論文2003-40SD-1-9

이중 포트 메모리를 위한 효과적인 테스트 알고리듬

(An Efficient Test Algorithm for Dual Port Memory)

金 智 惠 *, 宋 東 燮 *, 裵 相 民 **, 姜 成 昊 *

(Ji-Hye Kim, Dong-Sup Song, Sang-Min Bae, and Sung-Ho Kang)

요 약

회로의 설계기술, 공정기술의 발달로 회로의 복잡도가 증가하고 있으며 대용량 메모리의 수요도 급격하게 증가하고 있다. 이렇듯 메모리의 용량이 커질수록 테스트는 더더욱 어려워지고, 테스트에 소요되는 비용도 점차 증가하여 테스트가 칩 전체에서 차지하는 비중이 커지고 있다. 따라서 짧은 시간에 수율을 향상시킬 수 있는 효율적인 테스트 알고리듬에 대한 연구가 중요하게 여겨지고 있다. 본 논문에서는 단일 포트 메모리의 고장을 검출하는데 가장 보편적으로 사용되는 March C 알고리듬을 바탕으로 하여 이를 보완하고, 추가되는 테스트 길이 없이 단일 포트 메모리뿐만 아니라 이중 포트 메모리에서 발생할 수 있는 모든 종류의 고장이 고려되어 이중 포트 메모리에서도 적용 가능한 효과적인 테스트 알고리듬을 제안한다.

Abstract

Due to the improvements in circuit design technique and manufacturing technique, complexity of a circuit is growing along with the demand for memories with large capacities. Likewise, as a memory capacity gets larger, testing gets harder and testing cost increases, and testing process in chip development gets larger as well. Therefore, a research on an effective test algorithm to improve the chip yield rate in a short time period is becoming an important task. This paper proposes an effective, March C algorithm based, test algorithm that can also be applied to a dual-port memory since it considers all the fault types, which can be occurred in a single-port as well as in a dual-port memory, without increasing the test length.

Keywords : 이중포트메모리, 테스트, 고장모델

I. Introduction

메모리 설계 기술은 엄청난 속도로 발전해 왔으며, 특히 미세 가공기술의 발전에 따라 그 집적도가 급격하게 향상되었다. 그러나 이러한 메모리의 고집적화는 복잡하고 정밀한 제조 공정을 필요로 하고 이에 대한 테스트를 보다 어렵게 하고 있다. 또한 메모리는 고집적화 될 뿐 아니라 동작속도 또한 매우 빠른 추세로 증가하고 있기 때문에 높은 신뢰도를 유지하면서 빠른 시간 내에 테스트를 수행하는 것이 가장 중요하다. 이중 포트 메모리의 경우 두 개의 포트를 통해 메모리에 동시에 접근할 수 있으므로, 많은 양의 데이터를 빠른 시간에 처리할 수 있다. 따라서 영상처리나 네트워크 프로세서 등 여러 분야에서 그 사용이 증가하고 있으므로 이를 위한 테스트 알고리듬에 대한 연구가 매우 중요하다. 메모리를 테스트하기 위해 가장 먼저 고려되어야 할 점은 고장 모델을 설정하는 것이다. 고장 모델을 어떻게 정하느냐에 따라 테스트 알고리듬의 성능과

* 正會員, 延世大學校 電氣電子工學科
(Dept. of Electrical Eng., Yonsei Univ.)
** 正會員, 三星電子 CAE 센터
(Samsung Electronics, CAE Center)

※본 연구는 산업자원부와 과학기술부의 시스템 IC 2010 사업의 지원으로 수행하였습니다.
接受日字: 2002年5月14日, 수정완료일: 2003年1月7日

테스트 시간 등이 모두 달라질 수 있기 때문이다. 단일 포트 메모리에 대한 고장 모델은 이미 많은 연구를 통해 그 명칭과 종류 등이 정립되어 있는 상황이다. 또한 이에 대한 테스트 알고리듬 역시 그 성능과 길이에 따라 다양하게 연구, 발표되어 왔다. 그러나 이중 포트 메모리에 대한 고장 모델은 단일 포트 메모리의 고장 모델에 비해 확립되지 않은 상황이며, 이중 포트 메모리의 테스트 알고리듬에 대한 연구 또한 부족한 실정이다. 따라서 이중 포트 메모리에서 발생할 수 있는 다양한 고장 모델들이 고려된 효과적인 테스트 알고리듬에 대한 연구가 중요하게 여겨지고 있다. 본 논문에서는 이중 포트 메모리에서 발생할 수 있는 모든 결함에 대하여 IFA(Inductive Fault Analysis)를 통하여 다양한 고장 모델을 제시한 [1]을 바탕으로 하여 단일 포트 관련 고장 모델과 이중 포트 관련 고장 모델을 고려한 이중 포트 메모리를 위한 테스트 알고리듬을 제안한다. 제안하는 새로운 알고리듬은 단일 포트 메모리를 위한 대표적인 테스트 알고리듬인 March C-를 기반으로 하였으며, 이중 포트 메모리에서 발생 가능한 고장모델들을 추가하여 March C 를 보완하고, 변형하여 이중 포트 메모리에 적합하도록 고안된 새로운 알고리듬이다. 본 알고리듬은 단일 포트 메모리를 위한 테스트 알고리듬과 비교하여 추가로 증가되는 테스트 알고리듬의 길이 없이, 서로 다른 포트를 통해 적절한 동작을 동시에 수행해 줌으로써 단일 포트 메모리에서 발생할 수 있는 고장뿐만 아니라 이중 포트 메모리에서 발생할 수 있는 고장까지 검출할 수 있는 매우 효과적인 알고리듬이다. 본 논문은 다음과 같이 구성된다. 2장에서는 이중 포트 메모리의 테스트 알고리듬에 대한 기존의 연구에 대해 소개하고, 3장에서는 검출 대상 고장 모델들을 제시한다. 그리고 이러한 고장 모델들을 검출할 수 있는 테스트 알고리듬을 단계적으로 제안한다. 4장에서는 기존의 연구와 새롭게 제안된 알고리듬의 성능을 비교 평가하고, 마지막으로 5장에서 본 논문의 내용을 요약, 정리한다.

Ⅱ. 기존 연구

[2]에서는 단일 포트 관련 고장과 다중 포트 메모리에서 발생할 수 있는 서로 다른 포트의 비트라인이나 워드라인 사이의 잘못된 연결로 인한 고장을 검출하는 알고리듬 P를 제안하였다. 이 알고리듬은 하나의 포트를 통해 March C- 알고리듬을 가해주면서 동시에 다른 포트를 통해 같은 열, 다른 행의 주소에 읽기 동작을 수행하면서 고장을 검출하게 된다. 그러나 DRDF 관련 고장을 검출할 수 없으며 2PF2av의 일부를 제외한 나머지 이중포트 관련 고장도 검출할 수 없을 뿐만 아니라 다른 알고리듬에 비해 길이가 길어지는 단점이 있다. [3]은 한 셀의 인접한 4개의 이웃 셀을 March guard로 정의하여 하나의 포트를 통해서 March 알고리듬을 가해주고, 동시에 나머지 포트를 통해 March guard에 읽기 동작을 수행하도록 하여 고장을 테스트하게 된다. 테스트 알고리듬의 길이는 단일 포트 메모리를 테스트 할 때와 비교해 늘어나는 부분이 없지만 DRDF, CFdr 그리고 [2]의 경우와 마찬가지로 2PF2av를 제외한 나머지 이중포트 관련 고장을 검출하지 못하는 단점을 가지고 있다. [4]의 경우 시뮬레이션을 통해 다중 포트 메모리의 고장 모델을 정의하고 이에 대한 테스트 알고리듬을 제안하였지만, 이 역시 다중 포트 메모리에서 일어날 수 있는 특수한 고장만을 검출하기 위한 것이며, 각각의 고장모델 하나하나에 대한 테스트 알고리듬을 제시하였기 때문에 전체적으로 고장을 검출하기에 효과적이지 못하며 다양한 고장 모델이 고려되지 못한 단점을 가지고 있다. [5]의 경우, [1]의 고장 모델들을 바탕으로 하여 다양한 고장 모델이 고려되었지만, 이 역시 고장 모델에 대해 각각의 테스트 알고리듬을 제시하였기 때문에 실질적으로 이를 통합하여 효과적으로 가해줄 수 있는 새로운 테스트 알고리듬이 필요하다.

Ⅲ. 이중 포트 메모리를 위한 테스트 알고리듬

고장 모델에 따라서 가해지게 되는 테스트 패턴이 달라지게 된다. 때문에 효과적인 테스트 알고리듬을 만들기 위해서는 적절한 고장 모델을 결정하는 것이 중요하다. 단일 포트 메모리를 위한 고장 모델은 많은 연구를 통해 어느 정도 정립이 되어있는 상태지만 이중 포트 메모리에서 발생하게 되는 고장 모델은 그 종류나 명칭이 확정적이지 못하기 때문에 먼저 이를 결정하는 일이 선행되어야 한다. 본 논문에서 제안하는 테스트 알고리듬은 [1]의 고장 모델을 바탕으로 하였다. [1]에서는 메모리 셀 안에서 발생할 수 있는 개방(open), 단락(short), 연결(bridge) 등의 모든 결함(spot

defect)들을 바탕으로 시뮬레이션을 통해 fault primitive를 정하고 다시 이들을 분류하여 고장 모델을 결정하였다. 고장 모델은 크게 단일 포트 관련 고장과 이중 포트 관련 고장으로 나누어지며, 이는 다시 하나의 셀에 관련된 고장과 두 개의 셀에 관련된 고장으로 나누어진다.

(1) 단일 포트 관련 고장

단일 포트 관련 고장(1PFs)은 크게 하나의 셀과 관련된 고장(1PF1)과 두 개의 셀이 관련된 고장(1PF2)으로 나눌 수 있다. 하나의 셀과 관련된 고장은 셀 내부의 노드들간의 잘못된 연결이나 끊어짐 등으로 인해 생기는 고장으로 그 종류는 <표 1>과 같다. 이러한 고장들 중 읽기 동작에 대한 결과 값이 랜덤하게 나오는 RRF와 정의되지 않은 상태의 고장인 USE의 경우는 결과를 예측할 수 없는 형태의 고장으로 어떠한 알고리듬을 가해주든지 고장 검출을 보장할 수 없으며 데이터 보존 고장인 DRF는 알고리듬에 시간적 요소까지 포함시켜야 한다. 따라서 이와 같은 고장들은 검출 대상 고장모델에서 제외하고자 한다. 두 개의 셀이 관련된 고장은 두 개의 셀 내부의 노드들 간의 잘못된 연결이나 간섭에 의해 발생하게 되는 고장으로 고장을 유발시키는 결합셀과 고장이 유발되는 피결합셀이 존재하게 되며 그 종류는 <표 2>와 같다. 이러한 고장들 중 읽기 동작의 결과가 랜덤하게 나오는 결합고장인 CFrr의 경우도 그 결과를 예측할 수 없는 형태의 고장이므로 검출 대상 고장모델에서 제외한다.

단일 포트 메모리를 위한 가장 보편적인 테스트 알고리듬인 March C- 알고리듬을 사용하게 되면, 정해진 단일 포트 관련 고장 모델들 중 DRDF와 DRDF의 특성을 포함하는 고장인 CFdr을 제외한 모든 고장을 검출할 수 있다. DRDF는 읽기 동작을 수행하는 과정에서 원래 셀이 가지고 있던 값이 천이 되는 고장이 발생했음에도 불구하고, 그 읽기 동작에 대한 결과가 천이 되지 않은 것처럼 나타나게 되는 고장으로 첫 번째 읽기 동작을 통해서는 고장이 드러나지 않게 된다. 이 고장을 검출하기 위해서는 읽기 동작 수행 후, 다시 한 번 읽기 동작을 반복해 주면 고장을 검출할 수 있다. 따라서 <그림 1>과 같이 March C-의 각각의 읽기 동작에 대해 추가적으로 한번의 읽기 동작을 수행하게 되는 변형된 알고리듬을 가해주면 단일포트와 관련된 모든 고장을 검출할 수 있다.

$\Updownarrow w0; \Uparrow(r0, r0, w1); \Uparrow(r1, r1, w0); \Downarrow$

$(r0, r0, w1); \Downarrow(r1, r1, w0); \Updownarrow r0;$

그림 1. DRDF를 검출하기 위한 March C- 변형 알고리듬

Fig. 1. Modified March C- algorithm for detecting DRDF.

표 1. 단일 포트 메모리를 위한 고장 모델과 검출 대상 고장 모델(1PF1)

Table 1. Fault models and target fault models for single fault memories(1PF1).

분류	고장 모델	Fault Primitives	검출 대상
1PF1	SAF	<∀/0/->,<∀/1/->	○
	TF	<w↑/0/->,<w↓/1/->	○
	RDF	<r0/↑/1>,<r1/↓/0>	○
	DRDF	<r0/↑/1>,<r1/↓/0>	○
	IRF	<r1/↓/0>,<r0/0/1>	○
	RRF	<r0/0/?>,<r1/1/?>	×
	DRF	<1T/↓/->,<0T/↑/->,<xT/?/->	×
	NAF	<w↑/0/->,<w↓/1/->,<rx/x/?>	○
	USF	<wx/?/->,<rx/?/?>	×

표 2. 단일 포트 메모리를 위한 고장 모델과 검출 대상 고장 모델(1PF2)

Table 2. Fault models and target fault models for single fault memories (1PF2).

분류	고장 모델	Fault Primitives	검출 대상
1PF2	CFds	<wx;1/↓/->,<rx;0/↑/->,<rx;1/↓/->,<wx;0/↑/->	○
	CFst	<1;1/0/->,<1;0/1/->,<0;1/0/->,<0;0/1/->	○
	CFir	<0;r0/0/1>,<0;r1/1/0>,<1;r0/0/1>,<1;r1/1/0>	○
	CFrr	<0;r0/0/?>,<0;r1/1/?>,<1;r0/0/?>,<1;r1/1/?>	×
	CFdr	<0;r0/↑/0>,<0;r1/↓/1>,<1;r0/↑/0>,<1;r1/↓/1>	○
	CFrd	<0;r0/↑/1>,<0;r1/↓/0>,<1;r0/↑/1>,<1;r1/↓/0>	○
	CFtr	<0;w↓/1/->,<0;w↑/0/->,<1;w↓/1/->,<1;w↑/0/->	○

(2) 이중 포트 관련 고장

이중 포트 관련 고장은 이중포트 메모리에서 두 개의 포트를 통해 두 개의 셀 또는 하나의 셀에 동시에 접근하게 될 때 발생할 수 있는 고장으로 두 가지의 약고장(weak fault)이 동시에 발생하여 하나의 고장으로 나타나게 되는 경우를 말한다. 이중 포트 관련 고장도 하나의 셀과 관련된 고장과 두 개의 셀에 관련된 고장으로 나누어진다. 그리고 <표 3>과 같이 다시 두 개의 셀이 관련된 고장은 피결합셀과 결합셀의 동작 영향에 따라 세 가지로 분류된다.

이중 포트 메모리에서는 다음과 같은 동작들만이 수행 가능하다는 전제 하에 검출 대상 고장 모델을 결정한다.

- 두 포트를 통해 동시에 같은 셀을 읽는 동작
- 두 포트를 통해 동시에 서로 다른 셀을 읽는 동작
- 두 포트를 통해 동시에 서로 다른 셀에 쓰는 동작
- 동시에 한 포트를 통해 하나의 셀에는 쓰고, 또 다른 포트를 통해 다른 셀을 읽는 동작

먼저 하나의 셀에 관련된 고장(2PF1)은 세 종류의 고장 모델이 존재한다. 2PF1은 두 포트를 통해서 하나의 셀에 동시에 접근할 때 접근한 셀에 오동작이 일어나는 경우로, 이 중에 wRDF&wTF는 하나의 셀에 동시에 읽고 쓰는 동작이 수행될 때 발생하는 고장이다. 대부분의 메모리에서 이렇게 한 셀에 동시에 읽고 쓰는 동작이 불가능하므로, 검출 대상 고장 모델에서 제외한다.

두 개의 셀이 관련된 고장은 2PF2v, 2PF2a 그리고 2PF2av로 나누어진다. 2PF2v는 두 포트를 통하여 동시에 결합셀에 접근하게 될 때 피결합셀에 고장이 유발되는 경우를 말한다. 피 결합셀이 특정 데이터 상태에 있을 때 두 포트를 통하여 동시에 결합셀의 데이터를 읽어들이면 피결합셀의 데이터가 천이 되는 형태의 고장이 발생하게 된다. 하나의 셀에 동시에 읽고, 쓰는 동작은 불가능하기 때문에 하나의 셀을 동시에 읽을 때 발생하게되는 고장인 <rx:rx;0/↑/->, <rx:rx;1/↓/->만을 고려한다. 2PF2a는 결합셀이 특정 데이터 상태에 있고, 두 포트를 통해 동시에 피결합셀에 접근하게 될 때 발생하게 되는 고장이다.

이러한 2PF1, 2PF2v, 2PF2a 등의 고장의 경우 하나의 포트를 통해 March 알고리듬을 가해주고, 읽기 동작을 수행할 때마다 나머지 포트를 통해서 동시에 같은 셀을 읽는 동작을 수행하면 검출할 수 있게 된다. <그림 2>는 2PF2av를 제외한 나머지 고장들을 검출할 수 있는 이중 포트 메모리를 위한 테스트 알고리듬이다.

A포트 ⇕w0;⇑(r0,r0,w1);⇑(r1,r1,w0);⇓(r0,r0,w1);⇓(r1,r1,w0);⇕r0;
B포트 ⇕ −;⇑(r0,r0,−);⇑(r1,r1,−);⇓(r0,r0,−);⇓(r1,r1,−);⇕r0;

그림 2. 이중포트 메모리를 위한 테스트 알고리듬 (2PF2av검출 제외)

Fig. 2. A test algorithm for dual port memories (except detecting 2PF2av).

마지막으로 2PF2av는 각각의 포트를 통해 서로 다른 셀에 동시에 접근하여 결합셀에는 쓰기 동작을, 피결합

표 3. 이중 포트 메모리를 위한 고장 모델과 검출 대상 고장 모델(2PFs)

Table 3. Fault models and target fault models for dual port memories (2PFs).

분류	고장 모델	Fault Primitives	검출대상
2PF1	wDRDF&wDRDF	<r0;r0/↑/0>, <r1:r1/↓/1>	○
	wRDF&wRDF	<r1:r1/↓/0>, <r0:r0/↑/1>	○
	wRDF&wTF	<r0:w↑/0/->, <r1:w↓/1/->	×
2PF2v	wCFds&wCFds	<w0:rd;0/↑/->, <w0:rd;1/↓/->, <w1:rd;0/↑/->, <w1:rd;1/↓/->	×
		<rx:rx;0/↑/->, <rx:rx;1/↓/->	○
2PF2a	wCFdr&wDRDF	<0;r0:r0/↑/0>, <1;r0:r0/↑/0>, <0;r1:r1/↓/1>, <1;r1:r1/↓/1>	○
	wCFrd&wRDF	<0;r1:r1/↓/0>, <1;r1:r1/↓/0>, <1;r0:r0/↑/1>, <0;r0:r0/↑/1>	○
2PF2av	wCFds&wRDF	<w0:r0/↑/1>, <w0:r1/↓/0>, <w1:r0/↑/1>, <w1:r1/↓/0>	○
	wCFds&wIRF	<w0:r0/0/1>, <w0:r1/1/0>, <w1:r0/0/1>, <w1:r1/1/0>	○
	wCFds&wRRF	<w0:r0/0/?>, <w0:r1/1/?>, <w1:r0/0/?>, <w1:r1/1/?>	×

셀에는 읽기 동작을 수행해 줄 때 피결합셀의 값이 바뀌게 되는 고장을 말한다. 이 고장은 같은 열의 또는 이웃한 열의 서로 다른 포트의 비트라인 사이에 잘못된 연결이 있을 경우 발생하는 고장으로 <그림 4>의 알고리듬을 가해주면 검출할 수 있다[5].

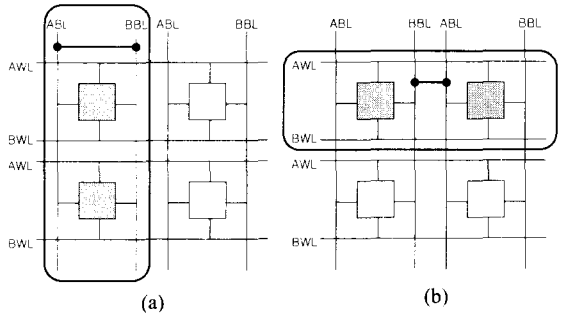


그림 3. 2PF2av가 나타나게 되는 예
(a) 같은 열의 비트라인 연결 (b) 이웃한 열의 비트라인 연결

Fig. 3. Examples of 2PF2av
(a) Bridge of bit lines which are on the same column
(b) Bridge of adjacent bit lines

$$\Uparrow_{i=0}^{R-1}(w_{i,j}0:NOP\ ,w_{i,j+1}0:NOP\ ,w_{i+1,j}0:NOP\ );$$
$$\Uparrow_{i=0}^{R-1}(w_{i,j}1\ :r_{i+1,j}0\ ,w_{i,j}0\ :r_{i+1,j}0\ );$$
$$\Uparrow_{i=0}^{R-1}(w_{i,j}1\ :r_{i,j+1}0\ ,w_{i,j}0\ :r_{i,j+1}0\ );$$
$$\Uparrow_{i=0}^{R-1}(w_{i,j}0:NOP\ ,w_{i,j+1}0:NOP\ ,w_{i+1,j}0:NOP\ );$$
$$\Uparrow_{i=0}^{R-1}(w_{i,j}0\ :r_{i+1,j}1\ ,w_{i,j}1\ :r_{i+1,j}1\ );$$
$$\Uparrow_{i=0}^{R-1}(w_{i,j}0\ :r_{i,j+1}1\ ,w_{i,j}1\ :r_{i,j+1}1\ );$$

그림 4. 2PF2av를 검출하기 위한 테스트 알고리듬
Fig. 4. A test algorithm for detecting 2PF2av.

<그림 3(a)>와 같이 같은 열의 비트라인이 연결되어 고장이 생겼을 경우 같은 열의 서로 다른 행 위에 있는 두 셀을 테스트 해 주면 고장을 검출할 수 있으며, <그림 3(b)>와 같이 이웃한 열의 비트라인간 연결로 고장이 발생했을 경우에는 같은 행의 이웃한 열 위의 두 셀에 대해 테스트 알고리듬을 가해주면 고장을 검출할 수 있다. 따라서 <그림 4>의 알고리듬을 가해주게 되면 모든 셀상에서 2PF2av의 고장 유무가 테스트 되게 된다.

<그림 5>와 같은 순서로 메모리 셀에 테스트 알고리듬이 가해지게 되는데 진한 색의 두 셀에 테스트 알고리듬을 가해주면, 그 셀들이 해당된 열의 고장 유무를 알 수 있다. 이와 같이 사선으로 진행하며 알고리듬을 가해주면, 전체 메모리 셀 어레이에 대해 고장을 판별할 수 있다.

효율적인 테스트 알고리듬을 위해 이와 같은 2PF2av에 대한 테스트 알고리듬도 다른 고장 종류들과 함께 March C- 변형 알고리듬에 포함시킨다면, 보통의 단일 포트 메모리의 테스트 알고리듬과 같은 길이의 알고리듬으로 이중 포트 메모리에서 발생하는 모든 고장을 검출할 수 있게 된다. <그림 2>의 테스트 알고리듬에서는 하나의 포트를 통해 쓰기 동작이 수행될 때 나머지 포트는 아무런 동작을 하지 않게 된다(NOP). 이 NOP 대신 피결합셀에 읽기 동작을 적절히 추가해 주면 <그림 4>의 테스트 알고리듬을 포함시킬 수 있게 된다. 마지막으로, 두 포트에 고르게 읽기와 쓰기 동작을 나누어 수행하도록 알고리듬을 짜 주면 <그림 6>의 테스트 알고리듬을 얻을 수 있다.

$\Updownarrow_{i=0}^{R-1}\Updownarrow_{j=0}^{C-1} w_{i,j}0;$ 은 0부터 R-1까지 행의 주소를, 0부터 C-1까지 열의 주소를 증가 또는 감소시키면서 i행, j열의 셀에 0을 쓰는 동작을 수행함을 나타낸다.

$\Updownarrow_{i=0}^{R-1}\Updownarrow_{j=0}^{C-1} r_{i,j}0;$ 은 마찬가지로 주소를 변화시키며 0을 읽는 동작을 나타내며, NOP는 아무런 동작을 하지 않는 것을 나타낸다. $\Uparrow$ 는 주소를 증가시키고, $\Downarrow$ 는 주소를 감소시키는 동작을 뜻하는 표기이다. 또한 :는 서로 다른 포트를 통하여 동시에 :의 좌변과 우변의 동작을 수행하는 것을 나타낸다.

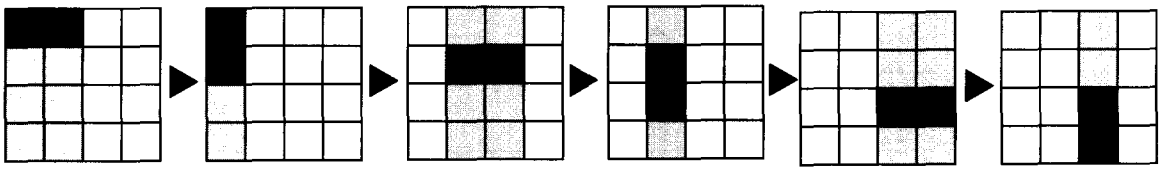


그림 5. 2PF2av의 테스트 알고리듬이 가해지는 순서
Fig. 5. Applying order of the test algorithm for 2PF2av

$$\Updownarrow_{i=0}^{R-1}\Updownarrow_{j=0}^{C-1}(w_{i,j}0{:}NOP);$$

$$\Uparrow_{i=0}^{\frac{R-1}{2}}\Uparrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,r_{i,j}0{:}r_{i,j}0\ ,w_{i,j}1{:}r_{i+1,j}0);\quad \Uparrow_{i=\frac{R-1}{2}+1}^{R-1}\Uparrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,r_{i,j}0{:}r_{i,j}0\ ,w_{i,j}1{:}r_{i-1,j}1);$$

$$\Uparrow_{i=0}^{\frac{R-1}{2}}\Uparrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,r_{i,j}1{:}r_{i,j}1\ ,w_{i,j}0{:}r_{i+1,j}1);\quad \Uparrow_{i=\frac{R-1}{2}+1}^{R-1}\Uparrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,r_{i,j}1{:}r_{i,j}1\ ,w_{i,j}0{:}r_{i-1,j}0);$$

$$\Downarrow_{i=\frac{R-1}{2}+1}^{R-1}\Downarrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,r_{i,j}0{:}r_{i,j}0\ ,r_{i,j-1}0{:}w_{i,j}1);\quad \Downarrow_{i=0}^{\frac{R-1}{2}}\Downarrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,r_{i,j}0{:}r_{i,j}0\ ,r_{i,j+1}1{:}w_{i,j}1);$$

$$\Downarrow_{i=\frac{R-1}{2}+1}^{R-1}\Downarrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,r_{i,j}1{:}r_{i,j}1\ ,r_{i,j-1}1{:}w_{i,j}0);\quad \Downarrow_{i=0}^{\frac{R-1}{2}}\Downarrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,r_{i,j}1{:}r_{i,j}1\ ,r_{i,j+1}0{:}w_{i,j}0);$$

$$\Updownarrow_{i=0}^{R-1}\Updownarrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0);$$

그림 6. 이중 포트를 위한 새로운 테스트 알고리듬
Fig. 6. A new test algorithm for dual port memories.

<그림 6>의 테스트 알고리듬은 단일 포트 메모리의 테스트 알고리듬과 동일한 길이의 알고리듬으로, 단일 포트 메모리에서 발생할 수 있는 고장뿐만 아니라, 이중 포트 메모리에서 발생하게 되는 모든 고장을 검출할 수 있는 효과적인 테스트 알고리듬이다.

$$\Updownarrow_{i=0}^{R-1}\Updownarrow_{j=0}^{C-1}(w_{i,j}0{:}NOP);$$

$$\Uparrow_{i=0}^{\frac{R-1}{2}}\Uparrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,w_{i,j}1{:}r_{i+1,j}0);\quad \Uparrow_{i=\frac{R-1}{2}+1}^{R-1}\Uparrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,w_{i,j}1{:}r_{i-1,j}1);$$

$$\Uparrow_{i=0}^{\frac{R-1}{2}}\Uparrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,w_{i,j}0{:}r_{i+1,j}1);\quad \Uparrow_{i=\frac{R-1}{2}+1}^{R-1}\Uparrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,w_{i,j}0{:}r_{i-1,j}0);$$

$$\Downarrow_{i=\frac{R-1}{2}+1}^{R-1}\Downarrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,r_{i,j-1}0{:}w_{i,j}1);\quad \Downarrow_{i=0}^{\frac{R-1}{2}}\Downarrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0\ ,r_{i,j+1}1{:}w_{i,j}1);$$

$$\Downarrow_{i=\frac{R-1}{2}+1}^{R-1}\Downarrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,r_{i,j-1}1{:}w_{i,j}0);\quad \Downarrow_{i=0}^{\frac{R-1}{2}}\Downarrow_{j=0}^{C-1}(r_{i,j}1{:}r_{i,j}1\ ,r_{i,j+1}0{:}w_{i,j}0);$$

$$\Updownarrow_{i=0}^{R-1}\Updownarrow_{j=0}^{C-1}(r_{i,j}0{:}r_{i,j}0);$$

그림 7. DRDF관련 고장검출을 제외한 이중 포트 메모리를 위한 새로운 테스트 알고리듬
Fig. 7. A new test algorithm for dual port memories except detection of faults which has a DRDF properties.

DRDF와 같이 읽기 동작을 추가했을 때 검출되는 고장 모델의 발생 확률은 상대적으로 높지 않다. 그러나 레이아웃에 따라 고장 발생 확률은 달라지게 된다. 때문에 높은 수율을 얻기 위해서는 모든 종류의 고장을 고려해 주어야 한다. 이럴 경우에 <그림 6>과 같이 제안된 테스트 알고리듬을 가해주게 되면, 모든 종류의 고장 모델에 대한 테스트가 가능하게 된다. 그러나 DRDF와 같은 고장 모델들을 고려하지 않을 경우에 읽기 동작을 한번만 해주게 되면, <그림 7>과 같이 기존의 March C-와 같은 길이의 테스트 알고리듬이 된다.

Ⅳ. 성능평가

<표 4>는 각각의 고장 모델에 대해 고장 검출여부와 테스트 알고리듬의 길이를 통해 성능을 분석한 것이다. (N은 메모리 셀의 수, R은 메모리 행의 수) [2][3]의 경우 이중 포트 관련 고장 중 서로 다른 포트의 비트라인 또는 워드라인들 간의 잘못된 연결에 의한 고장만을 모델링하여 테스트 알고리듬을 만들었기 때문에 2PF2av의 일부를 제외한 다른 종류의 이중 포트 관련 고장을 검출하지 못한다. 또한 읽기 동작이 연속적으로 가해지는 부분이 없으므로 DRDF와 관련된 고장들 역시 검출하지 못한다. [5]의 경우 단일 포트 메모리를 위한 고장은 기존의 테스트 알고리듬을 이용하는 것을 가정하고, 이중 포트와 관련된 고장에 대한 검출 패턴을 제안하였다. [5]는 본 논문에서 고려하고 있는 고장 모델들을 바탕으로 테스트 알고리듬을 만들었기 때문에 이중 포트 메모리에 대한 모든 고장이 검출 가능하다. 따라서 <표 4>에서는 각각의 고장 모델에 대해 필요한 패턴의 길이를 표시하였다. [5]는 각각의 고장 모델에 대해 독립적인 테스트 알고리듬이 존재하므로 모든 고장을 검출하기 위해서 필요한 패턴을 각각 가해준다면 상당히 긴 테스트 알고리듬이 가해지게 되는 단점이 있다. 이들 테스트 패턴에는 중복되는 동작이 많으므로, 이들을 효과적으로 배치하여 모든 고장을 검출할 수 있는 알고리듬이 필요하다. 본 논문에서 제안하는 테스트 알고리듬은 전체 셀의 수가 N일 때 10N (새 알고리듬 A:<그림 7>)과 14N(새 알고리듬 B:<그림 6>)으로 알고리듬 A의 경우, DRDF와 관련된 고장은 고장 발생 확률이 상대적으로 적기 때문에 이러한 고장을 테스트 대상에서 제외하면, 연속적인 읽기 동작

표 4. 알고리듬별 성능 비교
Table 4. Comparisons of performances.

	테스트 알고리듬	[2]	[3]	[5]	새알고리듬 A	새알고리듬 B
	Test Length / Fault Model	21N	10N	고장모델마다 독립적임	10N	14N
1PF1s	SAF	○	○	기존의 단일포트 메모리를 위한 테스트 알고리듬 이용을 가정 (기존의 테스트 패턴으로는 DRDF와 CFdr이 검출되지 않음)	○	○
	TF	○	○		○	○
	RDF	○	○		○	○
	DRDF	×	×		×	○
	IRF	○	○		○	○
	NAF	○	○		○	○
1PF2s	CFds	○	○		○	○
	CFst	○	○		○	○
	CFir	○	○		○	○
	CFdr	×	×		×	○
	CFrd	○	○		○	○
	CFtr	○	○		○	○
2PF1	wDRDF&wDRDF	×	×	7N	×	○
	wRDF&wRDF	×	×		○	○
2PF2a	wCFds&wCFds	×	×	10N	○	○
2PF2v	wCFdr&wDRDF	×	×	10N	×	○
	wCFrd&wRDF	×	×		○	○
2PF2av	wCFds&wRDF	△	△	18R	○	○
	wCFds&wIRF	△	△		○	○

을 수행하지 않아도 되므로 테스트 알고리듬 길이가 March C-와 같아진다. 알고리듬 B의 경우는 높은 수율을 얻기 위해 모든 종류의 고장을 검출할 수 있도록 읽기 동작을 추가하여 10N에서 4N 늘어난 14N의 테스트 알고리듬의 길이를 갖으며, 위에서 언급한 모든 종류의 고장을 검출할 수 있는 알고리듬이다. 따라서 본 논문에서 제안한 테스트 알고리듬은 단일 포트 메모리의 고장을 테스트하는 것과 비교하여 길이가 늘어나지 않으며, 보다 다양한 고장을 검출할 수 있는 매우 효과적인 알고리듬이다.

V. 결 론

메모리의 고집적화에 따라 대용량 메모리의 테스트 시간과 이에 따르는 비용이 증가하게 되었다. 특히, 이중 포트 메모리의 수요가 증가함에 따라 짧은 시간에 효과적인 테스트를 가능하게 하는 테스트 알고리듬에 대한 연구가 중요하게 여겨지고 있다. 따라서 본 논문에서는 단일 포트 메모리 테스트에 가장 널리 사용되는 March C-알고리듬을 바탕으로 하여 읽기 동작을 추가하여 DRDF와 관련된 고장들이 검출 될 수 있도록 보완하였고, 두 포트를 사용하여 적절히 March C-의 변형 알고리듬을 가하여 이중포트에서 발생하는 여러 가지 다양한 고장들을 검출할 수 있는 테스트 알고리듬을 제안하였다. 새로운 테스트 알고리듬은 이렇듯 다양한 고장을 검출할 뿐만 아니라 단일 포트 메모리를 위한 테스트 알고리듬에 비해서 추가로 늘어나는 테스트 길이의 증가분이 없기 때문에 매우 효율적인 알고리듬이라고 할 수 있다.

참 고 문 헌

[1] S. Hamdioui, A.J. van de Goor, "Thorough testing of any multiport memory with linear tests," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, pp. 217~231.

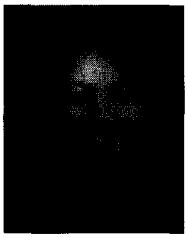
[2] W. Yuejian, S. Gupta, "Built-in self-test for multi-port RAMs," *Proc. of Test Symposium*, 1997, pp. 398~403.

[3] C. F. Wu, C. T. Huang, K. L. Cheng, C. W. Wang and C. W. Wu, "Simulation-based test algorithm generation and port scheduling for multi-port memories," *Proc. of Design Automation Conference*, 2001, pp. 301~306.

[4] P. Nagaraj, S. Upadhyaya, K. Zarrineh, D. Adams, "Defect analysis and a new fault model for multi-port SRAMs," *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2001*, pp. 366~374.

[5] S. Hamdioui, M. Rodgers, A.J. Van de Goor, "March tests for realistic faults in two-port memories," *Proc. of IEEE International Workshop on Memory Technology, Design and Testing, 2000*, pp. 73~78.

저 자 소 개



金 智 惠(正會員)

2002년 2월 연세대 기계전자공학부 졸업.(학사). 현재 연세대 전기전자공학과 석사과정

裵 相 民(正會員)

1998년 2월 광운대 제어계측공학과 졸업.(학사). 2000년 9월~2002년 8월 연세대 전기전자공학과 졸업(공학석사). 2002년 9월~현재 삼성전자 CAE 센터

宋 東 燮(正會員)

2000년 2월 건국대 전기공학과 졸업.(학사). 2000년 3월~2002년 2월 연세대 전기전자공학과 졸업(공학석사). 2002년 9월~현재 연세대 전기전자공학과 박사과정

姜 成 昊(正會員)

1986년 2월 서울대 공대 제어계측공학과 졸업. 1988년 5월 The University of Texas at Austin. 전기 및 컴퓨터공학과 졸업(석사). 1992년 5월 The University of Texas at Austin. 전기 및 컴퓨터공학과 졸업(공박). 미국 Schlumberger 연구원. Motorola 선임 연구원. 현재 연대 공과대학 전기전자공학과 부교수