병렬 신호처리 시스템 개발에 관한 연구

오형근*, 김욱, 정수운, 이동호, 박성주, 전창호
한양대학교 전자컴퓨터 공학부
hkoh@mslab.hanyang.ac.kr, wkim@para1.hanyang.ac.kr swjung@image.hanyang.ac.kr,
dhlee@image.hanyang.ac.kr, parksj@mslab.hanyang.ac.kr, chjeon@para1.hanyang.ac.kr

Development of a Parallel DSP System

Hyungkeun Oh*, Wook Kim, Suwoon Jung, Dongho Lee, Sungju Park, Changho Jeon
School of Electrical and Computer Engineering, Hanyang University

요 약

방대한 양의 실시간 연산을 요구하는 영상 신호처리, 소나, 레이다와 같은 시스템에서는 성능을 최대화하기 위해 병렬 신호처리 시스템의 사용이 불가피하다. 본 논문은 2개의 DSP칩(TMS320C6701)을 사용하여 설계 및 구현한 병렬 신호처리보드의 구성과 이를 구동시키기 위한 소프트웨어 구성체계를 제시한다.

1. 서 론

최근에는 고속 신호처리 시스템을 구현하는데 DSP칩을 다중으로 연결하는 병렬 시스템 구조가 많이 채택되고 있다. 그 이유는 DSP소자들의 개별적인 연산능력이 우수하긴 해도 단일 프로세서구조로 높은 성능요구조건을 충족시키기 어렵기 때문이다. 특히 영상 신호처리, 소나, 레이더와 같은 신호처리분야에서는 GFLOPS (Giga Floating Point Operation Per Second)수준의 성능을 요구하기 때문에 필연적으로 병렬 신호처리가 필수적이며, 이에 따라 현재까지 병렬 신호처리에 관한 연구와 함께 병렬 신호처리 시스템이 많이 개발되었다. 본 논문에서는 현재 개발 중인 병렬 신호처리보드의 구성과 각 블록별 기능을 2장에서 설명하고, 3장에서는 소프트웨어 구성에 대하여 설명하며, 4장에서는 본 논문의 결론 및 향후 연구계획을 기술한다.

2. 병렬 신호처리보드의 구성

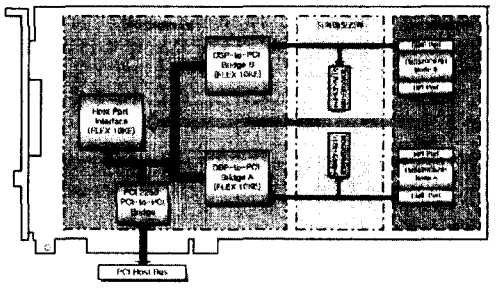

그림 1. 병렬 신호처리보드 전체 블록도

<그림 1>은 구현된 병렬 신호처리보드의 전체 하드웨어 구성을 나타내는 블록도이다. 이 보드는 PCI기반의 펜티엄 PC를 호스트로 하여 동작할 것이다. 병렬 신호처리보드는 크게 DSP신호처리부, 지역메모리부, PCI인터페이스부로 구성되어 있다. DSP를 통하여 고속으로 처리된 신호는 DSP의 EMIF포트를 통해 지역메모리인 SDRAM에 임시 저장되며, PCI버스를 통해 저장된 데이터를 호스트로 보낸다. 중요부분의 자세한 설명은 다음과 같다.

2.1 신호처리부(TMS320C6701)

병렬 신호처리보드의 연산소자로 TI사의 TMS320C6701은 고성능의 부동소숫점 DSP칩으로서 167MHz의 클럭속도와 1GFLOPS의 산술연산능력을 가지고 있다.[1,2] 그리고 1Mbit의 프로세서 내부메모리와 초당 400Mbyte의 대역폭을 가지는 32비트 외부 메모리 인터페이스를 가지고 있다. EMIF포트를 통하여 직접적으로 지역메모리(SDRAM)를 액세스할 수 있으며, DSP가 마스터일 때 다른 DSP칩의 지역메모리를 액세스할 수 있다. 또한 HPI(Host Port Interface)포트를 통하여 외부의 호스트가 DSP에 프로그램을 다운로드 할 수 있다.

2.2 지역메모리부(SDRAM)

지역메모리인 SDRAM은 DSP에서 고속으로 처리된 데이터를 임시 저장하기 위해 사용된다.[3] 평상시에는 DSP가 SDRAM을 직접적으로 제어하다가 외부에서 SDRAM을 사용하겠다는 홀드 요구신호가 들어오면 DSP는 지역메모리와의 연결을 끊고 이때부터 DSP-to-PCI제어기가 지역메모리를 제어한다. 지역메모리는 DSP가 제어할 때 모두 DSP의 외부클럭에 의해 동작되며, DSP-to-PCI제어기가 제어할때는 DSP-to-PCI 제어기에서 나오는 외부클럭에 의해 동작된다.

2.3 PCI인터페이스부(DSP-to-PCI 제어기)

DSP-to-PCI제어기는 크게 마스터모드, 타겟모드, 그리고, HPI(Host Port Interface)모드로 구성되며, CPLD (Complex Programmable Logic Device)칩으로 구현하였다.[4-9]

3. 소프트웨어 구성

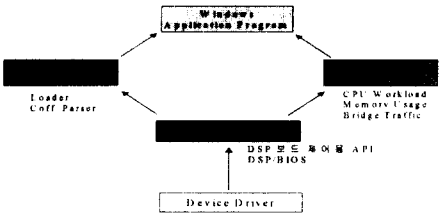

그림 2. 소프트웨어 상호 관계도

신호처리 보드가 동작하기 위해서는 시스템 및 응용 소프트웨어를 갖추어야 한다.

<그림 2>는 신호처리보드를 구동시키기 위한 소프트웨어 상호관계를 나타낸다.

3.1 Device Driver 함수

Device Driver는 하드웨어와 운영체제 및 응용 프로그램간의 연결 통로가 되는 프로그램이며, 여러 개의 하드웨어 구성요소가 운영체제 아래에서 제대로 동작하기 위해서는 필수적인 요소이다.[10,11,12] Device Driver 함수의 종류는 크게 DSP보드 엑세스, 메모리 엑세스로 구분할 수 있으며, 다음 16개의 기본함수를 제공한다.

3.2 DSP API 함수

Windows 응용 프로그램을 개발하기 위해서는 Device Driver와 응용 프로그램 간 인터페이스 기능을 하는 소프트웨어 컴포넌트를 제공해야 한다. 이것은 DSP보드 제어용 컴포넌트로 DSP 보드의 응용 프로그램을 구성하는 모든 부분에 이용되는데 우리는 DSP API 함수를 두 부분으로 나누어 구현하였다.

- OpenDevice
- DeviceDriverAPIInit
- bSetLocalSpace
- CloseDevice
- WriteLocalSpace
- dGetLocalSpace
- DDriverConfig
- ReadLocalSpace
- wGetLocalSpace
- CreateInterrupt
- dSetLocalSpace
- bGetLocalSpace
- DDriverGetVersion
- MakeInterrupt
- DeleteInterrupt
- wSetLocalSpace

3.2.1 DSP보드 제어용 API 함수

표 1 : DSP 보드 제어용 Component

DSP 보드 핸들 모듈	
DSP_open	· DSP 보드에 대한 Device Driver를 호스트와 연결 · DSP 보드를 제어하기 위하여 반환된 핸들 사용
DSP_close	· DSP 보드에 대한 Device Driver의 연결 단절
DSP_reset	· DSP 보드를 초기화하고 상태를 Boot 모드로 변환
메모리 제어 모듈	
DSP_read	· PCI 버스를 통해 DSP 보드의 메모리 읽기 · 정해진 시간 내 데이터 이동 필요
DSP_write	· PCI 버스를 통해 DSP 보드의 메모리 쓰기 · 정해진 시간 내 데이터 이동 필요
HPI 운용 모듈	
DSP_hpi_open	· DSP 보드의 HPI 포트를 PCI 버스와 연결 · 연결 후, DSP 보드와의 읽기, 쓰기 기능 지원
DSP_hpi_close	· DSP 보드의 HPI 포트와 PCI 버스의 연결 단절
DSP_hpi_read	· HPI 포트를 이용하여 DSP 보드의 메모리 읽기
DSP_hpi_write	· HPI 포트를 이용하여 DSP 보드의 메모리 쓰기

<표 1>은 DSP보드 제어용 API 함수의 모듈의 구성과 각각의 기능을 보여준다.[2]

DSP보드 접근 모듈은 DSP보드에 대한 직접적인 핸들을 가지며, 핸들을 이용하여 응용 프로그램과 인터페이스를 적용한다. 메모리 제어 모듈은 Host와 DSP보드의 메모리를 상호 Mapping하는 역할을 하고 HPI 운용 모듈은 DSP보드의 상태플래그를 지정하며, DSP가 처리해야 할 데이터를 전송하는 역할을 한다.

3.2.2 DSP/BIOS API함수

<표 2>는 DSP/BIOS API함수의 구성과 기능이다.[13,14]

표 2 : DSP/BIOS API Component

Clock 모듈	
CLK_gethtime	높은 해상도의 시간을 얻음
LOG 모듈	
LOG_printf	형식화된 메시지를 메시지 Log에 추가
MEM 모듈	
MEM_alloc	Memory Section으로부터 Buffer Size 만큼 할당
MEM_free	Memory Section에서 Buffer Size 만큼 해제
MEM_stat	Memory Section의 상태를 반환
TSK 모듈	
TSK_sleep	현재 작업의 실행을 지연시킴
TSK_self	현재 실행되고 있는 작업으로 핸들러를 전환
RTDX 모듈	
RTDX_CreateInputChannel	Input에 대한 RTDX 데이터 채널을 선언하고 초기화
RTDX_CreateOutputChannel	Output에 대한 RTDX 데이터 채널을 선언하고 초기화
RTDX_enableInput	Input에 대한 데이터 채널을 가능/불가능 하게함
RTDX_enableOutput()	Output에 대한 데이터 채널을 가능/불가능 하게함
RTDX_read()	Input Channel로부터 데이터를 읽음
RTDX_write()	Output Channel로 데이터를 씀

Clock모듈은 System Clock 을 핸들링하고, Log 모듈은 실시간에 Event 를 캡쳐, MEM 모듈은 메모리 Section 을 핸들링, TSK 모듈은 Multitasking을 제공한다. 특히, RTDX 모듈은 실시간에 데이터 전송을 가능하게 한다.

3.3 DSP 보드 운용 프로그램

DSP보드에서 신호처리프로그램의 실행을 위한 COFF파일(Common Object File Format)을 분석하고, 분석된 결과를 바탕으로 COFF파일을 DSP보드에 다운로드하는 Loader를 제작하였다.[2] 아래그림은 COFF파일을 분석하고 DSP보드에 다운로드시킨 후 메모리내용을 확인하는 화면이다.

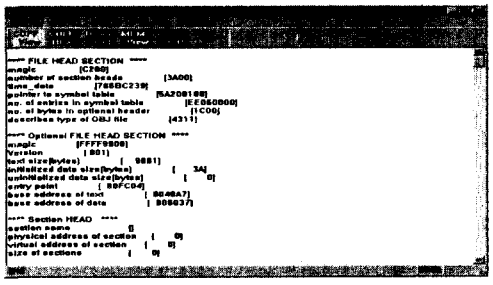

그림 3. COFF 파일 분석

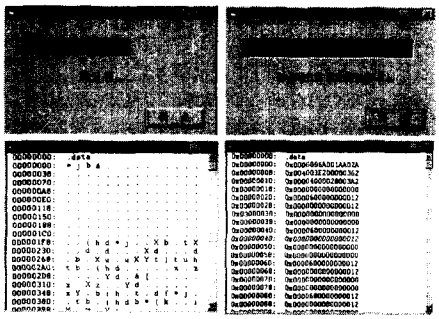

그림 4. 다운로드 과정과 메모리 내용확인

3.4 성능 모니터 프로그램

성능 모니터링의 목적은 병렬 신호처리 보드의 자원 활용도에 대한 이해를 향상시키는 것으로 이를 통하여 효율적으로 시스템의 자원을 관리할 수 있다.[13,14]

본 연구에서는 DSP Workload, Memory Usage, Bridge Traffic을 모니터링 하였는데 다음 그림은 각각을 모니터링한 화면이다.

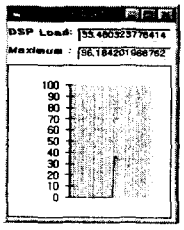

그림 3. DSP Workload

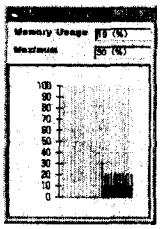

그림 4. Memory Usage

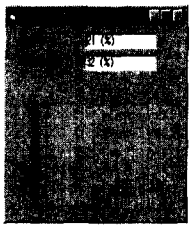

그림 5. Bridge Traffic

3.5 GUI 기반의 Demo Window

COFF Parser 와 Loader 및 성능 모니터 프로그램을 통합한 Demo Window를 구현하였으며 많이 쓰이는 기능은 아이콘화 하여 도구모음으로 모아놓았다. 이를 통하여 사용자는 입출력 데이터를 용이하게 분석할 수 있고 소나 신호처리 보드에 대한 이해를 높일 수 있다. <그림 8>은 구현된 Demo Window 화면이다.

그림 8. Demo Window

4. 결 론

본 논문에서는 소나신호를 고속으로 처리하기위해 2개의 DSP칩(TMS320C6701)을 사용하는 병렬 신호처리보드를 설계 및 구현하고, 이를 구동시키기 위한 소프트웨어를 제작하는 방법을 제시하였다. 특히 DSP-to-PCI 제어기는 하드웨어 언어인 VHDL로 설계하고, Synopsys와 Altera MaxplusII상에서 시뮬레이션을 통하여 검증하였으며, 최종적으로 Altera의 CPLD칩을 사용하여 구현하였다.

또 신호처리보드 동작을 위해 필요한 소프트웨어 상호관계도 와 DSP API 및 신호처리 응용 프로그램에 대해 설명하였다.

향후 계획으로는 최종적으로 개발된 병렬신호처리 보드를 소프트웨어와 통합하여 검증하고 소나 신호처리 알고리즘을 적용한 응용 프로그램을 개발하여 시연할 것이다.

5. 참고 문헌

[1] Texas Instruments TMS320C6701 Peripherals Reference Guide

[2] Texas Instruments TMS320C6701 Evaulation Module Technical Reference

[3] Samsung K4S641632D 1M*16bit*4bank CMOS SDRAM Datasheet

[4] Texas Instruments PCI2250 PCI-to-PCI Bridge User Guide

[5] PLX Technology PCI9054 Data Book

[6] PLX Technology PCI9050 Data Book

[7] ALTERA FLEX10K50, FLEX6000 Data Book

[8] Spectrum Signal "Datona" Dual C6201 PCI Board Technical Reference

[9] PCI System Architecture Fourth Edition, MindShare, Inc.

[10] Walter Oney, "Programming the Microsoft Windows Driver Model", Microsoft Press, USA, 1999

[11] Chris Cant, "Writing Windows WDM Device Drivers", R&D Books, USA, 1999

[12] Karen Hazzah, "Writing Windows VxDs and Device Driver", R&D Books, USA, 1997

[13] Texas Instruments TMS320C6000 DSP/BIOS User's Guide

[14] Texas Instruments TMS320C6000 DSP/BIOS Application Programming Interface(API) Reference Guide