한국음향학회지 제23권 제5호 pp. 420~427 (2004)

부동 소수점 DSP를 이용한 4kbps EHSX 음성 부호화기의 실시간 구현

Real-Time Implementation of the EHSX Speech Coder Using a Floating Point DSP

김 정 호*, 이 인 성*, 박 동 원**
(Jeong-Ho Kim*, In-Sung Lee*, Dong-Won Park**)

*충북대학교 전파공학과, **배재대학교 IT공학부

(접수일자: 2004년 5월 3일; 수정일자: 2004년 6월 24일; 채택일자: 2004년 7월 14일)

본 논문에서는 선형예측 잔여신호에 대한 하모닉 벡터 여기 코딩과 시간 대역 분리 혼합 코딩을 결합한 4kbps EHSX (Enhanced Harmonic Stochastic Excitation) 음성부호화기 실시간 구현한 내용을 기술한다. 유성음 구간에서는 하모닉 여기 코딩에 무성음 구간에 대해서는 분석-합성 구조의 벡터 여기 코딩을 사용하였으며, 유/무성음이 혼재하는 전이구간에서는 시간 분리 전이 코딩을 사용하였다. 이 음성부호화기 구현을 위해 부동소수점과 고정소수점을 모두 지원하는 DSP인 TMS320C6701 을 사용하였고, 연산량을 줄이기 위해 IFFT를 사용한 저 복잡도 정현파 합성법을 사용하여 알고리즘의 최적화를 이루었으며, 복잡도의 문제가 되는 부분을 고정소수점으로 변환한 후 파이프라인을 적용한 핸드 어셈블리 코딩을 하여 구현에서의 최적화를 이루었다. 또한, 메모리의 효율성을 극대화하기 위해 캐쉬 메모리 할당과 데이터를 내부 메모리에 할당하였고 수학 연산의 최적화를 위해 FastRTS67x 라이브러리를 사용하였다. 개발 환경은 DSP EVM 보드를 사용하였으며 음성 신호의 입출력 확인으로 동작 및 기능을 검증하여 실시간 구현하였다.

핵심용어: EHSX 음성부호화기, 하모닉 벡터 여기, 전이, 최적화

투고분야: 음성처리분야 (2.4)

This paper presents real time implementation of 4kbps EHSX (Enhanced Harmonic Stochastic Excitation) speech coder that combines the harmonic vector excitation coding with time-separated transition coding. The harmonic vector excitation coding uses the harmonic excitation coding for voiced frames and used the vector excitation coding with the structure of analysis-by-synthesis for unvoiced frames, respectively. For transition frames mixed with voiced and unvoiced signal, we use the time-separated transition coding. In this paper, we present the optimization methods of implementation speech coder on the EMS320C6701[R] DSP. To reduce the complex for real-time implementation, we perform the optimization method in algorithm by replacing the complex sinusoidal synthesis method with IFFT, and we apply fully pipelines hand assembly coding after converting it from floating source to fixed source. To generate a more efficient code, we also make use of the available EMS320C6701[R] resources such as Fastest67x library and memory organization.

Keywords: EHSX (Enhanced Harmonic Stochastic Excitation), Harmonic vector exciting, vector excitation, Transition, Optimization

ASK subject classification: Speech signal processing (2.4)

I. 서 론

음성통신을 위해 음성을 전송하거나 디지털 녹음기, 인터넷폰과 같이 음성을 저장하는 음성처리 시스템에 대한 수요가 급증하고 있다. 이러한 시스템에서 제한된 미디어를 효율적으로 사용하기 위해서는 음성을 압축/복원하는 음성부호화 (speech coding)가 가장 핵심적인 기술이다. 음성부호화는 입력된 음성신호로부터 신호가 갖고 있는 상관도를 바탕으로 주요 파라미터를 추출하여 이를

책임저자: 김 정 호 (orijjoi@hanmail.net)
361-763 충북 청주시 흥덕구 개신동 산 48번지
충북대학교 공과대학 전파공학과
(전화: 043-261-3223; 팩스: 043-271-4647)

부호화하는 과정이다. 음성 부호화 알고리듬은 다양한 방법이 있지만, 대부분의 저 전송률음성부호화 표준안은 시간 축에서 이루어지는 분석에 의한 합성 방법에 기반한 CELP (Code-Excited Linear Prediction)구조가 주류를 이루고 있다[1-3]. 최근에 여기 필터에 분석-합성 방법을 사용하여 저 전송률에서도 고음질의 음성을 복원할 수 있게 되었다. 그러나 저 전송률에서의 CELP구조는 어느 정도의 좋은 음질을 보이긴 하나 연산량이 많아 처리시간이 오래 걸린다는 단점을 가진다. 이런 단점을 보다 효과적으로 보완하기 위해 선형예측 잔여신호의 주파수 영역 하모닉 코딩 방법이 제안되었으며, 하모닉 코딩은 유성음의 주기적인 신호 복원에 대해서 적합하고, 주파수가 대략 100Hz인 작은 무성음에서도 효과적이다. 하지만 이 하모닉 모델은 단점이 있는데 유성음의 개시, 파열음, 비주기적인 펄스 부분인 전이구간에서 비효율적인 면을 보인다[4]. 이러한 문제점을 보완한 혼합 방식의 부호화에 대한 연구가 진행되고 있으며 본 논문에서는 전이구간에서 시간기점을 사이로 하모닉 코딩과 벡터 여기 코딩을 함께 쓰는 방법을 사용하였고, 기존의 선형예측 기반 하모닉 코딩의 유/무성음 혼재 구간을 유성음 단독 구간으로 하여 얻어지는 기계적인 소리와 하모닉 크기의 벡터 양자화 과정에서의 비효율적인 비트 할당 등의 문제점을 보완한 4kbps EHSX 음성부호화기를 제안하여 왔다[2,3]. 본 논문에서는 실시간 구현을 위해 두 가지 접근 방식으로 접근하였는데 첫째는, IFFT를 사용한 저 복잡도 정현파 합성법을 사용하여 알고리즘의 최적화를 이루었다[5,6]. 둘째는, 연산량이 많은 부분에서는 파이프라인을 적용한 핸드 어셈블리 코딩을 하여 구현에서의 최적화를 이루었다. 이 내용은 Ⅱ절에서는 EHSX 코더의 알고리즘에 대한 개요를 설명하였으며, Ⅲ절에서의 DSP의 구조와 특징, Ⅳ절에서는 EHSX 음성 부호화기의 실시간 구현, Ⅴ절에서는 구현 결과에 대하여 설명하고, Ⅵ절에서는 결론을 맺었다.

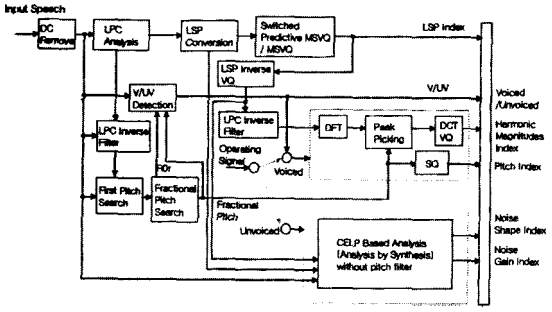

그림 1. EHSX 인코더 블록도
Fig 1. Block diagram of EHSX encoder.

Ⅱ. 4kbps EHSX 음성 코덱

2.1. 4kbps EHSX 음성 부호화기

인코딩 과정은 LPC 분석과정과 LPC 잔여신호를 각 모드에 따라 분석하는 과정으로 분류될 수 있다. 아래 그림 1은 음성 부호화기의 블록도이다. 여기서 LPC 계수는 LSP로 변환된 다음, 전통적 2단계 벡터양자화와 Inter-frame 벡터양자화 중 양자화 에러가 더 적은 것으로 양자화 된다. 그런 다음 양자화 된 LSP 계수를 보간하여 입력 음성신호를 역 LPC 필터링 함으로써 잔여신호를 추출하게 된다. 이때의 추출된 잔여신호는 유/무성음과 전이구간을 판별하는 분류기에 의해 각 대응 코더에 의해 부호화된다. 여기서, 유/무성음 모드 값과 이에 따른 각 코더의 파라미터 값 그리고 LSP 인덱스가 디코더에 전송되어진다. 인코딩 과정에서의 피치는 1차 피치검색 과정과 분수 (Fractional) 피치검색 과정을 거쳐 최종 피치 주기를 구하게 된다. 예측된 피치 주기 값은 유/무성음 모드를 결정하기 위해 사용된다.

2.2. 4kbps EHSX 음성 복호화기

디코더 과정에서는 스펙트럴 외형정보를 나타내는 파라미터인 LSP (Line Spectral Pair)파라미터, 유/무성음 모드와 여기신호 파라미터를 전송 받아 합성과정을 수행한다. 여기서 LSP는 역 양자화를 거쳐 보간 되며, 모드에 따라 여기신호 파라미터는 각각 다른 합성과정을 거친다. 모드가 유성음일 경우 하모닉 및 노이즈 합성과정을 거치며 무성음일 경우 Stochastic 합성과정만을 수행한다. 여기서 합성된 여기신호는 최종적으로 LPC 합성필터를 통과한 후 후단 여파기를 통과함으로써 최종 복원된 음성을 얻게 된다. 그림 2는 음성 복호화기의 블록도이다.

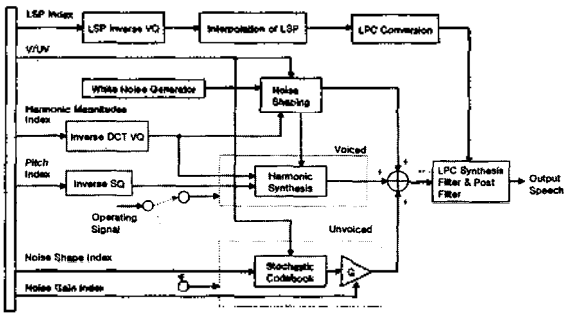

그림 2. EHSX 디코더 블록도
Fig 2. Block diagram of EHSX decoder.

2.3. IFFT를 사용한 저 복잡도 정현파 합성법

4kbps EHSX의 복호화기에서의 정현파 합성 모델은 식 (1)과 같다[5,6].

$$\hat{s}(n) = \sum_{l=1}^{L} A_l(n) e^{j\theta_l(n)} \tag{1}$$

k 번째 프레임의 정현파 합성 모델은 식 (2)와 같다.

$$\hat{s}^k(n) = \sum_{l=1}^{L^k} A_l^k(n) e^{j\theta_l^k(n)} \tag{2}$$

프레임 사이의 연속성을 보장하기 위해 식 (3)과 같이 하모닉 크기 파라미터들은 이전 프레임 파라미터들과 선형 보간 된다.

$$A_l^k(n) = (1-\lambda(n))A_l^k + \lambda(n)A_l^{k+1} \tag{3}$$

여기서, $\lambda(n) = n/N$인 선형증가 함수이고, N은 한 프레임의 전체 샘플수, n은 각각의 샘플수이다. l은 하모닉 순열 번호, k는 현재 프레임 번호를 뜻한다.

$\theta_l^k(n)$은 위상 항으로써 식 (4)와 같이 나타낼 수 있으며, $\gamma_l^k, \alpha_l^k, \phi_l^k$의 2차식으로 정의된다.

$$\theta_l^k(n) = \psi_l^k(n) + \phi_l^k = \gamma_l^k n + \alpha_l^k n^2 + \phi_l^k \tag{4}$$

여기서, $\gamma_l^k = \gamma^k \cdot l$이고, γ^k는 k번째 프레임의 기본주파수이며, 피치 검색 과정으로부터 측정되는 값이다. α_l^k, ϕ_l^k 파라메터 값은 프레임 사이의 주파수 및 위상에 대한 경계조건을 만족하도록 해를 구할 수 있으며, 그 값은 다음과 같다.

$$\alpha_l^k = \frac{(\gamma^{k+1} - \gamma^k)l}{2N}$$

$$\phi_l^{k+1} = \frac{(\gamma^{k+1} + \gamma^k)lN}{2} + \phi_l^k - 2\pi M \tag{5}$$

여기서, M은 정수인 임의의 상수 값이다. 식 (2)는 IDFT 식을 포함한 형태로 유도하기 위해 다음과 같이 표현될 수 있다.

$$\hat{s}^k(n) = \sum_{l=1}^{L^k(n)} A_l^k(n) e^{j\phi_l^k} e^{j\frac{2\pi \cdot l}{B} \cdot \frac{\psi_l^k(n) \cdot B}{2\pi \cdot l}} \tag{6}$$

여기서, B는 IFFT의 포인트 수로 2의 승수인 상수로 정해진다. 그 다음과정으로 오버 샘플 함수인 $f(n) = \frac{\psi_l^k(n) \cdot B}{2\pi \cdot l}$을 정의하면, $f(n)$은 l에 독립이므로 식 (6)은 최종적으로 IFFT를 사용한 형태의 식으로 바뀌어 진다.

$$\begin{aligned}
\hat{s}^k(n) &= \sum_{l=1}^{L^k(n)} A_l^k(n) e^{j\phi_l^k} e^{j\frac{2\pi \cdot f(n) \cdot l}{B}} \\
&= (1-\lambda(n)) \cdot \sum_{l=1}^{B} A_l^k e^{j\phi_l^k} e^{j\frac{2\pi \cdot f(n) \cdot l}{B}} + \lambda(n) \cdot \sum_{l=1}^{B} A_l^{k+1} e^{j\phi_l^k} e^{j\frac{2\pi \cdot f(n) \cdot l}{B}} \\
&= (1-\lambda(n)) \cdot \mathrm{IDFT}\left(A_l^k e^{j\phi_l^k}, B, f(n)\right) + \lambda(n) \cdot \mathrm{IDFT}\left(A_l^{k+1} e^{j\phi_l^k}, B, f(n)\right) \\
&= (1-\lambda(n)) \cdot \mathrm{IFFT}\left(A_l^k e^{j\phi_l^k}, B, f(n)\right) + \lambda(n) \cdot \mathrm{IFFT}\left(A_l^{k+1} e^{j\phi_l^k}, B, f(n)\right)
\end{aligned}$$

where,

$$\mathrm{IDFT}(X, N, n) = \sum_{l=1}^{N} X \cdot e^{j\frac{2\pi \cdot n \cdot l}{N}}$$

$$N = 2^C,\ C \text{ is integer}$$

여기서, $B > L^k$이고 $B > l > L^k$ 의 범위의 $\overline{A}_l^k$와 $\overline{A}_l^{k+1}$은 0 값을 가진다.

본 연구에서는 저 복잡도 정현파 합성을 위해 128 포인트 실수 IFFT의 구현을 2 radix 64 포인트 복소수 FFT를 사용하여 구현하였다.

2.4. 피치 검색

EHSX 음성 부호화기의 피치 분석 과정은 개루프(open-loop) 검색과 폐루프 (closed-loop) 검색과정으로 나뉘어져 있다. 첫 번째로 기존 시간 축 자기 상관(Autocorrelation) 근사 값을 이용한 1차 피치 검색을 하며, 다시 교차 상관 (Cross-Correlation) 값을 통해 분수 (Fractional) 피치를 얻는다. 이러한 구조는 FFT를 통해 적은 계산 량으로 자기 상관 근사 값을 계산하는 Open-loop 피치 검색을 이용하여 대강의 피치를 얻은 다음, 피치 값의 주변에 대해 교차 상관 값이 최대가 되는 피치 값을 구함으로써 전체적인 정확한 피치를 얻을 수 있도록 한다.

2.5. 코드북 검색

실제 적용된 알고리즘은 순환 코드북을 사용한다. 순환 코드북의 크기는 256개의 배열 요소를 가진다. 코드

북에 대한 참조는 다음과 같다. 이러한 순환 코드북을 사용함으로써 메모리 크기를 줄일 수 있다. 아래의 식 (7)은 순환 코드북을 생성하는 식을 나타낸다.

$$c[n][m] = c_{circular}[(m+n)\%256] \tag{7}$$

먼저, 코드북 인덱스를 찾는 알고리즘은 각 부 프레임에 대해 LSP와 역 양자화된 LSP값으로부터 선형 보간된 LPC 값들을 구한다. 그 다음 입력 음성신호로부터 선형 보간된 LPC 값들을 사용하여 ZSR(Zero State Response)신호를 구한다. 이 ZSR신호가 입력 레퍼런스 신호인 $r(n)$이며 $syn(n)$은 코드북에서 나온 신호를 지각 가중치 LPC 합성 필터를 이용하여 ZSR신호를 구한 것이다. 이때 코드북 검색은 식 (8)을 최대화하는 G (이득) 인덱스를 찾음으로써 수행한다[5,6].

$$G = \frac{\sum_{n=0}^{N-1} r(n) \times syn(n)}{\sum_{n=0}^{N-1} (syn(n)^2)} \tag{8}$$

2.6. 양자화 및 비트 할당

EHSX 음성 부호화기는 두 개의 비트율 (2.4kbits/s, 4kbits/s)로 동작하는 저 전송률 부호화 알고리즘으로 이루어져 있다[5][6]. 각각의 양자화 과정은 크게 LSP 및 유/무성음 양자화 과정과 양자화 과정으로 구성된다. 우선 LSP 양자화 과정은 3단계 분할 벡터 양자화 (3-Stage Split Vector Quantization)를 적용하여, 24 비트를 할당하였고, 유성음/무성음을 표현하기 위한 모드 값에서는 2비트가 할당된다. 따라서, 4kbps에 할당된 총 80비트 중 54비트가 여기 신호 표현을 위해 할당되며, 2가지 코딩 파라미터에 따라 비트 할당이 바뀌게 된다. 비트 할당은 피치 값에 7 비트, 1단계에 Gain 5비트 Shape 10 비트를 2단계에는 1 단계의 에러 값을 Full search 코드북 4개를 사용하여 모두 32 비트를 할당하였다. 벡터 여기 코딩은 10ms 단위의 2개의 부 프레임으로 구성되며, 각 프레임은 같은 2 단계 벡터 양자화를 적용한다. 1 단계는 Gain 4 비트 Shape 7비트를 2단계는 다시 5ms 단위로 하여 Gain 3비트, shape 5비트를 적용하였다. 지금 까지 설명한 비트 할당을 표 1의 유성음과 무성음 구간 그리고 전이구간의 전송률 및 모드에 따른 비트 할당에 대해 나타내었다[5,6].

표 1. 전송률 및 모드에 따른 비트 할당
Table 1. Bit allocation to bitrate and mode.

<table>
<tr><td>전송률</td><td colspan="4">4 kbps</td><td colspan="2">4 kbps (Silence Coder)</td></tr>
<tr><td rowspan="4">10차 LSP 양자화값</td><td colspan="4"></td><td>Silence</td><td>전이구간 (Silence count=1)</td></tr>
<tr><td colspan="4">1차 상태 : 6 비트</td><td rowspan="3">·</td><td>1차 상태 : 6비트</td></tr>
<tr><td colspan="4">2차 상태 : 10 비트</td><td>2차 상태 : 10비트</td></tr>
<tr><td colspan="4">3차 상태 : 8 비트</td><td>3차 상태 : 8비트</td></tr>
<tr><td>V/UV/S</td><td colspan="4">2 비트</td><td>2 비트</td><td>2 비트</td></tr>
<tr><td rowspan="13">Exitation 관련 인덱스값</td><td colspan="2">무성음</td><td colspan="2" rowspan="2">유성음</td><td rowspan="13">6 비트 (모두0값)</td><td colspan="2"></td></tr>
<tr><td>LSP</td><td>6 비트</td><td>LSP</td><td>6 비트</td></tr>
<tr><td rowspan="2">5ms</td><td>형태0: 8비트</td><td rowspan="2">피치: 7비트</td><td rowspan="2">5ms</td><td>형태0: 8비트</td></tr>
<tr><td>이득0: 4비트</td><td>이득0: 4비트</td></tr>
<tr><td rowspan="2">5ms</td><td>형태1: 8비트</td><td>DCT0: 5비트</td><td rowspan="2">5ms</td><td>형태1: 8비트</td></tr>
<tr><td>이득1: 4비트</td><td>DCT1: 5비트</td><td>이득1: 4비트</td></tr>
<tr><td rowspan="2">5ms</td><td>형태2: 8비트</td><td>이득: 5비트</td><td rowspan="2">5ms</td><td>형태2: 8비트</td></tr>
<tr><td>이득2: 4비트</td><td>잔차: 8비트</td><td>이득2: 4비트</td></tr>
<tr><td rowspan="2">5ms</td><td>형태3: 8비트</td><td rowspan="2">잔차: 26비트 (7+7+6+6)</td><td rowspan="2">5ms</td><td>형태3: 8비트</td></tr>
<tr><td>이득3: 4비트</td><td>이득3: 4비트</td></tr>
<tr><td>총비트수</td><td colspan="4">80 비트</td><td>8 비트</td><td>80 비트</td></tr>
</table>

III. TMS320C6701® DSP 구조 및 특징

TMS320C6701는 고성능의 부동소수점 DSP로서 6-ns 의 명령어 수행시간을 가지고 167㎒로 동작하며 8개의 32비트 명령어를 한 사이클에 수행한다. 또한 VLIW (Very Long Instrcution Word)의 구조로 되어 있고, 최대 1GFLOPS (one billion floating-point operations per second)의 성능을 나타낸다[10].

주요특징은 다음과 같다.

- 4개의 산술 연산기
  - 부동소수점과 고정소수점
- 2개의 곱셈기
  - 부동소수점과 고정소수점
- 32개의 32비트 범용 레지스터
- 내부 프로그램 메모리(16K Words)
- 내부 데이터 메모리(16K Words)

IV. 실시간 구현

4.1. FastRTS67x 라이브러리를 이용한 최적화

많은 어플리케이션이나 DSP 알고리즘에서 수학 연산은 많이 필요로 하게 된다. 하지만 실제 수학 연산이 돌

어간 C 소스를 핸드 어셈블리 코딩하여 최적화하는 것은 많은 시간과 노력이 필요한데 이런 어려움을 쉽게 해결해 주는 라이브러리가 있다. 바로 표 2에서 적용된 연산량을 나타낸 FastRTS67x library다.

표 2의 분석에서 알 수 있듯이 상당한 연산량 감소를 볼 수 있어서 핸드 어셈블리 코딩에 익숙지 않는 사용자에게 쉽게 개발을 할 수 있도록 도움을 준다. 각 수학 연산의 연산량 감소는 수학 함수당 3~5배 정도이다.

4.2. IFFT를 사용한 알고리즘 최적화

TMS320C6701 의 블록 다이어그램은 그림 3과 같다[10]. 표 3은 128 포인트 IFFT를 사용한 저 복잡도 정현파 합성법을 구현하여 IDFT와의 연산량을 비교한 표이다.

IDFT알고리즘을 사용하는 대신 IFFT를 사용함으로써 많은 연산량 감소를 달성하였다. 이 IFFT를 사용한 정현파 합성은 내부연산에서 3개의 스테이지를 나누어 1을 곱하는 부분에 대해서는 내부 루프 연산을 전개하여 곱셈을 곱하는 횟수를 최소화하여 연산량을 최소화하였다.

4.3. 메모리의 효율적인 사용으로의 최적화

1) 캐쉬 메모리의 사용

메모리의 효율적인 사용을 위해 메모리의 재구성이 필요했다. 즉, 코드의 양은 큰데 비해 실제 잡혀있는 내부 메모리는 한정되어 있다. 그래서 코드의 많은 부분이 외부 메모리에 할당 될 수밖에 없다. 여기서 다른 DSP와는 달리 TMS320C6701 DSP의 장점이 드러나는데 전 코드를 외부 메모리에 할당하고 캐쉬 메모리를 사용하는 것이다. 표 4는 메모리의 효율적인 사용인 캐쉬 메모리의 사용 유무에 따른 연산량을 나타내었으며 실시간 구현이 된 상태에서의 캐쉬 메모리 사용 여부에 따른 연산량을 나타낸 것이다. 위 표에서 볼 수 있듯이 연산량이 대략 1/8 차이가 나는 것을 볼 수 있다.

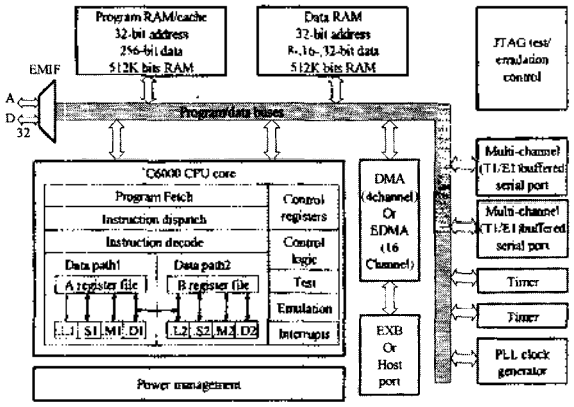

그림 3. TMS320C6701 DSP의 블록도
Fig. 3. Block Daigram of TMS320C6701 DSP

표 2. FastRTS67x library 사이클 비교
Table 2. Cycle comparison to rts6701 and FastRTS67x library.

	rts6701	FastRTS67x 사용
Total (cycle)	62,345,720	43,656,900

표 3. IFFT사용의 사이클 비교
Table 3. Cycle comparison to use IFFT.

	IDFT 사용	IFFT 사용
Total (cycle)	2,148,038	84,871

표 4. 캐쉬 메모리 사용 여부에 따른 연산량
Table 4. Cycle comparison to cache memory.

캐쉬사용 / 함수별	캐쉬 미사용	캐쉬 사용
Encoder (cycle)	16,544,562	2,070,710
Decoder (cycle)	5,673,132	659,476
Total (cycle)	22,217,694	2,730,186

2) 데이터를 내부 메모리에 할당

표 5는 데이터를 내부 메모리에 할당할 수 있을 만큼 내부 메모리의 사이즈가 크지 않다. 그러므로 연산량이 많은 함수에서의 변수들을 내부 메모리로 할당하여 코드와 마찬가지로 데이터도 메모리를 효율적으로 사용하였다.

4.4. 파이프라인을 적용한 핸드 어셈블리 코딩

4kbps에서는 2.4kbps의 하모닉 파라메터를 양자화하는 DCT양자화기에서 2.4kbps 의 양자화기에 향상 계층이 더 추가되고, 무성음 부분의 CELP구조에서도 1/4 프레임인 40샘플 당 양자화 파라미터를 구하는 부분이 추가된다. 따라서 이에 해당하는 만큼의 분석 복잡도도 더욱 커진다. 이러한 연산에는 DCT 양자화기 인덱스를 구하기 위해 상관도 (Correlation)를 구하는 부분 또한, CELP부분의 ZSR (Zero State Response), ZIR (Zero Impulse Response) 부분에는 지각 가중치를 고려한 필터계수를 구하기 때문에 더욱 연산량은 많을 수밖에 없다. 그래서 연산량이 많은 부분은 Q포맷을 적용해 부동소수점에서 고정 소수점으로 변한 후, 파이프라인을 적용한 핸드 어셈블리 코딩을 하여 최적화 하였다.

표 5. 데이터를 내부 메모리에 할당한 예
Table 5. Allocated example to internal data memory.

```
char bufferA[512];
#pragma DATA_SECTION(bufferB, "my_sect")
char bufferB[512];
```

표 6. LPC 합성 필터의 C 소스
Table 6. C source of LPC synthesis filter.

```
for(i=0;i<num;i++)
temp=L_mult(in[i-num],lpcfir[0]) //Q25
for(j=1; j<=NFIR; j++)
temp=L_msu(temp ,in[i-j], negate(lpcfir[j]));
temp = L_shl(temp, 3); //Q(25+3)=Q28
out[i] = round(temp);    //Q(28-16)=Q12
```

1) 부동소수점에서 고정소수점으로의 변환

부동소수점에서 고정소수점으로의 변환 목적은 크게 두 가지 이유가 있다. 첫째, 각 명령어 실행 다음의 딜레이(nop)가 고정소수점이 더 적으며 둘째, 두 가지 기능을 하는 명령어를 쓰기 쉽다. 예를 들어 오버플로우가 발생할 경우 포화 (saturation)을 해주고 쉬프트 (shift) 해주는 명령어를 1cycle에 사용할 수 있다. 즉, 더 많은 연산량을 줄일 수 있기 때문에 변환을 하는 것이다. 아래의 표 6는 지각가중치를 가지는 LPC 합성필터의 ZSR (Zero State Response) 부분을 실제 Q포맷을 이용하여 고정 소수점으로 변환한 C 소스이다.

표 6에서 Q포맷 변환 에러를 최소화하기 위해 데이터를 분석하여 오차율을 최소화하였다.

2) 파이프라인을 적용하기 위한 테이블 작성

다음 표 7는 파이프라인 적용하기 위해서 데이터 load부분과 각각의 연산들을 single-cycle loop처리를 해주기 위한 표이다. 또한, 표 8은 표 6에서 Q포맷을 이용하여 변환한 고정 소스점 소스를 표 7에서 파이프라인을 적용하기 위한 single-cycle loop 테이블에 의해 짜여진 파이프라인이 적용한 어셈블리 코딩한 소스와 C 코딩한 소스의 연산량을 비교해는 표이다. 마지막으로 표 9는 실제 single-cycle loop를 구성해는 소스의 예이며, 표 8에서 보았듯이 연산량의 c소스에 비해 1/5의 연산량이 감소하였다.

V. 검증 및 구현 결과

5.1. 구현 결과

표 10은 EHSX의 4kbps 전송률의 함수별 연산량 최적

표 7. 파이프라인을 적용한 single-cycle 루프 테이블
Table 7. fully pipelined single-cycle loop table.

	loop prolog							
unit/cycle	0	1	2	3	4	5	6	7,8,9...
.D1	ldh	ldh	ldh	ldh	ldh	ldh	ldh	ldh
.M1						mpy	mpy	mpy
.L1								add
.S1		sub	sub	sub	sub	sub	sub	sub
.S2			b	b	b	b	b	b

표 8. c coding과 assembly coding의 사이클 비교
Table 8. Cycle comparison to c coding and assembly coding.

	c coding	fully pipelined hand assembly coding
Total (cycle)	13,275	2,800

표 9. 파이프라인을 적용한 어셈블리 코딩의 LPC 합성필터
Table 9. LPC synthesis filter of fully pipelined hand assembly coding.

```
  mvk .s2 12, b1          ; set up inloop counter
 || ldh  .d1 *a4--, a7    ; load pres[i-j]
 || ldh  .d2 *b4++, b7    ; load lpcfir[j]
  … (중략)
 || ldh  .d1 *a4--, a7    ; load pres[i-j]
 || ldh  .d2 *b4++, b7    ; load lpcfir[j]
 || mpy .m1x a7, b7, a5   ; pi=a4 * b4

 || [b1] b .s1 inloop          ; branch to inloop
 || [b1] sub .s2 b1, 1, b1 ;decrement inloop
counter
inloop:
  ldh .d1 *a4--, a7      ; load pres[i-j]
 || ldh .d2 *b4++, b7     ; load lpcfir[j]
 || mpy .m1x a7, b7, a5 ; pi=a4 * b4
 || add .l1 a5, a8, a8    ; temp=temp+pi
 || [b1] b .s1 inloop     ; branch to inloop
 || [b1] sub .s2 b1, 1, b1 ; decrement counter
 ; inner loop branch occurs here
```

표 10. 주요 함수별 연산량
Table 10. the principle function cycle.

	Function block	최적화전 (cycle)	최적화후 (cycle)
Encode	Pre-processing	487,611	106,720
	Compute LPC	817,632	10,123
	LSP conversion	311,066	7,469
	LSP quantization	2,187,193	8,727
	VAD	3,091	563
	Open-loop pitch	6,676,336	54,441
	Closed-loop pitch	2,495,168	266,775
	Codebook search	36,067,010	726,339
	LPC filter	246,510	2,800
Sub total	Encoder	52,566,261	2,070,710
	Decoder	9,896,702	659,476
Total	Encode+Decoder	62,462,963	2,730,186

화 전·후의 최대 사이클(cycle)를 나타낸 것이다.

5.2. 음성 파형

그림 4와 그림 5는 각각 C 소스의 파형과 TI 실시간 구현 파형을 나타낸 것이다. 유성음의 하모닉 성분은 C 소스 파형이나 TI구현 파형이 구분이 안갈 정도로 같게 나왔으나, 무성음의 경우 연산량 감소를 위해 부동소수점에서 고정소수점으로 변환한 차이 때문에 무성음 부분이 약간 다르게 나왔다. 그렇지만, 전체적인 파형의 그림은 그림 4와 그림 5에서 볼 수 있듯이 비슷하게 나왔으며, 두 음성 파일의 SNR은 90.36[dB]가 나왔다. 대략 1bit도 차이가 나지 않는 값이다.

3) MOS 테스트

본 음성 부호화기의 MOS (Mean Opinion Score) 시험 결과를 표 11에서 나타내었다. MOS 시험은 ITU 표준안인 CS-ACELP와 8kbps QCELP 그리고, 4.8kbps DOD-CELP를 비교 대상으로 하였으며, 음성실험 시험에는 발화자가 다른 남자, 여자 각각 10개 문장을 사용하여, 8명의 청취자를 대상으로 수행하였다. 실험 결과, 제안된 4kbps 음성코더는 8kbps QCELP 알고리즘보다 나은 음질을 보였으며, CS-ACELP에 비해 약 MOS 0.41 정도 낮게 평가되었다. 또한 EHSX 음성 코더의 C 알고리즘의 MOS값과 구현한 코더의 모스값이 약간 차이가 있는데 이 결과는 연산량의 최적화를 위해 부동소수점에서 고정소수점으로 변환하는 과정에서의 약간의 오차이며, 음질이 크게 저하되지는 않는다.

표 11. MOS 시험 결과
Table 11. Result of MOS test.

분 류	남자	여자	전체
원음	4.76	4.70	4.72
8kbps CS-ACELP	4.19	3.70	3.94
8kbps QCELP	3.61	3.38	3.48
4.8kbps DOD-CELP	3.03	3.66	2.79
EHSX 음성 코더(C 알고리즘)	3.73	3.33	3.53
EHSX 음성 코더(DSP 구현)	3.67	3.28	3.45

VI. 결 론

본 논문에서는 TMS320C6701 DSP를 이용하여 4kbps EHSX 음성 부호화기를 구현한 내용을 다루며 이 음성부호화기의 유성음의 경우에 하모닉 여기 코딩, 무성음인 경우에는 CELP 코딩 방식을 사용였다. 실시간 구현을 위해 2가지 접근방식으로 접근하였는데 첫째는, 2 radix 64 포인트 IFFT를 사용한 저 복잡도 정현파 합성법을 구현하여 알고리즘의 최적화에 접근하였으며, 두번째는, 구현에 있어서의 접근 방법을 통한 최적화였다. 그 내용은 DSP에 적합하지 않는 반복루프내 조건문 삭제, 불필요한 계산 삭제, 메모리 복사 방지, 캐쉬 메모리 할당, 테이블을 이용하는 방법 등을 이용하는 방법, TI (Texas Instruments)사의 컴파일러가 제공해주는 C Compiler Intrinsic 사용, -o3 옵션에 의한 최적화, IIR, FIR필터에 사용되는 곱하기 연산 등 복잡도의 문제가 되는 부분을 파이프 라인을 적용한 핸드 어셈블리 코딩을 하였다. 사용된 프로그램 메모리는 201KByte, 데이터 메모리 98KByte이며 실시간 동작 시간이 300만 clock안에 들어와야 하는데 대략 280만clock이므로 DSP 성능의 93%를 사용하였고, 음성 신호의 입·출력이 왜곡되지 않고 정확히 실시간으로 동작하였다.

감사의 글

본 연구는 2003년도 한국 과학재단의 특정 기초연구

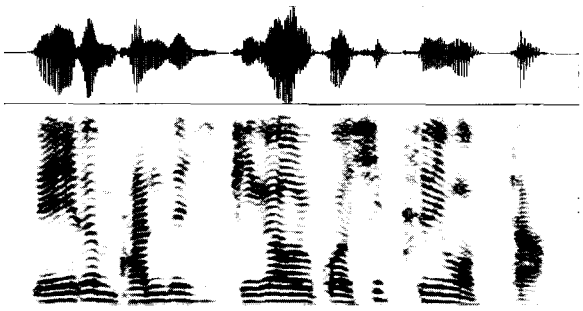

그림 4. C 소스의 파형
Fig 4. Waveform of C source.

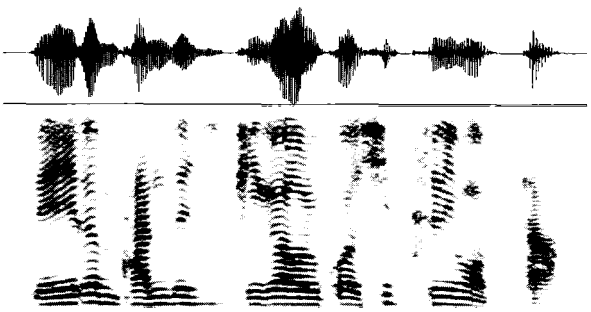

그림 5. DSP 구현의 파형
Fig 5. Waveform of Dsp Implementation.

사업 (과제번호 R01-2003-000-11620-0)의 지원으로 수행되었습니다.

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. D. Childer, R. V. Cox, R. DeMori etc, "The Past, Present, and Future of Speech Processing," IEEE Signal Processing Magazine, pp. 24-48, B. H. Juang Eds., May 1998.
3. A. M. Kondoz, ,"Coding Strategies and Standards", Digital Speech, 5, 117-140, 1994.
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, 1, 204-207, May 1996.
5. 김종학, 이인성 "시간영역 전이를 고려한 2.4 kbps 하모닉 벡터 여기 음성 부호화기 설계", 한국음향학회지, 2000.10 19(7), pp.53-58
6. 김종학, 이인성 "하모닉 코딩과 CELP 방법을 이용한 저 전송률 음성 부호화 방법", 한국음향학회지, 2000.04 19(3), pp.26-34
7. Masayuki, Nishiguchi and J. Matsumotor, "Harmonic and Noise Coding of LPC Residuals with Classified Vector Quantization", in Proc, ICASSP-95, pp. 484-487, 1995
8. Texas Instruments, TMS320C6000 Assembly Language Tools Users Guide, Literature ID# SPRU 186E,1998.
9. Texas Instruments, TMS320C6000 Programmers Guide, Literature ID# SPRU 198B, 1998.
10. Texas Instruments, TMS320C670 Floating-Point DSP (Rev. F), 2004.

저자 약력

• 김 정 호 (Jeong-Ho Kim)

2002년 2월: 배재대학교 전자공학과 학사
2002년 3월~현재: 충북대학교 전파공학과 석사
※주관심분야: 음성, 오디오, 신호처리, DSP

• 이 인 성 (In-Sung Lee)

1983년 2월: 연세대학교 전자공학과 학사
1985년 2월: 연세대학교 전자공학과 학사
1992년 12월: Texas A&M University 전기공학과 박사
1986년 5월~1987년 7월: 한국 통신 연구 개발단 전임 연구원
1993년 3월~1995년 8월: 한국 전자통신 연구원 이동통신 기술 연구단 선임 연구원
1995년현재: 충북대 전기전자컴퓨터 공학부 부교수
※주관심분야: 음성, 오디오, 신호처리, 적응필터, 이동통신

• 박 동 원 (Dong-Won Park)

1983년 : 고려대학교 전기공학과 (공학사)
1985년 : Florida Institute of Technology 공학석사 (전산학 전공)
1993년 : Texas A&M University 공학박사 (전산학 전공)
1994년~현재: 배재대학교 IT공학부 부교수
※주관심분야: 실시간통신, 영상처리