DSP를 이용한 가변어휘 음성인식기 구현에 관한 연구

Implementation of Vocabulary-Independent Speech Recognizer Using a DSP

정 익 주*
Ik-Joo Chung

ABSTRACT

In this paper, we implemented a vocabulary-independent speech recognizer using the TMS320VC33 DSP. For this implementation, we had developed very small-sized recognition engine based on diphone sub-word unit, which is especially suited for embedded applications where the system resources are severely limited. The recognition accuracy of the developed recognizer with 1 mixture per state and 4 states per diphone is 94.5% when tested on frequently-used 2000 words set. The design of the hardware was focused on minimal use of parts, which results in reduced material cost. The finally developed hardware only includes a DSP, 512 Kword flash ROM and a voice codec. In porting the recognition engine to the DSP, we introduced several methods of using data and program memory efficiently and developed the versatile software protocol for host interface. Finally, we also made an evaluation board for testing the developed hardware recognition module.

Keywords: Vocabulary-independent speech recognizer, TMS320VC33, Diphone

1. 서 론

최근 음성인식 기술의 실질적인 활용 및 상용화에 큰 관심이 고조되면서 지난 수십 년 간 지속되어온 음성인식 연구는 어느 정도 그 성과가 나타나고 있다. 특히, 반도체 기술의 획기적인 발전으로 고성능의 마이크로프로세서와 고용량의 메모리를 저렴한 가격에 사용할 수 있게 되면서 연구실에서 얻어진 성과물들이 속속 상용화되고 있다.

음성인식 시스템은 응용 방식에 따라서 두 가지 영역으로 나누어진다. 첫 번째는 서버 방식으로, 원격지에서 발성된 음성이 통신 채널을 통하여 서버에 전송되어진 후, 서버에서 인식이 되는 방식이다. 서버 방식의 경우 동시에 여러 채널로부터 들어오는 음성을 처리해야하며, 자원(resource) 사용에 큰 제약이 없으므로 자연어 처리를 포함한 화자독립 가변어휘 인식기술과 같은 대용량 인식기술이 적용된다. 전화망이나 VoIP를 통한 음성인식 응용이 대표적인 예이다. 두 번째 응용 방식은 음성이 입력되어지는 단말기에서 곧 바로 인식이 수행되는 클라이언트 방식이다. 음성인식이 적용

* 강원대학교 전기전자정보통신공학부

된 가전제품, 음성인식 휴대폰등과 같은 제품들이 이 범주에 속한다[1]. 한편 최근에는 서버와 클라이언트가 음성인식 과정을 분담하여 처리하는 분산형 음성인식 기술도 선보이고 있다.

본 연구는 위에서 설명한 분야 중 클라이언트 방식에 적용할 수 있는 기술에 관한 것이다. 이 분야의 기술은 휴대폰이나 PDA와 같은 임베디드 장치에 음성인식 기술을 적용하게 되므로 임베디드(embedded) 음성인식 기술이라고도 하며, 최종 사용자(end-user) 제품에 적용된다는 측면에서 볼 때, 활용 가능성 및 이 기술을 적용한 제품의 시장 규모가 매우 크다고 할 수 있다[2]. 한편, 임베디드 환경에서 음성인식 기술을 적용할 때에는 서버 방식과 달리, 임베디드 시스템의 성격 상, 사용할 수 있는 자원에 많은 제약이 따른다. 높지 않은 성능의 CPU와 적은 량의 메모리만으로 음성인식기를 구현하여야 하므로 음성인식 기술 중 비교적 규모가 작은 화자종속 기술이나 고립단어 방식의 화자독립 기술 정도가 적용되고 있다. 임베디드 장치에서 음성인식을 구현하는 방식은 소프트웨어적인 방식과 하드웨어적인 방식으로 나누어진다. 소프트웨어적인 방식은 임베디드 장치가 이미 포함하고 있는 CPU와 메모리를 활용하여 소프트웨어적으로 음성인식기를 구현하는 것이다. 소프트웨어적으로 구현하므로 비용 상승의 부담이 크지 않은 반면, 음성인식이 적용될 임베디드 시스템 내의 다른 소프트웨어들과 함께 운용이 되어지므로 개발 초기 단계부터 같이 운용되는 소프트웨어들과의 연동성 및 안정성을 고려해야하는 개발상의 부담이 있게 된다. 하드웨어 방식은 음성인식을 전담하는 음성인식 칩이나 저가의 DSP를 포함하는 음성인식 모듈을 이용하는 것이다. 이 방식은 추가의 재료비 상승 요인이 있으나 음성인식을 적용할 시스템과는 비교적 독립적이므로 적용이 용이하다. 뿐만 아니라, 소프트웨어 방식의 경우 어느 정도 연산능력을 가지는 CPU가 있어야하며 간단한 실시간 OS도 필요한 반면, 하드웨어 방식의 경우에는 이러한 제약을 받지 않으므로 어떠한 제품에도 적용이 가능하다는 장점이 있다.

본 논문에서는 이러한 배경을 바탕으로 현재 발표된 부동소수점 DSP 중 가장 저가이며 널리 사용되는 Texas Instruments 사의 TMS320VC33 DSP를 이용하여 가변어휘 음성인식기를 구현하였다. 가변어휘 음성인식기를 구현할 경우 비교적 많은 자원을 요구하므로 하드웨어 방식으로 구현할 시에는 많은 재료비가 요구되나 본 연구에서는 최소의 부품만으로 구현을 하였다. 이를 위하여 인식 알고리즘 및 하드웨어 설계를 최적화하였으며 최종적으로는 DSP, 플래쉬 메모리, 코덱 만으로 이루어진 소형의 하드웨어 음성인식 모듈 및 이를 평가할 수 있는 평가 보드를 제작하였다. 논문의 구성은 2장에서 구현을 위하여 개발된 다이폰 기반의 가변어휘 음성인식 알고리즘, 3장에서는 하드웨어 구성 및 특징, 4장에서 인식알고리즘의 DSP 구현 그리고 5장의 결론으로 이루어져 있다.

2. 다이폰 기반의 가변어휘 음성인식

가변어휘 음성인식은 어떤 종류의 서브워드(sub-word) 유닛을 사용하느냐에 따라서 인식기의 규모가 달라진다. 모노폰(monophone)을 사용할 경우 인식기를 구성하는 서브워드 유닛 모델 DB의 규모가 적음으로 적은 메모리 사용량을 요구하지만 문맥 종속적인 표현이 어려워 인식률이 낮아진다. 반면, 트라이폰(triphone)의 경우 문맥 종속적인 표현 능력이 좋으므로 높은 인식률을 얻을 수 있으나 임베디드 응용에 적용하기에는 인식기의 규모가 너무 크다. 따라서 본 구현에서는 트라이폰

보다는 적은 유닛을 가지면서도 문맥 종속적 유닛인 다이폰(diphone)을 서브워드 유닛으로 사용하였다. 다이폰은 전음소의 후반부 반과 후음소의 전반부 반이 연결되는 형태의 유닛이다. 다이폰의 경계는 조음의 변화가 적어 안정적이므로 유닛 연결 시에 스펙트럼의 왜곡이 적다는 장점을 가지고 있다. 또한 전음소와 후음소 사이의 전이구간에 나타나는 조음효과를 잘 보전하고 있으므로 음성합성에도 적합한 유닛이다[3]. 우리말의 경우 약 1,300여 개의 다이폰이 있는 것으로 알려져 있다[4]. 본 가변어휘 음성인식기에서는 총 1,375 개의 다이폰 유닛을 사용한다. 이 중 350여 개의 다이폰은 순수 우리말에서는 거의 발생 빈도가 없는 다이폰이므로 메모리 사용을 줄이기 위해서는 유사 발음의 다이폰으로 대치할 수 있다. 그러나 최근에는 외래어 및 영어 고유명사를 한글로 표기하여 인식어휘로 사용하는 경우가 늘어나고 있으므로 이들 저빈도 발생의 다이폰도 모두 포함하였다. 그림 1은 전체적인 가변어휘 인식기의 구조를 나타낸다.

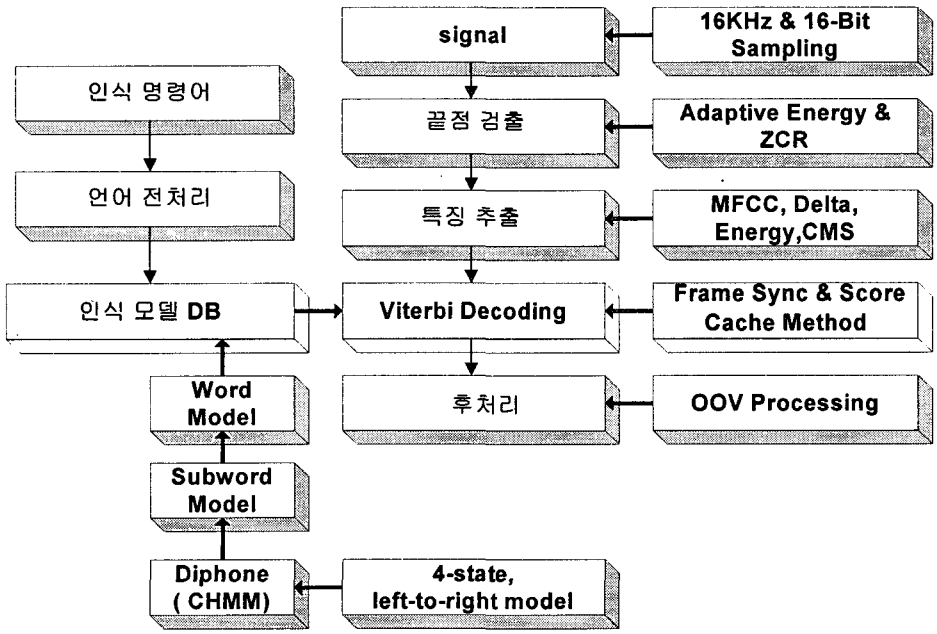

그림 1. 다이폰 서브워드 유닛을 이용한 가변어휘 인식기

일반적인 임베디드용 음성인식기의 경우 사용할 수 있는 자원의 한계 등으로 인하여 8 KHz 또는 11.025 KHz의 표본화 주파수를 사용하고 있으나 본 인식기에서는 음성의 표본화를 16 KHz, 16 비트로 하였다. 표본화 주파수가 높은 것은 인식률 측면에서는 도움이 되나 실시간 처리를 위해서는 좀 더 높은 성능의 CPU가 필요하며 실시간 처리를 위해 사용되는 버퍼 메모리가 증가한다는 단점이 있다. 표본화 주파수를 16 KHz로 선택한 이유는 인식률 향상의 긍정적인 효과를 고려한 측면도 있으나, 임베디드용 화자독립 가변어휘 인식기를 위해 사용할 수 있는 8 KHz나 11.025 KHz의 음성 데이터 베이스가 충분하지 않았기 때문이다. 이용 가능한 8 KHz 표본화 주파수의 음성데이터의 경우 대부분 전화망을 위한 데이터들이므로 사용할 수 없었다. 반면 국내 여러 기관에서 배포된

상당량의 음성 데이터들이 16 KHz의 표본화 주파수로 되어 있으므로 본 인식기의 경우 16 KHz의 표본화 주파수를 사용하였다. 한편 16 KHz의 데이터들을 8 KHz로 다운 샘플링하여 사용하는 것도 고려하여 보았으나 다운 샘플링 과정 자체에서 왜곡이 발생할 수 있으므로 최종적으로 16 KHz의 표본화 주파수로 결정을 하였다.

표본화된 음성신호는 끝점 검출을 위하여 DC 오프셋 및 저주파 노이즈를 제거하기 위하여 150 Hz를 cutoff 주파수를 가지는 고역 필터를 통과시킨 후, 끝점 검출기로 입력된다. 끝점 검출기는 통상의 방식과 같이 프레임 에너지와 프레임 ZCR을 변형한 LCR(Level Crossing Rate)을 이용하였다. 그리고 주변 배경 잡음의 변화에 대응하도록 음성이 아니라고 판명된 프레임의 에너지 및 LCR을 이용하여 이들 파라메터의 문턱값을 적응시켰다. 끝점 검출을 위한 프레임의 길이는 200 샘플로 하였다. 한편, 임베디드 음성인식기의 경우 잡음이 심하거나 마이크의 거리가 30 cm 이상 떨어져 사용해야 하는 경우가 있는데 이 경우 ZCR에 기반한 파라메터는 끝점 검출에 큰 도움이 되지 못한다. 따라서 LCR 파라메터의 사용 여부는 옵션으로 설정 가능하게 하였다.

끝점 검출과 동시에 특징 추출을 실시간으로 행한다. 사용한 특징 파라메터는 12차의 MFCC 파라메터와 에너지 파라메터를 사용하였다. 에너지 파라메터는 프레임 로그에너지 대신 0차 MFCC 파라메터인 C0 파라메터를 사용하였다. 델타 파라메터도 사용하였으므로 총 26차의 파라메터를 사용하였다. 한편, 훈련에 사용된 대부분의 음성데이터들이 다이나믹 마이크로폰을 이용하여 녹음된 것인데 반하여 임베디드 음성인식기의 경우는 대부분 크기가 작은 콘덴서 마이크로폰을 사용한다. 이는 테스트 음성이 훈련 모델과 다른, 소위 불일치(mismatch) 문제를 야기시키며 인식률에 나쁜 영향을 미친다[5]. 모델 불일치와 관련된 여러 기법들이 제안되고 있으나 본 인식기에서는 적은 연산량만으로도 어느 정도 성능 향상에 기여하는 CMS 알고리즘을 적용하였다. CMS 알고리즘이 모델 불일치 문제의 완벽한 해결책은 아니나 마이크의 전달특성의 차이에서 기인하는 특징 벡터 평균의 차이로부터 발생하는 인식률 저하를 적은 연산만으로 해결할 수 있다는 측면에서 볼 때 임베디드용 음성인식기에서 사용할 수 있는 좋은 대안이다. 분석 구간은 25 msec이며 매 프레임 분석 시마다 이전 프레임과 12.5 msec의 오버랩을 하였다.

끝점 검출과 동시에 특징 분석이 끝나면 인식을 위해 추가된 인식 어휘의 문자정보 및 다이폰 모델들을 이용하여 만들어진 각각의 인식 어휘 모델들과 스코어링(scoring)을 위한 비터비 디코딩을 수행한다. 이 부분은 이미 인식 어휘를 위한 모델들이 구성되어 있으므로 고정어휘 방식의 인식과 유사하다. 따라서 word-by-word 방식의 비터비 디코딩을 수행하면 메모리 사용량을 최소화하면서 구현이 가능하나, 인식 속도 향상 및 향후 연결단어나 핵심어 인식기로의 확장성을 위하여 프레임 동기(frame synchronous) 방식의 비터비 디코딩을 수행하였다. 프레임 동기 방식의 비터비 디코딩을 할 경우, 인식 시 포함되는 모든 스테이트(state)에 대하여 계산된 관측 확률을 저장할 수 있는 공간인 캐쉬(Cache)를 할당하고 이미 계산된 관측 확률을 저장하여 둠으로서 한번 계산된 스테이트의 관측 확률을 다시 계산하지 않는 스코어 캐쉬 방식을 적용할 수 있으므로 연산량을 줄일 수 있다[6]. 특히 인식 어휘가 늘어날수록 반복 사용되는 다이폰 유닛이 증가하므로 인식 어휘 수가 많을 경우 인식속도를 증가시키는 데 큰 효과가 있다. 또한 인식속도 향상을 위하여 인식률 저하가 없는 범위 내에서 비터비 탐색 시에 푸르닝(pruning)을 하였다. 비터비 디코딩이 완료되어 얻어진 인식 결과는 Out-of-Vocabulary(OOV) 처리를 위하여 후처리 과정을 거친다. OOV 기술은 음성인

식 기술 중에서도 가장 어려운 기술 중에 하나이다. OOV가 제대로 처리되기 위해서는 상당히 복잡한 처리를 요구한다. 현재 알려져 있는 반음소 방식[7]과 같은 OOV 알고리즘을 자원 사용을 최소로 하고자하는 임베디드 시스템에 적용하기에는 무리가 따른다. 본 연구에서는 그림 2와 같이 비교적 적은 자원을 사용하면서도 어느 정도 성능을 발휘하는 단어 단위 기반의 가비지 방식 OOV 알고리즘을 적용하였다.

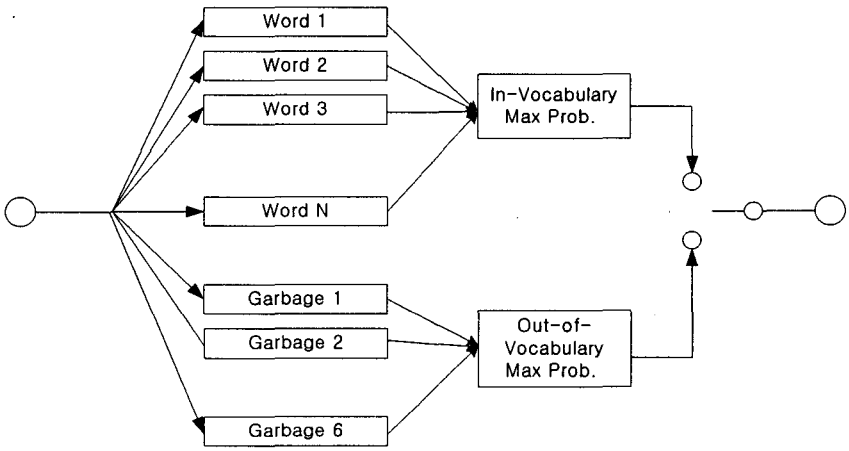

그림 2. Out-of-Vocabulary 처리를 위한 구조

가비지를 만들기 위해 Phone Balanced Word(PBW) 음성 DB에서 단음절, 2 음절, 3 음절 4 음절, 5 음절 그리고 5 음절 이상으로 단어들을 음절 길이별로 분류하여 각각에 대하여 훈련을 통해 총 6개의 가비지를 만들어 OOV 처리에 사용하였다. 한편, 거부율을 조정하기 위하여 최종적으로 식(1) 같은 과정을 수행함으로서 최종 인식 결과를 얻게 된다.

$$P(W_{in} \mid O) - P(W_{oov} \mid O) > thr \tag{1}$$

여기서 *thr*은 조정 가능한 거부율이며 $P(W_{oov} \mid O)$은 In-Vocabulay에서 얻은 최대 비터비 확률, $P(W_{oov} \mid O)$은 가비지들에서 얻은 최대 비터비 확률이다.

가변어휘 인식기는 비터비 디코더를 중심으로 하는 인식기 외에도 인식 어휘의 문자 정보를 기반으로 단어 기반의 인식 모델을 생성하는 부분이 필요하다. 인식 어휘가 추가되면 우선 전처리 과정을 거친다. 전처리 과정에서는 음운 현상을 고려하여 소리 나는 대로 변환하게 되는데 서버 환경에서 사용할 수 있는 본격적인 전처리기를 포함하는 것은 제한된 메모리 상황에서는 어려움이 따른다. 본 구현에서는 인식에 가장 영향을 미치는 연음법칙만을 적용하였다. 이렇게 전처리 과정을 거쳐서 나온 다이폰 열을 이용하여 다이폰 모델 DB로부터 다이폰 모델을 연결함으로서 가변어휘 단어 모델을 생성한다. 가변어휘 인식기는 고정어휘 인식기와는 달리 서브워드 유닛으로 구성된 모델 DB를 구현하는데 많은 시간과 노력이 필요하다. 논 연구에서는 다음과 같은 과정으로 연속

(continuous) HMM에 기반하여 다이폰 모델 DB를 만들었으며 이 과정에서 HTK를 이용하여 훈련을 하였다[6]. 우선 정확한 레이블링이 포함된 부트스트랩(boot strap)용 음성데이터를 위하여 남녀 각각 30 명의 음성을 녹음 하였다. 이 녹음은 모든 다이폰을 포함하는 발성 조합을 발음한 것이다. 녹음된 음성은 수작업으로 레이블을 하였으며 이렇게 레이블링된 1,375 개의 다이폰 유닛은 whole-word 방식으로 개별적으로 훈련되었다. 다이폰 모델은 그림 3과 같이 Entry와 Exit 노드를 제외하고 4 개의 스테이트를 가지는 Left-Right 모델을 이용하였다.

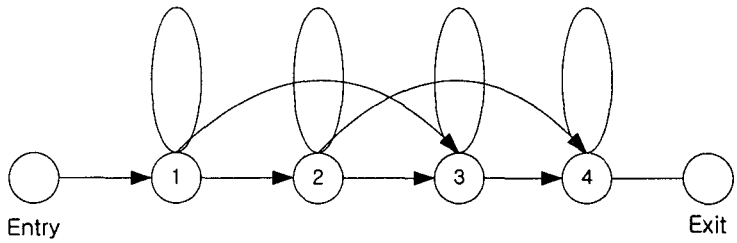

그림 3. 다이폰 모델을 위한 Left-Right 모델

이렇게 만들어진 1,375 개의 다이폰 모델은 다량의 음성데이터를 이용하여 임베디드 훈련을 하였다. 이 과정에서 사용한 음성데이터는 여러 기관에서 배포한 표준 음성데이터들로서 약 1,500 명의 화자를 포함하고 있다. 훈련된 다이폰 모델 DB를 포함하여 최종적인 가변어휘 인식기는 우선 DSP에 포팅하기 이전 단계로 오프라인 상에서 인식률 테스트를 하였다. 테스트로 사용한 음성은 국어공학연구소에서 배포한 고빈도 2,000 어절 음성데이터로서 남녀 각각 1인이 고빈도의 2,000 단어를 발성한 것이다. 표 1은 개발된 가변어휘 인식기의 인식률이다.

표 1. 가변어휘 인식기의 인식률

다이폰 모델 DB	인식률
1 Mixture, Skip Transition을 허용하지 않음	93.6%
1 Mixture, Skip Transition을 허용	94.5%
2 Mixture, Skip Transition을 허용	95.9%

스테이트 당 믹스쳐(mixture)를 2 개를 쓸 경우 인식률이 향상되나 믹스쳐 1 개인 경우에 비하여 다이폰 모델을 위한 메모리와 인식 시 수행되는 연산량이 거의 2 배로 늘어나므로 스테이트 당 믹스쳐를 1로 결정하였다. 또한 스테이트 간 전이의 경우 그림 3처럼 스킵(Skip)이 있는 경우 약간 더 좋은 성능을 나타내었는데 이 경우에도 모든 다이폰에 스킵을 허용하는 것보다 자음+자음(CC)에 해당하는 다이폰을 제외한 나머지 다이폰들에만 스킵을 허용하는 경우에 더 좋은 인식률을 보였다.

3. TMS320VC33을 이용한 하드웨어 설계

다음은 TMS320VC33 DSP를 이용하여 구현된 하드웨어 모듈의 블록도이다.

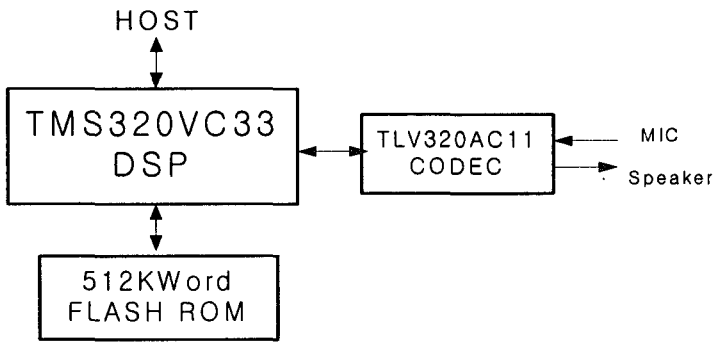

그림 4. TMS320VC33 DSP를 이용한 하드웨어 모듈의 블록도

TMS320VC33은 기존의 TMS320C31과 동일한 구조를 가지지만 DSP 코아 전압으로 1.8 V를 사용하여 전력소비를 줄였으며 내부에 32 K워드의 메모리를 추가하여 총 34 K워드의 내부 메모리를 가지고 있다[8]. 비록 충분한 메모리는 아니지만 새로 추가된 내부메모리 덕분에 외부에 별도의 SRAM을 사용하지 않았다. 외부메모리로는 프로그램 코드 및 다이폰 모델 DB를 위해 512 K워드의 플래쉬 메모리를 사용하였다. 플래쉬 메모리는 프로그램 코드와 다이폰 모델 DB를 저장하는 ROM의 역할을 하지만 사용자가 정의한 인식어휘에 대한 문자정보도 저장한다. 따라서 전원이 새로 인가될 경우 이전에 정의하였던 인식어휘가 인식대상이 된다. TMS320VC33 DSP의 시리얼포트에는 A/D 변환을 위하여 Texas Instruments사의 TLV320AC11 코덱을 연결하였다. TLV320AC11 코덱은 시그마 델터 방식의 코덱으로 최대 22 KHz의 표본화 주파수를 지원한다. 또한 마이크로폰 입력 신호 증폭을 위한 증폭기를 내장하고 있어서 별도의 아날로그 증폭 회로 없이 곧바로 마이크로폰를 연결할 수 있다. TLV320AC11 코덱을 16 KHz의 표본화 주파수로 설정하기 위해서는 16.384 MHz의 코덱 마스터 클럭이 필요한데 별도의 오실레이터를 추가하지 않고 DSP의 타이머 단자로부터 DSP의 머신 클럭을 분주하여 사용하였다. DSP 타이머에서 16.384 MHz의 클럭을 얻기 위해 DSP의 머신 클럭으로 40.96 MHz를 사용하였다. 한편, 코덱의 D/A는 인식된 결과를 음절합성으로 출력하는데 사용하였다. 구현된 가변어휘 인식기의 활용성을 높이기 위하여 음절합성 기능을 추가하였는데 이는 명료도 위주의 편집합성기로, 저장되어 있는 음절들을 연결하여 합성하는 합성기이다. DSP 머신 클럭으로는 최대 60 MHz까지 사용할 수 있으므로 향후 추가적인 연산을 필요로 하는 알고리즘을 추가할 경우 좀 더 높은 클럭을 사용하여 여분의 연산 능력을 확보할 수 있다. 단, 이 경우 DSP로부터 코덱을 위한 적절한 주파수의 클럭을 얻을 수 없으므로 코덱에 별도의 오실레이터를 연결하여 주어야한다. 이상에서 설명한 바와 같이 하드웨어 설계의 중점은 재료비의 최소화를 위하여 최소의 부품만으로 설계하는데 두었으며 다음과 같이 요약할 수 있다.

1. 연산 알고리즘의 최적화 및 코드 최적화를 통하여 외부 SRAM 추가 없이 내부 메모리만으로 동작하도록 하였다.
2. 적절한 코덱을 선택하여 마이크 입력을 위한 아날로그 증폭회로를 추가하지 않으며 DSP 머신 클럭의 적절한 설정으로 코덱을 위한 오실레이터를 사용하지 않았다.
3. 외부 장치와의 인터페이스를 위하여 최소의 단자만을 할당하였으며 추가의 로직 소자를 사용하지 않고 외부 장치와의 인터페이스가 가능하도록 하였다.

구현된 음성인식 하드웨어 모듈은 궁극적으로는 외부 장치에 의하여 제어될 것이므로 외부 장치와의 인터페이스가 필요하게 된다. 외부 장치와의 인터페이스는 호환성 및 융통성을 높이기 위하여 최소한의 단자 사용 및 표준 통신 방법을 채택하였다. 외부와의 통신은 5 개의 단자를 통하여 이루어지는데 이중 3 개는 표준 통신 방식 중에 하나인 SPI 통신 방식을 위한 단자이며 2 개의 단자는 각각 인식기 상태를 출력하는 출력단자 및 인식기를 사용하는 호스트의 상태를 읽는 입력 단자로 사용된다. 이렇게 적은 단자를 이용하여 인터페이스를 설계한 이유는 첫째로 인터페이스의 융통성을 높이기 위해서이다. 대부분의 호스트들은 이미 인터페이스를 위하여 사용할 수 있는 단자들을 다른 목적으로 활용하고 있으므로 여분의 단자가 충분하지 않을 수 있다. 이런 경우 최소한의 단자만을 사용하게 함으로서 대부분의 경우 인터페이스가 가능하다. 두 번째로는 TMS320VC33 DSP의 경우 외부 인터페이스를 위해 활용 가능한 단자가 매우 부족하다. 외부 인터페이스를 위한 장치로는 시리얼 포트가 있으나 이는 이미 코덱에 연결되어 있으므로 그 외의 단자들을 이용하여 인터페이스를 만들어야 한다. 별도의 로직을 사용하지 않고 여분의 몇몇 I/O 단자 및 인터럽트 단자를 이용하여 소프트웨어적으로 통신 인터페이스를 구현하기 위해서는 최소한의 단자만을 사용할 수밖에 없었다. 그런 덕분에 추가의 로직 소자를 전혀 사용하지 않고 외부 인터페이스를 구현할 수 있었다.

한편, 가변어휘 인식기의 경우 고정어휘와 달리 복잡한 소프트웨어 인터페이스를 필요로 한다. 여기서 소프트웨어 인터페이스라 함은 소프트웨어 방식의 인식기 경우 API(Application Programming Interace)에 해당하는 것으로 흔히 인식기 제어를 위한 프로토콜이라고도 한다. 단순히 인식을 개시하거나 인식 결과를 전달하는 기본적인 동작 이외에 새로운 단어를 추가하거나 기존의 명령어를 삭제하기 위해서는 음성인식기와 호스트 사이에 약속된 프로토콜이 필요하다. 이를 위하여 다음과 같이 호스트가 사용할 수 있는 커맨드를 규정하였다. 하나의 커맨드는 16 bit로 되어 있다.

RECOG COMMAND : 인식을 개시하는 명령어. 이 명령어에는 마이크의 입력 볼륨 및 인식기의 거부율 설정, 음절합성 여부 등의 옵션이 포함된다.

ADD_LIST COMMAND : 인식 어휘 리스트의 전송을 개시하기 위한 명령어. 전송하고자하는 인식 어휘의 개수와 전송되는 인식 어휘의 한글 방식(완성형 또는 조합형)이 포함된다. ADD_LIST COMMAND를 전송하고 난 후에는 개별적인 인식 어휘의 문자를 전송하기 위하여 STRING 전송 소프트웨어 모듈을 이용하여 전송하고자 하는 인식 어휘 수만큼 루프를 돌면서 인식 어휘의 문자를 전송한다.

APPEND COMMAND : 새로운 인식 어휘 하나를 추가한다. 새로 추가되는 인식 어휘는 인식 어휘 리스트의 마지막에 추가된다.

DELETE COMMAND : 인식 어휘를 삭제한다. 이 명령어에는 삭제하고자 하는 명령어의 인덱스를 포함한다.

위에서 언급한 커맨드들은 내부적으로 구현된 16bit 데이터 송수신 모듈, 8bit 데이터 송수신 모듈들과 같은 하위 계층의 소프트웨어 전송모듈을 이용하여 호스트와의 통신을 수행하게 된다.

4. TMS320VC33 DSP를 이용한 소프트웨어 구현

음성인식기 구현을 설명하기 전에 구현된 소프트웨어의 메모리 맵 및 사용량을 살펴보면 다음과 같다.

표 2. 512 K (0x80000) 워드 플래쉬 메모리 맵

용도	시작 번지	사용량
인터럽트 벡터 테이블	0x00000	0x00040
프로그램 코드 및 테이블	0x00040	0xAA72
음절합성용 DB	0xC000	0x12024
다이폰 모델 DB	0x20000	0x5BFA0
사용자 명령어 저장용	0x7F000	0x1000

가장 많은 메모리를 차지하는 것은 역시 다이폰 모델 DB이며 512 K 워드 플래쉬 메모리의 대부분의 공간이 ROM의 역할로 사용된다. 그러나 0x7F000 이후의 0x1000 크기의 공간은 사용자가 설정한 인식어휘를 저장하여 두는 용도로 사용되므로 읽고 쓰기가 이루어진다. 3 절에서 언급한 음절합성용 DB는 우리말에서 대표음으로 발음되는 1,700여 개의 음절 중 유사발음을 통합하여 1,095 개의 음절을 사용하였으며 메모리 공간 절약을 위하여 G.723.1 음성 부호화 방식으로 압축하여 저장하였다. 음절합성의 경우 문장을 합성하는 데는 적합하지 않지만, 인명이나 고유명사, 짧은 단어를 합성할 경우에는 합성음의 자연성 보다는 명료도가 우선하므로 적은 메모리를 사용하여 인식된 단어를 합성하는 데는 적합한 방식이다. 따라서 표 2의 프로그램 코드 및 테이블 영역에는 음성인식을 위한 코드뿐만 아니라 음절합성을 위한 G.723.1 디코더 코드와 테이블들을 포함하고 있다.

34 K 워드의 내부 메모리는 실행코드를 위한 공간, 실시간 동작을 위한 핑퐁 버퍼, 스택, 그리고 마지막으로 정적데이터 및 아래에서 설명할 동적 할당을 위한 메모리 풀(Pool)을 위하여 사용된다. 일반적으로 DSP의 동작은 속도가 느린 외부 ROM 또는 플래쉬 메모리에 들어 있는 실행 코드를 속도가 빠른 내부 메모리 또는 SRAM으로 옮긴 후 실행을 한다. 그런데 표 2에 나와 있는 프로그램 코드의 크기를 보면 내부 메모리의 크기보다 훨씬 크므로 단순히 프로그램 코드를 내부 메모리

로 옮긴 후 사용하는 것은 불가능하다. 더구나, 만약 평균 3 음절의 인식어휘 200 개를 인식하고자 한다면 34 K 워드 중 5.5 K 워드만을 프로그램 코드 공간으로 사용할 수 있으며 나머지 공간은 버퍼나 스택, 인식어휘를 위한 데이터 공간으로 사용하여야한다. 5.5 K 워드 정도의 적은 내부 메모리로 프로그램 코드를 실행을 하기 위하여 프로그램 코드를 분석한 후 다음과 같이 코드를 분류하였다.

1. 프로그램 동작 시 초기화를 위하여 1번만 실행되는 코드
2. 실시간 동작 시에 사용되나 코드 길이에 비하여 MIPS 사용이 적은 코드
3. 실시간 동작 시에 사용되나 액세스 빈도가 낮은 상수나 테이블
4. 실시간 동작 시에 사용되며 액세스 빈도가 높은 상수나 테이블
5. 실시간 동작 시에 사용되며 많은 MIPS를 사용하는 코드

위와 같이 분류된 코드 중에서 1 번, 2 번과 3 번은 외부 플래쉬 메모리에서 직접 실행이 되며 4 번과 5 번의 코드와 데이터 상수들만 내부메모리로 옮긴 후 실행하게 하였다. 한편 4 번과 5 번의 경우도 한꺼번에 내부 메모리에 옮겨서 실행하는 것이 불가능 하였는데 이를 가능하게하기 위하여 다시 수행 시간 상 겹치지 않는 코드로 분류를 하였다. 프로그램의 동작은 인식이 완료된 후 인식된 결과를 음절 합성기를 통하여 합성을 하게 된다. 음절 합성기는 일종의 편집 합성으로 G.723.1으로 압축된 음절들을 복원한 후 연결하여 출력한다. 인식 알고리즘과 합성 알고리즘은 시간 상 겹치지 않으므로 우선 인식과 관련된 코드만을 내부메모리로 로딩한 후 인식을 수행한다. 인식이 종료되면 합성을 위하여 G.723.1 디코더를 포함한 합성과 관련된 코드를 로딩한 후 합성을 수행하고 합성이 끝나면 곧 바로 인식기 관련 코드를 다시 내부메모리로 로딩하여 다음 인식이 수행될 수 있도록 한다. 내부 메모리의 대부분은 가변어휘 인식기의 인식 어휘를 위한 데이터 메모리로 사용되는데, 필요로 하는 메모리 양은 인식 어휘 수에 비례하게 된다. 적은 데이터 메모리 상에서 많은 수의 인식 어휘를 포함하기 위해서는 인식 알고리즘, 특히 비터비 디코더에서 사용되는 데이터들의 구조가 효율적으로 설계되어야한다. 가변어휘 인식기에서 데이터 메모리를 효율적으로 사용하기 위해서는 동적 메모리 할당이 필수적이므로 이 기능을 효과적으로 구현하여야 한다. 동적 메모리 할당을 사용하지 않고 정적 메모리 구조를 이용할 경우에는 인식과 관련된 데이터의 구조 및 이를 운용하는 프로그램도 간단해지지만 메모리의 낭비가 심하게 된다. 예를 들어 설명하면, 인식 어휘들은 이들을 표현하기 위하여 특정한 데이터 구조를 가지게 된다. 또한 대부분의 인식기들의 경우 한 인식 어휘의 최대 길이를 제한하는데 만약 인식 어휘의 길이를 10 음절로 제한할 경우, 이 명령어를 위한 데이터 구조상에는 10 음절 내에 포함 가능한 최대 다이폰의 개수에 해당하는 배열이 미리 확보되어 있어야 한다. 이럴 경우 인식 어휘들이 대부분 10 음절 미만일 경우 메모리의 낭비가 많게 되며, 다른 의미로는 사용할 수 있는 인식 어휘의 수가 줄어들게 된다. C언어 개발 환경에서는 동적 메모리 할당과 관련된 함수가 런타임 라이브러리에서 제공이 되나 이를 사용하는 것은 관련 함수 자체가 사용하는 메모리 사용량이 크므로 본 인식기처럼 메모리 사용에 제약이 많은 경우에는 라이브러리에서 제공되는 관련 함수를 사용하기 어렵다. 따라서 이런 동적 메모리 할당과 관련된 부분은 본 인식기에서 필요한 기능만을 최소화하여 구현하였다. 이를 위하여 그림 5와 같이 필요한

데이터 구조마다 메모리 풀(Pool)을 만들어서 동적 메모리 할당과 같은 기능을 수행하도록 하였다.

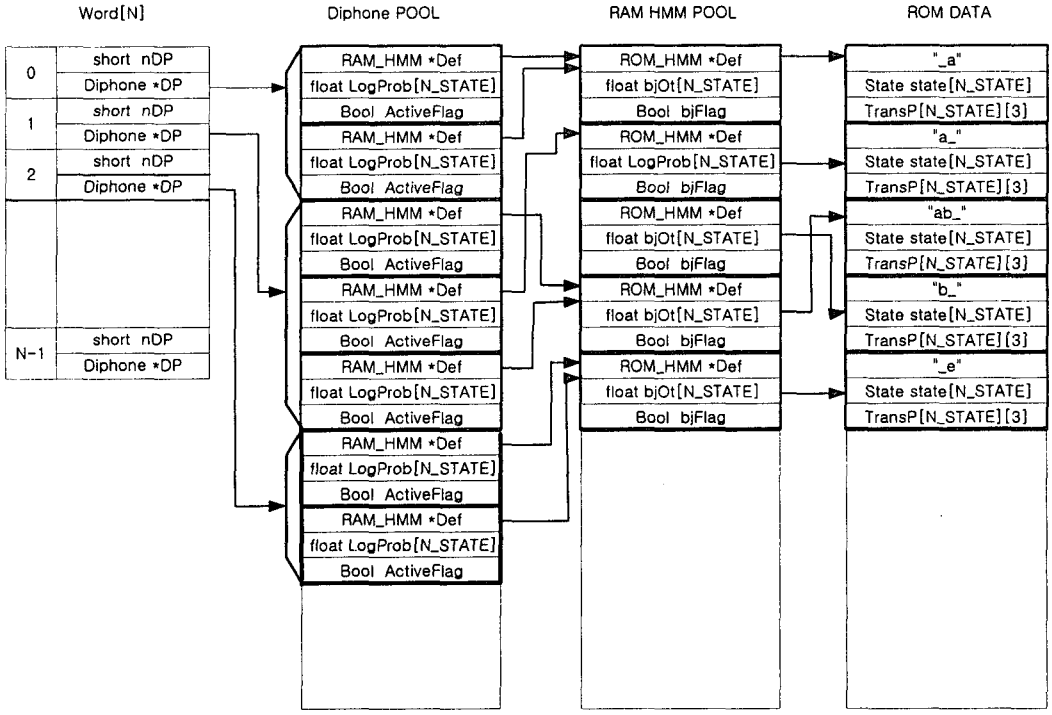

그림 5. 동적 메모리 할당을 위한 데이터 메모리 풀

우선 인식 어휘를 나타내는 Word[N]은 크기가 크지 않으므로 정적으로 할당한다. 여기서 N은 최대 인식 어휘의 수인데 200으로 설정하였다. 이는 가용한 내부 데이터 메모리를 고려하여 산정한 결과, 최대 1,400 개의 다이폰 유닛을 사용할 수 있었는데 이는 3 음절 단어의 경우 평균 7 개의 다이폰으로 구성이 되므로 3 음절 단어 200 개까지 인식이 가능하기 때문이다. 물론 인식 어휘들의 음절수가 평균 3 음절보다 길 경우에는 인식 어휘를 200 개까지 사용할 수는 없다. 한 인식 어휘의 길이가 가변적이므로 한 인식 어휘당 연결되는 다이폰은 동적으로 할당된다. 다이폰의 정보는 Diphone POOL이라는 메모리 영영을 설정하여 사용한다. Diphpone POOL은 Diphone 구조체를 최대 1,400 개까지의 동적으로 할당할 수 있는 공간이다. Diphone 구조체는 한 다이폰을 구성하는 스테이트의 로그 확률값과 푸르닝을 위한 활성화 정보를 저장하는 ActiveFlag, 스코어 캐쉬 방식을 이용하기 위한 연결정보로 구성된다. 한편 RAM HMM POOL은 다이폰을 구성하는 스테이트에 할당된 캐쉬 메모리와 캐쉬 적중정보를 기록하는 bjFlag, 최종적으로 플래쉬 메모리에 저장된 해당 다이폰 HMM 모델을 연결하는 연결정보를 가지고 있다. RAM_HMM 구조체의 구성요소들을 직접 Diphone 구조체에 포함시킬 수 있으나 이럴 경우 메모리 사용의 효율성이 떨어진다. 왜냐하면 Diphone POOL에서는 동일한 다이폰들이 여러 번 나타날 수 있는데 이들 동일한 다이폰에 포함된 스테이트의 로그확률을 한번만 계산하고 계산된 스코어를 캐쉬메모리에 저장하여 나머지 동일한

다이폰에서는 재연산을 하지 않고자 하는 것이 목적이므로 동일한 다이폰들에 대하여는 하나의 캐쉬 메모리만 있으면 되기 때문이다. 이런 개념이 그림에서 Diphone POOL에서 RAM HMM POOL의 연결선을 통하여 보여주고 있다. 따라서 RAM HMM POOL을 별도로 지정하여 메모리 사용의 효율성을 높였다. 그림 5에서 맨 오른쪽의 메모리는 플래쉬 메모리 공간에 해당하며 다이폰 모델들의 HMM 정보들이 State 구조체와 TransP[N_STATE][3]의 전이확률 행렬에 들어 있게 된다. 여기서 N_STATE는 한 다이폰 모델이 가지고 있는 스테이크의 수이다. 다음은 플래쉬 메모리에 저장된 State 구조체의 구조이다.

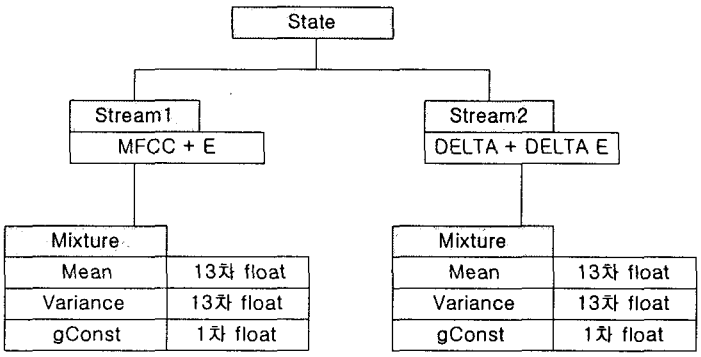

그림 6. State 구조체의 구조

한편 전이확률 행렬은 통상의 경우 TransP[N_STATE][N_STATE]와 같이 정방행렬로 정의 되지만 그림 3과 같이 Left-Right 모델의 경우에는 대각 요소 부근에만 그 값을 가지므로 TransP[N_STATE][3]으로 정의하여 메모리 사용량을 줄였다. 요컨대, 그림 5의 구조는 비터비 디코딩을 위한 인식 어휘의 다이폰 모델 정보 및 필요한 메모리들을 효율적으로 데이터 메모리에 할당하고 비터비 디코딩 시에 실제 다이폰 모델은 플래쉬 메모리로부터 직접 읽어서 사용한다.

그림 7은 제작된 가변어휘 음성인식 모듈과 이 모듈을 평가해 보기 위한 평가 보드이다. 평가 보드는 호스트의 역할을 하는 마이컴이 포함되어 있으며 이를 이용하여 음성인식 모듈을 제어하도록 하였다. 마이컴은 인식을 개시시키고 인식된 결과를 음성인식 모듈로부터 수신하여 결과를 LCD 화면에 출력한다. 음성인식 수행뿐만 아니라 인식어휘의 변경도 수행한다. 다만 인식어휘를 변경하기 위해서는 인식어휘 정보를 입력 받아야하는데 이를 위하여 RS-232 단자를 장착하고 있다. 이 RS-232 단자를 PC의 직렬 포트에 연결하여 PC로부터 인식 어휘의 문자 정보를 수신하도록 하였다. 이를 위하여 사용자가 인식어휘를 직접 화면상에서 입력하거나 인식 어휘들을 저장하고 있는 텍스트 파일을 선택하여 평가보드로 전송하는 간단한 PC 프로그램도 작성하였다.

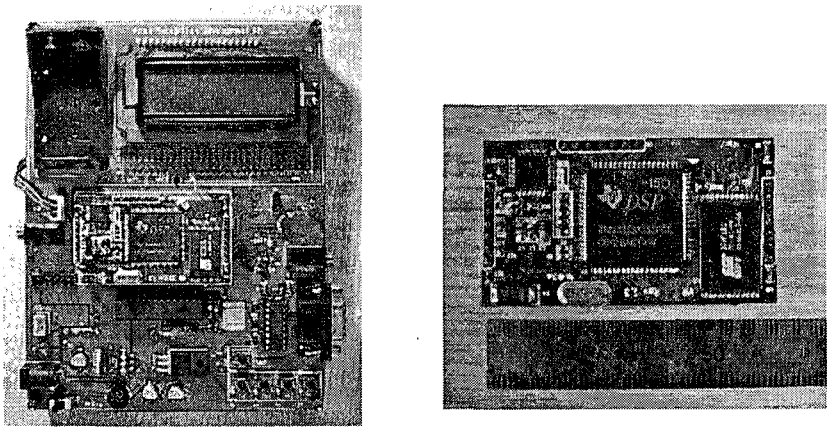

그림 7. 제작된 가변어휘 음성인식 모듈과 평가보드

5. 결 론

본 논문에서는 TMS320VC33 DSP를 이용한 가변어휘 음성인식기 구현을 위하여 사용한 하드웨어 설계 및 소프트웨어 최적화 방법을 제시하였다. 이를 위하여 우선 임베디드 환경에 적합한 다이폰 서브워드 유닛 기반의 가변어휘 인식엔진을 개발하였으며, 개발된 엔진은 국어공학 연구소에서 배포한 고빈도 2,000 어절을 테스트 데이터로 사용하였을 경우 94.5%의 인식률을 보였다. 이렇게 개발된 가변어휘 인식 엔진은 최소한의 부품만을 사용하도록 설계 제작된 DSP 기반의 하드웨어에 이식되었다. 가용할 수 있는 메모리의 제한으로 여러 기법의 최적화 방법을 사용하였다. 우선 프로그램 코드의 경우 분석을 통하여 성능에 영향을 미치는 부분만을 내부 메모리로 로드하여 동작시켰다. 뿐만 아니라 수행 순서 상 겹치지 않는 코드를 나누어 순서대로 로딩하므로서 최소의 메모리로도 프로그램이 수행될 수 있도록 하였다. 데이터 메모리의 경우에는 필요한 데이터 구조들을 동적으로 할당 할 수 있도록 메모리 풀을 만들어 자체적으로 동적메모리 할당을 수행하므로서 메모리 사용의 효율성을 높였다. 이렇게 구현된 가변어휘 음성인식 모듈은 평균 3 음절 인식 어휘를 기준으로 200 어휘까지 인식이 가능하였다. 인식 속도는 3 음절로 된 인식어휘 100 개의 경우 1초 내외가 소요되었으며 200 개의 경우에는 1.5 초 내외의 인식시간이 소요되었다. 한편, 구현된 음성인식 모듈의 활용도를 높이기 위하여 부가적으로 음절합성 기능을 추가하였다. 최종적으로 제작된 음성인식 모듈은 3.5 cm x 6.3 cm의 작은 크기를 가지며 음성인식 모듈을 평가하기 위한 평가보드도 함께 제작하였다.

참 고 문 헌

[1] 2001. *음성처리 시스템. 기술/시장 보고서.* 한국전자통신연구원.
[2] Jean-Claude Junqua. 2000. *Robust Speech Recognition in Embedded Systems and PC Applications.* Kluwer Academic Publishers.
[3] Allen, J., Klatt, D. 1987. *From Text to Speech: The MITALK System.* Cambridge Univ. Press.
[4] 1993. *양질의 음성합성을 위한 최적의 합성 단위 추출에 관한 연구.* 한국전자통신연구원.
[5] Huang, X., Acero, A., Hon, H. 2001. *Spoken Language Processing.* Prentice Hall PTR.
[6] Odell, Steve Young Julian. 2002. *The HTK Book for HTK Vesion 3.2.* Cambridge Univ. Engineering Dept.
[7] 김우성, 구명환. 1999. "반음소 모델링을 이용한 거절기능에 대한 연구." *한국음향학회지,* 제18권 3호, pp. 3-9.
[8] 1999. *TMS320VC33 Digital Signal Processor Data Sheet.* Texas Instruments.

접수일자: 2004. 07. 30
게재결정: 2004. 08. 31

▲ 정익주
강원도 춘천시 효자2동 (우: 200-701)
강원대학교 전기전자정보통신공학부
Tel: +82-33-250-6322
E-mail: ijchung@kangwon.ac.kr