論文 86-23-1-14

실시간 SIFT 기본주파수 검출기의 구현

(Implementation of a Real-time SIFT Pitch Detector)

李 鍾 碩*, 李 商 郁*

(Jong Seok Lee and Sang Uk Lee)

要 約

본 논문에서는 LPC보코더 개발을 위해서 고속 신호처리 프로세서인 NEC 7720을 사용하여 실시간 동작 기본주파수 검출기를 구현하였다. 기본주파수 검출을 위하여 기본적으로 SIFT알고리듬을 사용하였다. SIFT알고리듬은 디지탈 저대역 필터, inverse필터, ACF의 계산, 최대값 추출, 유성음/ 무성음의 판단, interpolation 그리고 smoothing으로 이루어진다. 그러나 유성음/무성음 판단 및 smoothing 부분은 수정하여 사용하였다. 모든 연산은 16bit 고정숫자점 연산으로 하였으며 고정 숫자점 연산에 의한 컴퓨터 시뮬레이션 결과는 실제 음성 데이타로부터 목측된 기본주파수와 비교, 확인하였다. 실제 제작된 기본주파수 검출기는 분석구간인 한 frame을 128개의 음성샘플로 하였을 경우 98.8%의 instruction ROM, 37%의 데이타 ROM과 94%의 내부 RAM을 사용하여 15.2msec의 시간이 소요되었으며 실험 결과는 실제 시뮬레이션 결과와 거의 일치함을 확인하였다.

Abstract

In this paper, a real-time pitch detector for LPC vocoder as implemented on a high speed digital signal processor, NEC 7720, is described. The pitch detector was based mainly on the SIFT algorithm.

The SIFT pitch detector consists primarily of a digital low pass filter, inverse filter, computation of autocorrelation, a peak picker, interpolation, V/UV decision and a final pitch smoother. In our approach, modification, mainly on the V/UV decision and a final pitch smoother, was made to estimate more accurate pitches. An 16-bit fixed-point arithmatic was employed for all necessary computation and the simulated results were compared with the eye detected pitches obtained from real speech data.

The pitch detector occupies 98.8% of the instruction ROM, 37% of the data ROM, and 94% of internal RAM and takes 15.2 ms to estimate a pitch when an analysis frame is consisted of 128 sampled speech data. It is observed that the tested results were well agreed with the computer simulation results.

I. 序 論

1960년부터 컴퓨터를 이용한 디지탈 신호처리 이론이 발달된 이래 음성신호를 디지탈 형태로 변환하는 음성코우딩에 관한 연구가 활발히 진행되어 왔다[1,2] 그러므로 하드웨어에 의한 실시간 동작 CODEC (coder/decoder)의 개발은 아주 중요한 의미를 갖는다[3] 그러나 지금까지는 비교적 코우딩 알고리듬이 간단한 DM (delta modulation) 이나 DPCM (differential pulse code modulation) CODEC의 개발이 많이 연구되어 왔다[4-6] 그런데 한정된 채널에서 채널의

*正會員, 서울大學校 制御計測工學科

(Dept. of Control and Instrumentation Eng., Seoul National Univ.)

接受日字 : 1985年 3月 6日

전송 대역폭을 더욱 줄이기 위해서는 전송 정보량이 적을수록 유리하므로 waveform 코우딩 방법에 비해 전송 정보량이 적은 source 코우딩 방법에 대한 관심이 높아지고 있지만 source 코우딩은 알고리듬이 복잡하여 실시간 동작 CODEC 개발에 많은 어려움이 있었다. 그러나 최근 VLSI 기술을 이용한 고속 디지탈 신호 처리용 DSP (digital signal processor)의 개발로 경제적이며 하아드웨어 구성이 간단한 source코우딩 CODEC의 개발이 가능하게 되었다.[7]

source코우딩 방법의 대표적인 방법이 선형예측법이다. 음성파형은 인접한 샘플사이에 상호연관이 높기 때문에 현재의 음성샘플을 과거의 음성샘플들의 선형결합에 의해 예측하는 방법이 선형예측법이다.[8]

인체에서 음성의 생성과정을 보면 폐에서 나가는 공기에 의해 성대가 주기적으로 진동하거나(유성음), 구강내의 수축에 의한 난류성 공기진동(무성음)에 의한 음원이 조음기관인 성도에서 공명에 의해 음성이 생성된다. 이러한 음성 생성 과정은 성대대신 펄스 발생기, 구강내의 난류 음원대신 백색 잡음 발생기등의 음원과 성도대신 느린 시변형 디지탈 필터로 모델화할 수 있다. 음성에서 파라미터를 얻는 과정을 음성의 분석(speech analysis)이라 하며 분석과정에서 얻은 파라미터를 이용하여 적절한 구동 신호에 의해 음성을 발생시키는 과정을 음성의 합성(speech synthesis)이라 한다. 그러므로 분석과정에서추출해야 하는 파라미터는 성도에 대한 파라미터 (prediction coefficient 또는 reflection coefficient), 필터의 이득(gain), 유성음/ 무성음의 판단, 그리고 유성음의 경우 음성신호의 기본주파수이다. 음성의 합성과정에서 유성음의 경우 음성의 기본주파수의 주기를 갖는 impulse가 성도를 모델링하는 필터를 구동시키게 되는데 인간의 귀는 일반적으로 이 펄스의 주파수에 대해서 아주 민감하다고 알려져 있다.[8] 따라서 선형예측 코우딩에서 양질의 음성을 얻으려면 정확한 기본주파수의 검출이 절대적으로 필요하다.

그러므로 1939년 Dudley에 의해 처음으로 보코더가 소개된 이래로 기본주파수 검출에 대한 많은 연구가 이루어져 왔다.[9-16] 그러나 주위 상황에 관계없이 정확히 기본주파수를 검출해 낼 수 있는 알고리듬에 대한 연구는 아직도 미해결 과제로 남아 있다. 특히 기본주파수를 정확히 검출해낼 수 있는 알고리듬일수록 복잡하여, 하아드웨어 구현시 많은 로직과 연산을 필요로하기 때문에 실시간 동작 구현에는 많은 어려움이 따른다. 그러므로 Gold 알고리듬이나 Sondi의 center clipping같이 비교적 간단한 알고리듬들이 실제 LPC보코더에 채택되고 있는 실정이다[17-19] 그러나 앞에서 언급한 DSP를 사용하게 되면 DSP는 내부에 RAM과 ROM 및 하아드웨어 승산기를 내장하고 있어 시스템을 구현할 때 부수적인 회로가 거의 필요없어 하아드웨어 구성이 간단해지고 내부 프로그램의 수정이 가능하므로 소형이고 유연성있는 시스템을 구성할 수 있다.

따라서 본 논문에서는 DSP의 일종인 NEC7720을 사용하여 실시간 동작 기본주파수 검출기를 구현하였다. 기본주파수 검출 알고리듬은 동작 상태가 비교적 양호하다고 알려져 있는[15] SIFT (simplified inverse filter tracking) 알고리듬[11]을 근거로 하였다. 그러나 본래 제안된 SIFT알고리듬의 유성음/무성음 판단 및 doulbling이나 halving 에러를 제거시키기 위한 smoothing부분은 너무 간단하여 화자가 바뀔 경우 오동작을 잘하는 단점이 있다고 판단되어 이 부분을 수정, 보완하여 비교적 정확한 기본주파수 검출기를 구현할 수 있었다. 기본주파수 검출에 필요한 모든 연산은 16bit 고정숫자점 연산으로 하였으며 컴퓨터 시뮬레이션을 이용한 고정숫자점 연산으로 얻은 기본주파수와 실제 하아드웨어로 구현한 기본주파수 검출기의 결과는 실험을 통하여 거의 일치함을 확인하였다.

그러면 지금까지 제안된 많은 기본주파수검출 알고리듬 중에서 SIFT알고리듬을 본 논문에서 채택한 배경을 간단히 설명하면 다음과 같다. 지금까지 소개된 여러 알고리듬중 가장 널리 사용되고 있는 방법은 ACF (autocorrelation function)이나 AMDF (average magnitude difference function)을 이용하는 방법이다.[9-11,16] 그런데 본 논문에서는 실시간 동작 기본주파수 검출을 위하여 NEC 7720을 사용하고 있는데 NEC7720은 내부에 데이타를 저장할 수 있는 RAM이 128개로 제한되어 있다. 그러므로 ACF나 AMDF를 채택할 경우 1 frame의 크기를 128개 (8 KH_z 샘플링 경우 16msec) 이하로 제한할 수 밖에 없다. 여기에서 frame은 음성신호를 한번 처리하는 기본 단위이다. frame의 크기를 128개 이하로 제한하는 경우 정확한 기본주파수 검출이 어렵다. 왜냐하면 보통 음성의 기본주기는 3~15msec정도이므로 적당한 1frame의 크기는 frame의 overlapping을 고려하면 20~30 msec정도이기 때문이다. 따라서 ACF나 AMDF는 NEC 7720으로는 구현이 어렵다고 판단하여 음성신호를 저역필터로 통과시킨후 down샘플링하여 데이타 수를 줄여 ACF를 구하는 SIFT[11]방식을 채택하였다. 물론 TMS320 같은 off-chip addressing

능력이 있는 DSP를 사용하면 1frame의 크기는 원하는 갯수만큼 늘릴 수 있으나 본 논문에서는 여러가지의 사정상 고려하지 않았다. 여기서 한가지 부언할 것은 NEC7720을 사용할 경우 ACF나 AMDF의 연산시간이 같다는 것이다. 지금까지는 AMDF방식이 ACF방식에 비해 곱하기 연산이 필요치 않아 하아드웨어 구현시 유리하다고 알려져 있으나 NEC7720은 곱하기나 더하기 연산이 다같이 1instruction cycle이면 충분하므로 AMDF를 사용하더라도 시간상 및 하아드웨어 구성상 잇점은 더이상 없다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 하드웨어로 구현한 SIFT 알고리듬을 설명하였고 3장에서는 컴퓨터 시뮬레이션 결과를 보였다. 4장에서는 기본주파수 검출기의 설계와 제작에 대한 하아드웨어 설명을 하였고 5장에서는 기본주파수 검출을 위한 NEC7720 소프트웨어에 대한 설명을 그리고 6장에서는 결론을 제시하였다.

Ⅱ. SIFT알고리듬

음성신호에는 기본주파수와 성도의 공명주파수인 formant 주파수가 복합적으로 연관되어 있다. 특히 첫번째 formant주파수의 대역폭은 기본주파수 대역폭내에 있기 때문에 기본주파수 검출에 에러를 발생시킬 수 있다. 따라서 저역 필터와 inverse필터를 통과시켜 harmonics와 성도의 특성을 제거할 필요가 있다. 이러한 이론적인 근거에서 개발된 알고리듬이 SIFT이다. 그림1에 SIFT알고리듬의 블록선도를 도시하였고 각 블록선도에 대한 설명은 다음과 같다.

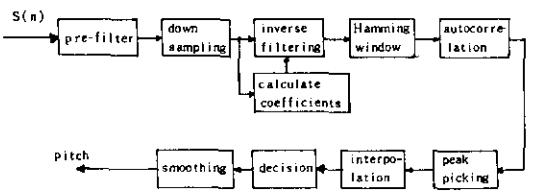

그림1. 알고리듬의 블록선도.
Fig. 1. Block diasram of the SIFT algorithm.

1) Pre-Filter

인간의 음성신호 기본주파수는 보통 66~400H_z이다. 따라서 이 기본주파수를 충분히 포함시킬 수 있는 차단주파수가 800H_z인 저역필터로 100~3400H_z 대역폭의 음성신호를 pre-filtering 하여도 기본주파수 성분은 남게 된다. pre-filter는 elliptic필터로 3차 저역필터, 차단주파수가 800H_z, pass band ripple은 0.8 dB가 되도록 설계하였다. 이때 pre-filter의 주파수 특성을 그림2에 보였고 필터의 계산은 그림3의 direct형태 필터로 하였다. 사용한 필터의 계수는 표1에 보였다. 그러나 필터의 계수가 1보다 큰 경우가 있기 때문에 고정숫자점연산이 가능하도록 각 계수를 scaling하여 1보다 작게 수정한 다음 필터 계산 결과에서 scaling factor를 곱해주는 방법을 택하였다.

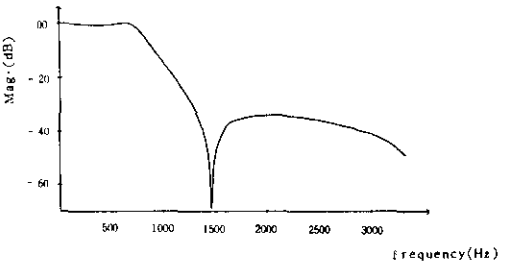

그림2. pre-filter의 주파수 특성
Fig. 2. Frequency characteristics of pre-filter

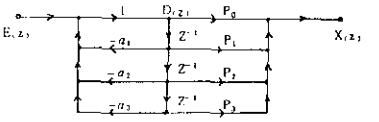

그림3. direct형태필터
Fig. 3. Direct form filter.

표 1. Pre-filter의 계수
Table 1. Pre-filter coefficients.

j	필 터 계 수		수정된 필터계수 (scaling factor 4)	
	a(j)	p(j)	a(j)	p(j)
0	1	0.03849	0.250000	0.009613
1	−2.084879	0.01789	−0.521209	0.004425
2	1.7116958	0.01789	0.427917	0.004425
3	−0.514247	0.03849	−0.128540	0.009613

2) Down 샘플링과 Interpolation

8KH_z로 샘플링된 음성신호를 차단주파수가 800H_z인 저역필터를 통과시켰으므로 Nyquist 샘플링 이론에 의해 샘플링 주파수가 2KH_z이면 되므로 4대1 down 샘플링이 가능하여 처리하여야 할 데이타량도 줄고 따라서 계산량도 줄어들게 된다. 음성신호가 down 샘플링이 되었을 경우 기본주파수의 정확도를 높이기 위해서 interpolation이 필요하다. interpolation을 그림4에 도시된 바와같이 down 샘플링된 데이타로 부터 얻은 ACF의 최대치 R(N)과 그 좌우값 R(N−1), R(N+1)로 부터 down샘플링 되지 않았을 때의 ACF

의 최대치 R(peak)를 추정하여 그에 대응하는 index를 그림5의 과정을 거쳐 기본주파수로 취하게 된다.[11]

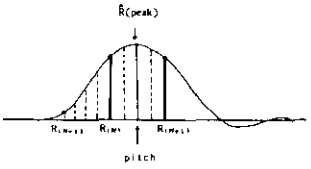

그림 4. Interpolation 함수
Fig. 4. Interpolation function.

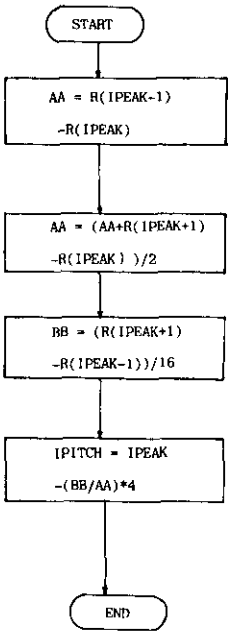

그림 5. Interpolation 연산 흐름도
Fig. 5. Flow chart for interpolation.

3) ACF의 계산과 최대값추출

ACF는 R(0)에서 최대값을 갖는다. 주기함수에서 ACF를 구할 경우 그 주기에 해당되는 lag에서도 최대값을 갖게 된다. 음성신호에서 유성음의 경우 성도의 주기적인 진동에 의해 준주기신호 (quasi-periodic signal)가 되기 때문에 기본주파수에 해당되는 lag에서 R(0)와 비슷한 큰 값을 갖게 된다. 따라서 기본주파수가 존재하는 구간에서 최대값을 구함으로써 기본주파수를 구할 수 있게 된다. 무성음의 경우는 주기가 없으므로 ACF역시 주기가 없음을 알 수가 있다. 기본주파수를 계산하기 위하여 128개의 음성신호를 한 frame으로 하였다. 그러나 급격한 기본주파수의 변화를 피하기 위해서 실제 ACF의 계산은 두 frame 즉 중복된 256개의 음성신호를 사용하였다. 그러므로 4 : 1 down 샘플링을 하면 64개의 데이타가 되고 2차 inverse 필터링하면 62개의 데이타가 된다. 샘플링 주파수가 8 KH_z인 경우 보통 기본주파수가 20~120개의 샘플간격을 갖게 되므로 4 : 1 down샘플링 되면 5~30개의 샘플간격을 갖게되어 기본주파수를 구하기 위해서는 ACF를 R(5)~R(30)만 구하면 된다. 그림6에 ACF의 최대값을 추출하는 과정을 도시하였다.

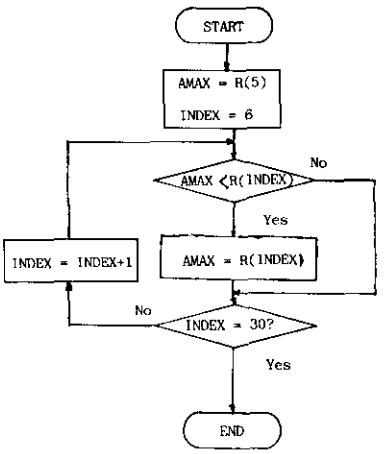

그림 6. 최대값 추출 연산 흐름도
Fig. 6. Flow chart for maximum value extraction.

4) inverse 필터링

기본주파수는 음성신호에서 구하는 것보다는 inverse 필터를 통과시킨 residual신호에서 구하는 것이 정확도가 높다고 알려져 있다.[8] 왜냐하면 음성신호에는 기본주파수 성분과 성도특성이 상호작용으로 섞여 있으므로 inverse 필터를 통과시켜 성도특성을 제거시키면 inverse필터의 출력, 즉 residual신호는 구동신호와 유사한 impulse 모양을 갖게 되어 보다 정확한 기본주파수 검출이 가능해진다. inverse 필터링을 하려면 예측계수(prediction coefficient) 또는 반사계수(reflection coefficient)가 필요한데 일반적으로 예측계수는 반사계수에 비해서 dynamic range가 크고, 반사계수는 이론상으로 −1과 1 사이의 값을 갖게되어 고정숫자점 연산이 용이하므로 반사계수를 이용하여 inverse 필터링을 하였다. 기본주파수 검출에 많은 영향을 주는 첫번째와 두번째 formant 주파수를 제거시키기 위해서는 4차 inverse 필터링이면 충분[8]하지만 4차가 되면 반사계수를 구하는 과정이 복잡하고 또한 계산시간이 많이 필요하여 실시간 동작 구현

에 어려움이 따르므로 2차 inverse 필터링을 하였고 이 결과는 컴퓨터 시뮬레이션을 통하여 4차 inverse 필터링의 결과에 비해 큰 차이가 없음을 확인 하였다. 그런데 반사계수 K_1, K_2는 ACF R(0), R(1) 및 R(2)로 부터 간단히 구할 수 있는데 그 관계식은 다음과 같다.

$$K_1 = R(1)/R(0), \quad (1)$$

$$K_2 = \frac{R(2)\cdot R(0) - R(1)^2}{R(0)^2 - R(1)^2} \quad (2)$$

이렇게 구한 반사계수 K_1, K_2를 이용하여 그림 7 의 latttice 필터로 inverse 필터를 구현하였다.

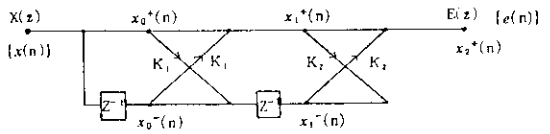

그림 7. Lattice inverse 필터

Fig. 7. Lattice inverse filter.

5) 유성음/무성음의 판단과 smoothing

음성합성시 필터의구동신호는 유성음의 경우에는 기본주파수의 주기를 갖는 impulse신호, 무성음의 경우는 백색잡음신호가 되어야하므로 유성음, 무성음의 판단이 음성해석시 필요하다. 음성신호의 유성음은 주기성을 갖기 때문에 어떤 경우 기본주파수의 두배값 또는 절반값을 찾는 경우가 발생한다. 즉 doubling에러 또는 halving에러가 발생한다. 이 경우 음성합성시 해당 frame의 합성음은 전혀 다른 소리가 되기 때문에 에러를 수정하는 smoothing은 아주 중요한 문세이다.

기본주파수값들을 비교함으로써 에러의 판단이 가능해진다. 음성신호에서 유성음의 경우는 에너지가 큰 반면 ZCR(zero crossing rate)가 작고 무성음의 경우는 반대로 에너지는 작으나 ZCR이 크다. 또한 유성음이나 무성음은 여러 frame연속이 되므로 과거 frame의 판단결과도 필요하다. 따라서 판단의 인자로 현재 frame의 에너지(ENG(2)), 직전 frame의 에너지(ENG(1)), 현재 frame의 ZCR(ZCR(2)),직전frame의 ZCR(ZCR(1)), 직전 frame의 유성음/무성음 판단결과(IVUV(3)) 등을 사용하여 현재 frame의 유성음/무성음 판단결과(IVUV(4))를 구하였다. 또한 유성음에서 무성음으로 또는 무성음에서 유성음으로 변환될 때 에너지와 ZCR의 변화가 큰것을 고려해서 에너지의 차 DENG(ENG(2)-ENG(1))와 ZCR의 차 DZCR((ZCR(1)-ZCR(2))를 사용하였다.

유성음/무성음의 판단로직에 필요한 에너지값의 문턱값인 ALϕ는 무성음일때는 매 frame마다 식(3)과 같이 변화시켜 나갔다. 만일 무성음이 연속될 경우 문턱값이 너무 작아져 버리기 때문에 무성음이 10frame 이상 연속되면 ALϕ값을 초기치 값으로 해주었다.

$$AL\phi = (ENG(2)/64) + (63/64)\cdot AL\phi, \quad (3)$$

ALϕ초기치=75(dB).

Smoothing은 연속된 무성음사이에서 유성음으로 잘못 판단하는 경우 또는 연속된 유성음사이에서 무성음으로 잘못 판단하는 경우를 수정하는데 필요하고 특히 음성신호의 주기성때문에 발생하는 doubling에러나 halving에러를 교정하기 위해서 인접한 frame의의 기본주파수 값들과 비교 수정하였다. 본 연구에서는 연속된 네 frame의 기본주파수값과 유성음/무성음의 판단 결과를 이용하여 smoothing로직을 구성하였다. 즉 현재 frame의 기본주파수값 IP(4), 유성음/무성음 판단 결과 IVUV(4)와 한 frame 직전값 IP(3), IVUV(3), 두 frame 직전값 IP(2), IVUV(2), 세 frame 직전값 IP(1), IVUV(1)을 사용하였다. 이렇게 할 경우 세 frame이 지연되지만 시간적으로는 16×3=48(ms) 지연되므로 그렇게 큰 문제는 안된다고 판단된다. 그림 8과 그림 9에 유성음/무성음 판별로직 흐름도와 smoothing 로직흐름도를 도시 하였다.

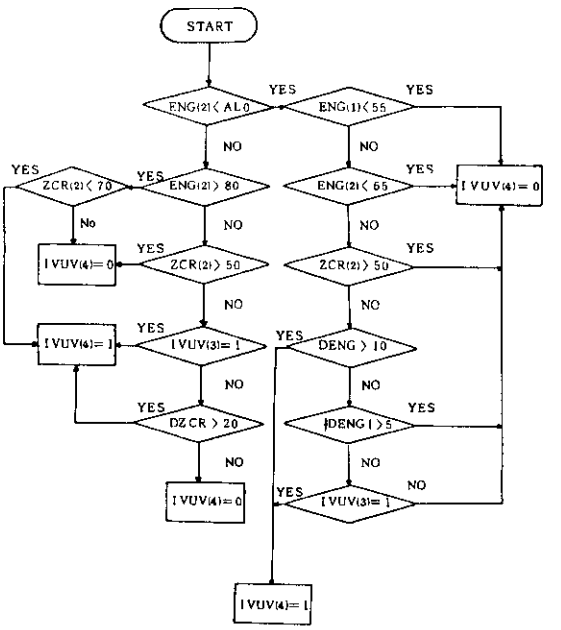

그림 8. 유성음/무성음 판별 로직 흐름도

Fig. 8. Flow chart for voice/unvoice decision.

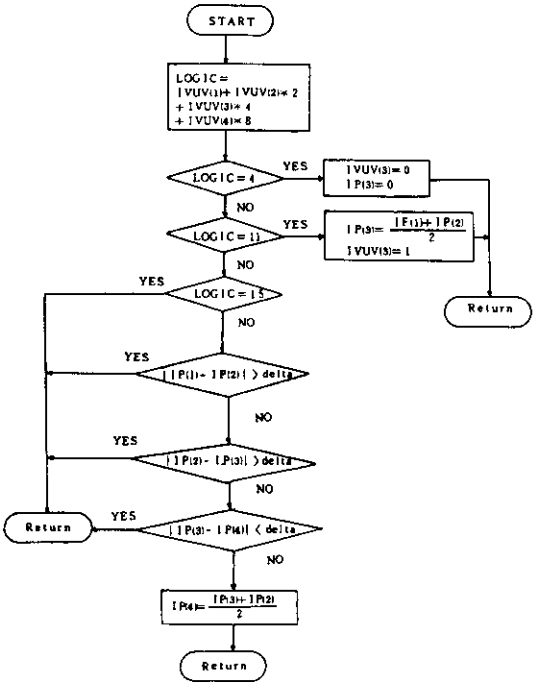

그림 9. Smoothing로직 흐름도
Fig. 9. Flow chart for smoothing.

Ⅲ. 기본주파수 검출 알고리듬의 컴퓨터 시뮬레이션

제 2 장에서 설명한 SIFT알고리듬을 PDP 11/44 16 bit 소형 컴퓨터로 유동숫자점 연산에 의해 시뮬레이션하였다. 또한 NEC 7720은 16bit 고정숫자점 연산을 하기 때문에 고정숫자점 연산으로 시뮬레이션하여 그 동작 특성을 조사하였다. 시뮬레이션에 사용된 음성신호 데이타는 A/D변환기에서 8 KH_z 샘플된 μ-law PCM 데이타를 선형변환시켜 PDP 11/44 컴퓨터의 데이타 화일로 저장하여 사용하였다. 이때 사용한 A/D변환기는 3507 CODEC chip(AMI)를 사용하였다. 정확한 기본주파수는 샘플링하여 얻은 음성신호를 256 음성샘플단위로 plotting하여 목측(eye-deteoted)에 의하여 구하였다. 목측에 의하여 기본주파수를 구하기 어려운 부분은 주로 유성음에서 무성음으로 또는 무성음에서 유성음으로 변환하는 부분인데 이런 부분은 실제 음성을 합성하여 원음과 비교함으로써 정확한 기본 주파수를 구하였다.

고정숫자점 연산을 시뮬레이션하는 방법에는 두가지 방법이 있다. 첫째 방법은 모든 값을 정수로 가정하여 연산하는 방법이다. PDP 11/44는 16bit 컴퓨터이므로 정수 연산을 할 경우 NEC 7720과 같은 정확도를 갖게 된다. 하지만 16bit 정수값은 32768 (^{15})보다 작으므로 에너지를 구하는 경우같이 계산이 불가능한 경우가 생길 수 있다. 따라서 둘째 방법은 모든 값을 1보다 작도록 한다음 16bit의 정확도를 갖도록 절삭(truncation) 해주는 방법이다. 이 경우는 모든 값이 1보다 작은 값을 갖기 때문에 곱하기 연산의 경우에도 1을 넘지않아 모든 연산이 가능해진다. 그러나 고정숫자점 연산의 경우 제한된 레지스터의 길이 때문에 더하기 연산을 할 경우 overflow가 발생할 수도 있다. overflow유무는 더하기 연산마다 일일이 판단이 어렵기 때문에 반사계수를 구할 때 사용되는 나누기 연산에서 판단하였다. 유동숫자점연산에 의한 시뮬레이션 결과는 목측된 기준기본주파수와 1~2 샘플 정도의 차이가 생기지만 음성을 합성하여 들어보았을때 거의 구별이 되지 않았다. 고정숫자점 연산에 의한 시뮬레이션 결과는 연산의 정확도가 낮기 때문에 유동숫자점 연산보다 doubling 에러가 다소 발생하였지만 이 에러는 smoothing 과정에서 수정하였다. 그림10에 시뮬레이션 결과를 도시하고 있다. 그림 10-(a)는 20대 중반 남성의 음성으로서, 사용한 음성은 "산에는 꽃피네 꽃이 피네 갈봄 여름없이 꽃이"이다. 목측에 의한 기본주파수 값(×)은 SIFT 알고리듬의 고정숫자점 연산에 의한 기본주파수 값(0)과 큰차이가 없음을 알 수 있다. 그러나 그림 10-(b)는 20대 중반 여성의 음성으로서 사용한 음성은 "눈이 부시게 푸르는 날은 그리운 사람을 그리워 하자"이다. 그림에서 알 수 있듯이 여자 음성은 남자음성에 비해 smoothing 없이는 다소 doubbing에러(+)가 발생하고 있으나 대부분의 doubling 에러는 smoothing에 의해 교정되고 있음을 알 수 있다. 대부분의 doubling 에러는 기본주파수는 연속된 frame 간의 큰 변화가 없는 성질을 이용하여 과거 frame 값을 취함으로서 교정 할 수 있다.

Ⅳ. 기본주파수 검출기의 하드웨어 구성

본 연구에서 사용한 NEC7720은 NMOS로 만들어진 16bit one chip 컴퓨터로 chip 내부에 instruction을 저장할 수 있는 instruction ROM, 계산과정에 필요한 데이타를 저장할 수 있는 데이타 ROM, accumulator 및 RAM 등이 있어 chip 자체내에서 연산이 수행되므로 하드웨어의 구성이 간단해진다. 한 instruction의 수행은 8MH_z 클럭 동작시 250ns가 필요하며 16bit×16bit 곱하기 연산도 한 instruction cycle에 수행된다. 데이타의 입출력은 직렬 또는 병렬 입출력이 가능하며 병렬의 경우 DMA 기능이 있다. 기본주파수 검출기에 NEC 7720을 사용할 경우 하드웨어의 설계는 데이타의 입출력이 주된부분이다. 그런데 NEC 7720 내부에 두 frame의 256개 데이타를 저장

그림10-a. SIFT에 의한 pitch와 목측에 의한 pitch(남성음성)

Fig. 10-a. Pitches obtained from SIFT and eye-detection (male utterance).

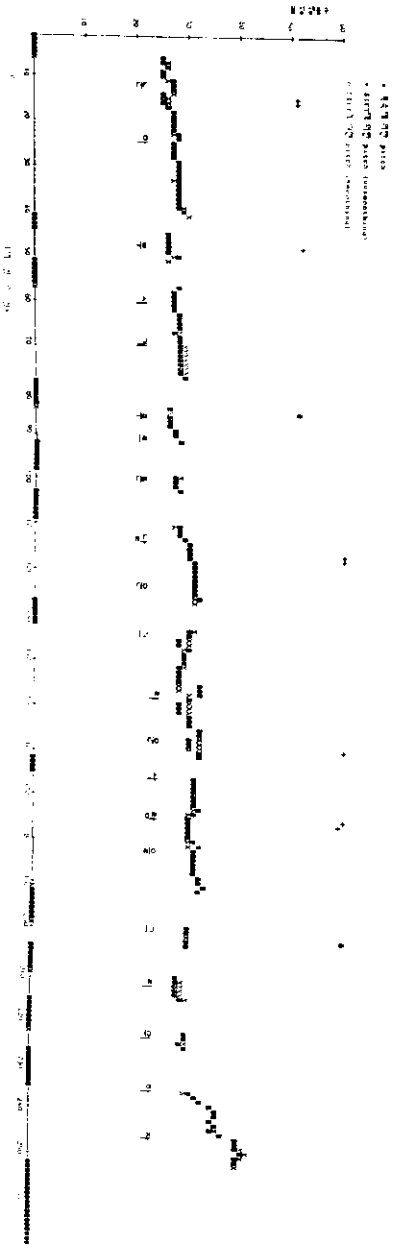

그림10-b. SIFT에 의한 pitch와 목측에 의한 pitch (여성음성)

Fig. 10-b. Pitches obtained from SIFT and eye-detection. (female utterance)

시키고 연산을 수행할 수 없기 때문에 CODEC의 출력인 음성데이타를 외부 RAM에 저장시켰다가 NEC 7720에 데이타를 보내주어야 한다. NEC 7720에서는 제한된 instruction 갯수 때문에 기본주파수 검출 알고리듬의 수행과 유성음/무성음의 판단을 수행하였고

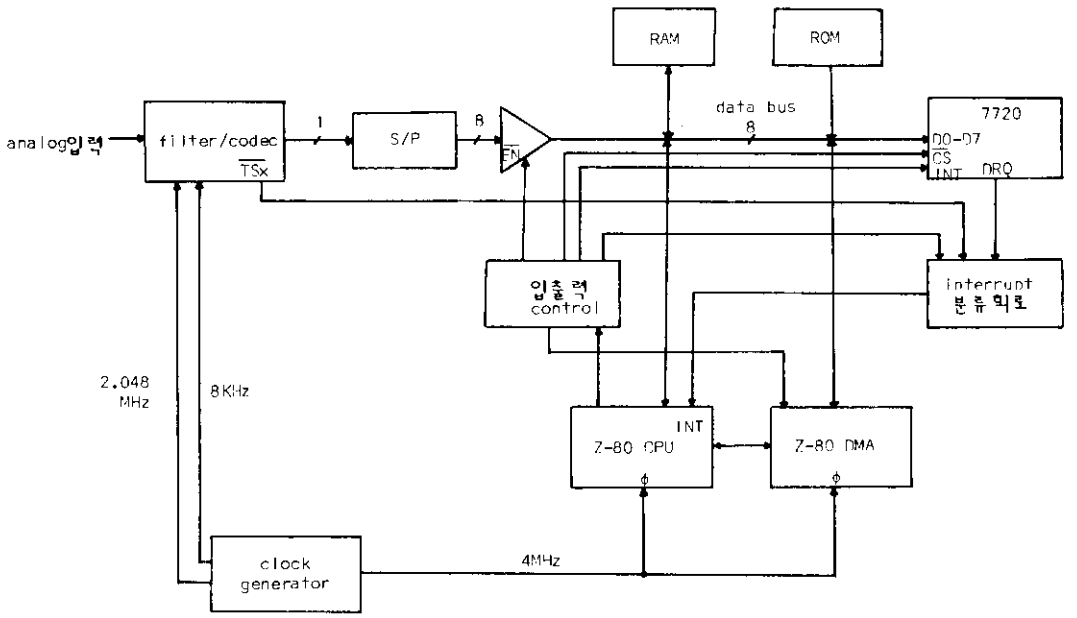

그림11. 전체시스템의 블록선도

Fig. 11. Block diagram of the pitch detection.

smoothing은 Z-80 CPU를 사용하여 수행하였다. 따라서 기본주파수 검출기를 아래와 같이 크게 3부분으로 나눌 수 있고 기본주파수 검출기의 전체 블록선도를 그림11에 도시하였다.

(1) 필터와 CODEC부

(2) NEC 7720 콘트롤부

(3) I/O 처리 및 smoothing부

각 부분에 대한 기능별 설명은 다음과 같다.

1) 필터와 CODEC부

음성신호는 저주파쪽 formant 3~4개만 있으면 충분히 좋은 음질을 유지할 수 있기 때문에 불필요한 고주파 성분과 아주 낮은 저주파 성분을 제거시키기 위해 보통 100~3400H_z의 대역필터를 통과시키고 8KH_z로 샘플링한다. 본 논문에서는 100~3400H_z의 대역필터(BPF)와 오디오 엠프를 내장하고 있는 TP 3040 (National semiconductor co.)를 사용하였다. 아날로그 음성신호는 TP3040에서 필터를 거쳐 TP3020 μ-law PCM CODEC에 입력된다. CODEC에서 8KH_z로 샘플링된 8bit μ-law PCM 데이타가 MSB부터 직렬로 출력된 다음 병렬 변환되어 RAM에 저장되었다.

2) NEC 7720 콘트롤부

기본주파수는 frame 단위로 계산하게 되므로 frame을 동기시키기 위해서 NEC 7720의 interrupt 신호를 사용하였다. CODEC의 음성데이타 128개를 RAM에 저장시켜 한 frame이 형성되면 CPU가 NEC7720에 interrupt 신호를 보내고 NEC7720의 요구에 의해 데이타를 보내 알고리듬의 연산을 수행시킨다. 모든 연산이 끝나면 다음 interrupt 신호가 올때까지 NEC 7720은 대기상태로 기다린다. NEC7720의 입출력 콘트롤은 DR레지스터를 사용하여 DMA 모드로 주고받았다. NEC7720의 IDB (internal data bus)에서 DR 레지스터에 데이타를 보내거나 받을 때는 다음과 같은 instruction을 수행한다.

MOV @DR, × 또는 MOV @×, DR

이때 NEC 7720에서는 DRQ (data request) 신호가 나오는데 이 DRQ신호는 DACK(data acknowledge) 신호가 올때까지 high 상태를 유지한다. 데이타의 입출력을 구별하기 위해서 상태레지스터의 P_0, P_1 port를 사용하였다. 즉 DRQ와 P_0는 NEC7720의 데이타 입력을 나타내어 DMA의 ready 신호로, DRQ와 P_1은 출력을 나타내어 Z-80의 interrupt 신호로 하였다. DRQ 신호에 의해 외부에서 NEC7720에 데이타를 넣어줄 때에는 DACK와 WR신호가 같이 인가되고 외부에서 NEC7720의 데이타를 읽을 때는 DACK와 RD신호가 같이 인가된다. 그림12에 NEC7720 콘트롤 회로를 그림13에 타이밍선도를 도시하였다.

3) I/O 처리 및 Smoothing부

기본주파수 검출기에서의 I/O처리 과정은 크게 다

음과 같이 분류된다.

(1) CODEC의 출력인 디지탈 음성샘플을 frame단위로 RAM에 저장시키는 과정

(2) NEC7720이 데이타를 요구할 때 두 frame의 데이타, 즉 256개의 음성샘플을 보내주는 과정

(3) 계산된 결과인 기본주파수를 읽어 smoothing하는 과정

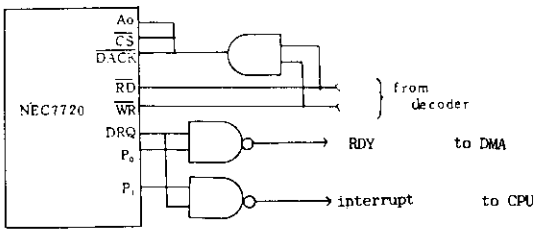

그림12. NEC 7720 입출력 콘트롤회로

Fig. 12. I/O control circuit of NEC7720.

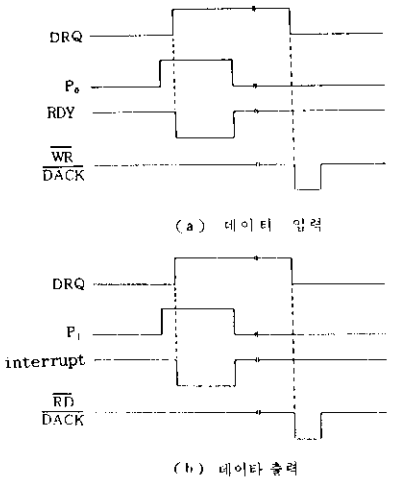

그림13. NEC 7720 데이타 입출력 타이밍 선도

Fig. 13. I/O timins chart of NEC7720.

이중 음성데이타의 저장 및 NEC7720의 출력처리는 Z-80 CPU를 사용하여 처리하였고 NEC7720으로 데이타를 보내고 과정은 CPU를 사용하는 경우 너무 속도가 느리기때문에 DMA를 사용하여 처리 하였다. 8KHz클럭으로 CPU에 interrupt를 걸어 음성신호가 샘플링될 때마다 CODEC으로부터 음성샘플을 읽어 RAM에 저장시켜 나갔다. NEC7720의 데이타 입력은 NEC7720이 연산을 위해 데이타를 요구할 때마다 Z-80 DMA에 신호를 보내 한 frame씩 중복시켜 두 frame의 데이타를 입력시켰다. 이때 NEC7720이 DMA로 부터 음성샘플을 받으면서 pre-filtering을 수행하므로 두 frame의 데이타를 다 받게 되면 64개의 샘플만 NEC7720 내부에 남게 된다. 한편 Z-80 CPU는 한 frame인 16ms동안 입출력 처리만 하므로 시간상 여유가 많기 때문에 나머지 시간동안 smoothing을 수행하였다. 그림14에 한 frame에 있어 NEC7720의 I/O타이밍선도를 도시하였다.

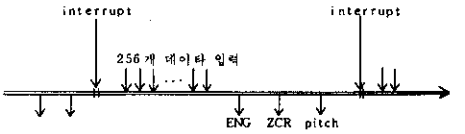

그림14. 한 frame의 데이타 입출력 타이밍 선도

Fig. 14. I/O timins chart of one frame.

4) 실험결과

이와같이 제작된 음성검출기는 다음장에서 설명하는 소프트웨어를 개발하여 real-time emulator인 EV-AKIT-7720[20]으로 그 동작여부를 시험하였고 또한 NEC7720에 직접 소프트웨어를 load하여 실시간 동작이 이루어짐을 확인하였다. 이때 사용한 입력은 목측에 의하여 기본주파수를 이미 알고 있는 음성데이타이다. 샘플링주파수 8KHz로 음성데이타를 기본주파수 검출기에 인가시키면서 실시간 동작시켜 기본주파수를 구해 RAM에 저장시킨 다음 그 결과를 고정숫자점 연산 컴퓨터 시뮬레이션 결과와 비교하여 완전 일치함을 확인하였다. 표2에 결과 일부를 도시하였다. 또한 제작된 기본주파수 검출기의 실제 사진을 그림15에 보였다.

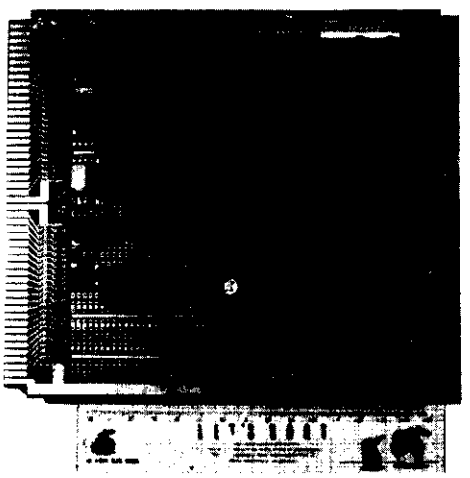

그림15. 실제 제작된 기본주파수 검출기

Fig. 15. Photosraph of the pitch detector.

표 2. 실시간 동작에 의한 기본주파수 검출기의 결과예

Table 2. Some results obtained from a real-time operation.

frame	목 측	SIFT	frame	목 측	SIFT
35	0	0	55	49	49
36	0	0	56	48	47
37	0	0	57	48	48
38	0	0	58	47	47
39	0	0	59	47	47
40	0	0	60	47	45
41	53	50	61	47	46
42	55	54	62	47	47
43	55	55	63	47	47
44	54	56	64	48	47
45	54	54	65	48	48
46	52	52	66	49	48
47	52	52	67	48	48
48	51	51	68	48	48
49	50	50	69	49	48
50	50	50	70	49	51
51	50	49	71	49	51
52	50	49	72	49	51
53	49	48	73	50	50
54	49	49	74	50	50

V. NEC7720 소프트웨어 설명

NEC7720 내부에는 23bit×512개의 instruction ROM과 13bit×512개의 데이타 ROM 및 16bit×128개의 RAM이 있다. 분석구간 한 frame은 8 KH_z 로 샘플된 128개의 데이타로 형성되므로 한 frame은 16 msec가 되어 NEC7720을 8 MH_z 클럭으로 동작시킬 경우 한 instruction은 250 nsec에 수행되므로 모든 계산이 64,000개의 instruction cycle이내에 끝나도록 프로그램되어야 실시간 동작이 가능하다. NEC7720 프로그램은 crossassembler와 non-real time simulator를 사용하여 debugging하였다. simulator는 입력 데이타 화일과 타이밍 화일을 같이 사용하면 실제 하드웨어 시스템과 같은 과정으로 프로그램이 실행된다. 모든 연산은 16bit 고정숫자점연산으로 하였고 ACF는 정확도가 높아야 하기 때문에 32bit 연산으로 하였다. NEC7720 내부에서 수행되는 기본주파수 검출 알고리듬의 흐름도를 그림16에 도시 하였다. 알고리듬의 흐름도에서 중요한 몇 부분에 대해 설명하고저 한다.

1) μ-law PCM/선형 PCM 변환

본 시스템은 A/D변환을 위해 TP3020 μ-law PCM CODECchip을 사용하고 있으므로 μ-law PCM데이타를 선형 PCM 데이타로 바꾸어야 한다. 변환 알고리듬은 CCITT권고안 G.711에 따랐다.[11] 그런데 이와같은 과정을 NEC7720에서 수행하려면 계산시간이 많이 소요되므로 미리 μ-law PCM값을 선형 변환시켜 NEC7720 데이타 ROM에 저장시켜 놓고 μ-law PCM 값에 대응하는 선형PCM 값을 찾아내는 look-up table방식을 사용하였다. NEC7720 데이타 ROM은 13 bit로 되어 있으므로 선형 PCM 데이타의 크기만을 저장시키고 μ-law PCM 데이타로부터 부호 비트를 복원시켜 14bit 선형 데이타를 얻었다.

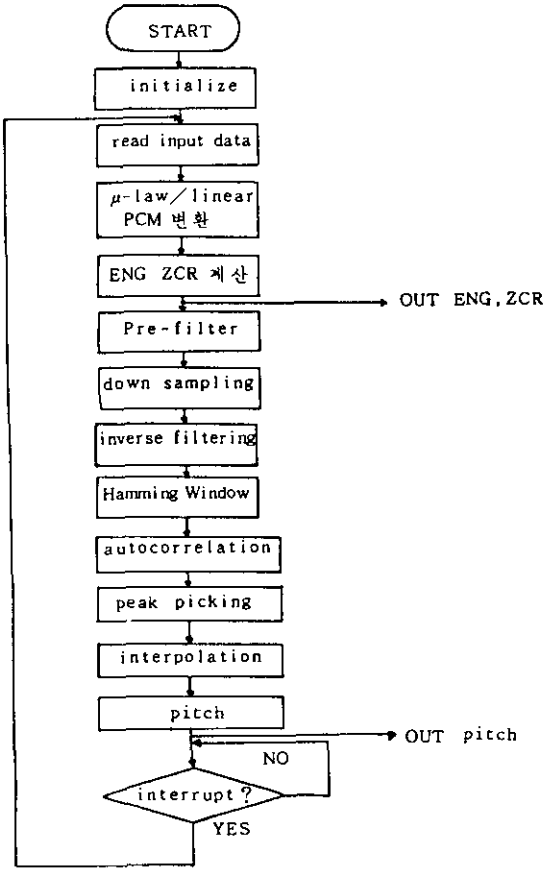

그림16. 기본주파수 검출 알고리즘의 흐름도

Fig. 16. Flow chart for pitch detection algorithm.

2) Pre-Filtering과 Down 샘플링

Pre-filtering은 그림3과 같은 direct 형태 필터를 사용할 경우 식(4)로 표시된다.

$$y(n)=a_1y(n-1)+a_2\cdot y(n-2)+a_3y(n-3)+p_0x(n)+p_1x(n-1)+p_2x(n-2)+p_3x(n-3) \quad (4)$$

DMA로부터 음성샘플을 한개씩 읽으면서 선형변환을 하고 pre-filter 출력을 네개마다 한개씩 취함으로써 down 샘플링을 하였다.

3) 반사계수 계산과 inverse 필터링

식(3)의 k_1, k_2를 이용하여 2차 inverse필터링을 하므로 ACF R(0), R(1), R(2)를 먼저 구해야 한다. 그러나 고정숫자점 연산으로 인해서 반사계수가 1보다 큰 값이 생길 수 있는데 이때는 모든 연산을 중지하고 기본주파수 대신 FF값을 내보내주도록 하였다. 연산의 정확도를 높이기 위해서 ACF의 연산은 32bit로 하였다. 그림17에 알고리듬의 흐름도를 보였다.

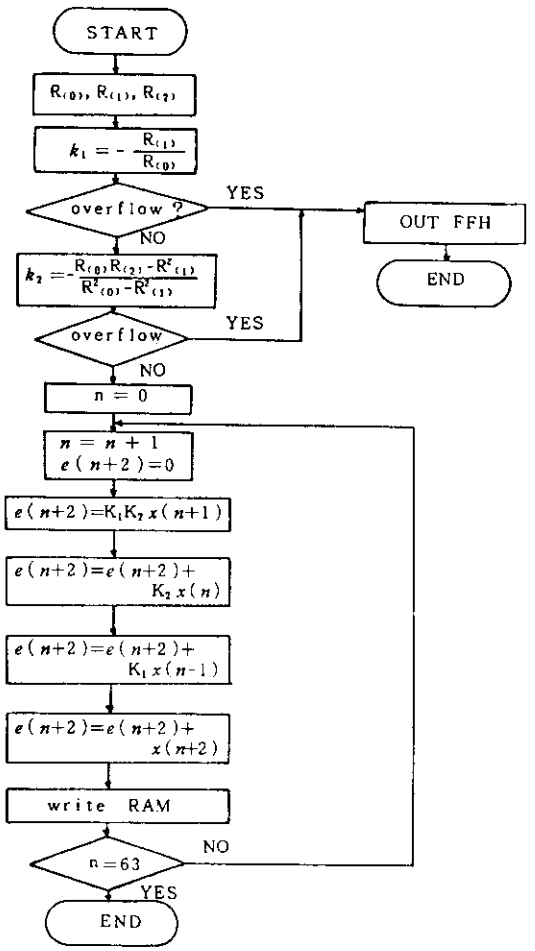

그림17. Inverse 필터링 알고리듬의 흐름도

Fig. 17. Flow chart for inverse filtering.

4) 윈도우와 ACF의 계산

Hamming 윈도우 값을 유동숫자점 연산에 의해 계산을 하여 13bit로 절삭(truncation)한 뒤 데이타 ROM에 저장시켰다. Hamming 윈도우는 다음과 같이 식(5)로 표현된다.

$$W(n) = 0.54 - 0.46\cos\{(2\pi n)/61.\} \qquad 0 \le n \le 61 \qquad (5)$$

NEC 7720 ROM pointer는 1씩 감소만 되고 윈도우 함수는 좌우 대칭이므로 마지막 ROM값을 갖는 주소를 지정해 놓고 데이타에 윈도우 값을 곱하고 주소를 하나씩 감소시켰다. ACF는 정확도를 높이기 위해 32bit 연산을 하였으나 최종 결과에서 16bit로 절삭하여 사용하였다.

5) 나누기 연산의 알고리듬

NEC 7720에는 나누기 instruction이 없기 때문에 restoring 알고리듬에 의해 16bit÷16bit 연산을 하였다. 젯수가 피젯수보다 작으면 나누기 결과가 1보다 커져 overflow가 일어난다. 나누기는 subprogram으로 하였고 총56개의 instruction으로 16bit÷16bit의 경우 총 264 instruction cycle이 필요하여 66μs가 소요된다.

표 3. Instruction ROM 사용표
Table 3. Instruction ROM.

용 도	instruction수	instruction cycle수
initialize	41	56
linear conversion	20	
ENG, ZCR	20	
pre-filter	24	
down-sampling	20	29454
inverse filter	109	6377
window	25	1226
autocorrelation	53	22968
peak picking	19	226
interpolation	36	290
d ivision	56	264 (*)
v/uv decision	61	48
기 타	22	16
합 계	506 (98.8%)	60661 (94.8%)

(*) 다른 과정에 포함되므로 합계에서 제외

6) instruction ROM과 데이타 ROM 사용표

기본주파수 검출 연산에 소요된 instruction수는 총 506개로 instruction ROM의 98.8%를 사용하였고 instruction cycle은 64,000개중 60,661개가 소요되었다. 표3에서 알 수 있듯이 pre-filtering과 ACF 계산에 80% 이상의 시간이 소요되었다. 데이타 ROM은 μ-law PCM 변환 테이블과 윈도우 값의 저장을 위해 사용하였고(표4), ACF계산과 나누기 연산과정

표 4. 데이타 ROM 사용표
Table 4. Data ROM instruction.

용 도	갯 수
μ-law/PCM 변환 테이블	128
Hamming 윈도우 값	62
합 계	190(37%)

은 subprogram으로 하였다.

Ⅵ. 結 論

본 논문에서는 LPC보코더 개발을 위해서는 필수적인 실시간 동작 음성 기본주파수 검출기를 고속 신호처리 프로세서인 NEC7720을 사용하여 구현하였다.

DSP를 사용하므로써 SIFT알고리듬에 필요한 대부분의 연산을 소프트웨어로 처리할 수 있어 하드웨어의 복잡성을 줄임은 물론 시스템의 유연성을 강조할 수 있었다. 설계 제작된 검출기는 목측에 의해 기본주파수를 이미 알고있는 음성데이타를 사용하여 그 동작 여부를 확인하였다. 수정된 SIFT 알고리듬을 수행하는데 1frame을 128개의 샘플(16msec) 로 하였을 때 총 15.2msec가 소요되었으며 instruction ROM의 98.8%, 데이타 ROM의 37% 및 내부 RAM의 94%를 사용하였다. 그러나 디지탈 pre-filter대신 아날로그 필터를 사용할 경우 많은 연산 시간의 감소가 있을 것이다.

여기서 한가지 부언할 것은 본 SIFT 검출기는 기존의 검출기 보다 좀더 복잡한 알고리듬을 채택하여 NEC7720으로 실시간 구현하였다는데 그 의의가 있을 것이며 robust한 실시간 동작 기본주파수 검출기 개발은 앞으로 계속되어야 할 것이다.

끝으로 본 검출기는 LPC 보코더는 물론 APC 및 TDHS(time domain Harmonic Scaling)[23] CODEC 등에 응용될 수 있으며 또한 소프트웨어의 수정만으로 음성해석, 음성인식 등을 위한 실시간 하드웨어 개발에 직접 이용될 수 있을 것으로 믿는다.

參 考 文 獻

[1] J.L. Flanagen et. al., *Speech Coding.* IEEE Trans. Comm., COM-27, pp. 710-736, April, 1979.

[2] L.R. Rabiner and R.W. Schafter, *Digital Processing of Speech Signals,* Prentice-Hall Inc., 1978.

[3] Joint Special Issue on Integrated Circuits for Speech, *IEEE Trans. ASSP,* vol. ASSP-31, Part II, February 1983.

[4] K. Irie, et. al., "A Single-Chip ADM LSI Codec," *IEEE Trans. ASSP,* vol. ASSP-31, Part II, pp. 281-287, February, 1983.

[5] J.B. O'Neal, et. al., "The Design of an ADPCM/TASI System for PCM Speech Compression," *IEEE Trans. Comm.,* vol. COM-29, pp. 1393-1398, Sep. 1981.

[6] K.S. Lee, *A Study on the Real-time Implementation of ADPCM CODEC for Multichannel Using DSP.* M.S. Thesis, S.N.U., Feb. 1985.

[7] R. Brodersen, "VLSI for Signal Processing,' Trends and Perspective in Signal Processing, vol. 1, pp. 7-11, January 1981.

[8] J.D. Markel and A.H. Gray, *Linear Prediction of Speech,* Springer Verlag, 1976.

[9] M.M. Sondhi, *New Methods of Pitch Extraction.* IEEE Trans. Audio and Electro-acoust. June 1968.

[10] M.J. Ross, H.L. Shaffer, A. Cohen, R. Freudberg, and H.J. Maley, *Average Magnitude Difference Function Pitch Extractor.* IEEE Trans. ASSP pp. 353-362.

[11] J.D. Markel, *The SIFT Algorithm for Fundamental Frequency Estimation.* IEEE Trans. Audio and Electro. pp. 367-377 Dec. 1972.

[12] B. Gold and L.R. Rabiner, *Parallel Processing Techniques for Estimating Pitch Period of Speech in the time domain.* Jour. A.S.A., Aug. 1969.

[13] Neil J. Miller, "Pitch Detection by data Reduction," *IEEE Trans.* vol. ASSP-23, pp. 72-79 Feb. 1975.

[14] A.M. Noll, "Cepstrum pitch determination," *J. A.S.A.,* vol. 41, pp. 293-309, Feb. 1967.

[15] L.R. Rabiner, M.J. Cheng, A.E. Rosenberg, C.A. McGonesal, "A comparative performance study of several Pitch Detection Algorithm," IEEE Trans. ASSP pp. 196-215 Oct.1976.

[16] Chong Kwan Un, Shin-Chien Yang, "A pitch Extraction Algorithm Based on LPC inverse filtering and AMDF," *IEEE Trans. ASSP.* vol. ASSP-25, pp. 565-572, Dec. 1977.

[17] J. Dubnowski, R.W. Schafter and L.R. Rabiner," Real Time Digital Hardware

Pitch Detector," *IEEE Trans. ASSP*, vol. ASSP-24, pp. 2-8 Feb. 1976.

[18] J.A. Feldman and et. al., "A Compact, Flexible LPC Vocoder Based on a Commercial Signal Processing Micro Computer,' *IEEE Trans. ASSP*, vol. ASSP-31, pp. 252-258, Feb., 1983.

[19] H. Hassanein and B. Bryden, "Implementation of the Gold-Rabiner Pitch Detector," *IEEE Trans. ASSP*, vol. ASSP-33, pp. 319-320, Feb. 1985.

[20] N.E.C. Microcomputers, uPD 7720 Signal Processing Interface(SPI) User's Manual.

[21] CCITT Yellow Book, vol. III-Fascimile III.3 Digital Networks-Transmission System and Multiplexing Equipment, Genera, Nov. 1980.

[22] D. Malah, "Time-Domain Algorithm for Harmonic Bandwidth Reduction and Time Scaling of Speech Signals," *IEEE Trans. ASSP*, vol. ASSP-27, pp. 121-133, Apr. 1979.