PCI-Express의 재전송 버퍼 관리 기법

*장형식, 현유진, 성광수
영남대학교 전자 정보 공학부
e-mail : *jhs456@yumail.ac.kr, braham@yumail.ac.kr, kssung@yu.ac.kr*

Effective Management for Retry Buffer on PCI-Express Interface

*Hyung-sik Jang, Eu-gin Hyun, Kwang-soo sung
Yeungnam University Electrical engineering & Computer science

Abstract

The PCI Express spec introduces retry buffer in Data Link Layer For data integrity. But this buffer have some restrictions as buffer's usage. So we proposed effective management for retry buffer on PCI-Express interface. Proposed buffer management give increase performance and data integrity simultaneously

I. 서론

지난 91년 처음 등장한 PCI 버스는 10년 이상 PC의 표준 버스로 사용되었다. 그러나 인터넷 시대에 접어들면서 폭발적으로 증가하는 데이터 전송량을 처리하기 위해서 보다 넓은 대역폭의 버스 아키텍처가 필요하게 되었다. 대역폭의 문제점을 국소적으로 해결하기 위해 AGP, PCI-X와 같은 것이 발표되었으나 이들 역시 여러 I/O 디바이스가 I/O 버스를 공유(multi-droop)하는 병렬버스(parallel bus technology)구조로 되어 있어 성능향상의 한계를 가지고 있다.[1]

이러한 PCI 버스의 한계를 극복하기 위해 PCI SIG에서 기존의 PCI를 계승하는 새로운 I/O 표준으로 PCI Express를 발표하였다. 기존의 병렬 전송을 하는 PCI가 133MB/s의 전송 속도를 가진 것에 반해, PCI Express는 point-to-point 방식을 이용한 직렬 전송 방식으로 2.5GB/s의 전송 속도를 가진다. PCI Express의 기본 아키텍처는 최하위 계층인 물리계층에서 교체되기 때문에 기존의 PCI 장치에 사용된 운영체제와 디바이스 드라이버 소프트웨어를 그대로 사용할 수 있다. 이와 같은 장점으로 인해 PCI Express가 차세대 컴퓨터 I/O 시스템의 표준으로 자리잡을 것으로 보인다.[1]

본 논문에서는 PCI Express spec에서 소개된 패킷 전송 프로토콜을 소개하고, 이를 Endpoint가 효과적으로 지원하기 위한 새로운 버퍼 관리 구현 기법을 제안하고자 한다.

II. PCI Express 개요

II-I. 계층 구조

PCI Express 디바이스는 3개의 계층으로 구성되어 있다. 송신단(Transmitter)을 살펴보면, 먼저 최상의 계층인 전송계층(Transaction Layer)은 요청하고자 하는 명령어와 전송하고자 하는 데이터를 TLP(Transaction Layer Packet)로 변경하여 하위 계층인 데이터 연결 계층(Data Link Layer)으로 전송한다. 데이터 연결 계층에서는 상위 계층에서 전송된 패킷 정보에 대한 신뢰성을 확보하기 위하여 일련번호와 데이터 보호 코드(LCRC)를 TLP에 붙여서 하위 계층인 물리 계층(Physical Layer)으로 전송한다. 물리 계층은 상위 계층으로부터 받은 병렬 데이터를 직렬로 변환하여 외부로 송신한다.

수신단의 경우 직렬로 받은 패킷을 물리 계층에 의해 병렬로 변환하여 상위 계층으로 전송한다. 데이터 연결 계층에서는 받은 패킷의 데이터 보호 코드를 확인하고 일련 번호를 확인하여 이상이 없는 경우 전송 계층으로 전송을 한다.[2]

II-II. 흐름 제어(Flow Control)

PCI Express는 크게 3가지 명령어인 Posted

Request, Non-Posted Request, 그리고 Completion Request 명령어를 지원한다.[2] Posted Request는 Memory Write 명령어와 같이 Request 디바이스가 데이터를 송신하면 Completer 디바이스가 바로 데이터를 수신 받을 수 있는 명령어이다. Non Posted Request는 Memory Read 명령어와 같이 Requester 디바이스가 데이터를 가져오기 위해 Completer 디바이스에 요청만 하는 명령어이다. 이 경우 Completer는 Requester의 요청을 받아들여 내부적으로 데이터를 준비하여 Completion 명령어를 통해 원래의 Requester에 데이터를 전송하여 준다.

이러한 3가지 명령어의 효과적인 흐름 제어를 위해 수신단은 각각의 명령어별로 버퍼 공간이 할당되고 이를 시스템이 초기화 될 때 송신단에 알려준다. 시스템이 정상화되어 데이터가 전송되어 질 수 있는 상황 중에도, 수신단은 주기적으로 송신단에 버퍼가 얼마나 비어 있는지를 알려주어야 한다. 이를 Credit을 전송한다고 한다.

송신단의 전송계층은 요청 받은 TLP와 수신단의 Credit을 비교하여 현재 TLP의 전송 가능 여부를 결정하는 흐름제어를 할 수 있다.

II-III. 재전송 버퍼 관리 (Retry Buffer Management)

데이터 연결 계층은 데이터의 신뢰성을 확보하기 위해 물리 계층으로 보내는 패킷에 일련 번호를 순차적으로 할당한다. 즉 수신단의 데이터 연결 계층이 수신 패킷의 오류 여부를 확인 후 이상 없는 경우 상위 계층인 전송계층으로 전송하게 되는데, 이 경우 일정 시간이 되면 가장 마지막으로 처리한 패킷의 일련 번호를 Ack DLLP(Acknowledge Data Link Packet)와 함께 원래의 송신단에 보내게 된다. 이것으로 송신단은 해당 패킷의 정상 수신 여부를 확인 할 수 있다. 하지만 수신단이 받은 패킷에 데이터 보호 코드오류 등이 발생한 경우, 수신단은 송신단에 즉시 방금 받은 패킷의 바로 전의 일련 번호를 Nak DLLP(Negative Ack DLLP)와 함께 보내어 전송하게 되는데 이 경우 송신단은 즉시 이 일련번호의 다음 패킷부터 마지막으로 전송한 패킷까지 재전송을 해주어야 한다. 즉 송신단의 데이터 연결 계층은 이러한 재전송 메커니즘을 지원하기 위해 송신버퍼와 별도로 연결 계층 내부에 재전송 버퍼를 가지고 있어야 한다.

송신단에서는 패킷을 전송하면 재전송 버퍼에 저장을 하게 되고, 여기에 저장된 패킷은 수신단으로부터 Ack DLLP나 Nak DLLP를 받기 전에는 폐기 할 수 없다. 또한 송신단은 일정 시간이 지나도 수신단에서 Ack DLLP나 Nak DLLP가 전송되어 오지 않는다면, 재전송버퍼에 있는 모든 패킷을 다시 전송 해주어야 한다.[2]

하지만 이렇게 송신버퍼와 재전송 버퍼를 구분하여 사용하는 경우 데이터 전송 효율을 저하시킬 수 있는데, 본 논문에서는 이를 분석하고 한 개의 버퍼만으로 송신버퍼와 재전송버퍼의 모든 기능을 제공 할 수 있는 방법을 제안하고자 한다.

III. 제안된 버퍼의 구조 및 관리 기법

본 논문에서는 재전송버퍼를 효과적으로 관리하기 위한 방법을 소개하고자 한다. 먼저 그림 1은 일반적인 송신버퍼와 재전송버퍼를 구분하여 사용하는 경우이다. 데이터 연결 계층은 송신버퍼의 데이터를 하위 계층으로 전송을 하게 되면 반드시 재전송 버퍼에 저장을 해두어야 한다. 송신단은 일정 시간이 지난 후에 수신단에서 응답이 없는 경우 재전송버퍼의 데이터를 다시 전송해 주어야 한다. 결국 그림에서 보듯이 물리 계층으로 전송되는 패킷은 송신버퍼와 재전송버퍼에 저장된 패킷 중하나이다.

만약 수신버퍼의 크기가 유한의 값으로 제한되어 진다면 그림 1의 (a)와 같이 송신 버퍼가 가득 차버려 전송계층에서 더 이상 패킷을 전송할 수 없는 경우가 발생한다. 수신단의 전송계층이 버퍼의 크기가 작거나 버퍼로부터 데이터를 느리게 가져가는 경우, 수신단의 버퍼는 충분히 비어있지 않는 상황이 발생하게 될 것이다. 수신단은 Credit이 충분하지 않는 상황에서 송신단은 패킷을 전송할 수 없게될 것이고 그로 인해 송신버퍼는 가득 차게될 것이다. 즉 재전송버퍼와 무관하게 전송 효율이 떨어질 것이다.

만일 수신버퍼의 크기를 무한히 크게 한다면 다음 그림 1의 (b)와 같이 재전송버퍼의 크기에 의해 영향을 받게 된다.

재전송버퍼가 모두 차 버린다면 송신단은 비록 전송할 패킷이 송신버퍼에 있더라도 전송할 수가 없다. 결국 수신단에서 Ack DLLP나 Nak DLLP를 보내어 재전송버퍼가 비어질 때까지 기다려야 한다. 일반적으로 수신단은 Ack DLLP의 경우 즉시 전송하여 주는 것이 아니라 내부 타이머에 의해 일정 시간 후 인터럽트가 걸리면 전송하도록 되어 있다. 이런 문제는 재전송버퍼를 작게 설계한다면 더욱 자주 발생할 것이다.

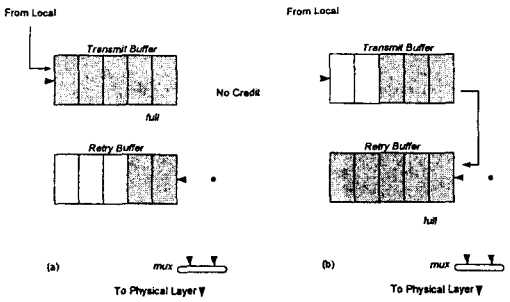

그림 1. 재전송 버퍼 사용 시 고려할 사항.

또한 송신버퍼의 크기가 제한적이면 수신단의 버퍼와 재전송버퍼의 크기가 아무리 커진다고 하더라도 성능의 향상을 기대 할 수 없을 것이다. 왜냐하면 전송할 수 있는 양이 제한적이기 때문이다.

즉 위와 같은 제약사항들을 해결하기 위해서는 송신버퍼뿐 아니라 재전송버퍼의 크기도 함께 늘여야한다. 이렇게 된다면 성능향상을 위해 많은 양의 버퍼가 사용되어 져야 할 것이다.

결국 송신버퍼나 재전송버퍼의 크기를 무한히 크게 하지 않는 경우 버퍼 사용의 효율을 떨어뜨리게 된다. 송신버퍼와 재전송 버퍼를 구분하여 사용하는 경우 이러한 버퍼 사용의 효율을 떨어뜨리는 문제가 발생하는데, 만약 송신버퍼와 재전송버퍼를 구분하지 않고 한 개의 버퍼를 서로가 필요에 따라 유연하게 공간을 사용할 수 있다면 이를 해결 할 수 있을 것이다. 그래서 본 논문에서는 그림 2와 같은 방법을 제안하고자 한다.

제안된 방법은 버퍼와 이를 관리하는 Control Unit에 다음 그림 2에서와 같이 세 개의 Pointer를 사용하는 것이다. 그림 2 (a)에서 보면 현재 4개의 패킷이 버퍼에 저장되어 있으며, 상위 계층에 의해 다음에 저장될 버퍼의 주소인 WSP(Write Start Pointer)와 현재 데이터 연결 계층에 의해 하위 계층으로 전송되어야될 버퍼 주소인 TSP(Transmit Start Pointer) 그리고 재전송을 해야할 때 시작하는 주소인 RSP(Replay Start Pointer)가 있다.

그림 2 (a)에서 논리 주소 0, 1, 2의 패킷은 모두 전송되어 졌고, 수신단으로부터 Ack DLLP는 전송 받지 못한 경우이다. 만약 일정 시간이 지난 후, 송신단이 재전송을 시작할 때 RSP 주소에서 TSP 주소 직전까지의 패킷을 전송하면 될 것이다.

그림 2 (b)은 논리 주소 0에 저장되어있던 Posted Request에 대한 Ack DLLP가 수신된 경우로, 송신단은 이제 더 이상 이 패킷을 저장해 둘 필요가 없으므로 버퍼에서 폐기시키고 RSP도 1 증가시킨다.

이렇게 한 개의 버퍼로 재전송버퍼의 기능을 함께 하는 경우, 재전송버퍼를 따로 두는 것에 비해 버퍼사용 효율을 훨씬 향상시킬 수 있다. 송신버퍼와 재전송버퍼의 크기가 정해진 것이 아니라 상황에 따라서 버퍼 전체를 모두 사용할 수도 있기 때문이다.

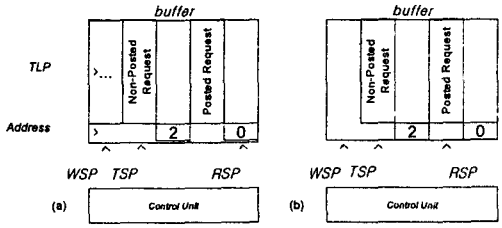

그림 2. 제안된 재전송 버퍼 관리 기법

지금까지 데이터 연결 계층의 효과적인 재전송버퍼 관리기법을 제안하였다. 본 방법에서 제안된 버퍼를 이용하면 송신버퍼와 재전송버퍼 관리를 모두 지원함으로써 전송계층 및 데이터 연결 계층의 설계를 간단하게 할 수 있을 것이다.

Ⅳ. 실험 및 결과

본 실험을 위해 PCI Express의 물리계층을 제외한 전송계층과 데이터 연결 계층을 C언어를 이용하여 동작모델로 구현하였다. 버퍼관리를 위한 테스트 환경을 위해 물리계층을 통한 데이터 전송 부분을 제외하고 데이터 연결계층간의 통신을 C언어로 구현하였다. 전송 결과는 Memory Write 명령을 20번하여 이 전체 명령에 대해 Ack가 오는 시간을 측정하여 송신버퍼와 재전송버퍼를 분리하여 관리하는 경우(a)와 한 개의 버퍼로 관리하는 경우(b)에 대해 비교하였다.

다음 그림 3과 그림 4는 전송량을 512에서 4K까지 증가하면서 수신단의 버퍼의 크기를 크게 하는 경우 전송효율에 대해 알아보았다. 이때 송신버퍼와 재전송버퍼의 크기는 각각 4KByte이고 전체 버퍼의 크기는 8KByte가 된다. 그림 3은 수신단의 버퍼가 4KByte인 경우이고 그림 4는 수신단의 버퍼가 8KByte인 경우에서의 비교이다. 그림에서 전송량이 4KByte일 때를 보면 전송버퍼와 재전송버퍼를 분리해서 쓰는 경우 수신단의 버퍼 크기가 커진다 하더라도 전송 효율이 증가하지 않음을 알 수 있다. 반면 제안된 버퍼 기법에 대해서는 수신단버퍼의 크기에 전송효율이 증가함을 알

수 있다. 즉 앞에서 언급한 것처럼 수신단의 버퍼가 아무리 커진다 하더라도 기존버퍼 관리기법에서 송신 버퍼가 4KByte이므로 성능향상을 기대 할 수 없지만 제안된 버퍼기법에서는 송신버퍼를 8KByte모두를 사용할 수 있기 때문에 성능이 향상됨을 알 수 있다.

다음 그림 5는 송신버퍼와 재전송 버퍼의 크기를 각각 1K, 2K, 4K, 8K씩 늘이면서 각 전송량에 대한 전송효율을 비교하였다. 수신단의 버퍼크기에 대해 영향을 받지 않고 순수하게 송신버퍼와 재전송버퍼 크기의 영향을 살펴보기 위해 수신단의 버퍼 크기를 8KByte로 고정하였다. 송신버퍼와 재전송버퍼의 크기가 각 1KByte인 경우 전송할 수 있는 최대 크기가 1KByte이기 때문에 전송량이 1KByte인 경우에 대해서 그래프를 비교하였다. 제안한 버퍼기법과 기존의 버퍼기법 모두 버퍼의 크기가 무한히 커지는 경우에서는 성능의 차이는 없다 하지만 전체 버퍼의 크기가 작아질수록 제안된 버퍼기법에 대해서 성능이 향상되었다. 앞에서 언급한 것처럼 버퍼의 크기가 적은 경우에서 버퍼를 더욱 효과적으로 사용할 수 있다.

V. 결론

PCI Express에서는 데이터 전송의 신뢰성을 확보하기 위해 재전송 버퍼를 사용하였다. 하지만 이러한 재전송 버퍼로 인해 버퍼 사용의 효율이 떨어지는 몇 가지 요인을 발견할 수 있었다.

본 논문에서는 데이터 전송의 신뢰성과 버퍼 사용의 효율을 향상시키기 위한 새로운 버퍼 관리 기법을 제안하였다. 전송버퍼와 재전송버퍼를 하나 버퍼로 관리함으로써 버퍼의 사용 효율을 높일 수 있음을 실험을 통해 알 수 있었다.

참고문헌

[1] 정만환, "어떤 버스를 타시겠습니까"*microsoftware*, 2002년 3월

[2] PCI SIG, PCI Express Base Specification Revision 1.0a, *PCI SIG*, April 15, 2003

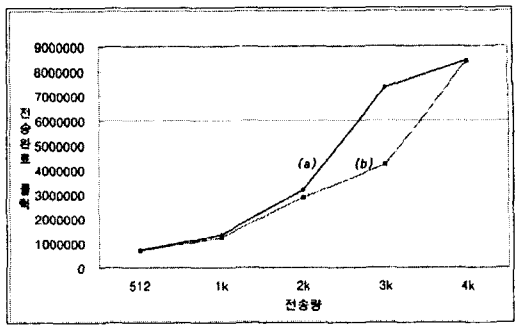

그림 3 수신단 버퍼가 4K일 때, 전송량에 따른 전송 효율비교
(a) 버퍼를 분리하여 관리 (b) 하나의 버퍼로 관리

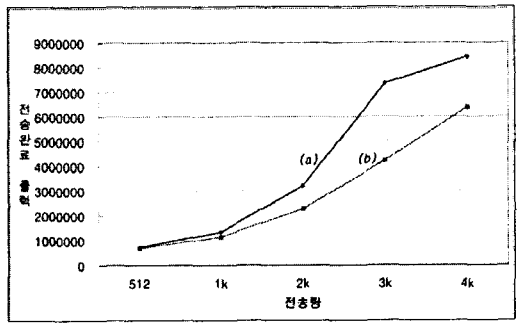

그림 4 수신단 버퍼가 8K일 때, 전송량에 따른 전송 효율비교
(a) 버퍼를 분리하여 관리 (b) 하나의 버퍼로 관리

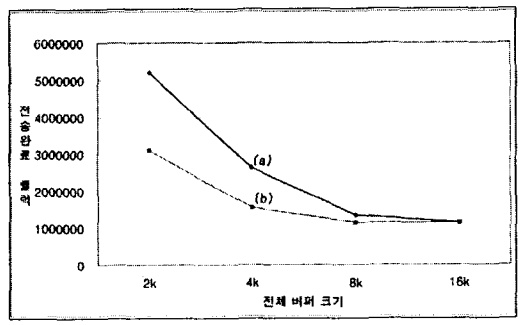

그림 5 송신버퍼와 재전송버퍼 크기에 대한 전송효율 비교
(a) 버퍼를 분리하여 관리 (b) 하나의 버퍼로 관리