論文2003-40CI-4-4

임베디드 시스템에서의 다중 표준 영상 코덱

(Multi-standard Video Codec on Embedded System)

金 起 徹 *, 金 民 **

(Kichul Kim and Min Kim)

요 약

본 논문에서는 H.261과 H.263 표준을 모두 만족하는 영상 코덱을 임베디드 시스템에서 구현한다. 효율적인 실시간 처리를 위하여, 영상 코덱은 하드웨어 모듈과 소프트웨어 모듈로 구분되어 임베디드 시스템에서 통합 설계된다. 소프트웨어 모듈은 실시간 운영체제와 RISC 프로세서를 이용하여 수행되며, 하드웨어 모듈과 연동하여 실시간으로 영상을 압축하고 복원한다. 시스템 버스로는 AMBA AHB가 사용되며 하드웨어 모듈은 AMBA AHB의 마스터(master)와 슬레이브(slave)의 역할을 모두 수행한다. 영상 압축과정을 실시간으로 처리하기 위하여 인코더의 하드웨어 모듈은 파이프라인으로 설계된다. 구현된 영상 코덱은 H.261과 H.263 표준에 준하여 33Mhz의 동작 주파수에서 1초 동안에 CIF 화면 15장을 동시에 압축하고 복원한다.

Abstract

This paper shows an implementation of video codec (coder/decoder) on an embedded system. The video codec supports both H.261 and H.263 standards. For efficient real-time processing, the video codec is partitioned into a software module and a hardware module. Both modules are codesigned on an embedded system. The software module is processed on a real-time operating system and a RISC processor. It cooperates with the hardware module to compress and decompress images in real time. AMBA (Advanced Microcontroller Bus Architecture) AHB (Advanced High-performance Bus) is used as the system bus. The hardware module works both as AHB masters and as AHB slaves. The encoder part of the hardware module operates in a pipelined mode to compress images in real time. The video codec compresses 15 CIF frames and simultaneously decompresses 15 CIF frames in a second according to H.261 or H.263 standard at 33 Mhz frequency.

Keywords : 영상 코덱, H.261/H.263, 임베디드 시스템, ARM, AMBA AHB

1. 서 론

오늘날 프로세서, 집적 회로, 통신 및 신호처리 기술의 발전에 힘입어서, 본격적인 멀티미디어 시대가 열리게 되었다. 많은 사람들은 개인용 휴대장비를 사용하여 멀티미디어 및 인터넷 서비스를 받고 싶어하고, 사용이 편리한 고성능의 소형 장비를 원하고 있다. 이와 같은 요구를 만족하기 위하여, 하드웨어와 소프트웨어가 조합된 시스템 혹은 전용하드웨어와 프로세서를 구동하여 특정한 기능을 수행하도록 프로그램이 내장되어 있는 시스템, 즉 임베디드 시스템(Embedded System)이

* 正會員, ** 學生會員, 서울市立大學敎 電子電氣컴퓨터工學部

(Department of Electrical & Computer Engineering, University of Seoul)

接受日字:2002年7月2日, 수정완료일:2003年6月16日

부각되고 있다.

H.324/M은 일반 전화망에서 음성, 동영상 및 데이터 통신 단말에 관한 ITU-T의 표준이며[1], 3G-324M은 IMT-2000 무선 회선 교환망에서 단말기를 사용하여 멀티미디어 서비스를 제공하기 위한 3GPP(3rd Generation Partnership Project) 규격이다[2]. 3G-324M 규격은 3GPP에서 요구하는 성능을 구현하기 위해 기존의 H.324/M 규격의 구성 요소를 새롭게 정의한 것이다. H.324/M에는 저전송률 영상 압축 및 복원 규약으로 H.263 뿐만 아니라 H.261도 만족하기를 권고하고 있으며, 3G-324M 단말은 H.263 영상 코덱을 반드시 지원해야하고 H.261에 대한 지원은 선택 사항이다. 본 논문에서는 H.261과 H.263 표준을 모두 만족하는 영상 코덱을 구현한다.

H.261과 H.263 표준은 영상을 압축하기 위하여, 한 화면 내에서 공간상으로 이웃하는 화소간의 상관성을 바탕으로 불필요한 데이터를 제거하는 인트라(intra)코딩 방식과, 시간 축에서 이웃하는 프레임간의 상관성을 바탕으로 불필요한 데이터를 제거하는 인터(inter)코딩 방식을 사용한다. 인트라 코딩 방식으로는 직교 변환의 일종인 DCT(Discrete Cosine Transform), 양자화(quantization), 엔트로피 코딩(entropy coding)이 이용된다. 인터 코딩 방식으로는 현재 영상과 이전 영상을 비교하여 유사한 부분이 발견되면 영상간의 차이 값과 움직임 벡터를 부호화하는 움직임 예측(motion estimation) 및 움직임 보상(motion compensation) 기법이 이용된다[3, 4].

영상을 압축하고 복원하는 과정은 많은 연산량과 높은 데이터의 입출력을 요구하여 실시간 처리가 쉽지 않다. 따라서 실시간으로 처리되는 영상 코덱을 구현하기 위하여, 다양한 방법이 제안되고 연구되어 왔다. H. Fujiwara[5]는 전용 하드웨어 ASIC(Application Specific Integrated Circuit)구현을 제안하였으며, M. Harrand[6]나 M. H. Miki[7]도 하드웨어만으로 영상 코덱을 단일 칩으로 구현하였다. 그러나 하드웨어에만 의존하는 구현 방법은 영상 처리 알고리듬의 복잡성 때문에 비용이 증가하고 효율성이 떨어지며, 다양한 응용을 위한 적응성을 전혀 갖지 못하기 때문에 한계성을 드러낸다.

반도체 공정 기술과 집적회로 기술의 발전으로 인하여 높은 성능의 범용 프로세서들이 개발되었고, 이러한 프로세서를 바탕으로 소프트웨어에 의존하여 영상 코덱을 구현하려는 노력이 이루어진 바 있다. 그러나 소프트웨어만으로 영상 코덱을 실시간으로 처리하기 위해서는 W. S. Chen[8]이나 H. Igura[9]가 제안한 경우와 같이 고성능의 프로세서가 탑재된 플랫폼이나 프로세서의 병렬처리가 필요하다.

하드웨어 혹은 소프트웨어만으로 영상 코덱을 구현하던 이전의 방법에서, 하드웨어와 소프트웨어를 적절히 절충하여 영상 코덱을 구현하는 하드웨어-소프트웨어 통합 구현(hardware and software codesign) 방식의 연구가 활발히 진행되고 있다. M. Harrand[10]나 M. J. Kim[11]등은 많은 연산량을 요구하는 기능을 하드웨어로, 다양한 영상 표준안의 요구 사항들은 소프트웨어로 처리하는 영상 코덱을 개발하였다.

본 논문에서는 실시간으로 처리되는 영상 코덱을 구현하기 위하여, 임베디드 시스템에서 하드웨어-소프트웨어 통합 구현 방법을 보인다. 연산량, 시스템 효율성, 구현 비용과 적응성을 고려하여 하드웨어 모듈과 소프트웨어 모듈로 구분하여 효율적인 영상 코덱을 구현한다. 구현된 영상 코덱은 H.261 또는 H.263 영상 코덱에 적은 양의 하드웨어를 추가하여 H.261과 H.263 표준을 모두 만족한다.

본 논문에서 구현된 영상 코덱이 M. Harrand[10]나 M. J. Kim[11]의 영상 코덱들과 다른 점은 멀티태스킹(multitasking)이 가능한 실시간 운영체제를 탑재하여 여러 어플리케이션(application)을 동시에 수행한다는 것이다. 이러한 특징은 프로세서의 효용성을 크게 높일 수 있다. 예로서 H.324/M 멀티미디어 단말기를 구현할 시에 본 논문의 영상 코덱에 사용되는 프로세서는 영상 코덱의 소프트웨어 모듈뿐만 아니라 오디오 코덱이나 단말기의 제어 기능도 수행하는 것이 가능하다. 또한 대부분의 반도체 회사에서 IP(Intellectual Property)로 제공되는 ARM 프로세서와 AMBA(Advanced Microcontroller Bus Architecture) AHB(Advanced High-performance Bus)를 채용하여 단일 칩 구현이 용이하며, AMBA AHB를 사용하는 모든 시스템에 IP로의 사용이 가능하다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 임베디드 시스템에서의 영상 코덱의 개발 환경을 소개한다. Ⅲ장에서는 실시간으로 처리되는 영상 코덱의 하드웨어-소프트웨어 통합 구현 방법을 보인다. Ⅳ장에서는 구현된 영상 코덱의 구조와 동작을 보인다. 마지막으로 Ⅴ장에서는 결론을 맺는다.

Ⅱ. 임베디드 시스템 개발 환경

본 논문에서 구현된 영상 코덱은 시스템 버스가 구현되어 있고 RISC 프로세서와 메모리 및 FPGA가 탑재되어 있는 임베디드 시스템에서 개발되었다.

소프트웨어 모듈를 수행하기 위하여, RISC 프로세서와 실시간 운영체제를 다음과 같이 선정하였다. 프로세서는 화상전화 또는 휴대장치로의 응용을 고려하여 저전력 고성능의 32비트 RISC 프로세서인 ARM940T를 채택하였고 실시간 운영체제로는 VxWorks를 사용하였다. ARM940T는 ARM9TDMI를 프로세서 코어로 사용한다. ARM940T에는 4KB 인스트럭션 캐시(instruction cache) 및 4KB 데이터 캐시(data cache)가 포함되어있고, 실시간 운영체제를 지원하기 위한 메모리 프로텍션 유닛(MPU : Memory Protection Unit)이 존재한다. 또한 단일 칩 구현시 IP로 대체가 가능하다. VxWorks는 선점형 실시간 운영체제로서, 네트워킹과 파일 시스템을 내장하고 있으며 POSIX 1003.1b, ANSI C, TCP/IP 등과 같은 산업계 표준을 지원하고 ROM, Disk, Network 부팅 방법을 사용한다.

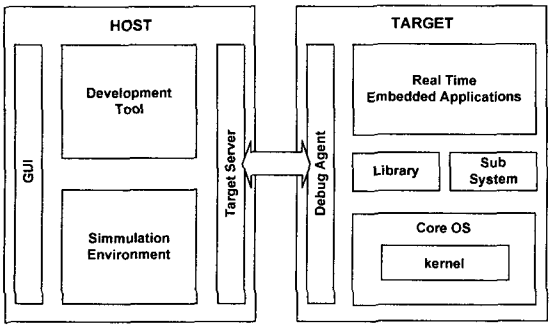


그림 1. 소프트웨어 모듈의 개발 환경
Fig. 1. Development environment of software module.

<그림 1>은 소프트웨어 모듈의 개발환경을 나타낸다. 소프트웨어 모듈의 개발은 VxWorks를 지원하는 Tornado 환경에서 이루어졌다. 소프트웨어 개발 프로그램 및 시뮬레이션 환경을 포함하는 호스트(PC)는 이더넷, 시리얼 통신 등을 통하여 구현하고자 하는 대상 시스템과 연결되어 있어, 하드웨어 설계 및 디버깅과 함께 소프트웨어 개발이 병행되었다.

프로세서와 주변 모듈간의 데이터 전송을 위해서는 시스템 버스 구조를 선정해야 한다. 하드웨어 모듈은 ARM940T 프로세서와 시스템 메모리에 연결되어 데이터를 전송하고 수신해야하므로 높은 대역폭의 고성능의 시스템 버스를 필요로 한다. 또한 IP로의 사용을 고려하여 범용성이 높은 버스 구조를 선정해야 한다. 본 논문에서는 ARM사에서 마이크로 프로세서 시스템용 버스로 제안한 AMBA AHB를 시스템의 버스 구조로 선정하였다.

하드웨어 모듈은 VHDL로 구현하고 FPGA에 프로그래밍하여 검증하였다. 하드웨어 모듈용 FPGA에는 200만 게이트 급인 Xilinx Virtex XCV2000E가 사용되었다.

Ⅲ. 영상 코덱의 구현 방법

본 장에서는 저전송률 영상 표준인 H.261과 H.263을 만족하는 영상 코덱의 구조를 제시하고 실시간으로 동영상을 처리하기 위한 영상 코덱의 하드웨어-소프트웨어 통합 구현 방법을 보인다.

H.261과 H.263의 표준안을 만족시키는 데 필요한 기능 블록의 구성은 다음과 같다. H.261과 H.263에 공통적으로 포함되는 기능 블록은 움직임 예측기 및 움직임 보상기, FDCT와 IDCT, 양자화기와 역 양자화기, 지그재그 스캔과 역 지그재그 스캔, 런 길이 부호기와 런 길이 복호기, 가변길이 부호기와 가변길이 복호기, 비트 스트림 발생기, 그리고 전체 시스템 제어기이다. H.261의 경우는 루프 필터가, H.263의 경우는 반 화소 움직임 예측기가 포함된다.

H.261과 H.263은 각각의 표준에 준하여 서로 다른 비트 스트림 구조와 가변길이 코드를 갖는다. 따라서 가변길이 부호기와 가변길이 복호기, 비트 스트림 발생기, 그리고 전체 시스템 제어기는 H.261과 H.263에 공통적으로 포함되는 기능 블록이지만 서로 다른 동작을 수행한다.

<그림 2>는 H.261과 H.263 표준을 만족하는 영상 인코더(encoder)의 구조를 나타낸다. 영상의 압축과정은 움직임 예측 및 보상 부호화, FDCT(Forward DCT), 양자화 그리고 엔트로피 코딩으로 구성된다. 움직임 예측 및 보상 부호화는 현재 영상과 이전 영상간의 가장 유사한 부분을 찾고, 현재 영상의 크기와 변화를 줄인 영상간의 차이 값을 구한다. FDCT는 영상간의 차이 값을 주파수 성분으로 변환시킨다. 대부분의 영상 데이

터는 낮은 주파수 성분에 존재하므로, 양자화기는 높은 주파수 성분의 값들을 '0'(zero)으로 만들어 데이터의 양을 줄인다. 이어서 지그재그 스캔을 한 후, 엔트로피 부호화를 이용하여 영상을 압축한다[3, 4, 12].

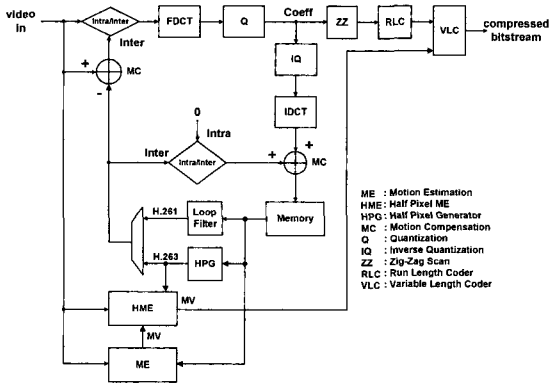


그림 2. 영상 인코더의 구조
Fig. 2. The structure of video encoder.

<그림 3>은 H.261과 H.263 표준을 만족하는 영상 디코더(decoder)의 구조를 나타낸다. 영상의 복원과정은 엔트로피 디코딩, 역 양자화, IDCT(Inverse DCT) 그리고 움직임 보상으로 구성된다. 입력된 비트 스트림(bit stream)은 엔트로피 디코딩, 역 지그재그 스캔, 역 양자화, IDCT 그리고 움직임 보상기를 순차적으로 처리하여 영상을 복원한다[3, 4, 12].

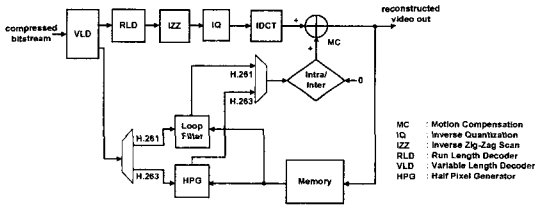


그림 3. 영상 디코더의 구조
Fig. 3. The structure of video decoder.

영상 코덱을 구성하는 기능 블록을 하드웨어 모듈과 소프트웨어 모듈로 어떻게 분리하여 구현하느냐에 따라, 영상 코덱의 시스템 성능 및 비용 등이 달라진다. 따라서 기능 블록별로 처리해야 할 연산량, 시스템의 효율성, 구현 비용과 적응성 등 다양한 요소가 고려되어야 한다. <표 1>은 영상 코덱을 기능 블록 별로 구분하여 구현 방법과 H.261과 H.263에의 포함 유무를 나타낸 것이다.

각 기능 블록에 필요한 연산량은 H.261과 H.263에서 어느 정도의 차이는 있지만 크게 다르지 않으며 H.261의 각 기능 블록에서 필요한 연산량에 대한 자세한 연구 결과를 [13]에서 찾아볼 수 있다. [13]의 연구 결과에 의하면 움직임 예측 및 움직임 보상기, 반 화소 움직임 예측기, 루프 필터, 그리고 FDCT 와 IDCT는 연산량이 많으므로 하드웨어로 구현하는 것이 바람직하다.

양자화기, 역 양자화기, 지그재그 스캔, 역 지그재그 스캔, 런 길이 부호기, 그리고 런 길이 복호기는 적은 하드웨어 비용으로 효율적인 구조를 가지므로 하드웨어로 구현 가능할 뿐만 아니라, 연산량이 적으므로 소프트웨어로도 구현 가능하다. 그러나 이와 같은 기능 블록들은 응용 프로그램을 처리해야 하는 프로세서의 부담을 줄이고 매크로 블록 단위로 처리하는 영상 코덱의 시스템 효율성을 향상시키기 위하여 하드웨어로 구현한다.

가변 길이 부호기, 가변 길이 복호기, 비트 스트림 발생기, 그리고 전체 시스템 제어기는 하드웨어 구현시 비용이 증가하고 H.261과 H.263 표준을 모두 만족시키는 데 필요한 적응성이 떨어지므로 소프트웨어로 구현하는 것이 바람직하다.

표 1. H.261과 H.263 기능블록의 구현 방법
Table. 1. The design method of H.261 and H.263 functional block.

기능 블록	H.263의 포함 유무	H.261의 포함 유무	구현방법	비고
motion estimation	○	○	하드웨어	연산량
half pixel motion estimation	○	×	하드웨어	연산량
loop filter	×	○	하드웨어	연산량
motion compensation	○	○	하드웨어	연산량
forward DCT inverse DCT	○	○	하드웨어	연산량
quantization inverse quantization	○	○	하드웨어	효율성
zig-zag scan inverse zig-zag scan	○	○	하드웨어	효율성
run length encoder run length decoder	○	○	하드웨어	효율성
variable length encoder variable length decoder	○	○	소프트웨어	적응성
bit stream generation	○	○	소프트웨어	적응성
system control	○	○	소프트웨어	적응성

Ⅳ. 영상 코덱의 구조와 동작

<그림 4>는 구현된 영상 코덱의 시스템 구조를 나타낸다. 구현된 영상 코덱은 소프트웨어 모듈, 하드웨어 모듈, 시스템 메모리 그리고 시스템 버스로 구성된다. 소프트웨어 모듈은 ARM940T와 실시간 운영체제인 VxWörks에 의해서 처리된다. 하드웨어 모듈은 인코더 하드웨어 모듈과 디코더 하드웨어 모듈로 구성되며 SSRAM을 지역 메모리(local memory)로 사용한다. H.261 또는 H.263 동작 모드에 따라서, 약간 다른 하드웨어 모듈의 기능 블록들이 동작한다. 루프 필터기는 H.261 동작 모드에서만, 반 화소 움직임 예측기는 H.263 모드에서만 동작한다. 움직임 예측기는 H.261 또는 H.263 동작모드에 따라서 서로 다른 탐색 영역을 가지면서 동작한다.

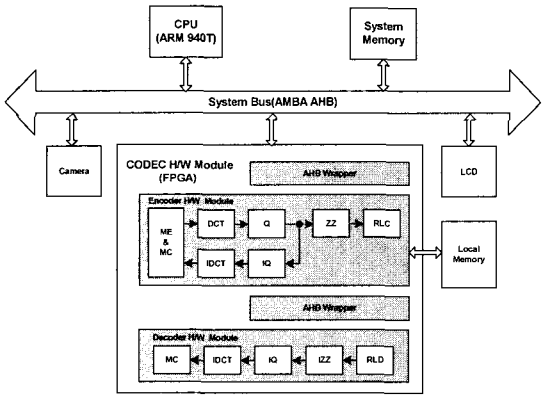


그림 4. 영상 코덱의 시스템 구조
Fig. 4. The system structure of video codec.

구현된 하드웨어 모듈과 RISC 프로세서 및 시스템 메모리간의 데이터 통신을 위하여, 시스템 버스로는 AMBA AHB를 사용하였다. 구현된 인코더 하드웨어 모듈과 디코더 하드웨어 모듈은 AHB 마스터(master)와 슬레이브(slave)로 동작하며, 이를 위해서는 AMBA AHB의 인터페이스 역할을 수행하는 AHB Wrapper가 구현되어야 한다. 구현된 AHB Wrapper가 AMBA AHB의 프로토콜에 해당하는 부분을 담당함으로써 AMBA AHB를 사용하는 하드웨어 모듈이 독립적으로 설계될 수 있었다. AHB Wrapper를 사용함으로써 본 논문에서 제안된 영상 코덱용 하드웨어는 AMBA AHB 버스를 사용하는 모든 시스템에 사용 될 수 있다.

영상 입출력 부분과 같은 데이터의 이동량이 많은 부분을 슬레이브로 구현하였을 경우, 프로세서가 마스터로서 동작하여 많은 양의 데이터를 하드웨어 모듈에게 전송해야하므로 프로세서에게 큰 부담을 주게되어 비효율적이다. 그러므로 데이터의 이동량이 많은 영상 입출력 부분은 마스터로서 동작하도록 구현하였다. 구현된 하드웨어 모듈은 프로세서로부터 동작에 필요한 기본정보를 전송 받아야 하므로 프로세서로부터 입력받는 레지스터(register) 부분은 슬레이브로서 구현하였다.

영상 코덱의 입출력 영상은 네 개의 계층 구조로 구성되며, 픽쳐(picture), 블록군(GOB: Group of Block), 매크로 블록(macro block), 블록(block)이 있다[3, 4]. 블록은 가장 작은 단위로 8×8 크기의 픽셀(pixel)로 구성된다. 매크로 블록은 네 개의 명암 블록과 두 개의 색상 블록으로 이루어져있다. 즉, 16×16 크기의 Y 영상과 8×8 크기의 U, V영상이 모여서 매크로 블록이 된다. 블록군은 매크로 블록이 모여서 이루어지며, 이 블록군들이 모여 픽쳐를 형성한다.

하드웨어 모듈로 구현된 대부분의 모듈은 8×8 크기의 블록 데이터를 처리하지만, 움직임 예측기와 반 화소 움직임 예측기는 16×16 크기의 Y 영상인 매크로 블록의 데이터를 이용하여 동작한다. 따라서 구현된 하드웨어 모듈은 매크로 블록 단위로 데이터를 처리하도록 설계하였다. 인코더 하드웨어 모듈은 매크로 블록 단위로 영상 정보를 시스템 메모리로부터 읽어와서, 가변 길이 부호화에 필요한 정보를 시스템 메모리에 저장한다. 디코더 하드웨어 모듈은 시스템 메모리로부터 매크로 블록 단위로 영상 복원에 필요한 정보를 읽어와서, 매크로 블록 단위로 영상을 복원한다.

움직임 예측기는 수평·수직 탐색범위에서 16×16 크기의 Y 영상에 대하여, 현재 영상과 이전 영상을 비교하고 움직임 벡터를 찾는 기능 블록이다. 움직임 예측기에 사용된 알고리즘과 움직임 예측기의 구조에 따라서 인코더의 하드웨어 비용, 속도, 압축 비율이 크게 달라진다[13]. 구현된 인코더 하드웨어 모듈에서, 움직임 예측에는 완전 탐색 블록 정합(full search block matching) 알고리듬이 적용되었다. 완전 탐색 블록 정합 알고리듬은 정확도면에서 움직임 예측을 위한 알고리듬 중 가장 우수하며, 데이터 공급이 용이한 장점이 있다[14]. 하지만 많은 연산량과 긴 처리 시간을 요구하기 때문에, 인코더 하드웨어 모듈의 동작에는 효율적인 스케줄링(scheduling)이 필요하다.

<그림 5>는 인코더 하드웨어 모듈의 동작 과정을 나타낸다. 구현된 인코더 하드웨어 모듈의 동작 과정은 3 단계로 나눌 수 있다.

(1) 단계 : 하드웨어 모듈이 시스템 메모리로부터 압축할 영상 데이터를 읽어 들이는 부분
(2) 단계 : 정수 화소 움직임 예측기
(3) 단계 : 반 화소 움직임 예측기(H.263), 루프 필터(H.261), 움직임 보상기, FDCT, IDCT, 양자화기, 역 양자화기, 지그재그 스캔, 런 길이 부호기, 부호화된 코드인 last, run, level값을 시스템 메모리에 저장하는 부분

순차적으로 (1), (2), (3) 단계를 수행하면, 오랜 시간이 소요되기 때문에 실시간 처리가 불가능해진다. 따라서 실시간으로 영상을 압축하기 위하여 <그림 5>와 같이 각 단계가 3 단계 파이프라인 모드로 동작하게 하였다. 즉, i 번째 매크로블록(MB: Macro Block)에 대하여 정수 화소 움직임 예측기가 움직임 벡터를 구하는 동안, 정수 화소 움직임 예측이 끝난 i-1 번째 매크로블록은 런 길이 부호화를 포함한 (3) 단계가 처리되어 시스템 메모리에 저장되고, 이어서 압축될 i+1번째 매크로블록이 시스템 메모리로부터 읽혀진다.

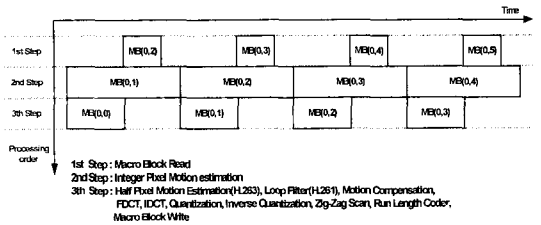


그림 5. 인코더 하드웨어 모듈의 스케줄링
Fig. 5. The scheduling of encoder hardware module.

디코더 하드웨어 모듈은 부호화된 코드워드인 라스트(last), 런(run), 레벨(level)값을 시스템 메모리로부터 읽어와서 매크로블록 단위로 영상을 복원한 후에 복원된 영상을 시스템 메모리에 저장한다. 이와 같은 과정을 수행하는 데 소요되는 시간은 인코더 하드웨어 모듈이 하나의 매크로블록을 처리하는 데 걸리는 시간에 비하여 매우 적다. 그러므로 디코더 하드웨어 모듈은 파이프라인을 사용하지 않고 매크로블록을 순차적으로 처리하도록 구현하였다.

구현된 영상 코덱에서, 하드웨어 모듈과 소프트웨어 모듈간의 연동 과정이 <그림 6>에 있다. 하드웨어 모듈과 소프트웨어 모듈간의 연동을 위하여, 레지스터 세팅(setting) 방법과 인터럽트(interrupt) 방법이 사용되었다.

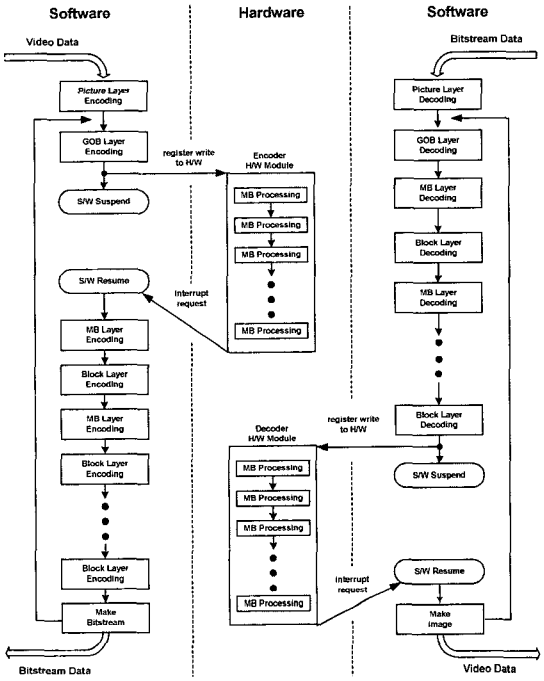


그림 6. 영상 코덱의 하드웨어 모듈과 소프트웨어 모듈의 연동 흐름도
Fig. 6. The cooperation flow of implemented video codec.

압축하고자 하는 영상 정보가 카메라를 통하여 입력되면, 인코더와 관련된 소프트웨어가 실행되어 비트 스트림 레이어(layer)의 부호화 작업이 행해진다. 복원할 비트 스트림이 통신 채널을 통하여 입력되면, 디코더와 관련된 소프트웨어는 복호화 과정을 수행한다.

소프트웨어가 실행되는 중에 영상 데이터의 하드웨어 부호화 및 복호화 과정이 요구되면, 하드웨어 부호화 또는 복호화에 필요한 파라미터와 시스템 메모리의 주소 정보 등을 하드웨어 모듈의 레지스터에 적는다. 이어서, 하드웨어 부호화 또는 복호화의 시작을 알려주는 특정 레지스터를 세팅 해주면, 구현된 하드웨어 모듈이 동작하게 된다. 하드웨어 모듈의 동작과 동시에, 하드웨어 모듈과 관련된 소프트웨어 모듈은 하드웨어 부호화 또는 복호화 과정이 끝날 때까지 일시적으로 정지(suspending)된다.

하드웨어 모듈은 레지스터에 저장된 주소 정보를 참

조하여 시스템 메모리로부터 하드웨어 부호화 또는 복호화에 필요한 영상의 데이터를 읽어들인다. 읽어온 정보는 하드웨어 모듈 내부에 마련되어 있는 버퍼에 저장이 되고, 하드웨어 모듈의 기능 블록에 의하여 처리된다. 하드웨어 모듈에 의하여 처리된 데이터는 소프트웨어 처리를 위해 버퍼에 임시로 저장되고, 매크로블록의 하드웨어 처리가 모두 끝나면, 버퍼의 데이터를 시스템 메모리에 저장한다.

시스템 메모리로의 데이터 이동이 끝나면, 하드웨어 모듈은 인터럽트 레지스터를 세팅함으로써 프로세서에게 인터럽트를 걸어, 정지되어 있던 소프트웨어를 다시 실행(resuming)시킨다. 이러한 일련의 과정이 수행되면서, 하드웨어 모듈과 소프트웨어 모듈간의 연동을 통하여 영상 데이터의 압축 및 복원이 이루어진다.

표 2. 하드웨어 모듈의 설계 요약
Table 2. The design summary of hardware module.

설계 요약	하드웨어 모듈
Number of Slices	11,629 of 19,200
	60%
Number of Slice Flip Flops	12,020
Total Number 4 input LUTs	15,027
Number of Block RAMs	43
Total gate count	940,094

(a) 원 영상 (b) H.261 (c) H.263

그림 7. 원 영상과 H.261이나 H.263으로 인코딩한 후 디코딩한 영상
Fig. 7. Original source video image and decoded image after encoding.

구현된 영상 코덱은 H.261과 H.263 표준을 만족하면서 CIF 크기의 영상화면을 1초에 15장 처리한다. 하드웨어 모듈은 VHDL로 구현되어 XCV2000E에 프로그래밍 하였다. <표 2>는 하드웨어 모듈의 설계 요약을 나타낸다. 구현된 하드웨어 모듈은 33Mhz에 동작하며 약 94만 게이트를 사용하였다.

<그림 7>은 영상 코덱의 동작을 검증하기 위하여 사용된 원 영상과 H.261이나 H.263으로 인코딩한 후 디코딩한 영상이다. 160Kbps의 비트레이트(bitrate)로 Suzie 영상을 인코딩했을 때, H.263의 경우에는 PSNR이 36.90이였고 H.261의 경우에는 33.11이였다. Trevor의 경우에는 250Kbps로 인코딩하였는데 H.263의 경우에는 PSNR이 33.93이였고 H.261의 경우에는 29.43의 결과를 얻었다.

V. 결 론

본 논문에서는 H.261과 H.263 표준을 모두 만족하는 실시간 영상 코덱의 구현을 보였다. 구현된 영상 코덱은 실시간 운영체제와 RISC 프로세서를 이용한 소프트웨어 모듈과 하드웨어 모듈로 구분되어 임베디드 시스템에서 통합 구현되었다. 움직임 예측 및 보상기, 반 화소 움직임 예측기, 루프 필터, 그리고 FDCT와 IDCT는 연산량이 많으므로 하드웨어로 구현하였다. 양자화기, 역 양자화기, 지그재그 스캔, 역 지그재그 스캔, 런 길이 부호기, 그리고 런 길이 복호기는 연산량은 적지만 영상 코덱의 시스템 효율을 높이기 위하여 하드웨어로 구현하였다. 가변 길이 부호기, 가변 길이 복호기, 비트 스트림 발생기, 그리고 전체 시스템 제어기는 하드웨어 구현시 비용이 크게 증가하고 H.261과 H.263 표준을 모두 만족시키는 데 필요한 적응성이 떨어지므로 소프트웨어로 구현하였다.

하드웨어 모듈과 RISC 프로세서 및 시스템 메모리간의 데이터 통신을 위하여, 시스템 버스로 AMBA AHB가 사용되었다. 인코더 하드웨어 모듈과 디코더 하드웨어 모듈은 AHB 마스터(master)와 AHB 슬레이브(slave)로 동작하도록 설계되었다. AHB 마스터와 AHB 슬레이브로 동작하기 위하여 AMBA AHB의 인터페이스 역할을 하는 AHB Wrapper를 설계하였다.

하드웨어로 구현된 부분은 VHDL로 설계되었고 소프트웨어로 구현된 부분은 실시간 운영체제와 저전력 RISC 범용 프로세서를 사용하여 수행되었다. 구현된 영상 코덱의 움직임 예측기에는 완전탐색 블록정합 알고리즘 방법이 사용되었고 실시간으로 영상을 압축하기 위하여 하드웨어 모듈의 구조를 파이프라인으로 구

현하였다. 구현된 영상 코덱은 H.261과 H.263 표준에 준하여 33Mhz의 동작 주파수에서 1초 동안 CIF 15장을 동시에 압축하고 복원한다.

참 고 문 헌

[1] ITU-T, "Video coding for low bit-rate communication," H.324, February 1998.

[2] 3GPP Technical Specification 3G TS 26.111 V5.0.0, "Modifications to H.324," 2002.6.

[3] ITU-T, "Video CODEC for audio-visual services at p × 64kbit/s," ITU-T Recommendation H.261, 1993.

[4] ITU-T, "Video coding for low bit-rate communication," Draft Recommendation H.263, 2 May 1996.

[5] H. Fujiwara, M. L. Liou, M.-T. Sun, K.-M. Yang, M. Maruyama, K. Shomura, and K. Ohyama, "An all-ASIC implementation of a low bit-rate video codec," IEEE Trans, Circuits Systems Video Technol., vol. 2, no, 2, pp. 123~134, June 1992.

[6] M. Harrand, M. Henry, P. Chaisemartin, P. Mougeat, Y. Durand, A. Tournier, R. Wilson, J.-C. Herluison, J.-C. Longchambon, J.-L. Bauer, M. Runtz, and J. Bulone, "A single-chip videophone video encoder/decoder," in Proc. ISSCC95, pp. 292~293.

[7] M. H. Miki, G. Fujita, T. Onoye, I. Shirakawa, "Low-Power H.263 Video Codec Dedicated to Mobile Computing," Proceedings., pp. 80~83, 1997.

[8] W. S. Chen, Y. Y. Peng, Y. T. Chang, J. T. Wang, "Design and implementation of real-time software-based H.261 video codec," Consumer Electronics, 2001. ICCE. International Conference, 2001.

[9] H. Igura, S. Narita, Y. Naito, K. Naito, K. Kazama, I. Kuroda, M. Motomura, and M. Yamashima, "An 800MOPS 110mW 1.5V parallel DSP for mobile multimedia processing," in Proc. ISSCC98, pp. 293~293.

[10] M. Harrand, et al., "A Single-Chip CIF 30Hz H.261, H.263 and H.263+ Video Encoder/Decoder with Embedded Display Controller," ISSCC Digest of Technical Papers, 1999, pp. 268~269.

[11] 김명진, 이상희, 김근배, "모듈화된 구조에 기반한 H.263 비디오 코덱 VLSI의 설계," 전자공학회 논문지, 제39권 SP편 제 5호, 2002년 9월, pp. 477~485

[12] K. R. RAO, J. J. HWANG. "Techniques & Standards For Image · Video & Audio Coding," Prentice Hall, 1996.

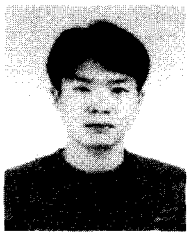
[13] M. N. Pettigrew and V. K. Madisetti, "Progresses in algorithms and VSLI architecture for motion estimation and compensation in video compression," Technical Report, Center for Signal and Image Processing, Georgia Institute of Technology, 1995.

[14] K. M. Yang, M. T. Sun, L. Wu, "A Family of VLSI Design for the Motion Compensation Block Matching Algorithm," IEEE Trans. on circuits and systems for Video Tech., Vol. 36, No. 10, pp. 1317~1325, Oct. 1989.

저 자 소 개

金 起 徹(正會員)

1982년 2월 : 서울대학교 전기공학과 졸업(공학사). 1984년 2월 : 서울대학교 대학원 전기공학과 졸업(공학석사). 1991년 8월 : University of Southern California Electrical Engineering systems(공학박사). 1984년 3월~1994년 2월 : 한국전자통신연구원 선임연구원. 1994년 3월~현재 : 서울시립대학교 전자전기컴퓨터공학부 부교수. <주관심분야 : SoC 및 임베디드 시스템 설계, 멀티미디어 시스템, 병렬처리, Reconfigurable Computing>



金 民(學生會員)

2000년 2월 : 서울시립대학교 전자전기공학부 졸업(공학사). 2002년 2월 : 서울시립대학교 전자전기컴퓨터공학부 졸업(공학석사). 2002년 3월~현재 : 서울시립대학교 전자전기컴퓨터공학부 박사과정. <주관심분야 : SoC 및 임베디드 시스템 설계, 멀티미디어 시스템, 생체인식, 병렬처리, 컴퓨터 구조>