論文2001-38SD-10-10

내장 메모리 테스트를 위한 BIST 회로 자동생성기

(Automatic BIST Circuit Generator for Embedded Memories)

梁 善 雄 *, 張 勳 **

(Sunwoong Yang and Hoon Chang)

요 약

본 논문에서 구현한 GenBIST는 메모리 테스팅을 위한 정보를 입력으로 받아 테스트용 회로를 VerilogHDL 코드로 자동 생성해 주는 설계 자동화 툴이다. 상용 툴들을 포함한 기존의 툴들은 대부분 메모리 테스트를 위한 알고리즘들을 라이브러리화하고 이를 회로로 생성해주는 방식인데 반해, 본 논문에서 구현한 툴은 사용자가 정의한 알고리즘대로 회로를 생성해 줌으로써 새로운 알고리즘의 적용을 용이하게 하였다. 또한 다중 메모리를 지원할 수 있게 함으로써 메모리 BIST 회로를 공유할 수 있게 하였고 serial interfaceing 기법을 사용함으로써 경계 주사 기법과 함께 사용될 경우 메모리 테스트를 위한 부가적인 핀을 필요로 않는다.

Abstract

GenBIST implemented in this paper is an automatic CAD tool, which can automatically generate circuitry in VerilogHDL code based on user defined information for the memory testing. While most commercial and conventional CAD tools adopt a method in which they make memory-testing algorithms as a library to generate circuitry, our tool can generate circuitry according to the user-defined algorithm, which makes application of various algorithms easier. In addition, memory BIST circuitry can be shared with other memories by supporting embedded memories in our tool. Also, extra pins for the memory testing are not required when boundary scan technique is combined.

Ⅰ. 서 론

최근 들어 하나의 칩에 대한 회로의 집적도는 시스템의 고성능화, 고기능화 및 소형화 요구와 함께 설계, 공정 기술의 발달에 힘입어 급속하게 증가되고 있다. 이에 따라, 제한된 면적 안에 더 많은 트랜지스터를 집적시킬 수 있게 되었으며, 칩의 기능을 더욱 향상시키기 위해 예전에는 칩 외부에 배치하였던 메모리 같은 모듈들도 이제는 칩 안에 내장되는 추세이다.

이와 같이 복잡해진 칩의 테스트는 갈수록 어려운 문제가 되어가고 있으며, 특히 내장된 메모리의 테스트는 가장 어려운 부분으로 여겨지고 있다. 메모리가 내장되기 전의 칩 테스트는 테스트 용이화 설계 기법(design for testability)과 경계 주사(boundary scan) 기법을 사용하여 어느 정도의 오버헤드를 감수하는 수준에서 만족할 만한 테스트 결과를 얻을 수 있었다[1,2].

* 正會員, 崇實大學校 컴퓨터學科
(School of Computing, Graduate School, Soongsil University)
** 正會員, 崇實大學校 컴퓨터學部
(School of Computing, Soongsil University)
接受日字:2000年12月4日, 수정완료일:2001年9月27日

그러나 내장된 메모리의 테스트는 메모리 자체의 기능적 특성으로 인하여 조합회로(combinational logic)에서 사용되는 고착 고장(stuck fault) 모델만으로는 테스트하기가 어렵고, 내장된 메모리의 입출력 신호를 칩의 외부에서 제어하거나 관찰하기 어렵기 때문에 기존의 방식으로는 효율적인 테스트를 수행하기가 어렵다[3,4,5].

이러한 문제들을 해결하기 위하여 가장 널리 사용되는 방법이 내장된 자체 테스트 기법(BIST : Built-In Self Test)이다. 내장된 자체 테스트 기법은 칩의 내부에 테스트 회로를 내장하여 자체적으로 테스트를 수행하는 기법으로 부수적인 면적의 증가 등과 같은 오버헤드를 갖게 되지만, 각 모듈별로 자체적인 테스트가 수행되므로 전체 시스템의 테스트에 있어서 테스트의 복잡도가 크게 줄어들고, 고가의 외부 테스트 장비를 사용하지 않고도 빠른 시간에 테스트를 수행할 수 있다는 장점 때문에 내장된 메모리의 테스트에 있어서는 BIST 회로의 사용이 확산되고 있다. 더욱이 내장된 메모리의 크기가 점차 커져감에 따라 내장된 자체 테스트 회로의 단점인 면적 오버헤드가 상대적으로 크게 감소하게 되므로 그 장점이 더욱 부각되고 있으며, 메모리 테스트를 위한 설계 자동화 툴들이 많이 개발되었다.

그러나 이러한 툴들의 대부분은 메모리 테스트 알고리즘들을 라이브러리화하여 사용하기 때문에, 그 툴에서 지원하지 않는 알고리즘은 적용할 수 없다는 문제점이 있다[6,7]. 본 논문에서 제안한 설계 자동화 툴인 GenBIST는 제어가 비교적 단순한 SRAM 테스팅을 위하여 사용자가 정의한 알고리즘대로 메모리 테스트 회로를 생성해 줌으로써 기존의 알고리즘 및 새로운 모든 알고리즘에 적용될 수 있게 하였다.

본 논문의 전체적인 구성은 다음과 같다. 2장에서는 메모리 BIST 회로의 설계 자동화 툴인 GenBIST 대하여 설명한다. 3장에서는 GenBIST에 의해 생성된 회로에 대하여 설명하고, 4장에서는 본 논문에서 제안한 GenBIST에 의해 생성된 회로의 성능 및 다양한 메모리에 적용한 결과에 대하여 설명한다.

Ⅱ. GenBIST의 구성

GenBIST는 생성할 메모리 테스트 회로에 대한 설정을 사용자가 정의한 파일로부터 입력으로 받는 부분과, 이 설정을 기반으로 VerilogHDL 코드를 생성하는 부분으로 구성되어 있다. GenBIST의 구성은 그림 1과 같다.

그림 1. GenBIST의 구성
Fig. 1. Structure of GenBIST.

GenBIST의 설정 파일은 생성할 메모리 BIST에 대한 코어 부분을 설정하는 부분과, 읽기/쓰기 동작 시에 제어 신호의 동작에 대하여 정의한 부분, 테스트될 메모리에 대한 정보를 기술하는 부분으로 구성되어 있다. 설정 파일의 구성은 표 1과 같다.

표 1. 설정 파일의 구성
Table 1. Configuration file.

코어부분	- 생성될 모듈 이름 기술 - 알고리즘 기술 - 메모리의 제어 신호 명 기술 - clock, reset 신호 명 기술
읽기/쓰기	- 읽기/쓰기 시의 동작 신호에 대한 기술
메모리	- 메모리의 입출력 포트 이름 및 크기 - 배경 데이터

• 알고리즘 기술

GenBIST 설정 파일의 코어부분에서 알고리즘은 다음과 같은 방식으로 기술한다. 알고리즘의 기술에서 u는 주소가 증가하고 d는 주소가 감소함을 의미한다. 괄호안에 기술된 동작은 한 주소에 대해서 수행될 읽기/쓰기 동작을 의미하며 March 단계라고 칭한다. 각각의 읽기 또는 쓰기 동작 하나를 수행 단계라고 하기로 한다. 다음 예는 3개의 March 단계와 9개의 수행(operation) 단계를 갖는 알고리즘을 보여주고 있다.

ALGORITHM u(w0)u(r0,w1,r1,w0,w1)d(r1,w0,r0)

• 메모리 제어 신호 기술

메모리의 제어 신호의 정의는 쓰기 제어 신호(WRITE_EN), 메모리 선택 제어 신호(RAM_EN), 출력 제어 신호(OUTPUT_EN)에 대해 기술한다. 이 제어 신호의 기술은 실제 메모리의 포트 이름과 assert될 값을 기술한다. 예는 다음과 같다.

WRITE_EN WenB low
RAM_EN Ramen high
OUTPUT_EN OenB high

• 읽기/쓰기 동작 기술

GenBIST 설정 파일에서 읽기/쓰기 부분은 메모리에 읽기와 쓰기를 수행할 때, 각 신호의 타이밍을 기술하는 부분이다. change는 값의 변화를 의미하고, assert는 제어 신호가 활성화되어야 함을 의미한다. 그리고 wait는 다음 클럭을 의미하고, move는 반 클럭 다음을 의미한다.

```
WRITE_CYCLE (
        change ADDRESS;
        change DATA move;
        assert WRITE_EN;
        assert RAM_EN move;
        wait;
        deassert RAM_EN move;
)
READ_CYCLE (
        change ADDRESS;
        assert RAM_EN move;
        assert OUTPUT_EN;
        change REFERENCE move;
        wait;
        deassert RAM_EN move;
        assert COMPARE;
)
```

위의 기술을 파형으로 표현하면 그림 2와 같다.

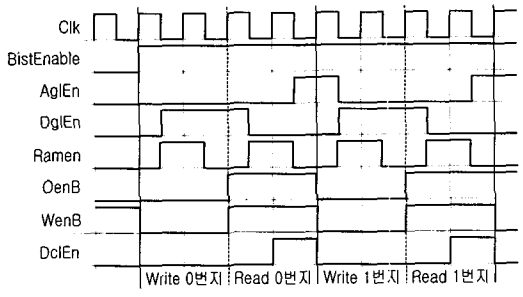

그림 2. u(w0,r0) 동작 파형
Fig. 2. Waveform of u(w0,r0).

COMPARE 타이밍에 의해 생성되는 DclEn 신호는 메모리에서 읽은 값과 reference 데이터를 비교하기 위한 신호이다. 그리고 DglEn 신호는 쓰기 모드에서는 DATA 타이밍, 읽기 모드에서는 REFERENCE 타이밍에 의해 생성된다. DglEn 신호의 경우는 알고리즘을 실행하면서 테스트 데이터 또는 reference 데이터를 변경을 해야할 경우에만 정의된 타이밍에 의해 신호가 생성된다. 예를 들어 u(w0)u(r0,w1,r1,w0,w1)d(r1,w0,r0) 알고리즘을 수행할 경우의 신호는 March2 단계의 첫 번째 수행 단계(r0)는 이전 수행 단계(w0)에서 메모리에 쓰여진 값을 읽는 단계이므로 DglEn 신호가 0값을 유지한다. 그리고 March2 단계의 두 번째 수행 단계(w1)에서는 첫 번째 수행 단계의 반전된 값이 필요하므로 DglEn 신호는 1값을 갖게 된다. 그리고 네 번째 수행 단계(w0)는 세 번째 수행 단계(r1)의 반전된 값이 필요하기 때문에 DglEn 신호는 0값을 갖게 된다. 이와 같은 방식으로 DglEn 신호는 생성되며 신호의 값은 설정 파일에서 정의한 타이밍에 의해 바뀐다.

• 메모리 정보

GenBIST를 위한 메모리 정보는 다음과 같이 기술하며, 메모리의 개수에 따라서 MEMORY1, MEMORY2, ... 같은 방식으로 8개의 메모리까지 지정할 수 있다.

```
MEMORY1 (
        BACKGROUND1 00001111
        BACKGROUND2 00110011
        BACKGROUND3 01010101
        BACKGROUND4 00000000
        TESTENABLE bistenable1
        ADDRESS address 3
        TOMEMORY output 8
        FROMMEMORY input 16
```

BACKGROUND1, BACKGROUND2, ...는 배경 데이터를 의미한다. 그리고 TESTENABLE 신호는 해당 메모리를 테스트하기 위한 제어신호이다. ADDRESS는 해당 메모리의 주소 포트 이름과 크기를 기술한다. TOMEMORY는 해당 메모리의 입력 포트 이름과 크기를 기술하고, FROMMEMORY는 해당 메모리의 출력 포트 이름과 크기를 기술한다.

Ⅲ. 메모리 BIST 회로의 구조

GenBIST에 의해 생성되는 메모리 BIST 회로는 BIST 회로의 동작을 제어하는 모듈(CON), 테스트 패턴이 인가될 메모리의 주소를 생성하는 주소 생성기 모듈(AGL), 테스트 패턴을 생성하는 테스트 데이터 생성기 모듈(TPG), 그리고 테스트 결과를 비교하는 모듈(DCL)로 구성된다. 그림 3은 구현된 BIST 회로의 구조를 보여준다.

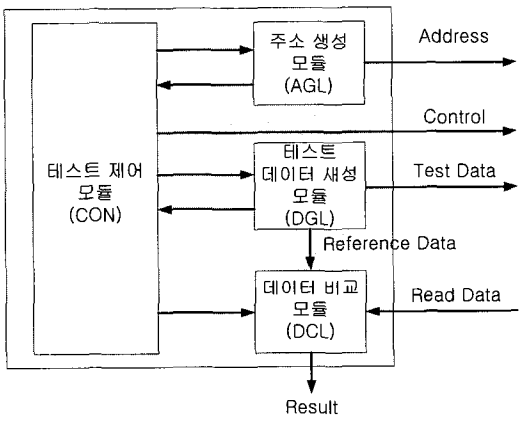

그림 3. 메모리 BIST 회로의 구조
Fig. 3. Structure of memory BIST.

1. 테스트 제어 모듈

CON 모듈은 테스트 모드시에 필요한 제어 신호를 생성하는 모듈로 설정 파일에서 기술한 알고리즘을 위한 수행 상태 머신과 각각의 메모리에 대하여 정의한 배경데이터를 위한 배경데이터 상태 머신을 가지고 있다.

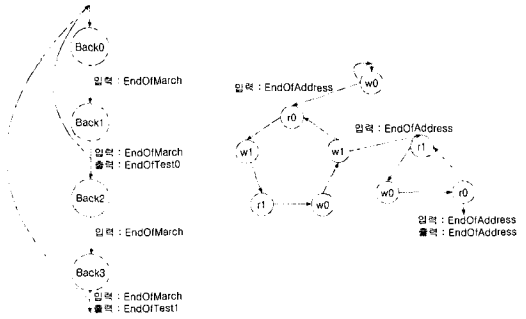

그림 4. 배경데이터 및 수행 상태 머신
Fig. 4. Finite state machine of background and operation.

수행 상태 머신은 AGL 모듈로부터 EndOfAddress가 생성될 때까지 하나의 March 단계를 반복한다. 배경데이터 상태 머신은 수행 상태 머신으로부터 EndOfMarch 신호가 생성되면 다음 상태로 이동하며, 현재 테스트가 수행 중인 메모리에 따라 배경데이터 상태 머신에서 EndOfTest라는 신호를 발생시킨다. 그림 4는 수행 상태 머신과 배경데이터 상태 머신의 동작을 보여주고 있다.

CON 모듈은 수행데이터 상태 머신의 각 상태에서 메모리를 위한 제어 신호, AGL 모듈을 위한 AglEn 신호, DGL 모듈을 DglEn 신호, DCL 모듈을 위한 DclEn 신호를 생성한다. 생성된 AglEn, DglEn, DclEn 신호는 각각 모듈의 동작을 활성화시키는 신호로 주소와 테스트 데이터가 생성되는 시점은 그림 5와 같다.

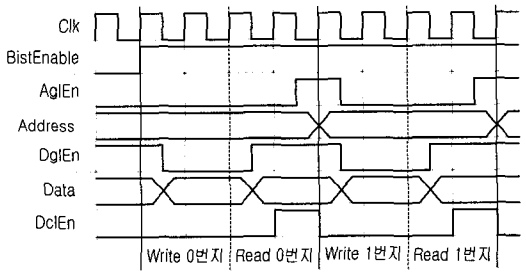

그림 5. AglEn, DglEn, DclEn 신호
Fig. 5. Waveform of AglEn, DglEn, and DclEn.

CON 모듈에서 생성되는 메모리 관련 제어 신호 RamEn, Oen, Wen 신호는 설정 파일에 기술된 내용에 따라서 그림 6과 같은 종류의 파형을 생성할 수 있다.

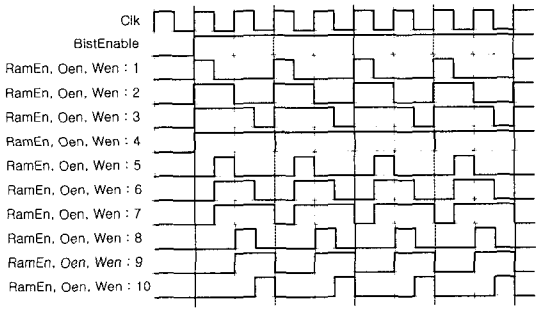

그림 6. CON 모듈에서 생성되는 메모리 관련 제어신호
Fig. 6. Control signals generated from CON.

2. 주소 생성 모듈

AGL 모듈은 테스트 모드시에 CON 모듈로부터 생성

되는 AglEn이라는 신호에 의해 테스트 데이터를 적용할 메모리 주소를 생성하는 모듈이다. 메모리 테스트 알고리즘에서 사용되는 주소는 0번지부터 마지막 번지까지 주소가 증가하는 방향 또는 감소하는 방향으로 변하게 된다. 따라서 본 논문에서는 주소가 0번지부터 마지막 번지까지 계속 반복되는 카운터를 만들고, 감소하는 방향의 주소가 필요한 경우 카운터 출력의 보수를 사용하였다. 또한 여러 개의 메모리가 사용될 경우 현재 테스트가 진행중인 메모리의 주소의 크기에 맞도록 마지막 메모리 주소를 조정할 수 있게 하였다.

그림7은 AGL 모듈의 구조를 보여주고 있으며 EndOfAddress는 CON 모듈의 입력으로 사용되는데, 이 신호는 주소의 마지막까지 동작이 진행되었음을 CON 모듈에게 알리는 역할을 한다. EndOfAddress-Wire0, EndOfAddressWire1, ... 신호는 현재 테스트가 진행중인 메모리의 마지막 주소를 생성할 때 발생되는 신호이다.

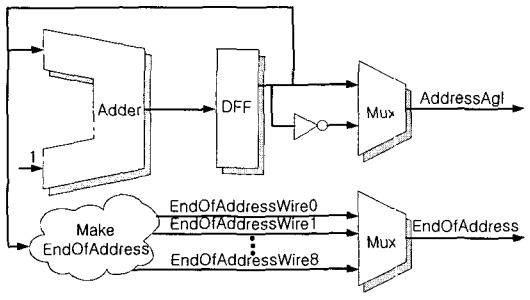

그림 7. AGL 모듈의 구조
Fig. 7. Structure of AGL.

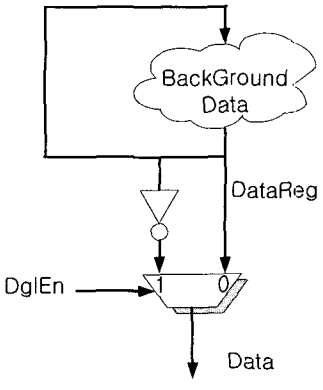

그림 8. DGL 모듈의 구조
Fig. 8. Structure of DGL.

3. 테스트 데이터 생성 모듈

DGL 모듈은 테스트 모드시에 CON 모듈에서 생성되는 DglEn이라는 신호에 의해 테스트 패턴을 생성하는 모듈이다. 그림 8에서 Data라는 신호가 실제 메모리에 인가된 테스트 데이터로, 이 신호는 DglEn 신호의 값이 0인 경우는 DataReg 값이 그대로, 1인 경우는 DataReg 보수 값이 출력된다. DataReg 값은 현재 테스트가 수행중인 메모리를 위한 데이터를 의미하며, 배경데이터를 정의한 메모리인 경우에는 현재의 단계에서 필요한 배경데이터를 의미한다.

4. 데이터 비교 모듈

DCL모듈은 메모리에서 읽혀진 값과 DGL 모듈에서 생성된 reference 데이터 값을 읽어 비교를 수행하는 모듈이다. 모듈의 구조는 그림 9와 같다.

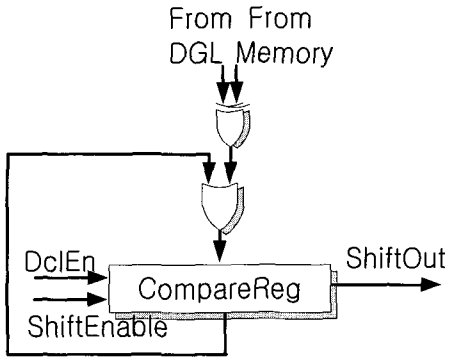

그림 9. DCL 모듈의 구조
Fig. 9. Structure of DCL.

그림에서 CompareReg는 테스트 결과를 저장하고 있는 레지스터이다. 그림에서 알 수 있듯이 테스트 결과는 메모리에서 읽혀진 값과 DGL 모듈에서 생성된 reference 값을 xor 연산을 한 후, CompareReg 값과 or 연산을 수행한 후, 결과를 저장한다. 비교 결과는 클럭의 edge에서 DclEn 신호가 활성화되었을 경우에 CompareReg에 저장된다. 그리고 클럭의 edge에서 ShiftEnable 신호가 활성화되면 CompareReg가 쉬프트 연산을 수행함으로써 ShiftOut 포트로 테스트 결과가 출력된다.

Ⅳ. 실험 결과

본 논문에서 구현한 GenBIST는 PC 기반의 GUI 환

경을 지원하고 있으며, 현존하는 모든 March 테스트 알고리즘 및 사용자가 정의한 알고리즘에 대한 회로를 생성할 수 있다. 그리고 생성된 하나의 BIST 회로는 8개까지의 메모리 테스트를 지원할 수 있다. 그림 10은 실행중인 GenBIST를 보여주고 있고, 그림 11은 생성된 BIST 회로의 일부를 보여주고 있다.

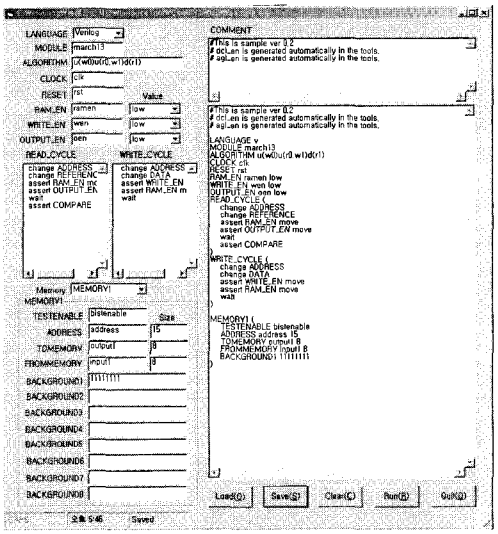

그림 10. GenBIST의 GUI
Fig. 10. GUI of GenBIST.

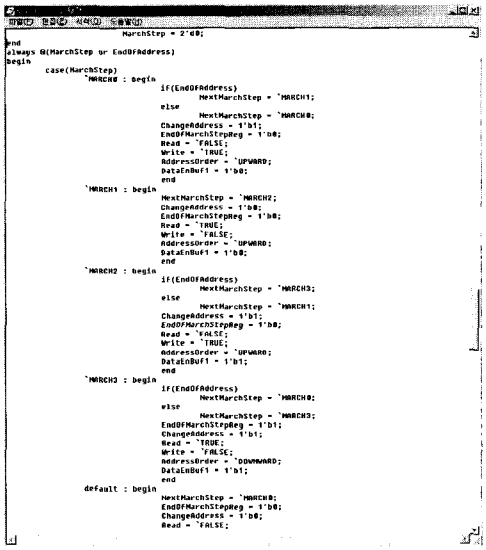

그림 11. GenBIST의 결과
Fig. 11. Result of GenBIST.

본 논문에서는 구현된 GenBIST의 성능 검증을 위하여 잘 알려진 march 테스트 알고리즘에 대한 회로를 생성하고 Verilog XL 컴파일러, 현대 메모리 컴파일러, Xilinx 툴킷을 이용하여 동작 검증을 수행하였다[8]. 검증에 사용된 march 테스트 알고리즘의 고장 검출, 면적, 테스트 수행 시간은 표 2와 같다. 각 알고리즘이 적용된 BIST 회로는 메모리의 주소가 10 bits, 데이터가 8비트, 배경 데이터는 8비트 4개인 하나의 메모리를 위한 회로로 알고리즘에서 ↓는 주소가 감소하는 방향, ↑는 주소가 증가하는 방향, 그리고 ↕는 주소가 증가하거나 또는 감소하는 방향을 의미한다. [11]에서 구현한 BIST 회로와 동일한 규격의 회로를 생성하였을 때, 면적은 [11]에서 구현한 회로에 비해 75.01이 적은 914.99가 나왔다.

표 2. March 테스트 알고리즘의 비교
Table 2. Comparison of march test algorithms.

이름	알고리즘	복잡도	고장검출	면적(cell)*	클럭 수**
Mats	{↕(w0) ↕(r0,w1) ↕(r1)}	4n	Some AFs, SAFs,	530.02	32768
Mats+	{↕(w0) ↑(r0,w1) ↓(r1,w0)}	5n	AFs, SAFs	616.35	40960
Mats++	{↕(w0) ↑(r0,w1) ↓(r1,w0,r0)}	6n	AFs, SAFs, TFs	616.68	49152
MarchX	{↕(w0) ↑(r0,w1) ↓(r1,w0) ↕(r0)}	6n	AFs, SAFs, TFs, CFins	621.02	49152
MarchY	{↕(w0) ↑(r0,w1,r1) ↓(r1,w0,r0) ↕(r0)}	8n	AFs, SAFs, CFins, TFs	616.02	65536
MarchC-	{↕(w0) ↑(r0,w1) ↑(r1,w0) ↓(r0,w1) ↓(r1,w0) ↕(r0)}	10n	AFs, SAFs, TFs, CFins, CFids	642.35	81920
PMOVI	{↓(w0) ↑(r0,w1,r1) ↑(r1,w0,r0) ↓(r0,w1,r1) ↓(r1,w0,r0)}	13n	AF, SAF, TF, SOF, CFid, CFinv	645.02	106496
March1/0	{↑(w0) ↑(r0,w1,r1) ↓(r1,w0,r0) ↑(w1) ↑(r1,w0,r0) ↓(r0,w1,r1)}	14n	AFs, SAFs, TFs	672.35	114688

* Synopsys의 Area Report기능을 이용한 cell만의 면적, ** 복잡도 x 2(읽기 또는 쓰기를 위한 클럭 수) x 주소 x 배경 데이터 수

위에서 기술한 알고리즘과 2절에서 언급한 알고리즘에 대한 회로를 생성하고 동작 검증을 모두 마쳤다. 그리고 2절에서 언급한 알고리즘에 대한 회로를 합성한 후, netlist를 현대 메모리 컴파일러에 의해 생성된 메모리 모델에 적용하여 동작 검증을 수행하였다.

그림 12는 두 개의 내장 메모리를 갖는 메모리 BIST 회로의 동작을 검증하기 위해 구성한 회로의 구성을 보여준다. 그림에서 알 수 있듯이 메모리 BIST 회로의 동작을 활성화시키는 회로는 IEEE 1149.1의 TDI 포트를 통해 IEEE1149.1 회로에 명령어를 적재한 후, 이 명령어를 디코딩하여 활성화 신호를 생성하게 하였다. 또한 테스트 결과는 IEEE1149.1 회로의 ShiftDR 상태에서 TDO 포트를 통해 칩 외부로 출력되게 하였다[9,10].

사용된 메모리는 현대 메모리 컴파일러에 의해 생성된 메모리로 주소는 11비트, 입출력 포트는 32비트의 폭은 갖고 있다. 그리고 프로그램 메모리를 위해 사용된 배경 데이터는 8비트 4개, 데이터 메모리를 위한 배경 데이터는 4비트 2개이다. 구현된 메모리 BIST 회로는 Synopsys를 이용하여 합성되었으며, 합성된 메모리 BIST회로는 그림 13과 같다.

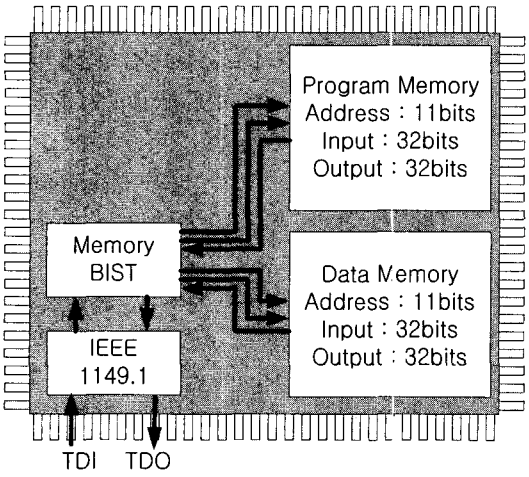

그림 12. 메모리 BIST 회로를 이용한 테스트
Fig. 12. Method of Simulation.

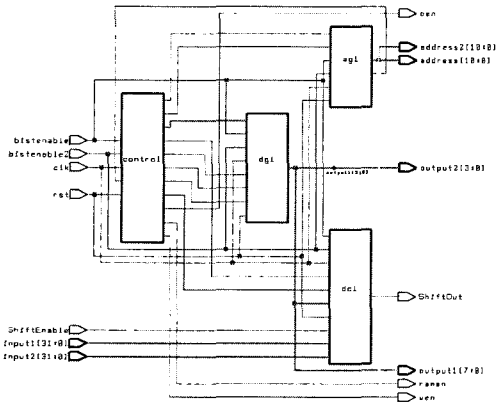

그림 13. 합성된 메모리 BIST 회로
Fig. 13. Synthesis of memory BIST.

그림14는 설정 파일에 의해 생성된 메모리 BIST 회로를 합성한 후, netlist와 현대 메모리 컴파일러에 의해 생성된 메모리를 이용하여 시뮬레이션한 결과로 두 번째 마치 단계의 동작 파형을 보여준다.

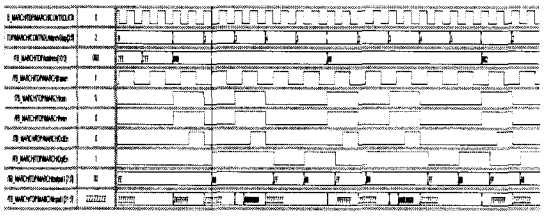

그림 14. 메모리 BIST 회로의 동작 시뮬레이션 결과
Fig. 14. Simulation of memory BIST.

구현된 GenBIST로 생성된 메모리 BIST 회로의 동작을 검증하기 위하여 GenBIST를 이용하여 Xilinx 툴킷에 적재된 메모리를 위한 메모리 BIST 회로를 생성한 후, 툴킷의 FPGA에 생성된 메모리 BIST 회로를 매핑하여 동작을 검증하였다. 그림 13에서 Address는 현재 읽기 또는 쓰기가 수행되는 메모리의 주소를 의미하며, Write Data/Reference Data는 읽기 동작에서는 Reference data를 쓰기 동작에서는 메모리에 쓰기할 데이터를 의미한다. 그리고 Read Data는 메모리로부터 읽혀진 값을 의미한다. 그림 14는 메모리 2C2C 번지에서 FF라는 값을 읽어서 reference 데이터 FF와 비교를 수행하는 동작을 보여주고 있다.

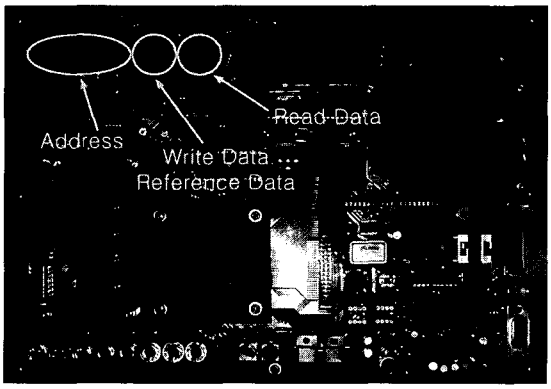

그림 15. Xilinx 툴킷을 이용한 실험
Fig. 15. Experiment using Xilinx tool kits.

V. 결 론

본 논문에서 구현한 설계 자동화 도구인 GenBIST는 사용자가 설정한 내용에 따라서 behavioral Verilog HDL로 기술된 메모리 BIST 회로를 자동 생성하여 준다. 기존의 방법들은 대부분 메모리 테스트 알고리즘을 라이브러리화하고 이를 이용하여 회로를 생성하여 주는 방식을 취하고 있다. 이러한 방법은 새로운 테스트

알고리즘이나 툴에서 지원하지 않는 알고리즘을 적용할 수 없다는 단점이 있다. 본 논문에서 구현한 GenBIST는 사용자가 메모리 테스트 알고리즘 및 메모리 동작 신호에 대하여 정의하고 이를 이용하여 메모리 BIST 회로를 생성함으로써 이러한 문제를 해결하였고, 다중 메모리 모듈을 지원하게 함으로써 메모리 BIST 회로로 인한 오버헤드를 최적화하였으며, 다양한 크기의 폭을 갖는 주소와 데이터를 지원하고, IEEE 1149.1 회로와의 인터페이스도 고려하였다. GenBIST를 이용하여 메모리 테스트를 위한 회로를 생성함으로써 메모리 BIST 회로의 설계에 소요되는 많은 시간과 노력을 줄일 수 있다. 특히, 메모리 사양의 변경에 따른 메모리 BIST 회로의 재설계를 위한 시간과 노력을 줄일 수 있다.

참 고 문 헌

[1] M. Abramovici, M. A. Breuer and A. D. Friedman, Digital system testing and testable design, Computer Science Press, 1990.

[2] Test Technology Standards Committee, IEEE Standard Test Access Port and Boundary Scan Architecture, IEEE Computer Society Press, 1990.

[3] R. Rajsuman, Digital Hardware Testing : Transistor-Level Fault Modeling and Testing, Artech House Boston London, 1992.

[4] 홍성제, 강성호, 박은세, 장훈, 최호용, 테스팅 및 테스팅을 고려한 설계, 반도체 설계 교육 센터, 1997

[5] H.Bleeker and P. Eijnden, Boundary-Scan Test-A Practical Approach, Kluwer Academic Publishers, 1993.

[6] Memory BistCoreTM User's Reference Manual, GeneSys TestWare, Revision 1.4, June, 1998.

[7] 장종권, "ASIC에 실장되는 다중 RAM 모듈 테스트를 위한 BIST 회로 생성기의 구현," 한국정보처리학회 논문지, 제5권, 제6호, pp. 1633~1638, 1998

[8] I. Schanstra and A. Goor, "Industrial Evaluation of Stress Combination for March Tests applied to SRAMs," International Test Conference, 1999.

[9] Kenneth P. Parker, THE BOUNDARY-SCAN HANDBOOK, Kluwer Academic Publishers, 1992.

[10] H. Bleeker, P. Eijnden and F. Jong, Boundary-Scan Test-A Practical Approach, Kluwer Academic Publishers, 1993.

[11] 양선웅, 박재홍, 장훈, "IEEE 1149.1을 이용한 March 알고리듬의 내장형 자체 테스트 구현," 한국정보과학회 논문지, 제7권, 제1호, 2001

저 자 소 개

梁 善 雄(正會員) 第36券 C編 第8號 參照

張 勳(正會員) 第36券 C編 第8號 參照