TMS320C62xx를 이용한 2.4kb/s MELP 보코더의 실시간 구현

고은경, 정재호
인하대학교 전자공학과

Real-time Implementation of 2.4kb/s MELP Vocoder on the TMS320C62xx

Eun-Kyoung Go, Jae-Ho Chung
Dept. of Electronic Engineering, Inha University
e-mail : g2001112@inhavision.inha.ac.kr, jhchung@inha.ac.kr

Abstract

본 논문에서는 TI사의 고정 소숫점 연산을 하는 DSP 중 TMS320C62xx를 이용하여 미 국방성의 2.4kbps MELP(Mixed Excitation Linear Predictive) 보코더의 실시간 구현을 목표로 최적화 과정을 수행하였다. 연구에서 사용된 TMS320C62xx의 경우 1,200~2,400MIPS의 성능을 가지므로 PC용 C컴파일러에서도 최적화 되지 않은 MELP의 복잡도가 일정 레벨에서 실시간이 가능하도록 하였다. 먼저 C레벨에서 최적화 작업을 거친 후, 논문에서 사용된 DSP에서 제공하는 컴파일러에서의 최적화 과정을 통해 실시간 동작하도록 하였다. 또한 PC용 C 컴파일러에서 시뮬레이션 한 결과와 DSP상에서 구현한 복호화기의 출력이 정확히 일치함을 검증하였다.

Ⅰ. 서론

현대 사회에서 통신의 발전과 함께 음성 처리의 중요도는 점점 증가하고 있으며 음성 처리 중에서도 음성 코딩에 대한 많은 연구가 이루어지고 있다. 음성 코딩을 하는데 있어서 가장 중요한 것은 얼마나 빠르게 전송 가능하며, 원래의 음성을 얼마나 정확하게 왜곡이나 손실 없이 전달 가능한지가 중요한 초점이 되고 있다. 이런 음성 코딩의 방법 중 파형 부호화 방식으로 알려진 ADPCM, DPCM등은 처리시간이 빠르고 좋은 음질은 갖지만 일정 전송률에서 음질이 심하게 왜곡되는 단점이 있다. 또 혼성부호화 방식인 CELP, MPLPC등은 일정 대역에서 좋은 음질을 보이긴 하나 계산량이 많아 처리시간이 오래 걸린다는 단점을 가지고 있다. 따라서 본 논문에서는 소스 부호화 방식으로 선형 예측 계수(Linear Prediction Coefficient)를 기반으로 하고 새로운 5가지 특징을 추가하여 4.8kbps의 CELP 알고리즘 보다 2.4kbps의 전송률에서 더 좋은 음질을 가지는 미 국방성의 2.4kbps MELP 보코더를 선택하여 실시간 처리를 수행하였다.

본 논문에서는 우선 2.4kbps MELP 보코더의 알고리즘에 대해 전반적인 설명을 한 후 실시간 구현에 사용된 TI사의 TMS320C62xx가 가지는 특징들을 설명하도록 하겠다. 그리고 실시간 처리를 위한 소스 코덱의 최적화 방법을 설명한 후 그 결과를 소개하고 결론을 맺기로 하겠다.

Ⅱ. 2.4kbps MELP 알고리즘

MELP 부호화기의 경우 선형 예측 계수를 기반으로 한 코딩 방식으로 합성단에 여러 코딩 종류 중 다른 방법과는 비교되는 5가지 특징을 추가함으로써 음질향상에 도움을 주게 된다. 추가된 5가지 특징들은 그림 1을 통해 확인 할 수 있다.

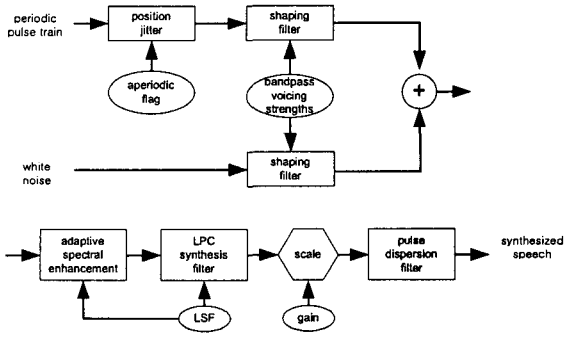

그림 1. MELP 디코더의 흐름도

· Mixed excitation linear prediction
· Periodic & Aperiodic pulse (voiced & unvoiced)
· Adaptive spectral enhancement
· Pulse dispersion (energy)
· Fourier magnitude

MELP 코딩 방식은 프레임 사이즈 180샘플 씩 음성 신호를 입력받아 체비 셰프 고역 통과 필터를 통과함으로써 60Hz이하의 buzzy 잡음을 제거한 음성 신호를 가지고 6차 버터 워스 대역 통과 필터를 사용하여 아래와 같이 5단계의 주파수 대역으로 나누어 피치 주기를 구하게 된다.

· 0 ~ 500 Hz
· 500 - 1000Hz
· 1000 - 2000Hz
· 2000 - 3000Hz
· 3000 - 4000Hz

피치를 구하는 방법은 integer 피치와 fractional 피치를 정규화 된 자기상관 함수를 이용하여 첫 번째 밴드에서의 피치를 구한다. 그리고 나머지 4개의 밴드에 대해서는 첫 번째 밴드의 fractional피치와 envelope를 이용하여 피치를 구한다. 이렇게 각 밴드별로 구해진 피치 값은 잔차 신호에서 구해진 피치 값과 다시 일정 문턱치와 비교하여 좀 더 정확한 피치를 구한다.

MELP 보코더에서는 음성을 유성음, 무성음, jitter로 분류하여 특징 파라미터를 추출해 내며 입력된 프레임이 유성음일 경우 주기적인 펄스와 비 주기적인 펄스를 이용하여 합성해낸다. 또 예측 잔차 신호의 퓨리에 크기 값을 벡터 양자화로 수행하게 된다. 에러 정정 코드로는 (8, 4)해밍코드와 (7, 4)해밍코드를 사용한다. 사용한 2.4kbps 전송률을 갖는 MELP의 경우는 표 1처럼 프레임 당 54bits를 전송하게 된다.[1]

표 1. 비트 할당

Parameters	voiced	unvoiced
LSF's	25	25
Fourier Magnitudes	8	-
Gain(2 per frame)	8	8
Pitch, overall voicing	7	7
Bandpass Voicing	4	-
Aperiodic Flag	1	-
Error Protection	-	13
Sync Bit	1	1
Total Bits/22.5ms Frame	54	54

Ⅲ. TMS320C62xx의 특징

TI사의 TMS320C62xx는 VelociTI™로 Very Long Instruction Word(VLIW)구조를 가지며 고성능 범용 고정 소수점 연산 DSP(Digital Signal Processor)이다. TMS320C62xx는 중앙 처리 장치, 메모리, 주변장치 등으로 구성되며, 각각의 구성 요소들은 프로그램 버스, 데이터 버스, 주변기기 버스 등을 통해 연결된다. 이 DSP의 특징은 아래와 같다.[2]

· 4~6.7ns 명령어 실행 시간
· 150~300Mhz Clock Rate
· 8개의 32Bit Instructions/Cycle
· 1,200~2,400MIPS
· 8개의 독립적인 함수연산(6개의 ALU)
· 32개의 32비트 범용 레지스터
· 64k,128k,512k 바이트의 내부 프로그램/캐쉬 메모리
· 32Bit의 외부 메모리 인터페이스(EMIF)

TMS320C62xx는 고효율의 C 컴파일러 기능을 제공하고 있어 최적화 하기 적합한 특징을 갖는다.

Ⅳ. MELP 코덱의 최적화 과정

최적화 과정을 수행하기 전 가장 먼저 해야할 것은 우선 제공된 원래 소스가 어느 정도의 복잡도를 가지고 있는지 점검해 보아야 한다. MELP는 8Khz 16비트로 샘플링 된 음성 데이터를 입력으로 하여 한 프레임 당 처리시간이 22.5ms이므로 프레임 사이즈는 180이다. 따라서 프레임 당 처리속도가 22.5ms이하여야 실시간 처리가 가능해 진다. 체크된 복잡도는 프레임 당 실행 시간을 체크하여 가장 최대 값을 정리한 것이다. 그리고 원래 소스에 DSP 컴파일 레벨에서 제공되는 파일 최적화 옵션을 추가한 결과를 표 2에서 정리하여 원래 소스와 비교하였다. file최적화 옵션을 추가하는 것은 함수호출 등에 대해 자동 최적화 작업을 수행해 준다. 이 옵션을 추가한 상태의 프로그램을 가지고 앞으로의 최적화 작업을 수행한다.

표 2. 컴파일 결과 비교

함수	*Original source*	*-o3 Option*
Encoder(Cycle)	182,524,177	134,613,005
Decoder(Cycle)	51,843,722	38,720,890
Total(Cycle)	234,367,899	173,333,895
Total(ms)	1172	867

MELP 코덱의 실시간 구현을 위해 C소스 프로그램을 기반으로 PC레벨에서 최적화 과정을 수행한 후 샘플 대 샘플 값이 변화가 없도록 하여 DSP에서 제공하는 컴파일러와 DSP의 특징을 이용하여 작업을 진행하였고 수행한 최적화 과정을 정리하면 다음과 같다.

① MELP 코덱 알고리즘을 충분히 분석하여 전체 흐름과 각 함수별 수행 알고리즘을 이해한다.
② 전체 알고리즘에 전혀 상관없는 함수 호출 등은 제거한다. (data move(), compare(), Logic()등)
③ 원래 소스를 컴파일 하여 각 함수별 계산량을 많이

필요로 하는 것부터 정리해보면 표 3과 같게 나타나므로 계산량이 많은 부분부터 최적화 작업을 수행하면 많은 속도 향상을 볼 수 있다.

④ PC용 C레벨에서 가장 많이 알려진 최적화 방법 중 loop-unrolling방법을 수행하되 DSP 레벨에서 제공하는 profile기능을 이용하여 가장 싸이클이 적게 소모되는 루프 단계를 파악한다.

⑤ 알고리즘을 정확히 분석한 후 알고리즘에 위배되지 않는 범위 내에서 루프 통합, 루프 분리 방법을 이용한다. 다중 루프의 경우에도 내부루프를 순차적으로 전개하면 다중루프가 제거되어 많은 양의 계산량 감소를 가져온다. 마찬가지로 외부 루프가 반복 횟수가 적은 경우도 루프를 제거한다.

⑥ 계산 순서의 변경, 연산자 변경 등 결과 값에 영향을 주지 않으면서 최적화의 효율을 높일 수 있는 방법을 선택한다.

⑦ Intrinsic함수로 제공되는 함수 호출을 하지 않고, 함수 호출 자리에서 바로 연산을 수행 할 수 있도록 함수에서 수행하는 일을 정확히 파악하여 간단하게 처리 할 수 있도록 inline화하여 연산 속도를 줄였다.

⑧ 알고리즘 분석을 하다보면 같은 연산을 여러 번 하는 경우 불필요한 연산은 제거한다. 피치 주기를 찾는 pitch_ana()내에서 fractional 피치를 구하기 전 f_pitch_scale()이라는 함수를 수행함으로써 스케일을 저역 통과 필터를 통과시키는데 앞단의 피치계수가 문턱치 보다 작은 경우에 대해서 fractional 피치를 구하기 위해 다시 한번 f_pitch_scale()을 수행하게 된다. 소스 내에서 이런 불필요한 연산을 찾아 제거한다.

⑨ 전체 MELP 코덱 중 입력 값에 상관없이 항상 같은 계수를 만드는 곳은 미리 코드 북으로 만들어 사용한다.

V. 실험 결과

설명한 최적화 방법들에 의해 작업을 수행하게 되면 PC용 C컴파일 레벨에서는 프로파일 결과 원래 소스의 경우 36.6s이 걸렸지만 현재 0.92s로 수행 속도를 단순히 비교해 보면 97.5% 향상되었다는 것을 볼 수 있다. 마찬가지로 최적화 된 소스를 DSP 레벨에서 컴파일한 결과를 정리하여 원래 최적화 되지 않은 소스와 비교해 보면 표 3과 같이 성능이 향상되었음을 확인 할 수 있다. 최적화 결과를 비교해 보면 알 수 있듯이 수학적인 연산이나 다중 루프 등을 포함하고 있던 함수들에서는 특히 많은 효과를 볼 수 있었다. 지금까지의 최적화 과정을 거치게 되면 원래 소스가 한 프레임 처리 속도가 1172ms였던 것이 약 57.6ms정도의 속도를 가지므로 약 95%이상의 속도 향상을 가져 왔다.

표 3. 최적화 전 후 속도 비교 [단위:cycle]

함수	최적화 전	최적화 후	향상도
msvq_enc()	22,070,163	2,678,589	87.85 %
find_harm()	13,743,709	667,460	95.1 %
zerflt_Q()	8,419,693	49,697	99.4 %
lpc_pred2lsp()	3,232,925	350,939	89.1 %
frac_pch()	2,526,453	225,391	91.0 %
iir_2nd_s()	2,499,600	234,257	90.6 %

C레벨에서의 최적화 과정을 모두 수행한 후 마지막으로 좀더 빠른 성능을 위해 보통 자주 사용되는 함수나 소스 코덱 전체를 내부 메모리를 사용하게 되면 빠른 수행 시간을 얻을 수 있다. 그러나 DSP레벨에서 모든 소스를 내부 메모리 사이즈가 64k바이트의 크기에 모든 코덱을 올리기는 불가능하여 자주 사용되는 값들에 대해 캐쉬 메모리를 사용하여 표 4에서 처럼 캐쉬를 사용하지 않고 최적화만 한 경우와 비교해 보면 50%이상 빠른 결과를 얻을 수 있었다. TMS320C6201에서 컴파일 하는 경우 표 5와 같이 메모리를 사용하였다.

표 4. 캐쉬 사용

함수	최적화 소스	캐쉬사용
encoder(cycle)	9,376,515	4,107,959
decoder(cycle)	2,134,160	316,320
Total(cycle)	11,510,675	4,424,279
Total(ms)	57.6	22.125

표 5. 메모리 사용[단위 : hex]

Memory	사용 Length
Internal program memory	200h
SBSRAM program memory	23D40h
SBSRAM data memory	120h
SDRAM data memory	14A84h
Internal data memory	bafch

따라서 본 논문에서는 좀더 큰 내부 메모리를 갖는 TMS320C6202를 사용하여 내부 메모리에 MELP코덱을 모두 올림으로 TMS320C6201과 비교해 보면 표 6에서 보는 것처럼 상당히 빠른 효과를 보았다. 표 6의 최적화 결과는 DSP 보드를 이용하여 사운드 입출력, 즉 마이크와 스피커를 통하여 음질의 왜곡 없이 실시간 처리가 가능하도록 하였다. 본 논문에서는 8kHz로 샘플링된 음성 데이터를 받아서 MELP 코덱을 거친 후 스피커를 통해 실시간으로 출력하도록 하였다. 이때 실제로 실시간 음성 입출력을 수행하고자 할 때는 사운드 입출력에 관련한 함수를 포함하고 있는 TI사의 라이브러리인 dev6x.lib와 drv6x.lib를 링크 시켜야 한다.

표 6. 성능 평가(TMS320C6202)

함수	최종 결과
encoder(cycle)	1,275,214
decoder(cycle)	266,204
Total(cycle)	1,541,418
Total(ms)	7.72

최적화 과정을 수행하면서 고려해야 할 중요한 사항은 단순히 수행 속도만 단축을 시키면 되는 것이 아니라 음질도 원래 소스의 결과와 비교하여 저하되지 않아야 한다는 것이다. 따라서 최적화 과정을 수행하면서 계속 적으로 원래 소스와 출력된 결과 값을 비교해 보아야 한다. 음질 변화를 비교하기 위해서 DSP상의 결과 값을 파일로 저장하여 원래 ANSI C레벨에서의 결과를 파일 compare툴을 이용하여 비교하는 방법을 사하여 일치함을 확인하였다. 아래 그림 2, 3, 4는 오리지날 입력 신호와 PC용 C컴파일러를 통한 결과, DSP레벨에서의 컴파일 결과를 파일을 통해 비교해 본 것이다.

그림 2. 입력 신호

그림 3. PC레벨 C 컴파일 결과

그림 4. DSP레벨 컴파일 결과

VI.결론 및 향후 과제

본 논문에서는 미 국방성에서 표준화된 2.4kbps의 전송률을 갖는 MELP를 고정 소숫점 DSP인 TMS320C62xx를 이용하여 실시간 구현하였다. 먼저 TMS320C6201에서 작업한 결과 22.125ms까지 성능 향상이 있었으니 좀더 빠른 수행을 위해 내부 메모리가 큰 TMS320C6202를 사용함으로써 마이크와 스피커를 이용한 실시간 처리가 가능하도록 하였다. 최적화 작업은 먼저 PC용 C레벨에서 디버그 한 결과와 DSP상에서의 결과가 샘플 대 샘플로 정확히 일치 되게 만든 후 최적화 작업을 수행하였다. 최적화 작업을 하면서 초점을 둔 것은 원래 MELP의 출력 데이터와 최적화를 한 소스의 출력 데이터가 음질의 왜곡이나 저하가 전혀 없이 정확히 일치시키며 작업을 수행하였다. 현재 C레벨에서의 최적화 작업을 통해 본 연구에서 사용한 TMS320C6202보드 성능의 약 34.3%정도만을 사용하여 실시간 처리가 가능하게 하였다. 본 연구에서 사용된 최적화 방법은 일반적으로 알려진 방법들과 MELP 알고리즘을 분석하여 그 특성에 맞는 방법들을 사용하였다. 앞으로 연산량이 요구되어 많은 클럭이 걸리는 fft(), iir_2nd_d(), iir_2ne_s(), msvq_enc()등의 함수에 대하여 보드 레벨에서 최적화 할 수 있도록 어셈블러 작업을 하여 현재 사용한 보드보다 성능이 낮은 보드에서도 실시간 가능하도록 할 것이다.

참고 문헌

[1] "Specification for the Analog to Digital Conversion of Voice by 2,400Bit/Second Mixed Excitation Linear Prediction" Draft May 28, 1998. www.plh.af.mil/ddvpc

[2] Texas Instruments Inc. *TMS320C6201/6701 Evaluation Module User's Guide.*

[3] A.E.Ertan "Implementation of an enhanced fixed point variable bit-rate MELP vocoder on TMS320C549" Acoustics, Speech, and Signal Processing, 1999 IEEE International Conference on , Volume: 4

[4] Kondoz *Digital speech coding for low bit rate communications systems.*

[5] Texas Instruments Inc.*TMS320C6000 Peripherals Reference Guide.*

[6] Texas Instruments Inc. *TMS320C6000 Optimizing C Compiler User's Guide.*

[7] Texas Instruments DPS R&D Center. "Fixed - Point and TMS320C5x Implementation of the 2.4kbps MELP Federal Standard speech coder "

[8] Texas Instruments Inc. *TMS320C6201/6701 Evaluation Module User's Guide.*

[9] Texas Instruments Inc. *Code Composer Studio User's Guide*