AVX2–OPTIMIZED KVAZAAR HEVC INTRA ENCODER

Ari Lemmetti, Ari Koivula, Marko Viitanen, Jarno Vanne, Timo D. Hämäläinen

Department of Pervasive Computing, Tampere University of Technology, Finland
{ari.lemmetti, ari.koivula, marko.viitanen, jarno.vanne, timo.d.hamalainen}@tut.fi

ABSTRACT

This paper presents efficient SIMD optimizations for the open-source Kvazaar HEVC intra encoder. The C implementation of Kvazaar is accelerated by Intel AVX2 instructions whose effect on Kvazaar ultrafast preset is profiled. According to our profiling results, C functions of SATD, DCT, quantization, and intra prediction account for over 60% of the total intra coding time of Kvazaar ultrafast preset. This work shows that optimizing primarily these functions doubles the coding speed of a single-threaded Kvazaar intra encoder for the same rate-distortion performance. The highest performance boost is obtained by deploying the proposed optimizations jointly with multithreading. On the Intel 8-core i7 processor, the AVX2-optimized 16-threaded Kvazaar ultrafast preset achieves real-time (30 fps) intra coding speed up to 1080p resolution. Compared to AVX2-optimized ultrafast preset of x265, Kvazaar is 20% times faster and still obtains 9.1% bit rate gain for the same quality. These results justify that Kvazaar is currently the leading open-source HEVC intra encoder in terms of real-time coding speed and efficiency.

Index Terms— High Efficiency Video Coding (HEVC), Kvazaar, intra coding, single instruction multiple data (SIMD), Advanced Vector Extensions 2 (AVX2)

1. INTRODUCTION

High Efficiency Video Coding (*HEVC*) [1], [2] is the latest international standard in video coding. It has been developed by *Joint Collaborative Team on Video Coding* (*JCT-VC*) as a joint activity of ITU-T *Video Coding Experts Group* (*VCEG*) and ISO/IEC *Moving Picture Experts Group* (*MPEG*). HEVC is published as twin text by ITU, ISO, and IEC as ITU-T H.265 | ISO/IEC 23008-2. Its first edition contains Main, Main Still Picture, and Main 10 profiles.

This paper addresses *all-intra* (*AI*) coding [3] configuration of HEVC Main Profile. It is reported to improve coding performance by 23% over that of AVC intra coding [4] for the same objective quality but at a cost of over 3× encoding complexity [5]. Therefore, implementing a real-time HEVC intra encoder with a reasonable cost and power budget requires efficient optimizations.

In software encoders, the complexity of HEVC can be tackled by two primary techniques: multithreading through data-level parallelization and *single instruction multiple data* (*SIMD*) optimizations. The implementation details of these optimizations are kept strictly confidential in commercial HEVC encoders, so only open-source implementations are considered in this paper.

Currently, there exist three noteworthy open-source HEVC encoders: HM [6], x265 [7], and our Kvazaar [8]. The SIMD and data-level optimizations presented for HEVC typically deal with HM decoder [9], [10] but there also exists a couple of works that focus on HM encoder optimization. Chen et al. [11] speed up individual algorithms of HM 6.2 encoder with *Streaming SIMD Extensions 4* (*SSE4*) instruction set. Ahn et al. [12] accelerate inter coding of HM 9.0 encoder by 1.2× through SSE3 and multithreading. However, the original implementation of HM is inherently slow so the reported optimizations are not still sufficient for real-time HEVC encoding. Hence, x265 and Kvazaar are the only practical open-source HEVC encoders at the moment.

Both x265 and Kvazaar support multithreading and SIMD implementations. Kvazaar intra coding performance and parallelization on multi-core processors have already been considered in our previous works [13], [14] so the main emphasis here is on Kvazaar SIMD optimizations. Intra coding in Kvazaar is more time-consuming than inter coding. Optimizing intra coding is crucial for AI coding, but it is also important for other coding configurations that contain intra only pictures. Since our target is real-time coding speed, the main focus is on Kvazaar ultrafast preset whose C implementation is accelerated by *Advanced Vector Extensions 2* (*AVX2*) instructions.

The remainder of this paper is organized as follows. Section 2 presents the intra coding tools of Kvazaar ultrafast preset and the respective complexity profiling results of Kvazaar C implementation. Section 3 introduces the proposed SIMD optimization techniques for Kvazaar. Section 4 measures the performance gain of the proposed optimizations and benchmarks the *rate-distortion-complexity* (*RDC*) characteristics of the optimized Kvazaar over those of x265. Section 5 concludes the paper.

2. KVAZAAR INTRA ENCODER

Kvazaar intra encoder supports HEVC Main profile for 8-bit 4:2:0 video with ten recently specified presets: *ultrafast*, *superfast*, *veryfast*, *faster*, *fast*, *medium*, *slow*, *slower*,

veryslow, and *placebo* whose naming convention follows that of x264 [15]. In this work, the ultrafast preset is considered for the highest possible coding speed.

2.1. Ultrafast preset

Kvazaar ultrafast preset reduces encoding complexity by disabling or limiting configurable encoder features. This inevitably induces some loss in RD performance. The intra coding settings affecting the RDC characteristics of Kvazaar the most include a simplified selection of *intra prediction* (*IP*) mode, limitations of initial prediction block sizes, and a disabled *RD optimized quantization* (*RDOQ*).

Table 1 tabulates the settings of ultrafast presets in Kvazaar and x265. The main difference between them is that Kvazaar makes use of the large 64 × 64 coding unit sizes, while x265 limits the size to 32 × 32. In addition, the deblocking filter is disabled in Kvazaar.

2.2. Profiling results

A single-threaded C implementation of Kvazaar ultrafast preset was profiled with Intel VTune Amplifier XE 2015 using advanced hotspots feature. Our profiling environment is detailed in Table 2. Each sequence in the HEVC common test conditions (classes *A-F*) [16] was encoded with a *quantization parameter* (*QP*) value of 32 using the command line parameters listed in Table 3. Inclusive CPU times were gathered from VTune reports for each test sequence and then averaged. The distribution of relative CPU time between the essential encoding functions is summarized in Fig. 1.

As is shown in Fig. 1, *Sum of Absolute Transformed Differences* (*SATD*), *Discrete Cosine Transform* (*DCT*), angular IP, planar IP, and quantization account for more than half of the CPU cycles in Kvazaar. These functions are also suitable for SIMD acceleration, so they were selected as our primary targets for optimization.

3. KVAZAAR AVX2 OPTIMIZATIONS

The majority of SIMD optimizations for Kvazaar is implemented with AVX2 instruction set extension and only these optimizations are considered in this paper. Like AVX, AVX2 utilizes 256-bit wide registers, called YMM. These instructions also allow for non-destructive operations where the result is written in a register other than one of the inputs. Compared to AVX, the main advantage of AVX2 is the increased number of the instructions supporting the YMM registers.

The chosen method for AVX2 optimization is the usage of Intel Intrinsics instead of a x86-64 assembly language. Intrinsics for AVX2 are usable in C language with the inclusion of corresponding header file. The header defines

Table 1. Encoder settings in ultrafast presets

Feature	Ultrafast presets	
	Kvazaar	x265
Coding Unit sizes	64, 32, 16, 8	32, 16, 8
Prediction Unit sizes	64, 32, 16, 8	32, 16, 8
IP modes	35	35
SAO	off	off
Signhide	off	off
RDOQ	off	off
TU split	off	off
Deblock	off	on

Table 2. Profiling environment

Processor	Intel Core i7-5960X Extreme (8 × 3.0 GHz)
Memory	32 GB
L1 cache	8 × 32 KB (instruction) + 8 × 32 KB (data)
L2 cache	8 × 256 KB
L3 cache	20 MB
Compiler	MS Visual Studio 12.0.30723.00 Update 3
Operating system	64-bit MS Windows 8.1

Table 3. Encoding parameters

Encoder	Parameters
Kvazaar	--wpp --no-info --preset=ultrafast -n (frames) -p 1 -q (qp)
no-asm	--cpuid=0
x265	--tune=psnr --psnr --no-info --preset=ultrafast --no-progress --hash=3 -q (qp) -f (frames) -I1 --ipratio=1 --input-res=(res) --no-scenecut --pools=(threads) --fps=(fps) --log-level=debug
single	--frame-threads=1
no-asm	--no-asm

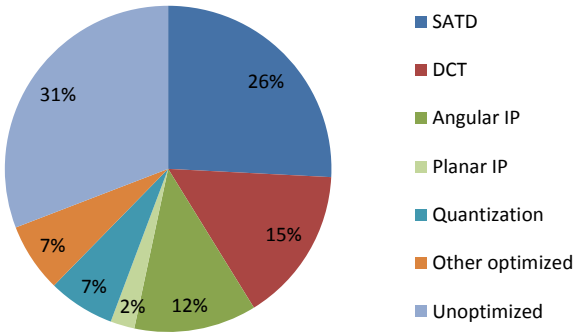

Fig. 1. Relative CPU usage in Kvazaar C implementation

necessary data types such as __m128i and __m256i for 128-bit and 256-bit integer vector operations, respectively. In addition, the header file provides callable functions that are mapped practically one-to-one to AVX2 instructions. Having variables and corresponding tools available, much of the manual labor is avoided and left to the compiler.

In theory, a transition from 128-bit to 256-bit SIMD registers doubles the number of parallel processed elements. However, the 128-bit wide instructions extended to YMM registers often behave as if the YMM registers were divided into two separate 128-bit registers. These 128-bit wide units are called *lanes*. This needs to be addressed, since loading sequential elements into YMM registers and operating on them can result in an unintuitive result due to in-lane limitations of certain AVX2 instructions. Fig. 2 demonstrates such situation where the division into lanes affects the order of resulting elements.

In some cases, it is challenging to fill all registers with meaningful data, especially when operating on the smallest block sizes of HEVC. A general approach to solve this issue is to eliminate cross-lane dependencies by dedicating each lane for a separate task. This also makes porting 128-bit algorithm implementations very straightforward to YMM registers and even further to the 512-bit wide registers of AVX-512.

At the moment, the implementations of SATD, DCT, quantization, and angular prediction (for blocks larger than 8x8) in Kvazaar utilize the whole 256 bits of the YMM registers at least once, whereas the rest of the optimizations use at most half of the register capacity. Furthermore, most of the functions in Kvazaar are exclusive for 8-bit encoding.

3.1. SATD

The rough intra mode search of Kvazaar approximates the coding costs of two angular intra modes at every iteration. The modes are then compared to the previously found best mode which is updated when a better one is found. The two SATDs are computed simultaneously, dedicating the lower lane for the first mode and the upper lane for the other mode. The SATD criterion is calculated using the fast Walsh—Hadamard transform. The AVX2 optimized transform is performed in multiple steps using shuffle, sign, and packed add operations for horizontal transform. Vertical transforms are achieved with packed additions and subtractions. Each of these instructions has low latency and high throughput on Haswell CPUs.

3.2. DCT

The C implementation of Kvazaar utilizes a partial butterfly method for the transforms. However, a straightforward matrix multiplication is used in AVX2 implementation for the forward and inverse DCT instead. Rows of one matrix are interleaved with unpack operations and multiply-added horizontally with appropriate values from the other matrix that are broadcasted to fill the whole register. The values are accumulated in a buffer which is then shifted and value clipped to the right magnitude to match the HEVC specification. The 2D transform is achieved using the matrix multiplication twice with the correct order of input matrices and predefined values.

	256 bits							
	Lower lane				Upper lane			
YMM A	a0	a1	a2	a3	b0	b1	b2	b3
YMM B	c0	c1	c2	c3	d0	d1	d2	d3
_mm256_hadd_epi32(A, B)	a0 + a1	a2 + a3	c0 +c1	c2 + c3	b0 + b1	b2 + b3	d0 + d1	d2 + d3

Fig. 2. Horizontal addition of 32-bit integers

3.3. Angular and planar prediction

Angular prediction consists of linear filtering functions that generate the prediction block. In the AVX2 optimized function, reference pixels are loaded into two registers with the other one having an offset of one pixel. To linearly filter the values, reference pixels are unpacked and then multiply-added horizontally with a register filled with weights. The calculated pair of weights is packed into a single value that is broadcasted over the register.

Planar prediction is filtered in the vertical direction as well. Regular packed multiplication and addition is performed to filter reference pixels in both directions instead of using horizontal multiply-add. Broadcast instruction is used to set reference pixel values and weights in registers when necessary. As in many functions, packed shift and pack operation is performed to scale the values to get the total average.

3.4. Quantization and dequantization

This function quantizes coefficients in groups of 16. Values are extended to 32 bits and they use two YMM registers. Each coefficient is multiplied with predefined scaling values and rounded with addition and shifting. In the optimized function, flat scaling values are assumed. Dequantization follows the same principle.

3.5. Other optimized functions

Other smaller trivial functions, such as computation of block residuals and reconstruction of blocks from quantized values, have been optimized as well.

4. PERFORMANCE EVALUATION

Fig. 3 depicts the complexity distribution of Kvazaar after our SIMD optimizations. In the original C implementation, the functions selected for SIMD acceleration represent 69 % of total runtime. Now, their respective share is 35%.

The RDC characteristics of Kvazaar v0.8.2.5 and x265 v1.8 intra encoders are further evaluated by benchmarking their AVX2-optimized ultrafast presets on Core i7 processor (Table 2) using the command line parameters listed in Table

3. Release versions of Kvazaar and x265 were compiled with Visual Studio 2013 using the build files included in each project.

4.1 Kvazaar intra coding speed

Table 4 tabulates intra coding speeds in *frames per second* (*fps*) for three Kvazaar implementations: 1-threaded C, 1-threaded C/AVX2, and 16-threaded C/AVX2. The average coding speeds were measured separately for each sequence using the QP values of 22, 27, 32, and 37. For a reliable comparison, only one encoder instance was run at a time. All these versions attain an equivalent RD performance.

The C implementation of Kvazaar is able to encode WQVGA (416 × 240) sequences in real time (30 fps). AVX2 optimizations accelerate the C implementation averagely by 2.0× and parallelization to 16 threads further by 8.7× on the 8-core processor. Hence, their joint performance gain is over 17× enabling Kvazaar to encode 1080p (1920 × 1080) format in real-time.

4.2. Comparison of Kvazaar and x265

Table 4 also compares RDC characteristics of the 16-threaded AVX2-optimized Kvazaar with the respective implementation of x265.

Their bit rate differences were compared in terms of the *Bjøntegaard delta bit rate* (*BD-rate*) [17]. The RD curves for the BD-rate computations were interpolated with the piecewise cubic interpolation [18] through experimentally specified RD points that represent the QP values of 22, 27, 32, and 37. For a fair comparison, 66 bytes per picture were deducted from the bit rate values of x265 to compensate its bug that caused parameter information to be attached to every coded picture.

According to our results, Kvazaar is 1.2× faster than x265 on average and is still able to obtain 9.1% average bit rate gain. Reasons for better RDC characteristics are described in [14], with SIMD capabilities disabled. The RD performance of Kvazaar is unaffected by the proposed optimizations while they accelerate the intra coding speed of Kvazaar over that of AVX2-optimized x265. Against HM 16.0, the average bit rate of Kvazaar ultrafast preset is 32% larger, but the SIMD optimized version is also 52 times faster with a single thread, or 475 times faster with 16 threads.

5. CONCLUSIONS

This paper presented AVX2 optimizations for our Kvazaar open-source HEVC encoder which is originally written in C. Based on our profiling results, the most complex functions include SATD, DCT, quantization, and intra prediction. Accelerating them with AVX2 doubles the speed of Kvazaar

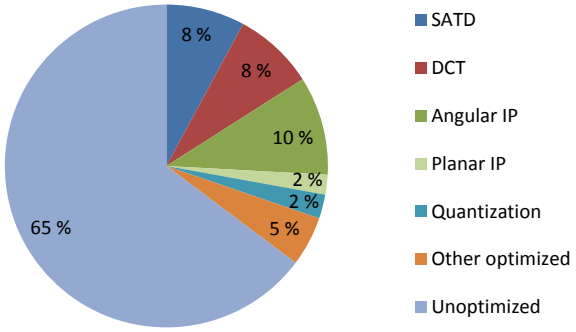

Fig. 3. Relative CPU usage of SIMD optimized Kvazaar

Table 4. Coding speed of Kvazaar and comparison to x265

Sequence	Kvazaar Coding Speed (fps)			Kvazaar vs. x265	
	C 1 thread	C/AVX2 1 thread	C/AVX2 16 threads	Speedup	BD-rate
Traffic	0.8	1.8	16.6	1.2×	1.9%
PeopleOnStreet	0.8	1.8	16.8	1.2×	-1.0%
Kimono	1.7	4.1	36.9	1.2×	8.0%
ParkScene	1.6	3.4	31.2	1.1×	1.5%
Cactus	1.6	3.5	32.2	1.2×	-2.5%
BQTerrace	1.7	3.4	31.7	1.2×	-7.5%
BasketballDrive	1.7	3.9	35.6	1.2×	-1.9%
RaceHorses	7.7	15.7	143.5	1.1×	-6.9%
BQMall	8.4	17.5	160.2	1.2×	-9.9%
PartyScene	7.1	12.8	121.4	1.1×	-9.0%
BasketballDrill	8.4	17.4	157.7	1.2×	-9.7%
RaceHorses	30.4	58.9	504.2	1.1×	-14.3%
BQSquare	28.9	51.2	450.2	1.1×	-16.4%
BlowingBubbles	28.1	50.0	438.6	1.1×	-11.3%
BasketballPass	31.9	65.4	512.5	1.0×	-18.5%
FourPeople	4.1	9.5	86.5	1.3×	-2.9%
Johnny	4.6	11.1	99.9	1.3×	-5.3%
KristenAndSara	4.4	10.6	95.8	1.3×	-8.1%
BasketballDrillText	8.2	16.4	150.0	1.2×	-14.5%
ChinaSpeed	4.2	8.7	81.2	1.2×	-22.4%
SlideEditing	3.6	6.4	60.4	1.2×	-25.1%
SlideShow	5.9	13.9	115.5	1.4×	-25.0%
Average				1.2×	-9.1%

and together with multithreading, the performance of Kvazaar is increased by over 17× making real-time 1080p intra encoding possible. Our results show that Kvazaar is currently the leading open-source real-time intra encoder. Compared to its closest competitor x265, Kvazaar improves BD-rate by 9.1% and is still around 20% faster.

6. ACKNOWLEDGMENT

This work was supported in part by the European Celtic-Plus Project 4KREPROSYS. The authors would particularly like to thank all contributors of our Kvazaar open-source project [8].

7. REFERENCES

[1] *High Efficiency Video Coding*, document ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC), ITU-T and ISO/IEC, Apr. 2013.

[2] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1649-1668.

[3] J. Lainema, F. Bossen, W. J. Han, J. Min, and K. Ugur, "Intra Coding of the HEVC Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1792-1801, Dec. 2012.

[4] *Advanced Video Coding for Generic Audiovisual Services*, document ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC, Mar. 2009.

[5] J. Vanne, M. Viitanen, T. D. Hämäläinen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1885-1898.

[6] *Joint Collaborative Team on Video Coding Reference Software, ver. HM 16.0* [Online]. Available: http://hevc.hhi.fraunhofer.de/

[7] *x265* [Online]. Available: http://x265.org/

[8] *Kvazaar HEVC encoder* [Online]. Available: https://github.com/ultravideo/kvazaar

[9] L. Yan, Y. Duan, J. Sun, and Z. Guo, "Implementation of HEVC decoder on x86 processors with SIMD optimization," in *Proc. IEEE Visual Communications and Image Processing*, San Diego, CA, USA, Nov. 2012.

[10] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel scalability and efficiency of HEVC parallelization approaches," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, Dec. 2012, pp. 1827-1838.

[11] K. Chen, Y. Duan., L. Sun, and Z. Guo, "Efficient SIMD optimization of HEVC encoder over x86 processors," in *Proc. APSIPA Annual Summit Conf.*, Hollywood, CA, USA, Dec. 2012.

[12] Y. J. Ahn, T. J. Hwang, D. G. Sim, and W. J. Han, "Implementation of fast HEVC encoder based on SIMD and data-level parallelism," *EURASIP J. Image Video Process.*, vol. 16, Dec. 2014, pp. 1-19.

[13] M. Viitanen, A. Koivula, A. Lemmetti, J. Vanne, and T. D. Hämäläinen, "Kvazaar HEVC encoder for efficient intra coding," *in Proc. IEEE Int. Symp. Circuits Syst.*, Lisbon, Portugal, May 2015.

[14] A. Koivula, M. Viitanen, J. Vanne, T. D. Hämäläinen, and L. Fasnacht, "Parallelization of Kvazaar HEVC intra encoder for multi-core processors," *in Proc. IEEE Workshop Signal Process. Syst.*, Hangzhou, China, Oct. 2015.

[15] *x264* [Online]. Available: http://www.videolan.org/developers/x264.html

[16] F. Bossen, "Common test conditions and software reference configurations," *Document JCTVC-J1100*, Stockholm, Sweden, Jul. 2012.

[17] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," *Document VCEG-M33*, Austin, TX, USA, Apr. 2001, pp. 1-4.

[18] J. Wang, X. Yu, and D. He, "On BD-rate calculation," *Document JCTVC-F270*, Torino, Italy, Jul. 2011.