논문 2005-42CI-4-8

차세대 통신 플랫폼을 위한 입출력 컨트롤러 설계

(Den of I/O Controller for Future Communication Platform)

현 유 진*, 성 광 수**

(Eugin Hyun and Kwang-Su Seong)

요 약

본 논문에서는 차세대 통신 플랫폼을 위한 PCI 익스프레스의 전송계층과 데이터 연결계층의 모든 기능을 지원하는 PCI 익스프레스 컨트롤러를 설계하였다. 설계된 컨트롤러는 재전송 메커니즘을 효과적으로 지원하기 위해 제안된 송신버퍼 구조를 가지고 있다. 이 버퍼 구조는 전송 버퍼와 재전송 버퍼를 한 개의 버퍼로 통합하여 재전송 버퍼의 공간을 유동적으로 할당할 수 있는 방법이다. 또한 설계된 컨트롤러의 송신단 전송계층은 제안된 버퍼 구조를 효과적으로 지원하도록 설계되어 졌다. 흐름제어를 지원하기 위해 PCI 익스프레스의 모든 수신 디바이스는 각 명령어를 위한 3개의 수신 버퍼를 가지고 있어야 하며, 각 수신 버퍼의 빈 공간을 주기적으로 상대 디바이스에 알려주어야 한다. 설계된 컨트롤러에서는 단지 하나의 수신 버퍼를 이용하여 흐름제어를 보다 쉽게 지원하기 위한 방법을 제안하였다. 또한 설계된 컨트롤러는 제안된 테스트 벤치를 통해 검증되었고 동작함을 확인할 수 있었다.

Abstract

In this paper, we design a PCI Express controller for future communication system. The controller supports the full functionality of Transaction Layer and Data Link Layer of PCI Express. The designed controller has the proposed transmitter buffer architecture to obey Replay mechanism. This scheme merges the transmitting buffer and the replay buffer. The proposed buffer has the higher data transfer efficiency than the conventional buffer architecture because it can dynamically adjust size of a replay buffer space. We also design transmitter of Transmitter Transaction Layer to effectively support the proposed buffer. The receiver device of PCI Express must possess the buffer for three types of transaction to support Flow Control. And it must report the amount of the buffer space regularly to the port at the opposite end of the link. We propose the simple receiver buffer scheme using only one buffer to easily support Flow Control. And the designed controller is verified under proposed test bench

Keywords: Future communication, Communication platform, PCI Express, Controller, Verification

I. 서 론

오늘날 통신 기술과 컴퓨터 기술은 하나로 통합되고 있다. 인터넷과 네트워크 사용의 폭발적인 증가로 인해 최근 대부분의 컴퓨터 시스템에서는 통신 기술을 제공한다. 즉 일반적인 PC는 물론이고 노트북 그리고 PDA에 이르기까지 유무선 통신 기술이 시스템의 한 부분으로 자리 잡고 있다[1]. 반대로 통신 시스템 역시 효과적인 시스템 제어를 위해 컴퓨터 기술을 필요로 하고 있다[1]. 하지만 지금까지 통신 및 임베디드 솔루션은 칩과 칩 혹은 시스템간의 연결에 있어 여러가지 인터페이스를 이용한데 반해, 컴퓨터 시스템은 PCI와 같은 입출력 표준안을 인터페이스로 사용해왔다. 결국 통신 시스템과 컴퓨터 시스템의 인터페이스가 서로 다름으로 인해 전체 시스템 통합 구성에 있어서 상호 연결의 문제점이 발생한다. 이는 코딩 및 물리적 설계를 복잡하게 만들

* 학생회원, 대구경북과학기술연구원(DGIST) 핵심부품연구팀
(Daegu Gyeongbuk Institute of Science & Technology)
** 정회원, 영남대학교 전자정보공학부
(Department of Electrical Engineering and Computer Science, Yeungnam University)
※ 본 연구보고서는 산업자원부에서 지원하고 있는 경상북도 RFID 산업 혁신기반 구축사업의 연구결과입니다.
접수일자: 2005년5월4일, 수정완료일: 2005년7월2일

어 시스템 전체의 가격을 증가시킨다. 따라서 통신 및 컴퓨터 시스템의 통합 발전을 위해서는 보다 간단한 인터페이스 표준안이 필요하게 되었다[1][2].

더욱이 오늘날의 통신 및 임베디드 시스템은 비디오 및 오디오의 스트리밍 데이터(streaming data)를 실시간으로 처리 할 수 있어야 되기 때문에, 시스템의 입출력 인터페이스는 높은 데이터 전송률을 가져야 한다. 하지만 여러 디바이스가 하나의 버스를 공유하는 병렬 버스 구조로 되어있는 기존의 입출력 인터페이스인 PCI 2.2와 PCI-X는 이러한 요구를 충족시킬 수가 없다[1][2].

이에 PCI SIG에서는 차세대 컴퓨터 및 통신 시스템을 위한 새로운 입출력 표준안으로 PCI 익스프레스(Expresss)를 소개하였다. PCI 익스프레스는 점대점(point-to-point) 방식을 이용한 직렬 전송 방식으로 현재 2.5Gpbs의 전송 속도를 가진다. 또한 기존 PCI 장치에 사용되는 OS와 디바이스 드라이버 S/W를 그대로 사용할 수 있어 차세대 컴퓨터 및 통신 시스템의 입출력 표준안으로 자리 잡을 것으로 보고 있다[4][5].

그림 1은 PCI 익스프레스를 이용한 통신 시스템의 구성도이다. PCI 익스프레스 시스템은 호스트 브리지(Host Bridge), 스위치(Switch), 그리고 여러 개의 엔드포인트(Endpoint)로 구성된다. 스위치는 데이터 전송을 위한 패스(path)를 연결해주는 라우터(Router) 장치로 기존 PCI의 버스를 대체하는 디바이스 이다. 엔드포이트는 PCI 익스프레스에 연결된 모든 입출력 디바이스로써 Gb 이더넷 카드(Ethernet Card), 그래픽 컨트롤러,

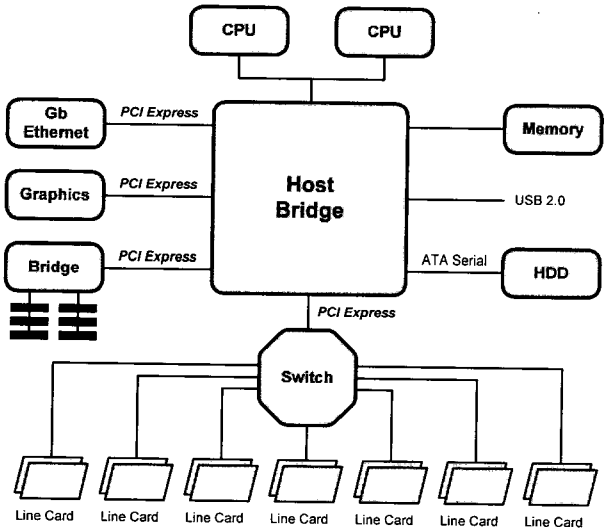

그림 1. PCI 익스프레스를 이용한 네트워킹 통신 시스템 구성도

Fig. 1. PCI Express based networking communication system.

PCI 2.2 브리지, PCI-X 브리지, 그리고 라인 카드 등을 예로 들 수 있다.

본 논문에서는 PCI 익스프레스 엔드포인트 컨트롤러를 설계하였다. 설계된 컨트롤러는 PCI 익스프레스의 전송계층과 데이터 연결계층의 모든 기능을 지원한다. PCI 익스프레스의 송신 디바이스는 재전송 메커니즘을 지원하기 위해 전송할 패킷을 저장하는 전송 버퍼(Transmitting buffer)와 전송하였지만 아직 상대 디바이스로부터 승인 받지 못한 패킷을 저장하고 있는 재전송 버퍼(Retry buffer)를 가지고 있어야 한다[4][5]. 하지만 이렇게 두개의 버퍼를 구분하여 사용하는 경우 버퍼 사용 효율이 떨어 질수 있다[7][8]. 그래서 이를 해결하기 위해 전송 버퍼와 재전송 버퍼를 한 개의 버퍼로 통합하여 재전송 버퍼의 공간을 유동적으로 할당할 수 있는 방법을 앞선 연구[7][8]에서 제안 하였고 설계된 컨트롤러는 이 버퍼 구조를 사용한다.

데이터 연결계층은 전송 계층이 생성한 패킷에 LCRC(Link to link CRC)와 일련번호를 첨부한다. 따라서 데이터 연결계층이 패킷을 재전송을 하고자 할 때도 다시 똑같은 LCRC와 일련번호를 붙여야 하므로 인해 데이터 연결계층의 설계를 복잡하게 만든다. 설계된 컨트롤러 이를 해결하기 위해 LCRC, 일련번호, 그리고 물리계층에서 생성하는 프래임(Frame)까지 모두 전송계층에서 생성하도록 하였다. 이러한 구조는 또한 제안된 송신 버퍼를 더욱 효과적으로 사용할 수 있다.

흐름제어(Flow control)을 지원하기 위해 PCI 익스프레스의 모든 수신 디바이스는 3개의 각 명령어를 위한 수신 버퍼를 가지고 있어야 하며, 3개의 각 수신 버퍼의 빈 공간은 주기적으로 상대 디바이스에 알려주어야 한다. 설계된 컨트롤러에서는 단지 하나의 수신 버퍼를 이용하여 흐름제어를 보다 쉽게 지원하기 위한 방법을 제안하였다.

이렇게 설계 된 컨트롤러의 각 블록을 효과적으로 관리하기 위해 80C51 마이크로프로세서를 이용하여 PCI 익스프레스 프로토콜을 제공하는 프로그램을 작성 후 포팅(Porting)하였다. 또한 설계된 컨트롤러가 실제 PCI 익스프레스 프로토콜 중에 발생 할 수 있는 모든 시나리오에서 검증될 수 있도록 어셈블러 명령어를 정의 하였으며 이 명령어를 지원하는 호스트 브리지, 로컬 마스터 디바이스, 로컬 슬레이브 디바이스의 버스 동작 모델로 구성된 테스트 벤치(Test bench)도 제안하였다.

본 논문 II장에서는 PCI 익스프레스 프로토콜에 대해 설명하고 III장에서는 설계된 PCI 익스프레스 컨트

롤러와 검증 환경에 대해 소개한다. Ⅳ장에서는 시뮤레이션 결과를 보여주고 Ⅴ장에서 결론을 맺는다.

Ⅱ. PCI 익스프레스 개요

1. 계층구조

PCI 익스프레스는 그림 2(a)와 같이 전송계층(Transaction Layer), 데이터 연결계층(Data Link Layer), 물리 계층(Physical Layer)으로 구성된 계층 구조를 가진다. 그리고 각 계층은 송신단과 수신단으로 나누어진다. 디바이스 코어로부터 요청된 데이터는 송신단의 전송계층에 의해 패킷으로 생성되는데 헤드, 데이터, 그리고 ECRC(End to End CRC)로 구성되며 이를 TLP(Transaction Layer Packet)라 한다. 데이터는 명령어의 종류에 따라 TLP에 포함되어있지 않을 수도 있다.

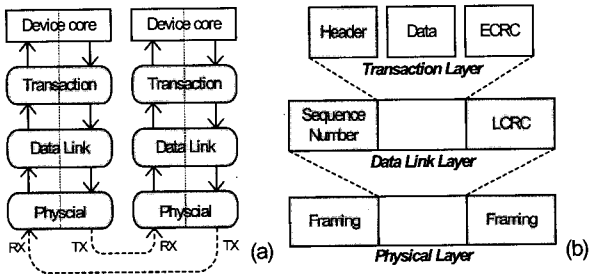

그림 2. PCI 익스프레스 계층 구조, (a) 계층 구조, (b) 패킷 구조

Fig. 2. Layer of PCI Express, (a) High-level layering diagram, (b) packet flow through the layers

데이터 연결계층에서는 이 TLP의 신뢰성을 확보하기 위해 LCRC(Link to Link CRC)와 일련 번호(Sequence Number)를 첨부한다. 물리 계층은 데이터 연결계층으로부터 받은 TLP에 시작과 끝을 알리는 프레임(Frame) 정보를 붙인 다음 직렬 정보로 변환 후 외부로 전송한다.

2. 흐름제어

PCI 2.2와 PCI-X에서 송신 디바이스가 데이터 전송을 요청한 경우 만약 수신 디바이스가 충분한 버퍼 공간을 확보하지 못했다면 전송 프로토콜은 수신 디바이스에 의해 바로 거부되어진다. 그리고 송신 디바이스는 이 데이터 전송이 완성될 때 까지 수신 디바이스에 계속 요청을 시도할 것이다. 이러한 상황은 전체 버스 사용 효율을 떨어뜨리게 된다. 이러한 문제를 해결하기

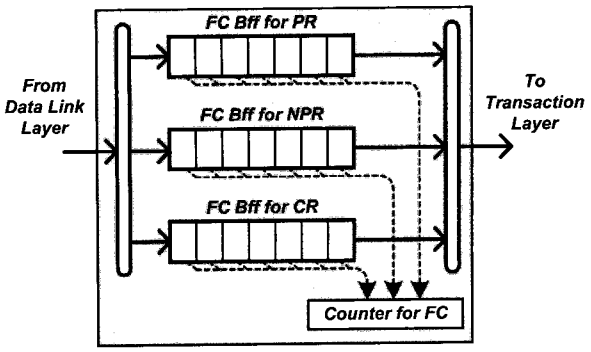

그림 3. 수신 버퍼.

Fig. 3. Receiver buffer.

위해 PCI 익스프레스는 흐름제어(Flow Control)를 지원한다. 즉 PCI 익스프레스의 송신단은 TLP를 전송하기 전에 링크 반대편의 수신 디바이스가 전송하려는 데이터를 받을 수 있는 버퍼 공간이 충분한지를 반드시 확인한다[4][5]. 그리고 이러한 흐름제어를 지원하기 위해 수신 디바이스의 데이터 연결계층은 자신의 수신버퍼 크기를 FC DLLP(Flow Control Data Link Layer Packet)를 이용하여 주기적으로 링크의 반대편 디바이스에 알려주어야 하다. 여기서 DLLP는 링크 관리를 위해 데이터 연결계층에 의해 생성되어지는 패킷이다. 이러한 흐름제어를 지원하기 위해서는 수신단은 그림 3과 같이 포스티드 요청(Posted Request), 난포스티드 요청(Non-Posted Request), 완성 요청(Completion Request)을 위한 각각의 수신버퍼를 가지고 있어야 한다[4][5].

포스티드 요청은 메모리 쓰기 명령어와 메시지 전송 명령어이다. 메모리 쓰기 명령어는 메모리 공간을 가지는 디바이스에 데이터를 전송하는 명령어이고 메시지 명령어는 PCI 익스프레스 디바이스간에 에러 및 기타 정보를 전송하기 위한 명령어이다. 난포스티드는 모든 읽기 명령어와 IO 명령어 그리고 컨피규레이션(Configuration) 명령어이다. IO 명령어는 IO 공간을 가지는 이전의 디바이스와 호환을 이루기 위한 명령어이며 컨피규레이션 명령어는 컨피규레이션 레지스터(Configuration register)에 접근하기 위한 명령어이다. 컨피규레이션 레지스터는 모든 PCI 익스프레스 컨트롤러의 상태와 동작 제어 등을 할 수 있는 내부 레지스터로 호스트 브리지에 의해서만 접근이 가능하다. 메모리 읽기 명령어를 수신한 디바이스는 데이터를 준비하여 이를 원래 요청한 디바이스에 전송하게 되는데 이를 완성 요청이라 한다. IO 쓰기 명령어와 컨피규레이션 읽기 명령어도 완성 요청을 요구하게 되는데 이 경우에는 데이터 전송이 이루어졌음을 알리는 정보만 원래 요청

디바이스에 알리게 된다. 즉 완성 요청은 난포스티드 요청을 수신한 경우에만 요구되어지는 명령어이다.

3. 재전송 메커니즘(Replay Mechanism)

전송계층에 의해 전송되어지는 각 TLP에는 데이터 연결계층에 의해 일련번호가 순차적으로 첨부된다. 또한 헤드, 데이터, ECRC, 그리고 일련번호를 이용하여 계산한 LCRC도 TLP에 붙게 된다.

링크 반대편의 수신 디바이스가 송신 디바이스로부터 TLP를 수신한 경우, 수신 디바이스의 데이터 연결계층은 수신한 TLP의 일련번호가 순차적으로 할당이 되어 있는지와 LCRC에 오류가 있는 지를 먼저 확인한다. 만약 에러가 발생하지 않았다면 수신 디바이스의 데이터 연결계층은 정상적으로 TLP를 수신하였음을 원래의 송신 디바이스에 알리기 위한 승인 절차로 ACK(Acknowledge) DLLP를 전송한다. 수신단은 한 개의 TLP를 받을 때 마다 ACK DLLP를 전송하는게 아니라 일정 시간을 기다렸다가 가장 마지막으로 정상 수신된 TLP의 일련번호를 붙여 전송하게 된다. 이를 위해 내부에 승인 타이머(ACK 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 전송 할 수 있는 최대 데이터 크기에 의해 결정된다.

만약 수신한 TLP의 일련번호나 LCRC에 오류가 발생한 경우 수신 디바이스는 NACK(Negative ACK) DLLP를 전송하게 된다. NACK DLLP를 받은 송신 디바이스의 데이터 연결계층은 이미 전송은 되어졌지만 아직 승인되지 않은 TLP들을 모두 다시 전송하게 되는데 이를 재전송 메커니즘이라 한다[4][5]. 또한 송신 디바이스가 TLP를 전송한 후 일정 시간이 지난 후에도 상대 디바이스로부터 ACK DLLP를 받지 못한 다면 이때 역시 재전송을 해야 한다. 이를 위해 송신 디바이스는 재전송 타이머(Replay Timer)를 두어야 한다.

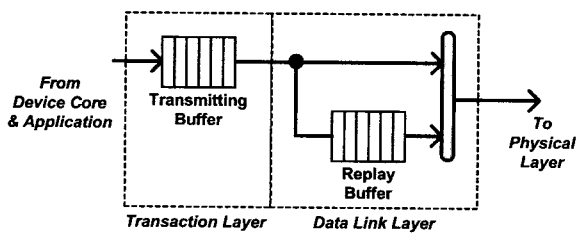

그림 4. 전송 버퍼와 재전송 버퍼
Fig. 4. Transmitting buffer and replay buffer.

이러한 재전송 메커니즘을 지원하기 위해 데이터 연결계층은 송신버퍼와 별도로 재전송 버퍼를 그림 4와 같이 두어야 한다. 전송 버퍼에 있는 TLP는 물리 계층으로 전송된 후 재전송 버퍼에 백업된다. 만약 재전송을 해야 할 상황이 발생한다면 재전송 버퍼에 있는 모든 TLP는 다시 전송되어질 것이다. 그러나 상대 디바이스로부터 ACK DLLP를 받는 경우 재전송 버퍼에 있는 해당 TLP들은 모두 폐기 된다.

III. 컨트롤러의 설계 및 검증

1. 설계

본 논문에서는 PCI 익스프레스 전송계층과 데이터 연결계층의 모든 기능을 지원하는 PCI 익스프레스 컨트롤러를 설계하였다.

APCE라고 명칭 된 컨트롤러의 블록 다이어그램은 그림 5(a)와 같다. APCE는 모두 8개 블록인 송신단 전송계층(TxTL), 송신버퍼(TxBf), 송신단 데이터 연결계층(TxDL), 수신단 전송계층(RxTL), 수신버퍼(RxBf),

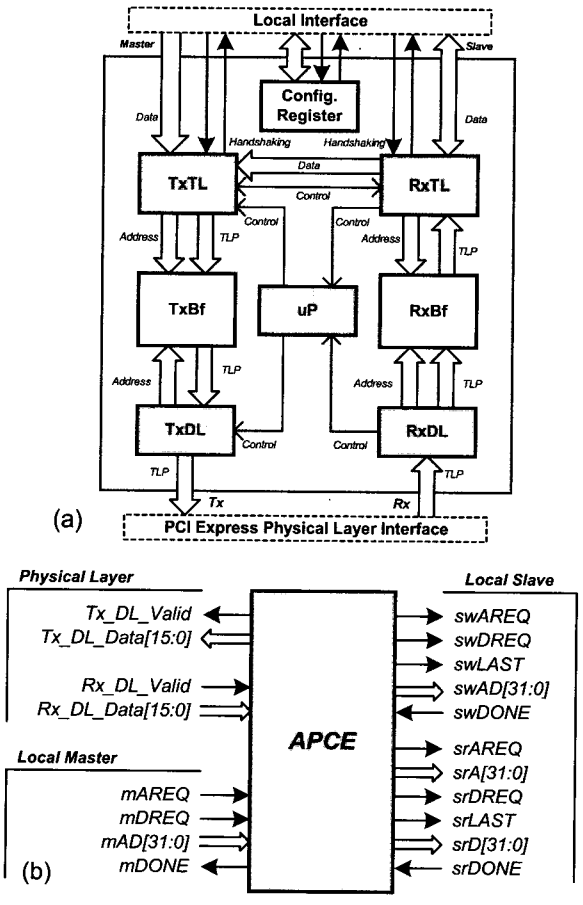

그림 5. 설계된 컨트롤러 블록 다이어그램
Fig. 5. Top block of the designed controller.

수신단 데이터 연결계층(RxDL), 마이크로프로세서(uP), 그리고 컨피규레이션 레지스터로 구성 되어 있다. 또한 APCE는 그림 5(b)와 같이 모두 3개의 외부 인터페이스를 가진다. 즉 로컬 마스터 인터페이스, 로컬 슬레이브 인터페이스, 그리고 PCI 익스프레스 인터페이스이다. APCE는 데이터 신호인 Tx_DL_Data[15:0]와 이 값이 유효함을 알리는 Tx_DL_Valid 신호를 이용하여 PCI 익스프레스 물리 계층으로 패킷을 전송한다. 또한 Rx_DL_Data[15:0]과 Rx_DL_Valid 신호는 APCE가 물리 계층으로 패킷을 수신 할 때 사용된다.

가. 송신버퍼(TxBf)

송신단의 데이터 연결계층은 재전송 메커니즘을 지원하기 위해 그림 4와 같이 재전송 버퍼를 가지고 있어야한다. 이 버퍼 구조는 아주 간단함으로 인해 인텔 등 대부분의 PCI 익스프레스 컨트롤러는 이 버퍼 구조를 사용하고 있다[6]. 하지만 그럼에도 불구하고 두개의 버퍼가 분리 되어 있음으로 인해 전송 효율 및 버퍼 사용 효율이 떨어지게 된다. 그래서 본 논문의 앞선 연구에서 재전송 메커니즘을 지원할 수 있는 효과적인 버퍼 구조를 제안하였다[7][8]. 제안된 버퍼 구조는 전송 버퍼와 재전송 버퍼로 나누어진 두개의 버퍼를 단지 하나의 버퍼로 통합시킴으로 인해 재전송 버퍼 공간을 유동적으로 사용할 수 있는 방법이 다. 그림 6은 이 버퍼구조의 간단한 예를 보여준다.

새로운 TLP를 전송하고자 하는 전송계층은 WSP(Write Start Point)가 가리키는 FIFO 공간부터 패킷을 저장하면 된다. 데이터 연결계층은 TSP(Transmit Start Point)가 가리키는 FIFO 공간에 저장되어있는 TLP 부터 물리계층으로 전송하면 된다. 그리고 RSP(Replay Start Pointer)에서 TSP-1 까지의 주소에 저장된 패킷들은 재전송 되어질 TLP들이다. 그림의 예제와 같이 먼저 5개의 TLP가 FIFO에 저장되어 있다고 가정하자. 이 경우 전송계층은 다음 새로운 패킷을 WSP가 가리키는 5번 주소 부터 저장할 것이다. 또한 TLP0, TLP1, 그리고 TLP2는 이미 전송되어 졌지만 상대 디바이스에 의해 아직 승인이 나지 않은 패킷이라고 가정하자. 이 경우 데이터 연결계층은 TSP가 가리키는 3번 주소부터 WSP-1인 4번 주소까지의 패킷들을 새로 전송할 수 있다. 또한 데이터 연결 계층은 TLP0, TLP1, TLP2는 승인될 때 까지 절대 폐기해서는 안 된다. 만약 재전송 상황이 발생한다면, 데이터 연결계층은 RSP가 가리키는 0번 주소부터 TSP-1이 가리키는 3번 주소의 패킷 들을 재전송해야 한다. APCE의 송신버퍼는 이러한 구조를 가지며 송신단 전송계층이 WSP를 생성하고 송신단 데이터 연결 계층이 TSP와 RSP를 관리한다. 또한 재전송 지시는 마이크로프로세서에 의해 이루어진다.

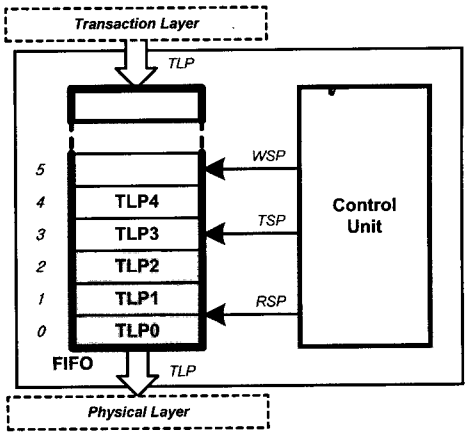

그림 6. 제안된 송신 버퍼의 구조
Fig. 6. Diagram of the proposed transmitter buffer

나. 송신단 전송계층(TxTL)

TxTL은 로컬 마스터 의해 요청된 메모리 읽기 명령어와 메모리 쓰기 명령어를 받아 TLP로 생성하여 송신버퍼에 저장한다. 이때 메모리 읽기 명령어인 경우 상대 디바이스는 완성 요청을 APCE에 할 것이다. 따라서 후에 수신될 완성 요청이 APCE에 의해 요청되었든 메모리 읽기 명령어인지를 확인하여 아닌 경우 에러로 취급을 하여야 한다. 따라서 TxTL은 메모리 읽기 명령어를 전송한 후 해당하는 완성 요청이 수신될 때 까지 TLP 정보를 보관하고 있어야한다.

TxTL은 수신단 데이터 연결 계층에 의해 요청된 완성 요청과 마이크로프로세서로부터 지시된 전송 요청 또한 TLP로 생성하여 송신 버퍼에 저장한다.

데이터 연결계층은 32비트 CRC와 16비트 일련번호를 전송할 TLP에 첨부하여야 한다. 또한 물리계층은 8비트 시작 프레임과 끝 프레임을 TLP의 앞과 끝에 붙여야 하다. 만약 재전송 상황이 발생하면 송신단 데이터 연결계층은 재전송할 TLP에 똑같은 일련번호를 다시 붙이고 LCRC를 다시 계산하여야 한다. 그리고 물리계층은 다시 프레임 정보를 붙여야한다. 따라서 데이터 연결계층은 아직 승인되지 않은 TLP들의 모든 일련번호를 따로 기억하고 있어야 하므로 이는 데이터 연결계층의 설계를 복잡하게 만든다. 그래서 APCE의 전송계

Address	31		0
0	Header	Sequence Number	Start Frame
1	1st Data	Header	
...	.....	1st Data	
...	Last Data	.....	
N-2	LCRC	Last Data	
N-1	End Frame	LCRC	

그림 7. 전송계층 송신단에 의해 송신 버퍼에 저장되는 패킷의 포맷
Fig. 7. Format of packet transferred by TxTL into TxBf.

층은 그림 7과 같이 LCRC, 일련번호, 시작 프레임, 그리고 끝 프레임을 모두 이용하여 TLP를 생성하여 송신 버퍼에 저장하도록 하였다. 따라서 송신단 데이터 연결계층은 일련번호와 LCRC 생성 없이 송신 버퍼에 있는 TLP를 그냥 전송하면 된다. 또한 이 패킷은 32비트 형식으로 정렬되어 있어 버려지는 버퍼 공간을 없앨 수 있다.

다. 송신단 데이터 연결계층(TxDL)

TxDL는 송신 버퍼에 전송해야할 새로운 패킷이 송신 버퍼에 저장되면 상대 디바이스의 수신버퍼 공간을 확인한 다음 TLP를 전송한다. 또한 마이크로프로세서에 의해 재전송이 요청되면 송신 버퍼에 저장된 TLP중 승인되지 않은 모든 패킷을 전송해야 한다. TLP 전송 시 TxDL은 일련번호와 LCRC 생성 없이 저장된 패킷 그대로 바로 물리 계층으로 전송하면 된다.

TxDL은 또한 ACK DLLP와 FC DLLP와 같은 모든 DLLP도 생성 및 전송한다.

라. 수신버퍼(RxBf)

앞에서도 언급했듯이 모든 PCI 익스프레스 수신 디바이스는 흐름제어를 지원하기 위해 그림 3과 같이 각 명령어 별로 모두 3가지 FIFO 버퍼를 가지고 있어야 한다. 그리고 각 버퍼의 사용가능한 공간을 항상 계산하여 주기적으로 상대 디바이스에 보고 하여야 한다. 하지만 이렇게 명령어 별로 버퍼를 각각 따로 두는 경우 같은 명령어들은 수신 버퍼에 저장된 순서를 알 수 있지만 전체 패킷의 수신 순서는 알 수가 없다. 따라서 수신단의 전송계층이 버퍼에 저장된 패킷을 순서대로 처리하는데 어려움이 있다. APCE에서는 그림 8과 같이

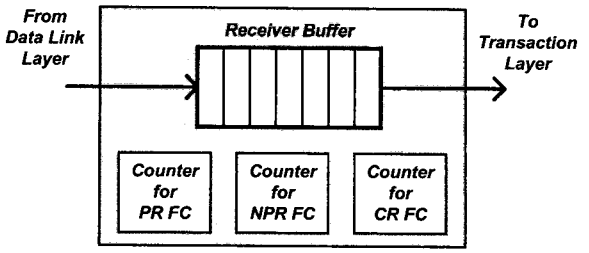

그림 8. 제안된 수신 버퍼의 구조
Fig. 8. Diagram of the proposed receiver buffer.

물리적으로는 FIFO 버퍼를 한개만 사용하여 전송계층이 패킷을 저장된 순서대로 처리하는데 어려움이 없도록 하였다. 대신 수신버퍼에는 3개의 명령어를 위한 공간을 추상적으로 각각 할당하였다. 그리고 각 명령어별로 카운터를 두어 각 명령어의 패킷이 한 개의 버퍼에 얼마나 차있는지를 항상 계산하도록 하였다. 즉 3개의 카운터 값이 각 명령어 별 버퍼 사용 공간을 나타내므로 이를 이용하여 흐름제어를 지원 할 수 있다.

마. 수신단 데이터 연결계층(RxDL)

RxDL은 물리계층으로부터 수신된 TLP의 LCRC와 일련번호를 체크하여 그 결과를 마이크로프로세서에 보고한다. 만약 에러가 있다면 마이크로프로세서는 송신단의 데이터 연결 계층에 NACK DLLP 전송을 요구할 것이다. 반대로 에러가 없다면 ACK DLLP 전송을 데이터 연결 계층에 요청한 후 이를 수신 버퍼에 저장 할 것이다.

RxDL은 수신한 DLLP의 에러를 체크한 후 이 처리하고 그 결과를 마이크로프로세서에 보고하다.

바. 수신단 전송계층(RxTL)

RxDL은 수신 버퍼에 TLP가 저장되면 이 패킷이 APCE에 의해 지원되는 명령어인지를 먼저 확인한다.

수신 패킷이 메모리 쓰기 명령어이면 이를 로컬 슬레이브 디바이스가 지원하는 형태로 변환한 다음 이를 그냥 로컬 인터페이스로 전송 한다. 그러나 메모리 읽기 명령어이면 완성 요청이 요구되므로 RxDL은 로컬 디바이스로부터 데이터를 읽어 완성 버퍼(Completion Buffer)에 저장을 하고 이를 송신단 전송 계층에 알리게 된다. 그러면 이 완성 요청은 원래의 요청 디바이스로 전송되어질 것이다. 다음으로 수신한 패킷이 메시지라면 마이크로프로세서에 보고한다. 마지막으로 수신 버퍼에 있는 패킷이 APCE가 지원하지 않는 명령어이면 이를 마이크로프로세서에 보고한다. 그러면 마이크

로프로세서는 송신단 전송 계층에 에러 메시지 전송을 요청한다.

사. 마이크로프로세서(uP)

APCE는 각 블록을 효과적으로 제어하기 위해 80C51 마이크로프로세서를 내장하여 아래와 같은 서비스를 제공하는 프로그램을 C로 작성하였다.

- 흐름 제어 관리.
- 재전송 타이머를 이용한 재전송 지시.
- 각종 에러 보고를 받아 송신단 전송 계층에 메시지 전송 요청.
- 수신된 TLP에 대한 ACK 혹은 NACK DLLP 전송을 송신단 데이터 연결계층에 요청. 이때 승인 타이머 관리.
- 수신된 메시지의 체크와 이를 컨피규레이션 레지스터에 보고.
- 수신단 ACK와 NACK DLLP의 처리.

2. 검증

먼저 설계된 APCE의 각 블록을 Verilog HDL을 이용하여 코딩하였다. 설계된 APCE를 검증하기위해 그림 9(a)와 같은 테스트 벤치(Test bench)를 제안한다. 테스트 벤치는 호스트 브리지, 로컬 마스터, 그리고 로컬 슬레이브로 구성되어 있으며 이들은 C 언어를 이용하여 클럭 단위로 동작하는 행위 모델(Behavioral model)로 코딩하였다. 호스트 브리지는 전송계층과 데이터 연결계층의 모든 기능을 지원한다. 또한 PCI 익스프레스 모니터(Monitor)를 구현하여 PCI 익스프레스 인터페이스에서 발생하는 프로토콜을 체크하여 에러 발생시 이를 알리는 로그(log)파일을 생성하도록 했다.

설계된 APCE와 각 C 모델들을 그림 9(b)와 같이 PLI(Programming language interface)를 이용해 연결하였다. 여기서 PLI는 Verilog HDL과 C로 코딩된 프로그램을 연결해주는 인터페이스이다[9].

일반적으로 마이크로프로세서는 지원하는 각 명령어에 대한 정확한 동작을 확인함으로써 검증을 할 수 있다. 그러나 APCE와 같은 컨트롤러는 입출력 인터페이스 프로토콜에 의해 동작이 되므로 이를 위한 다른 검증 방법이 요구된다. 따라서 본 논문에서는 APCE의 효과적인 검증을 위한 명령어를 정의 하였다. 이 명령어는 사용자에 의해 테스트벤치의 명령어 파일(Command file)로 작성된다. 이 명령어는 크게 세 가지의 형태로

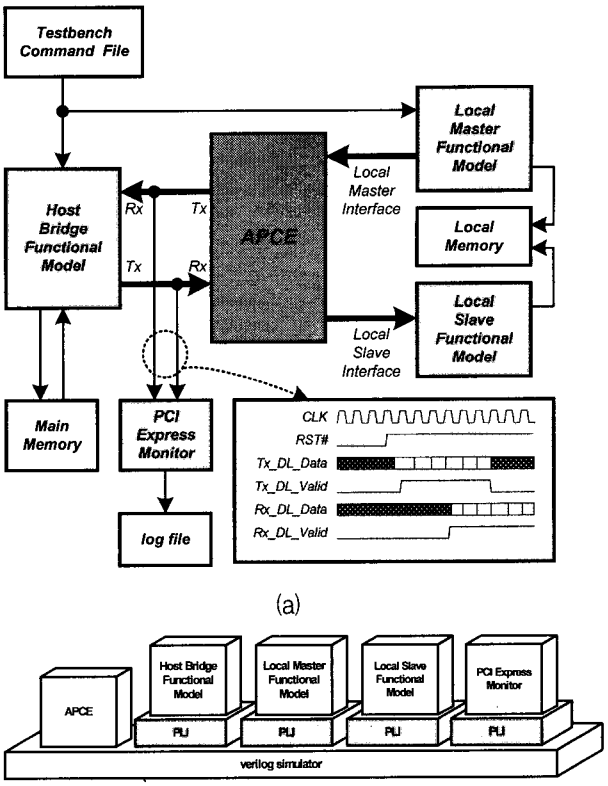

그림 9. 테스트 환경 (a) 테스트 벤치, (b) PLI를 이용한 각 블록들의 연결

Fig. 9. Top-level logic simulation environment, (a) test-bench, (b) interconnection using PLI between functions.

나누어진다.

첫 번째는 테스트 벤치의 호스트 브리지와 로컬 마스터에 데이터 전송을 지시하는 명령어이다.

두 번째 명령어 형태는 테스트 벤치의 메인 메모리와 로컬 메모리에 접근하기 위한 명령어이다. 즉 각 메모리에 특정 또는 임의의 데이터를 저장하거나, 메모리의 데이터 값을 모니터로 확인하거나, 혹은 두 메모리에 저장된 데이터를 비교하기 위해 사용되어지는 명령어이다.

첫 번째와 두 번째의 명령어를 이용한 예제가 아래에 나와 있다.

- MSave 0x300, 0x01234567;
- HB_MWrite 0x100, 0x500, 4;
- HB_MRead 0x300, 0x700, 4;
- LM_MWrite 0x200, 0x600, 8;
- LM_MRead 0x400, 0x800, 8;

첫 번째 'MSave'는 메인 메모리 300번지에 01234567h 데이터를 저장하라는 명령어 이다. 두 번째 'HB_MWrite'는 호스트 브리지에게 메모리 쓰기 명령어를 이용하여 데이터를 전송을 지시하는 명령어이다. 즉 호스트 브리지는 500h 주소가 가리키는 메인 메모리로부터 데이터 4개를 읽은 다음 수신 주소가 100h인 패킷을 생성하여 APCE로 송신하라는 명령어이다. 곧 로컬 슬레이브는 APCE로부터 패킷을 수신 받아 다시 주소 100h가 가리키는 로컬 메모리에 저장한다. 세 번째 'HB_MRead'는 호스트 브리지에 메모리 읽기 명령어를 지시 하는 명령어이다. 마지막으로 'LM_MWrite'와 'LM_MRead' 명령어는 로컬 마스터에 메모리 쓰기와 읽기를 지시하는 명령어이다.

마지막 명령어 타입은 각 인터페이스에서 데이터 전송 시 발생할 수 있는 모든 상황을 설정하기 위한 파라미터를 설정하기 위한 명령어이다. 즉 정의된 각 파라미터를 설정하면 호스트 브리지와 로컬 디바이스는 이 파라미터를 이용하여 다양한 프로토콜로 동작한다. 파라미터의 몇몇 예제가 아래에 소개되어 있다.

- 호스트 브리지와 로컬 마스터가 전송하려는 패킷을 생성하는데 필요한 정보.
- 로컬 인터페이스에서 데이터 전송 시 발생하는 대시 시간(wait phase)
- 호스트 브리지가 APCE로부터 TLP를 수신한 후 NACK DLLP를 송신할 확률.
- 호스트 브리지가 APCE로부터 요청된 메모리 읽기 명령어를 수신한 후 정상적인 완성요청을 전송할 확률.
- 호스트 브리지가 LCRC나 일련번호 에러를 가지고 TLP를 APCE에 전송할 확률.

이러한 명령어들을 이용하여 설계된 APCE는 PCI 익스프레스의 기본적인 명령어 수행이 올바르게 이루어졌는지를 검증하였다.

Ⅳ. 시뮬레이션 결과

그림 10은 로컬 마스터 디바이스가 메모리 쓰기 명령어와 메모리 읽기 명령어를 수행한 시뮬레이션 결과이다. 시스템이 초기화 된 후 APCE와 호스트 브리지는 (a)와 같이 흐름제어를 위해 수신버퍼 정보를 교환한다. 먼저 (b)와 같이 호스트 브리지는 APCE가 정상적으로

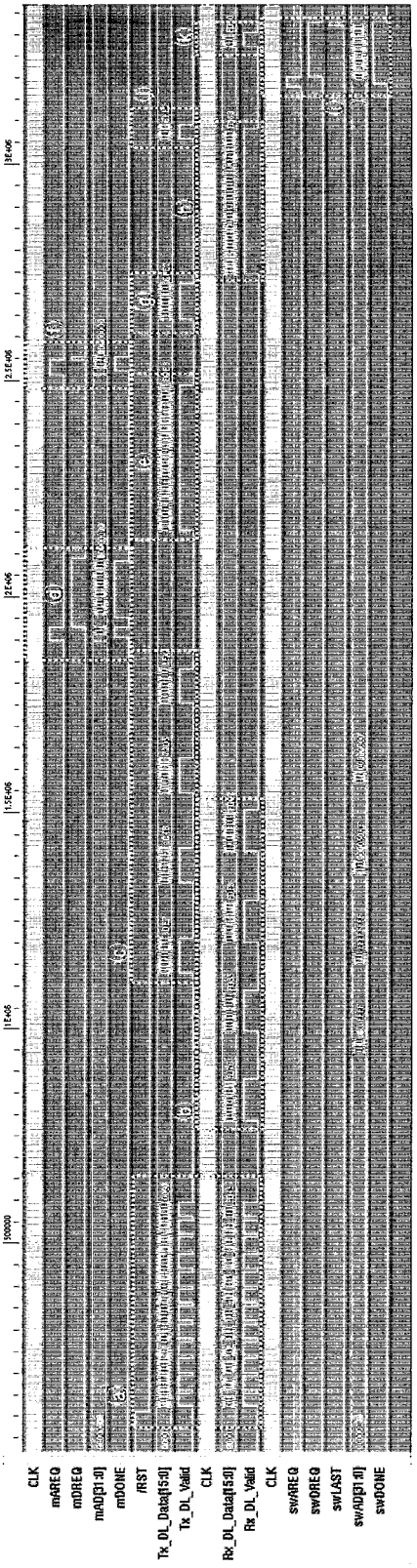

그림 10. 로컬 마스터에 의해 메모리 쓰기와 메모리 읽기 명령어가 요청되어 진 경우의 시뮬레이션 결과 파형.

Fig. 10. The wave of simulation results when the local master device requests Memory Write and Memory Read.

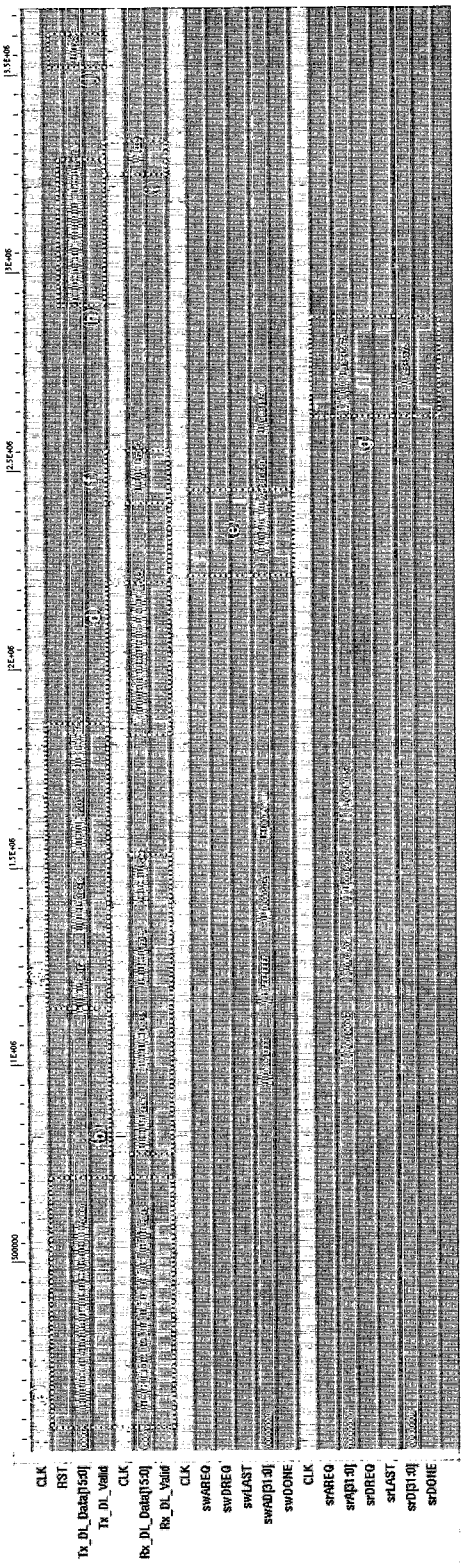

그림 11. 호스트 브리지 의해 메모리 쓰기와 메모리 읽기 명령어가 요청되어 진 경우의 시뮤레이션 결과 파형.

Fig. 11. The wave of simulation results when the host bridge requests Memory Write and Memory Read.

동작하는데 필요한 파라미터를 세팅하기 위해 컨피규레이션 쓰기 명령어를 시작한다. 이 명령어를 수신한 APCE는 얼마 후 (c)와 같이 이에 상응하는 완성 요청을 실행한다. 로컬 마스터는 (d)와 같이 로컬 인터페이스를 통해 메모리 쓰기 명령어를 시작하고 APCE는 TLP를 생성한 다음 (e)와 같이 이를 호스트 브리지에 전송한다. 로컬 마스터는 다시 메모리 읽기 명령어를 (f)와 같이 APCE에 요청을 하고 APCE는 (g)와 같이 호스트 브리지에 이를 전송한다. 그러면 호스트 브리지는 로컬 메모리로부터 데이터를 읽어 (h)와 같이 APCE에 완성 요청을 실행한다. (g)에서 요청한 메모리 읽기에 대한 완성 요청을 호스트 브리지로부터 받은 다음 APCE는 그 데이터를 로컬 슬레이브에 (i)와 같이 전송한다. APCE는 지금 까지 수신한 패킷에 대한 ACK DLLP를 (j)와 같이 전송하고 호스트 브리지 역시 APCE로부터 받는 패킷에 대한 ACK DLLP를 (k)와 같이 전송한다.

그림 11은 호스트 브리지가 메모리 쓰기 명령어와 메모리 읽기 명령어를 수행한 시뮬레이션 결과이다. (a), (b), (c)는 그림 10의 경우와 똑같다. 얼마 후 호스트 브리지는 그림 (d)와 같이 메모리 쓰기 명령어를 PCI Express 인터페이스를 통해 APCE에 요청한다. APCE는 수신한 데이터를 (e)와 같이 로컬 슬레이브에 전송한다. 다시 호스트 브리즈는 (f)와 같이 메모리 읽기 명령어를 APCE에 요청을 하고 APCE는 (g)와 같이 로컬 슬레이브로부터 데이터를 읽어 들인다. APCE는 (g)에서 읽은 데이터를 완성 요청을 통해 호스트 브리지에 (h)와 같이 전송한다. 얼마 후 호스트 브리지는 (i)와 같이 APCE에 ACK DLLP를 전송한다. 마찬가지로 APCE 역시 (j)와 같이 호스트 브리지에 ACK DLLP를 전송한다.

V. 결 론

본 논문에서는 PCI 익스프레스의 전송계층과 데이터 연결계층의 모든 기능을 지원하는 PCI 익스프레스 엔드포인트 컨트롤러인 APCE를 설계하였다. APCE는 재전송 메커니즘을 효과적으로 지원하기 위해 제안된 송신버퍼 구조를 가지고 있다. 또한 APCE의 송신단 전송계층은 제안된 송신 버퍼 구조를 효과적으로 지원하도록 설계되어 졌으며 수신 버퍼는 흐름제어를 효과적으로 지원하도록 설계되어 졌다. 설계된 APCE의 각 블록을 효과적으로 제어하기 위해 80C51 마이크로프로세서

를 내장하여 PCI 익스프레스의 프로토콜을 제공하는 프로그램을 C로 코딩하였다. 설계된 컨트롤러의 검증을 위해 호스트 브리지, 로컬 마스터 디바이스, 로컬 슬레이브 디바이스를 버스 동작 모델로 구성된 테스트 벤치도 제안하였다. 또한 실제 PCI 익스프레스 프로토콜 상에서 발생할 수 있는 모든 경우를 발생 하도록 하기 위해, 각 버스 동작 모델에 필요한 어셈블러 명령어들을 정의 하였다. 제안된 테스트 벤치와 3가지 형태의 이용하여 설계된 APCE가 제대로 동작함을 확인 할 수 있었다.

참 고 문 헌

[1] Intel whitepaper, "Advanced Switching for the PCI Express Architecture", www.intel.com, 2002.

[2] Intel whitepaper, "Creating a PCI Express Interconnect", www.intel.com, 2002.

[3] http://www.pcisig.com

[4] PCI SIG, PCI Express Base Specifications Revision 1.0a, PCI SIG, 2003.

[5] Ravi Budruk, Don Anderson, and Tom Shanley, PCI Express System Architecture, MindShare, 2003.

[6] Intel, "The Complete PCI Express Reference Overview of Tutorial and Book", Intel press on http://www.intel.com/intelpress/pciexpresscomplete/PCIEC_Tutorial.pdf, 2004.

[7] Eugin Hyun, Kwang-Su Seong, "The effective buffer architecture for data link layer of PCI express", ITCC2004, Vol. 1, p.p 809-813, April 2004.

[8] 현유진, 성광수, "PCI익스프레스의 데이터 연결 계층에서 송신단 버퍼 관리를 위한 효과적인 방법", 전자공학회논문지 CI, 제41권, 제5호, p.p 451-458, 2004년 9월.

[9] Cadence, Verilog-XL Reference version 3.4, Cadence, 2002.

저 자 소 개

현 유 진(학생회원)
영남대학교 전자공학과 학사졸업.
영남대학교 전자공학과 석사졸업.
영남대학교 전자공학과 박사졸업.
현재 대구경북과학 기술연구원 (DGIST) 연구원

<주관심분야 : 디지털시스템, PCI 컨트롤러, 무선 통신 시스템>

성 광 수(정회원)
한양대학교 전자공학과 학사졸업.
한양대학교 전자공학과 석사졸업.
한양대학교 전자공학과 박사졸업.
현재 영남대학교 전자정보공학부 조교수

<주관심분야 : 디지털시스템, 집적회로 및 CAD, 임베디드 시스템>