내장형 네트워크 프로세서의 설계 및 구현

정진우*·김성철*

Design and implementation of an Embedded Network Processor

Jinoo Joung* · Seong-cheol Kim*

요 약

Embedded system은 소수의 System-On-Chip (SOC)으로 대부분의 기능이 구현되어지는 추세이며, 이러한 SOC의 구조는 대체로 RISC 기반의 내장 마이크로프로세서를 중심으로 발전해 왔다. 하지만 RISC 기반의 ARM, MIPS등의 범용 프로세서들은 점차 그 필요성이 커지고 있는 네트워크 기능과 멀티미디어 처리 기능 등에 대해서는 많은 고려 없이 설계된 프로세서들이다. 소규모 사업자 및 개인 사용자를 위한 네트워크 기기의 경우는 가격대비 성능이 우수한 제품이 시장을 차지하는데 유리하므로, 지금까지 대부분의 경우에서 전용 하드웨어를 사용하지 않고, PHY와 MAC layer 일부의 기본적인 기능을 제외한 나머지 네트워크 기능을 모두 상기한 내장 마이크로프로세서로 처리하고 있다. VDSL, FTTH과 같이 고속 인터넷을 가능하게 하는 기술이 발전함에 따라, 기존의 범용 프로세서에 기반을 둔 네트워크 기기는 빠른 속도로 그 성능의 한계에 다다르고 있다. 이는 단순히 프로세서의 동작 속도를 높이는 것으로 해결할 수 있는 문제가 아닌 것으로 보이며, 네트워크 프로토콜의 처리에 최적화 되어 있지 않은 범용 프로세서의 사용에 근본적인 문제점이 있다고 하겠다. 본 연구를 통하여 네트워크 기능 수행에 효율적인 네트워크 프로세서를 설계하고 이를 Home gateway용 SOC에 내장하고 성능을 측정하여 그 상용화 가능성을 타진한다.

ABSTRACT

Current generation embedded systems are built around only a small number of SOCs, which are again based on general-purpose embedded micro-processors, such as ARM and MIPS. These RISC-based processors are not, however, designed for specific functions such as networking and multimedia processing, whose importances have increased dramatically in recent years. Network devices for small business and home networks, are especially dependent upon such SOCs based on general processors. Except for PHY and MAC layer functions, which are built with hardware, all the network functions are processed by the embedded micro-processor. Enabling technologies such as VDSL and FTTH promise Internet access with a much higher speed, while at the same time explore the limitations of general purpose microprocessors. In this paper we design a network processor, embed it into an SOC for Home gateway, evaluate the performance rigorously, and gauge a possibility for commercialization.

키워드

네트워크 프로세서, Embedded System, System-on-Chip (SOC), Network Address Translation (NAT)

* 상명대학교 소프트웨어학부

접수일자 : 2005. 5. 4

Ⅰ. 서 론

Embedded system은 소수의 System-On-Chip (SOC)으로 대부분의 기능이 구현되어지는 추세이며, 이러한 SOC의 구조는 대체로 RISC 기반의 embedded processor를 중심으로 발전해 왔다. 따라서 대부분의 embedded SOC는 ARM, MIPS등의 market-dominant한 RISC 프로세서의 발전 방향에 따라 그 특성이 결정지어져 왔다. 하지만 ARM과 MIPS, 혹은 Motorola의 PowerPC계열의 dominant design들은 네트워크 기능이나 멀티미디어 기능 등의 특정 application에 대해서는 전혀 고려되지 않고 설계된 프로세서들이다. 이들은 networking application에는 매우 비효율적인 명령어 집합 구조 (Instruction Set Architecture)와 마이크로구조 또는 운영 체제를 기반으로 설계되어 있다. 하지만 최근의 embedded 프로세서의 용도를 살펴보면 단순 computing 기능보다는 networking, communication, signal processing등의 네트워크 data 전달에 필요한 기능에 치우쳐 있다. 이렇게 용도에 맞지 않는 프로세서 및 구조를 채용함으로써 power, performance, chip size 등의 측면에서 기회 손실적인 요소가 계속 증대되어 왔다. [1]. 상기한 바와 같은 문제점을 염두에 두고 현재의 중소형 네트워크 기기들을 살펴보면, PHY와 MAC layer 일부의 기본적인 기능을 제외한 나머지 네트워크 기능을 모두 ARM등의 embedded CPU로 처리하고 있음을 알 수 있다. 소규모 사업자 및 개인 사용자를 위한 네트워크 기기의 경우는 가격대비 성능이 우수한 제품이 시장을 선점하는데 유리하므로, 지금까지 대부분의 경우에서 전용 하드웨어를 사용하지 않고 프로그램 가능한 프로세서 상에서 소프트웨어적으로 기능을 구현하는 것이 보편적이다. VDSL과 같은 고속 인터넷을 가능하게 하는 기술이 발전함에 따라, 기존의 범용 프로세서에 기반을 둔 네트워크 기기는 빠른 속도로 그 성능의 한계에 다다르고 있다. 이는 단순히 프로세서의 동작 속도를 높이는 것으로 해결할 수 있는 문제가 아닌 것으로 보이며, 네트워크 프로토콜의 처리에 최적화 되어 있지 않은 범용 프로세서의 사용에 근본적인 문제점이 있다고 하겠다. Home network과 SOHO network에서 많이 사용되는 Internet gateway router를 구체적으로 살펴보면, IP NAT (Network address translation) processing의 주 기능을 100Mbps Ethernet 환경에서 처리하는 역할을 맡는다. 향 후 Cable 망, FTTH (Fiber to the home)등의 기술이 일상화되면 100Mbps 이상, 1Gbps 정도 까지 처리해 주어야할 필요가 있다. 이러한 data rate의 NAT 처리는 범용 RISC로는 불가능하다. 따라서 본 연구에서는 chip화 했을 경우 1Gbps로 NAT를 처리할 수 있게 하는 여러 가지 방안을 고려 해 보고 그 중 하나로 SOC에 내장되는 네트워크 프로세서를 개발하는 것을 목표로 삼는다.

비용을 현재의 수준으로 유지하면서, 높아지는 네트워크의 요구조건을 맞출 수 있는 방법은 네트워크 프로토콜의 처리에 최적화 되어 있는 프로그램 가능한 보조 프로세서를 설계하여, 기존의 범용 프로세서로는 효율성이 떨어지던 부분을 대치하는 것이 좋은 대안이 될 수 있다. 이 방법에는 다음과 같은 선행 조건이 따르게 된다. 모든 기능이 소프트웨어적으로 구현되어 있는 기존의 시스템 상에서 어떠한 부분을 하드웨어로 전환하는 것이 좋은 것인지를 결정해야 하며, 보조 프로세서가 데이터를 처리하고 있는 동안 주 프로세서가 어떤 일을 수행 할 수 있도록 스케줄링 할 수 있는 방법이 있어야 할 것이고, 마지막으로 추가되는 보조 프로세서는 항상 변화하고 있는 네트워크 프로토콜 환경에 대응하기 위하여 높은 수준의 유연성, 프로그램 가능성을 갖추고 있어야 하며, 동시에 하드웨어 고유의 효율성을 유지하여야 한다[2]. 본 연구는 하나의 프로세서를 기반으로 구성된 네트워크 라우터에 대해서 여러 가지 응용 프로토콜과 여러 상황 하에서의 성능을 분석하여 효율적인 동작을 방해하는 부분을 파악하고, 해결책으로 네트워크 응용 분야에 최적화된 보조 프로세서를 설계하는 것을 그 목표로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 출시된 상용 SOC의 성능을 평가한다. 3장에서 네트워크 프로세서의 기능과 구조를 정의하고 이를 내장하는 SOC 전체의 구조를 결정한다. 이를 바탕으로 평가 시스템을 구현하여 성능을 측정한다. 4장에서는 측정한 성능을 바탕으로 네트워크 프로세서의 상업적 성공 가능성을 타진하고, 구현 시에 겪었던 어려움과 이를 극복할 수 있는 방안에 대해 토의한다.

Ⅱ. 기존 시스템 성능 평가

우선 현재 출시되어 있는 상용 SOC의 성능을 평가하기 위해서 Internet gateway나 SOHO용 router의 핵심 SOC로 가장 많이 판매되는 제품 중 하나인 S3C2510을 선택하여 이를 기반으로 시스템을 구성하였다. S3C2510은 ARM940T를 내장하고 있으며 별도로 2개의 Fast Ethernet MAC 하드웨어 module을 가지고 있다. MAC에서 처리된 Ethernet frame이 DMA 방식으로 ARM CPU로 전달되면 IP layer에서 필요한 처리는 이 CPU가 담당한다[3]. 이 chip의 evaluation board인 SMDK2510을 이용하여 Internet gateway기능을 emulate하였다. 이 emulator에 test packet을 넣어주는 장치로 일반 PC와 Smartbit장비를 준비하였다. 그림 1은 이러한 성능 평가 환경을 보여준다.

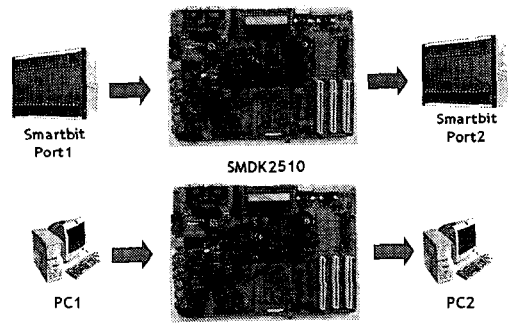


그림 1. 기존 시스템 성능 평가 환경
Figure 1. Test environment for performance evaluation of an existing system

Smartbit장비로 여러 가지 다른 크기의 IP packet을 만들어서 continuous하게 입력하여 성능을 평가하였고, PC로는 FTP application을 사용하여 커다란 message를 양 방향으로 전달케 하여 성능을 측정하였다. 모든 경우에 있어서 Chip의 동작 속도는 166Mhz로 일정하게 유지하였다. 실험 결과는 아래의 표 1과 2에 나타내었다.

표 1. Smartbit를 사용하여 측정한 성능
Table 1. Performance probed with Smartbit equipment

Ethernet frame size (byte)	Throughput (Mbps)
1514	52.18
256	9.14
64	2.33

표 2. PC의 FTP기능을 이용하여 측정한 성능
Table 2. Performance probed with PC using FTP application

FTP direction	Throughput (Mbps)
PUT	30.50
GET	40.60

위 표의 실험 결과에서 볼 수 있듯이 최대 크기의 IP packet을 처리하는데 있어서 50Mbps정도의 성능을 나타낸다.

Ⅲ. SOC 구조 및 네트워크 프로세서 구현

1Gbps의 NAT 처리가 가능한 네트워크 프로세서를 구현하고 이를 내장하기 위해서 무엇보다 먼저 과연 어떠한 기능이 네트워크 프로세서에서 처리되어야 할지가 결정되어야 한다.

3.1. Function definition

흔히 router나 switch와 같은 네트워크기기들의 기능은 크게 control-plane과 data-plane으로 나눈다. 그림 2의 개념도에서 Control & Management로 표시된 부분이 control-plane이며 routing algorithm, management, configuration등의 일을 포함한다. Data-plane은 channel management, forwarding table과 routing table의 update, statistics update등의 "slow-path" data processing과 Medium Access Control (MAC), Segmentation and Reassembly (SAR), 그리고 그 밖의 Data manipulation, Lookup, Classification, Policy enforcement, QoS handling 등의 "fast-path" data processing을 포함한다. 기본적인 네트워크 기능인, Packet header를 검색해서 destination address정보를 추출하여 다음 hop을 결정한다든지, 혹은 header의 error를 check한다든지 하는 packet-by-packet operation은 fast-path data-plane 에 해당하며 본 연구에서 개발될 네트워크 프로세서는 이 기능을 수행하기로 한다. 기존 고가의 네트워크 장비용 네트워크 프로세서들도 주로 data-plane의 기능을 수행하는 것을 볼 수 있다[4].

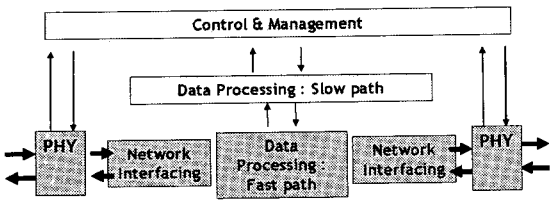


그림 2. 네트워크 기기의 기능블록 개념도
Figure 2. Conceptual function blocks of network devices

본 연구에서 다루는 Internet gateway 장비에서의 fast-path data processing은 Ethernet MAC과 IP/NAT packet processing이며 구체적으로 다음의 기능을 포함한다.

1) Ingress processing: IP over Ethernet packet이 도착하면, 기본적인 MAC receive processing을 수행하고, packet을 MAC block의 FIFO에 저장한 후, IP header를 추출해서 프로세서로 넘긴다. Packet 전체를 외부의 memory에 저장한다.

2) Main processing: IP version, header length를 확인하고 header checksum을 check하여 error 여부를 결정한다. 기존의 NAT connection 정보를 찾는다. 만약 찾지 못 하는 경우는 NAT table을 lookup한다. 지정한대로 address변경을 한 후에 새로운 NAT connection 정보를 집어넣는다. Source address와 destination address 정보를 기반으로 필요한 경우에 filtering을 하고 Time to live (TTL) field를 check한다. Destination address를 가지고 다음 port를 결정하고 (IP address lookup) TTL을 감소시킨 후 Header checksum 값을 update한다. 이렇게 변경된 packet header를 외부 memory에 저장된 packet에 overwrite한다.

3) Egress processing: 외부 memory의 packet을 FIFO로 읽어온 후, 기본적인 MAC transmission processing을 통해서 내보낸다.

상기한 세 가지 기능 중 Main processing을 네트워크 프로세서가 담당하고, Ingress와 Egress processing은 MAC 하드웨어 block에서 담당하기로 한다.

3.2. 네트워크 프로세서 구조

정의된 기능을 기존의 ARM instruction으로 처리했을 때를 살펴본 결과, 복잡한 bit manipulation 동작이 많았으며, data move (load, store 등) instruction과 brach, jump등의 instruction이 각각 27.7%와 10.9%를 차지할 정도로 많이 사용되었음을 알 수 있었다. 이러한 분석 결과를 바탕으로 이 들 동작에 특화된 instruction들과 data 이동이 병렬적으로 처리 가능한 instruction들을 포함하는 Instructions set architecture (ISA)를 결정하였다. 추가적으로 assembler와 compiler, Instructions set simulator (ISS) 등 소프트웨어 개발용 toolset을 구축하였다.

3.3. SOC 구조

네트워크 프로세서의 구조를 결정한 후, SOC 구조를 어떻게 가져가야할 지를 정하는 것이 다음으로 해야 할 일이다. 먼저, 목표 성능을 만족시키기 위해 네트워크 프로세서가 몇 개가 필요한지를 알아볼 필요가 있다. 1500Byte IP packet을 Gigabit Ethernet으로 전송하는데 걸리는 시간은 약 13us이다. SOC의 clock 속도를 166Mhz로 가정했을 때 13us는 2158 cycle에 해당한다. 네트워크 프로세서의 instruction을 분석한 결과, Packet 하나의 IP/NAT 처리에 필요한 instruction의 개수는 688개 이며, 이 중 385개는 register/immediate operation이고 나머지 303개의 instruction이 memory access가 필요하다. 이 memory access에 대략 5 cycle정도가 걸린다고 가정하면 IP/NAT 처리에는 1900 cycle이 필요하다. 이를 바탕으로 네트워크 프로세서 하나가 충분히 1Gbps 성능을 확보한 다는 결론을 얻을 수 있었다.

다음으로는 네트워크 프로세서의 SOC 내부 위치를 결정하는 것이 필요하다. 기존 block인 MAC과 ARM CPU와의 관계를 고려하면 그림 3과 같은 세 가지 후보를 생각할 수 있다.

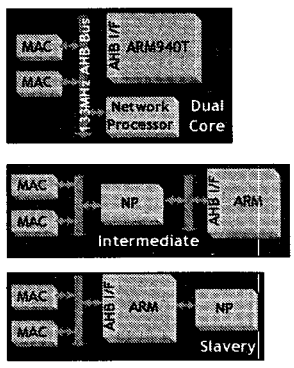


그림 3. SOC 구조 후보 세 가지
Figure 3. Three candidates for SOC architecture

세 가지 후보 구조들 중, Dual core 구조는 기존 소프트웨어의 변경에 있어서 유리하며, Intermediate 구조는 data throughput을 최대로 이끌어 낼 수 있다. Slavery 구조는 ARM를 이용해서 전체 SOC를 관리하는 데에 편한 장점이 있다. 본 연구에서는 목표인 data throughput에 주안점을 두고 intermediate 구조를 선택하여 최종적으로 그림 4와 같은 SOC를 구현하기로 결정하였다.

1절에서 언급된 ingress processing과 egress processing을 담당하는 Ingress controller와 Egress controller, 그리고 이들 controller들과 네트워크 프로세서간의 packet 교환을 관장하는 Queue manager를 하드웨어로 설계하였다. 네트워크 프로세서는 외부에서 입력되는 모든 packet에 대해서 일차 fast-path processing을 처리하며, error나 기타 exceptional case에 대해서 ARM CPU로 보고하고 ARM은 필요한 경우 packet을 직접 외부 memory로부터 불러들일 수 있다.

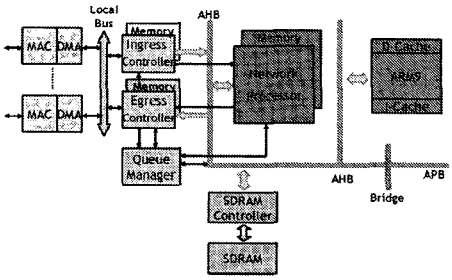


그림 4. 최종적으로 채택한 SOC 구조
Figure 4. Final SOC architecture

3.4. 구현된 SOC의 성능 측정

IP/NAT 처리용 application 소프트웨어, 하드웨어 block 및 네트워크 프로세서 block의 전체적인 성능 및 기능검증을 위해 그림 5와 같이 FPGA를 이용하여 evaluation board를 구현하였다.

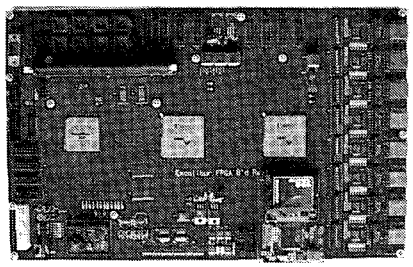


그림 5. FPGA를 이용한 검증용 보드
Figure 5. Evaluation board with FPGAs

검증용 보드는 Altera의 Excalibur와 Xilinx의 Virtex 8000 (XC2V8000) FPGA 두개를 사용하였다 [5][6]. Excalibur에 내장되어 있는 ARM922T core가 host 프로세서의 역할을 하였으며, 두개의 FPGA에는 각각 네트워크 프로세서와 Queue manager, 그리고 다른 하나에는 Ingress controller, egress controller와 8개의 Ethernet MAC을 내장하였다. Excalibur와 두개의 FPGA간의 connection을 위하여 기존 AMBA 2.0 AHB specification에 호환되는 interface block을 설계하여 각각에 추가하였다. 외부 interfacing 기능은 8개의 Fast Ethernet PHY chip을 사용하여 이들이 담당하게 하였다. 각 block들의 gate count는 표 3에 표시하였다.

표 3. Block별 gate 수 및 memory 크기
Table 3. Gate counts and memory sizes for each blocks

Block	NP	Ingress	Egress	QM	Total
Logic gate	46070	39527	38285	8024	132006
Memory	96Kbits	16Kbits	16Kbits	16Kbits	144Kbits

표 3에서 알 수 있듯이 전체적으로 gate 수는 13만 정도로 크지 않으며, 만약 SRAM 1bit를 1gate로 변환한다면 총 30만 gate 정도의 크기이다.

"1Gbps의 IP/NAT 처리"라는 목표성능을 만족하는지 여부를 알아보기 위해서 상기한 보드를 이용해서 다음과 같이 실험환경을 꾸몄다. FPGA로 동작 가능한 clock 주파수는 25Mhz 정도이다. 이는 실제 이 네트워크 프로세서가 chip으로 출시되었을 경우 동작하게 될 166Mhz와 비교하여 약 1/6.64 정도의 수치이다. 따라서 25Mhz로 동작하는 FPGA 보드가 155Mbps 이상의 성능을 보이면 목표 성능을 만족하는 것이라고 할 수 있다.

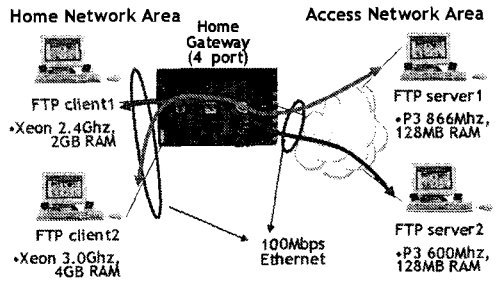


그림 6. 성능평가 실험 환경
Figure 6. Performance evaluation environment

실험 환경은 다음과 같이 꾸몄다. 먼저 대용량의 file을 준비하고, 그림 6과 같이 4개의 PC를 이용해서 두 개의 FTP session을 열어서 각각 100Mbps Fast Ethernet connection을 통하여 동시에 file을 전송한다. 실제로 file을 만들어 전송해본 결과 유효한 data throughput은 평균 80Mbps 정도임을 확인할 수 있었다. 이렇게 유효 throughput이 100Mbps이하로 나오는 데에는 여러 가지 이유가 있을 수 있으나, PC의 사양에 민감하게 변하는 것으로 보아 HDD access 시간이 가장 큰 요인 이라고 생각된다. 실험 결과 85Mbps로 양 FTP session이 문제없이 동시에 통신하는 것으로 나타났고, 따라서 Home gateway 역할을 하고 있는 FPGA board는 160Mbps 이상의 처리 속도를 보인다고 할 수 있다. 그림 7은 실험 결과를 screen capture한 것이다.

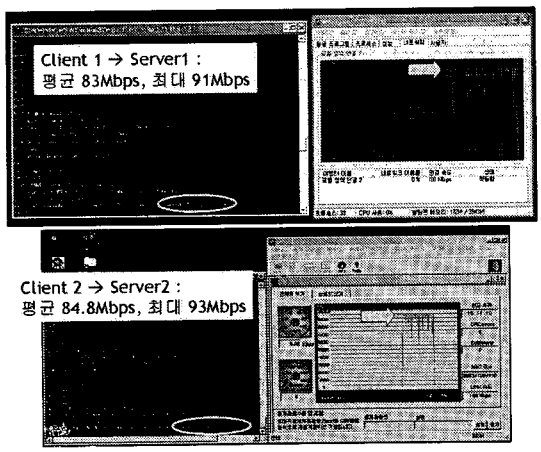


그림 7. 성능평가 실험 결과
Figure 7. A snapshot of the performance evaluation

Ⅳ. 결 론

본 연구에서는 Embedded system을 위한 SOC에 내장할 수 있는 네트워크 전용 프로세서를 설계하여 성능과 기능을 검증하였고, 비교적 적은 gate 수의 하드웨어로 상대적으로 높은 성능을 얻을 수 있음을 보였다. 이 결과는 네트워크 프로세서를 쓰지 않은 기존 제품에 비하여서는 물론이고, 기존 고성능 네트워크 프로세서 제품들과 비교했을 때도 오히려 뛰어난 것이다.

그림 8은 Intel 등에서 제시한 문서를 근거로 본 연구 결과 시스템과의 성능을 비교한 것이다. Intel의 high-end 프로세서인 IXP2800은 16개의 pipeline된 네트워크 프로세서를 1.4Ghz로 동작을 시키는 데 이 chip의 성능은 IP packet forwarding의 application에 대하여 20Gbps정도이다[7]. 이 수치를, 처리된 bit수/clock수/NP의 개수로 표시하면 대략 0.89정도의 값을 얻는다. Intel의 low-end 프로세서인 IXP425와 Wireless LAN AP용 chip인 Ubicom의 IP2000등의 제품에 이러한 방식으로 적용했을 때 0.75와 0.19의 값을 가진다는 것을 알 수 있다[8]. 본 연구에서 개발된 시스템은 6.4의 값을 가짐을 알 수 있다. 이렇게 월등한 결과가 나온 데에는 여러 가지 요인이 있지만 무엇보다 먼저 IP/NAT 처리라고 하는 specific한 application을 대상으로 전체 system과 네트워크 프로세서설계가 이루어진 점을 꼽을 수 있다. 좀 더 광범위한 application을 대상으로 실험을 해 본다면 시스템의 더 자세한 특성을 파악할 수 있을 것이다. 또 한 가지 주의할 것은 기존 상용 제품들의 성능이 최대 성능을 표시한 것이 아니라는 점이다. 따라서 이들과 본 연구 결과를 단순히 비교하는 것은 무리가 있다. 하지만 전체적으로 보았을 때 세계 최고 수준과 동일하거나 그 이상이라는 것은 확실하다.

	# of NP in SOC	NP clock speed	HW accelerators	Host Processor	Throughput	Bit/Cycle /NP
Intel IXP2800	16 pipelined	1.4Ghz	Queue Manager	Xscale (70Mhz)	20Gbps with IP packet forwarding	0.89
Intel IXP425	1 per port	133Mhz	Queue Manager	Xscale	100Mbps (per port) Ethernet switching	0.75
Ubicom IP2000	1	160Mhz	None	External	30Mbps 802.11b	0.19
Our System	1	25Mhz	Queue Manager	ARM940T (25Mhz)	160Mbps IP/NAT processing	6.4

그림 8. 성능 비교표
Figure 8. Performance comparison table

본 연구를 진행하면서 어려웠던 점은 기존 host 프로세서에서만 전적으로 처리되던 application 소프트웨어를 수정하여 일부를 네트워크 프로세서에서, 나머지는 기존 방식대로 host 프로세서에서 처리하게 하는 것이었다. 이를 위해 수작업으로 application을 분석하고, 기존 code를 수정하였으며 최종적으로 하드웨어가

완성된 이후에 co-simulation을 통해서 검증을 완료할 수 있었다. 가까운 시일 안에 Multi-processor를 내장한 SOC들이 high-end SOC의 주류로 등장할 것을 고려하면 이러한 소프트웨어의 수정을 손쉽게 하기 위한 개발 도구에 대한 연구가 필요할 것으로 보인다[9][10].

다른 한 편으로는, 특정 application에 대해서 ISA를 정의하면 이로부터 설계 자동화 tool의 도움으로 손쉽게 프로세서의 micro-architecture와 이에 상응하는 assembler, compiler등을 개발하는 것을 가능케 하기 위한 연구가 한창 진행 중이다[11]. 이러한 tool 관련해서는 소수의 업체들이 출현하고는 있지만 아직은 본격적인 상용화 단계에 이르지는 못했다. 하지만 이러한 연구가 어느 수준 이상이 되는 경우에 자동화 tool을 이용한 embedded application-specific 프로세서의 개발 방법론에 대한 논의가 본격화될 전망이며 이를 활용한 SOC 개발의 효율성을 검증해 볼 필요가 있을 것이다.

참고문헌

[1] Semiconductor Industry Association, The International Technology Roadmap for Semiconductors, 2003.

[2] P. Crowley, et. al., "Network Processor Design: Issues and Practices", Morgan Kauffman, vol. 1., Oct. 2002.

[3] H. Jang, et. al., "High-level system modeling and architecture exploration with systemC on a network SoC: S3C2510 case study", In Design, Automation and Test in Europe Conference (DATE'04), Paris, France, March 2004.

[4] N, Shah, "Understanding network processors", Master's thesis, Dept. of EECS, UC Berkerley, CA, USA, Sep. 2001

[5] http://www.altera.com/products/devices/arm/arm-index.html, Altera Excalibur product page

[6] http://www.xilinx.com/products/, Xilinx virtex product page

[7] Network Processor Forum, "IXP2800 Intel Network Processor IP Forwarding Benchmark Full Disclosure Report for OC192-POS", Oct. 30. 2003.

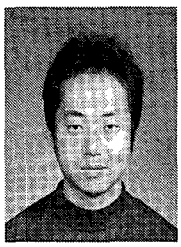
[8] http://www.ubicom.com/processors/ip2000-family.htm, UbiCom IP2000 product page

[9] W. Cesário, et. al., "Component-Based Design Approach for Multicore SoCs," Proc. of 39th Design Automation Conference, New Orleans, June 2002.

[10] Niraj Shah, et. al., "Comparing Network Processor Programming Environments: A Case Study", Workshop on Productivity and Performance in High-End Computing (P-PHEC), 10th International Symposium on High Performance Computer Architectures (HPCA), Feb 2004.

[11] Armita Peymandoust, et. al., "Automatic Instruction Set Extension and Utilization for Embedded Processors", Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03), IEEE COMSOC, 2003.

저자약력



정진우(Jinoo Joung)

1992 KAIST 전기전자공학과 공학사
1997 Polytechnic Univ., NY, USA, 공학박사 (Ph.D.)
1997~2001 삼성전자 중앙연구소
2001~2005 삼성종합기술원
2005~ 상명대학교

※ 관심분야: 컴퓨터 네트워크, SOC, Embedded systems, 무선 네트워크 및 통신

김성철(Seong-Cheol Kim)

1995. 6 Polytechnic Univ., NY, USA, 공학박사(Ph.D.)
1994. 6 ~1997. 2 삼성전자(주) 수석연구원
1997.2 ~ 현재: 상명대학교 부교수

※ 관심분야: WLAN, 초고속통신망, QoS, 멀티미디어 통신