'89 KACC 1989. 10. 27～28

A Built-In Structure for Pseudorandom Testing of RAM

Gururaj K.Rao and Hiroshi Kashiwagi

Faculty of Engineering, Kumamoto University, Kumamoto,Japan

Abstract : This paper deals with test-pattern generation and diagnoses of pattern-sensitive faults in RAM by use of simple pseudorandom M-sequences with an emphasis to built-in structure of these schemes. The problems that may arise during their implementation are discussed and an approach to built-in testing of RAM by such a scheme is given not bothering too much about the silicon area required.

1. Introduction

Random-Access Memories (RAM) can be tested for some of their faults with very simple test-patterns because of high degree of regularity in their structure. However, as the storage capacity of the IC RAM's are exponentially increasing, many of the conventional testing schemes require uneconomically large amount of time for testing. Moreover, the possibility of occurence of pattern-sensitive faults (interaction between the cells) is increasing in these high density IC's. A number of schemes are available for test pattern generation to cover these faults effectively, most of them being deterministic schemes.[3]-[8] Also, the trend is to provide Built-In Testing (BIT) schemes by assigning a small portion of the circuits to test the function which the remainder are intended to implement. As an alternative to these deterministic schemes, pseudorandom schemes for testing RAM by use of M-sequences for two different types of neighborhood was proposed by the authors[9],[10]. In this paper, pseudorandom testing of RAM as a system is discussed with an emphasis to its built-in structure. The proper choice of characteristic polynomial and test scheduling to minimize test-time are treated. Finally, an approach to realize the scheme for BIT is given without much attention to the necessary silicon area for the implementation of BIT.

2. Pseudorandom testing of RAM

Pseudorandom M-sequences can be used for the detection of Static Pattern-Sensitive Faults (SPSF) in RAM[9],[10]. In reference [9], SPSF detection in a neighborhood of five cells, as shown in Fig. 1 is considered. Three different schemes based on fourth and fifth order M-sequences were proposed for the detection of SPSF. In reference [10] the schemes were extended to a neighborhood of nine cells.

Here, a scheme to detect SPSF using a fifth order M-sequence is described. A memory cell under test is termed as a *base cell*. *Adjacent neighborhood* of a base cell is shown in Fig. 1. A base cell is said to have SPSF if its content is influenced by a set of patterns in the considered neighborhood.

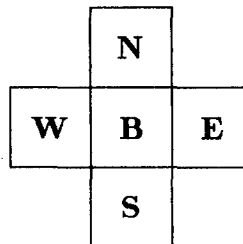


Fig. 1. Adjacent Neighborhood of a base cell.

2.1 Test Pattern Generation :

For adjacent neighborhood, a fifth order M-sequence as in (1) is used to fill the first row of the memory array under test.

$$M_0 = \{m_0, m_1, m_2, \cdots, m_{30}, m_0, \cdots\} \quad (1)$$
$$m_i = 0 \text{ or } 1; \; i = 0, 1, 2, \ldots 30$$

The following row sequences

$$\begin{aligned} M_L &= \{m_L, m_{L+1}, \cdots, m_{30}, m_0, \cdots\} \\ M_{2L} &= \{m_{2L}, m_{2L+1}, \cdots, m_{30}, m_0, \cdots\} \quad (2) \\ &\vdots \end{aligned}$$

are formed for the second row, the third row, and so on of the memory. L is the *lag* between the row sequences. The test procedure is as follows.

1. Clear all the memory cells.
2. Read the memory cells and check if they are cleared.
3. Fill the first row with sequence M_0, the second with M_L, the third with M_{2L}, ... and so on.
4. Read and check if the written message exists.
5. Steps 3 and 4 together represent one *phase*. Now advance M_0, M_L, $M_{2L}, \ldots$ one bit and repeat steps 3 and 4 till all the $31(2^5-1$ for a fifth order M-sequence) phases are over.

The above steps create all the required 32 patterns in an adjacent neighborhood if appropriate lags are applied between the row sequences. The problem of choosing appropriate lags to obtain complete set of test patterns is discussed in reference [9]. Table 1 gives the usable lags for different polynomials based on the analysis for existance of complete set of test patterns. These are cross checked by actual simulation of patterns around the base cells.

Table 1
Usable lags for different Polynomials

Characteristic Polynomial	Some usable lags	Total usable lags
X^5+X^2+1 X^5+X^3+1	$2, 3, 7, \ldots,$ $\ldots, 24, 28, 29$	14
$X^5+X^3+X^2+X+1$ $X^5+X^4+X^3+X^2+1$	$2, 3, 9, \ldots,$ $\ldots, 22, 28, 29$	14
$X^5+X^4+X^2+X+1$ $X^5+X^4+X^3+X+1$	$2, 4, 7, \ldots,$ $\ldots, 24, 27, 29$	16

2.2 Nature of Test Patterns and Usable lags

It is shown in reference [10] that the pseudorandom testing schemes cover the three basic fault types, stuck-at faults, pattern-sensitive faults and decoder faults fairly well in comparison to other conventional schemes. An advantage of pseudorandom testing against other deterministic schemes for PSF detection is that the number of phases required to obtain high fault coverage can be made less than n phases based on the previous knowledge of the occurence of the faults. This is because, during any phase, all the patterns except all-zero pattern can be found around some memory location when M-sequences are used for test pattern generation. It can be seen that, if lag L is relatively prime with n, the patterns repeat in blocks of $n \times n$. Therefore, in any phase, all the cells in a row that are 31 cells apart have the same pattern around them. Also, the pattern around the ith cell in any row during pth phase, is same as the pattern around the $(i+1)$th cell in the same row during $(p-1)$th phase. In effect, the entire row is a shift register without any feed back. With each clock pulse, a new element enters the last cell of the row and the element stored in the first cell of the row leaves the row. This definite order of the appearance of the test patterns helps test response collection.

The usable lags shown in Table 1 have some properties as follows.

P1. If L is a usable lag for a characteristic polynomial $P(X)$, then it is also a usable lag for the reciprocal polynomial $P^*(X)$.

P2. If L is a usable lag, then $(n-L)$ is also usable lag for any characteristic polynomial, where n is the period of the test sequence.

These properties are made use of during the choice of proper lags for BIT schemes.

2.3 Fault Diagnoses : For complete diagnoses of SPSF, the location of the faulty memory cell, the generalized pattern around a base cell and the phase number are necessary. As an illustration, let a cell in the cth column and rth row be faulty during pth phase and let $i_1 = p + c + rL$, then the pattern around the faulty cell is as follows

$$\begin{matrix} & X^{(i_1-L)} & \\ X^{(i_1-1)} & X^{i_1} & X^{(i_1+1)} \\ & X^{(i_1+L)} & \end{matrix} \quad (3)$$

where X represents a delay operator and all the powers are taken modulo n, the period length of the test sequence. From this we can say that the base cell is sensitive to a pattern $N = m_{i_1-L}$, $E = m_{i_1+1}$, $W = m_{i_1-1}$, $S = m_{i_1+L}$, when the base cell contains m_{i_1}.

Consequently, when the scheme is realized, it is necessary to apply *lag* between the row sequences, regenerate the test sequence during read operation and track the pattern around the cell being read. It is easier to incorporate all these facilities in external testers built to test IC memories. In the following section, a built-in structure for the proposed scheme is given so that the above necessities, i.e., application of lag, regeneration of test sequence, and tracking of the patterns around the base cell, can be included and the extra time consumed for initialization, and so on are minimized.

3. BIT Structure

A BIT structure for the proposed scheme is given in Fig. 2. The monitor block receives start test signal and controls test by sending the clock and control sequences to the other test modules. Once the test is over this block sends test over signal and the contents of the Test Response Collector (TRC) can be flushed out for external reading. Test Sequence Generator (TSG) is simply a linear feedback shift register with appropriate feedback connections. Address Generation Logic

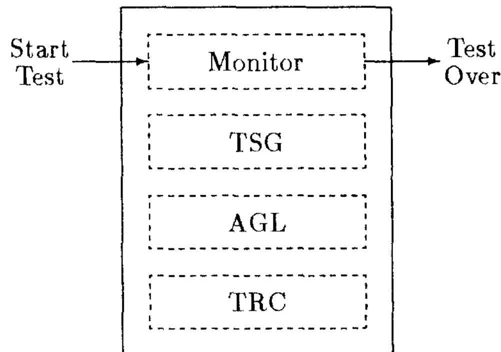
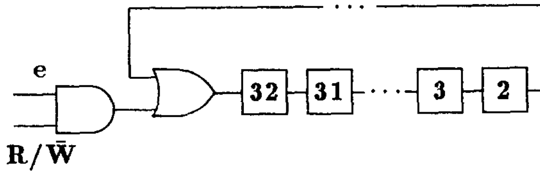


Fig. 2. Test generation module for BIT.

(AGL) is a pair of counters that can count the number of cells in a row and total number of rows respectively. The toughest part of pseudorandom testing is tracking the patterns around the base cell and this can be carried out as follows.

3.1 Test Response Collection : Generally, it is believed that the compression of the test response is difficult if random patterns are applied to detect pattern sensitive faults. However, if pseudorandom sequences are used there is a definite order in which the patterns appear around the base cells, as explained in section 2.2. Therefore, the possible 32 patterns around any base cell can be assigned with numbers from 1–32, for example, the patterns around the first cell during the each phase can be assigned with the respective phase number. Note that, as mentioned before, all-zero pattern (numbered one) appears only during the first phase and all the remaining patterns appear during each of the phases 2–32. Now, a 32bit register can be used to indicate faults due to these pattens by writing 1's into the bit positions of the patterns for which a fault has occured. Bits 2–32 can be connected in a shift register fashion as shown in Fig 3 so as to track the patterns around the base cells during read operations of phases 2–32. Nevertheless, it is necessary to adjust the contents of this shift register to account for lag and reinitialization during Read/Write operations. The time for these adjustments can be minimized by proper choice of the lags and characteristic polynomials.



e : Error Signal.
R/W̄ : Read-Write Signal.

Fig. 3. Test Response Collector (TRC).

3.2 Choice of Lags :

For discussion, let L be an usable lag, m be the order of the test-sequence, $n(n = 2^m - 1)$ be the period of the test-sequence, N_C be the number of cells in a row (or the number of columns), $N_{Cn} = N_C \bmod n$, and let N be the total number of cells in the memory array. Choose L such that $L > N_{Cn}$ but $(L - N_{Cn})$ as small as possible. The choice of such lags is possible and can be guaranteed, because from the property **P2** of the usable lags mentioned previously, if L is a usable lag for a characteristic polynomial, then $(n - L)$ is also a usable lag. Moreover, practically, the rowsize N_C is of the form 2^c, where c is an integer, meaning $N_{Cn} < (n-1)/2$, which assures the existence of lags $L > N_{Cn}$. In order to apply lags at the beginning of a new row, and adjust the TRC of Fig. 3 and so on, it is necessary to generate blocking pulses to block AGL, comparator functions and block R/W signal as indicated below.

1. $T_1 = (L - N_{Cn})$ number of clock pulses at the end of each row so as to account for the lag between the row sequences and adjust TRC.
2. $T_2 = (n - L_1)$ number of clock pulses at the end of write operation of the last row so as to adjust the TSG and TRC to their initial values of that phase. Where $L_1 = (L \times N_R)$ modulo n, and N_R is the number of rows in the memory array.
3. $T_3 = (n - L_1 + 1)$ number of clock pulses at the end of read operation as an initialization of TSG and TRC for the next phase.

3.3 Polynomial Selection :

In this paper BIT scheme only for adjacent neighborhood is considered. There are totally six primitive polynomials of order five including reciprocals. It can be observed that the usable lags are identical for a primitive polynomial and its reciprocal. Table 2 gives appropriate lags for $L_1 = L$, different row sizes and for different characteristic polynomials along with the timings T_1, T_2, T_3 . From the table it is clear that if these lags are used, the time for adjusting TSG and TRC can be minimized. Also, it can be observed that if primitive polynomial $X^5 + X^4 + X^2 + X + 1$ or its reciprocal is used for test-sequence generation, then application of lag between the row sequences can be eliminated for four out of five of the possible row sizes. Therefore, by judicious choice of the characteristic polynomial for test-sequence generation it is possible to eliminate some of the blocking pulses, leading to the minimization of time for testing and simplicity of the circuits involved.

Table 2
Suitable lags for different row sizes

Characteristic Polynomial	N_{Cn}	L_s	T_1	T_2	T_3
$X^5 + X^2 + 1$ and $X^5 + X^3 + 1$	1	2	1	29	30
	2	2	0	29	30
	4	7	3	24	25
	8	9	1	22	23
	16	16	0	15	16
$X^5 + X^3 + X^2 + X + 1$ and $X^5 + X^4 + X^3 + X^2 + 1$	1	2	1	29	30
	2	2	0	29	30
	4	9	5	22	23
	8	9	1	22	23
	16	16	0	15	16
$X^5 + X^4 + X^2 + X + 1$ and $X^5 + X^4 + X^3 + X + 1$	1	2	1	29	30
	2	2	0	29	30
	4	4	0	27	28
	8	8	0	23	24
	16	16	0	15	16

L_S ⇒ Suitable lag.
$L_1 = (N_R \times L)$ modulo n, is assumed to be L.
Other notations are as indicated in the text.

To further simplify the hardware a modulo $(n+1)$ counter, that stops counting when its content turns zero, can be used to generate these blocking pulses. The counter should be preset to $(n+1-T_1)$, $(n+1-T_2)$ and $(n+1-T_3)$ and clocked to obtain T_1, T_2, and T_3 respectively.

4. Conclusion

It is shown that the pseudorandom testing and dignoses of pattern sensitive faults by use of M-sequences of fifth order is possible in a neighborhood of five cells called adjacent neighborhood. The usable lags have to satisfy some conditions to obtain complete set of test patterns.

The scheme is suitable for built-in testing because the test-sequences can be generated by linear feedback shift registers. Even the address generation logic is simple due to sequential addressing of the memory cells in this scheme. The most complex part is to keep track of the patterns around a base cell for diagnoses, because during any phase of the test, there are 31 random patterns around base cells. This can be achieved by assigning the patterns with numbers to recognise these patterns. A 32 bit register (test response collector) can be set at positions corresponding to the number of the patterns whenever a fault occurs.

The properties of the usable lags can be taken advantage to minimize the time for adjustments of the test response collector and test sequence generator. By judicious choice of polynomials for test sequence generation, application of lag between the row sequences can be eliminated and hence time for testing and necessary hardware can be further minimized.

A widely conceived requirement of the built-in testing is that the extra hardware added for the purpose of testing should be within 10% of the total hardware. Hence, an assessment of the hardware requirement is necessary. However, the authors believe that this requirements will be met especially when this structure is used for multi-megabit RAM's. This BIT structure can also be modified for the scheme using fourth order M-sequences proposed in reference [9].

References

1. Golomb, S.W., *Shift Register Sequences*, pp. 85-88, Holdenday (1969).
2. Kashiwagi, H., Recent Topics On M-sequences, *Journal of SICE*, Vol.20, No.2, pp. 236-245, (in Japanese) (1981).
3. Kinoshita, K. et al., Built-In Testing of Memory using an On-chip compact testing scheme, *IEEE Trans. Comput.*, Vol.C-35, No.10, pp. 862-870, Oct. 1986.
4. Saluja, K.K. et al., Built-In Self-testing RAM : A practicle alternative, *IEEE Design & Test*, Vol.4, No.1, pp. 42-51, Feb, 1987.
5. Hayes, J.P., Detection Pattern-Sensitive Faults in Randomaccess memories, *IEEE Trans. Comput.*, Vol.C-24, No.2, pp. 150-157, Feb. 1975.
6. Sohl, W.E., Selecting test patterns for 4K RAM's, *IEEE Trans. Manufacturing Tech.* Vol. MFT-6, pp. 51-60, Sept. 1977.
7. Nair, R. et al, Efficient algorithm for testing Semiconductor Randomaccess Memories, *IEEE Trans. Comput.*, Vol. C-27, No.6, pp. 572-576, June 1978.
8. Suk, D.S. et al., Test procedures for a class of Pattern-Sensitive faults in Semiconductor Randomaccess memories, *IEEE Trans. Comput.*, Vol.C-29, No.6, pp. 419-429, June 1980.
9. Rao, G. K. and Kashiwagi, H., Pattern-Sensitive Fault Detection in RAM, *The Transactions of The IEICE*, Japan, Vol. E72, No.5 pp. 502-506, May 1989.
10. Kashiwagi, H. and Rao, G.K. , Pseudo-random Testing of RAM for Pattern-Sensitive Faults, *The Proceedings of SICE '89*, Matsuyama, Japan, pp. 1125-1129, July 25-27, 1989.