논문 2008-45SD-4-12

SystemC를 이용한 아키텍처 탐색과 네트워크 SoC 성능향상에 관한 연구

(Architecture Exploration Using SystemC and Performance Improvement of Network SoC)

이 국 표*, 윤 영 섭**

(Kook Pyo Lee and Yun Sup Yoon)

요 약

네트워크 SoC 칩을 대상으로 SystemC를 이용한 High-level 설계 방법을 연구하였다. 실제 Verilog RTL 모델과 비교하여 깊이있는 Architecture 구조탐색과 정확한 SystemC 모델 cycle 검증을 토대로 하여 High-level 설계를 강조할 것이다. 대다수 High-level 설계와 접근방법과 다르게, SystemC 모델과 Verilog RTL 모델의 성능을 비교해 보고, SystemC-based platform을 검증하기 위해 On-chip test board 측정 데이터를 이용하였다. 이 논문에서는 High-level 설계기법이 RTL 모델과 같은 정확성을 얻을 수 있을 뿐만 아니라, RTL 모델보다 100배 이상 빠른 시뮬레이션 속도를 달성할 수 있음을 보여 주었다. 그리고, 아키텍처 구조탐색을 통해서 시스템 성능하락의 원인을 파악하고, 대안을 찾아보았다.

Abstract

This paper presents a high-level design methodology applied on an SoC using SystemC. The topic will emphasize on high-level design approach for intensive architecture exploration and verifying cycle accurate SystemC models comparative to real Verilog RTL models. Unlike many high-level designs, we started the project with working Verilog RTL models in hands, which we later compared our SystemC models to real Verilog RTL models. Moreover, we were able to use the on-chip test board performance simulation data to verify our SystemC-based platform. This paper illustrates that in high-level design, we could have the same accuracy as RTL models but achieve over one hundred times faster simulation speed than that of RTL's. The main topic of the paper will be on architecture exploration in search of performance degradation in source.

Keywords : Architecture Exploration, Ethernet, Architecture Performance, SoC, SystemC

Ⅰ. 서 론

SoC가 ASIC 시장에서 두드러지기 시작함에 따라, 성능과 가격의 경쟁이 증가되었고 시장은 급속도로 빠르게 움직이고 있다. 소비자들은 낮은 가격의 더 높은 성능의 칩을 요구하게 되고, 이를 만족시키기 위해 대다수 회사들은 제작 전, 또는 RTL 개발 전에 칩을 측정하고 분석하기 위해 High-level 설계 방법연구를 채택하려고 노력하고 있다.[1~2] 새로운 High-level 설계 방법을 이용하여, 대다수 칩 제조사들은 칩의 성능을 신속하게 확신하고 싶어하며, 확신을 바탕으로 새로운 칩을 개발하려고 한다. 이러한 요구에 부응하고 TAT (Turn-Around-Time)을 짧게 하기 위해, SystemC를 이용하여 회로를 설계하여 Transaction Level Modeling(TLM) 방법으로 시뮬레이션하는 방법이 고안되었다.[3~4] 일반적으로 HDL로 코딩된 RTL 수준의 시뮬레이션의 경우 실제 칩의 동작을 예측하는데 효과적이지만, 시뮬레이션 시간이 오래 걸려서 실제 응용 소프트웨어의 장시간 검증벡터를 이용한 시뮬레이션을 할 수 없으며 단지 간단한 function검증 용도로만 사용된다. 보다 실제적이고 깊이 있는 검증을 위해서

* 학생회원, ** 정회원, 인하대학교 전자공학과
(Dept. of Electronics Engineering, Inha University)
접수일자: 2007년10월31일, 수정완료일: 2008년3월19일

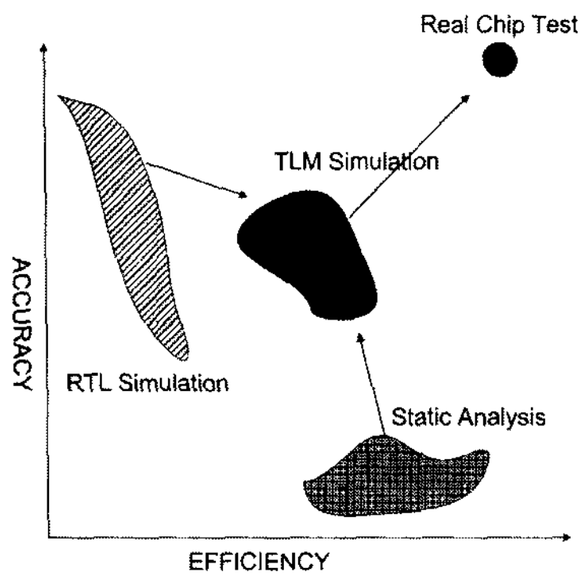

그림 1. RTL, TLM 시뮬레이션, 정적인 분석, 실제칩 테스트에 대한 효율성과 정확도 비교

Fig. 1. Comparision of efficiency and accuracy about RTL, TLM simulation, static analysis and real chip test.

SPECMAN, VERA 등의 function검증툴이 광범위하게 이용되고 있다. 그러나 function 검증툴의 경우, 칩의 오동작을 검증하는 것을 목적으로 하기 때문에, 실제 동작을 예측하고, 성능을 분석할 수는 없다.[5]

그림 1에 칩의 성능을 분석하기 위해서 사용되는 RTL 시뮬레이션, TLM 시뮬레이션, 정적인 분석, 그리고, 실제칩 테스트에 대한 특징이 나타나 있다. RTL 시뮬레이션은 실제칩 테스트에 근접할 정도로 정확도는 높지만 시뮬레이션 시간이 길어 효율적이지 못한 단점이 있으며, 반대로 정적인 성능분석은 분석시간은 적게 걸리지만 정확성이 심각하게 떨어진다. 결국 TLM 방식의 시뮬레이션이 실제칩에 근접한 성능분석을 위한 대안이 될 수 있다.

이에 본 연구에서는 RTL 수준의 시뮬레이션 보다 100배 이상 빠른 동작속도로 칩의 동작을 예측하고, 성능을 분석할 수 있는 SystemC 설계 플랫폼을 TLM방식으로 개발하려고 한다. 기존에 개발되어진 RTL 수준의 동작과 SystemC로 새로 개발된 플랫폼의 동작을 비교하여, 1% 이하의 정확도로 개별 블록 등을 SystemC로 구현하려고 한다. 이렇게 구현된 SystemC 플랫폼은 RTL 수준 보다 100배 이상 빠른 속도로 동작을 분석할 수 있으므로, 실제 응용 소프트웨어를 이용한 장시간 검증벡터를 이용한 시뮬레이션을 할 수 있으며 실제 응용에서 사용되는 리눅스 등 OS를 포팅한 임베디드시스템 응용코드를 시뮬레이션하는 것도 가능하다. 현재 인터넷 라우터 등에 사용되는 이더넷 칩의 리눅스 OS 포팅에 걸리는 시간은 1~2초 정도 소요되는데, RTL 시뮬레이션 속도가 너무 느려서 RTL 시뮬레이션 재현은 현실적으로 불가능하다. 그러나 본 연구에서 완성된 SystemC 플랫폼 시뮬레이션은 RTL 시뮬레이션 보다 약 100배이상 빠른 속도가 가능하므로 OS 포팅 후, 이더넷 동작에 대한 시뮬레이션이 가능하다. 만약 OS 포팅과 개별 블록 시뮬레이션이 가능하다면, 칩 동작에서는 확인할 수 없는 설계 아키텍처 내부 탐색을 할 수 있으며, 칩 성능에 영향을 주는 인자를 찾아내는 것이 가능하다. 본 연구에서는 완성된 SystmeC 플랫폼으로 설계 아키텍처 내부의 CPU profiling, Bus utilization등 다양한 특성들을 분석해 보고, 성능에 영향을 주는 인자를 찾아 고성능 칩을 위한 제안을 하려고 한다.

II. 모 델

그림 2는 기존 연구에 의해서 개발된 이더넷 내장 SoC 칩의 블록도이다. AHB 인터페이스의 ARM940T 프로세서, DMA를 이용하여 데이터를 전송하는 2개의 이더넷, External Bus Master에 의해서 SDRAM 과 Flash/ROM/SRAM 등이 중재되는 메모리 컨트롤러와 타이머 등을 비롯한 peripheral 블록들로 구성되어 있다.

그러나 본 연구의 목적은 이더넷 통신의 성능을 분석이므로, 그림 2의 블록도를 단순화시켜서 모델링하는 것이 효율적이다. 그림 3은 실제 이더넷 성능에 필요한 IP만으로 구성된 블록도이다. 네트워크 통신성능에 대한 분석이 목적이므로, 데이터 전송의 척도가 되는 NAT(Network Address Translation) 성능에 직접적으로 관련되어지지 않은 주변회로 모듈은 SystemC 모델링을 하지 않았다. 여기서, NAT 성능은 이더넷을 통하여 64, 256, 1514 바이트 등의 용량을 갖는 패킷을 외부

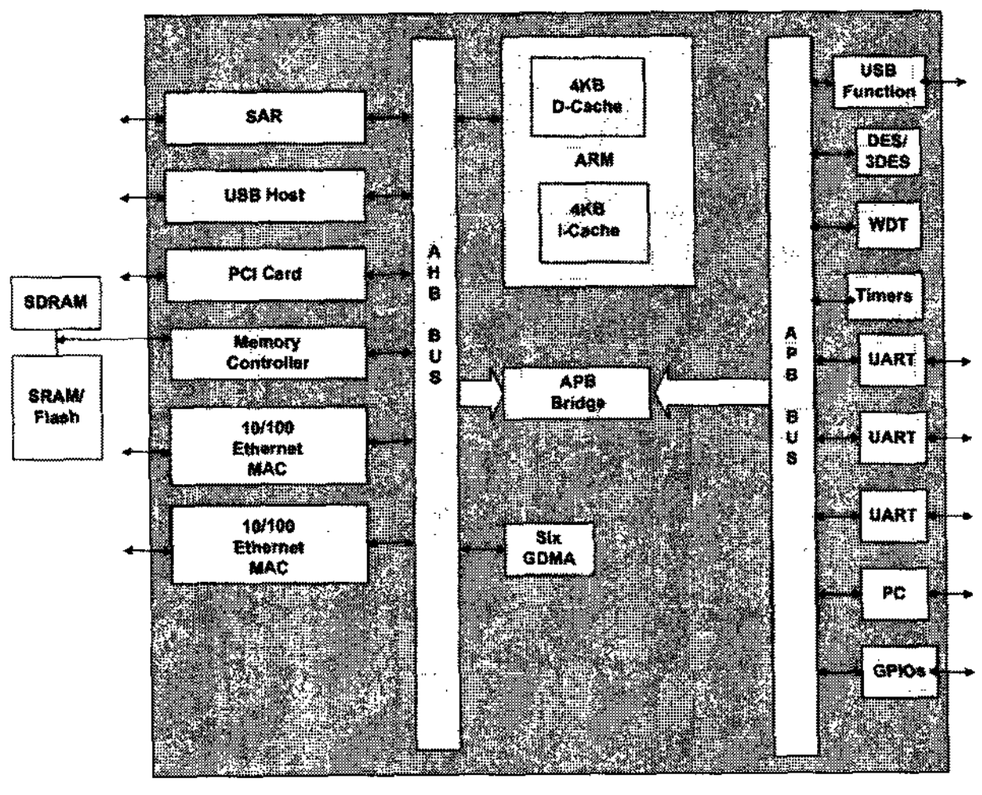

그림 2. 기존에 개발된 이더넷 임베디드 칩의 블록도

Fig. 2. Block diagram of ethernet embedded chip previously developed.

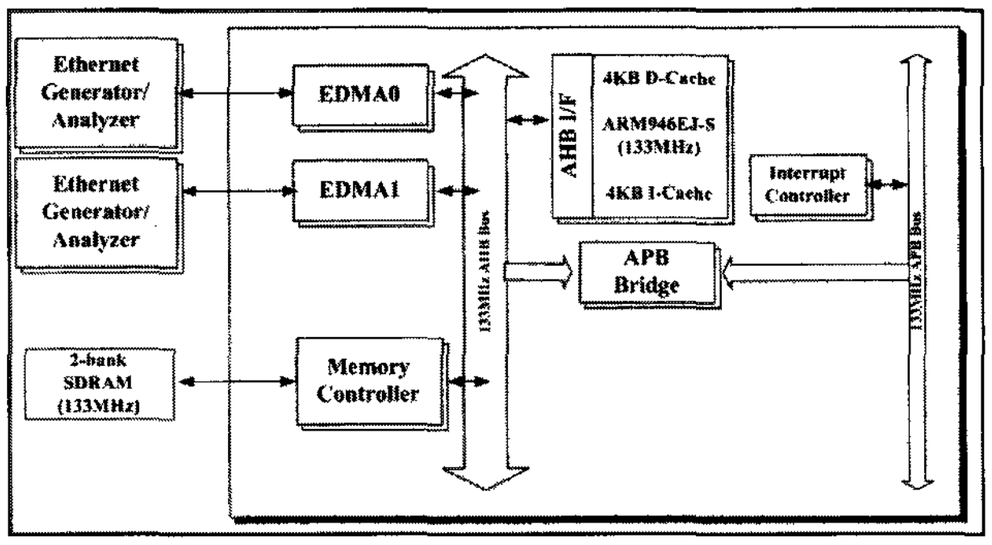

그림 3. NAT 성능측정을 위해 간략화한 이더넷 임베디드 칩의 블록도

Fig. 3. Block diagram of simplified ethernet embedded chip for measurement of NAT performance.

로 전송하고, 전송받은 패킷을 이더넷을 통하여 다시 받아서, 메모리에 저장하는데 걸리는 속도를 측정하여 계산하게 된다. 본 모델링은 ARM 프로세서와 그와 관련된 AMBA 동작 블록, 이더넷 두 채널, 메모리 컨트롤러, 인터럽트 컨트롤러, 타이머 등으로 구성했으며, 이더넷 성능을 파악하기 위해 이더넷 generator, analyzer 등을 추가하였다.

위와 같이 구성된 Soc 플랫폼을 Synopsys의 Cocentric System Studio로 구현하기 위해서, ARM 프로세서와 AMBA bus는 Synopsys DesignWare SystemC 라이브러리를 이용하였으며, 2채널 이더넷 DMA, SDRAM 컨트롤러, Arbiter, 인터럽트 컨트롤러, GDMA 와 타이머를 실제 RTL 코드와 동일한 Transaction Level의 IP들을 SystmeC를 이용하여 설계하였다.

Ⅲ. 결과 및 토의

3-1. bridge test

SystemC 개발환경에서 만들어진 IP들이 실제 RTL

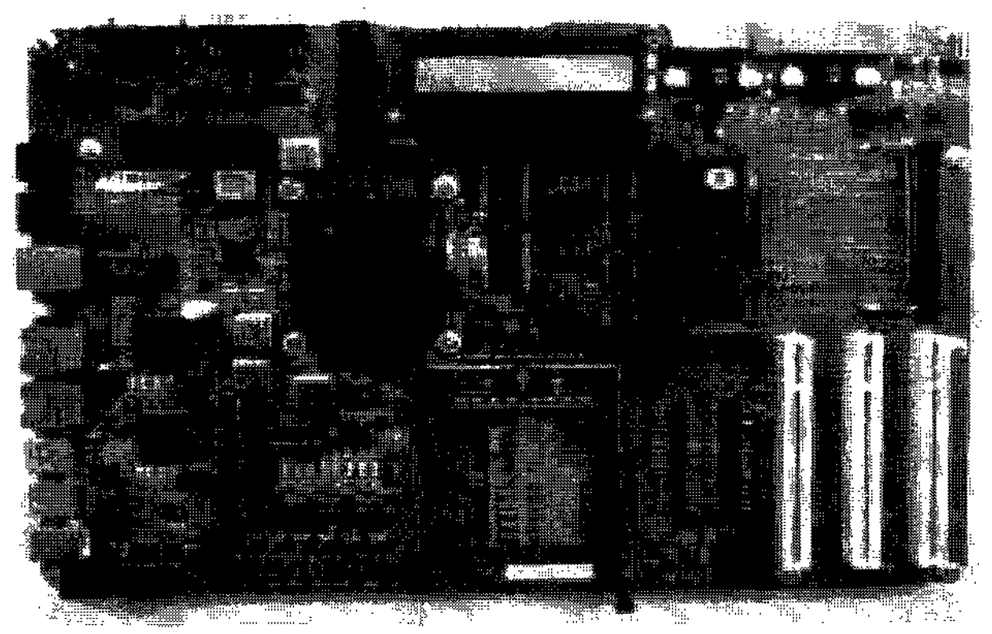

그림 5. 이더넷 임베디드칩 평가를 위한 응용보드

Fig. 5. Application board For evaluation of ethernet embedded chip.

또는 실제 칩과 동일한지 평가하기 위해서, 각 IP별 검증벡터 수행 결과 비교, 브릿지 테스트를 통한 성능비교, RTOS 장착 후 성능비교 등의 절차를 거쳤다.

첫 번째로 각 IP들은 RTL 모델을 참고하여 SystmeC로 개발하였으며, 검증 vector를 이용한 IP별 성능검증으로 실제 RTL 모델과 1% 이하의 성능 정확도를 확인하였다. 그리고 SystemC 플랫폼으로 시스템을 완전히 집적화한 후, 각 모델의 신호파형과 RTL의 파형 결과와 비교하여 모든 모델 cycle이 정확히 RTL과 같은 파형을 나타낸다는 것을 확인했다. 그림 4에는 Modelsim 시뮬레이터를 이용한 SystemC 플랫폼의 이더넷 DMA 파형이 나타나 있다. SDRAM 데이터를 읽는 DMA 레이턴시를 RTL과 비교해 보았을 때, 정확하게 일치함을 확인하였다.

그러나 IP별 성능검증은 일반적인 동작에 한해서 수행된 것이고, 특별한 상황에 의한 corner case 검증이 100% 이루어졌다고 확실할 수 없다. 그래서 본 연구에서는 실제 칩에 대하여 그림 5의 응용보드를 이용하여 이더넷 통신 성능을 측정한 후, 그 동일한 응용코드를

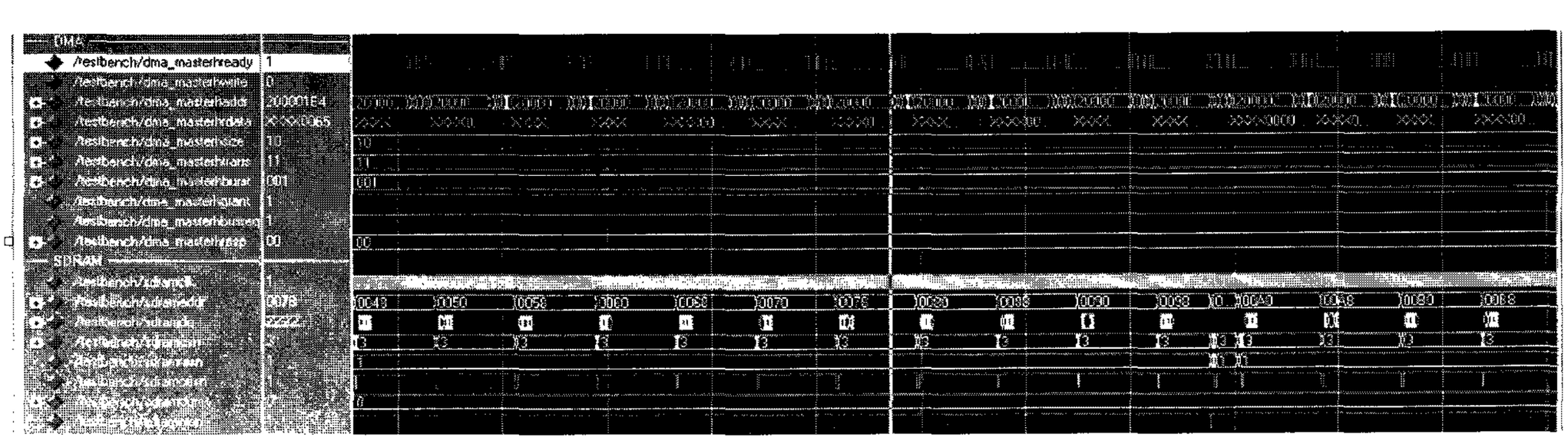

그림 4. SystemC를 이용한 이더넷 DMA 파형

Fig. 4. Waveform of ethernet DMA using SystemC.

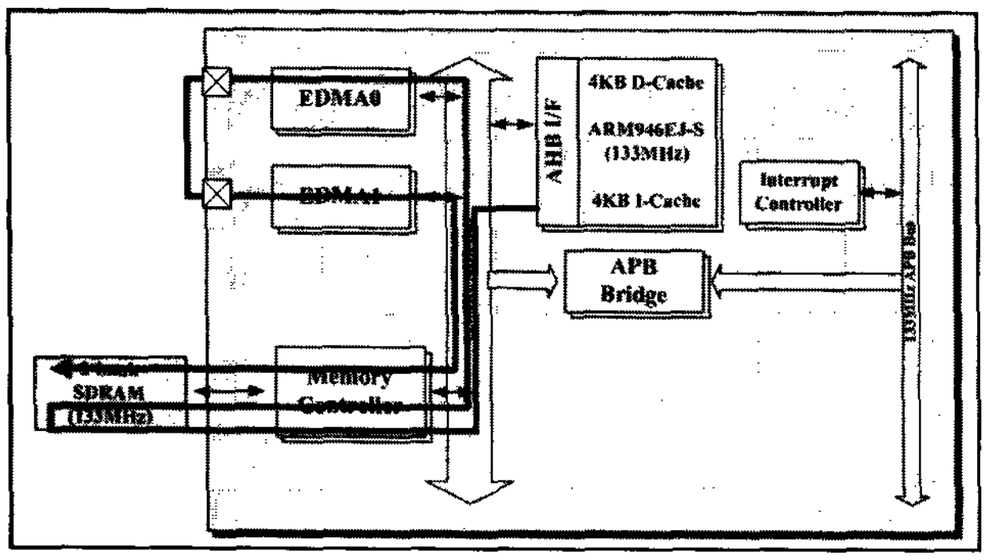

그림 6. Bridge 테스트를 보여주는 블록도
Fig. 6. Block diagram showing bridge test.

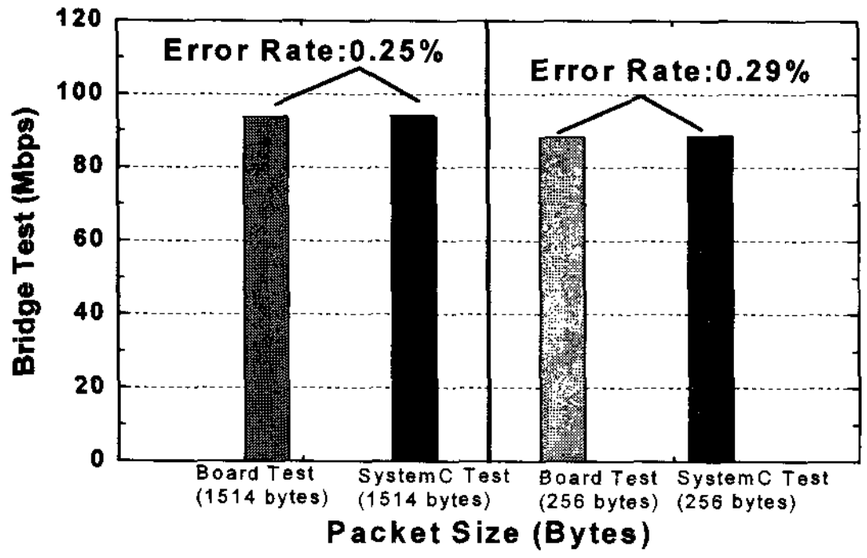

그림 7. 테스트보드의 bridge 테스트 측정결과와 SystemC 시뮬레이션 bridge 테스트 결과
Fig. 7. Bridge test results of test board measurement. and SystemC simulation.

SystemC 플랫폼로 시뮬레이션 해 보았다.

그림 5의 응용보드를 이용하여 칩의 성능을 측정하는 검증을 bridge 테스트라고 하는데, 이 방법은 그림 6에서 보듯이 EDMA0으로 TX 데이터를 전송하고 EDMA1로 RX데이터를 받는 테스트 방법으로, 데이터가 적절하게 라우팅되는지 확인하고, 시스템의 라우팅 속도를 측정하기 위해 일차적으로 수행한다. 그림 6에 굵은 선으로 표시된 것이 데이터 흐름이다.

그림 7의 결과에서 사용된 bridge 테스트는 Real Time OS가 장착되기 전에 TX, RX 데이터 전송이 올바르게 이루어지는지 확인하는 방법으로, 패킷사이즈는 일반적으로 1514, 256 바이트를 사용한다.

응용보드를 통한 실제칩의 브릿지 테스트 성능은 동작주파수 133MHz에서 1514, 256 바이트 패킷사이즈에서 약 90Mbps이었다. 그림 6의 브릿지 테스트 결과로부터 1%이하의 정확도로 응용보드와 SystemC 플랫폼의 성능이 일치함을 확인할 수 있었다. 뿐만 아니라 SystemC 플랫폼을 이용한 브릿지 테스트 벡터 시뮬레이션으로 이더넷 패킷 전송이 ARM9 프로세서에 의해서 적절하게 이루어짐을 확인할 수 있었다.

3-2. Real Time OS 장착

3-1절에서는 SystemC 플랫폼에서 Read Time OS를 장착하지 않은 상태에서 이더넷과 인터럽트 등을 셋팅한 후, 데어터 패킷을 전송해 보고, 실제 전송과의 정확도를 파악해 보았다. 그러나 실제 실장에서 이더넷통신을 할 경우에는 Read Time OS를 장착하고, 이더넷 통신을 수행한다.

본 연구에서는 Real Time OS로 경제성과 사용의 편의성으로 임베디드시스템에 광범위하게 사용되는 리눅스를 사용하였다.

리눅스를 장착하기 위해서 우선적으로 ARM 프로세서 사양을 설정해야 한다.[6] 실제 칩에서의 ARM 프로세서 사양과 동일하게 하기 위해서, ARM926E-S 모델의 TCM(Tightly Coupled Memory)를 OFF시키고, Instruction 캐쉬, Data 캐쉬를 모두 4K 바이트로 설정하였다.[7] 다음으로 기존에 응용보드에서 사용하는 부트코드를 SystemC 플랫폼에 맞게 수정작업을 진행하였다. 이더넷 성능측정과 관련없는 블록에 관한 부트 코드를 모두 제거하였으며, 이더넷 기능과 관련한 부트 코드는 그대로 유지시켰다. 최종적으로 SystemC 플랫폼에 리눅스 OS를 장착시켰으며 소요시간은 약 50분 정도이었다. 실제 응용보드에서 리눅스 장착해서 부팅하는데 약 1~2초가 소요된 것에 비하면, 상당히 느리지만 표 1에 의한 논리적인 계산값과 일치하는 결과이다.

표 1에서 SystemC 플랫폼과 RTL 시뮬레이션의 경우, 메모리의 3G 바이트의 900MHz CPU 사양의 Sun Blade 2000 워크스테이션에 의한 속도측정 결과이다.[8] 표 1에서 보듯이 SystemC 플랫폼에서의 수행속도가 실제 칩에 비해서 약 7000배 정도 느리다는 사실을 알 수 있다.

그러나 SystemC 플랫폼을 Verilog RTL 시뮬레이션과 비교해 보았을 때, 약 160배 이상 빠른 속도를 나타낸다. 표 1을 근거로 하여 Verilog RTL 시뮬레이션 환

표 1. 실제 칩, RTL 그리고 SystemC 플랫폼에서의 동작속도 비교
Table 1. Operation speed comparision of real chip, RTL and SystemC platform.

Type of speed Measurement	Operation speed [cycle/sec]
Real Chip	133M
SystemC platform	18K
Verilog RTL simulation	112

경에서 리눅스 OS를 장착할 경우, 약 130시간 정도 소요될 것으로 파악된다. 리눅스 OS 부팅코드 같은 복잡한 코드를 약 130시간 동안 시뮬레이션을 진행하는 것은 거의 불가능하며, 만약 시뮬레이션이 수행된다하더라도 응용부트코드와 시뮬레이션과의 불일치, 워크스테이션 용량부족 등의 원인에 의해서 시뮬레이션이 hang 될 수 밖에 없다. 실제로 RTL 시뮬레이션 환경에 리눅스를 장착시켜려는 시도를 수차례 해보았으나, 알 수 없는 원인으로 시뮬레이션이 hang되었다.

3-3. 이더넷 성능측정

리눅스 OS가 장착된 상태에서 이더넷의 성능을 평가해 보았다. 응용실장에서 이더넷 성능 측정은 NAT(Network Address Translation) 방법을 사용하는데, 256, 1514 바이트의 TX 데이터 패킷을 이더넷 한단으로 전송한 후, 다른 단의 이더넷으로 전송받아서 속도를 계산하는 방법이다.

그림 8에서 보듯이 응용보드의 측정결과에 비해 SystemC 플랫폼에서 NAT성능이 약 5~6% 정도 큰 것으로 나타났다. 그리고 bridge 테스트의 성능이 256, 1514 바이트 패킷사이즈에서 약 90Mbps인데 반하여, NAT성능은 1514바이트 패킷사이즈에서 약 60Mbps 이고, 256 바이트 패킷사이즈에서 10Mbps로 나타났다.

Bridge 테스트와 같은 OS가 없는 상태에서 이더넷을 통한 데이터 전송에 비해, 실제 OS level의 이더넷 통신은 훨씬 복잡하다는 사실을 알 수 있으며, 이더넷 블록 외에 ARM 프로세서에서 사용되는 시간이 길다는 사실을 추론할 수 있다.

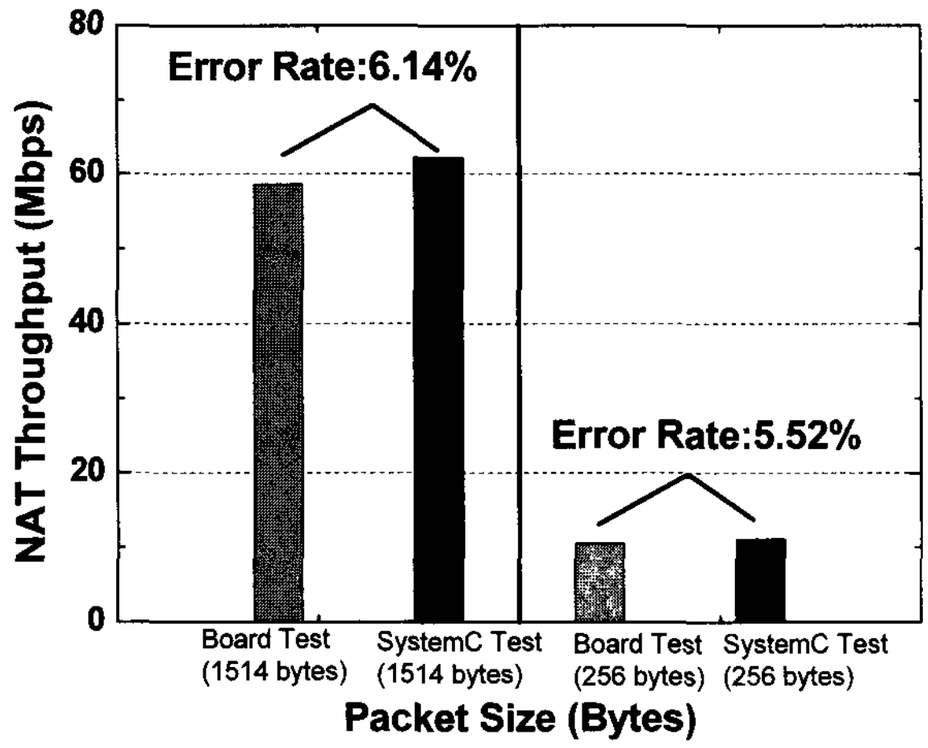

그림 8. 응용보드와 SystemC 플랫폼에 대한 NAT 성능분석 비교

Fig. 8. NAT throughput comparison between application board and SystemC platform.

1514 바이트 패킷 전송이 256 바이트 패킷 전송에 비해 성능이 약 6배 정도 우수한 반면에, 256 바이트 패킷 전송이 1514 바이트 패킷 전송에 비해 패킷사이즈가 약 6배 작다. 정성적으로 보았을 때, 패킷을 보내거나 받기 위해 필요한 ARM 프로세서 동작이 1514 바이트 패킷 전송에 비하여 256 바이드 패킷 전송이 약 6배 정도 더 많이 필요하게 된다. 이의 영향에 의해 NAT성능이 직접적으로 영향을 받는다는 사실을 쉽게 추론할 수 있다. 리눅스 OS가 장착되어 ARM 프로세서의 작업량이 크게 증가된 형태의 NAT 성능측정에서 응용보드의 측정치와 SystemC의 시뮬레이션 결과가 약 5~6%정도의 차이를 나타내는 것은 ARM 프로세서 모델상의 차이와 버스 아키텍처의 불일치에 의한 것으로 생각된다. 50분이상의 장시간에서 얻어진 시뮬레이션 결과를 측정치와 비교한 결과이므로, SystemC 플랫폼 모델은 실제 칩에 대한 충분한 신뢰성을 갖는다.

3-4 이더넷 성능분석

(1) I/D Cache 사이즈

bridge테스트 결과와 달리, 리눅스가 장착된 NAT 성능측정결과에서 전송속도는 1514바이트 패킷사이즈에서 약 60Mbps정도 이었다. 만약 리눅스 장착에 따른 성능감소의 원인을 구조적으로 찾아낸다면 bridge테스트 결과인 90Mbps까지 성능을 향상시킬 수 있다.

이더넷 마스터에 의한 데이터 전송방법이 두 경우 모두 동일하므로, 성능감소원인으로 1차적으로 생각할 수 있는 부분은 바로 ARM 프로세서의 역할이다. 리눅스 OS가 장착된 경우 ARM 프로세서에서 처리해야 할 업무량이 많아지게 되는데, ARM 프로세서의 성능은 ARM 프로세서에 내장된 Instruction/Data Cache에 크게 좌우된다. 그래서 우선적으로 I/D Cache 크기에 다른 성능변화를 파악해 보았다.

AMBA 버스 및 ARM922 프로세서에 대하여 Synopsys DesignWare SystemC 라이브러리를 이용하면, 시뮬레이션 완료 후, CPU profiling 기능을 통하여 Cashe 사용분포, Bus contention 등을 확인할 수 있다. 표 2에는 I/D Cache 크기에 따른 NAT 성능이 나타나 있는데, I/D Cache 크기가 8K바이트 이상일 때, 약 100Mbps의 NAT 성능을 보여준다. 결국, Cache 크기를 키운다면, 이더넷 통신에서 원하는 성능을 쉽게 확보할 수 있게 된다.

다음으로 Cache 크기가 NAT성능에 영향을 미치는

표 2. I/D cache 크기와 NAT성능의 비교
Table 2. Comparison of I/D cache size vs. NAT performance.

ARM946E-S BUS/CPU Frequency: 133/133MHz			
I/D Cache size	4K/4K	8K/8K	16K/16K
FrameSize(Bytes)	1514	1514	1514
NAT (Mbps)	62	99	99

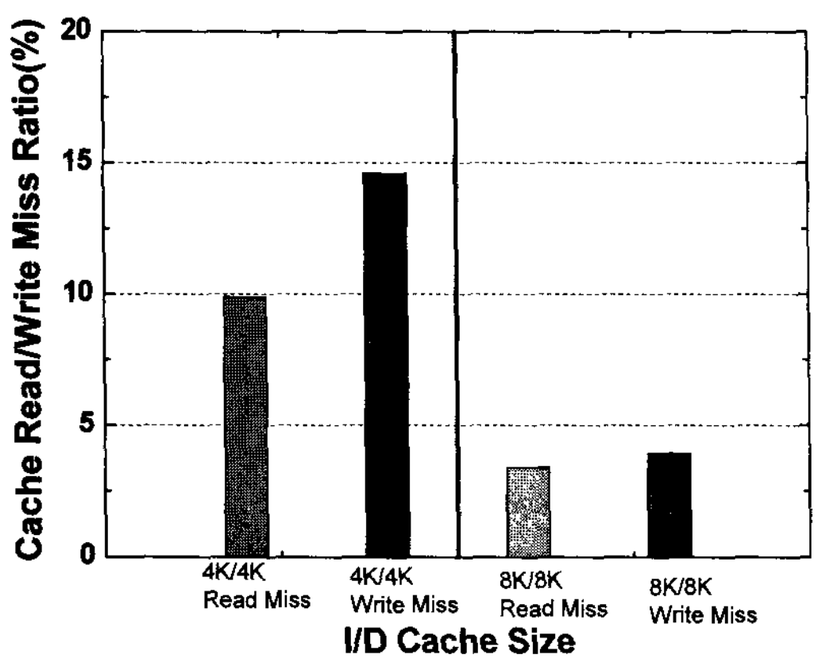

그림 9. Cache Miss 비율의 프로파일
Fig. 9. Cache Miss Ratio Profiling.

원인에 대해, 설계 내부를 탐색할 수 있는 SystemC 플랫폼을 이용하여 보다 구체적으로 파악해 보았다. 그림 9에는 I/D Cache 크기가 4K/4K, 8K/8K일 경우의 프로파일이 나타나 있다. I/D Cache 크기가 4K/4K에서 Cache read miss 비율과 write miss비율이 각각 약 10, 14%인데 반해서, I/D Cache 크기가 8K/8K에서 Cache read miss 비율과 write miss비율이 모두 4%이하로 줄어들었다. 결국 기존의 칩에 리눅스 OS를 장착할 경우, Cache miss 비율이 크게 증가하게 되어 성능이 떨어진다는 사실을 알 수 있다.

데이터 cache read misses ratio의 영향을 정량적으로 파악하기 위해 메모리 엑세스 사이클을 계산해 보았다.

$$C_{access} = \lambda \cdot C_{ExtMem} + (1-\lambda) \cdot C_{Cache} \qquad (1)$$

식(1)에서 C_{access}는 ARM 프로세서의 데이터 쓰기 또는 읽기에 필요한 사이클 수, λ는 Cache miss비율, C_{ExtMem}은 외부메모리 엑세스 사이클 수, C_{Cache}는 Cache 엑세스 사이클 수이다.

ARM 프로세서가 SDRAM에서 데이터를 읽어올 경우 8 word를 버스트로 읽어오게 되며, 소요되는 사이클 수는 13이며, Cache에서 데이터를 읽거나, 쓸 경우 1사이클이 소요된다. Cache miss 비율이 10%일 때, ARM 프로세서의 데이터 쓰기 또는 읽기에 필요한 사이클 수는 2.2이며, Cache miss 비율이 5%로 감소될 경우의 사이클 수는 1.6이다. 결국 ARM 프로세서에 내장된 Chache의 사이즈를 4K바이트에서 8K바이트로 증가시켰을 경우, 평균적으로 약 30%정도의 성능향상을 가져올 수 있다.

(2) ARM 프로세서 Frequency

ARM 프로세서의 성능은 Cache 특성 외에도 주파수에도 민감하게 변화할 수 있다. 그래서 AMBA 버스의 주파수는 133MHz으로 유지시키고, ARM 프로세서의 주파수를 변화해 보았다. 그림 10에 보듯이 Cache의 크기가 8K 바이트일 경우, 주파수의 증가함에 관계없이 일정한 NAT 성능을 보여 준다. 이는 주파수 133MHz에서 이미 최대의 성능을 나타낸다는 것을 말해 준다. Cache 크기가 4K 바이트인 경우, ARM 프로세서의 주파수가 200MHz일 경우에도 이더넷의 NAT 성능은 크게 증가하지 않았다. ARM 프로세서가 266MHz일 때, 비로서 최대 성능인 약 100Mbps를 나타내었다. 실제 현재 목표로 하는 ARM9 프로세서의 최대주파수는

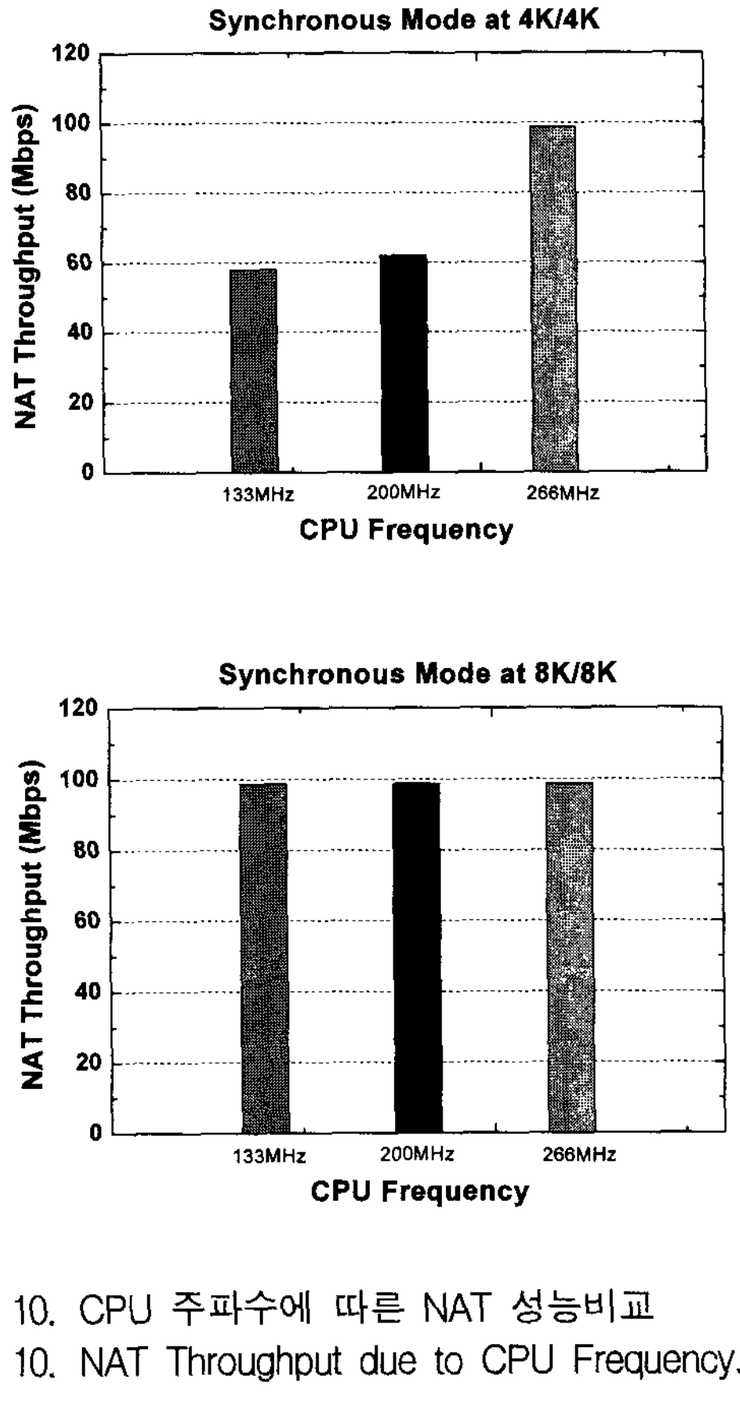

그림 10. CPU 주파수에 따른 NAT 성능비교
Fig. 10. NAT Throughput due to CPU Frequency.

166MHz이므로, 주파수를 증대시켜서 성능을 향상시키는 것은 크게 한계가 있다.

(3) Bus Contention

ARM 프로세서 외에 영향을 줄 수 있는 부분은 버스 아키텍처 부분이다. 그래서 I/D cache 크기를 8K/8K으로 설정하고 데이터를 전송하고 받는 동안에 아무런 packet 손실이 없는 85Mbps의 비율로 프레임을 전송하였다. 그림 11의 "New Master Added Case"는 AMBA 버스에 많은 데이터가 처리될 때의 변화를 파악해 보기 위해 임의로 DMA 마스터를 추가시킨 경우이다. 그런데, 그림 11에 보듯이 새로운 마스터를 추가하여 AMBA 버스에 overload를 가하는 경우에도 CPU, 이더넷에 의한 버스점유율은 크게 변화하지 않았으며, 단지 Idle 부분 중 일부가 새 마스터 점유에 사용되었다.

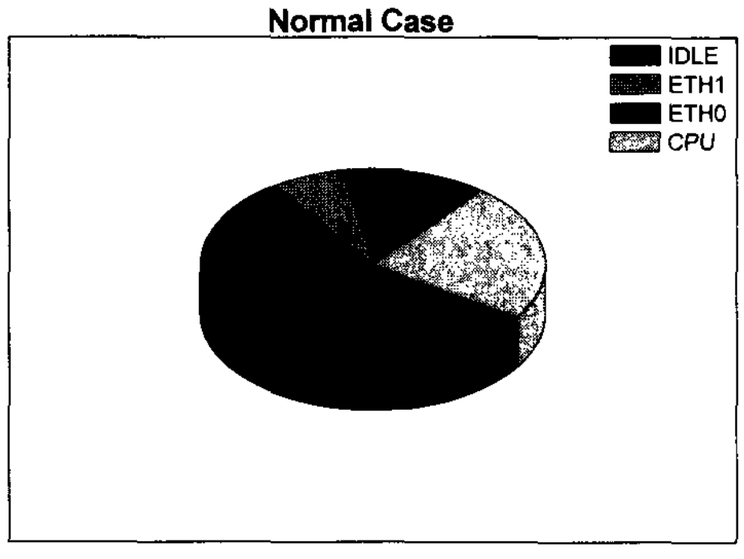

그림 11. Traffic loader를 통한 버스 contention
Fig. 11. Bus Contention by traffic loaders.

뿐만 아니라 새 마스터의 추가에도 NAT 처리량은 크게 영향을 미치지 않는다. 결국, 버스 아키텍처 구조에 의한 NAT 성능의 영향은 크지 않음을 추측할 수 있다.

Ⅳ. 결 론

본 논문에서는 SystemC 플랫폼 하에서 리눅스 OS 장착을 시킨 후, 시스템의 성능을 평가해 보았다. 그리고 시스템 성능에 영향을 주는 인자가 Cache miss 비율임을 확인하고, Cache 사이즈를 4K 바이트에서 8K 바이트로 증대시키는 것을 제안하였다. AMBA 버스 시스템 하에서, 본 SystemC 플랫폼은 칩을 설계하기 전에 성능분석용으로 유용하게 사용될 수 있으리라 생각된다.

참 고 문 헌

[1] Kanishka Lahiri, Anand Raghunathan, and Sujit Dey, "Design Space Exploration for Optimizing On-Chip Communication Architectures", IEEE Trans. Computer-Aided Design, vol. 23, pp.952.961, June. 2004.
[2] K. Lahiri, A. Raghunathan, G. Lakshminarayana, and S. Dey, "Communication architecture tuners: A methodology for the design of high performance communication architectures for system-on-chips," in Proc. Design Automation Conf., June 2000, pp. 513.518.
[3] CoCentric System Studio Data Sheet, Synopsys http://www.synopsys.com
[4] Preeti Ranjan Panda: "SystemC . A modeling platform supporting multiple design abstractions", Synopsys Inc, http://www.synopsys.com.
[5] http://www.cadence.com/verisity/
[6] http://www.uclinux.org
[7] http://www.arm.com
[8] http://www.sun.com/desktop/FinalSB2000ds.pdf

저 자 소 개

이 국 표(학생회원)

1999년 인하대학교 전자재료 공학과 학사 졸업.
2001년 인하대학교 전자재료 공학과 석사 졸업.
2005년~현재 인하대학교 전자 공학과 박사과정 (박사수료)

<주관심분야 : 반도체, SoC, 디지털, 아날로그 회로설계, FPGA설계, 아키텍처 성능분석>

윤 영 섭(정회원)

1975년 서울대학교 금속공학과 학사 졸업.
1977년 한국과학원 재료공학과 석사 졸업.
1988년 Univ. Southern California 전자공학과 박사 졸업.
1987년~1988년 Oklahoma State University 대우교수
1988년~1989년 UCLA Device Research Lab. 연구원
1989년~1992년 삼성전자 기흥반도체연구소 수석연구원
1992년~현재 인하대학교 전자공학과 교수

<주관심분야 : ULSI DRAM 을 위한 신물질 개발, 강유전성 박막, Pyroelectric 센서, SAW device, 회로설계>