논문 2006-43CI-4-5

차세대 입출력 인터페이스 컨트롤러를 위한 효율적인 기능 검증 환경 구현

(The Design of Efficient Functional Verification Environment for the future I/O Interface Controller)

현 유 진*, 성 광 수**

(Eugin Hyun and Kwang-Su Seong)

요 약

본 논문에서는 차세대 입출력 인터페이스인, PCI 익스프레스 컨트롤러의 기능 검증을 위해 효율적인 검증 환경을 제안하였다. 검증 환경은 벡터 생성 부분, 테스트 벤치, 그리고 두개의 메모리로 구성된다. 이러한 효율적인 검증 환경을 제대로 동작시키기 위한 명령어 세트를 정의하였다 이 명령어는 어셈블리 구조를 가지며, PCI 익스프레스의 모든 프로토콜을 지원하며, 설계된 PCI 익스프레스 컨트롤러를 검증하기 위한 모든 시나리오를 제공하도록 정의되었다. 또한 본 논문에서는 랜덤 벡터 생성 부분, 시뮬레이션 부분, 그리고 비교 엔진으로 구성된 랜덤 검증 환경을 제안하였다. 본 랜덤 테스트 검증 환경에서 수행된 결과는 일반적인 기본 동작 검증과 설계 기반 검증에서 찾지 못한 특수 경우의 에러도 발견 할 수 있다.

Abstract

This paper proposes an efficient verification environment of PCI Express controller that is the future I/O interface. This verification environment consists of a test vector generator, a test bench, and two abstract memories. We also define the assembler set to generate the verification scenarios. In this paper, we propose the random test environment which consists of a random vector generator, a simulator part, and a compare engine. This verification methodology is useful to find the special errors which are not detected by the basic-behavioral test and hardware-design test.

Keywords: 입출력 컨트롤러, PCI 익스프레스, 기능 검증, 랜덤 테스트 검증

I. 서 론

공정 기술과 회로 기술의 급격한 발달은 고성능 마이크로프로세서 출현이 가능케 하였으나 하드웨어의 복잡도는 더욱 증가되었다. 이로 인해 하드웨어의 기능을 검증하는 일은 설계하는 일보다 더욱 어렵고 복잡한 일이 되어가고 있다. 설계되어지는 하드웨어를 검증함에 있어, 가장 핵심적인 단계는 설계 초기에 HDL로 기술된 하드웨어에 대한 RTL(Register Transfer Level) 기능 검증이다. RTL 기능 검증은 게이트 레벨이나 회로 레벨 검증에 비해 훨씬 적은 검증 시간을 가지고도 비교적 정확한 검증을 할 수 있으므로, 하드웨어의 복잡도가 증가 될수록 더욱 중요한 요소가 되었다[1-6].

일반적인 마이크로프로세서는 지원하는 모든 마이크로 명령어에 대해 올바르게 동작하는 지를 확인함으로써 RTL 레벨의 기능 검증을 할 수 있다. 그러나 USB, PCI-X, PCI 익스프레스 등의 데이터 입출력 컨트롤러는 명령어에 기반을 두고 동작을 하는 것이 아니라 외

* 학생회원, 대구경북과학기술연구원 IT 연구부 모바일 멀티미디어 SoC 연구팀
(Daegu Gyeongbuk Institute of Science & Technology, Dep. of IT, Mobile Multimedia SoC Team)
** 정회원, 영남대학교 전자정보공학부
(Department of Electrical Engineering and Computer Science, Yeungnam University)
※ 이 논문은 2005학년도 영남대학교 학술연구조성비 지원에 의한 것임.
접수일자: 2006년3월10일, 수정완료일: 2006년7월2일

부 인터페이스와의 프로토콜에 의해 동작을 하기 때문에 검증을 위한 보다 유용한 방법이 요구된다.

본 논문에서는 차세대 입출력 인터페이스인 PCI 익스프레스 컨트롤러의 기능 검증을 위한 효율적인 검증 환경을 제안하였다. 검증 환경은 벡터 생성 부분, 테스트 벤치, 그리고 메모리로 구성된다. 벡터 생성 부분은 테스트 명령어 파일을 이용하여 테스트 벤치를 위한 2진 코드로 바꾸고 테스트 벤치는 설계된 PCI 익스프레스의 HDL 모델을 검증하기 위해 구현된 주변 디바이스의 동작 모델이다. 또한 검증 결과를 비교하기 위해 메인 메모리와 로컬 메모리를 두었다. 본 논문에서 제안한 검증 환경을 위해 앞선 연구[7]에서 개발한 PCI 익스프레스 컨트롤러를 이용하여 성능과 효율성을 측정하였다.

또한 본 논문에서는 설계된 컨트롤러의 HDL 모델과 행위 모델로 구성된 테스트 환경을 하나의 가상 마이크로프로세서로 가정하고, 이 가상 마이크로프로세서에 사용될 어셈블리 수준의 명령어 세트를 제안한다. 물론 이 명령어들은 PCI 익스프레스의 모든 프로토콜을 지원하며, APCE를 검증하기 위한 모든 시나리오를 제공하도록 정의되었다.

본 논문에서는 일반적인 기본 동작 검증과 설계 기반 검증에서 찾지 못한 특수 경우의 에러를 찾기 위해, 랜덤 벡터 생성 부분, 시뮬레이션 부분, 그리고 비교 엔진으로 구성된 랜덤 검증 환경을 제안한다. 랜덤 벡터 생성을 하기 위해 랜덤 테스트 파라메터를 정의 하였고 설계된 컨트롤러의 참조모델을 설계하여, 랜덤 테스트 벡터를 똑같이 시뮤레이션 되도록 하였다. 수행된 결과는 제안된 메모리 맵에 저장이 되어지고 비교 엔진을 통해 오류가 발견된다.

본 논문 Ⅱ장에서는 PCI 익스프레스 프로토콜에 대해 설명하고 Ⅲ장에서는 제안된 검증 환경에 대해 소개한다. Ⅳ장에서는 제안된 랜덤 테스트 검증을 소개하고 Ⅴ장에서는 결과 및 고찰을, 그리고 Ⅵ장에서 결론을 맺는다.

Ⅱ. PCI 익스프레스 개요

1. 개요

PCI 익스프레스는 차세대 컴퓨터를 위한 새로운 입출력 표준안으로 기존의 버스 병렬 방식이 아닌, 직렬 점대점(point-to-point) 방식을 통해 최소 2.5Gpbs의 전송속도를 가진다.

PCI 익스프레스는 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, 데이터로 구성되며 이를 TLP(Transaction Layer Packet)라 한다. 데이터는 명령어의 종류에 따라 TLP에 포함되어있지 않을 수도 있다.

PCI 익스프레스에서 사용되어지는 명령어는 포스티드 요청{Posted Request), 난포스티드 요청(Non-posted Request), 그리고 완성 요청(Completion Request)으로 크게 3가지이다. 포스티드 요청은 메모리 쓰기 명령어 명령어이다. 난포스티드는 모든 읽기 명령어와 IO 명령어 그리고 컨피규레이션(Configuration) 명령어이다. 메모리 읽기 명령어를 수신한 디바이스는 데이터를 준비하여 이를 원래 요청한 디바이스에 전송하게 되는데 이를 완성 요청이라 한다. IO 쓰기 명령어와 컨피규레이션 쓰기 명령어의 경우, 데이터 전송이 이루어졌음을 알리는 정보를 원래 요청 디바이스에 알리게 된다.

2. 흐름제어(Flow Control)

PCI 익스프레스는 흐름제어를 지원한다. 즉 PCI 익스프레스의 송신단은 TLP를 전송하기 전에 링크 반대편의 수신 디바이스가 전송하려는 데이터를 받을 수 있는 버퍼 공간이 충분한지를 반드시 확인한다[8-9]. 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. 이를 Update Flow Control이라하고 이를 위해 송신단은 타이머를 가지고 있어야 한다. 따라서 수신단은 포스티드 요청(Posted Request), 난포스티드 요청(Non-Posted Request), 완성 요청(Completion Request)을 위한 각각의 수신버퍼를 가지고 있어야 한다[8-9].

2. 흐름제어(Flow Control)

전송계층에 의해 전송되어지는 각 TLP에 데이터 연결계층은 신뢰성을 확보하기 위해 CRC와 일련 번호(Sequence Number)를 첨부한다. 링크 반대편에서 TLP를 수신한 디바이스의 데이터 연결계층은 수신한 TLP의 일련번호가 순차적으로 할당이 되어 있는지와 CRC에 오류가 있는 지를 확인한다. 만약 에러가 발생하지

않았다면 데이터 연결계층은 정상적으로 TLP를 수신하였음을 원래의 송신 디바이스에 알리기 위한 승인 절차로 ACK(Acknowledge) DLLP를 전송한다. 이 승인 절차는 주기적으로 이루어 지며 이를 위해 내부에 승인 타이머(ACK Timer)를 둔다. 만약 수신한 TLP의 일련번호나 CRC에 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 재전송해야된다[8-9]. 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.

3. 승인(Acknowledgement)과 재전송(Replay)

전송계층에 의해 전송되어지는 각 TLP에 데이터 연결계층은 신뢰성을 확보하기 위해 CRC와 일련 번호(Sequence Number)를 첨부한다. 링크 반대편에서 TLP를 수신한 디바이스의 데이터 연결계층은 수신한 TLP의 일련번호가 순차적으로 할당이 되어 있는지와 CRC에 오류가 있는 지를 확인한다. 만약 에러가 발생하지 않았다면 데이터 연결계층은 정상적으로 TLP를 수신하였음을 원래의 송신 디바이스에 알리기 위한 승인 절차로 ACK(Acknowledge) DLLP를 전송한다. 이 승인 절차는 주기적으로 이루어 지며 이를 위해 내부에 승인 타이머(ACK Timer)를 둔다. 만약 수신한 TLP의 일련번호나 CRC에 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 재전송해야된다[8-9]. 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.

3. 설계된 PCI 익스프레스 컨트롤러

그림 1은 앞선 연구[7]에서 HDL로 설계된 PCI 익스프레스 컨트롤러이다. APCE라고 명칭 된 이 컨트롤러는 전송 계층과 데이터 연결 계층의 모든 기능을 제공하도록 설계되었고, 프로토콜을 지원하기 위해 마이크로프로세서가 내장되어있다.

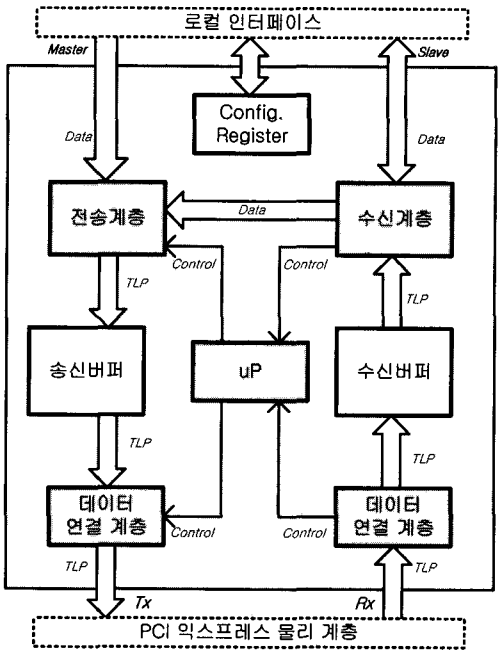

그림 1. 설계된 PCI 익스프레스 컨트롤러
Fig. 1. PCI Express Controller Block-diagram.

III. 검증 환경의 구성

1. 검증 환경의 구조

APCE의 HDL 모델의 효과적인 기능 검증을 위해 테스터 벡터 생성 부분, 테스트 벤치(Test bench), 그리고 메모리로 구성된 검증 환경을 그림 2(a)와 같이 나타내었다. 테스터 벡터 생성 부분은 사용자에 의해 작성된 테스트 명령어 파일을 테스트 벤치를 위한 2진 코드로 바꾸는 부분이다. 메모리는 메인 메모리와 로컬 메모리로 나누어지며, 메모리에 저장된 데이터를 확인 함으로써 시뮤레이션 결과의 오류를 확인 할 수 있다. 테스트 벤치는 호스트 브리지, 로컬 마스터, 그리고 로컬 슬레이브의 동작 모델(Behavioral model)로 이루어진다. 이들 동작 모델은 C 언어를 이용하여 클럭 단위로 동작하도록 설계되었다. 호스트 브리지는 전송계층과 데이터 연결계층의 모든 기능을 지원한다.

APCE와 각 C 모델들을 그림 4(b)와 같이 PLI (Programming language interface)를 이용해 연결하였다. 여기서 PLI는 Verilog HDL과 C로 코딩된 프로그램을 연결해주는 인터페이스이다[10].

일반적으로 마이크로세서는 지원하는 각 어셈블러 명령어에 대한 정확한 동작을 확인함으로써 검증을 할 수 있다[11-12]. 그러나 APCE와 같은 컨트롤러는 입출력 인터페이스 프로토콜에 의해 동작되므로 이를 위한 다

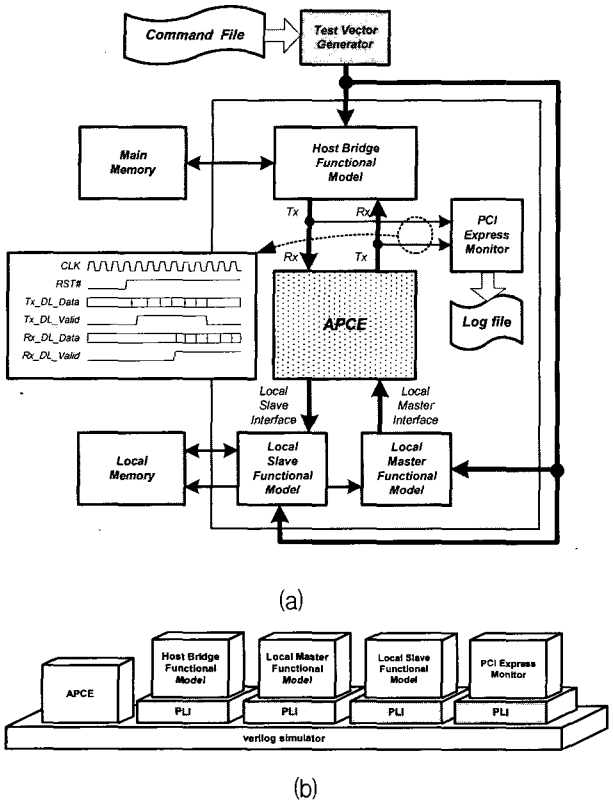

그림 2. 검증 환경, (a) 검증 환경 구조, (b) PLI를 이용한 각 블록들의 연결

Fig. 2. Top level logic simulation environment, (a) the structure of the test environment, (b) interconnection using PLI between functions.

른 검증 방법이 요구된다. 따라서 본 논문에서는 APCE의 HDL 모델과 행위 모델로 구성된 테스트 벤치를 하나의 가상 마이크프로세서로 가정하고, 이 가상 마이크로프로세서에 사용될 어셈블리 수준의 명령어들을 제안한다. 이 명령어들은 PCI 익스프레스의 모든 프로토콜과 APCE를 검증하기 위한 모든 시나리오를 지원하도록 정의되었다.

사용자는 어셈블리 명령어를 이용하여 원하는 시나리오데로 프로그래밍하여 명령어 파일을 만든다. 이 파일은 테스터 벡터 생성기에 의해 2진 코드로 바뀐 후 테스트 벤치의 각 모델로 입력된다. 특히 호스트 브리지와 로컬 마스터는 명령어들을 순서대로 읽어 각각 독립적으로 APCE와 함께 데이터 전송을 수행한다.

2. 테스트 벡트 생성을 위해 사용된 명령어

제안된 어셈블리 명령어는 모두 5가지로 구분된다.

첫 번째는 테스트 벤치의 호스트 브리지와 로컬 마스터에 데이터 전송을 지시하는 명령어로 표1에 나와 있다. 이들 명령어는 아래의 예제 처럼 데이터가 저장된 메모리 소스 주소와 명령어 수행후 저장할 메모리 타겟 주소, 그리고 전송할 데이터의 양을 사용자가 지정할 수 있다.

- *HB_MWrite Dst_Addr, Src_Addr, Data_Cnt;*
- *HB_IOWrite Dst_Addr, Sre_Addr;*

두 번째 명령어 형태는 메인 메모리와 로컬 메모리에 접근하기 위한 것으로 표2에 나타내었다. 특히 *'MSaveDefault'*와 *'LSaveDefault'*는 뒤에 소개될 랜덤 테스트 방법에 유용하게 사용된다. *'CompareMem'* 명령어는 메인 메모리와 로컬 메모리 에 저장된 데이터의 일정 양을 비교하라는 명령어로, 수행 한 후 오류가 있는 경우 로그 파일로 에러를 보고한다. 사용 예가 아래에 나와 있다.

- *MSave Mem_Addr, Data;*
- *MSaveDefault;*
- *MDisplay Mem_Addr, Data_Cnt;*
- *CompareMem Mem_Addr, Data_Cnt;*

세번째 명령어 타입은 PCI 익스프레스 인터페이스, 로컬 마스터 인터페이스, 그리고 로컬 슬레이브 인터페이스에서 데이터 전송 시 발생할 수 있는 모든 시나리오들을 설정하기 위한 명령어로 표3에 나타나있다. 이들 명령어를 통해 호스트 브리지와 로컬 디바이스가 다양한 프로토콜로 동작하도록 지시 할 수 있다.

호스트 브리지가 APCE에 데이터를 전송 할 때 발생할 수 있는 여러 가지 상황을 발생시키기 위해 확률 표현되는 명령어 들이 있다. 호스트 브리지가 APCE로부터 수신한 TLP에 대해 Nack DLLP를 응답할 확률, APCE로부터 수신된 요청에 대해 APCE가 오류를 가지고 완성 요청을 해야 할 확률, 호스트 브리지가 APCE에 패킷을 송신 할 때 일련번호에 오류를 낼 확률, 마지막으로 호스트 브리지가 APCE에 패킷을 송신 할 때 CRC 오류를 낼 확률이 이에 해당한다.

PCI 익스프레스의 프로토콜을 수행하기 위해서는 여러 가지 타이머를 내장하고 있어야 한다. APCE의 경우 마이크로프로세서가 이들을 담당하여 주지만 호스트 브리지의 경우 동작 모델 자체가 이들 값을 계산하여야 한다. 즉, 호스트 브리지의 승인 타이머 주기, 흐름 제어를 위한 Update Flow Control을 해야 되는 주기, 그리고 호스트 브리지가 재전송을 해야 되는 주기를 정하기 위한 명령어가 필요하다.

표 1. 데이터 전송을 위한 명령어
Table 1. Commands for data transfer.

명령어	설명
HB_MWrite	호스트 브리지가 APCE의 메모리 영역에 특정 양만큼의 데이터를 전송하라는 명령어.
HB_MRead	호스트 브리지가 APCE의 메모리 영역으로 부터 특정 양만큼의 데이터를 전송 읽어라는 명령어
HB_CWrite	호스트 브리지가 APCE의 컨피규레이션 레지스터에 데이터를 저장하라는 명령어.
HB_CRead	호스트 브리지가 APCE의 컨피규레이션 레지스터로부터 데이터를 읽어오라는 명령어.
HB_IOWrite	호스트 브리지가 APCE의 I/O 영역에 데이터를 저장하라는 명령어.
HB_IORead	호스트 브리지가 APCE의 I/O 영역으로 부터 데이터를 읽어오라는 명령어.
LM_MWrite	로컬 마스터가 APCE의 메모리 영역에 특정 양만큼의 데이터를 전송하라는 명령어.
LM_MRead	로컬 마스터가 APCE의 메모리 영역으로 부터 특정 양만큼의 데이터를 전송 읽어라는 명령어

표 2. 메인 및 로컬 메모리를 관리하기 위한 명령어
Table 2. Commands to manage main and local memory.

명령어	설명
MSave	지정된 메인 메모리 주소에 특정 데이터를 저장하기 위한 명령어.
MSaveDefault	메인 메모리 전체에 데이터를 무작위로 저장하기 위한 명령어.
MDisplay	지정된 특정 메인 메모리 주소로부터 특정 양만큼의 데이터를 화면에 보여주는 명령어.
LSave	지정된 로컬 메모리 주소에 특정 데이터를 저장하기 위한 명령어.
LSaveDefault	로컬 메모리 전체에 데이터를 무작위로 저장하기 위한 명령어.
LDisplay	지정된 특정 로컬 메모리 주소로부터 특정 양만큼의 데이터를 화면에 보여주는 명령어.
CompareMem	지정된 특정 메인 메모리와 로컬 메모리에 저장된 데이터를 비교하라는 명령어.

네 번째 명령어 타입은 호스트 브리지와 로컬 마스터가 APCE에 데이터 전송을 요청하기 위해 패킷을 생성시, 필요한 파라미터를 사용자가 직접 설정하기 위한 명령어이다. 태그(Tag) 정보 등 모든 헤드의 필드를 사용자가 직접 정할 수 있다. 표4는 그 중 호스트 브리지에 관련된 간단한 예제만을 나타내었다.

다섯 번째 명령어 타입은 제안된 검증 방법을 효과적으로 관리하기 위한 명령어로 표 5에 나와 있다. 이들 명령어를 이용하면 APCE 통해 전송되는 패킷의 이동 흐름을 사용자가 알 수 있어 하드웨어 검증 초반에 아주 유용하게 사용되어 질 수 있다.

그림 3의 테스트 명령어 파일 예제는 간단한 메모리 쓰기 명령어이다. 이 명령어 파일이 컴파일되면 호스트 브리지를 위한 명령어와 로컬 마스터를 위한 명령어로 나누어져, 호스트 모델과 로컬 마스트 모델로 입력된다. 예제를 보면 호스트 브리지는 컨피규레이션 쓰기 명령어 4개를 먼저 수행한다. 그리고 로컬 마스터는 '*Sync*' 명령어에 의해 메모리 쓰기 명령어를 수행하지 않고 기다리게 되는 상황이 발생한다. 그 이유는 시스템이 초기화시, 호스트 브리지가 APCE의 컨피규레이션 레지스터에 쓰기 명령어로 초기화를 다 수행 하고 나서야 비

표 3. 각 인터페이스의 프로토콜을 지원하기 위한 명령어
Table 3. Commands to support both interfaces.

명령어	설명
HB_CmdWaitMax	호스트 브리지가 하나의 명령어를 수행 한 후 다음 명령어를 수행 할 때 까지 기다려야 되는 최대 지연 시간을 설정하기 위한 명령어.
HB_CmdWaitMin	호스트 브리지가 하나의 명령어를 수행 한 후 다음 명령어를 수행 할 때 까지 기다려야 되는 최소 지연 시간을 설정하기 위한 명령어.
HB_Nack_Prob	호스트 브리지가 APCE로부터 패킷을 전송 받은 후 Nack로 응답 할 확률을 설정하기위한 명령어.
HB_CompErr_Prob	호스트 브리지가 APCE로부터 패킷을 전송 받은 후 오류를 가지는 완성 응답을 할 확률을 설정하기위한 명령어.
HB_SNErr_Prob	호스트 브리지가 APCE에 패킷을 전송 할 때 Sequence Number에 오류를 발생 시킬 확률을 설정하기위한 명령어.
HB_CRCErr_Prob	호스트 브리지가 APCE에 패킷을 전송할 때 잘못된 CRC를 발생 시킬 확률을 설정하기위한 명령어.
HB_Ack_Period	호스트 브리지가 APCE로부터 전송 받은 패킷들에 대해 Ack DLLP를 보내야 되는 주기를 설정하기 위한 명령어.
HB_UpFC_Period	호스트 브리지가 APCE에 Update Flow Control을 보내야 되는 주기를 설정하기 위한 명령어.
HB_Replay_Period	호스트 브리지가 APCE에 재전송을 해야 되는 주기를 설정하기 위한 명령어.

표 4. 패킷 생성 시 필요한 파라미터를 위한 명령어
Table 4. Commands to form packets.

명령어	설명
HB_BusNum	호스트 브리지의 Bus Number를 설정하기 위한 명령어.
HB_DevNum	호스트 브리지의 Device Number를 설정하기 위한 명령어.
HB_FuncNum	호스트 브리지의 Function Number를 설정하기 위한 명령어.
HB_LastBE	호스트 브리지가 패킷을 생성 할 때, 헤드의 마지막 Byte Enable 값을 설정하기 위한 명령어.
HB_FirstBE	호스트 브리지가 패킷을 생성 할 때, 헤드의 첫 Byte Enable 값을 설정하기 위한 명령어.
HB_Tag	호스트 브리지가 패킷을 생성 할 때, 헤드의 Tag 값을 설정하기 위한 명령어.
LM_LastBE	로컬 마스터가 패킷을 생성 할 때, 헤드의 마지막 Byte Enable 값을 값을 설정하기 위한 명령어.
LM_FirstBE	로컬 마스터가 패킷을 생성 할 때, 헤드의 첫 Byte Enable 값을 설정하기 위한 명령어.

표 5. 제안된 검증 방법을 효과적으로 관리하기 위한 명령어

Table 5. Commands to manage the proposed verification method.

명령어	설명
End	수행하고자 하는 명령어의 끝을 알리는 명령어.
Sync	호스트 브리지와 로컬 마스터의 명령어 수행 동기를 맞추기 위한 명령어.
LM_TransID	로컬 마스터가 전송하는 패킷의 전송 ID를 설정하기 위한 명령어.
LM_WaitUntilRx	로컬 마스터가 전송한 특정 전송 ID에 해당하는 패킷을 호스트 브리지가 수신할 때 까지 기다렸다가 다음 명령어를 수행하라는 명령어.
LM_WaitUntilCmp	로컬 마스터가 전송한 특정 전송 ID에 해당하는 패킷이 완성 요청으로 로컬 슬레이브에 수신될 때 까지 기다렸다가 다음 명령어를 수행하라는 명령어.
HB_TransID	호스트 브리지가 전송하는 패킷의 전송 ID를 설정하기 위한 명령어.
HB_WaitUntilRx	호스트 브리지가 전송한 특정 전송 ID에 해당하는 패킷을 로컬 슬레이브가 수신할 때 까지 기다렸다가 다음 명령어를 수행하라는 명령어.
HB_WaitUntilCmp	호스트 브리지가 전송한 특정 전송 ID에 해당하는 패킷이 완성 요청으로 수신될 때 까지 기다렸다가 다음 명령어를 수행하라는 명령어.

로소 로컬 마스터가 데이터 전송을 시작해야 하기 때문이다. 이러한 상황을 연출하기 위해 '*Sync*' 명령어를 이용한다.

마지막 명령어는 호스트 브리지에 의해 요청된 패킷이 제대로 전송되고 있는지를 확인하기 위한 명령어이다. '*HB_TransID*' 명령어는 호스트 브리지에 전송할 패킷에 ID로 실제 패킷에 붙는 것은 아니고 단순히 테스트를 위해 가상으로 붙이는 ID이다. 사용 예제가 아래에 나와 있다.

- *HB_WaitUntilRx HB_TransID;*
- *HB_WaitUntilComp HB_TransID;*

그림 4(a) 보면, 호스트 브리지 송신단은 APCE에 명령어를 요청하면서, 해당 ID와 헤드 정보를 아웃스탠딩 큐에 저장을 해둔다. 로컬 슬레이브는 APCE로부터 패킷을 수신하면, 헤드 정보를 이용하여 아웃스탠딩 큐를 검색하게 되고, 상응하는 전송 ID가 있는 경우 정상적으로 수신되었음을 플래그 레지스터에 세팅해둔다. 즉, '*HB_WaitUntilRx*' 명령어는, 호스트 브리지가 해당 전송 ID의 플래그 레지스터가 1이 될 때 까지 다음 명령어를 수행하기 않고 대기하라는 명령어이다. 이를 이용하여 호스트 브리지에 의해 요청된 명령어가 APCE를 통해 로컬 슬레이브에 제대로 수신되었는지를 확인할 수 있다. 그림 4(b)는 '*HB_WaitUntilComp*' 명령어를

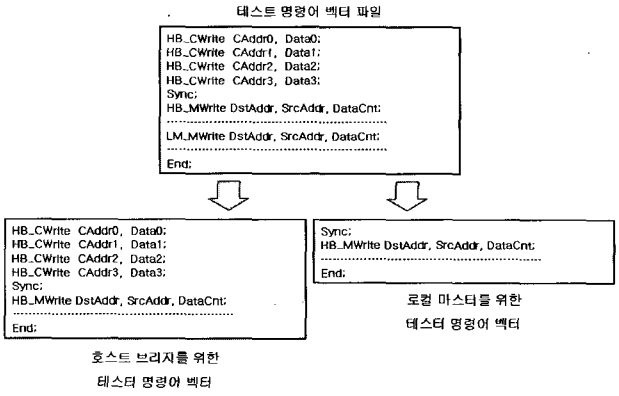

그림 3. Sync 명령어의 사용 예

Fig. 3. Example for Sync command.

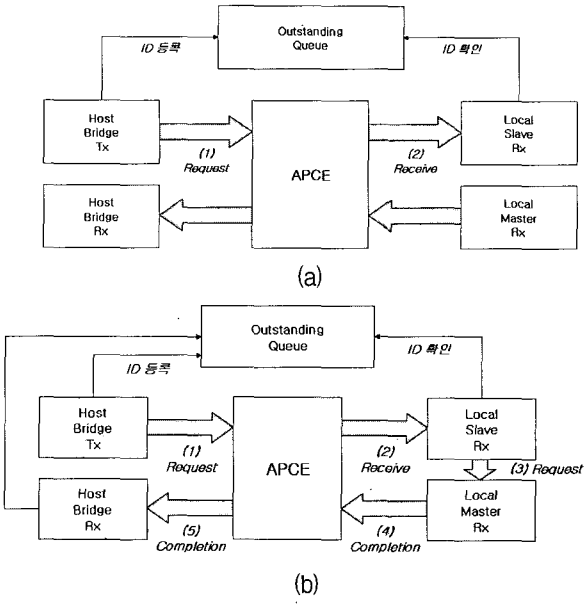

그림 4. 전송 ID를 이용한 패킷 트랙, (a) 수신 확인, (b) 완성 확인

Fig. 4. Tracking packet using transaction ID, (a) in case tracking the receive state, (b) in case tracking the completion state.

사용하는 경우를 나타내다. 호스트 브리지에 의해 요청된 난포스티드 명령어를 로컬 디바이스가 APCE를 통해 호스트 브리지로 완성되어 졌는지를 확인하기 위한 명령어이다. 앞에서와 마찬가지로 호스트 브리지 송신단은 난포스티드 명령어를 전송시 해당 ID와 헤드 정보를 아웃스탠딩 큐에 저장하게 된다. 호스트 브리지 수신단은 상응하는 전송 ID의 완성 요청을 수신 할 때 까지 다음 명령어를 수행하지 않은 체 대기하게 된다.

Ⅳ. 제안된 랜덤 테스트 검증

설계된 APCE를 효과적으로 검증하기 위해 3단계 방

법으로 일반적인 방법인 기본 동작 검증과 설계 기반 검증을, 그리고 본 논문에서 제안한 랜덤 테스트 검증이다.

1. 기본 동작 검증과 설계 기반 검증

기본 동작 검증(Basic-behavioral verification)은 APCE가 PCI 익스프레스 명령어와 프로토콜을 올바르게 지원하는 지를 검증하는 단계이다. 아래에 명령어 파일의 간단한 예제가 나와 있고, 이 명령어들이 테스트 벤치에서 수행되는 과정을 그림 5에 나타내고 있다.

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

먼저 '*MSave*'는 그림 5(a)와 같이 메인 메모리 500번지에 01234567h 데이터를 저장하라는 명령어이다. 두 번째 '*HB_MWrite*'는 호스트 브리지가 500h 주소가 가리키는 메인 메모리로부터 데이터 4개를 읽어, 수신 주소 100h인 패킷을 생성하여 APCE로 송신하라는 명령어이다. 곧 로컬 슬레이브는 APCE로부터 패킷을 수신 받아 주소 100h가 가리키는 로컬 메모리에 저장한다. 세 번째 '*HB_MRead*'가 수행되면 그림 5(b)에서와 같이 호스트 브리지는 APCE에 500h 주소와 함께 메모리 읽기 명령어를 요구한다. APCE는 다시 로컬 슬레이브에 이를 요청하고 로컬 슬레이브는 로컬 메모리 500h 번지로부터 데이터를 4개를 읽는다. 로컬 마스터는 APCE를 통해 호스트 브리지에 완성 요청을 전송하면 메인 메모리 700h 번지에 데이터를 저장하게 된다.

'*LM_MWrite*'와 '*LM_MRead*' 명령어는 로컬 마스터에 메모리 쓰기와 읽기를 지시하는 명령어로, '*MSave*', '*HB_MWrite*', '*HB_MRead*'와 동시에 수행된다.

기본 동작 검증이 PCI 익스프레스 스펙에 명시된 기본 명령어들을 제대로 수행하는지를 확인한 것이라면, 설계 기반 검증(Hardware-Design verification)은 설계되어진 하드웨어 구조들을 하나씩 검증하는 단계이다. 제안된 APCE의 데이터패스(Datapath)와 컨트롤유닛(Control Unit)의 모든 패스(path), 특히 상태도(State machine)의 모든 분기를 트랙(Track)할 수 있도록 검증한다. 이는 APCE의 데이터패스, 컨트롤유닛, 그리고 상태도에서 실제로 절대로 발생하지 않는 조건들이 설계

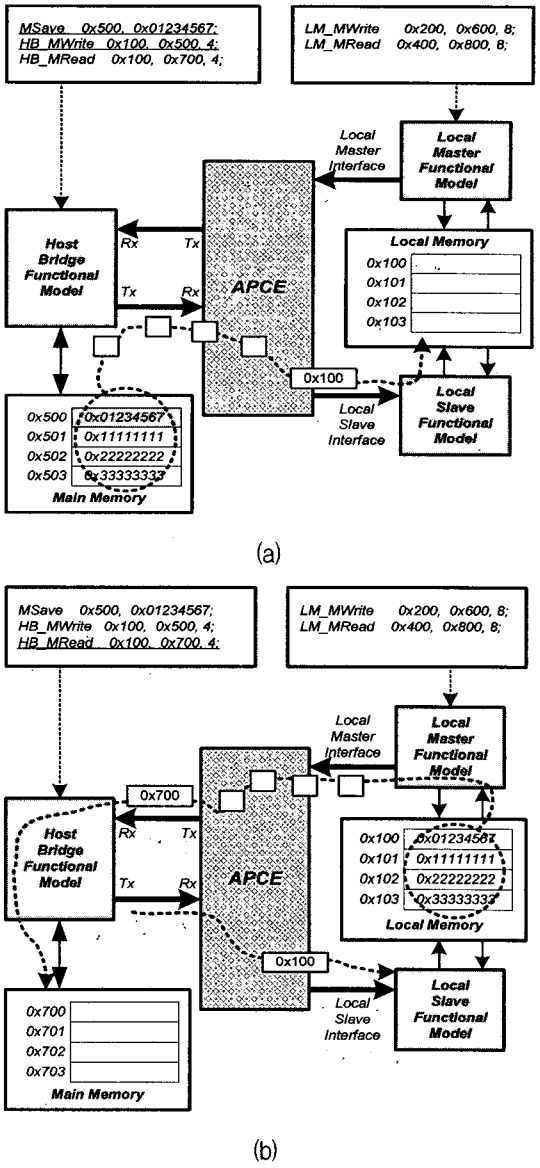

그림 5. 예제 명령어 수행 과정, (a) 호스트 브리지가 메모리 쓰기 명령어를 요청한 경우, (b) 호스트 브리지가 메모리 읽기 명령어를 요청한 경우

Fig. 5. Executing process of commands, (a) in case the Host bridge requests Memory Write command, (b) in case the Host bridge sends Memory Read request.

단계에서 삽입되었는지를 혹인 필요한 패스가 빠지지 않았는지를 검증하기 위해서 이다.

지금까지의 기본동작 검증과 설계 기반 검증은 일반적인 검증으로써, 사용자가 결과를 이미 예측한 상태로 명령어 파일을 만들어 시뮬레이션 한다. 하지만 사용자가 예상치 못한 코너 케이스 등의 시뮬레이션이 이루어지지 않는다면 검증이 충분히 이루어 졌다고 할 수 없다. 이론적으로 완벽한 검증의 신뢰성을 얻기 위해서는

존재 가능한 모든 명령어의 조합 유형에 대한 시뮬레이션이 이루어져야 하지만, 사실 이것은 실제적으로 불가능하다. 이를 위해 본 논문에서는 랜덤 테스트 검증 방법을 제안한다.

2. 제안된 랜덤 테스트 검증

본 논문에서는 PCI 익스프레스 프로토콜을 효과적으로 검증하기 위한 랜덤 테스트 환경을 그림 6과 같이 소개한다. 이는 랜덤 벡터 생성 부분, 시뮬레이션 부분, 그리고 비교 엔진으로 구성되어있다.

랜덤 벡터 생성 부분에서는 랜덤 벡터 파라메터를 설정하면 랜덤 벡터 생성기에 의해 자동으로 테스트 명령어 파일로 바뀌게 된다. 시뮤레이션 부분은 테스트 벤치 부분과 APCE 참조 모델 부분으로 나누어진다. 테스트 벤치 부분은 그림 2(a)의 메인 메모리와 로컬 메모리를 포함한 부분을 나타낸 것이고, APCE의 참조 모델은 단순한 명령어 수준에서만 돌아가는 C 언어로 구현된 행위 모델로, 이 참조 모델 역시 내부엔 메인 메모리와 로컬 메모리를 가지고 있다. 테스트 벤치와 APCE 참조 모델은 랜덤 벡터 생성 부분에서 생성된 테스트 명령어 파일을 똑같이 각각 입력 받아 시뮬레이션을 하고, 그 결과를 각각의 메모리에 저장하게 된다.

비교 엔진 부분은 시뮬레이션이 끝난 후 테스트 벤치의 메인 메모리와 로컬 메모리, 참조 모델의 메인 메모리와 로컬 메모리를 각각 비교하여, 오류가 발생 했을 때 그 결과를 로그 파일로 남기게 된다.

일반적으로 검증 방법은 설계자가 미리 발생 가능한 오류 상황을 예측하여 입력 테스트 벡터를 생성하는 방법이 있다. 이 방법은 사용자가 오류 검출에 유리한 방향으로 명령어 조합을 조절하기 힘들기 때문에 부적절하다[11-12]. 반면 랜덤 테스트 방법은 작성이 매우 간단하고, 설계자가 미리 예측하지 못한 부분의 오류를 찾아 낼 수 있는 장점이 있다. 그러나 실제 오류를 검출하는 테스트 벡터 표본은 모집합에 대하여 균일하게 분포하지 않고 특정 유형과 강한 상관관계를 지니는 경우가 대부분이다[11-12]. 따라서 완전한 랜덤 벡터의 경우 이러한 상관관계를 제대로 반영하지 못하기 때문에 효율성이 매우 떨어지는 단점이 있다.

따라서 본 논문에서는 위 두 방법의 장점을 모두 이용하여, 사용자가 미리 검증하고자 하는 기능을 설정하면 그 설정된 범위 내에서 랜덤 테스트 벡터가 생성되는 방법을 제안한다. 이를 위해 필요한 명령어는 앞에서 소개되어진 표2와 표3, 그리고 랜덤 테스트를 위해 제안된 표6의 랜덤 벡터 파라메터이다. 표6의 명령어는 호스트 브리지가 데이터 전송을 위해 수행할 명령어의 전체 개수, 호스트 브리지가 전송할 데이터 최대 및 최소 개수, 로컬 마스터가 데이터 전송을 위해 수행할 명령어의 전체 개수, 로컬 마스터가 전송할 데이터 최대 및 최소 개수를 세팅할 수 있다. 또한 호스트 브리지나 로컬 마스터가 수행할 메모리 쓰기 명령어와 메모리 읽기 명령어 중 메모리 쓰기 명령어를 수행할 확률을 세팅할 수 있다. 결국 이들 명령어에 의해 세팅된 정보를 바탕으로 랜덤 발생기는 명령어 파일을 생성하게 된다.

그림 7은 랜덤 벡터 파레미터 사용 예제이다. 예제에

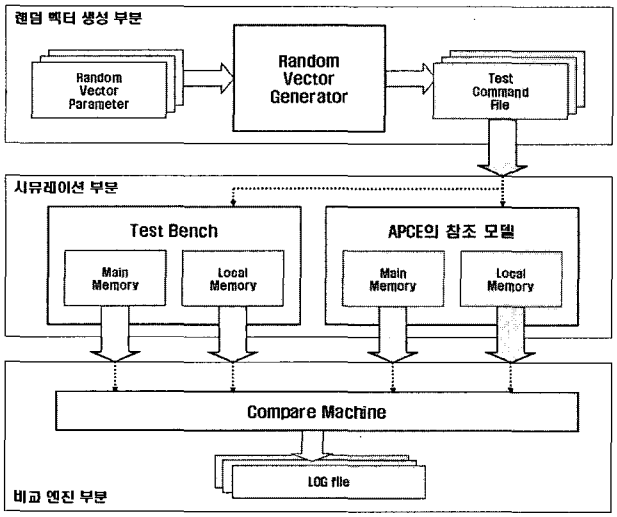

그림 6. 랜덤 테스트를 위한 검증 환경
Fig. 6. Verification environment for random test.

표 6. 랜덤 테스트 검증을 위해 랜덤 벡터 파라메터
Table 6. Parameter of random test.

명령어	설명
HB_CmdNum	호스트 브리지에 의해 APCE에 요청되는 명령어의 수를 성정하기 위한 파라메터.
HB_DataCntMax	호스트 브리지에 의해 APCE에 전송 요청 가능한 데이터의 최대 수를 성정하기 위한 파라메터.
HB_DataCntMin	호스트 브리지에 의해 APCE에 전송 요청 가능한 데이터의 최소 수를 성정하기 위한 파라메터.
HB_MWriteProb	호스트 브리지가 APCE에 요청하는 메모리 쓰기 요청과 메모리 읽기 요청 중 메모리 쓰기 명령어가 차지하는 비율을 나타내는 를 성정하기 위한 파라메터.
LM_CmdNum	로컬 마스터에 의해 APCE에 요청되는 명령어의 수를 성정하기 위한 파라메.
LM_DataCntMax	로컬 마스터에 의해 APCE에 전송 요청 가능한 데이터의 최대 수를 성정하기 위한 파라메터.
LM_DataCntMin	로컬 마스터에 의해 APCE에 전송 요청 가능한 데이터의 최소 수를 성정하기 위한 파라메터.
LM_MWriteProb	로컬 마스터가 APCE에 요청하는 메모리 쓰기 요청과 메모리 읽기 요청 중 메모리 쓰기 명령어가 차지하는 비율을 나타내는 를 성정하기 위한 파라메터.

서 보면 데이터 전송 관련 파라미터와 프로토콜 관련 파라미터가 세팅되어 있다. 사용되어질 데이터 전송 명령어는 호스트 브리지의 경우 100개, 로컬 마스터인 경우 80개이다. 또한 호스트 브리지는 1~4096 바이트만큼을, 로컬 마스터의 경우에는 1~1024 바이트만큼 데이터 전송이 허락된다. 또한 호스트 브리지의 경우 메모리 쓰기 명령어와 메모리 읽기 명령어가 똑같이 50%의 확률로 생성되도록 하였으며 로컬 디바이스의 경우에는 메모리 쓰기 명령어가 메모리 읽기 명령어에 비해 20% 높은 확률로 생성되도록 되어 있다. 그리고 나머지 프로토콜을 위한 파라미터도 설정되어 있다.

이렇게 작성된 명령어는 랜덤 발생기를 통해 그림 8과 같은 테스트 명령어 파일로 생성된다. 먼저 호스트 브리지가 패킷을 생성하기 위해 필요한 파라미터를 세팅을 하고 메인 메모리와 로컬 메모리에 데이터를 무작위로 저장시킨다. 이때 두개의 메모리 모두에 데이터 저장이 완료 될 때 까지 로컬 마스터와 호스트 브리지가 프로토콜을 시작하지 못하도록 동기 명령어 '*Sync*'를 이용하였다. 다음으로 호스트 브리지는 APCE의 컨피규레이션 레지스터에 기본적인 제어값 들을 세팅한다. APCE의 컨피규레이션 레지스터 세팅이 완료 될 때까지 로컬 마스터가 프로토콜을 시작하지 못하도록 명령어 '*Sync*'를 이용하였다. 그 다음부터 호스트 브리지에 의한 데이터 전송과 로컬 마스터에 의한 데이터 전송 명령이 기술 되어 있다.

이렇게 랜덤 벡터 생성기를 통해 생성된 랜덤 테스트 벡터 파일은 테스트 벤치에 입력된 후, 명령어 수행에 따라 메인 메모리와 로컬 메모리간의 데이터 이동을 수행한다. 이때 똑같은 테스트 벡터 명령어는 APCE의 참조 모델에도 인가되어 독립적으로 시뮬레이션을 수행한다. APCE의 참조 모델의 경우 실제 설계된 APCE와 달리 단순히 명령어 기반으로 동작을 하기 때문에 훨씬 더 빨리 명령어를 수행하게 된다. 똑같은 테스트 벡터를 APCE의 HDL 모델과 참조모델이 오류 없이 수행하였다 하더라도 메모리에 데이터를 저장하는 시간 순서가 서로 다르게 된다. 따라서 그림 9와 같이 메모리 맵을 구성하여 이용하여 이러한 오류를 막을 수 있다. 테스트 벤치와 참조 모델 모두 이와 같은 메모리 구조를 가진다. 메인 메모리와 로컬 메모리는 호스트 영역과 로컬 영역으로 나누어진다. 호스트 영역은 호스트 브리지가 메모리 쓰기 명령어를 수행시에는 소스 영역으로, 메모리 읽기 명령어를 수행 시엔 타겟 영역으로 사용된다. 로컬 영역도 마찬가지이다.

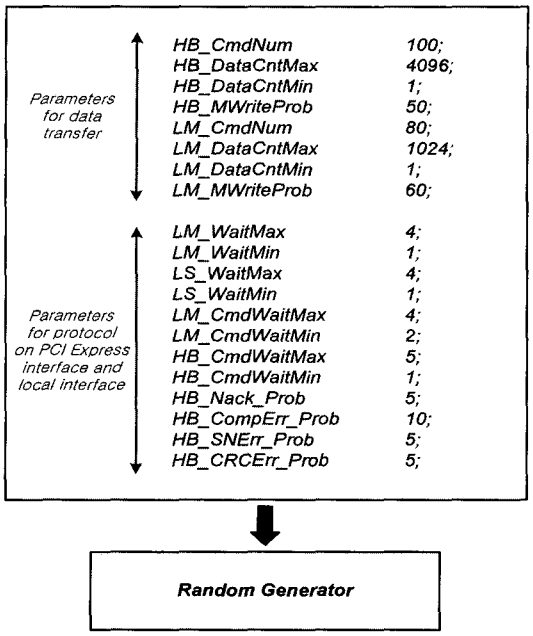

그림 7. 랜덤 벡터 파레미터 사용 예
Fig. 7. Example of random vector parameter.

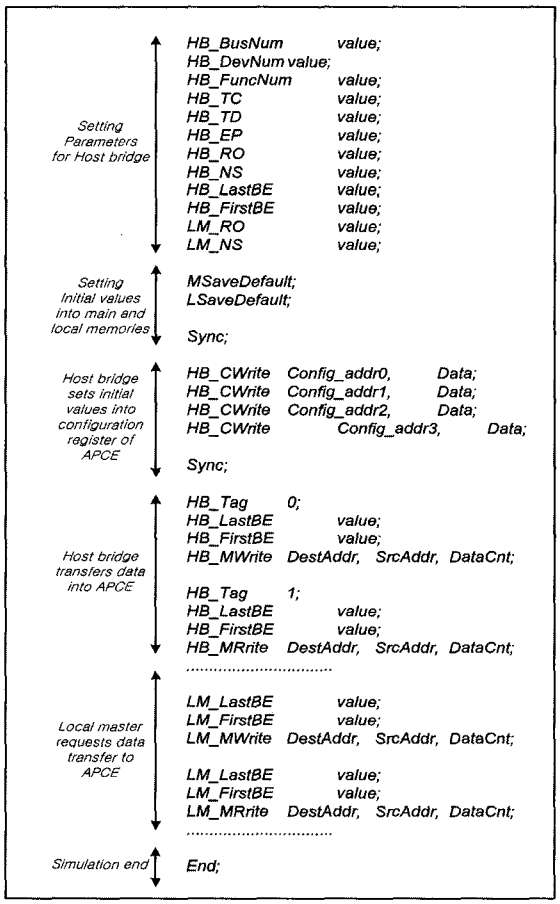

그림 8. 랜덤 벡터 생성기를 이용해 생성된 랜덤 테스트 벡터 예
Fig. 8. Random test vector using random vector generator.

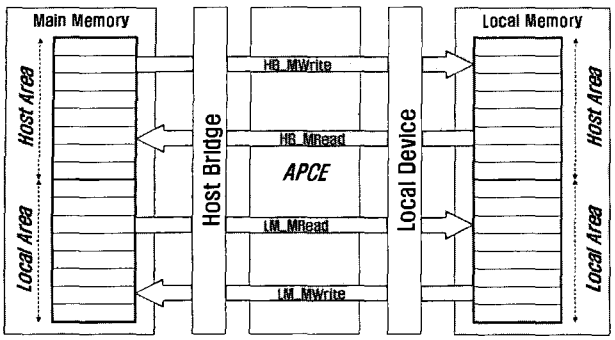

그림 9. 랜덤 테스트를 위한 메모리 맵 구성
Fig. 9. Memory map for random test.

테스트 벡터의 모든 명령어를 수행하고 나면 마지막으로 비교엔진은 검증 목적인 APCE의 시뮬레이션 결과와 참조모델의 시뮬레이션 결과를 비교하게 된다. 즉 각 메모리 영역을 비교하여 로그 파일을 생성한다.

로그 파일에 표시된 오류 위치와 명령어 테스트 파일 내에서 그 오류 위치에 해당하는 명령어, 그리고 시뮬레이션 결과 파형을 이용하여 오류가 발생한 부분을 쉽게 찾을 수 있다.

V. 결과 및 고찰

먼저 제안된 APCE는 기본 동작 검증과 설계 기반 검증을 실행하였다. 표 6은 검증 결과로 나타난 오류들을 설계 관점에서 통계 내어 놓은 것이다. 외부 인터페이스의 프로토콜을 관리하고 APCE의 데이터 패스를 제어하기 위한 상태도에서 가장 많은 오류가 발생함을 확인할 수 있다.

다음은 랜덤 테스트 검증을 위해 10,000개의 데이터 전송 명령어를 수행하여 랜덤 환경에서 시뮬레이션 한 결과를 PCI 익스프레스 스펙 관점에서 표 7에 나타내었다. 즉, APCE의 HDL 모델은 이미 기본 동작 검증과 설계 기반 검증을 완벽하게 수행하였음에도 불구하고,

표 6. 기본 동작 검증과 설계 기반 검증 결과
Table 6. Behavioral verification and hardware design verification.

에러의 유형	개수
코딩 오타	28
데이터 패스의 추가	6
데이터 패스의 신호 수정과 삽입, 그리고 패스 삽입	20
컨트롤 유닛의 신호 수정 및 삽입	9
I상태도에서 상태 추가	8
상태도에서 컨트롤 신호 수정 및 삽입	31
합계	102

표 7. 랜덤 테스트 검증 결과
Table 7. Random test verification.

에러의 유형	개수
PCI 익스프레스의 데이터 연결 개층의 프로토콜 어김	4
전송 계층의 로컬 마스터 인터페이스상에서 프로토콜 어김	1
전송 계층의 로컬 슬레이브 인터페이스상에서 프로토콜 어김	2
컨피규레이션 레지스터 구조 수정	2
마이크로 프로세서를 위한 상태 레지스터 수정	3
기타	4
합계	16

랜덤 테스트 결과 16개의 에러가 발견되었다. 즉, 앞서 동작 검증과 설계 기반 검증을 통해서 찾지 못하는 오류를 랜덤 테스트 검증을 통하면 쉽게 찾을 수 있음을 알 수 있다. 오류는 크게 두 부분으로 나누어지는데, 첫 번째는 프로토콜 어김이다. 일반적인 상황에서는 발생하지 않지만 몇 개의 상황이 아주 공교롭게 겹쳐서 발생하는 경우로, 예를 들어 아주 특수한 상황에서 몇몇 신호가 교착상태가 되어 버리는 경우이다. 이 경우는 설계자가 충분히 PCI 익스프레스의 스펙데로 구현하였음에도 미처 간파하지 못한 부분에서 발생하는 설계상의 오류들이다. 두 번째는 레지스터에 관계되는 에러이다. APCE에서는 컨피규레이션 레지스터와 마이크로 프로세서를 위한 레지스터가 존재하는데, 많은 레지스터가 존재하고 몇몇 레지스터들은 아주 복잡한 상호 관계로 연결되어 있다. 이들 역시 설계자가 충분히 생각하여 설계되어 졌다고는 하나, 생각지 못하는 시나리오에서 오류가 발생한다.

VI. 결 론

본 논문에서는 PCI 익스프레스 컨트롤러의 기능 검증을 위해 효율적인 검증 환경을 제안하였다. 검증 환경은 벡터 생성 부분, 테스트 벤치, 그리고 메모리로 구성된다. 벡터 생성 부분은 테스트 명령어 파일을 이용하여 테스트 벤치를 위한 2진 코드로 바꾸는 부분이다. 그리고 테스트 벤치는 설계된 APCE의 HDL 모델을 검증하기 위해 구현된 주변 디바이스의 동작 모델이다. 또한 검증 결과를 비교하기 위해 메인 메모리와 로컬 메모리를 두었다. 또한 본 논문에서는 이러한 효율적인 검증 환경을 제대로 동작시키기 위한 명령어 세트를 제안하였다. 이 명령어는 어셈블리 구조를 가지며, PCI 익스프레스의 모든 프로토콜을 지원하며, APCE를 검증하

기 위한 모든 시나리오를 제공하도록 정의되었다.

본 논문에서는 일반적인 기본 동작 검증과 설계 기반 검증에서 찾지 못한 특수 경우의 에러를 찾기 위한 검증을 위해, 랜덤 벡터 생성 부분, 시뮬레이션 부분, 그리고 비교 엔진으로 구성된 랜덤 검증 환경을 제안한다. 랜덤 벡터 생성을 하기 위해 랜덤 테스트 파라메터를 정의 하였다. 시뮬레이션 부분에서는 설계된 컨트롤러의 참조모델을 설계하여, 랜덤 테스트 벡터를 똑같이 수행하도록 하였다. 수행된 결과는 제안된 메모리 맵에 저장이 되어지고, 비교 엔진을 통해 오류가 발견된다. 기본 동작 검증과 설계 기반 검증을 모두 마친 후 랜덤 테스트 검증을 통해 16개의 오류를 찾을 수 있었으며, 전체 오류 중 14%를 랜덤 테스트 검증을 통해 찾을 수 있었다.

이러한 검증 방법은 PCI 익스프레스 뿐 아니라, 기존 PCI 버스, ATA, MIPI, Infinite-band 등 각종 버스 혹은 직렬 인터페이스 컨트롤러의 설계에서 아주 유용하게 사용되어질 것으로 본다. 다만 각 프로토콜에 맞는 테스트 명령어를 얼마나 잘 정의하느냐에 따라, 검증 결과의 충실도가 달라질 것으로 생각되어진다.

참 고 문 헌

[1] 김연선, 서범수, "64비트 RISC 마이크로프로세서의 기능 검증에 관한 연구", 대한전자공학회 추계종합학술대회, 755-758쪽, 1998년.

[21 기안도, "단일칩시스템 설계검증을 위한 가상프로토타이핑", 대한전자공학회지, 제30권 9호, 965-975쪽, 2003년.

[3] 이승호, 이현룡, 장종권, "SMV를 이용한 Pipeline 시스템의 설계 검증", 대한전자공학회 하계종합학술대회, 제26권, 제1호, 939-942쪽, 2003년.

[4] C. Pixley, N. Strader, W. Bruce, J. Park, M. Kaufmann, K. Shultz, M. Burns, J. Kumar, J. Yuan, and J. Nguyen, "Commercial Design Verification : Methodology and Tools", Proc. IEEE int. Test Conf., pp. 839-848, 1996

[5] P.J Windley, "Formal modeling and verification of mucroprocessor", IEEE Transactions on Computers, Vol. 44, No. 1, pp.54-72, Jan. 1995.

[6] Ta-Chung Chang, "A Biased Random Instruction Generation Environment for Architectural Verification of Pipelined Processors", in Journal of Electronics Testing : Theroy and Applciations16, pp.13-27, 2000.

[7] 현유진, 성광수, "차세대 통신 플랫폼을 위한 입출력 컨트롤러 설계", 대한전자공학회논문지 CI, 제42권 제4호, 59-68쪽, 2005년.

[8] PCI SIG, PCI 익스프레스 Base Specifications Revision 1.0a, PCI SIG, 2003.

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

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

[11] 권오현, 이문기, "마이크로프로세서를 위한 효율적인 기능 검증 환경 구현", 대한전자공학회논문지 SD, 제41권 7호, 43-52쪽, 2004년.

[12] 권오현, 양훈모, 이문기, "마이크로프로세서 기능 검증을 위한 바이어스 랜덤 벡터 생성기 설계", 대한전자공학회 하계종합학술대회, 121-124쪽, 2002년 6월.

저 자 소 개

현 유 진(학생회원)
영남대학교 전자공학과 학사졸업.
영남대학교 전자공학과 석사졸업.
영남대학교 전자공학과 박사졸업.
현재 DGIST IT 연구부, 모바일 멀티미디어 SoC팀 연구원

<주관심분야 : 디지털시스템, PCI 컨트롤러, 모바일 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>