SDR(Software Defined Radio)에 적합한 네트워크 코프로세서 구조의 설계

정회원 김 현 필*, 정 하 영*, 함 동 현*, 이 용 석*

The Design of a Structure of Network Co-processor for SDR(Software Defined Radio)

Hyun-pil Kim*, Ha-young Jeong*, Dong-hyeon Ham*, Yong-Surk Lee* ***Regular Members***

요 약

디지털 컨버전스가 이루어지면서 무선기기들 간의 호환성은 단말기의 중요한 특성이 되었고, SDR은 가장 필요한 기술이고 표준이다. 하지만 통신 프로토콜이 다른 무선 환경에서 호환성을 갖는 단말기를 하드웨어만을 이용한 ASIC이나 SoC로 만들기는 어려운 실정이다. 그래서 본 논문은 여러 통신 프로토콜을 가속화 시킬 수 있는 코프로세서의 구조를 제안하였다. 메인 프로세서와 쉽게 연동이 되고, 네트워크의 PHY 레이어에 특화된 코프로세서가 바로 그것이다. 통신 시스템에서 가장 많이 사용하는 변조 방식인 OFDM과 CDM을 사용하는 무선 랜 표준 IEEE802.11a와 IEEE802.11b를 모델링한 C 프로그램을 ARM cross 컴파일러를 이용해 컴파일 하였고, Simplescalar-Arm 버전을 이용해 시뮬레이션 및 프로파일을 수행하였다. 프로파일 결과 비터비 연산과 부동 소수점 복소수 연산이 가장 많은 연산을 차지하였다. 프로파일 결과를 바탕으로 비터비 연산과 부동 소수점 복소수 연산을 가속화 할 수 있는 코프로세서를 제안하여 명령어를 추가했으며, 추가된 명령어는 Simplescalar-Arm 버전을 이용해 시뮬레이션 하였다. 시뮬레이션 결과 ARM 코어 하나만 사용 했을 때보다 비터비 연산은 약 4.5배, 부동 소수점 복소수 연산은 약 2배의 성능 향상을 보였다. IEEE802.11a에서는 일반 ARM 코어보다 약 3배의 성능 향상을 보였고, IEEE802.11b에서는 약 1.5배의 성능 향상의 보였다.

key Words : SDR, 코프로세서, IEEE802.11, OFDM, CDM

ABSTRACT

In order to become ubiquitous world, the compatibility of wireless machines has become the significant characteristic of a communication terminal. Thus, SDR is the most necessary technology and standard. However, among the environment which has different communication protocol, it's difficult to make a terminal with only hardware using ASIC or SoC. This paper suggests the processor that can accelerate several communication protocol. It can be connected with main-processor, and it is specialized PHY layer of network. The C-program that is modeled with the wireless protocol IEEE802.11a and IEEE802.11b which are based on widely used modulation way; OFDM and CDM is compiled with ARM cross compiler and done simulation and profiling with Simplescalar-Arm version. The result of profiling, most operations were Viterbi operations and complex floating point operations. According to this result, we suggested a co-processor which can accelerate Viterbi operations and complex floating point operations and added instructions. These instructions are simulated with Simplescalar-Arm version. The result of this simulation, comparing with computing only one ARM core, the operations of Viterbi improved as fast as 4.5 times. And the operations of complex floating point improved as fast as twice. The operations of IEEE802.11a are 3 times faster, and the operations of IEEE802.11b are 1.5 times faster.

※ 본 연구는 정보통신 연구 진흥원이 지원한 차세대 다목적 프로세서 아키텍처 연구 과제를 통해 수행되었습니다.

* 연세대학교 전기전자공학과 프로세서 연구실 (hpkim@dubiki.yonsei.ac.kr)

논문번호 : KICS2006-10-432, 접수일자 : 2006년 10월 10일, 최종논문접수일자 : 2007년 2월 28일

Ⅰ. 서 론

SDR(Software Defined Radio) 기술은 통신 방식의 다양화에 의한 문제를 해결하기 위한 새로운 통신 시스템의 개념으로서, 단일 시스템 하드웨어 상에서 여러 가지 통신 프로토콜을 소프트웨어를 기반으로 다양한 통신 서비스를 지원하는 것을 말한다[1-3]. 이는 하나의 단말기에서 소프트웨어 기반의 여러 통신 서비스를 이용할 수 있을 뿐만 아니라, 효율적으로 시스템을 업그레이드하고, 쉽게 보수 및 유지를 할 수 있다는 장점을 가진다. 결국 SDR 기술은 통신 시스템 이론을 핵심으로 하여 디지털 신호 처리 기술, 디지털 및 아날로그 반도체 기술 및 RF 기술을 바탕으로 한 다목적 시스템을 이끌어 내고, 이 시스템을 운영하는 구체적인 소프트웨어를 개발하는 과정과 관련된 총체적인 통신 시스템 기술을 의미한다.

초기의 SDR은 전통적인 통신 시스템 방식과는 달리 아날로그-디지털 변환을 통해 단순히 디지털 영역에서 신호를 처리하는 것으로 정의하였다[1,2]. 하지만 최근에는, 안테나와 아날로그-디지털 변환, 디지털-아날로그 변환만을 하드웨어로 구현하고, 나머지 통신 규약이나 서비스 등은 모두 소프트웨어로 구현하여, 쉽게 시스템 변경이 가능한 넓은 의미의 통신 체계를 일컫는다.[2,3]

이런 SDR의 가장 큰 장점은 다중대역/다중모드를 지원하고, 이기종간 및 기존 무선 시스템과의 상호 연동이 가능하며, 고속 통신과 네트워킹이 가능하다는 것이다. 그래서 이를 위해 기존 시스템과의 상호 운용 능력 향상 및 새로운 시스템으로의 확장성을 만족하기 위한 개방형 소프트웨어 구조를 정의한 SCA(Software Communication Architecture) 규격이 발표되었고, 이를 근간으로 SDR 포럼 등에서 상용화를 위한 표준화가 진행 중이다[4]. 최근 SDR 포럼에서는 SDR의 요건으로 ① 다중대역/다중모드 지원을 위해 재구성 가능한 개방형 구조일 것, ② 미들웨어, OS, 인터페이스 등이 개방 표준일 것, ③ 재사용성 극대화를 위한 객체 지향 프레임워크, ④ SCA 프레임 워크의 서비스 및 제어 인터페이스를 지원할 것으로 소프트웨어의 구조적측면이 대두되었다. 이에 따라 구성된 이상적인 SDR의 구조는 아래의 그림1과 같다[5].

SDR의 정의에 대한 초점은 과거 하드웨어 중심에서 소프트웨어 중심으로 바뀌어져 있으나 주요 기반 기술은 계속해서 많은 연구가 이루어지고 있다. 특히 SDR 포럼에서는 광대역 RF/안테나, 프로세서 등의 하드웨어, SCA, IDL 등의 소프트웨어, 다중안테나 등의 디지털 신호처리의 분야에 대해 기술개발 분과를 따로 두어 운영하고 있다[6,7].

SDR의 이상적인 형태는 모든 처리를 소프트웨어

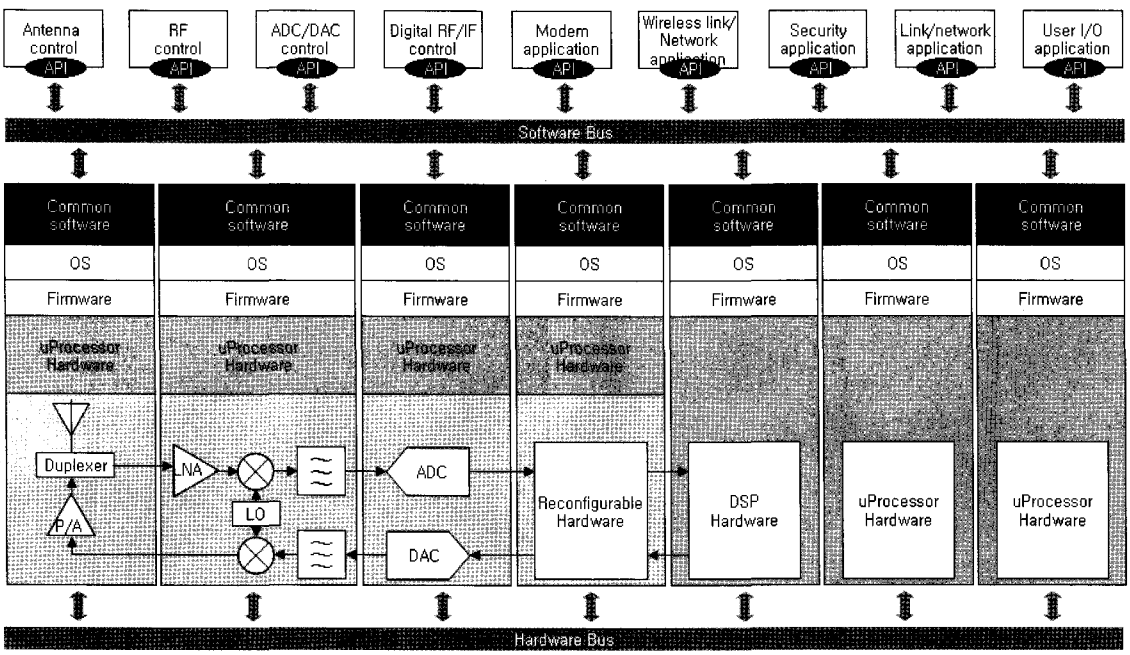


그림 1. 이상적인 SDR의 구조

로 처리함을 목적으로 하므로, 하드웨어 분야에서는 보다 고성능의 프로세서가 필요하다. 하지만 단일의 프로세서가 단말기의 시스템과 SDR을 모두 처리하기에는 부하가 너무 크다. 그래서 본 논문에서는 SDR을 구현하기 위한 PHY 레벨의 소프트웨어를 가속화시켜 구동시킬 수 있고 단말기와의 연계성과 확장성이 높은 통신 프로토콜에 특화된 코프로세서를 제안하고자 한다.

Ⅱ. 관련 연구

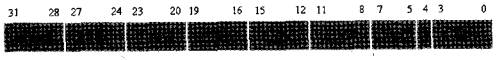
90년대 중반 이후 인터넷 속도와 대역폭이 증가하면서, 고속이면서 낮은 유연성을 보이는 ASIC과 저속이면서 높은 유연성을 보이는 임베디드 프로세서의 장점을 결합한 네트워크 프로세서가 등장하였다. 하지만 인텔의 IPX1200과 같은 네트워크 프로세서들은 주로 CRC(Cyclic Redundancy Checking) 나 ECC(Error Correction Code)등 MAC 레이어 이상의 연산에서 빠른 수행 속도를 보이고, 라우터 등의 고속 광통신에서 주로 사용된다[8]. 뿐만 아니라 상위 네트워크 레이어로 올라가면서 프로그램이 점점 복잡해지므로 C/C++과 같은 상위레벨 언어를 지원해야하고 복잡한 프로그램의 검증을 위한 시뮬레이터와 컴파일러, 디버거 등이 요구된다[9]. 하지만 SDR은 무엇보다도 이동성과 임베디드 특성, 프로그램의 유연성을 많이 요구하는 시스템이다. 이런 특성들로 인해 기존의 네트워크 프로세서로 SDR을 구현하기에는 사실상 어려운 실정이다.

SDR 구현을 위한 프로세서 구조로 multi-threaded baseband processor가 있다. 이 프로세서의 구조는 하나의 메인 코어가 있고, 통신 프로토콜에 특화된 연산을 수행하는 DSP(Digital Signal Processor)를 주변에 두어 서로 관련성이 없는 프로세스를 메인 코어와 DSP로 나누어 동시에 해당 프로세스를 수행하는 구조이다[10]. 하지만 이런 프로세서 구조는 각 프로세서들 간의 thread 연산을 위해서 컴파일러가 미리 관련이 없는 프로세스를 분리해야 한다. 따라서 컴파일러의 성능이 전체 시스템에 많은 영향을 끼치게 되며, 사용하는 DSP 마다 다른 컴파일러를 사용해야 한다는 단점이 있다. 또한 메모리와 레지스터, 내부 인터페이스에서의 병목현상이 발생하기도 한다.

앞의 두 방법 외에 SDR에 적합한 프로세서 구조로 메인 프로세서와 그 메인 프로세서와 쉽게 연동이 가능한 코프로세서를 생각해 볼 수 있다. 대부분의 모바일 기기에 많이 사용되는 ARM 코어와 같은 메인 프로세서들은 임베디드 특성도 우수할 뿐만 아니라 대중성으로 인한 컴파일러나 디버거, 시뮬레이터 기능이 아주 우수하다.[11] ARM 코어와 ARM 코어와 연동이 가능한 코프로세서로 시스템을 구현한다면 ARM 코어와 코프로세서의 컴파일러를 개별적으로 디자인 할 필요 없이 ARM 컴파일러에 코프로세서 부분만 추가하면 된다. ARM 코어가 메인 프로세서로서 코프로세서들을 중재하기 때문에 메모리나 레지스터, 인터페이스에서의 병목현상도 방지할 수 있다. 이런 장점들로 SDR을 구현에 가장 적절한 하드웨어 시스템으로 메인 프로세서와 통신 프로토콜 프로그램을 가속화 할 수 있는 여러 개의 코프로세서로 이루어진 프로세서 구조를 제안한다.

Ⅲ. 본 론

ARM 프로세서는 전 세계적으로 가장 많이 사용되고 있는 상용 프로세서 코어이다. 이는 저전력이며, 32 비트 RISC(Reduced Instruction Set Computer)의 고성능 프로세서로서 독자 생산 방식이 아닌 라이센스 방식으로, 라이센스를 사들인 업체가 주변회로를 구성하여 독자적으로 생산하고 판매할 수 있으며 그림 2와 같이 다양한 코프로세서를 지원할 수 있는 명령어 세트가 할당되어 있기 때문이다. 본 논문에서도 ARM 코어에서 제공하는 코프로세서 영역을 이용하여 SDR에 적합한 코프로세서를 설계하였다.



CDP{<cond>} <CP#>, <Cop1>, CRd, CRn, CRm{, <Cop2>}

그림 2. ARM 프로세서의 명령어 세트

3.1 실험 방법

현재 가장 많이 사용하고 있는 디지털 무선 통신 방식은 코드 분할 다중(CDM) 방식과 직교 주파수 분할 다중(OFDM) 방식이다. CDM과 OFDM은 이동성과 각종 노이즈에 강한 특성 때문에 휴대폰(WCDMA)뿐만 아니라 무선 랜(IEEE802.11)에서부터 DMB(Digital Multimedia Broadcasting), 차세대 통신(Wibro, 4G)에서도 각광받고 있는 변조 방식이다. 무선 통신 시스템을 분석하기 위해, 무선 통신 시스템의 대표적인 변조기술인 CDM과 OFDM을

모두 사용하는 무선 랜 802.11a와 802.11b를 모델링한 C 프로그램[12]을 사용하였다. 컴파일을 위해서는 ARM cross 컴파일러를 이용 하였으며 Simplescalar-Arm 버전을 사용하여 시뮬레이션 하였다.

3.2 Workload 분석

3.2.1 802.11a

802.11a는 IEEE에서 개발한 무선 랜 규격의 경로 공유 프로토콜로서 이더넷 프로토콜과 반송파 감지 다중 접근/충돌 회피(CSMA/CA) 프로토콜을 사용한다. 무선 비동기 전송 방식(ATM) 시스템에 적용되고, 5GHz와 6GHz 사이의 주파수에서 운용되며, 데이터 속도는 최대 54Mbps이나 공통적으로 통신 시에는 6Mbps, 12Mbps, 24Mbps를 사용한다.

그림 3은 802.11a OFDM 방식의 무선 랜을 모델링한 프로그램을 ARM 코어로 시뮬레이션 했을 때, 각 함수별 수행 시간을 나타낸 그래프이다. 그림에서 볼 수 있듯이, 802.11a에서 가장 많은 수행시간을 차지하는 함수는 비터비 함수이다. 즉, 메인 프로세서인 ARM 코어는 802.11a를 연산하는데 약 50%의 시간을 비터비 함수를 수행하는데 사용한다. 그 외의 나머지 약 40% 정도를 idft 함수와 복소수 연산에 사용한다. 시뮬레이션에 사용된 C 프로그램은 기본적인 AWGN(Addictive White Gaussian Noise) 채널 특성까지 고려한 모델이기 때문에 이 채널 특성을 수행하는 함수를 제외한다면, 앞에서 언급한 비터비 함수와 idft함수, 복소수 연산이 거의 99%정도 차지한다고 할 수 있다.

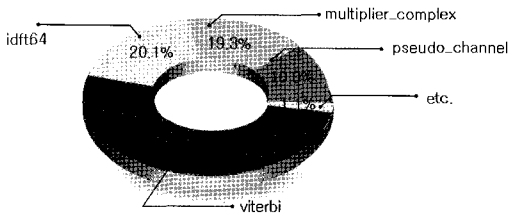


그림 3. 802.11a profile 결과

3.2.2 802.11b

802.11b는 IEEE가 1997년에 정한 무선 LAN 규격인 IEEE 802.11의 차세대 규격 'IEEE 802.11b high rate'라고도 하며, 2.4GHz대의 주파수를 사용한다. 데이터 전송 속도는 IEEE 802.11에서는 최고 2Mbps였으나 IEEE 802.11b에서는 11Mbps까지 증가시켰으며, IEEE 802.11과도 호환성이 있다.

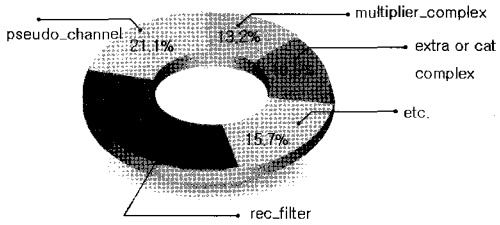


그림 4. 802.11b profile 결과

그림 4은 802.11a와 마찬가지로, 802.11b CDM 방식의 무선 랜을 모델링한 C 프로그램을 ARM 코어로 시뮬레이션 했을 때 각 함수 별 수행시간을 나타낸 결과이다. 그림에서 볼 수 있듯이 802.11b는 전체 수행시간 중에서 receiver filter 함수가 약 40%, 복소수 연산 함수가 약 13%의 수행시간을 차지한다. 여기에서도 802.11a와 마찬가지로 AWGN 채널을 제외한다면 앞의 두 연산이 전체 연산의 약 78%를 차지한다.

3.3 C코드 분석 및 코프로세서 명령어 추가

앞 절에서 살펴본 workload 분석을 토대로 가장 많은 수행시간을 보이는 함수를 살펴보면 비터비 함수, idft 함수, 복소수 연산 함수, receiver filter 함수 등이 있다. 앞의 함수들 내에서 많이 사용된 연산을 찾아, 그 연산을 가속화 시킬 수 있는 Co-processor 명령어를 추가하였다.

3.3.1 비터비 연산을 가속화 할 수 있는 명령어

표 1은 비터비 함수에서 많이 사용되는 연산을 나타낸다. 표 1에서 볼 수 있듯이, 가장 많이 사용되는 연산은 메모리에서 두 개의 데이터를 읽어와 이것을 쉬프트하거나 더하고 빼는 과정을 통해 다시 메모리에 저장하는 연산이다. 이를 단순히 ARM 어셈블리로 연산한다면 매번 데이터를 load하고 다시 레지스터에 move하고 shift한 후 add나 and 등의 산술 연산을 수행하게 된다. 그리고 이 함수에서 사용하는 변수는 short(8bit)로 선언되어 있는데, 실제 프로세서 내부에서는 32비트로 확장되어 연산된다.

C코드를 보면, 이 연산들은 서로의 결과가 다른 연산에 영향을 미치지 않기 때문에 병렬적으로 수행할 수 있다. 그래서 이점에 착안해 표2와 같이 비터비 함수를 가속할 수 있는 코프로세서 명령어를 추가했다.

3.3.2 복소수 연산을 가속화 할 수 있는 명령어

표 3, 4, 5의 복소수 곱셈 함수와 receiver filter

표 1. 비터비 함수에서 많이 사용된 연산

```
brmetric[0] = tmp1 + tmp2;
brmetric[1] = (0xf-tmp1) + tmp2;
brmetric[2] = tmp1 + (0xf-tmp2);
brmetric[3] = (0xf-tmp1) + (0xf-tmp2);
...
tmp0 = (jj>>1) + 0;
tmp1 = (jj>>1) + 0x20;
tmp0 = metric0[tmp0] + brmetric[ conv[jj][0] ];
tmp1 = metric0[tmp1] + brmetric[ conv[jj][1] ];
...
temp1 = (shifter &0x1) ;
temp1 ^= ((shifter>>2) &0x1) ;
temp1 ^= ((shifter>>3) &0x1) ;
temp1 ^= ((shifter>>5) &0x1) ;
temp1 ^= ((shifter>>6) &0x1) ;
temp2 = (shifter &0x1) ;
temp2 ^= ((shifter>>1) &0x1) ;
temp2 ^= ((shifter>>2) &0x1) ;
temp2 ^= ((shifter>>3) &0x1) ;
temp2 ^= ((shifter>>6) &0x1) ;
```

표 2. 비터비 함수에서 코프로세서에 추가된 명령어

명령어	형식	설명
ldc2{n}	ldc2{n} cp#, crd, M[]	M[]번지부터 n비트씩 load하여 crd에 저장
str2{n}	str2{n} cp#, crd, M[]	crd의 n비트를 M[]번지에 store
cdp0	cdp cp#, 0, crd, crs	crs의 상·하위 16비트 데이터를 더해서 crd에 저장
cdp1	cdp cp#, 1, crd, crs	crs의 상위 16비트를 inverse 후 하위 16비트와 더해 crd에 저장
cdp2	cdp cp#, 2, crd, crs	crs의 하위 16비트를 inverse 후 상위 16비트와 더해 crd에 저장
cdp3	cdp cp#, 3, crd, crs	crs의 상·하위 16비트를 inverse 후 더해 결과를 crd에 저장
cdp4	cdp cp#, 4, crd, crs, crt	crs을 1비트 shift right 한 후 crt와 더해 crd에 저장
cdp5	cdp cp#, 5, crd, crs, crt	crs을 1비트 shift left 한 후 crt와 더해 crd에 저장
cdp6	cdp cp#, 6, crd, crs, #n	crd을 n번 shift right 한 후 crs와 and연산, 결과를 crs에 저장
cdp7	cdp cp#, 7, crd, crs, #n	crd을 n번 shift left 한 후 crs와 and연산, 결과를 crd에 저장

함수, idft 함수에 자주 나오는 연산들은 모두 부동소수점의 복소수 연산을 기반으로 하고 있다. 현재 ARM 코어뿐만 아니라 대부분의 상용 프로세서들

표 3. 복소수 곱셈 연산

```
product.realp = a.realp*b.realp - a.image*b.image ;
product.image = a.realp*b.image + a.image*b.realp ;
```

표 4. idtf 함수에서 많이 사용된 연산

```
tmp1.realp += tmp2.realp ;
tmp1.image += tmp2.image ;
```

표 5. receiver filter 함수에서 많이 사용된 연산

```
top[jj-MM+1].realp += datain.data[jj+ii-MM+1].realp * tmp0;
top[jj-MM+1].image += datain.data[jj+ii-MM+1].image * tmp0;
```

표 6. 복소수 연산을 위해 코프로세서에 추가된 명령어

명령어	형식	설명
cdp8	cdp cp#, 8, cfd, cfs, cft	cfs와 cft에 있는 복소수를 더하여 cfd에 저장
cdp9	cdp cp#, 9, cfd, cfs, cft	cfs와 cft에 있는 복소수를 뺀 후 cfd에 저장
cdp10	cdp cp#, 10, cfd, cfs	cfd와 cfs의 복소수를 더해 cfd에 저장
cdp11	cdp cp#, 11, cfd, cds	복소수 cfs에서 cfs를 뺀 후 cfd에 저장
cdp12	cdp cp#, 12, cfd, cfs, cft	cfs와 cft를 곱한 뒤 cfd의 값과 더해서 cfd에 저장
ldc	ldc cp#, cfd, M[]	M[]주소의 데이터를 cfd에 저장
stc2{n}	stc2{n} cp#, cfs, M[]	cfs의 n비트를 M[]주소에 저장

중에 부동 소수점 연산을 한 클록 사이클에 수행할 수 있는 프로세서는 없다. 대부분의 프로세서가 2~5 클록 사이클을 필요로 한다. ARM 코어의 경우에는 코어 내부에 부동 소수점 연산을 위한 연산기가 없으며, 부동 소수점 연산을 위해서는 메인 코어 외부에 부동 소수점 코프로세서를 필요로 하는데, 이 경우에 부동 소수점 곱셈이 5 클록 사이클이 필요하다.[8] 이는 네트워크 연산기로서는 상당히 느린 속도라고 할 수 있다. 이런 느린 부동 소수점 연산을 개선하기 위해서는 부동 소수점 복소수 연산을 가속화할 수 있는 새로운 형태의 프로세서가 필요하다.

복소수 연산을 살펴보면, 실수부와 허수부의 연산은 따로 수행하고 두 연산 사이에는 연관성이 존

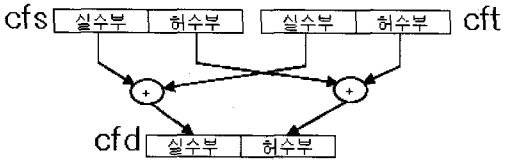


그림 5. 복소수 SIMD 덧셈 연산

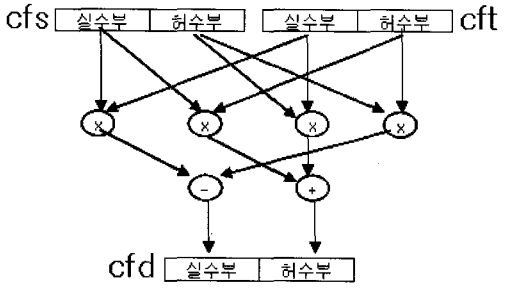


그림 6. 복소수 SIMD 곱셈 연산

재하지 않는다. 즉, 실수부와 허수부를 동시에 연산할 수 있다. 이런 점에 착안 해 본 논문에서는 허수부와 실수부를 동시에 연산할 수 있는 SIMD(Single Instruction Multi-Data)형태의 부동소수점 연산기를 이용하여 복소수 연산을 가속하는 방법을 채택하였다.

SIMD 연산을 일반 연산과 같이 사용하기 위해서는 동시에 연산되는 데이터의 크기만큼 레지스터의 크기도 커져야 한다. 그래서 32비트의 크기를 갖는 부동 소수점 레지스터를 64비트로 확장하여, 상위 32비트에는 실수부 수를, 하위 32비트에는 허수부 수를 저장하여 각각 독립적으로 연산이 되도록 하였다.

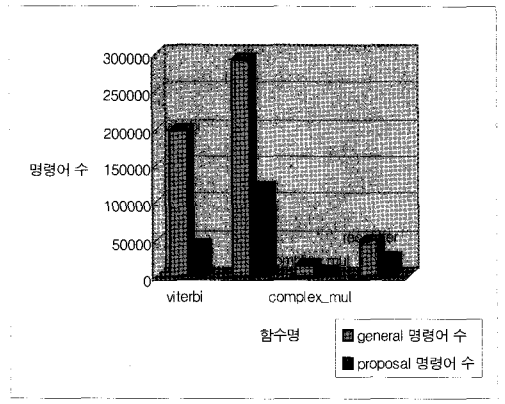


그림 7. 함수별 명령어 수 비교

덧셈을 할 경우에는 그림 5와 같은 방식으로 코프로세서의 각 부동 소수점 레지스터의 상·하위 32비트를 각각 더한다. 그리고 곱셈을 할 경우에는 그림 6와 같은 방식으로 곱셈이 이루어진다. 이런 방식으로 부동 소수점 복소수 연산을 할 경우 이론적으로 두 배 이상의 연산 속도 향상을 기대할 수 있다.

3.4 실험 결과

코프로세서 명령어가 추가된 Simplescalar-Arm 시뮬레이터로 시뮬레이션 한 결과 그림7과 같은 결과가 나왔다. 시뮬레이션 결과 비터비 함수는 단순히 ARM 코어 하나만 가지고 연산했을 때 보다 약 4.5배 정도의 속도 향상이 있었고, 복소수 연산의 경우 idft 함수, 복소수 곱셈 함수, receiver filter함수 각각에서 약 2.4, 2.1, 1,9 배 정도의 성능 향상이 있었다. 802.11a 무선 랜 표준에서 비터비 함수가 전체 수행 시간의 50%정도 차지하고, 복소수 연산인 idft 함수와 복소수 곱셈 함수가 전체 수행 시간의 약 40%정도를 차지하므로 전체적인 성능은 약 3배정도 빨라졌으며, 802.11b 무선 랜 표준에서는 receiver filter가 약 40%, 복소수 곱셈이 약 20%정도 차지하므로, 전체적으로는 약 1.5배의 성능 향상을 보였다. 이는 채널을 모델링하는 연산까지 포함된 수치이므로 이 부분을 제외한다면 더 높은 성능 향상이 기대된다.

Ⅳ. 결 론

디지털 컨버전스로 인한 무선기기들 간의 호환성으로 SDR은 가장 필요한 기술이고 표준이 되었다. 하지만 통신 프로토콜이 다른 무선 환경에서, 호환성을 갖는 단말기를 ASIC이나 SoC로 만들기는 어려운 실정이다. 이점에 착안 해 본 논문은 호환성도 높고 일반 프로세서보다 네트워크 프로그램을 가속화 시킬 수 있는 코프로세서제안 하였다.

현재 통신 시스템에서 가장 많이 사용되고 있는 OFDM과 CDM을 사용하는 무선 랜 표준인 IEEE802.11 a와 IEEE802.11b를 모델링한 C 프로그램을 타깃으로 메인 프로세서는 ARM 코어를 사용하였으며, 시뮬레이션은 Simplescalar-Arm 버전을 이용 하였다. 시뮬레이션 결과 비터비 함수와 부동소수점 복소수 연산 함수가 가장 많은 연산을 차지하였다.

함수 별 수행 시간을 바탕으로 비터비 연산과 부

동 소수점 복소수 연산을 가속화 할 수 있는 Co-processor를 제안하여 명령어를 추가하였다. 그리고 추가된 명령어를 컴파일러와 시뮬레이터에 추가 하여 시뮬레이션 하였다. 그 결과 비터비 연산은 약 4.5배, 부동 소수점 복소수 연산은 약 2배의 성능 향상을 보였다. IEEE802.11a에서는 일반 ARM 코어만 쓰는 것보다 약 3배의 성능 향상을 보였고, IEEE802.11b에서는 약 1.5배의 성능 향상의 보였다.

참 고 문 헌

[1] J. Mitola Ⅲ, Software Radio Architecture, John Wiley & Sons, Inc, 2000.

[2] Software defined radio(SDR) forum, http://www.sdrforum.org

[3] J. H. Reed, Software radio: a morden approach to radio engineering, Prentice Hall, 2002

[4] Software communications architecture Specification V2.2, JTRS Joint Program Office, 2001.

[5] G. Fogarty, "SDR complication," SDR forum Contribution, Jun. 2002.

[6] R. Morelos-Zaragoza, S. Haruyama, M. Abe, N. Sasho, L. B. Michal, R. Kohno, " A software radio receiver with direct conversion and its digital processing," IEICE Trans. Commun., Vol. E85-B, No. 12, Dec. 2002.

[7] H. Tsurumi, Y. Suzuki, "Broadband RF stage architecture for software-defined radio in handheld terminal applications," IEEE Commun. Mag., Vol.37, No. 2, pp. 90-95, Feb 1999.

[8] http://www.intel.com

[9] http://www.mpu.yonsei.ac.kr

[10] John Glossner, Daniel Iancu, Jin Lu, Erdem Hokenek, and Mayan Moudgill, "A SOFTWARE DEFINED COMMUNICATIONS BASEBAND DESIGN", IEEE Communications Magazine,. Vol. 41, No. 1, January, 2003, pp. 120-128.

[11] David Seal, Architecture Reference Manual, Addison wesley

[12] http://www.wifi-forum.com

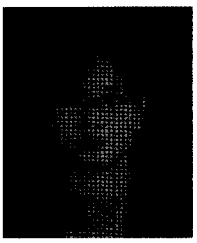
김 현 필 (Hyun-pil Kim) 정회원

2005년 2월 연세 대학교 전자공학과 학사

2005년 9월~현재 연세대학교 전기전자공학과 석사과정

<관심분야> 프로세서, SDR, ASIC, SoC, 영상처리

정 하 영 (Ha-young Jeong) 정회원



2003년 2월 중앙대학교 전자공학과 학사

2005년 2월 연세대학교 전자공학과 석사

2005년 3월~현재연세대학교 전기전자공학과 박사과정

<관심분야> 영상처리, 마이크로프로세서, SoC

함 동 현 (Dong-hyeon Ham) 정회원

2006년 2월 성균관대학교 정보통신공학사

2006년 3월~현재 연세대학교 전기전자 공학과 석사과정

<관심분야> 영상처리, 프로세서, SoC

이 용 석 (Yong-Surk Lee) 정회원

1973년 2월 연세대학교 전기공학과 학사

1977년 2월University of Michigan, Ann Arbor 석사

1981년 2월 University of Michigan, Ann Arbor 박사

1993년~현재 연세대학교 전기전자공학과 교수

<관심분야> 마이크로프로세서, 네트워크 프로세서, 암호화 프로세서, SoC