IEEE 1149.1을 이용한 March 알고리듬의 내장형 자체 테스트 구현

(Implementation of March Algorithm for Embedded Memory Test using IEEE 1149.1)

양 선 웅 † 박 재 흥 † 장 훈 ††
(Sun-Woong Yang) (Jae-Heung Park) (Hoon Chang)

요 약 본 논문에서는 내장 메모리 테스트를 위해 메모리 테스트 알고리즘인 10N March 테스트 알고리즘을 회로로 구현하였으며, 구현된 내장 메모리 BIST 회로를 제어하기 위해 IEEE 1149.1 표준안을 회로로 구현하였다. 구현된 내장 메모리 테스트 회로는 워드 단위의 메모리를 위한 배경 데이터를 이용함으로써 워드 단위 메모리의 고착 고장, 천이 고장, 결합 고장을 완전히 검출할 수 있다.

구현된 회로는 Verilog-HDL을 이용하여 구현하였으며, Synopsys에서 합성하였다. 합성된 메모리 테스트 회로와 IEEE 1149.1 회로의 검증은 메모리 컴파일러에 의해 생성된 메모리 셀과 VerilogXL을 이용하여 수행하였다.

Abstract In this paper, we implemented memory BIST circuit based on 10N march algorithm, and the IEEE 1149.1 has been designed as main controller for embedded memory testing. The implemented memory BIST can be used for word-oriented memory since it adopts background data, this is avaliable for word-oriented memory. It is able to detect all stuck-at faults, transition faults, coupling faults, and address decoder faults in the word-oriented memory.

Memory BIST and IEEE 1149.1 are described at RTL level in Verilog-HDL, and synthesized with the Synopsys. The synthesized circuits are fully verified using VerilogXL and memory cell generated by memory compiler.

1. 서 론

최근에 생산되는 마이크로 프로세서의 경우 내장 메모리가 칩의 전체 게이트 수의 대부분(70~90%)을 차지하고, 전체 다이 면적의 상당 부분(30~50%)을 차지하게 됨에 따라, 내장 메모리를 위한 자체 테스트의 중요성이 더욱 부각되고 있다[1, 2]. 내장 메모리의 테스트는 메모리 자체의 기능적 특성으로 인하여 고착 고장(stuck-at fault) 모델만으로는 테스트하기 어려우며, 내장 메모리의 입·출력 신호를 칩의 외부에서 제어(control)하거나 관찰(observe)하기 어렵기 때문에 기존의 방식으로는 테스트하기 어렵다. 이러한 문제들을 해결하기 위해 가장 널리 사용되는 방법은 내장된 자체 테스트 기법(BIST: Built-In Self Test)을 사용하는 것이다. BIST 기법은 칩의 내부에 테스트 회로를 내장하여 자체적으로 테스트를 수행하는 기법이다. 자체 테스트 회로가 내장된 칩은 부수적으로 면적의 증가 등과 같은 오버헤드를 갖게 되지만, 다음과 같은 장점들을 갖는다[3].

- 각 모듈별로 자체적인 테스트가 수행되므로 전체 시스템의 테스트에 있어서 테스트의 복잡도가 크게 줄어든다.
- 각 모듈별로 적절한 BIST 회로가 내장되므로 모듈별로 가장 적합한 방식의 테스트가 가능하다.
- 고가의 외부 테스트 장치를 사용하지 않고도 빠른 시간에 테스트를 수행할 수 있다.

· 이 논문은 1998년 한국학술진흥재단의 학술연구비(과제번호:1998-016-E00054)에 의하여 지원되었음.
† 학생회원 : 숭실대학교 컴퓨터학과
swyang@watt.soongsil.ac.kr
jhpark@watt.soongsil.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
hoon@computing.soongsil.ac.kr
논문접수 : 1999년 5월 10일
심사완료 : 2000년 3월 13일

이런 장점들로 인하여 내장 메모리의 테스트에 있어서는 BIST 회로의 사용이 확산되고 있다. 더욱이 내장 메모리의 크기가 점차 커져감에 따라 자체 테스트 회로의 단점인 면적 오버헤드(area overhead)가 상대적으로 크게 감소하게 되므로 그 장점이 더욱 부각되고 있다.

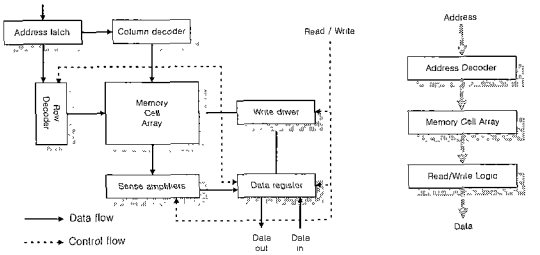
본 논문에서는 대표적인 메모리 테스트 알고리즘인 10N March 테스트 알고리즘에 대하여 고찰하고, Verilog-HDL(Verilog Hardware Description Language)을 이용하여 내장 메모리를 위한 자체 테스트 회로를 구현하였다. 구현된 회로는 CMOS로 구성된 현대 라이브러리를 Synopsys를 이용하여 합성하였으며, 현대 메모리 컴파일러에 의해 생성된 메모리 셀과 Verilog-XL을 사용하여 회로 동작을 검증하였다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 메모리 테스트의 기반이 되는 메모리의 고장 모델을 설명하였고, 3장에서는 10N March 테스트 알고리즘과 워드 단위의 메모리 테스트 알고리즘에 대하여 설명하였다. 4장에서는 Verilog-HDL을 사용한 10N March 테스트 알고리즘의 BIST 구현과 IEEE 1149.1 표준안을 이용한 메모리 테스트에 대해 설명하였고, 5장에서는 실험 결과와 이에 따른 결론을 수록하였다.

2. 메모리의 고장 모델

실제 메모리에서의 고장은 매우 다양한 상태로 나타나게 된다. 따라서 메모리의 정상적인 동작에 영향을 미칠 수 있는 고장의 모든 경우에 대해서 테스트를 수행한다는 것은 실질적으로 불가능하다. 그러나 메모리 테스트의 목적은 특수한 경우를 제외하고는 고장의 유형이나 위치를 파악하기보다는, 단순히 고장의 발생유무를 파악하는 것이다 [1, 4, 5, 6, 7].

그러므로 일반적인 메모리 테스트에서는 먼저 메모리의 구조를 기능 모델(functional model)로 단순화시킨다. 이 경우 메모리는 메모리 셀 배열(memory cell array), 주소 디코더(address decoder), 읽기·쓰기 회로(read · write logic)로 구성된다. 그림 1의 (a)는 메모리의 완전한 기능 모델을 보여 주고 있으며, (b)는 고장 검출에 사용되는 축소된 메모리의 기능 모델을 보여 준다. 축소된 기능 모델에서 발생 가능한 고장들은 발생위치에 따라 나눌 수 있다. 이 경우에 주소 디코더와 읽기·쓰기 회로에서 발생되는 고장들은 고착-개방(stuck-open) 고장과 같은 일부 경우를 제외하고는 대부분 메모리 셀 배열의 고장으로 매핑(mapping)시켜 동시에 테스트할 수 있으며 메모리 셀에서 발생되는 기능 고장들은 다음과 같다.



(a) 완전한 기능 모델 (b) 축소된 기능 모델

그림 1 메모리의 기능 모델

2.1 고착, 천이, 결합 고장 모델

고착 고장은 메모리 셀의 값이 논리값 0이나 1로 고정되어 그 논리값이 변하지 않는 고장으로서, 논리값이 0에 고정되는 고착-0(stuck-at-0) 고장과 1에 고정되는 고착-1(stuck-at-1) 고장이 있다. 고착 고장을 검출하기 위한 테스트는 각각의 모든 셀에 0과 1을 읽고 쓸 수 있어야 한다.

천이 고장은 메모리 셀의 논리값이 0에서 1(상향 천이), 또는 1에서 0(하향 천이)으로의 천이가 되지 않는 고장이다. 천이 고장을 검출하기 위한 테스트는 각각의 모든 셀에 상향 천이와 하향 천이를 일으킬 수 있어야 하고, 다른 천이가 더 이상 발생되기 전에 셀의 값을 읽을 수 있어야 한다.

결합 고장은 특정 메모리 셀에서 논리값의 천이가 일어날 때, 이 셀과 연관된 다른 메모리 셀의 값이 변하는 고장이다. 결합 고장에는 한 셀의 천이가 다른 셀의 내용을 바꾸는 반전 결합 고장(inversion coupling fault)과 한 셀의 천이가 다른 셀의 내용을 0이나 1의 논리값으로 고정시키는 동행 결합 고장(idempotent coupling fault)이 있다. 결합 고장을 검출하기 위해서는 발생 가능한 모든 경우의 결합 고장을 활성화시키고, 결합된 셀에 어떤 값을 쓰기 전에 이를 읽어볼 수 있으면 된다.

3. 메모리 테스트 알고리즘

3.1 10N March 테스트 알고리즘

본 논문에서 구현한 10N March 테스트 알고리즘은 고착, 천이, 결합 고장과 고착-개방 고장을 검출할 수 있는 알고리즘으로서, 10N(N은 전체 접근 가능한 메모리 주소의 크기)의 시간 복잡도를 가지며, 비트 단위 및 워드 단위 메모리에 모두 사용이 가능하다. 10N March 테스트 알고리즘은 크게 초기화하는 부분과 5개의 March 요소로 이루어져 있다. 하나의 March 요소는 읽기와 쓰기를 각각 한번씩 수행한다. 그림 2는 비트 단

Addr.	Init.	March1	March2	March3	March4	March5
0	W0	R0, W1	R1, W0	R0, W1	R1, W0	R0
1	W0	R0, W1	R1, W0	↑	↑	↑
2	W0	R0, W1	R1, W0			
I	↓	↓	↓	R0, W1	R1, W0	R0
I				R0, W1	R1, W0	R0
N-1	W0	R0, W1	R1, W0	R0, W1	R1, W0	R0

10N Test Algorithm

그림 2 10N March 테스트 알고리즘

위의 메모리에 대한 10N March 테스트 알고리즘의 진행 과정을 보여 준다.

10N March 테스트 알고리즘에서 W0, W1, R0, R1의 정의는 다음과 같다.

- W0: 메모리 셀에 0을 쓴다.
- W1: 메모리 셀에 1을 쓴다.
- R0: 메모리 셀에서 0을 읽는다.
- R1: 메모리 셀에서 1을 읽는다.

3.2 10N March 테스트 알고리즘에서의 고장 검출

(가) 고착-개방 고장 검출

Stuck-open 고장은 메모리 내의 셀을 액세스할 수 없는 고장이며, 이를 검출하기 위해서는 'Rx, W$\bar{x}$, R$\bar{x}$' 순서로 테스트하여야만 한다. 10N March 테스트 알고리즘에서는 March1 요소와 March2 요소에서 'R0, W1, R1'의 순서로 테스트를 수행하고, March4 요소와 March5 요소에서는 'R1, W0, R0'의 순서로 메모리를 테스트한다. 따라서, 10N March 테스트 알고리즘을 이용해 고착-개방 고장을 검출할 수 있다.

(나) 고착 고장 검출

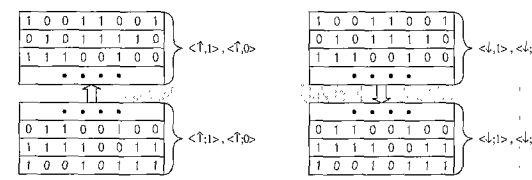
고착 고장을 검출하기 위해서는 메모리의 각 셀에 대해 0을 쓰고 읽을 수 있어야 하며, 또한 1을 쓰고 읽을 수 있어야 한다. 10N March 테스트 알고리즘은 초기화 과정과 March2 요소에서 모든 셀에 0을 쓰고 March1 요소와 March3 요소에서 이를 읽는 작업을 수행하므로 고착-1 고장을 검출할 수 있다. 또한, March1과 March 3 요소에서 모든 셀에 1을 쓰고 March2와 March4 요소에서 이를 읽음으로써 고착-0 고장을 검출할 수 있다.

(다) 천이 고장 검출

천이 고장은 메모리의 각 셀이 상향 천이와 하향 천이를 하도록 만들고, 천이가 일어난 직후 셀의 값을 읽어봄으로써 검출이 가능하다. 10N March 테스트 알고리즘에서는 두 번의 상향 천이와 하향 천이가 발생한다. 초기화 과정과 March1 요소, March2 요소와 March3 요소에 의해 상향 천이가 발생하며, March1 요소와 March2 요소, March3 요소와 March4 요소에 의해 하향 천이가 발생한다. 초기화 과정과 March1 요소에 의해 발생하는 상향 천이에 대한 고장은 March2 요소의 읽기에 의해 검출이 가능하며, March2 요소와 March3 요소에 의해 발생하는 상향 천이에 대한 고장은 March4 요소의 읽기에 의해 검출이 가능하다. 마찬가지로, March1 요소와 March2 요소에 의해 발생하는 하향 천이는 March3 요소의 읽기에 의해 검출이 가능하다.

(라) 결합 고장 검출

단일 고장 모델을 사용할 때 메모리 셀 어레이 안에서 발생할 수 있는 결합 고장은 결합하는 셀에서 일어나는 천이의 종류, 결합하는 셀(coupling cell)과 결합된 셀(coupled cell)의 위치, 결합된 셀이 갖고 있는 논리값 등에 따라 그림 3에서처럼 8가지로 구분할 수 있다.



(a) 상향 천이가 일어났을 때 (b) 하향 천이가 일어났을 때

그림 3 발생 가능한 결합 고장

그림 3의 (a)는 어느 메모리 셀에서 상향 천이가 일어났을 때 발생할 수 있는 다른 워드 내의 셀과 결합 고장을 보여주고, (b)는 하향 천이가 일어났을 때 다른 워드 내의 셀과의 결합 고장을 보여준다. 그림 3에서 사용된 기호는 다음과 같은 의미는 갖는다.

<↑;0> : 결합하는 셀에 상향 천이가 발생하였을 때 결합되는 셀의 값이 1에서 0으로 바뀐다.

<↑;1> : 결합하는 셀에 상향 천이가 발생하였을 때 결합되는 셀의 값이 0에서 1로 바뀐다.

<↓;0> : 결합하는 셀에 하향 천이가 발생하였을 때 결합되는 셀의 값이 1에서 0으로 바뀐다.

<↓;1> : 결합하는 셀에 하향 천이가 발생하였을 때 결합되는 셀의 값이 0에서 1로 바뀐다.

그림 4는 10N March 테스트 알고리즘의 각 March 요소에서 테스트시 변화되는 주소의 방향과 발생할 수 있는 결합 고장을 보여준다.

- 초기화: ↓(W0)

초기화를 수행한다. 그림 4의 (a)는 초기화 단계에서 메모리 셀 어레이 값의 변화를 보여준다. 이 단계에서는 실제 고장이 발생될 수 있으나 이것을 파악할 수 없다. 결합 고장은 어느 셀에 천이가 발생했을 때 다른 어떤

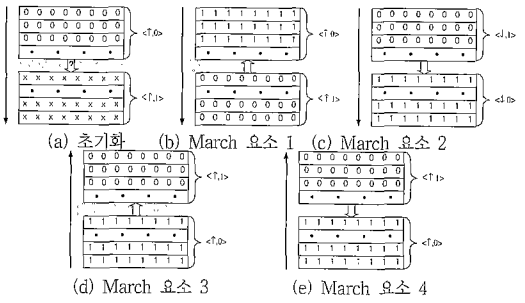


그림 4 10N March 테스트 알고리즘에서 발생 가능한 결합 고장

셀이 지니고 있는 값이 천이의 영향으로 변하는 것인데, 초기화 단계에서는 메모리 셀들이 어떤 값들을 지니고 있는지 알 수 없기 때문에 천이가 발생했는지 안 했는지를 알 수 없다. 따라서, 초기화 단계에서는 고장이 발생하지 않는다고 가정한다.

- March1 요소 : ↓(R0,W1)

그림 4의 (b)는 March1 요소에서 메모리 셀 어레이 값의 변화를 보여준다. i 번지에 W1을 했을 때, 0~(i-1) 번지에서는 <↑;0> 고장이 발생할 수 있다. 왜냐하면 i 번지까지 테스트가 진행되었고, 전혀 고장이 발생하지 않았다면 0~(i-1) 번지에는 모두 1값이 쓰여져 있기 때문이다. 이 고장은 March 요소 2에서 검출이 된다. (i+1)~(n-1) 번지는 초기화 단계에서 0으로 초기화되었기 때문에 <↑;1> 고장이 발생할 수 있다. 이 고장은 March1 요소가 진행되면서 검출이 된다.

- March2 요소 : ↓(R1,W0)

그림 4의 (c)는 March2 요소에서 메모리 셀 어레이 값의 변화를 보여준다. i 번지에 W0을 했을 때, i 번지까지 전혀 고장이 발생하지 않고 테스트가 진행되었다면 0~(i-1) 번지에는 모두 0값이 쓰여져 있기 때문에 0~(i-1) 번지에서는 <↓;1> 고장이 발생할 수 있다. 이 고장은 March3 요소에서 검출이 된다. (i+1)~(n-1) 번지는 이전 단계에서 1이 쓰여졌기 때문에 <↓;0> 고장이 발생할 수 있다. 이 고장은 March2 요소가 진행되면서 검출이 된다.

- March3 요소 : ↑(R0,W1)

그림 4의 (d)는 March3 요소에서 메모리 셀 어레이 값의 변화를 보여준다. i 번지에 W0을 했을 때, March 요소 2에서 0값을 써 놓았기 때문에 0~(i-1) 번지에서는 <↑;1> 고장이 발생할 수 있다. 이 고장은 March3 요소가 진행되면서 검출이 된다. (i+1)~(n-1) 번지는 이 단계에서 1을 썼기 때문에 <↑;0> 고장이 발생할 수 있다. 이 고장은 March4 요소에서 검출이 된다.

- March4 요소 : ↑(R1,W0)

그림 4의 (e)는 March4 요소에서 메모리 셀 어레이 값의 변화를 보여준다. i 번지에 W0을 했을 때, 이전 단계의 테스트에서 1값을 써 놓았기 때문에 0~(i-1) 번지에서는 <↓;0> 고장이 발생할 수 있다. 이 고장은 March4 요소가 진행되면서 검출이 된다. (i+1)~(n-1) 번지는 이 단계에서 0을 썼기 때문에 <↓;1> 고장이 발생할 수 있다. 이 고장은 March5 요소에서는 검출된다.

3.3 워드 단위 메모리의 테스트

워드단위의 메모리는 비트 단위의 메모리와는 달리 워드 단위로 읽기와 쓰기가 일어나게 되고, 워드는 두 개 이상의 비트로 구성되므로 워드 단위의 메모리 테스트는 하나의 워드 내에서 발생될 수 있는 fault masking의 문제를 고려해야만 한다. 예를 들어 그림 5과 같은 4비트(b_w, b_x, b_y, b_z)의 워드에 b_w의 값이 0에서 1로 바뀔 때 b_z의 값이 1이 되는 동행 결합 고장이 있다고 하자. 만약 이 워드의 값이 "0000"이었고, 새롭게 반전된 논리값 "1111"을 쓸 때 b_z의 값은 '1'로 고정되지만 이것은 워드에 쓰여진 값과 같으므로 고장이 감지되지 않는다. 이 고장을 감지하기 위해서는 "1xy0"의 값(x, y는 임의의 값)이 워드에 쓰여지고 읽혀져야 한다.

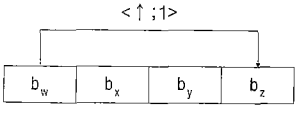


그림 5 워드 내에서의 동행 결합 고장

이러한 워드단위의 메모리 테스트에서 하나의 비트 패턴만으로는 워드 내에서 비트간에 발생할 수 있는 고장 마스크의 문제를 해결할 수 없으므로 이 고장들을 검출하기 위해서는 배경 데이타(data background)라고 불리는 비트 패턴들이 필요하다. 배경 데이타로 사용될 수 있는 비트 패턴들은 앞에서 언급한 고장 마스크 문제를 해결할 수 있어야만 하며, 필요한 비트 패턴들의 수는 한 워드의 비트 수에 의해 결정된다. 한 워드의 비

표 1 배경 데이타

	W(0)	W(1)
P0	00000000	11111111
P1	01010101	10101010
P2	00110011	11001100
P3	00001111	11110000

트 수가 m이라면 $\lceil \log_2 m \rceil + 1$개 이상의 배경 데이타가 사용되어야 한다 [1, 8, 9]. 워드의 크기를 8비트라고 가정할 때 배경 데이타의 한 예는 표 1과 같다.

워드 단위로 확장된 10N March 테스트 알고리즘을 수행하기 위해서는 위에서 정의된 W0, R0, W1, R1 대신에 W(0), W(1), R(0), R(1)이 사용된다

- W(0):배경 데이타를 메모리에 쓴다.
- W(1):역전된(inverted) 배경 데이타를 메모리에 쓴다.
- R(0):배경 데이타를 메모리에서 읽는다.
- R(1):역전된 배경 데이타를 메모리에서 읽는다.

4. 10N March 테스트 알고리즘을 이용한 자체 테스트 회로의 구현

본 연구에서는 Verilog-HDL을 사용하여 10N March 테스트 알고리즘을 이용하여 그림 6과 같은 구조를 갖는 자체 테스트 회로를 구현하였다.

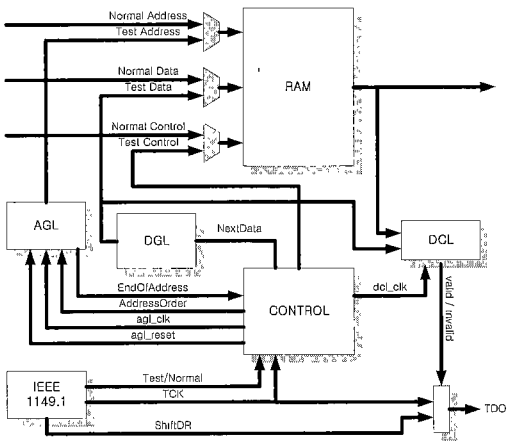


그림 6 10N March 테스트 알고리즘의 구현

가) 제어 회로(CONTROL)

제어 회로는 테스트 진행 과정 중에 BIST 회로의 각 모듈의 동작을 제어하는 회로이다. 이 회로는 테스트의 시작과 종료를 판단하고, 테스트가 진행되는 동안 BIST 회로의 각 모듈에 적절한 제어 신호를 보낸다. CONTROL은 두 개의 FSM과 각 FSM의 상태에 따라 제어 신호를 생성하는 회로로 이루어져 있다. 그림 7은 제어 회로의 구조를 보여 준다.

그림 7에서 위쪽의 FSM은 March 단계를 제어하기 위한 FSM이며, 아래쪽 FSM은 사용되는 배경 데이타를 제어하기 위한 FSM이다. ClockControl은 메모리

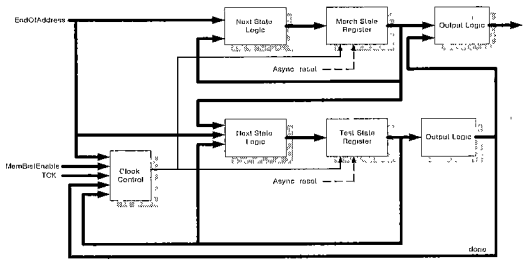
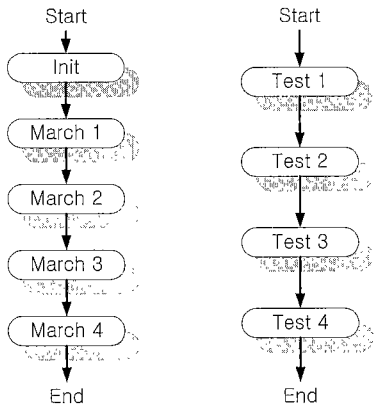


그림 7 제어 회로의 구조



(a) March 단계 제어 FSM (b) Test 단계 제어 FSM

그림 8 제어 회로에서 사용된 상태 전이도

BIST가 Enable되지 않을 경우나 모든 테스트가 끝났을 경우 TCK를 CONTROL 내부에 인가시키지 않음으로써 BIST 회로가 동작하지 않도록 한다. 그림 8는 제어 회로에서 사용한 FSM의 상태 전이도를 보여준다.

나) 주소 생성 회로(AGL:Address Generation Logic)

주소 생성 회로는 테스트 모드에서 테스트될 워드를 지정하는데 사용되는 주소를 생성하는 회로 제어 회로에서 인가되는 클럭에 동기화되어 동작한다. 그림 9는 주소 생성 회로의 구조를 보여준다.

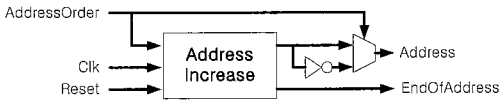


그림 9 주소 생성 회로의 구조

그림 9에서 AddressOrder는 제어 회로에서 인가되는 신호로서 주소를 증가하는 방향 또는 감소하는 방향으로 만들어 낼 것인지를 지정하는 신호이다. 주소 증가기

(Address Increase)에서 생성되는 신호는 Address Order 신호에 따라서 주소 증가기에서 나오는 신호를 그대로 주소로 사용하거나, 혹은 반전 시켜서 주소로 사용한다.

생성된 주소는 Address 포트로 출력되며, 증가하는 방향으로의 주소 생성에서 생성된 주소가 111..11 일 경우나 감소하는 방향으로의 주소 생성에서 생성된 주소가 000..00 일 경우에 EndOfAddress 신호가 1이 된다. 이 신호는 하나의 March 요소가 끝났음을 의미하며, 제어 회로로 전달되어 다음 March 요소가 수행될 수 있도록 하였다.

(다) 데이타 생성 회로(DGL; Data Generation Logic)

데이타 생성 회로는 테스트 알고리즘의 진행 순서에 따라 테스트될 메모리에 배경 데이타를 공급해 주는 회로이다. 그림 10은 데이타 생성 회로의 구조를 보여준다.

그림 10 데이타 생성 회로의 구조

그림 10에서 NextData 신호는 제어 회로에서 생성되는 신호로서 March 요소 5에서만 1을 갖고 그 외의 March 요소에서는 0을 갖는다. 하나의 배경 데이타를 이용하여 10N March 테스트를 끝냈을 때 이 신호에 falling edge가 발생하게 되고, 이에 맞추어 배경 데이타를 바꾸어주게 된다.

(라) 데이타 비교 회로(DCL: Data Comparison Logic)

데이타 비교 회로는 메모리에서 읽어온 데이타와 DGL에서 생성되는 참고(reference) 데이타를 비교하여 메모리 고장 여부를 기록하는 레지스터에 비교 결과를 기록하는 역할을 한다. 그림 11는 데이타 비교 회로의 구조를 보여준다.

그림 11에서 DclEn은 데이타 비교 회로에서 사용하는 클럭으로 제어 회로에서 생성되는 신호이다. 이 신호

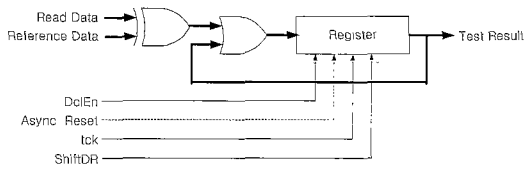


그림 11 데이타 비교 회로의 구조

는 초기화 단계를 제외한 나머지 March1부터 March5 단계까지 발생한다. TCK와 ShiftDR 신호는 메모리 자체 테스트가 끝난 후 경계 주사 회로를 이용하여 압축치 레지스터의 값을 TDO 포트를 통해 외부에 출력시키기 위한 신호이다. Read Data는 메모리에서 읽어오는 값이며, Reference Data는 DGL에서 생성되는 값으로 정상 동작시 메모리에서 읽어오는 결과값을 의미한다.

(마) IEEE 1149.1 회로

기판 수준의 테스팅을 지원하기 위하여 IEEE 1149.1이 표준안으로 제정되어 있다[10]. 따라서, 모든 칩에 삽입되어 있는 IEEE 1149.1을 이용하여 March 알고리듬 내장형 자체 테스트 회로를 동작시킬 수 있도록 기능을 확장하였다.

TAP 제어기의 Shift-IR 상태에서 TDI를 통하여 IEEE 1149.1 회로의 명령어 레지스터에 메모리 테스트를 위한 명령어가 적재된다. Updata-IR 상태에서 적재된 명령어가 디코딩되고, TAP 제어기가 Test-Logic Idle 상태가 될 때 테스트 실행 신호인 Test/Normal 신호가 엑티브된다. 그러면, CONTROL 회로가 동작하여 메모리 테스트를 수행하게 되고, 메모리 테스트가 끝날때까지 TAP 제어기는 Test-Logic Idel 상태를 유지한다.

테스트가 끝난 후 TAP 제어기는 Shift-DR 상태로 이동하여 DCL에 저장되어 있는 결과값을 TDO를 통하여 외부로 내보내게 된다.

5. 실험결과

본 논문에서는 구현된 10N March 알고리즘과 경계 주사 회로의 동작을 검증하기 위하여 현대 메모리 컴파일러를 이용하여 생성된 메모리 셀을 이용하였다. 동작 검증은 다음과 같이 수행되었다. 경계 주사 회로에 내장 메모리 테스트를 위한 명령어를 적재함으로써, 내장 메모리 테스트를 위한 제어 신호를 생성하였고, 이 신호에 따라 내장된 자체 테스트 회로가 테스트 패턴을 생성하였다. 그리고 내장된 자체 테스트 회로에서 출력된 테스트 패턴을 이용하여 메모리 셀에 읽기와 쓰기를 수행하여 구현된 회로의 동작을 검증하였다.

본 논문에서 사용된 현대 메모리 셀의 동작 파형을 살펴보면 그림 12과 같다. 그림 12은 같은 메모리 셀에 대하여 읽기와 쓰기를 할 때, 각 제어 신호의 타이밍을 보여주고 있다. 데이타를 읽을 경우는 먼저 읽을 데이타의 유효한 주소가 address 포트에 인가되어야 하고, WEB, OE 신호가 논리값 1이 되어야 한다. 이 값들이 안정된 후에 RAMEN 신호가 논리값 1이 되면, 메모리

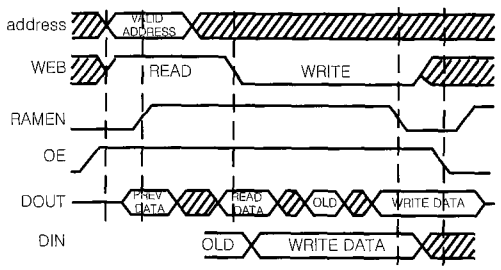


그림 12 메모리의 읽기/쓰기 동작

에서 새로운 데이타를 읽어서 DOUT 포트로 출력하게 된다. 데이타를 쓸 경우는 WEB 신호가 논리값 0으로 되고, OE, RAMEN 신호는 값을 유지하고 있어야 하며, 쓸 데이타가 DIN 포트에 인가되어야 한다.

그림 13는 메모리 BIST 회로와 IEEE 1149.1 회로를 synopsys를 이용하여 합성한 결과이다.

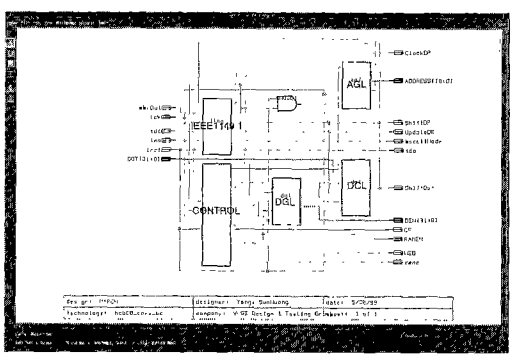


그림 13 메모리 BIST 회로와 IEEE 1149.1 회로의 합성도

그림 14와 그림 15은 IEEE 1149.1 회로의 TAP 제어기 상태의 변화와 TAP 제어기의 출력 파형을 보여주고 있다. 그림 14는 테스트 데이타 레지스터에 관련된 TAP 제어기의 상태 변화와 출력 신호를 보여주고 있다. 그림 14의 UpdateDR 신호는 TAP 제어기의 상태가 Update-DR 상태에 들어갔을 때 falling edge에서 ~TCK의 파형을 생성하고 있다. enable 신호는 TAP 제어기의 상태가 Shift-DR 상태에 들어갔을 때 falling edge에서 논리 1이 된다. 즉 TDO 포트로 출력되는 값은 falling edge에서 유효하게 됨을 알 수 있다. sel 신호는 Select-DR Scan 상태가 되었을 때 rising edge에서 논리 0이 됨으로써 테스트 레지스터의 출력이 TDO 포트로 연결되게 한다. ClockDR 신호는 Capture-DR

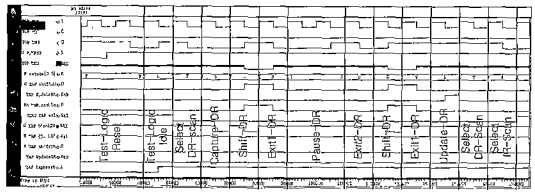


그림 14 테스트 데이타 레지스터를 위한 TAP 제어기의 동작

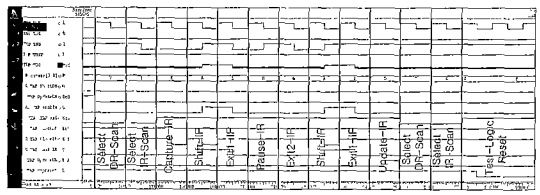


그림 15 명령어 레지스터를 위한 TAP 제어기의 동작

상태와 Shift-DR 상태에서 TCK와 같은 위상을 갖는 클럭을 생성한다. 그림 15은 명령어 레지스터에 관련된 TAP 제어기의 상태 변화와 출력 신호를 보여주고 있으며, 명령어 레지스터에 관련된 출력 신호를 생성하는 점을 제외하고 동작은 테스트 데이타 레지스터에 관련된 TAP제어기의 동작과 동일하다.

그림 16은 메모리 테스트가 시작되는 부분의 시뮬레이션 결과를 보여 주고 있다. TAP 제어기의 Test-Logic Idle(그림 15의 다섯 번째 신호 : C) 상태에서 메모리 테스트가 수행됨을 볼 수 있다.

그림 17은 메모리 테스트의 끝 부분과 결과값이 TDO로 출력되는 부분의 시뮬레이션 결과를 보여주고

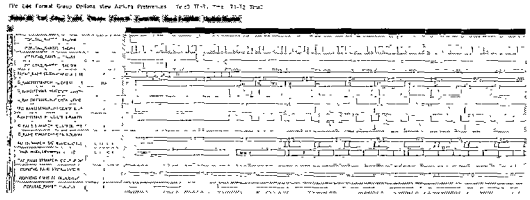


그림 16 10N March 테스트의 수행(1)

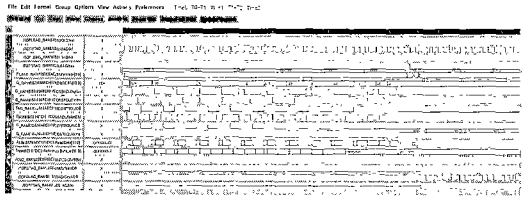


그림 17 10N March 테스트의 수행(2)

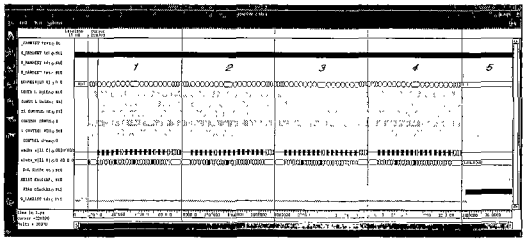


그림 18 10N March 테스트의 수행

있다. 메모리 테스트가 끝난 후 TAP 제어기가 Shift-DR 상태(2)로 이동하고 테스트 결과가 IEEE 1149.1의 TDO 포트(그림 16의 가장 아래에 있는 신호)를 통해 칩 외부로 출력되는 것을 보여준다.

그림 18는 메모리 BIST 회로의 주소를 축소하여 시뮬레이션한 결과를 보여준다. 그림에서 1, 2, 3, 4는 각각의 배경 데이타를 가지고 10N의 테스트가 수행되는 과정이고, 5는 테스트 결과가 TDO 포트를 통해 칩 외부로 출력되는 것을 보여준다.

구현한 메모리 BIST 회로의 최대 동작 주파수는 게이트 레벨의 유닛 딜레이를 고려하여 Verilog-XL로 시뮬레이션한 결과 100MHz 이며, 전력 소모는 synopsys의 report 명령의 power를 이용하여 산출한 결과 97.9898mW이다.

그리고, 구현된 메모리 BIST 회로의 크기는 메모리의 주소가 1K이고 데이타 폭이 32 비트인 메모리에 대해 2-입력 NAND 게이트를 1이라고 했을 때 990이다. 그림 19는 메모리의 크기가 1K일 때의 메모리 BIST 회로의 크기를 1이라고 했을 때, 메모리의 크기를 16K까지 확장하면서 메모리 BIST 회로의 증가율을 비교한 그래프이다. 메모리 크기의 증가율에 비해 메모리 BIST 회로의 증가율이 매우 적음을 알 수 있다. 그래프에서 1부터 16까지 증가하는 그래프는 메모리 크기의 증가율이며, 다른 그래프는 메모리 BIST 회로의 크기 증가율이다

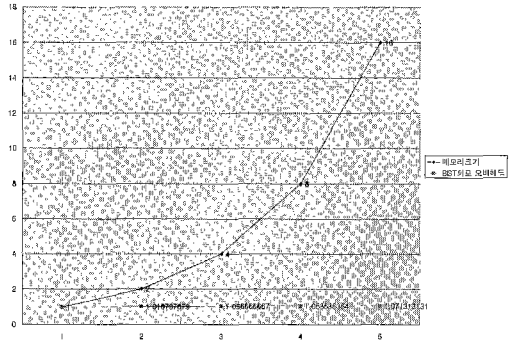


그림 19 메모리 크기의 증가에 따른 메모리 BIST 회로의 증가율 비교

6. 결론 및 향후과제

본 논문에서는 내장 메모리를 위하여 10N March 알고리즘을 이용한 메모리 BIST 회로 및 요즈음 상용칩에 반드시 적용되고 있는 IEEE 1149.1 회로를 구현하였다. 구현된 회로는 Verilog HDL을 이용하여 디자인하였으며, 현대 라이브러리를 이용해 Synopsys에서 합성하였다. 합성된 회로의 동작 검증은 현대 메모리 컴파일러에서 생성된 메모리 셀과 Cadence 사의 Verilog-XL 시뮬레이터를 사용하여 수행하였다.

구현된 IEEE 1149.1 회로는 내장 메모리를 위한 자체 테스트 회로를 제어 할 수 있도록 설계하여 내장 메모리 테스트를 위한 전용 핀을 따로 추가하지 않도록 하였습니다.

메모리 BIST 회로의 최대 동작 주파수는 Verilog-XL로 시뮬레이션한 결과 100MHz 이며, 전력 소모는 synopsys를 이용하여 산출한 결과 97.9898mW이다.

향후 연구과제에서는 칩에 내장되는 멀티 포트 메모리 테스트, 캐쉬 메모리 테스트 및 다수의 메모리를 갖는 시스템에의 적용을 수행할 것이다. 그리고 사용자로부터 메모리 셀에 대한 정보를 받아서 내장 메모리 테스트를 위한 자체 테스트 회로를 자동 생성할 수 있는 시스템을 개발할 것이다.

참 고 문 헌

[1] A. J. Goor, Testing Semiconductor Memories, John Wiley & Sons Ltd., 1991.

[2] MICROPROCESSOR TEST SECTION, Design & Test of Computers, Vol. 15, No. 3, 1998pp. 56-96.

[3] Memory BistCore™ User's Refernece Manual, GeneSys TestWare, Revision 1.4, June, 1998.

[4] Tom Chen and Glen Sunada, "A Self-Testing and Self-Repairing Structure for Ultra-Large Capacity Memories," International Test Conference, 1992.

[5] J. V. Sas, G. V. Wause, E. Huyskens and D. Rabaey, "BIST for Embeded Static RAMs with Coverage Calculation," International Test Conference, 1993.

[6] V. G. Mikitjuk, V. N. Yarmolik, A. J. van de Goor, "RAM Testing Algorithms for Detection Multiple Linked Faults," International Test Conference, 1996.

[7] R. Nair, S. M. Thatte and J. A. Abraham, "Efficient Algorithms for Testing Semiconductor Random-Access Memories," IEEE Transactions on Computers, Vol. C-28, No. 3, March 1979, pp. 258-261.
[8] Pinamki Mazumder and Kanad Chakraborty, "Testing and Testable Design of High-Density Random-Access Memories," Kluwer Academic Publishers, 1996.
[9] R. Dekker, F. Beenker, L. Thijseen, "Fault Modeling and Test Algorithm Development for Static Random Access Memories," International Test Conference, 1988.
[10] IEEE std 1149.1-1990(including IEEE std, 1149.1a-1993), "IEEE Standard Test Access Port and Boundary-Scan Architecture."

양 선 웅
1996년 숭실대학교 전자계산학과 졸업(B.S). 1998년 숭실대학교 대학원 전자계산학과 졸업(M.S). 1998년 ~ 현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 컴퓨터 구조, VLSI 설계 및 테스팅, CAD

박 재 흥
1999년 숭실대학교 컴퓨터학부 졸업(B.S). 1999년 ~ 숭실대학교 대학원 컴퓨터학과 석사과정. 관심분야는 컴퓨터 구조, CAD. VLSI 설계. VLSI 테스팅.

장 훈
1987년 서울대학교 전자공학과 졸업(B.S.). 1989년 서울대학교 전자공학과 졸업(M.S.). 1993년 University of Texas at Austin 박사학위 취득. 1991년 IBM Inc. 1993년 Motorola Inc. Senior Membger of Technocal Staff. 1994년 ~ 현재 숭실대학교 컴퓨터학부 조교수. 관심분야는 컴퓨터 시스템, VLSI 설계, VLSI 테스팅