PCI 비젼 시스템 개발

김정훈, 전재욱, 변종은
성균관대학교, 성균관대학교, 넥스트아이

Development of PCI Vision System

Jeong-Hun Kim, JaeWook Jeon, Jong-Eun Byun
Sungkyunkwan Univ., Sungkyunkwan Univ. NextEye

Abstract - Vision systems need to have high speed transfer methods for transferring large data. After PC accepts PCI, PCI becomes a more effective method for data translation. PCI substitutes previous ISA. This paper proposes an architecture of vision system and window driver based on PCI.

1. 서 론

PC 시스템이 보다 빨라짐으로 해서 기존의 ISA보다 빠른 전송률을 갖는 PCI가 생기게 되었다. 인터페이스 하는 방식이 PCI로 대체됨에 따라 기존에 ISA에서 설계하던 방식에서 벗어나 새로운 구조를 생각하게 되었다. 비젼 시스템은 영상이라고 하는 많은 데이터를 카메라를 통해서 PC로 가져가야 하기 때문에 비젼 처리를 하지 않고 순수하게 데이터를 얻는 시간동안에도 시스템이 부하가 걸리게 된다. 때문에 PCI의 특징인 Scatter & Gather DMA 전송 방식을 사용해서 시스템을 구성하게 되었다. 물론 이것이 DMA라고 하는 기존 전송 방식에 특정목적으로 메모리를 여러 조각으로 할당하고 디스크립터를 두어 데이터를 전송하는 방식이기 때문에 효율이 떨어질지 모르나 특별히 하드웨어를 복잡하게 구성하지 않고도 데이터를 효과적으로 전송할 수 있는 장점을 갖기 때문에 본 논문에서는 비젼 시스템 설계에서 이 방식을 사용하였다. 특히 요즘 나오는 시스템들이 보다 사용자가 쉽게 사용할 수 있게 윈도우즈 환경의 GUI와 Plug&Play를 요구하고 있기 때문에 윈도우즈와 관련된 PCI 드라이버에 대해서도 소개한다. 여기서는 PCI 비젼 시스템을 만드는데 있어 고려해야 하는 사항들을 단계 별로 설명하고자 한다.

2. 본 론

2.1 PCI 비젼 시스템 전체 구성

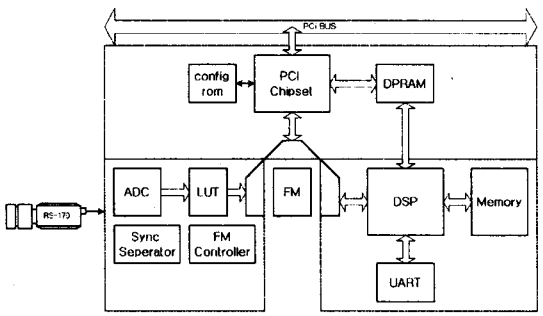

그림 1 PCI 비젼 시스템 전체 구성

ISA와는 달리 PCI Spec.에 따라 데이터를 전송하기 위해서는 PCI 칩셋을 사용해서 구성해야 한다. 그림 1은 PCI 비젼 시스템의 전체 구성을 나타내고 있다. 그림에서 보듯 사용자의 목적에 맞게 Config ROM을 변경함으로써 PCI 자원을 할당하고 관리할 수 있다. PC와 DSP 사이에 메시지 교환을 위해서 DPRAM(Dual Port RAM)이 존재하며 FM(Frame Memory)를 기준으로 3종류의 버스가 연결되어 있다. 그것은 ADC의 버스, DSP의 버스, PCI의 로컬버스가 그것이다. 즉 FM을 기준으로 3 입력 MUX를 구성하고 있다. 실제 비젼시스템의 동작 순서를 살펴보면 그림 2와 같다.

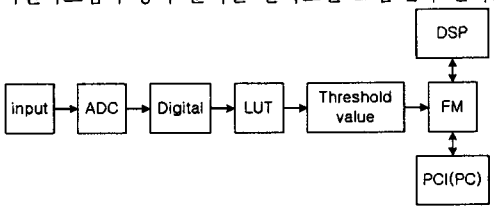

그림 2 비젼 시스템 동작 순서

카메라의 아날로그 입력이 ADC에 의해서 디지털 값으로 변환되고 LUT(Look Up Table)을 거쳐서 필터링 된다. 필터링 된 영상은 프레임 메모리에 저장되며 DSP나 PC에 의해서 억세스가 되고 비젼 처리된다.

2.2 DSP(Digital Signal Processor)

TI 사의 TMS320C32를 사용하였다. 어드레스가 24비트로 구성되어 있는 것이 특징이며 32, 16, 8비트로 데이터 폭을 선택할 수 있다. 부트로드를 해서 시스템을 구동하는 방법을 사용한다. 디코드 없이 외부에 디바이스를 붙일 수 있도록 3개의 스트로브 신호(STRB0, STRB1, IOSTRB)가 존재한다. 표 1은 사용된 메모리 영역을 나타내고 있다.

표 1 Device decoder map

/STRB0	1xxxxxh : 메인 메모리
	2xxxxxh : FM
/IOSTRB	8000xxh : LUT
	8001xxh : LUT bank
	8002xxh : register
	8003xxh : DPRAM
	8004xxh : RS232c
	8005xxh : inport
/STRB1	BOOT 3 : ROM

DSP와 PC사이의 메시지 전달은 DPRAM을 통해서 이루어진다.

2.3 PCI

PCI(Peripheral Component Interconnect)는 동기 클럭 33MHz를 사용하고 32비트 데이터 폭을 갖기 때문에 최대 132MBytes/sec 까지 전송할 수 있다. PCI에 대한 스팩은 PCI SIG(Special Interest Group)에서 관리하고 있기 때문에 자세한 내용을 확인할 수 있다. 여기서는 PLX Tech. 사의 PLX9080을 사용하였다. 이 칩의 특징을 간단히 정리하면 Master

칩이고 DMA 컨트롤이 가능하다는 것이 특징이다. PC가 부팅할 때 PCI 디바이스를 인식하기 위해서는 각각의 PCI에 대해서 Configuration을 해야한다. 그렇지 않으면 부팅 할 때 어떤 디바이스가 슬롯에 장착되어 있는지 PC는 알 수 없게 된다. 또한 Widows환경에서 PCI 디바이스를 사용하기 위해서는 디바이스 드라이버를 작성해서 PCI와 연결시키는 과정이 필요하다.

2.3.1 Configuration

PCI Configuration은 PCI BIOS에서 관리되는 PCI 버스에 관한 정보가 기록되는 영역이다. 이곳에는 디바이스 ID, 벤더 ID 뿐만 아니라 자원 영역까지의 정보를 적어야 OS에서 PCI로 억세스가 가능하다. 처음에 시스템이 부팅할 때 PC에서는 각각의 PCI 디바이스의 configuration 레지스터로부터 원하는 리소스 정보를 확인하고 할당 가능한 영역을 리턴 해주는 방식으로 PCI는 플러그 앤 플레이의 기본 기능을 수행한다. 즉 ISA와는 다르게 기본적으로 플러그 앤 플레이가 수행된다. 그림 3과 같이 전체 Configuration 블록을 구성하고 있고 사용하지 않는 영역은 00으로 설정하는 것이 원칙이다.

오프셋	31	24	23	16	15	8	7	0
00h	Device ID				Vendor ID			
04h	Status				Command			
08h	Class Code							Revision ID
0Ch	BIST		Header Type		Latency Timer		Cache Line Size	
10h	PCI Base Address 0							
14h	PCI Base Address 1							
18h	PCI Base Address 2							
1Ch	PCI Base Address 3							
20h	PCI Base Address 4							
24h	PCI Base Address 5							
28h	Cardbus CIS Pointer							
2Ch	Subsystem ID				Subsystem Vendor ID			
30h	PCI Base Address for Local Expansion ROM							
34h	예약되어있음							
38h	예약되어있음							
3Ch	Max_Lat		Min_Gnt		Interrupt Pin		Interrupt Line	

그림 3 PCI Configuration block

오프셋 00h와 08h의 클래스 코드는 각각 908010b5h와 0400h를 적어 넣어 주었다. 오프셋 10h ~ 12h 사이는 메모리 영역을 나타내고 있는데 'f'에 의해서 영역을 PCI BIOS에 알려주면 남아있는 리소스 중에 영역이 할당되어 리턴 된다. 메모리 영역으로 0xfff00000를 I/O 영역으로 0xfffffe1을 써주었고 본 시스템에서는 메모리 영역으로 0xe0800000과 I/O 영역으로 0x0000b801을 할당받았다. 즉 메모리 영역은 프레임 메모리와 직접 연결되는 영역이고 I/O 영역은 DPRAM과 연결되는 영역이다. 이렇게 하면 PC는 개발된 비젼 시스템을 인식하게 된다.

2.3.2 PCI windows driver

윈도우즈에서 PCI의 리소스를 사용하기 위해서는 각각의 디바이스 ID/벤더 ID에 해당하는 드라이버가 필요하다. 윈도우가 부팅 될 때 위의 Configuration 과정을 수행하게 되면 PCI 디바이스가 인식되고 그에 따라 드라이버를 요구하게 된다. 드라이버 안에 들어가야 하는 기능은 다음과 같다. 윈도우즈 상의 Configuration 싸이클을 수행하고 인식된 디바이스 ID와 벤더 ID를 레지스트리에 추가한다. 각각의 리소스에 대한 자원 할당과정을 수행하며 할당된 자원정보 또한 레지스트리에 등록한다. 경우에 따라서 메모리 영역은 lock 된다. 다른 자원과는 다르게 인터럽트 정보는 처음 부팅 될 때만 설정을 할 수 있기 때문에 이 또한 드라이버 초기화 과정에 포함된다.

그 외에 PLX9080 의 기능 설정용 라이브러리가 추가된다. 여기에는 DMA를 제어하거나 Scatter & Gather List를 만들기 위한 함수가 포함된다. 작성된 드라이버 소스는 DDK(Device Driver Kit)에 의해서 컴파일 되고 원하는 *.vxd 코드를 얻게 된다. 생성된 드라이버를 윈도우즈에 등록하기 위해서는 INF(information file)을 사용해서 등록하게 된다. 그림 4는 INFEDIT.EXE를 사용해서 INF 파일을 만드는 과정을 보여 주고 있다.

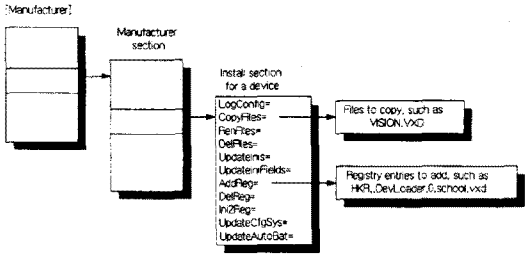

그림 4 INF 파일을 만드는 과정

이렇게 만들어진 INF 파일과 VxD파일을 바탕으로 개발된 비젼 시스템을 등록 하게되면 윈도우즈 환경의 응용프로그램에서 비젼시스템을 억세스할 수 있게 된다.

2.3.3 Configuration Register에 접근

PC내의 Configuration 레지스터는 I/O 영역을 통해서 접근할 수 있다. 두 개의 I/O 어드레스 0xcf8과 0xcfc를 이용한다. 여러 개의 PC 디바이스를 각각 구분하기 위해서 버스 번호와 디바이스 번호를 가지고 구분하면 configuration 내의 옵셋과 조합해서 전체 어드레스를 결정한다. 처음에는 버스 번호와 디바이스 번호에 대한 값을 알지 못하기 때문에 원하는 벤더 ID와 디바이스 ID를 갖는 영역을 찾아야 한다. 버스번호와 디바이스 번호를 증가시키면서 루프를 돌리게 된다. 그림 5는 이런 과정을 수행하는 순서도 이다.

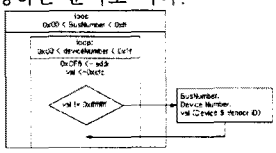

그림 5 PCI 디바이스를 찾는 루틴

버스 번호와 디바이스 번호를 바탕으로 메모리 영역과 I/O영역의 어드레스 값을 읽어서 그 값은 기록해 두면 프레임메모리 영역과 I/O영역을 쉽게 억세스 할 수 있다.

2.3.4 SGL에 의한 DMA전송

SGL(Scatter & Gather List)에 의한 DMA 전송을 하기 위해서는 전송하기 위한 메모리 블록을 여러 블록으로 나누어서 전송하고자 하는 메모리 블록과 그것을 설명하는 블록을 구별해서 설명하는 블록에는 전송 순서대로 시작 어드레스와 길이를 기록하게 된다. SGL 모드로 전송을 하게 되면 비록 메모리 영역이 끊어져 있다고 해도 마치 연결된 메모리를 전송하듯 DMA를 사용해서 전송할 수 있다. 개발된 비젼 시스템에서는 y축 영상을 480개의 블록으로 나누어서 전송하고 있다. SGL로 전송하는 것과 일반적인 블록으로 DMA를 전송하는데 있어서 속도 차이는 있다. 하지만 전송하고자 하는 데이터가 연속적으로 되어 있지 않고 끊어져 있는 경우 이용 가치가 크다. 다음 표 1은 실험에 의해 구해진 비트별 전송 속도를 나타내고 있다. 블록 DMA전송에 비해서 SGL DMA가 약94%의 전송률을 보인다고 볼 수 있다.

표 1 비트별 전송 속도

width	sgl 개수	nonburst (Mbytes/sec)	burst (Mbyte/sec)
8 bit	480	16.17 ~ 18.07	25.60~33.00
	1	16.17 ~ 20.48	27.92~33.00
16 bit	480	30.72 ~ 43.88	51.20~66.00
	1	43.80 ~ 51.2	66.00
32 bit	480	70.22 ~ 81.92	81.92~122.88
	1	81.92 ~122.88	98.30~132.00

2.3.5 인터럽트에 의한 디스플레이

DSP에서는 DPRAM을 통해서 PC로 디스플레이 명령을 보내주면 PCI쪽으로 인터럽트가 발생하게 된다. 미리 만들어 진 드라이버의 인터럽트 콜백 함수에 의해서 어플리케이션에 메시지를 전달하게 된다. 발생된 메시지에 해당하는 디스플레이를 위한 루틴을 수행하게 된다. 디스플레이 루틴은 다이렉트 X에 의해서 작성되며 디스플레이 속도를 최대화하였다. 다음 그림은 각각 LUT를 Linear, Inverse로 설정했을 때 디스플레이 되는 결과를 나타내고 있다.

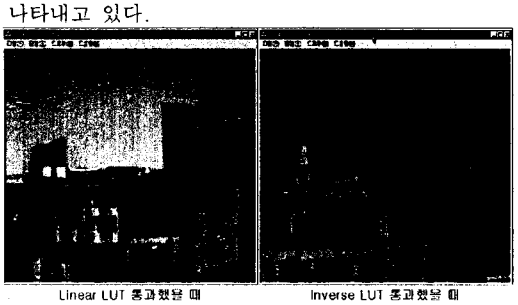

그림 6 디스플레이 된 결과

2.4 실험 환경

아래의 표 2에 나와 있는 환경에서 비젼 시스템을 작동해 보았다. 기존에 ISA 환경에서 오버레이 칩셋을 사용해서 PC 모니터에 뿌려주던 비젼 시스템과는 달리 PC의 성능을 요구하는 것은 사실이다. 그 이유는 다이렉트 드로우를 사용해서 PC 모니터에 뿌려주기 위해서는 무시 못할 만큼의 시스템 자원을 사용하기 때문이다. 그러나 다이렉트 드로우를 지원하는 모든 그래픽 카드에 대해서 충돌 없이 사용할 수 있는 장점을 갖는다. 기존의 비젼시스템은 오버레이 칩이 지원하는 그래픽 카드만 사용할 수 있다는 제한 사항이 있었다. 그리고 PC가 PCI BIOS에서 버전 2.1을 지원하는지를 확인할 필요가 있다. 구형 PC에서는 작동할 수 없었다.

표 2 실험 환경

종류	특징
CPU	Pentium II-300
OS	Windows 98
Graphic	Matrox Millinium II G200
PCI SPEC.	version 2.1
Direct-X	version 6.0

그림 7은 완성된 비젼 시스템을 나타내고 있으며 3부분으로 나누어져 있다. 첫 번째는 RS-170 모듈로서 카메라로부터 영상을 받아서 순서대로 프레임 메모리에 저장하는 부분이고 두 번째 PCI 보드는 PC 슬롯에 연결되는 부분이고 세 번째 DSP보드는 비젼 처리를 하기 위해 부분이다. 각각의 모듈이 플랫케이블로 서로 연결되어 있는 구조로 되어 있다.

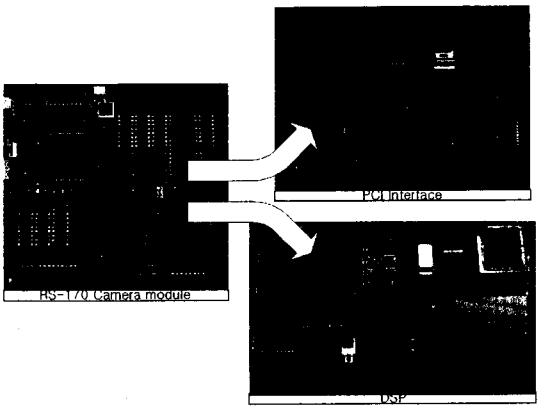

그림 7 개발된 시스템

3. 결 론

본 논문에서는 PCI 인터페이스를 사용해서 비젼 시스템을 개발하는데 있어서 필요한 하드웨어와 소프트웨어 각각의 측면에서 연구하였다. PCI는 기존의 ISA 시스템에 비해서 소프트웨어의 비중의 크다는 사실을 알 수 있다. 특히 데이터를 전송하는데 있어서 SGL DMA를 사용하면 인터레이스와 넌인터레이스를 모두 지원하는 비젼 시스템을 개발하고자 할 때 매우 적합함을 알 수 있다. 640픽셀씩 나누어서 전송을 하기 때문에 결국 PCI 메모리 맵에 저장되어 있는 영상의 배열 순서는 같게 된다. 대부분의 제어는 기존의 ISA에서는 하드웨어 점퍼 핀을 설정하던 방식에서 PCI에서는 Configuration 레지스터를 소프트웨어로 바꾸는 방식으로 바뀌게 되었다. 즉 보다 시스템을 융통성 있게 사용할 수 있게 된 것이다. 아직은 Windows 98에서만 테스트가 된 상태이고 앞으로 Windows NT나 2000에서도 테스트를 계속할 계획이다.

[참 고 문 헌]

[1] Tom Shanley, "PCI System Architecture", Mindshare, 1997

[2] Walter Oney, "System Programming for Windows 95", MSpress, 1996

[3] Walter Oney, "Programming the Microsoft Windows Driver Model", MSpress, 1999

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

[5] Edward N.Dekker, "Developing Windows NT Device Driver", Addision-Wesley, 1999

[6] "PCI Local Bus Specification Rev 2.2", PCISIG, 1998

[7] Ramesh Jain, "MACHINE VISION", McGraw-Hill, 1995

[8] Louis J. Galbiati, "Machine Vision and Digital Image Processing Fundamentals", Prentice-Hall, 1990