VVdeC
Fraunhofer Versatile Video Decoder

v0.2.0.0

Adam Wieckowski, 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 will be 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 an VVC encoder and decoder.

The Fraunhofer Versatile Video Decoder (VVdeC) development was initiated to provide a publicly available and fast VVC decoder implementation. The VVdeC software is based on VTM, with optimizations including software redesign to mitigate performance bottlenecks, extensive SIMD optimizations and extensive multi-threading support to exploit parallelization.

VVdeC can decode raw bitstreams created by any VVC standard compliant encoder, e.g. the VTM-10.0 reference software encoder or the Fraunhofer Versatile Video Encoder (VVenC) [6]. More information on the decoder can be found in [7].

Supported features

- Main10 profile without sub-pictures

Features in development/experimental

- Other chroma sub-sampling formats (422, 444)

Not yet supported features

- Sub-pictures
- Other profiles

2 GETTING STARTED

2.1 HOW TO OBTAIN VVDEC?

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

To clone the project use:

> git clone https://github.com/fraunhoferhhi/vvdec

2.2 HOW TO BUILD VVDEC?

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/** 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/** subdirectory.

2.3 HOW TO USE VVDEC?

The VVdeC project provides a single application, which allows to decode raw VVC bitstreams into raw YUV files (for 10-bit input, the output is yuv420p10le format in *ffmpeg*; for 8-bit input, the output is yuv420p).

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

OPTION	DEFAULT	DESCRIPTION
--help,-h	-	Show basic help
--bitstream <str>	-	Raw bitstream input file
--output <str>	-	The name of the output raw yuv file (yuv420p10le format)
--threads <int>	-1	Size of the threadpool allocated for decoding. Set to 0 for single-threaded execution and -1 to allocate one thread per core available
--parsedelay <int>	-1	Maximal number of parsed frames waiting for reconstruction. Increases the decoding latency but improves MT scaling. Set to -1 to allow one parse frame delay per core available.
--SEIDecodedPictureHash	not set	If the bitstream contains Decoded Picture Hash SEI information, the switch enables the decoder to check the values against the reconstruction
--loops <int>	0	Expert: decode the bitstream file multiple times
--verbosity <int>	3	Verbosity level

Example usage: Given a compliant VVC input file str.266, the following call will decode the file into raw YUV raw_yuv.yuv:

    vvdeccapp -b str.266 -o raw_yuv.yuv

3 DECODER 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) [7]. All presented results are shown for the CTC test sequences, i.e. classes A1 and A2 for UHD and class B for HD sequences. The test bitstreams were encoded using the VTM-10.0 reference encoder with constant QP encoding with every second QP value between 21 and 43. The bitstreams were decoded on two different computers with different number of threads used for decoding.
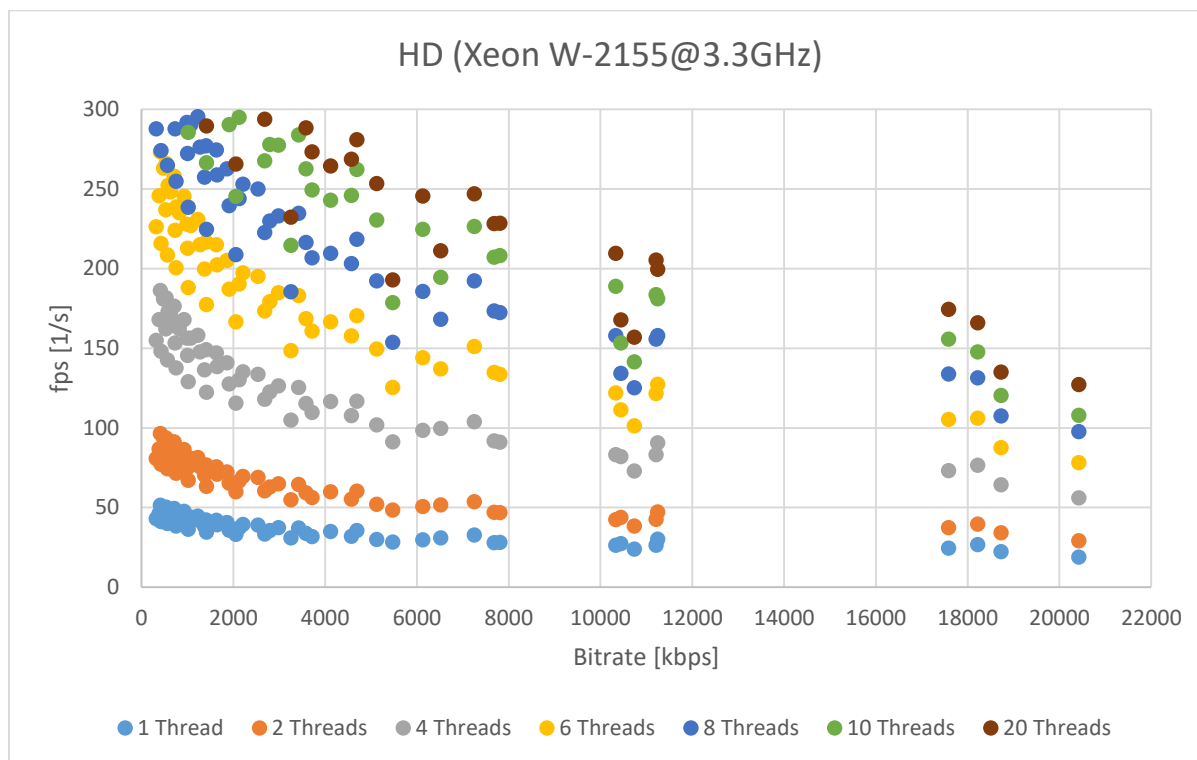
3.1 HD RESULTS

Figure 1: The achieved FPS of the decoding of 10-bit HD sequences depending on the bitrate and number of threads used for decoding on an Intel Xeon W-2155 processor.
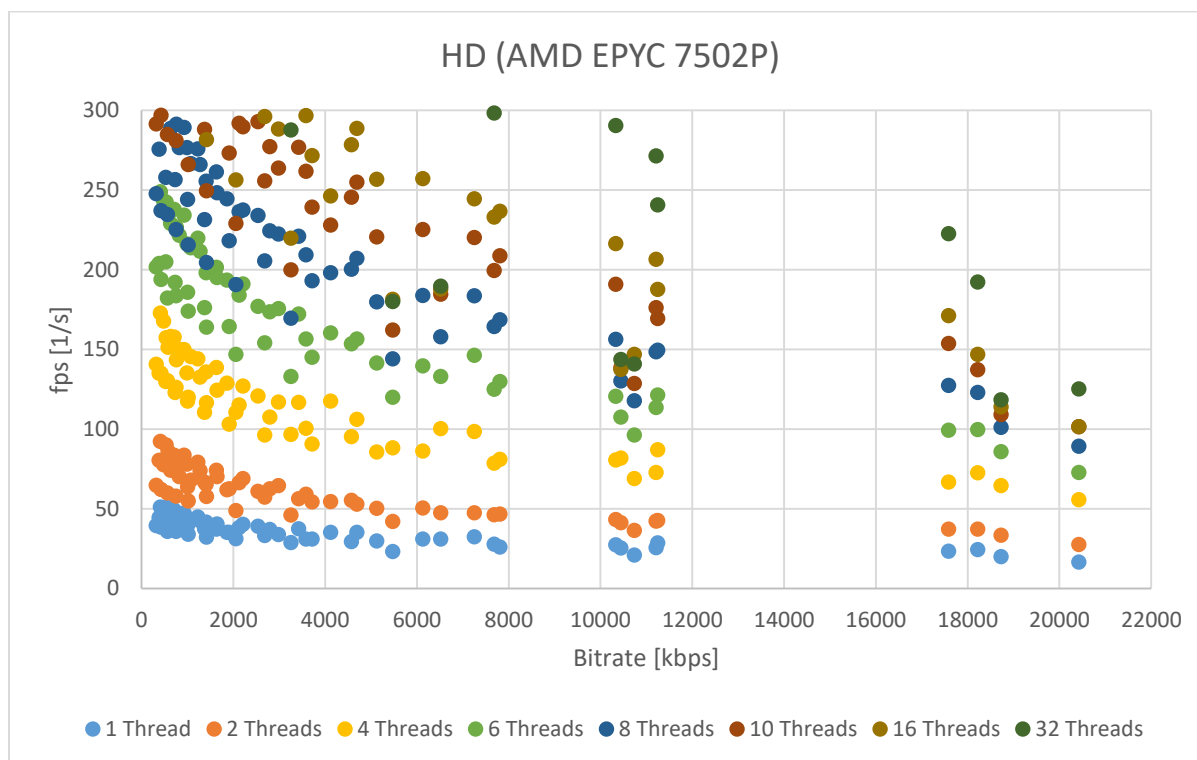

Figure 2: The achieved FPS of the decoding of 10-bit HD sequences depending on the bitrate and number of threads used for decoding on an AMD EPYC 7502P processor.
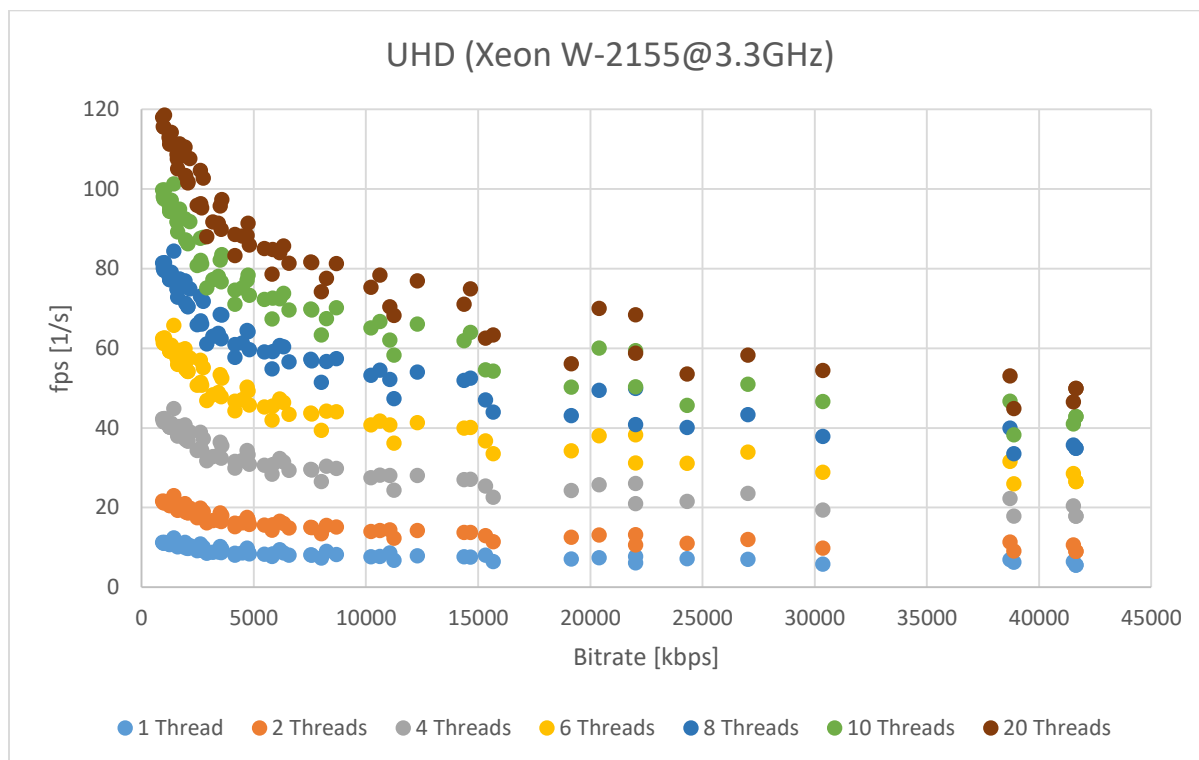
3.2 UHD RESULTS

Figure 3: The achieved FPS of the decoding of 10-bit UHD sequences depending on the bitrate and number of threads used for decoding on an Intel Xeon W-2155 processor.
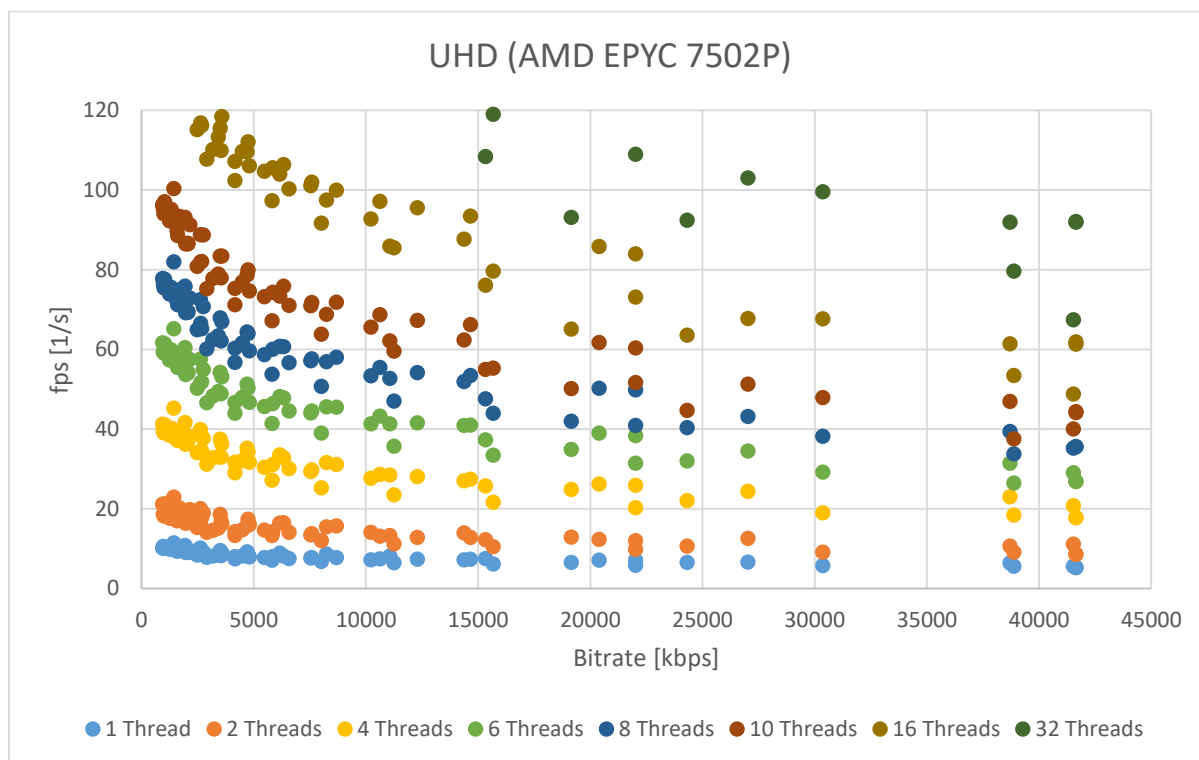

Figure 4: The achieved FPS of the decoding of 10-bit UHD sequences depending on the bitrate and number of threads used for decoding on an AMD EPYC 7502P processor.

4 CHANGELOG

4.1 v0.2

- Changed license to modified 3-clause BSD
- Memory reductions by removing obsolete look-up-tables
- Fixed syntax for multiple slices
- Fixed syntax for Decoded Picture Hash SEI
- Fixed and extended the conformance testset

4.2 v0.1.2

- Various bugfixes
- Fixed handling of multiple slices
- Fixed handling of multiple tiles
- Cleanup of CABACReader and VLCReader to align with spec [1]
- Extended the set of tested confromance sequences
- Changed semantics of **--threads** and **--parsedelay** parameters

4.3 v0.1.0

- Initial release

5 REFERENCES

[1] B. Bross, J. Chen, S. Liu and Y.-K. Wang, "Versatile Video Coding (Draft 10)," document JVET-S2001, Joint Video Experts Team (JVET), Jul. 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-10.0. Available online: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

[6] VVenC software repository. Available online: https://github.com/fraunhoferhhi/vvenc

[7] A. Wieckowski el al., "Towards a Live Software Decoder Implementation for the Upcoming Versatile Video Coding (VVC) Codec," IEEE International Conference on Image Processing (ICIP 2020), Virtual Conference, 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.

6 LICENSE

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

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