공학 연구논문집 제30권 1호 pp. 299~313, 울산대학교.
Journal of Engineering Research Vol. 30, No. 1, pp. 299~313, 1999
University of Ulsan.

Backplane Interconnect 테스트를 위한 BIST 회로 설계

이형운 · 장종권
울산대학교 컴퓨터 · 정보통신공학부

<요 약>

Backplane은 여러 개의 보드(board), 즉 부시스템(sub-system) 간의 연결을 제공하는 전형적인 방법이다. 각 보드들은 Backplane의 Interconnect를 통하여 서로 신호를 교환한다. Backplane은 전체 시스템의 주 통신 링크(main communication link)이기 때문에 Interconnect의 고장(fault)은 전체 시스템을 오동작하게 한다. Interconnect의 고장은 보드를 제거하거나, 교체, 또는 추가할 경우에 발생될 수 있다. Backplane Interconnect 테스트는 이러한 고장을 검출하여 시스템의 동작을 검증할 수 있는 중요한 과정이다. 본 연구에서는 BIST(Built-In Self-Test) 기법을 채택하여 테스트 장비의 사용으로 발생될 수 있는 비용을 최소화하였고 Interconnect의 고장을 자동으로 검출할 수 있게 하였다. 고장 검출을 위해 N+1 알고리즘을 하드웨어로 구성하여 다중 고장에 대해 완전한 검출이 이루어질 수 있도록 설계하였다. 그리고 고장 점검 및 진단을 위한 테스트 패턴 주입을 위해 보드 수준의 테스트에 사용되는 IEEE Std 1149.1 Boundary Scan Architecture 기법을 적용하였다. 시스템 수준에서는 시스템의 다양한 사양에 적용이 용이하도록 설계하는 것이 중요하다. 본 논문의 BIST 회로는 카운터 크기의 변경만으로 전체 회로의 수정 없이 적용될 수 있게 하였다.

BIST Circuit Design for Backplane Interconnect Test

Hyung-Woon Lee and Jong-Kwon Chang
Dept. of Computer Information Communication, Univ. of Ulsan

<Abstract>

Backplane is a well-known method for inter-board connections. These boards are usually plugged into the backplane of the communication systems and are used to

exchange signal with each other via backplane interconnects. Since the backplane is the main communication link, its error free operation is crucial to the system's operability. Generally, faults can be introduced whenever a board is removed or replaced or added. This backplane interconnect testing is a very important process to verify the operation of system. In this paper, the BIST(Built-In Self Test) methodology is adopted to minimize the cost of test equipments and to detect faults automatically. This BIST circuit was implemented using the N+1 algorithm to detect multiple faults and the IEEE Std. 1149.1 Boundary Scan Architecture methodology to inject test patterns. At the system level, the BIST circuit must be designed to be easily modified for the potential system configuration. By changing the size of counter only, the proposing BIST circuit can be adapted to many types of system configurations without any modification of entire system.

1. 서 론

Backplane은 전체 시스템의 주 통신 링크(main communication link)이기 때문에 Interconnect의 고장(fault)은 전체 시스템을 오동작하게 한다. Interconnect의 고장은 보드를 제거하거나, 교체, 또는 추가할 경우에 발생될 수 있다.[1] Backplane Interconnect 테스트는 이러한 고장을 검출하여 시스템의 동작을 검증할 수 있는 중요한 과정이다.

본 논문에서는 Backplane Interconnect를 테스트하기 위하여 BIST(Built-In Self-Test) 기법을 사용하였다. BIST 기법은 부가적인 면적을 요구하지만 테스트 장비의 사용으로 인하여 발생하는 비용을 최소화할 수 있으며 Interconnect의 고장을 자동으로 검출할 수 있게 한다.

시스템 수준에서의 테스트는 보드 수준, 즉 칩(chip) 사이의 연결선을 테스트하는 방법과 유사하므로 보드 수준에서 사용되는 IEEE 1149.1 Boundary Scan Architecture를 적용하여 테스트할 수 있다.[1,2] 그리고 각 보드들에 있는 BIST 회로와의 동기화(synchronization), 테스트 Scheduling, 테스트를 위해 필요한 정보를 교환하기 위하여 시스템 버스 이외에 부가적으로 테스트 버스(Test bus)를 사용하였다.

테스트를 위한 회로는 BIST와 테스트 버스 컨트롤러(Test bus controller)로 구성된다. 첫째로 BIST 회로는 Master BIST와 Slave BIST의 두 종류로 나눌 수 있다. Master BIST는 테스트 패턴(pattern)을 생성하여 보드의 BSC(Boundary Scan Cell)에 인가하고, 각 보드의 Response를 분석하여 고장 검출 기능을 수행한다. Slave BIST는 Backplane 버스를 통해 전달된 패턴을 받아 테스트 버스를 통해 Master로 전송하는 기능을 수행한다. 둘째로 테스트 버스 컨트롤러는 Master 버스 컨트롤러와 Slave 버스 컨트롤러의 두 종류로 나눌 수 있다. Master 버스 컨트롤러는 테스트 Scheduling, 테스트 동기화의 기능을 수행한다. Slave 버스 컨트롤러는 테스트 버스를 통해 전달되는 주소를 비교하여 자신의 주소와 같을 경우 테스트 회로를 Enable 시키는 기능을 수행한다.

제안한 BIST 회로는 다중 고장의 검출을 위해 N+1 알고리즘을 사용하였다. N+1 알고리즘은 다중 고장에 대해 완전한 검출과 고장 분석이 가능하다.[7,8] 시스템 수준에서는 시

스템의 다양한 사양에 적용이 용이하도록 설계하는 것이 중요하다. 시스템 사양에 쉽게 변경시킬 수 있도록 컨트롤러의 주요 부분에 카운터를 사용하였다. 시스템 버스의 크기나 Slave 보드의 수에 따라 전체 회로의 수정 없이 카운터만 변경시켜 사용할 수 있다.

본 논문에서는 Backplane Interconnect 테스트를 위한 BIST 회로 설계를 위하여 다음과 같은 내용을 다루고자 한다.

제 2장에서는 테스트를 위한 회로의 전체 구성과 각 부분의 기능에 대해 개략적으로 설명한다. Interconnect 테스트를 위해 필요한 기본 원리인 Boundary Scan Architecture, 고장 모델, 테스트 알고리즘은 제 3장에서 다룬다. 그리고 제 4장에서는 BIST 회로의 기능과 동작, 테스트 수행 과정, 시뮬레이션 결과에 대해 설명한다. 끝으로 제 5장에서는 결론을 맺고 향후 연구 방향을 제시한다.

2. 시스템의 개요

2.1 Backplane 시스템

대형 산업 시스템의 경우 여러 개의 부 시스템의 조합으로 구성된다.

Backplane 시스템은 [그림 2.1]과 같이 여러 개의 보드, 즉 부 시스템간의 연결을 제공하는 전형적인 방법이다. 각 보드들은 슬롯(slot)형태로 Backplane에 연결되어 동작을 하게 된다. Backplane 버스는 각 보드들간의 신호교환을 위해 이용된다.

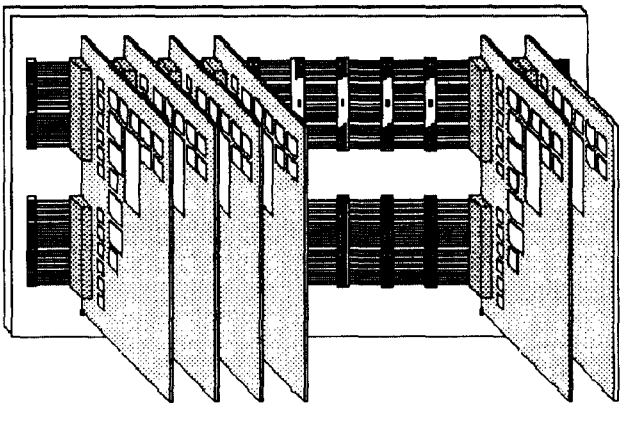


그림 2.1 Backplane 시스템

2.2 전체 시스템의 구조

[그림 2.2]는 Backplane Interconnect의 테스트를 위한 전체 시스템의 구성을 나타낸 것이다.

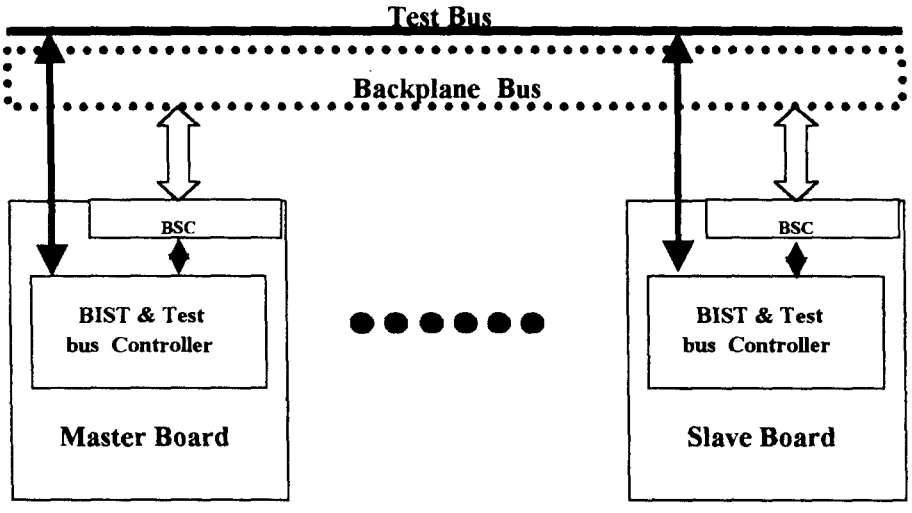


그림 2.2 Backplane 시스템의 테스트를 위한 전체 시스템의 구성도

[그림 2.2]에서 보는 것과 같이 각 보드는 크게 BIST와 버스 컨트롤러의 두 부분으로 나눌 수 있다. 부가적으로 Backplane Interconnect의 테스트를 위해 별도의 테스트 버스를 사용하고 있다. 각 부분의 기능은 다음과 같다.

2.2.1 BIST (Built-In Self-Test)회로

BIST 회로는 테스트 패턴 생성, 고장 검출의 기능을 수행한다.

(1) 테스트 패턴 생성(Test pattern generation)

테스트 패턴은 Master 보드의 TPG (Test pattern generator)에서 생성하여 BSC에 인가하게 된다. BSC는 테스트 패턴을 Backplane 버스에 인가하여 테스트 Scheduling에 의해 Enable된 Slave 보드의 BSC에 전달되게 된다.

(2) 고장 검출(Fault detection)

Slave 보드에 인가된 테스트 패턴은 테스트 버스를 통해 Master의 ORA(Output Response Analyzer)에 전달된다. ORA는 Master 보드의 TPG에서 생성된 테스트 패턴과 Backplane 버스를 통해 Slave 보드에 인가된 패턴을 비교하여 인가된 패턴과 전달받은 패턴이 다를 경우 고장 신호를 내보내게 된다.

2.2.2 테스트 버스 컨트롤러(Test bus controller)

테스트 버스 컨트롤러는 BIST 회로에 의해 제어되며 테스트 Scheduling의 기능을 수행한다. 테스트 버스 컨트롤러의 주소 생성기(Address generator)는 테스트 대상이 되는 Slave

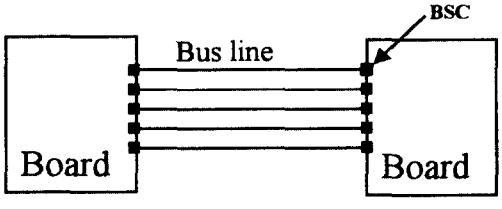
보드의 해당 주소를 생성하게 된다. 생성된 주소는 테스트 버스를 통해 각 Slave 보드의 버스 컨트롤러에 인가되며 주소가 일치하는 보드는 테스트를 위해 Enable 되게 된다. 테스트는 한번에 하나의 Slave 보드에 대해서만 수행한다.

3. Backplane Interconnect 테스트

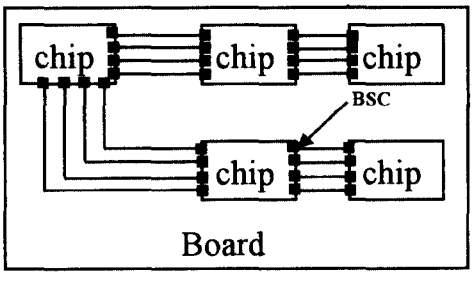
이 장에서는 Backplane Interconnect 테스트를 위해 필요한 기본 원리와 고장 모델 그리고 테스트 알고리즘에 대해 기술한다.

3.1 BSA (Boundary Scan Architecture)

Boundary Scan 설계는 다른 패키지(package), 즉 칩, 보드, 모듈 등의 수준에서 고장 점검 및 진단을 위한 테스트 패턴의 주입을 쉽게 하기 위하여 개발된 테스트 설계 기술이다. Boundary Scan은 각각의 칩에 전용 Boundary Scan 레지스터(Register)를 사용하여 칩 등의 입출력 핀에 테스트 패턴을 직접 주입 및 관측할 수 있게 한 보드 수준에서의 테스트 설계 기술이다. 현재 별도의 테스트 프로토콜에 의한 IEEE 1149.1 Boundary Scan이 업체의 표준으로 선택되어 있어 제작되는 비메모리 칩에 널리 내장 설계되고 있다.[3-7]



(1) 시스템 수준 테스트



(2) 보드 수준 테스트

그림 3.1 시스템 수준과 보드 수준에서의 연결선 테스트

시스템 수준에서의 Backplane Interconnect 테스트는 보드와 보드간의 연결선을 테스트하는 것으로 보드 수준에서의 연결선 테스트와 유사하기 때문에 Boundary Scan 기법을 그대로 적용할 수 있다.[1,2] [그림 3.1]

3.1.1 BSC (Boundary Scan Cell) 구조

BSC는 칩이나 보드 모듈의 입출력 부분에 위치하여 다른 칩이나 모듈로의 연결선상의 논리 값을 인가하거나 관찰할 수 있게되어 있다.

BSC의 기본 구조는 [그림 3.2]에서와 같이 2개의 멀티플렉서(multiplexor)와 2개의 플립플롭(Flip-Flop)으로 구성되어 있다.

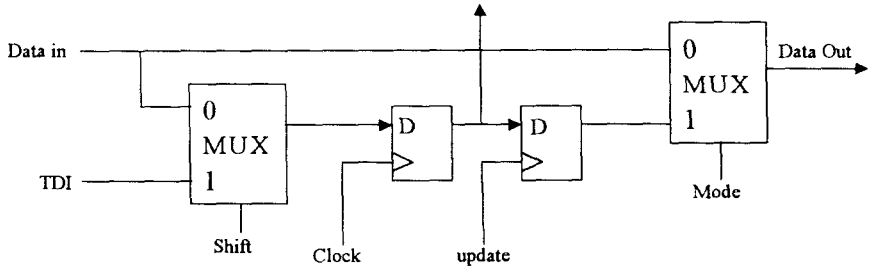


그림 3.2 Boundary Scan Cell 구조

첫 번째 멀티플렉서는 테스트 데이터의 주입, 다른 모듈에서 보낸 데이터 또는 모듈 내부에서 전달되는 신호를 선택하는 역할을 한다.

두 번째 멀티플렉서는 다른 모듈로 테스트 데이터를 전송하거나 BSC를 거치지 않고 데이터를 전송하는 기능을 선택하는 역할을 한다.

TDI(Test Data Input)는 TPG에서 생성된 테스트 데이터가 입력되는 부분이며 TDI에서 입력된 신호는 첫 번째 플립플롭을 거쳐 TDO(Test Data Output)로 보내진다. TDO는 다음 BSC단의 TDI로 입력되거나 ORA에서 고장 검출을 위한 신호로 사용된다.

첫 번째 플립플롭의 Clock 신호에 의해 멀티플렉서에서 받은 신호를 TDO 쪽으로 내보내게 된다.

두 번째 플립플롭의 Update 신호에 의해 두 번째 멀티플렉서의 Mode 신호가 1일 때 BSC에 저장된 신호를 다른 모듈이나 모듈 내부로 전달하게 된다.

3.1.2 BSC의 동작

BSC의 동작은 No Operation, Shift, Update, Capture로 구분할 수 있다. [표 3.1]

표 3.1 BSC 동작 모드

동작 모드	shift	Clock	update	mode
No Operation	X	0	0	0
Shift	0 or 1	⎍	0	X
Update	0 or 1	⎍	⎍	1
Capture	0	⎍	0	X

- No Operation 모드
  No Operation 모드는 시스템이나 칩이 일반적인 동작을 할 때 사용하는 모드이다. 이 모드에서 BSC는 동작을 하지 않고 Data In 이 Data Out으로 바로 연결된다.
- Shift 모드
  Shift 모드는 TPG 등으로부터 전달된 테스트 데이터를 BSC에 인가할 때 사용한다. BSC에 인가된 테스트 데이터는 첫 번째 플립플롭의 Clock에 의해 다음 단의 BSC로 전달된다.
- Update 모드
  Update 모드는 BSC에 인가된 테스트 데이터를 모듈의 내부나 다른 모듈로 연결되는 전송선로에 인가할 때 사용한다.
- Capture 모드
  Capture 모드는 전송 선로에 전달된 신호를 BSC에 인가하여 Shift나 기타 모드에 의해 고장 판단을 할 때 사용한다.

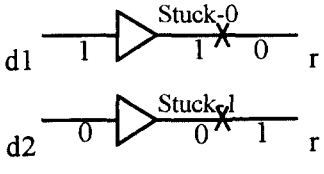
3.2 고장 모델 (Fault model)

보드나 칩간의 연결선상에서 발생 가능한 고장의 종류에는 Stuck-at 고장, Short 고장 그리고 Open 고장이 있다. [1,7,8,9]

3.2.1 Stuck-at 고장

Stuck-at 고장은 접지나 전원이 고착된 고장을 말하며 Stuck-at 0 와 Stuck-at 1 의 두 가지 형태로 분류할 수 있다. [그림 3.3]

- Stuck-at 0 : 논리적으로 0 의 값을 가진다.
- Stuck-at 1 : 논리적으로 1 의 값을 가진다.



Stuck-at fault
그림 3.3 Stuck-at 고장

3.2.2 Short 고장

Short 고장은 Bridging 고장이라고도 하며 두 선로가 물리적으로 서로 연결되어 발생하는 고장을 말한다. Bridging 고장에는 Wired-AND, Wired-OR, Dominator 형태가 있다. [그림 3.4]

● Wired-AND 형태
Wired-AND 형태는 두 선로 중 하나가 논리적으로 0 이면 두 선로 모두 0 의 값을 가진다.

● Wired-OR 형태
Wired-OR 형태는 두 선로 중 하나가 논리적으로 1 이면 두 선로 모두 1 의 값을 가진다.

● Dominator 형태
Dominator 형태는 지배적인 선의 입력 값이 두 선 모두에게 영향을 주어 두 선로 모두 지배적인 선의 값을 가진다.

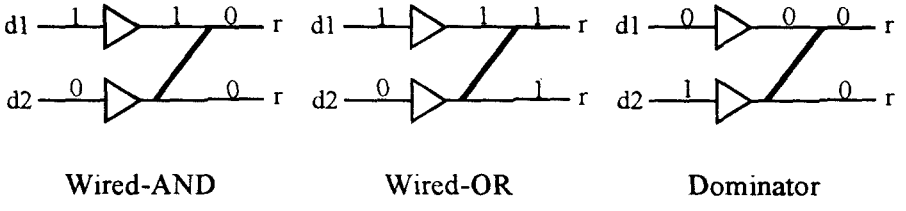


그림 3.4 Short 고장

3.2.3 Stuck-open 고장

Stuck-open 고장은 연결선이 끊어진 상태를 나타낸다.

Open 고장이 발생하면 연결선상의 논리 값은 초기 값을 유지하므로 Stuck-at 고장으로 모델링(Modeling)할 수 있다.

3.3 테스트 알고리즘 (Test algorithm)

연결선 테스트를 위한 테스트 알고리즘에는 Log(N), 2Log(N), N+1 등의 알고리즘이 사용되고 있다.[7,8]

3.3.1 Log(N) 알고리즘

N-bit으로 구성된 모듈에 대해 Log(N) 개의 패턴을 생성한다.

이 알고리즘은 Stuck-at 고장을 제외한 모든 Short 고장을 검출할 수 있다.

3.3.2 2Log(N) 알고리즘

Log(N) 계열 알고리즘은 연결선 상의 고장을 검출하기 위한 목적으로 사용되었다. 하지만 2Log(N) 알고리즘은 검출과 분석을 위한 목적으로 사용된다.

이 알고리즘은 모든 고장을 검출할 수 있으며 부분적인 분석이 가능하다.

3.3.3 N+1 알고리즘

N+1 알고리즘은 모든 고장에 대해 검출이 가능하며 다중 고장에 대해서도 완전한 분석이 가능하다.

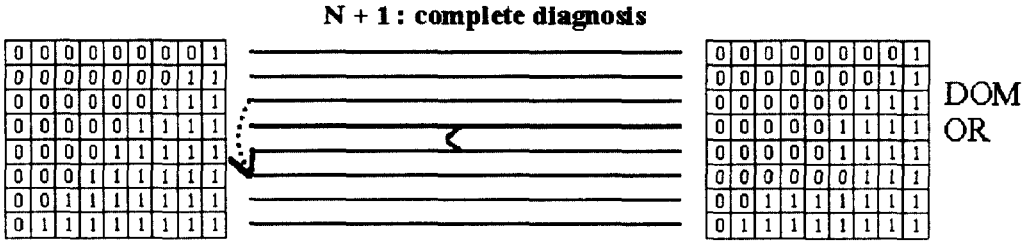


그림 3.5 N+1 알고리즘

[그림 3.5]와 같이 8-bit의 연결선에 대해 9가지의 패턴을 생성한다.
본 논문에서는 N+1 알고리즘을 사용하여 테스트 패턴을 생성하였다.

4. BIST 회로 설계

BIST를 이용한 테스트 방법은 단일 보드나 단일 칩 내에서 이루어져 왔다. Backplane 시스템에서는 서로 다른 보드간의 테스트를 위하여 각 보드에 BIST 회로를 내장해야 한다.

BIST 회로는 부가적인 면적을 최소화하고 테스트의 효율성을 위해 Master-Slave 구조를 가진다. 패턴의 생성과 고장 검출, scheduling은 Master 보드의 BIST 가 담당하며 Slave 보드의 BIST 회로는 연결 선로에 의해 전달된 신호를 Master 쪽으로 보내주는 역할을 한다.

각 BIST 회로의 동기화를 위해 시스템 Clock과는 별도로 테스트용 Clock을 사용하여 테스트 버스를 통해 Master와 Slave BIST에서 공통으로 사용한다.

4.1 BIST 회로 구조

BIST 회로는 Master 보드에 사용되는 회로와 Slave 보드에 사용되는 회로로 구성된다.

4.1.1 Master BIST

Master BIST는 테스트 패턴을 BSC에 전달하여 연결선을 통해 Slave 보드의 BSC로 전송하고 테스트 버스에서 전달되는 Slave 보드의 패턴을 생성된 패턴과 비교하여 고장 검출의 기능을 수행한다.

Master BIST 회로는 테스트 버스 컨트롤러와 TPG, ORA, Master 컨트롤러로 구성된다.
[그림 4.1]은 Master BIST의 구조를 나타내고 있다.

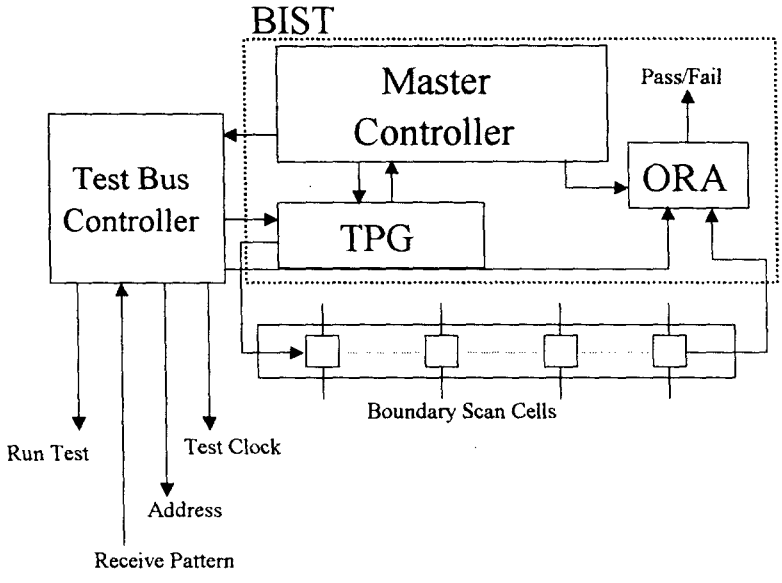


그림 4.1 Master BIST 구조

4.1.2 Slave BIST

Slave BIST는 Master BSC에서 연결선에 인가한 테스트 패턴을 BSC를 이용하여 capture하고 테스트 버스를 통해 capture한 패턴을 Master로 전송하는 기능을 수행한다.

Slave BIST는 Slave 컨트롤러와 테스트 버스 컨트롤러로 구성된다.

[그림 4.2]는 Slave BIST의 구조를 나타내고 있다.

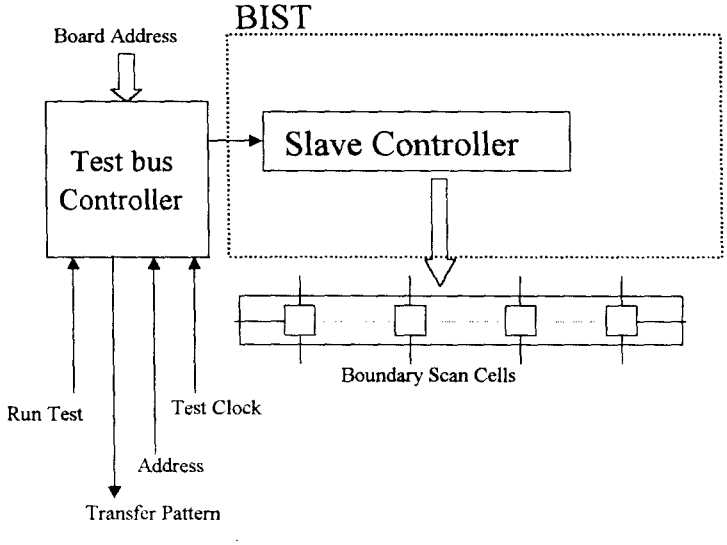


그림 4.2 Slave BIST 구조

4.3 테스트 수행 과정

[그림 4.3]은 Master와 Slave의 테스트 수행 과정을 도식화 한 것이다.

테스트 시작 신호가 들어오면 RUN_TEST를 High로 만들어 Master와 Slave의 회로가 테스트 모드로 들어가게 된다.

Master 회로의 버스 컨트롤러에서 첫 번째 주소 생성이 완료되면 TPG가 테스트 패턴을 생성하고 BSC에 인가하게 된다. 이때 생성된 주소와 일치하는 Slave 회로는 Enable 된다.

첫 번째 패턴이 Interconnect 버스에 인가되면 Master는 다음 패턴 생성을 계속하고, Slave는 그 패턴을 자신의 BSC로 capture하여 테스트 버스를 통해 패턴을 전송한다.

테스트 버스를 통해 전송된 패턴을 Master의 ORA에서 고장 검출과정을 수행한다.

첫 번째 보드의 패턴 인과 과정이 끝나면 Master 버스 컨트롤러는 두 번째 주소를 생성시킨다. 이때 첫 번째로 Enable된 Slave 보드는 Disable되고 두 번째로 생성된 주소와 일치하는 Slave 회로가 Enable 된다.

모든 과정이 끝나면 RUN_TEST 신호가 Low가 되고 Master와 Slave의 회로는 테스트 과정을 끝내게 된다.

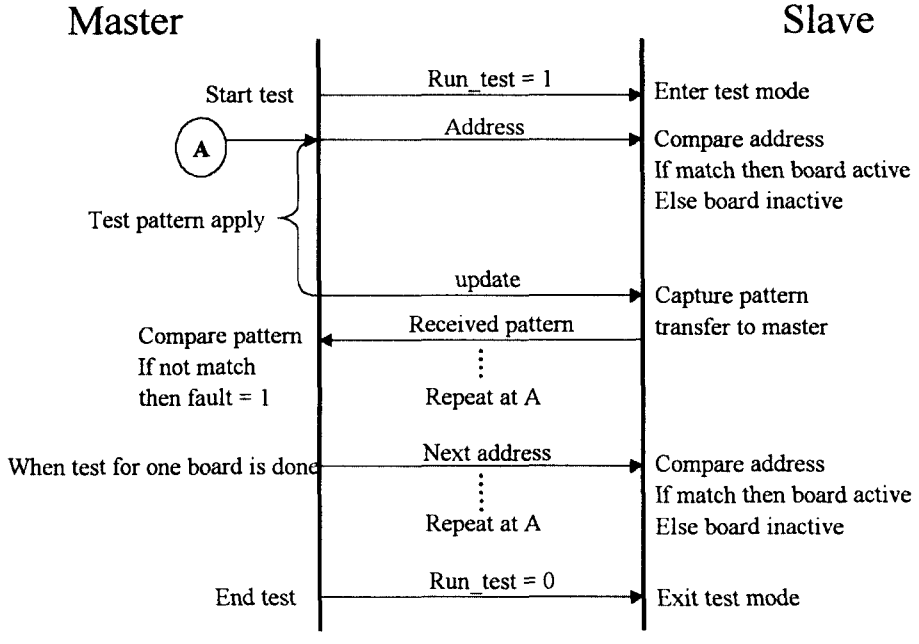


그림 4.3 테스트 수행 과정

4.4 시뮬레이션

본 논문에서 제안한 BIST 회로는 Xilinx사의 Foundation 1.4를 이용하여 설계와 시뮬레이션을 하였다.[10] 시뮬레이션은 고장이 없는 상태와 Stuck-at 고장, 다중 Short고장에 대해 수행하였다. TCK의 주기는 20ns이며 첫 번째 Slave 보드의 주소는 (0,1), 두 번째 Slave 보드의 주소는 (1,0)이다.

[그림4.4~4.6]은 3가지 경우에 대한 시뮬레이션 결과를 나타내고 있다. 고장 검출은 MTO(Master Test data Output)와 STO(Slave Test data Output) 출력 값의 비교에 의해 수행되며 두 신호의 출력이 다를 경우 FAULT 출력이 HIGH가 되어 고장 발생을 표시한다.

[그림 4.4]는 고장이 없는 상태의 결과이며 Backplane 버스를 통해 전달된 Master 패턴의 출력(MTO)과 Slave에서 보내온 패턴의 출력(STO)이 서로 같음을 알 수 있다.

Backplane Bus에서 발생한 Stuck-at 1 고장이 두 번째 보드에 영향을 주는 상태의 결과와 두 번째 보드의 테스트 과정에서 MTO의 값과 STO의 값의 차이를 [그림 4.5]에서 확인 할 수 있다.

[그림 4.6]은 다중 고장이 발생했을 경우를 나타내낸다. [그림 4.5]와 [그림 4.6]의 고장 발생 위치는 STO 값의 분석을 통해 확인 할 수 있다.

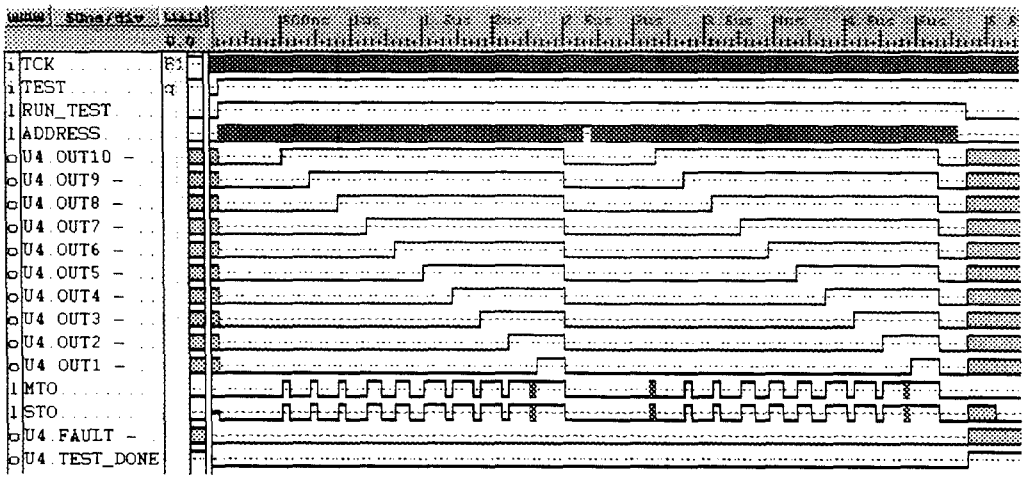


그림 4.4 Fault free 상태의 결과

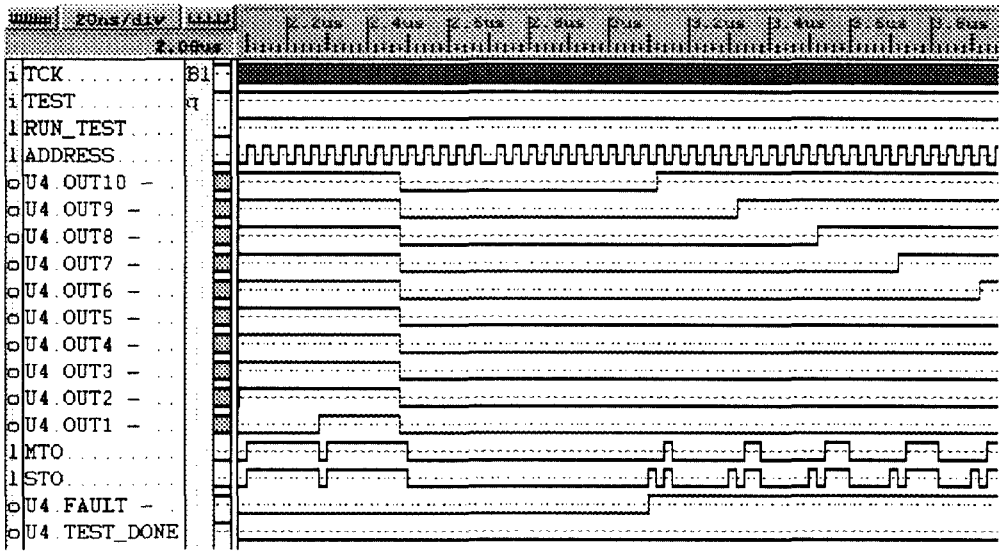


그림 4.5 두 번째 Slave 보드에서 Stuck-at 1 고장이 발생한 경우

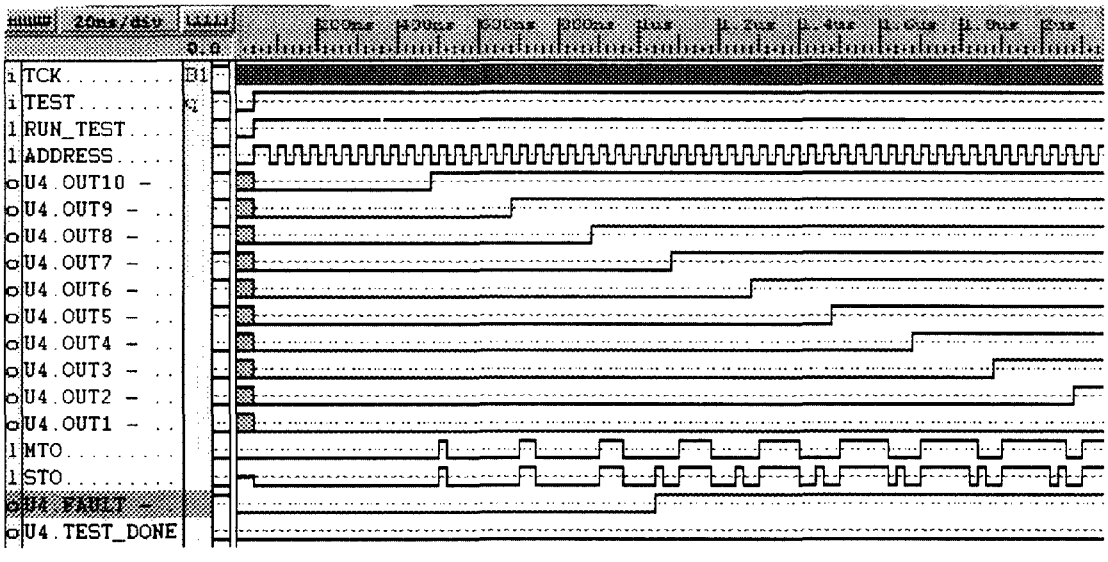


그림 4.6 Wired-AND 와 Wired-OR 고장이 발생한 경우

5. 결 론

본 논문에서는 Backplane 시스템의 Interconnect 버스를 테스트하기 위한 BIST 회로를 제안하였다. 제안한 BIST 회로는 테스트를 위해 필요한 부가적인 면적을 최소화하고 시스템 사양에 쉽게 변화시킬 수 있으며 다중 고장에 대한 완전한 검출이 가능하도록 설계하였다.

먼저 부가적인 면적을 최소화하기 위해 테스트 패턴 생성과 고장 검출 등의 주요 기능을 Master에서 수행하게 하고 Slave는 전달된 패턴을 Capture하여 Master로 전송하는 최소한의 기능을 수행하도록 하였다. 무엇보다도 Master에 부가되는 부하를 줄이기 위해 테스트를 위해 필요한 최소 기능만을 사용하였다.

두 번째로 시스템 사양에 쉽게 변화시키게 하기 위하여 컨트롤러의 주요기능을 카운터를 사용하여 설계하였다. 시스템 버스의 크기나 Slave 보드의 개수에 따라 전체 회로의 수정 없이 카운터만 변경시켜 사용 가능하게 하였다.

끝으로 고장 검출을 위해 N+1 알고리즘을 하드웨어로 구성하여 다중 고장에 대해 완전한 검출이 이루어질 수 있도록 설계하였다.

향후 연구과제는 첫째로 다양한 사양의 Slave 보드들로 구성된 Backplane 시스템을 설계자의 회로 수정 없이 사용할 수 있는 방법을 연구하는 것이다. 다양한 사양의 Slave 보드를 테스트하기 위해서는 BIST 회로 내부에 메모리 소자를 사용하여 각각의 Slave 보드들이 사용하고 있는 Backplane 버스의 크기를 기억시켜 Slave 보드의 크기에 적절한 패턴을 생성해야 한다.

두 번째 연구과제로는 Backplane Interconnect 테스트와 보드 수준의 테스트를 동시에 수행할 수 있는 BIST 회로의 개발이다. 시스템 수준의 테스트와 보드 수준의 테스트를 동시에 수행하기 위해서는 설계 초기에 테스트에 필요한 기능을 정확히 정의하고 회로의 유연성을 극대화하여 컨트롤러의 크기를 최소화하고 테스트 Scheduling의 간략화에 중점을 두어야 한다.

<참고 문헌>

[1] Chen-Huan Chiang and Sandeep K. Gupta, "BIST TPG for Faults in System Backplanes", IEEE/ACM International Conference on Computer Aided Design, pp.406-413, 1997.

[2] C. Su, S.-J. Jou, and Y.-T. Ting, "Decentralized BIST for 1149.1 and 1149.5 Based Interconnects", In proceedings European Design and Test Conference, pp.120-125, 1996.

[3] Kenneth P. Parker, "The Boundary-Scan Handbook", Kluwer Academic Publishers, 1992.

[4] Parag K. Lala, "Digital Circuit Testing and Testability", Academic Press, 1997.

[5] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman, "Digital Systems Testing and Testable Design", IEEE Press, 1990.

[6] Texas Instruments, "IEEE Std 1149.1 (JTAG) Testability Primer", Texas Instruments Inc., 1997.

[7] 반도체설계교육센터(IDEC), "디지탈 VLSI 테스팅 기초", 반도체설계교육센터, pp.201-271, 1996.

[8] Sungju PARK and Gueesang LEE, "Complete Diagnosis Patterns for Wiring Interconnects", IEICE TRANS. Fundamentals, Vol.E81 A. No.4, pp.672-676, April. 1998.

[9] Wuudiann Ke, "Backplane Interconnect Test In A Boundary-Scan Environment", IEEE International Test Conference, pp.717-724, 1996.

[10] XILINX, "The Programmable Logic Data Book", XILINX Inc., 1996.