특 집

# SOC 구현 기술 동향

정 세 웅, 임 경 묵
삼성전자 System LSI사업부 SOC연구소 Home Platform팀

## I. 서 론

최근 들어 시스템 LSI(혹은 비메모리 반도체)의 가장 큰 경향의 하나는 시스템온칩(System On Chip 또는 SOC)이다. SOC라는 용어는 넓은 의미에서 시스템레벨에서 보드상에서 구현되던, 여러 컴포넌트를 단일 칩으로 구현하는 기술 및 그 결과로서의 고집적 회로를 의미한다. SOC의 기술적 근간은 Deep Sub-Micron(DSM) 공정의 발달과 소프트웨어-하드웨어 동시 설계 기술을 포함한 수백만 수천만 게이트를 포함할 수 있는 설계 기술의 발전으로 볼 수 있다. DSM 공정에서는 그 이전 공정에서는 (0.25 $\mu$m 급 이하 공정) 실제로 고려하지 않아도 될 정도의 미미한 물리적 현상등이 두드러지게 나타나게 된다. 예를 들어 배선간의 상호 간섭 효과(cross coupling), 디바이스의 비선형적 특성 등은 SOC 설계자들에게 폭발적 집적도의 증가로 인한 복잡성의 체계적 이해뿐만 아니라, 회로의 DSM 물리적 특성까지 이해해야 하게 만들고 있다. 이러한 DSM과 연관되어진 SOC의 설계는 설계 방법론(Design Methodology)과 검증 방법론(Verification Methodology)의 환경 자체를 바꾸도록 만들고 있다. 설계자 관점에서 이러한 환경의 변화는 크게 아래의 세가지로 나타난다.

- 계층적 설계/검증. 특히 하드웨어-소프트웨어 동시설계/동시검증(Hardware-Software Co-Design/Co-Verification)
- 회로 설계 시 Layout의 고려. 즉 Link to Physical Layout
- 검증된 IP(Intellectual Property)의 재사용

사실은 이러한 환경 변화는 계층적 레이아웃 설계 툴, Physical Compiler(PC)등의 CAD 소프트웨어의 등장으로 더욱 두드러지고 있다.

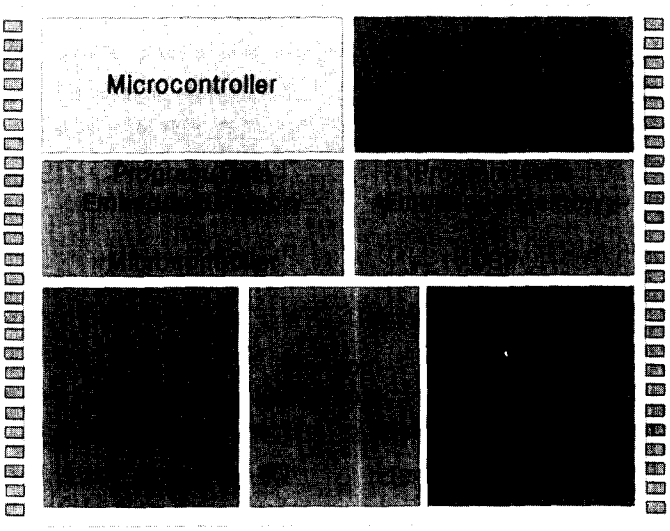
계층적 설계는 설계의 복잡성이 폭발적으로 늘어나는 점을 고려할 때 너무나 당연한 현상으로 볼 수 있다. 계층적 설계는 논리 설계 또는 그 이상의 상위 단계에서부터 전체 설계를 계층별로 작은 블록으로 나누어서 진행하고, 이러한 계층적 구조를 레이아웃 단계까지 유지하는 것을 의미한다. 이렇게 함으로써 전체 설계의 복잡도를 단순화하여, 설계 오류를 줄이고, 설계 기간을 단축 할 수 있다. 이러한 계층적 설계의 초점 중 하나는 이미 검증된 프로그래머블 코어(Programmable Core)를 중심으로 한 하드웨어-소프트웨어 동시설계/동시검증이다. 대부분의 SOC는 프로그래머블 코어가 집적되고 있고, 이에 따라 해당 프로그래머블 코어에서 수행되는 소프트웨어가 SOC에 임베드되거나, 오프칩(off-chip) 메모리로 제공되게 된다. 프로그래머블 코어를 이용한 SOC 설계는 기능의 상당 부분을 소프트웨어로 전가함으로써, SOC 설계의 유연성을 담보하고, 기능 추가나 변경에 따른 설계 변경의 부담을 하드웨어 수정 없이 소프트웨어만 변경하여 대응이 가능하므로, 매우 매력적인 설계 방법이다. 두 번째로 레이아웃을 고려한 설계는 설계 단계별 간의 상호 정보를 공유하여 논리 설계와 레이아웃 설계 단계 간의 반복 수행을 줄이는데 그 목적이 있다. 예를 들어 0.25 $\mu$m급 이하 공정에

서의 설계는 논리 설계 단계와 레이아웃 설계 단계가 거의 독립적으로 이루어 졌고, 레이아웃 설계 단계에서 독립적으로 수행된 배치 배선 정보를 논리 설계 단계에 반영하여 타이밍 검증을 하는 것이 통상적인 방법이었다. 이렇게 하더라도 논리 설계와 레이아웃 설계 단계 간의 반복 수행이(Iterations for Timing Closure) 3~4 정도면 레이아웃을 완성할 수(즉, sign-off) 있었다. 그러나, DSM에서는 논리 설계 단계에서 사용되는 배선 지연 모델(Wire Load Model)이 실제 배치 배선으로부터의 지연 시간과 차이가 현저하여, 논리 설계 단계와 레이아웃 설계를 독립적으로 수행할 경우 두 단계간의 반복 수행이 현저하게 늘어나게 되어 레이아웃을 주어진 시간 안에 완성하는 것이 불가능할 경우가 자주 있게 된다. 이러한 문제를 해결하기 위해 당사를 포함한 많은 회사가 물리 합성(Physical Compiler 또는 PC) 툴을 사용한다. 논리 합성 단계에서 대략적인 배치 정보(Floor Plan)를 이용하여 실제 레이아웃의 배치 배선 결과를 잘 반영할 수 있는 설계배선 지연 모델(Wire Load Model)을 도출하고, 논리 설계 후 레이아웃 설계의 배치 배선이 끝난 후 실제 배치 배선 정보를 반영한 논리 최적화를 PC가 수행한다. 마지막으로 SOC에서는 집적도가 늘어남에 따른 다양한 설계 블록들이 단일 칩에 포함되게 되는데, 설계 기간의 단축 및 설계 오류를 방지하기 위해서는 이미 실리콘에서 증명된 설계 IP(Intellectual Property)의 사용이 필수적이다. 이러한 증명된 IP의 재사용은 정형화된 형식의 설계 데이터 뿐만 아니라 테스트 벡터의 재사용을 위한 설계 가이드라인을 포함한 시스템의 구축을 필요로 하게 된다.

본 논문에서는 당사에서 개발된 MP3 압축/재생복원 및 정지 화상 복원을 가능케 하는 SOC C-PAD(CalmRISCTM for Portable Audio Devices)를 중심으로 SOC설계에서 MCU와 DSP의 통합 설계에 대해서 자세히 소개하도록 한다.

## II. MCU와 DSP의 통합 설계

현재 SOC라고 주장하는 대부분의 칩은 시스템 콘트롤을 담당하는 마이크로콘트롤러(Mciro-controller)와 디지털 신호 처리를 담당하는 프로그래머블 DSP코어(Digital Signal Processor Core)를 포함하고 있다[1]. 서론에서 이미




〈그림 1〉 전형적인 SOC의 예

언급한 바와 같이 이러한 경향을 두드러지게 하는 이유는 고성능 DSM 공정을 이용한 수백만 또는 수천만 게이트에 이르는 SOC 설계가 게이트 용량 면에서 별도의 마이크로콘트롤러를 포함하는데 문제가 없을 뿐더러 고성능의 프로그래머블 DSP 코어를 중심으로 신호 처리 기능의 일부 또는 대부분을 소프트웨어로 처리하는 것이 가능하게 되었기 때문이다. 〈그림 1〉은 마이크로콘트롤러와 프로그래머블 DSP코어를 포함하고 있는 전형적인 SOC의 블록 다이어그램을 나타내고 있다.

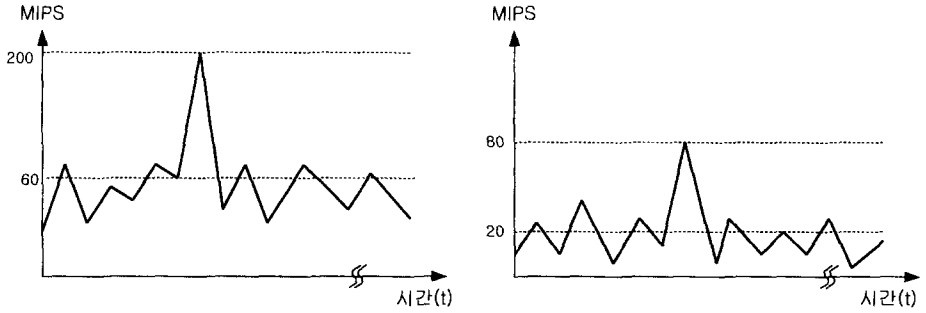
〈그림 1〉의 예와 같은 SOC를 설계하는 설계자가 현실적으로 부딪히는 문제는 어떠한 마이크로콘트롤러와 프로그래머블 DSP코어를 사용하는 것이 가장 효율적인가 하는 문제로부터 출발하여, 어떠한 식으로 시스템 콘트롤 태스크 및 신호 처리 태스크를 나누어서 각각의 코어에 할당할 것이며, 두 코어를 어떠한 식으로 연결할 것이냐 하는 것이다.

목표로 하는 SOC가 지원해야 하는 응용이 실시간 요구가 있는 태스크(즉, 신호 처리 태스크)와 그렇지 않은 태스크(시스템 콘트롤 태스크)가 혼재 되어 있을 때, 태스크 할당 이전에 결정해야 할 사항은 어떠한 프로그래머블 코어를 사용할 것인가 하는 점이다. 이러한 SOC의 설계에 있어서 취할 수 있는 프로그래머블 코어의 선정에는 크게 세 가지 방법이 있다.

- 별개의 마이크로콘트롤러와 DSP코어를 사용하는 방법
- 마이크로콘트롤러에 적합한 명령어와 DSP 명령어를 같이 갖고 있는 단일 코어를 사용하는 방법
- 마이크로콘트롤러와 DSP기능에 적합한 보조 DSP 프로세서(DSP Co-Processor)를 사용하는 방법

첫번째 방법의 장점은 각각의 독립된 코어를 임베드 해서 SOC를 구성하면 되므로 전체 칩 통합구성이 쉽다는 점이다. 그러나, 두 코어 사이에 중복된 블록이 존재하므로 하드웨어 측면에서 잉여 부분이 있을 수 있다는 단점이 있다. 또한 무엇보다도 별개의 코어에 대한 소프트웨어를 별개의 개발환경을 이용하여 개발해야 하므로 각 코어의 소프트웨어 자체 간에 더나아가서 개발자 간에 밀접한 인터페이스가 필요로 하게 된다. 예를 들어 마이크로콘트롤러에서 수행되는 소프트웨어 태스크와 DSP코어에서 수행되는 소프트웨어 태스크간의 동기(Synchronization)를 맞추고, 데드락을 없애야 하는 등의 인터페이스에 오버헤드와 어려움이 있게 된다. 두번째 방법은 단일 코어를 사용하므로 태스크 할당의 문제가 없고, 소프트웨어 개발 환경 측면에서도 가장 바람직한 방법이다[2]. (왜냐면 단일 코어이므로) 그러나 신호 처리 태스크가 추가가 된다든지 등의 이유로 DSP의 성능이 현재 수준 보다 더 요구된다면 다른 성능이 좋은 코어를 사용하게 되는 최악의 경우가 발생하게 되어 기존의 검증된 소프트웨어를 바꾸어야 하는 상황이 벌어질 가능성이 생기게 된다. 세번째 방법은 앞선 두 방법의 절충안으로 생각할 수 있는 방법으로 호스트 코어로서의 마이크로콘트롤러는 시스템 콘트롤 태스크를(실시간 요구가 없는) 담당하고, 실시간 요구가 있는 DSP 태스크는 보조 DSP 코어(DSP Co-Processor)가 담당하게 된다. 이 경우 DSP 태스크의 추가로 인한 성능 개선 요구는 보조 DSP 코어에만 반영을 하면 된다. (즉, 호스트 마이크로콘트롤러는 수정이 없게 되고) 또한, 수동적인 보조 DSP 코어의 경우 프로그램 인출이(Program Fetch) 호스트 마이크로콘트롤러에 의해서만 이루어지므로 단일 개발 환경을 유지할 수 있는 장점이 있다. 그러나, 두 코어간의 통신 효율성을 증가시키기 위해 호스트 마이크로콘트롤러와 보조 DSP 코어의 효율적인 인터페이스가 전제가 되어야 하고, 개발환경 측면에서 다양한 보조 DSP 지원이 가능하도록 준비가 되어있어야 한다.

단일 코어가 아닌 방법에서는 (첫번째와 세번째 방법) 어떤 태스크를 어떤 프로세서 코어에 할당할 것인지를 정해야 하는데, 이러한 결정의




(a) 태스크 1
평균요구 MIPS : 60MIPS
최대요구 MIPS : 200MIPS

(b) 태스크 2
평균요구 MIPS : 20MIPS
최대요구 MIPS : 80MIPS

〈그림 2〉 평균요구 MIPS와 최대요구 MIPS

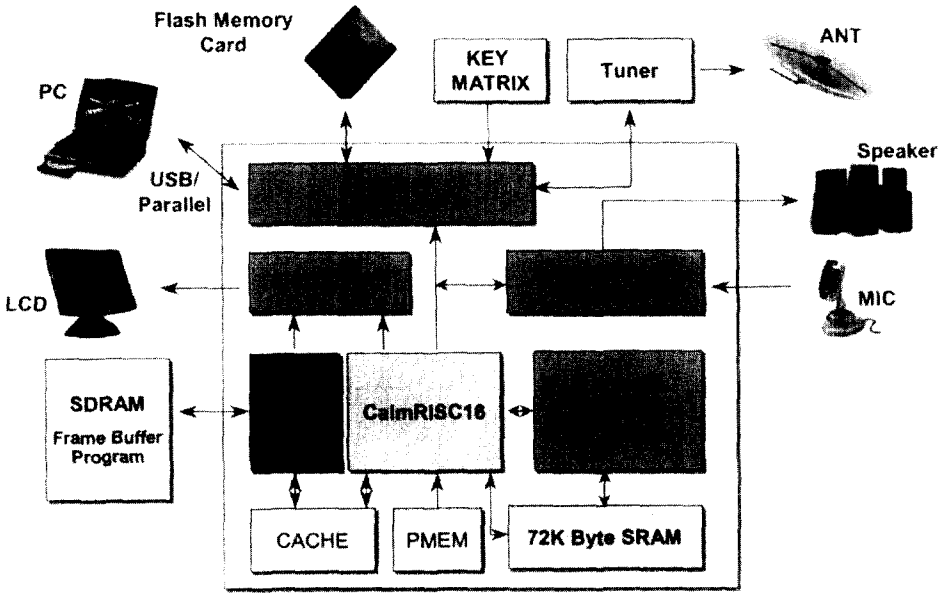
중요한 판단 기준에는 각 태스크 별로 필요 성능 및 칩의 크기/전력소모가(Chip Size/Power Budget) 포함 된다. 예를 들어 사용하고자 하는 마이크로콘트롤러의 성능이 신호 처리 기능을 담당할 정도로 충분하고, 전력 소모의 요구 조건을 만족한다면 별도의 DSP코어를 사용치 않고, 마이크로콘트롤러가 직접 신호 처리 기능을 담당하는 것도 가능하다. 마이크로콘트롤러의 성능과 DSP코어의 성능을 해당 SOC가 목표로 하는 태스크의 요구 성능(주로 MIPS치로 계산. MIPS= Million Instructions Per Second)과 연관 지을 때 중요한 고려 사항은 시스템 콘트롤 태스크의 경우는 평균적인 MIPS치로 계산이 되어야 하고, 신호 처리 태스크의 경우는 실시간 요구를 (Real-Time Requirement) 만족하기 위해서는 최대 MIPS치로 계산이 되어져야 한다는 점이다. 예를 들어 어떤 태스크의 시간에 따른 MIPS 요구량이 〈그림 2〉와 같다면 실시간 요구가 있을 경우와 그렇지 않을 경우에 따라 태스크 할당 방법이 다르게 된다. 즉, 〈그림 2-(a)〉에서 실시간 요구가 있는 경우라면 최대 MIPS요구량인 200MIPS를 감당할 수 있는 프로그래머블 코어가 (마이크로콘트롤러이던가 DSP코어이던가) 필요하게 된다.

〈그림 2〉의 태스크 1과 2를 해당 SOC에서 동시에 지원을 해야 하고, 사용 가능한 마이크로콘트롤러의 성능과 DSP코어의 성능이 각각 최대 100MIPS와 210MIPS라고 가정하자. 만약 태스크 1은 실시간 요구가 필요한 태스크이고, 태스크 2는 실시간 요구가 필요없는 태스크라고 하면, 태스크 1은 210MIPS DSP코어에서 태스크 2는 100MIPS 마이크로콘트롤러에서 처리하도록 하면 된다. 그러나, 만약 두 태스크 모두가 실시간 요구가 필요가 없다면 태스크 1과 2를 모두 100MIPS의 마이크로콘트롤러에서 처리 가능하게 되어, 별도의 DSP코어가 필요없게 된다. 이처럼 태스크의 성질 및 개수에 따라 현명한 태스크 할당이 전체 SOC의 경쟁력을 결정할 수 있는 중요한 요소가 된다.

다음 절에서 예로 들어 기술한 C-PAD(CalmRISC for Portable Audio Devices) SOC는 호스트 마이크로콘트롤러와 보조 DSP코어를(세 번째 방법) 기본 플랫폼으로 택한 경우이다.

## III. 설계 사례 CPAD(CalmRISC™ for Portable Audio Devices)

고성능 오디오 응용 칩인 C-PAD는 MP3, AAC, WMA를 포함한 다양한 형태로 압축된 음악을 디코딩할 수 있으며 실시간 MP3 압축, 음장 효과 등 고급의 오디오 처리가 가능하다. 최대 동작 주파수는 80MHz로 높은 성능을 이용하




〈그림 3〉 C-PAD의 구성도

여 오디오 뿐만 아니라 JPEG등의 압축 이미지를 (320×240의 경우 초당 약 10장) 디코딩할 수 있으며 내장된 LCD controller를 통해 디코딩된 영상을 디스플레이할 수 있다. 지원되는 주변 장치로는 USB, Smart Card, 2 채널 오디오 codec, IIC, UART, parallel port, SIO 등이 있으며 SDRAM을 포함하여 최대 32M 바이트까지 외부 메모리를 장착할 수 있다. 〈그림 3〉은 C-PAD의 전체 구성도를 보이고 있다. 16비트 MCU인 CalmRISC16은 외부 인터페이스를 포함하는 시스템 태스크를 주로 수행하고 24-bit 고정 소수점 보조 DSP인 MAC2424는 오디오 및 정지영상의 신호처리를 담당한다[3].

MAC2424는 수동적인 보조 DSP로 모든 명령어들은 CalmRISC16에 의해 프로그램 메모리로부터 읽혀진다. CalmRISC16과 MAC2424는 파이프라인이 서로 동기되어 있어 데이터 메모리 접근 시 충돌이 없고 보조 DSP 명령 수행 시 성능손실이 없다. 데이터 메모리는 두 프로세서가 공유를 하며 CalmRISC16은 16비트 접근을 하고 MAC2424는 두개의 24비트 동시 접근이 가능한 구조를 가지고 있다[4].

C-PAD의 SOC 설계시 가장 중요한 고려사항으로 효율적인 소프트웨어의 수행과 개발에 두었다. 구조적인 측면에서 C-PAD의 가장 큰 특징은 프로그래머블 MCU/DSP를 기반으로 하는 SOC라는 것으로 거의 모든 신호처리를 소프트웨어로 수행하기 때문에 하드웨어로 신호처리를 수행하는 방식에 비해 매우 다양한 응용에 적용 가능하다는 장점을 가지고 있다. 전체 칩의 성능은 장착된 소프트웨어가 얼마나 효율적으로 수행하느냐에 달려있으며 최적화된 프로그램의 개발 또한 매우 중요한 사항이기 때문에 효율적인 소프트웨어의 수행과 개발이 가장 우선시 되었다.

신호처리 알고리즘의 효율적인 수행을 위해서 칩 내부에 16K 바이트의 프로그램 SRAM과 72K 바이트의 데이터 SRAM을 내장하여 내부 메모리만을 사용하여 신호처리가 가능하게 하였고 외부 메모리와의 데이터 교환을 위해 이중버퍼(double buffer)를 지원하는 DMA를 (Direct Memory Access) 두었다. 내부 메모리 대신 캐쉬를 사용하는 경우 데이터의 집약성(locality)이 떨어지는 신호처리 알고리즘의 특성 때문에 성능저하가 발생되고 특히 실시간 제한 조건을

만족하기 위한 최대요구 MIPS 값이 매우 커지게 되어 신호처리 알고리즘의 성능에 치명적인 영향을 미치게 된다. 평균요구 MIPS 값은 내부 메모리를 사용하는 경우에 비해 많은 차이를 보이지는 않기 때문에 실시간 요구가 필요하지 않은 태스크의 경우에는 캐쉬를 사용하는 것이 적합하다.

시스템 소프트웨어의 효율적인 수행을 위해서 지연시간(latency)이 매우 짧은(1 싸이클) 인터럽트를 지원한다. 일반적으로 수십 종의 인터럽트가 있는 경우 별도의 하드웨어 지원이 없이 발생된 인터럽트의 종류를 판단하고 이를 처리하는 프로그램으로 가기 위해서는 많게는 수십 싸이클까지 필요하며 C-PAD와 같이 실시간으로 동작해야 하는 경우 인터럽트 지연시간은 매우 중요한 의미를 가진다.

신호처리 알고리즘과 시스템 소프트웨어의 개발을 위해서는 소프트웨어의 동작을 제어하고 처리과정을 자세히 분석할 수 있는 개발환경이 필요하다. CalmRISC16과 MAC2424는 소프트웨어 개발이 편리한 구조를 가지고 있으며 내장된 디버그 블럭(CalmBreaker)은 이러한 구조를 이용하여 디버그 기능을 지원한다. 외부 PC에서 동작하는 사용자 인터페이스와 연계하여 C-PAD는 효율적이고 강력한 개발환경을 제공한다. 또한 하드웨어를 개발하기 전에 먼저 PC상에서 동작하는 CalmRISC16과 MAC2424의 시뮬레이션 환경 및 컴파일러, 어셈블러, 링커 등을 포함하는 개발 환경을 소프트웨어 개발자에게 제공함으로써 하드웨어와 소프트웨어가 동시에 개발을 시작하여 개발 및 검증 시간을 단축할 수 있다.

C-PAD는 SOC로 인하여 칩 크기가 크기 때문에 동작 주파수를 80MHz(0.35$\mu$ 공정, 3.3 V)까지 올리기 위해서는 레이아웃(Layout) 단계 뿐만 아니라 설계 초기 단계에서 구조적인 고려가 필요하다. 기본적으로 사용된 방법은 성능에 영향이 적은 임계경로(critical path)는 그 처리에 하나 또는 그 이상의 싸이클을 더 배정하여 임계경로에서 제외시키는 것으로 레이아웃시 성능에 영향이 큰 임계 경로에 집중을 하도록 하였다. 이런 방법을 사용한 예로 주변 회로(peripheral)와의 인터페이스가 있다. 주변회로와의 인터페이스는 기본적으로 2 싸이클 인터페이스로 하였고 필요한 경우 여러 싸이클에 걸쳐서 동작된다. 또 다른 예로는 프로그램 메모리 접근방식이 있다. 앞에서 설명한 바와 같이 칩 내부에 프로그램 SRAM이 있어 매 싸이클마다 현재 접근하려고 하는 곳이 내부 프로그램 SRAM 영역인가 아니면 캐쉬를 통한 외부 메모리 영역인가를 판단하여야 하는데 이 동작은 임계경로 상에 있는 동작이다. C-PAD에서는 판단 결과를 보지 않고 바로 전에 접근한 곳을 다시 접근한다고 가정하고 프로그램을 접근한 후 만일 판단 결과가 그렇지 않다고 나온 경우 한 싸이클 후에 정확한 메모리 영역을 접근한다. 데이터 메모리에 연속적으로 읽기와 쓰기를 반복할 때 메모리 구조상 임계경로가 발생하게 되는데 이 때는 write buffer를 사용하여 pipelined 동작을 수행함으로써 임계경로를 제거하였다. 이러한 방법의 사용으로 인한 성능저하는 1% 미만이고 최대 동작 주파수는 20% 이상 증가한 것으로 평가된다.

## 참 고 문 헌

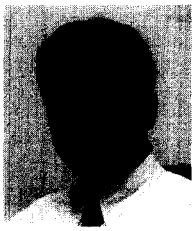

[1] Jennifer Eyre *et al*, "DSP Processors Hit the Mainstream", *IEEE Computer*, Aug., 1998, pp51-59.

[2] Michael Dolle *et al*, "A 32-b RISC/DSP Microprocessor with Reduced Complexity", *IEEE JSSC*, Vol. 32, No. 7, July 1997, pp1056-1065.

[3] C-PAD III CalmRISC16 for Portable Audio Device User's Manual, Samsung Electronics, 2001.

[4] K. M. Lim and S. W. Jeong, "CalmRISC : a low power microcontroller with efficient coprocessor interface", *Microprocessors and Microsystems*, 25, pp.247-261, Aug, 2001.

## 저 자 소 개



**鄭 世 雄**

1962년 10월 10일생 1985년 2월 서울대학교 공과대학 전자공학과 학사, 1987년 2월 한국 과학 기술원 전기 및 전자공학과 석사, 1992년 6월 University of Colorado at Boulder, Electrical & Computer Eng. Ph.D., 1985년 2월~1988년 11월 : C & B Technology, 1992년 5월~1992년 12월 : Motorola, 1993년 2월~현재 : 삼성전자 System LSI사업부, 〈주관심 분야 : DSP/MCU Architecture, Multi-Media Architecture, MPEG SOC's for DVD Players and Digital TV's〉

**林 慶 默**

1966년 5월 2일생 1989년 2월 서울대학교 물리학과 학사, 1991년 2월 한국 과학 기술원 전산학과 석사, 2002년 2월 한국 과학 기술원 전산학과 박사, 1997년 2월~현재 : 삼성전자 System LSI사업부, 〈주관심 분야 : DSP/MCU Architecture, MPEG SOC's for DVD Players and Digital TV's〉