An Efficient Built-in Self-Test Algorithm for Neighborhood Pattern- and Bit-Line-Sensitive Faults in High-Density Memories

Dong-Chual Kang, Sung Min Park, and Sang-Bock Cho

As the density of memories increases, unwanted interference between cells and the coupling noise between bit-lines become significant, requiring parallel testing. Testing high-density memories for a high degree of fault coverage requires either a relatively large number of test vectors or a significant amount of additional test circuitry. This paper proposes a new tiling method and an efficient built-in self-test (BIST) algorithm for neighborhood pattern-sensitive faults (NPSFs) and new neighborhood bit-line sensitive faults (NBLSFs). Instead of the conventional five-cell and nine-cell physical neighborhood layouts to test memory cells, a four-cell layout is utilized. This four-cell layout needs smaller test vectors, provides easier hardware implementation, and is more appropriate for both NPSFs and NBLSFs detection. A CMOS column decoder and the parallel comparator proposed by P. Mazumder are modified to implement the test procedure. Consequently, these reduce the number of transistors used for a BIST circuit. Also, we present algorithm properties such as the capability to detect stuck-at faults, transition faults, conventional pattern-sensitive faults, and neighborhood bit-line sensitive faults.

Keywords: Memory, BIST, NBLSF, NPSF, testing.

Manuscript received Feb. 23, 2004; revised July 16, 2004.

This work was supported by 2002 research fund of the University of Ulsan.

Dong-Chual Kang (email: kangdc@isselah.com) and Sang-Bock Cho (email: sbcho@mail.ulsan.ac.kr) are with School of Electrical Engineering, University of Ulsan, Ulsan, Korea.

Sung Min Park (phone: +82 2 3277 4085, email: smpark@ewha.ac.kr) is with the Department of Information Electronics Engineering, Ewha Womans University, Seoul, Korea.

I. Introduction

Pattern-sensitive faults (PSFs) may be considered the most general case of k-coupling faults, for k = n, where n represents all cells in the memory. The PSF is defined as the susceptibility of the contents of a cell in a memory array to the contents of all other cells [1]-[3]. A group of cells that influence the base cell's behavior is called the neighborhood of the base cell. PSFs occur primarily due to the high component densities of RAMs and unwanted interference between closely packed lines and signals. Also, various physical fault mechanisms are responsible for PSFs, making it difficult to define the logical fault models. RAM testing for unrestricted PSFs would be very costly and impractical since it requires a test length of $(3n^2 + 2n)2^n$ [3]-[4]. Thus, the PSF models assume that interaction occurs only between the cells within a certain physical proximity. These are called physical neighborhood pattern-sensitive faults (NPSFs). PSF models are classified into active NPSFs (ANPSFs), passive NPSFs (PNPSFs), and static NPSFs (SNPSFs) [4]-[6]. Algorithms for RAM testing have been reported [7]. However, they are impractical because the test complexities of Type-1 and Type-2 neighborhoods, as shown in Fig. 1, are O(n). Meanwhile, P. Mazumder introduced a parallel testing algorithm for PSFs whose test complexity is reduced to $O(\sqrt{n})$ by adding the extra circuits, 'parallel comparator' and 'error detector' [6]. Considering the interaction between cells only, Type-2 has better fault coverage than Type-1. In fact, the leakage is at maximum when the symmetrically located cells contain the same bit patterns, and thus a variant of the 4-cell neighborhood introduced in [6] can cover Type-2.

However, these algorithms focus on the interaction between cells and partially consider the coupling noise through the C_g (gate capacitance), C_{BB} (bit-line to bit-line coupling capacitance), $C_{Pull\text{-}Down}$ (capacitance of a pull-down circuit in memory elements), and C_{BW} (bit-line to word-line coupling capacitance). Among these, C_{BB} is the most critical parasitic capacitance that affects the cells in read/write operations. Mostly, conventional fault models can mask faults due to C_{BB}. Therefore, it is difficult to consider maximum coupling noise in the test algorithms in the normal read/write operations of a memory. Hence, it mandates the usage of a parallel test.

The more neighborhood cells that are considered, the more the interaction between neighborhood cells can be known. On the other hand, the number of test patterns for PSFs detection is increased, and a longer test-time for memory cells is needed. The five-cell and nine-cell physical neighborhoods, which are typical tiling methods, need more test patterns than the four-cell physical neighborhood proposed in this paper and are not appropriate for testing bit-line coupling noise because of the redundant cells discussed in the last paragraph of section IV. By using the four-cell physical neighborhood, we show that both NPSFs and a neighborhood bit-line sensitive fault (NBLSF) can be detected with fewer neighborhood cells. Also, a built-in self test (BIST) circuit is proposed for parallel testing [8], [9]. Using the four-cell physical neighborhood provides a number of advantages, such as easier hardware implementation and smaller overhead. Also, the proposed method utilizes parallel testing, which is very useful for testing bit-line coupling noise. In order to test exact NBLSFs, the decoding time to access all cells should be constant and fast enough to maximize the bit-line coupling noise. This work shares and modifies the basic building blocks, such as a CMOS column decoder, a parallel comparator, and an error detector, which were originally proposed by P. Mazumder [6], due to their easy controllability and the reduced number of transistors for BIST circuits. Its detailed structure and discussion will be given in section VI. Figure 1 shows the four-cell and five-cell physical neighborhoods. These two types have inappropriate structures to test NBLSFs because cells 2 and 4 in Fig. 1(a), and cells 2, 4, 5, 6,

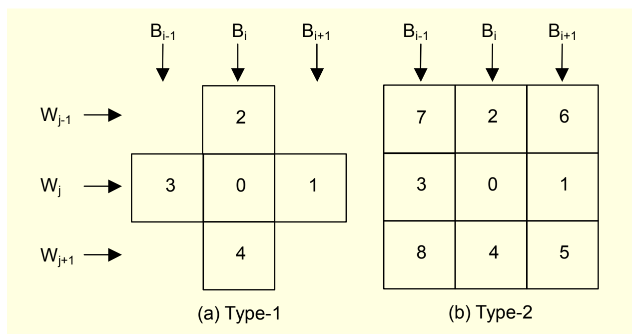


Fig. 1. Five-cell and nine-cell physical neighborhoods.

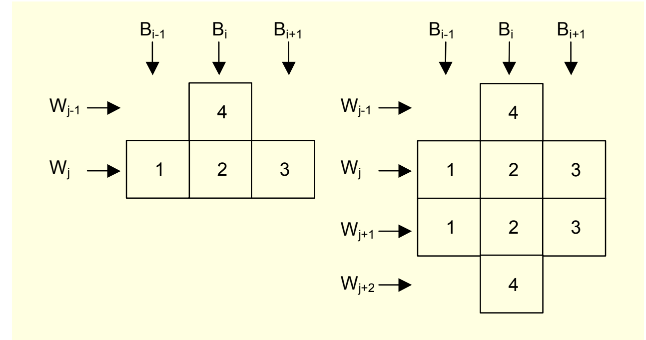


Fig. 2. A four-cell physical neighborhood.

7, and 8 in Fig. 1(b) are connected to the same bit-lines. Meanwhile, the proposed type, shown in Fig. 2, has only one redundant cell 4 that exists for NPSFs testing.

II. Memory Array Tiling

Since every cell in the k-cell neighborhood is assigned an integer number from 0 to k−1, every cell in the memory will be marked with numbers from 0 to k−1 and be a base cell. Thus, the four-cell neighborhood shown in Fig. 2 indicates the reduction of cells and test patterns. All possible tiling groups include a specific cell ($C_{i,j}$) and form a new effective physical size of the neighborhood, which is called neighborhood effective size [6]. Figure 3 shows the neighborhood effective sizes of Type-1 and Type-2.

Figure 4 illustrates the procedure for the four-cell tiling neighborhood marking with a fixed pattern that is assigned to every cell. The sum of each dark part is shown more clearly in Fig. 5. The cell number assignment in the memory is divided into two groups, as shown in Fig. 4. In one group (A or B), neighborhood cells of a tiling group are independent of those of other groups in different row addresses, i.e., no interference. This can be simply solved by assigning a cell number, as shown in group B of Fig. 4. Every cell labeled as 1, 2, 3, or 4 can be a base cell, and therefore each group has 4 subgroups according to four base cells. Now the specific cell ($C_{i,j}$) in the memory can be tiled in $C_{i,j}(i-1<i<i+1/j-1<j<j+1)$, $C_{i,j\pm2}$, and $C_{i\pm2\,j}$. This neighborhood effective size, shown in Fig. 5, is the same as that of Type-1 neighborhoods, indicating that both types have the same size such that a certain cell of a four-cell layout or five-cell layout can be tiled.

The fault coverage for PSFs is generally related with the used pattern size, which is determined by the number of neighborhood cells. However, a cell can be influenced by all the cells in the neighborhood effective size, which is determined by the tiling method of the used pattern [6]. In other words, the fault coverage cannot be justified by the pattern size only. The pattern size of a four-cell layout is smaller then that of a five-cell layout, but the neighborhood effective size of a four-cell layout is the

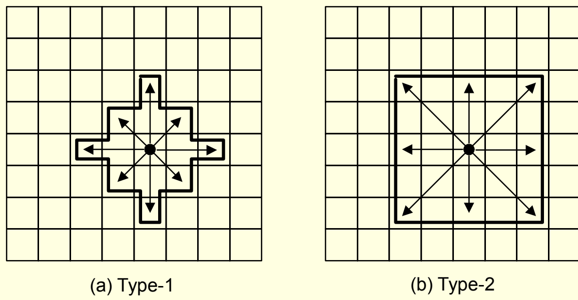


Fig. 3. Neighborhood effective size.

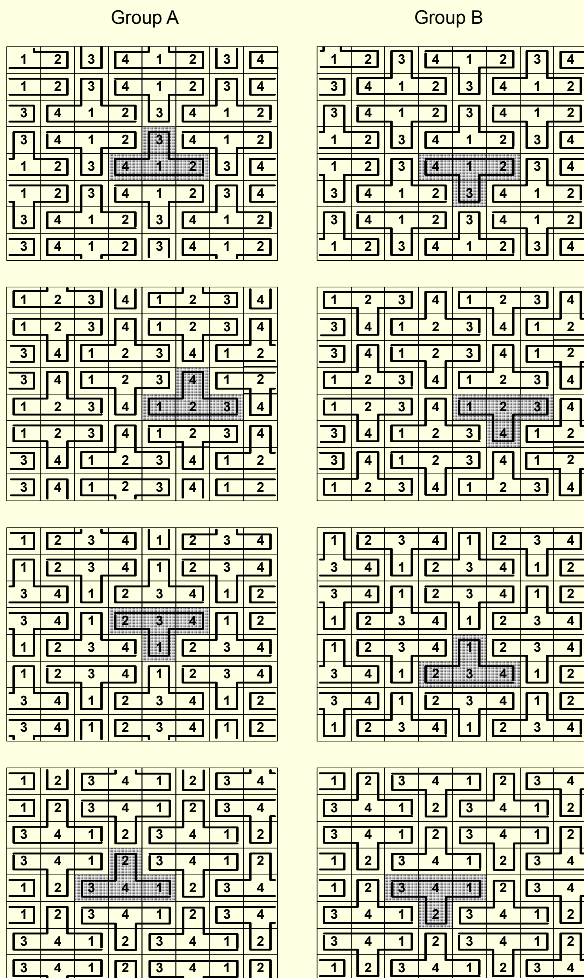


Fig. 4. Procedures for a four-cell tiling neighborhood marking with a fixed pattern assigned to every tile.

same with that of a five-cell layout by the proposed tiling method.

III. Pattern Sensitive Faults

The NPSFs are often classified into the following three categories:

- ANPSF (Active NPSF) in which a base cell changes its contents as a result of changes in the pattern of the neighborhood cells. Uses an Eulerian sequence.
- PNPSF (Passive NPSF) in which the contents of a base cell cannot be changed due to the influence of an existing pattern in the neighborhood cells. Uses an Eulerian sequence.
- SNPSF (Static NPSF) in which the contents of a base cell are forced to be a certain value by the contents of an existing pattern in the neighborhood cells. Uses a k-bit Hamiltonian sequence.

A general approach to test base cells for ANPSF and PNPSF is either to sensitize any fault using an appropriate neighborhood pattern, or to read the state of a base cell after each sensitizing pattern to detect any fault. The number of patterns needed to sensitize all the ANPSFs for a neighborhood of size k is given by $(k-1)2^k$ since each one of the $(k-1)$ neighbors of the base cell must execute both $1\rightarrow0$ and $0\rightarrow1$ transitions while $(k-1)$ neighbors take all possible binary values. The number of patterns necessary to sensitize all the PNPSFs in a k-cell neighborhood is 2^k.

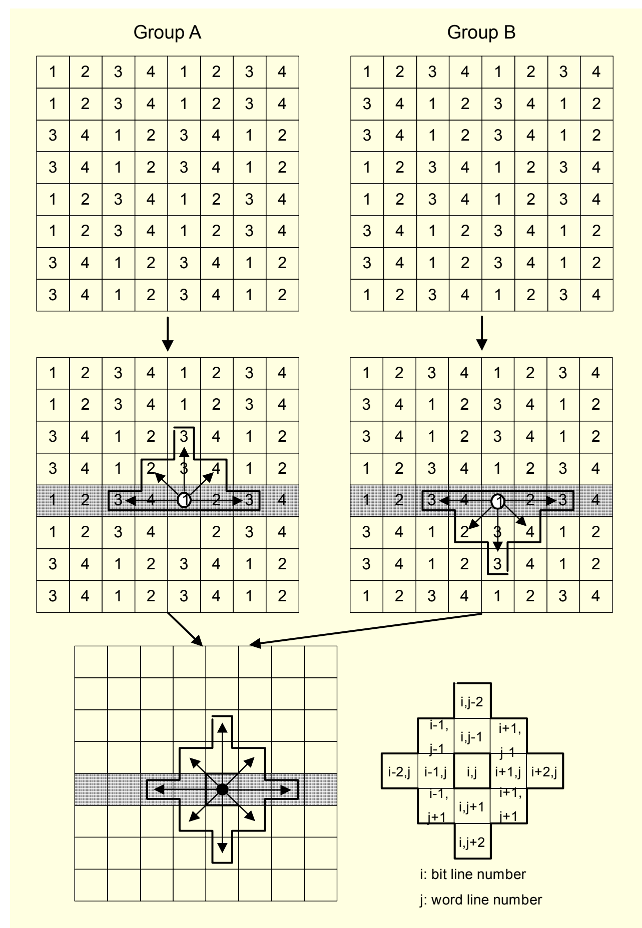


Fig. 5. Neighborhood effective size of the proposed type.

Table 1. Test patterns.

TP#	B3	B2	B1	B0	TP#	B3	B2	B1	B0	TP#	B3	B2	B1	B0	TP#	B3	B2	B1	B0
1	0	0	0	0	17	1	0	0	1	33	0	1	0	1	49	0	0	1	1
2	0	0	0	1	18	0	0	0	1	34	0	0	0	1	50	0	0	0	1
3	0	0	1	1	19	0	0	0	0	35	1	0	0	1	51	0	1	0	1
4	0	0	1	0	20	1	0	0	0	36	1	1	0	1	52	0	1	1	1
5	0	1	1	0	21	1	0	1	0	37	1	1	0	0	53	1	1	1	1
6	0	1	1	1	22	0	0	1	0	38	1	0	0	0	54	1	1	0	1
7	0	1	0	1	23	0	0	1	1	39	0	0	0	0	55	1	0	0	1
8	0	1	0	0	24	1	0	1	1	40	0	1	0	0	56	1	0	1	1
9	1	1	0	0	25	1	1	1	1	41	0	1	1	0	57	1	0	1	0
10	1	1	0	1	26	0	1	1	1	42	0	0	1	0	58	1	0	0	0
11	1	1	1	1	27	0	1	1	0	43	1	0	1	0	59	1	1	0	0
12	1	1	1	0	28	1	1	1	0	44	1	1	1	0	60	1	1	1	0
13	1	0	1	0	29	1	1	0	0	45	1	1	1	1	61	0	1	1	0
14	1	0	1	1	30	0	1	0	0	46	1	0	1	1	62	0	1	0	0
15	1	0	0	1	31	0	1	0	1	47	0	0	1	1	63	0	0	0	0
16	1	0	0	0	32	1	1	0	1	48	0	1	1	1	64	0	0	1	0

The sum of all ANPSFs and PNPSFs is given by

$$(k-1)2^k + 2^k = k \cdot 2^k . \qquad (1)$$

Table 1 shows the required test patterns. NPSFs are better detected by transition writes rather than by non-transition writes. With the change of neighborhood cells and base cells, leakage currents appear in PSFs. If faults rise after non-transition writes, they may be caused by another factor, such as coupling noise by parasitic elements, sense amplifier sluggishness, etc.

This algorithm is based on the assumption that the memory array's physical and logical neighborhoods are identical, and also on the assumption of the bit-oriented memory NPSFs detection. Every cell can become a base cell by an Eulerian sequence. Sixty-four test patterns generated by a test pattern generator (TPG) are repeated, and thus 128 test patterns are ultimately loaded to the memory cells.

1. SNPSFs Detection

To sensitize all the SNPSFs, all possible states of the cells in the tiling neighborhood should be written and read. This can be realized by steps 3 and 4 of the test procedure in Fig. 6. The number of SNPs is given by 4×2^k (number of base cells × SNPs). Table 2 shows the SNPs with respect to test patterns when all the base cells are considered.

2. PNPSFs Detection

Each PNP contains exactly one transition write operation ($0 \rightarrow 1(\uparrow)$, $1 \rightarrow 0(\downarrow)$), and the number of the PNFs is given by 4×2^k (number of base cells × PNPs). Table 3 shows the PNPs with respect to 64 test patterns when all the base cells are considered.

3. ANPSFs Detection

The number needed to sensitize all the ANPSFs is given by $(k-1) \times 2^k$ per base cell. Each active neighborhood pattern (ANP) has exactly one transition write operation. Table 4 shows the ANPs with respect to four base cells. This test procedure sensitizes all the NPSFs and is done with the four-cell layout, where the memory is totally covered by a group of neighborhoods that do not overlap. As read and write operations with patterns are performed in steps 3 and 4 in Fig. 6, respectively, the algorithm can detect not only stuck-at faults, transition faults, PNPSFs, and SNPSFs, but also ANPSFs. A stuck-open fault indicates that a cell cannot be accessed because of an open word

```
Test procedure for detecting NPSFs is as follows.
Step 1: Initialize memory cells with "0"
Step 2: TP[3,2,1,0]:=TP#n['B3"B2"B1"B0']
Step 3: //WRITE ACTION
        for row:=0 to n-1 do
          begin
              //col:= Label number whose value is changed
             Data: = TP[col];
             C[row,col]:=   TP[col];
          end;
Step 4: //READ ACTION
        for row:=0 to n-1 do
          begin
            Data: = Base cell;
            if (C[row,col]<>Data) then output('Error at cell', cell);
          end;
Step 5: repeat step 2, 3, and 4 until all of the 128 test patterns are applied;
```

Fig. 6. Algorithm for NPSFs detection.

Table 2. Static neighborhood patterns (SNPs) with respect to test patterns.

Base cell	SNPs	Test patterns	Base cell	SNPs	Test patterns
Label 1	0 0 1 -	TP#22 - TP#23	Label 2	0 0 - 0	TP#63 - TP#64
	0 0 0 -	TP#1 - TP#2		0 0 - 1	TP#2 - TP#3
	0 1 1 -	TP#5 - TP#6		0 1 - 0	TP#40 - TP#41
	0 1 0 -	TP#30 - TP#31		0 1 - 1	TP#51 - TP#52
	1 0 1 -	TP#12 - TP#13		1 0 - 0	TP#20 - TP#21
	1 0 0 -	TP#16 - TP#17		1 0 - 1	TP#55 - TP#56
	1 1 1 -	TP#44 - TP#45		1 1 - 0	TP#59 - TP#60
	1 1 0 -	TP#9 - TP#10		1 1 - 1	TP#10 - TP#11
Label 3	0 - 0 0	TP#39 - TP#40	Label 4	- 0 0 0	TP#19 - TP#20
	0 - 0 1	TP#50 - TP#51		- 0 0 1	TP#34 - TP#35
	0 - 1 0	TP#4 - TP#5		- 0 1 0	TP#42 - TP#43
	0 - 1 1	TP#47 - TP#48		- 0 1 1	TP#23 - TP#24
	1 - 0 0	TP#58 - TP#59		- 1 0 0	TP#8 - TP#9
	1 - 0 1	TP#35 - TP#36		- 1 0 1	TP#31 - TP#32
	1 - 1 0	TP#43 - TP#44		- 1 1 0	TP#27 - TP#28
	1 - 1 1	TP#24 - TP#25		- 1 1 1	TP#52 - TP#53

Table 3. Passive neighborhood patterns (PNPs) with respect to test patterns.

Base cell	PNPs	Test patterns	Base cell	PNPs	Test patterns
Label 1	0 0 0 -	TP#1-2, TP#18-19	Label 2	0 0 - 0	TP#63-64, TP#64-1
	0 0 1 -	TP#22-23, TP#3-4		0 0 - 1	TP#2-3, TP#49-50
	0 1 1 -	TP#5-6, TP#26-27		0 1 - 0	TP#40-41, TP#61-62
	0 1 0 -	TP#30-31, TP#7-8		0 1 - 1	TP#51-52, TP#6-7
	1 0 1 -	TP#12-13, TP#56-57		1 0 - 0	TP#20-21, TP#57-58
	1 0 0 -	TP#16-17, TP#15-16		1 0 - 1	TP#55-56, TP#14-15
	1 1 0 -	TP#9-10, TP#36-37		1 1 - 0	TP#59-60, TP#28-29
	1 1 1 -	TP#44-45, TP#11-12		1 1 - 1	TP#10-11, TP#53-54
Label 3	0 - 0 0	TP#39-40, TP#62-63	Label 4	- 0 0 0	TP#19-20, TP#38-39
	0 - 0 1	TP#50-51, TP#33-34		- 0 0 1	TP#34-35, TP#17-18
	0 - 1 0	TP#4-5, TP#41-42		- 0 1 0	TP#42-43, TP#21-22
	0 - 1 1	TP#47-48, TP#48-49		- 0 1 1	TP#23-24, TP#46-47
	1 - 0 0	TP#58-59, TP#37-38		- 1 0 0	TP#8-9, TP#29-30
	1 - 0 1	TP#35-36, TP#54-55		- 1 0 1	TP#31-32, TP#32-33
	1 - 1 0	TP#43-44, TP#12-13		- 1 1 0	TP#27-28, TP#60-61
	1 - 1 1	TP#24-25, TP#45-46		- 1 1 1	TP#52-53, TP#25-26

line. When a read operation is performed on a cell, the differential sense amplifier has to sense a voltage difference between the bit lines of the cell. In the case of a stuck-open fault, both bit lines will have the same voltage.

IV. Neighborhood Bit-Line Sensitive Faults (NBLSF)

Figure 7(a) shows representative coupling capacitors in the memory array [10]. These capacitors appear between neighbor elements as parasitic capacitance, causing noise. Even though they cannot change the data of a cell, they can delay write/read operations of memory cells. If a word-line is fed with high voltage (5 to 7 V), bit-lines across the word-line raise the voltage level to 100 to 250 mV by the C_{BW} (bit-line to word-line coupling capacitance). This unexpected voltage can be mostly eliminated by the folded bit-lines, as shown in Fig. 7(b).

Table 4. ANPs with respect to test patterns.

Base cell	ANPs	Test patterns	Base cell	ANPs	Test patterns
Label 1	0 0 - 0	TP#63-64, TP#64-1	Label 2	0 0 1 -	TP#1-2, TP#3-4
	0 0 - 1	TP#2-3, TP#49-50		0 0 0 -	TP#22-23, TP#18-19
	0 1 - 0	TP#40-41, TP#61-62		0 1 1 -	TP#5-6, TP#26-27
	1 0 - 0	TP#51-52, TP#6-7		0 1 0 -	TP#30-31, TP#7-8
	1 0 - 1	TP#20-21, TP#57-58		1 0 1 -	TP#12-13, TP#56-57
	1 1 - 0	TP#55-56, TP#14-15		1 0 0 -	TP#16-17, TP#15-16
	1 1 - 1	TP#59-60, TP#28-29		1 1 1 -	TP#44-45, TP#11-12
		TP#10-11, TP#53-54		1 1 0 -	TP#9-10, TP#36-37
	0 - 0 0	TP#39-40, TP#62-63		0 - 0 0	TP#39-40, TP#62-63
	0 - 0 1	TP#50-51, TP#33-34		0 - 0 1	TP#50-51, TP#33-34
	0 - 1 0	TP#4-5, TP#41-42		0 - 1 0	TP#4-5, TP#41-42
	0 - 1 1	TP#47-48, TP#48-49		0 - 1 1	TP#47-48, TP#48-49
	1 - 0 0	TP#58-59, TP#37-38		1 - 0 0	TP#58-59, TP#37-38
	1 - 0 1	TP#35-36, TP#54-55		1 - 0 1	TP#35-36, TP#54-55
	1 - 1 0	TP#43-44, TP#12-13		1 - 1 0	TP#43-44, TP#12-13
	1 - 1 1	TP#24-25, TP#45-46		1 - 1 1	TP#24-25, TP#45-46
	- 0 0 0	TP#19-20, TP#38-39		- 0 0 0	TP#19-20, TP#38-39
	- 0 0 1	TP#34-35, TP#17-18		- 0 0 1	TP#34-35, TP#17-18
	- 0 1 0	TP#42-43, TP#21-22		- 0 1 0	TP#42-43, TP#21-22
	- 0 1 1	TP#23-24, TP#46-47		- 0 1 1	TP#23-24, TP#46-47
	- 1 0 0	TP#8-9, TP#29-30		- 1 0 0	TP#8-9, TP#29-30
	- 1 0 1	TP#31-32, TP#32-33		- 1 0 1	TP#31-32, TP#32-33
	- 1 1 0	TP#27-28, TP#60-61		- 1 1 0	TP#27-28, TP#60-61
	- 1 1 1	TP#52-53, TP#25-26		- 1 1 1	TP#52-53, TP#25-26
Base cell	**ANPs**	**Test patterns**	**Base cell**	**ANPs**	**Test patterns**
Label 3	0 0 1 -	TP#1-2, TP#3-4	Label 4	0 0 1 -	TP#1-2, TP#3-4
	0 0 0 -	TP#22-23, TP#18-19		0 0 0 -	TP#22-23, TP#18-19
	0 1 1 -	TP#5-6, TP#26-27		0 1 1 -	TP#5-6, TP#26-27
	0 1 0 -	TP#30-31, TP#7-8		0 1 0 -	TP#30-31, TP#7-8
	1 0 1 -	TP#12-13, TP#56-57		1 0 1 -	TP#12-13, TP#56-57
	1 0 0 -	TP#16-17, TP#15-16		1 0 0 -	TP#16-17, TP#15-16
	1 1 1 -	TP#44-45, TP#11-12		1 1 1 -	TP#44-45, TP#11-12
	1 1 0 -	TP#9-10, TP#36-37		1 1 0 -	TP#9-10, TP#36-37
	0 0 - 0	TP#63-64, TP#64-1		0 - 0 0	TP#39-40, TP#62-63
	0 0 - 1	TP#2-3, TP#49-50		0 - 0 1	TP#50-51, TP#33-34
	0 1 - 0	TP#40-41, TP#61-62		0 - 1 0	TP#4-5, TP#41-42
	0 1 - 1	TP#51-52, TP#6-7		0 - 1 1	TP#47-48, TP#48-49
	1 0 - 0	TP#20-21, TP#57-58		1 - 0 0	TP#58-59, TP#37-38
	1 0 - 1	TP#55-56, TP#14-15		1 - 0 1	TP#35-36, TP#54-55
	1 1 - 0	TP#59-60, TP#28-29		1 - 1 0	TP#43-44, TP#12-13
	1 1 - 1	TP#10-11, TP#53-54		1 - 1 1	TP#24-25, TP#45-46
	- 0 0 0	TP#19-20, TP#38-39		0 0 - 0	TP#63-64, TP#64-1
	- 0 0 1	TP#34-35, TP#17-18		0 0 - 1	TP#2-3, TP#49-50
	- 0 1 0	TP#42-43, TP#21-22		0 1 - 0	TP#40-41, TP#61-62
	- 0 1 1	TP#23-24, TP#46-47		0 1 - 1	TP#51-52, TP#6-7
	- 1 0 0	TP#8-9, TP#29-30		1 0 - 0	TP#20-21, TP#57-58
	- 1 0 1	TP#31-32, TP#32-33		1 0 - 1	TP#55-56, TP#14-15
	- 1 1 0	TP#27-28, TP#60-61		1 1 - 0	TP#59-60, TP#28-29
	- 1 1 1	TP#52-53, TP#25-26		1 1 - 1	TP#10-11, TP#53-54

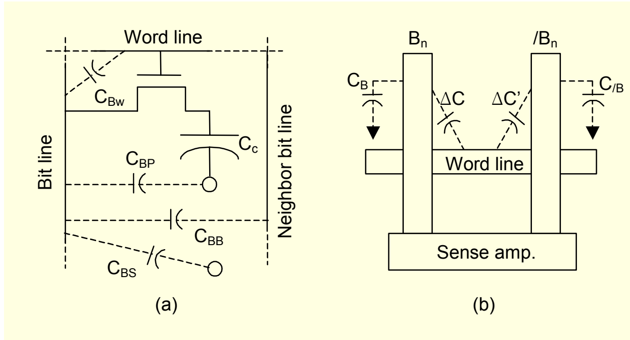


Fig. 7. Parasitic capacitors and structure of the folded bit-line.

It is typically utilized in high-density memories. Yet, there are noise factors due to the difference between the coupling capacitances of B_n and $/B_n$. The coupling capacitance of bit-lines connected to cells is given by C_g (gate capacitance) + $C_{BW,}$ which is larger than that of the bit-lines not connected to cells.

As a memory is highly dense, the crosstalk between bit-lines is tremendously increased by C_{BB} (bit-line to bit-line coupling capacitance) during read/write operations. The crosstalk occurs during charge sharing and sense amplification and reaches its maximum when the values of the bit-lines are complement and its minimum when the values of neighborhood bit-lines are equal. The crosstalk is considerably reduced by twisting bit-lines with similar mechanisms such as coaxial cables. However, the data of a cell may be changeable. Hence, C_{BB} becomes one of the most important factors to delay the response for valid memory operations. Figure 8 shows the C_{BB} between bit-lines and the timing diagram for normal read operations. It is seen that C_g, C_{BB}, $C_{Pull\text{-}Down}$ (capacitance of a pull-down circuit in memory elements), and C_{BW} affect charge sharing, and that C_B (bit-line capacitance), C_{BB}, and C_{BW} affect amplification. C_{BW} has little effect on neighbor bit-lines, since it can be mostly removed by using the folded bit-lines and since a high-density memory indicates the minimum distance between bit-lines, rather than between word-lines and bit-lines. Meanwhile, C_{BB} sharply increases as the distance between neighbor bit-lines decreases. Thus, crosstalk, voltage-drop, and voltage-rise are expected to occur in each neighbor bit-line. As a consequence, they affect the data of cells especially during high-speed read/write operations.

A parallel test is utilized in order to maximize the unexpected effects. It is satisfactory for both a high-speed test and the maximum possibility to cause faults based on NPSF. During write/read operations, charge sharing always occurs. The voltage level of the bit-lines are then calculated by

$$V_H=(C_BV_{DD}+C_CV_{DD})/(C_B+C_C),$$

$$\text{therefore, } \Delta V_H=V_H-\Delta V_{DD}/2= (V_{DD}/2)/(1+C_B/C_C), \text{ and} \quad (2)$$

$$V_L= (C_BV_{DD}/2)/(C_B+C_C),$$

$$\text{thus, } \Delta V_L=V_L-V_{DD}/2=-(V_{DD}/2)/(1+C_B/C_C), \quad (3)$$

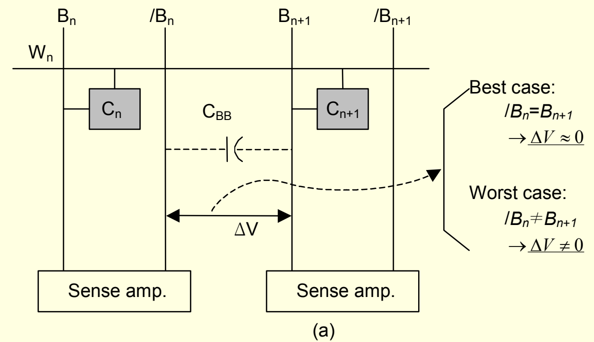
where C_C (cell-capacitance) and C_B are 30 to 40 fF and 250 to 300 fF, respectively. Since C_B is 7 times larger than C_C at maximum, the above equations can be given by

$$\Delta V_H=(V_{DD}/2)/(1+C_B/C_C)=2.5/8 \fallingdotseq 0.3 \text{ (worst case)}, \quad (4)$$

$$\Delta V_L= -(V_{DD}/2)/(1+C_B/C_C) \fallingdotseq -0.3 \text{ (worst case)}. \quad (5)$$

Considering the coupling capacitance, C_B should be replaced by (C_B+C_{BB}) in (4) and (5). Therefore, the increase of the bit-line capacitance by C_{BB} results in the decrease of both $\triangle V_H$ and $\triangle V_L$. In the worst case, the contents of the cell can be changed. As the initial voltage for charge sharing is about ±0.3 V, the increase of C_B causes a critical problem.

Now consider faults by the coupling capacitance between bit-lines. Both interaction between cells and coupling noise are equivalently considered, and non-transition write as well as transition write are also important because coupling noise may occur in non-transition write operations. The B_i of a certain cell $C_{i,j}$ are well affected by adjacent bit-lines $/B_{i-1}$, not by adjacent cells in high-speed read/write operations. The coupling noise by neighbor bit-line $/B_{i-1}$ reflects on cell $C_{i,j}$. Coupling noise around



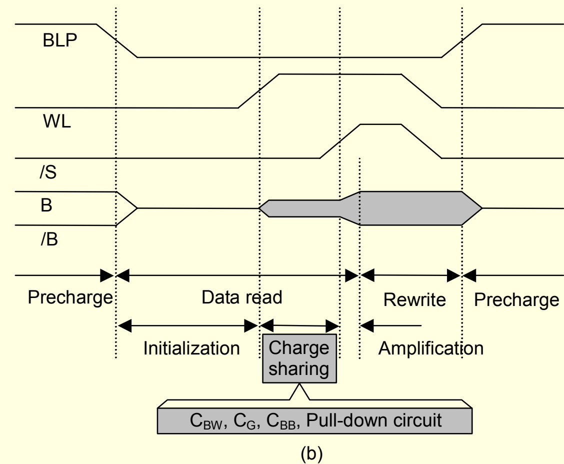


Fig. 8. The C_{BB} and charge sharing.

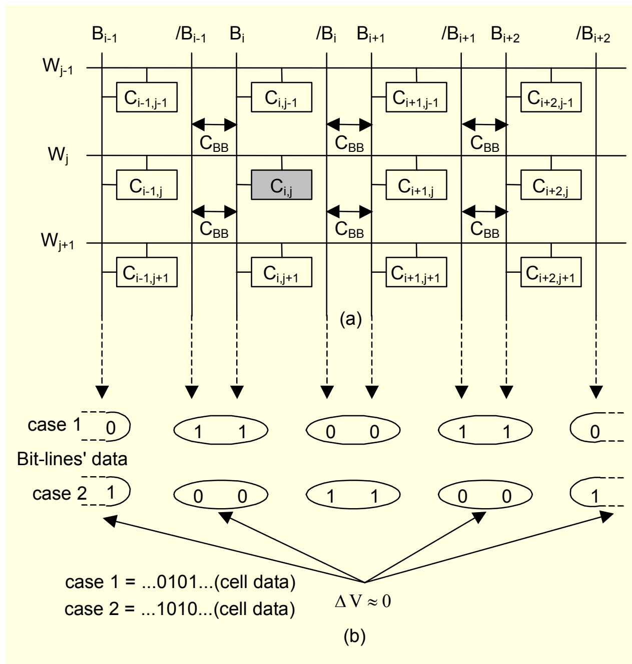


Fig. 9. Memory structure and the condition of minimum coupling noise.

$C_{i,j}$ is activated by the read/write operations of $C_{i-1,j}$ connected to $/B_{i-1}$ and $C_{i+1,j}$ connected to B_{i+1}. This is the main factor to determine tiling shape. Every neighborhood cell connected to the same bit-line is not necessary to test faults caused by neighborhood bit-lines. However, every cell in the same word-line must be tested by proper test patterns in parallel in order to consider coupling noise by neighbor bit-lines.

Figure 9 shows the condition of minimum coupling noise between adjacent bit-lines, of which the values are the same but the data of cells in the same word-lines are opposite one another. This implies that the simultaneous consideration of both maximum leakage currents between cells and maximum coupling noise by adjacent bit-lines are not feasible. Thus, it mandates an effective tiling and parallel testing method appropriate for both NPSF and NBLSF detection.

Figure 10 shows that the crosstalk between bit-lines is maximized just after refresh operations, when the data values of adjacent bit-lines are opposite each other. On this condition, frequency characteristics become worse. In the refresh mode, $C_{i,j}$ are well affected by both $/B_{i-1}$ and B_{i+1}. According to the refresh time and coupling noise, $C_{i,j}$ voltage may not go up to V_{DD} or down to V_{SS}. That is, the cell charge does not reach the expected value. The cell in the following read operation is also under the coupling noise because of the leftover voltage of neighborhood bit-lines; therefore, $|\Delta V_{H,\,L}|$ becomes smaller during charge sharing. Consequently, the sense amplifier cannot amplify $|\Delta V_{H,L}|$ in the proper time period, which means the cell is not accessed for some time. This kind of fault is classified as a data retention fault. Also, data inversion may be expected because of severe coupling noise and a small amount of charge leaking away from the cell. To test faults using the coupling noise of bit-lines, write operations must be implemented in parallel to test NBLSFs, such as the phenomenon after a refresh operation. Read/write operations to test NPSFs have a period of $\sqrt{n}$, whereas the period of read/write operations for NBLSFs detection is “1”. The change of base cells should be checked after every write operation. The NBLSFs can be classified into the following two categories:

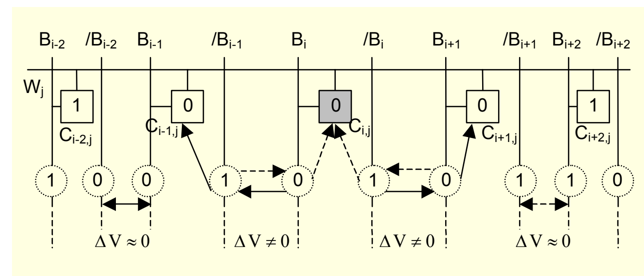


Fig. 10. Maximum coupling noise.

- ANBLSF (Active NBLSF) in which the base cell changes its contents as a result of changes in the neighborhood bit-lines. Uses an Eulerian sequence.
- PNBLSF (Passive NBLSF) in which the contents of the base cell cannot be changed due to the influence of an existing coupling noise in the neighborhood bit-lines. Uses an Eulerian sequence.

To sensitize all NBLSFs, read operations must follow write operations with a constant decoding time. As the same parallel testing method is used, one of the four cells marked with a fixed pattern is not utilized. This cell is always independent of the other three cells of one tiling group. Figure 11 shows the cells to be considered per test pattern. As the labeling period is four, the interference between the same labeled cells does not occur when read/write operations are implemented. Furthermore, it provides convenience for hardware implementation. For example, a bit-line decoder in the memory can also be used as a label selector by adding one more transistor every four bit-lines. Figure 11 also shows that the number of cells considered for NBLSFs detection is three. In the case when five-cell and nine-cell layouts are used, there are two and six redundant cells, respectively. However, the four-cell layout has only one redundant cell and is therefore more useful for NBLSFs detection.

V. Algorithm for NBLSF Detection

The number of test patterns for PNBLSF and ANBLSF detection are calculated by 2^{k-1} and $(k-2)\times 2^{k-1}$, respectively, and

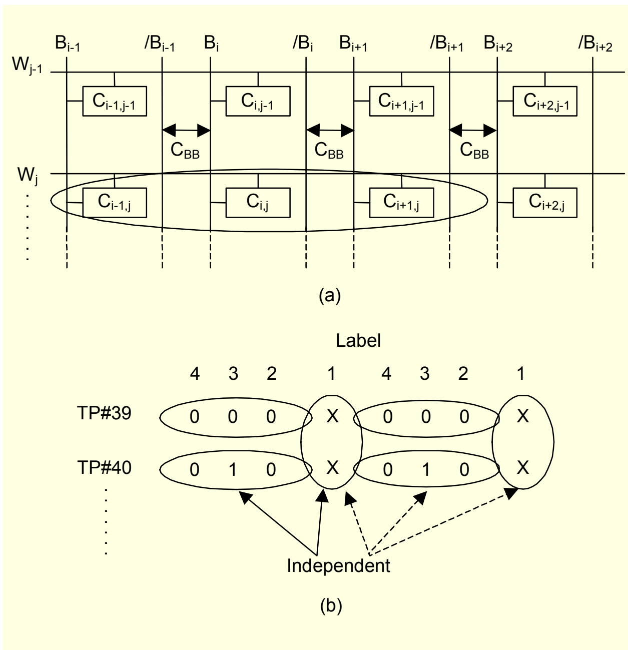


Fig. 11. Tiling method for NBLSFs detection.

```
Test procedure in detail for NBLSFs is as follows.
Step 1: Initialize memory cells with "0"
Step 2: TP[4,3,2,1]:=TP#n['B3"B2"B1"B0']
Step 3:
        for row:=0 to n-1 do
          begin
                    //WRITE ACTION
              //col:= Label number whose value is changed
              C[row,col]:=    TP[col];

                     // READ ACTION
              Data := Base cell
              if (C[row,col]<>Data) then output('Error at cell', cell);
          end;
Step 4: repeat steps 2 and 3 until next TP#1;
```

Fig. 12. Algorithm for NBLSFs detection.

the required total number of test patterns are $(k-1)\times 2^{k-1}$. However, the number of test patterns for NBLSFs, 64, is nearly equal to that of the number of test patterns for NPSFs because the period is four and labels 1, 2, 3, and 4 should each be the base cell in turn. Figure 12 shows the test procedure for NBLSFs. Read operations follow write operations. If there is an error after a read operation, the error sign rises simultaneously. The test procedure is based on the assumption that the interference of cells in different word-lines does not exist. The test complexity to detect the NBLSFs is given by

$$2 \text{ (non-transition \& transition)} \times 2 \text{ (read \& write)} \times \sqrt{n} \times 64 \text{ test patterns } = 256\sqrt{n}\,. \tag{6}$$

1. Passive Neighborhood Bit-Line Sensitive Fault Detection

To sensitize all the PNBLSFs, all the possible states of the cells should be written and directly read per one word-line. The number of PNPs is given by $4\times 2^{k-1}$ (number of base cells× PNPs). Table 5 shows all of the PNPs for PNBLSF detection.

2. Active Neighborhood Bit-Line Sensitive Fault Detection

ANBLSF detection is the same as for ANPSFs, except for the test procedure where a write operation must be directly followed by a read operation. The number of ANPs for ANPSF detection is given by $(k-2)\, 2^{k-1}$ per one base cell. Table 6 shows all of the ANPs for ANBLSFs detection.

As a four-cell physical layout is used, the number of test patterns required for the PNBLSF detection of Table 5 and the ANBLSF detection of Table 6 is 60. However, 64 test patterns are used for convenient testing despite being a small increase in the number of test patterns.

3. Sense Amplification Recovery Fault Detection

Sense amplifier recovery is a fault where the sense amplifiers get saturated after reading or writing a long string of bits of the same value (0/1); after reading such a long string from the same column, a memory with faulty (slow) sense amplifiers will not be fast enough to read the opposite value [4]. A test procedure for this fault is simply implemented by parallel testing. Table 7 shows the fault detection scheme. The test complexity is given by

$$2 \text{ cases} \times 2 (\text{1's complement data}) \times (4\sqrt{n-1} + 4 (\text{1's complement data}) \text{ for write operation} + 4 \text{ for read operation}) = 16\sqrt{n-1} + 32. \tag{7}$$

4. Address Faults Detection

An address decoder is a combination circuit that selects a unique RAM cell for each given RAM address. Assuming that a faulty address decoder does not become sequential in its operation, a faulty address decoder behaves in one of two ways [11]:

i) No-access: the decoder will not access the addressed cell. It may access non-addressed cell(s).
ii) Multiple-access: the decoder will access multiple cells, including the addressed cell.

In the case of no-access, the cell contains either an SA0 or SA1 fault. In the case of multiple-access, the fault is a RAM-matrix coupling fault between different cells. In other words, decoder faults manifest themselves as RAM-matrix faults, which cannot be tested by using conventional algorithms, e.g.,

Table 5. PNPs for PNBLSF detection.

Base cell	PNPs	Test patterns	Base cell	PNPs	Test patterns
Label 1	0 x 0 -	TP#1-2, TP#18-19	Label 2	x 0 - 0	TP#63-64, TP#64-1
	0 x 1 -	TP#22-23, TP#3-4		x 0 - 1	TP#2-3, TP#49-50
	1 x 0 -	TP#16-17, TP#15-16		x 1 - 0	TP#40-41, TP#61-62
	1 x 1 -	TP#13-14, TP#56-57		x 1 - 1	TP#51-52, TP#6-7
Base cell	**PNPs**	**Test patterns**	**Base cell**	**PNPs**	**Test patterns**
Label 3	0 - 0 x	TP#39-40, TP#62-63	Label 4	- 0 x 0	TP#19-20, TP#38-39
	0 - 1 x	TP#4-5, TP#41-42		- 0 x 1	TP#34-35, TP#17-18
	1 - 0 x	TP#58-59, TP#37-38		- 1 x 0	TP#8-9, TP#29-30
	1 - 1 x	TP#43-44, TP#12-13		- 1 x 1	TP#31-32, TP#32-33

note: 'x' means 'don't use.'

Table 6. ANPs for ANBLSF detection.

Base cell	ANPs	Test patterns	Base cell	ANPs	Test patterns
Label 1	0 x - 0	TP#63-64, TP#64-1	Label 2	x 0 0 -	TP#1-2, TP#18-19
	0 x - 1	TP#2-3, TP#49-50		x 0 1 -	TP#22-23, TP#3-4
	1 x - 0	TP#20-21, TP#57-58		x 1 0 -	TP#30-31, TP#7-8
	1 x - 1	TP#55-56, TP#14-15		x 1 1 -	TP#5-6, TP#26-27
	- x 0 0	TP#19-20, TP#38-39		x - 0 0	TP#39-40, TP#62-63
	- x 0 1	TP#34-35, TP#17-18		x - 0 1	TP#50-51, TP#33-34
	- x 1 0	TP#42-43, TP#21-22		x - 1 0	TP#4-5, TP#41-42
	- x 1 1	TP#23-24, TP#46-47		x - 1 1	TP#47-48, TP#48-49
Base cell	**ANPs**	**Test patterns**	**Base cell**	**ANPs**	**Test patterns**
Label 3	0 0 - x	TP#63-64, TP#64-1	Label 4	0 0 x -	TP#1-2, TP#18-19
	0 1 - x	TP#40-41, TP#61-62		0 1 x -	TP#30-31, TP#7-8
	1 0 - x	TP#20-21, TP#57-58		1 0 x -	TP#16-17, TP#15-16
	1 1 - x	TP#59-60, TP#28-29		1 1 x -	TP#9-10, TP#36-37
	- 0 0 x	TP#19-20, TP#38-39		0 - x 0	TP#39-40, TP#62-63
	- 0 1 x	TP#42-43, TP#21-22		0 - x 1	TP#50-51, TP#33-34
	- 1 0 x	TP#8-9, TP#29-30		1 - x 0	TP#58-59, TP#37-38
	- 1 1 x	TP#27-28, TP#60-61		1 - x 1	TP#35-36, TP#54-55

note: 'x' means 'don't use.'

Table 7. S/A recovery fault detection scheme.

Case for S/A recovery faults	Operations
1 : Write (a long string) - read	TP#7(write) - TP#13(read) TP#13(write) - TP#7(read)
2 : Read (a long string) - read	TP#13(read) - TP#7(read) TP#7(read) - TP#13(read)

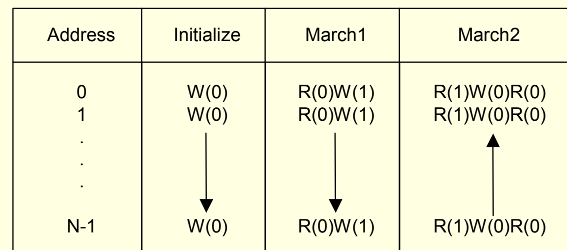


Fig. 13. 6n march test algorithm.

a 6n march test, as the test procedures show in Fig. 13. For the row address decoder test, only the first bit-line becomes active, as shown in Fig. 14. For the column address decoder test, only the first word-line becomes active. As a result of the decoder tests, the additional test size is given by

$$6\sqrt{n}\ \text{(column address decoder test)} + 6\sqrt{n}\ \text{(row address decoder test)} = 12\sqrt{n}. \quad (8)$$

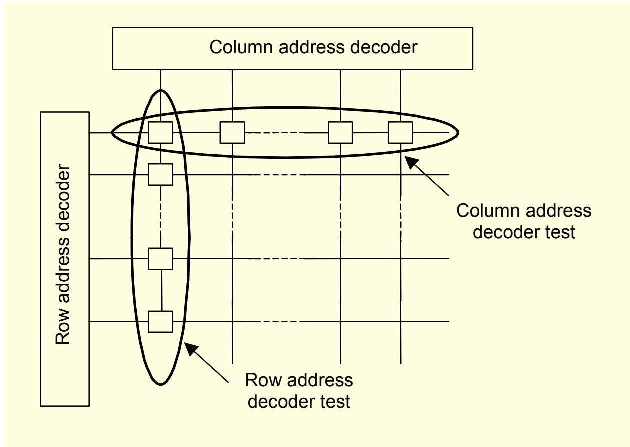


Fig. 14. Decoder test.

VI. Bist Architecture

Read and write operations are generally performed when the row address strobe (RAS) signal is driven low, prior to the time when the column address strobe (CAS) signal is driven low. On the contrary, the time to start testing the memory is when the CAS signal is driven low, prior to the RAS signal, as shown in Fig. 15. Here, it is assumed that memories do not use the CAS before RAS for other purposes, such as in auto-refresh mode.

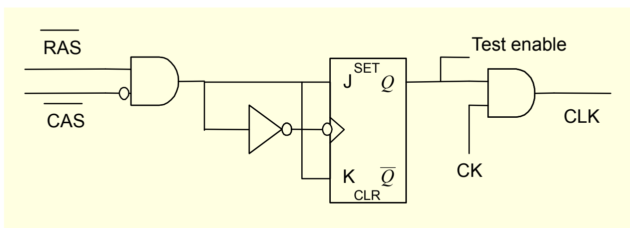


Fig. 15. Test enable circuit.

1. Test Pattern Generator

Figure 16 shows a functional unit, a test pattern generator (TPG). It generates 64 test patterns using an Eulerian sequence. Size 4 × 16 test patterns can be simply made by using a 4-bit gray code. For example, when C0 = 1 and C1 = C2 = C3 = 0, the first 16 patterns are generated, then B3 = G3, B2 = G2, B1 = G1, and B0 = G0. The patterns are provided through the data input pins.

2. Modified Decoder Circuit

Figure 17 shows the architecture and circuit diagram of the modified decoder, where an NMOS transistor is added per four bit-lines. The NMOS transistors are controlled by signal ϕ_4 every fourth bit-line. In the normal mode, ϕ_4 is '0', whereas ϕ_4 is '1' in the test mode. When E_a transistors are ON, $A_2, A_3, \ldots, A_{n-1}$ are useless. Therefore, the final output of the modified

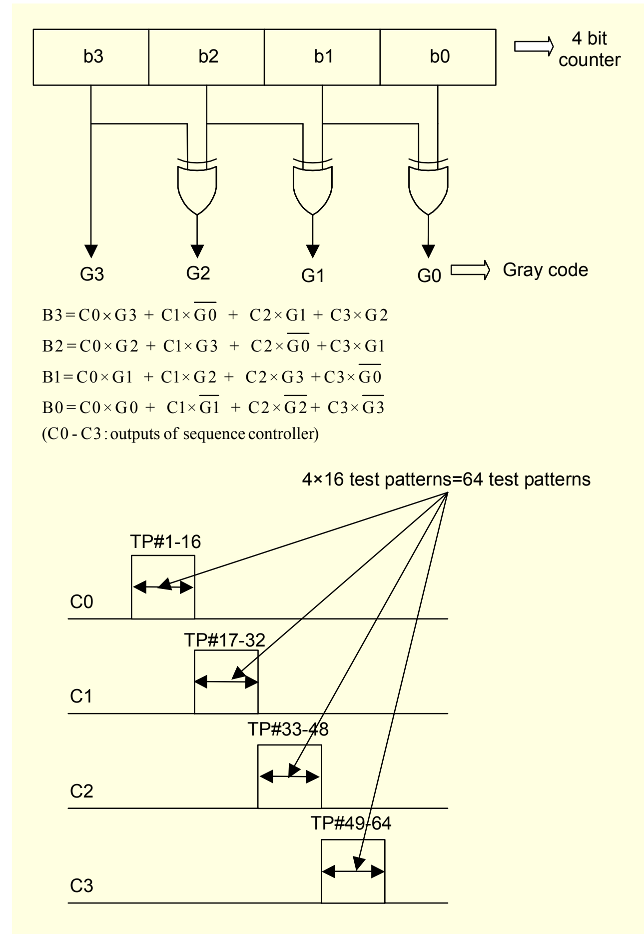


Fig. 16. Test pattern generator.

decoder is determined by A_0 and A_1. Address bits except A_0 and A_1 are decoded, and during phase ϕ_4, E_a (a = 0,1,…,(n/4−1), transistors can force the decoded logic value of $A_2A_3, \ldots$,and A_{n-1} to be driven by "0". Bits A_1 and A_0 of the address bits make it possible to select the cell numbered "k" (k∈{0,1,2,3}), concurrently. In consequence, the modified decoder can simultaneously read and write the selected data to all of the same labels. It should be noted that only the bit-line decoder is modified to allow multiple access of cells on the selected word-line. As the sense amplifier driver size is increased, it consumes more power. Also, due to the large gate capacitance, the sense amplifier slew rate decreases.

3. Parallel Comparator and Error Detector

Figure 18 shows the modified 'parallel comparator and error detector', where the inputs are even and odd bits. It shares the basic configuration of the circuits in [6]. However, we modify them to compare the contents of the same labeled in parallel. Each transistor P_{m-1} and T_{m-1} are connected to four PMOS transistors every fourth bit-line. According to the logic values of L_1, L_2, L_3

and L_4, labels 1, 2, 3 and 4 in a word-line can be selected and compared. In the normal mode, the logic values of L_1, L_2, L_3 and L_4 are all "1"s. The total transistors used for the modified "parallel comparator and error detector" are calculated by 1.5b (b + 0.5b, where b is the number of bit lines).

During phase ϕ_1, transistors P_1 to P_{m-1} and T_1 to T_{m-1} are pre-charged. During phase ϕ_2, the output of the coincidence detector is connected to the error amplifier through S_3. During phase ϕ_3, the error latch output is ERROR = 1, when S_1 and S_2 still remain cut-off, and the selected bit lines are not identical. If the bit lines are all '1s', S_1 conducts and S_2 remains cut-off. If the bit lines are all '0s', S_1 remains cut-off and S_2 conducts.

Figure 19 shows the BIST scheme in the memory structure. The test mode is set by the test enable circuit (the test enable unit shown in Fig. 15) when the CAS signal is driven low, prior to the RAS signal. Then, the TPG starts to generate patterns. The logic values selected are written and read to the selected cells in parallel by the modified decoder, which is shown in Fig. 17. If the contents of all the same labeled cells are not identical, the parallel comparator and error detector output becomes ERROR = "1".

The same algorithm is used to compare the four-cell layout with Type-1 and Type-2 neighborhoods. Table 8 shows that the four-cell layout has lower complexity than Type-1 and Type-2 neighborhoods and also requires fewer test patterns. The numbers of Type-1 and Type-2 are odd, whereas that of the proposed type is even. This proves a degree of simplicity to implement proper hardware for parallel testing. Table 9 describes the comparison of extra hardware. The parallel comparator consists of 1.5b + 12 transistors and the extra hardware in the modified decoder is b/4 transistors. Thus, the overall extra hardware is only 1.75b + 12.

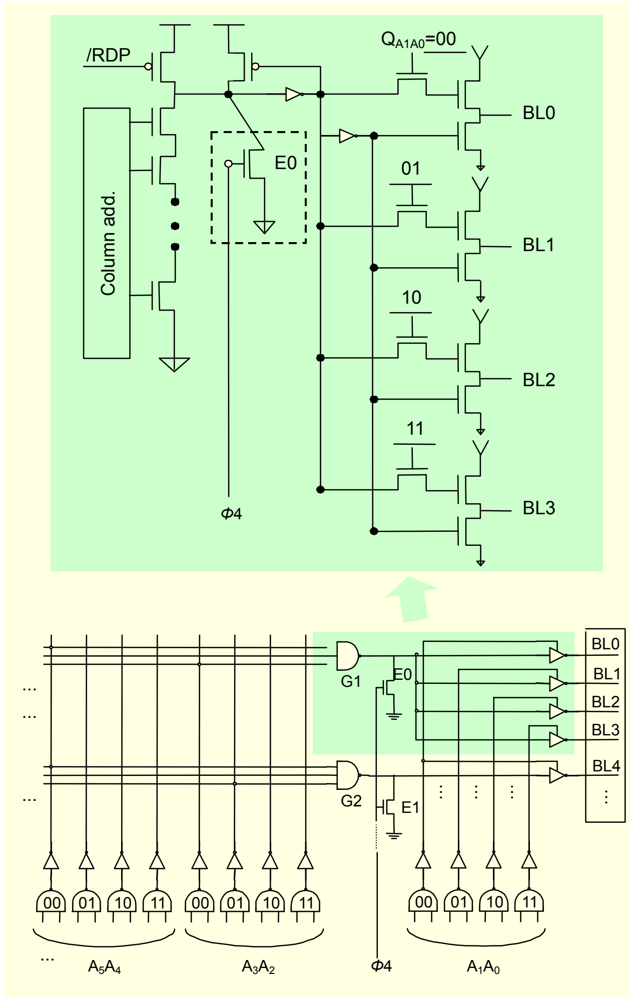


Fig. 17. Modified decoder.

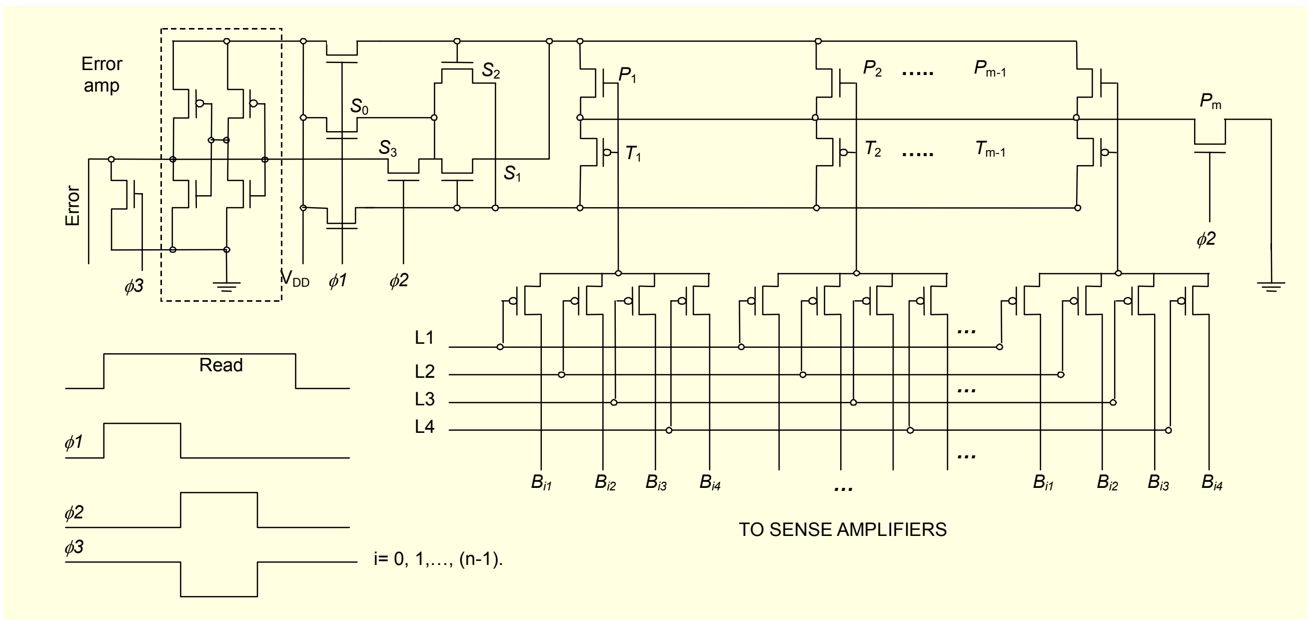


Fig. 18. Parallel comparator and error detector.

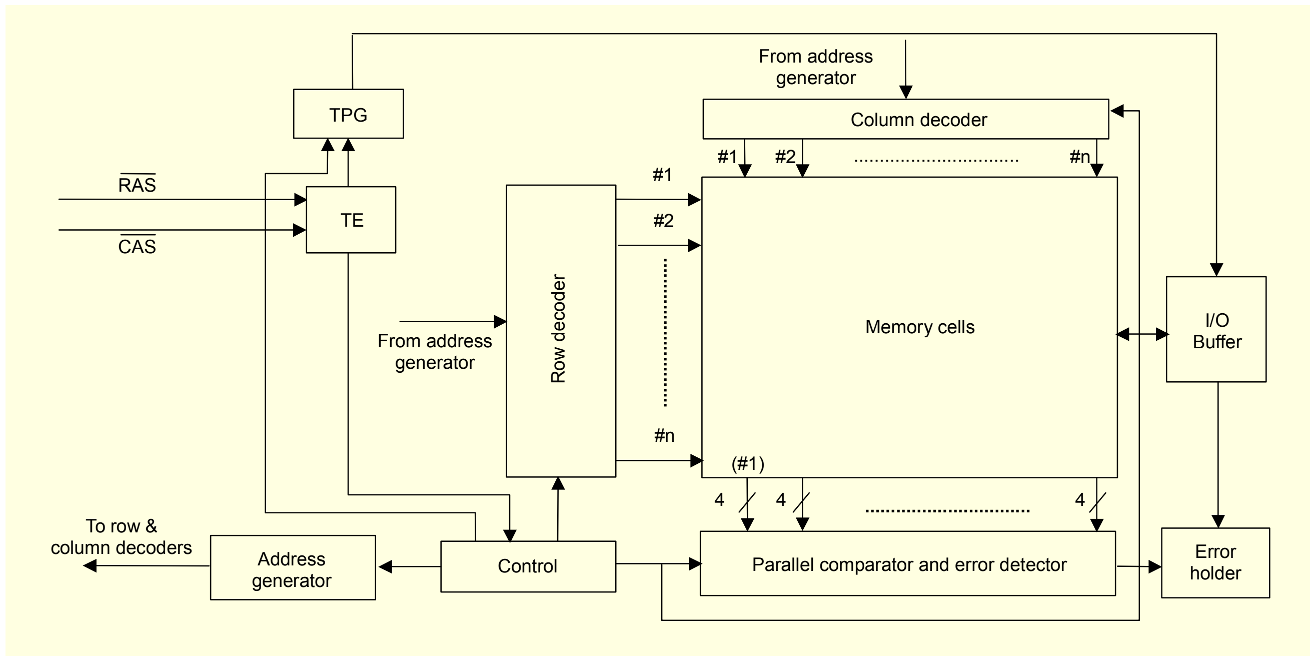


Fig. 19. Organization of BIST circuits in a memory.

Table 8. Comparison of the conventional types.

	Type-1	Type-2	Proposed type
Number of neighborhood cells (= k)	5	9	4
Test patterns (=k·2^k)	160	4,608	64
Transition faults	√	√	√
SNPSFs, ANPSFs, PNPSFs	√	√	√
Stuck-at faults	√	√	√
Stuck-open faults	√	√	√
Test complexity (= 2 (read & write) × no. of test patterns × $\sqrt{n}$)	$320\sqrt{n}$	$9216\sqrt{n}$	$256\sqrt{n}$ $128\sqrt{n} \times 2$ (groups A & B)
A degree of easiness to implement Hardware for parallel testing	Low	Low	High
ANBLSFs, PNBLSFs	Not used	Not used	√ √
Test complexity for NBLSFs (except initial value)	Not used	Not used	$128\sqrt{n}$ (Transition)

Table 9. Comparison of extra hardware.

	P. Mazumder [6]	This work
Transistors for parallel comparator	2b+12	1.5b+12
Transistors added to modified decoder	$2\log_2 b$	b/4

note: b is the number of bit lines

VII. Conclusions

A novel method called a 'four-cell physical neighborhood' has been proposed, which enables testing and considering NPSFs and NBLSFs in parallel. The method requires fewer test patterns and less test time than Type-1 and Type-2 neighborhoods. Additional circuits, i.e., a parallel comparator and an error detector are almost the same as those of P. Mazumder's. The proposed four-cell layout facilitates hardware implementation, requiring fewer transistors. Therefore, the overall chip area

expansion becomes certainly less than 0.4% [6]. As the density of a memory increases, coupling noise between bit-lines as well as interference between cells sharply increases. Most conventional test algorithms mask such coupling noise.

In order to achieve higher fault coverage including faults by bit-line coupling noise, we have also proposed a new fault model for bit-line coupling noise and a method to test them. Also, the proposed type and the BIST circuit are verified to be appropriate structures for testing NBLSFs. The test complexity is $256\sqrt{n}$ (non-transition & transition). The proposed tiling shape gives convenience to test S/A (sense amplifier) recovery faults whose test complexity is $16\sqrt{n}+32$. In addition, imbalance faults of bit-lines can be easily tested with the proposed tiling shape whose test complexity becomes O($\sqrt{n}$). This new tiling method is more effective than conventional ones in test time and overhead.

Acknowledgment

The authors thank God for His assistance with all their hearts.

References

[1] T.C. LO and M.R. Guidry, “An Integrated Test Concept for Switched-Capacitor Dynamic MOS RAM's,” *IEEE J. Solid-State Circuits*, vol. 12, Dec. 1977, pp. 693-703.

[2] J.P. Hayes, “Testing Memories for Single-Cell Pattern-Sensitive Faults,” *IEEE Trans. Comput.*, vol. 29, no. 3, Mar. 1980.

[3] J.P. Hayes, “Detection of Pattern-Sensitive Faults in RAMs,” *IEEE Trans. Comput.*, vol. 24, no. 2, Feb. 1975, pp. 150-157.

[4] A.K. Sharma, *Semiconductor Memories*, IEEE PRESS, 1996, pp. 151-154.

[5] D.S. Suk and S.M. Reddy, “Test Procedures for a Class of Pattern-Sensitive Faults in Semiconductor Random-Access Memories,” *IEEE Trans. Comput.*, vol. 29, no. 6, June 1980, pp. 419-429.

[6] P. Mazumder and J.H. Patel, “Parallel Testing for Pattern-Sensitive Faults in Semiconductor Random-Access Memories,” *IEEE Trans. Comput.*, vol. 38, no 3, Mar. 1989, pp. 394-407.

[7] M. Franklin and K.K. Saluja, “An Algorithm to Test RAMs for Physical Neighborhood Sensitive Faults,” *Proc. IEEE ITC 1991*, 1991, pp. 675-684.

[8] D.-C. Kang and S.-B. Cho, “Implementation of a Built-in Self Test Circuit for High Density Memory Test Using a New Tiling Method,” *Proc. ITC-CSCC'98*, July 1998, pp. 1781-1784.

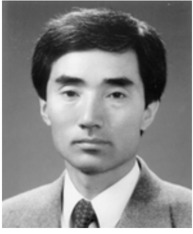
[9] D.-C. Kang, J.-H. Lee, and S.-B Cho, “A New Test Algorithm for Bit-Line Sensitive Faults in Super High-Density Memories,” *Proc. IEEE The Fifth Russian-Korean Int'l Sym. on Science and Technology*, vol. 1, 2001, pp. 198-201.

[10] H.-J. Yoo , *DRAM Design*, IDEC Press, 1996, pp. 13-71.

[11] M. Sachdev, “Open Defects in CMOS RAM Address Decoders,” *IEEE Design & Test of Computers*, vol. 14, no. 2, June 1997, pp. 26-33.

Dong-Chual Kang is a CTO of ISSELAH Co., LTD, Ulsan, Korea. He is currently working on automotive electronic system designs, and especially tries to apply a SF (Surface Fog) sensor which can automatically detect and remove fog on the interior windshield of a vehicle. He received the BS, MS, and PhD degrees from University of Ulsan, Korea in 1997, 1999, and 2003. He invented a SF (Surface Fog) sensor using a new innovative technology in March 2001 and received the excellence award in the 3rd IT Venture Conference in November 2001 that was supported by Korean Ministry of Information and Communication. He founded Dionyx Co., LTD for the business of the SF sensor in November 2001. His research includes the design and test of high-density memories, and automotive electronic systems including the SF sensor.

Sung Min Park received the BSc degree in electrical and electronic engineering from KAIST, Korea in 1993. He received the MSc degree in electrical and electronic engineering from University College London, UK in 1994 and the PhD degree in electrical and electronic engineering from Imperial College of Science, Technology and Medicine, UK in May 2000. From 1998 to 2000, he was a member of the research staff at Imperial College working on gigabit SiGe multi-channel optical receiver designs. From 2000 to 2001, he was a Senior Research Staff member at Satellite Technology Research Center at KAIST in Korea. From 2001 to 2002, he was a Research Professor at KAIST. From 2003 to 2004, he was with the School of Electrical Engineering at University of Ulsan as an Assistant Professor. In 2004, he joined the faculty of the Department of Information Electronics Engineering at Ewha Womans University in Seoul as an Assistant Professor. His research interests include high-speed analogue integrated circuit designs using sub-micron CMOS and SiGe technologies for optical and RF communication applications. He is a member of International Technical Program Committee at IEEE ISSCC 2005 (International Solid-State Circuits Conference) in the area of wireline communications.



Sang-Bock Cho is a Professor in the Department of Electrical Engineering, University of Ulsan (UOU), Ulsan, Korea. He has been with UOU since March 1986. Prior to joining UOU, he was a lecturer in the Department of Electronic Engineering Hanyang University, Seoul, Korea. Dr. Cho received the BS, MS, and PhD degrees from the Hanyang University in 1979, 1981, and 1985. He was a Visiting Scholar in Computer Engineering Research Center (CERC), University. of Texas at Austin from 1994 to 1995. Also, he had studied about VLSI CAD in University. of California at San Diego from 2003 to 2004. For the last few years, he has been working on various research projects on design and testing for CMOS VLSI and automotive electronic systems. He has been a member of IEEE since 1978 and a member of IEEK, KIPS and KISS. His research interests include SoC design and testing, DRAM design and testing, automotive electronic system design, and high precision vision systems.