논문 2012-49SD-6-2

NetFPGA를 이용한 고성능 오버레이 멀티캐스트 패킷 전송 엔진 구현

(Implementation of High Performance Overlay Multicast Packet Forwarding Engine On NetFPGA)

전 혁 진*, 이 현 석*, 정 용 진**

(Hyuk-Jin Jeon, Hyunseok Lee, and Yong-Jin Jeong)

요 약

인터넷상에서 고품질 멀티미디어 서비스는 화상회의나 실시간 인터넷 방송 등 여러 분야에 적용 될 수 있기 때문에 주목받고 있다. 이러한 서비스에서 네트워크 자원을 효율적으로 사용하기 위해 IP 멀티캐스트가 해결책으로 제시되고 있지만 관리상의 문제점으로 인해 실제로 사용되지 못하고 있다. 대안으로 기존의 라우터들의 하드웨어를 변경하지 않고 상위계층에서 라우팅을 하는 오버레이 멀티캐스트가 제시되고 있다. 하지만 오버레이 멀티캐스트는 상위계층에서 멀티캐스팅 동작을 수행하기 때문에 최대 전송속도가 낮아서 고속 멀티미디어 데이터 전송에 부적합하다. 본 논문에서는 NetFPGA를 이용하여 고속의 처리가 필요한 부분인 멀티캐스팅 동작을 위한 패킷의 복제와 전송, 터널링 기능을 설계 하였다. 그 외에 비교적 고속의 처리가 필요하지 않은 부분은 소프트웨어로 구현하였다. 이로 인하여 실시간 처리가 가능하도록 하였다. 향후 성능 개선을 통하여 복제 가능한 지점의 수를 늘리고, 최적화를 통해 처리속도를 증가 시킬 연구를 진행할 것이다.

Abstract

High-quality multimedia on the Internet has attracted attention because of its wide application area. IP multicast has been proposed as a solution to use efficient network resources in these services. However, IP multicast has not been commonly used due to a number of practical issues such as security and management. As an alternative, an overlay multicast routing which is performed in upper protocol layers on legacy networks without changing hardware has been presented. Yet, the maximum data transmission capacity of the overlay multicast is not sufficient for real time transmission of multimedia data. In this paper, we have implemented an overlay multicast engine on NetFPGA which allows us to perform packet replication and tunneling which need high-speed. In addition, we have implemented extra portions which need low-speed in software. From now on, we will progress research which increase the number of terminal spots which can be replicated by improvement and amplify throughputs by optimization.

Keywords : Overlay Network, Multimedia, Multicast, NetFPGA

Ⅰ. 서 론

인터넷상에서 고품질 멀티미디어는 현실감 있는 화상회의, 실시간 인터넷 방송, 고품질의 VOD(Video On Demand) 서비스 등 여러 분야에 적용될 수 있기 때문에 최근에 많은 주목을 받고 있다[1]. 그중 화상회의나

* 정회원, ** 정회원-교신저자,
광운대학교 전자통신공학과
(Kwangwoon University)
※ 본 논문은 광운대학교 2011년 교내학술연구 및 한국연구재단 (NRF-2010-0014557)의 지원을 받아 이루어졌습니다.
접수일자: 2011년11월23일, 수정완료일:2012년5월23일

실시간 인터넷 방송의 경우 서비스 제공자와 단일 소비자 간의 일대일 전송 보다는 다수의 소비자에게 단일 데이터를 동시에 제공하는 일대다 전송을 하게 되는 경우가 많다. 이러한 서비스들은 데이터를 실시간으로 끊김 없이 전송하는 것이 매우 중요하다. 수많은 소비자들이 서버에 접속을 하게 된다면 트래픽이 과도하게 늘어나고 대역폭의 한계로 인해 지연 시간이 증가되기 때문에 기존의 일대일 통신인 유니캐스트(Unicast)방식으로는 높은 수준의 서비스 제공이 어렵다. 그래서 멀티미디어 데이터를 일대다 전송할 때 IP 멀티캐스트(Multicast)가 해결책으로 제시되고 있다[2].

네트워크 계층에서 멀티캐스트 라우터에 의해 수행되는 IP 멀티캐스트는 패킷의 복제와 전송을 라우터가 담당하게 된다. 하지만 현재 상용망에서 IP 멀티캐스트는 라우터의 구현 및 확장, 혼잡 제어, 신뢰성 있는 전송, 주소 할당 문제 등의 여러 가지 이유로 실제로 사용되지 못하고 있는 실정이다[3]. 그래서 IP 멀티캐스트의 대안으로 기존의 라우터들의 하드웨어를 변경하지 않고 상위계층에서 라우팅을 하는 오버레이(Overlay) 멀티캐스트가 제시되고 있다[4].

오버레이 멀티캐스트는 앞서 언급한 것과 같은 장점을 가지고 있으나 해결해야 할 몇몇 중요한 문제가 존재한다. 그중 가장 큰 문제는 IP 멀티캐스트보다 전송 속도가 떨어진다는 점이다. 이 방식에서는 패킷의 복제와 전송을 소프트웨어가 처리하기 때문이다. 이는 실시간 멀티미디어 전송에 있어서 심각한 문제가 된다. 양방향 대화형 통신의 경우 원활한 대화가 가능하기 위해서는 100ms 이하의 지연시간이 요구되며, 스트리밍 서비스에서는 최소 수 초 이내의 지연시간이 요구된다. 전송 속도가 떨어지면서 나타나는 또 다른 문제는 지터(Jitter)의 발생이다. 동영상이나 음성 같은 시간 제약을 가지는 미디어 서비스에서 끊김없는 재생을 위해서는 지터를 최소화하는 것 또한 매우 중요하다[5].

본 논문에서는 오버레이 멀티캐스트의 주요 기능을 하드웨어 상에서 구현하여 IP 멀티캐스트의 단점을 보완한 시스템을 제안한다. 이 시스템은 멀티캐스팅 동작에 필요한 패킷의 복제와 전송, 터널링(Tunneling) 동작을 FPGA를 내장한 개발용 플랫폼인 NetFPGA 상에서 수행하도록 하여 고속으로 데이터 처리가 가능하며 더나가 일대다 멀티미디어 전송에서 실시간성과 고화질을 동시에 보장 할 수 있도록 하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 관련 연구를 기술하고 Ⅲ장에서는 제안된 오버레이 멀티캐스트 시스템의 개요와 구조를 설명하며 하드웨어로 구현 하려는 이유를 기술 하였다. Ⅳ장에서는 구현한 하드웨어에 대하여 기술하고 Ⅴ장에서는 설계한 내용의 검증 결과와 분석을 기술하며 Ⅵ장에서 결론 및 향후 연구방향에 대하여 논한다.

Ⅱ. 관련 연구

본 논문에서 구현한 엔진은 오버레이 네트워크를 구현하는데 필요한 터널링, 패킷 복제, 고속 데이터 포워딩 기능을 모두 포함하고 있으며 하드웨어에 위치하는 포워딩 정보를 별도의 제어기(Controller)에서 입력하도록 설계하였다. 이와 관련된 연구로는 오픈플로우(Openflow) 프로토콜[6], IPSec(Internet Protocol Security Architecture)를 위한 하드웨어 가속기[7], 고속으로 라우팅을 하기위해 외부 메모리에 존재하는 대용량 룩업테이블 검색엔진[8] 등이 있다.

오픈플로우 프로토콜은 <그림 1>과 같이 제어기로 하여금 망을 통해 이더넷 스위치 또는 IP 라우터의 데이터패스(Datapath)에 대한 제어가 가능하도록 하는 통신 프로토콜로서 Software Define Network 구현의 예라고 볼 수 있다.

IPSec를 위한 하드웨어 가속기는 보안 프로토콜에서 기존의 IP 패킷에 캡슐화 하여 전송하는 과정에서 필요한 ESP (Encapsulating Security Payload)와 AH (Authentication Header) 알고리즘을 하드웨어 가속기 형태로 구현한 것이다.

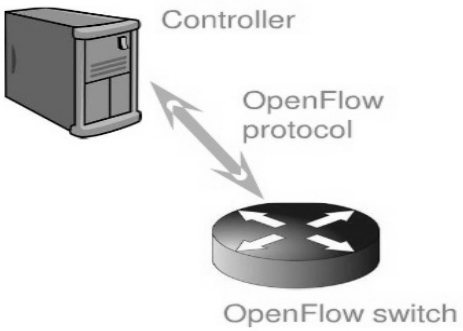


그림 1. 오픈플로우 프로토콜
Fig. 1. OpenFlow protocol.

대용량 룩업테이블 검색 엔진은 고속으로 IP 패킷을 라우팅을 하기위해 필요한 메모리를 반도체 내부와 외부에 적절히 배치하여 구현성과 동작 속도를 동시에 만족 시키고 있다.

이외에도 상위계층에서 IP 멀티캐스트를 수행하는 오버레이 멀티캐스트 연구도 활발히 진행 중이다[9~10].

Ⅲ. 본 론

1. 배경이론

가. 오버레이 네트워크(Overlay Network)

오버레이 네트워크란 <그림 2>와 같이 기존의 물리적인 네트워크 위에 논리적인 노드들(Nodes)과 링크들(links)을 정의하여 이루어진 가상의 네트워크이다. 오버레이 네트워크에서는 기존 라우터에서 수행되는 라우팅 기능이 상위계층에서 처리된다. 이는 새로운 망 규격을 기존 망에 적용할 때 별도로 망 장비를 변경할 필요 없다는 장점을 가지고 있다. 그러나 상위 계층에서 라우팅이 이루어지므로 데이터 전송속도가 느려지는 단점을 가지고 있다.

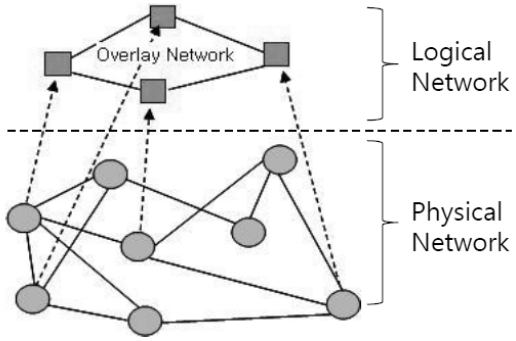


그림 2. 오버레이 네트워크의 구조
Fig. 2. Structure of overlay network.

나. 오버레이 멀티캐스트(Overlay Multicast)

오버레이 멀티캐스트는 오버레이 네트워크를 구성하여 기존의 네트워크 계층에서 담당하던 IP 멀티캐스트 기능을 네트워크 계층보다 상위계층에서 담당하도록 구현한 것이다. 오버레이 멀티캐스트는 기존의 IP 멀티캐스트와 기능적으로 동일하다. 하지만 구현방법을 달리하여 하드웨어 대신 상위계층인 소프트웨어에서 멀티캐스트 관련 기능을 담당한다.

IP 멀티캐스트가 기존 라우터들을 멀티캐스트 라우터로 변경해야 가능한 반면 오버레이 멀티캐스트는 기존 라우터들을 교체할 필요가 없는 장점이 있다. 반면 상위계층에서 데이터 복제기능을 담당하기 때문에 고속으로 많은 양의 데이터를 실시간으로 처리하지 못하는 문제점을 가지고 있다.

다. NetFPGA 플랫폼

<그림 3>은 NetFPGA 플랫폼으로서 개발자가 네트워크 프로토콜을 소프트웨어와 하드웨어로 모두 설계할 수 있도록 만들어진 개발용 플랫폼이다. 이 플랫폼은 FPGA를 포함하고 있어 하드웨어의 기능도 개발자가 변경할 수 있다. NetFPGA는 PCI 버스를 이용하여 Linux OS가 설치된 PC에 연결하여 사용할 수 있으며 4개의 1기가비트(Gbps) 이더넷 포트가 있다. 또한 4.5Mbytes ZBT SRAM과 64Mbytes DDR2 RAM을 가지고 있다. 사용자 정의 기능을 하드웨어적으로 구현할 수 있도록 Xilinx사의 Virtex-2 Pro FPGA[11]가 장착되어 있다.

NetFPGA에서 저속처리가 허용되지만 동작구조가 복잡한 라우팅 프로토콜 처리와 라우팅 테이블관리는 소프트웨어가 담당한다. 반면에 고속처리가 필요하지만 동작구조가 간단한 데이터 스위칭은 하드웨어가 담당하여 하나의 라우터로서 동작하게 된다.

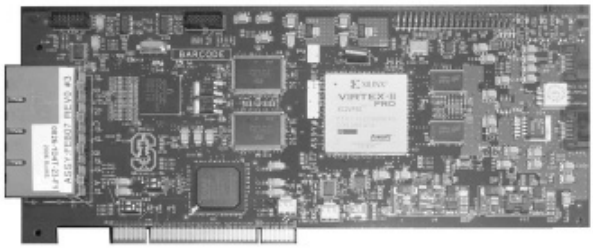


그림 3. NetFPGA platform
Fig. 3. NetFPGA platform.

라. 오픈플로우(OpenFlow) 프로토콜

오픈플로우는 2008년 미국의 스탠포드대학교에서 처음 제시된 통신 알고리즘이다. 아직 표준화가 이루어지지 않았으며 상용화도 되지 않은 발전단계의 알고리즘이다. 그러나 최근 들어 라우터/스위치 벤더인 시스코, 주니퍼, HP 네트워킹, 익스트림, 피카8, 노달 등이 테스트베드에 참여하고 있다. 그뿐만 아니라 MS, Facebook

등과 같은 클라우드 서비스 업체에서 많은 관심을 가지고 있으며 구글 에서는 얼마 전 공식석상에서 자사의 모든 내부 네트워크들을 위한 맞춤설계 하드웨어에서 오픈플로우를 사용하고 있다고 발표했다.

오픈플로우는 스위치와 라우터에 플로우기반 사용자 테이블을 생성하고 이들 간 통신에 필요한 표준화된 프로토콜을 제공하여 사용자가 원하는 형태의 프로그램을 작성 가능하도록 도와준다. 이를 활용하여 네트워크 관리자가 새로운 정책을 네트워크 인프라에 적용한다면 하나하나의 장비마다 설정을 적용하고 재검토 해야하거나, 만약 잘못된 부분이 있다면 장비 하나하나를 일일이 확인해야 하는 문제점을 해결해 준다. 이러한 단점은 필연적으로 네트워크 장비를 복잡하게 만들고 신기술 도입을 더디게 한다.

오픈플로우는 네트워크 영역에서 다뤄질 부분을 컨트롤러로 분리시키는데 주안점을 두고 있다. 컨트롤러가 장비를 관리하도록 한 것이다. 이렇게 하면 장비가 각각 개별적인 컨피그레이션을 할 필요가 없어진다. 운영이 단순해지고, 거기에 스위치를 추가적으로 다루면 되므로 유연한 네트워크를 만들 수 있게 해준다. 이런 구조적인 특성은 보안, 확장성 등 측면에서도 가용성이 뚜렷하다.

2. 연구 동기

오버레이 멀티캐스트는 상위계층에서 IP 멀티캐스트를 수행하기 때문에 지연시간과 전송지연 시간의 변화가 커지는 현상을 발생시킨다. NetFPGA로 구현된 라우터를 이용하여 소프트웨어로만 IP 멀티캐스트 기능을 처리할 경우 전송 지연의 발생은 필연적이며 10Gbps의 회선에서도 약 10Mbyte의 최대 처리량만을 얻을 수 있다[12]. 그렇기 때문에 회선속도(line-rate)에 근접하는 수준으로 오버레이 멀티캐스트 패킷을 처리하기 위해서는 하드웨어에서 터널링, 패킷 복제, 포워딩(Forwarding) 기능을 구현해야만 한다.

3. 시스템 기능 설명

본 논문에서 구현한 시스템은 멀티미디어 데이터 전송을 위한 NetFPGA 플랫폼과 오픈플로우 프로토콜 기반의 고성능 오버레이 멀티캐스트 패킷 전송 엔진이다. 더 나아가 기존의 IP 스위치로서 동작하는 오픈플로우 프로토콜을 개선하여 IP 스위치 기능뿐만 아니라 오버레이 멀티캐스트를 가능하도록 하였다. 이를 이용하여 멀티캐스트 기능을 사용하지 않고 있는 라우터들로 구성된 망에 오버레이 네트워크를 구성하여 고속으로 멀티캐스트 패킷을 전송한다.

IP 스위치 기능을 위해 포워딩 테이블, ARP, MAC learning 기능을 구현하였으며 오버레이 네트워크 기능을 위해 터널링을 위한 캡슐화/역캡슐화 기능을 구현하였다. 또한 IP 멀티캐스트 기능을 위해 IGMP와 패킷복제 기능을 구현하였으며 포워딩, 터널링, 패킷복제 기능을 구현하는데 필요한 플로우 테이블을 구현하였다.

가. 소프트웨어와 하드웨어 분류

NetFPGA 플랫폼의 FPGA 용량의 한계로 인해 모든 기능을 하드웨어로 구현하지는 못한다. 그렇기 때문에 앞서 나열한 기능들 중 제어평면(Control Plane)에 해당하는 기능들과 대용량의 멀티미디어 데이터를 고속으로 멀티캐스팅 하는 것과 별도로 동작하는 ARP 처리 기능, MAC learning 기능, IGMP를 이용한 IP 멀티캐스트 그룹 가입/탈퇴 기능, L2 스위치 기능, 하드웨어에 구현된 플로우 테이블을 관리하는 기능은 소프트웨어로 구현하였다. 이와 반대로 데이터 평면(Data Plane)의 기능으로 분류되어 실시간으로 처리가 요구되는 플로우 테이블을 이용한 L3 스위치 기능과 터널링을 위한 캡슐화/역캡슐화 기능 그리고 다수의 노드로 패킷을 복제하는 기능은 하드웨어로 구현하였다.

Ⅳ. 하드웨어 구현

1. 전체 구조

오버레이 멀티캐스트 전송을 위한 하드웨어 구조는 <그림 4>와 같다. 이 그림은 패킷의 흐름에 따른 구조도이다. 이더넷으로부터 입력되는 패킷이나 소프트웨어에서 생성한 패킷은 Input buffer를 거쳐 SRAM interface를 통해 외부 SRAM에 일차적으로 저장된다.

그 후 다시 순차적으로 읽혀진 패킷은 Packet Processing 과정을 거친다. SRAM에 패킷을 저장하는

이유는 Packet Processing모듈에서의 처리속도보다 입력속도가 더 클 경우 패킷 손실을 방지하기 위해서다. Packet Processing 에서는 포워딩, 캡슐화/역캡슐화, 패킷 복제 동작을 수행하여 입력된 패킷이 다지점으로 전송되도록 한다. Packet Processing 과정을 거친 패킷

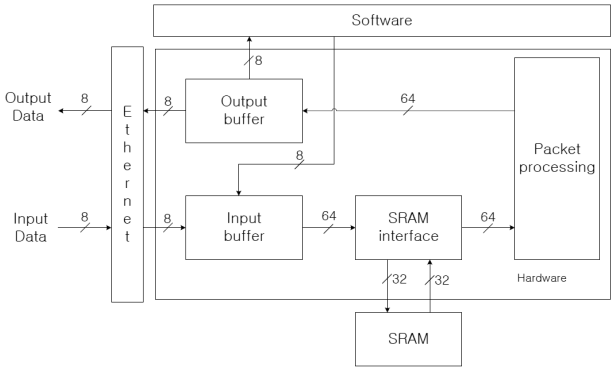


그림 4. 하드웨어 시스템 구조도
Fig. 4. Hardware system block diagram.

은 Output buffer를 통해 이더넷으로 전송되거나 상위 소프트웨어로 전송된다.

2. Packet Processing 모듈

Packet Processing 모듈은 입력된 패킷을 어디로 보낼지 결정 하거나, 필요에 따라 MAC헤더와 IP헤더 정보를 변환하거나, 캡슐화/역캡슐화 등을 수행하며 그 구조는 <그림 5>과 같다. 입력된 패킷은 FIFO(First In First Out) 메모리에 저장되며 동시에 MAC header parser, IP header parser, Match processing모듈에 입력되어 처리된다.

Packet processing 모듈에서 한 클럭당 처리하는 데이터의 비트수는 64비트이기 때문에 현재 클럭에서 입력된 데이터가 패킷의 몇 번째 워드인지 알아야 한다. 이 내용은 Preprocess controller 모듈에서 판단하여 MAC header parser, IP parser, Match processing 모듈로 알려 준다.

MAC header parser모듈과 IP header parser모듈은

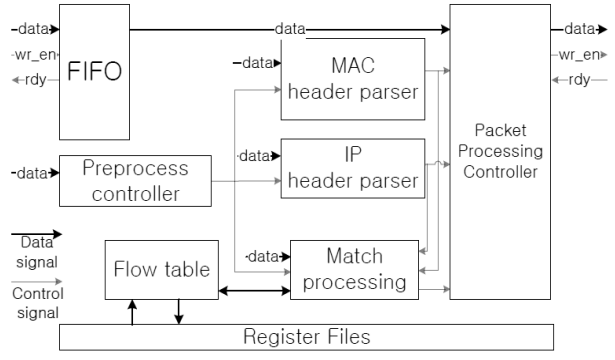


그림 5. Packet Processing 모듈 구조도
Fig. 5. Packet Processing module block diagram.

각각 MAC 헤더정보와 IP 헤더정보를 분석하여 출발지/목적지 MAC주소, Ethernet type, 출발지/목적지 IP 주소 등의 내용을 Packet Processing Controller와 Match processing모듈로 알려 준다. Match processing모듈은 이 정보를 이용하여 플로우 테이블을 검색하여 패킷을 어떻게 처리할지에 대한 정보를 얻어서 Packet Processing Controller로 알려준다.

여기까지가 전처리 과정이며 전처리 과정에서 얻은 정보를 이용하여 Packet Processing Controller는 출력 포트를 정해주거나, MAC헤더와 IP헤더 정보를 변경하거나, 패킷을 복제하여 다지점으로 보내주거나, 캡슐화/역캡슐화 동작을 한다.

가. 플로우 테이블 구현 및 동작

플로우 테이블은 두 종류로 구현하였다. 하나는 Xilinx사의 FPGA 내부에 하드 IP 형태로 포함된 메모리인 BRAM(Block RAM)과 또 하나는 TCAM (Ternary Content Addressable Memory)으로 구현 하였다[13][14]. CAM은 일반 메모리처럼 주소를 이용하여 데이터를 읽는 것이 아니라 '0' 또는 '1'을 이용한 검색어를 제공하면 자신의 메모리 공간 전체를 탐색하여 해당 검색어가 위치하고 있는 주소 및 경우에 따라서는 검색어와 연관된 데이터를 반환한다. TCAM의 경우에는 '0', '1' 뿐만 아니라 'don't care'비트까지 검색어를 제공한다.

플로우 테이블을 BRAM과 TCAM두 개로 구현한 이유는 다음과 같다. TCAM의 경우 검색어를 이용하여 고속으로 검색이 가능한 메모리이며 'don't care'비트를 사용하여 검색을 하기 때문에 네트워크 구성에따라 실제 항목보다 수 배에서 많게는 수 백배 많은항목을 가지고 검색하는 효과가 있다. 그러나 구현 시 검색 가능한 항목의 개수에 비해 하드웨어 사용량이 큰 단점을 가지고 있기 때문에 한정적으로 사용할 수 밖에 없다. 이를 보완하기 위해 검색 속도는 느리지만 보다 간단한 하드웨어로 많은 항목을 검색할 수 있는 RAM을 병용하여 구현하였다.

나. 터널링을 위한 캡슐화/역캡슐화 구현 및 동작

터널링을 위한 캡슐화는 기존의 IP 패킷에 또 다른 헤더를 추가 시키는 것이다. 캡슐화 하는 방법은 기존의 패킷을 전송하기 전에 새로운 헤더를 생성하여 다음

단계 모듈로 보낸 다음 연이어서 기존의 패킷을 보낸다. 역캡슐화 방법은 이와 반대로 받은 헤더를 삭제 시킨 후 내부에 있던 패킷만 다음 단계의 모듈로 전송시킨다.

3. ASIC 칩 제작

FPGA에서 동작 검증을 완료한 부분 중 가장 많은 연산처리를 담당하는 Packet Processing 부분을 0.13um 공정으로 칩 제작을 완료하였다. 칩 레이아웃과 세부 성능은 <그림 6>과 <표 1>와 같으며 Verilog HDL 코드는 Synopsys사의 Design Compiler를 이용하여 합성하였으며 그 결과 732,052 게이트 가운트를 가지며 1.3Mbits의 메모리를 사용하였다.

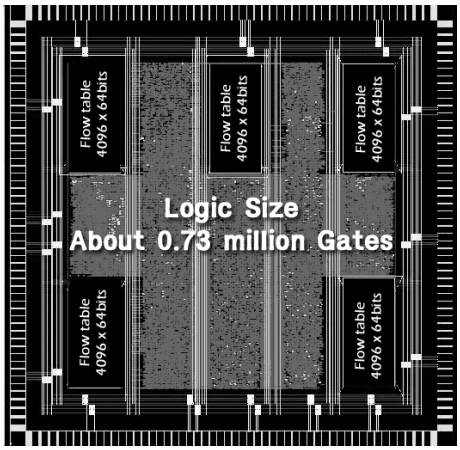


그림 6. 칩 레이아웃
Fig. 6. Chip layout.

표 1. 칩 성능
Table 1. Chip specification.

동작 클럭	125MHz
공정	0.13um
로직 사이즈	732,052gates
메모리	1.3Mbits

Ⅴ. 결과 및 분석

하드웨어 성능은 <표 2>에서 보는 것과 같다. 동작 클럭은 125MHz 이며 내부 메모리는 1.5Mbits인데 이는 대부분 패킷 저장용으로 사용된다. 하드웨어 내부에서의 패킷 처리 속도는 최대 3.6Gbps가 된다.

표 2. 하드웨어 성능
Table 2. Hardware specification

FPGA	Xilinx Virtex-2 Pro 50
동작 클럭	125MHz
내부 메모리	1.5Mbits
외부 메모리	SRAM 4.5MBytes DDR2 64MBytes
최대 패킷 처리 속도	3.6Gbps
로직 사이즈	38,534LUTs

설계에 대한 검증은 상위 프로토콜 소프트웨어를 동작시키지 않고 하드웨어만 이용하여 수행하였다. 검증에 사용한 툴은 패킷 캡쳐 프로그램인 Wireshark와, 패킷 생성 프로그램인 PackETH, 네트워크 속도 측정 툴인 iperf를 사용하였다. 네트워크 구성은 NetFPGA가 설치된 리눅스 컴퓨터 한 대와 다른 리눅스 컴퓨터 두 대를 NetFPGA 상에 구현한 멀티캐스트 엔진에 바로 연결하여 구성 하였다. 검증은 기능 검증과 성능 검증으로 나눠서 진행 하였다. 기능 검증 항목으로는 터널링을 위한 캡슐화/역캡슐화 기능 검증, 패킷 복제 기능 검증이 있다. 성능 검증 항목으로는 성능에 영향을 가장 많이 끼치는 패킷 복제 개수에 따른 속도를 검증하였다.

1. 터널링 기능 검증

터널링 기능 검증은 캡슐화 기능검증과 역캡슐화 기능검증으로 구성된다.

패킷 송신 측에서는 일반 IP 패킷을 보내고 수신 측에서는 IP 헤더가 한 개 더 붙은 IP-in-IP 패킷을 수신하는 것을 확인하는 방법으로 검증을 진행하였다.

검증 결과 송수신이 모두 잘 되었으며, 새로이 IP 헤더가 추가된 것도 확인 할 수 있었다. 또한 MAC 주소도 변경되어 IP 헤더가 추가된 패킷이 Layer-2 에서도 스위칭이 됨을 알 수 있었다.

역캡슐화 기능 검증은 위와 반대로 진행하였다. 송신 측에서는 IP 헤더가 한 개 더 붙은 IP-in-IP 패킷을 보내고 수신 측에서는 외부 IP 헤더가 제거된 일반 IP 패킷을 수신하는 것으로 검증을 진행 하였다.

검증 결과 송수신이 모두 잘 되었으며 역캡슐화 후 추가 되었던 IP 헤더가 제거되고 원본 패킷이 유지되었음을 확인 할 수 있었다.

2. 패킷 복제 기능 검증

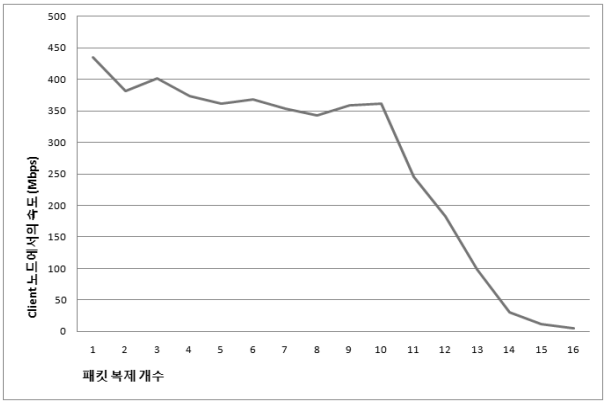
패킷 복제 기능 검증은 패킷을 서로 다른 목적지로 복제 시키는 방법으로 하였다.

첫 번째 복제된 패킷과 두 번째 복제된 패킷 모두 송수신이 잘 되었으며, 목적지 주소 또한 원하는 주소로 변경이 되어 서로 다른 지점으로 복제되어 전송된 것을 확인할 수 있었다.

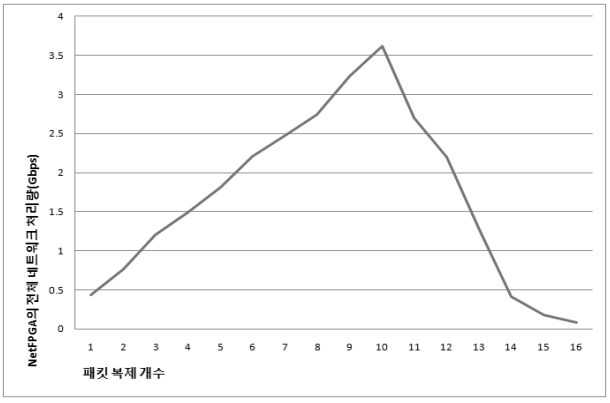
3. 하드웨어 성능 검증

하드웨어 성능 검증은 하드웨어에서의 패킷 처리량을 측정하여 검증하였다. 별도의 네트워크 측정 장비 없이 iperf 프로그램을 사용하여 송신측에 연결된 두 대의 PC에서 측정하였으며 복제개수에 따른 하드웨어의 최대 처리량을 계산 하였다. 패킷의 크기는 처리 가능한 최대크기인 1500바이트로 고정하여 실험 하였다.

<그림 7-(a)>는 이더넷 포트 0, 3에 연결된 PC에서 측정한 처리량 수치이다. 네트워크 측정 장비의 한계로 인해 최대 전송량은 450Mbps이다. 패킷을 1~10개 사이로 복제했을 경우 350Mbps~450Mbps 사이의 속도로 안정된 수치가 나왔지만 11개 이상으로 복제 했을 때부터는 복제되는 패킷의 수에 따라 처리량이 떨어지는 것을 알 수 있었다. 이를 통해 하드웨어에서 패킷을 10개 까지 복제 하는 동작에서는 성능 저하가 발생하지 않는 것을 알 수 있었다.



(a) 개별 노드에서의 복제된 개수에 따른 처리량 수치



(b) NetFPGA의 전체의 처리량 수치

그림 7. 복제 개수 증가에 따른 처리량의 변화
Fig. 7. Change in the throughput as an increase in the number of duplications.

<그림 7-(b)>는 <그림 7-(a)>의 결과 값에 패킷 복제 횟수만큼을 곱한 NetFPGA 내부 전체 처리량의 이론수치이다. 10개 이하로 패킷을 복제 하더라도 패킷 처리의 속도가 저하되지 않기 때문에 처리량이 400 Mbps에서 3.6Gbps 까지 안정적으로 증가하는 것을 볼 수 있었으며 NetFPGA내부의 최대 처리량이 3.6Gbps임을 알 수 있었다.

하드웨어의 동작 클럭은 125MHz이며 1500Bytes의 데이터를 처리하는데 350클럭이 소모된다. 이를 이용하여 계산한 처리량은 약 4Gbps이며 실험에서 나온 최대 처리량 결과와 비슷한 수치를 보여주었다.

VI. 결 론

본 논문에서는 NetFPGA 플랫폼을 이용하여 고성능 오버레이 멀티캐스트를 위한 하드웨어 엔진을 설계 및 구현 하였다. 오버레이 네트워크를 위한 터널링 기능을 구현하였고, IP 멀티캐스트를 위한 패킷 복제 기능을 구현하였다. 패킷 처리를 위한 정보가 담겨있는 플로우 테이블 구현에 BRAM과 TCAM을 이용하여 사용되는 하드웨어의 양을 최소화 하였다. 처리량을 증가시키기 위해 파이프라인으로 구현 하였으며 IP 스위칭 가능 하도록 MAC 헤더와 IP 헤더를 변경 할 수 있게 구현 하였다. 패킷 처리를 위한 전처리 과정은 속도 증가시키기 위해 병렬처리된다.

구현된 하드웨어를 이용하여 가장 긴 수행 시간이 걸리는 패킷 복제에 따른 처리량 검증을 한 결과 복제 개수가 1개부터 10개 까지는 1기가비트 이더넷 상에서 오버 플로우(Over flow) 없이 안정적으로 동작하는 것을 확인 하였으며 최대 처리량이 3.6Gbps임을 확인 하였다. 그러나 복제 개수가 11개 이상인 경우에는 NetFPGA의 내부 처리량의 한계 때문에 속도가 점차 떨어지는 것을 확인할 수 있었다.

참 고 문 헌

[1] 손승철, 광용완, 허권, 이형옥, 남지승, “오버레이 멀티캐스트를 위한 HDTV 중계전송 시스템 설계 및 구현”, 한국통신학회논문지, Vol. 32 No. 1A, 2007년

[2] Christophe Diot, Brian Neil Levine, Bryan Lyles, Hassan Kassem, Doug Balensiefen, “Deployment Issue for the IP Multicast Service and Architecture”, Network, IEEE, 2000.

[3] 고석주, 강신각, “인터넷 멀티캐스트 신기술 동향”, 전자통신동향분석, 16권 2호(통권 68), 2001년

[4] Sherlia Y. Shi, Jonathan S. Turner, “Routing in overlay multicast networks”, INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2002.

[5] 강호종, 송황준, 민경원, “실시간 동영상 오버레이 멀티캐스트 시스템”, 한국통신학회논문지, 제31권 2C호, pp.139~147, 2006년

[6] Nick McKeown 외, “OpenFlow : enabling innovation in campus networks.” http://www.openflowswitch.org

[7] Chang-Soo Ha, Jong Hyoung Lee, Duck Soo Leem, Myoung-Soo Park, Byeong-Yoon Choi, “ASIC Design of IPSec Hardware Accelerator for Network Security”, IEEE Asia-Pacific Conference on Advanced System Integrated Circuits, 2004

[8] Pankaj Gupta, Steven Lin, Nick McKeown, “Routing Lookups in Hardware at Memory Access Speeds”, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1998

[9] Fahmy S, Minseok Kwon, “Characterizing Overlay Multicast Networks and Their Costs”, Networking, IEEE/ACM Transactions on.

[10] Xing Jin, Wanqing Tu, Chan. S -H.G, “Challenges and advances in using IP multicast for overlay data delivery”, Communication Magazine, IEEE.

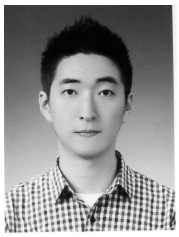
[11] Virtex-II Pro and Virtex-II Pro X FPGA User Guide, http://www.xilinx.com/support/documentation/user_guides/ug012.pdf

[12] 유태완, 최훈규, 손석신, 권태경, 최양희, “NetFPGA를 이용한 멀티캐스트 오버레이 네트워크 설계”, 서울대학교 컴퓨터 공학부, internal document.

[13] Block RAM Block Data Sheet, http://www.xilinx.com/support/documentation/ip_documentation/bram_block.pdf

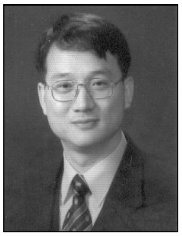
[14] Content-Addressable Memory Data Sheet, http://www.xilinx.com/support/documentation/ip_documentation/cam_ds253.pdf

저 자 소 개



전 혁 진(정회원)-교신저자
2009년 광운대학교 전자통신 공학과 학사 졸업.
2011년 광운대학교 전자통신 공학과 석사 졸업.

<주관심분야 : 통신, 반도체, SoC설계, 입베디스 시스템 설계>



정 용 진(정회원)-교신저자
1983년 서울대학교 제어계측 공학과 학사 졸업.
1983년 3월~1989년 8월 한국전자 통신연구원.
1995년 미국 UMASS 전자전산 공학과 박사 졸업.
1995년 4월~1999년 2월 삼성전자 반도체 수석 연구원.
1999년 3월 광운대학교 전자통신공학과 부교수
<주관심분야 : 무선통신, 정보보호, SoC 설계, 영상처리 및 인식, 임베디드 시스템>

이 현 석(정회원)
1992년 KAIST 전기및전자공학과 공학사.
1995년 POSTECH 전자전기과 공학석사.
2007년 Univ. of Michigan, Computer Science and Engineering (CSE), Ph.D.
1992년~2008년 삼성전자 통신연구소 수석연구원
2008년~현재 광운대학교 전자통신공학과 조교수
<주관심분야: Software defined radio, 통신용 DSP구조, Embedded system, VLSI>