TMS320C6201을 이용한 적응 다중 전송율을 갖는 광대역 음성부호화기의 실시간 구현

정회원 이 승 원*, 배 건 성**

Real-Time Implementation of Wideband Adaptive Multi Rate (AMR-WB) Speech Codec Using TMS320C6201

Seung-won Lee*, Keun-sung Bae** *Regular Members*

요 약

본 논문에서는 적응 다중 전송율을 갖는 광대역 음성부호화기인 AMR-WB의 알고리즘을 분석하고, TI사의 고정소수점 DSP인 TMS320C6201를 이용한 실시간 구현 결과를 제시한다. AMR-WB 음성부호화기는 두 가지 대역으로 분리된 신호가 독립적으로 부호화되며, 저대역 신호는 ACELP 방식으로, 고대역 신호는 잡음 여기신호와 선형예측 합성필터를 사용하는 방식으로 각각 합성된다. 구현된 AMR-WB 음성부호화기는 프로그램 메모리와 데이터 메모리가 각각 218 kbytes, 92 kbytes의 크기를 가지며, 한 프레임인 20 ms를 처리하는데 평균 920,267 정도의 클럭 수가 사용되어 약 5.75 ms의 시간이 소요되었다. 또한, DSP로 구현한 AMR-WB 음성부호화기의 결과와 PC에서 시뮬레이션 한 결과가 서로 일치함을 확인하였다.

Key Words : AMR-WB, speech codec, DSP, real-time implementation, TMS320C6201

ABSTRACT

This paper deals with analysis and real-time implementation of a wideband adaptive multirate speech codec (AMR-WB) using a fixed-point DSP of TI's TMS320C6201. In the AMR-WB codec, input speech is divided into two frequency bands, lower and upper bands, and processed independently. The lower band signal is encoded based on the ACELP algorithm and the upper band signal is processed using the random excitation with a linear prediction synthesis filter. The implemented AMR-WB system used 218 kbytes of program memory and 92 kbytes of data memory. And its proper operation was confirmed by comparing a decoded speech signal sample-by-sample with that of PC-based simulation. Maximum required time of 5.75 ms for processing a frame of 20 ms of speech validates real-time operation of the implemented system.

Ⅰ. 서 론

오늘날 사용되는 대부분의 음성부호화 시스템에서는 일반적으로 300~3400 Hz의 대역폭을 갖는 협대역 음성신호를 고려한다. 이는 전통적인 PSTN(Public Switched Telephone Network) 망에서의 대역폭 제한에 기인한 것으로 음성의 명료도 및 자연성 측면에서 통화품질에 제한을 주게 된다. 그러나 제 2세대 무선 이동통신 시스템과 ISDN(Integrated Services Digital Network), 패킷 기반의 디지털망의 등장으로 보다 넓은 대역폭을 사용할 수 있게 되었고, 그 결과로 기존의 PSTN

* LG전자 DND 영상제품 연구소, ** 경북대학교 전자공학과 신호처리연구실
논문번호 : 040085-0219, 접수일자 : 2004년 2월 24일
※이 논문은 2002년도 경북대학교 특성화사업팀(KNURT) 연구비에 의하여 연구되었음.

망에서의 통화품질을 월등 능가하는 통신 품질을 제공할 수 있게 되었다. 대역폭 제한이 줄어들면서 음성신호에 대한 샘플링 주파수를 16 kHz로 하고 50~7000 Hz의 대역폭을 고려하는 광대역 음성부호화 방식이 등장하게 되었다. 광대역 음성부호화 방식에서는 증가한 저주파 성분이 음질의 자연스러움을 개선시키고, 고주파 성분이 마찰음의 구분을 명확 하여 전체적으로 음성의 자연성 및 명료도가 향상된다[1].

16 kHz로 샘플링된 광대역 음성신호에서 각각의 샘플값을 16 bits의 해상도로 표현할 경우, 압축되지 않는 신호의 비트율은 256 kbit/s가 되어 음성부호화 및 음성 압축 방식이 중요한 문제가 된다. 광대역 음성부호화 방식에 대한 연구로, CCITT(Consultative Committee on International Telephone and Telegraphy)가 1988년에 처음으로 광대역 음성부호화 표준안으로 48, 56, 64 kbit/s의 비트율을 갖는 G.722 방식을 선정하였고, 1999년에는 ITU-T에서 24, 32 kbit/s의 비트율을 갖는 G.722.1 방식을 정하였다[2]. 최근에는 ETSI(European Telecommunications Standards Institute) 및 3GPP(Third Generation Partnership Project)에 의해 새로운 방식의 광대역 음성부호화 표준안을 위한 활동이 진행되었으며, 그 결과 2000년에 AMR-WB(Adaptive Multi Rate-Wideband) 방식이 GSM 망 및 3세대 무선 이동통신 시스템에서의 광대역 부호화 방식의 표준안으로 채택되었다. AMR-WB 방식은, 또한, ITU-T (International Telecommunications Union-Telecommunications standard sector)의 표준안 선정 과정에 참여하여 2001년에 G.722.2 방식으로 채택되었는데, 이는 동일한 부호화 방식이 처음으로 유선통신 및 무선통신망에서 동시에 채택되어 불필요한 중복부호화의 생략을 통해 광대역 음성통신 응용 및 서비스 구현에 편의를 제공한다는 점에서 중요성을 가진다[2].

본 연구에서는 광대역 음성부호화 분야의 최신 기술인 AMR-WB 음성부호화 알고리즘을 분석하고 Texas Instruments(TI)사의 TMS320C 6201 EVM 보드상에서 실시간으로 구현하였다. 실시간 동작을 위한 소스코드 분석 및 최적화 과정을 통하여 20 ms 길이의 음성신호 한 프레임을 인코딩 및 디코딩하는데 약 5.75 ms가 소요되었다. 본 논문의 구성은 다음과 같다. 2장에서는 AMR-WB 음성부호화 시스템 구성과 알고리즘에 대해 설명한다. 3장에서는 TMS320C6201 DSP를 이용한 실시간 구현 과정을 설명하고 구현결과를 4장에서 제시한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. AMR-WB 음성부호화 시스템

1. AMR-WB 시스템 구성

AMR-WB 시스템은 9가지 비트율을 갖는 음성부호화기, SCR(Source Controlled Rate)를 위한 VAD(Voice Activity Detector), 묵음구간에서의 배경잡음 생성을 위한 CNG(Comfort Noise Generation), 손실된 프레임의 보상을 위한 ECM(Error Concealment Mechanism) 등으로 구성된다[3]. 음성부호화기는 ACELP (Algebraic Code Excited Linear Prediction) 방식에 기반하고 있으며, 6.6 kbit/s 부터 23.85 kbit/s 사이의 9가지 전송모드를 갖는다[4]. 또한 DTX(Discontinuous Transmission)[5]에 사용되는 1.75 kbit/s의 배경잡음 부호화 모드를 갖는다. AMR-WB 음성부호화기는 20 msec 길이의 프레임마다 비트율을 선택할 수 있는데, 9가지 전송모드의 비트율은 표 1과 같다.

표 1. AMR-WB 음성부호화기의 비트율

Codec mode	Source codec bit-rate
AMR-WB_23.85	23.85 kbit/s
AMR-WB_23.05	23.05 kbit/s
AMR-WB_19.85	19.85 kbit/s
AMR-WB_18.25	18.25 kbit/s
AMR-WB_15.85	15.85 kbit/s
AMR-WB_14.25	14.25 kbit/s
AMR-WB_12.65	12.65 kbit/s
AMR-WB_8.85	8.85 kbit/s
AMR-WB_6.60	6.60 kbit/s
AMR-WB_SID	1.75 kbit/s(*)

(*) SID(Silence Descriptor) 프레임이 연속적으로 전송된다고 가정

아이

SCR 동작은 음성프레임이 배경 잡음만을 포함하고 있을 경우, 음성부호화기가 아주 낮은 비트율로 프레임을 부호화하는 것을 말한다. 이를 통해 전력소모를 줄여줌으로써 단말기의 배터리 수명을 연장할 수 있고, 부호화에 요구되는 평균 비트율을 감소시켜줌으로써 채널을 효율적으로 활용할 수 있게 된다. SCR 동작을 위해서는 송신단에서 VAD를 통한

음성구간의 검출과 배경잡음 추정과정이 필요하다. 이때, 배경잡음에 대한 정보의 전송은 SID(Silence Descriptor)[5] 프레임의 전송을 통해 이루어지며, 수신단에서 CNG에 이용된다. VAD는 20 ms 길이의 음성프레임마다 음성신호를 포함하는가의 여부를 플래그(VAD_FLAG)로 표현한다. 최종 VAD_FLAG는 음성신호를 필터뱅크를 통과시켜서 구한 대역별 에너지 정보와, 음성부호화기에서 구한 개루프 피치 이득 정보를 이용하는 톤 성분 검출결과를 모두 고려하여 구해진다. CNG는 수신단에서 인위적인 잡음성분을 발생시켜 비음성 프레임구간에 대해 듣기에 거북하지 않은 소리를 발생시키는 과정이다. 이를 위해 송신단에서는 배경 잡음을 추정하여 낮은 비트율의 SID 프레임 형태로 정보를 전송한다. ECM은 송신단에서 전송한 정보에 오류가 생기거나 손실될 경우 음성신호와 상관성이 없는 단순한 묵음을 재생하지 않고, 이전의 성공적으로 수신된 프레임 파라미터를 바탕으로 합성음을 복원하여, 프레임 오류로 인한 합성음의 음질 저하 방지를 목적으로 수행된다.

2. AMR-WB 음성부호화 알고리즘[4]

AMR-WB 음성부호화기는 16 kHz의 샘플율에서 20 ms 길이의 프레임단위로 동작하는데, 각 프레임은 5 ms 길이를 갖는 4개의 서브프레임으로 나누어진다. 이때, 계산량의 감소와 중요한 정보를 가진 대역폭에서의 비트 할당에 초점을 맞추기 위해 입력신호를 50~6400 Hz의 저대역 신호와 6400~7000 Hz의 고대역 신호로 나누어 각각의 대역을 독립적으로 부호화 한다. AMR-WB 인코더의 구성은 그림 1과 같다. 저대역 신호의 부호화를 위해 입

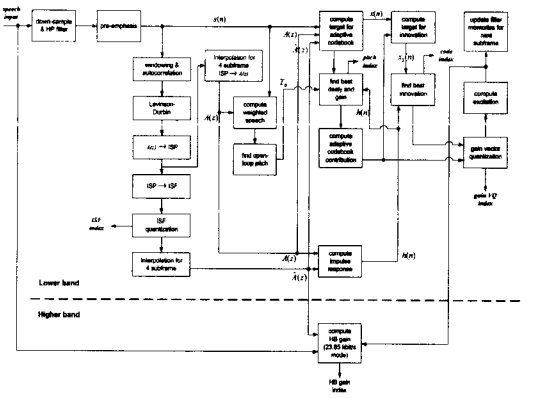


그림 1. AMR-WB 인코더의 구조

력신호는 12.8 kHz의 샘플율로 다운샘플링 되고 pre-emphasis 과정이 적용된 후, 협대역 음성부호화기의 표준안에 널리 사용되는 ACELP 방식으로 부호화 된다. 선형예측분석은 20 ms의 프레임마다 한 번씩 수행되고, 선형예측계수는 양자화 및 전송특성을 고려하여 ISP(Immittance Spectral Pairs)[6]로 변환된다. 또한, 고정 여기신호 및 적응 여기신호를 위한 코드북의 탐색은 5 ms 길이의 서브프레임 단위로 수행된다. 광대역 음성신호의 스펙트럼은 협대역 신호에 비해 보다 넓은 동적범위를 갖기 때문에 AMR-WB에서는 스펙트럼의 포만트 특성뿐만 아니라 포락선의 경사도를 고려하는 지각가중 필터가 적용되며, 유성음 구간에서의 피치 특성을 보다 잘 표현하기 위해 다양한 특성 필터를 피치 분석에 적용하고 있다[7].

디코더에서는 그림 2와 같이 수신된 정보 비트열로부터 ISP 벡터, 4개의 분수 피치 값, LTP(Long

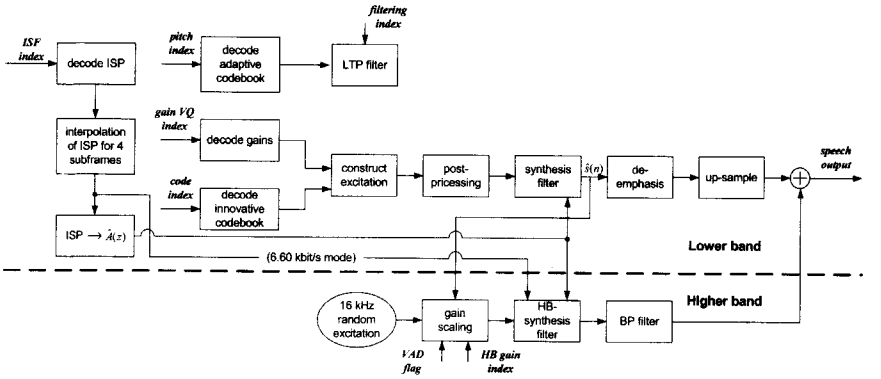


그림 2. AMR-WB 디코더 구조

Term Prediction) 필터 파라미터, 여기신호 벡터, 그리고 벡터 양자화된 적응 코드북과 고정 코드북의 이득을 추출한다. 이들 파라미터를 이용하여 ACELP 합성 모델을 이용해서 서브프레임 단위로 여기신호를 구성하고 선형예측 합성필터를 통과시켜 12.8 kHz 샘플율을 갖는 저대역 음성신호를 복원한다. 또한 복원된 음성신호는 인코더와 같은 값을 사용하여 de-emphasis를 적용하고, 마지막으로 16 kHz 샘플율의 신호로 업샘플링된다. 고대역 신호는 저대역 신호에서 분석한 선형예측 필터특성을 이용하여 재현한 파라미터와 랜덤 여기신호를 이용해서 합성하게 된다. 고대역 신호의 이득은 23.85 kbit/s 모드에서는 인코더에서 전송된 값이 사용되고, 나머지 모드에서는 유성음화정보(voicing information)를 이용하여 조정된다. 합성된 고대역 음성신호는 저대역 음성신호와 더해져서 최종 합성음을 구성하게 된다.

Ⅲ. TMS320C6201 EVM을 이용한 실시간 구현

1. TMS320C6201 DSP 및 개발환경

TMS320C6201 DSP는 TI사의 'C62xx 계열의 고정소수점 DSP 중 하나로서, CPU는 200 MHz의 clock rate에서 최대 성능 1600 MIPS의 성능을 발휘할 수 있다. 본 연구에서는 TMS320C6201 EVM(Evaluation Module) 보드와 code generation tool을 하나의 통합 소프트웨어로 제공하는 TI사의 Code Composer Studio(CCS)[8]를 개발환경으로 이용하였다. 그림 3에 TMS320C6201 EVM 보드 구조의 블록도를 나타내었다. EVM 보드에 장착된 메모리는 내부 메모리와 외부 메모리로 구성되며 내부 메모리는 2k x 256 bits 프로그램 메모리와 64 kbytes의 데이터 메모리로 나누어진다. 그리고 외부 메모리는 256 kbytes의 SBSRAM 및 2×4 Mbytes의 SDRAM으로 구성된다. PCI 버스를 이용해서 PC와의 인터페이스를 제공하며, 주변장치로는 신호의 입출력을 위한 오디오 코덱 및 기타 직렬장치 등과의 직접적인 통신을 위한 2개의 MCBSP(Multi Channel Buffered Serial Port), 2개의 32 bits 타이머, PLL 클럭 발생기 등이 있다. 본 연구에서 사용한 소프트웨어 툴은 2.10 version의 CCS이며, CCS에 내장되어 사용된 C/C++ Compiler Shell, COFF Assembler, Assembly

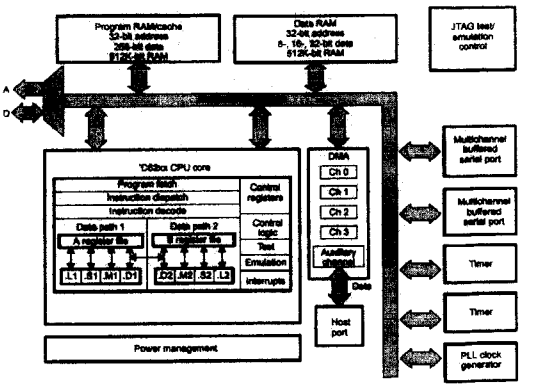


그림 3. TMS320C6201 EVM 보드 구조의 블록도

Optimizer, COFF Linker 및 Standalone Simulator는 모두 version 4.20이다.

AMR-WB 음성부호화기의 실시간 구현은 ETSI 및 3GPP에서 제공되는 ANSI-C 코드(version 5.0.0)[9]를 기반으로 하였으며, 소스 코드에는 9가지 비트율의 음성부호화 알고리즘 뿐만 아니라 VAD, CNG, SCR, ECM을 모두 포함하고 있다. AMR-WB ANSI-C 코드는 58개의 소스코드 파일(*.c), 19개의 헤더 파일(*.h) 그리고 ROM 데이터를 정의하고 있는 9개의 데이터 파일(*.tab)들로 구성되어 있다. 우선, 소스코드의 최적화 작업에 앞서 이들 파일들의 분석을 통해 음성부호화 알고리즘과 직접적인 관련이 없는 [count.c] 파일 등과 printf(), fwrite() 등의 함수의 사용을 모두 제외시켰다. [count.c] 파일은 알고리즘의 복잡도 등을 계산하기 위해 Init_WMOPS_counter(), WMOPS_output() 등과 같은 함수들을 정의하고 있는데, 음성부호화 알고리즘과는 직접적인 연관이 없다.

2. 실시간 동작을 위한 최적화 과정

AMR-WB 음성부호화기의 실시간 동작을 위해 다음과 같은 최적화 작업을 수행하였다[10].

1) [basicop2.c]에 정의되어 있는 ETSI의 기본 산술연산 함수들을 intrinsic으로 대체하였다.
2) [oper_32b.c]에 정의되어 있는 double precision 연산 함수들을 inline subroutine으로 대체하였다.
3) 고정된 값을 갖는 ROM 데이터를 정의한 배열을 const형으로 재선언 하여 메모리의 .const 영역에 위치시켰다.
4) 함수내부의 지역 변수의 선언을 short형에서

int형으로 변환하였다. 이때, 변수 값의 saturation 가능성이 없는 loop trip counter 변수를 우선적으로 선별하여 적용하였다.

5) Loop trip counter가 작을 경우, loop 내부의 연산을 직접 반복 사용하여 for 문의 사용빈도를 줄였다.
6) 내부메모리의 접근 속도가 외부메모리의 접근 속도보다 빠른 장점을 이용하기 위해 내부메모리의 사용을 최대화하였다. 이때, 함수의 사용빈도가 많은 함수들을 우선적으로 내부메모리에 위치시켰다.
7) #pragma directive를 사용하여 compiler에게 알고리즘 분석을 통해 얻은 특정 함수나 소스 코드의 일정 부분을 어떻게 취급할 것인가에 대한 정보를 주었다. #pragma MUST_ITERATE( ); 및 #pragma PROB_ITERATE( ); 등과 같은 선언을 통해 반복 횟수가 고정되어 있지 않는 loop의 trip count에 대한 정보를 compiler에게 제공하였다.
8) _nassert( ) intrinsic을 사용하여 optimizer에게 최적화에 관한 정보를 제공하였다.

최적화 과정에 사용된 주요 내용을 요약하면 다음과 같다.

(1) Intrinsic 연산자의 사용

C6000 compiler는 intrinsic 연산자를 인식할 수 있다. Intrinsic 연산자는 C 문법으로 표현할 수 없거나, 표현하기 어려운 연산과정을 간단하게 사용할 수 있도록 assembly instruction에 mapping 시켜 구현되어 있는데, 일반 함수처럼 호출하여 사용할 수 있다. 즉, C 소스 코드에서 선언한 변수를 일반 함수에서의 사용과 같이 intrinsic의 인자로 사용할 수 있다. Intrinsic 연산자의 사용은 아래 예에서와 같이 다음의 일반 함수의 사용과 같이 생각할 수 있다. 이때, _sadd intrinsic은 assembly instruction의 SADD와 동일한 연산을 수행한다.

```
int x1, x2, y;
y = _sadd(x1, x2);
```

(2) Inline 키워드의 설정

함수의 원형을 선언할 때, inline 키워드를 설정하면 그 함수가 호출되어 사용될 때, 일반적인 함수의 호출 과정을 따르는 것이 아니라 inline으로 확장되어 사용된다. 즉, inline 키워드와 함께 선언된 함수를 호출하여 사용할 경우, 함수의 호출은 함수 내부의 코드로 대체된다. 따라서 본 연구에서는 double-precision 연산을 위한 함수들을 정의한 [oper_32b.c] 내부의 모든 함수들에 inline 키워드를 설정하여 아래의 예와 같이 사용하였다.

```
void L_Extract (Word32 L_32, Word16 *hi,
Word16 *lo)
{
        *hi = extract_h (L_32);
        *lo = extract_l (L_msu (L_shr (L_32,
              1), *hi, 16384));
        return;
}

static inline void L_Extract(int L_32, short *hi,
short *lo)
{
        *hi = extract_h(L_32);
        *lo = extract_l(L_msu(L_shr(L_32,
              1),*hi,16384));
        return;
}
```

(3) ROM 데이터의 const형 선언

음성부호화 과정에서 함수에 의해 값이 변하지 않고 항상 일정한 값을 유지하고 있는 변수를 const형으로 선언하고, 이를 통해 고정된 테이블 ROM 데이터를 메모리의 constants 영역(.const)에 위치시켰다. 이는 compiler가 DP addressing (.bss)을 사용하는 대신 absolute addressing을 사용하게 하고, 멀티 채널의 부호화기를 구현할 경우에 모든 ROM 데이터를 global하게 사용할 수 있다는 장점이 생긴다. 또한 .bss 영역을 다른 변수들[연산과정에서 값이 변하는 변수]을 위해 할당할 수 있다는 장점이 있다. 실제 선형예측계수의 분석 과정에서 사용되는 hamming window의 값을 저장하고 있는 변수를 const형으로 선언한 예는 아래와 같다.

```
#define L_WINDOW 384
static Word16 window[L_WINDOW] = {}

#define L_WINDOW 384
static const Word16 window[L_WINDOW] = {}
```

(4) #pragma directive의 사용(loop trip counter 정보관련)

*#pragma directive*는 *compiler*가 특정 함수나 코드의 일부 영역을 어떻게 처리해야 하는가에 대한 정보를 제공하게 된다. C6000 C compiler는 18가지의 pragma를 지원하게 되는데, 이들 중 loop문의 trip counter에 대한 정보를 제공할 수 있는 PROB_ITERATE 또는 MUST_ ITERATE를 반복 횟수가 일정하지 않은 loop문에 선언하였다. 또한 이와 유사한 역할을 하는 _nassert intrinsic 또한 적용하였다. 실제 개루프 피치 검색에 사용되는 함수인 Pitch_med_ol함수를 동일한 build option을 적용하고, directive 및_nassert intrinsic을 적용하기 전후의 실행 사이클 수를 측정한 결과, 이 방법을 통한 최적화 작업이 유효함을 확인하였다. Compiler가 생성한 assembly 파일을 비교한 결과는 표 2와 같다.

표 2. Pragma directive 및 _nassert 적용 전/후의 결과 비교

	Pragma directive 및 _nassert	
	적용 전	적용 후
Local Frame Size	36 bytes	32 bytes
실행 사이클 수	126	110

(5) 지역변수의 형변환 과정

short[Word16] 형으로 선언된 지역변수 중 loop counter와 같은 saturation 가능성이 없는 변수들을 int[Word32] 형으로 재선언하였다. 이는 사용되지 않는 bit 영역의 masking 작업을 감소시키기 위한 것이며, 아래의 예와 같이 적용하였다.(이때, L은 subframe의 길이를 나타내는 64의 값으로 고정된 값이다.)

```
Word16    n;
for (n = 0; n < L; n++)
{
}
Word32    n;
for (n = 0; n < L; n++)
{
}
```

(6) Loop 문의 전개

Loop문의 사용에서 trip counter의 크기가 작을 경우 loop 문 내부의 코드를 전개하여, loop문의 실행시 실행되는 branch instruction을 불필요하게 하였다. 이러한 과정은 변수의 초기화 과정 및 다중 loop 구조에서의 작은 trip counter를 갖는 loop문 등에 적용하였다.

Ⅳ. 실시간 구현 및 결과

실시간으로 구현한 AMR-WB 음성부호화 시스템의 성능을 평가하기 위해 프로그램의 실행속도와 메모리 사용량을 측정하였다. 프로그램의 실행 속도는 clock() 함수를 Standalone Simulator[load6x]와 같이 사용하여 cycle 수를 측정하였으며, 이때, 음성신호의 입/출력과 시스템 초기화 및 종료 과정에 소요되는 cycle 수는 순수한 음성부호화 과정과는 무관하다고 간주하고 생략하였다. 프로그램 및 데이터 메모리의 측정은 COFF link시 생성한 memory map 파일을 참조하여 측정하였다. 실험에 사용한 음성데이터는 남성화자가 발성한 2초 길이의 문장이며, 그림 4에 주어진 것과 같이 대부분의 구간에서 음성이 존재하는 문장이다.

구현한 AMR-WB 음성부호화기의 전체 시스템에서 지원되는 DTX(Discontinuous Transmission), VAD 등의 기능들을 모두 ENABLE 시킨 상태에서 측정하였고, 비트율을 가변시키는 경우와 고정시키는 경우로 나누어서 각각 측정하였다. 비트율을 가변하는 경우, 매 프레임마다 가장 낮은 비트율에서 가장 높은 비트율까지 단계적으로 비트율을 증가시키고, 가장 높은 비트율을 적용한 직후의 프레임에서는 다시 가장 낮은 비트율을 적용하였다. 가변 비트율을 갖도록 실험했을 때의 비트 조정은 그림 5와 같다. 이때, Codec Mode 0는 AMR-WB_6.60 모드를, Codec Mode 8은 AMR-WB_ 23.85 모드를 각각 나타낸다.

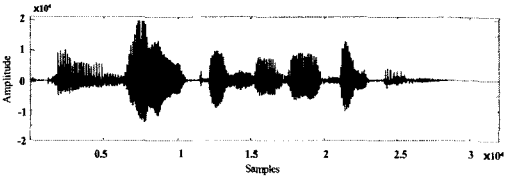


그림 4. 실험에 사용한 테스트 음성데이터

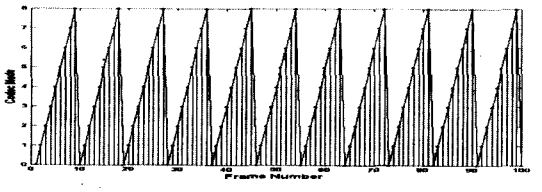


그림 5. 실험에 사용한 가변 비트율 분포

20 ms 길이를 갖는 음성신호 한 프레임의 처리에 소요되는 시간은 TMS320C6201 EVM 보드의 clock cycle rate가 160 MHz 이므로 측정된 클럭 수로부터 식(1)과 같이 구할 수 있으며, 그 결과는 표 3과 같다.

$$Execution\ Time = \frac{cycles}{160M} \times 1000\,[ms] \quad \text{식(1)}$$

표 3. 구현한 음성부호화기의 수행 속도

한 프레임 수행속도	AMR-WB_6.60	가변비트율	AMR-WB_23.85
Cycles	788,414	920,267	1,132,774
시간 [ms]	4.928	5.752	7.080

EVM 보드에서 구현한 AMR-WB 음성부호화기의 디코딩된 결과와 PC에서 시뮬레이션 한 결과가 서로 일치함을 검증하기 위해서 샘플 음성에 대하여 9가지 모드 각각으로 디코딩된 결과 파형을 PC 결과와 비교하였다. 그림 6은 9가지 모드 중에서 23.85kbps 모드에 대한 결과 파형을 보이고 있으며, PC 결과와 DSP 결과가 일치함을 확인할 수 있다.

메모리 할당을 위해 작성한 link-command 파일은 표 4와 같다. 또한 최종 메모리 할당을 위한 map 파일을 만들 때, 프로그램 메모리의 경우 64 kbytes의 제한된 내부 메모리 영역에는 프로그램의 실행시 사용 빈도와 중요성을 고려하여 인코더에서는 메인 코드인 cod_main .obj(.text)와 높은 비트율을 갖는 부호화 모드에서의 코드북 탐색 코드인 c4t64fx.obj(.text)를 배치하였으며, 디코더에서는 메인 코드인 dec_main.obj(.text) 등을 배치하였다. 메모리 맵 파일의 정보를 이용하여 측정한 메모리 사용은 데이터 메모리의 사용량이 16f2a(h) [91.8 kbytes] 이고, 프로그램 메모리의 사용량은 36820(h) [218.0 kbytes]로 나타났다.

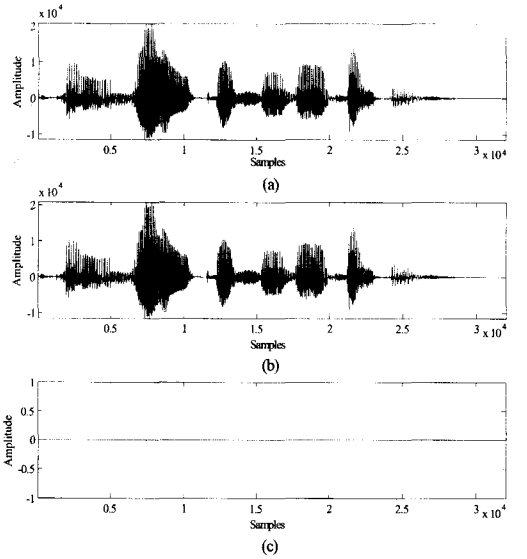


그림 6. PC 연산 결과와 DSP 연산 결과의 비교 (a) PC 연산 결과 파형 (b) DSP 연산 결과 파형 (c) PC와 DSP 연산 결과의 오차

표 4. Link-command 파일

```
-c
-heap       0x2000
-stack      0x4000
MEMORY
{
   INTPROG: origin = 0x00000000, length = 0x10000
   INTDATA: origin = 0x80000000, length = 0x10000
   EXTMEM0: origin = 0x00400000, length = 0x400000
   EXTMEM1: origin = 0x02000000, length = 0x400000
}
SECTIONS
{

   .vec       >          INTPROG
   .text      >          EXTMEM0
   .text_move            { cod_main.obj(.text)
                           c4t64fx.obj (.text)
                           dec_main.obj(.text)
                           isfextrp.obj(.text)
                         }      >     INTPROG

   .cinit          >     INTDATA
   .const          >     INTDATA
   .data           >     INTDATA
   .bss            >     INTDATA
   .stack          >     INTDATA
   .far            >     EXTMEM0
   .cio            >     INTDATA
   .sysmem    >          INTDATA
}
```

Ⅳ. 결 론

본 연구에서는 광대역 음성부호화의 새로운 표준안인 AMR-WB 방식을 분석하고, TMS320C6201 EVM 보드에서의 실시간 구현 결과를 제시하였다. 구현된 음성부호화기의 성능을 평가하기 위해서 프로그램 메모리와 데이터 메모리의 크기, 그리고 한 프레임당 수행 시간을 측정한 결과, 프로그램 메모리는 218 kbytes, 데이터 메모리는 92 kbytes 정도의 크기를 나타내었다. 또한 구현된 AMR- WB 음성부호화기는 한 프레임인 20 ms를 처리하는데 걸리는 cycle 수가 평균 920,267로 측정되었다. 이는, 160 MHz clock rate를 갖는 TMS320C6201 EVM 보드에서 약 5.75 ms에 해당되며, 최대 성능의 28.76 %정도를 차지하여 full-duplex 모드로 세 채널 정도의 음성부호화 시스템을 실시간으로 처리할 수 있음을 확인하였다. 또한, EVM 보드에서 구현한 AMR-WB 음성부호화기의 디코딩된 결과와 PC에서 시뮬레이션 한 결과가 서로 일치함을 검증하였다.

참 고 문 헌

[1] X. Maitre, 7 kHz audio coding within 64 kbits/s," IEEE Journal on Selected Areas in Communications, pp. 283-298, February, 1988.

[2] Wideband speech coding standards and applications," VoiceAge White Paper, September, 2001.

[3] 3GPP TS 26.171 AMR Wideband Speech Codec; General Description," 3GPP Technical Specification.

[4] 3GPP TS 26.190 AMR Wideband Speech Codec; Transcoding Functions," 3GPP Technical Specification.

[5] 3GPP TS 26.193 AMR Wideband Speech Codec; Source Controlled Rate operation," 3GPP Technical Specification.

[6] Y. Bistritz and S. Peller, Immittance spectral pairs (ISP) for speech encoding," International Conference on Acoustics and Signal Processing, vol. 2, pp. 9-12, 1993.

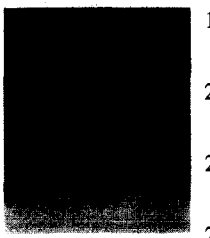
[7] B. Bessette and R. Lefebvre et al., Techniques for high-quality ACELP coding of wideband speech," Eurospeech 2001, pp. 1997-2000, 2001.

[8] Code Composer Studio Getting Started Guide," Texas Instruments Technical Document, November, 2001.

[9] 3GPP TS 26.173 AMR Wideband Speech Codec; ANSI-C code," 3GPP Technical Specification.

[10] TMS320C6000 Programmer's Guide," Texas Instruments Technical Document, February, 2001.

이 승 원(Seung-won Lee) 정회원



1999년 2월: 경북대학교 전자공학과 졸업

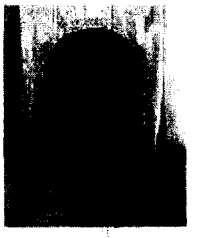
2001년 2월: 경북대학교 전자공학과 석사

2003년 2월: 경북대학교 전자공학과 박사과정 수료

2003년 3월 " 현재: LG전자 DND 영상제품 연구소

<관심분야> 음성신호처리, 디지털신호처리, 음성 및 오디오 코딩 등

배 건 성(Keun-sung Bae) 정회원



1977년 2월: 서울대학교 전자공학과 졸업

1979년 2월: 한국과학기술원 전기 및 전자공학과 석사

1989년 5월: University of Florida 공학박사

1979년 3월" 현재: 경북대학교 전자·전기공학부 교수

<관심분야> 음성신호처리, 디지털신호처리, 디지털통신, 웨이브렛이론, 오디오신호처리 등