Analysis of HEVC transform throughput requirements for hardware implementations

Analysis of HEVC Transform Throughput Requirements for Hardware Implementations

Maurizio Masera[a], Lorenzo Re Fiorentin[a], Enrico Masala[b,*], Guido Masera[a], Maurizio Martina[a]

[a] *Electronics and Telecommunications Department*
Politecnico di Torino, corso Duca degli Abruzzi 24, 10129 Torino - Italy
[b] *Control and Computer Engineering Department*
Politecnico di Torino, corso Duca degli Abruzzi 24, 10129 Torino - Italy

Abstract

Hardware implementations can provide significant speedup and saving for video compression applications. In this work we focus on the transform coding sub-module analyzing the performance required by actual encoder implementations. We show that the throughput assumption about the transform sub-module in most research works is overly optimistic since it does not consider the complexity of a rate-distortion optimized video coding process. Many HEVC coding options are compared in terms of impact on quality and throughput, so to recommend the most efficient settings without excessively penalizing quality. Moreover, comparisons with the AVC standard show that, in general, HEVC presents much higher complexity to deliver its claimed compression advantages. Finally, a practical case study is shown to highlight how the proposed transform throughput analysis could be used to determine the throughput for a transform sub-module hardware design.

Keywords: HEVC, Video coding, Transform throughput requirements

*Corresponding author
Email addresses: maurizio.masera@polito.it (Maurizio Masera), lorenzo.refiorentin@polito.it (Lorenzo Re Fiorentin), enrico.masala@polito.it (Enrico Masala), guido.masera@polito.it (Guido Masera), maurizio.martina@polito.it (Maurizio Martina)

1. Introduction

The High Efficiency Video Coding (HEVC) [1] is the latest video compression standard developed by the ITU-T Video Coding Experts Group (VCEG) and by the ISO/IEC Moving Picture Experts Group (MPEG). The aim of this new block-based hybrid video coding standard is to double the coding efficiency with respect to the previous Advanced Video Coding (AVC) standard [2]. It achieves substantially better coding efficiency compared to its predecessors by means of a larger number of coding tools, such as new prediction modes, larger transform sizes and new picture partitioning, which require a much higher computational effort with respect to the previous standards [3–6]. As with previous standards, custom hardware implementations can accelerate some operations [7–10]. Such hardware is typically integrated in many devices, including cameras and mobile phones. Moreover, for real-time encoding on such types of devices, hardware implementations are strongly required [11]. Examples of hardware architectures for real-time HEVC encoding can be found in [12, 13].

Many encoding operations can benefit from hardware assistance. Among them, in this paper we focus on the transform operation which is a fundamental one for HEVC. Indeed, many custom architectures have been proposed during last years for this purpose [14–19]. Almost all the works in literature assume that each input pixel of each frame is transformed only once during the whole video coding process, as in [15–18], where the throughput is calculated resorting to the transform size and to the system clock frequency only. In particular, the folded DCT architecture in [15] achieves a constant throughput of 2.992 Gsamples/s (16 samples $\times$ 187 MHz) independently of the transform size, while the architectures in [16, 18] achieve on average 4.69 and 8.8 Gsamples/s respectively. Thus the authors claim to support 8K UHD @60 fps video encoding. In practice, however, this is not true because real encoder implementations require to compute the transform operation more than once per each pixel. This increased throughput stems from the fact that recent video coding standards, such as HEVC, exploit a large number of coding tools, thus optimal settings in rate-distortion sense can be selected only through extensive coding mode analysis.

Nevertheless, quantifying the amount of transform operations actually performed by the encoder in a realistic application is often difficult, despite being extremely important [20]. Note also that the transform operation has greater importance in HEVC than in AVC, since the Discrete Cosine Trans-

form (DCT) block sizes have been extended from 4×4 and 8×8 to 16×16 and 32×32. Additionally, a 4×4 Discrete Sine Transform (DST) operation has been included. The actual number of transform operations performed during the encoding process is primarily influenced by the coding decision tree, which typically relies on the rate-distortion (RD) optimization [21]. RD-optimized schemes can achieve good performance but might require a large number of tests to determine the combination of the parameters for the coding tools that yield the lowest Lagrangian cost function.

The issue of evaluating the complexity of an HEVC encoder can be analyzed for both software and hardware design methodologies [22]. On the software side, it has already been addressed by some works available in the literature [4–6]. All these works performed encoder profiling by measuring the overall encoding time and the processing time spent during the computation of the different operations. This analysis allows to identify the tools that most affect the coding efficiency and the computational complexity of the whole HEVC encoder. However, the encoding time can only partially catch the required information about the transform coding complexity, because it strongly depends on the most time-consuming operations of the encoder, such as the intra prediction and the motion estimation. Indeed, the transform stage takes 8.7% and 4.0% for all-intra and random-access configuration respectively, when performing software encoding of 1080p video sequence in a fully sequential way and, most notably, not in real-time, as observed in [4]. Therefore, software profiling provides only information about the computational burden of each processing stage, while it can not be used to derive actual throughput requirements for the transform submodule of hardware encoders targeting real-time processing. Indeed, when designing a completely hardware system for real-time encoding, all the processing stages are required to satisfy strict throughput constraints. This raises the issue to have a specific metric to measure the transform complexity, hence to determine the actual throughput, which is independent of the platform and of the other encoder operations. To this purpose, we use the complexity index that we initially defined in [20], which relies on counting the transform operations performed during the encoding process. Differently from the encoding time, which provides only information about the profiling and the overall encoding process, the complexity index can precisely quantify the throughput requirement needed to design a transform sub-module for real-time applications, thus also serving as an index to measure the complexity of the HEVC encoding process.

The adopted methodology is depicted in the diagram in Fig. 1. First of all, the quality-complexity analysis of the HEVC encoder has been assessed by performing coding simulations on a large dataset of video sequences. The results of this analysis are the transform complexity index (C_I) and the Bjøntegaard-Delta Bit-Rate (BDBR). These metrics are then used in the hardware implementation stage to derive the transform throughput requirement of a real-time HEVC encoder with given specifications for the hardware transform design.

Stemming from these observations, this work extends and completes our previous one [20] by providing the following major contributions. I) An in-depth investigation which assesses the impact of each coding tool on video quality and transform complexity, so that combinations of HEVC coding tools are identified to achieve the best tradeoff between quality and transform complexity. II) The application of the presented results to a practical case study in order to show how they can guide the design of a real-time hardware transform module for a typical coding configuration. Moreover, two minor contributions are also provided. I) A detailed comparison between AVC and HEVC standards focusing specifically on the comparison of the transform complexity and the corresponding rate-distortion performance in order to determine a term of comparison for complexity-constrained applications. II) A significantly larger set of video sequences used in the experiments with respect to [20], including UltraHD (UHD) content.

The paper is organized as follows. Section 2 briefly overviews the main characteristics of the AVC and HEVC standards and the optimization algorithm employed by both the AVC and HEVC encoding software used in the experiments. Section 3 addresses the issues of transform coding in actual encoder implementations and defines the complexity index. Then, the simulation setup is presented in Section 4, followed by results in Section 5: they show both the HEVC transform complexity, also comparing them to the AVC case, and an analysis of the dependency of the complexity index on the different coding tools. A practical method to easily determine the requirements for the hardware implementation of the transform sub-module is then presented in Section 6. Finally, Section 7 draws conclusions.

2. HEVC and AVC Overview

This Section briefly summarizes the main features and characteristics of the two considered standards, as well as the Lagrangian rate-distortion

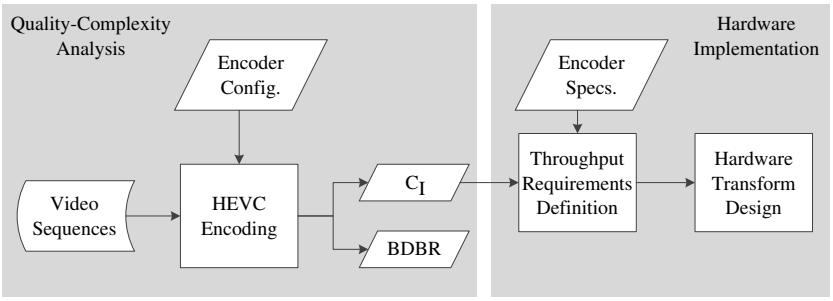

Figure 1: Diagram of the methodology.

optimization method, which is employed by the encoders to choose the most efficient set of coding modes.

2.1. Codecs

Recent video coding standards are based on block-based hybrid video coding [1, 2], i.e., they adopt motion estimation and intra prediction to exploit the video temporal and spatial correlation. Both AVC and HEVC split the image into blocks, named macroblocks and coding tree units (CTU) in the two standards, respectively. Such basic blocks can be recursively divided into smaller sub-blocks. The decision about how to split a block into sub-blocks, their coding mode (i.e., *Intra* or *Inter*), and which transform size to use is up to the encoder. In this context, the difference between AVC [2] and HEVC [1] relies on the different maximum sizes allowed for the sub-blocks, being 8×8 for AVC and 32×32 for HEVC. The minimum is 4×4 for both. Moreover, HEVC introduces the Asymmetric Motion Partitioning (AMP), which enables support for non-squared prediction units. AVC standardizes a 4×4 DCT as well as an 8×8 DCT for some profiles. HEVC is more flexible, providing DCTs from 4×4 to 32×32, as well as a 4×4 DST and a Transform Skip (TS) mode in which the transform is not applied (mostly useful for synthetic content). It is worth noting that, in HEVC block partitioning into prediction units (PU) for motion estimation and transform application are independent coding parameters. Such flexibility provides higher coding gains but also much higher complexity in coding decisions. In HEVC, transform coding is performed by the encoder on the basis of the so-called residual quad-tree (RQT) structure, which is composed of transform units (TUs) organized as a tree with the root at the coding unit (CU) level.

2.2. Rate-Distortion Optimization

In order to fully exploit AVC and HEVC capabilities, Lagrangian rate-distortion optimization [21] is necessary. Such an approach tries to choose the most efficient coding modes (in particular for prediction and partitioning) adapting to scene content. In short, the Lagrangian optimization algorithm aims to solve the following unconstrained minimization problem: $\min\{D + \lambda \cdot R\}$, where D is the distortion introduced by the coding process, λ is the Lagrange multiplier and R represents the number of bits associated to the chosen coding mode. Specifically, D can be calculated as: I) the sum of squared differences (SSD); II) the sum of absolute differences (SAD) or III) the sum of absolute Hadamard transformed differences (SATD) between the reference and the test blocks of pixels. Since the SSD cost function calculates the difference between the original and the reconstructed block, it is worth noting that it implies the computation of DCT and DST. On the other hand, the SAD and SATD cost are calculated by means of simple accumulation and Hadamard transform respectively, which acts as a rough approximation of the DCT.

To reduce the computational complexity, separate Lagrangian optimizations are performed for motion estimation and coding mode decision [3]. The main challenge for the encoding process is to decide how to perform the partitioning of the frame and which is the best coding mode of each CU. To accomplish this task, the reference HEVC encoder adopts the mode decision process, illustrated in Fig. 2. For every possible CU the SSD cost is initialized to the one obtained by performing inter prediction on a block of size $2N \times 2N$ (*Inter_2N×2N*). Then, Early Skip Detection (ESD) and Coding Flag Mode (CFM) are tested. If the motion vector difference is null or all the transform coefficients are zeros, the decision process skips all the other modes. Otherwise, it computes all the other prediction modes from *Skip* to *PCM*, while also checking the Asymmetric Motion Partitioning (AMP) modes. Finally, it tests the Early Coding Unit (ECU) condition to detect whether the *Skip* mode has been chosen as the best coding mode. If the condition is true, the process stops. Otherwise, the current CU is further divided into four CUs of half size and four new mode decision processes are created and executed. This recursion ends when the smallest CU size is found.

Each prediction mode is characterized by the proper rate-distortion optimization algorithm. In particular, each *Intra* mode selects a pre-determined number of intermediate candidates using the SATD cost. Then, it uses the SSD cost only to refine the selection of the best mode. Other methods to

lower the computational complexity of the *Intra* mode selection process are available in the literature [23]. On the other hand, each *Inter* mode exploits the SAD and SATD cost functions, for integer-pixel and sub-pixel refinements, respectively, to identify (for each PU) the best prediction candidate among available blocks (in the previously coded frames). When the mode is *Skip*, the SSD cost function is minimized to select the best motion parameters among all merge candidates.

As far as the transform coding structure is concerned, the reference encoder performs a recursive exhaustive exploration of the TU partitioning for both *Inter* mode luma/chroma components and *Intra* mode luma component. On the other hand, for *Intra* mode chroma, TU partitioning is kept equal to the one derived for the luma component. This recursive partitioning process has the root at the CU level and is executed recursively up to the RQT depth, which is specified as one of the encoder parameters. An example of transform coding structure, including different TU partitioning, is reported in Fig. 3. Moreover, Rate-Distortion Optimized Quantization (RDOQ) is employed to perform soft decision quantization for each transform coefficient. The process minimizes the Lagrangian cost function employing SSD by trying to quantize each coefficient c with values in the set $\{0, c, c+1\}$.

As shown, to determine D and R of the Lagrangian cost function several coding attempts need to be carried out, including repeated transform operations on the same original pixels when different modes are tested. In particular, the transform throughput requirement addressed in this work is directly related to the SSD cost calculation, which is employed both in the mode decision process and in the rate-distortion algorithms of the most complexity demanding tasks, shaded in gray in Fig. 2. Indeed, SSD cost calculation requires to compute the difference between the original and the reconstructed blocks by applying repeatedly the complete coding chain composed of residual calculation, transform computation, quantization and their inverse functions. To the best of our knowledge this aspect has been partially considered only in two recent works [20, 24], despite its relevance for the design requirements of any hardware implementation of the transform sub-module, especially when oriented to real-time applications.

3. Transform Coding Complexity

To evaluate the real requirements needed by the transform sub-module in an actual encoder implementation, we rely on the definition of the complexity

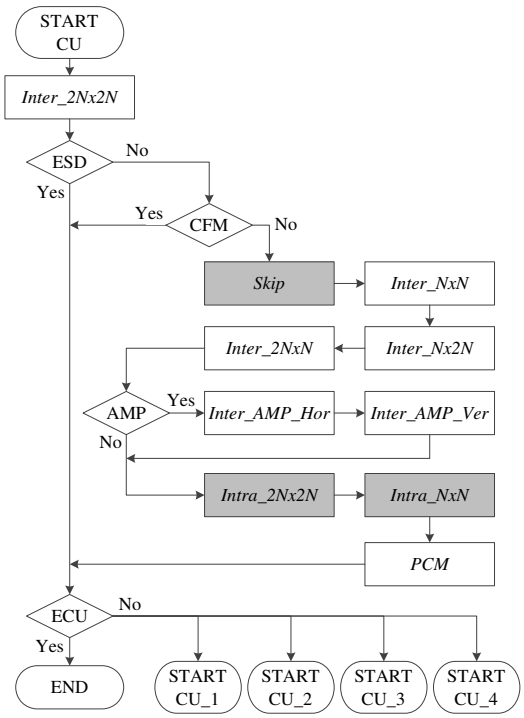

Figure 2: Flowchart of the mode decision process.

index (C_I) of a given encoding process, that we initially proposed in [20]. The approach of counting the number of transform operations performed by an HEVC encoder is similar to the one exploited in [25, 26] to measure the computational complexity of the motion compensation and deblocking stages in an HEVC decoder, which supports the scalable extension. Specifically, the C_I expresses how many times a pixel is considered for a transform operation, and it is defined as

$$C_I = \frac{T_A}{T_H} = \frac{P_T}{W \cdot H \cdot S_c \cdot N_f}, \tag{1}$$

where $T_H = W \cdot H \cdot F_s \cdot S_c$ is the hypothetical reference throughput, which is the one calculated assuming that the transform operation is performed only once for each input pixel, W and H are the width and the height of the frame respectively, F_s is the frame rate and S_c is the chrominance sub-sampling factor which can be 3, 2, 1.5, 1.5 depending on the adopted scheme (4:4:4, 4:2:2, 4:2:0, 4:1:1, respectively). The term T_A in Eq. (1) is the actual

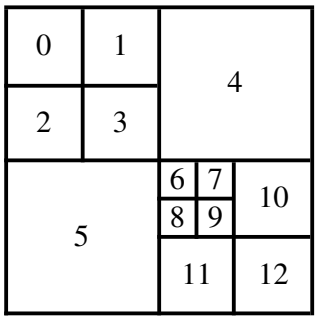

Figure 3: Example of transform structure RQT partitioning.

throughput:

$$T_A = \frac{P_T \cdot F_s}{N_f}, \tag{2}$$

where N_f is the number of frames in the video sequence and P_T is the total number of transformed samples. For HEVC, the value of P_T is obtained by counting, for each size of the transform, the actual number of transform operations performed by the encoder and adding them together weighting them with the corresponding block size:

$$P_T = \left(\sum_{i=2}^{5} \alpha_i \cdot N_{DCT2^i} \right) + \beta \cdot N_{DST4}, \tag{3}$$

where $\alpha_i = 2^{2i}$ and $\beta = 16$ express the number of pixels covered by a $2^i \times 2^i$ DCT. N_{DCT4}, N_{DCT8}, N_{DCT16}, N_{DCT32} and N_{DST4} are the transform counts. It is worth noting that this formula does not capture the hardware demands of different transform sizes, since it is focused on the throughput.

It is important to observe that the C_I definition can be easily extended to other video compression standards such as AVC. In this case, the only relevant DCT transform sizes are 4×4 and 8×8, therefore only the terms N_{DCT4} and N_{DCT8} exist.

4. Simulation Setup

To compare the coding efficiency and the complexity of the transform sub-module of HEVC and AVC, we run coding experiments referring to a generic entertainment scenario. The experiments will also attempt to assess the influence of each single coding tool on the global performance. The latest

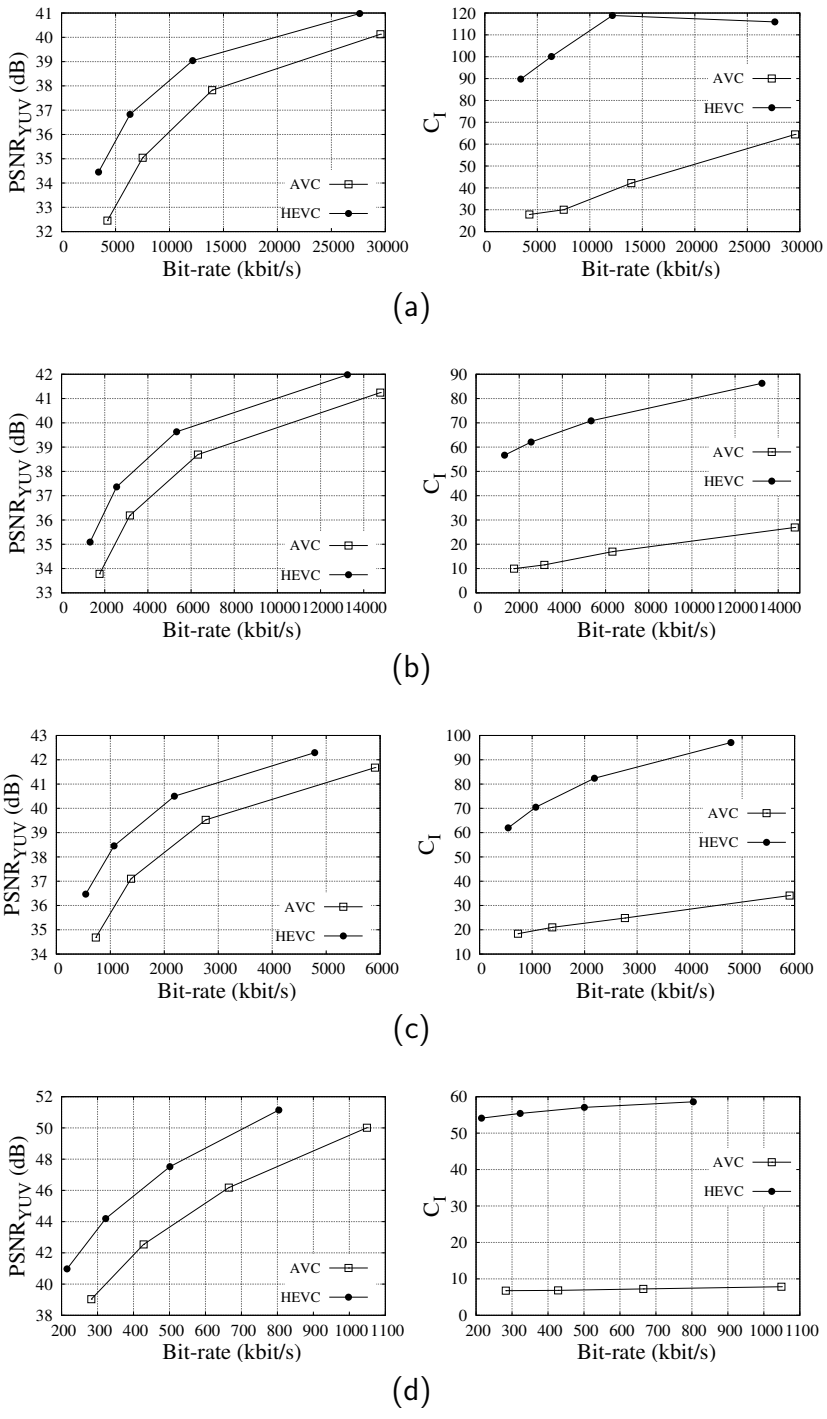

Figure 4: Rate-distortion and rate-complexity-index curves for the selected classes. (a) Wood, 3840×2160, 30 fps (b) Traffic, 2560×1600, 30 fps (c) Kimono 1920×1080, 24 fps (d) SlideShow, 1280×720, 20 fps.

Table 1: Selected Settings for the Reference Software Models.

Codec	AVC	HEVC
Software Version	JM 18.6	HM 16.3
Profile	High	Main
Level	5.1	6.2
Reference Frames	4	4
Group of Pictures	8	8
Hierarchical Encoding	On	On
Temporal Levels	4	4
Intra Period	1s	1s
Search Range	128	64
Coding Unit size / depth	N/A	64 / 4
Transform Unit size min / max	N/A	4 / 32
8x8 Transform	On	N/A
EarlySkipEnable	On	N/A
SelectiveIntraEnable	On	N/A
Rate Control	Off	Off

versions of the reference AVC and HEVC encoder software models (JM 18.6 and HM 16.3) have been used, both configured according to the so-called random-access main configuration parameters, summarized in Table 1.

High resolution sequences used in the official HEVC common test conditions (CTC) [27] have been employed for these experiments. The CTC classify sequences according to (decreasing) resolution into different classes named from A to E, plus class F which contains synthetically generated content, with mixed resolutions. Our tests include the highest resolution classes A and B, and class F. Moreover, to address the recent trend in resolution increase [28], four UHD video sequences from the SJTU dataset [29] have been added, despite they have not been originally included in the official HEVC CTC. The complete list of tested video sequence is reported in Table 2. Each video sequence has been coded using four different fixed quantization parameters (QP) for I pictures, namely 22, 27, 32 and 37, as suggested in the CTC.

To compare different measures across several bit-rates, the Bjøntegaard-Delta Bit-Rate (BDBR) measurement method [30] has been used. Therefore, the BDBR expresses the average rate variation when considering the same combined PSNR [3] (i.e., $PSNR_{YUV}$), which can be easily calculated by

Table 2: Test Sequences Used for Simulations.

Class	Resolution	Length	Sequence	Frame Rate
UHD	3840×2160	10 s	Tree Shade	30 Hz
			Wood	30 Hz
			Campfire Party	30 Hz
			Construction Field	30 Hz
A	2560×1600	5 s	Traffic	30 Hz
			People On Street	30 Hz
			Nebuta Festival	60 Hz
			Steam Locomotive	60 Hz
B	1920×1080	10 s	Kimono	24 Hz
			Park Scene	24 Hz
			Cactus	50 Hz
			BQ Terrace	60 Hz
			Basketball Drive	50 Hz
F	832×480	10 s	Basketball Drill Text	50 Hz
	1024×768	16 s	China Speed	30 Hz
	1280×720	10 s	Slide Editing	30 Hz
	1280×720	25 s	Slide Show	20 Hz

means of a weighted sum of the PSNR of each component as:

$$PSNR_{YUV} = \frac{6 \cdot PSNR_Y + PSNR_U + PSNR_V}{8}. \tag{4}$$

The transform complexity has been quantified through our proposed index C_I. Results show the percentage of variation with respect to a reference case ($C_I[\%]$) averaged over all the selected QP values. In addition, the difference between the maximum and minimum value taken by $C_I[\%]$, indicated by $\Delta C_I[\%]$, is shown as well. Therefore, the latter value represents how much $C_I[\%]$ depends on the bit-rate. All the indicators are averaged over all the considered test sequences.

5. Results

The aim of this Section is to analyze the transform complexity in HEVC by first investigating the quality-complexity difference between HEVC and

AVC and then analyzing the transform complexity contributions of each coding tool. Finally, the complexity analysis of a real-life HEVC encoder is reported as well.

5.1. HEVC Complexity vs AVC

First, quality and complexity curves are shown in Fig. 4, for both HEVC and AVC, as a function of the bit-rate, for a representative sequence belonging to each of the considered video classes. As expected, HEVC significantly outperforms AVC in terms of coding efficiency, with higher gains for the low bit-rate case (graphs in the left part). However, such performance increase is tied to higher coding complexity, which is reflected by the higher C_I value (graphs in the right part). Moreover, the much higher C_I values for the HEVC case underlines the need for an accurate estimation of the actual computational burden of the transform sub-module, that would be largely underestimated using the hypothetical reference throughput T_H defined in Section 3.

The value of the metrics introduced in Section 4 are reported in Table 3 for each class of tested video sequences. Comparisons express the AVC value using HEVC as the reference. While AVC requires about double bit-rate for equal quality with respect to HEVC for classes of natural content, the difference is lower when the synthetic content is considered. As expected, significant reductions of the transform complexity of AVC with respect to HEVC are observed, due to the lower number of coding tools. Moreover, the $\Delta C_I[\%]$ value shows that the higher is the resolution, the larger is the percentage of variation of the C_I index with respect to the HEVC case.

It is worth noting that the results of the comparison between AVC and HEVC serve as reference for the HEVC complexity analysis of the next session. Indeed, they define a term of comparison from both the rate-distortion performance and complexity point of view. In complexity-constrained real-time HEVC applications, the complexity can be reduced by limiting the usage of coding tools at the encoder side, still producing HEVC compliant bitstreams but lowering the rate-distortion performance as well. Therefore, when searching for an efficient encoding configuration for HEVC, it is important to check that the new configuration still outperforms AVC, thus guaranteeing that the use of HEVC worths the way.

Table 3: BDBR and C_I variation: AVC vs HEVC.

Class	BDBR[%]	C_I[%]	ΔC_I[%]
UHD	79.3	-65.2	18.7
A	106.7	-72.0	10.6
B	91.9	-71.6	8.5
F	54.9	-82.3	4.4

Table 4: HEVC Coding Tools Considered for Testing.

Coding tool	Tested	CTC	[31]
Largest Coding Unit (LCU) sizes	32×32, 16×16	64×64	64×64
Residual Quad-Tree (RQT) depth	2 and 1	3	1
Max Transform Unit (TU) sizes	16×16, 8×8	32×32	32×32
Transform Skip (TS)	Disabled	Enabled	Disabled
Sign Data Hiding (SDH)	Disabled	Enabled	Enabled
RDOQ	Disabled	Enabled	Enabled
Quarter-pel motion vector	Off	On	On
Half-pel motion vector	Off	On	On
Asymmetric Motion Partit. (AMP)	Disabled	Enabled	Disabled
Merge mode candidates (N_{mc})	3 and 2	5	5
Number of reference frame (N_{rf})	1	4	4
ECU, ESD and CFM	Enabled	Disabled	Enabled

5.2. HEVC Analysis

In order to analyze the dependency of transform complexity on HEVC coding tools, we adopted the same setup used in [31]. The full list of the considered coding tools is reported in Table 4, where both tested and common test conditions (CTC) are shown, as well as the conditions used in [31] for UHD content, as it will be discussed in Section 6.

Table 5 reports extensive results about the rate-distortion performance and C_I variations due to each individual coding tool with respect to the HEVC encoder configured according to the CTC. Each encoder configuration is defined by enabling only one among the tested coding options of Table 1 at a time and leaving the other parameters as specified in the CTC. Results can be interpreted as for Table 3. Therefore, a low value of ΔC_I[%] indicates that, for a given coding tool, C_I[%] has no strong dependency on the bit-rate. While the rate-distortion results on coding tools are validated by the analysis

Table 5: BDBR and C_I variation: HEVC Coding Tools.

Coding Tool	Class	BDBR[%]	C_I[%]	ΔC_I[%]	Coding Tool	Class	BDBR[%]	C_I[%]	ΔC_I[%]
	UHD	4.4	-23.5	3.5		UHD	5.9	-1.7	1.5
	A	6.1	-24.2	3.1		A	5.3	-1.8	2.2
LCU size	B	3.7	-23.8	1.3	RDOQ off	B	5.8	-1.5	1.5
32 × 32	F	2.4	-23.4	1.4		F	2.9	-1.1	0.7
	AVG	**4.1**	**-23.8**	**2.3**		**AVG**	**5.0**	**-1.5**	**1.5**
	UHD	20.9	-53.7	9.2		UHD	1.9	0.3	0.9
	A	32.2	-55.6	4.6		A	2.2	0.3	0.8
LCU size	B	19.2	-54.9	2.0	Quarter-pel off	B	2.9	0.4	0.7
16 × 16	F	10.9	-54.3	1.6		F	1.4	0.2	0.4
	AVG	**20.8**	**-54.6**	**4.3**		**AVG**	**2.1**	**0.3**	**0.7**
	UHD	0.2	-27.1	6.0		UHD	7.9	1.6	3.4
	A	0.3	-29.7	2.1		A	7.4	1.3	2.1
RQT depth 2	B	0.4	-29.8	1.0	Half pel off	B	10.2	1.4	2.2
	F	0.4	-29.7	0.9		F	4.3	0.5	0.8
	AVG	**0.3**	**-29.0**	**2.5**		**AVG**	**7.4**	**1.2**	**2.1**
	UHD	0.5	-55.0	5.3		UHD	0.3	-8.3	14.6
	A	0.8	-55.2	4.3		A	0.6	-14.6	7.6
RQT depth 1	B	1.1	-56.3	2.9	AMP off	B	0.7	-12.2	8.6
	F	1.3	-56.2	1.8		F	0.6	-9.9	3.0
	AVG	**0.9**	**-55.7**	**3.6**		**AVG**	**0.6**	**-11.2**	**8.5**
	UHD	2.6	-21.4	1.9		UHD	0.3	-3.7	11.9
	A	3.6	-21.3	2.8		A	0.4	-8.5	6.6
TU size	B	2.4	-20.8	1.2	$N_{mc} = 3$	B	0.4	-6.5	6.1
16 × 16	F	1.2	-20.9	1.8		F	0.2	-5.8	3.2
	AVG	**2.5**	**-21.1**	**1.9**		**AVG**	**0.3**	**-6.1**	**6.9**
	UHD	10.0	-47.1	20.1		UHD	0.5	-4.8	11.8
	A	11.4	-44.5	8.1		A	1.0	-12.5	10.2
TU size	B	9.0	-40.5	2.2	$N_{mc} = 2$	B	0.9	-9.6	9.3
8 × 8	F	4.2	-40.9	2.7		F	0.5	-8.6	4.8
	AVG	**8.7**	**-43.2**	**8.3**		**AVG**	**0.7**	**-8.9**	**9.0**
	UHD	-0.2	-0.1	0.2		UHD	3.0	0.7	1.0
	A	0.0	0.0	0.1		A	3.6	0.4	0.5
TS off	B	-0.1	0.0	0.1	$N_{rf} = 1$	B	3.5	0.1	0.3
	F	8.2	0.0	0.4		F	2.5	0.2	0.2
	AVG	**2.0**	**0.0**	**0.2**		**AVG**	**3.1**	**0.4**	**0.5**
	UHD	0.7	-0.1	0.1		UHD	0.3	-51.9	34.4
	A	1.0	-0.1	0.2		A	1.4	-36.7	38.0
SDH off	B	0.8	-0.1	0.1	ECU, ESD	B	1.5	-47.0	36.2
	F	0.1	0.0	0.1	and CFM on	F	1.2	-53.3	18.0
	AVG	**0.6**	**-0.1**	**0.1**		**AVG**	**1.1**	**-47.2**	**31.7**

in [3, 31], it is worth noting that the focus of this work is on the transform throughput requirements. Therefore, this Section mainly explains how the different coding tools and parameters of the rate-distortion process affect the complexity index.

First, Table 5 shows that the main C_I variations are due to parameters related to CU and TU partitioning (i.e., first six coding tools). This observation is in agreement with [20]. Indeed, both the BDBR and C_I[%] are affected by LCU size reduction, especially when high resolution video sequences are considered. This is due to the fact that 32×32 and 64×64 coding blocks can better compress large uniform regions of the frame [3, 20, 31]. The impact of the LCU size on C_I is twofold. It acts directly on the number of executions of the mode prediction decision algorithm in Fig. 2, which is called for each CU level, and also indirectly since the CU is the root of the RQT partitioning. Concerning the TU partitioning, this is regulated by the RQT depth, which defines the maximum recursion in the quad-tree partitioning and the

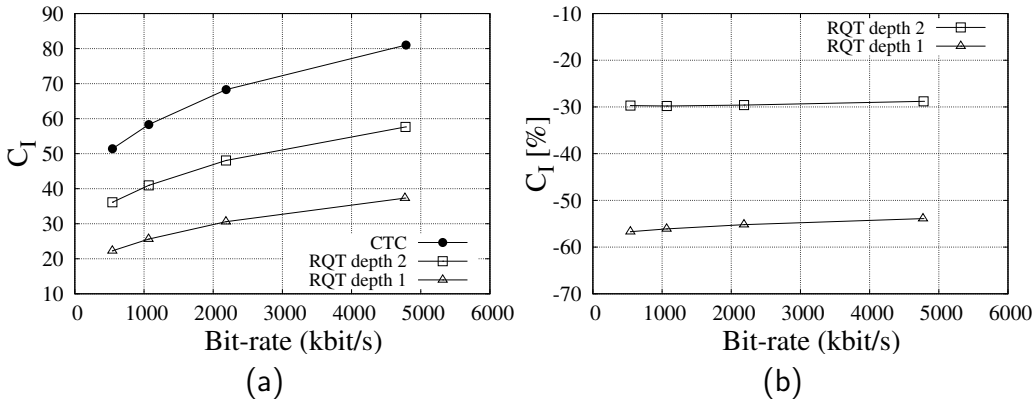

Figure 5: Curves of absolute and relative C_I variations over RQT depth for Kimono 1920×1080, 24 fps.

maximum allowed TU size. Results show that a reduction of the RQT depth produces a rate increase of about 1% while it achieves a complexity reduction up to about 56%. The absolute value and the relative variation of the C_I over the bit-rate for different RQT depths are shown in Fig. 5. As it can be observed from Fig. 5b, the variation of the C_I proves to be almost constant across different rate points, as also shown by the $\Delta C_I[\%]$ metric in Table 5. Unlike the RQT depth, the maximum TU size, which sets the largest usable transform in the TU partitioning, causes non negligible losses. Therefore, similarly to the CU partitioning, it can be noticed that, in case high resolution video sequences have to be encoded, large TUs effectively compress the residual redundancy. Since the first parameters affects the coding decision tree of the encoder and the transform partitioning (which are explored exhaustively), it is worth noting that they feature a complexity reduction which is independent of the considered class.

As far as the Transform Skip (TS) option is concerned, it can be observed that it does not lead to either significant complexity or rate increase for all the tested classes, except for synthetic content of Class F. Indeed, the usage of TS is recommended when encoding such content, for which turning off the tool causes a BDBR increase of about 8%. Moreover, also the Sign Data Hiding (SDH) and Rate Distortion Optimized Quantization (RDOQ) coding tools produce negligible variations in terms of transform coding complexity.

Concerning motion estimation, results show that several parameters have a significant role. First, sub-pixel refinements are needed in order to avoid sig-

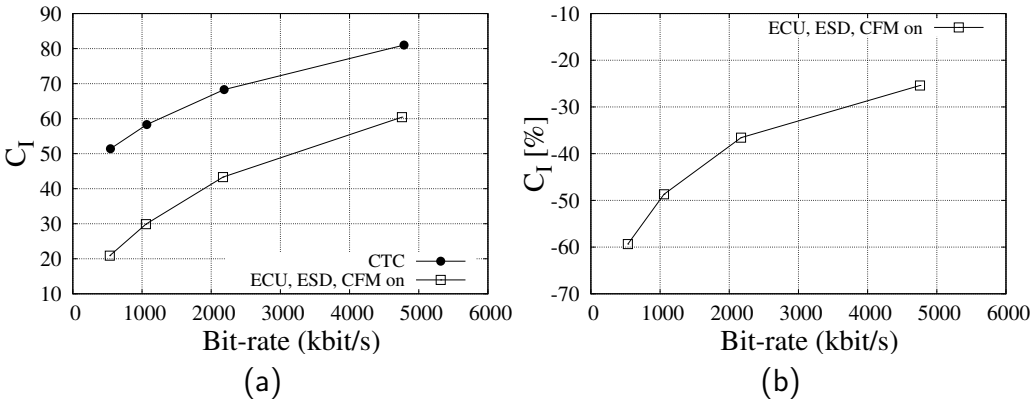

Figure 6: Curves of absolute and relative C_I variations over the enabling of ECU, ESD and CFM fast-encoding conditions for Kimono 1920×1080, 24 fps.

nificant rate increases, in particular the half-pixel motion estimation. Moreover, an average C_I reduction of about 11% can be achieved at the cost of a rate increase of only 0.6% when the Asymmetric Motion Partitioning (AMP), which allows non-square prediction units during motion estimation (see Fig. 2), is turned off. As for the AMP tool, the effect of the number of merge candidates (N_{mc}) on C_I variations depends on the content and the bit-rate, especially in the case of high resolution video sequences. By reducing N_{mc} from 5 to 3 (or to 2) it is possible to obtain a transform complexity reduction of about 6% (or 9%) with small rate increases. On the contrary, a higher rate increase is observed when limiting the number of reference frames to 1 ($N_{rf} = 1$), which affects only the rate without any effect on C_I.

Finally, the Early Coding Unit (ECU) termination, Early Skip Detection (ESD) and Coding Flag Mode (CFM) of the HM encoder have been analyzed. According to [32], ECU termination stops the CU partitioning in the intra-inter decision tree when the best coding mode for that CU is determined to be the *Skip* mode. ESD checks if the motion vector and the transform coefficients calculated for the *Inter_2N×2N* mode are all zeros, while CFM checks only if there are non-zero transform coefficients. As shown in Fig. 2, an activation of one of the two conditions forces the encoder to skip the testing of all the other coding modes for the current CU, thus greatly reducing the overall complexity. Since the number of non-zero coefficients strongly depends on the quantization step size defined by the QP value, the impact of these coding tools on the transform complexity is also dependent on the

bit-rate (see $\Delta C_I[\%]$). As shown in Fig. 6, the activation of these conditions is more likely to happen at low bit-rates, leading to larger C_I reductions with respect to high bit-rates. As summarized in the right part of the last row of Table 5, the activation of these tools provides a C_I reduction of about 47% while incurring in a 1% BDBR increase only, thus representing one of the most effective ways to reduce transform complexity.

In summary, the previous results showed that the following coding tools considerably affect the complexity index: LCU size, RQT depth, TU size, AMP and the ECU, ESD and CFM fast-encoding conditions.

5.3. x265 Analysis

In order to assess the complexity index in a real-life encoder, we employed the open-source x265 encoder [33], which has been developed by MulticoreWare by reusing most of the x264 AVC coding optimizations and adding new HEVC features. The x265 encoder provides ten predefined set of coding options (also known as presets), from Placebo to Ultrafast, which offer different trade-offs between encoding complexity and compression efficiency.

To perform the analysis, four different presets have been chosen, namely Placebo, Veryslow, Medium and Ultrafast, while the default group-of-pictures adopted by x265 has been kept unchanged. The simulations have been performed by encoding all the video sequences belonging to the four classes reported in Table 2. Two modes of encoding have been addressed. The first one is based on fixed quantization, which means that the QP has been kept constant throughout the encoding process, as specified in the CTC [27]. On the other hand, the second one adopts adaptive quantization, which is an x265 coding tool that adjusts the per-block QP depending on the complexity of the content. Specifically it lowers the number of bits used to code more complex areas, trying to neutralize the bias of the encoder, which typically uses too many bits to represent complex areas and not enough for flat areas. Since the adaptive quantization is an optimization tool which trades-off the PSNR score for perceived visual quality, it is worth noting that the related objective BDBR results are affected by the adoption of this tool.

Fig. 7 shows an example of the x265 rate-distortion and rate-complexity-index curves. As it can be observed, the Placebo and the Ultrafast curves identify a region which contains all the other preset curves. Tables 6 and 7 report the BDBR and the C_I of each tested preset with respect to the Placebo configuration, which is used as the anchor, respectively for the two encoding modes. As expected, higher performance degradation is achieved

Table 6: BDBR and C_I variation: Constant Quantization.

Preset	Class	BDBR[%]	C_I[%]	ΔC_I[%]
Veryslow	UHD	0.4	-21.3	1.5
	A	0.4	-21.2	1.2
	B	0.2	-22.0	1.5
	F	7.6	-22.6	0.9
	AVG	**2.2**	**-21.8**	**1.3**
Medium	UHD	24.4	-91.4	3.1
	A	21.5	-92.9	1.9
	B	24.3	-92.4	3.0
	F	58.2	-91.1	0.9
	AVG	**32.1**	**-92.0**	**2.2**
Ultrafast	UHD	85.6	-96.8	1.0
	A	50.8	-96.9	0.6
	B	58.2	-97.1	0.7
	F	106.0	-96.2	0.2
	AVG	**75.1**	**-96.8**	**0.7**

for larger complexity reduction. Considering natural content (see classes UHD, A and B), the Veryslow preset is the best trade-off from the coding efficiency point of view, because it allows to reach almost no rate-distortion loss with transform complexity reduction of about 21%. On the other hand, the Medium preset shows very low transform complexity ($C_I < 10$) at the cost of significant quality loss. The Ultrafast preset represents the worst-case encoding mode with very poor performance. Moreover, performance loss are larger also for synthetic content (see class F), especially when the Ultrafast preset is chosen. This is due to the fact that the x265 encoder enables the Transform Skip tool only in those presets which guarantee high compression efficiency. From Table 7 it can be observed that the adaptive quantization tool does not cause transform complexity variations, whereas it introduces additional rate-distortion penalty in terms of BDBR score, as previously explained.

6. Throughput Determination for Hardware Design

The purpose of this Section is to show how to determine the throughput requirements for the design of a real-time hardware implementation of the

Table 7: BDBR and C_I variation: Adaptive Quatization.

Preset	Class	BDBR[%]	C_I[%]	ΔC_I[%]
Veryslow	UHD	0.1	-22.2	1.3
	A	1.4	-21.9	1.3
	B	0.8	-22.7	0.7
	F	8.0	-23.2	0.9
	AVG	**2.6**	**-22.5**	**1.1**
Medium	UHD	30.1	-90.8	2.4
	A	22.7	-91.9	2.7
	B	28.6	-91.5	2.8
	F	67.1	-91.2	1.3
	AVG	**37.1**	**-91.3**	**2.3**
Ultrafast	UHD	118.4	-96.5	0.8
	A	67.7	-96.9	0.8
	B	105.6	-96.7	0.8
	F	159.8	-96.2	0.3
	AVG	**112.9**	**-96.6**	**0.7**

transform sub-module in an HEVC encoder. We propose to use a worst case method, based on the absolute value of the C_I index, considering all the overheads due to the rate-distortion optimization process for a given coding configuration. Four different configurations have been considered in the following: the random-access main configuration of the HM encoder, which comprises all the coding tools specified in the HEVC standard as shown in the column labeled CTC of Table 4, the HM encoder configured according to [31], the x265 encoder with Veryslow and Medium presets.

The following steps summarize the practical procedure to calculate the actual throughput: I) select the maximum resolution and frame rate, i.e., the worst-case scenario for these parameters; II) evaluate the C_I as a function of the bit-rate, taking into account different types of content.

For the sake of clarity, we propose a practical case study which addresses the encoding of video sequences with maximum resolution equal to 3840×2160 pixels and frame rate equal to 30 fps. We chose to configure the encoder according to the random-access main configuration, which is typical for entertainment applications. Moreover, the whole SJTU data set [29] (which includes 15 UHD sequences) has been analyzed. Such analysis provides the results shown in Fig. 8, where the maximum rate-complexity-index

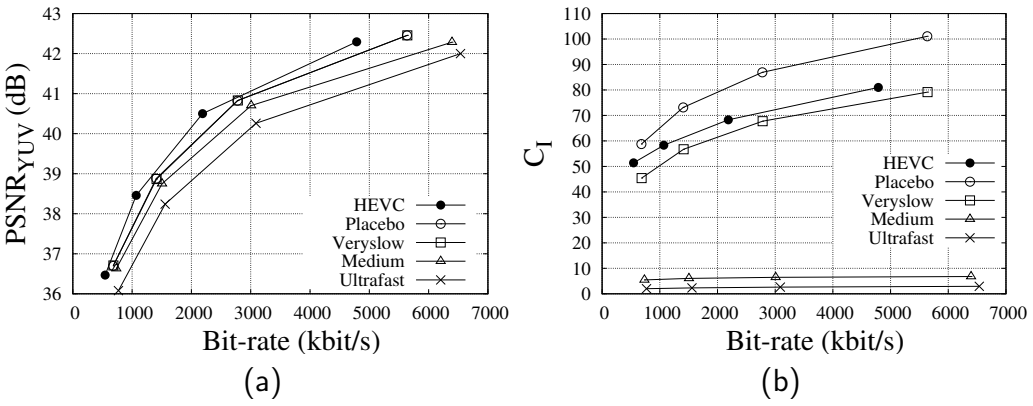

Figure 7: Rate-distortion and rate-complexity-index curves of Kimono 1920×1080, 24 fps, for different x265 presets.

curve and the corresponding rate-distortion curve of the SJTU dataset are reported. From the graph in Fig. 8b, the maximum value of C_I can be evaluated for the bit-rate of interest. In the proposed case study, we choose to work with a bit-rate of 12 Mbits/s, therefore the corresponding C_I results to be equal to 134. This value can be used to determine the actual throughput by reversing Eq. (1):

$$T_A = T_H \cdot C_I = W \cdot H \cdot S_c \cdot F_s \cdot C_I, \tag{5}$$

which yields T_A equal to 50 Gsamples/s in our case.

The same approach can also be followed when different encoder configurations are desired. For instance, using the configuration suggested in [31] for UHD video content and reported for completeness in the last column of Table 4, the corresponding rate-distortion and rate-complexity-index curves, shown in Fig. 8, are obtained. As it can be observed, the performance degradation is very low (BDBR nearly 1.28%), whereas the complexity is reduced by about 80%. This is because the considered configuration differs from the random-access main of the CTC only in those parameters which effectively reduce the complexity index without significantly impairing the quality: RQT depth, TS, AMP tools and the fast-encoding settings. Similarly to the random-access main configuration, the C_I and the actual throughput can be calculated. For the same bit-rate C_I is equal to 27, therefore Eq. 5 indicates 10 Gsamples/s as throughput requirement. Moreover, Fig. 8 also

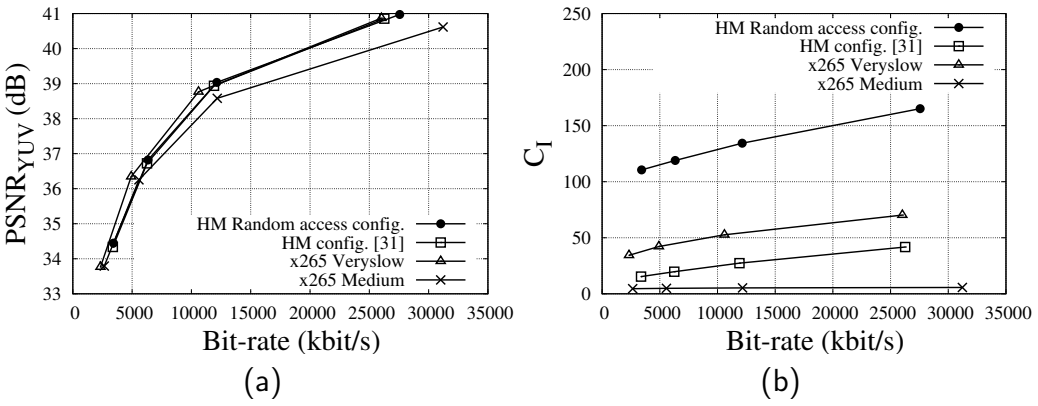

Figure 8: Rate-distortion and rate-complexity-index curves of worst-case UHD content for the HM encoder configured in the random-access main and in [31], and for the x265 encoder in the Veryslow and Medium presets.

shows the rate-distortion and rate-complexity-index curves of the x265 encoder configured with the Veryslow and the Medium presets. In these cases, at bit-rate of 12 Mbits/s the C_I is about 54 and 5 respectively, thus determining a throughput requirement of 20 Gsamples/s and 1.8 Gsamples/s respectively.

As it can be observed, both configurations have very high throughput requirements, which can only be achieved with the aid of hardware implementations. For instance, we consider the DCT hardware architecture for HEVC, which has been proposed in [18]. When working at an operating clock frequency of 400 MHz, it is able to process 32, 32, 16 and 8 samples/cycle, respectively for DCT size equal to 4, 8, 16 and 32. The resulting throughput for the two-dimensional DCT are 12.8, 12.8, 6.4 and 3.2 Gsamples/s respectively. Considering the DCT statistics in [19], the computed average throughput is equal to 12.38 Gsamples/s. Therefore, the architecture in [18] supports the real-time encoding of UHD video sequences using the configuration suggested in [31] (10 Gsamples/s) and the x265 encoder with Medium preset (1.8 Gsamples/s), while its throughput is not large enough to achieve the requirement imposed by the reference encoder in the random-access main configuration (50 Gsamples/s) and by the x265 encoder in the Veryslow preset (20 Gsamples/s).

This result suggests that hardware designers of the transform sub-module should resort to additional techniques to sustain the actual required through-

put. As an example, the throughput of small size DCTs can be increased by reusing the hardware resource required for large size transforms, to concurrently compute several small size ones, as proposed in [15]. Moreover, it is worth noting that transform operations on different TU partitioning can be computed in parallel, because there is no data dependency. This allows the designers to create a transform sub-module by instantiating many parallel hardware architectures as the ones in [15–17].

7. Conclusion

In this work a careful analysis of HEVC transform module to set the actual throughput requirement for hardware implementations, has been presented. Stemming from our recently proposed metric, named complexity index C_I, this work evaluates both HEVC and AVC transform coding complexity. This comparison points out that HEVC achieves significant bit-rate reduction at the cost of much higher transform complexity. Moreover, a tool-by-tool investigation has been conducted to quantify the complexity of the transform stage as function of different coding options. It has been shown that several coding tools among the considered ones have a significant impact on both rate-distortion performance and C_I. In particular, it is worth noting that the largest C_I variations are caused by the coding options related to the CU and TU partitioning. Finally, a practical procedure to exploit the results of this analysis for the design of real-time hardware implementations in different configurations has been proposed and applied to some case studies.

Acknowledgment

The authors would like to thank the HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it), which has provided the computational resources.

References

[1] G. J. Sullivan, J. R. Ohm, W. J. Han, T. Wiegand, Overview of the High Efficiency Video Coding (HEVC) Standard, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1649–1668. doi:10.1109/TCSVT.2012.2221191.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 560–576. doi:10.1109/TCSVT.2003.815165.

[3] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand, Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC), IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1669–1684. doi:10.1109/TCSVT.2012.2221192.

[4] F. Bossen, B. Bross, K. Suhring, D. Flynn, HEVC Complexity and Implementation Analysis, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1685–1696. doi:10.1109/TCSVT.2012.2221255.

[5] J. Vanne, M. Viitanen, T. D. Hamalainen, A. Hallapuro, Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1885–1898. doi:10.1109/TCSVT.2012.2223013.

[6] G. Correa, P. Assuncao, L. Agostini, L. A. da Silva Cruz, Performance and Computational Complexity Assessment of High-Efficiency Video Encoders, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1899–1909. doi:10.1109/TCSVT.2012.2223411.

[7] J.A. Michell, J.M. Solana, G.A. Ruiz, A high-throughput ASIC processor for 8x8 transform coding in H.264/AVC, Signal Processing: Image Communication 26 (2) (2011) 93–104. doi:10.1016/j.image.2011.01.001.

[8] K. Yoo, K. Sohn, Hardware design of motion data decoding process for H.264/AVC, Signal Processing: Image Communication 25 (3) (2010) 208–223. doi:10.1016/j.image.2009.12.001.

[9] M. Shafique, L. Bauer, J. Henkel, Optimizing the H.264/AVC Video Encoder Application Structure for Reconfigurable and Application-Specific Platforms, Journal of Signal Processing Systems 60 (2) (2010) 183–210. doi:10.1007/s11265-008-0304-5.

[10] M. U. K. Khan, J. M. Borrmann, L. Bauer, M. Shafique, J. Henkel, An H.264 Quad-FullHD Low-latency Intra Video Encoder, in: Design, Automation Test in Europe Conference Exhibition, San Jose, CA, USA, 2013, pp. 115–120. doi:10.7873/DATE.2013.037.

[11] J. Meehan, S. Busch, J. Noel, F. Noraz, Multimedia IP architecture trends in the mobile multimedia consumer device, Signal Processing: Image Communication 25 (5) (2010) 317–324. doi:10.1016/j.image.2010.04.001.

[12] K. Miyazawa, H. Sakate, S. i. Sekiguchi, N. Motoyama, Y. Sugito, K. Iguchi, A. Ichigaya, S. i. Sakaida, Real-time hardware implementation of HEVC video encoder for 1080p HD video, in: Picture Coding Symposium, 2013, pp. 225–228. doi:10.1109/PCS.2013.6737724.

[13] G. Pastuszak, A. Abramowski, Algorithm and Architecture Design of the H.265/HEVC Intra Encoder, IEEE Transactions on Circuits and Systems for Video Technology 26 (1) (2016) 210–222. doi:10.1109/TCSVT.2015.2428571.

[14] M. Budagavi, V. Sze, Unified forward+inverse transform architecture for HEVC, in: IEEE International Conference on Image Processing, 2012, pp. 209–212. doi:10.1109/ICIP.2012.6466832.

[15] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, C. Yeo, Efficient Integer DCT Architectures for HEVC, IEEE Transactions on Circuits and Systems for Video Technology 24 (1) (2014) 168–178. doi:10.1109/TCSVT.2013.2276862.

[16] J. Zhu, Z. Liu, D. Wang, Fully pipelined DCT/IDCT/Hadamard unified transform architecture for HEVC Codec, in: IEEE International Symposium on Circuits and Systems, 2013, pp. 677–680. doi:10.1109/ISCAS.2013.6571937.

[17] A. Ahmed, M. U. Shahid, A. Rehman, N Point DCT VLSI Architecture for Emerging HEVC Standard, VLSI Design 2012, Article 752024 (2012) 1–13. doi:10.1155/2012/752024.

[18] G. Pastuszak, Hardware architectures for the H.265/HEVC discrete cosine transform, IET Image Processing 9 (6) (2015) 468–477. doi:10.1049/iet-ipr.2014.0277.

[19] W. Zhao, T. Onoye, T. Song, High-performance multiplierless transform architecture for HEVC, in: IEEE International Symposium on Circuits and Systems, 2013, pp. 1668–1671. doi:10.1109/ISCAS.2013.6572184.

[20] M. Masera, L. Re Fiorentin, M. Martina, G. Masera, E. Masala, Optimizing the Transform Complexity-Quality Tradeoff for Hardware-Accelerated HEVC Video Coding, in: Conference on Design and Architectures for Signal and Image Processing, 2015, pp. 1–6. doi:10.1109/DASIP.2015.7367269.

[21] G. J. Sullivan, T. Wiegand, Rate-distortion optimization for video compression, IEEE Signal Processing Magazine 15 (6) (1998) 74–90. doi:10.1109/79.733497.

[22] J. Henkel, M. U. K. Khan, M. Shafique, Energy-efficient multimedia systems for high efficiency video coding, in: IEEE International Symposium on Circuits and Systems, 2015, pp. 613–616. doi:10.1109/ISCAS.2015.7168708.

[23] M. U. K. Khan, M. Shafique, J. Henkel, Fast hierarchical intra angular mode selection for high efficiency video coding, in: IEEE International Conference on Image Processing, 2014, pp. 3681–3685. doi:10.1109/ICIP.2014.7025747.

[24] L. Shen, Z. Zhang, X. Zhang, P. An, Z. Liu, Fast TU size decision algorithm for HEVC encoders using Bayesian theorem detection, Signal Processing: Image Communication 32 (2015) 121–128. doi:10.1016/j.image.2015.01.008.

[25] C. Feldmann, F. Jäger, M. Wien, Decoder complexity reduction for the scalable extension of HEVC, in: IEEE International Conference on Image Processing, 2014, pp. 3729–3733. doi:10.1109/ICIP.2014.7025757.

[26] M. Wien, M. Budagavi, K. Mishra, K. Ugur, X. Xiu, JCT-VC AHG report: Single-loop scalability (AHG16), Doc. JCTVC-O0016 (Jul 2013).

[27] F. Bossen, Common test conditions and Software Reference Configurations, Doc. JCTVC-J1100 (Jul. 2012).

[28] R. Weerakkody, M. Mrak, High Efficiency Video Coding for Ultra High Definition Television, in: Proc. 2013 NEM Summit, 2013, pp. 9–14.

[29] L. Song, X. Tang, W. Zhang, X. Yang, P. Xia, The SJTU 4K video sequence dataset, in: Fifth International Workshop on Quality of Multimedia Experience, 2013, pp. 34–35. doi:10.1109/QoMEX.2013.6603201.

[30] G. Bjøntegaard, Calculation of Average PSNR Differences Between RD Curves, Doc. VCEG-M33 (Apr. 2001).

[31] M. Naccari, A. Gabriellini, M. Mrak, S. G. Blasi, I. Zupancic, E. Izquierdo, HEVC coding optimisation for Ultra High Definition television services, in: Picture Coding Symposium, 2015, pp. 20–24. doi:10.1109/PCS.2015.7170039.

[32] K. McCann, C. Rosewarne, B. Bross, M. Naccari, K. Sharman, G. Sullivan, High Efficiency Video Coding (HEVC) Test Model 16 (HM16) Improved Encoder Description, Doc. JCTVC-N14970 (Oct. 2014).

[33] MulticoreWare, x265 HEVC Encoder.
URL https://bitbucket.org/multicoreware/x265/wiki/Home