Performance Analysis of Hardware-Accelerated HEVC Encoders

Sara Vitória Henssler
Universidade Federal do Pampa (UNIPAMPA)
Bagé, RS, Brasil
sarahenssler.aluno@unipampa.edu.br

Marcio Spenst
Instituto Federal Sul-rio-grandense (IFSul)
Bagé, RS, Brasil
marciospenst@ifsul.edu.br

Luciano Agostini
Universidade Federal de Pelotas (UFPel)
Pelotas, RS, Brasil
agostini@ufpel.edu.br

Marcel Corrêa
Instituto Federal Sul-rio-grandense (IFSul)
Bagé, RS, Brasil
marcelcorrea@ifsul.edu.br

Abstract

Hardware-accelerated HEVC encoding is now widely available across modern consumer devices, yet the performance and compression efficiency of these implementations vary significantly across architectures. This work presents a comprehensive evaluation of three representative hardware encoders, a desktop GPU, a laptop CPU, and a mobile SoC, together with a widely used software encoder, x265. All results are compared against the HEVC Test Model (HM) reference encoder under the Common Test Conditions, using 26 test sequences spanning Classes A1 through F. The analysis examines real-time encoding capability, bitrate efficiency through BD-BR, and the impact of resolution, frame rate, and content type on encoder behavior. The results highlight clear trade-offs between throughput and compression efficiency across architectures: desktop GPUs achieve the highest speed, mobile SoCs prioritize power-efficient operation at the cost of BD-BR, and software encoding remains competitive only at lower resolutions. These findings provide a practical benchmark for researchers and developers working on video-coding algorithms and hardware architectures, enabling direct comparison against the performance characteristics of widely deployed HEVC encoders.

Keywords

Video Coding, HEVC, Hardware Acceleration

1 Introduction

Digital video services have become integral to people's daily routines, supporting leisure, education, work, and communication. As a result, efficient video compression has become essential, since representing visual content without compression requires an impractically large amount of data. For example, an uncompressed 40-minute Ultra-High-Definition (UHD) 4K episode (3840×2160 pixels, 24 frames per second, 24 bits per pixel) would occupy roughly 1.5 terabytes—far exceeding the storage capacity of typical personal computers and requiring bandwidths well beyond what is commonly available to consumers.

Video codecs address this challenge by reducing the amount of data required for storage and transmission. A codec (encoder/decoder) performs video compression and decompression and may be implemented in software or hardware. Different codecs employ a variety of algorithms to exploit spatial and temporal redundancies. However, in fast or low-power encoding scenarios, whether in software or hardware, many of these algorithms must be simplified or omitted, leading to reduced compression efficiency compared to encoders that implement a more complete feature set.

High Efficiency Video Coding (HEVC), also known as H.265 [1], was jointly developed by the Video Coding Experts Group (VCEG) and the Moving Picture Experts Group (MPEG). Introduced in 2013 as the successor to H.264/AVC [2], HEVC provides up to a 50% reduction in bit rate for equivalent perceptual quality [1]. Although more recent standards, such as Versatile Video Coding (VVC) [3] and AOMedia Video 1 (AV1) [4], surpass HEVC in compression performance, HEVC remains highly relevant due to its widespread hardware acceleration support across modern consumer devices, including televisions, smartphones, laptops, and gaming consoles. In contrast, hardware support for VVC and AV1 is still limited, as the development and deployment of new hardware encoders is a lengthy process. Currently, only a small number of high-end general-purpose and graphics processors offer hardware acceleration for these newer codecs. Therefore, even in the presence of more modern codecs, HEVC maintains significant relevance in the industry.

An analysis of the release history of devices supporting hardware accelerated HEVC encoding at 4K resolution shows that the first graphics processing unit (GPU) to provide this capability was the Nvidia GTX 950, based on the Maxwell 2nd-generation microarchitecture, introduced in 2014 [5]. For general-purpose processors (CPU), initial support appeared with Intel 6th-generation Core processors (Skylake microarchitecture) in 2015 [6]. In the mobile domain, the Qualcomm Snapdragon 650 was the first System-on-Chip (SoC) to offer HEVC hardware encoding, also released in 2015 [7]. Notably, even before hardware-accelerated solutions became commercially available, the x265 encoder [8], a fast, open-source software implementation—had already been widely adopted since 2013 by users seeking higher compression efficiency than H.264/AVC.

This study evaluates the current state of HEVC hardware acceleration in modern devices. The analysis includes a CPU, a GPU, and a mobile SoC. For comparison, a fast software encoder, commonly used when hardware acceleration is unavailable, was also included. The performance of these encoders is compared directly against the HEVC reference encoder.

The results presented here are valuable for researchers developing algorithms and architectures aimed at improving video encoding efficiency. They provide insight into whether the quality vs bit-rate trade-offs introduced by new techniques align with the capabilities and constraints of devices currently available on the market.

This paper is organized as follows: Section 2 provides background on video coding and an overview of HEVC features. Section 3 describes the experimental setup. Section 4 presents and discusses the results. Finally, Section 5 concludes the paper.

2 Video Coding Background

Most modern video encoders rely on the following signal- and data-processing operations: (i) block partitioning, (ii) intra- and inter-frame prediction, (iii) transforms (T module), (iv) quantization (Q module), and (v) entropy coding, as illustrated in Figure 1. In addition, encoders include a reconstruction path (decoding loop), shown by the dashed line in Figure 1, which contains the inverse quantization (IQ module) and inverse transforms (IT module). This design ensures that the encoder depends exclusively on reference frames that are also available to the decoder, allowing both to perform identical predictions. An optional in-loop filtering module may also be included at the end of the reconstruction path to improve the subjective visual quality of the reconstructed frames.

Block partitioning consists of dividing the frame into smaller units that can be processed independently for compression purposes. This strategy increases adaptability to the spatial and temporal characteristics of the content, enabling more efficient coding. In HEVC, block partitioning is based on Coding Tree Units (CTUs) of up to 64×64 pixels, which can be recursively split into Coding Units (CUs) as small as 8×8 using a quadtree structure, providing fine-grained adaptation to local image characteristics [9].

After partitioning, the encoder performs prediction for each block using available information. In intra-frame prediction, blocks are predicted from already-coded samples within the same frame, exploiting spatial correlations. In inter-frame prediction, blocks from previously coded reference frames are used, exploiting temporal redundancy across neighboring frames. In both cases, the prediction result is subtracted from the original block, producing a residual signal. The effectiveness of prediction techniques directly affects the magnitude of the residual and, consequently, the overall compression efficiency [10]. In HEVC, intra prediction supports 35 directional modes, while inter prediction uses motion vectors with quarter-pixel precision and advanced motion compensation tools, significantly improving prediction accuracy [9].

The residual generated after prediction is subjected to a transform, whose purpose is to concentrate the signal energy into a reduced number of low-frequency coefficients, facilitating subsequent compression. In modern standards, this stage may involve an additional partitioning step, in which residual blocks are divided into smaller transform blocks that better match local content characteristics [10]. While older standards rely solely on the Discrete Cosine Transform (DCT), more recent standards allow a broader set of transforms [11]. In HEVC, transform blocks range from 4×4 to 32×32 pixels, and both DCT-II and DST-VII (for 4×4 intra blocks) are supported, improving energy compaction across diverse content types [11].

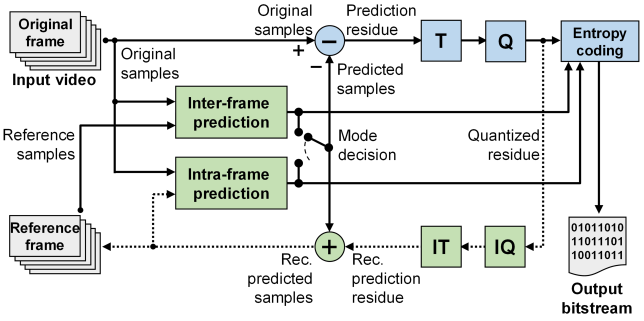

Figure 1: Diagram of a typical hybrid block-based video encoder.

Next, the transform coefficients are quantized, reducing their precision to eliminate perceptually less relevant information. The degree of irreversible loss introduced in this stage is controlled by the Quantization Parameter (QP): higher QP values introduce more loss and yield higher compression, and vice versa [11]. This step defines the concept of lossy compression [10]. In HEVC, quantization is applied independently to each transform block, and the QP can vary across blocks through the use of QP deltas, enabling fine-grained rate control.

Finally, the quantized coefficients, along with the coding parameters—such as prediction modes, motion vectors, block sizes, and other syntax elements—are passed to the entropy-coding stage. This stage applies lossless compression algorithms that reduce statistical redundancy by assigning shorter representations to more frequent symbols. The final result is a compressed bitstream structured according to the video-coding standard, which can then be stored or transmitted for later decoding and playback [10]. In HEVC, entropy coding is performed exclusively using Context-Adaptive Binary Arithmetic Coding (CABAC), which provides substantial gains in compression efficiency compared to previous standards [9].

Beyond the stages described above, modern encoders employ control mechanisms based on Rate–Distortion Optimization (RDO) [12] to select, among multiple possible coding options, the one that offers the best trade-off between bit rate and distortion. In this process, decisions such as prediction modes, block sizes, motion vectors, partition structures, and transform choices are evaluated using a cost function that combines distortion and the number of bits required to signal each alternative. Extensive use of RDO is one of the main factors behind the high coding efficiency observed in reference implementations, at the cost of significantly increased computational complexity due to the exploration of a large combinatorial decision space.

3 Experimental Methodology

In this study, encoding experiments were conducted following the Common Test Conditions (CTC) [13]. Among other guidelines, the document defines a standard set of test videos and coding parameters to be used with the HEVC Test Model (HM) reference software [14]. It also recommends encoding each video four times with the HM, using the Quantization Parameter (QP) set {22, 27, 32, 37}, which represents the operating points. These guidelines are essential for ensuring comparability and reproducibility of results across different studies.

The videos used in this work cover Classes A through F of the CTCs, totaling 26 test sequences with diverse resolutions and characteristics. Class A1 and A2 videos contain ultra-high-resolution

content. Class B videos have 1080p resolution (1920×1080 pixels), Class C videos are 480p (832×480 pixels), and Class D videos are 240p (416×240 pixels). Class E sequences have 720p resolution (1280×720 pixels) and are characterized by low motion and near-static content. Finally, Class F includes videos with various resolutions designed for Screen Content Coding (SCC), containing typical screen-based elements such as text, graphics, and user interfaces [13].

The performance of each encoder was measured in terms of encoding time, bitrate, and distortion, the latter quantified using the Peak Signal-to-Noise Ratio (PSNR) metric.

The bitrate is a measure that indicates the amount of data transmitted or processed per second in a video file. It is typically expressed in kilobits per second (kbps). A high bitrate may suggest better visual fidelity, but not necessarily, as it can also result from inefficient compression. Likewise, a low bitrate does not automatically imply poor quality, as it may reflect a more efficient compression process.

Distortion in a video refers to the differences introduced between the original signal and the compressed signal, resulting in loss of visual quality or fidelity. These differences may appear as blurring, blocking artifacts, noise, or loss of fine details. To quantify this error, objective metrics that compare the reconstructed video to the reference video are commonly used. One of the most widely adopted metrics is the Peak Signal-to-Noise Ratio (PSNR), which is based on the computation of the Mean Squared Error (MSE) between the two signals. The MSE measures the average squared error between corresponding pixels, while the PSNR expresses this relationship in decibels (dB), providing an indication of perceived quality: higher PSNR values indicate lower distortion and greater similarity to the original.

When bitrate and PSNR results are obtained for at least four operating points of an encoder, these points can be plotted on a rate–distortion (RD) curve. These points allow visualization of the bitrate used and the distortion introduced at each operating point. However, comparing two encoders is not straightforward when considering only a single metric.

This difficulty motivates the use of more robust comparative metrics capable of evaluating the overall performance of encoders across the entire rate–distortion curve. In this context, measures such as Bjøntegaard Delta Bitrate (BD-BR) are commonly employed [15]. The central idea is that the operating points obtained for each encoder (bitrate and PSNR values under different configurations) are used to generate interpolated rate–distortion curves.

Once the RD curves are constructed, the average difference between them is computed over their common operating interval. For BD-BR, the logarithmic difference in bitrate is integrated as a function of PSNR, resulting in the average percentage change in bitrate required for one encoder to achieve the same quality as the other [16]. The final metric corresponds to the mean value of this integrated area over the considered interval. Figure 2 illustrates this procedure for computing BD-BR.

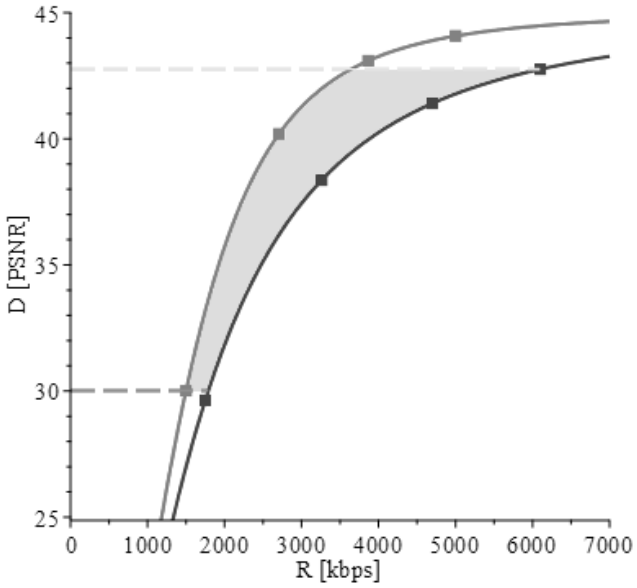

Figure 2: Computation of BD-BR. The dashed lines indicate the range of integration along distortion (PSNR), and the painted region shows the area of the integral. Source: [16].

3.1 Software Experiments

The first stage of this work consisted of generating the encoding results using HM v18.0 [14]. The HM, maintained by the VCEG and MPEG groups, serves as the reference implementation to ensure full compliance with the HEVC standard. Because the reference software is not designed for real-time operation, encoding even a few seconds of video may require several hours or days to complete. For this reason, the results of the fast encoders are compared against the HM outputs to quantify the impact of real-time encoding in terms of BD-BR.

Subsequently, similar experiments were performed using the x265 [8] encoder through its libx265 library integrated into FFmpeg v6.1.1 [17]. It is important to note that x265 provides ten speed presets, each modifying a wide range of encoding parameters to accelerate processing at the cost of compression efficiency. For example, the fastest preset significantly restricts the block-partitioning tree (maximum block size = 32, minimum = 16), whereas the slowest preset enables the full set of block sizes defined by the standard [8]. Therefore, for each test sequence, the slowest preset capable of achieving real-time encoding was selected to ensure the highest possible quality.

All experiments were executed on a server equipped with an Intel Xeon Silver 4314 processor operating at 2.4 GHz.

3.2 Hardware Experiments

Encoding experiments were conducted using the following devices:

- CPU Intel Core i7-11800H (2.30 GHz) from a Dell G15 5511 laptop;
- GPU Nvidia GeForce RTX 4070 from a custom desktop computer;
- SoC Qualcomm Snapdragon 8 Gen 3 from a Samsung Galaxy S24 Ultra smartphone.

The corresponding hardware-accelerated encoders were accessed through FFmpeg using the qsv, nvenc, and mediacodec libraries,

respectively. An x86-64 build of FFmpeg v6.1.1 was used for the laptop and desktop experiments, while an aarch64 build was used for the smartphone.

Similar to libx265, all these libraries provide multiple presets that balance encoding speed and compression efficiency. For each device, the preset was chosen to maintain real-time encoding while maximizing output quality.

4 Results

Table 1 presents the encoding-speed results for the HM reference software and the evaluated fast encoders, where a speed of 1.00× corresponds to real-time performance. The table also reports the BD-BR values of each fast encoder relative to the HM. Table 2 summarizes the same metrics as per-class averages.

As expected, the HM reference encoder exhibits extremely low encoding speeds, reflecting its exhaustive RDO [12] process and lack of real-time constraints. For instance, the Tango sequence, an HDR 10-bit UHD 4K video at 60 fps with a duration of four seconds, required approximately 75 hours to encode using the HM.

All fast encoders, including the software-based x265, achieved real-time performance for all sequences up to 1080p resolution (Classes B and below). However, for UHD 4K content, x265 was unable to encode any sequence in real time. The Intel and Qualcomm hardware encoders achieved real-time performance for only two out of six UHD 4K sequences, while the Nvidia encoder failed to do so only for the Drums sequence, which is the most computationally demanding test case due to its 100 fps frame rate. When considering the per-class averages in Table 2, the Nvidia encoder achieved real-time performance across all classes, which is consistent with the substantially higher thermal design power (TDP) and parallel processing capabilities of desktop-class GPUs.

Figure 3 illustrates the average BD-BR results summarized in Table 2. The Qualcomm encoder exhibits significantly higher BD-BR values than the other implementations, indicating a larger loss in compression efficiency relative to the HM reference. This behavior is consistent with the strict power and thermal constraints of mobile SoC platforms, where encoding architectures are designed to prioritize energy efficiency and sustained real-time operation over exhaustive exploration of coding options. From this perspective, the observed rate-distortion trade-off reflects a deliberate design choice rather than a limitation of the encoding hardware.

Another notable observation is that the Intel hardware encoder consistently outperforms the Nvidia encoder in terms of BD-BR across most resolution classes, suggesting superior compression efficiency under comparable real-time constraints. This difference may be attributed to architectural and implementation choices, such as differences in motion-estimation strategies, block-partitioning flexibility, or internal RDO heuristics, although the exact mechanisms are not exposed by vendor-specific hardware encoders.

The x265 software encoder demonstrates improved rate-distortion efficiency as resolution decreases. This trend is explained by the extensive configurability provided by its preset system, which allows the encoder to progressively enable more complex coding tools, such as deeper block-partitioning structures and more thorough RDO, while still maintaining real-time performance at lower resolutions. In contrast, hardware-based encoders expose limited configurability, resulting in largely fixed behavior that cannot be tailored to different resolutions or content characteristics with the same granularity.

It is important to note that internal rate-control mechanisms and QP mapping strategies vary across implementations. Hardware encoders, in particular, employ proprietary rate-control algorithms that may introduce additional variability in rate-distortion behavior that is not fully captured by QP-based comparisons alone.

For Class F sequences, which consist of screen content featuring sharp edges, text, and synthetic graphics, some fast encoders exhibit atypical behavior, including occasional negative BD-BR values relative to the HM reference. These results should not be interpreted as general superiority over the reference encoder, but rather as sequence-specific effects. Screen content often presents simpler prediction and residual characteristics, and differences in the implementation or optimization of Screen Content Coding (SCC) tools may lead to favorable outcomes for certain encoders. Consequently, these results are highly content-dependent and do not necessarily generalize to natural video sequences.

Finally, it should be noted that the BD-BR values reported in this work should be interpreted with appropriate caution. The fast encoders operate under strict real-time constraints and therefore achieve substantially lower rate-distortion performance than the HM reference encoder, which limits the overlap of their rate-distortion curves along the PSNR axis. As a result, the BD-BR metric may be less robust in some cases, and the reported values should be understood as indicative of relative trends rather than exact measurements of compression efficiency.

5 Conclusion

This work presented a comprehensive evaluation of hardware-accelerated HEVC encoders across desktop, laptop, and mobile platforms, complemented by a widely used software encoder for scenarios in which hardware acceleration is unavailable. By comparing all results against the HM reference software under the HEVC CTC [13], the study quantified the trade-offs between real-time performance and compression efficiency across diverse architectures.

The results highlight clear distinctions in encoder behavior. Desktop-class GPUs consistently delivered the highest throughput

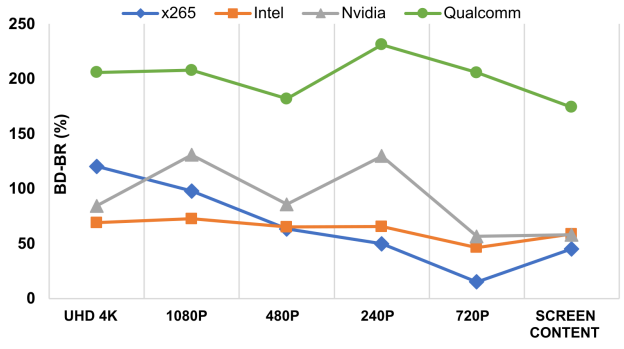

Figure 3: BD-BR comparison among fast HEVC encoders.

Table 1: Execution speed of HM reference software and rate-distortion and speed results for evaluated encoders.

Class	Sequence	HM Speed	x265		Intel		Nvidia		Qualcomm	
			BD-BR (%)	Speed	BD-BR (%)	Speed	BD-BR (%)	Speed	BD-BR (%)	Speed
A1	*Tango*	0.000148×	118.6	0.55×	80.5	0.79×	90.0	2.70×	208.5	0.88×
	Drums	0.000127×	215.2	0.30×	70.9	0.43×	98.6	0.65×	232.1	0.42×
	Campfire	0.000393×	92.1	0.95×	73.5	1.51×	47.1	1.70×	125.5	1.20×
A2	*CatRobot*	0.000239×	98.8	0.50×	67.2	0.81×	98.1	2.20×	192.7	0.86×
	TrafficFlow	0.000447×	98.8	0.95×	60.4	1.32×	80.1	2.30×	287.9	1.20×
	DayLightRoad	0.000648×	99.4	0.50×	62.1	0.73×	92.6	1.15×	189.7	0.58×
B	*Kimono*	0.002042×	57.7	1.30×	100.0	2.83×	100.0	6.25×	100.0	5.68×
	ParkScene	0.002646×	51.7	1.35×	52.7	2.73×	116.4	6.25×	201.5	5.70×
	Cactus	0.001172×	86.5	1.65×	54.4	2.45×	102.6	3.10×	194.1	2.85×
	BasketballDrive	0.000890×	100.4	1.60×	76.5	2.28×	115.3	3.10×	193.0	2.86×
	BQTerrace	0.001143×	192.6	1.95×	80.2	2.07×	220.3	2.65×	350.9	2.37×
C	*BasketballDrill*	0.005556×	43.6	1.50×	62.2	3.39×	10.4	14.30×	163.3	5.89×
	BQMall	0.005051×	74.7	1.60×	70.1	2.99×	120.5	12.50×	198.4	4.81×
	PartyScene	0.005910×	54.4	1.50×	64.7	2.78×	133.1	14.30×	229.8	5.62×
	RaceHorsesC	0.006460×	81.1	1.75×	64.2	4.44×	79.4	20.00×	136.5	6.30×
D	*BasketballPass*	0.021368×	55.4	1.70×	61.8	5.81×	93.9	33.35×	139.4	14.50×
	BQSquare	0.025253×	52.8	1.70×	59.7	5.25×	189.2	16.70×	406.7	13.67×
	BlowingBubbles	0.002796×	47.8	1.80×	71.6	6.10×	162.4	12.50×	233.4	15.08×
	RaceHorses	0.003075×	43.8	1.80×	69.9	8.63×	74.2	10.00×	145.3	17.77×
E	*FourPeople*	0.003053×	6.3	1.35×	33.6	2.66×	41.4	3.60×	145.0	4.41×
	Jhonny	0.003388×	19.0	1.60×	50.5	3.03×	66.3	2.20×	230.0	4.28×
	KristenAndSara	0.002271×	20.3	1.50×	55.7	2.92×	62.4	1.55×	242.2	4.32×
F	*BasketballDrillText*	0.005669×	47.8	1.50×	61.0	3.38×	111.6	1.40×	192.6	5.32×
	ChinaSpeed	0.003735×	78.6	1.50×	45.9	3.40×	47.1	8.35×	309.6	8.71×
	SlideEditing	0.007310×	-31.9	1.50×	-3.8	5.46×	-53.0	1.25×	95.2	7.82×
	SlideShow	0.008469×	86.9	1.72×	132.3	7.70×	127.1	6.70×	100.0	12.30×

Table 2: Per-class average execution speed of HM and rate–distortion and speed results for the evaluated encoders.

Class	HM Speed	x265		Intel		Nvidia		Qualcomm	
		BD-BR (%)	Speed	BD-BR (%)	Speed	BD-BR (%)	Speed	BD-BR (%)	Speed
A1+A2	0.000333×	120.4	0.62×	69.1	0.93×	84.4	1.78×	206.0	0.85×
B	0.001578×	97.9	1.57×	72.7	2.47×	130.9	4.27×	207.9	3.90×
C	0.005744×	63.4	1.58×	65.3	3.40×	85.8	15.27×	181.9	5.65×
D	0.013123×	49.9	1.75×	65.7	6.44×	129.9	18.13×	231.2	15.25×
E	0.008712×	15.2	1.48×	46.6	2.87×	56.7	2.45×	205.7	4.00×
F	0.006295×	45.3	1.55×	58.8	4.99×	58.2	4.42×	174.3	8.54×

and were the only hardware platform capable of achieving real-time performance across all resolution classes on average. The Intel hardware encoder demonstrated superior rate-distortion performance under real-time constraints, while the Qualcomm mobile SoC achieved real-time operation at the cost of higher BD-BR, reflecting design choices driven by strict power and thermal limitations. The x265 software encoder remained competitive at lower resolutions due to its flexible preset system, but was unable to sustain real-time performance for UHD 4K content.

Overall, these findings provide a practical and reproducible reference point for researchers and developers working on video-coding algorithms and hardware architectures. By characterizing the performance of widely deployed HEVC encoders under standardized experimental conditions, the results enable meaningful contextualization of new algorithmic or architectural proposals targeting real-time video encoding. As future work, we plan to investigate equivalent architectures across desktop and mobile platforms. For example, comparing desktop CPUs with their laptop counterparts, to further analyze how power and thermal constraints influence encoding quality and performance.

Acknowledgments

This research was funded by the *Instituto Federal Sul-rio-grandense* (IFSul) and the *Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul* (FAPERGS) – Grant number 25/2551-0000903-0.

References

[1] ITU-T. High Efficiency Video Coding. https://www.itu.int/rec/t-rec-h.265, Jul. 2024. Recommendation ITU-T H.265 (V10) (07/2024).

[2] ITU-T. Advanced video coding for generic audiovisual services. https://www.itu.int/rec/t-rec-h.264, Aug. 2024. Recommendation ITU-T H.264 (V15) (08/2024).

[3] ITU-T. Versatile Video Coding. https://www.itu.int/rec/t-rec-h.266, Sep. 2023. Recommendation ITU-T H.266 (V3) (09/2023).

[4] AOMedia. AV1 Bitstream & Decoding Process Specification (v1.0.0-errata1). https://github.com/AOMediaCodec/av1-spec/releases/tag/v1.0.0-errata1, Jan. 2019.

[5] Nvidia. Video encode and decode gpu support matrix. https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new, 2025. [Online].

[6] FFmpeg. Hardware quicksync. https://trac.ffmpeg.org/wiki/Hardware/QuickSync, 2025. [Online].

[7] Qualcomm. Qualcomm snapdragon 650 processor. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/snapdragon_product_brief_650_102816.pdf, 2025. [Online].

[8] MulticoreWare Inc. libx265. https://bitbucket.org/multicoreware/x265_git, 2024. [Online].

[9] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. doi: 10.1109/TCSVT.2012.2221191.

[10] I.E. Richardson. *Coding Video: A Practical Guide to HEVC and Beyond*. Wiley, 2024. ISBN 9781118711781.

[11] V. Sze, M. Budagavi, and G.J. Sullivan. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Integrated Circuits and Systems. Springer International Publishing, 2014. ISBN 9783319068954.

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

[13] K. Sharman and K. Sühring. Common test conditions for hm video coding experiments. JCT-VC document JCTVC-AC1100, January 2018. URL http://phenix.int-evry.fr/jct/doc_end_user/documents/29_Macau/wg11/JCTVC-AC1100-v1.zip. [Online].

[14] JVET. Hm, 2025. URL https://vcgit.hhi.fraunhofer.de/jvet/HM. [Online].

[15] G. Bjøntegaard. Calculation of average psnr differences between rd-curves. VCEG document VCEG-M33, March 2001. URL https://www.itu.int/wftp3/av-arch/video-site/0104_Aus/VCEG-M33.doc. [Online].

[16] Nabajeet Barman, Maria Martini, and Yuriy Reznik. Bjøntegaard delta (bd): A tutorial overview of the metric, evolution, challenges, and recommendations. *arXiv*, 01 2024. doi: 10.13140/RG.2.2.18622.05444.

[17] FFmpeg. Ffmpeg. https://ffmpeg.org/, 2024. [Online].