DSP 기반 멀티미디어 스트리밍 플랫폼

*이완규, 홍근표, 장태규
중앙대학교

A DSP Based Multimedia Streaming Platform

Lee Wan Gyu
Chung-Ang University

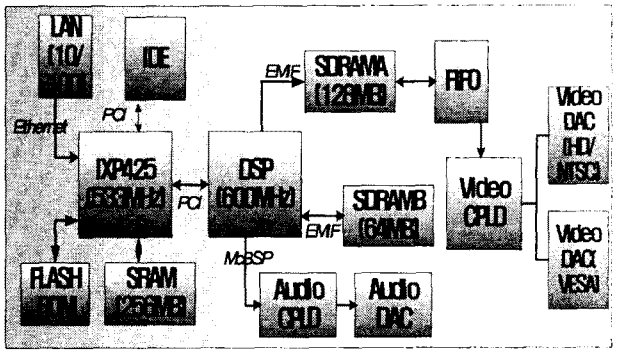
Abstract - 본 논문에서는 고속의 멀티미디어 Streaming data를 처리하기 위한 DSP 기반의 플랫폼을 설계하였다. 개발한 하드웨어는 오디오/비디오 데이터를 처리하는데 있어서 유연성과 확장성을 부여하기위하여 DSP를 사용하였으며 기존의 DSP플랫폼과 다르게 PC에 의존하지않는 독립적인 H/W를 만들기 위하여 ARM Processor를 사용하였다. 이는 저해상도, 저비트율의 오디오/비디오가 분리된 형태의 플랫폼이아닌 DSP기반의 독립적인 유연 플랫폼 구조를 통하여 고속, 고해상도를 가능하게 한다. LAN을 통하여 전송받은 Streaming data는 PCI Interface를 이용하여 DSP로 전달되며 디코딩되어 출력된다.

1. 서 론

최근 네트워크 속도가 고속화됨에 따라서 네트워크환경에서 멀티미디어 데이터의 실시간 Streaming 서비스가 큰 관심을 끌고있다. 기존의 PC 환경에서만 가능하던 서비스를 좀더 넓은 범위로 확대하기위해서는 새로운 하드웨어 플랫폼의 개발이 필수불가결할것이다..

이제까지 실시간 Streaming 서비스를 하기위해서는 PC에 종속적인 환경에 국한되어있었다. 본 논문에서는 독립적인 하드웨어 플랫폼 개발을 위하여 ARM 을 Main Processor로 사용하였으며 스트리밍 Data는 DSP에서 처리하도록 H/W를 설계하였다. 설계된 H/W는 비디오와 오디오의 입/출력이 가능하다. 실제 실시간 스트리밍 서비스를 위해서는 복잡한 연산이 필요할 뿐만아니라 데이터 Format의 다양성 때문에 이에따른 Decoding과정도 유연한 구조를 가지고 있어야 한다. Data 처리에 있어서 위와 같은 유연성과 확장성을 가져오기 위하여 DSP를 사용하여 H/W를 설계하였다.

2. 본 론



<그림 1> ARM Processor based DSP Platform 기능별 구성도

실시간 스트리밍 서비스를 위해서 IXP425는 LAN을 통하여 전송받은 데이터를 일정한 속도(33MHz/66MHz)로 PCI Interface를 통하여 DSP에 바로 전달한다. 또한 스트리밍 받은 데이터를 바로 처리하지 않고 하드디스크에 저장할수도 있다. DSP는 전송받은 데이터를 오디오와 비디오 데이터로 Demuxing한후 외부메모리에 저장한다. 저장된 데이터를 Decoding하여 실시간으로 모니터 뿐만이 아니라 TV로도 이미지를 출력해줄수 있는 구조로 설계하였다. DSP가 전송받은 데이터를 실시간으로 Decoding하기위해서는 외부메모리와 외부 인터페이스로 디스플레이를 담당하는 부분에 있어서는 고속의 데이터를 처리할수 있는 구조로 설계하였다.

H/W의 독립성을 위해서는 OS(Operating System)가 포팅되어있어야 한다.FLASH ROM과 256MB의 SDRAM을 사용하였다.

본 논문에서 구현한 하드웨어는 One-Board System이다. 이해를 돕기위하여 Main Processor부분과 데이터 처리부분을 구분지어 H/W 구조를 보여주겠다.

2.1 IXP425(XScale) 의 H/W 구조

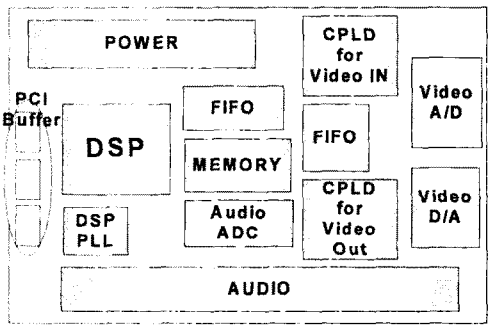
메인 프로세서의 주된 역할은 네트워크를 통해서 원하는 스트리밍 데이터를 받은후 PCI 인터페이스를 통해서 DSP로 전송해주는 것이다. 이 2가지 기능에 초점을 두어 메인 프로세서는 Intel의 Network Processor인 IXP425를 사용하였다. 설계에 있어서 한가지 더 고려할 점은 네트워크 환경이다. 우리가 사용하고 있는 네트워크 속도는 항상 일정한 것이 아니라 환경에 따라서 변화한다.따라서 PCI BUS를 통해서 데이터를 전송할 때 네트워크 속도 변화에 독립적으로 데이터를 전송할수 있도록 네트워크를 통해 받은 데이터를 하드 디스크에 저장한 후에 일정한 속도로 DSP로 전송할 수 있도록 설계하였다.

독립적인 H/W의 동작을 위해서는 OS가 Porting되어야한다. 이를 위해서 16MB의 FLASH ROM과 256MB의 SDRAM을 사용하였다.

IXP425의 동작 실험은 Embedded Linux를 Porting하여 검증하였다. 부트로더는 RedBoot을 사용하였으며 kernel은 2.4.19이고 RamDisk 이미지 까지 Porting하였다. OS에는 LAN 드라이버까지 포함시켜서 실제 LAN을 이용하여 DATA를 받을수 있다.

2.2 DSP(TMS320C6416) 의 H/W 구조

고속의 데이터 처리를 위해 600MHz의 동작속도를 갖는 TMS320C6416을 사용하였으며 IXP와는 PCI 인터페이스를 통하여 Data를 전송받는 구조를 갖는다. DSP는 별도의 외부메모리(EMIFA, EMIFB)를 할당할수 있다. EMIFA는 동작속도가166 MHz인 메모리 128MByte를 사용하고 EMIFB는 100MHz의 64MBytes를 사용하였다.



<그림 2> DSP(TMS320C6416) H/W 구성도

DSP의 동작실험은 MPEG-2 video MP@ML 과 48kHz 5.1ch AC3(5.1ch 디코딩 후 2ch로 downmix)가 포함된 MPEG-2 program stream coyote.mov file을 호스트 PC에서 본 논문에서 제안한 DSP 플랫폼으로 스트리밍해서 테스트한 결과 데이터의 오버런과 언더런 margin 사이를 선회하고 2회 정도의 버퍼 언더런/오버런이 발생하였다. 비디오의 경우 I-FRAME을 바로 찾아서 출력하기 때문에 크게 깨지는 현상은 보이지 않는다. 오디오의 경우도 비디오와 같은 시점에 같은 현상을 보이므로 오디오와 비디오간의 싱크는 깨지지 않는다.

2.2.1 비디오/오디오의 디코딩을 위한 고속 메모리/FIFO구조

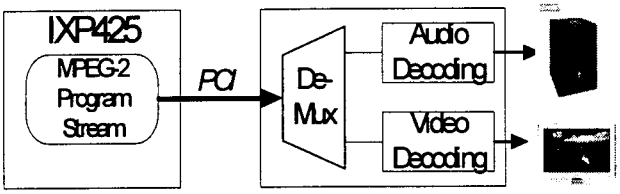
DSP의 EDMA 를 통한 블록단위의 전송을 위해 고속, 고용량의 FIFO를 사용하여 고용량, 고화질의 비디오 출력이 가능하면서 오디오의 출력과 오디오/비디오의 디코딩을 동시에 할 수 있다. 이는 디코딩과 메모리를 ACCESS하는 시간이 최대한 적어야 끊기지 않는 비디오와 오디오의 출력을 얻을 수 있다. 평균 29frame/sec의 화면을 얻기 위해서는 dmuxing 및 오디오/비디오 프레임 디코딩이 35ms 이내에 이루어져야 한다. 이를 위해서는 고속 메모리 구조가 필수적이다.

DSP가 데이터를 처리하는데 있어서 차지하는 최대 메모리 동작속도에 보장하기 위하여 최대 동작속도가 166MHz인 FIFO를 사용하여 고속 데이터 처리를 가능하게 설계하였다. 또한 실제 Video D /A Converter에 데이터를 전달해주는 CPLD에서의 Delay Time을 줄이기위해 Gate delay가 7~10ns인 칩을 사용하였다.

2.3 멀티미디어 스트리밍

2.3.1 멀티미디어 스트리밍 구조 및 디코딩 구조

멀티미디어 스트리밍 구조는 IXP425가 PCI를 통해서 오디오/비디오 데이터가 포함된 MPEG-2 Program Stream(PS)를 DSP 플랫폼에 전송한다. 이를 구현하기 위해서 API 기반의 전송 프로그램을 별도로 개발하였다. 아래 그림 3은 전체적인 스트리밍 및 DSP 디코딩 과정을 나타내었다.



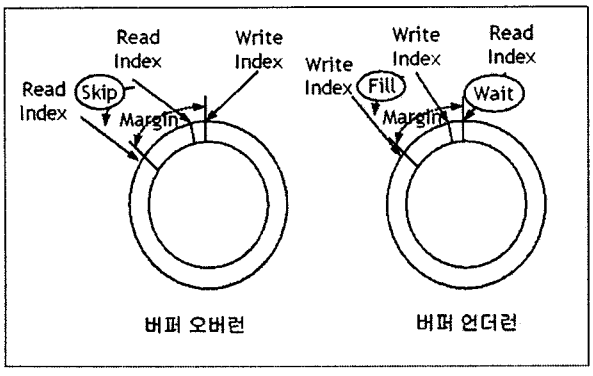
<그림 3> 멀티미디어 스트리밍 DSP 플랫폼 구조

IXP425에서는 쓰레드를 발생시켜 일정한 시간(200ms)마다 DSP 플랫폼의 외부 메모리(SDRAMA)에 접근하여 일정한 량의 PS 데이터를 쓴다. 이 때 DSP의 EDMA(Enhanced Direct Memory Access)를 이용하기 때문에 DSP 프로그램 실행에는 영향을 주지 않는다. PCI를 통해서 PS 데이터를 쓴 후에 인터럽트가 발생하고 Demuxer 루틴을 호출하여 가장 오래된 PS을 읽어 오디오와 비디오의 데이터를 각각의 버퍼에 저장한다. 비디오 디코딩은 timer interrupt를 사용한다. 40ms마다 timer interrupt가 걸리면 비디오 버퍼에서 데이터를 읽어 1 frame 디코딩을 수행하고. 디코딩된 결과인 RGB 데이터를 CPLD로 전송한다. 오디오 디코딩도 timer interrupt 를 사용한다. interrupt가 걸리면 디코딩된 raw 데이터를 저장하는 버퍼 포인터를 보고 판단하여, 언더플로우 또는 오버플로우가 발생하지 않았다면 1 frame 디코딩을 한다. 그리고 EDMA(MCBSP channel) interrupt가 발생하면 오디오 CPLD로 디코딩된 raw 데이터를 출력한다.

본 논문에서 제안한 DSP 플랫폼에서 실제 테스트한 결과, 128KByte의 PS 데이터를 PCI(33MHz)로 전송하는 시간은 1ms 미만이다. 프레임마다 틀리지만, video decoding은 평균 20ms, audio decoding은 6ms, demux는 2ms이므로 다음 video decoding 할 때까지 모든 일을 마친다.

2.3.2 버퍼 Control 기법

본 논문에서의 제안한 스트리밍 환경은 플랫폼의 상황을 고려하지 않은 일정한 시간마다 데이터를 전송하는 구조이므로 버퍼의 언더런 또는 오버런이 발생할 수 있다. 이러한 제약을 극복하기 위하여 오디오와 비디오 버퍼 포인터를 보고 디코딩의 유/무 또는 출력의 skip/wait 기법을 제안하였다.



<그림 4> DSP 플랫폼의 Process Block Diagram

플랫폼 프로그램에서 사용되는 PS 데이터 버퍼, 오디오/비디오 버퍼, 디코딩 된 데이터를 저장하는 버퍼들은 모두 circular 버퍼이다. 각 버퍼들은 write/read pointer, gap 변수를 가지고 있다. 쓰는 시점에서 쓴 양 만큼 write pointer와 gap 변수를 증가시킨다. 처리하는 시점에서 읽은 양 만큼 read pointer를 증가시키고 gap 변수를 감소시킨다. 버퍼의 마지막 부분을 접근한 후에는 이 포인터 값들을 버퍼의 처음 주소 값으로 지정한다. write/read pointer를 이용하여 항상 버퍼 이외의 메모리영역을 접근하지 못하게 하여 일정한 량의 메모리공간을 사용하여 지속적으로 발생하는 데이터를 처리할 수 있다. 변수 gap을 항상 체크하여 gap이 음수가 되었다면 언더런이 발생한 경우이고 버퍼크기보다 크게 되었다면 오버런이 발생한 경우이다.

그림 4는 버퍼 오버런 및 버퍼 언더런을 나타내었다. 버퍼 언더런이 발생하기 전에, gap이 일정한 margin보다 작아지면 데이터가 부족한 경우이므로 이전에 출력한 프레임을 다시 출력하여 wait해야 한다. 이러한 비디오의 경우 화면이 정지되어 있을 것이다. 다시 gap이 margin보다 커진다면 최대한의 화면의 깨짐을 방지하기 위하여 다음 I-FRAME을 찾아서 디코딩을 이어가도록 하였다. 오버런이 발생하기 전에, gap이 일정한 margin보다 커진다면 사용해야 할 데이터를 훼손할 수 있기 때문에 남아 있던 데이터들을 소비하기 위해서 다음 I-FRAME으로 skip 하여 디코딩을 하였다.

3. 결 론

본 논문에서는 고속,고해상도의 멀티미디어 스트리밍 서비스를 위하여 DSP기반의 독립적인 유연 플랫폼을 개발하였으며 이에 탑재될 소프트웨어 모듈과 구조에 대하여 중점적으로 기술하였다. S/W에서는 버퍼의 오버런 또는 언더런을 방지하기위하여 skip/wait 버퍼 control을 제시하였고 H/W에서는 DSP에서 고속의 데이터 처리를 위하여 적합한 메모리/FIFO구조를 제시하였다. 또한 DSP와 IXP 각각의 기능 검증은 테스트베드를 통하여 검증을 하였다. OS는 Embedded Linux를 사용하였고 DSP의 동작 테스트에 사용된 데이터는 MPEG-2 Video, 48kHz 5.1ch AC3(5.1ch 디코딩 후 2ch로 downmix)가 포함된 데이터이다.

차후 논문에서는 DSP에서 다양한 Format의 데이터를 처리할수 있는 소프트웨어 모듈이 개발될 것이다. 이는 본 논문에서 제시한 플랫폼의 적용 분야를 더욱더 확장시키는 연구가 될것이다.

[참 고 문 헌]

[1] Decina, M. "The Internet revolution: reshaping business for the 21st century," Broadband Switching Systems, 1997. Proceedings. 2nd IEEE International Workshop on , 2-4 Dec. 1997

[2] ETSI. TS-101-812, Edition1.2.1, "Digital Video Broadcasting (DVB) Multimedia Home Platform (MHP)", 2003, 6

[3] ATSC Standard A/100-1, "DTV Application Software Environment. Level 1 (DASE-1)," Part1, 2003. 3

[4] 김종윤, 이재식, 김재화, 장태규, "다채널 오디오 equalizer의 ASIC 구현을 위한 time-sharing multiplier 구조에 관한 연구," 한국음향학회 하계학술대회논문집, 제19권 제1(s)호, pp. 343-346, 2000년 7월 8일.

[5] Texas Instrument, SPRU424, "The TMS320 DSP Algorithm Standard - Developer's Guide", 2002, 9.

[6] ISO/IEC JTC1/SC29/WG11 No.6164 "IS 21000-1 (MPEG-21 Digital Item Declaration, DID)," Apr. 2003.

[7] Texas Instrument, SPRA636A, "Application Using the TMS320C6000 Enhanced DMA"

[8] INTEL, "Intel® IXP42X Product Line of Network Processors and IXC 1100 Control Plane Processor Datasheet"

[9] INTEL, "IXP400-RedBoot-2_0-relnotes",April 21.2005