특 집

고성능 VLSI의 테스트 용이화를 위한 설계 방법

조 창 현, 김 헌 철
삼성전자(주) System LSI 사업부 CPU사업팀

I. 서 론

Design for Testability는 회로의 설계 단계에서부터 테스트를 고려함으로써 효율적인 테스트가 가능하도록 해주는 설계 방법이다. 고성능 VLSI들이 원하는 Quality Goal과 Manufacturability를 만족하기 위해서는 Design for Testability가 고려되어야만 한다.

본 글은 Design for Testability의 이해를 돕기 위한 여러 개념의 소개와 실제 적용 예를 보여주고자 한다. 이를 위하여 먼저 고장 모델, 고장의 분류, Fault Coverage 등에 관하여 언급하고, Fault Coverage를 얻기 위한 고장 시뮬레이션과 효율적인 테스트 패턴의 생성을 위한 ATPG(Automatic Test Pattern Generation)의 과정에 관하여 보여준다. 다음으로 Testability의 개념과 대표적인 Structural Design for Testability 방법인 Scan Design, BIST(Built-In Self-Test), Core-based Design 방법을 소개하며 이들의 적용 예를 보여주고 고성능 VLSI의 대표적인 예인 Alpha 21164와 같은 마이크로프로세서들의 테스트 전략들을 보여준다.

II. 고장 모델(Fault Model)과 Fault Coverage

1. 고장 모델

테스트되는 시스템의 물리적 고장(Physical Failure)을 수학적, 논리적으로 모델링한 것을 논리적 고장(Logical Fault), 또는 일반적으로 고장(Fault)이라고 한다. 이와 같이 물리적 고장을 논리적 고장으로 모델링한 것을 고장 모델(Fault Model)이라고 하는데 여러 가지 방법에 의해 여러 가지 고장 모델이 존재하게 된다[1].

현재 디지털 로직에서 가장 많이 사용되고 있는 고장 모델로는 Stuck-at 고장 모델, Stuck-open 고장 모델, Bridging 고장 모델, IDDQ 고장 모델, Delay 고장 모델 등이 있으며, 메모리의 경우에는 Stuck-at 고장 모델 이외에 Transition 고장 모델, Coupling 고장 모델, Neighborhood Pattern Sensitive 고장 모델, Address Decoder 고장 모델 등 다양한 메모리 고장 모델을 사용한다. 이와 같은 고장 모델들은 한번의 테스트 시에 하나의 고장만이 존재한다고 가정하는 Single Fault Assumption하에서 많이 사용되고 있다. 회로에 고장을 주입하는 추상 레벨(Abstraction Level)에 따라 전체 고장의 수가 크게 달라질 수 있다. 또한, 사용한 라이브러리의 입출력 포트에 고장을 주입하는 포트 고장(Port Fault)을 고려하였는가 아니면 회로 내에 존재하는 내부 네트에 고장을 주입하는 네트 고장(Net Fault)을 고려하였는가에 따라서도 전체 고장의 수가 크게 달라질 수 있다[1][2][3].

2. 고장 분류(Fault Classification)

고장은 검출 여부에 따라 Detected Fault, Undetected Fault 또는 Undetectable(Untestable) Fault로 나뉘어 질 수 있다. 테스트 벡터에 의해 검출된 고장을 Detected Fault라고 하며 검출 방법에 따라 Solid(Hard) Detected Fault와 Potentially

(Possibly) Detected Fault 등으로 나눌 수 있다. 일반적으로 Solid Detected Fault는 시뮬레이션에 의해 고장 값과 정상 값의 차가 검증된 고장을 일컫고 Potentially Detected Fault는 고장 시뮬레이션(Fault Simulation) 결과 정상 값은 known-value를 가지나 고장 값이 unknown-value를 가지는 경우이며 검출 여부를 결정하기 위해서는 고장 시뮬레이션시에 Potential Count를 지정해 주어야 한다.

테스트 벡터에 의해 검출되지 않은 고장을 Undetected Fault라고 한다. Undetected Fault가 많을수록 Fault Coverage는 낮아지게 되므로 높은 Fault Coverage를 위해서는 Undetected Fault가 가능한 한 적게 발생하도록 테스트 벡터를 생성해야 한다.

어떠한 테스트 벡터에 의해서도 검출될 수 없는 고장을 Undetectable(Untestable) Fault라고 한다. 회로내의 Undetectable Fault를 검출할 수 있는 벡터는 존재하지 않으며, 따라서 고장 시뮬레이션이나 테스트 패턴 생성시에는 이러한 Undetectable Fault를 검출하는 벡터를 찾는다는 것은 무의미하므로, 이들에 대한 분석을 먼저 실행한 후 Undetectable 고장들을 전체 고장 집합에서 제외한 후에 시뮬레이션 등을 실행하는 것이 효과적이다[1][4][5].

3. Fault Coverage

Fault Coverage는 테스트의 효율성과 질을 나타내는 척도로서, 가정한 고장 모델에 의해 회로 내에 존재하는 모든 고장에 대하여 고장의 검출 여부를 계산하여 구해진다. 일반적으로 Fault Coverage가 높은 제품은 낮은 Defect Level을 가진다.

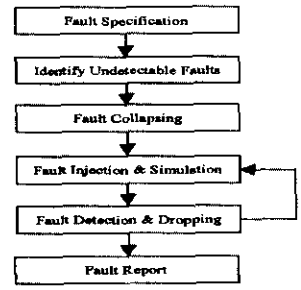
Fault Coverage를 산출하는 공식은 여러 가지로 정의될 수 있다. 고장 시뮬레이터나 ATPG 툴들은 각각 서로 다른 고장 모델, Fault Collapsing 방법, 고장 검출 테이블, Undetectable Fault 추출 능력, Fault Coverage 계산 방식을 가지고 있기 때문에 같은 회로에 대해서 서로 다른 Fault Coverage를 산출해 낸다. Fault Coverage는 Detected Fault의 수를 전체 Fault 수로 나눈 것으로, 경우에 따라서는 전체 Fault 수를 구할 때 Undetectable Fault의 수를 제외하여 계산하는 수도 있다[1][3][4][5].

III. 고장 시뮬레이션과 ATPG

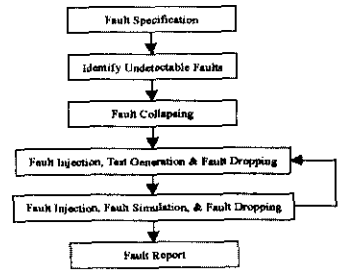
1. 고장 시뮬레이션

테스트 벡터에 대한 회로의 응답을 고장이 존재한다는 가정아래 계산해내는 시뮬레이션을 고장 시뮬레이션이라고 한다. 고장 시뮬레이션은 주어진 테스트 벡터가 얼마나 많은 고장들을 검출해낼 수 있는지를 알아내기 위해 사용하고 Fault Coverage를 산출함으로써 주어진 테스트 벡터의 효율성과 질을 평가할 수 있다.

일반적으로 고장 시뮬레이션은 〈그림 1〉의 (a)와 같은 단계에 의해 진행된다. 먼저 가정한 고장 모델을 이용하여 고장 집합(Fault Universe)의 지정이 이루어진다. 이를 통하여 얻어진 고장집합에 대하여 Undetectable Fault를 선별해내고 나머지에 대하여 Equivalent Fault들을 합치는 Fault Collapsing을 수행한다. Collapsing된 고장들에 대



(a) 고장 시뮬레이션 과정



(b) ATPG 과정

〈그림 1〉 고장 시뮬레이션과 ATPG 과정

하여 Fault Injection을 하고 시뮬레이션을 수행하여 고장의 검출 여부를 판별하고 검출된 고장은 전체 고장 집합에서 제외(Fault Dropping)하고 나머지 고장들에 대하여 Fault Injection과 고장 시뮬레이션을 반복적으로 수행한다. 모든 벡터에 대한 시뮬레이션이 종료되면 결과를 출력하고 전체 과정을 마친다.

많이 사용되고 있는 고장 시뮬레이션 방법으로는 Parallel Fault Simulation, Deductive Fault Simulation, 그리고 Concurrent Fault Simulation 등이 있다. 이들 방법은 회로를 Machine-executable 코드로 번역한 뒤 테스트 벡터에 대하여 평가를 수행하는 컴파일러 구동 방식 시뮬레이션(Compiler-driven Simulation)이나 입력에 이벤트가 발생한 회로내의 게이트에 대해서만 평가를 수행하는 이벤트 구동 방식 시뮬레이션(Event-driven Simulation)으로 구현될 수 있다. 컴파일러 구동 방식은 전체 회로를 모든 입력에 대하여 평가하는데 비해 이벤트 구동 방식은 이벤트 가 발생한 회로 내의 게이트에 대해서만 평가를 수행하므로 평가의 수를 줄일 수 있다[3].

2. ATPG(Automatic Test Pattern Generation)

순차 회로는 회로의 특성상, 어떤 고장을 검출해 낼 수 있는 테스트 패턴이 여러 Time Frame(임의의 패턴이 인가된 후 다음 패턴으로 변하기 전까지의 시간 단위)으로 이루어진 일련의 입력 패턴을 필요로 하게 되며, 이러한 패턴을 사람이 만들기는 매우 어렵다. 일반적으로, 순차 회로의 Testability는 회로내 Flip-flop의 개수와 순차 궤환 루프(Sequential Feedback Loop)와 같은 구조적인 복잡성에 좌우된다. 현재 개발되고 있는 제품들은 대부분 순차 회로이고 다수의 Flip-flop뿐만 아니라, 다양한 Compiled Datapath Cell, RAM, ROM과 같은 메모리 블록과 복잡한 Core 블록을 포함하고 있으므로, 이러한 회로의 Testability는 점점 더 낮아지고, 테스트 벡터를 만들기는 더욱 어려워지고 있다. 단순히 기능 검사 벡터를 추가하여 Fault Coverage를 높이는 것은 현실적으로 한계가 있다. 이에 반하여, ATPG(Automatic Test Pattern Generation)는 특정 고장을 검출해낼 수 있는 테스트 패턴을 정교한 알고리듬을 기초로 한 컴퓨터 프로그램에 의해 체계적으로 생성해 낸다. 〈그림 1〉의 (b)에서는 ATPG의 과정을 보여주고 있다[3].

IV. Design for Testability

1. Testability

어떤 회로의 입력에 임의의 값을 설정하여 회로 내부의 어느 한 노드가 원하는 값을 가지도록 할 수 있는 정도를 Controllability라고 하고, 회로 내부의 어느 한 노드의 값을 출력을 통하여 관찰할 수 있는 정도를 Observability라고 한다.

Testability는 Controllability와 Observability를 합친 상대적인 개념의 수치로 회로내의 여러 노드들의 테스트의 난이도를 나타낸다[1][3]. Testability를 개선하기 위해서는 회로 내에 관찰 가능점(Observable Point)과 제어 가능점(Controllable

〈표 1〉 테스트 방법의 비교[6]

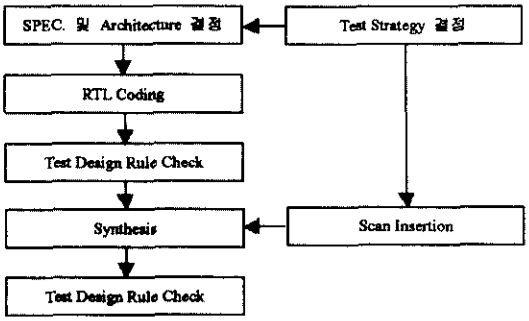
Test Methodology	Time to Market	Area Penalty	Speed Penalty	Coverage(%)	No. of Vectors	Failure Isolation	Speed Grade Support	Burn-in Support	IDDQ Support	Test Time
Functional	Slow	None	None	Low	High	Low	Low	Low	Med.	Slow
BIST	Slow-Med.	Med.	Low-Med.	Low-Med.	Med.	Med.	Low	High	Med.	Slow
Partial Scan	Med.	Med.	Low-Med.	Med.-High	Med.	Med.	Low	Low	Med.	Med.
Full Scan	Fast	Med.-High	Med.	High	Low	High	Med.-High	High	High	Fast

Point)을 증가시켜야 한다. 많은 설계자들이 Ad-hoc 방법을 이용하여 Testability를 개선하였으나 회로의 복잡도가 크게 증가함에 따라 관찰 가능점과 제어 가능점의 선정 문제 및 선정할 수 있는 수의 제한 등과 같은 Ad-hoc 방법의 한계가 문제로 대두되고 있으며 이에 따라 구조적인 DFT 방법의 구현이 크게 늘어나고 있다. 구조적인 DFT 방법에는 Scan Design, Scan-based BIST(Built-In Self-Test), BIST, Cross-check Design 등의 방법들과 보드 테스트(Board Test)를 위한 IEEE 1149.1 Boundary Scan Design 등 여러 가지가 있다[1][3]. 〈표 1〉은 여러 가지 테스트 방법에 따른 테스트 결과의 특성을 비교 정리한 것이다.

2. Scan Design

2.1 Scan Design

Scan Design이란 회로 내에 존재하는 순차 소자(Sequential Element)를 Scan Equivalent Cell로 대치하고 이들을 Shift Chain으로 구성하여 순차 회로를 조합 회로화 함으로써 테스트가 쉽게 되도록 설계하는 방법으로 많이 사용되고 있는 구조적인 DFT방법이다.



〈그림 2〉 Scan Design Flow의 예

〈그림 2〉는 대표적인 Scan Design Flow를 보여주고 있다. 회로 설계의 초기 단계에서 회로의 Specification과 구조를 결정할 때 회로에 대한 테스트 전략(Test Strategy)이 결정되어야 효율적인 DFT 회로의 설계가 가능해진다. 테스트 전략 결정 단계에서 고려될 수 있는 항목으로는 전체적인 테스트 방법을 정한 Test Plan, Scan Design 적용시 Scan에 관계된 사항을 정한 Scan Configuration, 테스트 Clocking Scheme, 테스트 핀의 결정, 메모리 테스트 방법, 그리고 Core 테스트 방법 등이 있으며 이들은 Design Specification과 함께 RTL 코드 생성 및 게이트 레벨의 회로 합성과정에서 반영되어야 한다. Test Design Rule Checking 과정은 RTL 코드 또는 게이트 레벨의 회로에 대한 Testability 검사를 통하여 테스트 전략에서 정한 방법이 회로에 잘 반영되었는가를 확인하는 과정이다. 테스트 전략에 의해 Scan Configuration이 결정되면 Scan Insertion 과정에서 구체적인 방법이 제시되어야 한다. 이러한 항목으로는 Scan Methodology, Scan Style, Scan Chain의 개수, Scan Length, Scan Ordering, Target Fault Coverage, 그리고 각 Scan Chain의 Scan-in/out 포트의 결정 등이 있다. 이와 같은 DFT Specification을 가지고 회로를 설계할 때 꼭 지켜져야 할 것이 Scan Design Rule이다. 만일 Scan Design Rule이 지켜지지 않은 부분이 존재한다면 그 부분의 영향으로 높은 Fault Coverage를 갖는 테스트 벡터는 기대하기 힘들게 된다. Scan Design Rule을 준수하여 설계를 하면 ATPG 프로그램이 테스트 벡터를 생성하기 쉬운 회로가 만들어지게 된다.

2.2 Scan Methodology & Style

Scan Design은 회로내의 순차 소자들에 대한 Scan Design의 적용 정도에 따라 Full Scan Design과 Partial Scan Design으로 나눌 수 있다. Full Scan Design은 회로 내의 모든 순차 소자들을 Scan Equivalent 소자들로 대치한 후 이를 Scan Chain으로 구성하는 방법으로 테스트 벡터 생성 시간이 짧고 높은 Fault Coverage를 가질 수 있으나 Area Overhead가 클 수 있다.

Partial Scan Design은 여러 가지 조건에 의해 일부의 순차 소자들을 Scan Design에서 제외하여 Scan Design을 구현하는 방법으로, 제외하는 방법에 따라 Constraint-driven Partial Scan Design과 Partition-driven Partial Scan Design 등으로 나눌 수 있다. 일반적으로 Partial Scan Design은 Scan에 의한 Overhead가 Full Scan Design에 비해 작지만 Scan Design Rule을 잘 지키지 않을

경우 낮은 Fault Coverage를 가질 확률이 높아서 구현시 주의하여야 한다.

Scan Cell을 정하는 것을 Scan Style을 결정한다고 한다. Scan Style은 사용하는 Scan Cell의 종류에 따라 Multiplexed Flip-flop Scan Design, Clocked Scan Design 그리고 LSSD(Level-Sensitive Scan Design) 등으로 나눌 수 있다[1]. 이들 각각은 서로 다른 장단점을 가지고 있으며 테스트 클럭의 생성 및 Scan 제어 신호의 생성 등이 서로 다르게 된다.

2.3 Scan Design Rule

Scan Design을 이용하여 높은 Fault Coverage를 가지는 테스트 벡터를 쉽게 생성하기 위해서는 Scan Design Rule을 잘 지키면서 Scan Design을 구현하여야 한다.

Scan Design에서 가장 중요한 것은 어떤 Scan Methodology와 Scan Style을 사용하였는가 보다는 Scan Design Rule을 얼마나 잘 지켰는가이다. Scan Design Rule에는 테스트 클럭의 제어 규칙과 다중 클럭 사용 권장, 클럭 신호의 데이터로의 사용금지 규칙, Lockup Latch의 사용 권장, 그리고 Scan Ordering 규칙과 같은 클럭 규칙(Clock Rule)이 있고, Scan Shift시 Asynchronous Set/Reset의 제어와 같은 Set/Reset 신호 규칙이 있으며, 내장 메모리와 같은 Compiled Cell의 Scan Shift시의 disable 규칙, Bus Contention 방지 회로 삽입 규칙, 다중 Scan Chain의 사용 권장, Scan In/Out Port의 공유 권장, Cross-coupled NAND나 NOR 사용 금지 규칙과 같은 기타의 규칙들이 있다. 이들 규칙들은 Scan이 동작하는데 필수적인 규칙과 높은 Fault Coverage를 얻기 위해 필요한 규칙으로 자세히 나눌 수도 있다[4][5].

3. Built-In Self-Test(BIST)

BIST는 회로 내에 테스트 회로를 내장하여 스스로 테스트할 수 있도록 하는 설계 방법으로서, 사용되는 테스트 알고리듬과 구현 방법에 따라 여러 가지 구조가 있다. BIST는 크게 로직 테스트에 적용되는 로직 BIST와 메모리 테스트에 적용되는 메모리 BIST로 나눌 수 있다. 현재 로직 BIST는 테스트 알고리듬이 복잡하고 효율성이 떨어지기 때문에 많이 사용되지 않지만, 규칙적인 구조를 가지고 있어서 테스트 알고리듬이 단순하고 하드웨어로 구현이 용이한 메모리 BIST는 널리 사용되고 있다.

메모리 BIST는 메모리의 종류에 따라 SRAM BIST와 DRAM BIST로 구별될 수 있으며 구현된 테스트 알고리듬과 BIST 구현 방법에 따라 다양

〈표 2〉 BIST 구현의 예

구 분	용 도	내장 메모리		클럭 Cycle	테스트 시간	Gate Count
		개수	종 류			
Chip 1	통신용	13	Single-port, Synchronous SRAM	10MHz	59.6ms	3367
Chip 2	통신용	10	Single-port, Synchronous SRAM	40MHz	0.5ms	1260
Chip 3	Graphic용	8	Single-port, Synchronous SRAM	27MHz	5.8ms	4035
Chip 4	Graphic용	10	Dual-port, Synchronous SRAM(8개) Single-port, Synchronous SRAM(2개)	36MHz	8.4ms	4501
Chip 5	LCD용	4	ROM(2개) Dual-port, Asynchronous SRAM(2개)	4MHz	38.5ms	RTL Code
Chip 6	통신용	1	Single-port, Asynchronous SRAM	60MHz	12.5ms	876
Chip 7	System Bus	4	Single-port, Asynchronous SRAM	10MHz	1.9ms	2314
Chip 8	DSP용	1	4M EDO DRAM	40MHz	240ms	1218
Chip 9	Graphic용	1	16M Synchronous DRAM	100MHz	189ms	1636
Chip 10	Graphic용	2	16M Synchronous DRAM	125MHz	152ms	3260

한 형태를 가질 수 있다. 테스트 알고리듬으로는 March 테스트 알고리듬[2]이 많이 사용되고 있다.

〈표 2〉는 다양한 형태의 메모리 BIST 구현의 예를 보여주고 있다. 대부분의 회로들은 다양한 종류의 여러 개의 내장 메모리를 가지고 있으며 이에 따라 BIST도 다양하게 구현되었다. 〈표 2〉에서 Chip 1, Chip 2, Chip 3은 회로 내에 여러 개의 Single-port, Synchronous SRAM이 있는 경우에 대해 구현 방법에 따른 차이를 보여주고 있고, Chip 4는 두 종류의 SRAM이 여러 개 존재하는 경우이다. Chip 5는 ROM과 RAM이 같이 있는 경우의 예이고 Chip 6과 Chip 7은 Single-port, Asynchronous SRAM이 있는 경우이다. Chip 8, Chip 9, Chip 10은 DRAM이 내장된 경우의 예로 EDO DRAM과 Synchronous DRAM의 예를 보여주고 있다.

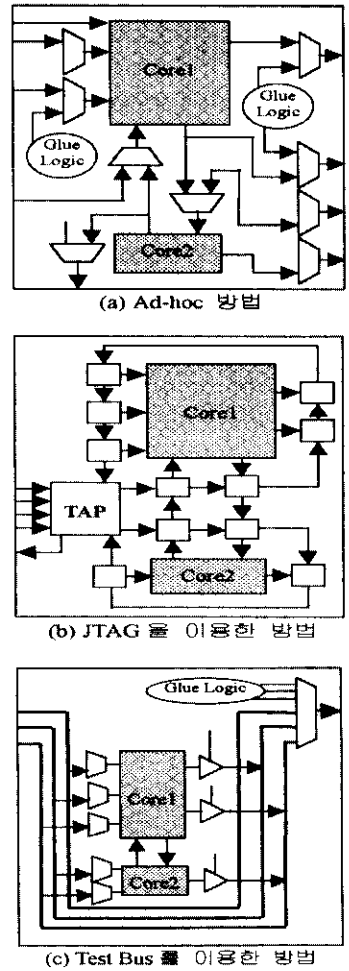
4. Core-based Design

최근 들어 회로의 크기가 점점 커지고 디지털 시스템이 하나의 칩으로 집적화되는 System-on-a-chip 경향에 따라 기존에 설계된 기능 블록(Functional Block)인 Core(또는 Intellectual Property)를 이용하는 Design-reuse 방법이 많이 사용되고 있다.

이러한 Core는 크게 Soft, Firm, 그리고 Hard core로 나눌 수 있다. Soft Core는 합성이 가능한 상위 수준의 기술(High-level Description) 또는 행위 기술(Behavioral Description) 형태로 존재하고, 따라서 Core의 테스트 벡터는 Core 사용자에 의해서 생성이 가능하다. Firm Core도 역시 합성이 가능한 형태로 존재하지만 Floorplanning을 통해 성능과 크기가 구조적으로 최적화된 형태를 가진다. Hard Core는 레이아웃(Layout)으로 완전히 구현된 회로 형태로 존재한다. Firm Core와 Hard Core는 Core 설계자가 제공하는 테스트 벡터가 존재해야 한다[7].

위에서 언급한 어떤 형태의 Core라도 Core 사용자에 의해 시스템으로 집적화 되면 테스트가 어려워진다. 즉, Core 테스트 벡터를 시스템의 Primary Input(PI)들을 통하여 입력하고 시스템의 Primary Output(PO)로 테스트 결과를 전달하는 것이 매우 어렵게 된다. 이러한 Core 테스트의 문제를 해결하기 위하여 제안된 방법으로는 Ad-hoc 접근 방법, Boundary Scan Design을 이용한 접근 방법, 그리고 Test Bus를 이용한 접근 방법 등이 있다. 〈그림 3〉은 이들의 구성 예를 보여주고 있다.

Ad-hoc 접근 방법은 Core의 모든 입출력을 시스템의 입출력에서 단수개의, 또는 복수개의 Multiplexer를 통하여 직접적으로 접근이 가능하게 하는 방법으로, 쉽게 구현이 가능하고 외부에서 직접적으로 Core로의 접근이 가능하므로 테스트가 쉽고 빠르다는 장점이 있으나 긴 Routing Overhead가



〈그림 3〉 시스템 내에서 Core 테스트를 위한 회로 설계의 예

〈표 3〉 Core 테스트 방법[9]

Core	Company	Function	Gate Count	Models	Test
ARM7	Advanced RISC Machines	32-bit RISC Processor	9K	ASIC/ASP	JTAG
Oak	GEC Plessey Semiconductors	DSP	25K	Instruction Accurate, Behavioral, RTL	Not applicable
68030	Motorola	Microprocessor	96K	Gate Level Timing Accurate	Scan
Power PC 401	IBM	Microprocessor	25K	VHDL&Verilog	Full Scan/ATPG
CW4001	LSI Logic	MiniRISC 32-bit MIPS RISC CPU	–	Behavioral, VHDL& Verilog	Mentor Fastscan, Vector : >99% F/C

존재하고 Pin수의 제한이 존재하여 사용이 불가능한 경우가 있다.

Boundary Scan Design을 이용한 접근 방법은 Boundary Scan Cell을 이용하여 Core를 테스트 시에 시스템 로직과 분리하고 TAP(Test Access Port)을 통하여 Core로의 Serial Access를 제공한다. 이 방법은 IEEE 1149.1의 표준안 방법을 통한 구조적인 접근 방법을 제공한다는 장점이 있지만 DFT회로의 Area Overhead와 Timing Delay가 크고 테스트 시간이 매우 길어지게 되는 단점을 가지고 있다.

Test Bus를 이용하는 DFT 방법은 시스템의 Primary Input으로부터 Primary Output까지 연결된 Test Bus를 두고 이를 통하여 Core가 외부와 연결이 되게 한다. 이 방법도 높은 Fault Coverage를 가질 수 있으나 DFT 회로의 Area Overhead와 Timing Delay가 크고 Core들 사이에 존재하는 Interconnection의 테스트가 어려워 진다.

위에서 언급한 방법 이외에도 여러 가지 방법들이 사용될 수 있다. IEEE P1500 표준안은 이러한 Core-based Design을 위하여 Core Test Description Language, Core Test Control Mechanism, 그리고 Core Data Access Mechanism 등에 관한 표준을 제시하기 위하여 만들어졌다. 그러나, Core Test Control Mechanism과 Core Data Access Mechanism은 System-chip Integrator나 제조업체들이 각자의 방법을 선호하기 때문에 직접 규정화 하지 않고, 이를 만족하는 Scalable Architecture와 Core Test Description Language의 규정이 현재 진행되고 있다[8]. 〈표 3〉은 여러 제조업체에서 제공하는 Core들에 관한 정보와 그들이 제공하는 Core 자체의 테스트 방법에 관한 내용을 보여주고 있다.

5. DFT Examples

삼성전자(주)에서는 회로의 Testability를 개선하여 Fault Coverage를 높이려는 노력을 여러 형태로 지속해 왔다. 〈표 4〉는 Ad-hoc 방법을 이용한 결과를 보여주고 〈표 5〉는 Scan Design을 적용한 결과를 보여주고 있다.

〈표 4〉의 회로는 전체 고장 수에서 알 수 있듯이 대부분 작은 회로들로, Ad-hoc 방법을 이용하여 Testability를 개선하고 기능 테스트 벡터(Functional Test Vector)를 이용하여 고장 시뮬레이션을 실시한 결과를 보여주고 있다. 〈표 4〉에서와 같이 Ad-hoc 방법은 작은 회로에 대해서도 많은 테스트 벡터와 시뮬레이션 시간이 요구되며 기능 테스트 벡터를 사용하여서도 높은 Fault Coverage를 얻을 수도 있지만 Chip 2의 경우와 같이 많은 테스트 벡터를 사용하였음에도 불구하고 낮은 Fault Coverage를 얻을 수도 있음을 보여준다.

〈표 5〉는 Scan Design과 ATPG를 적용한 예를 보여 주고 있다. 〈표 5〉의 Gate Count 열에서 보는 것과 같이 회로들은 크기가 매우 크고 매우 복잡한 기능을 가진 고부가 제품들로 디지털 회로

〈표 4〉 Ad-hoc DFT를 적용한 회로의 고장 시뮬레이션 결과

구 분	Vector Size	Total Faults	Fault Coverage	Simulation Time
Chip 1	46M	24476	90.13%	60 Hours
Chip 2	241K	289120	66%	4.6 Days
Chip 3	600K	74792	90.79%	13 Days
Chip 4	74K	61002	90.54%	36 Hours(Parallel Sim)
Chip 5	177K	36687	90.14%	40 Hours
Chip 6	287K	24680	90.75%	28 Hours
Chip 7	1M	99650	93.84%	2.5 Weeks
Chip 8	5K	3119	91.4%	49 Min

〈표 5〉 Scan Design과 ATPG 적용한 회로의 예(*:IP Core 내장)

구 분	Digital/ Mixed	Gate Count	Test Clock 수	Clock Cycle	Scan Chain 수	F/F 수	Scan Ratio	Max. Scan Length	Vector Length	Fault Cov.
Chip 1	Digtal	100K	3	1MHz	27	5.5K	95%	199	89K	97.3%
Chip 2*	Digtal	300K	5	1MHz	18	6.4K	99%	355	170K	94.7%
Chip 3	Mixed	280K	3	20MHz	24	11.3K	99%	518	900K	97%
Chip 4	Mixed	48K	1	1MHz	4	1.3K	99%	338	142K	99.9%
Chip 5	Mixed	583K	1	40MHz	16	1.5K	99%	950	2.6M	96%
Chip 6*	Mixed	96K	6	1MHz	6	1.8K	98%	355	156K	90.7%
Chip 7	Digtal	399K	1	66MHz	14	8.7K	100%	700	807K	96.2%

또는 디지털 회로가 포함된 Mixed-signal 회로이고 특히 Chip 2와 Chip 6은 IP Core를 내장하고 있다. 이들에 대한 Scan Design의 구현을 보면 한 개 또는 다수개의 테스트 클럭을 가지고, 테스트 클럭의 주파수는 다양하며, ATPG에 의해 생성되는 테스트 패턴의 수를 줄이기 위하여 다중 Scan Chain을 사용하여 최대 Scan Length를 줄였고, 이들의 Scan-in/out 포트는 회로의 정상 입출력과 모두 공유하였으며, Full Scan Design 또는 Full Scan에 가까운 Partial Scan Design을 적용하였다. Scan Insertion은 Synopsys사의 Test Compiler와 Mentor Graphics사의 DFTadvisor 등을 이용하였고 ATPG는 Synopsys사의 Sunrise TestGen과 Mentor Graphics사의 FastScan과 FlexTest 등을 이용하였다.

마이크로프로세서는 컴퓨터 등의 시스템에 있어서 가장 중요한 제품중의 하나로 여러 응용 분야를 볼 때 신뢰도와 성능이 매우 중요한 제품이다. 따라서, 여러 제조업체에서 생산되는 여러 마이크로프로세서들은 높은 신뢰도와 초고성능의 두가지 목적을 얻기 위해 여러 가지 테스트 전략을 가지고 있다. 〈표 6〉에서는 여러 마이크로프로세서들의 구조와 테스트 전략들을 간략히 비교하였다. 〈표 6〉에서와 같이 여러 마이크로프로세서들은 각자 고유의 테스트 전략을 가지고 있다. 이들은 Scan Design 또는 Scan-like Technique과 같은 구조적인 방법으로 로직 테스트를 하는 경우가 있고 Full Custom 테스트 전략을 가지고 있는 것들도 있다. 내장되어 있는 메모리의 테스트 또한 BIST를 이용하는 방법, Direct Access 테스트를 이용하는 방법, Functional Test를 이용하는 방법 등 다양한 형태를 가지고 있다.

삼성전자(주)에서 생산하고 있는 Alpha 21164는 400MHz에서 667MHz까지의 클럭 주파수에 동작하며 4-way Instruction Issue가 가능한 Superscalar, Pipelined 64-bit Advanced RISC

〈표 6〉 마이크로프로세서들의 구조와 테스트 전략[6, 10, 11, 12, 13, 14, 15]

Device	Architecture	Test Strategy
PowerPC 603 (IBM, Motorola)	• 1.6M Transistors • 80MHz Operation • 8KB Instruction Cache • 8KB Data Cache	• Full Scan Design • LSSD+Ad-hoc : Scan Test+Functional Test • Array BIST for Caches • Functional Test for Small Size Memories • 3 Test Pins : 2 for Test Clock Pins 1 for LSSD Testing Mode • IDDQ Test 적용 • Support IEEE 1149.1
Super SPARC (TI)	• 3.1M Transistors • 20KB Instruction Cache • 16KB Data Cache • Multiple Clocking Mechanism (Both rising and fallingedge)	• Full Scan Design(4 Scan Chains) −Chain 0 : all falling-edge flip-flop(3100개) −Chain 1 : rising-edge flip-flop(2176개) −Chain 2 : rising-edge flip-flop(996개) −Chain 3 : rising-edge flip-flop(1000개) • BIST : 17-bit LFSR+31-bit MISR(Logic Test에 사용) • Support IEEE 1149.1 • SRAM Test Mode : Direct Access Test
Pentium Pro (Intel)	• 5.5M Transistors • 8KB Instruction Cache • 8KB Data Cache	• Observation-only Scan-like Technique : over 2000 Scan out Nodes • Control Register Bus Access • SSF Fault Coverage=96% • Memory Test : Array Freeze and Dump • BIST for Burn-in testing • IDDQ Test • Support IEEE 1149.1 • DFT-related Die Area Increase : 15%
R10000 (MIPS)	• 5.9M Transistors • 200MHz Operation	• Observability Registers : LFSRs for Internal Node Observation • Memory Test Mode • Direct Clock Controllability for AC Speed analysis
MC 68060 (Motorola)	• 2.5M Transistors • 66MHz Operation • 8KB Instruction Cache • 8KB Data Cache	• Full Scan Design • Memory Test : Ad-hoc Direct Access Method(Burn-in시에는 BIST이용) • Support IEEE 1149.1 • Path Delay Test
AMD-K6 (AMD)	• 8.8M Transistors • 32KB Instruction Cache • 32KB Data Cache	• Full Scan Design • Support IEEE 1149.1 • BIST for Instruction/Data Cache, TLB, ROM • IDDQ 적용 • Path Delay Fault Test 적용
Alpha 21164 (Digital)	• 9.3M Transistors • 8KB Instruction Cache • 8KB Data Cache • 96KB Secondary Cache	• Custom DFT : Structural DFT+Ad-hoc • Controllability : IPRs(Internal Processor Registers) • Observability : OBLs(Observability LFSRs)+IPRs • BIST/BISR on Instruction Cache • Support IEEE 1149.1 • Multi-mode Test Port : 3 Interface Mode(Normal, Manufacturing, Debug) • Testability Circuit : 2.2% of Die Area

(Reduced Instruction Set Computing) 마이크로프로세서로 18.0 SPECint95, 27.0 SPECfp95, 그리고 2.4 BIPS(Billions of Instructions/sec)의 Estimated Performance를 가지는 마이크로프로세서이다. 21164는 8Kbyte의 Instruction Cache와 Data Cache를 각각 가지고 있고, 96Kbyte의 Secondary Cache를 가지고 있다.

Alpha 21164는 하나의 Chip으로 구현되는 회로의 Complexity와 Density가 매우 크고, Performance Target이 매우 높으며, 까다로운 Design Resource와 빠른 개발 계획 등을 가지고 있으면서 높은 Quality와 Fault Coverage, 빠른 Test Pattern 개발 기간, 그리고 높은 양산성을 가져야 하는 제한이 있어서 Custom Design for Testability 방법을 사용하였다[10]. 내장 메모리는 Hardware/Software Assisted Self-Test와 Self-Repair를 이용하여 테스트를 실시하고 Repairable Cache에 대하여 Laser Repair를 실시하여 수율(Yield)을 향상하였다.

Controllability Feature는 외부에서 직접 값을 지정할 수 있는 IPR(Internal Processor Register)들을 회로 내에 두고 테스트 조건과 테스트 모드를 설정하고 테스트 패턴을 인가하는 Ad-hoc 방법을 이용하였다.

Observability Feature는 앞에서 언급한 IPR들과 회로 전체에 흩어져 있는 27개의 OBL(Observability Linear Feedback Shift Register)을 이용하여 550개의 내부 노드를 관찰할 수 있도록 한 Ad-hoc 방법을 이용하였다. OBL들은 MISR(Multiple Input Signature Register)로 동작할 수도 있으며 Scan Register로 동작할 수도 있어서 Manufacturing 테스트에서 At-speed Data Compression이 가능하게 하며 Chip Debugging시에 At-Speed Single-cycle Snap-Shot을 가능하게 해준다.

Chip내의 이와 같은 Customer-usable Testability Feature는 Multi-mode, Multi-purpose 테스트 포트를 통하여 접근이 가능하다. 테스트 포트는 13개의 전용 핀으로 구성되어 있고 Normal, Manufacturing, 그리고 Debugging과 같은 3가지 모드를 지원한다. Die Area의 약 2.2%를 차지하는 이와 같은 Testability Feature와 Manually Developed Functional Test Pattern을 이용하여 만족할만한 Chip의 Quality와 Manufacturability를 구하였다.

V. 결 론

본 글에서는 Design for Testability를 위한 기본 개념들과 Scan Design, 메모리 BIST, 그리고 Core-based Design 방법과 같은 Structural Design for Testability 방법들에 관하여 설명하였으며 Scan Design 및 ATPG의 적용 예, 메모리 BIST의 설계 예, 그리고 고성능 마이크로프로세서들의 테스트 전략 등을 보여주었다.

Scan Design은 크기가 매우 큰 고성능, 고부가의 여러 회로에 대하여 적용되었으며 높은 Fault Coverage와 효율적인 테스트 패턴의 빠른 개발이라는 결과를 보여주었다. BIST는 구조가 규칙적이고 테스트 알고리듬이 간단한 메모리 BIST의 경우를 보여주었으며 회로 내에 여러 종류의 메모리가 여러개 존재하는 경우에도 성공적으로 적용되었음을 보여주었다. ASIC 제품과 달리 대부분 Full Custom으로 설계되는 마이크로프로세서들은 높은 Quality Level과 Performance Goal, Design Constraint 등에 의해 Custom DFT가 적용되는 경우가 많았으며 본 글에서는 Alpha 21164의 예를 통하여 Custom DFT의 구현 예를 자세히 보여주었다.

고성능 VLSI의 Design Constraint를 만족시키면서 Quality Goal을 만족시키기 위해서는 Design for Testability가 반드시 고려되어야 한다. Design for Testability는 회로의 여러 가지 특성에 따라서 Scan Design, BIST, Full Custom 방법 등 다양한 형태로 구현될 수 있으며 설계 초기 단계에서부터 테스트 전략에 따라 적용되어야 효과적으로 구현될 수 있다.

참 고 문 헌

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design, Computer Science Press, 1990.

[2] A.J. van de Goor, Testing of Semiconductor Memories : Theory and Practice, J. Wiley & Sons, 1991.

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

[4] Sunrise TestGen Reference Manual Release 3.0, Synopsys, December, 1997.

[5] FastScan and FlexTest Reference Manual, Mentor Graphics, 1996.

[6] A. L. Crouch, M. Pressly, and J. Circello, "Testability Features of the MC68060 Microprocessor," Proc. IEEE International Test Conference, IEEE Computer Society Press, pp. 60-69, 1994.

[7] I. Ghosh, N. K. Jha, and S. Dey, "A Low Overhead Design for Testability and Test Generation Technique for Core-based Systems," Proc. IEEE International Test Conference, IEEE Computer Society Press, pp. 50-59, 1997.

[8] R. K. Gupta and Y. Zorian, "Introducing Core-based System Design," IEEE Design & Test of Computers, pp. 15-25, 1997.

[9] Integrated System Design Magazine web page : http://www.isdmage.com/EEdesign/HardCoretables.html.

[10] D. Bhavsar and J. Edmondson, "Alpha 21164 Testability Strategy," IEEE Design and Test of Computers Magazine, vol.14, no. 1, pp. 25-33, Jan-Mar 1997.

[11] E. K. Vida-Torku, C. H. Malley, S. Park, and R. Reed, "Design and Test of the PowerPC™ 603 Microprocessor," Proceedings of European Design and Test Conference, pp. 378-384, 1995.

[12] R. Patel and K. Yarlagadda, "Testability Features of the SuperSPARC™ Microprocessor," Proc. IEEE International Test Conference, IEEE Computer Society Press, pp. 773-781, 1993.

[13] A. Carbine and D. Feltham, "Pentium® PRO Processor Design for Test and Debug," Proc. IEEE International Test Conference, IEEE Computer Society Press, pp. 294-303, 1997.

[14] J. Mori, B. Mathew, D. Burns, and Y. Mok, "Testability Features of R10000 Microprocessor," Proc. of 6th Asian Test Symposium, pp. 108-111, 1997.

[15] R. S. Fetherston, I. P. Shaik, and S. C. Ma, "Testability Features of AMD-K6™ Microprocessor," Proc. IEEE International Test Conference, IEEE Computer Society Press, pp. 406-413, 1997.

저 자 소 개

趙 昌 賢

1954년 11월 18일생, 1976년 2월 서울대학교 전자공학과 학사, 1986년 3월 Virginia Tech 전기공학과 석사, 1994년 3월 Virginia Tech 전기공학과 박사, 1976년 3월~1983년 7월 국방과학연구소 연구원, 1994년 5월~현재 삼성전자(주) 반도체 CPU사업팀 연구위원 〈주관심 분야: Design for Testability, BIST, VLSI Failure Analysis〉

金 憲 哲

1968년 10월 26일생, 1991년 2월 서강대학교 전자공학과 학사, 1993년 2월 서강대학교 전자공학과 석사, 1993년 2월~현재 삼성전자(주) 반도체 CPU사업팀 전임연구원, 〈주관심 분야: Design for Testability, ATPG, Microprocessor Testing〉