사운드 합성을 위한 DSP의 설계 및 검증

Design and Verification of Sound Synthesis DSP

장 호 근*, 권 민 도**, 박 주 성*
(Ho Keun Jang*, Min Do Kwon**, Ju Sung Park*)

※이 연구는 1995~1996년 정보통신부 핵심기술개발사업의 연구비 지원에 의해 부산대학교 정보통신 연구소 주관으로 이루어졌습니다.

요 약

이 논문은 사운드 합성을 위한 전용 DSP의 설계에 관한 내용이다. 설계된 음원 DSP는 사운드 카드나 전자 악기, 혹은 노래방 기기 등에서 미디 신호를 입력으로 받아서 사운드를 합성하는데 사용된다. 합성 알고리즘으로는 FM 방식과 PCM 방식을 지원하며, 구조가 다르고 합성 방식이 서로 다른 8개의 알고리즘을 하드웨어적으로 구현하였다. 설계된 DSP는 44.1KHz의 16비트 출력으로 32개의 음을 동시에 낼 수 있다. DSP 내부 구조의 최적화와 마이크로 동작의 병렬화를 통해 실제 필요한 주파수보다 낮은 주파수에서 동작시킴으로써 전력 소모와 칩 구현에서 많은 이점을 가져올 수 있었다. 설계된 DSP는 COMPASS 툴에서 0.8μm 표준 셀로 합성되어 칩으로 제작되었으며, 동작 주파수는 33MHz이다. 제작된 칩을 검증하기 위해 PC에 삽입되는 음원 모듈 카드를 제작하여 미디 음악을 연주시켜 보았다. 그 결과 원하는 동작 주파수에서 완벽하게 사운드를 합성해내는 것을 확인할 수 있었다.

ABSTRACT

This paper presents a design of sound synthesis IC. The IC is used for synthesizing musical sound with MIDI signal in a sound card or an electronic musical instrument or a karaoke system. The IC integrated a microprocessor which is used for MIDI controller or system controller and a sound DSP into one chip. The sound DSP implements two methods of sound synthesis algorithms, FM and PCM synthesis. It can synthesize 32 voices simultaneously at 44.1KHz sampling rate and the output data is 16 bit width for high quality audio. The architecture and micro-operations are designed for parallel execution of several micro-operations in a clock cycle. This VLIW-like architecture results in lower clock frequency, which leads to lower power consumption and easier hardware implementation. The IC was synthesized with 0.8μm standard cell library in COMPASS tool. The IC was tested on a MIDI sound module card on a PC. Through the test it was verified that the IC synthesized sound perfectly.

I. 서 론

현대의 전자음악 기기는 컴퓨터와 반도체 기술의 발달로 고속의 VLSI 칩을 사용함으로써, PC에 삽입되는 작은 카드 형태의 음원 모듈만으로도 오케스트라 수준의 음악을 연주하는 것이 가능하게 되었다. 이에 따라 컴퓨터를 이용한 사운드 합성은 사운드 카드 및 노래방 기기 등을 통해 이제 우리 주변에서 흔히 접할 수 있는 매체로 자리 잡고 있으며, 멀티미디어 시대의 필수적인 요소로 인식되고 있다. 그럼에도 불구하고 우리 나라에서는 컴퓨터 음악에 대한 본격적인 연구 환경이 조성되어 있지 않아 하드웨어 설계 수준이 매우 열악한 형편이며, 사운드 합성 시스템의 핵심 부품들을 거의 외국에서 수입해 쓰고 있는 실정이다. 사운드 합성 시스템의 개발은 사운드 합성을 위한 디지털 신호 처리 알고리즘의 개발, 이를 실시간으로 처리할 수 있는 DSP(Digital Signal Processor)의 설계, 사운드의 원음을 샘플링하고 알고리즘에 필요한 각종 파라메터의 추출과 원음 파형의 적절한 가공을 통한 악기음 샘플 데이터의 데이터 베이스화, 그리고 시스템 제어를 위한 프로그램의 개발 등으로 이루어진다. 따라서 고성능의 하드웨어 설계 기술과 여러 가지 복잡한 소프트웨어 기술이 결합되어야만 상품성 있는 사운드 합성 시스템을 구현할 수 있다고 볼 수 있다.

그림 1은 전자 음악 기기 간의 공통 신호 포맷인 미디(MIDI : Musical Instrument Digital Interface)[1] 신호에 기

*부산대학교 전자공학과
**LG 반도체

접수일자 : 1997년 12월 30일

반을 둔 간단한 사운드 합성 시스템의 구조를 보여주고 있다.

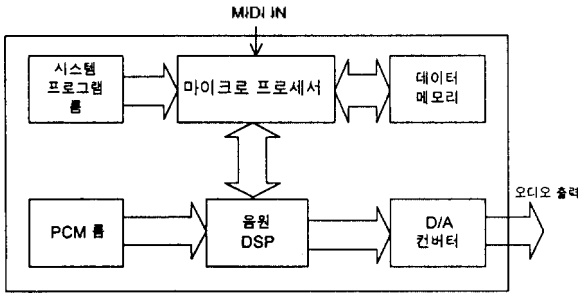


그림 1. 사운드 합성 시스템의 구조
Fig. 1 Sound synthesis system.

여기에서 마이크로프로세서는 전체 시스템의 컨트롤러로 사용되는데, 미디 신호를 입력받아 이를 해독하고 사운드를 합성하기 위한 각종 파라메터를 결정해서 사운드 합성 DSP로 전달함으로써 DSP의 합성 동작을 제어하는 역할을 수행한다. 사운드 합성 DSP(음원 DSP)는 마이크로프로세서로부터 합성에 필요한 파라메터를 넘겨받아, 해당 사운드를 합성하기 위한 알고리즘을 실행시켜 외부 D/A(Digital to Analog) 컨버터로 보내질 샘플 데이터를 만들어 낸다. 이 때 내부 파형 발진기나 외부 메모리에 저장되어 있는 실제 악기음의 샘플 데이터를 이용하게 된다. 음원 DSP에서 발생된 오디오 신호는 D/A 컨버터를 통해 아날로그 신호로 변환되어 스피커를 통해 출력된다.

이 논문은 사운드 합성을 위한 DSP의 설계에 관한 것이다. 설계된 DSP는 44.1KHz의 샘플링 주파수로 32개의 음을 동시에 합성해 낼 수 있다. 이를 위한 DSP의 내부 구조는 연산 기능의 병렬화를 통해 한 개 이상의 기능 동작이 동시에 한 사이클 내에 처리되도록 하였다. 이에 따라 DSP의 동작 주파수를 최대한 낮춤으로써 전력 소모를 적게 하고, 칩 구현에 있어 낮은 수준의 공정 기술을 사용하는 것이 가능하였다. 합성 알고리즘은 FM(Frequency Modulation) 방식과 PCM(Pulse Code Modulation) 방식을 사용한다. 그리고, 설계된 DSP는 4개의 출력 채널을 가지고 있는데, 이것은 각기 다른 음을 내는 2개의 스테레오 채널로 구성 되어 있다. 한편으로는 각 악기음이 출력되는 채널을 네 개 중 한 개의 채널로 프로그래밍 할 수 있어 입체감 있는 사운드를 들을 수 있다. 출력 오디오 샘플 데이터는 16비트로 96dB의 SNR을 실현시키고, 최대 64MB의 사운드 샘플 데이터 메모리를 액세스 할 수 있다.

실제 구현된 IC는 시스템 제어로 쓰이는 마이크로프로세서와 DSP를 한 칩에 집적시킴으로써, 이를 적용한 시스템의 저가격화와 소형화를 가능하게 하였다. 마이크로프로세서는 인텔의 8비트 마이크로컨트롤러인 8051을 사용하여 상용의 개발 툴로써 쉽게 프로그램을 개발 할 수 있다. 설계된 칩은 COMPASS의 0.8μm CMOS 표준 셀을 이용해 구현되었고, 동작 주파수는 33MHz이다.

Ⅱ. 사운드 합성 알고리즘

사운드를 합성하기 위한 알고리즘으로는 가산 합성 방식[2], 감산 합성 방식[4], 비선형 합성 방식[4], FM 합성 방식[4], 그리고 웨이브 테이블 합성 방식이라 불리는 PCM 합성 방식[1]등 여러 가지가 있다. 이 중에서 현재 상용 전자 음악 기기에서 가장 널리 사용되고 있는 것으로서 FM 방식과 PCM 방식을 들 수 있다.

FM 방식은 무선 통신에서 사용되는 주파수 변조 방식을 사운드 합성에 적용한 것으로 변조 지수, 변조 주파수 등을 시간에 따라 변화시킴으로써 부드럽고 동적 스펙트럼이 풍부한 음을 얻을 수 있다. FM 방식의 알고리즘은 좀 더 풍부한 하모닉 성분을 얻기 위해 여러 개의 정현파를 복잡한 구조로 연결함으로써 기존의 악기로 낼 수 없는 다양한 음을 발생 시킨다. 그러나 피아노 음이나 타악기 음과 같이 음의 시작과 끝 부분에서 복잡한 하모닉 성분을 가지는 악기음에 대해서는 제대로 합성해 내지 못하는 단점이 있다. 그림 2는 설계된 DSP에서 사용하는 FM 합성 알고리즘 중 하나를 나타내고 있다[5]. 그림에 나타난 바와 같이 이 알고리즘은 두 개의 FM 신호를 더한 것으로, 두 개의 캐리어 주파수를 사용하여 두 개의 포만트 주파수 성분에 대한 주변 주파수 성분을 표현하고 있다. 출력값은 다음과 같은 식으로 표시된다.

$$FM = A0(t)\sin(2\pi fc0(t) + A1(t)\sin(2\pi fm1(t))) + A2(t)\sin(2\pi fc2(t) + A3(t)\sin(2\pi fm3(t))) \quad (1)$$

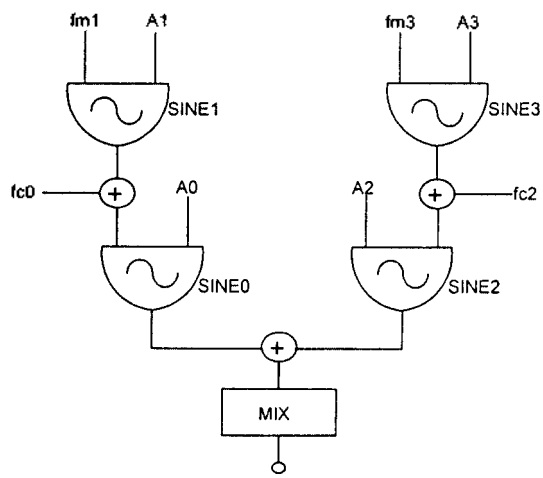


그림 2. FM 합성 알고리즘의 예
Fig. 2 FM synthesis algorithm.

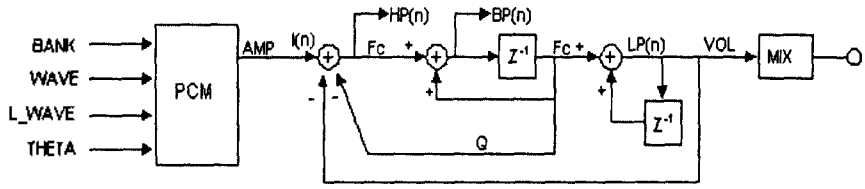


그림 3. PCM 알고리즘의 예
Fig. 3 PCM synthesis algorithm.

PCM 방식은 실제 악기의 원음을 샘플링 하여 메모리에 저장시켜 놓고, 이를 바탕으로 사운드를 합성해 내는 방법이다. 이 방식은 실제 악기음에 매우 근접한 사운드를 만들어 낼 수 있는 반면, 실제음에 더욱 가까운 음을 만들어 내기 위해서는 메모리가 많이 든다는 단점이 있다. 실제로 모든 종류의 악기에 대해 모든 음정에 해당하는 음을 샘플링 하는 것은 불가능한 일이다. 따라서 한 악기에서 대표음을 선정해 이것만을 샘플링 해서 메모리에 저장시켜 놓고, 다른 음정에 해당하는 음을 만들어 내게 된다. 이 과정에서 인터폴레이션, 데시메이션, 필터링 등의 디지털 신호 처리 기법이 이용된다. 또한 하나의 음을 샘플링 하는 경우에도 그 음의 처음부터 끝까지를 모두 샘플링 해서 저장하는 것이 아니라, 파형의 규칙성이 보이는 곳에서 몇 주기 정도의 파형에 대한 샘플 값만을 저장하고, 합성 시에는 이것을 반복적으로 읽어와 합성 함으로써 메모리 용량을 줄일 수 있다[11]. 그림 3에 설계된 DSP에서 사용된 PCM 알고리즘의 예를 보였다. 이 알고리즘은 주기 부분의 파형 반복 알고리즘과 -12dB lowpass 필터를 사용한다. 그림 3에서 BANK, WAVE, 그리고 THETA는 PCM 메모리의 악기음 샘플 데이터 번지를 나타내고, L_WAVE는 반복 구간에 해당하는 번지를 나타낸다. Lowpass 필터로 사용된 필터는 상태 변수 필터[10]로서, 이 필터는 그림에서 보는 바와 같이 하나의 필터에서 lowpass, highpass, bandpass 출력을 동시에 얻을 수 있고, 컷오프 주파수 Fc와 공진 주파수 Q를 독립적으로 제어할 수 있는 장점이 있다. 각 필터의 출력식은 다음 식으로 표현된다.

$$LP(n) = LP(n-1) + Fc * BP(n-1) \quad (2)$$

$$HP(n) = I(n) - LP(n) - Q * BP(n-1) \quad (3)$$

$$BP(n) = Fc * HP(n) + BP(n-1) \quad (4)$$

여기서 $LP(n)$, $HP(n)$, $BP(n)$이 각각 lowpass, highpass, bandpass 출력을 나타낸다. 필터의 입력 신호는 원래 악기음 샘플 데이터로부터 DTHETA만큼 변위를 가지고 읽어 온 것으로, 여기서 DTHETA는 샘플링된 악기음의 음정에서 다른 음정으로 전환할 때 피치 주파수를 결정하는 파라메터가 된다. 샘플 데이터는 필터의 입력이 되고, Q와 Fc 값을 조정해 줌으로써 원하는 악기음의 스펙트럼을 얻을 수 있다.

표 1은 설계된 DSP에서 사용된 합성 알고리즘의 종류와 특징을 나타내고 있다.

Ⅲ. DSP 구조 및 로직 설계

3.1 동작 개요

DSP가 44.1KHz의 샘플링 주파수로 32개의 음을 동시에 내기 위해서는 한 음에 대한 샘플 데이터가 0.71μs 안에 만들어져야 한다. 이 시간을 하나의 샘플 데이터가 만들어지는 합성 슬롯(Synthesis Slot)이라고 부른다. DSP의 동작 주기는 32개의 합성 슬롯으로 이루어지는 합성 프레임(Synthesis Frame)에 맞추어져 있다. 이것은 32개의 음에 대한 샘플 데이터가 모두 만들어지는 시간으로서 외부 D/A 컨버터로 출력되는 주기(1/44.1k = 23μs)와 같다. 합성 슬롯 동안에는 해당 음의 샘플 값을 만들어 내기 위해 지정된 알고리즘에 따라 많은 DSP의 마이크로 동작들이 실행된다. 여기에서 만들어진 샘플 데이터는 외부로 출력되기 전 DSP 내부의 누적기에 이전 합성 슬롯의 샘플 값들과 더해져 저장된다. 하나의 합성 프레임이 끝나면 누적기에 저장된 32개의 음에 대한 샘플 값이 쉬프트 레지스터로 옮겨지고, 그 다음 합성 프레임 동안

표 1. 음원 DSP에서 사용된 알고리즘 종류
Table 1. Synthesis algorithms of sound DSP.

알고리즘 번호	알고리즘 명	특 징
0	FM1	2 Modulator 사용, 2개의 FM 신호를 더한 것
1	FM2	2 Modulator 사용, 1개의 모듈레이터에 대해 feedback
2	PCM1	Wave looping 방식, 선형 인터폴레이션, 6dB IIR Lowpass 필터
3	PCM2	Wave looping 방식, 12dB 상태 변수 필터
4	PCM3	Sampling, no wavelooping, 타악기 합성 알고리즘
5	PCM4	Sampling, 선형 인터폴레이션
6	PCM5	Wave looping 방식, 선형 인터폴레이션, 6dB IIR Lowpass 필터
7	PCM6	삼각파 +Sampling, 12dB 상태 변수 필터

한 비트씩 외부 D/A 컨버터로 전달된다.

합성 알고리즘은 DSP의 마이크로 동작에 의해 구현되는데, 모든 합성 알고리즘은 일정한 갯수의 클럭 사이클 안에 끝나게 된다. 한 사이클마다 수행되어야 할 동작들은 마이크로코드화 되어 DSP 내부의 마이크로 롬에 저장되어 있다. 설계된 DSP는 8개의 합성 알고리즘을 내장하고 있으며, 각각은 서로 다른 구조의 FM 합성과 서로 다른 방식의 PCM 합성 방식을 구현한다(표 1).

DSP 내부에는 마이크로프로세서로부터 전달되는 합성 파라메터들을 저장하는 램이 있다. 이것은 파라메터 램(PRAM)으로 불린다. PRAM은 동일한 갯수의 어드레스 영역을 가지는 32개의 세그먼트로 나누어져 있다. 각 세그먼트는 하나의 사운드를 합성하는데 필요한 여러 가지 파라메터들이 저장된다. 따라서 하나의 세그먼트는 한 합성 슬롯 동안 하나의 알고리즘이 수행되면서 데이터를 액세스 하기 위한 메모리 영역으로 볼 수 있다. 한 음을 만들어 내기 위한 많은 파라메터 중에는 그 음을 합성하는데 사용되는 알고리즘의 번호를 가진 파라메터가 있다. 이상을 종합해 볼 때 합성 프레임 동안 32개의 음을 합성하기 위한 DSP의 동작은 다음과 같다. PRAM의 각 세그먼트는 합성 슬롯 번호를 나타내는 슬롯 카운터에 의해 차례로 액세스 된다. 어떤 합성 슬롯이 시작될 때 가장 먼저 일어나는 동작은 PRAM의 해당 세그먼트에 저장된 알고리즘 코드 번호를 읽어와서 알고리즘 번호 레지스터에 실어 주는 것이다. 이것은 어떤 알고리즘 코드가 저장된 마이크로 롬의 시작 번지를 나타낸다. 그 다음으로는 마이크로 롬의 프로그램 카운터가 하나씩 증가하면서 합성 알고리즘에 대한 마이크로 동작들이 순차적으로 실행된다. 이 때 필요한 파라메터 값은 슬롯 카운터에 의해 지정된 PRAM의 일정 영역만을 액세스 하게 된다. 최종적으로 만들어진 샘플 값은 누적기에 저장되고, 슬롯 카운터가 1 증가하면서 다음 합성 슬롯으로 넘어 가게 된다. 32개의 합성 슬롯이 끝나면 누적기에 저장된 32개 음에 해당하는 샘플 값은 쉬프트 레지스터로 옮겨지고, 그 다음 합성 프레임 동안 한 비트 씩 외부 D/A 컨버터로 출력된다.

3.2 합성 알고리즘의 마이크로 동작 구현

사운드 합성 알고리즘은 크게 덧셈, 곱셈, 내·외부 파형의 샘플 값 발생 등으로 이루어지며, 이것이 음원 DSP와 마이크로 동작의 근간이 된다. 설계된 DSP는 각 연산 동작과 기능에 관계되는 특정한 레지스터들을 사용하여, 이 레지스터들에 값을 실어주기만 하면 해당 동작이 수행되도록 설계되었다. 알고리즘 수행을 위한 레지스터는 9 가지 종류가 있으며, 각각의 비트수와 기능은 다르다(표 2). 이에 따라 DSP의 마이크로 동작은 데이터 버스에 실린 값을 내부의 레지스터로 저장하는 것이 대부분이다. 내부 데이터 버스에 실리는 값은 파라메터 메모리로, 덧셈기 혹은 곱셈기의 연산 결과가 된다.

표 2. 음원 DSP의 레지스터 종류와 기능
Table 2. Registers in sound DSP.

레지스터 종류	크기 (비트수)	기 능
AR	20	덧셈기의 입력으로 사용되며, wave looping과 엔빌로프 계산시 특정한 값을 저장함으로써 덧셈의 결과값을 제어
BR	20	덧셈기의 입력으로 사용
SAHR	9	샘플데이터의 상위 어드레스를 내보내며, 여러 종류의 파형 중 어느 하나를 선택하는 기능을 수행
SALR	12	샘플데이터의 하위 어드레스를 발생시킨다.
XR	12	파형 발생기의 출력 샘플 값을 받아들여 곱셈기의 입력으로 사용
YR	12	곱셈기의 입력값을 저장
MIXR	3	오른쪽 채널의 MIX 값 설정
MIXL	3	왼쪽 채널의 MIX 값 설정
ACC	24	출력 샘플데이터 값 저장

그림 4와 그림 5는 그림 2와 그림 3에 보인 FM 합성 알고리즘과 PCM 합성 알고리즘에 대한 DSP의 마이크로 동작을 보인 것이다. 그림 2에서 파라메터화 되는 값은 주파수 성분에 해당하는 정현파의 위상값(fc0→THETA0, fm1→THETA1, fc2→THETA2, fm3→THETA3), 위상값의 증가분(DTHETA0, DTHETA1, DTHETA2, DTHETA3), 각 정현 파에 대한 진폭값(A0, A1, A2, A3), 그리고 진폭값에 대한 증가분(DA0, DA1, DA2, DA3) 등이 된다. 그림 3의 PCM 알고리즘에서는 BANK, WAVE, THETA 등의 파라메터 값으로 외부 PCM 메모리에서 해당 파형의 샘플 값이 읽혀지고, 이것이 상태 변수 필터의 입력이 된다. 상태 변수 필터에서는 먼저 이전의 LP 값과 BP 값으로 현재 프레임에서의 lowpass 출력을 구한다. ((2)에서 (4)식 참조) 이것이 출력 진폭값인 VOL과 곱해져 내부 누적기에 저장된다. 다음으로 다음 프레임에서 사용될 BP 값을 구한다. 이것은 현재 읽어 들인 샘플 값과 이전의 BP 값, 그리고 현재의 LP 값으로 구할 수 있다.

3.3 마이크로 동작의 병렬화를 이용한 성능 향상

설계된 DSP는 알고리즘의 실행 시간을 줄이기 위해 여러 개의 마이크로 동작들이 동시에 수행되도록 하였다. 한 알고리즘 수행에 필요한 클럭 갯수가 작을수록 유리한 이유를 두 가지 관점에서 생각해 볼 수 있다. 먼저 고정된 동작 주파수 관점에서 보면, 일정한 시간의 합성 프레임 동안 수행될 수 있는 알고리즘의 갯수가 많아지게 된다. 이것은 그만큼 44.1KHz의 샘플링 주기 내에 합성할 수 있는 사운드의 갯수가 많아진다는 것을 의미한다. 한편 일정한 폴리포니에 대한 관점에서 보면, 한 클럭 사이클에 대한 주기가 늘어나는 것이므로 그만큼 낮은 주

파수에서 동작시킬 수 있게 된다. 그림 4와 그림 5에서 보인 것처럼 합성에 사용된 알고리즘은 32 사이클 이내에 실행이 끝나도록 되어 있다. 이에 따라 32개의 음을 동시에 합성하기 위해서는 48MHz의 동작 주파수가 요구된다. 그러나, 설계된 DSP에서는 여러 개의 다른 기능을 수행하는 마이크로 동작들이 병렬로 수행되게 함으로써 모든 알고리즘 코드의 실행 사이클을 24 사이클 이내로 줄일 수 있었다. 이에 따라 동작 주파수를 33MHz로 낮출 수 있었고, 0.8μm의 공정을 이용해 칩으로 구현하는 것이 가능하였다. 이것은 칩 구현과 전력 소모 측면에서 많은 이점을 가져 오는 것이다.

마이크로 동작의 병렬 수행은 파라메터 램과 내부 데이터 버스를 두 개로 나누고, 각 레지스터 입력단에 두 개의 데이터 버스로부터 입력을 선택하는 멀티플렉서를 추가함으로써, 마이크로 동작 수행시 두 개의 파라메터에 대한 동작이 동시에 일어나게 하는 것이다. 이 때 알고리즘 수행에 필요한 파라메터들은 두 개의 PRAM에 동일한 개수로 나누어져 저장된다. 그러므로, PRAM의 용량은 마이크로 동작이 순차적으로 실행되면서 한 개의 PRAM을 사용하는 것과 비교해 볼 때 변함이 없다. 동일한 세그먼트에 속하면서 두 개의 PRAM에 나누어 저장되어 있는 파라메터들은 동시에 두 개가 읽혀져서 각기 다른 기

사이클	마이크로 동작	설 명
1	MIXR, MIXL ← PRAM[MIX]	좌우 MIX 양을 설정한다.
2	SALR, AR ← PRAM[THETA3] SAHR ← 100H	파형발생기에서 내부 정현파형을 선택 SINE3에 대한 어드레스 발생
3	BR ← PRAM[DTHETA3]	다음 프레임에서 사용될 SINE3에 대한
4	PRAM[THETA3] ← AR+BR	어드레스 계산 THETA3(n+1)=THETA3(n)+DTHETA3
5	XR ← SIN[SALR] YR ← PRAM[A3]	SINE3에 대한 샘플 값을 읽어들인다. XR*YR=A3*SINE[THETA3]
6	AR, SALR ← PRAM[THETA2]	SINE2에 대한 어드레스 발생
7	BR ← XR*YR	다음 프레임에서 사용될 SINE2에 대한
8	BR ← AR+BR	어드레스 계산
9	AR ← PRAM[DTHETA2]	THETA2(n+1)=THETA2(n)+DTHETA+A3*SIN
10	PRAM[THETA2] ← AR+BR	[THETA3(n)]
11	XR ← SIN[SALR] AR, YR ← PRAM[A2]	SINE2에 대한 샘플 값을 읽어들인다. XR*YR=A2*SIN[THETA2]
12	BR ← PRAM[DTHETA2]	다음 프레임에서 사용될 SINE2에 대한
13	PRAM[A2] ← AR+BR	진폭값을 계산 A2(n+1)=A2(n)+DA2
14	SALR, AR ← PRAM[THETA1] ACC ← MIX[XR*YR]	SINE1에 대한 어드레스 발생 첫번째 FM 신호를 Accumulator에 저장
15	BR ← PRAM[DTHETA1]	다음 프레임에서 사용될 SINE1에 대한
16	PRAM[THETA1] ← AR+BR	어드레스 계산 THETA1(n+1)=THETA(n)+DTHETA1
17	XR ← SIN[SALR] BR, YR ← PRAM[A1]	SINE1에 대한 샘플을 읽어들인다. XR*YR=A1*SIN[THETA1]
18	AR ← PRAM[DA1]	SINE1에 대한 다음 진폭값 계산
19	PRAM[A1] ← AR+BR	A1(n+1)=A1(n)+DA1
20	SALR,AR ← PRAM[THETA0]	SINE0에 대한 어드레스 발생
21	BR ← XR*YR	SINE0에 대한 다음 어드레스 계산 THETA0(n+1)=THETA0(n)+DTHETA0 +A1*SIN[DTHETA1]
22	BR ← AR+BR	
23	AR ← PRAM[DTHETA0]	
24	PRAM[THETA0] ← AR+BR	
25	XR ← SIN[SALR] BR, YR ← PRAM[A0]	SINE0에 대한 샘플 값을 읽어들인다. XR*YR=A0*SIN[THETA0]
26	AR ← PRAM[DA0]	다음 프레임에서 사용될 SINE0에 대한
27	PRAM[A0] ← AR+BR	진폭값을 계산 A0(n+1)=A0(n)+DA0
28	ACC ← MIX[XR*YR]	두번째 FM 신호를 Accumulator에 저장

그림 4. FM 합성 알고리즘에 대한 마이크로 동작
Fig. 4 Micro-operations of FM algorithm.

사이클	마이크로 동작	설 명
1	MIXR,MIXL,SAHR← PRAM[WAVE_R]	좌우 MIX양을 설정한다.
2	SALR ← PRAM[FC]	Lowpass 필터의 출력값을 계산한다. LP(n)=LP(n-1)+FC*BP(n-1)
3	XR ← RAMP[FC]	
4	YR ← PRAM[BP]	
5	BR ← XR*YR	
6	SALR, PRAM[LP] ← AR+BR	
7	XR ← RAMP[LP] BR, YR ← PRAM[VOL]	Accumulator에 저장될 출력값을 계산하고, 다음 프레임에서 사용될 출력 진폭값을 계산한다. XR*YR=LP(n)*VOL(n), VOL(n+1)=VOL(n)+DVOL
8	AR ← PRAM[DVOL]	
9	PRAM[VOL] ← AR+BR	
10	SALR ← PRAM[WAVE] ACC ← MIX[XR*YR]	Accumulator에 현 프레임의 출력값 저장 ACC(n)=ACC(n-1)+MIX[LP(n)*VOL(n)]
11	SAHR ← PRAM[BANK]	현재 프레임의 입력 샘플값을 읽어 들이기 위해 PCM 메모리의 어드레스를 발생시킨다.
12	AR, SALR ← PRAM[THETA]	
13	BR ← PRAM[DTHETA]	다음 프레임에서 사용될 THETA 값을 계산 THETA(n+1)=THETA(n)+DTHETA
14	PRAM[THETA] ← AR+BR	
15	AR ← (200H,100H) BR ← PRAM[WAVE]	다음 프레임의 WAVE 값 계산 if carry=0 WAVE(n+1)=WAVE(n) if carry=1 and cur_wave≠end_wave WAVE(n+1)=WAVE(n)+1 if carry=1 and cur_wave=end_wave WAVE(n+1)=L_WAVE
16	PRAM[WAVE] ← AR+BR	
17	BR ← PRAM[L_WAVE]	
18	PRAM[WAVE] ← AR+BR	
19	XR ← PCM[SAMPLE] YR ← PRAM[AMP]	PCM 메모리부터 샘플을 읽어 들인다. XR*YR=SAMPLE*AMP=I(n)
20	SAHR ← PRAM[WAVE_IR]	필터에서 HP(n)을 계산한다. 최종적으로, AR=I(n)-LP(n) BR=-Q*BP(n-1) HP(n)=AR+BR=I(n)-LP(n)-Q*BP(n-1)
21	SALR ← PRAM[Q]	
22	AR ← XR*YR	
23	XR ← IRAMP[Q] YR ← PRAM[BP]	
24	BR ← PRAM[LP]	
25	AR ← AR+BR	
26	BR ← XR*YR	
27	SALR ← PRAM[FC]	필터에서 BP(n)을 계산한다. BP(n)=FC*HP(n)+BP(n-1)
28	SAHR ← PRAM[WAVE_R]	
29	XR ← PRAM[FC] YR ← AR+BR	
30	AR ← PRAM[BP]	
31	BR ← XR*YR	
32	PRAM[BP] ← AR+BR	

그림 5. PCM 합성 알고리즘에 대한 마이크로 동작
Fig. 5 Micro-operations of PCM algorithm.

능을 수행하는 레지스터로 전달된다. 이 때 두 개의 파라메터가 동시에 동일한 레지스터에 쓰여질 수는 없다. 레지스터는 저장된 값에 따라 고유의 동작을 수행하기 때문에 기본적으로 덧셈, 곱셈, 파형 발생 동작 등 세 가지의 동작이 동시에 수행 될 수 있다. 예를 들면, 데이터 버스0에 실린 파라메터는 AR 혹은 BR 레지스터에 저장됨으로써 덧셈 연산이 이루어지고, 데이터 버스1에 실린 파라메터는 전체 20비트 중 일정한 부분이 YR에 쓰여짐으로써 곱셈 연산이 수행되고, 또 다른 부분은 SAHR 혹은 SALR에 쓰여짐으로써 외부 PCM 메모리를 액세스 하는 동작을 수행할 수 있다.

마이크로 코드는 처음에 한 개씩 순차적으로 실행되는 것으로 하여 코딩 되었으며, 각 마이크로 동작의 분석과 타이밍 제한 조건 등을 고려하여 병렬로 코딩 되었다. 이상적으로는 데이터 버스를 두 개 사용함으로써 기존의 32 사이클에서 16 사이클로 줄일 수 있으리라 생각할 수 있지만, 데이터 hazard 문제와 곱셈 연산 시간, 그리고 파형의 샘플값을 읽어 오는데 걸리는 시간 등의 이유로 제한이 따르게 된다. 그림 6은 그림 4에서 보인 FM 합성에 대해 병렬로 수행되는 마이크로 동작을 보여 주고 있다.

사이클	마이크로 동작0(DBUS0)	마이크로 동작1(DBUS1)
1		MIXR, MIXL ← PRAM[MIX]
2	BR ← PRAM[DTHETA3]	SALR, AR ← PRAM[THETA3] SAHR ← 100H
3		PRAM[THETA3] ← AR+BR
4	XR ← SIN[SALR] YR ← PRAM[A3]	AR, SALR ← PRAM[THETA2]
5		
6	BR ← XR*YR	
7	AR ← PRAM[DTHETA2]	BR ← AR+BR
8	XR ← SIN[SALR] AR, YR ← PRAM[A2]	PRAM[THETA2] ← AR+BR
9		BR← PRAM[DA2]
10	PRAM[A2] ← AR+BR	SALR, AR ← PRAM[THETA1]
11	BR ← PRAM[DTHETA1]	ACC ← MIX[XR*YR]
12		PRAM[THETA1] ← AR+BR
13	XR ← SIN[SALR] BR, YR ← PRAM[A1]	AR ← PRAM[DA1]
14	PRAM[A1] ← AR+BR	SALR, AR ← PRAM[THETA0]
15	BR ← XR*YR	
16	AR ← PRAM[DTHETA0]	BR ← AR+BR
17	XR ← SIN[SALR] BR, YR ← PRAM[A0]	PRAM[THETA0] ← AR+BR
18		AR ← PRAM[DA0]
19	PRAM[A0] ← AR+BR	
20		ACC ← MIX[XR*YR]

그림 6. FM 알고리즘에 대해 병렬화 된 마이크로 동작
Fig. 6 Parallel micro-operations of FM algorithm.

3.4 기능 블록 설계

그림 7은 설계된 DSP의 내부 기능 블록도를 나타내고 있다. 그림을 보면 설계된 DSP는 인터페이스 유닛(IU), 메모리 유닛(MU), 알고리즘 실행 유닛(AEU), 덧셈 연산 유닛(ADU), 파형 발생 유닛(WGU), 곱셈 및 어큐뮤레이션 유닛(MACU), 그리고 샘플 데이터 출력 유닛(SDOU) 등의 7개 기능 블록으로 이루어져 있음을 알 수 있다. 각 기능 블록에는 특정한 레지스터가 있어서, 여기에 저장된 파라메터 값으로 해당 기능 블록의 동작이 이루어진다.

각 기능 블록을 살펴보면 다음과 같다.

인터페이스 유닛은 마이크로프로세서와의 데이터 전송을 담당하는 기능 블록이다. 이 기능 블록에는 5개의 레지스터가 있다. 제어 레지스터(Control Register)는 마이크로프로세서가 DSP의 합성 동작을 제어하거나, 파라메터 읽기, 쓰기 등의 동작을 행하는데 사용된다. 제어 레지스터는 두 비트로 이루어지는데 상위 비트가 0이면 DSP의 합성 동작이 이루어지고, 1이면 DSP에서 합성 동작이 일어나지 않는다. 하위 비트가 0이면 PRAM으로부터 파라메터를 읽어오는 동작이 행해지고, 1이면 PRAM에 파라메터를 쓰는 동작이 행해진다. 마이크로 프로세서와 DSP의 인터페이스 유닛과의 데이터 전달은 DSP에서 알고리즘이 수행되는 어느 때라도 일어날 수 있지만, 인터페이스 유닛과 PRAM과의 데이터 전달은 정해진 클럭 사이클에서만 일어날 수 있다. 이것은 알고리즘 수행 중에는 인터페이스 유닛이 PRAM을 액세스 할 수 없기 때문이다. 어드레스 레지스터는 전송 파라메터의 PRAM 주소를 저장한다. 데이터 레지스터는 전송 파라메터가 일시 저장되는 레지스터이다. 마이크로프로세서는 인터페이스 유닛의 각 레지스터를 바로 어드레싱 함으로써 데이터를 전송하게 된다.

메모리 유닛은 두 개의 PRAM을 포함하고, 이를 위한 어드레스와 제어 신호를 발생시킨다. 각각의 PRAM은 독립된 어드레스 발생기와 제어 신호 발생기를 가진다. PRAM의 어드레스의 상위 5비트는 0에서 31까지를 나타내는 슬롯 카운터의 출력으로 각 합성 슬롯 동안 액세스 되는 세그먼트 영역을 가리킨다. 이것은 PRAM0와 PRAM1에 공통으로 가해진다. 나머지 3비트는 세그먼트 내의 특정 파라메터 번지를 가리키는 것으로, 이 번지는 매 사이클마다 발생되는 마이크로 코드가 보유하고 있는 것이다.

알고리즘 실행 유닛은 프로그램 카운터에 따라 지정된 알고리즘의 마이크로 코드를 발생시키고, DSP 내부의 각 실행 유닛과 메모리 액세스, 그리고 버스 제어에 필요한 제어 신호를 발생시키는 곳이다. 하나의 마이크로 코드는 29비트로 이루어지고, 두 개의 필드로 나누어져 하나는 데이터 버스0를 제어하는데 사용되고, 나머지 하나는

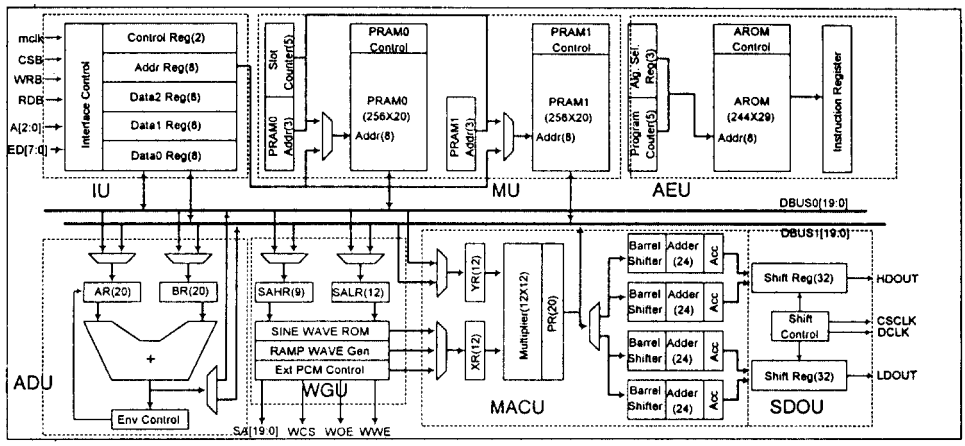


그림 7. 음원 DSP의 내부 기능 블록도
Fig. 7 Block diagram of sound DSP.

데이터 버스1을 제어하는데 사용한다. 마이크로 롬의 어드레스는 상위 3비트가 알고리즘 번호 레지스터에 의해 발생되며, 나머지 5비트는 0에서 23까지를 세는 프로그램 카운터의 출력이다.

파형 발생 유닛은 합성에 사용될 샘플 데이터를 발생시킨다. FM 합성의 경우에는 정현파의 샘플 데이터 값이 될 것이고, PCM 합성의 경우에는 외부 메모리에 저장된 샘플 값이 된다. 파형 발생부는 정현파의 샘플 데이터를 발생시키는 부분과 외부 PCM 메모리의 어드레스와 읽기 신호 등을 발생시키는 부분으로 구성된다. 정현파에 대해서는 한 주기의 샘플 값을 롬의 형태로 저장할 수 있지만, 실제로는 정현파의 대칭성으로 인해 1/4 주기의 샘플 값만을 저장해 놓고 이것으로부터 전체 주기에 해당하는 샘플 값을 발생시킨다. 정현파 외에도 내부 파형 발생기에는 4 종류의 램프 파형을 발생시키는 로직이 있다. 내부 파형 발생기나 외부 음원 롬의 어드레스는 파형 발생부에 있는 두 개의 레지스터 SAHR(9비트)과 SALR(12비트)에 의해 발생된다. 내부 파형 발생기에 대한 어드레스는 SALR만을 가지고 발생시킨다. 이 때 SAHR의 내용은 파형의 종류를 선택하는데 사용한다. 외부 음원 롬에 저장된 실제 사운드의 파형에 대한 샘플 데이터는 512개의 샘플로 이루어진 파형 블록으로 나누어 저장된다. 외부 음원 롬을 액세스 할 때 SAHR은 파형 블록의 어드레스를 나타내고, SALR은 한 파형 블록에서의 특정 샘플 데이터에 대한 어드레스를 나타내게 된다. 이렇게 파형 블록으로 나누어 저장함으로써, DSP가 액세스 할 수 있는 음원의 어드레스 영역을 확장시킬 수 있고, 반복 구간의 파형을 독립적으로 저장해서 액세스 할 수 있다

덧셈 연산 유닛은 일반적인 덧셈 연산과 더불어 주로 샘플 데이터를 읽어 올 어드레스를 계산하거나 악기음의 엔빌로프 값을 제어하는데 사용한다. 두 개의 레지스터 AR과 BR은 덧셈을 위한 두 개의 파라메터가 저장되는 곳이다. 어떤 파형 블록의 샘플 데이터를 읽어 올 번지는 AR, 얼마 간격으로 읽어 올 것인가 하는 파라메터가 BR에 실리면, 두 개의 덧셈 결과는 다음에 읽어 올 새로운 샘플 데이터의 번지가 된다. 이 과정에서 발생되는 캐리는 샘플 데이터의 번지를 결정하는데 중요한 역할을 한다. 만약 캐리가 1로 세트 된다면 한 파형 블록의 끝을 나타내므로 새로운 파형 블록-예를 들면, 반복 구간 파형 블록-의 번지를 지정해야 한다. 이것은 파형 블록을 나타내는 파라메터 값을 지장하는 번지에 새로운 파형 블록 어드레스를 저장함으로써 가능하다. 따라서 덧셈 연산 유닛에서 발생하는 캐리는 메모리 쓰기 신호를 발생시키는 하나의 조건이 된다. 엔빌로프에 대해서는 그 값이 서서히 증가하기나 감소하다가 오버플로우가 나게 되면, 덧셈 결과가 PRAM에 저장되지 않는다. 따라서 다음 프레임에서도 계속 같은 값을 사용함으로써, 진폭값의 오버플로우를 방지하게 된다.

곱셈 및 어큐뮤레이션 유닛은 일반 곱셈 연산과 더불어 최종적으로 외부로 출력될 샘플 값을 계산하고 이를 누적기에 저장하는 역할을 수행한다. 보통 XR에는 파형 발생부에서 발생된 샘플 값이 들어가고, YR에는 진폭 값이 들어가게 된다. 곱셈기의 출력 값은 하위 비트가 버려져 PR 레지스터에 저장됨으로써 다음에 내부 버스로 실리거나 배럴 쉬프터의 입력이 된다. 배럴 쉬프터는 곱셈의 결과 값을 일정한 양만큼 쉬프트 시킴으로써 외부 D/A 컨버터로 출력될 최종 샘플 데이터 값을 만들어 낸다. 이 때 쉬프트 하는 양은 MIXR과 MIXL 레지스터에 저장되어 있다. 배럴 쉬프터의 출력은 해당 합성 슬롯의 사운드가 어떤 채널로 출력될 것인가를 나타내는 채널 선택 신호에 의해 상위 혹은 하위 채널이 선택되어, 해당 채널의 누적기에 저장된다.

샘플 데이터 출력 유닛은 32비트의 쉬프트 레지스터 2개로 구성된다. 32 중 16비트는 스테레오의 왼쪽 채널에

해당하는 값이고, 나머지 16비트는 오른쪽 채널에 해당하는 값이다. 매 프레임이 끝날 때마다 누적기에 저장된 샘플 값은 16비트로 변환되어 쉬프트 레지스터로 옮겨지고, 이 블록에서 발생되는 클럭에 따라 한 비트 씩 외부 D/A 컨버터로 전달된다.

Ⅳ. 하드웨어 구현 및 검증

설계된 DSP는 COMPASS 툴에서 0.8μm 표준 셀 라이브러리로 합성되어 칩으로 제작되었다. 실제로 제작된 칩은 음원 DSP 뿐만 아니라 마이크로프로세서를 포함하고 있다. 여기에 사용된 마이크로프로세서는 인텔의 8비트 마이크로컨트롤러인 8032이다. 8032도 VHDL로 기술되어 표준 셀 라이브러리로 합성되었다. 마이크로컨트롤러까지 포함된 칩의 총 게이트 수는 약 3만 게이트(메모리 제외)이고, 코어 면적은 14.93×17.39 mm^{2}이다.

칩을 검증하기 위해 그림 10과 같이 PC에 들어가는 간단한 미디 음원 모듈을 제작하였다. PC에서 Cakewalk와 같은 미디 음악 프로그램을 연주하면 미디 인터페이스 카드를 거쳐 미디 신호가 발생되어 음원 모듈로 입력된다. 이에 따라 음원 모듈은 음을 합성해 내고 외부 앰프와 스피커에 의해 연주되는 음악을 들을 수 있다. 많은 양의 미디 데이터로 테스트 해 본 결과 설계된 칩이 원하는 동작 주파수에서 완벽하게 사운드를 합성해 내는 것을 확인할 수 있었다.

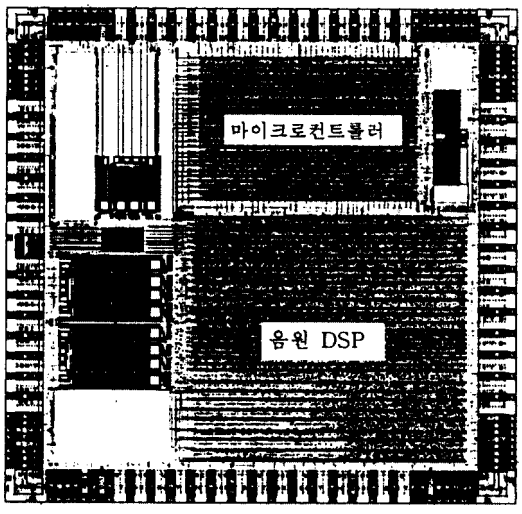


그림 8. 음원 칩의 레이아웃 사진
Fig. 8 Layout of sound chip.

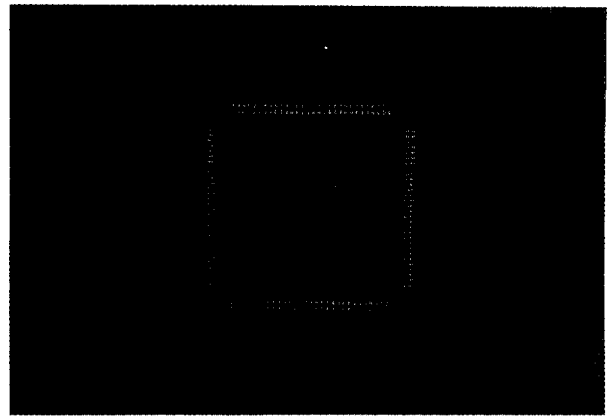


그림 9. 음원 칩의 사진
Fig. 9 Photograph of sound synthesis IC.

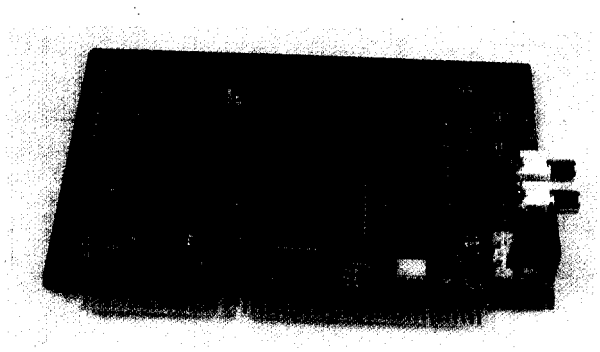


그림 10. 음원 칩을 적용한 음원 모듈 사진
Fig. 10 Sound module with the designed sound chip.

Ⅴ. 결 론

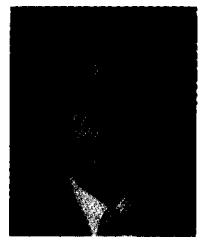
이 논문에서는 미디 신호를 받아 들여 사운드를 합성해 내는 사운드 합성 전용의 칩 설계에 대하여 논하였다. 설계된 음원 DSP는 FM 합성 방식과 PCM 합성 방식을 사용하며, CD 급의 음질로 32개의 음을 동시에 합성할 수 있다. 내부 구조에 있어서는 여러 개의 마이크로 동작이 동시에 수행될 수 있도록 함으로써 낮은 주파수에서 고음질의 폴리포니를 실현할 수 있었다. 설계된 칩은 0.8μm 셀 기반 IC로 구현되어 33MHz의 동작 주파수에서 완벽하게 사운드를 합성해 내는 것을 확인하였다.

FM 합성과 PCM 합성은 적은 양의 계산으로 사운드를 합성해 낼 수 있어 쉽게 하드웨어로 구현될 수 있지만, 음질 면에서는 부족한 점이 많다고 볼 수 있다. 최근에는 사운드 합성 알고리즘으로 physical modeling[10]이나 spectral modeling[10]과 같이 고품질의 사운드를 합성하는 알고리즘이 개발되고 있으며, 이를 실시간으로 구현하는 방법에 대한 연구가 이루어져야 할 것이다.

참 고 문 헌

1. Curtis Roads, "the computer music tutorial," MIT Press, 1996, pp. 117-133, pp. 969-1016.
2. F. Richard Moore, "Elements of Computer Music," Prentice Hall, 1990, pp. 207-263.
3. Hal Chamberlin, "Musical Applications of Microprocessors," Hayden Book Company, 1985, pp. 489-492.
4. Charles Dodge, Thomas A. Jerse, "Computer Music Synthesis, Composition, and Performance," Shirmer Books, 1985, pp. 105-194.
5. 권민도, 장호근, 김재용, 박주성, "FM 합성 방식을 이용한 악기음 합성용 DSP 설계," 한국 음향학회지, 제14권 제6호, pp. 63-73, 1995.
6. 박주성, 김형순 외, "전자악기용 디지털 신호처리 개발," 과학기술처 보고서, 1993.
7. Dream, "SAM8905 User's Guide," Dream, France, 1991.
8. John W. Gordon, "System Archetectures for Computer Music," Computing Surveys, Vol. 7, No. 2, June 1985.
9. 박주성, 장호근 외, "스테레오 사운드 카드 개발," 정보통신부 보고서, 1997.
10. Xavier Serra, "Current Perspectives in the Digital Synthesis of Musical Sounds," http://www.iua.upf.es/~xserra/articles, 1997.

▲장 호 근(Ho-Keun Jang) 1968년 1월 15일생



1993년 2월 : 부산대학교 전자공학과 (공학사)
1995년 2월 : 부산대학교 전자공학과 (공학석사)
1995년 3월~현재 : 부산대학교 전자공학과 대학원 박사과정
※주관심분야 : DSP 설계, 사운드 합성

▲권 민 도(Min Do Kwon) 1969년 4월 9일생
1992년 2월 : 부산대학교 전자공학과(공학사)
1994년 2월 : 부산대학교 전자공학과(공학석사)
1994년 3월~1996년 6월 : 부산대학교 전자공학과 대학원 박사과정
1996년 7월~현재 : LG 반도체
※주관심분야 : 마이크로프로세서 설계, DSP 설계, 신호처리

▲박 주 성(Ju Sung Park) 1953년 12월 19일생
1976년 2월 : 부산대학교 전자공학과(공학사)
1978년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)
1978년 3월~1985년 7월 : 한국전자기술연구소
1985년 8월~1989년 7월 : University of Florida, Ph.D.
1989년 8월~1991년 3월 : 한국전자통신연구소 책임연구원
1991년 3월~현재 : 부산대학교 전자공학과 부교수
※주관심분야 : DSP 설계 및 응용, 사운드 합성, 반도체 소자 모델링