MULTI-CORE SOFTWARE ARCHITECTURE FOR THE SCALABLE HEVC DECODER

Wassim Hamidouche, Mickael Raulet and Olivier Déforges

IETR-INSA, UEB, UMR 6164
Rennes, 35708, FRANCE

ABSTRACT

The scalable high efficiency video coding (SHVC) standard aims to provide features of temporal, spatial and quality scalability. In this paper we investigate a pipeline and parallel software architecture for the SHVC decoder. The proposed architecture is based on the *OpenHEVC* software which implements the high efficiency video coding (HEVC) decoder. The architecture of the SHVC decoder enables two levels of parallelism. The first level decodes the base layer and the enhancement layers in parallel. The second level of parallelism performs the decoding of both the base layer and enhancement layers in parallel through the HEVC high level parallel processing solutions, including tile and wavefront. Up to the best of our knowledge, it is the first real time and parallel software implementation of the SHVC decoder. On an Intel Xeon processor running at 3.2 GHz, the SHVC decoder reaches the decoding of 1600p enhancement layer at 40 fps for x1.5 spatial scalability with using six concurent threads.

***Index Terms*—** HEVC, Scalable HEVC, High level parallel processing and wavefront parallel processing.

1. INTRODUCTION

The high efficiency video coding (HEVC) standard was finalized in January 2013 by the Joint Collaborative Team on Video Coding (JCT-VC) as joint effort between ITU-T and ISO/IEC [1]. HEVC standard can reach the same subjective video quality as its predecessor H.264/AVC at about a half bitrate [2]. This gain is obtained thanks to new tools adopted in the HEVC standard, such as quadtree-based block partitioning, large transform and prediction blocks, accurate intra/inter predictions and the in-loop sample adaptive offset (SAO) filter. Moreover, HEVC standard was designed with a particular attention to complexity, where several steps can be easily peformed in parallel [3, 4]. These tools allow to leverage multicore processors and achieve a real time encoding/decoding of high resolution videos (HD, 4K2K and 8K4K). The JCT-VC is currently developing the scalable extension of the HEVC standard (SHVC). The objective behind SHVC is to provide features of temporal, spatial and quality (SNR) scalability with a simple and efficient coding architecture [5]. Since the temporal scalability is enabled in HEVC with a hierarchical temporal prediction structure, SHVC concentrates on spatial and SNR scalability. Several scalable solutions [6, 7, 8, 9] were proposed as a response to the SHVC call for proposal [5]. The approved approach is based on multi-loop decoding structure (i.e. all intermediate layers need to be decoded) and uses the same technologies of HEVC with an inter-layer prediction to improve the coding efficiency. This solution allows a gain of 15%-30% in terms of rate-distortion compared to a simulcast coding solution. The HEVC standard defines three main concepts, including tile, slice and wavefront, that enable a high level parallel processing of both encoding and decoding [3]. Tile and slice concepts break at their boundaries the dependencies of both intra predictions and the probabilities of the context-adaptive binary arithmetic coding (CABAC). This allows to encode/decode slices or tiles of one frame on separate cores. However, the intra prediction limitation and the initialization of the CABAC context decrease the coding performance in terms of rate distortion, especially for large number of tiles/slices per frame. Moreover, the in-loop filters cannot be performed in parallel at the tile/slice edges without extra control mecanisme. The wavefront parallel processing (WPP) solution was proposed to the HEVC standard in [10]. The WPP concept enables the decoding of several coding tree block (CTB) rows in parallel. This is possible by the initialization of the CABAC context at the start of each CTB row. The overhead caused by this initialization is limited since the CABAC context at each CTB row is initialized by the CABAC context state at the second CTB of the previous CTB row. Therefore, the decoding of each CTU row can be carried out on separate threads with a minimum delay of two CTBs between adjacent CTB rows. Authors in [11] proposed a real time and parallel implementation of the SHVC decoder. This decoder is CTB groups based parallelism and performs a real time decoding of 1080p50 enhancement layer (EL) for x1.5 spatial scalability on an Intel i7 processor with using 8 concurent threads.

In this paper we propose a pipeline and parallel architecture for the SHVC decoder [12]. This architecture enables two levels of parallelism where the base layer (BL) and the EL frames are decoded in parallel thanks to the pipeline architecture. The parallel architecture, as a a second level of parallelism, performs the decoding of each frame in parallel with WPP solution. The SHVC decoder is based on the *Open-*

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

HEVC software [13] which implements a conforming HEVC decoder. On an Intel Xeon processor running at 3.2 GHz, the SHVC decoder reaches the decoding of 1600p EL at 40 frames per second (fps) for x1.5 spatial scalability with using 6 concurent threads.
This paper is organized as follows. Section 2 describes the architecture of the *OpenHEVC* decoder enabling wavefront parallel processing solution. The pipeline and parallel architecture of the SHVC decoder is investigated in Section 3. The performance of the SHVC decoder is assessed and discussed in Section 4, and finally Section 5 concludes this paper.

2. PARALLEL SINGLE LAYER HEVC DECODER

2.1. WPP solution under the OpenHEVC decoder

The architecture of the *OpenHEVC* software is quite simple and is based on a coding tree unit (CTU) decoding. The proposed WPP implementation performs all decoding steps at the level of the CTU in a single pass. Figure 1 shows an overview of the *OpenHEVC* architecture. The *hls_decode_row* function decodes all CTUs of one row in the slice. It browses in raster scan the CTUs within the row and calls the recursive function *hls_coding_tree* to decode each CTU. There are specific functions that handle the prediction and the transform of the prediction and the transform units, namely *hls_prediction_unit* and *hls_transform_unit*, respectively. Once all coding units (CU) within a CTU are decoded, the deblocking filter (DF) and then the SAO filter are performed on the decoded CTU. However, when performing the DF of the current CTB, the right and the down CTB neighborhoods are not available (ie. not yet decoded). Therefore, the right and down edges of the current CTB are filtered when its right and down CTBs are being filtered, respectively. In this solution, the DF and the SAO filters are delayed with one CTU and one CTU row for only the right and the down edges of a CTB, respectively.
The WPP extension in the *OpenHEVC* architecture is straight forward. This is possible by running the *hls_decode_row* function on separate threads to decode several adjacent CTU rows in parallel. The delay in terms of CTU, noted d, required by the wavefront solution between two adjacent CTU rows is managed by an integer type array shared by all threads. The i^{th} value of the array is used to count the number of decoded CTUs within the i^{th} CTU row. Thus, the *hls_decode_row* function increments the related array value for each decoded CTU and decodes a new CTU only if the d next CTUs of the previous CTU row are decoded.

2.2. Analytical performance of the WPP solution

The analytical speedup of the WPP solution represents the upper bound of its experimental performance. It also gives an idea on the parameters that control the performance of the wavefront solution. Let us consider x the number of CTB columns, y the number of CTB rows and d the delay in terms

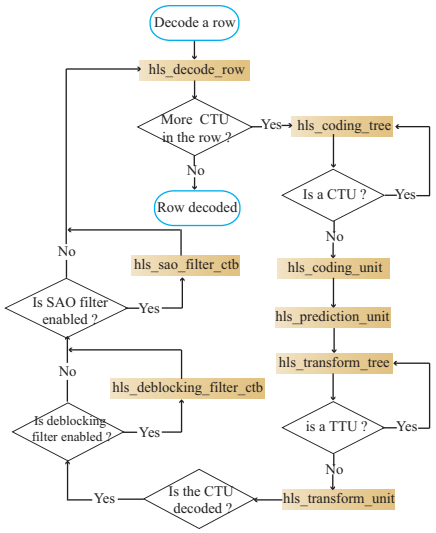

Fig. 1. Blocks diagram of the OpenHEVC decoder

in CTB between two adjacent CTB rows required by the wavefront solution. The effective number of threads n used in the wavefront solution is given as follows:

$$n = \min\left(nb\_cpu\_threads, \left\lfloor \frac{x}{d} \right\rfloor\right) \quad (1)$$

where $d \in \mathbb{N}^+$ and $nb\_cpu\_threads$ is the number of threads selected to decode the video sequence.
The analytical speedup γ is derived as follows:

$$\gamma = \begin{cases} \frac{xy}{\frac{xy}{n}+d(n-1)}, & \text{if } y \bmod n = 0 \\ \frac{xy}{x\left\lceil \frac{y}{n} \right\rceil + d((x \bmod n)-1)}, & \text{if } y \bmod n \neq 0 \end{cases} \quad (2)$$

where $x, y, n \in \mathbb{N}^+$.
We can notice from Equation 2 that the analytical speedup depends on three main parameters: the video resolution (x and y), the effective number of threads (n) and the CTB delay between two adjacent CTB rows (d). In addition, a large division remainder between the number of CTB rows and the effective number of threads ($y \mod n$) also decreases the performance of the wavefront solution. The division remainder corresponds to the inactive threads waiting for the decoding of the last CTB rows of the frame.

3. PIPELINE AND PARALLEL SHVC DECODER

3.1. Overview of the SHVC standard

In the case of spatial scalability with two layers, the SHVC encoder consists of two encoders, one for each layer. The BL HEVC encoder encodes the downsampled version of the original video and feeds the second encoder with the decoded picture and the corresponding motion vectors (MVs). The EL

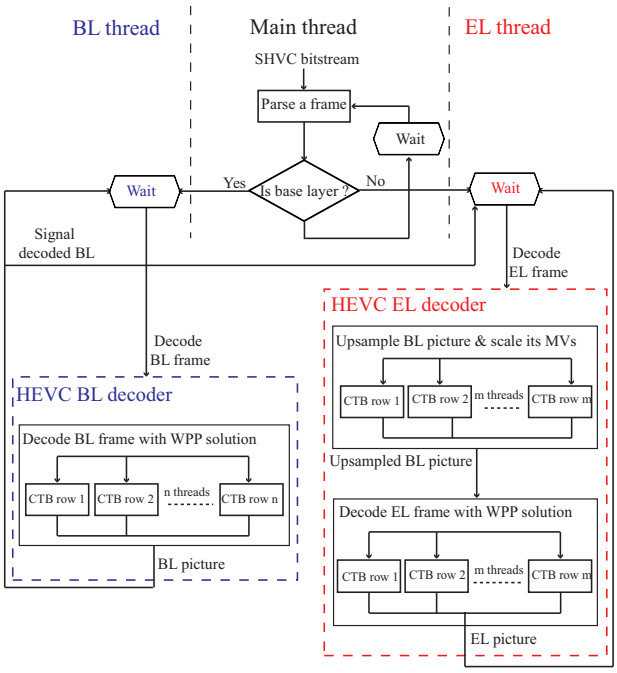

Fig. 2. Pipeline and parallel architecture of the SHVC decoder for spatial scalability with two layers

SHVC encoder encodes the original video with using the upsampled BL picture and its upscaled MVs for inter-layer prediction. Concerning the SNR scalability, the encoding process remains unchanged except that the BL picture and its MVs are not upsampled and upscaled, respectively, at the EL encoder. The SHVC standard also supports a BL coded with H.264/AVC encoder. In this case, only the decoded BL, without its MVs, is provided to the SHVC EL encoder.

3.2. Real time and pipeline SHVC decoder

In this section we introduce a pipeline and parallel architecture for the SHVC decoder. The first step consists in extending the *OpenHEVC* software to support the new operations introduced in the SHVC standard, namely upsampling of the decoded BL frame, scaling its MVs and managing the upsampled BL picture as an additional reference picture in the EL decoder. Thus, the SHVC decoder consists of l instances of the *OpenHEVC* decoder, one HEVC decoder for each layer, with $l = 1, ..., L$ the number of layers.

The SHVC decoder enables two levels of parallelism. The first level performs the decoding of the BL and the EL frames simultaneously on separate threads. For each decoder, the second level of parallelism carries out the decoding of both the BL and the EL frames in parallel through the HEVC high level parallel processing solutions. Moreover, the upsampling of the BL and the upscaling its MVs are also carried out in parallel. In fact, when parallel decoding of the EL frame is enabled, the CTB rows of the BL picture and the corresponding MVs are upsampled and upscaled in parallel. Figure 2 summarizes the proposed pipeline and parallel architecture of the SHVC decoder for spatial scalability with two layers ($l = 2$). As illustrated in Figure 2, two instances of *OpenHEVC* decoder are created and run on separate threads. These two decoders correspond to the HEVC BL decoder and SHVC EL decoder, respectively. The parser running on the main thread parses the SHVC bitstream and feeds the two decoders with the corresponding frame (access units). For the first decoded frame, the EL decoder waits until the BL frame has been decoded and then the EL frame i is simultaneously decoded with the next BL frame (frame $i+1$). To limit the BL buffer to one frame, the BL decoder might not decode the frame $i+2$ since the EL frame i has not yet been decoded.

System		Software	
Processor	Intel Xeon	Compiler	GCC-4.6
	E5-1650	OS	Ubuntu 12.04
ISA	X86-64	Kernel	3.5.0-34
Clock frequency	3.2 GHz	OpenHEVC	cff4b48a94
Level 3 cache	12 MB	release	(based on HM11.0)
Cores	6		

Table 1. Configuration of the experiments

4. RESULTS AND DISCUSSIONS

4.1. Experimental configuration

We run the SHVC decoder on a computer fitted with 6 cores Intel Xeon processor. Table 1 summarizes the system and software configurations used to carry out the experiments. Concerning the video coding configuration, the common test conditions defined in the HEVC standard [14] were considered. In order to show the performance for larger resolution video, we added to the set of conformance video sequences two 3840×2160 video sequences from the STV High Definition Multi Format Test Set. All the selected video sequences were encoded with SHVC reference software [15] in two layers ($l = 2$) and two scalability configurations were considered: x2 and x1.5. The SHVC video sequences were coded in low delay coding configuration with enabling the wavefront feature where the delay $d = 2$. The quantization parameter (QP) of the BL was set to 27 and 32, while the QP of the EL is equal to the BL QP minus 2. The performance of the proposed pipeline and parallel SHVC decoder is compared to the sequential SHVC decoder: the decoding of the BL and the EL frames are carried out in sequential order. The number of threads used to decode the BL and the EL are noted n and m, respectively. In addition to the single thread configuration ($m = n = 1$), the number of threads in the sequential SHVC decoding configuration is set as follows $n \in \{2, 3, 4, 5, 6\}$ with $m = n$. The corresponding decoding configuration in the pipeline SHVC architecture is the follow-

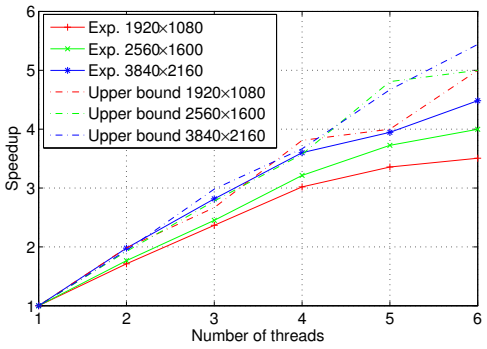

Fig. 3. Speedup performance of the WPP solution (QP=32)

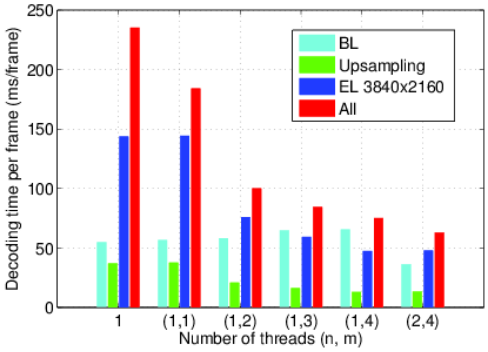

Fig. 4. Decoding time performance of the SHVC decoder

ing $(n, m) \in \{(1,1),(1,2),(1,3),(1,4),(2,4)\}$.

4.2. Results

Figure 3 illustrates the speedup of the WPP implementation under the single layer *OpenHEVC* decoder for different video resolutions. The experimental speedup is compared to the upper bound performance of the wavefront solution computed with Equation 2. We can notice that the experimental speedup is close to the optimal speedup, especially when the number of threads is below 5. With using 6 threads, the performance of the proposed implementation decreases and reaches an accelerating factor of 4.5 for 3840x2160 resolution videos, instead of the upper bound value of 5.5. This is because we use the maximum number of CPU cores including the one used by the operating system. Figure 4 shows the performance of the pipeline SHVC architecture in terms of decoding time per frame for the three main decoding steps: BL decoding, upsampling the BL picture and the EL decoding. We can notice that the decoding time of the three steps remain constant with using one thread and two threads (two decoders in parallel and n = m = 1). In these two decoding configurations each decoding step is performed on a single thread. However, the whole decoding time, in the later configuration, decreases by the BL decoding time since the decoding of the BL and the EL are performed in parallel in the pipeline architecture. For number of threads between 3 and 5, we only increases the number of threads for the EL, since the decoding time of the EL is higher than the decoding time of the BL and the WPP solution is more efficient with large video resolutions. Figures 5 compares the performance of sequential and pipeline SHVC decoders in terms of decoding frame rate for different video resolutions. We can notice that the pipeline architecture is more efficient for videos of low resolutions (EL 1080p). In fact, the pipeline parallelism enables adjusting the number of threads for each layer by providing more threads to the EL for which the WPP solution is more efficient. However, for high resolution videos (EL 1600p and 2160p) the performance of the pipeline and sequential architecture is similar. The performance of the pipeline architecture decreases when the number threads is equal to 2 (1,1). Indeed, the decoding time of the EL including the upsampling is much higher than the decoding time of the BL. Therefore, running in parallel the BL and the EL decoders both on a single thread is less efficient than the WPP with 2 threads for each step in sequence. The pipeline solution could provide better performance for SHVC bitstreams with more than one EL since the decoding time of the ELs is similar.

5. CONCLUSION

In this paper we proposed a multiple threads architecture for the SHVC decoder. This decoder enables two levels of parallelism where the decoding of the SHVC layers is pipelined, and each layer is decoded in parallel based on the HEVC high level parallel processing solutions. The first end-to-end video demonstration using the proposed real time SHVC decoder within the GPAC player was presented in the 106^{th} MPEG meeting [16].

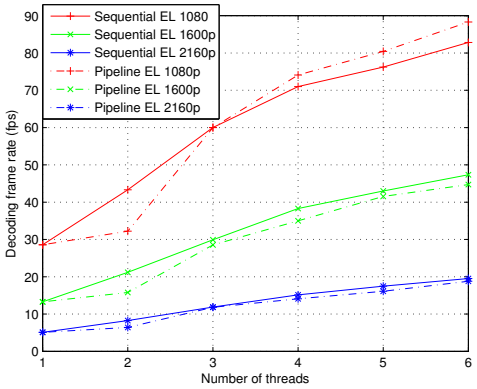

Fig. 5. Decoding frame rate performance of the SHVC decoder (QP=32)

6. REFERENCES

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1648–1667, December 2012.

[2] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparaison of the Coding Efficiency of Video Coding standards including High Efficiency Video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1969–1684, December 2012.

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

[4] J. F. Bossen, B. Bross, K. Suhring, and D. Flynn, "Hevc complexity and implemnation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1685–1696, December 2012.

[5] ISO/IEC-JTC1/SC29/WG11 and ITU-T-SG16, "Joint Call for Proposals on Scalable Video Coding Extensions of High Efficiency Video Coding (HEVC)," in *ISO/IEC JTC 1/SC 29/WG11 (MPEG) Doc. N12957 or ITU-T SG 16 Doc. VCEG-AS90*. Stockholm, Sweden, July 2012.

[6] Z. Shi, X. Sun, and F. Wu, "Spatially Scalable Video Coding for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1813–1826, December 2012.

[7] P. Helle, H. Lakshman, M. Siekmann, J. Stegemann, T. Hinz, H. Schwarz, D. Marpe, and T. Wiegand, "A Scalable Video Coding Extension of HEVC," in *IEEE Conference on Data Compression*, March 2013, pp. 201–210.

[8] J. Chen, K. Rapaka, X. Li, V. Seregin, L. Guo, M. Karczewicz, G. V. Auwera, J. Sole, X. Wang, C. Tu, Y. Chen, and R. Joshi, "Scalable Video Coding Extension for HEVC," in *IEEE Conference on Data Compression*, March 2013, pp. 191–200.

[9] Z. Zhao, J. Si, J. Ostermann, and W. Li, "Inter-layer Intra Mode Coding for the Scalable Extension of HEVC," in *IEEE International Symposium on Circuits and Systems*, May 2013, pp. 1636–1639.

[10] G. Clare, F. Henry, and S. Pateux, "Wavefront parellel processing for HEVC Encoding and Decoding," in *document JCTVC-F274*. Torino, Italy, Jully 2011.

[11] S. Gudumasu, Y. He, Y. Ye, and Y. He, "Real time SHVC software decoding with multi-threaded parallel processing," in *document JCTVC-O0165*. Geneva, Switzerland, October 2013.

[12] W. Hamidouche, M. Raulet, and O. Deforges, "Pipeline and parallel architecture for the SHVC decoder," in *document JCTVC-O0115*. Geneva, Switzerland, October 2013.

[13] "Open source HEVC decoder (OpenHEVC)," in *https:://github.com/OpenHEVC*.

[14] F. Bossen, "Scalable high efficiency video coding test model 3 (SHM 3)," in *document JCTVC-H1100*. 8th Meeting: San Jose, CA, USA, February 2012.

[15] "SHVC Reference Software (SHM): https://hevc.hhi.fraunhofer.de/svn/svn_SHVCSoftware/," .

[16] W. Hamidouche, J. Le Feuvre, and M. Raulet, "A scalable HEVC demonstration within GPAC player," in *document MPEG-m31397*. Geneva, Switzerland, October 2013.

[17] "H2B2VS project: http://h2b2vs.epfl.ch," .