VVenC
Fraunhofer Versatile Video Encoder

v0.3.1.0

Jens Brandenburg, Adam Wieckowski, Tobias Hinz, Benjamin Bross
Video Coding & Analytics Department,
Fraunhofer Heinrich Hertz Institute (HHI), Berlin, Germany

1 INTRODUCTION

In July 2020, the Joint Video Experts Team (JVET), a collaborative project of the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG), has finalized a new video coding standard called Versatile Video Coding (VVC) [1][2]. VVC is the successor of the High Efficiency Video Coding (HEVC) standard [3][4] and has been released by ITU-T as H.266 and by ISO/IEC as MPEG-I Part 3 (ISO/IEC 23090-3). The new standard targets a 50% bit-rate reduction over HEVC at the same visual quality. In addition, VVC proves to be truly versatile by including tools for efficient coding of video content in emerging applications, e.g. high dynamic range (HDR), adaptive streaming, computer generated content as well as immersive applications like 360 degree video and augmented reality (AR).

The VVC test model (VTM) [5] serves as a common reference implementation, i.e. a test bed for evaluation and verification of proposed technologies during standardization. While VTM used to be the only publicly available encoder and decoder implementation of the VVC standard, it is aimed at correctness, completeness and readability and should not serve as a real-world example of a VVC encoder and decoder.

The Fraunhofer Versatile Video Encoder (VVenC) development was initiated to provide a publicly available, fast and efficient VVC encoder implementation. The VVenC software is based on VTM, with optimizations including software redesign to mitigate performance bottlenecks, extensive SIMD optimizations, improved encoder search algorithms and multi-threading support to exploit parallelization. Additionally, VVenC supports real-world encoder features, including frame-level rate control and perceptually optimized encoding in order to provide a flexible, fast and easy to use video encoding solution for the VVC standard.

Bit-streams encoded with VVenC can be decoded by any VVC standard compliant decoder, e.g. the VTM-12.0 reference software decoder. Alternatively, the fast Fraunhofer Versatile Video Decoder (VVdeC) solution can be used [6].

Features at a Glance

- Easy to use encoder implementation with five predefined quality/speed presets.
- Optimized VVC encoder providing speedups between 12x and 1100x over VTM-12.0 for HD and UHD test sequences depending on the chosen quality/speed preset.
- Perceptual optimization to improve subjective video quality.
- Frame-level rate control supporting VBR encoding.
- Expert mode encoder interface available, allowing fine-grained control of the encoding process.

2 GETTING STARTED

2.1 HOW TO OBTAIN VVENC?

The software is hosted at GitHub under: https://github.com/fraunhoferhhi/vvenc

To clone the project use:

```
git clone https://github.com/fraunhoferhhi/vvenc
```

2.2 HOW TO BUILD VVENC?

The software uses CMake to create platform-specific build files. A working CMake installation is required for building the software. Download CMake from http://www.cmake.org/ and install it. The following targets are supported: Windows (Visual Studio), Linux (gcc) and MacOS (clang).

How to build for Windows?

In order to compile the software for Windows, Visual Studio 15 2017 or higher and CMake Version 3.12 or higher are required. Install gnuwin32 that provides make for Windows. To build the software open a command prompt window, change into the project directory and use:

```
make install-release
```

This will create the statically linked release version of the encoder applications in the install/bin/release-static/ subdirectory.

How to build for Linux/MacOS?

In order to compile the software for Linux, gcc-7.0 or higher and CMake Version 3.12 or higher are required. For MacOS Xcode and CMake Version 3.12 or higher are required. To simplify the build process a Makefile with predefined targets is available. To build the VVenC encoder applications open a terminal, change into the project directory and use:

```
make install-release
```

This will create the statically linked release version of the encoder applications in the install/bin/release-static/ subdirectory.

2.3 HOW TO USE VVENC?

The encoder project includes two encoder executables, a simple encoder app (vvencapp) and a full featured expert encoder (vvencFFapp).

How to use the simple encoder app?

The simple encoder app (vvencapp) can be used in one of five predefined presets. Each preset represents a different tradeoff between encoder runtime and video quality. In the slowest preset, the encoder reaches the highest compression gain, whilst in the fastest preset the runtime is significantly decreased. These preset configurations have been determined based on the Pareto optimal configuration set of the encoder configuration space, which is detailed in [7]. A list of the main encoder command line parameters is shown in Table I.

Table I: List of important encoder options. For full list please use the --help option.

OPTION	DEFAULT	DESCRIPTION
--help,-h	-	Show basic help
--input <str>	-	Raw yuv input file
--size <wxh>	1920x1080	Input file resolution (width x height)
--framerate <int>	60	Temporal rate of input file. Required for rate control and calculation of output bit-rate. Also recommended with perceptual QP adaptation (see `--qpa` option below).
--format <str>	yuv420	Set input format to YUV 4:2:0 8bit (yuv420) or YUV 4:2:0 10bit (yuv420_10)
--output <str>	not set	Bit-stream output file
--preset <str>	medium	Preset for specific encoding setting (faster, fast, medium, slow, slower)
--qp <int>	32	Quantization parameter (0..63)
--bitrate <int>	0	Bit-rate for rate control (0 constant QP encoding rate control off, otherwise bits per second). Rate control requires correct framerate.
--passes <int>	2	Number of passes used for rate control
--qpa <int>	1	Perceptual QP adaptation (QPA) to improve subjective video quality (0: off, 1: on)
--threads <int>	Size > 832x480: 8 else : 4	Number of threads (1..N)

Example usage: Given a YUV 4:2:0 input file with a bit-depth of 8bit and a resolution of 176x144 pixels, the following call will encode the input file with the medium speedup preset:

```
vvencapp --preset medium -i BUS_176x144_75@15.yuv -s 176x144 -o str.266
```

How to use the full featured expert mode encoder?

The expert mode encoder (**vvencFFapp**) is based on the VTM configuration scheme. Most of the parameters have been kept similar to VTM, but for some parameters, additional modes are available. Furthermore, not supported options have been removed. Example configuration files for the expert mode encoder can be found in the **cfg** sub-directory (see Table II).

Example usage: In order to start your first experiments with the expert mode encoder, adapt the **sequence.cfg** configuration file to your input YUV source file and use the following command:

```
vvencFFapp -c randomaccess_medium.cfg -c sequence.cfg
```

Table II: Expert mode encoder configuration files provided in the ./cfg sub-directory.

CONFIGURATION FILE	DESCRIPTION
sequence.cfg	Sequence specific configuration parameters. Must be always adapted to the input sequence.
randomaccess_[faster, fast, medium, slow, slower].cfg	Random access configuration for different presets. Each configuration file corresponds to one of the 5 preset modes.
qpa.cfg	Perceptually optimized QPA configuration file.
rc1p.cfg	Single pass rate control configuration, overriding default fix QP setup.
rc2p.cfg	Two pass rate control configuration, overriding default fix QP setup

3 ENCODER PERFORMANCE

The encoder performance tests focus on the most relevant HD (1920x1080) and UHD (3840x2160) resolution use cases with random access encoding as defined in the JVET common test conditions (CTC) [8]. All presented results are shown for the CTC test sequences, i.e. classes A1 and A2 for UHD and class B for HD sequences. Constant QP encoding is used with QP values of 22, 27, 32 and 37 according to VTM CTC. Reported rate distortion results are calculated as Bjøntegaard delta rate (BD-rate) [9]. For the multi-threading use case 8 threads have been enabled in the encoder configuration. All tests have been performed on Dell Super Micro servers with Intel Xeon processors E5-2697A v4 @2.6GHz.

3.1 PSNR OPTIMIZED USE CASE

The PSNR BD-rate gain of VVenC over the HEVC test model reference software HM-16.22 is shown in Figure 1. PSNR BD-rate represents the approximate average bit-rate savings between two encoders for the same objective quality (PSNR). Here lower values mean larger bit-rate savings with respect to the HM-16.22 anchor. Please note the logarithmic scale of the relative encoder runtime in comparison to HM-16.22.

With the slower preset and multi-threading enabled the BD-rate gain of VVenC over HM is similar to VTM-12 common test conditions (CTC), but a speedup of more than 16x for UHD and 12x for HD sequences is achieved over VTM.

With the faster preset and multi-threading the BD-rate gain over HM is still approx. 10.8%, with a speedup of more than 1000x for UHD and 900x for HD respectively over VTM. Comparing the runtime with HM gives a speedup of 140x for UHD and 120x for HD.

As a good tradeoff between encoder runtime and BD-rate performance, we recommend the medium preset with multi-threading enabled. Here, the BD-rate gain over HM is approx. 37.6%, which is quite close to the slow preset and VTM CTC, but in comparison to VTM-12.0 the encoder runs 210x faster for UHD and 160x faster for HD sequences. Compared to HM-16.22 this is an encoder runtime speedup of 28x for UHD and 21x for HD. A summary of all results is shown in Table III.

Table III: PSNR BD-rate and multi-threaded encoder speedup for HD and UHD test sequences.

PRESET	HD			UHD		
	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	PSNR BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM
FASTER	-7.9%	120x	900x	-13.1%	140x	1000x
FAST	-26.0%	67x	510x	-28.2%	80x	590x
MEDIUM	-36.6%	21x	160x	-38.5%	28x	210x
SLOW	-40.1%	7.4x	57x	-41.9%	10x	74x
SLOWER	-43.3%	1.6x	12x	-45.2%	2.2x	16x

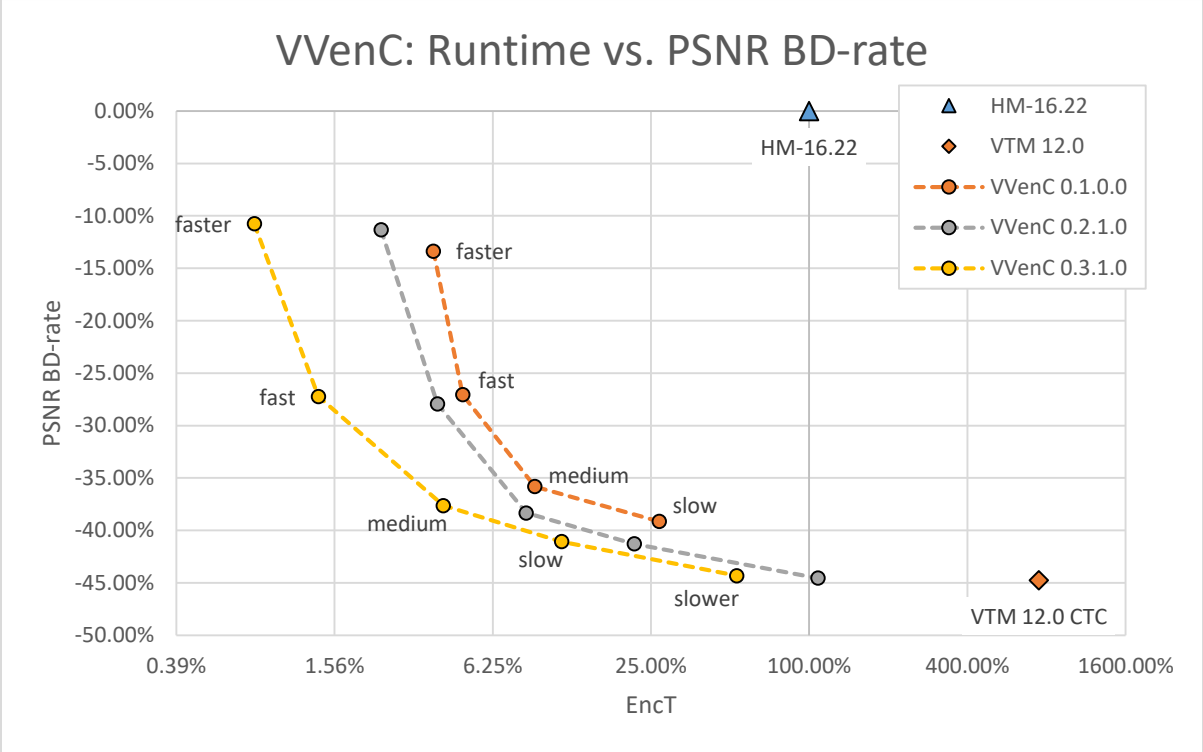

Figure 1: PSNR BD-rate gain and relative encoder runtime in comparison to HM-16.22 for VVenC and VTM. Results are given for the 5 preset options: faster, fast, medium, slow and slower. Lower PSNR BD-rate values mean a better compression for the same objective quality in terms of PSNR.

Additionally, Figure 2 includes multi-threaded results for the HEVC open-source encoder x265 v3.4 at comparable speed presets [10]. For the comparison with VVenC, also x265 was configured to run with 8 threads. Besides sequence-specific parameters, the following parameter settings have been used:

```
--preset {0,1,2,3,…,9} --tune psnr --crf {17,22,27,32} --keyint 1s --min-keyint 1s
--profile main10 --output-depth 10
```

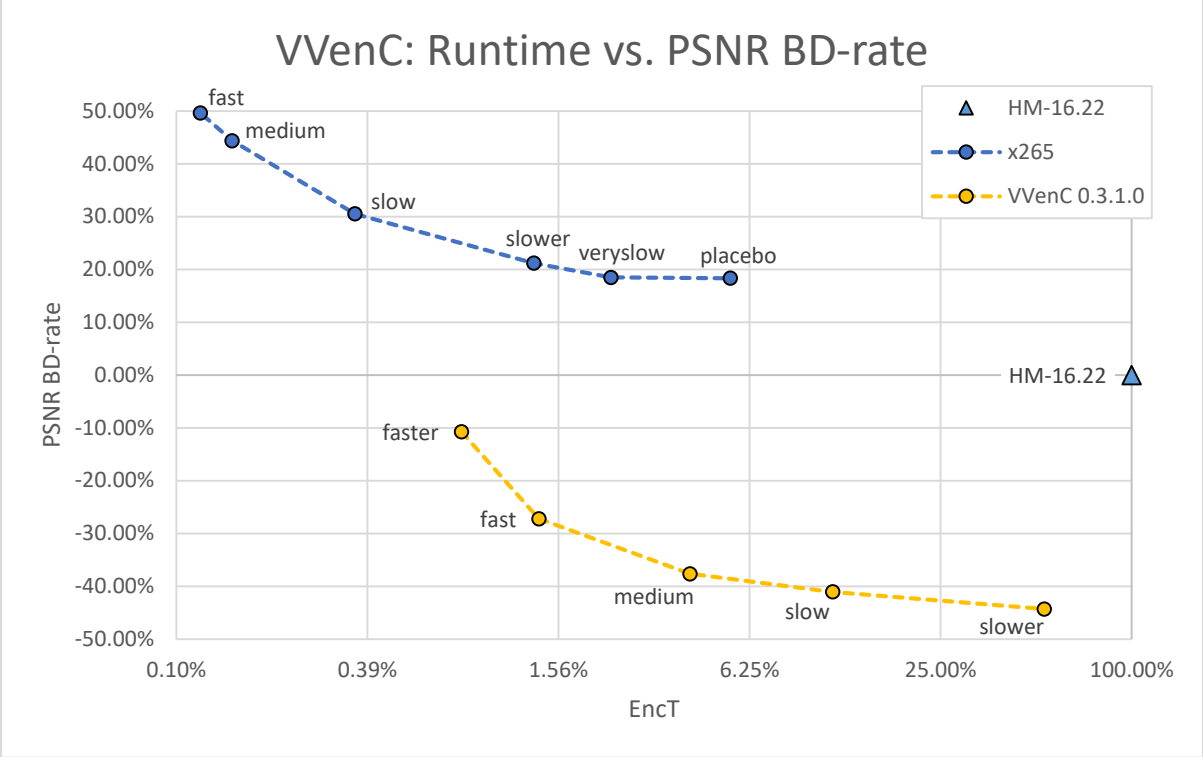

Figure 2: PSNR BD-rate gain and relative encoder runtime in comparison to HM-16.22 for VVenC and x265 running with 8 threads. Lower PSNR BD-rate values mean a better compression for the same objective quality in terms of PSNR.

3.2 Perceptually Optimized Quantization Parameter Adaptation

To improve the perceived (subjective) coding quality, VVenC supports a low-complexity quantization parameter adaptation (QPA) algorithm based on a model of the human visual system. To evaluate the quality of the perceptually optimized quantization parameter adaptation (PQPA), multiscale structural similarity measure (MS-SSIM) [11] is used.

In Figure 3 the MS-SSIM BD-rate gain of VVenC over the HEVC test model reference software HM-16.22 is shown (lower is better). For VTM simulation, common test conditions CTC with additional PQPA is used (--SliceChromaQPOffsetPeriodicity=1, --PerceptQPA=1). With PQPA enabled, the speedups achieved over HM and VTM are similar to the Non-PQPA results presented in the previous section. This demonstrates the low-complexity nature of the PQPA algorithm. Comparing the MS-SSIM based BD-rates, additional bit-rate reduction can be achieved. Especially for the slower preset, an MS-SSIM BD-rate gain of more than 3.3% over VTM CTC without PQPA is realized. We recommend using the medium preset with multi-threading and PQPA enabled as a good tradeoff between encoder runtime and resulting perceived video quality. A summary of the MS-SSIM results for PQPA is shown in Table IV.

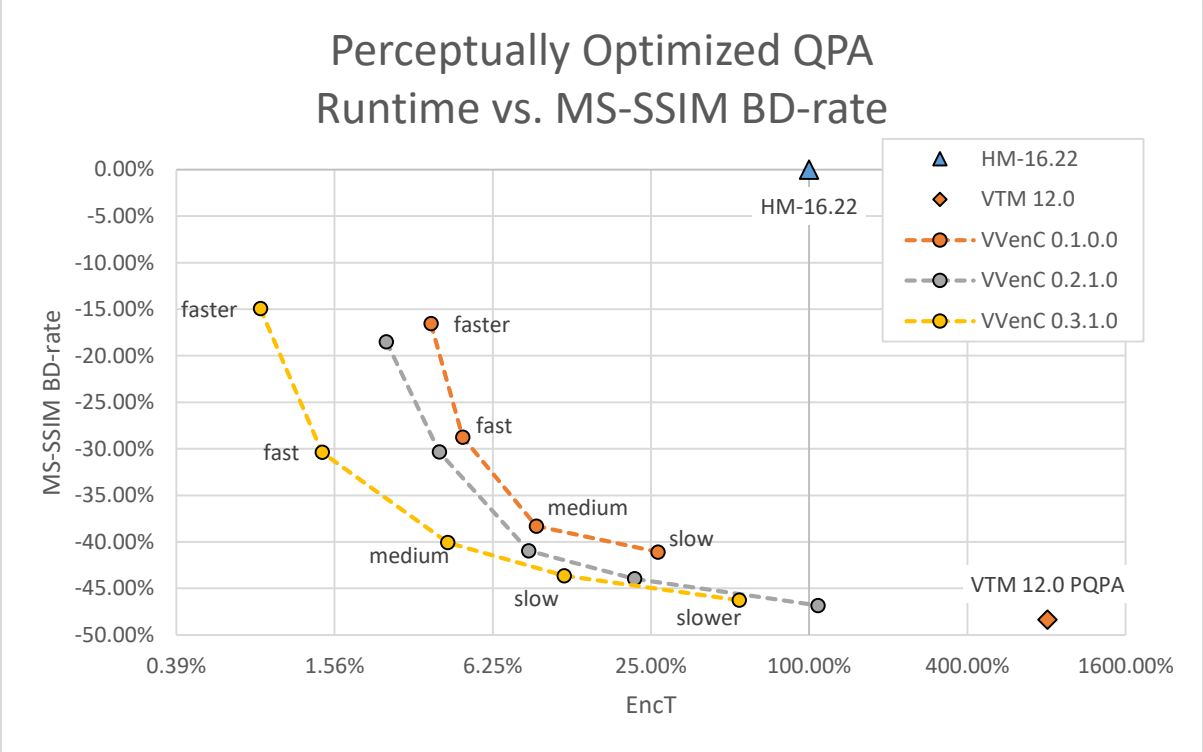

Figure 3: MS-SSIM BD-rate gain and encoder runtime in comparison to HM-16.22 for VTM and VVenC with perceptually optimized quantization parameter adaptation enabled. VVenC results are given for the 5 preset options: faster, fast, medium, slow and slower. Lower MS-SSIM BD-rate values mean a better compression for the same quality in terms of MS-SSIM.

Additionally, Figure 4 includes multi-threaded results for the HEVC open-source encoder x265 v3.4 tuned for SSIM at comparable speed presets [10]. For the comparison with VVenC, also x265 was configured to run with 8 threads. Besides sequence-specific parameters, the following parameter settings have been used:

```
--preset {0,1,2,3,…,9} --tune ssim --crf {17,22,27,32} --keyint 1s --min-keyint 1s
--profile main10 --output-depth 10
```

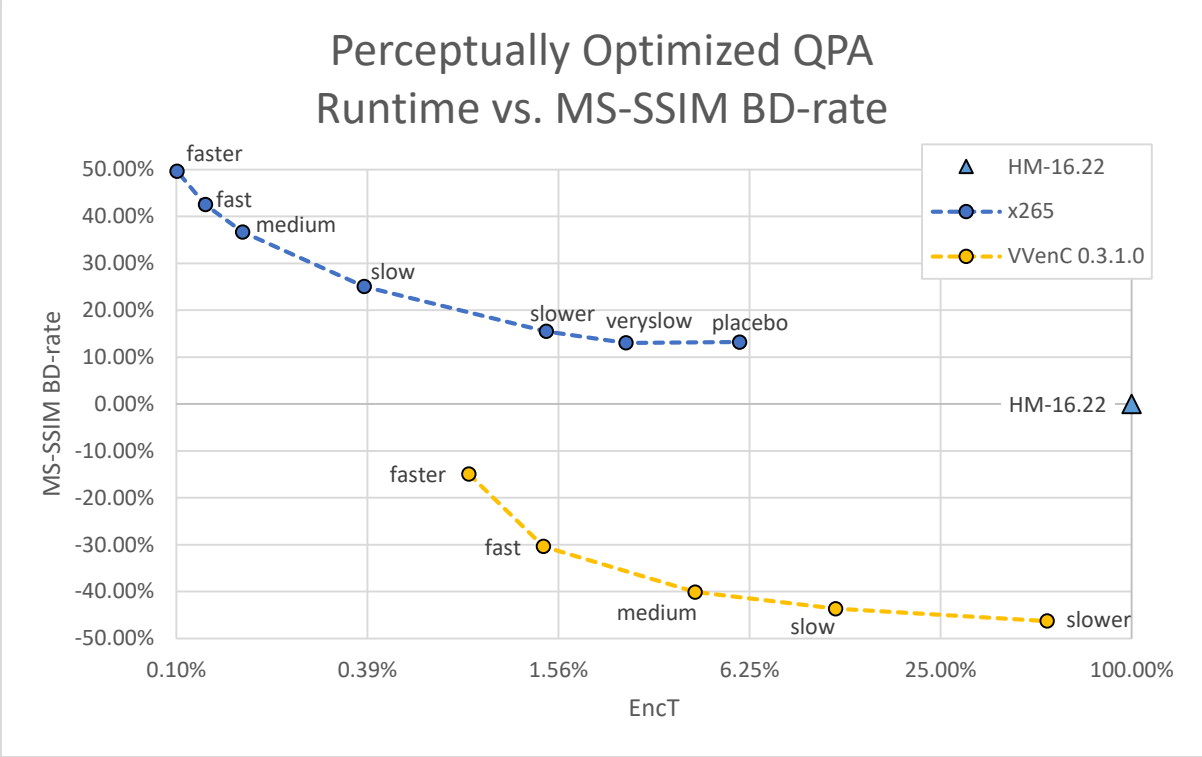

Figure 4: MS-SSIM BD-rate gain and encoder runtime in comparison to HM-16.22 for VVenC with QPA enabled and x265 with --tune=ssim. Lower MS-SSIM BD-rate values mean a better compression for the same quality in terms of MS-SSIM.

Table IV: MS-SSIM BD-rate gain and multi-threaded encoder speedup for HD and UHD test sequences for VVenC with perceptually optimized QPA enabled.

PRESET	HD			UHD		
	MS-SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM	MS-SSIM BD-rate vs. HM	Speedup vs. HM	Speedup vs. VTM
FASTER	-17.3%	110x	870x	-12.9%	130x	1100x
FAST	-31.2%	64x	510x	-29.7%	78x	630x
MEDIUM	-39.7%	20x	160x	-40.4%	27x	220x
SLOW	-43.2%	7.3x	58x	-44.0%	9.8x	80x
SLOWER	-45.6%	1.6x	13x	-46.8%	2.1x	17x

3.3 MULTI-THREADING

In Figure 5 the multi-threading performance scaling is shown for the faster and fast presets for up 16 threads and up to 32 threads for HD and UHD test sequences respectively. Additionally, the scaling for the medium preset for up to 16 threads is given. The advanced threading since version 0.3 greatly improves scaling over earlier versions. It is apparent, that the threading performance is better with the fast and faster presets when utilizing more than 8 cores for HD or 12 for UHD.

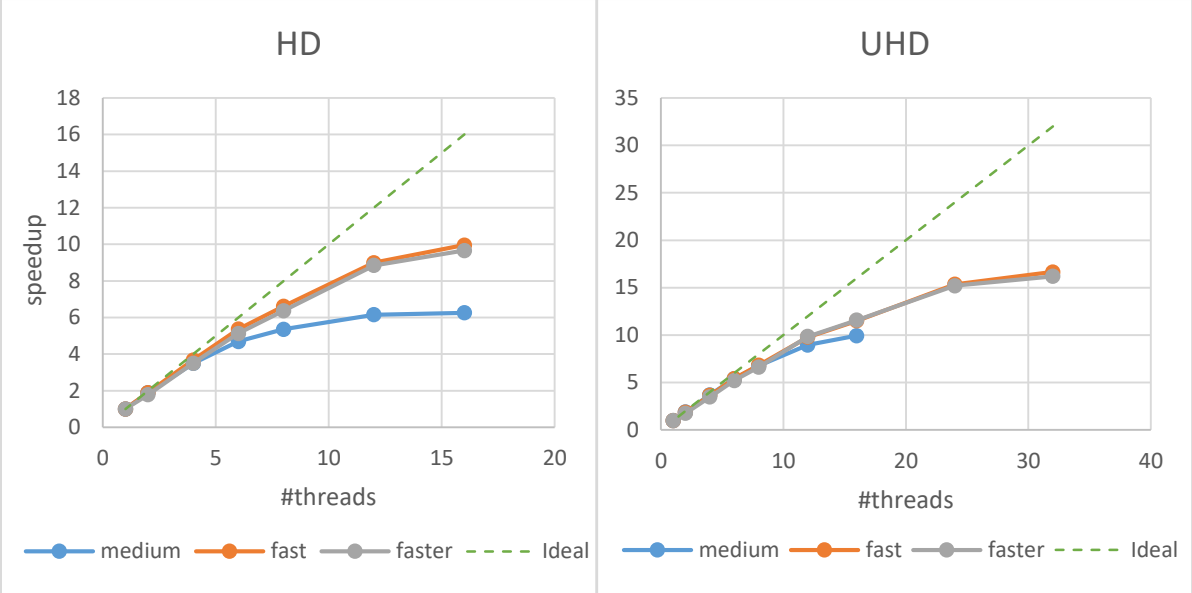

Figure 5: Multi-threading speedup dependent on the number of used threads. The tests were performed on high-performance computer with an AMD EPYC 7502P 32-core processor.

While the current multi-threading approach has only minimal influence on the compression performance of less than 1% bit-rate increase for the same visual quality, there are additional options allowing to significantly increase the multi-threading performance.

When operating the encoder in an area where the speedup curve saturates, the multi-threading performance can be improved by setting some additional parameters. For all presets `--WaveFrontSynchro=1` can be enabled, which reduces the encoding lag of CTU lines within a picture. Additionally, for the medium preset, `--CTUSize=64` can be set, allowing it to achieve a scaling similar to the faster and fast presets. Both options are only available in the expert app and come with a gain trade-off. The latter option is enabled by default for the fast and faster preset.

4 CHANGELOG

4.1 v0.3.1.0

- Corrected compatibility with C++17
- Rate control cleanup (changes to external interface)
- Added vvencCfgExpert::m_log2MinCodingBlockSize interface variable
- Added Log2MinCodingBlockSize parameter to the expert app to restrict minimal allowed block size
- Fixed interaction between LMCS and ISP
- Fixed motion clipping in DMVR
- Reduced and defragmented memory usage
- Adapted and improved fast and faster presets
- Minor speed-ups to other presets

4.2 v0.3.0.0

- Minor bugfixes
- Preset refinement
- Added HDR parameter support
- Memory reduction
- Improved multi-threading
- Added a mode allowing for RPR switching with constrained drift
- Additional improvements and speed-ups
- New parameters: **-aud**, **-vui**, **-hrd**, **-hdr** (controlling the generation of appropriate parameter sets and optimized HDR settings)
- Changed parameter: **--qpa** (now only 0/1)
- Lib-interface changed (unified between the apps, now supporting all options available to the expert app)

4.3 v0.2.1.0

- Harmonization of QPA and 2-pass rate control
- Memory savings
- Minor bugfixes

4.4 v0.2.0.0

- Changed the license to modified 3-clause BSD
- Bugfixes to AUD and Decoded Picture Hash SEI generation
- New tools added: Intra Sub-Partitions (ISP), Transform Skip Residual Coding (TSRC), Block-level Differential Pulse Code Modulation (BDPCM)
- Added 2-pass rate control
- Added 1-pass rate control support for GOP32
- Using GOP32 as default in all presets

- Added slower preset
- New simple-app parameters: --refreshsec, --internal-bitdepth, --passes, --segment
- Using GNUInstallDirs
- Using Cmake versioning
- Added basic tests ("make test")
- Added address sanitizer support
- Many small speed-ups and improvements

4.5 v0.1.0.1

- Fix for SIMD setting using the simple-app

4.6 v0.1.0.0

- Initial release

5 REFERENCES

[1] B. Bross, J. Chen, S. Liu and Y.-K. Wang, "Versatile Video Coding Editorial Refinements on Draft 10," document JVET-T2001, Joint Video Experts Team (JVET), Oct. 2020.

[2] ITU-T and ISO/IEC JTC 1, Versatile Video Coding, Rec. ITU-T H.266 and ISO/IEC 23090-3 (VVC), July 2020.

[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, vol. 22, no. 12, pp. 1649–1668, 2012.

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

[5] VTM software repository, version VTM-12.0. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

[6] VVdeC software repository. Available online: https://github.com/fraunhoferhhi/vvdec

[7] J. Brandenburg et al., "Towards Fast and Efficient VVC Encoding", IEEE 22nd Workshop on Multimedia Signal Processing (MMSP 2020), Tampere, Finland, 2020.

[8] F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Suehring, "JVET common test conditions and software reference configurations for SDR video," document JVET-N1010, Joint Video Experts Team (JVET), Apr. 2019.

[9] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, USA, 2001.

[10] x265 software repository, version 3.4. Available online: https://github.com/videolan/x265/tree/Release_3.4

[11] Z. Wang, E. Simoncelli, and A. C. Bovik, "Multi-Scale Structural Similarity for Image Quality Assessment," in Proc. IEEE Asilomar Conf. Signals, Systems, and Comp., Pacific Grove, 2003.

6 LICENSE

Please see the file https://github.com/fraunhoferhhi/vvenc/blob/master/LICENSE.txt in the repository for the terms of use of the contents of the VVenC repository.

For more information, please contact: vvc@hhi.fraunhofer.de