기술해설

DSP를 이용한 음성 및 오디오 시스템 설계

김성수*, 조성호**
(*한양대 대학원 전자공학과 석사과정, **한양대 공대 전자공학과 조교수)

1. 서 론

최근 그 활용도가 많아진 DSP (digital signal processor)란 디지탈 신호를 고속이면서 효율적으로 처리하기 위한 프로세서이다. 일반적인 범용 CPU (central processing unit) 프로세서와는 달리, DSP는 고속의 연산 기능을 제공하면서도 가격이 저렴해 다양한 분야의 실시간 시스템 구현을 가능하게 해 주고 있다. 한가지 예로, 현재 컴퓨터를 이용한 멀티미디어 산업이 매우 빠른 속도로 발전하고 있는데 이 변화의 핵심에는 바로 DSP가 있다고 해도 지나침이 없다.

현재 DSP는 음성 및 오디오 신호처리 시스템, 디지털 통신 시스템, 제어 시스템, 영상처리 시스템 등 많은 영역에 걸쳐 성공적으로 사용되고 있다. 몇가지 대표적인 활용분야를 살펴보면, 음성신호 압축 분야 [1-4], MPEG (moving picture expert group)과 같은 오디오신호 압축분야 [5, 6], 그리고 디지털 통신 시스템에서의 적응 반향제거기, 적응 등화기, 채널간섭 제거, 변복조기, 채널 코딩, 암호화기 [7-14] 등에서도 DSP가 사용되고 있다. 그리고 수중 음향 신호처리[15], 디지털 필터 디자인, 전력 스펙트럼 추정, 수중 음향 신호처리 같은 디지털 신호처리 분야[16-23]와 적응 신호처리[24-26], 이외에도 능동 소음 제어기 및 적응 제어기와 같은 제어 시스템 [27]에도 유용하게 이용되고 있다. 또한 영상 압축, 디지털 방송, 의료기기 등과 같은 영상처리 분야[28-32] 및 그밖의 많은 분야에서 DSP의 활용은 점점 커져 가고 있는 추세이다.

2. 일반적인 DSP 구조 및 특성

범용 DSP는 크게 부동소수점 (floating point) 방식과 고정소수점 (fixed point) 방식의 두가지 종류로 나뉘어 진다. 부동소수점 방식은 DSP내부에 실수 연산기능이 있어서 실수 연산에 관한 모든 것을 DSP 자체가 제공을 한다. 그러므로 프로그램 작성은 쉬워지지만 그 만큼 DSP의 하드웨어 구조가 복잡해지므로 가격이 높아지는 단점을 가지고 있다. 반면에 고정소수점 방식은 실수 연산 기능을 DSP가 지원을 하지 않는다. 따라서 연산에 관한 내용을 프로그래머가 하나하나 신경을 써서 해결해 주어야 올바른 실수연산이 가능하다. 따라서 부동소수점 방식 보다 프로그램 작성이 더 어려워 지는 반면, 구조가 간단하여 가격이 비교적 저렴하고 동작속도가 높다.

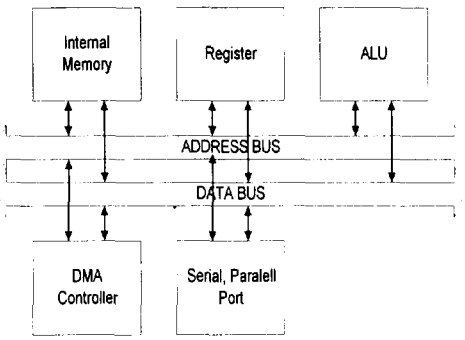

그림 1. 범용 DSP의 기본 구조

DSP를 생산하는 업체는 대표적으로 Texas Instruments, Motorolla, Analog Devices, AT&T, Advanced Micro Devices, NEC 등 매우 많으며, 각 회사별로 다양한 제품을 만들어 내고 있다. 이들 제품은 광범위한 특성과 여러 가지 성능을 제공하고 있으므로, 시스템 설계자는 설계 규격에 잘 맞고 개발이 용이한 Chip을 선정하여 사용하면 되겠다. 그럼 현재 널리 쓰이고 있는 범용 DSP의 주요 특징을 간략하게 살펴보자. (그림 1)은 일반적인 범용 DSP 구조를 개괄적으로 나타낸 것이다.

내부 구조를 잠깐 살펴보면, 우선 연산기인 ALU, 정밀한 연산을 위한 내부 레지스터, 내부 메모리(RAM 또는 ROM), DMA(Direct Memory Access) controller, 그리고 외부와의 통신을 위한 parallel port 또는 serial port로 이루어져 있다. 이밖에도 캐쉬(cache), 인터럽트(interupt) 콘트롤러 등이 있으나, DSP의 특징이라고 할 수 있는 것들만 나타내었다.

최근의 DSP는 내부 구조에 파이프라인 구조를 취하고

있다. 파이프라인 구조란 한가지 명령어를 여러 단계에 걸쳐 동시에 수행할 때 동시에 다음 단계를 위한 준비를 하는 것으로서, 프로세서가 고속의 연산 능력을 내는데 매우 중요한 역할을 한다. 프로세서는 일반적으로 한 개의 명령어를 수행하는데 몇가지 단계를 거친다. 우선 메모리에서 명령을 가지고 와서 어떠한 명령어인지 해독하고, 그 명령을 행하는데 필요한 데이터를 읽어서 실행을 하게 된다. 파이프라인 구조가 되어 있으면, 현재의 일을 하면서 동시에 다음에 예정된 일을 미리 준비하게 된다. 예를 들면, 우선 처음에 명령어를 읽어 들이고 다음 어떠한 명령이었는지 해독을 하게 되는데, 이때에 그 다음에 실행할 명령을 메모리로 부터 읽어 들여 그 이후 해독할 상황에 대비하게 된다.

이러한 파이프라인 구조는 매우 획기적인 구조로 프로세서의 성능을 극대화 시킬 수 있다. 하지만 언제나 이러한 파이프라인 구조가 제대로 행하여 지는 것은 아니다. 몇가지 이유로 이러한 구조가 실패하는 경우가 가끔씩 발생하게 되는데, 가능한 한 이러한 연속된 연산이 지속되도록 프로그램을 작성하는 것이 매우 중요하다.

최근의 DSP가 제공하는 기능 가운데 중요한 또하나는 명령어의 병렬처리 기능이다. 보통의 DSP가 기준이 되는 1 clock 동안 1개의 명령어를 실행하는데 반해, 이 기능은 1 clock 동안 2개의 명령어를 동시에 처리 할 수 있는 기능이다. 모든 명령어가 이렇게 병렬 처리 기능을 갖는 것은 아니지만, 프로그램 작성시 이를 잘 이용하면 좋은 효과를 기대할 수 있다.

또한 DSP에는 DMA 기능이 일반적으로 탑재되어 있다. 이는 프로세서가 명령수행과는 별도로 독립적으로 메모리를 액세스를 할 수 있도록 하는 기능이다. 실제로 DSP가 연산을 수행하는데 있어서 가장 시간이 많이 걸리는 부분 가운데 하나가 메모리 액세스 부분이다. DMA를 적절히 사용하면 이러한 메모리 액세스에 필요한 절대 시간을 현격히 줄일 수 있다.

DSP가 내부에 메모리를 탑재하고 있는 것이 있다. 예를 들면, Texas Instrument사의 TMS320C3x 계열의 DSP에는 1K word에서 2K word의 내부 RAM이 탑재되어 있고, TMS320C54x계열의 DSP에는 32k word의 내부 메모리가 준비되어 있다. 이는 매우 중요한 장점의 하나이다. DSP가 내부 메모리를 사용하게 되면 외부 메모리를 사용하는 것보다 약 2배의 속도를 줄일 수 있다. 더우기 내부 메모리가 크면 프로그램과 프로그램 동작에 필요한 메모리를 내부 메모리만 이용하여 처리를 할 수 있으므로 DSP를 최적의 속도로 동작을 시킬수 있고, 또한 시스템 하드웨어의 실제 크기를 최소화 할 수 있다는 장점이 있다.

3. DSP를 이용한 개발 환경

DSP를 사용하여 임의의 시스템을 개발하기 위한 하드웨어 및 소프트웨어 개발 흐름도는 (그림 2)와 같다.

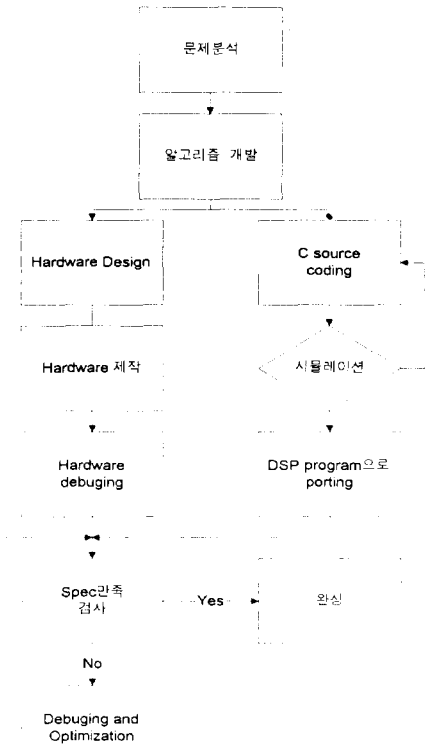

그림 2. DSP 시스템 개발 흐름도

우선 주어진 문제를 분석 한다. 시스템을 구성하는데 있어서 주어진 규격을 파악하고 하드웨어 및 소프트웨어의 성능과 필요한 기능 등을 설정한다. 이를 위해 구현하고자 하는 시스템의 DSP 구동 알고리즘을 개발한다. 구현하고자 하는 알고리즘은 주어진 성능과 규격을 최적의 상태로 만족할 수 있도록 개발한다. DSP시스템의 실시간 동작을 위해서는 알고리즘 차원에서의 최적화 과정이 매우 중요하다. 알고리즘이 개발된 다음에는 소프트웨어 개발 작업과 하드웨어 개발 작업이 본격적으로 나뉘어 수행된다.

먼저 소프트웨어 개발 부분을 살펴 보면, 개발된 알고리즘을 우선 프로그램 언어로 구현하는 것이다. 보통 소프트웨어 개발에 사용되는 프로그래밍 언어는 C 언어나 asembly 언어이다. 시뮬레이션 단계에서는 보통 C 언어를 사용한다. 다른 컴퓨터 언어 즉, PASCAL이나 FORTRAN으로도 하는 경우도 있기는 있으나 target DSP용 cross-compiler가 있지 않으면 불가능하고 또 더 최적화된 성능을 위해서는 C 언어를 사용하는 것이 여러 측면에서 유리하다. 다음은 완성된 프로그램을 가지고 시뮬레이션을 하는 것이다. 보통 이때는 evaluation board라 불리는 것을 가지고 하게 된다. 이 eavaluation board에는 우리가 사용하려는 DSP가 탑재 되어 있다. 주로 PC나 workstation base로 작성된 프로그램을 target DSP 전용 compiler를 가지고 compile한 후에 이 evaluation board에서 시뮬레이션을 하게 된다. 시뮬레이션 결과가 제대로 나오지 않으면 다시 알고리즘 수정을 하거나 프로그

램을 좀 더 optimize또는 디버깅해서 시뮬레이션 결과가 제대로 나와야 한다. 시뮬레이션을 통한 알고리즘 검증이 끝나면 실제 하드웨어에 탑재할 프로그램으로 바꾸는 작업이 필요하다. 이 작업은 간단할 수도 있고 매우 복잡할 수도 있다. 이 과정에서 주로 실시간 운영을 위해서 assembly 작업을 하여 optimization을 하게 된다.

하드웨어 개발을 보면 주어진 스펙에 따라 하드웨어 디자인을 한다. 그리고 디자인을 가지고 제작을 하고 제작된 하드웨어를 1차적으로 기본적인 디버깅을 하게 된다.
이렇게 해서 하드웨어와 소프트웨어 개발이 되면 실제로 소프트웨어를 하드웨어에 탑재하여 테스트를 하게 된다. 이때 스펙을 만족하면 비로소 시스템 개발이 완료 되는 것이나, 주어진 스펙을 만족 시키지 못한다면 디버깅이나 최적화를 반복해야 한다. 디버깅시에는 emulation board라 불리는 시스템을 사용하게 된다. 이 또한 PC나 workstation base로 운용되는데 target system의 DSP와 그 메모리 상황을 손쉽게 알아볼 수 있어 디버깅에 많은 도움이 되어 개발 기간을 단축시킬 수 있다. 시중에 다양한 종류의 DSP를 탑재한 evaluation board와 다양한 DSP를 지원하는 emulation board가 나와 있으니 사용하고자 하는 DSP에 해당하는 제품을 선택하여 사용하면 된다.

4. 범용 DSP 응용

일반적으로 DSP를 사용하는 회로는 (그림 3)과 같은 구조를 가지게 된다.

입력되어지는 아날로그 신호는 어떠한 신호라도 상관이 없다. LPF(Low Pass Filter) 앞에는 어플리케이션에 따라 다른 부가적인 회로가 올 수 있다. Low Pass Filter는 샘플링시에 생기는 aliasing을 피하기 위해 입력 신호의 bandwidth를 어느 선에서 정하기 위해서 사용한다. ADC와 DSP간의 동작 속도를 완화하기 위해서 버퍼를 사용한다. 일반적으로 DSP는 고속으로 동작을 하고 ADC는 상대적으로 동작속도가 느리다. DSP가 ADC로부터 나오는 sampled data를 읽기위해 기다리는 것은 실시간 동작에 있어서 치명적인 자원 낭비이다. 실시간 동작(realtime processing)을 위해서는 버퍼를 사용하여 프로세서가 쓸데없이 프로세싱 시간을 낭비하지 않게 디자인을 하는 것이 가장 중요하다. 출력 버퍼 역시 같은 이유로 사용한다. DSP는 ADC로부터 샘플된 데이터를 가지고 실제로 원하는 일을 처리하게 된다. 처리된 데이터는 다시 DAC를 거쳐 아날로그 신호로 바뀌게 된다. DAC에서 나오는 출력 신호는 'staircase' waveform 즉 계단 형태의 모양을 가지는데 이를 부드럽게 하기 위해서 다시 LPF를 사용하여 완전한 아날로그 신호로 변환시킨다.

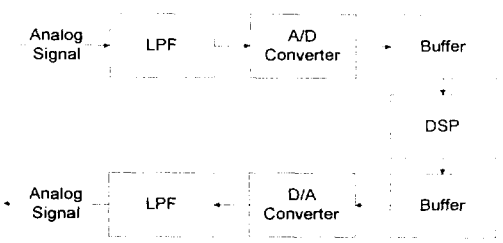

그림 3. 일반적인 DSP 회로 블럭도

DSP를 사용하는 어플리케이션은 거의 대부분 실시간 처리(realtime processing)를 목적으로 하기 때문에 DSP의 올바른 선정이 무엇보다 중요하다. 필요한 퍼포먼스를 내지 못하는 DSP를 선택을 하여서도 곤란하고 반대로 필요한 퍼포먼스이상의 성능을 내는 DSP의 선택은 경제적으로 낭비이다. 또한 일단 DSP를 선정했으면 그 DSP의 기능을 자세히, 정확하게 파악할 필요가 있다. DSP가 제공하는 기능을 적절하고 정확하게 사용을 하면 DSP가 가지고 있는 성능을 최대한으로 낼수 있다. 이를 위해서는 assembly언어로 프로그램을 코딩을 하는 것이 필수적이다.

4.1 범용 DSP를 사용한 음성 신호 처리

범용 DSP가 많이 쓰이고 있는 분야 중의 하나가 음성 신호(speech signal) 처리 분야이다. (그림 4)가 일반적인 음성 신호 처리를 위한 DSP 회로 디자인이다. 일반적인 인간의 음성이 주로 3KHz 이내라고 볼 때 샘플링 주파수는 8KHz 정도면 충분하다. 또 보통은 8 bit로 양자화를 한다. 좀더 정밀도를 높이고 싶으면 9 bit정도로 양자화 하여 최하위 비트를 버리는 방법도 간혹 사용된다. AD conveter를 사용하는 분야는 다 그렇겠지만 시스템의 성능은 AD converting에 의해서 크게 영향을 받는다. AD converting시에 최대한 잡음(noise)이 없도록 세심한 주의를 기울여 시스템을 디자인 해야 한다.

(그림 4)를 보면 입력되는 신호가 AD converter에 들어가기 전에 low pass filter를 거친다. 샘플링된 데이터들은 버퍼에 저장되어 지고 DSP는 버퍼에서 데이터를 읽어다가 일을 하게 된다. 처리된 데이터들은 출력 버퍼에 저장되어 지고 이 데이터들은 DA converter를 거쳐 다시 아날로그 신호로 바뀌거나 아니면 application circuit에 들어가 저장

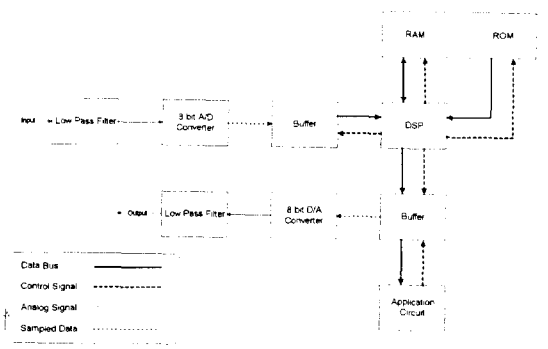

그림 4. 음성 처리 시스템 설계 예

되어 지거나 또 다른 프로세싱을 거치게 된다. application circuit 부분은 용도에 따라 달라 7질수 있으며 없을 수도 있다. 그림을 자세히 보면 회로를 크게 두가지 파트로 나눌 수 있음을 알 수 있을 것이다. 아날로그 파트와 디자탈 파트이다. 보통의 DSP 응용이 다 이러한 구조로 되어 있다. 아날로그 파트를 설계할 때는 잡음(noise)에 특히 신경을 써야 한다. 잡음은 시스템 전체의 성능에 영향을 미칠 수 있기 때문이다.

AD converting 회로를 설계할 때에는 입력 되어 지는 신호의 특성을 면밀히 검토해야 한다. 입력되는 신호가 unipolar signal인지 아니면 bipolar signal인지를 파악해야 하고, 입력되는 신호의 크기(amplitude)와 주파수(frequency)를 고려해야 한다. 이 예에서는 음성신호를 처리 하는 것이기 때문에 샘플링 주파수는 8KHz로 사용하였다. 하지만 입력 신호의 amplitude는 어플리케이션 마다 다르기 때문에 AD converter의 특성을 살펴서 입력 신호의 amplitude의 range와 AD converter의 input range가 맞도록 해야 한다. 이를 위해서는 증폭기(amplifier)가 주로 사용된다. 또 입력 신호가 unipolar인지 bipolar인지에 따라 증폭기 구조와 AD converter를 선정이 달라진다. 보통 요즘의 AD converter는 unipolar, bipolar 두 신호를 모두 입력 받을 수 있기 때문에 크게 염려하지 않아도 된다. 참고로 unipolar, bipolar에 대해서 설명을 하면, 어떠한 신호의 amplitude range(peak-to-peak voltage)가 5V라고 하면 unipolar signal이라면 0V ~ 5V사이의 signal이고 bipolar signal은 -2.5V ~ 2.5V 사이의 signal을 말하는 것이다.

디지털 파트 디자인에 신경을 써야 하는 것은 control signal 부분이다. 디지털 회로이기 때문에 잡음(noise)는 크게 신경을 쓰지 않아도 되나, 간혹 ground bounce등이 생길 수도 있기 때문에 그냥 지나 칠수는 없다. 역점을 두고 신경을 써야 할 부분이 control signal timing이다. control signal에 따라서 data가 data bus를 통해 움직이는데 데이터를 주고 받는 디바이스간에 타이밍이 맞지 않으면 데이터를 읽거나 쓰는데 잘못된 결과를 초래하게 된다. 문제가 생기면 찾기도 힘들고 고치기도 힘든 부분이 바로 이러한 부분이다. 음성 처리용 DSP 시스템은 처리할 데이터양이 비교적 적은 편이라 시스템이 그리 복잡하지 않은 편이다. 음성 신호 시스템의 예는 디지털 통신시에 사용되는 음성 coding, decoding 시스템이 있다.

4.2 오디오 신호 처리

오디오 신호 처리을 위한 DSP 시스템은 (그림 5)과 같다. 오디오 신호를 샘플링 하기 위해서 left channel, right channel을 위해 2개의 A/D Converter를 사용한다. 오디오 신호는 샘플링시에 보통 16 bit, 44.1KHz나 48KHz로 샘플링을 한다. 그림에서 보면 A/D converter 2개, D/A converter 2개, 그리고 각각을 위해 buffer 4개가 사용된다. 그리고 음성 처리 시스템과는 달리 demultiplexer가 있다. 이는 DSP에서 나

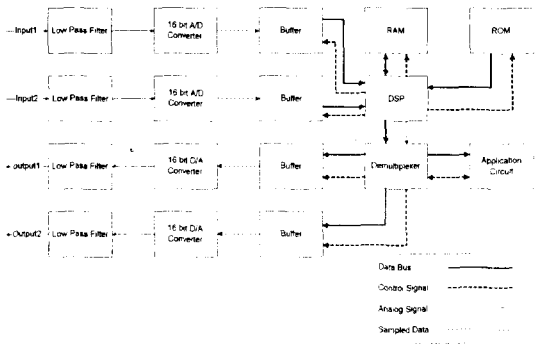

그림 5. 오디오 처리 시스템 설계 예

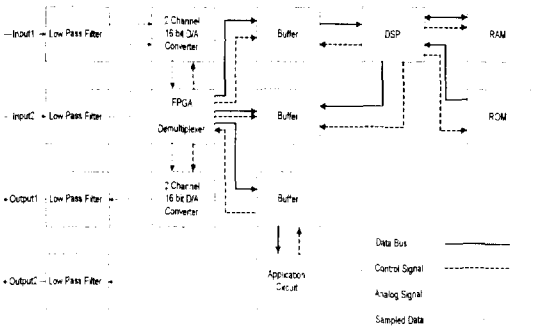

그림 6. FPGA를 이용한 오디오 처리 시스템 설계 예

오는 데이터가 보통 하나의 bit stream으로 되어 있기 때문에 다시 2개의 데이터열 즉 left channel, right channel 데이터를 분리하기 위하여 사용되어 진다. 그러나 없을 수도 있다. 그러나 어찌하던 D/A converter를 거쳐 다시 아날로그 신호로 복원을 하려면 demultiplex과정을 거쳐야 한다. 그래서 따로 이 demultiplexer를 두지 않은 경우에는 보통 DSP에서 demultiplex를 하여 left channel buffer, right channel buffer에 데이터를 따로 저장하게 된다. 이경우에는 DSP에 부하가 추가로 생기게 되므로 이를 감수 하여야 한다. 요즘엔 이러한 demultiplex같은 부분을 FPGA(File Programmable Gate Array)를 사용하여 외부에서 따로 수행을 하여 DSP에 부하를 주지 않는 방향으로 보통 시스템 디자인을 한다.

FPGA를 사용하는 경우에는 (그림 5)보다 더 간단하게 시스템을 디자인 할 수가 있다. (그림 6)가 그 예이다. (그림 6)를 보면 multiplexer, demultiplexer부분이 있는데 이 부분이 바로 FPGA이다. 또한 각 디바이스의 콘트롤을 FPGA를 이용하여 수행하면 전체 시스템도 컴팩트 해지고 디버깅도 훨씬 용이하다. A/D converter는 16 bit 2 channel 오디오 전용 A/D converter를 사용한다. 이는 한 A/D converter가 2 channel을 동시에 sampling을 할 수가 있으므로 편리하고 디자인이 간단해 진다. 2 channel A/D converter에서 나오는 데이터를 FPGA가 읽어서 각 channel의 데이터를 두 개의 buffer에 나누어 저장한다. 그리고 DSP는 두 개의 buffer에 저장된 두 channel의 데이터를 읽어서 처리를 한다음 하나의

bit stream으로 출력 buffer에 다시 저장하게 된다. FPGA는 이 출력 buffer에 저장된 하나의 bit stream을 각 channel에 해당하는 데이터로 나누어 이를 2 channel D/A converter에 보내어 다시 아날로그 신호로 변환시키게 된다. 이처럼 FPGA를 쓰면 시스템이 간단해 지고 좀더 flexible하게 된다. FPGA는 Feild Programmable Gate Array의 약어로 자유롭게 프로그래밍이 가능한 디바이스이다. 즉 디바이스의 기능을 프로그램이나 기타 다른 툴로 자유롭게 정의할 수 있다는 말이다. 이는 시스템 디자이너에게 매우 큰 이점을 준다. 우선 시스템을 좀더 간단하게 만들 수 있다. 모든 TTL IC와 왠만한 기능을 넣을 수 있기 때문에 시스템이 간단해지고 따라서 시스템의 신뢰성이 높아진다. 또한 프로그래밍이 가능한 디바이스이므로 시스템 디버깅이 간단해 진다. 이는 개발 기간을 단축하여 주므로 경제적으로 시스템 개발을 가능하게 하여준다. FPGA는 보통 VHLD(Very high speed Hardware Description Language)나 Verilog HDL(Hardware Description Language)로 프로그램된다. 보통 Synopsys나 Maxplus2 같은 툴을 이용하여 FPGA를 프로그램하고 디버깅을 한다. 이밖에도 현재 여러 가지 PLD(Programming Logic Device)들이 시중에 나와있다. 가장 적합한 것을 골라 적절히 사용함으로서 시스템 디자인, 개발을 좀더 효율적으로 할 수 있을 것이다.

5. 결 론

지금까지 범용 DSP를 이용하여 음성 신호 처리, 오디오 신호처리, FPGA를 이용한 디자인에 대해서 살펴 보았다. 예를 보면 알겠지만 그 구조는 (그림 3)의 일반적인 DSP 회로 구성과 별로 다르지 않다는 것을 알 것이다. 범용 DSP를 이용한 회로 디자인에서 가장 중요한 것은 사용하고자 하는 DSP의 효율적인 이용이다. DSP의 기능을 제대로 알고 그 기능을 가장 잘 사용하는 것이 무엇보다 중요하다. DSP의 기능을 잘 사용하는 것은 DSP에 맞게 회로를 디자인 하는 것도 중요하지만 프로그램을 작성할 때 DSP의 기능을 잘 이용하는 것도 중요하다.

참 고 문 헌

[1] M. Kondoz, *Digital speech - Coding for low bit rate communications systems*, John Wiley&Sons, 1994

[2] John R. Deller Jr., John G. Proakis and John H. L. Hansen, *Discrete-time processing of speech signals*, Macmillan publishing, 1993

[3] Sadaoki Furui, M. Mohan Sondi, *Advances in speech signal processing*, Dekker, 1992

[4] L.R. Rabiner and R. W. Schafer, *Digital processing of speech signals*, Prentice-Hall, 1978

[5] Ken Stelglitz, *A digital signal processing primer with applications to digital audio and computr music*, Addison Wesley, 1996

[6] The art of digital audio

[7] DR. Kamilo Feher, *Wireless digital communications - Modulation and spread spectrum applications*, Prentice-Hall, 1995

[8] Shu Lin and Daniel J. Costello Jr., *Error control coding*, Prentice-Hall, 1983

[9] Lynn Schafer Gross, *Telecommunications - An introduction to electronic media*, Brwon & Benchmark, 1995

[10] John G. Proakis, *Digital communications*, McGraw-Hill, 1995

[11] Bernard Sklar, *Digital communications - Fundamentals and applications*, Prentice-Hall, 1988

[12] John G. Proakis and Masoud Salehi, *Communication system engineering*, Prentice-Hall, 1994

[13] Edward A. Lee and David G. Messerschmitt, *Digital Communication*, Kluwer Academic Publishers, 1988

[14] N. S. Jayant and Peter Noll, *Digital coding of waveforms - Priciples and applications to speech and video*, Prentice-Hall, 1984

[15] Williams S. Burdic, *Underwater acoustic system analysis*, Prentice-Hall, 1991

[16] Sophocles J. Orfanidis, *Introduction to signal processing*, Prentice-Hall, 1994

[17] Oktay Alkin, *Digital signal processing - A laboratory approach using PC-DSP*, Prentice-Hall, 1994

[18] Alan Kamas and Edward A. Lee, *Digital signal processing experiments using s personal computer with software provided*, Prentice-Hall, 1989

[19] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-time signal processing*, Prentice-Hall, 1989

[20] Paul A. Lynn and Wolfgang Fuerst, *Introductory digital signal processing with computer applications*, Wiley, 1994

[21] Leland B. Jackson, *Digital filters and signal processing*, Kluwer academic, 1989

[22] Jae S. Lim and Alan V. Oppenheim, *Advanced topics in signal processing*, Prentice-Hall, 1988

[23] Richard J. Higgins, *Digital signal processing in VLSI*, Prentice-Hall, 1990

[24] Bernard Widraw and Samuel D. Sterns, *Adaptive signal Processing*, Prentice-Hall, 1985

[25] Victor Solo and Xuan Kong, *Adaptive signal processing algoithms - Stability and perfomance*, Prentice-Hall, 1995

[26] Simon Hakin, *Adaptive filter theory*, Prentice-Hall, 1996

[27] Graham C. Goodwin and Kwai Sang Sin, *Adaptive filtering prediction and control*, Prentice-Hall, 1984

[28] Khalid Sayood, *Introduction to data compression*, Morgan Kaufman Publishers, 1996

[29] Anil K. Jain, *Fundamentals of digital image processing*, Prentice-Hall, 1989

[30] Willis J. Tompkins, *Biomedical digital signal processing - C language examples and laboratory experiments for the IBM PC*, Prentice-Hall, 1993

[31] Ioannis Pitas, *Digital image processing algorithms*, Prentice-Hall, 1993

[32] Keith Jack, *Video Demystified - A handbook for the digital engineer*, Hightext, 1996

[33] *TMS320C3X User's guide*, Texas Instruments, 1994

[34] *DSP56000/DSP56001 Digital Signal Processor User's Manual*, Motorolar, 1990
[35] *DSP56116 Digital Signal Processor User's Manual*, Motorola, 1990
[36] *DSP96002 IEEE Floating-Point Dual-Port Processor User's Manual*, Motorola, 1989
[37] *DSP56301 24-bit Digital Signal Processor User's Manual*, Motorola, 1995
[38] *TMS320C5X User's Guide*, Texas Instruments, 1997
[39] *TMS320C54X User's Guide*, Texas Instruments, 1995
[40] 박귀태, 이상락, *C언어로 쉽게 쓰는 DSP TMS320C31*, 대영사, 1994
[41] *VLSI signal processing*, IEEE Press, 1984
[42] B. A. Bowen and W. R. Brown, *VLSI systems design for digital signal processing*, Prentice-Hall
[43] Mohamed El-Sharkawy, *Signal processing, image processing and graphics applications with Motorola's DSP96002 processor*, Prentice-Hall, 1994

저 자 소 개

김성수(金成洙)

1970년 9월 29일생. 1996년 2월 한양대 공대 전자공학과 졸업. 1996년 3월 ~ 현재 한양대 대학원 전자공학과 석사과정 재학. 관심분야 신호처리, DSP 응용, 디지털 통신, 멀티미디어 시스템

조성호(趙誠鎬)

1960년 2월 21일생. 1982년 2월 한양대 공대 전자공학과 졸업. 1984년 12월 Univ. of Iowa 전기 및 컴퓨터공학과 졸업(석사). 1989년 8월 Univ. of Utah 전기 및 컴퓨터공학과 졸업(공박). 1989년 8월~92년 8월 한국전자통신연구소 부호기술부 선임연구원. 1992년 9월~현재 한양대 공대 전자공학과 조교수. 관심분야 : 디지털통신, 무선통신, 신호처리, 정보통신 시스템.