論文 90-27-5-7

Built-In 테스트 방식을 이용한 RAM(Random Access Memory)의 고장 검출

(Fault Detection of Semiconductor Random Access Memories Using Built-In Testing Techniques)

金 倫 弘* 林 寅 七*

(Yun Hong Kim and In Chil Lim)

要 約

본 논문은 반도체 RAM의 기능 고장을 검출하기 위한 테스트 프로시듀어(test procedure)와 Built-In 자체 테스트 방식을 이용한 새로운 테스트 스킴(scheme)을 제안한다.

제안된 테스트 프로시듀어는 두가지로 구분되며, 그중 한 테스트 프로시듀어는 stuck-at 고장, 결합 고장 및 디코더 고장을 검출하기 위한 프로시듀어로서, 고장 검출을 위해 필요한 동작(operation)수를 기존의 테스트 프로시듀어보다 감소시켜서, 19N 개의 동작으로 고장검출이 가능하게 하였다. 또 다른 테스트 프로시듀어는 Hamilton 경로를 사용하여 필요한 write 동작수를 감소시키므로써 69N개의 동작으로 제약된 pattern-sensitive 고장(PSF)을 검출가능하게 하였다.

또 제안된 테스트 스킴은 Built-In 자체 테스트 방식을 이용하여 구성되고, 테스트 모드에서, I/O 핀에 의한 것보다 한번에 write 할 수 있는 셀수를 증가시키므로써 전체적인 write 수를 감소시킬 수 있어서 많은 테스트 시간을 절약할 수 있게 하였다.

Abstract

This paper proposes two test procedures for detecting functional faults in semiconductor random access memories (RAM's) and a new testing scheme to execute the proposed test procedures.

The first test procedure detects stuck-at faults, coupling faults and decoder faults, and requires 19N operations, which is an improvement over conventional procedures. The second detects restricted patternsensitive faults and requires 69N operations.

The proposed scheme uses Built-In Self Testing (BIST) techniques. The scheme can write into more memory cells than I/O pins can in a write cycle in test mode. By using the scheme, the number of write operations is reduced and then much testing time is saved.

I. 서 론

반도체 기술의 급속한 발전으로 단일 칩상에 집적되는 소자수가 증가되고 있다. 특히 RAM, ROM, PLA와 같은 규칙 구조 회로에서 소자밀도의 증가가 빠른 진전을 이루고 있는데, 그 중에서도 RAM이 가

*正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1990年 1月 6日

장 집적도가 높은 회로이다. 따라서 RAM을 테스트하는데 다음과 같은 문제들이 발생한다.[1-3] 많은 메모리 셀들이 단일 칩상에 집적되어 있으므로 다양한 형태의 고장이 발생하고, 메모리 셀 밀도에 비례하여 테스트 시간이 증가하게 되어, 대용량 메모리인 경우 많은 테스트 시간이 필요하게 된다. 따라서 RAM 테스트를 위해서는 D알고리듬이나 PODEM과 같은 일반적인 고장 검출 방법이 부적합하고, 위와같은 문제를 해결할 수 있는 RAM 칩에 적합한 테스트방식이 필요하다. 큰 기억용량을 갖는 RAM 내의 모든 고장을 짧은 시간내에 검출할 수 있는 고장 모델을 설정하여, 이에 대한 고장을 검출할 수 있는 최적의 테스트 프로시듀어를 설계하여야 한다.

최근 메모리 칩의 테스트를 위해서 Built-In 테스트(BIT)방식이 연구되고 있다.[6-10] 이와같은 BIT 방식을 사용하면 RAM 칩의 테스트가 자체적으로 이루어지므로, 별도의 복잡한 테스트 장비가 필요없게 된다. 따라서 테스트 비용이 절감될 수 있다.

일반적으로 널리 사용되는 RAM은 워드(word) 당 m 비트(bit) 이지만($m \geqq 1$), 본 논문에서는 편의상 특별히 언급이 없는 한 $m=1$로 가정한다.

본 논문에서는 RAM 칩에서 발생하는 일반적인 고장 모델을 효율적으로 검출할 수 있는 두가지의 테스트 프로시듀어와 Built-In 자체 테스트 스킴을 제안한다. 제안된 테스트 스킴은 한 write 싸이클동안에 복수개의 메모리 셀에 write 할 수 있어서 테스트시에 많은 시간을 절약할 수 있다.

Ⅱ. 고장 모델과 정의

RAM은 그림 1과 같이 메모리 셀 배열(memory cell array), 어드레스 디코더(address decoder), 메모리 어드레스 레지스터(MAR), 메모리 데이터 레지스터(MDR), 입출력 회로등으로 구성된다.

메모리 테스트에서 가장 널리 사용되는 고장 모델은 다음과 같다.

1. 메모리 셀 배열

(1) stuck-at 고장 : 메모리 셀이 지속적으로 0이나 1로 고정되어 있는 고장이다. 따라서 한 셀에 stuck-at 고장이 발생하면 그 셀은 어떠한 read나 write 동작에도 관계없이 X를 유지한다. (X는 0이나 1)

(2) 천이 고장 : 메모리 셀이 X'의 쓰기동작에도 불구하고 X에서 X'로의 천이가 불가능한 고장이다.

(3) 결합 고장 : 한 셀 Ci상에 0→1(또는 1→0)의쓰기동작을 했을 때 그 영향을 받아 다른 셀 Cj의 내용이 변화하는 경우 서로 다른 두 셀 Ci와 Cj에 결

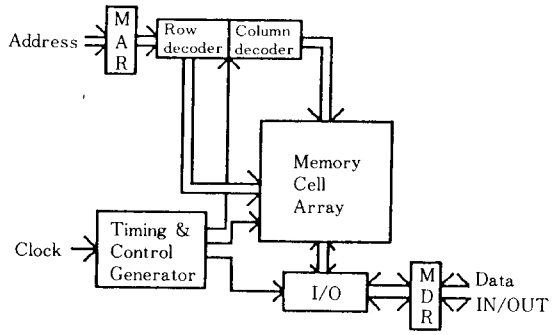


그림 1. 반도체 RAM의 일반 구성도
Fig. 1. General organization of a semiconductor RAM.

합 고장이 발생했다 라고 한다. 그러나, 이때 Cj 상의 쓰기동작이 Ci의 내용을 변화시킴을 의미하지는 않는다.

2. 어드레스 디코더

(1) no memory access

주어진 어드레스에 대하여 어떤 셀도 선택하지 않는다.

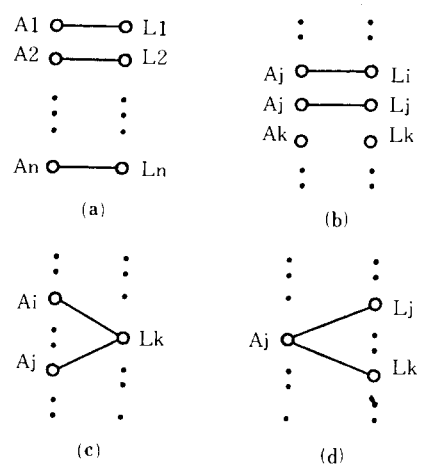


그림 2. 디코더의 3가지 고장형태
(a) 정상 동작
(b) no memory access
(c) many-to-one 형태
(d) one-to-many 형태

Fig. 2. Three types of decoder faults.
(a) normal operation,
(b) no memory access,
(c) many-to-one type,
(d) one-to-many type.

(2) many-to-one type memory access
여러 어드레스에 대하여 한 셀을 반복적으로 선택한다.

(3) one-to-many type memory access (또는 multiple memory access)
한 어드레스가 주어졌을 때, 복수개의 셀을 선택한다.

위의 3가지 고장 형태를 어드레스와 셀간의 대응관계로서 그림 2에 나타내었다.

3. 그외의 블럭(입출력 회로, MAR, MDR)

입출력회로, MAR, MDR에서 발생하는 고장은 메모리 셀 배열과 어드레스 디코더에서의 고장으로써 모델화가 가능하므로 별도의 고장 모델이 필요없다.

본 논문에서는 RAM의 테스트를 위해 행하는 동작을 기술하기 위해 다음과 같은 표기를 사용한다.

R : 한 셀에 대한 읽기동작
W : 한 셀에 대한 쓰기동작
W0 : 한 셀에 0를 쓰는 동작
W1 : 한 셀에 1을 쓰는 동작
↑ : 0상태의 셀에 1를 쓰는 동작
↓ : 1상태의 셀에 0을 쓰는 동작

[정의 1]

각 셀에 동일하게 수행되는 연속적인 동작들을 테스트 요소라 하고, 이 테스트 요소들의 연속적인 집합을 march 테스트라고 한다.

예를 들어, 어드레스의 순서대로 RW0↑을 각 셀에 수행한다면 RW0↑는 하나의 테스트 요소가되고, $(\overline{RW}, \underline{R\uparrow R}, \overline{RW1R})$과 같은 것은 march 테스트가 된다. 여기서 윗줄은 어드레스의 오름차순으로, 밑줄은 어드레스의 내림차순으로 수행됨을 나타낸다.

[정의 2]

임의의 셀 Ci의 상태는 S(Ci)로 표시한다. 이때 S(Ci)가 갖을 수 있는 값들은 0, 1, ↑, ↓이다.

[정의 3]

두 셀 Ci와 Cj(i<j)간의 결합 고장은 (S(Ci), S(Cj))로 표시한다. 여기서 Ci의 어드레스는 Cj 의 어드레스보다 작다.

예를 들어, Ci와 Cj(i<j)에 0과 s가 각각 기억되어 있다고 가정한다. (S는 0이나 1)만약 Ci에 W1을 하였을 때, Cj가 s에서 $\bar{s}$로 상태가 변하였다면, (↑,s)의 결합 고장이 Ci와 Cj에 발생하였다고 말한다. 이를 도식적으로 표현하면 그림 3과 같다.

여기서 Ci를 couplig 셀이라 하고, Cj를 coupled셀이라고 한다. 화살표는 coupling 셀에서 시작하여 coupled 셀에서 끝난다.

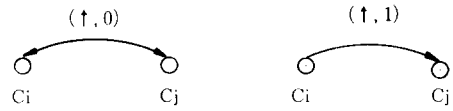


그림 3. (↑,s) 결합 고장(i<j)
(a) s=0 (b) s=1
Fig. 3. (↑,s) coupling fault. (i<j)
(a) s=0, (b) s=1.

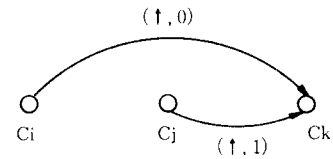


그림 4. 복잡한 결합 고장(i<j<k)
Fig. 4. Interacting coupling fault. (i<j<k)

한편 위와같이 한 coupled 셀이 한 coupling셀과 결합하지 않고 2개 이상의 coupling셀과 결합되었을 때는 새로운 문제가 발생한다. 그림 4가 그 한 예이다.

그림 4에서 coupled 셀 Ck가 Ci뿐만 아니라 Cj와도 결합되어, 결합고장(↑,0)와(↑,1)의 상호간섭으로 인하여 고장 마스킹(fault masking)현상이 발생할 수도 있다.

이것을 결합 고장의 상호작용(interacting) 이라고 한다. 그렇지 않으면 결합 고장의 비상호작용(noninteracting)이라고 한다.

4. pattern-sensitive 고장

소자밀도의 증가로 메모리 셀들이 더욱 근접함에 따라 셀간에 전자기적 영향이나 기생 커패시턴스, 전하 누설 등으로 인한 다른 셀들의 간섭에 의해 정보가 소실될 수 있다. 즉 어떤 셀의 내용이 주위 다른

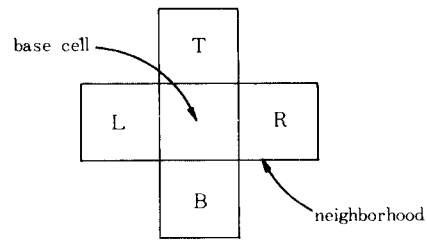


그림 5. 주변 셀과 기본 셀의 보기
Fig. 5. An illustration of neighborhood cells and base cell.

셀들의 상태(0, 1, ↑, ↓)의 조합인 어떤 패턴에 의해 바뀔 수 있다. 이러한 고장을 pattern-sensitive 고장(PSF)이라 한다.

일반적인 PSF를 검출하는 것은 많은 시간이 필요하기 때문에, 메모리 설계나 억세스 방법등을 고려하여 한 셀에 영향을 줄 수 있는 다른 셀의 범위를 보통 그림 5와 같이 주변 4개의 셀로 제한한다. 여기서 중앙의 셀을 기본 셀이라 하고 상하좌우 4개의셀을 주변셀이라 하며, 각각 T, B, L, R로 표시한다.

[정의 4]

주변 셀 T, R, B, L의 어떤 내용에 의해 기본 셀의 천이가 불가능하거나, 그 내용이 바뀌는 고장을 static PSF라고 한다.

[정의 5]

주변 셀 T, R, B, L의 패턴이 기본 셀의 내용을 바꾸어 놓을때, 그 패턴을 임계 고장 패턴(Critical Fault Pattern; CFP)이라고 정의한다.

본 논문에서는 [정의 4]에서 정의된 PSF만을 고려한다.

Ⅲ. 테스트 프로시듀어

본 논문에서 제안하는 테스트 프로시듀어는 stuck-at고장, 천이고장과 결합고장을 검출하는 프로시듀어 I과 static PSF를 검출하는 프로시듀어 II로 나뉜다.

1. 프로시듀어 I

stuck-at고장은 메모리내의 어떤 부분에서도 발생할 수 있지만, 메모리 셀 배열에서의 stuck-at 고장과 어드레스 디코더 고장만을 고려하여도 메모리내의 모든 stuck-at 고장을 검출할 수 있다.

표 1. 프로시듀어 I
Table 1. Procedure I.

	S0	S1	S2	S3	S4	S5
1	W0	R↑	R↓	R↑	R↓	R
2	W0	R↑	R↓	R↑	R↓	R
3	W0	R↑	R↓	R↑	R↓	R
•	•	•	•	•	•	•
•	•	•	•	•	•	•
N	W0	R↑	R↓	R↑	R↓	R
	S6	S7	S8	S9	S10	
1	R↑↓	R	W1	R↓↑	R	
2	R↑↓	R	W1	R↓↑	R	
3	R↑↓	R	W1	R↓↑	R	
•	•	•	•	•	•	
•	•	•	•	•	•	
N	R↑↓	R	W1	R↓↑	R	

time

메모리 셀 배열의 stuck-at 고장, 결합 고장, 어드레스 디코더 고장을 검출하기 위한 프로시듀어 I은 march 형태로서 다음과 같다.

$(\overline{W0}, \overline{R\uparrow}, \underline{R\downarrow}, \overline{R\uparrow}, \overline{R\downarrow}, \underline{R}, \overline{R\uparrow\downarrow}, \underline{R}, \overline{W1}, \overline{R\downarrow\uparrow}, \underline{R})$

프로시듀어I를 시간에 따라 나타내면 표 1과 같다.

[정리 1]

프로시듀어I은 메모리 셀 배열의 stuck-at 고장, 결합고장과 어드레스 디코더 고장을 완전히 검출한다.

[증 명]

1) 메모리 셀 배열에서 0상태와 1상태를 유지할수 있는지와 0에서 1로, 1에서 0으로의 천이가 가능한지를 보아야 한다. 따라서 각 셀에 0과 1을 쓰고, 각각 그값을 읽어보면 stuck-at 고장을 검출할 수 있다. (S0, S1, S2, S3에서 수행)천이고장은 각 셀에 ↑과 ↓을 수행하고, 각각 읽어보면 검출된다. (S1, S2, S3에서 수행)

2) 디코더 고장중에서 우선 no memory access는 메모리 셀 배열의 stuck-at고장 검출 과정에서 동시에 검출된다.

many-to-one 형태의 고장은 S1(R↑)에서 검출된다. 예를 들어 어드레스 Ai와 Aj(i<j)가 동일한 셀 Ck를 선택한다면, Ai에 의해 R↑이 Ck를 0에서 1로 바꾸었기 때문에 다음에 Aj를 통해 Ck의 내용을 읽어보면 0이 아닌 1이 읽혀진다. 따라서 many-to-one 형태의 고장을 검출할 수 있다.

one-to-many 형태의 고장은, 동시에 선택되는 셀들의 내용이 OR된 결과가 출력될 때는 S1(R↑)에 의해 검출되고, AND된 결과가 출력될 때는 S4(R↓)에 의해 검출된다.

3) 모든 가능한 비상호작용 결합고장과 그것을 검출하는 테스트 요소를 표 2에 나타내었다.

그러나 결합고장간에 상호작용이 있을 때는 표2의 테스트 요소로 검출이 불가능한 고장이 발생할 수있

표 2. 비상호작용 결합고장과 검출 테스트 요소
Table 2. Noninteracting coupling faults and test elements.

	Fault	Sequence		Fault	Sequence
1	(↑, 0)	S1($\overline{R\uparrow}$)	5	(1, ↑)	S1andS2($\overline{R\uparrow}$ and $\underline{R}$)
2	(1, ↓)	S2($\underline{R\downarrow}$)	6	(↓, 0)	S2andS3($\underline{R\downarrow}$ and $\underline{R}$)
3	(0, ↑)	S3($\underline{R\uparrow}$)	7	(↑, 1)	S3andS4($\underline{R\uparrow}$ and $\overline{R}$)
4	(↓, 1)	S4($\overline{R\downarrow}$)	8	(0, ↓)	S4andS5($\overline{R\downarrow}$ and $\underline{R}$)

는데, 표 3에 그런 고장과 그것을 검출하는 테스트 요소를 나타내었다.

표 3. 상호작용 결합고장과 검출 테스트 요소
Table 3. Interacting coupling faults and test elements.

Interacting Fault	Invalid Sequence in Table 2	Necessary Sequence	Proposed Test Procedure
(↑, 0) (↑, 1)	1	$\overline{R\downarrow\uparrow}$	S9
(0, ↓) (1, ↓)	2	$\overline{R\uparrow\downarrow}$, or $\overline{R\uparrow\downarrow}$ and R	S6 and S7
(1, ↑) (0, ↑)	3	$\underline{R\downarrow\uparrow}$	S9
(↓, 0) (↓, 1)	4	$\overline{R\uparrow\downarrow}$	S6
(1, ↑) (0, ↓)	5	$\overline{R\uparrow\downarrow}$ and $\underline{R}$	S6 and S7
(0, ↓) (1, ↑)		$\overline{R\downarrow\uparrow}$ and $\underline{R}$	S9 and S10
(↓, 0) (1, ↑)	6	$\overline{R\uparrow\downarrow}$	S6
(↑, 1) (↓, 0)	7	$\overline{R\downarrow\uparrow}$ and $\underline{R}$	S9 and S10
(↓, 0) (↑, 1)		$\overline{R\uparrow\downarrow}$ and $\underline{R}$	S6 and S7

프로시듀어 I은 표 3에 나열된 테스트 요소를 모두 포함하고 있으므로 상호작용 결합고장도 검출할 수 있다. 그러므로 프로시듀어 I은 static PSF를 제외한 앞절에서 모델화된 모든 고장을 검출하는 완전한 테스트 프로시듀어이다. (증명 끝)

프로시듀어I이 전체 메모리를 테스트하기 위해 필요한 동작수는 19N이다. 여기서 N은 메모리내의 전 메모리 셀 수이다. 이것은 프로시듀어I과 같은 범위의 고장을 검출하는 기존의 테스트 프로시듀어와 비교해 볼 때 훨씬 효과적이다.

2. 프로시듀어 II

기본셀에서의 static PSF를 검출하기 위해서는 4개의 주변셀 T, R, B, L에 모든 가능한 16(=2^4) 개의 CFP를 인가하고, 각 CFP 마다 기본셀에 R↑R↓의 연속적인 4동작을 수행해야 한다. 메모리의 static PSF검출을 위해 필요한 동작수를 줄이기 위해서는 한 CFP를 인가할 때 최소한의 white 동작을 사용하여 다음 CFP를 얻을 수 있도록 하는 것이다. T, R, B, L의 내용을(TRBL)로 나타낸다고 할 때 한 CFP (0001)에서 다음 CFP가(0111)일 경우 셀 R과 B에 1을 인가하기 위해서는 2개의 W1동작이 필요하다. 그러나 인가될 CFP의 순서를 적절히 조절하여, CFP 간의 Hamming 거리가 1이 되도록 한다면 16개의 CFP를 인가하기 위해서 16개의 write 동작만이 필요하다. 이를 위해 16개의 CFP를 노드(node)로 하고 Hamming 거리가 1인 CFP 간에 아크(arc)를 연결하여 그래프를 구성한 후에, 그래프내의 모든 노드를 한번씩 통과하는 경로인 Hamilton 경로를 구한다면 그 경로상에 있는 각 노드의 순서가 바로 CFP의 인가순서가 된다. 이와같이 구한 CFP 씨이퀀스는 표 4와 같다.

표 4. CFP 씨이퀀스
Table 4. A sequence of CFP's.

	TBLR	Base		TBLR	Base
1	0 0 0 0	R↑R↓	9	1 1 0 0	R↑R↓
2	0 0 0 1	R↑R↓	10	1 1 0 1	R↑R↓
3	0 0 1 1	R↑R↓	11	1 1 1 1	R↑R↓
4	0 0 1 0	R↑R↓	12	1 1 1 0	R↑R↓
5	0 1 1 0	R↑R↓	13	1 0 1 0	R↑R↓
6	0 1 1 1	R↑R↓	14	1 0 1 1	R↑R↓
7	0 1 0 1	R↑R↓	15	1 0 0 1	R↑R↓
8	0 1 0 0	R↑R↓	16	1 0 0 0	R↑R↓

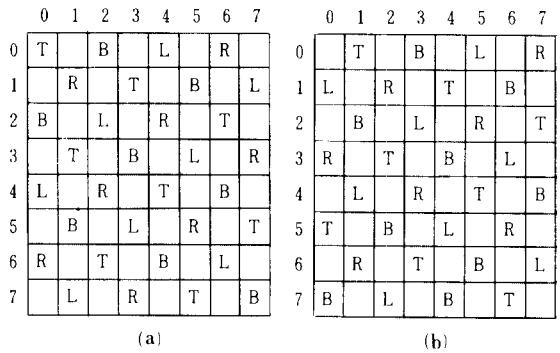


그림 6. 셀의 할당
(a) 집합 A (b) 집합 B
Fig. 6. Cell assignment.
(a) Set A, (b) Set B.

표 4의 CFP 씨이퀀스는 하나의 기본셀만을 고려하였으므로, 실제로 전체 메모리 셀 배열에 적용하기 위해서는 배열내의 각 메모리 셀들을 기본셀로 하여 CFP를 인가해야 한다. 이를 체계적이고 효과적으로

수행하고 어드레스 씨이퀀스 생성을 용이하게 하기 위해 그림 6과 같이 메모리 셀 배열을 두 집합으로 나누어 주변셀을 할당한다. 각 집합에서 심볼이 없는 셀이 기본셀이다.

여기서 메모리셀이 i번째 행과 j번째 열에 위치할 때 어드레스를 (i, j)로 나타내고, 셀 배열에서 상단좌측 셀의 어드레스를 (0, 0)로 하여 연속적으로 행과 열에 따라서 어드레스가 증가한다고 가정한다.

프로시듀어 II는 그림 6의 두 집합에 대하여 다음과 같이 수행된다.

(프로시듀어 II)

[단계 1] 모든 메모리 셀을 0으로 초기화한다.

[단계 2] 집합A의 셀에 대하여

for i=1 to 16 {

CFP i에 따라 T, R, B, L에 write 동작을 한다;

각 기본셀에 R↑R↓을 수행한다;}

T 셀에 0을 쓴다;

[단계 3] 집합B의 셀에 대하여

단계 2를 반복한다;

프로시듀어 II는 [정의4]에서 정의된 메모리 셀 배열내의 static PSF를 검출하기 위해 단계 1에서 N개의 동작, 단계2, 3에서 각각 16×(N/8+N/2×4)개의 동작을 사용하여 총 69N개의 동작이 필요하다. 위와 같이 프로시듀어 II를 수행하게 되면 모든 셀 주변에 필요한 16개의 모든 CFP가 인가되고 해당셀에 R↑R↓ 동작이 수행되므로 모든 static PSF가 검출될 수 있다. 그러나 69N개의 동작수는 다음절에서 제안되는 테스트 스킴을 사용하여 감소될 수 있다.

Ⅳ. Built-In 테스트 스킴

앞에서 제안된 프로시듀어를 효과적으로 수행하기 위해 그림 7과 같은 Built-In테스트 스킴을 제안한다.

기존의 메모리 회로에 부가되는 하아드웨어는 다음과 같이 3개의 모듈로 나뉜다.

(1) 어드레스 생성기 (Address Generator; AG)

(2) 응답 압축기 (Compactor)

(3) 병렬 write 논리블럭(Parallel Write Logic; PWL)

제안된 테스트 스킴은 프로시듀어 I과 II를 수행한다. 프로시듀어I은 march 테스트이기 때문에 어드레스가 1씩 증가하거나 감소하므로 어드레스 생성기는 기본적으로 카운터를 사용하여 구성된다. PWL의주요 기능은 write 동작을 동시에 수행할 수 있도록 하는 것이다. 일반적인 RAM은 한 싸이클동안에 하나의 셀에 write 동작을 수행할 수 있지만, 제안된

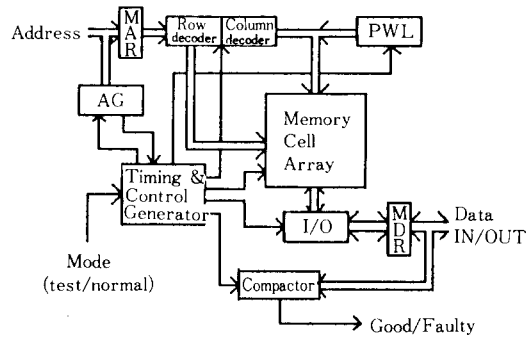


그림 7. 제안된 테스트 스킴의 구성도
Fig. 7. Block diagram of the proposed testing scheme.

테스트 스킴은 테스트(프로서듀어II 수행)시에 열 디코더(column decoder)대신 PWL을 이용하여 여러 개의 셀을 동시에 억세스할 수 있도록 하므로써 한 싸이클에 복수 개의 셀에 write 동작을 수행할 수있다. 따라서 많은 테스트 시간을 절약할 수 있다. 위의 3모듈은 논리레벨(logic-level)에서 설계되었다.

1. 어드레스 생성기

어드레스 생성기의 기본 기능은 필요한 어드레스를 체계적으로 생성하는 것이다. AG는 플립플롭, 2:1 MUX, 기본 게이트 등으로 구성되며, 논리도는 그림 8과 같다.

AG는 모드 1, 2, 3으로 나뉘어 동작하는데 각 모드는 제어신호 M1, M2에 의해 표 5와 같이 선택된다.

모드1에서 AG는 프로시듀어I의 읽기와 쓰기동작을 위해서 선형적으로 어드레스를 생성한다. 이 모드에서 AG는 단순한 업-다운 카운터(up-down counter)로 동작한다.

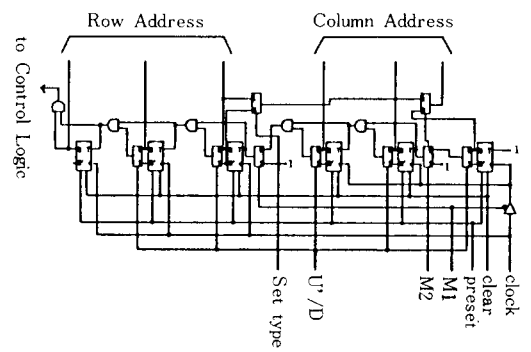


그림 8. 어드레스 생성기의 논리도
Fig. 8. Logic diagram of Address Generator.

표 5. AG의 동작 모드
Table 5. Operation mode of AG.

M1	M2	동작 모드
0	0	모드 1
0	1	모드 2
1	*	모드 3

*는 don't care

모드2에서 AG는 프로시듀어 II의 read 동작 수행 중 모든 기본셀의 어드레스를 생성한다. 선택될 집합(A 또는 B)은 제어신호 Set에 의해 결정된다. AG는 두 부분으로 나뉜다. 한 부분은 행 어드레싱(row addressing)을 위한 것이고 또 다른 부분은 열 어드레싱(column addressing)을 위한 부분이다. 집합A의 경우에 짝수행 상의 기본셀은 홀수열 상에 놓이고, 홀수행 상의 기본셀은 짝수열 상에 놓이게 된다. 한편 집합B의 경우에 짝수행 상의 기본셀은 짝수열 상에 놓이고 홀수행 상의 기본셀은 홀수열 상에 놓인다. 그러므로 집합A의 기본셀의 어드레스를 생성하는 동안에는 열 어드레스의 LSB(least significant Bit)는 행 어드레스의 LSB의 보수가 된다. 또한 집합 B의 기본셀에 대해서는 행 어드레스의 LSB는열 어드레스의 LSB로서 사용된다.

모드3에서 AG는 단지 행 어드레스만을 생성한다. 이때의 열 어드레스는 병렬 쓰기 동작을 위해 PWL에 의해서 적절히 생성된다.

2. 응답 압축기

응답 압축기는 인가된 테스트 패턴에 대한 테스트 응답을 압축하여 놓는 논리 블럭이다. 응답 압축기의 논리도는 그림 9와 같다.

그림 9의 응답 압축기는 기존에 제안된 압축기[10]와는 달리 압축함수가 필요없고 어떤 형태의 카운터도 사용되지않고 하나의 플립플롭과 기본 게이트들로 실현된다. 프로시듀어 I의 정상적인 출력 씨이퀀스는(0**N)이나 (1**N)이 된다. (여기서 N은 전체 셀의 수).

그림 9에서 출력은 제어신호 N'/C에 의해 인버터나 버퍼를 거쳐서 D-플립플롭에 전달된다. 출력 씨이퀀스가(0**N)일 경우에는 N'/C는 0가 되어 출력이 바로 D-플립플롭에 전달되고, (1**N)인 경우에는 N'/C가 1로 고정되어 출력의 보수값이 전달되도록 한다. 그러므로 출력 씨이퀀스가 정확하다면 D-플립플롭의 최종 상태값은 두 형태의 출력 씨이퀀스에 대해 항상 0가 된다.

만약 메모리가 잘못된 테스트 응답을 출력하였다면 D-플립플롭의 상태값(Q)은 1이 되므로 Q'은 0가 되어 AND 게이트는 클럭을 봉쇄하게 된다. 따라서 D-플립플롭에 1값이 저장된 이후에는 클럭이 전달안되므로 어떠한 입력 D에도 영향을 받지않고 잘못된 1값을 유지하게 된다. 결국 플립플롭의 상태 g'/f만을 관측하므로써 출력 씨이퀀스의 오류를 검출할 수 있다.

3. 병렬쓰기 논리블럭

PWL은 병렬패턴쓰기(Parallel Pattern Writer; PPW)와 쉬프트 레지스터 로더(Shift Register Loader;SRL)로 구성된다. PPM의 기능은 행 어드레스 디코더에 의해 선택된 행의 셀들에 대하여 병렬로 쓰기동작을 수행하는 것이다. PPW는 m개의 쉬프트 레지스터들로 구성되는데, 각 쉬프트 레지스터는 전단의 플립플롭에 연결된 마스터 슬레이브 플립플롭이다. 여기서 m은 메모리 배열에서 열(비트선)의 수가 된다. PPW는 열 디코더전에 위치하며 모든 열에 연결되어 있다. PPW의 한단은 그림10과 같이 구성된다.

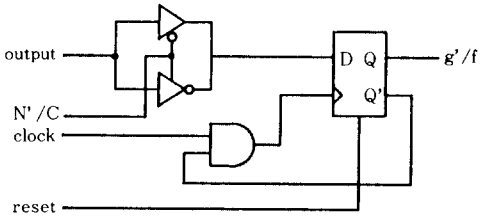


그림 9. 응답 압축기
Fig. 9. Compactor.

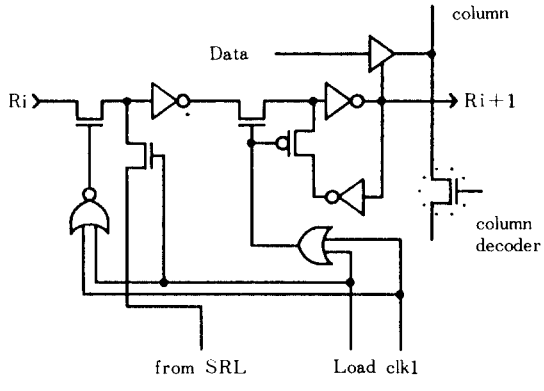


그림10. PPW의 한 단에 대한 논리 다이어그램
Fig. 10. Logic diagram for one stage of PPW.

한 단이 1값을 저장하고 있을 때, 그 단에 연결된 열이 선택된다. 따라서 필요하다면 PPW의 복수개의 단이라도 1값으로 고정하여, 복수개의 열을 자유롭게 선택할 수 있다. 입력신호인 Data와 Load는 PPW에 대한 병렬초기화를 위해 사용된다. Data 신호는 SRL로 부터 생성된다.

PPW는 clk1에 따라 PPW의 새로운 상태를 만들기 위해 쉬프트된다. 예를 들어, 그림 6 (a)에서 PPW가 행0에서 T로 할당된 셀들을 선택하기 위해 로드된 후에 행1에서 T로 할당된 셀들을 선택하기 위해서는 오른쪽으로 세번 쉬프트하면 된다. 이를 위해서 clk1에 3개의 클럭이 필요하다.

한편 SRL은 PPW를 로드하기 위해 사용된다. SR의 기능은 선택될 Data 신호를 생성하는 것이다. SRL의 논리 다이어그램은 그림11과 같다.

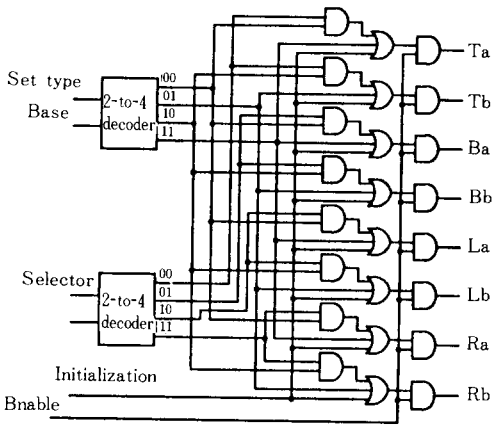


그림11. 그림 7의 메모리배열에 대한 SRL
Fig. 11. A SRL for the memory array in Fig. 7.

출력 심볼에서 첨자는 집합협태를 나타낸다.Enable 신호는 병렬쓰기동작이 필요할 때마다 1로 세트된다. 모든 메모리 셀이 초기화되어야만 할 때는 PPW 의 모든 단에1이 로드되도록 Initialize신호가 1로 세트되어 모든 열이 선택된다. 그리고 행 어드레스 디코더에 의해 모든 행이 순차적으로 하나씩 선택된다. 프로시듀어II에서 두 집합의 각 기본 셀에 R↑R↓을 수행할 때는 Base 신호를 사용하여 기본셀을 선택한다.

V. 결과 비교 및 검토

[12]에서 Thatte에 의해 제안된 알고리듬 A는 stuck-at고장, 천이 고장, 결합 고장을 검출하는데 30N의 동작이 필요하였다. 또한 [5]에서는 stuck-at 고장, 천이 고장, 디코더 고장, 입출력회로상의 고장, 비상호작용 결합 고장을 검출하는데 36N의 동작이 필요하였다. 프로시듀어 I과 다른 테스트 절차간의 비교를 위해 표 6에 4R*1, 16K*1, 64K*1 RAM의 테스트시 소요시간을 나타내었다. 여기서 read 싸이클 시간과 write 싸이클 시간은 같으며 한 싸이클 시간은 100ns라고 가정한다.

표 6. 여러가지 메모리에 대한 테스트 시간
Table 6. Testing time for some memories.

Type of Test Procedure	Time Complexity	4K*1 RAM	16K*1 RAM	64K*1 RAM
Proposed Test Procedure	19N	7.6ms	30.4ms	121.6ms
Sahgal's Test Procedure	36N	14.4ms	57.6ms	230.4ms
Thatte's Algorithm	30N	12ms	48ms	192ms
GALPAT	4N*N	6.4s	102.4s	1638.4s

제안된 테스트 스킴에서 각 모듈의 논리 다이어그램은 메모리 칩마다 종속적인 부분이 많기 때문에 타이밍과 딜레이 시뮬레이션은 수행하지 않고 논리 레벨 시뮬레이터인 HILO-3상에서 논리 시뮬레이션만을 수행하였다.

제안된 테스트 스킴을 사용할 경우에 많은 테스트 시간을 절약할 수 있는데, 그 예로써 몇가지 메모리에 대한 초기화에 필요한 시간을 표 7에 나타내었다.

표 7. 몇가지 메모리에 대한 초기화 시간
Table 7. Times taken for initialization.

Memory	Array	Time taken for initialization	
	row * column	existing scheme	proposed scheme
16K	64*256	1.638ms	0.006ms
	128*128	1.638ms	0.013ms
	256*64	1.638ms	0.026ms
64K	128*512	6.554ms	0.013ms
	256*256	6.554ms	0.026ms
	512*128	32.768ms	0.051ms

VI. 결 론

본 논문에서는 두가지 테스트 프로시듀어가 제안되었다. 프로시듀어 I은 stuck-at 고장, 결합 고장 및

더코더 고장을 검출하기 위해 필요한 동작수를 기존의 테스트 프로시듀어보다 감소시켜서, 19N 개의 동작으로 고장검출이 가능하게 하였다. 프로시듀어 II는 본 논문에서 정의된 static pattern-sensitive 고장을 검출하기 위해 Hamilton 경로를 사용하여 필요한 write 수를 감소시키므로써, 테스트 씨이퀀스가 짧아지고 테스트 시간이 절약된다.

또한 Built-in 자체 테스트 방식을 이용한 새로운 테스트 스킴이 제안되었다. 프로시듀어 I과 II가 제안된 스킴에 의해 수행되며, 테스트 모드에서 복수개의 셀을 병렬로 동시에 write 할 수 있기 때문에 많은 테스트 시간을 절약할 수 있다. 또한 제안된 테스트 스킴내의 응답 압축기는 하나의 플립플롭과 간단한 기본 게이트로만 구성되기 때문에 기존의 방법보다 회로가 매우 간단하므로 부가회로의 면적이 감소될 수 있다.

參 考 文 獻

[1] E.R. Hnatek, "4-Kilobit memories present a challenge to test," *Computer Design,* vol 14, pp. 117-125, May 1975.

[2] J.P. Hayes, "Detection of pattern-sensitive faults in random access memories," *IEEE Trans. Comput.*, vol. C-34, pp. 150-157, Feb. 1975.

[3] D.S. Suk and S.M. Reddy, "Test procedures for a class of patternsensitive faults in semiconductor random-accss memories," *IEEE Trans. Comput.*, vol. C-29, pp. 419-429, June 1980.

[4] D.S. Suk and S.M. Reddy, "A march test for functional faults in semiconductor random access memories, " *IEEE Trans. Comput.*, vol. C-30, pp. 982-985, Dec. 1981.

[5] C.A. Papachristou and N.B. Sahgal, "An improved method for detecting functional faults in semiconductor random access memories," *IEEE Trans. Comput., vol.* C-34, pp. 110-116, Feb. 1985.

[6] K.K. Saluja, S.H. Sng, and K. Kinoshita, "Built-in self-testing RAM: a practical alternative," *IEEE Design & Test,* pp. 42-51, Feb. 1987.

[7] K. Kinoshita and K.K. Saluja, "Built-in testing of memory using on-chip compact testing scheme," *Proc. Int'l Test Conf.*, pp. 271-281, Oct. 1984.

[8] Z. Sun and L.T. Wang, "Self-testing of embeddd RAMs," *Proc. Int'l Test Conf.*, pp. 148-156, Oct. 1984.

[9] S.K. Jain and C.E. Stroud, "Built-in self testing of embedded memories," *IEEE Design & Test,* pp. 27-37, Oct. 1986.

[10] K. Kinoshita and K.K. Saluja, "Built-in testing of memory using an on-chip compact testing scheme," *IEEE Trans. Comput.*, vol. C-35, pp. 862-870, Oct. 1986.

[11] Knaizuk, Jr., and C.R.P. Hartmann, "An optimal algorithm for testing stuck-at faults in random access memories," *IEEE Trans. Comput.*, vol. C-26, pp. 1141-1144, Nov. 1977.

[12] R. Nair, S.M. Thatte, and J.A. Abraham, "Efficient algorithms for testing semiconductor random-access memories," *IEEE Trans. Comput.*, vol. C-27, pp. 572-576, June 1978.

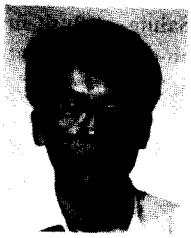
[13] V.P. Srini, "API tests for RAM chips," *IEEE Computer,* vol. 10, pp. 32-36, July 1977.

[14] W.H. McAnney and V.P. Gupta, "Random testing for stuck-at storage cells in an embedded memory," *Proc. Int'l Test Conf.*, pp. 157-166, Oct. 1984.

[15] T. Sridha, "A new parallel test approach for large memories," *IEEE Design & Test,* pp. 15-22, Aug. 1986.

[16] 김윤홍, 한석붕, 정준모, 임인칠, "반도체 RAM의 고장검출을 위한 테스트 절차," 대한전자공학회 추계학술발표대회, vol. 10, no. 1, pp. 796-800, 1987.

著 者 紹 介



金 倫 弘(正會員)

1964年 5月 5日生. 1986年 2月 한양대학교 전자공학과 졸업. 1988年 2月 한양대학교 대학원 전자공학과 졸업(공학석사). 1988年 3月～현재 한양대학교 대학원 전자공학과 박사과정 재학중. 주관심분야는 VLSI Logic Testing, Testable Design 및 Test Generation 등임.

林 寅 七 (正會員) 第25卷 第8號 參照
현재 한양대학교 전자공학과 교수