KIPS Tr. Comp. and Comm. Sys.
Vol.3, No.5 pp.129~140 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2014.3.5.129

Built-In Self Repair for Embedded NAND-Type Flash Memory

Tae Hwan Kim† · Hoon Chang††

ABSTRACT

BIST(Built-in self test) is to detect various faults of the existing memory and BIRA(Built-in redundancy analysis) is to repair detected faults by allotting spare. Also, BISR(Built-in self repair) which integrates BIST with BIRA, can enhance the whole memory's yield. However, the previous methods were suggested for RAM and are difficult to diagnose disturbance that is NAND-type flash memory's intrinsic fault when used for the NAND-type flash memory with different characteristics from RAM's memory structure. Therefore, this paper suggests a BISD(Built-in self diagnosis) to detect disturbance occurring in the NAND-type flash memory and to diagnose the location of fault, and BISR to repair faulty blocks.

Keywords : NAND-type Flash Memory, BISD, BISR, Diagnosis Algorithm, Redundancy Analysis

임베디드 NAND-형 플래시 메모리를 위한 Built-In Self Repair

김 태 환† · 장 훈††

요 약

기존의 메모리에서 발생하는 다양한 고장들을 검출하기 위한 기법으로 BIST(Built-in self test)가 있고 고장이 검출되면 Spare를 할당하여 수리하는 BIRA(Built-in redundancy analysis)가 있다. 그리고 BIST와 BIRA를 통합한 형태인 BISR(Built-in self repair)를 통해 전체 메모리의 수율을 증가시킬 수 있다. 그러나 이전에 제안된 기법들은 RAM을 위해 제안된 기법으로 RAM의 메모리 구조와 특성이 다른 NAND-형 플래시 메모리에 사용하기에는 NAND-형 플래시 메모리의 고유 고장인 Disturbance를 진단하기 어렵다. 따라서 본 논문에서는 NAND-형 플래시 메모리에서 발생하는 Disturbance 고장을 검출하고 고장의 위치도 진단할 있는 BISD(Built-in self diagnosis)와 고장 블록을 수리할 수 있는 BISR을 제안한다.

키워드 : NAND-형 플래시 메모리, BISD, BISR, 진단 알고리즘, 중복 분석

1. 서 론

최근 스마트폰, 태블릿 PC, 울트라북 등을 사용하는 사용자가 증가하게 되면서 휴대 가능한 모바일 장치의 수요가 급속도로 증가하고 있다. 모바일 장치는 적은 전력을 소모하면서 고용량과 빠른 속도 낮은 비용의 메모리가 필요하다. 이러한 조건을 충족하기 위한 메모리로 플래시 메모리가 적합하다. 플래시 메모리는 비휘발성, 낮은 비용, 적은 전력소모량과 높은 신뢰도 등의 특징으로 모바일 장치의 메모리로 증가하고 있다. 그리고 플래시 메모리는 셀 배열의 구조에 따라 NOR-형 플래시 메모리와 NAND-형 플래시 메모리로 구분되어진다[1]. NOR-형 플래시 메모리는 셀 단위 접근이 가능하며 속도가 빠른 장점이 있지만 비용이 높기 때문에 모바일 장치의 메모리로 사용하기에는 부적합하다. NAND-형 플래시 메모리는 NOR-형 플래시 메모리에 비해 Read/Write 속도는 느리지만 Erase 속도가 빠르고 면적당 집적도가 높아서 대용량이 가능하기 때문에 NOR-형 플래시 메모리보다 모바일 장치의 메모리로 사용이 용이하다. 또한 최근에는 HDD보다 고속인 SSD(Solid State Disk)의 수요가 증가하면서 NAND-형 플래시 메모리의 수요도 함께 증가하고 있다. 따라서 NAND-형 플래시 메모리를 위한 테스트 기술과 연구는 점점 중요해지고 있다.

새로운 시스템이 개발될수록 메모리는 전체 시스템의 많은 영역을 차지하게 되는데 이는 전체 시스템 신뢰도에도 많은 영향을 주게 된다. 또한 NAND-형 플래시 메모리는 고용량, 셀의 밀집도가 높기 때문에 다양한 형태의 고장이

※ 본 논문은 2012년도 정부(교육학과 기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2011-0010065).
† 준 회 원 : 숭실대학교 컴퓨터학과 석사
†† 정 회 원 : 숭실대학교 컴퓨터학부 교수
Manuscript Received : January 17, 2014
First Revision : March 19, 2014
Accepted : April 9, 2014
* Corresponding Author : Hoon Chang(hoon@ssu.ac.kr)

발생할 수 있고 이러한 고장을 검출하거나 수리할 수 있는 방법이 필요하다.

전체 시스템의 신뢰도를 향상시키기 위한 다양한 기법들이 이전에 제안되어 왔다. 먼저 메모리에서 발생하는 다양한 고장들을 검출하고 고장 위치를 확인하기 위한 BIST(Built-In Self Test)와 BISD(Built-In Self Diagnosis)가 있으며[2] 이러한 고장 정보를 통해서 메인 메모리에 Redundancy를 할당하여 메모리를 수리하기 위한 BIRA(Built-In Redundancy Analysis)가 있다. 이러한 BIST/BISD모듈과 BIRA모듈을 통합하여 메모리 테스트와 메모리 수리를 수행할 수 있는 형태인 BISR(Built-In Self Repair)를 통해 전체 메모리의 수율을 증가시킬 수 있다. 그러나 이전에 제안된 이러한 기법들은 대부분 SRAM이나 DRAM을 위해 연구되어 왔고 이러한 기법들은 메모리의 구조가 다른 NAND-형 플래시 메모리에 실제로 사용하기에는 한계가 있다. 그 이유는 셀 단위로 Random-Access가 가능한 RAM과는 달리 NAND-형 플래시 메모리는 Page단위로 동작을 수행하고 발생하는 고장의 종류도 다르기 때문에 기존의 기법으로는 NAND-형 플래시 메모리에서 발생하는 모든 고장을 테스트/진단하는데 어려움이 있다.

본 논문에서는 NAND-형 플래시 메모리의 수율 향상을 위해 BISR의 알고리즘과 구조를 제안한다. 그리고 본 논문은 고장 검출의 확인만을 수행하는 BIST[11]보다 고장 검출뿐만 아니라 고장의 종류와 위치도 확인할 수 있는 BISD를 사용한다. 그리고 이미 제안된 RA알고리즘을 통해 BIRA를 사용하고 제안하는 BISD와 통합해서 NAND형 플래시 메모리의 전체 수율 향상을 위한 BISR 구조를 제안한다.

먼저 2장에서 플래시 메모리의 일반적인 구조를 소개하고 3장에서는 NAND-형 플래시 메모리 고장 모델을 설명한다. 그리고 4장에서는 제안하는 BISD를 위한 Diagnosis 알고리즘과 BISD구조를 설명하고 5장에서는 이미 제안된 RA 알고리즘과 고장 수리를 위한 Spare 할당 BIRA구조를 설명한다. 마지막으로 6장에서 제안하는 BISD모듈과 BIRA 모듈을 결합해 동작하는 BISR에 대해 설명하고 7장을 통해 다른 기존의 연구와 비교해 제안하는 알고리즘과 구조를 시뮬레이션을 통해 증명하고 8장의 결론 순서로 논문이 진행된다.

2. 플래시 메모리의 구조

일반적인 플래시 메모리의 구조는 셀 배열, Row 디코더, Column 디코더, 감지 증폭기, HV(High Voltage) 컨트롤 회로를 포함하고 있다[3].

Fig. 1은 플래시 메모리의 구조를 보여준다.

전체 셀 배열은 셀의 전하를 포함하는 많은 서브 배열로 나누어져 있는데 서브 배열을 Block이라고 한다. Row 디코더와 Column 디코더는 주소에 따라 셀 배열에 접근하고 감지 증폭기에서 셀 배열의 값들을 확인할 수 있다.

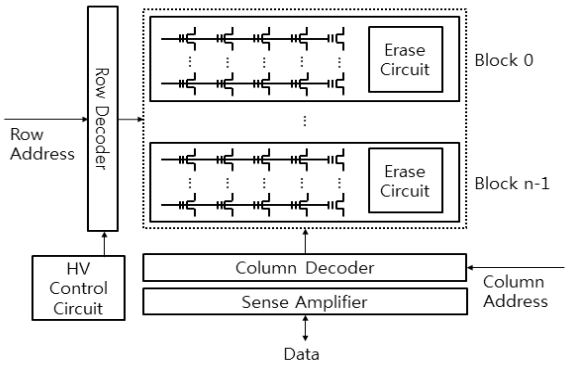


Fig. 1. Flash Memory Architecture

각 블록은 Row 주소로 주소 지정을 할 수 있다. 예를 들면, 행 주소의 MSB는 블록의 위치를 나타내고 LSB는 블록의 접근 행을 위해 사용되는데 만약 사용자가 Block 0의 n번째 행의 특정 셀을 Read하고 싶은 경우에는 MSB의 값을 0으로 할당하고 LSB의 값은 n으로 할당하면 된다.

Erase동작은 각 서브 배열에 내장되어 있는 Erase 회로를 통해 수행이 되며 Erase는 Block단위로 수행이 된다. Program 동작은 셀에 논리 값 0을 Write하게 되고 Erase동작은 셀에 논리 값 1을 Write한다.

Program과 Erase동작을 수행하기 위해서는 HV가 필요하므로 HV 컨트롤 회로는 플래시 메모리 셀에서 Program이나 Read 동작에 사용되는 전압들을 컨트롤하기 위해 존재한다. 플래시 메모리는 셀 배열의 구조에 따라 NOR-형 플래시 메모리와 NAND-형 플래시 메모리로 나누어지게 되는데 먼저 Fig. 2는 NOR-형 플래시 메모리의 구조를 보여준다.

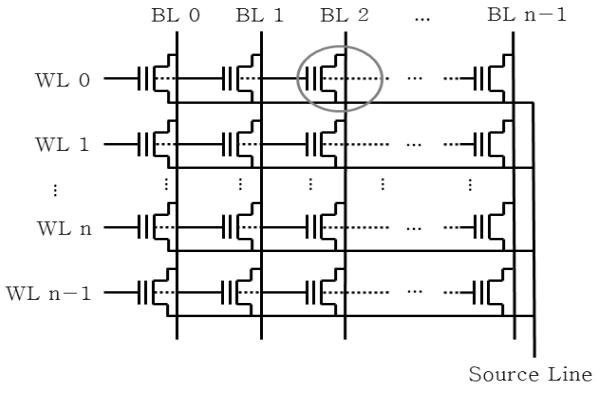


Fig. 2. NOR-Type Flash Memory Architecture

Fig. 2는 일반적인 플래시 메모리 구조에 포함되어 있는 Row 디코더, Column 디코더, 감지 증폭기 등의 회로를 제외한 NOR-형 플래시 메모리의 셀 배열만을 보여주고 있다. Fig. 2는 BL과 WL이 각각 Bit-Line과 Word-Line을 포함하는 한 블록의 NOR-형 플래시 메모리의 일반적인 셀의 배열을 보여주고 있다. 이러한 셀의 배열로 인해 NOR-형 플래시 메모리의 액세스 방법은 랜덤 액세스이다. 즉, 사용자

는 임의로 메모리 셀을 Program동작을 하거나 Read동작을 수행 할 수 있다.

예를 들어, 그림에 표시되어 있는 셀 (0,2)에 Program동작을 수행하기 위해서는 먼저 WL0에 HV(12V)를 인가하고 BL2에는 낮은 전압(6V)이 인가되어야 한다.

NOR형 플래시 메모리의 주소 유닛은 대부분 Byte이기 때문에 NOR-형 플래시 메모리는 랜덤 바이트 액세스할 수 있다. 따라서 NOR-형 플래시 메모리는 사용자 관점에서 볼 때 Erase동작을 제외하고 SRAM이나 DRAM에 액세스하는 것과 유사하다.

Fig. 3은 NAND-형 플래시 메모리의 셀 배열의 구조를 보여주고 있다[4].

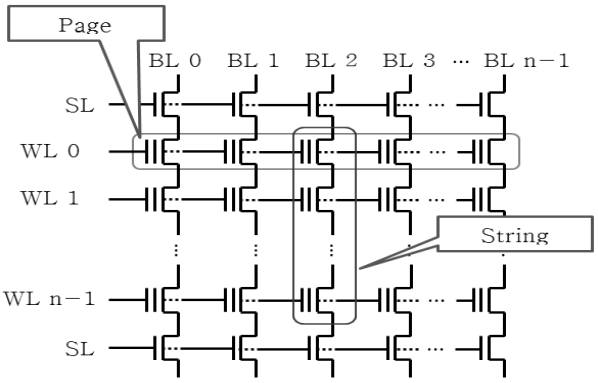


Fig. 3. NAND-Type Flash Memory Architecture

Fig. 3에서 WL은 Word-Line을 나타내고 BL은 Bit-Line을 나타내고 있다.

Word-Line을 Page로 표현하고 Bit-Line을 String으로 표현할 수 있다. 그리고 SL은 Select-Line으로 NAND형 플래시 메모리의 Page를 위한 Row를 표시하는 데 사용한다.

Fig. 3에서 여러 셀들이 두 개의 Select-Line의 셀 사이에 직렬로 연결되어 있는 것을 확인할 수 있다. 그리고 직렬 셀은 일반적으로 32개 또는 64개의 셀을 갖는다. 이것은 결국 NAND-형 플래시 메모리의 블록들이 같은 수의 셀로 구성되어 있다는 것을 확인할 수 있다. 따라서 본 논문에서는 Fig. 3에 표시되어 있는 구조를 NAND-형 플래시 메모리의 기본 단위로 BNB(Base NAND Block)로 참조한다.

또한 WL의 셀에 접근하기 위해서는 SL의 셀이 활성화되어야 하며 본 논문에서는 이런 형태의 NAND-형 플래시 메모리의 WL을 수리하기 위해 SRAM과 DRAM에서 사용하는 Redundancy 구조와는 다른 형태의 새로운 Redundancy 구조가 필요하다.

3. NAND-형 플래시 메모리 고장 모델

기본적으로 NAND-형 플래시 메모리에서 발생할 수 있는 고장 모델을 설명한다. NAND-형 플래시 메모리에서 발생할 수 있는 고장으로 크게 두 가지로 구분할 수 있다. 먼저 기존의 RAM에서 발생했던 RAM-형 고장으로 SAF(Stuck-At Fault), TF(Transition Fault), AF(Address Decoder Fault), CF(Coupling Fault), SOF(Stuck-Open Fault)와 같은 고장들[5]과 플래시 메모리의 고유 고장인 Disturbance 고장[6]이 발생할 수 있다. 각각의 고장 모델의 설명은 다음과 같다.

3.1 SAF(Stuck-At Fault)

SAF(Stuck-At Fault) 고장은 셀의 값이 고착되어 “0” 또는 “1”로 셀의 값이 다른 값으로 전환되지 않는 고장을 말한다. 예를 들어 셀의 값이 “0”으로 고정되어 있으면 SAF0으로 표현하고 셀의 값이 “1”로 고정되어 있으면 SAF1로 표현한다. SAF 고장을 검출하기 위해서는 메모리 셀에 “0”과 “1”의 값을 Program하고 Read하는 동작이 필요하다.

3.2 TF(Transition Fault)

TF(Transition Fault) 고장은 셀의 상태전환 고장으로 셀의 값이 다른 값으로 전환이 되지 않는 것을 말한다. 예를 들어 메모리 셀의 값이 “0”으로는 Program이 되어 전환이 되지만 반대로 값이 “1”로 Erase가 전환이 되지 않는 고장을 말한다. 이 고장을 검출하기 위해서는 메모리 셀의 값을 “0”에서 “1”로 Erase 동작을 수행하거나 “1”에서 “0”으로 Program 동작을 수행하면서 해당 셀을 Read하는 동작이 필요하다.

3.3 AF(Address Decoder Fault)

AF(Address Decoder Fault) 고장은 어드레스 디코더 부분의 고장을 말한다. AF는 물리적인 고장이지만 메모리를 테스트하는 과정에서 메모리 셀의 고장으로 확인이 가능하다. AF는 주소와 셀이 연결되지 않는 4가지 상태가 있고 이러한 4가지 상태가 서로 결합하여 발생하게 된다.

3.4 CF(Coupling Fault)

CF(Coupling Fault) 고장은 메모리의 직접도가 높아지면서 메모리 셀들의 간격이 좁아져서 발생하게 되는 고장이다.

CF는 기본적으로 Inversion CF와 Idempotent CF가 있다. Inversion CF는 두 개의 셀 A(희생자 셀)와 B(공격자 셀)에서 발생한다. Inversion CF 고장은 Program 동작에 의해 셀 B의 값이 “0”으로 Write될 때, 셀 A의 원래 값이 반전(“1”→“0”)된다. Idempotent CF는 Inversion CF와 비슷하게 Program 동작에 의해 셀 B의 값이 “0”으로 Write될 때, 셀 A의 값이 “0”이나 “1”로 고정된다.

3.5 SOF(Stuck-Open Fault)

SOF(Stuck-open Fault) 고장은 Bit-Line의 물리적인 고장에 의해서 셀에 접근할 수 없는 고장이다. SOF고장도 AF 고장과 비슷하게 물리적인 고장이지만 메모리 테스트를 통해 메모리 셀의 고장으로 감지가 가능하다. SOF고장이 발생하게 되면 셀의 값은 감지 증폭기(Sense Amplifier)에 걸려 있는 이전에 Read된 값이 지속적으로 확인된다.

3.6 Disturbance Fault

Fig. 4는 Disturbance Fault 고장이 발생했을 때의 NAND-형 플래시 메모리 셀의 배열을 보여준다.

먼저 Disturbance Fault 고장은 GD(Gate Disturbance), DD(Drain Disturbance), OE(Over Erase), 그리고 RD(Read Disturbance)로 4가지의 Disturbance가 있다.

GD는 동일한 Word-Line의 어떠한 셀에 간섭으로 인접한 다른 셀이 Erase(셀 값이 "1"로 전환)되거나 Program(셀 값이 "0"으로 전환)이 되는 고장이며 Fig. 4에서 보이는 것처럼 WED(Word-Line Erase Disturbance)와 WPD(Word-Line Program Disturbance)가 있다.

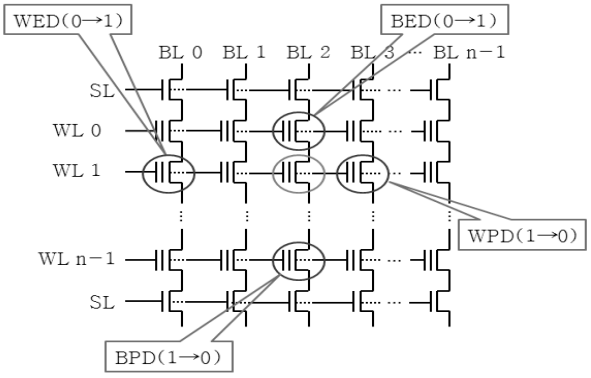


Fig. 4. Disturbance Fault

DD는 동일 Bit-Line상의 어떠한 셀에 간섭으로 인접한 다른 셀이 Erase되거나 Program되는 고장으로 Fig. 4에서 보이는 것처럼 BED(Bit-Line Erase Disturbance)와 BPD(Bit-Line Program Disturbance)가 있다. 그 밖에 OE는 원하지 않게 넓은 범위에 Erase되는 현상이며 RD는 Read 동작으로 인해 원하지 않은 잘못된 Write 동작이 발생하는 고장이다. OE는 블록단위로 진단을 수행하기 때문에 확인이 가능하며 RD는 Read 동작 이후에 똑같은 Read 동작을 한 번 더 수행하는 것으로 간단하게 검출이 가능하다.

이러한 Disturbance Fault 고장은 플래시 메모리에서 발생하는 고유 고장이기 때문에 수율 향상을 위해 필수적으로 검출되어야 하는 고장이다.

4. 제안하는 Diagnosis 알고리즘과 BISD 구조

현재 플래시 메모리의 고장 유무를 테스트하기 위해서 가장 많이 사용되는 알고리즘은 March 테스트 알고리즘이다[7].

March 테스트 알고리즘은 각각의 Elements의 조합에 따라 다양한 고장 검출이 가능하다. 기존의 SRAM과 DRAM 등 RAM형 메모리 테스트에 많이 사용되어 왔다. 하지만 NAND형 플래시 메모리에 그대로 사용하기에는 한계가 있다. 그리고 테스트 알고리즘은 일반적으로 고장의 검출만을 확인이 가능하다[8].

따라서 본 논문에서는 NAND-형 플래시 메모리에서 사용할 수 있는 형태로 March 테스트 알고리즘을 수정하고 고장을 진단[9]할 수 있는 Diagnosis 알고리즘의 형태로 알고리즘을 제안한다. 제안하는 Diagnosis 알고리즘을 사용하기 위한 BISD의 구조도 함께 제안한다.

4.1 제안하는 Diagnosis 알고리즘

Diagnosis 알고리즘을 통해 고장 검출뿐만 아니라 고장의 유형을 분류하고 고장 위치의 정보도 확인할 수 있다. 일반적인 Test 알고리즘보다 복잡도가 높은 편이지만 Diagnosis 알고리즘은 높은 고장 검출 기능과 정확한 고장 위치의 확인을 위해 적합하다.

Diagnosis 알고리즘은 다양한 고장을 검출하고 분류하기 위해 NSF-PT(NAND Single-Cell Fault-Pattern), NMF-PT (NAND Multi-Cell Fault-Pattern), INMF-PT(Inverse NAND Multi-Cell Fault-Pattern)로 구성되어 있고 알고리즘의 표기법은 Table 1과 같다.

Table 1. Algorithm notation

Notation	Description
E	Block unit Erase Performs
$R_{:n}(0)$	"0" Read form n Page
$R_{:n}(1)$	"1" Read form n Page
$R_{:0\rightarrow n-1}(0)$	"0" Read from 0 Page to n-1 Page
$R_{:n-1\rightarrow 0}(0)$	"0" Read from n-1 Page to 0 Page
$W_{:n}(0)$	"0" Write form n Page
$W_{:n}$(testPT)	test Pattern Write form n Page
$R_{:n}$(testPT)	test Pattern Read form n Page

Table 1에서 E는 전체 플래시 메모리 셀의 값을 "1"로 Erase 동작을 수행한다. R:n(0)은 n번째 해당하는 Page에서 "0"의 값을 Read하는 동작을 수행하고 R:n(1)은 n번째 해당하는 Page에서 "1"의 값을 Read하는 동작을 수행한다. W:n(0)은 n번째 Page에 "0"을 Write하는 동작을 수행하며 셀에 "1"을 Write하는 W:n(1)은 Erase가 있기 때문에 존재하지 않는다.

R:0→n-1(0)은 0번째 Page부터 n-1번째 Page까지 주소를 증가시키면서 "0"의 값을 Read하는 동작을 수행한다. 예를 들어 만약에 W:0→n-1(0)이라면 0번째 Page부터 n-1번째 Page까지 주소를 증가시키면서 셀에 "0"을 Write하게 된다. R:n-1→0(0)도 비슷하게 동작을 수행하는데 반대로 주소를 감소시키면서 Read를 수행하게 된다.

W:n(testPT)는 n번째 Page의 셀에 테스트 패턴을 Write하는 동작이고 R:n(testPT)는 n번째 Page의 셀로부터 테스트 패턴을 Read하는 동작을 수행한다. 사용할 테스트 패턴은 Table 2에서 보여주고 있다.

Table 2에 보이는 것처럼 다양한 비트수에 따른 Word 단위의 테스트 패턴을 사용함으로써 Disturbance Fault 고장을 검출할 수 있게 된다. 예를 들어 4번의 테스트 패턴을 사용하게 될 경우 해당 Page의 셀은 "1111111100000000"가

Table 2. Test Pattern

Number	Pattern
1	010101010101……010101010101
2	110011001100……110011001100
3	000011110000……111100001111
4	111111110000……000011111111
5	111111111111……111100000000
6	101010101010……101010101010
7	001100110011……001100110011
8	111100001111……000011110000
9	000000001111……111100000000
10	000000000000……000011111111

반복적으로 Write 동작이 수행되거나 Read 동작을 수행하게 된다.

만약 32비트 Word라면 16비트는 “1”을 패턴으로 나머지 16비트는 “0”을 패턴으로 사용하여 “11111111……00000000”과 같은 패턴을 사용하게 된다.

단일 셀 고장이나 다중 셀 고장을 위한 Diagnosis 알고리즘의 NSF-PT, NMF-PT, INMF-PT의 대한 설명은 다음과 같다.

1) NSF-PT

먼저 NSF-PT는 SAF, TF, AF, SOF와 같은 단일 셀 고장의 진단을 위한 알고리즘이다. NSF-PT의 알고리즘은 [Equation 1]과 같다.

$$\begin{aligned} &E, R_{0\to n-1}(1), W_{0\to n-1}(0), R_{0\to n-1}(0), \\ &E, R_{n-1\to 0}(1), W_{n-1\to 0}(0), R_{n-1\to 0}(0) \end{aligned} \tag{1}$$

NAND-형 플래시 메모리의 특징으로 인해 Write 동작을 수행하기 전에 먼저 Erase 동작을 수행해야 한다. 그래서 먼저 Block에 Erase 동작을 수행하고 전체 메모리의 셀을 Erase동작이 수행된 “1”값을 0번째 Page부터 마지막 Page까지 Read를 하면서 SAF0 고장의 유무를 진단하게 된다.

고장이 발생하지 않으면 0번째 Page부터 마지막 Page까지 Write동작을 수행하면서 셀에 “0”값을 Write하게 된다. 그리고 0번째 Page에서 마지막 Page까지 “0”값을 Read하면서 SAF1의 고장 유무를 진단한다. 그리고 동일한 동작을 반대로 주소를 감소시키면서 수행한다. 이때 TF고장을 진단하게 되고 전체 Read 동작에서 AF와 SOF의 고장도 확인할 수 있다. 만약에 고장이 진단되면 해당 고장의 고장 유형과 고장 발생 Page와 셀의 위치를 확인할 수 있다.

2) NMF-PT

NMF-PT는 CF, Disturbance Fault의 BPD와 WED의 고장과 같은 다중 결합 셀 고장을 진단할 수 있는 알고리즘이다.

Erase를 수행하고 Read하면서 Disturbance Fault의 OE를 진단하고 추가적인 Read 동작을 통해 Disturbance Fault의 RD를 진단할 수 있다. NMF-PT의 알고리즘은 [Equation 2]와 같다.

$$\begin{aligned} &E, W_0(testPT), R_0(testPT), R_{1\to n-1}(1), \\ &E, W_1(testPT), R_0(1), R_1(testPT), \\ &R_{2\to n-1}(1), \\ &\ldots, \\ &E, W_{n-1}(testPT), R_{0\to n-2}(1), \\ &R_{n-1}(testPT), \\ &E, R_{0\to n-1}(1) \end{aligned} \tag{2}$$

NSF-PT와 마찬가지로 처음 동작은 Erase 동작을 수행한다. 그리고 0번째 Page에 testPT에 따른 테스트 패턴을 Write한다. 그러면 0번째 Page의 셀에는 테스트 패턴의 값이 있고 나머지 Page의 셀에는 “1”의 값이 있게 된다. 이 상태에서 0번째 Page를 Read하면서 테스트 패턴의 값을 확인한다.

만약 고장이 발생하게 되면 WED가 진단된다. 그리고 0번째 Page를 제외한 나머지 Page의 셀 값 “1”을 Read하게 되고 만약 고장이 발생하면 CF와 BPD가 진단된다.

마찬가지로 마지막 Page까지 Page 주소를 증가시키면서 테스트 패턴을 Write하고 테스트 패턴을 Write한 Page는 테스트 패턴의 값을 Read하고 나머지 Page의 셀 값은 “1”을 Read하면서 알고리즘을 진행한다. 그리고 지속적으로 Erase동작을 수행하고 Read동작을 수행하면서 OE를 진단할 수 있고 마지막에 추가적으로 Read를 수행하면서 RD를 진단할 수 있다.

3) INMF-PT

INMF-PT는 CF, Disturbance Fault의 BED와 WPD의 고장과 같은 다중 결합 셀 고장을 진단할 수 있으며 알고리즘의 형태가 NMF-PT 알고리즘의 반전된 모습과 같다. INMF-PT의 알고리즘은 [Equation 3]과 같다.

$$\begin{aligned} &E, W_0(testPT), W_{1\to n-1}(0), \\ &R_0(testPT), R_{1\to n-1}(0), \\ &E, W_0(0), W_1(testPT), W_{2\to n-1}(0), \\ &R_0(0), R_1(testPT), R_{2\to n-1}(0), \\ &\ldots, \\ &E, W_{0\to n-2}(0), W_{n-1}(testPT), \\ &R_{0\to n-2}(0), R_{n-1}(testPT), \\ &E, R_{0\to n-1}(1) \end{aligned} \tag{3}$$

INMF-PT도 처음 동작은 Erase 동작을 수행하게 된다. 그리고 0번째 Page에 테스트 패턴의 값을 Write하고 나머지 Page들에는 “0”의 값을 Write한다. 그러면 0번째 Page의 셀은 테스트 패턴의 값이 있고 나머지 Page의 셀은 “0”값이 있게 된다. 이 상태에서 0번째 Page를 Read하면서 테스트 패턴의 값을 확인하게 되고 만약 고장이 발생한다면 WPD의 고장을 진단할 수 있게 된다.

그리고 나머지 Page의 셀 값 “0”을 Read하면서 BED와 CF를 진단할 수 있게 된다. 마찬가지로 마지막 Page까지 Page 주소를 증가시키면서 테스트 패턴을 Write하고 나머지 Page에는 “0”값을 Write한다. 그리고 테스트 패턴을

Write한 Page는 테스트 패턴을 Read하면서 고장을 진단하고 “0”을 Write한 Page는 “0”값을 Read하면서 마찬가지로 고장을 진단한다.

INMF-PT도 NMF-PT와 동일한 방법으로 OE와 RD를 진단할 수 있다.

Fig. 5는 본 논문에서 제안한 Diagnosis 알고리즘의 Flow을 보여준다.

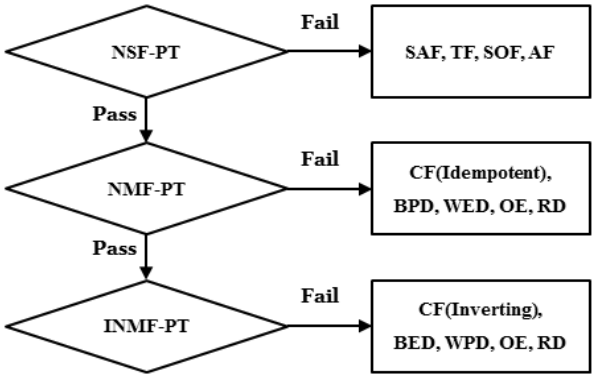


Fig. 5. Diagnosis Algorithm Flow

Fig. 5에서 보이는 것처럼 먼저 NSF-PT 알고리즘을 수행하면서 검출이 쉬운 단일 고장을 진단하게 된다. 만약 고장이 발생하면 Fail을 통해서 해당하는 고장의 유형과 위치를 확인할 수 있게 된다. 만약 NSF-PT 알고리즘에서 고장이 발생하지 않으면 Pass를 통해 다중 결합 셀 고장을 위한 NMF-PT 알고리즘을 수행한다. 마찬가지로 고장이 발생하면 Fail신호를 통해 해당하는 고장의 유형과 위치를 확인할 수 있다.

그리고 고장이 발생하지 않으면 Pass를 통해 INMF-PT 알고리즘을 수행하면서 고장 진단을 하면서 전체 알고리즘을 종료하게 된다.

Diagnosis 알고리즘은 Test 알고리즘에 비해 알고리즘의 복잡도가 높아서 동작 시간이 증가할 수 있다. 하지만 공정 단계가 진행될수록 진단되는 고장의 수와 유형이 적기 때문에 NSF-PT, NMF-PT, INMF-PT 알고리즘 중에 사용자의 선택에 따라 해당 알고리즘을 동작하면서 NAND-형 플래시 메모리의 진단 시간을 감소시킬 수 있다.

4.2 제안하는 BISD 구조

제안하는 Diagnosis 알고리즘의 동작을 위한 BISD 구조를 제안한다. Fig. 6은 제안하는 BISD 구조를 보여준다.

제안하는 BISD의 구성 요소는 외부로부터 신호를 받아서 처리해주는 Controller, 해당하는 알고리즘의 각 Elements와 테스트 패턴을 사용하여 메모리 진단을 수행하는 Test Pattern Generator, 일반 모드와 테스트 모드를 선택할 수 있게 해주는 Mode Selector로 구성되어 있다.

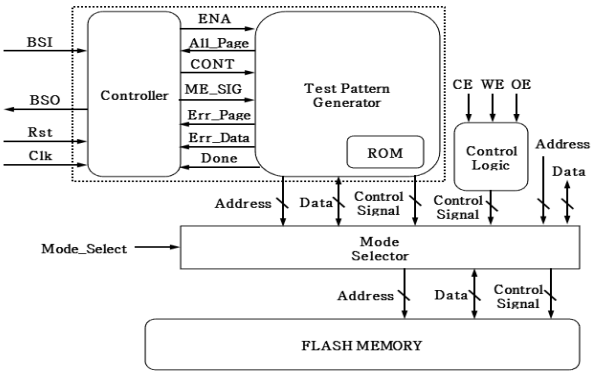


Fig. 6. Proposed BISD Architecture

BISD 모듈을 사용하기 위해서 먼저 Mode Selector에 BISD 모드를 선택하기 위한 Mode_Select신호를 인가한다. BISD 모드가 되면 BISD 모듈은 먼저 Serial 입력 신호인 BSI 신호를 통해 사용하는 Diagnosis 알고리즘과 패턴을 외부에서 입력받는다. 그러면 Controller는 먼저 ENA 신호를 Test Pattern Generator에 보내면서 테스트 수행을 준비시킨다. Test Pattern Generator으로부터 알고리즘을 수행할 NAND-형 플래시 메모리의 전체 Page의 수만큼 All_Page 신호를 Controller가 받게 되면 Controller는 사용자가 선택한 Diagnosis 알고리즘의 전체 Elements를 생성하여 순차적으로 Test Pattern Generator에 ME_SIG를 통해 보내게 된다. 여기서 알고리즘의 Elements는 예를 들어 NSF-PT의 E, $R_{:0\rightarrow n-1}(1)$, $W_{:0\rightarrow n-1}(0)$ 등과 같은 동작 요소들을 말한다. 그리고 Test Pattern Generator는 해당하는 Diagnosis 알고리즘의 Elements를 수행하면서 선택된 테스트 패턴을 ROM으로부터 받아서 사용하게 된다.

만약에 Diagnosis 알고리즘을 수행하면서 고장이 발생할 경우 Test Pattern Generator에서는 고장 발생 위치를 나타내는 Err_Page신호와 고장 유형 정보를 나타내는 Err_Data 신호를 Controller에 전송하게 된다. 그러면 Controller는 Serial 출력 신호인 BSO신호를 통해 해당 고장 정보를 출력하게 된다.

Test Pattern Generator는 FSM기반으로 2개의 FSM이 존재한다. Fig. 7은 Test Pattern Generator의 FSM을 보여준다.

Test Pattern Generator의 FSM은 먼저 Idle 상태로 외부로부터 신호를 기다리다가 ENA신호가 Controller를 통해 전송되면 ME fetch상태로 이동하여 ME_SIG신호를 기다린다. ME_SIG가 전송되면 ME fetch에서는 해당하는 Elements를 Fetch하고 PT fetch상태로 이동하여 사용하려는 테스트 패턴을 Fetch하게 된다. 그리고 Exec상태로 이동하여 Fetch한 Elements와 테스트 패턴에 해당하는 동작을 수행하게 된다.

D fetch상태에서는 알고리즘을 통해 메모리에 인가된 Data를 Fetch하게 된다. 그리고 Compare상태로 이동하여 Fetch한 Data와 알고리즘을 수행하면서 예상했던 값을 비교하게 된다. 만약 고장이 없으면 다시 ME fetch 상태로 돌아

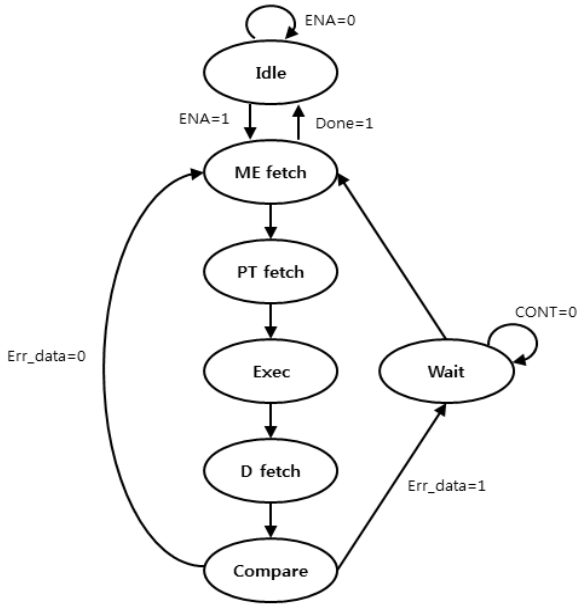


Fig. 7. Test Pattern Generator FSM

가 Controller로부터 신호를 기다린다. 하지만 만약 고장이 발생하게 되면 Wait상태로 이동하여 해당하는 고장 주소인 Err_Page신호와 고장 유형의 정보 신호인 Err_Data신호를 Controller에 보내면서 ME fetch상태로 돌아가 다음 신호를 기다린다.

Fig. 8은 Exec의 FSM을 보여준다. Exec 상태로 이동되면 해당하는 Elements에 따라 Erase 동작, Write 동작과 Read 동작을 수행하게 된다. 메모리의 주소에 따라 해당 동작을 수행하면서 만약 AD_Done신호가 “1”이 되면 다시 Exec상태로 돌아와 다음 상태인 D fetch상태로 이동된다.

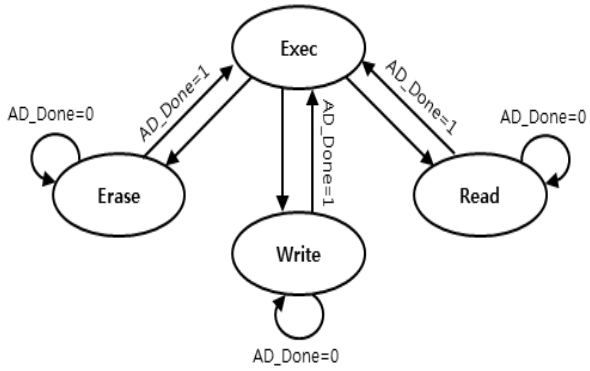


Fig. 8. Exec FSM

제안하는 BISR 구조를 통해 메모리 수율 증가를 위해서 BISD 모듈은 Diagnosis 알고리즘을 통해 얻어진 고장 정보들을 BIRA 모듈로 전송해야 하며 BIRA 모듈은 이러한 고장 정보를 통해 고장이 발생한 위치에 Redundancy를 할당하면서 메모리를 수리하게 된다.

5. 제안하는 BISR를 위해 제안된 BIRA 구조

제안하는 BISR를 위해서 고장을 진단할 수 있는 BISD 모듈과 Redundancy를 고장 위치에 할당할 수 있는 BIRA 모듈이 필요하다.

기존의 SRAM이나 DRAM에서 사용하는 BIRA 알고리즘으로는 ESP(Essential Spare Pivoting)알고리즘이 가장 효과적으로 많이 사용되어지고 있다[10]. ESP 알고리즘은 고장이 발생한 Row나 Column의 Line에서 고장 셀의 최대 수에 따라 Spare를 할당하는 방법이다. 그러나 이러한 ESP 알고리즘은 NAND-형 플래시 메모리에 사용하기에는 부적합하다. 따라서 NAND-형 플래시 메모리를 위해 ESP 알고리즘의 수정이 필요한데 기존의 [11]에서 NAND-형 플래시 메모리를 위해 수정된 ESP 알고리즘과 BIRA 모듈을 제안하고 있다.

본 논문에서는 [11]에서 제안하는 수정된 ESP-1, ESP-2 알고리즘과 BIRA 모듈을 사용하여 앞에서 제안하는 BISD와 통합하여 고장 수율 향상을 위한 전체 BISR 구조를 제안한다. Fig. 9는 [11]에서 제안된 NAND-형 플래시 메모리의 Redundancy 구조를 보여준다.

기존의 BIRA 구조는 Spare Row와 Spare Column이 존재하고 고장이 발생한 메인 메모리의 수리를 위해 Row나 Column을 Line 단위로 할당했다. 그러나 NAND-형 플래시 메모리의 특성으로 수리를 위해 Spare Row가 아닌 Spare Block을 사용해야 한다.

Fig. 9의 BNB는 NAND-형 플래시 메모리의 기본 Block으로 2장에서 설명했다. 2장에서 설명한 것처럼 NAND-형 플래시 메모리는 셀 배열의 SL이 활성화되어야 WL의 셀에 접근할 수가 있다. 따라서 NAND-형 플래시 메모리의 고장 수리를 위한 여분의 SNB(Spare NAND Block)는 BNB의 구조와 같다. SNB에서도 SL이 존재하고 SL이 활성화되어야만 SNB의 WL의 셀에 접근하는 것은 BNB와 동일하다. 그래서 메인 메모리 Blcok의 고장 셀을 수리하기 위해서 Block 단위 Spare인 SNB를 할당해야 한다.

Fig. 9에서 먼저 각 BNB의 고장 셀의 수를 확인한다. BNB0은 2개의 단일 고장이 감지되었고 BNB1은 WL1이 다

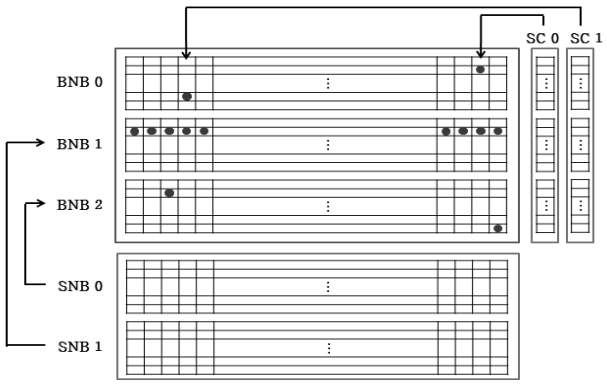


Fig. 9. Redundancy Architecture for Flash Memory

중결합 고장의 의해 많은 수의 고장이 감지되고 BNB2는 2개의 단일 고장이 감지되었다. 수정된 B ESP 알고리즘을 수행하게 되면 먼저 고장의 수가 가장 많은 BNB1을 SNB1로 수리하게 되고 BNB0에서 발생한 2개의 단일 고장은 각각 SC0과 SC1로 수리를 하게 된다. 그리고 마지막으로 BNB2에서 2개의 단일 고장은 남아 있는 Spare가 SNB0만 있으므로 SNB0으로 고장을 수리하게 된다.

Fig. 9에서 보이는 것처럼 BNB의 고장 수리를 위한 Spare로 SNB의 크기가 메인 메모리인 BNB와 동일하기 때문에 오버헤드가 커지는 단점이 있다. 하지만 오버헤드가 증가해도 Spare를 사용하면 BNB1처럼 많은 고장이 발생한 메모리 Block의 수리가 가능하기 때문에 메모리를 재사용할 수 있다는 장점이 있다.

6. 제안하는 BISR 구조

제안하는 BISD구조는 앞에서 제안한 BISD와 [11]에서 제안된 BIRA를 사용한다. BISD에서 Diagnosis 알고리즘을 사용하여 고장을 진단하게 되면 해당하는 고장의 정보를 BIRA에 전송하게 된다. BIRA는 BISD로부터 받은 고장 정보를 통해 고장이 발생한 메모리 셀에 가장 적합한 방법으로 Spare를 할당한다.

Fig. 10은 제안하는 BISR의 구조를 보여준다. Fig. 10에서 Mode Selector의 신호에 따라 일반 모드와 테스트 모드를 선택하게 된다. 테스트 모드를 선택하게 되면 BISD 모듈이 작동되면서 사용자가 정의한 알고리즘이 수행된다. 그리고 메인 메모리의 고장을 진단하면서 해당하는 고장 정보(고장 유형, 고장 위치)를 BIRA 모듈에 전송한다. 그러면 BIRA 모듈은 BISD 모듈에서 전송한 고장 정보를 통해 수정된 ESP알고리즘을 사용하여 가장 효율적으로 Spare를 메인 메모리에 할당하게 된다. 그리고 BIRA 모듈은 고장을 수리한 정보(수리 위치)를 CAM에 저장한다.

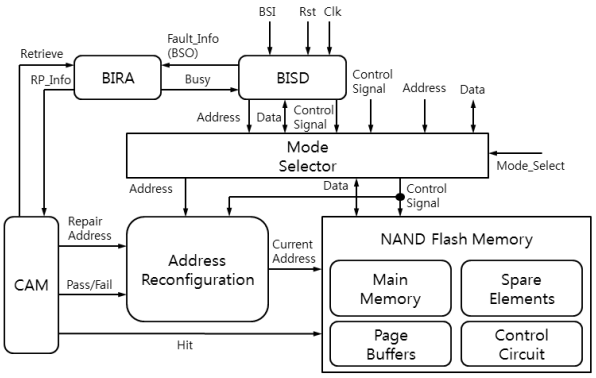


Fig. 10. Proposed BISR Architecture

모든 BISR 동작이 완료한 후에 일반 모드로 메인 메모리를 사용하려 하면 고장이 발생한 주소로 접근하지 않도록 주소 재구성 모듈에서는 일반모드에서 받은 주소와 CAM에서 저장한 주소를 비교하게 된다. 만약 고장이 있었고 수리되었던 주소이면 Fail 신호에 의해 수리 주소로 NAND-형 플래시 메모리에 접근한다. 만약 고장이 없었고 수리하지 않은 주소이면 Pass 신호에 의해 일반 모드에서 받은 주소로 NAND-형 플래시 메모리에 접근하게 된다.

Fig. 11은 BISR의 수리 데이터와 주소를 저장하는 CAM의 Format을 보여준다.

Format:	Redundancy Type	Redundancy Location	Block	Fault Address	SNB	Column	Faulty IO	Valid
Spare Column:	0	Redundancy Location	Block	Fault Address	(Reserved)	Column	Faulty IO	Valid
Spare Block:	1	Redundancy Location	Block	Fault Address	SNB	(Reserved)	(Reserved)	Valid

Fig. 11. Format of CAM

CAM은 전체 수리 과정 이후에 수리 데이터와 주소를 저장한다. Redundancy 유형이 “0”이면 SC로 고장을 수리하기 위해 사용된 것이고 “1”이면 SNB로 고장을 수리하기 위해 사용된 것을 나타낸다. Redundancy 위치는 사용한 Redundancy의 위치를 나타낸다. 그리고 Block과 고장 주소 필드는 메인 메모리에서 고장이 발생한 Block과 Block에서 고장이 발생한 주소를 나타낸다. SNB, Column, Faulty IO는 사용한 Redundancy의 유형에 따라 할당되는 주소 값을 나타낸다. 마지막으로 Valid 필드는 CAM에 저장된 정보들이 사용할 수 있는지를 식별한다.

Fig. 12는 제안하는 BISR의 전체 동작과정을 Flow로 보여준다.

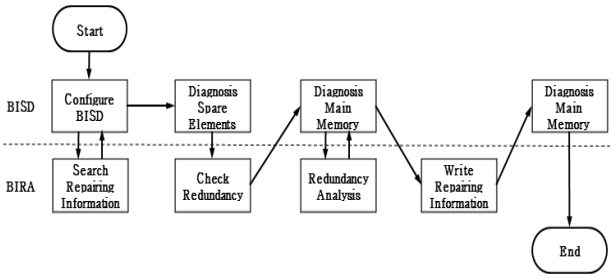


Fig. 12. BISR Flow

Fig. 12의 제안하는 BISR의 동작과정은 먼저 BISD 구성단계가 수행이 된다. BISD 구성단계에서는 사용자가 정의한 Diagnosis 알고리즘을 BISD 모듈에 인가하게 된다. 그리고 BIRA 모듈에서는 수리 정보 검색 단계를 통해 CAM에 저장된 수리데이터의 유무를 먼저 확인하게 된다. BISD 구성 단계 후에 SC와 SNB와 같은 Spare 메모리에 대한 고장을 먼저 진단하게 된다. 만약에 Spare에 고장이 발생하게 되면 메인 메모리가 고장이 발생해도 수리를 할 수 없기 때문이다.

Spare 메모리의 고장을 진단한 정보를 통해 BIRA 모듈

은 Redundancy 확인 단계에서 고장이 발생한 Spare 메모리를 수리 과정에서 제외한다. 그리고 메인 메모리 진단 단계에서 메인 메모리에 대한 고장을 진단하게 되고 만약 고장이 발생하게 되면 BIRA 모듈은 Redundancy 분석 단계를 통해 고장 위치에 Spare를 할당하여 고장을 수리하게 된다.

수리가 완료된 후에 BIRA 모듈은 수리 정보 기록 단계에서 수리 데이터와 주소를 CAM에 저장한다. 마지막으로 Spare를 할당하고 수리가 완료된 후에 추가적으로 메인 메모리를 다시 고장을 진단하면서 수리하면서 혹시 고장이 발생하지 않았는지 확인하고 BISR의 동작이 종료된다.

7. 실험 결과

제안하는 BISR 구조 시뮬레이션을 위해 Xilinx사의 ISE 14.1을 사용하고 Verilog를 통해 구현하였다.

Fig. 13은 Diagnosis 알고리즘에서 NMF-PT 알고리즘의 동작 과정을 보여주고 고장이 진단되지 않았을 때의 정상 동작 시뮬레이션 결과를 보여준다.

먼저 Mode 신호가 “1”이면 BISD 모드로 BISD 모듈이 동작하게 된다. BISD 모듈에서 BSI 신호를 통해 사용자가 정의한 Diagnosis 알고리즘과 패턴을 준비하게 된다. 그리고 Controller는 Test Pattern Generator에 ENA 신호를 보내게 되고 Test Pattern Generator는 고장 진단을 수행할 메인 메모리의 전체 Page의 수를 All_Page 신호로 다시 Controller에 보내게 된다.

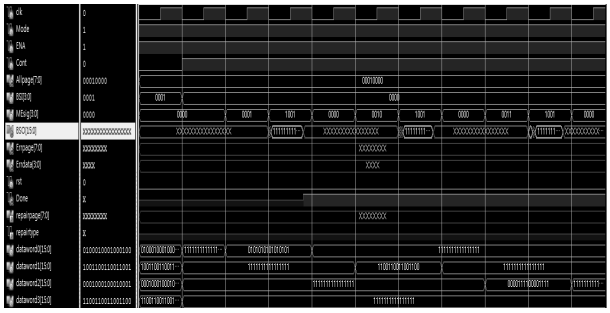


Fig. 13. Normal Operation

Fig. 13에 All_Page 신호를 보면 “00010000”으로 고장 진단을 수행할 NAND-형 플래시 메모리의 Page의 수가 총 32라는 것을 확인할 수 있다. 그리고 ME_SIG 신호에 따라 메인 메모리의 고장 진단을 위한 Elements 동작을 수행하게 되고 BSO를 통해 Read동작에 의한 정보를 확인할 수 있다. 그리고 Err_Page, Err_Data, Repair _Page, Repair_Type은 고장이 진단되지 않았기 때문에 아직은 값을 확인할 수 없다.

Fig. 14는 NMF-PT 알고리즘의 첫 번째 Erase 동작 이후 W:0(testPT) 동작에 따른 테스트 패턴이 정상적으로 입력되었는지 확인할 수 있다.

먼저 ME_SIG 신호 “0000”에 의해 Block에 Erase 동작을 수행한다.

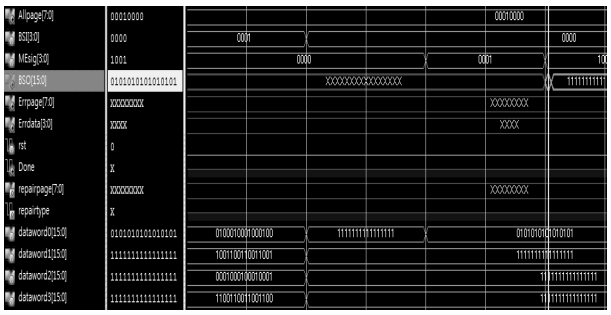


Fig. 14. Erase Operation and W:0(testPT) Operation

그리고 “0001”신호에 의해 W:0(testPT) 동작이 수행되고 첫 번째 Page의 셀 값이 테스트 패턴인 “0101010101010101”로 Write된다. 정상적으로 동작이 수행되었는지 확인하기 위해 BSO 신호를 보면 첫 번째 Page는 테스트 패턴으로 값이 확인되고 나머지 Page는 Erase된 값 “1111111111111111”으로 확인된다.

Fig. 15와 Fig. 16은 NMF-PT 알고리즘의 두 번째 Erase 동작 이후 W:1(testPT) 동작에 따른 정상 동작을 확인할 수 있다.

Fig. 15는 ME_SIG신호 “0000”에 의해 Block에 Erase 동작을 수행하고 첫 번째 Page에 “1111111111111111”으로 Erase된 정상적인 값을 보여준다.

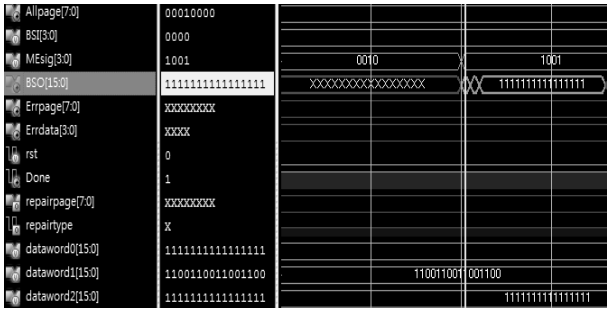


Fig. 15. Second Elements

Fig. 16은 두 번째 Page의 값이 ME_SIG “0010”에 의해서 W:1(testPT) 동작이 수행되는 것을 보여준다. 테스트 패턴 “1100110011001100”의 값이 두 번째 Page에 Write되고 BSO 신호를 통해 확인할 수 있다.

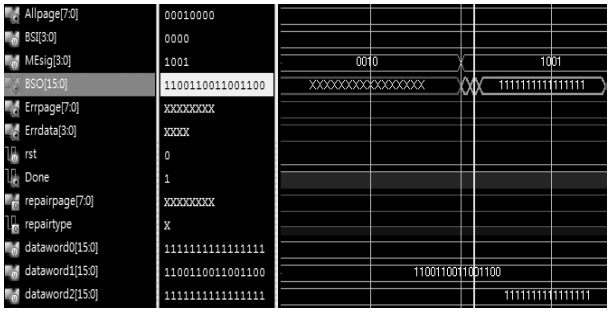


Fig. 16. W:1(testPT) Operation

Fig. 17과 Fig. 18도 마찬가지로 NMF-PT 알고리즘의 세 번째 Erase 동작 이후 W:2(testPT) 동작에 따른 정상 동작을 확인할 수 있다.

Fig. 17은 ME_SIG신호 “0000”에 의해 Block에 Erase 동작을 수행하고 첫 번째 Page의 값과 두 번째 Page의 값이 “1111111111111111”으로 Erase된 정상 동작을 보여준다.

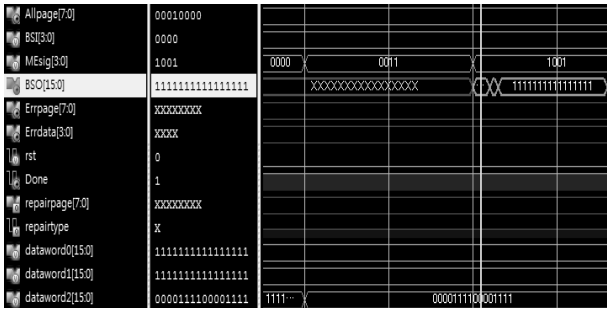


Fig. 17. Third Erase Operation

Fig. 18은 세 번째 Page의 값이 ME_SIG “0011”에 의해서 W:2(testPT) 동작이 수행되는 것을 보여준다. 테스트 패턴 “0000111100001111”의 값이 세 번째 Page에 Write되고 BSO 신호를 통해 확인할 수 있다.

Fig. 18. $W_{:2}$(testPT) Operation

Fig. 13에서 Fig. 18까지는 제안하는 BISR 구조의 고장이 발생하지 않은 경우로 정상 동작을 수행하는 시뮬레이션의 결과를 확인할 수 있다. 다음은 고장이 발생했을 때의 신호들의 시뮬레이션 결과를 설명한다.

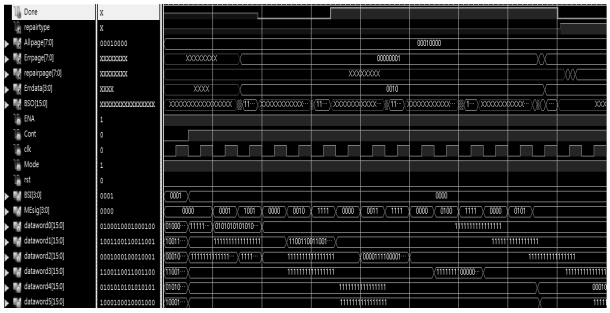


Fig. 19. Fault Detection

Fig. 19는 고장이 발생했을 때의 시뮬레이션 결과를 보여준다.

Fig. 19는 앞에서 설명한 정상 동작과 동일하게 동작한다. 먼저 Mode신호를 받고 BISD 모드를 선택해서 BISD 모듈을 동작한다. 그리고 BSI 신호를 통해 알고리즘을 정의하고 ME_SIG신호를 통해 해당 Elements를 수행하게 된다.

고장이 발생했을 경우 Err_Page신호, Err_Data신호, Repair_Page와 Repair_Type신호를 출력하게 된다.

Fig. 20은 NMF-PT 알고리즘의 첫 번째 Erase동작을 수행하고 W:0(testPT)했을 때 고장을 보여준다.

Fig. 20도 Fig. 14와 마찬가지로 NMF-PT의 첫 번째 Erase동작을 Block에 수행하고 첫 번째 Page에 W:0(testPT)동작을 수행한다. 이때 두 번째 Page에 고장이 발생하는 것을 확인할 수 있다.

Erase를 수행하고 두 번째 Page부터 마지막 Page까지는 Erase된 값인 “1111111111111111”이 인가되어야 한다.

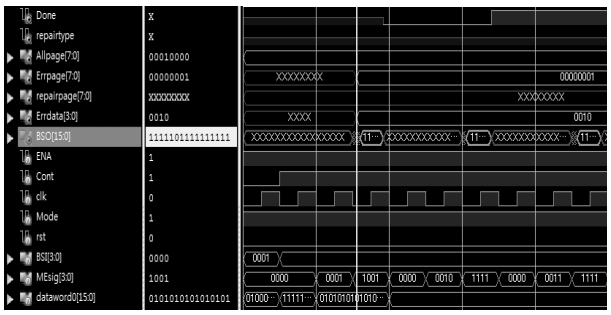


Fig. 20. Fault Detection for $W_{:0}$(testPT) Operation

그러나 Fig. 20에서 보이는 것처럼 두 번째 Page의 여섯 번째 열의 셀 값이 “0”으로 변경되어 “1111101111111111”으로 고장이 진단되는 것을 확인할 수 있다. Err_Page는 두 번째 Page를 나타내는 “00000001”을 출력하고 Err_data를 보면 “0010”으로 WPD 고장이 발생했다는 것을 확인할 수 있다.

Fig. 21과 Fig. 22는 여섯 번째 Page에서 고장이 추가적으로 발생한 것을 확인할 수 있다.

Fig. 21은 Erase 동작 이후에 W:5(testPT)동작을 수행할 때 테스트 패턴 “0101010101010101”을 사용했다. 하지만 여

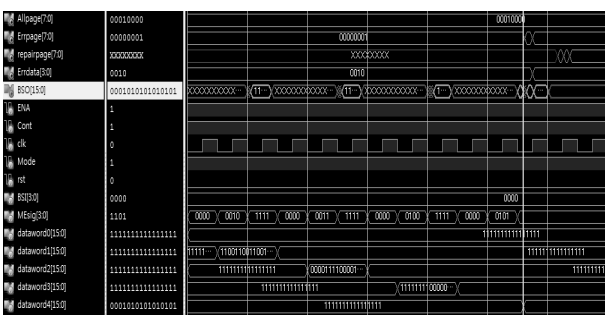


Fig. 21. Fault Detection for $W_{:5}$(testPT) Operation

셋 번째 Page의 두 번째 열의 셀 값이 “0”으로 변경되어 “0001010101010101”으로 고장이 감지되는 것을 확인할 수 있다. 그리고 Err_Data 신호로 이 고장이 WPD라는 것을 진단할 수 있다.

Fig. 22는 고장이 발생한 Page의 위치를 보여준다.

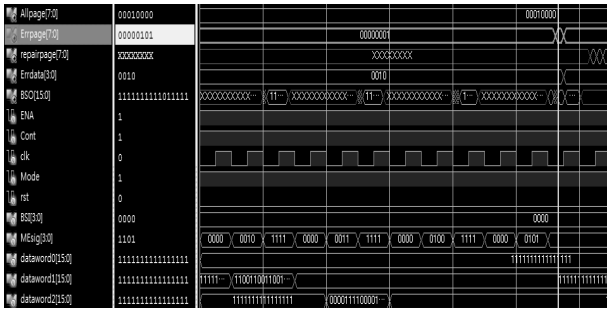


Fig. 22. Fault Page for W:5(testPT) Operation

Fig. 23과 Fig. 24는 동일하게 W:5(testPT)동작을 수행하는 Page의 다음 Page에서 고장이 발생하는 것을 보여준다.

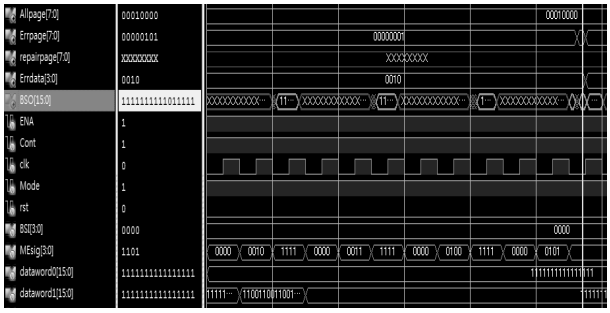


Fig. 23. Additional Fault Detection

Fig. 23은 W:5(testPT)동작을 수행하고 그다음 Page의 값을 확인했을 때 고장이 발생하는 것을 확인할 수 있다. 테스트 패턴을 Write한 Page를 제외한 다른 Page의 값은 “1111111111111111”로 Erase된 값이 인가되어야 한다. 그러나 Fig. 23에서 일곱 번째 Page의 값이 “1111111111011111”으로 일곱 번째 Page의 열한 번째 열의 셀의 값이 “0”으로 변경된 것을 확인할 수 있고 Err_Data 신호의 값이 “0110”으로 이 고장이 BPD 고장인 것을 알 수 있다.

Fig. 24는 고장이 발생한 고장 Page를 보여준다.

그리고 메인 메모리의 수리를 위한 Spare로 두 개의 SC와 두 개의 SNB로 구성되어 있는데 현재 고장 진단을 수행하는 메모리의 고장이 세 개이기 때문에 SC를 이용한 고장 수리보다 SNB를 할당하여 메모리를 수리해야 한다. 따라서 Fig. 24의 Repair_Type을 보면 “1”로 SNB가 할당되어 메모리가 수리되었다는 것을 확인할 수 있다.

제안하는 BISR 구조의 시뮬레이션을 통해 NAND-형 플래시 메모리에서 발생하는 고장을 검출하고 고장의 종류와 위치와 같은 고장정보를 얻을 수 있었다. 그리고 고장이 발생

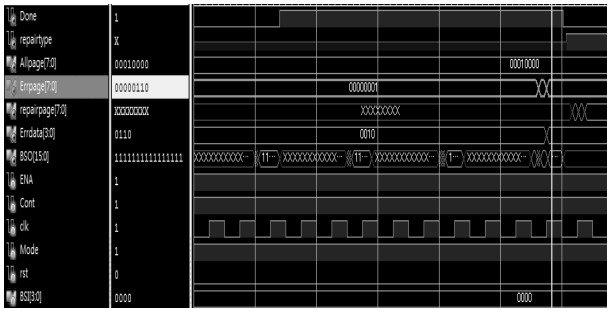


Fig. 24. Additional Fault Page

한 위치를 수리하기 위해 사용되는 SC와 SNB의 수에 따른 수율을 얻을 수 있었다. 그리고 결과적으로 SC의 수보다 SNB의 수가 많을수록 고장 수율이 증가한다는 것을 확인할 수 있었다.

Table 3. Repair Rates for Spare

Fault number	3 SNB, 1 SC	2 SNB, 5 SC
Single-cell fault :3 Multi-cell fault :2	100%	100%
Single-cell fault :6 Multi-cell fault :4	94%	93%
Single-cell fault :9 Multi-cell fault :6	88%	83%
Single-cell fault :12 Multi-cell fault :8	80%	76%
Single-cell fault :15 Multi-cell fault :10	71%	64%

Table 3은 이러한 수율을 보여준다. NAND-형 플래시 메모리를 위한 BISR 구조는 수리를 위한 Redundancy로 Spare Block방법을 사용하기 때문에 SNB의 수가 많아질수록 시스템 오버헤드가 증가하는 단점이 발생할 수 있다. 하지만 제안하는 방법을 통해 고장이 발생한 NAND-형 플래시 메모리의 고장을 수리하고 재사용할 수 있는 장점으로 이러한 단점은 어느 정도 허용될 수 있다. 그리고 [9]에서 RAM을 위한 Diagnosis 알고리즘 방법은 단일 셀 고장과 CF 고장에 대한 고장 종류와 위치를 진단할 수 있지만 NAND-형 플래시 메모리의 고유 고장인 Disturbance 고장을 진단할 수 없다. 그리고 [11]에서의 BISR은 BIST를 사용함으로써 단일 셀 고장과 Disturbance 고장의 유무를 확인하여 고장을 수리할 수는 있지만 이러한 고장의 종류를 구분하고 고장의 위치까지는 진단할 수가 없다. 하지만 본 논문에서는 BISD를 사용하는 BISR을 통해 NAND-형 플래시 메모리에서 발생하는 단일셀 고장과 Disturbance 고장의 유무를 확인할 수 있고 뿐만 아니라 고장의 종류와 고장의 위치도 진단할 수 있는 장점이 있다. 이것은 만약 공정단계에서 NAND-형 플래시 메모리에서 동일한 위치로 계속 어떠한 고장이 지속적으로 발생한다면 그것은 메모리의 문제보다 공정단계에서 문제가 있음을 인지할 수 있게 한다.

8. 결 론

본 논문에서는 NAND-형 플래시 메모리를 위한 BISR 구조를 제안하였다. 먼저 NAND-형 플래시 메모리에서 발생하는 고장 위치와 고장 유형을 진단하기 위한 Diagnosis 알고리즘과 BISD 구조를 제안하였다.

제안하는 Diagnosis 알고리즘은 NSF-PT, NMF-PT, INMF-PT 알고리즘이 있다. 먼저 NSF-PT 알고리즘은 SAF, TF, SOF, AF와 같은 단일 셀 고장을 진단하고 NMF-PT와 INMF-PT 알고리즘은 NAND-형 플래시 메모리의 고유 고장인 Disturbance 고장을 진단할 수 있다.

제안하는 BISD 모듈은 전송하는 고장 정보를 분석하고 NAND-형 플래시 메모리의 수리를 위해 Spare를 할당하기 위한 기존의 제안된 BIRA 구조를 사용하였다. 제안하는 BISD 모듈과 제안된 BIRA 모듈을 통합하여 고장 진단과 수리를 동시에 할 수 있는 BISR을 시뮬레이션을 통해 검증하였고 각 Spare에 대한 수율 증가를 확인할 수 있었다.

제안하는 BISR을 통해 NAND-형 플래시 메모리에서 발생하는 고장에 대한 진단과 수리가 가능하며 이는 높은 수율과 메모리의 효율적인 재생산을 기대할 수 있게 한다.

Reference

[1] M. G. Mohammad, K. K. Saluja, and A. Yap, “Testing Flash Memories”, In Proceedings of Thirteenth Intel Conference on VLSI Design, pp.406-411, 2000.

[2] M. F. Chang, W. K. Fuchs, J. H. Patel, “Diagnosis and repair of memory with coupling faults,” computers, IEEE Transactions on, Vol.38, No.4, April, 1989, pp.493-500.

[3] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, Flash Memories, Boston, MA: Kluwer Academic, 1999.

[4] Stefano DI CARLO, Michele FABIANO, Roberto PIAZZA, and Paolo PRINETTO, “Exploring Modeling and Testing of NAND Flash memories,” Test Symposium East-West Design, pp.47-50, 2010.

[5] V. G. Mikitjuk, V. N. Yarmolik, and A. J. van deGoor, “RAM Testing Algorithms for Detection Multiple Linked Faults,” Proc. European Design and Test Conf., pp.435-439, 1996.

[6] M. Mohammad and K. K. Saluja, “Flash memory disturbances : modeling and test”, in Proc. IEEE VLSI Test Symp. (VTS), Marina Del Rey, California, Apr., 2001, pp.218-224.

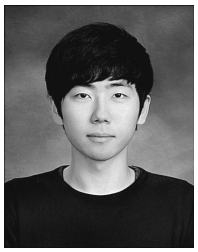
[7] J.-C. Yeh, C.-F. Wu, K.-L. Cheng, Y.-F. Chou, C.-T. Huang, C.-W. Wu, “Flash memory built-in self-test using March-like algorithms”, IEEE Int. Workshop on Electronic Design, Test and Applications, 2002, pp.137-141.

[8] Nur Qamarina Mohd Noor, Azilah Saparon, and Yusrina Yusof, “An Overview Of Microcode based and FSM based Programmable Memory Built-In Self Test(MBIST) Controller for Coupling Fault Detection”, IEEE Symposium on Industrial Electronics and Applications(ISIEA 2009).

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

[10] C.-T. Huang, C.-F. Wu, J.-F. Li, and C.-W. Wu, “Built-in redundancy analysis for memory yield improvement”, IEEE Transactions on Reliability, Vol.52, pp.386-399, December, 2003.

[11] Y.-Y. Hsiao, C.-H. Chen, and C.-W. Wu, “Built-In Self-Repair Schemes for Flash Memories”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.29, No.8, pp.1243-1256, Aug., 2010.



김 태 환

e-mail : ktaehwan@ssu.ac.kr

2010년 숭실대학교 전자계산원(학사)

2014년 숭실대학교 컴퓨터학과 석사

관심분야 : 컴퓨터구조, 메모리테스트, VLSI 설계 및 테스트, 임베디드 시스템, 메모리 에러정정

장 훈

e-mail : hoon@ssu.ac.kr

1987년 서울대학교 전자공학과(학사)

1989년 서울대학교 전자공학과(석사)

1993년 University of Texas at Austin (박사)

1991년 IBM Inc. Senior Member of Technical Staff

1993년 Motorola Inc. Senior Member of Technical Staff

1988년~현 재 숭실대학교 컴퓨터학부 교수

관심분야 : 컴퓨터구조, 메모리 테스트, VLSI 설계 및 테스트, 임베디드 시스템