논문 2004-41CI-5-2

PCI 익스프레스의 데이터 연결 계층에서 송신단 버퍼 관리를 위한 효과적인 방법

(An Effective Method to Manage the Transmitter's Buffer in the Data Link Layer of the PCI Express)

현 유 진*, 성 광 수**

(Eugin Hyun and Kwang-Su Seong)

요 약

PCI 익스프레스 디바이스의 데이터 연결 계층은 전송할 패킷을 저장하고 있는 송신 버퍼와 전송하였지만 아직 타겟 디바이스로부터 승인 받지 못한 패킷을 저장하고 있는 재전송 버퍼를 가지고 있어야 한다. 이렇게 전송 버퍼와 재전송 버퍼를 구분하여 구현할 경우 전송할 패킷이 전송 버퍼에 있다 하더라도 재전송 버퍼에 여유 공간이 없다면 더 이상 패킷을 전송 할 수 없다는 단점이 있다. 본 논문에서는 한 개의 버퍼로 전송 버퍼와 재전송 버퍼를 구현하는 방법을 제안한다. 제안된 버퍼 구조에서는 필요에 따라 버퍼의 공간을 유연하게 사용할 수 있기 때문에 버퍼 사용 및 데이터 전송 효율을 향상시킬 수 있다. 모의 실험 결과 버퍼의 전체 크기가 8K 바이트일 경우 제안된 방법이 버퍼를 분리하여 사용하는 방법에 비해 데이터 전송 효율이 평균 39% 향상되었다.

Abstract

The data link layer of the PCI Express must have the transmitting buffer that contains the packets to transmit next time. Also it must have the retry buffer that contains the packets which were already transmitted but have not been acknowledged by the corresponding target device. In the separated buffer architecture, the data link layer can not transmit the packets in the transmitting buffer if the retry buffer space is not enough. In this paper, we propose an efficient buffer architecture which merges the transmitting buffer and the retry buffer to a single buffer. Since the proposed buffer can dynamically assign the size of the transmitting buffer and the retry buffer, it can improve the buffer usage efficiency and the data transfer efficiency. The simulation result shows that the proposed buffer has the higher data transfer efficiency than the separated buffer architecture about 39% when the total buffer size is 8K byte.

Keywords : PCI, PCI Express, Data Link Layer, 전송효율

I. 서 론

1992년 PCI 2.1 표준안이 만들어진 이후 지금까지 10년간 PCI 버스는 PC뿐 아니라 디지털 셋탑 박스, 통신 시스템, RAID 시스템 등 I/O 시스템의 표준안으로 다양한 곳에 사용되고 있다[1][2][3][4]. 그러나 근래에 들어와서 CPU와 메모리의 성능이 향상되고 네트웍 카드 및 디스크 컨트롤러 등을 포함한 주변장치들이 고성능화 됨에 따라 이들을 연결하는 PCI 버스가 전체 시스템의

* 학생회원, 영남대학교 대학원 전자공학과
(Dept. of Electronic Engineering, Yeungnam Univ.)
** 정회원, 영남대학교 전자정보공학부
(Dept. of Electrical Engineering and Computer Science, Yeungnam Univ.)
※ 본 연구보고서는 정보통신부 정보통신 연구진흥원에서 지원하고 있는 정보통신 기초연구 지원사업의 연구결과입니다. (University fundamental Research Program supported by Ministry of Information & Communications in republic of Korea)
접수일자: 2004년2월20일, 수정완료일: 2004년9월1일

병목 현상이 되고 있다[5][6][7][8]. PCI에 대한 이러한 문제점을 국소적으로 해결하기 위해 PCI-X 표준안이 발표되었으나 이것 역시 여러 I/O 디바이스가 하나의 버스를 공유하는 병렬버스 구조로 되어있어 성능향상이 한계에 다다르고 있다[5][6][7][8].

이러한 PCI 버스의 한계를 극복하기 위해 PCI SIG에서는 기존의 PCI를 계승하는 차세대 I/O 표준으로 PCI 익스프레스를 발표하였다[5][6][7][8]. PCI 익스프레스는 점대 점(point-to-point) 방식을 이용한 직렬 전송 방식으로 2.5Gbps의 전송 속도를 가진다. 이런 직렬 전송선 수를 최대 32개 까지 연결 가능하므로 최대 전송 속도는 32×2.5Gbps가 된다. PCI 익스프레스는 소프트웨어 관점에서 PCI와 호환되므로 기존 PCI 장치에 사용되는 운영체제와 디바이스 드라이버 소프트웨어를 그대로 사용할 수 있다. 이러한 장점들로 인해 PCI 익스프레스가 차세대 컴퓨터 I/O 시스템의 표준안으로 자리잡을 것으로 보인다[5][6][7][8].

그림 1은 PCI 익스프레스 시스템의 구성도이다. 그림에서와 같이 PCI 익스프레스 시스템은 루트 콤플렉스(Root complex), 스위치(Switch) 그리고 엔드포인트(Endpoint)로 구성된다[8]. 루트 콤플렉스는 I/O 시스템의 최상위 디바이스로 기존 PCI의 호스트 브리지(Host Bridge)와 유사한 역할을 하고, 스위치는 각 I/O 디바이스를 연결하는 PCI-PCI 브리지와 유사한 역할을 한다. 또한 엔드포인트는 PCI 익스프레스에 연결되는 디바이스로, 예를 들면 PCI 익스프레스에 장착되는 그래픽 카드를 들 수 있다. 루트 콤플렉스와 스위치, 그리고 스위치와 각 엔드포인트들은 각각 링크로 연결되어 있고, 이 링크를 통해 데이터를 전송하게 된다. 본 논문에서는 3가지의 PCI 익스프레스 디바이스 중 엔드포인트를 위한 버퍼 구조를 제안하고자 한다.

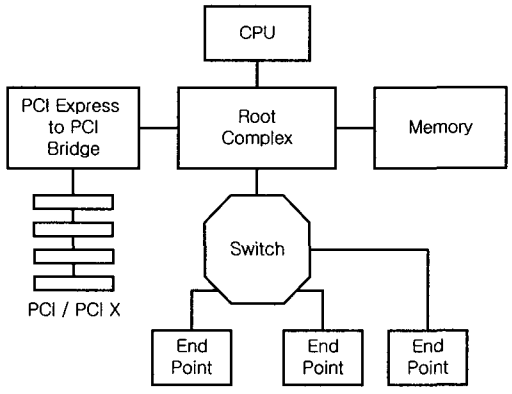


그림 1. PCI 익스프레스의 구성도
Fig. 1. PCI Express topology.

PCI 익스프레스에서는 그림 2와 같이 전송 계층(Transaction Layer), 데이터 연결 계층(Data Link Layer), 그리고 물리 계층(Physical Layer)으로 구성된 계층적 구조로 되어있다[7][8]. 각 계층은 송신단 부분과 수신단 부분으로 나누어져서 동작하게 된다.

송신단에서는 전송하고자 하는 데이터를 패킷으로 생성하여 상대 디바이스의 수신단으로 보내게 되는데 각 계층에서 생성하는 패킷의 형태는 그림 3과 같다. 최상위 계층인 전송 계층은 요청하고자 하는 명령어 등의 전송 정보를 가지고 있는 헤더, 데이터, ECRC(End to End CRC Code)로 구성된 TLP(Transaction Layer Packet)를 생성하여 하위 계층인 데이터 연결 계층으로 전송한다. 데이터 연결 계층에서는 TLP 대한 신뢰성을 확보하기 위하여 일련 번호(Sequence Number)와 LCRC(Data Link Layer CRC Code)를 붙여 하위 계층인 물리 계층으로 전송한다. 물리 계층은 상위 계층으로부터 받은 패킷의 시작과 끝을 알리기 위해 프레밍(Framing) 정보를 붙인 다음 직렬로 변환하여 외부로 송신한다.

수신단에서는 링크를 통해 상대 디바이스의 송신단으로부터 받은 패킷을 물리 계층에서 병렬로 변환한 뒤 상위 계층인 데이터 연결 계층으로 전송된다. 데이터 연결 계층에서는 받은 패킷의 일련 번호와 LCRC 코드를 확인하여 이상이 없는 경우 전송 계층으로 전송을 한다.

마스터 디바이스로부터 패킷을 받은 타겟 디바이스의 데이터 연결 계층은 오류 여부를 확인 후 이상 없는 패킷들에 대해서는 일정한 시간이 지난 후 패킷에 붙어 있는 일련 번호를 이용하여 정상적으로 승인되었음을 마스터 디바이스에 알려준다. 만약 일정한 시간이 지났는데도 타겟 디바이스가 승인되었음을 알려주지 않았다

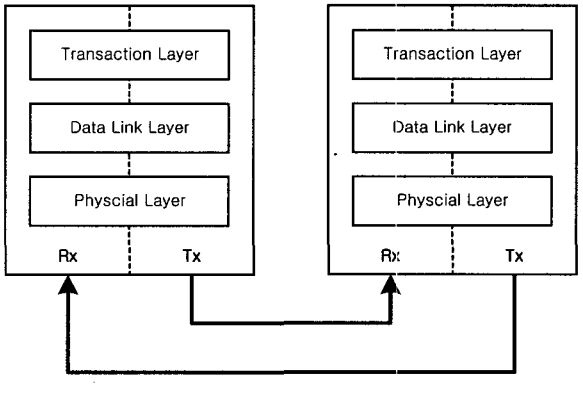


그림 2. PCI 익스프레스의 계층 구조
Fig. 2. High level layering diagram n PCI Express.

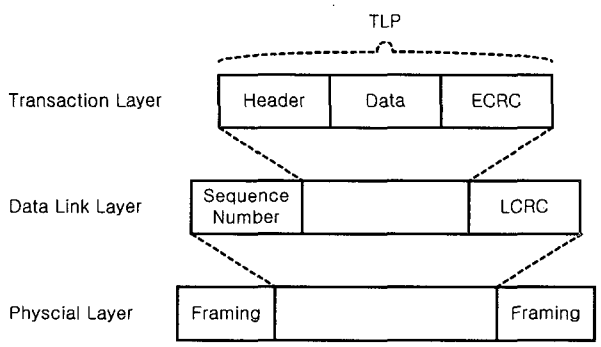


그림 3. PCI 익스프레스의 패킷 구조
Fig. 3. Packet flow through the layers in PCI Express.

면, 마스터 디바이스는 전송한 패킷 중 타겟 디바이스로부터 승인 받지 못한 모든 패킷을 다시 전송해 주어야 한다[7][8]. 이를 지원하기 위해 PCI 익스프레스 디바이스는 전송할 패킷을 저장하는 전송 버퍼(Transmitting Buffer)와 전송하였지만 아직 타겟 디바이스로부터 승인 받지 못한 패킷을 저장하고 있는 재전송 버퍼(Retry Buffer)를 가지고 있어야 한다. 하지만 이렇게 전송 버퍼와 재전송 버퍼를 구분하여 두는 경우 전송할 패킷이 전송 버퍼에 있다 하더라도 재전송 버퍼에 여유 공간이 없다면 더 이상 패킷을 전송 할 수 없다는 문제점이 있다. 결국 타겟 디바이스의 승인에 의해 재전송 버퍼가 비어 질 때까지 더 이상 패킷을 전송하지 못한 체 기다려야 하므로 버퍼 사용 효율을 떨어뜨릴 뿐 아니라 데이터 전송 효율 또한 떨어뜨리게 된다.

본 논문에서는 한 개의 버퍼로 전송 버퍼와 재전송 버퍼의 기능을 지원하게 함으로써 필요에 따라 버퍼의 공간을 유연하게 사용할 수 있어 버퍼 사용 및 데이터 전송 효율을 향상시킬 수 있는 방법을 제안한다. 본 논문 Ⅱ장에서는 재전송 기법에 대해 소개하고 Ⅲ장에서는 제안된 버퍼 구조 및 관리 기법에 대해 소개하며 Ⅳ장에서는 시뮬레이션 결과를 보여주고 Ⅴ장에서 결론을 맺는다.

Ⅱ. 재전송 기법 (Retry Mechanism)

패킷을 전송하고자 하는 마스터 디바이스의 데이터 연결 계층은 상대 디바이스의 수신단이 패킷을 정확하게 수신하였는지를 확인하기 위해 전송하고자 하는 TLP에 일련 번호를 순차적으로 할당한 후 물리 계층을 통해 전송하게 된다. TLP를 전송 받은 타겟 디바이스의 데이터 연결 계층은 오류 여부를 확인 후 이상 없는 경우 상위 계층으로 전송한다. 또한 정상 수신되었음을 마스터 디바이스에 알리기 위한 패킷을 보내게 되는데 이를 Ack DLLP(Acknowledge Data Link Packet)라 한다. 이 Ack DLLP에는 마스터 디바이스로부터 받아 정상 처리된 TLP의 일련번호가 첨부되어있다[7][8].

수신단은 한 개의 TLP를 받을 때마다 Ack DLLP를 전송하는 게 아니라 일정 시간을 기다렸다가 가장 마지막으로 정상 수신된 TLP의 일련번호를 붙여 전송하게 된다. 이를 위해 내부에 승인 타이머(Acknowledge Timer)를 두어야 하며 이 타이머는 첫 번째 TLP의 마지막 비트를 수신한 후부터 카운터를 시작하게 된다. 이 타이머의 종료 시간은 PCI 익스프레스 디바이스가 한 번에 전송할 수 있는 최대 데이터의 크기에 의해 결정된다[7][8].

만약 타겟 디바이스가 마스터 디바이스로부터 받은 TLP에 일련 번호가 순차적으로 할당되지 않았든지 혹은 LCRC에 오류가 발생한 경우에 이 오류를 알리기 위해 가장 최근에 정상적으로 전송 받은 TLP의 일련번호와 함께 Nack DLLP(Negative Ack DLLP)를 보낸다. 이 Nack DLLP를 받은 마스터 디바이스는 그 일련번호의 다음 TLP부터 가장 최근에 전송했던 TLP까지 다시 전송을 하여야 한다. 이를 재전송 기법(Retry mechanism)이라 한다. 예를 들면 그림 4(a) 같이, 마스터 디바이스가 일련 번호 1부터 4까지의 TLP를 타겟 디바이스에 전송하였는데 일련번호 3의 TLP에 오류가 발생하였다면 타겟 디바이스는 즉시 일련 번호 2의 Nack DLLP를 보내야 한다. 그러면 송신단은 일련 번호 3과 4의 TLP부터 재전송 한다.

다음으로 고려해 보야 할 것은 마스터 디바이스가 TLP를 전송한 후 일정 시간이 지난 후에도 타겟 디바이스로부터 Ack DLLP를 받지 못한 경우이다. 이때 마스터 디바이스는 승인 받지 못한 TLP부터 마지막으로 전송했던 TLP까지 다시 전송을 하여야 한다[7][8]. 이를 위해 마스터 디바이스의 내부에 타이머를 두어야 하며 이를 재전송 타이머(Retry Timer)라 한다. 예를 들면 그림 4(b)와 같이, 마스터 디바이스가 일련 번호 1부터 5까지의 TLP를 전송한 후, 재전송 타이머가 종료되었음에도 불구하고 어떤 Ack DLLP도 타겟 디바이스가 보내어 주지 않는다면 마스터 디바이스는 일련 번호 1부터 5까지의 TLP를 다시 전송해야 한다.

PCI 익스프레스 엔드포인트 디바이스가 이러한 재전송 기법을 지원하기 위해서 송신단의 데이터 연결 계층에 전송 버퍼와 별도로 그림 5와 같이 재전송 버퍼를 두어야 한다. 전송 계층으로부터 요구되어진 TLP들은

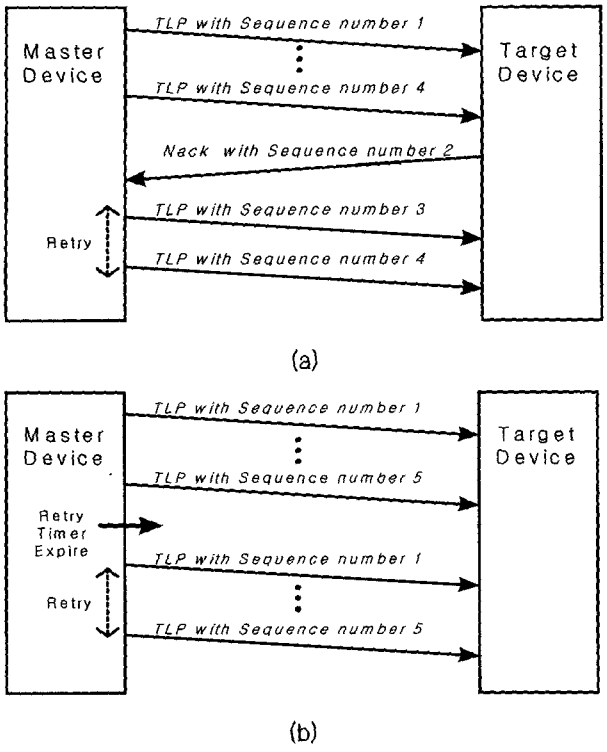


그림 4. 재전송 기법. (a) 타이머의 종료에 따른 재전송, (b) Nack DLLP 수신에 따른 재전송

Fig. 4. Retry mechanism. (a) When retry timer expira -tion, (b) When receiving Nack DLLP.

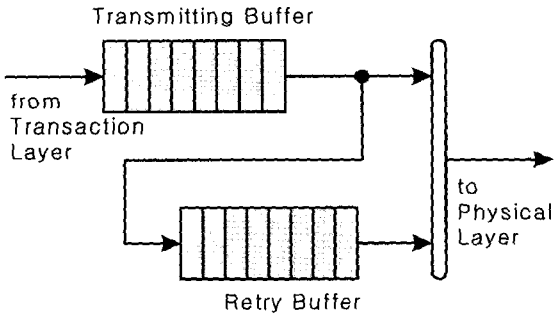


그림 5. 전송 버퍼와 재전송 버퍼

Fig. 5. Transmitting buffer and retry buffer.

전송 버퍼에 저장되고 데이터 연결 계층은 전송 버퍼에 저장된 TLP들을 물리 계층을 통해 상대 디바이스의 수신단으로 전송된다. 전송된 TLP들은 재전송 버퍼에 저장이 되는데 이 TLP들은 상대 디바이스로부터 Ack DLLP나 Nack DLLP를 받기 전에는 폐기 할 수 없다. 또한 상대 디바이스로부터 Nack DLLP 받거나 혹은 재전송 타이머가 종료될 때까지 Ack DLLP를 받지 못하면 재전송 버퍼 내의 TLP들을 재전송해야 한다.

하지만 이렇게 송신버퍼와 재전송 버퍼를 구분하여 사용하는 경우 데이터 전송 효율 및 버퍼 사용 효율을

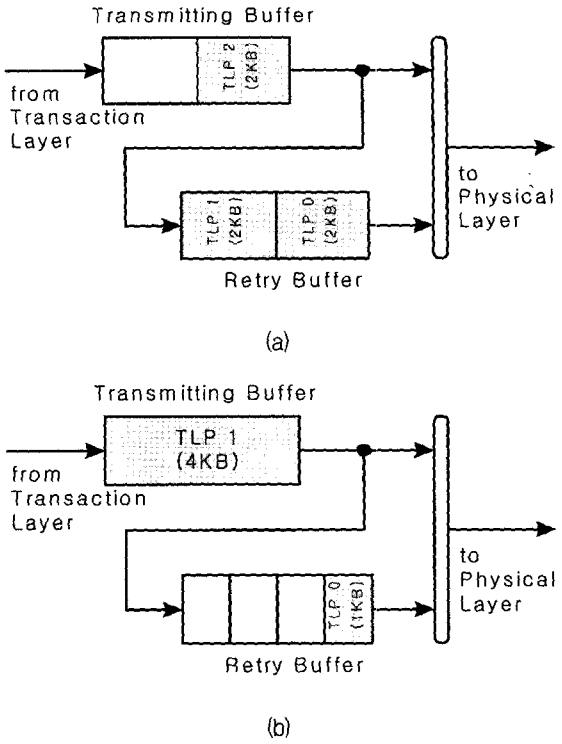


그림 6. 전송 버퍼와 재전송 버퍼를 따로 두는 경우의 문제점

Fig. 6. The restrictions when the transmitting buffer and the replay buffer are separated.

저하시키는 문제점이 있다. 그림 6과 같이 송신 버퍼가 4K 바이트이고 재전송 버퍼가 4K 바이트인 엔드포인트를 예로 들어보자. 여기서 버퍼 크기를 4K로 한 것은 PCI 익스프레스에서 한번에 전송할 수 있는 데이터의 최고 크기가 4K 바이트이기 때문이다[7][8].

만약 그림 6(a)과 같이 엔드포인터가 2K 바이트의 데이터를 가지는 TLP 0과 TLP 1을 전송한 후 재전송 버퍼에 저장해 두었고, 송신 버퍼에는 전송해야할 2K 바이트의 데이터를 가지는 TLP 2가 저장되어 있다고 가정하자. 이 경우 비록 송신 버퍼에는 전송하고자 하는 TLP가 저장되어 있음에도 불구하고 재전송 버퍼가 가득 차버려 더 이상 TLP를 전송 할 수 없다. 상대 디바이스가 Ack DLLP를 보내어 주어 최소한 2K 바이트 만큼의 재전송 버퍼가 빌 때까지는 기다려야 한다. 따라서 전송 버퍼에 저장된 패킷들을 전송하지 못 한 채 기다리는 시간을 낭비함으로 인해 데이터 전송 효율을 떨어뜨리게 된다.

다음으로 그림 6(b)과 같이 엔드포인터가 1K 바이트의 데이터를 가지는 TLP 0을 전송한 후 재전송 버퍼에 저장해 두었고 송신 버퍼에는 전송해야할 4K 바이트의 데이터를 가지는 TLP 1이 저장되어 있다고 가정하자.

이 경우 재전송 버퍼의 남은 공간이 부족하여 TLP 1을 전송 할 수 없다. 즉 상대 디바이스가 TLP 0에 대한 Ack DLLP를 보내어 주어 완전히 재전송 버퍼가 빌 때까지 기다려야 한다. 만약 이때 전송 계층이 다른 TLP를 전송 버퍼에 저장할 준비를 하고 있다면 TLP 1이 전송될 때까지 기다려야 한다. 이는 결국 재전송 버퍼에 3K 바이트만큼 비어 있음에도 불구하고 더 이상 TLP를 전송하지 못한 체 기다려야 하므로 버퍼 사용 효율을 떨어뜨릴 될 뿐 아니라 데이터 전송 효율 또한 떨어뜨리게 된다. 이러한 상황은 전송 계층에서 데이터 양이 큰 TLP의 전송을 연속적으로 요구할 때 더욱 그러할 것이다.

물론 버퍼의 공간을 크게 하면 이러한 문제는 해결될 수 있지만 전체 하드웨어가 커짐으로 인해 이 역시 효율적인 방법이 되지 못 한다.

Ⅲ. 제안된 버퍼의 구조 및 관리 기법

본 논문에서는 앞에서 설명한 전송 버퍼와 재전송 버퍼를 따로 사용하는 경우에 데이터 전송 및 버퍼 사용 효율을 떨어뜨리는 문제점을 해결하기 위한 새로운 방법을 제안한다. 제안된 버퍼 구조는 한 개의 버퍼로 전송 버퍼와 재전송 버퍼의 기능을 지원하게 함으로써 필요에 따라 버퍼의 공간을 유연하게 사용할 수 있어 버퍼의 효율을 최적화 할 수 있다. 제안된 방법을 설명하기 위한 버퍼 구조와 그 예제를 그림 7에 나타내었다.

제안된 구조는 하나의 버퍼와 이를 관리하는 컨트롤 유닛(Control Unit)으로 나누어진다. 컨트롤 유닛은 세 개의 포인터(Pointer)를 이용하여 버퍼를 관리한다. WSP(Write Start Pointer)는 전송 계층에 의해 다음에 저장될 버퍼의 주소를 나타내며 TSP(Transmitting Start Pointer)는 데이터 연결 계층이 이번에 새로 전송해야할 TLP가 저장되어 있는 주소를 나타낸다. 그리고 RSP(Retry Start Pointer)는 재전송을 해야할 TLP가 저장된 버퍼의 시작 주소를 나타낸다.

그림 7(a)과 같이 4개의 TLP가 버퍼에 저장되어 있다고 가정하자. 또한 TLP 0, TLP 1, 그리고 TLP 2는 전송되어진 TLP들로 아직 상대 디바이스에 의해 승인되지 않은 TLP이며 TLP 3은 이번에 전송해야될 TLP라고 가정하자. 결국 RSP가 가리키는 주소 0부터 TSP가 가리키는 주소 3의 앞인 주소 2까지가 재전송 버퍼로 사용되는 것이고 나머지 부분이 전송 버퍼로 사용되는 것이다. 만약 재전송이 요구되는 상황이 발생하면

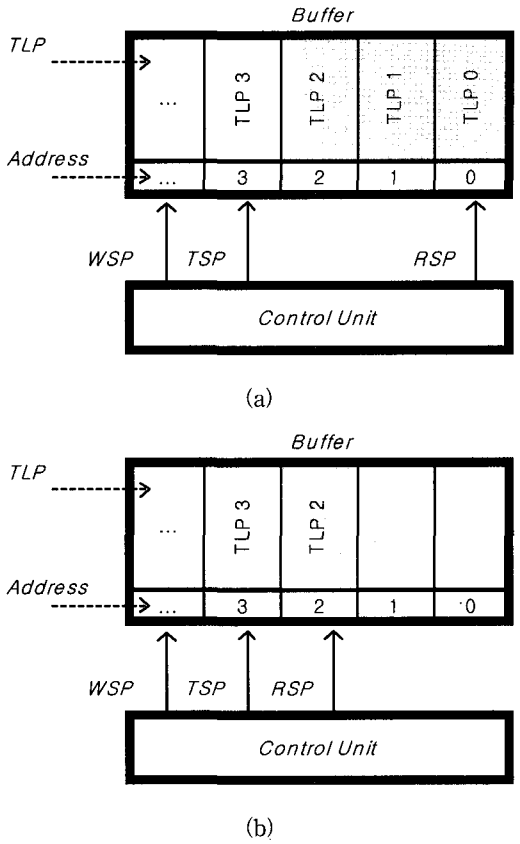


그림 7. 제안된 재전송 버퍼 구조 및 관리 기법
Fig. 7. The proposed buffer architecture and management technique.

주소 0부터 주소 2까지의 TLP들을 전송하면 된다.

만약 상대 디바이스가 TLP 1에 대한 Ack DLLP를 전송해왔다면 TLP 0과 TLP 1은 더 이상 버퍼에 저장해 둘 필요가 없으므로 폐기하고 RSP의 위치를 그림 7(b)와 같이 2로 이동하면 된다. 결국 주소 0과 1의 공간은 전송 버퍼로 사용되어 질 수 있다.

하나의 버퍼로 전송 버퍼와 재전송 버퍼의 기능을 담당하게 하는 제안된 방법은 버퍼를 따로 두는 기존의 버퍼 구조에 비해 버퍼 사용 및 데이터 전송 효율을 향상시킬 수 있다. 예를 들어 재전송 버퍼를 4K 바이트 그리고 전송 버퍼를 4K 바이트로 이용하는 앞의 예제와 비교해 8K 버퍼 한 개를 가지고 이용하면 8K 바이트만큼의 TLP들을 모두 전송한 후에 버퍼 전체를 재전송 버퍼로 사용 할 수 있기 때문에, 마치 8K 바이트의 재전송 버퍼를 사용하는 것과 거의 같은 성능을 얻을 수 있다. 재전송 버퍼가 가득 참으로 인해 더 이상 새로운 TLP를 전송하지 못한 체, 상대 디바이스가 전송해주는 Ack DLLP에 의해 재전송 버퍼가 비어 질 때까지 기다려야하는 상황을 기존 버퍼 구조에 비해 훨씬 줄일 수 있다. 이는 송신 버퍼와 재전송 버퍼의 크기가

고정적으로 정해진 것이 아니라 상황에 따라서 버퍼 공간을 유연하게 사용할 수도 있기 때문이다.

물론 제안된 버퍼의 경우에도 타겟 디바이스의 수신 버퍼의 크기가 작거나 타겟 디바이스의 전송 계층이 수신 버퍼에 저장된 패킷을 느리게 읽어가서 버퍼의 여유 공간이 느리게 생기는 경우라면 많은 성능 향상을 기대할 수 없다. 왜냐하면 마스터 디바이스가 비록 전송할 TLP를 가지고 있다 하더라도 수신단의 버퍼에 공간이 충분하지 못하다면 더 이상 TLP를 전송 할 수 없기 때문이다. 하지만 수신단의 버퍼의 크기가 충분히 크다면 제안된 버퍼가 기존의 버퍼 구조에 비해 데이터 전송 효율이 훨씬 나을 것이다.

Ⅳ. 모의 실험 및 결과

본 실험을 위해 그림 8(a)과 같이 마스터 디바이스의 데이터 연결 계층, 로컬 마스터 디바이스, 타겟 디바이스의 데이터 연결 계층, 그리고 로컬 타겟 디바이스를 C 언어를 이용하여 클럭 단위로 동작하는 행위 모델(Behavioral model)로 구현하였다. 그리고 각 C 모델들을 Verilog 시뮤레이터 PLI(Programming language interface)를 이용해 연결하였다. 본 실험은 제안된 버퍼 구조의 효율을 검증하기 위한 것으로써 PCI 익스프레스의 데이터 연결 계층 위주로 모델링 하였다.

전체의 시뮬레이션 환경을 살펴보면 그림 8(b)과 같다. 테스트 벡터는 사용자가 직접 파일로 작성하거나 혹은 랜덤 발생기에 의해 파일로 자동 생성된다. 이 파일에는 로컬 마스터 디바이스가 수행하기 위한 명령어들이 열거되어 있다. 예를 들면, 로컬 메모리 100번지로부터 1K 바이트의 데이터를 읽어 PCI 익스프레스 인터페이스를 통해 다시 로컬 메모리 200번지에 저장하도록 하려면 mWrite 0x200, 0x100, 1024라고 코딩한다. 이 명령어를 테스트 벡터 파일로부터 읽은 로컬 마스터는 로컬 메모리로부터 데이터를 읽어 TLP를 생성한 다음 이를 데이터 연결 계층의 전송 버퍼에 저장한다. 이 TLP는 마스터 디바이스의 데이터 연결 계층에 의해 타겟 디바이스의 데이터 연결 계층으로 전송되어 수신 버퍼에 저장된다. 수신 버퍼에 저장된 TLP는 로컬 타겟 디바이스에 의해 읽혀져 로컬 메모리에 저장된다.

그림 9는 마스터 디바이스에서 타겟 디바이스로 전송하는 데이터의 크기를 증가시키면서 그때 소요되는 클럭수를 측정한 것이다. PCI 익스프레스에서 한번에 최대로 전송할 수 있는 데이터의 크기가 4K 바이트이

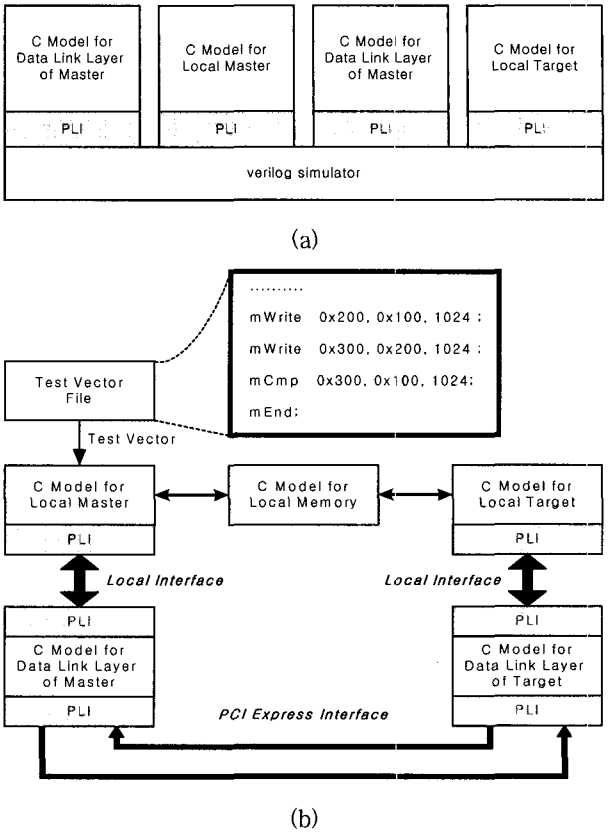
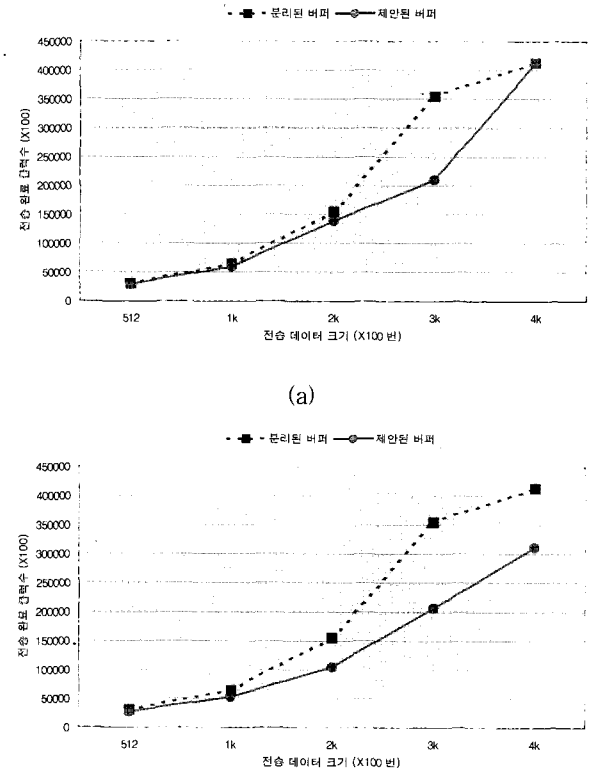


그림 8. 모의 실험 환경
Fig. 8. Simulation environment.

기 때문에 이를 지원하기 위해 전송 버퍼와 재전송 버퍼는 4K 바이트로 두었다. 물론 제안된 방법에서는 8K 바이트 버퍼 하나를 사용하였다. 또한 타겟 디바이스의 수신 버퍼 크기는 4K 바이트, 8K 바이트, 그리고 16K 바이트에 대해 각각 시뮬레이션 하였다. 먼저 512 바이트의 데이터를 연속으로 100개 전송하도록 시뮬레이션 하였으며, 다음으로 전송할 데이터를 4K 바이트까지 증가시키면서 에 각각에 대해 연속 100개씩 전송하도록 시뮬레이션 하였다. 그림 9의 가로축은 전송할 데이터의 크기이며 세로축은 전송이 완료될 때까지 소요된 클럭수이다.

그림 9(a)는 수신단의 버퍼 크기가 4K 바이트인 경우로 분리된 버퍼를 사용하는 경우가 제안된 버퍼에 비해 전송 시간이 더 많이 소요됨을 알 수 있다. 하지만 전송할 데이터의 양이 4K 바이트인 경우에는 성능이 같음을 알 수 있다. 그 이유는 비록 마스터 디바이스가 데이터 전송하고자 하더라도 타겟 디바이스의 수신단의 버퍼가 4K 바이트로 한정되어 있으므로, 타겟 디바이스가 연속된 4K 바이트 데이터를 수신 할 수 없기 때문이다.

그림 9(b)는 수신단의 버퍼 크기가 8K 바이트인 경



(a)

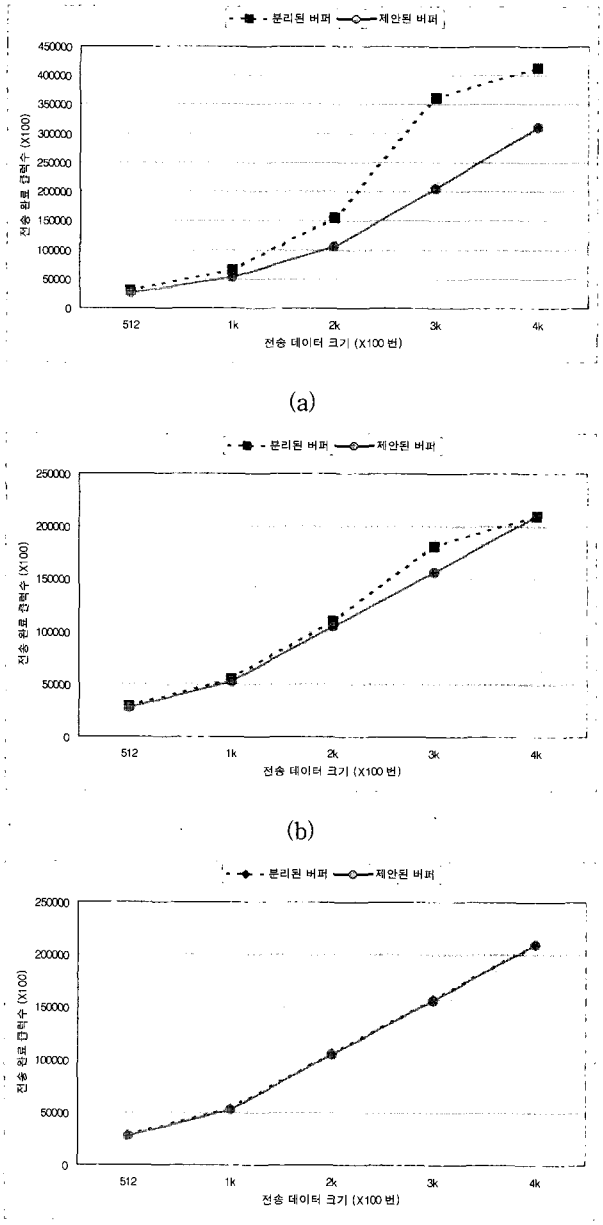
(b)

그림 9. 전송할 데이터 크기 변화와 수신단 버퍼 크기 변화에 따라 데이터 전송에 소요되는 클럭수. (a) 수신단의 버퍼 크기가 4K 바이트, (b) 수신단의 버퍼 크기가 8K 바이트.

Fig. 9. The number of clock cycle to transfer data with varying the size of the data to transmit and the size of the receiver's buffer. (a) The size of the receiver's buffer is 4KB, (b) The size of the receiver's buffer is 8KB.

우로 4K 바이트일 때 보다 제안된 방법의 성능이 더 향상된 것을 알 수 있다. 또한 수신단의 버퍼가 충분히 크기 때문에 4K 바이트의 데이터를 전송할 때도 제안된 구조의 성능이 향상됨 됨을 알 수 있다. 수신단의 버퍼 크기가 16K 바이트 이상인 경우 그림 9(b)와 같이 유사한 결과를 얻었다. 즉 수신단의 버퍼 크기가 8K 바이트 이상이면 전송 데이터 크기에 상관없이 항상 제안된 방법의 성능이 향상됨을 알 수 있다.

그림 10은 마스터 디바이스의 송신단 버퍼 크기를 증가시키면서 전송할 데이터 크기에 따른 데이터 전송 소요 시간을 나타낸 것이다. 타겟 디바이스의 수신 버퍼 크기는 무한대로 두었다. 먼저 512 바이트의 데이터를 연속으로 100개 전송하도록 시뮬레이션 하였으며, 다음으로 전송할 데이터를 4K 바이트까지 증가시키면서 각각에 대해 연속 100개씩 전송하도록 시뮬레이션 하였



(a)

(b)

(c)

그림 10. 전송할 데이터 크기 변화와 송신단 버퍼 크기 변화에 따라 데이터 전송에 소요되는 클럭수. (a) 송신단의 버퍼 크기가 8K 바이트, (b) 송신단의 버퍼 크기가 16K 바이트, (a) 송신단의 버퍼 크기가 32K 바이트

Fig. 10. The number of clock cycle to transfer data with varying the size of the transferred data and the size of the transmitter's buffer. (a) The size of the transmitter's buffer is 8KB, (b) The size of the transmitter's buffer is 16KB, (c) The size of the transmitter's buffer is 32KB.

다. 그림 10의 가로축은 전송할 데이터의 크기이며 세로축은 전송이 완료될 때까지 소요된 클럭수이다.

그림 10 (a)는 송신단의 버퍼 크기가 8K 바이트인 경

우로 분리된 버퍼인 경우 전송 버퍼와 재전송 버퍼가 각각 4K 바이트이다. 그림 10 (b)는 송신단의 버퍼 크기가 16K 바이트인 경우로 분리된 버퍼인 경우 전송 버퍼와 재전송 버퍼가 각각 8K 바이트이고, 그림 10 (c)는 송신단의 버퍼 크기가 32K 바이트인 경우로 분리된 버퍼인 경우 전송 버퍼와 재전송 버퍼가 각각 16K 바이트이다. 그림에서 보듯이 송신단의 버퍼 크기가 8K 바이트인 경우에는 분리된 버퍼구조에 비해 제안된 버퍼의 구조의 성능이 많이 향상된다. 그러나 송신단의 버퍼 크기가 증가함에 따라 성능 향상이 줄어들어 32K 바이트 이상이 되면 성능이 거의 같음을 알 수 있다. 송신단의 버퍼 크기가 충분히 크면 두 버퍼 구조의 성능 차이가 없지만 작은 버퍼 크기인 경우에는 제안된 방법이 훨씬 높은 성능을 가짐을 알 수 있다.

마지막으로 전송할 데이터의 크기를 4 바이트에서 4K 바이트까지 무작위로 500개를 선택되도록 하여 시뮬레이션 하였다. 타겟 디바이스의 수신 버퍼 크기는 무한대로 두었다. 그 결과 송신단의 버퍼 크기가 8K 바이트 일 때는 39%의 성능 향상이, 버퍼 크기가 16K 바이트일 때는 7%의 성능 향상이 됨을 알 수 있었다.

V. 결 론

본 논문에서는 한 개의 버퍼로 전송 버퍼와 재전송 버퍼를 구현하는 방법을 제안하였다. 제안된 버퍼 구조에서는 필요에 따라 버퍼의 공간을 유연하게 사용할 수 있기 때문에 버퍼 사용 및 데이터 전송 효율을 향상시킬 수 있다. 모의 실험 결과 송신단 버퍼 크기가 8K 바이트 일 경우 제안된 방법이 버퍼를 분리하여 사용하는 방법에 비해 데이터 전송 효율이 평균 39% 향상되었음을 확인 할 수 있었다.

참 고 문 헌

[1] 동역메카트로닉스연구소 기술 정보실, *"PCI 버스 해설과 인터페이스 카드 설계"*, 국제 테크노 정보 연구소, 2001.
[2] Edward Solari and George Willse, *"PCI hardware and software: architecture and design"*, Anna -books, 1998.
[3] Don Anderson and Tom Shabnley, *"PCI System Architecture, Mindshare"*, 1999.
[4] PCI SIG, *"PCI Local Bus Specification Revision 2.2"*, PCI SIG, 1998.
[5] http:// www.pcisig.com
[6] Intel whitepaper, "Advanced Switching for the PCI Express Architecture", http://www.intel.com/design/network/papers/251737.htm.
[7] Ravi Budruk, Don Anderson, and Tom Shanley, *"PCI Express System Architecture"*, MindShare, 2003.
[8] PCI SIG, *"PCI Express Base Specifications Revision 1.0a"*, PCI SIG, 2003.

저 자 소 개

현 유 진(학생회원)
제37권 SD편 제10호 참조
2001년 영남대학교 대학원 전자공학과 석사 졸업.
2003년 영남대학교 대학원 전자공학과 박사 수료
<주관심분야: 디지털 시스템 설계, PCI 컨트롤러 설계, 집적회로 및 CAD>

성 광 수(정회원)
제37권 SD편 제2호 참조
현재 영남대학교 전자정보공학부 조교수