하드웨어 소프트웨어 통합 설계에 의한 H.263 동영상 코덱 구현

정회원 장 성 규*, 김 성 득*, 이 재 헌*, 김 진 수**, 정 의 철**, 최 건 영**, 김 종 대**, 나 종 범*

An Efficient Hardware-Software Co-Implementation of an H.263 Video Codec

Sung Kyu Jang*, Sung Deuk Kim*, Jae Hun Lee*, Jin Soo Kim**, Ui chel Joung**, Gun Young Choi**, Jong Dae Kim**, Jong Beom Ra* ***Regular Members***

요 약

이 논문에서는 하드웨어와 소프트웨어의 통합 설계에 의한 H.263 동영상 코덱을 구현한다. 동영상의 부호화와 복호화를 실시간으로 수행하기 위해 동작 속도 및 응용성을 동시에 고려하여 H.263 코덱의 각 부분 중 어느 부분이 하드웨어 또는 소프트웨어로 구현되는 것이 바람직한지 결정하였다. 하드웨어로 구현하는 부분은 움직임 추정부 및 보상부와 메모리 제어부이고, 나머지 부분은 RISC (reduced instruction set computer) 프로세서를 사용하여 소프트웨어로 처리한다. 이 논문에서는 하드웨어 및 소프트웨어 모듈의 효과적인 구현 방법을 소개한다. 특히 하드웨어로 구현되는 움직임 추정부를 위해서 주변 움직임 벡터의 상관성 및 계층적 탐색을 이용한 다수의 움직임 후보를 가지는 알고리즘을 사용하였으며, 이 알고리즘에 기반한 소면적 구조를 제안한다. 소프트웨어로 처리되는 DCT (discrete cosine transform) 부분의 최적화를 위해서 움직임 추정부에서 얻어진 SAD (sum of absolute difference) 값에 근거하여 DCT 이후 양자화된 계수들의 통계적 특성을 분류하는 기법을 사용한다. 제안된 방법을 실제 RISC 프로세서와 gate array를 이용하여 구현하고, 그 성능이 우수함을 확인하였다.

ABSTRACT

In this paper, an H.263 video codec is implemented by adopting the concept of hardware and software co-design. Each module of the codec is investigated to find which approach between hardware and software is better to achieve real-time processing speed as well as flexibility. The hardware portion includes motion-related engines, such as motion estimation and compensation, and a memory control part. The remaining portion of the H.263 video codec is implemented in software using a RISC processor. This paper also introduces efficient design methods for hardware and software modules. In hardware, an area-efficient architecture for the motion estimator of a multi-resolution block matching algorithm using multiple candidates and spatial correlation in motion vector fields (MRMCS), is suggested to reduce the chip size. Software optimization techniques are also explored by using the statistics of transformed coefficients and the minimum sum of absolute difference (SAD) obtained from the motion estimator.

* 한국과학기술원 전기 및 전자공학과 영상시스템 연구실 ([skjang, sdkim, jhlee]@issserver.kaist.ac.kr, jbra@ee.kaist.ac.kr)
** (주) 삼성전자 멀티미디어 연구소 ([jinskim, ujoung, choigy]@mmrnd.sec.samsung.co.kr)
논문번호 : 99422-1019, 접수일자 : 1999년 10월 19일

Ⅰ. 서 론

멀티미디어 환경이 도래함에 따라, 최근 동영상 처리에 대한 관심이 증대되고 있다. 이에 따라 화상회의, 화상전화, 감시카메라 등의 낮은 전송률 환경에서 사용할 수 있는 H.263이 ITU에서 표준화 되었다[1]. 동영상 부호화 및 복호화는 처리해야 할 연산량이 방대하고 메모리 입출력을 빈번히 요구하기 때문에 이를 위한 전용 하드웨어의 개발이 필수적이다[2]. 그러나 이러한 전용 하드웨어는 다양한 응용을 위한 가변성을 가지지 못하기 때문에 그 이용에 한계가 있다. VLSI 기술이 점차적으로 발전함에 따라 범용 프로세서의 계산능력이 향상되어 왔다. 이러한 향상된 프로세서를 사용하여 동영상 부호화 및 복호기화의 실시간 처리를 위해서 단지 소프트웨어만으로 구현하고자 하는 경향이 있다. 이러한 소프트웨어적인 방식을 이용한 실시간 처리를 만족시키기 위해서 병렬처리나 멀티미디어 전용 명령어를 내장한 프로세서들이 개발되었다[3,4]. 그러나 이러한 구현 방식 또한 실시간 처리라는 엄격한 요구사항을 충분히 충족할 수 없기 때문에 그 한계가 있다.

소프트웨어 방식과 하드웨어에 의한 방식을 모두 고려해 볼 때 그 사이에는 절충점이 있다. 응용성, 개발 비용, 전력 소모 및 처리 속도 등의 다양한 요소들을 염두에 두어야 한다. 하드웨어에 기반한 구현 방식은 처리 속도나 전력 소비면에서 소프트웨어적인 구현 방식에 비해 이점이 있으며, 소프트웨어적 처리는 하드웨어 구현에 비해 더 가변적이어서 최소한의 수정을 거쳐 다양하게 사용될 수 있다. 소프트웨어로 처리를 하게 되면 그 연산량으로 인해 많은 시간이 필요로 하지만, 하드웨어로 구현할 경우 효율적인 소면적 구조를 취할 수 있는 부분들은 하드웨어로 구현되고, 통계적 특성 등을 이용하여 최적화가 가능한 부분들은 소프트웨어로 구현하는 것이 바람직하다.

따라서 이 논문에서는 H.263 동영상 코덱의 어떤 부분을 하드웨어 또는 소프트웨어로 처리할 것인지를 올바르게 결정하기 위해 단지 소프트웨어로만 이루어진 코덱을 구현하고 그 복잡도를 각 부분별로 조사하였다. 결국 복잡도가 높은 움직임 추정부와 움직임 보상부는 하드웨어로 구현하고, 나머지 부분은 소프트웨어로 구현 하였다. 그리고 또한 효율적인 소면적 움직임 추정기와 보상기의 구조를 제안하였으며, 소프트웨어 최적화 기법도 기존의 DCT/Q를 생략법 (skipping) 을 사용하고 계산량을 좀 더 줄이기 위해서 가지치기 (pruning) 방법을 추가하여 확장된 형태의 skipping 기법을 제안하였다.

이 통합시스템의 주목할만한 특징은 하드웨어에서 부가적으로 이용할 수 있는 SAD (sum of absolute difference) 정보를 사용하여 소프트웨어 최적화에 사용 할 수 있는 방안을 모색하였다. 즉, 움직임 추정부 하드웨어는 매크로 블럭 SAD 뿐만 아니라 블럭 SAD도 추가적인 계산량 없이 얻을 수 있는 구조로 고안되어 있으며 이 블럭 SAD를 사용하여 DCT나 양자화 (quantization) 와 같은 소프트웨어 연산을 최소화 할 수 있다.

이 논문의 구성은 다음과 같다. II 장에서는 전체 시스템을 조망하고 하드웨어 소프트웨어 분할에 관해서 다룬다. III 장에서는 효율적인 소면적 하드웨어 구조에 대해서 언급한다. IV 장에서는 H.263 각 부분에서 처리될 데이터들의 통계적 특성을 사용한 최적화 기법을 설명한다. 하드웨어와 소프트웨어의 스케줄링 문제를 V 장에서 다루고 구현된 시스템의 검증 및 그 결과를 VI 장에서 기술한다. 마지막 VII 장에서 결론을 맺는다.

Ⅱ. 시 스 템 구 조

1. 시스템 개괄

그림1은 개발된 동영상 압축기의 블럭도를 보여준다. 구현된 시스템은 범용 RISC프로세서와 시스템에서 사용하는 SDRAM (synchronous dynamic random access memory) 및 최소한의 부가적인 하드웨어로 구성되어 있다. 본 시스템에서 사용되는 RISC프로세서 (구현된 시스템에서는 ARM을 사용:

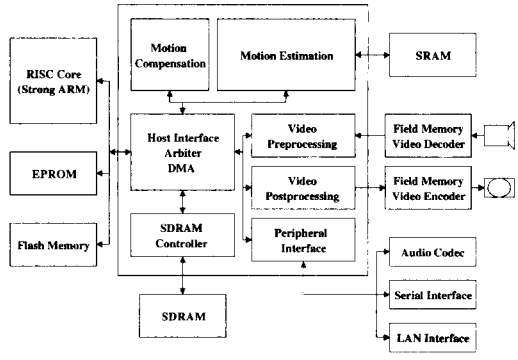


그림 1. 동영상 통신 시스템의 전체 블럭도.

Advanced RISC machine, 200 MHz) 의 사용 가능한 연산량 중 단지 30% ~ 40% 정도의 연산량만이 H.263을 위해 할당되며 나머지는 G.723, H.223 및 H.245와 같은 H.324 시스템을 위해 할당하였다. 실시간 처리라는 요구 사항과 확장성 등을 고려하여 소프트웨어-하드웨어 통합설계 방식이 사용된다.

최소화된 게이트 수를 갖는 하드웨어 설계와 소프트웨어 최적화가 본 논문의 핵심 사항이다. 움직임 추정부를 위해서 특별히 사용된 소면적 SRAM (static random access memory) 을 제외하고는 전체 시스템이 단일 메모리를 사용한다. 하드웨어 부분은 움직임 추정부 및 보상부, SDRAM 제어기, host 인터페이스부 및 arbiter 이다. 하드웨어들은 ASIC (application specific integrated circuit) 으로 설계되었다. RISC 프로세서는 인터럽트와 제어 가능한 레지스터를 통해서 하드웨어 모듈과 통신을 한다. Arbiter는 모든 데이터의 흐름을 제어하며, SDRAM 제어부는 SDRAM 데이터 입출력을 제어한다.

2. 하드웨어 소프트웨어 분할

하드웨어-소프트웨어 통합설계를 위해서 단지 소프트웨어만으로 구성된 코덱을 먼저 구현하고 실시간 처리를 위해서 계산량에 걸림돌이 되는 부분은 하드웨어로 처리하는 방식을 취하였다. 64 kbps 정도의 낮은 전송률에서 QCIF 크기의 영상의 실시간 처리를 목표로 하여 어느 부분을 하드웨어로 수행할지 결정하기 위해서 다음과 같은 추정을 하였다. 소프트웨어로만 구현된 H.263 동영상 코덱의 경우 QCIF, 64 kbps에서 대략 120 ~ 140 MOPS 정도의 계산량이 소요된다 (30 Hz, 전처리 필터의 사용, ±7 탐색영역, 3단계 움직임 추정법)[3]. 그러나 H.263을 위해서 할당된 연산량은ARM 프로세서의 경우 200 MOPS의 30 ~ 40% 정도인 60 ~ 80 MOPS에 불과하다. 그러므로 적어도 60 MOPS 이상의 계산량을 하드웨어 부분에서 해결해야만 한다. 움직임 추정부와 보상부는 계산량이 다른 부분에 비해서 월등히 높기 때문에 하드웨어로 구현되는 것이 효율적이다. 이와는 반대로 VLC (variable length coding) 의 경우에는 표준안마다 상이한 방식을 사용하며, 하드웨어로 제작하기에 균일한 구조를 가지고 있지 않기 때문에 소프트웨어 방식으로 구현하는 것이 바람직하다. DCT (discrete cosine transform) 이나 IDCT (inverse discrete cosine transform) 등도 많은 계산량을 필요로 하지만 낮은 전송률 환경에서 이들의 계산량을 줄일 수 있는 기회가 충분히 많기 때문에 소프트웨어로 구현하는 것이 바람직하다.

Ⅲ. 하드웨어 최적화 기법

1. 움직임 추정부 최적화

하드웨어 최적화에서 움직임 추정부의 최적화는 중요한 위치를 차지한다. 계산량이 많은 전역 탐색 (FSBMA; full search block matching algorithm) 대신 다중의 탐색 후보를 갖는 MRMCS-3라고 불리는 새로운 계층 탐색 기법을 사용한다 (이는 [5]에서 HSBMA3S와 동일하다)[5]. MRMCS-3 방식은 주변 움직임 벡터들의 상관성을 사용하는 방식으로 적은 계산량으로도 전역 탐색에 근접한 성능을 발휘할 수 있다. 표 1은 전역 탐색과 MRMCS-3의 성능을 비교하였다. 그림 2에 MRMCS-3의 동작 개념도를 보였다. MRMCS-3은 4:1로 샘플링 된 상위 영역에서 탐색 범위 ±4로 전역 탐색을 수행하며, 상위 영역에서 최소의 SAD를 갖는 2개의 움직임 변위 후보들을 중간 영역의 초기 탐색점으로 사용한다. 2:1로 샘플링 된 중간 영역에서는 상위 영역의 움직임 변위 후보뿐만 아니라 인접 매크로 블럭에서 구해진 움직임 변위 후보들도 함께 고려한 총 3 개의 후보들에 대해서 ±2 탐색을 수행한다. 중간 영역에서 최소의 SAD를 가지는 하나의 움직임 변위를 초기 탐색점으로 하여 하위 영역에서 ±2의 국부 탐색을 통해 최종 움직임 변위를 결정한다.

표 1. 다양한 QCIF 비디오 동영상에 대한 FSBMA와 MRMCS-3 의 성능 비교 (64 kbps 근처, 30Hz, 300 프레임).

Sequence	QP	FSBMA		MRMCS-3	
		PSNR [dB]	비트율 [kbps]	PSNR [dB]	비트율 [kbps]
Hall monitor	7	35.45	68.85	35.47	68.75
Contain-er ship	7	34.47	71.74	34.47	71.55
Mother & daughter	7	35.98	56.23	36.00	56.31
Foreman	17	29.67	64.83	29.50	65.44
Coast guard	17	28.27	65.43	28.24	65.68
Silent voice	9	32.70	67.29	32.74	67.66

일반적으로 움직임 추정부 하드웨어 구현 시에 이점을 있는 시스톨릭 어레이 (systolic array) 구조

를 제안된 움직임 추정부 하드웨어도 채택하였다. 시스톨릭 어레이 구조에서는 PE (processing elements) 의 개수가 매크로 블럭의 크기 및 탐색 영역의 크기에 비례하게 된다. 따라서 전역 탐색을 위해서는 64 혹은 256 개의 PE가 필요로 하지만 MRMCS-3를 하드웨어로 구현하기 위해서 플립플롭, 멀티플렉서 및 간단한 부가적인 회로를 갖는 PE를 단지 5개만 사용하여 기본 탐색부 (basic search unit; BSU) 를 구현하였다. 기본 탐색부는 그림 3(a) 에서 보듯이 가장 기초가 되는 4×4 화소에 대해 ±2 탐색영역에서 최소의 SAD를 계산할 수 있다. MRMCS-3는 다 해상도의 탐색을 수행하기 때문에 각 영역에서 각기 상이한 크기의 블럭과 탐색영역을 가진다. 그러나 기본 탐색부 (BSU) 를 중복적으로 사용함으로써 하드웨어의 크기를 획기적으로 줄일 수 있다. 기본 탐색부를 중복적으로 사용하는 것이 가능한 것은 MRMCS-3의 계산량이 전역 탐색에 비해 극히 적기 때문에 적은 수의 PE를 순차적으로 사용하여도 원하는 연산 속도를 발휘할 수 있기 때문이다. 상위 영역에서는 ±4 (81 탐색점) 탐색영역을 4 부분으로 나누어서 기본 탐색영역을 4 번 중복적으로 사용한다.

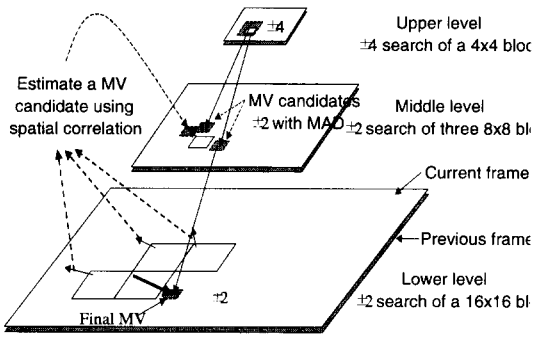


그림 2. MRMCS-3의 개념적 그림.

중간 영역에서는8×8 블럭을 4개의 4×4 화소 단위로 분할하여 기본 탐색부를 12번 사용하며 3개의 초기 탐색점에 대한 탐색을 수행하며, 하위 영역에서는 매크로 블럭을 16개의 4×4 화소 단위로 분할하여 기본 탐색부를 16 번 순차적으로 사용함으로써 탐색을 끝마친다. 중간 영역과 하위 영역의 계산된 SAD들은 4×4 화소의 국부적인 SAD이기 때문에 원하는 최종 매크로 블럭 SAD를 구하기 위해서 현재 구해진 SAD를 일시적으로 저장할 공간과 각 국부 SAD를 더하는 연산이 부가적으로 필요하다.

그림 3(b)는 구현된 움직임 추정부의 블럭도이다. 움직임 추정부는 소프트웨어 최적화에 사용되는 8x8 블록 SAD도 추가적인 계산 없이 구할 수 있으며 이를 호스트쪽으로 전송한다.

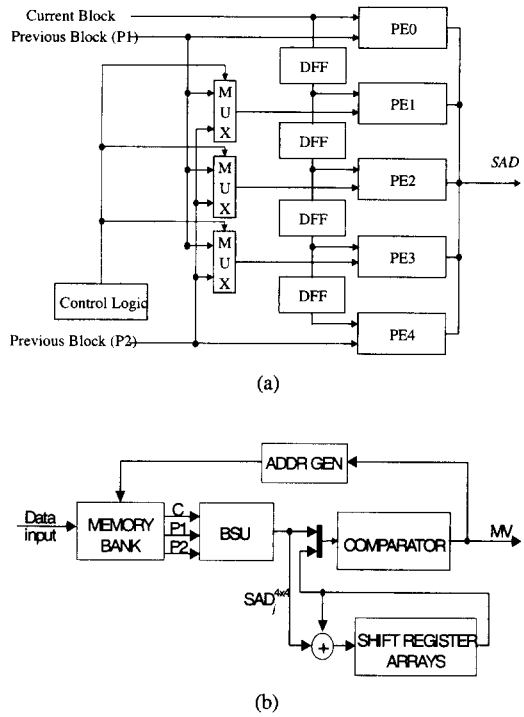


그림 3. 제안된 움직임 추정기의 블록도 (a) 기본 탐색기 (BSU) (b) 기본 탐색기를 기초한 전체적인 움직임 추정기의 블록도.

2. 움직임 보상부 최적화

움직임 보상부는 움직임 추정부와 함께 계산량이 많은 부분 중의 하나이다. 수행해야 할 연산은 고정된 형태이지만, 구조적인 측면에서 소면적 구현이 가능하다.

소면적 구현을 위해서 단 하나의 보간기를 (interpolater) 사용하였다. 움직임 보상이 시작되면 수평 방향 움직임 변위의 반화소 성분에 따라 17 또는 16 개의 화소가 적재된다 (색차 성분에 대해서는 9 또는 8개의 화소). 만약 움직임 변위의 수직 반화소 성분이 0일 경우에는 적재된 값들이 순차적으로 수평축으로 인터폴레이션 되고, 수직 성분을 가질 경우는 이전에 적재되었던 값들을 모두 사용하여 수평 및 수직으로 반화소 성분에 따라서 인터폴레이션 된다. 움직임 보상부는 부호기와 복호기에 모두 사용된다.

Ⅳ. 소프트웨어 최적화 기법

일반적으로 소프트웨어 최적화에는 두 가지의 방향이 가능한데, 하나는 코드 크기 최적화이며 또 다른 하나는 실행 속도 최적화이다. 본 시스템에서는 실시간 처리가 중요한 관심사이기 때문에 속도 최적화를 우선적으로 생각하였다. 속도 최적화도 여러 가지 레벨의 최적화가 가능하지만 무엇보다도 알고리즘의 변형을 통한 최적화가 실행 속도에 절대적인 영향을 끼친다. 하드웨어로 구현된 움직임 추정부 및 보상부를 제외하면 DCT/Q 및 IQ/IDCT가 많은 계산량이 필요하다. 또한 낮은 전송률 환경에서는 IDCT 보다는 DCT가 더 빈번히 수행되기 때문에 그 계산량이 많다. EOB (end of block) 의 예측을 통해서 DCT/Q를 생략법 (skipping) 과 그 계산량을 더 줄이기 위해서 가지치기 (pruning) 방법을 도입한다. EOB 예측을 위해서 사용되는 SAD는 하드웨어로 구현된 움직임 추정부에서 얻어진 것이다.

1. 'Not coded block'예측을 통한 DCT/Q 계 산 감소기법

각종 동영상 표준안이 DCT를 채택한 이후 많은 고속 알고리즘들이 개발되어 왔으며 최근에는 입력 데이터의 통계적 특성에 기반한 고속 방식들이 제안되었다[6-8]. 낮은 전송률 환경에서 동작하는 H.263 코덱에서 DCT 계수가 양자화 이후 모두 0의 값을 가지는 경우가 JPEG과 같은 정지 영상에 비해서 더 빈번히 발생하는데 이 같은 경우 그러한 계산이 실질적으로 불필요하다. 양자화 이후 모든 DCT 계수가 0의 값을 가지는 블럭을 'not coded block' 이라고 부르며, 이 'not coded block' 에 대해서도 불필요한 DCT가 우선 수행될 수 밖에 없다. 'Not coded block' 을 예측하기 위해서 기존의 방법을 사용하여 DCT 계산을 최소화 하였다[6]. 예측을 위한 정보로서 현재 블럭의 최소 SAD값이 사용될 수 있다[9,10]. 인터 모드인 경우 만약 SAD 값이 양자화 계수 (quantization parameter; QP) 가 연관된 문턱치 보다 낮다면 그 블럭은 'not coded block' 으로 간주된다. 기본 탐색부를 중복적으로 사용하는 움직임 추정부는 최소 블럭 SAD를 용이하게 구할 수 있으며 여기서 구한 SAD를 사용하여 'not coded block' 예측에 사용한다. SAD 가 적을 수록 모든 DCT 계수가 양자화 이후 DCT 계수들이 0의 값을 가질 가능성이 더 높다. 그림 4는 휘도 성분 매크로 블럭에 대한 최소 SAD와 'not coded block' 의 계수와의 상관성을 나타낸다. 그림에서 나타나듯이 간단한 문턱치 만으로도 'not coded block' 을 예측할 수 있어서 상당한 계산량을 줄일 수가 있다. 실험적으로 얻어진 20×QP라는 값이 문턱치로 사용하였다.

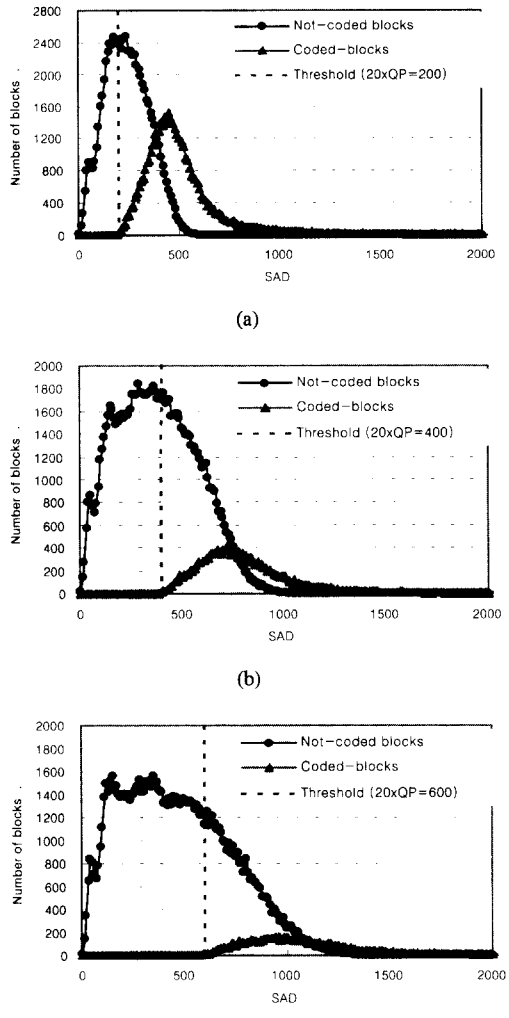


그림 4. coded block 들과 not-coded block들이 가지는 최소 SAD 히스토그램. 주어진 그림은 Foreman QCIF 비디오를 사용하였다 (300 프래임, 30Hz, 고정된 QP). 단지 휘도 성분만 고려되었음.
(a) QP=10, (b) QP=20, 및 (c) QP=30.

DCT/Q 생략법은 색차 성분의 블럭에 대해서도 적용가능 하다. 이 경우 SAD를 소프트웨어적으로 구해야 하는 부담이 있지만, DCT/Q의 예측을 통해 얻는 계산량 이득이 더 크다. 그림 5는 'not coded block' 의 비중을 나타낸다.

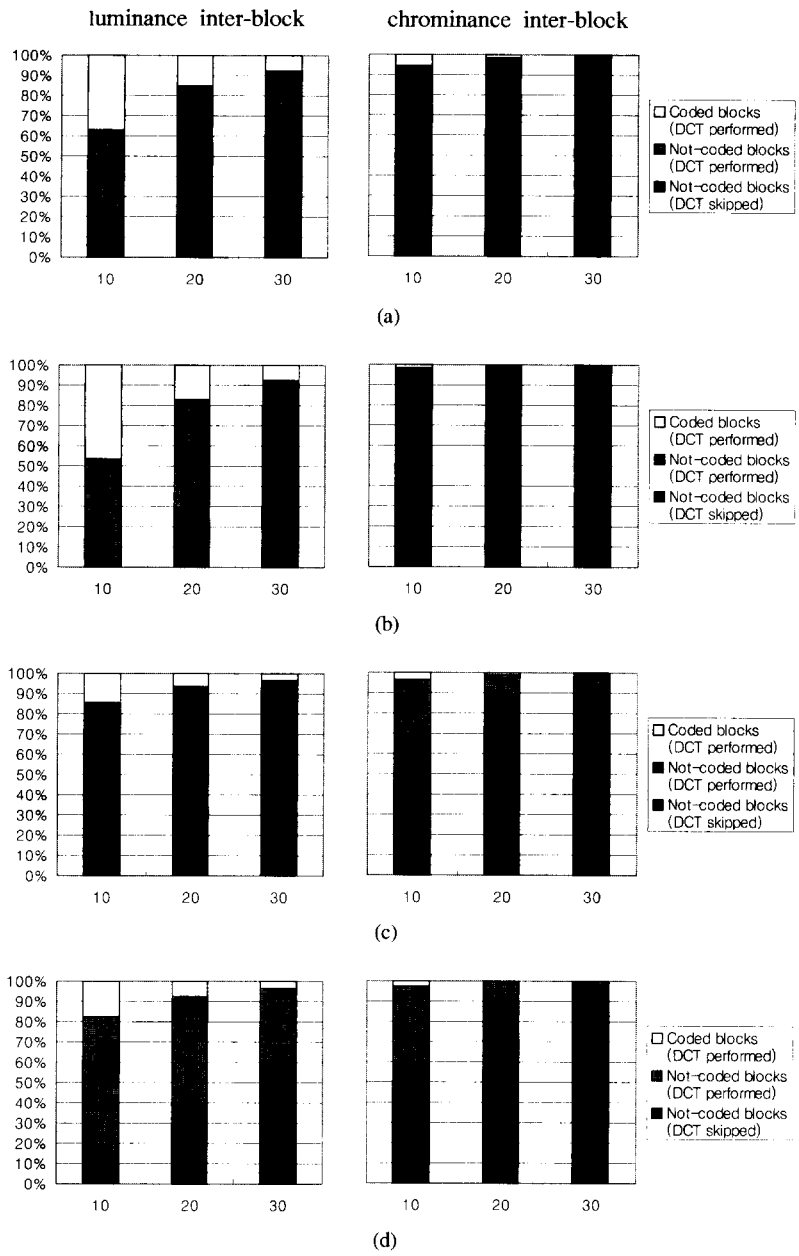


그림 5. DCT 및 양자화를 수행하기 전에 미리 'not coded block' 을 예측한 결과. 가로축 방향의 숫자는 QP를 의미하며 검은색 막대가 'not coded block' 으로 예측된 부분을 나타낸다.
(a) Foreman, (b) Coast guard, (c) News, 및 (d) Silent voice (QCIF, 300 frames, 및 30Hz).

그림 5는 또한 얼마나 많은 'not coded block' 이 미리 예측되는지를 나타낸다. 그림에서 보듯이 전체 'not coded block' 가운데 절반 정도는 예측을 통해 계산량을 줄일 수 있다. 색차 성분의 경우 예측할 수 있는 비율이 더 큼을 알 수 있다. 라는 문턱치를 사용할 때 잘못 예측할 경우가 0.1% 이내임을 확인하였다.

2. EOB 예측을 통한 계산량 감소기법

DCT/Q 계산은 가지치기 (pruning) 방식에 의해 추가적인 계산량 감소가 가능하다. 일반적으로 낮은 전송률 환경에서 많은 양자화된 DCT 계수가 0의

값을 가지게 되며, 지그재그 스캔에서 EOB를 미리 예측할 수 있다면 계산량을 부가적으로 더 줄일 수 있다. DCT 계수 가지치기를 통해서 원하는 영역의 DCT 계수만 계산을 하는 방식이 존재한다[11-12]. 이 논문에서는 이 가지치기 기법을 SAD와 연관시켜 사용하였다. 만약 예측된 EOB가 1×1 사각형 내에 존재한다면, 그 해당 영역 내의 DCT 계수만 계산하며 나머지 고주파 부분은 양자화 이후 모두 0으로 간주한다. 그림 6에서 양자화된 DCT 계수의 분류 방식을 도시하였다. 그림에서 알 수 있듯이 만약 0이 아닌 계수가 모두 4×4 영역에서만 존재한다면 그 블럭은 class 4라고 불린다. 이 경우에 DCT, 양자화 및 variable length coding (VLC)의 계산량을 줄일 수가 있다. 표 2 는 가지치기 기법의 분류 기준을 나타낸다. 그림 7은 DCT 최적화의 전체 알고리즘을 나타낸다. 만약 주어진 블럭의 EOB가 예측되면 양자화 및 VLC를 모든 64개의 계수에 대해서 수행할 필요가 없다.

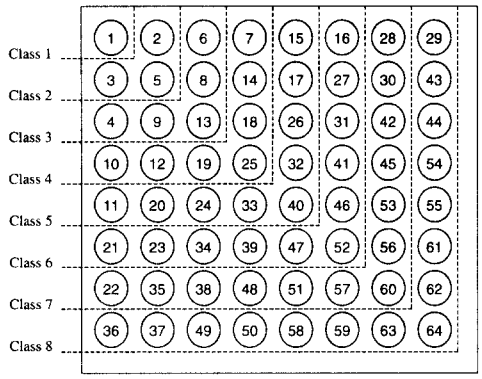


그림 6. DCT 되고 양자화된 88 블럭의 분류도.
동그라미 속의 숫자는 zig-zag 스캔 순서를 나타낸다. 만약 DCT 및 양자화 후 0이 아닌 계수가 모두 $i \times i$ 사각형 이내에 존재 한다면, 주어진 블럭은 class i로 분류된다.

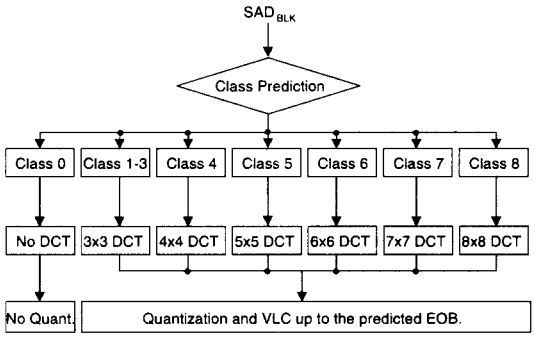


그림 7. DCT 최적화 알고리즘.

표 2. 양자화된 블럭의 분류 기준.

Class	실제적인 DCT 커널 크기	휘도 성분	색차 성분
Class 0	DCT skip	SAD < 20×QP	SAD < 20×QP
Class1-3	3×3	20×QP ≤ SAD < 21×QP	20×QP ≤ SAD < 22×QP
Class 4	4×4	21×QP ≤ SAD < 22×QP	22×QP ≤ SAD < 24×QP
Class 5	5×5	22×QP ≤ SAD < 23×QP	24×QP ≤ SAD < 26×QP
Class 6	6×6	23×QP ≤ SAD < 26×QP	26×QP ≤ SAD < 30×QP
Class 7	7×7	26×QP ≤ SAD < 32×QP	30×QP ≤ SAD < 56×QP
Class 8	8×8	32×QP ≥ SAD	56×QP ≥ SAD

단지 해당 class의 최대 EOB 까지만 양자화를 수행한다. 또한 양자화 연산에서도 양자화에 필요한 나누기 연산은 2.5xQP와의 비교하는 연산으로 대치하여 계산량을 줄일 수 있다.

3. IDCT/Q 계산량 감소기법

입력 계수의 통계적 특성을 이용하여 IDCT/IQ 계산량을 줄일 수 있는 방안들이 존재한다[6]. 고속 IDCT 연산에서 만약 국부적인 버터플라이의 입력 데이터들이 모두 0일 경우 그 연산을 수행하지 않아도 무방하게 되는데, 이럴 경우 실제적인 IDCT 계산량은 0과의 비교할 때 필요로 하는 계산량과, 실제 버터플라이의 복잡도, 버터플라이 입력 데이터가 0을 가질 확률에 의존한다. VLC 과정에서 0이 아닌 계수들의 위치들을 파악할 수 있기 때문에 입력 데이터가 0의 값을 가지는 것을 조사하는 것은 큰 부담이 되지 않는다. 우리는 DCT 계수가 0의 값을 가지는지 사전에 조사하여 ARM 이라는 RISC 프로세서를 기반으로 IDCT/Q 계산량 감소 방식을 조사하였다. 모든 1차원 행/열 DCT에서 입력 데이터가 0 값을 가지는지 비교하여, 약 32% 의 계산량 감소를 얻을 수 있었다. 또한, 만약 EOB가 3 보다 작다면 간략화된 IDCT를 수행할 수 있다. 그림 8은 Foreman QCIF 동영상의 EOB의 통계적 특성을 나타낸다. 그림에서 보듯이 많은 블럭들이 'not coded block' 로 결정되었으며 'coded block' 의 20% ~ 50% 정도가 3 이하의 EOB를 가진다.

그러므로 이 경우 단순화된 IDCT를 통해서 계산량을 추가적으로 줄일 수 있다. 그림 9는 전체적인 IDCT 수행 방식을 나타낸다.

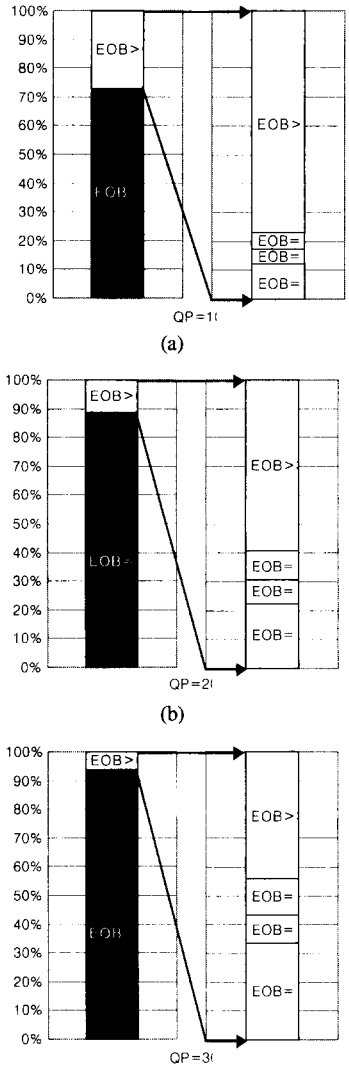


그림 8. Foreman QCIF 동영상에 대한 EOB 값의 분포 (고정된 QP, 30Hz, 300 프래임). EOB 값이 0 이라는 것은 그 블럭이 not coded 임을 의미한다.

V. 하드웨어 소프트웨어 스케줄링

잘 정의된 소프트웨어와 하드웨어의 동시 처리는 또 하나의 중요한 구현 문제이다. 간단하지만 효과적인 스케줄러가 동영상 코덱을 위해서 마련하였다. 스케줄러는 ME/MC 부를 제어할 수 있는 특별한 레지스터를 관장 할 수 있게 되어있다. Start, abort, stop 및resume 와 같은 명령을 정해진 레지스터로 전달하면 연속해서 k 매크로 블럭을 처리한 후 인터럽트가 호스트로 전송된다. 여기서 k 는 동작 카운터에 담겨진 내용으로 한번에 처리해야 할 매크로 블럭의 수를 선택할 수 있도록 하여 스케줄러에게 유연성을 제공한다. 하드웨어와 소프트웨어의 동작만을 검증해 보기 위해서 부호기와 복호기의 스케줄링을 독립적으로 고려하였다. 부호기 과정에서 움직임 추정은 움직임 보상 보다 선행되어야 하며 DCT나 양자화와 같은 소프트웨어 작업은 이들이 끝난 후에 수행되어야만 한다. 이러한 조건들을 고려해 주기 위하여 5 개의 특별한 레지스터를 두었다 (MBIHME, MBIHMC, FlagMEBusy, FlagMCBusy 그리고 MBIS). MBIHME 과 MBIHMC 는 움직임 추정 및 보상이 어느 매크로 블럭부터 재개되어야 할지를 나타내는 인덱스이다. FlagMEBusy 과 FlagMCBusy 는 움직임 추정 및 보상 하드웨어의 동작하고 있는 상태를 나타낸다. MBIS 는 소프트웨어 작업이 어디서부터 시작해야 할지를 나타낸다. 그림 10 은 부호기에서 기본적인 스케줄링 과정을 보여준다 (k 가 1일 경우에 해당). 그림에서 보여지듯이 하드웨어와 소프트웨어 작업은 동시적으로 수행될 수 있다. 소프트웨어 작업을 수행하기 위해서 대기하는 시간은 하드웨어 작업이 얼마나 빨리 수행 되느냐에 달려 있다. 복호기의 스케줄링도 유사하게 구현된다.

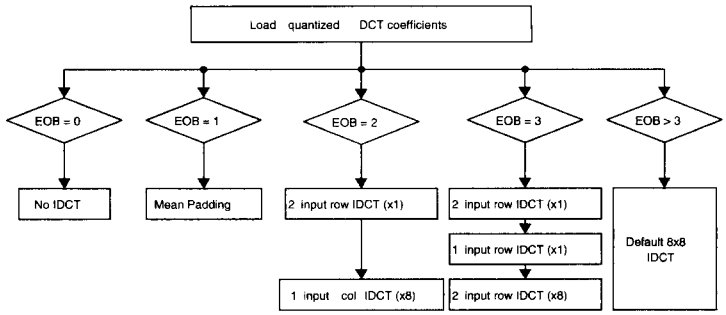


그림 9. IDCT 최적화 알고리즘.

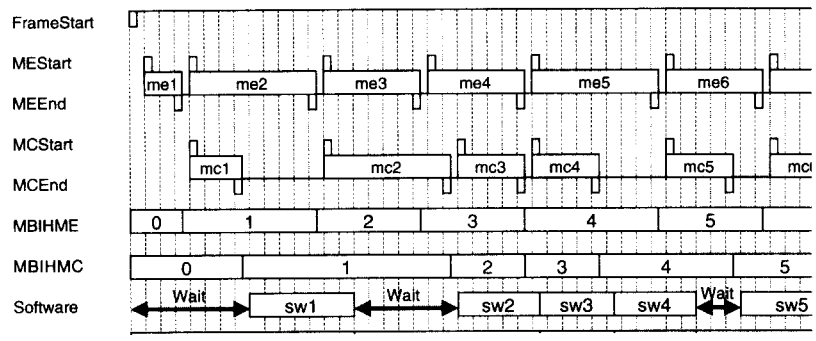


그림 10. 구현된 부호기에서 하드웨어 소프트웨어 스케줄링 방식.

Ⅵ. 구현 결과

하드웨어를 가변적으로 구성할 수 있는 APTIX라 불리는 에뮬레이션 보드에서 개발된 시스템을 검증하였다. APTIX위에 Strong ARM 110 프로세서와 두개의 FPGA (field programmable logic gate array) 를 구성하여 제작된 나머지 하드웨어를 실장할 수 있도록 하였다. 구현된 하드웨어 모듈은 움직임 추정부 및 보상부와, SDRAM 제어부, 호스트 인터페이스, DMA 그리고 비디오 전, 후처리 부분이다.

부호화를 수행할 때 발생되는 데이터 흐름을 그림 1의 동영상 통신 시스템의 블럭도에서 살펴보면 다음과 같다. 우선 최적화된 소프트웨어는 EPROM 또는 flash memory로 적재된다. 전체 시스템이 초기화되고 나면, 카메라로부터 입력된 RGB 영상이 video decoder를 거치게 되어 YUV 형태로 변환되어서 SDRAM에 적재된다. 그리고 나서 스케줄러가 움직임 추정을 시작하도록 명령한다. 이 때, arbiter는 움직임 추정에 필요한 영상을 움직임 추정부에서만 전용으로 사용하는 SRAM으로 적재한다. 움직임 추정부는 SRAM에 있는 영상을 이용하여 움직임 추정을 수행하게 되고, 움직임 추정이 끝나고 나면 arbiter에게 인터럽트를 발생시켜서 움직임 변위와 블럭 SAD를 재 전송하게 된다. 이 후에 스케줄러는 움직임 보상 명령을 전달하게 되며, 움직임 보상에 필요한 데이터는 역시 arbiter가 공급하게 된다. 움직임 보상을 수행한 결과 영상은 다시 SDRAM 특정 어드레스로 놓여지게 되며, 이렇게 움직임 보상된 영상을 RISC core에서 읽어서 DCT, Q 및 VLC 등의 소프트웨어 작업을 수행하게 된다. 부호화 된 정보들은 SDRAM의 특정 영역에 쌓여 있게 되며, peripheral interface를 거쳐서 통신 채널을 통하여 외부로 전송되게 된다. SDRAM에 놓여 있는 복원된 영상은 video preprocessing을 거쳐서 다시 RGB 형태로 변환되어 화면으로 출력될 수 있도록 되어있다. 복호기에서의 데이터 흐름 과정도 움직임 추정을 제외하면 유사하게 진행된다. FPGA로 구현된 하드웨어들은 지연 특성만 추가적으로 고려하여 표준셀 (standard cell) 로 구현될 수 있다. QCIF 동영상을 30 Hz로 처리하기 위해서 움직임 추정부 하드웨어는 대략 11 MHz, 움직임 보상부는 9.5 MHz 정도이면 충분하다. 움직임 추정부와 보상부를 구현하기 위해 각각 25,000 게이트 및 15,000 게이트가 사용되었다. 구성된 시스템에서 라우팅 디바이스에 의한 지연으로 인해서 동작 주파수가 최대 10MHz 까지로 제한된다. 그리하여 우리는 시스템 클럭을 9.5 MHz로 프로세서 클럭을 86 MHz로 설정하였다. 그래서 단지 하드웨어와 소프트웨어의 동작 검증을 하며, 개발된 시스템의 실시간 동작 가능성을 예측하였다. 실제로 동작할 시스템의 동작 주파수는 66 MHz 이며 프로세서의 동작 주파수는 200 MHz이다. 표 3 은 두 가지 다른 특성을 가지는 영상에 대한 부호기의 프로파일 결과를 나타낸다. 여기서는 10으로 고정된 QP 값을 사용하였고, 기본 사양 (baseline) 으로 100 프레임이 부호화되었다. 실제로는 66 MHz로 동작해야 할 움직임 추정부와 움직임 보상부가 9.5 MHz로 동작함으로 인하여 DCT나 양자화와 같은 소프트웨어 작업이 시작하기 전에 많은 시간을 스케줄러가 하드웨어 작업이 끝나기를 기다리는데 소모한다. DCT나 양자화와 같은 소프트웨어 작업은 입력 영상이 더 복잡하거나 움직임이 많을수록 그 처리 시간이 더 늘어나게 됨을 알 수 있다. 만약 프로세서의 클

럭 주파수가 200 MHz, 시스템의 클럭 주파수가 66 MHz 로 설정되면 현재 구현된 시스템 보다 대략 2.3 ~ 7.0 배 정도는 더 빠르게 동작할 것으로 예상된다.

표 3. 재배치 가능한 에뮬레이션 보드에서 하드웨어 소프트웨어 통합 검증의 결과. 두 가지 다른 영 상이 카메라에서 입력되어 100 프레임이 부호 화되었다. QP는 10 고정되어 있다.

Functions	영상 1		영상 2	
	Time [sec]	Fract-ion [%]	Time [sec]	Fract-ion [%]
색깔 성분 변환 (RGB 4:2:2 에서 YUV 4:2:0 으로)	5.45	37.33	5.45	36.72
소프트웨어 작업의 ME/MC 대기 시간	5.67	38.81	3.04	20.45
SubBlock	0.64	4.41	0.82	5.54
DCT	0.94	6.40	1.47	9.89
Quant	0.25	1.71	0.43	2.91
VLC	0.33	2.28	0.95	6.42
DeQuant	0.03	0.22	0.15	0.99
IDCT	0.16	1.12	0.68	4.57
AddBlock	0.10	0.68	0.38	2.58
나머지	1.01	6.91	1.47	9.91
총 소요 시간 [sec]	14.60		14.84	
부호화율 frames/sec [Hz]	6.97		6.87	
비트율 [kbit/sec]	11.12		60.14	

이것은 memory bandwidth 가 병목현상을 초래하여 시스템의 동작을 제한하는 주요한 요소이기 때문에 전체 동작속도는 프로세서의 동작 주파수 보다는 시스템의 동작 주파수에 더 큰 영향을 받게 된다. 따라서 100 % 의 계산량으로 도달할 수 있는 최대 프래임 율은 만약 하드웨어 작업으로 인한 시간 지연을 무시하게 되면 약 29 Hz (≒ 100 frames/3.5 sec) 및 16 Hz (≒ 100 frames/6.3 sec) 가 된다. 따라서 전체 시스템의 동작 속도가 2.3 ~ 7.0 배 빨라지게 된다면, 전체 연산량은 1/7.0~ 1/2.3배로 감소하게 되며, 위 실험 결과 자료를 근거하면 ARM 프로세서의 단지 30 % 내외의 계산량으로 QCIF 영상의 30 Hz로 부호화가 가능한 것을 알 수 있다.

Ⅶ. 결 론

본 논문에서는 하드웨어와 소프트웨어의 통합 설계에 의한 H.263 동영상 코덱을 설계하였다. RISC 프로세서의 연산량을 최대한 줄이기 위해 움직임 추정부와 보상부는 하드웨어로 구현되었다. 하드웨어의 크기를 줄일 수 있는 획기적인 방안들이 모색되었으며, 소프트웨어 최적화에서는 하드웨어에서 얻어진 SAD 정보를 이용하여 DCT/IDCT를 중심으로 몇 가지 최적화 방안들이 모색되었다. 하드웨어와 소프트웨어의 통합 설계에 의해 구현된 H.263 코덱은 전체 프로세서의 계산 능력의 30 % ~ 40 % 만을 필요로 하며 하드웨어는 모두 40,400 게이트 정도 소요되었다. 이 동영상 코덱은 H.324 시스템과 병합될 것이다.

참 고 문 헌

[1] Draft ITU-T Recommendation H.263, "Video coding for low bitrate communication," Mar. 1996.

[2] 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.

[3] W. Lin, K. H. Goh, B. J. Tye, G. A. Powell, T. Ohya, and S. Adachi, "Real time H.263 video codec using parallel DSP," in Proc. Int. Conf. Image Processing, pp. 586-589, 1997.

[4] W. Ding, "A standard-based software-only video conferencing codec on Ultra SPARC," in Proc. SPIE Visual Commun. Image Processing, vol. 3309, pp. 535-542, Jan. 1998.

[5] K.W. Lim and J.B. Ra, "Improved hierarchical search block matching algorithm by using multiple motion vector candidates," IEE Electronics Letters, pp. 1771-1772, Oct. 1997.

[6] K. Lengwehasatit and A. Ortega, "DCT computation with minimal average number of operations," in Proc. SPIE Visual Commun. Image Processing, vol. 3024, pp. 71-82, 1997.

[7] A. Yu, R. Lee, and M. Flynn, "Early detection of all-zero coefficients in H.263," in Proc. Picture Coding Symposium, pp. 159-164, Sept. 1997.

[8] I.-M. Pao and M.-T. Sun, "Approximation of calculations for forward discrete cosine

transform," IEEE Trans. Circuits Systems Video Technol., vol. 8, no. 3, pp. 264-268, June 1998.

[9] S.D. Kim, S.K. Jang, J. Lee, J.B. Ra, J.S. Kim, U. Joung, G.Y. Choi, and J.D. Kim, "Efficient hardware-software co-implementation of H.263 video codec," in Proc. of IEEE Workshop on Multimedia Signal Processing, pp. 305-310, Redondo Beach, CA, Dec. 1998.

[10] Z. Xuan, Y. Zhenghua, and Y. Songyu, "Method for detecting all-zero DCT coefficients ahead of discrete cosine transformation and quantization," IEE Electronics Letters, pp. 1839-1840, Sept. 1998.

[11] Z. Wang, "Pruning the fast discrete cosine transform," IEEE Trans. Commun., vol. 39, no. 5, pp. 640-643, May 1991.

[12] A. N. Skodras, "Fast discrete cosine transform pruning," IEEE Trans. Signal Processing, vol. 42, no. 7, pp. 1833-1837, July 1994.

[13] I.-M. Pao and M.-T. Sun, "Modeling DCT coefficients for fast video encoding," IEEE Trans. Circuits Systems Video Technol., vol. 9, no. 4, pp. 264-268, June 1999.

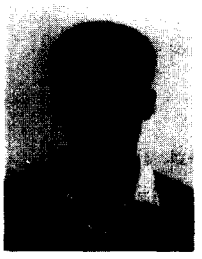
장 성 규(Sung Kyu Jang)

1996년 2월 : 부산대학교 전자공학과 학사
1998년 2월 : 한국과학기술원 전기및전자 공학과 석사
1999년 3월 ~ 현재 : 한국과학기술원 전자전산학과 전기전자전공 박사과정
<주관심 분야> 디지털 영상처리, 영상 압축

김 성 득(Sung Duek Kim)

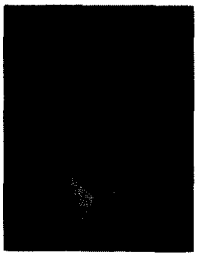
1994년 2월 : 경북대학교 전자공학과 학사
1996년 2월 : 한국과학기술원 전기 및 전자공학과 석사
2000년 2월 : 한국과학기술원 전자전산학과 전기전자전공 박사
<주관심 분야> 디지털 영상처리, 동영상 전후처리 기법

이 재 헌(Jae Hun Lee)



1996년 2월 : 한국과학기술원 전기및전자 공학과 학사
1998년 2월 : 한국과학기술원 전기및전자 공학과 석사
1999년 3월 ~ 현재 : 한국과학기술원 전자전산학과 전기전자전공 박사과정
<주관심 분야> 디지털 영상처리, 움직임 추정 기법 구현

나 종 범(Jong Beom Ra) 정회원



1975년 2월 : 서울대학교 전자 공학과 학사
1977년 2월 : 한국과학기술원 전기및전자 공학과 석사
1983년 2월 : 한국과학기술원 전기및전자 공학과 박사
1987년 7월 ~ 현재 : 한국과학기술원 전자전산학과 교수
<주관심 분야> 디지털 영상처리, 비디오 신호처리, 3차원 시각화, 의료영상처리

김 진 수(Jin Soo Kim)

1995년 2월 : 부산 대학교 전자공학과 학사
1997년 2월 : 부산 대학교 전자공학과 석사
1997년 ~ 현재 : (주)삼성전자, 멀티미디어 연구소 연구원
<주관심 분야> 디지털 영상처리, 하드웨어 구현.

정 의 철(Uichel Joung)

1995년 2월 : 미국 New York Institute of Technology 전자공학과 학사
1997년 2월 : University of Southern California석사
1997년 ~ 현재 : (주)삼성전자, 멀티미디어 연구소 연구원
<주관심 분야> 디지털 영상처리, 하드웨어 구현.

최 건 영(Geon Young Choi)
1990년 2월 : 연세대학교 전자공학과 학사
1997년 2월 : 한국과학기술원 전기및전자 공학과 석사
1997년~현재 : (주)삼성전자, 멀티미디어 연구소 연구원
<주관심 분야> 디지털 영상처리, 하드웨어 구현.

김 종 대(Jong Dae Kim)
1982년 2월 : 서울대학교 전자공학과 학사
1984년 2월 : 한국과학기술원 전기및전자 공학과 석사
1990년 2월 : 한국과학기술원 전기및전자 공학과 박사
1990년~현재 : (주) 삼성전자, 멀티미디어 연구소 연구원
<주관심 분야> 디지털 영상처리, 통신 시스템.