논문 2008-45SD-10-8

네트워크 패킷 처리를 위한 효율적인 비트 스트림 명령어 세트

(An Efficient Bit Stream Instruction-set for Network Packet Processing Applications)

윤 여 필*, 이 용 석*, 이 정 희**

(Yeo Phil Yoon, Yong Surk Lee, and Jung Hee Lee)

요 약

본 논문은 네트워크 프로세서의 패킷 처리 능력 향상을 위한 새로운 명령어 세트를 제한한다. 제안하는 명령어는 패킷 헤더의 결합 연산을 가속화 할 수 있으므로 보다 효율적인 패킷 처리를 수행할 수 있다. 또한 overlay 명령어 처리를 위한 전용 하드웨어 구조를 설계하여 추가 하드웨어로 인한 비용을 최소화 하였다. 이를 위해 LISA 언어를 이용하여 네트워크 프로세서 기본 아키텍처를 설계하고 overlay 블록을 배럴 시프터를 기반으로 최적화 하였다. 이를 합성하여 면적 및 동작 지연시간을 비교하였으며, 컴파일러의 CKF(Compiler Known Function)를 이용하여 C레벨의 매크로 함수에 할당하고 어플리케이션 프로그램에 대한 실행 사이클 및 실행 시간을 비교하여 성능 향상을 확인하였다. Coware사의 processor designer, compiler designer를 이용하여 실험하였으며 Synopsys의 TSMC 0.25um로 합성한 결과 20.7%의 동작 지연시간 감소를 보였고, 전체 실행 사이클에선 제안하는 명령어 세트에 의해 30.8%의 성능 향상을 보였다.

Abstract

This paper proposes a new set of instructions to improve the packet processing capacity of a network processor. The proposed set of instructions is able to achieve more efficient packet processing by accelerating integration of packet headers. Furthermore, a hardware configuration dedicated to processing overlay instructions was designed to reduce additional hardware cost. For this purpose, the basic architecture for the network processor was designed using LISA and the overlay block was optimized based on the barrel shifter. The block was synthesized to compare the area and the operation delay, and allocated to a C-level macro function using the compiler known function (CKF). The improvement in performance was confirmed by comparing the execution cycle and the execution time of an application program. Experiments were conducted using the processor designer and the compiler designer from Coware. The result of synthesis with the TSMC (0.25 μm) from Synopsys indicated a reduction in operation delay by 20.7% and an improvement in performance of 30.8% with the proposed set of instructions for the entire execution cycle.

Keywords : Network, Instruction set, Bit Stream Processing, Accelerator, CKF

I. 서 론

일반적으로 임베디드 시스템 환경을 위해서 범용 프로세서나 전용 ASIC(Application Specific Integration Circuit)이 사용되어 지나 범용 프로세서는 시스템만의 특성화된 성능을 맞출 수 없고 ASIC의 경우에는 유연성이 떨어져 새로운 응용분야를 추가할 수 없다. 따라서 특정 시스템을 위한 프로세서 개발은 효율적인 ASIC 과 범용 프로세서의 저비용, 적은 개발노력 사이에서 사양을 결정할 필요가 있다. 임베디드 시스템과 같은 제한된 응용분야를 위한 특화된 프로세서 개발은

* 정회원, 연세대학교 전기전자공학과
(Electrical and Electronic Engineering, Yonsei University)
** 정회원, 한국전자통신연구원
(Electronics and Telecommunications Research Institute)
※ This work was supported by ETRI from the Research Program of multimedia convergence network on chip technology. EDA tools which were used in this work were supported by the IC Design Education Center (IDEC).
접수일자: 2008년5월19일, 수정완료일: 2008년10월7일

많은 연구가 진행되어 왔으며 시스템을 가속하기 위한 특별한 명령어 세트를 추가한 ASIP(Application Specific Instruction set Processor)는 매우 일반적이다. 이러한 ASIP은 RISC형태의 코어를 가지고 시스템만의 전용 명령어 세트를 가지고 있어 소자 간의 낮은 지연 시간과 고속의 동작속도 그리고 코드 사이즈를 감소시켜 저 전력의 장점을 가지고 있다.

네트워크 환경은 대표적인 임베디드 시스템으로서 전용 프로세서 개발이 필수적인 분야이다. 네트워크 통신의 물리적인 속도는 계속 빨라지고 있으나 IPTV, VOIP, VOD 등과 같은 멀티미디어에 의한 패킷 양은 더욱더 증가하면서 빠르고 저 전력으로 패킷을 처리할 수 있는 프로세서가 요구되어 진다. 이러한 ASIP을 네트워크 프로세서라 한다.

본 논문에서는 이와 같은 네트워크 환경을 가속하기 위해 네트워크 연산을 분석하고 이를 가속할 수 있는 확장 명령어 세트를 제안하고자 한다. 또한 명령어 세트를 위한 하드웨어 가속기 모듈을 설계하고 CKF를 이용하여 전용 컴파일러를 생성하여 C 코드의 어플리케이션에 대한 네트워크 프로세서의 성능을 비교하고자 한다.

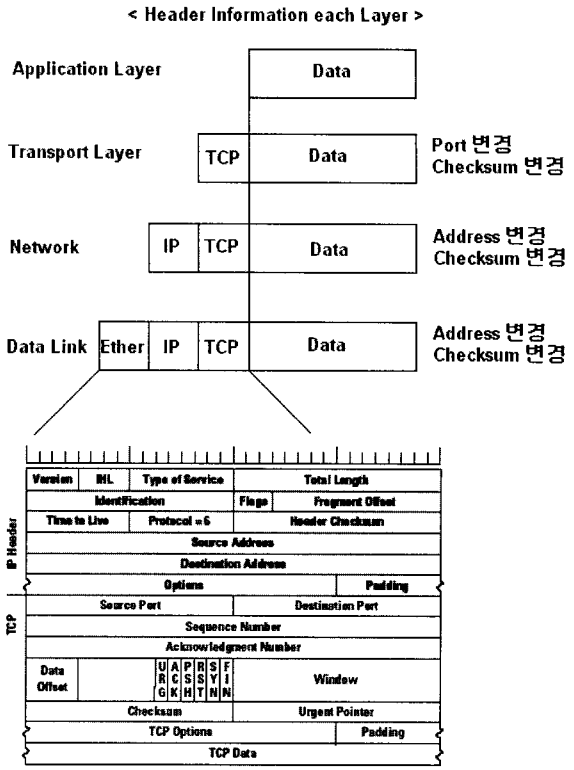

그림 1. 네트워크 패킷 구조
Fig. 1. Structure of the packet structure.

Ⅱ. 네트워크 연산

네트워크 환경에서의 데이터는 <그림 1>의 패킷 구조에서 알 수 있듯이 데이터를 각 레이어별 헤더 정보가 감싸고 있고 네트워크 프로세서는 레이어별 헤더를 분리하여 정보를 취득하고 새로운 헤더를 추가하여 전송한다. 여기에서 통신을 위한 패킷은 워드단위로 정렬되어 전송되어지나 수신 단에서 헤더를 분류하고 저장하는 단계에서 서로 다른 길이의 정보로 나누어지게 된다. 이를 다음 라우터로 전송하기 위하여 일부 정보를 수정하여 패킷 구조로 재구성해야 하는데 이러한 과정에서 비트 스트림 연산을 반복 수행하게 된다.

네트워크 시스템에서 최대 문제는 고속 통신을 위해 각각의 패킷을 비트 스트림 연산하는데 매우 짧은 시간이 주어진다는 것이다. 때문에 네트워크 프로세서의 명령어 세트는 효율적인 통신 프로토콜 연산에 맞추어져야 하며, 이를 위해 다양한 크기의 비트 스트림을 바로 처리할 수 있도록 디자인되어야 한다. 그러나 일반 프로세서에서는 비트 스트림 연산을 위하여 마스크와 시프트 연산을 반복 수행하기 때문에 발전하는 네트워크 환경에 대응할 수 없다.

Ⅲ. 제안하는 구조

네트워크 프로세서의 주된 연산은 패킷을 디코딩하고 압축, 암호화, 발송하면서 수반되는 패턴 매칭, lookup 테이블 검색, checksum 계산 그리고 IPX와 UDP 패킷을 위한 endian 변경 등이다. 특히, 이들 연산의 대부분은 패킷을 분류하고 해석하여 정보를 읽어내고 재조립하여 송신하는 단계에서 포트나 주소의 변경, 플래그변경, checksum 수정 등과 같이 비트 스트림 연산을 반복 수행하게 되는데 이들은 시프트와 마스크 그리고 패킹(or) 등의 연산으로 구성되어 진다.

네트워크 프로세서는 <그림 2>와 같이 새로운 패킷

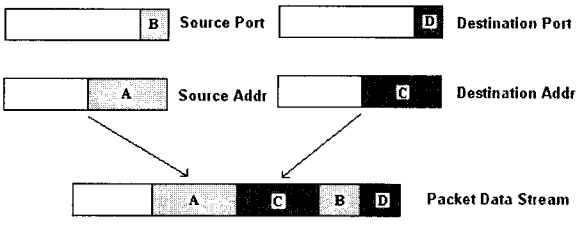

그림 2. 네트워크 비트 스트림 연산
Fig. 2. Bit stream operation for the network.

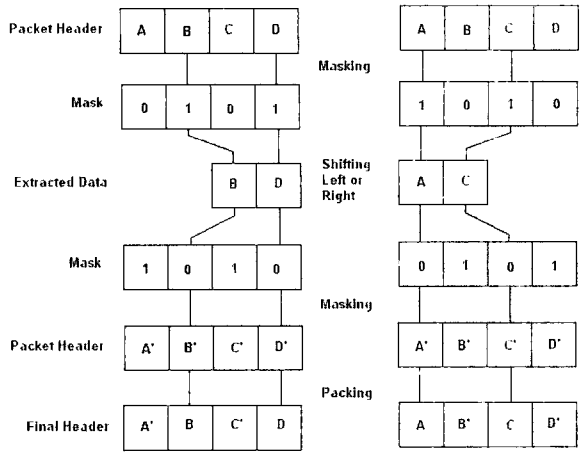

그림 3. 네트워크 비트 스트림 연산
Fig. 3. Bit stream operation for the network.

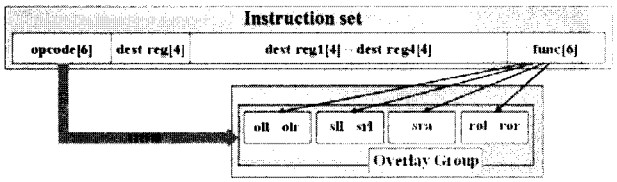

그림 4. 명령어 세트 구조
Fig. 4. Structure of the instruction set.

헤더를 구성하기 위하여 메모리로부터 서로 다른 길이의 정보를 패킷 구조에 맞게 조합하여 데이터 스트림을 구성한다. 네트워크 환경에서의 연산과정을 자세히 살펴보면, <그림 3>과 같이 들어온 패킷 헤더에서 정보를 발췌하기 위한 마스크와 이를 메모리로부터 가져와 패킷구조에 맞게 시프트한 후 마스크 된 새로운 헤더에 패킹해 주게 된다. 다시 말해, 패킷 연산은 두 번의 마스크와 한 번의 시프트 그리고 한 번의 패킹 연산으로 구성된다.

이러한 비트 스트림 연산을 위하여 <그림 4>와 같이 확장 명령어 세트를 제안하고 이를 overlay 그룹으로 정의하였다.

```
overlay : OPcode_rd_rs1_rs2_rs3_rs4_func
    sll :  111110  rd  rs1,2,3,4     000000
    srl :  111110  rd  rs1,2,3,4     000001
    sra :  111110  rd  rs1,2,3,4     000011
    rol :  111110  rd  rs1,2,3,4     000100
    ror :  111110  rd  rs1,2,3,4     000100
    oll :  111110  rd  rs1,2,3,4     001000
    olr :  111110  rd  rs1,2,3,4     001001
%func = 0_0_Overlay_Arith_Rotate_Direct
```

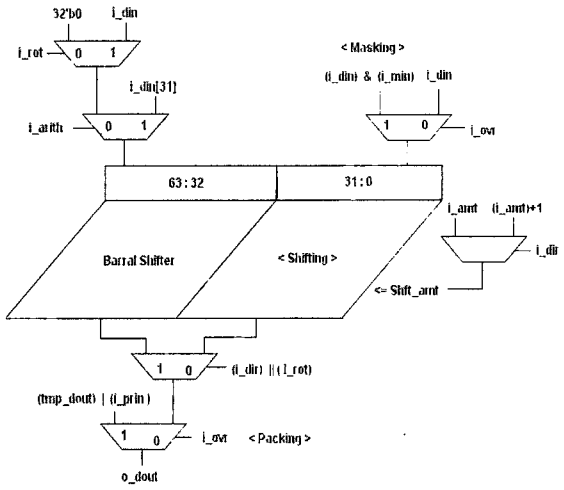

그림 5. Overlay 가속기 구조
Fig. 5. Structure of the accelerator for Overlay.

Overlay 명령어 세트는 논리/산술 시프트, 회전 연산과 비트 스트림 연산을 위한 overaly 연산을 지원한다. 이는 overlay left/right 와 shift left/right, shift arithmetic right, rotate left/right 로 구성되어 있다. Function은 LSB부터 direction, rotate, arithmetic, overlay를 의미하며 4개의 소스레지스터와 1개의 목적레지스터를 갖는다. rs1은 추가해야 할 정보이며, rs2는 시프트 양, rs3는 마스크, rs4는 수정되어야 할 패킷이며, rd는 연산결과가 저장되어야 할 레지스터이다.

이러한 overlay 명령어 세트를 한 사이클에 실행하기 위해 RTL을 이용하여 overlay 모듈을 설계하였다.

<그림 5>와 같이 barrel 시프터를 이용하여 최적화하였으며, 여기에 비트 스트림 연산을 위한 마스크와 패킹을 적용하여 두 개의 패킷으로부터 하나의 데이터 스트림을 구성하도록 설계하였다. 이는 한 모듈 안에 마스크, 시프트와 패킹을 처리함으로서 한 사이클에 비트 스트림 연산이 가능하다.

Ⅳ. CKF (Compiler Known Function)

네트워크 프로세서에서 코드 생성 시 가장 중요한 문제는 ASIP을 위한 효율적인 컴파일러를 사용하는 것이다. 네트워크와 같은 임베디드 시스템에서 전용 명령어 세트는 아주 일반적이다. 이를 위한 ASIP의 컴파일러 지원은 필수적이며, 효율적인 컴파일러는 개발 시간 소비를 줄일 수 있고 실수하기 쉬운 어셈블리 프로그래밍을 줄일 수 있다. 이 논문에선 패킷 연산을 가속하기 위

한 확장 명령어 세트를 가진 네트워크 프로세서를 생성하는 것으로 컴파일러의 intrinsics와 레지스터 할당기, 스케줄러를 이용하여 어셈블리 프로그래밍 없이 C 코드 레벨에서 명령어 세트에 할당할 수 있다.

그러나 ASIP의 특별한 아키텍처 때문에 일반적인 컴파일러로는 불충분하며 보다 특성화된 코드 생성과 최적화를 위해 새로운 컴파일러가 필요하게 된다. 이를 위하여 비트 스트림 연산을 위한 확장 명령어 세트를 CKF (compiler known functions)로 지정하여 컴파일러를 생성하고 intrinsic를 이용하여 쉽게 프로그래밍 할 수 있다. 이 때 컴파일러는 일반적인 함수 호출을 하지 않고 CKF에 호출을 할당 하여 로컬 함수 호출에 따른 오버헤드 없이 C레벨 매크로로서 사용되어 질 수 있다.

위와 같이 설계된 명령어 세트와 가속기를 상위 레벨의 어플리케이션에서 사용하기 위해선 컴파일러의 도움이 필요하다. C레벨에서 비트 스트림 연산이 불가능한 것은 아니나, 비록 가능할 지라도 프로그래머 입장에서 상당히 불편하며, 임베디드 시스템에서 중요시 되는 코드 사이즈가 상당히 커진다. 또한 더 큰 문제는 이들은 각각의 어셈블리로 표현될 수밖에 없어 명령어 수가 증가한다. 따라서 컴파일러에서는 이러한 시프트, 마스크와 패킹을 하나의 특별한 함수로 지정할 수 있다. 이를 CKF 라 한다. 이를 위해선 범용 gcc 컴파일러가 아닌 전용 컴파일러가 필요하게 된다. 대부분의 다른 컴파일러와 같이 네트워크 프로세서 컴파일러는 frontend 와 backend로 나누어진다. Frontend에선 코드를 분석하고 의존성 등을 체크하는 반면, backend에선 코드 셀렉터를 이용하여 어셈블리 코드로 변환한다. 다시 말해, 코드 셀렉터에서 CKF를 찾아 호출하며 어셈블리 명령어에 할당한다. 또한 register allocator에서 레지스터에 연결한다.

Overlay 연산을 위한 비트 스트림 연산 예제를 들자면 아래와 같이 표현 되어 질 수 있다.

```
C = (( A & C ) << B ) | ( D & ~(C <<B) );
C = (( A & C ) >> B ) | ( D & ~(C >>B) );
```

이러한 표현은 표준 컴파일러에 의해 최적화되어 간단해 질 수 있지만 어셈블리 레벨로 번역되어 지면 매우 많은 명령어들로 번역 될 수 있다. 반면 본 논문의 네트워크 프로세서에선 단 하나의 함수로 표현되어 CKF에 의해 확장 명령어 셋에 할당되고 설계된 하드웨어 가속기에 의해 구현되어 질 수 있다.

```
unsigned int overlayl (unsigned int A,
int B, unsigned int C, unsigned int D);
unsigned int overlayr (unsigned int A,
int B, unsigned int C, unsigned int D);
```

A는 rs1에 해당하는 값으로 추가되어야 할 정보이고, B 는 rs2에 해당되는 시프트 되어야 할양이고, C는 rs3에 해당되는 마스크이다. 그리고 D는 rs4에 해당되는 수정되어야 할 패킷이다. 일반 C언어 레벨에서의 많은 명령어 처리를 위한 과부하를 줄이기 위해선 CKF를 이용하여 함수를 선언하고 컴파일러의 backend에서 register allocator에 의해 레지스터에 연결하고 명령어 셋에 할당할 수 있다.

C함수에서 overlayl 은 'oll' 명령어 셋으로 할당되며, 마스킹한 후 왼쪽 시프트 하여 이를 새로운 패킷에 합치게 되며, overlayr은 'olr' 명령어 셋으로 할당되어, 마스킹한 후 오른쪽 시프트 하여 이를 새로운 패킷에 합치는 구조이다.

V. 실 험

ASIP을 기본 설계하기 위하여 LISA 언어를 이용하여 RISC기반의 프로세서를 설계하였다. 이는 5단 파이프라인을 가진 MIPS기반의 기본 프로세서로서 32비트 ALU 2개, 32비트 시프트, 32*32비트 MAC, Load /Store를 위한 메모리, 점프와 같은 Control 블록과 확장 명령어 블록 등 6개의 블록을 가진 프로세서이다.

이는 Coware사의 프로세서 디자이너를 이용하였으며 여기서 설계된 아키텍처를 기본으로 블록별 HDL code를 추출하고, 실행 단의 시프트 블록을 패킷의 비트 스트림 연산을 위한 overlay 블록으로 대치하였다. 이를 최적화하기 위하여 시프트 모듈을 barrel 시프터 모듈로 변경하였으며 비트 스트림 연산을 위한 모듈을 추가하였다.

아래의 표와 같이 기준 프로세서의 시프트 블록과 비트 스트림 연산을 위한 가속기가 추가된 블록의 합성을 통해 면적과 동작 지연시간 및 동작 주파수 확인 결과

표 1. 설계된 ASIP에 대한 합성 결과 비교
Table 1. Synthesis results for proposed ASIP.

	기준 프로세서	네트워크 프로세서
총 면적	167574	77566
동작 지연시간	11.394 ns	9.033 ns
동작 주파수	83.79 MHz	110.70 MHz

는 아래의 표와 같다. 이는 Synopsys의 Design Compiler를 이용하여 TSMC 0.25um로 합성하였다.

위의 결과와 같이 네트워크 환경을 위한 가속기를 이용하여 최적화하기 전과 후를 비교한 결과 동작지연 시간 및 총 면적에서 모두 개선되었다. barrel 시프터를 썼음에도 면적이 개선된 이유는 기준 프로세서에선 시프트, 회전, overlay 각각에 시프트 모듈이 생성되었기 때문으로 barrel 시프터 하나의 모듈을 레지스터를 통해 공유함으로 면적과 주파수 모든 면에서 개선을 얻었다.

앞에서 말한 것처럼 이를 C레벨의 어플리케이션에 적용하기 위하여 컴파일러가 필요하게 된다. 이는 범용 gcc 컴파일러로는 만족시킬 수 없으며 ASIP을 위한 전용 컴파일러가 필요한데 이를 위해 Coware사의 Compiler designer를 이용하여 lcc 컴파일러를 생성하였으며 비트 스트림 연산을 위한 가속기와 overaly 명령어 셋을 연결하기 위해 CKF를 정의한 후 컴파일러에 적용하였다. 이는 프로그래머 입장에서 하나의 C 매크로로 인식되어 하나의 명령어로 네트워크 패킷 연산이 가능하며 성능 면에서는 비트 스트림 연산을 위하여 1 사이클 내에 처리가 가능하게 된다.

아래는 이를 실험하기 위하여 8개의 패킷에 주어진 정보를 추가하기 위하여 비트 스트림 연산을 하는 것으로 CKF를 적용 하였을 때 와 CKF를 적용하지 않고 시프트와 마스트 연산에 의해 구현되었을 때를 C레벨에서 구현하였다. 이를 lcc를 이용하여 컴파일 한 후 디

그림 6. CKF test 어플리케이션 코드
Fig. 6. Application code for the CKF.

표 2. 네트워크 비트 스트림 연산
Table 2. Bit stream operation for the network.

	기준 프로세서	네트워크 프로세서
명령어 사이클 수	4	2
전체 사이클 수	1246	862
동작 시간	14196.92 ns	7872.64 ns

버거에서 전체 사이클 수와 동작 시간을 비교하였다.

위의 어플리케이션에서 알 수 있듯이 CKF 적용시 C 레벨에서의 코드 사이즈 감소뿐만 아니라 어셈블러 레벨에서의 전체 명령어 수를 감소시킬 수 있다. 이는 다시 말해 성능 향상 및 스위칭 동작을 줄려 소비전력 감소의 효과가 있다.

주어진 어플리케이션을 디버거에서 동작시킨 결과 표 2와 같이 비트 스트림 연산을 위해 마스크, 시프트, 마스크, 패킹 등 4 사이클이 소요되던 것이 마스크와 오버레이 2 사이클로 줄었으며, 전체 실행 사이클에서는 1246번에서 862번으로 30.8% 감소하여 성능향상을 보였다.

이를 종합해 보면, 네트워크를 위한 패킷 처리 시 overaly 명령어를 통해 비트 스트림 연산을 2배 가속시킬 수 있으며 시프트, 회전, 오버레이를 최적할 경우 동작 지연 시간과 실행 사이클에서 각각 20.7%와 30.8% 감소시켜 네트워크 환경을 가속함을 볼 수 있다.

VI. 결 론

본 논문에서는 네트워크 환경에서 패킷 연산을 가속하기 위한 ASIP을 구성하였다. 이 과정에서 비트 스트림 연산을 가속할 수 있는 명령어 세트를 제안하였으며, 이를 위한 가속기 모듈을 설계하였다. 이 가속기는 barrel 시프터를 기반으로 최적화 하였으며, 빠른 패킷 처리를 위해 시프트, 마스크, 패킹을 모듈화 하였다.

네트워크 어플리케이션을 입력으로 한 실행시간 시뮬레이션에서 6.4usec의 성능 향상을 확인하였다. 기준 프로세서는 CoWare사의 Processor Designer를 이용하여 각 블록별 RTL 코드를 생성하고 overlay 모듈을 최적화하여 이를 Synopsys의 Design Compiler를 이용하여 TSMC 0.25um 공정으로 합성한 결과, 제안하는 명령어를 추가할 경우, 동작 지연시간은 약 20.7% 감소하였다. 동작 지연시간의 감소에 따라 네트워크 패킷연산 처리의 실행 사이클 수는 30.8% 감소하였다. 면적과 동작 지연시간 모두에서의 이와 같은 개선은 각각에서 생성된 시프트 모듈을 하나의 레지스터를 이용하여 공용

화하면서 최적화하였기 때문이다. 이로 볼 때, 본 논문에서 설계한 ASIP은 충분히 네트워크 환경을 가속한다고 볼 수 있다. 추가로 향후 진행되어야 할 연구는 네트워크 테스트 벤치에서 본 논문의 가속기와 컴파일러를 적용하였을 때의 성능 향상을 확인함으로서 실제 네트워크 패킷 처리 시 성능 향상을 볼 수 있다.

참 고 문 헌

[1] Haiyong Xie, Li Zhao and Laxmi Bhuyan, "Architectural Analysis and Instruction-set Optimization for Design of Network Protocol Processors.", ACM, October 2003.

[2] Matthias Grunewald and 8 person, "Network Application Driven Instruction Set Extention for Embeded Processing Clusters.", in Proceedings of PARELEC, September 2004.

[3] Bengu Li and Rajiv Gupta, "Bit Section Instruction Set Extention of ARM Processor Accelator Using Reconfiguration Logic.", ACM, October, 2002.

[4] Gokhan Memik, Seda Ogrenci Memik and William H.Mangione-Smith, "Design and Analysis of a Layer Seven Network Processor Accelator using Reconfigurable Logic." in IEEE Symposium, 2002.

[5] J.Wagner and R. Leupers, "C Compiler Design for a Network Processor.", in IEEE Transctions on Computer-Aided Design, November 2001.

[6] Tilman Wolf, "Design of a Instruction set for Modular Network Processor.", IBM Research Report, 27 October 2000.

[7] Network Test Bench, EEMBC Inc. [Online]. Available: http://www.eembc.com

[8] Lal George and Mathias Blume, "Taming the IXP Network Processor", ACM 2003.

[9] Woo-Kyeong Jeong and Yong-Surk Lee, "A Universal Shifter with Packed Data Formats", AEU 2003.

저 자 소 개

윤 여 필(정회원)

2001년 인하대학교 전기전자 공학과 학사 졸업.

2001년~현재 삼성전자 DM총괄 선임연구원.

2007년~현재 연세대학교 전기전자공학과 석사과정.

<주관심분야 : 마이크로프로세서, 네트워크프로세서, SOC>

이 정 희(정회원)

1984년 경북대학교 전자공학과 학사 졸업.

1990년 경북대학교 전자공학과 석사 졸업.

1984년~현재 한국전자통신 연구원 광대역통합망 연구원.

<주관심분야 : 네트워크자원관리, 인터넷QOS>

이 용 석(정회원)

1973년 연세대학교 전기공학과 학사 졸업.

1977년 Univesity of Michi-gan, Ann Arbor 석사 졸업.

1981년 University of Michi-gan, Ann Arbor 박사 졸업.

1993년~현재 연세대학교 전기전자공학과 교수

<주관심분야 : 마이크로프로세서, 네트워크프로세서, 암호화프로세서, SOC>