내장 소프트웨어를 위한 JTAG 어뎁터의 구현

김용수*, 한판암**
*거창전문대학 컴퓨터정보시스템과
**경남대학교 컴퓨터공학부
e-mail:yskim@kc.ac.kr

Implementation of JTAG Adapter for Embedded Software

Yong-Soo Kim*, Pan-Am Han**
*GeoChang Provincial College Dept Of Computer & Information System
**Kyungnam University Division of Computer Engineering

요 약

내장 소프트웨어는 실제 시스템의 자원과 원격지의 시스템의 환경에 민감하므로 실제 시스템과 동일한 환경에서 디버깅해야 한다. 그러나 대부분의 내장 소프트웨어를 탑재하는 실제 시스템은 시스템 상태를 조사하거나 제어하는 것이 제한되어 있는 소프트웨어를 디버깅하는 것은 매우 어렵다. 본 논문에서는 원격지의 USB와 실제 시스템의 JTAG을 기반으로 내장 소프트웨어를 디버깅할 수 있는 어뎁터 제안한다. 본 논문은 실제 시스템내의 내장 소프트웨어를 디버깅할 수 있는 경제적인 인터페이싱 환경을 제공한다.

키워드: 내장 소프트웨어, JTAG, USB

1. 서론

내장 운영체제를 중심으로 내장 소프트웨어[1, 2, 3, 4]의 수요가 가히 폭발적으로 증가되고 있다. 그로 인해서 내장 소프트웨어 또한 점차 복잡하고 다기능적인 동작을 요구하게 된다. 내장 소프트웨어는 해당 시스템의 자원과 타이밍에 민감하여 실제 시스템과 동일한 환경에서 디버깅해야 하지만, 메모리나 입출력 장치 등과 같은 모듈들을 하나의 프로세서에 집적한 SoC(System-On-a-Chip)[5, 6]는 내부 신호나 자원에 대한 접근이 제한되어 디버깅을 더욱 힘들게 한다. 아울러, 시스템의 자원도 충분하지 않기 때문에 요구조건에 적합한 SoC 프로그램을 개발하기 위해서는 자원이 풍부한 원격 시스템에서 디버깅할 수 있는 원격 디버깅 도구의 필요성이 증대되고 있다. 본 논문에서는 산업 표준화된 JTAG(Joint Test Action Group)[7]을 기반으로 원격지의 타켓 시스템에 접근하여 내장 소프트웨어를 디버깅할 수 있는 어뎁터 제안한다. 제안된 어뎁터 원격지 시스템의 USB포트를 사용하여 대상 시스템의 JTAG에 연결되는 구조를 가진다. 이는 대상 시스템에 영향을 주지 않고 고속의 USB포트를 사용하므로서 보다 경제적이고 효과적인 원격 디버깅을 가능하게 한다. 2절에서는 본 연구의 배경으로 내장 소프트웨어 디버깅을 위한 전통적인 디버깅기법과 JTAG과 USB에 대해 살펴보고, 3절에서는 제안된 환경의 핵심적인 부분인 JTAG 어뎁터에 대하여 설명하고 핸드쉐이킹를 위한 어뎁터내의 펌웨어에 대해 설명한다. 마지막으로 결론 및 향후 과제를 제시한다.

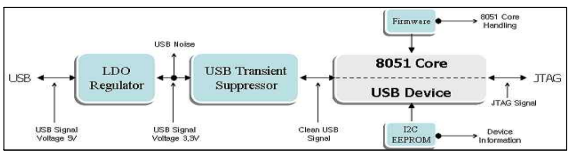
2. 연구배경

본 절에서는 연구배경으로 디버깅 기법과 JTAG에 대해서 살펴보고, 본 환경의 기반이 되는 USB에 대해서 설명한다. 기존의 기법 중 ICE는 하드웨어 모방이 가능하여 실시간 개발에 편리한 수단으로서 실시간 입출력의 오류 정정을 위한 하드웨어 및 소프트웨어 기능을 가진다. logic analyzer는 디지털 시스템의 논리 신호를 포착, 기록하고 나타내는 다수의 채널을 가지는 도구로 2진 형태와 사각 파형 모양의 일련의 펄스 형태로 출력한다. 이러한 기법들은 대상 시스템의 자원에 직접적으로 접근하여 시스템 상태

를 조사하거나 제어하므로 내부 신호나 자원에 대한 접근이 제한되어 있는 SoC 프로그램을 위한 디버깅으로는 부적합하다. 또한, 매우 고가의 장비로 비실용적인 제약사항을 가지고 있다. 그러므로 SoC 프로세서 자체에서 제공되는 디버그 모듈을 이용하여 내장 소프트웨어를 디버깅하는 도구를 제안하기 위해서 JTAG을 기반으로 한다. Boundary-Scan으로 더 많이 알려져 있는 JTAG은 칩 내부에 외부 핀과 일대 일로 연결되어 있는 Boundary Cell을 두어 프로세서가 할 수 있는 모든 동작을 Cell을 통하여 모든 동작을 인위적으로 수행하므로써 하드웨어 테스트나 연결 상태 등을 체크할 수 있는 기능을 가진다. 전체적인 인터페이스는 TAP(Test Access Port)라고 하는 5개의 핀(TDI, TMS, TCK, nTRST, TDO)에 의해서 제어되며, 이를 이용하여 프로세서의 상태와는 상관없이 디바이스의 모든 외부 핀을 구동시키거나 값을 읽어 들일 수 있다. USB는 계층적 구조로 구성되며, master/slave protocol을 사용해서 USB device와 통신한다. 즉, Device 측에선 Host측으로의 통신선로를 만들지 못한다. 따라서 Host에서만 통신의 시작을 할 수 있다. USB Host Controller Driver는 대부분 이미 구현된 상태이고 안정 되었기에, 본 논문에서는 해당되는 어뎁터의 USB Client Driver만을 구현한다.

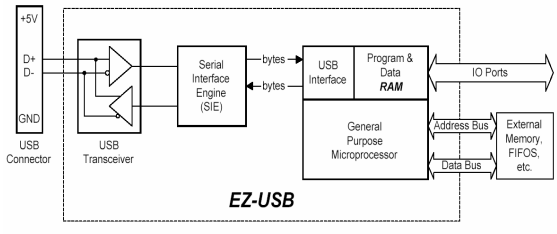
3. JTAG 어뎁터의 설계와 구현

본 절에서는 호스트 시스템의 USB포트와 원격 대상 시스템의 JTAG 인터페이스를 사용한 원격 디버깅용 USB-JTAG 어뎁터의 설계와 구현을 살펴본다. 본 도구는 내장 시스템의 자원 제약성 문제를 해결하기 위해서 호스트 부분과 대상 부분으로 구성된다. [그림 1]은 전체 어뎁터의 구조를 보이고 있다.



[그림 1] 어뎁터의 구조

어뎁터의 중심적인 역할을 담당하는 USB 칩셋은 Cypress 사의 EZ-USB칩[8]을 사용하였다. [그림 2]는 EZ-USB칩의 구조를 보인 그림이다. 그림에서도 알 수 있듯이 USB 신호를 JTAG 신호로 즉, SIE(Serial Interface Engine)에서 입출력되는 USB 신호를 적절히 제어하는 것이 필요하다.



[그림 2] EZ-USB Chipset

이를 위하여 내부메모리에 펌웨어가 요구된다. 어뎁터에는 8KB의 내부메모리가 존재하는데 USB포트로부터 전달된 신호를 대상 시스템의 JTAG 단으로 내보내기 위해서는 이 내부 메모리에 적절한 펌웨어가 다운로드 되어야 한다. 펌웨어는 일반적으로 C 코드로 작성하여 Hex code로 변환 후 호스트의 디바이스 드라이브를 통해 내부 메모리에 다운로드 되게 된다. 펌웨어가 정상적으로 연결간의 핸드쉐이킹을 원활하게 하기 위해서는 TAP(Test Access Port) 컨트롤러의 16가지 테스트 로직 오퍼레이션의 순서를 제어하기 위한 동기적인 유한 상태 머신을 이해해야 한다. TAP은 버스 마스터를 통해 제어된다. 버스 마스터는 자동적인 테스트 장비이거나 프로그램이 가능한 로직 디바이스로 테스트 액세스 포트를 인터페이스한다. TAP 컨트롤러에서 발생하는 각 상태 전이전를 위한 값은(0 or 1) TMS 값을 의미한다. 모든 응용 프로세서의 디지털 신호들은 PWR_EN 핀을 제외하고 바운드리 스캔에 참여한다는 것에 주의해야 한다. 이것은 스캔 오퍼레이션으로 하여금 응용 프로세서의 파워를 오프하는 것을 금지시킨다.

본 연구에 사용된 EZ-USB칩은 8051 코어를 사용하므로 프로그램을 작성할 때 8051용 개발툴을 사용할 수 있다. 또 Re-Numeration 을 통해 펌웨어를 소프트웨어적으로 다운로드 할 수 있으므로 롬라이터가 별도로 필요 없는 장점이 있다. 칩의 하드웨어가 기본적인 USB와 관련된 기본적인 기능은 내장하고 있으므로 펌웨어는 JTAG과 관련된 변환 모듈로 구성된다. 또한, Full Speed 를 지원하고, 충분한 수의 입출력 핀을 갖고 있다. 입출력은 A,B,C 이렇게 3 개의 8비트 포트가 있으며 한 포트 내의 핀이라도 각각 입출력을 선택하여 설정할 수 있다. 물론 입출력 핀 중 일부는 특수한 목적의 핀으로 사용될 수도 있고 이는 펌웨어로 조작이 가능하다. 참고로 아래의 [그림 3]은 펌웨어 세부 모듈중에서 JTAG를 초기화 시키는 모듈을 C 코드로 보이고 있다.

```
void jtagReset(void)
{
   unsigned char loop_count;
   OUTB=JTAG_PWR;
   OUTB=JTAG_PWR | JTAG_CLK;
   for(loop_count = 0; loop_count < 5; loop_count++)
   {
      OUTB=JTAG_PWR | JTAG_TMS;
      OUTB=JTAG_PWR | JTAG_TMS | JTAG_CLK;
   }
   OUTB=JTAG_PWR;
   OUTB=JTAG_PWR | JTAG_CLK;
   OUTB=JTAG_PWR;
   OUTB=JTAG_PWR | JTAG_CLK;
   OUTB=JTAG_PWR;
   jtagRestarted = 1;
}
```

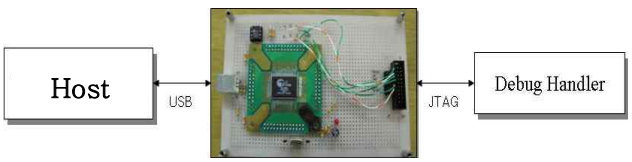
[그림 3] 펌웨어의 C코드 일부

[표 1]은 구현된 어뎁터에서 사용되는 칩셋의 입출력 핀(PB)과 대상 시스템의 JTAG 포트에 전달될 신호의 매핑을 보인 것이다. 입출력 핀을 제어하기 위하여 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. 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. EZ-USB 코어는 이때 Get_Descriptor -Device 요구의 일부로서 호스트에게 이 바이트들를 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.

[표 1] USB 와 JTAG 핀의 연결

B Port	JTAG
PB0	TCK
PB1	TDO
PB2	TMS
PB3	TRST
PB4	RESET
PB5	TDI
PB6	Not used
PB7	Not used

[그림 4]는 구현된 어뎁터의 실제 모습이고 있다.



[그림 4] 실제 어뎁터의 모습

4. 결론 및 향후연구

본 논문에서는 내장 소프트웨어를 JTAG 어뎁터를 제안하였다. 제안된 어뎁터는 저가의 경제성 있는 장비로 이용될 수 있다. 향후 연구로는 본 어뎁터를 이용하여 기존의 디버깅 도구를 이용하여 그 동작을 확인할 수 있는 디버깅 도구를 개발할 것이다. 이러한 도구의 개발로 인해 향후 폭발적으로 증가하게 될 내장 소프트웨어의 요구에 부응할 수 있으며, 높은 경제성을 가질 수 있을 것이다. 또한 지속적인 연구로 속도적인 측면과 활용적인 측면을 극대화 할 수 있는 연구가 이루어 져야 할 것이다.

참고문헌

[1] Heung-Nam Kim, and Chea-deok Lim, *Technology Trends and Development Strategies on Embedded Software for Ubiquitous Computing Era*, Feb. 2003.

[2] Linda J. Moore, Angelica R. Moya, "Non-intrusive debug technique for embedded programming," *Proceedings of the 14th International Symposium on Software Reliability Engineering*, pp. 375-380, Nov. 2003.

[3] Straunstrup, J., Andersen, H.R., Hulgaard, H., Lind-Nielsen, J., Behrmann G., Kristoffersen K., Skou A., Leerberg HH., Theilgaard N.B., "Practical verification of embedded software," *Computer*, Vol. 33(5), pp. 68-75, May 2000.

[4] Alessandro Rubini, and Jonathan Corbert, *Linux Device Drivers Second Edition*, O'Reilly & Associate, Inc., June 2001.

[5] Ziruanm, Y., Dcym, S., Rodgers, M., "Test of future system-on-chips," International Conference on Computer Aided Design, IEEE/ACM, pp. 392-398, 2000.

[6] Kenneth H.Peters, Software Development and Debug for System On-A-Chip," *Embedded Systems Conference*, 1999.

[7] Asset InterTech, Inc., and R. G. *Bennefits, Boundary-Scan Tutorial*, 2000.

[8] Cypress Semiconductor, EZ-USB Technical Reference Manual, 2002.