HARDWARE-FRIENDLY INTER PREDICTION TECHNIQUES FOR AV1 VIDEO CODING

Zhipin Deng, zhipin.deng@intel.com
Iole Moccagatta, iole.moccagatta@intel.com

Intel Corporation

ABSTRACT

The Alliance for Open Media (AOMedia) aims at generating a new, royalty-free video codec specification named AV1 that is scheduled for completion in the 2nd half of 2017. To facilitate AV1 market acceptance and rapid deployment, attention has been paid toward development of hardware friendly coding tools. This paper presents the following techniques to facilitate hardware implementation of AV1 inter prediction: 1) a simplified motion vector prediction scheme to reduce memory footprint / bandwidth; 2) methods to limit / remove the dependency between the computation of probability context model and the reconstruction of motion vectors to improve hardware throughput. Experimental results on a wide range of test clips show negligible quality impact for 1) and minor quality impact for 2) while the motion vectors line buffer memory footprint is reduced by 50%, motion vectors local memory is reduced by 33%, and the throughput of the arithmetic decoder remains unchanged.

Index Terms — reference motion vector, probability context model, hardware implementation, AV1, VP9.

1. INTRODUCTION

The Alliance for Open Media (AOMedia) is a non-profit organization whose first project is to develop an open and royalty-free video codec. The first generation of the AOMedia Video codec, called AV1, is scheduled for completion by the 2nd half of 2017.

During the AV1 development, a number of tools have been proposed to improve inter prediction. Among them, a scheme to exploit neighboring temporal predictor and refine block-based motion compensation for a better representation of motion model in a real life scenario; a dynamic reference motion vector coding (REFMV) to utilize all the available motion information from previously coded neighboring blocks to improve the inter mode coding efficiency [3]; 10-tap and 12-tap interpolation filters in addition to traditional 8-tap filter; global motion and warped motion schemes for a better prediction efficiency, and non-squares inter prediction shapes.

This paper presents several methods to reduce AV1 inter prediction hardware implementation complexity measured in terms of memory footprint/bandwidth and gate count. It also proposes methods to limit or remove the dependency between the computation of probability context model and the motion vector reconstruction in a video decoder pipeline for the purpose of improving throughput. Some of the proposed methods have been integrated into the AV1 codebase. Experimental results demonstrate that these methods have negligible quality impact on the coding gain but substantial hardware design advantages.

2. MOTION VECTOR PREDICTIONS IN AV1

At its starting point, AV1 used the same motion vector prediction as VP9, referred to as AV1 baseline in the following. Along the way, a new coding tool, REFMV [3], has been proposed to enhance the coding efficiency of inter prediction. In this coding tool, a RefMV mode is implemented to replace the NearestMV mode and NearMV mode in AV1 baseline. A length-adaptively motion vector candidate list is generated using available motion vectors from the neighboring four rows above and neighboring four left columns. These neighboring motion vectors are also referred to as reference motion vectors in the motion vector prediction literature and in the following.

Because the reference motion vectors are computed when decoding previous blocks, they must be stored in memory, either line buffer or local buffer, depending on their spatial location with respect to the currently decoded block.

In the case of REFMV, the reference motion vectors from both 8x8 and 4x4 neighboring blocks should be stored in memory for the motion vector candidate list generation. The generated motion vector candidate list is used to choose the best motion vector predictor by using RDO optimization on the encoder side, and finally a ref-mv candidate index pointing to the selected motion vector predictor in the list is transmitted to the decoder.

In addition, the REFMV coding tool changes the context based probability model selection for motion vector entropy coding with respect to AV1 baseline. Specifically, in AV1 baseline, the probability model selection depends on the prediction modes of left and top neighbor blocks, whereas in REFMV the probability model of inter prediction depends

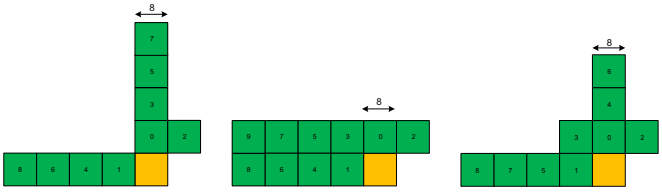
on the block's motion information. For example, to select the probability context model of a specific inter mode and that of the RefMV mode's candidate index, the arithmetic decoder needs to know the reconstructed motion vectors in the prediction list, in addition to their corresponding prediction block sizes and modes. In the case of hardware pipelined video decoders, this dependency may cause the entropy decoder to stall waiting for data from the back-end of the video decoder pipeline. In state-of-the-art arithmetic decoders, the probability context model is computed using information from blocks neighboring to the one currently decoded. Some of this information could be known to the arithmetic decoder – for example previously arithmetic decoded inter prediction modes. Other could be only computed by different units of the video decoder hardware pipelines – for example reconstructed motion vectors and corresponding prediction blocks' size of blocks which are neighbors of the one currently decoded. For the case when the information could be only computed by other units of the video decoder hardware pipelines, the video decoder pipeline is usually generating such information with same latency. Because of such latency, the arithmetic decoder will be stalled, waiting for the information to be generated. This stall limits the throughput/performance of the arithmetic decoder, which is the major bottleneck to achieve overall decoding high speed.

3. PROPOSED HARDWARE-FRIENDLY MOTION VECTOR PREDICTION IN AV1

3.1. Line buffer reduction for motion vector candidate derivation

The REFMV tool requires four line buffers to store all possible 8x8 neighbor block reference motion vectors (in green), as shown in Figure 1 (a) – where the size of the current inter predicted block (in orange) is 8x8. To reduce the size of the line buffers used in REFMV, different motion vector candidate search patterns have been tested, including using only candidates in the first top row – left side, and using candidate in three top rows, as illustrated in Figure 1 (b) and Figure 1 (c), respectively. Experimental results have shown that the search pattern in Figure 1 (c) achieves the best trade-off between coding efficiency and memory footprint.

In addition, 4x4 block motion vectors in REFMV coding tool has been removed, as shown in Figure 2.



(a) (b) (c)

Figure 1. An example of the line buffer reduction for an 8x8 block (in orange): (a) original search pattern in REFMV; (b) search pattern using one row buffer; (c) search pattern with best trade-off between coding gain and memory footprint.

(a) (b)

Figure 2. The spatial neighbor list (in green) for a 4x4 block (in orange): (a) original candidate list in REFMV coding tool; (b) proposed candidate list to remove 4x4 block motion vector dependencies.

In summary, the proposed line buffer reduction technique consists of:

1) Reducing spatial neighboring top rows in REFMV coding tool from 4 to 3.
2) Removing the use of 4x4 block motion vectors in REFMV coding tool, which results in removing two 4x4 block motion vectors in first row above and two 4x4 block motion vectors in first column on the left.
3) Adding one extra candidate in the first top row – left side (candidate "3" in Figure 1 (c)) compared to REFMV, in order to achieve better prediction results.
4) Removing predicted motion vectors from the motion vectors' context computation. The REFMV proposal uses both predicted and reconstructed motion vectors of the selected candidate to compute the context of each residual motion vector. Therefore, the REFMV coding tool requires both prediction and reconstructed motion vectors to be stored for each candidate motion vector. The proposed method uses only the reconstructed motion vectors of the selected candidate to compute the context of the residual motion vector, thus it removes the need to store predicted motion vectors for each spatial and temporal neighbor.

3.2. Relax/remove the context model dependency

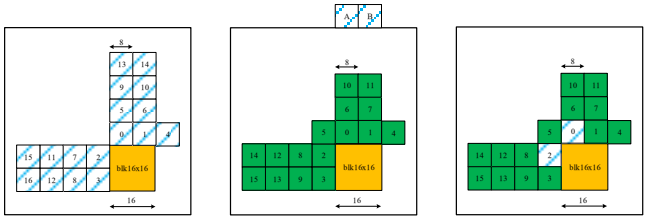
The REFMV coding tool changes how the probability context model of inter modes is selected compared to AV1 baseline. In the REFMV coding tool, the arithmetic decoder needs to know the reconstructed candidate motion vectors in the motion prediction list, as well as their corresponding prediction block sizes and prediction modes. This creates a dependency that may causes the entropy decoder to stall waiting for data from the back-end of the video decoder hardware pipeline. This section proposes methods to limit or remove this dependency for the purpose of improving the throughput of the arithmetic decoder.

In the first method, named "Method #1", the context model used to arithmetic decode the inter mode of the current block

uses only information which is outside the current block's super-block. In the case of REFMV tool, this information is the reconstructed motion vector, the prediction mode, the block size, etc. A number of hardware decoder's pipelines are super-block based, where super-blocks are defined either by the video codec specifications, of by the architecture of the decoder implementation. A super-block based pipeline means that the pipeline operates in unit of super-blocks. If the data needed for the current block's arithmetic decoding is outside the current super-block, such data has been already decoded and it is available by the time the entropy decoder needs it to compute the context models of the current super-block. Therefore, the arithmetic decoder is not stalled.

Figure 3 (b) describes an example of how Method #1 is applied. The current 16x16 block (in orange) belongs to a super-block. The spatial neighbor blocks "0" to "15" (in green) are inside the current super-block and are of size 8x8. The blocks "A" and "B" (in blue pattern) are outside the current super-block. The motion vectors of block "A" and "B" are used to compute the context model used by the arithmetic decoder to decode the bits representing the mode of the current block. Therefore, the dependency between the computation of probability context model and the reconstruction of motion vectors in a video decoder hardware pipeline is reduced / relaxed.

In the second method, named "Method #2", such dependency is completely removed by replacing data which is reconstructed by other units of the video decoder hardware pipelines with data generated by the arithmetic decoder. Because this data is generated by the arithmetic decoder, it can be stored locally and it can be accessed as soon as it is needed. In this case, the data generated by the arithmetic decoder and used by the same arithmetic decoder to compute the probability context model consists of the inter mode of neighboring blocks. Figure 3 (c) describes an example of how Method #2 is applied. The current 16x16 block (in orange) belongs to a super-block. The spatial neighbor blocks "0" to "15" are inside the super-block and used to generate the motion vector candidate list. However, only the prediction modes of block "0" and "2" (in blue pattern) are used to compute the probability context model used by the arithmetic decoder to decode the bits representing the mode of the current 8x8 block. As the prediction modes of block "0" and "2" are already decoded and accessible, the arithmetic decoder does not stall. This improves the throughput of the original REFMV design, and therefore the overall throughput of the hardware decoder.



(a) (b) (c)

Figure 3. The context model dependency between the entropy decoder and motion vector reconstruction: (a) original REFMV coding tool, in which the reconstructed motion vectors of all the 16 spatial neighbor blocks (in blue pattern) of current 16x16 block (in orange) are required to compute the context model; (b) Method #1 to relax the context model dependency, in which only the reconstructed motion vectors of block "A" and "B" (in blue pattern) outside the current super-block are required to compute the context model; (c) Method #2 to remove the context model dependency, in which the coding modes of 8x8 neighbor in the top row and left column of the current 16x16 block are needed to compute the context model.

4. EXPERIMENTAL RESULTS

The proposed methods were implemented in the AV1 codebase [4] and tested using the objective-1-fast test data set [5] which is used in the AV1 development process. All experiments were run according to the AV1 common test conditions [6], and using QPs = [22, 32, 43, 55]. Simulations were also run in AWCY [7], and generated consistent results.

4.1. Line buffer reduction

The proposed line buffer reduction technique described in Section 3.1 has been implemented in the AV1 codebase (git commit 4c98c4). The bit-rate reduction of the proposed technique integrated with the REFMV coding tool is shown in Table 1. See [6] for details of the AV1 High and Low Delay testing conditions.

The Bjøntegaard-Delta bit-rate (BD-rate) [11] measurement method utilized in the AV1 common test conditions has been used to generate all results shown in this paper. The BD-rate computes the bitrate change, either reduction or increase, compared to an anchor while maintaining the same quality as measured by objective metrics. The bitrate change is computed in percent. Negative values indicate bit rate reduction, and positive values indicate bit rate increase for the same quality metric. The objective quality metrics used are PSNR, PSNR-HVS [8], SSIM [9], and MS-SSIM [10].

Table 1 shows small average BD-rate gains for both High Delay and Low Delay testing conditions when the anchor is the original REFMV. For comparison, the BD-rate of the

original REFMV coding tool using the AV1 baseline as anchor is shown in Table 2.

The impact on hardware implementation of the proposed line buffer reduction technique is to remove two motion vectors of 4x4 blocks in the first row, and one motion vector of 8x8 block in the fourth row. So, the proposed method required three motion vectors for each 8x8 blocks, and the memory/memory bandwidth saving is (6-3)/6 = 50% (six motion vectors stored per each 8x8 block in the original REFMV coding tool vs. three motion vectors stored for each 8x8 block in the proposed technique). Furthermore, two motion vectors of 4x4 blocks are removed from the left column buffers. So, now the first column buffer contains only one motion vector of 8x8 block, and the second, third, and fourth columns still contains one motion vector of 8x8 block each, respectively. Therefore, the local memory/gate count saving to store the motion vectors is (6-4)/6 = 33% (six motion vectors stored per each 8x8 block in the original REFMV coding tool, and four motion vectors stored for each 8x8 block in the proposed technique).

4.2 Relaxing / removing context model dependency

The Method #1 and Method #2 techniques described in Section 3.2 have also been implemented in the same AV1 codebase and tested using the same test sequences and test conditions as above.

Simulation results of the two techniques when applied to relax or remove the context model dependency in REFMV coding tool are listed as Table 3 and Table 4, respectively.

Table 3 shows that relaxing the dependency results in an average 0.74% BD-rate PSNR loss for Low Delay and average 0.43% BD-rate PSNR loss for High Delay testing conditions compared to the original REFMV coding tool.

Table 4 shows that removing completely the dependency has a similar average BD-rate PSNR loss for Low Delay, and 0.56% BD-rate PSNR loss for High Delay. This quality impact assures that the REFMV proposal has no negative impact on the speed of an AV1 hardware arithmetic decoder.

Subtracting the quality impact of Table 3 and 4 from the quality improvements of REFMV shown in Table 2 shows that that overall average bitrate reduction of REFMV w/o any impact on an AV1 hardware arithmetic decoder's speed is -1.14% BD-rate PSNR gain for Low Delay and -0.70% BD-rate PSNR gain for High Delay.

5. CONCLUSIONS

This paper presents a number of methods to facilitate hardware implementation of inter prediction coding tools in the upcoming AV1 royalty-free video codec. The proposed methods reduce memory footprint and preserve hardware arithmetic decoding speed with limited impact on coding gain. Additional work is ongoing to improve the hardware friendliness of the overall AV1 codec.

Table 1. The BD-rate of the proposed simplified REFMV as compared to the REFMV coding tool.

	Low Delay				High Delay			
	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)
1080p	-0.01	-0.03	-0.05	-0.02	-0.14	-0.13	-0.16	-0.19
1080p-screen	-0.02	-0.07	-0.03	-0.09	0.08	0.04	-0.01	0.02
720p	0.01	0.00	-0.04	0.05	-0.04	-0.03	-0.01	-0.08
360p	-0.12	-0.12	-0.01	-0.01	0.04	-0.04	0.08	0.08
All	-0.03	-0.05	-0.04	-0.02	-0.01	-0.04	-0.03	-0.04

Table 2. The BD-rate of the REFMV coding tool as compared to AV1 baseline.

	Low Delay				High Delay			
	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)
1080p	-2.27	-2.07	-2.18	-2.22	-1.27	-1.01	-0.89	-1.10
1080p-screen	-2.27	-2.12	-2.14	-2.12	-2.00	-2.16	-2.19	-2.14
720p	-1.94	-1.86	-2.06	-2.03	-1.06	-0.90	-1.10	-1.19
360p	-1.01	-0.85	-1.03	-1.00	-0.72	-0.82	-0.89	-0.86
All	-1.87	-1.72	-1.85	-1.84	-1.26	-1.22	-1.27	-1.32

Table 3. The BD-rate quality impact of the relaxing the context model dependency in REFMV coding tool as compared to the original REFMV coding tool.

	Low Delay				High Delay			
	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)
1080p	0.79	0.75	0.84	0.82	0.45	0.45	0.41	0.41
1080p-screen	0.55	0.55	0.58	0.55	0.52	0.64	0.56	0.46
720p	1.08	1.02	1.07	1.14	0.43	0.40	0.37	0.32
360p	0.52	0.47	0.61	0.54	0.30	0.25	0.33	0.35
All	0.74	0.70	0.78	0.76	0.43	0.43	0.42	0.39

Table 4. The BD-rate quality impact of the removing the context model dependency in REFMV coding tool as compared to the original REFMV coding tool.

	Low Delay				High Delay			
	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)	PSNR (%)	PSNR HVS (%)	SSIM (%)	MS SSIM (%)
1080p	0.84	0.78	0.88	0.87	0.52	0.53	0.51	0.51
1080p-screen	0.52	0.60	0.59	0.55	0.76	0.89	0.96	0.76
720p	1.11	1.09	1.02	1.08	0.62	0.58	0.53	0.53
360p	0.44	0.45	0.54	0.54	0.33	0.18	0.26	0.28
All	0.73	0.73	0.76	0.76	0.56	0.55	0.56	0.52

REFERENCES

[1] J.D. Cock, A. Mavlankar, A. Moorthy, A. Aaron, “A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications”, *proceedings of SPIE applications of digital image processing*, vol. 9971, 2016.

[2] M.P. Sharabayko, N.G. Markov, “Contemporary video compression standards: H.265/HEVC, VP9, VP10, Daala”, *proceedings of the international Siberian conference on control and communications (SIBCON)*, 2016.

[3] J. Han, Y. Xu, J. Bankoski, “A dynamic motion vector reference scheme for video coding”, *proceedings of the IEEE international conference on image processing (ICIP)*, pp.2032-2036, 2016.

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

[5] https://media.xiph.org/video/derf/

[6] T. Daede, A. Norkin, I. Brailovskiy, “Video Codec Testing and Quality Measurement 2.5”, *AOM document*, 2016.

[7] Xiph.Org, "Are We Compressed Yet", 2016.

[8] K. Egiazarian, J. Astola, N. Ponomarenko, V. Lukin, F. Battisti, and M. Carli, “A new full-reference quality metrics based on HVS,” *CD-ROM proceedings of the second international workshop on video processing and quality metrics*, pp. 2-5, 2006.

[9] Z. Wang, A.C. Bovik , H.R. Sheikh, E.P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600-612, 2004.

[10] Z. Wang, E.P. Simoncelli, and A.C. Bovik, "Multi-Scale Structural Similarity for Image Quality Assessment", *proceedings of the 37th IEEE asilomar conference on signals, systems and computers*, Pacific Grove, CA, 2003.

[11] Bjontegaard, G. “Calculation of average PSNR differences between RD-curves”, *VCEG-M33*, Austin, TX, 2001.