고성능, 저전력 임베디드 비디오 프로세서를 위한 YUV 인식 명령어의 시뮬레이션

(Simulation of YUV-Aware Instructions for High-Performance, Low-Power Embedded Video Processors)

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

요 약 멀티미디어 응용과 무선통신 네트워크의 발전 속도가 급속하게 빨라짐에 따라 고성능, 저전력 멀티미디어 처리기술에 대한 소비자의 요구가 급증하고 있다. 이에 본 논문은 고성능, 저전력 임베디드 비디오 프로세서를 위한 YUV (Y: 휘도신호, U, V: 색차신호) 인식 명령어를 제안하고자 한다. 기존의 멀티미디어 전용 명령어 (e.g., MMX, SSE, VIS, AltiVec)는 일반적인 서브워드 병렬 기법을 이용하여 적당한 성능향상을 꾀하는 반면, 제안하는 YUV 인식 명령어는 두 쌍의 16-bit YUV (6-bit Y, 5-bits U, V) 데이타를 32-bit 레지스터에 저장하여 동시에 처리함으로써 칼라 비디오 처리 성능을 효율적으로 향상시킬 수 있다. 또한 데이타 포맷 사이즈를 줄임으로써 전체 시스템의 비용을 절감할 수 있다. 임베디드 슈퍼스칼라 프로세서에서 모의 실험한 결과, YUV 인식 명령어 기반 프로그램은 baseline 프로그램에 비해 3.9배 성능 향상을 보인 반면, 동일한 프로세서 환경에서 Intel의 대표적인 멀티미디어 명령어인 MMX기반 프로그램은 baseline 프로그램보다 단지 2.1배의 성능 향상을 보인다. 또한 YUV 인식 명령어는 멀티미디어 애플리케이션에 대해 평균 75.8% 소모 에너지를 감소시킨 반면, MMX는 단지 54.8%의 소모 에너지를 감소시키는 결과를 보인다.

키워드 : YUV 비디오 데이타 처리, 멀티미디어 명령어, 임베디드 슈퍼스칼라 프로세서, 고성능 비디오 프로세서

Abstract With the rapid development of multimedia applications and wireless communication networks, consumer demand for video-over-wireless capability on mobile computing systems is growing rapidly. In this regard, this paper introduces YUV-aware instructions that enhance the performance and efficiency in the processing of color image and video. Traditional multimedia extensions (e.g., MMX, SSE, VIS, and AltiVec) depend solely on generic subword parallelism whereas the proposed YUV-aware instructions support parallel operations on two-packed 16-bit YUV (6-bit Y, 5-bits U, V) values in a 32-bit datapath architecture, providing greater concurrency and efficiency for color image and video processing. Moreover, the ability to reduce data format size reduces system cost. Experiment results on a representative dynamically scheduled embedded superscalar processor show that YUV-aware instructions achieve an average speedup of 3.9x over the baseline superscalar performance. This is in contrast to MMX (a representative Intel's multimedia extension), which achieves a speedup of only 2.1x over the same baseline superscalar processor. In addition, YUV-aware instructions outperform MMX instructions in energy reduction (75.8% reduction with YUV-aware instructions, but only 54.8% reduction with MMX instructions over the baseline).

Key words : YUV video data processing, Multimedia instructions, Embedded superscalar processors, High-performance video processors

· 이 논문은 2007년 울산대학교의 연구비에 의하여 연구되었음(2007-0045)
† 정 회 원 : 전남대학교 전자컴퓨터공학부 교수
cheolhong@gmail.com
†† 정 회 원 : 울산대학교 컴퓨터정보통신공학부 교수
jongmyon.kim@gmail.com
(Corresponding author임)
논문접수 : 2007년 9월 6일
심사완료 : 2007년 10월 15일

1. 서 론

정보화 사회가 급속하게 발전하고 모바일 컴퓨팅 환경이 크게 대두됨에 따라 멀티미디어의 방대한 데이타를 얼마나 효율적으로 처리하는가 하는 문제가 중요한 이슈가 되고 있다. 이러한 요구에 맞추어 개인용 컴퓨터

나 워크스테이션에서 사용되는 있는 고성능 범용프로세서 제조회사들은 멀티미디어 처리 성능을 효율적으로 향상시킬 수 있는 멀티미디어 전용 명령어를 프로세서 내부에 구현하였다. 멀티미디어 전용 명령어의 예로는 Intel의 MMX[1]와 SSE[2], Hewlett Packard의 MAX-2 [3], Sun Microsystems의 VIS[4], Motorola의 AltiVec [5] 등이 있다. 이러한 멀티미디어 전용 명령어들은 SIMD(Single Instruction, Multiple Data) 가속기능을 이용하여 하나의 넓은 레지스터에 (e.g., 32-, 64-bit, and 128-bit) 짧은 데이타를 (e.g., byte-wise pixel values) 패킹하여 동시에 병렬 연산하는 방법(subword parallelism)을 사용하여 영상 및 오디오 처리에 수 배 이상의 성능 향상을 가져온다[4].

그러나, 벡터 픽셀(e.g., 3-bytes in RGB, YUV) 처리에 사용되는 애플리케이션[6]들은 멀티미디어 SIMD 명령어의 입력으로 요구되는 2의 자승 영역(boundaries)으로 데이타가 정렬되어 있지 않기 때문에 처리 성능은 그다지 효율적이지 못하다. 더불어 SIMD연산을 위해 데이타의 패킹이나 언패킹(packing/unpacking)과 같은 전·후 처리 오버헤드(overhead)가 상당해 심지어 성능 저하를 초래한다[7]. 그림 1은 기존의 멀티미디어 SIMD 명령어가 band-interleaved RGB 구조에서(한 픽셀에 대하여 여러 밴드의 데이타를 순차적으로 기록하고, 다음 픽셀에 대하여도 같은 방법을 사용하여 기록) 동작하는 예를 보여준다. 이와 같은 경우, unused field에서는 SIMD 연산의 효율을 얻지 못한다. 이러한 문제점은 전체 영역에 대하여 한 밴드의 데이타를 저장하는 band-separated RGB구조를 통해 해결할 수 있지만, 상당한 데이타 전·후 처리가 요구되므로 벡터 픽셀 처리에는 적합하지 못한 단점을 가지고 있다. 또한 RGB 칼라 모델은 인간의 지각(perceptual) 속성에 적합하지 않기 때문에 대부분의 이미지/비디오 처리 애플리케이션에서 YUV(luminance-chrominance) 칼라 모델이 채택되고 있다[8].

이에 본 논문은 두 쌍의 16-bit YUV 데이타를 32-bit

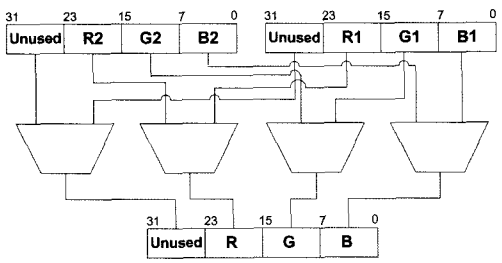


그림 1 Band-interleaved RGB 구조에서 멀티미디어 SIMD 명령어 동작의 예

레지스터에 저장하여 동시에 병렬 연산하는 YUV 인식 명령어(YUV-aware instructions)를 제안하여 RGB형 멀티미디어 명령어에 내재한 문제점을 해결한다. 기존의 멀티미디어 SIMD 명령어와 달리, YUV 인식 명령어는 휴먼 지각 칼라 모델의 특성을 이용하여 높은 병렬성의 추구할 뿐만 아니라 벡터 픽셀 처리의 효율성을 높인다. 4-way 슈퍼스칼라(superscalar) 프로세서에서 선택된 칼라 이미징(imaging) 애플리케이션들을 수행한 결과, YUV 인식 명령어 기반 프로그램은 기본(baseline) 프로그램 성능보다 3~5.8배(평균 3.9배) 성능을 향상시킨다. 반면에 Intel의 대표적인 멀티미디어 명령어인 MMX기반 프로그램은 동일한 슈퍼스칼라 프로세서에서 baseline 프로그램보다 단지 1.6~3.2배(평균 2.1배) 성능 향상의 결과를 나타낸다. 또한 YUV 인식 명령어는 MMX 명령어보다 소모 에너지 측면에서도 우월성을 보여준다. YUV 인식 명령어 기반 프로그램은 baseline 프로그램에 비해 68%~83%(평균 75.8%) 에너지를 감소시킨 반면, MMX 기반 프로그램은 baseline 프로그램에 비해 단지 39%~69%(평균 54.8%) 에너지를 감소시킨다. 더불어 YUV 인식 명령어는 슈퍼스칼라 프로세서의 low-issue rate에서 상대적으로 높은 성능 향상을 보여준다. YUV 인식 명령어는 baseline 1-way issue 성능 보다 4.7배 성능 향상을 나타낸 반면 baseline 16-way issue 성능 보다는 3배 성능 향상을 보여준다. 이러한 결과들로부터 YUV 인식 명령어는 high-issue rate 하드웨어가 너무 비싼 임베디드 멀티미디어 시스템에 적용될 경우 상당한 성능 향상 및 에너지 효율을 기대할 수 있다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하고자 하는 임베디드 슈퍼스칼라 프로세서용 YUV 인식 명령어를 요약하고, 3장에서는 선택된 칼라 이미징 애플리케이션, 슈퍼스칼라 아키텍처 모델링, 그리고 YUV 인식 명령어의 성능 평가를 위한 시뮬레이션 방법론을 소개한다. 4장에서는 제안된 방법의 시뮬레이션 결과와 그에 대한 분석을 기술하고, 끝으로 5장에서는 이 논문의 결론을 맺는다.

2. YUV 인식 명령어

YUV 인식 명령어는 칼라 이미지/비디오 처리 가속화에 목표를 두고 있다. YUV 인식 명령어는 32-bit 데이타패스 프로세서와 연계하여 두 쌍의 16-bit YUV (6-bit Y, 5-bits Cr and Cb) 데이타를 하나의 32-bit 레지스터에 저장하여 동시에 처리함으로써 성능 향상을 추구할 뿐만 아니라 YUV 비디오 데이타 처리에 높은 유연성을 보여준다. 그림 2는 32-bit YUV 인식 명령어의 한 예를 보여준다. 4개의 5-bit와 2개의 6-bit 기능

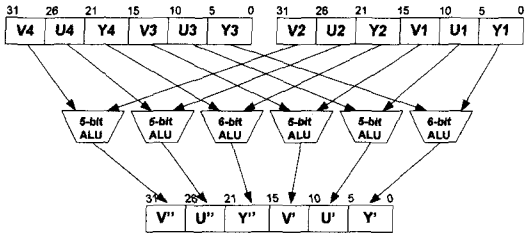


그림 2 32-bit YUV 인식 명령어

유닛(functional units, FUs)을 사용하여 두 쌍의 6-5-5 bit, 한 쌍의 11-11-10 bit, 그리고 32 bit 단위로 연산할 수 있는 분할된 구조로 설계된다. 또한 YUV 인식 명령어는 칼라 처리 가산기(accumulator)를 내장하여 타이트(tight)하게 패킹(packing)된 데이타 처리에서 발생하는 오버플로(overflow)와 같은 문제를 없앤다. YUV 인식 명령어는 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.

2.1 병렬 산술·논리(ALU) 명령어

병렬 산술·논리 명령어에는 가산(ADD_VUY), 감산(SUBTRACT_VUY), 그리고 평균 명령어(AVERAGE_VUY)가 있다. 가산/감산 명령어에는 오버플로가 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. 평균 명령어는 각 레지스터에 저장된 각 쌍의 서브워드(subword)를 동시에 평균 처리하기 때문에 blending과 같은 알고리즘에 유용하다.

2.2 병렬 비교(Compare) 명령어

병렬 비교 명령어에는 CMPEQ_VUY, CMPNE_VUY, CMPGE_VUY, CMPGT_VUY, CMPLE_VUY, CMOV_VUY(conditional move), MIN_VUY, MAX_VUY 등이 있다. 이러한 명령어는 두 개의 소스 레지스터(source register)에 저장되어 있는 서브워드의 쌍들을 동시에 비교하는 연산자이다. 처음 7개의 명령어는 입력 데이타에 수행되는 condition query를 추출할 때 유용하며, 마지막 두 명령어(MIN_VUY, MAX_VUY)는 픽셀들의 최소 및 최대값을 추출하는 필터링(filtering) 알고리즘에 유용하다.

2.3 치환 (Permute) 명령어

치환 명령어에는 BCAST_VUY(broadcast), MIX_VUY, ROTATE_VUY 등이 있다. 이러한 명령어들은 패킹된 데이타 형에서 각 데이타 요소들의 순서를 재구성하기 위해 사용된다. 그림 3, 4는 MIX_VUY 명령어

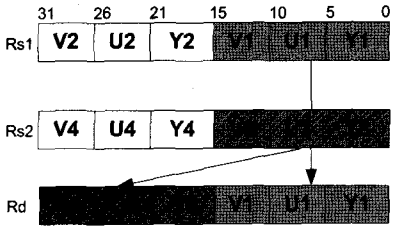


그림 3 MIX_VUY 명령어 예

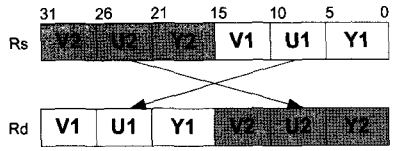


그림 4 ROTATE_VUY 명령어 예

와 ROTATE_VUY 명령어를 나타내며, 이러한 명령어들은 벡터 픽셀 전환(vector pixel transposition)이나 매트릭스 변환(matrix transposition) 등에 유용하다[9].

2.4 특정목적(Special-Purpose) 명령어

특정목적 명령어에는 ADACC_VUY (absolute-differences accumulate), MACC_VUY(multiply-accumulate), RAC(read accumulate), ZACC(zero accumulate) 등이 있으며, 계산처리 성능 면에서 가장 유용한 YUV 인식 명령어들이다. 예를 들어, ADACC_VUY는 다양한 움직임 추정(motion estimation) 알고리즘에 사용된다. 그림 5에서 보듯이 두 개의 소스 레지스터에 있는 각 쌍의 서브워드는 절대 차로 계산되고 그 결과 값은 패킹된 가산기에 저장된다. MACC_VUY는 디지털 필터링(digital filtering) 또는 컨볼루션(convolution)과 같은 vector dot-product 계산을 수행하는 DSP 알고리즘에 유용하다. 마지막 두 명령어(RAC, ZACC)는 가산기를 관리하는 명령어이다.

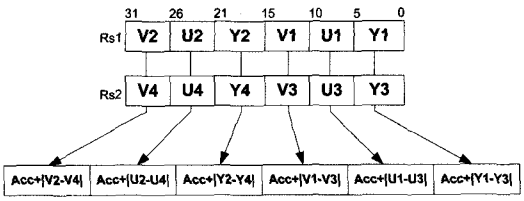


그림 5 ADACC_VUY 명령어 예

3. YUV 인식 명령어 평가방법

3.1 칼라 이미징 애플리케이션

YUV 인식 명령어의 성능 평가를 위해 다음과 같은 다섯 가지 이미징(imaging) 애플리케이션을 선택하고

구현되었다. 선택된 이미징 애플리케이션은 Sobel 오퍼레이터(operator)를 이용한 칼라 에지 검출(Edge Detection, ED), 분리 미디언 필터링(Separable Median Filtering, SMF), 벡터 미디언 필터링(Vector Median Filtering, VMF), 벡터 양자화(Vector Quantization, VQ), MPEG의 핵심 커널(core kernel)인 움직임 추정(Motion Estimation, ME) 등이다. 표 1에서 간단하게 요약한 이러한 애플리케이션들은 인터넷의 스트리밍 비디오(streaming video), 실시간(real-time) 비디오 개선 및 분석, 씬 시각화(scene-visualization)와 같은 실 세계(real-world)에서 중요한 컴포넌트(component)로서 사용되고 있고 미래에도 사용될 것이다. 위의 애플리케이션들은 모의 실험에서 3밴드(band) CIF 해상도(resolution) 입력을 가지고 구현된다.

3.2 아키텍처 모델 및 툴

그림 6은 임베디드 슈퍼스칼라 프로세서에서 제안하는YUV인식 명령어의 모의 실험 방법을 보여준다. 모의 실험은 비순차적(out-of-order) 슈퍼스칼라 모델링 툴인 SimpleScalar 시뮬레이터[10]에 MMX 및 YUV 인식 명령어를 추가하여 수행되었다. MMX와 YUV 인식 명령어 기반 프로그램을 생성하기 위해, 어셈블리 파일(assembly files)에 MMX와 YUV 인식 명령어를 기술하고, 프로파일링(profiling)을 통해 가장 수행시간이 많이 소모되는(time-consuming) 커널(kernel)들의 baseline 어셈블리 랭귀지 부분들을 해당 MMX 혹은 YUV 인식 명령어로 대치하였다. 본 연구의 타겟 플랫폼(target platform)은 임베디드 시스템(embedded system)이므로 오퍼레이팅 시스템 인터페이스 코드(e.g., 파일 시스템 접근)는 포함하지 않았다. 또한 각 명령어 기반 프로그램들에 대해 칼라 변환(color conversion)에서 요구되는 오버헤드는 성능 평가에서 제외하였다. 예를 들어, baseline, MMX, 그리고 YUV 인식 명령어는 같은 데이타 포맷에서 YUV 데이타를 입력으로 받는다(e.g., 32-bit 레지스터에서 baseline과 MMX는 |Unused|V|U|Y|형으로 레지스터에 저장하고, YUV 인식 명령어는 |V|U|Y|V|U|Y|형으로 레지스터에 저장한다).

또한, 각 프로그램에서 소모되는 에너지를 평가하기 위해 아키텍처 레벨 전력 모델링 툴인 Wattch 시뮬레이터[11]를 사용하였다. MMX와 YUV 인식 명령어의 기능 유닛(functional units, FUs)의 전력소모 평가를 위해 각 baseline, MMX, YUV인식 명령어의 기능 유닛들은 0.18 um 공정을 가정하여 구현되었고, 얻어진 전력 변수 값들은 Wattch 시뮬레이터에 적용되었다. 표 2는 각 명령어 기능 유닛의 소모 전력 비교를 보여준다.

표 3은 제안하는 YUV 인식 명령어의 성능 및 에너지 효율성을 검증하기 위해 사용된 프로세서 모델의 사양을 나타낸다. YUV 인식 명령어를 다양한 범위의 슈퍼스칼라 프로세서 모델에서 검증하기 위해 한 사이클당 1~16 명령어의 이슈 넓이(issue width) 와 16~256 명령어 윈도우 사이즈를 변화시키면서 수행하였다. 모의

표 1 칼라 이미징 애플리케이션 요약

애플리케이션	설 명
Edge Detection	3가지 밴드(Y,U,V)에 Sobel 오퍼레이터를 동시 적용하여 얻어진 지역 변화(local changes)를 통하여 칼라 에지 정보를 검출한다.
Separable Median Filtering	이미지로부터 노이즈(noise)를 제거하기 위해 각 밴드의 어떤 특정한 주변 영역내의 (3×3 window) 화소 농도의 중간 값을 구하여 원하는 화소의 농도로 처리한다.
Vector Median Filtering	Separable Median Filtering과 마찬가지로 이미지로부터 노이즈를 제거하는 방법으로서 3가지 밴드(Y,U,V)에 미디언 필터링을 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 핵심기술로서 이전 프레임(frame)과 현재 프레임의 차를 이용하여 움직임을 추정하고 이를 보상해주는 압축 기법이다.

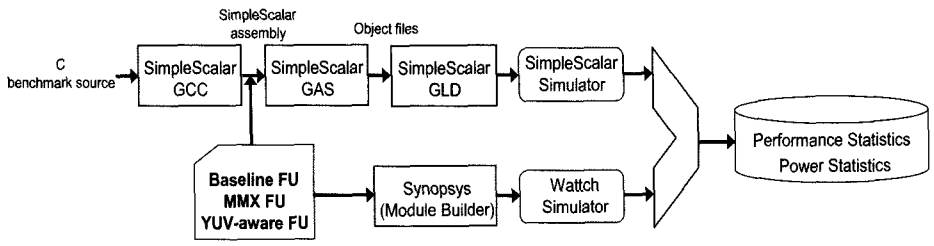


그림 6 임베디드 슈퍼스칼라 프로세서에서 YUV 인식 명령어 기반 프로그래밍 방법

표 2 1.62 voltage에서 1GHz로 동작하는 32-bit FU의 동적 전력 평가

Instructions	ALU	MAC
Baseline	12.5mW	262.2mW
MMX	15.0mW	305.2mW
YUV-aware	18.8mW	299.9mW

실험에서 에너지 소모 관련 변수들은 600MHz, 0.18 um 공정을 가정하여 구현되었다.

4. 모의 실험 및 결과 분석

이 장에서는 제안하는 YUV 인식 명령어의 성능 및 에너지 효율성을 검증하기 위해 baseline 슈퍼스칼라 명령어, MMX 명령어, YUV 인식 명령어 기반 프로그램을 구현하고 비교 분석한다. 성능평가 실험은 Simple-Scalar 시뮬레이터를 수정하여 수행하였고, 각 명령어 기반 프로그램은 동일한 인자(parameters), 데이타 셋(data sets), 콜링 시퀀스(calling sequences)를 가진다. 또한, 소모 에너지를 평가하기 위해 Wattch power 시뮬레이터를 수정하였다. 모의 실험의 비교 지표로써 실행된 다이나믹(dynamic) 명령어 수, 실행 사이클 수, 소모 에너지가 사용된다.

4.1 성능평가 결과

그림 7은 다섯 쌍의 서로 다른 웨이(way) 슈퍼스칼라 프로세서에서 수행된 baseline 프로그램 성능 대비 MMX및 YUV 인식 명령어 기반 프로그램 성능(speedup in executed cycles)을 보여준다. 모의 실험 결과, YUV 인식 명령어는 모든 애플리케이션에서 MMX 명령어보다 우수한 성능을 보여준다. 예를 들어, 4-way 슈퍼스칼라 프로세서에서 YUV 인식 명령어 기반 프로그램은 baseline 프로그램 성능 보다 3~5.8배 성능 향상을 보여주는 반면, MMX명령어 기반 프로그램은 baseline 프로그램에 비해 단지 1.6~3.2배 성능 향상을 보여준다. 한가지 흥미로운 점은 YUV 인식 명령어는 low-issue rates에서 상대적으로 높은 성능 향상을 보여준다. 예를 들어, YUV 인식 명령어는 baseline 1-way issue 성능보다 4.7배의 성능 향상을 보인 반면, baseline 16-way issue 성능보다는 3배의 성능 향상을 보여준다. 이러한 결과들로부터 YUV 인식 명령어는 high-issue rate가 너무 비싼 멀티미디어 임베디드 시스템에 적합한 구조임이 증명된다. 다음 장에서는 YUV 인식 명령어의 이점에 대해서 세부적으로 기술한다.

4.2 YUV 인식 명령어의 이점

그림 8은 4-way 슈퍼스칼라 프로세서에서 수행된 baseline 프로그램 대비 MMX 및 YUV 인식 명령어 기반 프로그램의 다이내믹 명령어 분포도를 보여준다. 각 바(bar)는 기능 유닛(functional unit, FU), 제어(control), 메모리(memory), MMX, YUV 인식 명령어로 세분화 된다. 그림에서 보는 바와 같이 YUV 인식 명령어는 모든 프로그램에서 상당한 양의 명령어 개수를 감소시키는 결과를 보여준다.

FU 명령어 개수 감소. YUV 인식 ALU 명령어는 두 쌍의 16 비트 YUV 데이타를 동시에 산술 처리하므로 다수의 baseline ALU 명령어와 인덱스 및 어드레스 생성에 필요한 루프(loop) 오버헤드 명령어를 줄인다. YUV 인식 명령어 기반 프로그램은 baseline 프로그램에 비해 73%~86%(평균 81%)의 FU 명령어 개수를 감소시킨다. 반면에, MMX 기반 프로그램은 baseline 프로그램보다 단지 47%~73%(평균 64%)의 FU 명령어 개수를 감소시킨다.

표 3 모의 실험에서 사용되는 슈퍼스칼라 프로세서 인자

Parameter	1-way	2-way	4-way	8-way	16-way
Fetch/decode/issue/ commit width	1	2	4	8	16
intALU/intMUL/fpALU/ fpMUL/Mem	1/1/1/1	2/1/1/2	4/2/2/1/4	8/4/2/1/8	16/8/4/1/16
RUU (window) size	16	32	64	128	256
LSQ (Load Store Queue)	8	16	32	64	128
Main memory width	32 bits	64 bits	128 bits	256 bits	256 bits
Branch Predictor	Combined predictor (1 K entries) of bimodal predictor (4 K entries) table and 2-level predictor (2-bit counters and 10-bit global history)				
L1 D-cache	128-set, 4-way, 32-byte line, LRU, 1-cycle hit, total of 16 KB				
L1 I-cache	512-set, direct-mapped 32-byte line, LRU, 1-cycle hit, total of 16 KB				
L2 unified cache	1024-set, 4-way, 64-byte line, LRU, 6-cycle hit, total of 256 KB				
Main memory latency	50 cycles for first chunk, 2 thereafter				
Instruction TLB	16-way, 4096 byte page, 4-way, LRU, 30 cycle miss penalty				
Data TLB	32-way, 4096 byte page, 4-way, LRU, 30 cycle miss penalty				

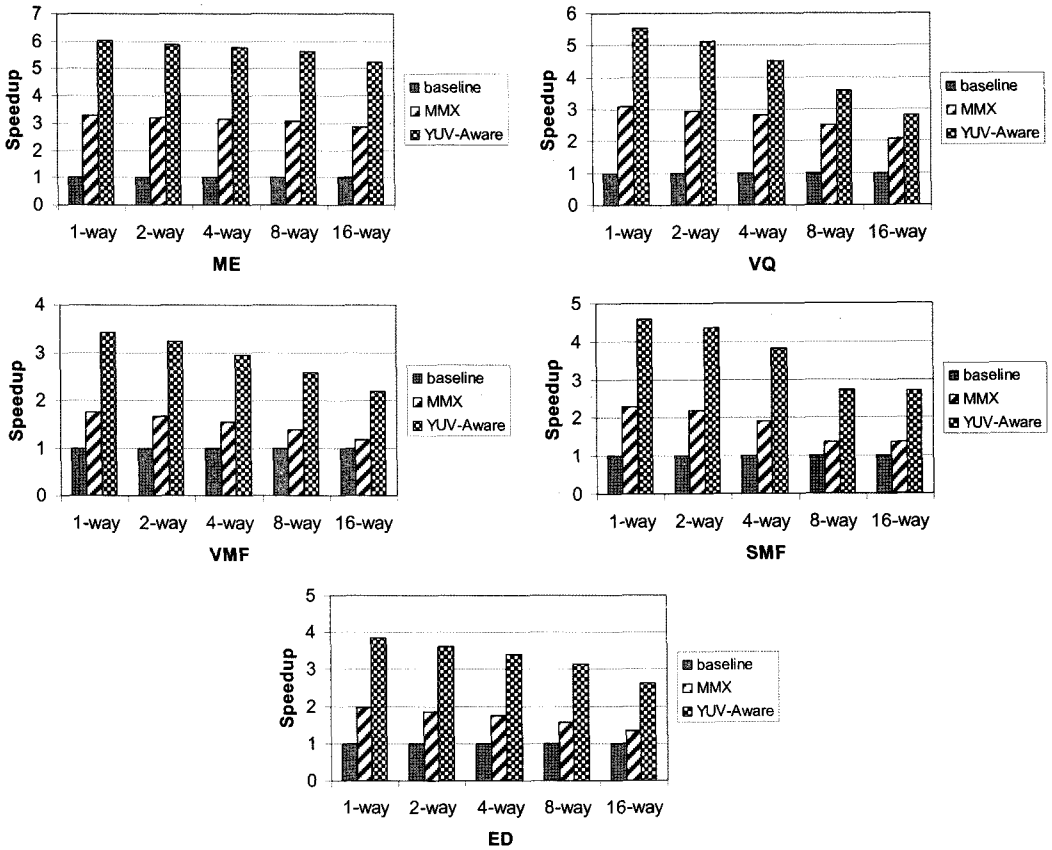


그림 7 다른 issue-rate 슈퍼스칼라 프로세서에서 baseline 프로그램 성능 대비 MMX 및 YUV 인식 명령어 기반 프로그램 성능 비교

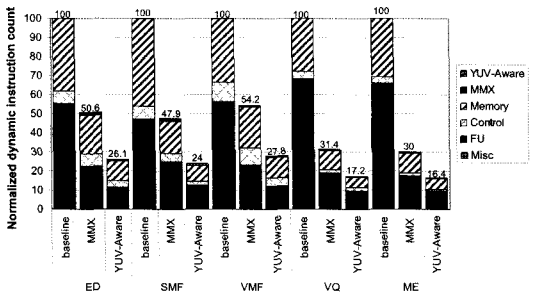


그림 8 수행된 다이내믹 명령어 수에서 YUV 인식 명령어의 효과

제어 명령어 개수 감소. YUV 인식 제어(control) 명령어는 다수의 조건 혹은 분기 명령어를 하나의 YUV 인식 명령어로 대치할 수 있다. 그 결과 YUV 인식 명령어는 모든 프로그램에서 상당한 양의 제어 명령어를 감소시킨다. YUV 인식 명령어 기반 프로그램은 baseline 프로그램보다 47%~76%(평균 60%)의 제어 명령어를 감소시킨 반면, MMX 기반 프로그램은 단지 2%~57%(평균 26%)의 제어 명령어를 감소시킨다.

메모리 명령어 개수 감소. 하나의 YUV 인식 명령어는 다수의 패킹된 데이타를 메모리로부터 레지스터로, 또는 레지스터로부터 메모리로 전송할 수 있다. 게다가 특정 목적 YUV 인식 명령어(MACC_VUY and ADACC_VUY)는 중간 계산 결과값을 메모리에 저장하는 대신에 가산기에 저장함으로써 훨씬 많은 메모리 명령어를 없앤다. YUV 인식 명령어 기반 프로그램은 baseline 프로그램보다 68%~83%(평균 78%)의 메모리 명령어를 감소시킨 반면, MMX 기반 프로그램은 단지 37%~66%(평균 57%)의 메모리 명령어를 감소시킨다.

전체적으로 YUV 인식 명령어는 MMX 명령어에 비해 각 프로그램에서 요구되는 다이내믹 명령어 개수를 감소시키는 능력이 우월하다.

4.2 에너지 소모 결과

그림 9는 4-way 슈퍼스칼라 프로세서에서 baseline 프로그램 대비 MMX 및 YUV 인식 명령어 기반 프로

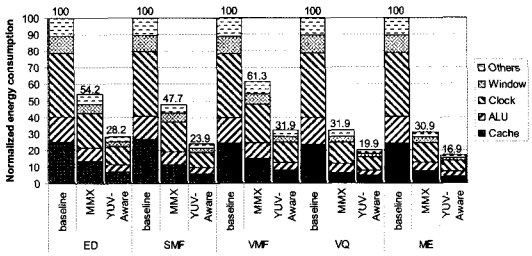


그림 9 소모 에너지에서 YUV 인식 명령어의 효과

그램의 에너지 소모 분포를 보여준다. 각 바(bar)는 에너지 소모를 캐쉬(cache), ALU, 클럭(clock), 윈도우(window), 그리고 나머지(branch prediction, rename, load-store queue, result bus) 하드웨어로 구분한다. 동일한 클럭 주파수, 공정, 프로세서 인자에서 프로그램의 수행시간은 에너지 소모에 비례한다[12]. 예상한 대로, YUV 인식 명령어 기반 프로그램은 baseline 프로그램과 비교하여 상당한 양의 에너지를 감소시킨다. YUV 인식 명령어 기반 프로그램은 baseline 프로그램에 비해 68%(VMF)~83%(ME)의 에너지를 감소시킨 반면, MMX 기반 프로그램은 baseline 프로그램에 비해 단지 39%(VMF)~69%(ME) 에너지를 감소시키는 결과를 보인다. 특히, YUV 인식 명령어는 상당한 수의 ALU, branch, 그리고 캐쉬 접근 수를 줄이기 때문에 적은 양의 에너지가 speculative execution과 캐쉬 접근 유닛에서 소모된다.

5. 결 론

본 논문에서 고성능, 저전력 임베디드 비디오 프로세서에 적합한 새로운 YUV 인식 명령어를 제안하였다. 멀티미디어 성능 향상을 위해 고성능 범용 마이크로프로세서에서 채택한 멀티미디어 전용 명령어(MMX, SSE, VIS, and AltiVec)는 일반적인 서브워드 병렬기법(subword parallelism)을 이용하는 반면, YUV 인식 명령어는 두 쌍의 16-bit YUV 데이타를 32-bit 레지스터에 저장하고 동시에 처리함으로써 성능을 향상시키는 동시에 휴먼 지각 YUV 데이타를 효율적으로 처리할 수 있다. 제안한 YUV 인식 명령어를 4-way 슈퍼스칼라 프로세서에 구현하여 모의 실험한 결과, YUV 인식 명령어 기반 프로그램은 baseline 프로그램보다 3~5.8배(평균 3.9배) 성능 향상을 보인 반면, MMX 기반 프로그램은 baseline 프로그램보다 단지 1.6~3.2배(평균 2.1배) 성능 향상을 보였다. 또한, YUV은 MMX보다 소모 에너지 감소 측면에서도 우위를 보였다. YUV 인식 명령어 기반 프로그램은 baseline 프로그램에 비해 68%~83%의 에너지를 감소시킨 반면, MMX기반 프로그램은 baseline 프로그램보다 단지 39%~69%의 에너지를 감소시키는 결과를 보였다. 그러므로, 제안하는 YUV 인식 명령어가 칼라 이미지/비디오 데이타를 다루는 다양한 임베디드 시스템에 적용될 경우 상당한 성능 향상 및 소모 에너지 감소가 기대된다.

참 고 문 헌

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

[2] 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.

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

[4] 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.

[5] 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.

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

[7] 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.

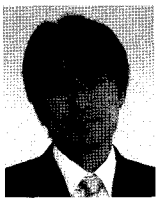
[8] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd Ed., Prentice Hall, 2002.

[9] 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.

[10] D. Burger, T. M. Austin, and S. Bennett, "Evaluating future micro-processors: the SimpleScalar tool set," Tech. Report TR-1308, Univ. of Wisconsin-Madison Computer Sciences Dept., 1997.

[11] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. of the IEEE Intl. Symp. on Computer Architecture*, pp. 83-94, June 2000.

[12] 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 전임연구원. 2007년~현재 울산대학교 컴퓨터정보통신공학부 교수 관심분야는 임베디드 SoC 설계, 컴퓨터구조, Application-Specific 프로세서 설계, 병렬처리