32비트 RISC 프로세서를 위한 JTAG 기반의 재사용 가능한 임베디드 디버거 설계

정 대 영, 최 광 재, 곽 승 호, 이 문 기
연세대학교 전기전자공학과
전화 : 02-2123-4731 / 핸드폰 : 018-292-1876

Design of the Reusable Embedded Debugger for 32bit RISC Processor Using JTAG

Dae-Young Jung, Kwang-Jae Choi, Sung-Ho Kwak, Moon-Key Lee
Dept. of Electrical and Electronic Eng., Yonsei University
E-mail : jdy21@spark.yonsei.ac.kr

Abstract

The traditional debug tools for chip tests and software developments need a huge investment and a plenty of time. These problems can be overcome by Embedded Debugger based the JTAG boundary Scan Architecture. Thus, the IEEE 1149.1 standard is adopted by ASIC designers for the testability problems. We designed the RED(Reusable Embedded Debugger) using the JTAG boundary Scan Architecture. The proposed debugger is applicable for not a chip test but also a software debugging. Our debugger has an additional hardware module (EICEM : Embedded ICE Module) for more critical real-time debugging

Ⅰ. 서 론

칩 설계의 복잡도가 증가함에 따라 시스템 테스팅과 소프트웨어 개발을 위한 디버깅에 사용되는 비용과 시간은 프로세서 및 프로그램의 시장 가격에 큰 비중을 차지한다. 따라서 제품의 시장 경쟁력을 갖추기 위해서는 이러한 비용과 시간을 줄여야만 한다.

그러나 IMS 장비나 로직 애널라이져에 의한 전통적인 테스팅 및 디버깅 방법은 많은 비용 뿐 아니라 많은 시간을 요구한다. 더욱이 VLSI 설계가 고집적화 되고 동작 주파수가 높아짐에 따라 검증과 디버깅은 더욱 어려워지고 있다.[1]

위와 같은 문제들을 해결하기 위하여 ASIC 설계자들은 마이크로프로세서 상에서 소프트웨어의 동작을 검증하기 위한 in-circuit 디버거를 내장시키고 있다.[2]

본 논문의 디버거는 JTAG (Joint Test Action Group ; IEEE 표준 1149.1)을 기반으로 하여 32bit RISC 마이크로프로세서에 적합한 in-circuit 디버거를 구현하였다. 또한 좀 더 세밀한 디버깅을 위한 모듈 (Embedded ICE Module : EICEM)을 추가하였다.

본 논문의 디버거는 PC와 연결하여 단독으로 동작을 수행한다. 따라서 IMS나 로직 애널라이저와 같은 고가의 장비가 필요 없기 때문에 칩 테스와 소프트웨어 디버깅을 위한 시간과 비용을 줄일 수 있다.

설계의 재사용을 고려하여 상위 수준에서 구조를 정의하고 설계하였고 설계된 디버거는 0.25㎛ 오중 금속 CMOS 표준 라이브러리를 이용하여 구현되었다.

본 논문의 Ⅱ장에서는 디버그 시스템과 RED (Reusable Embedded Debugger)의 구조를 소개하고, Ⅲ장에서는 RED를 위한 Boundary Scan Architecture와 스캔 체인에 대해 언급하였다. Ⅳ장에서는 RED의 디버깅 방법에 대해서 설명하고 Ⅴ장에서는 RED의 합성 결과 및 32bit RISC 프로세서에 실장하여 테스트한 결과를 분석하였다. 마지막으로 Ⅵ장에서는 본 논문의 결론을 도출하였다.

Ⅱ. RED의 구조

JTAG을 기반으로 하는 디버거 시스템은 일반적으로 크게 세 개의 부분으로 구성된다. 디버거 호스트, 프로토콜 변환기 그리고 타겟 시스템이다. 디버거 호스트는 디버거 소프트웨어 프로그램을 탑재한 개인용 컴퓨터나 웍스테이션이 될 수 있다. [그림 1]처럼 디버거 호스트는 프로토콜 변환기를 거쳐 타겟 시스템과 연결된다.

그림 1. The debug system

제안된 RED는 타겟 시스템의 프로세서와 함께 집적되어 JTAG을 통해 프로토콜 변환기와 직접 연결된다.

RED는 세 개의 블록으로 구성된다. Boundary Scan Architecture, EICEM 그리고 Boundary Scan 레지스터이다.

*Boundary Scan Architecture*는 EICEM과 Boundary Scan 레지스터의 동작을 제어하며, EICEM은 세부적인 디버깅을 위한 추가적인 모듈이며, Boundary Scan 레지스터는 프로세서와 EICEM의 입/출력 핀에 연결되어 데이터 및 신호를 추출하거나 변경시키는 기능을 한다. 특히 EICEM을 둘러싼 Boundary Scan 레지스터는 EICEM의 레지스터에 데이터를 직렬로 읽고 쓰기를 하도록 함으로써 부가적인 데이터 버스로 인한 면적을 줄일 수 있고 EICEM의 레지스터의 추가 및 제거가 용이할 수 있다. RED의 대략적인 블록 다이어그램은 [그림 2]와 같다.

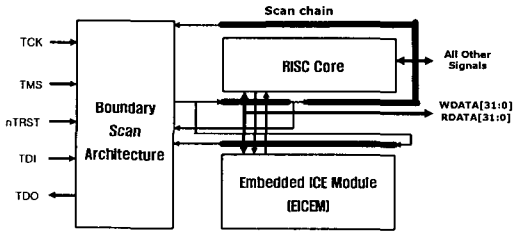

그림 2. The RED diagram

Ⅲ. Boundary Scan Architecture와 스캔 체인

[그림 3]은 RED를 위한 Boundary Scan Architecture를 보여준다. 그리고 [표 1]은 이 Boundary Scan Architecture의 명령어 셋이다.

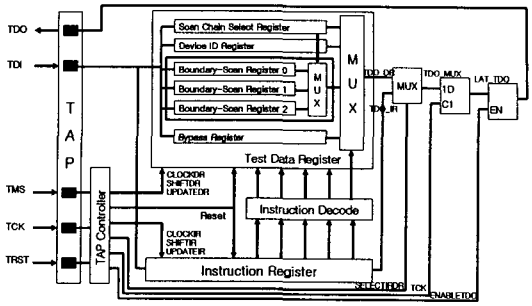

그림 3. The Boundary Scan Architecture

표 1. Instructions for Boundary Scan Architecture

Instructions	Binary Code
EXTEST	0000
SCAN_N	0010
INTEST	1100
IDCODE	1110
BYPASS	1111

일반적인 Boundary Scan Architecture는 한 개의 스캔 체인만을 갖는다. 따라서 EXTEST와 INTEST 명령어시 자동적으로 스캔 체인이 선택된다.[3][4][5] 그러나 RED는 세 개의 스캔 체인을 갖고 있다. 이 스캔 체인들을 선택하기 위한 Scan Chain Select Register와 명령어(SCAN_N)을 추가하였다.

[그림 4]는 세 개의 스캔 체인의 연결을 보여 주고 있다. 이들은 테스트와 디버깅과 EICEM을 프로그래밍하기 위하여 사용된다. 이러한 스캔 체인은 Boundary Scan Architecture를 통하여 제어된다. 하지만 이 스캔 체인은 JTAG에 완전히 부합하지는 않는다.[5]

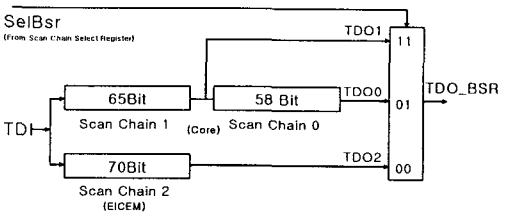

그림 4. The Boundary Scan Chains

스캔 체인 0는 코어의 데이터 버스와 각종 컨트롤 신호들의 입출력을 둘러싸고 있어서 코어의 입출력 데

이터를 제어한다. 스캔 체인 1은 스캔 체인 0의 부분이며 데이터 버스와 브레이크 포인트 신호만을 제어한다. 스캔 체인 2는 EICEM의 레지스터에 데이터를 읽고 쓸 수 있도록 한다. 스캔 체인 2는 입력 핀이 없기 때문에 스캔 체인 0,1과 달리 시프트 레지스터로 구현하였다.

Ⅳ. EICEM 구조 및 디버깅 방법

코어가 일반 동작을 할 때 RED는 코어와 격리되기 때문에 코어에 영향을 미치지 않는다. 그러나 디버깅 동작을 수행할 때는 RED에 의해 코어의 동작이 제어된다. 이를 디버그 상태라고 한다. 이는 디버그 상태에서 코어의 내부 레지스터 상태를 조사할 수 있고 필요하다면 입력/출력 핀의 값을 스캔 체인 0를 통해 변경시킬 수 있다.

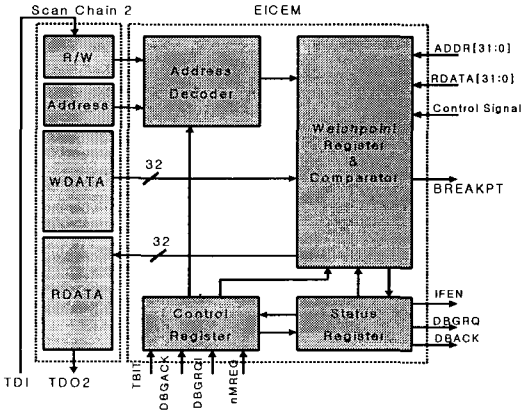

그림 5. The EICEM block diagram

[그림 5]에서 보는 바와 같이, EICEM은 크게 4개의 block으로 이루어진다. 스캔 체인 2를 통해 주소와 데이터 값이 EICEM에 전달된다. 주소값이 주소 디코더에 입력되면 watchpoint 레지스터가 선택된다. 레지스터가 선택되면 R/W 값에 따라 레지스터의 값이 읽혀지거나 쓰여지게 된다. 이러한 과정을 거쳐서 EICEM 레지스터에 특정 데이터나 주소 값에 해당하는 breakpoint와 watchpoint를 프로그래밍한다. 비교기에서는 코어의 주소와 데이터 버스와 제어 신호를 항상 검색하여 EICEM 레지스터에 프로그램 된 breakpoint나 watchpoint값과 일치할 때 BREAKPT 신호를 코어에 보내게 된다. BREAKPT 신호를 받은 코어는 디버그 승인 신호를 EICEM에 보내게 되는데 breakpoint와 watchpoint이냐에 따라 시기가 다르다. breakpoint 경우는 명령어가 코어 파이프라인의 실행 스테이지에 도착했을 때 승인 신호를 보내게 되고 watchpoint 경우는 명령어가 완전히 수행되었을 때 승인 신호를 보내게 된다.

코어는 디버그 요구시 디버그 상태로 들어간다. 이러한 경우는 EICEM 레지스터를 프로그램하는 경우와 외부 디버그 요구시에 일어나게 된다. 외부 디버그 요구 신호는 비동기 입력 신호이기 때문에 코어에 보내기전에 EICEM에서 동기를 맞추어 준다.

일단 코어가 디버그 상태로 들어가면 코어는 캐쉬에서 명령어를 가져오는 것을 멈추고 메모리 및 다른 외부 시스템과 격리된다. 비로소 RED는 스캔 체인 1을 통해 파이프라인에 명령어를 읽을 수 있다. 또한 레지스터 및 메모리 상태를 조사할 수 있다.

컨트롤 레지스터는 외부 디버그 요구 신호와 코어 신호를 통해 EICEM의 다른 부분들을 제어하고 스테이터스 레지스터는 EICEM의 상태를 저장한다.

Ⅴ. 합성 및 테스트 결과

RED는 Verilog HDL을 이용하여 합성 가능한 수준으로 설계 및 검증되고 Synopsys 사의 Design Compiler를 이용하여 합성하였다. 라이브러리는 0.25㎛ 오중 금속 CMOS 표준 라이브러리를 이용하였다.

설계의 재사용을 고려하여 상위 수준에서 구조를 정의하고 설계하였다. Top-down 방식의 ASIC 설계에 있어서 합성 가능한 형태로 HDL모델을 구현할 때 기능 모듈의 구성을 체계화함으로써 하위 시스템으로 설계된 블록의 집적, 합성, 검증을 수행하는 시간을 줄일 수 있다.

[그림 6]은 시뮬레이션 결과의 파형을 보여주고 [표 2]는 합성한 결과이다. RED는 약 3174 게이트 구성되고 최대 동작 주파수는 385MHz 이다.

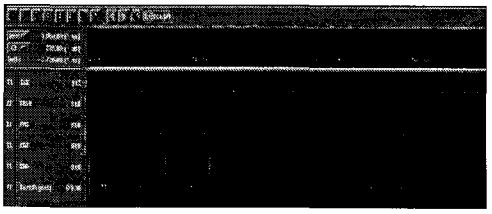

그림 6. Waveform of Simulation

RED는 어떤 32bit RISC 프로세서에서든지 내장되어 디버거 기능을 수행할 수 있으나 테스트를 위하여 연세대학교에서 개발한 32bit RISC 프로세서를 사용하였다.[8] [표 3]은 그 결과를 보여준다.

표 2. Gate Counts of modules

Module	Gate Counts
EICEM	412
Boundary Scan Architecture	446
Boundary Scan Chain 0,1	1662
Boundary Scan Chain 2	654
Total	3174
Max frequency	385MHz

표 3. Overhead of RISC Core with RED

Design	RISC Core	RISC Core + RED	Overhead (%)
Gate counts	50000	53174	6.3 %

VI. 결 론

프로세서 코어가 내장되는 ASIC 설계방식에서는 소프트웨어 디버깅을 위한 전통적인 방법은 더 이상 적합하지 않는다. 현재와 같이 시스템이 복잡해지고 VLSI 설계가 고집적화 되면서 제품의 개발 기간을 줄이기 위해서 코어에 디버거를 내장하게 된다.

본 논문에서는 32bit RISC 마이크로프로세서에 적합한 재사용 가능한 임베디드 디버거를 설계하였다. 설계된 디버거는 real-time 디버깅 기능을 강화시키기 위한 EICEM 모듈을 추가하여 설계하였다. 또한 설계의 재사용을 고려하여 상위 수준에서 구조를 정의하고 설계하였다.

설계된 디버거는 0.25㎛ 오중 금속 CMOS 표준 라이브러리를 이용하여 합성하였고 게이트 개수 3174개, 최고 동작 주파수 385MHz의 결과를 얻었다.

또한 32bit RISC 프로세서에 내장하여 테스트 하였고 6.3%의 면적 오버헤드를 보였다.

참고문헌(또는 Reference)

[1] Gustavo R. Alves and J. M. Martins Ferreira,"From design-for-test to design-for-debug-and-test: analysis of requirements and limitations for 1149.1",VLSI Test Symposium, Proceedings. 17th IEEE , 1999.

[2] Gustavo R. Daniel Aga, Ovidiu Mosuc and J.M.Martins Ferreira, "Debug and Test of Microcontroller Based Applications using the Boundary Scan Test Infrastructure", in Student Forum within the IEEE International Symposium on Indestrial Electronics, IEEE Industial Electronics Society, 1997.

[3] Harry Bleeker, Peter van den Eijnden and Frans de Jong, "Boundary-Scan Test : A Practical Approach", Kluwer Academic Publishers, 1993.

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

[5] IEEE Standard 1149.1-1990 Test Access Port and Boundary Scan Architecture, (ANSI/IEEE), IEEE, Piscataway, NJ, 1990.

[6] Ing-Jer Huang, Hsin-Ming Chen and Chung-Fu Kao, "Reusable Embedded In-Circuit Emulator", Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific , 30 Jan.-2 Feb. 2001.

[7] Jurgen Haufe, Christoph Fritsch, Matthias Gulbins, Volker Luck and Peter Schwarz, "Real-Time Debugging of Digital Integrated Circuits", Proc. Design, Automation and Test in Europe Conference, User Forum, Paris, March 27-30, 2000.

[8] Se-Hwan Lee, Sung-Ho Kwak and Moon Key Lee, "Reusable Design of 32-bit RISC Processor for System On-A Chip", Proceedings of IEEK Summer Conference 2001, Vol.23, Semiconductor Society, pp. 105-105, Yongpyong, Jung 28-30, 2001