester, 1991, p. 345.

[2]. Nicolaidis M., Transparent BIST for RAMs, Proc. *ITC'92*, Baltimore, 1992, pp. 596-607.

[3]. Siemens Data Book, *Simens: Embedded DRAM solutions,* 1998, pp.32-37.

[4]. Yarmolik V. N., Busłowska E., Testowanie Pamięci Oparte na Dwuwymiarowej Kompresji Danych, Proc. *IVth Conference Inzynieria Wiedzy I Systemy Ekspertowe*, Wrocław 2000, pp. 215-219.

[5]. Yarmolik V. N., Hellebrand S., Wunderlich H. J., Self-Adjusting Output Data Compression: An Efficient BIST Technique for RAMs; Proc. *Design, Automation and Test in Europe*, Paris, 1998, pp. 173-179.

[6]. Yarmolik V. N., Wyszyński K., Memory Testing Based on Modified Self-Adjusting Data Compression Algorithm, Proc. *6th International Conference MIXDES'99* Krakow Poland 1999; pp. 443-445.