논문 2006-43SD-7-6

임베디드 프로세서와 재구성 가능한 구조를 이용한 SoC 테스트와 검증의 통합

(Integration of SoC Test and Verification Using Embedded Processor and Reconfigurable Architecture)

김 남 섭*, 조 원 경*

(Namsub Kim and Wonkyung Cho)

요 약

본 논문에서는 SoC를 검증 및 테스트하기 위한 새로운 개념의 칩을 제안하고 이를 SwToC(System with Test on a Chip)라 명명한다. SwToC는 SoC의 임베디드 프로세서에 재구성 가능한 로직을 추가하여 칩의 물리적인 결함을 테스트할 수 있을 뿐만 아니라 기존의 기법으로는 수행이 어려웠던 테스트 단계에서의 디자인 검증이 가능하도록 한 칩을 말한다. 제안한 개념의 칩은 고속 검증이 가능하며 테스트를 위해 많은 비용이 소모되는 ATE가 불필요한 장점을 갖고 있다. 제안한 칩의 디자인 검증 및 테스트 기능을 평가하기 위하여 임베디드 프로세서가 내장된 상용 FPGA를 이용하여 SwToC를 구현하였으며, 구현 결과 제안한 칩의 실현 가능성을 확인하였고 적은 비용의 단말기를 통한 테스트가 가능함은 물론 기존의 검증기법에 비해 고속 검증이 가능함을 확인하였다.

Abstract

In this paper, a novel concept based on embedded processor and reconfigurable logic is proposed for efficient manufacturing test and design verification. Unlike traditional gap between design verification and manufacturing test, proposed concept is to combine both design verification and manufacturing test. The semiconductor chip which is using the proposed concept is named "SwToC" and SwToC stands for System with Test On a Chip. SwToC has two main features. First, it has functional verification function on a chip and this function could be made by using embedded processor, reconfigurable logic and memory. Second, it has internal ATE on a chip and this feature also could be made by the same architecture. To evaluate the proposed SwToC, we have implemented SwToC using commercial FPGA device with embedded processor. Experimental results showed that the proposed chip is possible for real application and could have faster verification time than traditional simulation method. Moreover, test could be done using low cost ATE.

Keywords : SoC, Test, Verification, HW/SW co-simulation, Reconfigurable Architecture

I. 서 론

제조된 반도체 제품이 최종 소비자에게 넘어가 원활한 동작을 보장하기 위해서는 제조된 칩에 대한 테스트 과정은 필요불가결한 요소이다. 이러한 반도체 제품에 대한 테스트는 설계단계에서의 디자인 오류(Design Error)를 검증하는 부분에서부터 설계검증이 끝난 후 제조공정에서 발생되는 오류에 대한 테스트 과정을 모두 포함하는 개념을 내포하고 있으나 일반적으로 설계단계에서의 오류를 검사하는 과정을 디자인 검증(Design Verification)이라 부르며 설계가 끝난 후 웨이퍼(wafer)단계에서부터 최종 생산된 된 칩을 검사하는 과정을 테스트(test)라 한다.

반도체 제조기술이 발달됨에 따라 SoC(System on a Chip)와 같이 복잡한 회로들이 하나의 칩 안에 집적되게 되었고, 이러한 칩의 복잡도(complexity)는 설계단계

* 정회원, 경희대학교 전자정보학부
(School of Electronics and Information, Kyung Hee University)
접수일자: 2006년5월26일, 수정완료일: 2006년7월3일

에서 많은 검증시간을 요구하고 테스트 단계에서는 많은 비용(cost)을 소비한다[1]. 또한 기존의 정형 검증(formal verification) 기법이나 IDDQ, Logic BIST(Built in Self Test)와 같은 테스트 기법으로는 검증 및 테스트가 어렵게 되었다. 따라서 빠른 시장대응(time-to-market)과 시장 경쟁력을 지닌 칩을 생산하기 위해서는 검증시간과 테스트에 소요되는 비용을 줄여야 하며, 이러한 목적을 달성하기 위하여 검증분야에서는 HW/SW(Hardware/Software) co-simulation[2] 기법이 연구되어 왔고 테스트분야에서는 Modular Testing[3] 기법이 연구되어 왔다.

HW/SW co-simulation은 90년대 초반에 처음으로 그 용어가 대두되었고 하드웨어와 소프트웨어가 공존하는 SoC와 같은 칩을 검증하기 위한 기법이며, 이에 대한 대표적인 연구들을 살펴보면 다음과 같다.

Becker, Singh, Tell[4]은 Verilog-XL, C++, Unix 소켓(socket)을 이용한 NIU(Network Interface Unit)의 co-simulation 기법을 제안하였고, Thomas, Adams, Schmit[5]는 Verilog-XL 시뮬레이터의 PLI (Programming Language Interface)와 UNIX 소켓을 사용하여 프로세서 없이 검증할 수 있는 기법을 제안하였다. Borgatti, Rambaldi, Gori, Guerrieri[6]는 하드웨어 에뮬레이션과 소프트웨어 시뮬레이션을 통합한 가상의 에뮬레이션 시스템을 개발하였다.

Modular Testing은 Divide and Conquer알고리즘을 칩의 테스트에 적용한 기법으로, 칩을 여러 개의 모듈로 나눈 후 각각의 모듈별로 따로 테스트를 진행하는 방식이며 이를 위한 다음과 같은 연구가 수행되어져 왔다.

Jas[7]와 Balakrishnan[8]는 ATE(Automatic Test Equipment)비용을 낮추기 위하여 임베디드 프로세서(embedded processor)를 사용한 테스트 기법을 제안하였고, Gang Zeng[9]은 HybridASIC을 이용하여 테스트의 효율성을 높이는 방법을 제안하였다 . 그 외에 He[10]는 HybridBIST를 이용하여 BIST(Built in Self Test)의 낮은 고장 검출율(fault coverage)을 내장된 테스터(tester)로부터 결정론적(deterministic) 테스트 패턴을 인가하여 테스트 효율성을 높인 테스트 기법을 제안하였고, Koranne[11] 와 Lei[12]는 coarse-grained형태의 재구성 가능한 로직(reconfigurable logic)을 이용하여 테스트를 수행하는 기법을 제안하였다.

이와 같은 연구들의 근본적인 목적은 신뢰성 있는 칩의 생산이다[13]. 그러나 신뢰성을 위해 많은 비용과 시간이 투자된다면 제품으로서의 가치가 없기 때문에 검증 분야에서는 빠른 검증을, 테스트 분야에서는 저비용(low cost) 테스트 기법을 연구하여 왔다. 따라서 같은 목적임에도 불구하고 서로 다른 방법을 사용하고 있기 때문에 검증을 수행할 때는 칩의 물리적인 검사를 할 수 없었고, 테스트를 진행하는 과정에서는 전에 수행되었던 검증 자체의 오류여부를 판단할 수가 없었다. 그러나 본 논문에서는 디자인 검증과 테스트가 근본적으로 검증 및 테스트를 위한 패턴(pattern)의 입력과 출력을 비교한다는 점과 SoC 내부에는 하나이상의 프로세서가 내장되어 있다는 점에 착안하여 빠른 속도의 검증과 저비용의 테스트를 통합하여 수행할 수 있는 새로운 형태의 칩을 제안하고 이러한 목적을 충족시키는 구조의 칩을 SwToC(System with Test on a Chip)이라 명명한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 제안하는 신개념의 칩에 대하여 설명하고 Ⅲ장에서는 기존의 FPGA(Field Programmable Gate Array)를 이용한 SwToC의 구현에 대하여 설명한다. Ⅳ장에서는 SwToC의 구현 결과 및 성능평가를 나타내고 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. System with Test on a Chip(SwToC)

1. SwToC의 개념

전통적인 반도체 칩의 이상 유무를 평가하는 과정은 그림 1과 같이 검증과 테스트가 분리되어 수행된다.

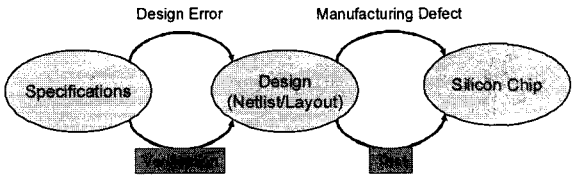

그림 1. 디자인 검증과 테스트
Fig. 1. Design Verification and Test.

이와 같은 방법은 다음과 같은 근본적인 문제점을 내포하고 있다.

(1) 디자인 검증의 부정확성
(2) 테스트를 위해 추가된 회로로 인한 기능 오류
(3) 테스트 단계에서의 기능검증 부재

첫째, 디자인 검증은 실제 회로에 대한 기능적 이상유무를 판단하는 것이 아니라 네트리스트(netlist)나 레

이아웃(layout)과 같은 형태의 회로를 시뮬레이션을 통해 모의실험을 하는 과정이다. 따라서 시뮬레이션 툴(tool)의 오류, 합성(synthesis) 및 PnR(Placement and Route)툴의 오류, 반복된 검증에 따른 설계자의 실수등과 같은 다양한 요인에 의하여 검증의 신뢰성을 상실하기 쉽고, 합성 및 PnR결과를 검증하기 위해 게이트 수준(gate level) 시뮬레이션 및 LVS(Layout versus Schematic)와 같은 검사를 수행하나 이로 인해 많은 검증시간이 소요되며 이러한 검사 또한 완벽한 검증을 보장할 수 없다.

두 번째, SoC는 modular testing을 통하여 테스트를 진행하며 이를 위해 원래의 회로에 TAM(Test Access Mechanism) 및 래퍼(wrapper)등과 같은 테스트를 위한 추가회로를 장착한다. 일반적으로 디자인 설계는 디자인 엔지니어(design engineer)가, 테스트는 테스트 엔지니어(test engineer)가 따로 담당하여 칩을 제작하기 때문에 테스트 엔지니어가 테스트를 위해 회로를 추가할 경우 원래의 기능적인 회로의 기능을 모르는 상태에서 회로를 추가하게 되며 이때 추가된 회로로 인한 기능상의 오류가 발생할 수 있다[14]. 특히 IP(Intellectual Property)를 이용하여 SoC를 설계하는 경우, IP 제공자(vendor)가 내부기술보호를 위해 회로를 제공하지 않을 때 테스트를 위해 추가된 회로로 인한 기능상의 오류가능성은 더욱 커지게 된다[15].

세 번째, 실제 테스트는 여러 단계를 거쳐 진행되며 이러한 테스트에 소요되는 비용은 Rule-of-Ten[1]의 이론에 의해 단계가 높아질수록 소요되는 비용은 전 단계 테스트 비용의 10의 좌승만큼 소비된다. 따라서 테스트 초기단계에서 칩의 오류를 찾게 되면 전체 테스트 비용을 절감할 수 있다. 그러나 기존의 테스트 기법은 설계 단계에서의 검증오류를 찾을 수 없기 때문에 설계 단계에서 발생된 검증오류는 칩이 제작되고 난 후 검증 기능을 확인할 수 있는 제품단계에 와서야 그 이상 유무를 확인할 수 있다.

이와 같은 문제점들의 해결책은 그림 2와 같이 검증과 테스트를 통합하여 수행할 수 있는 칩을 만들고 이를 이용하여 테스트 단계에서 검증을 수행할 수 있도록 하면 앞서 제기한 모든 문제들을 해결할 수 있으며 이와 같은 기능을 갖춘 칩을 SwToC라 명명한다.

검증과 테스트가 혼합된 칩은 다음과 같은 과정을 통하여 구현할 수 있다.

그림 3의 (a)와 같이 디자인 검증은 네트리스트와 같은 디자인 회로를 프로세서가 장착된 워크스테이션

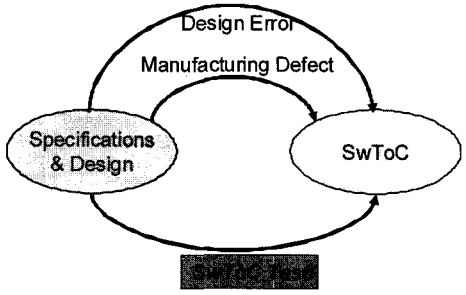

그림 2. SwToC 테스트의 개념
Fig. 2. SwToC test concept.

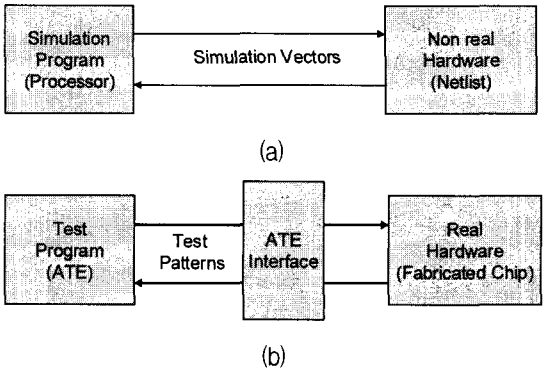

그림 3. (a) 디자인 검증구조 (b) 테스트구조
Fig. 3. (a) Verification structure. (b) Test structure.

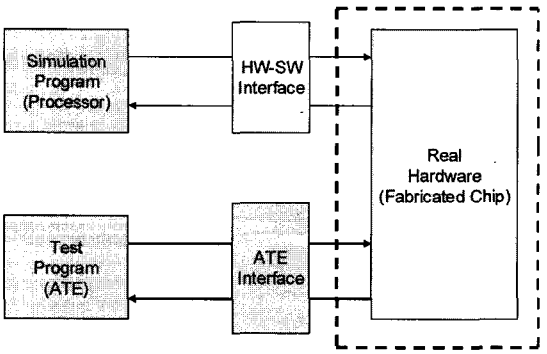

그림 4. HW/SW co-simulation개념의 적용
Fig. 4. Applying HW/SW co-simulation concept.

(workstation)등에서 시뮬레이션 프로그램을 사용하여 모의실험을 수행한다. 테스트의 경우 그림 3의 (b)와 같이 실제 제작된 칩을 ATE로부터 테스트 패턴을 인가받아 테스트를 수행하게 된다.

그림 3에 나타난 바와 같이 디자인 검증은 실제 칩이 아닌 칩의 모델을 시뮬레이션 프로그램을 통하여 모의실험 과정이기 때문에 HW/SW co-simulation의 개념을 적용하여 실제 칩을 검증하도록 하면 그림 4와 같이 테스트와 검증을 함께 수행할 수 있다.

그림 4에 나타난 테스트 및 검증 기법은 많은 문제점을 내포하고 있다. 먼저 검증을 수행하기 위한 추가적인 인터페이스 회로가 필요할 뿐만 아니라 테스트를 수

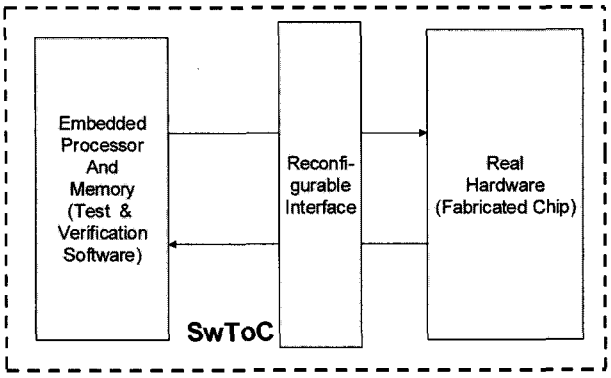

그림 5. 검증 및 테스트의 통합
Fig. 5. Integration of verification and test.

행할 칩 내부의 모듈의 수가 많아지면 외부에 추가적인 많은 I/O 핀(pin)이 필요로 하게 된다(resource constraint problem)[3][7][8][9][10]. 따라서 이를 해결하기 위해 그림 5와 같이 HW/SW 인터페이스 회로와 ATE 인터페이스회로를 재구성 가능한 로직을 사용하여 선택하게 하고 시뮬레이션 프로그램과 테스트 프로그램을 SoC 내의 임베디드 프로세서를 사용하여 수행한다면 고가의 ATE 장비를 사용하지 않고 테스트 및 검증을 수행할 수 있으며 검증 기능이 칩 안에 집적되어 있기 때문에 테스트 단계에서도 디자인 검증이 가능하다.

2. SwToC의 전체 하드웨어 구조

SwToC는 검증 및 테스트가 통합되어 있기 때문에 두 가지 역할을 하는 기능 블록이 칩 안에 내장이 되어 있어야 한다.

먼저 검증을 수행하기 위해서는 검증 프로그램이 수행될 프로세서와 메모리, 검증이 수행 될 코어(core)와의 인터페이스를 위한 회로가 필요하다. 따라서 자체 내장된 임베디드 프로세서와 메모리를 사용하여 검증 프로그램을 수행할 수 있도록 하며, 재구성 가능한 로직을 통해 검증 패턴을 각 코어에 전달한다. 검증패턴의 전달은 그림 6과 같이 Function BUS를 통하여 전달되어 지고 이때 재구성 가능한 로직은 Function BUS와 System BUS와의 브리지(bridge) 역할을 하게 된다.

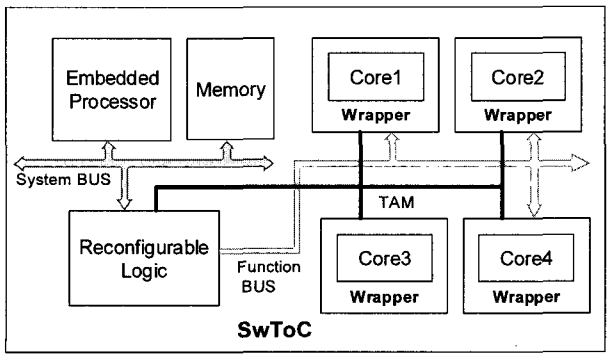

그림 6. 제안하는 SwToC의 구조
Fig. 6. Proposed architecture of SwToC.

테스트의 수행은 SoC의 modular testing을 기본으로 한다. 따라서 modular testing을 위한 ATE, 래퍼, TAM이 필요하게 되며 그림 6과 같이 임베디드 프로세서와 메모리가 ATE의 역할을 수행하며, 독립적인 테스트를 위해 래퍼를 이용한다. 테스트 패턴의 인가는 임베디드 프로세서로부터 전달되어진 패턴을 재구성 가능한 로직을 거쳐 TAM을 통해 해당코어에 전달되어 진다. 따라서 재구성 가능한 로직은 테스트와 검증의 스위치 역할을 담당한다.

현재 SoC설계 중 많은 부분이 IP를 이용하고 있으며, 이때 IP를 제공하는 제공자(vendor)가 자신의 기술을 보호하기 위해 내부를 공개하지 않고 테스트를 위한 패턴만 제공하는 경우가 있다[15]. 이와 같은 상황에서는 검증자체가 불가능 하며 이때는 Core3과 같이 Function BUS의 연결을 생략할 수 있다.

3. 재구성 가능한 로직의 구조

반도체 칩을 재구성하기 위한 구조는 재구성을 위한 로직의 입도(granularity)에 따라 fine-grained와 coarse-grained로 나뉜다[16]. Fine-grained는 FPGA, PLD(Programmable Logic Device)와 같이 단일 비트 단위로 재구성이 가능한 구조이며 coarse-grained는 그림 7과 같이 기 설계된 로직 블록의 집합을 멀티플렉서(multiplexor)를 사용하여 선택함으로써 재구성의 역할을 수행하도록 한 구조이다.

테스트와 검증을 수행할 때는 각기 다른 인터페이스를 사용하여야 하므로 인터페이스를 위한 로직이 변경이 되어야 한다. 따라서 하나의 칩 안에서 두 가지 기능을 선택적으로 사용하기 위하여 재구성 가능한 구조를

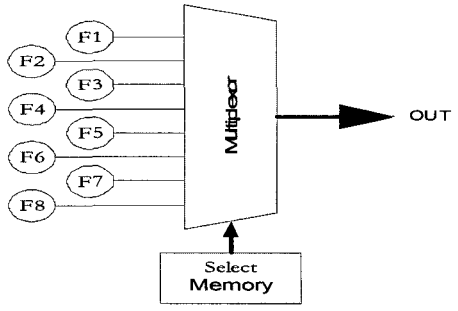

그림 7. Coarse-grained 재구성 가능한 구조
Fig. 7. Coarse-grained reconfigurable architecture.

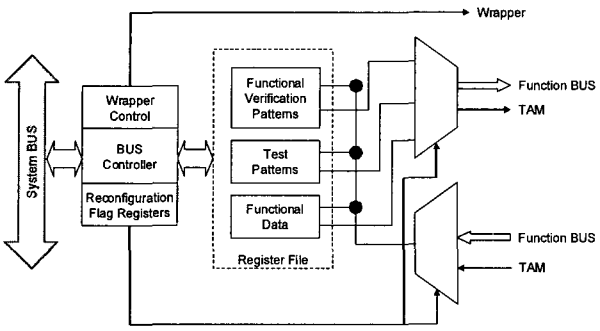

그림 8. SwToC의 재구성 가능한 구조
Fig. 8. Reconfigurable architecture of SwToC.

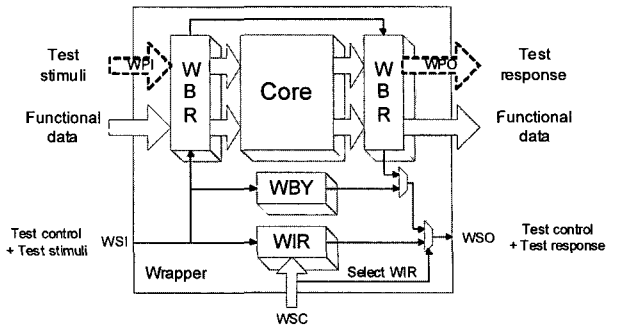

그림 9. IEEE P1500 래퍼의 구조
Fig. 9. Architecture of IEEE P1500 wrapper.

사용할 수 있으며, SwToC에서는 하드웨어의 오버헤드(overhead)가 적은 coarse-grained 형태의 재구성 가능한 구조를 사용하고 이를 그림 8에 나타내었다.

프로세서로부터 전달되어진 패턴은 독립적으로 구성이 되어진 레지스터 파일(register file)에 저장이 되어지며 재구성 정보 레지스터(reconfigurable flag register)에 따라 해당되는 패턴이 선택적으로 전송이 되게 된다. 검증 및 테스트 후의 결과는 레지스터 파일을 통해 프로세서로 전달되어 지며, 프로세서는 결과를 확인하여 검증 및 테스트의 이상 유무를 확인한다.

4. 래퍼의 구조

래퍼는 그림 9와 같이 IEEE P1500 표준 래퍼로 구성되어 진다[17].

IEEE P1500 래퍼는 코어와 주변회로간의 연결 및 분리가 수월 하고 현재 나와 있는 테스트 관련 CAD (Computer Aided Design)툴을 이용해 쉽게 합성해 낼 수 있는 장점이 있기 때문에 본 논문에서 제안한 SwToC의 래퍼는 IEEE P1500을 채택하였으며 다음과 같은 3가지 동작을 수행한다.

(1) 기능 모드
(2) 입력 모드
(3) 출력 모드

기능 모드는 다른 코어와 연결되어 회로 자체의 고유의 기능을 수행하는 모드이며 외부로부터 테스트 및 검증을 수행할 때는 입력 모드로, 수행된 결과를 출력할 때는 출력 모드로 동작한다.

IEEE P1500 래퍼는 테스트 데이터를 직렬(serial) 및 병렬로 입력받거나 출력할 수 있는 WSI(Wrapper Serial Input), WSO(Wrapper Serial Output), WPI (Wrapper Parallel Input), WPO(Wrapper Parallel Output) 포트를 갖추고 있기 때문에 이를 사용하여 검증 및 테스트를 수행할 수 있다. SwToC에서 테스트를 수행할 때는 WSI, WSO 단자를 사용하게 되며, 검증을 수행할 때는 WPI, WPO 단자를 사용한다.

전달된 데이터는 WBR(Wrapper Boundary Register)를 통하여 코어에 전송되고 WBY(Wrapper Bypass Register)를 통해 바로 출력될 수 있다. 또한 제어를 위하여 WIR(Wrapper Instruction Register)와 WSC (Wrapper Serial Control)를 사용한다.

5. 소프트웨어 구조

SwToC는 기존의 테스트 및 검증기법과 달리 칩 안의 내부 소프트웨어를 이용하기 때문에 소프트웨어와 하드웨어 사이의 인터페이스가 필요하며 그림 10과 같은 소프트웨어 소켓을 사용한다.

테스트 및 검증 프로그램은 어셈블러(assembler)를 이용하여 직접 하드웨어를 제어할 수 있도록 할 수도 있으나 많은 수의 코어를 테스트하고 다양한 형태의 검증 프로그램을 수행하기 위해서는 C언어와 같은 고급 언어(high level language)가 적합하다. 따라서 고급 언어가 칩 내에서 하드웨어를 제어하기 위하여 SwToC는 디바이스 드라이버(device driver)와 같은 소프트웨어

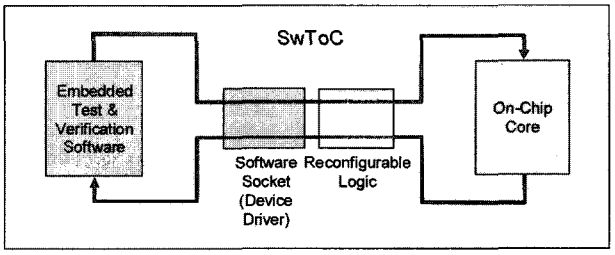

그림 10. SwToC의 소프트웨어 인터페이스
Fig. 10. Software interface in SwToC.

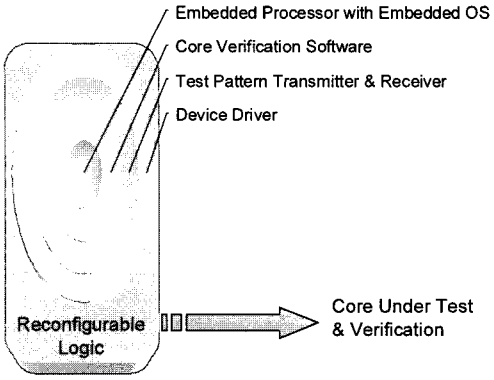

그림 11. SwToC의 소프트웨어 구조
Fig. 11. Software structure of SwToC.

소켓과 내장 운영체제(embedded OS)를 사용하며 이를 그림 11에 나타내었다.

그림 11에 나타난 바와 같이 SwToC는 검증이나 테스트가 수행될 코어와의 소통을 재구성 가능한 로직이 담당하고 재구성 가능한 로직과 검증 및 테스트 프로그램은 디바이스 드라이버와 같은 소프트웨어 소켓을 통하여 연결된다. 전체 프로그램은 내장 운영체제에 의하여 관리된다.

Ⅲ. SwToC의 구현

본 장에서는 SwToC의 실제 응용 가능성 및 성능을 확인하기 위하여 SwToC와 가장 유사한 구조를 갖추고 있는 기존의 칩을 이용하여 SwToC를 구현한다.

1. 구현한 SwToC의 구조

SwToC의 구현을 위해 ARM(Advanced Risc Machine)[18] 코어 기반의 임베디드 프로세서가 내장된 Altera사의 Excalibur[19] 칩을 사용한다. Excalibur는 200MHz의 성능을 갖는 ARM922T 임베디드 프로세서와 APEX20KE의 FPGA 로직으로 구성되어 있다. Excalibur는 재구성 가능한 FPGA와 임베디드 프로세서가 한 칩에 내장되어 있다는 점에서 SwToC와 유사한 구조를 갖고 있으나 결정적으로 테스트가 될 코어가 존재하지 않는다. 그러나 내장된 FPGA 로직의 크기가 충분히 크기 때문에 SwToC의 구성요소인 코어, 래퍼 및 TAM을 FPGA를 통하여 구현할 수 있다. 또한 FPGA를 이용하여 임베디드 메모리를 구현할 수 있으며 프로세서 시스템 안에 AMBA(Advanced Microprocessor Bus Architecture) BUS가 구축되어 있기 때문에 큰 데이터의 저장을 위한 외부 RAM의 사용 및

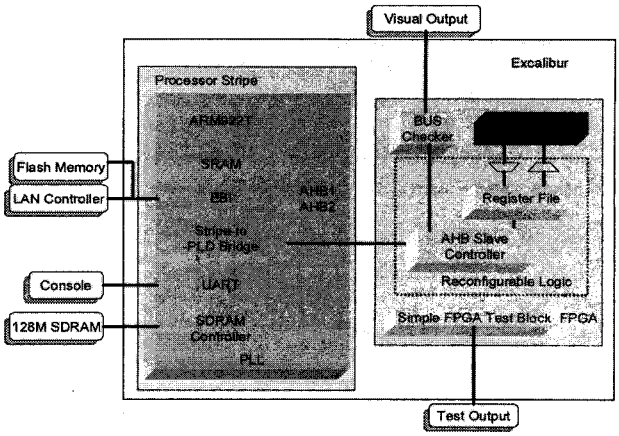

그림 12. 구현한 SwToC의 구조
Fig. 12. Implemented structure of SwToC.

내부 SRAM을 통한 SwToC에서의 저비용 ATE구현 및 외부 인터페이스가 용이하다. 따라서 이를 이용하여 구현한 SwToC의 구조를 그림 12에 나타내었다.

그림 12에 나타낸 바와 같이 구현한 SwToC는 ARM922T를 기본으로 한 임베디드 프로세서와 디자인 검증을 위한 코어 및 코어와의 인터페이스 회로로 구성된다. 프로세서 내에 있는 EBI(External Bus Interface)는 외부로부터의 인터페이스 역할을 하며 본 구성에서는 LAN(Local Area Network) 컨트롤러를 외부에 부착하여 LAN을 통하여 실험에 필요한 데이터를 사전에 SwToC에 전송할 수 있도록 하였다. 또한 플래시 메모리(flash memory)를 장착하여 전원 오프 시 실험용 데이터들이 사라지지 않게 하였다. 운영체제(Operating System)는 외부에 128Mbyte의 SDRAM을 사용하여 포팅(porting)하였다.

구현한 SwToC의 재구성 가능한 로직은 AMBA 버스 제어를 위한 AHB(Advanced High Performance Bus) slave 로직과 버스로부터 전송된 데이터의 임시 보관 장소인 레지스터 파일로 구성되어 있다. 테스트를 위한 코어 또한 FPGA를 이용하여 구현하였고 FPGA 자체의 이상 유무를 판단하기 위한 FPGA Test Block 및 버스 인터페이스의 동작을 확인하기 위한 BUS Checker도 삽입하였다.

실제 SwToC의 기본 요소 중 하나인 래퍼는 구현한 구조에서는 단일 코어를 이용하기 때문에 이를 생략하였으며 코어를 연결하는 TAM 또한 레지스터 파일로부터 나오는 단일라인의 TAM을 구축하였다.

2. 저비용 ATE의 구현

실제 SwToC의 검증 및 테스트의 성공(success) 또

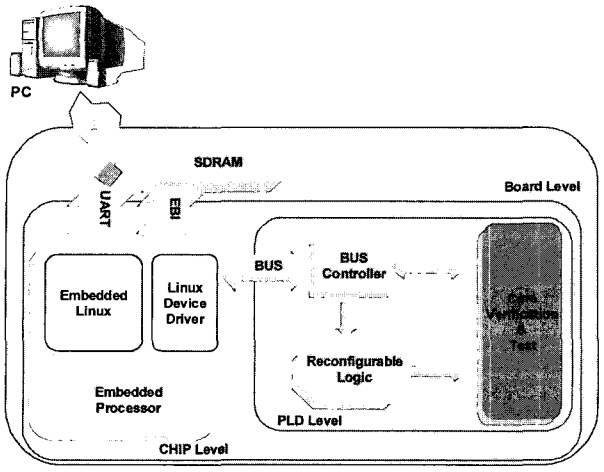

그림 13. 저비용 ATE 구현
Fig. 13. Implementation of low cost ATE.

는 실패(fail) 여부는 테스트 포인트 또는 칩의 외부 핀을 사용하나 본 장에서 구현한 SwToC에서는 검증 및 테스트 과정을 직접 눈으로 확인함과 동시에 저비용 ATE의 역할을 수행하기 위하여 그림 13과 같은 ATE 환경을 구축하였다.

그림 13과같이 데이터 입출력이 가능한 단말기(terminal)만 있으면 ATE의 역할을 수행할 수 있고 구현한 SwToC에서는 일반 PC(Personel Computer)를 사용하여 ATE의 역할을 수행하도록 하였다.

PC는 SwToC의 UART(Universal Asynchronous Receiver/Transmitter)를 통하여 데이터를 주고받을 수 있으며 내장 운영체제 장착 및 디바이스 드라이버 구축을 위하여도 사용된다. 디바이스 드라이버는 재구성 가능한 로직에 구성된 레지스터 파일을 하나의 장치로 인식하여 검증에 필요한 검증 패턴의 인가 및 검증 결과를 내장 운영체제에 전송하는 역할을 담당한다. 내장 운영체제에 전송된 결과는 UART를 통하여 PC에서 실시간으로 확인이 가능하다.

3. 재구성 가능한 로직의 구현

구현한 SwToC는 ARM을 기본 프로세서로 사용하기 때문에 재구성 가능한 로직의 버스 컨트롤러를 구현하기 위해 AMBA BUS를 구축하였다. AMBA BUS는 AHB, ASB(Advanced System Bus), APB(Advanced Peripheral Bus)로 구성되어 있으며, 다중 마스터(multi-master)의 기능을 갖고 있으나 SwToC는 단일 프로세서만을 사용하므로 그림 14와 같이 프로세서를 마스터로 구성한 단순한 형태의 AMBA BUS를 사용한다.

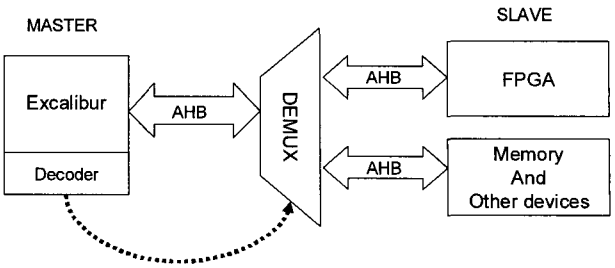

그림 14. 단순화된 AMBA BUS
Fig. 14. Simplified AMBA BUS for implementation.

단순화된 AMBA BUS는 AHB만을 사용하며 그림 14에 나타낸 바와 같이 디코더(decoder)만을 이용하여 버스를 제어한다. 따라서 FPGA로직 안에 AHB slave 컨트롤러를 삽입하여 프로세서와 데이터를 교환할 수 있으며, 제안한 그림 8의 재구성 가능한 로직을 FPGA 안에 구성하였다. 버스를 통해 전송된 데이터는 코어에 전송되기 전에 재구성 가능한 로직의 구성요소인 레지스터 파일에 저장되게 된다. 본 구현에서는 코어의 입력 값을 래치(latch)회로를 통해 레지스터 파일의 역할을 수행하게 하였으며 코어의 출력은 바로 버스에 전송되도록 하였다.

4. 코어의 구현

검증 및 테스트가 수행될 코어는 ISCAS-85 벤치마크 회로(benchmark circuit)를 선택하였다[20]. ISCAS-85 벤치마크 회로는 1985년 International Symposium of Circuits and Systems에서 처음으로 네트리스트 형태의 회로들이 소개된 산업용 회로들로서 그 후 많은 테스트 분야에서 벤치마크 회로로 사용되어 왔다. 그러나 구조 및 각 회로의 기능들에 대해서는 알려지지 않고 있다가 1999년 Hansen, Yalcin, Hayes가 리버스 엔지니어링(reverse engineering)을 통해 각 회로의 기능 및 구조를 분석하여 학회에 발표하였다[21].

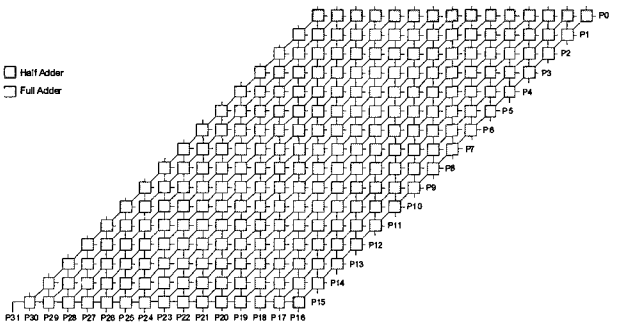

그림 15. C6288회로의 구조
Fig. 15. Structure of C6288 circuit.

검증 및 테스트가 통합되어 수행되는 SwToC에서는 내부의 구조 및 기능이 알려진 벤치마크 회로가 필요하며 본 구현에 사용된 코어는 발표된 회로 중 C6288회로를 선택하였다. C6288회로는 16bit 곱셈기(multiplier)로서 검증 및 테스트의 결과를 확인하기 용이할 뿐만 아니라 FPGA로의 구현이 쉽기 때문에 본 구현실험에 코어로서 채택이 되었으며 그 구조는 그림 15와 같다.

5. 소프트웨어 환경

SwToC는 자체적으로 내장 운영체제가 장착되어 있으나 본 설계에서는 FPGA를 사용하여 SwToC의 기능을 모의실험하기 때문에 따로 내장 운영체제를 포팅하는 작업이 필요하다. 본 설계에서는 임베디드 리눅스(embedded linux)를 내장 운영체제로 사용하였으며 전체 구축작업은 다음과 같은 단계를 거친다.

(1) 부트 로더(boot loader) 설치
(2) 커널 컴파일(kernel compile)
(3) 커널 및 디바이스 드라이버 설치
(4) 검증 및 테스트 프로그램의 크로스 컴파일
(5) 검증 및 테스트 프로그램 설치

부트 로더는 운영체제와 설계된 하드웨어사이의 인터페이스를 위한 프로그램으로 임베디드 리눅스를 구현한 SwToC에 포팅할 수 있도록 한다. 부트 로더를 사용하여 임베디드 리눅스의 커널을 장착하기 위해서는 기존의 커널을 구현한 하드웨어에 맞게 컴파일 하여야 하고 커널 컴파일 과정이 완료되면 바이너리(binary) 형태의 커널이 생성된다. 생성된 커널은 전 단계에서 설치했던 부트 로더를 이용해 임베디드 리눅스를 시스템 메모리로 불러와 운영체제를 구동하게 된다.

실제 SwToC에서는 패턴 생성프로그램이 내부에 장착되어 바로 테스트 및 검증이 가능하나 본 구현에서는 외부로부터 LAN을 이용하여 테스트 및 검증 프로그램을 다운로드(download) 받도록 하였다. 따라서 테스트 실행 소프트웨어는 외부 컴퓨터에서 만들어 지며 외부 컴퓨터에서 만들어진 실행코드를 구현한 SwToC 내부의 ARM 프로세서에서 동작시키도록 하기 위해서는 크로스 컴파일 과정을 거쳐야만 한다. 디바이스 드라이버 또한 크로스 컴파일 과정을 거쳐서 생성이 되며 생성된 실행파일은 LAN을 통하여 구현한 SwToC에 장착되어 검증 및 테스트가 수행될 수 있도록 하였다.

Ⅳ. 실험 및 결과

1. 실험 환경

구현에 사용된 Excalibur는 동작속도 및 내장 FPGA의 크기 등에 따라 여러 종류로 분류되며 실험을 위한 구현에 사용된 칩은 EPXA4F1020C3이다.

EPXA4F1020C3는 400K 게이트 용량의 FPGA와 ARM922T 임베디드 프로세서를 내부에 탑재하고 있고 내부에 128KB의 SRAM과 64KB의 DPRAM(Dual Port RAM) 또한 장착이 되어있으며 부가적으로 PLL(Phase Locked Loop), UART 및 Timer등을 내장하고 있다.

외부 장치의 연결은 이미 제작된 휴인스사의 XP-400 SoCMaster 교육용 장비를 이용하였고 장비에 포함된 EPXA4F1020C3를 사용하여 SwToC를 실험하기 위한 환경을 구축하였다. 실제 SwToC에는 칩 내부에 메모리가 장착되어 동작되나 본 실험에 사용된 EPXA4F1020C3 칩은 내부 메모리가 부족하므로 외부에 128MB의 SDRAM을 장착하여 실험하였다.

운영체제 구축을 위해 사용된 임베디드 리눅스는 개발 일자에 따라 여러 가지 버전이 공개되어 있으며, 본 실험에서는 2.4.19-rmk7 버전의 커널을 사용하였다. 부트 로더의 구축은 인터넷에 공개되어 있는 부트 로더 중 Blob 부트 로더를 설계된 SwToC에 맞게 수정하여 사용하였다. 디바이스 드라이버, 검증, 테스트 프로그램은 직접 C언어를 사용하여 코딩하였으며 전체 실험 환경을 표 1에 나타내었다.

표 1. 전체 실험 환경
Table 1. Overall experiment environment.

Software	
OS	Embedded Linux Kernel 2.4.19-rmk7
Boot Loader	Modification of Blob Boot Loader
Device Drivers	LAN, Console, Reconfigurable Logic Interface
Application Software	C6288 Test Software C6288 Verification Software
Hardware	
Device	EPXA4F1020C3
External Memory	128Mbytes SDRAM, 16Mbytes Flash Memory
BUS Control	AMBA AHB Slave Controller
Core	ISCAS-85 C6288

2. 구현결과 및 동작확인

최종적인 구현은 Altera사의 Quartus II 프로그램을

이용하여 그림 16과 같이 설계한 후 GCC 컴파일러와 ARM ADS v1.2 컴파일러를 이용하여 소프트웨어를 컴파일한 후 XP-400보드에 장착하였다.

결과확인을 하기 전에 설계된 블록 및 장착과정이 정확히 됐는가를 확인하기 위하여 보드에 장착된 7 세그먼트 LED(Light Emitting Diode)를 통하여 정상동작의 유무를 확인하였다.

소프트웨어와 하드웨어를 같이 설계하는 경우 소프트웨어의 오류로 인해 검증패턴이 제대로 인가되지 않은 상태에서 소프트웨어만의 출력을 보고 검증의 성공여부를 오판하기 쉽다. 따라서 설계된 AHB 컨트롤러의 데이터 출력 버스를 코어뿐만 아니라 7 세그먼트 LED에도 동시에 연결하여 검증프로그램으로부터 디바이스 드라이버를 통하여 전송된 패턴이 정상적으로 코어에 인가되는가를 확인하였으며 UART를 통하여 검증 및 테스트의 진행여부를 확인하였다.

테스트는 Synopsys사의 TetraMax ATPG 프로그램을 이용하여 결정론적(deterministic) 테스트 패턴을 만든 후 이를 설계된 SwToC내의 테스트 소프트웨어를 사용하여 순차적으로 패턴을 코어에 인가하여 테스트를 수행하였다.

검증은 그림 17과 같이 두 개의 피연산자(operand)를 하나의 32비트 버스로 출력하여 연산 결과를 비교하였다.

확인 결과 테스트 및 검증 결과가 정상적임을 확인하였으며 제안한 SwToC가 실무에서의 적용이 가능함을 확인하였다.

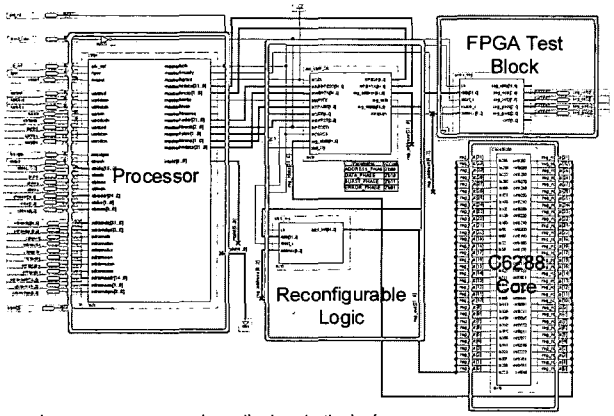

그림 16. SwToC 하드웨어 설계결과
Fig. 16. Hardware design of SwToC.

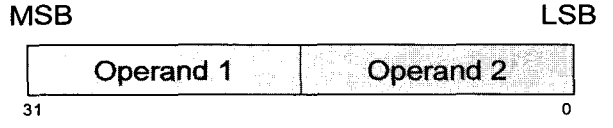

그림 17. 검증용 패턴의 구성
Fig. 17. Patterns for verification.

3. 검증 시간에 대한 비교

본 실험에서는 참고적인 비교를 위하여 설계된 SwToC를 이용하여 검증 패턴의 수를 점차적으로 증가시켜가며 PLI를 사용한 방법과 SwToC상에서의 검증시간을 비교하였다.

먼저 PLI를 사용하기 위하여 verilog 형태의 C6288 회로를 Hynix 0.35 마이크론 공정용 셀 라이브러리를 상용하여 합성한 후, 최종 연산결과가 나오기 위한 가장 긴 경로(critical path)를 조사하였다. 조사된 결과는 32.7 나노초 이었으며 각 패턴의 입력에 대하여 32.7 나노초의 지연(delay)을 시킨 후 출력된 결과를 PLI 인터페이스를 통하여 C언어로 작성된 검증 프로그램에 전달하였다. PLI를 이용한 검증에 사용된 장비는 350MHz 성능의 UltraSparc2 프로세서를 장착한 SUN 워크스테이션을 사용하였고 PLI 검증에는 Cadence사의 Verilog-XL 시뮬레이터를 이용하였으며 비교 결과를 표 2에 나타내었다.

실험 결과에서 나타난 바와 같이 SwToC를 사용한 검증은 200MHz로 동작하는 ARM 프로세서를 이용하여 구현되었음에도 불구하고 기존의 일반 검증기법에 비해 월등히 빠른 속도를 나타내었다. 또한 특이해야 할 사항은 일반 기법은 그림 18과 같이 검증패턴의 형태에 따라 시뮬레이션 소요시간이 다르게 나타난다는 것이다.

그림 18에서 수행된 패턴 실험은 총 100000개의 패

표 2. SwToC 기능 검증과 PLI를 이용한 기능 검증의 시간비교 (단위: 초)
Table 2. Comparison between SwToC verification and traditional PLI verification. (unit: seconds)

Number of Patterns	PLI Verification Time					SwToC Verification Time
	Compile Time	Link Time	Sim. Time	Ver. Time	Total	
100	0.4	0.7	0.1	0.1	1.3	0.04
1000	0.4	0.7	1.9	0.1	3.1	0.05
5000	0.4	0.7	13.9	0.2	15.2	0.08
10000	0.4	0.7	30.6	0.9	32.6	0.12
50000	0.4	0.7	190.2	3.7	194.8	0.47

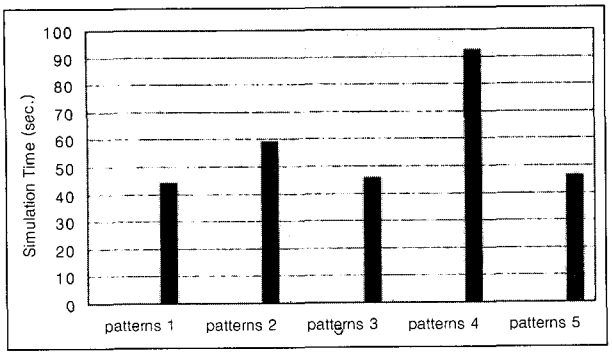

그림 18. 패턴의 형태에 따른 시뮬레이션 시간의 변동
Fig. 18. Variation of simulation time according to type of patterns.

턴을 사용하였다. Pattern 1은 0부터 100000까지의 숫자를 각각 두 피연산자로 사용하였고 pattern 2는 0부터 7씩 피연산자의 값을 증가시켜 가며 검증 코어에 인가하였다.

Pattern 3는 10000000부터 10100000까지의 값을 pattern 1과 동일하게 인가시켜 실험하였으며 pattern 4는 0부터 17100000까지 값을 171씩 증가시킨 값을 오퍼랜드로 사용하였다. 마지막 pattern 5는 700000부터 900000까지 그 값을 2씩 증가시켰다.

이와 같이 패턴의 종류에 따라 시뮬레이션 시간이 차이가 나는 이유는 일반 시뮬레이션이 이벤트 구동(event-driven) 방식으로 진행되기 때문에 패턴의 종류에 따라 발생되는 이벤트가 다른데서 기인한 것이다. 그러나 SwToC를 사용한 검증에서는 패턴의 종류가 달라도 동일한 결과를 나타내었다.

4. SwToC의 오버헤드

반도체 칩을 테스트하기 위해서는 테스트를 위한 추가적인 회로가 필수적이다. SwToC도 마찬가지로 재구성 가능한 로직, 래퍼 등이 원래의 회로에 추가되며 본 실험에서는 래퍼를 사용하지 않았으므로 재구성 가능한 로직에 인가되는 오버헤드만을 평가한다. 또한 테스트를 칩 내부에서 수행하기 때문에 이때 소요되는 메모리의 크기를 평가하기 위하여 C6288회로를 테스트 할 때 사용된 프로그램의 크기를 조사하였고 이를 표 3에 나타내었다.

표 3에 나타난 운영체제의 커널과 부트 로더는 테스트 및 검증이외에도 실제 칩의 기능적 역할을 수행할 때 반드시 필요한 요소이므로 실제 SwToC의 소프트웨어로 인한 오버헤드로 인해 요구되는 메모리 량은 디바이스 드라이버와 테스트 프로그램의 크기를 합한 291K

표 3. SwToC의 오버헤드
Table 3. Overhead of SwToC.

Hardware					
	Logic cells		Registers	Number of pins	
Register file	35		32	73	
Bus controller	85		46	208	
Software					
	Line # of source	Object code size	Executable code size	Compressed image size	Required memory
Device Driver	162	132728	N/A	N/A	160K
Program	195	4216	14468	N/A	131K
OS Kernel	N/A	N/A	N/A	978K	978K
Boot Loader	4495	441685	N/A	N/A	431K

가 된다. 따라서 이와 같은 오버헤드는 실무 응용에 충분히 가능하다. 또한 표 3의 실험결과에서와 같이 SwToC는 적은 양의 메모리와 재구성 가능한 로직의 추가로 고가의 ATE없이 테스트와 검증을 수행할 수 있기 때문에 검증 기능은 물론 ATE에 소요되는 테스트 비용 또한 절감할 수 있음을 알 수 있다.

V. 결론 및 향후 연구과제

본 논문은 테스트와 검증이 통합되어 수행될 수 있는 새로운 SoC 구조를 제안하고 이를 SwToC라 명명하였다. 제안한 SwToC는 임베디드 프로세서, 재구성 가능한 로직, TAM 및 래퍼로 구성되며 제안한 SwToC에 대한 평가를 위하여 기존의 상용 칩을 이용하여 구현하였다. 구현한 SwToC는 ISCAS-85 회로 중 내부 동작이 알려져 있는 회로를 선택하여 코어 형태로 구현한 후 임베디드 운영체제 포팅 및 검증 프로그램을 통하여 구현한 코어가 SwTOC내에서 실제 검증이 가능한지를 조사하였고 이에 따른 검증 시간 및 오버헤드를 평가하였다. 실험 결과 가상 SwToC는 현재의 SoC기술로 충분히 실무에 적용이 가능하며, 1,000,000개 이상의 패턴 실험을 통하여 일반적인 PLI 검증 기법에 비해 월등한 속도를 나타냄을 확인하였다. 또한 C6288회로의 예를 통하여 일반 FPGA를 사용 시 120개의 로직블록과 78개의 레지스터들을 사용하여 재구성 가능한 로직을 만들 수 있음을 확인하였고 저비용 ATE를 사용하여 테스트가 가능함을 확인하였다.

차후 연구과제로는 재구성 가능한 구조를 확장하여 ATE비용뿐만 아니라 전반적인 테스트 비용을 낮추는 연구를 진행할 예정이다.

참 고 문 헌

[1] Alexander Miczo, Digital Logic Testing and Simulation, John Wiley & Sons, New Jersey, 2003.

[2] W. Wolf, "A decade of hardware/software codesign," Computer Volume 36, Issue 4, pp. 38 - 43, April 2003.

[3] Q. Xu and N. Nicolici, "Resource-constrained system-on-a-chip test: a survey," IEE Proceedings on Computers and Digital Techniques, Vol. 152, Issue 1, pp. 67-81, Jan. 2005.

[4] D. Becker, R. K. Singh, and S. G. Tell, "An engineering environment for hardware/software co-simulation," Proceedings on Design Automation Conference, 29th ACM/IEEE 8-12, pp. 129 - 134, June 1992.

[5] D. E. Thomas, J. K. Adams, and H. Schmit, "A model and methodology for hardware-software codesign," Design & Test of Computers, IEEE Volume 10, Issue 3, pp. 6 - 15, Sept. 1993.

[6] M. Borgatti, R. Rambaldi, G. Gori, and R. Guerrieri, "A smoothly upgradable approach to virtual emulation of HW/SW systems, " Rapid System Prototyping, 1996. Proceedings., Seventh IEEE International Workshop on 19-21, pp. 83 - 88 June 1996.

[7] A. Jas and N. A. Touba, "Using an embedded processor for efficient deterministic testing of systems-on-a-chip," International Conference on Computer Design(ICCD'99), pp. 418-423, Oct. 1999.

[8] K. J. Balakrishnan and N. A. Touba, "Matrix-based test vector decompression using an embedded processor," IEEE International Symposium on Defect and fault Tolerance in VLSI Systems, pp. 159-165, Nov. 2002.

[9] Gang Zeng and Hideo Ito, "Efficient test data decompression for system-on-a-chip using an embedded FPGA core," 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 503-510, Nov. 2003.

[10] Z. He, G. Jervan, Z. Peng, and P. Eles, "Hybrid BIST test scheduling based on defect probabilities," 13th Asian Test Symposium, pp. 230-235, Nov. 2004.

[11] S. Koranne, "Design of reconfigurable access wrappers for embedded core based SoC test," IEEE Trans. on VLSI Systems, Vol.11, Issue 5, pp. 955-960, Oct. 2003.

[12] Li Lei and K. Chakrabarty, "Deterministic BIST based on a reconfigurable interconnection network," International Test Conference (ITC' 2003), pp. 460-469, 2003.

[13] J. J. Hallenbeck, J. R. Cybrynski, N. Kanopoulos, T. Markas, N. Vasanthavada, N., "The Test Engineer's Assistant: a support environment for hardware design for testability," Computer, Vol. 22, Issue 4, April 1989, pp. 59 - 68

[14] H. T. Nagle, S. C. Roy, C. F. Hawkins, M. G. McNamer, R. R. Fritzemeier, "Design for testability and built-in self test: a review," IEEE Transactions on Industrial Electronics, Vol. 36, Issue 2, May 1989, pp. 129 - 140

[15] M. S. Quasem, Zhigang Jiang and S. K. Gupta, "Benefits of a SoC-specific test methodology," IEEE Trans. on Design & Test of Computers, Vol. 20, Issue 3, pp. 68-77, May 2003.

[16] Katherine Compton and Scott Hauck, "Reconfigurable computing: a survey of systems and software," ACM Computing Surveys (CSUR), Vol. 34, Issue 2, pp. 171-210, June 2002.

[17] E. J. Marinissen, R. Kapur, Y. Zorian, "On using IEEE P1500 SECT for test plug-n-play," International Test Conference(ITC'2000), pp. 770-777, Oct. 2000.

[18] www.arm.com

[19] www.altera.com

[20] www.cbl.ncsu.edu/www/CBL_Docs/iscas85.html

[21] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," IEEE Trans. on Design & Test of Computers, Vol. 16, Issue 3, pp. 72-80, Jul. 1999.

저 자 소 개

김 남 섭(정회원)
1990년 경희대학교 전자공학과 학사
1992년 경희대학교 전자공학과 석사
2000년~현재 경희대학교 전자공학과 박사 과정.
<주관심분야 : 반도체 설계, 스테레오 비젼>

조 원 경(정회원)
1971년 경희대학교 전자공학과 학사
1973년 한양대학교 전자공학과 석사
1986년 한양대학교 전자공학과 박사
1980년~현재 경희대학교 전자공학과 정교수
<주관심분야 : VLSI설계, 마이크로프로세서>