멀티미디어 데이터 처리를 위한 임베디드 시스템에 효율적인 DMA 제어기의 사양

김 진 헌*

1. 서 론

컴퓨터 시스템은 크게 보아 CPU, 메모리, I/O 장치로 나눌 수 있다. 컴퓨터는 정해진 프로그램을 수행하는 과정에서 다양한 I/O 장치와의 교신을 수행하게 마련이며, 이렇게 I/O 장치의 데이터 전송을 효과적으로 수행하는 기법으로 활용되는 것이 DMA(Direct Memory Access)이다. 특히 대량의 영상 데이터와 음성 데이터를 다루는 멀티미디어 시스템의 구축을 위해서는 이러한 I/O 장치의 데이터 처리에 많은 부담이 따를 수밖에 없다.

DMA는 컴퓨터 시스템의 입출력 장치(I/O device)와 메모리간의 데이터 전송, 때로는 메모리와 메모리간의 데이터 전송을 담당하는 주요한 수단으로서 다음 표 1과 같은 I/O 장치의 처리기법[1] 중에서 가장 바람직한 기법으로 볼 수 있다.

DMA 방식은 잠시 CPU를 시스템 버스에서 격리하고 DMA 제어기가 I/O 데이터를 전송하는 하게 하여 빠른 속도로 대용량의 데이터를 전송할 수 있기 때문에 영상, 음성과 같은 대량의 멀티미디어 데이터 혹은 대용량의 통신 데이터의 처리에 이상적인 전송기법이라고 말할 수 있다. 영상 데이터의 경우 RGB(Red/Green/Blue) 3바이트의

Table 1. Implementation Methods of I/O Sub-System

Methods	Description	Features	Master of Transfer
Programmed I/O	• CPU waits until I/O device ready. • CPU polls multiple I/O devices.	• Waste of CPU Power • Inadequate for Multi tasking, Multi user Environment	CPU
Interrupt Driven I/O	CPU transfers I/O data when the interrupt is invoked.	• Low Efficiency due to the Massive Stack Operation • Adequate for Infrequent I/O operations (cf. Keyboard, Printer)	
DMA Driven I/O	DMA controller does transfer while the CPU is separated from the system bus	• High Efficiency • Several Transfer Cycles • Adequate for Massive Data Transfer (cf. LAN, HDD, Video)	DMA Controller

* 서경대학교 컴퓨터공학과 교수

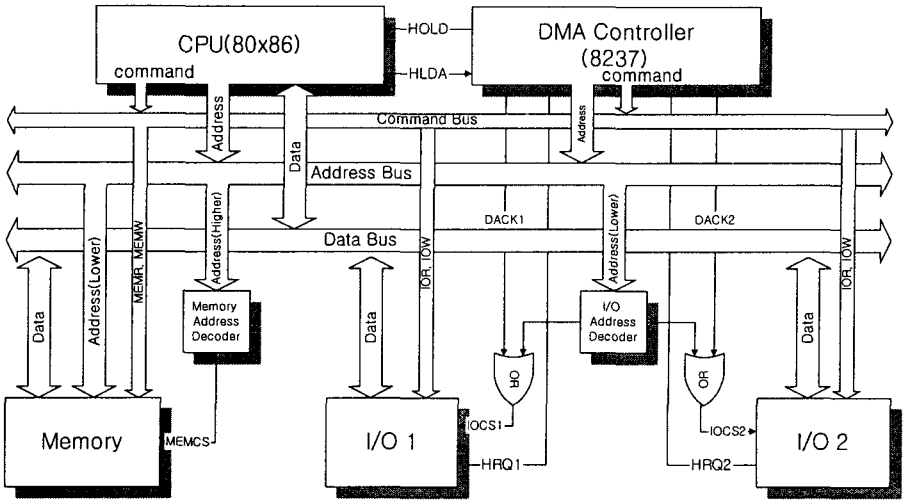

Fig. 1 ISA DMA Architecture

픽셀(pixel)로 이루어진 해상도가 640*480인 영상데이터를 실시간(30 frames/sec)으로 전송하자면 초당 약 26.4 MB/sec의 데이터 전송능력이 필요하다.

최근 들어 산업계에서 많이 쓰이는 임베디드 프로세서(ARM, MPC860) 혹은 DSP(Digital Signal Processor)들의 클럭 스피드의 급격한 신장, 구조적 측면의 개선(RISC 채택, 하버드 구조의 메모리 분리, 캐시의 내장, 속도 빠른 버퍼 메모리의 내장, SDRAM 메모리 제어기의 내장 등)으로 프로세서 코어의 처리속도는 급격히 신장하는 추세에 있다. 그러나 이들 프로세서 코어의 성능제고를 DMA 제어기가 따라가지 못하고 있어 종합적인 시스템 성능 관점에서 보면 효율성이 떨어지게 된다. 특히 멀티미디어 데이터 처리 관점에서는 I/O 장치와 메모리간의 전송 능력의 문제는 개선의 여지가 많아 보인다.

본 논문에서는 기존의 임베디드 프로세서 혹은 일반 프로세서 및 시스템에서의 DMA 구현 사례를 알아보고, 이들의 공통적인 개념을 도출하여 성능 평가의 지표를 살펴본 후 이를 통해 멀티미디어 데이터 처리에 효율적인 DMA 제어기의 사양 및 구조의 기본 개념을 도출해 보고자 한다.

2. DMA 구현 방법에 대한 고찰

본 절에서는 현존하는 시스템 혹은 프로세서의 DMA 구현 방법에 대해서 알아보고 이를 통해 멀티미디어 데이터를 처리하기 위한 관점에서 각 DMA 제어기가 어떻게 활용될 수 있는지 살펴보기로 한다.

2.1 PC ISA DMA - 8237

ISA(Industrial Standard Architecture)의 DMA는 DMA 서브 시스템의 가장 기본형이라고 말할 수 있다. 80×86 계열의 CPU는 분리된(Isolated) I/O 구조를 갖는다. 따라서 메모리와 I/O 영역이 구분되어 있으며, 별도의 메모리 읽기/쓰기(MEMR/MEMW) 신호와 I/O 읽기/쓰기(IOR/ IOW) 신호를 제공한다[1]. ISA에서 사용한 DMA 제어기는

1) 앞으로 제시될 신호들은 Active Low 신호도 있으나 본 본문에서는 개념 전개를 위해 구분하지 않기

8237[3]이다. 이것은 내부에 ① 모드정보(전송 방향. 메모리→I/O, 메모리 ←I/O, 메모리↔메모리) ② 메모리의 기준번지 ③ 전송할 데이터의 크기 정보를 담고 있다. 이와 같은 정보는 사전에 CPU에 의해 초기화된다. 그림 1에 8237에 의한 ISA DMA 구조의 데이터 흐름을 보이는 블록도를 보였다.

I/O 장치로부터 전송 요구(DRQx)가 들어오면 DMA 제어기는 CPU에 버스로부터 분리(3-State)될 것을 요구하는 신호(HOLD)를 방출하고, CPU로부터 버스를 놓았다는 신호(HLDA)를 접수하면 DMA 제어기는 자신이 버스 마스터가 되어 번지/명령(Address/Command) 버스를 장악하게 된다.

예를 들어 I/O 장치로부터 메모리로 전송하는 사례를 가정하면; DMA 제어기는 메모리에게는 번지와 메모리 읽기 신호(MEMR)를 제공하고, I/O 장치에게는 I/O 쓰기 신호(IOW)를 동시에 제공하여 I/O 장치의 데이터가 CPU를 거치지 않고 바로 메모리로 전송되는 플라이바이(fly-by) DMA를 구현하고 있다. CPU에 의한 일반적인 I/O 쓰기 동작은 제공되는 번지(I/O)에 의해 해당 I/O 장치에게 선택신호(CS)를 제공하게 되는데 DMA 사이클의 경우에는 I/O 장치에 제공되는 장치허가(DACKx)신호로 I/O 장치에게 선택되었음을 알린다. 따라서 I/O 장치는 CPU의 접근에 대비하여 칩선택 신호를 두 가지 방법으로 제공받는다.

이와 같은 방식의 DMA는 DMA 채널간의 연속호출(chaining)이 지원되지 않아서 불연속 메모리 블록[2]을 전송하기 위해서는 DMA 제어기를 매번 재 초기화해야 하는 단점이 있다. 따라서 실시간으로 가변 해상도의 영상 데이터를 전송하기에는 부적합하다.

2.2 MPC 860의 IDMA

MPC 860[3]의 DMA는 크게 통신 장치를 위한 전용 DMA인 S(Serial)DMA와 일반 I/O 장치를 위한 I(Independent)DMA로 나뉜다[3]. 실제로는 2개의 물리적인 직렬 DMA를 보유하면서 16개의 가상 SDMA를 구현하는데 보내기/받기용의 채널을 구분하여 4개의 SCC(SCC1~SCC4)와 SPI, I^2C, SMC(SMC1, SMC2) 총 8개의 직렬 DMA를 지원한다. IDMA는 2개의 물리적인 DMA 제어기로 '메모리 ↔ 메모리'와 'I/O 장치 ↔ 메모리' 간의 데이터 전송을 담당한다.

IDMA 데이터 전송은 그림 2에 보인 바와 같이 IDMA 외부에 있는 듀얼 포트램(dual-port RAM, 5KB)의 특정 부분에 있는 IDMA 파라미터 램의 정보영역을 참조로 해서 이루어진다. 파라미터 램에는 BD(Buffer Descriptor)의 기준 위치와 데이터의 전송방향(메모리↔I/O, 메모리↔메모리), 'Dual/Single Cycle' 여부 등을 선정한다. BD의 정보 중 가장 기본적인 것은 시작지점과 목표지점의 주소와 전송할 버퍼의 크기(길이)이다. 이외에도 본 BD의 유효성 여부, 이 버퍼가 끝인지의 여부, 전송 후에 인터럽트를 발생할 지의 여부, 연속해서 버퍼 전송을 계속할지의 여부 등을 결정한다. BD는 16바이트로 구성되며, 듀얼 포트램에 존재하여 빠른 접근이 용이하다. 또한 듀얼 포트램에는 총 192개의 BD를 저장할 수 있어서 다양한 블록으로의 전송이 가능하다.

BD의 옵션중 버퍼 체이닝 모드는 전송이 끝나면 BD의 유효비트를 해제하여 다른 연속한 다른

로 한다.

2) 영상 데이터의 경우 비디오 메모리상의 화면의 줄과 줄 사이에는 메모리 주소간의 불연속성을 갖는 것이 대부분이다.

3) 멀티미디어 데이터 처리 관점을 유지하기 위해 여기서는 IDMA만을 다루기로 한다.

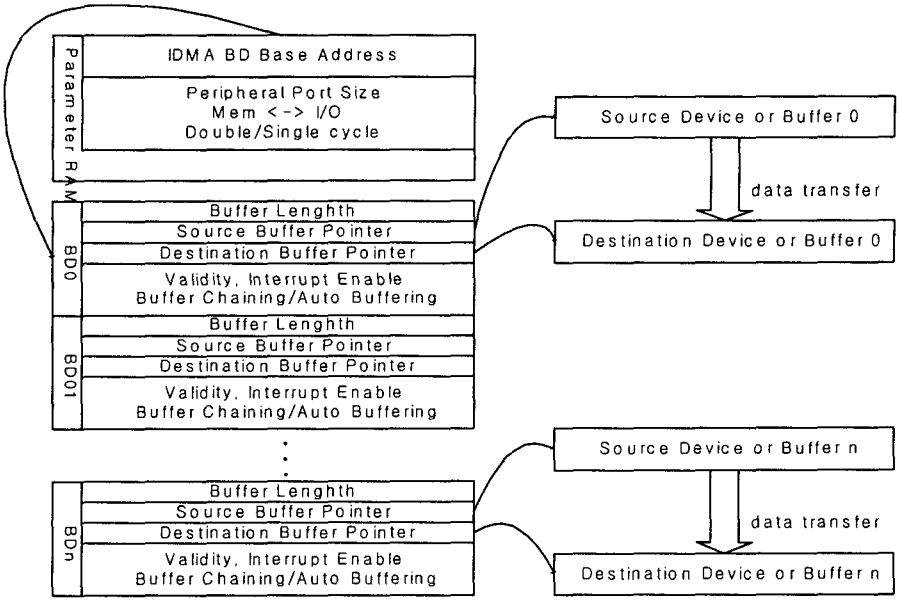

Fig. 2 IDMA BD Table & Parameter RAM

BD에 의해 전송을 계속하는 모드이다. 이는 대용량의 불연속 데이터를 전송하기 위해서 사용할 수 있다. 오토 버퍼링 모드는 전송을 마친 후에도 사용된 BD가 계속 유효함으로써 같은 영역으로 계속 데이터를 전송할 수 있게 하는 모드이다. 이런 모드를 사용한 한 사례로는 음성 A/D 입력을 모니터링하는 사례를 들 수 있다.

플라이바이 모드는 파라미터 램의 'Dual /Single Cycle' 비트로 설정한다. 'Dual Cycle'로 설정하면 전송되는 데이터가 DMA 제어기를 잠시 거쳐서 나가는 비 플라이바이 모드가 되며(메모리↔메모리 전송에서는 필수적이다), 'Single Cycle'로 설정하면 플라이바이 DMA 모드가 된다. 이를 위해 I/O 장치 전용의 SDACK# 신호를 제공하는데 이는 I/O 장치 전용 신호가 없는 메모리 맵 구조의 시스템에서 플라이바이 DMA를 위해 두개의 장치를 동시에 접근할 수 있는 방안의 한 사례를 보여준 것이라 할 수 있겠다.

860의 DMA는 메모리 테이블(고속으로 접근 가능한)을 이용하여 전송할 메모리 영역에 대한 정보를 지정한다는 점에서 DMA 데이터 전송 기법의 유연성을 제공해 준다는 점이 플라이바이 DMA를 위한 별도의 신호를 채용한 기법과 함께 바람직한 특징으로 꼽힌다.

2.3 Strong ARM SA-1110의 DMA

SA-1110[4]은 32 비트 Strong ARM RISC 마이크로프로서의 코어를 내장한 대표적 임베디드 프로세서이다. SA-1110의 DMA는 통신 목적과 LCD 제어기 전용으로 국한되어 일반적인 I/O 장치는 지원하지 않는다[4]. 통신장치는 총 5개의 유니트로 가동되는데 각 DMA의 채널은 이 유니트들에 대해 보내기 혹은 받기 용도로 설정하여 사용한다. 이 프로세서에서는 DMA 전송에 필요한

4) ARM은 임베디드 시스템에 많이 활용되는 프로세서이나 DMA 제어기는 프로세서 개발사마다 다양한 모습을 갖고 있다. SA-1110의 DMA는 일반 I/O 장치를 다루는 기능은 없다.

정보를 DMA 내의 레지스터에 담고 있다. DMA 제어기 내의 레지스터를 그림 3에 보였다.

DDARn중 주소필드는 직렬통신 장치의 주소를 지정하고, DBSAn은 전송할 메모리 영역의 시작지점을 설정한다. DDARn의 나머지 비트는 전송의 방향(메모리↔통신장치), I/O장치의 엔디안(Endian) 구조, 버스트 크기(연속적인 전송), 포트의 크기(8/16비트) 등의 정보를 지정한다. 이 DMA는 전송 정보를 담고 있는 버퍼를 채널 당 2개(A, B)를 사용하고 있는데 버퍼 A의 전송이 끝나면 자동적으로 버퍼 B의 전송이 계속된다. DCSRn은 전송 개시여부, 어떤 버퍼(A, B 중)가 사용되고 있는지, 전송이 종료될 때 인터럽트를 발생하게 할 것인지 등의 정보를 설정하거나 확인할 수 있다. DBTAn, DBTBn는 현재 전송중인 회수를 말한다.

SA-1110의 DMA는 프로세서의 주 응용분야인 통신 시스템에는 적합한 구조를 갖고 있으나, 데이터가 일단 DMA 레지스터의 내부에 들어가는 비 플라이바이 방식이며, 전송 정보를 담고 있는 레지스터가 단 2개만 준비되어 있고, 이들의 호출방식이 단조로워서 가변 해상도를 갖는 영상 데이터의 실시간 전송에는 부적합하다.

2.4 TMS320Cxxx의 EDMA

TMS320C6xxx[5]는 DSP이지만 임베디드 프로세서처럼 여러 제어기 및 I/O 장치를 내장하고 있고, 처리속도가 우수하기 때문에 음성을 포함한 신호처리용 임베디드 시스템에 많이 활용되고 있다. TMS320C6xxx의 E(Enhanced)DMA는 그림 4에서 보인 바와 같이 내부적으로는 EMIF, HPI, 주변장치가 L2 memory와 연결된 구조를 가진다. 따라서 CPU 코어가 L1 캐시 혹은 L2 메모리의 내부 메모리에 접근하고 있을 때는 외부 주변 장치, HPI, 외부 메모리 간에는 자유로운 데이터 전송이 이루어지는 장점이 있다. 이 계열의 DSP는 메모리 맵 방식의 I/O를 지원하지만 320C64X에서는 DMA 사이클에서 I/O 장치 전용의 제어신호(PDS)를 이용해 읽기 전용 혹은 쓰기 전용의 I/O 장치에 대해 명령신호를 제공함으로써 DMA 제

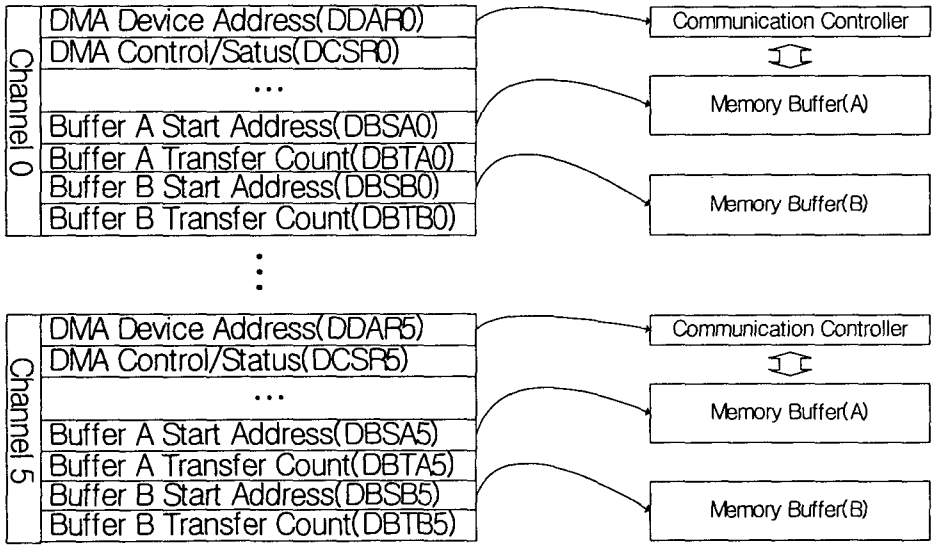

Fig. 3 DMA Registers of SA-1110

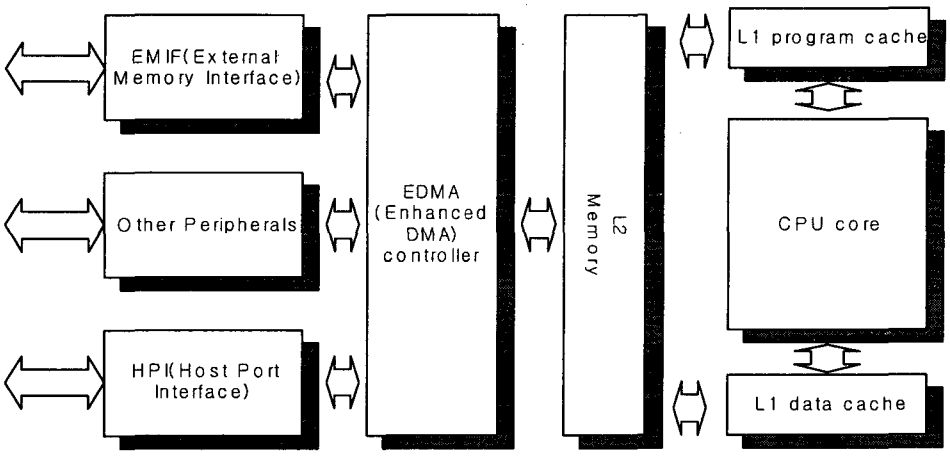

Fig 4. DMA Sub-System of TMS320C621x/C671x/C64x

어기를 거치지 않는 데이터 전송(플라이바이 DMA)를 지원한다.

EDMA의 데이터 전송은 그림 5에 보인 바와 같이 EDMA 안에 존재하는 내부 파라미터 램(PaRAM, 2KB)의 PS(Parameter Storage)라고 불리는 메모리의 데이터 블록의 정보에 따라 수행된다. 이 각각의 PS는 각 채널에 할당되어 되어 동시에 사용되며, 나머지 영역은 LPS(Link Parameter Storage) 영역으로 쓰인다. LPS는 전송이 끝난 후 다른 영역을 전송하고자 할 때 링크되어 사용되는 PS로서 C621x와 C671x의 경우 68개, C64x의 경우 20개가 제공된다.

PS는 C621x와 C671x의 경우 16개, C64x의 경우 64개를 갖고 있으며, 각 DMA 채널마다 하나의 PS가 할당되어 있다. DMA 전송을 위해서는 대

31 16	15 0	
Options (OPT)		Word 0
SRC Address (SRC)		Word 1
Array/Frame Count(FRMCNT)	Element Count(ELECNT)	Word 2
DST Address (DST)		Word 3
Array/Frame Index(FRMIDX)	Element Index(ELEIDX)	Word 4
Element Count Reload(ELERLD)	Link Address(LINK)	Word 5

Fig. 5 Parameter Storage for an EDMA Event

다수의 DMA 전송에 있어서 기본적인 정보, 즉 데이터 전송의 시작지점(SRC)과 목표지점(DST)과 전송할 데이터의 수량에 관한 정보가 있게 마련인데 이 경우에는 다양한 방법의 데이터 전송모드를 지원하기 위하여 ① 엘리먼트(Element : 전송의 최소단위로, 옵션 필드에서 8, 16, 32 비트 중 하나로 정함), ② 프레임(Frame : ELEIDX 만큼 떨어진 엘리먼트의 집합), ③ 어레이(Array : 연속된 엘리먼트의 집합), ④ 블록(Block : FRMIDX 만큼 떨어진 어레이의 집합)과 같은 데이터 전송의 단위를 정하고 있다. 이 시스템에서도 'I/O↔메모리' 간의 전송뿐 아니라 '메모리↔메모리' 간의 데이터 전송도 가능하다. PS의 옵션 필드에서는 이러한 다양한 데이터의 단위를 시작과 목표지점에 따라 메모리 블록을 1D 혹은 2D 배열로 전송할 것을 결정할 수 있고, DMA 이벤트를 프레임 단위로 할지 아닌지를 정할 수 있어서 표 2와 같이 크게 8가지의 전송모드를 지원한다(여기서 Sync. 항목은 해당 항목의 전송을 마치고 이벤트가 발생해야 전송이 시작됨을 의미한다.). 또한 당연히 I/O 장치에 대해서는 번지를 고정할 수 있도록 하는 비트 필드를 갖고 있다.

Table 2. EDMA Transfer Mode of TMS320C6XXX

Mode No.	Synchronization (Triggering Event)	Source	Destination
1	Element	1D	1D
2	Frame	1D	1D
3	Array	2D	2D
4	Block	2D	2D
5	Array	1D	2D
6	Block	1D	2D
7	Array	2D	1D
8	Block	2D	1D

그림 6에 1번 모드, 즉 'Element Synchronized 1-D to 1-D Transfers'의 한 사례를 보였다. 그림은 프레임 3개(0,1,2), 엘리먼트 4개(1,2,3,4)의 비연속한 시작지점의 영역 데이터를 전송하는 사례이다. PS에서 SRC, DST를 취득하여 시작지점, 목표지점의 번지정보를 알아내면, 이 영역간에 데이터 전송이 ELECNT(=ELDRLD)만큼 이루어진다. 이벤트가 들어오면 EIX에 지정된 간격만큼 떨어진 영역의 데이터 전송이 이루어지는데 4개의 엘리먼트 전송이 끝나고 난 후에 이벤트가 들어올 경우에는 FIX에 지정된 간격만큼 떨어진 영역의 데이터 전송이 이루어진다.

EDMA도 대다수의 DMA 제어기에서처럼 DMA 전송이 끝나면 해당 레지스터를 통하여 인터럽트를 발생하게 할 수 있다. PS의 옵션 중 TCC (Transfer Complete Code)는 전송을 모두 완료한 후 채널을 지정하여 다른 채널에 이벤트를 보내주는 일을 할 수 있다. 또한 PS의 옵션 필드에서는 모든 전송이 끝나지 않았더라도 각 이벤트에 의한 전송이 끝날 때마다 인터럽트를 발생하게 설정하는 정보도 담고 있다. 이를 통해 한 어레이의 전송이 끝나기 전에 각 프레임 단위에서 인터럽트를 발생하게 할 수도 있다. 또한 이 옵션 필드에는 한 이벤트에 의한 한 번의 전송이 완료되면 자기 자신의 PS를 다시 이용할 것인지 다른 PS를 불러 사용할 것인지를 결정하는 비트가 있어서 다양한 모습의 데이터 전송을 가능하게 하고 있다.

2.5 Bt878/848

Bt878/848[6]은 영상신호를 디지털로 변환하여 이를 PCI(Peripheral Computer Interconnect) 버스 통해 DMA로 시스템(내부 RAM 혹은 VGA의 비디오 메모리)에 전달하는 PCI 디바이스를 내장한 비디오 디코더이다. PCI 버스는 과거 ISA와는 달리 시스템 레벨에서 DMA를 제공하지는 않고 단지 버스의 사용권한을 DMA 제어기에 양도한다[7]. Bt878/848은 자체의 DMA 제어기로 영상/음성 데이터를 PCI 버스를 통해 시스템에 실시간으로 전달한다. 이 DMA 제어기의 커다란 특징은 자체에 프로세서(RISC)를 갖고 있다는 것이다.

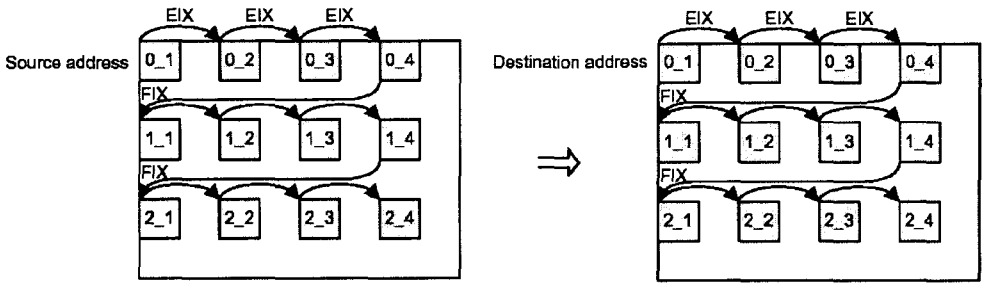

Fig. 6 An Example of Element Synchronized 1-D to 1-D Transfers

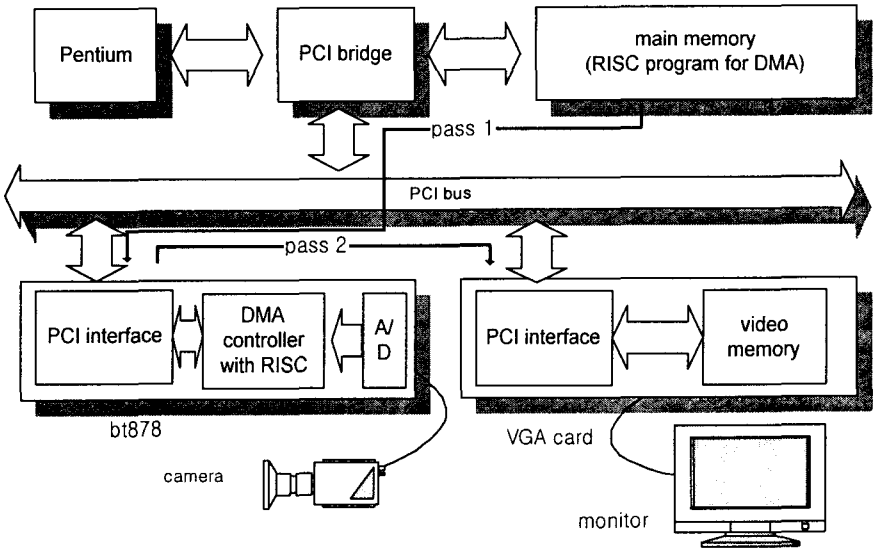

Fig. 7 Block Diagram of Bt878/848 DMA Transfer

그림 7에서 보듯 DMA 제어기용 RISC 프로그램은 펜티엄 프로세서의 일반 프로그램처럼 주 기억장치에 위치한다. DMA의 RISC는 패스 1의 경로를 따라서 주 기억장치에서 전송할 정보를 담고 있는 명령어를 가지고 와서 지정된 대로 데이터 전송을 수행한다. 영상 데이터는 주 기억장치로 보내질 수도 있지만 패스 2처럼 바로 비디오 메모리로 보내져 비디오 오버레이(overlay)를 실현할 수 있다.

RISC의 명령어는 전송할 목표 메모리의 번지, 데이터의 전송량 등의 정보가 담겨져 있는데 보통 1개의 명령어로 화면의 수평주사 1줄의 영상 정보를 보낸다. Bt878/848안에는 이 1개의 수평 영상을 저장할 수 있는 FIFO가 내장되어 있다. 프로그램은 동적으로 변경 가능하여, 화면의 크기 조정을 동적으로 변경할 수 있다. 프로그램은 다수(수직 주사선의 길이)의 Write 명령어로 구성되는데 반복적인 수행을 위해 Jump 명령어를 제공하여 데이터 전송을 반복하게 함으로써 비디오 오버레이를 구현하였다. 아래에 오버레이를 수행하는 RISC 프로그램의 사례를 보였다.

```
// even field가 될 때까지 대기하는 명령어
SYNC VRE // even field만 취득하는 경우.
// 유효 데이터가 들어올 때까지 기다리는 명령어
SYNC FM1[FM3]
WRITE .. // 수평선 한 줄의 데이터 전송 명령어
.        // 다음 줄 전송
JUMP 맨 앞  // overlay를 위한 명령어.
// 마지막 줄 전송이 끝나면 다시 전송 시작
```

주목할 만한 것은 DMA 처리기가 단순히 데이터의 전송만을 담당하는 것이 아니라 데이터의 가공한다는 것인데 여기서는 유효한 영상 데이터의 시작을 골라내는 명령어(SYNC)가 준비되어 필드(Field)와 필드사이의 불필요한 데이터 전송을 막는 기능과 YCrCb(휘도 및 색채 신호의 디지털 값) 평면을 필요에 따라 각각의 평면으로 별도로 전송하는 모드를 지원함으로써 CPU가 불필요하게 전송된 데이터를 처리하거나 데이터 평면을 재 정비해야하는 데에 따른 성능저하를 방지하였다. 이러한 특징은 멀티미디어 데이터를 처리하는 DMA 제어기의 성능 극대화의 한 방안을 보여주는 좋은 사례라고 할 수 있다.

Table 3. Review of DMA Functions and Features

DMA \ Items	① Flexibility		② Fly-by DMA	③ Continuous DMA Calling	④ Data Manipu-lation	⑤ Classifi-cation
	Adoption of Processor	Max. No. of Information Buffer per Ch.				
ISA 8237	×	1	○(I/O↔mem) ×(mem↔mem)	For the Same Area	×	1st Generation
MPC 860 IDMA	×	192	○	Multiple Areas	○(Endian , Port Size)	2nd Generation
SA-1110 Strong ARM	×	2	×	For the 2 Area	○(Endian , Port Size)	2nd Generation
TMS320C6xxx EDMA	×	68	○(When Read or Write Mode Only , I/O↔mem) ×(mem↔mem)	Multiple Areas, Multiple modes	○ (Data Size)	3rd Generation
Bt878/848	○ (Program in Main Memory)	× (Unnecessary)	○	Realizable by Program	○ (Data Size, Field Detection, YCrCb Plane Separation)	4th Generation

이와 같이 프로세서를 동반한 DMA[5]를 갖춘 범용 PCI 디바이스로는 PLX9054[8]를 들 수 있다. 이 디바이스도 자체적인 RISC 프로그램에 의해 다양한 형태의 데이터 전송 기법을 구현하고 있다. 그러나 이 디바이스는 일반적인 PCI 인터페이스를 위한 것이기 때문에 Bt848/878처럼 데이터를 검출하거나, 제거해내는 기능은 갖고 있지 않다. 그러나 8비트, 16비트 I/O 장치의 데이터를 32비트로 모아서 메모리에 전송하는 'Scatter & Gathering' 기능은 유연한 I/O 장치 인터페이스 설계 아이디어를 제시해 주고 있다.

3. DMA 구현 기법에 대한 일반화

지금까지 살펴본 DMA의 구현 기법은 각자의 응용분야에 적합한 사양을 가지고 제시된 것이라는 사실을 가만하여 여기서는 DMA 데이터 전송 성능과 유연성 측면 그리고 멀티미디어 데이터 처리의 적합성의 관점에서 이들 DMA 구현 기법을 한 단계 일반화하여 검토해 보기로 하자.

표 3에 DMA 구현 기법을 일반화된 지표로 검토하였다. 항목 ①의 유연성은 DMA 전송방법을 얼마나 다채롭게 할 수 있는가를 살펴본 것이다. 전송할 버퍼에 대한 정보를 취하는 방법으로 프로세서를 이용한 방법과 정보버퍼(Information Buffer)를 이용한 방법이 있다[6]. 정보버퍼를 사용하는 IDMA와 EDMA에서 IDMA의 버퍼의 수량이 많아 더욱 다양한 모습의 데이터 전송이 가능한 듯 보이지만 EDMA는 각 정보버퍼와의 차이 값을 지정하는 파라미터가 있어서 규칙적인 번지가 증가가 필요한 영상 데이터 전송에는 더 적합하다. 항목 ②는 시스템이 분리된 I/O 체제일 때 이상적이다. 하지만 IDMA와 EDMA는 메모리 맵

5) PCI 시스템에서는 이를 마스터 DMA라 한다.

6) 유연성 측면에서는 프로세서에 의한 방식이 우수하다. 그러나 실제 적용에 있어서는 가격/성능비, 응용분야를 고려해서 결정해야 할 할 것이다.

방식임에도 별도의 신호를 제공하는 방법을 통해 플라이바이 DMA를 구현하고 있다.

항목 ③은 정보버퍼의 수량과 관계되지만 8237을 제외하면 모두 어떤 형태로든 다른 버퍼로의 전송작업을 자동으로 호출하고 있다. 항목 ④는 멀티미디어 데이터 처리를 지향하는 DMA 제어기가 가장 역점을 두고 개선되어야 할 항목이다. 일반적인 프로세서의 DMA는 포트의 데이터를 재정렬해서 전송하거나 엔디안의 구조를 맞춰주는 정도의 작업을 담당하고 있으나, Bt878/848은 비디오 디코더에서 오는 영상의 시작 지점에 관한 정보를 살펴보아 데이터의 전송이 필요한 상황인지 아닌지를 DMA 스스로 판단한다. 이 기능은 일단 전송해서 버리는 방식을 취하거나, CPU가 일일이 점검하는 방식으로 해결할 때 생기는 전체적인 시스템 성능 저하를 방지하는 큰 장점이 있다. 또한 연속된 3개의 데이터(YCrCb)를 각각의 데이터 평면(Y, Cr, Cb)으로 골라서 보내주는 기능을 갖고 있어서 CPU의 부담을 최소화하는 데 기여하고 있다.

이상과 같은 평가지표를 통해 항목 ⑤에서 각 DMA 제어기를 멀티미디어 데이터를 처리하는 데 유리한 단계의 관점에서 분류하였다. 단순히 자신의 레지스터에 지정된 대로 데이터를 전송하는 DMA를 1세대, 다수의 전송정보를 정보버퍼(RAM 혹은 레지스터)에 갖고 다수의 메모리 영역에 전송하는 기능을 가진 DMA를 2세대, 여기에 규칙적인 간격으로 벌어진 메모리 블록에 데이터를 무한으로 보낼 수 있는 기능을 추가한 DMA를 3세대, 자체에 프로세서를 갖고 프로그래머가 원하는 다양한 데이터 전송을 실현할 수 있는 DMA를 4세대로 분류하였다.

4. 멀티미디어 데이터 처리를 위한 데이터 검출 및 제거 기능

멀티미디어 데이터를 처리하는 임베디드 시스템의 사례는 이제는 산업용, 가정용을 망라하고 다양한 분야에서 접촉할 수 있게 되었다. 특히 영상 데이터의 경우에는 용량이 방대하여 임베디드 시스템의 CPU에 부담을 많이 주는 것이 일반적이었다[7]. CPU가 이러한 영상 데이터를 가공(예: 압축, 복원)하는 사례도 있지만, 고성능/저가격을 실현하기 위해서는 CPU는 더 소형/경량화되어야 하고 가공하는 기능은 다른 H/W 소자가 처리하게 하는 것이 시장의 추세라고 할 수 있다. 따라서 멀티미디어(특히 영상) 데이터 전송을 위한 효율적인 DMA 서브 시스템의 개발이 점차로 중요한 문제로 자리잡아 가고 있다.

그림 8에 영상 데이터를 DMA로 전송하는 시스템의 사례(3가지)를 보였다. 그림에서 사례 1은 영상데이터를 디지털로 변환하여 이를 그대로 시스템의 메모리에 전송하는 경우이다[8]. 사례 2는 디지털로 변환된 데이터를 MPEG 압축 장치를 통해 압축하여 이를 메모리로 전송하는 경우이다. 사례 3은 JPEG으로 압축하여 전송하는 경우이다.

이렇게 영상 데이터의 대량 전송이 발생하는 응용분야를 들어보면 영상 데이터를 압축하여 HDD로 저장하는 DVR(Digital Video Recoder), 저장과 동시에 압축된 영상을 HDD로부터 전송받아 이를 화면에 표시하는 기능이 요구되는 PVR(Personal Video Recoder), 영상을 압축하고 압축된 영상 데이터를 전송하는 네트워크 카메라, JPEG 영상 데이터를 플래시 메모리에 저장하는 디지털 카메라 등이 있다. 이들 응용 제품은 사례 1, 2, 3 중의 적어도 어느 하나 이상을 이용하고 있다.

---

7) 특히 다채널 영상을 다루는 경우에는 비디오 장치와 메모리간의 데이터 전송 문제가 CPU의 처리속도 문제 못지않은 중요한 문제로 부각된다.

8) 이러한 시스템은 CPU가 직접 메모리의 데이터를 처리(예를 들어 압축)하는 경우이다.

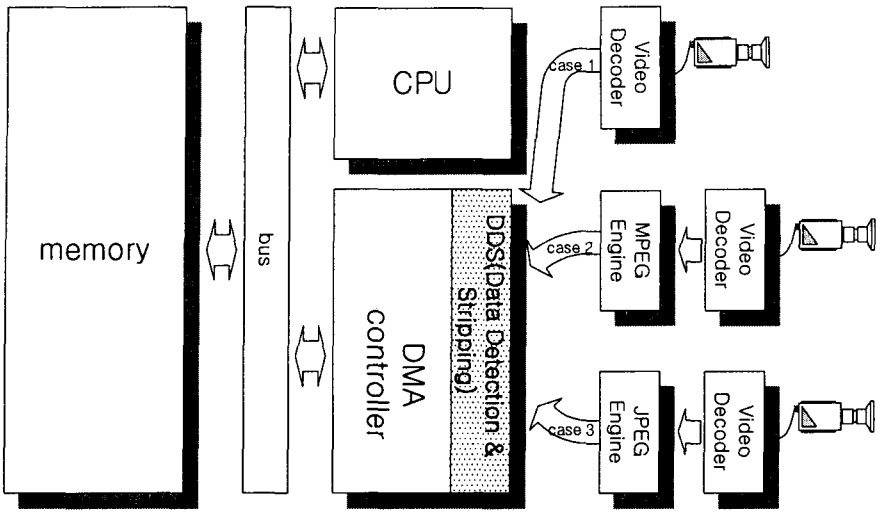

Fig. 8 DMA Architecture for Video Data Processing

DMA 서브시스템의 성능 제고를 위해서는 표 3에서 전술한 ① 데이터 전송 방법의 유연성, ② 플라이바이 DMA, ③ DMA 연속 호출, ④ 데이터의 가공 기능을 기본적으로 유지해야 한다. 멀티미디어 데이터에 대해서는 ④의 가공 기능 중에서 DDS(검출 및 제거) 기능이 있어야 한다. 아래에 이와 같은 멀티미디어(특히) 영상 데이터의 데이터 전송에 효율적인 DMA 제어기가 가져야 할 DDS 사양을 제시하였다.

1) CCIR 656 디코더 데이터 처리(사례 1)

CCIR 656[9]은 카메라를 컴퓨터 시스템에 편리하게 접속하기 위하여 카메라에서 출력한 아날로그 비디오 영상을 디지털로 변환한 것이다. 과거의 비디오 디코더는 디지털 영상 데이터와 함께 그 유효성을 표시하는 싱크 및 상태 신호를 함께 출력했으나, CCIR 656의 데이터에는 순수한 디지털 데이터 안에 영상의 시작, 필드(짝수/홀수)를 알리는 싱크(Sync) 정보와 여러 상태 정보가 영상 데이터와 함께 내장되어 있다[9].

9) 최근의 비디오 디코더는 종래의 제어/상태 신호 방식과 CCIR 656 양식을 모두 지원하고 있다.

사례 1과 같은 구조의 시스템에서는 디코더를 통해 들어오는 영상 데이터를 별도의 프레임 메모리 없이 CPU의 주 기억장치 공간으로 전송하는 구조를 갖고 있어야 설계의 단순화와 시스템의 성능을 고도화를 꾀할 수 있다. 이를 위해서는 인입되는 데이터 중에서 순수한 영상 데이터가 들어올 때를 가려낼 수 있는 기능이 있어야 하는데 이는 다음과 같은 CCIR 데이터의 분석을 통해 가능하다.

한 줄의 스캔라인이 시작되기에 앞서 타이밍 참조 코드(Timing Reference Code)가 시작되는데 이는 'FF0000'로 표시된다. 4번째 오는 정보에는 필드정보, 필드 블랭킹 여부, SAV(Start of Active Video)/EAV(End of Active Video) 여부 등의 정보가 있다. DDS 부에서는 이러한 코드를 가려내어 지금 유효한 영상 정보가 들어오는지, 짝수 필드인지 홀수 필드인지 등의 정보를 파악할 수 있다. 파악된 정보는 DMA 제어기로 하여금 영상 정보를 어떻게 처리할 것인지를 판단하는데 활용한다.

CCIR 656의 데이터 형식은 YCrYCbYCrYCb... 순으로 반복된다. 따라서 들어오는 데이터를 Y

평면, Cr 평면, Cb 평면으로 분리해서 전송하면 휘도 성분에 대한 영상 해석, 영상 데이터의 압축 등에 더욱 고속화 처리를 기할 수 있다.

2) MPEG 데이터 처리(사례 2)

MPEG 엔진을 통해 영상 데이터를 입수하는 사례의 경우 CPU는 MPEG 데이터를 메모리 혹은 HDD(DVR, PVR 응용 사례의 경우)에 전송하는 부하를 담당해야 한다. 이때 압축된 MPEG 영상 데이터는 시퀀스 헤더(0000 01B3 0200 1014 FFFF E0A0)와 시퀀스 종료 코드(0000 01B7)로 둘러싸여 있다. 이 시퀀스 헤더의 검출과 함께 시작코드(startup code:0000 01B3)를 검출하는 DDS 기능을 DMA 제어기가 보유하면 유효한 하나의 데이터 스트림의 시작부터 전송이 가능하므로 데이터 전송량과 CPU의 부담을 함께 경감시킬 수 있다.

3) JPEG 데이터 처리(사례 3)

JPEG으로 압축된 영상의 시작은 SOI(Start Of Image:FFD8)로 시작하고, EOI(End Of Image: FFD9)로 끝을 맺는다. SOI와 EOI 내부에 프레임들이 존재하는데 프레임 데이터는 SOF(Start Of Frame)로 시작하고 그 뒤에 프레임 헤더가 붙는다. 프레임 헤더는 프레임 헤더에 관한 여러 가지 파라미터(샘플링 바이트수, 줄 수, 줄 당 화소수, 샘플링비, 양자화 테이블 번호 등)가 포함되어 있다. SOF는 FFCx로 시작되는데 DDS에서 이러한 SOI, SOF 코드를 검출해서 그때부터 전송을 시작하게 하면 불필요한 데이터의 전송을 막을 수 있다.

5. 결 론

이상으로 현존하는 DMA 제어기/서브시스템에 대한 분석을 통해 DMA 제어기가 가져야할 기능을 도출하였고, 더불어 멀티미디어 데이터 처리를 위해 바람직한 사양을 제시하였다. 비록 제시된 사례들이 가장 이상적인 DMA 제어기의 성능을 담고 있지는 못하더라도 일반적으로 많이 쓰이는 프로세서/시스템을 대상으로 검토하였으며, DMA 제어기의 사양을 멀티미디어 환경에 대한 유연성의 측면을 기준으로 4 개의 세대로 분류해 보았다. 만약 제시된 바와 같은 멀티미디어 데이터 처리 능력(DDS)이 추가된다면 5세대 DMA라 칭할 만 할 것이다. 분류된 세대가 시대적 발전 역사와는 직접적인 관련이 없는 이유는 DMA 제어기가 자신의 응용 사례에 맞추어 가격/성능 면에서 최상의 기법을 선택한 했기 때문이며, 본 분류에서는 성능 구현을 위한 비용측면은 고려하지 않았기 때문이다.

끝으로 이 논문이 임베디드 프로세서의 개발자에게 자신 개발하고 있는 프로세서의 응용가치를 드높이는 하나의 아이디어가 되어 주기를 바라며 이 글을 마친다.

참 고 문 헌

[1] 김진헌, "PC 내부 구조 및 응용", 대광출판사, 2001.
[2] Tom Shanley/Don Anderson, "ISA System Architecture" 3th. ed., 1995.
[3] "MPC860 PowerQUICC User's Manual", Motorola, 1998.
[4] "Strong ARM SA-1110 Microprocessor- developer's manual", Intel, 2000.
[5] "TMS320C6000 Peripheral Reference Guide", Texas Instruments, 2001.
[6] "Bt848 Single Chip Video Capture for PCI - Reference Manual", Brooktree Corp., 1996.
[7] Don Anderson/Tom Shanley, "PCI System Architecture, 4th ed.", Addison Wesley, 1999.
[8] "PCI 9054 Data Book", PLX Tech., 1999.
[9] Rec. 656, The International Radio Consultative Committee, International Telecommunications Union,, 1990

김 진 헌

- 1982년 고려대학교 전기공학과(공학사)
- 1984년 고려대학교 대학원 전기공학과(공학석사)
- 1994년 고려대학교 대학원 전기공학과(공학박사)
- 1986년 9월~1989년 9월 삼성종합기술원(선임연구원)
- 1995년 3월~현재 서경대학교 컴퓨터공학과 교수
- 2002년 1월~현재 (사) IBS Korea 이사
- 관심분야 : 디지털 영상처리, 임베디드 시스템 영상처리 H/W 시스템
- 저서 : 어셈블리 프로그래밍(대광출판사), PC 내부구조 및 응용(대광출판사)