USB-JTAG Interface 를 이용한 EISC 프로세서 디버거 개발

이호균, 김선욱
고려대학교 전자전기공학과
hokyoon79@korea.ac.kr, seon@korea.ac.kr

Debugging Environment via USB-JTAG Interface for EISC Processor

Hokyoon Lee, Seon Wook Kim
School of Electrical Engineering, Korea University

요 약

많은 개발자들은 프로세서 디버깅을 위해 GDB 를 사용한다. 임베디드 시스템에서 GDB 의 원격 디버깅은 시리얼 통신을 사용한다. 그러나, 시리얼 통신은 속도에 제한이 있으며, 시리얼 포트 마저 점차 사라져 가는 추세이다. 이를 극복하기 위해 많은 임베디드 시스템이 JTAG 인터페이스를 탑재하고 있으며, USB 인터페이스를 사용하여 통신을 한다. 이 논문에서는 EISC 아키텍처 기반의 임베디드 시스템을 디버깅하기 위한 USB-JTAG 인터페이스 개발 방법을 제안하고, GDB 환경에서의 USB 인터페이스 구축 방법과 디버깅 패킷을 분석하기 위한 JTAG 모듈의 개발 방법을 소개한다.

1. 서론

현재 임베디드 시스템의 수요가 확산 되면서, 많은 개발자들은 편리한 디버깅 환경을 필요로 한다. 특히 대부분의 임베디드 시스템이 JTAG [1] 인터페이스를 탑재하면서 JTAG 을 이용한 디버깅 기법이 주목 받고 있다. EISC [2] 아키텍처 기반의 임베디드 시스템 역시 JTAG 인터페이스를 포함하고 있으며, 각종 어플리케이션을 디버깅하기 위해 사용되고 있다.

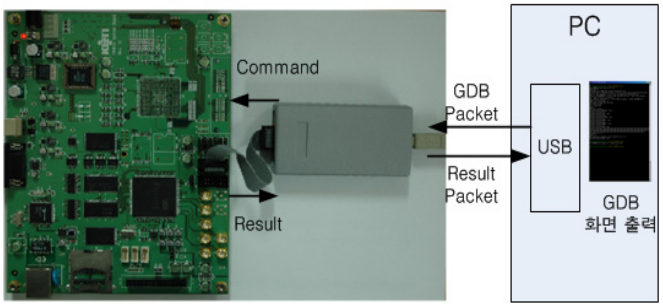
GDB [3]는 오픈 소스로써 폭넓게 사용되고 있는 GNU 디버거이다. 원격 디버깅을 위해 GDB 는 시리얼 통신만을 지원하고 있으며, 이는 개발자들에게 많은 제약을 가한다. EISC 프로세서가 탑재된 임베디드 시스템은 USB-JTAG 인터페이스를 제공하기 때문에 GDB 에서 USB 를 이용한 통신이 가능해야 한다. GDB 는 디버깅 과정에서 끊임없이 레지스터와 메모리의 값을 읽기 때문에, USB 인터페이스를 이용한 데이터의 전송은 개발자들에게 보다 빠른 디버깅 환경을 제공할 것이다.

이 논문은 USB-JTAG 인터페이스의 개발 방법을 제안하고, GDB 와 JTAG 모듈에서 수행하는 패킷의 분석과 흐름에 대하여 설명한다. 2 장에서는 이 논문에서 제안한 디버깅 프레임워크를 논의하고, 3 장에서는 구체적인 동작을 설명한다. 4 장은 실제 USB-JTAG 인터페이스를 사용한 GDB 구동화면을 보여주고, 5 장은 이 논문의 결론이다.

[1] 본 논문은 산업자원부가 지원하는 국가 반도체연구개발사업인 시스템집적 반도체기반기술개발사업(시스템 IC2010)을 통해 개발된 결과임을 밝힙니다.

2. Debugging Framework

그림 1 은 USB-JTAG 인터페이스를 이용한 전체 디버깅 환경을 나타내고 있다. GDB 프로그램은 사용자가 입력한 명령을 패킷 형태로 변환하여 JTAG 모듈로 USB 인터페이스를 통해 전송한다. JTAG 모듈은 GDB 로 부터 전송 받은 패킷을 분석하여 EISC 타겟 보드로 그에 해당하는 명령을 JTAG 을 통해 전송한다. 디버깅 모드로 부팅된 타겟 보드는 JTAG 모듈의 명령을 기다리고 있는 상태이며, 명령이 전달 되면 그 명령을 수행한다. 또한, 타겟 보드는 실행 결과를 JTAG 을 통해 JTAG 모듈로 전송하고, JTAG 모듈은 결과 데이터를 GDB 가 분석 가능한 형태의 패킷으로 변경 후 USB 를 통해 전송한다. 마지막으로, GDB 는 패킷 분석 후 사용자가 내린 명령에 대한 결과를 화면으로 출력한다.



(그림 1) EISC 타겟 보드, JTAG 모듈 및 디버깅 명령의 흐름

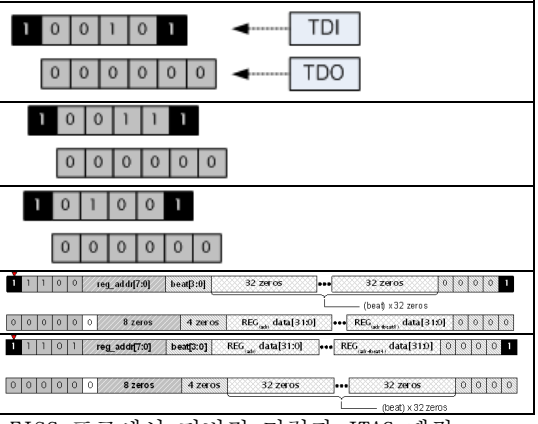
3. USB-JTAG Interface

사용자가 입력한 디버깅 명령은 GDB 내에서 패킷 형태로 재구성 되며, 이 패킷을 정해진 통신방법을 통해 외부로 전송한다. USB-JTAG 인터페이스를 이용하기 위해, GDB 는 USB 디바이스를 사용한다. GDB 에서는 USB I/O 함수가 구현되어 있는 DLL(Dynamic Link Library)을 통신 방법을 설정할 때 로딩하여 USB 읽기/쓰기를 할 수 있게 된다. 표 1 은 USB I/O DLL 에 구현되어 있는 함수들을 설명하고 있다. GDB 의 모든 패킷은 이 USB 읽기/쓰기 함수를 통해 송/수신 된다. 명령에 대한 결과로써 수신된 패킷은 결과값만을 선택하여 화면에 출력한다.

USB Function	Description
OpenDevice	USB 디바이스 사용을 시작한다.
Write	USB 를 통해 데이터를 전송한다.
Read	USB 를 통해 데이터를 수신한다.
Close	USB 디바이스를 중지한다.

<표 1> USB I/O 함수

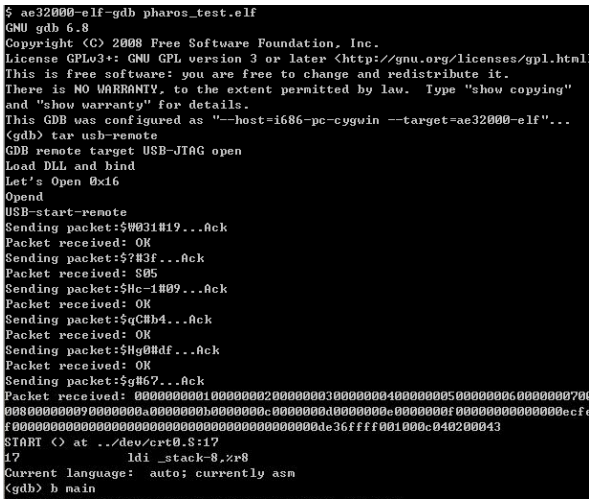
JTAG 모듈은 USB 를 통해 전송 받은 명령을 타겟 보드가 실행 할 수 있는 패킷으로 변환한다. 표 2 는 EISC 프로세서가 수행 할 수 있는 디버깅 명령과 그에 따른 JTAG 패킷 시퀀스를 보여준다. TDI 는 JTAG 의 입력이며, TDO 는 출력이다. GDB 패킷은 명령의 종류에 따라 표 2 의 패킷 시퀀스로 변환 되며, JTAG 을 통해 타겟 보드로 그 명령을 전달한다. Register Read 의 경우 TDI 패킷은 11100 / 레지스터 번호 / 레지스터 개수 / 레지스터 개수 X 32-zeros / 00001 로 구성되며, TDO 의 경우 18-zeros / 레지스터의 데이터 / 0000 으로 구성된다. 타겟 보드의 실행 결과는 표 2 와 같은 패킷 시퀀스의 형태로 JTAG 모듈로 수신되며, 이어서 결과 데이터만 추출 및 GDB 패킷으로 변환한 후 USB 디바이스를 통해 결과 데이터를 전송한다.

Processor Command	JTAG Module Packet Sequence
Break	
Resume	
Step	
Register Read	
Register Write	

<표 2> EISC 프로세서 디버깅 명령과 JTAG 패킷

타겟 보드는 Break 상태에서 JTAG 을 통해 명령을 전달 받는다. 그리고, 각각의 명령을 수행 후 결과를 JTAG 모듈로 전송한다. Breakpoint 또는 watchpoint 가 설정되어 있는 경우, 타겟 보드는 breakpoint 의 주소와 PC 값이 같을 때 또는 watchpoint 의 데이터가 변경되었을 때, break 상태가 되며 JTAG 모듈로 인터럽트 신호를 전송한다. GDB 는 인터럽트 신호를 받은 후, 현재 상태의 PC 값을 토대로, 디버깅 중인 소스의 위치를 가리키게 된다.

4. 구현 결과



```
$ ae32000-elf-gdb pharos_test.elf
GNU gdb 6.8
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --target=ae32000-elf"...
(gdb) tar usb-remote
GDB remote target USB-JTAG open
Load DLL and bind
Let's Open 0x16
Opend
USB-start-remote
Sending packet:$W031#19...Ack
Packet received: OK
Sending packet:$?#3f...Ack
Packet received: S05
Sending packet:$Hc-1#09...Ack
Packet received: OK
Sending packet:$qC#b4...Ack
Packet received: OK
Sending packet:$Hg0#df...Ack
Packet received: OK
Sending packet:$g#67...Ack
Packet received: 0000000001000000020000000300000004000000050000000600000070
0080000000090000000a0000000b0000000c0000000d0000000e0000000f000000000000000ecf
f000000000000000000000000000000000000000000de36ffff001000c040200043
START () at ../dev/crt0.S:17
17              ldi _stack-8,%r8
Current language:  auto; currently asm
(gdb) b main
```

(그림 2) GDB 구동 화면

그림 2 는 Cygwin 환경에서 GDB 를 실행 후 EISC 타겟 시스템을 디버깅하는 화면이다. 우선, 통신 방법 설정을 usb-remote 로 설정한 것을 볼 수 있다. GDB 는 USB I/O DLL 을 로딩하고, 연결을 확립(connection establish)한 후 사용자의 디버깅 명령을 기다린다. EISC 타겟 보드는 JTAG 디버깅 모드로 부팅되어 있으며, JTAG 을 통해 전송 받은 명령을 수행 할 준비가 되어 있다.

5. 결론

USB-JTAG 인터페이스를 사용한 디버깅 환경은 시리얼 통신을 사용하는 방법보다 더 빠르고 쉽게 이용 가능하다. 또한, JTAG 디버깅 모듈이 장착된 임베디드 시스템에서는 USB-JTAG 인터페이스를 이용한 디버깅 환경이 요구된다. GDB 에서 USB 디바이스와, 패킷을 분석하는 JTAG 모듈을 사용한다면, 임베디드 시스템 개발자에게 보다 쉬운 디버깅 환경을 제공할 수 있을 것이다.

참고문헌

[1] IEEE Std 1149.1-1990 IEEE Standard Test Access Port and Boundary-Scan Architecture –Description

[2] EISC (Extendable Instruction Set Computer). http://www.adc.co.kr

[3] GDB. http://www. gnu.org/software/gdb