

QUANTIZED COLOR INSTRUCTION SET FOR MEDIA-ON-DEMAND APPLICATIONS

Jongmyon Kim and D. Scott Wills

Microelectronics Research Center
Electrical and Computer Engineering, Georgia Institute of Technology
Atlanta, Georgia 30332-0250
{jmkim and scott.wills}@ece.gatech.edu

ABSTRACT

This paper presents Quantized Color Pack eXtension (QCPX) ISA to accelerate performance of pixel-oriented media processing applications. The QCPX ISA (with a 32 bit word size) supports two packed, quantized (reduced) 16-bit color pixels represented in a YCbCr (Y: luminance, Cr and Cb: chrominance) color format. Unlike typical multimedia instruction set extensions (e.g., MDMX, MMX, ALTIVEC), QCPX obtains substantial performance and code density improvements through implicit support for color pixel processing rather than depending solely on generic subword parallelism. To fully measure its impact, QCPX is evaluated in the context of a massively data-parallel SIMD execution platform where data parallelism is harnessed by an orthogonal mechanism. Simulation results indicate that the 32-bit QCPX ISA achieves an overall average speedup of 584% over the non-QCPX and 88% over the 32-bit MDMX-like ISA with four media applications in a same machine platform. In addition, QCPX results in a higher system utilization in excess of 95% due to a significant reduction of conditional instructions.

1. INTRODUCTION

Processing color images and video media demands tremendous computational and I/O throughput [1]. General-purpose processor manufacturers have incorporated multimedia extensions to improve the performance of certain multimedia applications. These extensions consist of SIMD-like (single instruction, multiple data) instructions that handle packed-data (subword parallelism). Among these multimedia extensions are Intel's MMX™ [2] and SSE™ [3], Hewlett Packard's MAX2 for the PA-RISC architecture [4], Sun Microsystems' VIS for SPARC [5], MIPS's MDMX [6], Alpha's MVI [7], and Motorola's ALTIVEC for PowerPC™ architecture [8]. These new instructions exploit data parallelism and low-precision data formats (e.g., 8-bit pixels) for media processing algorithms. Subword level parallelism (SLP) [4] instructions store a color pixel as a packed 32-bit word containing an 8-bit

red, blue, green and alpha field. While subword parallelism can be exploited on the RGB components, it is not exploited on the alpha value [2]. Also, although RGB representations are widely employed, the RGB tristimuli are highly correlated and therefore not well-suited for independent coding [9].

In this paper, we propose and evaluate a novel ISA, Quantized Color Pack eXtension (QCPX), which not only eliminates RGB limitations by employing the quantized 16-bit color pixels in commonly used YCbCr color space [9] but also achieves homogeneous (i.e., uniform) subword parallelism. A 16-bit color pixel representation includes an 8-bit luminance (Y) and two 4-bit chrominance (Cr and Cb). Since a luminance chrominance space (e.g. YCbCr) allows coding schemes to exploit the properties of human vision by allocating significantly less bits to the high-frequency chrominance components that are perceptually less significant, chrominance space could be implemented using shorter fields (e.g., Cb and Cr fields are 4 bits rather than 8) while providing satisfactory image quality. Figure 1 shows a normal 32-bit operation, 4x 8-bit SIMD operation used in many general-purpose processors, and 2x 16-bit QCPX operation employed heterogeneous (non-uniform) subword parallelism.

QCPX functionality differs from other multimedia extensions by including a specific color data representation supported in hardware. This eliminates the overhead of using SIMD instructions (misalignment-related instructions and packing/unpacking data related instructions) to support color operations. It is orthogonal to packed word-data-parallel-execution techniques (or subword parallelism) that can be exploited separately. In addition, a 32-bit integer functional unit allows two sets of the 16-bit (a 8-bit and 2x 4-bit) integer unit or a 32-bit integer unit to be performed in a single cycle at the same datapaths, the overhead cost is very small. This paper evaluates the QCPX ISA along with several applications that highlight some of its features and applications execution performance on 4,096 nodes array SIMPIL system [14,17].

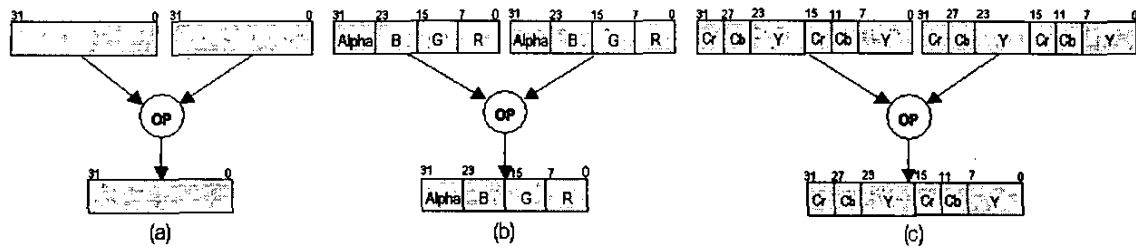


Figure 1. Type of operations. (a) Normal 32-bit operation. (b) 32-bit SIMD operation. (c) 32-bit QCPX operation.

2. QCPX INSTRUCTIONS

In QCPX, a standard unit of storage, a word, is partitioned into two set of the 16-bit (a 8-bit and 2x 4bit) or a 16-bit and 2x 8-bit that provides a very low-cost form using integer functional units granularity. The QCPX instruction set can be classified into four major groups based on the functional unit such as ALU instructions, memory instructions, compare instructions, and special-purpose instructions. A summary of the QCPX ISA is presented below.

2.1. Color-pack-arithmetic and logical instructions

Color-pack-arithmetic and logical instructions support ADD_CRCBY (signed, unsigned saturation, and modulo), SUBTRACT_CRCBY (signed, unsigned saturation, and modulo), MULTIPLY_CRCBY, BCAST_CRCBY (broadcast), AVERAGE_CRCBY, and SHIFT_CRCBY [left|right] operations, which are the most frequent operations in many color image and video computations. Figure 2 (a) and (b) show ADD_CRCBY and BCAST_CRCBY instructions, respectively.

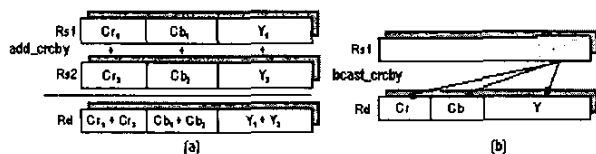


Figure 2. Operations of QCPX. (a) Color-pack-add instruction.(b) Broadcast instruction.

2.2. Memory instructions

Memory QCPX instructions support partial load and store (Y and CbCr) operations. Some applications utilize a luminance (Y) component while preserving chromaticity (Cb and Cr) components, and vice versa [10,11,12]. Therefore, blocked loads and stores transfer packed 16-bit YCbCr of data between memory and a partitioned QCPX registers without causing allocations.

2.3. Color-pack-compare instructions

Color-pack-compare instructions support color pixels comparison to produce masks that can be used to reduce branches. In some cases, there is a need to select different data based on a condition query performed on the incoming data [2]. Figure 3 shows two conditional select instructions. The CMPSELGE_CRCBY instruction is defined as compare and select the bigger or equal values of Y and quantized Cb and Cr in parallel. Similarly, the CMPSELLT_CRCBY instruction is defined as compare and select the smaller values of Y and quantized Cb and Cr in parallel.

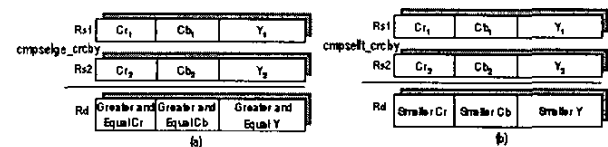


Figure 3. Compare QCPX operations. (a) Compare and select greater and equal instruction. (b) Compare and select smaller instruction.

2.4. Special-purpose instructions

Special-purpose QCPX instructions include MACC_CRCBY (multiply-accumulate) and ADACC_CRCBY (absolute-distance-accumulate) instructions that provide the most computational benefit of all QCPX instructions. Figure 4 shows an absolute-distance-accumulate instruction. Each of the ADACC_CRCBY instruction calls saves its sum in a partitioned 2x32-bit (where a 32-bit consists of a 8-bit Cr's sum, a 8-bit Cb's sum, and a 16-bit Y's sum) special purpose register until the ZACC (zero-accumulate) instruction is called. A 64-bit special-purpose register is included to avoid the result of width problems to act as the destination register. This mechanism supports the extra space required for the result to be stored in full precision.

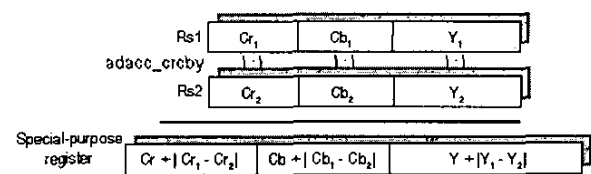


Figure 4. Absolute-distance-accumulate instruction.

3. APPLICATIONS AND PERFORMANCE RESULTS

To evaluate performance of QCPX, four widely studied media benchmarks have been implemented on a 4,096 PEs SIMD array simulator [13,14] in three different configurations: (1) a baseline RISC 32-bit ISA without subword parallelism, (2) the baseline RISC plus the 32-bit MDMX-like ISA [6], and (3) the baseline RISC plus the 32-bit QCPX. The four media benchmarks (edge detection, scalar median filtering, vector median filtering, and vector quantization) span a range of key tasks in color image processing workloads. All the color image benchmarks were run with 256x256 pixel 3-band (i.e., channel) input images obtained from Intel Media Benchmark.

In all cases, the applications are hand-coded and optimized in their respective assembly languages. For each application, the worst-case scenario was used. Both execution time (t_{exec}) and sustained throughput (Th_{sust}) are computed with reference to the 500 MHz target platform as follows:

$$t_{exec} = \frac{C}{f_{ck}}, \quad \text{and} \quad Th_{sust} = \frac{IC \cdot U \cdot N_{PE}}{t_{exec}},$$

where f_{ck} is the clock frequency, C is the cycle count for a given application, IC is number of instructions issued during the application (i.e., the executed PE instruction count), and N_{PE} is the number of processing elements in the PE array. Performance results are shown in Table 1 with and without QCPX as well as MDMX-like ISA.

3.1. Edge detection

Edge detection is a problem of fundamental importance in image analysis such as segmentation, registration, and identification of objects in a scene [11]. One of basic edge detection's techniques, Laplacian edge detector, uses a simple 3x3 convolution mask to create a series of 2-D spatial gradient magnitudes [15].

The most time critical operation in this application is multiplication accumulation between color pixels and Laplacian mask in 3x3 window. Using the `BCAST_CRCBY` instruction, each of mask values is distributed in Cr, Cb, and Y positions for future multiplications with color pixels in parallel. Then, the `MACC_CRCBY` instruction accumulates its sum in a partitioned 64-bit special-purpose register.

The QCPX version is 408% faster than the non-QCPX version and 108% faster than the MDMX-like version.

3.2. Scalar median filter

The scalar median filter (SMF) is an image restoration technique that eliminates impulse noise spikes from an image by replacing three-color channels with the median

channels of a small neighborhood surrounding the pixels. For a 3x3 filter, nine pairs are sorted by luminance and chrominance values in parallel, and then each of the fifth entries is selected as the new color pixels [12].

Using `CMPSELGE_CRCBY` and `CMPSELLT_CRCBY` instructions, the nine registers containing two sets of packed YCbCr data are compared and selected bigger and equal (or smaller) values of each color pixels in parallel. Therefore, a bubble-sort algorithm is highly accelerated. In addition to accelerating execution time, QCPX exploits a 100% of system utilization due to the reduction of the large number of conditional instructions.

The QCPX version achieves 554% speedup over the non-QCPX version and 98% over the MDMX-like version.

3.3. Vector median filter

The vector median filter (VMF) is widely used to remove impulse noise for color image because of their ability to reduce impulse noise while preserving edges [16]. The construction of VMF is based on norm (L1 in this paper) to order vectors according to their relative magnitude differences in a 3x3 window.

In this application, using the `ADACC_CRCBY` (absolute-distance-accumulate) instruction, nine pairs (in a 3x3 window) for each of color pixels are compared to one another via the sum of absolute magnitude differences in parallel. This instruction reduces the large number of ALU instructions as well as conditional instructions, while providing high system utilization (95.4%).

The QCPX version exploits an overall speedup of about 1,029% over the non-QCPX version and 100% over the MDMX-like version.

3.4. Real-time vector quantization

Vector Quantization (VQ) has become a popular image and video compression technique [17,18]. Compression is achieved by subdividing the image into small blocks (e.g., 4x4 pixels) and then finding the best match for each block among the available codewords. Clearly, the full-search vector quantizer is computationally very intensive.

Using the `ADACC_CRCBY` instruction, the distortion between the input vector and its corresponding codeword is performed efficiently. The real-time VQ optimized by the QCPX version exploits an overall speedup of about 345% over the non-QCPX version and 80% over the MDMX-like version. Although the method of VQ is similar to VMF described in section 3.3 above such as computation between the 4x4 input block and the local codeword (64 x 64) via the sum of absolute differences measure, a higher speedup as in VMF is not achieved due to the large inter-PE communications that are not affected by QCPX.

Table 1. Applications performance with (and without) the QCPX and MDMX-like ISA.

Applications	ISA Type	Total Cycles	System Utilization (%)	Sustained Throughput (Gops/s)	Execution Time (μsec)
Edge Detection	QCPX (non-QCPX)	865 (4,394)	97.86 (87.2)	2,004 (1,786)	1.7 (9)
	MDMX	1,802	96.98	1,986	4
SMF	QCPX (non-QCPX)	6,203 (40,593)	100 (85.1)	2,048 (1,550)	13 (81)
	MDMX	12,267	100	2,048	22
VMF	QCPX (non-QCPX)	5,814 (65,650)	95.4 (75.7)	1,954 (1,550)	12 (131)
	MDMX	10,462	94.1	1,927	22
VQ	QCPX (non-QCPX)	51,943 (231,241)	96.1 (90.2)	1,968 (1,847)	104 (463)
	MDMX	93,511	95.2	1,950	187

All speedups in together (32-bit QCPX vs. 32-bit baseline RISC-like ISA and 32-bit MDMX-like ISA vs. 32-bit baseline RISC-like ISA) are shown in Figure 5. In each case, the 32-bit QCPX version is much faster than the 32-bit MDMX-like version with the same datapath in a common machine platform.

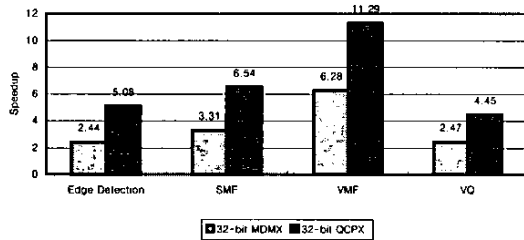


Figure 5. Speedups over the baseline ISA.

4. CONCLUSIONS

This paper examines the impact of the QCPX instruction set for several commonly used media applications on a SIMD pixel processor architecture. The 32-bit QCPX ISA results in a higher code density and execution time compared to that of the 32-bit MDMX-like multimedia extension. In addition, the high system utilization archived by using the QCPX ISA further indicates that the available data bandwidth and parallel execution schemes in the SIMD architecture are suitable for image and video processing applications. In addition, datapath scalability feature of QCPX is being evaluated on a processor with different datapath sizes (from 16-bit to 64-bit words) using the same instruction set without changing the ISA. This will allow more flexibility in balancing performance versus cost and power. Ongoing research will address QCPX effectiveness on other color image and video applications with more widely used ILP (pipelined, superscalar) processor architectures.

REFERENCES

[1] K. Diefendorff and R. Dubey, "How multimedia workloads will change processor design," *IEEE Computer*, vol. 30, no. 9, pp. 43-45, 1997.

[2] A. Peleg, S. Wilkie, and U. Weiser, "Intel MMX for multimedia PCs," *Communications of the ACM*, vol. 40, no. 1, pp. 25-38, 1997.

[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. 51-59,1996.

[5] M. Tremlay, J. M. O'Connor, V. Narayanan, and L. He, "VIS speeds new media processing," *IEEE Micro*, vol. 16, no. 4, pp. 10-20, 1996.

[6] MIPS extension for digital media with 3D. Technical Report <http://www.mips.com>, MIPS technologies, Inc., 1997.

[7] R. Sites, Ed. *Alpha Reference Manual*, Bulington, MA: Digital, 1992.

[8] H. Nguyen and L. John, "Exploiting SIMD parallelism in DSP and multimedia algorithms using the AltiVec technology," *In ICS'99*, pp. 11-20, 1999.

[9] G. Sharma, M. J. Vrhel, and H. J. Trussell, "Color imaging for multimedia," *Proc. of the IEEE*, vol. 86, no. 6, pp. 1088-1108, 1998.

[10] J. L. H. Webb, "Post processing to reduce blocking artifacts for low bit-rate video coding using chrominance information," *Proc. IEEE Conf. on Image Processing*, vol. 2, pp. 9-12, 1996.

[11] G. A. Baxes, *Digital image processing*, John Wiley & Sons, Inc., 1994.

[12] J. M. Kim, S. Ryu, A. Gentile, L. M. Wills, and D. S. Wills, "Impulse noise removal on an embedded, low memory SIMD processor," *Proc. of the 14th IEEE Intl. Conf. on DSP, DSP 2002*, pp. 1257-1260, July 2002.

[13] SIMPil Home Page, http://www.ccc.gatech.edu/research/pica/past_projects/simpil

[14] Huy H. Cat et al., "SIMPil: An OE integrated SIMD architecture for focal plane processing applications," *IEEE Proceedings of MPPOI '96*, pp.44-52, 1996.

[15] M. A. Ruzon and C. Tomasi, "Color edge detection with the compass operator," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 160-166, 1999.

[16] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proc. of the IEEE*, vol. 78, no. 4, pp. 678-689, 1990.

[17] A. Gentile, H. Cat, F. Kossentini, F. Sorbello, and D. S. Wills, "Real-time vector quantization-based image compression on the SIMPil low memory SIMD architecture," *IEEE Intl. Performance, Computing, and Communications Conference (IPCCC-97)*, pp. 10-16, 1997.

[18] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*: Kluwer Academic Press, 1992.