TMS320C6701™을 이용한 2.4kbps EHSX 음성 부호화기의 실시간 구현

준회원 양 용 호*, 정회원 이 인 성*, 권 오 주**

Real-time implementation of the 2.4kbps EHSX Speech Coder Using a TMS320C6701™ DSPCore

Yong-ho Yang* *Associate Member*, In-sung Lee*, Oh-Ju kwon** *Regular Members*

요 약

본 논문에서는 TI사의 부동소수점 DSP인 TMS320C6701™을 이용한 2.4kbps EHSX(Enhanced Harmonic Stochastic Excitation) 음성부호화기의 실시간 구현 방법에 대해서 논한다. EHSX는 4khz의 대역폭을 갖는 음성신호를 2.4kbps의 비트율을 갖는 압축 패킷으로 변환하는 부호화 방법으로, 유/무성음에 따라 하모닉(Harmonic) 여기 부호화 방법과 CELP 부호화 방법을 선택적으로 사용하는 구조를 갖는다. 본 논문에서는 이러한 EHSX의 실시간 구현을 위해 연산량의 큰 비중을 차지하는 CELP 분석의 코드북 검색부분과 일부 IIR 필터링 부분에 대한 고정소수점 변환 방법과, 부호화시 하모닉 검색 및 피치 검색방법에 대한 알고리즘 상 연산량 감소 방법, DSP의 구조를 고려한 코드를 배치방법 등 연산량을 감소시키기 위한 최적화 방법을 제시한다. 설계된 음성 부호화기는 PESQ(perceptual evaluation of speech quality) ITU-T Recommendation P.862를 이용한 음질 평가 결과로서 약 MOS 3.28을 얻었으며, 실시간으로 압축 및 복원을 수행한다.

Key Words : voice coder; signal processing; real-time implementation; EHSX; DSP

ABSTRACT

This paper presents an efficient implementation of the 2.4 kbps EHSX(Enhanced Harmonic Stochastic Excitation) speech coder on a TMS320C6701™ floating-point digital signal processor. The EHSX speech codec is based on a harmonic and CELP(Code Excited Linear Prediction) modeling of the excitation signal respectively according to the frame characteristic such as a voiced speech and an unvoiced speech. In this paper, we represent the optimization methods to reduce the complexity for real-time implementation. The complexity in the filtering of a CELP algorithm that is the main part for the EHSX algorithm complexity can be reduced by converting program using floating-point variable to program using fixed-point variable. We also present the efficient optimization methods including the code allocation considering a DSP architecture and the low complexity algorithm of harmonic/pitch search in encoder part. Finally, we obtained the subjective quality of MOS 3.28 from speech quality test using the PESQ(perceptual evaluation of speech quality), ITU-T Recommendation P.862 and could get a goal of realtime operation of the EHSX codec.

Ⅰ. 서 론

4kbps 이하의 저 전송률에서 CELP(Code Excited Linear Prediction) 기반의 음성부호화기는 LP

* 충북대학교 전파공학과 통신신호처리 연구실(forsy@just.chungbuk.ac.kr),
* 충북대학교 전기전자컴퓨터공학부 컴퓨터정보통신연구소(inslee@chungbuk.ac.kr), ** 국방 과학 연구소 (koj@add.re.kr)
논문 번호 : 040176-0507, 접수일자 : 2004년 5월 7일
※ 본 연구는 국방과학 연구소 연구과제(과제번호:03-06-03) 지원으로 수행되었습니다.

(Linear Prediction) 여기 신호의 양자화에 대한 제한된 비트 할당으로 명료한 음질을 얻기에 힘든 것으로 알려져 있다[1]. 최근, 이러한 LP 여기 신호의 효율적인 표현을 위한 모델링 방법이 활발히 연구되고 있으며, 그 중 정현파 생성모델에 바탕을 둔 하모닉 부호화 방법은 주기적인 신호 표현에 대해 효과적인 것으로 알려져 있다.[1][2] 하모닉 부호화에 대한 대표적인 예로는 STC(Sinusoidal Excitation Coding)와 MBE(Multi Band Excitation Coding)를 들 수 있으며, 4kbps 이하에서 좋은 음질을 제공한다. 특히, MELP[3](Mixed Excitation Linear Prediction)와 HSX[4] (Harmonic Stochastic Excitation)등의 음성 부호화기는 더 나은 음질을 얻기 위하여, 주파수 영역에서 대역별 유/무성음 모드나 비주기 신호 비율에 대한 분석 방법을 제공함으로써, 혼합신호에 대한 표현 방법을 보완하고 있다.[3][4] 본 논문에 사용된 EHSX[5]는 혼합신호의 표현을 위해 주기적인 신호 특성을 갖는 유성음 부분에는 주파수 영역 분석방법인 하모닉 부호화방법을, 비주기 적인 특성을 갖는 무성음 부분에는 CELP 부호화 방법을 사용함으로써, 각기 다른 특성을 갖는 신호에 대하여 다른 모델을 적용함으로써 모델링 오차를 줄이고 있다. 또한, 전이 구간에서의 고주파 비주기 신호를 표현하기 위하여 잡음 발생기를 사용함으로써 기계적인 소리를 감소시킨다. 비트율에 있어서는 평균 비트율을 감소시키기 위해 묵음구간 부호화 방법을 따로 두고 있다. 이러한, 특성을 갖는 부호화기는 각기 다른 모드에 따라 다른 계산 량을 갖고 있고 하나의 모델에 대해 복잡도가 최적화 되어 있는 구조가 아니기 때문에 상당히 큰 계산량을 갖는다. 복잡도를 줄이기 위한 방법으로는 알고리즘 상 음질의 열화가 적고 계산량이 많은 부분과 고정소수점으로 바꾸었을 때 계산량의 감소가 큰 부분에 대하여 중점적으로 이루어지며, DSP의 구조를 고려한 코드 배치 방법등이 사용된다.

본 논문에서는 EHSX[5]에 대한 내용과 알고리즘 상의 복잡도 감소 방법을 2장에서 설명하고, 사용된 DSP의 특징에 대해 3장에서 설명한다. 실시간 구현을 위해 쓰여진 방법과 효과에 대해서 4장에서 설명을 한 뒤 5장에서 결론을 맺는다.

Ⅱ. EHSX 음성부호화기

2.1. EHSX 인코더

EHSX는 4khz의 대역폭을 갖는 음성신호를 2.4kbps의 비트율을 갖는 압축 패킷으로 변환하는 부호화방법을 제공한다. 기본적으로 묵음 구간을 제외한 구조는 그림 1과 같이 유/무성음 모드에 따라 하모닉 여기 부호화(Harmonic Excitation Coding) 방법과 CELP 부호화 방법을 선택적으로 사용하는 구조로 구성되어 있다. 전체구조는 묵음구간, 유성음구간, 무성음구간에 따라 다른 구조의 부호화방법을 제공한다. 우선 분석 차례에 따라 묵음구간 검출을 먼저하고 V/UV 검출과정이 이루어진다. 묵음구간일 경우 과거 프레임의 파라미터 상태에 대해 영향을 받을 수 있는 LSP 양자화 과정 및 피치검색이 이루어지고, 양자화 파라미터는 전송되지 않는다. 묵음구간이 아닐 경우에 대해서는 LPC분석 및 LSP 양자화, 피치검색의 분석과정 후 V/UV 검출과정이 이루어진다. V/UV 판단 후 유성음일 경우는 LPC 잔여신호의 하모닉 구조를 표현하는 스펙트럼 크기 값의 파라미터를 추출하는 분석과정이 이루어진다. 무성음일 경우는 피치 분석과정이 생략된 CELP 분석과정이 이루어진다. 표1, 그림2는 각 모드에서의 전송 파라미터의 비트 할당과 분석 프레임의 구조를 나타낸다

2.2. EHSX 디코더

복호화 단에서는 묵음구간에서는 LSP 파라미터에 대한 보간이 이루어지고, 그림 3과 같이 유성음구간에서는 IFFT를 이용한 정현파 합성이, 무성음구간에서는 각 코드 인덱스에 따른 여기신호의 파형이 선택되어 각 모드에 따른 LPC합성과정과 후단 여파기(Post-filter)를 통해 최종 복원된 신호를 얻는다.

제안된 2.4kbps 음성부호화기는 하모닉 스펙트럼 및 노이즈 스펙트럼 정보를 추출하기 위해 고 해상도를 가지는 512 point FFT를 취하게 되며, 미래샘플 48 샘플에 0을 첨가하여 사용한다. 부호화 지연

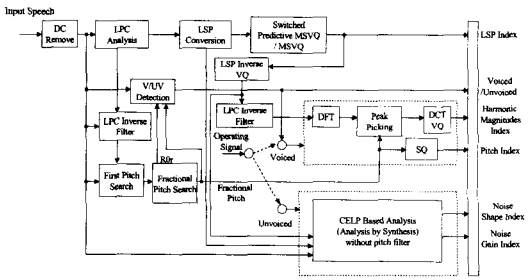


그림 1. EHSX 부호화기 블록도

은 26ms 이며, 디코딩 지연은 선형 보간 합성과 분석프레임의 중심 값에 기인하여 10ms가 된다. 총 지연 값은 36ms가 된다.

표 1. 부호화 파라미터에 대한 비트 할당

			Silence	전이구간 (Silence count=1)
1차 상태 : 6비트				1차 상태 : 6비트
2차 상태 : 10비트				2차 상태 : 10비트
1비트			2비트	2비트
무성음		유성음		
LSF	6비트			LSF 6비트
10 ms	형태0:8비트 이득0:4비트	피치:7비트 형태0:5비트	6비트 (모두0)	10 ms 형태0:8비트 이득0:4비트
10 ms	형태3:8비트 이득3:4비트	형태1:5비트 이득:5비트		10 ms 형태1:8비트 이득1:4비트
		잔차:8비트		
48bit				48bit

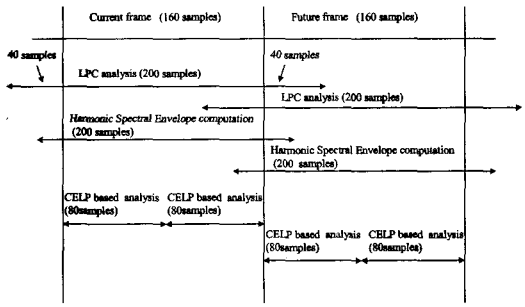


그림 2. EHSX 프레임 구조

2.3. 피치검색 및 하모닉 검출에 대한 복잡도 감소 방법

피치 검색은 1차 피치 검색과 2차 Fine 피치 검색으로 나뉘는데 1차 검색은 일반적인 자기 상관 방법을 사용한다. 이러한, 1차 피치 값을 토대로 구하여진 기본주파수(fundamental frequency)의 반 간격으로 각 하모닉 주파수 마다 l번째 하모닉에 대한 $e_l(T)$를 최소화 하는 최대 피크치를 찾는 기법을 사용하는데 이때, 검출된 하모닉에 대한 주파수 값과 그 크기 값이 파라미터로 이용된다. 여기서 계산량은 검색범위인 T의 값에 따라 좌우되는데, 각 T마다 w_0의 범위의 곱셈 과정이 있기 때문에 많은 계산량을 요구한다. 이러한 과정은 다음 식 (2.1)과 같이 검색범위를 $w_0 /2$ 의 범위로 좁힘으로써, 복잡도를 감소시킬 수 있다. 이때, 검색되지 않는 나머지 반은 스펙트럼에서 에너지가 작은 부분

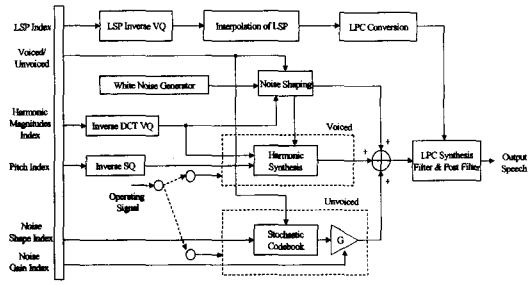


그림 3. EHSX 복호화기 블록도

(spectral valley)이기 때문에 피치 검색의 정확도에 큰 영향을 미치지 않는다.

$$e_l(T) = \sum_{i=a_l+\frac{w_0}{4}+T}^{b_l-\frac{w_0}{4}+T} (|X(i)| - |A_l||B(i)|)^2 \tag{2.1}$$

$$A_l(T) = \frac{\sum_{j=a_l+\frac{w_0}{4}+T}^{b_l-\frac{w_0}{4}+T} |X(j)||B(j)|}{\sum_{j=a_l+\frac{w_0}{4}+T}^{b_l-\frac{w_0}{4}+T} |B(j)|^2} \tag{2.2}$$

w_0는 샘플단위의 피치 주파수이며 a_m과 b_m의 각 하모닉에 대한 주파수 시작과 끝이 된다. 여기서 T값은 $-w_0 /2$에서 $w_0 /2$ 사이에서 일정간격으로 추출된 값으로 기본주파수의 후보가 된다.

그러나 위에서 설명한 스펙트럼을 이용한 2차 피치검색 법은 다소의 복잡도를 감소시켰음에도 불구하고 각 피치 후보 당 원본 값과의 스펙트럼 오차를 계산하기 위한 반복계산으로 인하여 큰 계산 량을 가지게 된다. 또한 고정 블록 분석법으로 인한 피치 오차 값을 유발시킬 수 있다. 이러한 점을 고려하여, 또 다른 Fine 피치검색 법을 모색하였다. 정규화된 자기 상관 값으로부터 구한 1차 피치 값과 1kHz 6차 Butter-worth 저대역 필터로 저주파 필터링된 선형예측 잔차신호를 입력으로 그림 4와 같이 프레임 중앙위치에서의 NCF(normalized

cross-correlation function)을 구하게 된다. 이때 분석 샘플 수는 2차 피치 후보 값이 된다.

이러한 과정은 전 단계에서 구한 NCF값을 이용 가능하기 때문에 빠른 피치검색 법을 가능하게 만든다. 이때 Fine 피치 검색을 위한 범위는 제한되는 look ahead 값으로 인해 20과 120샘플 사이로 한정된다. 그림 4는 NCF를 이용한 Fractional 피치 추출에 사용되는 프레임 구조를 나타낸 것이다. 그림 4에서 보는 바와 같이 프레임의 중심 포인트로부터 후보 피치길이 τ만큼 양쪽으로 쉬프트된 두 블록에 대한 교차 상관 값을 구한다. 구하여진 교차 상관 값이 최대가 되는 τ가 최종 분수(Fractional) 피치 값이 된다. 이때 각 분수(Fraction) 피치 후보 값에 대한 교차상관 값은 식(2.3)과 같은 보간 함수를 사용하여 구한다. 분수 피치의 정수 값인 T와 T+1 사이의 값들에 대한 교차상관 값을 r(T+Δ)라 정의하면, 최대 값을 가지는 T+Δ에서 Δ값은 식(2.4)와 같이 계산될 수 있다.

$$r(T+\Delta)=\frac{(1-\Delta)c_T(0,T)+\Delta c_T(0,T+1)}{\sqrt{c_T(0,0)[(1-\Delta)^2c_T(T,T)+2\Delta(1-\Delta)c_T(T,T+1)+\Delta^2c_T(T+1,T+1)]}} \tag{2.3}$$

$$\Delta=\frac{c_T(0,T+1)c_T(T,T)-c_T(0,T)c_T(T,T+1)}{c_T(0,T+1)[c_T(T,T)-c_T(T,T+1)]+c_T(0,T)[c_T(T+1,T+1)-c_T(T,T+1)]} \tag{2.4}$$

여기서,

$$c_\tau(m,n)=\sum_{k=-\lfloor\tau/2\rfloor-80}^{-\lfloor\tau/2\rfloor+79}S_{k+m}S_{k+n} \tag{2.5}$$

이고, 정수값을 제외한 나머지 부분인 Δ값은 분수(Fraction) 부분을 뜻하며, r(T+Δ) 값은 T+Δ에서의 교차 상관 값에 대한 최대값이 된다.

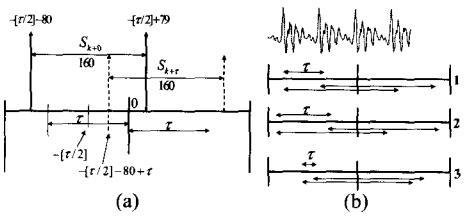


그림 4. (a) NCF값을 구하기 위한 프레임 구조,
(b) 피치 값에 따른 NCF 적용 예

Ⅲ. 실시간 구현

3.1. 부동소수점 DSP TMS320C6701™의 특징

EHSX 음성 부호화기의 실시간 구현에 사용된 Texas Instruments®(TI)사의 부동 소수점 DSP인 TMS320C6701™ DSP[6][7]는 VelociTI, 즉, 진보된 VLIW(Very-Long-Instruction-Word) 구조를 갖는 DSP로서 다음과 같은 특징을 가지고 있다.

- 150, 167MHz의 동작 주파수(최대 1GFLOPS 연산).
- 32개의 32비트 범용 레지스터.
- 8개의 연산모듈을 가지고 있으며, 한 사이클당 8개의 32비트 명령을 병렬로 수행.
- 8/16/32비트 데이터 지원(효율적 메모리 관리).
- 40비트 산술 연산이 가능하여 고정밀도의 응용에 적합.
- 다채널 DMA(Direct Memory Access), 다채널 BSP (Buffered Serial Port)
- 대용량 내부 RAM(프로그램/데이터)
- 고효율의 C 컴파일러, Assembly optimizer 제공으로 인한 개발 기간 단축.

전체적인 구조는 그림 5와 같으며, 내부 메모리로써 64Kbyte의 프로그램 메모리와 64Kbyte의 데이터 메모리를 가지고 있고, 프로그램 메모리는 실행 코드가 외부메모리에 있을 경우 캐쉬로서도 사용된다.

CPU core는 두 개의 데이터 경로(경로 A, 경로 B)를 가지고 있다. 각각의 데이터 경로는 4개의 연산모듈을 가지고 있으며 각각의 명칭은 .L, .S, .M,

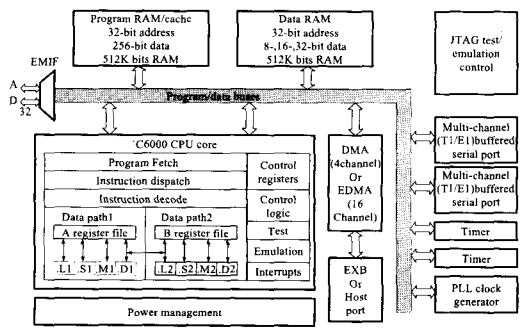


그림 5. TMS320C67x의 블록도

.D 이다. 연산모듈은 논리연산, 시프트 연산, 곱셈 연산, 데이터 어스레스 연산등을 수행하며, 연산시 모든 명령어들은 레지스터 파일을 통해서 로드되고 저장된다.

C67x의 큰 장점은 Assembly Optimizer에 의해서 C 코드만을 최적화하여도 원하는 성능을 얻을 수 있어서 개발 기간을 단축시킬 수 있다.

3.2. 최적화 방법

일반적으로 음성 코덱의 실시간 구현은 저가의 고정 소수점 DSP를 사용하지만, 본 논문의 경우에는 음질 향상 측면 뿐 만 아니라 EHSX 음성 부호화기가 위성단말기와의 연동을 목표로 하는 하나의 객체에 해당하며 다른 객체와의 처리를 목표로 하였기 때문에 부동 소수점 DSP를 사용하였다. EHSX 음성부호화기 알고리즘을 하나의 '6701 DSP 코어에서 실시간 처리하기 위해서는 20ms의 프레임 크기에 대해 최대 3.2million cycle 내에서 처리가 가능하여야 한다.

EHSX 음성부호화기의 최적화를 위해, 부동소수점 ANSI-C 소스의 알고리즘을 PC상에서 시뮬레이션을 통해 검증하였고, 그 후 TI사의 Code Composer Studio (CCS)와 TMS320C6701 EVM 보드를 사용했다. 효율적인 구현을 위해 각 메모리의 액세스 속도를 테스트 하여 코드를 배치하였으며, 모듈별로 분석을 하여 전체 프로그램의 call structure를 작성하였다. 그 후에 TI와 호환되지 않는 헤더파일과 함수들을 제거하고 모듈별로 연산량을 계산하였다. 전체적인 연산량을 감소시키기 위하여 컴파일러 최적화 옵션을 적용하였고, TI에서 제공되는 최적화 라이브러리와 캐쉬기능을 사용했다. 복잡도가 큰 함수의 부분 최적화를 위해서는 선형 어셈블리와 소프트웨어 파이프라인을 이용한 방법과 Q-function을 사용하여 코드를 부동소수점 연산에서 고정소수점 연산으로 변환하는 방법을 사용했으며, 기본적인 고정소수점 수학함수는 TI에서 제공되는 Intrinsics 함수를 사용하였다.

일반적인 최적화 기법을 좀더 상세히 설명하면 다음과 같다.

- 계산결과가 변하지 않는 초기화 루틴이나 테이블들은 배열로서 선언해 값을 넣어준다.
- Word-wide연산과 loop-unrolling을 적절히 사용하여 최적화 한다.
- 모든 루프에서는 가장 내부에 있는 루프 또는 함수만 -o3 옵션으로 자동 최적화되므로 루프 안에서는 가능한 함수 호출이나 루프를 사용하지 않는다.
- 루프에서 반복 횟수가 충분히 작으면 루프를 제거하고 반복횟수만큼 코드를 직접 전개한다.
- 다중 루프에서 외부 루프의 반복 횟수가 작은 경우에는 외부 루프를 제거하고 반복 횟수만큼 내부 루프를 순차적으로 반복한다.
- Restrict나 MUST_ITERATE과 같은 컴파일러 지시어를 이용하여 C소스 레벨 상에서 최적화를 수행한다.

1) Q-function을 이용한 최적화

표 2와 3는 부동소수점 연산에서 고정소수점 연산으로의 변환 예를 보여준다. 일반적으로 고정수소점 명령어의 Delay가 부동소수점 연산 명령어의 Delay보다 적기 때문에 Q-function을 이용한 변환은 복잡도 감소에 크게 영향을 준다. 다음의 zssynth함수는 연산량의 큰 비중을 차지하는 부분으로써 multiply 연산 명령의 경우 부동소수점 연산명령의 Delay는 4이고, 고정소수점 연산명령의 경우는 Delay가 1이다. 아래 소스의 경우에는 다중 반복처리로 인하여 많은 계산량을 필요로 하며, Q-function을 사용하여 코드를 변환한 후 표 4와 같은 계산량의 감소를 나타냈다.

표 2. 부동소수점 연산 소스

```
void zssynth(float *in,float *out,float *lpcfir,float *qlsp,short sw)
{
 for(i=0;i<=NFIR;i++) pres[i+FLTBUF-num-NFIR]=0.0;
 for(i=0;i<num;i++) {
  pres[i+FLTBUF-num]=in[i];
  for(temp=0.0,j=0;j<=NFIR;j++)
  temp+=pres[i+FLTBUF-num-j]*lpcfir[j];
  s[i]=temp;
 }
}
```

표 3. 고정소수점 연산 소스

```
void zssynth_fx(short *in,short *out,short *lpcfir,short *qlsp,short sw)
{
 for(i=0;i<=NFIR;i++) pres[i+FLTBUF-num-NFIR]=0;
 for(i=0;i<num;i++) {
  pres[i+FLTBUF-num] = in[i];
  temp = L_mult(pres[i+FLTBUF-num], lpcfir[0]);  //Q12+12+1 = 25
  for(j=1; j<=NFIR; j++)
        temp=L_msu(temp, pres[i+FLTBUF-num-j], negate(lpcfir[j]));
  temp = L_shl(temp, 3);
  s[i] = round(temp);
 }
}
```

표 4. zssynth 함수의 복잡도 비교

cycle	528000	224400

2) FastRTS67x 라이브러리의 이용

최적화 방법의 다른 예는 TI에서 제공해주는 최적화된 수학함수인 FastRTS67x 라이브러리[8]를 이용하는 것이다. FastRTS67x 라이브러리는 C67x를 위한 최적화된 26개의 부동소수점 연산 수학함수의 모음이다. 최적화된 연산속도를 갖으며 실시간 응용의 여러 측면에서 사용된다.

FastRTS 라이브러리의 특징은 다음과 같다.

- 최적화된 핸드 어셈블 루틴을 생성한다.
- C6x 컴파일러와 호환되는 C-callable 루틴이다.
- 제공되는 함수는 기존의 RTS함수와 C모델에 의하여 테스트된 함수이다.

다음 표 5는 FastRTS 라이브러리 함수의 일부 예이다.

FastRTS 라이브러리 함수는 기존의 수학함수에 비해서 약 3배정도의 성능향상을 가져오며, 소스 코드의 수정 없이 라이브러리를 링크시키는 것만으로, 명령어가 대체되어 연산 속도가 향상된다.

표 5. FastRTS 라이브러리 함수

arc tangent of one argument	atanf	atan
cosine of a radian argument	cosf	cos
exponential base e	expf	exp
logarithm base e	logf	log
power=X raised to power Y	powf	pow
sine of a radian argument	sinf	sin

3) 캐쉬와 내부데이터 메모리의 효율적 사용

다음 표 6는 각 메모리 위치에서의 wait state를 나타낸다. DSP 프로세서 코어에서 I/O 동작을 수행할 경우에는 매 사이클마다 여러개의 명령어를 수행하는 것은 불가능하다. 또한 빈번한 외부 메모리 액세스를 필요로 하기 때문에 결국 프로세서의 성능 저하를 가져온다. 결국 DSP의 최대 성능은 프로그램 코드와 데이터가 내부 메모리 영역에 있을 때 얻을 수 있다.

EHSX의 컴파일된 실행코드와 데이터는 큰 용량으로 인해 DSP 내부 메모리에 모두 로드할 수가 없다. 따라서 내부 프로그램 메모리는 캐쉬로 사용했으며, 데이터 메모리는 액세스가 많은 변수와 데이터를 우선순위로 다음의 예와 같이 컴파일러 지시어를 사용하여 변수나 테이블을 데이터 메모리로 로드하여 데이터 메모리의 사용률을 증가시켰다.

다음의 예는 변수 x를 데이터 섹션 .my_data로 할당하는 역할을 한다.

예) #Pragma DATA_SECTION(x ,".my_data")

프로그램 코드에 대해서는 C6000 에서는 내부 프로그램 메모리를 캐쉬로 사용할 수 있어 메모리 로딩과 관련된 오버헤드 없이 프로그램을 실행시킬 수 있다.

표 6. 메모리 위치에 따른 DSP 코어 wait state

internal	internal	4108	0
internal	SBSRAM	28684	7
internal	SDRAM	35567	9
SBSRAM	internal	57330	14
SBSRAM	SBSRAM	81906	20
SBSRAM	SDRAM	91297	22
SDRAM	internal	74431	18
SDRAM	SBSRAM	99525	24
SDRAM	SDRAM	205313	50

4) FFT 함수의 어셈블리 코드 적용

최적화된 FFT 어셈블리 코드를 적용하여 최적화를 수행하였다. TI FFT 어셈블리 코드는 입력 데이터 x, 계수 w, 길이 n의 3개의 파라미터를 갖는다. 계수 w를 생성하는 코드는 FFT의 길이에 따라 고정된 값을 생성하므로, 사용된 FFT의 길이에 해당하는 계수 테이블을 만들어서 사용했다. 이때 문제가 되는 것은 FFT 연산에 필요한 테이블의 메모리 위치가 문제가 된다. 어셈블리 루틴에서는 성능을 향상시키기 위해서 더블워드(8byte)로 데이터를 액세스 한다. 따라서 데이터를 메모리에 할당 시 더블워드 경계에 데이터를 위치해야 한다. 그렇게 하기 위해서 다음 예와 같이 처리한다.

예) #Pragma DATA_ALIGN(w ,8)

여기서, 변수 혹은 테이블 w는 더블워드를 경계로 하는 메모리에 위치하게 된다. 또한, 더블워드

(LDDW) 명령어가 내부 메모리 영역에서만 사용할 수 있는 제약이 있기 때문에 FFT 함수의 테이블과 데이터들을 내부 데이터 메모리에 위치 시켜야 한다. 다음 표 7, 8은 중간 최적화 결과와 설명한 최적화 방법들을 적용한 EHSX 음성부호화기의 중요 부분의 최적화 결과를 나타낸다.

표 7. EHSX 보코더 중간 최적화 결과

Encoding	40,928,816	8,400,539	3,967,697	90
Decoding	15,157,071	3,749,810	2,045,282	86.5
Total	56,085,887	12,150,349	6,012,979	89

$$* \; optimization\ ratio = \frac{unoptimization - opti}{unoptimizatio}$$

표 8. EHSX 보코더 최적화 결과

LPC analysis & LSP conversion	615,430	179,596	70.8
Pitch Estimation & Fine pitch search	1,799,520	829,146	53.9
Spectral envelope estimation	1,798,568	637,204	64.6
CELP analysis	704,749	388,079	44.9
Voiced excitation decoding	1,402,151	757,197	46
Unvoiced excitation decoding	363,343	191,958	47.2
Voiced LPC Filtering	507,073	321,561	36.6
Unvoiced LPC Filtering	343,956	212,184	38.3

Ⅳ. 성능 평가

아래의 표 9와 그림 6, 7은 EHSX 음성부호화기의 MOS 테스트 결과와 입력 음성 그리고 최적화된 음성부호화기로 처리된 결과음성을 나타낸다.

표 9. EHSX 보코더 MOS 테스트 결과

Original speech	4.5	4.5	4.5
8kbps CS-ACELP	3.732	3.924	3.828
2.4kbps MELP	2.975	3.288	3.132
2.4kbps EHSX	3.132	3.418	3.275

그림 6. 원본 음성 파형과 스펙트럼

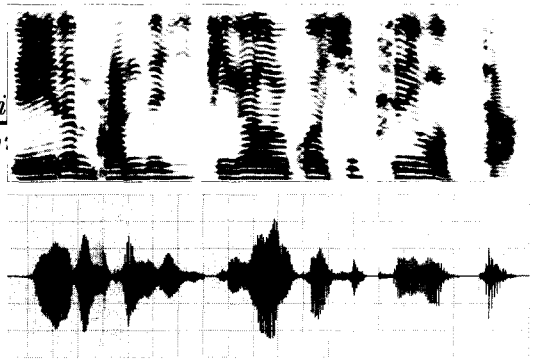


그림 7. 복원된 음성 파형과 스펙트럼.

MOS 테스트는 'PESQ, ITU-T recommendation p.862 Version 1.2-2 August 2002'를 프로그램을 이용하여 결과를 얻었다. 실험에 사용된 표준 음성은 발화자가 각각 다른 8개의 남자 음성과 8개의 여자 음성으로 테스트 했으며, 2.4kbps MELP와 비교했을 때 더 우수한 성능이 나타났다. 또한 신호대 잡음비(SNR)의 계산 결과는 약 90dB 정도를 나타냈고, 최적화 작업으로 인해서 발생한 결과로서 음질에 미치는 영향은 무시할 수준이다.

실시간 구현의 결과로서는 최적화 옵션 -o3와 내부 프로그램 메모리를 캐쉬메모리로 사용함에 따른 성능향상이 원시 C소스에 비해서 89%정도의 성능향상이 나타났고, 메모리 load와 store의 overhead를 줄이는 내부데이터 메모리의 사용이 성능 향상에 많은 영향을 준다. FastRTS 라이브러리, FFT 어셈블리 소스적용, 복잡도 큰 코드의 어셈블리 변환과 소프트웨어 파이프라인 코딩으로 인해서 중간 최적화 결과의 약 55%의 성능향상을 나타냈으며, 실시간 동작이 가능하다. TI FFT 어셈블리 코드는 원본 FFT 소스의 1/3의 복잡도를 나타냈고, 선형어셈블리 코딩과 소프트웨어 파이프라인 코딩은 루프의

반복 횟수가 많은 부분에 대해서 큰 최적화 효과를 나타냈다.

사용된 메모리는 외부 프로그램 메모리가 198Kbyte, 데이터 메모리가 외부 55Kbyte, 내부 47Kbyte가 사용되어 총 102Kbyte 사용되었다. 아래의 표 10, 11은 최적화 작업의 평균 수행시간과 사용된 메모리를 나타낸다.

표 10. EHSX 최적화 결과 평균 수행시간

Cache / -o3 option	6,223,057
내부데이터메모리 효율적사용	2,829,286
FFT 어셈블리 코드 적용	
부분 어셈블리 코딩	
FastRTS Library 사용	

표 11. EHSX 보코더 구현시 사용된 메모리

Code memory	198,088	
Data memory	internal	46,555
	external	54,878

V. 결 론

본 논문에서는 2.4kbps 저전송율을 갖는 EHSX (Enhanced Harmonic Stochastic Excitation) 음성 부호화기의 피치검색에 대한 알고리즘상 연산량감소 방법 및 실시간 구현 결과에 대해 논하였다. EHSX는 유/무성음에 따라 하모닉(Harmonic) 여기 부호화 방법과 CELP 부호화 방법을 선택적으로 사용하는 구조로 구성되어 있으며, 묵음구간 부호화 방법도 제공한다. 구현에 사용된 TMS320C6701™ DSP는 고속의 부동소수점 DSP로서, 고정밀도를 요하는 많은 분야에 응용되어 쓰이고 있다.

EHSX의 실시간 구현을 위해 연산량의 큰 비중을 차지하는 CELP 분석의 코드북 검색 부분과 일부 IIR 필터링 부분에 대한 고정소수점변환 방법과, 부호화시 하모닉 검색 및 피치 검색방법에 대한 알고리즘상 연산량 감소 방법, DSP의 구조를 고려한 코드 배치방법 등 연산량을 감소시키기 위한 최적화 방법을 제시했다.

구현된 음성부호화기는 주관적 음질 평가 결과 MOS 3.28을 나타냈으며, 2.4kbps MELP 음성부호화기 보다 더 우수한 결과를 나타냈고, 위성통신 단말기나 이동통신 단말기로 사용 가능하다. 단계별 최적화 단계를 거쳐서 최종 최적화 소스를 생성한 결과는 원본 소스에 비해서 약 95%의 성능향상을 가져왔다. 사용된 메모리는 코드 메모리 198Kbyte, 데이터 메모리 102Kbyte가 사용되었으며, 실시간으로 동작이 가능하다.

본 연구는 위성통신 단말기와의 연동을 위해 중간 객체의 개발단계로써 부동소수점 DSP를 사용하였으나, 저가의 고정소수점 DSP를 이용한 코덱 알고리즘의 최적화가 필요하며, 주변 객체와의 연동을 위한 시스템을 개발을 할 예정이다.

참 고 문 헌

[1] R. V. Cox, "Speech Coding Standards", *Speech Coding and Synthesis*, Chapter 2, W. B. Kleijn, and K. K. Paliwell Eds., Elsevier, 1995

[2] A. M. Kondoz, "Coding Strategies and Standards", *Digital Speech*, Chapter 5, John Wiley, 1994.

[3] A. V. McCree, K. Trung, E, B, George, T.P.Banwell and V. Viswanathan, " A 2.4kbit/s MELP Coder Candidate for the New U.S. Federal Standard", *in Proc IEEE Int. Conf. Acoust., Speech, Signal Processing,* vol.1 ,pp 200-203, May 1996.

[4] C.Laflamme, R. Slami, R.Matmi, J-P. Adoul, "Harmonic-Stochastic Excitation(HSX) Speech Coding Below 4Kbit/s", *in Proc IEEE Int. Conf. Acoust., Speech, Signal Processing,* vol.1, pp 204-207, May 1996.

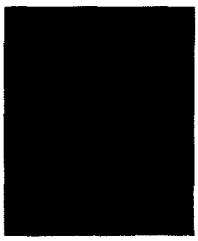
[5] 김종학, 이인성 "하모닉 코딩과 CELP 방법을 이용한 저 전송률 음성 부호화 방법 Low Rate Speech Coding Using the Harmonic Coding Combined with CELP Coding", *한국 음향 학회*, April 2000.

[6] Nasser Kehtarnavaz, Mansour Keramat, *DSP system design using the TMS320C6000*, Prentice Hall, 2001.

[7] Texas Instruments, *TMS320C6000 CPU and Instruction Set Reference Guide*, Literature ID# SPRU 189C, 1998.

[8] Texas Instruments, *TMS320C67x FastRTS Library Programmer's Reference*, Literature ID# SPRU 100A, 2002.

양 용 호 (Yong-ho Yang) 준회원

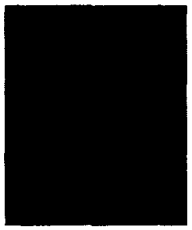


2002년 2월 : 충주대학교 전자공학과(공학사)
2002년 3월~ 현재 :충북대학교 전파공학과 석사과정
<관심분야> 음성오디오 부호화, DSP 실시간 구현

이 인 성(In-sung Lee) 정회원

1983년 2월 : 연세대학교 전자공학과학사
1985년 2월: 연세대학교 전자공학과석사
1992년 12월 : Texas A&M University 전기공학과 박사
1986년 5월~ 1987년 7월 : 한국통신 연구개발단 전임연구원
1993년2월~1995년 9월 : 한국전자통신연구원 이동통신기술연구단 선임연구원
1995년 ~ 현재 충북대학교 전기전자 및 컴퓨터 공학부 부교수
<관심분야> 음성 및 영상 신호압축, OCDMA, 적응필터, 이동통신

권 오 주 (Oh-Ju Kwon) 정회원



1989년 2월 : 경북대학교 전자공학과 학사
1993년 2월: 경북대학교 전자공학과 석사
2002년 2월 : 경북대학교 전자공학과 박사
1997년 9월~1999년 1월 : Matra Marconi Space UK 연구원
1993년 3월 ~ 현재 : 국방과학연구소 선임연구원
<관심분야> 위성통신, 디지털통신신호처리, 다중반송파신호처리, SDR, OFDM