VoWiFi 음질 향상을 위한 G.729.1 광대역 코덱의 ARM 프로세서에의 실시간 구현

A Real-time Implementation of G.729.1 Codec on an ARM Processor for the Improvement of VoWiFi Voice Quality

박남인, Nam In Park*, 강진아, Jin Ah Kang**, 김홍국, Hong Kook Kim***

요약 본 논문에서는 ARM 프로세서로 설계된 VoWiFi 단말기에서 광대역 음성 서비스를 가능하게 하기 위한 방법으로 ITU-T 표준 코덱인 G.729.1을 실시간으로 구현하고 그 성능을 평가한다. 실시간 G.729.1 코덱 구현은 C 코드 최적화 및 코덱 알고리즘의 고속화를 근간으로 한다. 이렇게 최적화된 코덱의 성능은 VoWiFi 단말기내에서 ARM 프로세서가 요구하는 CPU 동작 시간으로 평가된다. 실험 결과, ARM926EJ를 사용하여 최적화된 G.729.1 코덱이 실시간으로 동작함을 확인할 수 있으며, 기존의 G.729에 비해 넓은 대역폭의 음성 전송이 가능함을 보일 수 있다.

Abstract This paper addresses issues associated with the real-time implementation of a wideband speech codec such as ITU-T G.729.1 on an ARM processor in order to provide an improved voice quality of a VoWiFi service. The real-time implementation features in optimizing the C-source code of G.729.1 and replacing several parts of the codec algorithm with faster ones. The performance of the implementation is measured by the CPU time spent for G.729.1 on the ARM926EJ processor that is used for a VoWiFi phone. It is shown from the experiments that the G.729.1 codec works in real-time with better voice quality than G.729 codec that is conventionally used for VoIP or VoWiFi phones.

핵심어: VoWiFi, Speech codec, G.729.1, Real-time implementation, ARM Processor

* 주저자 : 광주과학기술원 정보통신공학과 e-mail: naminpark@gist.ac.kr
** 공동저자 : 광주과학기술원 정보통신공학과 e-mail: jinari@gist.ac.kr
*** 교신저자 : 광주과학기술원 정보통신공학과 교수 e-mail: hongkook@gist.ac.kr

1. 서론

2000년 초반에 등장한 VoIP(Voice over IP) 서비스는 지속적인 기술개발과 최근 번호체계 등의 정책적인 문제가 해결되면서 안정적인 성장을 보이고 있다. 특히 VoWiFi와 같은 VoIP 무선 서비스 기술의 등장은 VoIP 시장을 증폭시킬 차세대 서비스로 기대되고 있다. 그러나 현재 WiFi 무선 네트워크는 VoIP 서비스를 지원하기 위한 두 가지 주요 문제점을 가지고 있다. 그것은 불안정성과 긴 핸드오프와 그에 따른 버스트 패킷 손실 특성이다. 이러한 특성은 실시간 연속적으로 전달되는 음성 품질을 크게 저해한다 [1][2].

인터넷 전화 서비스가 처음 등장했을 때, 획기적인 서비스로 주목 받았지만, 음질이 기존의 음성 전화보다 좋지 못한 이유 때문에 크게 발전하기 못 했다. 이러한 인터넷 전화의 음질을 개선하기 위해 다양한 방법이 진행되어 왔는데 그중 하나가 50~7000 Hz 대역의 음성 신호를 처리하는 광대역 코덱의 사용이다. 이러한 고품질 인터넷폰 서비스를 위한 광대역 코덱으로 ITU-T에 의해 표준화된 최신의 G.729.1 코덱이 사용되며, 실제 상용화하기 위해서는 실시간 구현이 필요하다. 기존의 G.729.1 코덱은 계산량이 많이 소요되기 때문에 최적화 과정이 수행되어야 된다.

본 논문에서는 ITU-T에 의해 표준화된 최신의 G.729.1 음성 코덱[3]을 적용하여 VoWiFi의 성능을 개선하고자 한다. 본 논문의 구성은 다음과 같다. 2장에서는 G.729.1 음성 코덱에 대해 설명하고, 3장에서는 무선 응용에 적합한 ARM 프로세서에 기반한 G.729.1 구현에 대해 설명한다. 4장에서는 G.729.1 코덱의 실시간 구현을 위해 수행한 최적화 과정에 대해 설명한다. 5장에서는 이에 대한 성능 평가 결과를 설명한 후, 6장에서 결론을 맺는다.

2. G.729.1 구조

G.729.1 코덱은 가변대역 광대역 신호를 지원하는 코덱으로, 고품질의 대역폭 확장 및 품질 제어가 가능하여 유선 및 WiFi 무선 네트워크에 적합하다. G.729.1 코덱은 20 ms 단위로 16 kHz, 16 bit 선형 PCM 신호를 입력으로 받아 처리하며, 4 kHz의 낮은 대역폭 음성은 G.729의 형태로 압축하고, 상위 4 kHz의 음성에 대해서는 대역폭 확장 기법으로 광대역 음성을 압축하는 임베디드 코덱이다. 이는 전송 채널의 용량이 작을 경우, 즉 WiFi가 잡음 채널일 경우는 협대역 음성을 최소 8 kbit/s로, 반면 전송 채널의 용량이 충분한 경우는 32 kbit/s로 광대역 음성을 스케일러블하게 전송할 수가 있다. 또한 단말기에서 코덱의 조건도 서로 다르다. PC 환경에서 소프트 폰은 고품질 고덱을 처리하가에 충분한 계산량을 처리할 수 있지만, 복잡한 계산을 처리하기 위해 별도의 DSP를 사용하는 단말기에서는 더 많은 비용이 소요된다. 이와 같이 다양한 응용분야에서 상호 호환되는 코덱을 적용하기 위해서는 계층적 가변 비트열구조초 비트열 화장성을 제공할 필요가 있다. 현재 VoIP에서 가장 널리 사용되는 협대역 코덱은 G.729A이다. 기존의 VoIP 시스템을 유지하고 서로 다른 코덱간의 트랜스코딩을 피하기 위해서는 기존의 G.729 비트열과 호환되는 코덱이 요구된다. G.729.1는 G.729 비트열을 기본 계층으로 두고, 그 위에 비트열 계층을 쌓아가면서 고품질의 음질을 제공하는 구조이다. G.729.1 비트열 형식은 그림 1과 같다. G.729.1 코덱은 CELP, TDBWE, TDAC로 세 개의 모듈로 구성된다. CELP 모듈은 8~12 kbit/s에서 4 kHz까지의 저대역 신호를 생성하고, TDBWE 모듈은 14 kbit/s의 압축률로 고대역 신호를 생성한다.

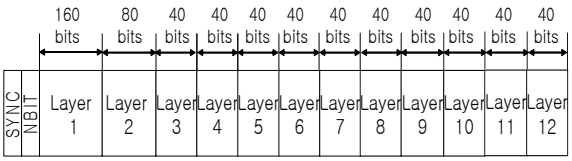

그림 1. G.729.1의 비트열 형식

저대역 및 고대역 신호의 음질을 개선하기 위해 16 kbit/s에서 32 kbit/s까지의 2 kbit/s 단위의 fine granularity를 제공하도록 설계된 TDAC 모듈을 사용한다. 이 모듈은 50~4000 Hz의 입력 신호, CELP 모듈에서 재합성한 신호 사이의 차이, 그리고 4000~7000 Hz의 입력 신호를 MDCT 영역에서 부호화한다. CELP 모듈은 10 ms마다 동작하며 20 ms의 G.729.1의 프레임을 생성하기 위해 CELP 모듈은 한 프레임에 대해 두 번의 동작을 수행하게 된다 [3][4].

2.1 G.729.1 부호화 알고리즘

G.729.1 코덱은 기본 계층인 G.729 코덱과 저대역 신호의 양자화 오차를 양자화하는 SNR scalable 구조, 대역폭 확장 기술 및 저대역과 고대역 신호로 나누어 분석하는 대역분할 방식이 혼재되어 구현되어 있다. 표 1은 G.729.1 코덱의 특성을 정리하였다.

G.729.1 부호화기는 기본적인 동작은 20 ms단위로 동작하고 슈퍼프레임(super frame)이라고 정의한다. 결과적으로 G.729.1 코덱의 한 프레임단위에서 10 ms로 동작하는 G.729는 두 번을 동작하게 된다. 320 샘플단위로 입력되는 광대역 신호는 64 탭을 가지는 QMF (Quadrature Mirror Filterbank) 필터를 통화하여 저대역 신호와 고대역 신호로 분리된다. 저대역 신호는 50 Hz cut-off 주파수를 가지는 고대역 통과 필터를 통해 Layer 1 (G.729)의 목적 신호를 만든다. Layer 1의 LP (Linear prediction) 잔여신호의 양자화 오차는 함성필터를 통과시켜 Layer 2의 목적 신호를 만

든다. Layer 2에서는 tri-pattern 구조의 펄스를 G.729의 고정 코드북과 같은 위치에서 검색한다. 즉, 한 번의 검색 펄스는 3개의 위치를 가지고 중심의 펄스 코드북은 G.729 고정 코드북과 같은 위치에서 ±1의 크기를 가지고 ±1의 위치에서 부가적인 펄스는 신호의 특성에 맞게 적응적인 크기를 갖는다.

고대역 신호를 부호화하기 위하여, 전처리된 저대역 신호와 CELP 모듈에 의해 재합성된 신호의 차이는 인지가중필터를 거친다.

표 1. G.729.1 코덱의 특성 정리

표본화율	8 kHz /16 kHz			
계층	기본계층	향상계층		
전송률 (kbit/s)	8	12	14	16-32
모드	Layer 1	Layer 2	Layer 3	Layer 4-16
방식	G.729A	ACELP	TD-BWE	TDAC
Bandwidth	NB		WB	
지연 (ms)	25		48.9375	
프레임크기 (ms)	10		20	
음질	G.722 56 kbit/s 이상			
복잡도	37 WMOPS 이하			
VAD/CNG	있음			

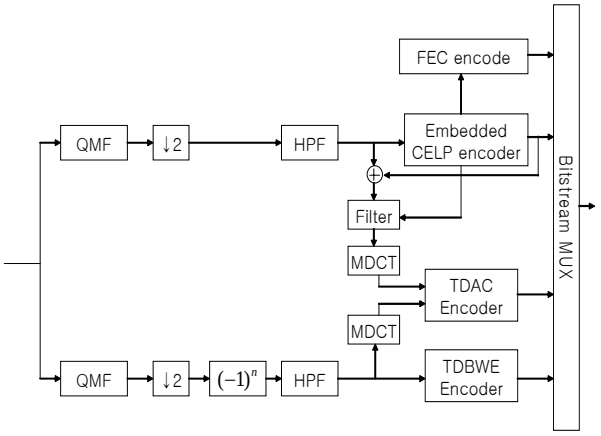

그림 2. G.729.1 코덱의 부호화기 구조

이 때 차이 신호와 고대역 입력 신호 사이에 스펙트럼 연속성을 보장하기 위해서 이득 보상 과정을 거친다. 이 신호는 MDCT를 사용하여 주파수 영역으로 변환된다. 한편 고대역 입력신호는 2:1 비율로 decimation을 수행하여 주파수 대칭을 시킨다. 이 때 3 kHz 이상 성분은 저대역 통과 필터에 의해 제거된다. 전처리된 신호는 TDBWE 모듈로 부호화되고, MDCT에 의해 주파수 영역으로 변환된다. 저대역과 고대역의 MDCT 계수들은 TDAC 모듈로 부호화되며 프레임 손실 발생시 음질 저하를 막기 위한 파라미터 또한 전송된다. 그림 2는 G.729.1 코덱의 부호화기의 구조이다.

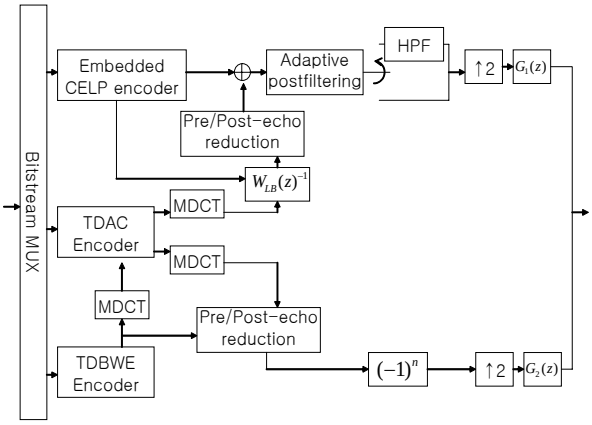

그림 3. G.729.1 코덱의 복호화기 구조

2.2 G.729.1 복호화 알고리즘

G.729.1 디코더는 수신된 비트율과 레이어의 수에 따라 다음과 같이 동작한다. 부호화 알고리즘과 마찬가지로 CELP, TDBWE, TDAC 세 개의 모듈로 구성된다. 우선 8~12 kbit/s에서는 CELP 모듈로, 14 kbit/s에서는 TDBWE 모듈로, 16 kbit/s 이상에서는 TDAC 모듈로 복호화를 수행한다. 8~12 kbit/s에서는 저대역 신호가 복원되고 14 kbit/s에서는 TDBWE 모듈로 고대역 신호를 발생시킨다.

이 신호는 3 kHz 이상의 신호를 0으로 놓고 MDCT를 사용하여 주파수 영역으로 변환시킨다. 16 kbit/s 이상에서는 TDAC 모듈로 저대역에서 인지 가중된 차이 신호와 고대역 신호를 재합성한다. TDAC 모듈에서 전송 받지 못한 대역은 TDBWE 모듈에서 생성한 신호로 대체한다. 이 때 스펙트럼 연속성을 위해 레벨 조정을 수행한다. 그림 3은 전체적인 G.729.1 부호화기 구조도를 나타낸다. 특히, G.729.1에서의 TDBWE라고 불리는 대역폭 확장 방법은 인코더에서 시간과 주파수의 envelope 정보만 전달하고 그림 4와 같이 디코더에서 저대역 신호로부터 신호의 주기적인 성질을 분석하여 최적의 고대역 여기신호를 자체적으로 만들어 사용하는 것이다. 저대역 신호의 유성음/무성음 여부를 분석하여 각각의 성분 가중치를 구하고, 무성음 성분 신호는 불규칙 신호를 사용하고, 유성음 성분 신호는 저대역 신호로부터 고해상도 피치 주기를 검색하고 해당 피치 주기를 가지는 glottal pulse를 자체적으로 생성하여 사용한다. 다음 전송된 T/F (Time/Frequency) envelope을 여기신호에 적용하여 고대역 신호를 최종 합성하고, 마지막으로 강한 펄스 성분을 제거하기 위하여 크기를 감소시키는 동작을 수행한다.

G.729.1은 다운샘플링 후에 folded된 고대역 정보를 다시 원래대로 unfold한 후 모든 동작을 수행하므로 저대역의 피치 하모닉이 고대역에 정확하게 복사된다.

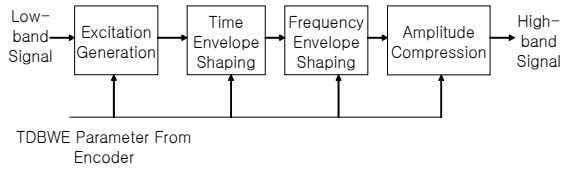

그림 4. G.729.1 코덱의 TDBWE 복호화기의 구조

3. G.729.1의 ARM 프로세서에의 실시간 구현

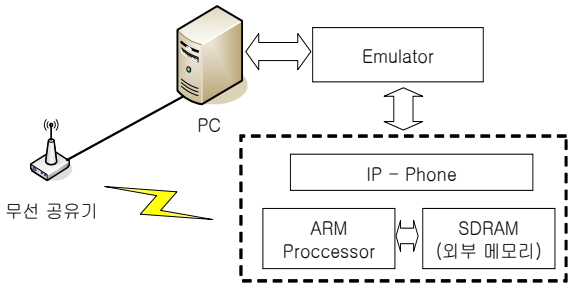

그림 5. 구현된 시스템 구성도

VoWiFi 단말기의 하드웨어 플랫폼으로는 ARM926EJ 프로세서가 탑재된 i.MX21S를 사용하였다. ARM926EJ 프로세서는 저전력, 고성능 32 bit RISC 마이크로프로세서와 자바 가속을 위한 기술이 포함되어 있다. 또한, 저전력으로 고성능을 구현하는 QDSP4000 DSP 코어가 집적되어 있으므로, 모바일 휴대용 단말기에 많이 쓰이고 있다 [5]. ARM926EJ의 프로세서는 프로그램의 명령어를 메모리부터 패치하여 디코드하고 수행하는 기능을 한다. 프로그램과 데이터를 저장하는 메모리는 롬(ROM)과 램(RAM)으로 구성되어 있으며, 입출력 장치는 센서, 직렬 포트(serial port), 병렬 포트(parallel port)로 되어 있다. ARM 코더의 구조는 일반적인 프로세서의 기본 구조와 동일하게 레지스터, ALU(Arithmetic Logic Unit), 제어 장치, 명령어 해석기와 내부에서 서로 정보를 교환하기 위한 데이터 경로로 구성된다. 명령어 해석기는 입력되는 명령을 해석하고, 제어 장치는 필요한 제어신호를 내부 및 외부로 구동하는 역할을 한다. ALU는 32 bits 산술 및 논리 연산을 수행하는 곳으로 레지스터 뱅크로부터 2개의 내부 버스가 연결되어 있고 연산 결과를 레지스터 뱅크 및 어드레스 레지스터로 저장하기 위한 ALU 출력 버스가 존재한다.

전제 시스템 구성도는 그림 5와 같다. 전체적인 흐름을 살펴보면 IP-Phone의 프로세서는 ARM926EJ 프로세서가 탑재되어 있고, 운영체제는 리눅스 기반이다. 이렇게 구성된 IP-Phone에 wireless LAN 혹은 에뮬레이터를 통한 PC와 IP-Phone간의 통신이 이루어지고, IP-Phone에서 인코딩된 결과를 wireless LAN을 통해 PC에 전송하게 되면, PC는 수

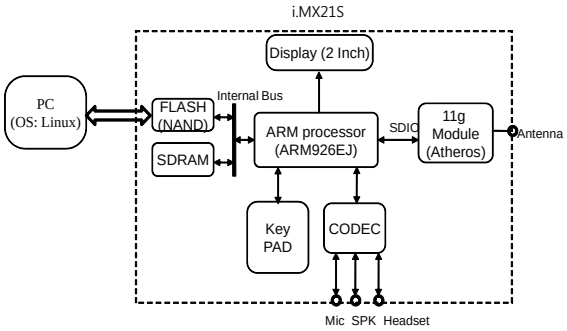

그림 6. i.MX21S 플랫폼으로 설계된 VoWiFi 단말기 내부 구조.

신된 비트스트림을 디코딩하여 스피커로 출력하게 된다. 그림 6은 이에 대한 과정 및 하드웨어 구조를 나타낸 것이다. G.729.1 코덱을 ARM926EJ 프로세서에 구현하기 위해, 우선 G.729.1 인코더를 최적화하여 구현하고, 이를 ARM926EJ에 적용하기 위해 크로스 컴파일러를 이용하여 컴파일을 수행한다. 이 후, 호스트 PC와 i.MX21S 보드를 연결하는 시리얼 포트를 통해 보드에 업로드한다. 또한 네트워크 전송을 위해 무선 랜을 설정한 후, 마이크 장치를 열어준 다음, G.729.1 인코더를 실행시킨다. 한편, PC에서는 스피커 장치를 열어준 후, G.729.1 디코더를 실행시킨다. 그림 6은 이에 대한 과정 및 하드웨어 구조를 나타낸 것이다. i.MX21S 보드의 마이크 장치에 입력 신호를 넣어주면, 인코더는 마이크의 입력버퍼에서 데이터를 읽은 후 실시간으로 인코딩을 수행한다. 인코더 과정이 완료되면 무선 랜을 통해 비트스트림이 호스트 PC로 전송되어, PC는 비트스트림을 디코딩해서 실시간으로 스피커 출력을 하게 된다.

4. G.729.1코덱의 최적화

G.729.1 코덱을 최적화하기 위한 방법으로 C-코드 최적화를 수행하였다. 기본적인 연산을 수행하기 위해서 오버플로우가 발생하는 것을 확인하는 함수가 자주 사용된다. 이와 같이 함수 호출이 잦아지면 CPU에 부하가 많이 걸려, 연산 속도가 늦어진다. 표 2와 같이 사용이 잦은 함수에 대해 __inline 선언을 하게 될 경우, 함수를 호출하는 것이 일종의 매크로처럼 작용하며, 함수가 호출되는 대신 함수의 본체가 직접 치환되어 버린다. 따라서 함수를 호출하는데 드는 비용을 줄일 수 있게 된다. 하지만 코드가 모두 치환되어 버리면 코드의 사이즈가 커지게 되므로, 적절히 사용해야 한다.

또한 기본 연산 중 2의 배수에 해당하는 곱셈 혹은 나눗셈의 경우 시프트 연산을 적용함으로써 최적화를 수행하였다. 곱셈의 경우 2 cycle이 소요되는 반면, 시프트연산은 1 cycle이 소요되므로 최적화가 가능하게 되었다.

또한 루프를 unrolling 과정을 수행함으로써 최적화를 수

행했다 unrolling 과정은 루프를 푸는 과정으로 일반적으로 2의 배수 혹은 4의 배수로 for 루프를 풀어버리는 것이 일반적이다. 표 3과 같이 루프를 풀어 버릴 경우, 루프 안에 코드가 실행되고 난 후, 비교하게 되는 코드의 횟수를 줄일 수 있으므로, 최적화 과정을 수행할 수 있었다.

표 2. __inline함수 사용 예

적용 전	add(x, y) sub(x, y)
전용 후	__inline add(x, y) __inline sub(x, y)

표 3. for 루프 unrolling 사용 예

적용 전	for(i=0; i<100; i++)
전용 후	for(i=0; i<100; i+=4)

표 4. 기본 연산 함수 최적화의 예

적용 전	b=norm_s(a); c=shl(a, b);
전용 후	b=norm_s(a); c=(a<<b);

2의 배수 혹은 4의 배수로 할 수 없는 경우나 for 루프의 횟수가 적은 경우에는 ARM 프로세서의 특성을 고려하여, for 루프의 인덱스를 감소시키는 방향으로 루프를 종료시키기 위한 검사는 항상 count-down-to-zero 방식을 사용하여 최적화를 수행하였다.

사용되는 변수가 음수가 되지 않는 값이라면, 데이터 형태를 unsigned로 선언해서 사용해야 한다. ARM 프로세서는 int 형 대신에 unsigned int형의 연산이 더 빠르기 때문에 절대값이나 for 루프에 사용되는 인덱스 값도 unsigned 를 명시하여 최적화를 수행하였다.

또한, 기본 연산 함수들을 오버플로우를 항상 확인하게 되어 있다. 하지만 오버플로우가 어떠한 경우에도 발생하지 않는 경우에, 오버플로우를 확인하는 함수를 호출하는 것은 성능을 저하시킨다. 이와 같이 알고리즘을 분석하여 오버플로우가 발생하지 않는 부분을 찾아 그 부분의 기본 연산 함수를 기본 연산으로 대체시켰다. 표 4는 정규화 시키기 위한 기본 연산 함수의 최적화의 예이다. 이와 같이 최적화를 수행하게 되면 shl() 함수를 호출하는 대신 직접 시프트 연산을 수행함으로써 최적화를 수행하였다.

또한, shr 경우 음의 정수가 아닌 양의 정수를 시프트시키는 것이라면, 직접 시프트 연산을 수행함으로써 최적화를 수행하였다.

5. 성능 평가

본 논문에서는 성능 평가를 위해 32초에 해당하는 16 kHz의 광대역 음성을 이용하였으며, 최적화 전 후의 G.729.1의 인코더와 디코더의 수행시간을 비교하였다. 표 5는 G.729.1의 실시간 구현에 대한 성능 시험 조건이다.

표 5. G.729.1의 실시간 구현에 대한 성능 실험 조건

PC	ARM 프로세서	Test File
CPU: Intel Pentium 4 3 GHz RAM: 1 GB	최대 동작 주파수 :200 MHz I-Cache: 16 KByte D-Cache: 16 KByte	sampling rate: 16 kHz bit rate: 32 kbit/s size: 1600 frame

표 6. PC에서 G.729.1 encoder 수행시간 (sec).

G.729.1 최적화 전	G.729.1 최적화 후	감소율
6.735	2.360	64.94%

표 7. pc에서 G.729.1 decoder 수행시간 (sec).

G.729.1 최적화 전	G.729.1 최적화 후	감소율
3.421	2.149	38.3%

표 6과 7은 PC에서 각각 G.729.1 인코더와 디코더의 CPU 동작 시간을 G.729.1을 최적화하기 전과 최적화한 후 측정한 결과를 각각 나타낸 것이다. 표에서 보는 바와 같이, 인코더인 경우 최적화 결과 약 64.94%의 수행시간 감소를 보였으며, 디코더는 약 38.35%의 수행시간 감소를 보였다. 또한 실험 측정에 사용된 파일이 32초의 음성 데이터이므로, G.729.1을 인코딩하는 데는 7.38%, 디코딩은 6.59%의 CPU 시간을 사용하게 되어 full-duplex로 동작할 경우에 14.22%의 시간이 걸려 실시간으로 동작함을 알 수 있다.

표 8. ARM프로세서의 G.729.1 encoder 수행시간 (sec).

G.729.1 최적화 전	G.729.1 최적화 후	감소율
65.23	31.40	51.86%

표 9. ARM프로세서의 G.729.1 decoder 수행시간 (sec).

G.729.1 최적화 전	G.729.1 최적화 후	감소율
35.32	26.03	26.30%

표 8과 9는 ARM 프로세서인 경우, CPU 동작 시간을 최적화 수행 전과 후를 비교한 것이다. ARM 프로세서에서의 구현을 위해 크로스 컴파일 옵션은 -Wall -O3 -march=armv4 -mtune=arm9tdmi -fomit-frame-point 으로 설정하였다. -Wall 옵션은 프로그램의 warning을 보여주는 옵션이다. -O3 옵션은 파일 사이즈는 증가하지만, 프로그램의 수행 속도를 최적화시켜주는 옵션이다. -march=armv4는 ARM 프로세서 아키텍처의 버전을 의미하고, -mtune=arm9tdm는 ARM 프로세서의 코어 버전을 의미한다. 마지막으로, -fomit-frame-point 옵션을 쓰면 frame pointer가 필요없는 함수들을 컴파일할 때 frame pointer를 생략하게 된다. 따라서 frame pointer를 저장하고,

다시 불러오는 부분에 해당하는 기계어를 만들지 않기 때문에 약간의 성능 향상을 기대할 수 있다. ARM 프로세서에서의 수행 시간 확인은 Linux에서 제공하는 time 명령어를 이용하여 수행 시간 확인을 하였다. 인코더와 디코더를 최적화되기 전에 ARM 프로세서에서 수행 시간은 각각 65.23초와 35.32초로 32초를 초과하므로 실시간 동작이 안 되지만, 최적화를 수행한 결과, 32초 이내로 동작하므로 실시간 동작이 되는 것을 확인할 수 있다.

6. 결론

본 논문에서는 VoWiFi 시스템에서의 음질을 개선하기 위해, ITU-T에서 표준화된 광대역 음성 부호화기인 G.729.1 코덱을 ARM 프로세서 구현하였다. 이를 위해 C-코드 최적화와 알고리즘의 최적화를 사용하여 G.729.1 코덱을 최적화 과정을 수행하였다. 또한 실험 측정은 PC와 ARM 프로세서에서 측정하였다. 먼저, PC상에서 성능 측정한 결과 인코더 및 디코더는 약 64.94% 및 38.35%의 수행시간 감소의 성능 개선을 보였고, ARM 프로세서 상에서 51.86% 26.30%의 수행시간 감소의 성능을 개선함으로써 실시간으로 동작함을 확인할 수 있었다.

감사의 글

이 연구는 광주과학기술원 과학기술응용연구소 (GTI)의 실용화 연구개발비 지원으로 수행되었습니다.

참고문헌

[1] 함창용, 곽정호, 맹승찬, 나상호, 천병준, "인터넷전화(VoIP) 시장의 국·내외 현황 및 시사점," *KISDI 이슈리포트*, 제2007권, 제13호, pp. 3-46, 2007년.

[2] A. F. da Conceicao, J. Li, D. A. Florencio, and F. Kon, "Is IEEE 802.11 ready for VoIP?," in *Proc. IEEE International Workshop on Multimedia Signal Processing*, pp. 108-113, Oct. 2007.

[3] ITU-T Recommendation G.729.1, *An 8-32 kbit/s scalable wideband coder bitstream interoperable with G.729,* May 2006.

[4] 김현우, 성종모, 이미숙, 김도영, 정해원, "G.729.1 광대역 멀티코덱 표준 기술 동향," *전자통신동향분석*, 제21권, 제6호, pp. 80-81, 2006년 12월.

[5] Y.-S. Choi and G.-S. Lee, "Real-time implementation of G.723.1A/G.729AB on a RISC processor for personal IP telephony devices," in *Proc. of the 9th IEEE International Symposium on Consumer Electronics,* pp. 20-24, Macao, June 2005.