論文97-34C-8-8

LPC-CEPSTRUM 추출을 위한 전용 프로세서의 설계

(A Design of the Processor Dedicated to LPC-CEPSTRUM)

黃 仁 哲 *, 金 成 男 *, 金 盈 佑 *, 金 太 根 **, 金 壽 遠 *

(In-Chul Hwang, Sung-Nam Kim, Young-Woo Kim, Tae-Geun Kim, and Soo-Won Kim)

요 약

본 논문에서는 음성인식용 LPC cepstrum 프로세서를 CMOS 게이트 어레이 공정으로 구현하였다. 설계된 프로세서는 전체 연산의 대부분을 차지하는 correlation 연산을 빠르게 처리하기 위하여 24 비트 부동소수점 MAC(Multiplier & ACcumulator)을 내장하였고, 임시변수 저장을 위해 22개의 레지스터 파일을 포함하고 있다. 부동소수점 MAC은 3단 파이프라인 구조로 설계하였으며 새로운 방식의 후처리 기법을 제안, 적용하여 그 실용성을 검증하였다. 설계에 소요된 게이트수는 27,760 게이트였으며, 프로토타입 보드를 제작하여 성능을 측정한 결과 15 MHz 클럭에서 200 샘플 한 프레임을 처리하는데 약 266 μs의 시간이 소요되었고, 최대 동작 클럭 속도는 16.6 MHz였다.

Abstract

An LPC cepstrum processor for speech recognition is implemented on CMOS gate array process. The designed processor contains a 24-bit floating-point MAC unit to perform the correlation quickly, which occupies the majority of operations used in the algorithm, and has 22 register files to store temporary variables. For the purpose of fast operations, the floating-point MAC consists of a 3-stage pipeline and the new post-normalization scheme is proposed and applied to it. Experimental result shows that it takes approximately 266 μs to process 200 samples/frame at 15 MHz clock rate. This processor runs at the maximum rate of 16.6 MHz and the number of gates are 27,760.

I. 서 론

Cepstrum 계수는 LPC 계수나 PARCOR 계수와 같은 여타의 음성 특징 계수보다 신뢰성이 높기 때문에 음성인식에서 가장 널리 사용되고 있는 특징 파라메타이다[1] [2] [3]. Cepstrum은 음성신호 스펙트럼의 로그값의 푸리에 변환으로 정의할 수 있는데, 음성을 전극(all-pole)필터 모델로 근사화하면 LPC 계수로부터 선형 전개에 의해 cepstrum계수를 얻을 수 있다. 이렇게 LPC 계수로부터 추출된 cepstrum계수를 LPC cepstrum 계수라고 한다[1].

LPC cepstrum 계수를 계산하는 알고리즘을 하드웨어로 구현하는 방법에는 크게 두가지 방식이 있다. 첫째는 범용 DSP나 마이크로프로세서에 주어진 알고리즘을 프로그래밍하여 소프트웨어 프로세서(software processor)를 구현하는 방식이고, 다른 하나는 알고리즘에 최적화된 아키텍처를 갖는 전용의 프로세서

* 正會員, 高麗大學校 電氣·電子·電波工學部
(School of EEE , Korea Univ.)
** 正會員, 現代電子 System IC Lab.
(System IC Lab., Hyundai Electronics Industries Co. Ltd.)
接受日字:1997年5月2日, 수정완료일:1997年7月26日

(dedicated processor)를 구현하는 방식이다. 이들을 비교하면 소프트웨어 프로세서는 프로세서의 여러 사양(specification)들을 프로그래밍할 수 있기 때문에 쉽게 프로세서를 수정할 수 있고 단일 프로세서를 이용하는 소규모 시스템에서부터 다중 프로세서 형태로 구성되는 대용량 고속 시스템까지 다양한 규모의 프로세서를 구현할 수 있다는 장점을 갖는다. 그러나 시스템 구성이 복잡해지고, LPC cepstrum 알고리즘과는 직접적으로 관련이 없는 명령어나 불필요한 기능블럭들, 그리고 필요이상의 내부 메모리들로 인해 잉여성분(redundancy)이 발생할 뿐만 아니라 외부 기억장치의 잦은 엑세스로 인한 속도저하가 발생한다는 문제점을 갖는다[4]. 이와 같은 문제점들때문에 소규모 시스템에서 음성인식 기능을 구현하기 위해서는 하드웨어 잉여성분을 최소화하여 프로세서를 단순화하고 프로세서 주변에 연결되는 메모리나 제어로직들이 필요없도록 전용 프로세서를 구현하여 시스템을 단순화 시켜야 한다. 이에 따라 본 논문에서는 LPC cepstrum알고리즘에 대해 시스템 구성이 간단하고 범용 프로세서에 비해 고속연산이 가능한 전용의 프로세서를 설계, 구현하였는데 cepstrum 알고리즘만을 구현한 기존의 것과는 달리 원음성부터 cepstrum까지의 전과정을 전용 프로세서로 구현하였다[9].

Ⅱ. 알고리즘

LPC계수를 계산하는 방법에는 자기상관법(auto-correlation)과 covariance법이 있는데, 자기상관법이 LP모델에 대해 더욱 안정된 성능을 제공하기 때문에 음성인식에서는 일반적으로 자기상관법이 사용된다[1][2].

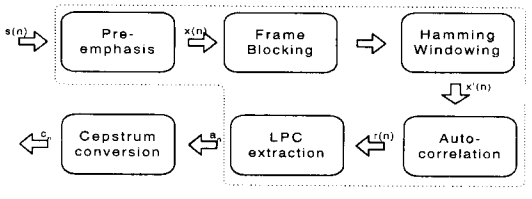


그림 1. LPC cepstrum을 계산하는 절차
Fig. 1. Procedure of calculating the LPC cepstrum.

LPC cepstrum 프로세서는 한 프레임(200 샘플, 20ms)의 음성 표본치 s(n)으로부터 그림 1과 같이 5단계의 연산과정을 거쳐 10개의 cepstrum계수 c_n을 추출하는데 점선으로 표시된 부분은 자기상관법을 이용한 LPC 계수 연산 절차의 일반적인 다이아그램이다[1].

그림 1의 과정을 자세히 살펴보면 먼저 입력된 음성 표본치 s(n)에 대해 프리엠퍼시스(preemphasis)를 하는 것을 알 수 있는데, 이는 (1)식과 같이 1차 고역통과 필터링으로서 연산중에 발생하는 오버플로우를 방지하기 위해 음성 스펙트럼의 동적영역(dynamic range)을 줄이는 역할을 한다.

$$x(n) = s(n) - \hat{a}s(n-1),$$

여기서 $\hat{a} = 1$, s(n)은 입력 음성 샘플 (1)

다음으로 연속된 음성 샘플을 200 샘플의 분석구간과 100 샘플의 중첩구간으로 나누는 frame blocking을 하고 이에 따른 프레임간의 불연속성을 보상하기 위해 (2)식과 같이 hamming windowing을 한다.

$$x'(n) = x(n)w(n), \quad n = 0, 1, \ldots, 99 \qquad (2)$$

, 여기서 w(n)은 Hamming window 계수로서,

$$w(n) = 0.54 - 0.46\cos\left(\frac{2n\pi}{N-1}\right)$$

Hamming windowing을 하고 나면 200 샘플의 분석구간이 제대로 정의되므로 (3)식과 같이 11개의 자기상관 계수 r(m)를 계산할 수 있다.

$$r(m) = \sum_{n=0}^{199-m} x'(n)x'(n+m), \quad m = 0, 1, \ldots, 10 \qquad (3)$$

그리고 이 11개의 계수로부터 Durbin의 방법을 이용하여 10개의 LPC 계수 a_n를 추출하게 되는데 Durbin의 방법은 (4)식과 같이 정리할 수 있다.

$$E(0) = r(0)$$

$$k_i = \frac{r(i) - \sum_{j=1}^{i-1} a_j(i-1)r(|i-j|)}{E(i-1)}$$

$$a_i(i) = k_i$$

$$a_j(i) = a_j(i-1) - k_i a_{i-j}(i-1), \quad j = 1, 2, \ldots, i-1$$

$$E(i) = (1-k_i^2)E(i-1), \quad i = 1, 2, \ldots, 10 \qquad (4)$$

마지막으로 LPC 계수로부터 cepstrum계수 c_n로 변환하는 식은 (5)식으로 표현된다.

$$c_1 = -a_1$$

$$c_n = -a_n - \sum_{m=1}^{n-1}(1-\frac{m}{n})a_m c_{n-m} \ , \ n = 2 \ , \ 3 \ , \ \cdots\cdots \ , \ 10 \quad (5)$$

위의 알고리즘 각 단계에서 사용된 연산자를 덧셈과 곱셈으로 나누어 각각의 연산횟수에 따라 정리하면 그림 2와 같은 결과를 얻을 수 있다. 그림 2의 결과로부터 LPC cepstrum계수를 계산하는 알고리즘에서 대부분의 계산량을 차지하는 단계가 자기상관계수 연산 단계라는 것을 알 수 있다. 그러므로 전체 연산과정을 빠르게 수행하기 위해서는 프로세서가 자기상관계수

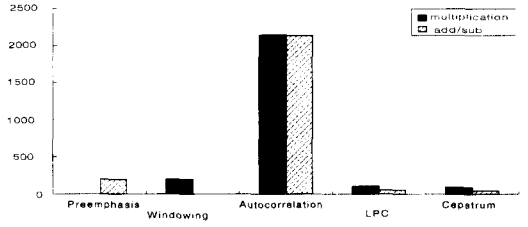


그림 2. 각 단계에서의 연산 발생 빈도
Fig. 2. Number of operations at each step.

연산 단계를 빠르게 처리할 수 있도록 하는 것이 효율적이다. 그런데 자기상관계수는 수식 (3)에서 알 수 있는 바와 같이 기본 연산단위가 곱의 합으로 구성된 루프연산이며, 매 루프마다 다른 두 개의 입력 데이타 x'(n) , x'(n+m)를 사용하기 때문에 메모리 엑세스에 따른 부하가 매우 큰 특성을 갖는다. 그러므로 자기상관계수 연산에서 발생하는 이러한 과도한 메모리 엑세스 횟수를 최소화하고, 자기상관 계수 연산 루프의 기본 연산인 곱셈과 덧셈을 동시에 수행할 수 있는 MAC (Multi-plier & ACcumulator)으로 산술연산부를 구성하면 프로세서의 처리 속도를 높일 수 있다.

Ⅲ. 아키텍쳐

설계된 프로세서는 산술연산부인 3단 파이프라인 FP-MAC과 임시저장용 레지스터 파일 그리고 이들 간의 데이타 흐름을 제어하는 FSM(Finite State Machine)의 3개 주요블럭으로 구성된다[5]. 그 밖의 구성블럭으로는 14비트 2의 보수형태인 입력 데이타를 24비트 부동 소수점 형태로 변환해주는 포맷 변환기와 해밍 윈도우 계수가 저장되어 있는 해밍 롬, 그리고 어드레스를 발생시키는 어드레스 발생기가 있다. 그림 3은 프로세서의 전체 구조를 블럭도로 도시한 것이다.

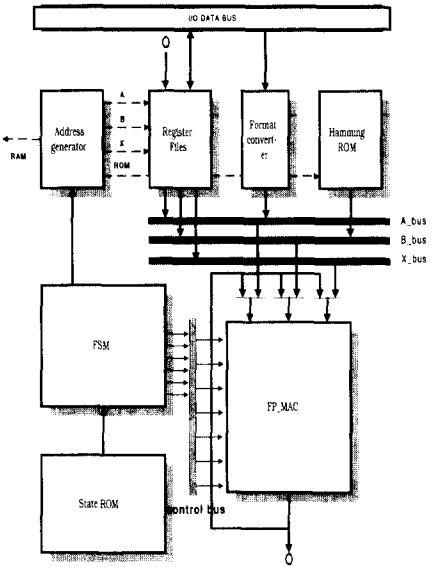


그림 3. 설계된 프로세서의 블록도
Fig. 3. Block diagram of the designed processor.

1. 제어부 (FSM)

파이프라인 형태로 구성된 FP-MAC의 효용도(throughput)를 높이기 위해 파이프라인에서 발생하는 데이타 의존도(data-dependency)을 줄이도록 알고리즘을 스케쥴링하여 제어부를 구성하였는데, 표 1에는 설계된 프로세서의 FP-MAC에서 발생하는 연산간의 데이타 의존도의 지표로서 파이프라인 FP-MAC의 활용도인 1 클럭당 평균 연산횟수를 제시하였다. 표 1의 결과를 살펴보면 FP-MAC의 전체 활용도는 74% 정도이지만 LPC 단계에서 활용도가 특히 낮아지는 것을 알 수 있는데, 그 이유는 LPC 연산 중에 발생하는 나눗셈 처리를 위해 곱셈을 이용하여 나눗셈을 수행하는 Newton-Raphson의 반복 연산을 적용하였기 때문이다. Newton-Raphson의 근사법은 젯수의 역수의 근사값을 계산하여 피젯수에 곱해주는 방식으로 나눗셈을 수행하는데, 젯수의 역수의 근사값을 계산하는 루프에서 루프내의 변수들이 상호 의존관계에 있기 때문에 연산횟수대 클럭비가 나빠진다.

2. 레지스터 파일

설계된 프로세서는 레지스터 파일을 그림 4와 같이 A, B 두 개의 뱅크로 나누어 A 뱅크에는 x'(n)을 queue형태로 저장하고 B뱅크에는 11개의 r(m)계수를

표 1. 파이프라인 FP-MAC의 활용도
Table 1. Usage of the pipelined FP-MAC.

	W&A	LPC	Cepstrum	Total
$\frac{\text{MAC operations}}{\text{clocks}} \times 100\%$	85.7%	34.4%	92%	73.7%

※ W & A는 Windowing & Autocorrelation

저장하여 한번 입력된 x'(n)이 각각의 r(m)계수 계산에 모두 이용되도록 구성하였다. 그리하여 자기상관계수 연산 중에 발생하는 하나의 입력 데이타에 대한 반복적인 메모리 엑세스를 방지하였고, 다른 연산단계에서도 계산 중간에 발생하는 임시 데이타들을 이 레지스터 파일로 처리하도록 하였다. 이와 같이 구성함으로써 외부 메모리는 원음성을 저장하는 메모리만 필요하게 되고, 프로그램 메모리나 여타의 데이타 메모리가 필요없게 된다.

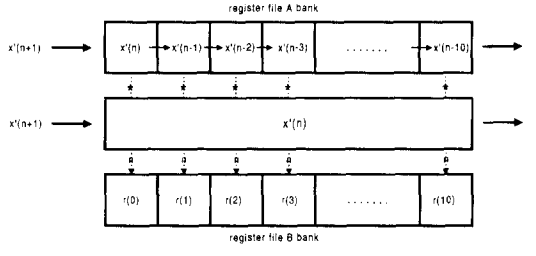


그림 4. 자기상관계수 연산에서 레지스터 파일의 데이타 구성
Fig. 4. The data structure of register files at auocorrelation step.

3. 24비트 FP-MAC

설계된 FP-MAC은 24비트 부동소수점 MAC연산(Y=X+A*B)을 수행하며, 3단 파이프라인으로 구성되었다. 동작은 그림 5의 FP-MAC 블럭도와 같이 파이프라인의 첫째단에서는 곱셈을 수행하고 둘째단에서는 덧셈을 수행하며 마지막 셋째단에서는 post-normalization과 round-off를 수행한다. 이와 같이 부동소수점 덧셈과 곱셈을 하나의 연산으로 처리하게 되면 부동소수점 덧셈기와 곱셈기의 연결에 따른 버스용량을 줄일 수 있고, post-normalization과 round-off 같은 후처리를 한 번에 처리하기 때문에 연산시간을 줄일 수 있는 장점이 있다.

이러한 후처리를 하기 위해 일반적으로 사용되는 방법은 LZD(Leading Zero Detector)를 이용하여 post-normalization에 필요한 천이값을 검출하는 방식이다. LZD는 부동소수점 덧셈기에서도 사용되는 기본 블럭으로서 덧셈 이후에 덧셈 결과로부터 MSB(Most Significant Bit)의 위치를 검출하여 post-normalization에 필요한 천이값으로 디코딩하는 역할을 수행한다. 이외에도 덧셈연산 전에 연산에 사용되는 가산자와 피가산자로부터 천이값을 미리 예측하는 LZA (Leading-Zero Anticipator)를 이용하는 방식이 있지만 설계가 복잡하고 많은 면적을 차지한다[8] [10].

반면에 LZD를 사용하는 방식의 단점은 LZD가 덧셈기와 shifter 사이에 직렬로 연결되기 때문에 LZA를 사용하는 경우보다 지연시간이 크다는 것인데, 본 논문에서는 이러한 단점을 보완할 수 있는 새로운 방식을 제안하였다. 제안된 방식은 FP-MAC의 첫째 단에서 A, B, X의 exponent를 비교하여 shifer &sticky를 통해 X에 대해 자리맞춤을 위한 천이를 하게 되는 데 이 때 얻어지는 shifter의 천이값으로부터 post-normalization에 필요한 천이값을 미리 예측할 수 있다는 원리를 이용한다.

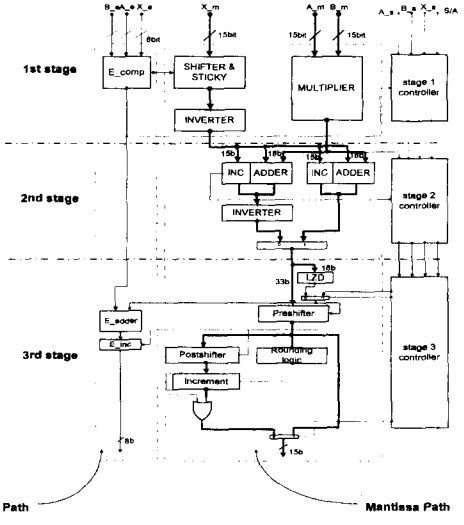


그림 5. 제안된 FP-MAC의 블록도
Fig. 5. Block diagram of the proposed FP-MAC.

그림 6은 FP-MAC의 덧셈 단의 상위 15비트 증가기(incrementer)와 하위 18비트 덧셈기를 region I과 region II로 나누어 도시한 것인데, 이 때 Y=X+A*B의 연산에서 곱셈기 출력(A*B)의 상위 15비트와 가드 비트(guard bit) 3비트를 덧셈기 입력 1 (어두운 부분)에 고정시키면, exponent 비교에 의해 결정된 X의 MSB의 위치는 region I이나 region II에 놓이게

된다.

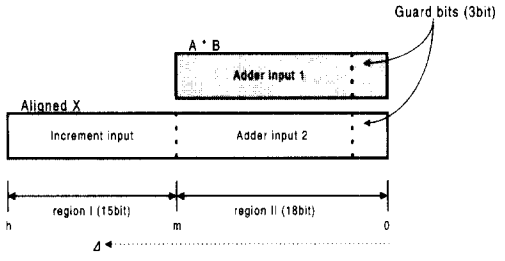


그림 6. 두 번째 단 덧셈기에서의 데이타 포맷
Fig. 6. Data format at adder input of the 2nd stage.

이 때 X의 MSB 위치와 수행된 연산의 종류를 알면 X+A*B 연산결과의 MSB 위치를 최대 1비트 오차로 예측할 수 있는데, 이를 분류하면 다음의 4가지로 정리될 수 있다.

여기서 Δ는 자리맞춤된 X의 MSB 위치로 가정하고 Υ는 X+A*B의 MSB 위치로 가정한다.

1. Δ가 region I에 있고 덧셈을 하는 경우,
Υ는 Δ 또는 Δ+1의 위치에 있게 된다.

2. Δ가 region I에 있고 뺄셈을 하는 경우,
Υ는 Δ 또는 Δ-1의 위치에 있게 된다.

3. Δ가 region II에 있고 덧셈을 하는 경우,
Υ는 m 또는 m+1의 위치에 있게 된다.

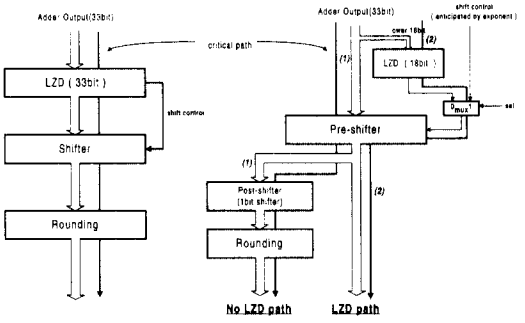
4. Δ가 region II에 있고 뺄셈을 하는 경우,

A. $|\Delta - m| > 1$ 인 경우,
Υ는 m 또는 m-1의 위치에 있게 된다.

B. $|\Delta - m| \leq 1$ 인 경우,
Υ는 예측 불가능.

제안된 방식의 post-normalization에서는 기본적으로 첫째단에서 얻은 X의 천이값을 이용하여 normalize값을 결정하고, 위의 경우중 4.(B)와 같이 예측이 불가능한 경우에 대해서만 LZD를 적용하도록 한다. 이를 기존의 방법과 비교하면 먼저 기존의 방법의 경우 그림 7(a)에서와 같이 덧셈기 출력 33비트에 대해서 LZD를 적용한 후에 이로부터 얻어진 천이값으로 post-normalization을 수행하고 마지막으로 round-off처리를 하게 된다.

그러나 제안된 방식은 그림 7(b)와 같이 경로 (1)에서는 LZD를 거치지 않고 preshifter와 1비트 post-shifter를 거친 다음 round-off처리를 하게 되는데 이때 사용되는 postshifter는 multiplexer 1단으로 구성되기 때문에 5단의 multiplexer로 구성된 33비트 LZD에 비해 1/5 정도의 게이트 지연을 갖는다.



(a) 기존의 방법 (b) 제안된 방법

그림 7. 기존의 방법과 제안된 방법의 비교
Fig. 7. Comparison between the conventional method and the proposed one.

그리고 경로 (2)는 18비트 LZD로 정확한 천이값을 얻은 후 preshifter를 통해 normalize를 하는데 결과값의 유효 비트수가 15비트 이내이기 때문에 round-off처리는 생략된다. 그리고 두 경로가 서로 분리되어 있기 때문에 LZD가 전체 경로에 대해 추가의 지연으로 작용하지 않음을 알 수 있다.

Ⅳ. 실험 결과 및 분석

1. 모의 실험 결과

설계된 프로세서의 동작 검증을 위해서는 추출된 LPC 계수에 대한 주파수 영역에서의 검증이 필요한데 이를 위해 그림 8과 같은 과정으로 모의실험하였다. 그림 8에서 상위 경로는 설계된 회로로부터 모의실험을 통해 LPC계수를 계산하여 MATLAB으로 주파수 영역 특성을 얻고, 하위 경로는 입력 음성 샘플에 대해 직접 MATLAB을 이용해 FFT 계산하여 주파수 영역 특성을 얻는다. 이러한 방식을 이용하여 "아버지" 라는 발음의 처음 연속 두 프레임에 대해 모의실험한 결과 그림 9의 주파수 특성을 얻을 수 있었다. 그림에서 x축은 radian 주파수를 나타내고, y 축은 정규화된 진폭을 나타낸다.

이와 같이 LPC 계수의 주파수 특성을 도시하면 입력음성에 대한 주파수특성 곡선의 포락선을 근사

(approximation)하게 되는 것을 알 수 있는데 여기서 중요한 것은 진폭의 피크(formant)가 발생하는 주파수 위치이다. 그런데 그림 9에서 알 수 있듯이 두 모의실험 결과는 서로 같은 주파수에서 비슷한 포락선을 나타내고 따라서 설계된 프로세서가 올바른 LPC계수를 추출하는 것을 알 수 있다.

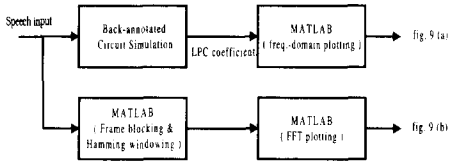


그림 8. 모의실험 확인 절차
Fig. 8. Procedure to verify simulation results.

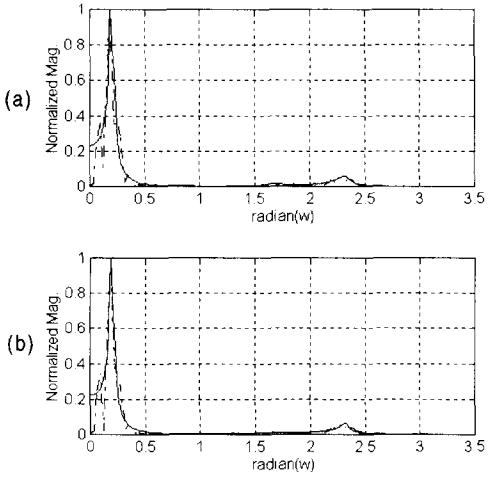


그림 9. LPC계수의 모의실험 주파수 특성
(a) 첫째 프레임에 대한 분석 (b) 두번 째 프레임에 대한 분석
(———, lpc 결과 ·····, fft 결과)
Fig. 9. Frequency response of the simulated LPC coefficients.

2. 실험 결과

본 LPC cepstrum 프로세서는 음성구간을 결정하는 끝점검출기와 패턴 매칭에 의해 인식을 수행하는 DTW(Dynamic Time Warp)프로세서가 함께 하나의 음성인식칩으로 구현되었다.

제작된 IC는 208핀 QFP로 패키지되었으며 전체 설계에 66,760게이트가 소요되었는데, 이 중 LPC cepstrum 프로세서가 차지하는 게이트수는 27,760게이트였다. 그림 10은 실험을 위해 제작된 프로토타입 보드의 기능적 블록도로서, 프로세서의 제어를 위한 호스트 프로세서로 NEC의 V25 프로세서를 사용하였다.

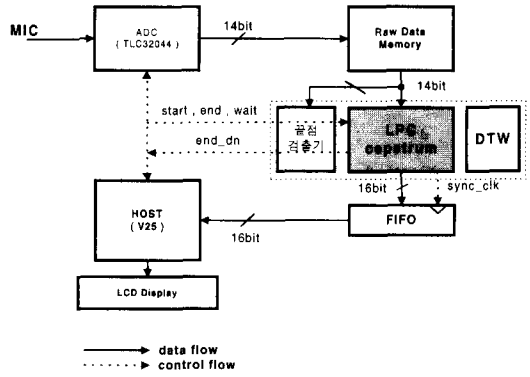


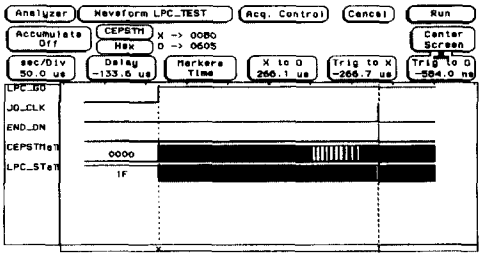
그림 10. 제작된 프로토타입 보드의 블럭도
Fig. 10. Block diagram of the prototype board.

그림 11은 설계된 전용 프로세서를 15MHz 클럭에서 동작시킬 때 한 프레임(20ms)의 음성샘플에 대한 동작 파형을 제시하고 있다. 먼저 그림 11(a)를 살펴보면, start(LPC_GO)신호가 '1'이 될 때 프로세서는 idle(LPC_ST="1F")상태에서 Window&Autocorrelation를 계산하는 상태로 천이하게 되며 한 프레임에 대해 LPC cepstrum계수가 추출되면 sync_clock (JO_CLK)과 end_dn(END_DN)의 두 동기신호를 이용하여 추출된 계수를 외부 FIFO로 출력한다. 이와 같이 LPC_GO가 활성화될 때부터 10개의 추출된 계수가 모두 출력된 후 end_dn 신호가 활성화될 때까지의 시간차가 한 프레임의 음성샘플을 처리하는데 소요되는 시간이 되는데 이 값이 266.1 μs이다. 그러므로 N개의 프레임으로 구성된 음성에 대해 N*266.1 [μs] 의 계산시간이 소요되는 것을 알 수 있다.

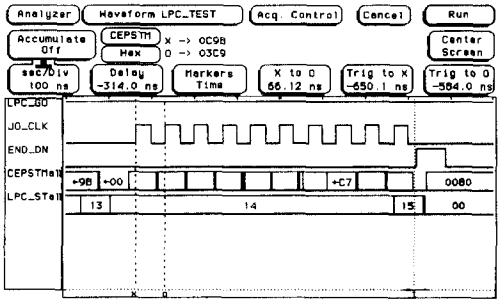
그림 11(b)는 추출된 LPC cepstrum계수를 외부 FIFO에 출력하는 부분을 확대한 것으로 동기에 사용되는 sync_clk는 시스템 클럭과 같은 주기인 66.12ns의 주기를 가지며 rising edge때에 계수 데이타가 유효한 것을 알 수 있다. 10개의 계수를 출력한 후에는 end_dn 신호가 한 클럭 주기동안 발생하게 된다. 참고로 프로토타입 보드에서 전체 음성인식 IC의 인식실험을 수행한 결과 50단어 화자종속 음성 인식실험에서 약 91%의 인식률을 얻을 수 있었다.

실험 결과로부터 LPC cepstrum 프로세서의 제원

을 정리하면 표 2와 같다.



(a) 음성 한 프레임의 동작 파형



(b) 추출된 계수를 출력하는 순간의 동기신호 파형

그림 11. 한 프레임에 대한 출력 결과
Fig. 11. Cepstrum results of 1 frame of speech.

표 2. 설계된 프로세서의 제원
Table 2. Specifications of the designed processor.

공 정	0.8um CMOS 게이트 어레이
Package	208pin QFP
게이트 수	27,760 (LPC cepstrum)
동작 주파수	10MHz (typical) 16.6MHz (maximum)
전원 전압	5V single supply

V. 결 론

본 논문에서는 음성인식용 LPC cepstrum 프로세서를 구현하였다. 설계된 프로세서는 LPC cepstrum 알고리즘 특성상 autocorrelation연산에서 연산 부하와 메모리 부하가 가장 크기 때문에 22개의 24비트 내부 레지스터 파일을 이용하여 하나의 원음성(raw speech) 샘플에 대해 1회의 엑세스로 연산 가능하도록 설계하였으며, 더불어 임시 변수 저장용 외부 메모리를 제거했다. 그리고 autocorrelation연산을 빠르게 처리하기 위하여 전용의 24비트 FP-MAC을 구현하였는데, 설계된 FP-MAC은 exponent 비교에서 얻은 정보를 이용하여 기존의 LZD에서 발생하는 속도저하를 보완하는 새로운 post-normalization기법을 적용하여 critical 경로상에 LZD 대신에 기존의 LZD보다 1/5 정도의 게이트 지연을 갖는 1비트 shifter를 사용함으로써 빠르게 post-normalization을 처리할 수 있었다. 그리하여 15MHz 클럭에서 프로세서를 동작시켰을 때 20ms의 음성 한 프레임으로부터 10개의 LPC cepstrum 계수를 추출하는 데 266.1μs이 소요되었다. 시스템 구성에서는 외부 메모리로 원음성 샘플을 저장하는 메모리만을 사용하고 프로세서와의 인터페이스도 제어신호 5개(start, end, wait, sync_clk, end_dn)로 처리가 가능하므로 시스템 구성이 용이하게 된다. 그러므로 본 프로세서는 음성인식 전화기와 같은 간단한 시스템에서 음성인식 기능을 구현하는데 이용될 수 있을 것이다.

참 고 문 헌

[1] Shuzo Saito, "Fundamentals of Speech Signal Processing," *Academic Press*, 1985.

[2] J.R. Deller, J.G. Proakis and J.H.L. Hansen, "Discrete-Time Processing of Speech Signals ," *Macmillan Publishing Company*, 1993.

[3] L. R. Rabiner and R. W. Schafer, "Digital Signal Processing of Speech Signal," *Prentice Hall*, 1978.

[4] H. M. Ahmed, R. B. Kline, "Recent Advances in DSP Systems," *IEEE Communication Magazine*, pp 32-45, May, 1991.

[5] D. A. Patterson, J. L. Hennessy, "Computer Organization & Design: The Hardware / Software Interface," *Morgan Kaufmann Publishers*, 1994.

[6] N. Ide et. al., "A 320-MFLOPS CMOS Floating-Point Processing Unit for Superscalar Processors," *IEEE Journal of Solid State Circuits*, pp 352-361, Mar., 1993

[7] R. K. Montoye, E. Hokenek, S. L. Runyon, "Design of the IBM RISC System/6000 floating-point execution unit," *IBM J. of RES. & DEVELOP.*, vol. 34, no. 1, pp 59-70, Jan., 1990.

[8] E. Hokenek et. al., "Second-Generation RISC Floating Point with Mltiply-Add Fused," *IEEE Journal of Solid State Circuits*, vol. 25 no. 5, pp 1207-1213, Oct., 1990.

[9] A. N. Suen et. al., "A Cepstrum Chip : Architecture and Implementation," *in Proceedings of IEEE International Symposium on Circuits and Systems*, (USA), vol. 2, pp 1428-1431, IEEE Circuits and Systems Society, Apr 1995.

[10] H. Suzuki et. al., "Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition," *IEEE Journal of Solid State Circuits*, vol. 31, no. 8, pp 1157-1164, Aug., 1996.

저 자 소 개

黃 仁 哲(正會員)

1970년 10월 1일생. 1993년 2월 고려대학교 전자공학과 졸업. 1995년 8월 고려대학교 대학원 전자공학과(공학석사). 1995년 9월 ~ 현재 고려대학교 대학원 전자공학과 박사과정. 주관심 분야는 저전력 디지털 회로 설계, clocking & I/O, clock 발생기용 PLL 등임

金 盈 佑(正會員)

1969년 12월14일생. 1994년 2월 고려대학교 전자공학과 졸업. 1996년 8월 고려대학교 대학원 전자공학과(공학석사). 1996년 9월 ~ 현재 고려대학교 대학원 전자공학과 박사과정. 주관심 분야는 저전압/전전력 회로 설계, 다치논리회로 설계, 마이크로프로세서 설계 등임

金 壽 遠(正會員) 第 33卷 A編 第 8號 參照

金 成 男(正會員)

1968년 8월 23일생. 1991년 2월 고려대학교 전자공학과 졸업. 1993년 2월 고려대학교 대학원 전자공학과(공학석사). 1993년 3월 ~ 현재 고려대학교 대학원 전자공학과 박사과정. 1995년 3월 ~ 현재 고려대학교 부설 정보통신기술공동 연구소 연구원. 주관심 분야는 저전력 디지털 회로 설계, 이동 통신용 부품 개발, 음성 신호처리 등임

金 太 根(正會員)

1955년 12월 9일생. 1979년 2월 서울대학교 전자공학과 졸업. 1981년 8월 서울대학교 대학원 전자공학과(공학석사).1988년 12월 Texas A&M University 대학원 전자공학과(공학박사). 1989년 3월 ~ 1993년 12월 현대전자 반도체 연구소(책임 연구원, 수석 연구원). 1994년 1월 ~ 1997년 5월 현대전자 반도체 제 2연구소(수석 연구원). 주관심 분야는 Digital Signal Processor 설계, Mixed Analog/Digital VLSI 설계 등임