FM 합성방식을 이용한 악기음 합성용 DSP 설계
Design of the DSP for the FM Sound Synthesis

권 민 도*, 장 호 근*, 김 재 용*, 박 주 성*,
김 형 순*, 윤 병 우**, 백 광 렬*, 임 창 헌***
(Min Do Kwon*, Ho Keun Jang*, Jae Yong Kim*, Ju Sung Park*,
Hyung Soon Kim*, Pyung Woo Yun**, Kwang Ryul Baek*, Chang Hun Im***)

요 약

주파수 변조(FM)의 원리를 이용하여 악기음을 합성하면, 주파수, 진폭, 변조계수 등의 파라메터들을 적절히 선택하므로써 기존의 음향 악기음과 비슷한 음을 만들 수 있다. 본 논문에서는 기존의 2 캐리어 FM 합성방식을 하드웨어적으로 구현이 용이하게 하기 위하여 합성 파라메터 수를 제한 했다. 조작된 알고리듬을 이용하여 16개 악기음을 합성할 수 있는 DSP (Digital Signal Processor)를 0.8μm의 CMOS 표준 셀을 이용하여 설계하였다. 설계된 DSP가 정상적으로 악기음을 합성하는 가를 확인하기 위하여 ASIC 에뮬레이터를 이용하여 2개의 음을 동시에 합성할 수 있는 DSP를 하드웨어적으로 구현하였다. 구현된 DSP로부터 합성된 음과 실제 악기음을 주관적인 평가와 객관적인 평가를 통하여 비교한 결과 두 음이 아주 근사함을 알았다. 최종적으로 VLSI 설계툴을 이용하여 설계된 DSP를 배치 배선한 후 시간 시뮬레이션한 결과 16개 악기음을 동시에 합성할 수 있음을 확인 했다.

ABSTRACT

The conventional acoustic sounds can be synthesized by Frequency Modulation which includes the variation of frequency, amplitude, and modulation index. In this paper the number of variable synthesis parameters are limited to easily implement the existing two carrier FM algorithm by hardware. The DSP(Digital Signal Processor), which is able to carry out the modified algorithm and synthesize 16 sounds at a time, is designed with 0.8μm standard sells. The DSP which can synthesize 2 sounds at a time is implemented by ASIC emulator to examine the sound quality of the designed DSP. Through the objective and subjective estimation, it is confirmed that the sounds of many instruments from the implemented DSP are very closed to their real sound. Finally the designed DSP is layouted and simulated by VLSI desgn tool. According to the simulation, the designed DSP has the sufficiently fast speed for synthesizing 16 sounds at a time.

I. 서 론

근세기에 들어서 전자 및 전기공학의 발달로 말미암아 전기적으로 음을 합성하는 악기들이 나타났으며, 1960년대에 들어서는 디지탈 컴퓨터가 등장함에 따라 디지탈 방식의 전자악기들이 등장하였다. 그 이후 반도체와 디지탈 신호처리 기술의 발달과 아울러 여러가지 악기음 합성방식이 개발되면서 오늘날과 같이 다양한 음색과 풍부한 기능을 가진 전자악기들로 발전했다. 악기음 합성방식에서 대표적인 것으로는 가산 합성방식[1], 감산 합성방식[2], FM 합성방식[3], Waveshaping에 의한 합성방식[4], 샘플링에 의한 합성방식[5]들이 있다. 그 중에서 John Chowning에 의해서 개발된 FM 합성방식은 간단한 구성으로도 다양한 음색을 창출해 낼 수가 있다[4][6]. FM 합성 알고리듬은 간단한 방법으로도 동적 스펙트럼을 어을 수 있으며 따라서 다양한 음색을 창출할 수 있다[7][8]. 그리고 다른 합성방식에 비하여 요구되는 메모리의 양과 파라메터들이 적게 소요되며 합성원리가 단순하므로 하드웨어 구현이 용이하며, 여러 개의 변조기와 캐리어를 이용하여 다양한 알고리듬을 구성할 수 있다.

1980년대 중반 이후 고속으로 디지탈 신호를 처리하는 DSP가 개발되어 악기음 합성에도 이용하게 되었다. 범

* 부산대학교 전자공학과
** 부산경성대학교 전기공학과
*** 부산수산대학교 전자공학과
접수일자 : 1995년 7월 19일

용 DSP를 이용하여 악기음을 합성한다고 하면 가장 빠른 DSP를 이용한다 하더라도 16개의 악기음을 동시에 처리하는 데에는 적절하지 않으며, 하드웨어가 낭비되고 가격도 비싸다. 본 논문에서는 경제적인 비용으로 악기음을 고속으로 합성하는 전용 DSP의 설계와 이의 하드웨어 구현에 관한 내용을 주로 다룬다. FM 방식으로 CD급 음질의 16개 악기음을 동시에 합성하기 위해서는 사분할 기법이 요구되며, 악기음 한 샘플 데이타를 처리하는 데 소요되는 시간은 약 1.5μs이다. 이와 같이 제한된 시간 안에 악기음을 합성하기 위해서는 고속으로 신호를 처리할 수 있는 특수한 구조를 가져야 한다. 한 명령어로 여러가지 동작을 수행하도록 하여 DSP의 throughput을 높이도록 하며, 명령어 자체를 디코딩하여 저장하므로써 알고리듬 수행시 디코딩 시간을 줄일 수 있도록 했다. 사인파를 DSP 내부 ROM에 저장하여 사인 파형 발생 시간을 줄이며, 빈번한 계산이 요구되는 주파수와 진폭 계산을 위해서 별도의 연산 블럭을 포함 시켰다.

본 논문에서는 기존의 2 캐리어 FM 합성 알고리듬을 소개하고, 그 알고리듬을 수행할 수 있는 DSP의 설계에 관한 내용을 다룬다. 제 2 장에서는 2 캐리어 FM 합성 알고리듬을 소개하며, 제 3 장에서는 설계된 DSP의 구조 설계와 명령어 세트를 설명하고, 명령어에 따른 동작 타이밍에 대해서 다룬다. 제 4 장에서는 로직 설계 및 검증을 보였는 데, 0.8μm의 표준 셀을 이용하여 COMPASS 툴로 설계한 후 툴에서 제공하는 시뮬레이터와 ASIC 에뮬레이터를 이용하여 설계된 DSP를 하드웨어적으로 구현하고 검증하는 내용을 다룬다. 제 5 장에서는 결론을 내리고 앞으로의 연구 방향에 대하여 언급한다.

Ⅱ. 2 캐리어 FM 합성 알고리듬

1973년에 J. M. Chowning이 그 당시 무선 통신에 사용되던 FM(Frequency Modulation)에 대해서 연구하던 중, 캐리어 주파수를 가청 주파수 영역으로 낮추어서 변조시키면 악기음을 합성할 수 있음을 발견하고, FM을 이용한 악기음 합성 방식을 제안했다[4]. 동적 스펙트럼은 기존의 악기음에서 음색을 결정짓는 중요한 요소인데, 모듈레이션 인덱스를 시간에 따라 변화시키면 동적 스펙트럼을 쉽게 얻을 수 있다는 데서 FM 합성의 장점을 찾을 수 있다. 또한 스펙트럼 포락선에서의 피크를 포만트라 하며 악기의 음색을 결정하는 데에는 이러한 포만트의 역할이 중요하다. 그러나 하나의 FM을 가지고서 원하는 포만트들을 만들기가 매우 힘들다. 그래서 FM 함수를 변형하여 sideband를 비대칭으로 분포시키는 방법[12], 변조기를 cascade 또는 병렬로 연결하거나 궤환(feedback)시키는 방법, 또는 2개 이상의 캐리어를 이용하여 포만트 시뮬레이션하는 알고리듬 등이 제안되었다[4][8][9][13].

Morrill은 그림 1과 같은 2 캐리어 FM 알고리듬을 이용하여 소프트웨어적으로 트럼펫 음을 합성하였다. 그림에서 사인파형 발생기의 왼쪽 입력은 진폭을 나타내고 오른쪽 입력은 주파수를 나타낸다. 그는 두개의 캐리어 주파수를 첫번째 포만트 주파수와 두번째 포만트 주파수에 맞추고, 두번째 캐리어 신호의 진폭은 첫번째 캐리어 신호의 진폭의 약 20% 정도로 해서 두 포만트의 상대적인 진폭을 적절한 레벨로 맞추었다. 그리고 2개의 캐리어 신호에 포락선 발생기를 걸어서 전체적인 악기음의 진폭에 대한 포락선을 구현했으며, 두 개의 변조 신호에 포락선 발생기를 걸어 동적 스펙트럼을 구현했다. 첫번째 캐리어 신호의 모듈레이션 인덱스를 두번째 캐리어의 모듈레이션 인덱스에 비하여 상대적으로 작게하여 첫번째 캐리어 주파수 근처에서 가장 강한 주파수 성분이 나타나도록 했다. 2 캐리어 알고리듬을 수식적으로 표현하면 (1)식과 같으며, 이를 Bessel 함수로 표시하면 (2)식과 같다.

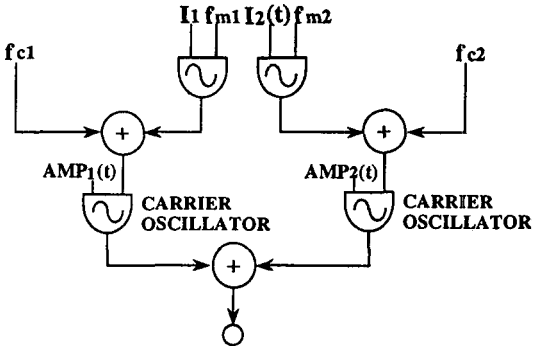
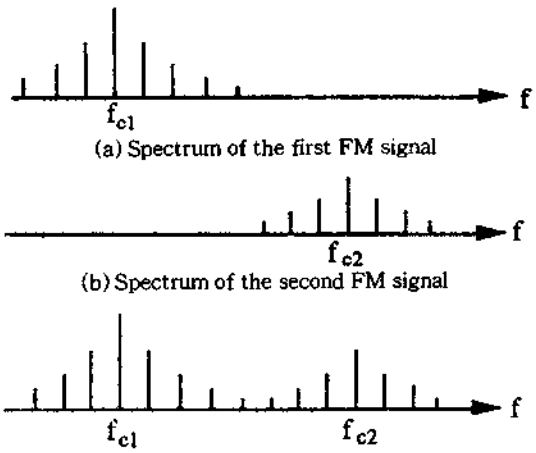


그림 1. 2 캐리어 FM 합성 알고리듬
Fig 1. 2 carrier FM synthesis algorithm

$$Y(t) = AMP_1(t)\sin\{2\pi f_{c1}t + I_{1(t)}\sin 2\pi f_{m1}t\} + AMP_2(t)\sin\{2\pi f_{c2}t + I_{2(t)}\sin 2\pi f_{m2}t\} \quad (1)$$

$$Y(t) = AMP_1(t)\sum_{n=-\infty}^{\infty} J_n(I_1(t))\sin(2\pi f_{c1}t + n2\pi f_{m1}t) + AMP_2(t)\sum_{m=-\infty}^{\infty} J_m(I_2(t))\sin(2\pi f_{c2}t + m2\pi f_{m2}t) \quad (2)$$

여기서 $J_n(I)$은 n차의 제 1 종 Bessel 함수이며, f_c는 캐리어 주파수, f_m은 변조 주파수, AMP(t)는 진폭이다. 모듈레이션 인덱스는 $I(t) = \Delta f/f_m$이며, Δf는 캐리어 주파수로부터의 변이(deviation)이다. FM 신호의 주파수 분포는 첫번째 캐리어 주파수 f_{c1}을 중심으로 nf_{m1}에 sideband가 나타나고, 두번째 FM 신호에 대해서도 f_{c2}를 중심으로 mf_{m2}에 sideband가 나타난다. 모듈레이션 인덱스가 증가할수록 캐리어에 존재하는 에너지가 sideband로 분포되면서 sideband의 진폭이 작아지며 수도 증가한다. 그러므로 그림 2와 같이 두개의 FM에 대한 파라메터를



(a) Spectrum of the first FM signal

(b) Spectrum of the second FM signal

(c) Addition of the spectra of two FM signals

그림 2. 두 FM 신호에 의한 스펙트럼 포락선
Fig 2. 2 Spectrum envelope by 2 FM signals

독립적으로 제어해서 원하는 주파수 분포를 얻을 수 있다.

설계하고자 하는 DSP의 하드웨어 구현을 고려하여 첫번째 모듈레이션 인덱스 $I_1(t)$의 크기를 고정 시켰다. 기존의 2 캐리어 FM 알고리듬과 비교해서 본 논문에서 응용한 2 캐리어 FM 알고리듬은 구현상에 있어서 다음과 같은 차이점이 있다. Morrill이 제안한 알고리듬을 충실하게 구현하려면 31개의 파라메터가 필요하지만[9], 이를 본 논문에서와 같이 응용하면 설계된 DSP 구조에서 파라메터를 16개로 줄일 수 있다. 이렇게 함으로써 DSP 내부의 메모리를 줄일 수 있으며 합성 속도도 빨라진다. 또한 2개의 FM 중 하나의 모듈레이션 인덱스만 제어하므로 구현하기가 쉽다. 그러나 첫번째 모듈레이션 인덱스가 고정되므로 합성하고자하는 악기음의 음색 표현에 제약이 따른다. 그리고 시간에 따라 포만트가 커지기도 하고 작아져서 없어지기도 하는 데, 2 캐리어 FM 알고리듬으로는 이러한 포만트의 생성, 소멸을 완벽하게 재현하는 데에는 한계가 있다. 이를 극복하기 위해서 3개 이상의 캐리어를 이용하기도 한다[16].

FM 방식을 사용하여 악기음을 합성하고자 할 때는 알고리듬 파라메터들의 선정이 중요하다. 이들 파라메터로는 악기음의 피치를 결정하는 주파수 파라메터, 음량을 결정하는 진폭 파라메터, 주파수 분포를 결정하는 모듈레이션 인덱스가 있다. 같은 알고리듬을 사용하더라도 이들 파라메터의 값에 따라 여러 악기음을 만들 수 있으며, 파라메터의 선택에는 어느 정도의 이론적인 지식과 더불어 경험적인 know how에 의존하는 경향이다. 본 논문에서는 합성하고자 하는 악기음의 샘플 데이타를 시간 영역과 주파수 영역에서 분석하여 다음과 같은 과정을 거쳐서 파라메터를 결정했다. 먼저 합성하고자 하는 악기음의 1차, 2차 포만트를 조사하여 2개의 캐리어 주파수 f_{c1}와 f_{c2}를 조사한 포만트에 위치하도록 선택한다. 두 개의 캐리어 주파수는 서로 연관 관계가 있다. 즉, 주파수 상에서 제일 낮은 위치에 있는 포만트에 에너지가 가장 많이 몰려 있으며 합성하고자 하는 악기음의 피치를 결정하므로 f_{c1}을 기본 주파수로써 첫번째 포만트 위치에 둔다. 두번째 캐리어 주파수(f_{c2})는 기본 주파수(f_{c1})의 하모닉스가 되면서, 두번째 포만트 주파수(f_{f2})에 가장 근접한 주파수로 잡는다. 주파수 상에서 각 sideband의 간격은 변조 주파수에 의해서 결정되므로 이에 따라 변조 주파수를 결정해야 한다. 특히 캐리어 주파수와 변조 주파수의 비를 하모니시티 비라고 하며 음색에 많은 영향을 미친다[8][11]. 이와 같이해서 f_{c1}, f_{c2}, f_{m1}, f_{m2}을 결정한 후 모듈레이션 인덱스와 진폭을 조정하여 전체적인 주파수 분포를 조절한다. 이 과정에서는 시간 영역과 주파수 영역에서 자연음에 대한 포락선과 각 주파수 성분들의 시간에 따른 변화를 보고, 진폭에 가해지는 포락선과 모듈레이션 인덱스에 가해지는 포락선의 모양과 크기, 길이등을 반복에 의해서 결정한다.

알고리듬은 4개의 정현파 발진기로 구성되는 데, 정현파 발진기는 정현파를 DSP 내부에 테이블 형태로 저장해 두고, 저장된 정현파 데이타를 읽어오는 간격을 조정하여 필요한 주파수를 결정한다[8][13][14]. 한 주기에 N개의 데이타를 가지는 정현파형에 대해서 주파수가 f_0인 음을 합성하고자 할 경우에 샘플의 증가분 SI(Sample Increment)와 f_0 사이에는 이미 널리 알려진 $SI = N(f_0/f_s)$와 같은 관계가 있다. 여기서 f_s는 샘플링 주파수이다. 발진기를 구현하기 위해서는 주파수와 진폭 파라메터가 요구되며, 이들 파라메터는 DSP의 하드웨어 구조와 관계있다. 주파수는 샘플의 증가분 SI에 의해서 결정되며, DSP의 샘플링 주파수가 44.1 KHz이므로 샘플의 증가분과 합성하고자 하는 주파수 사이에는 $SI = 11.88862734 \times f_0$라는 간단한 관계식이 성립한다. 여기서 상수 11.88862734는 N/f_s에서 $N = 2^{19} - 1$과 $f_s = 44100(Hz)$을 대입해서 나온 값이다. N 값은 알고리듬 파라메터 SI가 가질수 있는 최대값이며, 이는 파라메터의 워드 폭(19비트)에 의해서 결정된다. 그리고 악기음의 포락선은 발진기의 진폭을 시간에 따라 증감시켜서 구현하므로 진폭의 증감분을 나타내는 파라메터가 필요하다. 이 중에서 변조 신호의 진폭은 모듈레이션 인덱스를 나타내므로 캐리어 신호의 진폭과는 다른 값을 가지며, 그 크기와 모듈레이션 인덱스와의 관계는 $AMP_I = 83442.86 \times I$와 같이 결정된다. 여기서 I는 모듈레이션 인덱스이며 상수 83442.86는 $(2^{19}-1)/2\pi$에서 나온 값이다. 이 값은 19비트로 표현 가능한 모듈레이션 인덱스가 0에서 2π 범위의 sine 함수의 위상과 관련되므로 $2\pi/(2^{19}-1)$로 scaling되어야 하는 데, 이를 보상하기 위해서 역으로 변조항에 곱한 값이다.

하나의 발진기에 대해서 2개의 파라메터와 2개의 레지스터가 필요한 데, DSP 내부에는 별도의 레지스터가 없고 메모리를 레지스터로 사용한다. 2개의 파라메터에는 샘플의 증가분과 진폭의 증가분을 결정짓는 SI와 DAMP (변조신호인 경우에는 DI)가 있으며, 이들 파라메터에

대한 메모리 레지스터로는 사인 함수의 위상에 대한 번지값을 저장하는 FREQ와 진폭값을 저장하는 AMP(변조신호인 경우에는 I)가 있다. 이 중에서 첫번째 변조 신호에 대한 모듈레이션 인덱스의 크기는 일정하게 고정되므로 DI 파라메터가 없다. 여기에다 현재 알고리듬 파라메터가 저장되는 RAM 블럭에 대한 상태를 나타내는 파라메터가 있다. 그러므로 2 캐리어 FM 알고리듬으로 한 악기음을 합성하고자 할 때는 3개의 발진기에 각각 4개의 메모리 번지가 필요하며, 나머지 하나의 발진기에는 3개의 메모리 번지, 그리고 상태 파라메터가 필요하므로 총 16개의 메모리 번지가 요구된다. 따라서 한 음에 대한 전체 파라메터의 갯수는 메모리 레지스터를 포함하여 16개이다.

MATLAB과 C언어를 사용하여 2 캐리어 FM 알고리듬의 시뮬레이터를 만들어서, 악기 원음에 대한 샘플 데이타의 주파수 성분을 보고 캐리어 주파수와 변조 주파수, 모듈레이션 인덱스를 예측한 후, 반복에 의해서 이들 파라메터를 결정한다. 이와 같이 하여 하프시코드, 바순, 잉글리쉬 호른, 오보에등의 파라메터를 추출하여 악기음을 합성하였으며, 하나의 악기음에 대해서도 C4, A4 등 여러 음들을 합성했다. 예를 들어 하프시코드 음 중에서 A5 음에 대한 파라메터를 표 1에 보였다. 표에서 SI_{M1}, SI_{M2}는 두 변조 주파수의 샘플 증가분이고, SI_{C1}, SI_{C2}는 두 캐리어 주파수의 샘플 증가분이다. 이들 샘플 증가분은 매 합성 프레임마다 각각의 FREQ 레지스터에 더해진다. I_1, I_2는 첫번째와 두번째 모듈레이션 인덱스이며, DI_2는 두번째 모듈레이션 인덱스에 대한 변화량이다. $DAMP_1$, $DAMP_2$는 두 FM 신호의 출력 진폭에 대한 증가분이다. AMP_1, AMP_2는 2개의 캐리어 신호의 진폭이며, 초기값은 두 포만트의 크기에 의해서 결정된다. 포락선의 각 구간에 대한 진폭의 변화량 $DAMP_1$, $DAMP_2$는 샘플 데이타로부터 선형적으로 근사화해서 추출했다. 상태 파라메터가 저장되는 번지의 idle 비트를 해제시키면 알고리듬 ROM에 저장되어 있던 알고리듬이 수행되면서 악기음을 합성한다. 이들 파라메터 값들은 모두 19비트이며, SI에 대응하는 주파수는 $f_0=f_s/N\times SI$에 의해 결정되며, AMP는 전체 19비트중 상위 12비트에 의해서 결정된다.

Ⅲ. 구조 설계와 명령어 세트

FM 알고리듬을 이용하여 악기음을 합성하는 것 외에 동시에 16개의 악기음 합성, CD급 음질, 스테레오 효과, 정현파 발진기 내장등 이러한 요구 조건을 만족하는 DSP를 설계하고자 했다. 따라서 본 논문에서 설계한 DSP는 최소한의 비용으로 위의 요구 조건을 만족하면서 악기음을 합성하기 위해 시분할 합성 개념, 간단한 하드웨어 구조 설계, 명령어 구조 및 동작 타이밍에서 범용 DSP와는 다르다. 시분할 개념은 동시에 여러개의 악기음을 합성하기 위해서 사용되며 이를 그림 3에 나타내었다. CD급의 44.1 KHz의 샘플링 레이트로 16개의 악기음을 합성한다고 할 때, 한 악기음에 대한 샘플 데이타가 출력되는 시간 간격을 합성 프레임(22.68μs=1/44100sec)이라고 한다. 하나의 악기음 샘플이 처리되는 시간을 슬롯(1.42μs)이라고 하며, 한 타임 슬롯동안 FM 알고리듬은 알고리듬 파라메터들을 이용하여 하나의 샘플 데이타를 만들어낸다. 알고리듬은 32개의 명령어 사이클로 구성되며, 한 명령어 수행시간은 44.3ns이다. 이 중 1 사이클은 외부로부터 알고리듬 파라메터를 받아들이는 데 사용된다. DSP 내부에는 2 캐리어 FM 합성 알고리듬을 저장하는 ROM과 파라메터를 저장하는 RAM이 있다. RAM은 16개의 세그먼트로 구성되어 있으며 한 세그먼트에는 16개의 파라메터를 저장할 수 있다. RAM의 16개 세그먼트는 16개의 타임 슬롯과 일대일로 대응되며, 각 타임 슬롯마다 RAM의 세그먼트들이 연속적으로 엑세스되어 해당 악기음에 대한 샘플 데이타를 처리한다.

표 1. 알고리듬 파라메터 추출에 대한 예(하프시코드의 A5 음)
Table 1. The example about the extraction of the algorithm parameters(A5 note of Harpsichord)

Parameter	Value	Parameter	Value	Parameter	Value	Parameter	Value
SI_{M1}	51BBH (1760 (Hz))	SI_{M2}	28DDH (880 (Hz))	AMP_1	3C884H (0.9453)	I_1	145F2H (1)
SI_{C1}	28DDH (880 (Hz))	SI_{C2}	198ABH (8800 (Hz))	AMP_2	3C884H (0.9453)	I_2	1E8ECH (1.5)
$FREQ_{M1}$	0	$FREQ_{C1}$	0	$FREQ_{M2}$	0	$FREQ_{C2}$	0

Parameter	Decay 1		Decay 2		Sustain		Release	
	Value	Duration	Value	Duration	Value	Duration	Value	Duration
DI_2	-3H	23ms	0	67ms	-1BH	317ms	-1H	1587ms
$DAMP_1$	-1BH	23ms	-1BH	67ms	0	317ms	-2H	1587ms
$DAMP_2$	-D1H	23ms	0	67ms	0	317ms	-4H	1587ms

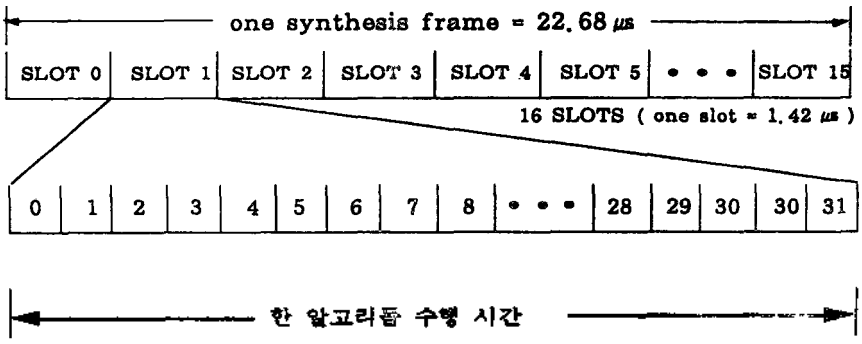


그림 3. 시분할 합성의 개념
Fig 3. Notion of the time division multiplexing

1. 구조 설계

설계된 DSP의 내부 구조는 그림 4와 같이 인터페이스 블럭, 메모리 블럭, 주파수/진폭 연산부, 파형 발생부, 데이타 출력부 등 기능별로 5개의 블럭으로 나누었으며, 내부 버스폭은 19 비트이다. 전체 동작 개요는 다음과 같다. 인터페이스 블럭을 통하여 입력되는 알고리듬 파라메터는 내부 RAM에 저장되고, ROM에 저장되어 있던 알고리듬은 이러한 파라메터를 이용하여 악기음을 합성한다. 주파수/진폭 연산부에서는 주파수와 진폭을 계산하고, 계산된 주파수 값에 따라 파형 발생부에서는 정현파 샘플 데이타를 읽는다. 이들 샘플 데이타와 진폭은 데이타 출력부에 있는 곱셈기에서 곱해져서 Barrel Shifter

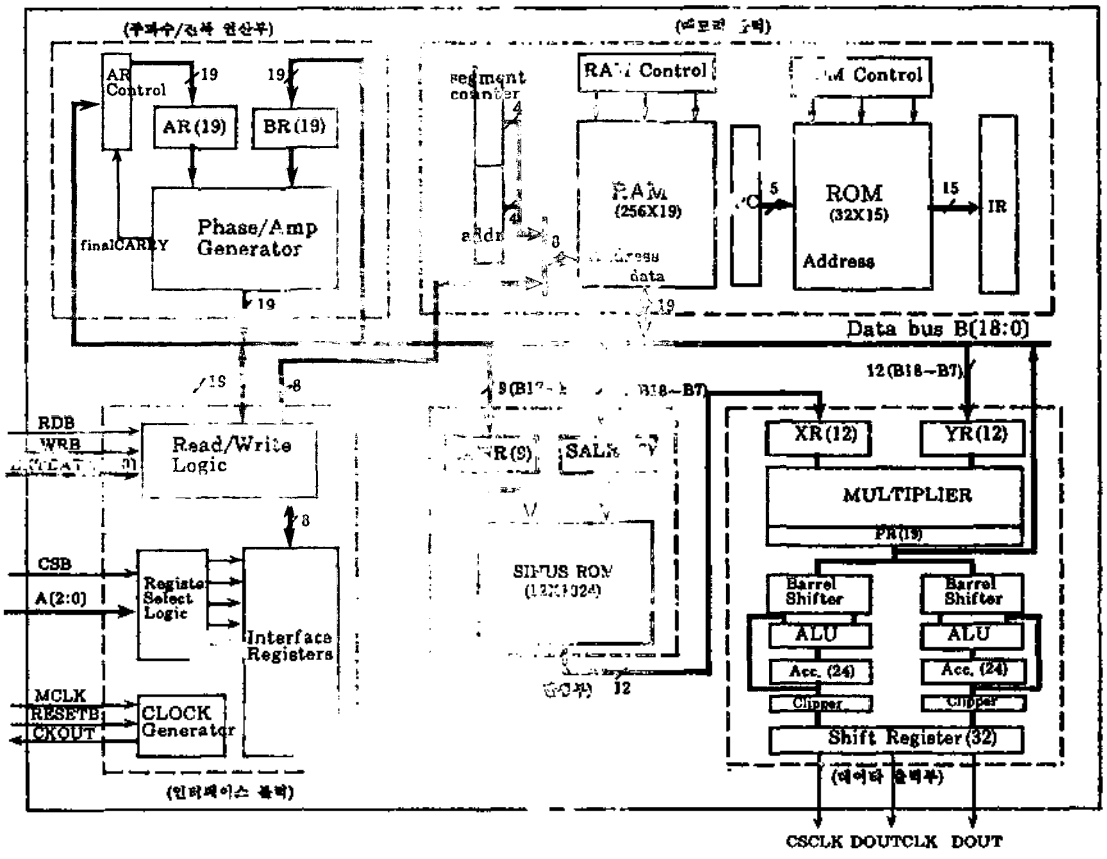


그림 4. FM 악기음 합성용 DSP의 내부 블럭도
Fig 4. The internal block diagram of the DSP for FM sound synthesis

를 거쳐 ACC에 저장된다. ACC에 저장되어 있던 악기음 데이타는 다음 프레임이 시작될 때 한 비트씩 쉬프트 레지스터를 거쳐 외부 DAC로 전송된다. 악기음 합성용 DSP 설계를 목표로 했기 때문에 다음과 같은 특정 기능 요소들이 필요하다.

인터페이스 블럭 : 악기음을 자연스럽게 합성하는 데에는 알고리듬 파라메터들이 시기적절하게 update되어야 하는 데 이러한 데이타는 인터페이스 블럭을 통하여 입/출력된다. 인터페이스 블럭은 데이타의 입/출력 통로로서 뿐만 아니라 블럭내에 있는 컨트롤 레지스터의 정보에 따라 전체 DSP의 동작을 제어한다.

메모리 : 신호처리 속도를 향상 시키기 위하서 FM 알고리듬을 저장하는 ROM과 파라메터를 저장하는 RAM을 따로 두어서 임의의 알고리듬 사이클 동안 알고리듬 파라메터들을 엑세스할 수 있도록 한다.

정현파 발생기 : 12 비트로된 4096개의 샘플 데이타 중 1/4 주기에 해당하는 1024개의 샘플 데이타를 ROM에 저장했다. 합성에 사용될 정현파의 값은 (4)식과 같이 결정된다.

$$X = 0.71875\sin(\pi/2048)\times FREQ + \pi/4096), \quad 0 \le FREQ \le 4095 \tag{4}$$

여기서 FREQ는 악기음의 위상과 관련되는 정현파의 어드레스이며, X는 출력되는 정현파의 값이다. 정현파 샘플 데이타가 4096개 이므로 정현파 샘플의 번지는 FREQ의 상위 12 비트로 표시 가능하다. 사인함수에 곱해진 상수와 위상에 더해진 반 주기는 look-up 테이블 방식에 의한 정현파 발생기에 대한 오차를 줄이기 위한 것이다 [14][15]. 상수 0.71875는 $1/\sqrt{2}$을 2진수로 표현했을 때 가장 근접한 수치이다.

주파수/진폭 연산기 : 주파수와 진폭은 악기음을 결정짓는 가장 중요한 파라메터이다. 이러한 주파수와 진폭 계산을 소프트웨어에 의존하지 않고 하드웨어로 구현하므로써 속도의 향상을 기하기 위한 기능 요소이다. 주파수 연산을 위한 위상 계산은 현재의 정현파 위상 FREQ에다 샘플의 증가분 SI를 더해서 다음에 읽어올 샘플의 위상을 구할 수 있다. 진폭 계산에서 만약 진폭에다 진폭의 변화량을 합한 결과에 오버플로우가 발생할 경우에는 최대 진폭값을 그대로 유지한다.

데이타 출력부 : 데이타 출력부에서는 샘플 데이타와 진폭을 곱하며, 스테레오 효과를 위하여 좌우 채널에 대해서 감쇄가 일어난다. 16개 악기음을 동시에 내는 듯한 효과를 내기 위하여 한 프레임 동안 합성된 16개 악기음 데이타를 ACC에 저장했다가, 좌우 채널당 16 비트씩 시리얼로 외부 DAC로 출력한다.

레지스터 : DSP 내부에는 별도의 범용 레지스터가 없는 대신 다음과 같은 특별 레지스터가 존재한다. 일반적인 덧셈과 아울러 주파수 계산과 진폭을 계산하는 데 사용되는 것으로 AR(A Register), BR(B Register)이 있다. 정현파 샘플 데이타 번지를 저장하는 것으로 SAHR, SALR(Sample Address High/Low Register)가 있다. 정현파 샘플 데이타는 XR(X Register)에 저장되며, 샘플 데이타에 곱해질 진폭은 YR(Y Register)에 저장된다. 곱셈의 결과값은 PR(Product Register)에 저장된다. LMR, RMR(Left/Right Mix Register)은 좌우 채널의 믹스 양을 저장한다.

2. 명령어 구조 및 동작 타이밍

명령어 세트를 크게 분류해보면 메모리나 레지스터의 내용을 내부 데이타 버스에 실어주는 읽기타입 명령어와 데이타 버스에 실린 내용을 메모리나 레지스터에 저장하는 쓰기타입 명령어로 나눌 수 있다. 명령어에 따라 기능 블럭이 데이타를 읽어들이면 자동적으로 특수한 기능을 수행하므로 읽기와 쓰기타입 명령어를 잘 조합하므로써 원하는 알고리듬을 수행할 수 있다. DSP 내에서 데이타를 버스로 내 보내는 것으로는 PR, 주파수/진폭 연산부, RAM이 있고, 버스로부터 데이타를 받는 것으로는 XR, YR, AR, BR, SALR, SAHR, LMR, RMR, RAM이 있다. 그리고 다른 명령어와 결합하여 특수한 동작을 수행하는 명령어와 no operation의 기능을 수행하기 위하여 이전의 버스값을 계속 유지하는 명령어가 있다. 버스에 데이타를 싣는 요소와 버스로부터 데이타를 받는 요소에 각각 1 비트씩을 할당하여 개별적으로 제어하므로, 파라메터를 엑세스하기위한 RAM의 번지를 포함하여 명령어를 총 15 비트로 구성했다.

DSP에서는 하나의 알고리듬이 실행되는 시간이 매우 제한적이므로, 명령어 디코딩 시간을 줄이기 위해 명령어 구조를 디코드된 형태로 ROM에 저장하여 디코더가 필요없도록 했다. 이를 위하여 DSP 어셈블러를 만들어서 알고리듬 소스 코드를 DSP 명령어로 디코딩하여 알고리듬 ROM을 컴파일 했다. 따라서 명령어 패치가 일어나면 바로 다음 사이클에서 명령어 실행이 일어나는 2 스테이지 파이프라인 구조로써 패치에서 실행까지의 시간이 매우 짧다.

명령어의 OP(operation) 코드는 각각의 비트가 컨트롤 신호가 되어 레지스터와 기능 블럭을 제한하므로 별도의 컨트롤 블럭이 필요없다. RAM의 read/write 컨트롤 신호는 패치된 명령어들의 조합으로 만든다. 8개 레지스터의 입력 제어신호는 명령어의 OP 코드의 특정 비트와 일대일 대응된다. 명령어에서 오퍼랜드는 4비트로써 RAM의 16개 파라메터를 엑세싱한다. OP 코드중 한 비트는 악기음 진폭이 오버플로우될 때 최대값을 유지하도록하는 기능을 수행하는 데 이용된다. 명령어의 나머지 2 비트는 조합하여 DSP 내부 데이타 버스에 데이타를 싣는 3개 블럭의 제어신호가 된다. 각 기능 블럭은 별도의 제어신호가 없는 대신에 매 사이클마다 입력 레지스터의 값을 조작하므로 알고리듬을 작성할 때 이를 잘 고려하

여 원하는 데이타를 원하는 시간에 읽고 쓸 수 있도록해야 한다. 쓰기타입 명령어의 경우에는 여러개가 동시에 수행이 가능하여 DSP의 throu-ghput을 높일 수 있다.

설계된 DSP는 단일 클럭을 사용하며 이 클럭에 모든 명령어의 페치와 실행이 동기된다. 그림 5에 이들 명령어의 동작 타이밍도를 나타내었다. 읽기타입 명령어는 명령어 페치 후 다음 사이클의 클럭 rising edge에서 IR에 저장되며 바로 읽기 동작이 일어난다. 쓰기타입 명령어는 IR에 저장된 후 반 사이클 후에 제어신호가 나온다. 이 제어 신호는 그 다음 클럭의 rising edge에서 레지스터에 데이타를 쓰게 한다. 쓰기 타입 명령어중 RAM에 데이타를 쓰는 명령어는 읽기 명령어와 동일한 동작 타이밍을 가진다. 이것은 rising edge로부터 다음 클럭의 rising edge 동안만 메모리에 쓰기 위한 데이타가 유효하기 때문이다.

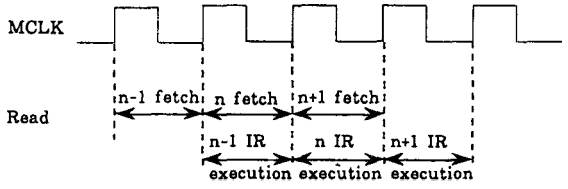


그림 5. 명령어 실행 타이밍도
Fig 5. Instruction execution timing diagram

DSP가 알고리듬 수행중에도 외부와 데이타를 주고 받기위하여 그림 6에서 보는 바와같이 알고리듬 수행중에 0번 명령어 페치가 일어나는 동안에 RAM과 인터페이스 블럭간에 데이타 전송이 일어난다. DSP를 제어하는 외부 디바이스(예 : 마이크로프로세서)에서 DSP로 데이타를 전송할 때는 DSP의 내부의 타이밍과는 상관없이 임의의 시간에 DSP 내부의 인터페이스 블럭에 데이타를 저장한다. 이들 두 디바이스 간에 synchronization problem을 해결하기 위하여 flip-flop으로 synchronizer 회로를 구성한다. DSP로 알고리듬 파라메터를 전송할 때는 현재 악기음을 합성하지 않는 idle RAM 블럭을 선택하여 순차적으로 알고리듬 파라메터를 전송한 다음, 마지막 알고리듬 파라메터에 있는 상태 플래그에 의해 선택된 RAM 블럭을 액티브 시킨다. 이와 같은 일들은 마이크로프로세서의 시스템 프로그램이 담당한다.

Ⅳ. 로직 설계 및 검증

본 논문에서 설계한 DSP는 인터페이스 블럭, 메모리 블럭, 파형 발생부, 주파수/진폭 연산부, 데이타 출력부로 나누어서 각 블럭별로 설계한 다음 전체 로직을 구성하는 bottom-up 방식으로 설계했다. 로직 설계는 0.8μm의 표준셀을 이용하여 VTI사의 ASIC 디자인 툴인 COMPASS을 이용했다. 각 블럭의 로직 설계는 다음과 같다.

인터페이스 블럭에서 외부로 연결된 신호로는 CSB, RDB, WRB, A[2:0], EXTDATA[7:0]가 있으며, 내부 메모리의 번지를 저장하는 레지스터, 3개의 8 비트 데이타 레지스터, DSP 전체의 동작을 제어하는 컨트롤 레지스터등 5개의 레지스터와 레지스터 선택 로직으로 구성된다. A[2:0]에 의하여 5개의 레지스터중 하나가 선택되며, EXTDATA[7:0]를 통하여 마이크로프로세서와 데이타를 주고 받는다.

메모리 블럭에서 FM 알고리듬은 32개의 명령어로 구성되며 ROM에 저장된다. 한 명령어는 15 비트이므로 ROM의 크기는 1×32×15=480 비트이다. 파라메터는 악기음 합성중에도 그 값이 계속 변하므로 RAM에 저장한다. RAM은 16개 세그먼트마다 16개의 파라메터를 저장하며, 파라메터의 폭은 19 비트이므로, RAM의 크기는 16×16×19=4864 비트이다. RAM과 ROM은 COMPASS에서 제공하는 셀 컴파일러를 이용하여 만들었다. 메모리 블럭은 RAM과 ROM 및 번지 생성 로직, 메모리 컨트롤 로직으로 구성된다. 한 프레임동안 RAM의 상위 번지값은 세그먼트 카운터에 의해, 하위 번지값은 페치된 명령어에 의해 지정되게 했다. 인스트럭션 레지스터는 ROM에서 페치된 명령어 코드를 저장하고 이것을 동작 타이밍에 맞추어 각종 제어 신호를 발생한다.

주파수/진폭 연산부는 AR, BR과 19 비트 가산기, 캐리 발생부, MW(Memory Write) 플립플롭으로 구성된다. 가산 결과에 의해 생기는 최종 캐리와 플립플롭의 출

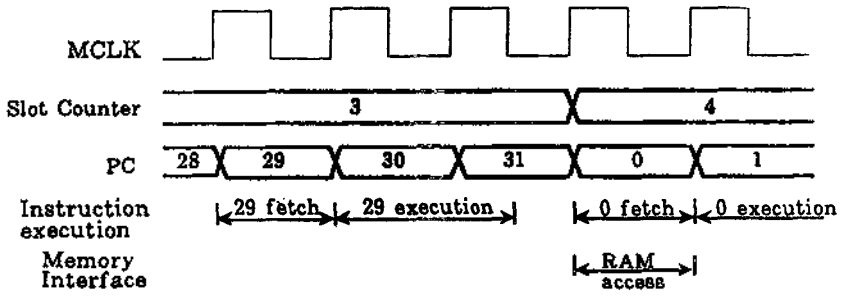


그림 6. 메모리 인터페이스 타이밍도
Fig 6. Memory interface timing diagram

력에 따라 데이타 버스에 실린다. AMP는 11.6ms마다 DAMP에 의해서 값이 update되는 데, 한 합성 프레임이 22.68μs이므로 11.6ms는 512 프레임에 해당한다. 512 프레임동안 DAMP가 1일 경우 AMP를 1만큼 변화 시키기 위해서는 12 비트 AMP의 하위에 7 비트가 필요하므로 DAMP는 19 비트가 되었다. 마찬가지로 FREQ도 12 비트지만 인터폴레이션을 위해서 12 비트 FREQ의 하위에 7 비트를 두어서 RAM 파라메터가 19 비트가 되었다. 따라서 RAM의 워드 폭이 19 비트로 결정되었으며 DSP 내부 버스폭도 19 비트이다.

파형 발생부는 SAHR, SALR, 정현파 테이블로 구성된다. PI 파라메터는 4096개의 정현파 샘플 데이타 번지를 지정하며 SALR에 저장된다. 정현파 테이블은 메모리 용량을 줄이기 위해 정현파의 1/4 주기를 SINUS ROM에 저장했다. 따라서 ROM의 크기는 12×1024 비트이며, COMPASS에서 제공하는 셀 컴파일러를 이용해서 생성했다.

샘플 데이타 출력부는 12×12 곱셈기와 곱셈기의 입력 레지스터인 XR, YR 그리고 Barrel Shifter, ALU, ACC, 클리핑(Clipping) 회로, 32 비트 쉬프트 레지스터로 구성된다. XR에는 정현파 샘플값이 실리며, YR에는 진폭값이 실린다. XR과 YR의 값은 12×12 signed multiplier에서 곱해져서 그 결과값이 19 비트로 변환되어 PR에 저장된다. 곱셈기 설계는 곱셈 계산 과정을 줄이고 동작 속도를 개선하기위해서 Modified Booth Algorithm과 CSA 기법들을 적용하여 설계 했다. PR의 출력값은 레프트 채널과 라이트 채널로 나뉘어져서 각각 Barrel Shifter에서 믹스 레벨만큼 감쇄된다. 감쇄된 후 ALU에서 이전 슬롯의 샘플과 더해져서 ACC에 저장되는 데, 이 때 오버플로우를 방지하기 위하여 24 비트로 확장되어 ACC에 저장된다.

블럭별로 로직 설계를 마친 후 COMPASS에서 지원하는 시뮬레이터를 이용하여 전체 로직을 시뮬레이션 했다. 정상적으로 DSP를 동작 시킨 후 II장에서 추출한 알고리듬 파라메터를 DSP 내부 RAM에 다운로드하여 악기음 샘플 데이타를 처리하도록 wave 화일을 작성하여 시뮬레이션 한 후, 매 프레임이 끝날 때마다 DAC로 출력되는 최종 데이타값을 살펴봄으로써 내부 로직이 제대로 동작하는 지를 확인했다.

설계한 DSP가 FM 방식으로 악기음을 제대로 합성하는 지를 확인하기 위하여 그림 7과 같이 간단한 보드를 구성하여 PIE사의 MARS II라는 ASIC 에뮬레이터를 이용했다. 보드는 PC(Personal Computer)로부터 MIDI (Musical Instrument Data Interface) 신호를 수신한다. 이를 마이크로프로세서가 프로그램 ROM에 저장된 프로그램에 따라 입력되는 MIDI에 해당되는 파라메터를 만들어, DSP의 내부 RAM에 전송한다. DSP에서는 내부 ROM에 저장된 알고리듬에 따라 RAM에 저장된 파라메터를 이용하여 악기음을 합성하여 외부 DAC로 내보낸다.

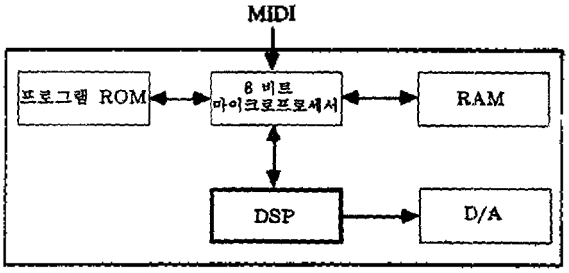


그림 7. 시스템 보드의 구성도
Fig 7. Block diagram of the system board

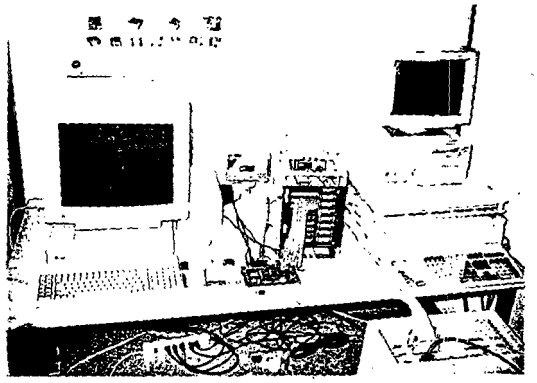


그림 8. ASIC 에뮬레이터를 이용한 실험 세트
Fig 8. The experiment set using the ASIC emulator

보드에서 DSP가 들어갈 자리에 ASIC 에뮬레이터를 연결한 후 전체 시스템을 동작 시켰다. MARS II는 LBM (Logic Block Module), 워크스테이션, Debugware, 그리고 PIA(Pod Interface Adaptor)로 구성된다[17]. LBM은 FPGA의 어레이로 구성된 로직 모듈로써 설계한 DSP의 로직들은 여기에서 구현된다. 워크스테이션은 ASIC 에뮬레이터에 대한 소프트웨어가 들어있어 설계된 로직을 받아들이고 로직에 맞게 FPGA를 구현하기 위한 정보를 만들어 낸다. 또 로직 디버깅 중에는 화면에 DSP 내부의 모든 신호에 대한 파형을 볼 수 있게 한다. 그림 8은 ASIC 에뮬레이터를 이용한 실험 세트이며, 그림 9는 DSP를 정상적으로 동작시켰을 때 DSP 내부의 각종 신호들을 워크스테이션 화면에 나타낸 것이다. 설계된 DSP는 원래 16 슬롯 동안에 16개의 악기음을 합성해 낼 수 있도록 되어 있는 데, 샘플링 레이트가 44.1KHz인 CD급의 음질을 합성하기 위해서는 22MHz의 시스템 클럭으로 칩이 동작해야 한다. 그러나 실제로 DSP를 하드웨어적으로 구현하는 ASIC 에뮬레이터의 동작 속도가 최대 10MHz 정도이므로, 한 프레임을 2개의 슬롯으로만 동작 시키고 CD급의 음질을 유지하도록 했다. 이를 위한 시스템 클럭 주파수는 2.8MHz이다.

합성된 음을 주관적인 평가에 의하여 원하는 음임을

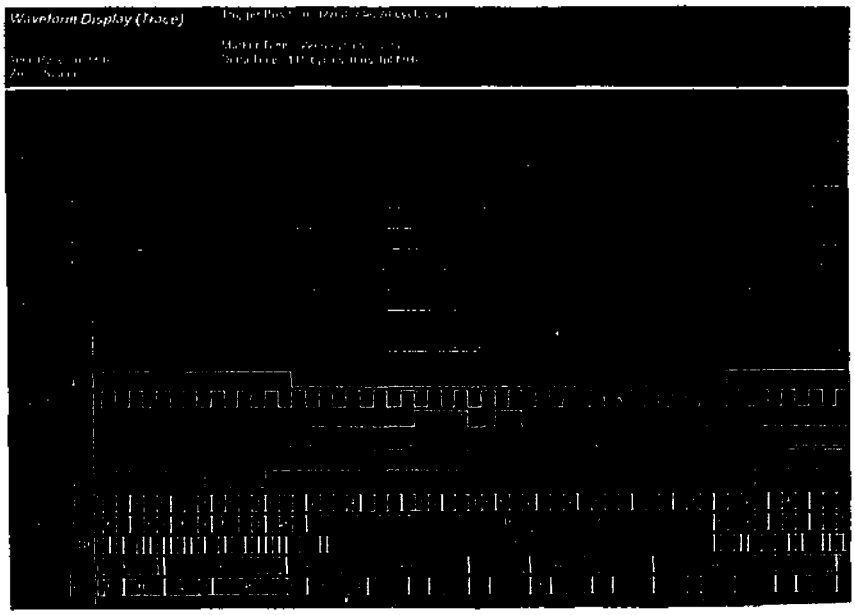


그림 9. 본 논문에서 설계한 DSP 내부의 각종 신호 파형
Fig 9. The internal signal waveform of the DSP designed in this paper

확인한 후, 합성한 악기음과 시중에서 구입한 악기원음의 샘플 데이타를 이용하여 그림 10과 같이 시간 영역과 주파수 영역에서 대해서 비교하였다. 이 때 합성한 악기음은 하프시코드의 A5음이다. 그림에서 전체 파형에 대한 포락선과 스펙트럼상에서의 포락선의 모양은 서로 유사하며, 첫번째와 두번째 포만트의 위치가 잘 맞음을 알 수 있다. 전체 포락선에서 악기 음색을 결정짓는 가장 중요한 부분은 Attack이다. 원래 악기음에서의 Attack 부분은 풍부한 동적 스펙트럼 특성을 가지며 포락선 모양도 복잡하다. 따라서 FM에 의해서 그대로 재현하는 것은 거의 불가능하지만 Sustain에 이르러서는 원래의 악기음과 비슷하다. 청취 실험에 의하면 이러한 점은 더욱 분명해 진다. 청취실험 결과 본 논문에서 설계한 DSP를 이용하여 합성한 FM 합성음이 원음과 거의 비슷함을 확인할 수 있었다. 다른 음 높이의 음과 다른 악기음들에 대해서도 같은 과정을 거쳐서 합성하면 된다. 오보에, 바순, 잉글리쉬 호른 등 목관 악기음들은 스펙트럼 분포와 포락선 모양이 단순하므로 파라메터의 추출이 비교적용이하며 합성음도 원음과 거의 같다. 그 중에서 하프시코드 음을 합성하는 것이 배우 까다로왔다. 한 음에서 다른

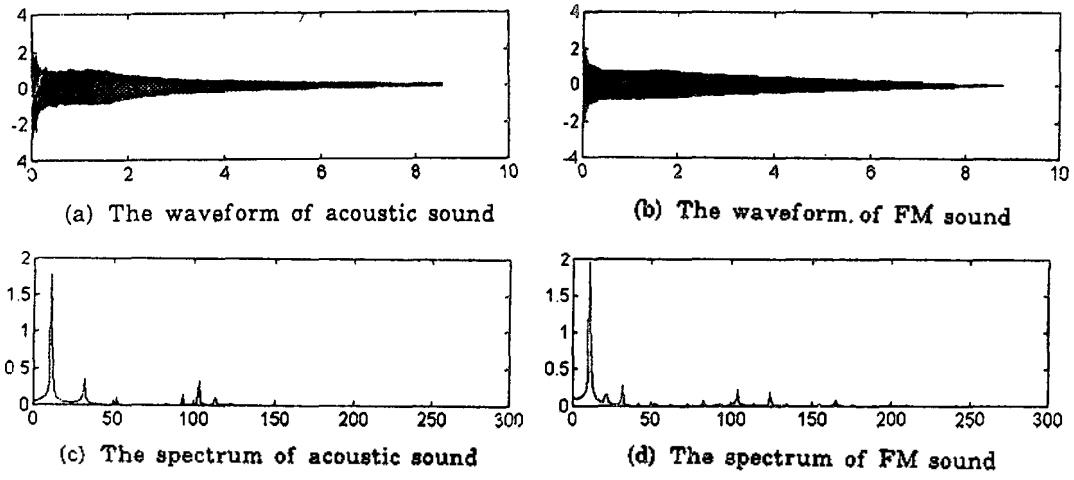


그림 10. 하프시코드에 대한 FM 합성음과 음향 악기음의 비교
Fig 10. Comparison the characteristics of the FM synthesis sound with those of the acoustic sound for harpsichord

음으로 변환하는 과정은 Decim-ation과 Interpolation을 이용한다. 이들 여러 악기에 대한 데이타들을 마이크로프로세서의 프로그램 ROM에 저장했다가 MIDI 신호가 수신되면 해당 악기에 대한 데이타 테이블을 참조하여 알고리듬 파라메터를 만들어서 DSP에 전송하여 DSP에서 악기음을 합성한다. 따라서 악기음을 합성하는 데 있어서 가장 중요한 관건은 각 악기에 대한 파라메터들을 어떻게 잘 추출하느냐에 달려있다. 또한 DSP 내부에 저장되어 있는 악기음 합성 알고리듬을 다양하게 변형하면 각 악기 특성에 맞는 음도 합성할 수 있다. 설계한 DSP는 0.8μm 게이트 어레이를 이용하여 place&routing해서 delay 시뮬레이션을 해 본 결과 최악의 조건에서 시간이 가장 많이 걸리는 부분인 RAM 엑세스가 44 ns이하였다. 따라서 설계한 DSP는 CD급 음질을 유지하면서 16개 악기음을 동시에 합성할 수 있음을 알 수 있다. 여기서 곱셈기와 ALU를 거치는 데이타 패스 부분이 가장 시간이 많이 걸리는 부분임에도 불구하고 RAM 엑세스 타임만을 고려하는 이유는 RAM 엑세스는 한 명령어 사이클(44 ns) 이내에 이루어져야만 하지만 데이타 패스는 3 사이클 내에 이루어져도 되도록 합성 프로그램을 만들었기 때문이다.

V. 결 론

본 논문에서는 2 캐리어 FM을 이용한 악기음 합성 알고리듬을 수행할 수 있는 DSP를 설계하였다. 합성에 필요한 변수를 추출하여 ASIC 에뮬레이터를 이용하여 DSP를 하드웨어적으로 구현하여 하프시코드, 오보에, 바순, 잉글리쉬 호른 음을 합성해서 청취해 본 결과 자연음과 아주 비슷함을 확인 했다. 그리고 합성 파형을 자연음과 시간 영역과 주파수 영역에서 비교해 보아도 유사한 특성을 갖는 것을 알 수 있었다. 이러한 확인과정은 ASIC 에뮬레이터의 동작속도 때문에 2 채널로 한정했다. 그러나 이러한 개념의 DSP를 0.8μm CMOS 게이트 어레이를 이용하여 설계할 경우는 CD급 음질을 유지하면서 16개 악기음을 동시에 합성할 수 있음을 VLSI 설계툴을 이용하여 레이아웃한 후 시뮬레이션을 통하여 확인 했으며 총 게이트 수는 3만 게이트 정도 되었다.

본 논문에서는 2 캐리어 알고리듬을 이용하여 일부 악기음에 대한 파라메터를 추출하여 악기음을 합성했지만, 여러가지 다른 악기음에 대한 파라메터를 추출한 후 다양한 알고리듬을 구성하여 여러 악기음을 합성해내는 방법에 대해서 연구되어야 한다. 그리고 FM 방식으로 보다 양질의 악기음을 합성하기 위해서는 3 캐리어 합성 알고리듬과 악기음의 변수 추출에 관한 연구가 있어야 할 것이다. 이러한 연구 결과가 하드웨어적으로 구현될 때 본 논문에서 설계된 DSP의 개념도 유용하게 이용되리라 생각한다.

감사의 글

본 연구는 94년도 한국과학재단 연구비 지원에 의한 결과임 (과제번호 : 94-0100-12-01-03)

참 고 문 헌

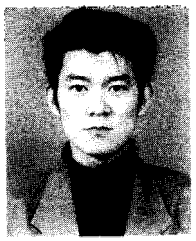
1. Oslon H. F., H. Belar, and J. Timmens, Electronic music systems, J. Acoust. Soc., Am., Vol. 32, No. 3, 1960.
2. Grey J. M., An exploration of musical timbre, Rep. STAN-M-2, Dept. of Music, Stanford Univ., 1975.
3. Roads C., A Tutorial on nonlinear distortion or wave-shaping synthesis, Computer Music J., Vol. 3, No. 2, 1979.
4. Chowning J. M., The synthesis of complex audio spectra by means of frequency modulation, J. Audio Eng. Soc., Vol. 21, No. 7, 1973.
5. 박주성, 김형순 외, 전자악기용 디지탈 신호처리 VLSI 개발, 과학기술처, 1993.
6. Thomas D. Rossing., The Sience of Sound, Addison-Wesley Publishing Company, 1990.
7. Risset, Jean-Claude, Computer Study of Trumpet Tones, Murray Hill, N. J., Bell Telephone Laboratories, 1966.
8. Charles Dodge, Thomas A. Jerse, Computer Music : Synthesis, Composition, and Performence, Schirmer Books, 1985.
9. Dexter Morrill, Trumpet Algorithms for Computer Composition, Computer Music Journal, 1 (1), pp. 46-52, 1977.
10. Benade, Arthur H., Fundamentals of Musical Acoustics, New York : Oxford University Press, pp. 66. 1976.
11. 주준열, 이문형 외, FM 방식의 디지탈 악기음 합성을 위한 소프트웨어 시뮬레이터 및 파라미터 추출알고리즘 개발, 전자공학 논문회지 제 31권 B편 제3호, pp. 225-238. 1994.
12. J. P. Palman, P. Palman, A. Ronveaux, A Method of Generating and Controlling Musical Asymmetrical Spectra, J. Audio Eng. Soc., Vol. 36, No. 9, pp. 671-685, September. 1988.
13. F. Richard Moore, Elements of Computer Music, Prentice Hall, 1990.
14. Hal Chamberlin, Musical Applications of Microprocessors, Hayden Book Company, 1985.
15. Richard J. Higgins, Digital Signal Processing in VLSI, Prentice Hall, 1990.
16. A. Horner, J. Beauchamp, L. Haken, Machine Tongues XVI : Genetic Algorithms and Their Application to FM Matching Synthesis, Computer Music Journal, 17 : 4, pp. 17-29, Winter 1993.
17. 박주성, 장호근, 하드웨어 에뮬레이션 기술, 대한 전자공학회지, 제 22권, 제 10호, 1995.

▲권 민 도(Min Do Kwon) 1969년 4월 9일생
1992년 2월 : 부산대학교 전자공학과(공학사)
1994년 2월 : 부산대학교 전자공학과(공학석사)
1994년 3월~현재 : 부산대학교 대학원 전자공학과 박사과정 재학

▲장 호 근(Ho Keun Jang) 1968년 1월 15일생

1993년 2월 : 부산대학교 전자공학과 (공학사)
1995년 2월 : 부산대학교 전자공학과 (공학석사)
1995년 3월~현재 : 부산대학교 대학원 전자공학과 박사과정 재학

▲김 재 용(Jae Yong Kim) 1971년 4월 16일생



1994년 2월 : 부산대학교 전자공학과 (공학사)
1994년 3월~현재 : 부산대학교 대학원 전자공학과 석사과정 재학

▲김 형 순(Hyung Soon Kim) 1960년 8월 21일생
1983년 2월 : 서울대학교 전자공학과 졸업(공학사)
1984년 2월 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사)
1989년 2월 : 한국과학기술원 전기 및 전자공학과 졸업(공학박사)
1987년 1월~1992년 6월 : (주) 디지콤 부설 정보통신 연구소 연구부장
1992년 7월~1995년 3월 : 부산대학교 전자공학과 전임강사
1995년 4월~현재 : 부산대학교 전자공학과 조교수
부산대학교 정보통신연구소 연구원

▲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 : 한국전자기술 연구소
1985년 8월~1989년 7월 : University of Florida, Ph.D.
1989년 8월~1991년 3월 : 한국전자통신연구소 책임연구원 (연구위원)
1991년 3월~현재 : 부산대학교 전자공학과 부교수

▲윤 병 우(Pyung Woo Youn)
1981년 2월 : 부산대학교 전자공학과 졸업(공학사)
1989년 2월 : 부산대학교 전자공학과 졸업(공학석사)
1992년 2월 : 부산대학교 전자공학과 졸업(공학박사)
1993년 5월~1995년 2월 : 한국전자통신연구소 선임연구원
1995년 3월~현재 : 부산 경성대학교 전기공학과 전임강사
※주관심분야 : 적응신호처리, ASIC

▲백 광 렬(Kwang Ryul Baek) 1961년 3월 25일생
1984년 2월 : 부산대학교 전자공학과 졸업(공학사)
1986년 2월 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사)
1989년 8월 : 한국과학기술원 전기 및 전자공학과 졸업(공학박사)
1989년 8월~1994년 2월 : (주) 터보테크 기술연구소 소장
1994년 3월~1995년 현재 : 부산대학교 전자공학과 전임강사

▲임 창 헌(Chang Hun Ieam)
1986년 2월 : 서울대학교 전자공학과 졸업(공학사)
1988년 8월 : 한국과학기술원 전기 및 전자공학과 졸업(공학석사)
1993년 8월 : 한국과학기술원 전기 및 전자공학과 졸업(공학박사)
1994년 3월~현재 : 부산수산대학교 전자공학과 전임강사