ARM926EJ-S 프로세서 코어를 이용한 G.729.1의 실시간 구현

정회원 소 운 섭*, 김 대 영**

Real-Time Implementation of the G.729.1 Using ARM926EJ-S Processor Core

Woon Seob So*, Dae Young Kim** *Regular Members*

요 약

본 논문에서는 ITU-T의 SG15에서 채택된 G.729.1 광대역 음성 코덱을 ARM926EJ-S® 프로세서 코어에 적용하기 위해 기본연산자 및 산술기능 함수를 포함한 G.729.1 코덱 프로그램 일부를 어셈블리어로 변환하여 실시간으로 동작하도록 구현한 절차 및 결과를 기술하였다. G.729.1은 8~32kbps의 가변 전송률을 갖는 ITU-T 표준 광대역 음성 코덱이며, 입력신호는 8kHz 또는 16 kHz로 샘플링 되어 샘플 당 16 비트로 양자화된 PCM 신호를 입력 받는다. 이 코덱은 앞서 표준화된 G.729 및 G.729A와 상호 호환이 가능하며 음질 향상을 위해 기존의 협대역(300~3,400Hz)에 비해 대역폭을 광대역(50~7,000Hz)으로 확장한 버전이다. 실시간으로 구현된 G.729.1 광대역 음성 코덱은 32kbps에서 인코더와 디코더 부분이 각각 약 31.2 MCPS 및 22.8 MCPS의 복잡도를 가지며, 실제 임베디드 시스템에서의 실행 시간은 인코더와 디코더 평균 6.75ms와 4.76ms로 총 11.5ms가 걸렸다. 또한 이 코덱은 ITU-T에서 제공하는 모든 테스트 벡터에 대해 비트 단위로 정확하게 시험하여 통과하였으며, 실제 인터넷 전화기에 적용한 실시간 음성통화에서 정상적으로 동작하였다.

Key Words : Wideband speech codec, G.729.1, ARM926EJ-S, ADS, Real-time implementation

ABSTRACT

In this paper we described the process and the results of real-time implementation of G.729.1 wideband speech codec which is standardized in SG15 of ITU-T. To apply the codec on ARM926EJ-S® processor core. we transformed some parts of the codec C program including basic operations and arithmetic functions into assembly language to operate the codec in real-time. G.729.1 is the standard wideband speech codec of ITU-T having variable bit rates of 8~32kbps and inputs quantized 16 bits PCM signal per sample at the rate of 8kHz or 16kHz sampling. This codec is interoperable with the G.729 and G.729A and the bandwidth extended wideband(50~7,000Hz) version of existing narrowband(300~3,400Hz) codec to enhance voice quality. The implemented G.729.1 wideband speech codec has the complexity of 31.2 MCPS for encoder and 22.8 MCPS for decoder and the execution time of the codec takes 11.5ms total on the target with 6.75ms and 4.76ms respectively. Also this codec was tested bit by bit exactly against all set of test vectors provided by ITU-T and passed all the test vectors. Besides the codec operated well on the Internet phone in real-time.

Ⅰ. 서 론

최근 인터넷 및 통신망 기술의 발달과 더불어 홈 네트워크, 초고속망, 이동통신망 등에서 고품질의 음성통신 서비스 요구가 증가하게 되었다. 특히 급속히 보급된 초고속 인터넷 서비스는 전용의 데이터

※ 본 연구는 본 연구는 지식경제부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발 사업의 일환으로 수행하였음.
* 한국전자통신연구원 멀티미디어통신연구팀(wsso@etri.re.kr), ** 충남대학교 전자전파정보통신공학과(dykim@cnu.ac.kr)
논문번호 : KICS2008-06-257, 접수일자 : 2008년 6월 4일, 최종논문접수일자 : 2008년 7월 16일

통신 기능에서 벗어나 지금까지 서로 다른 통신통신망에서 별개의 서비스로 제공되어 왔던 전화와 데이터 통신, 방송의 3가지 서비스를 광대역통합망과 같은 차세대 통신망에서 인터넷 서비스에 통합하여 제공하기 위한 연구 개발이 활발히 진행되어 왔다. 인터넷 전화의 경우 기존의 공중전화망에 비해 저렴한 통신 비용, 운영 및 관리의 효율성, 다양한 부가서비스스 제공 등으로 더욱 확산되고 있는 추세이다. 이러한 경향에 따라 고객들은 보다 향상된 품질의 음성통화 기능을 요구하고 있으며, 이러한 요구 조건에 부응하기 위해 ITU-T, 3GPP, 3GPP2 등 각 표준화 단체에서는 광대역 음성 코덱의 표준화를 추진 해 오고 있다.

2006년 5월에 ITU-T에서 표준화 된 G.729.1[1] 광대역 음성코덱은 8~32kbps의 가변 전송률을 가지며 50~7,000Hz의 대역폭을 갖는 표준 코덱이다. 이는 1995년 11월에 표준화되어 G.729[2] 코덱으로 알려진 8kbps CS-ACELP(Conjugate Structure Agebraic CELP)를 기본으로 하고 있어 G.729 및 G.729A와 상호 호환이 가능하며, VoIP 포럼 및 TTA에서 고품질 인터넷 전화기의 선택적인 광대역 음성 코덱으로 규정되어 있다. G.729 협대역 음성 코덱은 피치 및 코드북의 효율적인 탐색구조를 첨가하여 계산량을 줄이면서 성능을 극대화한 코덱이다. G.729A[3]는 1996년 11월에 표준화가 되었으며, 상호 호완성이 있는 50%정도 복잡도가 낮은 코덱으로 VoIP 프로토콜인 SIP(Session Initiation Protocol)에서 널리 사용되고 있다.

ARM926EJ-S® 프로세서 코어[4]는 ARM사에서 개발한 ARM9 계열 프로세서 중에서 DSP의 확장 기능을 가지는 16/32 비트 범용 RISC 프로세서 코어이며 낮은 전력소모와 높은 성능을 가진다. 이 프로세서 코어를 사용한 상용의 프로세서는 CMOS 공정을 사용하여 i.MX21의 경우 350MHz까지 i.MX27의 경우 400MHz까지의 동작 주파수를 가지며 MMU를 보유하고 있다[5]. 이러한 프로세서는 단일 프로세서에 DSP 확장 명령 세트를 가지고 있어서 음성 코덱 프로그램 처리에 적합하며 하나의 운영체제 위에서 시스템 전체적인 프로그램 제어 및 운용이 용이한 이점이 있다.

본 논문에서는 저렴하고 효과적으로 광대역 음성 코덱을 구현하기 위하여 최근 널리 사용되고 있는 ARM926EJ-S 프로세서 코어를 선택하였으며 여기에 탑재하여 사용하기 위한 코덱을 실시간으로 구현하였다. 향상된 음성 품질을 구현하기 위해 최근 ITU-T에서 새롭게 표준화된 G.729.1 광대역 음성 코덱을 사용하였으며, 임베디드 시스템에서 구동할 수 있도록 고정 소수점 형태의 기본연산자 및 산술기능 함수를 포함한 C 프로그램의 일부를 어셈블리 언어로 변환하여 프로세서의 계산량을 최소화 하도록 하였고, 그 결과 음성 코덱의 실행시간을 최적화 전보다 80% 이상 줄일 수 있었다. 기존의 최적화 과정을 살펴보면 먼저 음성 코덱을 DSP에서 실시간으로 구현하기 위해 함수를 인라인화 하고 최적의 레지스터 할당을 하는 방법이 있고[6], 추가로 프로파일러를 사용하여 C 코드의 WMOPS를 계산하여 최적화 블록을 찾아내어 최적화하고 적절한 메모리 분할을 하는 방법[7]이 있다. 또한 RISC 프로세서에서 실시간으로 구현하기 위해서 메모리 로드, 산술연산, 루프를 최적화하는 방법[8]이 있다.

본 논문의 구성은 서론에 이어 II장에서는 G.729.1 광대역 음성 코덱의 기본 구조와 특성에 대하여 살펴보고, III장에서는 ARM926EJ-S 코어 에서의 실시간 구현에 대해 설명한다. IV장에서 임베디드 시스템에서의 시험 및 결과에 대해 기술하고, V장에서 결론을 맺는다.

Ⅱ. G.729.1 광대역 음성 코덱

2.1 G.729.1 코덱 구성

G.729.1은 G.729 협대역 음성 코덱을 확장한 8~32kbps의 가변 전송률을 갖는 스케일러블 광대역 코덱이다. 기본적으로 인코더 입력과 디코더 출력 신호는 16 kHz로 샘플링 되며, 8kHz의 샘플링도 지원한다. 입력 신호는 20ms 단위로 샘플링 된 16비트 PCM 신호이며, 출력 신호는 8kHz 또는 16kHz의 샘플링 주파수를 갖는 16비트 PCM 신호이다. 인코더에 의해 만들어진 비트스트림은 스케일러블 하며 계층 1에서 12까지 12개의 임베디드 계층을 갖는다. 계층 1은 8kbps를 갖는 핵심 계층으로서 G.729 비트스트림과 호환된다. 계층 2는 협대역 향상 계층으로 4kbps를 더하여 12kbps가 되며, 계층 3부터 12는 광대역 향상 계층으로 2kbps씩을 더해 나가 32kbps까지 이른다.

이 코덱은 CELP(Code Excited Linear Prediction), TDBWE(Time Domain Bandwidth Extension), TDAC (Time Domain Aliasing Cancellation)의 세 개 모듈로 구성된다. CELP 모듈은 8, 12kbps에서 50~4,000Hz의 계층 1, 2 협대역 신호를 생성하고, TDBWE 모듈은 14kbps에서 50~7,000Hz의 광대역 신호를

생성한다. 신호의 음질을 개선하기 위해 16~32kbps (2kbps 단위)에서 TDAC 모듈을 사용한다. 이 모듈은 저대역(50~4,000Hz)의 입력 신호와 CELP 모듈에서 재합성한 신호 사이의 차이 및 고대역(4000~7000Hz)의 입력 신호를 MDCT(Modified Discrete Cosine Transform) 영역에서 부호화 한다. CELP 모듈은 G.729와 마찬가지로 10ms마다 동작한다. G.729.1의 수퍼프레임 길이는 20ms이므로 CELP 모듈은 한 수퍼프레임에 두 번 프레임이 동작한다.

2.2 G.729.1 인코더

G.729.1은 기본적으로 분석/합성 방식의 CELP구조를 이용하며 인코더의 기본 구조는 그림 1과 같다. 먼저 입력 신호를 QMF(Quadrature Mirror Filterbank) 필터를 사용하여 2개의 서브밴드인 고대역과 저대역 신호로 분리한다. 저대역 신호는 2만큼 다운 샘플링을 한 후 50Hz 이하 주파수 성분을 제거하기 위해 고대역 통과 필터를 거쳐 전처리 한다. 이 신호는 임베디드 CELP 인코더에 의해 8, 12 kbps로 부호화 된다. 전처리 된 저대역 신호와 임베디드 CELP 인코더에 의해 재 합성된 신호의 차이 신호는 인지가중 필터를 거친다. 이 때 차이 신호와 고대역 입력 신호 사이에 스펙트럼 연속성을 보장하기 위해서 인지가중 필터에서 이득 보상 과정을 포함한다. 인지가중 필터의 출력 신호는 MDCT에 의해 주파수 영역으로 변환된다. 한편 고대역 입력 신호는 2만큼 다운 샘플링을 하고 주파수 대칭을 시킨 다음 저대역 통과 필터를 거쳐 3kHz 이상의 주파수 성분을 제거한다. 전처리 된 이 신호는 TDBWE 인코더의 입력이 되고, MDCT에 의해 주파수 영역으로 변환된다. 저대역과 고대역으로 부터의 MDCT 계수들은 TDAC 인코더에서 부호화 된다. 이 외에 프레임 손실이 발생시 음질을 개선하기 위해 몇 개의 파라미터가 FEC(Frame Erasure Concealment) 인코더에 전송되어 부호화 된다.

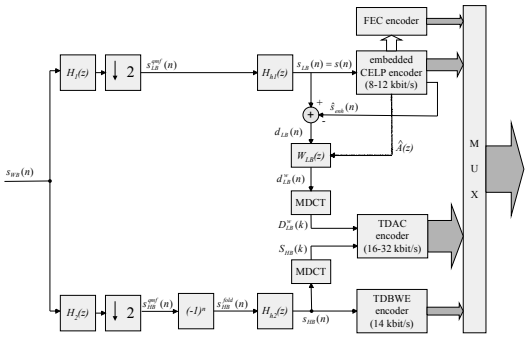

그림 1. ITU-T G.729.1 인코더 구조

2.3 G.729.1 디코더

인코더와 마찬가지로 디코더는 CELP, TDBWE, TDAC 세 개의 모듈을 사용하여 디코딩 한다. 디코더의 구조는 그림 2와 같고, 수신된 비트율에 따라 처리된다. 8, 12kbps에서는 임베디드 CELP 디코더로, 14kbps에서는 TDBWE 디코더로, 16kbps 이상에서는 TDAC 디코더로 디코딩 한다. 8kbps 에서는 임베디드 CELP 디코더에 의해 핵심 계층이 디코딩 되고, 12kbps에서는 임베디드 CELP 디코더에 의해 핵심 계층과 협대역 향상 계층이 디코딩 된다. 그리고 QMF 합성 필터뱅크는 고대역을 0으로 합성하여 저대역(50~4,000Hz) 신호를 출력한다. 14kbps에서는 TDBWE 디코더를 사용하여 고주파 합성신호를 만들어 낸다. 이 신호는 3kHz 이상의 신호는 0으로 놓고 MDCT를 사용해서 주파수 영역으로 변환되며, 다시 역 MDCT를 거쳐 시간 영역으로 변환되어 고주파 합성신호에 겹쳐서 더해진다. 이때 QMF 합성필터 에서는 재구성된 고대역 신호와 고역통과 필터를 거치지 않은 저대역 신호를 결합한다. 14kbps이상에서는 TDAC 디코더로 저대역에서 재구성된 가중 차이 신호와 고대역(4,000~7,000Hz) 신호를 재구성한다. TDAC 디코더에서 수신되지 않은 부대역들과 0비트로 할당된 부대역들은 TDBWE 디코더에서 생성된 레벨이 조정된 부대역 신호로 결합되고 업 샘플링 된다.

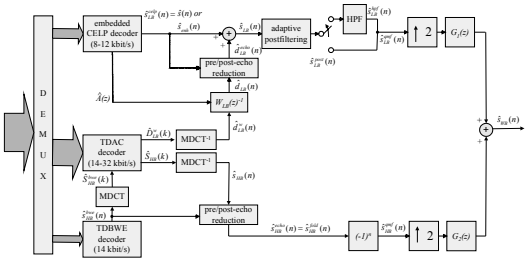

그림 2. ITU-T G.729.1 디코더 구조

Ⅲ. ARM926EJ-S 코어에서의 실시간 구현

3.1 코덱 성능 분석

실시간 구현 이전에 타겟이 되는 코덱의 계산량이 많이 소요되는 부분과 그렇지 않은 부분을 구분하여 최적화가 집중적으로 이루어져야 할 부분을 찾아내는 성능 분석이 필요하다. 일반적으로 많이 사용되는 성능 분석 방법은 코덱에서 산술 계산량

및 제어에 대해 가중치를 두어 계산량을 추정하는 WMOPS(Weighted Million Operations Per Second)를 계산하는 방법, 소스 코드에 대한 프로파일 정보를 이용하는 방법, 그리고 최적화된 코드를 실제 타겟 시스템에 탑재하여 수행 시간을 측정하는 방법 등이 있다. WMOPS는 최적화 이전의 계산량에 대한 추정의 의미가 더 있으며, 프로파일 정보는 함수의 호출회수와 함수의 실행시간에 대한 정보를 얻을 수 있으며, 실제 타겟 시스템에서 수행 시간을 측정하는 것은 최종 코드가 만들어 지기 전까지 사용하기에는 시간이 많이 소요된다. 따라서 코드 최적화 이전에는 WMOPS 계산과 프로파일 정보를 분석하고, 코드 최적화 이후에는 수행 시간을 측정하는 것이 필요하다.

WMOPS는 코덱의 C/C++ 코드 내에서 사용되고 있는 산술연산 및 제어 연산에 대해 가중치를 두어 한 프레임을 처리하는 동안 이들 연산의 값이 얼마인지를 알려주는 지표이다[9]. 이는 배포된 코덱에서 더 이상의 알고리즘의 개선이 필요 없고 WMOPS를 구하는 부분에서의 튜닝이 필요 없다고 가정한다면 실제 타겟 시스템에 코덱을 탑재하였을 때 성능은 추정된 WMOPS 값 이하로 도달할 수 없다[10]. 따라서 WMOPS의 의미는 실제 코덱을 탑재하고자 하는 타겟 시스템의 프로세서를 선정하는 가이드라인으로 이용된다. 또한 최적화 측면에서는 실제 최적화 작업이 이루어지기 전에 계산량이 많은 블록들을 추정할 수 있다. 배포된 G.729.1 코덱은 총 35.79WMOPS이다.

프로파일은 프로그램이 실행되는 동안, 함수의 호출 회수, 실행시간 등을 알게 해 준다. 이러한 정보는 사용자나 개발자가 코드의 어느 부분이 시간을 많이 소비하는지에 관해 알려주기 때문에 너무 크거나 복잡한 소스 코드에 대한 분석에도 사용될 수 있다. 이런 작업을 해주는 프로파일러는 여러 가지가 있지만 개발하고자 하는 시스템이 리눅스 OS 상에서 실행되어지기 때문에 GNU에서 제공하는 GNU 프로파일러인 gprof 를 사용하여 분석한다[11]. gprof 를 실행하면 플랫 프로파일과 콜 그래프를얻는다. 플랫 프로파일은 각 함수에서 얼마만큼의 시간을 소요했는가와 함수가 얼마나 호출됐는가에 관한 정보를 제공하며, 콜 그래프는 각 함수에 대해서 그 함수를 호출한 함수와 그 함수가 호출한 다른 함수들이 소요한 시간 등에 대해서 알려준다.

G.729.1 코덱의 인코더 및 디코더 주요 블록에 대한 WMOPS를 계산하면 각 블록이 전체 프로그램에 대해 차지하는 WMOPS의 비중을 알 수 있어 어느 블록이 계산량을 많이 차지하는지 알 수 있다. 또한 프로파일링 툴을 실행하여 결과를 살펴보면 계산량이 많아서 수행 시간을 많이 소요하는 블록을 알 수 있다. 이러한 결과를 가지고 G.729.1 고정 소수점 코드에 대한 WMOPS 계산 결과와 프로파일 분석 결과를 토대로 정리한 최적화 해야 할 블록들은 표 1과 같고 몇 가지 중요한 사실들을 정리하면 다음과 같다.

(1) 인코더가 디코더보다 WMOPS의 경우 1.57배, 프로파일의 결과로는 약 2.2 배 정도 계산량이 많다. 따라서, 우선적인 코드 최적화는 디코더보다 인코더에 초점을 맞추어야 한다.
(2) 디코더의 경우 8k, 12k 에서 보다 32k에서 상대적으로 계산량이 많이 증가한다.
(3) 인코더의 경우 CELP2S_encoder블록이 전체 전

표 1. G.729.1 최적화 대상 블록

Block		Priority
Encoder	CELP2S_ACELP_code_A_OTH	H
	CELP2S_ACELP_code_2NDLAYER	H
	Pitch_fr3	H
	Pitch_ol	H
	Qua_lsp	H
	Az_lsp	M
	Syn_filt2	M
	Syn_filt	H
	Residu2	M
	AutoCorrLPC	L
	Qua_gain	L
	Pred_lt_3, Int_qlpc	M
	Convolve	L
	TDAC_quantif	H
	TDAC_allocbit	H
	TDAC_mdct	H
Decoder	G729_pst_ltp	H
	TDBWE_frequency_envelope_shaping	H
	TDBWE_generate_excitation	H
	G729_Pred_lt_3	M
	G729_Int_qlpc	M
	TDAC_inv_mdct	H
	TDAC_mdct	H
	MAIN_QMF_syn	M
Others	MAIN_lp3k	M

체 계산량의 의 약 60% 이상을 차지하고, 한다, 그 다음으로 TDAC 블록이 약 25% 정도를 차지하고 있다.

(4) CELP2S_encoder 내에서는 CELP2S_ACELP_code_A_OTH, CELP2S_ACELP_code_2NDLAYER, Pitch_fr3, Pitch_ol 등이 많은 계산량을 차지한다.

(5) G729_Syn_filt, G729_Weight_Az, G729_Residu, G729_Syn_filt2, MAIN_lp3k, G729_Pred_lt_3, TDAC_mdct, TDAC_tfr 등의 블록들은 인코더와 디코더에서 공통으로 사용된다.

3.2 ARM926EJ-S 프로세서 코어의 특징

G.729.1 의 실시간 구현에 사용된 ARM926EJ-S 프로세서 코어는 전형적인 RISC 프로세서이며 고정 소수점 코어이다. 부동 소수점 연산을 위한 VFP (Vector Floating Point) 명령이 제공 되지만 고정 소수점 명령보다 많은 연산량을 필요로 한다. 현재 통신 및 기타 주요 알고리즘들은 많은 신호처리 능력뿐만 아니라 효율적인 제어 능력을 요구하고 있는 추세이며, 이에 따라 ARM사는 기존 ARM 명령 체계에 DSP 처리를 위한 몇 가지의 새로운 명령을 추가한 ARMv5TE 구조의 프로세서를 제공하고 있다[12]. 이러한 구조를 가진 ARM 프로세서들은 ARM926EJ-S, ARM946E-S, ARM966E-S, ARM968E-S 등과 같은 “E”의 확장자를 가진다. 일반적으로 디지털 신호처리 알고리즘 처리에서는 많은 양ㅅ의 곱셈-덧셈 연산이 필요하다. 이와 같이 DSP에서 요구되는 연산들을 효과적으로 처리하기 위해 ARMv5TE 구조에서 제공하는 명령들은 표 2와 같다. 위와 같은 특징을 가진 ARM926EJ-S 프로세서에서 G.729.1 코덱을 실시간으로 구현하기 위하여 기본산술 연산에 있어서 DSP 향상 확장 명령을 사용하여 DSP와 유사한 기능을 처리하도록 인라인 어셈블리 프로그램을 작성하였다.

3.3 인라인 어셈블러 구현

G729.1 의 고정 소수점의 경우 기본 산술연산(+, -, x, /) 및 데이터 처리 연산들이 함수 형태로 되어 있어 이를 그대로 사용할 경우 함수 콜 및 리턴에 따라 실행시간이 상당히 길어진다. 이들 함수들을 소스 코드 내에 C코드 형태로 인라인 시킬 수 있지만 이는 작성된 함수코드 자체가 처리시간을 많이 요구하는 형태로 되어있는 것이 있기 때문에 어셈블리 코드 형태로 인라인을 시키는 것이 성능 향상에 유리하다. 어셈블리 코드 형식의 인라인을 하기 위해서는 타겟 프로세서에서 지원하는 명령들을 파악하여 불필요한 코드 생성을 막는 것이 필요하다. ARM926EJ-S 프로세서의 경우 신호처리 작업을 지원해 주기 위한 효율적인 명령들을 가지고 있다. 기본 기능의 인라인은 먼저 ADSv1.2[13] 상에서 인라인 어셈블리를 작성한 다음 코드의 동작을 확인 한 후 타겟 시스템에 이식하기 위해 이들 인라인 어셈블리 언어를 gcc 인라인 어셈블리로 전환한다. 이들 둘 간의 신텍스는 완전히 다르므로 실제 언어 작성 측면에서는 새롭게 작성되는 것과 같다. gcc 인라인 어셈블리 언어는 gcc 컴파일러의 매뉴얼[14]을 참조하여 작성한다. 이들 인라인 함수 중 쉬프트 관련 연산자는 원시 코드 상에서는 쉬프트 함수의 인자가 음의 값을 가질 경우 그 방향을 바꾸어 쉬프트가 가능하게 되어 있다. 즉, shl 함수의 경우 함수의 쉬프트 횟수의 값이 음의 값을 가지게 되면 이는 쉬프트 라이트 기능을 하게 되어 있다. 따라서 이 부분은 실제 코드상에서 모든 쉬프트 함수가 음의 값을 가지는 것이 아니기 때문에 이 함수를 인라인 어셈블러로 작성 할 때는 양의 인자를 가지는 쉬프트 함수와 그렇지 않은 함수로 구분해서 작성하였다. 즉 원래 쉬프트 레프트 함수인 shl 인 경우에 있어서, 양의 쉬프트 회수만을 가지는 shl의 경

표 2. ARM926EJ-S DSP 향상 확장 명령

Instructions	Operations	Purpose
SMLAxy{cond}	16x16+32-> 32	SignedMAC
SMLAWy{cond}	32x16+32-> 32	SignedMACwide
SMLALxy{cond}	16x16+64-> 64	SignedMAClong
SMULxy{cond}	16x16-> 32	Signedmultiply
SMULWy{cond}	16x32-> 32	Signedmultiplylong
QADDRd,Rm,Rs	SAT(Rm+Rs)	SaturatingADD
QDADDRd,Rm,Rs	SAT(Rm+SAT(RsX2))	SaturatingADDdouble
QSUBRd,Rm,Rs	SAT(Rm−Rs)	SaturatingSUB
QDSUBRd,Rm,Rs	SAT(Rm-SAT(RsX2))	SaturatingSUBdouble
CLZ{cond}Rd,Rm	OOUNTZ(Rm)	Countleadingzeros

표 3. 어셈블러 코드 함수

Function Name	Assembler Function
Log2	Log2.s
Pow2	Pow2.s
div_s	div_s.s
G729EV_G729_AutocorrLSP	G729EV_G729_AutocorrLSP.s
G729EV_G729_Az_lsp	G729EV_G729_Az_lsp.s
G729EV_G729_ Cheb	G729EV_G729_ Cheb.s
G729EV_G729_ Convolve	G729EV_G729_ Convolve.s
G729EV_G729_ Cor_h_X	G729EV_G729_ Cor_h_X.s
G729EV_G729_ Lag_max	G729EV_G729_ Lag_max.s
G729EV_G729_ Lsp_lsf	G729EV_G729_ Lsp_lsf.s
G729EV_G729_ Lsp_pre_select12	G729EV_G729_ Lsp_pre_select12.s
G729EV_ G729_ Pred_lt_3	G729EV_ G729_Pred_lt_3.s
G729EV_G729_ Residu	G729EV_G729_ Residu.s
G729EV_G729_ Residu2	G729EV_G729_ Residu2.s
G729EV_G729_ Syn_filt	G729EV_G729_ Syn_filt.s
G729EV_G729_ Syn_filt2	G729EV_G729_ Syn_filt2.s
G729EV_ CELP2S_d4i40_17_fast	G729EV_ CELP2S_d4i40_17_fast.s

우는 그대로 shl 함수로, 음의 값을 가지는 경우는 shl_LR로 해서 함수를 작성하였다. shr인 경우도 shr, shr_RL로 분리해서 작성하였다. 어셈블러로 작성된 함수 목록은 표 3과 같다.

G.729.1 고정 소수점 코드들의 기본 연산자 대부분은 오버플로우를 검사하여 오버플로우 플래그를 설정하도록 되어있어 추가적인 많은 계산량이 포함되어 있다. 그러나 실제 각 연산자들이 코드 내에서 실행될 때 오버플로우가 발생하지 않는 부분이 대부분이기 때문에 이들 부분에 대해서도 오버플로우를 검사하는 것은 속도를 느리게 하는 요소가 된다. 이러한 부분을 분리해 내기 위해서 먼저 각 연산자에 대해 오버플로우를 유발하는 지정문을 찾아 내어 오버플로우를 유발하는 연산과 그렇지 않은 연산에 대해 인라인 어셈블리 코드를 작성하여 추가적으로 성능을 향상시켰다. 결국 프로세서에서 제공하는 확장 명령어를 사용하여 인라인 어셈블러 형태로 프로그램을 변환하고, 성능을 저하시키는 오우버플로우 검사 기능을 분리하여 처리하므로서 전체적인 성능 향상을 이룰 수 있었다.

Ⅳ. 임베디드 시스템 시험

4.1 시뮬레이션 및 검증

작성된 프로그램을 ADS의 CodeWarrior에서 컴파일하여 생성된 인코더와 디코더 실행 파일을 가지고 각각의 모듈을 ITU-T에서 제공된 테스트 벡터를 이용하여 검증하였다. ITU-T에서 제공하는 테스트 벡터는 총 44개로서, 인코더용 14개와 디코더용 30개의 벡터가 있다. 구현된 G.729.1 프로그램의 시뮬레이션 실행 결과는 테스트 벡터와 비트 단위로 정확하게 일치 하였다. 표 4는 구현한 결과의 평균 복잡도 비교를 나타내고 있다. 본 논문에서는 인코더와 디코더가 각각 31.2, 22.8 MCPS의 계산량을 가지는데 최적화 이전에 비해 약 80% 정도 개선된 것을 알 수 있다.

표 4. G.729.1의 MCPS 계산

구분	최적화 전후	MCPS
Encoder @32k	Before Optimization	168.2
	After Optimization	31.2
Decoder @32k	Before Optimization	91.2
	After Optimization	22.8
Total	Before Optimization	259.4
	After Optimization	54.0

최적화 성능의 지표로 많이 사용되는 MCPS (Million Cycles Per Second)[15]의 측정방법은 다음과 같다.

(1) ADS에서 인코더, 디코더의 실행 사이클 수를 알아내기위해 ADS 수행 후 statistics 에서 total cycle 수를 파악한다.

(2) 다음 식을 사용하여 MCPS를 계산한다.

$$MCPS = \frac{N_{cycles} \times F_s}{N_{frames} \times M_{samples}} \bullet 10^{-6} \quad (1)$$

N_{cycles} : 실행된 사이클 수, F_s : 샘플링 주파수,

N_{frames} : 프레임 수, $M_{samples}$: 한 프레임 당 샘플 수

위의 수식을 적용한 MCPS 값은 다음과 같이 32kbps에서 인코더 31.2MCPS, 디코더 22.8MCPS로 계산되었다. Armulator에서 고려한 전제 사항은 캐쉬의 히트 율이 100% 이고, 메모리는 양포트 램을 사용한다고 가정한다. 따라서, 이 수치는 성능 추정을 위한 일종의 가이드 라인으로 볼 수 있으며 실제 캐쉬를 사용하면 이보다 더 시간이 걸릴 수 있다.

Encoder = {(Total Cycles) / (Total # of frames)}
* (1/frame time)
= {(392825196) / (629 frames)}
* (1/20ms) = 31.2 MCPS

Decoder = {(286993554) / (629 frames)}
* (1/20ms) = 22.8 MCPS

4.2 실행시간 측정

최종적으로 시뮬레이션상에서 구현된 프로그램은 실시간 동작을 위해 ARM926EJ-S 코어를 가지는 타겟 시스템에 포팅하여 시험 및 검증하는 작업이 필요하다. 타겟 시스템에 포팅하여 디버깅 및 검증을 편리하게 하기 위해서 먼저 타겟 시스템과 호스트 컴퓨터 간 이더넷을 통한 파일 공유를 통해 최적화된 코드를 실행한다. 타겟 시스템에서 리눅스 운영체제를 사용하므로, 리눅스용 호스트 컴퓨터와 타겟 시스템 간 다음 절차에 따라 파일을 공유하여 프로그램을 실행한다.

(1) 타겟 시스템과 호스트 컴퓨터를 이더넷에 연결한다.
(2) 타겟 시스템과 호스트 컴퓨터의 시리얼 포트를 서로 연결하고, 하이퍼 터미널 프로그램을 구동한다.
(3) 타겟 시스템이 연결되어 프롬트가 뜨면 루트로 로그인 하여 네트워크를 설정하고, 인코더와 디코더의 실행파일이 있는 호스트 컴퓨터의 디렉토리를 공유한다.
(4) 리눅스에서 제공하는 time 명령을 사용하여 각각의 실행시간을 측정한다. 실행 결과에서 user 란에 표시된 수치가 실행 프로그램을 수행한 시간이 된다.

표 5는 타겟 시스템에서 time 명령을 사용하여 32kbps에서의 인코더 및 디코더의 실행시간을 측정한 결과이다.

구현한 코덱을 탑재한 실제 인터넷 전화기 시험에서 호처리 프로그램과 연동하여 음성 신호를 마이크로 입력하고 상대방 전화기의 스피커로 출력하여 실시간 음성 통화를 확인하였다.

표 5. 실행 시간 측정

구분	최적화 전후	실행시간 [ms/frame]
Encoder @32k	Before Optimization	37.39
	After Optimization	6.75
Decoder @32k	Before Optimization	20.45
	After Optimization	4.76
Total	Before Optimization	57.84
	After Optimization	11.51

V. 결 론

본 논문에서는 단일 프로세서이면서 DSP 향상 확장 명령을 가지는 32 비트 고정 소수점 프로세서 코어인 ARM926EJ-S 코어를 이용하여 G.729.1 광대역 음성 코덱을 임베디드 시스템에서 실시간으로 동작하도록 구현하였다. 먼저 WMOPS 계산과 프로파일을 수행하여 코덱의 성능을 분석한 다음 이를 토대로 복잡도가 높은 블록을 위주로 어셈블러를 사용하여 최적화를 하였다. 구현한 G.729.1 코덱은 인코더와 디코더 부분이 각각 약 31.2 MCPS 및 22.8 MCPS의 복잡도를 나타내며, 실제 임베디드 시스템에서의 실행 시간은 각각 6.75ms와 4.76ms로 총 11.5ms가 걸렸다. 그리고 20ms 마다 한 프레임씩 음성 데이터를 입출력하는 G.729.1 코덱을 사용한 실제 인터넷 전화기에 적용하여 호처리 프로그램과 연동되어 실시간 음성통화를 확인하였다. 본 논문에서 구현된 G.729.1 광대역 음성 코덱은 저전력, 저가격, 고성능 기능의 ARM926EJ-S 코어를 가지는 프로세서에 적용하여 고품질의 인터넷 전화기, 대화형 원격 교육장치, 인터넷 음성 응용장치, 디지털 음성저장 장치 등 다양한 음성 압축/재생 응용장치에 적용할 수 있다.

참 고 문 헌

〔1〕 ITU-T Rec. G.729.1, “An 8-32kbit/s scalable wi-deband coder bitstream interoperable with G.729,” May 1995.
〔2〕 ITU-T Rec. G.729, “Coding of speech at 8 kb/s using conjugate-structure algebraic code-excited linear prediction (CS- ACELP),” June 1995.
〔3〕 ITU-T Recommendation G.729 Annex A, “Reduced complexity 8 kbit/s CS-ACELP speech

codec," Nov. 1996.
〔4〕 http://www.arm.com/products/CPUs/ARM926EJ-S.html
〔5〕 http://www.freescale.com/
〔6〕 Mithun Banerjee, G. Radhi Krishna, "Optimal Real Time DSP implementation of ITU G.729 Speech Codec," VTC 2004-Fall, Vol.6.
〔7〕 David H Crawford, Emmanuel Roy, "Techniques for Real-Time DSP implementation of Speech Coding Algorithms," Proc. DSP world, ICSPAT, pp.2-7, 1-4 November 1999.
〔8〕 Manish Arora, Suresh Babu P.V, Vinay M.K, "RISC PROCESSOR BASED SPEECH CODEC IMPLEMENTATION FOR EMERGING MOBILE MULTIMEDIA MESSAGING SOLUTIONS," DSP 2002, Vol.2, 1-3 July 2002, pp.831-834.
〔9〕 ITU-T Software Tool Library 2005 User's Manual, August 2005.
〔10〕 Ethan Bordeaux, "Solving AMR Speech Codec Porting Challenges," CommsDesign Technical Report, Aug. 2004.
〔11〕 GNU gprof, http://www.cs.utah.edu/dept/old/textinfo/as/gprof.html
〔12〕 Hedley Francis, "ARM DSP-Enhanced Extensions," ARM White Paper, May 2001.
〔13〕 ARM Ltd., ARM Developer Suite Version 1.2-CodeWarrior IDE Guide, March 2003.
〔14〕 Richard M. Stallman, GCC Developer Community, "Using the GNU Compiler Collection," Last updated 23 May 2004 for GCC 3.4.6.
〔15〕 http://www.dsprelated.com/groups/speechcoding/show/940.php

소 운 섭 (Woon Seob So) 정회원

1988년 2월 한밭대학교 전자공학과 학사
1994년 8월 충남대학교 전자공학과 석사
2006년 2월 충남대학교 정보통신공학과 박사 수료
1982년~현재 한국전자통신연구원 책임연구원
<관심분야> VoIP, 무선랜, 임베디드 시스템, 고속 멀티미디어

김 대 영 (Dae Young Kim) 정회원

1975년 2월 서울대학교 전자공학과 학사
1977년 2월 한국과학기술원 전기전자공학과 석사
1983년 2월 한국과학기술원 전기전자공학과 박사
1983년~현재 충남대학교 교수
<관심분야> Advanced Communication Protocol, Advanced Internet Protocol, 무선 인터넷