J Real-Time Image Proc (2017) 13:5–24
DOI 10.1007/s11554-016-0602-2

SPECIAL ISSUE PAPER

Complexity control of HEVC encoders targeting real-time constraints

Mateus Grellert[1] · Bruno Zatt[2] · Muhammad Shafique[3] · Sergio Bampi[1] · Jörg Henkel[3]

Received: 5 June 2015 / Accepted: 9 May 2016 / Published online: 23 May 2016

Abstract High Efficiency Video Coding (HEVC) encoders impose several challenges in computing constrained embedded applications, especially under real-time throughput constraints. This paper proposes an adaptive complexity control scheme (CCS) that dynamically adjusts the encoder to the varying computing capabilities of the hardware platform. To design an efficient scheme, an extensive complexity analysis of key HEVC encoding parameters is herein presented. For this analysis, we developed a parameterized complexity model called "arithmetic complexity," which can be widely applied to any computing platform. Our results demonstrate that the proposed scheme provides time savings ranging from 10 up to 90 % with an average error (between target and effective complexity) of 1.2 %. Our adaptability and control performance analysis show that the scheme rapidly adapts to dynamic set-point adjustments. Compared to state of the art, our complexity control achieves more accurate results and extra features (such as dynamic set-point adjustment) at the cost of minor losses in coding efficiency.

✉ Mateus Grellert
mgsilva@inf.ufrgs.br

Bruno Zatt
zatt@inf.ufpel.edu.br

Muhammad Shafique
muhammad.shafique@kit.edu

Sergio Bampi
bampi@inf.ufrgs.br

Jörg Henkel
henkel@kit.edu

[1] PPGC, Federal University of Rio Grande do Sul, UFRGS, Porto Alegre, RS, Brazil

[2] PPGC, Federal University of Pelotas, UFPel, Pelotas, RS, Brazil

[3] Chair of Embedded Systems, CES, Karlsruhe Institute of Technology, Karlsruhe, Germany

Keywords Complexity control · Complexity analysis · Embedded systems · HEVC · Video coding

1 Introduction

The demand for cheaper and more powerful mobile devices with better quality media services is ever increasing, especially for digital video applications. Streaming services like YouTube and video conferencing have grown so rapidly that according to [1] the Internet video share of bandwidth will grow from 57 in 2012 to 69 % in 2017. To maintain efficient compression with high quality demands, the Joint Collaborative Team on Video Coding (JCT-VC) developed the High Efficiency Video Coding (HEVC, official draft released in 2013) [2]. HEVC outperforms H.264/AVC in terms of coding efficiency by 39.3 % on average for the same image quality (using the Bjøntegaard Difference metric BD-Bitrate or simply BD-BR as the efficiency metric) [3, 4].

The reference software for HEVC contains every tool defined in the JCT-VC work, and it is referred as HEVC model (HM) [5]. The HM code is not a suitable option for practical encoder implementations, mainly because its goal is to support and to document every tool defined in the standard. In other words, the HEVC model disregards computational costs or energy constraints typical of real-world hardware platforms. Adaptability to real-time encoding and variable computing constraints are not implemented in the HM software. Real-time HEVC encoding involves a scenario in which computing tasks share a limited platform. In addition, a variable amount of energy (in battery-powered platforms), computation, and memory resources are available at each time interval for video encoding.

All these scenarios demand algorithmic solutions that somehow reduce encoding complexity. One possible way to solve this is to design a low-complexity encoder that limits itself to a smaller amount of computations compared to the traditional approach. However, this implies that an encoded output of lesser quality will be produced even when there are resources available to achieve a better quality compression. Alternatively, an encoding system that is capable of dynamically scaling its computation up and down according to the underlying hardware characteristics (complexity control) represents a more promising solution, in particular for embedded systems running video processing. Therefore, complexity control techniques will be applied in the encoder envisioned herein. The scenario depicted in Fig. 1 clarifies one of the challenges regarding complexity control: The amount of computation spent to encode a video sequence is not evenly distributed throughout the encoding process. Figure 1a reveals that some frames require more computation to be encoded, whereas Fig. 1b shows the same occurs among regions inside frames. These results demonstrate that it is important to carefully distribute the available computation during encoding in order to keep coding efficiency (for instance, using framewise and/or CTU-wise budgeting techniques).

The goal of this paper is to introduce an advanced complexity control scheme (CCS) for HEVC encoders targeting real-time constrained applications. The following features were envisioned for this scheme:

- *Frame-level control* the encoding time of subsequent frames is highly variable, so a frame-level control must be employed to tackle these disparities as soon as they occur.
- *CTU-level budgeting* distributing the same computation to every CTU would disregard the fact that some regions of the frame require more optimization than others, compromising the coding efficiency.
- *Controller precision* it is important to accurately adjust the complexity in order to avoid the following situations: (1) achieving less savings than required, compromising throughput or battery life, or (2) providing more than the required savings, reducing the encoding efficiency.
- *Range of complexity targets* designing a control scheme that achieves several complexity targets is convenient to extend its application. While some systems may require a reduced set of targets, others could want to finely adjust this parameter in order to always provide the best rate–distortion values.

Three major contributions are set forth in the paper:

- A model named arithmetic complexity (AC) that is able to estimate coding complexity based on the number of calls of basic coding operations;
- A complexity analysis of key HEVC parameters measured in CPU time and in cycles (estimated with the AC model) that evaluates the coding efficiency of each case using a metric that we defined herein as rate–distortion–complexity efficiency (RDCE), also designed in this work;
- A novel CCS for HEVC encoders that dynamically adapts itself to computationally constrained situations and that relies on a PID (proportional–integral–differential) feedback control loop.

2 State of the art

2.1 Complexity analysis

The work of Vanne et al. [6] presents a comparative rate–distortion–complexity analysis of HEVC and AVC codecs and an analysis showing the computational effort distribution of the HEVC components. Their results show that the most complex ones are the Fractional ME, the Integer

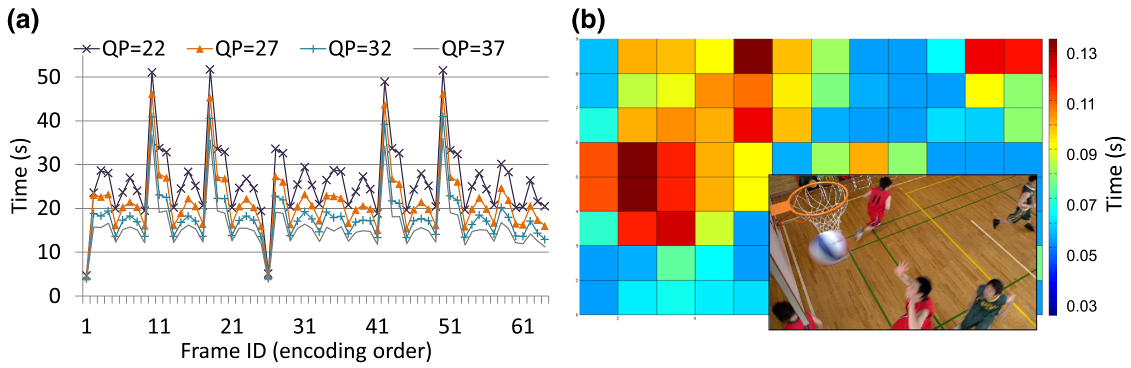

Fig. 1 Encoding time distribution among **a** frames and **b** among the 64 × 64 regions inside frames (sequence: *BasketballDrill*, 832 × 480 pixels)

ME and the T/Q/IQ/IT loop, consuming 54, 17 and 14 % of the encoding time, respectively. Hence, the tools and techniques related to the inter-prediction are responsible for most of the video encoding computations. A more fine-grained analysis of several HEVC coding parameters is presented in [7]. This analysis consisted in evaluating the PSNR, bitrate and complexity variations after varying a single encoding parameter in each test. The most relevant parameters were elected according to their impact on each of those three characteristics.

2.2 Complexity reduction

For HEVC, the most common approach toward complexity reduction consists in a strategy named *Early CU Quadtree Termination*, which saves computation by limiting the flexibility of the coding data structure. Works proposed in [8–14] belong to this category of complexity reduction. The key difference among them is how and when they decide to terminate the CU computation. None of these proposals are capable of dynamically adapting to the hardware properties and capabilities, which typically change in portable systems with varying power supplies.

Another drawback found in these works is that they also lead to quality degradation in their schemes, even when there are resources available to reduce distortion. In the opposite situation, if computations must be reduced by more than what these solutions can achieve, none of them will make it possible. The latter scenario is even more troubling in real-time applications, as it causes a frame-rate drop. As pointed in [15], frame-rate drops reduce the subjective video quality, and it should be avoided as a strategy to adapt the encoder to dynamically variable computational load or capabilities.

2.3 Complexity control

The work presented in [16] proposes a dynamic control solution for a H.264/AVC encoder, which reduces the number of modes evaluated in the mode decision phase to reduce computations and to reach target savings. In [17], a complexity-scalable Motion Estimation (ME) algorithm for H.264/AVC is proposed, which consists in reducing the complexity of this process depending upon the budget associated with each region inside the frame. The work of [18] implements a frame-level complexity control for HEVC that applies an *Early CU Quadtree Termination* algorithm until it reaches target percentage, e.g., 60 % of the regular encoding time (with no termination involved). The same authors proposed an improved scheme in [19], achieving different target complexity factors ranging from 20 to 90 % of the original encoding time. This scheme achieves good results in compression efficiency, but it does not support variable target complexity during runtime. The authors in [20] present a complexity control algorithm that decreases the number of evaluated modes based on temporal and spatial correlation, achieving a maximum reduction of 48 %. Finally, Kannangara et al. [21] present a detailed discussion on complexity control and reduction. The final work of the first author [22] was a hybrid complexity control system coupled to an H.264/AVC encoder, considering real-time and constant frame-rate constraints. Their control mechanism is very simple, since it consists in turning on and off its complexity reduction algorithm (namely SKIP prediction) based on the complexity budget.

None of the previous works cited present a detailed discussion about the characteristics of the controller. In addition, only [19] presented an algorithm capable of achieving target complexities as low as 20 %, and the presented solution does not support scenarios in which the target complexity varies dynamically. This is a common situation in practical applications: For instance, the battery level decreases constantly as a video is being encoded, so the encoder should be able to react to this and to reduce its complexity in order to record the scene as long as possible with a continuous and controlled drop in video quality or coded video bit rate.

3 Complexity assessment of HEVC parameters

This section presents an analysis of how each coding parameter affects the overall complexity of HEVC encoders. Two metrics were used: (1) processing time and (2) arithmetic complexity (later explained in this section). The results of this analysis helped identifying the parameters that were relevant to design the schemes detailed in Sect. 4.

High Efficiency Video Coding divides the frame into blocks named CTU. Each CTU can be encoded in a variety of distinct ways with the possibility to iteratively subdivide themselves into four smaller coding units (CUs), forming a quadtree structure. Each CU can be further subdivided into prediction units (PUs) during prediction and into transform units (TUs) at the transforms stage. Figure 2 shows an example of a quadtree partitioning (Fig. 2a), as well as the symmetric and asymmetric PUs evaluated in each CU node (Fig. 2b). The relationship between coding efficiency and quality of each possible encoding mode is measured by a metric called rate–distortion (RD) cost, so this process is named rate–distortion optimization (RDO) mode decision. Determining the RD cost of each mode requires a significant amount of computations.

3.1 HEVC encoding parameters

In this work, we decided the set of parameters relevant to complexity control based on results published in [6], which show that more than 85 % of the encoding time is spent in

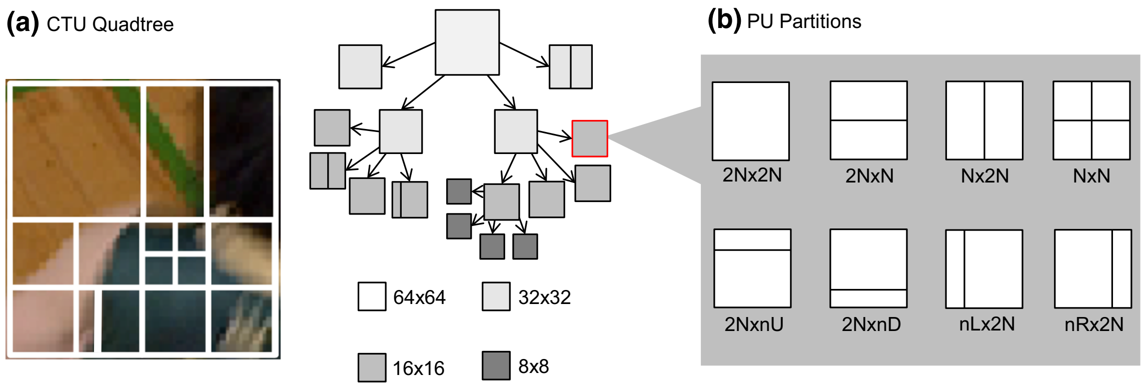

Fig. 2 **a** A CTU and its correspondent decision quadtree and **b** symmetric (*top*) and asymmetric (*bottom*) PU partitions inside each CU

the inter-prediction and transform/quantization modules. Therefore, the parameters related to these steps are more likely to have a higher impact in the overall encoding computation effort. The following paragraphs describe each of them.

- *Asymmetric Motion Partition* (*AMP*) this parameter switches on and off the evaluation of asymmetric partitions during inter-prediction.
- *Hadamard ME* (Had_{ME}) it enables the Sum of Absolute Transformed Differences (SATD) as similarity criterion during fractional ME. When $\mathrm{Had}_{\mathrm{ME}}$ is enabled, SATD is used in substitution to the sum of absolute differences (SAD).
- *Max CU partition depth* (CU_D) it controls the maximum CTU quadtree depth allowed during mode decision. This parameter ranges between 1 and 4 levels. Both the inter-prediction and the transforms are affected by this parameter.
- *Search Range* (*SR*) it defines the range in which the integer ME is applied. The SR directly affects ME complexity, as the fast algorithm implemented in HEVC uses its search range as one of its termination conditions [5].
- *Fractional Motion Estimation* (*FME*) this parameter was introduced as a fractional ME enabler. Three modes were defined: no FME (FME = 0), half-pel only (FME = 1) and quarter-pel FME (FME = 2). FME is very time-consuming due to its half-/quarter-pixel interpolations.
- *Number of Reference Frames* (*REF*) by default, the HM software runs ME on four reference frames. This parameter was introduced to dynamically reduce this number.
- *Max TU depth* (TU_D) similar to CU_{D}, controls the maximum depth in the TU quadtrees processed after the inter-prediction. The amount of operations done in the transforms module is directly affected by this parameter.

3.2 The AC metric

The HM software takes a considerable amount of time to encode, taking approximately 1 hour to encode a single second of 2560 × 1600 videos in an Intel i7 CPU at 3.4 GHz. When exploring different execution platforms, this process can become even more time-consuming. In the Gem5 architecture simulator, for instance, each simulated second takes about 10^4 host seconds [23]. Although processing time is the default measurement for many complexity-driven studies, following this methodology can be quite discouraging for computation-intensive applications such as the HM reference, especially if simulators like Gem5 are intended.

The lack of a general complexity model for video encoding applications forces researchers to spend a lot of time encoding sequences under different configurations. In addition, the HM software also limits the scope of architectural analysis, as it does not support parallelism optimizations in its current version.

To solve part of this problem, a parameterized model is introduced and defined in this work, which is able to estimate encoding complexity based on two measurements: (1) the number of calls of encoding kernels, which is *application* dependent, and (2) the time spent in each kernel, which depends on the *hardware platform*. The resulting metric herein proposed is named AC, presented in the equation below:

$$\begin{aligned} AC(s) &= T_{DISTORTION}(s) + T_{INTERPOLATION}(s) + T_{TRANSFORMS}(s) \\ AC(s) &= (N_{SAD} \cdot t_{SAD} + N_{SSE} \cdot t_{SSE} + N_{SATD} \cdot t_{SATD}) \\ &\quad + (N_{HI} \cdot t_{HI} + N_{QI} \cdot t_{QI}) + (N_T \cdot t_T) \end{aligned} \tag{1}$$

In (1), $N_{\langle op \rangle}$ and $t_{\langle op \rangle}$, respectively, represent the amount of occurrences of an operation and the time spent on it, whereas *HI*, *QI* and *T* stand for half-/quarter-pel interpolation and transforms. The $N_{\langle op \rangle}$ and $t_{\langle op \rangle}$ values must be accounted for each block size *s* (32 × 32–4 × 4 for transforms and $64 \times 64 - 8 \times 8$ for others), because analysis showed that these values do not scale linearly with size.

The main advantage of the AC metric is that it is possible to run the HM encoder a single time on any platform and use the acquired $N_{\langle op \rangle}$ values to estimate the encoding time on different scenarios. One must simply adjust the time required for each operation involved in the computation kernels in order to obtain the corresponding complexity estimation. The results presented herein will make use of this feature to provide results for three case studies.

3.3 Methodology and test setup

Figure 3 details the methodology (left side of Fig. 3) and resources (right side) applied in this work to compute the AC on different platforms.

In Fig. 3, RDTSC stands for Read Time Stamp Counter. This tool is an assembly implementation that is able to monitor CPU cycles at nanosecond accuracy, which is more accurate than standard C library timers [24]. The three case studies considered in step 3 are detailed in Table 1.

The SIMD instructions were disabled in all cases, as it would reduce the measurement accuracy (some kernels were computed in less than 1 ns with SIMD). Case Studies 1 and 2 are modeled to resemble desktop and embedded platforms, whereas Case Study 3 represents an embedded device that contains dedicated hardware accelerators for some kernels. These accelerators were modeled from published results of hardware architectures for HEVC. In [25], a parallel 64×64 SAD architecture is proposed that is capable of processing a 64×64 CU in 64 cycles operating at 171.8 MHz. In [26], an N-Point DCT architecture for HEVC encoders is presented. The authors proposed a solution that is capable of processing a 32×32 TU in 136 cycles at 150 MHz frequency. Lastly, [27] presents an interpolation filter accelerator that achieves a 12-pel/cycle throughput operating at 312 MHz, adding up to 1 and 2 k cycles for 64×64 half- and quarter-pel interpolations.

Table 2 lists the sequences encoded with the HM software, as well as the frame count for each. The simulations were executed for QP values of 22, 27, 32 and 37, as recommended in the Common Test Conditions (CTC) document [28].

The coding configurations used in the complexity sensitivity analysis are listed in Table 3. The default configuration (c0 in Table 3) was used as reference, and the others were defined by varying a single parameter from c0. The parameter that is altered in each case is displayed in bold. This was done in order to accurately define the relation between each coding parameter and its coding efficiency.

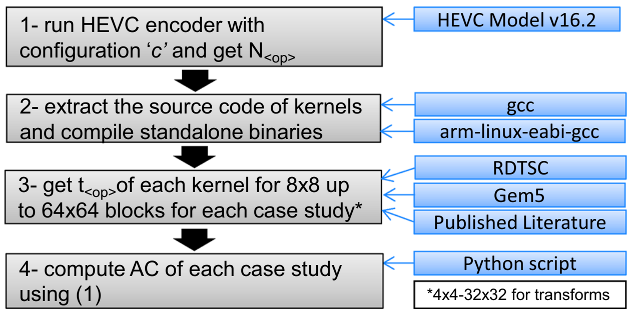
1- run HEVC encoder with configuration ‘c’ and get N<op>
2- extract the source code of kernels and compile standalone binaries
3- get t<op> of each kernel for 8x8 up to 64x64 blocks for each case study*
4- compute AC of each case study using (1)
HEVC Model v16.2
gcc
arm-linux-eabi-gcc
RDTSC
Gem5
Published Literature
Python script
*4x4-32x32 for transforms

Fig. 3 Methodology used to extract the arithmetic complexity of each case study

3.4 AC analysis

Table 4 shows the acquired timing results for each block size, considering the platforms of each case study. These were the values combined with the number of calls of each kernel to calculate the AC.

The following charts present encoding time estimations for the three case studies, measured with the AC metric, as well as the actual processing time extracted by running the HM software on an Intel i7 host with the same specs from Case Study 1. Figure 4 displays the time to encode 64 frames of sequences with different video resolutions under the HM default configuration.

Table 1 Specifications of the three case studies

	Case Study 1	Case Study 2	Case Study 3
Processors	Intel i7-4770	ARM (ARMv7 ISA)	ARM + Accelerators
Measurement	RDSTC [24]	Gem5	Gem5 + Python
Max. Freq.	3.4 GHz	2.2 GHz	2.2 Ghz (ARM CPU)
RAM	4 GB	2 GB	4 GB
L1i/d Cache	32 kB	16 kB	16 kB
L2 Cache	256 kB	1 MB	1 MB
L3 Cache	8 MB	N/A	N/A

Table 2 Input sequences and specifications

Class	Resolution	Sequences	Frame count
A	2560 × 1600	PeopleOnStreet	64
B	1920 × 1080	BasketballDrive	64
C	832 × 480	BQMall	64
D	416 × 240	BlowingBubbles	64
E	1280 × 720	Johnny	64
F	1024 × 768	ChinaSpeed	64

Table 3 Configurations used in the analysis

	AMP	FME	Had_{ME}	CU_D	REF	SR	TU_D
c0[a]	1	2	1	4	4	64	3
c1	**0**	2	1	4	4	64	3
c2	1	**0**	1	4	4	64	3
c3	1	**1**	1	4	4	64	3
c4	1	2	**0**	4	4	64	3
c5	1	2	1	**1**	4	64	3
c6	1	2	1	**2**	4	64	3
c7	1	2	1	**3**	4	64	3
c8	1	2	1	4	**2**	64	3
c9	1	2	1	4	**1**	64	3
c10	1	2	1	4	4	**16**	3
c11	1	2	1	4	4	**4**	3
c12	1	2	1	4	4	**0**	3
c13	1	2	1	4	4	64	**1**
c14	1	2	1	4	4	64	**2**

[a] Default/reference configuration

The first conclusion from Fig. 4 is that the AC metric presents a very high correlation (0.99) with the actual processing time on a host with the same specifications, proving that this model is extremely accurate. The ratio between both curves shows that the AC kernels alone consume 57 % of the processing time. The remaining 43 % is spent on other kernels, memory traffic, data structures, stack management and control overhead, etc. However, this ratio becomes smaller as resolution increases, showing that the time spent on memory traffic is more important for larger resolutions. Following this trend, more than 50 % of the encoding time is expected to be spent on memory traffic for 4 K UHD (3840 × 2160 pixels) sequences. Still in Fig. 4, one can notice the importance of dedicated accelerators for real-time encoding applications, as Case Study 3 presented an average speedup of 12.9 against its CPU-only counterpart. Note, however, that this gain is only in processing and real implementations should also consider the overhead of data transfers.

Since the AC measurements proved to be accurate, this metric was used in the control scheme later presented in this manuscript.

3.5 Encoder sensitivity analysis

The goal of this analysis is to identify the parameters that reduce complexity and use this knowledge in the control scheme design. Quality is another important factor, so the Bjontegaard-Delta Bitrate (BD-BR) [4] results were included in the analysis. The time savings results were measured with processing time. The AC savings were omitted, as they were very similar (average deviation of 0.7 %).

Figure 5 shows the time savings of each test with respect to the default HM configuration (c0 in Table 3). This configuration is taken as a baseline for all the comparisons discussed in this section.

Table 4 Time per call of each kernel for all case studies

Block size[a]	Platform	SAD	SATD	SSE	Transf.	Half Int.	Quarter Int.
8 × 8	Intel	0.05	0.22	0.16	0.04	0.95	1.78
	ARM	0.11	0.44	0.08	0.11	4.72	8.41
	ARM + Acc.	0.01	0.44	0.08	0.04	0.06	0.12
16 × 16	Intel	0.09	0.88	0.52	0.17	2.47	4.64
	ARM	0.20	1.71	0.31	0.55	15.55	31.70
	ARM + Acc.	0.02	1.71	0.31	0.17	0.22	0.44
32 × 32	Intel	0.36	4.26	2.16	1.19	7.67	14.42
	ARM	0.73	6.77	1.28	3.73	59.55	128.19
	ARM + Acc.	0.09	6.77	1.28	1.19	0.85	1.71
64 × 64	Intel	1.43	14.05	8.67	7.81	27.39	50.05
	ARM	3.33	38.27	11.65	29.12	230.41	510.68
	ARM + Acc.	0.37	38.27	11.65	7.81	3.35	6.70

Unit: µs

[a] For transforms: 4 × 4 up to 32 × 32

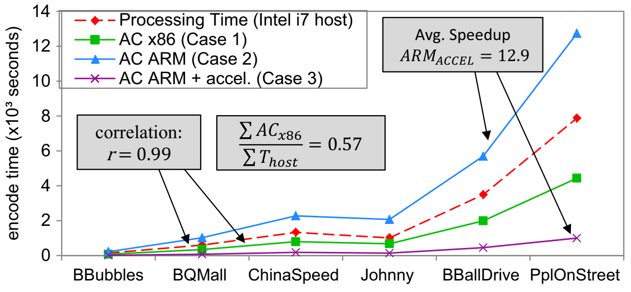

Fig. 4 Time to encode 64 frames of sequences with different resolutions using the default configuration (QP average)

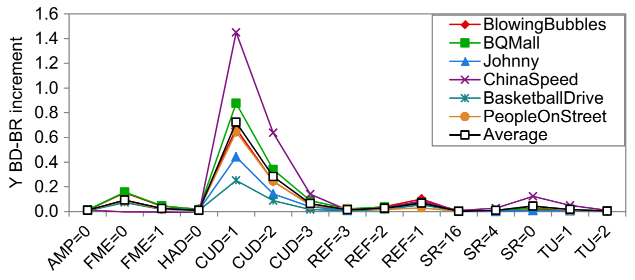

Fig. 6 Unitary Y BD-BR increment ratio of each configuration with respect to the default one

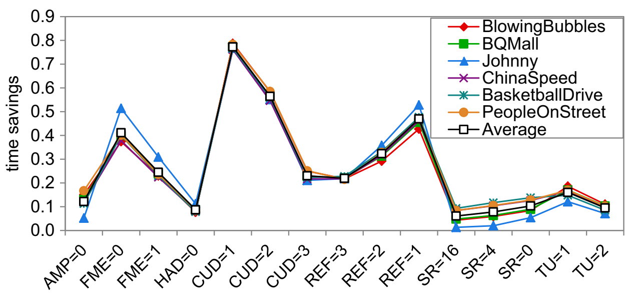

Fig. 5 Unitary time savings with respect to the default HM configuration (QP average)

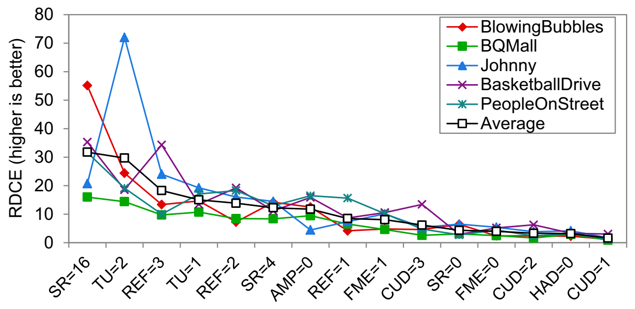

Fig. 7 Rate–distortion–complexity efficiency of the configurations used in the analysis (sorted by average RDCE)

As Fig. 5 shows, reducing the maximum CU Depth is by far the most efficient way of reducing encoding complexity, saving 77 % of the encoding time. The parameters related to Motion Estimation take the second place, achieving average savings of 47 % (reference frames = 1) and 41 % (FME mode = 0). This shows that adding these two parameters to the HM configuration set improved the quality of this analysis.

The following analysis (in Fig. 6) contains the Y BD-BR increment for the same test cases. A 1 % BD-BR increment means that bitrate was increased by 1 % for the same objective quality (measured in PSNR), so the goal is to minimize this value in order to maintain coding efficiency. Note that the parameter that most increased the bitrate is also the CU depth (72 % on average). It is also worth mentioning that limiting the number of reference frames not only saves more time, as it is also better than disabling the FME for coding efficiency (6.7 % against 9.3 % BD-BR increase).

With separate metrics, it is difficult to rank the parameters according to their performance in the rate–distortion–complexity trade-off. Therefore, in this work, we defined a metric that packs both complexity and BD-BR in a single value. With the values of complexity savings (ΔAC) and BD-BR increment (ΔBD_{BR}), the rate–distortion–complexity efficiency (RDCE) is calculated as shown in (2).

$$RDCE_x = \frac{\Delta AC}{\Delta BD_{BR}} = \frac{1 - (AC_x)/(AC_{Ref})}{1 - (BD_{BR_x})/(BD_{BR_{ref}})} \tag{2}$$

Each RDCE unit represents how much time is reduced per 1 % BD-BR increment. Thus, larger values of RDCE indicate that more complexity savings were achieved with small BD-BR increment and vice versa. The RDCE results are displayed in Fig. 7. The *ChinaSpeed* sequence was left out of this analysis, as its behavior deviated significantly from the remaining sequences, disturbing the average. Note that the REF parameter is now better ranked than FME due to its smaller BD-BR increment.

Based on the RDCE results, the search range is a high-sensitivity parameter in terms of compounded BD-BR and complexity. The average RDCE is at its peak (31.7) at SR = 16, denoting an efficient parameter in terms of BD-BR and complexity. From this analysis, limiting the number of reference frames and the maximum depth in the TU quadtree are also prominent parameters for reducing complexity. Lastly, the parameters related to restricting the CU quadtree depth, disabling the FME and the Hadamard ME should be avoided at all costs, as they achieved the smallest RDCE values.

In summary, the analysis in this section leads us to conclude that:

- The estimations obtained with AC model were very similar to processing time measurements. The advantage of this metric is that it can be calculated for different architectures and it is less platform-sensitive. This model is used in the CCS to measure frame time.
- Modifying some encoding parameters reduces complexity at the cost of high BD-BR increments, so these should be avoided. The rate–distortion–complexity Efficiency metric helped identifying the parameters that produce the highest savings per increment. Section 4.1 will explain how the RDCE was employed in the CCS designed in this work.

4 Complexity control scheme

In this section, a CCS composed of a PID-based control feedback mechanism and a budget allocation algorithm is presented. Two algorithms (Priority-Oblivious and Priority-Aware Allocation) were designed and implemented in the HEVC model software. The controller is applied at frame level, whereas budget allocation works at CTU-level. Figure 8 depicts the concept model of the scheme designed and tested in this work. Henceforth, the terms set point (SP) and process variable (PV) will be used interchangeably as the target computation effort and the effective computation, respectively.

Each component is detailed as follows:

- *Monitor* this component records the encoding frame time (measured with the AC metric) achieved by the HEVC encoder. When a new frame is encoded, the previous frame time (PV in Fig. 8) is sent to the controller.
- *Control logic* computes the frame budget (Budget$_{\mathrm{FRAME}}$) based on the difference between the PV and the SP. In this work, a PID controller was employed (see in Sect. 4.4).
- *Configuration Lookup Table* (*CFG-LUT*) stores the parameter presets defined in this work and their time savings (explained in Sect. 4.1).
- *CTU History Table* (*CTU-HT*) stores the depth of each CTU in the previous frame. This is used during budgeting to classify CTUs that require more or less computations.
- *Budgeting algorithm* this unit is responsible for distributing Budget$_{\mathrm{FRAME}}$ among the CTUs in the frame via parameter sets. It is composed of the CTU-HT and the CFG-LUT. This is a key component, as it tries to use as much available computation as possible, improving coding efficiency. Section 4.3 explains the two budget allocation algorithms implemented in this work.

4.1 Parameter presets

Given the fact that some encoding parameters directly affect the computations spent to encode a sequence, it is possible to grant variable computations for each CTU in a frame by assigning these parameters appropriately. Hence, six parameter sets (PS) were defined in this work, each targeting a specific complexity reduction.

The analysis presented in Sect. 3 was in fact of great importance to define which parameters are going to be used. The parameters were ranked according to their RDCE and then assigned to each PS until the target savings is reached. Procedure 1 shows the algorithm used to build each PS.

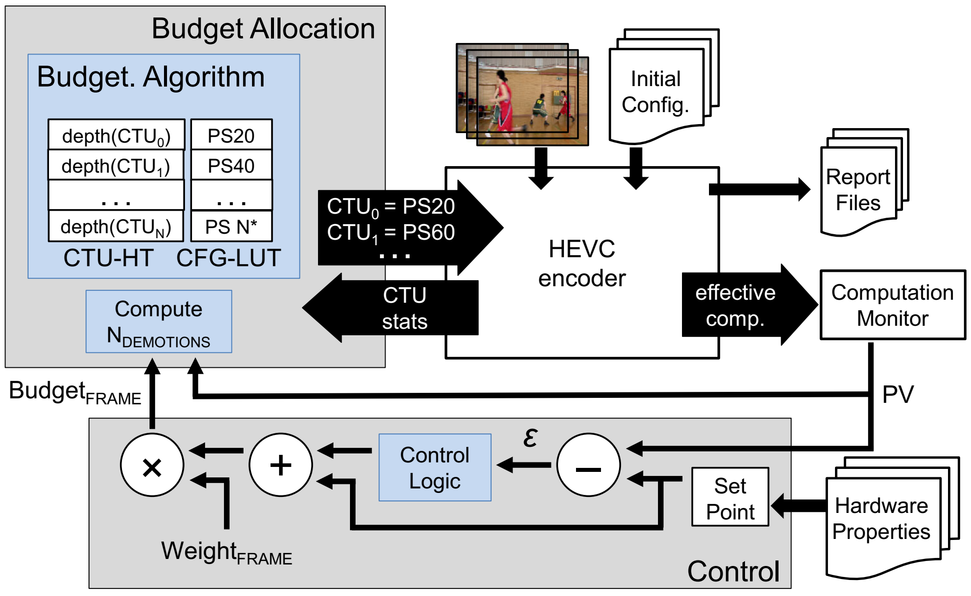

Fig. 8 Concept model of the complexity control scheme

Procedure 1 Generate parameter presets

```
Input: cfgList: [ $c0, c1, c2, \ldots$ ]                          ▷ see section 3
Input: targetSavings: [ 0.2, 0.4, 0.6, 0.8, 0.9 ]

1:  refTime $\leftarrow$ time ( encode( c0 ) )
2:  sort ( cfgList, key=RDCE )
3:  preset $\leftarrow \emptyset$
4:  presetTable $\leftarrow$ [ ]

5:  for each target in targetList do
6:      for each cfg in cfgList do
7:          preset $\leftarrow$ preset $\cup$ cfg
8:          testTime $\leftarrow$ time ( encode( preset ) )
9:          timeRatio $\leftarrow$ testTime/refTime
10:         if timeRatio $\leqslant$ target then
11:             presetTable[target] = preset
12:         else
13:             goto (line 3)
```

The input is a list of initial configurations and their RDCE values (from Sect. 3). The starting point is the default parameter preset found in the HM reference (empty preset in line 3), and a single parameter is modified in each loop (represented by the preset union in line 7). When a target is satisfied, the current preset is stored. Each new preset generated aims to achieve computation savings 20 % higher than the previous one. Sorting the configurations according to their RDCE (line 2 in Procedure 1) raises the odds of finding the best configuration in terms of BD-BR. The 90 % savings target was added in order to enable a 10–90 % savings spectrum.

Table 5 shows the final parameter sets for each target complexity. The sequences used in the analyses presented in Sect. 3 were also used to validate the time savings and AC savings of each PS. The complexity and quality results are presented in Table 6.

The target and effective savings are very similar, but not exactly the same. This is not precisely a problem, since it is possible to employ a mechanism that circumvents frame budgeting errors dynamically. In this work, this task is fulfilled with the PID controller, to be discussed in Sect. 4.4.

Table 5 Parameter sets used in the CTU budgeting

Name	Target savings (%)	AMP	FME	Had ME	CU_D	REF	SR	TU_D
PS0	0[a]	1	2	1	4	4	64	3
PS20	20	1	2	1	4	3	64	3
PS40	40	1	2	1	4	2	16	1
PS60	60	0	2	1	4	2	4	1
PS80	80	0	1	1	3	1	4	1
PS90	90	0	0	1	2	1	0	1

[a] Default configuration

Table 6 Average AC, time, and BD-BR increment variations of each PS with respect to PS100 (default configuration)

Parameter set	Target savings (%)	Effective savings (AC) (%)	Effective savings (time) (%)	Y BD-BR increment (%)
PS20	20	22.3	21.9	1.5
PS40	40	50.2	48.9	4.8
PS60	60	70.3	66.7	9.0
PS80	80	85.5	81.1	29.8
PS90	90	93.8	90.5	88.9

The parameters of each set and the average AC savings shown in Table 6 were recorded in a configuration lookup table (CFG-LUT), which is used for the complexity estimation, as detailed in the next section.

4.2 Frame budget calculation

Cycle distribution is not uneven only among CTUs in a frame, but also among frames in a group of pictures (GOP) (as Fig. 1 clearly shows). This comes from the scene characteristics, but also from the fact that HEVC defines a QP offset for each frame in a GOP, and lower QP values lead to an increase in the encoding time. Therefore, a weighting factor was defined in this work as follows:

$$w_i = \frac{\propto -QP_{OFFSET}(i)}{Size_{GOP}} \tag{3}$$

where $QP_{OFFSET}(i)$ denotes the QP increase of the ith frame in the GOP, and α must be obtained empirically, so that the sum of weights is equal to the size of the GOP. For a GOP size of four, for instance, $\propto = 6.25$.

The final budget sent to this unit is the frame budget multiplied by w_i (from (3)). The frame budget can be either the SP per se or the output of a control mechanism. The following sections describe two CTU-level budgeting algorithms designed to distribute the frame budget among its CTUs.

4.3 CTU-level budget allocation

Budget allocation is fulfilled by assigning a suitable preset for each CTU in the frame. All CTUs start with the default preset (PS0), and then, they are demoted (assigned to a less time-consuming preset) when time savings are required. The same goes when there is time to spare, allowing some CTUs to be promoted in order to minimize coding efficiency losses. Both allocation algorithms designed in this work initially execute the following steps:

Step 1. Compute the required savings as:

$$Savings_{REQ} = 1 - \left(\frac{Budget_{FRAME}}{PV}\right) \tag{4}$$

Step 2. Each consecutive preset reduces time by around 20 % (if applied to every CTU in the frame), so we calculate number of required demotions as:

$$N_{DEMOTIONS} = \frac{(Savings_{REQ} * N_{CTUs})}{0.2} \tag{5}$$

Note that when $Savings_{REQ}$ is negative (signaling complexity can be increased), $N_{DEMOTIONS}$ (also negative) will be instead the number of promotions required.

Step 3. Assign minimum and maximum presets: This limits the set of possible presets according to the required savings. For instance, if the target is 40 %, the minimum and maximum presets are PS20 and PS60. This prevents unsuited allocations if the controller overshoots.

Promote/demote the CTUs in the frame until $N_{DEMOTIONS} = 0$ or no promotions/demotions can be done.

Two procedures were used to define which CTUs should be promoted/demoted first. The first one is called Priority-Oblivious Allocation (POA) and simply demotes/promotes CTUs indiscriminately until the counter reaches zero. In contrast, the Priority-Aware Allocation algorithm uses knowledge of CTUs in the previous frame to demote the ones that were encoded with a small average depth first and also to promote first the ones with larger average depths. Procedures 2 and 3 display the pseudo-code of each procedure.

Procedure 2 Priority-Oblivious Allocation

```
Input: budget : frame budget
Input: PV : previous frame time

 1: reqSav ← 1 − ( budget / PV )                    ▷ as per (5)
 2: demotions ← nCTUs * reqSav/0.2                   ▷ as per (6)
 3: (minPSet, maxPSet) ← getMinMaxPSet ( reqSav )
 4: while demotions > 0 do
 5:     for all CTU in Frame do
 6:         success ← demote ( CTU, maxPset )
 7:         if success then               ▷ true if demotion was possible
 8:             demotions−−

 9: while demotions < 0 do
10:     for all CTU in Frame do
11:         success ← promote ( CTU, minPset )
12:         if success then
13:             demotions++
```

Procedure 3 Priority-Aware Allocation

```
Input: budget : frame budget
Input: PV : previous frame time
Input: CTU-HT : CTU depth history

1: reqSav ← 1 − ( budget / PV )                          ▷ as per (5)
2: demotions ← nCTUs ∗ reqSav/0.2                        ▷ as per (6)
3: (minPSet, maxPSet) ← getMinMaxPSet ( reqSav )
4: depth ← 0
5: while demotions > 0 do
6:     for all CTU with depth in Frame do
7:         success ← demote ( CTU, maxPset )
8:         if success then                 ▷ true if demotion was possible
9:             demotions−−
10:    depth++

11: depth ← 3
12: while demotions < 0 do
13:     for all CTU with depth in Frame do
14:         success ← promote ( CTU, minPset )
15:         if success then
16:             demotions++
17:     depth−−
```

Both algorithms have their advantages: POA is very straightforward and does not require any additional data structures, making it easy to implement in hardware encoding systems, whereas PAA has a better chance of assigning suitable presets for CTUs that require more/less computational effort, reducing coding efficiency losses. The effective savings and the rate–distortion performance of both algorithms are presented in Table 7.

As expected, the coding performance of the Priority-Aware Algorithm is far superior compared to the Priority-Oblivious strategy, presenting a BD-BR increment 47 % smaller. In addition, the PAA also achieved savings closer to the set point. Thus, we decided to employ the PAA algorithm in our control despite its extra data structure requirements.

Table 7 Allocation algorithm comparison (sequence: BQMall)

Algorithm	Target savings (%)	Effective savings (%)	Y BD-BR increment (%)
POA	40	38.8	7.1
PAA	40	39.4	3.9
POA	60	58.3	16.3
PAA	60	61.4	8.7

4.4 PID-based feedback control

The Controller calculates the budget available to encode the next frame based on how far the achieved computation is from the target computation. In this work, the set point is calculated by the following equation:

$$SP = T_{Default} * (1 - targetSavings) \qquad (6)$$

In (6), $T_{Default}$ stands for the time required to encode a frame with default configuration and *targetSavings* is the desired reduction percentage. This percentage can represent, for instance, the reduction required to encode a sequence in real time. One must simply divide the current frame rate by the desired one and use it to adjust the SP.

In this work, a proportional, integral and derivative (PID) controller [29] was implemented. A PID controller is a control feedback loop mechanism that is widely known in the industry due to its application in many systems. This controller involves three separate constant parameters: the proportional (P), integral (I) and derivative (D) values. The proportional value P depends on the present error, whereas I depends on the past values, and D can be considered as a prediction of the future ones. Our first PID-based controller for complexity control in HEVC was implemented in [30]. In this work, a more comprehensive analysis was entailed in order to produce a more robust scheme.

The PID controller receives the difference between the SP and the actual output (PV) of the system, which is considered an error (e). With this, for each time instant t, the output is computed as follows:

$$C_{PID}(t) = K_p e + K_i \sum_{i=0}^{t} e(i)\Delta t + K_d \frac{(e(t) - e(t - \Delta t))}{\Delta t} \quad (7)$$

This controller tries to match its output up with the set point at a certain speed, which depends upon the three constant parameters, namely K_p, K_i and K_d. These coefficients must be finely tuned, since inefficient configurations may cause large oscillations in the controller output (the controller output rapidly crossing the set point) or even cause the controller to diverge. There are different parameter-tuning techniques for the PID controller, and the next section describes the one used in this work.

4.4.1 PID parameters tuning

In order to employ a PID controller effectively, one must determine the most suitable values for the K_p, K_i and K_d parameters. There are several techniques to accomplish this, from manual tuning to sophisticated genetic algorithms. In this work, the widely used Ziegler–Nichols method is used [31]. According to this method, the integral and derivative component must be removed from the equation, and the proportional component must be increased at small steps until it reaches maximum gain K_u in which the output oscillates around the set point. Figure 9 shows the analysis performed using the Priority-Based Budgeting algorithm. Values from 0.2 to 3.0 were employed, but only a few were selected for a better visual analysis.

As seen in Fig. 9, some results (e.g., when $K_P = 0.2$) did not meet the set point, so they were easily discarded. For K_P equals 0.8 and 1.4, the set point was met. We concluded that a K_P of 0.8 was more suitable than 1.4, as it presented ripples with smaller magnitudes and a slightly smaller SP/PV error sum (0.96 against 1.99). Thus, we decided to choose a K_P of 0.8 as K_U.

With these values, the constants are derived from simple equations. The classic PID equations for determining the control parameters were used in this work, because they are known to fit most systems without causing the controller to diverge. The classical equations for deriving the PID parameters with the Ziegler–Nichols method are shown below:

$$K_P = 0.6K_u; K_i = 2K_p/T_u; K_d = K_p T_u/8 \quad (8)$$

Figure 10 shows an execution profile of the PBB algorithm with a 60 % target complexity. In the first test, the set point was used as budget and no PID controller was used, whereas the second profile shows the results using a PID control loop.

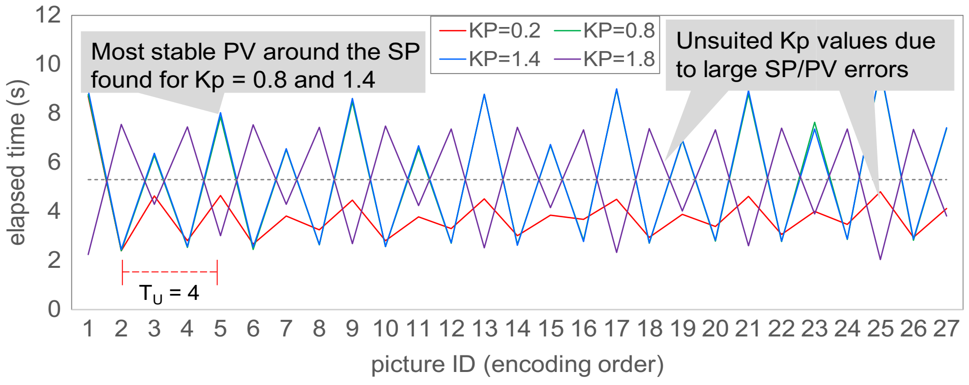

Fig. 9 PID parameter tuning with the Ziegler–Nichols Method—RaceHorses (832 × 480)

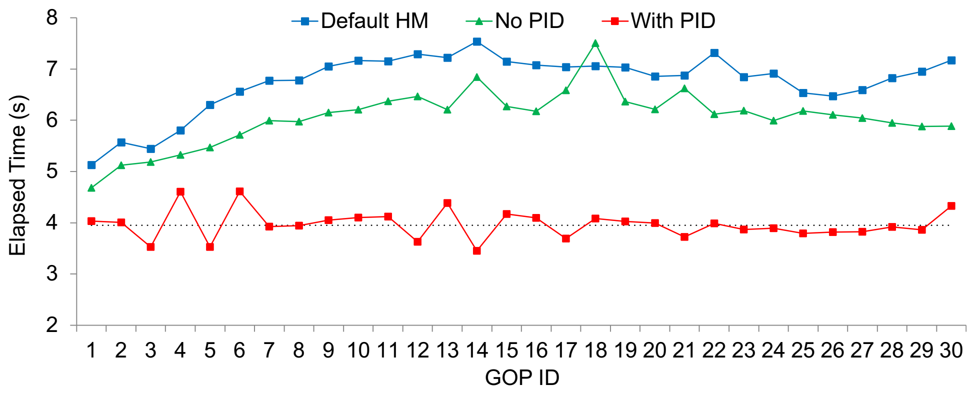

Fig. 10 Performance comparison of a complexity control scheme with and without a PID controller—BQMall (832 × 480)

It is clear that the target computation was not achieved in the first case due to estimation errors, because the sequence used in this analysis was not part of the test sequences used to build the parameter sets. Therefore, the scheme designed in this work benefits greatly with the use of a PID controller.

The following section evaluates the performance of these budgeting strategies, as well as of the PID controller against a simpler control mechanism. Afterward, a final CCS implementation that combines the best budgeting strategy with a PID feedback controller is proposed and tested against related solutions found in the literature.

5 Detailed results and comparisons

This section reports a frame-by-frame comparative analysis of the different CCSs proposed and implemented in this work. Initially, the PID controller efficacy is put to the test.

Table 8 Test conditions for the in-depth analysis

Spatial resolution	Sequences (class)
2560 × 1600	*Traffic* (A), *PeopleOnStreet* (A)
1920 × 1080	*ParkScene* (B), *BasketballDrive* (B)
1280 × 720	*FourPeople* (E), *Johnny* (E)
1024 × 768	*ChinaSpeed* (F)
832 × 480	*BQMall* (C), *RaceHorsesC* (C), BasketballDrillText (F)
416 × 240	*BasketballPass* (D), *BlowingBubbles* (D)
Frame count	64
GOP structures (mode)	IPPP (low delay), IBBB (low delay)
QP	22, 27, 32, 37, 42
Time savings	10–90 %
PID constants	*Kp, Ki, Kd* = {0.48, 0.24, 0.24}
Budget allocation	*Priority-Aware Allocation*

Afterward, the CCS is compared against the HM reference implementation and also with solutions published in related works. The Common Test Conditions define two prediction modes: low delay and random access. Low delay can be used with P or B frames, whereas only B frames are recommended in the random access mode. The modes used in this work were of low delay with P and with B frames. The CTC also defines classes for sequences based on resolution and content type. The class of each sequence will be shown in the following table.

The test configurations and the specific videos used in this analysis are detailed in Table 8. The time savings targets consider the default HM operation as 100 %. The total amount of simulations for this section alone added up to 864.

5.1 Control system response to set-point adjustments

This analysis consisted in evaluating the performance of the CCS for different target complexities. Two sequences from each class were tested to ensure input variability. The complexity targets ranged from 90 to 10 %. Lower targets are not achievable, because the results from the parameter set analysis never went below the 10 % mark. Figure 11 shows the average GOP time for different complexity targets to encode a 2560 × 1600 sequence. The SP for each target is represented by the dotted lines, and its value is indicated in the right end of the chart. In some cases, it is difficult to evaluate whether the target was achieved or not, so the incremental average is also presented in Fig. 12. The incremental average per frame was calculated using the equation below.

$$AVG_i = \frac{\sum_{j=0}^{i} PV_j}{i} \tag{9}$$

In (9), the PV for Frame$_i$ is the sum of previous results averaged to the current frame count.

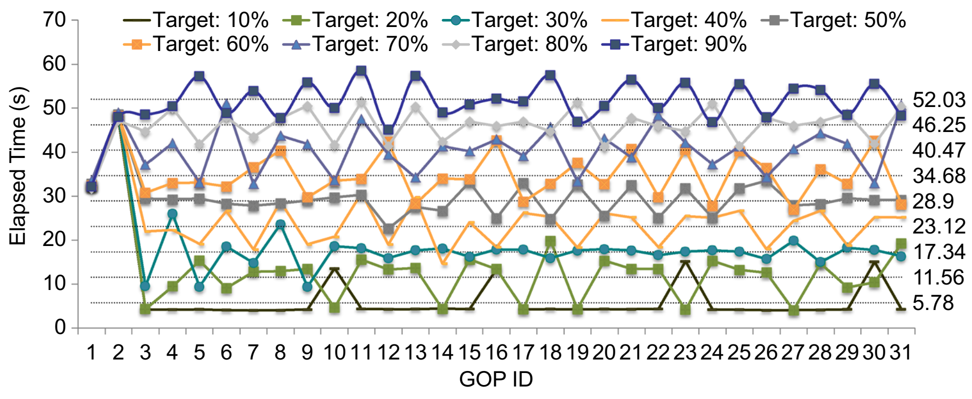

Fig. 11 Average GOP time for different target complexities (10–90 %)—Traffic (2560 × 1600)

This analysis demonstrates that the designed scheme achieves targets as low as 10 %, showing that it is highly flexible. This is very desirable, since in situations when the computation constraints are very tight (e.g., critical battery state), lower target complexities can be used to ensure that the sequence will be encoded for as long as possible until battery runs out of charge.

The accuracy of the results is also worth mentioning. The average effective complexity achieved by our CCS never differed from the target by more than 1 %, except in the very low effort case, i.e., 10 % target, which presented a 2 % difference from set point and effectively achieved. As previously stated, this great level of adaptation would not be possible without a feedback control loop.

Similar behavior can be observed in Figs. 13 and 14. In this case, a sequence with lower spatial resolution was encoded (832 × 480). The remaining sequences presented very similar results.

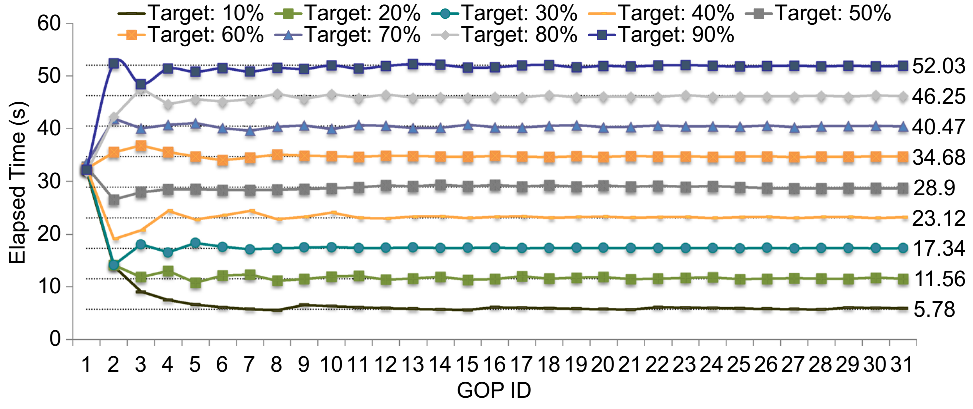

Fig. 12 Incremental average time for different target complexities (10–90 %)—Traffic (2560 × 1600)

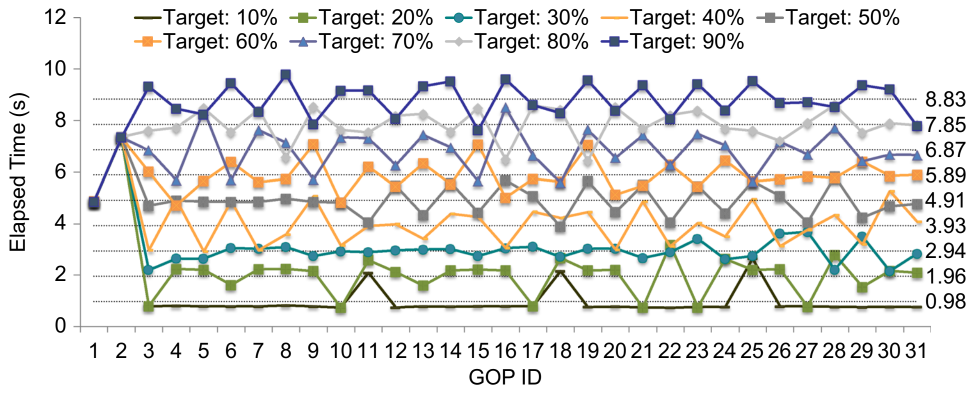

Fig. 13 Average GOP time for different target complexities (10–90 %)—RaceHorses (832 × 480)

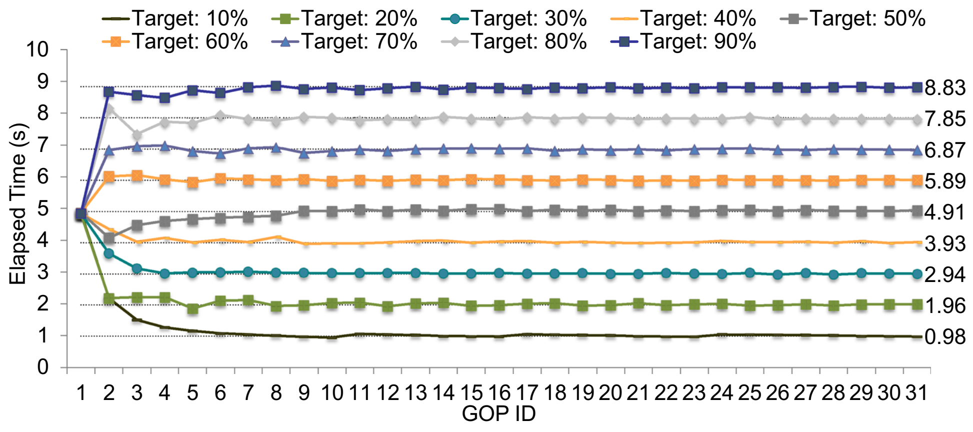

Fig. 14 Incremental average time for different target complexities (10–90 %)—RaceHorses (832 × 480)

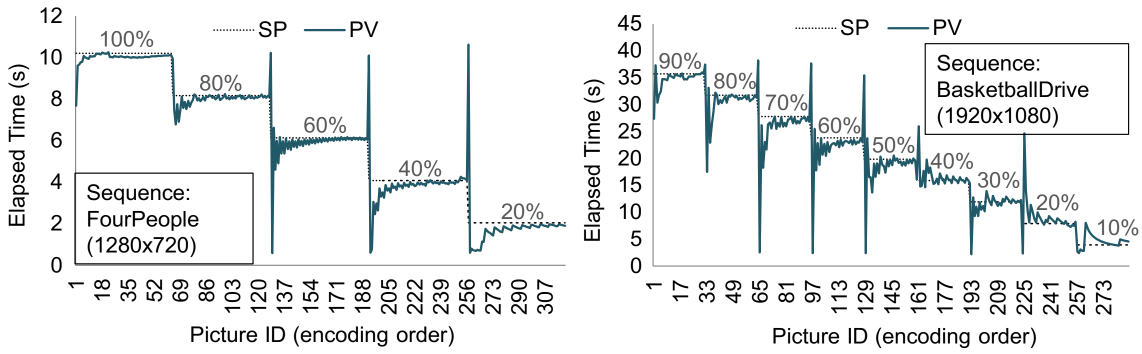

Fig. 15 Performance under dynamic set-point adjustments for sequences with different spatial resolutions

The second test simulates a situation in which the target changes over time during the video encoding runtime. This is also very applicable to practical scenarios for embedded applications, especially when the processing unit is shared among other tasks, reducing the computation share of the encoder in a given time frame. The results for two different sequences are shown in Fig. 15.

On the left side of Fig. 15, the target complexity was reduced by 20 % each 64 frames, whereas on the right, it was reduced by 10 % every 32 frames. When the set-point changes during runtime, the designed scheme resets the controller variables for new adjustments, explaining the high oscillations in each transition. Our CCS takes on average 6 video frames to stabilize around the set point. This is an insignificant period considering frame rates of at least 30 frames per second. It is possible to conclude from this analysis that the designed scheme is also capable of achieving different targets even when they change during runtime.

5.2 Comparison with the HEVC model reference

The compression performance of the designed scheme was also tested using the default HM encoder implementation. Tables 9 and 10 show the BD-BR results for a 40 % target saving. In Table 9, the results were averaged for each class, whereas Table 10 presents results for each sequence. Every sequence was encoded with two GOP structures: IPPP and IBBB.

The results demonstrate that significant encoding time savings were achieved with a modest average BD-BR increment. The best and worst results are the same for both GOP structures, indicating that reference frame distribution does not affect the performance of the controller. The results show higher increments for classes C and F. Class F contains a sequence with a sliding text box and one with the screen capture of a racing game, which probably explains the losses above the average.

The rate–distortion plots are displayed in Fig. 16 for sequences of each resolution; each plot contains the default HM (in black) and the HM with the CCS attached (in red). Each point marked in the charts represents the Y-PSNR and bitrate obtained by encoding these sequences with the four QP values listed in Table 8.

Table 9 BD-BR increment for all tested sequences—Target Savings: 40 %

GOP structure	IPPP low delay			IBBB low delay		
QP	27, 32, 37, 42			22, 27, 32, 37		
Color channel	Y (%)	U (%)	V (%)	Y (%)	U (%)	V (%)
Class A	2.68	1.72	1.47	3.40	2.85	3.07
Class B	2.10	2.45	1.74	2.32	2.10	2.00
Class C	4.59	4.80	5.22	4.39	4.36	4.27
Class D	2.58	1.99	2.78	2.76	2.07	2.27
Class E	1.28	0.76	0.31	1.76	1.26	0.77
Class F	4.58	5.13	4.72	5.38	5.22	4.63
Overall	2.97	2.81	2.71	3.33	2.98	2.83
Enc time (%)	60.1			59.6		

These results further confirm that the achieved savings were obtained at the cost of tolerable compression penalties. For the *BasketballDrive* and *Johnny* sequences, both curves coincide perfectly, proving how the control scheme herein proposed can be efficient, especially for high-resolution sequences. This is a favorable aspect for real-time applications, for it is very likely that a control mechanism will be required more often when encoding sequences with higher resolutions, as they demand an enormous amount of computations per second. Even in the worst case (*RaceHorses* and *ChinaSpeed*), the curves are very close.

5.3 Comparison to state of the art

This section compares our scheme with the most competitive results found in the literature. The work of Correa et al. [19] was selected, as it is the one that presented an analysis for several complexity targets and also for

Table 10 BD-BR increment for all tested sequences—Target Savings: 20–60 %

Sequence	20 % savings		40 % savings		60 % savings	
	BD-BR Inc. (%)	Time savings (%)	BD-BR Inc. (%)	Time savings (%)	BD-BR Inc. (%)	Time savings (%)
BlowingBubbles	0.94	20.7	4.0	40.1	9.7	61.7
BasketballPass	−0.48	19.3	1.6	39.4	3.9	61.3
RaceHorsesC	1.25	20.5	4.8	40.8	10.4	61.5
BQMall	1.03	19.5	4.0	39.4	8.7	61.4
ChinaSpeed	1.45	20.3	4.7	40.3	12.9	61.4
BasketballDrillText	1.09	20.3	6.0	40.0	9.6	61.4
Johnny	0.17	20.4	1.7	40.5	3.0	61.6
FourPeople	0.38	20.1	1.8	40.2	2.3	61.5
BasketballDrive	0.24	20.1	2.1	40.1	3.8	61.5
ParkScene	0.46	20.1	2.5	39.8	6.7	61.7
Traffic	0.72	20.2	4.4	40.2	9.4	61.7
PeopleOnStreet	0.53	20.5	2.6	40.9	5.2	61.5

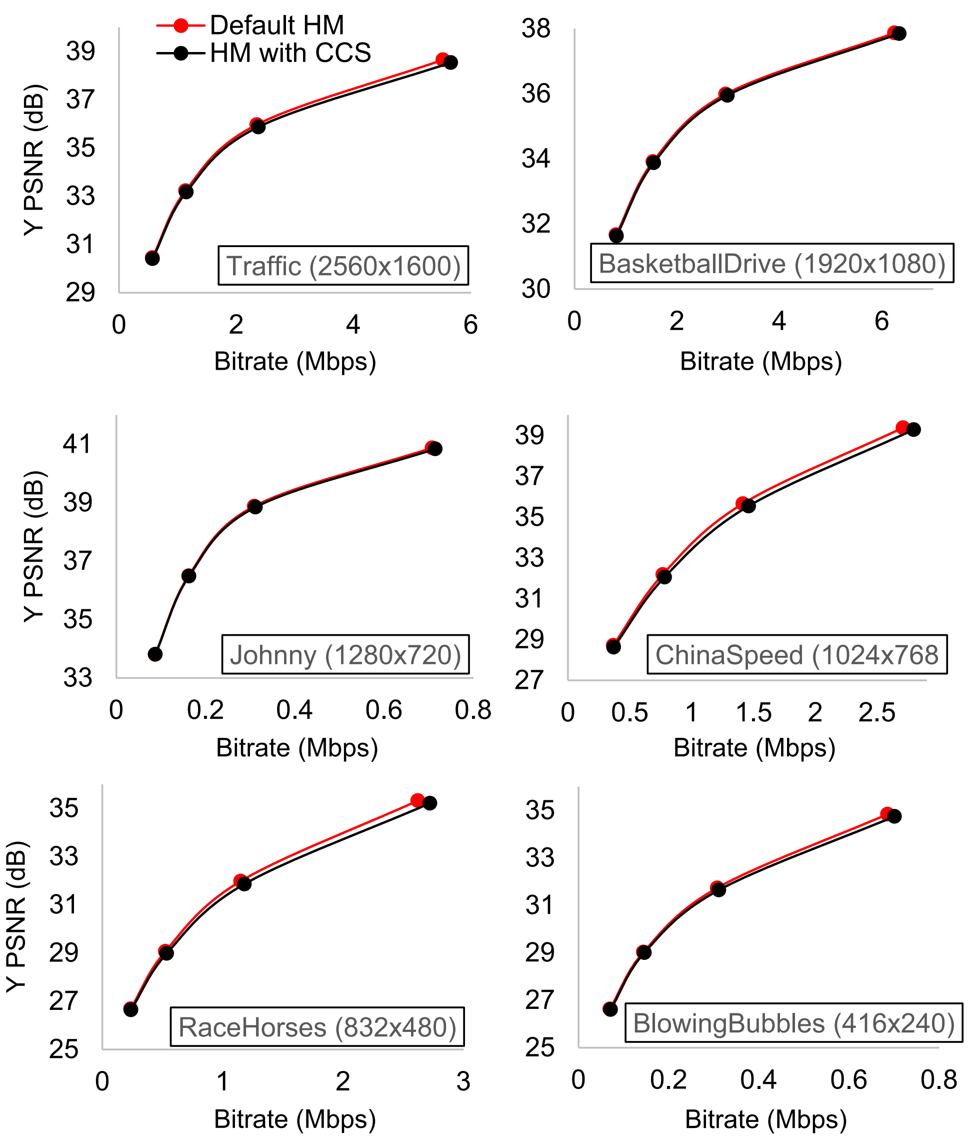

Fig. 16 Rate–distortion curves of one sequence of each class (target savings: 40 %, IPPP)

Table 11 Related work comparison (average results)—IPPP GOP structure

	[19]	This work (%)
Target: 20 %		
Effective time	17.4 %	20.1
Y BD-BR increment	0.23 %	0.45
Bitrate increment	N/A	0.21
Target: 40 %		
Effective time	39.25 %	39.88
Y BD-BR increment	0.8 %	2.97
Bitrate increment	N/A	1.84
Target: 60 %		
Effective time	57.8 %	61.45
Y BD-BR increment	3.9 %	4.9
Bitrate increment	N/A	3.2

Table 12 Related work comparison—Control system aspects

	[19]	This work
Achieved targets	20–90 %	10–90 %
Dynamic SP adjustment support	No	Yes
Maximum target error[a]	14 %	2 %
Average target error[a]	2.71 %	1.2 %

[a] Calculated for all complexity targets

achieving the best coding performance results. Table 11 shows the coding efficiency comparison for different complexity targets. Note that the BD-BR results differ from the ones previously presented, since the authors in [19] used different QP values (27, 32, 37 and 42), so new simulations were entailed to ensure a fair comparison.

Both implementations managed to reduce the encoding complexity, but our scheme yielded a better average accuracy than [19]. This is explained because the compared reference does not contain a mechanism that deals with estimation errors dynamically. In our work, this is solved with the PID controller. Also in Table 11, the compared solution achieved slightly better coding efficiency results, but it does not support interesting features such as dynamic set-point adjustment. The BD-BR increment is considerably small for all compared targets, and this is confirmed by the negligible bitrate increment. Table 12 shows a comparison of both solutions from a control system perspective.

The designed scheme is capable of achieving lower complexity targets, because the parameter sets included a really fast preset. The compared solution is able to achieve a 20 % target only for some cases. Secondly, no dynamic set-point adjustments are supported, which is also a problem when estimation errors occur. This is emphasized with the maximum error of 14 % in [19] against the 2 % generated by our scheme.

To conclude, the [19] might be more interesting for applications that prioritize image quality, but our work is more suited for systems that demand an accurate complexity control with dynamic SP adjustment support due to hard computation constraints. It is important to note, however, that the best of both solutions could be jointly implemented to produce an accurate control scheme with improved coding efficiency. In other words, if the allocation strategy of [19] is combined with the control framework implemented in this work, the resulting system is expected to present the best of both implementations: coding efficiency and dynamic control with minimum SP/PV errors (although the 90 % target savings would no longer be possible due to the limitations of [19]).

6 Conclusions

Our CCS for HEVC encoders introduced the new AC metric, which can be easily calculated during encoding for complexity estimation. A detailed assessment was first presented for encoder parameters sensitivities. The compound metric of rate–distortion–complexity efficiency defined in this work was used for key parameter adaptation.

Two budget allocation algorithms that we designed were implemented and analyzed in the control scheme, and the one with and the Priority-Aware Allocation was elected as the most efficient one. To solve estimation errors, a PID control feedback loop was also implemented. The PID parameters were tuned with the Ziegler–Nichols method, and the results proved that the scheme benefits greatly from this component. A detailed analysis over the best CCS configuration was then presented in order to test the complexity control accuracy and its adaptability to dynamic set-point adjustments.

The rate–distortion (RD) results against the HM reference show that this solution can achieve 40 % time savings with small BD-BR penalties (3.15 % Y BD-BR increase on average). When compared against related work, our CCS achieved better control accuracy, lower target complexities and dynamic support for video encoding, at the cost of small BD-BR increments and negligible bitrate increases.

As future work, more sophisticated controllers are being investigated. The budget allocation algorithms can also be widely explored, since there are many techniques that can be borrowed from operating systems' workload control literature. Finally, designing complexity-scalable algorithms for each encoding component and combining this with the current CCS is a promising branch of research as well.

References

1. CISCO: Cisco Visual Networking Index: Forecast and Methodology, [Online]. www.cisco.com/ (2012). Accessed 19 June 2012
2. ITU-T: ITU-T Recommendation H.265: High Efficiency Video Coding (2013)
3. Grois, D., Marpe, D., Mulayoff, A., Itzhaky, B., Hadar, O.: Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders. In: Picture Coding Symposium (PCS), San Jose (2013)
4. Bjontegaard, G.: Calculation of average PSNR differences between RD-curves, Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, TX, USA (2001)
5. McCann, K., Bross, B., Han, W.J., Kim, I.K., Sugimoto, K., Sullivan, G.J.: High efficiency video coding (HEVC) test model 10 (HM 10) encoder description, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 12th Meeting, Geneva (2013)
6. Vanne, J., Viitanen, M., Hamalainen, T.D., Hallapuro, A.: Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1885–1898 (2012)
7. Correa, G., Assunção, P., Agostini, L., da Silva Cruz, L.A.: Performance and computational complexity assessment of high-efficiency video encoders. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1899–1909 (2012)
8. Jianfeng, R., Kehternavaz, N.: Fast adaptive termination mode selection for H.264 scalable video coding. J. Real-Time Image Proc. **4**(1), 13–21 (2009)
9. Choi, K., Park, S.-H., Jang, E.: Coding tree pruning based CU early termination, Torino (2011)
10. Kim, J., Yang, J., Won, K., Jeon, B.: Early determination of mode decision for HEVC. In: Picture Coding Symposium (PCS), Krakow (2012)
11. Cassa, M.B. Naccari, M., Pereira, F.: Fast rate distortion optimization for the emerging HEVC standard. In: Picture Coding Symposium (PCS), Krakow (2012)
12. Shen, X., Yu, L., Chen, J.: Fast coding unit size selection for HEVC based on Bayesian decision rule. In: Picture Coding Symposium (PCS), Krakow (2012)
13. Grecos, C., Yang, M.: Fast mode prediction for the Baseline and main profiles in the H.264 video coding standard. IEEE Trans. Multimed. **8**(6), 1125–1134 (2006)
14. Sung, Y.-H., Wang, J.-C.: Fast mode decision for H.264/AVC based on rate-distortion clustering. IEEE Trans. Multimed. **14**(3), 693–702 (2012)
15. Ou, Y.-F., Ma, Z., Liu, T., Wang, Y.: Perceptual quality assessment of video considering both frame rate and quantization artifacts. IEEE Trans. Circuits Syst. Video Technol. **21**(3), 286–298 (2011)
16. Jiménez-Moreno, A., Martínez-Enríquez, E., Díaz-de-María, F.: Mode decision-based algorithm for complexity control in H.264/AVC. IEEE Trans. Multimed. **15**(5), 1094–1109 (2013)
17. Huijibers, E.A.M., Ozelebi, T., Bril, R.J.: Complexity scalable motion estimation control for H.264/AVC. 2011 IEEE International Conference on Consumer Electronics (ICCE), pp. 49–50, (2011)
18. Corrêa, G., Assunção, P., da Silva Cruz, L.A., Agostini, L.: Adaptive coding tree for complexity control of high efficiency video encoders. In: Picture Coding Symposium (PCS), Krakow (2012)
19. Correa, G., Assuncao, P., Agostini, L., da Silva Cruz, L.A.: Complexity scalability for real-time HEVC encoders. J. Real-Time Image Process. **1**(1), 1–16 (2014)
20. Zhao, T., Wang, Z., Kwong, S.: Flexible mode selection and complexity allocation in high efficiency video coding. IEEE J. Select. Top. Signal Process. **7**(6), 1135–1144 (2013)
21. Kannangara, C.S.: Complexity Management of H.264/AVC video compression. [Online]. https://openair.rgu.ac.uk/bitstream/10059/643/1/Kannangara%20PhD.pdf (2006). Accessed 2013
22. Kannangara, C.S., Richardson, I.E., Bystrom, M., Zhao, Y.: Complexity control of H.264/AVC based on mode-conditional cost probability distributions. IEEE Trans. Multimed. **11**(3), 433–442 (2009)
23. Binkert, N.: The gem5 simulator. ACM SIGARCH Computer Architecture, pp. 1–7, (2011)
24. Paoloni, G.: How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures, Intel Corporation (2010)
25. Nalluri, P., Alves, L., Navarro, A.: A novel SAD architecture for variable block size motion estimation in HEVC video coding. In: International Symposium on System on Chip (SoC) (2013)
26. Ahmed, A., Shahid, M.U., Rehman, A.: N-point DCT VLSI architecture for emerging HEVC standard. Hindawi: VLSI Design **2012**, 13 (2012)
27. Diniz, C.M., Shafique, M., Bampi, S., Henkel, J.: High-throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for HEVC. In: International Conference on Image Processing, Melbourne (2013)
28. Bossen, F.: Common test conditions and software reference configurations. Geneva (2011)
29. Astrom, K.J., Hagglund, T.: PID Controllers: Theory, Design and Tuning, 2 ed., ISA: The Instrumentation, Systems, and Automation Society (1995)
30. Grellert, M., Shafique, M., Khan, M.U.K., Agostini, L., Mattos, J.C.B., Henkel, J.: An adaptive workload management scheme for Hevc encoding. In: International Conference on Image Processing (2013)
31. Ziegler, J.G., Nichols, N.B.: Optimum settings for automatic controllers. J. Dyn. Syst. Meas. Control **115**(2B), 220–222 (1993)

Mateus Grellert received the M.Sc. degree in Computer Science from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2014. He is currently pursuing his Ph.D. title at the same university. He has been doing research in embedded systems solutions for more than 7 years. His published works include topics like reconfigurable architectures, memory-aware and energy-aware design and efficient video-coding systems. His current research interests involve efficient video-coding algorithms and architectures for video-coding systems in constrained, embedded applications.

Bruno Zatt received his B.E. and M.Sc. in Computer Engineering from the Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil, in 2006 and 2008, respectively. Mr. Zatt received his Ph.D. degree on Microelectronics from the PGMICRO (Graduate Program on Microelectronics) at the same university in 2012 with "summa cum laude" distinction. Currently, Bruno Zatt is a Professor at the Federal University of Pelotas (UFPel), Pelotas-RS, Brazil, and a member of the Group of Architectures and Integrated Circuits (GACI). He has 9+ years research experience on algorithms and hardware architectures for video processing including 3 years researching on low-power embedded realization for the Multiview Video Coding as intern researcher at the Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. His main research interests are fast algorithms and low-power hardware architectures for video processing and 2D/3D video coding for mobile applications.

Muhammad Shafique (M'11) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2011. He is currently a Research Group Leader at the Chair for Embedded Systems, KIT. He has over 10 years of research and development experience in power-/performance-efficient embedded systems in leading industrial and research organizations. He holds one US patent. His current research interests include design and architectures for embedded systems with focus on low power and reliability. Dr. Shafique was the recipient of 2015 ACM/SIGDA Outstanding New Faculty Award, six gold medals, the CODES + ISSS 2015, 2014 and 2011 Best Paper Awards, AHS 2011 Best Paper Award, DATE 2008 Best Paper Award, DAC 2014 Designer Track Poster Award, ICCAD 2010 Best Paper Nomination, several HiPEAC Paper Awards and the Best Master Thesis Award. He is the TPC co-Chair of ESTIMedia 2015 and 2016, and has served on the TPC of several IEEE/ACM conferences like ICCAD, CASES, ASPDAC, and DATE.

Sergio Bampi received the B.Sc in Electronics Engineering and the B.Sc. in Physics from the Federal University of Rio Grande do Sul (UFRGS, 1979), Brazil, and the M.Sc. and Ph.D. degrees in electrical engineering from Stanford University (USA) in 1986. He has been a Full professor in the Microelectronics field at UFRGS, where he has been a professor at the UFRGS Informatics Institute since 1986, and a Ph.D. advisor and project leader in the Microelectronics and Computer Science Programs at UFRGS since 1988. He served as Distinguished Lecturer (2009–10) of IEEE Circuits and Systems Society, worked as the technical director of the Microelectronics Center CEITEC (2005–2008) and is the past President of the FAPERGS Research Funding Foundation and of the SBMICRO Society (2002–2004). His research interests are in the area of IC design and modeling, nano-CMOS devices, mixed signal and RF CMOS design, low-power digital design, dedicated complex algorithms, architectures and ASICs for image and video processing. He has co-authored more than 300 papers in these fields and in MOS devices, digital circuits design, analog and RF circuits, technology and CAD. He was Coordinator of the Graduate Program on Microelectronics at Federal University UFRGS (2003–2007) and since 1988 has advised 44 Master's and 16 Ph.D. thesis. He is a member of IEEE, SBMICRO, SBPC and SBC scientific societies. He was the Technical Program Chair of IEEE SBCCI Symposium (1997, 2005), SBMICRO (1989, 1995), IEEE LASCAS (2013), VARI Workshop, SuperComp 94 and many other conferences.

Jörg Henkel (M'95-SM'01-F'15) is currently with Karlsruhe Institute of Technology (KIT), Germany, where he is directing the Chair for Embedded Systems CES. Previously, he was a Senior Research Staff Member at NEC Laboratories in Princeton, NJ. He received his PhD from Braunschweig University with "Summa cum Laude". Prof. Henkel has/is organizing various embedded systems and low-power ACM/IEEE conferences/symposia as General Chair and Program Chair and was a Guest Editor on these topics in various Journals like the IEEE Computer Magazine. He was Program Chair of CODES'01, RSP'02, ISLPED'06, SIPS'08, CASES'09, Estimedia'11, VLSI Design'12, ICCAD'12, PATMOS'13 and NOCS'14 and served as General Chair for CODES'02, ISLPED'09, Estimedia'12, ICCAD'13 and ESWeek'16. He is/has been a steering committee member of major conferences in the embedded systems field like at ICCAD, ESWeek, ISLPED, Codes+ISSS and CASES and is/has been an editorial board member of various journals like the IEEE TVLSI, IEEE TCAD, IEEE TMSCS, ACM TCPS, JOLPE, etc. In recent years, Prof. Henkel has given around ten keynotes at various international conferences primarily with focus on embedded systems dependability. He has given full-/half-day tutorials at leading conferences like DAC, ICCAD, DATE, etc. Prof. Henkel received the 2008 DATE Best Paper Award, the 2009 IEEE/ACM William J. McCalla ICCAD Best Paper Award, the Codes+ISSS 2015, 2014, 2011 Best Paper Awards and the MaXentric Technologies AHS 2011 Best Paper Award as well as the DATE 2013 Best IP Award and the DAC 2014 Designer Track Best Poster Award. He is the Chairman of the IEEE Computer Society, Germany Section, and was the Editor-in-Chief of the ACM Transactions on Embedded Computing Systems (ACM TECS) for two consecutive terms. He is an initiator and the coordinator of the German Research Foundation's (DFG) program on "Dependable Embedded Systems" (SPP 1500). He is the site coordinator (Karlsruhe site) of the Three-University Collaborative Research Center on "Invasive Computing" (DFGTR89). He is the Editor-in-Chief of the IEEE Design & Test Magazine since January 2016. He holds ten US patents and is a Fellow of the IEEE.