SYNTAX ELEMENT PARTITIONING FOR HIGH-THROUGHPUT HEVC CABAC DECODING

Philipp Habermann[⋆]*, Chi Ching Chi*[†]*, Mauricio Alvarez-Mesa*[†] *and Ben Juurlink*[⋆]

[⋆] Embedded Systems Architecture Group, Technische Universität Berlin, Germany
[†] Spin Digital Video Technologies GmbH, Berlin, Germany
Email: {p.habermann, b.juurlink}@tu-berlin.de, {chi, mauricio}@spin-digital.com

ABSTRACT

Encoder and decoder implementations of the High Efficiency Video Coding (HEVC) standard have been subject to many optimization approaches since the release in 2013. However, the real-time decoding of high quality and ultra high resolution videos is still a very challenging task. Especially entropy decoding (CABAC) is most often the throughput bottleneck for very high bitrates. Syntax Element Partitioning (SEP) has been proposed for the H.264/AVC video compression standard to address this issue and the limitations of other parallelization techniques. Unfortunately, it has not been adopted in the latest video coding standard, although it allows to multiply the throughput in CABAC decoding.

We propose an improved SEP scheme for HEVC CABAC decoding with eight syntax element partitions. Experimental results show throughput improvements up to 5.4× with negligible bitstream overhead, making SEP a useful technique to address the entropy decoding bottleneck in future video compression standards.

***Index Terms*—** HEVC, H.265, CABAC, Parallelization

1. INTRODUCTION

High Efficiency Video Coding (HEVC, [1]) is the most recent video coding standard developed by the Joint Collaborative Team on Video Coding (JCT-VC). It allows the compression of videos with the same perceptive quality as its predecessor H.264/AVC [2] while requiring only half the bitrate. Context-based Adaptive Binary Arithmetic Coding (CABAC, [3]) is the entropy coding module in the HEVC standard and the main throughput bottleneck for high bitrates because the sequential algorithm makes parallelization very challenging.

Many optimization approaches have been implemented to improve the throughput of the critical CABAC decoding. First of all, two high-level parallelization techniques have been adopted in the HEVC standard, as it was not only designed for high compression rates but also for high throughput. By using Tiles, a frame is split into multiple rectangular areas that can be decoded simultaneously. Wavefront Parallel Processing (WPP) allows the parallel decoding of consecutive rows of Coding Tree Units (CTUs) in the same frame. Both techniques require the replication of the complete CABAC decoding hardware. Tiles lead to a decreased compression rate which is proportional to the number of tiles, because there cannot be any inter-tile dependencies. The use of WPP affects the CABAC learning process as the context variables are reset at the beginning of every CTU row. However, the coding losses are minimal for high resolution videos. Furthermore, WPP has scalability issues as there is a ramp-up and -down in active parallel threads due to the delayed decoding start of consecutive CTU rows. Overlapped Wavefront Processing (OWF) has been proposed by Chi et al. [4] as an implementation optimization that extends WPP to multiple parallel frames. This avoids the ramp-up/down phase in every frame and scales to many more parallel threads. Tiles and WPP are not mandatory, which means that they can only be used for improved decoding throughput when they were enabled in the encoding process.

There are also low-level parallelization approaches for CABAC hardware decoding. Pipelining can be used to overlap the decoding of consecutive binary symbols (bins). Among others, this has been implemented by Chen and Sze who used a five-stage pipeline [5]. It is also possible to decode multiple bins per clock cycle (e.g. Lin et al. [6] or Kim and Park [7]). Unfortunately, the efficient implementation of both techniques is limited to few parallel bins due to strong data and control dependencies.

To address the drawbacks of the described parallelization approaches, Sze et al. have proposed Syntax Element Partitioning (SEP, [8]) for H.264/AVC. Parallelism is exploited by distributing syntax elements among different partitions, so that they can be decoded simultaneously. This enables a significant decoding speed-up with only minimal losses in coding efficiency. As only parts of the decoding hardware need to be replicated, there is only a 50 % increase in hardware cost for five parallel partitions. This proposal requires a modification of the bitstream format and is therefore not compliant with the H.264/AVC standard. However, the multiplication of the decoding throughput with minimal coding losses and moderate hardware requirements makes SEP a promising

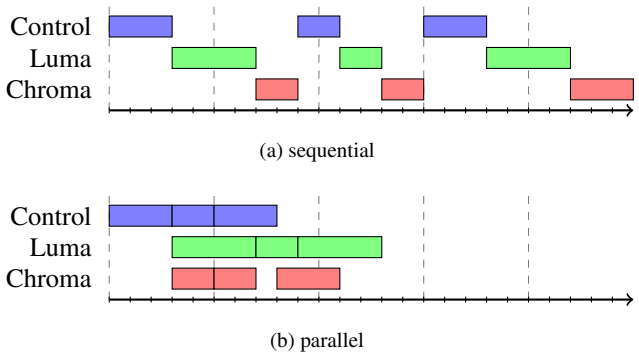

Fig. 1. Decoding of syntax element partitions

candidate for adoption in future video compression standards.

In this paper we present an improved SEP scheme for HEVC CABAC decoding. Section 2 describes the general SEP functionality and the implementation of our SEP scheme. Experimental results for decoding speed-up and bitstream overhead are presented in Section 3. Finally, the work is concluded in Section 4 and an overview of future work is provided.

2. SYNTAX ELEMENT PARTITIONING

Syntax element partitioning aims to divide a common bitstream in multiple parts that can be decoded in parallel. Figure 1 illustrates the effect by showing the decoding process for three groups of syntax elements. In the example, there is one for luma and one for chroma transform blocks, as well as a control group that contains all remaining syntax elements, e.g. for prediction modes, prediction units and loop filters. In a common HEVC bitstream, all syntax elements are coded consecutively in a single partition, which makes their sequential decoding necessary (a). However, if they are distributed among different partitions, parallel decoding is possible (b). Luma and chroma transform blocks are completely independent from each other. Their decoding process can be started as soon as the corresponding control block is decoded. At the same time, the decoding of the next control block can be initiated. This allows the overlapped decoding of all three partitions. As a result, less time is required to decode all partitions. The resulting video is the same as with the corresponding sequential bitstream as the same syntax elements are only distributed in a different way.

2.1. Implementation

The proposed SEP scheme consists of eight partitions. First, the common bitstream is divided into three parts according to the example in Figure 1: control, luma and chroma. Each of these partitions is further split into separate parts for context-coded (cc) and bypass-coded (bc) bins. The latter are coded without context models, which simplifies the decoding process. In fact, a bc bin corresponds to a bit and does not need to be encoded or decoded at all, if it is not interleaved with cc bins in a common bitstream. This allows the highly parallel retrieval of bc bins as they only need to be read from memory. Unfortunately, the Luma and Chroma CC Partitions still contain significantly more bins than others. To achieve a more balanced distribution, these partitions are divided into two parts that contain the syntax elements for the significance map and the coefficient level. All other bins are moved to the Control CC Partition.

A further split into partitions for both chroma components is not gainful as they use the same context models, thus making their parallel decoding impossible. In contrast to the proposal of Sze et al. we use a static partitioning scheme which does not adapt to video characteristics. A dynamic scheme allows a balanced distribution of bins to the partitions for all test sequences. However, the Luma/Chroma Significance Map Partitions most often contain the majority of bins for high bitrates, so the maximum speed-up is determined by these partitions. As they cannot be split further, a dynamic partitioning would not lead to a higher speed-up. On the other hand, the corresponding decoding hardware can be simplified for static partitions. The decoding of low bitrate videos is most often dominated by the size of the Control CC Partition and does not benefit from this static partitioning. Nevertheless, their throughput requirements are very low, so that real-time decoding is possible even without the use of SEP. An overview of the proposed distribution among syntax element partitions is provided in Table 1. It should be noted that some syntax elements appear in more than one partition as they consist of cc and bc bins. Also the same syntax elements exist for luma and chroma transform blocks.

2.2. Bitstream Overhead

The ability to decode multiple bitstream partitions in parallel comes at the cost of additional bitstream overhead. First, there is a variable-sized length field for every partition (1-4 bytes) to signal the starting position of the next partition. Additionally, there is an arithmetic coding overhead for each of the five cc partitions (2 bytes). Finally, byte alignment bits are added to all partitions (3.5 bits on average). This adds 16-47 bytes of additional bitstream size per slice. The relative overhead depends on the bitrate of the video and can be significant for very low bitrates. SEP can be disabled for these videos with a single bit in the sequence parameter set or the slice header as CABAC decoding is usually not critical in these cases.

3. EVALUATION

The HEVC reference software [9] has been modified to encode and decode bitstreams according to the proposed SEP scheme. Furthermore, a cycle-accurate architectural model of

Partition	Syntax elements
Control CC	end_of_slice_segment_flag, end_of_subset_one_bit, sao_merge_left_flag, sao_merge_up_flag, sao_type_idx_luma, sao_type_idx_chroma, split_cu_flag, cu_transquant_bypass_flag, cu_skip_flag, pred_mode_flag, part_mode, pcm_flag, prev_intra_luma_pred_flag, intra_chroma_pred_mode, rqt_root_cbf, merge_flag, merge_idx, inter_pred_idc, ref_idx_l0, mvp_l0_flag, ref_idx_l1, mvp_l1_flag, split_transform_flag, cbf_luma, cbf_cb, cbf_cr, abs_mvd_greater0_flag, abs_mvd_greater1_flag, cu_qp_delta_abs, cu_chroma_qp_offset_flag, cu_chroma_qp_offset_idx, log2_res_scale_abs_plus1, res_scale_sign_flag, transform_skip_flag, explicit_rdpcm_flag, explicit_rdpcm_dir_flag, last_sig_coeff_x_prefix, last_sig_coeff_y_prefix, coded_sub_block_flag
Control BC	sao_type_idx_luma, sao_type_idx_chroma, sao_offset_abs, sao_offset_sign, sao_band_position, sao_eo_class_luma, sao_eo_class_chroma, part_mode, mpm_idx, rem_intra_luma_pred_mode, intra_chroma_pred_mode, merge_idx, ref_idx_l0, ref_idx_l1, abs_mvd_minus2, mvd_sign_flag, cu_qp_delta_abs, cu_qp_delta_sign_flag, last_sig_coeff_x_suffix, last_sig_coeff_y_suffix
Luma Sig Map	sig_coeff_flag
Luma Coeff Level	coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag
Luma BC	coeff_sign_flag, coeff_abs_level_remaining
Chroma Sig Map	sig_coeff_flag
Chroma Coeff Level	coeff_abs_level_greater1_flag, coeff_abs_level_greater2_flag
Chroma BC	coeff_sign_flag, coeff_abs_level_remaining

Table 1. Syntax element partitions (CC: context-coded, BC: bypass-coded)

the corresponding hardware decoder has been implemented to estimate the maximum speed-up that can be achieved with the parallel decoding. To cover a wide range of video sequences, the following JCT-VC test sets are used for evaluation.

- Common test conditions (class A-E) [10]
- Natural content coding conditions for HEVC range extensions (YCbCr 4:2:2, YCbCr 4:4:4, RGB 4:4:4) [11]

They are encoded in all-intra (AI), random-access (RA) and low-delay (LD) modes with quantization parameters (QP) from 12 up to 37 (common test set only specified for QP 22 to 37). In general, higher QPs result in lower bitrates and lower video quality. The presented results are the geometric means of all test sequences of a specific class.

The remaining evaluation section covers the speed-up and bitstream overhead resulting from the implementation of the proposed SEP scheme.

3.1. Speed-up

The parallel decoding of multiple syntax element partitions reduces the processing time and results in a speed-up (see Figure 2). The most significant improvements can be reached for AI sequences. They require the highest bitrates as they go without the effective inter-picture prediction. Smaller QPs also raise the speed-ups as the resulting increased bitrates lead to a more balanced distribution of bins among the different partitions. For very low bitrate sequences, the Control CC Partition contains most bins and determines the overall decoding throughput. Furthermore, the fraction of bc bins grows with decreasing QPs. This also improves the throughput as they can be decoded in a highly parallel way.

For all high bitrate sequences from the common test set (Figure 2 a), the Luma CC Partition is the decoding bottleneck. The maximum speed-up for a single sequence is 3.8×. This is a significant improvement compared to the implementation of Sze et al. who reached up to 2.3× speed-up for high bitrates. The sequences from the range extensions test set (Figure 2 b) allow an even better distribution of bins among the partitions due to the reduced chroma subsampling. 4:2:2 subsampling results in the best balanced partitions, while the decoding of 4:4:4 sequences is dominated by the size of the Chroma CC Partition. The result is a maximum speed-up of 5.4× for a single test sequence.

3.2. Bitstream Overhead

The partitioning of the bitstream for the purpose of parallel decoding comes at the cost of additional bitstream overhead (see Figure 3) as described in Section 2.2. In general, the overhead depends strongly on the bitrate. This means in relative terms that AI videos add less bytes to the bitstream than RA and LD videos. Also, lower QPs relatively add less bytes. Except for the very low bitrate videos in LD mode or with high QPs, the overhead is less than one per cent and therefore negligible. This is especially true for the range extensions test set. SEP can be disabled for videos where it results in a significant overhead as their throughput requirements are very low.

There is one abnormal value in the results, because a single test sequence (DucksAndLegs) has 60× more overhead than the other sequences from the RGB 4:4:4 class when encoded in AI mode with QP 12. The reason is that there are many zero bytes in one of the partitions. According to the HEVC standard, an *emulation_prevention_3_byte* is al-

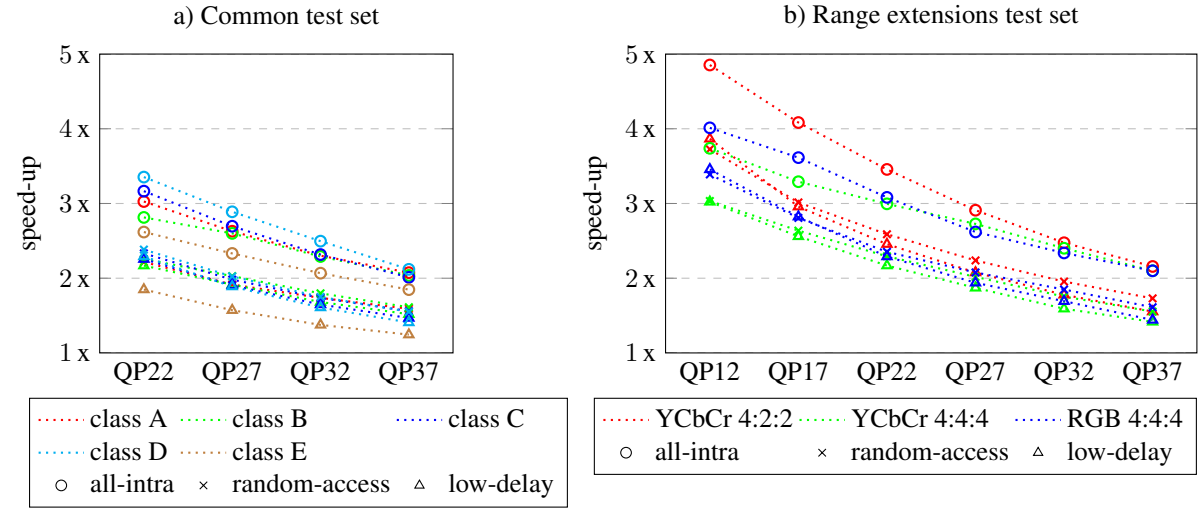

Fig. 2. Speed-up

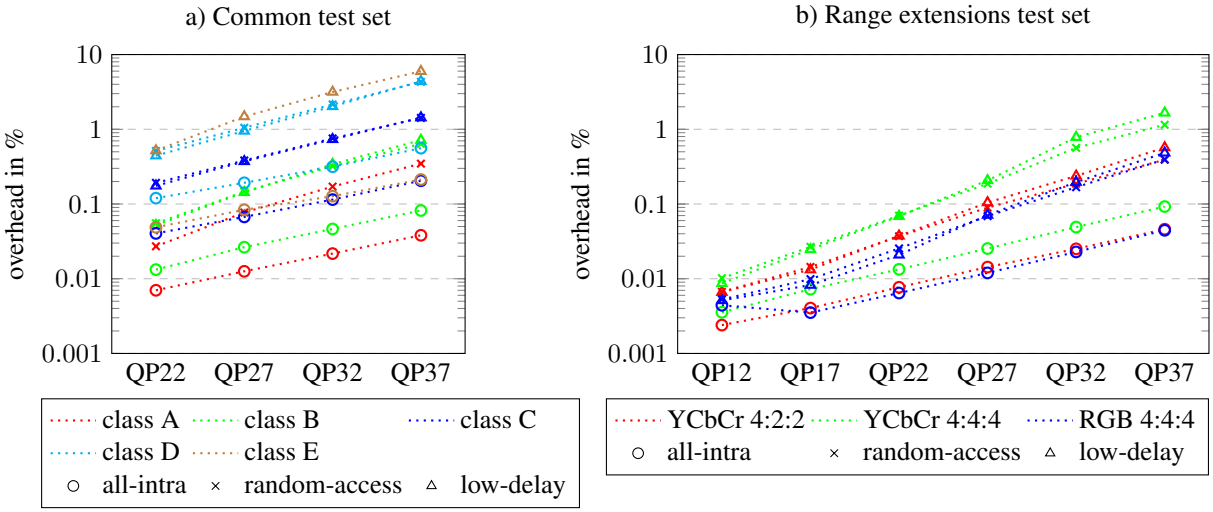

Fig. 3. Bitstream overhead

ways added after two consecutive zero bytes. This behavior depends on the video characteristics and cannot be avoided. However, the resulting overhead of the specific sequence is still only 0.024% and therefore negligible.

4. CONCLUSIONS

We have presented a Syntax Element Partitioning scheme for HEVC CABAC decoding. Bins of different syntax elements are distributed among eight partitions to enable their parallel decoding. As a result, a speed-up of up to 5.4 × can be achieved with negligible bitstream overhead. The overhead can exceed 5 % for very low bitrates, however, SEP can be disabled for these sequences because the very low throughput requirements even allow sequential real-time decoding. The proposed optimization is most effective for high bitrates where CABAC decoding throughput is most critical for the overall decoding performance, thus making it a reasonable choice for adoption in future video compression standards.

Future work will cover the implementation of the corresponding hardware decoder for the proposed SEP scheme. An additional speed-up is expected as the clustering of the common decoder will result in multiple faster decoders for the different partitions due to smaller state machines and context model memories. Furthermore, a higher level of customization can be achieved due to the specialized operation of the decoders for the fixed syntax element partitions. We expect that the parallel CABAC hardware decoder will consume less than 2× the hardware resources of a sequential decoder because only parts need to be replicated.

5. REFERENCES

[1] G. J. Sullivan, J. 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, Volume 22, Issue 12, pp. 1649-1668, December 2012

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, *"Overview of the H.264/AVC Video Coding Standard"*, IEEE Transactions on Circuits and Systems for Video Technology, Volume 13, Issue 7, pp. 560-576, July 2003

[3] V. Sze and M. Budagavi, *"High Throughput CABAC Entropy Coding in HEVC"*, IEEE Transactions on Circuits and Systems for Video Technology, Volume 22, Issue 12, pp. 1778-1791, December 2012

[4] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux and T. Schierl, *"Parallel Scalability and Efficiency of HEVC Parallelization Approaches"*, IEEE Transactions on Circuits and Systems for Video Technology, Volume 22, Issue 12, pp. 1827-1838, December 2012

[5] Y.-H. Chen and V. Sze, *"A Deeply Pipelined CABAC Decoder for HEVC Supporting Level 6.2 High-tier Applications"*, IEEE Transactions on Circuits and Systems for Video Technology, Volume 25, Issue 5, p. 856-868, May 2015

[6] P.-C. Lin, T.-D. Chuang and L.-G. Chen, *"A Branch Selection Multi-symbol High Throughput CABAC Decoder Architecture for H.264/AVC"*, IEEE International Symposium on Circuits and Systems (ISCAS 2009), pp. 365-368, Taipei, Taiwan, May 2009

[7] C.-H. Kim and I.-C. Park, *"High Speed Decoding of Context-based Adaptive Binary Arithmetic Codes using Most Probable Symbol Prediction"*, IEEE International Symposium on Circuits and Systems (ISCAS 2006), pp. 1707-1710, Island of Kos, Greece, May 2006

[8] V. Sze, A. P. Chandrakasan, *"A High Throughput CABAC Algorithm using Syntax Element Partitioning"*, IEEE International Conference on Image Processing (ICIP 2009), pp. 773-776, Cairo, Egypt, November 2009

[9] HEVC Test Model v16.7, `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/`

[10] F. Bossen, *"Common HM test conditions and software reference configurations"*, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L1100, Geneva, Switzerland, January 2013

[11] C. Rosewarne, K. Sharman and D. Flynn, *"Common test conditions and software reference configurations for HEVC range extensions"*, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-P1006, San Jose, CA, USA, January 2014