범용 영상보드를 위한 디바이스 드라이버 설계

*김혁중, **최성혁, **김중배, ***이형, *박종원
충남대학교 *정보통신공학과, ***컴퓨터공학과, **전자통신연구소

Design of the device driver for the General Purpose Image Processing Board

*Hyok-joong Kim, **Sung-hyuk Choi, **Jung-bae Kim, ***Hyung Lee, *Jong-won Park
Department of Information and Communications Engineering, Chungnam National University
Department of Computer Engineering, Chungnam National University, ETRI

요 약

영상을 실시간으로 처리하기 위한 연구들이 다양한 방법으로 진행되어 왔다. 실현 가능한 접근 방법중 영상관련 프로그램의 실시간 처리를 수행하기 위해 전용 이미지 처리 보드를 제작하는 방법이 있다. 본 논문에서는 다양한 영상 응용프로그램(얼굴인식, 렌더링, 문서인식 등)을 수행할 수 있는 범용 영상보드의 설계 및 다양한 응용 프로그램들을 개인용 컴퓨터에서 사용할 수 있도록 범용 영상보드에 적합한 Windows NT용 디바이스 드라이버를 설계하는 방법을 제시한다.

1. 서론

영상을 실시간으로 처리하기 위한 연구들을 살펴보면 크게 소프트웨어와 하드웨어적인 접근 방법을 통하여 진행되어 오고 있다. 소프트웨어적인 방법으로는 알고리즘 개발 및 개선이 있으며, 하드웨어적인 방법으로는 범용 프로세서의 지속적인 발전과 특정 영상에 적합한 하드웨어 구조의 개발 및 개선이 있다.

본 논문에서는 하드웨어적인 방법으로 접근하여 특정 영상관련 응용프로그램에 적합한 하드웨어 구조가 아닌 다양한 영상관련 응용프로그램들을 수행할 수 있는 범용 영상보드 설계 및 이에 따른 Windows NT의 디바이스 드라이버의 설계 방법을 제시한다.

범용 영상 보드는 병렬 처리 시스템으로써 기본적으로 SIMD (Single Instruction Multiple Data) 구조로 되어 있으며, 다양한 영상의 병렬 처리를 수행하기 위해서 MIMD (Multiple Instructions Multiple Data) 구조의 기능을 수행하도록 설계되어 다양한 영상 관련 응용 프로그램들을 위한 이식성 및 성능 향상을 위한 하드웨어 구조의 확장성을 갖고 있다. 얼굴인식 및 검색, 문화재 검색, 대용량 인쇄체 문서인식 및 기본적인 영상처리 알고리즘들(필터링, DCT, FFT 등)을 병렬 알고리즘으로 변화시켜 모의실험을 통하여 검증하였다[1,2,3].

범용 영상 보드를 개인용 컴퓨터에 장착하여 다양한 영상 관련 응용프로그램들을 개발하고 응용하기 위해 필요한 최소의 요소가 디바이스 드라이버이며, Windows NT는 여러 플랫폼의 응용프로그램들을 실행시킬 수 있도록 다중 API를 지원하기 때문에 Windows NT 4.0 Workstation 상에서 DDK로 설계하였다.

본 논문의 구성은 다음과 같다. 2장에서는 범용 영상보드 설계, 3장에서는 Windows NT의 구조, 디바이스 드라이버의 개요, 그리고 범용 영상보드를 위한 디바이스 드라이버의 설계를 기술할 것이다. 마지막으로 4장에서는 앞으로 수행될 연구방향에 대하여 논의한다.

2. 범용 영상처리 보드

범용 영상처리 보드는 PCI9080 Server, Intel I960Kx, local memory , 병렬 처리 모듈로 구성되어 있으며 블록도는 그림 1

과 같다. 병렬 처리 모듈은 MAMS (Multi-Access Memory System)[4], n개의 PE (Processing Element), Intel960 인터페이스로 구성되어 있다. 또한 MAMS은 n+1개의 전용 메모리 , 주소계산회로, 데이터 라우팅 회로, 그리고 전용 메모리 선택 회로로 구성되어 있다.

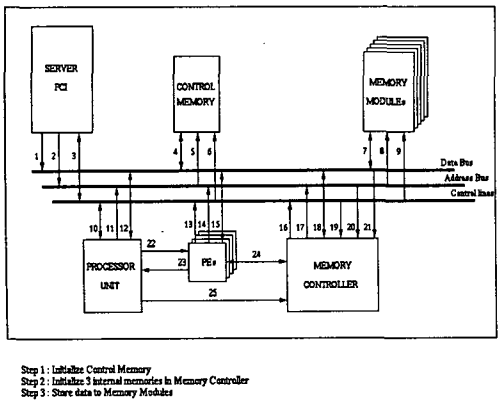

그림 1. 범용 영상 보드의 블록도

수행 과정은 응용 프로그램이 local memory에 저장되면 Intel I960이 명령어들을 읽어와서 병렬처리모듈을 제어하게 된다. 이 과정에서 병렬처리모듈은 호스트 시스템으로부터 PCI버스로 전송된 영상 데이터들을 MAMS 내의 n+1개의 전용 메모리에 저장되게 된다. Intel I960은 병렬처리모듈 내부의 n개의 PE들을 제어하여 동시에 PE들이 n개의 영상데이터를 SIMD 또는 MIMD 방식으로 처리하게 된다. 그리고 수행한 결과는 다시 호스트 시스템으로 전송하게 된다. 이러한 수행과정은 하드웨어 모의실험 패키지인 CADENCE Verilog-XL을 이용하여 검증하였다(그림 2)

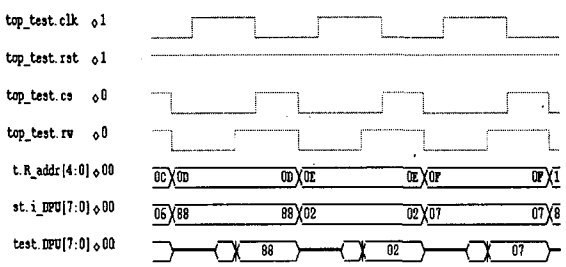

그림 2. Intel960 프로세서 모듈과의 데이터 전송 상태를 보여주는 파형

3. NT Device Driver 설계

디바이스 드라이버는 보드를 사용하기 위한 최소한의 필요한 요소이다. Windows NT 4.0 Workstation상에서 DDK로 작성하였다.

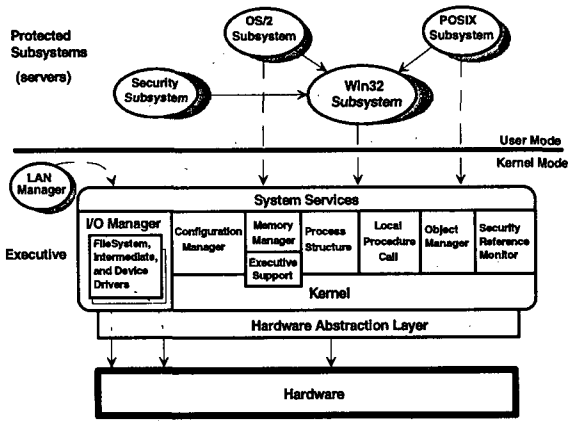

그림3. Windows NT의 구조

3-1. Windows NT의 구조

Windows NT는 여러 플랫폼의 응용 프로그램들을 실행 시킬수 있도록 다중 API를 지원한다[5]. 즉, Win32, 16bits windows 응용 프로그램, OS/2 응용 프로그램, MS-DOS 응용 프로그램, POSIX 응용 프로그램등을 제공하는 클라이언트/서버 모형이다. 또한 NT는 객체모형을 가지며, 공유 메모리 다중처리기(shared-memory multiprocessor)시스템이다. 대칭적 다중처리(symmetric multiprocessing)방식을 채택해 운영체계의 코드는 사용 가능한 여러개의 프로세서에서 동시에 수행한다.

Windows NT의 구조(그림 3)는 사용자 영역 부분(Protected subsystem)과 커널 영역부분(NT Executive)으로 나누어진다[6,7]. 커널은 모듈들로 나누어져있는데 각 모듈들은 함수 호출의 형태로 서로에게 자신의 기능을 제공해 준다. 시스템 서비스는 UNIX의 시스템 호출과 유사한 형태로서, NT 실행자(NT Executive)가 사용자 수준에게 제공하는 서비스들의 집합을 의미한다. 시스템 서비스는 NT 실행자에 존재하는 여러 모듈들이 제공하는 기능을 조합하여 만들어진다. NT 실행자 안에서 커널이라 지칭되는 모듈은 NT실행자의 핵심이라 불리울 수 있는 모듈로, 다른 모듈들의 기능이 이 커널에서 구현된 기능을 근간으로 하여 구현된다. Windows 커널과 입출력 장치들의 기능 구현시 물리적 하드웨어에 의존적인 부분에 대한 기능은 HAL(Hardware Abstraction Layer)이라고 불리는 모듈에 의해 구현된다.

3-2. 디바이스 드라이버

디바이스 드라이버는 크게 세 가지 기준에 의해 나누어질 수 있다[6,7]. 실행환경(user-mode, kernel-mode drivers), Laying (Device drivers, Intermediate drivers, FSDs), 그리고 하드웨어 (SCSI drivers, network drivers, Special device drivers)가 기준이며. 구현하고자 하는 device driver는 kernel mode이며, Device drivers이다.

디바이스 드라이버의 구조

디바이스 드라이버는 (표 1)과 같은 기본적인 요소들을 따라서 구현이 된다[6,7].

표 1. 표준 루틴

Routines	설 명
Initialization routine	Device Driver 및 H/W 초기화
A set of Dispatch routine	I/O 요청에 대한 진입구
A start I/O routine	physical device에게 I/O 요청
An ISR routine	Interrupt Service Routine 높은 IRQL에서 수행됨
An DPC routine	대부분의 인터럽트 일 처리 낮은 IRQL. 대기중인 I/O 요청처리
An unload routine	resource 해제, driver 제거

기본적으로 드라이버의 Entry point와 리소스들을 초기화 시켜주고, object들을 생성하는 디바이스가 구동되기 위한 Driver Entry routine (initialization routine)으로부터 디바이스 드라이버는 시작된다. 그리고, 디바이스를 사용하고자 할 때, Dispatch routine을 통해 I/O요청을 받아들여, 디바이스에 I/O를 요청하여 처리하게 된다. 이 경우 IoGetCurrentIrpStack Location(IRP)->MajorFunction으로 구분 사용한다. 인터럽트가 발생 시에 그것을 처리해주고(ISR, DSP routine), 모든 디바이스의 일이 끝났을 경우, unload routine을 통해 사용했던 리소스들을 해제하고, driver를 제거하여 초기화상태로 만들어준다.

I/O의 처리시 IRP(I/O request packet)를 사용하여서 처리한다. 또한 NT의 특성처럼 object를 이용하여 device를 구동시킨다.

3-3. 디바이스 드라이버 디자인

디바이스 드라이버의 기본적인 고려 사항은 다음과 같다.

; 사용 Bus 종류
· 사용 interrupt
· 메모리 접근 방식

보드와 메인보드와의 하드웨어 인터페이스의 복잡성이 디바이스 드라이버의 입장에서는 중요한 고려 사항이 아니며 디바이스 드라이버는 보드에서 발생하는 해당 시그널(인터럽트)을 해석하여 그에 따른 기능을 수행하면 된다.

4. 향후 연구 방향

본 논문에서 제시한 보드는 PCI Bus를 사용하며, memory mapped 방식으로 데이터를 전송한다. 인터럽트에 관한 내용은 우선 고려치 않도록 하였다. 현재 타겟 보드는 제작중이기 때문에 제작하는 보드와 비슷한 상용 보드를 대상으로 테스트를 수행하였다. 앞서 말한 바와 같이 디바이스 드라이버를 디자인하는데 필요한 사항은 버스의 종류, 메모리 억세스방법, 인터럽트 등이다.

테스트중인 상용 보드는 'PCI-SDK Evaluation Platforms'라는 것으로 PCI 버스를 사용하며, 메모리 맵 방식으로 데이터를 전송하며, 또한 타겟 보드의 구조와 비슷하다는 점에서 같은 구조의 디바이스 드라이버를 구현할 수 있다. 제작되는 타겟 보드와 같은 메인보드에서 데이터의 시작주소와 크기를 전달함으로써 memory mapping 시키는 매커니즘으로 디바이스 드라이버를 작성하여 테스트중이다.

드라이버의 작성은 Read, Write 기능만을 가지고 구현하고, Buffered I/O 방식으로 사용하였으며, memory mapping 실험을 하고 있다. 현재는 간단한 I/O는 성공한 상태이며, memory mapping은 진행중이다. 이런 과정을 거친 후에, 인터럽트 부분과 제작중인 타겟 보드에 필요한 옵션들을 모두 테스트한 뒤, 타겟 보드가 제작됨과 동시에 드라이버를 테스트, 어플리케이션을 작성 사용할 것이다.

참고문헌

[1] 유기현, 박종원,"효율적인 DCT 처리기의 개발", 대한 전자공학회지, Vol.32, No.8, pp.41-46, Aug., 1995

[2] 문경애,김의정,이형,박종원,"다중접근기억구조를 이용한 고속 인쇄체 문서인식," 제10회 신호처리학술대회 논문집 제10권 1호, pp. 327 ~ 330, Dec. 1997.

[3] 이형,박종원,한기선,김원봉,"얼굴 검색을 위한 병렬처리기 구조," 한국정보과학회, '97 가을 학술발표 논문집(II), pp. 471 ~ 474, Dec. 1997

[4] 김길윤, 박종원,"효율적인 영상분석 메모리 시스템," 한국정보과학회, Vol.16, No.16, pp.559-566, Sep., 1989.

[5] "Inside Windows NT", Helen Custer, Microsoft Press

[6] "The Windows NT Device Driver Book", Art Baker, Prentice Hall

[7] "Kernel mode Drivers", Microsoft, Windows NT DDK