임베디드 병렬 프로세서를 위한 칼라미디어 명령어 구현

(Color Media Instructions for Embedded Parallel Processors)

김 철 홍 † 김 종 면 ††
(Cheol-Hong Kim) (Jong-Myon Kim)

요 약 최근 모바일 컴퓨팅 환경의 변화로 멀티미디어 데이타의 고성능, 저전력 처리에 대한 수요가 증가하고, 프로세서에 있어서 멀티미디어 전용 가속기 기능의 중요성이 크게 부각되고 있다. 이에 본 논문은 고성능, 저전력 멀티미디어 처리를 위한 SIMD 병렬 프로세서용 칼라미디어 명령어를 제안한다. 기존의 범용 마이크로프로세서 전용 멀티미디어 명령어 (e.g., MMX, VIS, AltiVec)는 4개의 8 비트 픽셀을 32 비트 레지스터에 저장하고 처리하는 반면에, 제안하는 칼라미디어 명령어는 인간의 시각이 칼라에 덜 민감한 점을 고려하여 32비트 데이타패스 아키텍처에서 두 쌍 (6개의 픽셀)의 압축된 16비트 YCbCr (6비트 Y, 5비트 Cb와 Cr) 데이타를 32비트 레지스터에 저장하고 동시에 처리함으로써 YCbCr 데이타 처리에서 높은 병렬성과 효율성을 보여준다. 또한 칼라미디어 명령어는 데이타 포맷 사이즈를 줄임으로써 전체 시스템의 비용을 절감할 뿐만 아니라 데이타 대역폭의 감소로 시스템 디자인을 간소화한다. SIMD병렬 프로세서 아키텍처에서 모의 실험한 결과, 칼라미디어 명령어 기반 프로그램은 baseline 명령어 프로그램보다 평균 6.3배 성능향상을 보여준다. 반면, Intel의 대표적인 멀티미디어 명령어인 MMX 기반 프로그램은 동일한 SIMD 병렬 프로세서에서 baseline 명령어 프로그램보다 단지 3.7배 성능향상을 나타낸다. 또한, 칼라미디어 명령어는 MMX보다 시스템 면적 효율 (52% 증가 대비 13% 증가)과 시스템 전력 효율 (50% 증가 대비 11% 증가)에서 우수성을 보여준다. 칼라미디어 명령어는 이러한 성능과 효율을 단지 3%의 시스템 면적과 5%의 시스템 전력의 증가로 얻는 반면, MMX는 14%의 시스템 면적과 16%의 시스템 전력 증가가 요구된다.

키워드 : 칼라 이미지/비디오 프로세싱, 멀티미디어 명령어, 임베디드 SIMD 병렬 프로세서

Abstract As a mobile computing environment is rapidly changing, increasing user demand for multimedia-over-wireless capabilities on embedded processors places constraints on performance, power, and size. In this regard, this paper proposes color media instructions (CMI) for single instruction, multiple data (SIMD) parallel processors to meet the computational requirements and cost goals. While existing multimedia extensions store and process 4 8-bit pixels in a 32-bit register, CMI, which considers that color components are perceptually less significant, supports parallel operations on two-packed compressed 16-bit YCbCr (6 bit Y and 5 bits Cb, Cr) data in a 32-bit datapath processor. This provides greater concurrency and efficiency for YCbCr data processing. Moreover, the ability to reduce data format size reduces system cost. The reduction in data bandwidth also simplifies system design. Experimental results on a representative SIMD parallel processor architecture show that CMI achieves an average speedup of 6.3x over the baseline SIMD parallel processor performance. This is in contrast to MMX (a representative Intel's multimedia extensions), which achieves an average

· 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 육성지원사업의 연구결과로 수행되었습니다(IITA-2008-(C1090-0801-0039)).

† 정 회 원 : 전남대학교 전자컴퓨터공학과 교수
cheolhong@gmail.com
†† 정 회 원 : 울산대학교 컴퓨터정보통신공학부 교수
jongmyon.kim@gmail.com
(Corresponding author임)

논문접수 : 2007년 10월 12일
심사완료 : 2008년 4월 21일

정보과학회논문지: 시스템 및 이론 제35권 제7호(2008.8)

speedup of only 3.7x over the same baseline SIMD architecture. CMI also outperforms MMX in both area efficiency (a 52% increase versus a 13% increase) and energy efficiency (a 50% increase versus an 11% increase). CMI improves the performance and efficiency with a mere 3% increase in the system area and a 5% increase in the system power, while MMX requires a 14% increase in the system area and a 16% increase in the system power.

Key words : Color image and video processing, multimedia instructions, embedded SIMD parallel processors

1. 서 론

근래 모바일 멀티미디어의 방대한 데이타를 얼마나 효율적으로 처리하는가 하는 문제가 크게 대두되고 있다. 특히 칼라 특성은 주위환경 감지, 물체 식별 등과 같은 중요한 정보를 전달하는 수단이므로 칼라 이미지/비디오 데이타 처리에 대한 연구가 활발하게 진행되고 있다[1]. 그러나, 이러한 칼라 이미지/비디오 애플리케이션은 상당한 양의 연산과 입출력 처리를 요구한다. 더욱이 무선 네트워크에서 멀티미디어 통신이 증가함에 따라 고성능, 저전력에 대한 수요가 증가하고 있다.

ASIC(Application-Specific Integrated Circuit)은 이러한 모바일 멀티미디어 애플리케이션에서 요구되는 고성능, 저전력을 충족시킬 수 있지만 다양한 애플리케이션에서 요구되는 programmability와 유연성(flexibility)을 만족시키지 못한다.

반면에 범용 마이크로프로세서(general-purpose microprocessors, GPPs)들은 다양한 애플리케이션에 대해 충분한 programmability와 유연성을 제공할 뿐만 아니라 멀티미디어 확장 명령어를 내장하여 멀티미디어 애플리케이션의 성능을 향상시킨다. 범용 마이크로 프로세서에 사용되는 멀티미디어 확장 명령어의 예로는 Intel MMX [2]와 SSE[3], Hewlett Packard MAX-2[4], Sun VIS [5], Alpha MVI[6], Motorola ALTIVEC[7] 등이 있다. 이러한 멀티미디어 명령어들은 SIMD(Single Instruction Multiple Data) 가속기능을 이용하여 하나의 넓은 레지스터(e.g., 32, 64, 128비트 폭)에 여러 개의 작은 데이타(e.g., 8비트 픽셀)를 패킹하여 동시에 처리함으로써 성능을 향상시킨다. 이러한 기술은 DSP(Digital Signal Processor) 디자인에도 채택되고 있다. 예로는 Texas Instruments TMS320C64x families[8], Analog Devices TigerSharc processors[9], ARM processors [10] 등이 있다.

그러나, 멀티미디어 확장 명령어를 통한 성능 향상에도 불구하고 GPP나 DSP는 차세대 멀티미디어 애플리케이션에서 요구되는 높은 레벨의 성능을 만족시키지 못할 것이다. 왜냐하면 GPP나 DSP는 프로세서 구조의 특성상 멀티미디어 애플리케이션에 내재한 모든 데이타 병렬성 (data parallelism)을 활용하지 못하기 때문이다.

이러한 멀티미디어 애플리케이션의 고성능 요구를 충족시킬 수 있는 프로세서 모델 중에서 SIMD 병렬 프로세서 아키텍처가 유망한 대안으로 부각되고 있다[11,12]. Instruction-level이나 thread-level 프로세서들은 실리콘 면적을 멀티포트 레지스터 파일(multiported register file), 캐쉬(cache), deep pipelined 기능 유닛 등으로 사용하는 반면, SIMD 병렬 프로세서는 수천 개의 저비용 프로세싱 엘리먼트(processing element, PE)들을 이용하여 고성능을 추구하고 동시에 저장장소와 데이타 통신 요구를 최소화하기 위해 PE 와 데이타 입출력을 동일위치에 배치함으로써 저전력을 만족시킨다. 특히, SIMD 병렬 프로세서는 지역성(locality)이나 규칙성(regularity)이 있는 2 차원 패턴의 이미지나 비디오 픽셀 처리에 있어서 최적의 프로세서 구조이다. 하지만 칼라 이미지/비디오의 멀티채널 픽셀(e.g., RGB, YCbCr, YUV, YIQ) 연산은 프로그램의 가장 안쪽 루프에서 수행되기 때문에 이러한 멀티채널 픽셀 처리 성능은 SIMD 병렬 프로세서의 PE개수 증가와 무관하다.

본 논문은 이러한 문제를 해결하기 위해 SIMD 병렬 프로세서 아키텍처를 위한 칼라미디어 명령어(Color Media Instruction, CMI)를 제안한다.(본 논문은 [13]의 확장 논문으로 CMI의 성능평가와 더불어 면적 및 에너지 효율 측면에서 CMI의 효율성을 기술한다.) CMI는 32비트 데이타패스 프로세서에서 두 쌍의 16비트 YCbCr (6비트 Y, 5비트 Cr and Cb) 데이타를 하나의 32비트 레지스터에 저장하여 동시에 처리함으로써 성능 향상을 추구할 뿐만 아니라 칼라 이미지/비디오 데이타 처리에 있어서 높은 유연성을 보여준다. YCbCr 칼라 스페이스는 휴먼 비전(human vision)의 특성을 이용하여 사람의 시각이 덜 민감한 높은 주파수의 chrominance 컴포넌트(Cb, Cr)들에 있어서 subsampling을 가능하게 하거나, 덜 중요한 데이타에 truncation을 가능하게 함으로써 데이타의 저장장소 사이즈를 줄일 수 있다. 그러므로 6비트 luminance(Y)와 5비트 chrominance(Cr, Cb)로 구성되는 16비트 칼라 표현은 이미지 quality 저하 문제를 초래하지 않는다[14]. 이러한 16비트 YCbCr 데이타를 처리하는 CMI는 데이타 포맷 사이즈를 줄임으로써 상당한 시스템 비용을 줄일 뿐만 아니라 여러 개의 픽

셀을 동시에 처리함으로써 성능도 향상시킬 수 있다. 또한, 데이타 대역폭을 감소시킴으로써 시스템 디자인을 간소화할 수 있다.

본 논문은 SIMD 병렬 프로세서 아키텍처에 CMI를 구현하여 칼라 이미지/비디오 애플리케이션의 성능, 에너지 효율(energy efficiency) 및 면적 효율(area efficiency)에서 어떤 영향을 미치는지 평가한다. 또한, Intel의 대표적인 멀티미디어 확장 명령어인 MMX를 동일한 SIMD 병렬 프로세서에 구현하여 성능을 비교 평가를 한다. 모의 실험 결과, CMI 기반 프로그램은 basline 프로그램보다 5.2~8.8배(평균 6.3배) 성능향상을 보인 반면 MMX기반 프로그램은 동일한 SIMD 병렬 프로세서에서 baseline 프로그램보다 단지 3~5배(평균 3.7배) 성능향상을 보여준다. 또한 CMI 프로그램은 baseline 프로그램보다 52% 에너지 효율, 50%의 시스템 면적 효율을 증가시킨 반면, MMX 프로그램은 baseline 프로그램보다 단지 13% 에너지 효율, 11% 시스템 면적 효율 증가를 나타낸다. CMI는 이러한 성능과 효율을 단지 3%의 실리콘 면적, 5%의 시스템 전력 증가로 얻는 반면, MMX는 14%의 실리콘 면적, 16%의 시스템 전력 증가가 요구된다. 이러한 결과들로부터 CMI를 임베디드 멀티미디어 시스템에 적용할 경우 상당한 성능 향상 및 에너지 효율 증가가 기대된다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 범용 마이크로프로세서 전용 멀티미디어 명령어 및 관련 연구를 소개한다. 3장에서는 본 논문에서 제안하고자 하는 SIMD 병렬 프로세서용 칼라미디어 명령어를 요약하고, 4장에서는 채택된 칼라 이미지/비디오 애플리케이션, SIMD 병렬 프로세서 아키텍처 모델링, 그리고 칼라미디어 명령어의 성능 평가를 위한 methodology infrastructure를 소개한다. 5장에서는 각 경우에 대한 성능 및 효율 결과를 분석하고, 끝으로 6장에서는 이 논문의 결론을 맺는다.

2. 관련 연구

2.1 범용 마이크로프로세서 전용 멀티미디어 명령어

범용 마이크로프로세서 제조회사는 멀티미디어 애플리케이션의 성능을 향상시키기 위해 멀티미디어 명령어를 그들의 instruction set architecture(ISA)에 첨가하였다. 표 1은 현재 모든 마이크로프로세서 제조회사에서 발표한 멀티미디어 명령어의 리스트를 보여준다. 이러한 멀티미디어 명령어의 주요 장점은 하나의 넓은 레지스터(64 비트 혹은 128비트 예)에 여러 개의 작은 데이타를 저장하고 동시에 처리함으로써 성능을 향상시킨다(그림 1 참조).

제조회사의 타깃 애플리케이션에 따라 이러한 멀티미

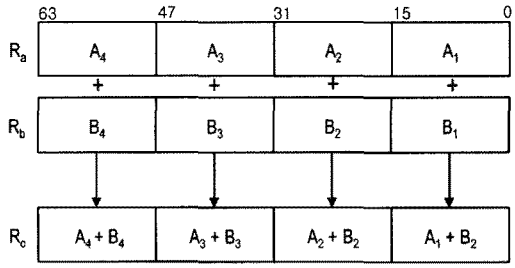

그림 1 병렬 덧셈 명령어

표 1 범용 마이크로프로세서용 멀티미디어 명령어 리스트

Processor	Extension	Product	Instructions	Register File
HP	MAX-1	1994	9	Integer (31x64b)
Sun	VIS	1995	121	FP (32x64b)
HP	MAX-2	1995	8	Integer (32x64b)
MIPS	MIPS-V	(-)	29	FP (32x64b)
MIPS	MDMX	(-)	74	FP (32x64b), Acc. (1x92b)
Intel	MMX	1997	57	FP (8x64b)
DEC	MVI	1997	13	Integer (31x64b)
Cyrix	Extended MMX	1997	12	FP (8x64b)
AMD	3D Now!	1998	21	FP (8x64b)
Intel	SSE	1999	70	8x128b
Motorola	AltiVec	1999	162	32x128b
MIPS	MIPS-3D	(-)	23	FP (32x64b)
AMD	Enhanced 3D Now!	1999	24	FP (8x64b)
Intel	SSE2	(-)	144	8x128b

디어 명령어는 다양하다. Motorola AltiVec은 가장 많은 수의 SIMD 명령어(162개)를 가지고 있는 반면, HP MAX-1은 단지 8개의 SIMD 명령어를 가지고 있다. 대부분의 멀티미디어 명령어(AMD 3DNow!, DEC MVI, Intel MMX, Sun VIS)는 64 비트 레지스터를 사용하는 반면, Motorola AltiVec과 Intel SSE는 128 비트 레지스터를 이용한다. 한가지 주목할 만한 예외는 MIPS MDMX인데 이 명령어는 여러 번의 연산에 의한 결과를 축적하기 위해 하나의 넓은 누산기를 가지고 있다. 이러한 명령어의 유사성에도 불구하고 각각의 명령어는 독특한 특징을 가지고 있다. 예를 들어, MAX-2는 실행 유닛과 정수 레지스터를 재사용하여 추가적인 하드웨어를 필요로 하지 않는 반면, AltiVec은 전적으로 새로운 실행 유닛을 요구한다.

2.2 관련 연구

멀티미디어 애플리케이션에 대한 데이타 레벨 병렬성(data-level parallelism, DLP)에 관한 연구는 크게 두 개의 연구 그룹으로 나누어진다. (1) 현재의 SIMD 명령어를 이용하여 성능을 향상시키는 그룹[15-17]과 (2) 병렬 프로세서를 이용하여 성능을 향상시키는 그룹 [12, 18]. 많은 연구 그룹 혹은 개인들이 범용 마이크로프로세서에서 멀티미디어 애플리케이션에 대한 SIMD 명령어의 효율성에 대하여 분석하였다. [15]에서는 Ultra-SPARC 프로세서에서 이미지와 비디오 처리에 대한 VIS 명령어의 효율성을 기술하였다. 4-way out-of-order 프로세서는 single in-order 프로세서보다 2.3배~4.2배의 성능을 향상시켰고 더불어 VIS 명령어는 1.1배~4.2배의 성능을 더 향상시켰다. [16]에서는 DSP와 멀티미디어 애플리케이션에 대한 MMX 명령어의 성능 평가를 기술하였다. MMX 명령어는 81%의 다이내믹 명령어를 감소시켜 평균 5.5배의 성능 향상을 보였다. 이러한 결과에서 보는 바와 같이 SIMD 명령어는 적당한 수준의 성능을 향상시킨다. 하지만 멀티미디어 애플리케이션에 내재한 완전한 데이타 병렬성을 얻지 못하기 때문에 다양한 형태의 멀티미디어에서 요구되는 상당한 양의 성능 요구를 만족시키지 못할 것이다.

SIMD 병렬 프로세서는 공간적 병렬성(spatial parallelism)을 실현하기 위해 여러 개의 동기화된 프로세싱 유닛(processing unit)들을 사용한다. 이 유닛들은 하나의 제어 유닛으로부터 동시에 전송되는 동일한 연산 명령을 서로 다른 데이타에 대하여 수행한다. 따라서 데이타 병렬 모델을 이용하여 성능을 향상시킨다. Massively data parallel array들은 거의 30년 동안 이미지 처리에 사용되어 왔지만, 초기의 SIMD 병렬 프로세서(TMC Connection Machine 1[19])는 I/O 테크놀로지에 의해 제한되었다. 이후의 SIMD 병렬 프로세서인 TMC CM-2[20]와 MasPar MP-2[21]는 버퍼 이미지의 큰 병렬 디스크 어레이의 사용을 통해 이러한 제한을 극복하였지만 비용과 portability를 희생하였다. Fine-grained 병렬 프로세서인 MGAP[22]와 ABACUS [23]는 이러한 portability이슈를 해결하였지만, 그들의 성능은 I/O bandwidth와 latency에 의해 제한되었다.

이러한 SIMD 병렬 프로세서와 다르게, 본 논문에서 모의실험을 위해 사용한 SIMD Pixel Processor(SIMPil) [12]는 프로세서와 센서의 직접적 연결을 통해 I/O 대역의 문제를 해결하고, 또한 짧은 와이어의 사용으로 높은 면적과 에너지 효율을 보인다. 이러한 2-D SIMD 병렬 프로세서는 많은 데이타에 동일한 명령어를 동시에 수행하여 성능을 향상시키는 반면에, 가장 안쪽 루프에서 수행되는 멀티채널 처리에는 비효율적이다. 본 논문은 멀티채널 칼라 이미지 및 비디오 처리를 위해 휴먼 지각용 칼라 인식 명령어를 SIMPil 프로세서에 구현하여 성능 향상을 추구한다.

3. 칼라 이미지/비디오 애플리케이션을 위한 칼라미디어 명령어

멀티채널(multichannel) 픽셀 코딩에서 표준 이미지의 각 픽셀은 세가지 컴포넌트(빨강: Red, 초록: Green, 파랑: Blue)의 신호로 구성되어 있다. 그러나, RGB 칼라 스페이스는 인간이 칼라를 인식하는 장치로 적합하지 않기 때문에 이미지 처리 기술을 적용할 때 칼라 변형이 자주 발생한다[24]. 또한 각 R, G, B 컴포넌트 사이에는 밀접한 상관관계가 있어서 독립적인 코딩이 불가능하다. 이러한 문제의 해결책으로서 휴먼 인식 기반의 YCbCr 칼라 스페이스가 이미지/비디오 처리 분야에서 널리 사용되고 있다. YCbCr 칼라 스페이스의 장점은 인간의 눈이 chrominance의 높은 주파수에 덜 민감하기 때문에 chrominance 컴포넌트인 Cr과 Cb는 칼라 변형 없이 subsampling 이 가능하다. 또한luminance (Y) 컴포넌트는 chrominance 컴포넌트와 독립적으로 처리 가능하다. 이러한 YCbCr 칼라 스페이스의 특징을 이용한 분리 채널 및 luminance-only 처리 기술이 칼라 이미지/비디오 애플리케이션에서 적용되기도 한다 [25,26]. 하지만 이러한 이미지/비디오 처리 기법들은 칼라 채널들 사이의 연관성을 고려하지 않기 때문에 칼라에 의해 전달되는 중요한 정보를 얻을 수 없으므로 이미지 처리의 정확도 및 전체적인 이미지의 질을 저하시킨다. 그러므로 멀티채널 처리 방법은 칼라 이미지 처리의 정확도 증가시킬 뿐만 아니라 이미지의 질을 개선시키는 장점이 있다. 이러한 이점에도 불구하고 멀티채널 처리 기법이 이미지/비디오 애플리케이션에 널리 사용되지 않는 이유는 칼라 채널 사이의 복잡한 처리에서 발

생하는 상당한 연산 오버헤드 때문이다.

이러한 문제를 해결하기 위해 본 논문은 칼라미디어 명령어(Color Media Instructions, CMI)를 제안한다. CMI는 32비트 데이타패스 프로세서에서 두 쌍의 16비트 YCbCr(6-5-5 bit) 데이타를 32비트 레지스터에 저장하고 동시에 병렬 처리함으로써 성능을 향상시킬 뿐만 아니라 효율적인 칼라 처리를 제공한다. CMI는 일반적인 서브워드 병렬 기법(subword parallelism)을 이용하여 적당한 성능향상을 꾀하는 기존의 멀티미디어 확장 명령어들과 차별화된다. 그림 2는 세가지 다른 형태의 명령어를 보여주고 있다. (1) 기본적인 32비트 명령어, (2) 4 × 8 비트 SIMD 명령어, (3) 2 × 16 비트 CMI.

칼라 이미지에서 밴드 데이타는 인터리브 (interleave: 한 픽셀에 대하여 여러 밴드의 데이타를 순차적으로 기록하고 다음 픽셀에 대하여도 같은 방법을 사용하여 기록)되거나 분리(separate: 전체 영역에 대하여 한 밴드의 데이타를 저장)되어 저장된다. 비록 밴드 분리 포맷은 SIMD 연산에 가장 적합한 구조이지만 SIMD 연산을 위한 데이타 정렬과 같은 전처리(preprocessing)에 상당한 오버헤드가 수반된다. 이러한 문제를 해결하기 위해 SIMD 명령어가 밴드 인터리브 포맷으로 픽셀 정보를 레지스터에 저장하고 병렬 처리할 수 있지만, unused field에서 연산 낭비가 발생한다(see 그림 2(b)). 또한 RGB 칼라 스페이스는 휴먼 비전의 지각 속성을 위한 모델이 아니므로 RGB로부터 YCbCr로 칼라 변환이 요구된다.

제안하는 CMI는 두 쌍의 16비트 YCbCr 데이타를 32비트 영역에 적당하게 정렬하고 병렬처리 함으로써 패킹된 RGB 확장 명령어들에 내재한 문제들을 효율적으로 해결할 수 있다. CMI는 다음과 같은 4개의 그룹[(1) 병렬 산술·논리(ALU) 명령어, (2) 병렬 비교(Compare) 명령어, (3) 치환(Permute) 명령어, (4) 특정목적(Special-purpose) 명령어]으로 구성된다.

3.1 병렬 산술·논리(ALU) 명령어

병렬 산술·논리 명령어에는 가산(ADD_CMI), 감산(SUBTRACT_CMI), 그리고 Average 명령어(AVERAGE_CMI)가 있다. 가산/감산 명령어에서 오버플로(overflow)가 발생할 때 주어진 데이타 형에서 가장 큰 값이나 작은 값을 반환하는 포화 연산자(saturation operator)를 포함하고 있다. 특히 포화 산술 연산자는 오버플로가 발생하면 검정색에서 흰색으로의 변환을 막는 픽셀 관련 연산에 유용하다. Average 명령어는 각 레지스터에 저장된 각 쌍의 서브워드(subword)를 동시에 평균 처리하기 때문에 blending과 같은 알고리즘에 유용하다.

3.2 병렬 비교(Compare) 명령어

병렬 비교 명령어에는 CMPEQ_CMI, CMPNE_CMI, CMPGE_CMI, CMPGT_CMI, CMPLE_CMI, CMOV_CMI(conditional move), MIN_CMI, MAX_CMI 등이 있다. 이러한 명령어는 두 개의 소스 레지스터(source register)에 저장되어 있는 서브워드의 쌍들을 동시에 비교하는 연산자이다. 처음 7개의 명령어는 입력 데이타

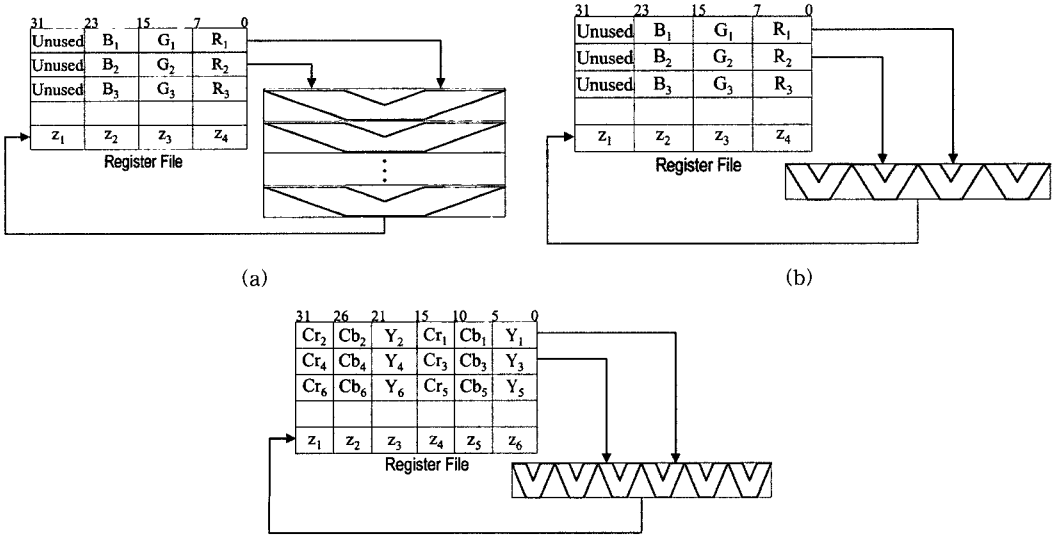

그림 2 (a) 기본적인 32비트 명령어, (b) 4 × 8 비트 SIMD 명령어, (c) 2 × 16 비트 칼라 미디어 명령어

에 수행되는 condition query를 추출할 때 유용하며, 마지막 두 명령어 (MIN_CMI, MAX_CMI)는 픽셀들의 최소 및 최대값을 추출하는 필터링 (filtering) 알고리즘에 유용하다.

3.3 치환(Permute) 명령어

치환 명령어에는 BCAST_CMI (broadcast), MIX_CMI, ROTATE_CMI 등이 있다. 이러한 명령어들은 패킹된 데이타 형에서 각 데이타 요소들의 순서를 재구성하기 위해 사용된다. 그림 3, 4는 MIX_CMI 명령어와 ROTATE_CMI 명령어를 보여주며, 이러한 명령어들은 픽셀 전환(pixel transposition)이나 매트릭스 변환(matrix transposition) 등에 유용하다[27].

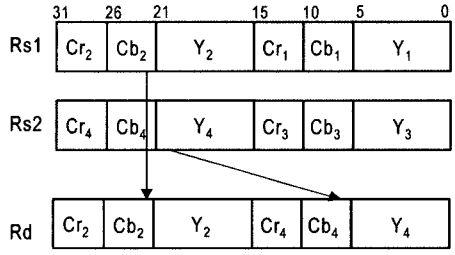

그림 3 MIX_CMI 명령어 예

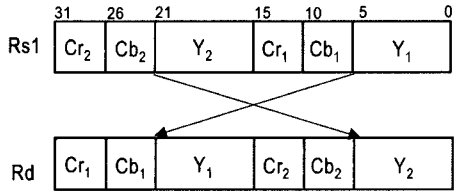

그림 4 ROTATE_CMI 명령어 예

3.4 특정목적(Special-Purpose) 명령어

특정목적 명령어에는 ADACC_CMI(absolute-differences accumulate), MACC_CMI(multiply-accumulate), RAC(read accumulate), ZACC(zero accumulate) 등이

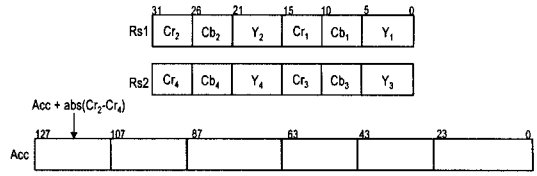

그림 5 ADACC_CMI 명령어 예

있으며, 성능 면에서 가장 뛰어난 명령어들이다. 예를 들어, ADACC_CMI는 다양한 움직임 추정(motion estimation)과 같은 알고리즘에 사용된다. 그림 5에서 보듯이 두 개의 소스 레지스터에 있는 각 쌍의 서브워드는 절대 차로 계산되고 그 결과 값은 패킹된 누산기(accumulator)에 저장된다. MACC_CMI는 디지털 필터링(digital filtering) 또는 컨볼루션(convolution)과 같은 vector dot-product 연산을 수행하는 DSP 알고리즘에 유용하다. 마지막 두 명령어(RAC, ZACC)는 누산기를 관리하는 명령어이다.

4. 평가 방법론

4.1 칼라 이미지/비디오 애플리케이션

다양한 형태의 멀티채널 처리 기법을 구현하기 위해 본 논문은 다섯 개의 칼라 이미지/비디오 애플리케이션을 채택한다. 채택된 애플리케이션의 종류에는 크로마키(chorma-key, CHROMA), Sobel 오퍼레이터(operator)를 이용한 칼라 에지 검출(Edge Detection, ED), 벡터 미디언 필터링(Vector Median Filtering, VMF), 벡터 양자화(Vector Quantization, VQ), MPEG의 핵심 커널(core kernel)인 움직임 추정(Motion Estimation, ME) 등이 있다. 표 2에 설명되어 있는 이러한 애플리케이션들은 인터넷의 스트리밍 비디오(streaming video), 실시간(real-time) 비디오 개선 및 분석, 씬 시각화(scene-visualization)등과 같은 실 세계(real-world)에서 중요한 컴포넌트로서 사용되고 있다. 모의 실험에서 애플리케이션들은 3밴드(band) QCIF(176×144) 해상도(reso-

표 2 채택된 칼라 이미지/비디오 애플리케이션 요약

애플리케이션	설 명
Chroma-Keying	하나의 영상 신호에 다른 하나의 영상을 겹치게 하는 기술이다.
Edge Detection	3가지 밴드(Y,Cb,Cr)에 Sobel 오퍼레이터를 동시 적용하여 얻어진 지역 변화(local changes)를 통하여 칼라 에지 정보를 검출한다.
Vector Median Filtering	칼라 이미지로부터 noise를 제거하는 방법으로서 3가지 밴드(Y,Cb,Cr)에 벡터 미디언을 동시에 적용하여 어떤 특정한 주변 영역내의(3×3 window) 화소 농도의 중간 값을 구하여 원하는 화소의 농도로 처리한다.
Vector Quantization	훈련 벡터 셋(training vector set)을 그룹으로 분할하고 그 그룹의 대표하는 값을 코드 벡터(code vector)로 정한 코드 벡터를 모아 코드북(codebook)을 작성하여 입력되는 벡터와 비교하여 최적의 코드 벡터를 인덱스(index)로 출력하는 압축 기법이다.
Motion Estimation	MPEG 비디오의 핵심기술로써 이전 프레임(frame)과 현재 프레임의 차를 이용하여 움직임을 추정하고 이를 보상해주는 압축 기법이다.

lution)의 입력 이미지로 구현된다.

4.2 SIMD 병렬 프로세서 모델

그림 6은 본 논문에서 사용된 SIMD 병렬 프로세서 아키텍처와 상호연결 네트워크를 보여준다. 각 PE (Processing Element)의 이미지 센서는 이미지 프레임으로부터 4×4 픽셀 데이타를 추출하여 특정 레지스터 파일에 저장한다. 데이타가 각 PE 들에 분배되고 나면 PE들은 촘촘한 배열 형태에서 명령어들을 수행한다. 각 PE는 다음과 같은 특징을 가진 RISC(Reduced Instruction Set Computer) 아키텍처이다.

- 32비트 폭의 128개 워드로 구성된 로컬 메모리
- 32비트 폭의 16개 3 포트 범용 레지스터
- 기본적인 산술/논리 연산을 수행하는 ALU
- 멀티 비트 산술/논리 시프트 연산을 수행하는 배럴 시프트(Barrel Shifter)
- 32비트 곱셈 및 누산기(multiply-accumulator, MACC)
- 지역 정보를 이용해 각 PE들을 활성 및 비활성 시키는 Sleep 유닛
- 지역 이미지 센서로부터 픽셀 데이타를 샘플링 하는 Pixel 유닛
- 칼라 변환 유닛(RGBtoYCC, YCCtoRGB)
- 이웃하는 PE들과 데이타 통신을 위한 NEWS (north-east-west-south) 네트워크 및 serial I/O 유닛

이러한 SIMD 병렬 프로세서는 2-D 이미지 처리에서 높은 성능을 보여주지만 YCbCr 데이타를 위한 멀티채널 처리용 애플리케이션에는 부적합하다. 왜냐하면 멀티채널 연산은 프로그램의 가장 안쪽에 있는 루프에서 수행되므로 그것의 성능은 PE 개수의 증가와 무관하다. 이러한 성능 제한의 문제점을 해결하기 위해 본 논문은 칼라미디어 명령어(CMI)를 SIMD 병렬 프로세서의 instruction set architecture(ISA)에 첨가한다. 또한 Intel의 대표적인 멀티미디어 전용 명령어인 MMX[2]와 유사한 명령어를 동일한 SIMD 병렬 프로세서에 첨가하여 성능을 비교 분석한다. 공정한 성능 평가를 위해 모의실험에서 칼라 변환의 오버헤드는 제외시켰다 (RGB to/from YCbCr). 예를 들어, baseline ISA, MMX, CMI 기반 프로그램들은 band-interleaved 포맷에서 YCbCr 데이타를 직접 처리하도록 구현되었다. 즉, baseline 구현에서는 32비트 |unused|Cr|Cb|Y| 칼라 데이타를 32비트의 레지스터에 저장하고 unpack하여 각각의 분리된 Y, Cb, Cr 컴포넌트에 대하여 수행하는 반면, MMX는 32 비트의 패킹된 |unused|Cr|Cb|Y|를 32비트의 레지스터에 저장하고 3개의 Y, Cb, Cr 데이타에 대해 동시에 수행한다. CMI는 MMX와 유사하게 32비트의 패킹된 2쌍의 |Cr|Cb|Y|Cr|Cb|Y| 데이타를 32비트의 레지스터에 저장하고 6개의 Y, Cb, Cr 데이타에 대해 동시에 수행한다. CMI 기반 프로그램은 baseline과 MMX 기반 프로그램들 보다 적은 픽셀 저장장소가 요구되므로 CMI 프로그램은 64개의 32비트 워드 메모리가 사용되는 반면 baseline과 MMX 프로그램은 128개의 32비트 워드 메모리가 사용되었다. 이러한 메모리 사이즈는 채택된 애플리케이션을 구현하는데 충분하다. 표 3은 사용된 아키텍처 모델들의 변수를 보여준다.

4.3 실험 방법론 구조

그림 7은 세가지 레벨(애플리케이션, 아키텍처, 테크놀로지)로 구성되어 있는 실험 방법론 구조를 보여준다.

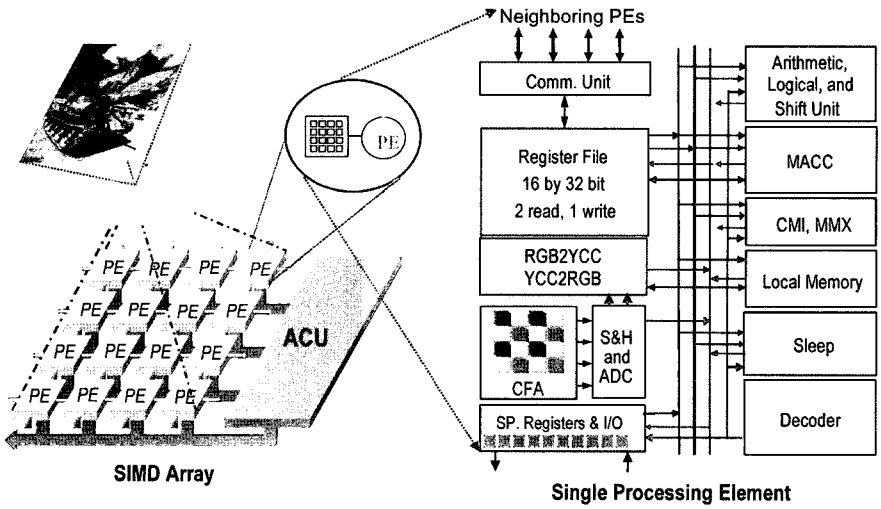

그림 6 SIMD 병렬 프로세서 아키텍처와 싱글 프로세싱 엘리먼트

표 3 SIMD 병렬 프로세서 아키텍처 파라미터

Parameter	Value
System Size	44×38 (1,584 PEs)
Image Sensor per PE (pixel per PE ratio)	4×4 (16 VPPE)
VLSI Technology	100 nm
Clock Frequency	80 MHz
Interconnection Network	Mesh
intALU/intMUL/Barrel Shifter/intMACC/Comm	1 / 1 / 1 / 1 / 1
MMX/CMI: intALU/intMACC	1 / 1
Local Memory Size (baseline/MMX/CMI)	128 32-bit word / 128 32-bit word / 64 32-bit word

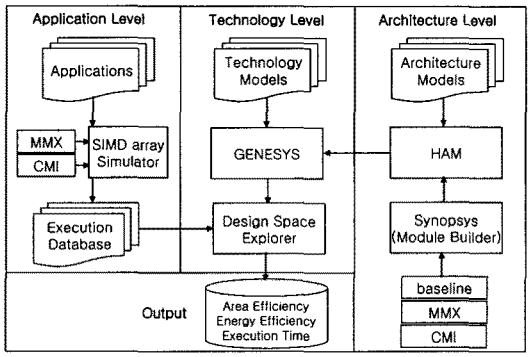

그림 7 SIMD 병렬 프로세서를 위한 실험 방법론 구조

애플리케이션 레벨에서는 명령어 레벨(instruction-level)의 SIMD 시뮬레이터가 사용되었다. 이 시뮬레이터는 세가지 형태의 프로그램(1. baseline ISA, 2. MMX ISA, 3 CMI ISA)의 사이클 개수, 동적 명령어 빈도, PE 이용률(utilization) 등의 실행 데이타를 제공한다. 아키텍처 레벨에서는 모델링된 아키텍처의 디자인 변수들을 계산하기 위해 Chai에 의해 제안된 SIMD 병렬 프로세서를 위한 이종 아키텍처 모델링 툴을 사용하였다[28]. 테크놀로지(technology) 레벨에서는 각 아키텍처 모델들의 테크놀로지 변수(latency, area, power, clock frequency)를 계산하기 위해 Generic System Simulator(GENESYS)를 사용하였다[29]. 마지막으로 위의 세 레벨에서 구해진 데이타베이스를 조합하여 각 경우에 대한 실행시간, 시스템 면적 효율성, 에너지 효율성을 결정하였다. 다음 절은 모델링된 아키텍처의 시스템 면적과 전력에 대해 설명한다.

4.4 테크놀로지 모델링을 사용한 시스템 면적 및 전력 평가

그림 8은 baseline-SIMD 아키텍처 대비 MMX-SIMD와 CMI-SIMD 아키텍처의 시스템 면적 및 전력을 보여준다. MMX는 전체 시스템 면적 및 전력에서 14%와 16%의 증가를 보인 반면, CMI는 줄어든 픽셀 워드 저장장소(local memory)를 사용하기 때문에 단지 3%의 시스템 면적 증가와 5%의 전력 증가를 보인다. 이러한 시스템 면적과 전력 결과는 애플리케이션 시뮬레이션 결과와 결합되어 각 경우의 성능, 면적 효율성, 에너지 효율성이 결정된다.

5. 모의 실험 및 결과 분석

각 아키텍처 모델링에 구현된 애플리케이션 태스크의 성능 및 효율을 결정하기 위해 본 논문은 cycle-accurate 시뮬레이션과 테크놀로지 모델링을 사용하였

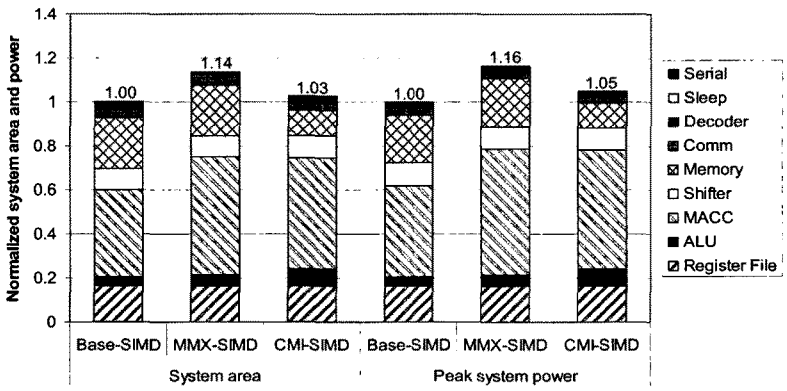

그림 8 Base-SIMD 대비 MMX-SIMD, CMI-SIMD의 시스템 면적 및 전력

표 4 평가 지표 요약

exec. time	sustained throughput	energy efficiency	area efficiency
$t_{exec} = \frac{C}{f_{ck}}$	$Th_{sust} = \frac{O_{exec} \cdot U \cdot N_{PE}}{t_{exec}}$	$\eta_E = \frac{O_{exec} \cdot U \cdot N_{PE}}{Energy}[\frac{\text{Gops}}{\text{Joule}}]$	$\eta_A = \frac{Th_{sust}}{Area}[\frac{\text{Gops}}{\text{s} \cdot \text{mm}^2}]$

C는 사이클 개수, f_{ck}는 클럭 주파수, O_{exec}는 수행된 연산 개수, N_{PE}는 PE 개수를 나타낸다.

다. 각 애플리케이션은 어셈블리 언어로 구현되었고, 세 가지 버전 프로그램(baseline, MMX, CMI)들은 같은 변수, 데이타 셋, 콜링 시퀀스를 가지고 있다. 표 4는 각 경우의 성능 비교를 위한 4가지 지표(execution time, sustained throughput, energy efficiency, area efficiency)를 보여 주고 있다.

5.1 성능평가 결과

그림 9는 SIMD 병렬 프로세서에서 수행된 baseline 프로그램 성능 대비 MMX및 CMI 기반 프로그램 성능(speedup in executed cycles) 비교를 보여준다. 모의 실험 결과, CMI 기반 프로그램은 모든 애플리케이션에서 MMX기반 프로그램보다 우수한 성능을 보여준다. CMI는 baseline 프로그램 대비 5.2~8.8배(평균 6.3배) 성능향상을 보여주는 반면, MMX는 baseline 대비 3~5배(평균 3.7배) 성능향상을 보여준다.

또한 CMI기반 프로그램(평균 194 Gops/sec)은 모든 애플리케이션에서 baseline(평균 113 Gops/sec)과 MMX 기반 프로그램(평균 155 Gops/sec)보다 높은 sustained throughput을 보여준다. 표 5는 1,584 PE 시스템에서 수행된 baseline, MMX, CMI 기반 프로그램들의 전체 성능을 보여준다. 다음 장에서는 CMI의 이점에 대해서 세부적으로 기술한다.

5.2 CMI의 이점

그림 10은 SIMD 병렬 프로세서에서 수행된 baseline 프로그램 대비 MMX 및 CMI 기반 프로그램의 벡터 명령어 분포도를 보여준다. 각 바(bar)는 논리/연산 유닛(ALU), 메모리(MEM), 커뮤니케이션 유닛(COMM), PE 동작 컨트롤 유닛(MASK), 이미지 픽셀 로딩 유닛(PIXEL), MMX, CMI로 세분화 된다. 그림에서 보는 바와 같이 CMI는 모든 프로그램에서 상당한 양의 명령어 개수를 감소시키는 결과를 보여준다. CMI프로그램은 baseline 프로그램에 비해 80.7%(ME)~88.6%(VMF) 명령어 개수를 감소시킨 반면, MMX 프로그램은 baseline 프로그램에 비해 단지 66.6%(ME)~80.1%(VMF) 명령어 개수를 감소시킨다. 특히 CMI는 정의된 명령어 특성에 의해 ALU와 메모리에서 많은 명령어 개수를 줄인다. 전체적으로 CMI는 MMX에 비해 각 프로그램에서 요구되는 벡터 명령어 개수를 감소시키는 능력이 우월하다.

5.3 에너지 효율 비교 결과

그림 11은 SIMD 병렬 프로세서에서 baseline 프로그램 대비 MMX 및 CMI 기반 프로그램의 에너지 소비 분포를 보여준다. 각 바(bar)는 기능 유닛(FU: ALU, Barrel Shifter, MACC), storage(Register file, Memory), others(Communication, Sleep, Serial I/O, Decoder)하드웨어의 에너지 분포를 나타낸다. 동일한 클럭 주파수(80MHz), 공정(100nm), 프로세서 파라미터에서 프로그램의 수행시간은 에너지 소비에 비례한다[30]. 예상한 대로, CMI 기반 프로그램은 baseline 프로그램과 비교하여 상당한 양의 소비 에너지를 감소시킨다. CMI 기반 프로그램은 baseline 프로그램 대비 80%(ME)~89%(VMF) 소비 에너지를 감소시킨 반면, MMX 기반 프로그램은 baseline 프로그램 대비 단지 60%(ME)~79%(VMF) 소비 에너지를 감소시키는 결과를 보인다. 특히, CMI는 상당한 수의 ALU와 memory 접근 수를

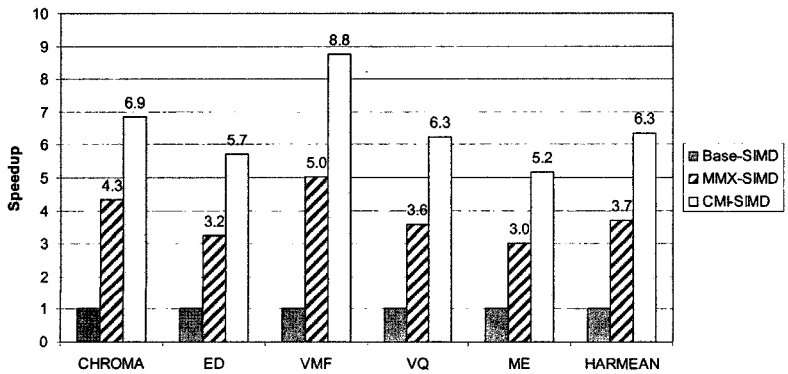

그림 9 Base-SIMD 병렬 프로세서 대비 MMX-SIMD, CMI-SIMD의 성능

표 5 80MHz에서 동작하는 1,584 PE 시스템에서 baseline, MMX, CMI 프로그램의 성능

Application	ISA	Memory Size		Vector Instruction	Scalar Instruction	System Utilization [%]	Sustained Throughput [Gops/sec]
		PE [Byte]	System [KB]				
CHROMA	Base	192	768	1,227	106	88	122
	MMX	192	768	283	106	100	155
	CMI	128	512	179	58	100	183
ED	Base	344	1,376	7,177	1,011	100	122
	MMX	344	1,376	2,117	371	100	160
	CMI	216	864	1,257	195	100	207
VMF	Base	244	976	37,397	7,891	93	118
	MMX	244	976	7,430	3,027	97	158
	CMI	172	688	4,264	1,523	94	203
VQ	Base	204	816	180,551	20,755	91	115
	MMX	204	816	50,503	20,755	97	133
	CMI	136	544	28,863	11,715	94	151
ME	Base	196	784	97,229	10,043	95	120
	MMX	196	784	32,502	10,379	98	140
	CMI	100	400	18,784	7,706	96	159

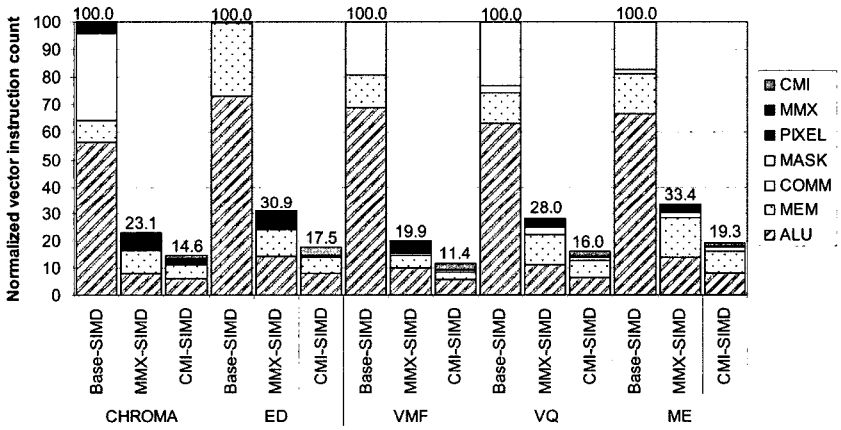

그림 10 Baseline 프로그램 대비 MMX 및 CMI 프로그램의 수행 벡터 명령어 개수

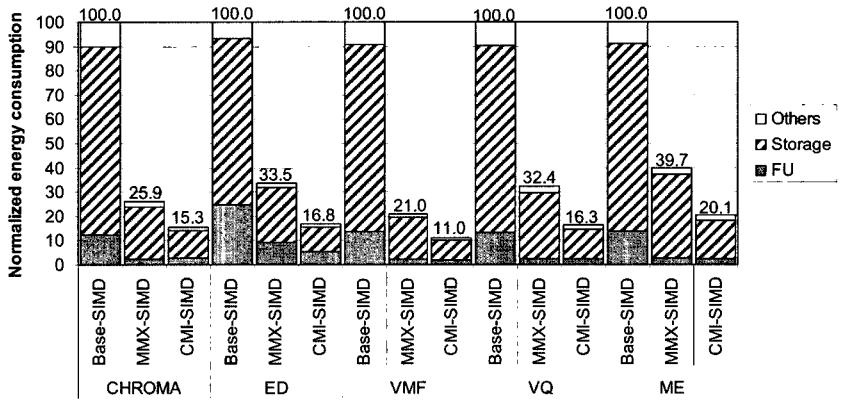

그림 11 Baseline 프로그램 대비 MMX 및 CMI 프로그램의 소비 에너지

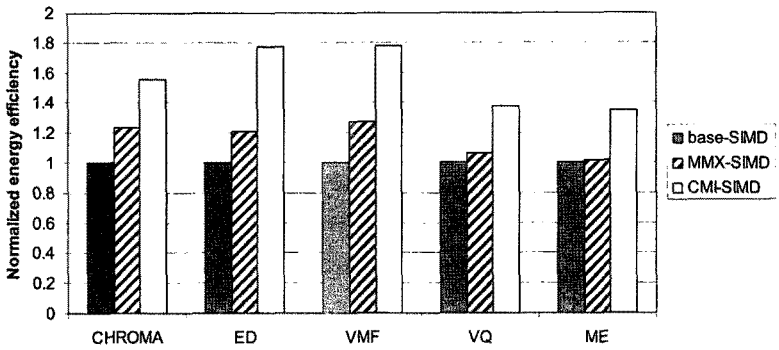

그림 12 Baseline 프로그램 대비 MMX 및 CMI 프로그램의 에너지 효율

줄이기 때문에 적은 양의 에너지가 ALU와 storage 유닛에서 소모된다.

그림 12는 baseline 프로그램 대비 MMX 및 CMI 기반 프로그램의 에너지 효율을 보여준다. CMI프로그램은 baseline프로그램 대비 50% 에너지 효율을 증가시킨 반면, MMX프로그램은 baseline 프로그램 대비 단지 11% 에너지 효율을 증가 시킨다. 이러한 결과는 CMI가 적은 시스템 전력을 증가시키는 반면, 높은 sustained throughput을 얻기 때문이다. 에너지 효율의 증가는 시스템의 배터리 수명을 증가시키는 결과를 가져온다.

5.4 시스템 면적 효율 비교 결과

그림 13은 SIMD 병렬 프로세서에서 baseline 프로그램 대비 MMX 및 CMI 기반 프로그램의 시스템 면적 효율을 보여준다. 에너지 효율 결과와 마찬가지로 CMI는 모든 애플리케이션에서 MMX보다 높은 시스템 면적 효율의 결과를 보여준다. CMI 프로그램은 baseline 프로그램 대비 52% 면적 효율을 증가시킨 반면, MMX프로그램은 baseline 프로그램 대비 단지 13% 면적 효율을 증가시킨다. 이러한 결과는 CMI가 적은 시스템 면적을 증가시키는 반면, 높은 sustained throughput을 얻기 때문이다. 시스템 면적 효율의 증가는 시스템의 컴포넌트 이용률을 증가시키는 결과를 가져온다.

6. 결 론

본 논문에서 고성능, 저전력 SIMD 병렬 프로세서를 위한 칼라미디어 명령어를 제안하였다. 제안한 칼라미디어 명령어는 기존의 멀티미디어 전용 명령어와 다르게 두 쌍의 16-bit YCbCr 데이타를 32-bit 레지스터에 저장하고 동시에 처리함으로써 성능을 향상시키는 동시에 칼라 이미지/비디오 데이타를 효율적으로 처리할 수 있다. 칼라미디어 명령어를 SIMD 병렬 프로세서에서 모의 실험한 결과, 칼라미디어 명령어 기반 프로그램은 baseline 프로그램보다 5.2~8.8배(평균 6.3배) 성능향상을 보인 반면, MMX 기반 프로그램은 baseline 프로그램보다 단지 3~5배(평균 3.7배) 성능향상을 보였다. 또한, 칼라미디어 명령어 기반 프로그램은 MMX 기반 프로그램보다 시스템 에너지 및 면적 효율 측면에서도 우수성을 보였다. 칼라미디어 명령어는 baseline 보다 에

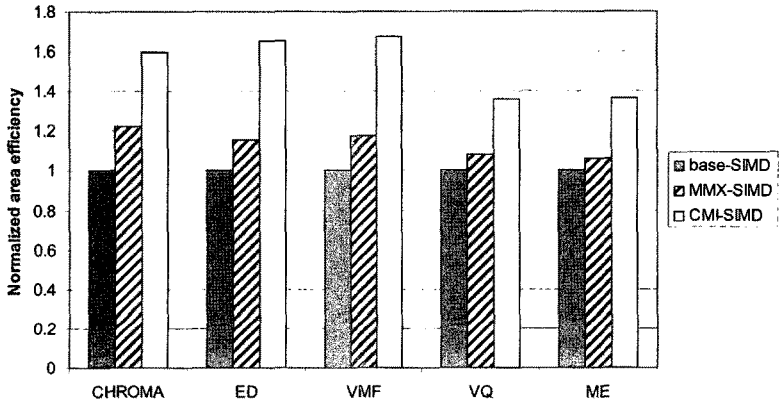

그림 13 Baseline 프로그램 대비 MMX 및 CMI 프로그램의 시스템 면적 효율

너지 효율에서 50%와 면적 효율에서 52%를 증가시킨 반면, MMX는 baseline 보다 에너지 효율에서 11%와 면적 효율에서 13%의 증가를 보였다. 칼라미디어 명령어는 이러한 에너지 및 면적 효율을 단지 3% 실리콘 면적 증가와 5% 시스템 전력 증가로 얻은 반면, MMX는 14%의 실리콘 면적 증가와 16% 시스템 전력 증가를 요구한다. 이러한 결과에서 칼라미디어 명령어가 칼라 이미지/비디오 데이타를 다루는 다양한 모바일 시스템에 적용될 경우 상당한 성능 향상 및 소비 에너지 감소가 기대된다.

참 고 문 헌

[1] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*, Springer Verlag, 2000.

[2] A. Peleg and U. Weiser, "MMX Technology Extension to the Intel Architecture," *IEEE Micro*, Vol.16, No.4, pp. 42-50, Aug. 1996.

[3] S. K. Raman, V. Pentkovski, and J. Keshava, "Implementing Streaming SIMD Extensions on the Pentium III Processor," *IEEE Micro*, Vol.20, No.4, pp. 28-39, 2000.

[4] R. B. Lee, "Subword Parallelism with MAX-2," *IEEE Micro*, Vol.16, No.4, pp. 51-59, Aug. 1996.

[5] M. Tremblay, J. M. O'Connor, V. Narayanan, and L. He, "VIS Speeds New Media Processing," *IEEE Micro*, Vol.16, No.4, pp. 10-20, Aug. 1996.

[6] R. Sites, Ed., *Alpha Reference Manual*, Burlington, MA: Digital, 1992.

[7] H. Nguyen and L. John, "Exploiting SIMD Parallelism in DSP and Multimedia Algorithms using the AltiVec Technology," in *Proc. Intl. Conf. on Supercomputer*, pp. 11-20, June 1999.

[8] TMS320C64x families: http://www.bdti.com/procsum/tic64xx.htm.

[9] J. Fridman and Z. Greenfield, "The TigerSHARC DSP architecture," in *Proc. IEEE/ACM Intl. Sym. on Computer Architecture*, pp. 124-135, May 1999.

[10] ARM9 Family: http://www.arm.com/products/CPUs/families/ARM9Family.html.

[11] A. D. Blas et. al., "The UCSC Kestrel Parallel Processor," *IEEE Trans. on Parallel and Distributed Systems*, Vol.16, No.1, pp. 80-92, Jan. 2005.

[12] A. Gentile and D. S. Wills, "Portable Video Supercomputing," *IEEE Trans. on Computers*, Vol.53, No.8, pp. 960-973, Aug. 2004.

[13] J. Kim and D. S. Wills, "Quantized color instruction set for multimedia-on-demand applications," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 141-144, July 2003.

[14] J. Kim and D. S. Wills, "Evaluating a 16-bit YCbCr (6:5:5) color representation for low memory, embedded video processing," in *Proc. of the IEEE Intl. Conf. on Consumer Electronics*, pp. 181-182, Jan. 2005.

[15] P. Ranganathan, S. Adve, and N. P. Jouppi, "Performance of image and video processing with general-purpose processors and media ISA extensions, in *Proc. of the 26th Intl. Sym. on Computer Architecture*, pp. 124-135, May 1999.

[16] R. Bhargava, L. John, B. Evans, and R. Radhakrishnan, "Evaluating MMX technology using DSP and multimedia applications," in *Proc. of IEEE/ACM Sym. on Microarchitecture*, pp. 37-46, 1998.

[17] N. Slingerland and A. J. Smith, "Measuring the performance of multimedia instruction sets," *IEEE Trans. on Computers*, Vol.51, No.11, pp. 1317-1332, Nov. 2002.

[18] A. Krikelis, I. P. Jalowiecki, D. Bean, R. Bishop, M. Facey, D. Boughton, S. Murphy, and M. Whitaker, "A programmable processor with 4096 processing units for media applications," in *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Vol.2, pp. 937-940, May 2001.

[19] L. W. Tucker and G. G. Robertson, "Architecture and applications of the connection machine," *IEEE Computer*, Vol.21, No.8, pp. 26-38, 1988.

[20] "Connection machine model CM-2 technical summary," Thinking Machines Corp., version 51, May 1989.

[21] MarPar (MP-2) System Data Sheet. MarPar Corporation, 1993.

[22] M. J. Irwin, R. M. Owens, "A Two-Dimensional, Distributed Logic Processor," *IEEE Trans. on Computers*, Vol.40, No.10, pp. 1094-1101, 1991.

[23] M. Bolotski, R. Armithrajah, W. Chen, "ABACUS: A High Performance Architecture for Vision," in *Proceedings of the International Conference on Pattern Recognition*, 1994.

[24] C. C. Yang, "Effects of coordinate systems on color image processing," MS Thesis, University of Arizona, Tucson, 1992.

[25] H.-M. Hang and B. G. Haskell, "Interpolative vector quantization of color images," *IEEE Trans. Commun.*, Vol.COM-36, No.4, pp. 465-470, April 1988.

[26] S. C. Kwatra, C. M. Lin, and W. A. Whyte, "An adaptive algorithm for motion compensated color image coding," *IEEE Trans. Commun.*, Vol. COM-35, pp. 747-754, July 1987.

[27] J. Suh and V. K. Prasanna, "An Efficient Algorithm for Out-of-core Matrix Transposition," *IEEE Trans. on Computers*, Vol.51, No.4, pp. 420-438, April 2002.

[28] S. M. Chai, T. M. Taha, D. S. Wills, and J. D. Meindl, "Heterogeneous architecture models for interconnect-motivated system design," *IEEE Trans. VLSI Systems*, special issue on system level interconnect prediction, Vol.8, No.6, pp. 660-670,

Dec. 2000.
[29] J. C. Eble, V. K. De, D. S. Wills, and J. D. Meindl, "A generic system simulator (GENESYS) for ASIC technology and architecture beyond 2001," in *Proc. of the Ninth Ann. IEEE Intl. ASIC Conf.*, pp. 193-196, Sept. 1996.
[30] V. Tiwari, S. Malik, and A. Wolfe, "Compilation Techniques for Low Energy: An Overview," in *Proc. of the IEEE Intl. Symp. on Low Power Electron.*, pp. 38-39, Oct. 1994.

김 철 홍

1998년 서울대학교 컴퓨터공학과 학사 2000년 서울대학교 대학원 컴퓨터공학부 석사. 2006년 서울대학교 대학원 전기컴퓨터공학부 박사. 2005년~2007년 삼성전자 반도체총괄 SYS.LSI사업부 책임연구원. 2007년~현재 전남대학교 전자컴퓨터공학부 교수. 관심분야는 임베디드시스템, 컴퓨터구조, SoC 설계, 저전력 설계

김 종 면

1995년 명지대학교 전기공학과 학사. 2000년 University of Florida Electrical & Computer Engineering 석사. 2005년 Georgia Institute of Technology Electrical & Computer Engineering 박사. 2005년~2007년 삼성종합기술원 전문연구원. 2007년~현재 울산대학교 컴퓨터정보통신공학부 교수 관심분야는 임베디드 SoC 설계, 컴퓨터구조, Application-Specific 프로세서 설계, 병렬처리