논문 2007-44SD-6-5

플래시 메모리를 위한 유한 상태 머신 기반의 프로그래머블 자체 테스트

(FSM-based Programmable Built-In Self Test for Flash Memory)

김 지 환*, 장 훈**

(Ji-Hwan Kim and Hoon Chang)

요 약

본 논문에서 제안한 FSM 기반의 프로그래머블 BIST(Built-In Self-Test)는 플래시 메모리를 테스트하기 위한 기존의 알고리즘들을 코드화 하여 그중에서 선택된 알고리즘의 명령어 코드를 받아서 플래시 메모리 테스트를 수행한다. 또한 제안하는 구조는 각 알고리즘에 대한 테스트 절차를 간단하게 한다. 이외에도 플래시 메모리 BIST를 재구성하는데 걸리는 시간도 기존의 BIST와 비교해 볼 때 매우 적다. 우리가 제안한 BIST 구조는 자동적으로 Verilog 코드를 생성해주는 프로그래머블 플래시 메모리 BIST 생성기이다. 만약 제안된 방법을 실험하게 되면, 제안된 방법은 이전의 방법들과 비교해서 크기도 더 작을 뿐만 아니라 융통성 면에서도 좋은 성과를 얻었다.

Abstract

We propose a programmed on-line to FSM-based Programmable BIST(Buit-In Self-Test), with selected command, to select a test algorithm from a predetermined set of algorithms that are built in the Flash memory BIST. Thus, the proposed scheme greatly simplifies the testing process. Besides, the proposed FSM-based Programmable BIST is more efficient in terms of circuit size and test data to be applied, and it requires less time to configure the Flash memory BIST. We also will develop a programmable Flash memory BIST generator that automatically produces Verilog code of the proposed BIST architecture for a given set of test algorithms. If experiment the proposed method, the proposed method will achieves a good flexibility with smaller circuit size compared with previous methods.

Keywords : 플래시 메모리, 외란 고장, FSM 기반의 프로그래머블 BIST

I. 서 론

최근 전자기기의 소형화 증가와 확대로 인해 플래시 메모리가 각광받고 있다. 기존의 자기메모리인 하드디스크나 플로피디스크의 크기가 플래시 메모리에 비해 수배나 크며, 고전력을 요하기 때문에 플래시 메모리로의 교체가 대두 되고 있다.

그림 1은 플래시 메모리의 성장 동향을 보여주는 그림이다. 매해 거듭 될수록 플래시 메모리의 성장 폭이 증가함을 한눈에 알 수 있다.

비활성 메모리인 플래시 메모리는 기존의 EPROM에 UV(자외선)을 이용하여 부유 게이트로부터 전자들을 이동시키던 방식을 전기적인 방식으로 전환시킨 것으로, 이러한 특성 때문에 플래시 메모리는 전기로 데이터를 읽고 쓰고, 지울 수 있는 특징을 가지고 있다. 플래시 메모리는 FG(Floating Gate)에 전기를 가함으로써 데이터를 저장하거나 삭제한다[1]. 이러한 특징 때문에 플래시 메모리는 소프트웨어를 업데이트하기 위해 논리 시스템에 저장될 수 있어 SoC(System On Chip) 환경에서 중요한 역할을 담당하고 있다.

플래시 메모리를 생산하는데 있어 최소한의 시간과 최대한의 수율 보장하기 위해 본 논문에서는 플래시 메모

* 학생회원, ** 정회원, 숭실대학교 컴퓨터학과
(Computer, Soongsil University)

※ 이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R01-2006-000-11038-0)

접수일자: 2006년11월17일, 수정완료일: 2007년4월11일

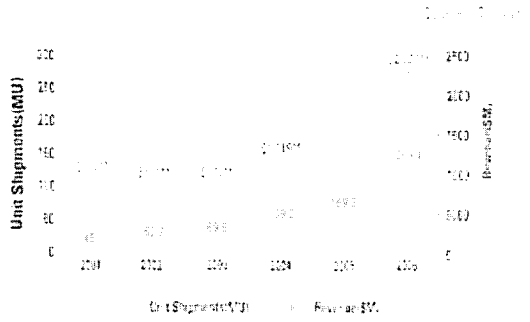


그림 1. 플래시 메모리의 동향
Fig. 1. A Tendency of Flash Memory.

리를 테스트하기 위해 사용자가 테스트 알고리즘을 선택하게 되면 테스트 벡터들이 제안된 FSM(Finite State Machine)에 의해 자동적으로 테스트 알고리즘을 생성해 주는 구조를 갖는 프로그래머블 BIST(Built-In Self Test) 구조를 제안하고자 한다.

본 논문의 구성은 다음과 같다. Ⅱ장 기존연구에서는 플래시 메모리에 존재하는 고장의 종류와 이런 고장을 검출하는 테스트 알고리즘에 대해서 언급한다. Ⅲ장은 제안된 FSM 기반의 플래시 메모리 BIST 설계와 구조에 대해서 설명한다. Ⅳ장은 제안한 FSM 기반의 플래시 메모리 BIST 방법의 효율성 실험 결과를 통해 검증하고, 마지막으로 Ⅴ장 결론으로 본 논문을 마친다.

Ⅱ. 본 론

1. 기존연구

플래시 메모리에서 발생되는 고장들을 검출하기 위한 테스트 알고리즘들이 기존에 제안되어져 있다[3]-[7]. 플래시 메모리는 기존의 일반적인 메모리들의 고장과는 다르게 고유하게 발생하는 외란(Disturbances)이 있다. 외란의 종류로는 Disturbances- Programming (DP)와 Disturbances-Erasure(DE), Drain Disturbances (DD)들이 있다. 이 세 가지의 외란 종류들은 플래시 메모리 셀에 프로그램(Program : '0' 인가)하거나 지우(Erasure : '1' 인가)는 과정에서 컨트롤 게이트(Control Gate)와 드래인(Drain)에 높은 전압이 가해짐으로써 발생하는 고장들이다.

가. 고장의 종류

프로그램하거나 지우는 동안에 플래시 메모리는 외란이나 고장을 겪게 된다. 외란 이외의 고장들은 일반적인 메모리 테스트에서 사용하는 알고리즘을 가지고도

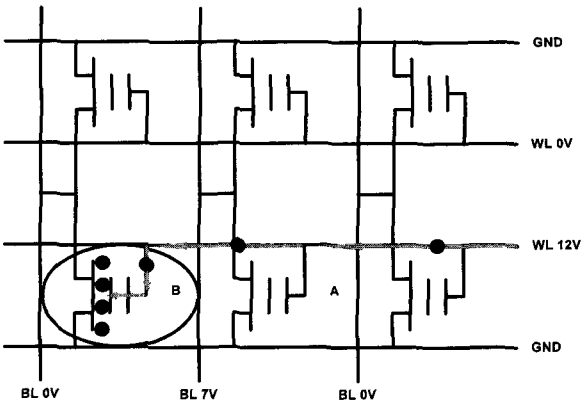


그림 2. 외란 지우기(DE)
Fig. 2. Disturbances-Erasure(DE).

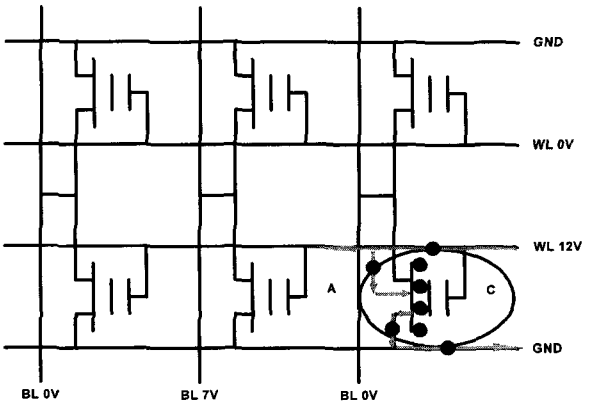


그림 3. 외란 프로그램(DP)
Fig. 3. Disturbances-Programming(DP).

플래시 메모리를 테스트해 고장을 검출할 수 있다. 하지만 외란 고장의 경우는 고장 검출이 불가능하다. 따라서 본 논문에서는 일반적인 메모리 고장과 함께 외란 고장도 검출할 수 있는 알고리즘을 소개하며, 이 알고리즘을 FSM 기반으로 적용할 수 있는 구조를 제안하고자 한다.

그림 2는 외란 고장 중에 DE(Disturbances-Erasure)를 표현한 그림이다. A라는 곳에 프로그램하기 위해 워드 라인(WL : Word Line)에 12V 전압를 가한다. 그러면 같은 워드 라인에 있는 다른 트랜지스터에도 12V 전압이 가해지게 된다. 그 결과로 프로그램된 B의 셀에 전압이 가해지게 된다. 이로 인해 B의 트랜지스터는 '1'이 되는 현상이 발생한다. 즉, 프로그램된 셀이 지워져 버리게 된다.

그림 3은 DP(Disturbances-Programming)를 나타낸 그림으로써, A라는 곳에 프로그램하기 위해 WL에 12V 전압를 가한다. 이로 인하여 기판으로 전하가 흘러 근처의 트랜지스터에 영향을 끼치게 된다.

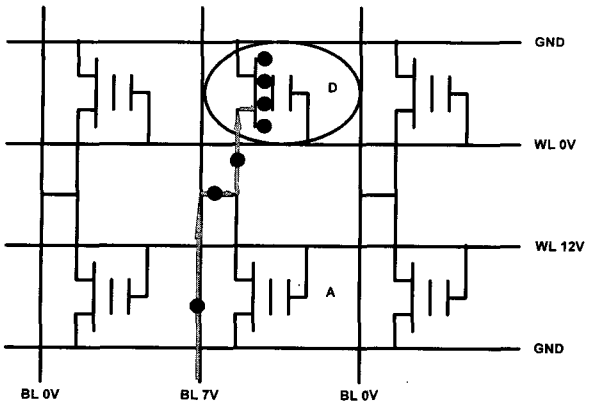


그림 4. 드래인 외란(DD)
Fig. 4. Drain-Disturbances(DD).

그 결과 지워진 C 셀에 있는 트랜지스터에 영향을 끼치게 되어 C의 트랜지스터로 이동한 전하들은 소스 라인 쪽으로 흘러 '0'이 되는 현상이 발생한다. 즉, 지워진 셀이 프로그램된다.

그림 4는 DD(Drain-Disturbances)의 그림이다. A라는 곳을 프로그램하기 위해 WL과 비트 라인(BL : Bit Line)에 12V 전압을 동시에 가한다. 그 때, 위의 두 경우와는 다르게 BL에 12V 전압이 걸리므로 생기는 고장이다. BL에 전압이 가해지면 같은 BL에 있는 다른 셀에도 영향을 끼치게 된다.

그 결과 프로그램된 D의 셀에 있는 트랜지스터에 영향을 끼치게 되고 그에 따라 전하가 D의 트랜지스터에 이동해 '1'이 되는 현상이 나타나게 된다. 즉, 프로그램된 셀이 지워져 버린다.

나. 고장 테스트 알고리즘

기존에 제안되어 졌던 플래시 메모리의 외란을 검출하는 알고리즘들 중 [3]에서 제안된 March EF 알고리즘은 복잡성이 적은 테스트 알고리즘으로써 플래시 메모리의 외란을 검출한다. [4]-[6]에서는 일반적인 메모리 알고리즘인 MATS++ 가지고 플래시 메모리 테스트를 하기 위해 변형시킨 알고리즘을 제안했다.

표 1. 플래시 메모리 테스트 알고리즘의 고장 검출률
Table 1. Fault Coverage of Flash Memory Test Algorithm.

	SAF	TF	SOF	AF	CFst	WPD	WED	BPD	BED	RD	OE
Test1[5]	1	0.5	0.002	0.5	0.5	0.5	0.5	0.5	0.5	0	1
Test2[5]	1	0.5	1	0.5	0.75	0.5	0.5	0.5	0.5	1	1
Test3[5]	1	1	1	1	0.88	1	0.5	1	0.5	1	1
EF[3]	1	0.934	0.001	0.06	0.5	1	1	0	1	0	1
Flash-March[6]	1	1	0.002	1	0.75	1	1	1	1	0	1
Diagonal-FT[7]	1	1	1	0.81	0.89	1	1	1	1	1	1
March-FT[5,6]	1	1	1	1	1	1	1	1	1	1	1

[7]은 플래시 메모리를 진단하고 테스트하기 위해 향상된 March-like 알고리즘을 제안하고 있다. 아래 표 1에서는 플래시 메모리를 테스트하기 위한 알고리즘으로 March-FT가 가장 고장 검출률이 높은 것을 볼 수 있다[5], [7]-[9].

표 1에서 확인 할 수 있듯이 고장 검출률이 100%인 March-FT를 이용해 FSM 기반인 메모리 자체 테스트 회로를 구현하며 사용자에 요구에 맞게 다른 알고리즘을 쉽게 적용할 수 있도록 프로그래머블한 플래시 메모리 내장 자체 테스트 구조를 제안하고자 한다.

2. 제안하는 FSM 기반의 프로그래머블 BIST

가. 기호법

셀에 고장이 있는지 검사하기 위해 읽기와 쓰기 동작의 순차적인 수행이 필요하다. 플래시 메모리를 테스트하는 알고리즘들은 여러 가지가 있으며, 그에 대한 명령어들은 E, R0, R1, P로 구성된다[5], [7]-[9]. E는 모든 셀을 '1'로 초기화하며, 모들 셀을 '1'로 초기화한다는 의미를 가지고 있다. R0은 셀의 '0' 데이터를 읽는다. R1은 데이터 '1'을 읽고, P는 해당 셀에 0을 쓰는 동작을 한다. 주소의 변화는 주소의 증가 또는 감소의 명령어로 이루어지며, 그 기호법은 표 2에 요약되어져 있다.

플래시 메모리를 테스트하는데 주로 사용되는 알고리즘들은 표 3에 제시되어 있다. March-FT 알고리즘을 구성한다고 하면 첫째로 모든 셀을 1로 초기화 한다. 그리고 주소를 증가시켜 R1과 P, R0을 차례로 수행한 후 주소의 증•감에 관계없이 R0을 한다.

세 번째는 다시 모든 셀을 1로 초기화하고 두 번째 수

표 2. 기호법의 요약
Table 2. A Summary of Notation.

E	셀에 1을 씀
R	Read 동작
P	Write 동작
↑	주소 증가
↓	주소 감소
↕	양 방향 모두 가능

표 3. 플래시 메모리 테스트 알고리즘
Table 3. Flash Memory Test Algorithm.

Algorithm	Testing sequence
test1[5]	E;↑(R1, P);↓(R0, P);
test2[5]	E;↑(R1);↑(R1, P, R0);↓(R0, P);
test3[5]	E;↑(R1, P); E;↓(R1, P, R0),↑(R0);
Flash-March[6]	E;↑(R1, P);↕(R0); E;↓(R1, P);↕(R0);
Diagonal-FT[7]	E;↑(R1);↑(R1, P ,R0); E;↑(R1);↓(R1, P, R0, R0);↑(R0, P);↑(R0);
March-FT[5,6]	E;↑(R1, P, R0);↕(R0); E;↓(R1, P, R0);↕(R0);

행했던 동작을 이번에는 주소를 감소시키면서 수행한다. 마지막으로 주소의 증•감에 관계없이 R0을 수행하여 March-FT 알고리즘을 마치게 된다.

본 논문에서는 표 3으로부터 서로 다른 테스트 알고리즘들이 같은 구성원들을 가지고 있다는 것을 알 수 있다. 이 공통된 부분을 가지고 융통성 있는 BIST 컨트롤러를 설계했다.

나. FSM 기반의 프로그래머블 BIST 설계

6개의 알고리즘들은 3비트의 A[2:0]으로 부호화한다. BS(BIST Start) 신호에 의해 5개의 테스트 알고리즘 중에서 하나를 선택하게 되면 그에 따른 신호로 부호화 비트 A[2:0]가 전달된다. 표 4로부터 알 수 있듯이 test1과 test2는 2개의 서로 다른 구성원이 있다는 것을 확인 할 수 있다. test1에서는 (R1, P)이고 test2에서는 (R1), (R1, P, R0)이다. 그에 해당하는 구성들은 표 5에서 보는 것처럼 E[3:0]의 4비트로 부호화된다.

그림 5는 테스트 알고리즘 생성기에서 선택된 알고리즘에 대한 구성들이 어떻게 구성되는지에 대해 설명하는 그림이다. 예를 들어 test1은 알고리즘을 수행을 위해 A[2:0]를 '001'로 부호화 한다. 테스트 알고리즘 생성기는 BS 신호와 A의 값을 받아 test1에 대한 수행을 시작한다.

test1의 알고리즘은 E0, E3, E4의 구성들로 구성된다. BS=1이면 idle 상태에서 E0 상태로 이동한다. E0 상태의 수행을 완료한 후 조건에 의해 E3로 이동한다. E3은 읽기/쓰기 생성기의 수행이 끝났다는 MF 신호를 받고 나서 조건에 의해 E4의 상태로 이동한다. 마찬가지로 E4에서도 읽기/쓰기 생성기의 수행이 끝났다는 MF 신호를 받으면 해당 알고리즘을 수행을 완료하였으므로 조건에 의해 처음 상태인 idle의 상태로 이동한다. 여기서 주목할 점은 각 단계는 알고리즘을 구성하는데 있어서 해당 조건이 주어지는데 이 조건에 따라 구성원들의 상태 전이가 발생하게 된다.

그림 6은 읽기/쓰기 생성기의 각 구성원에 대한 읽기와 쓰기가 어떻게 구성되는지 보여주고 있다. 예를 들어 E5 수행을 위해 E[3:0]은 '0110'으로 부호화된다면. 최상위 1비트는 주소 증•감을 나타내기 위해 사용 하고, 하위 3비트는 부호화 신호를 가지고 E5를 수행하게 된다.

읽기/쓰기 생성기는 MF 신호와 E의 값을 받아 E5에 대한 수행을 시작하게 된다. E5는 각 메모리 셀에 3개의 동작(R1, P, R0)을 수행한다. MF가 0이고 E가 '110'이면 idle 상태에서 R1 상태로 이동한다. R1 상태에서는 조건에 의해 P 상태로 이동한다. 마지막으로 R0의 상태로 이동하면, 주소 생성기로부터 End_A 신호를 받으면 다른 알고리즘을 수행하기 위해 테스트 알고리즘

표 4. 테스트 알고리즘과 부호화
Table 4. Test Algorithm and Encoding.

Algorithm	Code	Testing sequence
test1[5]	001	E; ↑(R1, P); ↓(R0, P);
test2[5]	010	E; ↑(R1); ↑(R1, P, R0); ↓(R0, P);
test3[5]	011	E; ↑(R1, P); E; ↓(R1, P, R0); ↑(R0);
Flash-March[4]	100	E; ↑(R1, P); ↑(R0); E; ↓(R1, P); ↕(R0);
Diagonal-FT[7]	101	E; ↑(R1); ↑(R1, P ,R0); E; ↑(R1); ↓(R1, P, R0, R0); ↑(R0, P); ↑(R0);
March-FT[5,6]	110	E; ↑(R1, P, R0); ↕(R0); E; ↓(R1, P, R0); ↕(R0);

표 5. 플래시 구성원과 그에 대한 부호화
Table 5. Flash Elements and Encoding.

Flash element	Code	Operation
E0	0001	E
E1	0010	R0
E2	0011	R1
E3	0100	R1, P
E4	0101	R0, P
E5	0110	R1, P, R0
E6	0111	R1, P, R0, R0

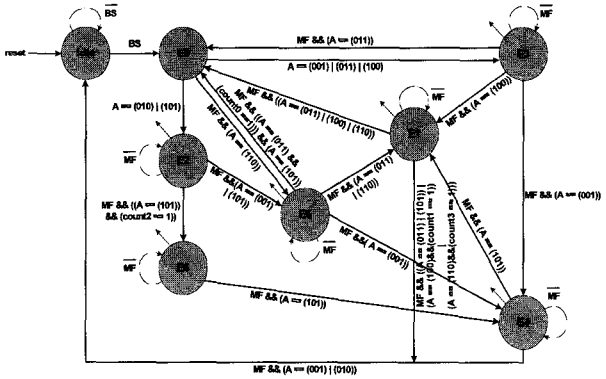


그림 5. 테스트 알고리즘 생성기의 FSM
Fig. 5. FSM of Test Algorithm generator.

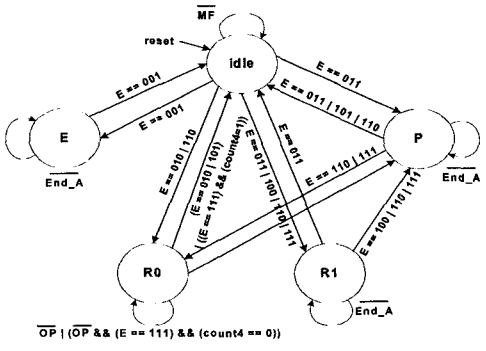


그림 6. 읽기/쓰기 생성기의 FSM
Fig. 6. FSM of Read/Write generator.

생성기에게 MF 신호를 보내고 idle 상태로 돌아온다.

본 논문에서 제안하는 방법은 테스트 알고리즘을 선택하면 A라는 신호로 부호화하여 전반적인 플래시 메모리에 대한 BIST를 수행한다. 테스트 알고리즘 생성기는 A의 신호를 받아 스스로 해당 알고리즘의 구성원을 구성하고 읽기/쓰기 생성기는 해당 구성원의 실질적인 플래시 메모리 테스트를 수행하기 위한 명령어들을 구성한다.

다. FSM 기반의 프로그래머블 BIST 구조

제안하는 FSM 기반의 프로그래머블 BIST 구조는 그림 7에서 보는 것처럼 크게 세부분으로 구성되어 진다.

테스트 알고리즘 생성기는 사용자가 플래시 메모리를 테스트하기 위해 선택한 알고리즘에 따라 그에 해당하는 구성원들을 구성하는 작업을 수행한다. 읽기/쓰기 생성기는 테스트 알고리즘 생성기에서 선택된 구성원들 가운데서 각 구성원에 대한 플래시 메모리를 실질적으로 테스트할 명령어들을 생성한다.

주소 생성기는 해당 구성원 명령어를 모든 셀에 수행할 수 있도록 플래시 메모리의 주소를 증•감하는 역할을 수행한다. 플래시 메모리 전체에 수행을 완료하게 되면 테스트 알고리즘 생성기에 MF_End 신호를 보내고 테스트 알고리즘 생성기는 모든 구성원들의 수행이 끝나면 BIST가 끝났다는 의미로 BIST_End 신호를 보내고 테스트를 종료한다.

테스트 알고리즘 생성기는 선택된 알고리즘에 대한 구성원들을 구성하는 모듈이다. 이 모듈은 주어진 A(Algorithm) 신호의 비트 값을 분석하여 그에 해당하는 절차에 따라 테스트 알고리즘을 구성한다. 해당 알고리즘에 대한 구성원들은 차례로 읽기/쓰기 생성기에

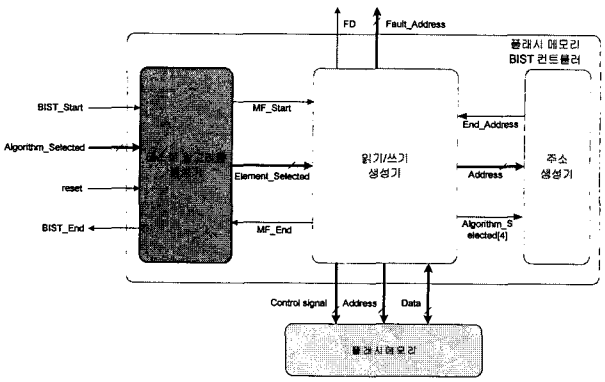


그림 7. 제안하는 FSM 기반의 프로그래머블 BIST 구조
Fig. 7. Proposed FSM-based of Programmable BIST Architecture.

E(Element)와 MF_Start(MS) 신호를 전달하여 해당 구성원의 명령어를 구성하도록 지시한다.

읽기/쓰기 생성기는 해당 구성원에 대한 명령어 동작을 구성하는 모듈이다. 읽기/쓰기 생성기는 테스트 알고리즘 생성기의 MS 신호를 받아서 동작하고, E 신호의 비트 값을 분석하여 그에 따른 절차에 따라 해당 구성원에 대한 읽기, 프로그램과 지우기 명령어를 구성한다. 생성된 명령어들을 플래시 메모리의 주어진 주소에 해당 명령어들을 수행하고 데이터를 읽기는 과정에서 해당 플래시 메모리에 고장이 존재하는지 여부를 파악할 수 있다. 만약, 고장이 발생하게 되면 고장난 곳의 주소와 FD(Fault Detection) 신호를 발생한다.

주소 생성기는 읽기/쓰기 생성기에서 구성된 구성원의 명령어들이 플래시 메모리의 전체에 실행될 수 있도록 주소를 증•감해주는 모듈이다. 주소 생성기는 E의 최상위 1비트 값을 보고 주소를 증•감할 것인지 결정한다. 그리고 주소 증•감을 실행하다가 주소가 맨 처음이나 마지막에 도달하게 되면 End_A라는 신호를 읽기/쓰기 생성기에 보내 다른 구성원에 대한 명령어를 수행할 수 있도록 한다.

Ⅲ. 실 험

본 논문에서 제안한 FSM 기반의 프로그래머블 BIST 구조의 효율성 검증은 Verilog HDL로 기술하여 구현하였다. 구현에 대한 검증은 Xilinx사의 Xilinx Foundation에서 제공하는 시뮬레이터를 사용하여 RTL 검증을 하였다. 아래의 그림 8은 표 3에 있는 플래시 메모리 테스트 알고리즘들 중 test1[5]를 수행한 정보를 출력해 주는 파형의 일부이다.

그림 8의 (1)는 clk신호이다. (2)는 사용자가 선택한 알고리즘의 부호화한 신호(001)이며 (3)는 선택한 알고리즘을 구성하는 구성원들을 수행하라는 신호이다. (4)는 구성원을 부호화(1001)한 4비트 신호로서 (5)는 (4)의 최상위 1비트의 값으로 주소의 증•감을 나타내고, (6)는 (4)의 3비트까지의 값으로 해당 구성원을 구성하는 읽기, 쓰기, 지우기 동작들의 부호화한 것이다. (7)~(13)은 플래시 메모리의 컨트롤 신호들이다. (14)는 플래시 메모리의 주소이며 (15)는 해당 구성원의 수행을 완료했을 때 '1'의 값을 갖는다. (16)와 (17)는 고장이 발생 시 고장난 주소와 고장이 발생했다는 것으로 FD 신호가 '1'로 된다. (18)는 선택된 알고리즘을 완벽하게 수행한 후에 '1'의 값으로 인가되어 해당 알고리즘을 완

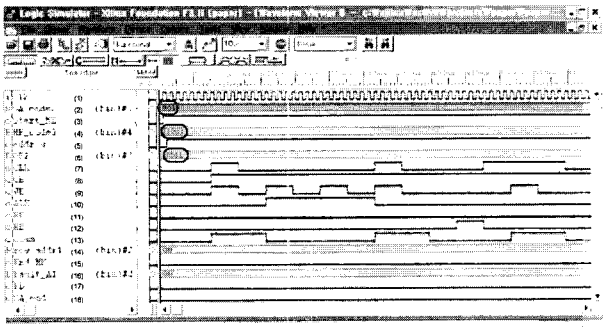


그림 8. test1에 대한 제안한 FSM 기반의 프로그래머블 BIST의 수행 결과

Fig. 8. performance result of Proposed FSM based of Programmable BIST to test1.

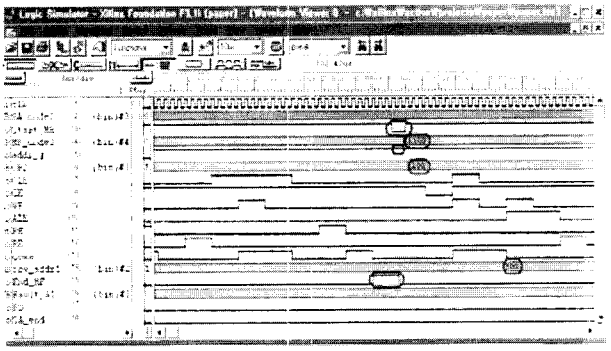


그림 9. test1에 대한 구성원 교체 과정

Fig. 9. element change process to test1.

료했다는 것을 나타낸다.

test1[5]는 E -> R1, P -> R0, P의 순서로 테스트가 된다. 그림 9는 두 번째 단계(R1, P)를 수행하는 과정이다. 첫 번째 단계(E)에 대한 수행을 완료했다는 표시로 End_MF 신호가 '1'의 값을 가지고 나서 Start_ME 신호가 '0'의 값으로 된다. 두 번째 단계를 수행하기 위해 Start_ME 신호가 '1'의 값을 가지고, R1, P에 해당하는 E[3:0]인 '1100'으로 MF_code 값이 바뀐다. 그리고 addr_g인 E의 최상위 비트 한 비트와 OP인 E의 하위 3비트 값으로 변경되고 우선 R1에 대한 컨트롤 신호를 생성한다.

그림 10은 test1[5]를 끝까지 수행한 결과에 대한 파형이다. 마지막 구성원인 E4의 수행을 마치면 Start_ME 신호를 '0'으로 인가한다. 그러나 E4가 test1[5]에 대한 마지막 구성원이기 때문에 테스트를 완료했다는 표시로 (18)인 SA_end 신호가 '1'로 인가되며, test1[5]의 알고리즘을 다 수행했기 때문에 컨트롤 신호에 대한 초기화가 수행된다.

그림 11은 test1[5] 수행 중 고장이 발생한 경우이다. 두 번째 구성원인 E3을 수행하던 중에 플래시 메모리

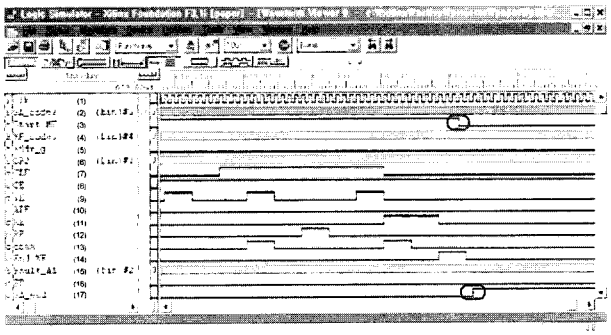


그림 10. test1을 끝까지 수행한 결과

Fig. 10. result after performance completed test1.

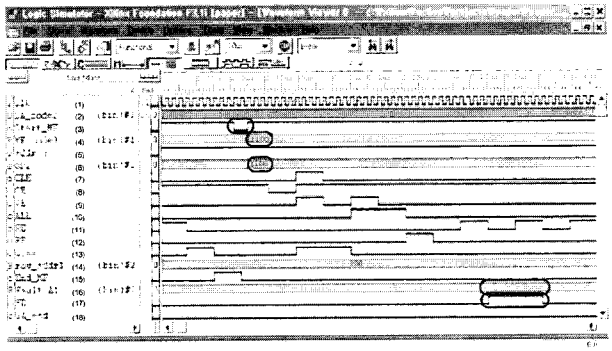


그림 11. test1 수행 중 고장 발생

Fig. 11. fault generation among test1 performance.

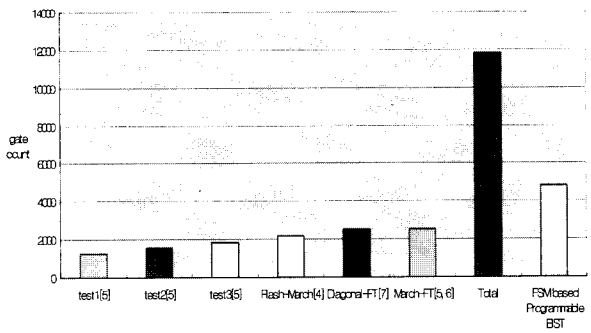


그림 12. 제안한 FSM 기반의 프로그래머블 BIST 게이트 수 결과

Fig. 12. Proposed FSM based of Programmable BIST gate count result.

에서 R1을 읽는 과정에서 읽어 들인 데이터가 '1'이 아니라는 것을 발견하고 FD 신호를 '1'으로 인가한 후, 고장이 발생한 메모리 주소인 00번지 주소를 Fault_A로 출력한다. 현재 Fault_A가 00으로 되어 있기 때문에 그림 11에서는 변화가 없다.

그림 12와 그림 13은 크기가 2MB인 플래시 메모리를 기준으로 측정한 게이트 수와 하드웨어 오버헤드의 수치들을 나타낸다. 그림 12는 기존에 제안한 6개의 테스트 알고리즘들과 본 논문에서 제안한 방법으로

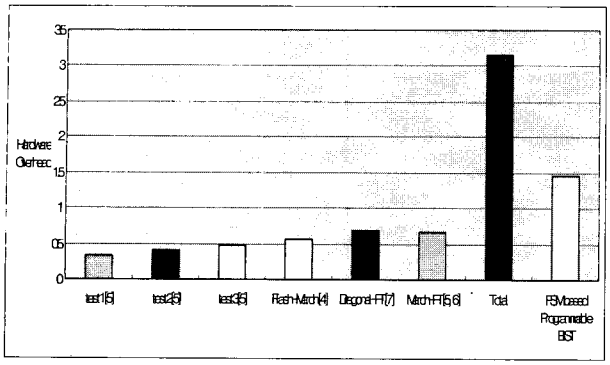


그림 13. 제안한 FSM 기반의 프로그래머블 BIST 하드웨어 오버헤드 결과

Fig. 13. Proposed FSM based of Programmable BIST Hardware Overhead result.

Xilinx FPGA툴을 이용하여 해당 알고리즘을 수행한 후의 게이트 수이다. total은 제시한 6개의 테스트 알고리즘들의 게이트 수를 합친 것이다. 그림에서 확인할 수 있듯이 제안한 방법이 약 3배의 효율성이 좋다는 것을 알 수 있다.

그림 13은 기존에 제안했던 6개의 테스트 알고리즘들과 본 논문에서 제안한 방법으로 Xilinx FPGA툴을 이용하여 해당 알고리즘을 수행한 후 하드웨어 오버헤드 수치를 나타낸 그림이다. total은 위에서와 마찬가지로 제시한 6개의 테스트 알고리즘들의 하드웨어 오버헤드 수치를 합친 것이다. 이것과 제안한 방법을 비교해 보면 제안한 방법이 약 2배의 효율성이 좋다는 것을 알 수 있다.

그리고 제시한 각 6개의 테스트 알고리즘을 각기 수행하기 위해서는 재정립시간도 많이 걸린다. 반면 본 논문이 제안하는 방법을 이용하면 재정립 시간이 많이 단축할 수 있다. 본 논문에서 제안하는 FSM 기반의 프로그래머블 BIST의 게이트 수와 하드웨어 오버헤드 수치를 각각 비교해 보았을 때 본 논문이 제안한 방식이 더 효율적이라는 것을 알 수 있다.

IV. 결 론

본 논문에서 제안한 FSM 기반의 프로그래머블 BIST 구조는 사용자들이 테스트 알고리즘만 선택하면 자동적으로 해당 알고리즘을 구성하여 BIST를 수행하고 결과를 알려준다. 기존의 논문들은 단일 알고리즘이나 고장 종류에 따른 분류를 통하여 플래시 메모리 BIST를 수행하고 있다[8~9].

본 논문이 제안한 FSM 기반의 프로그래머블 BIST 구조는 단일 테스트보다 새로운 테스트 알고리즘에 대한 재구성 절차도 매우 간단할 뿐만 아니라 그에 따른 전체 테스트 시간도 감소되므로 기존의 단일 플래시 메모리 BIST 보다 효율성면에서 더 높다. 그러나 단일 플래시 메모리 BIST 보다는 복잡성과 오버헤드가 더 증가한다는 단점이 있다. 이 문제는 실직적인 테스트 시간과 전체 테스트 알고리즘이 포함된 오버헤드를 분석해보면 그렇지 않다는 것을 알 수 있으므로 결과적으로 본 논문이 제안한 구조가 더 좋다는 것을 알 수 있다.

본 논문에서 제안한 FSM 기반의 프로그래머블 BIST가 실제 실험 결과에서 얼마만큼의 효율성을 보이는가에 대한 테스트를 수행하였다. 테스트 메모리로는 임베디드용 메모리인 삼성의 2MB NAND 플래시 메모리를 사용하였다. 그리고 테스트를 위해 Xilinx Foundation 3.1i 툴을 사용하여 synthesis를 거쳐 시뮬레이션을 수행하였으며 그 결과로 시뮬레이션을 통해 각각의 알고리즘 총 게이트 수와 하드웨어 오버헤드의 수치를 비교하여 얼마나 효율성이 좋은지를 비교 분석하였다.

단일 플래시 메모리 테스트 알고리즘들로 플래시 메모리에 여러 테스트 알고리즘들을 적용하기 위해서는 매번 해당 알고리즘에 대한 재정립의 시간과 코드의 수정이 필요했으나 본 논문에서 제안하는 FSM 기반의 프로그래머블 BIST는 그럴 필요가 없기 때문에 본 방법을 이용하면 재정립에 대한 시간 감소와 코드의 수정이 필요하지 않다는 것을 알 수 있다. 더 나아가 새로운 테스트 알고리즘을 추가하더라도 코드에 약간의 수정만 하면 그 테스트 알고리즘도 수행할 수 있다.

참 고 문 헌

[1] P. Pavan, R. Bex, D. Olivo, and E. Zanon, "Flash memory cells- an overview," in Proceeding of IEEE, vol.85, pp. 1248-1271, August 1997.

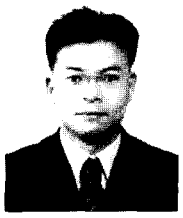
[2] A. J. van de Goor, John Wiley & Sons Chichester, "Testing Semiconductor Memories: Theory and Practice", in Proceeding of England, 1991.

[3] M. G. Mohammad, K. K. Saluja, and A. Yap, "Testing flash memories," VLSI Design 13th, pp. 406-41, January 2000.

[4] M. Mohammad and K. K. Saluja, "Flash memory disturbances: modeling and test," in Proceeding IEEE VLSI Test Symp.(VTS) on Monterey California, pp. 218-22, April 2001.

[5] K. -L. Cheng, J. -C. Yeh, C. -W. Wang, C. -T. Huang, and C. -W. Wu, "RAMSES-FT: A Fault Simulator for Flash Memory Testing and Diagnostics", in Proceeding IEEE VLSI Test Symp.(VTS) on Monterey California, pp. 281-286, April 2002.
[6] J. -C. Yeh, C. -F. Wu, K. -L. Cheng, C. -W. Wu, "Flash memory built-in self-test using march-like algorithms," in Proceeding IEEE International Workshop on Electronic Design, Test, and Applications (DELTA), Christchurch, pp. 137-141, January 2002.
[7] S. -K. Chiu, J. -C. Yeh, C. -T. Huang, and C. -W. Wu, "Diagonal test and diagnostic schemes for flash memories," in Proceeding International Test Conference(ITC) on Baltmore, pp. 37-46, October 2002.
[8] J. -C. Yeh, Y. -T. Lai, Y. -Y. Shih, and C. -W. Wu, C. -H. Ho, Y. -T. Lin, "Flash memory built-in self-diagnosis with test mode control", VLSI Test Symposium in Proceedings 23rd IEEE, pp. 15-20, May 2005.
[9] Banerjee S, Chowdhury D. R, "Built-in self-test for flash memory embedded in SoC", Electronic Design, Test and Applications in DELTA 2006, January 2006.

저 자 소 개



김 지 환(학생회원)
2005년 용인대학교 전산통계학과 학사 졸업.
2007년 숭실대학교 컴퓨터학과 석사 졸업.
<주관심분야 : VLSI 설계 및 테스트, 컴퓨터구조, 임베디드>

장 훈(정회원)
1987년 서울대학교 공대 전자공학과 학사 졸업.
1989년 서울대학교 공대 전자공학과 석사 졸업.
1993년 University of Texas at Austin 졸업.
1991년 IBM Inc. Senior Member of Technical Staff.
1993년 Motorola Inc. Senior Member of Technical Staff.
1994년~현재 숭실대학교 컴퓨터학부 부교수.
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>