논문 2006-43SD-10-19

다양한 코어의 병렬 테스트를 지원하는 효과적인 SoC 테스트 구조

(An Efficient SoC Test Architecture for Testing Various Cores in Parallel)

김 현 식*, 김 용 준*, 박 현 태*, 강 성 호**

(Hyun-Sik Kim, Yongjoon Kim, Hyuntae Park, and Sungho Kang)

요 약

본 논문은 SoC 내부의 다양한 코어들을 효율적으로 테스트하기 위한 하드웨어 구조에 초점을 두고 있다. 기존의 한 번에 한 개의 코어만을 순차적으로 테스트하는 방식은 많은 테스트 시간을 요구한다. 이를 보완하고자 본 논문에서는 병렬적으로 여러 코어를 테스트할 수 있는 S-TAM 구조 및 컨트롤러를 제안한다. S-TAM 구조는 테스트 버스 공유 방식을 이용하여 브로드캐스트 방법을 지원하며 이를 기반으로 하여 임의의 코어만을 선택적으로 테스트할 수도 있다. 이뿐 아니라 S-TAM 컨트롤러는 IEEE 1149.1 및 IEEE 1500 등과 같은 서로 상이한 테스트 기반을 통해 구현된 다양한 코어들을 동시에 제어함으로써 효과적인 SoC 테스트를 가능하게 한다.

Abstract

In this paper, we present a new hardware architecture for testing various cores embedded in SoC. The conventional solutions need much testing time since only one core is tested at single test period. To enhance this, S-TAM, a novel test architecture, and its controller which enable parallel testing of various cores are proposed. S-TAM supports bus sharing to broadcast testing and cores to be tested are selected by using it. In addition, S-TAM controller enables the effective SoC test by simultaneous controlling the various test cores which are based on the different test architectures like IEEE 1149.1 and IEEE 1500.

Keywords: SoC, TAM, IEEE 1500, core test

I. 서 론

최근의 IC 설계 분야에서의 가장 큰 화두는 SoC(Sytstem on a Chip)이다. 하나의 완성된 SoC를 위해서는 시스템 설계 기술, 칩 설계 기술, 반도체 공정 기술 등 여러 분야의 첨단 기술이 모두 이용된다.

하지만, SoC 내부에 많은 기능들이 구현되면서 칩 구성은 점점 복잡해질 수밖에 없으며 그에 따른 테스트 비용의 문제가 대두되고 있다. 따라서 이를 해결하려는 계속된 연구가 이루어지고 있고, 이러한 연구의 목적은 테스트 시간 감소를 통한 테스트 비용 절감에 있다고 할 수 있다.

SoC 테스트를 위한 기본 구조는 IEEE 1500[1][2]에 근간을 두고 있다. 이 구조는 IEEE 1149.1[3]과 유사한 기능을 SoC 수준에서 수행할 수 있도록 제안되었다. 이 표준화를 바탕으로 테스트를 하기 위해서는 Core Test Wrapper와 Test Access Mechanism(TAM)의 추가적인 모듈이 SoC 내부에 필요하게 된다. 이 중 Core Test Wrapper는 표준화가 되어 있는 반면에, Test Access Mechanism은 아직 표준화가 되어 있지 않아 사용자가 정의하여 사용하도록 하고 있으며 그에 관한 연구가 진

* 학생회원, ** 평생회원 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
※ 본 논문은 IDEC(IC Design Education Center)의 CAD tool 지원을 받은 것임.
접수일자: 2006년5월12일, 수정완료일: 2006년9월7일

행되고 있다.

테스트 시간이 오래 걸리는 boundary scan의 단점을 SoC 수준에서 극복하기 위해 존재하는 TAM 역시 한계를 가지고 있다. 테스트하는 코어의 TAM이 제한된 수의 테스트 버스를 독점하기 때문이다. 즉, 테스트 패턴 당 하나의 코어만을 테스트하게 된다. 각 코어마다 TAM이 존재하지만 칩과 연결된 테스트 버스(test bus)는 오직 하나의 TAM만이 사용하는 것이다. CAS-Bus[4] 역시 여러 TAM이 테스트 버스를 나누어 사용하지만 해당 TAM이 사용하는 테스트 버스는 다른 TAM에서는 사용할 수가 없고 단지 여러 테스트 버스를 몇 개의 TAM에 분배해줄 뿐이다. TAM을 이용한 분배의 문제는 각 코어에 얼마만큼의 테스트 버스를 할당해야 하는 또 다른 문제이며 시간과 하드웨어 측면 사이에서 trade-off 관계를 분석해야 한다[5][6]. 테스트 버스의 분배를 고려하지 않고 TAM과 wrapper를 동시에 제어하는 추가적인 TAM 컨트롤러(controller)를 통해 병렬적인 테스트가 가능한 TAM-Bus 구조가 제안되었다[7].

이러한 TAM 구조와 별개로 테스트 시간을 개선하고자 한 번에 한 개의 코어가 아닌 여러 개의 코어를 동시에 테스트하는 병렬적인 방법이 연구되었다. 이러한 병렬적인 테스트의 한 종류로 브로드캐스트(broadcast) 방법이 제안되었다[8][9]. 이 방법의 핵심은 여러 코어들을 한 개씩 테스트하는 것이 아니라 그림 1과 같이 테스트 패턴을 공유하여 똑같은 테스트 데이터(test data)를 여러 코어에 동시에 전달하는 방식으로서 테스트 시간과 비용을 줄일 수 있다[10].

그러나 표준화와 성격에 따라 다양한 종류의 코어들이 존재하므로 코어간의 동작 불일치 문제로 인해 테스트 패턴을 공유한다고 해도 여러 종류의 코어를 동시에 테스트할 수 있는 것은 아니다. 왜냐하면 코어의 종류마다 제어하는 방법이 다르기 때문이다. IEEE 1149.1에 기반을 둔 코어(TAPed core)는 TAP 컨트롤러에 의해 동작을 하며 IEEE 1500에 기반을 둔 코어(wrapped core)는 SoC 컨트롤러를 통해서 테스트 명령을 수행한다. 코어 내부에 코어들이 존재하는 hierarchical core의 경우에는 컨트롤러가 존재하지 않는 IEEE 1500기반의 효율적인 hierarchical 코어 모델이 제안되었다[11]. 결국, 세 종류의 다른 코어들을 동시에 테스트하려면 테스트 패턴 공유 이전에 모든 코어가 동일한 동작을 할 수 있도록 적절한 제어를 필요로 한다. 이러한 동작 일치 작업은 병렬 테스트뿐만 아니라 코어 간의 연결선(interconnect) 테스트에서도 매우 중요하다.

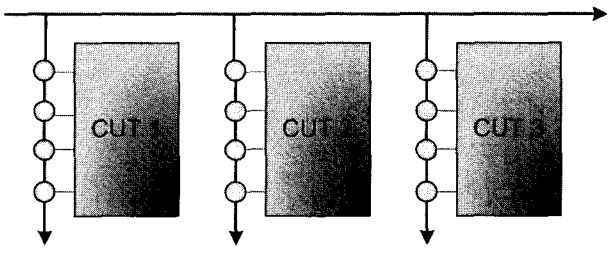


그림 1. 브로드캐스트 방법에 기반을 둔 연결
Fig. 1. A connection between cores based on the broadcast method.

코어의 병렬 테스트를 위해서는 테스트 패턴 공유와 동시에 코어간의 동작 일치 작업이 충족되어야 하며 하드웨어 구조에서 이를 지원할 수 있어야 한다. CAS-Bus는 세 종류의 코어를 지원하며 테스트 버스 분배에 의한 병렬 테스트를 부분적으로 지원하지만 제어와 TAM 구조가 복잡하며 하드웨어 오버헤드가 크다. 반면에 TAM-Bus는 병렬적인 테스트가 가능하지만 TAPed core와 hierarchical core를 지원하지 못하며 TAP 컨트롤러와 별개의 TAM 컨트롤러를 추가적으로 필요로 하는 단점이 있다.

본 논문에서는 이러한 문제점과 한계를 개선하여 여러 유형의 코어들을 동시에 테스트할 수 있는 병렬적인 TAM 구조와 컨트롤러를 제안한다.

Ⅱ. IEEE 1500 구조

IEEE 1500은 SoC 코어의 효율적인 테스트를 위한 표준을 제공한다. IP에 기반을 둔 SoC 설계의 주요 목적은 코어의 재사용(reuse)이다. 재사용에 대한 보장을 위해서는 테스트 역시 재사용이 되어야 한다. 이를 위해 코어 테스트를 위한 표준이 필요하다. 여러 종류의 코어들을 표준 없이 설계한다면 최종적인 SoC 설계에 있어서 테스트에 많은 어려움이 발생하게 된다. 결국 재사용이 가능한(reusable) 코어는 테스트 표준을 이용하여 설계된 코어라고 말할 수 있다. IEEE 1500은 코어 테스트를 위한 표준을 바탕으로 코어 사용자와 코어 제공자 사이의 효율적인 인터페이스(interface)를 제공한다.

1. 전체적인 구조

IEEE 1500의 큰 방향은 고정된 표준화와 유연한 표준화 사이에서 적절한 균형을 제공하는데 있다. 즉,

IEEE 1500이 테스트에 관한 모든 부분을 표준화 시키는 것이 아니라 핵심 부분만을 표준화하고 나머지 부분은 SoC 설계자에게 남겨두는 것이다. IEEE 1500은 테스트를 지원하는 부분에만 초점을 두고 그 외에 시스템적인 문제나 DFT(Design For Testability) 등에 관한 모든 사항을 제시하지는 않는다. 또한 이러한 테스트에 관한 방법 및 구조들은 IP 코어의 기본적인 기능을 제한해서는 안 된다. IEEE 1500에서 제안하는 테스트를 위한 기본 구조는 Core Test Wrapper, Test Access Mechanism, Test Control Mechanism으로 크게 세 가지가 있다. Core Test Wrapper, Test Control Mechanism과 프로토콜은 표준화가 되어 있고 Test Access Mechanism은 사용자가 정의하도록 하고 있다.

2. Core Test Wrapper

wrapper는 SoC 내부의 코어와 TAM 사이의 인터페이스 역할을 한다. wrapper는 크게 WIR(Wrapper Instruction Register)과 WBR(Wrapper Boundary Register)로 이루어져 있다. WIR에서는 wrapper의 테스트 모드(test mode)를 결정하고, WBR은 IEEE 1149.1의 boundary register와 같은 역할을 수행하며 테스트 데이터를 직접 코어와 주고받는 기능을 수행한다. wrapper는 TAM으로부터 테스트 데이터를 전달받거나, 테스트하지 않는 코어에 대해서는 테스트에서 고립(isolation)시키는 역할을 한다. wrapper는 코어의 입력과 출력을 조절할 수 있고 코어의 기본적인 기능에 영향을 미치지는 않는다. wrapper는 SoC 중앙 컨트롤러에 의해서 제어되며 코어 내부에 테스트 데이터를 전달해준다.

3. Test Access Mechanism (TAM)

TAM은 칩 외부로부터 테스트 데이터를 받아서 wrapper에 전달해주는 역할을 한다. 칩의 입출력(I/O) 핀과 wrapper 사이의 가교 역할을 한다고 볼 수 있다. TAM은 테스트 데이터를 직렬(serial)이나 병렬(parallel)적으로 wrapper에 전달할 수 있다. 일반적으로 직렬적으로 연결되는 TAM은 IEEE 1149.1 기반의 TDI를 통해서 들어오며, 병렬적인 테스트 데이터는 각 코어에 연결된 자신의 TAM을 통해서 전달된다. TAM은 IEEE 1500에 의해 표준화가 되어 있지 않고 설계자에게 유연한(flexible) 설계를 제공하지만 TAM 역시 IEEE 1500에 기반을 둔 wrapper와 테스트 방법과의 호환성을 고려하여 설계되어야 한다. TAM이 IEEE 1500과 호환성이 없다면 코어를 테스트하는데 어려움이 발생하고 코어의 재사용에 많은 시간이 소요될 것이다.

4. Test Control Mechanism

SoC에는 여러 코어가 존재하며 보드(board) 상에서도 다른 칩들과 함께 테스트를 해야 하기 때문에 이러한 테스트 동작을 적절히 조절하는 모듈이 필요하다. 컨트롤러는 wrapper와 TAM을 제어해서 테스트를 수행하며 다양한 테스트 모드와 테스트 신호(test signal) 등을 결정한다. 컨트롤러는 표준화된 wrapper의 기능을 지원할 수 있어야 하고 추가적으로 설계자에 의해서 설계된 TAM도 역시 제어할 수 있어야 한다. 또한 컨트롤러의 프로토콜(protocol)과 신호의 처리에 관련된 방법은 IEEE 1500에 의해 표준화되어 있다. 따라서 본 논문도 IEEE 1500의 기반을 따르며 제안한 TAM을 처리할 수 있는 컨트롤러를 제안한다.

III. 제안하는 SoC 테스트 구조

1. 전체적인 구조

제안하는 SoC 테스트 구조는 병렬적인 TAM과 컨트롤러를 포함한다. 병렬적인 TAM은 브로드캐스트 방법을 지원하며 단일 코어에 대한 접근도 가능하다. 또한 병렬적인 테스트를 보장하기 위해서 서로 다른 기반의 코어들(IEEE 1500, IEEE 1149.1, hierarchical)간의 동작 일치를 지원하는 컨트롤러도 제안한다. 그림 2는 제안하는 전체적인 SoC 테스트 구조를 보여준다.

이 테스트 구조는 두 개의 wrapped core와 각각 한 개씩의 TAPed core와 hierarchical core를 포함한 SoC 구조를 보여준다. 각각의 코어에는 필요로 하는 컨트롤 신호(control signal)가 입력된다. wrapped core는 Select WIR, Capture WR, Shift WR, Update WR로 구성되는 WCS(Wrapper Control Signal)가 입력된다. TAPed core는 이미 내부에 TAP 컨트롤러가 존재하므로 wrapped core와 달리 TMS, TCK, TRST로 구성되는 TCS(TAP Control Signal)가 TAP 컨트롤러를 제어한다. 내부에 별도의 컨트롤러가 존재하지 않는 hierarchical core는 wrapped core와 같은 WCS를 이용하며 코어 내부의 코어를 테스트하기 위해서 WIR을 확장하여 추가적인 테스트 모드를 지원한다.

2. Shared TAM-Bus (S-TAM)

본 논문은 Shared TAM-Bus(S-TAM) 구조를 제안

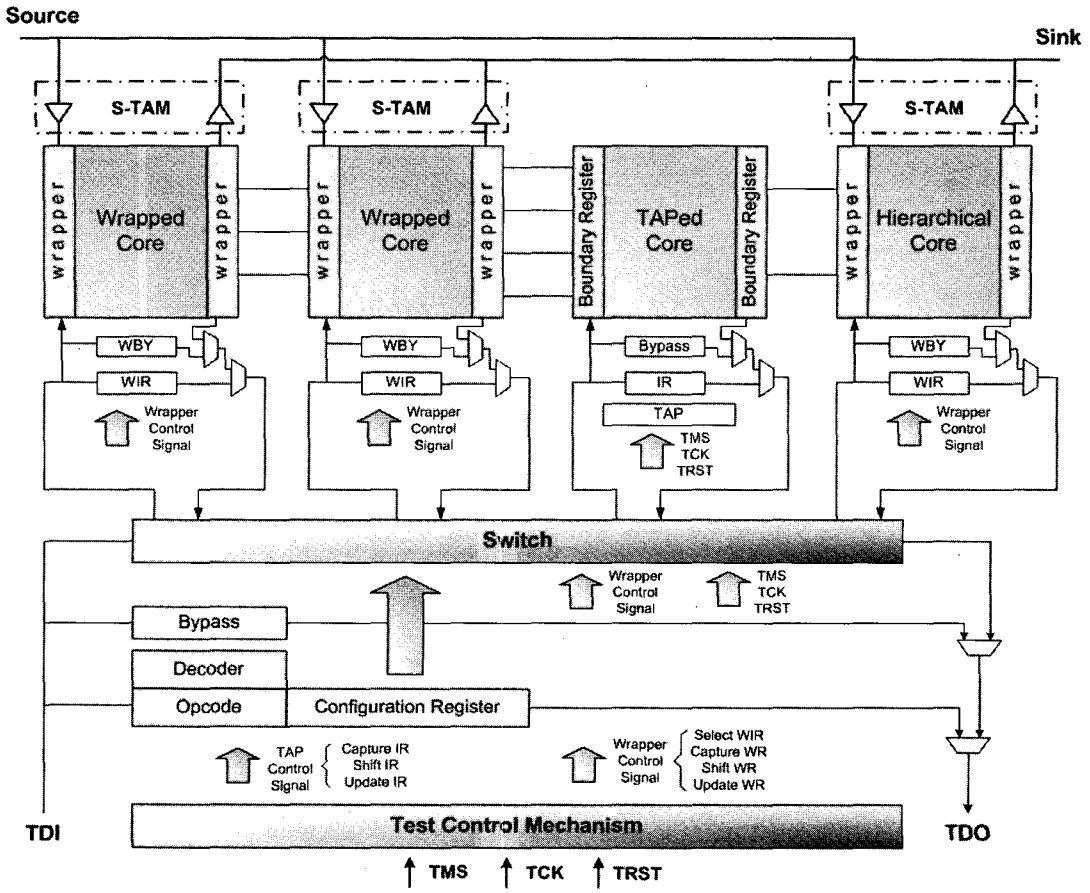


그림 2. SoC 테스트를 위한 전체적인 구조
Fig. 2. Overall architecture for testing SoC.

한다. S-TAM 구조의 가장 중요한 핵심은 S-TAM을 병렬로 연결하여 여러 코어를 동시에 테스트하는 것이다. S-TAM의 병렬적인 연결은 SoC 테스트 스케줄링(scheduling)에서 연구되는 테스트 버스의 분배가 아닌 테스트 버스 공유를 기반으로 한다.

기존의 TAM의 한계는 한 개의 TAM만이 테스트 버스를 점유하게 되어 한 개의 테스트 패턴에 대해 한 개의 코어만 테스트를 하는 것이다. 그러나 S-TAM은 브로드캐스트 방법을 지원하여 병렬적인 코어 테스트가 가능할 뿐만 아니라 기존의 TAM의 방식처럼 한 개의 코어만을 선택하여 테스트하는 방식도 지원한다. 그래서 기존의 방식과 같이 정해진 패턴을 해당되는 코어에만 전달할 수 있게 한다.

우선 S-TAM은 tri-state 게이트로만 구성되어 있다. 소스(source)로부터 들어오는 테스트 버스와 연결된 tri-state 게이트는 테스트 버스에서 코어로의 연결을 제어하고, 싱크(sink)로 나가는 테스트 버스와 연결된 tri-state 게이트는 코어에서 테스트 버스로의 연결을 제어한다. 그림 2의 S-TAM 연결 예는 입력과 출력을 위한 두 개의 테스트 버스와 해당 버스와 연결되는 두 개의 tri-state 게이트로 이루어져 있다. 각각의 코어는 두 개의 tri-state 게이트를 이용하여 테스트 버스에 접근할 수 있게 되며 결과적으로 모든 코어는 테스트 버스를 공유하게 된다. 테스트 버스 수는 입력과 출력을 위해 최소 2개가 필요하며 그 이상이 되어도 tri-state 게이트를 코어와 테스트 버스 사이에 연결만 시켜주는 방식으로 확장이 가능하다.

모든 코어의 tri-state 게이트를 활성화(enable) 시켜주면 모든 코어는 테스트 버스를 동시에 공유할 수 있고 같은 테스트 데이터를 전달받을 수 있게 된다. 이러한 S-TAM의 선택은 명령어 레지스터(instruction register)의 CR(Configuration Register)에서 이루어진다. CR의 값을 통해서 해당되는 S-TAM의 tri-state 게이트의 활성화와 비활성화(disable)를 제어할 수 있다.

S-TAM은 테스트 버스 공유를 통한 병렬적인 테스트를 지원하는 동시에 기존의 TAM의 방식처럼 한 개의 코어만을 선택하여 정해진 패턴을 입력하는 동작을 필요로 한다. 이러한 방식은 S-TAM에서 쉽게 구현될

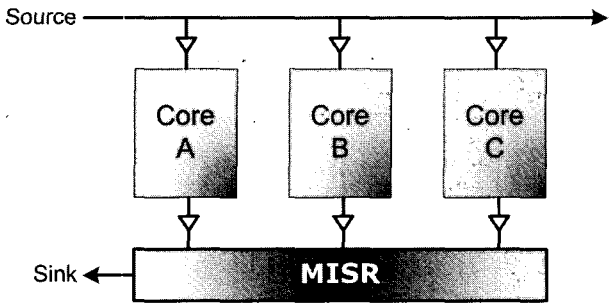


그림 3. MISR을 이용한 브로드캐스트 방법 구성
Fig. 3. A connection of the broadcast method with MISR.

수 있다. CR을 통해서 해당되는 코어와 S-TAM만을 선택하여 tri-state 게이트를 활성화 시켜주고 나머지 S-TAM은 비활성화를 시켜주기만 하면 된다.

병렬적인 테스트를 극대화하기 위해서는 코어에서 테스트 버스로의 출력에 MISR(Multiple Input Shift Register)을 이용하여 여러 코어의 출력을 한 번에 처리하여 칩 외부로 전달하는 것이 필요하다. 앞선 그림 2에서는 MISR을 사용하지 않고 코어의 출력 결과를 바로 테스트 버스로 전달하는 방식을 보여준다. 이 방식을 이용할 경우 테스트 입력에서는 테스트 데이터 공유가 가능하지만 테스트 결과 출력을 하기 위해서는 출력을 위한 테스트 버스를 코어들이 공유하지 않고 순차적으로 이용해야 한다. 이러한 출력방식은 기존의 TAM과 같은 방법이지만 그림 3과 같이 코어의 출력 단에 연결된 tri-state 게이트와 MISR을 연결하여 테스트 버스를 구성하여 칩 외부로 테스트 결과를 전달하는 방식을 사용하면 테스트 버스를 공유한 입력과 같이 동시에 출력된 결과를 얻을 수 있다.

3. 명령어 레지스터 (instruction register)

명령어 레지스터(IR)는 opcode와 CR로 구성된다. opcode는 IEEE 1149.1 표준화를 따르며 반드시 필요한 EXTEST, BYPASS, SAMPLE/PRELOAD 명령을 포함한다[12]. 추가적으로 IEEE 1149.1의 선택적인 명령어도 포함시킬 수 있으며 IEEE 1149.1 호환성을 바탕으로 확장이 가능하다. 또한 코어를 테스트하기 위한 추가적인 명령어들도 포함시킬 수 있다. opcode에 코어 테스트를 위한 wCORETEST 명령어를 추가하여 CR과 별도로 코어로 입력되는 컨트롤 신호들을 통제할 수 있다. 결과적으로 opcode는 SoC 테스트를 위한 추가적인 명령어 확장이 가능하며 또한 IEEE 1149.1과의 호환성도 가지고 있다.

CR의 한 개의 비트는 코어 한 개를 나타낸다. CR의 해당 비트의 값이 1이면 그에 대응되는 코어의 S-TAM과 스위치가 활성화된다. S-TAM이 활성화되면 테스트 버스를 코어와 연결해주고, 스위치는 TDI와 TDO 그리고 컨트롤 신호가 해당 코어의 wrapper와 직접적으로 연결된다. wrapped core와 hierarchical core의 경우에는 WCS가 전달되고, TAPed core의 경우에는 TCS가 전달된다. 즉, CR은 테스트하는 코어를 선택하여 TAM과 스위치를 활성화시키고, 테스트하지 않는 코어에 대해서는 비활성화를 통해 고립을 시킨다.

4. 스위치 (switch)

스위치는 TDI를 통해 들어온 데이터를 다음 코어의 스위치로 바로 통과시키거나 해당 코어에 데이터를 입력하는 역할을 한다. 일반적으로 wrapped core와 hierarchical core의 경우에는 TDI를 이용하여 WIR과 연결선 테스트를 위한 데이터를 입력하고 TAPed core는 자체 TDI를 연결하여 사용한다.

그림 4는 스위치의 기본적인 구조를 보여준다. 스위치는 코어 한 개당 tri-state 게이트 3개와 해당 코어의 컨트롤 신호 입력으로 이루어지며, tri-state 게이트는 CR을 통해서 활성화와 비활성화가 조합되어 TDI를 코어와 연결해주거나 통과시키는 역할을 한다. 이렇게 구성할 경우 테스트하는 코어들을 한 개의 스캔 체인으로 엮는 효과를 얻을 수 있으며 IEEE 1149.1 기반의 boundary scan 구조와 같은 기능을 수행할 수 있다. 또한 테스트를 하지 않는 코어의 경우에는 테스트 데이터가 입력되지 않고 스위치를 통해 바로 다음 코어의 스위치로 연결되기 때문에 불필요하게 스캔 체인의 길이를 늘이지 않는다.

또한 wrapped core와 hierarchical core의 경우에는 WCS가 필요하고 TAPed core는 TCS가 필요하기 때문에 해당되는 코어의 종류에 따라 컨트롤 신호를 전달받

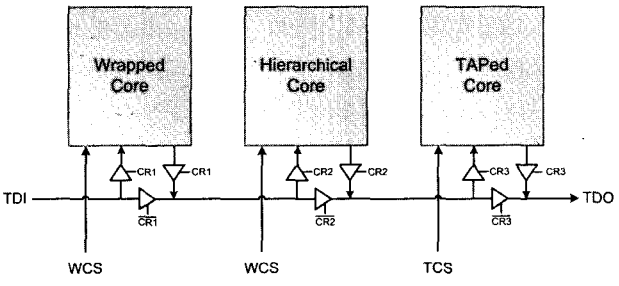


그림 4. 스위치 구조
Fig. 4. The switch architecture.

기만 하면 된다. 스위치에서 컨트롤 신호를 WCS와 TCS만을 선택하여 연결해주기만 하면 되므로 어떠한 종류의 코어에 대해서도 유연하고 확장이 용이하다.

5. SoC 컨트롤러 (controller)

SoC 컨트롤러는 제안한 S-TAM과 wrapped core와 hierarchical core에 직접적인 컨트롤 신호 WCS를 발생시킨다. S-TAM은 IEEE 1500 코어와 함께 동작을 해야 하므로 이미 IEEE 1500에 기반을 두고 설계된 컨트롤러에 추가적인 영향을 미쳐서는 안 된다. 또한 SoC도 보드 수준의 테스트를 필요로 하기 때문에 SoC 컨트롤러는 기존의 IEEE 1149.1 기반의 TAP 컨트롤러와의 호환이 필요하다. 따라서 SoC 컨트롤러는 IEEE 1500과 IEEE 1149.1의 두 표준화를 바탕으로 설계되어야 한다. 그래서 제안한 SoC 컨트롤러는 TAP 컨트롤러의 유한 상태기(finite state machine)를 확장한 형태로 설계하였다.

그림 5는 제안한 SoC 컨트롤러의 상태 천이도(state diagram)를 보여준다. 상태천이도의 왼쪽은 기존의 TAP 컨트롤러를 지원하는 부분이고 오른쪽은 wrapper를 컨트롤하는 부분이다. TAP 컨트롤 부분은 IEEE 1149.1의 TAP 컨트롤러와 동일한 기능을 수행한다. Select-DR 이하 부분은 SoC의 boundary register에 데이터를 입력하는 역할을 수행하고 Select-IR 이하는 명령어 레지스터에 데이터를 입력하는 기능을 수행한다. 오른쪽 wrapper/TAM 부분은 wrapper의 WIR과 WBR에 데이터 입력을 제어한다. wrapper와 TAM을 제어할 수 있는 상태로 이동하려면 Test-Logic-Reset 스테이

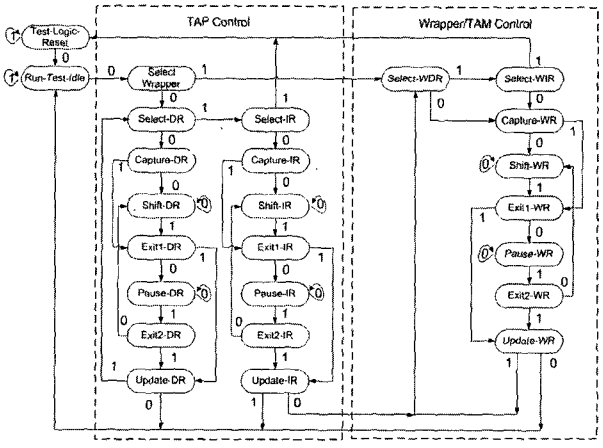


그림 5. SoC 컨트롤러의 유한상태기의 상태 천이도 (state diagram)

Fig. 5. The state diagram of finite state machine of SoC controller.

트(state)에서 단지 0, 0, 1의 3번의 TMS값이면 충분하다. wrapped core의 자가 테스트(self test)를 위한 Run-Test-Idle 스테이트는 가장 왼쪽의 명령어 레지스터의 Run-Test-Idle 스테이트를 이용한다. 명령어 레지스터의 opcode에 wrapper의 자가 테스트를 지원하는 RUNBIST_WR을 추가하여 코어의 WIR에 RUNBIST 명령이 입력되었을 때 Run-Test-Idle 스테이트에서 이를 지원하도록 한다. 명령어 레지스터에 칩 자체의 자가 테스트를 위한 RUNBIST 명령이 입력되었을 경우에는 코어가 아닌 칩을 위한 자가 테스트가 실행된다.

Ⅳ. 코어간의 동작 일치화 과정

1. 코어간의 동작 호환성 문제

코어들간의 병렬적인 테스트를 하려면 테스트 패턴 공유와 더불어 코어간의 동작이 일치해야 한다. 예를 들어, 같은 테스트 데이터를 공유하려면 모든 코어가 동시에 시프트(shift)명령을 수행할 수 있어야 한다. SoC 내부에 wrapped core만이 사용된다면 모든 코어는 같은 컨트롤 신호상에서 동작을 하기 때문에 이러한 문제는 발생하지 않는다. 하지만 TAPed core와 hierarchical core가 사용될 경우에는 wrapped core와 같은 테스트 상태에서 동작할 수 있도록 추가적인 과정이 필요하다. hierarchical core는 wrapped core와 동일한 WCS에 의해 움직이므로 동작 일치 문제가 발생하지 않는다. 하지만 TAPed core는 별도의 컨트롤러를 포함하고 있어 wrapped core와 hierarchical core와 같이 테스트를 할 경우 WCS를 발생시키는 SoC 중앙 컨트롤러와의 동작 일치 작업을 필요로 한다. 이러한 코어간의 동작 일치 작업은 코어에 전달되는 WCS와 TMS를 제어함으로써 같은 테스트 상태를 유지하게 할 수 있다.

2. TAPed core와의 동작 일치화

TAPed core와 wrapped core간의 동작 일치는 제안한 SoC 컨트롤러 상태 천이도의 wrapper/TAM 컨트롤 동작과 TAP 컨트롤러 상태천이도의 동작 실행 순서를 정확히 일치시키는 것이다. 기존 TAP 컨트롤러의 상태 천이도는 그림 6과 같다. 그림 5의 SoC 컨트롤러에서는 서로간의 동작 일치를 위해 Select-Wrapper 스테이트가 추가되었다. 이 스테이트에서는 추가적인 신호가 나오는 것이 아니라 단지 TAPed core의 TAP 컨트롤러와 동작을 일치시켜 주기 위한 임시적인 스테이트이다.

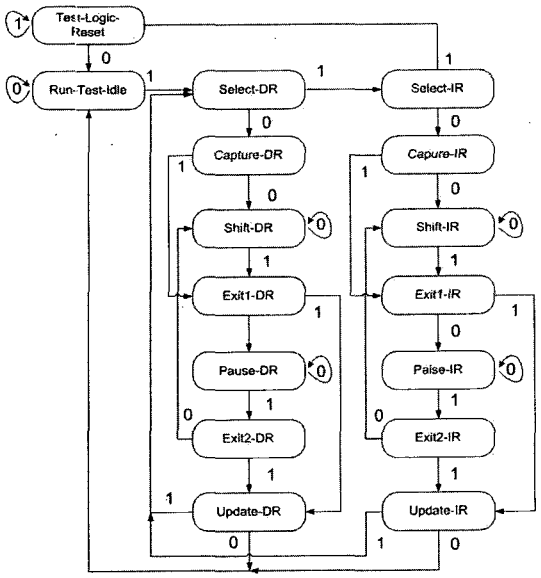


그림 6. TAP 컨트롤러의 상태 천이도
Fig. 6. The state diagram of TAP controller.

이 스테이트에서는 wrapper를 제어하는 스테이트(Select WBR)로 이동하거나 SoC의 명령어 레지스터와 boundary register를 위한 스테이트(Select DR)로 이동하는 역할을 한다. 또한 Run-Test-Idle 스테이트와 Update-IR 스테이트에서 다음 스테이트로 이동할 때 필요한 TMS 값을 0에서 1로, 1에서 0으로 바꾸었다. 결과적으로 스테이트를 한 개 추가하고 기존의 두 스테이트의 TMS값을 바꾼 것만으로 wrapped core와 TAPed core간의 동작 일치를 보장할 수 있게 된다.

코어간의 동작 호환성을 이루는 과정을 살펴보면, SoC 컨트롤러와 TAPed core는 SoC 외부의 TMS에 의해 유한 상태기의 스테이트가 이동하면서 해당 컨트롤 신호를 출력한다. TAPed core가 활성화된 상태이고 SoC 컨트롤러에서 wrapped core에 컨트롤 신호를 입력하는 경우에 TAPed core와 wrapped core의 스테이트 동작을 일치시켜 주어야 한다.

먼저 SoC의 IR과 CR을 이용하여 TAPed core와 wrapped core를 선택한다. 이때 SoC 컨트롤러의 위치는 Update-IR이며 TAPed core의 경우에는 TMS 값을 받아서 TAP 컨트롤러가 동작하기 시작한다. TAPed core의 TAP 컨트롤러는 어느 스테이트에 존재하던지 TMS에 1값을 연속으로 5번을 주면 Test-Logic-Reset 스테이트로 이동한다. TMS에 1값을 연속으로 5번을 주게 되면 TAPed core의 TAP 컨트롤러는 Test-Logic-Reset 스테이트로 이동하게 되고 SoC 컨트롤러는 Run-Test-Idle 상태에서 계속 머물게 된다. 실제적으로 Run-Test-Idle에서 테스트 동작이 이루어지는 경우는 IR에 RUNBIST와 같은 명령어가 존재할 경우이다. 이때 자가 테스트가 실행되는데 해당 명령어가 존재하지 않는다면 아무런 동작도 실행되지 않는다. 이 점을 이용하여 SoC 컨트롤러가 Run-Test-Idle에서 계속 유지하면서 TAP 컨트롤러가 Test-Logic-Reset으로 올 때까지 기다리는 역할을 수행한다. SoC 컨트롤러는 Select-Wrapper 스테이트가 존재하므로 TAP 컨트롤러가 Run-Test-Idle 스테이트를 거치는 만큼의 효과를 얻을 수 있다. 그 이후에는 Select-DR과 Select WDR의 동작 일치가 이루어져 코어간의 동작 호환성을 이루어 연결선 등의 테스트를 효과적으로 진행할 수 있다.

그러나 동작 일치 이후에 테스트 과정에서 Update 스테이트로 가면 문제가 발생할 수 있다. 이 때 TMS값이 1이 되면 동작 일치가 계속해서 이루어지지만, TMS 값에 0이 입력되면 TAP컨트롤러와 SoC 컨트롤러 모두 Run-Test-Idle로 가게 된다. SoC 컨트롤러는 중간에 Select-Wrapper 스테이트가 존재하므로 동작 호환성이 이루어지지 않게 된다. 이 문제는 TMS값에 1을 연속적으로 세 번 입력함으로써 해결될 수 있다. TMS에 세 번의 1을 입력하면 TAP 컨트롤러는 다시 Test-Logic-Reset으로 가게 되고 SoC 컨트롤러는 계속해서 Run-Test-Idle 스테이트에 존재하므로 처음과 같이 동작 일치를 위한 준비가 이루어진다. 결과적으로 어느 스테이트에 있던지 SoC 컨트롤러를 Run-Test- Idle 스테이트로 이동력하여 TAP 컨트롤러를 Test-Logic-Reset으로 이동시키면 동작 호환성이 이루어진다. 이러한 추가적인 TMS값은 최대 5 클록을 넘지 않는다.

3. hierarchical core와의 동작 일치화

hierarchical core 내부에 별도의 컨트롤러가 존재하지 않기 때문에 wrapped core와 동일하게 WCS를 통해 제어가 되므로 문제가 되지 않는다. 다만 그 이전에 hierarchical core의 WIR을 통해서 테스트 하려는 hierarchical 내부 코어를 미리 선택을 해야 한다. hierarchical 내부 코어가 정해지면 WCS를 통해 제어받는 상태로 이전되므로 wrapped core와 동일한 동작을 수행할 수 있다.

V. 결과 분석

본 장에서는 동작 일치 과정까지의 시간과 컨트롤러의 크기를 분석하고 다른 SoC 테스트 구조와의 비교를

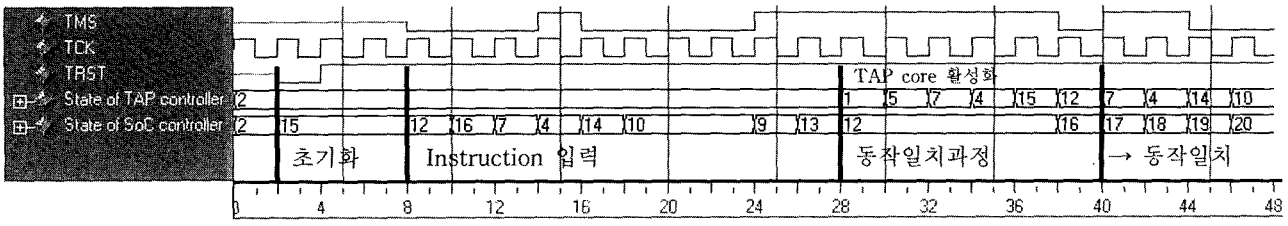


그림 7. TAPed core와 wrapped core간의 동작일치를 위한 스테이트 과정
Fig. 7. The process of the state between a TAPed core and a wrapped core in synchronization.

통해서 제안하는 구조의 성능을 보여준다.

SoC 테스트 성능을 보여주는 가장 중요한 핵심은 테스트 시간이다. 본 논문의 SoC 테스트 구조는 병렬 테스트를 지원하므로 테스트 패턴이 입력되는 클록수는 기존의 직렬 방식보다 줄어들게 된다. 하지만, 병렬 테스트 과정에 도달하기까지의 과정에 따라서 오히려 더 비효율적이 될 수 있다. 이에 본 논문에서는 wrapped core/hierarchical core와 TAPed core상의 동작 일치 과정을 verilog HDL를 이용하여 실험하였다.

본 논문에서 제안한 구조는 추가적인 하드웨어 없이 TAP 컨트롤러와 SoC 컨트롤러 사이의 TMS만으로 동작 일치 과정을 이루므로 각 컨트롤러를 설계하여 해당하는 스테이트를 조사함으로써 동작 일치 과정을 확인할 수 있다.

그림 7은 두 컨트롤러의 동작 일치 과정을 보여준다. 각 스테이트에 해당되는 10진수 값은 표 1과 같다. 먼저 TRST 신호가 들어오면 SoC 컨트롤러는 Test-Logic-Reset 스테이트로 초기화된다. IR에 명령어를 입력하고 Update-IR 스테이트에서 update 신호를 입력하는 순간 TAP 컨트롤러가 TMS에 의해 작동하기 시작한다. 이때 5번의 연속적인 TMS 1값을 주면(28~38 구간) TAP 컨트롤러는 Test-Logic-Reset 스테이트로 이동하고 SoC 컨트롤러는 계속해서 Run-Test-Idle 스테이트에 머물고 있는 것을 알 수 있다. 0, 1의 TMS 값이 들어오면 두 컨트롤러 모두 select-DR/Select-WBR 스테이트로 이동하게 되어 결국 같은 동작을 수행하게 된다. 따라서 총 7클록만으로 TAPed core와 wrapped core간의 동작 일치가 이루어지고 이때부터 두 코어간의 병렬테스트가 가능하게 된다.

다음으로 SoC 컨트롤러를 합성 툴을 이용하여 기존의 TAP 컨트롤러와의 크기를 비교하였다. 합성에는 Synopsys사의 Design Analyzer를 이용하였다. 또, 컨트롤러간의 공정한 크기 비교를 위하여 TAP 컨트롤러는 Mentor사의 BSDA(Boundary Scan Design Architect)

표 1. 각 스테이트의 10진수
Table 1. States to the decimal.

컨트롤러	스테이트	10진수
TAP Controller	Exit2-DR	0
	Exit1-DR	1
	Shift-DR	2
	Pause-DR	3
	Select-IR	4
	Update-DR	5
	Capture-DR	6
	Select-DR-scan	7
	Exit2-IR	8
	Exit1-IR	9
	Shift-IR	10
	Pause-IR	11
	Run-Test-Idle	12
	Update-IR	13
	Capture-IR	14
	Test-Logic-Reset	15
SoC Controller	Select-Wrapper	16
	Select-WBR	17
	Select-WIR	18
	Capture-WR	19
	Shift-WR	20
	Exit1-WR	21
	Pause-WR	22
	Exit2-WR	23
	Update-WR	24

표 2. 합성 결과 비교
Table 2. Comparison of the synthesis result.

	TAP 컨트롤러	SoC 컨트롤러
입력 포트수	3	3
출력 포트수	11	16
스테이트수	16	25
면적(Area)	2033	3513

표 3. 성능 비교
Table 3. Performance evaluation.

	CAS-Bus[4]	TAM-Bus[7]	S-TAM
병렬 테스트 지원 방식	버스분배방식	버스 공유	버스 공유
TAPed core 지원	지원	×	지원
Hierarchical core 지원	core 수정 필요	×	지원
테스트 버스 수	코어수	2	2
CR 크기	코어수	코어수×2 + 2	코어수
컨트롤러 스테이트 수	32	39	25
TAM 구성	3개 tri-state 1개 MUX 명령어레지스터 디코더	2개 MUX	2개 tri-state

를 사용하여 발생하였다.

표 2는 두 컨트롤러의 면적을 비교한 것으로 면적 1은 NAND 게이트 수를 나타낸다. SoC 컨트롤러의 면적이 TAP 컨트롤러에 비해 72.8%가 더 많은 것을 알 수 있다. 기존의 TAP 컨트롤러에 72.8%의 증가만으로 SoC 컨트롤러는 구성된다. 이는 TAP 컨트롤러에 wrapped core와 hierarchical core를 제어하는 부분이 추가되었기 때문이다. SoC 컨트롤러는 칩 자체를 위한 컨트롤러를 공유하기 때문에 추가되는 기능에 비해서 SoC 컨트롤러의 면적은 상대적으로 적은 부분만이 증가하였다고 볼 수 있다. 왜냐하면 TAP 컨트롤러는 SoC 칩 자체만을 위한 것으로 wrapped 코어를 테스트하기 위해서는 별도의 추가적인 컨트롤러가 필요하기 때문이다. 그렇기 때문에 오히려 칩과 내부 코어를 위한 컨트롤러를 공유함으로서 더 효과적인 제어가 가능하다.

표 3은 제안한 구조와 기존의 TAM에 관한 구조와 제어 방식을 비교적 정확히 기술하고 있는 논문[4][7]과의 비교를 보여준다. CAS-Bus의 경우에는 테스트 버스를 분배한 부분적인 병렬 테스트를 지원하는 형태이므로 테스트 버스 수가 제한되어 있다면 병렬 테스트가 불가능하다. 따라서 TAM-Bus와 S-TAM과 달리 테스트 버스가 코어 수만큼 필요하게 된다.

TAM-Bus의 경우에는 TAPed core와 hierarchical core를 지원하지 못하며 오직 wrapped core만을 테스트할 수 있다. CAS-Bus는 hierarchical core와 TAPed core를 지원한다. 하지만 hierarchical core의 내부 코어에 모두 CAS가 달려있어야 하며 TAPed core와의 동작 일치를 위해 총 32개의 스테이트를 사용하고 카운터와 3개의 신호가 추가적으로 필요하다. 또한 TAPed core의 자가 테스트를 위한 RUNBIST 명령을 지원하지 못하는 단점이 있다.

하드웨어 측면에서도 S-TAM이 가장 적은 하드웨어를 사용한다. S-TAM은 단지 2개의 tri-state 게이트로 구성되어 있고, 컨트롤러의 스테이트 수는 25개로 가장 적게 사용한다. 반면에, TAM-Bus 경우에는 TAM에 MUX만을 사용하여 적은 하드웨어를 요구하지만 CR의 크기가 S-TAM과 CAS-BUS보다 2배가 더 많이 필요하다. 또한 컨트롤러에서는 추가적인 TAM 컨트롤러에 23개의 스테이트를 사용하고 TAP 컨트롤러에 16개의 스테이트를 사용해 총 39개의 스테이트를 필요로 한다. CAS-BUS는 TAM 내부에 명령어 레지스터와 디코더를 사용하기 때문에 제어 과정이 복잡하고 테스트 버스 등 많은 하드웨어 오버헤드를 요구하며 컨트롤러 역시 32개의 스테이트를 요구한다.

이상 결과에서 보듯이 제안한 S-TAM은 기존의 구조와 방법보다 적은 하드웨어를 사용하면서 제어가 간단하고 병렬적인 테스트 지원으로 가장 빠른 테스트 시간을 보여준다. 또한 다양한 종류의 코어들간의 동작 일치가 가능함으로써 병렬 테스트를 보다 효율적으로 지원할 수 있다.

VI. 결 론

본 논문에서 제안하는 SoC 테스트 구조는 코어 테스트를 보다 효율적이고 빠르게 진행할 수 있다. 새롭게 제안한 S-TAM 구조는 테스트 버스 공유를 통해 여러 코어들을 병렬적으로 테스트가 가능하며 브로드캐스트 방법을 지원한다. 또한 임의의 코어들만을 선택적으로 테스트가 가능하며 단일 코어에 대해서도 기존의 TAM과 같은 기능을 제공한다. S-TAM 구조는 tri-state 게이트와 최소한 2개의 테스트 버스만을 이용하여 적은 하드웨어 오버헤드를 가진다. 또한 S-TAM은 코어수와 상관없이 기존의 구조에 S-TAM만을 쉽게 추가가 가능함으로써 높은 확장성 및 유연성을 가진다. 제안한 컨트롤러는 wrapper를 컨트롤하는 것만으로 S-TAM을 쉽게 제어할 수 있고 기존 구조에 비해 적은 스테이트만을 사용한다. 병렬적인 테스트를 효율적으로 지원하기 위해서 여러 기반(IEEE 1500,

IEEE 1149.1, hierarchical)의 코어들간 동작을 일치시킬 수가 있으며 별도의 하드웨어가 필요하지 않고 기존의 컨트롤러를 확장하여 최대 7클록만을 추가함으로써 모든 종류의 동작 일치를 지원할 수 있다.

참 고 문 헌

[1] P1500 Scalable Architecture Task Force Members, "Preliminary Outline of the P1500 Scalable Architecture for Testing Embedded Cores", in Proceeding of 17th VLSI Test Symposium, pp. 483-488, 1999.

[2] IEEE P1500 General Working Group, "IEEE P1500 Standard For Embedded Core Test", see http://grouper.ieee.org/groups/1500/

[3] Kenneth P. Parker, "The Boundary-Scan Handbook", third edition, Kluwer Academic Publishers, 2003, pp.11-78.

[4] Mounir Benabdenbi, Walid Maroufi and Meryem Marzouki, "CAS-BUS: a Test · Access Mechanism and a Toolbox Environment for Core-Based System Chip Testing", Journal of Electronic Testing: Theory and Applications, Vol. 18, pp. 453-473, 2002.

[5] Kuo-Liang Cheng et al., "An SOC test integration platform and its industrial realization", in Proceeding International Test Conference, pp. 1213-1222, 2004.

[6] V. Iyengar and A. Chandra, "Unified SOC test approach based on test data compression and TAM design", in Proceeding of Computer Digital Technology, Vol. 152, pp. 82-88, 2005.

[7] Wang Yong-sheng, XIAO Li-yi, Wang Jin-xiang and Ye Yi-zheng, "Test Control of TAM-Bus: A Solution for Testing SoC", in Proceeding 5th International Conference on ASIC, Vol.2, pp.1124-1127, 2003.

[8] Keun-Jong Lee, Jih-Jeen Chen and Cheng-Hua Huang, "Using a Single Input to Support Multiple Scan Chanis", in Proceeding of International Conference Computer-Aided Design, pp. 74-78, 1998.

[9] Ilker Hamzoglu and Janak H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores", in Proceeding of International Symposium on Fault-Tolerant Computing, pp. 260-267, 1999.

[10] J. H. Jiang et al., "Embedded Core Test Generation Using Broadcast Test Archiecture and Netlist Scrambling", IEEE Transactions on Reliability, Vol. 52, pp. 95-103, 2003.

[11] Anuja Sehgal et al., "IEEE P1500-Compliant Test Wrapper Design for hierarchical Cores", in Proceeding of International Test Conference, pp. 1203-1212, 2004.

[12] Michael. Bushnell and Vishwani D. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI Circuits, Kluwer Academic Publishers, 2001, pp. 549-574.

[13] Erik Jan Marinissen et al., "On IEEE P1500's Standard for Embedded Core Test", Journal of Electronic Testing: Theory and Applications, Vol. 18, pp.365-383, 2002.

[14] 송동섭, 김용준, 신용승, 강성호, "An Efficient Test Control Methodology for SoC based on IEEE P1149.1", 제10회 한국반도체학술대회지, 743-744쪽, 2003.

[15] Kuen-Jong Lee, Cheng-I Huang, "A Hierarchical Test Control Architecture for Core Based Design", 9th Asian Test Symposium, pp. 248-253, 2000.

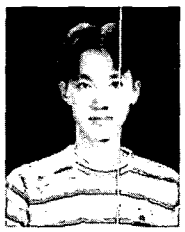
[16] Lee Whetsel, "Addressable Test Ports An Approach to Testing Embedded Cores", in Proceeding of International Test Conference, pp. 1055-1064, 1999.

[17] Feng Jianhua et al., "An improved Test Access Mechanism Structure and Optimization technique in System-on-Chip", in Proceeding of the ASP-DAC, Vol. 2, pp. 23-24, 2005.

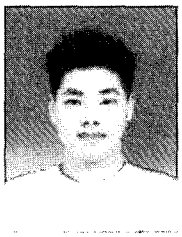
저 자 소 개

23x30

김 현 식(학생회원)
2006년 연세대학교 전기전자 공학과 학사 졸업.
2006년 현재 연세대학교 전기전자 공학과 석사 과정.
<주관심분야 : SoC 설계 및 응용, SoC 테스트>



박 현 태(학생회원)
2004년 연세대학교 전기전자 공학과 학사 졸업.
2006년 연세대학교 전기전자 공학과 석사 졸업.
2006년 현재 연세대학교 전기전자 공학과 박사 과정.
<주관심분야 : 고속네트워크 관련 SoC 설계 및 네트워크 프로세서 설계>



김 용 준(학생회원)
2002년 연세대학교 전기공학과 학사 졸업.
2004년 연세대학교 전기전자 공학과 석사 졸업.
2005년 현재 연세대학교 전기전자 공학과 박사 과정.
<주관심분야 : CAD, Testing>

23x30

강 성 호(평생회원)
1986년 서울대학교 제어계측 공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업
1992년 The University of Texas, Austin 전기 및 컴퓨터 공학과 박사 졸업.
1992년 미국 Schlumberger Inc. 연구원
1994년 Motorola Inc. 선임연구원
2006년 현재 연세대학교 전기전자공학과 교수
<주관심분야 : SoC 설계 및 응용, DFT, SoC 테스트>