논문 2004-41SD-5-14

# 이중 포트 메모리를 위한 효과적인 테스트와 진단 알고리듬

# (An Efficient Test and Diagnosis Algorithm for Dual Port Memories)

김 지 혜*, 김 홍 식**, 김 상 욱***, 강 성 호***

(JiHye Kim, Hong-Sik Kim, Sang-Wook Kim, and Sung-Ho Kang)

요 약

이중 포트 메모리의 사용이 증가함에 따라, 이중 포트 메모리의 테스트와 진단이 중요하게 여겨지고 있다. 본 논문에서는 메모리의 테스트 과정에서 고장이 검출되었을 때, 발생한 고장의 종류를 세부적으로 분류할 수 있는 새로운 진단 알고리듬을 제안한다. 본 알고리듬에서는 진단을 위한 패턴뿐만 아니라 테스트 결과를 통하여 얻을 수 있는 정보를 이용하여, 진단 과정의 효율성을 증대하였으며, 이중 포트 메모리에서 발생할 수 있는 다양한 고장에 대하여 진단이 가능하다.

Abstract

As dual port memories are being frequently used, test and diagnosis for dual port memories becomes more important. In this paper, anew diagnosis algorithm which can classify faults in detail when the fault is detected during test process is developed. The new algorithm increases its efficiency by using the information that can be obtained by test results as well as results using additional diagnostic pattern set. In addition the algorithm can diagnose various fault models for dual port memories.

**Keywords**: 이중 포트 메모리, 고장 모델, 고장 테스트, 고장 진단

## I. 서 론

메모리 설계 기술과 미세 가공기술의 발전에 따라 메모리의 집적도가 급격하게 향상되었다. 이러한 메모리의 고집적화는 복잡하고 정밀한 설계와 제조 공정을 필요로 한다. 이렇듯 복잡하고 정밀한 설계와 공정은 더욱 많은 고장을 야기한다[1],[2]. 따라서 메모리에서 발생하게 되는 고장을 분석하는 고장 진단 과정을 통해, 고장 난 메모리를 수리하여 재생산하고, 설계와 공정상의 문제점을 파악하는 과정은 더더욱 중요하게 여겨지고 있다. 최근 들어 이중 포트 메모리의 사용이 증가하고 있다[3]-[5]. 이중포트 메모리는 두 개의 포트를 통해 메모리에 접근할 수 있으므로 많은 데이터를 짧은 시간에 처리하는 영상 처리나, 마이크로프로세서, 네트워크 애플리케이션 등에 사용된다. 이중포트 메모리는 단일 포트 메모리에 비해 더욱 복잡한 회로를 가지고 있다. 그러므로 이중포트 메모리를 테스트하거나 고장을 진단하는 것은 단일 포트 메모리에 비해 더욱 복잡하고 어려워진다. 메모리의 고장 진단은 크게 두 가지로 나눌 수 있다. 하나는 고장이 발생한 위치를 알아내는 것이고, 다른 하나는 고장의 종류를 알아내는 것이다. 고장이 발생한 위치에 대한 정보는 고장 난 셀을 수리하여 재생산하는데 사용되며, 고장 종류에 대한 정보는 메모리 설계나 공정상의 문제점 파악에 유용하게 사용될 수 있다. 메모리의 경우, 다른 논리 회로들과 달리, 주소를 가지고 접근하게 되기 때문에, 고장이 발생한 위치는 고장이 발견된 셀의 주소 정보를 이용하게 되면 어렵지 않게 알아낼 수 있다. 그러나 고장의 종류까지 알아내

* 정회원, 삼성전자 반도체 총괄 시스템 LSI 사업부
(Samsung Electronics, System LSI Division)
** 정회원, LG전자 시스템 IC 사업부
(LG Electronics, System IC Division)
*** 정회원, 연세대학교 전기전자공학부
(Dept. of Electrical and Electronic Engineering, Yonsei Univ.)
※본 연구는 정보통신부 대학 IT연구센터 육성 지원사업의 연구결과로 수행되었습니다.
접수일자: 2004년1월6일, 수정완료일: 2004년5월3일

기 위해서는 진단을 위한 추가적인 알고리듬이나, 고장 딕셔너리 등이 필요하다.

[6]에서는 고착고장과 결합고장에 대한 진단 알고리듬이 제안되었다. 이는 고장이 발생한 부분의 셀을 수리하기 위해 고장 위치를 알아내는 진단을 목적으로 하고 있으나, 고착고장과 결합고장만을 고려하고 있기 때문에 다양한 고장을 진단할 수 없다는 단점을 가지고 있다. [7]의 경우는 길이 12N의 March 알고리듬을 가하여 5개의 고장 그룹으로 나눈 후, 고장을 유발시키는 셀(Aggressor Cell)을 찾기 위한 패턴과 워드라인내의 결합고장을 검출하기 위한 패턴 그리고 어드레스 고장을 검출하기 위한 MATS+ 패턴들을 가하여 고장의 위치와 종류를 진단한다. 이와 같은 경우 다양한 고장이 고려되었지만 어드레스 고장이 고착고장 또는 결합고장의 형태로 나타나게 되는 경우가 나타나기 때문에 각각의 고장에 대한 구분이 모호한 경우가 생기게 되어 정확하게 고장 진단이 되지 않는 경우가 생기게 되는 문제점을 갖고 있다. [8]에서는 메모리의 수리를 위한 위치 진단과 고장 종류의 분석을 구별하여 서로 다른 진단 패턴을 가해주는 알고리듬을 제안하였다. 수리를 위해서는 March-CW라는 테스트 알고리듬을 가하여 고장이 발생한 경우 고장 셀의 주소와 고장을 유발시키는 동작의 정보 등을 EOP(Error Operation) 레지스터에 저장하여 고장 수리를 위한 위치 정보로 이용하는 방법을 사용한다. 고장 분석을 위해서는 IFA 9N이라는 테스트 알고리듬에 대한 고장 딕셔너리를 생성하여 이를 이용하게 되는데, 이 경우에도 고착고장과 결합고장의 일부 정도가 구별되는 정도이며, 고장 종류의 분석을 위한 정확한 정보를 제공하지는 못한다. [9]의 경우, March C-와 March 17N의 테스트 알고리듬에 대해 고장 딕셔너리를 생성하여 고착고장과 결합고장에 대한 진단을 수행하게 되는데, 이 경우도 마찬가지로 다양한 다른 종류의 고장에 대해 진단이 이루어지지 못하다는 단점을 가지고 있다. [10]에서는 March CL 알고리듬이 가해진 후에 고장이 9개의 그룹으로 분류되고 그 후에 3N에서 4N의 길이를 갖는 March-like 알고리듬이 가해져 고장을 유발시키는 셀의 위치를 찾는다. [6-10]의 경우, 모두 정확하고 효과적인 진단이 불가능할 뿐만 아니라 단일 포트 메모리에서만 적용 가능한 알고리듬으로 이중 포트 메모리에서는 사용할 수 없다. [11]은 이중 포트 메모리에 대한 고장 진단 알고리듬을 제안하였다. 테스트가 수행된 이후에 고장이 발견되었을 때 추가적으로 가해줄 수 있는 알고리듬으로, 단일 포트와 관련된 고장에서부터 시작하여 이중 포트와 관련된 고장까지 순차적으로 진단 알고리듬을 가하여, 해당 고장 종류를 알아내는 방식이다. 비교적 다양한 고장 종류에 대해 고려되었으며, 이중포트 메모리에서도 적용 가능한 알고리듬이라는 장점을 가지고 있지만, 이 역시 고착고장과 어드레스 디코더 고장, 결합고장 그리고 두 개의 포트를 동시에 사용할 때 발생하게 되는 고장 등의 구별이 모호한 경우 존재하며, 테스트와 별도로 추가적인 진단 알고리듬이 가해지기 때문에 수행시간이 길어지는 단점을 가지고 있다. 이와 같이 지금까지의 연구는 단일 포트 메모리의 고장을 진단하기 위한 진단 알고리듬이 대부분이며 이중 포트 메모리의 사용이 증가하는데 비해 이중 포트 메모리의 고장 진단에 관한 연구, 특히 고장의 종류 분석을 위한 연구는 매우 부족한 실정이다. 이렇듯 기존의 연구에서는 다양한 고장에 대한 고려, 고장 모델간의 구별의 모호성, 수행 알고리듬의 길이 그리고 이중 포트 메모리의 적용 가능성 등의 문제점이 나타났다. 이러한 문제점들을 극복하기 위해서는 다양한 고장을 검출할 수 있는 효과적인 테스트 알고리듬이 필요하며, 고장 모델간의 동작을 명확히 기술한 새로운 고장 모델의 도입이 필요하다. 또한 고장 딕셔너리 등을 효과적으로 이용하여 추가되는 진단 알고리듬의 길이를 최적화하면서도 이중포트 메모리에 대해서도 적용이 가능하도록 하여야 할 것이다. 따라서 본 연구에서는 위에서 지적한 문제점들을 극복하고, 고장 딕셔너리와 추가적인 진단 알고리듬을 이용하여 효율성을 최대화시키고, 이중 포트 메모리에서 발생 가능한 다양한 고장에 대해 그 종류를 알아낼 수 있는 효과적인 진단 알고리듬을 제안하고자 한다.

## Ⅱ. 고장 모델

기존의 진단 알고리듬의 가장 큰 문제점 중의 하나가 다양한 고장이 고려되지 못하고 있다는 점이다. 진단 알고리듬은 테스트의 결과에 따라 달라진다. 테스트를 통해 검출될 수 있는 고장의 종류에 따라 이를 구별할 수 있는 알고리듬이 달라질 수 있기 때문이다. 설계와 공정상의 문제점 파악을 위한 정보를 얻기 위해서는 가능하면 다양한 고장 모델들을 고려한 테스트가 이루어지고, 이들의 종류를 구별할 수 있어야 할 것이다. 따라서 선행 연구를 통해 다양한 고장을 검출하기 위해 개발된 테스트 알고리듬을 소개하고, 이를 바탕으로 한 진단 알고리듬을 제안하고자 한다.

[12]에서는 메모리 셀 안에서 발생할 수 있는 개방(open), 단락(short), 연결(bridge) 등의 모든 결함(spot defect)들을 바탕으로 IFA(Inductive Fault Analysis)를 통해 새로운 고장 모델을 제시하였다. 이 고장 모델들은 실제 메모리 셀의 각 노드에서 발생 가능한 결함에 대해 고장이 발현되는 특성에 따라 고장 모델들로 구분하였으므로, 진단 과정에서 고장의 종류를 구분해 내면, 그 고장에 해당하는 발생 가능한 실제 결함들의 종류가 결정되므로 고장에 대한 분석에 매우 유용하게 사용될 수 있다. 따라서 [12]의 [13]의 테스트 알고리듬을 [12]의 고장 모델을 기반으로 제안되었다. [7]의 고장 모델들을 바탕으로 하여 이들을 검출하기 위한 테스트 알고리듬을 단계적으로 제안하였다. [7]에서의 고장 모델 중 읽기의 결과가 랜덤하게 나타나는 고장이나, 시간적 요소를 포함시켜야 하는 고장 모델 그리고 이중 포트와 관련된 고장 중 동일한 셀에 읽기와 쓰기 동작을 동시에 수행해 주어야 발생하는 고장 등은 어떠한 테스트 패턴으로도 고장 검출을 보장할 수 없는 고장 모델이거나 본 논문에서 고려하고 있는 일반적인 이중 포트 메모리의 동작에 해당하지 않는 동작으로 인한 고장모델이므로 검출 대상에서 제외한다.

이 장에서는 이중 포트 메모리를 위한 고장 모델과 검출 대상이 되는 고장 모델을 소개한다.

표 1. 하나의 포트와 관련 된 고장(1PF1s)
Table 1. The target fault models related to one port(1PF1s).

| 분류 | 고장모델 | 고장 프리미티브 |
|---|---|---|
| 1PF1 | SAF | <∀/0/->,<∀/1/-> |
| | TF | <w↑/0/->,<w↓/1/-> |
| | RDF | <r0/↑/1>,<r1/↓/0> |
| | DRDF | <r0/↑/0>,<r1/↓/1> |
| | IRF | <r1/1/0>,<r0/0/1> |
| | NAF | <w↑/0/->,<w↓/1/-> |

(a) 하나의 셀과 관련 된 고장
(a) Fault models involved one cell.

| 분류 | 고장모델 | 고장 프리미티브 |
|---|---|---|
| 1PF2 | CFds | <wx;1/↓/->,<rx;0/↑/-><br><rx;1/↓/->,<wx;0/↑/-> |
| | CFst | <1;1/0/->,<1;0/1/-><br><0;1/0/->,<0;0/1/-> |
| | CFir | <0;r0/0/1>,<0;r1/1/0><br><1;r0/0/1>,<1;r1/1/0> |
| | CFdr | <0;r0/↑/0>,<0;r1/↓/1><br><1;r0/↑/0>,<1;r1/↓/1> |
| | CFrd | <0;r0/↑/1>,<0;r1/↓/0><br><1;r0/↑/1>,<1;r1/↓/0> |
| | CFtr | <0;w↓/1/->,<0;w↑/0/-><br><1;w↓/1/->,<1;w↑/0/-> |

(b) 두 개의 셀과 관련 된 고장
(b) Fault models involved two cells.

## 2.1 하나의 포트와 관련 된 고장(1PFs)

하나의 포트와 관련 된 고장은 다시 하나의 셀과 관련 된 고장(1PF1s)과 두 개의 셀이 관련 된 고장(1PF2s)으로 나누어진다.

1PFs 중 검출 대상이 되는 고장 모델은 표 1과 같으며 고장 프리미티브의 표기법은 [6]의 표기법을 사용하였다. 표 1(a)는 1PF1s이다. 1PF1s는 메모리 셀 내부의 단락과 개방에 의해 야기되는 고장을 말한다. ∀는 임의의 동작을, w↑(↓)는 상승(하강) 천이 동작을 나타내며 w1(0)은 1(0)을 메모리 셀에 쓰는 동작을, r1(0)은 읽는 동작을 나타낸다. 고장 프리미티브를 <S/F/R>과 같은 형태로 표기할 때 S는 고장을 유발시키는 동작을 나타내며, F는 그 때의 셀이 갖는 데이터나 셀 내부의 동작 상태를 나타내며 R는 읽기 동작을 수행하였을 때의 출력단에 나타나는 결과 값을 나타낸다. 표 1(b)는 1PF2s 이다. 1PF2s는 두 개의 셀 내부의 노드 간의 연결이나 간섭 현상에 의해 발생하는 고장이다. 고장과 관련 된 두 셀 중 고장을 유발시키는 셀을 결합셀(aggressor cell : a-cell)이라고 하며 고장이 유발되는 셀을 피결합셀(victim cell : v-cell)이라고 하며 실제 고장을 v-cell에서 나타난다. 표 2에서의 고장 프리미티브의 표기는 <Sa;Sv/F/R>로, Sa는 고장을 유발시키는 a-cell의 상태나 동작을 Sv는 v-cell의 상태나 동작을 나타내며 기호 ';'은 a-cell과 v-cell의 동작을 구분하는 기호이다.

## 2.2 두 개의 포트와 관련 된 고장 (2PFs)

두 개의 포트와 관련 된 고장(2PFs)은 이중 포트 메모리에서 하나 또는 두 개의 셀이 두 포트를 통해 동시에 접근 되었을 때 나타날 수 있는 고장을 말한다. 2PFs는 두 개의 약고장이 합쳐져 하나의 고장의 형태로 나타나는 경우를 말하며 2PFs 또한 하나의 셀과 관련된 고장(2PF1s)과 두 개의 셀이 관련 된 고장(2PF2s)으로 나누어진다. 2PF2s는 다시 고장을 유발하는 동작

이 v-cell에 가해지는 2PF2v와 a-cell에 가해지는 2PF2a 그리고 a-cell과 v-cell에 동시에 가해지는 2PF2av의 세 종류로 구분된다.

표 3에서의 고장 프리미티브의 표기는 2PF1의 경우 <S1:S2/F/R>로, 기호 ":"은 두 포트를 통해 동시에 일어나는 동작을 기술한다. 다시 말해, S1과 S2는 두 포트를 통해 동일 셀에 동시에 수행되는 동작으로 고장을 유발하는 동작을 말한다. 따라서 2PF1s는 두 포트를 통해 고장 셀에 각각 S1과 S2의 동작을 가해 주었을 때 고장이 유발된다. 2PF2a의 경우, <Sa:Sa;Sv/F/R>로 두 개의 Sa는 두 포트를 통해 a-cell에 동시에 가해지는 동작이며, Sv는 이 때의 v-cell의 상태를 나타내게 된다. 2PF2v를 나타내는 <Sa;Sv:Sv/F/R>의 경우 Sa는 a-cell의 상태를 나타내고 두 개의 Sv는 v-cell에 두 포트를 통해 동시에 가해지는 동작을 나타낸다. 2PF2av의 경우는 <Sa:Sv/F /R>이며, Sa와 Sv 부분에 기술된 동작이 각각 a-cell과 v-cell에 두 포트를 통해 동시에 가해지는 동작을 나타낸다. 검출 대상이 되는 고장 모델은 다음과 같은 동작들만이 이중 포트 메모리에서 수행 가능한 동작이라는 가정 하에 결정 되었다.

- 하나의 셀에 두 개의 읽기 동작을 동시에 수행한다.
- 서로 다른 셀에 동시에 읽기 동작을 수행한다.
- 서로 다른 셀에 동시에 쓰기 동작을 수행한다.
- 서로 다른 셀에 동시에 읽기 동작과 쓰기 동작을 수행할 수 있다.

2PFs의 고장 모델 중 하나의 셀에 동시에 읽고 쓰는 동작이 수행되었을 때 발생할 수 있는 고장 모델은 위에서 가정한 일반적인 이중 포트 모델의 동작에 해당하지 않으므로 제외하였다. 표 2는 검출 대상으로 하는 2PFs 고장 모델이다.

표 2. 이중 포트 메모리를 위한 검출 대상 고장 모델 (2PFs)

Table 2. The fault models related to two ports(2PFs).

| 분류 | 고장 모델 | 고장 프리미티브 |
|---|---|---|
| 2PF1 | wDRDF&wDRDF | <r0;r0/↑/0>, <r1:r1/↓/1> |
| | wRDF&wRDF | <r1:r1/↓/0>, <r0:r0/↑/1> |
| 2PF2v | wCFds&wCFds | <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> |

## Ⅲ. 알고리듬

본 논문에서 제안하는 알고리듬은 그림 1의 흐름도와 같다. 먼저 테스트와 진단 과정이 시작되면, 그림 1의 테스트 패턴(14N)이 메모리 셀에 가해져 테스트가 수행된다. 이 테스트 패턴은 이중 포트 메모리의 고장 모델들 중 2PF2av의 고장모델을 제외한 모든 검출 대상 고장들을 검출할 수 있는 알고리듬이다. 따라서 테스트 과정을 수행하여 고장이 검출되면 2PF2av고장 이외의 고장이라고 할 수 있으므로 이를 확인하는 진단 과정이 수행되게 된다. 진단 과정 중 먼저 테스트과정에서 얻을 수 있는 응답 정보와 고장 딕셔너리를 이용하여 A부터 P까지의 고장의 그룹이 분류된다. 고장 그룹이 정해지게 되면, 각 그룹마다 세부적으로 고장 종류를 구분할 수 있도록 고려된 추가 진단패턴이 가해지게 되어 고장의 종류가 분석된다. 길이 *14N*의 테스트 패턴이 가해졌을 때 고장이 나타나지 않게 되면, 그림 2에서의 길이 8C의 테스트 패턴이 추가적으로 가해지게 된다. 이 테스트 과정을 통해서는 2PF2av고장 존재 여부를 테스트하게 되고, 고장이 발견되면, 진단패턴이 추

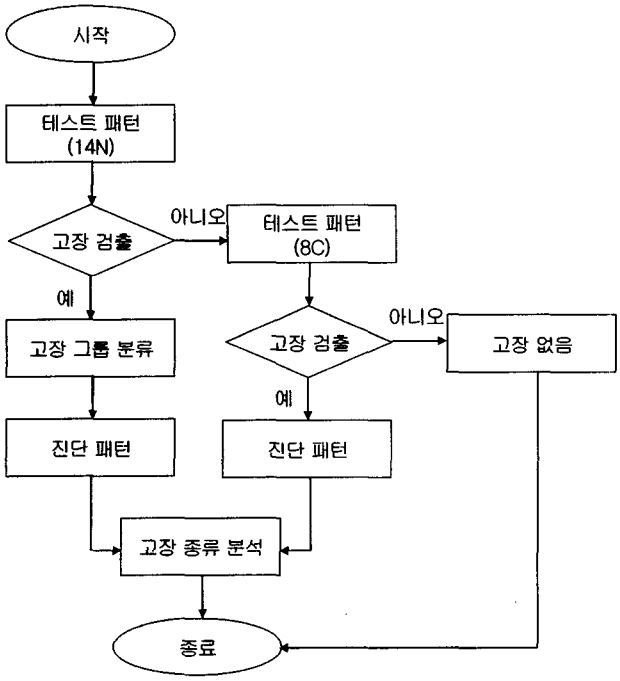



그림 1. 테스트와 진단 알고리듬

Fig. 1. A test and diagnosis algorithm.

가로 가해져 세부적인 고장 종류를 분석하게 되며, 고장이 발견되지 않게 되면, 고장이 없는 것으로 판단되어 알고리듬이 종료된다. 따라서 전체적으로 두 가지의 테스트 과정을 통해 고려하고자 하는 검출 대상 고장 모델에 대해 고장 유무가 테스트되며, 고장이 발견되었을 경우, 테스트 응답 정보와 딕셔너리 그리고 추가 진단 패턴 등을 이용하여 세부적으로 고장의 종류를 분석하게 된다.

## Ⅳ. 테스트 패턴과 고장 딕셔너리

어떠한 테스트 패턴을 이용하여 테스트를 수행하느냐에 따라 그에 따르는 진단 패턴이 달라진다. 테스트 패턴이 검출할 수 있는 고장을 대상으로 진단 과정이 수행되기 때문이다. 테스트 패턴은 고려하고자 하는 모든 검출 대상 고장들을 검출할 수 있어야 하며, 진단을 용이하고 명확하게 할 수 있도록 하는 테스트 패턴이어야 한다. 적합한 테스트 패턴이 정해진 후에는 그 패턴에 대한 결과를 예측하여 고장 딕셔너리를 생성하게 된다. 따라서 이중 포트 메모리의 진단에 용이한 테스트 패턴과 고장 딕셔너리의 생성 과정에 대해 소개하고자 한다.

### 4.1 테스트 패턴

[13]에서 최종적으로 제안된 테스트 패턴의 경우 검출 대상으로 하는 모든 검출 대상 고장 모델에 대해 고장 검출이 가능하다. 그러나 이 테스트 패턴의 경우, 두 포트를 통해 서로 다른 주소의 셀에 접근하여 읽기와 쓰기 동작을 각각 수행하는 동작으로 인해 같은 고장에 대해 고장이 발생한 위치에 따라서도 테스트의 결과가 다르게 나타날 수가 있기 때문에 진단을 위한 고장 딕셔너리를 생성하기가 어렵게 된다. 따라서 같은 셀에 동시에 읽고 쓰는 동작을 포함하지 않는 그림 2의 테스트 패턴을 이용한다. 그리고 이 테스트 패턴으로 검출할 수 없는 2PFav의 고장을 검출하고 진단하기 위한 추가적인 테스트와 진단 알고리듬을 가해주는 방법을 사용하고자 한다. 그림 2의 테스트 패턴을 사용하면, 2PFav고장을 제외한 모든 검출 대상 고장을 검출할 수 있다.

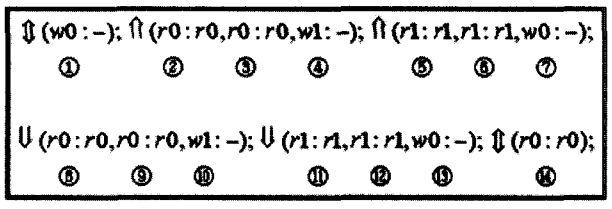



그림 2. 2PF2av의 검출을 제외한 이중포트를 위한 테스트 패턴

Fig 2. A test algorithm for dual port memories (Except detection of 2PF2av).

그림 2의 테스트 패턴의 길이는 14N(N은 메모리 셀의 수)으로 모두 14가지의 마치요소로 구성되어 있으며 각각의 마치요소마다 나타나게 되는 결과에 따라 고장 딕셔너리를 만들게 된다. w0(w1)와 r0(r1)은 각각 0(1)의 쓰기 동작과 0(1)의 읽기 동작을 나타내며, 기호 ':'의 좌변과 우변에 오는 동작은 두 개의 포트를 통해서 동시에 수행함을 나타낸다. ( )안의 동작들은 순서대로 한번씩 수행시키고 난 후 주소값을 변경시켜주게 되며 ⇑, ⇓, ⇕이 각각 주소의 진행방향을 나타내며 ⇑는 주소가 증가하는 방향의 순차적 증가를 뜻하며, ⇓는 감소하는 방향으로 진행을 뜻한다. ⇕는 증가와 감소의 방향에 상관없이 순차적으로 진행시킴을 나타낸다. 본 논문에서 ⇕의 경우, ①의 경우에는 증가의 방향으로, ⑭의 경우에는 감소의 방향으로 가정하고 고장 딕셔너리를 만든다.

그림 2의 테스트 패턴을 가해줄 때 검출할 수 없는 고장 모델은 2PF2av에 속하는 고장들로, 이러한 고장은 두 포트를 통해 서로 다른 셀에 동시에 접근하여 각각 읽고 쓰는 동작을 수행할 때 유발시킬 수 있다. 따라서 그림 2의 테스트 패턴을 가해준 후, 고장이 발견되지 않으면, 그림 3의 테스트 패턴을 추가로 가하여 2PF2av 고장 여부를 알아내어야 한다. 2PF2av의 경우 모든 셀에 대하여 테스트를 수행할 필요가 없고 메모리의 비트라인에 대해서만 테스트를 해 주면 되기 때문에 패턴의 길이가 상대적으로 짧아지게 된다. 2PF2av를 위한 테스트 패턴의 길이는 8C(C는 비트라인의 수)이다.

$\Uparrow_{i=0}^{C-1}(w_{1,i}1:r_{2,i}0);$ 의 경우, i는 메모리의 열주소값을 나타내므로, 행주소는 고정시키고 열주소를 0에서 C-1

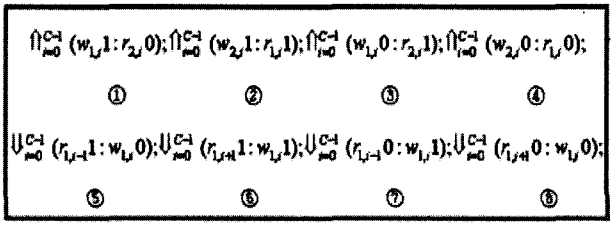



그림 3. 2PF2av를 검출하기 위한 추가적인 테스트 패턴

Fig. 3. An additional test algorithm for detecting 2PF2av.

표 3. 14N 테스트 패턴을 위한 고장 딕셔너리 (파트 1)-계속
Table 3. A fault dictionary for test pattern 14N (part 1)-continued.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Deceptive Read Destruction Coupling Fault(CFdr)** | | | | | | | | | | | | | | | | |
| <0;r0/↑/0> | a>v | | S | D | | | | | | | | | | | S | M |
| | a<v | | | | | | | | S | D | | | | | S | N |
| <0;r1/↓/1> | a>v | | | | | | | | | | | S | D | | | O |
| | a<v | | | | | S | D | | | | | | | | | P |
| <1;r0/↑/0> | a>v | | | | | | | | S | D | | | | | | N |
| | a<v | | S | D | | | | | | | | | | | | M |
| <1;r1/↓/1> | a>v | | | | | S | D | | | | | | | | | P |
| | a<v | | | | | | | | | | | S | D | | | O |
| **Read Destructive Coupling Fault(CFrd)** | | | | | | | | | | | | | | | | |
| <0;r0/↑/1> | a>v | | SD | D | | | | | | | | | | | SD | F |
| | a<v | | | | | | | | SD | D | | | | | SD | I |
| <0;r1/↓/0> | a>v | | | | | | | | | | | SD | D | | | G |
| | a<v | | | | | SD | D | | | | | | | | | H |
| <1;r0/↑/1> | a>v | | | | | | | | SD | D | | | | | | K |
| | a<v | | SD | D | | | | | | | | | | | | J |
| <1;r1/↓/0> | a>v | | | | | SD | D | | | | | | | | | H |
| | a<v | | | | | | | | | | | SD | D | | | G |
| **Transition Coupling Fault(CFtr)** | | | | | | | | | | | | | | | | |
| <0;w↓/1/-> | a>v | | | | | | | | | | | | | S | D | L |
| | a<v | | | | | | | S | D | D | | | | | | K |
| <0;w↑/0/-> | a>v | | | | S | D | D | | | | | | | | | H |
| | a<v | | | | | | | | | | S | D | D | | | G |
| <1;w↓/1/-> | a>v | | | | | | | S | D | D | | | | | | K |
| | a<v | | | | | | | | | | | | | S | D | L |
| <1;w↑/0/-> | a>v | | | | | | | | | | S | D | D | | | G |
| | a<v | | | | S | D | D | | | | | | | | | H |
| **wDRDF&Wdrdf** | | | | | | | | | | | | | | | | |
| <r0;r0/↑/0> | | | S | D | | | | | S | D | | | | | | C |
| <r1;r1/↓/1> | | | | | | S | D | | | | | S | D | | | D |
| **wRDF&Wrdf** | | | | | | | | | | | | | | | | |
| <r0;r0/↑/1> | | | SD | D | | | | | SD | D | | | | | SD | B |
| <r1;r1/↓/0> | | | | | | SD | D | | | | | SD | D | | | A |
| **wCFds&wCFds** | | | | | | | | | | | | | | | | |
| <r0;r0;0/↑/-> | a>v | | | | | | | | SD | D | | | | | SD | I |
| | a<v | | SD | D | | | | | | | | | | | S | J |
| <r0;r0;1/↓/-> | a>v | | S | | | D | D | | | | | | | | | H |
| | a<v | | | | | | | | S | | | D | D | | | G |
| <r1;r1;0/↑/-> | a>v | | | | | S | | | D | D | | | | | | K |
| | a<v | | | | | | | | | | | S | | | D | L |
| <r1;r1;1/↓/-> | a>v | | | | | | | | | | | SD | D | | | G |
| | a<v | | | | | SD | D | | | | | | | | | H |
| **wCFdr&wDRDF** | | | | | | | | | | | | | | | | |
| <0;r0;r0/↑/0> | a>v | | S | D | | | | | | | | | | | S | M |
| | a<v | | | | | | | | S | D | | | | | S | N |
| <0;r1;r1/↓/1> | a>v | | | | | | | | | | | S | D | | | O |
| | a<v | | | | | S | D | | | | | | | | | P |
| <1;r0;r0/↑/0> | a>v | | | | | | | | S | D | | | | | | N |
| | a<v | | S | D | | | | | | | | | | | | M |
| <1;r1;r1/↓/1> | a>v | | | | | S | D | | | | | | | | | P |
| | a<v | | | | | | | | | | | S | D | | | O |
| **wCFrd&wRDF** | | | | | | | | | | | | | | | | |
| <0;r0;r0/↑/1> | a>v | | SD | D | | | | | | | | | | | SD | F |
| | a<v | | | | | | | | SD | D | | | | | SD | I |
| <0;r1;r1/↓/0> | a>v | | | | | | | | | | | SD | D | | | G |
| | a<v | | | | | SD | D | | | | | | | | | H |
| <1;r0;r0/↑/1> | a>v | | | | | | | | SD | D | | | | | | K |
| | a<v | | SD | D | | | | | | | | | | | | J |
| <1;r1;r1/↓/0> | a>v | | | | | SD | D | | | | | | | | | H |
| | a<v | | | | | | | | | | | SD | D | | | G |

까지 증가시키면서 1행 i열에 1을 쓰고 동시에 2행 i열의 데이터 0을 읽는 동작을 수행함을 나타낸다. 기본적인 표기법은 그림2의 테스트 패턴에서와 동일하다.

### 4.2 고장 딕셔너리

고장의 종류를 구분하기 위한 고장 진단을 효율적으로 하기 위해서는 테스트를 통해서 얻을 수 있는 정보를 최대한 활용해야 한다. 테스트를 통해서는 고장의 존재 유무에 대한 정보뿐만 아니라 테스트 패턴이 가해졌을 때 어떤 부분에서 고장이 검출되었는가에 대한 정보까지 얻을 수 있다. 이러한 정보를 진단에 이용하기 위해서는 가해질 테스트 패턴에 대한 고장 딕셔너리를 사전에 생성해 두어야 한다. 고장 딕셔너리는 각각의 고장에 대해, 하나의 고장이 메모리셀 내에 존재할 때의 테스트 패턴에 대해 예상되는 응답 리스트를 만들어 놓은 것이다. 이러한 고장 딕셔너리는 메모리 셀에 가해지게 테스트 패턴에 따라 달라지게 되며, 기존의 논문이 단일 포트 메모리에 대해 테스트 패턴을 가하고 각각의 고장 모델마다의 응답 리스트를 만들어 놓은 것에 비해, 본 논문에서는 이중 포트 메모리를 위한 테스트 패턴인 그림 1의 테스트 패턴에 대해 고장 모델보다 더 세분화하여 각각의 고장 모델의 세부 동작을 기술한 고장 프리미티브마다, 결합고장의 경우 고장 프리미티브(fault primitive) 정보를 이용하게 되면 어렵지 않게 알아낼 수 있다. 그러나 고장의 종류까지 알아내기 위해 뿐만 아니라 결합셀과 피결합셀의 위치 관계에 따라 더욱 자세하게 고장에 대한 결과를 나타내도록 하는 새로운 고장 딕셔너리를 제안한다. 이는 본 논문에서 제안하는 알고리듬이 단일 포트 메모리의 진단에 비해, 또 기존의 이중포트 메모리를 위한 진단에 비해 더욱 다양한 고장 모델을 고려하고 있기 때문에 이들의 종류를 구분하기 위해서는 테스트 과정에서 최대한 자세하고 정확한 정보를 얻기 위해서이며, 실제로 같은 고장 모델이더라도 프리미티브나 결합셀과 피결합셀의 위치 관계에 따라 테스트 응답이 달라지므로 본 논문에서 제안하는 고장 딕셔너리는 기존의 것보다 정확하고 자세한 정보를 제공한다. 실제의 고장 딕셔너리는 각각의 고장에 대해 테스트 패턴이 메모리 셀에 순차적으로 가해질 때 나타나는 고장 응답이 고장이 없는 메모리 셀에서의 응답과 동일하면, 그 동작에 대해 0을 동일하지 않으면 1을 저장하는 방식으로 생성된다. 그러나 본 논문에서는 이해를 돕기 위해, 각각의 고장 모델과 그에 대한 응답에 대해 고장을 유발시키는 동작에 S(Sensitization)를 표시하고, 고장이 검출되는 동작에 대해 D(Detetion)를 표시하고, 나머지 동작에 대해서는 아무런 표시도 하지 않았다. 예를 들어 고착고장 (Stuck-At Fault :SAF)는 메모리 셀의 데이터가 특정 값으로 고정되어 변하지 않는 형태의 고장을 말한다. <∀/0/->의 경우, 셀 값이 0으로 고정되어 있는 고장이므로, 0을 쓰고(w0) 읽는 동작(r0)에서는 고장이 발견되지 않지만, 이 셀에 1을 쓰고(w1), 읽으면(r1) 고장이 발견되게 된다. 따라서 테스트 알고리듬 수행 과정 중, w1에서 고장이 유발되고, r1에서 고장이 검출되게 된다. 그러므로 w1의 동작인 4, 10에서 고장이 유발되며, r1의 동작인 5, 6, 11, 12 에서 고장이 검출된다. <∀/0/->처럼 5, 6, 11, 12 동작에서 고장이 검출되는 고장 종류들을 묶어 그룹 A로 정하였다. <∀/1/->의 경우, 셀의 값이 1로 고정되어 있는 고장이므로, <∀/0/->와 반대로 w0과 r0 동작에서 고장이 유발되고, 검출되게 된다. 따라서, w0의 동작인 1,7, 13에서 고장이 유발되고, r0인 2, 3, 8, 9, 14에서 고장이 검출되게 된다. 이와 같이 2, 3, 8, 9, 14에서 고장이 검출되는 고장 종류들을 그룹 B로 정하.

표 4. 8C 테스트 패턴을 위한 고장 딕셔너리 (파트 2)
Table 4. A fault dictionary for test pattern 8C (part 2)

| | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ | ⑧ | Fault group |
|---|---|---|---|---|---|---|---|---|---|
| **wCFds&wRDF** | | | | | | | | | |
| <w0:r0/↑/1> | | | | SD | | | | SD | Q |
| <w0:r1/↓/0> | | | SD | | SD | | | | R |
| <w1:r0/↑/1> | SD | | | | | | SD | | S |
| <w1:r1/↓/0> | | SD | | | | SD | | | T |
| **wCFds&wIRF** | | | | | | | | | |
| <w0:r0/0/1> | | | | SD | | | | SD | Q |
| <w0:r1/1/0> | | | SD | | SD | | | | R |
| <w1:r0/0/1> | SD | | | | | | SD | | S |
| <w1:r1/1/0> | | SD | | | | SD | | | T |

표 3. 14N 테스트 패턴을 위한 고장 딕셔너리 (파트 1)
Table 3. A fault dictionary for test pattern 14N (part 1)

| | | ① | ② | ③ | ④ | ⑤ | ⑥ | ⑦ | ⑧ | ⑨ | ⑩ | ⑪ | ⑫ | ⑬ | ⑭ | Fault group |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Stuck-At Fault (SAF)** | | | | | | | | | | | | | | | | |
| <∀/0/-> | | | | | S | D | D | | | | S | D | D | | | A |
| <∀/1/-> | | S | D | D | | | | S | D | D | | | | S | D | B |
| **Transition Fault (TF)** | | | | | | | | | | | | | | | | |
| <w↑/0/-> | | | | | S | D | D | | | | S | D | D | | | A |
| <w↓/1/-> | | (S) | (D) | (D) | | | | S | D | D | | | | S | D | B |
| **Read Destructive Fault (RDF)** | | | | | | | | | | | | | | | | |
| <r0/↑/1> | | | SD | D | | | | | SD | D | | | | | SD | B |
| <r1/↓/0> | | | | | | SD | D | | | | | SD | D | | | A |
| **Deceptive Read Destructive Fault (DRDF)** | | | | | | | | | | | | | | | | |
| <r0/↑/0> | | | S | D | | | | | S | D | | | | | S | C |
| <r1/↓/1> | | | | | | S | D | | | | | S | D | | | D |
| **Incorrect Read Fault (IRF)** | | | | | | | | | | | | | | | | |
| <r0/0/1> | | | SD | D | | | | | SD | D | | | | | SD | B |
| <r1/1/0> | | | | | | SD | D | | | | | SD | D | | | A |
| **Disturb Coupling Fault(CFds)** | | | | | | | | | | | | | | | | |
| <w0;0/↑/-> | a>v | S | D | D | | | | S | D | D | | | | | | E |
| | a<v | (S) | (D) | (D) | | | | | | | | | | S | D | F |
| <w0;1/↓/-> | a>v | | | | | | | | | | | D | D | S | | G |
| | a<v | | | | | D | D | S | | | | | | | | H |
| <r0;0/↑/-> | a>v | | | | | | | | SD | D | | | | | SD | I |
| | a<v | | SD | D | | | | | | | | | | | S | J |
| <r0;1/↓/-> | a>v | | S | | | D | D | | | | | | | | | H |
| | a<v | | | | | | | | S | | | D | D | | | G |
| <w1;0/↑/-> | a>v | | | | | | | | D | D | S | | | | | K |
| | a<v | | D | D | S | | | | | | | | | | | J |
| <w1;1/↓/-> | a>v | | | | S | D | D | | | | | | | | | H |
| | a<v | | | | | | | | | | S | D | D | | | G |
| <r1;0/↑/-> | a>v | | | | | S | | | D | D | | | | | | K |
| | a<v | | | | | | | | | | | S | | | D | L |
| <r1;1/↓/-> | a>v | | | | | | | | | | | SD | D | | | G |
| | a<v | | | | | SD | D | | | | | | | | | H |
| **State Coupling Fault(CFst)** | | | | | | | | | | | | | | | | |
| <1;1/0/-> | a>v | | | | S | D | D | | | | S | D | D | | | A |
| | a<v | | | | S | D | D | | | | S | D | D | | | A |
| <1;0/1/-> | a>v | | | | | | | S | D | D | | | | | | K |
| | a<v | | D | D | S | | | | | | | | | S | D | F |
| <0;1/0/-> | a>v | | | | S | D | D | | | | | D | D | S | | A |
| | a<v | | | | | D | D | S | | | S | D | D | | | A |
| <0;0/1/-> | a>v | S | D | D | | | | S | D | D | | | | S | D | B |
| | a<v | S | D | D | | | | S | D | D | | | | S | D | B |
| **Incorrect Read Coupling Fault(CFir)** | | | | | | | | | | | | | | | | |
| <0;r0/0/1> | a>v | | SD | SD | | | | | | | | | | | SD | F |
| | a<v | | | | | | | | SD | SD | | | | | SD | I |
| <0;r1/1/0> | a>v | | | | | | | | | | | SD | SD | | | G |
| | a<v | | | | | SD | SD | | | | | | | | | H |
| <1;r0/0/1> | a>v | | | | | | | | SD | SD | | | | | | K |
| | a<v | | SD | SD | | | | | | | | | | | | J |
| <1;r1/1/0> | a>v | | | | | SD | SD | | | | | | | | | H |
| | a<v | | | | | | | | | | | SD | SD | | | G |

였다

전체 딕셔너리는 두 파트로 나뉜다. 표 3과 표4는 이와 같은 방식으로 생성 된 고장 딕셔너리이다. 표 3은 길이 14N의 테스트 패턴이 가해졌을 때의 고장 딕셔너리(파트 1)이며 표 4는 길이 8C의 테스트 패턴이 가해졌을 때의 고장 딕셔너리(파트 2)이다. 이는 고장을 유발하고 고장이 검출되는 동작을 표시하고 동일한 동작에서 고장이 검출되는 것들을 고장 그룹으로 분류하였다. 따라서 테스트 결과를 본 딕셔너리를 통해 분석하게 되면 A부터 T까지의 고장으로 분류될 수 있다.

## V. 진단 패턴

고장 프리미티브는 부록(appendix)에서와 같이 고장 딕셔너리에 의해 20개의 그룹으로 구분된다. 그러나 좀 더 정확한 고장 종류 진단을 위해서는 추가적인 진단 패턴이 필요하다. 같은 그룹 안에 존재하는 고장 프리미티브들은 다음과 같은 특징에 의해 구분될 수 있다.

1. 고장이 하나의 포트를 통해 유발되는지 두 개의 포트를 통해 유발되는지
2. 고장이 하나의 셀과 관련되어 있는지 두 개의 셀과 관련되어 있는지
3. 고장을 유발시키는 a-cell과 v-cell의 조건 (결합 고장의 구분을 위함)

1PFs와 2PFs를 구별하기 위해서는 고장을 유발시킬 수 있는 동작을 하나의 포트를 통해 가한다. 이 때 고장이 발생하면 1PFs에 해당하고 발생하지 않으면 2PFs에 해당한다. 1PF1s와 1PF2s는 고장을 유발하고 검출 할 수 있는 동작을 고장 셀(v-cell)에만 가하여 구분할 수 있다. 이 때 고장이 발생하면 1PF1s이고 그렇지 않으면 1PF2s라고 할 수 있다. 2PF1s와 2PF2s의 경우도 마찬가지이다. 위와 같은 두 가지 과정에 의해 구분되지 않는 나머지 고장들은 모두 결합 고장들인데 이들은 a-cell과 v-cell에 가해지는 동작들과 상태들을 조절하여 여러 고장 모델들 중 일부의 고장만을 유발, 검출할 수 있는 조건을 만들어 주어 구분한다. 표 5는 결합 고장의 종류에 따른 고장을 유발, 검출할 수 있는 a-cell과 v-cell의 동작 또는 상태를 나타낸 것이다.

표 3. 각각의 결합 고장의 구별 특성
Table 3. Distinguishing features of each coupling fault.

| 고장 | 조건 | | v-cell의 값과 읽기 동작에 대한 결과 비교 (F:R) |
|---|---|---|---|
| | Aggressor cell | Victim cell | |
| CFds | 읽기 또는 쓰기동작 | 특정 상태 | 같음 |
| CFst | 특정 상태 | 특정 상태 | 같음 |
| CFir | 특정 상태 | 읽기 동작 | 다름 |
| CFdr | 특정 상태 | 읽기 동작 | 첫 번째 읽기 동작에 대해서는 같음<br>두 번째 읽기 동작에 대해서는 다름 |
| CFrd | 특정 상태 | 읽기 동작 | 같음 |
| CFtr | 특정 상태 | 쓰기 동작 | 같음 |

예를 들어 CFir과 CFrd는 a-cell이 특정 상태에 있을 때 v-cell에 읽기 동작을 수행하면 고장이 유발된다. 두 고장 종류의 테스트 결과는 동일하게 나온다. 그러나 CFrd이 존재할 때 메모리 셀의 값은 실제로 변하고 CFir이 존재할 때 메모리 셀의 값은 실제로는 변하지 않지만 변한 것처럼 나타나게 된다. 따라서 두 고장은 다음과 같은 과정에 의해 구분될 수 있다.

1. v-cell에 읽기 동작을 수행하여 고장이 유발되었음을 확인한다.
2. a-cell의 상태를 바꾸어 준다.
   (a-cell에 쓰기 동작 수행)
3. 다시 v-cell에 읽기 동작을 수행하여 고장을 재확인한다.

그림 4는 CFir<0;r1/1/0>(a<v)와 CFrd<0;r1/↓/0>(a<v)을 구분하는 과정을 그림으로 나타낸 예이다. 과정 1은 a-cell이 0의 값을 가지고 있는 상태에서 v-cell에 읽기 동작을 수행하는 것이다. 과정 1에 의해

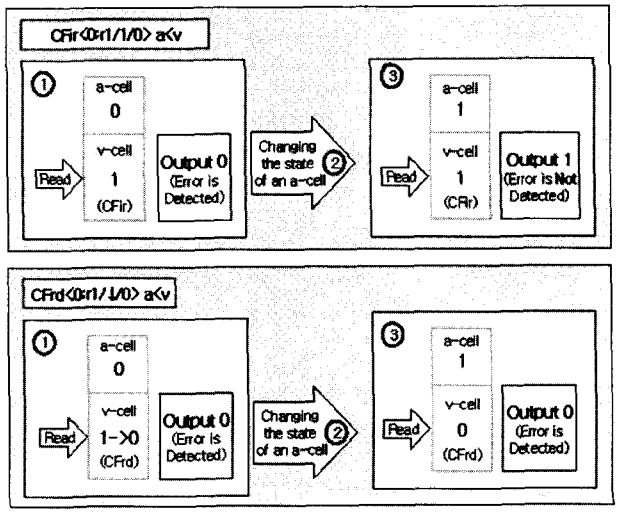



그림 4. CFir과 CFrd를 구분하는 과정의 예
Fig. 4. An example of the process which distinguishes CFir and CFrd.

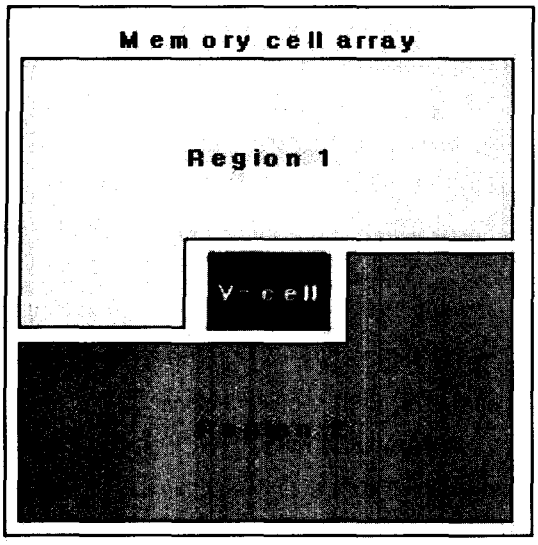



그림 5. 메모리 셀 어레이의 구분
Fig. 5. Partitioning of the memory cell array.

표 6. 고장 그룹 G
Table 6. A fault group G.

| 고장 그룹 | 고장 번호 | 고장 프리미티브 |
|---|---|---|
| G | 1 | CFds<w0;1/?/-> a>v<br>CFds<r0;1/?/-> a<v<br>CFds<w1;1/?/-> a<v<br>CFds<r1;1/?/-> a>v2 |
| | 2 | CFir<0;r1/1/0> a>v<br>CFir<1;r1/1/0> a<v3 |
| | 3 | CFrd<0;r1/?/0> a>v<br>CFrd<1;r1/?/0> a<v4 |
| | 4 | CFtr<0;w?/0/-> a<v<br>CFtr<1;w?/0/-> a>v5 |
| | 5 | wCFds&wCFds<r0;r0;1/?/-> a<v<br>wCFds&wCFds<r1;r1;1/?/-> a>v |
| | 6 | wCFrd&wRDF<0;r1;r1/ ?/0> a>v<br>wCFrd&wRDF<1;r1;r1/?/0> a<v |

두 고장은 모두 유발되어 고장이 검출된다. 이 때 CFrd를 가진 메모리 셀의 값은 실제로 변하여 0의 값을 가지고 있으며, CFir의 경우 메모리 셀의 값이 실제로는 변함없이 1을 가지고 있지만, 고장에 의해 0인 것처럼 보이게 되는 것이다. 그 후 과정 2에 의해 a-cell의 값이 바뀌게 되어 고장을 유발 할 수 있는 조건이 사라지게 된다. 그리고 과정 3에 의해 다시 읽기 동작을 수행하였을 때 CFir 고장을 가진 메모리 셀은 고장이 검출되지 않을 것이며 CFrd 고장을 가진 메모리 셀은 과정 1에서 실제로 메모리 셀의 값이 변화되었으므로 계속

표 7. 각각의 구역에 가해지는 그룹 G를 위한 진단 패턴
Table 7. The diagnostic pattern for group G which are applied at each zone.

| | 요소 1 | 요소 2 | 요소 3 | 요소4 | 요소 5 | 요소 6 | 요소 7 | 요소 8 | 요소 9 | 요소 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 구역 1 | w1 | | | w1,r1 | | | | w0 | | |
| v-cell | | w1 | r1 | | r1 | w1 | r1 | | r1 | r1 |
| 구역 2 | | | | w0,r0 | | | | w1 | | |

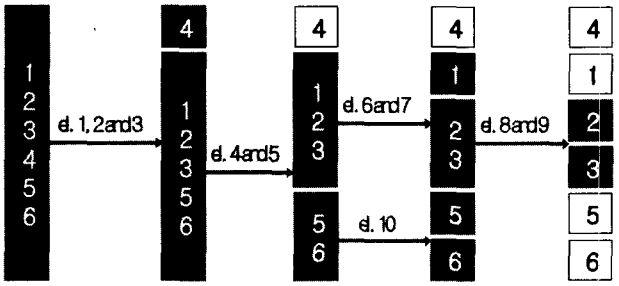



그림 6. 추가 고장 진단 패턴에 의한 그룹 G의 고장 구분 과정
Fig. 6. Fault divisions by additional diagnostic pattern for fault group G.

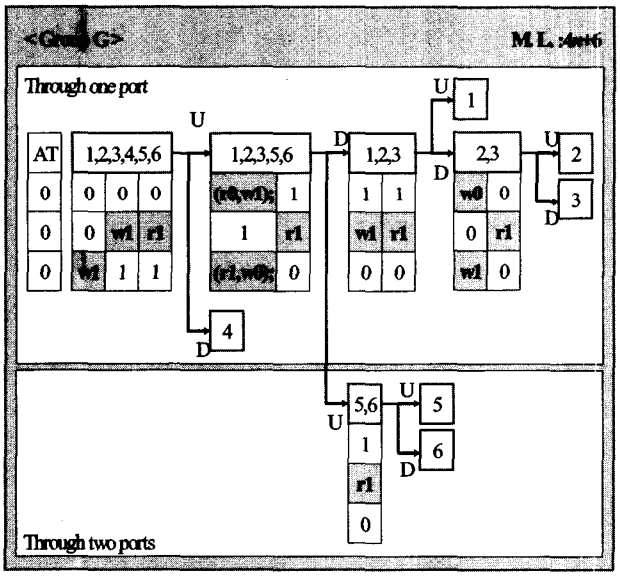



그림 7. 고장 그룹 G를 위한 추가적인 고장 진단 과정
Fig. 7. An additional diagnostic process for fault group G.

고장이 존재하는 것처럼 보이게 된다. 따라서 위의 세 동작에 의해 두 고장은 구별될 수 있다.

메모리 셀 어레이는 3 구역으로 나뉘어 진다. 그림 5에서와 같이 구역1과 v-cell 그리고 구역 2로 나뉘어 진다. 구역 1은 v-cell 보다 주소 값이 작은 셀들을 나타내며 구역 2는 v-cell 보다 주소 값이 큰 셀들을 나타낸다. 진단을 위한 추가 진단 패턴은 때대로 각 구역별로

서로 다른 동작을 수행하거나 한 두 개의 구역에만 동작을 가할 수도 있다.

예를 들어, 고장 그룹 G는 표 6에서와 같이 6개의 고장 모델 (14개의 고장 프리미티브)로 구성되어 있다. 이들은 아래의 10 개의 추가 진단 요소(diagnostic elements)에 의해 구분 될 수 있다.

[요소 1] 하나의 포트를 통해 구역 2에 w1: 고장 번호 4의 a-cell

[요소 2] 하나의 포트를 통해 v-cell에 w1: 고장 번호 4의 v-cell

[요소 3] 하나의 포트를 통해 v-cell에 r1 : **고장 번호 4 검출**

[요소 4] 하나의 포트를 통해 구역 1에 (r0, w1), 구역 2에 (r1, w0) : 고장 번호 1, 2, 3의 c-cell

[요소 5] 하나의 포트를 통해 v-cell에 r1:고장 번호 2, 3의 v-cell과 **고장 번호 1, 2, 3 검출**

[요소 6] 하나의 포트를 통해 v-cell에 w1: 고장 번호 2, 3의 v-cell

[요소 7] 하나의 포트를 통해 v-cell에 r1: **고장 번호 2, 3 검출**

[요소 8] 하나의 포트를 통해 구역 1에 w0, 구역 2에 w1: 고장 번호 2, 3의 고장 유발 조건 제거

[요소 9] 하나의 포트를 통해 v-cell에 r1: **고장 번호 3 검출**

[요소 10] 두 포트를 통해 v-cell에 r1: **고장 번호 5 검출**

표 7은 고장 그룹 G를 위해 메모리 셀에 가해지는 고장 진단 패턴을 나타낸 것이다. 각각의 열은 진단 요소를 나타내며 각각의 진단 요소마다 세 구역에 가해지는 동작들을 표시 되어 있다. 빈 칸은 아무런 동작을 가하지 않는 것을 뜻하며 모든 동작은 메모리 셀의 주소를 증가시키는 방향으로 수행 된다.

그림 6은 고장 그룹 G의 6개의 고장 모델이 위의 진단 요소에 의해 구분 되는 과정을 나타낸 것이다. 고장 모델 {1, 2, 3, 4, 5, 6}은 요소 1, 2와 3에 의해 {4}와 {1, 2, 3, 5, 6}으로 구별 되고, 고장 모델 {1, 2, 3, 5, 6}은

표 8. 단일 포트 관련 고장의 분류
Table 8. Possible diagnostic bounds.

| | 고장 모델명 | 진단 결과( 고장그룹-고장 번호) |
|---|---|---|
| 1PF1s | Stuck-At Fault (SAF)<br>Transition Fault (TF)<br>Read Destructive Fault (RDF)<br>Incorrect Read Fault (IRF)<br>No Access Fault (NAF) | A-1 B-1 |
| | Deceptive Read destructive Fault (DRDF) | C-1 D-1 |
| 1PF2s | Disturb Coupling Fault (CFds) | E-1, F-1, G-1, H-1, I-1, K-1, L-1 |
| | State Coupling Fault (CFst) | A-2, B-2, F-2, K-2 |
| | Incorrect Read Coupling Fault (CFir) | F-3, G-2, H-2, I-2, J-2, K-3 |
| | Deceptive Read Destructive Coupling Fault (CFdr) | M-1, N-1, O-1, P-1 |
| | Read Destructive Coupling Fault (CFrd) | F-4, G-3, I-3, J-3, K-4 |
| | Transition Coupling Fault (CFtr) | G-4, H-4, K-5, L-2 |
| 2PF1s | wDRDF&wDRDF | C-2, D-2 |
| | wRDF&wRDF | A-3, B-3 |
| 2PF2v | wCFds&wCFds | G-5, H-5, I-4, J-4, K-6, L-3 |
| 2PF2a | wCFdr&wDRDF | M-2, N-2, O-2, P-2, |
| | wCFrd&wRDF | F-5, G-6, H-6, I-5, J-5, K-7 |
| 2PF2av | wCFds&wRDF | Q-1, R-1, S-1, T-1 |
| | wCFds&wIRF | Q-2, R-2, S-2, T-2 |

다시 요소 4와 5에 의해 {1, 2, 3}과 {5, 6}으로 구별 된다.

그림 7은 고장 그룹 G의 추가 고장 진단 요소와 가해지는 순서, 고장 진단 과정과 최대 고장 진단 패턴의 길이 등 추가적인 고장 진단 패턴을 설명할 수 있는 모든 정보를 담고 있다. 최대 고장 진단 패턴의 길이(M.L.)는 4n+6(n은 메모리 셀의 개수)이다. 'AT'는 테스트가 수행 된 후 각각의 세 구역의 상태를 나타내는 부분이다. 1부터 6까지의 숫자는 고장 모델을 나타내는 고장 번호이며 이 번호 아래 부분에는 이 고장들을 구별하기 위한 고장 진단 요소들이 적혀있다. 세 개의 상자로 이루어 진 세로의 열 하나가 하나의 진단 요소를 나타낸다. 'Through one port'라고 적혀있는 큰 상자 안에 들어 있는 모든 동작들은 하나의 포트를 통해서 가해지게 되며, 'Through two ports'라고 적혀있는 큰 상자 안에 들어있는 동작들은 두 개의 포트를 통해서 동시에 가해지게 된다. 'U'라고 표시 되어 있는 화살표는 앞에 가해 진 진단 요소에 의해 고장이 검출되지 않는 고장 모델을 가리키며 'D'라고 표지 된 화살표는 진단 요소에 의해 고장이 검출 된 고장 모델을 가리킨다. 이외의 각각의 고장 그룹을 위한 추가적인 고장 진단 과정은 부록(appendix)에 있다.

## VI. 결 과

고려하고 있는 모든 검출 대상 고장 모델에 대해, 2PF2av를 제외한 나머지 고장에 대한 테스트와 진단을 위한 패턴의 길이는 14N의 테스트 패턴과 최대 5N+2의 추가 진단 패턴이 필요하므로 최대 19N+2가 된다. 2PF2av 고장의 경우, 테스트와 진단을 위한 패턴의 길이로, 14N의 테스트 패턴을 가하여 다른 고장이 없음을 확인한 후, 8C의 추가 테스트 패턴이 가해지게 되고, 길이 2의 진단 패턴이 더 가해지므로 14N+8C+2의 패턴 길이로, 테스트와 진단이 모두 가능하게 된다. 전체 메모리 셀의 수가 N이고, 열의 수가 최대 전체 셀 수의 절반정도라고 봤을 때, 18N+2정도의 길이라고 할 수 있으므로 전체적으로 최대 19N+2의 패턴으로 이중 포트 메모리를 위한 테스트와 세부적인 고장 종류의 진단이 가능하게 된다. SAF, TF, RDF, IRF 그리고 NAF 등은 고장 그룹 A와 B의 고장 번호 1번들로, 이들은 고장이 존재할 경우 고장에 대한 효과가 동일하게 발현되므로 어떠한 패턴을 가해주더라도 구분될 수 없다. 따라서 본 논문에서 제안하는 고장 진단 알고리듬을 사용할 경우, 이들 5종류의 고장 모델을 제외하고는 모든 고장 모델에 대해서 세부적으로 고장 종류의 구분이 가능하다. 표 8은 고장 진단 알고리듬의 고장 구분 결과를 나타내는 표이다. 각각의 고장 모델에 대해 해당 고장 그룹과 고장 번호를 함께 표기하였다.

표 9. 각각의 알고리듬의 성능
Table 9. Performance of each algorithm.

| 알고리듬 / 패턴 길이 / 고장 | [7] | [8] | [9] | [10] | [11] | 제안된 알고리듬 |
|---|---|---|---|---|---|---|
| | 17N | 9N | 17N | 12N | Test pattern+ $19N+6M^2+3M+15$ | **19N+2** |
| 1PF1s | D | D | D | D | D | **D** |
| 1PF2s | P | P | P | P | P | **D** |
| 2PF1s | U | U | U | U | U | **D** |
| 2PF2s | U | U | U | U | P (only 2PF2av) | **D** |

D: 구별 됨, P: 일부분만 구별 됨, U: 구별되지 않음

표 9는 기존의 고장 진단 알고리듬과 본 논문에서 제안하는 고장 진단 알고리듬의 성능을 비교한 것이다. 표 9에서는 각 알고리듬별로 구분 가능한 고장과 고장 진단을 위한 알고리듬의 길이 등으로 성능을 표기하였다. [7]-[10]의 진단 알고리듬은 단일 포트 메모리만을 고려하고 있다. 이들은 각각 17N, 9N, 17N 그리고 12N의 테스트 패턴만을 가지고 그 응답을 비교하여 고장의 종류를 구분하는 형태를 취하고 있으며 이는 고착 고장과 결합 고장 등 일부의 고장만을 고려하고 있다. 이에 비해 본 알고리듬은 기존의 단일 포트 메모리를 위한 테스트 패턴과 비슷한 형태의 테스트 패턴을 가지면서도 이중포트 메모리에 적용 가능하도록 고안되었으며, 다양한 고장들을 최대한 자세하게 구분할 수 있는 알고리듬이다. 포트의 수가 늘어나고, 고려하는 고장의 종류가 늘어날수록 고장의 종류를 구별하는 과정은 더욱 복잡해지고, 힘들어지기 때문에 단일 포트 메모리를 위한 고장 진단 알고리듬과 이중 포트 메모리를 위한 고장 진단의 직접적이고 명확한 비교는 불가능할 분만 아니라 아무런 의미가 없다. 그러나 [11]의 진단 알고리듬은 이중 포트 메모리를 위한 고장 진단 알고리듬이나, 테

스트를 제외한 진단 알고리듬만으로, $19N+15+3M+6M^2$(N은 셀의 수, M은 행의 수)이므로 여기에 테스트 패턴이 추가될 경우 테스트와 진단을 모두 수행할 때의 길이는 더욱 늘어날 것이며, 본 논문에서 제안하는 알고리듬에 비해 진단 가능한 고장 종류가 매우 적다는 점을 생각할 때, 본 논문에서 제안하는 알고리듬이 패턴의 길이나, 성능 면에서 더욱 효과적인 알고리듬이라고 할 수 있다. 이 외에는 이중 포트 메모리를 위한 고장 종류 분석에 관한 연구가 없어 성능의 비교 분석이 불가능하였다.

## Ⅷ 결 론

본 논문에서는 이중 포트 메모리를 효과적으로 테스트하여, 진단과정을 통해 고장의 종류를 자세하게 분류할 수 있는 진단 알고리듬을 제안하였다. 이는 14N의 길이를 갖는 테스트 패턴으로 테스트를 수행한 후, 그 결과를 바탕으로 만든 딕셔너리를 이용하여 고장을 크게 분류한 뒤, 분류된 고장 그룹에 따라 적절한 진단 패턴을 가하여 고장의 종류를 세부적으로 분류하였다. 본 논문에서는 테스트를 통해 얻을 수 있는 정보인 딕셔너리를 이용하여 진단 패턴의 길이를 최소화 하였으며 이중 포트 메모리에서 발생할 수 있는 모든 고장 중, 기능 테스트를 통해 검출 할 수 있는 모든 고장을 고려하였다는 점에서 매우 효과적인 알고리듬이라고 할 수 있다.

## 참 고 문 헌


[1] S. J. Wang, C. J. Wei, "Efficient built-in self-test algorithm for memory", Proc. Of Asian Test Symposium, pp. 66-70, 2000.

[2] C. T. Huang, J. R.Huang, "A programmable built-in self-test core for embedded memories," Proc. Of Asia and South Pacific Design Automation Conference, pp. 11-12, 2000.

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

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

[5] 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, pp. 366 -374, 2001.

[6] M. F. Chang, W. K. Fuchs and J. H. Patel, "Diagnosis and repair of memory with coupling faults," IEEE Trans. on computers, pp. 493-500, 1989.

[7] T. J. Bergfeld, D. Niggemeyer and E. M. Rudnick, "Diagnostic testing of embedded memories using BIST,"Proc. of IEEE Design Automation and Test in Europe Conference and Exhibition, pp. 305-309, 2000.

[8] C. W. Wang, C. F. Wu, J. F. Li, C. W. Wu, T. Teng , K. Chiu and H. P. Lin "A Built-In Self-Test and Self-Diagnosis Scheme for Embedded SRAM," Proc. of the NinthAsian Test Symposium, pp. 45 -50, 2000.

[9] J. F. Li, K. L. Cheng, C. T. Huang and C. W. Wu, "March-based RAM diagnosis algorithms for stuck-at and coupling faults," Proc. of International Test Conference, pp.758 -767, 2001.

[10] V.A. Vardanian, Y. Zorian, " A March-based fault location algorithm for static random access memories," Proc, of IEEE International Workshop on Memory Technology, Design and Testing, pp. 10-12, 2002.

[11] 박한원, 강성호, "이중 포트 메모리를 위한 고장 진단 알고리듬," 전기학회 논문지 50권 4호, pp. 192-200, 2001.

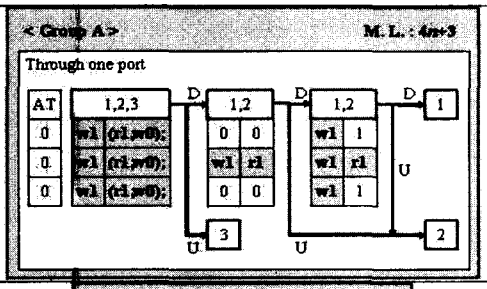
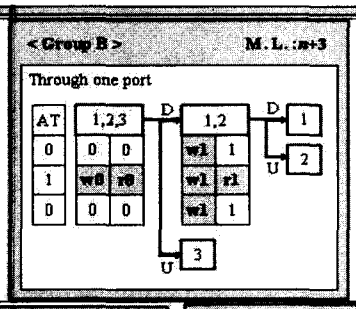
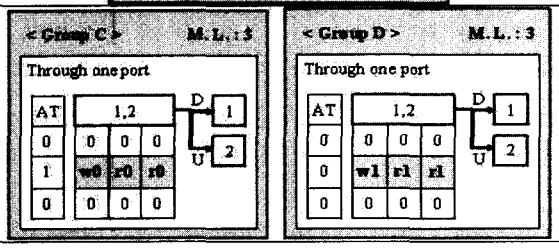
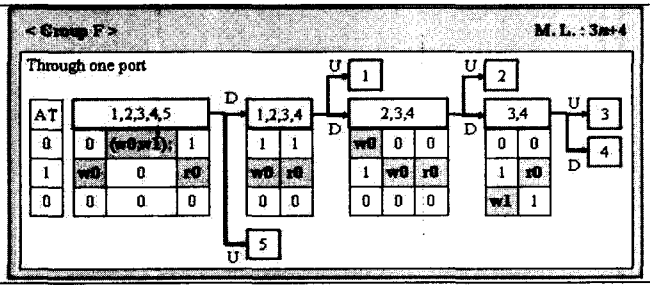
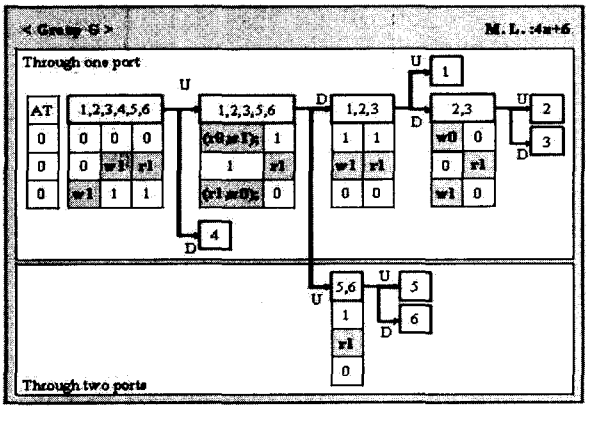
[12] S. Hamdioui,A. J. van de Goor, "Thorough testing of any multiport memory with linear tests," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, pp. 217 -231, 2002.

[13] 김지혜, 배상민, 송동섭, 강성호,"March C-에 바탕을 둔 이중 포트 메모리를 위한 테스트 알고리듬," 제 3회 한국 테스트 학술대회, pp. 140-144, 2002.

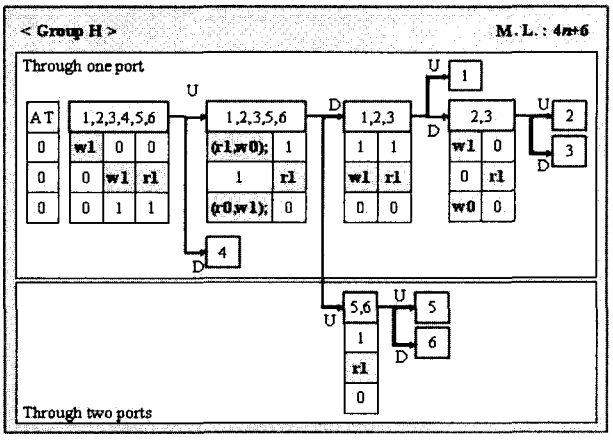
## 부록(Appendix)

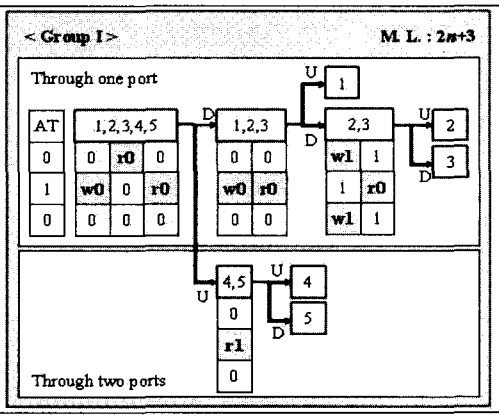
각각의 고장 그룹에 대한 고장 진단 과정

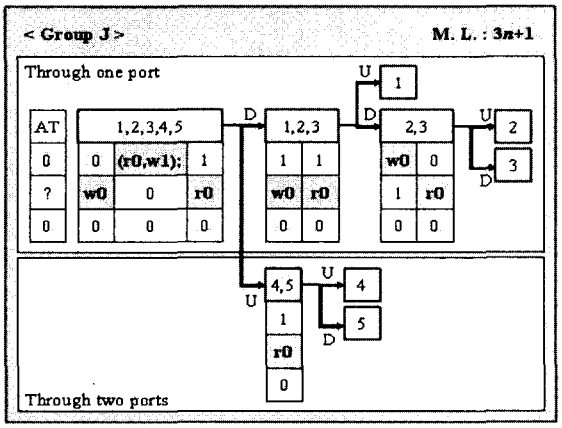
Diagnosis process of each fault group

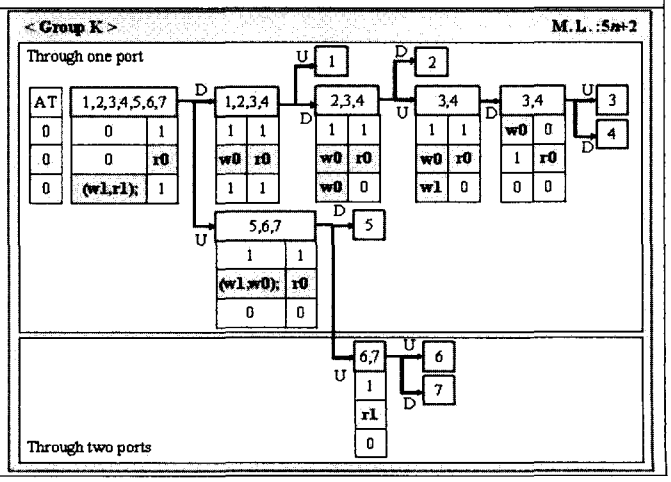
| Fault Group | Fault Number | Fault primitive | Diagnosis process |
|---|---|---|---|
| A | 1 | SAF<∀/0/-><br>TF<w↑/0/-><br>RDF<r1/↓/0><br>IRF<r1/1/0> |  |
| | 2 | CFst<1;1/0/-> a>v<br>CFst<1;1/0/-> a<v<br>CFst<0;1/0/-> a>v<br>CFst<0;1/0/-> a<v | |
| | 3 | wRDF&wRDF<r1:r1/↓/0> | |
| B | 1 | SAF<∀/1/-><br>TF<w↓/1/-><br>RDF<r0/↑/1><br>IRF<r0/0/1> |  |
| | 2 | CFst<0;0/1/-> a>v<br>CFst<0;0/1/-> a<v | |
| | 3 | wRDF&wRDF<r0:r0/↑/1> | |
| C | 1 | DRDF<r0/↑/0> |  |
| | 2 | wDRDF&wDRDF<r0:r0/↑/0> | |
| D | 1 | DRDF<r1/↓/1> | |
| | 2 | wDRDF&wDRDF<r1:r1/↓/0> | |
| E | 1 | CFds<w0;0/↑/-> a>v | Unnecessary |
| F | 1 | CFds<w0;0/↑/-> a<v |  |
| | 2 | CFst<1;0/1/-> a<v | |
| | 3 | CFir<0;r0/0/1> a>v | |
| | 4 | CFrd<0;r0/↑/1> a>v | |
| | 5 | wCFrd&wRDF<0;r0:r0/↑/1> a>v | |
| G | 1 | CFds<w0;1/↓/-> a>v<br>CFds<r0;1/↓/-> a<v<br>CFds<w1;1/↓/-> a<v<br>CFds<r1;1/↓/-> a>v |  |
| | 2 | CFir<0;r1/1/0> a>v<br>CFir<1;r1/1/0> a<v | |
| | 3 | CFrd<0;r1/↓/0> a>v<br>CFrd<1;r1/↓/0> a<v | |
| | 4 | CFtr<0;w↑/0/-> a<v<br>CFtr<1;w↑/0/-> a>v | |
| | 5 | wCFds&wCFds<r0:r0;1/↓/-> a<v<br>wCFds&wCFds<r1:r1;1/↓/-> a>v | |
| | 6 | wCFrd&wRDF<0;r1:r1/↓/0> a>v<br>wCFrd&wRDF<1;r1:r1/↓/0> a<v | |

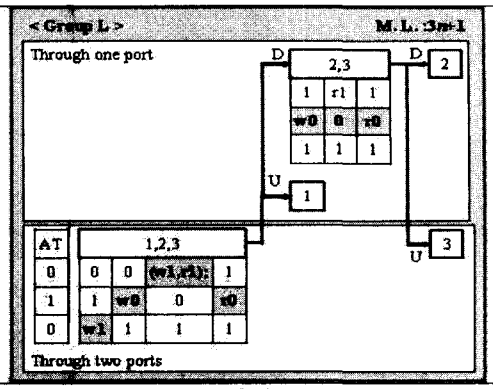
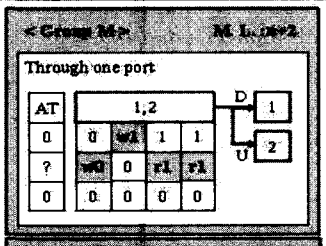
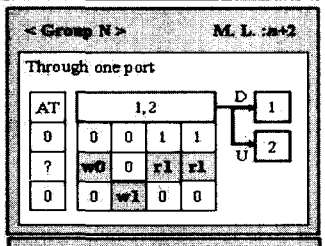
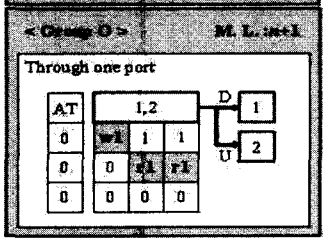
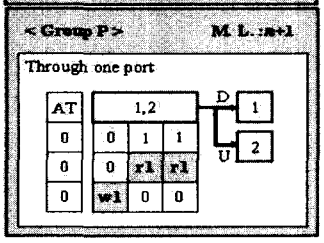
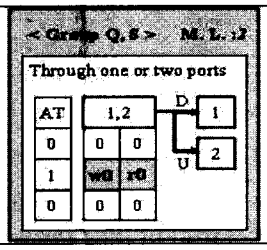
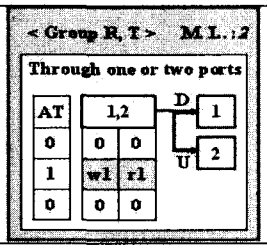
| | | |
|---|---|---|
| H | 1 | CFds<w0;1/↓/-> a<v<br>CFds<r0;1/↓/-> a>v<br>CFds<w1;1/↓/-> a>v<br>CFds<r1;1/↓/-> a<v |
| | 2 | CFir<0;r1/1/0> a<v<br>CFir<1;r1/1/0> a>v |
| | 3 | CFrd<0;r1/↓/0> a<v<br>CFrd<1;r1/↓/0> a>v |
| | 4 | CFtr<0;w↑/0/-> a>v<br>CFtr<1;w↑/0/-> a<v |
| | 5 | wCFds&wCFds<r0:r0;1/↓/-> a>v<br>wCFds&wCFds<r1:r1;1/↓/-> a<v |
| | 6 | wCFrd&wRDF<0;r1:r1/↓/0> a<v<br>wCFrd&wRDF<1;r1:r1/↓/0> a>v |
| I | 1 | CFds<r0;0/↑/-> a>v |
| | 2 | CFir<0;r0/0/1> a<v |
| | 3 | CFrd<0;r0/↑/1> a<v |
| | 4 | wCFds&wCFds<r0:r0;0/↑/-> a>v |
| | 5 | wCFrd&wRDF<0;r0:r0/↑/1> a<v |
| J | 1 | CFds<r0;0/↑/-> a<v<br>CFds<w1;0/↑/-> a<v |
| | 2 | CFir<1;r0/0/1> a<v |
| | 3 | CFrd<1;r0/↑/1> a<v |
| | 4 | wCFds&wCFds<r0:r0;0/↑/-> a<v |
| | 5 | wCFrd&wRDF<1;r0:r0/↑/1> a<v |
| K | 1 | CFds<w1;0/↑/-> a>v<br>CFds<r1;0/↑/-> a>v |
| | 2 | CFst<1;0/1/-> a>v |
| | 3 | CFir<1;r0/0/1> a>v |
| | 4 | CFrd<1;r0/↑/1> a>v |
| | 5 | CFtr<0;w↓/1/-> a<v<br>CFtr<1;w↓/1/-> a>v |
| | 6 | wCFds&wCFds<r1:r1;0/↑/-> a>v |
| | 7 | wCFrd&wRDF<1;r0:r0/↑/1> a>v |



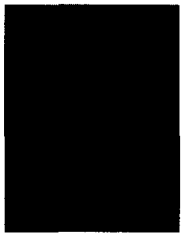



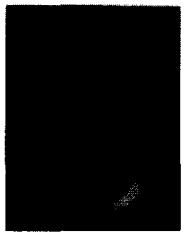



| | | | |
|---|---|---|---|
| L | 1 | CFtr<0;w↓/1/-> a>v<br>CFtr<1;w↓/1/-> a<v |  |
| | 2 | CFds<r1;0/↑/-> a<v | |
| | 3 | wCFds&wCFds<r1:r1;0/↑/-> a<v | |
| M | 1 | CFdr<0;r0/↑/0> a>v<br>CFdr<1;r0/↑/0> a<v |  |
| | 2 | wCFdr&wDRDF<0;r0:r0/↑/0> a>v<br>wCFdr&wDRDF<1;r0:r0/↑/0> a<v | |
| N | 1 | CFdr<0;r0/↑/0> a<v<br>CFdr<1;r0/↑/0> a>v |  |
| | 2 | wCFdr&wDRDF<0;r0:r0/↑/0> a<v<br>wCFdr&wDRDF<1;r0:r0/↑/0> a>v | |
| O | 1 | CFdr<0;r1/↓/1> a>v<br>CFdr<1;r1/↓/1> a<v |  |
| | 2 | wCFdr&wDRDF<0;r1:r1/↓/1> a>v<br>wCFdr&wDRDF<1;r1:r1/↓/1> a<v | |
| P | 1 | CFdr<0;r1/↓/1> a<v<br>CFdr<1;r1/↓/1> a>v |  |
| | 2 | wCFdr&wDRDF<0;r1:r1/↓/1> a<v<br>wCFdr&wDRDF<1;r1:r1/↓/1> a>v | |
| Q | 1 | wCFds&wRDF<w0:r0/↑/1> |  |
| | 2 | wCFds&wIRF<w0:r0/0/1> | |
| R | 1 | wCFds&wRDF<w0:r1/↓/0> |  |
| | 2 | wCFds&wIRF<w0:r1/1/0> | |
| S | 1 | wCFds&wRDF<w1:r0/↑/1> | |
| | 2 | wCFds&wIRF<w1:r0/0/1> | |
| T | 1 | wCFds&wRDF<w1:r1/↓/0> | |
| | 2 | wCFds&wIRF<w1:r1/1/0> | |

## 저 자 소 개

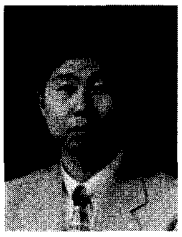



김 지 혜(정회원)
2002년 연세대학교 기계전자공학부 전기전자 전공 학사 졸업
2004년 연세대학교 전기전자공학과 석사 졸업
2004년 현재 삼성전자 반도체 총괄 System LSI 사업부
<주관심분야 : DFT, CAD, VLSI, Testing>



김 상 욱(정회원)
2001년 연세대학교 전기공학과 학사 졸업.
2003년 연세대학교 전기전자공학과 석사 졸업
현재 LG전자 시스템 IC 사업부

김 홍 식(정회원)
1997년 연세대학교 전기공학과 학사 졸업.
1999년 연세대학교 전기전자공학과 석사 졸업.
2004년 현재 연세대학교 전기전자공학과 박사 과정.
<주관심분야 : CAD 및 VLSI>



강 성 호(정회원)
1986년 서울대학교 공대 제어계측공학과 학사 졸업
1988년 The University of Texas at Austin 전기 및 컴퓨터공학과 석사 졸업
1992년 The University of Texas at Austin 전기 및 컴퓨터공학과 박사 졸업
미국 Schlumberger 연구원, Motorola 선임 연구원
현재 연세대학교 전기전자공학과 부교수
<주관심분야 : SoC 설계 및 SoC 테스트 >