IMPROVING THE PARALLELIZATION EFFICIENCY OF HEVC DECODING

Chi Ching Chi[1] * , *Mauricio Alvarez-Mesa*[1,2], *Ben Juurlink*[1], *Valeri George*[2], *Thomas Schierl*[2]

[1]Embedded Systems Architectures, Technische Universität Berlin, Berlin, Germany.
[2]Multimedia Communications, Fraunhofer HHI, Berlin, Germany.

ABSTRACT

In this paper we present a new parallelization approach for HEVC decoding called Overlapped Wavefront (OWF). It is based on wavefront processing and improves its parallelization efficiency by allowing overlapped execution of consecutive pictures. Furthermore, in this strategy of the decoding steps are performed on a CTB basis rather than on a picture basis, which improves data locality. Our implementation achieves between 29.6%, 42.4%, and 66.6% higher frame rates compared to previous results and 11.3%, 21.0%, and 38.0% higher frame rates compared to Tiles, for 2160p, 1600p, and 1080p, respectively.

Index Terms— HEVC, video codecs, parallel processing.

1. INTRODUCTION

Recent demands on video coding support for high resolutions such as 4k or UHD in consumer devices have further driven the video coding development. The Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T and ISO/IEC MPEG has started a new project to develop a new video coding standard, called High Efficiency Video Coding (HEVC) [1], that aims to reduce the bitrate of H.264/AVC [2] by another 50%.

In the development of the new video codec standard it is being taken into account that contemporary and future computer architectures are parallel (multi- and many-core). For high-level parallelism, HEVC currently supports different picture partition strategies such as entropy slices, wavefront parallel processing (WPP) and Tiles. None of these approaches, however, completely fulfill the requirements of a scalable and efficient parallel decoder.

In this paper we propose a parallelization strategy approach based on WPP, called Overlapped Wavefront (OWF), which has several advantages compared to the existing ones. The method consist of first, creating one picture partition per row, second, to include all the decoding steps in a single pass and third to allow the overlapped execution of consecutive pictures. Parallel implementations for OWF and Tiles have been performed on top of HEVC test Model (HM) reference software.

*C. C. Chi has received funding from the ENCORE European Project (contract n° 248647).

2. PARALLELIZATION APPROACHES

For parallelization of video decoders picture-level partitioning has several advantages compared to other approaches. In previous video codecs, like H.264/AVC, picture partitions were only possible with slices with a high cost in terms of coding efficiency. For scalable parallel H.264/AVC decoding it is necessary to combine macroblock-level parallelism for picture reconstruction and frame-level parallelism for entropy decoding [3]. his approach, however, provides limited reduction in picture latencies and results in high memory usage.

In order to overcome these limitations, new parallelization strategies have been included in the HEVC codec. The HEVC draft standard contains 4 different approaches: slices, entropy slices [4], wavefront parallel processing (WPP) [5] and Tiles [6].

Slices have the largest coding penalty as they break entropy decoding and prediction dependencies. Entropy slices, like slices, break entropy decoding dependencies but allow prediction (and filtering) to cross slice boundaries. In WPP there is one picture partition per row and both entropy decoding and prediction are allowed to cross partitions. In this way coding losses are minimized while at the same time wavefront parallelism can be exploited. Tiles define horizontal and vertical boundaries that partition a picture into tile columns and rows. Similar to slices, Tiles break entropy decoding and prediction dependencies, but does not require a slice header for each tile.

For slices, entropy slices, and Tiles the number of partitions can be freely chosen by the encoder. In general having more partitions increases parallelism but also results in lower compression efficiency. For WPP, the number of partitions is fixed to one per row. This guarantees some amount of parallelism that grows with the resolution independent of the encoding scheme

In Table 1 the coding losses of the different approaches are presented. Slices, entropy slices and WPP are configured using one picture partition per row and compared to Tiles configurations that have (approximately) the same amount of parallelism. Having one picture partition per row for Tiles is less advantageous as entropy and prediction dependencies cannot cross tile boundaries. We compared vertical and rectangular

Resolution	1080p		1600p		2160p	
	Part.	BD-br	Part.	BD-br	Part.	BD-br
1 slice / row	17	8.65	25	8.26	34	5.15
1 ent. slice / row	17	5.59	25	6.29	34	3.88
1 WPP sub. / row	17	1.35	25	1.33	34	0.55
Tiles 1x15	15	4.41				
Tiles 4x4	16	3.45				
Tiles 1x20			20	4.97		
Tiles 5x5			25	4.32		
Tiles 6x6					36	2.08

Table 1: Number of picture partitions (Part.) and % of Y BD-rate losses for different picture partitioning approaches compared to one slice per frame

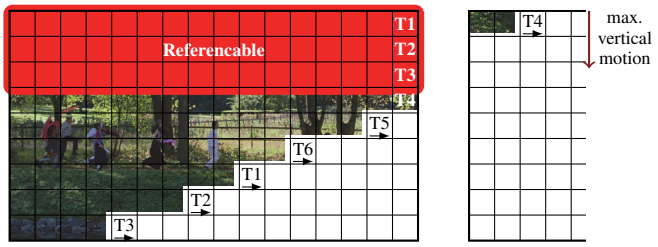

Fig. 1: Pictures can be overlapped with a restricted motion vector size, because the reference area is fully decoded.

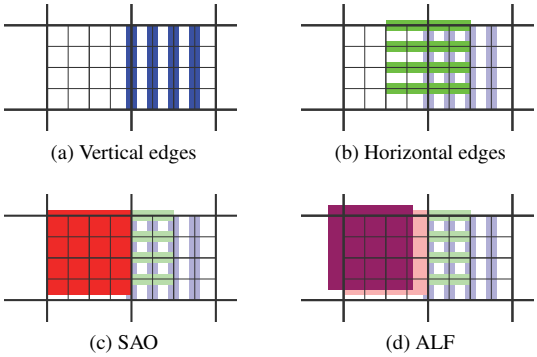

Fig. 2: Delay of CTB filtering due to pixel dependencies.

Tiles, and we observed that for a given amount of parallelism the best configuration is to have N×N tiles (N tile rows and N tile columns) in a picture. The table shows that using 1 WPP sub-stream/row gives the lowest coding losses followed by Tiles.

3. OVERLAPPED WAVEFRONT PROCESSING

Although WPP has low coding losses, wavefront processing suffers from inefficiencies due to parallelism ramp-up and ramp-down. Tiles do not suffer from this and potentially can provide better parallel efficiency.

The inefficiencies of wavefront processing can be mitigated by overlapping the execution of consecutive pictures. Figure 1 shows that when a thread has finished a row in the current picture and no more rows are available it can start processing the next picture instead of waiting for the current picture to finish. We call this technique Overlapped WaveFront (OWF).

To overlap the execution of consecutive pictures minor modifications in the codec and the decoder implementation are required. First on the codec side, the size of the motion vectors must be restricted to ensure that all the reference area is available. This can be guaranteed by limiting only the maximum downwards length of the vertical component.

On the one hand, limiting the size of the motion vectors reduces the ability of the codec to describe fast motion. On the other hand, reducing the size further would allow more decoders to be used. For our experiments we have limited the downwards size to 1/4 of the picture height. With this setting there was no compression losses observed with any of the videos, while the usable number of decoders is 12, 18, and 25 for 1080p, 1600p, and 2160p, respectively.

Another requirement for overlapped execution is that all the decoding steps leading to the final reference picture have to be performed on a Code Tree Block (CTB) granularity instead of separated picture passes. In HEVC this means that in addition to the entropy decode, reconstruction, and deblocking filter, the SAO filter and ALF must also be performed in the CTB decoding loop. Due to pixel dependencies the filter steps cannot be performed for the current CTB, but must be delayed and performed in the order depicted in Figure 2. Horizontal edge deblocking is delayed by half CTB horizontal, SAO is delayed by 1 CTB horizontal and 4 pixels vertical, and ALF is delayed by 1 CTB and 12 pixels horizontal and 8 pixels vertical.

4. PARALLEL DECODER IMPLEMENTATION

For OWF as well as Tiles a pipelined decoder organization is used as illustrated in Figure 3. It consists of 3 pipeline stages: *parse*, *issue*, and *output*. Each of these stages is performed by a different thread. The parse thread performs emulation prevention, high-level syntax parsing, and allocates an entry in the decoded picture buffer. When a slice NAL unit is detected, its payload is propagated to the issue queue. The issue thread partitions the payload into rows or tiles and sends this, encapsulated in a work unit, to the shared decoder queue. After processing a complete work unit, a decoder thread fetches a new work unit.

For OWF, the wavefront synchronization between the decoder threads is performed using a lock protected counter for each row. This counter indicates the progress in CTB count of the row. When a decoder thread finishes decoding the last picture partition in the picture, a signal is sent to the output thread, which performs the picture reordering and picture buffer management. Picture overlapping is enabled by being

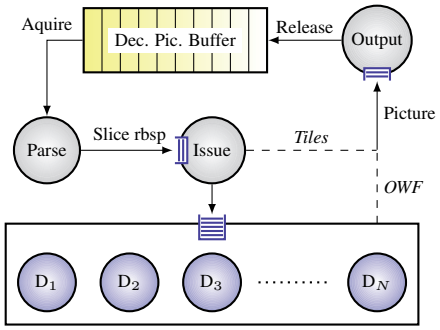

Fig. 3: General decoder architecture.

able to parse and issue the work units of the next slice NAL while the current slice is being decoded.

For Tiles, entropy decoding and reconstruction can be performed in parallel without dependencies to other tiles. The filtering stages, however, do cross tile boundaries and can therefore not be performed in the tile reconstruction loop. But in HEVC each of the filter stages (deblocking, SAO and ALF) is parallel, meaning that they can be applied to each CTB in parallel for the complete picture, one filter after the other. Our Tiles decoder is implemented by parallelizing the tiles decoding and filtering stages with a barrier between each stage.

Barrier synchronization is implemented by having the issue thread wait for the completion of each stage, after which the next stage is started. In the filter steps, eight consecutive CTBs are processed as a single task to increase spatial locality and reduce synchronization overhead. Among the decoder threads an atomic counter is used to distribute the CTBs that need to be filtered. When the last decode step has completed for the current picture, the issue thread will signal the output thread, and continues with issuing the tiles for the next picture.

5. EXPERIMENTAL SETUP

We selected the Random Access High Efficiency (RA-HE) "profile" which targets the most demanding application scenarios of the current HEVC proposal. Table 2 shows the main encoding parameters of the JCT-VC common conditions [7]. All the videos from the HEVC test sequences are encoded using these parameters with the HM-4.1 reference encoder [8]. Due to space reasons, and because we are mainly interested in high definition applications, we only present results for 1600p (2560×1600 pixels) and 1080p (1920×1080 pixels) sequences. Additionally, we also evaluated 2160p videos (3840×2160) from the SVT High Definition Multi Format Test Set.

For our parallel decoding experiments we used a cache-coherent shared memory machine with two Intel Xeon X5680 processors that have 6 cores each. Main parameters of the architecture and software environment are listed in Table 3.

Options	Value
CU structure: CTB size, partition depth	64×64, 4
Period of I-frames	32
Number of B-frames (GOP size), reference frames	8, 4
Motion estimation: algorithm, search range	EPZS, 64
Entropy coding	CABAC
Adaptive Loop Filter (ALF), Sample Adaptive Offset (SAO)	enabled
Quantization Parameter (QP)	22, 27, 32, and 37

Table 2: Coding Options

System		Software	
Processor	Intel Xeon X5680	Boost C++	1.46.1
μarchitecture	Westmere	Compiler	gcc 4.6.1
Sockets	2	Opt. level	-O3
Cores/socket	6	Kernel	3.0.0-14
Clock frequency	3.33 GHz	Operating system	Ubuntu 11.10
Last level cache	12MB / socket	HEVC software	HM-4.1-r1527
SMT & TurboBoost	disabled	Perf. counters	linux perf

Table 3: Experimental setup

6. RESULTS AND ANALYSIS

The speedup of the decoder is presented for OWF and tiles in Figure 4. The figure shows that for both OWF and tiles the speedup is higher with larger resolution sequences. Also it can be observed that OWF has higher performance compared to Tiles, and that the difference is growing with the number of threads. The speedup difference is most profound with the 1080p sequences. In Table 4 the maximum speedup and frames per second are reported. Compared to previous work [9], the absolute performance of OWF is 29.6%, 42.4%, and 66.6% higher for 2160p, 1600p, and 1080p, respectively.

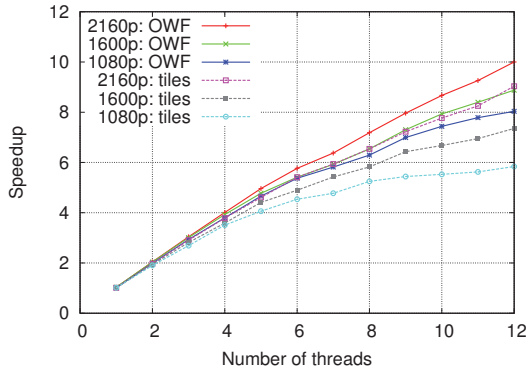

Fig. 4: Speedup for OWF and tiles

To investigate the performance difference between OWF and Tiles the speedup can be decomposed in the total CPU time increase and CPU usage factor. These metrics are derived from the linux `time` command. The total CPU time indicates the total time spend by all the threads of the program. The CPU usage factor indicates the average number of cores used during the execution. Factoring the total CPU time increase into the CPU usage factor results back into the

speedup. The relative increase of the total CPU time and the CPU usage factor are plotted for the three resolutions in Figure 5.

Video Class	2160p		1600p		1080p	
	owf	tiles	owf	tiles	owf	tiles
Max speedup	10.0	9.04	8.88	7.35	8.04	5.84
Frames / second	19.9	17.9	42.1	34.8	88.5	64.2
LLC misses / kInst.	1.65	2.35	2.17	2.84	2.22	2.79

Table 4: Speedup, frames per second and LLC misses at highest core count

Ideally the CPU usage factor is equal to the number of threads and the total CPU time does not increase. A lower CPU usage factor indicates that the parallelization strategy has scalability limitations. Increases in total CPU time originate from threading overhead, reduced cache locality, and memory access contention.

The plots for the CPU usage factor show an increasing trend with higher resolution for both Tiles and OWF, which is expected as higher resolutions sequences contain more picture partitions for both Tiles and OWF. It can, however, be observed that OWF has higher CPU usage compared to Tiles, especially at higher thread numbers and lower resolution. This shows that the scalability of the OWF parallelization strategy is higher compared to having many parallel phases separated with barriers in Tiles.

The plots for the CPU time increase show higher increases for higher thread counts. Crossing the socket boundary at 6 threads shows a jump in CPU time increase due to NUMA effects. For Tiles, however, a larger increase in the total CPU time is observed overall compared to OWF.

We have used hardware performance counters to analyze the CPU time differences between OWF and tiles. The most relevant factor is the number of Last Level Cache (LLC) misses, which are shown in Table 4 (per thousand instructions). On average, the LLC misses are 24.6% lower for OWF compared to Tiles. This confirms that doing the filter stages in a single pass improves cache locality and reduce main memory contention. Although it is possible to implement most of the filtering in a single pass for Tiles, this requires an extra boundary filtering stage afterwards. However, implementing boundary filtering is difficult when all the filters are enabled and can cross tile boundaries.

7. CONCLUSIONS

In this paper we have improved the parallelization efficiency of HEVC decoding using a new parallelization strategy called Overlapped Wavefront (OWF). This strategy improves the efficiency by allowing threads to start processing the next picture before the current picture is fully decoded. Furthermore, all the decoding steps are performed on a CTB basis

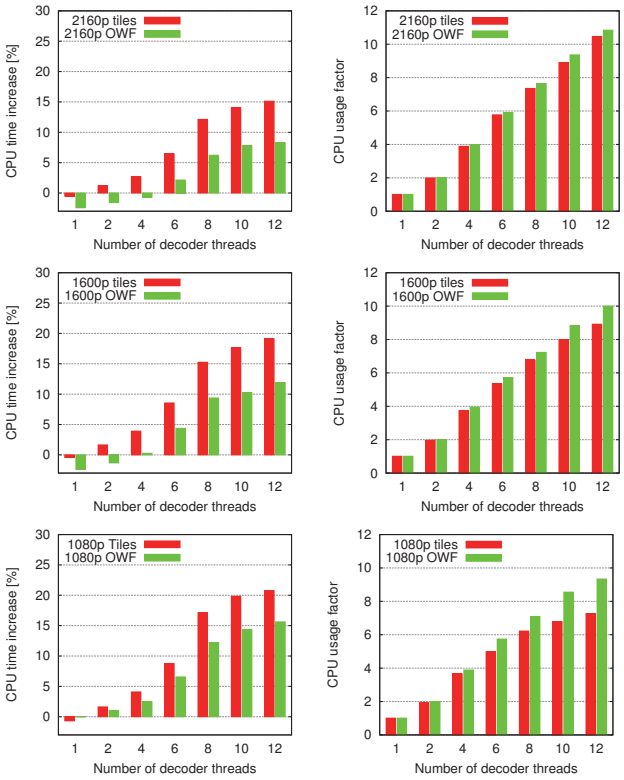

Fig. 5: CPU usage factor and CPU time increase

(rather than on a picture basis) improving the data locality. The OWF and Tiles parallelization have been implemented on top of HEVC reference software. Compared to previous work the performance has been improved by 29.6%, 42.4%, and 66.6%, and compared to Tiles the performance is 11.3%, 21.0%, 38.0% higher, for 2160p, 1600p, and 1080p, respectively.

8. REFERENCES

[1] G. J. Sullivan and J.-R. Ohm, "Recent Developments in Standardization of High Efficiency Video Coding (HEVC)," in *Proc. of SPIE*, 2010.

[2] "Advanced Video Coding for Generic Audiovisual Services. ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC)," 2003.

[3] C. C. Chi and B. Juurlink, "A QHD-capable Parallel H.264 Decoder," in *Proc. of the Int. Conf. on Supercomputing*, pp. 317–326, 2011.

[4] K. Misra, J. Zhao, and A. Segall, "Lightweight slicing for entropy coding," Tech. Rep. JCTVC-D070, Jan. 2011.

[5] F. Henry and S. Pateux, "Wavefront Parallel Processing," Tech. Rep. JCTVC-E196, March 2011.

[6] A. Fuldseth, M. Horowitz, S. Xu, and M. Zhou, "Tiles," Tech. Rep. JCTVC-E408, March 2011.

[7] F. Bossen, "Common Test Conditions and Software Reference Configurations," Tech. Rep. JCTVC-F900, Dec. 2011.

[8] "HM-4.1 Reference Software." `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-4.1`.

[9] M. Alvarez-Mesa, C. C. Chi, B. Juurlink, V. George, and T. Schierl, "Parallel Video Decoding in the Emerging HEVC Standard," in *Proc. of ICASSP 2012*, March 2012.