논문 2009-46SP-3-15

ARM Core®를 이용한 AMR-WB+ 오디오 부호화기의 실시간 구현

(Real-time Implementation of the AMR-WB+ Audio Coder using ARM Core®)

원 양 희*, 이 형 일**, 강 상 원***

(Yang-Hee Won, Hyung-il Lee, and Sang-Won Kang)

요 약

본 논문에서는 AMR Core®를 이용해서 AMR-WB+ 오디오 부호화기를 실시간 구현하였다. 구현 시 사용된 최적화 방법은 어셈블리어 단계에서 수행되었고, latency를 제거하고 32비트 레지스터를 사용하였다. 구현된 음성 부호화기는 평균 복잡도가 ARM9E 버전에서 인코더 160.76MHz, 디코더 33.05MHz으로 총 193.81MHz로 측정되었다. 사용된 ROM의 크기는 인코더 65.21Kbyte, 디코더 32.01Kbyte, 공통소스 279.81Kbyte이다. 구현된 AMR-WB+ 소스 코드는 3GPP에서 제공하는 테스트 벡터들을 CodeWarrior와 목표 PDA 상에서 모두 bit-exact하게 통과함을 보임으로써 검증되었다.

Abstract

In this paper, AMR-WB+ audio coder is implemented, in real-time, using Intel 400MHz Xscale PXA250 with 32bit RISC processor ARM9E-J®core. The assembly code for ARM9E-J®core is developed through the serial process of C code optimization, cross compile, assembly code manual optimization and adjusting the optimized code to Embedded Visual C++ platform. C code is trimmed on Visual C++ platform. Cross compile and assembly code manual optimization are performed on CodeWarrior with ARM compiler. Through these stages the code for both ARM EVM board and PDA is implemented. The average complexities of the code are 160.75MHz on encoder and 33.05MHz on decoder. In case of static link library(SLL), the required memories are 65.21Kbyte, 32.01Kbyte and 279.81Kbyte on encoder, decoder and common sources, respectively. The implemented coder is evaluated using 16 test vectors given by 3GPP to verify the bit-exactness of the coder.

Keywords : AMR-WB+, Speech codec, Real-time implementation

Ⅰ. 서 론

최근 32kbps 이하의 저속에서 음성 및 음악 신호를 효과적으로 처리할 수 있는 통합 코덱이 활발히 연구되어 왔다. 음성 또는 음악과 같은 단일 콘텐츠를 처리하는 코덱은 이동통신에서의 음성통신, MP3와 같은 음악 재생기에서는 각각 매우 효과적으로 사용되고 있으나, 방송과 통신의 융합으로 보다 다양한 콘텐츠의 전송이 요구되는 응용들에서는 비효율적이다. 아울러 공중 채널의 근본적인 용량 한계로 인하여 오디오 신호의 저속 전송이 불가피하게 되었다. 이에 따라 음성 및 음악 신호를 통합적이고 효율적으로 처리할 수 있는 저속 코딩 기술이 필요하게 되었다.

음성 신호를 위한 대표적인 표준코덱으로 3GPP의 워킹그룹 4에서 각각 2000년 및 2001년 표준화된 협대

* 정회원, 삼성전자 정보통신총괄
(SAMSUNG Electronics)
** 정회원, LG전자 CTO 사업부
(LG Electronics)
*** 평생회원, 한양대학교 전자컴퓨터공학부
(School of Elect. Eng. and Comp. Sci., Hanyang Univ.)
접수일자: 2008년8월12일, 수정완료일: 2009년4월15일

역 AMR[1]과 광대역 AMR-WB[2] 코덱을 들 수 있다. 이러한 코덱들은 통신망의 채널상태에 따라 소스코딩의 모드를 달리하는 적응모드 기능을 보유하여, 전송로상의 다양한 오류상황에서도 향상된 음질을 제공한다. 특히 AMR-WB은 6.6~23.85kbps 범위에서 동작하는 9개 모드가 정의되어 있으며, GSM full-rate에서 동작한다. 코딩방식은 대수 CELP방식[3]을 사용한다. AMR-WB은 3GPP에서 뿐만 아니라, 현재의 GSM시스템에도 응용가능하다.

음악 신호를 위한 대표적인 표준코덱으로 2005년 표준화된 오디오 부호화용 Enhanced AAC+방식[4]을 들 수 있다. Enhanced AAC+방식은 spectral band replication(SBR)기술과 binaural cue coding(BCC) 방식을 이용하여 최고 48kHz로 샘플링된 오디오 신호를 48 kbps 전송속도로 부호화가 가능하다.

이러한 기존의 코딩 기술은 음성 및 음악 신호에 대하여 각각 독립적으로 개발되어 왔으며, 음성 신호에 대하여 CELP(Code-Excited Linear Prediction) 구조라는 최적의 기술이 개발되었고, 음악 신호에 대해서는 AAC+ 기술[4]이 완성되었다. 그러나 각 코딩 기술은 서로 다른 영역의 특성을 가지는 입력에 대해서는 큰 성능 저하를 유발하는 문제점을 가지고 있다. 즉, CELP 코덱에 음악 신호가 입력되면 잘못된 모델링에 의하여 정확한 주파수 정보를 표현하지 못하여 매우 큰 성능 저하가 발생하고, 저속 AAC+ 코덱에 음성 신호가 입력되면 한정된 비트로 인하여 하모닉 구조를 정확하게 표현하지 못하여 성능이 저하된다.

최근에 이동통신 분야에서 다양한 특성의 신호를 처리하기 위하여 다중 구조를 가지는 AMR-WB+ 코덱[5]이 표준화 되었다. 이는 CELP 구조의 AMR-WB와 변환 구조의 TCX 모듈을 가지는 이중 구조로서, 입력 신호에 따라 CELP와 TCX 구조를 선택하여 사용한다. 결국, 음성은 주로 CELP로 처리하고 음악은 변환 코덱으로 처리하는 기본적인 구조를 제공하며, 하나의 코덱으로 32kbps 이하의 저속에서 음성과 음악 모두에 대하여 우수한 성능을 제공한다.

본 논문에서는 윈도우즈 CE를 운영 체제로 사용하고 ARM9Thumb 코어를 내장한 상용 PDA를 사용하여 AMR-WB+ 오디오 부호화기를 실시간 구현 및 검증하였다. 진행과정은 C 소스 최적화, 크로스 컴파일, 어셈블리 레벨에서의 최적화, 마지막으로 API 형태의 프로그램으로 목표 PDA 상에서 검증하는 일련의 과정으로 이루어졌다. 검증 과정에서 목표 PDA 및 ADS[6] 모두에서 C 소스와 bit-exact함을 확인하였다.

구현된 AMR-WB+ 오디오 부호화기의 목표 PDA로 Intel 400MHz Xscale PXA250 프로세서를 내장한 HP iPAQ 포켓 PC h3950 모델을 사용하였고, 어셈블리 디버깅 작업과 검증과정을 수행을 위해 ADS v1.2용 Metrowerks CodeWarrior를 사용하였다.

Ⅱ장에서 AMR-WB+의 기본 구조와 특성에 대하여 설명 하고, Ⅲ장에서는 실시간 구현 및 결과에 관하여 논하며, 마지막으로 Ⅳ장에서 결론을 맺도록 한다.

Ⅱ. AMR-WB+ 오디오 코덱

3세대 이동통신 시스템에서 오디오 미디어를 이용한 스트리밍, 메세징, 방송 서비스 등을 위해 높은 성능을 제공하는 낮은 전송률의 오디오 코딩 방법이 필요하게 되었다. 이를 위해 3GPP에서는 10kbps 미만에서부터 24kbps까지의 낮은 비트 전송률에서 우수한 성능을 제공하는 AMR-WB+ 코덱을 2005년에 표준화하였다.

2.1 AMR-WB+ 오디오 부호화기

AMR-WB+ 코덱은 하이브리드 ACELP/TCX 모델에 기반한다. 이는 입력되는 신호의 특성에 따라 LP-기반 코딩과 변환-기반 코딩 사이의 스위칭을 이용한다. 입력 신호는 16kHz에서 48kHz 범위의 샘플링 주파수를 갖는 모노 또는 스테레오 신호이다. 만일 스테레오 신호가 입력된다고 가정하면, 합 신호와 차이 신호 또는 부가 신호가 먼저 계산된다. 합 신호는 AMR-WB+ 코덱의 내부 주파수인 12.8kHz로 다운-샘플링 되는 저주파 신호 S_L과 6.4kHz 이상의 모든 주파수 성분들을 포

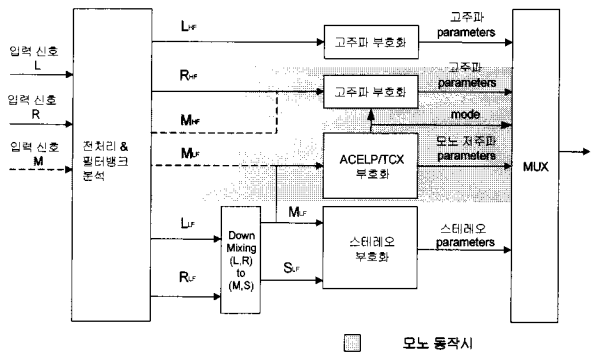


그림 1. AMR-WB+ 오디오 부호화기의 기본구조
Fig. 1. Basic structure of AMR-WB+ encoder.

함하는 고주파 신호 S_H의 두 개의 대역으로 나눠진다. S_L을 부호화하기 위해 하이브리드 ACELP/TCX부호화 모델이 사용되고, S_H를 부호화하기 위해 대역폭 확장(BWE) 방식이 사용된다. 부가 신호는 낮은 전송률의 준모수적 방법을 사용하여 부호화되고 이는 스테레오 형태를 가지며, 전체 부호화 과정의 기본 개념도는 그림 1과 같다.

저주파 모노 신호 S_L은 하이브리드 ACELP/TCX 모델을 사용하여 부호화된다. AMR-WB 코덱이 ACELP 모드에 이용되는 반면에, 대수 VQ가 사용되는 TCX는 변환 코딩 모드에 사용된다. 신호 S_L은 1024-샘플의 수퍼 프레임 단위로 처리되고, 이 수퍼 프레임은 256, 512, 1024-샘플 프레임들로 구성된다. 256-샘플 프레임은 ACELP나 TCX를 사용하여 부호화 될 수 있다. 하지만, 512-샘플 프레임과 1024-샘플 프레임은 TCX로만 부호화가 가능하다. 수퍼 프레임에는 이러한 모드들의 26가지 조합이 가능하다.

모드 선택 과정은 부호화기의 계산량을 제어할 수 있도록 하기 위해 폐루프 혹은 개루프에서 수행된다. 폐루프에서는 수퍼 프레임 내 26개 모드 조합에 의해 모드 선택 과정이 수행된다. 하지만, 개루프에서는 수퍼 프레임이 네 개의 256-샘플 프레임으로 나눠지고 각 프레임은 단지 ACELP나 256-샘플 TCX 모드만이 가능하다. 부호화 과정에서 계산량을 감소시키기 위해 모드 선택은 폐루프 검색 과정 대신 계산량이 상대적으로 적은 개루프 방법으로 결정할 수 있다. 결론적으로, 부호화기의 입력 신호가 분석되면 모드 선택은 분석된 오디오 신호 특성에 기반하여 수행된다.

TCX는 변환 기반 코딩 모드이기 때문에, 목표 신호 또는 가중된 신호에 적용되고 비-장방형 중첩 윈도우가 사용되어 코딩 이득을 향상시켜 준다. 이에 반해, ACELP모드는 목표 신호에 직접 적용되고 장방형 윈도우를 사용한다.

6.4kHz 이상의 주파수 성분을 갖는 고주파 신호 S_H는 BWE 방식을 사용하여 부호화된다. 이 방법은 스펙트럼 포락선과 이득 파라미터를 분석하고 양자화하여 복호화기에 전송한다. S_H의 다운-샘플링된 신호에 대해 계산된 스펙트럼 포락선은 8차 LP필터에 의해 모델링 된다. LP계수들은 프레임당 한 번씩 전송된다. 이 LP필터의 전송률은 수퍼 프레임 내에서의 모드 선택과 프레임 길이에 따라 갱신된다. 이득 수정은 각 서브프레임에 대해 계산되어 전송된다. 이는 저주파 대역과 고주파 대역 사이의 경계선(6.4kHz)에서의 연속성을 보장한다. BWE 방식은 단지 몇 개의 파라미터들만 전송하기 때문에 BWE 정보를 위해 사용되는 전체 비트 전송률은 0.8kbps 이하이다.

스테레오 부호화는 모노의 경우와 같은 대역 분할이 사용된다. 저주파 대역 스테레오 신호 코딩은 새로운 준모수적 기술을 이용한다. 두 채널은 AMR-WB+ 코어 코덱에 의해 부호화되는 모노 신호의 형태로 다운-믹스된다. 스테레오 이미지 정보는 저주파 대역을 두 개의 대역으로 분할하여 부호화된다. 매우 낮은 주파수 대역(0-1.0kHz)에 대해 스테레오 밸런스 요소가 모노 신호와 부가 신호 사이의 레벨 비를 나타내기 위해 유도된다. 낮은 대역 스테레오 이미지에 지각적으로 중요한 시간 해상도를 제공하기 위하여, 정규화된 부가 신호를 임계적으로 다운-샘플링한 후 파형 부호화한다.

부호화는 폐루프 가변 프레임 방식과 대수 VQ를 사용하여 주파수 도메인에서 수행된다. 프레임 길이의 후보는 하나의 수퍼 프레임의 전체 길이 또는 수퍼 프레임 전체 길이의 1/4, 1/2에 해당하는 길이 중에서 선택된다. 저주파 대역 신호의 고주파 부분(중간대역, 1.0-6.4kHz)은 채널 간 예측 기술과 유사한 형상-이득 제한 시간 영역 필터 방식에 따라 부호화된다. 이러한 새로운 방식은 안정적인 스테레오 이미지를 제공함으로써 채널 간 예측 방법의 문제점을 극복하고, 1.0-6.4kHz 대역에서 스테레오 정보의 효율적인 표현을 가능하게 한다.

2.2 AMR-WB+ 오디오 복호화기

복호화기의 기능은 전송된 파라미터들(LP파라미터,

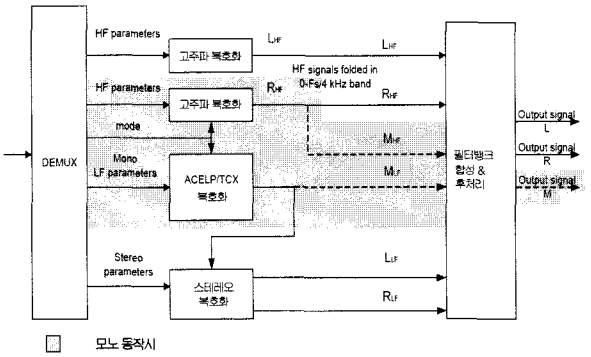


그림 2. AMR-WB+ 오디오 복호화기의 기본구조
Fig. 2. Basic structure of AMR-WB+ decoder.

적응 코드북 벡터, 적응 코드북 이득, 고정 코드북 벡터, 적응 코드북 이득, 고대역 이득)의 복호화와 재구성된 음성을 얻기 위한 합성 과정으로 구성되어있다. 그리고 재구성된 음성은 후 처리되고, 업샘플 된다. 고대역 신호는. 6-7kHz의 주파수 대역으로 생성된다. 복호화기에서의 신호 순서도가 그림 2에 도시되어 있다.

Ⅲ. 실시간 구현

본 논문에서는 ARM프로세서를 코어로 탑재한 Intel 400MHz Xscale PXA250 프로세서를 이용하여 ARM9E버전으로 구현된 소스의 실시간 실험 및 검증을 하였다.

실시간 구현은 개발 환경 구축, C 프로그램 최적화, 어셈블리 코딩 및 최적화, 그리고 ARM9E버전 스위칭 과정 및 bit-exactness 검증의 총 4단계의 과정으로 진행되었다. 그림 3은 이러한 개발 단계를 보였다.

첫 번째 단계인 C 소스 최적화 단계에서는 C소스[7]를 분석하여 불필요한 코드를 제거하고 크로스 컴파일에 적합하도록 C소스 코드를 변환하였다.

두 번째 단계에서는 변환된 C소스 코드를 크로스 컴파일한다. 이 과정을 거치면 어셈블리어로 컴파일된 코드가 생성되는데, 잉여 코드가 많지만 정확하기 때문에 간단한 모듈은 수작업 없이 그대로 이용할 수 있다. 크로스 컴파일의 다른 한 가지 이점은 복잡한 조건문 코딩에서 얻을 수 있는데. 조건문의 경우 연산량은 적지만 수작업으로 코딩할 때 디버깅 과정에 많은 시간을 필요로 한다. 크로스 컴파일을 이용함으로써 코딩 시간 및 디버깅 시간을 단축 하여 최종 구현 시간의 단축이 가능하다.

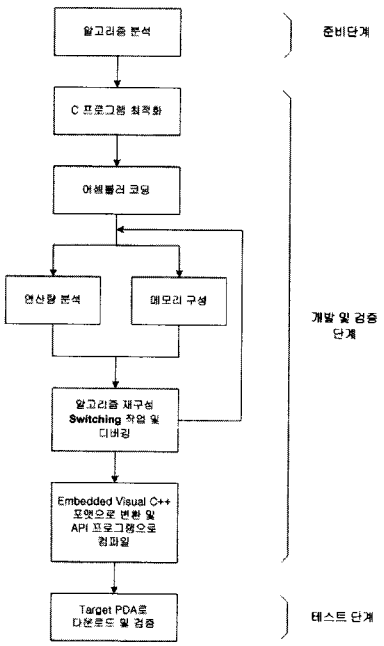


그림 3. DSP 프로그램 개발 절차
Fig. 3. Block diagram of the DSP programming.

세 번째 단계인 어셈블리 소스 최적화는 연산량이 큰 부분 특히 루프문, 베이직 연산, 어드레싱 방식 등을 중심으로 수작업으로 진행되었다. 루프문의 경우 사용 빈도가 높은 변수들은 레지스터에 할당하고 사용 빈도가 낮은 변수들은 스택 상의 메모리에 할당함으로써 데이터 이동을 최적화하였다. 또한 MAC과 같은 이중 명령을 적극 사용하여 연산량을 감소시켰다. 베이직 연산의 경우 호출함수의 상황에 맞도록 적용해서 잉여 코드를 제거하였다. 어드레싱 방식의 최적화는 규칙적으로 증가 혹은 감소하는 포인터 변수를 중심으로 이루어졌다. 크로스 컴파일과 어셈블리 소스 최적화는 ARM 컴파일러를 내장한 CodeWarrior상에서 이루어졌다. 그래서 위의 3가지 과정을 거쳐 최적화된 어셈블리 소스는 ARM EVM 보드 상의 실시간 구현에 바로 이용될 수 있었다.

네 번째 단계에서는 목표 PDA의 운영체제인 윈도우즈 CE용 API 프로그램으로 컴파일하기 위해 최적화된 어셈블리 소스를 임베디드 Visual C++에 호환되는 형태로 변형하였다. 이는 함수 호출 시 16비트 변수 전달 방식에 차이가 있기 때문이다.

3.1 최적화 기법

실시간 소스코드의 최적화를 위해 네 가지 기법이 사용되었다.

첫째로, Latency 문제를 해결하기 위해 파이프라인 적용을 전제로 코딩하였다. 예를 들어 결과값을 저장한 레지스터가 다음 라인에서 입력값으로 사용된다면 연산량이 증가하므로 이러한 경우를 피함으로서 latency를 줄일 수 있다. 아래 예제는 로드 명령어 때문에 한 사이클의 인터로크가 발생한 경우를 나타낸다.

```
Ex)  LDR   r1, [r0, #+0]
     ADD   r3, r3, r1
```

Latency를 제거하기 위해 다음과 같이 로드 명령어 바로 다음에 이를 입력값으로 쓰는 명령어를 사용하지 않는다.

```
Ex)  LDR   r1, [r0, #+0]
```

```
MOV   r2, #+0
ADD   r3, r3, r1
```

둘째로, 최적의 루프문 구현을 위하여, 루프 안에 사용되는 레지스터 값을 최소한의 레지스터로 루프 밖에서 구현하였다. 예로서 아래와 같은 구문이 있다고 가정 시

```
Ex)
  for (i = 0; i < lg; i++)
  {
      L_tmp = L_mult(x[i], a0);
      for (j = 1; j <= 4; j++)
          L_tmp = L_msu(L_tmp, a[j], yy[i - j]);
      L_tmp = L_shl(L_tmp, 3);
      y[i] = yy[i] = round(L_tmp);
  }
```

크로스 컴파일을 하게 되면, a[j]값을 사용하기 위해 로드 명령어를 ix4만큼 실행하지만, a[j]값은 루프문 안에서 변하지 않으므로 명령어 수를 최소화하기 위해 a[j]값을 루프 밖에서 다음과 같이 구현할 수 있다.

```
Ex)  LDRH    r5, [r0, #+2*1]
     LDRH    r6, [r0, #+2*3]
     LDRH    r7, [r0, #+2*2]
     LDRH    fp, [r0, #+2*4]
     ORR     r5, r5, r7, LSL #+16
     ORR     r6, r6, fp, LSL #+16
```

이와 같이 한 개의 레지스터에 두 개의 값을 대입하여, 루프문 안에서 로드 명령어 없이 레지스터를 바로 사용함으로써 루프문 안을 최적화하였다.

셋째로, 16bits 곱셈기능에서는 32bits 레지스터에 T(Top 16bits) & B(Bottom 16bits)를 사용하여 16bits 곱셈기능을 가능하게 하였다. 예로써 SMULBT, SMULBB, SMLATT, UMULL 등의 명령어를 들 수 있다. 예로서 아래에서 보여진 구문과 같이 로드 명령어를 두 번 사용할 것을 한번으로 줄일 수 있다.

```
Ex)  LDR       r5, [r0, #+0]
     SMULBB    r1, r5, r5
     SMULTT    r2, r5, r5
```

넷째로. QADD, QSUB, QDADD 등의 명령어를 사용하여 saturation을 쉽게 해결하였다.

표 1. 구현된 AMR-WB+ 실시간 소스코드의 테스트 벡터 별 평균 복잡도(MCPS-Million Clocks Per Second)

Table 1. Average complexity of the AMR-WB+ real-time source code (MCPS).

테스트 벡터	부호화기	복호화기	전체
concatener_green_test_amr.wav	164.8	36.63	201.4
m_po_x_1_org8km.wav	136.79	19.31	156.10
m_po_x_1_org8ks.wav	172.63	36.75	209.38
m_po_x_1_org11k.wav	171.55	36.76	208.31
m_po_x_1_org16k.wav	173.55	36.78	210.33
m_po_x_1_org22k.wav	172.24	36.78	209.02
m_po_x_1_org24k.wav	173.86	36.78	210.64
m_po_x_1_org32k.wav	174.10	36.78	210.88
m_po_x_1_org44k.wav	164.24	36.78	201.02
m_po_x_1_org48k.wav	165.16	36.78	201.94
m_po_x_1_org48km.wav	132.66	19.33	151.99
sw_48.wav	163.41	36.33	199.74
sw1_48.wav	162.37	35.14	197.51
TH_16m.wav	136.05	18.85	154.90
TH_16s.wav	172.73	36.50	209.23
T_mode.wav	136.13	32.65	168.78

표 2. 구현된 AMR-WB+ 실시간 소스코드의 메모리 사용량 (Kbytes)

Table 2. Memory requirements of the AMR-WB+ real-time source code (Kbyte).

모 듈	Kbytes
부호화기	65.21
복호화기	32.01
Common	75.58
Lib_amr	196.13
스테레오	8.10
합 계	377.03

구현된 AMR-WB+ 실시간 소스코드의 테스트 벡터 별 평균 복잡도를 표 1에 도시하였다.

구현된 AMR-WB+ 실시간 소스코드의 메모리 사용량을 표 2에 나타냈다.

VI. 결 론

본 논문에서는 음성/오디오 통합코덱으로 3GPP에서 표준화된 AMR-WB+ 오디오 부호화기를 32비트 RISC 프로세서인 Intel 400MHz Xscale PXA250을 사용하여 ARM9E버전으로 실시간 구현하였다. 구현된 소스의 검증을 위해 3GPP에서 제공된 16개 테스트 벡터들을 사

용해서 C 시뮬레이션의 결과와 bit-exact함을 확인하였다. ARM9E의 구조적인 장점과 명령어들의 효율적인 사용을 통하여 최소의 복잡도를 가지도록 ARM9E 어셈블리어 소스 코드를 구현하였다.

구현된 AMR-WB+ 실시간 소스코드는 3GPP에서 제공된 테스트 벡터들에 대해 인코더와 디코더 모듈이 각각 160.76MHz 및 33.05MHz의 평균 복잡도를 나타내었다. 사용된 ROM의 크기는 인코더 65.21 Kbytes, 디코더는 32.01 Kbytes, 공통소스는 279.81 Kbytes이다.

DSP 확장 버전인 ARM9E 버전을 사용함으로써 연산량 측면에서 보다 나은 성능 향상을 보임은 물론, ARM계열 코어의 특징인 저전력 소비 및 ASIC 구현의 용이함 등의 장점으로 인해 음성 압축/재생 핵심모듈의 저가 구현이 가능하다.

참 고 문 헌

[1] GSM 06.90 "Digital cellular telecommunication system; Adaptive Multi-Rate speech Transcoding," 2000.

[2] 3GPP TS 26.190, "AMR Wideband Speech codec Transcoding Functions (Release 5)," 2001.

[3] B.S. Atal and M.R. Schroeder, "Stochastic coding of speech at very low bit rate," *Proc. Int. Conf. Comm.*, Amsterdam, pp. 1610-1613, 1984.

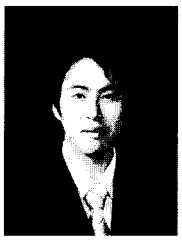
[4] 3GPP TS 26.401, "Enhanced aacPlus general audio codec General description," 2006.

[5] 3GPP TS 26.290, "Extended Adaptive Multi-Rate Wideband (AMR-WB+) codec; Transcoding functions," 2006.

[6] Andrew N. Sloss, Dominic Symes and Chris Wright, "ARM System Developer's Guide," ELSEVIER Inc., pp.207-256, 2004.

[7] 3GPP TS 26.273, "ANSI-C code for the fixed-point Extended Adaptive Multi-Rate - Wideband (AMR-WB+) speech codec," 2006.

저 자 소 개



원 양 희(정회원)
1999년～2006년 한양대학교 전자컴퓨터공학부 (공학사)
2006년～2008년 한양대학교 전자전기제어계측공학과 (공학석사)
2008년～현재 삼성전자
<주관심분야 : 음성/오디오 부호화, 신호처리, 이동통신>

이 형 일(정회원)
1999년～2006년 한양대학교 전자컴퓨터공학부 (공학사)
2006년～2008년 한양대학교 전자전기제어계측공학과 (공학석사)
2008년～현재 LG전자
<주관심분야 : 음성/오디오 부호화, 신호처리, 이동통신>

강 상 원(평생회원)
1976년～1980년 한양대학교 전자공학과 (공학사)
1980년～1982년 서울대학교 전자공학과 (공학석사)
1985년～1990년 Texas A&M 대학교 전기공학과 (공학박사)
1982년～1994년 2월 한국전자통신연구원 신호처리 연구실
1994년 3월～현재 한양대학교 전자컴퓨터공학부 교수
<주관심분야 : 음성/오디오 부호화, 신호처리, 이동통신>