HEVC ALF DECODE COMPLEXITY ANALYSIS AND REDUCTION

Madhukar Budagavi, Vivienne Sze, Minhua Zhou
{madhukar, sze, zhou}@ti.com
Texas Instruments Inc.
Dallas, TX – 75093, USA

ABSTRACT

This paper analyzes the decoder implementation complexity of a new tool called Adaptive Loop Filtering (ALF) being considered for the ITU-T/ISO/IEC High Efficiency Video Coding (HEVC) standard, and proposes new luma filters (Nx7 and Nx5) for ALF that reduce memory bandwidth, memory size requirements, and number of computations. The luma filters in ALF of the initial version HEVC Test Model (HM-1.0) have a maximum vertical size of 9. The vertical size of the ALF filters determines the memory size (line buffers) and memory bandwidth requirements. Accordingly, this paper proposes reducing the vertical size of ALF filters to 7 and 5, which are referred to as Nx7 and Nx5 filter sets respectively. These filters reduce memory bandwidth and size requirements by 25% and 50% respectively with minimal impact on coding efficiency. In addition, the worst case computational complexity is reduced by ~10% and ~20% respectively. Reduced vertical size luma ALF filters are under consideration for inclusion in HEVC standard with Nx7 being been adopted into HM-2.0 and Nx5 being under consideration for HM-4.0.

Index Terms— HEVC, video coding, loop filter

1. INTRODUCTION

The Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T WP3/16 and ISO/IEC JTC 1/SC 29/WG 11 is currently developing the next-generation video coding standard referred to as High Efficiency Video Coding (HEVC). HEVC is expected to provide around 50% improvement in coding efficiency compared with AVC/H.264. Furthermore, HEVC is intended for larger resolutions and higher frame rates. In order to address these requirements, HEVC utilizes larger "macroblocks" compared to AVC/H.264. In HEVC, the largest coding unit (LCU) can go up to 64x64, while in AVC/H.264 the macroblock size is fixed to 16x16.

The larger resolution and higher frame rates also increase the memory bandwidth of video codec implementations. Memory bandwidth impacts both power and processing speed. Frequent memory access significantly increases the power consumption of a video codec. In particular, accesses to large external/off-chip memory, such as a frame buffer, account for a significant portion of the overall system power of a video codec [1]. Low power consumption is critical for future video codecs as video compression and decompression is increasingly being done on mobile battery-operated devices where power is limited. On-chip caching can be used to reduce external memory bandwidth; however, it comes at a cost of increased hardware area. Memory bandwidth also impacts processing speed as the number of ports available in a memory architecture, both on and off-chip, is limited. Increasing memory bandwidth can result in more read/write conflicts, which can cause stalls leading to degradation in processing speed. This makes it more challenging to deliver the desired high resolution and frame rates.

In order to meet the power and performance requirements for future video coding applications, it is necessary to minimize the memory bandwidth demands of HEVC. Several new tools proposed for HEVC provide improved coding efficiency at a cost of increased memory bandwidth. It is important to analyze these tools and develop methods of reducing their memory bandwidth while maintaining the coding efficiency benefits. The memory size requirements should also be reduced to minimize area cost of on-chip buffers.

This paper proposes methods of reducing memory bandwidth, memory size requirements, and number of computations of a new tool in HEVC called the adaptive loop filtering (ALF). In the current ALF design in HEVC Test Model (HM), a line buffer is needed to store previous lines of the deblocked image to reduce memory bandwidth requirements in ALF decode. The size of the line buffer is determined by the vertical size of the filter. This paper presents two ALF filter sets – Nx7 and Nx5 – with reduced vertical size thereby reducing line buffer size and memory bandwidth requirements and number of computations. The paper will focus on the filters that are applied to the luma components of video.

This paper is organized as follows: Section 2 provides an overview of ALF filters. Section 3 discusses the

implementation challenges of ALF used in HM-1.0, particularly in terms of memory bandwidth and size requirements. Section 4 describes the proposed new set of filters for ALF. Section 5 presents the simulation results and finally, Section 6 summarizes the benefits of the proposal and describes the status of its adoption into the HEVC standard.

2. OVERVIEW OF FILTERS IN ALF

Adaptive loop filtering (ALF) is a new coding tool that has been introduced into HEVC. ALF is applied on the output of the deblocking filter as shown in Figure 1. The output of ALF is stored as the reference picture. The goal of ALF is to reduce the distortion between the input picture and the deblocked picture as shown in Figure 2. At the encoder, the filter coefficients are estimated using traditional Wiener filter estimation process by computing the auto-correlation of deblocked picture and cross-correlation of the deblocked picture and input picture.

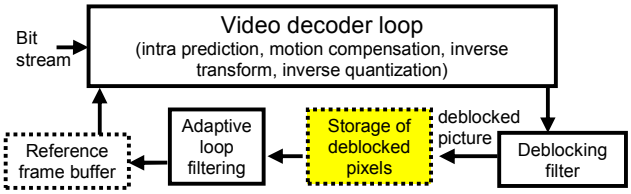

Figure 1: Location of adaptive loop filtering in video decoder. Dashed lines indicate memory storage units. This paper focuses on reducing memory bandwidth of the unit that stores deblocked pixels (highlighted in yellow).

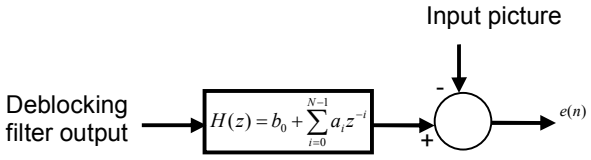

Figure 2: ALF filter estimation using Wiener filter.

ALF was first proposed to the ITU-T standards body in [2][3] and had square filters. The ALF was carried out on the entire deblocked picture. Subsequently, block-based adaptive loop filtering was proposed where the ALF could be enabled and disabled on a block (i.e. coding unit) basis [4]. The encoder would signal to the decoder the map of blocks of deblocked picture on which to apply ALF to improve the reconstructed quality. A further refinement to block-based adaptive loop filtering was the quadtree adaptive loop filtering [5] which signaled the map of where to apply ALF by using a quadtree. Diamond shaped ALF filters were used in [6] for luma components to reduce computation complexity.

The luma ALF in first version of the HEVC Test Model (HM-1.0) is based on filter presented in [6] and uses three diamond shaped filters of sizes 9, 7, 5 as shown in Figure 3 (the chroma filters in HM-1.0 have square kernels). The filter size is allowed to change for each frame, but all filters used within a frame have the same size. Up to 16 different filters can be used for each frame. The set of filter coefficients used for the ALF can also change for every frame. Accordingly, the set of filter coefficients and filter size are signaled at a frame level (i.e. 16 sets of coefficients and filter size are sent to the decoder every frame). At the decoder, a laplacian-based local activity is used to switch between the different filters on a block-by-block basis.

The filters have 180-degree rotation symmetry as indicated in the 9-diamond in Figure 3 where coefficients in similar shaped boxes are equal. While only a few boxes have been used in Figure 3 to illustrate the type of symmetry, all the coefficients in the filter are in fact symmetric. As a result, a filter of size N requires (N*N/4+1) multiplications. For size 9 filter, this translates to 21 multiplications. The number of pixels that need to be read from memory to carry out one filtering operation is 41.

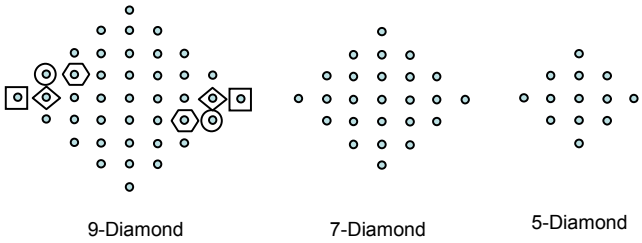

Figure 3: HM-1.0 Luma ALF filter set. Symmetry in the filters is highlighted using different shaped boxes.

3. ALF IMPLEMENTATION

In hardware and embedded software implementations of video codecs, processing is typically done on a macroblock [7]. For HEVC, this processing is expected to be done on LCU basis. In LCU-based ALF, shown in Figure 4, filtering is carried out on entire LCU before moving on to the next LCU. The left part of Figure 4 shows ALF filtering of LCU(0,0). The red lines show the deblocked pixel lines (which is input to ALF filters) and the blue lines are the ALF filtered output. Since the ALF filter is a non-causal filter and uses right and bottom LCU data (which are not yet available), not all deblocked pixels in an LCU can be ALF filtered. In Figure 4, the LCU pixels that *can* be ALF filtered when deblocked pixels for LCU are generated are shown in solid blue line and the LCU pixels that *cannot* be filtered when the deblocked pixels for LCU are generated are shown in dotted blue lines. Accordingly, these unfiltered pixels at the bottom of the LCU, which cannot be immediately filtered, need to be stored. In order to reduce external

memory bandwidth, these pixels can be cached in an on-chip line buffer. The unfiltered pixels (as shown on the left of Figure 4) are written to the line buffer by the deblocking filter; these pixels are then read from the line buffer by the ALF when it processes the next LCU row (as shown on the right of Figure 4).

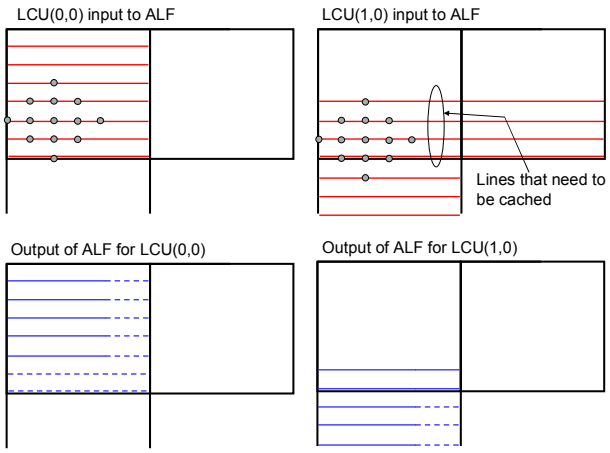

Figure 4: LCU-based ALF at the decoder with line buffer cache. Red lines are the available deblocked pixels for a given LCU. Solid blue lines are the pixels that can be filtered immediately by the ALF, while the dotted lines need to wait until other LCUs have been deblocked, and therefore need to be cached.

For a filter with vertical size M, the previous M-1 lines need to be cached in an on-chip line buffer to avoid incurring additional off-chip memory bandwidth. Assuming a 4Kx2K picture with maximum ALF filter size = 9 and 12-bit intermediate pixel values, the line buffer memory size is: 4K * 8 lines *12 bits = 48 Kbytes. Line buffers are also required for other modules in the decoder (e.g. deblocking filter). The data that needs to be stored for ALF accounts for ~50% of the total line buffer memory required for HM-1.0 decoding. Hence, it is important to reduce the line buffer requirements for ALF in the decoder for ALF to be implemented cost-effectively.

4. ALF FILTERS WITH REDUCED VERTICAL SIZE

For a given image size, the vertical size of ALF filters determines the size of line buffer and memory bandwidth requirements. These requirements can be reduced by reducing the vertical size of ALF filter. This paper presents two ALF filter sets that reduce the vertical size of the luma filter. Figure 5 shows both these filter sets – Nx7 and Nx5. Nx7 and Nx5 filter sets have maximum vertical size of 7 and 5 respectively. Since Nx7 operates on only 7 lines instead of the original 9 line, the size of the line buffers goes down from 8 lines to 6 lines. For Nx5, the size of line buffers goes down to 4 lines. Table 1 provides a summary of the line buffer size of different filters (and also the worst case computations and BD-Rate performance). Nx7 reduces ALF memory bandwidth and memory size requirements by 25% whereas Nx5 reduces ALF memory bandwidth and memory size requirements by 50%. In addition, Nx7 and Nx5 filter sets reduce worst case number of multiplications by ~10% and ~20% respectively when compared to HM-1.0 ALF filters (See Table 1). Accordingly, the number of pixels that need to be read from the line buffer to carry out one filtering operation is also reduced (e.g. a 9x5 filter now only requires 33 reads instead of 41).

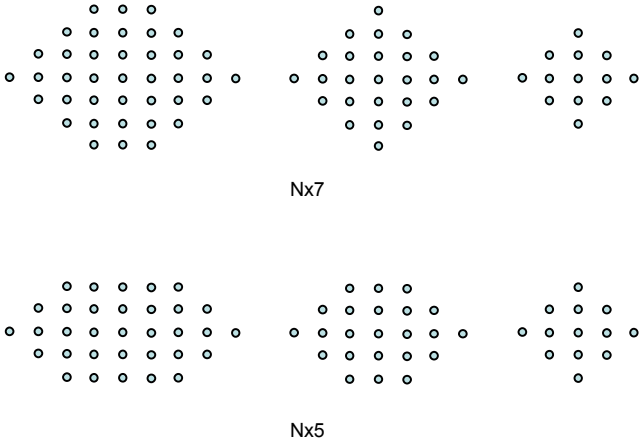

Figure 5: ALF filters with reduced vertical size.

5. SIMULATION RESULTS

Nx7 and Nx5 filter sets were integrated into HM-1.0 version of the Test Model software being used in HEVC standardization. Testing was carried under the common conditions defined by JCT-VC in [8]. The common conditions use 20 video sequences with different resolutions from WQVGA (416x240) to 2560x1600. Most of the sequences are of 10 seconds duration. There are three test conditions: (a) Intra – where all frames are coded in Intra mode, (b) Random access – where Hierarchical-B coding structure with both forward and backward prediction is used, and (c) Low delay – where only forward predicted frames are used.

Table 1 shows the BD-Rate [9] savings of existing HM-1.0 ALF filter set, proposed Nx7 and Nx5 filter sets. Also shown in the table are results when only HM-1.0 5x5 and 5x5/7x7 filters are used. It can be seen from Table 1 that the proposed ALF filter sets capture most of the ALF coding gains while reducing line buffer size, memory bandwidth and worst case computations.

			Intra	Random access	Low delay
	Line buffer size (lines)	Worst case multiplies	BD-Rate	BD-Rate	BD-Rate
No ALF	0	0			
HM-1.0 ALF	8	21	-3.3	-4.1	-4.1
HM-1.0 5x5	4	7	-2.6	-3.4	-3.0
HM-1.0 7x7, 5x5	6	13	-3.1	-3.8	-3.3
Nx7	6	19	-3.3	-4.1	-3.8
Nx5	4	17	-3.2	-4.0	-3.7

Table 1: Summary of line buffer size, worst case multiplies, and BD-Rate of ALF filters. All data compared to case of no ALF.

6. DISCUSSIONS AND CONCLUSIONS

This paper analyzes the implementation complexity of adaptive loop filtering (ALF) for luma at the decoder. Implementation complexity analysis involves not only analysis of computations, but also analysis of memory bandwidth and memory size (i.e. area cost of the on-chip line buffer). Line buffers can be used to store previous lines of deblocked picture to reduce external memory bandwidth. In this work, we present two ALF filter sets – Nx7 and Nx5 – that reduce vertical size of filter thereby reducing the line buffer size and memory bandwidth requirements. Nx7 has a maximum vertical size of 7 and Nx5 has a maximum vertical size of 5. Nx7 reduces ALF memory bandwidth and memory size requirements by 25% whereas Nx5 reduces ALF memory bandwidth and memory size requirements by 50% when compared to HM-1.0 ALF filter set. In addition, Nx7 and Nx5 filter sets reduce worst case number of multiplications by ~10% and ~20% respectively when compared to HM-1.0 ALF filter set.

Existing ALF filters in HM-1.0 provide average BD-Rate savings in range of 3.3% to 4.1%. Nx7 provides average BD-Rate savings in the range of 3.3% to 4.1% where as Nx5 provides average BD-Rate savings in the range of 3.2% to 4%. Both the proposed ALF filter sets capture most of the ALF coding gains.

Nx7 and Nx5 filters presented in this paper were proposed to JCT-VC and decoder implementation complexity issues of ALF (especially memory bandwidth and memory size requirements) were highlighted in [10]. Reduced vertical size luma ALF filters are now under consideration for inclusion in HEVC standard. Nx7 filter set proposed in this paper has been adopted into HM-2.0 as a first step [11]. Nx5 filter [12] in conjunction with cross-shaped filter of [13] modified to have maximum vertical size of 5 [14] is under consideration for inclusion in HM-4.0.

7. REFERENCES

[1] M. Budagavi, M. Zhou, "Video coding using compressed reference frames," IEEE ICASSP 2008.

[2] T. Chujoh, A. Tanizawa and T. Yamakage, "Adaptive loop filter for Improving Coding Efficiency," ITU-T SG16 Contribution, C402, Geneva, April 2008.

[3] Yi-Jen Chiu and L. Xu, "Adaptive (Wiener) Filter for Video Compression," ITU-T SG16 Contribution, C437, Geneva, April 2008.

[4] T. Chujoh, G. Yasuda, N. Wada, T. Watanabe and T. Yamakage, "Block-based Adaptive loop filter," ITU-T SG16 Q.6 Document, VCEG-AI18, Berlin, July 2008.

[5] T. Chujoh, G. Yasuda, N. Wada, "Quadtree-based adaptive loop filter," ITU-T SG16 Contribution, C181, January 2009.

[6] M. Karczewicz et. al., "A hybrid video coder based on extended macroblock sizes, improved interpolation, and flexible motion representation," IEEE Trans. CSVT, pp. 1698-1708, Vol. 20, No. 12, Dec. 2010.

[7] T-D. Chuang et. al., "A 59.5mW Scalable/Multi-View Video Decoder Chip for Quad/3D Full HDTV and Video Streaming Applications," IEEE ISSCC 2010.

[8] F. Bossen, "Common test conditions and software reference configurations," JCT-VC contribution, JCTVC-C500, Guangzhou, CN, Oct. 2010.

[9] G. Bjøntegaard, "Calculation of average PSNR differences between RD-Curves," ITU-T SG16 Q.6 Document, VCEG-M33, Austin, April 2001.

[10] M. Budagavi, V. Sze, M. Zhou, "ALF decode complexity analysis and reduction," JCT-VC contribution, JCTVC-D039, Daegu, KR, Jan. 2011.

[11] T. Wiegand et al., "WD2: Working Draft 2 of High-Efficiency Video Coding," JCT-VC contribution, JCTVC-D503, Daegu, KR, Jan. 2011.

[12] M. Budagavi, V. Sze, M. Zhou, "CE8 Subtest 5: Luma ALF with reduced vertical filter size," JCT-VC contribution, JCTVC-E060, Geneva, CH, Mar. 2011.

[13] F. Kossentini et al., "Adaptive Loop Filtering Using Two Filter Shapes", JCT-VC contribution, JCTVC-E342, Geneva, CH, Mar. 2011.

[14] F. Kossentini et al., "CE8 Subset 5: Results on combination of JCTVC-E342 + JCTVC-E060", JCT-VC contribution, JCTVC-E492, Geneva, CH, Mar. 2011.