Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

J. Samuelsson
P. Hermansson
Divideon

March 5, 2018

The xvc video codec
draft-samuelsson-netvc-xvc-00

Abstract

This document presents a high-level technical description of the xvc video codec. The xvc video codec is a novel video codec that was released in its first version in September 2017. This document provides some technical information as well as a discussion section around how the xvc codec meets the objectives of the NETVC Working Group. The document also includes results comparing a recent revision of xvc with recent revisions of AV1 and HM for three different test cases; All-Intra, single pass Random-Access, and multi-pass Random-Access. The average compression gain (BD-rate for PSNR-Y) for xvc ranges from 3% to 20% for the different test cases.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction 2
2. Technical overview of the xvc video codec 3
2.1. Block structure 4
2.2. Intra prediction 4
2.3. Inter prediction 4
2.4. Transforms 5
2.5. Entropy coding 5
2.6. In-loop filtering 5
2.7. Restriction flags 6
2.8. Version handling 6
2.9. Temporal scalability and parallel decoding 7
3. The xvc reference software 8
4. Results 8
4.1. All-Intra 9
4.2. Single pass Random-Access 10
4.3. Multi-pass Random-Access 11
4.4. Computational complexity 11
5. Discussion 11
6. IANA Considerations 12
7. Security Considerations 13
8. Informative References 13
Authors' Addresses 14

1. Introduction

In September 2017, the new video compression format xvc was released, with source code publicly available at xvc.io [xvcio]. The xvc codec is a software-defined video codec with conformance for bitstreams, decoders, and encoders, defined with reference to the publicly available reference software. The first version of xvc has been developed by Divideon but the source code of the reference software has been made publicly available with a desire to invite to broad usage and continuous development. The developers of the xvc codec strongly believe in open, transparent and collaborative processes for technical development, and the purpose of this contribution is to provide information about the xvc project and to present xvc as a candidate for the Internet Video Codec.

The xvc codec was developed with a desire to make efficient compression methods available to a global market under as open,

transparent and fair conditions as possible. From the start, the ambition has been to construct a codec, which is easy to deploy, taking into account both technical and non-technical considerations. A deeper analysis of such considerations can be found in the xvc introduction white paper [xvcWp], which provides additional background information related to xvc. In summary, it can be said that one of the most central properties of the xvc codec is that it is not a frozen codec and there is no desire to make it a frozen codec. Instead, conformance is defined with respect to the current official version of the xvc reference software. New versions of the software may be (and is intended to be) released, with addition and/or removal of technologies as deemed feasible. This makes it possible to let the codec evolve over time and ensure that the xvc codec only make use of functionality that is available for licensing under the xvc license [xvcLicense].

2. Technical overview of the xvc video codec

The xvc codec has been developed completely from scratch and is built primarily using technology that has been investigated in MPEG and VCEG, for example during the HEVC [HEVC] standardization and in exploratory activities after the finalization of HEVC. One of the most fundamental technical properties of the xvc codec is that the different tools (or processing steps) of the codec, have been isolated within separate conditional clauses, controlled by information (restriction flags) included in the bitstream. This makes the xvc codec extremely flexible and it makes it possible to turn off individual processing steps during run-time, if deemed necessary. This design principle is inspired by the proposal in JCTVC-X0034 [Wenger16].

As a summary of the technology in xvc, it can be said that xvc is a block-based hybrid (inter/intra) codec that operates on raw YUV pictures and compresses them to a NAL (network abstraction layer) unit structured bitstream. Each picture in a video sequence is divided into rectangular blocks of samples, which are predicted from samples in the same picture (intra prediction) or samples in previously coded pictures (inter prediction). Residuals are transformed using non-square transforms and the coded symbols are compressed using a context-adaptive binary arithmetic coder. Block boundaries are filtered using a deblocking filter. The xvc codec supports bit-depths of 8, 10 and 12 bits per sample and chroma formats 4:2:0, 4:2:2, 4:4:4 and monochrome.

2.1. Block structure

A picture that is to be encoded with xvc is divided into blocks of 64x64 pixels, called CTUs. A CTU can either be coded in its full size, or it can be split using a binary split, resulting in two coding units or a quad-tree split, resulting in four coding units. These coding units can be further split into smaller blocks of samples all the way down to coding units of size 4x4 (2x2 for chroma). The splits of a CTU gives rise to a coding unit tree where prediction and transform is performed on the leaf of the trees. Two important differences compared to HEVC can be noted here:

1. In xvc there is no distinction between coding units, prediction units and transform units.

2. In xvc, transform blocks can be non-square.

In the default configuration of xvc, the intra pictures use one coding unit tree for luma and a different coding unit tree for chroma. For inter pictures, a single coding unit tree is used for all color components.

2.2. Intra prediction

There are 67 intra prediction modes in xvc, representing DC prediction, planar prediction and 65 different angular directions. Intra prediction uses previously coded samples to the left and/or above the current block. Reference samples are filtered before being used for prediction. For vertical, horizontal and DC prediction there is also a post filter applied to smooth out the edge to the neighboring block before applying the transform. There is a scheme for estimating which modes are most probable to be used for the next block and through using this scheme the code-words for indicating one of these "most probable modes" can be made shorter than the code-words of other modes. Both chroma components of a block are predicted with the same intra prediction mode, but the chroma prediction mode does not have to be the same as the luma prediction mode. There is also a mode in which the chroma samples are predicted from the luma samples using a linear model.

2.3. Inter prediction

Inter prediction is performed using either uni-prediction (where a single reference picture is used) or bi-prediction (where two reference pictures are used). The xvc codec supports translational motions and affine motions which is applied with different motion vectors for each 4x4 block of a coding unit. The syntax for signaling inter prediction contains shortcuts for indicating that a

candidate motion vector is used (merge mode) with a special case for when there is no residual (skip mode). Motion interpolation filters of length up to 8 are used and applied with quarter-pel resolution for luma. In xvc there is a specific inter prediction mode that accounts for local changes between neighboring samples in the current picture and neighboring samples in the reference picture. This mode gives for example good prediction when the light level of a scene changes over time.

2.4. Transforms

On the encoder side, residual data can be derived from taking the original samples of a block and subtract the predicted samples value of the block. This residual can then be forward transformed to result in a set of coefficients that are quantized and signaled in the bitstream. On the decoder side a corresponding (but inverse) process is applied so that coefficients are inverse-quantized, then inverse-transformed and then added to the predicted sample values. In the first version of xvc, a separable DCT-like transform was used where width and height do not need to be of equal length. The size of the transform is always the same as the size of the coding unit. For transforms of size 64, high frequency components are zeroed out. The latest revision of xvc includes additional transforms that are evaluated and selected based on Rate-Distortion optimization. Regardless of the size of the coding unit, transform coefficients are always coded in subblocks of size 4x4 to enable efficient signaling of subblocks without coefficients. The xvc codec also uses a scheme called sign hiding which makes it possible to derive the sign of one coefficient (instead of signaling it) by rounding the values of the coefficients of a subblock to match the hiding criteria.

2.5. Entropy coding

A Context Adaptive Binary Arithmetic Coder (CABAC) is used in xvc. It resembles the entropy coding method in AVC and HEVC. When a symbol has been coded, the state of the context is updated in order to adjust the probabilities (i.e. the cost) for signaling the same symbol when the same conditions occur again in the same picture (i.e. when the same context is used).

2.6. In-loop filtering

The xvc codec includes a single in-loop filter, which is a deblocking filter, applied on a 4x4 grid of the picture, but only when certain conditions are met. For intra pictures, an edge is filtered if and only if the samples on the different sides of the edge belongs to different coding units. The filtering operation modifies up to two

samples on each side of the edge, depending on the local characteristics of the edge.

2.7. Restriction flags

An xvc bitstream starts with a segment header which provides information about the properties of the coded video such as width, height, bitdepth, chroma format etc. The segment header also contains flags that indicate for each individual tool of the codec whether the tool is used in this bitstream or not. These flags correspond to the fundamental design principle in the xvc codec which is to isolate each individual feature (coding tool) within conditional clauses. The conditional clauses are evaluated during run time which means that the decoder is capable of executing both options of each conditional clause. This makes it easy to evaluate the impact of a specific tool in terms of compression and in terms of complexity, without having to compile different versions of the software for each evaluated combination of tools.

In the xvc software, the restriction flags are used instead of #define clauses for different tools. This ensures that the whole codebase is exercised during compilation and makes it easy to verify that different combinations of tools turned on and off work well together. The only macro-defined constant in the xvc software is XVC_HIGH_BITDEPTH, which controls if internal bitdepths above 8 are supported. The first version of xvc contained 62 restriction flags that applies to different areas of the codec. The latest revision of xvc, in the dev branch of the GitHub repository, adds another 14 restriction flags corresponding to new tools that will be part of the next official version of xvc.

2.8. Version handling

The xvc codec has a simple but powerful scheme for version handling. All xvc bitstreams include, in the beginning of each segment header, one syntax element that represents the major version and one syntax element that represents the minor version.

The major version is increased when non-backwards compatible changes are introduced, typically when new technology is added to xvc. When a new major version of xvc is released, all existing decoders need to be updated in order to support bitstreams of the new major version. Existing bitstreams do not have to be updated when the major version of xvc is increased, since new technology is activated only for new bitstreams that uses the new major version.

The minor version is increased when backwards compatible changes are introduced. In practice, this occurs when technology is being

disabled through setting their restriction flag equal to one. When a new minor version of xvc is released, existing decoders do not need to be updated immediately, since they already have support for decoding bitstreams in which the disabled technology is turned off. However, the xvc license requires that they are updated at some point so that they do not include support for the technology that is no longer available in xvc. Existing bitstreams have to be updated when the minor version of xvc is increased since the new version of the reference decoder will reject bitstreams with too low minor version (since they make use of technology that is no longer available in xvc). In practice, it is expected to be extremely rare that the minor version has to be increased, but the framework for how to handle it is in place to ensure that there is always a deployable and licensable version of xvc available, and that no patent holder (or "patent troll") would be able to block the codec from being used altogether.

The separation of major version and minor version makes it possible to always allow for a transition period when changes are introduced to the xvc codec. In the case of a major version increase, new bitstreams will only be created once decoders have been updated to support the new version. In the case of a minor version increase, new decoders will not replace old decoders until existing bitstreams have been updated to use the new minor version. By applying this scheme, the codec can evolve over time without causing any interoperability problems.

2.9. Temporal scalability and parallel decoding

The default coding structure of the xvc codec is a static hierarchical B-picture structure with 15 B-pictures. The length of the hierarchical structure, called SubGOP, is indicated in the Segment Header. A SubGOP length of 16 corresponds to 15 B-pictures in a dyadic hierarchy, forming four temporal layers above temporal layer zero.

Pictures in temporal layers above 0, only predict from temporal layer 0 or pictures with lower temporal layers within the same SubGOP. This makes it possible to scale of an arbitrary number of temporal layers in any SubGOP without affecting decodability of any pictures outside the same SubGOP. This is a very useful property in a software decoder since it makes it possible to adjust the decoding complexity in response to sudden changes in available processing resources. Instead of freezing the video and build up a delay, the framerate can momentarily be reduced and full framerate can be resumed as soon as enough processing resources are available.

A fixed hierarchical coding structure is also very useful in order to enable efficient parallel decoding. The xvc reference decoder includes support for picture level threaded decoding. When decoding bitstreams with SubGOP length 16 a speed-up factor of between 3x and 4x is typically achieved on a CPU with 4 active physical cores.

Providing support for temporal scalability and picture level parallel decoding generally comes at a very low cost in terms of compression performance. However, on very specific sequences there might be a compression penalty due to the constraint of not predicting from previously coded pictures in the same or higher temporal layers.

The picture level parallelism that is currently implemented in xvc can be combined with other tools for parallelism such as tiles, wavefronts or slices if a higher degree of parallelism is desired, but support for these tools have not yet been added to xvc.

3. The xvc reference software

The implementation of the xvc reference software has been made from scratch using C++11. Some of the advantages of C++11 compared to earlier versions of C++ is improved support for type management, improved pointer handling, availability of lambda expressions, threading and useful new keywords such as auto. The xvc implementation strictly follows the Google C++ Style Guide [cppStyle] which is a well established formatting recommendation for C++, intended to provide readability and consistency across various C++ project.

The low-complex design of the decoding process of xvc makes it possible to run the decoder efficiently on a large variety of devices. The xvc reference decoder has been demonstrated to run FullHD decoding in realtime on mobile devices, and a JavaScript xvc decoder, based on the reference xvc decoder, is able to run xvc decoding directly in browsers, as shown on the Divideon demo page [xvcDemo]

The total number of lines of code for the latest revision of the xvc software is around 24,000. This can be compared to the around 240,000 lines of code in the AV1 software.

4. Results

Experimental results for three different test cases are reported; All-Intra, single pass Random-Access and multi-pass Random-Access. The latter two correspond to "High Latency CQP" and "High Latency Unconstrained" from the Video Codec Testing and Quality Measurement I-D [Daede17]. The results have been generated using the

AreWeCompressedYet? framework [AWCY]. The full set of results are available at https://awcy.divideon.com [DAWCY].

For xvc [xvcSource], the commit f0b31546a3251aeb43f7928338b12aa349156139, from 2018-02-28 has been used, with command line parameters: -tune 1 -speed-mode 0 -chroma-qp-offset-u -1 -chroma-qp-offset-v -1. The chroma qp offset is used to better match bit distribution between luma and chroma relative to AV1. The quantizer values that have been used for xvc are: 20, 25, 30, 35, 40.

For AV1 [av1Source], the commit 1a7099443894d07fdf3acbb7e12ed04c2d62b9c5, from 2018-02-02 has been used, with build settings: -DCONFIG_EXPERIMENTAL=1 and --tune-content=screen as extra command line argument. The following quantizer values are used: 20, 32, 43, 55 and 63. For multi-pass coding the run "debargha-adopted-0104@2018-01-05T00:55:50.198Z" was copied from AreWeCompressedYet? [AWCY], since that was the most recent run that was finsihed at the time of this submission. That run was using commit 5a31bc899b308da9b2cacd339d0b19374a74b906 from "2018-01-04"

For HM [hmSource], version 16.17 from 2017-10-18 has been used with the default configuration files provided in the cfg folder of the source code repository. These configuration files correspond to the JCT-VC common test conditions described in JCTVC-Z1100 [hmCtc]. The following quantizer values are used: 20, 25, 30, 35, 40.

For All-Intra, results are presented for the Subset1 test set, and for the two Random-Access cases results are presented for the Objective-1-fast test set [Testset].

The tables below show the average percentual rate difference measured using the Bjontegaard rate difference, also known as BD-rate [Bjontegaard01]. The BD-rate is measured using the following objective metrics: PSNR, PSNR-HVS [Egiazarian2006], SSIM [Wang04] and MSSIM [Wang03].

4.1. All-Intra

In the tables below, results are presented for the Subset1 test set which consist of 50 different still images.

The following table present results of xvc relative to HM with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate).

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
Average	-9.3	-33.3	-30.3	-9.8	-9.9	-9.5

All-intra xvc relative to HM

The following table present results of xvc relative to AV1 with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate).

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
Average	-3.1	-7.4	-7.9	-2.1	-2.8	-3.0

All-intra xvc relative to AV1

4.2. Single pass Random-Access

The following table present results of xvc relative to HM with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate).

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
1080p	-16.8	-29.9	-28.9	-15.0	-17.9	-16.8
1080psc	-13.7	-44.5	-40.4	-15.4	-16.7	-17.0
720p	-20.8	-30.0	-32.6	-20.1	-23.8	-22.7
360p	-26.1	-24.7	-28.8	-26.4	-30.2	-29.6
Average	-19.5	-30.7	-31.3	-19.1	-22.0	-21.2

Single pass Random-Access xvc relative to HM

The following table present results of xvc relative to AV1 with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate). Results are presented for 360p and 720p only, since we were not able to run encodings for the complete test set for AV1 in time.

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
720p	-13.3	-0.8	-4.4	-16.4	-20.2	-20.5
360p	-19.7	-9.6	-3.4	-22.5	-23.0	-26.1
Average	-16.5	-5.2	-3.9	-19.4	-21.6	-23.3

Single pass Random-Access xvc relative to AV1

4.3. Multi-pass Random-Access

The following table present results of xvc relative to AV1 with numbers indicating the percentual bitrate difference for equivalent quality (negative numbers means that xvc reduces the bitrate). The results for AV1 are imported from AreWeCompressedYet? [AWCY].

	PSNR	PSNR Cb	PSNR Cr	PSNR HVS	SSIM	MS SSIM
1080p	-6.0	-4.5	-3.2	-5.6	-10.9	-9.7
1080psc	8.3	18.5	15.8	5.9	6.2	3.9
720p	-0.5	0.4	4.8	-1.4	-6.0	-5.5
360p	-15.9	-6.2	11.2	-19.9	-19.0	-21.2
Average	-5.1	-0.7	4.6	-6.4	-9.4	-9.6

Multi-pass Random-Access xvc relative to AV1

4.4. Computational complexity

The single pass Random-Access configurations have been run on the same platform (Intel Xeon E5-2660). The encoding times and decoding times can be used to estimate the computational complexity when running that particular setting on that particular hardware. On average the encoding time is around 340% higher for AV1 than for xvc. The decoding time is around 70% higher for AV1 than for xvc.

5. Discussion

The xvc codec has been developed with the aim of creating the most efficient video codec that can be made available under a single reasonable license. Different coding tools have been included based solely on their technical merits. Technology is replaced or removed only if the patent holder of that particular technology has requested for it to be removed, or if the patent holder by other means have made it clear that they intend to seek royalties for the technology outside of the xvc licensing program or institute patent litigation

related to the technology. This approach has made it possible to make use of the best available technology rather than having to compromise, exclude and/or design around technology to create a lesser efficient alternative.

Divideon is in continuous dialog with potential patent holders and has recently sent out a Call for Patents in xvc [xvcCall], to make sure that the license actually covers all the necessary technology to use, implement and distribute conforming xvc implementations. As with all modern video codecs it is not possible to give a definite answer to the patent situation, but what is unique with xvc is the well-defined framework for handling third party patent assertions.

In the Call for Patents in xvc that was sent out by Divideon there was also a request for feedback regarding the idea of defining a royalty-free profile of xvc. Hopefully, it will be possible to create a "safe" and free baseline profile of xvc, simply by requiring certain restriction flags to be set equal to one in that profile.

The xvc codec is brought to the NETVC Working Group as a candidate for the Internet Video Codec. The objectives of the Working Group states that "This WG is chartered to produce a high-quality video codec that meets the following conditions:

1. Is competitive (in the sense of having comparable or better performance) with current video codecs in widespread use.

2. Is optimized for use in interactive web applications.

3. Is viewed as having IPR licensing terms that allow it to be widely implemented and deployed."

The authors of this document believe that xvc is well positioned to fulfill all of these conditions. A framework with a well-defined single license and possibly a royalty-free baseline profile ensures that condition 1 and 3 can both be fulfilled, and the reference software of xvc constitutes a good example of that the codec (and especially the decoder) can be run in realtime on common hardware including mobile devices, and thereby fulfill the second condition.

6. IANA Considerations

This document has no IANA considerations.

7. Security Considerations

This document has no security considerations.

8. Informative References

[av1Source]
Alliance for Open Media, "The av1 source code", n.d., <https://aomedia.googlesource.com/aom/>.

[AWCY] Xiph.Org Foundation, "Are We Compressed Yet?", n.d., <https://arewecompressedyet.com>.

[Bjontegaard01]
Bjontegaard, G., "Calculation of average PSNR differences between RD-curves", 2001, <http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc>.

[cppStyle]
Google, "Google C++ Style Guide", n.d., <https://google.github.io/styleguide/cppguide.html>.

[Daede17] Daede, T., Norkin, A., and I. Brailovsky, "Video Codec Testing and Quality Measurement", IETF NETVC Internet-Draft draft-ietf-netvc-testing-06, October 2017.

[DAWCY] Xiph.Org Foundation, "Are We Compressed Yet? Divideon clone", n.d., <https://awcy.divideon.com>.

[Egiazarian2006]
Egiazarian, K., Astola, J., Ponomarenko, N., Lukin, V., Battisti, F., and M. Carli, "Two new full-reference quality metrics based on HVS", Proceedings of the Second International Workshop on Video Processing and Quality Metrics for Consumer Electronics VPQM, January 2006.

[HEVC] ISO/IEC/ITU-T, "High efficiency video coding", n.d., <https://www.itu.int/rec/T-REC-H.265>.

[hmCtc] Sharman, K. and K. Suehring, "Common test conditions", 2017, <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>.

[hmSource]
ISO/IEC/ITU-T, "The HM source code", n.d., <https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/>.

[Testset] Daede, T., "Test Sets", n.d., <https://people.xiph.org/~tdaede/sets/>.

[Wang03] Wang, Z., Simoncelli, E., and A. Bovik, "Multiscale structural similarity for image quality assessment", The 37th Asilomar Conference on Signals, Systems Computers Volume 2, November 2003.

[Wang04] Wang, Z., Bovik, A., Sheikh, H., and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", issn 1057-7149, IEEE transactions on image processing Volume 13, number 4, April 2004.

[Wenger16] Wenger, S., "On profiles and per-feature Flags", 2016, <http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=10492>.

[xvcCall] Divideon, "Call for Patents in xvc", 2018, <http://www.releasewire.com/press-releases/call-for-patents-in-xvc-929388.htm>.

[xvcDemo] Divideon, "Mobile video streaming with xvc", 2018, <https://www.divideon.com/products-and-services/mobile-video-streaming-with-xvc/>.

[xvcio] Divideon, "The xvc web page", n.d., <https://xvc.io/>.

[xvcLicense] Divideon, "The xvc license", n.d., <https://xvc.io/license>.

[xvcSource] Divideon, "The xvc source code", n.d., <https://github.com/divideon/xvc/tree/dev>.

[xvcWp] Samuelsson, J. and P. Hermansson, "Introducing xvc - a Divideon white paper", 2017, <https://www.divideon.com/wp-content/uploads/2017/09/Introducing-xvc-a-Divideon-whitepaper-v1.0.pdf>.

Authors' Addresses

Jonatan Samuelsson
Divideon
Kulstoetarvaegen 14
Stockholm 12240
Sweden

Email: jonatan.samuelsson@divideon.com

Per Hermansson
Divideon
Kulstoetarvaegen 14
Stockholm 12240
Sweden

Email: per.hermansson@divideon.com