논문 2011-48CI-1-15

ASIP를 이용한 다중 비디오 복호화기 설계 및 최적화

(Design and Optimization of Multi-codec Video Decoder using ASIP)

안 용 조*, 강 대 범**, 조 현 호*, 지 봉 일*, 심 동 규***, 엄 낙 웅****

(Yong-Jo Ahn, Dae-Beom Kang, Hyun-Ho Jo, Bong-Il Ji, Dong-Gyu Sim, and Nak-Woong Eum)

요 약

본 논문은 다양한 비디오 표준의 복호화가 가능한 프로세서를 설계하고, MPEG-2, MPEG-4 및 AVS(Audio video standard)를 이용하여 프로세서의 성능을 검증하였다. 일반적으로 하드웨어 비디오 복호화기는 고속의 복호가 가능하나 설계 및 수정이 어렵다. 반면, 소프트웨어기반의 경우에는 구현이 상대적으로 수월하고 수정이 용이하나, 동작 성능이 낮아 기대하는 속도를 얻기 어렵다. 본 연구에서는 두 가지 연구 설계방법의 장점을 동시에 충족시키는 방법으로 ASIP(Application specific instruction-set processor) 프로세서를 설계하였다. 또한, 비디오 복호화기의 공통 모듈을 연구하여 8개의 모듈로 나누었고, 각 모듈에 공통적으로 적용할 수 있는 다수의 멀티미디어 전용 명령어를 프로세서에 추가하였다. 비디오 복호화기를 위해 개발된 프로세서는 Synopsys 플랫폼 시뮬레이터와 FPGA 보드에서 성능을 평가하였다. 결과적으로 MPEG-2, MPEG-4 및 AVS에 적용하여 평균 37%의 복호 속도를 향상시켰다.

Abstract

In this paper, we present a multi-media processor which can decode multiple-format video standards. The designed processor is evaluated with optimized MPEG-2, MPEG-4, and AVS (Audio video standard). There are two approaches for developing of real-time video decoders. First, hardware-based system is much superior to a processor-based one in execution time. However, it takes long time to implement and modify hardware systems. On the contrary, the software-based video codecs can be easily implemented and flexible, however, their performance is not so good for real-time applications. In this paper, in order to exploit benefits related to two approaches, we designed a processor called ASIP(Application specific instruction-set processor) for video decoding. In our work, we extracted eight common modules from various video decoders, and added several multimedia instructions to the processor. The developed processor for video decoders is evaluated with the Synopsys platform simulator and a FPGA board. In our experiment, we can achieve about 37% time saving in total decoding time.

Keywords: ASIP, 비디오 코덱, 복호화기, 최적화

I. 서 론

최근 멀티미디어 환경과 컴퓨터 하드웨어의 발달로 인하여 우리는 손쉽게 압축된 영상을 부호화 혹은 복호화 할 수 있게 되었다. 이러한 발달과 더불어 고화질, 고해상도의 영상에 대한 요구가 점차 증가하고 있다. 이러한 요구에 따라 고화질, 고해상도의 비디오 콘텐츠를 효율적으로 압축하기 위하여 ITU-T, ISO/IEC와 같은 국제 표준화 기관은 물론, Microsoft(社)와 같은 일반 기업 등에서도 비디오 코덱에 관한 표준화를 다양하게 진행하고 있다.

다양한 단체에서 비디오 압축 기술들이 개발됨과 더

* 학생회원, *** 정회원-교신저자, 광운대학교 컴퓨터공학과
(Dept. of Computer Engineering, Kwangwoon University)
** 정회원, 삼성전자(주)
(Samsung Electronics)
**** 정회원, 한국전자통신연구원 멀티미디어프로세서연구팀
(Multimedia Processor Research Team, Electronics and Telecommunications Research Institute)
※ 본 연구는 지식경제부의 MPcore 플랫폼 기반 다중 포맷 멀티미디어 SoC사업의 일환으로 수행하였음.
접수일자:2010년9월10일, 수정완료일:2010년12월27일

불어 어플리케이션의 목적과 특징에 따라 비디오 코덱들이 사용되고 있다. 예를 들어, MPEG-2는 디지털 방송과 DVD에서 사용되고 있으며, MPEG-4는 VoD, H.264/AVC는 DMB, VC-1은 비디오 스트리밍 및 Blue-ray에서 대표적으로 사용된다[1]. 또한, 비디오 코덱의 종류가 다양해지면서 하나의 하드웨어로 여러 개의 비디오 코덱을 처리할 수 있는 기술에 대한 요구가 증가하고 있다. 지금까지는 단일 비디오 코덱에 적합한 하드웨어 구조 및 동작에 대한 연구는 활발히 이루어져 왔으나, 여러 개의 비디오 코덱을 하나의 하드웨어에서 효과적으로 처리하기 위한 연구는 부족하였다.

본 연구에서는 다양한 비디오 코덱을 하나의 하드웨어에서 효과적으로 처리할 수 있는 다중 비디오 복호화기의 설계 및 최적화를 제안한다. 이를 위하여 비디오 코덱들의 기능별 유사성을 토대로 공통 모듈을 분류 및 설계하고 이에 대한 최적화를 진행하였다. 또한, ASIP(Application specific instruction-set processor)를 이용하여 비디오 코덱을 위한 전용 명령어를 추가함으로써 각 모듈에 대한 최적화를 수행하였다.

ASIP는 특정 어플리케이션에 적합한 프로세서를 설계하는 방법 혹은 그러한 방법으로 설계된 프로세서를 의미한다. 그림 1은 하드웨어 설계방법에 따른 복잡도 및 성능의 관계를 나타낸다. 그림 1에서 나타나는 바와 같이, ASIP는 주문형 반도체의 저비용, 저전력, 고성능과 범용 프로세서의 유연성 (Flexibility)이 결합된 새로운 형태의 프로세서이다. 해당 프로세서는 특정 어플리케이션의 처리를 위하여 설계되지만 주문형 반도체와는 달리 특정 명령어 세트와 레지스터 파일 및 내부 인터페이스 등을 포함하고 있으며 고유의 컴파일러를 가진 프로그램 가능 가속기 (Accelerator)이다[2].

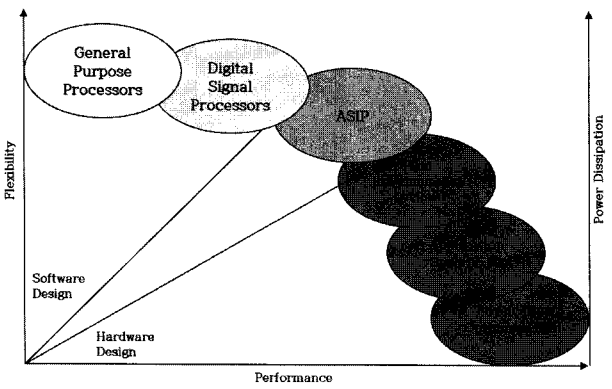


그림 1. 프로세서 설계방법에 따른 특징 비교
Fig. 1. Comparisons according to the processor design methodology.

본 연구에서는 ASIP 설계를 위하여 Synopsys(社)의 Processor designer[3]를 이용하였으며 명령어 구현을 위해 LISA(Language for instruction-set architectures)[4]를 사용하였다. 또한, 본 연구의 성능 평가를 위하여 MPEG-2, MPEG-4, AVS 비디오 복호화기를 이용하였으며, ASIP 설계 후 각 비디오 복호화기는 소프트웨어의 알고리즘 최적화 및 C-레벨에서의 코드 최적화를 적용하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 연구에 사용된 MPEG-2, MPEG-4, AVS 비디오 표준에 대하여 간략히 알아보고, 이러한 비디오 표준을 구현함에 있어 DSP (Digital signal processor)와 GPP (General purpose processor)를 이용한 소프트웨어 구현과 ASIC(Application specific integrated circuit)과 같은 하드웨어 구현에 대한 장/단점을 소개한다. Ⅲ장에서는 제안하는 ASIP를 이용한 다중 비디오 복호화기의 설계에 대하여, 기능별 모듈화 설계, 공통 명령어 설계 및 적용, C-레벨 최적화 방법에 대하여 차례로 기술한다. Ⅳ장에서는 비디오 표준의 기본 성능, C-레벨 최적화와 제안한 명령어 적용 후의 성능을 각각 비교하며, 마지막으로 Ⅴ장에서는 결론 및 향후 연구에 대하여 논의한다.

Ⅱ. 기존의 비디오 표준 구현

본 연구에서 구현한 비디오 표준은 MPEG-2, MPEG-4, AVS로 이번 장에서는 각 표준의 특징과 부호화 효율을 높이기 위한 기술들을 간략하게 살펴보기로 한다. 또한, 비디오 표준의 소프트웨어 구현과 하드웨어 구현의 장, 단점에 대하여 기술한다.

1. 기존 비디오 표준의 특징

MPEG-2 복호화기는 MPEG-1과의 역 호환성을 갖으며, 화면 내 예측시 비트 깊이의 절반 값을 예측 값으로 사용하는 특징을 갖는다. 화면 간 예측을 위한 화소의 보간에는 양선형 보간(Bi-linear interpolation)을 이용하며, 움직임 보상을 위해 16×16, 16×8, 8×16 블록 크기의 모드를 지원한다. 또한 가변 길이 복호화 과정을 거쳐 나오는 변환 계수들은 그림 2와 같이 두 가지의 스캐닝 모드를 사용하여 정렬한다.

그림 2에서 나타난 얼터네이트 스캔(Alternate scan) 방식은 지그-재그 스캔(Zig-zag scan) 방식에 비하여

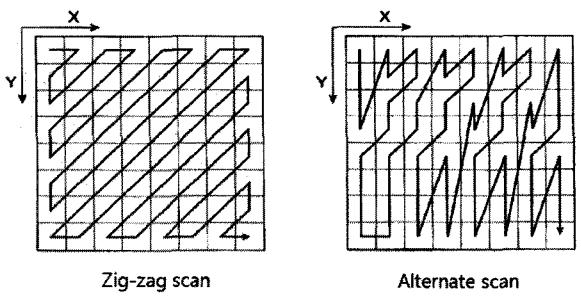


그림 2. MPEG-2 스캐닝 순서
Fig. 2. MPEG-2 scanning order.

이산여현변환 영역에서 수직 방향의 고주파 성분을 상대적으로 일찍 주사하는 방식으로 움직임이 큰 비월주사 방식에서 효과를 보인다.

MPEG-4는 기존의 MPEG-비디오 표준에 대한 하위 호환성을 갖지 않으며, 고압축률 부호화를 위하여 개발되었으며, 휴대 단말 및 휴대용 TV에서 주로 사용된다[5]. 부호화를 위한 세부 기술들로는 부호화 효율 향상을 위한 1/4 화소 정밀도를 갖는 움직임 탐색, B 프레임, 영상의 크기를 벗어나는 움직임 벡터까지 지원할 수 있는 UMV(Unrestricted motion vector)등의 사용이 있다.

AVS는 중국에서 자체 제작된 비디오/오디오 표준으로써 현재 사용되고 있는 가장 우수한 비디오 코덱인 H.264/AVC와 유사한 성능을 갖는 것으로 알려져 있다. 하지만 AVS는 H.264/AVC 보다 상대적으로 낮은 복잡도를 갖으며, 기존의 비디오 표준과 다른 부호화 방식을 채택하여 로열티를 낮추었다[6]. H.264/AVC와 다르게 기본적으로 8×8단위의 화면 내 예측 및 화면 간 예측을 사용하고 같은 크기의 정수화된 이산 여현 변환을 사용한다. 화면 간 예측 시에는 4-tap 필터를 이용하여 보간을 수행하며, 이를 통해 H.264/AVC의 6-tap 필터에 비해 복잡도를 낮췄다.

2. 비디오 표준의 소프트웨어 및 하드웨어 구현

일반적으로 비디오 표준을 소프트웨어로 구현하기 위해서는 GPP 및 DSP와 같은 프로세서를 이용한다. GPP의 경우 높은 클럭 수를 갖지만 프로세서 설계의 목적자체가 특정 어플리케이션이 아닌 보편적이고 일반적인 다수의 어플리케이션에서 좋은 성능을 내는 것이기 때문에, 고속 비디오 복호화를 위한 추가적인 명령어를 지원하지 않는다. 또한, DSP 에서는 신호처리에 유용한 명령어의 사용이 가능하고, SIMD(Single instruction multiple data) 명령어에 대한 처리가 가능하다[7]. 비디오 표준을 위한 명령어 추가가 이루어지고 있기는 하지만, 비디오 복호화기에서 사용하지 않는 명령어가 많고 원하는 명령어의 사용을 위해서는 기본적인 명령어 조합이 선행되어야 한다. 예를 들어, 복원 영상을 생성하기 위하여 화소 값을 일정한 범위로 조절하는데 사용할 수 있는 클리핑(Clipping) 연산에 대하여 살펴보면, 클리핑 연산을 수행하기 위해서는 min 명령어(입력 값 중 최솟값을 반환하는 명령어)와 max 명령어(입력 값 중 최댓값을 반환하는 명령어)의 조합을 사용해야한다.

비디오 표준을 하드웨어로 설계하는 경우 빠른 성능과 적은 전력 소비량의 장점을 갖는다. 하지만, 초기 개발 시간과 비용이 높고, 특정 목적에 최적화되어있기 때문에 재사용성과 융통성이 낮다는 단점을 갖는다. 또한, 추가적인 기능이 요구되면 이에 대한 수정에도 많은 시간이 소요된다[8].

따라서 이러한 문제점들을 해결하기 위하여 최근에는 하드웨어기반 기술의 장점인 높은 성능과 소프트웨어 기반 기술이 장점인 융통성을 결합시키는 기술에 대한 많은 연구들이 이루어지고 있다. 논문 [9]에서는 H.264/AVC를 위한 ASIP 프로세서를 개발하고 최적화하였다. 또한, 비디오 부호화에 필요한 몇 가지 명령어를 추가함으로써 비디오 부호화 성능을 향상시켰다. 하지만, 논문 [9]에서 제안하는 방법의 경우 H.264/AVC 복호화기를 포함하여 다른 표준 비디오 복호화기에서도 공통적으로 사용하기 힘들다는 단점이 있다.

III. 제안하는 공통 블록 구조의 ASIP 비디오 복호화기 설계

1. 공통 블록 구조의 모듈화 설계

MPEG-1/2부터 최근의 H.264/AVC에 이르기까지 비디오 코덱 등은 화면 내 예측 및 화면 간 예측, 변환 및 양자화 그리고 가변 길이 부호화를 통하여 압축의 성능을 향상시켜 왔다. 초기에 비해 현재 비디오 코덱들은 다양한 압축 기술을 제공하지만 비디오 복호화기의 기본 구조는 각 기능에 의한 블록들로 구분하여 나타낼 수 있다. 본 연구에서는 기능별 모듈로 분할된 복호화기를 제안하며, MPEG-2, MPEG-4, AVS 복호화기를 이용하여 성능을 평가하였다.

공통 블록 구조의 모듈화 설계를 위하여 전술한 MPEG-2, MPEG-4, AVS 복호화기를 그림 3과 같이

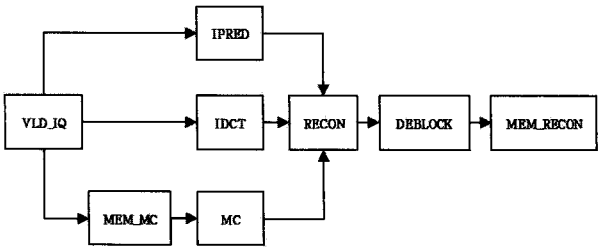


그림 3. 제안하는 공통 블록 비디오 복호화기의 블록 다이어그램

Fig. 3. The block diagram of the proposed common block video decoder.

기능별 모듈로 분할하였다. 기능별 모듈로 분할할 경우 비디오 복호화기에서 각 모듈의 복잡도를 쉽게 측정할 수 있다. 복잡도 측정은 비디오 복호화기의 최적화에 있어 선행되어야 할 필수 요소이며, 비디오 복호화기에 대한 분석에 해당한다. 기능 모듈별 복잡도 분석은 복잡도 높은 모듈에 대한 하드웨어 구현을 용이하게 해준다.

본 논문에서는 비디오 복호화기를 8개의 기능별 모듈로 분할하였으며, 각 모듈은 다음과 같은 역할을 수행한다. 먼저, VLD_IQ 모듈은 가변 길이 복호화 및 역양자화를 수행한다. 각각의 비디오 코덱 표준마다 복호화 방법의 차이는 있으나 공통적으로 VLD_IQ 모듈의 입력은 부호화된 비트스트림이고 출력은 복호화 정보가 저장된 변수들이다. 가변 길이 복호된 계수들과 문법 요소는 각각 변수나 메모리에 저장하게 된다. IPRED 모듈은 화면 내 예측 모드를 바탕으로 예측 블록을 생성하는 모듈이다. IPRED 모듈의 입력은 VLD_IQ 모듈로부터 복호화된 화면 내 예측 모드 및 주변 블록의 복원 화소들이며, 출력은 주변 블록을 바탕으로 생성한 예측 블록이다. IDCT 모듈은 역 이산 여현 변환을 수행하는 모듈로써 VLD_IQ 모듈에서 역양자화 된 계수들을 입력을 받고, 역 이산 여현 변환된 차분 영상을 출력한다. 역 이산 여현 변환에서 MPEG-2와 MPEG-4는 고정소수점 방식으로 구현되었으며 AVS에서는 정수 역변환으로 구현되어있다. MEM_MC 모듈은 움직임 보상에 사용될 움직임 벡터를 생성하고, 이 움직임 벡터를 이용하여 참조 프레임에서 예측 위치에 해당하는 화소들을 메모리로부터 획득한다. MEM_MC 모듈의 입력으로는 예측 움직임 벡터 및 현재 블록의 모드가 있으며, 출력으로는 예측 위치에 해당하는 화소정보가 있다.

표 2. 공용 명령어 리스트

Table 2. The list of common instruction.

명령어	인트린직 함수	세부 동작	사용 모듈
lmax	lmax_ckf	두 32비트 정수를 입력받아 둘 중 최댓값을 32비트 결과 레지스터에 저장	VLD_IQ
lmin	lmin_ckf	두 32비트 정수를 입력받아 둘 중 최솟값을 32비트 결과 레지스터에 저장	VLD_IQ
clz	clz_ckf	입력 레지스터의 MSB(Most significant bit)부터 연속되는 0의 개수를 결과 레지스터에 저장	VLD_IQ
labs	labs_ckf	두 32비트 정수를 입력받아 두 정수 차의 절댓값을 32비트 결과 레지스터에 저장	VLD_IQ
labs_4	labs_4_ckf	8비트 정수 4개로 구성된 32비트 워드 2개를 입력받아 차의 절댓값을 32비트 결과 레지스터에 저장	MC, VLD_IQ
lclip	lclip_ckf	범위를 지정하는 최댓값과 최솟값의 두 32비트 정수와 적용하는 32비트 정수 하나를 입력받아 클리핑 연산을 수행	VLD_IQ, IDCT
smul_4	smul_4_ckf	부호가 있는 8비트 정수 4개로 구성된 32비트 워드 2개를 입력받아 둘 간의 곱의 합을 구하여 32비트 결과 레지스터에 저장	MC, VLD_IQ
umul_4	umul_4_ckf	부호가 없는 8비트 정수 4개로 구성된 32비트 워드 2개를 입력받아 둘 간의 곱의 합을 구하여 32비트 결과 레지스터에 저장	MC, VLD_IQ
bext	bext_ckf	플래그가 1인 경우 MSB를 바이트 확장하고, 플래그가 0인 경우 LSB를 확장하여 결과 레지스터에 저장	Padding
bilf	bilf_ckf	두 32비트 정수를 입력받아 양선형 필터링을 수행	MC
add_clip4	add_clip4_ckf	16비트 정수 2개로 구성된 32비트 워드 2개와 8비트 정수 4개로 구성된 32비트 워드 1개를 입력받아 각각 합을 구한 결과에 클리핑연산을 수행하여 결과 레지스터에 저장	RECON

MC 모듈에서는 움직임 보상을 위한 화소간의 보간(Interpolation) 수행 및 화면 간 예측에서의 예측 영상을 생성하는 역할을 수행한다. 이때, 입력은 MEM_MC에서 전달된 예측 위치에 해당하는 화소정보이며, 출력은 보간된 화소로 생성한 예측 블록이다. RECON 모듈은 화면 내/간 예측을 통하여 얻은 예측 블록과 IDCT 모듈의 출력인 역 변환을 통해 출력된 블록간의 덧셈을 수행하는 모듈로 역 변환을 통해 출력되는 블록의 값은 16비트이며 예측된 블록은 8비트 값을 갖는다. 이렇게 더해진 매크로블록에 대하여 표준에 따라 블록킹을 제거하기 위한 필터링이 수행된다. MPEG-2와 MPEG-4는 복원 후 디블록킹 수행이 없으나, AVS는 디블록킹 필터가 루프 내 필터(In-loop filter)로 구현되어 있어, DEBLOCK 모듈을 추가로 수행한다. 이 후, 복원 완료된 블록을 프레임 메모리로 복사해 주는 과정이 MEM_RECON(혹은 MEM_DEBLOCK) 모듈이다.

MEM_RECON 모듈과 MEM_MC 모듈이 별도로 존재하는 이유는 하드웨어로 구현되었을 때, 내부 메모리와 외부 메모리간의 데이터 교환 유무를 명시하기 위함이다. 비디오 복호화기의 복원 영상을 저장하기 위한 버퍼 (DPB : Decoded picture buffer)의 경우 하드웨어로 구현할 경우 내부 메모리로 처리하기 힘든 데이터양을 갖기 때문에 일반적으로 외부 메모리를 사용한다. 이 경우 외부 메모리와 내부 메모리간의 데이터 교환이 필요한데, 모듈화 단계에서 데이터 교환이 이루어지는 모듈을 명시해 줌으로써 하드웨어 설계 단계에서 추가적인 메모리 교환 장치 (예를 들어, DMA 장치)의 필요여부에 대한 판단을 쉽게 할 수 있다.

본 논문에서는 기능별로 모듈화된 비디오 복호화기를 각 모듈단위로 복잡도 측정하고, 복잡도가 높은 모듈에 대하여 고속화 명령어추가, C-레벨 최적화 및 알고리즘 최적화의 과정을 수행하였다.

2. 공용 명령어 연구 및 최적화

전술한 바와 같이 모듈화된 비디오 복호화기는 표준에 따라 각기 다른 연산을 수행하지만 입/출력은 유사한 특징이 있다. 예를 들어, IDCT 모듈에서는 CBP (Coded block pattern)에 따라 블록단위로 역 이산 여현 변환 수행 여부를 결정하며, MC 모듈에서는 참조블록의 화소에 표준별 특정 보간 필터(Interpolation filter)를 적용하여 보간한다는 유사성을 갖는다.

따라서 본 논문에서는 다양한 비디오 복호화기를 적용할 수 있는 프로세서를 위하여 모듈간의 유사성을 기반으로 공용 명령어를 추가하였다. 표 2는 본 연구에서 사용하는 공용 명령어를 나타낸다.

공용 명령어는 기존 GPP나 DSP에서 사용되었던 명령어 중 비디오 복호화 연산에 필요한 명령어들을 포함하였다. 표 2에서 언급한 lmax_ckf, lmin_ckf, labs_ckf, labs_4_ckf, smul_4_ckf, umul_4_ckf, clz_ckf 명령어는 기존에 DSP에 존재하는 명령어로 비디오 복호화에서 계산량 감소를 위해 유용하게 사용할 수 있다. 예를 들어, clz 명령어를 사용하면 32비트의 데이터가 입력되었을 때 첫 번째로 0이 아닌 비트의 위치를 구하는 연산을 한 사이클에 처리할 수 있다. 그림 4는 clz 명령어가 대체할 수 있는 연산을 나타낸다.

본 논문에서는 인접한 화소에 대하여 효과적인 연산을 수행하기 위하여 SIMD형태의 명령어를 사용하였다[9]. smul_4와 umul_4는 32비트로 패킹 (Packing)되어 있는 4개의 8비트 값을 다른 32비트 패킹된 4개의 8비트 값과 각각 곱한 후 더해주는 역할을 하며, 부호가 있는 입력 값에는 smul_4를, 부호가 없는 입력 값에는 umul_4를 사용한다. smul_4와 umul_4 명령어의 사용으로 화면 내 예측이나 디블록킹 필터와 같은 필터링 연산에서 인접한 화소에 대한 고속 연산이 가능하다. 아래 그림 5는 smul_4 명령어의 연산 방식을 나타낸다.

본 논문에서는 비디오 복호화의 성능을 향상시키기 위하여 lclip_ckf, bext_ckf, add_clip4_ckf 명령어를 추

```
leadingZeroBits = -1;
for(b = 0; !b; leadingZeroBits++)
    b = read_bits(1);
```

그림 4. clz 연산
Fig. 4. clz operation.

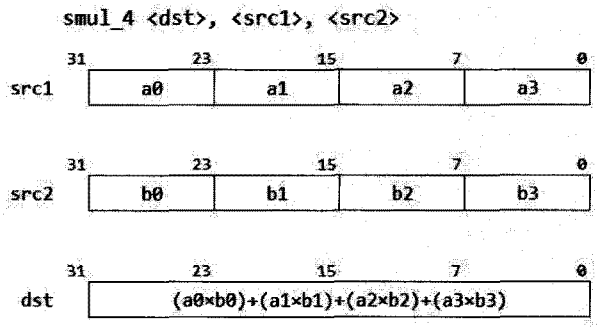


그림 5. smul_4 명령어
Fig. 5. smul_4 instruction.

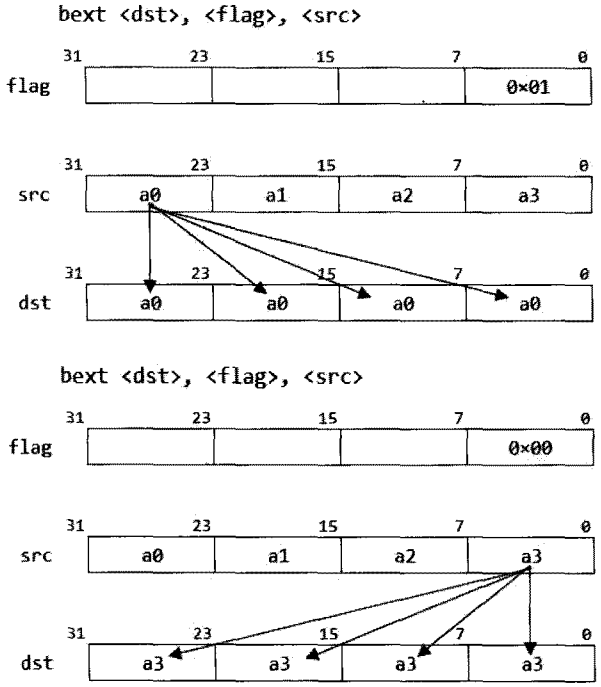


그림 6. bext 명령어
Fig. 6. bext instruction.

가하였다. 먼저 lclip_ckf 명령어는 입력된 최솟값과 최댓값 사이로 변수를 클리핑 해주는 명령어로서 특히 역이산여현변환 과정에서 발생하는 범위를 벗어나는 값들을 보정하는데 주로 사용된다. bext 명령어는 그림 6과 같이 선택된 8비트 데이터를 연속된 4개의 8비트 데이터로 복사해주는 역할을 한다. 이 명령어는 주로 프레임의 경계 영역을 패딩할 때 하나의 화소를 주변 화소로 복사하는데 사용된다. 이 명령어의 입력으로 32비트 워드 데이터와 복사 대상을 결정하는 플레그 (flag)비트를 사용한다. 플레그 비트가 1인 경우 상위 8비트의 값을 0인 경우 하위 8비트 값을 4번 복사한다.

add_clip_4 명령어는 32비트로 패킹되어 있는 4개의 8비트 데이터 1개와 32비트로 패킹되어 있는 2개의 16비트 데이터 2개를 이용하여 16비트 데이터 와 8비트 데이터 각각을 덧셈 후 클리핑 연산을 수행한다. 이 명령어는 RECON 모듈에서 복원 영상을 생성함에 있어 8비트 예측 영상과 16비트 차분 영상을 더한 후 클리핑 연산을 수행하는 과정에서 사용하여 계산 량을 감소시키는데 주로 사용된다. 그림 7은 add_clip_4 명령어를 나타낸다.

이러한 명령어는 소프트웨어에서 손쉽게 사용할 수 있도록 인트린직(Intrinsic)으로 구현하였다. 인트린직 명령어는 어셈블리 명령어를 C언어에서 호출할 수 있

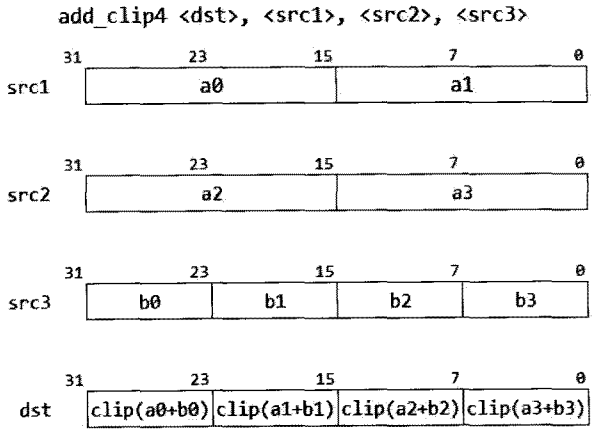


그림 7. add_clip4 명령어
Fig. 7. add_clip4 instruction.

도록 컴파일러에게 명시해 줄 수 있기 때문에 어셈블리어로 기술한 것과 동일한 효과를 얻을 수 있다.

3. 소프트웨어 최적화

ASIP는 특정 어플리케이션에 최적화된 프로세서이지만, 추가된 명령어만으로 성능을 향상시키는 것에는 한계가 있다. 또한, 프로세서는 명령어를 패치한 후 수행시키는 구조이므로 기본적으로 동작되는 소프트웨어의 알고리즘 최적화와 더불어 C-레벨에서의 코드 최적화가 필수적이다.

본 논문에서는 인트린직 형태의 명령어 사용 외에 루프 언롤링(Loop unrolling)이나 루프 스위칭(Loop switching)과 같은 방법으로 루프를 최적화 구현하였으며 마스크를 사용하여 분기문을 제거함으로써 명령어가 파이프라인에서의 스톨 (Stall)을 최소화 시켰다. 또한, 구조체 변수 및 지역 변수의 사용을 줄이고 불필요한 메모리 및 변수를 제거하여 구조적인 최적화를 이루었다.

표 3. 각 영상별 비트레이트
Table 3. Bitrate for each sequence.

영상	부호화 구조	비트레이트 (kbit/s)
Miss America	IPPP	300
Deadline	IPPP	600
Foreman	IPPP	900
Paris	IPPP	1200
Coastguard	IPPP	2300
Tempete	IPPP	2500
Funfair	IPPP	3400
Mobile	IPPP	3600

불필요한 분기문 및 메모리 제거뿐만 아니라, 8비트 화소 단위로 동작하는 비디오 표준의 특성으로 인한 잦은 메모리 참조를 줄이기 위하여 SIMD 형태로 데이터를 처리하도록 변경하는 작업을 진행하였다. 예를 들어, 네 화소씩 하나의 32비트 정수형 포인터를 사용하여 메모리를 접근하게 함으로써, 프로그램의 동작 속도를 향상시킬 수 있었으며 이러한 과정에서 분기문의 수도 감소하는 효과를 얻을 수 있었다.

Ⅳ. 실험 결과 및 토의

본 논문의 실험을 위하여, MPEG-2, MPEG-4 및 AVS 3개의 코덱을 구현하였으며, 복호화 성능 실험을 위한 입력 비트스트림 생성은 표준 부호화기를 사용하였다. 실험에 사용한 영상은 CIF (352×288)급 영상 8개로, 각 비디오 표준별로 QP 범위가 다르기 때문에 실험결과는 비트레이트(Bitrate)로 표기하였다. 실험에 사용된 영상별 비트레이트는 표 3과 같다. GOP(Group of picture) 구조는 IPPP의 동일한 조건으로 실험을 진행하였다. 또한, 본 논문의 실험에서는 ASIP로 설계된 VLIW-2i 프로세서를 이용한 다중 비디오 복호화기 플랫폼을 이용하였다. ASIP로 설계된 프로세서는 Ⅲ장에서 제안한 비디오 고속 복호화를 위한 명령어-셋(Instruction-set)이 추가되었으며, 2개의 슬롯을 가진 VLIW(Very long instruction word) 구조를 기반으로 개발되었다. 32비트 단위의 명령어는 병렬적으로 최대 2개씩 묶여서 패치(Fetch) 되고 프로그램과 데이터는 각각의 메모리를 가지는 구조이다. VLIW-2i 프로세서는 PF(Pre-fetch), FE(Fetch), DC(Decode), EX(Execute), MEM (Memory), WB (Write back)의 6단계로 구성되어있고, GPR(General purpose register)은 16개와 각각 1개의 ZR(Zero register), SPR(Stack pointer register) 와 FPR(Frame pointer register)가 포함되어 있다.

그림 8은 ASIP로 설계된 VLIW-2i 프로세서와 주변 장치들로 이루어진 ASIP 플랫폼의 블록 다이어그램을 나타낸다. ASIP 플랫폼은 메모리 컨트롤러(Memory controller)를 이용하여 프로그램/데이터 메모리(Program/data memory)와 프로세서간의 데이터 교환을 수행하며, 외부 메모리(External memory)와 데이터 교환은 버스(Bus)를 통해서만 가능하도록 설계되어 있다. 실험에서는 512MB의 외부메모리를 사용하여 DPB의 데이터를 저장하도록 설계하였다.

본 논문의 실험은 최적화 정도에 따라 세단계로 나누었으며 세부적인 내용은 다음과 같다. 최초 각 비디오 표준의 참조 소프트웨어(Reference software)를 분석하여 모듈화한 단계이며, 두 번째 단계는 추가적으로 C-레벨 최적화 및 알고리즘 최적화를 적용한 단계를 나타낸다. 마지막으로 C-레벨 및 알고리즘 최적화 단계에 제안한 최적화 명령어를 적용한 단계이다.

표 5~7는 총 8개의 CIF 실험 영상을 각각 MPEG-2, MPEG-4와 AVS 에서 IPP의 3프레임을 복호화한 실제 동작 사이클 수를 나타낸다. 최적화 비율은 참조 소프트웨어의 모듈화 단계와 C-레벨 최적화 및 명령어 추가 단계간의 사이클 감소비율을 나타낸다. 표 5~7의 실험 결과를 살펴보면, MPEG-2의 경우 C-레벨 최적화만 적용하였을 때 8개 영상에 대하여 평균 8%의 사이클 감소를 나타내었고, C-레벨 최적화와 제안한 명령어-셋을 추가하였을 경우 51%로 사이클 감소를 나타내어 3개의 표준 중 가장 높은 성능 향상을 나타내었다. MPEG-4는 C-레벨 최적화를 적용하였을 때 21%의 사이클 감소를 나타내어 MPEG-2나 AVS에 비하여 상대적으로 높은 C-레벨 최적화가 이루어 졌고, C-레벨 최적화에 명령어-셋을 추가하였을 경우 33%의 사이클 감소를 나타내었다. AVS의 경우 C-레벨 최적화에서 9%의 사이클 감소하였고, 제안한 명령어-셋을 추가하였을 경우 31%의 사이클 감소를 나타내었다.

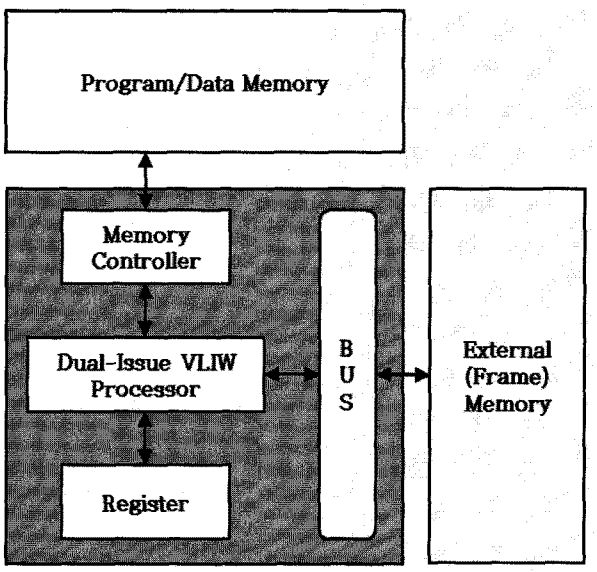


그림 8. 제안한 ASIP 기반 플랫폼의 블록 다이어그램
Fig. 8. The block diagram of the proposed ASIP-based platform.

표 5. 최적화 수준에 따른 MPEG-2의 동작 사이클
Table 5. Running cycles of MPEG-2 for optimization level.

최적화 단계 / 영상	모듈화 적용 (cycle)	C-레벨 최적화 (cycle)	명령어 추가 (cycle)	최적화 비율
Miss America	43,413,300	38,926,395	18,515,880	57.3%
Deadline	44,855,097	40,493,187	19,812,029	55.8%
Foreman	46,218,855	42,144,190	21,007,753	54.5%
Paris	47,041,886	42,693,537	22,032,288	53.2%
Coastguard	47,592,406	47,592,406	25,973,283	49.2%
Tempete	51,562,666	47,398,366	26,354,105	48.9%
Funfair	53,660,958	49,827,725	28,561,705	46.8%
Mobile	55,742,457	51,898,800	30,464,221	45.3%

표 6. 최적화 수준에 따른 MPEG-4의 동작 사이클
Table 6. Running cycles of MPEG-4 for optimization level.

최적화 단계 / 영상	모듈화 적용 (cycle)	C-레벨 최적화 (cycle)	명령어 추가 (cycle)	최적화 비율
Miss America	41,239,270	30,432,890	26,487,129	35.8%
Deadline	45,585,194	35,799,000	29,931,979	34.3%
Foreman	46,428,530	36,119,693	30,339,420	34.7%
Paris	48,699,767	38,071,437	32,018,297	34.3%
Coastguard	52,723,275	42,980,837	36,274,322	31.2%
Tempete	56,265,949	44,162,967	37,383,031	33.6%
Funfair	57,585,824	47,392,496	38,576,088	33.0%
Mobile	62,820,799	51,132,115	43,983,389	30.0%

표 7. 최적화 수준에 따른 AVS의 동작 사이클
Table 7. Running cycles of AVS for optimization level.

최적화 단계 / 영상	모듈화 적용 (cycle)	C-레벨 최적화 (cycle)	명령어 추가 (cycle)	최적화 비율
Miss America	63,259,833	57,392,730	40,458,561	36.0%
Deadline	72,792,618	66,115,072	48,881,041	32.8%
Foreman	85,831,183	75,671,860	55,590,007	35.2%
Paris	78,918,992	74,363,812	55,802,187	29.3%
Coastguard	91,448,863	86,925,569	66,859,570	29.1%
Tempete	101,885,098	88,792,341	68,992,151	32.3%
Funfair	101,523,944	93,531,092	72,428,443	28.7%
Mobile	111,585,644	99,356,913	78,468,365	29.7%

표 8~10은 각 표준의 기능 모듈별 복잡도에 대한 결과로써 참조 소프트웨어의 모듈화 단계와 C-레벨 최적화 및 명령어 추가 단계에서 각 모듈의 사이클과 모듈별 최적화 비율 나타낸다.

표 8. 명령어 추가에 따른 MPEG-2의 모듈별 소요 사이클
Table 8. Number of running cycle for each module of MPEG-2 with the proposed added instructions.

최적화 단계 / 모듈	모듈화 적용 (cycle)	명령어 추가 (cycle)	최적화 비율
VLD_IQ	149,697,714	87,571,449	41.5%
IDCT	119,282,898	72,241,272	39.4%
MEM_MC	9,035,766	9,035,766	0.00%
MC	18,116,013	7,167,385	60.4%
RECON	89,344,971	8,569,968	90.4%
MEM_RECON	8,192,448	8,135,424	0.70%
TOTAL	393,669,810	192,721,264	51.0%

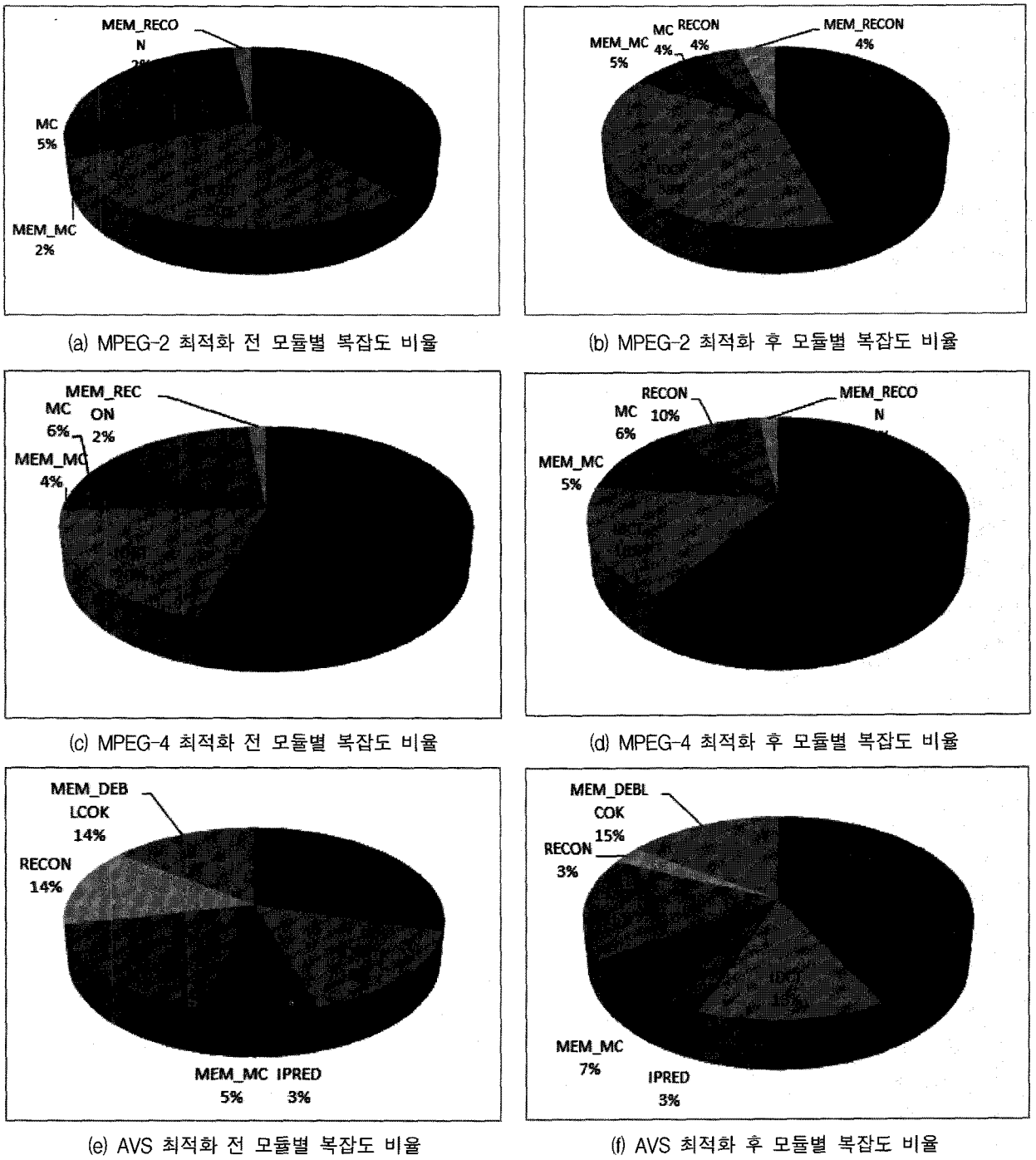
표 9. 명령어 추가에 따른 MPEG-4의 모듈별 소요 사이클
Table 9. Number of running cycle for each module of MPEG-4 with the proposed added instructions.

최적화 단계 / 모듈	모듈화 적용 (cycle)	명령어 추가 (cycle)	최적화 비율
VLD_IQ	224,788,151	162,846,887	27.6%
IDCT	82,899,494	49,595,048	40.2%
MEM_MC	15,204,686	14,410,493	5.2%
MC	23,269,829	16,472,644	29.2%
RECON	58,239,024	26,564,935	54.4%
MEM_RECON	6,947,424	5,103,648	26.5%
TOTAL	411,348,608	274,993,655	33.1%

표 10. 명령어 추가에 따른 AVS의 모듈별 소요 사이클
Table 10. Number of running cycle for each module of AVS with the proposed added instructions.

최적화 단계 / 모듈	모듈화 적용 (cycle)	명령어 추가 (cycle)	최적화 비율
VLD_IQ	208,157,690	206,223,211	0.90%
IDCT	112,953,397	63,472,582	43.8%
IPRED	23,678,549	15,508,840	34.5%
MEM_MC	36,638,669	34,480,898	5.90%
MC	124,455,988	78,179,385	37.2%
RECON	99,451,331	13,410,144	86.5%
MEM_DEBLOCK	101,880,541	74,205,265	27.2%
TOTAL	707,216,175	485,480,325	31.4%

표 8~10을 살펴보면, 3가지 표준에서 공통적으로 높은 최적화 성능을 나타내는 모듈은 RECON 모듈로써, MPEG-2와 AVS에서는 90%와 86.5%의 큰 사이클 감



(a) MPEG-2 최적화 전 모듈별 복잡도 비율

(b) MPEG-2 최적화 후 모듈별 복잡도 비율

(c) MPEG-4 최적화 전 모듈별 복잡도 비율

(d) MPEG-4 최적화 후 모듈별 복잡도 비율

(e) AVS 최적화 전 모듈별 복잡도 비율

(f) AVS 최적화 후 모듈별 복잡도 비율

그림 9. MPEG-2, MPEG-4, AVS의 최적화 전/후 모듈별 복잡도 비율
Fig. 9. Complexity ratio of MPEG-2, MPEG-4, and AVS before/after optimization.

소를 나타내었다. 이는 제안한 add_clip_4 명령어로 SIMD 형태로 분기를 감소시키고 메모리 접근 횟수를 줄일 수 있었기 때문으로, 이러한 SIMD 형태의 smul_4 명령어를 적용하여 보간 필터링을 수행하는 MC 모듈에서도 각각 60.4% 29.2%, 37.2%의 사이클 감소 결과를 확인할 수 있었다.

그러나 비디오 복화기의 복잡도 측면에서 살펴보면, VLD_IQ 모듈의 경우 비트스트림을 파일에서부터 읽어오는 파일 읽기 오버헤드 (File read overhead)와 잦은 분기로 인한 파이프라인 스톨로 인하여 가장 높은 복잡도를 나타내었다. 또한, 최적화 과정에서도 분기의 제거에 초점을 맞추었으나 다른 모듈에 비하여 최적화 비율

이 낮음을 확인 할 수 있다. 그림 9는 MPEG-2, MPEG-4 및 AVS의 최적화 이전과 이후의 비디오 복호화기에서 차지하는 모듈별 비율을 나타낸다. 그림 9에서 나타나는 바와 같이 최적화 이후에도 VLD_IQ 모듈의 복잡도는 MPEG-2에서는 45%, MPEG-4에서는 59%, AVS에서는 43%로 비디오 복호화기 복잡도의 절반가량을 차지하는 것으로 나타났다. 이는 비디오 복호화기의 구현에 있어 VLD_IQ 모듈은 프로세서 기반으로의 구현에 적합하지 않으며, 따라서 해당 모듈에 대해서는 하드웨어 가속 모듈로 처리할 필요가 있다.

전술한 바와 같이 복잡도가 높은 VLD_IQ 모듈에 대하여 추가적인 하드웨어 가속기를 추가할 경우, VLD_IQ를 제외한 최적화 비율을 살펴보면 MPEG-2의 경우 56.9%, MPEG-4에서는 39.9%, AVS에서는 44.0%의 보다 높은 사이클 감소 비율을 나타내었다.

V. 결 론

본 논문에서는 MPEG-2, MPEG-4, AVS 비디오 복호화기를 기능별 유사성에 기반한 모듈화 설계를 하였고, 모듈화 설계를 통해 하드웨어/소프트웨어의 개별적인 개발이나 통합 개발에 용이하도록 하였다. 또한, 모듈화 설계를 통하여 DMA 같은 하드웨어 모듈을 추가 구현하기 쉽도록 하였다. 또한, 2개의 슬롯을 가진 VLIW-2i 구조를 기본으로 한 프로세서를 ASIP를 이용하여 설계하고, 비디오 복호화기를 위한 공통의 전용 명령어를 추가함으로써 다중 비디오 복호화가 가능한 프로세서를 제안하였다. 추가적으로 소프트웨어 레벨에서 C 최적화 및 알고리즘 최적화를 수행하여 초기 프로세서에서의 동작 성능에 비하여 평균 37% (VLD_IQ 제외 46.6%)의 사이클 감소를 이루었다. 현재, 고속 비디오 복호화를 위하여 제안하는 프로세서를 이용, 멀티-코어 기반의 병렬화에 대한 연구를 진행 중에 있으며, 향후 고속 다중 비디오 복호화가 이루어질 것으로 기대된다.

참 고 문 헌

[1] Jung-Hak, Nam, Dong-Gyu Sim, "Lossless video coding based on pixel-wise prediction," *Multimedia Systems*, vol.14, no. 5, pp. 291-298, Nov. 2008.

[2] 이재진, 박성모, 엄낙웅, "멀티미디어 애플리케이션 처리를 위한 ASIP," *전자통신동향분석*, 제 24 권, 제 6 호, pp. 94-98, 2009.

[3] Synopsys Inc., System-level design tools, www.synopsys.com

[4] A. Hoffmann, O. Schliebusch, A. Hohl, G. Braun, and H. Meyr, "A Methodology for the Design of Application Specific Instruction Set Processors (ASIP) Using the Machine Description Language LISA," *In Proceedings of the ICCAD*, San Jose, USA, Nov. 2001.

[5] "ISO/IEC 14496-2: visual," Dec, 2001

[6] Wen Gao, "AVS-The Chinese Next-Generation Video Coding Standard," *International Journal of Control*, vol. 23, no. 4, pp. 123-145, May, 1989.

[7] Seogmo Park, et al "A Low Power Design of H.264 Codec Based on Hardware and Software Co-design," *한국통신학회지*, 제 25 권, 제 12호, pp. 10-18, 2008.

[8] 강대범, 심동규, "멀티미디어 DSP를 위한 AVS 비디오 복호화기 구현," *전자공학회논문지*, 제 46권 SP편, 제 5 호, 151-161쪽, 2009년.

[9] SungDae Kim, "ASIP Approach for Implementation of H.264/AVC," *Journal of Signal Processing Systems*, vol. 50, pp. 53-67, January, 2008.

[10] M. Hohenauer, et al. "Retargetable Code Optimization with SIMD Instructions," *International Conference on Hardware Software Codesign*, pp. 148 - 153, 2006.

저 자 소 개

안 용 조(학생회원)
2010년 광운대학교 컴퓨터공학과 학사
2010년~현재 광운대학교 컴퓨터공학과 석사과정
<주관심분야 : 영상압축, 멀티프로세서>

조 현 호(학생회원)
2008년 광운대학교 컴퓨터공학과 학사
2010년 광운대학교 컴퓨터공학과 석사
2010년~현재 광운대학교 컴퓨터공학 박사과정
<주관심분야 : 영상처리, 영상 압축>

심 동 규(정회원)-교신저자
1999년 서강대학교 전자공학과 공학박사
1999년~2000년 (주) 현대전자
2000년~2002년 (주) 바로비젼
2002년~2005년 Univ. of Washington
2005년~현재 광운대학교 컴퓨터공학과 (부교수)
<주관심분야 : 영상 신호처리, 영상 압축, 컴퓨터 비전>

강 대 범(정회원)
2008년 광운대학교 컴퓨터공학과 학사
2010년 광운대학교 컴퓨터공학과 석사
2010년~현재 삼성전자 무선사업부 선행개발팀 연구원
<주관심분야 : 영상압축, 멀티프로세서>

지 봉 일(학생회원)
2009년 광운대학교 컴퓨터공학과 학사
2009년~현재 광운대학교 컴퓨터공학과 석사과정
<주관심분야 : 영상압축, 멀티프로세서>

엄 낙 웅(정회원)
1984년 경북대학교 전자공학과 학사
1987년 한국과학기술원 전기 및 전자공학과 석사
2001년 한국과학기술원 전기 및 전자공학과 박사
2006년 한국전자통신연구원 고속통신IC설계팀장
2008년 한국전자통신연구원 융합부품소재연구부문SoC연구부장
2009년 한국전자통신연구원 시스템반도체연구부장
2010년~현재 한국전자통신연구원 멀티미디어프로세서연구팀장
<주관심분야 : 멀티미디어 프로세싱, FPGA 설계>