INTEGRATING THOR TOOLS INTO THE EMERGING AV1 CODEC

S. Midtskogen, A. Fuldseth, G. Bjøntegaard

Cisco Systems, Inc.,
Philip Pedersens vei 1, 1366 Lysaker, Norway

T. Davies

Cisco Systems, Inc.,
Bedfont Lakes, Feltham,
Middlesex TW14 8HA, United Kingdom

ABSTRACT

Over recent years there have been several efforts which aim to standardise a royalty-free video codec, such as Thor developed by Cisco, and AV1 developed by the Alliance for Open Media. In this paper we discuss how some compression tools in Thor were integrated into the emerging AV1 codec aiming to increase compression efficiency as well as to decrease computational complexity.

***Index Terms*—** Video compression, royalty-free, AV1, Thor, low complexity

1. INTRODUCTION

While video compression technologies have improved over recent years as seen in the HEVC/H.265 [1] standard issued jointly by ITU-T and ISO, the deployment of these standards may have been delayed or restricted due to their licensing terms, in particular for applications relying on wide and low-cost deployment, for instance in web browsers and mobile phones. For this reason the NETVC working group [2] was formed within IETF in March 2015 to develop a royalty-free video codec. NETVC is currently working on two proposals: the Thor codec [3, 4] developed by Cisco, and the Daala codec [5] developed by Mozilla. A similar, parallel effort was begun in September 2015 with the formation of the Alliance for Open Media (AOMedia) [6] with the ongoing development of the AV1 codec [7] which is largely based on Google's VP9 codec [8] with tools from Thor and Daala being integrated along with new tools and improvements.

Since VP9 lacks a deringing loop filter (such as the sample adaptive offset in HEVC), Thor's constrained low-pass filter (CLPF) was early identified as a candidate for inclusion in AV1. Several other tools in Thor were considered, and those which have so far also been successfully integrated are quantisation matrices and improved filter coefficients for the sub-pixel motion compensation.

At the time of writing, January 2017, the AV1 codec is still in development and its specification is yet not finalised.

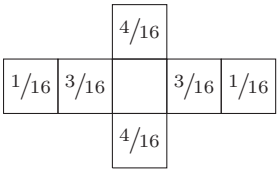

Fig. 1: The R neighbourhood and the values of $a(i, j)$

2. CONSTRAINED LOW-PASS FILTER

In AV1 CLPF is applied after the deblocking filter, as in Thor. The purpose of CLPF is to reduce ringing artefacts and improve overall image quality. It has some similarities with SAO [9] in HEVC.

CLPF is a non-linear filter which modifies a pixel $x(m, n)$ into $x'(m, n)$ at position m, n:

$$x'(m,n) = x(m,n) + \sum_{i,j \in R} a(i,j) f(x(m+i, n+j) - x(m,n), s)$$

where $f(x, y)$ returns $-y$ if $x < -y$, y if $x > y$ and x otherwise, s is the filter strength, R is the neighbourhood of $x(m, n)$ and $a(i, j)$ has the values shown in figure 1.

If $x(m + i, n + j)$ is outside the frame, it will have the value of the closest pixel inside the frame instead. The effect of the equation is to make x, constrained by the clip function, more like its neighbours.

The image to be filtered by CLPF is divided into filter blocks (FB), whose size does not depend on any other block divisions of the frame like superblock sizes, prediction block sizes or transform sizes. The filter is selectably applied to qualified FBs, or to every qualified FB in a frame if so indicated in the frame header, in which case no further signalling per FB is done. The FB size is 32×32, 64×64 or 128×128, signalled in the frame header. FBs must contain at least one non-skip block to be qualified for selective filtering if the FB size is 32×32 or 64×64. If the FB size is 128×128, the block is always qualified. When there is no signalling at FB level, the filter is applied to every non-skip block. The chroma planes never have signalling at FB level even if blocks are filtered selectably in the luma plane.

The filter can have three different strengths: 1, 2 or 4 for 8

pos	-2	-1	0	1	2	3
$0/16$	0	0	64	0	0	0
$1/16$	1	−3	63	4	−1	0
$2/16$	1	−5	61	9	−2	0
$3/16$	1	−6	58	14	−4	1
$4/16$	1	−7	55	19	−5	1
$5/16$	1	−7	51	24	−6	1
$6/16$	1	−8	47	29	−6	1
$7/16$	1	−7	42	33	−6	1
$8/16$	1	−7	38	38	−7	1

Table 1: Standard filter coefficents

pos	-3	-2	-1	0	1	2	3	4
$0/16$	0	0	0	64	0	0	0	0
$1/16$	−1	1	−3	63	4	−1	1	0
$2/16$	−1	3	−6	62	8	−3	2	0
$3/16$	−1	4	−9	60	13	−5	3	−1
$4/16$	−2	5	−11	58	19	−7	3	−1
$5/16$	−2	5	−11	54	24	−9	4	−1
$6/16$	−2	5	−12	50	30	−10	4	−1
$7/16$	−2	5	−12	45	35	−11	5	−1
$8/16$	−2	6	−12	40	40	−12	6	−1

Table 2: Sharp filter coefficents

bit content. For higher bit depths the strengths are scaled accordingly, so for 10 bit they become 4, 8 and 16. The strength for each plane is signalled in the frame header, so all qualified FBs in the same plane get filtered with the same strength.

CLPF was designed to offer a favourable trade-off between complexity and compression efficiency. An implementation for ARM/NEON (armv7) using SIMD intrinsics requires 4.9 instructions per pixel to filter an 8×8 block [10].

3. IMPROVED FILTER COEFFICIENTS

For motion compensation at sub-pixel resolution Thor uses 6 tap filters with 7 bit coefficients, whereas AV1 originally used 8 tap filters with 8 bit coefficients like VP9. Further differences from Thor include that AV1 has three different filters (standard, sharp, smooth) selectable at frame or prediction block level, whereas Thor has one filter if bi-prediction is enabled in the sequence header and another otherwise, and a special non-separable filter for the centre sub-pixel position somewhat resembling the smooth filter in AV1.

The standard filter in AV1 was changed to use the filter for uni-predicted frames in Thor. The sharp filter retained 8 taps, but the coefficients were modified and reduced to 7 bit, and the smooth filter was reduced to 6 taps and 7 bit. Additional phases for all filters had to be designed to account for the higher sub-pixel resolution used in AV1 ($1/16$ pixel resolution in chroma versus $1/8$ pixel resolution in Thor). No changes were made to the filters other than to replace the coefficients.

By reducing the number of taps, the number of multiplications and additions needed the filtering process is reduced. The reduction from 8 to 7 bits per coefficient ensures that the sums, after having applied the filter in one direction, stay within 16 bits for 8 bit input. Unlike Thor, AV1 shifts and clamps the sums to 8 bit after the first direction. The change from 8 to 7 bits can significantly reduce the complexity increase of a possible future proposal to keep the intermediate sums at full resolution.

pos	-2	-1	0	1	2	3
$0/16$	0	0	64	0	0	0
$1/16$	1	14	31	17	1	0
$2/16$	0	13	31	18	2	0
$3/16$	0	11	31	20	2	0
$4/16$	0	10	30	21	3	0
$5/16$	0	9	29	22	4	0
$6/16$	0	8	28	23	5	0
$7/16$	−1	8	27	24	6	0
$8/16$	−1	7	26	26	7	−1

Table 3: Smooth filter coefficents

The new filter coefficients are listed in tables 1, 2 and 3. There is symmetry around the middle position, so position $9/16$ uses the coefficients in position $7/16$ reversed and so on.

4. QUANTISATION MATRICES

Thor supports quantisation weighting matrices to vary the quantiser step size for different frequencies which may increase quality using certain metrics like FAST-SSIM and PSNR-HVS, but decrease quality using other metrics like PSNR. Originally, AV1 had flat weights except for the DC coefficient which was given a smaller quantiser step size.

Inverse quantisation weighting matrices (forward quantisation is non-normative) similar to those in Thor were introduced in AV1. A fixed matrix set is shared between the encoder and decoder containing matrices for each combination of: transform block size (4×4, 8×8, 16×16, 32×32), component type (luma, chroma) and block type (intra, inter). Only one set is allowed per frame, and is selected based on the base quantisation index for that frame. The quantisation index value is mapped to select one of 16 matrix sets in a configurable way so that the aggressiveness of the weighting can be controlled. The matrices become flatter as the quantisation index value increases (and the quality decreases). Inter matrices are slightly flatter than intra matrices.

Thor has different matrices for the two chroma components, and these were averaged so they could be shared by the chroma components in AV1. The reason for this was mainly to limit the amount of changes to the AV1 codebase. This also reduces the amount of memory needed for the matrices, which may be particularly useful in hardware implementations, but it may also degrade the chroma quality slightly.

Adding quantisation weighting matrices has a small impact on complexity. For 8 bit video, it was necessary to use similar internal bit widths as for high bit depth video, with the resolution of the multiplications in inverse quantisation increased from 32 to 64 bits.

5. RESULTS

The results in this section were compiled using the *Are We Compressed Yet* tool (AWCY) [11], which has been selected by AOMedia as the preferred way to measure compression efficiency of AV1. It offers a selection of different metrics measured by the Bjøntegaard Delta Rate [12]. Negative figures indicate bitrate reductions.

The development of AV1 is an ongoing effort, so the compression impact of every tool will vary somewhat over time. The results presented here can be reproduced using git SHAs `456e086` (CLPF), `ab44fd1` (interpolation filters) and `7386eda` (quantisation matrices) and the **objective-1-fast** [13] test set in AWCY.

Results for CLPF

In addition to CLPF and the deblocking filter there are two other loop filter proposals for AV1: *dering* from Mozilla, which is similar to CLPF but also looks for directional patterns in the image, and *loop restoration* from Google, which has some similarities with the ALF [14] proposal for HEVC. At the time of writing it had not yet been decided which one(s) AV1 will adopt. AV1 can be configured to run both CLPF and *dering* in which case it will apply *dering* before CLPF.

CLPF gives better results in low delay configurations than in high delay configurations. The effectiveness of the filter is also better in complexity constrained configurations. Since CLPF was designed to offer a good compromise between complexity and compression, the complexity changes are also shown for the encoder and decoder in a complexity constrained configuration. AWCY does not show decoding time, and the complexity change for the decoder was calculated by encoding at medium complexity by using the options `--cq-level=42 --passes=1 --end-usage=q --lag-in-frames=0` for the low delay configuration and the same for the high delay configuration except `--passes=2 --lag-in-frames=25 --auto-alt-ref=2` and then decoding the streams counting the executed instructions using *valgrind* [15]. The decoder was compiled and run on an SSE4.2 enabled x86_64 CPU.

It can be noted that on the encoder side the complexity slightly *decreases* when CLPF is enabled. The likely cause is that the filters reduce the bitrate and thereby decrease the remaining encoder complexity. We have not investigated whether the large complexity increase from the *loop restoration* is mainly a result of an inefficient implementation or intrinsic complexity of the filter.

The results are summarised in tables tables 4 to 8. Smaller gains can be seen for the high delay configurations which enables bi-prediction. Bi-prediction averages pixels from two frames and thereby introduces a smoothing which makes CLPF less efficient.

filter \ metric	PSNR	PSNR HVS	SSIM	CIEDE 2000	APSNR	MS SSIM
CLPF	−2.79%	−1.65%	−2.21%	−3.01%	−2.76%	−1.80%
dering	−2.59%	−1.86%	−2.50%	−2.13%	−2.54%	−1.98%
dering+CLPF	−3.61%	−2.31%	−3.06%	−3.39%	−3.56%	−2.45%
restoration	−4.19%	−1.77%	−3.12%	−3.32%	−4.14%	−1.79%

Table 4: Compression gains of CLPF and other AV1 loop filters. Low delay configuration.

filter \ metric	PSNR	PSNR HVS	SSIM	CIEDE 2000	APSNR	MS SSIM
CLPF	−1.16%	−0.41%	−0.84%	−1.42%	−1.18%	−0.50%
dering	−1.40%	−0.77%	−1.33%	−1.05%	−1.40%	−0.89%
dering+CLPF	−1.62%	−0.61%	−1.31%	−1.56%	−1.63%	−0.75%
restoration	−2.44%	−0.71%	−1.87%	−1.89%	−2.44%	−0.76%

Table 5: Compression gains of CLPF and other AV1 loop filters. High delay configuration.

filter \ metric	PSNR	PSNR HVS	SSIM	CIEDE 2000	APSNR	MS SSIM
CLPF	−5.93%	−3.92%	−5.86%	−5.87%	−5.85%	−4.40%
dering	−5.25%	−4.20%	−5.89%	−4.56%	−5.15%	−4.63%
dering+CLPF	−7.25%	−5.11%	−7.35%	−6.91%	−7.14%	−5.64%
restoration	−7.36%	−3.98%	−6.87%	−6.26%	−7.27%	−4.29%

Table 6: Compression gains of CLPF and other AV1 loop filters. Low delay and medium complexity (cpu-used=4) configuration.

filter \ metric	PSNR	PSNR HVS	SSIM	CIEDE 2000	APSNR	MS SSIM
CLPF	−2.95%	−1.50%	−3.14%	−3.24%	−2.99%	−2.04%
dering	−3.01%	−2.04%	−3.49%	−2.70%	−3.02%	−2.46%
dering+CLPF	−3.69%	−1.98%	−3.93%	−3.72%	−3.71%	−2.54%
restoration	−4.17%	−1.62%	−4.20%	−3.72%	−4.23%	−2.13%

Table 7: Compression gains of CLPF and other AV1 loop filters. High delay and medium complexity (cpu-used=4) configuration.

filter \ conf.	low delay enc.	high delay enc.	low delay dec.	high delay dec.
CLPF	−0.60%	−0.62%	8.64%	5.73%
dering	1.63%	0.72%	22.1%	16.7%
dering+CLPF	1.84%	0.76%	26.8%	19.8%
restoration	385%	346%	264%	239%

Table 8: Complexity changes of CLPF and other AV1 loop filters. High delay and medium complexity (cpu-used=4) configuration.

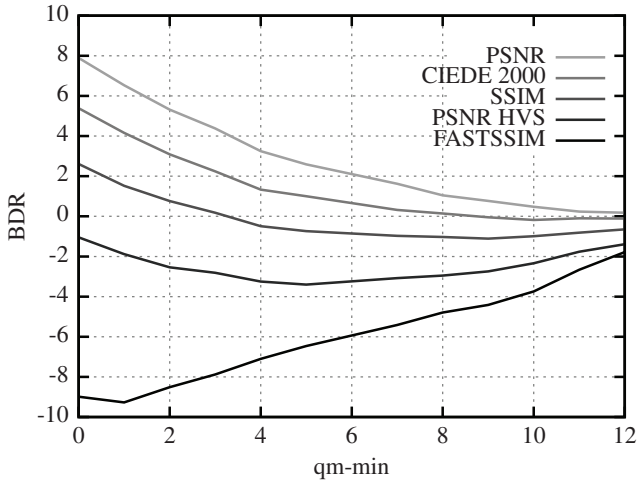

Fig. 2: BDR results for weighted quantisation matrices.

Results for quantisation matrices

Figure 2 summarises the effects of weighting matrices measured using different metrics. The matrices offer a way to control a trade-off between results using different metrics. They generally favour FASTSSIM and PSNR HVS and to some degree SSIM, but make CIEDE 2000 and PSNR worse. APSNR and MS SSIM are not shown in figure 2, but they closely follow PSNR and PSNR HVS respectively. The "strength", or flatness of the matrices, is most easily controlled by the *qm-min* parameter. A *qm-min* of 0 means that all 16 matrix sets may be used, while a *qm-min* of 12 means that only the four flattest matrix sets will be used.

Results for improved filter coefficients

The compression gains for the new filter coefficients are shown in table 9. The new filter coefficients give small, but consistent gains in the low delay configuration. In the high delay configuration there is little change. This is still an acceptable result since the main motivation for changing the coefficients was simplification.

AV1 shifts and clamps the sums to 8 bit after the first horizontal pass of the filter, then again after the vertical pass. Since the lower resolution of the new coefficients makes it less costly to avoid the precision loss between the passes, the combined effect of the new coefficients and full precision between the passes was also examined and is shown in table 10.

conf. \ metric	PSNR	PSNR HVS	SSIM	CIEDE 2000	APSNR	MS SSIM
low delay	−0.40%	−0.56%	−0.62%	−0.31%	−0.40%	−0.59%
high delay	−0.02%	−0.04%	−0.10%	0.13%	−0.03%	−0.07%

Table 9: Results with precision loss between the filter passes.

conf. \ metric	PSNR	PSNR HVS	SSIM	CIEDE 2000	APSNR	MS SSIM
low delay	−0.98%	−1.02%	−1.41%	−1.58%	−0.98%	−1.24%
high delay	−0.45%	−0.43%	−0.62%	−0.89%	−0.46%	−0.43%

Table 10: Results without precision loss between the filter passes.

6. REFERENCES

[1] ITU-T Rec. H.265 and ISO/IEC 23008-2 (2013) High efficiency video coding

[2] https://datatracker.ietf.org/doc/charter-ietf-netvc/

[3] Bjøntegaard, G., Davies, T., Fuldseth A., and Midtskogen, S., "The Thor video codec", Data Compression Conference 2016

[4] Thomas Davies ; Gisle Bjøntegaard ; Arild Fuldseth and Steinar Midtskogen, "Recent improvements to Thor with emphasis on perceptual coding tools ", proc. SPIE 9971, Applications of Digital Image Processing XXXIX, 997118 (September 27, 2016)

[5] https://www.xiph.org/daala/

[6] http://aomedia.org

[7] https://aomedia.googlesource.com/aom

[8] Adrian Grange, Peter de Rivaz, Jonathan Hunt, "VP9 Bitstream & Decoding Process Specification" (v0.6, draft), http://www.webmproject.org/vp9/

[9] C. M. Fu, E. Alshina, A. Alshin, Y. W. Huang, C. Y. Chen, C. Y. Tsai, C. W. Hsu, S. M. Lei, J. H. Park, W. J. Han, "Sample adaptive offset in the HEVC standard", IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1755-1764, December 2012

[10] Steinar Midtskogen, "Thor update", NETVC proceedings, IETF95, Buenos Aires, Argentina, April 2016, https://www.ietf.org/proceedings/95/slides/slides-95-netvc-1.pdf

[11] https://arewecompressedyet.com

[12] Bjøntegaard, G., "Improvements of the BD-PSNR model", ITU-T Q.6/SG16 VCEG, VCEG-AI11, Berlin, Germany, July 2008

[13] T. Daede, A. Norkin, I. Brailovskiy, "Video Codec Testing and Quality Measurement", https://tools.ietf.org/html/draft-ietf-netvc-testing-05

[14] Ching-Yeh Chen, Chia-Yang Tsai, Yu-Wen Huang, Tomoo Yamakage, In Suk Chong, et al., "The adaptive loop filtering techniques in the HEVC standard", Proc. SPIE 8499, Applications of Digital Image Processing XXXV, 849913, October 15, 2012

[15] http://valgrind.org