Conference
4–8 July, 2016.

Tile partition analysis for a parallel HEVC encoder

Pablo Piñol[1], Otoniel López-Granado[1], Héctor Migallón[1], Vicente Galiano[1] and Manuel P. Malumbres[1]

[1] *Department of Physics and Computer Architecture, Miguel Hernández University*

emails: pablop@umh.es, otoniel@umh.es, hmigallon@umh.es, vgaliano@umh.es, mels@umh.es

Abstract

The new video coding standard HEVC introduces a new concept named "tiles". Tiles are rectangular regions of a video frame which can be encoded in an independent way. In order to reduce the time needed to encode a video sequence with HEVC, we have used a parallel approach based on tiles, and have evaluated the benefits and drawbacks of this approach. In our tests we obtain speed ups of up to 9.3x using 10 processes with a low R/D loss.

Key words: tiles, HEVC, video coding, Parallel algorithms, multicore, performance

1 Introduction

The Joint Collaborative Team on Video Coding (JCT-VC) has developed a new video coding standard, named High Efficiency Video Coding (HEVC) [1]. JCT-VC is formed by members of the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG). The new standard achieves nearly a 50% of bit rate saving, when compared to the previous video coding standard H.264/AVC (Advanced Video Coding) [2]. The increase in the efficiency is bound to an increase in the computational complexity. To address the increase in complexity we make use of parallelization techniques, and thus, take advantage of the available parallel computing architectures.

HEVC includes some new features which allow high-level parallel computing (at a picture or subpicture level), like Wavefront Parallel Processing (WPP) and tiles, and some new features which allow low-level parallel computing (inside the encoding process), like Local Parallel Method [3] which allows parallel motion estimation.

Tiles are rectangular regions of a video frame which can be independently encoded. In this work we analyze the parallel behavior when tiles are used and how tile partition layout affects both the speed-up and the Rate/Distortion (R/D) performance.

 ISBN: 978-84-608-6082-2

Some works like [4] are focused on parallelizing the decoding side of HEVC, since they are looking for fast decoding of pre-encoded multimedia content, like digital cinema and video on demand. However, we think that more effort should be provided to the encoder side, where the highest computational complexity is found. Parallelizing the video encoding part can be useful for applications like video recording and live event streaming.

There are works that examine and propose low-level parallel techniques for encoding video sequences with HEVC, like the parallelization of the motion estimation module [5] and the parallelization of the intra prediction module [6]. Our work is based in high-level parallel techniques, by using tiles to take advantage of shared memory architectures.

In [7] authors compare slices and tiles encoding performance in HEVC. They show their results in terms of percentage of bit rate increase/decrease. In our study we evaluate tiles performance but we will focus on both complexity reduction related with the encoding process and R/D performance. Even more, we evaluate the impact of the tile partitioning.

The rest of the paper is organized as follows. In Section 2 we will present the main aspects of tiles in HEVC. In Section 3 the results of our tests will be presented and analyzed. At last, several conclusions will be drawn in Section 4.

2 Tiles partitioning

The division of a video frame into tiles is a new technique, included in HEVC, which did not exist in previous video coding standards. Tiles are rectangular regions of a video frame which can be independently encoded (and subsequently decoded), with the use of some common data for the whole frame. This independence allows the parallelization of the encoding and decoding processes at a subpicture level. Tiles contain an integer number of Coding Tree Units (CTUs). Each rectangular region results from the division of a frame into several columns and rows. The width of each column (in CTU units) and also the height of each row can be set individually. In the example shown in Figure 1, a full-HD (1920x1080) frame is divided into 10 tiles using a partitioning scheme of 5 columns with a width of 6 CTUs and 2 rows with a height of 8 and 9 CTUs, respectively.

The independence of tiles allows the parallelization of the encoding process but it has a main drawback: coding efficiency, regarding R/D performance, is reduced. This happens because tiles cannot use information from CTUs belonging to other nearby tiles to make any kind of prediction, and thus, the existing redundancy between nearby CTUs which belong to different tiles cannot be exploited.

In this work we have implemented a tile-level parallelization of the HEVC video encoder for shared memory platforms. The encoding of each tile is assigned to a different core. We have evaluated the performance of the parallel version of the video encoder for 2, 4, 6, 8, 9, and 10 processes and compared the obtained results with those provided by the sequential version, both regarding R/D performance and computing performance. As stated before,

 ISBN: 978-84-608-6082-2

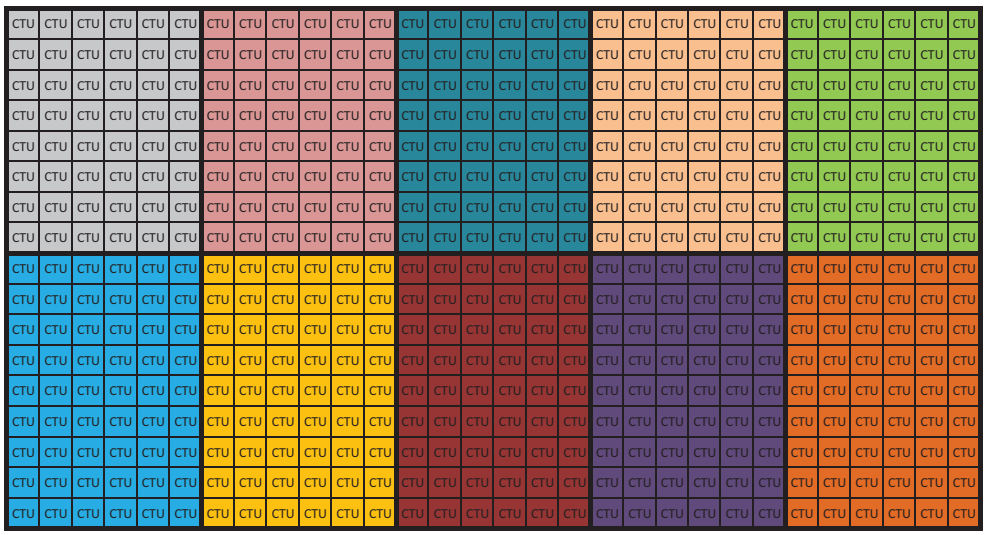


Figure 1: Division of a full-HD frame (1920x1080 pixels) into 10 tiles (5 columns with a width of 6 CTUs; 2 rows with a height of 8 and 9 CTUs each)

we map the tiles per frame onto the same number of processes. For a certain number of tiles per frame, we can find different frame partitions. For example, if we want to divide the frame into 9 tiles we can choose three main different distributions: 9x1 (9 columns by 1 row), 1x9 (1 column by 9 rows), and 3x3 (3 columns by 3 rows). Each one of these distributions can be further designed in different ways if we change the width and height of each one of the columns and rows. In order to get the maximum parallel computing efficiency we will use the layouts that provide the most balanced load distribution, i.e., producing tiles with equal or similar number of CTUs. A balanced load distribution does not always guarantee a balanced work distribution because the resources needed to encode a single CTU may vary, but a completely unbalanced load distribution will likely bring a low parallel computing efficiency. As a measure of the load distribution balance we have calculated the maximum theoretical parallel efficiency, considering the same computational complexity for each CTU. A value of 100% denotes that all the processes will encode the same number of CTUs (and this number is exactly the average value). In Table 1, we have enumerated the different tile partitions that we will use in our tests for different number of cores and two video formats.

The optimum theoretical parallel efficiency would be provided by the "balance %" column in Table 1. If we divide, for example, a frame with 2560x1600 pixels into 10 tiles, and choose the 10x1 layout, then we will have 10 tiles of 100 CTUs each, which means 100% of load balance (all tiles have the same number of CTUs). If we, instead, select

 ISBN: 978-84-608-6082-2

Table 1: Layouts and percentage of load balance

(a) 2560x1600 (40x25 CTUs)

#Proc	Layout	AvgCTU	MaxCTU	Bal %
1P	1x1	1000	1000	100%
2P	1x2	500	520	96%
	2x1	500	500	100%
4P	1x4	250	280	89%
	2x2	250	260	96%
	4x1	250	250	100%
6P	1x6	166.7	200	83%
	2x3	166.7	180	93%
	3x2	166.7	182	92%
	6x1	166.7	175	95%
8P	1x8	125	160	78%
	2x4	125	140	89%
	4x2	125	130	96%
	8x1	125	125	100%
9P	1x9	111.1	120	93%
	3x3	111.1	126	88%
	9x1	111.1	125	89%
10P	1x10	100	120	83%
	2x5	100	100	100%
	5x2	100	104	96%
	10x1	100	100	100%

(b) 1920x1080 (30x17 CTUs)

#Proc	Layout	AvgCTU	MaxCTU	Bal %
1P	1x1	510	510	100%
2P	1x2	255	270	94%
	2x1	255	255	100%
4P	1x4	127.5	150	85%
	2x2	127.5	135	94%
	4x1	127.5	136	94%
6P	1x6	85	90	94%
	2x3	85	90	94%
	3x2	85	90	94%
	6x1	85	85	100%
8P	1x8	63.8	90	71%
	2x4	63.8	75	85%
	4x2	63.8	72	89%
	8x1	63.8	68	94%
9P	1x9	56.7	60	94%
	3x3	56.7	60	94%
	9x1	56.7	68	83%
10P	1x10	51	60	85%
	2x5	51	60	85%
	5x2	51	54	94%
	10x1	51	51	100%

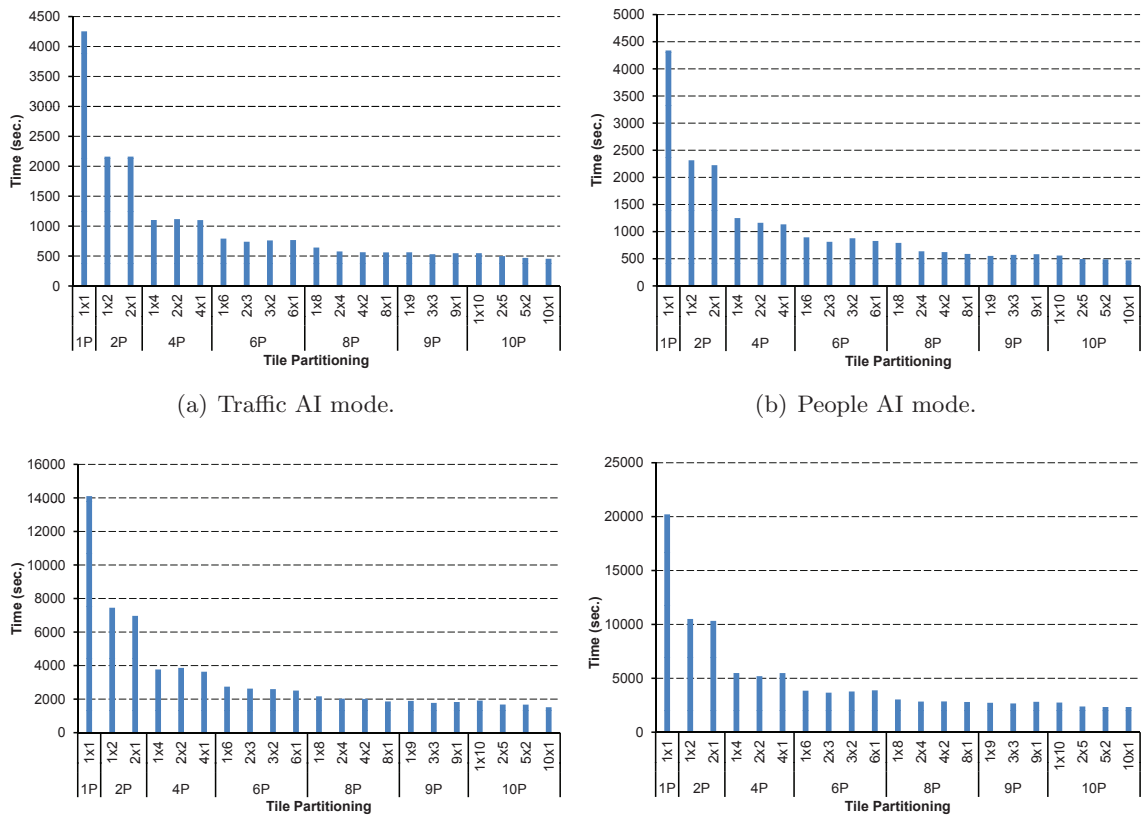
the 1x10 layout, then we will have 5 tiles with 80 CTUs each, and 5 tiles with 120 CTUs each. The most probable scenario is that the 5 processes managing the "small" tiles remain idle waiting for the processes in charge of the "big" tiles. In this case, a maximum load balance index of 83% would be achieved. So, for a specific number of processes, the selected layout may affect the parallel efficiency. Note also that a single layout can provide different load balance percentages depending on the resolution of the video sequence. For example, the 4x1 layout obtains 100% and 94% of load balance for 2560x1600 and 1920x1080 video resolutions, respectively.

In the next section we will present the results of encoding the selected video sequences by using all the layouts presented in Table 1.

3 Numerical experiments

The proposed parallel algorithm has been tested on a shared memory platform consisting of two Intel XEON X5660 hexacores at up to 2.8 GHz and 12MB cache per processor, and 48

 ISBN: 978-84-608-6082-2



(a) Traffic AI mode.

(b) People AI mode.

(c) Traffic LB mode.

(d) People LB mode.

Figure 2: Encoding time evolution for all video sequences with different number of processes and tile partitioning for QP=37

GB of RAM. The operating system used is CentOS Linux 5.6 for x86 64 bit. The parallel environment has been managed using OpenMP [8]. The compiler used is *g++* compiler *v*.4.1.2. The reference encoder software used is HM 16.3 [9].

The testing video sequences used in our experiments are Traffic (2560x1600), People on Street (2560x1600), Tennis (1920x1080), and Park Scene (1920x1080), and we present results using Low-delay B (LB) and All Intra (AI) coding modes, encoding 150 frames for Traffic and People sequences and 240 frames for ParkScene and Tennis sequences at different Quantization Parameters (QPs) (22, 27, 32, 37).

In Figure 2, we present the encoding time evolution for Traffic and People sequences in both AI and LB modes as a function of the number of processes with the proposed tile partitioning. As can be seen, the encoding time can be reduced up to 9.3 times when we

 ISBN: 978-84-608-6082-2

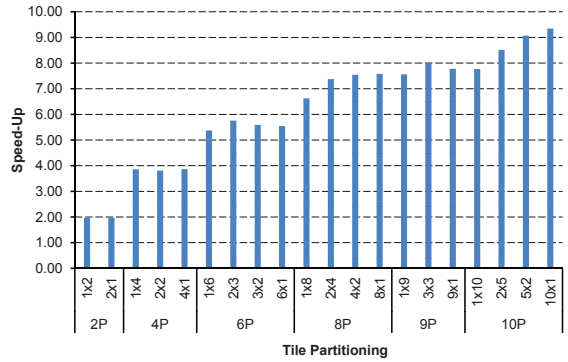
use 10 processes. As can be seen in Figure 3, the tile-level parallelization algorithm obtains a good parallel performance and and also nice scalability results. Looking at Figure 3, for 10 processes, there are differences in the parallel performance when we use different tile partitioning layouts. Specifically, in AI coding mode we can find differences of up to 1.58x from a tile partitioning scheme of 1x10 respect to the tile partitioning scheme of 10x1. In general, tile partitioning layouts based on columns of CTUs or square tiles obtain better parallel performance. As it would be expected, this effect will depend on the video resolution. Following with the previous example, for a video resolution of 2560x1600, and a CTU size of 64x64, the number of CTUs in a frame are 40x25. If we divide the frame with the 1x10 layout we have 5 processes with 40x2 CTUs and 5 processes with 40x3 CTUs. On the other side, if we divide the frame with the 10x1 layout we have 10 processes with 4x25 CTUs. In the first case (1x10), 5 processes have to perform a 50% more work than the other 5 proccesses. Usually the more balanced the computational load is, the better parallel performance is achieved, except for some sequences where this is not accomplished. In those exceptions, even when each process has the same number of CTUs, the computational complexity inherent to each CTU differs, producing that some processes finish before the others.

In Figure 4 we show the average parallel efficiency for all QP values of the different tested images encoded in both LB and AI modes. As can be seen good efficiencies are obtained for both LB and AI encoding modes, being the efficiency 87% on average. However, if we focus on square tile partitioning layouts, the average parallel efficiency rises till 91%.

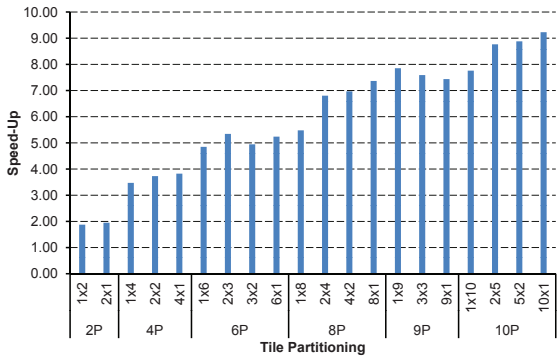
Regarding R/D behavior, in Figure 5 we present the % of BD-rate evolution for 4K video sequences as a function of the number of processes and the tile partitioning scheme. As can be seen, the % of BD-rate increases as the number of processes does. This is an expected behavior because tiles are independent structures and therefore, the arithmetic encoder works in an independent way on each tile and no information of previously encoded tiles is available. Furthermore, square tile partitioning performs slightly better in both LB and AI modes, because more information of neighbouring CTUs is available for inter and intra prediction.

Looking at the results, we can asses that parallelization of HEVC in tiles is worth the effort, because it drastically reduces the encoding time (up to 9.3x for 10 processes) with a low R/D increment, specially if square partitioning schemes are used (0.75% BD-rate for AI mode and 1.2% for LB mode on average). The maximum increment due to tile partitioning scheme is 5% BD-rate increment for Tennis sequence in LB mode using 10 processes. So, the main aspect that will affect the parallel performance is the computational load and this is why we should divide tiles in such a manner that all processes have the same number of CTUs. In the case of the video sequences analyzed in this work, tiles using square partitioning or tiles with more columns of CTUs than rows, have a better computational load distribution.

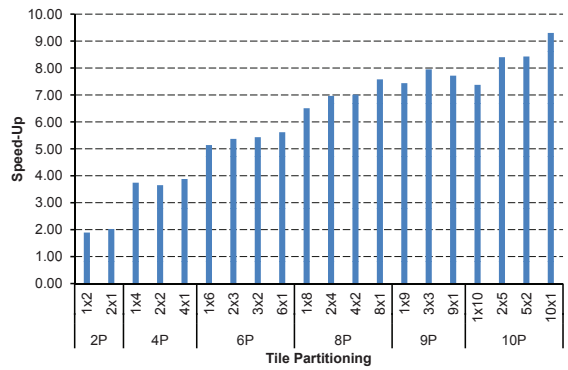
 ISBN: 978-84-608-6082-2



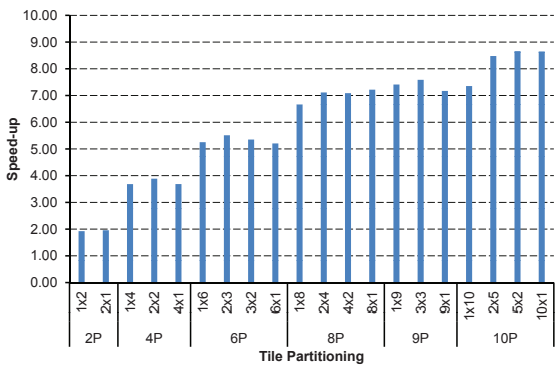
(a) Traffic AI mode.



(b) People AI mode.



(c) Traffic LB mode.



(d) People LB mode.

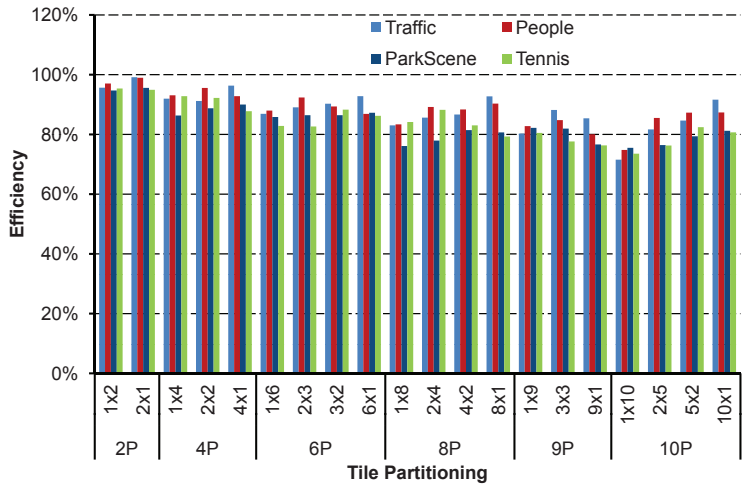
Figure 3: Speed-Up evolution for all video sequences with different number of processes and tile partitioning for QP=37

4 Conclusions

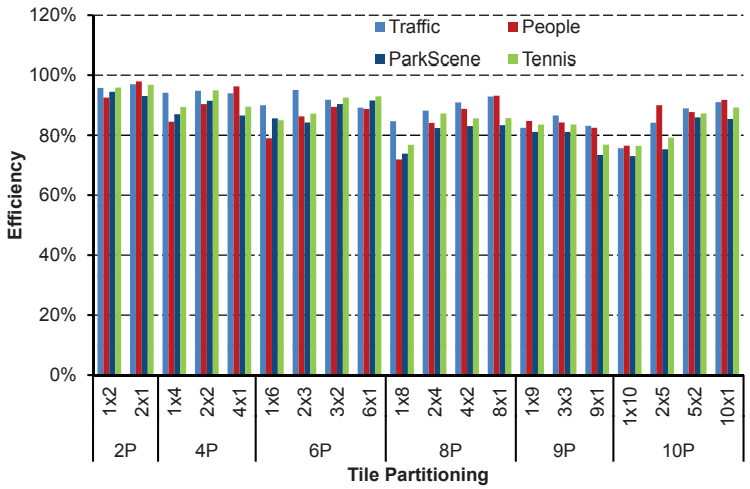
In this paper we present an study of the tile-level parallelization approach of HEVC encoder when different tile partitioning layouts are used. Results show that both square tile and column tile partitioning layouts obtain the best speed-ups (up to 9.3x for 10 processes) in the tested video sequences. Although in some experiments column-based tile partitioning obtain better parallel efficiency, on average, square tile partitioning layouts present a better behavior in both speed-up and R/D. Besides, the increment in BD-rate percentage is low in all cases, specially when square tile partitioning is applied, because more information of neighboring CTUs is available for the inter and intra prediction processes.

Finally, we should take into account the video sequence resolution in order to perform

 ISBN: 978-84-608-6082-2



(a) LB mode.

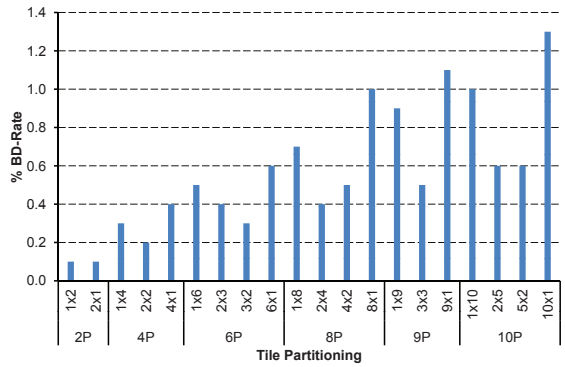


(b) AI mode.

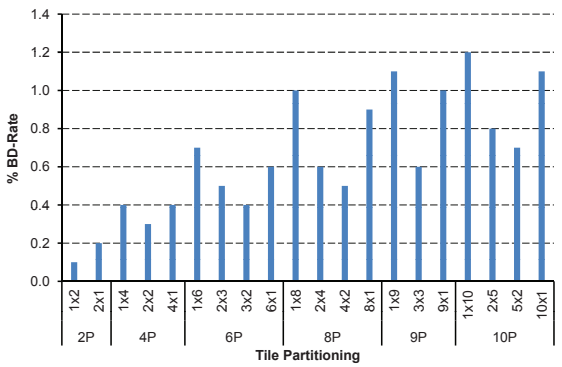
Figure 4: Average parallel efficiency for all tested video sequences with different number of processes and tile partitioning for all QPs

the tile partitioning in such a way that the number of CTUs on each tile will be nearly the same and thus the computational load will be more balanced, obtaining better speed-ups.

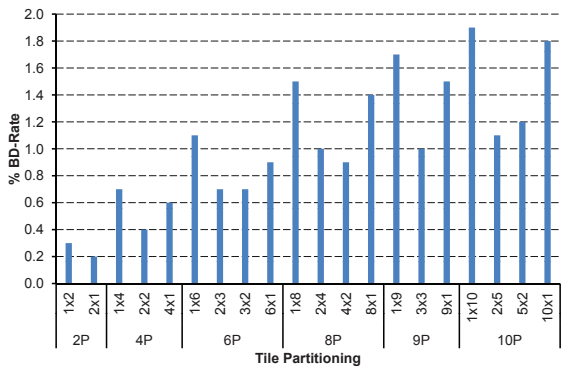
 ISBN: 978-84-608-6082-2



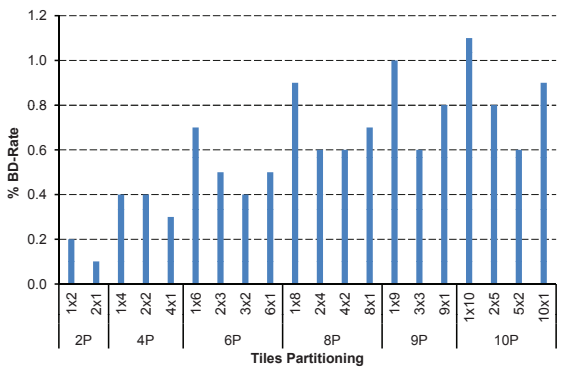
(a) Traffic AI mode.



(b) People AI mode.



(c) Traffic LB mode.



(d) People LB mode.

Figure 5: Average % BD-Rate evolution for all video sequences with different number of processes and tile partitioning for all QPs

Acknowledgments

This research was supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2015-66972-C5-4-R co-financed by FEDER funds.

References

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 10," Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), Tech. Rep. JCTVC-L1003, January 2013.

 ISBN: 978-84-608-6082-2

[2] 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*, 2012.

[3] M. Zhou, "AHG10: Configurable and CU-group level parallel merge/skip," Joint Collaborative Team on Video Coding-H0082, Tech. Rep., 2012.

[4] 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.

[5] Q. Yu, L. Zhao, and S. Ma, "Parallel AMVP candidate list construction for HEVC," in *VCIP'12*, 2012, pp. 1–6.

[6] 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.

[7] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 6, pp. 969–977, Dec 2013.

[8] "Openmp application program interface, version 3.1," *OpenMP Architecture Review Board. http://www.openmp.org*, 2011.

[9] HEVC Reference Software, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.3/.

 ISBN: 978-84-608-6082-2