논문 2012-49SD-8-9

이중 포트 메모리를 위한 효율적인 프로그램 가능한 메모리 BIST

(An Efficient Programmable Memory BIST for Dual-Port Memories)

박 영 규*, 한 태 우*, 강 성 호**

(Youngkyu Park, Taewoo Han, and Sungho Kang)

요 약

메모리 설계 기술과 공정 기술의 발달은 고집적 메모리의 생산을 가능하게 하였다. 전체 Systems-On-Chips(SoC)에서 내장 메모리가 차지하는 비중은 점점 증가하여 전체 트랜지스터 수의 80%~90%를 차지하고 있어, SoC에서 내장된 이중 포트 메모리에 대한 테스트 중요성이 점점 증가하고 있다. 본 논문에서는 이중 포트 메모리를 위한 다양한 테스트 알고리즘을 지원하는 새로운 micro-code 기반의 programmable memory Built-In Self-Test(PMBIST) 구조를 제안한다. 또한 제안하는 알고리즘 명령어 구조는 March 기반 알고리즘과 이중 포트 메모리 테스트 알고리즘 등의 다양한 알고리즘을 효과적으로 구현한다. PMBIST는 테스트 알고리즘을 최적화된 알고리즘 명령어를 사용하여 최소의 bit으로 구현할 수 있어 최적의 하드웨어 오버헤드를 가진다.

Abstract

The development of memory design and process technology enabled the production of high density memory. As the weight of embedded memory within aggregate Systems-On-Chips(SoC) gradually increases to 80-90% of the number of total transistors, the importance of testing embedded dual-port memories in SoC increases. This paper proposes a new micro-code based programmable memory Built-In Self-Test(PMBIST) architecture for dual-port memories that support test various test algorithms. In addition, various test algorithms including March based algorithms and dual-port memory test algorithms are efficiently programmed through the proposed algorithm instruction set. This PMBIST has an optimized hardware overhead, since test algorithm can be implemented with the minimum bits by the optimized algorithm instructions.

Keywords : Memory BIST, Dual-Port Memory, Test Algorithm, Micro-code

Ⅰ. 서 론

최근 반도체 공정 기술과 설계 기술이 발달함에 따라 많은 수의 코어들이 시스템 온 칩(SoC:System-On-Chip)화 되고 있다. 전체 SoC에서 내장 메모리가 차지하는 비중은 점점 증가하여 전체 트랜지스터 수의 80%~90%를 차지해 SoC에서 내장 메모리에 대한 테스트 중요성이 점점 증가하고 있다. 이중 포트 메모리는 두 개의 포트로부터 동시에 데이터의 입출력이 가능하여 많은 양의이중 포트 메모리는 두 개의 포트로부터 동시에 데이터의 입출력이 가능하여 많은 양의 데이터를 빠른 시간에 처리할 수 있는 장점을 가져 최근 많이 사용되고 있다. 이중 포트 메모리의 경우, 단일 포트 메모리의 고장 테스트뿐만 아니라, 두 포트를 통해 동시

* 학생회원, ** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
※ 본 연구는 지식경제부 및 정보통신산업진흥원의 IT 융합 고급인력과정 지원사업의 연구결과로 수행되었음(NIPA-2012-H0401-12-1006)
접수일자: 2012년4월18일, 수정완료일: 2012년7월5일

접근을 통해 발생하는 고장 테스트를 고려해야 한다. 이는 단일 포트 메모리를 위한 March 기반의 테스트 알고리즘 이외에 이중 포트 메모리에서만 발생하는 두 포트의 접근에 의해 발생하는 고장 검출을 위한 이중 포트 메모리 테스트 알고리즘을 사용하여 테스트를 수행해야 한다[1].

현재 내장 메모리를 테스트하기 위해 가장 많이 사용되는 방법은 메모리 BIST(Built-In Self-Test)기법이다[2]. 메모리 BIST 기법은 각 모듈 별로 자체적인 테스트가 수행되므로 전체시스템의 테스트에 있어서 테스트의 복잡도가 크게 줄어들고, 고가의 외부 테스트 장비를 사용하지 않고도 빠른 시간에 테스트를 수행할 수 있다는 장점을 가진다[3]. 그러나 메모리 BIST 기법은 제한적인 테스트 알고리즘만을 지원하여, flexibility가 낮고 고장 검출률이 제한적이라는 단점을 가진다. 그래서 기존의 메모리 BIST의 단점을 보완한 PMBIST (Programmable Memory BIST) 기법이 제안되었다[4]. PMBIST는 다양한 테스트 알고리즘을 지원하여 flexibility를 향상시켰으나, 지원하는 테스트 알고리즘이 March 기반 알고리즘(MATS+, March C 등)에만 국한되어 flexibility가 제한적이고 하드웨어 오버헤드가 크다는 단점을 가진다. 데이터를 빠른 시간에 처리할 수 있는 장점을 가져 최근 많이 사용되고 있다.

본 논문에서는 다양한 내장된 이중 포트 메모리를 효율적으로 테스트하기 위해 다양한 테스트 알고리즘의 적용이 가능한 새로운 알고리즘 명령어 구조와 PMBIST 구조를 제안한다. 일반적으로 많이 사용되는 March 기반의 테스트 알고리즘을 모두 지원하고, 이중 포트 메모리 테스트 알고리즘을 지원하는 PMBIST 구조를 제안한다. 따라서 제안하는 PMBIST는 flexibility와 고장 검출률이 높아 내장된 이중 포트 메모리의 신뢰성을 높일 수 있으며, 다양한 테스트 패턴을 생성하기 위한 최적의 알고리즘 명령어 구조와 주소 생성기를 통하여 하드웨어 오버헤드를 최소화하였다.

Ⅱ. 제안하는 Programmable Memory BIST

본 논문에서 제안하는 Programmable Memory BIST (PMBIST)는 다양한 테스트 알고리즘 지원이 가능한 micro code 방식의 프로그램 가능한 BIST 구조를 가진다. 제안하는 PMBIST는 외부로부터 테스트 알고리즘을 명령어 형태로 입력받아 내부의 명령어 메모리에 저장하고, 이 명령어를 사용하여 알고리즘을 구현하는 구조이다. PMBIST는 MATS+, March C-, March A, March LR, March SS[5] 등의 모든 March 기반의 알고리즘을 지원하고, March 2PF1, March 2PF2av, March s2PF[6], March A2PF[7] 등의 모든 이중 포트 메모리 테스트 알고리즘을 지원한다. .

기존에 제안된 이중 포트 메모리 테스트를 위한 BIST들은 단일 포트 메모리 테스트를 위한 March 알고리즘을 사용하여 한 포트로는 쓰기 동작, 다른 포트로는 읽기 동작을 수행하여 테스트를 하였다. 그리고 두 포트를 통하여 동일한 셀에 동시에 접근하여 쓰기와 읽기 동작을 수행하는 BIST도 제안되었지만, 이중 포트 관련 고장을 모두 검출하지는 못하였다. 제안하는 PMBIST는 기존 BIST들의 모든 동작이 가능하며, 동시에 두 포트를 통해 서로 다른 셀에 접근하는 동작이 가능하여 이중 포트 관련 모든 고장을 검출할 수 있다.

이중 포트 메모리에서 발생하는 고장의 90% 이상이 단일 포트 관련 고장이다. 그리고 두 포트와 관련하여 발생하는 고장은 10%정도 존재한다. 두 포트와 관련된 고장 검출이 가능하여야만 높은 신뢰성을 확보할 수 있다. 동시에 두 포트를 통해서 서로 다른 셀에 접근하여 발생하는 고장 모델을 2PF2av라고 한다[6]. 이 고장을 검출하기 위해서는 PMBIST가 동시에 두 포트를 통해서 서로 다른 셀에 접근하는 동작을 지원해야 한다.

따라서 제안하는 PMBIST는 모든 March 기반 알고리즘과 이중 포트 메모리 테스트 알고리즘을 지원하여 높은 고장 검출률을 확보하였다. 그리고 제안된 알고리즘 명령어 구조는 다양한 테스트 패턴을 효과적으로 생성이 가능하며, 또한 최소의 bit으로 알고리즘을 구현하여 하드웨어 오버헤드를 최소화하였다.

1. 알고리즘 명령어 구조

제안하는 PMBIST는 테스트 알고리즘을 제안하는 알고리즘 명령어 구조로 내부의 명령어 메모리에 저장하고, 저장된 명령어를 사용하여 패턴을 생성한다. 제안하는 알고리즘 명령어 구조는 11 bits의 사이즈로 단일 포트 관련 고장 검출을 위한 March 기반 알고리즘과 두 포트와 관련된 고장을 검출하기 위한 이중 포트 메모리 테스트 알고리즘을 구현한다. 따라서 알고리즘 명령어 구조는 각 포트의 동작을 효율적으로 구현할 수

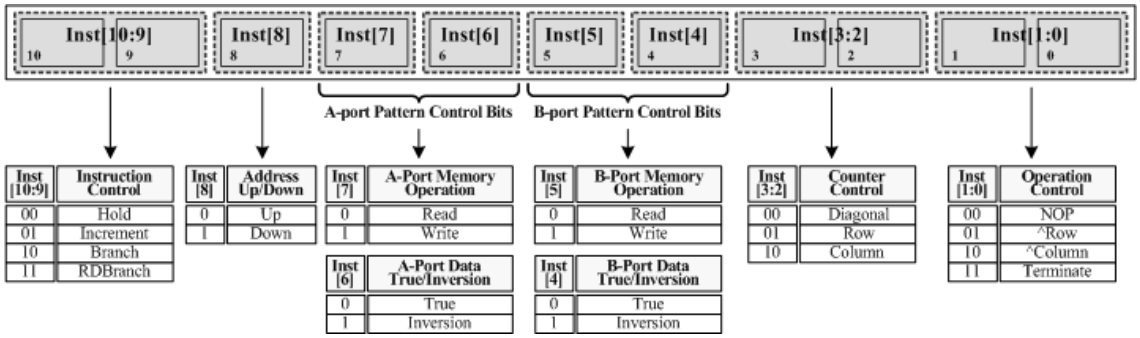


그림 1. 알고리즘 명령어 구조
Fig. 1. Algorithm instruction architecture.

있는 구조를 가진다. 알고리즘 명령어 구조는 March와 이중 포트 메모리 테스트 알고리즘을 최소의 bit으로 효과적으로 구현할 수 있다. 또한 테스트 알고리즘을 효과적으로 구현하기 위하여 멀티 루프를 지원한다. 그림 1은 알고리즘 명령어 구조를 보여주며, 각 bit을 살펴보면 다음과 같다.

Inst[10:9]는 명령어를 제어하는 부분으로 알고리즘을 구현하기 위한 명령어의 동작 상태를 지정한다. Hold는 현재의 명령어를 계속 실행하도록 하고, Increment는 현재의 명령어를 실행한 후 다음 명령어를 실행하도록 한다. Branch는 지정된 명령어로 점프를 하여 실행하도록 한다. RDBranch(Return Data inversion Branch)는 지정된 명령어로 점프를 하여 반전된 데이터 값으로 명령어를 다시 반복 실행하도록 한다. 명령어에서 Branch와 RDBranch의 경우에는 Branch 레지스터를 이용하여 점프를 한다. Branch 레지스터를 사용하여 멀티 루프를 지원하게 된다.

Inst[8]은 생성할 주소의 증/감을 제어하는 부분으로 March 알고리즘 및 이중 포트 메모리 테스트 알고리즘의 각 sequence의 주소 증/감 방향을 지정한다.

Inst[7]과 Inst[6]은 A 포트를 위해 생성되는 데이터 값을 제어하는 부분이다. Inst[7]은 A 포트에 적용될 데이터 패턴을 제어하는 부분으로 A 포트의 백그라운드 데이터를 '0'으로 쓸지 '1'로 쓸지를 결정한다. Inst[6]은 A 포트의 동작을 제어하는 부분으로 A 포트를 통해 데이터를 읽기 동작을 할지, 쓰기 동작을 할지 지정한다.

Inst[5]와 Inst[4]는 B 포트를 위해 생성되는 데이터 값을 제어하는 부분이다. B 포트에 적용될 패턴을 생성하기 위해 Inst[7], Inst[6]과 동일한 제어를 한다. 단일 포트 관련 고장 검출을 위한 알고리즘 구현시, Inst[5]와 Inst[4]는 '0'으로 고정되고 패턴 생성에 영향을 주지 않는다.

Inst[3:2]는 카운터를 제어하는 부분으로 주소 생성을 위해 fast diagonal, fast row 및 fast column 방식을 지원하기 위한 옵션이다. Inst[3:2]에서 Diagonal 옵션은 fast diagonal 방식으로 주소를 생성하며, Row와 Column 옵션은 fast row와 fast column 방식으로 주소 생성을 지원한다. Column은 column 주소가 끝까지 증가하며 연산을 하라는 옵션이고, Row는 row 주소가 끝까지 증가하며 연산을 하라는 옵션이다. 기존의 제안된 명령어 구조들은 대부분 fast diagonal만을 지원한다.

Inst[1:0]는 명령어 동작을 제어하는 부분으로 명령어의 옵션을 지정한다. 명령어 옵션은 주로 이중 포트 테스트 알고리즘에 사용된다. 이중 포트 메모리 테스트 알고리즘은 한 포트로 쓰기 동작을 수행할 때, 다른 포트로는 같은 행의 옆 셀 또는 같은 열의 아래 셀에 읽기를 하는 동작이 있다. Inst[1:0]='01'(또는 '10')이면, B 포트의 주소를 A 포트 생성되는 주소 값에 row(또는 column)로 '1'만큼 증가시키라는 동작을 지정하는 옵션이다.

알고리즘 명령어 구조는 March 알고리즘을 구현하는

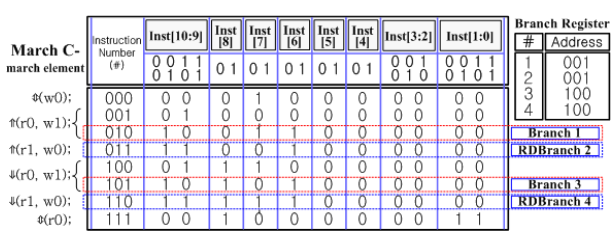
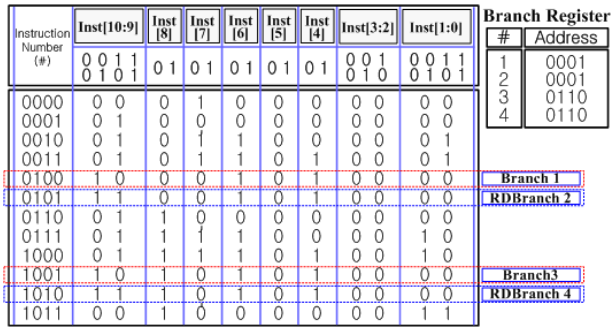


그림 2. March C- 알고리즘의 명령어 구현
Fig. 2. Instruction implementation of March C- algorithm.



Instruction Number (#)	Inst[10:9]	Inst [8]	Inst [7]	Inst [6]	Inst [5]	Inst [4]	Inst[3:2]	Inst[1:0]	
	0 0 1 1 0 1 0 1	0 1	0 1	0 1	0 1	0 1	0 0 1 0 1 0	0 0 1 1 0 1 0 1	
0000	0 0	0	1	0	0	0	0 0	0 0	
0001	0 1	0	0	0	0	0	0 0	0 0	
0010	0 1	0	1	1	0	0	0 0	0 1	
0011	0 1	0	1	1	0	1	0 0	0 1	
0100	1 0	0	0	1	0	1	0 0	0 0	Branch 1
0101	1 1	0	0	1	0	1	0 0	0 0	RDBranch 2
0110	0 1	1	0	0	0	0	0 0	0 0	
0111	0 1	1	1	1	0	0	0 0	1 0	
1000	0 1	1	1	1	0	1	0 0	1 0	
1001	1 0	1	0	1	0	1	0 0	0 0	Branch3
1010	1 1	1	0	1	0	1	0 0	0 0	RDBranch 4
1011	0 0	1	0	0	0	0	0 0	1 1	

Branch Register

#	Address
1	0001
2	0001
3	0110
4	0110

그림 3. March A2PF 알고리즘의 명령어 구현
Fig. 3. Instruction implementation of March A2PF algorithm.

데 최소의 명령어를 사용한다. March 기반의 알고리즘은 데이터 값을 반전하여 동일한 읽기와 쓰기 동작들이 연속적으로 사용되는 경우가 많다. 이런 경우에 제안하는 알고리즘 명령어는 RDBranch 옵션을 사용하여 1개의 명령어로 구현할 수 있다. 예를 들어, ⇑(r0, w1, r1), ⇑(r1, w0, r0)와 같이 6개의 element로 구성된 알고리즘의 경우를 보면, 기존에 명령어들은 최소 6개의 명령어를 사용하여 구현한다. 그런데 제안하는 명령어를 사용하면, ⇑(r0, w1, r1)은 3개의 명령어로 구현을 하고, ⇑(r1, w0, r0)은 RDBranch 옵션을 사용한 1개의 명령어로 구현한다. 따라서 총 4개의 명령어로 구현된다.

그림 2는 March C- (10N) 알고리즘을 제안한 알고리즘 명령어로 구현한 예제이다. 10개의 march element로 구성되는 March C-는 2개의 RDBranch 옵션을 사용한 명령어를 사용하여 8개의 명령어로 구현된다. 따라서 제안하는 알고리즘 명령어는 최소의 명령어로 테스트 알고리즘을 구현할 수 있는 최적의 구조이다.

그림 3은 이중 포트 메모리 테스트 알고리즘 중에 March A2PF(18N) 알고리즘을 제안한 알고리즘 명령어로 구현한 예제이다. 그림 3을 보면, 18N의 March A2PF 알고리즘을 14개의 명령어로 구현하였다. 최소의 명령어로 알고리즘의 구현이 가능하여 하드웨어 오버헤드를 최적화하였다.

2. PMBIST 구조

제안하는 PMBIST 구조는 Instruction Memory, Instruction Counter, Decoder, A-port Signal Generator, B-port Signal Generator, Response Analyzer, I/O selector 그리고 MBIST Controller로 구성된다. 여기서 A-port Signal Generator와 B-port Signal Generator는 각 Control Signal Generator, Data Generator, Address Generator로 구성된다. 그림 4는 제안하는 PMBIST의 구조이다. PMBIST의 각 블록을 간단히 설명하면, Instruction Memory는 제안한 알고리즘 명령어 구조로 구현된 테스트 알고리즘의 명령어를 저장하는 메모리이다. Instruction Counter는 Instruction Memory를 제어하여 저장된 알고리즘 명령어를 선택하여 Instruction Decoder로 보내며, Instruction Decoder는 Instruction Memory에서 선택된 명령어를 해독하여 각 패턴 생성기(A-port와 B-port Signal Generator)에서 패턴을 생성하도록 신호를 인가한다. A-port Signal Generator의 Control Signal Generator는 이중 포트 메모리의 A-port에 인가되는 제어신호를 생성하고, Data Generator는 이중 포트 메모리의 A-port에 인가되는 데이터 패턴을 생성한다. 또한 Address Generator는 이중 포트 메모리의 A-port에 인가되는 1씩 증가/감소하는 주소를 생성한다. B-port Signal Generator의 Control Signal Generator와 Data Generator는 이중 포트 메모리의 B-port에 인가되는 제어신호와 데이터 패턴을 생성한다. 그리고 Address Generator는 이중 포트 메모리의 B-port에 인가되는 A-port를 위한 생성되는 주소에 column 또는 row로 1씩 증가/감소된 주소를 생성한다. Response Analyzer는 테스트 결과 값을 비교하여 고장의 유무를 판별하며, I/O selector는 각 패턴 생성 블록에서 생성된 패턴을 이중 포트 메모리의 port에 연결해주는 회로이다. 이 회로는 이중 포트 메모리 테스트시 BIST에서 생성되는 패턴의 A port와 B port의 연결을 바꿔주어 테스트를 할 수 있게 한다. 마지막으로 MBIST Controller는 외부

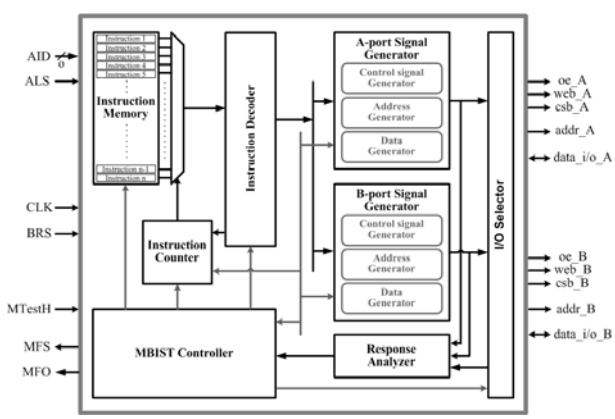


그림 4. PMBIST 구조
Fig. 4. The PMBIST architecture.

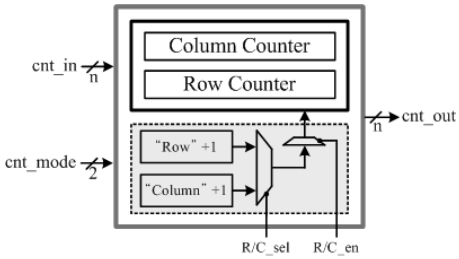


그림 5. 주소 생성기
Fig. 5. The address generator.

로부터 입력 받은 신호를 가지고 PMBIST를 전체적으로 제어한다.

그림 4에서 PMBIST의 입출력 신호들을 살펴보면, CLK, BRS, MTestH, Algorithm Loading Set(ALS), Algorithm Instruction Download (AID), MBIST Finish Signal(MFS), MBIST Fault Out(MFO) 등이 있다. MTestH는 PMBIST가 test mode로 테스트를 시작하라는 입력 신호이고, BRS는 PMBIST의 reset 신호이다. ALS는 외부로부터 instruction을 입력받아 명령어 메모리에 저장하는 mode를 지정하는 입력 신호이고, AID는 알고리즘을 구현한 instruction을 PMBIST로 입력해주는 핀이다. 그리고 MFS은 PMBIST의 동작이 끝났음을 알려주는 출력 신호이고, MFO는 PMBIST의 테스트 결과를 출력하는 핀이다.

제안하는 PMBIST는 이중 포트 메모리 테스트 알고리즘의 각 포트의 주소를 효과적으로 생성하기 위해 2가지의 주소 생성기를 제안하였다. A-port와 B-port를 위한 패턴을 생성하는 A-port Signal Generator와 B-port Signal Generator의 각 주소 생성기는 fast diagonal, fast row 및 fast column 방식을 지원하여 주소를 생성한다. 그런데 B-port Signal Generator의 주소 생성기는 이중 포트 메모리 테스트 알고리즘을 위해 A-port를 위해 생성된 주소 값에 row 또는 column으로 '1'만큼 증가/감소된 주소를 효과적으로 생성하기 위한 부분이 추가된 주소 생성기이다. 이 주소 생성기는 동시에 서로 다른 인접한 셀에 접근할 때 사용된다. 그림 5는 B-port를 위한 주소 생성기를 보여준다. 그림 5에서 회색 부분을 제외한 부분이 A-port를 위한 주소 생성기이다.

그림 5에 주소 생성기의입출력 신호들을 살펴보면, cnt_in와 cnt_out은 카운터의 입력과 생성된 출력 값이다. cnt_ mode는 주소 패턴 생성을 위한 카운터의 모드를 지정하는 입력이다. cnt_ mode가 '00'이면 fast diagonal, '01'이면 fast row 그리고 '10'이면 fast column 방식으로 주소를 생성한다. R/C_sel과 R/C_en은 이중 포트 메모리 테스트 알고리즘의 B-port의 주소를 생성하기 신호이다. R/C_sel은 row와 column 중에 어떤 값을 1을 증/감할지 선택하는 신호이고, R/C_en은 주소 생성이 이 값을 사용 유무를 선택하는 신호이다.

Ⅲ. PMBIST 검증

본 논문에서 제안하는 PMBIST 구조 검증을 위하여 이중 포트 메모리(8.2K×32)에 PMBIST를 연결하여 검증을 하였다. 검증에 사용된 메모리는 삼성전자의 9-bit row address, 4-bit column address 및 32-bit data word를 가지는 dual-port synchronous SRAM을 사용하였다. 그리고 PMBIST는 단일 포트 관련 고장 검출을 위한 March C-(10N), March C+(14N) 알고리즘과 이중 포트 관련 고장을 검출하기 위한 March s2PF(16N), March 2PFav(10N), March A2PF(18N) 알고리즘을 Mentor Graphics의 Modelsim을 이용하여 functional 시뮬레이션을 통하여 동작 검증하였다. PMBIST의 각 알고리즘별 functional 시뮬레이션은 각 알고리즘을 외부로부터 하나씩 입력하여 검증하였다. 표 1은 PMBIST의 패턴 생성을 위하여 제안한 11 bits의 알고리즘 명령어를 사용하여 각 알고리즘을 구현하는데 필요한 명령어 개수와 명령어 bit를 보여준다. 제

표 1. 알고리즘별 명령어 bit 크기
Table 1. Instruction bit sizes for the test algorithms.

알고리즘	명령어 개수	명령어 bit
March C- (10N)	8	88 bits(8×11bits)
March C+ (14N)	10	110 bits(10×11bits)
March s2PF (16N)	8	88 bits(8×11bits)
March A2PF (18N)	12	132 bits(12×11bits)

표 2. 합성 결과
Table 2. Synthesis result.

명령어 메모리의 크기	하드웨어 오버헤드	최대 동작속도
27.50 Byte(11bits x 20)	5,488 gates	300 MHz
41.25 Byte(11bits x 30)	6,138 gates	
55.00 Byte(11bits x 40)	6,812 gates	

표 3. PMBIST 성능 비교
Table 3. Performance comparison of the PMBIST.

	[8]	[9]	[10]	PMBIST
March based 테스트 알고리즘	Y	Y	Y	Y
이중 포트 메모리 테스트 알고리즘	N	N	Y	Y
Flexibility	Medium	Medium	Medium-High	High
최대 동작 속도	-	300 MHz	-	300 MHz
명령어 사이즈 (bits)	8	9	4	11
March C- 알고리즘의 명령어 bit (bits)	152	90	92	88
March A2PF 알고리즘의 명령어 bit (bits)	-	-	248	132
하드웨어 오버헤드 (gates)	12.4K	6.4K	9.9K	6.1K

안하는 알고리즘 명령어는 최소의 명령어로 테스트 알고리즘을 구현할 수 있는 최적의 구조이다. 따라서 10N의 March C-알고리즘은 8개의 명령어를 사용하여 88 bits(8×11bits)의 명령어로 구현된다. 또한 March s2PF(16N)과 March A2PF(18N) 알고리즘은 8개와 12개의 명령어로 구현이 가능하여 최소의 bit을 사용하여 구현된다.

표 2는 제안한 PMBIST를 TSMC 0.13μm 라이브러리를 사용하여 Design Compiler로 합성한 결과이다. PMBIST는 각 메모리를 테스트하기 위한 패턴을 외부로부터 입력받은 명령어 메모리에 저장된 명령어를 사용하여 생성한다. 명령어 메모리의 크기는 PMBIST의 하드웨어 오버헤드에 큰 영향을 준다. 따라서 명령어 메모리의 크기에 의해 최대 몇 개의 명령어를 이용하여 패턴을 생성할 수 있는지 결정된다. March 기반의 알고리즘들은 최대 30개의 명령어를 사용하면 구현이 가능하기 때문에 명령어 메모리의 크기를 41.25 Byte (11bits×30)로 정하였다. 41.25 Byte의 명령어 메모리를 사용한 PMBIST의 합성결과는 2-input nand gate를 기준으로 하드웨어 오버헤드는 6,138 gate이고, 최대 동작 속도는 300 MHz이다. 그 이외에 다양한 명령어 메모리 사이즈별 PMBIST의 하드웨어 오버헤드를 확인하였다.

Ⅳ. 성능평가

표 3은 기존의 PMBIST와 제안하는 PMBIST 구조를 비교한 것이다. 기존의 PMBIST 구조와 제안하는 PMBIST 구조의 지원하는 테스트 알고리즘, 명령어 사이즈 그리고 March C-, March A2PF 알고리즘을 구현하는데 필요한 명령어 bit 등으로 비교하였다. 그리고 기존의 PMBIST와 제안하는 PMBIST의 하드웨어 오버헤드를 비교하였다. 하드웨어 오버헤드는 명령어 메모리의 크기가 41.25 Byte인 PMBIST를 합성한 것이다.

기존에 제안된 [8], [9]는 March 기반의 알고리즘과 non-March 알고리즘을 지원하며, March C- 알고리즘을 구현하는데 많은 명령어가 필요하다. 또한 단일 포트 메모리만 테스트가 가능하고, 이중 포트 메모리는 테스트가 불가능하다. 따라서 이중 포트 메모리 테스트 알고리즘은 지원하지 못한다.

[10]은 이중 포트 메모리 테스트를 위한 PMBIST로 March 알고리즘과 이중 포트 메모리 테스트 알고리즘의 일부만을 지원한다. 이 구조는 한 포트 동작을 4 bits의 test program set을 사용하여 알고리즘을 구성하는 march element 별로 프로그램한다. 따라서 두 포트의 동작을 프로그램하려면 8 bits의 test program set이 필요하고, March A2PF 알고리즘을 구현하는데 248 bits의 명령어가 필요하다. 그리고 DRF 테스트를 지원하지 못하여 고장 검출률이 제한적이다. 또한 하드웨어 오버헤드는 BIST processor와 μprogram memory 블록만 9.9K gates를 가진다.

제안하는 PMBIST는 이중 포트 메모리의 테스트를 위해 모든 March 기반 알고리즘과 이중 포트 메모리 테스트 알고리즘 등을 모두 지원하여 높은 고장 검출률을 가진다. 그리고 March C- 알고리즘과 March A2PF 알고리즘을 88 bits와 132 bits으로 구현하여 기존의 구조보다 적은 bit가 필요하다. 또한 하드웨어 오버헤드도 2-input nand gate 기준으로 기존에 제안된 구조들보다 6.1K gates로 가장 적다. 따라서 제안하는 PMBIST는

다양한 테스트 알고리즘을 지원하여 flexibility와 fault coverage가 높고, 최적화된 알고리즘 명령어를 사용하여 최소의 bit으로 테스트 알고리즘을 구현하여 최적의 하드웨어 오버헤드를 가진다.

V. 결 론

최근에 SoC 환경이 급속히 늘어 가면서 칩 내부의 많은 비중을 차지하고 있는 내장된 메모리에 대한 테스트에 대한 많은 연구가 진행되고 있다. 제안하는 PMBIST는 March 기반 알고리즘과 이중 포트 메모리 테스트 알고리즘 등을 지원하여 높은 flexibility와 fault coverage를 가진다. 또한 제안한 알고리즘 명령어 구조를 사용하면 최소의 명령어를 사용하여 알고리즘의 구현이 가능하고, 기존의 알고리즘 이외에 모든 March 기반의 used defined algorithm의 구현이 가능하다. 즉, 기존에 생각하지 못한 고장에 대해서도 매우 효과적으로 대처할 수 있다. 또한 다양한 주소 생성 방법을 지원하는 주소 생성기는 복잡한 주소를 효과적으로 생성한다. 따라서 최적의 알고리즘 명령어와 효과적인 주소 생성기를 통해 최소의 하드웨어 오버헤드를 가지는 매우 효율적인 PMBIST 구조이다.

참 고 문 헌

[1] S. Hamdioui, Z. Al-Ars, and A. J. Van de Goor, “Testing static and dynamic faults in random access memories,” Proceeding of VLSI Test Symposium, pp. 395-400, Apr. 2002.

[2] S. Boutobza, M. Nicolaidis, K.M. Lamara, and A. Costa, “Programmable Memory BIST,” Proceeding of IEEE International Test Conference, Paper 45.2, Nov. 2005.

[3] Yamasaki, I. Suzuki, A. Kobayashi, K. Horie, Y. Kobayashi, H. Aoki, H. Hayashi, K. Tada, K. Tsutsumida, and K. Higeta, “External memory BIST for system-in-package,” Proceeding of International Test Conference, pp. 1145-1154, Nov. 2005.

[4] A. W. Hakmi, H. J. Wunderlich, C. G. Zoellin, A. Glowatz, F. Hapke, J. Schloeffel, and L. Souef, “Programmable deterministic Built-In Self-Test,” Proceeding of IEEE International Test Conference, pp. 1-9, Oct. 2007.

[5] S. Hamdioui, A. J. Van de Goor, and M. Rodgers, “March SS: a test for all static simple RAM faults,” Proceedings of IEEE International Workshop on Memory Technology, Design and Testing, pp. 95-100, Jul. 2002.

[6] S. Hamdioui, and A. J. Van de Goor, “Efficient tests for realistic faults in dual-port SRAMs,” IEEE Transactions on Computers, vol. 51, No. 5, pp.460-473, 2002.

[7] Y. Park, M. Yang, Y. Kim, D. Lee, and S. Kang, “An Efficiency Testing Algorithm for Realistic Faults in Dual-Port Memories,” Journal of the Institute of Electrical Engineers of Korea, vol.43-SD, No. 2, pp.72-85, 2007.

[8] X. Du, N. Mukherjee, C. Hill, W-T. Cheng, and S. Reddy, “A Field Programmable Memory BIST Architecture Supporting Algorithms with Multiple Nested Loops,” Proceeding of IEEE Asian Test Symposium, pp. 287-292, Nov. 2006.

[9] Y. Park, J. Park, T. Han, and S. Kang, “An Effective Programmable Memory BIST for Embedded Memory,” IEICE Transactions on Information and Systems, vol. E92-D, no. 12, pp. 2508-2511,Dec. 2009.

[10] A. Benso, S. Di Carlo, G. Di Natale, P. Prinetto, and M. Lobetti Bodoni, “A programmable BIST architecture for clusters of multiple-port SRAMs,” Proceeding of IEEE International Test Conference, pp. 557-566, Oct. 2000.

저 자 소 개

박 영 규(학생회원)
2004년 호서대학교 전기공학과 학사 졸업.
2007년 연세대학교 전기전자공학과 석사 졸업.
2007년 현재 연세대학교 전기전자공학과 박사 과정.
<주관심분야 : Memory test, BIST, DFT>

한 태 우(학생회원)
2008년 연세대학교 전기공학과 학사 졸업.
2008년 현재 연세대학교 전기전자공학과 석박사 통합 과정.
<주관심분야 : Memory test 및 repair, DFT>

강 성 호(평생회원)
1986년 서울대학교 제어계측공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원.
1994년 Motorola Inc. 선임연구원.
2007년 현재 연세대학교 전기전자공학과 교수
<주관심분야 : SoC 설계, SoC 테스트>