TMS320C6201를 이용한 CS-ACELP (G.729)의 실시간 구현

정회원 백 성 기*, 박 만 호**, 배 건 성*

Real-Time Implementation of the CS-ACELP (G.729) Using TMS320C6201 DSP

Sung Gi Baek*, Man Ho Park**, Kun Sung Bae* *Regular Members*

요 약

본 논문에서는 CS-ACELP 음성부호화기를 TMS320C6201 고정소수점 DSP 칩을 탑재한 EVM 보드 상에서 권고안(G.729)과 함께 제공되는 고정소수점 C 프로그램을 바탕으로 실시간 구현하였다. CS-ACELP 음성부호화기를 실시간 구현하기 위한 최적화 방법에 대해 기술하였으며, 구현된 시스템의 음질 평가를 위해서 음성신호에 대한 C 프로그램의 출력과 구현된 시스템의 출력을 비교하였다. 실험 결과, 최적화 작업을 통해 구해진 전체 프로그램 메모리의 크기는 약 14.04kWords였으며, 한 프레임(10 ms)을 처리하는데 2.5 ms가 소요되었다. 또한, 임의의 음성신호에 대한 C 프로그램의 출력과 구현된 시스템의 출력을 ITU-T에서 제공되는 test vector를 이용하여 bit-exact함을 확인하였으며, 위의 실험결과를 바탕으로 TMS320C6201 EVM 보드에서 마이크와 스피커를 이용하여 CS-ACELP 음성부호화기가 왜곡이나 지연없이 실시간 구현됨을 확인하였다.

ABSTRACT

This paper deals the procedure for real time implementation of CS-ACELP(G.729) speech codec based on the fixed point ANSI C source which is the part of the G.729 standard using TMS320C6201, i.e., a Texas Instrument's fixed-point DSP. We describe the optimization method for real-time implementation and make a comparison between the decoded result of original ANSI C program and that of optimized speech codec for a arbitrary speech signal. Implemented speech codec has the program size of 14.04 kWords. The time required for processing one frame of 10 ms length speech data is about 2.5 ms, and it is short enough for real-time operation. It is verified that the decoded result of the implemented speech codec on the DSP is identical with the PC simulation result using a ANSI C code fot test sequence. Also, actual sound input/output test using microphone and speaker demonstrates its proper real time operation without distortions or delays.

I. 서 론

디지털 이동통신 시스템에서 음성부호화기는 통화품질에 직접적 영향을 주는 부분중의 하나로서 낮은 전송률과 무선환경에 대한 강인성을 가지면서 양호한 음질의 합성음을 가져야 한다. 최근에 ITU-T에서 차세대 통신망(IMT-2000)과 휴대 통신(PCS)에 응용하기 위해 표준안으로 채택한 CS-ACELP (Conjugate Structure - Algebraic Code Excited Linear Prediction, G.729)[1]는 8kbps의 전송률과 채널 에러에 강한 특징을 가지며 32kbps ADPCM(Adaptive Differential PCM)과 대등한 음질을 나타내는 것으로 알려져 있다[2~4].

CS-ACELP 음성부호화 알고리즘은 CELP 구조

* 경북대학교 전자공학과 신호처리 연구실(ksbae@mir.knu.ac.kr),
** 한국전자통신연구소
논문접수번호 : 00169-0512, 접수일자 : 2000년 5월 12일

를 기반으로 하고 있으며, 피치 및 코드북 이득을 양자화하기 위해 결합구조(conjugate structure)[5] 형태를 가지는 2개의 벡터 테이블을 이용하고, 여기 신호로는 대수적 코드북(algebraic codebook)[6]을 사용하는 특징이 있다. 본 연구에서는 CS-ACELP 음성부호화기를 권고안(G.729)과 함께 제공되는 고정 소수점 C 프로그램을 바탕으로 TMS320C6201 고정소수점 DSP(Digital Signal Processor)를 이용하여 최적화 작업을 통해 실시간 구현하는 것을 목적으로 하였다. 그리고 실험결과로써 최적화 작업 전후의 프로그램 크기와 수행속도를 비교하였으며, 구현된 시스템의 음질 평가를 위해 임의의 음성신호에 대한 C 프로그램의 출력과 구현된 시스템의 출력을 비교하였다.

본 논문의 구성은 다음과 같다. 2장에서는 CS-ACELP 음성부호화 알고리즘에 대해 살펴보고, 3장에서는 TMS320C6201의 구성 및 사운드 입출력에 대해서 설명한다. 4장에서는 실시간 구현을 위한 최적화 과정에 대해 설명하고, 5장에서는 최적화 수행결과 및 실시간 구현에 대해 논하며, 마지막으로 6장에서 결론을 맺는다.

Ⅱ. CS-ACELP 음성부호화 알고리즘[1]

CS-ACELP 음성 부호화기는 16bits 선형 PCM 형식의 80샘플(10ms)을 하나의 프레임으로 사용하며, 8kHz의 샘플률과 8kbps의 전송률을 가진다. 부호화기는 그림 1과 같이 크게 전처리 과정, 선형 예측 알고리즘, 적응 코드북(adaptive codebook) 검색, 고정 코드북(fixed codebook) 검색, 이득 결정 과정으로 구성된다. 10ms 분석 프레임마다 전송되는 파라미터는 선형 예측 계수, 적응 코드북 인덱스, 고정 코드북 인덱스, 이득이며, 표 1은 각 파라미터에 대한 비트 할당을 나타낸다.

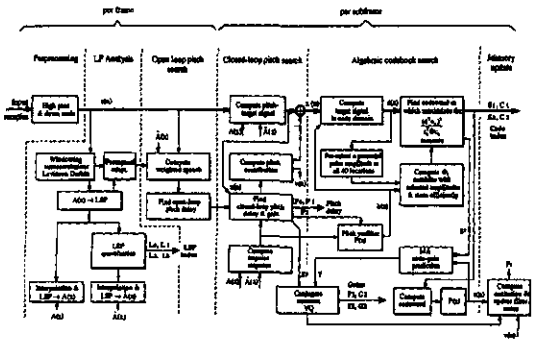


그림 1. CS-ACELP 부호화기의 구조도

표 1. CS-ACELP의 비트 할당

Parameter	Codeword	Subframe1	Subframe2	Total per frame
Line Spectrum Pairs	L0,L1,L2,L3			18
Adaptive-codebook delay	P1, P2	8	5	13
Pitch-delay parity	P0	1		1
Fixed-codebook index	C1, C2	13	13	26
Fixed-codebook sign	S1, S2	4	4	8
Codebook gains(stage 1)	GA1, GA2	3	3	6
Codebook gains(stage 2)	GB1, GB2	4	4	8
Total				80

부호화기의 입력신호는 전처리 과정으로 overflow의 발생가능성을 줄이기 위한 스케일링 과정과 바람직하지 않은 저주파 성분을 줄이기 위한 고역 여파기를 거친 후 10ms의 프레임과 5ms의 서브프레임으로 나뉘어 처리된다. 선형 예측 계수들을 추출하기 위한 30ms의 비대칭 윈도우에는 5ms의 새로운 프레임의 데이터가 포함되어 있어서 전체 알고리즘 지연시간은 15ms가 된다. 매 프레임에 대해서 10차의 선형 예측 계수가 계산되고, 여기서 얻어진 계수는 Line Spectral Pair(LSP)로 변환된 후 2 단계의 분리된 벡터 양자화기(VQ)에 의해 18bits로 양자화된다. 또한 LSP 양자화 과정에서 에러를 줄이기 위해서 4차의 MA(Moving Average) 예측기가 사용되며, 첫 번째 서브프레임에서의 LSP 값은 바로 전 프레임의 LSP 값과 보간되어 사용된다. LSP 계수는 양자화와 보간 과정을 거친 후 합성신호를 만들기 위해 다시 선형 예측 계수로 바뀐다.

다음의 과정을 위해 입력 프레임은 두 개의 서브프레임으로 나뉘며, 각각의 서브프레임에 대해서 피치 정보를 위한 적응 코드북과 여기신호를 얻기 위한 고정 코드북 인덱스가 구해진다[6]. 피치의 결정은 개루프 검색과 폐루프 검색을 통해 이루어진다. 서브프레임마다 결정되는 폐루프 검색은 매 프레임마다 개루프 검색을 통해 얻어진 피치를 기준으로 정수와 1/3의 정밀도로 비정수형 피치를 구하고, 여기서 구해진 적응 코드북 인덱스는 첫 번째 서브프레임에서 8bits, 두 번째 서브프레임에서는 5bits로 양자화된 후 전송된다. 고정 코드북 검색을 위한 목적 신호는 선형 예측 잔차 신호를 가중 합성 여파기에 통과시킴으로써 얻을 수 있으며, 피치 검색에서 사용된 목적 신호에서 적응 코드북에 의한 기여분을 제거함으로써 갱신되고, 이 새로운 목적 신호가 고정 코드북 검색에 사용된다. 고정 코드북은 대

수적 코드북을 사용하며, 여기신호로 사용되는 이 코드북은 표 2와 같이 각 서브프레임마다 4개의 펄스만이 지정된 위치에서 +1 또는 -1의 값을 가진다. 이러한 코드북의 구조에서는 4개의 검색루프를 통해 연속적으로 최적의 펄스 위치를 찾게 되므로 계산량의 감소와 함께 우수한 음질을 얻을 수 있다. 적응 코드북과 고정 코드북의 이득값 양자화는 채널 에러에 강한 특성을 갖기 위해 결합구조를 가지는 2개의 벡터 테이블을 이용한다. 각각의 테이블은 8개와 16개의 요소값을 가지며 구해진 이득값에 의해 미리 선택된 4개와 8개의 요소값들에 대해서만 검색을 하게 되어 검색시간을 단축시킨다.

표 2. 고정 코드북의 구조

Pulse	Sign	Position
i_0	s_0: ±1	m_0:0, 5, 10, 15, 20, 25, 30, 35
i_1	s_1: ±1	m_1:1, 6, 11, 16, 21, 26, 31, 36
i_2	s_2: ±1	m_2:2, 7, 12, 17, 22, 27, 32, 37
i_3	s_3: ±1	m_3:3, 8, 13, 18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39

복호화기에서는 4개의 파라미터(선형 예측 계수, 적응 코드북, 고정 코드북, 이득)가 복호화되며, 복호화된 파라미터를 이용하여 음성을 합성한 후, 음질을 향상시키기 위해 후처리 과정을 거친다. 그림 2는 CS-ACELP 복호화기의 구조를 나타낸 것이다.

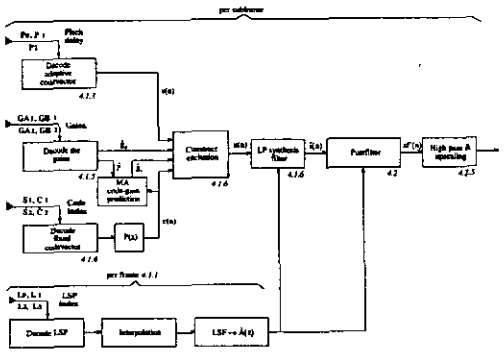


그림 2. CS-ACELP 복호화기의 구조도

복호화 과정은 다음과 같다. 먼저, 수신된 LSP 코드북 인덱스를 이용하여 양자화된 LSP 계수를 구하고, 보간 과정을 거쳐 각각의 서브프레임에 대한 선형 예측 계수를 구한다. 그리고 그 이후의 과정을 서브프레임 단위로 수행하여 합성음을 생성한다. 이렇게 만들어진 합성음은 보다 나은 음질을 위해 후처리 과정을 거치고, 후처리 여파기의 계수는 매 서브프레임마다 갱신된다. 후처리 과정은 long-term post-filter, short-term post-filter, tilt compensation filter가 직렬로 연결된 적응 여파기와, 100Hz의 cutoff 주파수를 가지는 고역 여파기, 그리고 2배의 업스케일링으로 구성된다.

Ⅲ. TMS320C6201 DSP의 구성 및 사운드 입출력

TMS320C6201 DSP는 Texas Instrument사의 TMS320C62xx 고정소수점 DSP 제품군의 하나이다. TMS320C6201 EVM 보드는 그림 3과 같이 CPU, 메모리, 주변장치 등으로 구성되며, 각각의 구성요소들은 program bus, data bus, peripheral bus 등을 통해 연결된다[8]. CPU는 A side와 B side로 분류되는 두 세트의 functional units를 가지고 있고, 각각의 functional units 세트는 4개의 units(.L1 .S1 .M1 .D1 또는 .L2 .S2 .M2 .D2)와 레지스터 파일을 포함하고 있다. 이 functional units를 적당하게 이용함으로써 최대 8개까지의 명령어를 동시에 사용할 수 있어서 병렬처리를 쉽게 할 수 있고, 최대 1600 MIPS(Million Instructions Per Second)의 성능을 발휘할 수 있다[9]. 또한 CPU는 200MHz의 클락률을 가진다.

TMS320C6201 EVM 보드는 내부 메모리와 외부 메모리 두 가지가 있다. 내부 메모리는 2k × 256bits의 프로그램 메모리와 64kbytes의 데이터 메모리로 나누어지며, 외부메모리는 동기식 메모리인 SBSRAM 및 SDRAM과 비동기식 메모리인 SRAM과 EPROM으로 나누어진다[10].

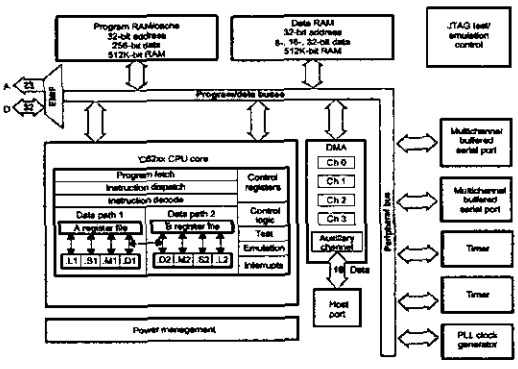


그림 3. EVM 보드의 구조도

McBSP 드라이버, 오디오 코덱 라이브러리, 그리고 보드 지원 라이브러리로 구성되어 있는 DSP 지원 소프트웨어는 TMS320C6201 EVM 보드를 제어하는데 필요한 함수 및 드라이버를 제공한다. 따라서 이러한 라이브러리를 이용함으로써 사운드 입출력 제어 및 호스트(PC)와의 통신을 쉽게 구현할 수 있다[8]. EVM 보드에서 사운드 입출력은 McBSP0를 통하여 이루어지며, 보드에서 사운드의 실제 출력에 관여하는 장치는 스테레오 오디오 코덱(CS4231A)으로써 16bits 스테레오 사운드에 5.5kHz에서 48kHz까지의 샘플률을 제공한다.

본 논문에서는 마이크로부터 음성신호를 입력받아서 CS-ACELP 음성부호화기를 거친 후, 스피커를 통해 실시간으로 출력하도록 구현하였다. 외부 디바이스(마이크, 혹은 스피커)로부터 실시간으로 음성신호를 입/출력하기 위해 비동기함수인 mcbsp_async_receive()와 mcbsp_ async_send()를 사용하였으며, 2개의 버퍼를 이용하여 첫 번째 버퍼가 입력 데이타를 받는 동안 두 번째 버퍼에 저장된 데이타를 처리하도록 하여 입력 프레임사이의 데이터 손실없이 실시간 구현을 가능케 하였다.

Ⅳ. 실시간 처리를 위한 최적화 과정

CS-ACELP를 실시간 처리하기 위해 G.729 표준안과 함께 공개된 고정소수점 C 프로그램을 바탕으로 C 프로그램 상에서 최적화작업을 수행하였고, 일부 함수는 어셈블러로 구현하였다. 수행된 최적화 작업들은 다음과 같다. 먼저 C 프로그램상에서 최적화작업을 수행하였으며, 어셈블러 변환시 더 나은 최적화가 이루어질 수 있는 것은 어셈블러로 변환하였다. 기본적인 연산을 수행하는 함수는 'C6x 컴파일러가 제공하는 intrinsic 함수로 대치하였고, 16bits 연산으로 이루어진 함수는 32bits 연산으로 수정하였다. 반복 횟수가 같은 여러 개의 for 루프가 존재할 경우 적당한 개수 만큼을 하나로 묶었으며, 유사한 구조를 가지는 함수의 경우 알고리즘에 위배되지 않는 범위 내에서 하나의 함수로 묶었다.

'C6x 컴파일러는 'C6x 명령어에 직접 맵핑이 되도록 미리 만들어둔 함수인 intrinsic을 제공하고 있다[9]. 예를 들어, 표 3과 같이 두 개의 16 bits 값을 곱해서 32 bits 값으로 리턴 해주는 L_mult( )라는 함수를 _smpy( )라는 intrinsic 함수로 나타낼 수 있는데, L_mult( )가 많은 명령어를 필요로 하는데 반해 _smpy( )는 하나의 명령어로 L_mult( )와 같은 기능을 수행할 수 있다. 따라서 C 소스 상에서 수행할 수 있는 최적화 작업의 기본이라 할 수 있다. 몇 개의 기본연산함수를 제외하고는 모두 intrinsic 함수를 이용하여 나타냈으며 프로그램 크기와 수행속도를 줄이는데 결정적으로 기여를 한다. 적용되는 알고리즘에 따라 차이가 있겠지만 intrinsic을 적용할 경우, 수행속도가 약 7~10배 정도 감소함으로써 가장 막강한 최적화 방법이라고 볼 수 있다.

표 3. L_mult() 함수의 intrinsic 변환

수정전	Word32 L_mult (Word16 var1, Word16 var2)
	{
	Word32 L_var_out;
	L_var_out = (Word32) var1 *(Word32) var2;
	if (L_var_out != (Word32) 0x40000000L) {
	L_var_out *= 2; }
	else {
	Overflow = 1;
	L_var_out = MAX_32; }
	return (L_var_out);
	}

수정후	#define L_mult(a,b) (_smpy(a,b))

G.729 표준안과 함께 제공되는 C 프로그램은 16bits 연산으로 만들어졌으나 TMS320C 6201은 32bits 연산을 지원하므로 굳이 32bits 변수를 16bits로 쪼갠 후 연산하고 다시 합칠 필요가 없다. 예를 들어, 표 4의 Mpy_32_16 (hi,lo,n)함수를 보면 세 개의 인자를 사용하는데, 처음 2개의 인자는 하나의 32bits 변수의 상위 16bits와 하위 16bits를 나타내고, 세 번째 인자는 16bits변수를 나타낸다. 이 함수를 32bits 연산으로 바꾸면 표 3과 같이 Mpy_32_16_(a, b)로 나타낼 수 있는데, 이렇게 함으로써 코드 사이즈와 사용되는 클락수를 줄일 수 있다. a는 32bits의 변수이고 b는 16bits의 변수이다.

표 4. 16bits 연산과 32bits 연산의 예

16 bits 연산의 Mpy_32_16()	int Mpy_32_16(short hi, short lo, short n)
	{
	int L_32;
	L_32 = L_mult(hi,n);
	L_32 = L_mac(L_32, mult(lo,n),1);
	return (L_32);
	}

32 bits 연산의 Mpy_32_16_()	#define Mpy_32_16_(a,b) (_sadd(_smpyhl(a,b),(_mpyus(a,b)>>16)<<1))

반복 횟수가 같은 여러 개의 for 루프가 존재할 경우 적당한 개수 만큼을 하나로 묶을 수 있으며, 유사한 구조를 가지는 함수의 경우 알고리즘에 위

표 5. pitch_ol() 함수의 최적화 과정

수 정 전

```
for(i=-pit_max;
i<L_frame;i++){
  t0=L_mac(t0,      signal[i],
signal[i]); }

if( t0==MAX_32 ){
      for(i=-pit_max;
i<L_frame;i++){

scal_sig[i]=signal>>3; }
      scal_fac = 3;}

else                    if(
t0<(Word32)1048576L){
      for(i=-pit_max;
i<L_frame;i++){

scal_sig[i]=signal<<3; }
      scal_fac=-3; }

else{
      for(i=-pit_max;
i<L_frame;i++){

scal_sig[i]=signal[i]; } }
```

수 정 후

```
for(i=0;  i<L_frame+pit_max
; i++){
   t0=L_mac(t0,
signal[i-pit_max],
 signal[i-pit_max])
r_sig[i]=signal[i-pit_max]>>3
;
l_sig[i]=_ext(signal[i-pitmax],
19,16);}

if( t0==MAX_32 ){

scal_sig=&r_sig[pit_max]
      scal_fac = 3; }

else                    if(
t0<(Word32)1048576L){

scal_sig=&l_sig[pit_max];
      scal_fac=-3; }

scal_sig = signal;
scal_fac = 0;
```

배되지 않는 범위 내에서 하나의 함수로 묶을 수 있다. 예를 들면, pitch_ol() 함수의 입력신호에 대한 스케일링 과정은 네 개의 for 루프로 이루어져 있으며, 그중 세 개의 for 루프는 조건문 안에 존재하게 되는데 이 루프들을 하나의 루프로 만들 수 있다. 최적화 과정은 표 5와 같다.

V. CS-ACELP 음성부호화기의 최적화 수행 결과 및 실시간 구현

앞에서 언급한 최적화 과정을 통해 CS-ACELP 음성부호화기를 TMS320C6201 EVM 보드를 이용하여 실시간 구현하였으며, 구현된 음성부호화기의 성능을 평가하기 위해 파일 입출력을 통한 실험을 수행하여, 프로그램 메모리와 데이터 메모리의 크기, 그리고 한 프레임을 처리하는데 소요되는 클락 수를 측정하였다. 구현된 시스템의 음질평가를 위해서는 음성신호에 대한 C 프로그램의 출력과 구현된 시스템의 출력을 파형, 스펙트럼을 통해 비교하였다. 또한 ITU-T G.729에서 시스템 구현의 정확성을 증명하기 위해 제공하는 test vector를 이용하여 최적화 결과가 올바름을 확인하였다. 파일 입출력은 TMS320C62xx의 통합개발환경인 code composer studio에서 제공하는 파일 입출력 옵션을 사용하였고, 실험에 사용된 입력 신호로는 ITU-T G.729에서 제공하는 test vector를 이용하였다. 표 6은 ITU-T에서 제공하는 원래의 C 소스 프로그램, intrinsic 함수를 적용한 프로그램, 그리고 최적화 과정을 모두 거친 프로그램의 메모리 크기와 처리속도를 나타낸 것이다.

- 음성부호화기 A : ITU-T에서 제공하는 원래의 C 소스로 구현된 음성부호화기
- 음성부호화기 B : 음성부호화기 A에 intrisic 함수를 적용하여 구현된 음성부호화기
- 음성부호화기 C : 음성부호화기 B에 최적화 과정을 적용하여 최종적으로 구현된 음성부호화기

표 6. CS-ACELP 음성부호화기의 메모리 크기 및 수행속도

	음성부호화기A	음성부호화기B	음성부호화기C
프로그램 메모리 kWords	17.768 (71.072 kbytes)	15.424 (61.696 kbytes)	14.040 (56.320 kbytes)
데이터 ROM kWords	1.542 (6.168 kbytes)	1.538 (6.152 bytes)	1.534 (6.136 kbytes)
데이터 RAM kWords	4.778 (19.114 kbytes)	4.777 (19.106 kbytes)	4.536 (18.144 kbytes)
1 프레임(10msec)에 소요되는 clock 수	34,636,553 (약 173 msec)	2,216,186 (약 11 msec)	499,927 (약 2.5 msec)

표 6에서 볼 수 있듯이 최종적으로 최적화된 프로그램의 크기는 원래의 C 소스 프로그램에 비해 상당히 줄어든 것을 볼 수가 있는데, 이 결과로 인해 모든 프로그램과 데이타가 내부 메모리에 로딩 가능해짐으로써 수행속도 면에서 상당한 이득을 볼 수 있다. 한 프레임(10ms)을 처리하는데 소요되는 클락 수의 경우 실시간 처리의 가능 여부를 평가하는 기준이 되며, 원래의 C 소스 프로그램의 경우 한 프레임을 처리하는데 소요되는 클락 수는 34,636,553 clocks(약 173ms)로 실시간 처리가 불가능하지만, 최적화된 프로그램의 경우 클락 수가 502,218clocks(약 2.5ms)이므로 실시간 처리됨을 확인하였다.

구현된 시스템의 음질평가를 위해서 임의의 음성신호에 대한 C 프로그램의 출력과 구현된 시스템의 출력을 비교하였다. 그림 4는 실험에 사용된 원음성과, ITU-T에서 제공된 원래의 C 프로그램으로 PC 상에서 C 컴파일러를 이용하여 구해진 합성음, 그

리고 DSP 보드상에서 최적화 후 구해진 합성음의 파형을 비교해서 그린 것이다. 그리고 그림 5는 이 파형의 스펙트럼을 비교해서 그린 것이다.

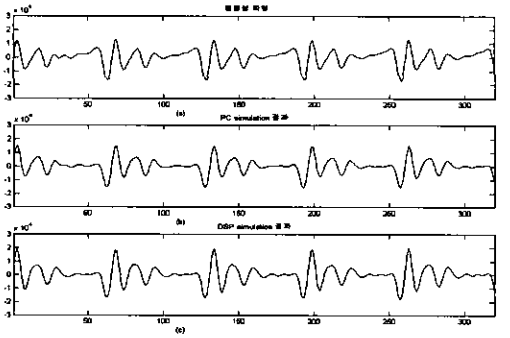


그림 4. 원음성과 합성음의 파형비교

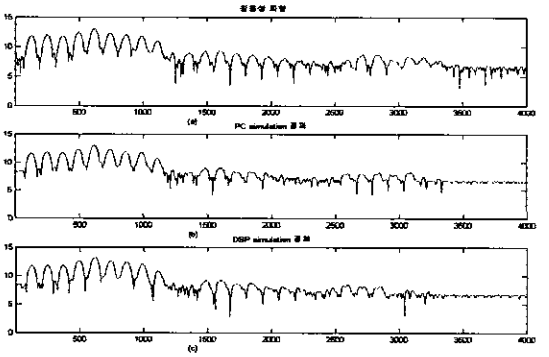


그림 5. 원음성과 합성음의 스펙트럼 비교

CS-ACELP 방식이 파형부호화 방식이 아니므로 그림 4에서 나타나듯이 원음성과 두 합성음의 파형이 서로 일치하지 않음을 볼 수 있다. 그러나, 그림 5의 스펙트럼 비교에서는 원음성과 두 합성음의 스펙트럼 궤적이 일치하며, 기본주파수와 하모닉 성분이 두 합성음에서도 모두 잘 나타나고 있음을 볼 수 있다. 그림 4에서 원래의 C 프로그램에 대한 합성 파형과 DSP 보드상에서의 최적화 후 구해진 합성 파형은 같은 결과를 나타내고 있는데, 이는 C 프로그램의 최적화 및 일부함수의 어셈블러 구현이 제대로 수행되었음을 의미한다. 단, 일부 파형에서 bit-by-bit으로 비교할 때 나타나는 약간의 오차는 ITU-T에서 제공하는 원래의 C 소스 프로그램의 16bits 연산의 일부를 32bits 연산으로 최적화하는 과정에서 발생한 오차이다. 이를 증명하기 위해 ITU-T G.729에서 제공하는 6개의 test vector를 이용하여 C 소스 프로그램의 출력과 16bits 연산을 32bits 연산으로 바꾸는 과정만을 제외하고 모두 최적화한 프로그램의 출력을 bit-by-bit으로 비교하였다. 비교 결과, 구현된 프로그램과 원래의 C 소스 프로그램에 대한 6개 test vector의 결과가 bit-exact 함을 확인하였다. 또한, 32bits 연산에 의한 결과가 16bits 연산에 의한 결과보다 더 정확한 결과라고 할 수 있다.

이상의 실험 결과를 바탕으로 TMS320C 6201 EVM 보드가 제공하는 사운드 입출력 관련 라이브러리 함수를 이용하여 CS-ACELP 음성부호화기를 실시간으로 구현해 본 결과 음질의 왜곡이나 지연없이 실시간 처리가 제대로 수행됨을 확인할 수 있었다.

Ⅵ. 결 론

본 연구에서는 CS-ACELP(G.729) 음성부호화기를 고정소수점 DSP인 TMS320C6201 EVM 보드를 이용하여 실시간으로 구현하였다. 최적화 작업을 통하여 구현된 음성부호화기의 성능을 평가하기 위해서 프로그램 메모리와 데이터 메모리의 크기, 그리고 한 프레임당 수행 시간을 측정하였다. 그 결과, 최적화 작업을 통해 구현된 프로그램 사이즈는 14.04 kWords로 나타났으며, 한 프레임(10ms)을 처리하는데 2.5ms가 소요됨으로써 실시간 처리됨을 확인하였다. 구현된 시스템의 음질평가를 위해서 권고안(G.729)과 함께 제공되는 고정소수점 C 프로그램에 대한 PC에서의 합성음과 DSP 보드에서 최적화 작업을 통해 얻어진 합성음을 비교하였으며 서로 일치함을 확인하였다. 또한, ITU-T의 G.729에서 제공하는 test-vector를 이용하여 최적화가 올바로 수행되었음을 검증하였다. 이상의 결과를 바탕으로 TMS320C6201 EVM 보드에서 마이크와 스피커를 이용하여 CS-ACELP 음성부호화기가 왜곡이나 지연없이 실시간 구현됨을 확인하였다.

참 고 문 헌

[1] ITU-T Recommendation G.729, "Coding of speech at 8kbps using conjugate structure algebraic code excited linear prediction (CS-ACELP)," Geneva, Switzer- land, Mar. 1995.

[2] R. V. Cox, "Three New Speech Coders from the ITU Cover a Range of Applications," IEEE Commun. Mag., Sep. 1997.

[3] Mark E. Perkins, Keith Evans and Dominique Pascal, "Characterizing the Subjective Performance of the ITU-T G.729," IEEE Commun. Mag., Sep. 1997.

[4] Simao Ferraz de Campos Neto and Warren Karapetian, "Performance of ITU-T G.729 8kbps CS-ACELP Speech Codec with Nonvoiced Narrowband Signals," IEEE Commun. Mag., Sep. 1997.

[5] Akitoshi Kataoka, Takehiro Moriya and shinji Hayashi, "An 8 kbps Speech Coder Based on Conjugate Structure CELP," Proc. 1993 IEEE Int. Conf. on Acoustics, Speech and Signal Proc., pp. 592-595, 1993.

[6] R.Salami, C. Laflamme, J-P. Adoul and D. Massaloux, "A toll quality 8kbps speech codec for the personal communication system (PCS)," Proc. IEEE Trans. Veh. Technol., vol. 43, no. 3, pp. 808-816, 1993.

[7] Texas Instruments, TMS320C6201/6701 Evaluation Module User's Guide

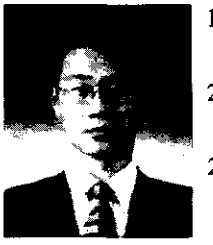
[8] Texas Instruments, TMS320C6201/6701 Evaluation Module Technical Reference

[9] Texas Instruments, TMS320C62xx Programmer's Guide

[10] Texas Instruments, TMS320C6200 Peripherals Reference Guide

[11] Texas Instruments, TMS320C6000 Assembly Language Tools User's Guide

백 성 기(Sung-gi Baek)



1999년 2월 : 경북대학교 전자공학과 졸업
2001년 2월 : 경북대학교 전자공학과 석사
2001년 3월~현재 : 삼성전자 근무

<주관심 분야> 전자공학, 통신공학