내장된 이중 포트 메모리 테스트를 위한 CM2 테스트 알고리즘

(CM2 Test Algorithm for Embedded Dual Port Memory)

양 선 웅 † 장 훈 ††
(Sunwoong Yang) (Hoon Chang)

요 약 본 논문에서는 10N March 테스트 알고리즘에 기반한 내장된 이중 포트 메모리를 위한 효율적인 테스트 알고리즘을 제안하였다. 제안된 알고리즘은 각각의 포트에 대해 독립적으로 테스트 알고리즘을 적용함으로써 각각의 포트에 대해서 단일 포트 메모리 테스트 알고리즘을 적용하는 방법에 비해 시간 복잡도를 20N에서 8.5N으로 시간 복잡도를 줄였다. 그리고 제안된 알고리즘은 주소 디코더 고장, 고착 고장, 천이 고장, 반전 결합 고장, 동행 결합 고장을 모두 검출할 수 있다.

Abstract In this paper, we propose an efficient memory test algorithm based on 10N march for embedded dual-port memories. Unlike the existing method which tests each port independently, the proposed algorithm tests all ports simultaneously, which results in a reduction of the time complexity from 20N to 8.5N. The proposed test algorithm can be used to completely detect address decoder faults, stuck-at faults, transition faults and coupling faults which are major target faults in embedded memories.

1. 서 론

최근 들어 하나의 칩에 대한 회로의 집적도는 시스템의 고성능화, 고기능화 및 소형화 요구와 함께 설계, 공정 기술의 발달에 힘입어 급속하게 증가되고 있다. 이에 따라, 제한된 면적 안에 더 많은 트랜지스터를 집적시킬 수 있게 되었으며, 칩의 기능을 더욱 향상시키기 위해 예전에는 칩 외부에 배치하였던 메모리 같은 모듈들도 이제는 칩 안에 내장되는 추세이다.

내장된 메모리로는 리프레쉬가 필요 없는 다중 포트 비동기식(multi-port asynchronous) SRAM이 가장 많이 사용되고 있고[1], 대부분의 다중 포트 메모리 테스트를 위한 알고리즘들은 다중 포트 메모리를 여러 개의 단일 포트 메모리로 간주하고, 각각의 포트에 대해 단일 포트 메모리 테스트 알고리즘을 적용하고 있으며, 이러한 방법의 경우 테스트 수행 시간이 포트 수에 비례하는 문제점을 갖고 있다[2],[3]. [4],[5]에서는 포트 간의 간섭을 고려한 고장 모델을 제안하고, 이를 위한 알고리즘을 제안하고 있다. 그러나 이러한 고장은 각각 포트의 bit 라인과 word 라인 사이에 V_{dd}와 V_{ss} 트랙을 놓음으로써 해결될 수 있다[2].

본 논문에서는 각각의 포트에 대해 read/write를 할 수 있고, 포트간의 간섭이 발생하지 않도록 한 이중 포트 메모리를 위한 10N March 테스트 알고리즘 기반의 CM2 (Compact March test for 2-port memory) 알고리즘을 적용함으로써 기존의 방법들보다 더 효과적으로 이중 포트 메모리를 테스트할 수 있는 방법을 제안하였다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 메모리 테스트의 기반이 되는 메모리의 고장 모델에 대하여 설명한다. 3장에서는 기존의 10N March 테스트 알고리즘에 대하여 살펴보고, 4장에서는 본 논문에서 제안한 CM2 알고리즘에 대하여 살펴보겠다.

2. 메모리의 고장 모델

실제 메모리에서의 고장은 매우 다양한 상태로 나타

• 이 연구는 99년도 한국과학재단 특정기초 연구비 지원으로 수행되었으며 지원에 감사드립니다. (과제번호 96-0102-16-01-3)
† 비 회 원 : 숭실대학교 컴퓨터학부
wyang@watt.soongsil.ac.kr
†† 정 회 원 : 숭실대학교 컴퓨터학부 교수
hoon@computing.soongsil.ac.kr
논문접수 : 1999년 8월 9일
심사완료 : 2001년 3월 19일

나게 된다. 따라서 메모리의 정상적인 동작에 영향을 미칠 수 있는 고장의 모든 경우에 대해서 테스트를 수행한다는 것은 실질적으로 불가능하다. 그러나 메모리 테스트의 목적은 특수한 경우를 제외하고는 고장의 유형이나 위치를 파악하기보다는, 단순히 고장의 발생유무를 파악하는 것이다 [6,7]. 그러므로 일반적인 메모리 테스트에서는 먼저 메모리의 구조를 기능(functional) 모델로 단순화시킨다[8]. 이 모델에서는 메모리를 메모리 셀 배열(memory cell array), 주소 디코더(address decoder), 읽기/쓰기 회로(read/write logic)로 구성된 형태로 표현할 수 있다. 그림 1은 고장 검출만을 위해 사용되는 축소된 메모리의 기능적 모델을 보여준다.

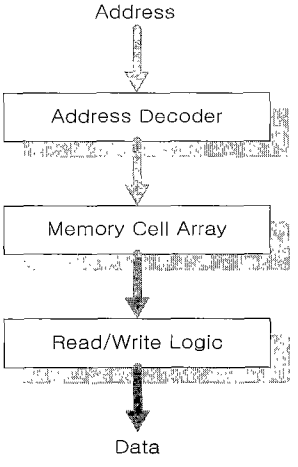

그림 1 축소된 메모리 기능 모델

이러한 축소된 기능적 모델에서 발생되는 기능 고장들은 주소 디코더 고장(address decoder fault), 고착 고장(stuck-at fault), 천이 고장(transition fault), 그리고 결합 고장(coupling fault)이 있다. 고착 고장 및 천이 고장은 하나의 셀만을 고려한 고장 모델이고, 결합 고장은 두 개의 셀을 고려한 고장이다.

• 주소 디코더 고장 모델

주소 디코더에서 발생할 수 있는 고장을 살펴보면 그림 2와 같이 분류될 수 있다. 그림 2에서 A의 경우는 주소 Ax로 메모리의 어떤 셀도 접근할 수 없는 경우이다. B는 주소 Ax가 두 개의 메모리 셀 Cx와 Cy를 접근하는 경우이고, C는 주소 Ax와 주소 Ay가 하나의 메모리 셀 Cx를 접근하는 경우이다[9,10,11,12].

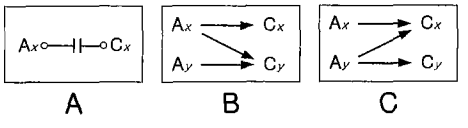

그림 2 주소 디코더 고장

주소 디코더에서 발생하는 고장 A는 메모리의 모든 셀에 x(0 또는 1)를 쓰고 x를 읽어보고, x′(1 또는 0)를 쓰고 x′를 읽어봄으로써 검출된다. 고장 B와 C는 다음과 같은 두 개의 조건을 만족하는 테스트 패턴을 인가함으로써 검출할 수 있다[1].

조건1: ⇑(rx, ... , wx′)

조건2: ⇓(rx′, ... , wx)

조건1에서 쓰인 기호 ⇑는 메모리의 주소를 낮은 주소에서 높은 주소로 변화시켜 가는 것을 의미하고, 조건2의 ⇓는 메모리의 주소를 높은 주소에서 낮은 주소로 변화시켜 가는 것을 의미한다.

• 고착 고장 모델

고착 고장은 메모리 셀의 논리값이 0이나 1로 고정되어 그 값이 변하지 않는 고장이다. 고착 고장에는 해당 셀이 0에 고정되는 stuck-at-0(SA0) 고장과 1에 고정되는 stuck-at-1(SA1) 고장이 있다.

SA1 고장은 해당 셀이 1에 고착되는 고장이기 때문에 0을 쓰고 0을 읽음으로써 고장을 검출할 수 있다. 마찬가지로 SA0 고장은 1을 쓰고 1을 읽어봄으로써 검출할 수 있다.

• 천이 고장 모델

천이 고장은 메모리 셀의 논리값이 0에서 1(상향 천이), 또는 1에서 0(하향 천이)으로의 천이가 되지 않는 고장이다. 상향 천이 고장(<↑/0>)은 셀의 초기값이 1인 경우는 0을 씀으로써 셀의 내용을 바꿀 수 있다. 그러나 일단 셀의 값이 0이 되면 0으로 고착되어 1이 쓰여지지 않는다. 하향 천이 고장(<↓/1>)은 셀의 초기값이 0인 경우 1을 씀으로써 셀의 내용을 바꿀 수 있다. 그러나 일단 셀의 값이 1이 되면 1로 고착되어 논리 0이 쓰여지지 않는다.

천이 고장을 검출하기 위한 테스트는 메모리의 각 셀이 상향 천이와 하향 천이하도록 만들고, 천이가 일어난 직후 셀의 값을 읽어봄으로써 검출이 가능하다. 즉, 상향 천이 고장은 초기값 0에서 해당 셀에 1을 쓰고 읽음으로써 고장을 검출할 수 있고, 하향 천이 고장은 초기값 1에서 해당 셀에 0을 쓰고 읽음으로써 고장을 검출할 수 있다.

• 결합 고장 모델

결합 고장은 특정 메모리 셀에서 논리값의 천이가 일

어날 때, 이 셀과 연관된 다른 메모리 셀의 값이 변하는 고장이다. 즉, 셀 i에서 논리값의 천이가 일어날 때, 셀 j의 논리값을 바꾸는 경우로서 셀 i는 결합 셀(coupling cell), 셀 j는 피결합 셀(coupled cell)이라 부른다.

결합 고장에는 한 셀의 천이가 다른 셀의 내용을 바꾸는 반전 결합 고장(Inversion Coupling Fault: CF_{in})과 한 셀의 천이가 다른 셀의 내용을 0이나 1로 고정시키는 동행 결합 고장(Idempotent Coupling Fault: CF_{id})이 있다. 결합 고장 <↑;1>은 결합 셀에서 상향 천이가 발생할 때, 피결합 셀의 값이 1로 고착되는 고장이고, <↑;0> 고장은 결합 셀에서 상향 천이가 발생할 때, 피결합 셀의 값이 0으로 고착되는 고장이다. <↓;0> 고장과 <↓;1> 고장은 결합 셀에서 하향 천이가 발생할 때, 피결합 셀의 값이 0과 1로 고착되는 고장이다. 이러한 결합 고장은 피결합 셀의 값을 확인한 후, 결합 셀에 적절한 천이를 발생시키고 피결합셀의 값을 확인함으로써 검출할 수 있다.

3. 10N March 테스트 알고리즘

주소 디코더 고장, 고착 고장, 천이 고장, 결합 고장을 검출할 수 있으며 10N(N은 전체 메모리 셀의 수)의 시간 복잡도를 갖는 10N March 테스트 알고리즘은 다음과 같다[13].

{⇓(w0);⇓(r0,w1);⇓(r1,w0);⇑(r0,w1);⇑(r1,w0);⇑(r0)}

M_0 M_1 M_2 M_3 M_4 M_5

알고리즘에서 ⇓, ⇑, w0, w1, r0, r1의 정의는 다음과 같다.

⇓ : 메모리 주소가 감소하는 방향
⇑ : 메모리 주소가 증가하는 방향
w0 : 메모리 셀에 0을 쓴다.
w1 : 메모리 셀에 1을 쓴다.
r0 : 메모리 셀에 0을 읽는다.
r1 : 메모리 셀에 1을 읽는다.

예를 들어 M_1 동작은 메모리 주소를 감소시키면서 현재 주소에 해당하는 셀에 0을 읽고 1을 쓰는 동작이다.

- 주소 디코더 고장 검출

주소 디코더 고장 모델의 A 경우는 M_2와 M_3에 의해 검출되고, B 경우는 M_3 단계에 의해 검출된다. 그리고 C 경우는 M_4 단계에 의해 검출된다.

- 고착 고장 검출

고착-1 고장은 M_1과 M_2 단계에 의해 검출되고, 고착-0 고장은 M_2와 M_3에 의해 검출된다.

- 천이 고장 검출

0에서 1로의 상향 천이 고장은 M_2와 M_3 단계에서, 1에서 0으로의 하향 천이 고장은 M_3와 M_4 단계에서 검출된다.

- 결합 고장 검출

동행 결합 고장을 검출하는 테스트 패턴은 반전 결합 고장도 검출할 수 있기 때문에 본 논문에서는 동행 결합 고장 검출에 대해서만 살펴보겠다[1]. 각각의 동행 결합 고장을 검출할 수 있는 경우는 다음과 같다.

그림 3은 <↑;1> 고장을 검출하는 과정을 보여준다. C_i 셀이 C_j 셀에 의해 결합 고장이 발생하는 경우는 그림3의 (a)에 해당된다. 그림에서 알 수 있듯이 C_i 에 1을 쓸 때, C_j 셀의 값이 1로 변하게 된다. 따라서 C_j 셀의 r0 동작에서 고장이 검출되게 된다. 반대로 C_i 셀이 C_j 셀에 결합되어 있는 경우는 그림3의 (b)에 해당된다. (b)에서 C_j 셀에 1을 쓰는 동작이 수행될 때, 결합 고장에 의해 C_i 셀의 값이 1로 변하게 된다. 그리고 C_i 셀의 r0 동작에 의해서 검출된다.

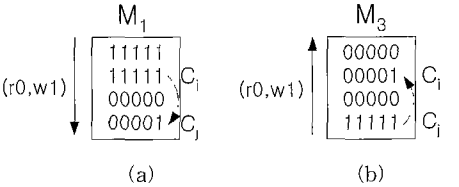

그림 3 <↑;1> 고장 검출

그림 4는 <↓;0> 고장을 검출하는 과정이다. 그림4의 (a)는 Cj 셀이 Ci 셀에 의해 결합 고장이 발생하는 경우이다. C_i셀에 0을 쓸 때, 결합 고장에 의해 C_j 셀의 값이 0으로 변하게되고 C_j셀의 고장은 r1 동작에 의해 검출된다.

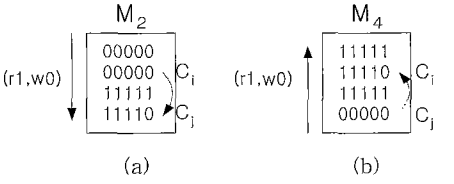

그림 4 <↓;0> 고장 검출

그림 5는 <↑;0> 고장을 검출하는 과정을 보여준다. 그림5의 (a)는 M_3 단계에서 C_i 셀에 1을 쓸 때, 결합 고장에 의해 C_j셀의 값이 0으로 바뀌게되고, M_4단계의 (r1,w0) 동작에서 고장이 검출된다. (b)는 M1 단계에서 C_j 셀에 1을 쓸 때, C_i 셀의 값이 0으로 바뀌게되고, M_2 단계의 (r1,w0) 동작에서 검출된다.

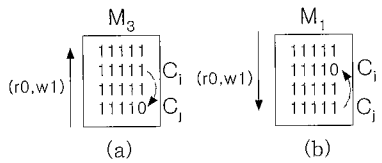

그림 5 <↑;0> 고장 검출

<↓;1> 고장을 검출하는 과정은 그림 6에 나와있다. 그림6의 (a)는 C_i 셀에 의해 C_j 셀에 고장이 발생하는 경우이다. C_i 셀에 0을 쓸 때, 결합 고장에 의해 C_j 셀의 값이 1로 바뀌게 되고, M_5 단계의 (r0) 동작에서 고장이 검출된다.

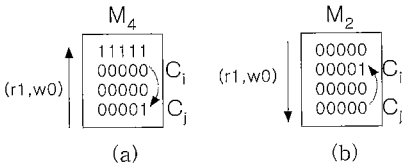

그림 6 <↓;1> 고장 검출

4. 이중 포트 메모리 테스트를 위한 알고리즘

본 논문에서는 각각의 포트에 대하여 독립적으로 쓰기와 읽기를 할 수 있는 이중 포트 메모리의 특성을 이용하여 각각의 포트에서 동시에 테스트를 수행함으로써 기존의 방법에 비해 더 짧은 시간에 주소 디코더 고장, 고착 고장, 천이 고장, 결합 고장을 완벽하게 검출할 수 있는 CM2 알고리즘을 제안하였다. 제안된 CM2 알고리즘의 절차는 다음과 같다.

```
for( i = 0 ; i < 9 ; i++)
   for( j = 0 ; j < (n-2)/2 ; j++)
      case i = 0      :  write(j, port1);
                         write((n-1)-j,port2);
      case i = 1 or 2 :  read(j, port1);
                         write(i, port1);
                         read((n-1)-j, port2);
                         write((n-1)-j,port2);
      case i = 3 or 4 :  read((n-2)/2-j, port1);
                         write((n-2)/2-j, port1);
                         read(((n-2)/2+1)+j,port2);
                         write(((n-2)/2+1)+j,port2);
      case i = 5 or 6 :  read(j, port1);
                         write(j, port1);
                         read(((n-2)/2+1)+j,port2);
                         write(((n-2)/2+1)+j,port2);
      case i = 7 or 8 :  read((n-2)/2-j, port1);
                         write((n-2)/2-j, port1);
                         read((n-1)-j, port2);
                         write((n-1)-j,port2);
```

이중 포트 메모리에 대한 테스트 CM2 알고리즘의 적용은 그림7과 같다. 그림 7에서 음영이 들어간 부분은 포트 A에 의해 테스트가 수행되고, 음영이 들어가지 않은 부분은 포트 B에 의해 테스트가 수행되는 부분이다. 제안된 알고리즘의 시간 복잡도는 8.5N이다(M_0는 0.5N, M_1, ... , M_8은 각각 1N).

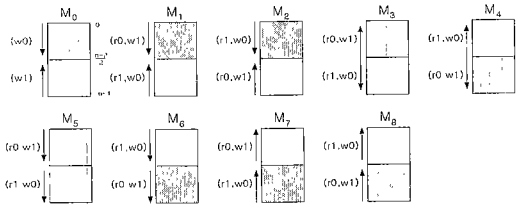

그림 7 제안된 이중 포트 메모리 테스트 알고리즘의 적용

그림7과 같이 10N March 테스트 알고리즘을 적용하면 $0 \sim \frac{(n-2)}{2}$ 사이의 메모리와 $\frac{(n-2)}{2}+1 \sim (n-1)$ 사이의 메모리에 대해 각각 10N March 테스트 알고리즘이 적용되었기 때문에 단일 셀에 대한 고장 모델인 고착 고장 및 천이 고장은 모두 검출된다.

- 주소 디코더 고장 검출

포트 A와 포트 B에 대해서 각각 모든 주소 디코더 고장을 검출하기 위해서는 ⇑(rx,...,wx´)와 ⇓(rx´, ... ,wx)의 테스트 패턴이 인가되어야 한다. 그림7에서 알 수 있듯이 A 포트는 M_2와 M_3, B 포트는 M_6와 M_7 단계에 의해 테스트된다.

- 결합 고장 검출

본 논문에서 제안하는 방법은 메모리를 $0 \sim \frac{(n-2)}{2}$ 메모리 블록과 $\frac{(n-2)}{2}+1 \sim (n-1)$ 메모리 블록으로 나누어 각각의 블록에 대해 10N March 테스트가 적용되기 때문에 각 블록 내에서의 결합 고장은 모두 검출된다. 또한 블록간의 결합 고장은 coupling 셀과 coupled 셀에 읽기 또는 쓰기 동작이 수행되는 순서에 따라 크게 세 가지의 경우로 구분할 수 있으며, coupling 셀과 coupled 셀의 주소의 크기에 따라 2가지의 경우로 분류될 수 있다. 즉, 모두 6가지의 결합 고장이 존재할 수 있는 경우가 있으며 그림 8은 여섯 가지 경우를 보여준다. 예를 들면, 1의 경우는 coupled 셀의 주소가 coupling 셀의 주소보다 크고, coupling셀 보다 coupled 셀에 대하여 먼저 읽기 또는 쓰기 동작이 수행된다.

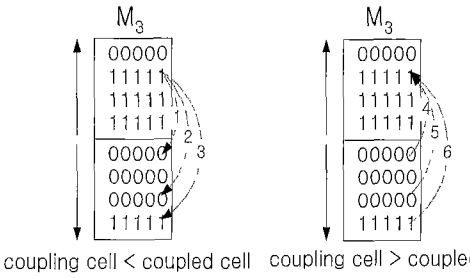

그림 8 제안된 방법에서의 결합 고장 분류

1의 경우와 4의 경우에 대한 결합 고장의 검출 과정이 그림9부터 그림12까지 나와있다. 그림 9부터 그림 12까지에서 (a)는 1의 경우이고 (b)는 4의 경우이다. 그림 9는 <↑;1> 고장을 검출하는 과정을 보여준다. (a)를 살펴보면 $\frac{(n-2)}{2}$ 에서부터 0번지 방향으로 (r0,w1) 동작이 수행되고, $\frac{(n-2)}{2}+1$번지에서부터 $(n-1)$번지 방향으로 (r1,w0) 동작이 수행된다. 따라서 셀 C_i에 1을 쓸 때, 결합 고장으로 인해 C_j 셀의 값이 1로 바뀐다. 이 고장의 결과는 M_4의 r0 동작에서 검출된다. (b)의 경우는 $\frac{(n-2)}{2}$ 에서부터 0번지 방향으로 (r1,w0) 동작이 수행되고, $\frac{(n-2)}{2}+1$번지에서부터 $(n-1)$번지 방향으로 (r0,w1) 동작이 수행된다. 따라서 C_j 셀에 1을 쓸 때, 결합 고장에 의해 C_i 셀에 1값이 쓰여지게 되고 r1 동작에 의해 검출된다.

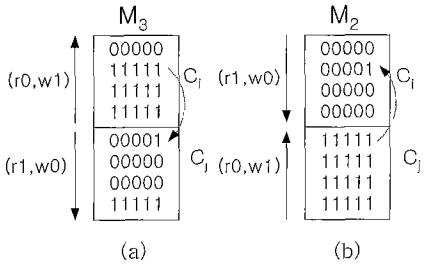

그림 9 <↑;1> 고장 검출

그림 10은 <↑;0> 고장을 검출하는 과정을 보여준다. (a)를 살펴보면 0번지에서부터 $\frac{(n-2)}{2}$ 번지 방향으로 (r0,w1) 동작이 수행되고, $(n-1)$번지에서 $\frac{(n-2)}{2}+1$ 번지 방향으로 (r1,w0) 동작이 수행된다. C_i 셀에 1을 쓸 때, 결합 고장에 의해 C_j 셀의 값이 0으로 변하게 되고 r1 동작에 의해 고장이 검출된다. (b)를 살펴보면 0번지에서부터 $\frac{(n-2)}{2}$ 번지 방향으로 (r1,w0) 동작이 수행되고, $(n-1)$번지에서 $\frac{(n-2)}{2}+1$번지 방향으로 (r0,w1) 동작이 수행된다. C_j 셀에 1을 쓸 때, 결합 고장에 의해 C_i 셀의 값이 0으로 변하게된다. 그리고 이 고장은 r1 동작에 의해 검출된다.

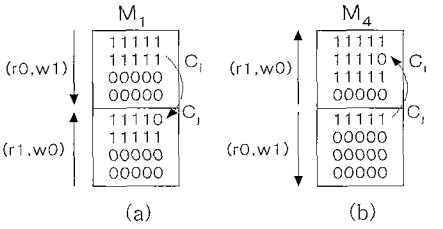

그림 10 <↑;0> 고장 검출

그림 11은 <↓;0> 고장을 검출하는 과정을 보여준다. (a)는 $\frac{(n-2)}{2}$ 번지에서부터 0번지 방향으로 (r1,w0) 동작이 수행되고, $\frac{(n-2)}{2}+1$ 번지에서 $(n-1)$ 번지 방향으로 (r0,w1) 동작이 수행된다. C_i 셀에 0이 쓰여질 때, 결합 고장에 의해 C_j 셀의 값이 0으로 변하게되고, 이 고장은 다음 단계인 M_5의 r1 동작에 의해 검출된다. (b)는 0번지에서부터 $\frac{(n-2)}{2}$ 번지 방향으로 (r0,w1) 동작이 수행되고, $(n-1)$번지에서부터 $\frac{(n-2)}{2}+1$번지 방향으로 (r1,w0) 동작이 수행된다. Cj셀에 0을 쓸 때, 결합 고장에 의해 Ci 셀에 0이 쓰여지게 되고, 이 고장은 다음 단계인 M2의 r1 동작에 의해 검출된다.

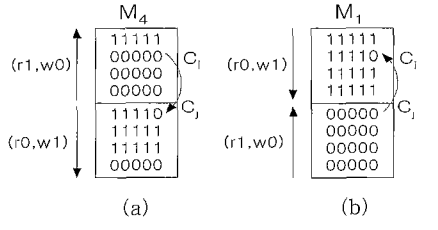

그림 11 <↓;0> 고장 검출

그림 12는 <↓;1> 고장을 검출하는 과정을 보여준다. (a)는 0번지에서부터 $\frac{(n-2)}{2}$ 번지 방향으로 (r1,w0) 동작이 수행되고, $(n-1)$번지에서 $\frac{(n-2)}{2}+1$번지 방향으로 (r0,w1) 동작이 수행된다. C_i 셀에 0이 쓰여질 때,

결합 고장에 의해 C_j셀의 값이 1로 바뀌게 되고 r0 동작에 의해 검출된다. (b)를 살펴보면 0번지에서부터 $\frac{(n-2)}{2}$번지 방향으로 (r0,w1) 동작이 수행되고, $(n-1)$번지에서 $\frac{(n-2)}{2}+1$번지 방향으로 (r1,w0) 동작이 수행된다. C_j셀에 0이 쓰여질 때, C_i 셀의 값이 결합 고장에 의해 1로 바뀌게 된다. 이 고장은 r0 동작에 의해 검출된다.

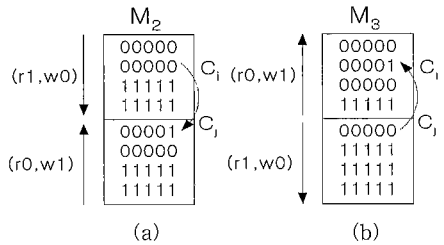

그림 12 <↓;1> 고장 검출

그림 8의 나머지의 경우에 대해서도 그림9부터 12까지의 경우와 같이 증명되고, 이를 정리하면 표1과 같다.

표 1 결합 고장 검출

		<↑;1>	<↑;0>	<↓;1>	<↓;0>
$C_i > C_j$	경우 1	$M_3 \sim M_4$	M_1	M_2	M_5
	경우 2	$M_1 \sim M_2$	M_5	M_6	$M_4 \sim M_5$
	경우 3	$M_1 \sim M_2$	M_3	M_4	$M_2 \sim M_3$
$C_i < C_j$	경우 4	$M_2 \sim M_3$	M_4	M_3	$M_1 \sim M_2$
	경우 5	$M_2 \sim M_3$	M_8	M_7	$M_1 \sim M_2$
	경우 6	$M_4 \sim M_5$	M_2	M_1	$M_3 \sim M_4$

5. 결론 및 향후 과제

기존의 다중 포트 메모리의 테스트는 단일 포트 메모리 알고리즘을 각각의 포트에 적용하여 테스트를 수행함으로써 테스트 수행 시간이 메모리 포트 수에 비례하여 증가하는 문제점이 있었다. 본 논문에서는 각각의 포트를 통해 독립적으로 읽기와 쓰기를 할 수 있는 이중 포트 메모리의 특성을 이용하여 기존의 알고리즘들보다 훨씬 빠르게 이중 포트 메모리를 테스트할 수 있는 알고리즘을 제안하였다. 제안된 알고리즘은 8.5N의 시간 복잡도를 가지고 주소 디코더 고장, 고착 고장, 천이 고장, 결합 고장을 모두 검출할 수 있다.

향후 연구에서는 다중 포트 메모리를 테스트할 수 있도록 제안된 알고리즘을 개선할 것이며, 제안된 알고리즘을 회로로 구현하여 회로의 성능 분석을 수행할 예정이다.

참 고 문 헌

[1] *Memory BistCore™ User's Reference Manual*, GeneSys TestWare, Revision 1.4, June, 1998.

[2] Yeujian Wu and Sanjay Gupta, "Built-In Self-Test for Multi-Port RAMs," *Asian Test Symposium*, 1997.

[3] 한재천, 양선웅, 진명구, 장훈, "내장된 이중포트 메모리의 효율적인 테스트 방법에 관한 연구", 전자공학회 논문지, 1999

[4] A. J. Goor, "Port Interference Faults in Two-Port Memories," *International Test Conference*, 1999

[5] A. Benso, S. D. Carlo and P. Prinetto, "A Programmable BIST Architecture for Cluster of Multiple-Port SRAMs," *International Test Conference*, 2000

[6] A. J. Goor, *Testing Semiconductor Memories*, John Wiley & Sons Ltd., 1991.

[7] R. P. Treuer and V. K. Agarwal, "Fault Location Algorithms for Repairable Embedded RAMs," *International Test Conference*, 1993.

[8] Pinamki Mazumder and Kanad Chakraborty, *Testing and Testable Design of High-Density Random Access Memories*, Kluwer Academic Publishers, 1996.

[9] Tom Chen and Glen Sunada, "A Self-Testing and Self-Repairing Structure for Ultra-Large Capacity Memories," *International Test Conference*, 1992.

[10] J. V. Sas, G. V. Wause, E. Huyskens and D. Rabaey, "BIST for Embeded Static RAMs with Coverage Calculation," *International Test Conference*, 1993.

[11] V. G. Mikitjuk, V. N. Yarmolik and A. J. van de Goor, "RAM Testing Algorithms for Detection Multiple Linked Faults," *International Test Conference*, 1996.

[12] M. Sachdev, "Test and Testability Techniques for Open Defects in RAM Address Decoder," *International Test Conference*, 1996.

[13] I. Schanstra and A. J. van de Goor, "Industrial Evaluation of Stress Combinations for March Tests applied to SRAMs," *International Test Conference*, 1999.

양 선 웅
1996년 숭실대학교 전자계산학과 졸업(B.S). 1998년 숭실대학교 대학원 전자계산학과 졸업(M.S). 1998년 ~ 현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 컴퓨터구조, VLSI 설계 및 테스팅, CAD

장 훈
1987년 서울대학교 전자공학과 졸업(B.S). 1989년 서울대학교 전자공학과 졸업(M.S). 1993년 University of Texas at Austin 박사학위 취득. 1991년 IBM Inc. 1993년 Motorola Inc. Senior Member of Technical Staff. 1994년 ~ 현재 숭실대학교 컴퓨터학부 조교수. 관심분야는 컴퓨터 시스템. VLSI 설계, VLSI 테스팅