SoC 프로그램의 원격디버깅 도구를 위한 USB-JTAG Adapter

구금서[*0], 박명철[**], 하석운[**], 전용기[*], 임채덕[***]
경상대학교 컴퓨터과학과, [***]한국전자통신연구원
[*]{goodman,jun}@race.gsnu.ac.kr, [**]{africa,swha}@nvision.gsnu.ac.kr,
[***]cdlim@etri.re.kr

A USB-JTAG Adapter for Remote Debugging Tool of SoC Programs

Geum-Seo Koo[*0], Myeong-Chul Park[**], Seok-Wun Ha[**], Yong-Kee Jun[*],
Chea-Deok Lim[***]
Dept. of Computer Science, Gyeongsang National University
[*]Electronics and Telecommunication Research Institute

요 약

SoC 프로그램은 타겟 시스템의 자원과 타이밍에 민감하여 실제 타겟 시스템과 동일한 환경에서 디버깅해야 하므로 호스트 시스템과 타겟 시스템간의 원격 접속을 위한 Adapter가 필수적으로 요구된다. 그러나 기존의 디버깅 도구에 사용되는 고가의 Adapter들은 도구에 제한적이기 때문에 다른 도구와의 연동이 불가능하다. 본 논문에서는 산업표준인 JTAG을 기반으로하는 SoC를 원격으로 제어하기 위한 고속의 USB-JTAG Adapter를 개발하여 GDB 기반의 경제적 원격 디버깅 도구를 개발할 수 있음을 보인다.

Ⅰ. 서 론

SoC (System-On-a-Chip)[9, 10] 기술은 MPU, DSP, Memory, 임베디드 소프트웨어 등을 하나의 시스템에 집적해 놓은 IC (Integrated Chip)를 말하며 SoC에서 수행되는 프로그램을 SoC 프로그램이라 한다. 이러한 SoC 프로그램을 안정적으로 개발하기 위해서는 효과적인 디버깅 도구가 필요하다. 그러나 SoC 프로그램은 일반적으로 한정된 자원과 타이밍에 민감하여 타겟 시스템과 동일한 환경에서 디버깅되어야 하므로 환경조성을 위한 어려움이 따른다.

SoC 프로그램을 디버깅하는 전통적인 기법으로 ICE (In Circuit Emulator)[1]와 Logic Analyzer[3]가 있지만 이러한 디버깅 장비를 통한 환경은 고가이며, 타겟 시스템의 자원에 직접적으로 접근하여 시스템 상태를 조사하고 제어하므로 내부 신호나 자원에 대한 접근이 제한되어 있는 SoC 프로그램을 위한 디버깅으로는 부적합하다. 그러므로 SoC 프로세서 자체에서 제공하는 디버그 모듈을 이용하여 디버깅하는 OCD[8] 기법을 이용한다. 그러나 OCD 기법을 이용하기 위해서 호스트 시스템과 타겟 시스템간의 원격 접속을 위한 Adapter가 필수적으로 요구되지만 기존의 Adapter들은 도구에 제한적이며 다른 도구와의 연동이 불가능하다.

본 논문에서는 산업 표준화된 JTAG[2, 5, 12]을 기반으로 원격지의 타겟 시스템에 접근하여 SoC 프로그램을 디버깅할 수 있는 Adapter를 제안한다. 제안한 Adapter는 호스트 시스템의 USB를 사용하여 타겟 시스템의 JTAG에 연결되는 구조를 가진다. 이는 타겟 시스템에 영향을 주지 않고 고속의 USB를 사용하므로 보다 경제적이고 효과적인 원격 디버깅을 가능하게 한다. 또한 Adapter의 내부메모리에 JTAG 핸드쉐이킹을 위한 펌웨어를 내장하므로서 사용자 변경이 용이하여 디버깅 환경의 유연성과 기존 디버깅 도구의 확장성을 가지도록 구현되었다. 구현된 Adapter의 동작 상태를 검정하기 위해서 Intel XScale[6, 7]용 JTAG 디버깅 도구인 EDebugger[13]를 사용해 본 결과, 정상적인 디버깅 환경을 구성할 수 있음을 확인하였다.

2절에서는 연구배경으로 SoC 디버깅 기법을 살펴보고 이를 이용한 기존의 도구와 인터페이싱에 사용되는 USB와 JTAG에 대해 살펴본다. 3절에서는 제안한 USB-JTA

G Adapter의 설계와 기능에 대해서 설명하고 디버깅 수행의 결과를 보인다. 마지막으로 결론 및 향후과제를 제시한다.

Ⅱ. 연구배경

본 절에서는 SoC 프로그램의 원격 디버깅을 위한 기법과 이들 도구에서 이용하는 USB와 JTAG에 대해서 살펴본다.

2.1 SoC 디버깅

SoC를 디버깅하는 전통적인 기법 중 ICE는 실시간 입출력 오류 정정을 위한 기능을 가지며, 하드웨어 모방이 가능하여 실시간 개발에 편리하다. Logic Analyzer는 디지털 시스템의 논리 신호를 포착, 기록하고 나타내는 다수의 채널을 가지는 도구이다. 이러한 기법들은 타겟 시스템의 자원에 직접적으로 접근하여 시스템 상태를 조사하거나 제어하므로 내부 신호나 자원에 대한 접근이 제한되어 있는 SoC 프로그램을 디버깅하기에는 부적합하며 매우 고가의 장비로 비실용적이다. 그러므로 SoC 프로세서 자체에서 제공되는 디버그 모듈을 이용하는 OCD 기법을 이용한다.

OCD (On-Chip Debugging) 기법은 SoC와 같이 여러 모듈이 하나의 칩으로 집적되어 내부 자원에 대한 접근 제한으로 인한 디버깅 문제를 해결하기 위해서, 칩 제조사들은 디버그 커널을 칩 회로의 일부분으로 제공한다. 디버그 커널의 기능이 칩 회로의 일부가 되면 프로세서가 시스템의 다른 부분과 통신을 할 수 없을 지라도 디버깅 도구들은 지속적으로 실행 제어를 전송하여 시스템 자원을 감시할 수 있다. 이러한 OCD 기법은 디버깅을 위해서 원격지에서의 접속이 필수적이므로 Adapter를 사용한다. 다음 절에서는 OCD 기법을 이용한 도구와 USB, JTAG에 대해서 살펴보겠다.

2.2 OCD 도구

OCD 기법을 이용하는 원격디버깅 도구는 호스트 시스템과 타겟 시스템을 연결하기 위한 Adapter가 필수적으로 요구된다. [표1]은 기존의 OCD 기법을 이용하는 Debugger와 이와 연동되는 Adapter를 보이고 있으며 다양한 인터페이싱 방식을 제공하고 있지만 일반적인 OCD 도구는 호스트 시스템의 USB와 타겟 시스템의 JTAG을 사용하고 있다.

USB (Universal Serial Bus)는 컴퓨터와 주변기기를 연결하는 인터페이스 규격화를 목적으로 1994년 Compaq, Intel, Microsoft, Nec등의 7개 업체가 주축이 되어 개발되었다. 현재 USB Spec. 2.0이 나와 있는 상태이며, Spec 1.1에서는 최대 12Mbps를 그리고 Spec. 2.0에서는 최대 480Mbps를 지원한다. USB는 계층적 구조로 구성되며 master/slave protocol을 사용해서 USB 디바이스와 통신한다. 디바이스 측에선 Host측으로의 통신선로를 만들지 못하므로 Host에서만 통신을 시작할 수 있다. USB Host 컨트롤러 드라이버는 대부분 이미 안정화되어 구현된 상태이기에 본 논문에서는 USB 클라이언트 드라이버만을 구현한다.

JTAG (Joint Test Action Group)은 Boundary-Scan으로 더 많이 알려져 있으며 칩 내부에 외부 핀과 일대 일로 연결되어 있는 Boundary Cell을 두어 프로세서가 할 수 있는 모든 동작을 Cell을 통해서 인위적으로 수행 가능하므로 하드웨어 테스트나 연결 상태 등을 체크할 수 있다. 전체적인 인터페이스는 TAP (Test Access Port)의 5개 핀(TDI, TMS, TCK, nTRST, TDO)으로 제어되며 이를 이용하여 프로세서의 상태와는 상관없이 디바이스의 모든 외부 핀을 구동시키거나 값을 읽어 들일 수 있다. 그러므로 JTAG을 이용하여 SoC 프로그램의 디버깅을 위한 하드웨어 인터페이스로 사용이 가능하다. [표1]에 보인 디버깅 도구와 Adapter들은 서로 종속적이기 때문에 다른 도구와의 연동이 불가능하며 이를 활용하기 위한 정보도 부족하다. 그리고 Adapter의 가격도 적게는 $750에서 $1800에 이르는 고가이다. 그러므로 새로운 디버깅 도구 개발시에 제약이 많이 따르게 된다. 본 논문에서는 JTAG을 기반으로 SoC를 원격으로 제어하기 위한 고속의 USB-JTAG Adapter를 개발하여 GDB[11] 기반의 경제적 원격 디버깅 도구를 개발할 수 있음을 보인다.

Adapter	Debugger	Interface	Debugger-Independent	S/W Base
usbSprite Macraigor Systems	GNU Tools	USB-JTAG/BDM	X	Linux
usbDemon Macraigor Systems [Macr04]	OCD Commander /GNU Tools	USB-JTAG/BDM	X	Windows /Linux
mpDemon Macraigor Systems [Macr02]	OCD Commander /GNU Tools	Parallel/Ethernet – JTAG/BDM	X	Windows /Linux/Solaris
HMI-BMD Avocet Systems [Avoc00]	SourceGate II	Parallel-JTAG/BDM	X	Windows /Sun/HP

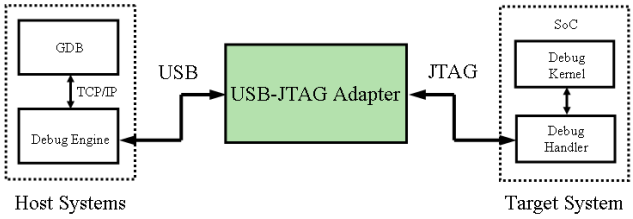
[표1] 디버깅 도구를 위한 Adapter의 비교

Ⅲ. USB-JTAG Adapter

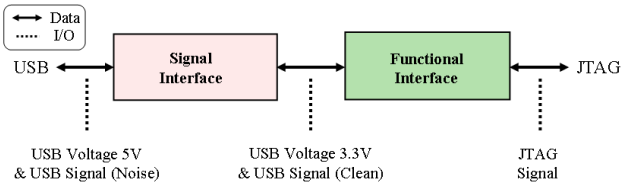
본 절에서는 호스트 시스템의 USB와 타겟 시스템의 JTAG을 인터페이싱하는 USB-JTAG Adapter의 설계와 기능 및 실험 결과를 보인다.

3.1 Adapter의 설계

SoC 프로그램을 원격디버깅하기 위한 호스트 시스템과 타겟 시스템의 전체 시스템 연결 구조를 [그림1]에서 보이고 있다. 이 그림에서 타겟 시스템은 Intel XScale PXA 2XX 프로세서를 탑재하고 있다. [그림2]는 이들간을 인터페이싱하는 USB-JTAG Adapter의 간략한 설계를 보인 것이다. Adapter는 기본적으로 두 통신 포트의 전송속도 차에서 발생하는 데이터 손실을 방지하고, 이질적인 신호간



[그림1] 원격디버깅 구조



[그림2] USB-JTAG Adapter의 전체 구조

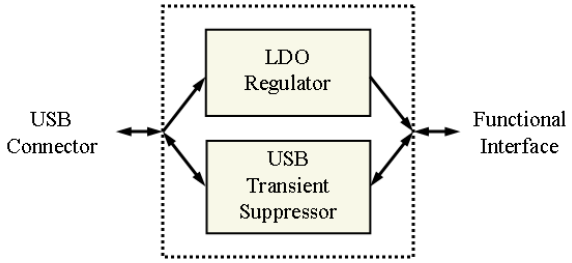
의 핸드쉐이킹과 전압차에 의한 시스템의 오동작을 방지하는 역할을 한다. 각 인터페이스를 자세히 살펴보면 [그림3]의 Signal Interface는 USB 포트에서 출력되는 전압(5V)을 USB 디바이스의 동작전압 및 JTAG과 동일한 전압인 3.3V로 만들어주는 LDO Regulator (LM1117)와 USB Noise를 제거하는 USB Transient Suppressor (SN65220)로 구성되어 있다. [그림4]의 Functional Interface는 Signal Interface에서 받은 USB Signal을 처리하는 USB Interface와 이 Signal을 JTAG Signal로 변환하는 JTAG Interface가 있으며 타겟 시스템의 IDCode 정보를 저장하는 Target Information으로 구성되어 있다.

3.2 Adapter의 기능

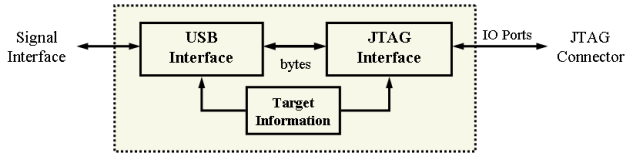
USB-JTAG Adapter의 중심적인 역할을 담당하는 USB 디바이스는 Cypress사의 AN2131QC[4]이다. 이 디바이스는 USB 신호를 JTAG 신호로 즉, SIE (Serial Interface Engine)에서 입출력되는 USB 신호를 JTAG 신호로 적절히 제어하는 것이 필요하며 이를 위해서 내부 메모리의 펌웨어를 이용한다. 펌웨어는 디바이스 내의 8KB 내부 메모리에 존재하며 일반적으로 C 코드로 작성하여 Hex code로 변환 후 호스트의 디바이스 드라이버를 통해서 다운로드 된다. [표2]는 펌웨어가 가져야 할 중요 모듈에 대한 요약이며 사용자의 요구사항에 따라 추가 및 변경이 가능하다. 펌웨어가 핸드쉐이킹을 원활하게 하기 위해서는 TAP (Test Access Port) 컨트롤러의 16가지 테스트 로직 오퍼레이션의 순서를 제어하기 위한 동기적인 유한 상태 머신을 이용한다. TAP은 버스 마스터를 통해 제어되며 버스 마스터는 자동적인 테스트 장비이거나 프로그램이 가능한 로직 디바이스로 TAP을 인터페이싱한다. TAP 컨트롤러는 TCK의 상승 에지나 파워업에 대해서만 상태를 변화시키며 TCK의 상승 에지에서 TMS의 입력 시그널 값은 상태 변화의 순서를 제어한다. TAP 컨트롤러는 자동적으로 파워업시에 초기화되며 5개의 TCK 주기동안 TMS에 1의 시그널을 입력함으로서 초기화될 수 있다. 모든 응용 프로세서의 디지털 신호들은 PWR_EN 핀을 제외하고 바운드리 스캔에 참여한다는 것에 주의해야 한다. 이것은 스캔 오퍼레이션으로 하여금 응용 프로세서의 파워를 오프하는

모듈명	역할
initPorts	USB 칩의 입출력 포트를 초기화 시키기 위한 모듈
jtagReset	JTAG를 초기화 시키기 위한 모듈
cpuReset	타겟의 프로세서를 초기화 시키기 위한 모듈
ireg	JATG의 명령레지스터에 접근하기 위한 모듈
dreg	JTAG의 데이터 레지스터에 접근하기 위한 모듈
jtagReg	JTAG의 레지스터에 접근하기 위한 모듈

[표2] 펌웨어 중요 모듈



[그림3] Signal Interface



[그림4] Functional Interface

것을 금지시킨다.

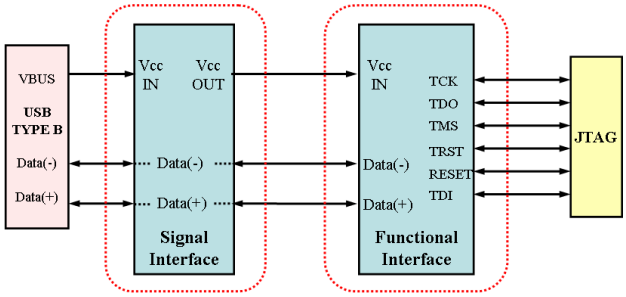
본 연구에 사용된 USB 디바이스는 8051 코어를 사용하므로 프로그램을 작성할 때 8051용 개발툴을 사용할 수 있다. 또한 Re-Numeration을 통해 펌웨어를 소프트웨어적으로 다운로드할 수 있으므로 롬라이터가 별도로 필요없는 장점이 있다. 디바이스의 하드웨어가 기본적인 USB 기능을 내장하고 있으므로 펌웨어는 JTAG과 관련된 변환 모듈로 구성된다. 또한 Full Speed를 지원하고, 충분한 수의 입출력 핀을 가지고 있다. 입출력은 3개의 8비트 포트가 있으며 한 포트의 핀이라도 각각 입출력을 선택하여 설정할 수 있다. 물론 입출력 핀 중 일부는 특수한 목적으로 사용될 수 있으며 이는 펌웨어로 조작이 가능하다.

3.3 Adapter의 구현 및 실험

제안한 Adapter에서 사용되는 디바이스의 입출력 핀(PB)과 타겟 시스템의 JTAG 포트에 전달될 신호의 매핑을 [표3]에서 보이고 있다. 입출력 핀을 제어하기 위해서 4개의 레지스터(PORTBCFG, OUTB, OEB, PINSB)를 두고 있는데, 먼저 전원 인가시에 포트의 기능을 선택할 수 있는 PORTBCFG 레지스터는 디폴트 값으로 0으로 초기화 되어야 한다. 0은 일반적인 입출력용도를 의미하며, 1은 정해진 특수한 용도로 사용하겠다는 의미이다. OEB 레지스터는 입력의 방향을 선택하는 용도인데, 출력 용도로 사용하기 위해 펌웨어에서 각 레지스터 값을 1로 초기화해야 한다. 전원이 칩에 인가되게 되면, EZ-USB 코어는 I2C 포트에 접속하고 있는 EEPROM를 찾게 된다. 만일 EEPROM이 탐지되게 되고 0 번지의 내용이 '0xB0'이면, EZ-USB 코어는 내부 기억 장치로 Vendor ID, Product ID, Device ID을 EEPROM에서 복사한다. 디바이스 코어는 이 때 Get_Descriptor-Device 요구의 일부로서

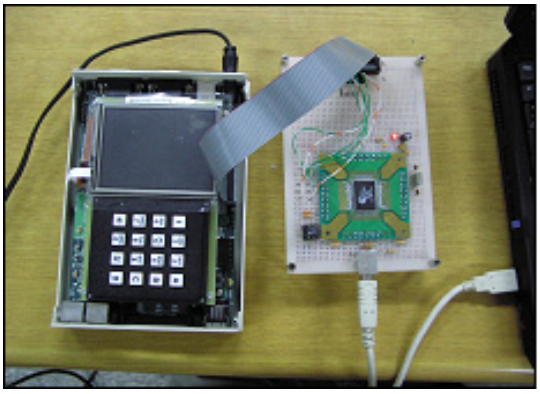
B Port	JTAG	B Port	JTAG
PB0	TCK	PB4	RESET
PB1	TDO	PB5	TDI
PB2	TMS	PB6	미사용
PB3	TRST	PB7	미사용

[표3] USB와 JTAG 핀 맵

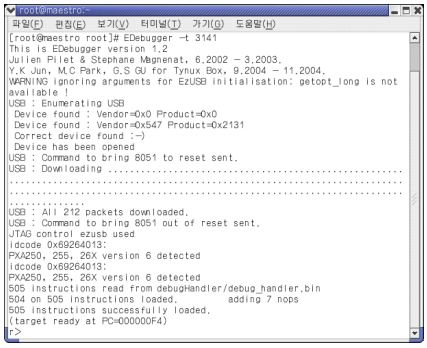


[그림5] USB-JTAG Adapter

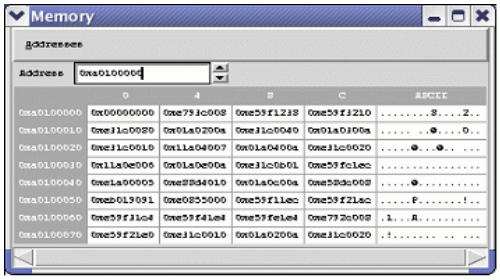
호스트에게 이 바이트를 공급한다. 만일 0 주소의 내용이 '0xB2'일 때는 7 주소부터 펌웨어가 있는 것으로 간주하고 내부 메모리에 다운로드하게 된다. 본 연구에서는 24LC00 칩을 사용하여 해당 칩의 ID만을 제공한다. 호스트 시스템은 이 정보에 일치되는 드라이버가 OS에 의해 로드되게 되고, 드라이버가 EZ-USB의 RAM에 8051를 위한 펌웨어 다운로드하고, 8051 코어가 동작을 개시하게 된다. [그림5]는 설계를 바탕으로 구현된 USB-JTAG Adapter의 전체 구조를 보이고 있다. [그림6]은 구현된 Adapter를 이용하여 타겟 시스템과 호스트 시스템을 연결한 실제 모습이며 타겟 시스템은 (주)팜팜테크 시스템이며 Intel XScale PXA 255 프로세서를 탑재하고 있다. [그림7]은 동작 실험을 위해서 공개 소프트웨어인 EDebugger를 사용하여 실제 타겟 시스템의 IDCODE를 읽어 타겟을 인식하고 펌웨어를 다운로드한 그림이다. [그림8]은 디버깅을 수행하여 타겟 시스템의 메모리 Dump 값을 가져온 그림이다. 이로서 제안한 Adapter가 원활히 동작함을 확인할 수 있다.



[그림6] 타겟 시스템과 연결된 Adapter



[그림7] 동작확인을 위한 화면



[그림7] Memory dump

Ⅳ. 결론 및 향후과제

본 논문에서는 원격디버깅을 위한 경제적인 USB-JTAG Adapter를 제안하였다. 제안한 Adapter는 고가의 ICE 장비를 대신할 수 있는 저가의 경제성있는 장비로서 새로운 디버깅 도구 개발시 활용될 수 있으며 기존의 공개 디버깅 도구와 연동하여 그 동작도 원활함을 확인하였다.

향후 지속적인 연구로 보다 향상된 USB 디바이스의 사용으로 속도를 개선하여 안정화시키고 다양한 도구와 인터페이싱할 수 있는 기능을 추가하여 활용성을 극대화하는 연구가 이루어져야 할 것이다.

참고문헌

[1] Akgul T., P. Kuacharoen, V. J. Mooney and V. K. Madisetti, "A Debugger RTOS for Embedded System," *proceedings of the 27th Euromicro Conference*, September 2001.

[2] ASSET InterTech, Inc., and R. G. Bennetts, *Boundary-Scan Tutorial*, 2000.

[3] Ball, Stuart R., *Debugging Embedded Microprocessor Systems*, Newnes, May 1998.

[4] Cypress Semiconductor Corp, *EZ-USB Technical Reference Manual*, Ver. 1.10, 2002.

[5] IEEE Std. 1149.1, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Computer Society, New York, 1993.

[6] Intel Corp., *Intel PXA250 and PXA210 Application Processors*, 2002.

[7] Intel Corp., *Intel XScale Microarchitecture for the PXA255 Processor*, 2003.

[8] Johnson, M., and N. Puthuff, *Debugging Embedded SoC Systems*, RF Design, February 2002.

[9] Kim, C., H. Kim, and C., Lim, *Technology Trends and Development Strategies on Embedded Software for Ubiquitous Computing Era*, Telecommunications Review, 13(1): 105-116, SKTelecom, 2003.

[10] Oakland, S.F., "Considerations for Implementing IEEE 1149.1 on System-on-a-Chip Integrated Circuits," *Proceedings of the International Test Conference*, pp. 628-637, October 2000.

[11] Stallman, R., R. Pesch, and S. Shebs, et al., *Debugging With GDB*, February 2003.

[12] XJTAG Ltd., *JTAG-A Technical Overview*, March 2003.

[13] **김병철, 강문혜, 전용기, 임채덕, "임베디스 소프트웨어 개발을 위한 JTAG 기반의 디버깅 도구," *한국정보과학회 추계학술발표논문집*, 31(1): 943-945, 한국정보과학회, 2004. 4.**