논문 02-27-4C-10 한국통신학회논문지 '02-4 Vol.27 No.4C

병목 현상 제거 및 효율적인 PCI 회로 설계에 관한 연구

정회원 이 인 섭*, 강 정 용*, 김 환 용*

A study on the Circuit Designed for Bottle-neck Rejection and Effective PCI

In-sup Lee*, Jeong-Yong Kang*, Hwan-yong Kim* ***Regular Members***

요 약

본 논문에서는 외부 영상에서 다중 처리된 디지털 신호를 PCI로 전송할 수 있는 시스템을 설계하였다. CPU와 주변기기들의 전송 율 제한에 따른 병목 현상을 개선한 것으로 실시간으로 처리되는 영상 데이터에 대하여 효율적으로 전송 및 제어할 수 있는 구조를 제안하였다. 또한 PCI로 빠른 데이터 전송 및 DMA 기능으로 자체적인 부하 사용량을 13% 줄였다. 설계는 Max+plus II를 이용한 기능 및 타이밍에 대한 동작 검증을 하였다.

ABSTRACT

In this paper external image multi-processing digital signal the transmit is the possibility of doing system with the PCI the design. The bottle-neck which it follows in transmission ratio limit of the CPU and the circumference machineries and tools against the image data which with the improve one thing becomes the processing with the real-time efficiently the transmit and the control is the possibility of doing structure the proposed. The also with the resource amount used 13% reduced which PCI fast data transfer and DMA function. The designed is operation verification against the function and the timing which use Max+plus II.

Ⅰ. 서 론

영상을 카메라로 전송 받아서 감시국에서 이를 모두 관리하는 시스템의 경우, 각 원격지의 영상을 쉽게 관리하고 감시할 수 있도록 화면을 분할하여 하나의 영상으로 보여주는 것이 효율적이다. 하지만 실시간 영상 재생의 경우, 해상도와 색상수가 증가함에 따라 데이터 전송량이 증가하게 되는데 예를 들면 1280× 1024 픽셀, 30프레임, 16bit 데이터를 실시간으로 재생하기 위해서는 78.64 Mbytes/sec의 전송 속도가 필요하게 된다.

그러나 기존의 ISA(Industry Standard Architecture) 버스 방식은 16Mbyte/sec로 많은 데이터를 전송할 수 없는 실정이다.[1]-[5]

이러한 문제를 극복하기 위해서 32bit 데이터 전송 폭을 지닌 EISA(Extended ISA), MCA(Micro Channel Architecture) 형태의 버스가 연구되었으나 전송 능력이 주변기기에 미치지 못하여 근본적인 해결책이 되지 못했다. 이러한 문제점은 Windows에서 GUI(Graphical User Interface) 운영체제가 사용되어 CPU와 주변기기의 직접적인 데이터 통신이 가능한 PCI(Peripheral Component Interconnect)로컬 버스가 연구되었다. 표1은 버스 종류에 따른 성능 비교를 나타내었다.[6]-[8]

* 원광대학교 전자공학과(insup_0113@yahoo.co.kr, rkdwjddyd@yahoo.co.kr, hykim@wonkwang.ac.kr)
논문번호 : #020027-0121, 접수일자 : 2002년 1월 28일
※ 이 논문은 2001년도 원광대학교의 교비지원에 의해서 연구됨

표 1. 버스 종류에 따른 성능 비교

항목 \ 종류	ISA	VESA	PCI
대역폭 [bits]	16	32/64	32/64
주파수 [㎒]	8	33	33/66
전송속도 [MB]	16	124/264	132~528

고속으로 동작하는 마이크로 프로세서는 하드디스크, 비디오 보드 등 주변 장치들과 데이터를 입출력할 때 많은 지연 시간으로 전체 시스템 성능을 저하시키게 된다. 본 논문에서는 입력된 여러 영상 데이터를 동일한 메모리에 저장하고 합성할 경우, 각각의 데이터를 모두 전송할 수 있도록 PCI 및 메모리 제어 회로를 설계하여 전송 율 제한에 의한 병목 현상을 개선한 것이다. 이를 위해서는 각 채널의 데이터 래치 할 때마다 플래그를 표시하고 레지스터 설정으로 원하는 데이터를 순차적으로 PCI로 전송할 수 있는 콘트롤러를 설계하였다.

Ⅱ. PCI 인터페이스 구조 및 DMA 제어

PCI 버스의 시스템 클럭은 33[㎒]이고 대역폭은 32[bit] 구조로 되어있다. 데이터의 입출력은 버스트(burst) 전송을 하므로 한 개의 시스템 클럭 사용으로 132Mbytes/sec로 데이터를 전송할 수 있고 추가적으로 32[bit] 인터페이스와 66[㎒]의 시스템 클럭을 사용할 수 있어 최대 528Mbytes/sec의 데이터 전송이 가능하다. 그림 1은 PCI 버스의 구조를 나타내었다. PCI 버스는 프로세서/캐시 메모리와 브리지를 통해 연결되는데 각 주변장치의 데이터들도 브리지를 통해서 데이터를 프로세서에 전달한다. 또한 브리지는 프로세서가 주변 장치를 직접 액세스 할 수 있을 뿐 아니라 데이터 버퍼, 마스터의 중재 역할을 수행한다.[9]~[11]

그림 1. PCI 버스의 구조

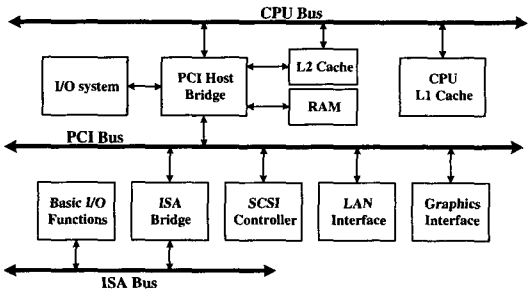

그림 2는 PCI 인터페이스 구조를 나타내었다. Application 버스에서 데이터를 PCI 버스에 쓰기 위해서는 transfer FIFO(First-In, First-Out)로 저장한다. Application 버스는 PCI 버스와 다른 클럭을 사용하므로 데이터를 PCI 버스에 동기 시킬 수 없다. PCI 버스의 특성은 현재 사용하고 있는 마스터가 버스 전송을 언제 마칠지 모르기 때문에 충분한 크기의 FIFO 사용으로 데이터를 저장하고 PCI 버스 클럭에 동기시켜 전송한다. 데이터 전송을 위한 일정양의 데이터가 FIFO에 저장되면 버스에 전송할 데이터가 있음을 알린다. 메모리의 데이터 전송을 위해서는 DMA를 사용하여 메모리에 직접 쓰게되는데 이를 위해서 application 버스에서 DMA 위한 데이터를 미리 전송하는데 데이터 크기, 전송 어드레스, 레지스터를 설정하여 출력한다. 표2는 상태 레지스터의 종류를 요약하였다.

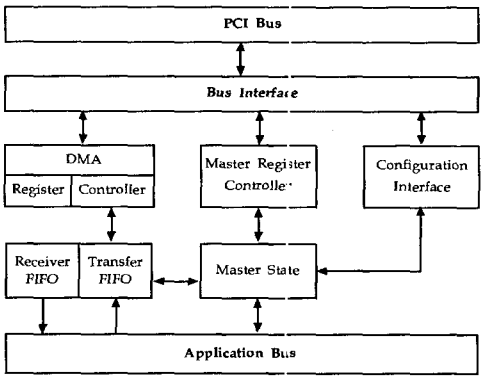

그림 2. PCI 인터페이스 구조

표 2. 상태 레지스터의 종류

레지스터 종류	설 명
Vendor ID	PCI SIG에 의해 허용된 device 제작자를 확인
Device ID	Vendor에 의해 허용된 device 형태를 확인
Revision ID	Vendor에 의해 허용된 revision 확인
Class Code	일반적인 device 기능을 확인
Command	PCI 사이클에서 응답하는 device의 동작을 제어
Status	PCI 버스 관련 상태 정보를 기록
Base	device의 Base Address를 규정

그림 3은 설계된 PCI 인터페이스 알고리즘을 나타내었다. 동작 순서는 Req 신호를 브리지에 전달했을 때 Frame 신호가 버스 사용 여부를 판단하고 버스가 사용할 수 있을 때까지 Frame 신호의 구동을 유보한다. 버스가 사용 중이 아니면 Frame과 Irdy를 동작시켜 Trdy, Irdy 신호가 각각 '0' 일 때 데이터 전송을 한다. 데이터 전송 시 내부의 데이터가 마지막 전송인지를 확인하여 버스의 사용을 중단하는데 버스가 사용 중단을 요구하는 신호인 Perr, Serr이 동작하면 버스의 상태에 관계없이 사용권한을 브리지에 반환한다.

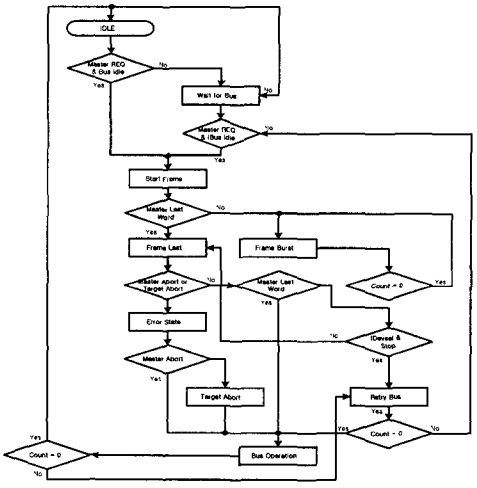

그림 3. PCI 인터페이스 알고리즘

그림 4는 DMA의 동작 알고리즘을 나타내었다. DMA의 동작 순서는 레지스터를 입력받고 전송되는 데이터가 유효한지 확인한다. 메모리에서는 데이터를 레지스터로 읽기, 쓰기를 판단하여 각각의 동작을 수행하는데 전송을 요구하면 DMA가 receiver FIFO로 데이터를 전송한다.

그림 5는 디지털 비디오 분할 시스템의 전체 블록도를 나타내었다. 구성은 FIFO 및 인코더, 메인 콘트롤, PCI 인터페이스 블록으로 나누어진다. 동작 순서는 비디오 인코더에서 입력되는 16채널의 데이터를 각각 FIFO에 순차적으로 저장한다. 저장된 FIFO의 데이터는 디바이스 드라이브와 연결된 프로그램 명령어를 통해 원하는 FIFO를 선택하여 데이터를 메모리에 저장하여 합성된 후에 PCI로 전달된다.

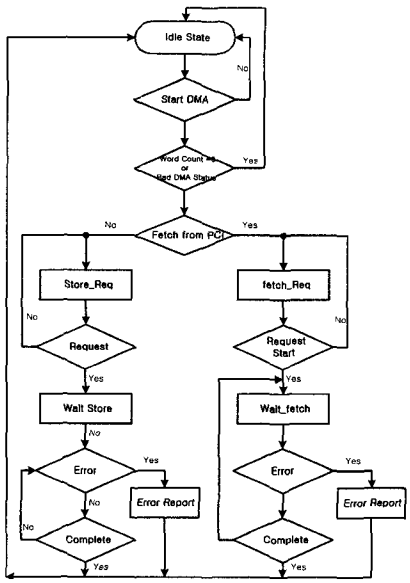

그림 4. DMA의 동작 알고리즘

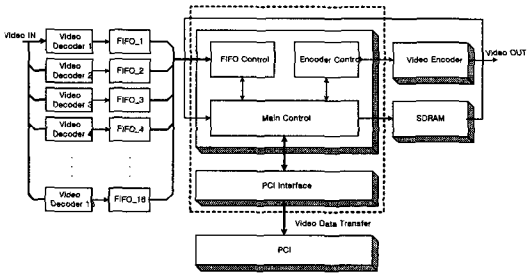

그림 5. 디지털 비디오 분할 시스템의 전체 블록도

Ⅲ. 설계 및 모의실험 결과

그림 6은 설계된 16채널 FIFO 및 디코더 제어 블록을 나타내었다. 비디오 디코더를 통하여 입력되는 데이터는 PCI에서 입력되는 모드에 따라 선택되며 Mode[1:0]의 내부 신호를 이용하여 해당 디코더와 FIFO를 동작시켜 채널 선택을 하도록 하였다.

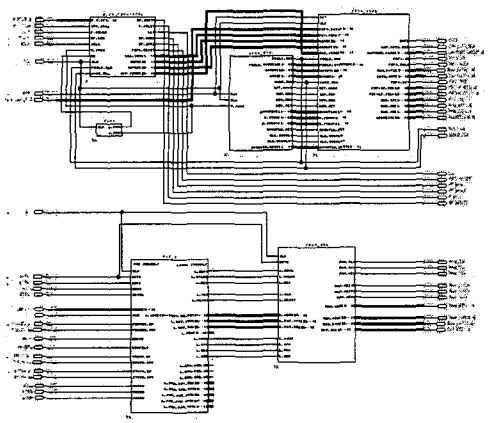

그림 6. 설계된 16채널 FIFO 및 디코더 제어 블록도

그림 7은 PCI의 상태 쓰기 동작의 모의실험 결과이다. Frame 신호가 동작되어 버스 전송이 시작되었음을 알리고 CBE[3:0]에서 상태에 대한 디바이스 어드레스를 선택하고 Devsel 신호가 구동한다. 이후 Trdy와 Irdy으로 상태가 이루어지고 다음 동작이 일어나는 클럭에서 동작이 끝나게 된다.

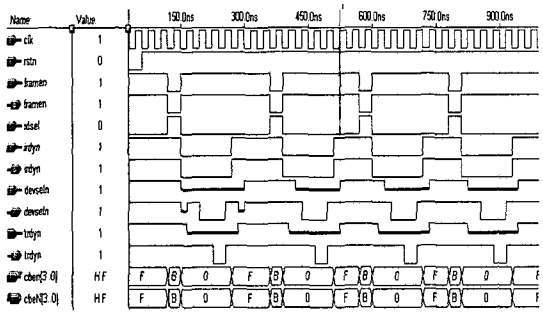

그림 7. PCI 상태 쓰기 동작

그림 8은 PCI 상태 읽기 동작을 나타내었다. 어드레스 명령어에 대해서 상태를 읽는 동작을 알리고 Irdy가 구동되어 마스터가 데이터를 받는다. 다음 클럭에서 Trdy가 구동됨으로 버스에서 요구된 데이터가 유효하다는 것을 알려주고 프레임이 끝나도록 Devsel을 제거한다. Frame 신호가 데이터 전송이 일어나는 클럭에서 구동되고 있으므로 상태 읽기가 계속된다.

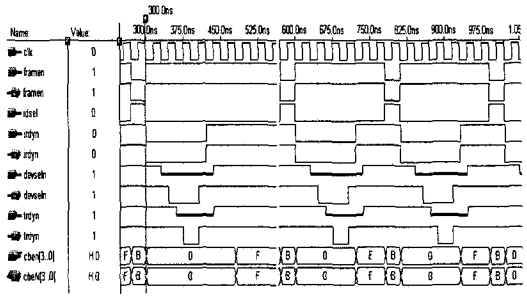

그림 8. PCI 상태 읽기 동작

그림 9는 transfer FIFO의 읽기, 쓰기에 대한 모의실험 결과이다. DMA 전송을 위해서 FIFO는 2개이나 메모리 크기만 다를 뿐 같은 구조로 되어 있다. 따라서 한 개의 transfer FIFO의 동작으로 DMA에서 PCI 버스를 통해 데이터를 메모리에 전달되는 것을 확인할 수 있다.

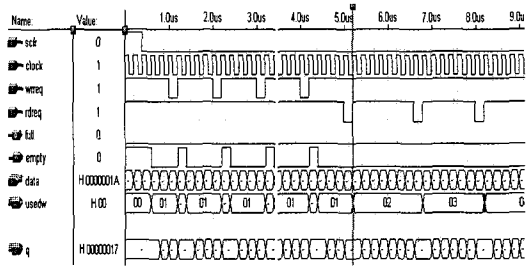

그림 9. Transfer FIFO의 읽기/쓰기 모의 실험 결과

그림 10은 DMA의 읽기 동작에 대한 모의실험 결과이다. DMA는 메모리 데이터를 읽어 외부 버스로 데이터를 전달한다. DMA 전송을 위해서는 필요한 레지스터를 각각 외부에서 입력하고 대역폭은 32[bit]로 설정하였다. 레지스터는 PCI_Addr, WordCount, command_reg, start_dma 순서로 입력하였다. 해당 레지스터는 32bit 버스와 입력을 위한 스트로브 신호(MP_wordcount_reg, MP_PCI_reg, MP_start_dma_reg)를 이용하였다. 레지스터 전송이 완료되었을 때 입력한 데이터 개수가 메모리의 데이터와 일치함을 확인하였다. 또한, 메모리 전송이 완료된 후 Frame 신호가 제거되고 Devsel, Irdy의 신호가 이전 상태로 되돌아갔음을 확인하였다.

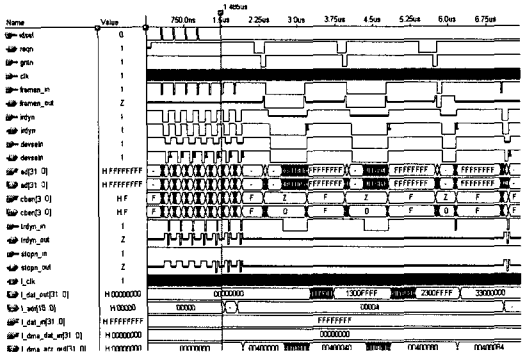

그림 10. DMA 읽기의 모의실험 결과

그림 11는 DMA의 쓰기 동작에 대한 모의실험 결과이다. 메모리의 데이터를 쓰는 동작으로 읽기와 같이 32[bit]의 값으로 설정하였다. 레지스터는 command_reg PCI_Addr, WordCount, start_dma 순서로 입력하였다. 레지스터 전송이 완료되면 입력한 개수의 데이터가 발생함을 확인하였고 AD[31:0]의 값을 비교하여 일치함을 확인하였다.

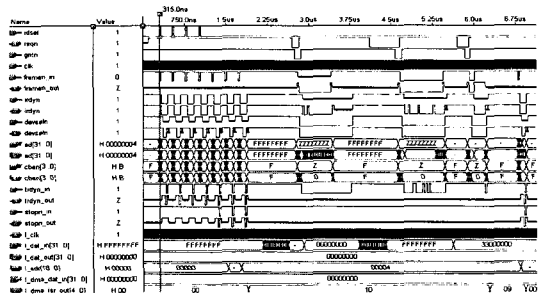

그림 11. DMA 쓰기의 모의실험 결과

그림12은 영상 전송 시스템의 모의 실험을 나타내었다. 입력된 FIFO 데이터는 전송 모드에 따라서 해당 데이터를 외부 메모리에 저장되고 순차적인 메모리 제어로 합성된 데이터가 출력됨을 확인하였다.

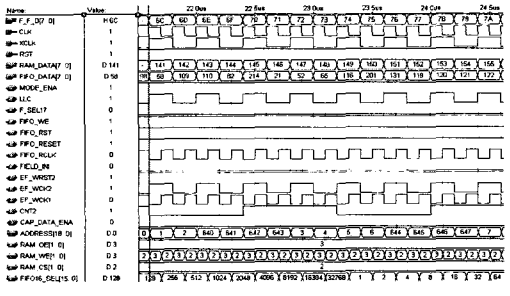

그림 12. 영상 전송 시스템의 모의실험 결과

Ⅳ. 결 론

본 논문에서는 병목 현상을 줄이기 위하여 데이터 전송 처리 조건이 가능하도록 설계하였다. 따라서 DMA 전송으로 자체적인 부하 사용량을 약 13% 줄일 수 있었다. 또한 입력된 여러 영상 데이터를 동일한 메모리에 저장하고 합성할 경우, 각각의 데이터를 모두 전송할 수 있도록 PCI 및 메모리 제어 회로를 설계하여 전송 율 제한을 해결하였다. 설계된 PCI 회로는 16채널에 640× 480 픽셀의 영상을 초당 30프레임으로 데이터를 전송할 수 있으며 사용된 디바이스는 Altera사의 EPF10K30AQC240-1 2개를 사용하였다.

참 고 문 헌

[1] Tom Shanley and Don Anderson PCI System Architecture, Addison-Wesley Publishing Company, 3rd ed., 1995

[2] PCI Local Bus Specification Revision 2.1 PCI SIG, June, 1996

[3] QAN10 PCI Using the QL2003 FPGA Data Sheet, Quicklogic, 1997

[4] CorePCITarget + DMA Data Sheet, Actel Corporation, May, 1998

[5] 32-bit PCI Bus to 32-bit Backend PCI Core Data Sheet, Mentor Graphics, Sept., 1997

[6] Dougls J. Smith, HDL Chip Design, Doone Publications 1996

[7] Keith Jack, Video Demystified : A Handbook for the Digital Engineer, High Text Interactive, Inc., 1995

[8] Edward Solari, George Willse, "PCI Hardware

and Software Architecture and Design", Annabooks, 3rd., edition 1996

[9] Wang, K., Bryant, C., Carlson, M., Elmer, T., Harris, A., Garcia, M., Hui, C.S., Leung, C.K., Reynolds, B., Tang, B., Weber, L., Wenzel, J., Wilson, G., and Becker, M., "Designing the MPC105PCI bridge/memory controller", IEEE Micro., vol., 15, pp., 44-49. 1995

[10] G. de Micheli and R.K. Gupta, "Hardware/ Software co-design", Proc., IEEE, 85(3): pp., 349-365. March 1997

[11] S. Browa and J. Rose. "FPGA and CLPD Architectures: A Tutorial", Proc., IEEE Design & Test of Computers, 13(2): pp., 42-57. Summer 1996

이 인 섭(In-sup Lee) 정회원

1997년 2월 : 원광대학교 전자공학과 공학사
1999년 2월 : 원광대학교 전자공학과 공학석사
1999년 3월~현재 : 원광대학교 전자공학과 박사과정
<주관심 분야> 영상신호처리, VLSI 시스템 설계, CAD & ASIC Design

강 정 용(Jeong-yong Kang) 정회원

1991년 2월 : 원광대학교 전자공학과 공학사
1997년 2월 : 원광대학교 전자공학과 공학석사
1997년 3월~현재 : 원광대학교 전자공학과 박사과정
〈주관심 분야〉 영상신호처리, 이동통신시스템, VLSI 시스템 설계

김 환 용(Hwan-yong Kim) 정회원
한국통신학회 논문지 제 21권, 11호 참조
현재 : 원광대학교 전자공학과 교수
<주관심 분야> 영상신호처리, 이동통신시스템, VLSI 시스템 설계, 회로 및 시스템