Journal of Real-Time Image Processing (2020) 17:787–797
https://doi.org/10.1007/s11554-018-0819-3

ORIGINAL RESEARCH PAPER

Parallelism exploration for 3D high-efficiency video coding depth modeling mode one

Gustavo Sanchez[1,2,3] · **Luciano Agostini**[2,4] · **Leonel Sousa**[2] · **César Marcon**[1]

Received: 24 January 2018 / Accepted: 23 August 2018 / Published online: 12 September 2018

Abstract
This article presents a parallelism exploration over the depth modeling mode 1 (DMM-1) encoding algorithm of the 3D high-efficiency video coding (3D-HEVC) standard and applied the proposed solutions in a multicore central processing unit (CPU) and two graphics processor unities (GPU). The article evaluates efficient parallel algorithms for DMM-1, which also take advantage of simplifications proposed in our previous works. We demonstrate that DMM-1 can obtain a scalable speedup when running in systems with several available cores even when simplifications are being applied. Experimental results for 1920×1088 resolution videos show that the proposed parallel algorithms achieved up to 2 frames per second (fps) in a four-cores (with eight threads) CPU and more than 30 fps in two different GPUs. Therefore, the speedup attained with GPU enables real-time 3D-video encoding applying the proposed parallelism strategies together with the DMM-1 proposed simplifications.

Keywords Depth modeling modes · 3D-HEVC · Multicore · Parallel algorithm · GPU

1 Introduction

High-efficiency video coding (HEVC) extensions [1] have been proposed to improve the encoding efficiency, such as 3D video coding with 3D-HEVC [2], Multiview Video Coding using MV-HEVC, screen content using HEVC-SCC, and Range Extension HEVC using REext-HEVC [3]. However, the encoding efficiency has as a counterpart, the rise of the computational effort, demanding new methods and techniques to speed up the encoding process.

The use of the multiview video plus depth (MVD) data format [4] is one of the main reasons for the raising of the computational effort of 3D-HEVC. MVD associates a depth map to each texture frame, encoding and packing both into a single bitstream. It allows synthesizing a dense set of intermediary high-quality virtual views at the decoder using techniques such as depth image-based rendering (DIBR) [5].

Figure 1a, b, from the Kendo video sequence [6], show a texture view and the associated depth map. The texture frame represents the color of the image while the depth map provides the distance between the camera and the objects of the scene. While texture frames typically exhibit smooth transitions, depth maps are composed of homogeneous regions and sharp edges. Homogeneous regions correspond to the background of the scene and the body of the objects, while sharp edges occur on the border of the objects.

The coding of depth maps has inherited algorithms designed for HEVC texture coding, although those algorithms are not specialized to explore the depth maps characteristics. 3D-HEVC overcomes this problem by inserting new encoding tools used as alternatives to the HEVC texture tools. Depth modeling modes 1 and 4 (DMM-1 and DMM-4) [7], segment-wise direct component coding (SDC) [8] and

✉ Gustavo Sanchez
gustavo.sanchez@acad.pucrs.br

Luciano Agostini
agostini@inf.ufpel.edu.br

Leonel Sousa
las@inesc-id.pt

César Marcon
cesar.marcon@pucrs.br

[1] Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil

[2] INESC-ID, IST, Universidade de Lisboa, Lisbon, Portugal

[3] IF Farroupilha, Alegrete, Brazil

[4] Video Technology Research Group (ViTech), Federal University of Pelotas (UFPel), Pelotas, Brazil

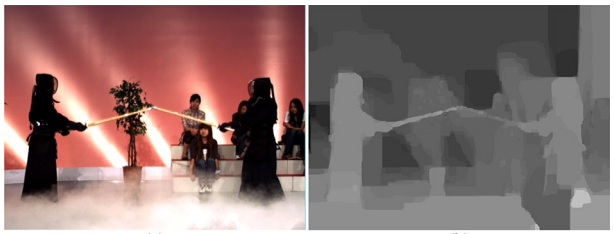


Fig. 1 Kendo video sequence **a** texture view and **b** depth map [6]

depth intra skip (DIS) [9] are examples of tools for intra-frame prediction. By combining the usage of these tools, 3D-HEVC reduces the bit rate required to encode homogeneous regions and sharp edges, defining new edge-aware ways of prediction.

In a previous work [10], we have evaluated the computational time required by the encoding steps of the intra-frame prediction. Figure 2 displays results of this study, highlighting the impact of SDC and DMM-1 on the encoding time, according to the quantization parameter (QP).

Limited parallelism can be exploited to speed up the computation of SDC since there are data dependencies with the entropy coding (EC). Thus, SDC is not a good target for parallelism exploration, due to the sequential and irregular nature of the EC [11]. However, the DMM-1 encoding tool can be parallelized up to a certain level by exploiting the application of similar operations to encode a frame at the right granularity level. Although techniques have been already proposed to speed up the execution of DMM-1 [12–17], they impose encoding efficacy losses. Therefore, an important challenge is to investigate ways to improve the performance of the most time-consuming tools for 3D-HEVC depth maps coding.

This article exposes the parallelism of DMM-1 for developing efficient algorithms suited for multicore processors and Graphics Processor Units (GPUs). The algorithms have been programmed with OpenMP [18] and CUDA [19]. Experimental results obtained with Intel Core i7 6700 K (Skylake) four-core processor and two GPUs (NVidia GTX TITAN X and NVidia TITAN Xp) show the practical interest of the proposed algorithms. To the best of our knowledge, this is a pioneer work on exploring parallelism for efficiently performing DMM-1 encoding.

The remaining of this article is organized as follows. Section 2 presents the background of the 3D-HEVC depth map intra-frame prediction, along with the related state-of-the-art on DMM-1 encoding algorithms. Section 3 proposes parallel algorithms to speed up the computation of DMM-1. Section 4 provides an experimental evaluation of the parallel algorithms on current multicore processors and GPUS. Section 5 concludes the article.

2 3D-HEVC depth maps intra-frame prediction

The 3D-HEVC depth maps intra-frame prediction adopted in this article follows the implementation of the 3D-HEVC Test Model 16.2 (3D-HTM) [20]. The encoding process of 3D-HEVC was inherited from HEVC texture coding, being the frame divided into coding tree units (CTUs) [21] that are individually encoded. Improved coding efficiency is achieved by splitting these CTUs into four coding units (CUs), which in turn can be recursively subdivided into smaller CUs. The encoding CU can also be partitioned into Prediction Units (PUs) when a block is encoded by adopting intra- or inter-frame prediction.

Figure 3 depicts the dataflow model for depth map intra-frame prediction, with the encoding tools applied to any block size. The best combination of block sizes and encoding tools is selected at the cost of high computational effort for minimizing the rate-distortion (RD-cost), which

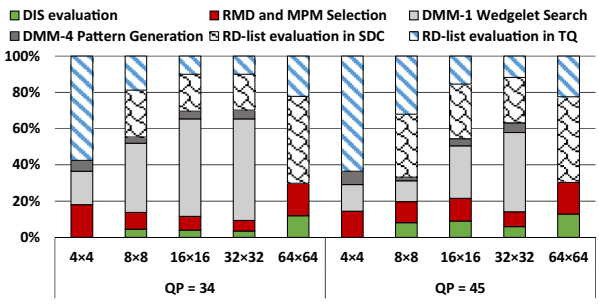


Fig. 2 Profiling of the 3D-HEVC depth maps intra-frame prediction [10]

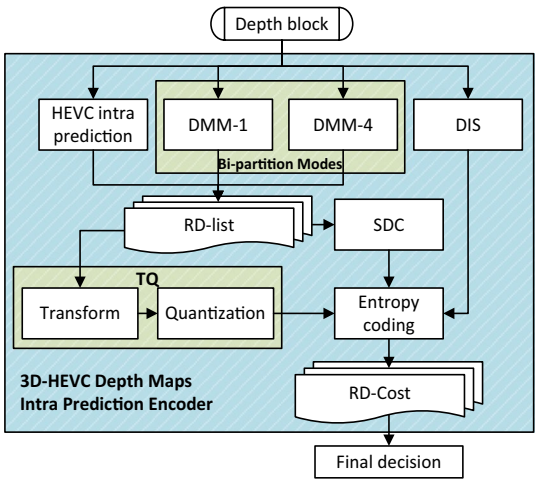


Fig. 3 3D-HEVC depth maps intra-frame prediction dataflow model [10]

is a function that ponders the required bandwidth and the quality of the encoded block. The encoding block should be evaluated using HEVC intra-frame prediction, DMM-1 or DMM-4 in transform–quantization (TQ) and SDC flows, and DIS mode. These evaluations converge to the EC, the RD-cost is estimated to identify the best solution to be inserted in the final encoded stream.

Since most of the depth map information corresponds to smooth areas, the DIS encoding mode focuses on achieving considerable bitrate reduction in these areas by not packing the residues into the bitstream. DIS allows four prediction modes, based on the neighbor samples without using TQ and SDC flows [9], being the results forwarded for EC.

The remaining encoding modes (i.e., DMM-1, DMM-4, and HEVC intra prediction) employ the TQ or SDC flows. The encoder performs local evaluation and selects a set of modes to be inserted in the rate-distortion list (RD-list). Subsequently, the modes inserted in this list are evaluated based on their RD-cost in both TQ and SDC flows.

The HEVC intra-frame prediction [22] was inherited from the texture coding, without any modification. It contains 35 modes (i.e., planar, DC and 33 directional modes), whose directions can be seen in Fig. 4. Instead of evaluating all these encoding modes based on the RD-cost, 3D-HTM evaluates these modes locally using Rough Mode Decision (RMD) [23].

RMD applies the sum of absolute transform differences (SATD) for comparing the block predicted with the original encoding block samples. Eight modes for 4×4 and 8×8, and three modes for 16×16, 32×32 and 64×64, with the lowest SATD, are selected to insert into the RD-list. Besides, the most probable modes (MPM) heuristic is applied after the RMD for selecting modes that were used to encode neighbor blocks (the left and above neighbors). Subsequently, MPM inserts them into the RD-list if they were not inserted into the RD-list in the RMD analysis.

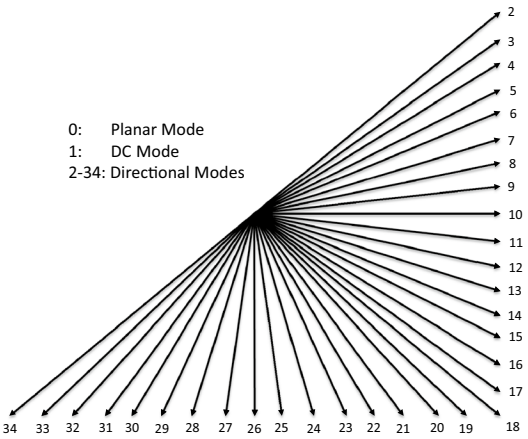


Fig. 4 HEVC intra-frame prediction directions

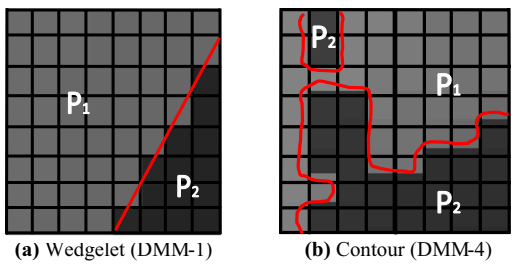
DMM-1 and DMM-4 are edge-aware encoding tools. They were developed for obtaining high quality when encoding edges regions, since low-quality encoding may lead to a wrog interpretation between background and foreground pixels when synthesizing new virtual views [24].

Figure 5a shows DMM-1 divides the encoding block using a wedgelet, which is a straight line that splits the encoding block into two regions. While DMM-1 assumes only the predefined wedgelets patterns defined by the 3D-HEVC standard, DMM-4 breaks the encoding block in regions using contours; each region can assume arbitrary patterns, consisting of several parts, as shown in Fig. 5b. Both DMMs are applied on blocks sizes ranging from 4×4 to 32×32.

DMM-4 generates the segmentation pattern using texture data. Since the texture data is available at both the decoder and the encoder, the decoder can perform the same algorithm to generate the pattern than the encoder, thus reducing the bitstream size. After executing DIS, HEVC intra-frame, DMM-1 and DMM-4 predictions, the depth maps encoding process finishes applying TQ, SDC, and EC. The TQ flow was inherited from the texture coding, without any modification [25]. The SDC tool has been designed as an alternative to TQ, to obtain higher efficiency by exploring depth maps properties. It obtains a higher efficiency when the HEVC intra prediction is used in a homogeneous region or DMMs are used to segment a block well divided into two homogeneous regions. Finally, EC uses context-based adaptive binary arithmetic coding (CABAC) [26], also inherited from texture coding. Since the DMM-1 is the focus of this work, the next subsection describes it more in detail.

2.1 Depth modeling mode One (DMM-1)

Figure 6 presents the encoding flowchart of the DMM-1 algorithm, composed of the *Main*, the *Refinement* and the *Residue* stages. The Main stage evaluates an initial predefined wedgelet set. It searches for the wedgelet that produces the lowest distortion at the synthesized views, in comparison to the synthesized view generated by the original encoding samples. For finding the lowest distortion, the encoding



(a) Wedgelet (DMM-1) **(b)** Contour (DMM-4)

Fig. 5 Example of a wedgelet and a contour segmentation

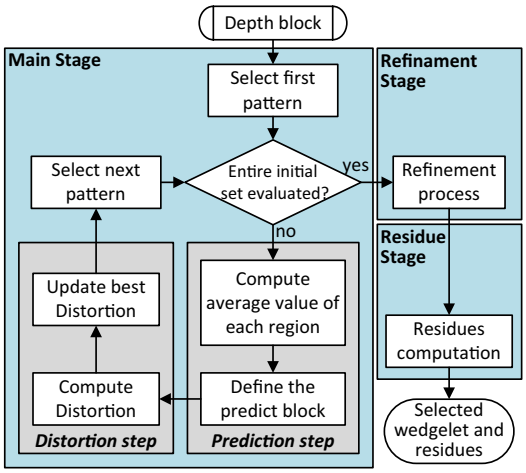


Fig. 6 DMM-1 encoding algorithm [27]

Table 1 Number of evaluated wedgelets in DMM-1

Block size	Total of possible wedgelets	Evaluated wedgelets in the main stage
4×4	86	58
8×8	802	314
≥ 16×16	510	384

block is mapped into the binary pattern defined by each wedgelet and the average value of each region is computed according to this mapping.

The Prediction step predicts the depth block using the average value of each region. The synthesized view distortion change (SVDC) [28] computes the distortion of a synthesized block compared to the synthesizing texture views using the original depth block values in the Distortion step. The wedgelet that leads to the lowest distortion is selected.

The Refinement stage uses SVDC to evaluate up to eight wedgelets (with slight differences from the best wedgelet selected in the Main Stage). Subsequently, the wedgelet with the lowest distortion is chosen to encode the current depth map block. This wedgelet is inserted in the RD-list with the residues of this block (that is generated in the Residue stage) to be used in the TQ and SDC evaluations.

Table 1 shows the number of wedgelets required to compute the DMM-1. Notice that there is a vast number of wedgelets evaluated on the Main stage and a high number of total possible wedgelets that needs to be stored and used during the DMM-1 computation.

The encoding of a DMM-1 block is independent of the encoding of other blocks. Therefore, parallelism can be explored at two different granularities to accelerate DMM-1 execution in a parallel system: (1) block granularity—each core encodes a given block, and (2) pattern granularity—the effort spent on encoding several wedgelets for a block is divided between the available cores.

2.2 Speeding up DMM-1: related work

Several algorithms/schemes have been proposed to simplify the DMM-1 computation, by skipping the entire DMM-1 computation or reducing the DMM-1 wedgelet list evaluation.

The works [12–14] proposed to skip the entire DMM-1 computation based on whether the coding block is smooth or represents an edge. Gu et al. [12] verify the block variance and the best-ranked mode at HEVC intra prediction to establish if the blocks tend to be smooth, the case when DMMs are rarely selected. It decides to skip the DMMs evaluation when the best-ranked mode in RD-list is the planar, or the variance of the encoding block is small. Zhang et al. [13] recommend skipping the DMMs evaluation if the best mode selected by HEVC intra-frame prediction is the planar or the DC mode. We have proposed the simplified edge detector (SED) for classifying the encoding block into homogeneous or edge [14]. SED computes the maximum difference of the four corner samples and compares this value with a defined threshold—when a block is classified as homogeneous, the DMMs are skipped. Among these approaches, only SED [14] does not depend on other encoding modules, such as HEVC intra prediction. Consequently, SED can be easily adopted to speed up the encoding process by exploiting parallelism, given its independence to the other encoding modules.

Several works ([13, 15–17]) simplify the DMM-1 algorithm reducing the wedgelet evaluation list. Zhang et al. [13] shrink the DMM-1 search only to the wedgelets that follow six HEVC intra prediction directions, without any pre-processing. It statically defines a sub-set of wedgelets to be always evaluated. Fu et al. [15] decrease DMM-1 wedgelet search space by organizing the encoding blocks into orientation classes. This classification is performed according to the variance on sub-regions of the encoding block, reducing the wedgelet pattern search only to the ones with similar orientations. In [16], we have designed the Gradient-based Mode One Filter (GMOF), which filters the encoding block borders seeking for the most promising wedgelets, thus skipping wedgelets evaluation that does not start on a block edge. Sanchez et al. [17] proposed the use of neighborhood blocks information to speed up the current DMM-1 encoding through the DMM-1 fast pattern selector (DFPS). This last method imposes data dependencies with neighbor blocks preventing block granularity acceleration. Except [17], which has these data dependencies, all other solutions for reducing the wedgelet assessment can be

easily integrated into data parallel algorithms to speed up the DMM-1 computation.

Table 2 summarizes the related work on simplifying the DMM computation. As a representative case of study, this work uses the SED [14] and GMOF [16] to assess the impact of skipping the DMM-1 computation and reducing the DMM-1 wedgelet evaluation in parallel processing systems.

Although works referred in this section may reduce the DMM-1 encoding time significantly, they also impose losses on the encoding quality. This work investigates the speedup of DMM-1 computation by balancing the encoding effort through multiple processing cores in a process that does not imply encoding losses. Besides, we show that by simplifying the DMM-1 procedure, we can take also an advantage of the proposed parallel approach aiming to reduce the encoding time significantly.

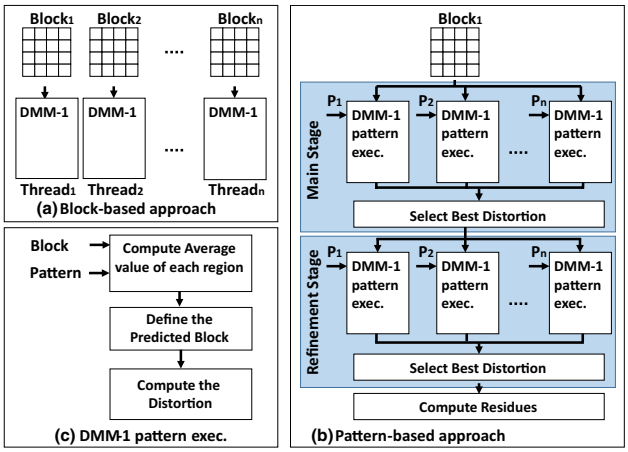


Fig. 7 Parallelism exploration: **a** block-based, **b** pattern-based, and **c** DMM-1 pattern execution

3 Parallel algorithms for DMM-1 encoding

This work targets the following parallel architectures: (1) a symmetric multiprocessor, with few but powerful multi-cores and memory coherence supported by hardware, as the current multi-core processors; and (2) a massively parallel GPU, with thousands of simple cores operating under the single instruction multiple data (SIMD) paradigm, which enables to explore data parallelism in a stream computing approach. In particular, as an example, the experiments of this work target CPUs, typically with 8 cores (16 threads), and GPUs with more than 3000 stream processors (simple cores). Although the experiments have been performed on a 4-core CPU, the proposed methodology is scalable for systems with a larger number of cores.

This article investigates two parallel approaches to handle the DMM-1 computation, which allow data parallelism exploration at block-based and pattern-based granularities. Figure 7a illustrates how the block-based approach explores parallelism. Since DMM-1 evaluates a block independent of its neighborhood, each block can be assigned to a given processing unit for parallel encoding. In this approach, we can consider that a thread is responsible for the entire DMM-1 encoding of a depth bloc, applying the Main, Refinement and Residue stages (see Fig. 6).

Since DMM-1 evaluates several wedgelets patterns, the proposed pattern-based approach assigns the evaluation of each pattern to a thread, as shown in Fig. 7b. The wedgelets evaluated in the Main stage are distributed by the threads balancing the computational effort between them. Therefore, the loop presented in Fig. 6 is eliminated, and each thread evaluates the encoding blocks for a given pattern using the flow presented in Fig. 7c. After computing the distortion of all wedgelets, the threads are synchronized and the distortion results are compared for selecting the best one. In the Refinement stage, the wedgelet patterns are also assigned to multiple threads for similar processing.

The SED [14] and GMOF [16] heuristics were also applied to the block-based parallel approach to evaluate the gains the DMM-1 simplification could bring into a parallel platform. Besides, we are interested in investigating the characteristics of the simplification algorithms that allow obtaining the best benefits from parallel architectures with different characteristics. The evaluation of these two heuristics was initially done through sequential processing using 3D-HTM. The SED algorithm skips the DMMs evaluation

Table 2 Related work for simplifying the DMM-1 algorithm

Simplifying type	Work	Technique
Skip entire DMM-1 calculation	Gu et al. [12]	Computes the variance and verifies the best-ranked modes in RD-list
	Zhang et al. [13]	Verifies the best-ranked modes in RD-list
	Sanchez et al. [14]	Computes the maximum difference of four corners samples
Reducing wedgelet list evaluation	Fu et al. [15]	Classifies the encoding block for finding the best wedgelets orientations
	Zhang et al. [13]	Reduces the wedgelet list to wedgelets that follow the HEVC intra prediction directions
	Sanchez et al. [16]	Filters the borders and select the most promising wedgelets to be evaluated
	Sanchez et al. [17]	Uses neighbor encoded blocks information to accelerate the DMM-1 encoding

for homogeneous blocks since they tend not to be selected in this scenario. This pre-processing reduces the DMM-1 computational effort in 93.4% (speedup of 15.1) with a degradation of 0.94% in the BD-rate. The GMOF directly simplifies the DMM-1 evaluation by applying a gradient filter in the border of the encoding block to identify the most promising wedgelets for block segmentation. Several wedgelets evaluations are skipped with this algorithm (reducing the DMM-1 computational effort in 66.9%, corresponding to a speedup of 3.0) with minor degradation in the encoding efficacy (0.33% in BD-rate, on average).

The parallelization strategies presented in this article do not insert any additional BD-rate degradation when using SED and GMOF algorithms. Then, the BD-rate obtained using the parallel coding is exactly the same result reached with a single thread execution.

4 Experimental results

Considering 3D-HTM was developed to evaluate the efficiency of the encoder tools, not being suitable for evaluating computational performance, we programmed in C++ a single-thread efficient implementation of DMM-1. Besides this base original algorithm, two other versions were implemented, one using the SED heuristic and another one applying the GMOF heuristic. OpenMP 3.1 and CUDA 7.0 were used to program the proposed algorithms for CPUs and GPUs, respectively.

In our experiments, the developed programs encoded the central view of the eight videos available at common test conditions (CTC) for 3D experiments [29]. Inputs to the developed program are the raw data of depth maps and the reconstructed texture video obtained with 3D-HTM. The programs, based on the proposed parallel algorithms, provide at the output residual data and the selected DMM-1 pattern.

We performed the experiments on an Intel Core i7 6700K (Skylake) with 4 cores (8 threads) running at 3.5 GHz and with 32 GB DDR3 memory. Two GPUs were used in this evaluation: (1) NVidia GTX Titan X (GM200—Maxwell) with 3072 CUDA cores running at 1.08 GHz (called Titan X in the rest of this article) and; (2) NVidia Titan Xp (GP102—Pascal) with 3840 CUDA cores running at 1.48 GHz (called Titan Xp in the rest of this article). Although we performed the evaluations using a four-core CPU and using two NVIDIA GPUs, the presented experiment aims to show that encoding blocks can be assigned to multiple cores to speed up the encoding process. Besides, the proposed methods and algorithms can be directly applied in systems with different levels of parallelism, resources and programmability characteristics, such as MPSoCs or dedicated hardware designs.

In this section, we present experimental results and perform the analysis considering: (1) parallelism granularity; (2) scalability; and (3) DMM-1 simplifications in multicore CPU.

4.1 Parallelism granularity analysis

Figure 8 displays the evaluation of both block- and pattern-based approaches in the CPU, comparing the execution time of eight threads to the execution time of a single thread. For a 1024×768 pixels frame, the average time required is 5.0 and 5.9 s for the block and pattern-based parallel approaches, respectively, whereas the sequential time is 30.5 s. For a frame with 1920×1088 pixels, the execution time was reduced from 83.2 to 16.2 s and 13.6 s, when the pattern and block-based parallel approaches are applied, respectively. These results highlight the timesaving when blocks are encoded in parallel on a multicore CPU.

The attained speedup over the single thread version is similar for the two frame sizes, reaching on average 6.1 and 5.1 for block- and pattern-based approaches, respectively. Since best results were always achieved for the block-based approach, it is the approach adopted in the next experiments for evaluating the scalability and the impact of DMM-1 simplifications. It is worthwhile to mention that the proposed parallel approaches do not insert any losses, regarding both the 3D-HEVC encoded video quality and bitrate.

4.2 Scalability analysis

We have evaluated the scalability of the block-based approach by varying the number of threads in the CPU from one to eight. Figure 9a shows the obtained results taking the single-thread as the baseline implementation. One can notice that the results obtained in different video sequences are very similar, (the curves are almost overlapped).

Figure 9b shows the efficiency regarding the number of threads that is given by Eq. (1), where E_N is the efficiency using N threads, Time_1 is the average time required

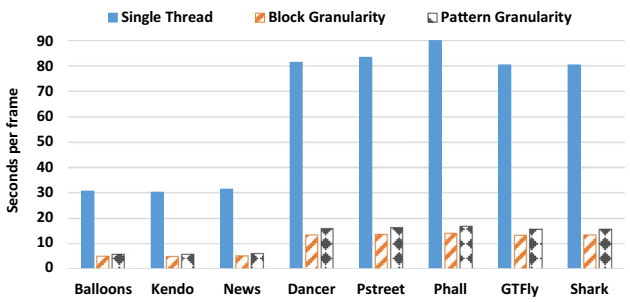


Fig. 8 Encoding time per frame considering the evaluated approaches for eight threads and one thread

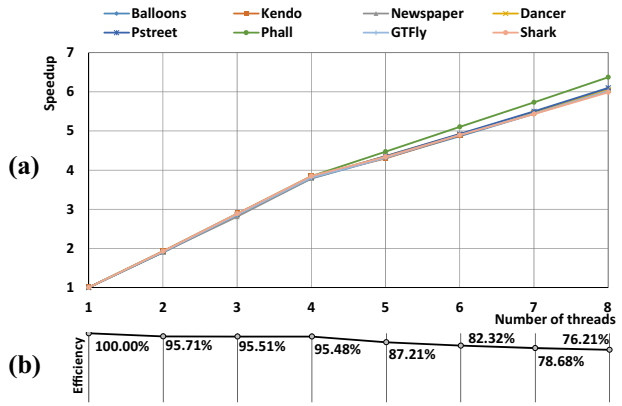


Fig. 9 Scalability analysis—**a** eight speedups and **b** the efficiency according to the number of threads for the block granularity

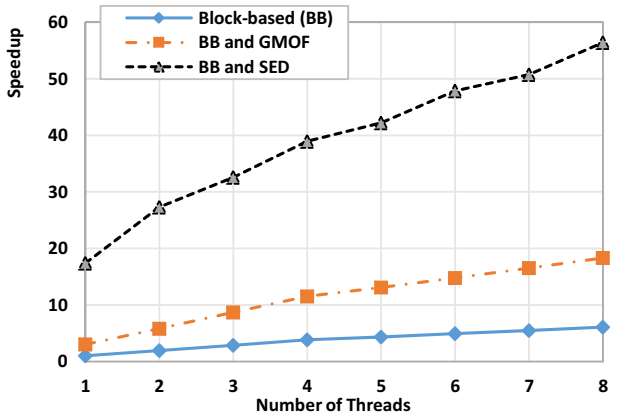


Fig. 10 Scalability analysis with DMM-1 simplifications

to encode with a single thread and Time_N is the average time required to encode using N threads.

$$E_N = \frac{\text{Time}_1}{\text{Time}_N \times N} \times 100\% \quad (1)$$

Notice that when using eight threads, the proposed approach speeds up processing in average 6.1. Besides, the efficiency decreases after five threads. This behavior is explained by the fact that the system has only four physical cores. When more than four threads are launched, hyper-threading is activated limiting the efficiency. However, the speedup curve does not saturate with a modest number of cores like the ones in current multicore systems, showing that higher speedups could be attained using processors with a higher number of CPU cores.

4.3 DMM-1 simplification analysis

To show that the proposed parallel algorithms can further integrate the DMM-1 simplification techniques, parallel programs were adapted to implement SED [14] and GMOF [16] techniques. Figure 10 depicts the speedup achieved with the two simplification schemes and varying the number of threads. The results are normalized according to the computation time of a single thread DMM-1 algorithm without any simplification.

The y axis of Fig. 10 shows that both algorithms scale with the number of cores. Table 3 summarizes the average time per encoded frame in the multicore CPU. For the 1024×768 videos, the encoding time per frame can be further reduced to 1.7 and 0.6 s when the GMOF and SED simplifications are applied, respectively. While for the 1920×1088 frames, GMOF and SED can reduce the encoding time per frame to 4.5 and 0.5 s, respectively.

4.4 DMM-1 implementation using GPU

We first used the TITAN X GPU in our analysis and later we used the TITAN Xp GPU to obtain the final results, to show the potential of further data parallelism in DMM-1 execution. We implemented the DMM-1 algorithm for the TITAN X adopting the same block-based approach employed in the multicore CPU implementation. It is expected that this approach provides even better results since data parallelism is fundamental to use the GPU resources efficiently. We have evaluated the processing rate in frames per second (fps) and the speedup achieved using the GPU programmed with CUDA, using as the baseline for the single thread CPU execution time. These results, along with the time spent during data transfer between CPU and GPU, are rendered in Table 4.

This basic GPU implementation running at the TITAN X achieves 18.6 fps@1024×768 and 14.6 fps@1920×1088. Comparing the processing rate with the CPUs one, one can notice that a higher increase in the processing rate is obtained for higher resolution videos, because more blocks are encoded by the CUDA cores, increasing the data parallelism explored. Moreover, the time spent on data transfers represents less than 2%, which means that most of the time is spent on the GPU processing. However, further investigation

Table 3 Multicore CPU results

Implementation	Time per frame (s)	
	1024×768	1920×1088
Single thread	30.5	83.2
Eight threads		
Block-based approach	5.0	13.6
Block-based approach with GMOF	1.7	4.5
Block-based approach with SED	0.6	0.5

Table 4 results for the basic GPU implementation—TITAN X

Resolution	Videos	fps	Communication and frame read percentage (%)
1024×768	Balloons	18.6	0.9
	Kendo	18.6	0.9
	News	18.5	0.9
	Average	18.6	0.9
1920×1088	Dancer	14.6	1.8
	PStreet	14.6	1.8
	PHall	14.7	1.7
	GTFly	14.7	1.8
	Shark	14.6	2.0
	Average	14.6	1.8

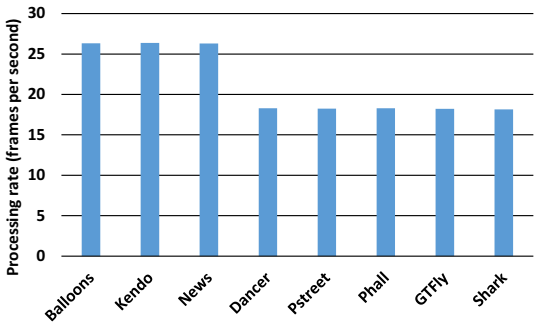


Fig. 11 Results of the optimization for TITAN X

is required to obtain an improved processing rate for real-time encoding of HD 1080p videos.

We improved the basic TITAN X GPU implementation by increasing the number of CUDA streams to maximize the usage of CUDA cores and by taking advantage of the GPU constant memory. With the CUDA streams optimization, data transfers between CPU and GPU are performed asynchronously, and each kernel is executed in a different stream, improving the use of the CUDA cores. Moreover, the constant memory was used to store the 4×4 and 8×8 pattern blocks. The 16×16 patterns were not stored in the constant memory because there was not enough space for them in TITAN X (neither in TITAN Xp that will be used later). Therefore, higher processing rates can be achieved employing a GPU with larger constant memory.

Figure 11 shows the processing rate obtained. On average, the system was capable of reaching 26.3 fps for 1024×768 videos and 18.2 fps for 1920×1088 videos.

We have also implemented and evaluated the two simplification techniques for DMM-1, GMOF and SED, in the TITAN X GPU, keeping the streams and the constant memory optimizations. Table 5 shows that, on average, GMOF and SED allow encoding 1024×768 resolution videos at 48.1 and 30.1 fps, respectively, while around 30 fps are encoded for 1920×1088 resolution videos. The best performance is achieved with GMOF, unlike the multicore CPU that takes more advantage of the SED. It happens because all threads of the GPU have to execute the same code in parallel in each warp. Consequently, if the SED algorithm of a thread decides to skip the DMM-1 evaluation, threads diverge and have to wait until the remaining blocks finish their computation. In this context, SED only takes full advantage of the GPU resources when all blocks inside a warp are skipped, i.e., when all blocks in the warp are classified as homogeneous. However, GMOF algorithm always provides a significant speedup. Considering the different characteristic of the two types of architectures, one can conclude that a multicore CPU execution can obtain higher benefits from algorithms that skips the entire DMM-1 evaluation, while GPUs obtain better results when applying algorithms that accelerate the processing of every executing block in a convergent way.

We also evaluated these two DMM-1 simplification techniques into a quite recent GPU platform: the NVidia TITAN Xp. The reached results for this second GPU implementation is also presented in Table 5, considering the block-based

Table 5 Titan X and Titan Xp processing rates results for GMOF and SED implementations

Resolution	Videos	TITAN X (fps)			TITAN Xp (fps)		
		Block-based	GMOF	SED	Block-based	GMOF	SED
1024×768	Balloons	26.3	50.3	30.2	40.5	106.8	42.3
	Kendo	26.4	49.2	31.7	40.6	102.4	45.0
	News	26.3	44.9	28.7	40.4	87.2	40.3
	Average	26.3	48.1	30.1	40.5	98.8	42.5
1920×1088	Dancer	18.3	30.9	27.3	26.3	56.1	36.9
	PStreet	18.2	30.1	37.1	26.2	51.7	56.1
	PHall	18.3	31.3	32.9	26.5	57.7	46.8
	GTFly	18.2	30.0	31.6	26.2	53.3	56.3
	Shark	18.1	29.9	27.0	26.0	54.2	37.6
	Average	18.2	30.4	30.8	26.2	54.6	44.6

approach with optimizations, and with the usage of GMOF and SED techniques. Similar conclusions than that reached for the previous experiment with the TITAN X GPU can be drawn. Again, GMOF algorithm presented higher processing rates because SED algorithm requires that some threads perform the full DMM-1 algorithm without any simplification. Therefore, the threads that skip DMM-1 processing early wait until all threads inside the warp finish. The processing rates in this evaluation reached up to 98.8 fps for 1024×768 videos and 54.6 fps for 1920×1088 videos using GMOF algorithm.

Finally, Table 6 summarizes the multicore CPU and GPU results obtained along this work. Our GPU implementation increases the processing rate from 0.03/0.01 fps (for 1024×768/1920×1088 videos) to 98.8/54.6 for these resolutions when using GMOF technique running on a TITAN Xp.

Experimental evaluation in this article was made for a specific CPU and two specific GPUs. However, one can conclude that the proposed parallel approach is scalable, then it can be extended to speed up the DMM-1 execution on other parallel systems, with different characteristics, such as MPSoCs or dedicated hardware design.

The best performance was achieved by applying the proposed parallel strategies and exploring the SED and GMOF simplification techniques. The parallelism exploration using both SED and GMOF techniques can be reached because these simplifications do not contain dependencies with the remaining blocks in the current frame.

SED skips the entire DMM-1 evaluation for some blocks, without any dependency on neighbor blocks. Simplifications like the ones presented in [12, 13] cannot obtain these acceleration benefits because they use the data contained in RD-list, and the RD-list construction requires data from neighbor blocks, avoiding a massively parallel exploration.

The GMOF technique reduces the number of wedgelets evaluated in DMM-1 execution. Again, for the parallelization purpose, it is necessary that the algorithm does not contain dependencies with neighbor blocks. Therefore, the algorithms designed in [13, 15] should obtain similar processing rate results than GMOF algorithm, while the algorithm proposed in [17] should not be a good candidate for exploring the parallelism since it has dependencies with neighbor blocks.

Considering the previous discussions about the reached results one can conclude that DMM-1 can be a good candidate for being accelerated using massive parallel approaches. In all cases, the frame rates were scalable with the explored parallelism according to the target CPU/GPU architectures. It is important to emphasize that the DMM-1 tool represents around 20% of the 3D-HEVC computational effort [10], then this is an important tool that must be accelerated to allow the design of real-time encoders able to process high-resolution videos at 30 fps or more. A similar approach than that used in this article for DMM-1 tool can be used for other encoder tools allowing a high throughput also for these modules. Besides, other solutions can also be explored in the other encoder modules, targeting the system acceleration, including the use of multiple GPUs, MPSoCs, dedicated VLSI designs or other high-performance solutions. Then, integrating these solutions will be possible to have a complete 3D-HEVC encoder processing high-resolution videos in real-time.

Table 6 Summary of the reached results

Implementation	Frames per second	
	1024×768	1920×1088
Single thread	0.03	0.01
Eight threads		
Block-based approach	0.22	0.07
Block-based approach with GMOF	0.59	0.22
Block-based approach with SED	1.80	2.05
TITAN X		
Block-based approach	26.30	18.20
Block-based approach with GMOF	48.10	30.10
Block-based approach with SED	30.40	30.80
TITAN Xp		
Block-based approach	40.50	26.20
Block-based approach with GMOF	98.80	54.60
Block-based approach with SED	42.50	44.60

5 Conclusions and future work

This article presented a parallelism exploration over the 3D high-efficiency video coding (3D-HEVC) Depth Modeling Mode 1 (DMM-1) encoding tool through multicore CPU and GPU implementations. Parallel exploration strategies were proposed for the DMM-1 tool and programmed using OpenMP and CUDA. Three main features were evaluated when defining the used strategy: parallelism granularity, scalability, and compatibility with simplification techniques. We used an Intel Core i7 6700K and two NVidia GPUs (GTX Titan X and Titan Xp) to evaluate the parallelism exploration results.

Our analysis demonstrated that the designed parallel approach can obtain scalable speedup benefits when running in systems with more available cores even using simplification heuristics. Two simplifications were explored together with the parallelism exploration: SED and GMOF. These heuristics can reduce significantly the processing time causing a BD-rate drop of only 0.94% and 0.33%, respectively. With the usage of our best parallel

approach together with a DMM-1 simplification algorithm, we demonstrated that up to 98.8 and 54.6 frames per second can be encoded using the massive parallelism provided by GPUs for 1024×768 and 1920×1088 video resolutions, respectively. Therefore, the results achieved with GPU enables real-time 3D-video encoding for these high resolutions.

This approach can be extended for other encoder tools also intending to design 3D-HEVC encoders able to process high-resolution videos in real-time.

Acknowledgements This article was achieved in cooperation with Hewlett–Packard Brazil Ltda. using incentives of Brazilian Informatics Law (Law no. 8.248 of 1991). Authors would like to thanks CNPq, FAPERGS and CAPES Brazilian research agencies (processes 88881135737/2016-01 and 88881119481/2016-01) to support the development of this work. National Funds also supported this work by the through Fundação para a Ciência e a Tecnologia (FCT) under Project UID/CEC/50021/2013.

References

1. Sullivan, G., Ohm, J., Han, W., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012)
2. Tech, G., Chen, Y., Muller, K., Ohm, J., Vetro, A., Wang, Y.: Overview of the multiview and 3D extensions of high efficiency video coding. IEEE Trans. Circuits Syst. Video Technol. **26**(1), 35–49 (2016)
3. Sullivan, G., Boyce, J., Chen, Y., Ohm, J., Segall, C., Vetro, A.: Standardized extensions of high efficiency video coding (HEVC). IEEE J. Sel. Topics Signal Process. (J-STSP) **7**(6), 1001–1016 (2013)
4. Kauff, P., Atzpadin, N., Fehn, C., Muller, M., Schreer, O., Smolic, A., Tanger, R.: Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability. Sig. Process. Image Commun. **22**(2), 217–234 (2007)
5. Fehn, C.: Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. Stereosc. Disp. Virtual Real. Syst. (SPIE) **5291**, 93–104 (2004)
6. Nagoya University. FTV Test Sequences. (2018). http://www.fujii.nuee.nagoya-u.ac.jp/~fukushima/mpegftv/. Accessed Jun 2018
7. Merkle, P., Muller, K., Marpe, D., Wiegand, T.: Depth intra coding for 3D video based on geometric primitives. IEEE Trans. Circuits Syst. Video Technol. **26**(3), 570–582 (2015)
8. Liu, H., Chen, Y.: Generic segment-wise DC for 3D-HEVC depth intra coding. In: Proc. IEEE International Conference on Image Processing (ICIP), pp. 3219–3222, (2014)
9. Lee, J., Park, M., Kim, C.: 3D-CE1: depth intra skip (DIS) mode. document JCT3V-K0033, Geneva, Switzerland (2015)
10. Sanchez, G., Agostini, L., Marcon, C.: 3D-HEVC depth maps intra prediction complexity analysis. In: Proc. IEEE international conference on electronics, circuits, & systems (ICECS), pp. 348–351 (2016)
11. Souza, D., Ilic, A., Roma, N., Sousa, L.: GHEVC: an efficient HEVC decoder for graphics processing units. IEEE Trans. Multimed. **19**(3), 459–474 (2017)
12. Gu, Z., Zheng, J., Ling, N., Zhang, P.: Fast intra prediction mode selection for intra depth map coding. ISO/IEC JTC1/SC29/WG11, Vienna (2013)
13. Zhang, Q., Yang, Q., Chang, Y., Zhang, W., Gan, Y.: Fast intra mode decision for depth coding in 3D-HEVC. Multidimension. Syst. Signal Process. **28**(4), 1203–1226 (Oct. 2017)
14. Sanchez, G., Saldanha, M., Balota, G., Zatt, B., Porto, M., Agostini, L.: Complexity reduction for 3D-HEVC depth maps intra-frame prediction using simplified edge detector algorithm. In: Proc. International Conference on Image Processing (ICIP), pp. 3209–3213 (2014)
15. Fu, C., Zhang, H., Su, W., Tsang, S., Chan, Y.: Fast wedgelet pattern decision for DMM in 3D-HEVC. In: Proc. IEEE International Conference on Digital Signal Processing (DSP), pp. 477–481 (2015)
16. Sanchez, G., Saldanha, M., Balota, G., Zatt, B., Porto, M., Agostini, L.: A Complexity reduction algorithm for depth maps intra prediction on the 3D-HEVC. In: Proc. Visual Communications and Image Processing (VCIP), pp. 137–140 (2014)
17. Sanchez, G., Jordani, L., Marcon, C., Agostini, L.: DFPS: a fast pattern selector for depth modeling mode 1 in three-dimensional high-efficiency video coding standard. J. Electron. Imaging **25**(6), 063011–063011 (2016)
18. Dagum, L., Menon, R.: OpenMP: an industry standard API for shared-memory programming. IEEE Comput. Sci. Eng. **5**(1), 46–55 (1998)
19. NVIDIA CUDA Programming guide, NVIDIA Corporation, version 2.0 (2008)
20. 3D-HEVC Test Model. (2017). https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-16.2/. Accessed Oct 2017
21. Marpe, D., Schwarz, H., Bosse, S., Bross, B., Helle, P., Hinz, T., Kirchhoffer, H., Lakshman, H., Nguyen, T., Oudin, S., Siekmann, M., Suhring, K., Winken, M., Wiegand, T.: Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding. IEEE Trans. Circuits Syst. Video Technol. (TCSVT) **20**(12), 1676–1687 (2010)
22. Lainema, J., Bossen, F., Han, W., Min, J., Ugur, K.: Intra coding of the HEVC standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1792–1801 (2012)
23. Zhao, L., Zhang, L., Ma, S., Zhao, D.: Fast mode decision algorithm for intra prediction in HEVC. In: Proc. IEEE visual communications and image processing (VCIP), pp. 1–4 (2011)
24. Vosters, L., Varekamp, C., Haan, G.: Overview of efficient high-quality state-of-the-art depth enhancement methods by thorough design space exploration. J. Real-Time Image Process. (JRTIP) 1–21 (2015)
25. Budagavi, M., Fuldseth, A., Bjontegaard, G.: HEVC transform and quantization. In: High Efficiency Video Coding (HEVC): Algorithms and Architectures, Springer, New York (2014)
26. Marpe, D., Schwarz, H., Wiegand, T.: Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. IEEE Trans. Circuits Syst. Video Technol. (TCSVT) **13**(7), 620–636 (2003)
27. Sanchez, G., Marcon, C., Agostini, L.: Real-time scalable architecture for 3D-HEVC bipartition modes. J. Real-Time Image Process. (JRTIP) **13**(1), 71–83 (2017)
28. Tech, G., Schwarz, H., Müller, K., Wiegand, T.: 3D video coding using the synthesized view distortion change. In: Proc. Picture Coding Symposium (PCS), pp. 25–28, (2012)
29. Rusanovskyy, D., Muller, K., Vetro, A.: Common test conditions of 3DV Core experiments. ISO/IEC JTC1/SC29/WG11 MPEG2011/N12745, Geneva (2013)

Gustavo Sanchez is Professor at the IF Farroupilha, Brazil, since 2014. Sanchez received the Electrical Engineer degree from the Sul-Rio-Grandense Federal Institute of Education, Science and Technology (2013) and B.S. degree in Computer Science from the Federal University of Pelotas (2012). In 2014, he received his M.Sc. degree in

Computer Science from the Federal University of Pelotas. Sanchez is currently pursuing his Ph.D. degree in Computer Science at the Pontifical Catholic University of Rio Grande do Sul. He has 9+ years of research experience in algorithms and hardware architectures for video coding. His research interests include complexity reduction algorithms, hardware-friendly algorithms and dedicated hardware design for 2D and 3D video coding.

Luciano Agostini received the M.S. and Ph.D. degrees from Federal University of Rio Grande do Sul, Porto Alegre, Brazil, in 2002 and 2007 respectively. He is a Professor since 2002 at Federal University of Pelotas (UFPel), Brazil, where he leads the Video Technology Research Group (ViTech) and the Group of Architectures and Integrated Circuits (GACI). From 2013 to 2017, he was the Executive Vice President for Research and Graduate Studies of UFPel. He has more than 200 published papers in journals and conference proceedings. His research interests include 2D and 3D video coding, algorithmic optimization, arithmetic circuits, FPGA-based design and microelectronics. Dr. Agostini is a Senior Member of IEEE and ACM, and he is a member of the IEEE CAS, CS, and SPS societies. He is also a member of the Multimedia Systems & Applications Technical Committee (MSATC) at the IEEE CAS and a member of SBC and SBMicro Brazilian societies.

Leonel Sousa received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Técnico, Universidade de Lisboa (UL), Lisbon, Portugal, in 1996. He is currently a Full Professor with UL. He is also a Senior Researcher with the Research and Development Instituto de Engenharia de Sistemas e Computadores. His research interests include VLSI architectures, computer architectures, parallel computing, computer arithmetic, and signal processing systems. He is a fellow of the IET and a Distinguished Scientist of ACM. He has contributed over 200 papers in journals and international conferences, for which he got several awards, such as, the DASIP'13 Best Paper Award, the SAMOS'11 Stamatis Vassiliadis Best Paper Award, the DASIP'10 Best Poster Award, and the Honorable Mention Award UTL/Santander Totta for the quality of the publications in 2009. He has contributed to the organization of several international conferences and has given keynotes in some of them. He has edited four special issues of international journals, and he is currently an Associate Editor of the IEEE Transactions on Multimedia, the IEEE Transactions on Circuits and Systems for Video Technology, the IEEE Access, the IET Electronics Letters, and the Springer Journal of Real-Time Image Processing, and an Editor-in-Chief of the EURASIP Journal on Embedded Systems.

César Marcon is Professor at the School of Computer Science of Pontifical Catholic University of Rio Grande do Sul (PUCRS), Brazil, since 1995. He received his Ph.D. in Computer Science from Federal University of Rio Grande do Sul, Brazil, in 2005. Professor Marcon is member of the Institute of Electrical and Electronics Engineers (IEEE) and of the Brazilian Computer Society (SBC). He is advisor of MsC. and Ph.D. graduate students at Graduate Program in Computer Science of PUCRS. He has more than 100 papers published in prestigious journals and conference proceedings. Since 2005, prof. Marcon coordinated nine research projects in areas of telecom, healthcare, and telemedicine. His research interests are in the areas of embedded systems in the telecom domain, MPSoC architectures, partitioning and mapping application tasks, fault-tolerance and real-time operating systems.