The Journal of Supercomputing manuscript No.
(will be inserted by the editor)

Slice-based parallel approach for HEVC encoder

P. Piñol · H. Migallón · O. López-Granado · M.P. Malumbres

Received: date / Accepted: date

Abstract The HEVC is the very last video coding standard that significantly increases the computing demands to encode video to reach the limits on compression efficiency. Our interest is centered on applying parallel processing techniques to HEVC encoder in order to significantly reduce the computational time without disturbing the coding performance behavior. We propose a parallelization approach to the HEVC encoder which is well suited to multicore architectures. Our proposal uses OpenMP programming paradigm working at slice parallelization level. We encode several slices of each frame at the same time using all available processing cores. The results show that speed-ups up to 9.8 can be obtained for the All Intra mode and up to 8.7 for Low-Delay B, Low-Delay P and Random Access modes with a negligible loss in coding performance.

Keywords Parallel algorithms, video coding, HEVC, multicore, performance

This research was supported by the Spanish Ministry of Education and Science under grant TIN2011-27543-C03-03, the Spanish Ministry of Science and Innovation under grant TIN2011-26254.

P. Piñol
Physics and Computer Architecture Dept. Miguel Hernández University, 03202 Elche, Spain.

H. Migallón
Physics and Computer Architecture Dept. Miguel Hernández University, 03202 Elche, Spain.
Tel.: +34-966658390
Fax: +34-966658814
E-mail: hmigallon@umh.es

O. López-Granado
Physics and Computer Architecture Dept. Miguel Hernández University, 03202 Elche, Spain.

M.P. Malumbres
Physics and Computer Architecture Dept. Miguel Hernández University, 03202 Elche, Spain.

1 Introduction

The new High Efficiency Video Coding (HEVC) standard has been recently developed by the Joint Collaborative Team on Video Coding (JCT-VC) which was established by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG). This new standard will replace the current H.264/AVC [1] standard in order to deal with nowadays and future multimedia market trends. 4K definition video content is a nowadays fact and 8K definition video will not last to become a reality too. The HEVC standard aims to improve coding efficiency with respect to the H.264/AVC High profile, delivering the same video quality at half the bit rate [2].

Regarding complexity, HEVC encoder is expected to be several times more complex than H.264/AVC encoder, so reducing its complexity will be a hot research topic in years to come. At the time of developing this work, the current version of the reference software, called the HEVC test model (HM), is HM 10.0 [3] which corresponds to the HEVC text specification draft 10 [4]. A good overview of HEVC standard can be found in [5].

We can find in the literature several works about complexity analysis and parallelization strategies for the emerging HEVC standard as in [6] [7] [8].

Most of the available HEVC parallelization proposals are focused in the decoding side, looking for the most appropriate parallel optimizations at the decoder that provide real-time decoding of High-Definition (HD) and Ultra-High-Definition (UHD) video contents [7]. In [9] and [10] authors presents a variation of Wavefront Parallel Processing (WPP) called Overlapped Wavefront (OWF) for the HEVC decoder in where the executions over consecutive pictures is overlapped. When a thread finish processing a Coding Tree Block (CTB) row, it can begin with the next picture instead of waiting until the end of the current picture. Recently, in [11] authors mixed Tiles, WPP and SIMD instructions to develop a real-time HEVC decoder.

Currently, there are few works focused at the HEVC encoder. In [12] authors propose a fine-grain parallel optimization in the motion estimation module of the HEVC encoder allowing to perform the motion vector prediction in all prediction units (PUs) available at the Coding Unit (CU) at the same time. In [13] authors propose a parallelization inside the Intra prediction module that consist on removing data dependencies among subblocks of a CU, obtaining interesting speed-up results. Other recent works are focused at changes in the scanning order. For example, in [14] authors propose a CU scanning order based on a diamond search obtaining a good scheme for massive parallel processing. Also in [15] authors propose to change the HEVC deblocking filter processing order obtaining time savings of 37.93% over many-core processing.

In this paper we present a parallelization at slice level for the HEVC encoder. In this approach, the number of slices per frame varies as a function of the number of processes used to encode the sequence. Depending on the number of processes used, each slice will contain a different number of consecutive CUs. Furthermore, we analyze the overall behavior in terms of complexity reduction and coding performance.

The remainder of this paper is organized as follows: Section 3 shows the proposed slice based parallelism strategy. In Section 4 we analyze the performance of the proposed parallel algorithm. Finally, in Section 5 some conclusions are drawn.

2 HEVC and slices

HEVC is a video codec that uses a block-based hybrid video coding scheme. This scheme is based in motion estimation/compensation and transform coding. In the encoding process, each frame is divided into blocks called Coding Units (CU). The maximum size of a CU in HEVC is 64x64 pixels. A CU can be divided into smaller blocks for the encoding process. These blocks can be encoded using one of three modes: (a) without any prediction, (b) with spatial prediction, or (c) with temporal prediction. Spatial prediction exploits spatial redundancy within a frame. In order to encode a block, it uses previous regions of the same frame to calculate a block candidate and then encodes only the error of that estimation. Temporal prediction uses previously encoded frames to estimate the block candidate, by means of a search of a similar block in other frames (called reference frames). It exploits temporal redundancy, taking advantage of the fact that nearby frames usually contain blocks which are very similar to the current block and so the residuum of the compensation is close to zero. For the three encoding modes a Discrete Cosine Transform (DCT) is applied to the resulting values so they are converted into the frequency domain. Then these coefficients are quantized and the result is compressed by an entropy encoder.

A block is referred to as “intra” when no information of other frames is used to encode it (modes (a) and (b)). A block is refered to as “inter” when temporal prediction is used (mode (c)).

Slices are partitions of an encoded frame which can be independently decoded (regarding the other slices of the same frame). As a slice can be decoded without any information from other slices, neither intra prediction nor inter prediction can cross slice boundaries using CUs from other slices of the same frame. So, every single slice can be encoded/decoded in an isolated way. Slices consist of an integer number of CUs. An I-slice is composed of intra CUs. P-slices and B-slices can contain both intra and inter CUs. Here ‘P’ stands for unidirectional prediction and ‘B’ stands for bidirectional prediction. CUs in a P-slice can only use one reference frame. CUs in a B-slice can use up to two reference frames, i.e. the estimation and compensation mechanism can use the combination of up to two blocks in different reference frames. One frame can be composed of only one slice or several slices.

In the configuration files provided within reference software package [3], four coding setups can be found: All Intra, Random Access, Low-Delay B, and Low-Delay P.

In All Intra (AI) setup every frame is coded as an I-frame (all its slices are I-slices), i.e. it is coded without any motion estimation/compensation. So each

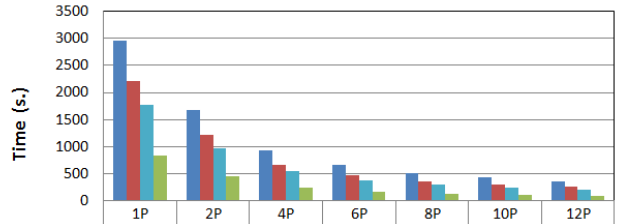
frame is independent from the other frames in the sequence. This mode gets lower compression rates (compared to the other 3 setups) because P-frames and B-frames can usually obtain better compression rates than I-frames at the same quality level. On the other hand, the encoding process for AI mode is faster than for the other 3 setups because no time is wasted in motion estimation. Every frame is coded in rendering order. Applications that require a fast encoding process and are not concerned about limited bandwidth or storage capacity, fit perfectly in this coding setup.

Random Access (RA) setup combines I-frames and B-frames in GOPs (Group Of Pictures) of 8 frames. B-frames usually achieve good compression rates. Each block of a B-frame can use up to 2 reference frames, so in the encoding process 2 lists of reference pictures are maintained. Reference frames are located earlier and later than the frame we are currently coding. In this setup, coding (and decoding) order is not the same as rendering order. To allow navigating along the coded sequence (pointing to a certain moment) or to allow functions like fast forward, an I-frame is inserted periodically. Depending on the frame rate of each sequence the intra refresh period varies. The intra period is a multiple of 8 (the size of the GOP) which inserts an I-frame approximately every second. Applications that do not have time constraints (when coding a video sequence) and need features like the aforementioned fast forward, are the target applications of this coding mode.

Low-Delay setups (LP and LB) encode each frame in rendering order. First an I-frame is inserted in the coded bit stream and then only P-frames (or B-frames) are used for the rest of the video sequence. As stated before, a B-frame uses a pair of frames to perform motion estimation and compensation. In LB mode this pair of frames is always selected from previous frames (in rendering order). On the contrary, RA mode uses past and future frames as reference for B-frames and this introduces a delay because you have to wait for future pictures to be available in order to encode/decode the present frame. In LP and LB setups GOP size is 4. This two modes achieve better compression performance than AI mode and do not suffer from the delay introduced by RA mode. Applications like video-conference which have bandwidth and time constraints can benefit from low delay modes.

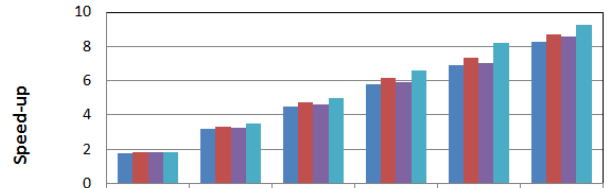
3 Parallel algorithms

In our parallel proposal of HEVC encoder we have divided each video frame in as many slices as the available number of parallel processes in such a way that if we are using 12 parallel processes we will divide each frame into 12 slices and each slice will be encoded by a different process. In RA, LP and LB setups, a frame used as a reference picture has to be available as a whole for every future slice that uses it for motion estimation/compensation. A synchronization mechanism is imposed by this restriction. Every process encodes a different slice in a parallel way but all those processes need to be synchronized before encoding the next frame. In AI mode this is not strictly necessary as there are



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	2964.7	1672.9	937.1	661.9	512.3	430.3	358.1
Random	2219.8	1215.4	668.7	467.8	361.0	302.5	255.8
LowDelay-P	1781.0	972.6	545.9	388.0	301.0	252.8	207.1
Intra	842.4	458.7	242.9	169.5	128.2	102.7	91.3

(a) Computational times.



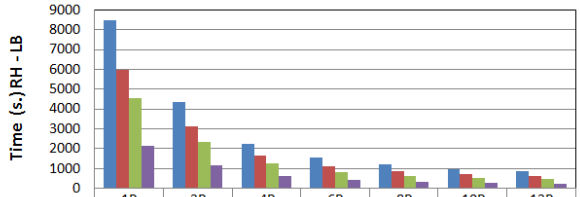
	2P	4P	6P	8P	10P	12P
LowDelay-B	1.8	3.2	4.5	5.8	6.9	8.3
Random	1.8	3.3	4.7	6.1	7.3	8.7
LowDelay-P	1.8	3.3	4.6	5.9	7.0	8.6
Intra	1.8	3.5	5.0	6.6	8.2	9.3

(b) Speed-ups.

Fig. 1 Computational times and speed-ups for the parallel algorithm encoded in AI, LB, LP and RA modes processing 120 frames with different number of processes at QP=32.

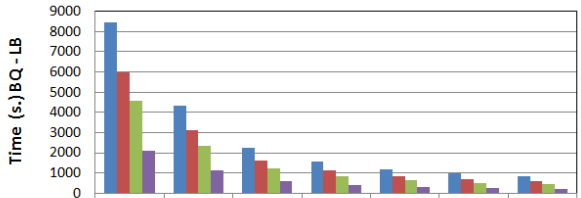
no reference pictures but we have followed the same synchronizing scheme for AI setup. In RA, LP and LB setups all decoded slices of the reference frames need to be available to encode the following frame, so to avoid unnecessary sendings of slices or decoded pieces of a frame between processes, we have used a shared memory approximation to store the Decoded Picture Buffer.

In our tests we have used three video sequences named "Four People (FP)" (which has a resolution of 1280x720 pixels), "Race Horses (RH)" (with a resolution of 832x480 pixels) and"BQ Terrace (BQ)" (with a resolution of 1920x1280 pixels). We have performed different tests by dividing frames into 1, 2, 4, 6, 8, 10 and 12 slices and by using the corresponding number of parallel processes. Note that in our experiments we obtain slices which have a similar number of CUs each, but this is absolutely not a restriction. We have chosen a uniform partition of the frame in order to homogenize the time that each process spends in encoding its slice so as to reduce the dead time of the processes (caused by the synchronization mechanism). Nevertheless we have observed that the time that each process needs to encode its slice can diverge even if slices have all



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(a) RH video sequence.



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(b) BQ video sequence.

Fig. 2 Computational times for RH and BQ video sequences processing 120 frames with different number of processes at QP=32.

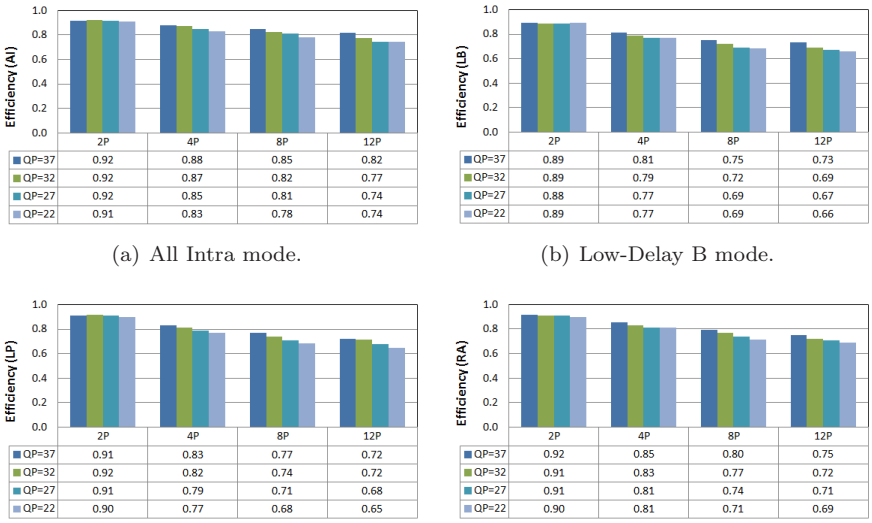
the same size. The reason is that algorithms like motion estimation or entropy encoding can differ in processing time for different regions of a frame.

4 Numerical experiments

The proposed algorithms have been tested on a shared memory platform evaluating parallel performance, PSNR and bit rate. The multicore platform used consists of two Intel XEON X5660 hexacores at up to 2.8 GHz and 12MB cache per processor, and 48 GB of RAM, with no hyperthreading enabled. The operating system used is CentOS Linux 5.6 for x86 64 bit. The parallel environment has been managed using OpenMP [16]. The compiler used was *g++* compiler *v*.4.1.2.

As we have said the testing video sequences used in our experiments are RH, FP and BQ, and we present results using LB, LP, RA and AI modes, encoding 120 frames at different Quantization Parameters (QPs) (22,27,32,37).

In Fig. 1 we present both the computational times and the speed-ups for the parallel algorithm using AI, LB, LB and RA modes for the FP video



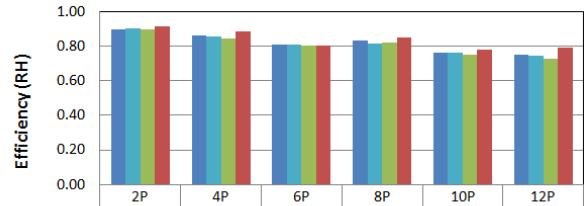
(a) All Intra mode. (b) Low-Delay B mode.

(c) Low-Delay P mode. (d) Random Access mode.

Fig. 3 Efficiencies for the FP video sequence with different number of processes and QPs processing 120 frames.

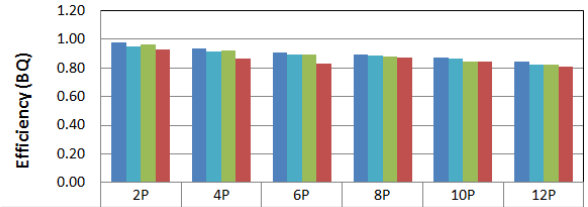
sequence, processing 120 frames. As it can be seen the proposed parallel algorithm obtains speed-ups up to 9.3 for 12 cores in AI mode and efficiencies of 0.8 on average. We can observe that AI mode obtains better efficiencies than LB, LP and RA modes because motion estimation/compensation process applied in LB, LP and RA modes produces different residual data on each slice and therefore the time required by the entropy encoder to process that residual data in each slice may diverge from one slice to another. Fig. 2 shows computational times for RH and BQ video sequences encoded setting QP equal to 32, as expected, this figure confirms the computational behavior is similar for the three different video sequences tested, we can extend this behavior for the other values of QP.

In Fig. 3 we present a comparison of efficiencies as a function of the number of processes as well as the compression ratio. As it can be seen, good efficiencies are obtained for all modes. As previously mentioned, there is a efficiency loss as the number of cores increases because of the different time required to encode each slice, note that, the parallel algorithm offers good scalability, i.e. the efficiency loss is low. However, that efficiency loss is lower at high compression ratios, where residual data tends to be more similar in each slice after a higher quantization and so, a similar time is required by each slice to encode that residual data. Therefore, the time spent by some threads waiting for the slowest one is reduced. On average, efficiencies of 0.75, 0.78, 0.79 and 0.83 are obtained for LB, LP, RA and AI mode respectively. Fig. 4 shows efficiencies when RH and BQ video sequences are encoded setting QP equal to 32, obtaining similar



	2P	4P	6P	8P	10P	12P
LowDelay-B	0.90	0.86	0.81	0.83	0.76	0.75
Random	0.91	0.86	0.81	0.82	0.76	0.75
LowDelay-P	0.90	0.85	0.80	0.82	0.75	0.73
Intra	0.92	0.89	0.80	0.85	0.78	0.79

(a) RH video sequence.



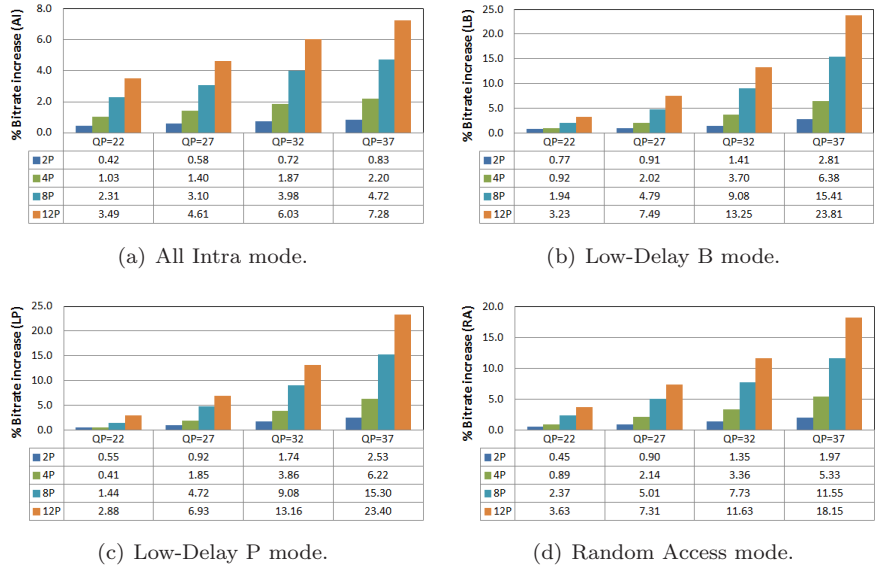
	2P	4P	6P	8P	10P	12P
LowDelay-B	0.98	0.94	0.91	0.89	0.87	0.84
Random	0.95	0.92	0.90	0.88	0.86	0.82
LowDelay-P	0.96	0.92	0.89	0.88	0.84	0.82
Intra	0.93	0.86	0.83	0.87	0.84	0.81

(b) BQ video sequence.

Fig. 4 Efficiencies for the RH and BQ video sequences at QP=32, processing 120 frames.

results for the RH video sequence. Also, it can be noticed that the results improve for a high resolution video sequence (BQ).

In Fig. 5 we show the bit-rate overhead as a function of the number of processes and the quantization parameter (QP). As it can be seen, all modes increase the resulting bit-rate as the number of slices and the compression ratio increase. In AI mode that overhead appears due to the higher number of headers used and the reduced number of CUs available for the intra prediction process at each slice. As shown in Fig. 5(a), the maximum bit-rate increment for AI mode is 7.28% with 12 slices per frame and a QP value of 37, being on average lower that 3%. In the rest of modes there is a similar behavior, increasing the bit-rate as the number of slices and compression ratio do. The bit-rate overhead in LB, LP and RA modes is also due to the higher number of headers and because motion vectors cannot be predicted outside the slice boundaries. On the other hand CABAC context models are not updated after the computation of each slice, being updated just after the frame computation. As shown in Fig. 5, in LB, LP and RA modes that overhead has a greater impact in the final bitstream, being the maximum bit-rate increment equal to 23.81%, 23.40% and 18.15% for LB, LP and RA modes respectively. Fig. 6



	QP=22	QP=27	QP=32	QP=37
2P	0.42	0.58	0.72	0.83
4P	1.03	1.40	1.87	2.20
8P	2.31	3.10	3.98	4.72
12P	3.49	4.61	6.03	7.28

(a) All Intra mode.

	QP=22	QP=27	QP=32	QP=37
2P	0.77	0.91	1.41	2.81
4P	0.92	2.02	3.70	6.38
8P	1.94	4.79	9.08	15.41
12P	3.23	7.49	13.25	23.81

(b) Low-Delay B mode.

	QP=22	QP=27	QP=32	QP=37
2P	0.55	0.92	1.74	2.53
4P	0.41	1.85	3.86	6.22
8P	1.44	4.72	9.08	15.30
12P	2.88	6.93	13.16	23.40

(c) Low-Delay P mode.

	QP=22	QP=27	QP=32	QP=37
2P	0.45	0.90	1.35	1.97
4P	0.89	2.14	3.36	5.33
8P	2.37	5.01	7.73	11.55
12P	3.63	7.31	11.63	18.15

(d) Random Access mode.

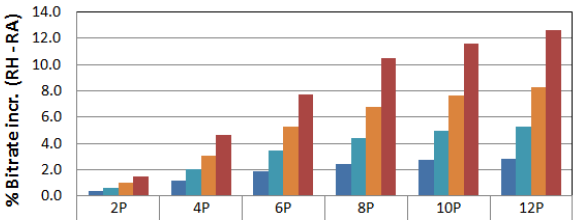
Fig. 5 Evolution of bit-rate increment (%) for FP video sequence with different number of processes as a function of the QP parameter encoding 120 frames.

presents results for RH and BQ video sequences encoded in RA mode, these results show that the bit-rate overhead is even lower in RH and BQ video sequences.

Regarding video quality, in Fig. 7 we present R/D information for all modes. These figures represent the bit-rate (vertical bars) and PSNR (horizontal lines) produced by the parallel algorithm using different number of cores (up to 12) and compressed at different QPs. As it can be seen, in all cases there are slightly differences both in PSNR and bit-rate between the sequential with one slice per frame algorithm and the parallel version with two or more slices per frame, being the maximum difference less than 0.09 dB at high compression ratios (QP=37) and an increment of bit-rate lower than 23.8% at high compression ratios using 12 processes when encoding 120 frames. As previously said, as the number of slices increases, a greater number of headers are included in the final bitstream and this has a greater impact at high compression ratios. Fig. 8 shows R/D results for RH and BQ video sequences using LP mode, note that the R/D behavior remains unchanged.

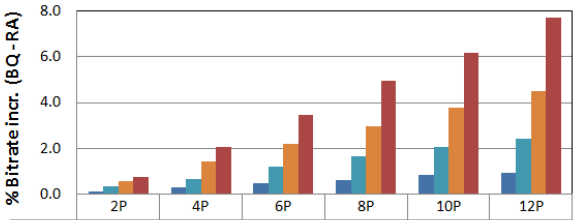
5 Conclusions

In this paper we have proposed a parallel algorithm for HEVC video encoder. This algorithm is based on a slice parallelization approach, dividing a frame into different number of slices depending on the number of processes used. After implementing the algorithms in HEVC software, some experiments were



	2P	4P	6P	8P	10P	12P
QP=22	0.37	1.20	1.88	2.42	2.73	2.83
QP=27	0.62	2.07	3.44	4.41	4.95	5.30
QP=32	0.99	3.07	5.28	6.80	7.62	8.24
QP=37	1.51	4.68	7.69	10.47	11.62	12.66

(a) RH video sequence.



	2P	4P	6P	8P	10P	12P
QP=22	0.12	0.32	0.49	0.61	0.85	0.95
QP=27	0.35	0.66	1.21	1.65	2.06	2.43
QP=32	0.58	1.45	2.20	2.96	3.77	4.50
QP=37	0.74	2.07	3.45	4.97	6.18	7.71

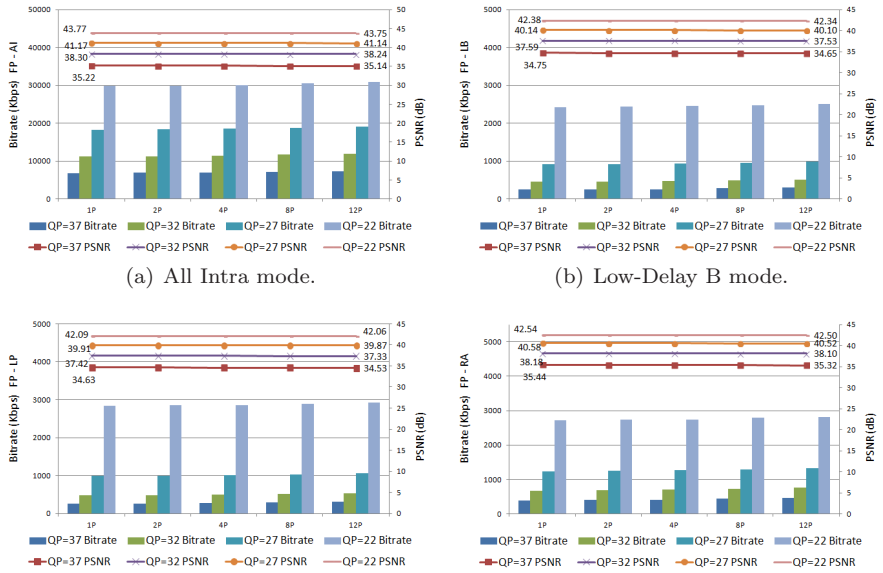
(b) BQ video sequence.

Fig. 6 Evolution of bit-rate increment (%) for RH and BQ video sequences encoded in RA mode with different number of processes as a function of the QP parameter encoding 120 frames.

performed for both All Intra and Low-Delay B modes. The results show that speed-ups up to 9.8x can be obtained for the All Intra mode and up to 8.8x for Low-Delay B mode at QP=37 when encoding 120 frames. Regarding coding performance, there is an increase in the bit-rate in both coding modes, being on average lower than 6% for the Low-Delay B mode and lower on average than 3% for the All Intra mode. Also note that the quality loss is negligible in all experiments reported. In general, the proposed version attains good parallel efficiency results, showing that parallelization approaches should be taken into account to reduce the HEVC video encoding complexity. As future work, we will explore hierarchical parallelization approaches combining GOP-based approaches with slice and wavefront parallelization levels.

References

1. ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16, 2012.*
2. J. Ohm, G. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards - including high efficiency video coding

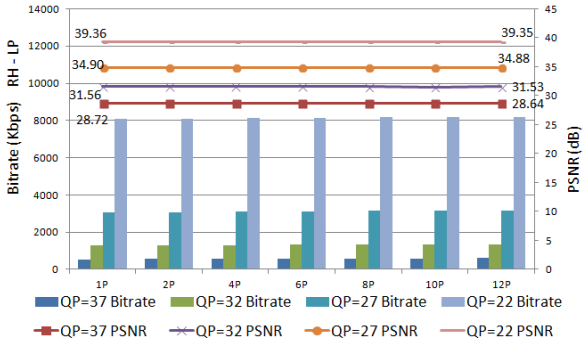


(a) All Intra mode. (b) Low-Delay B mode.

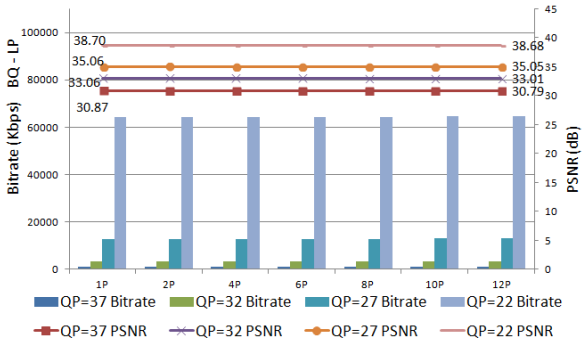
(c) Low-Delay P mode. (d) Random Access mode.

Fig. 7 R/D for FP video sequence with different number of processes, encoding 120 frames.

(hevc)," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1669–1684, 2012.
3. HEVC Reference Software, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-10.0/.
4. B. Bross, W. Han, J. Ohm, G. Sullivan, Y.-K. Wang, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 10," *Document JCTVC-L1003 of JCT-VC, Geneva*, January 2013.
5. G. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1648 –1667, December 2012.
6. F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1685–1696, 2012.
7. M. Alvarez-Mesa, C. Chi, B. Juurlink, V. George, and T. Schierl, "Parallel video decoding in the emerging HEVC standard," in *International Conference on Acoustics, Speech, and Signal Processing, Kyoto*, March 2012, pp. 1–17.
8. E. Ayele and S.B.Dhok, "Review of proposed high efficiency video coding (HEVC) standard," *International Journal of Computer Applications*, vol. 59, no. 15, pp. 1–9, 2012.
9. 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," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1827–1838, Dec 2012.
10. C. Chi, M. Alvarez-Mesa, J. Lucas, B. Juurlink, and T. Schierl, "Parallel hevc decoding on multi- and many-core architectures," *Journal of Signal Processing Systems*, vol. 71, no. 3, pp. 247–260, 2013.
11. B. Bross, M. Alvarez-Mesa, V. George, C. C. Chi, T. Mayer, B. Juurlink, and T. Schierl, "Hevc real-time decoding," pp. 88 561R–88 561R–11, 2013.
12. Q. Yu, L. Zhao, and S. Ma, "Parallel AMVP candidate list construction for HEVC," in *VCIP'12*, 2012, pp. 1–6.



(a) RH video sequence.



(b) BQ video sequence.

Fig. 8 R/D for RH and BQ video sequences, using LP mode with different number of processes and encoding 120 frames.

13. J. Jiang, B. Guo, W. Mo, and K. Fan, "Block-based parallel intra prediction scheme for HEVC," *Journal of Multimedia*, vol. 7, no. 4, pp. 289 –294, August 2012.
14. A. Luczak, D. Karwowski, S. Mackowiak, and T. Grajek, "Diamond scanning order of image blocks for massively parallel hevc compression," in *Computer Vision and Graphics*, ser. Lecture Notes in Computer Science, L. Bolc, R. Tadeusiewicz, L. Chmielewski, and K. Wojciechowski, Eds., vol. 7594. Springer Berlin Heidelberg, 2012, pp. 172–179.
15. C. Yan, Y. Zhang, F. Dai, and L. Li, "Efficient parallel framework for hevc deblocking filter on many-core platform," in *Data Compression Conference (DCC), 2013*, March 2013, pp. 530–530.
16. "Openmp application program interface, version 3.1," *OpenMP Architecture Review Board. http://www.openmp.org*, 2011.