Implementing and Evaluating Color-Aware Instruction Set for Low-Memory, Embedded Video Processing in Data Parallel Architectures*

Jongmyon Kim[1], D. Scott Wills[2], and Linda M. Wills[2]

[1] Chip Solution Center, Samsung Advanced Institute of Technology,
San 14-1, Nongseo-ri, Kiheung-eup, Kyungki-do, 449-712, South Korea
jongmyon.kim@samsung.com
[2] School of Electrical and Computer Engineering,
Georgia Institute of Technology, Atlanta, Georgia 30332-0250
{scott.wills, linda.wills}@ece.gatech.edu

Abstract. Future embedded imaging applications will be more demanding processing performance while requiring the same low cost and low energy consumption. This paper presents and evaluates a color-aware instruction set extension (CAX) for single instruction, multiple data (SIMD) processor arrays to meet the computational requirements and cost goals. CAX supports parallel operations on two-packed 16-bit (6:5:5) YCbCr data in a 32-bit datapath processor, providing greater concurrency and efficiency for color image and video processing. Unlike typical multimedia extensions (e.g., MMX, VIS, and MDMX), CAX harnesses parallelism within the human perceptual color space rather than depending solely on generic subword parallelism. 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 array architecture show that CAX achieves a speedup ranging from 5.2× to 8.8× (an average of 6.3×) over the baseline SIMD array performance. This is in contrast to MDMX (a representative MIPS multimedia extension), which achieves a speedup ranging from 3× to 5× (an average of 3.7×) over the same baseline SIMD array. CAX also outperforms MDMX in both area efficiency (a 52% increase versus a 13% increase) and energy efficiency (a 50% increase versus an 11% increase), resulting in better component utilization and sustainable battery life. Furthermore, CAX improves the performance and efficiency with a mere 3% increase in the system area and a 5% increase in the system power, while MDMX requires a 14% increase in the system area and a 16% increase in the system power. These results demonstrate that CAX is a suitable candidate for application-specific embedded multimedia systems.

1 Introduction

As multimedia revolutionizes our society, its applications are becoming some of the most dominant computing workloads. Color image and video processing in particular has garnered considerable interest over the past few years since color features are valuable in sensing the environment, recognizing objects, and conveying crucial

* This work was performed by authors at the Georgia Institute of Technology (Atlanta, GA).

information [10]. These applications, however, demand tremendous computational and I/O throughput. Moreover, increasing user demand for multimedia-over-wireless capabilities on embedded systems places additional constraints on power, size, and weight.

Single instruction, multiple data (SIMD) architectures have demonstrated the potential to meet the computational requirements and cost goals by employing thousands of inexpensive processing elements (PEs) and distributing and co-locating PEs with the data I/O to minimize storage and data communication requirements. The SIMD Pixel (SIMPil) processor [2, 4], for example, is a low memory, monolithically integrated SIMD architecture that efficiently exploits massive data parallelism inherent in imaging applications. It reduces data movement through a processing-in-place technique in which image data are directly transported into the PEs and stored there. Two-dimensional SIMD arrays, including SIMPil, are well suited for many imaging tasks that require processing of pixel data with respect to either nearest-neighbor or other 2-D patterns exhibiting locality or regularity. However, they are less amenable to vector (multichannel) processing in which each pixel computation is performed simultaneously on 3-D YCbCr (luminance-chrominance) channels [1], which are widely used in the image and video processing community. More specifically, since the 3-D vector computation is performed within innermost loops, its performance does not scale with larger PE arrays.

This paper presents a color-aware instruction set extension (CAX) for such SIMD arrays as a solution to this performance limitation by supporting two-packed 16-bit (6:5:5) YCbCr data in a 32-bit register, while processing these color data in parallel. (CAX was introduced previously for superscalar processors [7], but this paper is investigating its use in SIMD image processing architectures.) The YCbCr space allows coding schemes that exploit the properties of human vision by truncating some of the less important data in every color pixel and allocating fewer bits to the high-frequency chrominance components that are perceptually less significant. Thus, it provides satisfactory image quality in a compact 16-bit color representation that consists of a six-bit luminance (Y) and two five-bit chrominance (Cr and Cb) components [6]. In addition, CAX offers greater concurrency with minimal hardware modification. Fig. 1 shows an example of how a 32-bit ALU functional unit can be used to perform either a 32-bit baseline ALU or two 6:5:5-bit ALUs. The 32-bit ALU is divided into two six-bit ALUs and four five-bit ALUs. When the output carry (Cout) is blocked (i.e., Cin = 0), the six smaller ALUs can be performed in parallel.

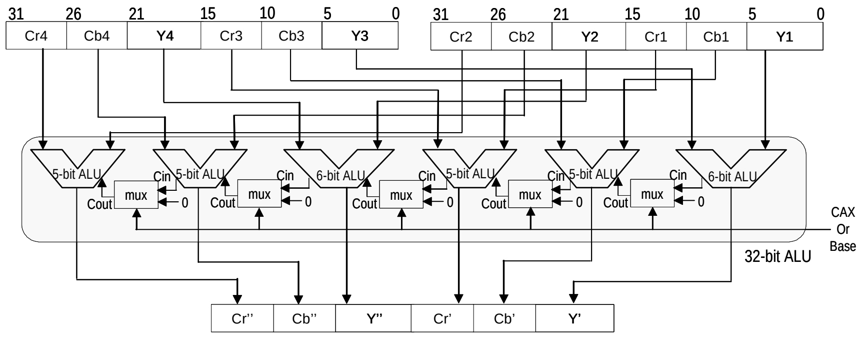

Fig. 1. An example of a partitioned ALU functional unit that exploits color subword parallelism

This paper evaluates CAX in comparison to a representative multimedia extension, MIPS MDMX [11], in a specified SIMD array architecture. MDMX was chosen as a basis of comparison because it provides an effective way of dealing with reduction operations, using a wide packed accumulator that successively accumulates the results produced by operations on multimedia vector registers. Other multimedia extensions (e.g., Intel MMX [9] and Sun VIS [14]) provide more limited support of vector processing in a 32-bit datapath processor without accumulators. To handle vector processing on a 64-bit or 128-bit datapath, they require frequent packing/unpacking of operand data, deteriorating their performance.

This evaluation shows that CAX outperforms MDMX in performance and efficiency metrics on the same SIMD array because CAX benefits from greater concurrency and reduced pixel word storage which can consume a large percentage of silicon area. In particular, the key findings are the following. MDMX achieves a speedup ranging from 3× to 5× (an average of 3.7×) over the baseline performance. However, MDMX requires a 14% increase in the system area and a 16% increase in the system power. As a result, MDMX improves energy efficiency from only 2% to 24% and area efficiency from 6% to 22% over the baseline. On the other hand, CAX achieves a speedup ranging from 5.2× to 8.8× (an average of 6.3×) over the baseline performance because of greater subword parallelism. Moreover, the higher performance is achieved with a mere 3% increase in the system area and a 5% increase in the system power. As a result, CAX improves area efficiency from 36% to 68% and energy efficiency from 35% to 77% over the baseline. These results demonstrate that CAX provides an efficient mechanism for embedded imaging systems.

The rest of the paper is organized as follows. Section 2 presents a summary of the CAX instruction set. Section 3 describes the modeled architectures and a methodology infrastructure for the evaluation of CAX. Section 4 evaluates the system area and power of our modeled architectures, and Section 5 analyzes execution performance and efficiency for each case. Section 6 concludes this paper.

2 Color-Aware Instruction Set for Color Imaging Applications

The color-aware instruction set (CAX) efficiently eliminates the computational burden of vector processing by supporting parallel operations on two-packed 16-bit (6:5:5) YCbCr data in a 32-bit datapath processor. In addition, CAX employs a 128-bit color-packed accumulator that provides a solution to overflow and other issues caused by packing data as tightly as possible by implicit width promotion and adequate space. Fig. 2 illustrates three types of operations: (1) a baseline 32-bit operation, (2) a 4 × 8-bit SIMD operation used in many general-purpose processors, and (3) a 2 × 16-bit CAX operation employing heterogeneous (non-uniform) subword parallelism.

For color images, the band data may be interleaved (e.g., the red, green, and blue data of each pixel are adjacent in memory) or separated (e.g., the red data for adjacent pixels are adjacent in memory). Although the band separated format is the most convenient for SIMD processing, a significant amount of overhead for data alignment is expected prior to SIMD processing. Moreover, traditional SIMD data communication operations have trouble with the band data that are not aligned on boundaries that are powers of two (e.g., adjacent pixels from each band are visually spaced three bytes apart) [12]. Even if the SIMD multimedia extensions store the pixel information in the band-interleaved format (i.e., |Unused|R|G|B| in a 32-bit register), subword parallelism cannot be exploited on the operand of the unused field. Furthermore, since the RGB space does not model the perceptual attributes of human vision well, the RGB to YCbCr conversion is required prior to color image processing. Although the SIMD multimedia extensions can handle the color conversion process in software, the hardware approach would be more efficient.

CAX solves problems inherent to packed RGB extensions by properly aligning two-packed 16-bit data on 32-bit boundaries and by directly supporting YCbCr data processing, providing greater concurrency and efficiency for processing color image sequences. The CAX instructions are classified into four different groups: (1) parallel arithmetic and logical instructions, (2) parallel compare instructions, (3) permute instructions, and (4) special-purpose instructions.

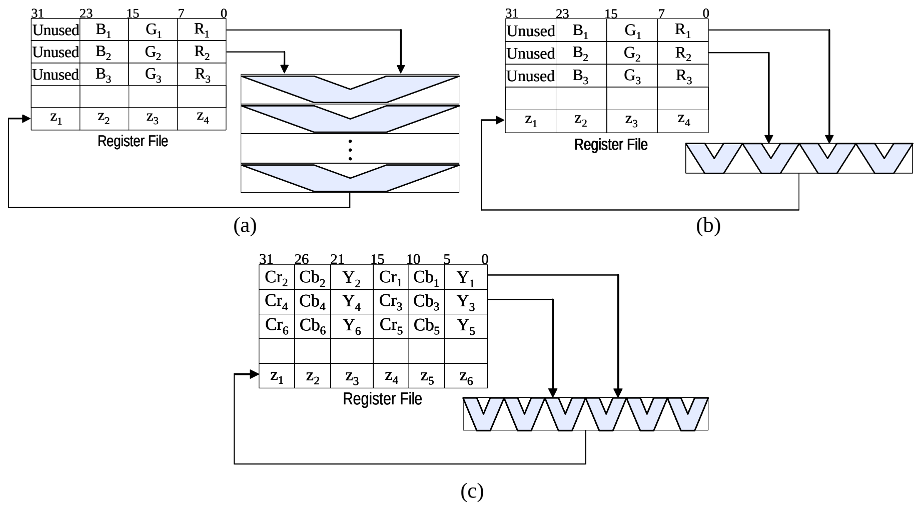

Fig. 2. Types of operations: (a) a baseline 32-bit operation, (b) a 32-bit SIMD operation, and (c) a 32-bit CAX operation

2.1 Parallel Arithmetic and Logical Instructions

Parallel arithmetic and logical instructions include packed versions of addition (`ADD_CRCBY`), subtraction (`SUBTRACT_CRCBY`), and average (`AVERAGE_CRCBY`). The addition and subtraction instructions include a saturation operation that clamps the output result to the largest or smallest value for the given data type when an overflow occurs. Saturating arithmetic is particularly useful in pixel-related operations, for example, to prevent a black pixel from becoming white if an overflow occurs. The parallel average instruction, which is useful for blending algorithms, takes two packed data types as input, adds corresponding data quantities, and divides each result by two while placing the result in the corresponding data location. The rounding is performed to ensure precision over repeated average instructions.

2.2 Parallel Compare Instructions

Parallel compare instructions include `CMPEQ_CRCBY`, `CMPNE_CRCBY`, `CMPGE_CRCBY`, `CMPGT_CRCBY`, `CMPLE_CRCBY`, `CMPLT_CRCBY`, `CMOV_CRCBY` (conditional move), `MIN_CRCBY`, and `MAX_CRCBY`. These instructions compare pairs of sub-elements (e.g., Y, Cb, and Cr) in the two source registers. Depending on the instructions, the results are varied for each sub-element comparison. The first seven instructions are useful for a condition query performed on the incoming data such as chroma-keying [9]. The last two instructions, `MIN_CRCBY` and `MAX_CRCBY`, are especially useful for median filtering, which compare pairs of sub-elements in the two source registers while outputting the minimum and maximum values to the target register.

2.3 Parallel Permute Instructions

Permute instructions include `MIX_CRCBY` and `ROTATE_CRCBY`. These instructions are used to rearrange the order of quantities in the packed data type. The mix instruction mixes the sub-elements of the two source registers into the operands of the target register, and the rotate instruction rotates the sub-elements to the right by an immediate value. These instructions are useful for performing a vector pixel transposition or a matrix transposition [13].

2.4 Special-Purpose Instructions

Special-purpose CAX instructions include `ADACC_CRCBY` (absolute-differences-accumulate), `MACC_CRCBY` (multiply-accumulate), `RAC` (read accumulator), and `ZACC` (zero accumulator), which provide the most computational benefits of all the CAX instructions. The `ADACC_CRCBY` instruction, for example, is frequently used in a number of algorithms for motion estimation. The `MACC_CRCBY` instruction is useful in DSP algorithms that involve computing a vector dot-product, such as digital filters and convolutions. The last two instructions `RAC` and `ZACC` are related to the managing of the CAX accumulator.

3 Methodology

This section describes modeled architectures and a methodology infrastructure for the evaluation of the CAX instruction set.

3.1 Modeled Architectures

The SIMD Pixel (SIMPil) processor is used as the baseline SIMD image processing architecture for this study. Fig. 3 shows the microarchitecture of the SIMD array, along with its interconnection network. When data are distributed, the processing elements (PEs) execute a set of instructions in a lockstep fashion. With 4×4 pixel sensor sub-arrays, each PE is associated with a specific portion (4×4 pixels or 16 pixel-per-processing-element) of an image frame, allowing streaming pixel data to be retrieved and processed locally. Each PE has a reduced instruction set computer (RISC) datapath with the following minimum characteristics:

- Small amount of local storage (128 32-bit words),
- Three-ported general-purpose registers (16 32-bit words),

- ALU – computes basic arithmetic and logic operations,
- Barrel shifter – performs multi-bit logic/arithmetic shift operations,
- MACC – multiplies 32-bit values and accumulates into a 64-bit accumulator,
- Sleep – activates or deactivates a PE based on local information,
- Pixel unit – samples pixel data from the local image sensor array,
- ADC unit – converts light intensities into digital values,
- RGB2YCC and YCC2RGB unit– converts RGB to/from YCbCr, and
- Nearest neighbor communications through a NEWS (north-east-west-south) network and serial I/O unit.

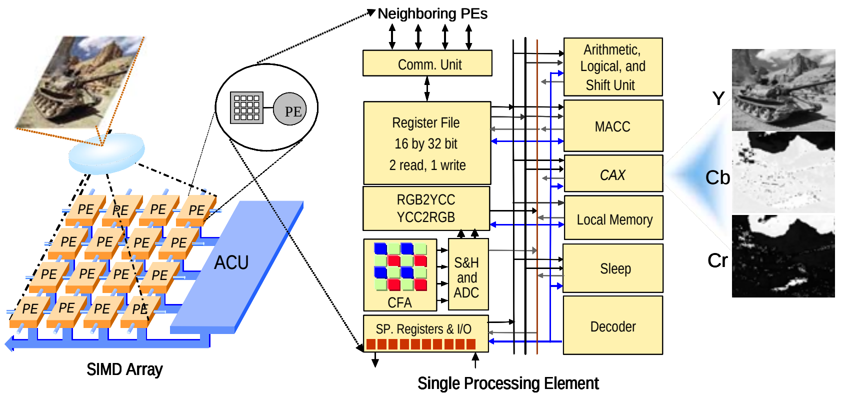

Fig. 3. Block diagram of a SIMD array and a processing element

To improve the performance of vector processing of color image sequences, CAX instructions are included in the instruction set architecture (ISA) of the SIMPil array. For a performance comparison, MDMX-type instructions are also included in the SIMPil ISA. Table 1 summarizes the parameters of the modeled architectures. An overall simulation infrastructure is presented next.

Table 1. Modeled architecture parameters

Parameter	Value
System Size	44×38 (1,584 PEs)
Image Sensor per PE (vertor 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
MDMX/CAX: intALU/intMACC	1 / 1
Local Memory Size (baseline/MDMX/CAX)	128 32-bit/ 128 32-bit/ 64 32-bit word

3.2 Methodology Infrastructure

Fig. 4 shows a methodology infrastructure that is divided into three levels: application, architecture, and technology.

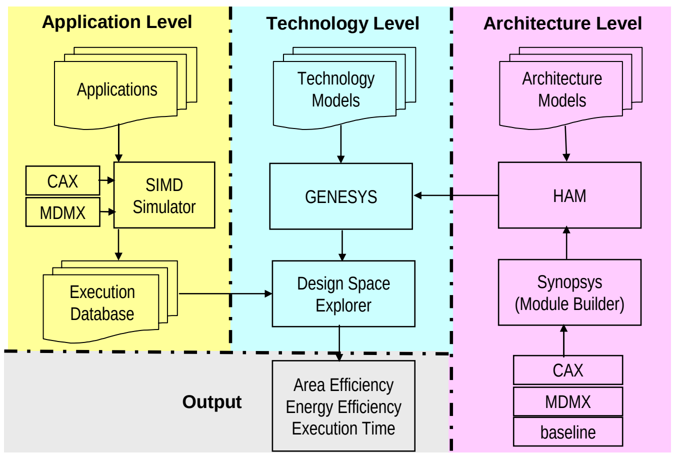

Fig. 4. A methodology infrastructure for exploring the design space of three modeled architectures: baseline SIMPil, MDMX-SIMPil, and CAX-SIMPil

At the application level, an instruction-level SIMD simulator, called SIMPilSim, has been used to profile execution statistics, such as cycle count, dynamic instruction frequency, and PE utilization, for the three different versions of the programs: (1) baseline ISA without subword parallelism (SIMPil), (2) baseline plus MDMX ISA (MDMX-SIMPil), and (3) baseline plus CAX ISA (CAX-SIMPil). The benchmark suite includes five imaging applications (see more details at [5]): a chroma-keying program (CHROMA), color edge detection using a vector Sobel operator (VSobel), the vector median filter (VMF), vector quantization (VQ), and the full-search vector block-matching algorithm of motion estimation (FSVBMA) within the MPEG standard.

At the architecture level, the heterogeneous architectural modeling (HAM) of functional units for SIMD arrays, proposed by Chai *et al.* [3], has been used to calculate the design parameters of modeled architectures. For the design parameters of the MDMX and CAX functional units (FUs), Verilog models for the baseline, MDMX, and CAX FUs were implemented and synthesized with the Synopsys design compiler (DC) using a 0.18-micron standard cell library. The reported area specifications of the MDMX and CAX FUs were then normalized to the baseline FU, and the normalized numbers were applied to the HAM tool for determining the design parameters of MDMX- and CAX-SIMPil. The design parameters are then passed to the technology level.

At the technology level, the Generic System Simulator (GENESYS) developed at Georgia Tech [8] has been used to calculate technology parameters (e.g., latency, area, power, and clock frequency) for each configuration. Finally, the databases (e.g., cycle times, instruction latencies, instruction counts, area, and power of the functional units) obtained from the application, architecture, and technology levels are combined to determine execution times, area efficiency, and energy efficiency for each case. The next section presents the system area and power of the modeled architectures.

4 System Area and Power Evaluation Using Technology Modeling

Fig. 5 shows the system area and power of MDMX-SIMPil and CAX-SIMPil, normalized to the baseline SIMPil. Experimental results indicate that MDMX requires a 14% increase in the entire system area and a 16% increase in the peak system power. However, CAX only requires a 3% increase in the system area and a 5% increase in the system area and power because of the reduced pixel word storage (local memory). These system area and power results are combined with application simulations for determining processing performance, area efficiency, and energy efficiency for each case, which is presented next.

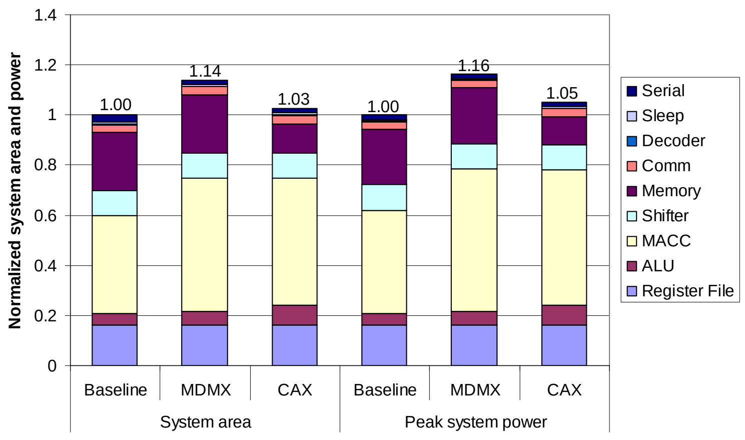

Fig. 5. System area and power of CAX-SIMPil and MDMX-SIMPil, normalized to the baseline SIMPil

5 Experimental Results

Cycle accurate simulation and technology modeling have been used to determine the performance and efficiency characteristics of modeled architectures for each application task. Each application was developed in its respective assembly languages for the SIMPil system, in which all three versions for each program have the same parameters, data sets, and calling sequences. In the experiment, the overhead of the color conversion was not included in the performance evaluation for all the versions. In other words, this study assumes that the baseline, MDMX, and CAX versions directly support YCbCr data in the band-interleaved format (e.g., |Unused|Cr|Cb|Y| for baseline and MDMX and |Cr|Cb|Y|Cr|Cb|Y| for CAX). The metrics of the execution cycle count, corresponding sustained throughput, energy efficiency, and area efficiency of each case form the basis of the study comparison, defined in Table 2.

Table 2. Summary of evaluation metrics

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

C is the cycle count, f_{ck} is the clock frequency, O_{exec} is the number of executed operations, U is the system utilization, and N_{PE} is the number of processing elements. Note that since each CAX and MDMX instruction executes more operations (typically six and three times, respectively) than a baseline instruction, we assume that each CAX, MDMX, and baseline instruction executes six, three, and one operation, respectively, for the sustained throughput calculation.

5.1 Performance Evaluation Results

This section evaluates the impact of CAX on processing performance for the selected color imaging applications on the SIMPil system.

Overall Results. Fig. 6 illustrates execution performance (speedups in executed cycles) attained by CAX and MDMX when compared with the baseline performance without subword parallelism. The results indicate that CAX outperforms MDMX for all the programs in terms of speedup, indicating a speedup ranging from 5.2× to 8.8× (an average of 6.3×) with CAX, but only 3× to 5× (an average of 3.7×) with MDMX over the baseline. The next section discusses the sources for the reductions in the issued instructions are discussed next.

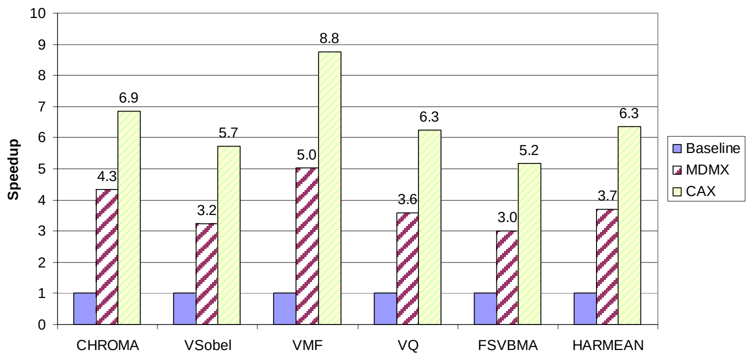

Fig. 6. Speedups for the SIMPil system with CAX and MDMX, normalized to the baseline performance. Note that HARMEAN is the harmonic mean

Benefits of CAX for Color Imaging Applications. Fig. 7 shows the distribution of issued vector instructions for the SIMPil system with CAX and MDMX, normalized to the baseline version. Each bar divides the instructions into the arithmetic-logic-unit (ALU), memory (MEM), communication (COMM), PE activity control unit (MASK), image pixel loading (PIXEL), MDMX, and CAX. The use of CAX reduces a significant number of the instruction counts for all of the programs, ranging from 88.6% (VMF) to 80.7% (FSVBMA) over the baseline. In particular, CAX reduces a significant number of ALU and memory instruction counts due to its instruction definition. An interesting observation is that the FSVBMA program has the smallest reduction in the instruction count with CAX. This is because it involves high inter-PE communication operations that are not affected by CAX. For example, each PE cannot directly support a macroblock size of 16×16 pixels because 4×4 pixels are mapped to each PE. As a result, the 4×4 distortions are computed in each PE separately. Each result is then combined through NEWS communication instructions for the final distortion between the 16×16 input and reference blocks.

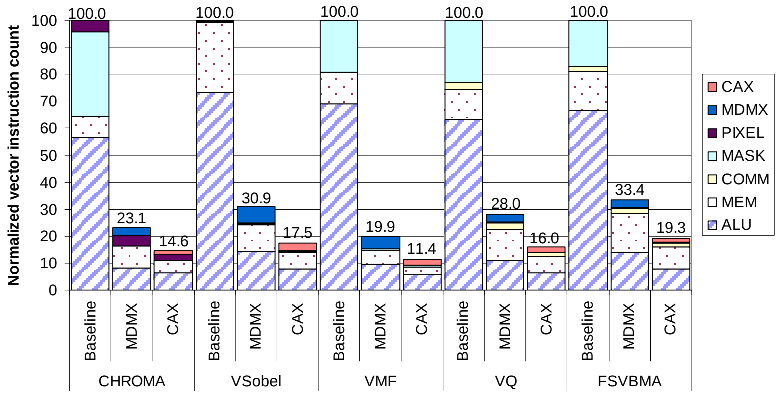

Fig. 7. The distribution of issued vector instructions for the SIMPil system with CAX and MDMX, normalized to the baseline version

5.2 Energy Evaluation Results

Fig. 8 shows energy efficiency, the task throughput achieved per unit of Joule, for the SIMPil system with MDMX and CAX, normalized to the baseline version. CAX outperforms MDMX across all the programs in the energy efficiency, indicating a 50% increase with CAX, but only an 11% increase with MDMX. This is because CAX achieves higher sustained throughputs with a smaller increase in the system power. Increasing energy efficiency improves sustainable battery life for given system capabilities.

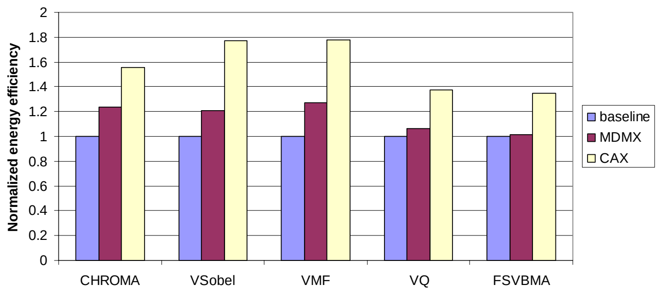

Fig. 8. Energy efficiency for the SIMPil system with CAX and MDMX, normalized to the baseline version

5.3 Area Evaluation Results

Fig. 9 shows area efficiency, the task throughput achieved per unit of area, for the SIMPil system with MDMX and CAX, normalized to the baseline version. As with energy efficiency, CAX outperforms MDMX for all the programs in the area efficiency, indicating a 52% increase with CAX, but only a 13% increase with MDMX. This is because CAX achieves higher sustained throughput with smaller area overhead. Increasing area efficiency improves component utilization for given system capabilities.

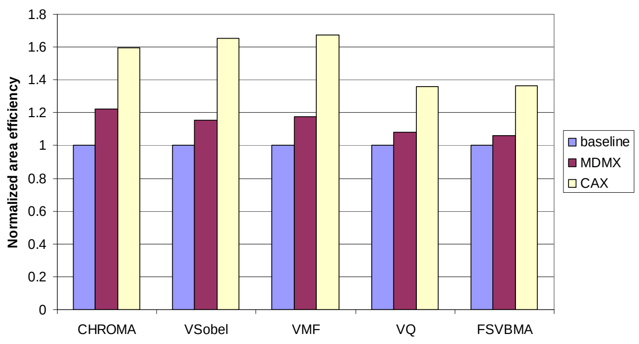

Fig. 9. Area efficiency for the SIMPil system with CAX and MDMX, normalized to the baseline version

6 Conclusions

As emerging portable multimedia applications demand more and more tremendous computational throughput with limited area and power, the need for high efficiency, high throughput embedded processing is becoming an important challenge in computer architecture. In this regard, this paper has addressed application-, architecture-, and technology-level issues in an existing processing system to efficiently support vector processing of color image sequences. In particular, this paper has focused on the color-aware instruction set (CAX) for memory- and performance-hungry embedded applications in a representative SIMD image processing architecture. Unlike typical multimedia extensions, CAX harnesses parallelism within the human perceptual color space (e.g., YCbCr). Rather than depending solely on generic subword parallelism, CAX supports parallel operations on two-packed 16-bit YCbCr data in a 32-bit datapath processor, providing greater concurrency and efficiency for color image and video processing. The key findings are as follows:

- CAX achieves a speedup ranging from 5.2× to 8.8× (an average of 6.3×) over the baseline SIMD array performance without subword parallelism. This is in contrast to MDMX, which achieves a speedup ranging from only 3× to 5× (an average of 3.7×) over the same baseline SIMD array.
- CAX reduces energy consumption from 80% to 89%, but MDMX reduces energy consumption from only 60% to 79% over the baseline version.
- Moreover, CAX benefits from reduced pixel word storage in addition to greater

concurrency. As a result, CAX outperforms MDMX for all the programs in area efficiency and energy efficiency. The area efficiency increases from 36% to 68% (an average of 52%) with CAX, but only 6% to 22% (an average of 13%) with MDMX. The energy efficiency increases from 35% to 77% (an average of 50%) with CAX, but only 2% to 24% (an average of 11%) with MDMX. Increasing area and energy efficiencies yield greater component utilization and sustainable battery life, respectively, for given system capabilities.

- Furthermore, CAX improves the performance and efficiency with a mere 3% increase in the silicon area and a 5% increase in the system power, while MDMX requires a 14% increase in the silicon area and a 16% increase in the system power.

In the future, a heuristic compiler support will be explored that extracts both data-level parallelism and color subword parallelism from high level language programs to overcome tedious hand optimization and/or special programming libraries.

References

1. V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards: Algorithms and Architectures, Kluwer Academic Publishers (1997)
2. H. H. Cat, A. Gentile, J. C. Eble, M. Lee, O. Verdier, Y. J. Joo, D. S. Wills, M. Brooke, N. M. Jokerst, A. S. Brown, and R. Leavitt, SIMPil: An OE integrated SIMD architecture for focal plane processing applications, in Proc. Massively Parallel Processing Using Optical Interconnection (MPPOI-96), pp. 44-52 (1996)
3. 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 (2000)
4. A. Gentile and D. S. Wills, Portable Video Supercomputing, IEEE Trans. on Computers, vol. 53, no. 8, pp. 960-973 (2004)
5. J. Kim, Architectural enhancements for color image and video processing on embedded systems. PhD dissertation, Georgia Inst. of Technology (2005)
6. 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 (2005)
7. J. Kim and D. S. Wills, Efficient processing of color image sequences using a color-aware instruction set on mobile systems, in Proc. of the IEEE Intl. Conf. on Application-Specific Systems, Architectures, and Processors, pp. 137-149 (2004)
8. S. Nugent, D. S. Wills, and J. D. Meindl, A hierarchical block-based modeling methodology for SoC in GENESYS, in Proc. of the 15th Ann. IEEE Intl. ASIC/SOC Conf., pp. 239-243 (2002)
9. A. Peleg and U. Weiser, MMX technology extension to the Intel architecture, IEEE Micro, vol.16, no. 4, pp. 42-50 (1996).
10. K. N. Plataniotis and A. N. Venetsanopoulos, Color Image Processing and Applications (2000)
11. MIPS extension for digital media with 3D. Technical Report http://www.mips.com, MIPS technologies, Inc. (1997)
12. N. Slingerland and A. J. Smith, Measuring the performance of multimedia instruction sets, IEEE Trans. on Computers, vol. 51, no. 11, pp. 1317-1332 (2002)
13. 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 (2002)
14. 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 (1996)