HEVC Real-time Decoding

Benjamin Bross[a], Mauricio Alvarez-Mesa[a,b], Valeri George[a], Chi-Ching Chi[a,b], Tobias Mayer[a], Ben Juurlink[b], and Thomas Schierl[a]

[a]Image Processing Department, Fraunhofer Institute for Telecommunications –
Heinrich Hertz Institute (HHI), Einsteinufer 37, 10587 Berlin, Germany;
[b]Embedded Systems Architecture Group – Technical University of Berlin, Einsteinufer 17,
10587 Berlin, Germany

ABSTRACT

The new High Efficiency Video Coding Standard (HEVC) was finalized in January 2013. Compared to its predecessor H.264 / MPEG4-AVC, this new international standard is able to reduce the bitrate by 50% for the same subjective video quality. This paper investigates decoder optimizations that are needed to achieve HEVC real-time software decoding on a mobile processor. It is shown that HEVC real-time decoding up to high definition video is feasible using instruction extensions of the processor while decoding 4K ultra high definition video in real-time requires additional parallel processing. For parallel processing, a picture-level parallel approach has been chosen because it is generic and does not require bitstreams with special indication.

Keywords: Parallel Video Decoding, SIMD Optimizations, Real-time Video Decoding, HEVC, H.265, Video Compression

1. INTRODUCTION

Recent studies analyzed that coded video data is becoming the major part in consumer internet traffic with a predicted share of 90% by 2015.[1] This is supported by mobile devices where increasing screen resolutions enable them to playback high definition (HD) video which is usually streamed or downloaded over mobile networks. Besides that, there are first attempts to broadcast 4K ultra-high definition (UHD) video in TV networks. All these developments are asking for a new, more efficient video codec that is able to reduce the bitrate without sacrificing video quality or, to increase the video resolution without increasing the bitrate. In 2010, the two premier video standardization organizations, the ITU-T Video Coding Expert Group (VCEG) and the ISO/IEC Moving Pictures Expert Group (MPEG), accepted this challenge and established a Joint Collaborative Team on Video Coding (JCT-VC) to develop a new international video coding standard. This standard should be able to reduce the bitrate by 50% for the same quality compared to the state-of-the-art H.264/MPEG4-AVC standard. Three years later, the first edition of the new standard called High Efficiency Video Coding (HEVC) was finalized in January.[2] In April 2013, the ITU-T published the HEVC specification text as Recommendation H.265 while in ISO/IEC, HEVC becomes MPEG-H Part 2 (ISO/IEC 23008-2).

The new HEVC standard provides improvements all over the hybrid video coding design, which is the same basic design already applied in previous video coding standards. A summary of its main features and a general codec design overview is given by Sullivan et al.[3] Ohm et al. analyzed the coding efficiency of HEVC and compared it with previous video coding standards like H.264/MPEG4-AVC and H.262/MPEG2-Video. They report bitrate reductions of 50% for the same subjective quality compared to H.264/MPEG4-AVC.[4] In order to clarify whether this coding efficiency gain comes along with increased complexity, Bossen et. al studied the complexity aspects of HEVC software encoding and decoding. This study concludes that the encoding is much more challenging than the decoding, e.g. encoding one second of a 1080p60 HD video with the HM reference encoder takes longer than an hour.[5] Hence, it is expected that first applications of HEVC will be offline coded video content, e.g. internet video, video on demand and the like. While application in broadcast usually requires hardware decoder chips as is in set-top boxes, people usually watch internet video on their computer where software decoding plays an important role. Therefore, this paper analyzes the decoding performance of an optimized, multi-threaded HEVC software decoder for HD and 4K/UHD video on a current mobile processor.

The rest of the paper is structured as follows. The next section reviews approaches that have been studied already for parallel HEVC decoding. Section 3 presents code optimizations and a picture-level parallel decoding approach and Section 4 reports runtime and profiling results for these techniques for HD and 4K/UHD test sequences. Finally Section 5 concludes this paper and gives a short outlook.

2. TOWARDS HEVC REAL-TIME DECODING

It has been shown that for resolutions up to HD (1920×1080), code optimizations including heavy use of single-instruction multiple-data (SIMD) instructions are sufficient to achieve HEVC real-time software decoding.[5] When it comes to decode UHD video (3840×2160), single threaded execution with code optimization is not enough anymore.

Several approaches to achieve HEVC real-time decoding of UHD video in software have been studied.[6–9] All studies are based on an optimized version of the HEVC test model (HM) reference software decoder[10] because the original HM software was developed as a reference implementation focussing on correctness and completeness. Hence, the HM reference software is fairly slow. For example, when decoding an HEVC bitstream with only intra-coded pictures and a QP value of 27, it takes the HM decoder two minutes to decode ten seconds of a 1080p60 video.[5] The aforementioned modifications of the HM decoder include code optimization and multithreading support, necessary to achieve real-time decoding. All of the studies are making use of the HEVC high-level tools that allow for parallel decoding, namely slices, wavefront parallel processing and tiles. In the first one, Alvarez et al. investigated a wavefront like concept using entropy slices in HM 3.0, which are not supported anymore in the final version of the standard.[6] The following two papers are suggesting a slightly modified version of wavefront parallel processing, called overlapped wavefronts.[7,8] This concept as well as parallel processing with tiles have been integrated in an optimized HM 4.1 decoder. The most recent publication shows results for overlapped wavefronts based on HM 8.0 and further reports speedup due to the use of SIMD code optimizations.[9] Although all these high-level concepts have been proven to provide real-time decoding, the main disadvantage of them is that they put constraints on HEVC bitstreams, i.e. require an explicit signaling, in order to do so. A more generic approach using picture-level is presented in the next section.

3. SIMD OPTIMIZATION AND PICTURE-LEVEL PARALLELISM

As already mentioned in the previous section, performing parallel decoding on a sub-picture granularity requires an indication of this sub-picture granularity in the bitstream, be it on a slice, tile or CTU line level as in wavefronts. Decoding whole pictures in parallel is a generic way to speed up a decoder using parallel decoding. Different from traditional picture-level parallelization, in which only completely independent pictures or slices are decoded in parallel, the employed parallelization strategy allows decoding dependent pictures in parallel by maintaining the dependencies in a more fine-grain manner. The execution of the picture will only stall if a particular reference region has not been decoded yet, allowing any bitstream to speedup independent of the employed referencing scheme. This picture-level parallelism has been integrated in an HEVC software decoder developed from scratch at the Fraunhofer Heinrich Hertz Institute (HHI). The single threaded version of the HHI decoder is already optimized with regard to code structure and SIMD instruction set extensions of x86 processors*.

In general, for all parts of a codec that perform the same operation on a large amount of data, e.g. a block of picture samples, can be sped up by optimizing these parts of the code for SIMD instruction set extensions. For HEVC, interpolation, intra-picture prediction, inverse transformation, de-blocking and memory copy operation were identified to benefit from SIMD optimizations. The sample adaptive offset filter operations are also well suited for SIMD optimizations but this has not been implemented in the current optimized decoder. Currently, processor extensions from SSE2 to AVX are supported. Results of how much the SIMD optimizations speed up the single-threaded decoder are presented in Section 4.2 where the HHI decoder with SIMD optimizations is compared to the HHI decoder without SIMD optimizations (scalar code) and the HM 12.0 reference decoder.

Besides code and SIMD optimization, a major speedup can be achieve by using multiple threads to run decoding operations in parallel. In the aforementioned picture-level parallel decoding approach, each picture to

*in cooperation with the Embedded Systems Architecture Group at the Technical University of Berlin be decoded is assigned one worker thread that performs the decoding. For coding structures where every picture is coded with intra-picture prediction, almost linear speedup can be achieved because and the synchronization overhead between threads is negligible. When inter-picture prediction coding structures are considered, the inter-picture dependencies require more complicated synchronization between the worker threads. This is further investigated in Section 4.3 where the speedup for different numbers of threads in intra-picture and inter-picture prediction coding structures are shown and analyzed.

4. RESULTS

In this section, results for the optimized HEVC software decoder described in Section 3 are presented and discussed. First, the experimental setup is described in Subsection 4.1. In Subsection 4.2, the single threaded performance of the HM reference decoder and the optimized HHI decoder is compared followed by an analysis of the multithreaded execution of the HHI decoder in Subsection 4.3. Finally, profiling results for the optimized HHI decoder are given in Subsection 4.4.

4.1 Experimental setup

The parallel HEVC decoder has been implemented from scratch and optimized with SIMD intrinsics for SSE extensions. Multithreading has been performed using the C++ Boost libraries, which offer a convenient C++ wrapper around platform-dependent threading libraries such as Pthreads. The optimized decoder has support for multiple operation systems such as Linux, Microsoft Windows and Apple OS X, but the performance experiments presented in this paper have been conducted under Linux.

4.1.1 System

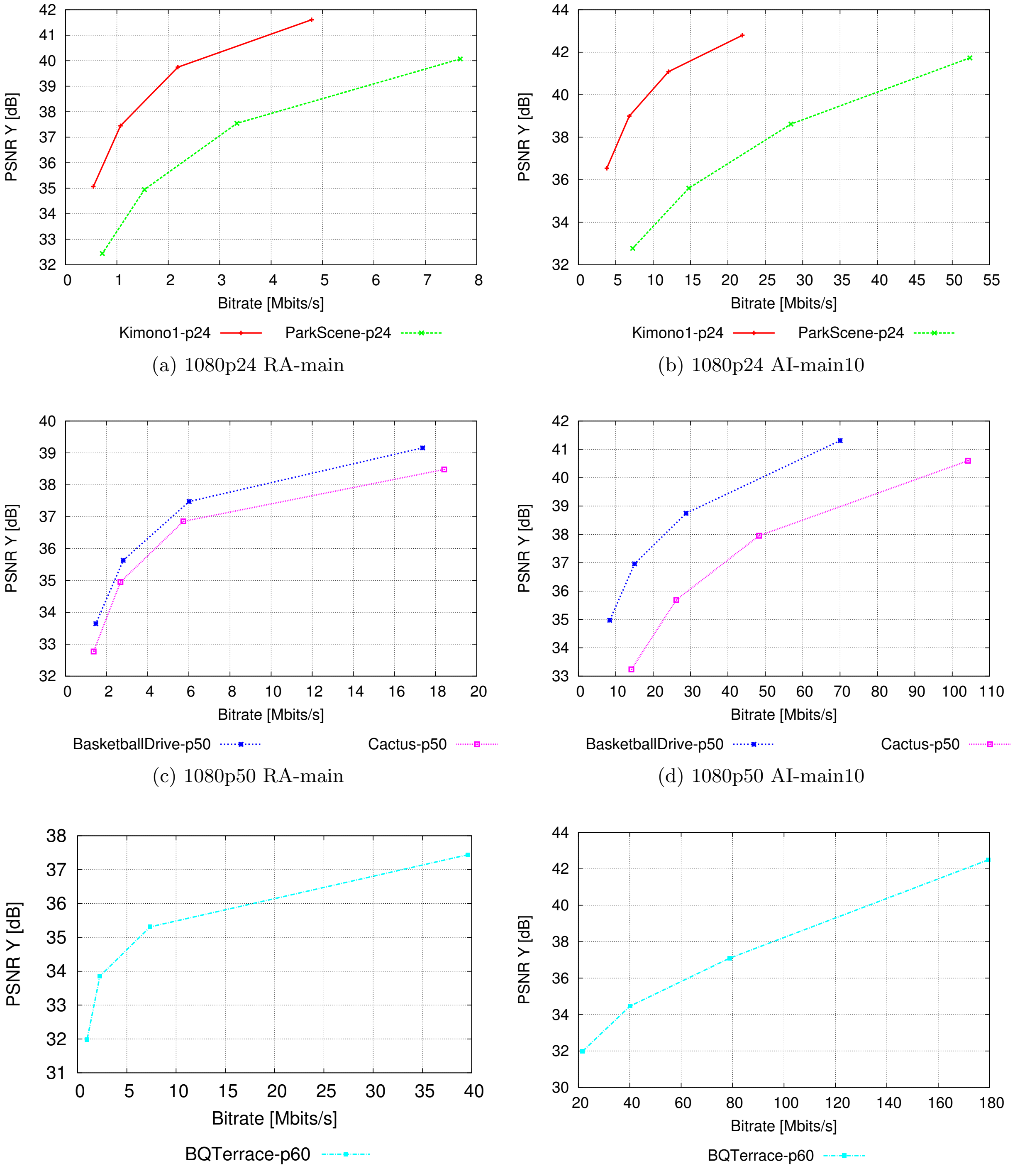
The system employed to measure performance is a Dell mobile workstation with an Intel i7-2920XM processor. This processor is based on the Sandy Bridge microarchitecture and includes four cores running at 2.5 GHz. The simultaneous multithreading feature (SMT, also called Hyperthreading by Intel) provides a total number of eight available hardware threads for the four cores. It has support for SSE (up to version 4.2) and AVX SIMD instructions. Although AVX only includes 256-bit SIMD registers for floating point instructions, integer SIMD instructions still benefit from the three operand mode (non-destructive instruction destination). All details of the hardware/software environment are listed in Table 1.

System		Software	
Processor:	Intel i7-2920XM	HEVC encoder:	HM-12.0
μ-architecture:	Sandy Bridge	HEVC decoder:	HHI-dec-0.29 / HM-12.0
Cores:	4	Boost C++:	1.54.0
Threads/core (SMT):	2	Compiler:	gcc 4.8.1
Frequency:	2.5 GHz	Opt. level:	-O3 -f-no-tree-vectorize
L3-cache:	8 MB	OS:	Linux Ubuntu 13.04
TurboBoost:	disabled	Kernel:	3.8.0-29

Table 1. Experimental setup.

4.1.2 Test Sequences and HEVC Encoding

Test sequences in two resolutions have been used: 1080p which is representative for current high definition systems, and 2160p which is representative for the next generation of high quality video. For 1080p, the five class B sequences from the JCT-VC test set have been used which have 24, 50 and 60 frames per second (fps). For 2160p50, five sequences from the EBU UHD-1 50 fps test set[11] have been selected, namely *Lupo confetti*, *fountain lady*, *rain fruits*, *studio dancer* and *waterfall pan*. All the test sequences have been encoded with the HEVC HM reference encoder version 12.0 using the JCT-VC common test conditions.[12] Encoding options are based on HEVC main and main 10 profiles using two configurations: random access (RA) and all intra (AI). Each video is encoded at four different QP points: 22, 27, 32, 37. The 1080p sequences were encoded with random access main profile and all intra main 10 profiles, while the 2160p sequences were encoded using the random access main 10 profile.



(a) 1080p24 RA-main

(b) 1080p24 AI-main10

(c) 1080p50 RA-main

(d) 1080p50 AI-main10

(e) 1080p60 RA-main

(f) 1080p60 AI-main10

Figure 1. Rate-distortion performance of the HD sequences.

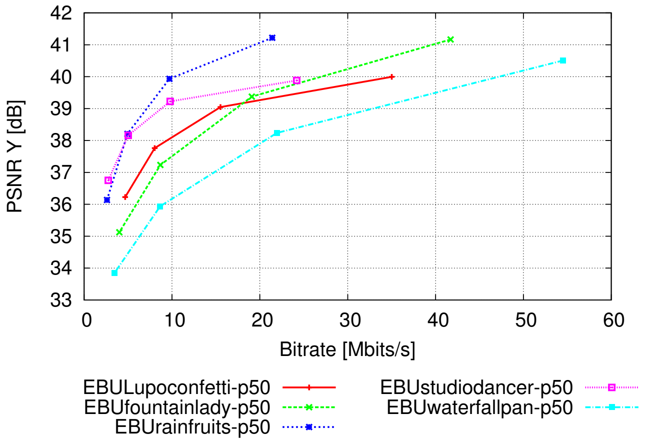


Figure 2. Rate-distortion performance of the UHD-1 sequences for the random access main10 configuration.

4.1.3 Rate-Distortion Performance

Figures 1 and 2 show the resulting rate-distortion performance of the considered HD and UHD-1 test sequences. Here, the peak signal to noise ratio between the original and the reconstructed luma samples (PSNR Y) is used as the distortion measurement. It can be observed that the RD-performance is highly content dependent. For example, an average luma PSNR value of around 41 is measured for the sequence *rain fruits* at 21 Mbits/s while coding *fountain lady* at the same luma PSNR value results in a bitrate of 41 Mbits/s.

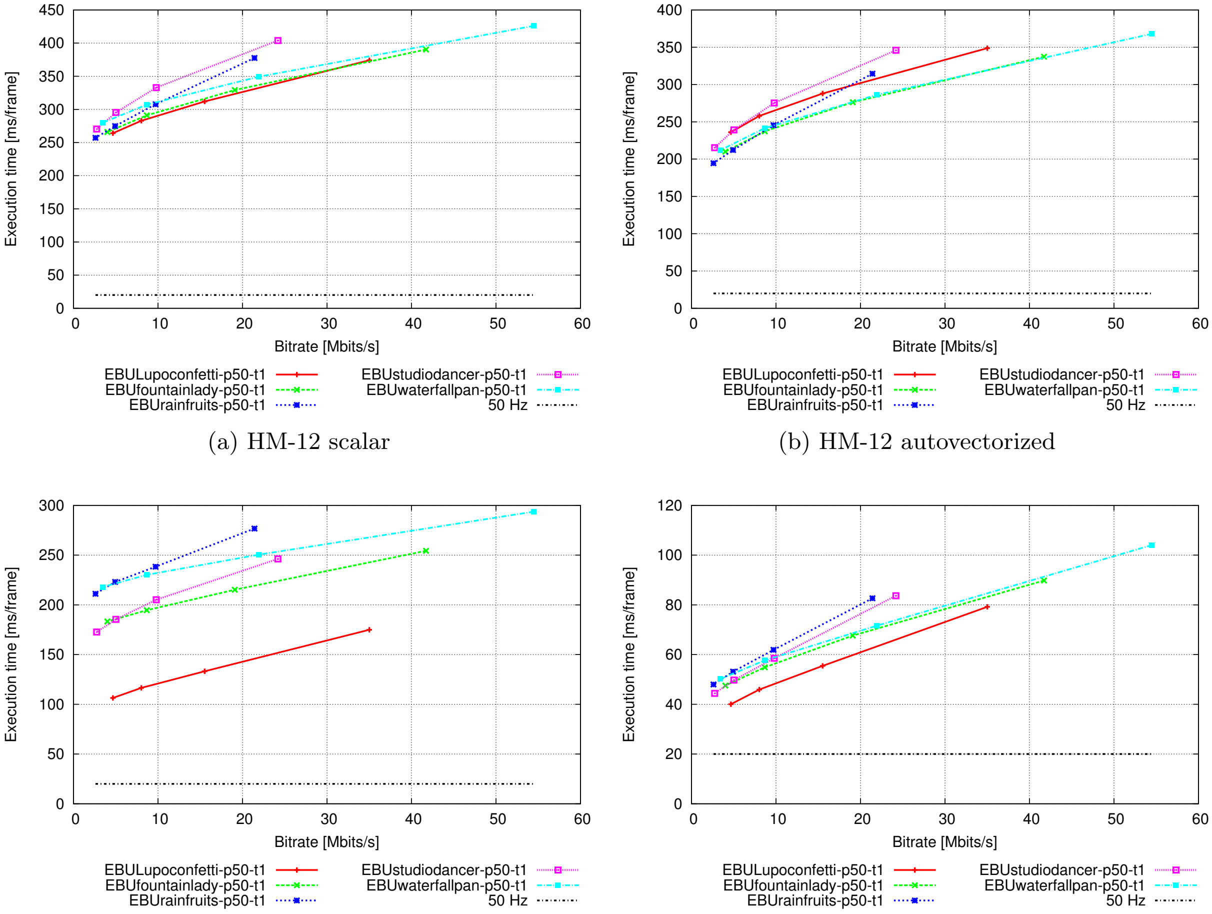
4.2 Comparison with single threaded HM reference decoder

As a first step in the decoding runtime analysis, the optimized HHI decoder is compared with the HM reference decoder. When compiling the HM decoder with default settings and a state-of-the-art compiler, so called auto vectorization already tries to automatically optimize the code for SIMD instructions. In order to perform a fair comparison of both decoders without SIMD optimizations, both have been compiled with the auto vectorization functionality turned off and the SIMD intrinsics have been disabled for the HHI decoder.

Table 2 shows the decoding runtimes in frames per second for all tested resolutions, frame rates, coding configurations and QP values averaged over all sequences for a given resolution, frame rate and QP value. In general, it can be observed that the scalar speedup is rather constant over bitrates (QP values) while the speedup achieved with SIMD optimizations decreases with decreasing QP values (increasing bitrate). This can be explained by the fact that with decreasing QP values, more and larger quantized transform coefficients are to be decoded by the CABAC entropy coding and this part of the code does not benefit from SIMD optimizations.

Enc. cfg.	Res.	HM-12-scalar				HHI-scalar				HHI-simd			
		QP				QP				QP			
		22	27	32	37	22	27	32	37	22	27	32	37
AI main10	1080p24	5.1	6.0	6.9	7.8	12.2	15.9	19.9	26.0	17.6	24.8	33.7	49.2
AI main10	1080p50	4.4	5.7	6.5	7.4	10.7	15.7	20.0	25.6	14.0	23.0	31.9	44.8
AI main10	1080p60	3.5	4.7	5.7	6.4	8.6	13.11	17.3	22.1	10.1	17.3	24.1	32.4
RA main	1080p24	8.8	10.7	12.2	13.6	12.3	14.2	15.8	17.2	39.0	58.5	76.5	94.8
RA main	1080p50	8.5	11.8	13.6	15.0	13.5	18.5	20.9	22.4	32.2	60.8	83.0	101
RA main	1080p60	6.5	11.4	13.9	14.9	9.7	15.3	17.4	17.9	21.3	55.9	89.9	107
RA main10	2160p50	2.5	3.1	3.5	3.7	4.2	5.0	5.6	6.0	11.5	16.00	19.3	21.9

Table 2. Single-threaded performance in frames per second.



(a) HM-12 scalar

(b) HM-12 autovectorized

(c) HHI-dec scalar

(d) HHI-dec SIMD

Figure 3. Avergae time per frame for 2160p50 RA-main10 for the HM reference and the optimized HHI decoder in scalar and SIMD modes (single threaded). Dotted line represents real-time operation.

Detailed results for every sequence of the UHD-1 test set are shown in Figure 3. Here, the execution time per frame is plotted over the bitrate. Since the frame rate of the UHD-1 test sequences is 50 fps, all points that lie under 1000 ms per 50 frames (20 ms/frame) can be considered as to be decoded in real-time. It can be seen that even with SIMD optimizations, single threaded decoding of 4K/UHD video is not possible. Furthermore, the impact of the auto vectorization for the HM decoder is not negligible because runtime reductions of around 50 ms/frame are observed for all rate points.

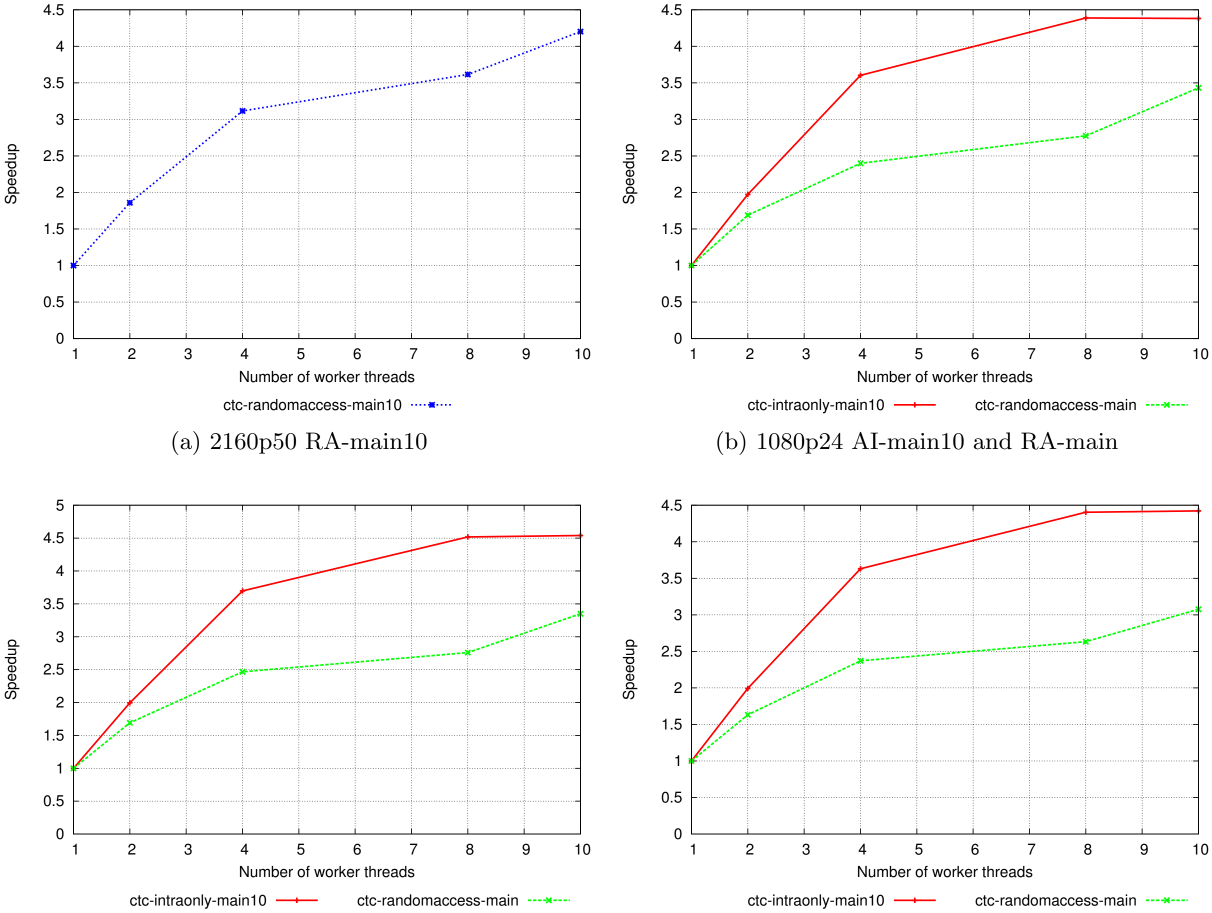
4.3 Multithreaded execution using picture-level parallelism

In a second step, the speedup when using the picture-level parallel decoding approach described in Section 3 is analyzed. In addition to the single threaded execution times presented in Subsection 4.2, the execution times of the HHI decoder with SIMD optimization are measured when two, four, eight and ten threads are used. Figure 4 illustrates the speedup compared to the single threaded execution averaged over all QP values and test sequences for a given resolution, frame rate and coding configuration. For all four subfigures, three different slope segments can be identified.

The first segment ranges from one to four worker threads where the the number of worker threads can be mapped to the number of physical CPU cores, which is four in the system used for the experiments. Here, the all intra configuration provides an almost one-to-one correlation between the number of threads and the speedup factor (speedup of almost 4 for four worker threads) while the random access configuration provides a more flat speedup (speedup between 2.5 and 3 for four worker threads). This can be explained by the fact that inter-picture prediction is used in the random access configuration, which introduces picture-to-picture dependencies. In the random access configuration pictures are encoded using QP cascading. This results in different QP values for different pictures, and, consequently, different execution times. Threads that process pictures which execute normally faster can stall because the required reference areas of depending pictures are not available yet.

The next segment ranges from five to eight worker threads, which corresponds to the number of hardware threads made available by SMT. These hardware threads provide significantly less speedup than a physical CPU core, e.g. only 4.5 for eight worker threads for all intra configurations. This is expected as the SMT threads are sharing the execution core with the ”normal” threads. In this particular processor each core is shared by two threads. The speedup achieved using five to eight threads originate from the additional instruction level parallelism (ILP) exposed by the additional threads, which increases the utilization of the functional units in the core.

In the last segment from nine worker threads on, no additional speedup is achieved for the all intra configuration, which is reasonable since no CPU resources are available anymore to decode a picture. For the random access configuration, however, using more than eight worker threads still provides an additional speedup. Due to the aforementioned inter-thread synchronization for inter-picture prediction, it may occur that one worker thread is idle. In that case, the associated CPU core can be used to start decoding another picture. Therefore, increasing the number of worker threads still provides a speedup for coding configurations using inter-picture prediction.



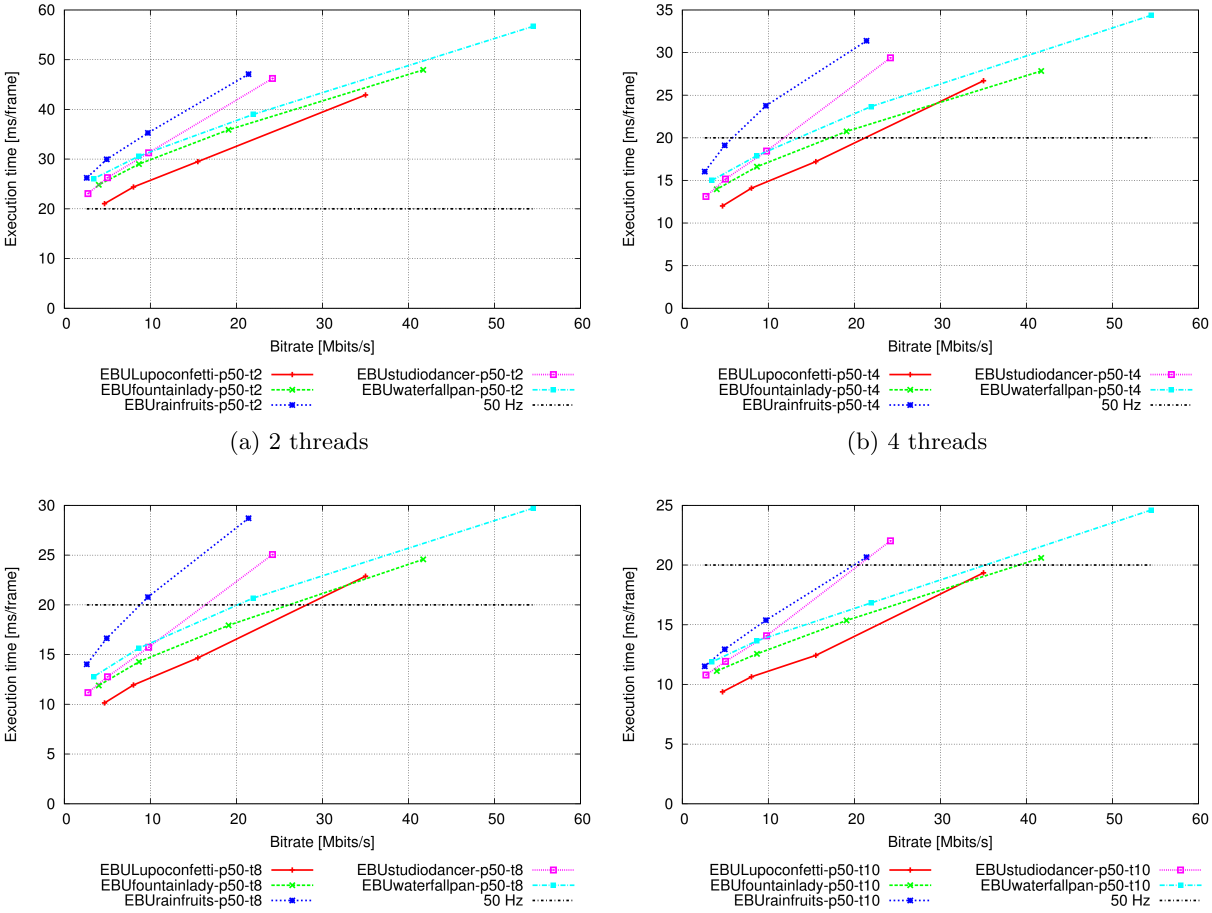
(a) 2160p50 RA-main10

(b) 1080p24 AI-main10 and RA-main

(c) 1080p50 AI-main10 and RA-main

(d) 1080p60 AI-main10 and RA-main

Figure 4. Parallelization speedup.



(a) 2 threads

(b) 4 threads

(c) 8 threads

(d) 10 threads

Figure 5. Average time per frame for 2160p50 RA-main10 for optimized decoder in SIMD modes (multi-threaded).

In order to illustrate what is needed to achieve real-time decoding of 4K/UHD video, detailed execution times for all tested UHD-1 sequences for two, four, eight and ten worker threads are shown in Figure 5. When two threads are used, it can be seen that all measured execution times are slower than the 20 ms/frame real-time limit. Allowing the HHI decoder to use four working threads already results in execution times for the two lowest bitrates (at 2.6 and 5 MBits/s) of the slowest sequence *rain fruits* being faster than 20 ms/frame. As already discussed, Figure 4a shows that the highest speedup for the UHD-1 test sequences and the random access main 10 configuration can be achieved when ten worker threads are used. Consequently, the execution times for at least all points below 20 MBits/s are faster than 20 ms/frame in that case. Overall, it can be said that HEVC real-time decoding of 50 Hz 4K/UHD video on a quad-core mobile CPU like the i7 Sandy Bridge at 2.5 GHz is possible.

4.4 Profiling results

After investigating the overall performance, the contribution of the different parts of the optimized HHI decoder has been analyzed by profiling. The decoding process has been broke down into the following parts:

- PS: Parsing of prediction data side information, e.g. motion vectors, splitting flags and the like.
- PC: Parsing of transform coefficients.
- IP: Intra prediction with SIMD optimized code.

- IT: Inverse transform with SIMD optimized code.
- MC: Motion compensation with SIMD optimized code.
- DF: Deblocking filter with SIMD optimized code.
- SF: Sample adaptive offset filter.
- OT: All other operations including high-level syntax parsing but mostly copying samples from local buffer to picture memory.

The detailed profiling results can be found in Table 3 for 1080p all intra main 10, in Table 4 for 1080p random access main and in Table 5 for 2160p50 random access main 10.

When comparing the results for the two QP values, the first thing that can be observed is that for increasing QP values, the prediction parts increase while the transform coefficient parsing part decreases. This is conclusive since a low QP value reduces the number and size of the transform coefficients. The other thing that can be noticed is, that the SAO filtering takes up more decoding time for reconstructed sample values with finer quantization while more time is spent on deblocking when the quantization is more coarse (more blockiness).

Looking at the results for the different configurations, the main difference between the all intra and the random access configuration is, that most decoding time in the all intra configuration is spent for coefficient parsing (34.2%) and intra prediction (14.5%) while the random access configuration on the other hand spends almost half of the decoding time for motion compensation (41.4% and 45.3%). This is plausible since intra-picture prediction generally produces a much higher residual that leads to more and larger transform coefficients.

Since the profiling results were generated using the optimized HHI decoder with SIMD optimizations, it would be interesting to know how much of the decoding time would be spent in the SIMD optimized parts when no SIMD optimizations are used. Therefore, average profiling results for the scalar code and a SIMD speedup factor for all parts that include SIMD optimized code are listed in the two lines below the average. It can be seen that the interpolation filter benefits the most with measured speedup factors of 8.1 and 5.47. For intra prediction, SIMD optimizations only reduce the decoding time for that part by a factor of 1.4 to 1.7. This matches with the decoding times shown in Table 2 where the overall speedup using SIMD optimization for all intra configurations is much less than for the random access configuration.

Sequence	PS (%)	PC (%)	IT (%)	IP (%)	MC (%)	DF (%)	SF (%)	OT (%)
BasketballDrive QP=27	12.1	32.8	11.8	14.3	0.0	8.4	16.9	3.8
BasketballDrive QP=32	12.6	23.9	14.1	17.5	0.0	11.2	15.5	5.2
BQTerrace QP=27	12.2	47.0	7.0	12.4	0.0	6.1	12.6	2.7
BQTerrace QP=32	14.3	33.0	7.4	15.9	0.0	8.6	17.3	3.6
Cactus QP=27	13.2	39.9	8.5	13.5	0.0	7.3	14.6	3.1
Cactus QP=32	14.6	29.5	10.0	16.6	0.0	9.9	15.2	4.2
Kimono1 QP=27	8.3	33.3	15.9	12.6	0.0	8.7	16.2	5.0
Kimono1 QP=32	8.8	25.9	18.4	15.4	0.0	11.2	13.8	6.6
ParkScene QP=27	12.8	43.5	8.4	12.1	0.0	6.5	13.9	2.9
ParkScene QP=32	13.7	32.8	10.0	14.4	0.0	9.0	16.3	3.8
Average	12.3	34.2	11.1	14.5	0.0	8.7	15.2	4.1
Average scalar	8.7	25.2	23.8	13.8	0.0	15.9	10.3	2.2
Speedup SIMD			3.55×	1.45×		2.83×		

Table 3. Profile 1080p all intra main 10. PS: parse side information / prediction data. PC: parse transform coefficients. IP: intra prediction. IT: inverse transform. MC: motion compensation. DF: deblocking filter. SF: sample adaptive offset filter. OT: other including high-level syntax parsing.

Sequence	PS (%)	PC (%)	IT (%)	IP (%)	MC (%)	DF (%)	SF (%)	OT (%)
BasketballDrive QP=27	13.9	13.0	6.6	4.2	34.7	7.2	13.5	6.8
BasketballDrive QP=32	12.6	7.7	5.9	4.1	43.5	8.1	8.7	9.5
BQTerrace QP=27	14.2	14.9	3.8	1.3	38.3	5.7	14.8	7.0
BQTerrace QP=32	11.2	6.6	3.0	1.6	52.4	5.7	8.4	11.2
Cactus QP=27	16.6	14.6	5.9	3.3	31.5	7.6	12.2	8.4
Cactus QP=32	15.0	8.7	5.6	3.5	37.8	8.3	9.7	11.5
Kimono1 QP=27	11.8	11.3	8.1	2.2	41.4	7.3	9.8	8.2
Kimono1 QP=32	10.0	7.0	7.0	2.3	47.7	7.5	8.1	10.5
ParkScene QP=27	16.3	14.1	3.1	2.9	39.9	6.4	10.1	7.1
ParkScene QP=32	13.4	8.4	2.8	3.0	46.5	6.7	9.6	9.6
Average	13.5	10.6	5.2	2.8	41.4	7.0	10.5	9.0
Average scalar	3.4	2.8	4.3	1.0	79.6	4.3	2.7	2.0
Speedup SIMD			3.43×	1.41×	8.10×	2.53×		

Table 4. Profile 1080p random access main. PS: parse side information / prediction data. PC: parse transform coefficients. IP: intra prediction. IT: inverse transform. MC: motion compensation. DF: deblocking filter. SF: sample adaptive offset filter. OT: other including high-level syntax parsing.

Sequence	PS (%)	PC (%)	IT (%)	IP (%)	MC (%)	DF (%)	SF (%)	OT (%)
EBUfountainlady QP=27	10.1	11.3	6.5	4.5	43.6	8.8	5.0	10.2
EBUfountainlady QP=32	8.1	6.3	5.9	3.9	50.6	8.8	3.9	12.5
EBULupoconfetti QP=27	9.3	12.0	15.2	9.6	20.5	15.3	6.0	12.1
EBULupoconfetti QP=32	8.5	7.6	14.1	10.1	23.8	16.5	4.9	14.6
EBUrainfruits QP=27	8.0	6.1	4.2	1.1	56.8	6.4	6.2	11.2
EBUrainfruits QP=32	6.3	3.5	2.7	1.1	62.5	5.5	5.5	13.0
EBUstudiodancer QP=27	7.9	6.8	7.1	5.1	45.8	10.8	5.0	11.7
EBUstudiodancer QP=32	6.6	4.0	5.4	4.6	51.3	10.5	3.9	13.7
EBUwaterfallpan QP=27	7.0	13.4	8.7	3.3	44.5	7.7	5.9	9.6
EBUwaterfallpan QP=32	5.3	6.9	7.4	2.8	54.1	7.1	4.6	11.9
Average	7.7	7.8	7.7	4.6	45.3	9.7	5.1	12.0
Average scalar	2.2	2.3	8.1	2.3	74.2	6.1	1.5	3.3
Speedup SIMD			3.65×	1.74×	5.47×	2.24×		

Table 5. Profile 2160p random access main 10. PS: parse side information / prediction data. PC: parse transform coefficients. IP: intra prediction. IT: inverse transform. MC: motion compensation. DF: deblocking filter. SF: sample adaptive offset filter. OT: other including high-level syntax parsing.

5. CONCLUSION

In this paper, it has been shown that HEVC software decoding of 4K 50Hz 10 bit video on a quad-core mobile CPU is possible for bitrates up to 20 Mbits/s. In order to achieve that, SIMD code optimization and parallel decoding is essential. In future developments, further speedup could be obtained by using most recent and upcoming SIMD instruction set extensions like AVX2 and by adding SIMD optimizations for the sample adaptive offset filter.

REFERENCES

1. Cisco, “Visual Networking Index (VNI): Forecast and Methodology, 2010-2015,” 2011.
2. B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, “High Efficiency Video Coding (HEVC) text specification draft 10 (for FDIS & Last Call).” document JCTVC-L1003 of JCT-VC, Jan. 2013.
3. G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Transactions on Circuits and Systems for Video Technology* **22**, pp. 1649–1668, Dec. 2012.
4. J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, “Comparison of the Coding Efficiency of Video Coding Standards Including High Efficiency Video Coding (HEVC),” *IEEE Transactions on Circuits and Systems for Video Technology* **22**, pp. 1669–1684, Dec. 2012.
5. F. Bossen, B. Bross, K. Sühring, and D. Flynn, “HEVC Complexity and Implementation Analysis,” *IEEE Transactions on Circuits and Systems for Video Technology* **22**, pp. 1685–1696, Dec. 2012.
6. M. Alvarez-Mesa, C. C. Chi, B. Juurlink, V. George, and T. Schierl, “Parallel Video Decoding in the Emerging HEVC Standard,” in *Proceedings of the 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2012.
7. C. C. Chi, M. Alvarez-Mesa, B. Juurlink, V. George, and T. Schierl, “Improving the Parallelization Efficiency of HEVC Decoding,” in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Oct 2012.
8. 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 Transaction of Circuits and Systems for Video Technology* **22**, pp. 1827 –1838, Dec. 2012.
9. C. C. Chi, M. Alvarez-Mesa, J. Lucas, B. Juurlink, and T. Schierl, “Parallel HEVC Decoding on Multi- and Many-core Architectures,” *Journal of Signal Processing Systems* , pp. 1–14, Dec. 2012.
10. JCT-VC, “Subversion repository for the HEVC Test Model (HM).” `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/`, 2013.
11. European Broadcast Union, “EBU UHD-1 Test Set.” `http://tech.ebu.ch/testsequences/uhd-1`, 2012.
12. F. Bossen, “Common HM test conditions and software reference configurations.” document JCTVC-L1100 of JCT-VC, Jan. 2013.