논문 2007-02-02

PCI 익스프레스 컨트롤러의 통합 설계 및 기능 검증

(A H/W & S/W Co-Design and Functional Co-Verification for PCI Express Controller)

현 유 진*, 성 광 수
(Eugin Hyun, Kwang-Su Seong)

Abstract : 본 논문에서는 차세대 통신 플랫폼을 위한 PCI 익스프레스의 전송계층과 데이터 연결계층의 모든 기능을 지원하는 PCI 익스프레스 컨트롤러를 설계하였다. 설계된 컨트롤러를 효과적으로 제어하기 위해 8051 마이크로프로세서를 이용하였다. 또한, 본 논문에서는 PCI 익스프레스 컨트롤러와 8051 마이크로프로세서의 통합 검증을 위한 방법으로 벡터 생성 부분, 테스트 벤치, 그리고 메모리로 구성된 테스트 벤치를 하나의 가상 마이크로프로세서로 가정하였다. 그리고 PCI 익스프레스의 모든 프로토콜을 지원할 수 있는 어셈블리 수준의 명령어들을 테스트 벤치에 적용되도록 하였다. 특히 일반적인 기본 동작 검증과 설계 기반 검증에서 찾지 못한 특수 경우의 에러를 찾기 위한 검증을 위해 랜덤 검증 환경 및 테스트 파라미터를 정의 하였다. 제안된 검증 환경과 명령어를 통해 설계된 PCI 컨트롤러의 검증 결과 랜덤 테스트 검증을 통해 효과적으로 오류를 찾을 수 있었다.

Keywords : PCI, PCI Express, Co-Design, Co-Verification

1. 서론

오늘날 통신 기술과 컴퓨터 기술은 하나로 통합되고 있다. 즉 일반적인 PC는 물론이고 노트북 그리고 PDA에 이르기까지 유무선 통신 기술이 시스템의 한 부분으로 자리 잡고 있다[1]. 또한 통신 시스템 역시 효과적인 시스템 제어를 위해 컴퓨터 기술을 필요로 하고 있다[1]. 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 인터페이스는 이러한 요구를 충족시킬 수가 없다[1-2].

이에 차세대 컴퓨터 및 통신 시스템을 위한 새로운 입출력 표준안으로 PCI 익스프레스(express)를 소개하였다. 이 방식은 점대점 방식을 이용한 직렬 전송 방식으로 현재 2.5Gpbs의 전송 속도를 가진다. 또한 기존 PCI 장치에 사용되는 S/W를 그대로 사용할 수 있어 차세대 컴퓨터 및 통신 시스템의 입출력 표준안으로 자리 잡을 것으로 보고 있다[3-5].

본 논문에서는 PCI 익스프레스 엔드포인트(end point) 컨트롤러를 설계하였다. 설계된 컨트롤러는 PCI 익스프레스의 전송계층(Transaction Layer)과 데이터 연결계층(Data Link Layer)의 모든 기능을 지원한다. PCI 익스프레스 프로토콜에는 일정한 주기로 보내야하는 패킷들이 있는데, 이를 효과적으로 관리하기 위해서는 마이크프로세서가 필요하다. 이를 위해 8051을 이용하여, 타이머 제어 뿐 아니라 컨트롤러의 각 블록을 효과적으로 관리하도록 하였다. 이 마이크프로세서와 각 기능 블록들을 온 칩(on chip)으로 설계하기 위해서는 마이크로프로세서와 기능 블록들이 필요한 신호를 효과적으로 주고 받을 수 있어야 한다. 하지만 이 경우 많은 신호선이 필요하기 때문에 이를 해결하기 위해 본 논문에서는 메모리-맵-I/O(memory mapped I/O) 방식을 사용하였다.

공정 기술과 회로 기술의 급격한 발달과 하드웨어의 복잡도가 높아짐에 따라, 하드웨어의 기능을 검증하는 일은 설계하는 일보다 더욱 어렵고 복잡한 일이 되어가고 있다. 이 중 가장 핵심적인 단계는 설계 초기에 HDL로 기술된 하드웨어에 대

한 RTL(register transfer level)의 기능 검증이다. RTL의 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[6-7]. 일반적인 마이크로프로세서는, 지원하는 모든 마이크로 명령어에 대해 올바르게 동작하는 지를 확인함으로써 RTL의 기능 검증을 할 수 있다. 그러나 AGP, ATA, USB, PCI 2.2, Infinite Band, PCI-X, PCI 익스프레스 등의 데이터 입출력 컨트롤러는, 명령어에 기반을 두고 동작을 하는 것이 아니라 외부 인터페이스와의 프로토콜에 의해 동작을 하기 때문에 보다 유용한 검증 방법이 요구된다. 특히 설계된 하드웨어가 마이크로프로세서를 내장하는 SoC(system on chip)이거나 또는 마이크로프로세서와 연동해서 동작해야 되는 경우, 마이크로프로세서에 포팅 되는 프로그램과 같이 검증을 할 수 있는 환경이 필요하다.

일반적인 설계흐름에서는 설계의 초기단계에 하드웨어와 소프트웨어를 분할하고 설계를 한다. 이 경우 이들 각 부분을 통합하는 과정에서 오류와 개선점들이 발견되므로 결국 시스템 완성에 비용 상승과 설계지연 등을 초래하게 된다. 따라서 설계 초기 단계부터 하드웨어 소프트웨어를 같이 통합하여 전체 기능을 검증하는 것이 바람직하다[7]. 이러한 통합설계 혹은 동시설계(co-design)를 통해 시스템을 구현하고, 동시 검증 혹은 통합 검증(co-verification)을 통해 설계된 하드웨어와 소프트웨어를 검증한다.

이에 본 논문에서는 통합 설계되어진 PCI 익스프레스 컨트롤러와 하드웨어 소프트웨어 통합 검증 방법에 대해 기술한다.

2. PCI 익스프레스 개요

PCI 익스프레스는 전송계층, 데이터 연결계층, 물리 계층으로 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 명령어이다. 난포스티드는 모든 읽기 명령어와 IO 명령어 그리고 컨피규레이션(configuration) 명령어이다. 마스터 디바이스 송신단에 의해 요청된 난포스티드 명령어는 슬레이브 디바이스에 의해 즉시 데이터 전송이 이루어지는 것이 아니라, 슬레이브 디바이스 자체적으로 데이터를 준비 한 후에 완성 요청을 통해 마스터 디바이스로 데이터를 전송한다.

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[3-5]. 그리고 이러한 흐름제어를 지원하기 위해 수신 디바이스의 데이터 연결계층은 자신의 수신버퍼 크기를 시스템 초기화 때, 그리고 초기화가 끝난 후에도 주기적으로 FC DLLP(flow control data link Layer Packet)를 이용하여 디바이스에 알려주어야 하다. 이러한 주기를 확인하기 위해 송신단은 타이머를 가지고 있어야 한다. 또한 수신단은 3가지 명령어를 위해 각각의 수신버퍼를 가지고 있어야 한다[3-5].

전송계층에 의해 전송되어지는 각 TLP는, 데이터 연결계층에 의해 CRC(cyclic redundancy check) 코드와 일련 번호(sequence number)를 첨부한다. 이는 TLP 전송의 신뢰성을 확보하기 위한 것으로써, 링크 반대편에서 TLP를 수신한 디바이스는 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 이루어지며 이를 위해 내부에 타이머를 둔다. 만약 수신한 TLP의 일련번호나 CRC에 오류가 발생한다면, 수신 디바이스는 NACK(negative ACK) DLLP를 즉시 전송하게 된다. NACK DLLP를 받은 송신 디바이스는, 수신디바이스로부터 아직 승인받지 못한 모든 TLP를 재전송해야 된다[3-5]. 또한 송신 디바이스가 TLP를 전송한 후 일정 시간이 지난 후에도 상대 디바이스로부터 승인 받지 못한 다면, 이때 역시 해당 TLP들을 재전송을 해야 한다. 이를 위해 송신 디바이스는 재전송 타이머를 두어야 한다.

3. 통합 설계

본 논문에서는 PCI 익스프레스 전송계층과 데이터 연결계층의 모든 기능을 지원하는 PCI 익스프레스 컨트롤러를 그림 1과 같이 설계하였다.

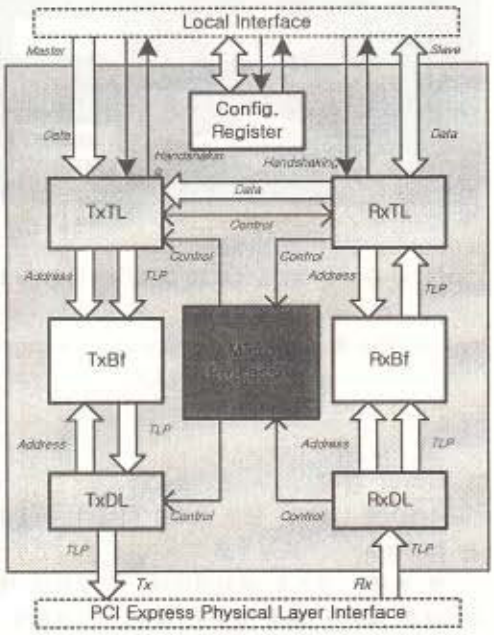

그림 1. 설계된 컨트롤러의 상위 블록
Fig. 1. Block diagram of designed controller

모두 6개 블록인 송신단 전송계층(TxTL), 송신버퍼(TxBf), 송신단 데이터 연결계층(TxDL), 수신단 전송계층(RxTL), 수신버퍼(RxBf), 그리고 수신단 데이터 연결계층(RxDL)으로 나누어진다. PCI 익스프레스의 프로토콜에서는 일정한 주기로 보내야하는 패킷들이 있는데, 이를 효과적으로 관리하기 위해 8051 마이크로프로세서를 내장하였다.

설계되어진 6개의 블록의 기능과 구조는 기존 연구[8]에서 소개하였으며, 본 논문에서는 이들 기능 블록들과 마이크로프로세서가 어떻게 연동 동작하도록 설계되었는지를 기술한다.

마이크로프로세서가 6개의 기능 블록을 효과적으로 관리하기 위해서는, 기능 블록들과 신호를 인터페이스 할 수 있어야 한다. 그러나 제안된 PCI 익스프레스 컨트롤러에 사용되어지는 8051의 입출력 핀 수는 제한되어 있기 때문에 신호선을 직접 연결하여 다른 블록과 인터페이스를 하는데 한계가 있다. 따라서 본 논문에서는 이를 해결하기 위해 그림 2와 같은 하드와이어(hardwired) 방식과 메모리 방식을 응용한 메모리 맵 I/O를 제안한다.

메모리의 각 비트를 I/O 목적으로 정의한 후, 이 비트의 값을 컨트롤 신호로 사용한다. 설계된 PCI 익스프레스 컨트롤러의 각 블록들과 메모리는 레지스터 인터페이스로 되어 있어, 각각의 비트를 컨트롤하는 신호로 메모리의 각 비트를 직접 쓰고 읽을 수 있는 구조로 되어 있다. 하지만 마이크로프로세서에서는 메모리 영역으로 보이므로 8개의 레지스터씩, 8비트씩 데이터를 전송하도록 하였다. 그래서 데이터를 쓰고 읽을 때에는 메모리에 데이터를 읽거나 쓰는 방식으로 관리해야 한다. 또한 마이크로프로세서에 포팅된 프로그램은 모든 레지스터를 주기적으로 스캔(scan)하여 읽혀진 값을 바탕으로 동작하도록 한다.

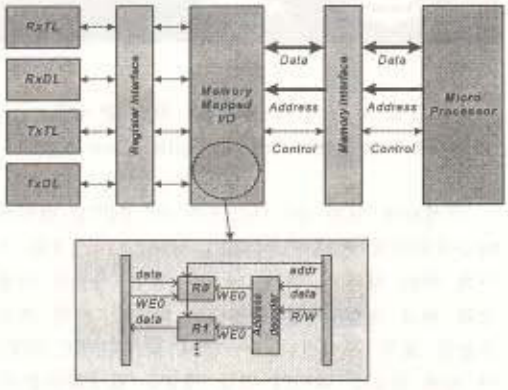

그림 2. 마이크로프로세서와 다른 블록간의 인터페이스를 위한 메모리 맵 I/O 구조
Fig. 2. The block diagram of memory mapped I/O between microprocessor and the other blocks.

앞에서 소개하였듯이 마이크로프로세서에는, PCI 익스프레스 프로토콜을 위해 각 기능 블록을 관리하기 위한 프로그램을 C로 코딩하여 포팅하였고, 그 기능은 다음과 같다

1. 흐름 제어 관리

시스템이 초기화가 되면 데이터 연결 계층은 즉시 흐름제어를 시작하여야 하는데 이를 위한 순서도가 그림 3에 나와 있다.

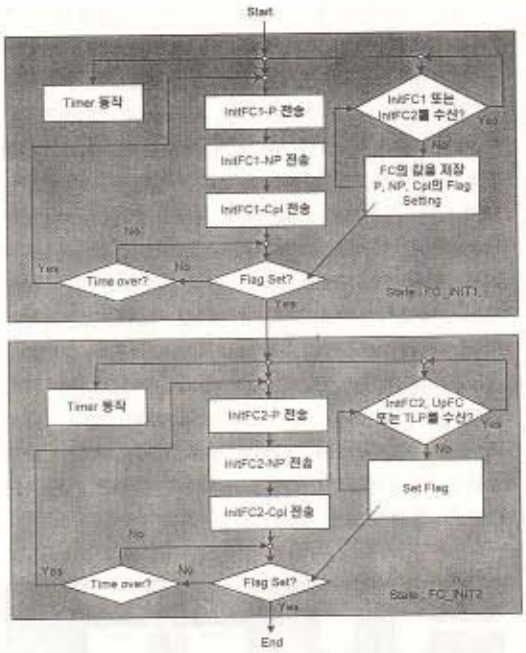

그림 3. 초기 흐름제어를 위한 순서도

Fig. 3. The flow chart of initial flow control

순서도에 보면 FC_INIT1과 FC_INIT2 상태로 나누어지는데 먼저 FC_INIT1 상태는 시스템을 초기화 하는 상태로, 이때 데이터 연결 계층은 비활성화 되고 송신단 전송 계층은 마스터 로컬 프로토콜을 모두 무시한다. 두 번째 상태인 FC_INT2가 되면 비로소 데이터 연결 계층은 활성화되는데, 마이크로프로세서는 수신되는 FC와 송신되는 FC를 관리하여 초기 흐름제어가 완성되면 다른 모든 블록에 데이터 전송이 정상적으로 이루어질 수 있음을 알려야한다.

2. 수신한 ACK와 NACK DLLP의 처리

수신단 데이터 연결계층은 수신된 ACK DLLP와 NACK DLLP의 오류 검사를 한 후 이상이 없는 경우에 마이크로프로세서에 보고한다. 이때 마이크로프로세서는 ACK DLLP와 NACK DLLP가 정상적인 일련번호를 가지는지를 확인하게 된다. 일련번호가 이상이 없다면 이를 송신단 데이터 연결 계층에 알려 재전송 버퍼에 있는 TLP를 폐기하도록 지시한다. 그림 4는 이를 설명하고 있는 순서도이다.

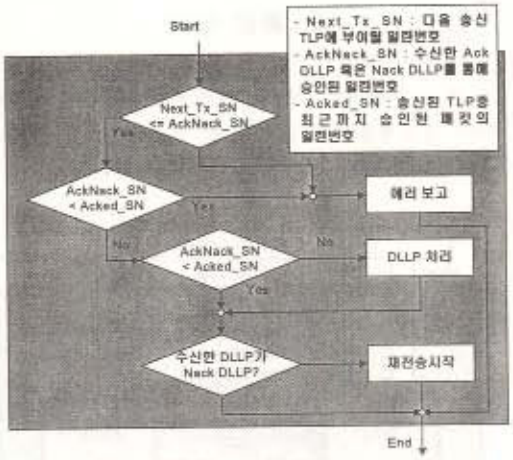

그림 4. 수신한 ACK/NACK DLLP 처리를 위한 순서도

Fig. 4. The flow chart of process for received ACK/NACK DLLP

3. 타이머 관리

마이크프로세서는 모두 3개의 타이머 인터럽트를 이용한다.

첫 번째는 재전송 타이머이다. 즉 송신단 데이터 연결 계층에 의해 새로운 데이터가 전송이 되면 재전송 타이머를 시작한다. 그리고 인터럽트가 걸리는 경우 송신단 데이터 연결 계층에 재전송을 지시하여야 한다.

두 번째는 승인 타이머 관리이다. 수신단 데이터 연결 계층은 수신된 TLP가 일련번호와 CRC 오류 여부를 마이크로프로세서에 보고 한다. 만약 수신한 TLP에 오류가 없고 승인 타이머에 인터럽트가 걸리면 마이크로프로세서가 ACK DLLP 전송을 지시하게 된다. 반대로 에러가 있는 경우, 승인 타이머에 상관없이 마이크로프로세서는 즉시 송신단 데이터 연결 계층에 NACK DLLP 전송을 지시한다.

마지막으로 흐름 제어용 타이머이다. 시스템 초기화와 그 후에도 주기적으로 수신단의 버퍼 크기를 상대 디바이스에 알려야 되는데 마이크로프로세서는 타이머를 이용하여 FC DLLP를 송신하는 시점을 확인한다.

4. 통합 검증 환경

제안된 PCI 익스프레스 컨트롤러 HDL 모델의 효과적인 기능 검증을 위한 검증 환경이 그림 5와 같이 나타나있다. 검증 환경은 테스터 벡터 생성 부분, 테스트 벤치(test bench), 그리고 메모리로 구성된다.

테스터 벡터 생성 부분은 사용자에 의해 작성된 테스트 명령어 파일을 테스트 벤치를 위한 코드로 바꾸는 부분이다. 메모리는 메인 메모리와 로컬 메모리, 그리고 프로그램 메모리로 나누어진다. 시뮬레이션 후의 오류 여부는 메인 및 로컬 메모리에 저장된 데이터를 통해 확인 할 수 있다. 프로그램 메모리는 마이크로프로세서가 동작하기 위한 프로그램이 저장되어있다. 테스트 벤치는 호스트 브리지, 로컬 마스터, 그리고 로컬 슬레이브의 동작 모델로 이루어져 있고, 이들은 C 언어를 이용하여 블록 단위로 동작하는 행위 모델(behavioral model)로 설계되었다.

그리고 8051 마이크로프로세서는 PCI 익스프레스의 흐름제어 관리, 수신한 패킷의 승인처리, 각 프로토콜 지원을 위한 타이머 관리 등을 수행한다.

제안된 PCI 익스프레스 컨트롤러, 8051, 그리고 각 블록 C 모델들은 PLI (programming language interface)를 통해 연결되었다. 여기서 PLI는 Verilog HDL과 C로 코딩된 프로그램을 연결해주는 인터페이스이다[9].

일반적으로 마이크로프로세서는 지원하는 각 명령어에 대한 정확한 동작을 확인함으로써 검증을 할 수 있다. 그러나 제안된 PCI 익스프레스 컨트롤러는 외부 인터페이스 프로토콜에 의해 동작되므로, 이를 위한 다른 검증 방법이 요구된다. 따라서 본 논문에서는 제안된 PCI 익스프레스 컨트롤러 HDL 모델과 행위 모델로 구성된 테스트 벤치 전체를 하나의 가상 마이크로프로세서로 보고, 이 가상 마이크로프로세서에 사용될 어셈블리 수준의 명령어들을 제안한다. 크게 5가지의 형태로 구분되는 이 명령어들은 PCI 익스프레스의 모든 프로토콜을 지원하며, 제안된 PCI 익스프레스 컨트롤러를 검증하기 위한 모든 시나리오를 제공하도록 정의되었다.

- 테스트 벤치의 호스트 브리지와 로컬 마스터에 데이터 전송을 지시하는 명령어.
- 메인 및 로컬 메모리에 접근하기 위한 명령어.
- PCI 익스프레스 인터페이스, 로컬 마스터 인터페이스, 그리고 로컬 슬레이브 인터페이스에서 데이터 전송 시 발생할 수 있는 모든 시나리오들을 설정하기 위한 명령어.
- 호스트 브리지와 로컬 마스터가 제안된 PCI 익스프레스 컨트롤러에 데이터 전송을 요청하기 위한 패킷을 생성 시, 필요한 파라미터를 사용자가 직접 설정하기 위한 명령어.
- 제안된 검증 방법을 효과적으로 관리함으로써, PCI 익스프레스 컨트롤러 통해 전송되는 패킷의 이동 흐름을 사용자가 알 수 있도록 하기 위한 명령어.

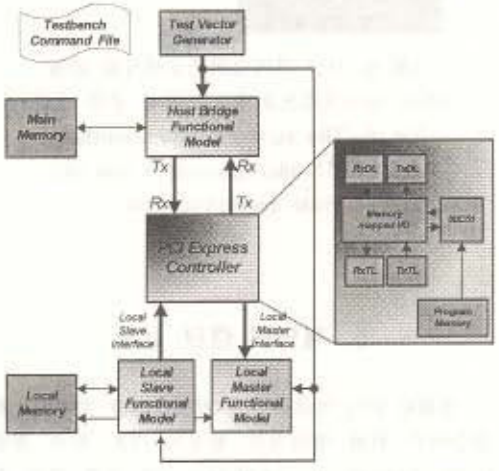

그림 5. 제안된 PCI 익스프레스 컨트롤러 검증 환경

Fig. 5. The verification environment of the proposed PCI Express controller

본 논문에서 제안된 통합 기능 검증 방법은 그림 6과 같이 하나의 가상 마이크로프로세서와 실제 8051 마이크로프로세서가 연동되어 동작하는 구조이다. 사용자는 PCI 익스프레스 프로토콜을 지원하기 위해 필요한 프로그램을 C 혹은 어셈블리로 작성 후 컴파일 하여 프로그램 메모리에 포팅을 한다. 또한 사용자는 제안된 PCI 익스프레스 컨트롤러 검증을 위해, 제안된 어셈블리 명령어를 이용하여 원하는 검증 시나리오 대로 프로그래밍하여 명령어 파일로 만든다. 이 파일은 테스터 벡터 생성기에 의해 코드로 바뀌고, 이 코드는 가상 마이크로프로세서에 의해 읽혀지고 실행된다.

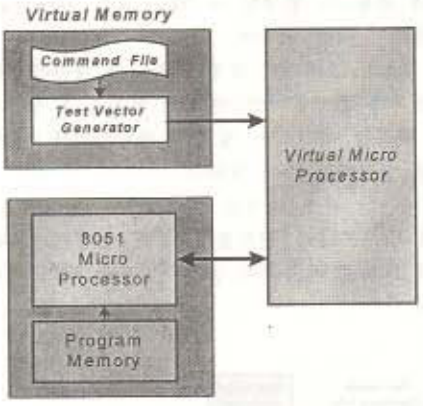

그림 6. 가상 마이크로프로세서와 실제 8051 마이크로프로세서의 통합 검증 방법
Fig. 6. The co-verification method using 8051 microprocessor and the virtual microprocessor

5. 제안된 랜덤 검증

설계된 PCI 익스프레스 컨트롤러를 효과적으로 검증하기 위한 일반적인 방법은 기본 동작 검증(basic-behavioral verification)과 설계 기반 검증(hardware-design verification)이다. 기본 동작 검증은 제안된 PCI 익스프레스 컨트롤러가 PCI 익스프레스 명령어와 프로토콜을 올바르게 지원하는지를 검증하는 단계이고, 설계 기반 검증은 설계되어진 하드웨어 구조들을 하나씩 검증하는 단계이다. 이를 위해 사용자는 결과를 이미 예측한 상태로 명령어 파일을 만들어 검증을 한다. 하지만 사용자가 예상치 못한 코너 케이스에서 시뮬레이션이 되지 않는다면 검증이 충분히 이루어 졌다고 할 수 없다. 이를 위해 본 논문에서는 벡터 생성 부분, 시뮬레이션 부분, 그리고 비교 엔진으로 구성된 랜덤 검증 환경을 그림 7과 같이 제안한다.

랜덤 벡터 생성 부분에서는 랜덤 벡터 파라미터를 설정하면 랜덤 벡터 생성기에 의해 자동으로 테스트 명령어 파일로 바뀌게 된다. 시뮬레이션 부분은 테스트 벤치 부분과 제안된 PCI 익스프레스 컨트롤러 참조 모델 부분으로 나누어진다. 설계 모델 부분은 제안된 PCI 익스프레스 컨트롤러 HDL 모델, 8051 및 메모리로 구성된 그림 5의 구조를 나타낸다. 참조 모델은 제안된 PCI 익스프레스 컨트롤러의 기능 및 8051의 기능 모두를 하나의 블록으로 구현한 명령어 수준에서만 돌아가는 C 언어로 구현된 행위 모델이다. 설계 모델과 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 설계 모델의 메인 메모리와 로컬 메모리, 참조 모델의 메인 메모리와 로컬 메모리를 각각 비교하여, 오류가 발생 했을 때 그 결과를 로그 파일로 남긴다.

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[7]. 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 모집 합에 대하여 균일하게 분포하지 않고 특정 유형과 강한 상관관계를 지니는 경우가 대부분이다[10-11]. 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 추가된 명령어가 표 1에 나와 있다.

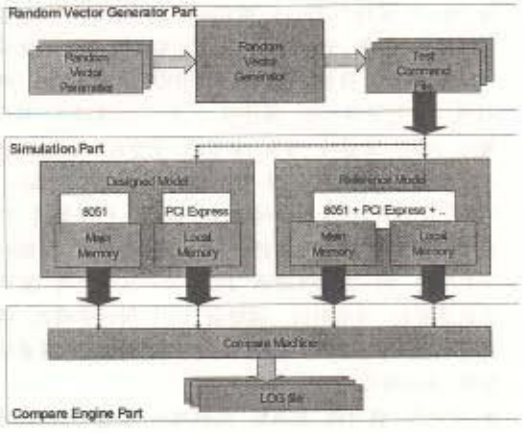

그림 7. 랜덤 테스트를 위한 검증 환경
Fig. 7. The random test verification

표 1. 랜덤 테스트 검증을 위한 랜덤 벡터 파라메터

Table. 1. The random vector parameter

명령어	설명
HB_CmdNum	호스트 브리지에 의해 PCI 익스프레스 컨트롤러에 요청되는 명령어의 수를 설정하기 위한 파라메터.
HB_DataCntMax	호스트 브리지에 의해 PCI 익스프레스 컨트롤러에 전송 요청 가능한 데이터의 최대 수를 설정하기 위한 파라메터.
HB_DataCntMin	호스트 브리지에 의해 PCI 익스프레스 컨트롤러에 전송 요청 가능한 데이터의 최소수를 설정하기 위한 파라메터.
HB_MWriteProb	호스트 브리지가 PCI 익스프레스 컨트롤러에 요청하는 메모리 쓰기 요청과 메모리 읽기 요청 중 메모리 쓰기 명령어가 차지하는 비율을 나타내는 수를 설정하기 위한 파라메터.
LM_CmdNum	로컬 마스터에 의해 PCI 익스프레스 컨트롤러에 요청되는 명령어의 수를 설정하기 위한 파라메터.
LM_DataCntMax	로컬 마스터에 의해 PCI 익스프레스 컨트롤러에 전송 요청 가능한 데이터의 최대수를 설정하기 위한 파라메터.
LM_DataCntMin	로컬 마스터에 의해 PCI 익스프레스 컨트롤러에 전송 요청 가능한 데이터의 최소수를 설정하기 위한 파라메터.
LM_MWriteProb	로컬 마스터가 PCI 익스프레스 컨트롤러에 요청하는 메모리 쓰기 요청과 메모리 읽기 요청 중 메모리 쓰기 명령어가 차지하는 비율을 나타내는 수를 설정하기 위한 파라메터.

표 1의 명령어를 이용하면 사용자는 각 인터페이스의 프로토콜 상에서 발생하는 다양한 시나리오의 복잡성을 미리 결정할 수 있다. 즉 이렇게 결정된 값은 로컬 마스터, 로컬 슬레이브, 그리고 호스트 브리지에 의해 사용되어져 다양한 프로토콜을 발생하게 된다. 이들 명령어에 의해 세팅된 정보를 바탕으로 랜덤 발생기는 명령어 파일을 생성하게 된다. 생성된 랜덤 테스트 벡터 파일은 설계 모델에 입력된 후, 명령어 수행에 따라 메인 메모리와 로컬 메모리간의 데이터 이동을 수행한다. 이 때 똑같은 테스트 벡터 명령어는 참조 모델에도 인가되어 독립적으로 시뮬레이션을 수행한다. 비교 엔진은 검증 목적인 설계 모델 내의 PCI 익스프레스 컨트롤러의 시뮬레이션 결과와 참조모델의 시뮬레이션 결과를 비교하게 된다. 로그 파일에 표시된 오류 위치와 그 오류 위치에 해당하는 명령어, 그리고 시뮬레이션 결과 파형을 이용하여 오류가 발생한 부분을 쉽게 찾을 수 있다.

6. 결과 및 고찰

PCI 익스프레스 컨트롤러는 먼저 기본 동작 검증과 설계 기반 검증을 실행하였다. 검증 결과 표 2와 같이 102개의 오류가 발생하였고, 외부 인터페이스의 프로토콜을 관리하고 PCI 익스프레스 컨트롤러의 데이터 패스(data path)를 제어하기 위한 상태도(state machine)에서 가장 많은 오류가 발생함을 확인하였다.

표 2. 기본 동작 검증과 설계 기반 검증 결과

Table. 2. The results of basic 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
상태도에서 상태 추가	8
상태도에서 컨트롤 신호 수정 및 삽입	31
합계	102

표 3. 랜덤 테스트 검증 결과

Table. 3. The results of random test verification

에러의 유형	개수
PCI 익스프레스의 데이터 연결 계층의 프로토콜 어김	4
전송 계층의 로컬 마스터 인터페이스 상에서 프로토콜 어김	1
전송 계층의 로컬 슬레이브 인터페이스 상에서 프로토콜 어김	2
컨피규레이션 레지스터 구조 수정	2
마이크로프로세서를 위한 상태 레지스터 수정	3
기타	4
합계	16

랜덤 테스트 검증을 위해 10,000개의 데이터 전송 명령어를 수행하여 랜덤 환경에서 시뮬레이션 한 결과를 표 3에 나타내었다. 즉, PCI 익스프레스 컨트롤러 HDL 모델은 이미 기본 동작 검증과 설계 기반 검증을 수행하였음에도 불구하고, 랜덤 테스트 결과 16개의 에러가 발견되었다. 이들 경우는 설계자가 충분히 PCI 익스프레스의 스펙대로 구현하였음에도 미처 간파하지 못한 부분에서 발생하는 설계상의 오류들이다.

7. 결 론

본 논문에서는 PCI 익스프레스의 전송계층과 데이터 연결계층의 모든 기능을 지원하는 PCI 익스프레스 엔드포인트 컨트롤러인 PCI 익스프레스 컨트롤러를 설계하였다. 설계된 PCI 익스프레스 컨트롤러의 각 블록을 효과적으로 제어하기 위해 8051 마이크로프로세서를 이용하여 PCI 익스프레스의 프로토콜을 제공하는 프로그램을 C로 코딩하였다. 또한 마이크로프로세서와 설계된 PCI 익스프레스 컨트롤러의 인터페이스를 위해 메모리 맵 I/O를 사용하였다.

또한 본 논문에서는 PCI 익스프레스 컨트롤러와 8051 마이크로프로세서의 통합 검증을 위한 방법을 제안한다. 먼저, 벡터 생성 부분, 테스트 벤치, 그리고 메모리로 구성된 테스트 벤치를 하나의 가상 마이크로프로세서로 구성하였다. 그리고 PCI 익스프레스의 모든 프로토콜을 지원할 수 있는 어셈블리 수준의 명령어들을 테스트 벤치에 적용되도록 제안하였다. 즉, 제안된 검증 환경은 가상 마이크로프로세서와 실제 8051 마이크로프로세서가 연동되어 동작하는 구조이다. 특히 일반적인 기본 동작 검증과 설계 기반 검증에서 찾지 못한 특수 경우의 에러를 찾기 위한 검증을 위해 랜덤 검증 환경 및 테스트 파라미터를 정의 하였다.

제안된 검증 환경과 명령어를 통해 설계된 PCI 컨트롤러의 검증 결과 랜덤 테스트 검증을 통해 16개의 오류를 찾을 수 있었으며, 전체 오류 중 14%를 랜덤 테스트 검증을 통해 찾을 수 있었다.

참고문헌

[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 Andreson, and Tom Shanley, PCI Express System Architecture, Mind Share, 2003.

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

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

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

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

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

[11] 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

영남대학교 전자공학과 학사. 영남대학교 전자공학과 석사. 영남대학교 전자공학과 박사. 현재, DGIST(대구경북과학기술연구원) 선임 연구원

관심분야: PCI 컨트롤러, 모바일 SoC 설계, 디지털 시스템,

Email: braham@dgist.ac.kr

김 길 동

한양대학교 전자공학과 학사. KAIST 전자공학 석사. KAIST 전자공학 박사. 현재, 영남대학교 전자정보공학부 부교수

관심분야: 집적회로 및 CAD, 모바일 SoC 설계, 임베디드 시스템

Email: kssung@yu.ac.kr