실제 트래픽 기반의 기가비트 스위치 칩의 검증

전 정 범(全正範), 장 유 성(張有成), 이 주 환(李周桓), 강 무 경(姜武炅),
이 승 왕(李承旺), 경 종 민(慶宗旻)
한국과학기술원 전기 및 전자공학과 VLSI System연구실
전화: (042) 869-8023/ 팩스: 869-4040

Real-Traffic Based Verification for Gigabit-Switch Chips

Jung-Bum Chun, You-Sung Chang, Ju-Hwan Yi, Moo-Kyung Kang, Seung-Wang Lee,
Chong-Min Kyung
e-mail: jbchun@duo.kaist.ac.kr

Abstract

As the Internet traffic increases, the demand for higher performance routers continues to grow, and it makes switch chips more complex. To make matters worse, these chips also need to handle high-level services. In this paper, we introduce an efficient verification methodology that can support real network traffics to satisfy the verification requirement of real complex situation even at the early design phase of switch chips.

서론

인터넷 서비스 및 프로토콜들이 다양해짐에 따라 라우터와 라우팅 스위치의 구조가 더욱 복잡해지고 그 디자인 또한 매우 복잡해졌다. 이로 인해 시스템의 검증에 있어서도 실제 상황에서 일어날 수 있는 복잡한 경우들을 테스트해볼 수 있는 완전한 테스트 환경이 요구되고 있다.

일반적으로 다계층 스위치와 같은 네트웍 장비의 시스템 수준에서의 동작은 일반적인 칩간 기능 검증으로 보장해 주기가 쉽지 않다. 두 독립된 호스트간의 복잡 다양한 소프트웨어 루틴들에 의해서 입력 패킷들이 생성되기 때문에 검증할 스위치 칩셋의 검증환경을 모델링하기 또한 어렵다. 더욱이 요구된 서비스 테스트를 위해 적절한 입력 패킷들을 생성하여 인가하였다할지라도 그 출력 패킷 스트림이 올바른 것인지를 보장할 수는 없다.

시스템 수준의 검증으로는 잘 알려진 하드웨어 에뮬레이션이 있다.[1],[2],[3],[4] 그러나 초기 디자인 시점에서 FPGA 게이트로 매핑될 수 있는 거의 완전한 디자인을 얻어낸다는 것이 불가능하기 때문에 하드웨어 에뮬레이션은 초기 시점에서 적용하기 어렵다. 하드웨어 에뮬레이션이 설사 가능하다 하더라도 디자인의 수정과 검증의 반복에 있어서 긴 반복시간을 요구한다.

이 논문에서 우리는 포트 콘트롤러와 스위치 패브릭으로 이루어져있는 기가비트 이더넷 스위치 칩들의 개발에 사용된 효율적인 디자인 검증 방법론을 소개하려 한다. 이 방식은 실제 네트워크 환경을 테스트 환경으로 사용하는 것으로, 미리 트래픽을 덤프 받아서 사용하는 이전의 방식[5]과는 다르다. 우리의 접근 방식은 테스트 환경을 호스트간의 상호 작용이 일어나는 실제 네트웍 환경으로 이끌어 내는 것이다.

본문I에서 우리는 검증과정에 있는 칩에 대한 간략한 소개와 우리 검증방식에 대한 개괄적인 설명을 하려고 한다. 그리고 나서 본 접근방식의 검증 환경을 본문II와 본문III에서 설명하겠다.

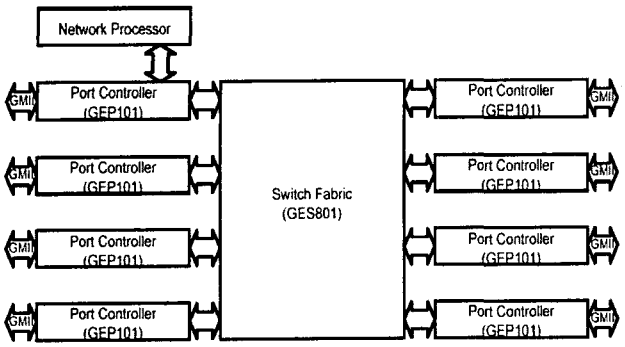

그림 1 기가비트 이더넷 칩 GES801과 GEP101을 이용한 8X8 IP 스위치의 구성

I 스위치 칩의 개요와 검증전략

검증하고자하는 대상인 기가비트 이더넷 스위치 시스템은 그림1과 같이 포트 콘트롤러 GEP101와 스위치 패브릭 GES801, 두 개의 칩을 기반으로 구성된다. 기본적으로 포트 콘트롤러는 입력되는 패킷의 헤더를 분석하고 스위치 시스템에 있는 포트들 중에 도착지를 결정해서 패킷을 전달하는 역할을 한다. 이러한 기본적인 기능과 더불어 RIP, IGMP, RSVP등과 같은 다양한 프로토콜을 포함하고 있는 제어패킷이 들어왔을 경우 처리 결과에 따라서 상위 수준의 서비스를 제공하기 위해 내부 상태와 구조를 변화시킨다.

그림2와 같이 칩의 검증은 블록 수준 검증, 칩 수준 검증을 거쳐 마지막엔 시스템 수준 검증을 거치게 된다.

각각의 블록의 검증에서 우리는 CoverMeter[1]에 의해 얻어진 테스트 벡터 커버리지 정도를 기반으로 테스트 벡터 시뮬레이션을 행하였다. 모든 기술과 조건, 상태 전이는 기본적으로 이 테스트 벡터 시뮬레이션을 통해 체크하게 된다. 이와 동시에 포트 콘트롤러내의 임베디드 프로세서에서 수행될 펌웨어는 이 프로세서의 동작 모델을 이용하여 독립적으로 테스트하였다.

각각의 블록들이 점진적으로 칩으로 통합되어지는 동안 단순하거나 특별한 경우에 관한 테스트는 위와 같이 인위적인 테스트 벡터들로 이루어지게 된다. 칩의 디자인이 완성되어 통합되면 가상 네트워크 환경에서 새로운 종류의 검증 과정이 시작된다. 가상 네트워크에서는 덤프된 실제 트래픽이나 적절히 처리된 데이터 스트림이 입력으로 인가된다.

시스템 수준 검증을 위한 실제 네트웍 환경에서의 시

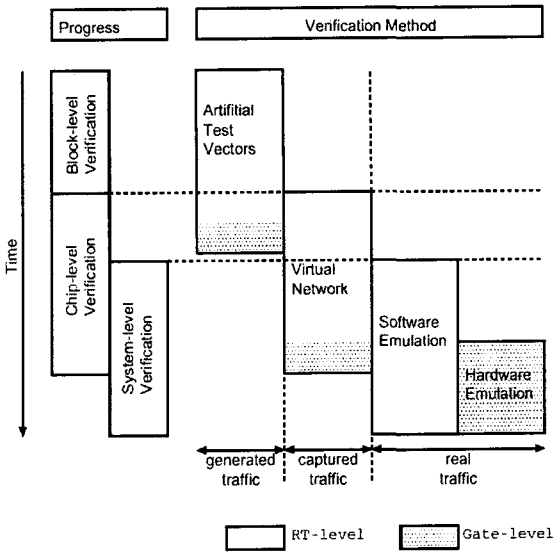

그림 2 기가비트 이더넷 칩의 검증 방식

뮬레이션은 칩 수준 검증의 초기 시점에서 시작된다. 이것이 가능한 것은 이 시점의 시뮬레이션이 소프트웨어 수준에서 이루어지고, 이 때문에 RT(register transfer)-수준 디자인이 게이트들의 합성이 없이 그대로 사용될 수 있기 때문이다. 우리의 검증 전략에서 실제 트래픽 기반의 시뮬레이션은 시스템 기능 검증에 있어서 주요한 역할을 수행하게 된다.

마지막 단계에서 PHY/MAC과 스위치 칩간의 인터페이스와 기능 검증을 위한 방법으로 하드웨어 에뮬레이션을 행하게 된다. 이것은 하드웨어 에뮬레이션의 역할이 일반적인 경우보다 상당부분 축소된 것으로 시스템의 기능 검증부분이 이미 바로 전 단계인 실제 트래픽 기반의 시뮬레이션 단계에서 행해졌기 때문이다.

II 칩 수준 검증을 위한 가상 네트워크 환경

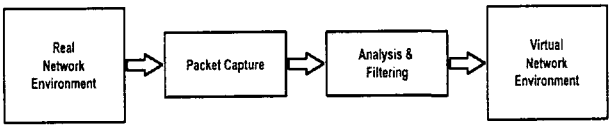

그림 3 입력 데이터 패킷 스트림의 구성

칩 수준 검증은 디버깅을 쉽게 하고 패킷 스트림을

패킷의 경로를 추적할 수도 있다. 실행 중에 오류가 발생하는 경우에는 그 패킷과 현재의 상태들이 디버깅을 위해 저장된다.

하드웨어를 사용하지 않는 이 시뮬레이션 단계의 약점은 MAC, 네트워크 프로세서, 메모리간의 인터페이스 검증에 있다. 마지막으로는 시스템 검증과 칩간 인터페이스 검증을 위해서 FPGA 구현을 통한 에뮬레이션을 행한다. 그림6이 마지막 검증과정을 위한 테스트 환경의 블록도를 보이고 있다. 이 시스템은 디자인된 FPGA들과 MAC 인터페이스 칩, 네트워크 프로세서, SDRAM, SSRAM등으로 구성되어 있다.

결론

이 논문에서 우리는 스위치 칩의 기능 검증을 위한 방법론을 제시하였다. 이는 쉬운 디버깅이라는 장점은 유지하면서 디자인의 초기 단계부터 실제 패킷 트래픽으로 시스템 수준의 검증을 할 수 있다는데 그 의미가 있다. 실제 트래픽을 이용한 검증은 특화된 서비스를 지원하는 다계층 스위치의 복잡한 기능들을 검증하는데 사용되어질 수 있다.

덤프된 패킷을 사용하여 입력 패킷들은 발생시킨 후 입력과 출력을 비교해 보는 가상 네트워크 환경에서 출발하여 결국 PC 환경을 통해서 RT-수준 디자인을 실제 네트워크 환경에 적용시켜 호스트와의 상호 작용을 하는 단계에 이르게 된다.

우리는 현재 전체 검증 계획의 중간 단계에 와 있으며 칩간 인터페이스 검증을 위한 하드웨어 에뮬레이션을 준비중이다.

참고도서

[1] S.Mehta, S. Al-Ashari, Dennis Chen, S.Cokmez, P. Desai, R. Eltejaein, P.Fu, J. Gee, T. Granvold, A. lyer, K. Lin, G. Maturana, D. McConn, H. Mohammed, J. Mostoufi, A. Moudgal, S. Nori, N. Parveen, G. Peterson, M. Splain and T. Yu, *"Verification of the UltraSPARC™ Microprocessor"* In COMPCON, pp.452-461 1995.

[2] Gopi Ganapathy, Ram Narayan, Glenn Jorden, and Denzil Fernandez. *"Hardware Emulation for Functional Verification of K5"* In Proceedings of the 33rd Design Automation Conference, pp.315-318 1996.

[3] Val Popescu and Bill McNamara, *"Innovative Verification Strategy Reduces Design Cycle Time For High-End SPARC Processor"* , In Proceedings of the 36th Design Automation Conference, pp.311-314 1999.

[4] You-Sung Chang, Seung-Jong Lee, In-Cheol Park and Chong-Min Kyung. *"Verification of a Microprocessor Using Real World Applications"* In Proceedings of the 36th Design Automation Conference, pp. 181-184 1999.

[5] Danial Geist, Giora Biran, Tamara Arons, Michael, Slavkin, Yvgeny Nustov, Monica Farkas, Karen Holtz, Andy Long, Dave King, and Steve Barret, *"A Methodology For the Verification of a "System on Chip"* In Proceedings of the 36th Design Automation Conference, pp.574-579 1999.

1) 주어진 테스트벡터의 코드 커버리지를 측정하는 보조 툴

제공하기 위해서 개발된 가상 네트워크 환경을 기반으로 수행되는데 디자인의 초기단계에서 버그를 일으킬 만한 패킷 스트림을 임의로 생성시킬 수 있다는 장점을 가진다.

그림3은 가상 네트워크 환경에서 입력 데이터를 구성하는 과정을 보이고 있다. 실제 네트워크에서 패킷들이 포획되면 사용자 명령에서 지시한대로 특정 포트별로 패킷들을 분류한다. 특정한 프로토콜을 담고있는 패킷들은 분석과 필터링 과정을 통해 선택된다. 가상 네트워크 환경은 포획된 패킷 스트림을 사용해서 실제 네트워크 환경을 흉내내는 역할을 한다.

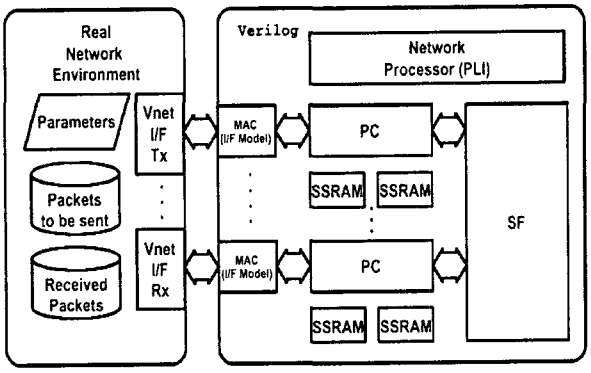

그림 4 가상 네트워크를 통한 기능 검증

그림4는 가상 네트워크 환경을 이용한 시뮬레이션 모델을 나타낸다. 가상 네트워크의 인터페이스 포트는 verilog PLI 루틴을 통해서 디자인된 칩의 verilog 모델과 연결된다. verilog 모델은 MAC 인터페이스 모델(LU5M31), 포트 콘트롤러, 스위치 패브릭, 네트워크 프로세서로 이루어져 있다. 여기서 네트워크 프로세서는 전반적인 시스템을 초기화하고 프로토콜들을 처리하는 기능을 한다.

III 시스템 수준 검증을 위한 실제 트래픽 기반의 시뮬레이션과 하드웨어 에뮬레이션

위의 시뮬레이션 단계를 거치고 나면 디자인은 최소한 가상 네트워크 환경에서는 완전한 모델로 보일 수 있지만 실제 네트워크 환경에서는 그렇지 못하다. 실제 환경에서 일어날 수 있는 여러 가지 복잡한 경우들에 대한 보장을 하지 못할 뿐 아니라 다른 호스트와의

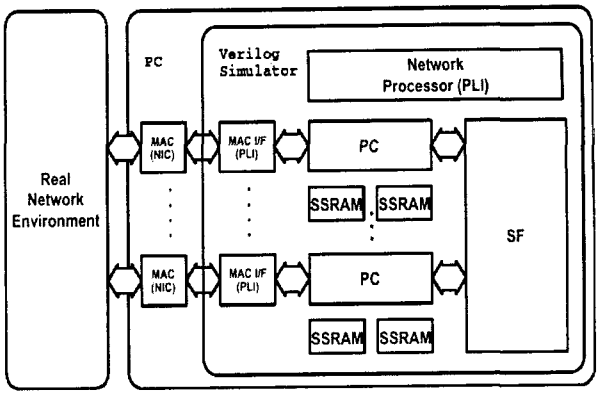

그림 5 실제 트래픽 기반의 기능 검증

프로토콜 상호 작용 등에 대한 검증은 이루어지지 않았기 때문이다. 그리하여 실제 네트워크 환경에서의 검증이 필요하게 된다. 일반적으로는 하드웨어 에뮬레이션이 적용되지만 이는 디버깅하기 어렵고 신호들을 일일이 확인하기 쉽지 않을 뿐 아니라 디버깅시간도 오래 걸린다. 그래서 일반적인 시뮬레이션과 에뮬레이션간의 새로운 단계가 필요하게 되었다.

그림5는 PC를 이용한 실제 트래픽 기반의 시뮬레이션 환경을 보이고 있다. 이 구성에서 실제 네트워크 환경은 가상 네트워크 환경을 대신하게 되었다. 이 경우에 MAC 인터페이스는 PC 시스템에서 전송/수신을 담당할 다수의 NIC를 직접 접근하는 PLI 루틴으로 작성되어 있다.

Verilog 시뮬레이터에 의해 디자인된 부분을 실행하면서 전체적인 시스템은 실제 네트웍 환경에서 동작한다. 이러한 구성의 특징은 실제 네트워크 트래픽 입력과 Verilog 시뮬레이터 환경에서의 편리한 디버깅에 있겠다. 모든 신호들을 동시에 확인할 수 있고 모든

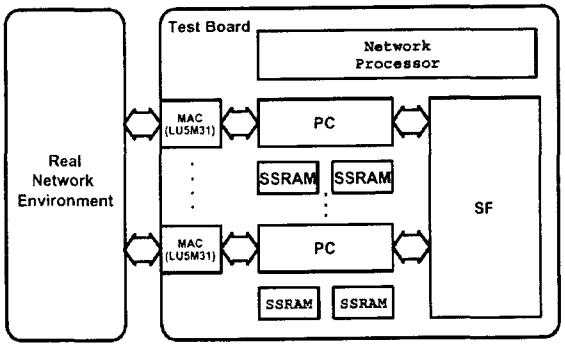

그림 6 하드웨어 에뮬레이션에 의한 기능 검증