Improved Wavefront Parallel Processing for HEVC Decoding

Philipp Habermann*, Chi Ching Chi†,
Mauricio Alvarez-Mesa†, Ben Juurlink*

*Berlin University of Technology, Berlin, Germany
†Spin Digital Video Technologies GmbH, Berlin, Germany

ABSTRACT

Wavefront Parallel Processing (WPP) is a high-level parallelization tool adopted in the High Efficiency Video Coding standard (HEVC/H.265). WPP allows the simultaneous processing of consecutive rows of coding tree units (CTUs) in a frame. A horizontal offset of at least two CTUs is commonly implemented to maintain all dependencies. This causes ramp-up and -down issues in active parallel decoding threads, which limits parallel scalability. However, a two CTU offset is not necessary for the whole decoding process. In this paper, we propose a fine-grained synchronization approach to address the ramping problem. A theoretical analysis of the decoding dependencies is performed to estimate the potential speed-up with this approach.

KEYWORDS: HEVC, H.265, Wavefront Parallel Processing

1 Introduction

High Efficiency Video Coding (HEVC/H.265) [1] is the latest video coding standard of the Joint Collaborative Team on Video Coding (JCT-VC). It provides 50% bitrate savings at the same subjective video quality compared to its predecessor H.264/AVC [2]. The HEVC video coding standard was not only designed for high compression rate, but also to allow high-throughput implementations. Therefore, two high-level parallelization techniques are explicitly integrated in the standard: Tiles and Wavefront Parallel Processing (WPP). By using Tiles, a frame is split into multiple rectangular areas that can be decoded simultaneously. This technique leads to a decreased compression rate which is proportional to the number of tiles, because there cannot be any inter-tile prediction.

WPP allows the parallel decoding of consecutive rows of coding tree units (CTUs) within a frame (see Figure 1). All prediction opportunities are preserved by maintaining a horizontal offset of two CTUs between consecutive CTU rows. WPP has scalability issues because 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. [3] as an optimization that extends WPP to multiple parallel frames. This avoids the ramp-up/down phase in every frame and scales to more parallel threads, but introduces

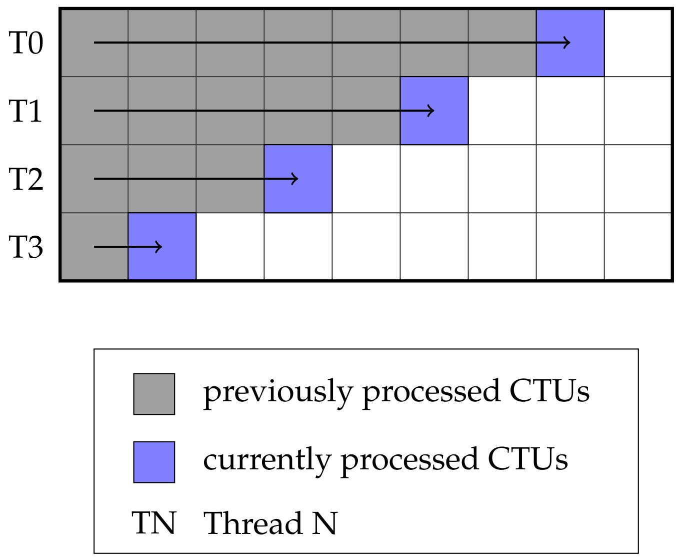


Figure 1: Wavefront Parallel Processing

0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61
42	43	46	47	58	59	62	63

Figure 2: Z-scan order in a CTU

restrictions for motion vectors. Our work provides an analysis of a standard-conforming fine-grained synchronization approach that addresses the ramp-up/down issue to improve WPP decoding throughput.

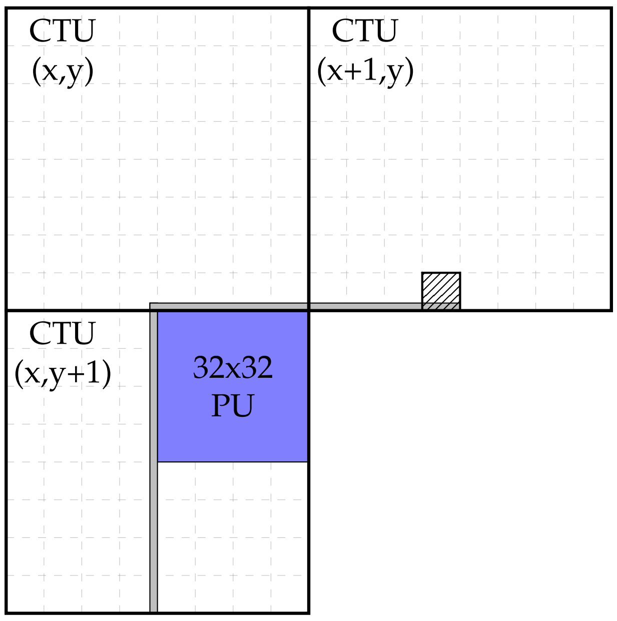
2 Wavefront Parallel Processing Dependencies

A conventional WPP-based HEVC decoder implements a horizontal offset of two CTUs between consecutive CTU rows. However, this is not necessary to maintain all dependencies. The decoding of CTU rows occurs at different speeds due to varying complexity of the corresponding frame content. That is why the offset can be reduced after the context initialization which is performed before decoding each row. Our approach implements a synchronization based on blocks of 8×8 samples instead of 64×64 sample CTUs. Threads are allowed to come below the two CTU offset and are only stalled if actual dependencies are detected. In this section we analyze the worst-case CTU offsets for Context-based Adaptive Binary Arithmetic Coding (CABAC), Intra Prediction, Motion Vector Derivation, as well as Deblocking and Sample Adaptive Offset (SAO) filters.

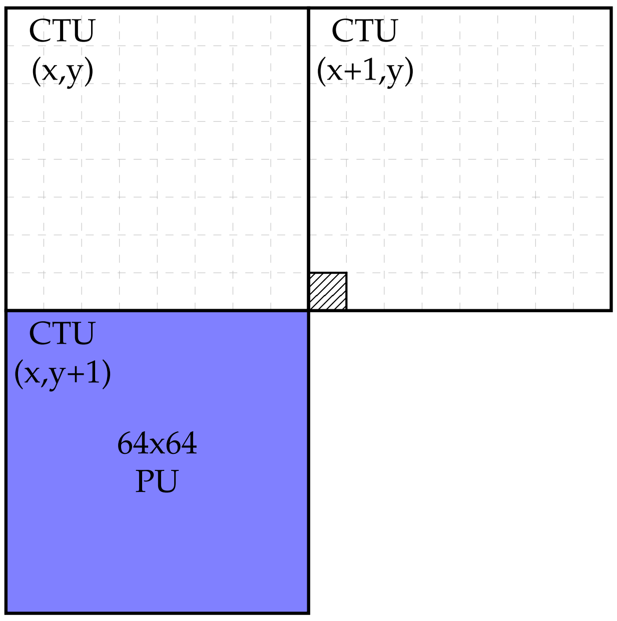
Figure 2 shows the z-scan order for processing all 64 8×8 blocks in a 64×64 CTU. Bigger blocks, e.g. 16×16, consist of multiple 8×8 blocks that are processed simultaneously. These block indices are used for CTU offset calculations between two threads N and $N-1$ in the following paragraphs.

$$CTU\,offset = CTUIdx_{N-1} - CTUIdx_N + \frac{blockIdx_{N-1} - blockIdx_N}{64}$$

The HEVC reference software [4] has been used to gather statistics from all class B (1080p) videos of the HEVC common test conditions [5]. The videos were encoded in random-access mode with a quantization parameter of 22, which represents high video quality.



(a) Intra Prediction



(b) Motion Vector Derivation

Figure 3: Worst-case dependencies (PU: prediction unit)

2.1 CABAC

CABAC decoding extracts syntax elements from the compressed bitstream which are used to control all other decoding steps. Before decoding a new CTU row, the context model state after decoding the second CTU in the previous row is copied. This introduces the commonly implemented offset of two CTUs. Afterwards, only information for intra prediction mode derivation, as well as for the context model selection for *split_cu_flag* and *cu_skip_flag* syntax elements might be required from the top CTU. As a result, the required offset is less than one CTU after the context initialization.

2.2 Intra Prediction

Intra prediction is used to exploit spatial similarities within a frame to predict a block based on neighboring samples. In the worst case, the top-right 32×32 block in a CTU (z-scan blocks 16-31) requires samples from block 47 of the top-right CTU (see Figure 3a). To maintain this dependency, a 1.5 CTU offset must be ensured between both CTU rows. However, these samples are only needed for a few intra prediction modes. The worst case occurs less than once per frame on average because there are also different block sizes and commonly only intra-predicted samples can be used as reference.

2.3 Motion Vector Derivation

Inter prediction is used to exploit temporal similarities between blocks in consecutive frames. Similar matching blocks can be copied instead of coding the information again. However, they might be at a different position in the reference frame. This difference is represented by a motion vector. The motion vector is most often derived from a set of candidates that have been used to predict neighboring blocks. In the worst case, a 64×64 block in a CTU (z-scan blocks 0-63) needs the motion vector that was used in block 42 of the top-right CTU (see Figure 3b), corresponding to an offset of 1.66 CTUs. This case occurs only 0.56 times per CTU row on average due to a variety of prediction unit sizes and different motion vector candidates.

2.4 Deblocking and SAO Filter

Deblocking and SAO filters are applied to remove compression artifacts that appear due to block-based coding. The filtering process has to be delayed even when WPP is not used, because reconstructed samples from all neighboring CTUs might be needed. That is why the filtering process is not relevant for the analysis of WPP dependencies.

3 Conclusions

An optimized approach for WPP-based HEVC decoding has been proposed in this paper. It has been shown that the commonly implemented horizontal offset of two CTUs between consecutive CTU rows is not necessary during the whole decoding process. The decoding throughput can be improved by implementing a fine-grained synchronization and exploiting that the amount of work per CTU heavily depends on the corresponding video content. A theoretical gain of 9.6% can be achieved for Full HD videos while preserving worst-case dependencies (1.66 CTU offset). As the worst case is very rare, even higher improvements might be achievable, e.g. 20.9% for a 1.33 CTU offset.

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] 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

[4] HEVC Test Model v16.15, `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/`

[5] 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