Study of EVC and VVC encoders and compute of their carbon footprint in local environment

TREBALL FI DE GRAU DE

Roger Fernández Hermoso

Director: Javier Brines Garcia

Grau en Enginyeria en Sistemes Audiovisuals

Curs 2022-2023

Universitat Pompeu Fabra Barcelona
Escola d'Enginyeria

A la meva família, per estar i confiar sempre. Sobretot quan ni jo ho feia.
I als nois del grup, per ser el meu punt de suport.

Acknowledgements

First of all, I would like to thank my tutor Javier Brines, for the help provided all the way, and for everything I have learned from him. But above all, for the closeness and values as a teacher and as a person.

From Pompeu Fabra University, I want to express my gratitude to the institution and the professors for the education received throughout the career.

To David Vila Garcia, Technical Consultant of Tractament Mediambiental Integral (TMI). For his help and knowledge contribution in the area of Carbon footprint calculation.

Finally, to the Greening of Streaming organization, specially to Dom Robinson for the meeting, providing knowledge and information about the sector.

Abstract

In this paper a study of EVC and VVC encoders and the calculation of their local carbon footprint is carried out. The structure of the paper is divided into seven chapters, starting with an introduction. Subsequently, the state of the art of the two codecs is defined, situating them with respect to their predecessors and the current state of the art in the industry. The third chapter defines what the carbon footprint is, and the tool and procedure used to calculate it. In the fourth chapter we find the description of the procedure. The whole process of collecting the results, implementing the codecs and the values obtained is explained here. In the fifth and sixth chapters, the results of the carbon footprints and the final calculation steps are presented together with the conclusions drawn from the work. Finally, the seventh chapter explains how all the information has been collected in a GitHub repository so that any reader can make their own calculations.

Resum

En aquest treball es realitza un estudi sobre els codificadors EVC i VVC, i el càlcul de la seva empremta de carboni de forma local. L'estructura del treball es divideix en set capítols, la qual comença per una introducció. Posteriorment, es defineix l'estat de l'art dels dos còdecs, situant-los respecte els seus predecessors i en el moment en la indústria. En el tercer capítol es defineix què és l'empremta de carboni i quina és l'eina i el procediment que s'ha seguit per a poder calcular-la. En el quart capítol trobarem la descripció del procediment. Aquí està explicat tot el procés de recapta de resultats, d'implementació dels còdecs i dels valors obtinguts. En el cinquè i sisè capítol son presentats els resultats de les empremtes de carboni i els passos finals per calcular-la i les conclusions que haurem extret del treball. Finalment, en el setè capítol s'explica com s'ha recollit tota la informació en un repositori de GitHub per a qualsevol lector pugui fer els seus propis càlculs.

Resumen

En este Trabajo se realiza un estudio sobre los codificadores EVC y VVC, y el cálculo de su huella de Carbono de forma local. La estructura del trabajo se divide en siete capítulos, la cual empieza por una introducción. Posteriormente, se define el estado de arte de los dos códecs, situándolos respecto a sus predecesores y en el momento en la industria. En el tercer capítulo se define que es la huella de carbono y cuál es la herramienta y el procedimiento que se ha seguido para poder calcularla. En el cuarto capitulo encontramos la descripción del procedimiento. Aquí se explica todo el proceso de recopilación de resultados, de implementación de los códecs y de los valores obtenidos. En el quinto y sexto capitulo se presentan los resultados de las huellas de carbono y los pasos finales para calcular juntamente con las conclusiones que se han obtenido del trabajo. Finalmente, en el séptimo capitulo se explica cómo se ha recogido toda la información en un repositorio de GitHub para que cualquier lector pueda hacer sus propios cálculos.

Summary

List of figures

List of tables

Chapter 1

INTRODUCTION

Nowadays, we are living in a society where mobiles, laptops or any other electronical devices have a huge relevance in our daily lives, such that we could define them as an extension of ourselves. All these technologies that process, store and let us use any information or audiovisual resource that completes our life, work through digital systems. Every simple order or video being processed on our computers are all translated orders that our device reads as zeros and ones [1].
Nevertheless, one of the most challenging and important characteristics in digital systems, is the volume of the information. Transmitting big files has always been a problem related with digital devices, but it is even more challenging nowadays since it is increasingly reaching to a higher resolution, demanding more audiovisual exigencies and faster data transmission, where we expect the highest velocity. That is where codecs magic happens.

The most basic and intuitive definition of a codec would be: a compression technology conformed with two components, an encoder to compress the files and a decoder to decompress [1], where video compression is the process of reducing the total number of bits needed to represent a given image or video sequence [2], and decompress, the backward process.
The present study emerges through the studies of Audiovisual Systems Engineering degree, more specifically, in the course of the subject Audio and Video Coding Systems, where we had a first approach to video codecs.

The aim of this project is not only to analyse in more detail the video codecs sector, but to try to provide new and useful information. That is the reason why the main objective of this project is to compute in a local environment the carbon footprint of each one of the three latest and more relevant video codecs in the sector available for free. Since I didn't have access to the LCEVC encoder, which does not have an open-source version, the study has been limited to only EVC and VVC encoders.

The carbon footprint has been in use for several decades, but it still has not a strict-standard definition. As we will see throughout the project, one of the most common interpretations is to be "a measure of the exclusive total amount of carbon dioxide emissions that is directly and indirectly caused by an activity or is accumulated over the life stages of a product" [3, Thomas and Jan 2007, p. 5], even though the most complete and used version would be the amount of total greenhouse gasses (GHGs) emitted, not only CO_2.
The carbon footprint is a good indicator of the levels of consumption and environmental contamination. Therefore, it is an innovative and newsworthy research area to focus on, not only for its technological characteristics, but also to try to provide valid data to contribute to chip in a little to protect the earth.

In the next chapter, it is described the state of art of the selected codecs, accompanied by a contextualisation with their predecessors and their corresponding advantages that position them as some of the best codecs nowadays. Then, in the third chapter it is introduced the carbon footprint concept and the methodology to obtain results. In the fourth chapter it is exposed the entire process and data results. Thereupon, the sixth chapter analyses the results and obtains the carbon footprint of each respective codec. Finally, the seventh chapter concludes with the conclusions generated throughout the project.

CHAPTER 2

STATE OF ART

Following the definition of codec in the introduction, below it is presented the state of art of them, defining the two codecs that we will implement and analyse through the study, comparing them with their predecessors.

2.1 Video Codecs

Video codecs have a massive impact in today's technology. Codecs reduce the bandwidth in transmission fluxes. That means that they are used in streaming, video shootings, and editing, among others. In other words, they are used everywhere.

But a codec structure is defined by two indispensable tools: the encoder and the decoder. Having this in consideration, encoders have a more sophisticated structure and are more complex compared to the encoder. The reason to create codecs following this procedure is that the most common usage of them is to have an encoder for the user who is sending a video encoded, and the recipients of the video, to have the decoders. So, computationally is essential to act in accordance with this idea. That is the reason this study is only focused on the encoders of these codecs' standards.

Here we will see a brief overview through some of the most important video codec standards over their history.

1. H.261: Defined by the International Telecommunication Union (ITU-T) in 1988 [4], H.261 is the first standard of the widely used family of h.26x video codecs. It was designed to be used in video telephony and video conferencing applications [5]. It aimed at bit rates around x64 kbits/s.
2. MPEG-1: Introduced in 1993 by the Moving Picture Experts Group (MPEG), this standard brought advances over H.261. The main usage goal of this codec was to be implemented with CD and offered better video quality at lower bit rates. MPEG-1 introduced most of the coding tools which would continue to be used in MPEG-2 and MPEG-4. It also meant a crucial tool for the first steps of internet video streaming [6].
3. MPEG-2: Released in 1995, MPEG-2 was a continuation of the MPEG-1 by "adding interlace capability as well as a greatly expanded range of picture sizes and bit rates" [6, John Watkinson 2001, p. 21]. It settled a precedent in codecs history by allowing high-quality video storage and transmission. It was implemented in DVD and digital television broadcasting.
4. H.263: Developed by ITU-T SG15 in 1996, this codec was designed to succeed h.261 with reducing bit rates with a significantly better video quality, reducing bit rates below

64 kbits/s. Its main implementations were for video conferencing and for video streaming [4],[7].

5. MPEG-4 Part 2: Released in 1999, MPEG-4 Part 2 introduced new coding tools to both his predecessors. The MPEG-4 standard had better compression techniques and object motion compensation which allowed better performance in bit rates with the same quality. It was used for online streaming and video compression for portable devices [6].
6. H.264 (AVC): Released in 2003, H.264 (also known as Advanced Video Coding or AVC) implemented a 50% reduction in bit rate compared to MPEG-2. It introduced HD video streaming, Blu-ray discs, and high-definition television (HDTV) broadcasting [8].
7. VC-1: This video encoder standard corresponds to WMV-x video compression technologies of Microsoft, being equal to WMV-9. They have been one of the most popular codecs since being implemented in Microsoft Operating Systems. It was released in 2006 and aimed to provide similar compression to AVC. VC-1 was used for video streaming and Blu-ray discs [9].
8. VP8 and VP9: VP8 was developed by On2 Technologies, which later would be acquired by Google in 2009. VP8 was released in 2010 and VP9 in 2013. They are royalty-free codecs provided by Google with the goal of replacing the h.264 standard. Their main implementation is on the YouTube platform, but also in general internet streaming, video display or video conferencing [10].
9. H.265 (HEVC): Jointly developed by ITU-T and MPEG organization, was approved in 2013 by ITU-T as H.265 and MPEG-H Part by ISO/IEC. The main goal of this standard was to significantly reduce the video compression, which allowed it to implement it for higher resolutions such as 4k and 8k [11].
10. AV1: Released in 2018 by the Alliance for Open Media, AV1 is an open and royalty-free video codec which reduced in a 30% the bit rate of its predecessor VP9. It is mainly being used in streaming services and online video platforms [12].

2.2 EVC

The Essential Video Coding (EVC) is a video compression standard developed as Part 1 of the MPEG-5 project to "meet the requirements of significantly improving compression efficiency over existing video coding standards with a timely publication of licensing terms. The normative standard was finalized by the ISO/IEC Moving Picture Experts Group (MPEG) in April 2020" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2]. MPEG is a prestigious group of experts that was formed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) to establish standards for media coding [14].
The main goal of the EVC is to provide a significantly improved compression capability over existing video coding standards, minimizing complexity and with timely publication of licensing terms.

One of the most important factors to develop this codec and with the following structure is that even though "there is a constant demand for more efficient video coding technologies" [13,

Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2], coding efficiency is not the only aspect to have in consideration. One of the main tenets in all industries is to cover real-world problems and to fusion technological and business perspectives.
To achieve this vision/objective/goal, the EVC standard is divided into two different profiles: "a royalty-friendly Baseline profile and a royalty bearing Main profile, which includes a small number of improved coding tools on top of the Baseline profile" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2].

Since Baseline profile aims for a royalty free codec, "it uses conventional technologies, which consist of traditional methods from the early 1980s to the end of the 1990s" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2].
The Main profile instead, aims to add those implementations to make a better and improved coding algorithm. It has a small number of tools with the only purpose of compression efficiency improvement which are capable of being activated or deactivated individually.

In the upcoming sections it is described the most important characteristics of the EVC codec. For a more detailed overview, the reader is referred to [13].

In the white paper specification [13] we can find a descriptive block diagram of the structure of the EVC coding tools [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 3]:

> EVC employs a hybrid block-based video coding approach. Each colour component of the input picture is partitioned into a group of non-overlapped blocks. Largest allowed block in each colour component is called Coding Tree Block (CTB), combination of luma CTB and associated chroma component's CTBs comprise Coding Tree Unit (CTU). A group of CTUs covering a rectangular region of the coded picture can be further grouped into a slice and/or a tile. CTUs are coded in a raster scan order utilizing a set of defined coding tools.
> Figure 1 shows a tool-level summary of EVC, which is illustrated in a block diagram of tools associated with its profile. All tools in the Baseline profile are based on technologies assessed as available for over 20 years. The tool set of the Main profile, which is added on top of the basic set of the Baseline profile, was carefully chosen by considering an interaction between the tools to enhance a performance with a limited number of coding tools.[1]

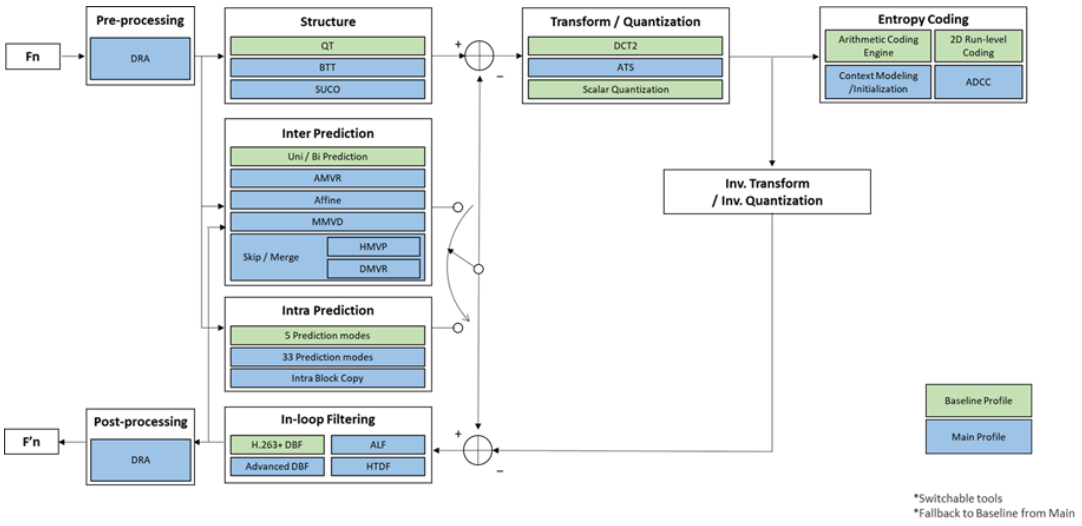

Figure 1. Block diagram of EVC coding tools

2.2.1 Baseline profile

2.2.1.1 Coding structure

Coding structure is one of the most important and correlated tools to improve compression performance. As it is explained in EVC manual [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 4]:

> Historically, new video codecs have always been proposed with new coding structures. A 16x16 sample macroblock structure was employed until the AVC standard [8]. Since a coding structure based on 16x16 sample blocks gave poor coding performance in the case of large resolutions such as 1080p or 4K, the HEVC standard and subsequent video coding standards employed larger block structures such as 64x64 with a flexible splitting mechanism to divide a large coding block into small blocks.
> The Baseline profile codes input picture with a partitioning to CTUs with a maximal size of 64x64 luma samples. Group of CTU can form a slice each of which can be coded independently.

2.2.1.2 Intra Prediction

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 4]:

> Intra prediction is a technology to exploit spatial correlation between neighbouring pixels. In h.263 and MPEG-4 visual, prediction is performed in random domain, while AVC introduced prediction "in pixel domain by referencing the neighbouring pixels of the previously coded blocks [6][7].
> In HEVC, by increasing the number of prediction directions a higher compression performance was achieved, in EVC Baseline profile 5 prediction modes are used: a so-called DC mode, where predicted samples are calculated as the arithmetic mean of the reference samples, and 4 directional prediction modes --– horizontal (H), vertical (V), diagonal left (DL), diagonal right (DR).

2.2.1.3 Inter Prediction

The Inter Prediction term is used in video compression as a kind of frame prediction where, using redundancy between consecutive frames, it is possible to reduce the bandwidth. "The EVC Baseline profile also allows a bi-directional prediction, which is a linear combination of two motion compensated blocks that involve two motion vectors" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 4].

2.2.1.4 Transform and quantization

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 4, 5]:

> As has been used in numerous codecs for video compression, the EVC Baseline profile also adopts a discrete cosine transform (DCT), which is well-known as having good energy compaction efficiency. After the transform is conducted, scalar quantization is applied to the transformed coefficients. The quantization parameter (QP) is taken from a range of 0.51 and a scaling factor (SF) corresponding to each QP is defined by a look-up table.

2.2.1.5 Loop filter

The loop filter reason to be implemented between blocks is to remove blocking artifacts that appear at applying per-block analysis processes. “In the EVC Baseline profile, a loop filter based on H.263 Annex J [6] was employed to increase objective and subjective image quality” [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 5].

2.2.1.6 Entropy Coding

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 5]:

> The state-of-the-art entropy coding scheme is context-based adaptive binary arithmetic coding (CABAC), introduced in the AVC standard. Instead of CABAC, the binary arithmetic coding scheme from JPEG Annex D is applied as the entropy coding engine of the EVC Baseline profile.

2.2.1.7 High Level syntax

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 5]:

> An EVC bitstream consists of network abstract layer (NAL) units with a 2-byte NAL unit header which contains properties of a NAL unit such as the type of data and temporal ID. A sequence parameter set (SPS) contains parameters that apply to the whole coded video sequence (CVS), a picture parameter set (PPS) contains parameters that apply to one or more pictures of a CVS, and an adaptation parameter set (APS) contains parameters of a coding tool which apply to one or more pictures of a CVS.

2.2.2 Main profile

2.2.2.1 Coding structure

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 6]:

> Main profile of EVC coding input picture with a partitioning into CTUs with a maximal size of 128x128 luma samples. Group of CTU can form a tile each of which can be coded independently. A group of tiles can be further grouped into a slice.
>
> The EVC Main profile supports a flexible block partitioning structure, which is based on the binary and ternary trees mixture scheme (BTT) with the Split Unit Coding Order (SUCO) method, for efficient and flexible representation of video content with various resolutions. BTT has coding unit (CU) shapes described by the ratio between the width and height of a block as shown in Figure 2. For instance, if the width and height of a block are the same it can be represented as a 1:1 ratio CU or a square CU, and if the width is equal to 64 samples and the height is equal to 16 samples it can be represented as a 1:4 ratio CU. CU partitioning is conducted based on the allowed CU shapes and their allowed maximum and minimum sizes.

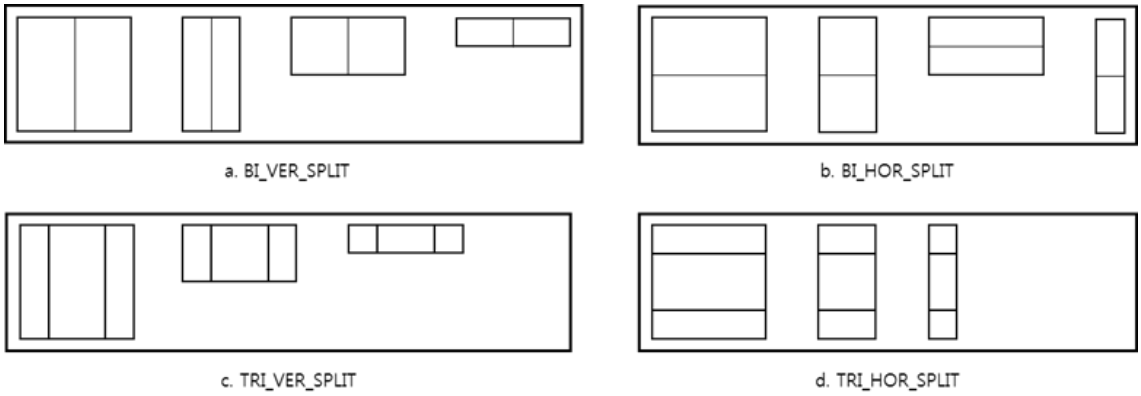

Figure 2: Binary/Ternary split modes.

As shown in Figure 3, the SUCO method enables a more flexible coding order, such as left to right (L2R) and right to left (R2L) orders, to allow intra prediction from right reference pixels and inter prediction with right motion vector predictors. If a split unit (SU) is partitioned vertically (vertical splitting), a flag is signalled to indicate L2R or R2L coding order of partitioned units. If an SU is partitioned by a quadtree structure, a flag is shared for the two above units and two bottom units. If no flag is signalled for the coding order of an SU, the coding order follows its parent's SU coding order.

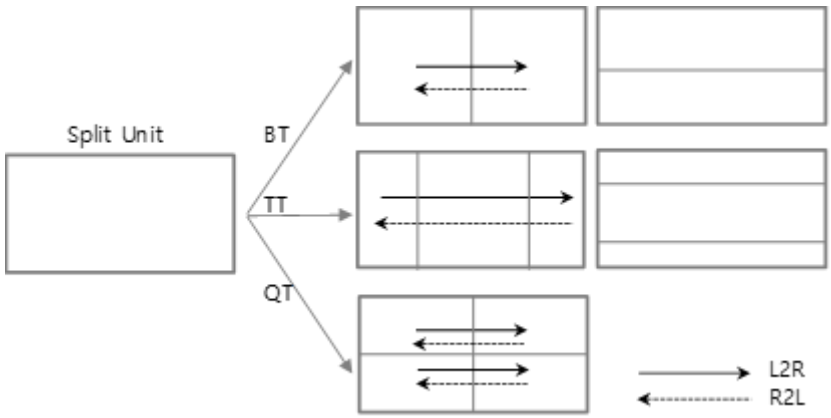

Figure 3: Allowed coding order in Quad-tree/Binary/Ternary split modes.

2.2.2.2 Intra Prediction

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 7]:

> To exploit spatial correlation efficiently based on flexible coding structure, a total of 33 intra prediction modes for luma component and 5 modes for chroma component are applied. DC, Bi-linear, Plane and Direct Mode (DM) modes are similar to that of the AVS2 standard [10], with a straightforward extension for a flexible block size while Plane and angular prediction mode are different from that in AVS2.

2.2.2.3 Merge with motion vector difference

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 7]:

> Merge with motion vector difference (MMVD) approach provides a new motion vector (MV) expression method with simplified signalling. Similar to the skip and merge modes in HEVC, MMVD makes a candidate list from neighbouring motion information, but MMVD can cover more extended motions that are not limited to the neighbouring motions. In order to construct more precise candidates, a starting point, a motion magnitude and a motion direction are utilized.

2.2.2.4 Adaptive motion vector resolution

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 7]:

> Adaptive motion vector resolution (AMVR) supports multiple motion vector resolutions. Information about the motion vector resolution is signalled at the CU level. Depending on the resolution of CU, both motion vector (MV) and motion vector predictor (MVP) of the CU are adjusted accordingly.

2.2.2.5 Affine mode

Extracted from the EVC white paper, "Affine prediction of the EVC Main profile allows the use three different affine motion modes, namely four and six parameters model modes and an affine merge mode" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 7].

2.2.2.6 Decoder-side motion vector refinement

Extracted from the EVC white paper, "the Decoder-side motion vector refinement (DMVR) method operates with the two motion vectors of the bi-prediction which are further refined by a bilateral matching process" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 7, 8].

2.2.2.7 History-based motion vector prediction

Extracted from the EVC white paper, "the History-based motion vector prediction (HMVP) method is an inter-coding tool which can be applied to both merge candidate list and motion vector prediction process" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 8].

2.2.2.8 Advanced deblocking filter

Extracted from the EVC white paper, "a deblocking filter based on the design utilized in ISO/IEC 14496-10 [8] is employed to increase objective and subjective image quality for the EVC Main profile" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 8].

2.2.2.9 Adaptative loop filter

Extracted from the EVC white paper, “to suppress compression artifacts and improve the visual and objective quality of the decoded and reference pictures, decoded samples are filtered with an adaptive loop filter (ALF)” [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 8].

2.2.2.10 Hadamard transform domain filter

Extracted from the EVC white paper, “in addition to the deblocking filter and ALF, Hadamard transform domain filter aims to reduce ringing artifacts caused by quantization of residual coefficients is introduced in EVC” [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 9].

2.2.2.11 Adaptive transform selection

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 9]:

> Adaptive transform selection (ATS) is exploited in the EVC Main profile. In addition to traditional for video compression DCT-II transform cores, EVC additionally introduces DST-VII and DCT-VIII transform cores that can be applied for intra and inter predicted residuals.

2.2.2.12 Advanced coefficient coding

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 9]:

> Transform coefficients of the coded block (residual data) after quantization are scanned in a predefined scan pattern and entropy coded. To employ statistical properties of transform coefficients, advanced coefficient coding (ADCC) utilizes the bit-plane-like coding approach.

2.2.2.13 Dynamic Range adjustment

Extracted from the EVC white paper, “to allow efficient coding of various types of video content, e.g., video in HDR/WCG formats, EVC utilizes Dynamic Range Adjustment (DRA) at the normative post-processing stage” [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 9,10].

2.2.2.14 High level syntax

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 10]:

> Reference picture management in the EVC Main profile is implemented by a modern RPL (reference picture lists) concept. It assumes explicit signalling of two reference picture lists, list 0 and list 1 that determine the reference picture management process for a specific picture in the Group of pictures (GOP) structure.

2.2.3 Performance

In the white paper [13], "a formal subjective verification test of EVC Main and Baseline profiles for standard dynamic range (SDR) content and high dynamic range (HDR) and wide colour gamut (WCG) content, respectively" [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 10]. In the document we can find the figures and the performance tests that they performed. In this project we will only analyse the numerical results.

2.2.3.1 SDR

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 10]:

> The MOS BD-rate calculation [14] shows that the average bit rate saving for the EVC Main profile compared to the HEVC Main10 profile was approximately 39% using UHD SDR content encoded in random access configuration, and approximately 41% using HD SDR content encoded in low delay configuration. The average bit rate saving for the EVC Baseline profile compared to the AVC High10 profile was approximately 39% using UHD SDR content encoded in random access configuration, and approximately 34% using HD SDR content encoded in low delay configuration.

2.2.3.2 HDR/WCG

Extracted from the EVC white paper [13, Convenor of ISO/IEC JTC 1/SC 29/AG 03 2021, p. 13]:

> Subjective testing in HDR category was conducted through a subjective evaluation, comparing coding performance of the EVC Main profile to the HEVC Main 10 profile for video sequences in BT.2100/PQ representation [15]. Figure 7 shows the visual testing results in HDR/WCG category. The average bit rate savings for the EVC Main profile (etm6.0) with HDR-UHD and HDR-HD are estimated at approximately 36% and 35% respectively compared to the HEVC Main10 profile (hm16.20).

2.3 VVenC

The Versatile Video Coding (VVC) standard was developed by the Joint Video Experts Team (JVET) of the ITU-T Video Coding Expert Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). It was published in July 2020 "as the most recent international video coding standard" [16, Bross et al., 2021, p. 3736].
This innovative standard in video codification provides "approximately 50% bit-rate savings over its predecessor, the High Efficiency Video Coding (HEVC) standard" [15, ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2], "and 75% over the currently most-used format, AVC" [16, Bross et al., 2021, p. 3736]. The main goals to define this new standard were to have high performance with the top resolution videos (8k and 4k), "wide colour gamut and 10-bit sample precision per colour component" [15, ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2].

It also implements the functionality of multi-application versatility, one of the top requisites in this industry, including “six profiles to serve as conformance points for applications with up to 10 bits per colour component sample” [15, ISO/IEC JTC 1/SC 29/AG 03 2021, p. 2].

“VVC was designed not only to provide a substantial bit rate reduction compared to its predecessor (HEVC) but also to be highly versatile, i.e., to cover all current and emerging media needs” [16, Bross et al., 2021, p. 3737]. This idea reflects the necessity of including 8k or beyond resolutions, HDR and WCG or other gaming implementations such as 360º video and augmented reality. [16]

In the upcoming sections it is described the most important characteristics of the VVC codec. For a more detailed overview, the reader is referred to [16].

2.3.1 High-Level Functionalities

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3737]:

> VVC inherited much of the high-level syntax (HLS) designs from AVC and HEVC, including the structuring of the bit stream into NAL units and the use of cached parameter sets with indexed referencing. This contrasts with the international video coding standards developed before AVC, e.g., H.262/MPEG-2 and H.263, in which a start-code based bit stream structure with simple headers has been used.
>
> This section discusses some of the high-level functionalities provided by the VVC HLS features and their uses in various applications.

2.3.1.1 Random access

Random access capability refers to the ability to start consuming video content from positions other than the very beginning of the bitstream.

2.3.1.2 Reference Picture Resampling

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3738]:

> Conventionally, e.g., in HEVC, the spatial resolution of a video bitstream can only change at an IDR picture or equivalent, as illustrated by the upper half of Fig. 1. VVC also allows the spatial resolution to change at inter-coded pictures, as illustrated by the lower half of Fig. 1, through the support of the feature referred to as reference picture resampling (RPR).

2.3.1.3 CTUs, Slices, Tiles, and Wavefronts

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3739]:

> The basic processing unit within a picture in VVC, as in HEVC, is the coding tree unit (CTU), which contains the luma and chroma samples of a square region of the picture (except for truncation of the CTUs at the right or bottom edges when the width or height is not divisible by the CTU size). In VVC the CTUs can be larger than in HEVC, but the concept is the same, and is similar to the concept of a macroblock in AVC.

2.3.1.4 Subpictures

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3740]:

> A functionality that is useful for some applications and is especially needed for high-resolution immersive video is the support of so-called bitstream extraction and merging (BEAM) operations. BEAM support was an important design goal in the development of VVC HLS.

2.3.1.5 Virtual Boundaries

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3740]:

> Virtual boundaries are boundaries within pictures where the in-loop filter operations that would apply across the boundaries are disabled. The granularity of the possible locations of virtual boundaries is eight luma samples. This feature serves two purposes. The first is for avoiding seam artifacts introduced by applying the in-loop filters across an artificial boundary created by a preprocessing step before encoding. The second purpose of virtual boundaries is for use with GDR.

2.3.1.6 Parameter Sets and Other "Header" NAL Units

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3741]:

> Parameter sets are syntax structures that are stored in a cache and have an associated index for identifying which parameter set of a given type is being referenced. They were first introduced in the AVC standard. Besides the VPS, SPS, and PPS, a new type of parameter set, the adaptation parameter set (APS), was introduced in VVC. Another type of NAL unit used in VVC is the picture header (PH).

2.3.1.7 Scalability and Layered Coding

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3741, 3742]:

> Scalable video coding refers to the structuring of a coded video bitstream by an encoder in a way that enables the extraction and decoding of subsets of the coded data to produce decoded content with lower quality or to produce alternative or supplemental decoded content.

2.3.2 Core Compression Technologies

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3742, 3743]:

> Fig. 4 shows the functional diagram of a typical hybrid VVC encoder, including a block partitioning that splits a video picture into CTUs, block-based intra- and inter-picture prediction, spatial transformation and quantization of the prediction residual, in-loop filtering of the reconstructed signal after scaling (a.k.a. "inverse quantization") and inverse transformation, followed by header formatting and context adaptive binary arithmetic coding (CABAC) entropy coding for bit stream generation.

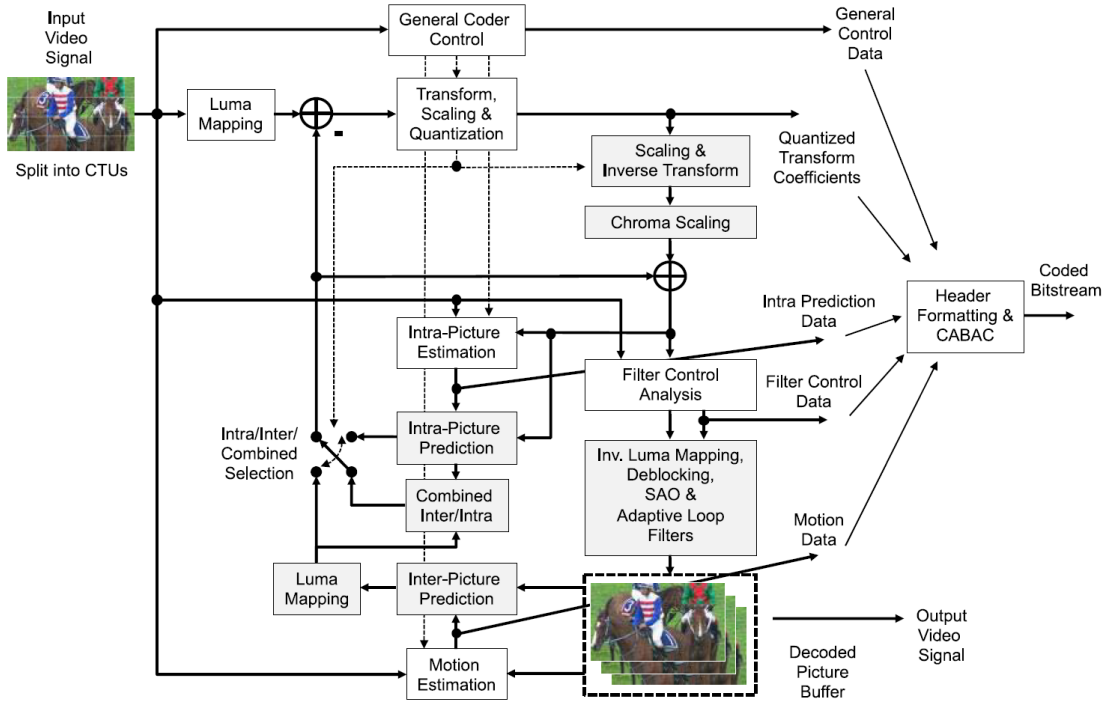

Figure 4: Typical VVC encoder.

2.3.2.1 Block Partitioning

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3743]:

> The HEVC quadtree partitioning of a CTU has been extended in VVC by enabling a more flexible partitioning and supporting larger block sizes. The main VVC partitioning techniques are listed in the following and a more detailed description can be found in [21]: Quadtree Plus Multi-Type Tree, Chroma Separate Tree (CST), Virtual Pipeline Data Units (VPDUs).

2.3.2.2 Intra-Picture Prediction

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3744]:

> Advanced intra-picture prediction techniques in VVC include the DC and planar modes similar to HEVC plus finer-granularity angular prediction with more angles compared to HEVC (93 vs 33), additional matrix-based prediction modes for luma, and cross-component prediction modes for chroma. The new intra coding tools are summarized in the following: Finer-Granularity Angular Prediction, Wide-Angle Intra Prediction (WAIP), 4-Tap Fractional Sample Interpolation Filters, Position-Dependent Prediction Combination (PDPC), Multiple Reference Lines (MRL), Matrix-Based Intra-Picture Prediction (MIP), Intra Sub-Partition (ISP) Mode, Cross-Component Linear Model (CCLM), Extended Most Probable Mode (MPM) Signalling.

2.3.2.3 Inter-Picture Prediction

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3745]:

> As in AVC and HEVC, inter-picture prediction in VVC uses either single-MV uniprediction referencing a picture in a list of previously decoded reference pictures or bi-prediction using

two MVs and indices into two lists of pictures called list 0 and list 1 to select the reference pictures to be used with the MVs to form two prediction signals that are then averaged together.

Beyond that, VVC introduces a variety of new coding tools for more efficient representation, prediction and coding of motion compensation control information, as well as for enhancing the motion compensation processing itself.

2.3.2.3.1 Advances in coding motion information

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3745]:

> Using motion information from temporally and spatially neighbouring blocks, HEVC introduced the merge mode and the advanced MV prediction (AMVP) mode for the prediction and coding of motion parameters in inter-picture prediction. In VVC, both of these modes are extended using improved predictors, enabling MV differences (MVDs) for the merge mode, and providing a more flexible MVD signalling for the AMVP mode to improve the trade-off between motion accuracy and motion overhead bits. These enhancements are described in the following, with a more detailed description provided in [26]: History-Based MV Prediction (HMVP), Symmetric MVD (SMVD), Adaptive MV Resolution (AMVR), Pairwise Average MV Merge Candidate, Merge with MVD (MMVD).

2.3.2.3.2 Advances in CU-level motion compensation

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3745]:

> VVC enhances CU-level motion compensation by introducing more flexible weighting of the prediction signals. This includes the ability to predict non-rectangular partitions inside a CU by applying weighting matrices to each prediction signal for bi-prediction combinations known as the geometric partitioning mode. Furthermore, VVC allows the combination of merge mode with intra-picture prediction and signalling of bi-prediction weights at the CU-level. A short summary of the three tools is provided in the following, and the reader is referred to [26] for further details: Geometric Partitioning Mode (GPM), Combined Intra/Inter-Picture Prediction (CIIP), Bi-Prediction With CU-Level Weights (BCW).

2.3.2.3.3 Refined sub block-based motion compensation

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3746]:

> VVC also introduces technologies that represent motion in higher granularity, e.g., with a subblock level, or further refine motion estimates at the decoder instead of using explicit signalling. VVC further increases the MV precision and fractional sample motion compensation to 1/16 luma sample in some modes. These new features are summarized in the following, and a more detailed description is provided in [27]: Subblock-Based Temporal MV Prediction (SBTMVP), Affine Motion, Prediction Refinement with Optical Flow (PROF), Decoder MV Refinement (DMVR), Bi-Directional Optical Flow (BDOF).

2.3.2.3.4 Horizontal wrap-around motion compensation

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3747]:

> For some specific immersive video projection formats further detailed in Section III.G, a special case of motion compensation can be applied to alleviate the appearance of seam artifacts in 360 video coded in the equirectangular projection (ERP) format or other 360 video projection formats that share similar properties [28].

2.3.2.4 Transforms and Quantization

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3747]:

> The basic concept of applying an integer transform to the prediction residual followed by quantization of the transform coefficients is well known from previous standards and is retained in VVC: Larger and Non-Square Transforms, Multiple Transform Selection (MTS), Low Frequency Non-Separable Transform (LFNST), Subblock Transform (SBT) Mode, Extended Quantization Control, Adaptive Chroma QP Offset, Dependent Quantization (DQ), Joint Coding of Chroma Residuals (JCCR).

2.3.2.5 Entropy Coding

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3748]:

> The entropy coding in VVC is performed using CABAC as the sole entropy coding method, as in HEVC. In VVC, the efficiency of CABAC is further improved by the following changes in the coefficient coding and probability estimation. More details on the VVC entropy coding can be found in [30]: Improved Coefficient Coding and High-Accuracy Multi-Hypothesis Probability Estimation.

2.3.2.6 In-Loop Filters

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3748]:

> In VVC, improved and new signal enhancing in-loop filters are applied to the reconstructed video signal before the pictures are used for output and as references for subsequent motion compensated prediction. This includes a new luma mapping with chroma scaling tool, where the inverse luma mapping part is applied before all other in-loop filters: Luma Mapping With Chroma Scaling (LMCS), Long Deblocking Filters, Luma-Adaptive Deblocking, Adaptive Loop Filter (ALF), Cross-Component ALF (CC-ALF).

2.3.2.7 Screen Content Coding Tools

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3749]:

> Special coding tools are included in VVC to increase the coding efficiency for video that has different characteristics from camera-captured content: Palette Mode, Adaptive Color

Transform (ACT), Intra-Picture Block Copy (IBC), Block-Based Differential Pulse-Code Modulation (BDPCM), Transform Skip Residual Coding (TSRC).

2.3.3 Performance

Extracted from the VVC Invited Paper [16, Bross et al., 2021, p. 3750 - 3752]:

> Standard dynamic range (SDR) with ultra-high definition (UHD) [19] (with the reference software test model showing an overall average bit-rate savings estimate of 43% and another publicly available encoder discussed in item 5 below showing 49%)
>
> Standard dynamic range (SDR) with high definition (HD) [20] (with the reference software test model showing an overall average bit-rate savings estimate of 49% and another publicly available encoder discussed in item 5 below showing 51%)
>
> 360° immersive projection formats [20] (with the reference software test model showing an overall average bit-rate savings estimate of 50–56%, depending on the projection format)
>
> High dynamic range (HDR) with high definition (HD) resolution [21] (with the reference software test model showing an overall average bit-rate savings estimate of 49–52%, depending on the HDR content type)

2.4 Comparison of codecs

Both codecs are successors of the consolidated HEVC codec and created by different associations but both with MPEG compression experts involved. In terms of compression, VVC offers a higher coding efficiency even though both present a significant improvement compared to HEVC. Even though, as it is explained in the video codecs industry, compression efficiency is not the only factor to consider. In this aspect, EVC offers better compatibility, since it is designed to offer backward compatibility with existing video coding standards and VVC, otherwise, has a more complex licensing structure and it may require hardware and software upgrades to support the whole set of characteristics this new standard provides.
To sum it up, both are great compression algorithms which provide different and useful implementations for the video codecs sector, and it seems safe to say both will be crucial in the near future.

CHAPTER 3

CARBON FOOTPRINT

In this chapter it is described the Carbon footprint definition and the history behind this concept. Finally, it is explained how it will be computed in this study.

3.1 Carbon footprint definition

As it is briefly described in the introduction, there is still not a standard international definition for carbon footprint, although all of them point to the same fundamental aspect: to quantify the amount of contaminant gasses emitted to the atmosphere. Nevertheless, we will study it in more detail, starting from the beginning.

The expression of Carbon footprint comes from the Ecological Footprint term. Introduced in the 1994 by Mathis Wackernagel [17, Wackernagel 1994, p. 68]:

> The Ecological Footprint or the Appropriated Carrying Capacity (EF/ACC) is defined as the aggregate land (and water) area in various categories required by the people in a defined region to provide continuously all the resources and services they presently consume, and to absorb continuously all the waste they presently discharge using prevailing technology.

Therefore, since the 1990s there was already the intention of defining a term to quantify the impact of human activity and the invested resources. Consequently, the carbon footprint unifies this definition with the Carbonium emissions, directly or indirectly.

3.1.1 IPCC definition

The Intergovernmental Panel on Climate Change (IPCC), which is an intergovernmental organization of the United Nations founded in 1988, defines carbon footprint as “Measure of the exclusive total amount of emissions of carbon dioxide (CO_2) that is directly and indirectly caused by an activity or is accumulated over the lifecycle stages of a product (Wiedmann and Minx 2008)” [18, van Diemen et al., 2022, p. 1795].

3.1.2 Kyoto Protocol

In 1992, The United Nations Framework Convention on Climate Change (UNFCCC) established an international environmental treaty to combat "dangerous anthropogenic interference with the climate system" [19, UNFCCC 1992, pg. 9], in part by stabilizing greenhouse gas concentrations in the atmosphere. [19]

The Kyoto Protocol extended the UNFCCC treaty and was the first international agreement that aimed to reduce carbon dioxide (CO_2) emissions and the presence of greenhouse gases (GHGs) in the atmosphere. The essential tenet of the Kyoto Protocol was that industrialized nations needed to lessen the amount of their CO_2 emissions.

After becoming a signatory in 2013, Afghanistan became the 192nd and last signatory of the Kyoto Protocol. [20]

In 2015, however, countries agreed on yet another legally binding climate treaty, the Paris Agreement, which entered into force in November 2016 and effectively replaced the Kyoto Protocol. The Kyoto Protocol required only developed countries to reduce emissions, while the Paris Agreement recognized that climate change is a shared problem and called on all countries to set emissions targets. By 2022, the UNFCCC had 198 parties. [21]
On 1 April 2016, the United States and China confirmed they would sign the Paris Climate Agreement. It laid down a change without precedents since they represented almost 40% of total global emissions. [22] Although with the Trump transition the United States exempted from the treaty, on 20 January 2021 President Biden signed the instrument to bring the United States back into the Paris Agreement. [23]

In terms of Carbon footprint, the Kyoto protocol provided a significant change afterward they ruled that the footprint is the measure of the total amount of emissions of greenhouse gases, not only CO2. They determined them as carbon dioxide (CO_2), methane (CH_4), nitrous oxide (N_2O), hydrofluorocarbons (HFCs), perfluorocarbons (PFCs), Sulphur hexafluoride (SF_6) and sulphur hexafluoride (SF_6). [24]

3.2 How to compute it

In this project, we have followed the definition and implementation provided by the Ecological transition and Demographic challenge Department[1], from the Spanish Government. The official definition provided is that "the carbon footprint of an organization is the totality of greenhouse gasses emitted directly or indirectly through the activity developed by the corresponding organization" [25, MITECO n.d., p. 1]. Also provides a generic computational formula, to calculate the carbon footprint in an organization, which is described as "the result of the product of the consumption data with their corresponding emission factor" [25, MITECO n.d., p. 1], as it can be seen in [25].

Besides all this information, they provide a Carbon footprint calculator for organizations. It has Range 1+2, where Range 1 encompasses direct emissions and Range 2, encompasses indirect emissions provided by the purchase of electricity and other energies. To compute the emissions of the codecs I will only use the Range 2 emissions since they only consume electricity. The rest of the fields and emissions included in the calculator will be omitted in the procedure.

Considering that the 2022 calculator is not correctly working at the time of this writing, in this study it is used the 2021 calculator.
The calculator can be found on [26].

3.2.1 Fields to fill in

In the calculator, we need to fill some specific cells to automatically calculate the carbon footprint. These are the following areas:

- Name of the energy supply company

[1] https://www.miteco.gob.es/es/

- Does it have a warranty of origin?
- Consumption data (kWh).

The consumption data is the information that we will compute later, on the practical part of the study. For the other two categories, we must choose as its name implies, the name of the electrical business provider and if it has a warranty of origin, which implement 3 options:

- GdO high-efficiency cogeneration.
- GdO renewable energy.
- No.

Depending on the GdO selection, the Factor Mix elec. kg CO2e/kWh changes.

Considering that the experiment has been executed at my house, I checked the electronic supply company hired at my place, and I contacted them to know the GdO they have.
At my house, Endesa Energía S.A.U is the energy supply company, and they have the GdO of high-efficiency cogeneration.

Even though this is the configuration of my place, I will compute the top five business suppliers of energy in Spain, which are Endesa, Iberdrola, Naturgy, Repsol and EDP. [27]

CHAPTER 4

PROCESS

In this section it is described the whole process of collecting information, defining the elements and technological instruments involved.

4.1 Local Environment Description

4.1.1 Computer

The computer where the codecs have been implemented is the Lenovo ideacentre All-In-One (AIO) Y910[2].

4.1.1.1 Features

The following list is a description of the most relevant features of the computer. [28]

- Processor: 6th Gen Intel Core i7 6700.
- Memory: 16GB DDR 2133 SODIMM.
- GPU: NVIDIA GeForce GTX 1080.
- Storage:
  - 1TB HD SATA.
  - 256GB SSD PCIe.
- Side Port:
  - 3 x USB 3.0.
  - Card Reader (6 in 1).
  - Headphone & Mic array.
  - HDMI In Switch Button.
- WLAN: Killer Wireless-AC 1535.
- Lan: Killer Double shot Pro LAN, Gigabit Ethernet.

4.1.1.2 Linux Operating System implementation

Since codecs implementation must be done in Linux operating system and my computer has Windows operating system, I had to find out the best option to change it.
The best implementation option is to install the operating system in the internal SSD or HDD to have the best performance. The problem is that I could not do that, because I didn't have space to split my SSD. Therefore, we decided that the best option would be to buy a new SSD, powerful enough to do the computations required and to implement through the USB 3.0 port with a USB C External NVMe SSD Enclosure and execute it natively.

The chosen solid-state drive is the Kingston NV2 500GB SSD PCIe 4.0 NVMe Gen 4x4[3], with the External Enclosure Ugreen M.2 NVMe 10 Gbps SSD[4].

To do the process, I followed the instructions in [29] and videos [30] and [31].

[2] https://www.lenovo.com/es/es/desktops-and-all-in-ones/ideacentre/y900-series/AIO-Y910-27
[3] https://www.pccomponentes.com/kingston-nv2-500gb-ssd-pcie-40-nvme-gen-4x4?
[4] https://www.amazon.es/UGREEN-Carcasa-10Gbps-Disco-Cable/

In Figure 4.1, we can see the rufus[5] implementation.

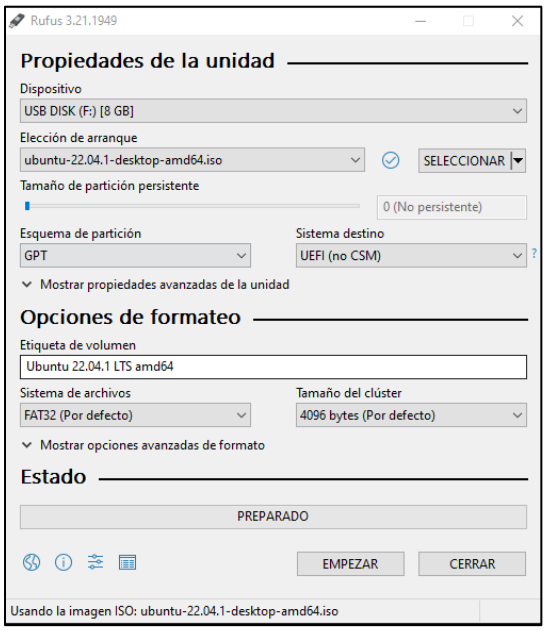

Figure 4.1: Rufus implementation.

The next steps are not documented since in BIOS mode it is not possible to take screenshots. When the procedure is done, we can execute the Linux operating system natively. I have implemented one of the most common and used open-source Debian-based Linux distributions, Ubuntu[6].

I have implemented the SSD following [32].

4.1.2 Electricity consumption meters

To obtain the electricity power consumption of the codecs we have obtained two electricity meters.

4.1.2.1 Gafild consumption meter

The Gafild electricity meter[7] is a multifunction meter with potency control (W), energy (kWh), voltage (V), amperage (A), potency factor, cost and maximum and minimum potency allowance settings. It will be the principal and reliable device that will be implemented through the process to calculate the electricity consumption that the carbon footprint requires. The data recollection will be implemented with images in the document since it is only displayed on its screen.

[5] https://rufus.ie/es/
[6] https://ubuntu.com/
[7] https://www.amazon.es/dp/B081TBGQVZ?psc=1&ref=ppx_yo2ov_dt_b_product_details

Figure 4.2: Gafild electricity meter.

4.1.2.2 TP-Link Tapo P110

The TP-Link Tapo P110[8] is an intelligent wall plug with Selectable monitoring of electricity consumption, with other functionalities as remote control, timer or voice control. The main usage that this meter will have, is to show big data files and process external recollection of information since the data can be exported in excel format.

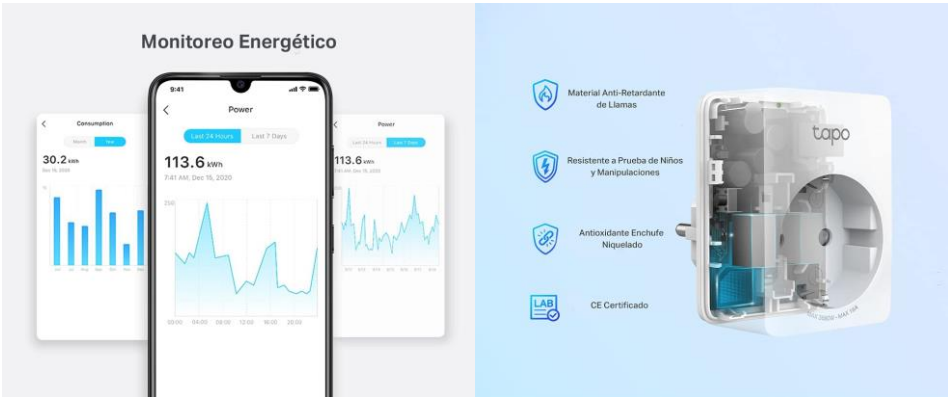

Figure 4.3: TP-Link Tapo P110.

4.2 Implementation of Codecs

First of all, I did a free GNU/Linux Online Terminal and Programming IDE learning course to get more fluent with Linux terminal commands and to collect command processes that I would need later to be fluent working with the huge number of files of this research. The online learning platform was Webminal[9].

Both codecs are open-source, and I have obtained them from GitHub repositories.

4.2.1 EVC

For the encoder, it has been repository cloned from eXtra-fast Essential Video Encoder (XEVE)[10], I have followed the installation procedure described in the "How to build" section. We can also find the usage specifications in the "How to use" section.

[8] https://www.amazon.es/dp/B09BFT7NZJ?psc=1&ref=ppx_yo2ov_dt_b_product_details
[9] https://www.webminal.org/
[10] https://github.com/mpeg5/xeve

See Annex A, Figures A1 to A6, to see the terminal output on the completed and correct installation of the encoder.

For the decoder, it has been repository cloned from eXtra-fast Essential Video Decoder (XEVD)[11]. The installation procedure has the same steps as the encoder.
See Annex A, Figures A7 to A9, to see the terminal output on the completed and correct installation of the decoder.

4.2.2 VCC

Repository cloned from Fraunhofer Versatile Video Encoder (VVenC)[12], I have followed the installation procedure described in the "Build information". We can also find the usage specifications on the "Usage documentation".
See Annex A, Figures A10 to A22, to see the terminal output on the completed and correct installation of the encoder.

For the decoder, it has been repository cloned from Fraunhofer Versatile Video Decoder (VVdeC)[13]. The installation procedure has the same steps as the encoder.
See Annex A, Figures A23 to A33, to see the terminal output on the completed and correct installation of the decoder.

4.3 Nmon

To track the computational cost of the computer, I have implemented the computer performance system monitor tool, nmon[14]. Created by Nigel Griffiths, the Linux version was released in 2009. [33]
I have chosen this monitoring tool because of its easy implementation, contrasted quality in data collection and for the facilities that allow me to work with them after the compilation.

The code line that is being implemented to track the computational cost, follows the next structure:

"nmon -f -s <seconds> -c <count>",

where -f saves to file mode, -s sets the time between captures and -c the number of captures. The creator suggests having between 300 and 600 data captures to obtain a detailed graph. The output is named like: <hostname>_<date>_<time>.nmon. [34]

Once the nmon file is created, to easily create graphs to interpret the data, we just need to execute the following command:

"./nmonchart <hostname>_<date>_<time>.nmon output.html",

which changes the nmon file to an html file, in this example named output.html, which now can just be executed to choose the desired graph, as we will see on the results. The nmonchart software used in this previous line is extracted from nmon[14] as well.

[11] https://github.com/mpeg5/xevd
[12] https://github.com/fraunhoferhhi/vvenc
[13] https://github.com/fraunhoferhhi/vvdec
[14] https://nmon.sourceforge.net/pmwiki.php?n=Site.Documentation

4.4 Coding procedure

In this section it is explained the procedure of the encoding experiments that will be used to extract the required information for the final results.

4.4.1 Video selection

In order to have different analysis parameters, I have selected three different resolution and duration videos. All public and copyright free. They have been named with their resolution, for the purpose of being more intuitive for the reader. These are the following:

- 8k.mov[15] (4320p): 7680x4320 pixels, 249 frames and duration of 8s.
- 4k.mp4[16] (2160p): 3840x2160 pixels, 752 frames and duration of 30s.
- HD.mp4[17] (1080p): 1920x1080 pixels, 241 frames and duration of 8s.

4.4.2 Executable scripts

4.4.2.1 Encoding process for each video and codec

To make the encoding process and corresponding measures more efficient, I created an executable script for each video and encoder. Each script contains the following first line:

"#!/bin/bash",

also known as the 'shebang', which is the command necessary to make the file executable. Then, the second line is the command "date" to exactly know when the process starts. The third corresponds to the nmon monitoring tool and finally, the last one is the encoder command. The scripts are all available on the Annex A, Figures A34 to A39.

All the processes were previously tested to know with a margin of seconds the exact duration of the process to select the better nmon configuration for each one. Also, it was very important since the electricity power consumption meter does not provide the data collection, as it is explained in section 4.1.2.1, and I had to stand by.

4.4.2.2 Encoding process of all videos consecutively for each codec

To make a deeper understanding in the electronical consumption of both codecs, I have also implemented two more scripts which will encode the three resolution videos that we have used to compute the Carbon footprint. The electricity consumption will be computed with TP-Link Tapo meter, described in section 4.1.2.2, which will provide a more graphical and intuitive representation to supplement the main results. The scripts are all available on the Annex A, Figures A40 and A41.

4.5 Experiment execution

First of all, I calculated the kWh of the computer without executing any process. As we can see in Figure B1, from Annex B, in one hour the computer consumed 0.083 kWh. Even though this consumption is not directly connected with the codecs, this study's objective is to compute a

[15] https://www.vecteezy.com/video/7672208-8k-pupa-phase
[16] https://www.pexels.com/es-es/video/quema-de-lena-1717998/
[17] https://www.vecteezy.com/video/3672428-abstract-blue-and-pink-particles-fluid-background-video

local environment carbon footprint, then the consumption that is related to the computer is also required by the encoders to execute their function. Therefore, it will not be excluded or subtracted from the consumption values that I will obtain.

All encoded videos have been decoded to verify the correctness of all the executed processes.

4.5.1 EVC encoder

For the three different resolution videos, I have followed the same procedure. As it is described in section 4.4.2.1, I have executed the corresponding file where it has already implemented the desired commands.
We can see the corresponding EVC encoder process information on the terminal output after a correct video codification. The main stats provided are Bitrate, Encoded frame count, Total encoding time, Average encoding time for a frame and Average encoding speed. All the terminal captures are available on Annex A, Figures A42 to A44.
After the encoded output has been created and the nmon process has finished, I executed the command to obtain the html file from the nmon data collection, as it is explained in section 4.3.

4.5.2 VCC encoder

Following the same steps done with the EVC encoder, I executed all the files with the encoding process implemented on them. They are all available on the annex.
In contrast with EVC encoding terminal information output, this codec shows the process for each frame, so I split the screen display into two images, first the execution file command and then the finished process output. The main stats provided are Total frames, Bitrate, Y-PSNR, U-PSNR, V-PSNR, YUV-PSNR, Total time, Average frames per second (fps) and Encoded frames. All the terminal captures are available on Annex A, Figures A45 to A50.
After the encoded output has been created and the nmon process has finished, I executed the command to obtain the html file from the nmon data collection, as it is explained in section 4.3.

4.5.3 Encoding process of all videos simultaneously

Following the non-principal line of the project, as it is described in the 4.2.2.2, I have executed the scripts where all the videos are encoded for each codecs simultaneously. The terminal output of the processes are available on Annex A, Figures A51 to A55.

CHAPTER 5

RESULTS

In this section it is shown the numerical and graphical results obtained from the encoding procedure, seen in the previous section. It is finally calculated the carbon footprint for each process executed.

5.1 Computational cost

In this section it is being analysed the computational costs of both codecs coding the same videos. The Swap back processes into memory per second graphics have been dismissed in the computational analysis since they don't provide any information.

5.1.1 Analysis for 8k resolution

5.1.1.1 CPU

In Figure 5.1 and 5.2, we can see the CPU utilisation Percentages of both codecs. There is a huge difference in the CPU usage between EVC and VVC. The Essential Video Coding standard has a more constant and significantly much lower CPU usage. The whole process has around 12% of User CPU usage, with a maximum value of 20% at 4:06:50 PM.
Meanwhile, the Versatile Video Coding standard uses around 98% to 99% of the CPU usage nearly the whole encoding process. It can be seen that there is a certain uniform pattern where approximately every two to three minutes the user CPU usage drops around 35% to 60%.

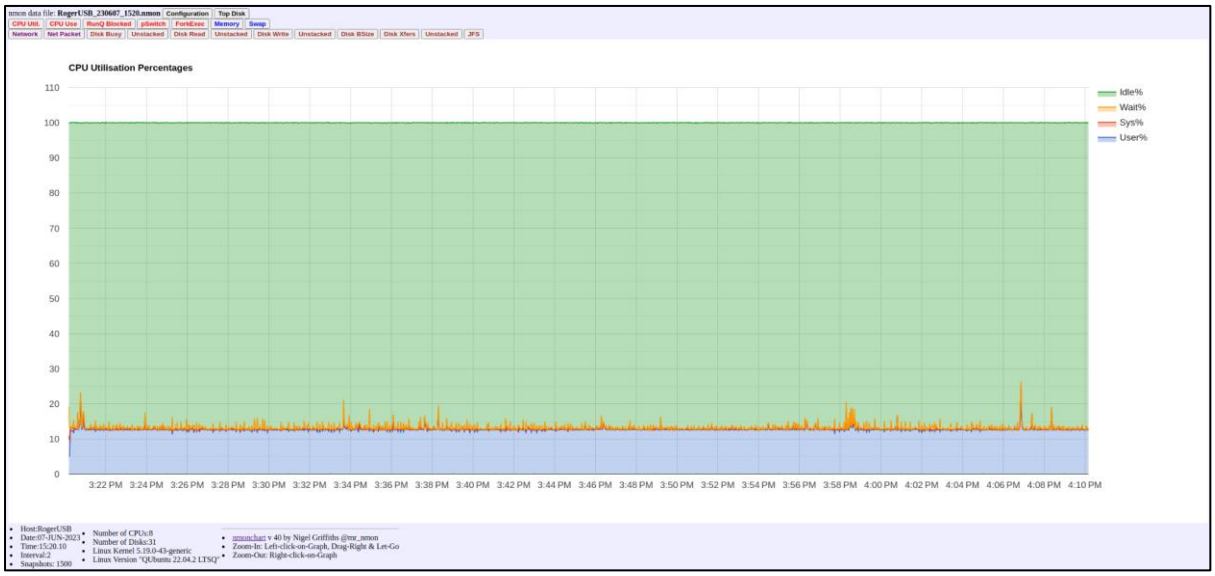

Figure 5.1: CPU usage on 8k resolution codification with EVC.

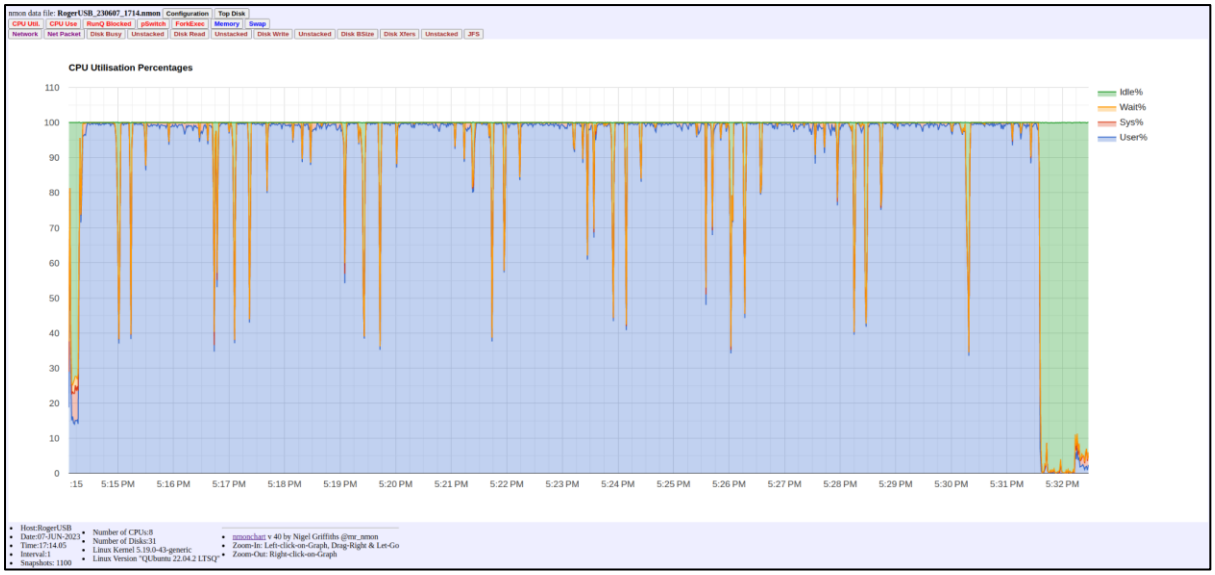

Figure 5.2: CPU usage on 8k resolution codification with VVC.

5.1.1.2 Run queue

In Figures 5.3 and 5.4, we can see the Run queue in the number of processes of both codecs. For the EVC encoder, there is a constant value of two running and 0 blocked processes through the whole process. Apart from that, there is an important number of increases to three and four processes, while there is a maximum value of 5. On the other hand, there is a similar quantity of increases in blocked processes compared to the running ones, but the amount is constant with 1 or 2 maximum blocked processes.
Meantime, with the VVC encoder, there is a constant value of 9 running and 0 blocked processes through the whole process. Even nine could be interpreted as the average running processes, there are a lot more fluctuations in the graph, dropping to 4 or 5 running processes or increasing them to 10 or 12, the most repeated, with maximum values of 19.
There is also 1 blocked process but with a very detached periodicity of time.

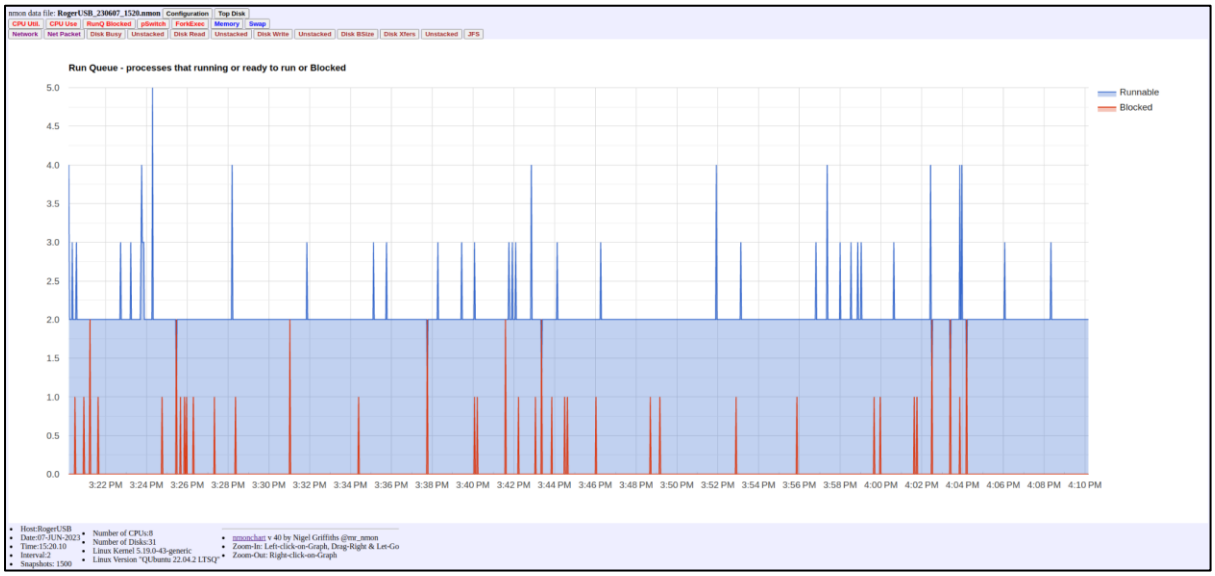

Figure 5.3: Run queue on 8k resolution codification with EVC.

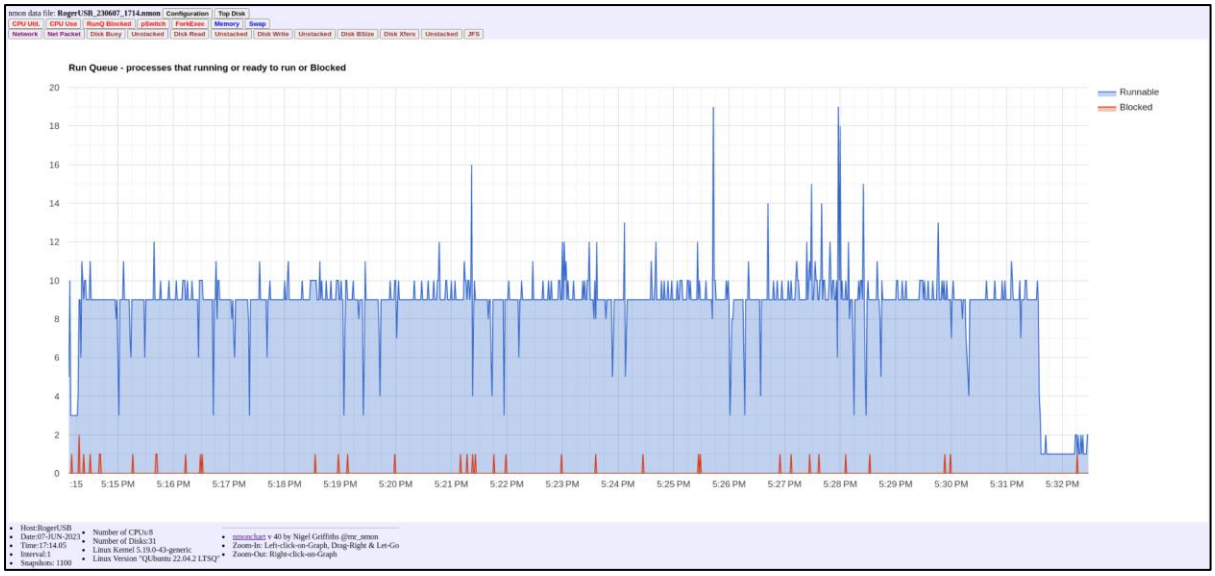

Figure 5.4: Run queue on 8k resolution codification with VVC.

5.1.1.3 Process Switch

In Figures 5.5 and 5.6, we can see the Process switches of both codecs. The Essential Video encoder has a constant value of processes switches which oscillates between 230 and 2000. At the beginning and at the end is where there are two major peaks, one at 85000 processes, and the other one at 93000.
The VVC instead, has a constant value of processes switches that oscillate between 500 and 8000. Also acts differently in the major peaks as it only has one at the end, but significantly higher, with 128000 process switches per second.

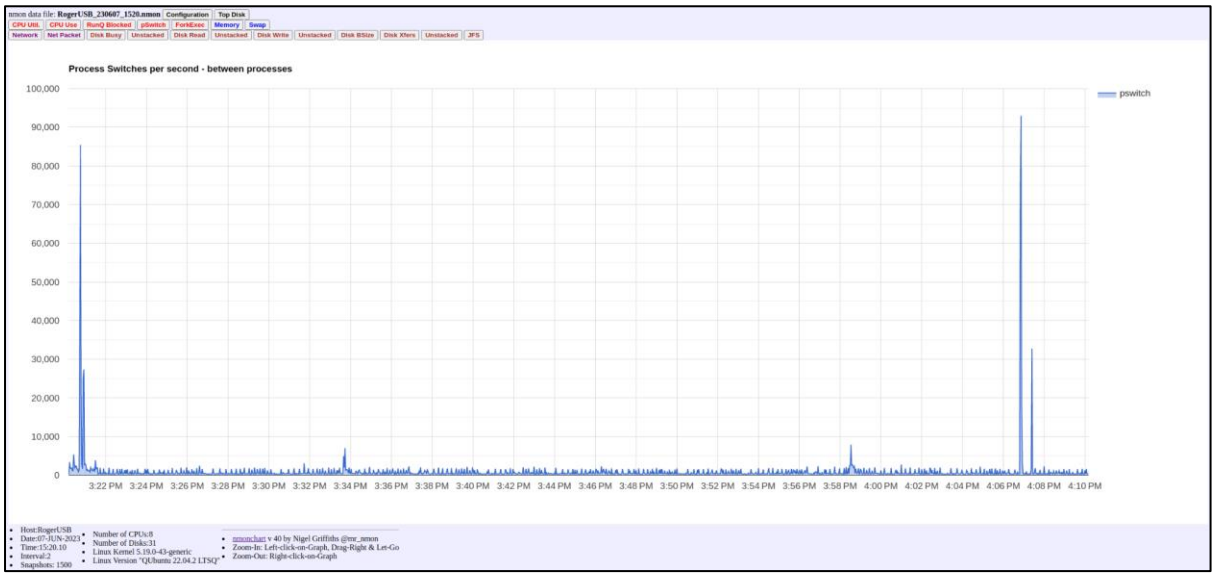

Figure 5.5: Process switch on 8k resolution codification with EVC.

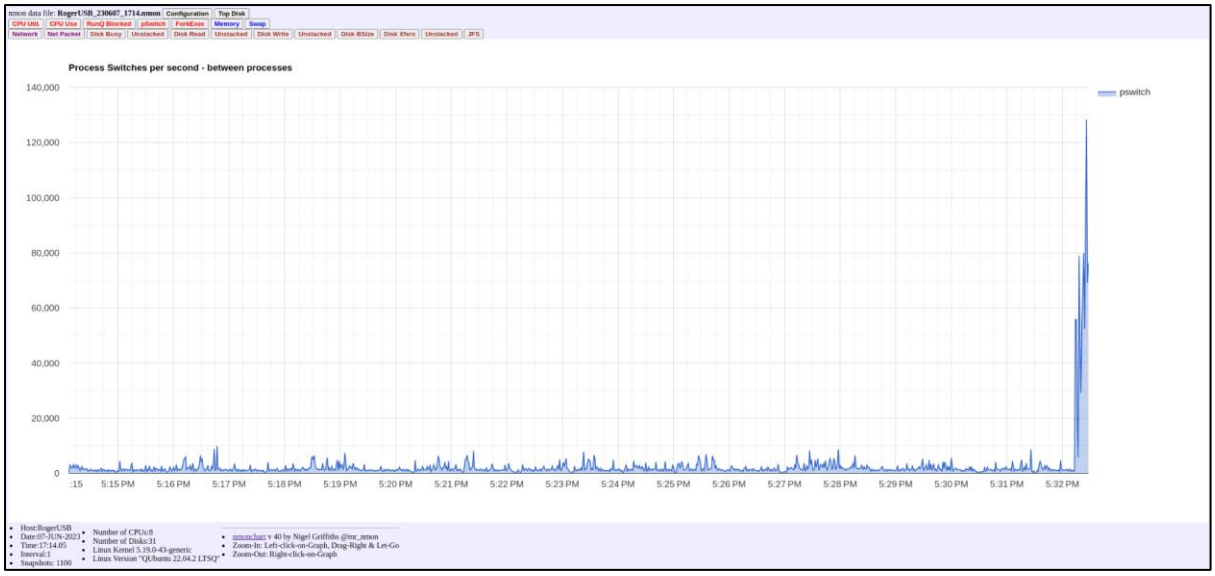

Figure 5.6: Process switch on 8k resolution codification with VVC.

5.1.1.4 System calls per second

In Figures 5.7 and 5.8, we can see the System calls per second of both codecs. In this graphic it will be analysed the System duplicated processes calls, called “fork”, and the overwrite current processes with a new program, called “exec”.
For the EVC encoder, we can see that it does multiple fork system calls per second. They range from 0.5 to 5, with two peaks at 7 system calls.
The VVC encoder behaves slightly differently, as it does not do as many calls. Even though the peaks have higher values, going from 1 to 8, with a maximum value of 10.

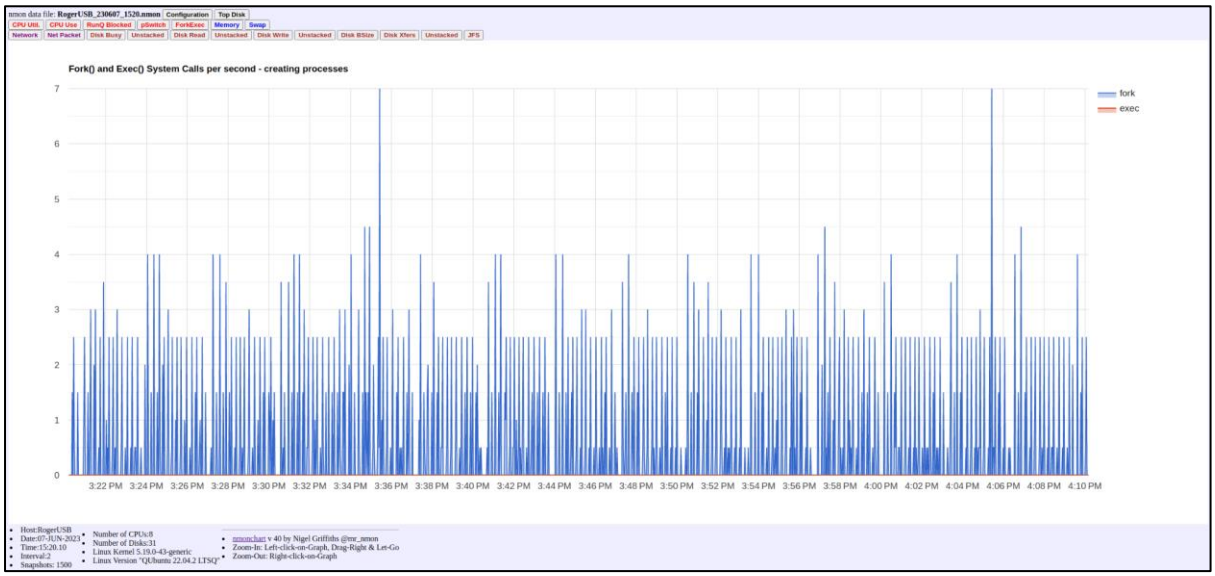

Figure 5.7: System calls per second on 8k resolution codification with EVC.

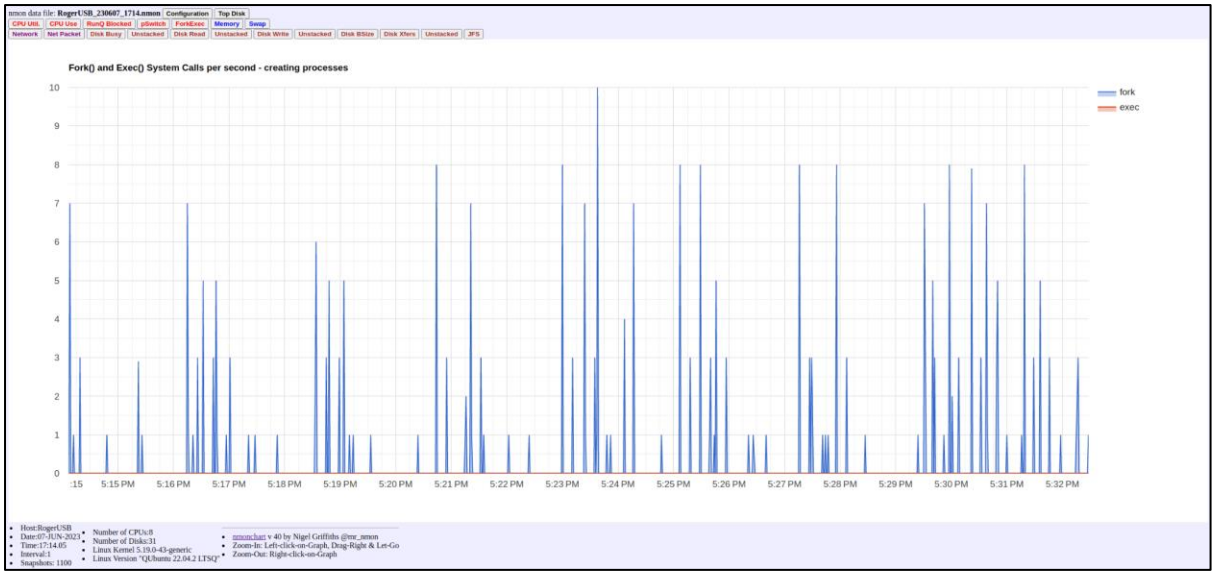

Figure 5.8: System calls per second on 8k resolution codification with VVC.

5.1.1.5 RAM memory usage

In Figures 5.9 and 5.10, we can see the Memory usage of both codecs. For the EVC encoder, as it is shown in the graphic, the total RAM memory available is 32041.9 MB. The cached memory has a constant value of 21000 MB, even though it reaches 25000 MB at the beginning of the process. The active memory ranges from 4450 MB to 7000 MB. Free memory and buffers are both around only 200 MB and the inactive memory ranges from 23000 MB to 26000 MB.

The VVC encoder has a different behaviour. We can see a much lower and non-constant cached memory usage, starting close to 25000 MB, but dropping up to 13000-14000 MB on the whole process. The active memory ranges from 5000 MB to 10000 MB, with a descendant flow. Free memory now has higher values, closer to 500 MB and buffers have decreased, with a constant value of 150 MB approximately. The inactive memory has similar values, ranging from 20000 MB to 25000 MB.

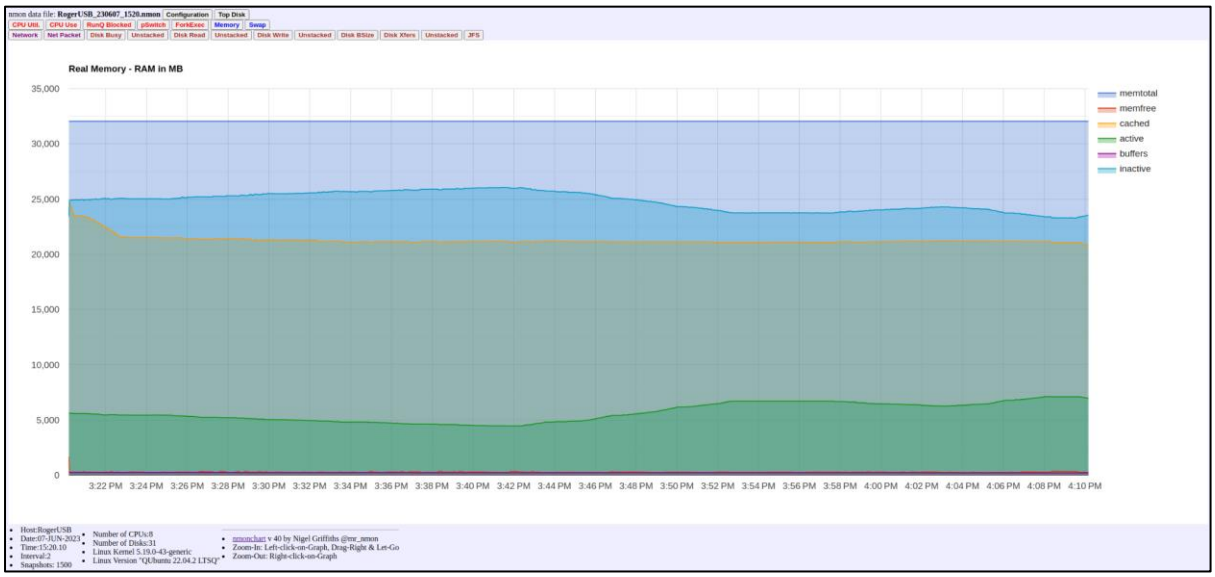

Figure 5.9: RAM memory usage on 8k resolution codification with EVC.

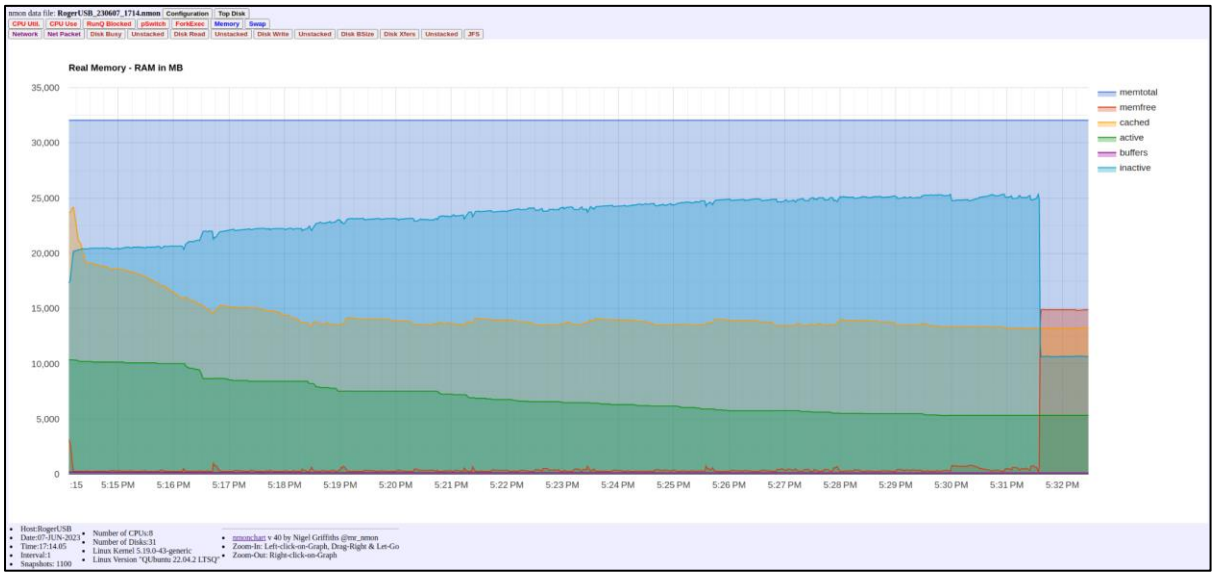

Figure 5.10: RAM memory usage on 8k resolution codification with VVC.

5.1.1.6 Time execution

EVC completed the coding process in 3008.718 s, and the VVC completed it in 1049.895 s.

5.1.2 Analysis for 4k resolution

5.1.2.1 CPU

In Figures 5.11 and 5.12, we can see the CPU utilisation Percentages of both codecs. The results are very similar to the 8k resolution. The EVC codec, once more, has a constant value of CPU usage. It has the same 12% of User CPU usage, again with a maximum of 20%.
With the VVC encoder we have a same constant value of User CPU usage of around the 99% the whole process, but the same certain uniform pattern of CPU usage drops are here much more present, such as the number of them increases from eight to twenty-three approximately, and with the drop of this time around 30% to 50%. The number of usage drops must be major now because of the encoding time.

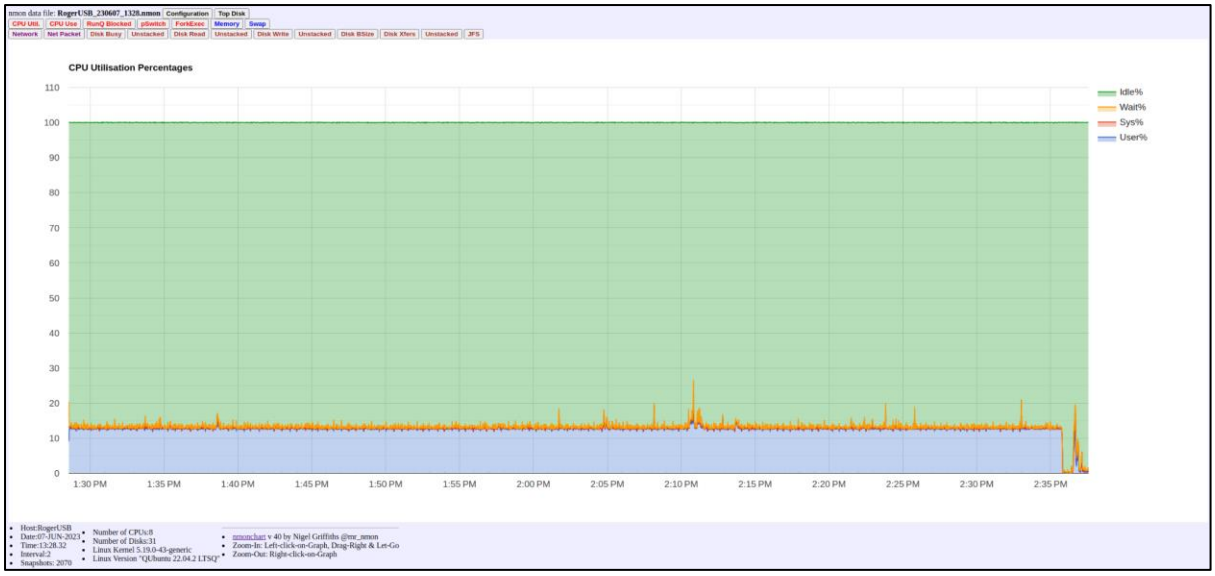

Figure 5.11: CPU usage on 4k resolution codification with EVC

Figure 5.12: CPU usage on 4k resolution codification with VVC.

5.1.2.2 Run queue

In Figures 5.13 and 5.14, we can see the Run queue in number of processes of both codecs. The Essential Video Coding standard behaves exactly the same as with 8k resolution, with the only difference that now has a maximum value in running processes of 12, and some peaks with 5, 6 and 7 running processes.
The VVC encoder behaves as well as the highest analysed resolution. Has the same average value at 9 running processes, with the same values and frequency in peaks.

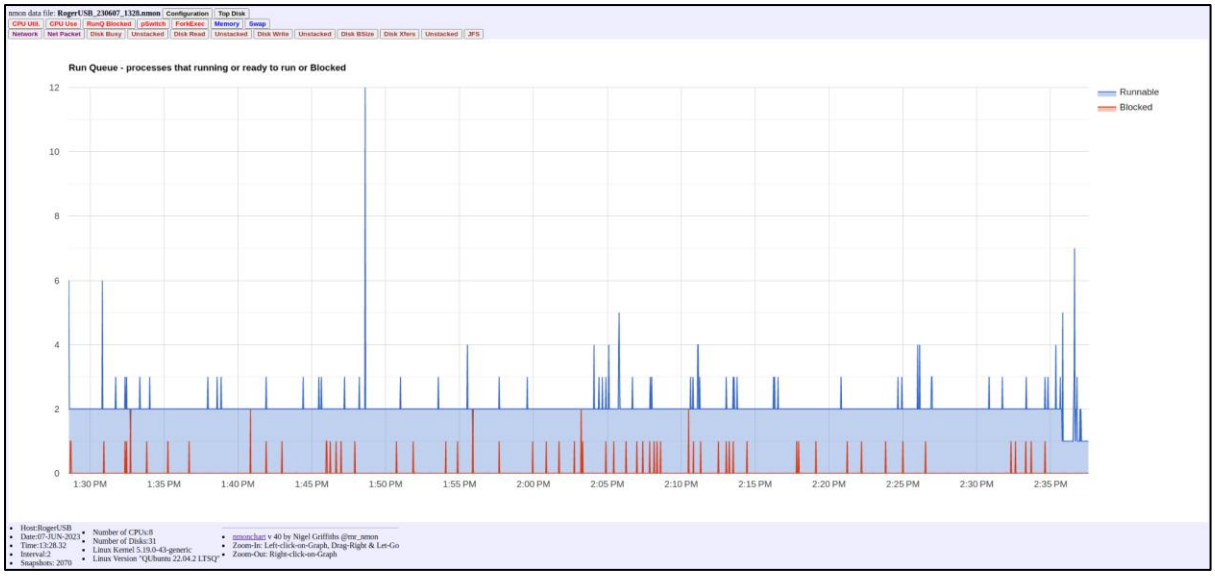

Figure 5.13: Run queue on 4k resolution codification with EVC.

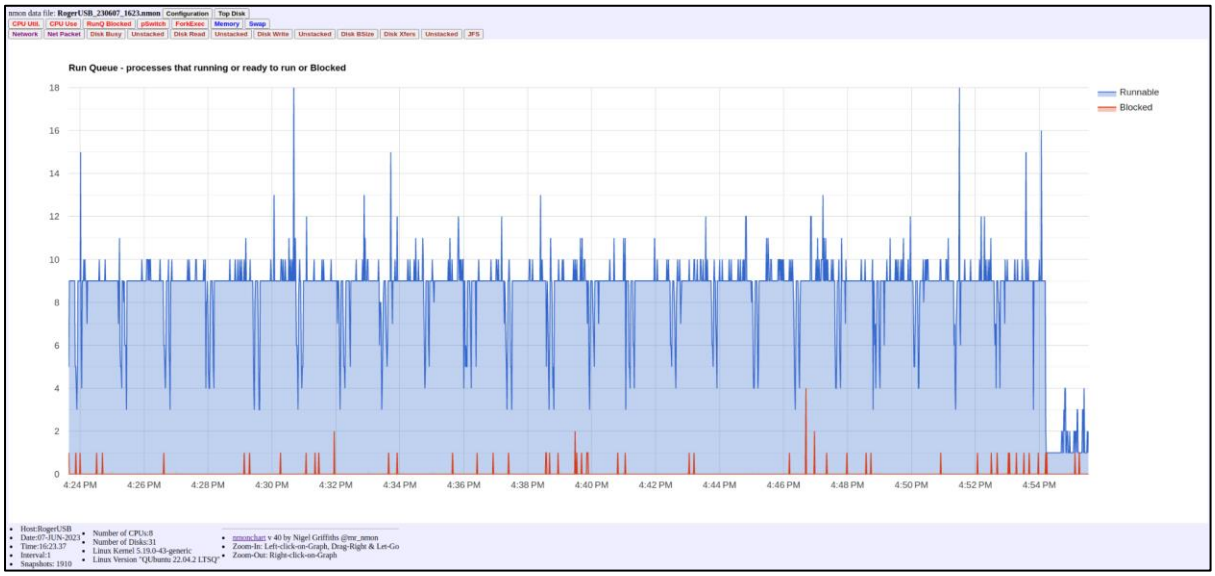

Figure 5.14: Run queue on 4k resolution codification with VVC.

5.1.2.3 Process Switch

In Figures 5.15 and 5.16, we can see the Process switches of both codecs. The EVC codec has the same behaviour as with 8k resolution, with the constant value of processes switches which oscillates between 300 and 2000, but with only one major peak at the end with value 130000 process switches per second.
The VVC also follows his superior resolution values, having a constant value of processes switches that oscillate between 300 and 12000-14000. The final peak has a value of 280000 process switches per second.

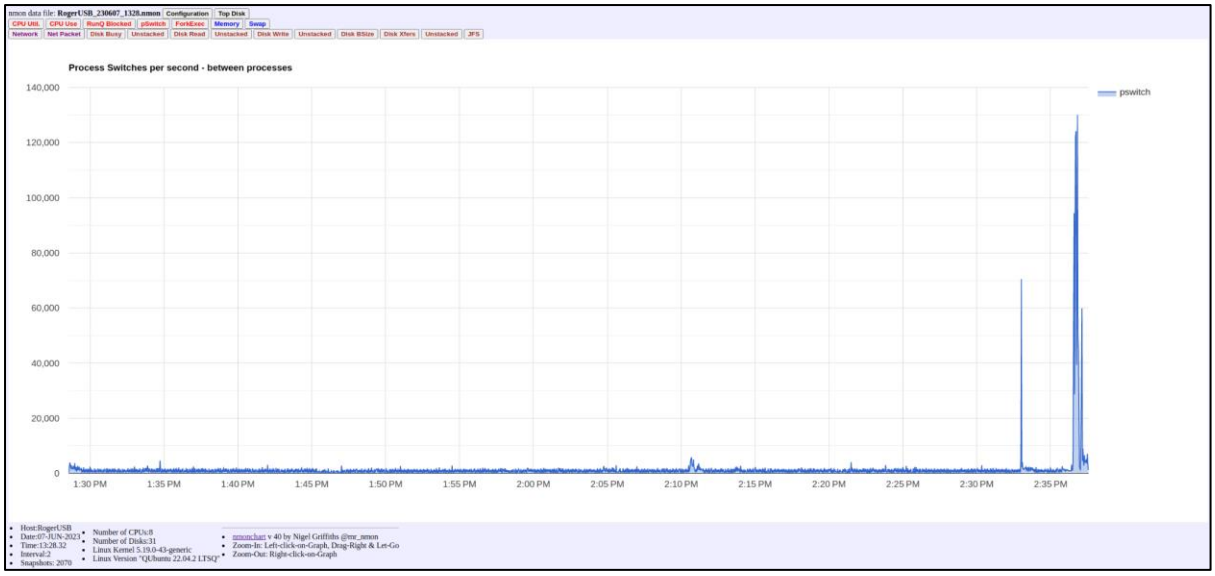

Figure 5.15: Process switch on 4k resolution codification with EVC.

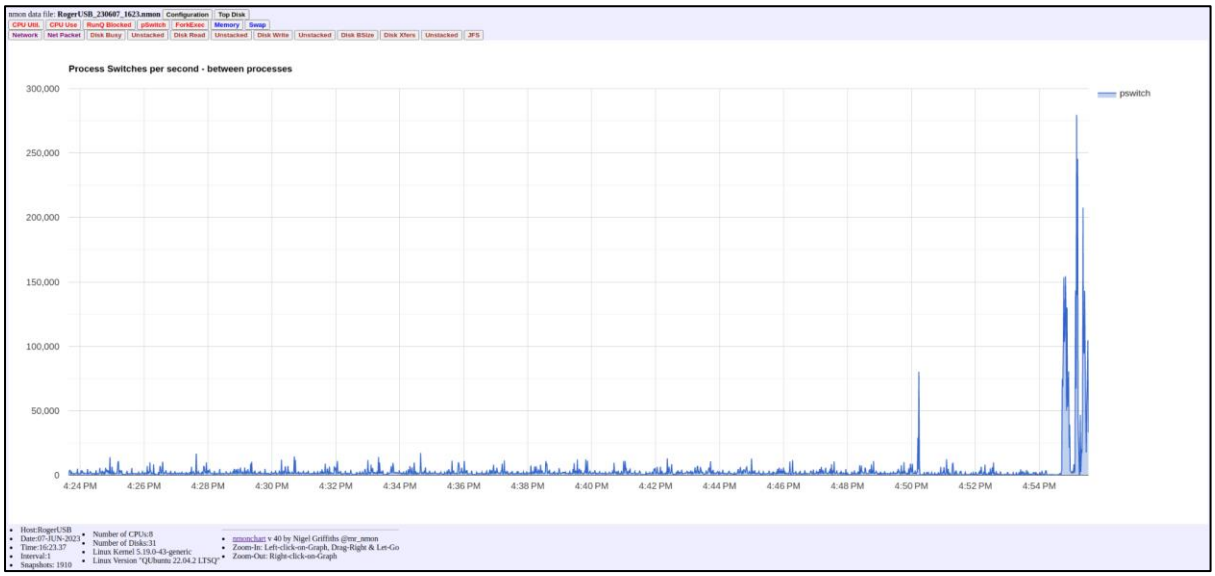

Figure 5.16: Process switch on 4k resolution codification with VVC.

5.1.2.4 System calls per second

In Figures 5.17 and 5.18, we can see the System calls per second of both codecs. The EVC encoder behaves exactly as with 8k resolution. The peaks range from 0.5 to 7, with one peak at 7 system calls.
The VVC encoder also has an identical behaviour, but this time it has a huge change in the last peak which registers a maximum value of 40.9 system calls.

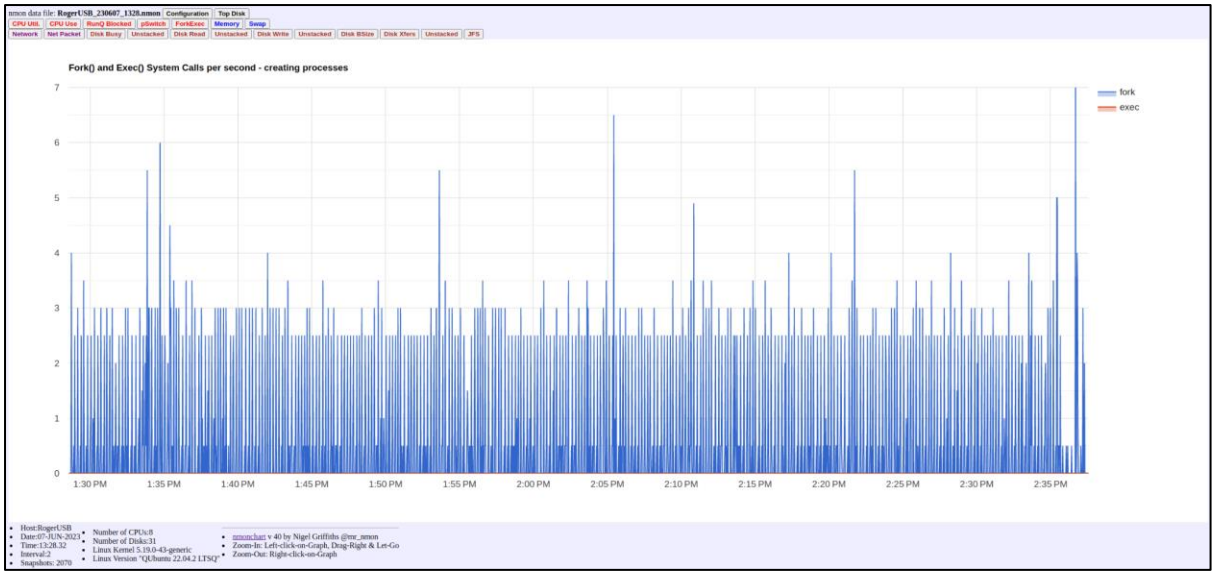

Figure 5.17: System calls per second on 4k resolution codification with EVC.

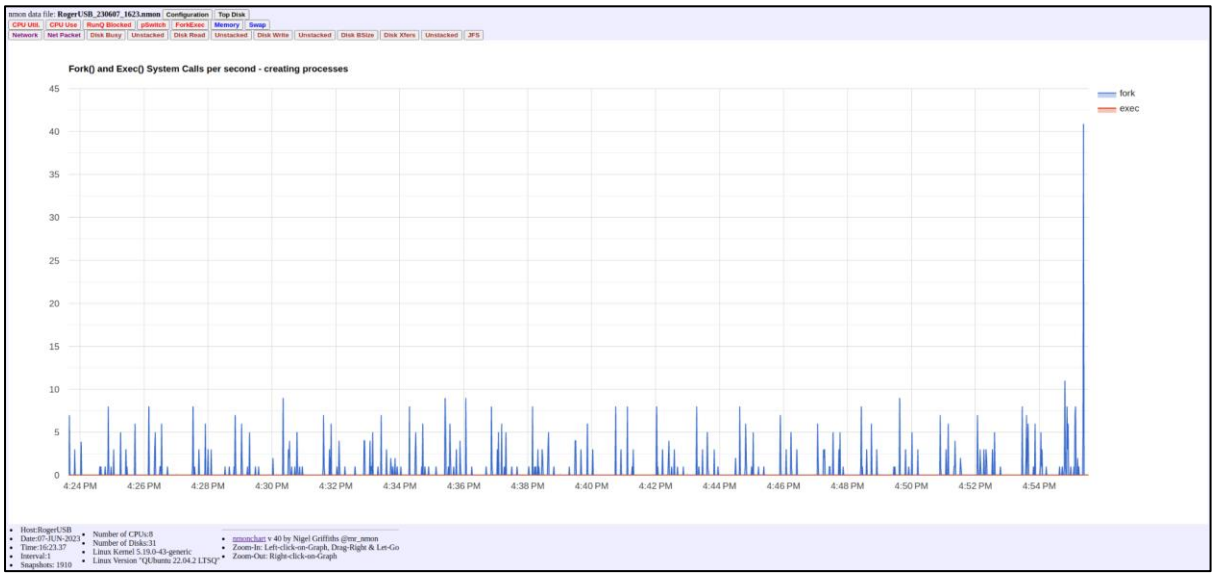

Figure 5.18: System calls per second on 4k resolution codification with VVC.

5.1.2.5 RAM memory usage

In Figures 5.19 and 5.20, we can see the Memory usage of both codecs. For the EVC encoder, as in the previous resolution, the total RAM memory available is 32041.9 MB. The cached memory has a constant value of 26000 MB. The active memory has a starting value of nearly 10000 MB, and it drops to 6500 MB at the end of the process. Free memory and buffers are both around only 200 MB and the inactive memory goes from 20000 MB to 24000 MB at the end of the encoding process.

With the VVC encoder, it has a different RAM memory usage conduct compared to 8k resolution. In this case, the memory usage is quite more stable throughout the whole process. The cached memory has constant values around 20000MB-23000 MB. The active memory has a starting value of 7000 MB and then reaches up to 10000 MB. Free memory starts at 5000 MB

and then it gradually decreases to 200 MB, while buffers memory is stable at 200MB the whole process.

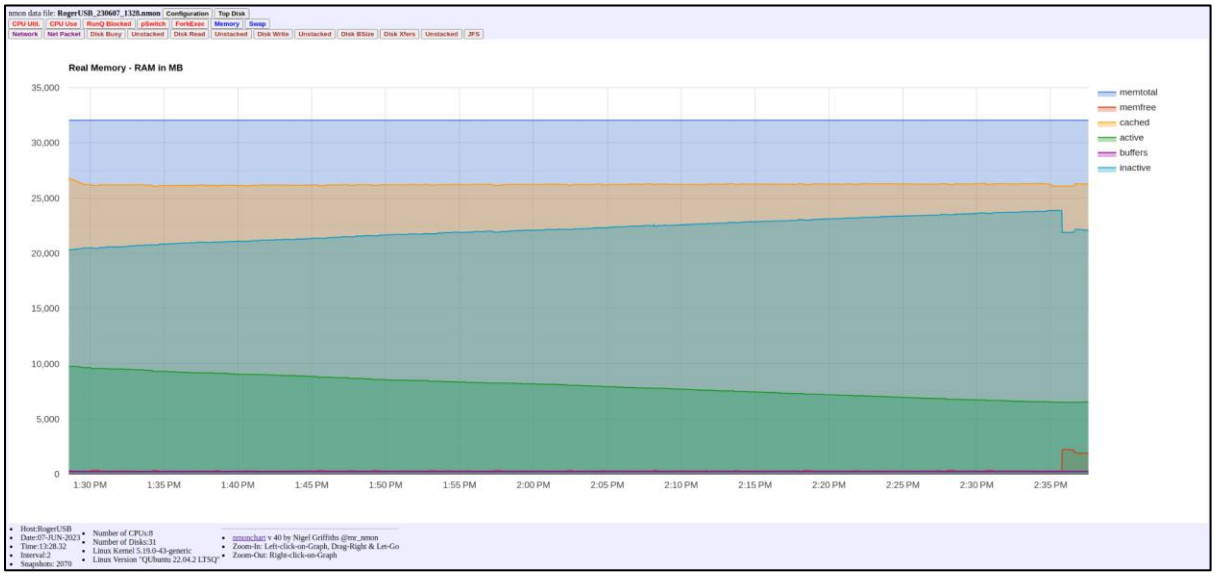

Figure 5.19: RAM memory usage on 4k resolution codification with EVC.

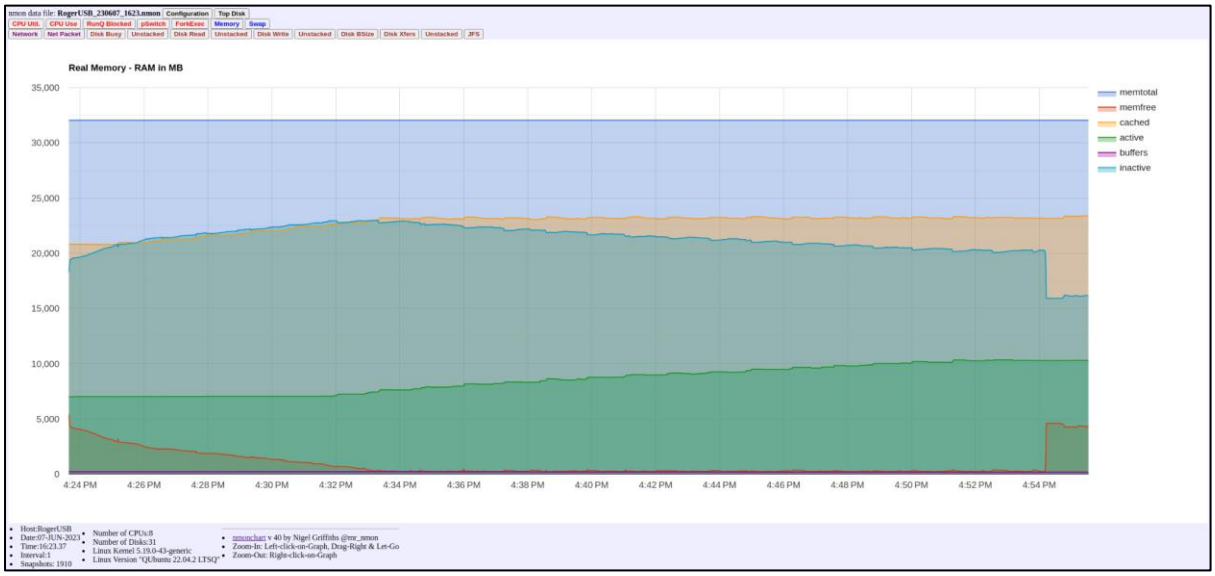

Figure 5.20: RAM memory usage on 4k resolution codification with VVC.

5.1.2.6 Time execution

EVC completed the coding process in 4007.898 s, and the VVC completed it in 1834.664 s.

5.1.3 Analysis for HD resolution

5.1.3.1 CPU

In images Figures 5.21 and 5.22, we can see the CPU utilisation Percentages of both codecs. With the Essential Video Coding standard, it obtained the exact same performance as the other resolutions. There is a constant User CPU usage of around 13%, with a maximum value of 20% in two instant moments.
For the VVC encoder, this result is significantly different compared to the higher resolutions. The main difference now is that close to half of the process time that the encoder needed to create the EVC file, the User CPU usage is around 40-45%, and the other half has the exact same maximum values around the 99% of the User CPU usage as the two other resolutions. This drop out in CPU usage is due to the lower image resolution.

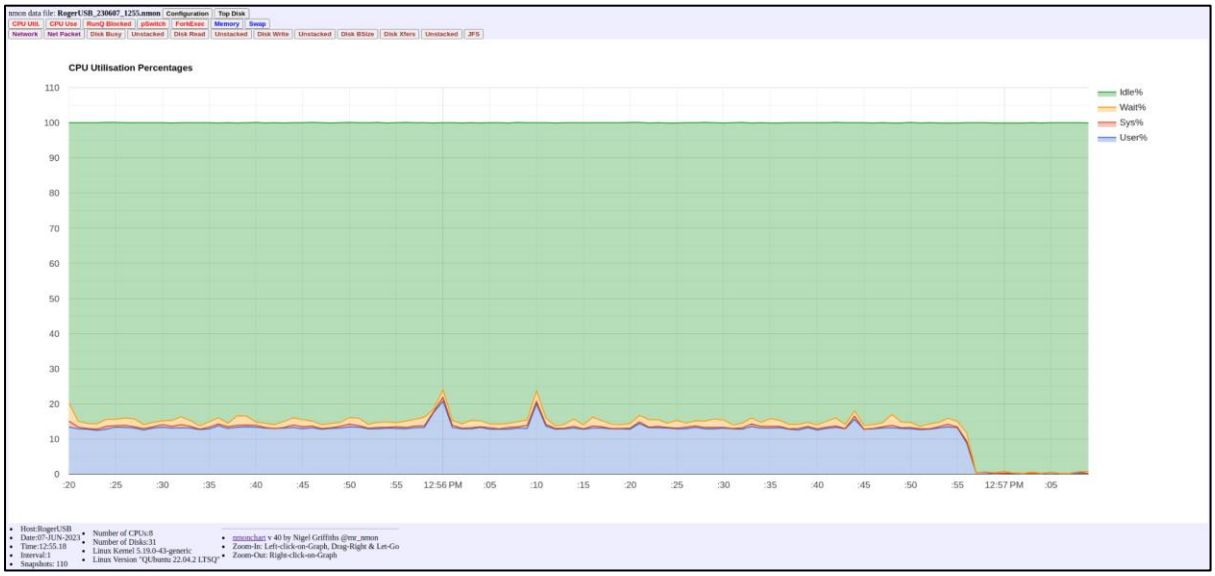

Figure 5.21: CPU usage on HD resolution codification with EVC.

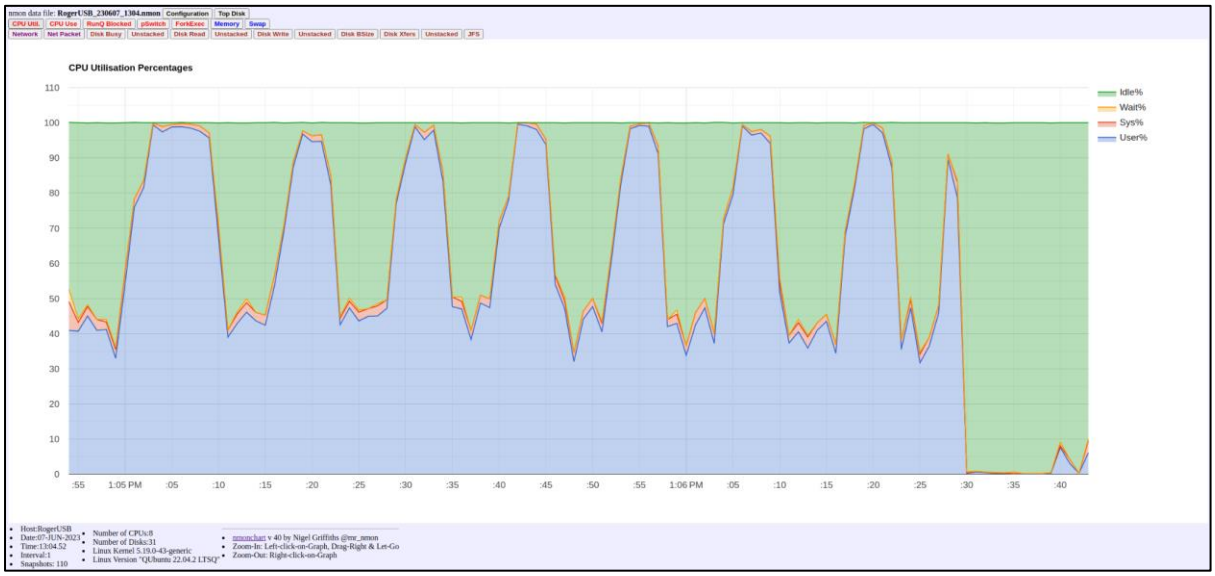

Figure 5.22: CPU usage on HD resolution codification with VVC.

5.1.3.2 Run queue

In Figures 5.23 and 5.24, we can see the Run queue in the number of processes of both codecs. The Essential Video Encoder behaves as the 8k resolution, with an average value of 2 running and 0 blocked processes with some peaks at 3 and 4 for running processes and some peaks at 1 for the blocked ones.
With the Versatile Video Coding standard, with HD resolution we have more fructual values for running processes, oscillating between 3 and 4 to 9 and 10, with a maximum value of 13. There is only one blocked process, probably due to the lower number of frames, duration and the lower resolution.

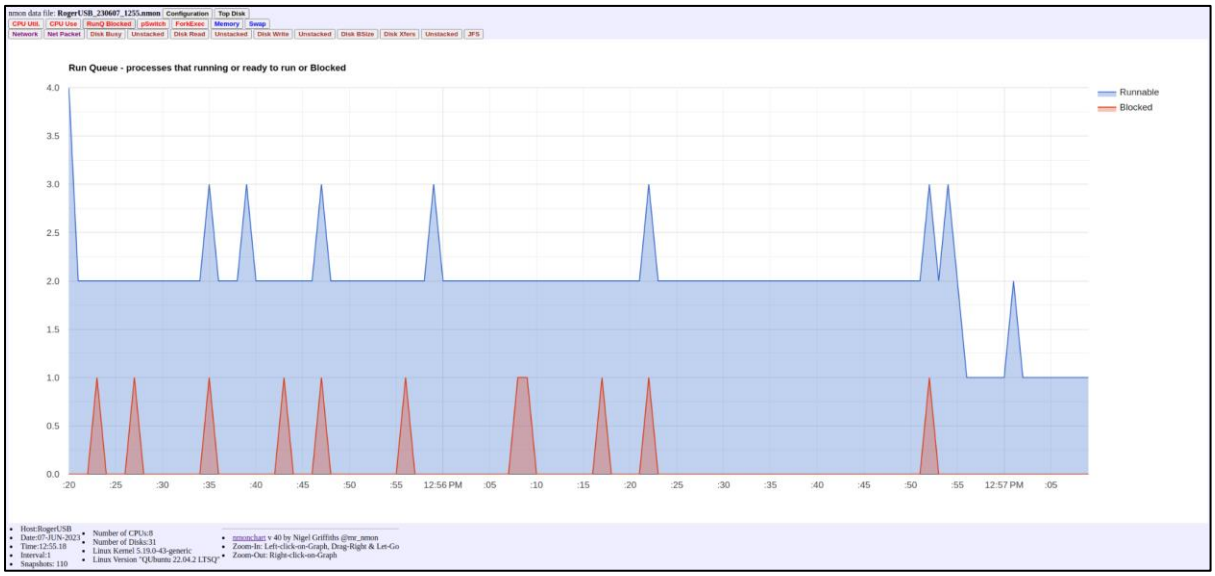

Figure 5.23: Run queue on HD resolution codification with EVC.

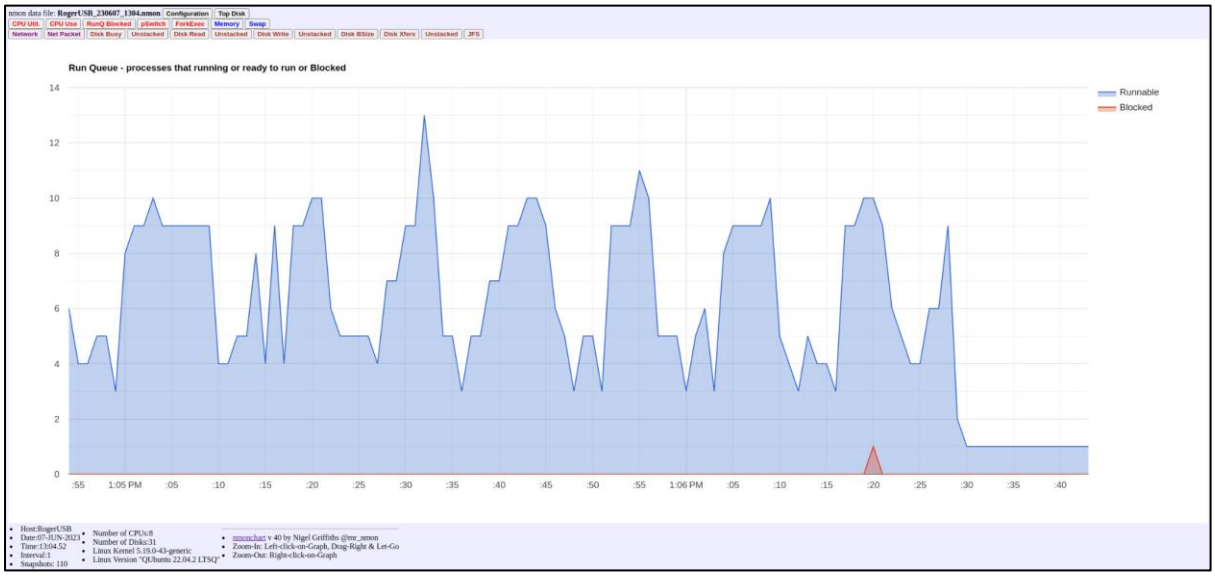

Figure 5.24: Run queue on HD resolution codification with VVC.

5.1.3.3 Process Switch

In Figures 5.25 and 5.26, we can see the Process switches of both codecs. Unlike previous resolutions, EVC now does not have such a difference between the whole process and the final peak. It fluctuates between 1500-2000 to 8000-8800 process switches per second, while the final peak only reaches 10000 switches per second.
The VVC codec does look like prior resolutions, having a constant value of processes switches that oscillate between 1000 and 19000 switches per second. The final peak has a value of 78000 process switches per second.

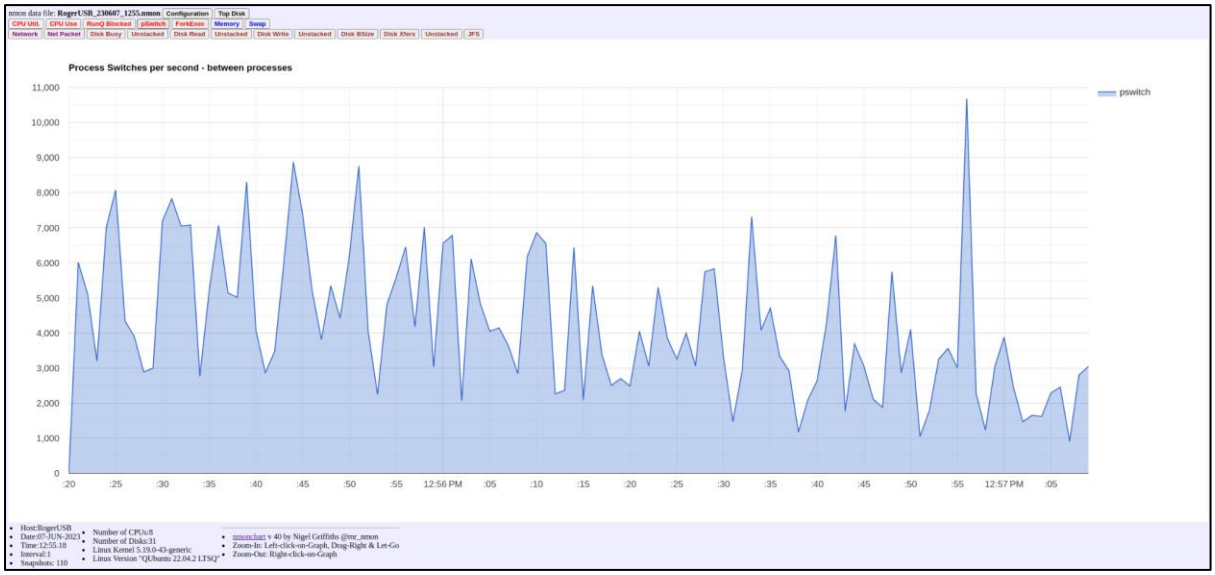

Figure 5.25: Process switch on HD resolution codification with EVC.

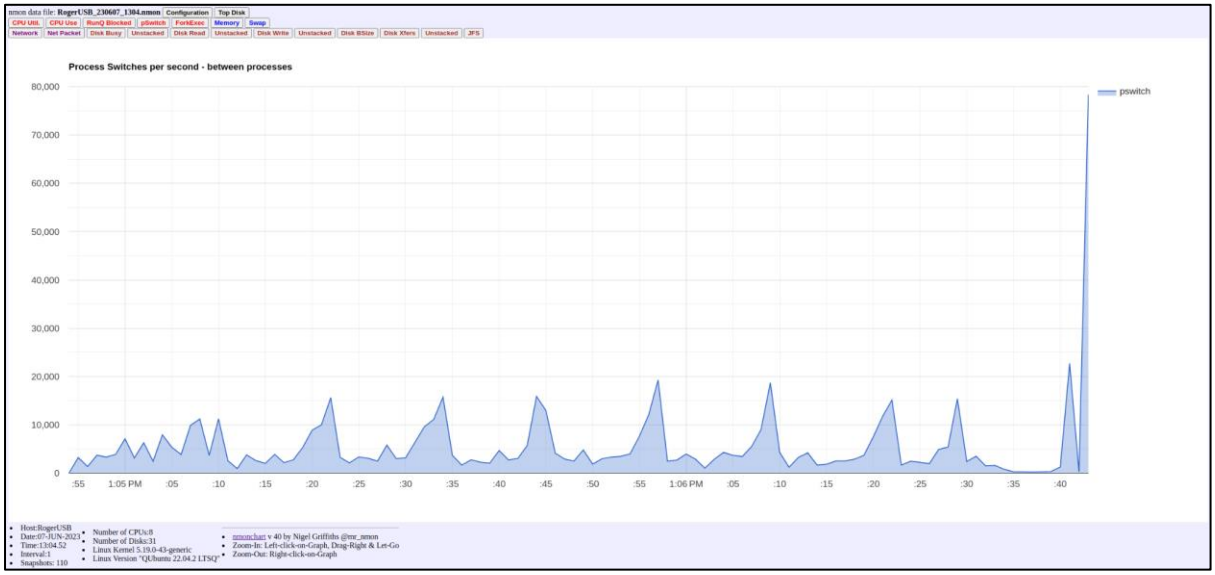

Figure 5.26: Process switch on HD resolution codification with VVC.

5.1.3.4 System calls per second

In Figures 5.27 and 5.28, we can see the System calls per second of both codecs. They both perform very similar graphics. With the same periodicity and same values in fork system calls per second.

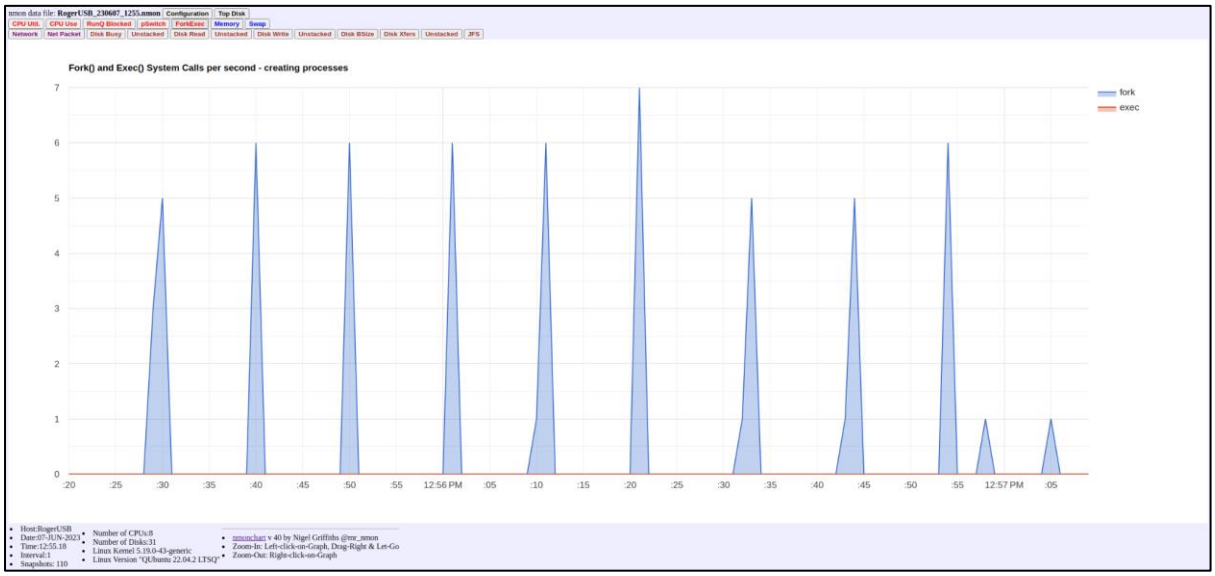

Figure 5.27: System calls per second on HD resolution codification with EVC.

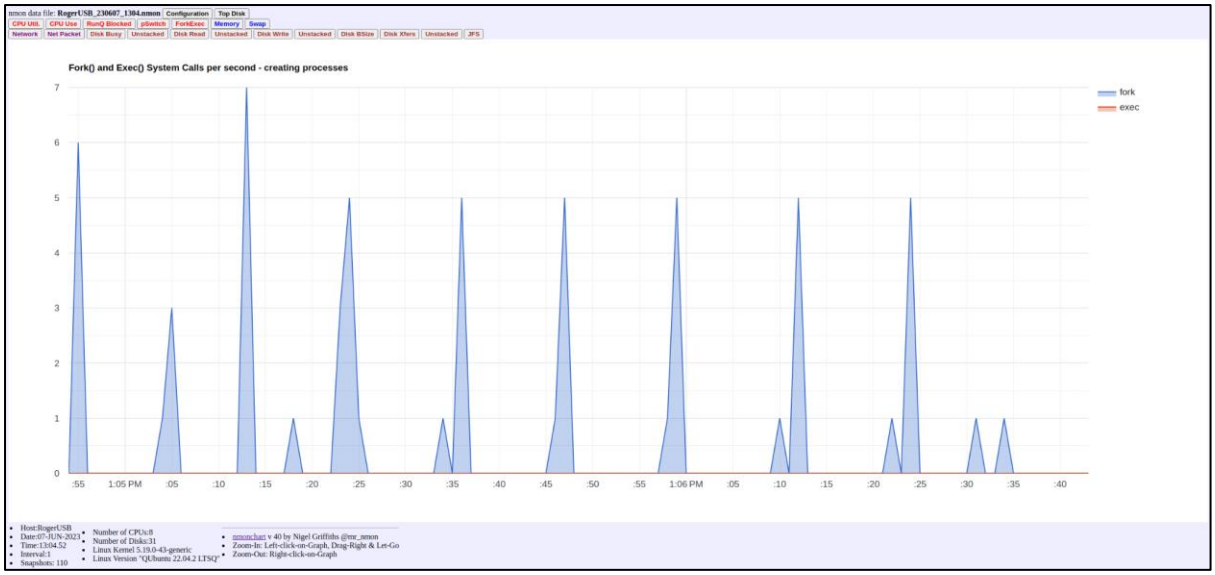

Figure 5.28: System calls per second on HD resolution codification with VVC.

5.1.3.5 RAM memory usage

In Figures 5.29 and 5.30, we can see the Memory usage of both codecs. For the EVC encoder, as in the previous resolutions, the total RAM memory available is 32041.9 MB. The cached memory has a constant value of 28000 MB. The active memory for this resolution also has a

constant value of 10000 MB. Free memory and buffers are both around only 200 MB and the inactive memory has a constant value of 20000 MB as well.
The VVC performance for HD resolution is quite the same as the EVC. The cached memory has constant values around 26000MB. The active memory also has constant behaviour at 10000MB and then, consequently, the inactive memory is also constant with values close to 20000MB. Free memory has two different constant values 250Mb and 650MB, while buffers memory is stable at 250MB the whole process.

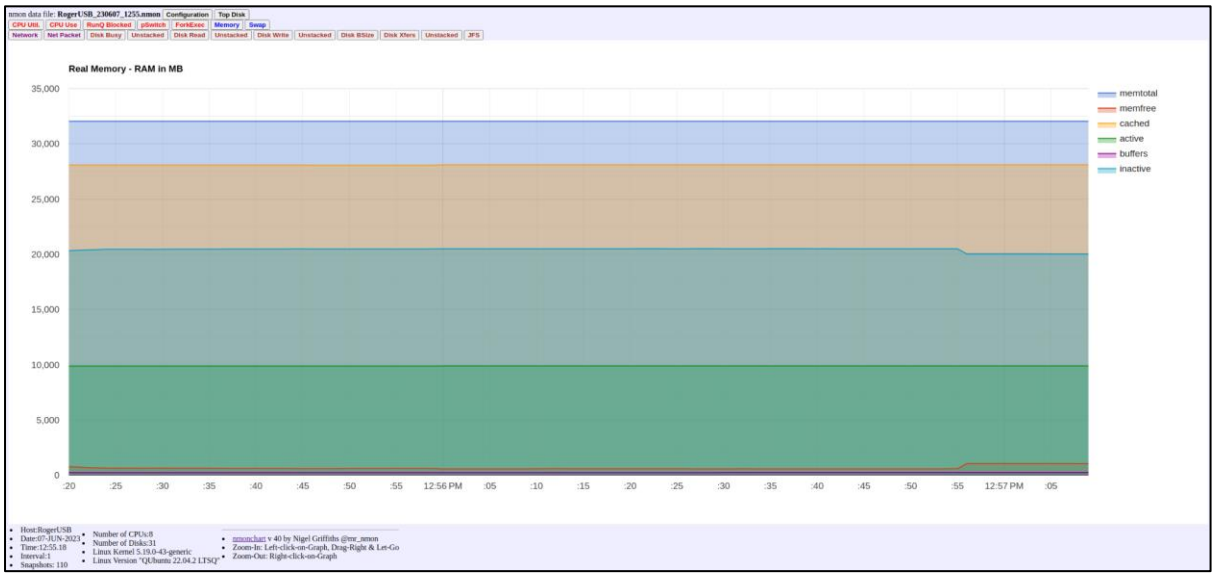

Figure 5.29: RAM memory usage on HD resolution codification with EVC.

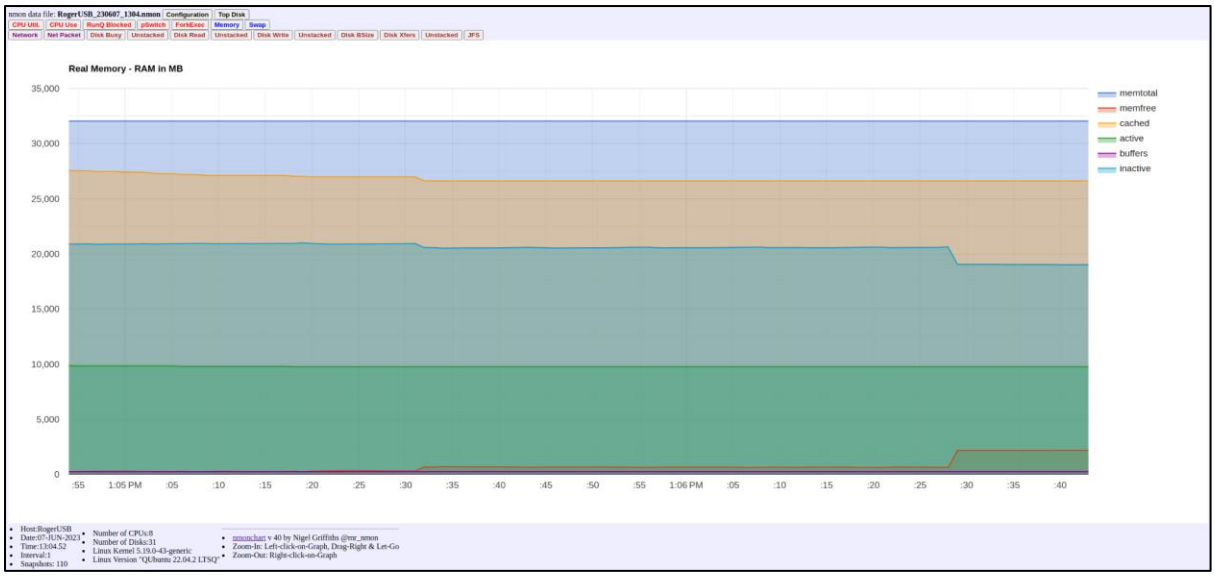

Figure 5.30: RAM memory usage on HD resolution codification with VVC.

5.1.2.6 Time execution

EVC completed the coding process in 96.5 s, and the VVC completed it in 96.488 s.

5.2 Electrical Consumption

After each encoding procedure, explained in 4.5.1 and 4.5.2 respectively, I took images of every process electrical consumption in kWh. All images are in Annex B. In the following table we can see the results.

Codecs\Resolutions	8k	4k	HD
EVC	0.092 kWh	0.122 kWh	0.003 kWh
VVC	0.044 kWh	0.079 kWh	0.004 kWh

Table 5.1: Electrical consumptions.

The electrical consumption results obtained executing each encoding video process consecutively are shown in the next figures. It clearly shows that, as with the results obtained with each codec being codificated alone, the VVC encoding procedure has a much higher values, with an unstable behaviour while the EVC encoder has a much lower values, with a nearly constant behaviour.

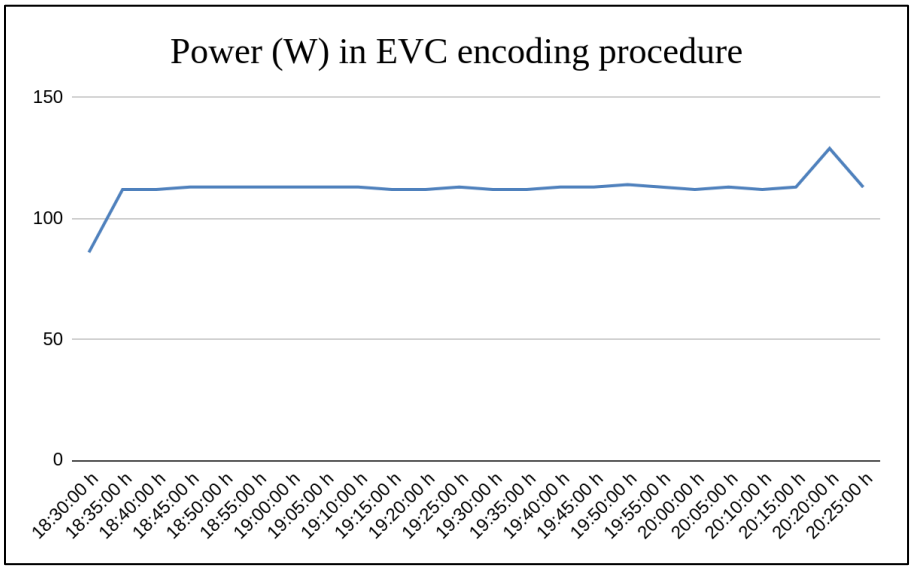

Figure 5.31: Power (W) in EVC encoding procedure.

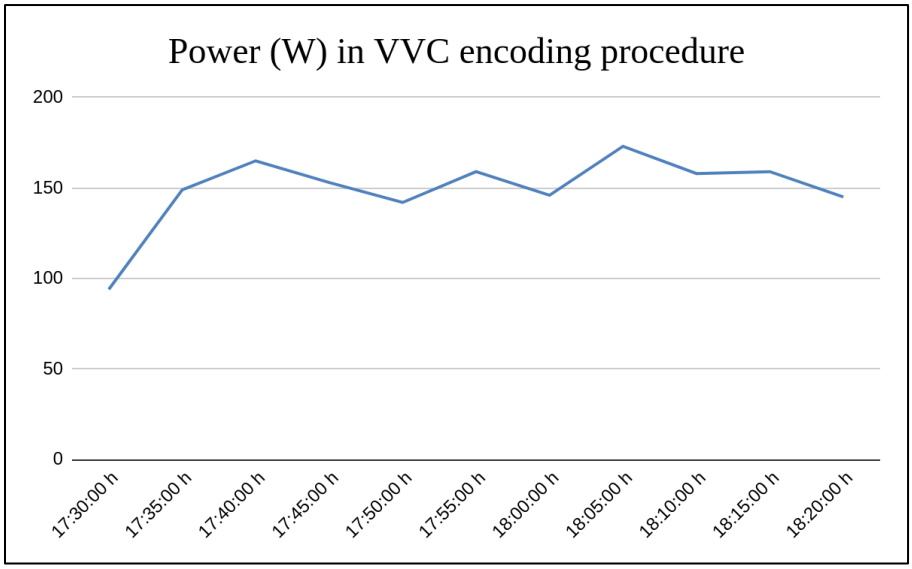

Figure 5.32: Power (W) in VVC encoding procedure.

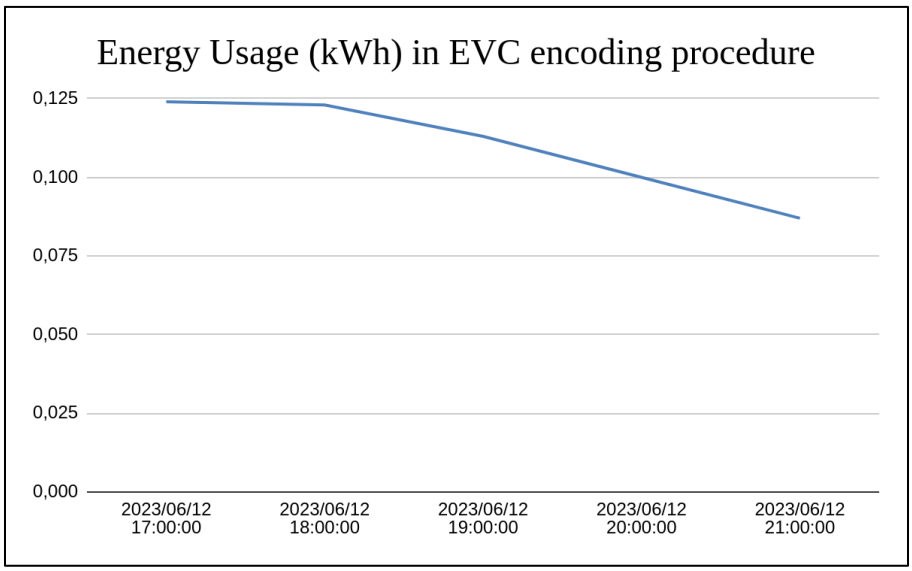

Figure 5.33: Energy Usage (kWh) in EVC encoding procedure.

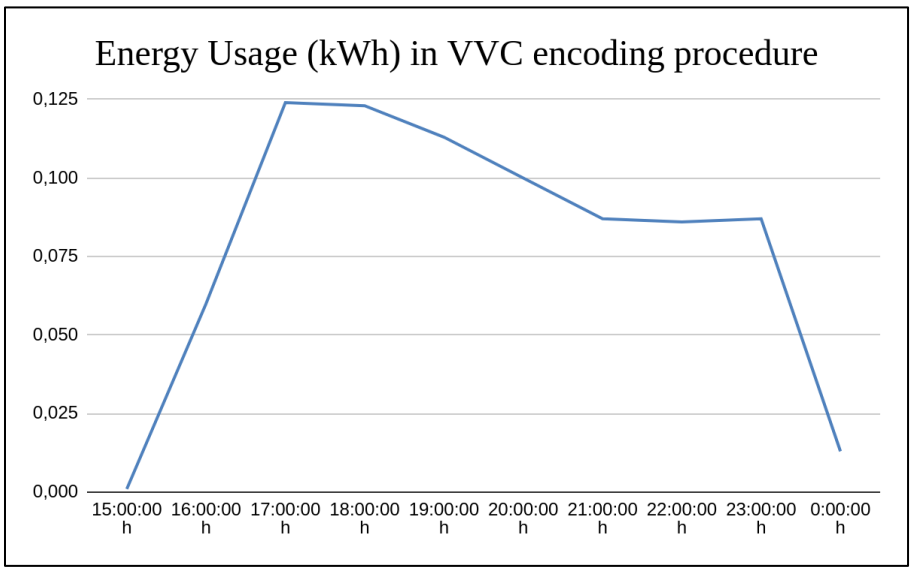

Figure 5.34: Energy Usage (kWh) in VVC encoding procedure.

5.3 Carbon footprint Calculation

As it has been explained in chapter 3, the carbon footprint is the totality of greenhouse gasses emitted directly or indirectly through the activity we want to measure. Therefore, I will compute a carbon footprint for each individual video process executed throughout this study. The value selection criteria has been explained on section 3.2.

I will calculate the total emissions of CO_2e in kg with the two most common factor mix elec. kg CO_2e/kWh of the five most important providers in Spain, as it is shown in Figure B8, from the Annex B.

The Tables 5.2 and 5.3 show the final values of the computation of the Carbon footprint of the experiments performed.

Codecs\Resolutions	**8k**	**4k**	**HD**
EVC	0.0276 kg CO_2e	0.0366 kg CO_2e	0.0009 kg CO_2e
VVC	0.0132 kg CO_2e	0.0237 kg CO_2e	0.0012 kg CO_2e

Table 5.2: Carbon footprint with Factor Mix elec. kg CO_2e/kWh of 0.3.

Codecs\Resolutions	**8k**	**4k**	**HD**
EVC	0.02392 kg CO_2e	0.03172 kg CO_2e	0.00078 kg CO_2e
VVC	0.01144 kg CO_2e	0.02054 kg CO_2e	0.00104 kg CO_2e

Table 5.3: Carbon footprint with Factor Mix elec. kg CO_2e/kWh of 0.26.

CHAPTER 6

CONCLUSIONS

6.1 Conclusions computational cost

For the CPU usage of both codecs, the results indicate that the VVC codec requires much more power and CPU deployment compared to EVC in high resolutions. In terms of CPU optimization, it is clearly shown that VVC demands too much, at least for the software of the local environment where the experiments have been executed, since with the three video resolutions the CPU usage was at 99%, which is clearly inadvisable. With EVC instead, it has performed with a much more stable CPU usage.
For HD resolution, very interesting data was obtained, since they performed with the same differences as the previous resolutions in CPU usage, but they lasted the same in terms of total encoding time.

The run queue graphics showed that, as it could be expected, the EVC has much fewer running processes for all resolutions even though it has the same blocked processes as VVC.
On the other hand, the VVC encoder has much higher values in this feature.

For the Process switches, has a similar interpretation as the other analysed characteristics. VVC requires much higher values compared to EVC. With that being said, they have a similar behaviour, which doesn't happen in other aspects.

For system calls per second they have similar behaviours, even though EVC has more peaks, but VVC has higher values on them.

In the analysis of the Memory usage, EVC has a constant behaviour in all resolutions, while VVC has higher performance at the beginning and then starts falling in cache memory usage after a few minutes. For the 8k resolution they have similar values. With 4k resolution, VVC has more stable values and finally, for HD, they have almost the same behaviour.

In terms of process swap in memory, there are no swaps in any video resolution, so it didn't bring any information.

The time execution is a very important characteristic of the encoding procedures. In general, the VVC codec had a much faster performance, except in HD resolution. The encoding process for VVC was 65.1% faster compared to EVC, and with 4k was 54.2% faster too. Nonetheless, the HD resolution provided a very interesting analysis, since they lasted exactly the same, but the VVC had a very high computational cost.

6.2 Interpretation of carbon footprint results

The results of the Carbon footprint computations are very clear. As we can see on the outputs obtained in section 5.3, the EVC had a major volume of greenhouse gasses emitted indirectly through the codification process compared to VVC. For the higher resolution, 8k, the VVC encoder had a 52.17% less carbon footprint and in 4k resolution had a 35.246% less.

Notwithstanding, for the lowest resolution and encoding time, the VVC obtained a minimally higher footprint.

It is necessary to say that these results are with the legislation implemented in Spain, and as it is explained in chapter 3, each country has his own carbon footprint definition and calculation methods.

6.3 Evaluation of codecs with results

In summary, with all the data collection and the analysis obtained, it is safe to state that for high resolutions the most optimal and good for the environment codec is the VVC and for low resolutions, the EVC. Even though, the VVC codec demands a most sophisticated and capable computational characteristics, requesting much more CPU usage, much more processes executed per second, process switches and system calls per second.
The EVC encoder has a very stable performance in all resolutions, therefore will be much more efficient for videos with a low number of frames and lower resolutions.
With the Carbon footprint the same happens: VVC is better for high resolutions and EVC for low resolutions.
Nonetheless, each codec has different goals and both will be very useful in the near future. They both can coexist and be used in different working areas.

This study has been implemented with the only existing software to process these new video codec standards. It cannot be dismissed that in the near future, there will be available new software which can optimize the internal processes in the computer.

6.4 Final assessment

With the conclusion of this study, I think it would be necessary to emphasize certain aspects necessary to understand the context of this work and the possible continuations of it.

Although obtaining contrasted results, these values are not conclusive at all. It would be required to do much more analysis and comparisons with other parameters in terms of video duration, frames, resolutions, between others, to see how codecs behave and the correlation that exists between each one of these factors and the carbon footprint of these processes.

Another esteemed point that must be considered is that all these encoding procedures have been executed on a standard local environment and that these results only are suitable in these circumstances. In fact, as it has been mentioned before, VVC demands a sophisticated computational environment and in my experience during the encoding procedures, the computer overheated due to this reason. Consequently, for big infrastructures it would require a good refrigeration system which could bring with a higher carbon footprint.

The carbon footprint is not a specific measure assigned to each encoder as it has been explained in this project, but a powerful tool to measure processes. It could be useful being implemented in different big companies which could compare different intervals of time comparing different codecs, and that would imply a more suitable and specific long-term results.

Therefore, it would be interesting to implement other studying factors as:

- Behaviour on different computers
- Behaviour with different dataset of videos.
- To calculate the carbon footprint of organizations on a long-term and then compare between codecs, having similar production on comparing processes.
- Provide a more efficient and precise studying methodology.

CHAPTER 7

GITHUB IMPLEMENTATION

With the purpose of providing the possibility to everyone to do this calculations, I have created a GitHub repository named CarbonFootprintComputeOfEVCandVVCcodecs[18]. This includes an executable script with the needed commands to compute the video codification and the formulas to compute the carbon footprint.

[18] https://github.com/RogerFernandezHermoso/CarbonFootprintComputeOfEVCandVVCcodecs

Bibliography

[1] Ozer, J. (2011, March 21). *What is a codec?*. Streaming Media Magazine. https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=74487

[2] *Video compression*. Haivision. (2023, February 20). https://www.haivision.com/glossary/compression/#:~:text=Video%20compression%20is%20the%20process,the%20size%20of%20the%20data

[3] Pertsova, C. C. (2007). *Ecological Economics Research trends*. Nova Science.

[4] H.261: Video codec for audiovisual services at P X 384 kbit/s. (s/f) Itu.int. https://www.itu.int/rec/T-REC-H.261-198811-S/en

[5] Chen, T. (n.d.). *VIDEO CODING STANDARDS FOR MULTIMEDIA COMMUNICATION*. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4d47cca33973f51da2740a6b78ef81dbf2c96325.

[6] Watkinson, J. (2013). *The MPEG handbook: MPEG-1, MPEG-2, MPEG-4*. Focal Press, Taylor & Francis Group.

[7] Rijkse, K. (1996). *H.263: video coding for low-bit-rate communication*. IEEE. Retrieved from https://ieeexplore.ieee.org/document/556485/authors#authors.

[8] Vranješ, M., Rimac-Drlje, S., & Žagar, D. (n.d.). *Subjective and Objective Quality Evaluation of the H.264/AVC Coded Video* . IEEE.

[9] Kalva, H., & Lee, J.-B. (2008). *The VC-1 and H.264 Video Compression Standards for broadband video services*. Springer.

[10] Uhrina, M., Bienik, J., & Vaculik, M. (2016). (rep.). *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7512044*. IEEE. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7512044.

[11] Keng Tan, T., Weerakkody, R., Mrak, M., Ramzan, N., Baroncini, V., Ohm, J.-R., & J. Sullivan, G. (2016). *Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance*. IEEE. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7254155.

[12] HAN, J., LI, B., MUKHERJEE, D., CHIANG, C.-H., GRANGE, A., CHEN, C., SU, H., PARKER, S., DENG, S., JOSHI, U., CHEN, Y., WANG, Y., WILKINS, P., XU, Y., & BANKOSKI, J. (9AD). *A Technical Overview of AV1*. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9363937.

[13] Convenor of ISO/IEC JTC 1/SC 29/AG 03. (2021). *White paper on Essential Video Coding (EVC)*. Retrieved from https://www.mpeg.org/whitepapers/.

[14] MPEG. (n.d.). https://www.mpeg.org/

[15] ISO/IEC JTC 1/SC 29/AG 3. (2021a). *White paper on Versatile Video Coding (VVC)*. Retrieved from https://www.mpeg.org/whitepapers/.

[16] Bross, B., Wang, Y.-K., Ye, Y., Liu, S., Chen, J., J. Sullivan, G., & Ohm, J.-R. (2021). *Overview of the Versatile Video Coding (VVC) Standard and Its Applications*. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9503377.

[17] WACKERNAGEL, M. (1988). *ECOLOGICAL FOOTPRINT AND APPROPRIATED CARRYING CAPACITY: A TOOL FOR PLANNING TOWARD SUSTAINABILITY*. Retrieved from https://open.library.ubc.ca/soa/cIRcle/collections/ubctheses/831/items/1.0088048.

[18] IPCC, 2022: Annex I: Glossary [van Diemen, R., J.B.R. Matthews, V. Möller, J.S. Fuglestvedt, V. Masson-Delmotte, C. Méndez, A. Reisinger, S. Semenov (eds)]. In IPCC, 2022: Climate Change 2022: Mitigation of Climate Change. Contribution of Working Group III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [P.R. Shukla, J. Skea, R. Slade, A. Al Khourdajie, R. van Diemen, D. McCollum, M. Pathak, S. Some, P. Vyas, R. Fradera, M. Belkacemi, A. Hasija, G. Lisboa, S. Luz, J. Malley, (eds.)]. Cambridge University Press, Cambridge, UK and New York, NY, USA. doi: 10.1017/9781009157926.020
Retrieved from https://www.ipcc.ch/report/ar6/wg3/downloads/report/IPCC_AR6_WGIII_Annex-I.pdf

[19] UNITED NATIONS. (1992). *UNITED NATIONS FRAMEWORK CONVENTION ON CLIMATE CHANGE*. Retrieved from https://unfccc.int/files/essential_background/background_publications_htmlpdf/application/pdf/conveng.pdf.

[20] United Nations Climate Change. (n.d.). Unfccc.int. https://unfccc.int/node/28571

[21] United Nations. (n.d.). *Marking the Kyoto Protocol's 25th Anniversary*. United Nations. https://www.un.org/en/climatechange/marking-kyoto-protocol%E2%80%99s-25th-anniversary#:~:text=In%202015%2C%20however%2C%20countries%20agreed,effectively%20replaced%20the%20Kyoto%20Protocol

[22] McGrath, M. (2016, March 31). *Paris Climate Treaty: "Significant Step" as US and China agree to sign*. BBC News. https://www.bbc.com/news/science-environment-35935756?ocid=global_bbccom_email_31032016_top%2Bnews%2Bstories

[23] U.S. Department of State. (2021, February 26). *The United States officially rejoins the Paris Agreement - United States Department of State*. U.S. Department of State. https://www.state.gov/the-united-states-officially-rejoins-the-paris-agreement/

[24] Lungley, H. (2015). (rep.). *Ecological and Carbon Footprints of Wales*. Stockholm Environment Institute and GHD. Retrieved from https://www.gov.wales/sites/default/files/publications/2019-04/ecological-and-carbon-footprint-of-wales-report.pdf.

[25] Ministerio para la Transición Ecológica. (n.d.). *HUELLA DE CARBONO DE UNA ORGANIZACIÓN*. Retrieved from https://www.miteco.gob.es/es/cambio-climatico/temas/mitigacion-politicas-y-medidas/huellacarbono_conceptosbasicos_tcm30-478999.pdf.

[26] Ministerio para la Transición Ecológica y el Reto Demográfico. (n.d.). Calculadoras. https://www.miteco.gob.es/es/cambio-climatico/temas/mitigacion-politicas-y-medidas/calculadoras.aspx

[27] CNMC. (n.d.). *Ranking de las eléctricas en España*. Retrieved from https://e00-expansion.uecdn.es/opinion/documentosWeb/2019/01/25/electricas_consumidores.pdf.

[28] IdeaCentre AIO Y910 (27"). *Lenovo*. Retrieved from https://www.lenovo.com/hk/en/desktops-and-all-in-ones/ideacentre/ideacentre-aio-y900-series/AIO-Y910-27/p/FFICFY90253?orgRef=https%253A%252F%252Fwww.google.com%252F#.

[29] Fernández, Y. (2019, May 26). *Cómo instalar linux en un disco duro externo para utilizarlo Cuando y Donde quieras*. Linux: instalar una distro GNU/Linux en disco duro externo. https://www.xataka.com/basics/como-instalar-linux-disco-duro-externo

[30] Filmyk. (2021). *Como Instalar Linux en disco duro externo USB 3.0 SSD - Ubuntu 20.04. YouTube*. YouTube. Retrieved June 16, 2023, from https://www.youtube.com/watch?v=yrjaQvYo_6E.

[31] YouTube. (2019). *Instalar Ubuntu en un Disco Duro Externo (UEFI/GPT). YouTube*. Retrieved June 16, 2023, from https://www.youtube.com/watch?v=HnRfTNgq3gI.

[32] Lenovo. (2016). ideacentre All-In-One Y910 Computer Hardware Maintenance Manual. https://download.lenovo.com/consumer/desktop_pub/y910_hmm_20160829.pdf

[33] Griffiths, N. (n.d.). *Documentation*. nmon for Linux | Site / Documentation. https://nmon.sourceforge.net/pmwiki.php?n=Site.Documentation

[34] YouTube. (2016). *AIX in Focus: nmonchart to graph your nmon data files (AIX & Linux). YouTube*. Retrieved June 16, 2023, from https://www.youtube.com/watch?v=5P4neOqoCTo&list=PLKQlFnmiWVyc89mDmaMoouao8CZcTp8YP&index=5.

ANNEX A

Terminal outputs of Process and Scripts used.

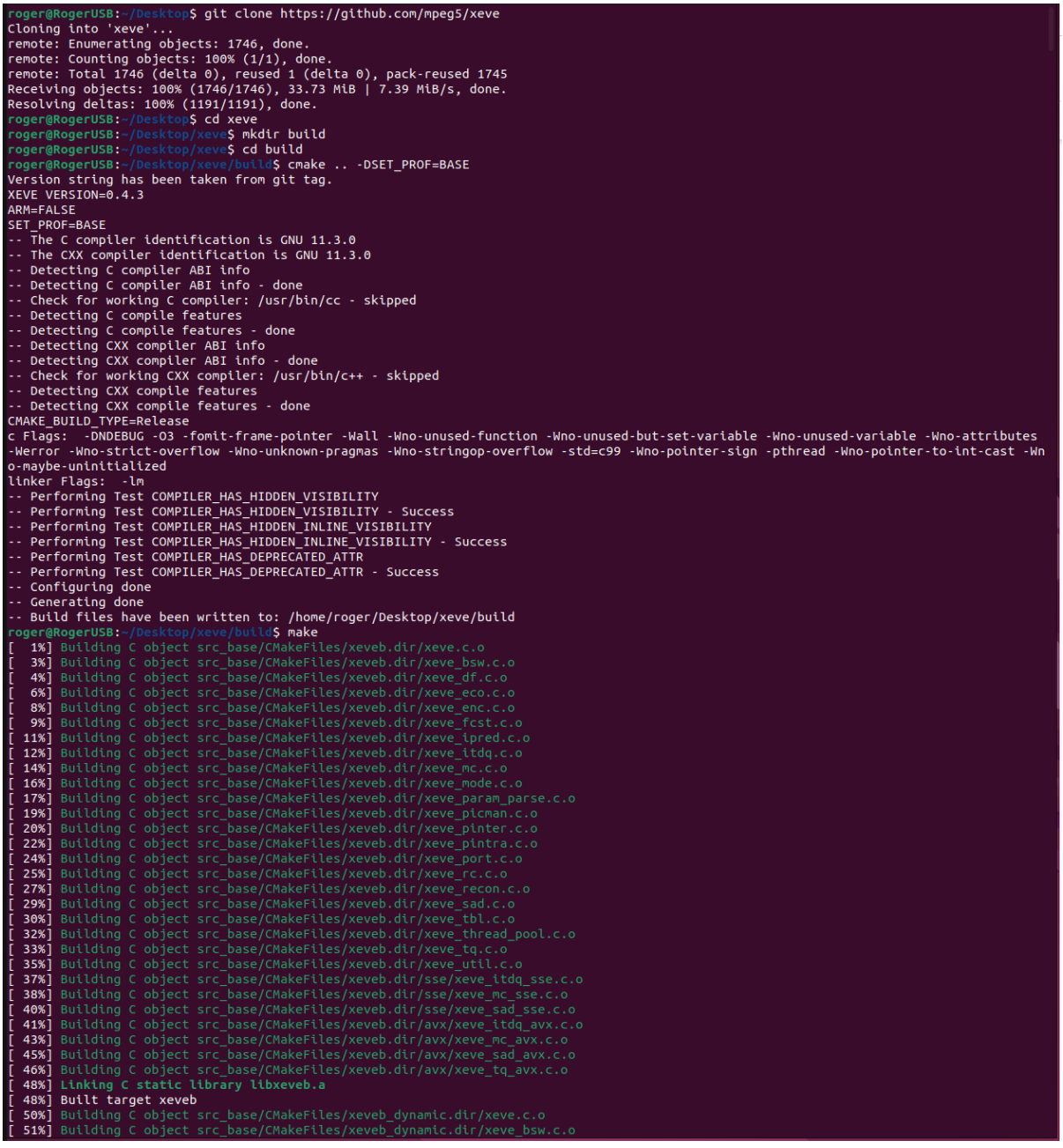

Figure A1: terminal procedure to install EVC encoder Part 1.

```
[ 48%] Built target xeveb
[ 50%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve.c.o
[ 51%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_bsw.c.o
[ 53%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_df.c.o
[ 54%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_eco.c.o
[ 56%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_enc.c.o
[ 58%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_fcst.c.o
[ 59%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_ipred.c.o
[ 61%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_itdq.c.o
[ 62%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_mc.c.o
[ 64%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_mode.c.o
[ 66%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_param_parse.c.o
[ 67%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_plcman.c.o
[ 69%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_pinter.c.o
[ 70%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_pintra.c.o
[ 72%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_port.c.o
[ 74%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_rc.c.o
[ 75%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_recon.c.o
[ 77%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_sad.c.o
[ 79%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_tbl.c.o
[ 80%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_thread_pool.c.o
[ 82%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_tq.c.o
[ 83%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/xeve_util.c.o
[ 85%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/sse/xeve_itdq_sse.c.o
[ 87%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/sse/xeve_mc_sse.c.o
[ 88%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/sse/xeve_sad_sse.c.o
[ 90%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/avx/xeve_itdq_avx.c.o
[ 91%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/avx/xeve_mc_avx.c.o
[ 93%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/avx/xeve_sad_avx.c.o
[ 95%] Building C object src_base/CMakeFiles/xeveb_dynamic.dir/avx/xeve_tq_avx.c.o
[ 96%] Linking C shared library ../lib/libxeveb.so
[ 96%] Built target xeveb_dynamic
[ 98%] Building C object app/CMakeFiles/xeveb_app.dir/xeve_app.c.o
[100%] Linking C executable ../bin/xeveb_app
[100%] Built target xeveb_app
roger@RogerUSB:~/Desktop/xeve/build$ sudo make install
[sudo] password for roger:
Consolidate compiler generated dependencies of target xeveb
[ 48%] Built target xeveb
Consolidate compiler generated dependencies of target xeveb_dynamic
[ 96%] Built target xeveb_dynamic
Consolidate compiler generated dependencies of target xeveb_app
[100%] Built target xeveb_app
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/lib/xeveb/libxeveb.a
-- Installing: /usr/local/include/xeveb/xeve.h
-- Installing: /usr/local/lib/libxeveb.so.0.4
-- Installing: /usr/local/lib/libxeveb.so.0
-- Installing: /usr/local/lib/libxeveb.so
-- Up-to-date: /usr/local/include/xeveb/xeve.h
-- Installing: /usr/local/include/xeveb/xeve_exports.h
-- Installing: /usr/local/lib/pkgconfig/xeveb.pc
-- Installing: /usr/local/bin/xeveb_app
roger@RogerUSB:~/Desktop/xeve/build$ mkdir build
roger@RogerUSB:~/Desktop/xeve/build$ cd build
roger@RogerUSB:~/Desktop/xeve/build/build$ cmake
Usage

  cmake [options] <path-to-source>
  cmake [options] <path-to-existing-build>
  cmake [options] -S <path-to-source> -B <path-to-build>

Specify a source directory to (re-)generate a build system for it in the
current working directory.  Specify an existing build directory to
re-generate its build system.

Run 'cmake --help' for more information.

roger@RogerUSB:~/Desktop/xeve/build/build$ cmake ..
Version string has been taken from git tag.
XEVE VERSION=0.4.3
ARM=FALSE
SET_PROF=BASE
CMAKE_BUILD_TYPE=Release
```

Figure A2: terminal procedure to install EVC encoder Part 2.

```
ARM=FALSE
SET_PROF=BASE
CMAKE_BUILD_TYPE=Release
c Flags:  -DNDEBUG -O3 -fomit-frame-pointer -Wall -Wno-unused-function -Wno-unused-but-set-variable -Wno-unused-variable -Wno-attributes
-Werror -Wno-strict-overflow -Wno-unknown-pragmas -Wno-stringop-overflow -std=c99 -Wno-pointer-sign -pthread -Wno-pointer-to-int-cast -Wn
o-maybe-uninitialized
linker Flags:  -lm
-- Configuring done
-- Generating done
-- Build files have been written to: /home/roger/Desktop/xeve/build
roger@RogerUSB:~/Desktop/xeve/build/build$ make
make: *** No targets specified and no makefile found.  Stop.
roger@RogerUSB:~/Desktop/xeve/build/build$ cd build
bash: cd: build: No such file or directory
roger@RogerUSB:~/Desktop/xeve/build/build$ cmake ..
Version string has been taken from git tag.
XEVE VERSION=0.4.3
ARM=FALSE
SET_PROF=BASE
CMAKE_BUILD_TYPE=Release
c Flags:  -DNDEBUG -O3 -fomit-frame-pointer -Wall -Wno-unused-function -Wno-unused-but-set-variable -Wno-unused-variable -Wno-attributes
-Werror -Wno-strict-overflow -Wno-unknown-pragmas -Wno-stringop-overflow -std=c99 -Wno-pointer-sign -pthread -Wno-pointer-to-int-cast -Wn
o-maybe-uninitialized
linker Flags:  -lm
-- Configuring done
-- Generating done
-- Build files have been written to: /home/roger/Desktop/xeve/build
roger@RogerUSB:~/Desktop/xeve/build/build$ make
make: *** No targets specified and no makefile found.  Stop.
roger@RogerUSB:~/Desktop/xeve/build/build$ sudo make install
make: *** No rule to make target 'install'.  Stop.
roger@RogerUSB:~/Desktop/xeve/build/build$ make package
make: *** No rule to make target 'package'.  Stop.
roger@RogerUSB:~/Desktop/xeve/build/build$ cd ..
roger@RogerUSB:~/Desktop/xeve/build$ ls
app             cmake_install.cmake      install_manifest.txt  xeveb.pc
bin             cmake_uninstall.cmake    lib                   xeve_exports.h
CMakeCache.txt  CPackConfig.cmake        Makefile
CMakeFiles      CPackSourceConfig.cmake  src_base
roger@RogerUSB:~/Desktop/xeve/build$ cd build
bash: cd: build: No such file or directory
roger@RogerUSB:~/Desktop/xeve/build$ make package
Consolidate compiler generated dependencies of target xeveb
[ 48%] Built target xeveb
Consolidate compiler generated dependencies of target xeveb_dynamic
[ 96%] Built target xeveb_dynamic
Consolidate compiler generated dependencies of target xeveb_app
[100%] Built target xeveb_app
Run CPack packaging tool...
CPack: Create package using DEB
CPack: Install projects
CPack: - Run preinstall target for: XEVE
CPack: - Install project: XEVE []
CPack: -   Install component: Runtime
CPack: -   Install component: Libraries
CPack: -   Install component: Development
CPack: Create package
-- CPACK_DEBIAN_PACKAGE_DEPENDS not set, the package will have no dependencies.
CPack: - package: /home/roger/Desktop/xeve/build/xeve-base-dev_0.4.3_amd64.deb generated.
CPack: - checksum file: /home/roger/Desktop/xeve/build/xeve-base-dev_0.4.3_amd64.deb.md5 generated.
CPack: - package: /home/roger/Desktop/xeve/build/xeve-base_0.4.3_amd64.deb generated.
CPack: - checksum file: /home/roger/Desktop/xeve/build/xeve-base_0.4.3_amd64.deb.md5 generated.
roger@RogerUSB:~/Desktop/xeve/build$ cpack -G "RPM" ..
CPack: Create package using RPM
CPack: Install projects
CPack: - Run preinstall target for: XEVE
CPack: - Install project: XEVE []
CMake Error at /home/roger/Desktop/xeve/build/cmake_install.cmake:60 (file):
  file failed to open for writing (Permission denied):

    /home/roger/Desktop/xeve/build/install_manifest.txt


CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ sudo cpack -G "RPM" ..
```

Figure A3: terminal procedure to install EVC encoder Part 3.

```
CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ sudo cpack -G "RPM" ..
CPack: Create package using RPM
CPack: Install projects
CPack: - Run preinstall target for: XEVE
CPack: - Install project: XEVE []
CPack: Create package
CMake Error at /usr/share/cmake-3.22/Modules/Internal/CPack/CPackRPM.cmake:822 (message):
  RPM package requires rpmbuild executable
Call Stack (most recent call first):
  /usr/share/cmake-3.22/Modules/Internal/CPack/CPackRPM.cmake:1968 (cpack_rpm_generate_package)


CPack Error: Error while execution CPackRPM.cmake
CPack Error: Problem compressing the directory
CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ cpack -G "DEB"
CPack: Create package using DEB
CPack: Install projects
CPack: - Run preinstall target for: XEVE
CPack: - Install project: XEVE []
CPack: -   Install component: Runtime
CPack: -   Install component: Libraries
CPack: -   Install component: Development
CPack: Create package
-- CPACK_DEBIAN_PACKAGE_DEPENDS not set, the package will have no dependencies.
CPack: - package: /home/roger/Desktop/xeve/build/xeve-base-dev_0.4.3_amd64.deb generated.
CPack: - checksum file: /home/roger/Desktop/xeve/build/xeve-base-dev_0.4.3_amd64.deb.md5 generated.
CPack: - package: /home/roger/Desktop/xeve/build/xeve-base_0.4.3_amd64.deb generated.
CPack: - checksum file: /home/roger/Desktop/xeve/build/xeve-base_0.4.3_amd64.deb.md5 generated.
roger@RogerUSB:~/Desktop/xeve/build$ cpack -G "RPM" ..
CPack: Create package using RPM
CPack Error: Problem removing toplevel directory: /home/roger/Desktop/xeve/build/_CPack_Packages/Linux/RPM
CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ cpack -G "ZIP" ..
CPack: Create package using ZIP
CPack: Install projects
CPack: - Run preinstall target for: XEVE
CPack: - Install project: XEVE []
CMake Error at /home/roger/Desktop/xeve/build/cmake_install.cmake:60 (file):
  file failed to open for writing (Permission denied):

    /home/roger/Desktop/xeve/build/install_manifest.txt


CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ cd..
cd..: command not found
roger@RogerUSB:~/Desktop/xeve/build$ cd ..
roger@RogerUSB:~/Desktop/xeve$ cpack -G "ZIP" ..
CPack Error: CPack project name not specified
roger@RogerUSB:~/Desktop/xeve$ cpack -G "RPM" ..
CPack Error: CPack project name not specified
roger@RogerUSB:~/Desktop/xeve$ cd built
bash: cd: built: No such file or directory
roger@RogerUSB:~/Desktop/xeve$ cd build
roger@RogerUSB:~/Desktop/xeve/build$ cpack -G "RPM" ..
CPack: Create package using RPM
CPack Error: Problem removing toplevel directory: /home/roger/Desktop/xeve/build/_CPack_Packages/Linux/RPM
CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ cpack -G "ZIP" ..
CPack: Create package using ZIP
CPack: Install projects
CPack: - Run preinstall target for: XEVE
CPack: - Install project: XEVE []
CMake Error at /home/roger/Desktop/xeve/build/cmake_install.cmake:60 (file):
  file failed to open for writing (Permission denied):

    /home/roger/Desktop/xeve/build/install_manifest.txt


CPack Error: Error when generating package: xeve-base
roger@RogerUSB:~/Desktop/xeve/build$ sudo cpack -G "ZIP" ..
CPack: Create package using ZIP
```

Figure A4: terminal procedure to install EVC encoder Part 4.

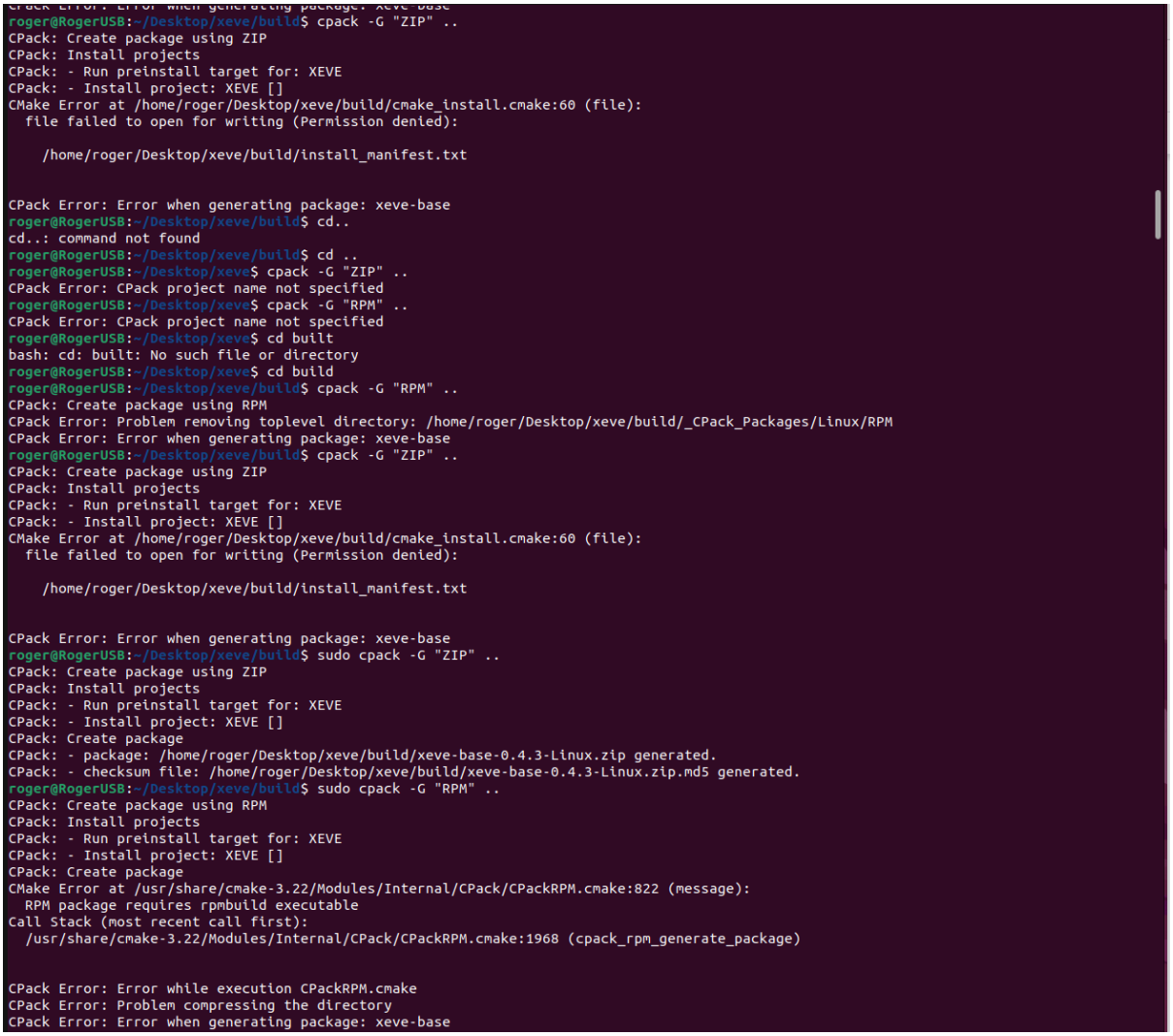

Figure A5: terminal procedure to install EVC encoder Part 5.

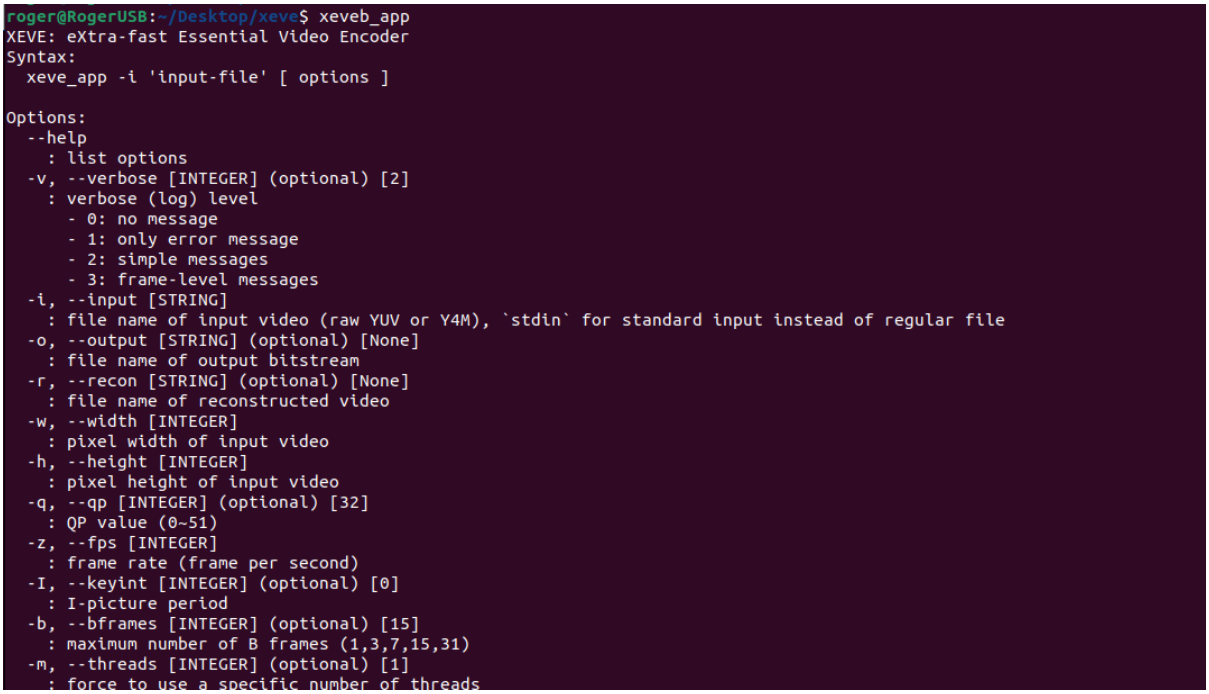

Figure A6: terminal procedure to install EVC encoder Part 6.

```
roger@RogerUSB:~/Desktop$ git clone https://github.com/mpeg5/xevd
Cloning into 'xevd'...
remote: Enumerating objects: 732, done.
remote: Counting objects: 100% (289/289), done.
remote: Compressing objects: 100% (130/130), done.
remote: Total 732 (delta 171), reused 235 (delta 151), pack-reused 443
Receiving objects: 100% (732/732), 804.87 KiB | 3.35 MiB/s, done.
Resolving deltas: 100% (487/487), done.
roger@RogerUSB:~/Desktop$ mkdir build
roger@RogerUSB:~/Desktop$ cd build
roger@RogerUSB:~/Desktop/build$ cmake ..
CMake Error: The source directory "/home/roger/Desktop" does not appear to contain CMakeLists.txt.
Specify --help for usage, or press the help button on the CMake GUI.
roger@RogerUSB:~/Desktop/build$ cd xevd
bash: cd: xevd: No such file or directory
roger@RogerUSB:~/Desktop/build$ cd ..
roger@RogerUSB:~/.local/share/Trash/files$ cd ..
roger@RogerUSB:~/.local/share/Trash$ cd ..
roger@RogerUSB:~/.local/share$ cd ..
roger@RogerUSB:~/.local$ cd ..
roger@RogerUSB:~$ cd Desktop
roger@RogerUSB:~/Desktop$ ls
RogerUSB_230216_1256.nmon  vvenc  xevd  xeve
roger@RogerUSB:~/Desktop$ cd xevd
roger@RogerUSB:~/Desktop/xevd$ ls
app              cmake_uninstall.cmake.in  inc        README.md  src_main
CMakeLists.txt   COPYING                   pkgconfig  src_base
roger@RogerUSB:~/Desktop/xevd$ mkdir build
roger@RogerUSB:~/Desktop/xevd$ cd build
roger@RogerUSB:~/Desktop/xevd/build$ cmake ..
Version string has been taken from git tag.
XEVD VERSION=0.4.1
SET_PROF=MAIN
-- The C compiler identification is GNU 11.3.0
-- The CXX compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
CMAKE_BUILD_TYPE=Release
c Flags:  -O3 -DNDEBUG -fomit-frame-pointer -Wall -Wno-unused-function -Wno-unused-but-set-variable -Wno-unused-variable -Wno-attributes
-Werror -Wno-unknown-pragmas -Wno-stringop-overflow -std=c99 -Wno-pointer-sign -pthread -Wno-pointer-to-int-cast -Wno-maybe-uninitialized
linker Flags:  -lm
-- Performing Test COMPILER_HAS_HIDDEN_VISIBILITY
-- Performing Test COMPILER_HAS_HIDDEN_VISIBILITY - Success
-- Performing Test COMPILER_HAS_HIDDEN_INLINE_VISIBILITY
-- Performing Test COMPILER_HAS_HIDDEN_INLINE_VISIBILITY - Success
-- Performing Test COMPILER_HAS_DEPRECATED_ATTR
-- Performing Test COMPILER_HAS_DEPRECATED_ATTR - Success
-- Configuring done
-- Generating done
-- Build files have been written to: /home/roger/Desktop/xevd/build
roger@RogerUSB:~/Desktop/xevd/build$ make
[  1%] Building C object src_main/CMakeFiles/xevd.dir/xevdm.c.o
[  2%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_bsr.c.o
[  4%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_df.c.o
[  5%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_eco.c.o
[  6%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_ipred.c.o
[  8%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_itdq.c.o
[  9%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_mc.c.o
[ 11%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_picman.c.o
[ 12%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_port.c.o
[ 13%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_recon.c.o
[ 15%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_tbl.c.o
[ 16%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_tp.c.o
[ 18%] Building C object src_main/CMakeFiles/xevd.dir/__/src_base/xevd_util.c.o
[ 19%] Building C object src_main/CMakeFiles/xevd.dir/xevdm_alf.c.o
[ 20%] Building C object src_main/CMakeFiles/xevd.dir/xevdm_df.c.o
[ 22%] Building C object src_main/CMakeFiles/xevd.dir/xevdm_dra.c.o
```

Figure A7: terminal procedure to install EVC decoder Part 1.

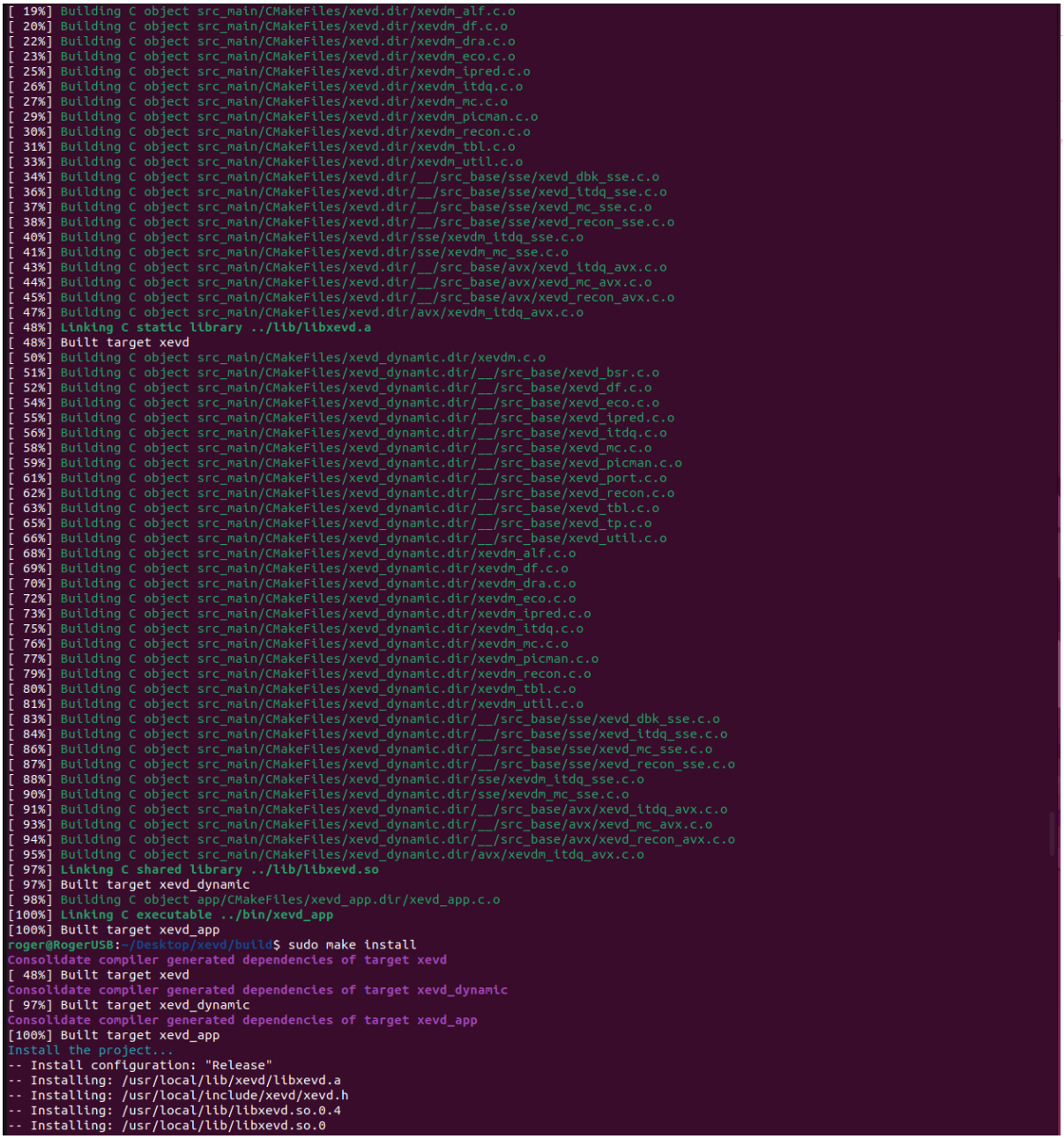

Figure A8: terminal procedure to install EVC decoder Part 2.

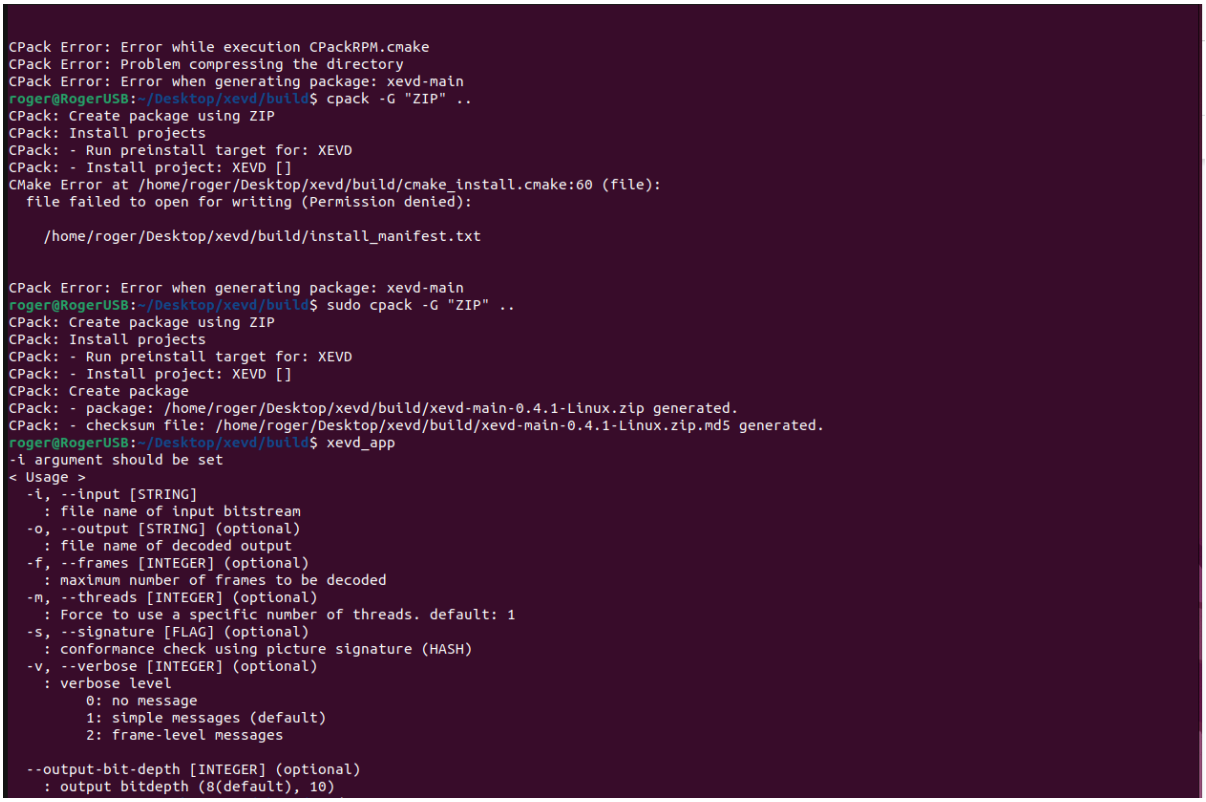

Figure A9: terminal procedure to install EVC decoder Part 3.

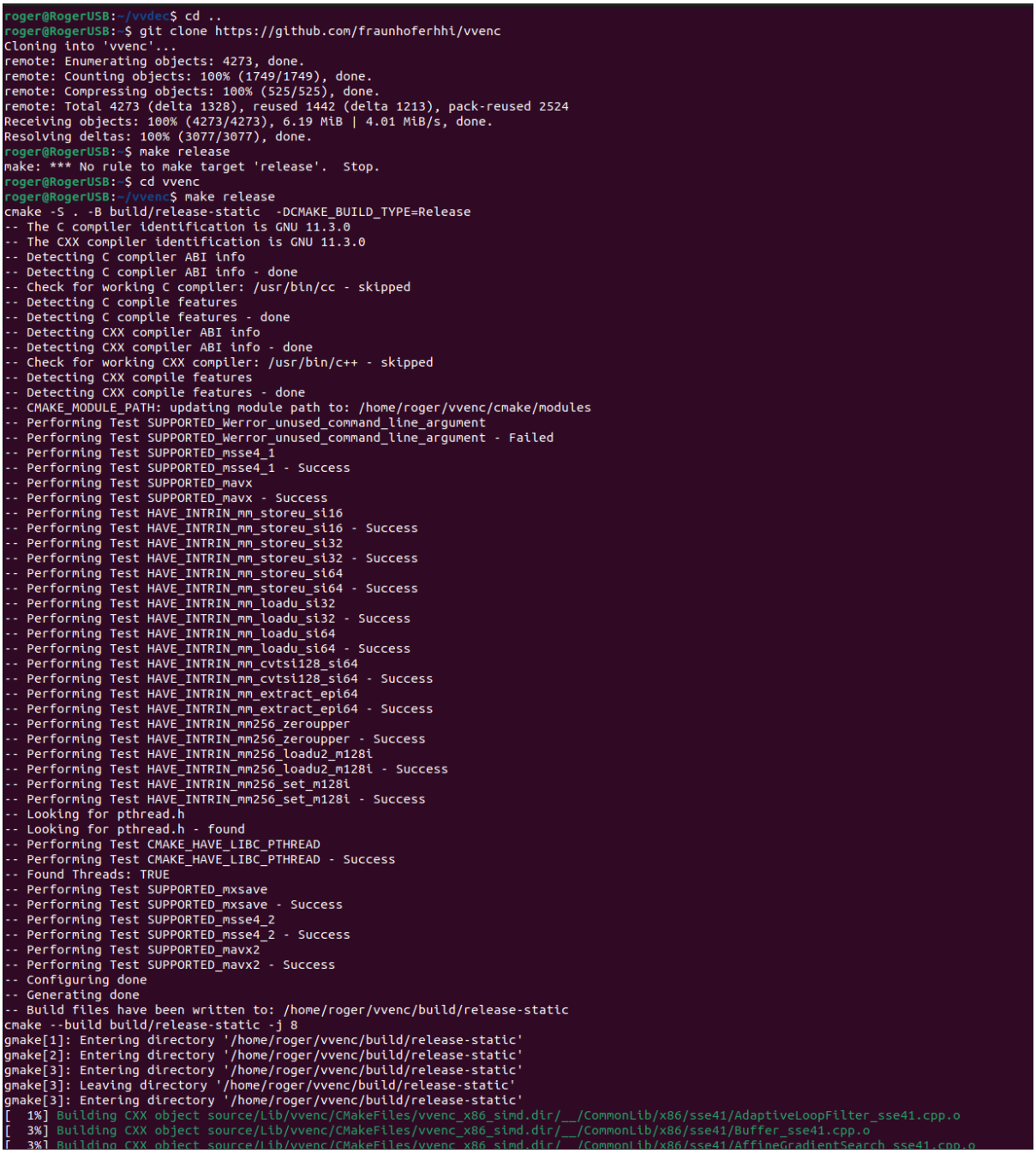

Figure A10: terminal procedure to install VVC encoder Part 1.

```
[ 26%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx/RdCost_avx.cpp.o
[ 27%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx/SampleAdaptiveOffset_avx.cpp.o
[ 28%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx/Trafo_avx.cpp.o
[ 29%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/AdaptiveLoopFilter_avx2.cpp.o
[ 29%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/AffineGradientSearch_avx2.cpp.o
[ 30%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/Buffer_avx2.cpp.o
[ 31%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/InterPred_avx2.cpp.o
[ 32%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/InterpolationFilter_avx2.cpp.o
[ 33%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/IntraPred_avx2.cpp.o
[ 34%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/LoopFilter_avx2.cpp.o
[ 34%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/MCTF_avx2.cpp.o
[ 35%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/Quant_avx2.cpp.o
[ 36%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/RdCost_avx2.cpp.o
[ 37%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/SampleAdaptiveOffset_avx2.cpp.o
[ 38%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc_x86_simd.dir/__/CommonLib/x86/avx2/Trafo_avx2.cpp.o
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 38%] Built target vvenc_x86_simd
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
[ 41%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/AffineGradientSearch.cpp.o
[ 41%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Buffer.cpp.o
[ 41%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/BitStream.cpp.o
[ 42%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/CodingStructure.cpp.o
[ 42%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/AdaptiveLoopFilter.cpp.o
[ 44%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Contexts.cpp.o
[ 44%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/DepQuant.cpp.o
[ 44%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/ContextModelling.cpp.o
[ 45%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/InterPrediction.cpp.o
[ 46%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/InterpolationFilter.cpp.o
[ 46%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/IntraPrediction.cpp.o
[ 47%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/LoopFilter.cpp.o
[ 48%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/MCTF.cpp.o
[ 49%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/MatrixIntraPrediction.cpp.o
[ 50%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Mv.cpp.o
[ 51%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/PicYuvMD5.cpp.o
[ 51%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Picture.cpp.o
[ 52%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/ProfileLevelTier.cpp.o
[ 53%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Quant.cpp.o
[ 54%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/QuantRDOQ.cpp.o
[ 55%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/QuantRDOQ2.cpp.o
[ 55%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/RdCost.cpp.o
[ 56%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Reshape.cpp.o
[ 57%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Rom.cpp.o
[ 58%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/RomTr.cpp.o
[ 59%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/SEI.cpp.o
[ 59%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/SampleAdaptiveOffset.cpp.o
[ 60%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/SearchSpaceCounter.cpp.o
[ 61%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Slice.cpp.o
[ 62%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/StatCounter.cpp.o
[ 63%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/TimeProfiler.cpp.o
[ 64%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/TrQuant.cpp.o
[ 64%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/TrQuant_EMT.cpp.o
[ 65%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Unit.cpp.o
[ 66%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/UnitPartitioner.cpp.o
[ 67%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/UnitTools.cpp.o
[ 68%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/dtrace.cpp.o
[ 68%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/DecoderLib/DecCu.cpp.o
[ 69%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/BinEncoder.cpp.o
[ 70%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/BitAllocation.cpp.o
[ 71%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/CABACWriter.cpp.o
[ 72%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncAdaptiveLoopFilter.cpp.o
[ 72%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncCfg.cpp.o
[ 73%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncCu.cpp.o
[ 74%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncGOP.cpp.o
[ 75%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncHRD.cpp.o
[ 76%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncLib.cpp.o
[ 77%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncModeCtrl.cpp.o
[ 77%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncPicture.cpp.o
[ 78%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncReshape.cpp.o
[ 79%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncSampleAdaptiveOffset.cpp.o
[ 80%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncSlice.cpp.o
[ 81%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/GOPCfg.cpp.o
[ 81%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/InterSearch.cpp.o
[ 82%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/IntraSearch.cpp.o
[ 83%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/NALwrite.cpp.o
```

Figure A11: terminal procedure to install VVC encoder Part 2.

```
[ 87%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/VLCWriter.cpp.o
[ 88%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/Utilities/NoMallocThreadPool.cpp.o
[ 89%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvenc.cpp.o
[ 90%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvencCfg.cpp.o
[ 90%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvencimpl.cpp.o
[ 91%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/x86/CommonDefX86.cpp.o
[ 92%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/x86/InitX86.cpp.o
[ 93%] Linking CXX static library ../../../../../lib/release-static/libvvenc.a
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 93%] Built target vvenc
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
[ 94%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/EncApp.cpp.o
[ 94%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/encmain.cpp.o
[ 94%] Building CXX object source/App/vvencapp/CMakeFiles/vvencapp.dir/vvencapp.cpp.o
[ 95%] Building CXX object test/vvenclibtest/CMakeFiles/vvenclibtest.dir/vvenclibtest.cpp.o
[ 96%] Building C object test/vvencinterfacetest/CMakeFiles/vvencinterfacetest.dir/vvencinterfacetest.c.o
[ 97%] Linking CXX executable ../../../../bin/release-static/vvencinterfacetest
[ 98%] Linking CXX executable ../../../../bin/release-static/vvenclibtest
[ 99%] Linking CXX executable ../../../../../bin/release-static/vvencFFapp
[100%] Linking CXX executable ../../../../../bin/release-static/vvencapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[100%] Built target vvencinterfacetest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[100%] Built target vvenclibtest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[100%] Built target vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[100%] Built target vvencapp
gmake[2]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[1]: Leaving directory '/home/roger/vvenc/build/release-static'
roger@RogerUSB:~/vvenc$ make debug
cmake -S . -B build/debug-static  -DCMAKE_BUILD_TYPE=Debug
-- The C compiler identification is GNU 11.3.0
-- The CXX compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- CMAKE_MODULE_PATH: updating module path to: /home/roger/vvenc/cmake/modules
-- Performing Test SUPPORTED_Werror_unused_command_line_argument
-- Performing Test SUPPORTED_Werror_unused_command_line_argument - Failed
-- Performing Test SUPPORTED_msse4_1
-- Performing Test SUPPORTED_msse4_1 - Success
-- Performing Test SUPPORTED_mavx
-- Performing Test SUPPORTED_mavx - Success
-- Performing Test HAVE_INTRIN_mm_storeu_si16
-- Performing Test HAVE_INTRIN_mm_storeu_si16 - Success
-- Performing Test HAVE_INTRIN_mm_storeu_si32
-- Performing Test HAVE_INTRIN_mm_storeu_si32 - Success
-- Performing Test HAVE_INTRIN_mm_storeu_si64
-- Performing Test HAVE_INTRIN_mm_storeu_si64 - Success
-- Performing Test HAVE_INTRIN_mm_loadu_si32
-- Performing Test HAVE_INTRIN_mm_loadu_si32 - Success
-- Performing Test HAVE_INTRIN_mm_loadu_si64
-- Performing Test HAVE_INTRIN_mm_loadu_si64 - Success
-- Performing Test HAVE_INTRIN_mm_cvtsi128_si64
-- Performing Test HAVE_INTRIN_mm_cvtsi128_si64 - Success
-- Performing Test HAVE_INTRIN_mm_extract_epi64
-- Performing Test HAVE_INTRIN_mm_extract_epi64 - Success
```

Figure A12: terminal procedure to install VVC encoder Part 3.

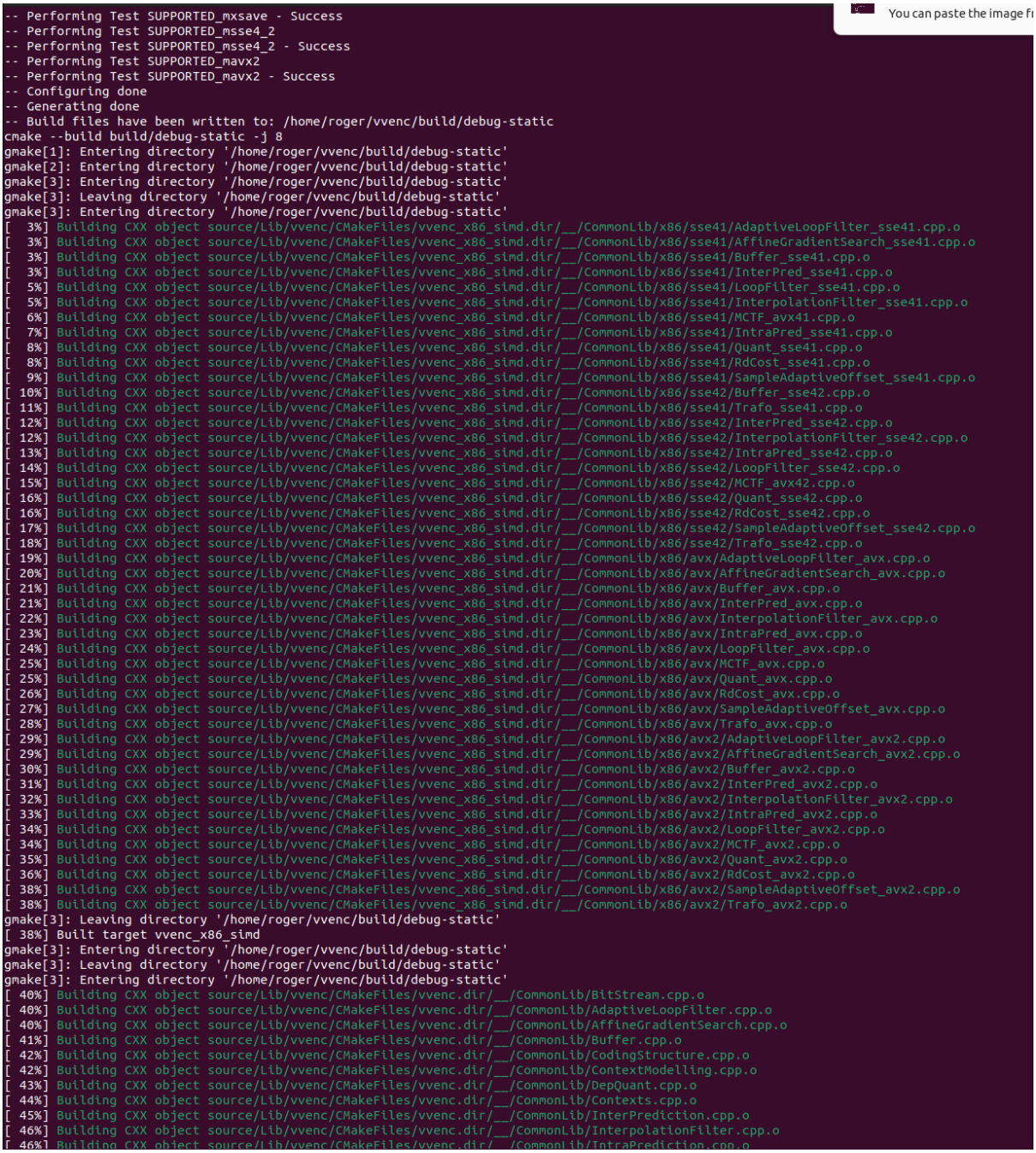

Figure A13: terminal procedure to install VVC encoder Part 4.

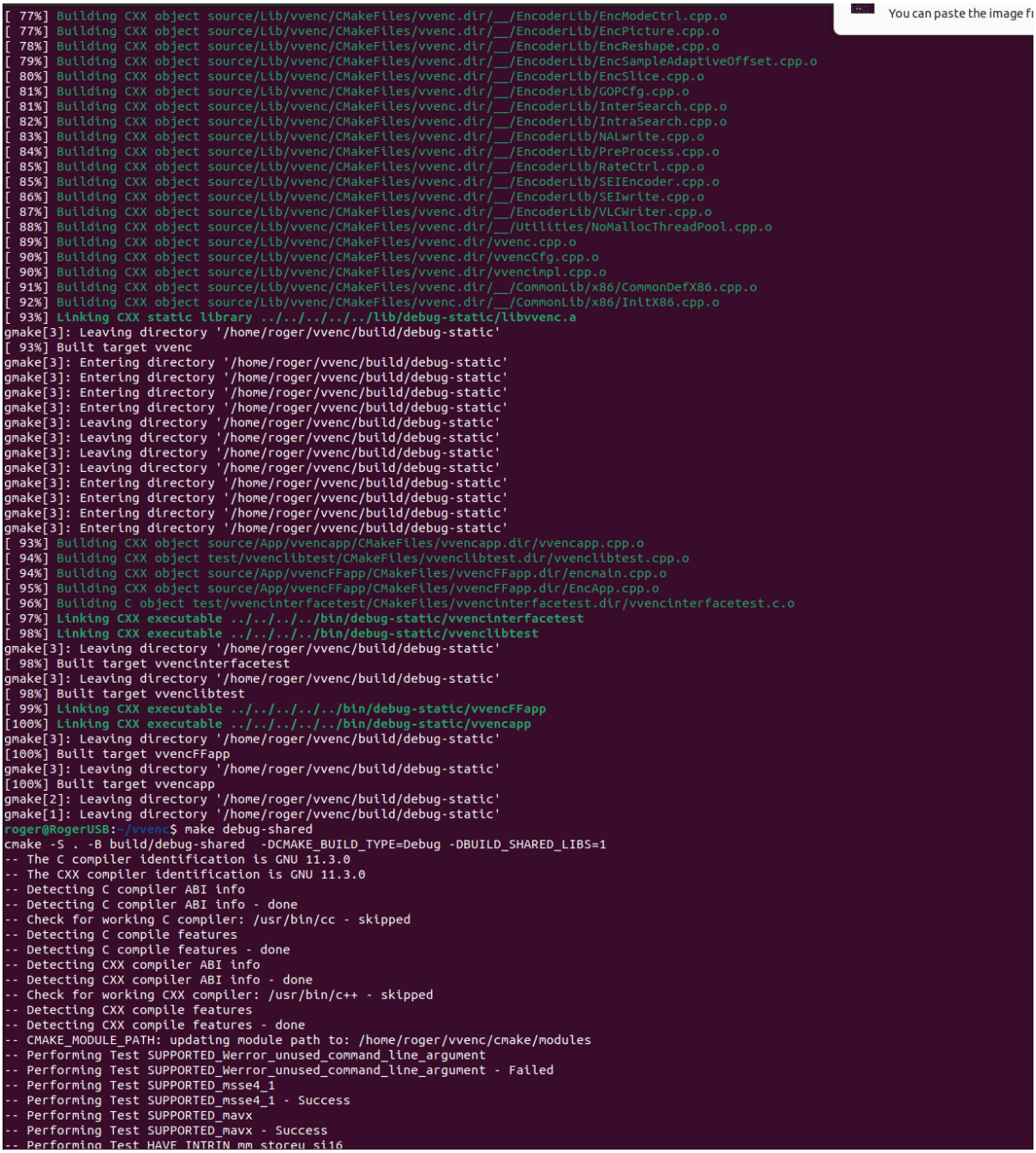

Figure A14: terminal procedure to install VVC encoder Part 5.

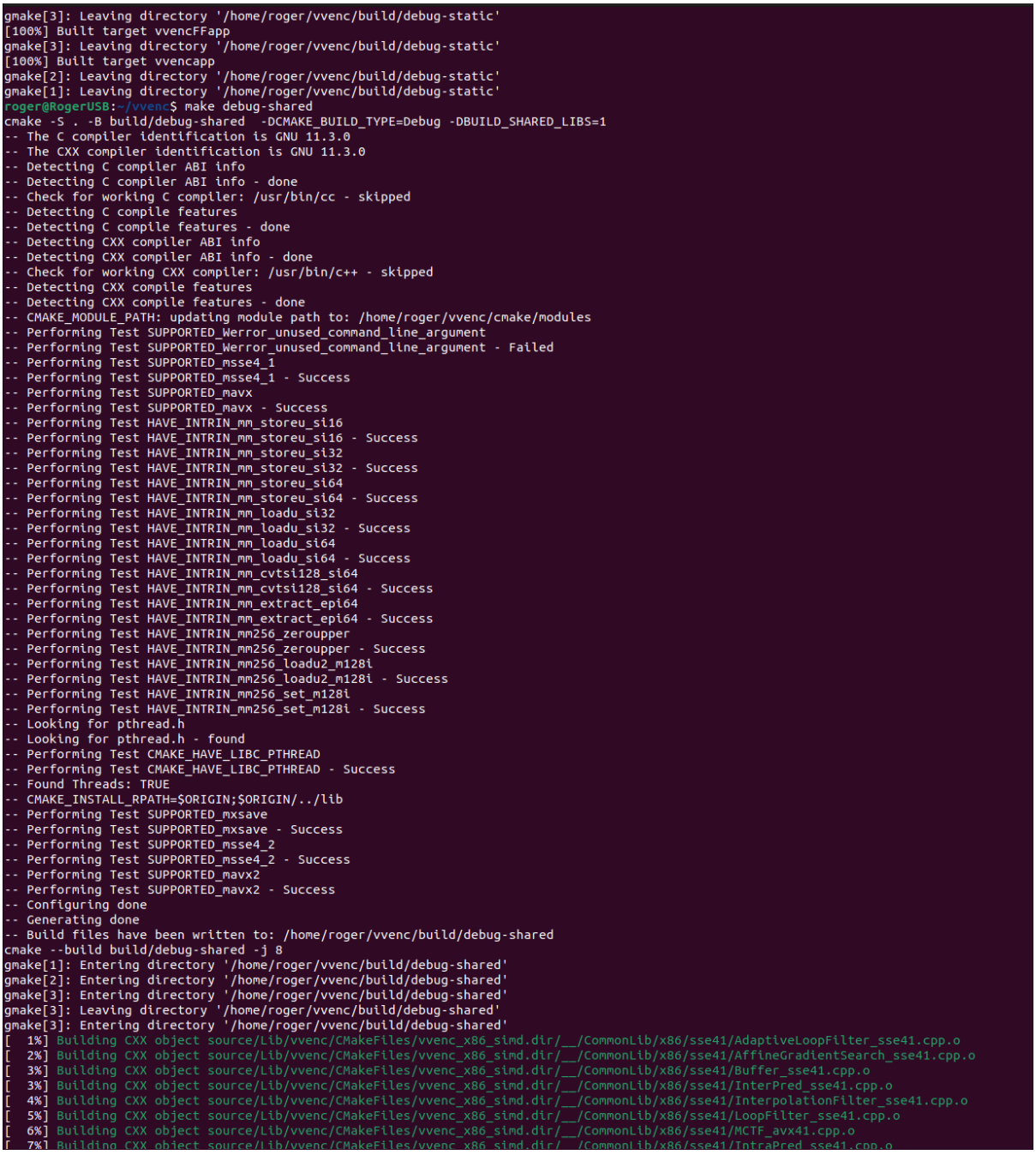

Figure A15: terminal procedure to install VVC encoder Part 6.

```
[ 46%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/InterpolationFilter.cpp.o
[ 46%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/IntraPrediction.cpp.o
[ 47%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/LoopFilter.cpp.o
[ 48%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/MCTF.cpp.o
[ 49%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/MatrixIntraPrediction.cpp.o
[ 50%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Mv.cpp.o
[ 51%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/PicYuvMD5.cpp.o
[ 51%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Picture.cpp.o
[ 52%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/ProfileLevelTier.cpp.o
[ 53%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Quant.cpp.o
[ 54%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/QuantRDOQ.cpp.o
[ 55%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/QuantRDOQ2.cpp.o
[ 55%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/RdCost.cpp.o
[ 56%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Reshape.cpp.o
[ 57%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Rom.cpp.o
[ 58%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/RomTr.cpp.o
[ 59%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/SEI.cpp.o
[ 59%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/SampleAdaptiveOffset.cpp.o
[ 60%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/SearchSpaceCounter.cpp.o
[ 61%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Slice.cpp.o
[ 62%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/StatCounter.cpp.o
[ 63%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/TimeProfiler.cpp.o
[ 64%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/TrQuant.cpp.o
[ 64%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/TrQuant_EMT.cpp.o
[ 65%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/Unit.cpp.o
[ 66%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/UnitPartitioner.cpp.o
[ 67%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/UnitTools.cpp.o
[ 68%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/dtrace.cpp.o
[ 68%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/DecoderLib/DecCu.cpp.o
[ 69%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/BinEncoder.cpp.o
[ 70%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/BitAllocation.cpp.o
[ 71%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/CABACWriter.cpp.o
[ 72%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncAdaptiveLoopFilter.cpp.o
[ 72%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncCfg.cpp.o
[ 73%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncCu.cpp.o
[ 74%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncGOP.cpp.o
[ 75%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncHRD.cpp.o
[ 76%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncLib.cpp.o
[ 77%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncModeCtrl.cpp.o
[ 77%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncPicture.cpp.o
[ 78%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncReshape.cpp.o
[ 79%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncSampleAdaptiveOffset.cpp.o
[ 80%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncSlice.cpp.o
[ 81%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/GOPCfg.cpp.o
[ 81%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/InterSearch.cpp.o
[ 82%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/IntraSearch.cpp.o
[ 83%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/NALwrite.cpp.o
[ 84%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/PreProcess.cpp.o
[ 85%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/RateCtrl.cpp.o
[ 85%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/SEIEncoder.cpp.o
[ 86%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/SEIwrite.cpp.o
[ 87%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/VLCWriter.cpp.o
[ 88%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/Utilities/NoMallocThreadPool.cpp.o
[ 89%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvenc.cpp.o
[ 90%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvencCfg.cpp.o
[ 90%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvencimpl.cpp.o
[ 91%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/x86/CommonDefX86.cpp.o
[ 92%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/x86/InitX86.cpp.o
[ 93%] Linking CXX shared library ../../../../../lib/debug-shared/libvvenc.so
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
[ 93%] Built target vvenc
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/debug-shared'
[ 95%] Building CXX object test/vvenclibtest/CMakeFiles/vvenclibtest.dir/vvenclibtest.cpp.o
[ 95%] Building CXX object source/App/vvencapp/CMakeFiles/vvencapp.dir/vvencapp.cpp.o
[ 95%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/EncApp.cpp.o
```

Figure A16: terminal procedure to install VVC encoder Part 7.

```
[ 97%] Built target vvencinterfacetest
[ 98%] Linking CXX executable ../../../../bin/debug-shared/vvenclibtest
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
[ 98%] Built target vvenclibtest
[ 99%] Linking CXX executable ../../../../../bin/debug-shared/vvencapp
[100%] Linking CXX executable ../../../../../bin/debug-shared/vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
[100%] Built target vvencapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/debug-shared'
[100%] Built target vvencFFapp
gmake[2]: Leaving directory '/home/roger/vvenc/build/debug-shared'
gmake[1]: Leaving directory '/home/roger/vvenc/build/debug-shared'
roger@RogerUSB:~/vvenc$ make install-release
cmake --build build/release-static -j 8
gmake[1]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[2]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
Consolidate compiler generated dependencies of target vvenc_x86_simd
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 38%] Built target vvenc_x86_simd
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
Consolidate compiler generated dependencies of target vvenc
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 93%] Built target vvenc
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
Consolidate compiler generated dependencies of target vvencapp
Consolidate compiler generated dependencies of target vvencinterfacetest
Consolidate compiler generated dependencies of target vvencFFapp
Consolidate compiler generated dependencies of target vvenclibtest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 95%] Built target vvencinterfacetest
[ 97%] Built target vvenclibtest
[ 98%] Built target vvencapp
[100%] Built target vvencFFapp
gmake[2]: Leaving directory '/home/roger/vvenc/build/release-static'
gmake[1]: Leaving directory '/home/roger/vvenc/build/release-static'
cmake --build build/release-static --target install
gmake[1]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[2]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 38%] Built target vvenc_x86_simd
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 93%] Built target vvenc
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 94%] Built target vvencapp
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 96%] Built target vvencFFapp
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[ 98%] Built target vvenclibtest
gmake[3]: Entering directory '/home/roger/vvenc/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-static'
[100%] Built target vvencinterfacetest
gmake[2]: Leaving directory '/home/roger/vvenc/build/release-static'
Install the project...
-- Install configuration: "Release"
-- Installing: /home/roger/vvenc/install/include/vvenc/version.h
-- Up-to-date: /home/roger/vvenc/install/include/vvenc
-- Installing: /home/roger/vvenc/install/include/vvenc/vvencCfg.h
-- Installing: /home/roger/vvenc/install/include/vvenc/vvencDecl.h
-- Installing: /home/roger/vvenc/install/include/vvenc/vvenc.h
-- Installing: /home/roger/vvenc/install/lib/libvvenc.a
-- Installing: /home/roger/vvenc/install/bin/vvencapp
-- Installing: /home/roger/vvenc/install/lib/cmake/vvenc/vvencConfig.cmake
-- Installing: /home/roger/vvenc/install/lib/cmake/vvenc/vvencConfigVersion.cmake
-- Installing: /home/roger/vvenc/install/lib/cmake/vvenc/vvencTargets-static.cmake
```

Figure A17: terminal procedure to install VVC encoder Part 8.

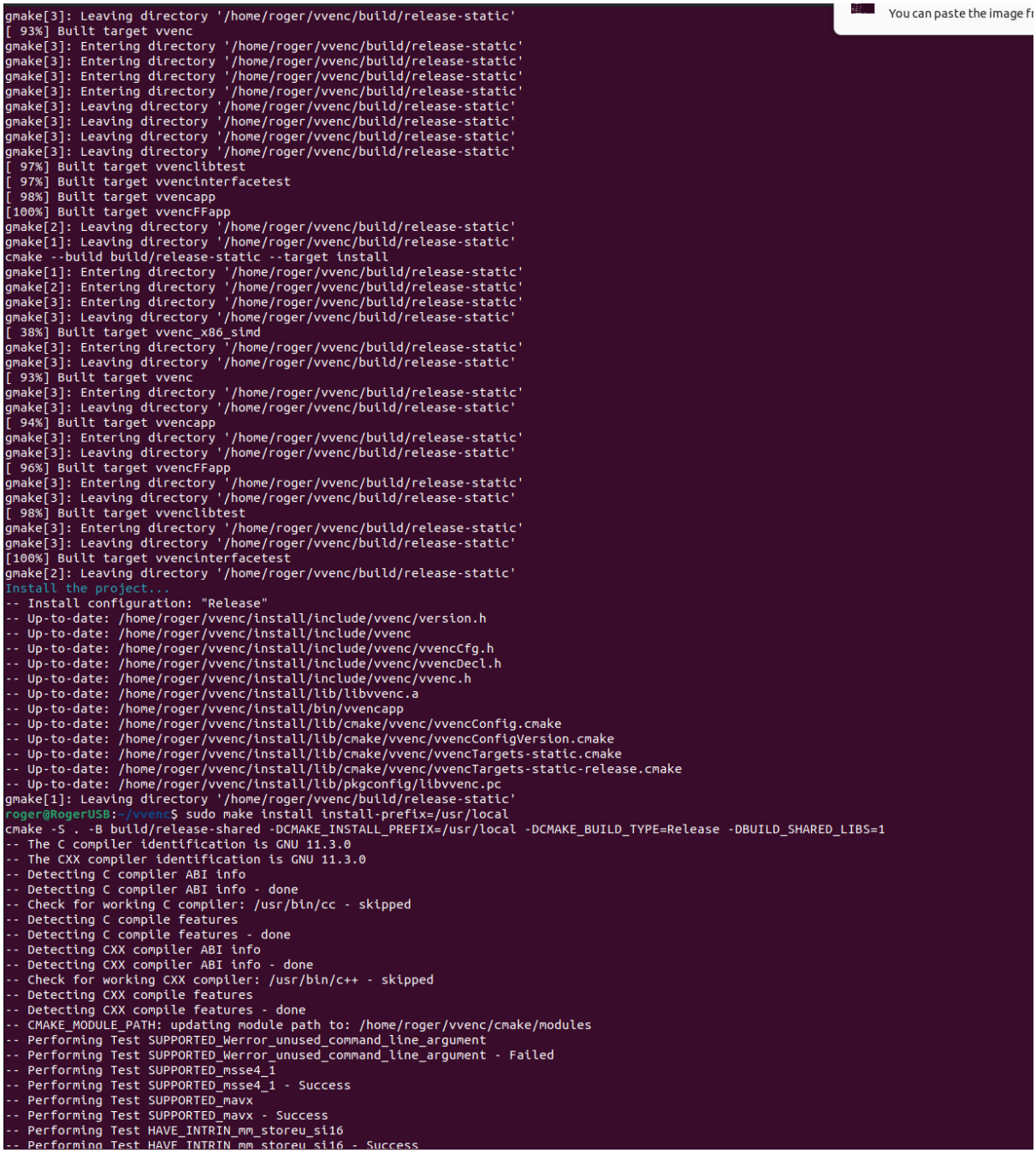

Figure A18: terminal procedure to install VVC encoder Part 9.

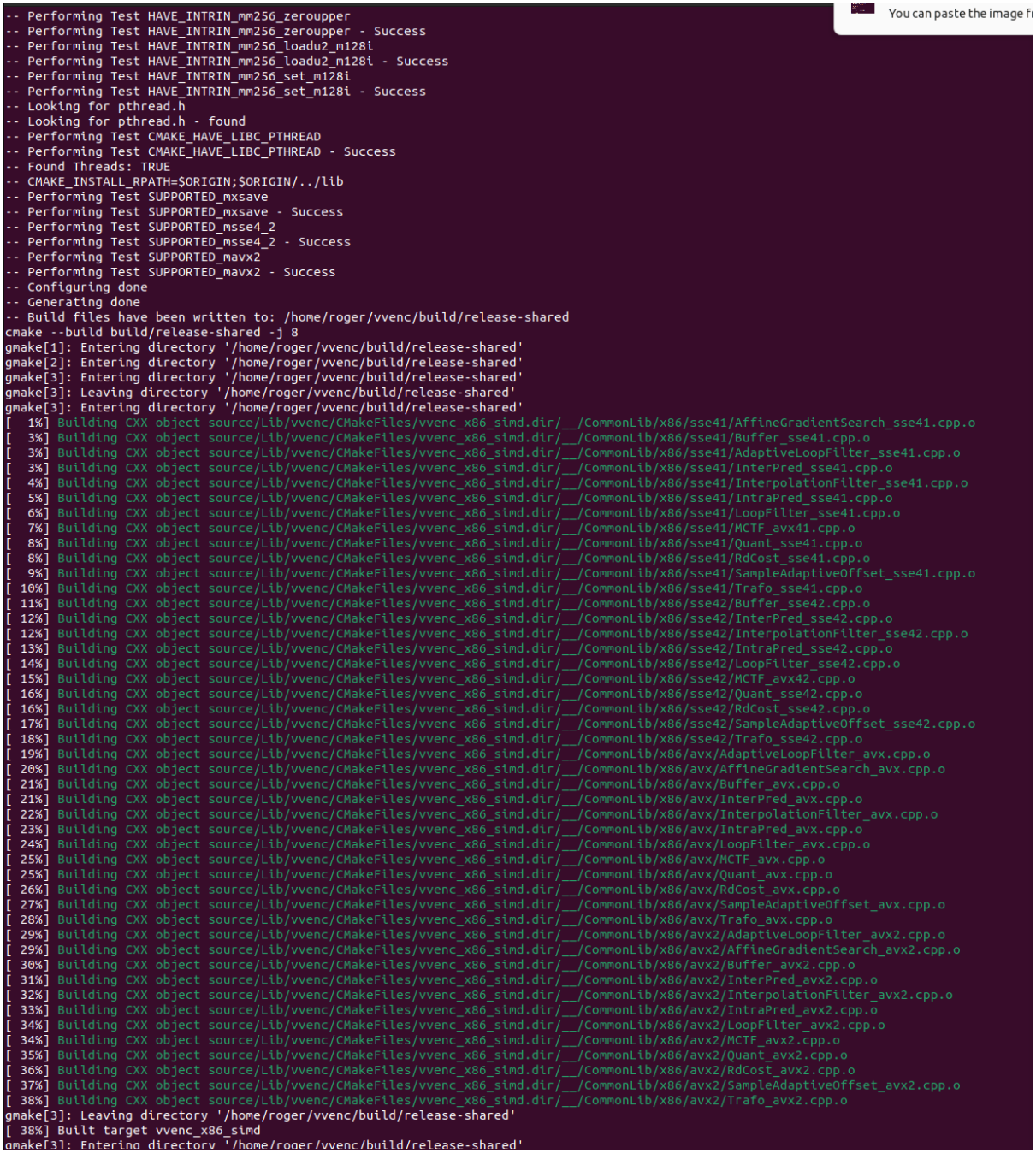

Figure A19: terminal procedure to install VVC encoder Part 10.

```
[ 66%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/UnitPartitioner.cpp.o
[ 67%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/UnitTools.cpp.o
[ 68%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/dtrace.cpp.o
[ 68%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/DecoderLib/DecCu.cpp.o
[ 69%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/BinEncoder.cpp.o
[ 70%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/BitAllocation.cpp.o
[ 71%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/CABACWriter.cpp.o
[ 72%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncAdaptiveLoopFilter.cpp.o
[ 72%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncCfg.cpp.o
[ 73%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncCu.cpp.o
[ 74%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncGOP.cpp.o
[ 75%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncHRD.cpp.o
[ 76%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncLib.cpp.o
[ 77%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncModeCtrl.cpp.o
[ 77%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncPicture.cpp.o
[ 78%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncReshape.cpp.o
[ 79%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncSampleAdaptiveOffset.cpp.o
[ 80%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/EncSlice.cpp.o
[ 81%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/GOPCfg.cpp.o
[ 81%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/InterSearch.cpp.o
[ 82%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/IntraSearch.cpp.o
[ 83%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/NALwrite.cpp.o
[ 84%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/PreProcess.cpp.o
[ 85%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/RateCtrl.cpp.o
[ 85%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/SEIEncoder.cpp.o
[ 86%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/SEIwrite.cpp.o
[ 87%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/EncoderLib/VLCWriter.cpp.o
[ 88%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/Utilities/NoMallocThreadPool.cpp.o
[ 89%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvenc.cpp.o
[ 90%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvencCfg.cpp.o
[ 90%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/vvencimpl.cpp.o
[ 91%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/x86/CommonDefX86.cpp.o
[ 92%] Building CXX object source/Lib/vvenc/CMakeFiles/vvenc.dir/__/CommonLib/x86/InitX86.cpp.o
[ 93%] Linking CXX shared library ../../../../../lib/release-shared/libvvenc.so
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 93%] Built target vvenc
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
[ 93%] Building CXX object source/App/vvencapp/CMakeFiles/vvencapp.dir/vvencapp.cpp.o
[ 94%] Building CXX object test/vvenclibtest/CMakeFiles/vvenclibtest.dir/vvenclibtest.cpp.o
[ 94%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/encmain.cpp.o
[ 95%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/EncApp.cpp.o
[ 96%] Building C object test/vvencinterfacetest/CMakeFiles/vvencinterfacetest.dir/vvencinterfacetest.c.o
[ 97%] Linking C executable ../../../../bin/release-shared/vvencinterfacetest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 97%] Built target vvencinterfacetest
[ 98%] Linking CXX executable ../../../../bin/release-shared/vvenclibtest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 98%] Built target vvenclibtest
[ 99%] Linking CXX executable ../../../../../bin/release-shared/vvencapp
[100%] Linking CXX executable ../../../../../bin/release-shared/vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[100%] Built target vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[100%] Built target vvencapp
gmake[2]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[1]: Leaving directory '/home/roger/vvenc/build/release-shared'
cmake --build build/release-shared --target install
gmake[1]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[2]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvenc_x86_simd
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 38%] Built target vvenc_x86_simd
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvenc
```

Figure A20: terminal procedure to install VVC encoder Part 11.

```
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
[ 93%] Building CXX object source/App/vvencapp/CMakeFiles/vvencapp.dir/vvencapp.cpp.o
[ 94%] Building CXX object test/vvenclibtest/CMakeFiles/vvenclibtest.dir/vvenclibtest.cpp.o
[ 94%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/encmain.cpp.o
[ 95%] Building CXX object source/App/vvencFFapp/CMakeFiles/vvencFFapp.dir/EncApp.cpp.o
[ 96%] Building C object test/vvencinterfacetest/CMakeFiles/vvencinterfacetest.dir/vvencinterfacetest.c.o
[ 97%] Linking C executable ../../../../bin/release-shared/vvencinterfacetest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 97%] Built target vvencinterfacetest
[ 98%] Linking CXX executable ../../../../bin/release-shared/vvenclibtest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 98%] Built target vvenclibtest
[ 99%] Linking CXX executable ../../../../../bin/release-shared/vvencapp
[100%] Linking CXX executable ../../../../../bin/release-shared/vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[100%] Built target vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[100%] Built target vvencapp
gmake[2]: Leaving directory '/home/roger/vvenc/build/release-shared'
gmake[1]: Leaving directory '/home/roger/vvenc/build/release-shared'
cmake --build build/release-shared --target install
gmake[1]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[2]: Entering directory '/home/roger/vvenc/build/release-shared'
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvenc_x86_simd
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 38%] Built target vvenc_x86_simd
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvenc
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 93%] Built target vvenc
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvencapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 94%] Built target vvencapp
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvencFFapp
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 96%] Built target vvencFFapp
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvenclibtest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[ 98%] Built target vvenclibtest
gmake[3]: Entering directory '/home/roger/vvenc/build/release-shared'
Consolidate compiler generated dependencies of target vvencinterfacetest
gmake[3]: Leaving directory '/home/roger/vvenc/build/release-shared'
[100%] Built target vvencinterfacetest
gmake[2]: Leaving directory '/home/roger/vvenc/build/release-shared'
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/include/vvenc/version.h
-- Up-to-date: /usr/local/include/vvenc
-- Installing: /usr/local/include/vvenc/vvencCfg.h
-- Installing: /usr/local/include/vvenc/vvencDecl.h
-- Installing: /usr/local/include/vvenc/vvenc.h
-- Installing: /usr/local/lib/libvvenc.so.1.8.0
-- Installing: /usr/local/lib/libvvenc.so.1.8
-- Set runtime path of "/usr/local/lib/libvvenc.so.1.8.0" to "$ORIGIN:$ORIGIN/../lib"
-- Installing: /usr/local/lib/libvvenc.so
-- Installing: /usr/local/bin/vvencapp
-- Set runtime path of "/usr/local/bin/vvencapp" to "$ORIGIN:$ORIGIN/../lib"
-- Installing: /usr/local/lib/cmake/vvenc/vvencConfig.cmake
-- Installing: /usr/local/lib/cmake/vvenc/vvencConfigVersion.cmake
-- Installing: /usr/local/lib/cmake/vvenc/vvencTargets-shared.cmake
-- Installing: /usr/local/lib/cmake/vvenc/vvencTargets-shared-release.cmake
-- Installing: /usr/local/lib/pkgconfig/libvvenc.pc
gmake[1]: Leaving directory '/home/roger/vvenc/build/release-shared'
```

Figure A21: terminal procedure to install VVC encoder Part 12.

```
roger@RogerUSB:~$ vvencapp
vvencapp: Fraunhofer VVC Encoder ver. 1.7.0 [Linux][GCC 11.3.0][64 bit][SIMD=AVX2]

#======== General Options ================
  -h,   --help [0]                  show default help
        --fullhelp [0]              show full help
  -v,   --verbosity [verbose]       Specifies the level of the verboseness (0: silent, 1: error, 2: warning, 3: info, 4:
                                    notice, 5: verbose, 6: debug)
        --version [0]               show version

#======== Input Options ================
  -i,   --input []                  original YUV input file name or '-' for reading from stdin
  -s,   --size [1920x1080]          specify input resolution (WidthxHeight)
  -c,   --format [yuv420]           set input format (yuv420, yuv420_10, yuv420_10_packed)
  -r,   --framerate [60]            temporal rate (framerate numerator) e.g. 25,30, 30000, 50,60, 60000
        --framescale [1]            temporal scale (framerate denominator) e.g. 1, 1001
        --fps [60/1]                Framerate as int or fraction (num/denom)
        --tickspersec [27000000]    Ticks Per Second for dts generation, (1..27000000, -1: ticks per frame=1)
  -f,   --frames [0]                max. frames to encode [all]
  -fs,  --frameskip [0]             Number of frames to skip at start of input YUV [off]
        --segment [off]             when encoding multiple separate segments, specify segment position to enable segment concatenation (first, mid, last)
[off]
                                    first: first segment
                                    mid  : all segments between first and last segment
                                    last : last segment

#======== Output Options ================
  -o,   --output []                 Bitstream output file name

#======== Encoder Options ================
        --preset [medium]           select preset for specific encoding setting (faster, fast, medium, slow, slower)
  -b,   --bitrate [0]               bitrate for rate control (0: constant-QP encoding without rate control; otherwise
                                    bits/second (use e.g. 1.5M, 1.5Mbps, 1500k, 1500kbps, 1500000bps, 1500000))
```

Figure A22: terminal procedure to install VVC encoder Part 13.

```
roger@RogerUSB:~$ git clone https://github.com/fraunhoferhhi/vvdec
Cloning into 'vvdec'...
remote: Enumerating objects: 3115, done.
remote: Counting objects: 100% (1196/1196), done.
remote: Compressing objects: 100% (540/540), done.
remote: Total 3115 (delta 754), reused 814 (delta 633), pack-reused 1919
Receiving objects: 100% (3115/3115), 3.92 MiB | 4.65 MiB/s, done.
Resolving deltas: 100% (2006/2006), done.
roger@RogerUSB:~$ cd vvdec
roger@RogerUSB:~/vvdec$ make release

cmake -S . -B build/release-static  -DCMAKE_BUILD_TYPE=Release
-- The C compiler identification is GNU 11.3.0
-- The CXX compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- CMAKE_MODULE_PATH: updating module path to: /home/roger/vvdec/cmake/modules
-- Performing Test SUPPORTED_Werror_unused_command_line_argument
-- Performing Test SUPPORTED_Werror_unused_command_line_argument - Failed
-- Performing Test SUPPORTED_msse4_1
-- Performing Test SUPPORTED_msse4_1 - Success
-- Performing Test SUPPORTED_msse2
-- Performing Test SUPPORTED_msse2 - Success
-- Performing Test SUPPORTED_mavx
-- Performing Test SUPPORTED_mavx - Success
-- Performing Test HAVE_INTRIN_mm_storeu_si16
-- Performing Test HAVE_INTRIN_mm_storeu_si16 - Success
-- Performing Test HAVE_INTRIN_mm_storeu_si32
-- Performing Test HAVE_INTRIN_mm_storeu_si32 - Success
-- Performing Test HAVE_INTRIN_mm_storeu_si64
-- Performing Test HAVE_INTRIN_mm_storeu_si64 - Success
-- Performing Test HAVE_INTRIN_mm_loadu_si32
-- Performing Test HAVE_INTRIN_mm_loadu_si32 - Success
-- Performing Test HAVE_INTRIN_mm_loadu_si64
-- Performing Test HAVE_INTRIN_mm_loadu_si64 - Success
-- Performing Test HAVE_INTRIN_mm256_zeroupper
-- Performing Test HAVE_INTRIN_mm256_zeroupper - Success
-- Performing Test HAVE_INTRIN_mm256_loadu2_m128i
-- Performing Test HAVE_INTRIN_mm256_loadu2_m128i - Success
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Performing Test SUPPORTED_mxsave
-- Performing Test SUPPORTED_mxsave - Success
-- Performing Test SUPPORTED_msse4_2
-- Performing Test SUPPORTED_msse4_2 - Success
-- Performing Test SUPPORTED_mavx2
-- Performing Test SUPPORTED_mavx2 - Success
-- Some bitstream files are missing.
--   If you want to run tests, reconfigure with -DVVDEC_ENABLE_BITSTREAM_DOWNLOAD=ON
--   (or using top level Makefile: make test enable-bitstream-download=1)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/roger/vvdec/build/release-static
cmake --build build/release-static -j 8
gmake[1]: Entering directory '/home/roger/vvdec/build/release-static'
gmake[2]: Entering directory '/home/roger/vvdec/build/release-static'
gmake[3]: Entering directory '/home/roger/vvdec/build/release-static'
gmake[3]: Leaving directory '/home/roger/vvdec/build/release-static'
gmake[3]: Entering directory '/home/roger/vvdec/build/release-static'
[  2%] Building CXX object source/Lib/vvdec/CMakeFiles/vvdec_x86_simd.dir/__/CommonLib/x86/sse41/Buffer_sse41.cpp.o
[  2%] Building CXX object source/Lib/vvdec/CMakeFiles/vvdec_x86_simd.dir/__/CommonLib/x86/sse41/AdaptiveLoopFilter_sse41.cpp.o
[  3%] Building CXX object source/Lib/vvdec/CMakeFiles/vvdec_x86_simd.dir/__/CommonLib/x86/sse41/InterpolationFilter_sse41.cpp.o
[  4%] Building CXX object source/Lib/vvdec/CMakeFiles/vvdec_x86_simd.dir/__/CommonLib/x86/sse41/IntraPred_sse41.cpp.o
[  5%] Building CXX object source/Lib/vvdec/CMakeFiles/vvdec_x86_simd.dir/__/CommonLib/x86/sse41/Quant_sse41.cpp.o
[  6%] Building CXX object source/Lib/vvdec/CMakeFiles/vvdec_x86_simd.dir/__/CommonLib/x86/sse41/Picture_sse41.cpp.o
```

Figure A23: terminal procedure to install VVC decoder Part 1.

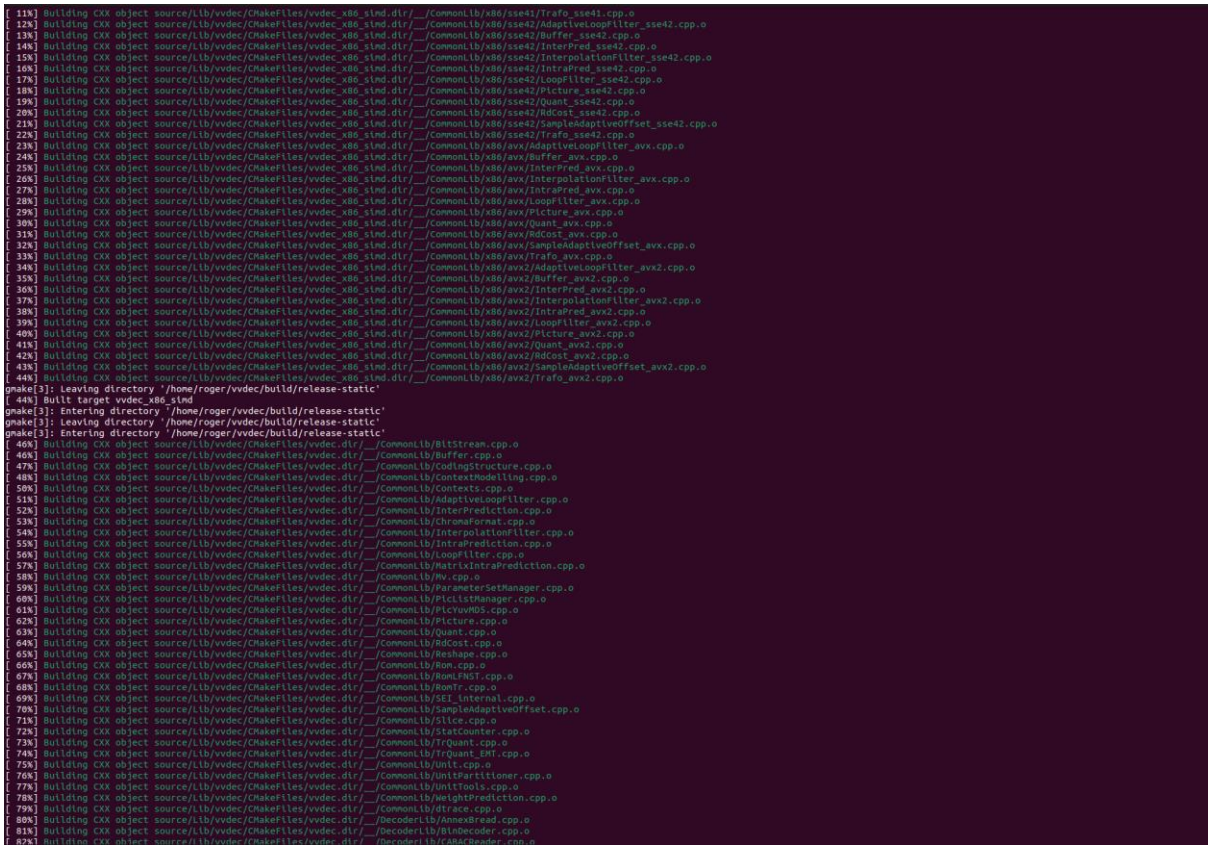

Figure A24: terminal procedure to install VVC decoder Part 2.

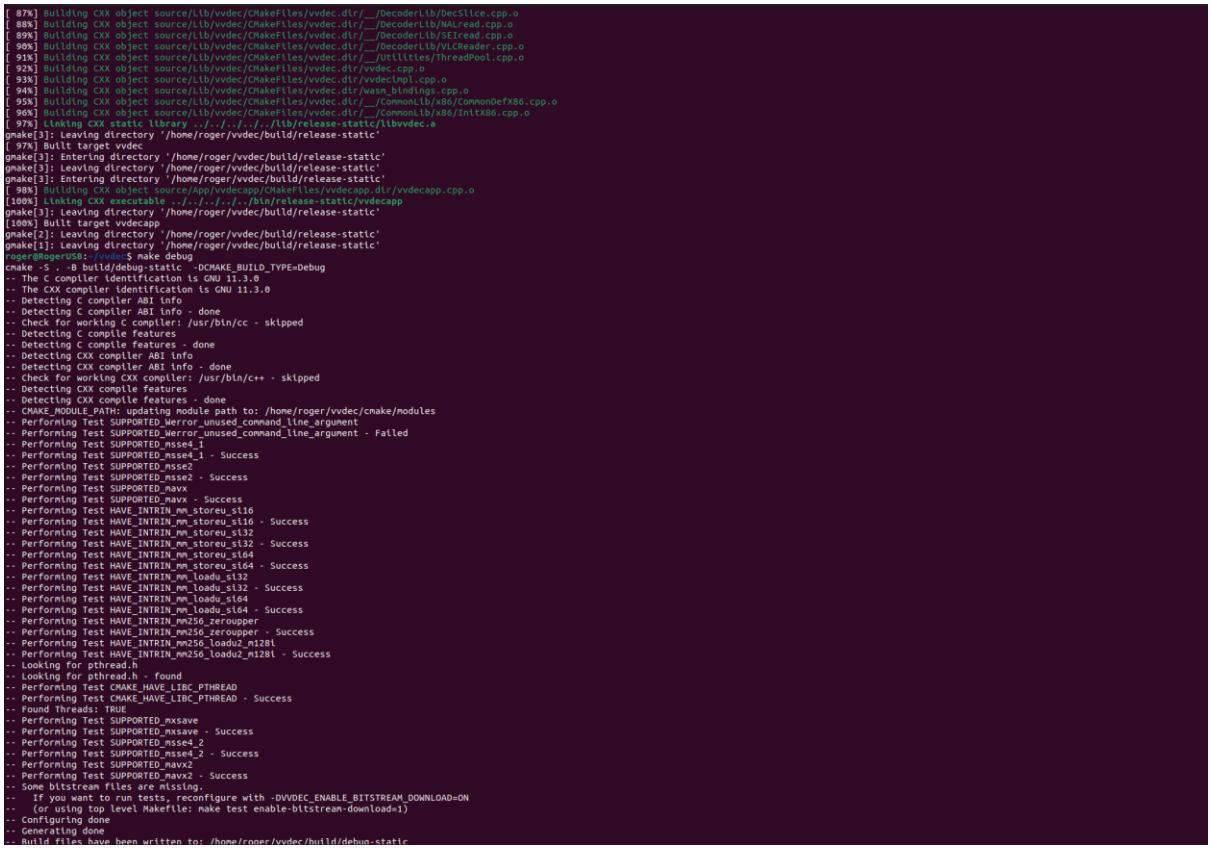

Figure A25: terminal procedure to install VVC decoder Part 3.

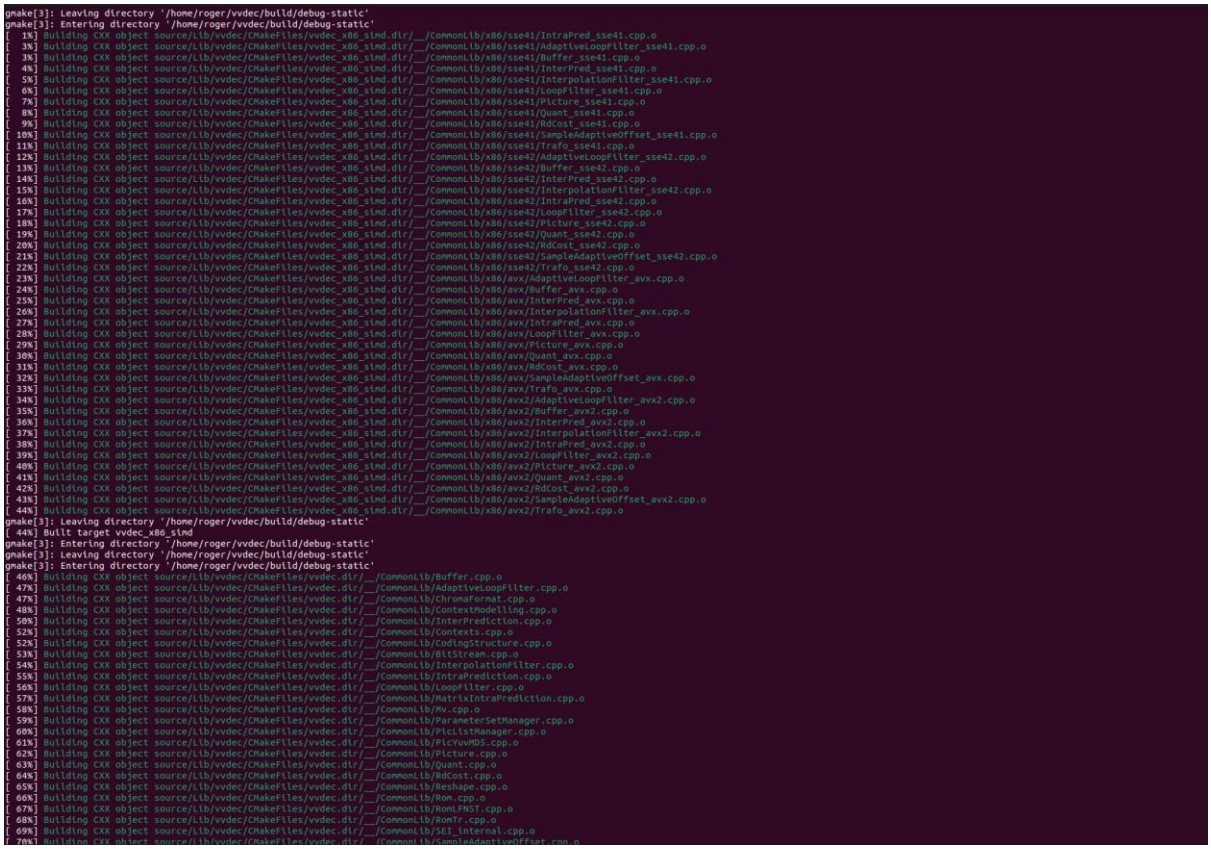

Figure A26: terminal procedure to install VVC decoder Part 4.

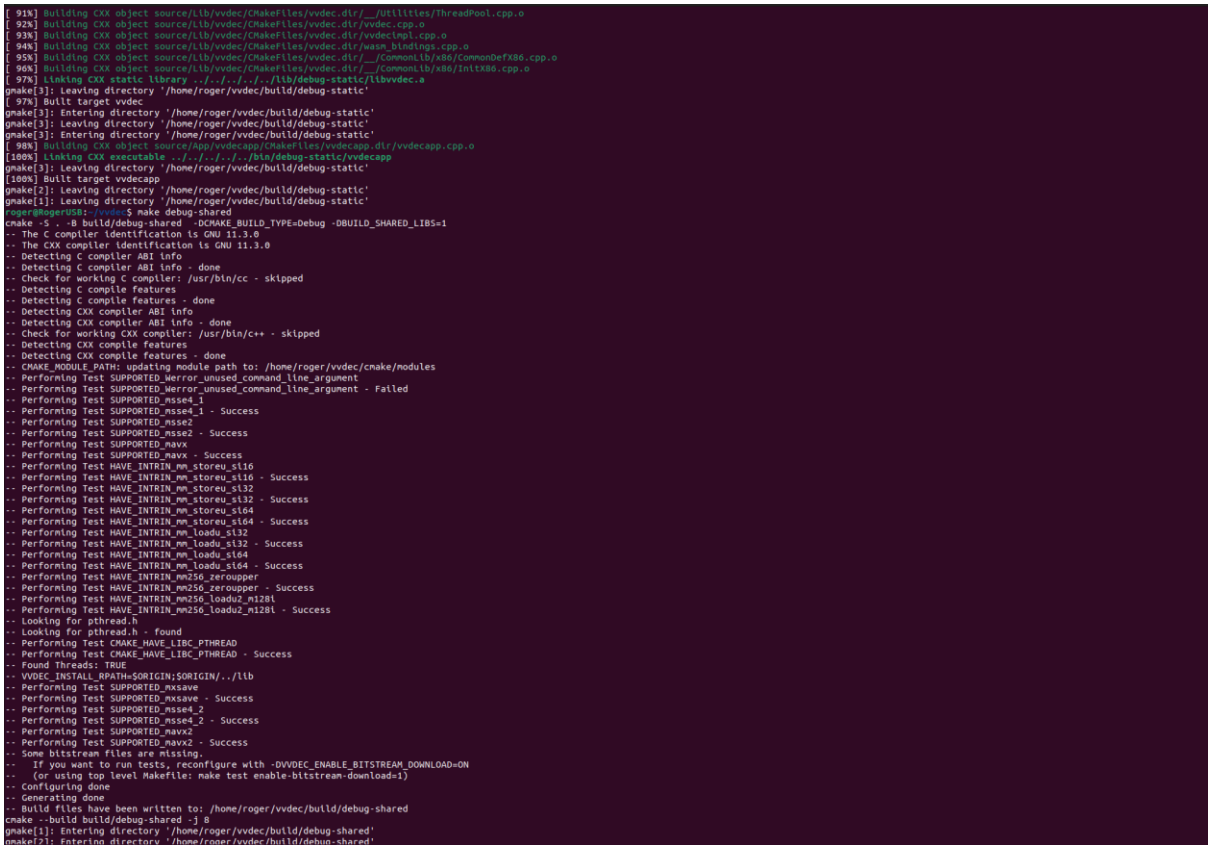

Figure A27: terminal procedure to install VVC decoder Part 5.

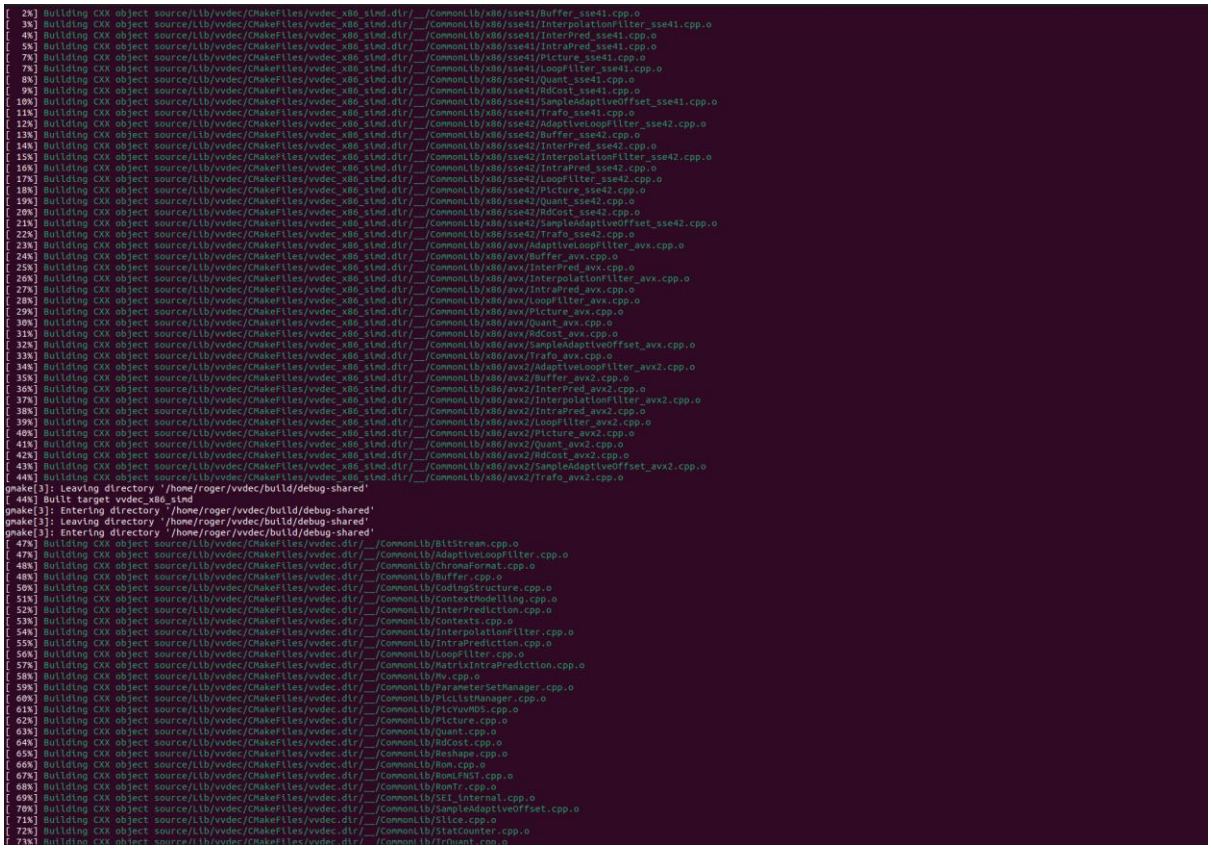

Figure A28: terminal procedure to install VVC decoder Part 6.

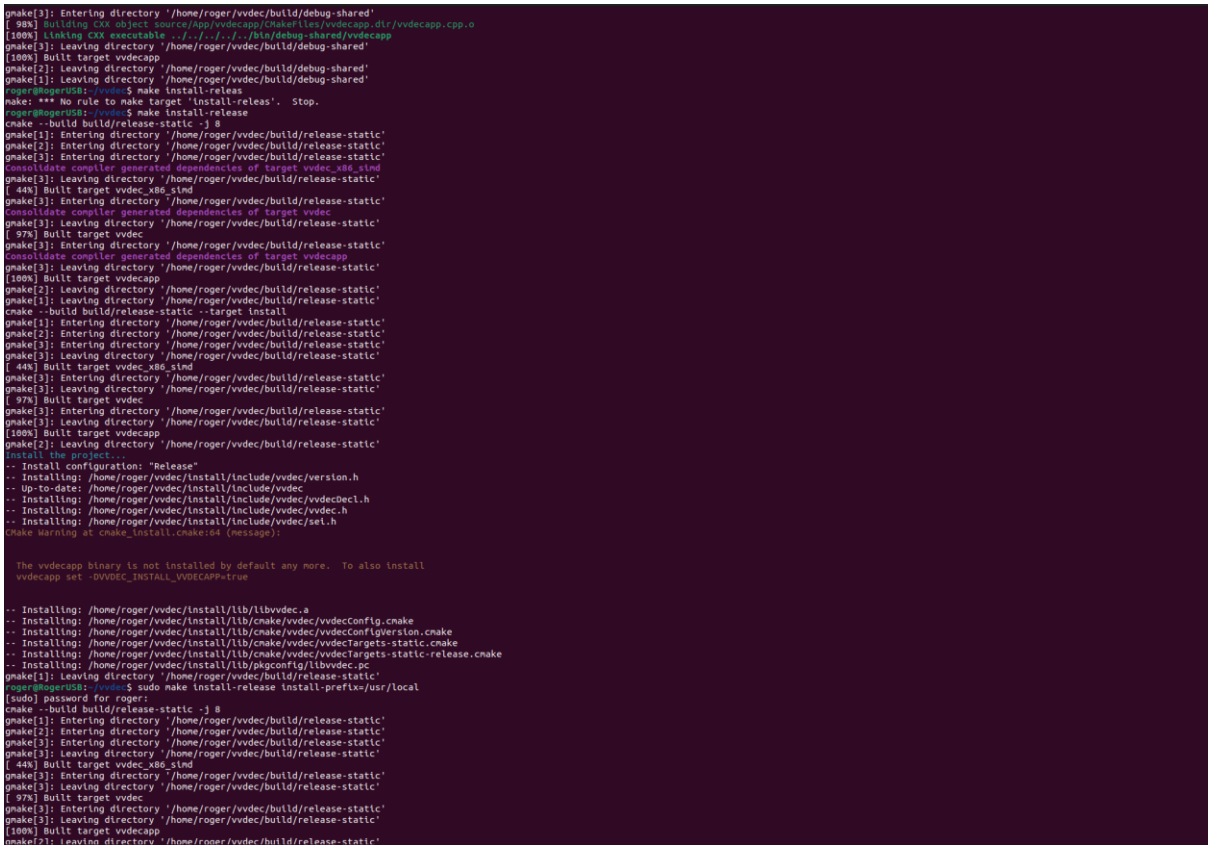

Figure A29: terminal procedure to install VVC decoder Part 7.

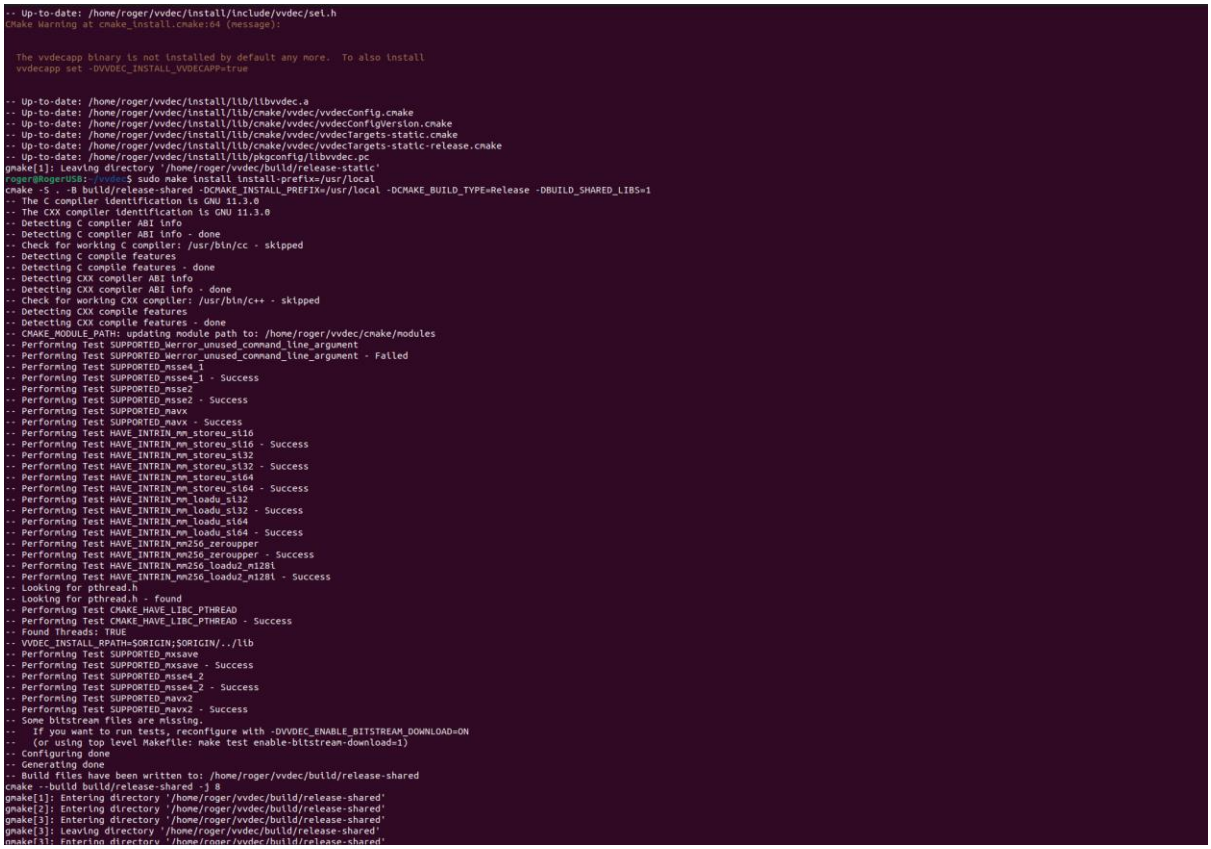

Figure A30: terminal procedure to install VVC decoder Part 8.

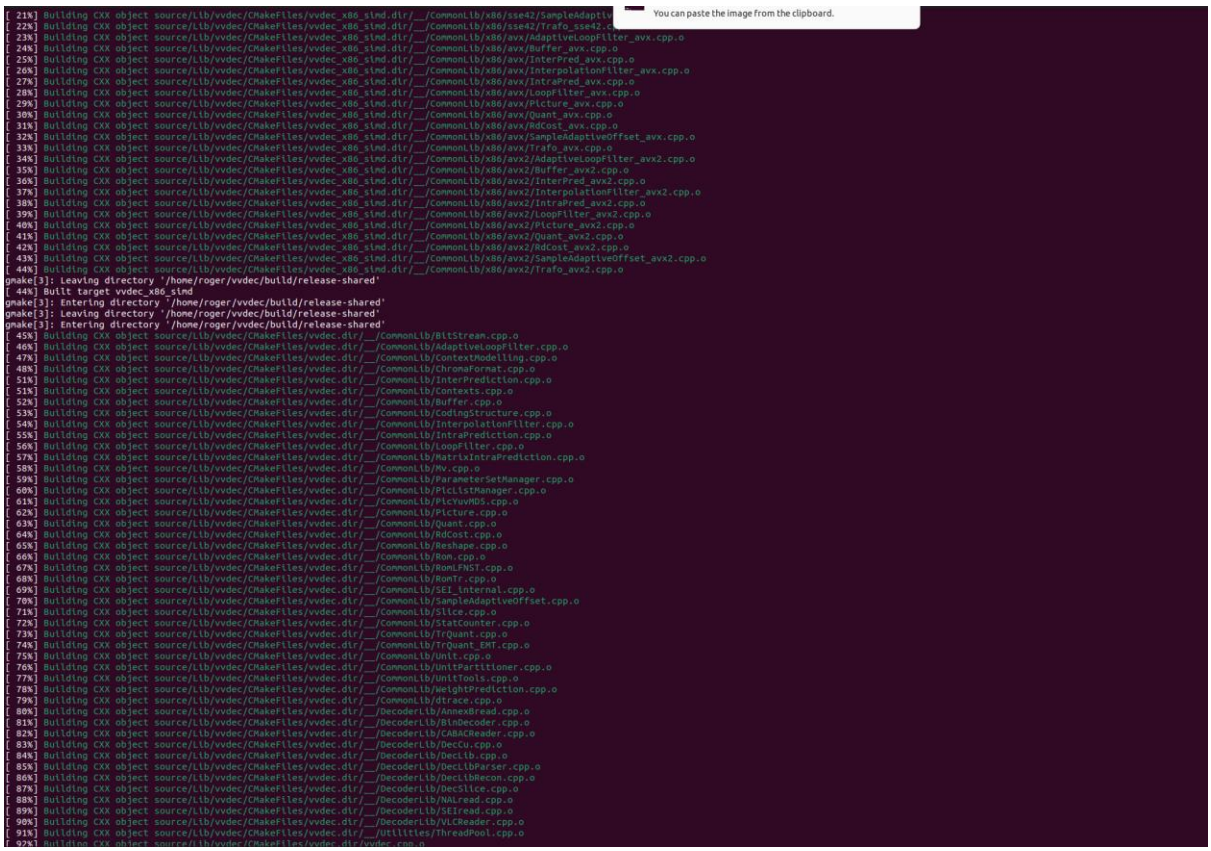

Figure A31: terminal procedure to install VVC decoder Part 9.

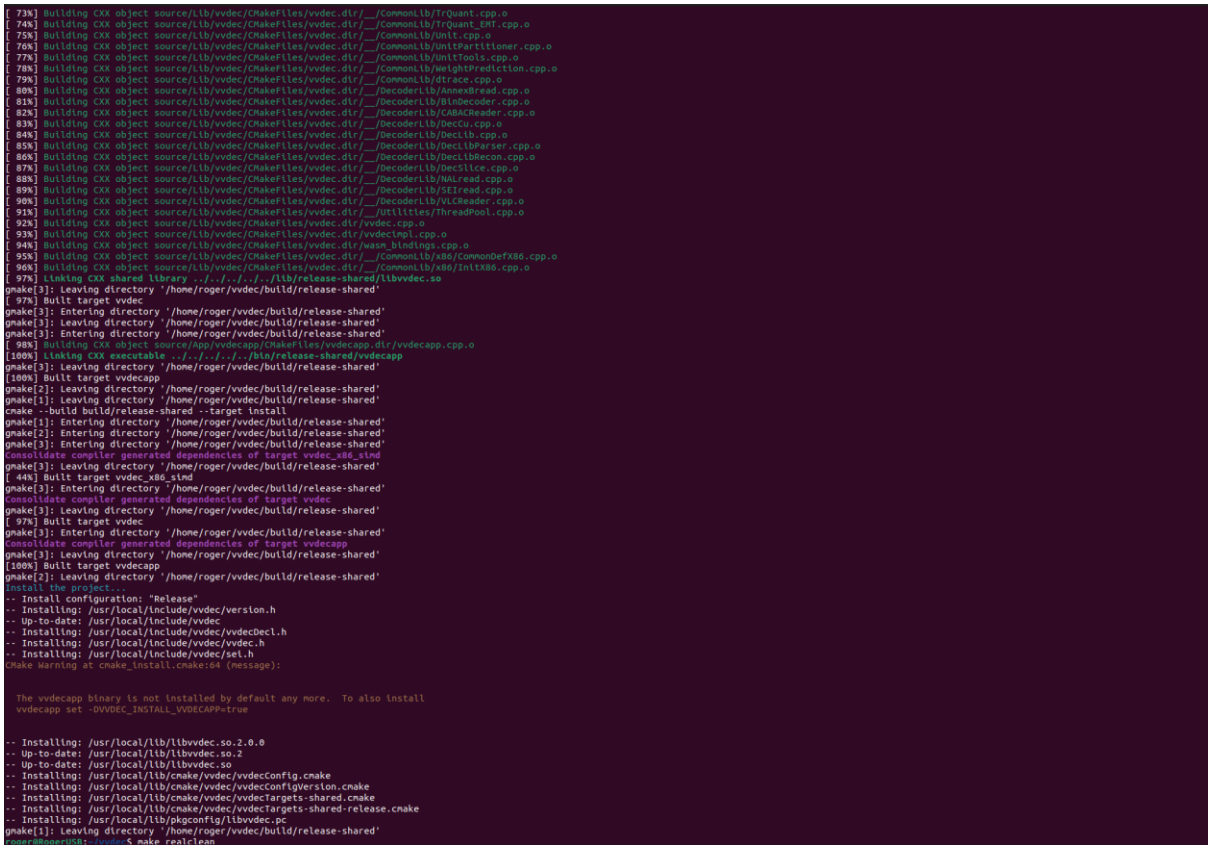

Figure A32: terminal procedure to install VVC decoder Part 10.

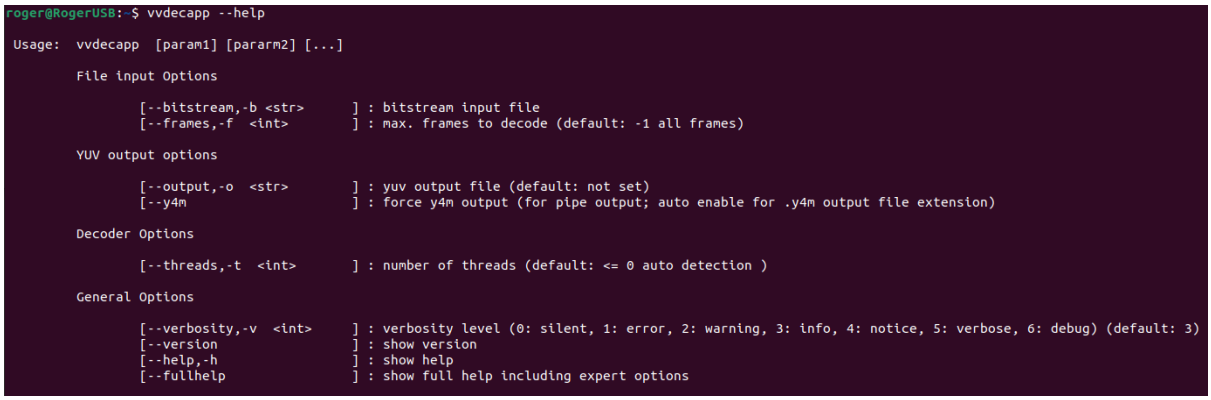

Figure A33: terminal procedure to install VVC decoder Part 11.

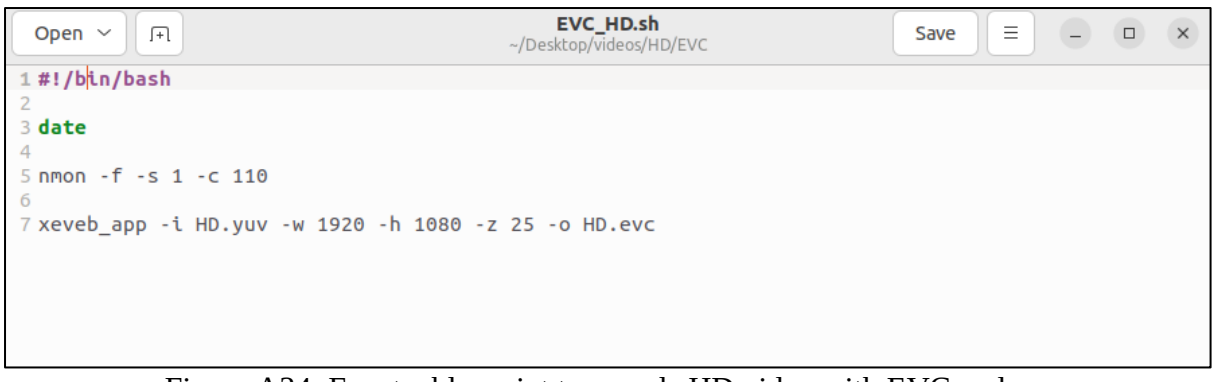

Figure A34: Exactuable script to encode HD video with EVC codec.

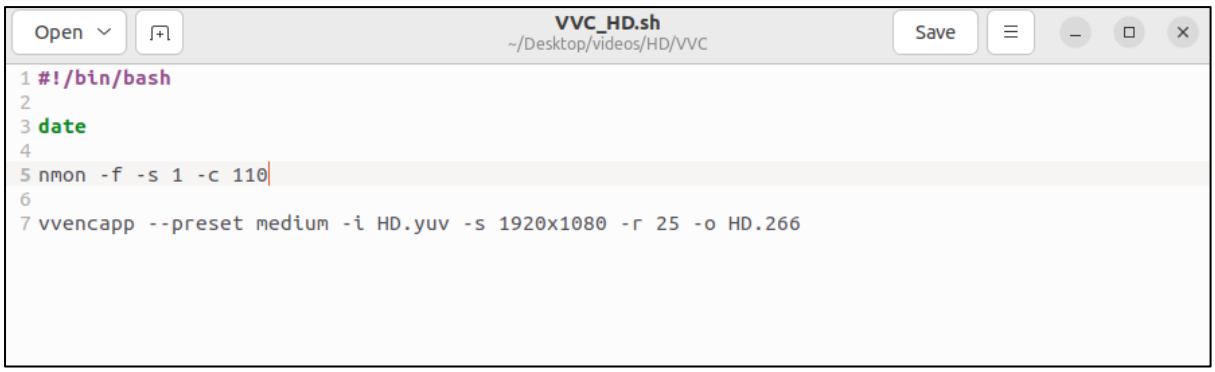

Figure A35: Exactuable script to encode HD video with VVC codec.

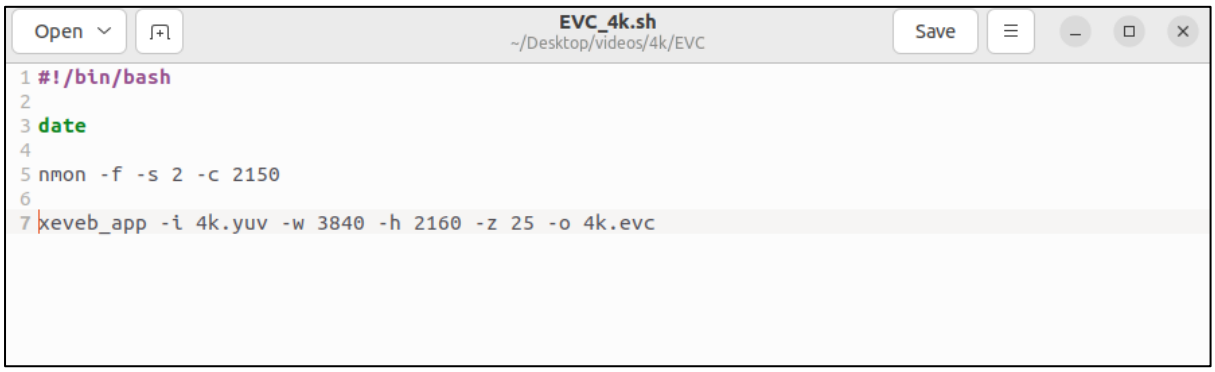

Figure A36: Exactuable script to encode 4k video with EVC codec.

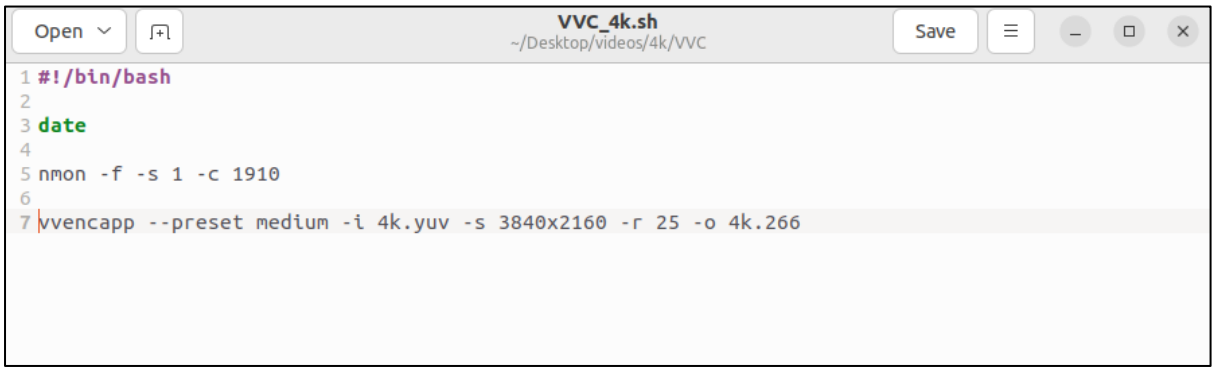

Figure A37: Exactuable script to encode 4k video with VVC codec.

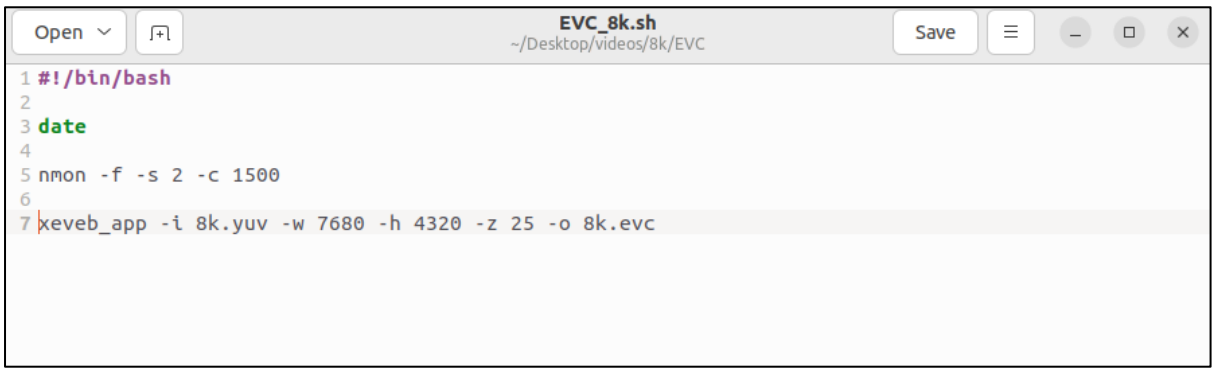

Figure A38: Exactuable script to encode 8k video with EVC codec.

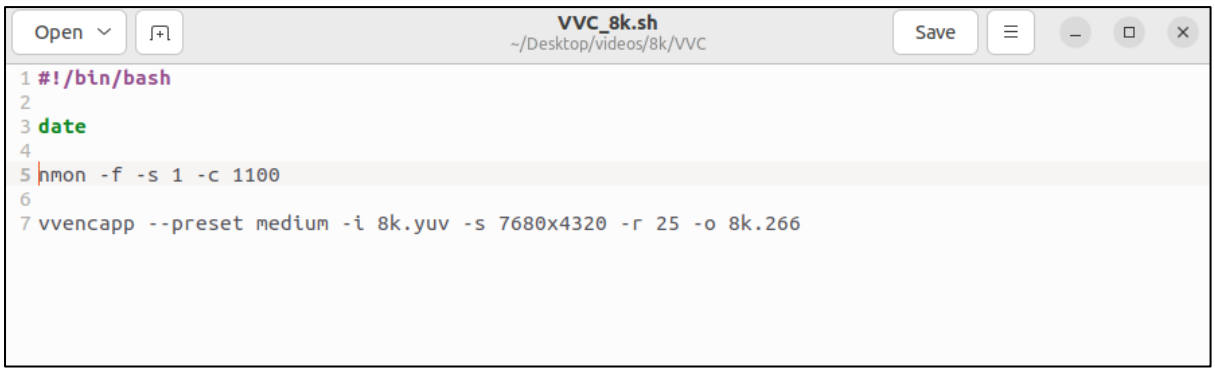

Figure A39: Exactuable script to encode 8k video with VVC codec.

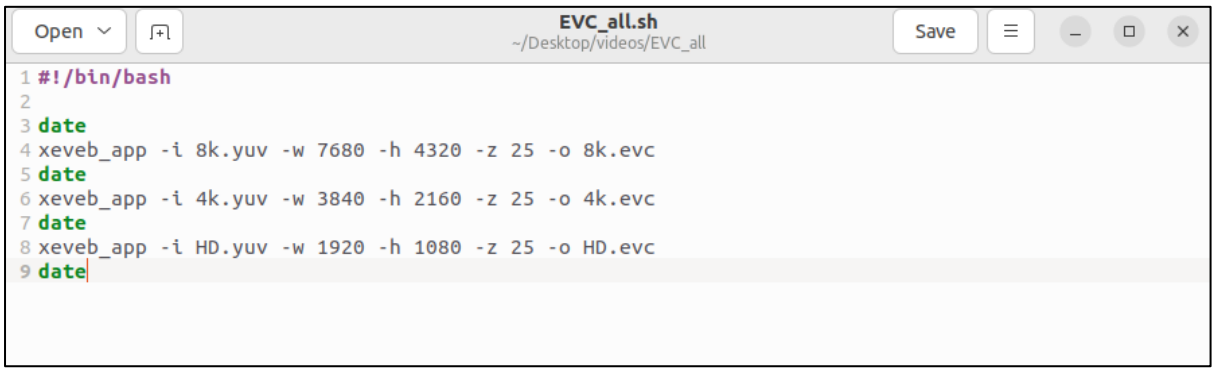

Figure A40: Exactuable script to encode all videos with EVC codec.

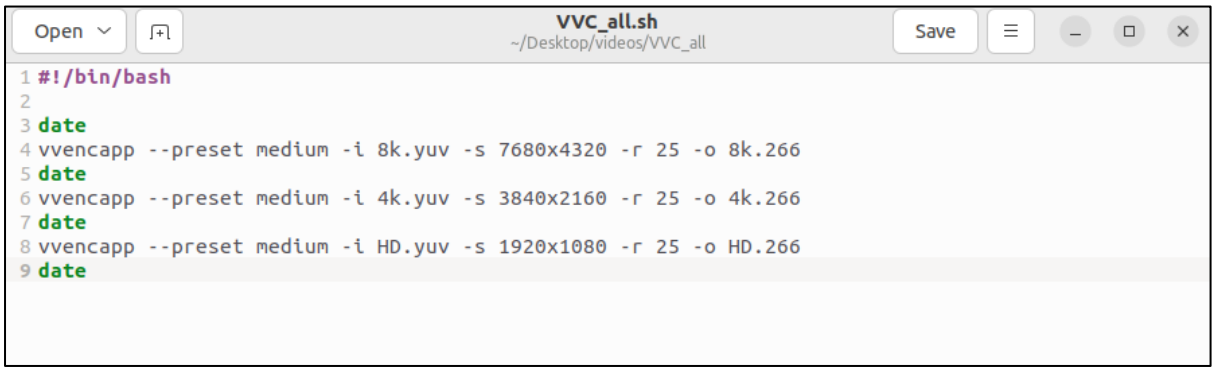

Figure A41: Exactuable script to encode all videos with VVC codec.

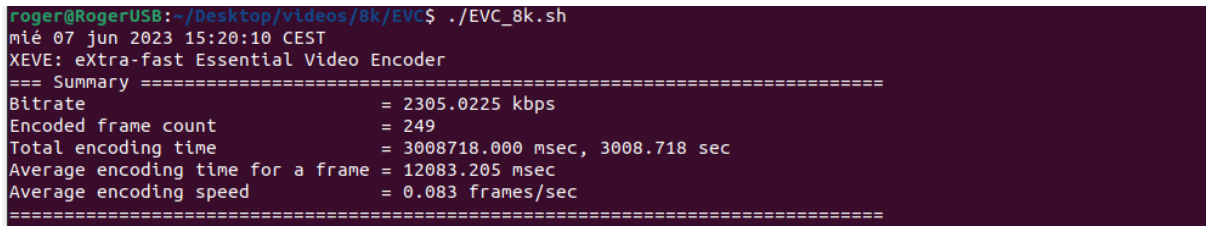

Figure A42: Terminal output with 8k resolution encoded video with EVC.

```
roger@RogerUSB:~/Desktop/videos/4k/EVC$ ./EVC_4k.sh
mié 07 jun 2023 13:28:32 CEST
XEVE: eXtra-fast Essential Video Encoder
=== Summary ========================================================================
Bitrate                          = 2361.4838 kbps
Encoded frame count              = 752
Total encoding time              = 4007898.000 msec, 4007.898 sec
Average encoding time for a frame = 5329.651 msec
Average encoding speed           = 0.188 frames/sec
=====================================================================================
roger@RogerUSB:~/Desktop/videos/4k/EVC$ ./nmonchart RogerUSB_230607_1328.nmon EVC_HD_1328.html
roger@RogerUSB:~/Desktop/videos/4k/EVC$
```

Figure A43: Terminal output with 4k resolution encoded video with EVC.

```
mié 07 jun 2023 12:55:18 CEST
XEVE: eXtra-fast Essential Video Encoder
=== Summary ========================================================================
Bitrate                          = 143.5062 kbps
Encoded frame count              = 241
Total encoding time              = 96500.000 msec, 96.500 sec
Average encoding time for a frame = 400.415 msec
Average encoding speed           = 2.497 frames/sec
=====================================================================================
roger@RogerUSB:~/Desktop/videos/HD/EVC$ ./nmonchart RogerUSB_230607_1255.nmon EVC_HD_1255.html
roger@RogerUSB:~/Desktop/videos/HD/EVC$
```

Figure A44: Terminal output with HD resolution encoded video with EVC.

```
roger@RogerUSB:~/Desktop/videos/8k/VVC$ ./VVC_8k.sh
mié 07 jun 2023 17:14:05 CEST
vvencapp: Fraunhofer VVC Encoder ver. 1.7.0 [Linux][GCC 11.3.0][64 bit][SIMD=AVX2]
Input     File : 8k.yuv
Bitstream File : 8k.266
Real Format    : 7680x4320  yuv420p  25 Hz  SDR  249 frames
               : encode 249 frames
Rate Control   : QP 32
CODING TOOL CFG: CTU128 QT44BTT2221111 IBD:1 SAO:1 ALF:1 (NonLinLuma:0 NonLinChr:0) CCALF:1 Tiles:1x1 Slices:1 WPP:0 WPP-Substreams:1 T
MVP:1 DQ:1 SDH:0 CST:1 BDOF:1 DMVR:1 MTSImplicit:1 SBT:0 JCbCr:1 CabacInitPresent:0 AMVR:5 SMVD:3 LMCS:2 (Signal:SDR Opt:0) CIIP:0 MIP:
1 AFFINE:2 (PROF:1, Type:1) MMVD:3 DisFracMMVD:1 SbTMVP:1 GPM:3 LFNST:1 MTS:0 ISP:3 TS:2 TSLog2MaxSize:4 useChromaTS:0 BDPCM:2 IBC:2 BC
W:0
ENC. ALG. CFG: QPA:1 HAD:1 RDQ:1 RDQTS:1 ASR:1 MinSearchWindow:96 EDO:2 MCTF:2 BIM:1
PRE-ANALYSIS CFG: STA:1 LeadFrames:0 TrailFrames:0
FAST TOOL CFG: ECU:0 FEN:3 FDM:1 FastSearch:4 (SCC:2) LCTUFast:1 FastMrg:3 PBIntraFast:1 AMaxBT:0 FastQtBtEnc:1 ContentBasedFastQtbt:1
FastMIP:3 FastIntraTools:1 IntraEstDecBit:2 FastLocalDualTree:1 IntegerET:0 FastSubPel:1 ReduceFilterME:2 QtbttExtraFast:3 FastTTSplit:
5 IBCFastMethod:3 FIM:0 ALFSpeed:0 QuantThr: 4
RATE CONTROL CFG: RateControl:0 QP:32 LookAhead:0
PARALLEL PROCESSING CFG: NumThreads:8 MaxParallelFrames:4 WppBitEqual:1 WF:0

started @ Wed Jun  7 17:14:05 2023

POC     0 TId: 0 (  IDR_N_LP, I-SLICE, QP 21, TF 2)    2445952 bits [Y 51.4542 dB    U 52.5775 dB    V 52.8334 dB] [ET    41 ] [L0 ] [L
1 ]
POC    32 TId: 0 (       CRA, I-SLICE, QP 25, TF 2)    1457328 bits [Y 46.1205 dB    U 49.0082 dB    V 48.0196 dB] [ET    32 ] [L0 ] [L
1 ]
POC    16 TId: 1 (      RASL, B-SLICE, QP 29, TF 1)     125352 bits [Y 46.4432 dB    U 49.3675 dB    V 48.6554 dB] [ET    12 ] [L0 0 32
 ] [L1 32 0 ]
POC     8 TId: 2 (      RASL, B-SLICE, QP 30, TF 0)      77912 bits [Y 46.3777 dB    U 49.2477 dB    V 48.7798 dB] [ET    19 ] [L0 0 16
 ] [L1 16 32 ]
POC     4 TId: 3 (      RASL, B-SLICE, QP 33)            26200 bits [Y 48.0262 dB    U 50.3632 dB    V 50.4430 dB] [ET    13 ] [L0 0 8
] [L1 8 16 ]
POC     2 TId: 4 (      RASL, B-SLICE, QP 35)             8032 bits [Y 49.6060 dB    U 51.5781 dB    V 52.0322 dB] [ET     8 ] [L0 0 4
] [L1 4 8 ]
POC     1 TId: 5 (      RASL, B-SLICE, QP 36)             5032 bits [Y 50.0295 dB    U 51.8675 dB    V 52.3251 dB] [ET     7 ] [L0 0 2
] [L1 2 4 ]
POC     3 TId: 5 (      RASL, B-SLICE, QP 36)             5472 bits [Y 48.4481 dB    U 50.8400 dB    V 51.0551 dB] [ET     7 ] [L0 2 0
] [L1 4 8 ]
```

Figure A45: Terminal output with 8k resolution encoded video with VVC Part 1.

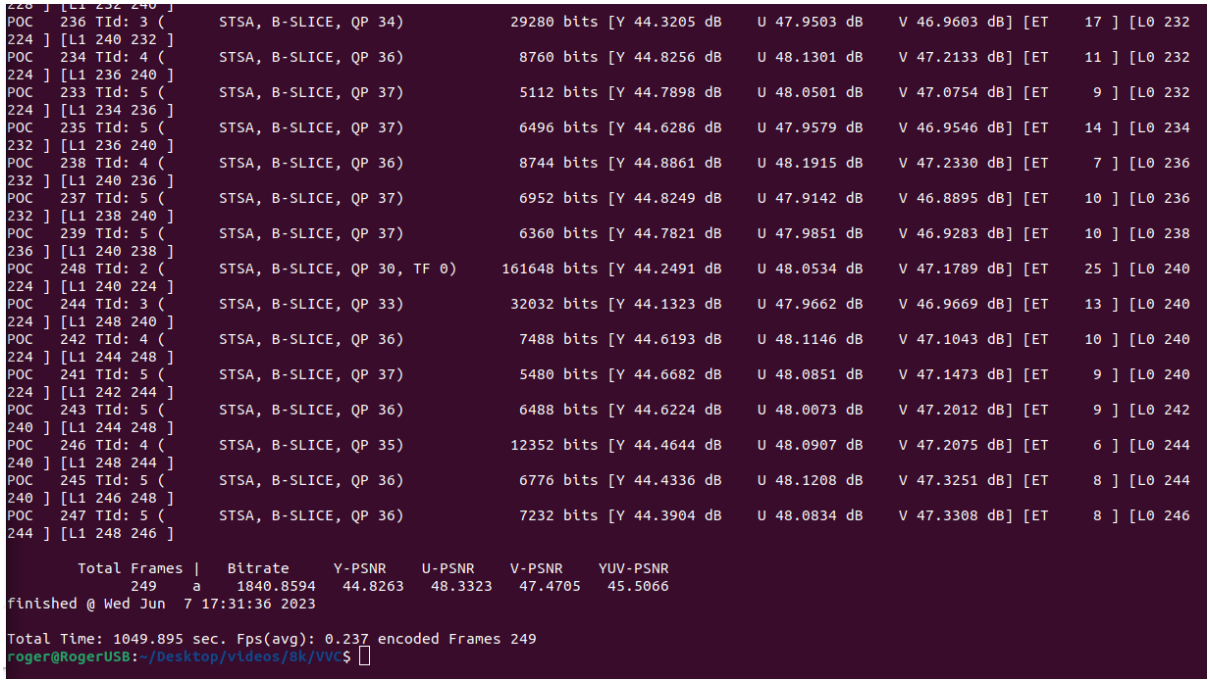

Figure A46: Terminal output with 8k resolution encoded video with VVC Part 2.

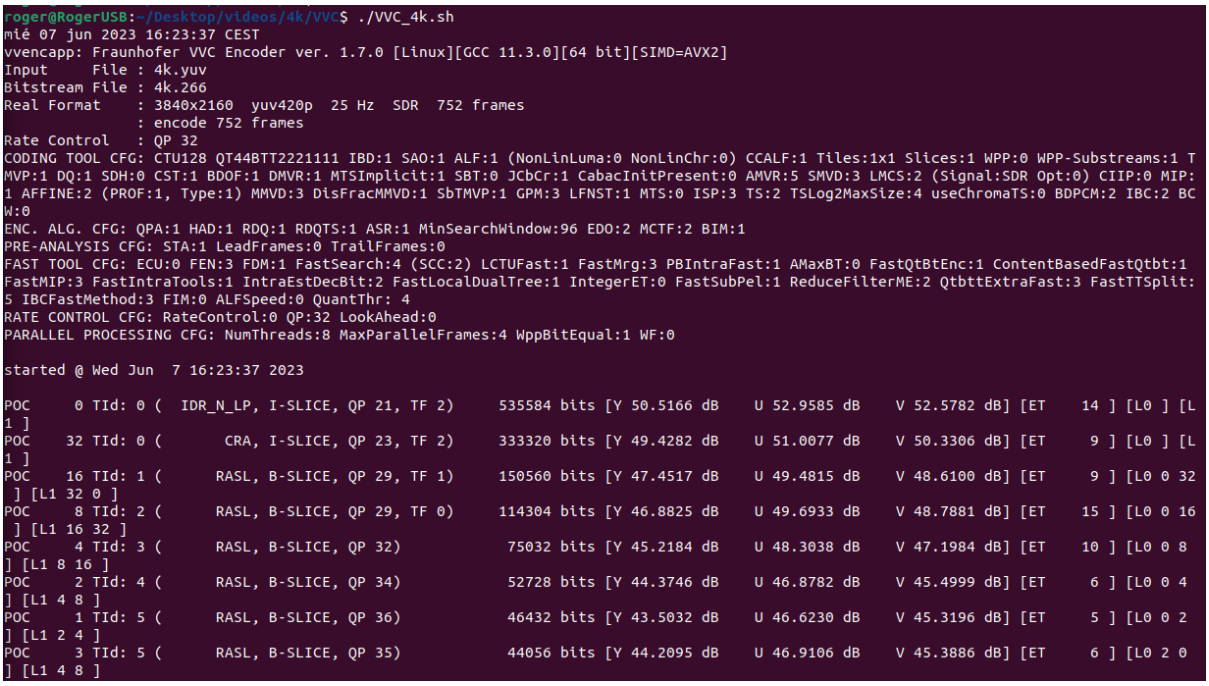

Figure A47: Terminal output with 4k resolution encoded video with VVC Part 1.

```
732 ] [L1 736 734 ]
POC   744 TId: 2 (      STSA, B-SLICE, QP 29, TF 0)      97968 bits [Y 48.0563 dB    U 50.6039 dB    V 49.5717 dB] [ET     4 ] [L0 736
] [L1 736 ]
POC   740 TId: 3 (      STSA, B-SLICE, QP 32)            66696 bits [Y 45.8175 dB    U 48.7212 dB    V 48.0015 dB] [ET     4 ] [L0 736
744 ] [L1 744 736 ]
POC   738 TId: 4 (      STSA, B-SLICE, QP 34)            47752 bits [Y 45.5189 dB    U 48.1558 dB    V 47.3365 dB] [ET     6 ] [L0 736
740 ] [L1 740 744 ]
POC   737 TId: 5 (      STSA, B-SLICE, QP 35)            38640 bits [Y 44.9394 dB    U 47.1681 dB    V 46.2272 dB] [ET     5 ] [L0 736
738 ] [L1 738 740 ]
POC   739 TId: 5 (      STSA, B-SLICE, QP 35)            41616 bits [Y 44.6766 dB    U 47.3482 dB    V 46.1557 dB] [ET     5 ] [L0 738
736 ] [L1 740 744 ]
POC   742 TId: 4 (      STSA, B-SLICE, QP 33)            46992 bits [Y 45.9249 dB    U 48.6767 dB    V 47.3999 dB] [ET     6 ] [L0 740
736 ] [L1 744 740 ]
POC   741 TId: 5 (      STSA, B-SLICE, QP 35)            37800 bits [Y 45.1100 dB    U 47.5401 dB    V 46.2243 dB] [ET     5 ] [L0 740
736 ] [L1 742 744 ]
POC   743 TId: 5 (      STSA, B-SLICE, QP 35)            39360 bits [Y 44.9354 dB    U 47.6088 dB    V 46.0871 dB] [ET     5 ] [L0 742
740 ] [L1 744 742 ]
POC   748 TId: 3 (      STSA, B-SLICE, QP 32)            68304 bits [Y 46.1678 dB    U 49.0337 dB    V 47.9134 dB] [ET     4 ] [L0 744
736 ] [L1 744 736 ]
POC   746 TId: 4 (      STSA, B-SLICE, QP 33)            50432 bits [Y 45.5781 dB    U 48.6284 dB    V 47.1672 dB] [ET     6 ] [L0 744
736 ] [L1 748 744 ]
POC   745 TId: 5 (      STSA, B-SLICE, QP 34)            39080 bits [Y 45.4423 dB    U 48.4498 dB    V 47.0694 dB] [ET     4 ] [L0 744
736 ] [L1 746 748 ]
POC   747 TId: 5 (      STSA, B-SLICE, QP 35)            39992 bits [Y 44.9910 dB    U 47.5362 dB    V 46.2923 dB] [ET     5 ] [L0 746
744 ] [L1 748 746 ]
POC   750 TId: 4 (      STSA, B-SLICE, QP 34)            51440 bits [Y 44.6508 dB    U 48.6701 dB    V 47.3082 dB] [ET    10 ] [L0 748
744 ] [L1 748 744 ]
POC   749 TId: 5 (      STSA, B-SLICE, QP 35)            43048 bits [Y 44.8942 dB    U 47.4614 dB    V 46.0435 dB] [ET     5 ] [L0 748
744 ] [L1 750 748 ]
POC   751 TId: 5 (      STSA, B-SLICE, QP 34)            42104 bits [Y 44.7110 dB    U 47.5334 dB    V 46.2608 dB] [ET     2 ] [L0 750
748 ] [L1 750 748 ]

       Total Frames |   Bitrate     Y-PSNR    U-PSNR    V-PSNR    YUV-PSNR
             752    a    1566.5394   45.5059   48.1133   46.9998   45.8982
finished @ Wed Jun  7 16:54:12 2023

Total Time: 1834.664 sec. Fps(avg): 0.410 encoded Frames 752
roger@RogerUSB:~/Desktop/videos/4k/VVC$
```

Figure A48: Terminal output with 4k resolution encoded video with VVC Part 2.

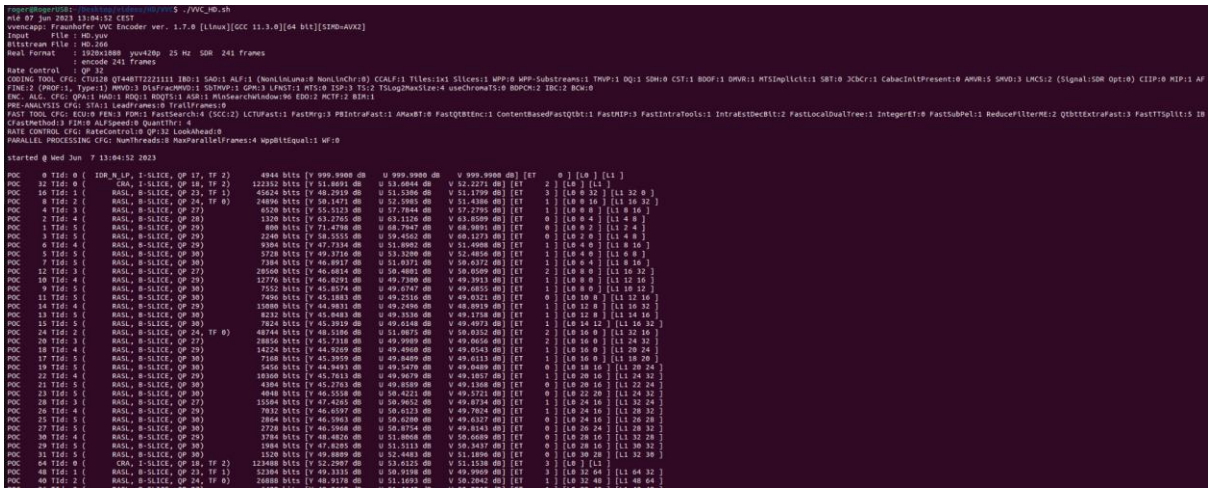

Figure A49: Terminal output with HD resolution encoded video with VVC Part 1.

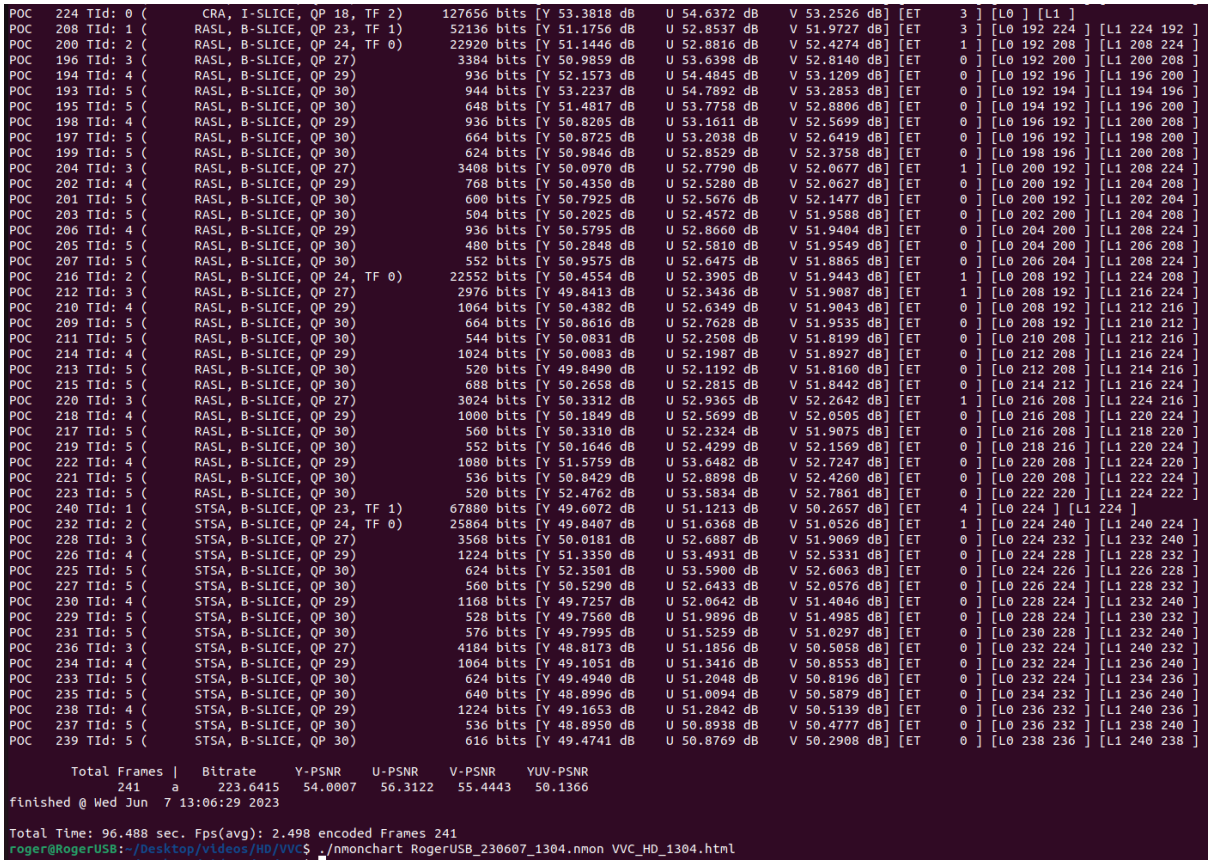

Figure A50: Terminal output with HD resolution encoded video with VVC Part 2.

```
roger@RogerUSB:~/Desktop/videos/EVC_all$ ./EVC_all.sh
lun 12 jun 2023 18:30:01 CEST
XEVE: eXtra-fast Essential Video Encoder
=== Summary ================================================================
Bitrate                          = 2305.0225 kbps
Encoded frame count              = 249
Total encoding time              = 2952901.000 msec, 2952.901 sec
Average encoding time for a frame = 11859.040 msec
Average encoding speed           = 0.084 frames/sec
============================================================================
lun 12 jun 2023 19:19:49 CEST
XEVE: eXtra-fast Essential Video Encoder
=== Summary ================================================================
Bitrate                          = 2361.4838 kbps
Encoded frame count              = 752
Total encoding time              = 4027120.000 msec, 4027.120 sec
Average encoding time for a frame = 5355.213 msec
Average encoding speed           = 0.187 frames/sec
============================================================================
lun 12 jun 2023 20:27:23 CEST
XEVE: eXtra-fast Essential Video Encoder
=== Summary ================================================================
Bitrate                          = 143.5062 kbps
Encoded frame count              = 241
Total encoding time              = 95154.000 msec, 95.154 sec
Average encoding time for a frame = 394.830 msec
Average encoding speed           = 2.533 frames/sec
============================================================================
lun 12 jun 2023 20:29:00 CEST
roger@RogerUSB:~/Desktop/videos/EVC_all$
```

Figure A51: Terminal output of all encoded videos consecutively with EVC.

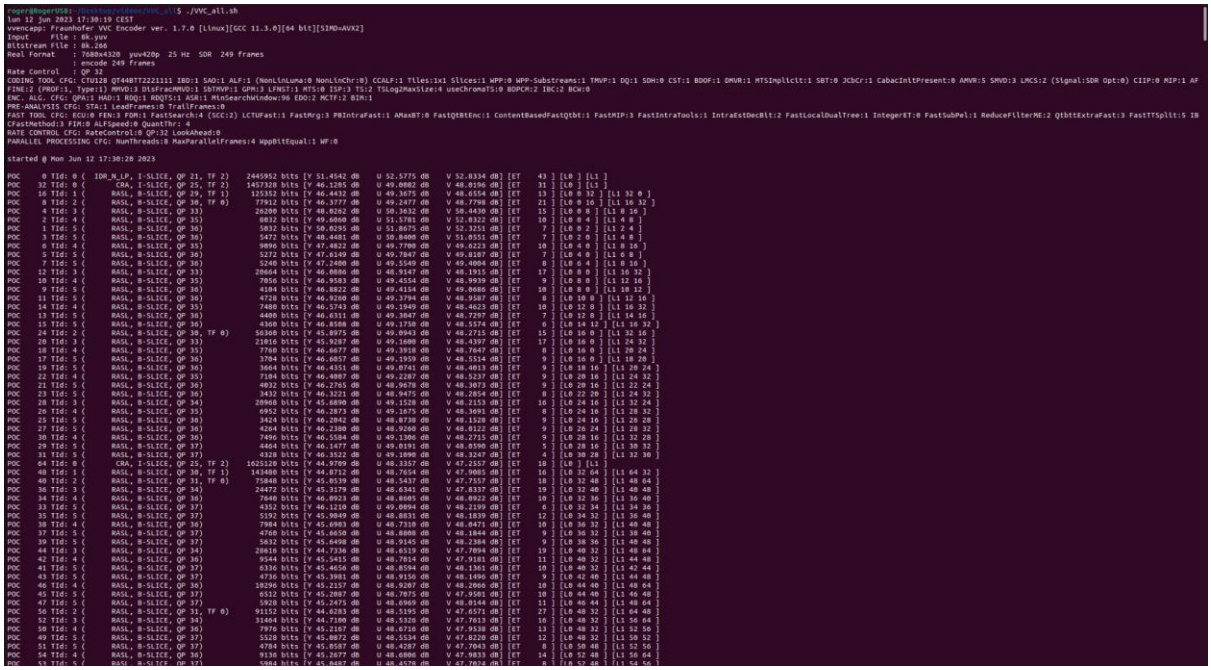

Figure A52: Terminal output of all encoded videos consecutively with VVC Part 1.

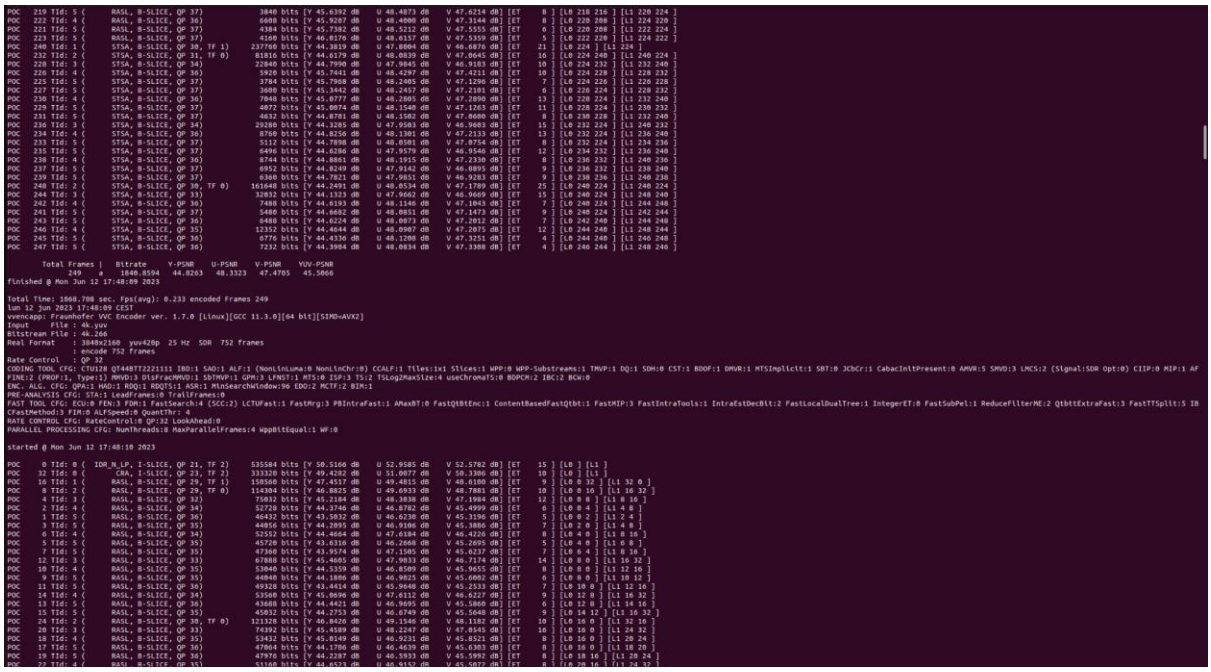

Figure A53: Terminal output of all encoded videos consecutively with VVC Part 2.

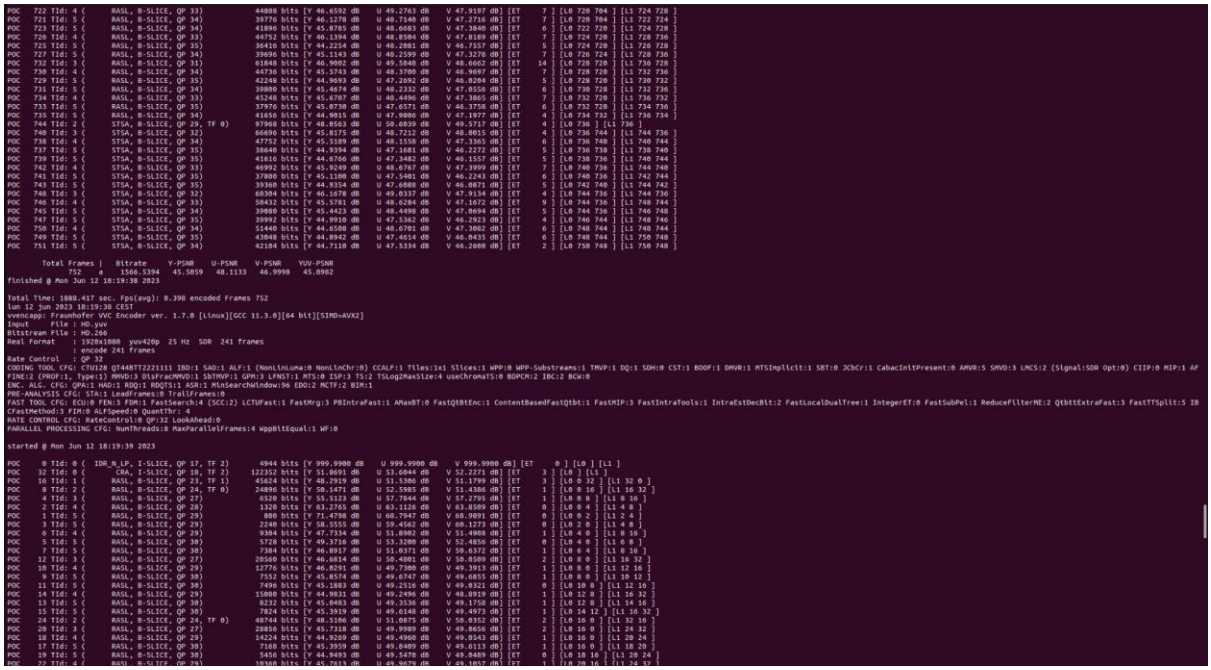

Figure A54: Terminal output of all encoded videos consecutively with VVC Part 3.

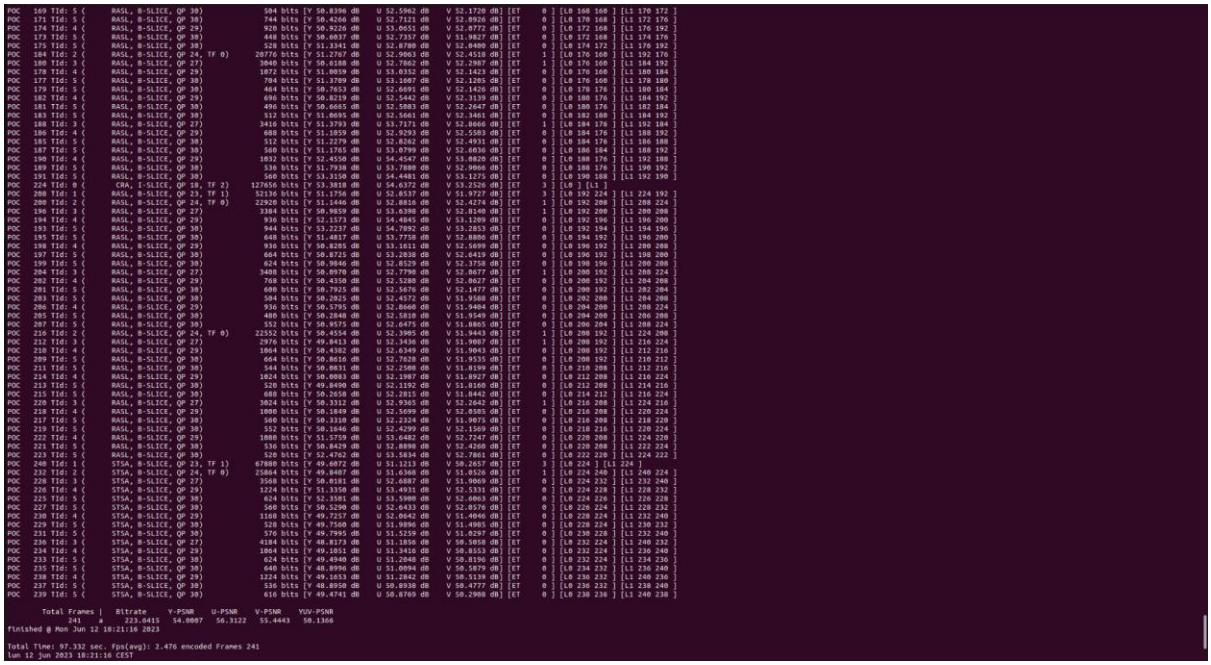

Figure A55: Terminal output of all encoded videos consecutively with VVC Part 4.

ANNEX B

Electrical Consumption Results

Figure B1: Electrical Consumption of computer in 1 hour without any executing process.

Figure B2: Electrical Consumption for 8k resolution with EVC encoder

Figure B3: Electrical Consumption for 8k resolution with VVC encoder

Figure B4: Electrical Consumption for 4k resolution with EVC encoder

Figure B5: Electrical Consumption for 4k resolution with VVC encoder

Figure B6: Electrical Consumption for HD resolution with EVC encoder

Figure B7: Electrical Consumption for HD resolution with VVC encoder

Edificio / Sede	Nombre de la comercializadora suministradora de energía[1]	¿Dispone de Garantía de Origen (GdO)?[2]	Dato de consumo kWh	Factor Mix eléc.(3) kg CO2e/kWh	Emisiones (4) kg CO2e
	ENDESA ENERGÍA S.A.U.	GdO cogeneración de alta eficiencia		0,30	0,00
	ENDESA ENERGÍA S.A.U.	GdO energía renovable		0,00	0,00
	ENDESA ENERGÍA S.A.U.	No		0,26	0,00
	ENDESA ENERGÍA RENOVABLE, S.L.	GdO cogeneración de alta eficiencia		0,30	0,00
	ENDESA ENERGÍA RENOVABLE, S.L.	GdO energía renovable		0,00	0,00
	ENDESA ENERGÍA RENOVABLE, S.L.	No		0,00	0,00
	IBERDROLA CLIENTES, S.A.U.	GdO cogeneración de alta eficiencia		0,30	0,00
	IBERDROLA CLIENTES, S.A.U.	GdO energía renovable		0,00	0,00
	IBERDROLA CLIENTES, S.A.U.	No		0,23	0,00
	IBERDROLA SERVICIOS ENERGETICOS, S.A.U.	GdO cogeneración de alta eficiencia		0,30	0,00
	IBERDROLA SERVICIOS ENERGETICOS, S.A.U.	GdO energía renovable		0,00	0,00
	IBERDROLA SERVICIOS ENERGETICOS, S.A.U.	No		0,00	0,00
	NATURGY IBERIA, S.A.	GdO cogeneración de alta eficiencia		0,30	0,00
	NATURGY IBERIA, S.A.	GdO energía renovable		0,00	0,00
	NATURGY IBERIA, S.A.	No		0,26	0,00
	NATURGY RENOVABLES, S.LU.	GdO cogeneración de alta eficiencia		0,30	0,00
	NATURGY RENOVABLES, S.LU.	GdO energía renovable		0,00	0,00
	NATURGY RENOVABLES, S.LU.	No		0,00	0,00
					0,00
	REPSOL COMERCIALIZADORA DE ELECTRICIDAD Y GAS, S.L.U	GdO cogeneración de alta eficiencia		0,30	0,00
	REPSOL COMERCIALIZADORA DE ELECTRICIDAD Y GAS, S.L.U	GdO energía renovable		0,00	0,00
	REPSOL COMERCIALIZADORA DE ELECTRICIDAD Y GAS, S.L.U	No		0,00	0,00
	EKILUZ ENERGÍA COMERCIALIZADORA, S.L.	GdO cogeneración de alta eficiencia		0,30	0,00
	EDP ESPAÑA, S.A	GdO energía renovable		0,00	0,00
	EDP ESPAÑA, S.A	No		0,26	0,00

Figure B8: Electrical Consumption for 8k resolution with EVC encoder