論文 98-23-8-27

# 네트워크 기반 객체 지향형 영상 처리를 위한 MPEG 디코더 코어 설계

정회원 박 주 현*, 김 영 민*

# Design of Core of MPEG Decoder for Object-Oriented Video on Network

Ju-Hyun Park*, Young-Min Kim* *Regular Members*

요 약

본 논문은 네트워크를 기반으로 한 객체 지향형 영상 처리를 하는 프로그램이 가능한 MPEG 디코더 설계를 다룬다. 설계된 MPEG 디코더는 객체 지향형 프로그램을 지원할 수 있도록 스택 버퍼를 이용한 콘트롤러를 내장하고 있어서 객체에 기반한 영상 처리에 효과적이며, 소프트웨어 지향적인 영상 표준에 적용되도록 다양한 포맷의 입력 데이터 처리가 가능하다. 또한 벡터 연산부에서는 MPEG-4의 반화소 단위 처리와 고급 모드 보상(Compensation), 예측(Prediction)이 가능하며, SA(Shape Adaptive)-IDCT 가 가능하다. 또한 벡터 처리기 내에 절대값기, 반감기를 두어 인코더로 확장할 수 있도록 하였다. 설계 및 검증은 0.6㎛ 5-Volt CMOS TLM(Three Layer Metal) COMPASS 라이브러리를 이용하였다.

ABSTRACT

This paper concerns a design of programmable MPEG decoder for video processing by object unit on network. The decoder can process video data effectively by a embedded controller with stack buffers for supporting OOP (Object-Oriented Programming). The controller offers extended instructions that process several data types including 32bit integer type. In addition to that, we have a vector processor, in this decoder that can execute advanced compensation and prediction by half pixel and SA(Shape Adaptive)-IDCT of MPEG-4. Absolutors and halfers in the vector processor make this architecture extensive to a encoder. We verified the decoder with 0.6㎛ 5-Volt CMOS COMPASS library.

## Ⅰ. 서 론

인터넷을 이용한 영상, 음성 등 다양한 서비스 수요의 증가는 플랫폼간의 정보 접근을 위해 새로운 표준의 필요성을 제기하였으며, 웹 브라우저는 모든 컴퓨팅 플랫폼에서 지원하는 범용 GUI로 자리를 잡아가고 있다. 이와 함께 인터넷 표준들은 다양한 응용에 적용할 수 있는 내장형 정보 접근 장치들에 널리 수용될 것으로 기대된다.

* 전남대학교 전자공학과
論文番號 : 98143-0326
接受日字 : 1998年 3月 26日

인터넷의 서비스 범주가 된 영상, 음성 데이터들은 종래의 마이크와 카메라에 의한 자연 영상, 음성의 단순한 녹음, 녹화 뿐만 아니라 컴퓨터 그래픽스 기술을 이용한 인공 데이터의 보편화에 따라 다른 형태를 갖는 데이터들 간의 자유로운 상호 편집이 가능한 기술 표준이 필요하게 되었다.[1]

이와 같이 인터넷을 이용한 서비스의 성숙과 영상 정보에 대한 패러다임의 변화로 영상 편집 및 생성, 전송, 접근을 포함하는 통합 표준 환경이 필요하게 되었으며, 이는 MPEG-4 표준화를 출범시키는 계기가 되었다. MPEG-4는 개방형 시스템을 취하고 있는데 하드웨어, 소프트웨어로 처리하던 디코더 시스템이 점차 플랫폼에 독립적인 시스템으로 설계되고 있고, 특정 알고리즘만을 실현하던 방법에서 벗어나 라이브러리 툴 박스와 같은 유연한 시스템을 목표로 하고 있다. 또한 기존의 화면 기반 부호화와는 달리, 객체(object)를 기반으로 한 부호화를 지향하며, 표준안에서도 신텍스(syntax) 기술 언어로 C++이나 자바와 같은 객체 지향적 언어를 권고하고 있듯 소프트웨어에 의한 구현을 염두에 둔 표준이기 때문에 융통성있는 디코더 구조 연구가 필요하다.

자바가 빠르게 수용된 측면에는 네트워킹 응용 환경을 구축하는 데 핵심적인 구현 도구가 되어 왔으며, 그 객체 지향적인 융통성과 이식성 때문이다. 자바는 웹 브라우징 서비스를 양방향 정보 처리 서비스로 바꾸는 수단으로써 매력적인 것이다. 자바는 원래 Sun Microsystems사에서 소형의 네트워킹 장치들에 소프트웨어를 구현하기 위한 방법으로서 개발한 것이었다.[2] 오늘날의 프로그래머들 가운데 90%이상이 C를 사용하고 있기 때문에 배우기 손쉬운 프로그래밍 환경을 제공하고, C++과 흡사하므로 쉽사리 C++로 전환할 수 있으며, 자바의 바이트 코드 해석기와 클래스 지지에 필요한 RAM은 매우 작기 때문에 비교적 "자원이 빈약한" 플랫폼에서도 기본적인 자바 기능을 지원받을 수 있는 장점을 가지고 있다.

소프트웨어적인 디코더를 지향하는 관점에서 보면, 자바 언어는 MPEG-4 디코더의 소프트웨어 관리 측면에서 대단히 매력적인 것이다. 새로운 알고리즘이 출현했을 때 서버단에서 이 알고리즘을 구현할 수 있는 자바 클래스를 다운로드함으로써 새로운 기능을 추가할 수 있는 장점도 가지고 있다. 그러나 현재의 네트워크 환경에서는 복원 툴의 실시간 동작이 어렵다. 또한 업계의 관심이 자바를 이용한 네트워크 컴퓨터에 쏠려 있기 때문에 로컬 환경은 단지 최소한의 하드웨어 세트로도 실행이 되도록 설계되었다. 따라서 이들의 로컬 처리 능력을 네트워크 대역폭 한계와 신중히 조화시킴으로써 로컬 장치의 성능이 너무 떨어지거나 혹은 값이 너무 비싸지는 일이 없도록 해야만 한다.

그러나 자바 실행을 위한 기저 하드웨어 구조들을 최적화하는 것만으로 자바가 가지고 있는 문제를 해결할 수는 없다. C 응용과 같은 다른 환경을 갖는 경우에 성능에 영향을 미칠 수 있으며, 서로 다른 응용 환경 및 성능 수준에서 구조의 연속성이 필요할 경우는 자바 프로세서를 사용하는 것이 최선이 아닐 수 있기 때문이다. 또한 많은 내장형 응용들은 기존의 기능들을 효과적으로 끌어내 자바 기반의 기능들과 통합시켜야 하지만 자바 전용 프로세서로 이식할 때는 그렇게 하기가 훨씬 더 어려워진다. 특히 영상 데이터와 같은 병렬 처리를 요하는 데이터를 스택 기반 구조로 처리하기엔 실시간 처리가 매우 어렵다. 따라서 자바 능력은 자바 기능을 갖춘 내장형 시스템에 배치해야만 하는 문제점을 갖는다.

따라서 영상을 이용한 다양한 응용을 포괄하고, 이후 추가될 자바 클래스를 수용하는 자바 기반 프로그래머블 프로세서 설계가 필요하다. 스택을 기반으로 하기 때문에 자바의 바이트 코드 해석기를 구현하기가 RISC(Reduced Instruction Set Computer)보다 훨씬 수월하다. 또한 영상 데이터를 처리해야 하기 때문에 벡터 처리가 가능한 전용 블록이 포함되어야 한다.

본 논문에서는 Ⅱ장에서 네트워크를 기반으로 한 영상 처리가 어떻게 되는지를 알아보고, Ⅲ장에서는 자바 기반 제어기를 내장한 MPEG-4 디코더 구조를 설명한다. Ⅳ장은 Ⅲ장에서 설명한 구조의 실험 결과 및 성능 평가를 하고, 마지막으로 Ⅴ장에서는 본 연구의 결과를 기술하고 향후 연구방향을 제시한다.

## Ⅱ. 네트워크 기반 영상 처리 개념

### 1. 프로토콜 계층

인터넷은 TCP/IP 프로토콜을 이용해 컴퓨터간 상호 통신을 한다. TCP/IP는 4단계의 계층으로 나뉘어진다. 그림 1은 TCP/IP 프로토콜의 4 계층을 보여주고 있다.

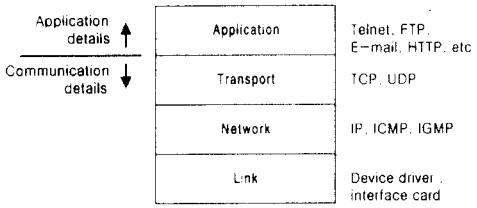



그림 1. TCP/IP 프로토콜 4 계층
Fig. 1 The four layers of the TCP/IP protocol suite

ISO(International Organization for Standardization)의 OSI(Open Systems Interconnection)모델은 더 많은 계층으로 이루어져 있으나, TCP/IP 네트워킹을 설명하는데는 4개의 계층으로 충분하다.[3] 각 계층은 패킷에 표제(header)와 종단부(trailer)를 덧붙여 캡슐화하며, 캡슐화된 정보 안에는 출발지(source), 목적지(destination) ID, 패킷 크기, 체크 섬(checksums) 등 다양한 제어 정보가 들어가 있다. 링크 계층(link layer)은 OS 커널 내에 디바이스 드라이버와 네트워크 카드를 포함하고 있기 때문에 상세하게 하드웨어를 다룬다. 네트워크 계층(Network Layer)은 라우팅을 포함해서 네트워크들간의 패킷 이동을 다룬다.[4] 전송 계층(Transport Layer)은 두 호스트간의 데이터 흐름을 제공한다. TCP(Transmission Control Protocol)[5]는 두 호스트 간 스트림의 신뢰성과 관련된 프로토콜이고, UDP (User Datagram Protocol))[5]는 단방향으로써 생산자로부터 소비자에게 영상 블록과 같은 데이터 패킷을 보내는 역할을 한다. 응용 계층(Application Layer)은 웹 서버와 웹 브라우저간의 정보 교환 방법을 정의하는 HTTP(Hyper Transfer Protocol)와 같은 공통 프로토콜을 가지고 있다.

위와 같이 네트워크를 이용한 응용은 데이터 전송율이 매우 중요하다. 만약 데이터가 중간 노드를 지나게 되면 최대 대역폭은 병목되는 지점에서 문제가 발생할 수 있기 때문이다. 따라서 망을 공유하고 있는 연결선 수, 라우팅 등에 의존하는 유효 대역폭을 변화시키는 문제는 네트워크를 통한 실시간 영상 데이터 전송에서 매우 중요하다. 대역폭의 감소를 보상하기 위해서는 전송될 정보를 감소시켜야 한다. 즉 양자화 레벨을 증가시켜서 더 많은 압축을 할 필요가 있는 것이다.

## 2. 웹 브라우저에서의 영상 솔루션

웹 브라우저에는 다양한 형태로 영상 데이터를 포함할 수 있다. 그 중 대표적인 경우가 다양한 응용을 지원할 수 있는 자바 에플릿[6]을 이용하는 것이다. 자바 프로그램은 독립형(stand-alone) 응용이나 인터넷의 어떤 호스트로부터 바이트 코드 형태로 로드되는 애플릿 형태로 실행된다. 자바 에플릿과 같은 번역된 코드를 사용하는 것은 순수 머신 코드를 실행하는 것보다 속도가 늦다. 그러나 자바는 흔히 호스트의 네이티브(native) 언어로 네트워크 입출력, 윈도윙, 영상처리 등 많은 영역의 일을 수행하는 컴파일된 자바 API(Applications Programmer's Interface)인 클래스 라이브러리와 함께 사용하기 때문에 충분한 라이브러리를 이용한다면 대부분의 수행 시간을 라이브러리 코드에서 사용하므로 번역된 코드 사용으로 인한 속도 저하를 최소화할 수 있다. 네이티브 코드는 프로그램의 일부분이므로 해석하는데 시간이 걸리지만 프로그램 실행 이전이나 실행 중에 자바 코드를 머신 코드로 해석할 수 있다면 네이티브 코드의 번역에 따른 시간 지연을 제거할 수 있을 것이다.

현재 자바 API에는 DCT, 허프만코딩, 압축 툴과 같은 영상 스트림을 처리하는 클래스가 존재하지 않는다. 물론 컴퓨터에 연결된 카메라로부터 영상을 패취하는 라이브러리 또한 존재하지 않는다. 따라서 하드웨어에 접속해야 하는 특별한 코드 라이브러리나 서버 프로그램이 필요하게 될 것이다. 이것을 해결하기 위해서는 네이티브 코드를 사용해 자바 라이브러리를 확장하거나 부족한 기능을 채우기 위해 자바 프로그램이 외부 프로그램과 통신할 수 있도록 내부 명령어를 추가해야 한다.

## 3. 자바 에플릿 클래스 구조

어떤 브라우저에서든 자바 에플릿은 브라우저 그 자체와 같은 쓰레드에서 실행되기 때문에 CPU 점유율이 높은 에플릿은 브라우저 속도를 떨어뜨리게 하고, 사용자 입력에 대한 반응을 지연시킨다. 이러한 문제를 해결하기 위해서는 에플릿 실행을 다른 쓰레드를 이용하여 하는 것이다. 그림 2는 몇 가지 영상 관련 클래스들 사이의 개념적인 표현을 하고 있다.

자바는 다양한 영상 처리를 지원하기 위한 ImageProducer와 ImageConsumer 인터페이스를 정의한다. ImageProducer를 구현한 클래스는 영상을 만들어 ImageCom-

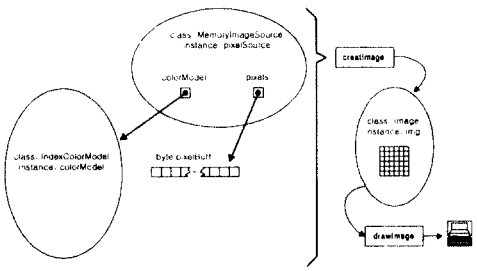



그림 2. 영상 관련 클래스들 사이의 개념적 표현
Fig. 2 Conceptual representation of the interworking of video related classes

sumer를 구현한 클래스에 전달하며, 에플릿은 네트워크로부터 블록을 수신하는 대로 내부 메모리 버퍼에 영상을 저장한다. ImageProducer를 구현한 MemoryImageSource 클래스는 ImageProducer 내의 버퍼 내용을 캡슐화하는데 사용되며, CreateImage 메소드를 이용해 자바 영상을 만들 수 있게 한다. CreateImage는 버퍼 내용을 복사하여 내부 메모리 데이터가 바뀔 때마다 새로운 영상을 만들어 낸다.

에플릿을 이용한 영상 클래스들의 개략적인 흐름도는 그림 3과 같다.

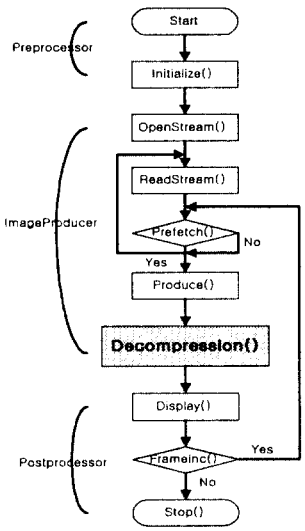



그림 3. 에플릿을 이용하는 영상 클래스들의 개략적인 흐름도
Fig. 3 Whole flow chart of video classes using Java applet [7]

Preprocessor 클래스는 버튼, 메뉴, 영상 스크린 등을 정의하고 영상 파일이 존재하는 곳의 위치등을 파악하며, 에플릿을 초기화한다. ImageProducer 클래스는 내부 메모리를 갱신할 때마다 새로운 영상을 리턴하는 클래스이다. 영상 스트림, 색깔 모델, 스트림 크기 등을 인스턴스 변수로 정의하고, 정의한 메소드 함수에 따라 영상을 처리한다. 스트림을 수신단으로 가져온 후(OpenStream), 스트림에서 바이트 단위로 영상 데이터를 읽는다(ReadStream). 또한 메모리에서 캐쉬로 여러 프레임을 미리 가져 오게 되는데(Prefetch) 입력 메모리를 채우고, 필요한 프레임이 끝날 때까지 내부 캐쉬 등의 메모리에 프레임을 가져온다. 메모리 등에 저장한 영상 데이터는 그 영상 데이터의 성질과 함께 전송되며(Produce), 이후 디코더 단에서는 복원(Decompression)과정을 거쳐 화면에 디스플레이한다(Postprocessor). 위 흐름도에서 복원 및 디스플레이 과정에 들어갈 수 있는 MPEG-4 디코더의 클래스 구조는 그림 4와 같다. [8]

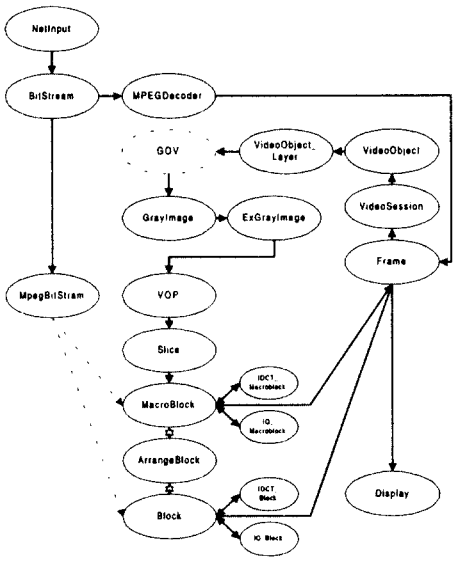



그림 4. MPEG-4 디코더의 클래스 구조
Fig. 4 Structure of MPEG-4 Decoder Classes

NetInput 클래스는 요청한 만큼의 데이터를 어레이에 채운다. BitStream 클래스는 디코더와 데이터 스트림 사이의 인터페이스 역할을 한다. MpegBitStream

클래스는 변환 데이터나 허프만 코드와 같은 비트스트림 신택스 전용 클래스이다. MPEGDecoder 클래스는 디코딩 과정을 구조화하고, 객체의 레퍼런스는 그대로 유지하며, BitStream 클래스로부터 포인터를 전달 받거나 디코딩하는 동안에 픽셀 어레이를 유지하고 있는 프레임 객체를 만든다. 또한 대부분의 MPEG 스트림에서 양자화 메트릭스와 같은 디폴트 값으로 사용하는 데이터를 보관하고 있다. 뿐만 아니라 P 타입 영상을 순방향 예측할 때 레퍼런스로 사용할 영상에 대한 포인터 등을 나타내는 함수를 제공한다. Frame 클래스는 휘도, 색도 데이터 어레이, 디스플레이 전의 마지막 픽셀 어레이를 보관하는 클래스이다. 또한 매크로블록이나 블록에서 처리한 결과를 복사하여 디스플레이 객체로 전달하는 역할을 한다. GrayImage 클래스는 단색(monochrome) 입력 시퀀스로 설계되어 있다. 예를 들면, 모양 정보 데이터는 알파 평면(alpha plane)으로 이 클래스에 포함될 수 있다. ExGrayImage 클래스는 그레이(gray) 영상에 기능성을 추가한 확장된 형태의 클래스라고 할 수 있다. 이것은 2D 필터링, 보간(interpolation) 알고리듬 등 여러 가지 기본적인 영상 처리 기능을 가지고 있으며, MC(Motion Compensation) 기능을 가지고 있다. GOV 클래스는 옵션 클래스이며, 그룹 데이터로 정의된 MPEG 스트림을 보관한다. VOP(Video Object Plane) 클래스는 YUV 형태로 영상을 읽고, 쓰며, I-VOP, B-VOP, P-VOP와 같은 더 높은 계층에 적용이 가능하도록 하는 기능을 갖도록 한다. 또한 이 클래스에서 인코더는 정해진 기준에 따라 모드를 결정한다. 또한 MPEG 스트림에서 VOP 데이터로 정의된 데이터를 보관한다. Slice 클래스는 매크로블록의 읽기와 프로세싱을 구조화한다. Macroblock 클래스는 코덱에서 가장 중요하다고 할 수 있다. 이 계층은 모든 알고리즘이 적용되는 곳이기 때문이다. 전에 사용한 벡터에 대한 레퍼런스 등의 데이터를 추적한다. 또한 MPEG 스트림에서 모든 매크로블록 표제와 벡터 정보를 읽는다. IDCTMacroBlock 클래스는 매크로블록 변환 데이터를 다루며, ArrangeBlock은 4개의 루미넌스 블록과 2개의 크로미넌스 블록 그룹을 보관한다. Block 클래스는 intra/non-intra 블록 데이터를 보관한다.

### 4. 네트워크와 프로세서의 정보 교류

자바는 클래스 파일에 객체 정보와 프레임 정보를 가지고 있다.[9] 객체는 메소드와 변수들로 구성되며, 객체 데이터는 포인터로 처리되며, 실제 영상 데이터는 서버에 저장되어 있다가 바이트 코드가 해석되면서 필요에 따라 서버에서 클라이언트로 전달된다. 불론 지역(local) 데이터는 터미널 내부 메모리에 저장되어 있다. 객체 정보는 데이터, 데이터 크기 및 종류, 클래스 파일 포인터 등으로 구성된다. 수행 환경을 제공하는 프레임 정보는 메소드 어레이에 대한 인덱스, 속성 어레이에 대한 인덱스, 레퍼런스 지역 변수, 오퍼랜드 스택의 최상위 위치, 클래스파일의 포인터 등을 가지고 있다.[9] 그림 5는 객체, 프레임 정보, 프로그램 스트림, 영상 데이터 스트림이 디코더 프로세서에 어떻게 입력되고 있는 지를 보여주고 있다.

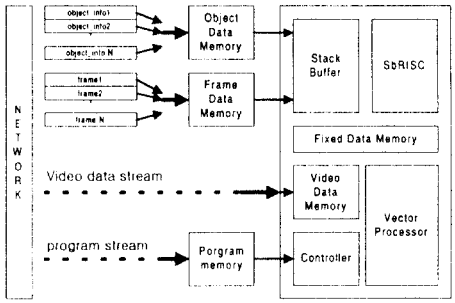



그림 5. 프로세서 내부로 입력되는 각 정보의 흐름
Fig. 5 Information flow fed into a processor

스택 버퍼는 객체, 프레임 정보를 저장하는 프로세서 외부 데이터 메모리로부터 데이터를 입력 받게 된다. 총 64개[10] 요소(elements)를 가지고 있으며, DS (Data Stack), RS(Retrun Stack) 등으로 구성된다. 본 프로세서에서는 명령어 세트에서 두 개의 스택 버퍼를 지원한다. 이는 단일 스택이 하드웨어는 단순하지만 리턴 어드레스 정보와 데이터 파라미터를 혼합하는데 많은 비용이 들고, 또한 파라미터와 리턴 어드레스가 서로 잘 놓이도록 강제되어야 하는 단점을 보완하기 위한 것이다. 단일 스택 버퍼는 모듈화된 객체 정보를 처리할 때 오버헤드가 발생하기 때문에 리턴 어드레스를 저장하는데 RS를 사용하고, 프로그램에서 사용하는 지역 변수 및 파라미터 값은 DS에 전달한다. 이와 같이 데이터 오퍼랜드와 제어 플로우 정보를 구분하게 되면 프로그램 작성시 여러 계층의

서브루틴를 통해 파라미터를 전달할 수 있기 때문에 데이터를 새로운 파라미터 목록에 올리기 위한 복사 등에 따르는 오버헤드가 줄어든다. 프레임 정보에 있던 코드 속성의 PC값에는 현재 클래스 파일 내에 있는 수행 명령어 시퀀스의 포인터 값이 저장되어 있다. 따라서 이 PC값을 참조해서 I/O 버스를 통해 프로그램 메모리에 프로그램을 로드한다. 영상 데이터 스트림을 저장하는 내부 메모리는 3개의 독립적인 메모리로 구성되며 64비트의 입출력을 갖는다. SbRISC (Stack-based RISC)는 프로세서를 제어하는 역할을 하며, VP(Vector Processor)는 IDCT나 MC 등과 같은 MPEG 디코더 동작을 처리한다.

## Ⅲ. 네트워크 기반 MPEG-4 영상 디코더 구조

### 1. MPEG-4 디코더 모델[11][12]

그림 6은 MPEG-4 수신 단말기 모델을 보여 준다.

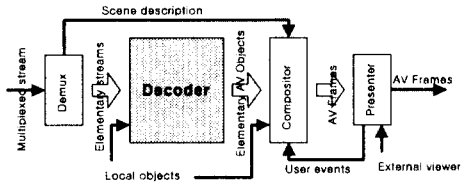



그림 6. MPEG-4 수신 단말기 모델
Fig. 6 MPEG-4 receiver terminal model

역다중화기(demultiplexer)는 망계층으로부터 기초 스트림(Elementary streams)을 분리하는 기능을 한다. 기초스트림과 함께 장면 정보(Scene description)도 역다중화기로부터 출력된다. 압축의 역과정인 복원과정(Decompression)을 거치면, 복원된 각 스트림 객체들은 재구성(Composition)을 위하여 버퍼에 임시 저장한다. 합성기에서는 타임 스탬프 등 장면 구성 정보를 참조하여 각 기초 스트림에 들어 있는 원시 AV 객체들을 합성하여 원래의 장면으로 재구성한다.[13] MPEG-4에서는 복호화 과정과 합성 과정이 분리되어 있기 때문에 사용자의 입력을 통하여 각 객체의 속성을 동적으로 변화할 수 있는 장점이 있으며, 기초 스트림을 얼마나 효율적으로 복원하느냐에 따라 합성 과정에서 객체의 합성을 용이하게 할 수 있다.

### 2. MPEG-4 디코더 구조

그림 7은 MPEG-4 디코더 프로세서 구조를 보여주고 있다.

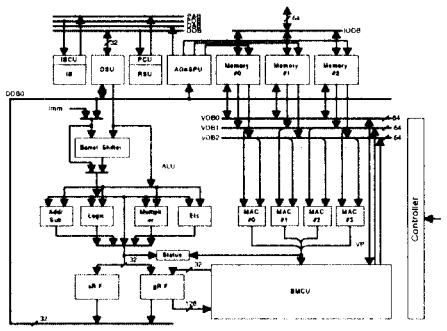



그림 7. MPEG-4 디코더 프로세서 구조
Fig. 7 The architecture of MPEG-4 decoder

그림 7의 IB(Instruction Buffer)는 명령어를 우선패취할 수 있게 하는 기능을 하며, DSU(DS Unit)는 데이터를 저장하는 스택 블록이다. RSU(RS Unit)는 리턴 어드레스를 저장하는 스택 블록이고, PCU(PC Unit)는 외부 프로그램 메모리의 어드레스를 연산한다. AGnSPU(Address Generation & Stack Pointer Unit)는 외부 스택 메모리와 DSU 사이의 데이터 송수신을 제어하고, VP는 영상 데이터를 처리하며, gRF(general Register File)는 영상 데이터를 임시 저장하기 위한 레지스터 파일이다. 또한 BMCU(Bus Mask Control Unit)은 SbRISC, VP와 내부 영상 메모리 사이의 데이터 이동을 제어한다.

SbRISC는 ALU와 레지스터 파일로 이루어져 있으며, F(Fetch)-D(Decode)-A(Address Generate)-E(Execute) 등 4단의 파이프라인을 가지고 있다. VP는 F-D-R (Read)-E-W(Write)의 5단 파이프라인을 사용하며, 읽기 사이클인 R과 쓰기 사이클인 W와는 2사이클 차이가 있으며, 이 2사이클 안에 하나의 메모리는 읽기나 쓰기 중 하나의 용도로만 사용되어야 한다.

명령어는 크게 5개 그룹으로 나눈다. 덧셈, 뺄셈, 곱셈 등 일반 연산을 수행하는 연산 그룹, 분기나 리턴 명령어를 수행하는 흐름 제어 그룹, 데이터 메모리와 스택간, 데이터 메모리와 레지스터간의 데이터 전송을 수행하는 메모리 참조 그룹, 다양한 스택 동작을

수행하는 스택 조정 그룹, 벡터 연산을 수행하는 벡터 연산 그룹 등이다. 벡터 연산 그룹을 제외한 모든 명령어는 DTOS(Top Of Data Stack) 레지스터를 기본 출발지 오퍼랜드로 사용하기 때문에 또다른 출발지 오퍼랜드 1개와 목적지 오퍼랜드 1개 등 2개 오퍼랜드로 모든 연산을 수행한다. 벡터 연산 명령어는 출발지 오퍼랜드 2개와 목적지 오퍼랜드 1개를 사용하여 총 3개의 오퍼랜드를 갖는다. 전체 명령어 개수는 62개이며, 연산 명령어와 메모리 명령어는 명령어 특성에 정수형, 문자형, 바이트형 데이터 처리를 지원하는 110개의 확장 명령어를 갖는다. 연산 명령어중 덧셈, 뺄셈, 곱셈, 비교 등은 문자형 데이터와 같은 부호없는 데이터에 대해서도 처리 가능하도록 설계하였으며, 객체형 데이터는 객체 어드레스로서 정수형 데이터와 같은 형태로 취급한다. 메모리 명령어 중에는 벡터 단위 영상 데이터 뿐만 아니라 픽셀 단위 데이터 처리가 가능하도록 내부 메모리로부터 8비트, 16비트 데이터를 레지스터에 가져올 수 있는 픽셀 명령어가 존재한다. 또한 스택 명령어는 DS내 데이터의 상호 이동, 복사를 하는 명령어가 존재한다.

SbRISC는 크게 ALU와 곱셈기, 배럴 쉬프터로 구성된다. ALU는 32비트 덧셈기,뺄셈기, 32비트 배럴 쉬프터, AND, OR, XOR, NAND 기능이 가능한 32비트 논리연산기로 구성된다. 덧셈기, 뺄셈기, 곱셈기는 부호 있는 정수 연산과 부호 없는 연산을 할 수 있도록 설계되었다. 그리고 연산 결과에 따라 N(Negative), Z(Zero), V(oVerflow), C(Carry) 등 4개의 상태값을 출력한다.

배럴 쉬프터는 쉬프트할 데이터와 몇 비트를 이동시킬 것인가를 입력한다. 각각 왼쪽, 오른쪽, 산술, 논리 이동이 가능하도록 설계되어 있으며, 왼쪽 쉬프트인 경우 오른쪽 비트들은 0으로 채워지므로 산술적, 논리적 쉬프트의 결과는 같으나 오른쪽 쉬프트인 경우 왼쪽 비트들은 산술적 쉬프트에서는 부호 비트로 채워지고 논리적 쉬프트에서는 0으로 채워지므로 결과값이 다르다. 산술적인 왼쪽 쉬프트인 경우 부호있는 정수형 32비트로 표현할 수 있는 값을 넘어서면 오버플로우가 일어난다. 배럴 쉬프터의 출력은 ALU에 입력되어 또 다른 연산을 만들며, 영상 처리와 같은 비트 조작이 많은 알고리즘에 유용하다.

덧셈, 뺄셈기는 이진 보수화 체계를 가지므로 부호없는 연산을 하는 경우 상태값만 바뀌고 출력값은 같기 때문에 한 블록으로 구현이 가능하지만 곱셈의 경우 부호있는 곱셈과 부호없는 곱셈 방법은 구조가 서로 다르고, 계산 결과 값 또한 다르기 때문에 각각 16비트 부호없는 곱셈기와 부호 있는 곱셈기를 사용한다. 또한 필요에 따라 부호없는 곱셈과 부호 있는 곱셈을 따로 계산할 수 있기 때문에 연산 속도를 높일 수 있는 장점이 있다.

오퍼랜드는 모두 32비트 부호있는 정수이며, 소스 오퍼랜드에서 32비트 정수의 부호 비트(MSB, 즉 비트 31)와 하위 15비트로 16비트 부호있는 정수를 생성한 후 연산을 하며 결과는 32비트 부호있는 정수가 된다. 단 소스 오퍼랜드가 16비트 부호있는 정수의 범위를 벗어나도 오버플로우를 발생하지 않는다. 즉 소스 오퍼랜드의 비트 30-비트16은 무시한다.

레지스터는 범용 레지스터 16개, PC, FLR(Flag Register), BsR(Base Register), IxR(Index Register), SP 등이 있다. 레지스터는 벡터 연산을 할 때 데이터를 내부 메모리에 직접 쓰거나 읽는 경우 이외에 레지스터를 이용한 데이터 처리가 가능하도록 하기 위해 4개씩 묶어 벡터 레지스터로도 사용이 가능하도록 설계하였다. 이는 VP에서 발생한 벡터 데이터를 단일 사이클에 레지스터에 저장하기 위해서이다. SP는 외부 데이터 메모리의 스택 영역과 레지스터 사이의 데이터 교환을 위해 스택의 어드레스를 저장하고 있으며, 현재 객체의 지역 변수를 저장하고 있는 곳의 메모리 어드레스, 현재 수행되고 있는 프레임 포인터, 오퍼랜드 스택의 최상위 위치 등을 저장한다.[14]

DDB0(Data Data Bus 0)는 데이타 버스로, 내부의 모든 레지스터와 연결되어 있어서 읽기, 쓰기가 가능하다. PC는 이 데이터 버스에 접속할 수 없다. 오퍼랜드를 코어 연산부로 가져오거나 레지스터 들간의 데이터 이동 및 스택과 레지스터간, 스택과 스택간의 데이터 이동을 하는데 이용한다. PAB(Program Address Bus), PDB(Program Data Bus)는 제어기에서 수행되는 외부 프로그램 메모리의 어드레스와 데이터를 입, 출력하기 위한 버스이며, DAB(Data Address Bus), DDB(Data Data Bus)는 외부 데이터 메모리의 어드레스와 데이터를 제어기에 입, 출력시키기 위한 버스이다. 또한 IODB(Input/Output Data Bus)는 VPU 내부에 있는 영상 데이터를 DRAM과 같은 외부 프레임 메모리로부터 프로세서 내부로 입, 출력하기 위한 버스이다. 또한 VDB0(Vector Data Bus 0), VDB1(Vector

Data Bus 1), VDB2(Vector Data Bus 2)는 64비트 데이터 버스로 VP와 내부 메모리 사이의 데이터가 오가는 버스이다.

VP 블록은 4개의 데이터를 동시에 처리하는 벡터 연산 구조를 가지고 있으며, MAC 연산기로 구성된다. MAC 블록은 16비트 덧셈기/뺄셈기, 16비트 절대값기/반감기, 16비트 곱셈기와 32비트 어큐뮬레이터로 구성된다. 16비트 곱셈기는 부호있는 연산을 하며 MAC연산인 경우 32비트 출력을 32비트 어큐뮬레이션 한다. VP는 두 개의 64비트 입력을 받는다. 16비트 4개의 64비트 데이터가 MAC 블록에서 연산이 된 후 명령어에 따라 버스로 출력되거나 어큐뮬레이터 레지스터에 저장된다. MAC 출력 128비트는 MAC에 있는 128비트 어큐뮬레이터에 저장된다. 덧셈기/뺄셈기의 출력은 절대값기/반감기의 입력으로도 사용될 수 있어 또 다른 연산을 만들어 낸다. 덧셈기와 반감기가 같이 사용되면 두 입력의 평균이 되며, 뺄셈기와 절대값기가 같이 사용되면 두 입력의 절대 차이가 된다. 이는 본 VP를 DCT 연산과 앞으로 ME 및 MC 등에 응용하기 위함이다.

## Ⅳ. 성능 평가 및 결과

설계된 네트워크 기반 MPEG-4 디코더는 코어에 대해 0.6㎛ 5-Volt CMOS COMPASS 라이브러리를 사용하여 검증하였다. 표 1은 여러 프로세서 내의 제어기 동작을 비교한 것이다.

표 1. 여러 프로세서 내의 제어기 동작 비교
Table 1. Comparison of controller operation in several processors

| | Technology | 동작주파수 | 성능 | 데이터버스 |
|---|---|---|---|---|
| A[19] | 0.3㎛CMOS 3LM | 70MHz | . | 16비트 |
| B[20] | 0.25㎛CMOS 4LM | 81MHz | 81MIPS | 16비트 |
| C[21] | 0.45㎛CMOS 2LM | 66.6MHz | 52.4MIPS | 32비트 |
| D[22] | 0.75㎛CMOS 2LM | 45MHz | 450MIPS | 16비트 |
| Ours | 0.6㎛CMOS 2LM | 50MHz | 200MIPS | 32비트 |

본 논문의 제어기는 연산부의 곱셈기에서 최장 경로를 형성한다. 데이터 처리는 32비트 단위이지만 32비트 곱셈기를 사용할 경우 단위 사이클당 40ns 이상의 시간이 지연되며, 이는 25MHz 동작 속도를 갖게 되므로써 현저한 성능 저하를 초래한다. 따라서 본 논문에서는 부호있는 곱셈 연산과 부호 없는 연산이 서로 다르기 때문에 16비트 부호 있는 곱셈기와 16비트 부호 없는 곱셈기를 각각 설계함으로써 최소 44MHz 동작 속도를 얻는다. 시뮬레이션 결과는 44~77MHz의 동작 속도를 갖는다. 정수형 데이터는 44MHz로 동작하며, 바이트 데이터는 77MHz의 동작 속도를 갖는다. 정수형 데이터는 부호 있는 곱셈의 경우 16비트 곱셈을 4 번 실행하면 그 결과를 얻는다. 문자형과 객체형은 부호없는 연산만을 하기 때문에 50MHz로 정수형에 비해 연산 속도가 약 12% 빠르다. 표 1의 A, B, D 예는 영상 프로세서의 제어기로서 데이터 버스는 모두 16비트이다. 이는 모든 영상 정보를 16비트 데이터로 표현할 수 있기 때문이다. C는 RISC 전용 프로세서로 32비트 데이터 버스를 갖는다. 본 논문의 제어기는 영상 데이터 뿐만 아니라 객체 지향 언어적 특성을 갖는 프로그램을 제어하기 위해 외부 데이터 메모리 접속 회수가 많고, 프레임 등 블록 단위 데이터의 보존과 이동이 많기 때문에 32비트 버스가 효율적이라고 판단된다. 또한 MPEG-4의 CAE(Contextbased Arithmetic Coding), 스프라이트(Sprite) 코딩[11] 등은 32비트 데이터를 필요로 하기 때문에 본 제어기의 확장성 면에서도 32비트 구조가 더 효율적이다. 표 2는 대표적인 스택 기반 내장형 제어기인 피코자바와의 특성 비교를 보여주고 있다.[2]

표 2. 피코자바와의 특성 비교
Table 2. Feature comparison with picoJava[15]

| | PicoJava | Ours |
|---|---|---|
| 명령어 수 | 224개 | 62개+110개(확장형) |
| 한 명령어당 사이클 수 | 3 번 | 1 번(2번) |
| 파이프라인 | 4단 | 4단 |
| 스택 버퍼 수 | 1 | 2(64 SEs) |
| 레지스터 수 | 4개 | 26개 + reserved |

본 논문의 제어기는 피코자바와 비교하여 더 작은 명령어 수를 가지고 있지만 피코자바 대부분의 명령어를 구현한다. 또한 32비트 명령어 포맷을 가지고 있기 때문에 한 명령어당 한 사이클 수행이 가능하다. 또한 스택을 기반으로 동작하기 때문에 4단의 파이프라인으로 충분하며, 스택 명령어를 제외한 대부분의 명령어가 단일 사이클에 수행이 가능하다. 외부

메모리를 읽고, 쓰는 메모리 명령어는 동시에 같은 어드레스를 접속하지 않으면 충돌이 일어나지 않는다. 스택 명령어중에서 스택 간의 데이터 상호 교환 등을 하는 명령어는 2 사이클이 필요하지만 이는 피코자바의 3사이클보다 한 사이클이 작다. 또한 범용으로 사용할 수 있는 레지스터 파일이 있으므로 범용 RISC와 같은 동작이 가능하며, 따라서 벡터 연산이 이루어질 경우 피코자바는 레지스터에 저장해야 할 값을 스택에서 가져와야 하기 때문에 패취에 따른 동작 사이클 수가 늘어나야 하지만 본 제어기는 단일 사이클 처리가 여전히 유효하다. 따라서 응용 범위가 피코자바에 비해 훨씬 넓다고 할 수 있다. 또한 스택 버퍼 수가 2개이므로 파라미터 값과 어드레스를 따로 저장할 수 있어 스캐쥴링에 따른 제어가 따로 필요없다.

표 3은 본 프로세서의 MPEG-4 코어 알고리즘에 따른 처리 성능을 보여 주고 있다.

표 3. MPEG-4 코어 알고리즘에 대한 프로세서 성능(15fps, QCIF)

Table 3. Processor performance for core MPEG-4 algorithms(15fps, QCIF)

| 알고리즘 | | | 처리 성능 |
|---|---|---|---|
| IDCT(without unpadding) | | | 95ms |
| SA-IDCT | | | 95ms |
| VOP 재생 | | | 0.002ms |
| 움직임 예측 | P-VOP | 일반 모드 | 12ms |
| | | 고급 모드 | 67ms |
| | B-VOP | 일반 모드 | 26ms |
| | | 고급 모드 | 29ms |

MPEG-4의 최대 해상도는 통상 VLBV(Very Low Bitrate Video)에서 화면율이 15Hz이하이며,[16] 화면의 크기가 176×144이다. 따라서 15fps(frame per second)으로 176×144 픽셀 수를 갖는 영상을 본 프로세서로 처리한다고 할 때 15fps×1.5×99MB(176×144pels)×4block/MB×(128회 MAC×2단+12회)×2-D IDCT = 4,775,760 사이클이 필요하다. 4개의 MAC을 가지고 있으므로 8개의 가로, 세로 픽셀은 4개의 픽셀씩 2-패스로 VP로 전달되기 때문에 64회×2회+12회(오버헤드) 동작을 한다. 사이클당 20ns 이하의 동작 속도를 가지고 있으므로 95,515,200ns(≒95ms) 지연 시간을 갖는다. 움직임 예측이 IDCT에 비해 다소 많은 시간을 차지하는 것으로 나타나고 있는데 이는 고급 모드에서 반화소 단위(half-pel) 예측 이외에 OBMC(Overlapped Block Motion Compensation)[17]를 하기 때문이다. B-VOP에서는 고급 모드에서 8×8 예측만 하고, OBMC를 하지 않도록 되어 있기 때문에 일반 모드와 별 차이가 나지 않는다. 또한 VP 블록을 이용한 IDCT와 SA-IDCT 처리 속도는 같다.[18] 그러나 IDCT 연산은 모양 정보를 이용한 패딩(padding)과정이 필요하기 때문에 처리 속도가 다소 많이 걸린다.

본 논문의 프로세서는 벡터를 전용으로 처리할 수 있는 VP블록을 가지고 있기 때문에 픽셀 단위 처리에는 다소 적응하기 힘든 면이 있다. 그러나 ME, MC 등의 동작이 가능하도록 설계되었기 때문에 이후 영상 코덱 칩의 모듈 코어로 확장하여 사용할 수 있는 장점을 가지고 있다.

## V. 결 론

본 논문에서는 네트워크 환경에 적응이 용이한 객체 지향형 영상 프로그램을 효율적으로 처리하기 위한 32비트 SbRISC와 4개의 고속 MAC을 내장한 VP를 설계하였다. 설계한 코어는 0.6㎛ 5-Volt TLM COMPASS 라이브러리를 사용하였으며, SA-IDCT, MC 등 MPEG-4 코어 알고리즘을 구현하고 있다. 또한 SbRISC는 4단의 파이프라인이 되도록 설계되어 최고 44~77MHz까지의 동작 속도를 가지고 있으며, 다양한 데이터 형태를 처리할 수 있도록 32비트 정수형을 기본으로 바이트형, 문자형 등 확장 명령어를 지원한다. 스택 명령어를 제외한 모든 명령어는 한 사이클에 한 동작이 가능하다. 레지스터는 스택 버퍼 이외에 영상 데이터의 벡터 연산 결과를 저장할 수 있도록 32비트 뿐만 아니라 4개의 레지스터를 단일 레지스터로 인식하도록 설계하였다. 전송율은 MPEG-4의 VLBV 프레임율을 만족하며, MAC 블록 내에 절대값기와 반감기가 있기 때문에 ME, MC 등의 동작이 가능해 향후 영상 프로세서의 내부 모듈로 확장할 수 있도록 하였다. 또한 고품질의 영상 데이터를 객체화 하여 처리할 수 있도록 일반 변수 데이터와 리턴 어드레스를 저장하는 스택 버퍼를 각각 둠으로써 스케줄링에 따른 오버헤드를 줄였다. 앞으로 픽셀 단위 영상 데이터 처리를 위한 VP 블록의 보완과 명령어를 추가하고, 모양 정보 디코딩 블록을 추가할 것이다. 또한 I/O에 대한

보다 구체적인 정의를 통하여 영상 데이터 메모리가 추가된 영상 처리 프로세서로 확장할 것이다.

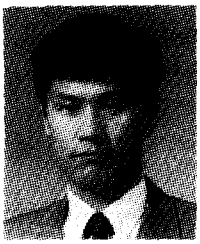
## 참 고 문 헌


1. 전병우, 이광기, "MPEG-4 응용", 한국통신학회지 제 14권 제 9호 ,pp.120-127, 1996.9.
2. Tatsuro Hokugo, "Technology trend : Java Chips Aim at Embedded Control of Java chip", http://www.nikkeibp.com/nea/september/septt.html, Sept. 1996.
3. Fred Halsall, *Data Communications, Computer Networks and Open Systems*. AW, third ed., 1994.
4. W.Richard Stevens, *Unix Network Programming*, PH Software Series, 1990.
5. Jon Postel. "Transmission control protocol", http://ds.internic.net/rfc/rfc793.txt, 1981.
6. T. Noble, T. Selling, S. Emery, "Java Virtual Machine and Library Implementation", http://chico.uccs.edu/~smemery/cs525pa2.htm#_Toc355347682, 1996.
7. "Java MPEG Source Files", http://yertle.csl.uiuc.edu/multim...et/applet-src/
8. J.B.Lee, et.al., "Java Downloadable MPEG-4 Decoder", http://www.ctr.columbia.edu/~jbl/FLEXVIEW.htm, 1997.
9. M.Hjersing and A.Ive, "JAVAX : An implementation of the Java Virtual Machine", Master Thesis, Dept. of Computer Science Lund Institute of Tech. Sweden, pp.10-19, 1996.
10. Philip J. Koopman, Jr., "Stack Computers : the new wave", http://www.cs.cmu.edu/~koopman/stack_computers/, 1989.
11. "MPEG-4 Video Verification Model Version 8.0", ISO/IECJTC1/SC29/WG11 MPEG97/N1796, July 1997.
12. "MPEG-4 Project Description", ISO/IEC/JTC1/SC29/WG11 DOC.NO.96/N1177, Jan.1996
13. "MPEG-4 Overview" http://drogo.cselt.stet.it/mpeg/standards/mpeg-4.htm, Feb. 1998.
14. Tim Lindholm and Frank Yellin, *The JavaTM Virtual Machine Specification*, Addison Wesley, 1996.
15. Marc Tremblay and Michael O'Connor, "picoJava : A Hardware Implementation of the Java Virtual Machine", http://infopad.eecs.berkeley.edu/HotChips8/4.3/, 1996.
16. Thomas Sikora, "MPEG-4 Very Low Bitrate Video", http : //wwwam.HHI.DE/mpeg-video/papers/sikora/vlbv.htm, 1997.
17. "Video Coding for Low Bitrate Communication", Draft ITU-T Recommendation H.263, Dec. 1995.
18. 박주현, 김영민, "데이터패스를 이용한 SA-DCT 구현", 대한 전자공학회 논문지, 35권 제 5호, 1998년 5월.
19. Masafumi Takahashi, et. al., "A 60mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme", ISSCC'98 Digest, pp.36-37, Feb. 1998.
20. E.Miyagoshi, et. al., "A 100㎟ 0.95W Single-chip MPEG2 MP@ML Video Encoder with a 128GOPS Motion Estimator and a Multi-Tasking RISC-Type Controller", ISSCC'98 Digest, pp.30-31, Feb. 1998.
21. T.Shimizu, et.al., "A Multimedia 32b RISC Microprocessor with 16Mb DRAM", ISSCC'96 Digest, pp.216-217, Feb. 1996.
22. D.Brinthaupt, et.al., "A Video Decoder for H.261 Video Teleconferencing and MPEG Stored Interface Video Applications", ISSCC'93 Digest, pp. 34-35, Feb. 1993.

박 주 현(Ju-Hyun Park) 정회원

1969년 7월 13일생
1993년 2월 : 전남대학교 전자공학과 졸업
1995년 2월 : 전남대학교 대학원 전자공학과 졸업(공학석사)
1997년 2월 : 전남대학교 대학원 전자공학과 수료(공학박사)
1998년 2월 ~ 현재 : 전남대학교 공과대학 전자통신기술연구소(ETTRC) 전임연구원
※주관심분야 : MPEG4 코덱 설계, 객체 지향 프로세서, DSP, RISC 프로세서 설계 등임.
e-mail : tjh@chonnam.chonnam.ac.kr

김 영 민(Young-Min Kim) 정회원



1954년 4월 18일생
1976년 2월 : 서울대학교 전자공학과 졸업(공학사)
1978년 2월 : 한국과학원 전기 및 전자공학과 졸업(공학석사)
1978년 3월 ~ 1979년 7월 : 한국선박해양 연구소(주임연구원)
1986년 : 오하이오 주립대학교 전기공학과(공학박사)
1988년 6월 ~ 1991년 8월 : 한국전자통신연구소(실장)
1991년 9월 ~ 현재 : 전남대학교 전자공학과 교수
1998년 3월 ~ 현재 : 전남대학교 공과대학 전자통신기술연구소(ETTRC) 소장
1997년 12월 ~ 현재 : 반도체설계교육센터(IDEC) 전남대학교 지역센터장
※주관심분야 : ADSL 모뎀 설계, MPEG2, MPEG4 시스템 설계 등
e-mail : kym@orion.chonnam.ac.kr