Performance and energy efficiency of new generation video decoding standards on low power consumption-based multi-cores architectures

Mohammed Bey Ahmed Khernache

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : « *Informatique* »

Par

« Mohammed BEY AHMED KHERNACHE »

« Performance and Energetical Efficiency of New Generation Video Decoding Standards on Low Power Consumption-based Multi-cores Architectures »

Thèse présentée et soutenue à « ENSTA Bretagne, Brest », le « 15/12/2021 »
Unité de recherche : « Lab-STICC »
Thèse N° : « 616 »

Rapporteurs avant soutenance :

Cécile BELLEUDY	Maître de conférences, HDR, Université de Côte d'Azur - LEAT
Smail NIAR	Professeur, Université Polytechnique Hauts de France - LAMIH

Composition du Jury :

Président :	Abdoulaye GAMATIÉ	Directeur de recherche CNRS - LIRMM Montpellier
Dir. de thèse :	Jalil BOUKHOBZA	Professeur, ENSTA Bretagne
Co-dir. de thèse :	Daniel MENARD	Professeur, INSA Rennes
Co-encadrant :	Yahia BENMOUSSA	Maitre de conférences, Université de Boumerdès

Invité(s) :

Philippe COUSSY, Professeur, Université de Bretagne-Sud - Lab-STICC

ACKNOWLEDGEMENT

In the period of working for my doctoral studies, many people were involved. First of all, I would like to thank my thesis supervisors: Pr. Jalil BOUKHOBZA (thesis director), Pr. Daniel MENARD (thesis co-director), and Dr. Yahia BENMOUSSA for their guidance, support, and trust. My acknowledgment and deep admiration to them.

I want to express my gratitude to Dr. Cécile BELLEUDY and Pr. Smail NIAR for being part of the Ph.D. defense committee and for taking time to review my thesis. I also want to thank DR. Abdoulaye GAMATIÉ and Pr. Philippe COUSSY for presiding and being invited member of my Ph.D. defense committee.

I would like to thank Pr. Emmanuel CASSEAU and Dr. Johann LAURENT for their annual evaluation of my thesis. In particular, their feedback and guidance were very helpful for my thesis. They are not only science-wise role models but also kind people who worry about the well-being of me during my thesis.

A special thank to Pr. Philippe COUSSY for his support during my thesis. He managed the adminstrative issues of the Lab-STICC participation to the EFIGI project. He also helped me change the direction of my Ph.D. without which the my thesis would be stopped in the middle-way.

Many thanks thanks to Mme. Virginie GUILLET, Mme. Béatrice GUERN, and Mme. Noluenn CHAUVIN for welcoming me to Lab-STICC and UBS. They did a great job to deal with all the administrative stuff related to my Ph.D. project.

My thanks go also to all my very talented office-mates during this work, starting from ENSIBS and Christian Huygens research centre in Lorient, and then at UBO and finally at ENSTA Bretagne in Brest. I learned a lot from them. I also passed very nice moments together with them: meetings, outings, convivial breaks, ...

Moreover, I am very grateful to all the people in different associations where I could have a volunteering experience during my thesis: ICP, Sterenn, Syklett, ACIL, BaPaV, and AFEV. It was a very rich experience where I could do my hobbies.

I would like to thank all the people from the EFIGI project. I really appreciated our technical discussions during the regular meetings. A special thank to BPi France which funded my thesis.

Finally, I want to dedicate the efforts of my doctoral work to my bride, my brothers and sisters, my friends, and the favorite person all over the earth, my mother, who patiently waited for me, even being far from me, during these four years.

I do not want to miss the name of anyone. However, this small space is not enough to list all the people to whom I am grateful.

For all of them I present my gratitude again and again.

Table of Contents

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

The present thesis addresses the issue of energy consumption of video decoding on low power mobile System on Chip (SoC). First of all, the thesis consists in understanding the energy consumption of video decoding by characterizing it. Then, it consists in proposing mechanisms and strategies for optimizing the energy consumption of video decoding.

The structure of this chapter is organized as follows. Section 1.1 presents the context and the motivation of our work. Then, Section 1.2 presents the problem statement that we willing to solve through this thesis. After that, Section 1.3 defines the scope and the position of our work compared to those of the state of the art. Next, Section 1.4 introduces our contributions that resolved the problem stated. Finally, Section 1.5 draws the outline of this thesis.

1.1 Context

In this section, the context in which our work was carried out is presented. First, some statistics and elements related to mobile video content are given, including how video data became vital in our daily life. Then, the evolution of video codec standards, in particular those of MPEG working group, are exhibited. After that, the importance of mobile platforms battery autonomy is highlighted by explaining the challenges to deal with. Next, the evolution of the heterogeneous mobile architectures is introduced. Finally, a short description of an Operating System (OS), such as Embedded Linux, is given.

1.1.1 Mobile video content

Web data are experiencing a proliferation of video content for mobile platforms. According to Cisco [2], by 2022, online videos will make up more than 82% of all consumed internet traffic. Smartphones, tablets, and media players are the favored and most frequently-used tools to consume these multimedia content. Around 75% of all video plays

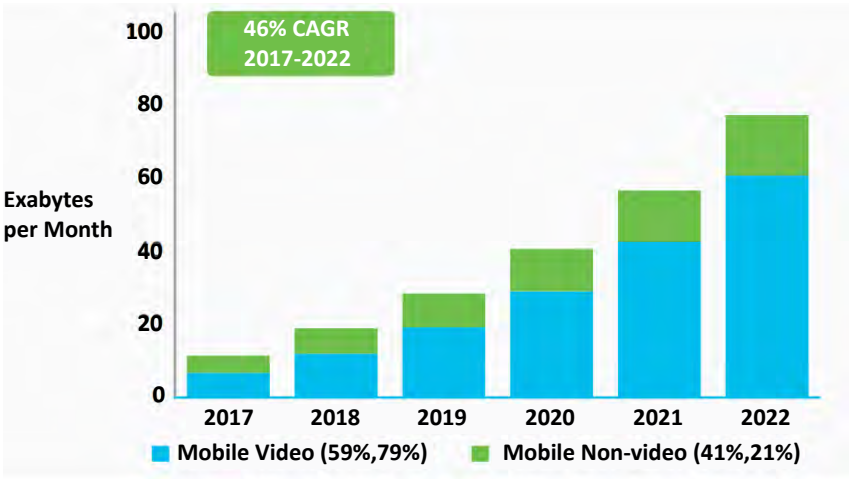


Figure 1.1 – Expectation for global mobile video growth from 2017 to 2022 from CISCO VNI, from [5]

are performed on mobile platforms [3] of which 80% are equipped with full high definition (1080p) or lower screen resolutions [4]. As depicted in Fig.1.1, the Visual Networking Index (VNI) from CISCO predicted that the mobile video data increase much faster than the non-video ones, accounting for 79% of total mobile data traffic by 2022 [5].

Mobile video content continues to be one of the most effective elements in the communication tools. Authors in [6] highlighted some factors behind this popularity: shared viewing experiences via social networks, an immersive solitary activity (e.g., on public transport), etc. In addition, viewers retain 95% of a message when they watch it in a video, compared to 10% when reading it in a text [7]. Furthermore, there is an increasing use of video-hosting platforms (e.g., AcFun, Vimeo, etc), mobile IPTV, etc.

Traditionally, a video eco-system consists of centralized servers, managed by broadcasters, and TV receivers allowing to display the produced video content. This has known a big progress nowadays. Indeed, the production of video content is much easier today than it used to be. First, video production is made possible by advanced video codec standards and processing units embedded in the mobile platforms where any one can produce video content in a very easy way. Then, video transmission is facilitated by advanced Internet services. Finally, video display system has greatly evolved in mobile platforms making viewing experience as close as that of traditional TV. The whole chain is facilitated thanks to mobile user-friendly applications.

Given that different mobile video consumers have different hardware capabilities in terms of bandwidth, processing units, etc., the requirements are different if the video con-

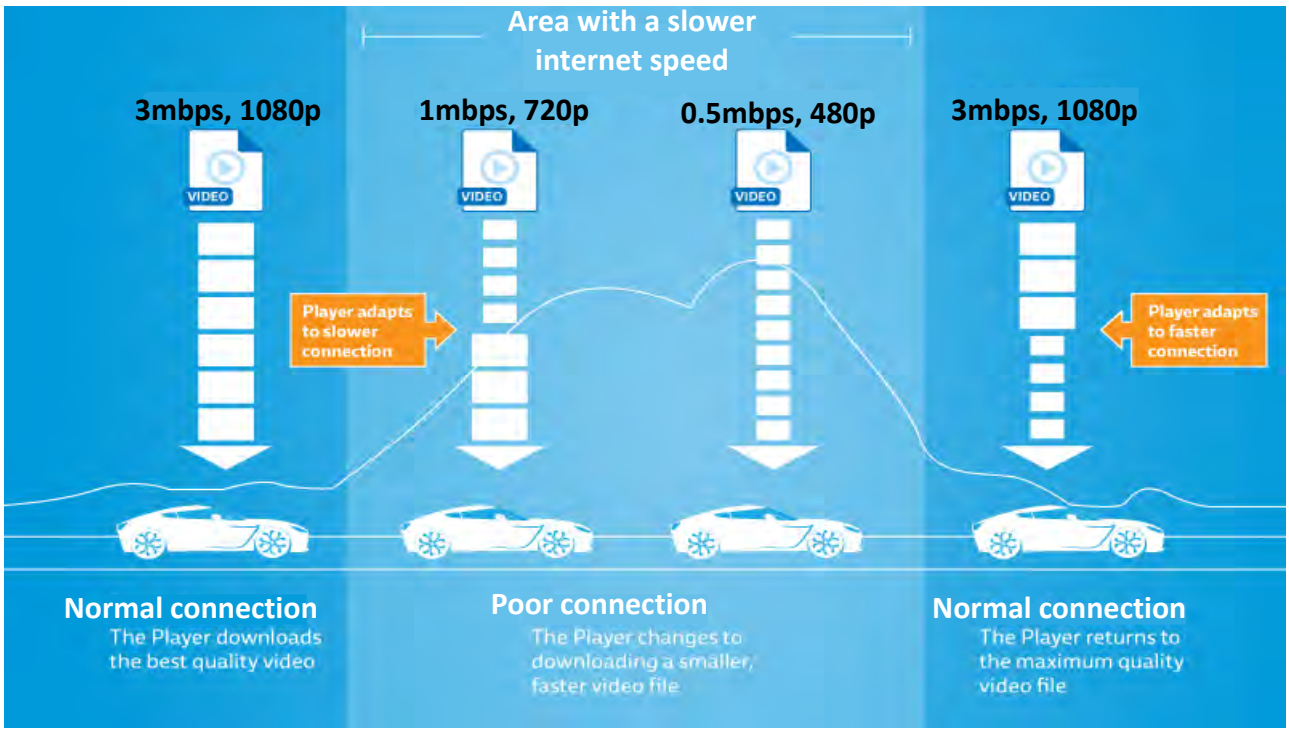


Figure 1.2 – Dynamic Adaptive Streaming over HTTP (DASH) example, from [9]

tent targets high definition or mid-range smartphones. In order to address different use cases and delivery scenarios, video content providers support Dynamic Adaptive Streaming over HTTP (DASH) [8]. DASH consists in adapting the video quality (bitrate or resolution) on the fly, in response to the heterogeneous capacity of the mobile platform in terms of remaining battery charge, internet bandwidth, or availability of processing resources. This technique (DASH) made end to the world where the video quality does not change from the producer to the consumer. Therefore, the end-user can watch the video at the most appropriate decoding rate. Fig. 1.2 depicts an example of applying DASH technique. The original video resolution (1080p), which requires an internet speed of at least 3 Mbps, is down-scaled to a lower resolution (480p), when the internet speed gets slower, i.e., less than 3Mbps. The decrease in video resolution allows a smooth playback of the original video but at a lower resolution.

In conclusion, mobile video content is in a continuous evolutionary state in terms of volume and ubiquitous existence. So, substantial efforts should be realized in order to adapt the mobile platforms to this evolution and facilitate the consumption of mobile video content to the end-users.

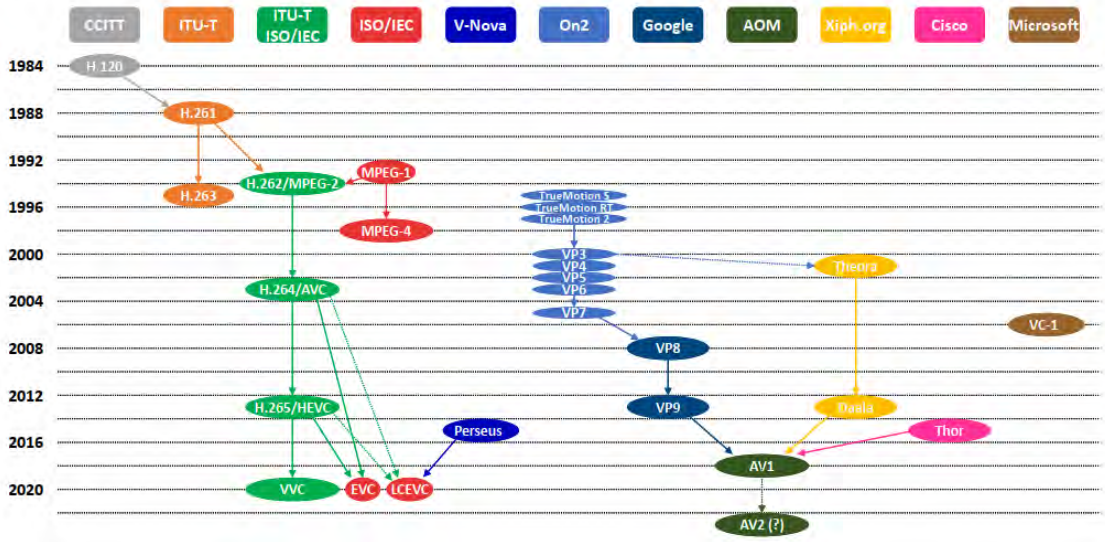


Figure 1.3 – Video codecs history

1.1.2 Evolution of video codec standards

In response to the growing need for video applications, such as Internet streaming, video-conferencing, digital storage media, and television broadcasting, Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) jointly developed several codecs during the last two decades, including: H.262/MPEG-2 [10], H.264/AVC [11][12], H.265/HEVC [13], and H.266/VVC [14]. The history of these codecs is presented in Fig.1.3, ITU-T ISO/IEC column. Each codec aimed at reducing the bitrate, and so the video file size, by achieving higher compression ratio while keeping the same visual quality.

For instance, the High Efficiency Video Coding (HEVC) codec standard, was released in 2013 as the successor to the predominant recommendation ITU-T Advanced Video Coding (AVC/H.264). The main purpose of HEVC is to decrease the bandwidth and storage requirements by roughly 50% with respect to its predecessor. HEVC presents efficient coding techniques that are capable of reducing the bitrate for ultra-high-definition resolution (2160p) videos.

In conclusion, the advantage of HEVC over AVC makes it present in almost all new mobile platforms, around 67.5% of all mobile platforms present in the market in 2018, including the old ones, according to [15]. Therefore, thanks to the predominance of HEVC, it was chosen as the main codec on which our study and experiments were conducted. It should be noted that VVC was not chosen because it was recently released in July 2020

[16], and to the best of our knowledge, there was no real-time implementation during this thesis.

1.1.3 Energy efficiency challenge

These new trends in video application usage combined with the market explosion of multimedia consumer electronics raised new challenges for mobile platform architecture designers. Indeed, to fit the important processing requirements of video applications, such as real-time constraint, processing resources embedded in these platforms tend to be more and more powerful and complex. One important issue resulting from these new tendencies in hardware (HW) architecture is an acute increase in power consumption.

In addition, the exploding usage of video content further accentuates the energy consumption issue. In fact, each new codec uses more advanced tools than its predecessor. These advanced tools, on the one hand, increase the coding efficiency and so decreases the bandwidth, and, on the other hand, gets compression algorithms more and more complex and greedy [17]. This results in a growing demand on processing resources and thus on the energy consumption at the decoder side. For instance, in [18], measurements show that the smartphone battery can entirely be consumed when decoding a 1080p video sequence in real-time. The video decoding process is run in general purpose processor (GPP) by applying the state-of-the-art HEVC codec for ∼4h. The measurements do not consider peripherals energy such as that of the display system. This leads to a forceful decrease in mobile platforms autonomy as the increased power demand could not be compensated by the improvements in battery technology [19].

One proposed solution, to reduce video decoding energy consumption while delivering high performance, is using the HW video decoding run by a HW decoder Intellectual Property (HDIP), such as the HEVC decoder [20] [21]. In a state-of-the-art work, dedicated processors outperform the general purpose processors (GPPs) by around 1000× in terms of energy efficiency [22]. As a consequence, most modern smartphones are equipped with an HDIP that consumes less energy with respect to the real-time video decoding constraint [23][24]. For instance, 67.5% of all mobile platforms present in the market, including the old ones, are equipped with the HEVC HDIP, according to [15]. However, HDIPs are not flexible and are costly to implement, which generate a long time-to-market for new video codecs [25].

In conclusion, energy efficiency has become one of the most important factors in modern microprocessor design, especially for video decoding applications. Its importance is

justified by the omnipresence of mobile platforms constrained by limited battery life-time, in our daily life.

1.1.4 Evolution of heterogeneous architectures

Mobile processors are gearing up for a drastic change last decade with the advent of 64-bit ARM based processors which are expected to provide up to 50% performance improvement over existing 32-bit ARM processors [26]. Moreover, their speed has been multiplied thanks to the advances in CMOS (Complementary Metal Oxide Semiconductor) technology process. Indeed, according to Moore law, the number of transistors per unit of area approximately doubles every eighteen months [27].

The increase in processor speed (via the number of transistors per unit of area and thus the clock frequency) has been limited by the power dissipation, a phenomenon known as the power wall [28]. Then, the semiconductor industry turned toward multi-core architectures in order to speed up the software execution time. Also, heterogeneous architectures have been developed to deal with the ever growing number of very heavy applications, such as video decoding.

New advances in multi-core heterogeneous GPPs embedded in mobile platforms offer a great opportunity to enhance both performance and energy efficiency of software (SW) video decoding. In fact, leveraging parallel processing among the available cores reduces the required operating clock frequency which decreases drastically the consumed dynamic power, i.e., when the clock frequency is decreased, the voltage supply is accordingly reduced which leads into decreasing the consumed power [29] [30]. In case of HEVC decoding, this is enabled by virtue of advanced parallelism schemes supported by this codec [31]. Furthermore, GPPs allow developing and rapidly deploying a broad range of applications, in particular new video codecs, such as VVC [14].

Dedicated processors offer the optimum trade-off between performance and energy efficiency. This is achieved thanks to the use of specialized processing units [32] which eliminate the power consumption related to instruction decoding and control logic characterizing GPPs [33].

Fig.1.4 illustrates a comparison among different processors types: GPP, GPU, DSP, and dedicated circuits (e.g., HDIP) in terms of energy efficiency (Million of OPerations per Second / mW). These processors will be presented later, in Section 2.1.4. The graph shows two important points. First, dedicated processors contain more cores compared to the general purpose one which makes them more powerful. Second, the more a processor

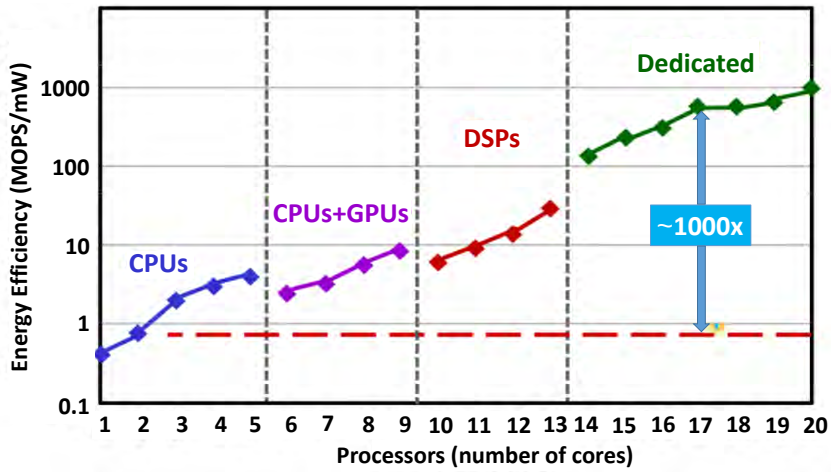


Figure 1.4 – Comparison of processors types in terms of energy efficiency, from [22]

is specialized, the more it is energy-efficient.

All the compared types of processors, in Fig.1.4, are able to perform video decoding, alone or with the help of GPP. Therefore, each processor may be integrated and used in a mobile platform to perform video decoding. The choice of a processor should take into account its advantages and drawbacks. For instance, dedicated processors are very energy-efficient. However, they are not flexible to support new video codecs and they take long time-to-market.

In conclusion, GPPs offer a great opportunity in terms of programmability. They are very flexible since developers can use them to implement any new solution. On the other hand, dedicated processors lack this feature, i.e., once deployed in a target platform, they are not re-configurable, but indeed, they are very energy-efficient.

1.1.5 Operating system (OS)

In general, an OS, such as Linux, is a piece of SW that manages all of the HW resources associated with the platform where the OS is hosted. It ensures, for instance, memory management, process scheduling, and Input/Output (I/O) support.

Actually, the mobile platforms, such as smartphones and tablets, tend to be more and more complex in terms of functionalities compared to a typical desktop system. An OS such as Android is not specific in scope and application as it manages a large scale of applications [35]. Accordingly, Android has been adapted to manage this increase in complexity and deal with the main characteristics of a mobile platform, e.g., limited

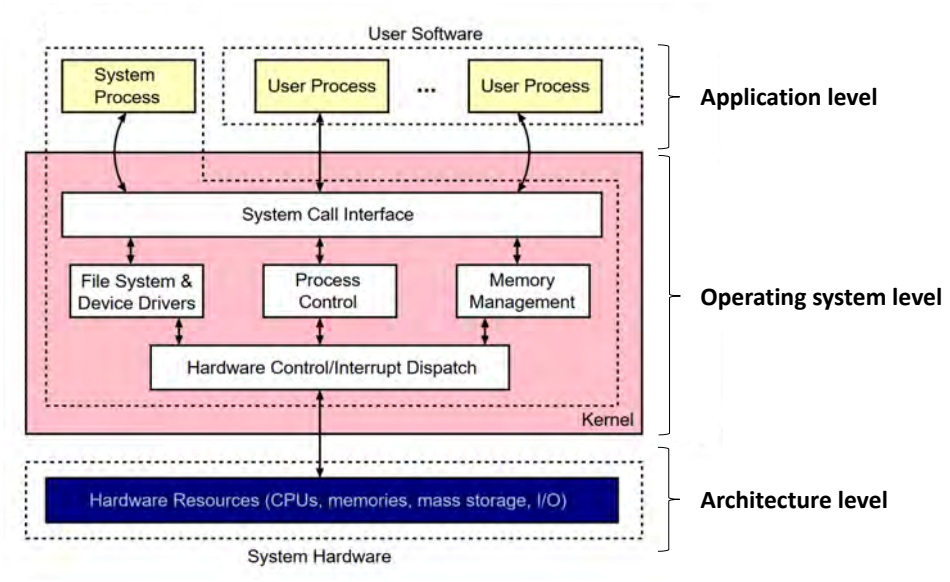


Figure 1.5 – System architecture overview, from [34]

battery life-time, limited resources performance, etc.

From the energy consumption point of view, the OS is both a consumer and a saver. First, an OS runs on a mobile platform as any other application. Therefore, it consumes power since it uses HW resources [36]. Then, the OS has a better knowledge of the environment on which the applications are running. It is positioned as a central component of a computer/mobile system, see Fig.1.5. It is responsible for controlling, integrating, and managing the individual HW components of a computer system.

The OS controls access to the HW resources. It then schedules applications to access those resources. It is aware of, for instance, which and how many resources are being in use by every single process present in the system. This knowledge makes it ideal for implementing energy saving strategies [37].

Generally, energy consumption can be considered as a purely HW issue. This means that, for example, to extend the battery life-time, one should develop new battery technology. However, according to [19], this research area has not grown fast enough compared to mobile applications. Indeed, the increased power demand caused by the software could not be compensated by the improvements in battery technology. Therefore, it is the responsibility of the OS to efficiently manage, from the one hand, the HW resources and, on the other hand, the SW applications that use those components in order to reduce the energy consumption.

In conclusion, given the essential work being done by an OS, it is absolutely relevant

to include it in the energy consumption considerations since the first stages of the system design.

1.2 Problem statement

We are experiencing a very challenging context. On the one side, mobile video content is in a continuous evolutionary state in terms of volume and ubiquitous existence. On the other side, battery autonomy does not grow fast enough to overcome the increase in mobile video applications.

To cope with this paradoxical situation, the HDIPs are a good solution that guarantees the trade-off between performance and energy consumption. However, they present a complex and costly design of integration in addition to the lack of flexibility. Therefore, one can exploit the advances in GPPs which offer two advantages: (i) programmability, and (ii) heterogeneity. Despite these advantages, the power consumption is still the main drawback of GPPs if the heterogeneity is not well designed.

Another challenging context is that the video content workload changes substantially from one frame to another, in a same video sequence. Optimizing the energy consumption of the video decoding is based on predicting the complexity of this workload in order to use the right resources. Predicting the complexity is based on having information about the frame to decode. However, these frame information are normally not known before to start the decoding process, except at the crude level of whether a frame is of type I/B/P and its size.

Therefore, in this context, the frequent Research Questions (RQ) we tried to answer in this thesis are formulated as follows:

1. RQ1: How relevant are the video decoders (HW and SW) with regards to the parameters impacting the video decoding process?
2. RQ2: To what extent do HW video decoders (HDIPs) outperform SW ones?
3. RQ3: How to balance the video content workload among the heterogeneous GPP cores, based on a little amount of information on the video to decode?
4. RQ4: Finally, once the GPP cores are selected, how to adjust their frequency in order to reduce the energy consumption?

Answering these questions depends mainly on the considered abstraction level of the targeted system. The next section delimits the scope of the thesis to answer the above

questions.

1.3 Focus and scope

In this section, the scope of the thesis is defined, i.e., the architecture and levels at which our contributions are carried out as well as the targeted video codecs.

Our work consists in understanding the energy consumption of video decoding on mobile platforms and proposing mechanisms and strategies for optimizing it. The first step is to characterize the HEVC video decoding. Then, it will be question of optimizing energy consumption.

Throughout this thesis, the above research questions are answered without taking into account the video decoding modules details, see Section 2.1.1. The video decoding modules are beyond the scope of this thesis.

1.3.1 Target architecture

In our experiments, the heterogeneous mobile architectures, e.g., ARM big.LITTLE architecture, are targeted, thanks to their wide integration into recent mobile platforms [38]. This latter offers the possibility to leverage processors of different performance and energy efficiency capabilities. On the one hand, it is a great feature where we can use the right processor at any moment in order to get the desired performance while consuming a little amount of energy [39]. On the other hand, in the context of video decoding application, it is a very challenging task to do such assignment as the frame complexity differs substantially from one frame to another.

1.3.2 Target levels

The study of the performance and energy consumption of video decoding can be carried out at three levels, corresponding to those shown in Fig.1.5:

— **Architecture level**: at the architecture level, the performance and energy consumption are studied on different processing resource configurations. As illustrated in Fig.1.4, the video decoding process can be performed in different architectures, each with a specific processing unit: GPP, DSP, GPP in addition to GPU, and HDIP.

— **Operating system level**: at this level, to study the energy consumption of video decoding, the impact of the heterogeneity of the processor (number of cores and their frequencies) is investigated. Also, the inter-processor communication (IPC), implemented by the OS for scheduling, between the processor and the other components required to perform video decoding is considered.
— **Application level**: here, the impact of the video quality (bitrate, frame rate, and resolution) on the energy consumption is studied.

Our study is performed at two levels: (ii) OS level, and (ii) application level. The architecture is not varied as we used the two widely used microprocessor architectures on mobile SoC : HDIP and GPP (ARM).

1.3.3 Target video codecs

Regarding the targeted video codec standard, the MPEG standards are chosen as they got to maturation and are widely investigated and deployed all over the world. Our work focused on the HEVC codec. Indeed, during the thesis, it was the most recent standard agreed upon and adopted by the international standardization community. Therefore, this choice ensures the timeliness of our work.

1.4 Thesis contributions

One of the effective ways to reduce power and thus energy is efficiently programming the software applications and scheduling them by the OS (parallelism, frequency scaling using DVFS, etc.) rather than changing the architectural specifications [40]. This thesis is composed of two major contributions: (i) HEVC decoding performance and energy consumption characterization, and (ii) HEVC decoding energy consumption optimization. The first contribution answers the research questions (RQ1 and RQ2), whereas the second one answers RQ3 and RQ4.

1.4.1 Contribution 1: HEVC HW vs SW decoding methodology

In the first phase of this thesis, the research questions (RQ1 and RQ2) defined in the problem statement are solved. The objective of this phase is to understand and characterize the performance and energy consumption of HEVC decoding, the newest stable

MPEG video codec during this thesis. The characterization is performed at two hierarchical levels: (i) operating system, and (ii) application. This phase consists in investigating, through measurement, the power/energy consumption of the HEVC decoding process implemented in HW and SW. We compared these measurements in order to understand the behavior of the two types of video decoding (HW and SW). HW video decoding essentially serves as a reference point.

At the operating system level, the performance and energy consumption are investigated for both video decoding types. First, the HW video decoding energy consumption is studied. In this step, the trade-off between performance and energy consumption is investigated. Then, the IPC between the GPP and HDIP is evaluated. Finally, the ratio between the GPP–HDIP energy consumption and that of the global mobile platform is analyzed.

Second, the SW video decoding energy consumption is studied. In this step, the trade-off between performance and energy consumption is investigated, by varying the number of GPP cores and their clock frequencies. Then, the impact of GPP heterogeneity architecture on the energy consumption is highlighted. It will be a question of finding the optimal configuration that can make it possible to carry out video decoding in real time and consume as little energy as possible. In this context, a particular attention will be paid to memory transfers during the video decoding process.

Then, at the application level, the comparison between the HW and SW video decoding energy consumption is carried out. The comparison is realized by varying parameters at application level: video bitrate, video frame rate, and video resolution. The objective of these comparisons is to find the suited video decoding type (HW or SW) for each video parameter, e.g., which video resolution is suited for HEVC SW decoding.

1.4.2 Contribution 2: HEVC SW decoding energy consumption optimization

In the second phase of this thesis, the research questions (RQ3 and RQ4) defined in the problem statement are solved. The objective of this phase is to reduce the HEVC SW decoding energy consumption. The proposed solution is divided into three phase: (i) modeling of frame complexity, (ii) classification of video frames for an adaptive assignment, and (iii) frequency scaling using feedback control to adjust the GPP frequency.

First, a logistical model is established. It allows to classify the frames to decode into

two categories: (i) most complex frames, and (ii) least complex frames.

Second, thanks to this technique, for each frame, the classifier decides to which GPP cores it should be submitted to. For instance, in an ARM big.LITTLE architecture, this allows to decode the most complex frames on the high performance GPP cluster and the least complex frames on the energy-efficient GPP cluster. Moreover, the classifier is based on some minimal knowledge before to start decoding (video bitrate, frame type, and frame size). Then, the frame partitions are decoded in parallel among the selected GPP cores. This is possible thanks to the parallelism schemes proposed by HEVC codec which are: tiling or wavefront parallel processing (WPP).

Third, the selected GPP cores frequency is adjusted using the feedback control technique proposed in [41]. This latter tries to keep the output buffer size at a certain value (an input parameter). Accordingly, the controller increses or decreases the frequency.

Finally, after a frame is decoded, the output buffer size is updated and, in response to that, this latter will be the input of the controller.

1.5 Manuscript outline

This thesis is organized as follows, and summarized in Fig.1.6:

Chapter 2 highlights the necessary knowledge to understand this thesis. In the first part, a short introduction to MPEG coding/decoding algorithms basics is given, followed by a focus on the HEVC decoding process (the newest stable codec during the thesis). Then, the strategies and techniques used to reduce video decoding energy consumption on mobile platforms, at the operating system level, are presented. In the second part, a summary of the literature review on the video decoding performance and characterization is given. Next, the most relevant strategies proposed to optimize the energy consumption of video decoding on mobile platforms, at the operating system level, are highlighted.

Chapter 3 describes our proposed methodology to resolve the problem statement (RQ1 and RQ2). First, the overall power model used to measure power and calculate the energy consumed when running video decoding is defined. Second, the characterization of the HEVC HW and SW decoding performance and energy consumption for an objective analysis and comparison is detailed.

Chapter 4 is dedicated to present the results of the performance and energy consumption characterization of HEVC decoding. According to our proposed methodology, the obtained results are presented in two sections, corresponding to two levels: (i) operating

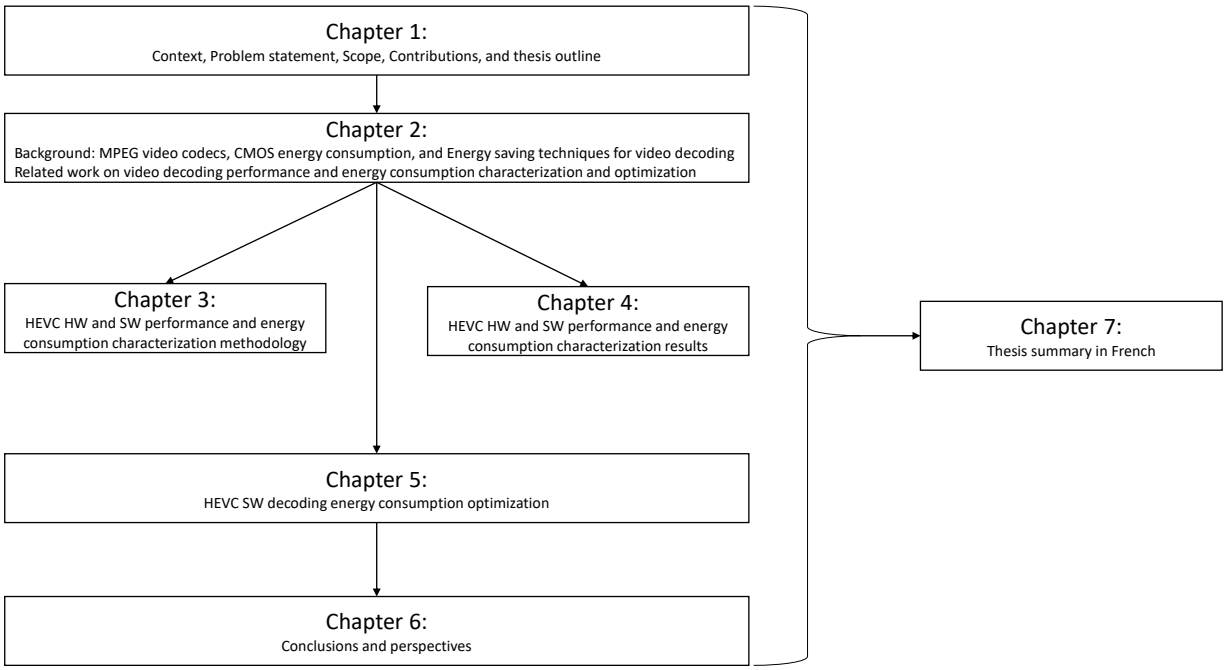


Figure 1.6 – Manuscript outline

system level, and (ii) application level.

Chapter 5 details our proposed strategy to optimize the energy consumption of HEVC decoding. First, the model established is explained. The model aims at predicting the complexity of a frame, before to start decoding, in order to submit it to high-performance or power-efficient cores in a heterogeneous mobile architecture. Second, the configuration of the feedback control strategy is explained. Actually, the feedback control is leveraged to adjust the GPP frequency, using DVFS. Finally, the results obtained by applying the proposed strategy are presented, compared to those obtained from the state of the art.

Chapter 6 draws the conclusions on this study. In the first part, the most relevant results obtained during this thesis concerning: (i) characterizing the HEVC decoding performance and energy consumption, and (ii) optimizing the energy consumption of HEVC decoding, on mobile platforms are summarized. In the second part, space is left for discussion on some perspectives related to video decoding energy consumption on mobile platforms.

Chapter 7 gives a substantial thesis summary in French.

Chapter 2

Background & Related Work

In this chapter, all the necessary background knowledge to understand this thesis is described. The chapter is divided into two main parts: (i) background, and (ii) related work. In the first part, a broad introduction to: (a) MPEG video codecs is given, and (b) the energy consumption of CMOS electronic circuits are given. In the second part, relevant studies found in the state of the art related to: (a) video decoding performance and energy consumption characterization, and (b) video decoding energy consumption optimization are highlighted.

2.1 Background

In this section, all the necessary background knowledge to understand this thesis is described. First, a short introduction to MPEG video codecs is given: the H.264/AVC codec is taken as an example. Then, the particularities of the H.265/HEVC codec are illustrated. Second, some basic and necessary knowledge on the energy consumption of CMOS electronic circuits is introduced. After that, some techniques and strategies used to save the video decoding energy consumption are presented.

2.1.1 MPEG video codecs

In this section, a broad introduction to MPEG video codec standards is given. First, the basic principles of MPEG video coding/decoding algorithms are explained. Then, a focus on the HEVC decoding, the newest stable video codec of MPEG during this thesis, is presented. After that, the main video coding evaluation metrics are described. Finally, some metrics to assess video decoding (playback) are exposed.

Who is MPEG ?

Moving Picture Experts Group (MPEG) is a working group of ISO/IEC, founded in 1988 [42]. It was created to undertake an effort and standardize efficient video compression systems within a single set of specifications. The MPEG activities cover more than video compression, since the compression of the associated audio and the issue of audio-visual synchronization cannot be worked independently of the video compression [43]. The group is composed of experts from the electronic component industry, information technology, and telecommunications. Their meetings are targeting more than 200 companies spanning all industry domains with a stake in digital audio, video, and multimedia [42].

MPEG video codecs are used everywhere in everyday life. They cover many applications from interactive systems on CD-ROM to delivery of video information over telecommunication networks. Their success came from the efficiency of the proposed algorithms, on the one side, to compress data, and, on the other side, to decompress them. Moreover, the efficient digital representation of video content has been the subject of considerable research studies over the past two decades.

MPEG video codecs are generic. That means they are independent of any particular application, without ignoring the requirements of the applications [43]. MPEG codecs have evolved over time passing from basic coding technologies to technologies supporting and complementing the audio and video compression formats such as synchronization, multiplexing, composition, graphics, metadata, and intellectual property management and protection, as depicted in Fig.1.3.

Concepts of MPEG video coding

In this section, an overview of the MPEG video coding/decoding algorithms is provided. The basic concepts and techniques developed within the MPEG video-compression standards are reviewed. Then, some details regarding the HEVC decoding algorithms are highlighted as HEVC is the targeted codec of this thesis.

Video coding aims at reducing the data size of the video content for efficient storage and easier transmission. For example, a typical bandwidth needed for transmitting a 1080p video content is 1.2 Gbits/s. Thanks to compression, and depending the coding parameters explained later in this section, 10 Mbits/s are sufficient to transport this content, as estimated by this website [44].

First of all, in MPEG video codecs, a video sequence can be thought of as a sequence

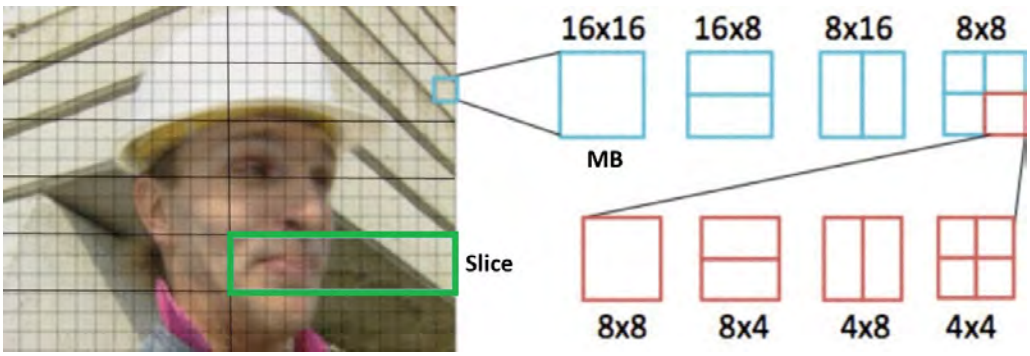


Figure 2.1 – MPEG frame structure, from [45]

of frames (images) to be coded individually, and then displayed sequentially at a given video frame rate. Typically, each frame may contain several slices and each slice may be segmented into several macro-blocks (MBs), as illustrated in Fig.2.1.

Then, the principle of video coding is based on the high statistical redundancy of video frames in both space and time domains. For instance, all MPEG video codecs (MPEG-2, AVC, HEVC, and VVC) make use of spatial (intra-frame) and temporal (inter-frame) redundancy to compress video data [17]:

1. Spatial redundancy reduction: is used to remove similarities among blocks of pixels belonging to the same frame, by predicting block of pixels from their neighbors. The redundancy reduction techniques usable to this effect are many, such as transform coding [46].
2. Temporal redundancy reduction: is used to remove similarities among blocks of pixels belonging to different frames (previous, next, etc.), by predicting block of pixels from other frames.

Therefore, thanks to these two reduction techniques, only reference frames in addition to the changes from a block of pixels to another one (within the same frame or in a different frame) are encoded. Furthermore, in a typical compression scheme, an adaptive combination of spatial and temporal redundancy reduction techniques is used to achieve significant data compression.

Also, these two reduction techniques are used to do prediction. The prediction aims at finding a correlation between the MB to be encoded (current MB) and a reference MB, i.e., searching the « best matched » MB. The reference MB may be in the same frame (intra-prediction) or in a past or future frame (inter-prediction).

Based on the above reduction techniques, there are three types of frames which are

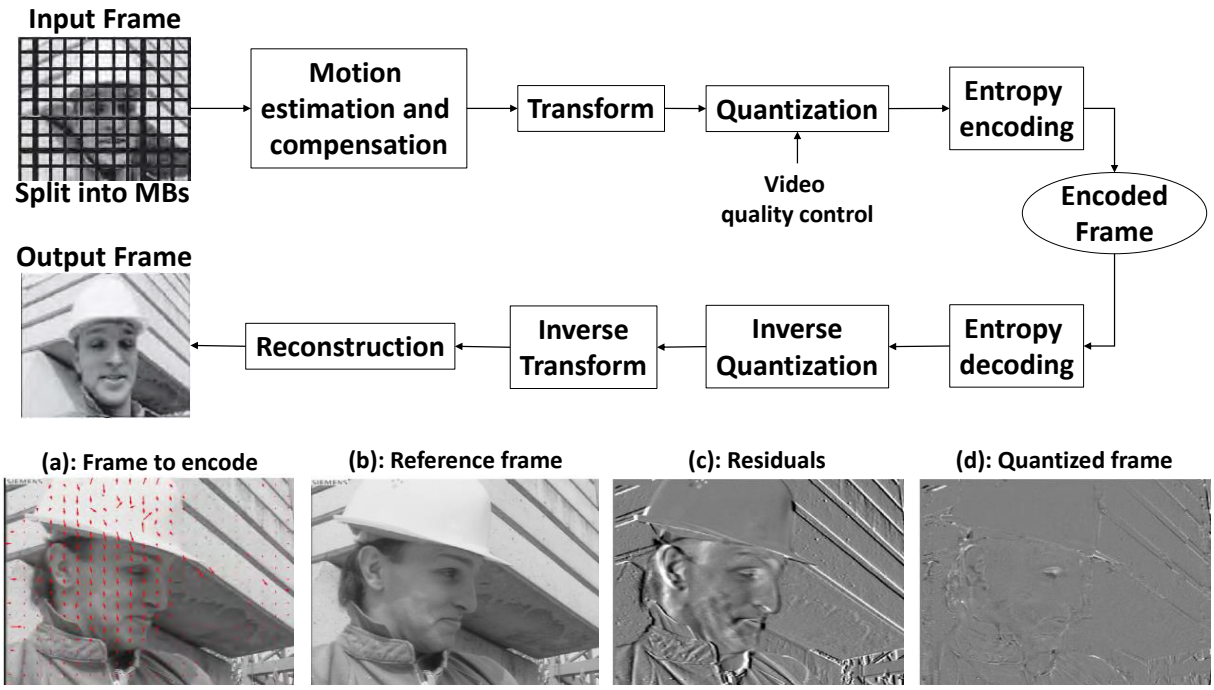


Figure 2.2 – MPEG video codecs general algorithm diagram

considered in MPEG video codecs [47]:

1. Intra frames (I): intra frames are coded without reference to other frames contained in the video sequence. They achieve a moderate compression because only intra-prediction is used for encoding,
2. Predicted frames (P): predicted frames are coded with reference to a past frame (Intra- or Predicted) and will, in general, be used as a reference for future predicted frames,
3. Interpolated frames (B-for bidirectional prediction): bidirectional frames are coded with reference to a past frame and to a future one. They provide the highest amount of compression. In addition, bidirectional frames are never used as reference [46], i.e., no frame (I, P, nor B) can be encoded based on a bidirectional one (B).

We show hereafter the common techniques used in the different generations of MPEG video codecs. For that, AVC is introduced here as a canonical example. The coding process is organized into several modules: (i) Motion estimation and compensation (MC), (ii) Transform, (iii) Quantization, and (iv) Entropy encoding, see Fig.2.2.

As illustrated in Fig.2.2, at the encoder side, before to start encoding a video, each frame is partitioned into MBs. Then, in the first step, for each MB, the motion estimation

and compensation (prediction) is applied. This results in finding the reference MB. The coordinates of the reference MB are stored in a motion vector (MV). The data obtained from subtracting the current MB from the reference MB are called residual MB. The MV and the residual MB allow reconstructing the current MB. Frame (a) in Fig.2.2 is an example of the current frame to encode. The red arrows in this frame represent the process of prediction, i.e., searching « best matched » MB. Frame (b) represents the reference image.

In the second step, the residual MB coefficients are transformed into another domain (frequency domain) in which they are represented by transform coefficients. The reason of this transformation is that the human eye is more sensitive to view in frequency domain than in the spatial one. After this transformation, the data should be decorrelated and separated into compact group of data with minimal interdependence where most of the information should be concentrated into a small number of values. The transform operation can be achieved, for example, using discrete cosine transform (DCT) [48], see Frame (c) in Fig.2.2.

In the third step, the coefficients obtained from the transform operation are quantized, i.e., divided by an integer number called: quantization parameter (QP), a.k.a., coding rate. The quantization allows to reduce the number of coefficients different from zero (non-zero coefficients) because these coefficients provide a more compact representation of the residual data. For example, in Fig.2.2, the original coefficient values representing Frame (c) are divided by a QP and rounded to the nearest integer. Typically, the result of the quantization is a block in which most or all of the coefficients are zero (see Frame (d)), with a few non-zero coefficients. Setting QP to a high value means that more coefficients are set to zero, resulting in a high compression at the expense of a poor decoded frame quality. Setting QP to a low value means that more non-zero coefficients persist after quantization, resulting in a better visual quality at the decoder side but also in a lower compression (higher bitrate). It should be noted that the quantization operation is not reversible, i.e., data obtained after this operation cannot be completely retrieved after decoding.

Finally, all the resulting data from these steps (transform coefficients, MV, etc.) are compressed using an entropy encoding algorithm, such as context-adaptive binary arithmetic coding (CABAC) [49]. This operation is reversible, i.e., data will be completely retrieved after decoding. At this stage, the compressed data are ready to be sent to the decoder.

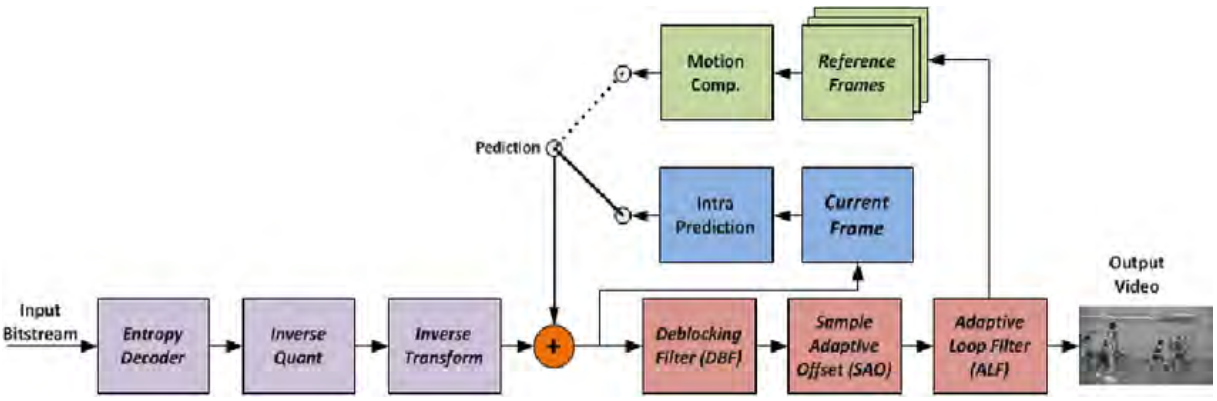


Figure 2.3 – HEVC decoding diagram, from [50]

At the video decoder side, the decoder reverses the encoding process and decompresses the video content to be ready for display. First, the entropy decoding is executed to extract the MV, the residual data, etc. The inverse quantization is then executed to obtain the transform coefficients, which are input to the inverse transform. Notice that, only the coefficients concentrating the most relevant information are recovered. The information associated to the null coefficients are thus lost. Consequently, the recovered coefficients resulting from the inverse transform are not identical to the original video data in Fig.2.2). Finally, the frame is reconstructed based on the decoded data.

HEVC decoding process

HEVC codec was standardized in 2013. It is organized into several modules, as a canonical MPEG video codec described in the previous section. Fig.2.3 shows a diagram of HEVC decoding. First of all, the Entropy Decoding module extracts the data payload and the syntax elements from the video bit-stream using an arithmetic coding algorithm such as CABAC. Then, the residual data are dequantized and inverse transformed using Inverse Discrete Cosine Transform (IDCT). This is the Dequantization and IDCT module. After that, the Prediction module is applied. Finally, to reduce the artifact caused by the decoding process and improve the visual quality of the reconstructed frame, the Deblocking Filter (DF) & Sample Adaptive Offset (SAO) module is applied, a.k.a., In-loop filtering.

In addition to the ability to decode frame by frame, HEVC includes two concepts that enable some degree of high-level parallelism schemes: (1) tiling, and (2) wavefront parallel processing (WPP). It should be noted that, in HEVC, a given coded video sequence

cannot include both tiling and WPP parallelism schemes [13].

1. Tiling: is used to split a frame horizontally and vertically into multiple independent, rectangular regions (tiles), see Fig.2.4. This division has the advantage of increasing the parallel friendliness of HEVC since different regions can be decoded simultaneously [51].

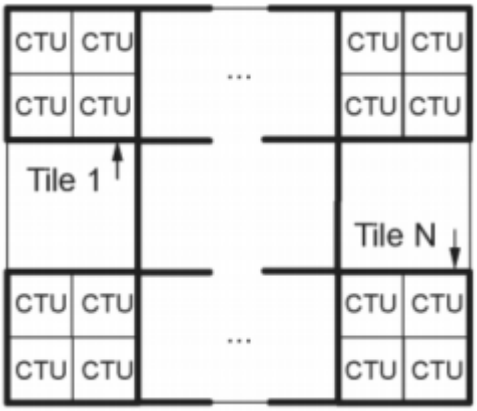


Figure 2.4 – Tiling scheme, from [52]

2. Wavefront parallel processing (WPP): this type of parallelism splits a frame into multiple rows. Each row contains multiple CTUs [1]. Each row can be processed in a different thread, in a pipeline fashion. This means that the decoding of a CTU of a given row must proceed with a delay of two CTUs with respect to those of the previous row. For instance, the first row is processed in an ordinary way. Then, the second row can begin to be processed after decoding two CTUs of the first row. Next, the third row can begin to be processed after decoding two CTUs of the second row, and so on [52]. Therefore, CTUs with same indices are decoded in parallel, see Fig. 2.5.

HEVC addresses a wide range of applications. To support the application variations without altering the core specifications, HEVC supports three profiles similarly to prior standards: (i) Main profile, (ii) Main Still Picture profile, and (iii) Main 10 profile [2]. These profiles have been defined in the first version of HEVC [54]. Then, in the Range Extensions (REext) version of HEVC, other profiles have been added, called REext profiles [54].

1. CTU (Coding Tree Unit): is the basic processing unit of the standard to specify the decoding process (conceptually corresponding to an MB in prior standards) [17].

2. A profile is a set of features and capabilities that the encoder and decoder agree on so that both can handle [53].

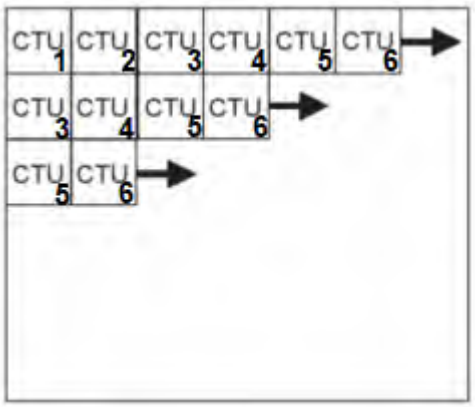


Figure 2.5 – Wavefront parallel processing (WPP) scheme, from [52]

In this thesis, all the experiments were conducted on the Main profile video sequences. The reason is that this profile is widely used in end-user systems, such as TV broadcasting or video on mobile platforms [53].

Visual video quality assessment metrics

The video compression methods are often lossy, which allow to reduce the video data size. This can induce a degradation in the video visual quality. Generally, the higher the compression ratio is, the lower the video quality we get. Therefore, one should tune the coding parameters in the manner to obtain an acceptable trade-off between the desired video visual quality and the video bitrate.

Therefore, in 1997 the Video Quality Experts Group (VQEG) is formed from experts of ITU-T and ITU-R. Their purpose is to evaluate perceptual quality models suitable for digital video quality measurement [55].

In general, there are two types of visual quality assessment:

1. Subjective: relies on the human observations. These metrics are the most reliable ones as the human eyes are in most cases the ultimate receivers of video content [56]. However, subjective metrics are costly and slow to set up [56].
2. Objective: this kind of metrics is introduced in order to replace the subjective ones due to their limitations. The objective metrics tend to assess the video visual quality in the same way the human does [56]. Below, some objective metrics used to evaluate the video visual quality are presented.

Bitrate: The bitrate refers to the number of bits that are processed or conveyed in

a unit of time, e.g., in one second. It may give an idea about the video quality as it is a consequence of the QP used to encode the video. Actually, for a given video sequence, the video bitrate increases every time we decrease the [57]. This is particularly true is case of constant bitrate (CBR). For instance, a video encoded at 2048 Kbps has a higher visual quality than the same video encoded at 1024 Kbps. However, the relationship between the bitrate and the visual quality is not linear [58].

PSNR: The Peak Signal to Noise Ratio (PSNR) evaluates the video quality more accurately, by analyzing the ratio between the error and the pixel values [59]. It is used to evaluate the difference (error) between a degraded (encoded) video and its reference version. PSNR is a widely recognized objective metric used in image and video processing domains [60], for instance, to compare video codecs [17].

Visual information fidelity (VIF): This metric quantifies the information available in the reference video and determines how much of these reference information can be extracted from the distorted video [61]. This model captures important and complementary distortion types: blur, additive noise, and global or local contrast changes.

Video structural similarity index measure VMSSIM: It is a quality evaluation metric for image, and then adapted for video. It is based on an assumption that natural video frames are highly structured, and that the human visual system (HVS) is highly adapted for extraction of structural information from the viewing field, such as blurring of edges or visibility of blocks. This metric compares not the pixel values but the frame elements perceived by the human: luminance, contrast, and texture [62].

Video multi-method assessment method VMAF: VMAF is typically used to assess video streaming services. It predicts subjective video quality based on a reference video sequence (video before encoding) and distorted video sequence (the same video sequence after encoding). It combines human vision modeling with machine learning, offering a good prediction of the video quality of experience (QoE) [63]. VMAF fused several existing features, such as VIF, using a supervised learning regression model to provide a single output.

Video playback evaluation metrics

In order to measure the ability of a given mobile platform to decode (playback) a video sequence, some quality of service (QoS) metrics are proposed. These metrics impact the visual perception of the video content [64]. Some of these metrics are presented hereafter.

Frames per second (FPS): FPS is defined as the average number of frames decoded

during one second. It expresses the speed at which the video is played back. It is a common specification used both in video capture (input FPS) and playback (output FPS). To ensure a smooth playback, the output FPS should be greater or equal to the input one.

Deadline miss rate (DMR): DMR is the number of frames which are not decoded before their deadline. The deadline corresponds to the display system rate. Although video decoding is considered as a soft real-time application, i.e., it can tolerate and allow some certain deadline miss [65], it is highly important to quantify them in order to justify whether the considered system is acceptable. The lower the DMR is, the smoother the visual quality we get. To avoid deadline miss, one condition, among others, is that the FPS should be greater or equal to the display system rate. The DMR may be caused, for instance, by the insufficient resources available to perform decoding.

2.1.2 Energy consumption in CMOS electronic circuits

In this section, some basic and necessary knowledge on the energy consumption of video decoding is introduced. First, the power consumption of a CMOS electronic circuit is defined. Then, the sources of power dissipation are presented. Next, different strategies of video decoding energy saving, at operating system and architecture levels, are explained. Finally, the trade-off between performance and energy consumption (referred to as performance-energy) of video decoding is discussed.

With advances in electronic circuits (e.g., high clock frequencies) and the increase in mobile applications and services requirements, pressure on energy consumption is also increasing. Therefore, it is more important for manufacturers to find effective means for increasing battery life of mobile phones, as the mobile device itself is energy consuming and longer operational times are highly requested by end-users [66]. It is important to understand the energy consumption at CMOS electronic circuit level in order to understand it and optimize it at the global mobile platform level.

The energy consumption (in Joule) of a CMOS electronic circuit is the amount of power P consumed during a time t.

$$E = P * t \tag{2.1}$$

where P is the power consumption (in Watt) of the circuit.

Three components determine the power consumption in a CMOS circuit, i.e., the total

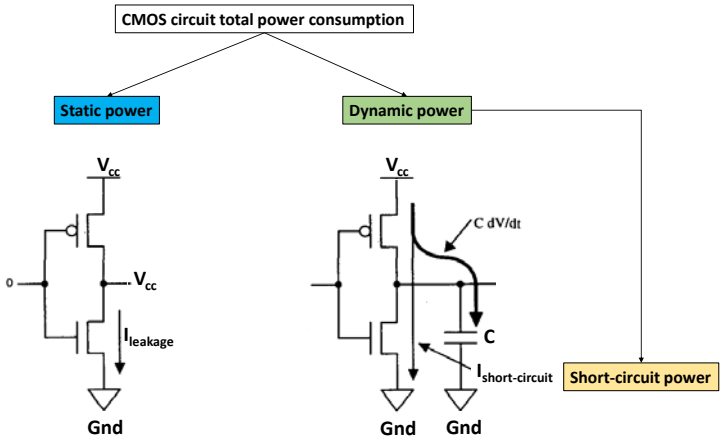


Figure 2.6 – CMOS circuit power consumption split into static and dynamic powers, from [67]

power consumption is the sum of the static power (P_{static}), dynamic power (P_{dyn}), and short power (P_{short}), see Fig.2.6. This can be translated by the following equation:

$$P = P_{static} + P_{dyn} + P_{short} \tag{2.2}$$

First, the static power is the power consumed in the state where there is no activity at the CMOS electronic circuit level, i.e., the transistors constituting the GPP are not switching between 0 and 1 (charging and discharging) [68] and vice versa. It is also dissipated by leakage currents that flow even when the device is inactive. In general, CMOS devices have very low static power consumption. Furthermore, the static power is consumed independently of any program being executed in the system. Below 65-nm circuits feature size, the static power consumption becomes significant and poses new low-power design challenges [69].

The static power can be formulated by the following expression [70], Equation (2.3).

$$P_{static} = V_{dd} * (I_{sub} + I_{gate} + I_{junct} + I_{contention}) \tag{2.3}$$

where $I_{sub}, I_{gate}, I_{junct}$, and $I_{contention}$ are the sub-threshold current, gate leakage, junction leakage, and contention current, respectively [70].

Second, the dynamic power is the power consumed when there is an activity, e.g., when

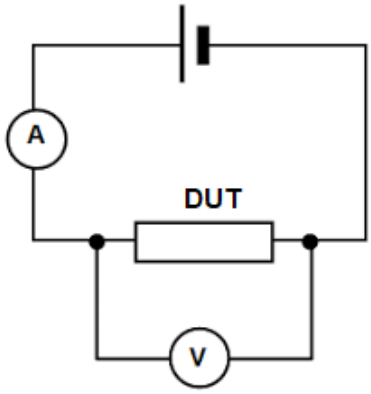


Figure 2.7 – CMOS circuit power measurement

the GPP is running a program. In idle state, i.e., in the state where there is no program running at the user level, the dynamic power may be zero if the clocks are gated, i.e., if the global clocks are turned off.

The dynamic power can be formulated by the following expression [71], Equation (2.4).

$$P_{\text{dyn}} = C_{\text{eff}} * V^2 * f \tag{2.4}$$

where C_{eff} represents the circuit effective capacitance and V the supply voltage associated to the clock frequency f [29]. In a microprocessor, the C_{eff} parameter represents the average capacitance of all the processor blocks (control unit, cache, etc.) which depends on the type of instructions executed and on the data accessed [71].

Practically, the power consumption is calculated using the following formula: $P = V*I$, where V is the difference in charge between two points of the device under test (DUT), in volts, and I is the current circulating via that device, in amperes. To measure the power consumption of a given DUT, one can use an ampere-meter in series with that circuit to measure the current intensity and a voltmeter in parallel to measure the voltage, see Fig.2.7.

Finally, P_{short} component of Equation(2.2) captures the power resulting from a short-circuit current. It momentarily flows between the supply voltage and the ground due to a short-circuit current appearing when a CMOS logic gate output switches. It is considered as a small part of the dynamic power as it is related to the circuit activity [67][72]. Therefore, it is neglected in the rest of this manuscript.

Fig.2.8 plots the relationship between the GPP total power consumption (static and

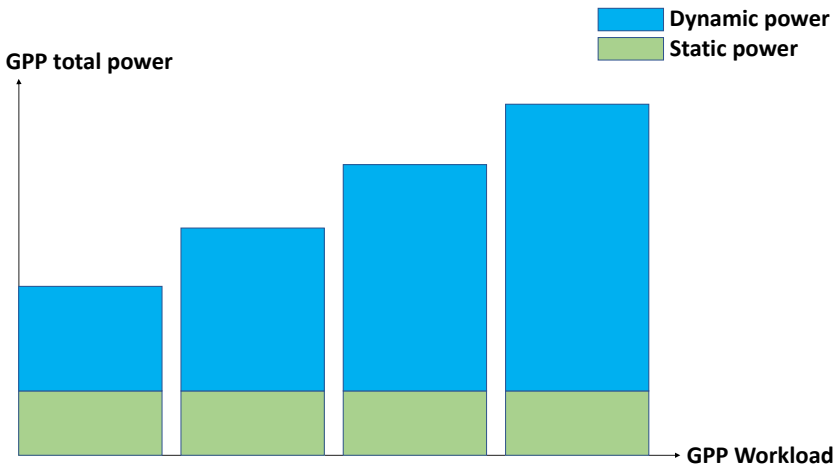


Figure 2.8 – Power consumption as a function of GPP workload

dynamic) and its workload. As explained before, the static power does not vary when the GPP workload varies because this power is dissipated at the CMOS circuit level regardless of any GPP activity. In contrast, the GPP dynamic power heavily depends on its workload because it is related to the switching state of the transistors. This switching is caused by the GPP when it is running a program.

In the rest of the manuscript, we consider that the total power consumption of a CMOS circuit is given by the following equation:

$$P = P_{static} + P_{dyn} \tag{2.5}$$

2.1.3 Energy saving techniques

Based on the definitions given in the previous section, a particular attention should be paid to how the power is managed. In this section, the most common techniques to reduce energy consumption (static and dynamic) are presented: (i) dynamic voltage and frequency scaling (DVFS), and (ii) dynamic power management (DPM).

Dynamic voltage and frequency scaling (DVFS)

DVFS technique provides ways to reduce power consumption, and thus energy consumption, of chips in run-time by scaling down the voltage (and frequency) based on the

targeted performance requirements of the application. In DVFS, the voltage levels of the targeted power domains are scaled in fixed discrete voltage steps, i.e., to each frequency value corresponds a voltage level. A voltage-frequency pair represents a power state and is called P-state.

According to Equations (2.1) and (2.4), the dynamic energy is:

$$E_{dyn} = P_{dyn} * t = C_{\text{eff}} * V^2 * f * t$$

Then, for the sake of simplicity, the processor frequency is considered proportional to the voltage (i.e., $V \propto f$) as assumed in [73][74]. The dynamic energy is then:

$$E_{dyn} = K * f^3 * t \tag{2.6}$$

where K is a constant parameter, f is the processor frequency, and t is the execution time. Using this equation, if, for instance, the frequency is divided by 2, the execution time, t, may be doubled but the energy, E_{dyn}, is divided by 4, which explains the energy reduction. Therefore, it is up to the SW designer to select the most relevant frequency which satisfies both performance and energy consumption.

Dynamic power management (DPM)

The DPM technique is based on power states called C-states [75]. C-states are states where the GPP is able to put under-utilized or idle resources into states of operation with reduced or null performance levels that require little or no power.

The advantage of DPM policies is that they allow to reduce both static and dynamic power dissipation. However, the inconvenience is that once in a power-efficient state, bringing a component back to active/running state requires additional energy and/or latency to service an incoming task. This energy overhead is not negligible in case of mobile platforms [76]. For this reason, in this thesis, we focused only on DVFS technique to save energy and satisfy the real-time constraint of video decoding.

In conclusion, for both techniques presented above, according to [77], the power management can be treated as a prediction problem. It seeks to forecast the idle period or the period where the GPP needs less resources in order to save power. The shortest idle period to save power is called the break-even time. It varies from a device to another and is independent of any running program [77].

Table 2.1
Video frame complexities in mega-cycles (MC)

F_i	$C_i(MC)$
F_1	50
F_2	32
F_3	13
F_4	25
F_5	30

2.1.4 Principles of energy saving in video decoding

This section describes some strategies that allow to reduce the energy consumption of video decoding at both architecture and operating system levels. We recall that to reduce video decoding energy consumption, one should try to find the best balance between the video visual quality (QoS metrics) and the energy consumption. Three strategies are discussed in this section: (i) frequency scaling, (ii) parallel multi-core video decoding, and (iii) specialized processing using dedicated processors.

To explain how strategies (i) and (ii) can be used to save energy, an example is introduced. In this example, a video decoding application executed by a GPP supporting DVFS mechanism is considered. The GPP frequency values belong to the interval $[f_{min}, f_{max}]$, and are assumed to be discrete values. The frequencies are expressed as $\alpha * f_{max}$, where f_{max} is the maximum clock frequency supported by the processor. In this example $f_{max} = 2GHz$ [3]. Then, α is a scaling factor. It depends, on the one hand, on the frame complexity, and, on the other hand, on the target architecture. It belongs to the interval $]0, 1]$. Actually, α takes only values corresponding to the available GPP frequencies of the target architecture. For instance, if the target architecture embeds a GPP that supports twenty frequencies, α takes twenty values in the interval $]0, 1]$.

For the sake of simplicity, a video sequence composed of five independent video frames is considered, $F_i, 1 \leqslant i \leqslant 5$. The video frame rate is 25 Hz which corresponds to the period of displaying rate. So, each frame should be decoded before the deadline $D_i, D_i = i * D$, where $D = \frac{1}{25} = 40ms$. In addition, every frame has its proper complexity, expressed in number of processor mega-cycles (MC) to be decoded, and presented in Table 2.1. The total number of cycles required to decode all the frames is denoted $C_{total} = \sum_{i=1}^{i=5} C_i$.

The time to execute C processor cycles at $\alpha * f_{max}$ frequency is assumed to be as

3. $f_{max} = 2GHz$ corresponds to the maximum clock frequency supported by the processor integrated in Exynos 5422 SoC, used in our experiments.

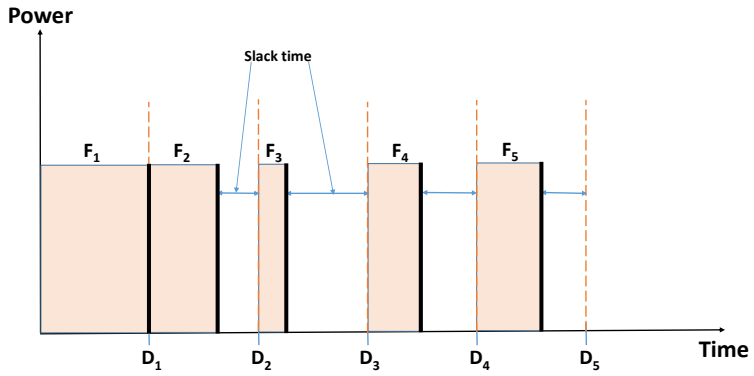


Figure 2.9 – Frequency scaling in video decoding

follows :

$$T_{\alpha}(C) = \frac{C}{\alpha * f_{max}} \tag{2.7}$$

Then, according to Equations (2.7) and (2.6), the energy consumed while decoding during C cycles and running at the frequency $\alpha * f_{max}$ is given by the following equation:

$$E_{dyn_{\alpha}}(C) = K * (\alpha * f_{max})^2 * C \tag{2.8}$$

Equation (2.8) will be used to explain how to save energy consumption of video decoding thanks to DVFS.

1- Frequency scaling: Performance vs energy consumption

According to the frame decoding complexities given in Table 2.1 and Equation (2.7), the clock frequency allowing all the frames to be decoded before their deadline is $f = \frac{5}{8} * f_{max}$ (i.e., $\alpha = \frac{5}{8}$). This frequency corresponds to the one which allows to decode the most complex frame (C = 50, see Table 2.1) before its deadline. According to Equation (2.8), the energy consumption in this case is:

$$E = K * f_{max}^2 * (\tfrac{5}{8})^2 (C_1 + C_2 + C_3 + C_4 + C_5) = \tfrac{1875}{32} * K * f_{max}^2$$

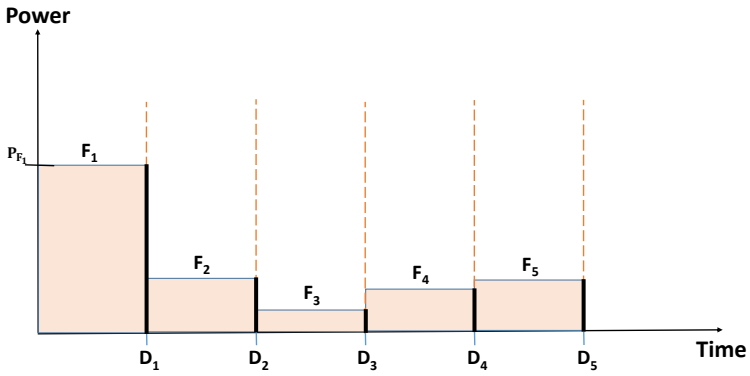


Figure 2.10 – Frame-by-frame-based DVFS

As illustrated in Fig.2.9, running constantly at this clock frequency leads to slack times because some frames are decoded early (F_2, F_3, F_4, and F_5)[4]. These slack times may be exploited to save energy if the frequency is adjusted using DVFS. Next, two DVFS strategies to achieve this objective are explained.

a- Frame-by-frame-based DVFS: Since video frames have different complexities, one can adjust the GPP frequency based on the complexity of every single frame. In the above example, according to Equation (2.7), if the frames F_1, F_2, F_3, F_4, and F_5 are decoded using the frequencies $\frac{50}{80} * f_{max}$, $\frac{32}{80} * f_{max}$, $\frac{13}{80} * f_{max}$, $\frac{25}{80} * f_{max}$, and $\frac{30}{80} * f_{max}$, they will be ready to be displayed just before their deadline as illustrated in Fig.2.10. Adjusting the frequencies allows to decrease the total energy to :

$$E = K * f_{max}^2 * ((\frac{50}{80})^2 * C_1 + (\frac{32}{80})^2 * C_2 + (\frac{13}{80})^2 * C_3 + (\frac{25}{80})^2 * C_4 + (\frac{30}{80})^2 * C_5) = \frac{20259}{640} * K * f_{max}^2$$

This represents approximately 46% of energy saving as compared to running at a constant frequency ($f = \frac{5}{8} * f_{max}$, which consumes $E = \frac{1875}{32} * K * f_{max}^2$).

b- Average workload-based DVFS: The averaging DVFS energy saving is based on the convexity of the E(f) model (see E(f) graph in Fig.2.11) and Jensen's inequality [1]. In fact, applying this inequality on a convex dynamic energy model results in :

$$E_{dyn}(\overline{f}) \leqslant \overline{E_{dyn}(f)} \tag{2.9}$$

4. The energy consumed during the slack times are assumed to be zero.

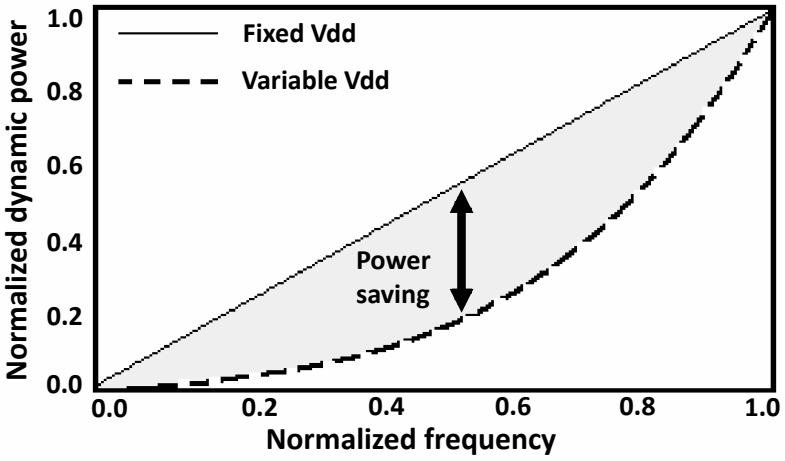


Figure 2.11 – Energy model convexity, from [78]

The above inequality shows that when the GPP works at a constant mean clock frequency it consumes less energy than when it works at different discrete clock frequencies [1].

To apply this approach in the above example, the frequency is set to the constant average frequency: $f_{avg} = \frac{1}{5}(\frac{50}{80} * f_{max} + \frac{32}{80} * f_{max} + \frac{13}{80} * f_{max} + \frac{25}{80} * f_{max} + \frac{30}{80} * f_{max}) = \frac{3}{8} * f_{max}$.

The total energy consumption, at the average frequency, is:

$$E_{avg} = K * (\tfrac{3}{8})^2 f_{max}^2 (C_1 + C_2 + C_3 + C_4 + C_5) = \tfrac{675}{32} * K * f_{max}^2$$

This represents approximately 64% of energy saving as compared to running at a constant frequency ($f = \frac{5}{8} * f_{max}$, which consumes $E = \frac{1875}{32} * K * f_{max}^2$). However, as illustrated in Fig.2.12, although the five frames are decoded within $5 * D$ time, the deadline of the frame F_1 is missed. This is explained by the fact that the frequency is set based on the average performance, not at a frame-by-frame basis. Thus, the most complex frames (e.g., frame F_1 in this example) cannot be decoded within their deadline. The deadline miss can be avoided if a buffer is inserted between the decoder and the displaying device but at a cost of an additional latency [79].

Discussion

The above presented strategies to save energy, (a) frame-by-frame-based DVFS and (b) average workload-based DVFS, require the following assumption: the decoder needs to have a prior knowledge of the video frames complexity. However, this is not always a valid assumption in real world scenarios. Indeed, the video workload (frame complexity)

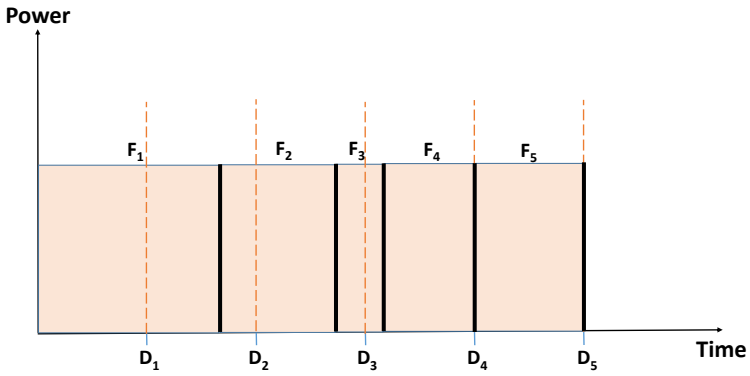


Figure 2.12 – Average workload-based DVFS

varies substantially from a frame to another within a same video sequence. Therefore, one should make prediction of the upcoming workload which is one of the most important steps in video-aware DVFS [80].

Another challenge to predict the video workload is the heterogeneity of components performance present in the SoC. Given that the video workload can be predicted by an accurate performance model, the video decoder can calculate the appropriate GPP clock frequency to decode the next frame before its deadline. The issue is the existing gap between the GPP speed and that of the off-chip memory access. In fact, video decoding is a memory-bound application which means that it makes use of a lot of instructions accessing to the external memory. Moreover, in GPP architectures, the bus used to access off-chip memory is clocked at a frequency which is independent of that of the GPP. So, the impact of scaling the GPP frequency on the performance may depend considerably on the off-chip memory access speed [81].

2- Parallel multi-core video decoding

Advances in multi-core SoCs architecture made GPPs more powerful even at lower frequencies compared to old GPPs. This gives the advantage of saving energy before applying any special mechanism at the operating system level. In order not to decrease the performance when scaling down the frequency, one may replicate HW, e.g., make use of two GPP cores clocked at $\frac{f}{2}$. This type of parallelism has shown good performance and energy efficiency for multimedia applications [82].

The parallelism of video decoding on multi-core processors can be achieved at the frame, slice, or MB level [83].

First, at the frame level, multiple frames may be decoded in parallel on different cores (one frame per core). This type of parallelism can work very well for intra-only mode videos where every single frame is encoded individually independently of the others. In contrast, for random-access mode videos, the number of independent frames is limited, which constitutes a real drawback of this type of parallelism.

Second, at the slice level, the parallelism among cores is more beneficial than the previous parallelism scheme. Indeed, slices inside a frame are independent of each other. So, submitting them to different GPP cores offers a higher scalability and a better opportunity to reduce energy without losing performance. One condition of this parallelism scheme is that it should be enabled at the encoder side, i.e., before to start decoding.

Third, at the MB level, the smallest unit of decoding can be decoded in parallel among different GPP cores since the MBs are independent of each other. The main drawback of this scheme is the communication overhead due to the high number of MBs inside a frame, especially for high resolutions. This parallelism scheme can be implemented more efficiently on specialized processors.

To illustrate this principle, we apply it to the previous example at the slice-level, as depicted in Fig.2.13. We suppose that each frame can be decomposed into two independent parts (slices) which are decoded in parallel using two identical GPP cores. This allows to achieve the same performance at $\frac{5}{16}f_{max}$, which is half the required frequency when using one GPP core.

The energy consumption in this case is:

$$E_{arch} = 2 * K * f_{max}^2 * (\tfrac{5}{16})^2(\tfrac{C_1}{2} + \tfrac{C_2}{2} + \tfrac{C_3}{2} + \tfrac{C_4}{2} + \tfrac{C_5}{2}) = \tfrac{1875}{128} * K * f_{max}^2 = \tfrac{1}{4}E$$

This represents approximately 75% of energy saving as compared to running at a constant frequency ($f = \frac{5}{8} * f_{max}$, which consumes $E = \frac{1875}{32} * K * f_{max}^2$).

Discussion

Thanks to the replication of HW, energy can be saved without a loss in performance, but at the expense of an additional circuit area. This increase implies an additional cost of the HW and then of its implementation [84].

In addition, the additional circuit area constitutes a real challenge for CMOS devices [84]. According to this study, the authors confirm that increasing circuit area results in

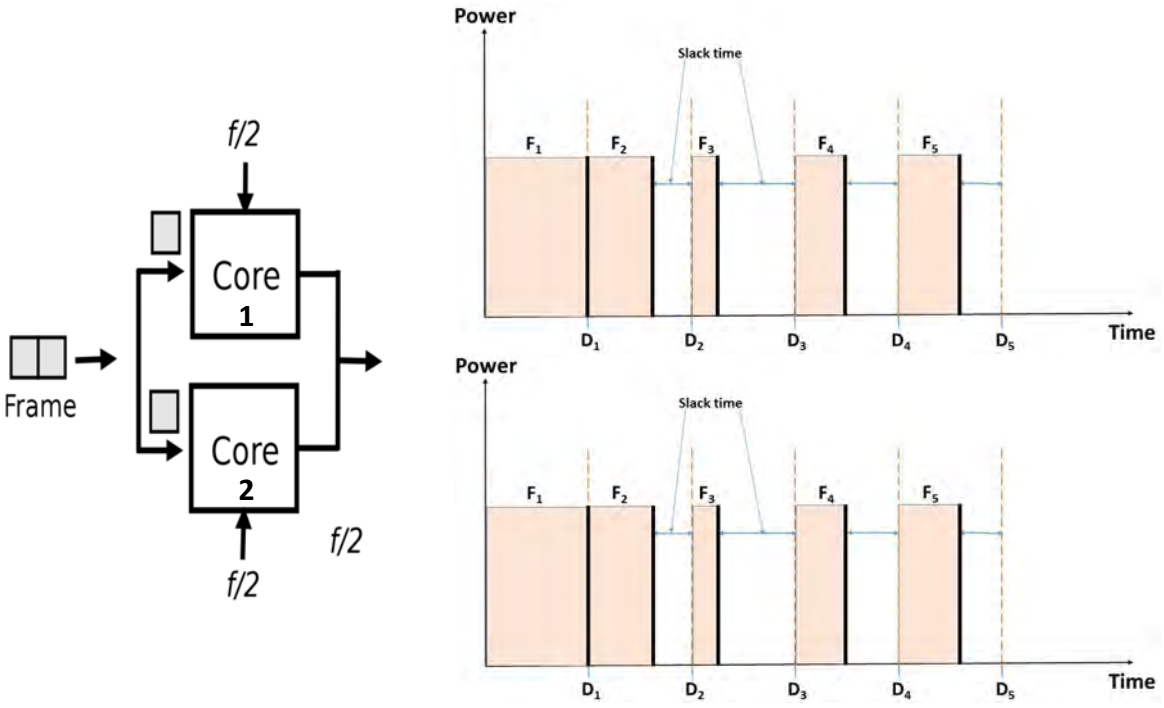


Figure 2.13 – Energy efficiency of parallel video decoding

an increasing static power consumption.

3- Specialized processing

In this section, different architectures using specialized processors for video decoding are discussed, from a high level point of view. HDIP, graphical processing unit (GPU), and digital signal processor (DSP) are presented. In this thesis, only the first one (HDIP) is studied.

Hardware video codecs (HDIPs): Generally speaking, a specialized processor is more energy efficient than a GPP [85]. The reason, according to [33][22], is that the energy inefficiency is intrinsic to the programmable nature (architecture) of GPPs, i.e., the control and communication overheads in executing an instruction in a GPP. For example, according to [22], only few pJ (less than 10pJ) are needed to execute an addition operation on 45-nm processor while 70pJ is needed to execute an entire instruction (containing the addition operation). This makes the specialized processors at least two orders of magnitude more energy efficient than GPPs [22].

In case of video decoding, thanks to customized architecture design, HDIPs can pro-

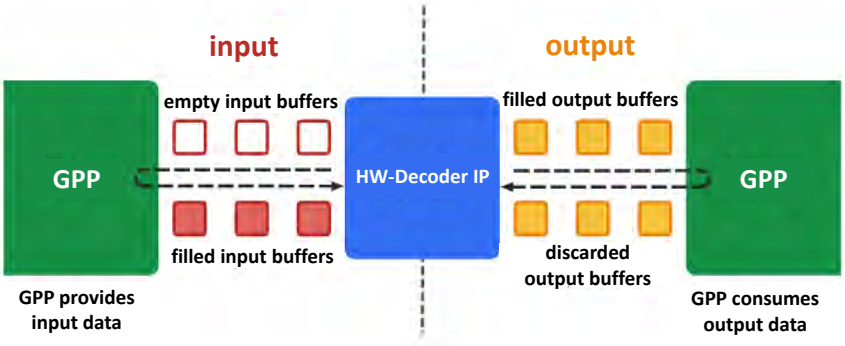


Figure 2.14 – HW video decoding diagram (Mediacodec API), from [86]

vide better energy consumption properties than GPPs by eliminating instruction fetching characterizing the programmable nature of GPP. In [32][33], the authors show that HDIPs achieve most of their energy efficiency gains by tuning data storage and compute structures and their connectivity to the data-flow and data-locality patterns in the codec.

In broad terms, an HDIP processes input data to generate output data. It processes video data asynchronously and uses a set of input and output buffers. Actually, asynchronously means that the HDIP receives a video frame to decode at regular intervals regardless of whether the previous frame was correctly decoded or not. This means that when receiving a video bitstream to decode, the video decoding period, D_{period}, is calculated using the following equation:

$$D_{\text{period}} = \frac{1}{video\_frame\_rate} \tag{2.10}$$

where $video\_frame\_rate$ is the frame rate of the decoded video sequence. Then, at the beginning of each period, as shown in Fig.2.14, the GPP requests (or receives) a reference of an empty input buffer, fills it up with data, and sends it to the HDIP for processing. This latter uses the data, transforms them, and copies the result into one of its empty output buffers. Finally, the GPP requests (or receives) a filled output buffer, consumes its content, and releases it back to the HDIP [86].

Furthermore, HDIPs are massively parallel and depend, for their performance, on the fact that the decoders output video data in 2-dimensional arrays [87]. Indeed, video

decoding functions, in general, exhibit massive data parallelism thanks to some schemes proposed by video codecs. Their architectures have been optimized for such parallelism. For example, they integrate extreme multi-threading HW or specific data handling and memory access optimization HW [88]. The main advantage of such accelerators is their energy efficiency.

The GPP generally considers the HDIP as an input/output (I/O) peripheral and communicates with it through I/O operations. This IPC may generate some overhead [89] [90]. The IPC also includes all other elements involved in the HW video decoding, such as memory transfers. When the HDIP is called to proceed with the decoding process, the GPP may enter the idle state and needs to handle the HW interrupt. This also generates some overhead.

The drawback of HDIPs is that they are not flexible and cannot be adapted to the evolution in video standards [91]. For example, HDIPs for MPEG VVC (Versatile Video Coding) standard is still not integrated on mobile devices at the time of writing this thesis.

In conclusion, a particular attention should be paid to the IPC overhead when comparing the performance of the HDIP with that of the GPP.

Graphical processing unit (GPU): A GPU is designed for parallel processing. It can process many pieces of data simultaneously, making it useful for video applications. It supports instruction set for accelerating geometric calculations such as the rotation and translation of vertices into different coordinate systems.

Thanks to the main characteristic of GPUs which is manipulating vector and matrix operations, they become more and more used for executing non graphical processing such as simulation, high performance computing [92], and especially video decoding [93][94][95].

In fact, the video decoding process is not entirely executed by the GPU. Only the modules that exhibit a high degree of data level parallelism and a low degree of branch divergence can lead to efficient GPU execution [94]. Therefore, the GPU works together with the GPP, forming a heterogeneous architecture. The video decoding modules need to be distributed properly between the GPU and GPP according to their computing characteristics. Usually, MC and color space conversion are better handled by the GPU, whereas the GPP is more suited for the inverse quantization, the inverse DCT, and the entropy decoding modules [93].

Besides, the video decoding operations between the GPP and GPU require efficient communication and pipeline consideration. Indeed, the control logic is executed at a fine

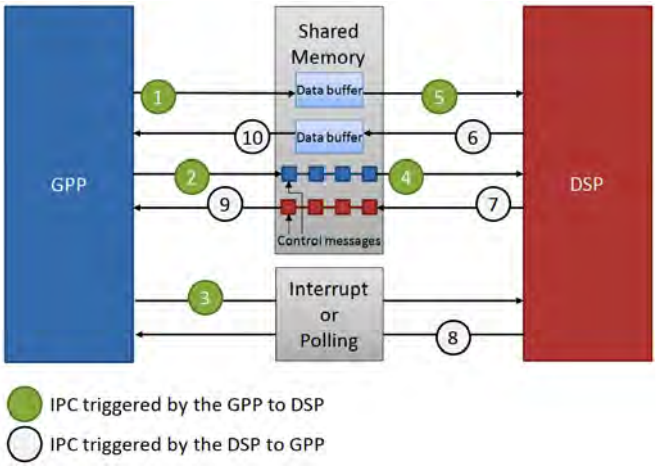


Figure 2.15 – IPC procedure diagram between GPP and DSP, from [98]

granularity in the GPP to synchronize the modules executed by the GPU and those executed by the GPP. This may have an impact on both performance and energy consumption [32]. For this reason, almost all GPU manufacturers do not propose pure GPU video decoding in their SoC. They, instead, propose video decoding solutions relying on hardware accelerator integrated in their GPU. Intel HD Graphics, Nvidia Pure Video, and AMD Unified Video Decoder can be cited as examples.

Digital signal processor (DSP): A DSP supports architecture and instruction set designed specifically for efficient implementation of digital signal processing algorithms such as Multiply Accumulate (MAC) and Fused Multiply Add (FMA) operations, which are extensively used in all kinds of matrix operations. It also makes use of parallelism by supporting Single Instruction Multiple Data (SIMD) parallelism, Very Long Instruction Word (VLIW) and super-scalar architecture.

DSP processor instruction sets are designed with the goal of enabling highly compact programs. Thus, conventional DSP processors use a relatively short instruction word—typically 16 or 24 bits long—to encode each multi-operation instruction [96].

Moreover, for example, since DSP often involves repetitive multiplications, DSP processors have fast multiplier hardware, explicit multiply instructions, and multiple bus connections to memory to retrieve multiple data operands at once.

Unlike GPU, a DSP has an advanced instruction set and is able to run complex programs. For example, a DSP is even able to run its own operating system [97]. As like the HDIP, a DSP is able to implement a full video codec leading to a limited coarse grained control from the GPP side.

The IPC procedure to communicate between the GPP and DSP is depicted in Fig.2.15 [98]. First, in step 1, the GPP copies data (the next frame to decode) to the shared memory which is accessible by both the GPP and DSP. Then, in step 2, the GPP sends a control message to the DSP. This message contains some information, such as the request which is the decoding function and the address of the frame to be decoded. It should be noted that the format and the address of control messages are communicated between the GPP and DSP once, at the beginning of the this procedure. After that, in step 3, the GPP notifies the DSP about the availability of a request. Next, in step 4, the DSP reads the control message to satisfy what the GPP requested from it. In step 5, the DSP gets the frame from the shared memory and then decodes it. In step 6, the DSP copies the decoded frame to the shared memory. In step 7, the DSP sends a control message, containing the decoded frame address and its length in the shared memory, to the GPP. In step 8, the DSP notifies the GPP that the request was processed. The GPP reads the response message in step 9. Finally, in step 10, the GPP gets the decoded frame from the shared memory.

Discussion

GPP processor vendors have maintained software compatibility among generations, whereas, historically, DSP processor vendors have not [96]. Upgrading to a newer, faster processor has typically required customers to learn a new architecture, new tools, and to rewrite their software completely [96].

The above sections highlighted two important points. First, to evaluate a video decoding system, both performance and energy consumption properties should be considered. Second, the study of a video decoding system (performance-energy) can be realized based on numerous parameters that can be triggered at different levels: (i) architecture, (ii) operating system, and (iii) application.

Accordingly, in the next sections, the studies focusing on characterizing and modeling both performance and energy consumption of video decoding at the operating system and application levels are surveyed. We consider mainly the impact of the video quality, processor frequency, and IPC overhead on the performance and energy consumption of video decoding.

At the architecture level, the video decoding is considered on different processor architectures: mono-core GPP, multi-core GPP, and HDIP in addition to GPP (referred to as GPP–HDIP, in this manuscript). It should be recalled that DSPs and GPUs are not

considered in our study.

2.2 Related work

In this section, the relevant studies related to subjects close to that of this thesis are summarized. First, the literature review on the video decoding performance and energy consumption characterization is highlighted. Second, the main contributions related to the reducing video decoding energy consumption are described.

2.2.1 Performances and energy consumption characterization of video decoding

In this section, research studies related to video decoding characterization are summarized. Various studies have been conducted for different generations of MPEG video codecs.

The performance and energy consumption characterization of the video decoding process consists in understanding this process with the objective of identifying the relevant parameters that impact its performance as well as its energy consumption. The characterization is one of the first elements to carry out when producing a new standard of video codec. Below, the state-of-the-art studies are grouped according to their level: (1) architecture, (2) OS, and (3) application.

1- Architecture level

At this level, the impact of the system architecture design on video decoding performance and energy consumption is studied. In the system architecture, the focus is put on the HW components and their types [5].

As explained in Section 2.1.4, different processor types (DSP, GPU, and HDIP) are able to perform video decoding with or without the help of GPP. Therefore, in [22], the authors studied the energy consumption of these processors and compared them for different applications. It should be noted that the authors studied only the energy related to the processors without taking into account any other component. The results showed that dedicated processors outperform GPPs by around 1000× in terms of energy efficiency. This is particularly true for multimedia applications.

5. In this manuscript, the processor types are those defined in Fig.1.4

Then, the efficiency of dedicated processors comes from the fact that they are very optimized for a very specific application, e.g., a video codec. Thus, implementing a new video codec requires studying it in detail and designing an appropriate system architecture. For instance, in [99], the authors showed that the increased complexity of HEVC over AVC entailed significant increase in HW complexity, both at the top-level of the video decoder and at the low-level processing blocks. For instance, the authors showed that HEVC needs more on-chip SRAM memory space than what is required in AVC. This extra-memory space is justified by the fact that the size of CTU (in HEVC) is 16× larger compared to that of MB (in AVC). This leads to additional area cost to implement HEVC on an Application-Specific Integrated Circuit (ASIC) test chip.

On the other hand, as explained in Section 2.1.4, modern GPPs gain their energy efficiency from the heterogeneity feature, via parallelism, when it is well exploited. For instance, in [100], the authors analyzed the parallel scalability of AVC decoding on many-core processors based on thread-level parallelism in terms of performance and energy efficiency. They discussed three levels of parallelism: (i) slice-level, (ii) frame-level, and (iii) MB-level. First of all, the authors showed that slice-level parallelism has two main limitations: (a) using many slices increases the bitrate, and (b) not all sequences contain many slices since the encoder determines the number of slices per frame. Second, the frame-level parallelism exploits the fact that some frames (B frames) are not used as reference frames and can therefore be processed in parallel. The authors showed that this level of parallelism is also not very scalable because usually there are no more than three B frames between consecutive P frames. Third, at the MB-level, it was shown that it is very scalable in many-core architecture. However, a particular attention should be paid to the communication and synchronization issues among the cores due to huge number of MBs being decoded in parallel.

2- OS level

In this section, the impact of parameters, triggered at the OS level, on video decoding performance and energy consumption is studied. At this level, the studies are mainly focused on the IPC, i.e., communication among the different processing units that participate to the video decoding process.

According to studies in [101][102][103], the IPC mechanism is critical, in particular for embedded systems with limited battery capacity. For instance, in [98], the authors evaluated the performance of the IPC mechanism required to exchange data between a

GPP and a DSP in an embedded heterogeneous multi-core architecture. Video decoding is used as a case study. It was shown that the memory choice influences the IPC. Indeed, using the GPP internal memory as a shared memory, i.e., the DSP can directly access the GPP internal memory, improves the IPC performance by 94%. Also, the results show that it is not always efficient to assign computation-intensive tasks to the DSP through IPC, as this introduces a considerable overhead. This result was also reported in [104]. Based on those findings, the authors developed IPC strategies. The system can dynamically adopt the IPC strategy based on the environmental parameters and system resource constraints.

The IPC can also take place inside the video decoding process. Indeed, in [105], the authors reported that HEVC decoding suffers from the difficulty of partitioning frame tiles among GPP cores when applying the loop-filtering module, see Section 2.1.1. The issue comes from the fact that this module makes frame tiles dependent of each other. These dependencies lead to a substantial communication overhead over different cores performing HEVC decoding.

3- Application level

In this section, the impact of parameters, triggered at the application level, on video decoding performance and energy consumption is studied. At this level, the studies can be grouped into two classes: (i) those which take into account the video decoding modules, and (ii) those which consider the video decoding process as a black box. In the latter case, the studies are mainly focused on the parameters described in Section 2.1.1.

In the first class, studies are interested in video decoding modules. For instance, HEVC has been profiled on both ARM and x86 architectures to study its complexity [106]. The profiling results on ARM and x86 are quite similar. On both architectures, the Motion Compensation and In-loop filters modules represent the most intensive computing tasks in the HEVC decoding process. Also, all intra (AI) mode decoding requires up to twice of the random access (RA) mode case at the same QP. Then, thanks to the profiling, the authors gave an indication of where HEVC may be more complex than its predecessor AVC and where it may be simpler. For instance, the deblocking filter is much lower in HEVC than it was in AVC, whereas the motion compensation is much higher in the later codec than it was in the earlier one. Finally, it was shown that 1080p60 and 480p30 decoding are feasible on laptops and mobile platforms, respectively (both within reasonable bitrate ranges).

Also, in [107], the authors studied the complexity of different modules of HEVC decoder, implemented on a mobile GPP, and their energy consumption. It was reported

that MC and In-loop Filtering blocks require together more than 83% of computation resources with respect to the global HEVC decoding complexity. Therefore, one can save energy by making more effort to reduce the complexity of these two block treatments.

The second class of studies do not take into account the video decoding modules. They consider the whole energy consumption of the video decoding application.

As mentioned in Section 2.1.1, the parameters triggered at the application level can be used to compare the coding efficiency of video codecs. For instance, in [17], the authors explained the syntax and coding structures of various video codecs: H.262/MPEG-2 Video, H.263, MPEG-4 Visual, H.264/MPEG-4 (AVC), and HEVC. The authors used two assessment metrics to compare those video codecs in terms of coding efficiency: (i) PSNR, and (ii) bitrate. The results showed that the subjective benefit for HEVC seems to exceed the objective one measured using PSNR and bitrate. In addition, the benefit is greater for low bitrates, higher-resolution video content, and low-delay encoding applications.

Then, multimedia applications use parameters at the application level to offer the best QoS to end users. For instance, the authors in [104] gave a comprehensive and comparative study of the performance and energy consumption of AVC decoding applications on embedded heterogeneous platforms containing a GPP and a DSP. The authors proposed guidelines for selecting the appropriate processor (GPP or DSP) of the studied SoC when decoding a video sequence given its bitrate and resolution.

In case of HDIP, the authors in [108] investigated the energy an HEVC HW decoder needs for decoding and displaying an HEVC-coded bit-stream. The authors studied the influence of the resolution, frame rate, and bitrate on the video decoding energy consumption. They found that these parameters are correlated to the HEVC HW decoder energy consumption. Therefore, the results were used to establish a model capable of estimating the HEVC HW decoding energy consumption of a video sequence given the aforementioned parameters.

2.2.2 Discussion

The characterization of the video decoding process is a very challenging task in a context of the ever growing parameters that impact, on the one hand, the video quality and, on the other hand, the energy consumption. Indeed, it is not obvious to make a clear general idea about the performance and energy consumption of video decoding. Each state-of-the-art work stated above tackled partially this issue, e.g., a subset of parameters and/or levels. Here are three examples. First, HEVC has been profiled on both

ARM and x86 architectures to study its complexity [106]. However, the impact on power consumption was not investigated. Second, in [109], application, system, and architecture parameters are considered for performance characterization, but only for an x86 processor. Third, in [110], different architectures were considered but the processor frequency parameter was not covered.

The difficulty to cover several parameters to characterize the performance and energy consumption of video decoding makes the comparison of the different results pretty impossible in terms of validation. The reason is that different studies do not conduct experiments on the same conditions: platforms (different technology process), heterogeneous software stacks, etc. Therefore, one should carry out these comparisons in same conditions and make different scenarios.

In this context, a characterization methodology to investigate the HEVC decoding performance and energy consumption on mobile platforms is proposed in this thesis. Two approaches of video decoding are analyzed: HW using the GPP–HDIP, and SW using a heterogeneous GPP. The performance and energy consumption of video decoding are characterized at two levels: (i) operating system level: GPP number of cores and their frequencies, and (ii) application level: video quality parameters. The objective is to compare the behavior of power consumption of both kinds of platforms to understand to what extent and in which cases GPP–HDIP decoding is much better than GPP decoding. To the best of our knowledge, there is no previous work that compared HW and SW HEVC decoding power consumption on mobile platforms, while considering all the aforementioned parameters.

2.2.3 Video decoding energy consumption optimization

In this section, research studies related to video decoding energy consumption optimization are summarized. Various state-of-the-art studies have been conducted on video decoding energy consumption for previous and current video codecs. These studies are grouped according to the decoding parallelism level: (i) frame-by-frame, (ii) tiling, (iii) WPP, and (iv) MB (for AVC and older video codecs).

Frame level parallelism

In [111], the authors proposed a light DVFS-enabled SW adapted to the much varying processing load of HEVC real-time decoding on mobile devices. The proposed solution

consists in dynamically adapting the processing frequency to the video frames characteristics. The complexity of the $(L+1)^{\text{th}}$ frame is estimated by the average decoding time of the last 'L' decoded frames, 'L' being a parameter. Experimental results showed more than 50% of power savings on a real-time video decoding when compared to the same software managed by the Ondemand Linux power management governor.

On a heterogeneous ARM big.LITTLE architecture, the authors, in [112], proposed strategies which make profit of data- and task-level parallelism as well as a new frequency control system using DVFS, based on an estimation of the decoding complexity. These strategies aim at optimizing the HEVC SW decoding power consumption. The experimental results showed that the proposed open-source implementation can reach an energy consumption below 21 nJ/px for 720p decoding at 2.2 Mbits/s.

In [113], the authors provided a systematic HEVC decoding solution, on GPP, consisting of structure-level, data-level, and task-level approaches. First, at the structure level, the solution consists in redesigning overall structure of an HEVC decoder with data redundancy: four-stage decoding workflow. Then, at the data level, the authors enhanced the SIMD mechanism by developing low-complexity MC, transpose-free transform, symmetric deblocking filter, and parallel-index sample adaptive offset. Finally, a frame-based task-level parallel framework is employed with a flexible entry scheme to efficiently support the simultaneous processing of multiple decoding tasks for different HEVC parallel strategies. The proposed solution assigns a frame to the first available core. The obtained results showed that the proposed decoding solution can well meet the demand of 4k HEVC video decoding on x86 processors and 720p video decoding on prevalent ARM processors.

In [114], the authors introduced a method, based on convex problem solving, that determines the most energy efficient operating point in terms of GPP frequency and number of GPP active cores in a mobile multi-processor SoC (MPSoC) to perform HEVC decoding. The proposed method jointly considers the DPM, DVFS, and parallelism capabilities of, on the one hand, the targeted MPSoC and, on the other hand, the HEVC application. The experimental results, conducted on a Samsung Exynos 5410 MPSoC, showed a reduction in energy of up to 27% when compared to ASAP (As-Slow-As-Possible) and AFAP (As-Fast-As-Possible).

In [115], the authors investigated the impact of code optimization techniques and low-power states of GPPs on the power efficiency of HEVC decoding. For that, they exploited the SIMD and multi-threading, to decode multiple frames in parallel, in addition to low-power states that reduce the active as well as the idle powers. The results showed that «

exploiting slack » is more power efficient than « race to idle » for all evaluated platforms representing smartphone, tablet, laptop, and desktop computing systems. Furthermore, on systems where the processor consumes only a small fraction of the energy, reducing the frequency would actually increase the total energy consumption due to longer execution times. Moreover, the energy savings SIMD provides are, in most cases, proportional to the speedup it provides because the power consumption of the SIMD code is hardly higher than that of the scalar code. Finally, current operating systems do not exploit the power saving modes of the processor effectively. Indeed, deep C-states are not entered as often as possible.

Another mechanism to reduce energy consumption of video decoding is to reduce the complexity of its modules by enabling/disabling some features on-the-fly. For example, in [116], the authors proposed an HEVC decoder with tunable decoding quality levels for maximum power savings. The tuning functionality is based on dynamic activation of in-loop and interpolation filters, reducing the complexity of the decoder. The results showed that the modified HEVC decoder can save up to 28% of power consumption in real-world platforms while keeping better quality than decoding with AVC.

Approximate computing can also be used to save energy when performing video decoding. In [117], the authors applied the approximate computing to lower the complexity of HEVC decoding, at the algorithmic-level. The idea of approximate computing is to skip some modules (blocks) or replace them by others of lower complexity. It aims at exploring the trade-offs between energy consumption and quality of service. For instance, the frequency of block skipping is set as a percentage of frames where filters are skipped. By setting the skip control parameter to 0%, the decoder is similar to the legacy HEVC. If the skip control is set to 100%, the in-loop filters and the MC interpolation filters are skipped permanently. The results showed that the applied approximate computing allowed energy reductions of up to 40% for a limited degradation of the application QoS.

Another way to minimize energy consumption of video decoding is to modify its algorithm in a way to reduce its complexity and thus its energy consumption. This was the idea proposed in [118]. The authors proposed a modified HEVC decoder that fosters energy-efficient implementations on embedded systems. The idea behind the proposed solution is that the HEVC decoder can change dynamically the filters inside its decoding architecture, specially the MC filters as they represent the highest share of the processing load. The authors also proposed to control the quality degradation thanks to Green Metadata embedded in the video stream, adopted within the MPEG standard. Experiments

showed that without changing the incoming bitstream, a software implementation of the modified decoder can save up to 28% of energy while introducing limited distortion.

In [41], the authors proposed a technique to reduce the energy consumption of video decoding using the feedback control technique. They modeled the multimedia decoding process as a discrete-time system excited by random input sequences representing the incoming stream. The proposed technique does not make any assumptions about the workload streams. For that, the solution was to control the GPP frequency (using DVFS mechanism) based on the output buffer size, i.e., the number of decoded frames waiting for display. The GPP frequency is scaled up or down depending on the buffer size and the display rate. Experimental results showed that the proposed design achieves equal or better energy efficiency than the existing techniques, brings close to ideal playback quality (about a 1% deadline miss rate), and only requires a moderate buffer size (5-frames buffer).

Tiling parallelism

In [119], the authors made use of mobile computing systems that have asymmetric processors, such as ARM big.LITTLE, to save energy when decoding HEVC video content. The solution is based on scheduling frame tiles among the heterogeneous cores. The proposed method partitions a frame into non-uniform tiles, i.e., tiles which do not have the same resolution, and allocates threads to suitable cores. The scheduling is based on the tile complexity as well as the performance ratio between big and LITTLE cores. The tile complexity is estimated by its resolution using a regression model « resolution-complexity ». Experimental results, conducted on the test sequences of common test condition (CTC), showed that the decoding speed was improved by 17% with implemented multi-threading module on ARM asymmetric multi-core systems.

In [120], the authors proposed an allocation method considering the computational ability of mobile asymmetric multi-cores, such as ARM big.LITTLE, as well as the computational complexity of each tile. The scheduling among the asymmetric cores is based on the tile complexity as well as the performance ratio between big and LITTLE cores. The tile complexity is estimated by the number of PUs [6] incorporated in the tile. This choice is justified by the fact that the conducted experiments showed a close correlation between the amount of PU partitioning and the computational complexity (decoding time). The results showed average performance gains of decoding: around 36% for 12 tiles, 28% for 18 tiles, and 31% for 24 tiles.

6. Prediction Unit (PU) is a sub-part of a tile.

In [121], the authors suggested a tile partitioning algorithm of HEVC encoding to improve overall HEVC decoder parallelization. In the proposed solution, a frame is divided into tiles that are processed independently and concurrently on multi-cores. The proposed solution consists in balancing the workloads of tiles among asymmetric cores. The tile complexity is estimated by the number of bits encoded in each CTU of the tile. Experimental results showed around 15% improvements of decoding speedup and 12% energy savings on a 16-core system.

Wavefront parallel processing parallelism

In [122], the authors developed a strategy which exploits the mobile asymmetric multi-core processors, such as ARM big.LITTLE. They proposed an architecture-aware implementation of an HEVC decoder that embeds a criticality-aware scheduling strategy tuned for a Samsung Exynos 5422 SoC. The proposed solution follows the parallelization approach dictated by WPP to distribute the workload (CTU rows/tasks) among the fast (big) Cortex-A15 and the slow (LITTLE) Cortex-A7 cores on-the-fly. The solution is based on task migration between big and LITTLE cores such that all cores are busy all the time. The experiments demonstrated that the exploitation of the Cortex-A7 cores not only enhances the overall performance, but also contributes to improve the energy efficiency of decoding pipeline. The obtained results exposed a 1080p real-time HEVC decoding at 24 frames/s and a reduction of energy consumption over 20%.

In [123], the authors studied the high-level parallelism schemes proposed with HEVC (slices, tiling, and WPP) which make it more parallel-friendly compared to its predecessors. All these parallelism schemes share in common the idea of creating frame partitions that can be processed in parallel by multiple threads/cores in a parallel system, and differ in terms of data dependencies. Then, the authors proposed an approach called Overlapped WaveFront (OWF) that can be implemented on top of WPP. The proposed decoder consists of three pipeline stages: parse, issue, and output. Each of these stages is performed by a different thread. The proposed solution (OWF) achieves higher performance and scalability than both WPP and tiling. Comparing WPP to tiling, the experiments showed that the tiling approach achieves slightly higher performance than WPP (7% higher on average over all resolutions, using 12 cores). In the presented implementation, however, WPP has higher memory bandwidth efficiency than tiling.

MB level parallelism

In [124], the authors presented an optimization of video decoding, exploiting parallelism at the block level (instruction and SIMD parallelism) and at higher levels (task and pipeline parallelism) to decompose any video decoder in a heterogeneous dual core SIMD processor. First, they profiled the video decoding code and tracked the most expensive modules on which the optimizations are focused. Second, the proposed optimizations consist in improving memory access and reducing the video decoding modules complexity. The proposed decomposition in a dual-core processor is the following: stream processor (SP): is used to parse and decode the video bitstream (Entropy Decoding), and pixel processor (PP): performs most of the heavy duty computations of video decoding using SIMD TIE instructions. Simulation results showed that the exploitation of parallelism at all levels of granularity, especially at higher levels, can give a total speed-up of more than 195x compared to a software x86-based implementation.

In [125], the authors presented three efficient DVS models for an MPEG decoder: two models are offline, and one model is online. For that, they developed a workload prediction model (linear) based on the block level statistics of each MPEG frame. First, the GOP-optimal algorithm buffers all the frames in a GOP [7], estimates their workload, and decodes the entire GOP using a constant voltage. Second, the Global-Grouping gathers consecutive intervals into groups. It divides the decoding time into groups. Inside a given group, a constant voltage is used. This voltage is selected such that all frames within this group can be decoded before their display deadline. Third, the Dynamic-Grouping is an online heuristic based on Global-Grouping. It buffers the input frames up to a certain window size (e.g., a GOP size). The workload of each frame inside the window is predicted and grouping is applied within this window. The experimental results showed that the Dynamic-Grouping algorithm gives the best trade-off between energy reduction and the quality of decoding, e.g., 10% of energy saving while using only a 2-frames buffer size to satisfy the real-time constraint.

2.2.4 Summary

The literature review studies are summarized in the following tables: Table 2.2 for the video decoding performance and power/energy consumption characterization, and Table

7. Group of Pictures (GOP): is a sequence of frames processed by group instead of frames-by-frame. It is comprised of I, P, and B frames.

2.3 for the video decoding power/energy consumption optimization.

Table 2.2
Video decoding performance and energy consumption characterization literature review summary

Ref	**Architecture level**				**OS level**			**Application level**				
	GPP	HDIP	DSP	GPU	IPC	#cores	Clock frequency	PSNR	Bitrate	Frame rate	Resolution	Internal modules
[22]	Yes	Yes	Yes	Yes	No			No				
[99]	No	Yes	No	No	No			No				
[100]	Yes	No	No	No	No			No				
[101][102] [103][98]	Yes	No	Yes	No	Yes	No	No	No				
[105]	Yes	No	No	No	Yes	No	No	No				
[106]	Yes	No	No	No	No			No	No	No	No	Yes
[107]	Yes	No	No	No	No			No	No	No	No	Yes
[17]	Yes	No	No	No	No			Yes	Yes	No	No	No
[104]	Yes	No	Yes	No	Yes	Yes	No	No	Yes	No	Yes	No
[108]	No	Yes	No	No	No			No	No	Yes	Yes	No
Our work	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

Table 2.3
Video decoding energy consumption optimization literature review summary

Ref	**Parallelism scheme**	**Scheduling with load balancing**	**DVFS**	**Output buffer**	**Complexity estimation metric**	**Codec**
[111]	Frame	No	Yes	Yes	The last 'L' decoded frames	HEVC
[112]	Frame	Yes	Yes	No	-	HEVC
[113][117][118]	Frame	No	No	No	-	HEVC
[114][116]	Frame	No	Yes	No	-	HEVC
[115]	Frame	No	Yes	Yes	-	HEVC
[41]	Frame	No	Yes	Yes	-	MPEG
[119]	Tiling	Yes	No	No	Tile resolution	HEVC
[120]	Tiling	Yes	No	No	Number of PUs incorporated in a tile	HEVC
[121]	Tiling	Yes	No	No	Number of bits of each CTU of a tile	HEVC
[122]	WPP	No	No	No	-	HEVC
[123]	WPP	No	No	Yes	-	HEVC
[124]	MB	No	No	No	-	Advanced Video Standard (AVS)
[125]	MB	No	Yes	Yes	Block (number of IDCT coefficients, motion compensation)	MPEG
Our work	Frame	Yes	Yes	Yes	Frame (type, size)	HEVC

2.2.5 Discussion

The different studies stated above were grouped based on the complexity prediction granularity: frame, tiling, wavefront parallel processing, and MB (for AVC and older video codecs). The video decoding energy saving on heterogeneous mobile architectures using DVFS requires a model having a prior knowledge of the workload in order to adjust the GPP frequency accordingly. Choosing or rejecting a model depends on the decoding latency tolerance, i.e., a model is accepted if the decoding latency is accepted, and is rejected when no latency is allowed.

Then, most of the presented studies rely on detailed information about the workload to decode, e.g., number of bits contained in a CTU, in order to estimate its complexity and then adjust the GPP clock frequency. However, this is not always the case in real-world scenarios. Indeed, these information are only available if they are collected at the encoder side.

Next, the number of state-of-the-art studies based on the frame-by-frame parallelism scheme is much greater than those based on a sub-part of frame (tiling, WPP, and MB (for AVC and older video codecs)). The reason is that decoding at block level requires more attention to the communication overhead among the threads dealing with each block. This is particularly true for MB parallelism scheme where a 1080p frame may contain thousands of $16x16$ MBs.

In this thesis, to save the energy of video decoding, a solution based on classification and frequency scaling is proposed. It is composed of three phases: (i) modeling of frame complexity, (ii) classification, and (iii) frequency scaling using feedback control. In the first phase, a logistic model is established. It is used to classify the frames into two groups: (a) the most complex frames, and (b) the least complex frames. The frames of the first group are submitted to high performance GPP cores, whereas the frames of the second group are submitted to the energy-efficient ones. The frames are decoded in parallel between the selected GPP cores thanks to the tiling or WPP parallelism schemes [8]. Next, in the second phase, for each frame, the classifier uses the model established to decide which GPP cores will decode it. In the third phase, the selected GPP cores frequency is adjusted using the feedback control technique proposed in [41]. This latter consists in monitoring the output buffer size. Depending on this size, the GPP cores are sped up or slowed down.

8. The choice between tiling or WPP depends on the encoding configuration, i.e., the parallelism scheme is decided during the encoding process.

2.2.6 Conclusion

In this chapter, all the necessary background knowledge to understand this thesis is described. The chapter is divided into two main parts: (i) background, and (ii) related work. In the first part, a broad introduction to MPEG video codecs and the energy consumption of video decoding are given. They are mainly based on reducing spatial and temporal redundancy among video frames. Then, via an example, the main strategies to save energy of video decoding were illustrated. We concluded that, thanks to DVFS, GPP processors have a great potential to save energy of video decoding, while paying attention to the heterogeneity of the cores.

In the second part, the most relevant studies found in the state of the art which are related to: (a) video decoding performance and energy consumption characterization, and (b) video decoding energy consumption optimization are highlighted. First, for the former, the studies were grouped according to the abstraction level: architecture, operating system, and application. In short, characterizing video decoding process in terms of performance and energy consumption should take into account different abstraction levels, and different parameters at each level. Then, for the latter, the optimization studies were classified according to the decoding parallelism granularity: frame, tiling, WPP, and MB (for AVC and older video codecs) levels.

Finally, our contributions are introduced by explaining the gaps that are filled to address the challenges stated in the problem statement.

These contributions will be described in detail in the next chapters. In chapter 3, our proposed methodology to characterize the performance and energy consumption of video decoding is described. This chapter resolves the problem statement (RQ1 and RQ2). Then, in chapter 4, the results of the performance and energy consumption characterization applied on HEVC decoding are presented. After that, in chapter 5, our proposed strategy to optimize the energy consumption of HEVC decoding on a heterogeneous mobile architecture, such as ARM big.LITTLE, is detailed. This chapter resolves the problem statement (RQ3 and RQ4)

Chapter 3

Video decoding performance and energy consumption characterization

In this chapter, our proposed characterization methodology to solve the problem statement (RQ1 [1] and RQ2 [2]) is presented. First, an overview of the proposed methodology is described. Second, the overall power model used to calculate the power/energy consumed when running video decoding is defined. Third, the characterization methodology of the HW and SW video decoding performance and energy consumption for an objective analysis and comparison is detailed. Finally, for better readability of the manuscript, the proposed methodology and all the notations used are summarized in Table 3.2 and Table 3.3, respectively.

3.1 Characterization methodology overview

This section gives a global overview of our characterization methodology to compare the performance and energy consumption of the HW and SW video decoding. First, a brief description of the power model used to calculate the power/energy consumption is given. Then, the levels at which our methodology intervenes as well as the parameters triggered at each level are presented

The video decoding characterization allows to find the parameters that impact the performance and energy consumption of video decoding. Our approach is based on experimental measurements, i.e., we calculate power, using real measurement tools, while varying different parameters in order to study their relevance. Our proposed methodology

1. RQ1: How relevant are the video decoders (HW and SW) with regards to the parameters impacting the video decoding process?
2. RQ2: To what extent do HW video decoders (HDIPs) outperform SW ones?

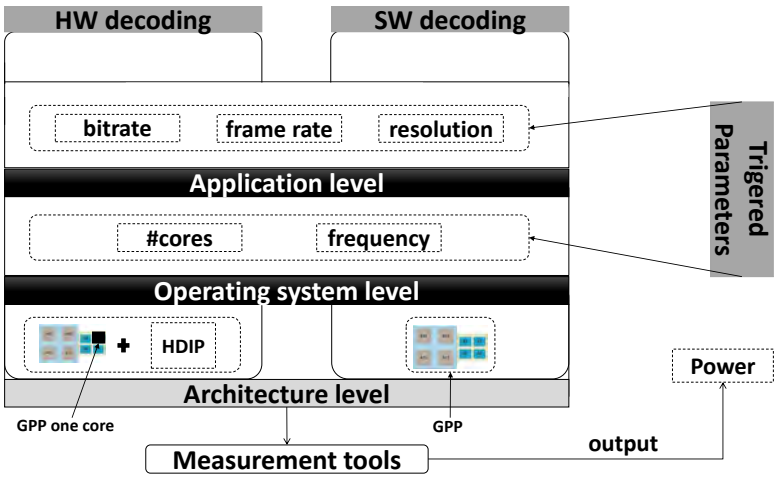


Figure 3.1 – The proposed characterization methodology overview

is designed independently of any underlying experimental environment in order to make it generalizable to almost any platform.

Fig.3.1 depicts an overview of our proposed methodology. The inputs are the parameters for which the impact on energy consumption is estimated and the output is the consumed power. Our objective is to study and compare, by measurement, the energy consumption of two video decoding approaches: HW and SW [3]. For that purpose, we first developed a power model which allows to calculate the consumed power/energy when performing video decoding. Then, we studied the impact, on energy consumption, of several parameters triggered at two hierarchical levels: operating system and application.

The established power model is based on a decomposition approach. This approach is used to break up the power into components related to different parts of a SoC in order to study each of the components separately, see Fig.3.2. Historically, this approach was introduced to solve complex problems by breaking them up into smaller ones, simpler to solve [126].

The decomposition approach consists in getting the different components of the power in a hierarchical form, e.g., a tree. The power components at $level_i$ are used to calculate a power component of $level_{i+1}$, via simple operations, such as addition or subtraction, see Fig3.2. For instance, in this figure: $P_0 = P_{1.1} + P_{1.2} + P_{1.3}$.

Thanks to this approach, the power/energy related to video decoding is separated from

3. Throughout the manuscript, the two video decoding approaches refer to: HW and SW video decoding.

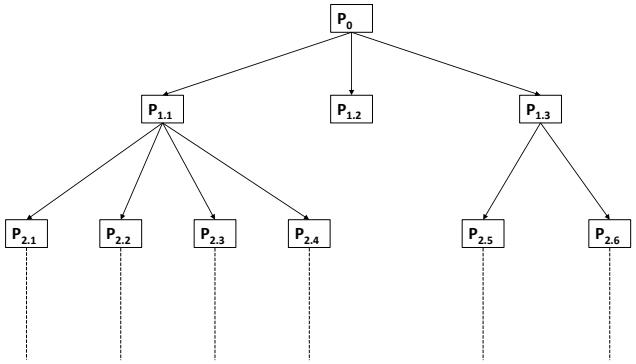


Figure 3.2 – Decomposition approach principle

that related to the global platform power/energy [4]. Also, we shall show that the impact of a heterogeneous GPP architecture on the energy consumption can be illustrated by the decomposition approach.

The architecture is the same in all the conducted experiments. An HDIP in addition to a GPP (referred to as GPP–HDIP, in this manuscript), are used for the HW video decoding, and a heterogeneous GPP is used for the SW one.

3.1.1 Operating system level

As explained in Section 1.1.5, the OS has a detailed knowledge about the processes and resources allocation as it is one of its main characteristics. Thus, it is worth exploiting this knowledge to study the performance and energy consumption of video decoding.

At this level, we proceed in three steps to evaluate the idle power of the tested platforms and their active power when performing the HW and SW video decoding. Then, we compare the two video decoding approaches.

In the first step, we study the power consumption of the HW and SW video decoding platforms during the idle state. The idle state corresponds to the state where there is no program running at the user level.

In the second step, we first study the HW video decoding power consumed by GPP–HDIP. Then, we investigate the power which is not consumed by GPP–HDIP. It includes the power related to the IPC between the HDIP and other processing elements involved

4. The global platform power consumption is the power consumed by the entire platform.

in the video decoding, e.g., memory, see Section 2.1.4.

In the third step, we first study the SW video decoding energy consumed by GPP. This study consists in varying parameters related to a heterogeneous multi-core GPP (number of cores and their clock frequencies). Next, we investigate the power which is not consumed by GPP. It includes the power corresponding to the remaining elements present in the platform.

3.1.2 Application level

As shown in Fig.1.5, the application level is at the top of the system hierarchy. At this level, the study is carried out on a specific process that can be controlled by an end-user, such as video decoding application. Also, video decoding is one of the applications which are very resilient at the application level. Indeed, it can tolerate inexact outputs, such as missed decoded frames [127]. For those reasons, we included this level study in our proposed methodology.

At this level, we evaluate the impact of the video parameters (bitrate, frame-rate, and resolution) on the video decoding energy consumption in both HW and SW implementations. These parameters may affect either the performance (decoding time) or the energy consumption. Then, we compare the energy consumption of HW and SW video decoding based on those parameters.

3.2 Overall power model

In our experiments, we calculate power to study and compare the HW and SW video decoding energy consumption. From Equation (2.5), which formulates the total power consumption of a CMOS circuit, we are only interested in the dynamic power part. The reason is that, in this thesis, we study the power/energy consumption of the video decoding process and that the static power is independent of any running process. The static power is thus considered as beyond the scope of this thesis.

In this thesis, our proposed power model is based on a decomposition approach. This approach allows to evaluate separately different components of the consumed power. First, we consider that the dynamic power of a processing unit is the sum of the idle and active

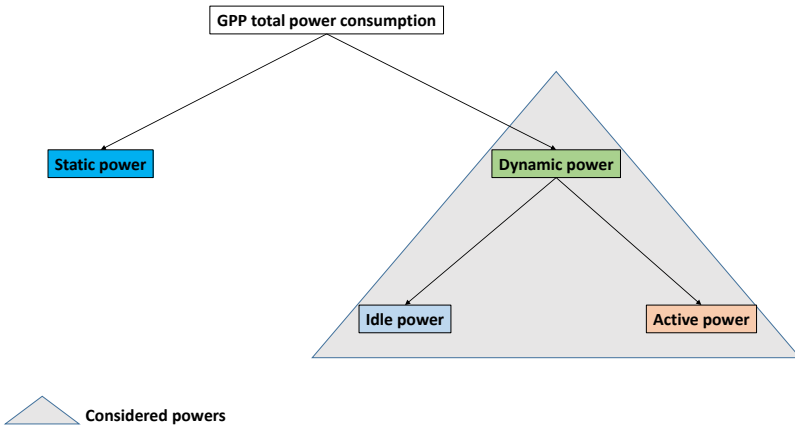


Figure 3.3 – Power decomposition (1)

powers, as depicted in Fig.3.3 and formulated in Equation (3.1) [5]:

$$P_{\text{dyn}} = P_{\text{idle}} + \hat{P}_{\text{active}} \tag{3.1}$$

Then, based on the above equation, the global platform dynamic power consumption is formulated in Equation (3.2):

$$P_{\text{glob\_dyn}} = P_{\text{glob\_idle}} + \hat{P}_{\text{glob\_active}} \tag{3.2}$$

where $P_{\text{glob\_dyn}}$ corresponds to the global platform dynamic power when running the video decoding process, $P_{\text{glob\_idle}}$ is the global platform power in idle state, and $\hat{P}_{\text{glob\_active}}$ is the effective power related to the video decoding.

The idle power is the power consumed in the state where there is no process running at the user level [6]. It should be noted that the idle power is different from that of the static power, see Section 2.1.2.

Regarding the active power, it is the power consumed in the state where there is one or more processes running at the user level. This power is only related to the processes being running, i.e., without taking into account the idle power.

5. Throughout the manuscript, a symbol with *circumflex* indicates a value (power, energy, or ratio) deduced from other calculated powers/energies which are obtained from measurement tools.

6. A process running at the user level is a process which is launched and can be controlled by a user, whereas a process running at the kernel level cannot be controlled by a user [128]

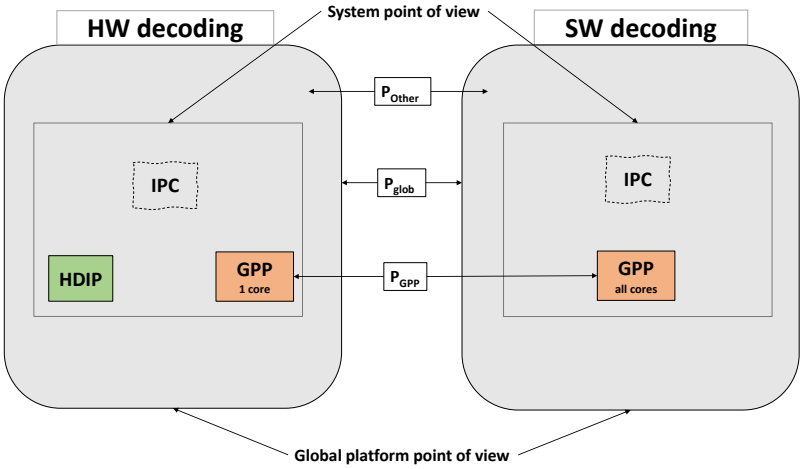


Figure 3.4 – Power decomposition (2)

Each power (idle or active) is composed of two power components as depicted in Fig.3.4 and formulated in the following equations:

$$P_{\text{glob\_idle}} = \hat{P}_{\text{GPP\_idle}} + \hat{P}_{\text{Other\_idle}} \tag{3.3}$$

$$\hat{P}_{\text{glob\_active}} = \hat{P}_{\text{GPP\_active}} + \hat{P}_{\text{Other\_active}} \tag{3.4}$$

where $\hat{P}_{\text{GPP}}$ and $\hat{P}_{\text{Other}}$ are the power consumption of the GPP and the remaining (other) elements present in the platform, respectively.

According to Equations (3.2), (7.7), and (7.8), the global dynamic power of a platform can be formulated as the sum of the dynamic power of the GPP, $\hat{P}_{\text{GPP\_dyn}}$, and that of the remaining elements present in the platform, $\hat{P}_{\text{Other\_dyn}}$. This can be translated by the following equation:

$$P_{\text{glob\_dyn}} = \hat{P}_{\text{GPP\_dyn}} + \hat{P}_{\text{Other\_dyn}} \tag{3.5}$$

Then, in a multi-core GPP architecture, the dynamic power consumption of the GPP is formulated as a function of the number of cores that are executing a program (e.g., video

decoding) and those that are not, i.e., in idle state, as given by the following equation.

$$\hat{P}_{\text{GPP\_dyn}} = nb_{\text{run}} * \hat{P}_{\text{GPP\_1\_core\_dyn\_run}} + nb_{\text{not\_run}} * \hat{P}_{\text{GPP\_1\_core\_dyn\_not\_run}}$$

where nb_{run} and $nb_{\text{not\_run}}$ are the number of cores that are executing a program and those that are not, respectively, whereas $\hat{P}_{\text{GPP\_1\_core\_dyn\_run}}$ and $\hat{P}_{\text{GPP\_1\_core\_dyn\_not\_run}}$ are the power consumption of one GPP core that is running a program and that which is not, respectively. It should be noted that we assume that the different GPP cores are identical, i.e., they consume the same amount of power either in idle or active state (when they perform the same process).

Next, as mentioned in Equation (3.1), the dynamic power consumption of a processing unit is the sum of the idle and active powers. This leads to Equation (3.6):

$$\hat{P}_{\text{GPP\_dyn}} = nb_{\text{run}} * (\hat{P}_{\text{GPP\_1\_core\_idle}} + \hat{P}_{\text{GPP\_1\_core\_active}}) + nb_{\text{not\_run}} * \hat{P}_{\text{GPP\_1\_core\_idle}} \tag{3.6}$$

In case of a heterogeneous multi-core GPP architecture, the dynamic power consumption of the GPP is formulated as a function of the number of cores that are executing a program (e.g., video decoding) and those that are not, i.e., in idle state, for each core type. For instance, in a SoC equipped with an ARM big.LITTLE architecture, the dynamic power consumption of the GPP is given by the following equation:

$$\begin{aligned}\hat{P}_{\text{GPP\_dyn}} = {} & nb_{\text{run\_big}} * (\hat{P}_{\text{GPP\_1\_core\_idle\_big}} + \hat{P}_{\text{GPP\_1\_core\_active\_big}}) \\ & + nb_{\text{not\_run\_big}} * \hat{P}_{\text{GPP\_1\_core\_idle\_big}} \\ & + nb_{\text{run\_LITTLE}} * (\hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}} + \hat{P}_{\text{GPP\_1\_core\_active\_LITTLE}}) \\ & + nb_{\text{not\_run\_LITTLE}} * \hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}}\end{aligned} \tag{3.7}$$

where $nb_{\text{run\_big}}$, $nb_{\text{run\_LITTLE}}$, $nb_{\text{not\_run\_big}}$, and $nb_{\text{not\_run\_LITTLE}}$ are the number of big and LITTLE cores that are executing a program and those that are not, respectively, whereas $\hat{P}_{\text{GPP\_1\_core\_idle\_big}}$, $\hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}}$, $\hat{P}_{\text{GPP\_1\_core\_active\_big}}$, and $\hat{P}_{\text{GPP\_1\_core\_active\_LITTLE}}$ are the power consumption of one big and LITTLE GPP core in idle and active states, respectively.

To illustrate Equation (7.10), let's take the following example, depicted in Fig3.5. In this example, a video decoding application is executed by an ARM big.LITTLE GPP.

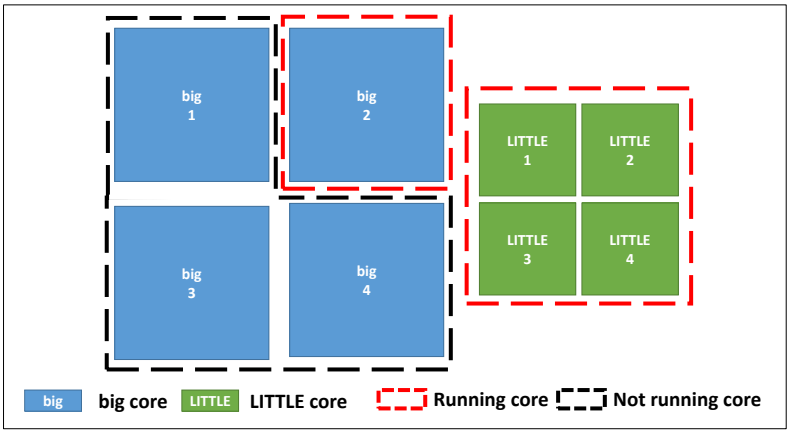


Figure 3.5 – Example of a heterogeneous ARM big.LITTLE GPP

This GPP is composed of eight cores: 4× big cores and 4× LITTLE cores. Suppose that the video decoding application needs one big core and four LITTLE cores in order to perform decoding in real-time. Therefore, the dynamic power consumption of this GPP is $\hat{P}_{\text{GPP\_dyn\_example}}$:

$$\begin{aligned}\hat{P}_{\text{GPP\_dyn\_example}} = &(\hat{P}_{\text{GPP\_1\_core\_idle\_big}} + \hat{P}_{\text{GPP\_1\_core\_active\_big}}) \\ &+ 3 * \hat{P}_{\text{GPP\_1\_core\_idle\_big}} \\ &+ 4 * (\hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}} + \hat{P}_{\text{GPP\_1\_core\_active\_LITTLE}})\end{aligned}$$

where $nb_{\text{run\_big}} = 1, nb_{\text{not\_run\_big}} = 3, nb_{\text{run\_LITTLE}} = 4$, and $nb_{\text{not\_run\_LITTLE}} = 0$.

Then, the power consumption of one GPP core in idle state, $\hat{P}_{\text{GPP\_1\_core\_idle}}$, is given by Equation (7.11). It is deduced, using Equation (7.7), by subtracting the global power consumption when turning on one GPP core from that when turning on two GPP cores, both in idle state.

$$\begin{aligned}P_{\text{glob\_2\_cores\_idle}} - P_{\text{glob\_1\_core\_idle}} &= (\hat{P}_{\text{GPP\_2\_cores\_idle}} + \hat{P}_{\text{Other\_2\_cores\_idle}}) \\ &\quad - (\hat{P}_{\text{GPP\_1\_core\_idle}} + \hat{P}_{\text{Other\_1\_core\_idle}}) \\ \hat{P}_{\text{GPP\_1\_core\_idle}} &\approx P_{\text{glob\_2\_cores\_idle}} - P_{\text{glob\_1\_core\_idle}} \qquad (3.8)\end{aligned}$$

where $P_{\text{glob\_2\_cores\_idle}}$ and $P_{\text{glob\_1\_core\_idle}}$ are the global power consumption of the platform in idle state when only two GPP cores and one GPP core are turned on, respectively. We suppose that $\hat{P}_{\text{Other\_1\_core\_idle}}$ and $\hat{P}_{\text{Other\_2\_cores\_idle}}$ are equivalent and thus the dif-

ference is rounded to zero. Also, the two GPP cores that are turned on are supposed to be identical and thus they consume the same amount of power each.

For instance, in the example presented in Fig.3.5, to calculate the power consumed by one GPP LITTLE core, one can calculate the global platform power when turning on only one GPP LITTLE core, $P_{\text{glob\_1\_core\_LITTLE\_idle}}$, i.e., the remaining three GPP LITTLE cores and four GPP big cores should be turned off. Then, calculate similarly the global platform power when turning on only two GPP LITTLE cores, $P_{\text{glob\_2\_cores\_LITTLE\_idle}}$, i.e., the remaining two GPP LITTLE cores and four GPP big cores should be turned off. Finally, apply Equation (7.11) to get the power consumed by one GPP LITTLE core.

Regarding $\hat{P}_{\text{Other}}$, it includes the power consumed by the HDIP. It also includes the IPC overhead power when performing HW video decoding. In addition, it includes the power related to memory, e.g., the transfers between Random Access Memory (RAM) and the HDIP internal buffer. $\hat{P}_{\text{Other}}$ includes a small amount of power needed to communicate with a host personal computer (PC) too, for instance, to receive video decoding commands. We consider that this amount of power is negligible.

$\hat{P}_{\text{Other\_idle}}$ ratio, $\hat{r}_{p\_other\_idle}$, is given by Equation (7.12). It is the proportion of the power consumption of all but GPP with respect to the global dynamic power, $P_{\text{glob\_dyn}}$.

$$\hat{r}_{p\_other\_idle} = \frac{\hat{P}_{\text{Other\_idle}}}{P_{\text{glob\_dyn}}} * 100 \tag{3.9}$$

Next, the effective amount of energy consumed for the video decoding process, $\hat{E}_{\text{glob\_active}}$, is given by Equation (3.11). To evaluate it, we first calculate the video decoding power consumption, $\hat{P}_{\text{glob\_active}}$.

$$\hat{P}_{\text{glob\_active}} = P_{\text{glob\_dyn}} - P_{\text{glob\_idle}} \tag{3.10}$$

Then, Equation (2.1) is applied to calculate the video decoding energy consumption.

$$\hat{E}_{\text{glob\_active}} = \sum(\hat{P}_{\text{glob\_active}} * \Delta t) \tag{3.11}$$

where $\hat{P}_{\text{glob\_active}}$ is the power consumption of video decoding at each sampling period of data-logging, and Δt is the duration of this period. Note that $\sum \Delta t$ is the time spent to decode a video sequence.

Finally, the ratio between the SW and HW video decoding energy, $\hat{r}_{\mathrm{sw/hw}}$, is calculated using the following equation:

$$\hat{r}_{\mathrm{sw/hw}} = \frac{\hat{E}_{\mathrm{glob\_active\_sw}}}{\hat{E}_{\mathrm{glob\_active\_hw}}} * 100 \tag{3.12}$$

where $\hat{E}_{\mathrm{glob\_active\_sw}}$ and $\hat{E}_{\mathrm{glob\_active\_hw}}$ are the energy consumption of the SW and HW video decoding, respectively.

Finally, Fig.3.6 illustrates the established power model in a graphical mode.

3.3 Performance and power consumption characterization methodology

In this section, we describe in detail the proposed characterization methodology to compare the energy consumption of the HW and SW video decoding. Our proposed characterization methodology intervenes at two hierarchical levels: (i) OS level, and (ii) application level. It is worth mentioning that the video decoding performance and power consumption characterization is done independently of the application design and its intricacies. This means that the study we conducted does not take into account the details related to video decoding modules, explained in Section 2.1.1 (ex: Entropy decoding, Motion compensation, etc.) [7]. Also, the decoded frames are not displayed [8].

3.3.1 Operating system level

The objective of this level is first to study the behaviors of the HW and SW video decoding power consumption by varying parameters triggered at the OS level. The second objective is to compare the energy consumption of these two approaches of video decoding. This helps to understand and assess the performance and energy consumption of video decoding at the application level. The measured video decoding power is given by Equation (3.2). The study at this level can be divided into the following three steps: idle state, HW video decoding, and SW video decoding. We first study the tested platforms in idle state,

7. Hence, our study does not allow us to say, for instance, that motion compensation is more optimized on the HDIP than on the GPP and so it consumes less power on the former. So, we cannot compare between the HW and SW video decoding power consumption of the video decoding modules.

8. The display system study is beyond the scope of this thesis.

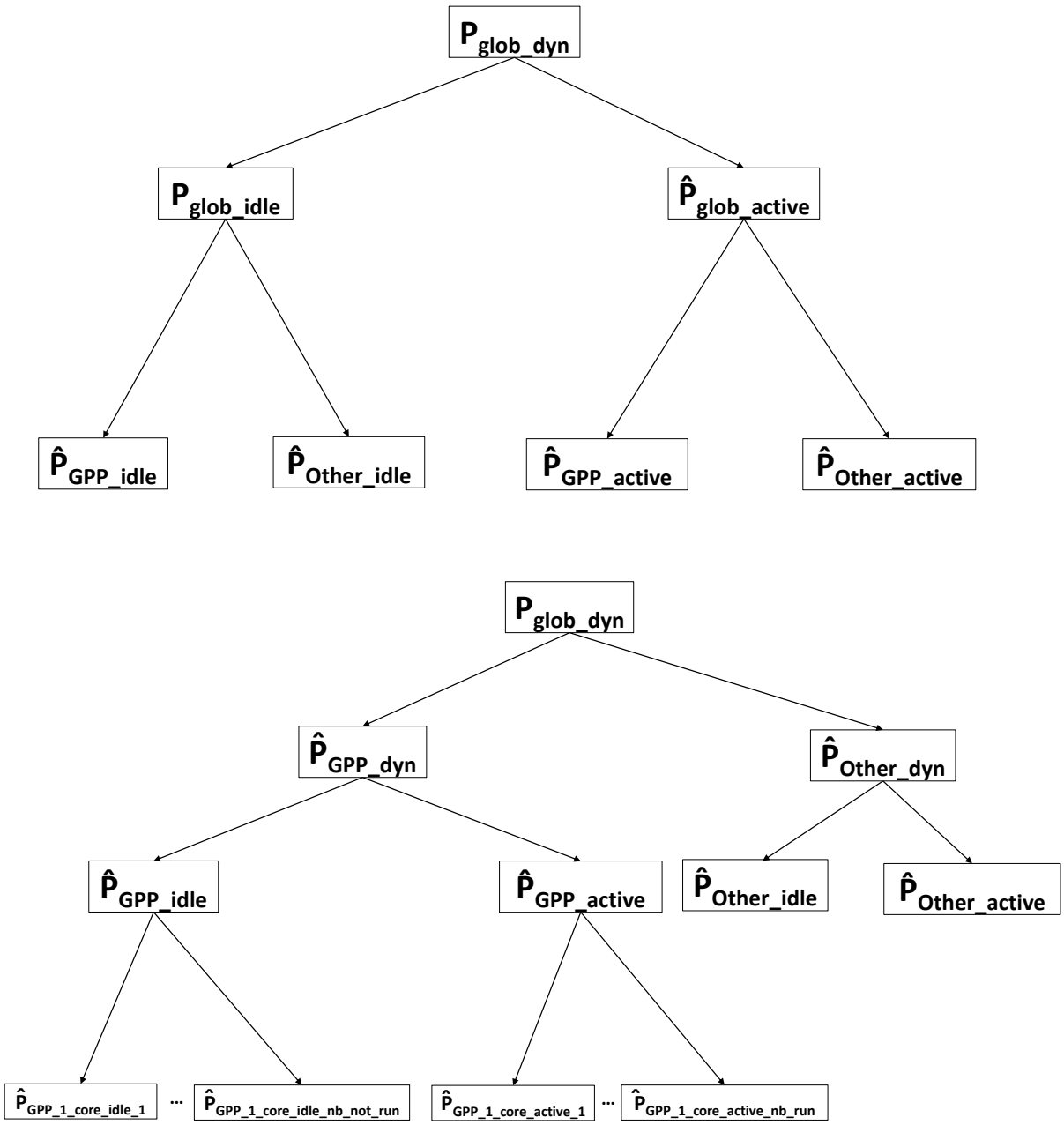


Figure 3.6 – Overall power model graph

i.e., independently of any running process. Then, we study the HW and SW video decoding power/energy consumption separately. Therefore, the first step is necessary to evaluate the second and third steps. Similarly, the second and third steps are, in turn, necessary to evaluate the HW and SW video decoding energy consumption at the application level.

Idle state

The objective of this step is to assess the power when the two video decoding platforms are in idle state. For that, $P_{\text{glob\_idle}}$ of Equation (3.2) is measured. This step is mandatory when the two video decoding approaches are run on two different platforms.

On the HW video decoding platform, only one GPP core is used. On this platform, we show how much the GPP core idle power, $\hat{P}_{\text{GPP\_1\_core\_idle}}$ of Equation (7.11), represents against that of the global platform, $P_{\text{glob\_idle}}$ of Equation (7.7), by comparing them. The GPP core frequency is fixed and clocked at the frequency that allows it to launch the target application (video decoding) [9]. Concerning the HDIP clock frequency, it is managed dynamically internally and is considered as a black box since we do not have access to its driver to handle it.

On the SW video decoding platform, all the GPP cores are used. On this platform, we show how much the GPP idle power, $\hat{P}_{\text{GPP\_idle}}$ of Equation (3.6), represents against that of the global platform, $P_{\text{glob\_idle}}$ of Equation (7.7), by comparing them. Then, we illustrate the impact of scaling down the GPP frequency and build an interval of power consumption values of the GPP in idle state. For that, the GPP is clocked at its lowest and then its highest supported frequency.

The measurements done in idle state are used to deduce the effective video decoding energy consumption, according to Equation (3.11).

HW video decoding

The aim of this step is threefold. First, we need to identify the periods where the GPP-HDIP units are in doze mode [10] and when they are performing the video decoding process. Here, $P_{\text{glob\_dyn}}$ is measured and compared to $P_{\text{glob\_idle}}$, according to Equation (3.2). For that, a set of video sequences are decoded to analyze the HW video decoding

9. Below this frequency threshold, the system will reach the application launch timeout and the frequency is automatically scaled up by the operating system (OS).

10. Doze mode: idle state concept used in Android OS

power consumption by zooming in one second of decoding time [11]. This time duration is chosen to evaluate the video bitrate and frame rate which are QoS metrics in video decoding applications, see Sections 2.1.1 and 2.1.1. Regarding the video resolution, it does not change during the video decoding execution. Second, we want to constitute a set of reference HW video decoding energy values. This set is used to evaluate the SW video decoding energy consumption, using Equation (7.15), by comparing them. Third, we investigate the $\hat{P}_{\text{Other}}$ ratio, using Equation (7.12), which includes the IPC overhead of the video decoding process.

SW video decoding

The goal of this step is to understand how the parallelism on a heterogeneous multi-core GPP architecture influences the video decoding energy consumption. This helps to find the optimal configuration (number of GPP cores, clock frequency) that satisfies the video decoding real-time constraint and diminishes the energy consumption.

Here, $\hat{P}_{\text{GPP\_active}}$ is deduced from Equation (3.2). Then, we investigate $\hat{E}_{\text{glob\_active}}$ which is calculated using Equation (3.11). This value is divided by the video sequence number of frames to obtain the average energy/frame. For that, a set of video sequences are decoded. The clock frequency at which the GPP cores are operating is varied. For simplicity, all GPP cores are clocked at the same frequency until the highest value supported by every core is reached. Beyond this frequency, the high-performance cores frequencies continue to be scaled until their highest supported value, whereas the remaining cores are clocked at their highest supported frequency.

Then, for each clock frequency, the number of GPP cores is also varied from one to the number of available cores by considering all combinations. Actually, we vary the number of idle and active cores as defined in Equation (3.6). For instance, in an ARM big.LITTLE SoC of eight GPP cores (4×big + 4×LITTLE), using five active cores gives four different combinations that can be evaluated, see Table 3.1.

Next, for each combination of the number of cores and frequency, only the one which consumes the minimum energy (energy/frame) is selected in order to compare it to the energy consumption of the HDIP. Finally, the average energy, decoding frame rate (*fps*), and GPP utilization are calculated as a function of the number of GPP cores and their clock frequencies.

11. The duration of 1 s is a snapshot. Both HW and SW video decoding are done for the entire video duration. Furthermore, before to start decoding, the platform is put to idle state for 10 s.

Table 3.1
Five heterogeneous GPP cores combinations

Combination	Number of big cores	Number of LITTLE cores
a	4	1
b	3	2
c	2	3
d	1	4

After that, the $\hat{P}_{\text{Other}}$ ratio is evaluated using Equation (7.12). $\hat{P}_{\text{Other}}$ corresponds to the remaining elements present in the SW video decoding platform. Finally, we compare the SW video decoding energy to that of the HW video decoding by calculating the ratio between them, using Equation (7.15).

3.3.2 Application level

The objective of this level is to explore how the video parameters (bitrate, frame rate, and resolution) impact the video decoding performance and energy consumption. For that, a set of video sequences encoded at different bitrates, frame rates, and resolutions are decoded. Then, the overall energy consumption, $\hat{E}_{\text{glob\_active}}$, is calculated using Equation (3.11). Next, this value is divided by the video sequence number of frames to obtain the average energy/frame. The video parameters may affect either the active power consumed, $\hat{P}_{\text{glob\_active}}$, or the decoding time, $\sum \Delta t$, according to Equation (3.11). Finally, we aim at finding the video decoding approach suited for each parameter value, e.g., which video decoding approach (HW or SW) is suited for a video resolution (720p, 1080p, etc.).

3.4 Summary

Finally, the proposed video decoding performance and power consumption characterization methodology is summarized in Table 3.2. Then, for better readability of the manuscript, all the notations used are summarized in Table 3.3.

Table 3.2
The proposed video decoding performance and power consumption characterization methodology summary

	HW video decoding	**SW video decoding**
Architecture	GPP+HDIP (referred to as GPP—HDIP)	GPP
Operating system level (studied parameters)	-	Number of GPP cores and GPP cores frequencies
Application level (studied parameters)	Bitrate, frame rate, and resolution	
Measured powers	$P_{\text{glob\_idle}}$ and $P_{\text{glob\_dyn}}$ of Equation (3.2)	
Calculated powers/energies	$\hat{P}_{\text{glob\_active}}$ of Equation (3.2) $\hat{P}_{\text{GPP\_dyn}}$ of Equation (3.6) $\hat{P}_{\text{GPP\_1\_core\_idle}}$ of Equation (7.11) $\hat{E}_{\text{glob\_active}}$ of Equation (3.11)	

Table 3.3
Notations used in the manuscript

Notation	Description	Measured or calculated
$P_{\text{glob\_dyn}}$	The global dynamic power of a platform	Measured
$P_{\text{glob\_idle}}$	The global idle power of a platform	Measured
$P_{\text{glob\_idle\_2\_cores}}$	The global power consumed by a platform when only 2 GPP cores are on	Measured
$P_{\text{glob\_idle\_1\_core}}$	The global power consumed by a platform when only 1 GPP core is on	Measured
$\hat{P}_{\text{glob\_active}}$	The global active power of a platform	Calculated
$\hat{P}_{\text{GPP\_idle}}$	The power consumed by GPP in idle state	Calculated
$\hat{P}_{\text{HDIP\_idle}}$	The power consumed by HDIP in idle state	Calculated
$\hat{P}_{\text{Other\_idle}}$	The power consumed by the remaining elements, in the platform, in idle state	Calculated
$\hat{P}_{\text{GPP\_active}}$	The power consumed by GPP in active state	Calculated
$\hat{P}_{\text{HDIP\_active}}$	The power consumed by HDIP in idle state	Calculated
$\hat{P}_{\text{Other\_active}}$	The power consumed by the remaining elements, in the platform, in active state	Calculated
$\hat{r}_{p\_other}$	The ratio of $\hat{P}$_Other with regards of the global power of a platform	Calculated
$\hat{E}_{\text{glob\_active}}$	The energy related to GPP–HDIP or GPP when running decoding process	Calculated
$\hat{r}_{\text{sw/hw}}$	The ratio between the SW (GPP) and HW (GPP–HDIP) video decoding energy consumption	Calculated
$\hat{E}_{\text{glob\_active\_hw}}$	The energy related to the active component in case of HW video decoding	Calculated
$\hat{E}_{\text{glob\_active\_sw}}$	The energy related to the active component in case of SW video decoding	Calculated
$\hat{P}_{\text{GPP\_HDIP\_idle}}$	The power consumed by GPP–HDIP in idle state	Calculated
$\hat{P}_{\text{Other\_idle\_2\_cores}}$	The power related to other components in a platform, in idle state, when only 2 GPP cores are on	Calculated
$\hat{P}_{\text{Other\_idle\_1\_core}}$	The power related to other components in a platform, in idle state, when only 1 GPP core is on	Calculated
$\hat{P}_{\text{HDIP\_idle\_2\_cores}}$	The power consumed by HDIP in idle state when only 2 GPP cores are on	Calculated
$\hat{P}_{\text{HDIP\_idle\_1\_core}}$	The power consumed by HDIP in idle state when only 1 GPP core is on	Calculated
D_period	The duration of period	-
Δt	The duration of data-logging period. Note that $\sum \Delta t$ is the time spent to decode an entire video sequence	-

3.5 Conclusion

In this chapter, our proposed methodology to solve the problem statement (RQ1 and RQ2) is described. First, the overall power model used to measure power and calculate the energy consumed when running video decoding is defined. In the proposed methodology, the power consumed by the video decoding process is extracted from that of the global platform. This is the power measured at the system level.

Second, the characterization methodology of the HW and SW video decoding performance and energy consumption for an objective analysis and comparison is detailed. For that, the impact of different parameters is studied. These parameters are triggered at two levels: (i) operating system, and (ii) application. For the former, the number of active GPP cores and their frequencies are varied, whereas for the latter, the video quality parameters: bitrate, frame rate, and resolution are varied. It should be noted that the architecture is the same throughout the experiments: an HDIP in addition to a GPP (referred to as GPP–HDIP), for the HW video decoding, and a heterogeneous multi-cores GPP for the SW one.

In the next chapter, the results obtained when applying the proposed methodology on the HEVC codec will be presented.

Chapter 4

Results & Analysis of HEVC decoding performance and energy consumption characterization

In this chapter, the obtained results of the HEVC decoding performance and energy consumption characterization are described and analyzed. These results correspond to answers to the research questions (RQ1 [1] and RQ2 [2]) described in the problem statement. First, the experimental HW and SW setups as well as datasets used to apply our proposed methodology and conduct our experiments are presented. Second, according to our proposed methodology described in chapter 3, the results are presented in two sections: (i) operating system level, and then (ii) application level. First, the operating system level is considered. The power consumed in the idle state is studied. Then, the power consumption of the HW video decoding is analyzed, followed by that of the SW video decoding. After that, the HW and SW video decoding are compared in terms of energy efficiency. Next, at the application level, the HW and SW video decoding are compared in terms of energy efficiency by studying the impact of video quality parameters: bitrate, frame rate, and resolution.

4.1 Experimental setup

In this section, all the environment setups used to conduct our experiments are described. First, the HW setups which include the description of the different platforms used in our experiments are presented. Then, the process to measure the power consumption when performing video decoding is explained. After that, the SW used to run video de-

1. RQ1: How relevant are the video decoders (HW and SW) with regards to the parameters impacting the video decoding process?
2. RQ2: To what extent do HW video decoders (HDIPs) outperform SW ones?

coding are stated. Finally, the video datasets used to validate the proposed methodology are given. A summary of the experimental setup is given in Table 7.2.

4.1.1 HW setup

In our work, the HEVC decoding performance and power consumption is characterized using three mobile heterogeneous multi-core platforms: Snapdragon 810 development board (APQ 8094) [129], Odroid-xu3 [130], and Qualcomm Robotics RB3 [131]. The first one was used to perform the HW video decoding and the second one for the SW video decoding. The third platform was used to perform the HW and SW video decoding.

Actually, at the beginning of our study, only Snapdragon 810 platform was available for experiments. Unfortunately, this platform did not allow to control the GPP frequency, which is necessary to perform the SW video decoding performance and energy consumption study, as explained in Section 3.3.1. For this reason, a different platform, which supports this feature (Odroid-xu3), was used. Then, in very late stages of this contribution (Contribution 1), a new platform (RB3) was made available for experiments. This latter implements both HW and SW video decoding. Therefore, RB3 platform was used only for results validation as presented in Section 4.1.1. This means that RB3 platform was used to show that the results of HEVC HW and SW decoding energy consumption comparison obtained from two different platforms (Snapdragon 810 and Odroid-xu3) are also valid when performing comparison on a single platform (RB3). Below, the HW and SW setups used in our experiments are described.

Snapdragon 810 development board

This platform is equipped with Snapdragon 810 ARM SoC made of 20-nm process technology. This SoC is featuring an ARM64-v8a octa-core big.LITTLE architecture (4× Cortex-A57 high performance cores and 4×Cortex-A53 power-efficient cores). Cortex-A57 and Cortex-A53 clusters operate at a range of frequencies of 384 MHz to 1.95 GHz and 384 MHz to 1.552 GHz, respectively. The Snapdragon 810 SoC integrates a 2160p (a.k.a. 4k) HEVC HDIP. Android debug bridge (adb) tool was used to communicate with this platform via a USB connection. As OS, Android 6.0 (Marshmallow, under Linux kernel 3.10.84), as the only supported OS on this platform is used. This platform was not used for SW video decoding because it does not allow to control the GPP clock frequency.

Figure 4.1 – Qualcomm Robotics RB3 overview, from [132]

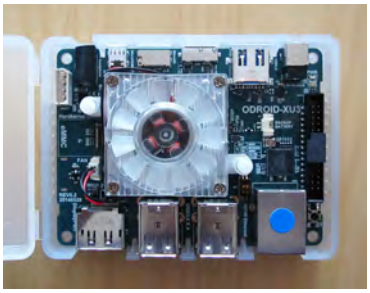


Figure 4.2 – Qualcomm Robotics RB3 overview, from [133]

Odroid-xu3

This platform has a smart-phone-like system architecture. It is based on the Samsung Exynos 5422 ARM SoC made of 32-nm process technology. Exynos 5422 is featuring an ARMhf octa-core big.LITTLE architecture (4× Cortex-A15 high performance out-of-order cores and 4× Cortex-A7 power-efficient in-order cores). Cortex-A15 and Cortex-A7 clusters operate at a range of frequencies of 200 MHz to 2.0 GHz and 200 MHz to 1.5 GHz, respectively. Secure shell (ssh) protocol was used to communicate with this platform via an Ethernet connection. As OS, Ubuntu 16.04 (Linux kernel 4.14.176+) distribution is used. The kernel was rebuilt to enable the userspace governor and global task scheduling (GTS). GTS permits to Exynos 5422 to utilize all eight cores simultaneously to manage computationally intensive tasks such as HEVC decoding. The userspace governor allows changing the GPP frequency on-the-fly from the user space. This platform was not used for HW video decoding because it does not integrate an HEVC HDIP.

Figure 4.3 – Qualcomm Robotics RB3 overview, from [134]

Qualcomm Robotics RB3 development platform

This platform is equipped with Qualcomm SDA845 SoC made of 10 nm process technology. This SoC is featuring a custom 64-bit ARM v8-compliant octa-core architecture (8 Qualcomm Kryo 385 CPU cores: 4× high performance cores and 4× power-efficient cores). The high performance and power-efficient cores operate at a range of frequencies of 825 MHz to 2.803 GHz and 300 MHz to 1.766 GHz, respectively. The SDA845 SoC integrates a 2160p@60fps (a.k.a. 4k) HEVC HW decoder. Ssh protocol was used to communicate with this platform via an Ethernet connection. As OS, Debian-based Linaro Linux 10.3 (Linux kernel 5.4.0) is used. The userspace governor was enabled.

Table 4.1 summarizes the above platforms capabilities in terms of supporting HEVC HW or SW decoding.

	Snapdragon 810	**Odroid-xu3**	**RB3**
HW	Yes	No	Yes
SW	No	Yes	Yes

Table 4.1 – HW and SW HEVC decoding support in the tested platforms

4.1.2 Power consumption evaluation

In our experiments, N6705A DC Power Analyzer, see Fig.4.4a, was used to measure power. The power measured by this tool is given by Equation (3.2). The power analyzer powers a device under test and samples the overall voltage and current simultaneously at 25 KHz. It is connected to the HW video decoding platform.

(a) N6701A DC Power Analyzer, from [135]

(b) Open-PEOPLE platform [136]

Figure 4.4 – Power measurement tools

In the case where the platform has on-board sensors, such as the SW video decoding platform (Odroid-xu3), Open-PEOPLE [136], see Fig.4.4b, is used as it measures $P_{\text{GPP\_idle}}$ and $P_{\text{GPP\_active}}$, from Equation (3.2), separately from the rest of the components power [3].

Then, to measure the power consumption of the video decoding process, first the power consumption data-logging is started, and then the system in idle state. After that, the video decoding application is launched. Finally, once the decoding is finished, the system goes back to idle state. The data-logging duration, T_{M} , is hence given by the following equation:

$$T_{\text{M}} = T_{\text{idle\_1}} + (T_{\text{L}} + T_{\text{D}}) + T_{\text{idle\_2}} \tag{4.1}$$

where T_{M} is the measurement time, $T_{\text{idle\_1}}$ is the time spent in idle state before video decoding, T_{L} is the application launching time, T_{D} is the decoding time, and $T_{\text{idle\_2}}$ is the time spent in idle state after video decoding. Note that T_{D} is calculated before doing the measurements and it includes only the decoding time, i.e., T_{L} is not included, and $T_{\text{idle\_2}}$ is obtained using dedicated graphical SW, e.g., matlab.

This method allows distinguishing the part which corresponds to the video decoding process from that corresponding to the system in idle state. Furthermore, the energy consumed while launching the application is eliminated.

3. $P_{\text{GPP\_idle}}$ and $P_{\text{GPP\_active}}$ are, in case of SW video decoding, without circumflex because they are directly measured by the Open-PEOPLE platform.

Table 4.2
Videos dataset characteristics

Parameter	Value
Resolution	720p, 1080p, 1600p, and 2160p
Frame rate	10, 15, 20, 25, 30, and 50 fps
Mode	Random Access
Profile	Main

4.1.3 SW setup

Concerning HEVC decoding, a simple Android application that leverages the HEVC HDIP available on the Snapdragon 810 platform was developed using Mediacodec application programming interface (API). On the other hand, Open-HEVC SW [137] was compiled on Odroid-xu3 and RB3 with NEON optimizations enabled. Ffmpeg SW [138] was used with v4l2 library to leverage the HEVC HDIP available on the RB3 platform.

The raw video sequences were encoded using ffmpeg SW. For analysis purposes, a bash script was developed for Odroid-xu3. In this script, data were read from /proc/pid/stat to retrieve and calculate the GPP utilization.

Finally, for accuracy purposes, the experiments were done three times and then averaged. Furthermore, the file cache (page cache, dentries, and inodes) was flushed at the beginning of each experiment in order to clean it from the temporary data used in previous experiments.

4.1.4 Videos dataset

In our work, two datasets were used. The first dataset was proposed by the joint collaborative team on video coding (JCT-VC) [139] as the reference common test sequences for HEVC. It contains a set of videos representing different scenarios and profiles. The second one represents some well-known video sequences, e.g., jellyfish [140] dataset and others [141]. The common characteristics of these video sequences are summed up in Table 4.2.

In this contribution, only the common parallelism scheme for all videos, which is frame-

Table 4.3
Experimental setup summary

	HW video decoding	**SW video decoding**
HW setup	Snapdragon 810: an octa-core big.LITTLE architecture (4× Cortex-A57 high performance cluster and 4×Cortex-A53 power-efficient cluster)	Odroid-xu3: and octa-core big.LITTLE architecture (4× Cortex-A15 high performance cluster and 4×Cortex-A7 power-efficient cluster)
	Qualcomm RB3: 8 Qualcomm Kryo 385 CPU cores (4x high performance cores and 4x power-efficient cores)	
Power consumption evaluation platform	N6705A Power Analyzer	Open-PEOPLE
OS	Android 6.0 (Marshmallow) with userspace governor enabled	Ubuntu 16.04 with userspace governor and GTS enabled
SW setup	On Snapdragon 810: an Android application that leverages HEVC HDIP using « Mediacodec » API	On both platforms (Odroid-xu3 and RB3): Open-HEVC with NEON optimizations enabled
	On Qualcomm RB3: ffmpeg (hevc_v4l2m2m)	
Videos dataset	JCT-VC, Jellyfish, and some well-known video sequences on the web. See their characteristics in Table 4.2	

by-frame, is studied. For other parallelism schemes, tiling and WPP, the decoding process depends on the encoding parameters.

Finally, the experimental setup summarized in Table 7.2.

In the next sections, the results obtained using the above described environment setups are presented.

4.2 Operating system level

At this level, idle power of the tested platforms is evaluated. Then, the behaviors of HEVC HW and SW decoding performance and power/energy consumption is studied. For that, *Kimono* video sequence is taken as an example to vary parameters at this level. Experiments were conducted on other video sequences as well and the curves show similar behaviors. Video sequences were decoded: (i) in HW using the GPP–HDIP (on the Snapdragon 810 platform), and (ii) in SW using a heterogeneous GPP (on the Odroid-

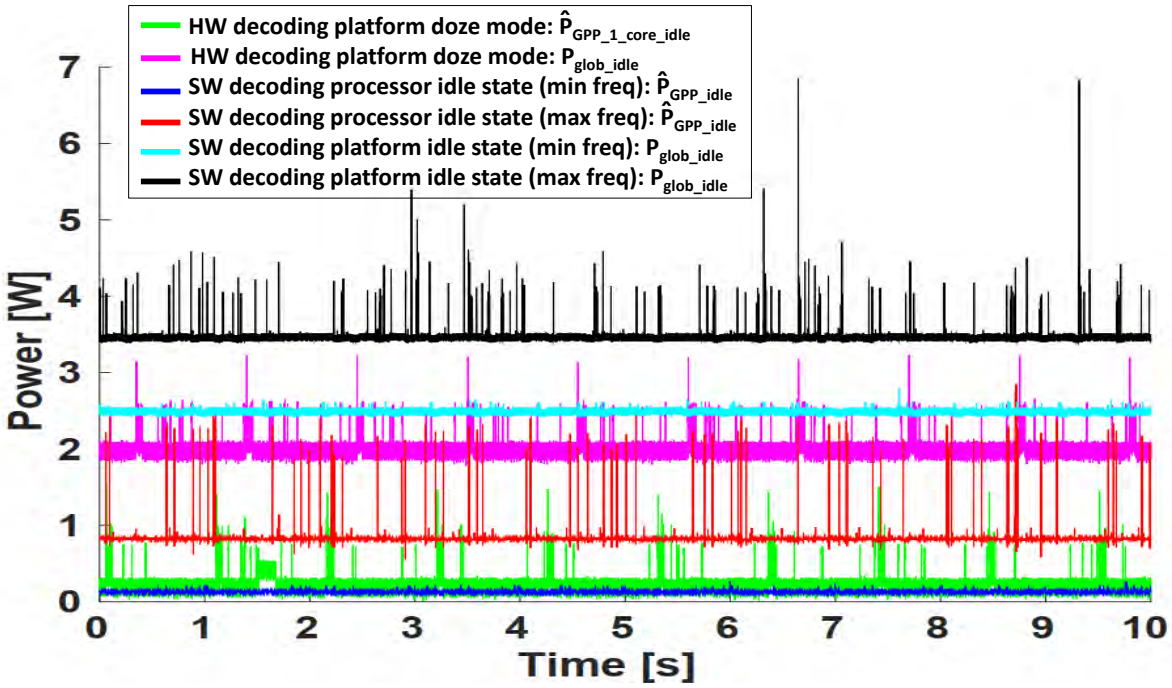


Figure 4.5 – HW video decoding platform (Snapdragon 810) versus SW video decoding platform (Odroid-xu3) idle power

xu3 platform) at the optimal configuration (number of cores, clock frequency) that offers the best trade-off between performance and energy consumption, as explained in Section 3.3.1.

4.2.1 Idle state

The first step of this level aims at studying the idle power of two tested platforms: (i) the HW video decoding platform (Snapdragon 810), and (ii) the SW video decoding platform (Odroid-xu3). For the former, the GPP (one core) idle power is presented and then compared to that of the global platform. Regarding the latter, the GPP (all cores) idle power is presented and then compared to that of the global platform. In addition, the GPP of the SW video decoding platform was clocked at its minimum and then its highest supported frequency. Finally, the idle powers of the two platforms are compared. These results are shown in Fig.4.5. This step was not applied on RB3 platform because, as mentioned in Section 3.3.1, this step is mandatory only when the two video decoding approaches are run on two different platforms and that RB3 platform supports both HW and SW HEVC decoding.

HW video decoding platform doze power consumption

On Android systems, if a user leaves a device unplugged and stationary for a period of time, with the screen off, the device enters doze mode. In the studied example, Snapdragon 810 consumes $P_{\text{glob\_idle}}$, on average 2.01 Watts, according to the measures. Periodically, the system exits doze mode for a short time to let the applications complete their deferred activities [142]. This is illustrated by the regular peaks in both HW video decoding platform doze power consumption curves in Fig.4.5.

Fig.4.5 additionally plots the Snapdragon 810 GPP one core power consumption at doze mode (without $\hat{P}_{\text{Other\_idle}}$). The GPP one core consumes $\hat{P}_{\text{GPP\_1\_core\_idle}}$, on average 0.22 Watts, according to Equation (7.11). Compared to the global platform doze power, one can observe that the $\hat{P}_{\text{Other\_idle}}$ component, which is the difference between $P_{\text{glob\_idle}}$ (pink curve) and $\hat{P}_{\text{GPP\_1\_core\_idle}}$ (green curve) in Fig.4.5, constitutes a huge part of the global platform power consumption. Indeed, it represents about 88%.

SW video decoding platform idle power consumption

On Linux-like systems, the idle state is managed by wait for interrupt (WFI) ARM instruction. WFI disables most of the clocks of the GPP. This corresponds to a low-power state of the processor. In this state, the SW video decoding platform consumes $P_{\text{glob\_idle}}$, according to Equation (3.2). On the tested platform (Odroid-xu3 platform in our case), $P_{\text{glob\_idle}}$ represents on average 2.49 Watts and 3.46 Watts for the lowest and highest supported GPP clock frequency, respectively. It should be noted that the WFI power consumption is different from the GPP static power [143].

Fig.4.5 additionally plots the Odroid-xu3 GPP power consumption in idle state (without $\hat{P}_{\text{Other\_idle}}$). The GPP consumes $\hat{P}_{\text{GPP\_idle}}$, on average 0.13 Watts and 0.82 Watts for the lowest and highest supported GPP clock frequency, respectively, according to Equation (3.6). Compared to the global idle power, one can observe that the $\hat{P}_{\text{Other\_idle}}$ component, which is the difference between $P_{\text{glob\_idle}}$ and $\hat{P}_{\text{GPP\_idle}}$ in Fig.4.5, constitutes a huge part of the global power consumption. Actually, it is about 94% and 76% for the lowest and highest supported GPP clock frequency, respectively. The percentages are not the same because when scaling the GPP clock frequency from the lowest to the highest supported values, the other components clock frequencies are not scaled accordingly.

As explained in Section 2.1.3, idling at low clock frequency allows a GPP, $\hat{P}_{\text{GPP\_idle}}$, to consume significantly less power. For instance, on the SW video decoding platform

(Odroid-xu3), clocking the GPP cores at their minimum frequency, by taking into account $\hat{P}_{\text{GPP\_idle}}$, would allow to reduce the power by approximately 84% of power compared to the state where the cores are clocked at their highest supported frequency, see Fig.4.5 (blue and red curves). This is because, in idle state, the clock tree system is still consuming power. Therefore, it should be scaled at low frequency to save power.

Summary: In case of HW video decoding platform (Snapdragon 810), $\hat{P}_{\text{GPP\_1\_core\_idle}}$ represents on average 0.22 Watts, according to Equation (7.11), whereas $P_{\text{glob\_idle}}$ represents on average 2.01 Watts, according to Equation (3.2). Therefore, $\hat{P}_{\text{Other\_idle}}$ constitutes on average 88% of the global platform idle power.

On the SW video decoding platform (Odroid-xu3), according to Equation (3.6), the GPP idle power values interval is on average between 0.13 Watts and 0.82 Watts. The interval boundaries correspond to the power at the lowest and highest supported GPP clock frequency, respectively. Thus, at the GPP level point of view, around 84% of power can be saved by scaling down the GPP frequency from the highest to the lowest value. On the other hand, $\hat{P}_{\text{Other\_idle}}$ constitutes about 94% and 76% of the global platform power for the lowest and highest supported GPP clock frequency, respectively.

4.2.2 HW video decoding

In the second step of the study at the OS level, the HEVC HW decoding power consumption using the GPP–HDIP on the Snapdragon 810 platform is described. $P_{\text{glob\_dyn}}$ and $P_{\text{glob\_idle}}$ values of Equation (3.2) are investigated. Then, the impact of $\hat{P}_{\text{Other}}$, which includes the IPC between the HDIP and other elements participating to the video decoding process, on the power consumption, using Equation (7.12) is assessed. For that, *Kimono* video test sequence, which possesses the characteristics given in Table 4.2 (resolution is 1080p), was decoded as an example. The video decoding process was done in real-time. Other tested video sequences showed similar behaviors.

Fig.4.6 shows the complete data-logging of the HW video decoding power consumption. It includes the different parts of Equation (4.1). It can be noted that the launching time was not negligible in case of the Android platform.

Fig.4.7 depicts the HW video decoding power consumption, $P_{\text{glob\_dyn}}$, compared to doze power, $P_{\text{glob\_idle}}$, using Equation (3.2), during one second. Given that the video example (Kimono) duration is 4 seconds, the position of the snapshot shown in Fig.4.7 is from 1 to 2 seconds of the video. Fig.4.8 depicts a zoom on decoding two frames to show the different video decoding phases. The HW video decoding ($P_{\text{glob\_dyn}}$) consumes, in this

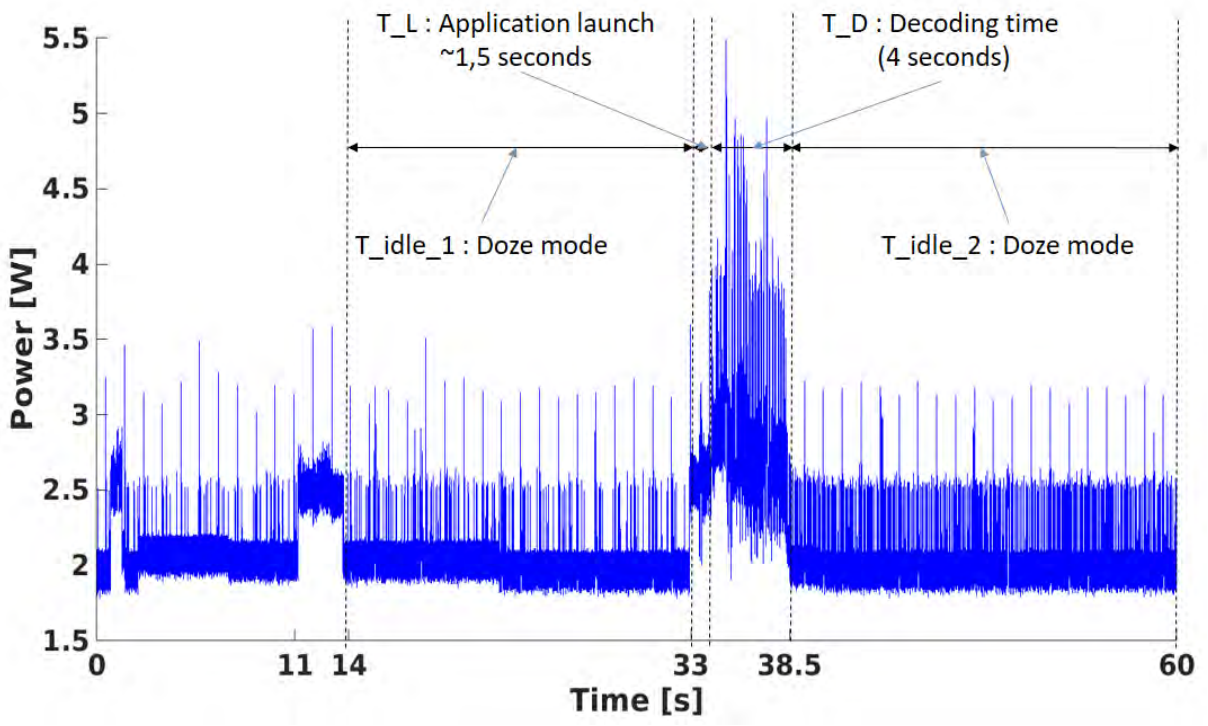


Figure 4.6 – HW video decoding power consumption complete data-logging (platform: Snapdragon 810, video sequence: Kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

example, on average 2.75 Watts while the doze power ($P_{\text{glob\_idle}}$) represents on average 2.01 Watts.

In Fig.4.7, one can observe that the HDIP enters doze mode whenever it finishes decoding before the end of the video decoding period, see Section 2.1.4, in order to save power. The presence and the duration of doze mode after the frame decoding depend on the frame complexity. For instance, in the example shown in Fig.4.8, two doze intervals of 7.5% and 2.5% of the video decoding period, respectively, can be observed. The ARM wake-up is represented by the power level transition after the doze mode level. The GPP then sends the parameters (next frame to decode) to the HDIP and triggers a HW video decoding function.

The frame processing period is composed of two intervals: (i) the decoding of the frame and (ii) doze mode whenever the decoding is finished before the next period starts. Fig.4.8 shows an example where the HDIP finished decoding a frame before the end of the frame processing period. This is due to the high performance of the HDIP which is designed to decode 2160p (a.k.a. 4K) video content. Moreover, One can observe that, at doze mode,

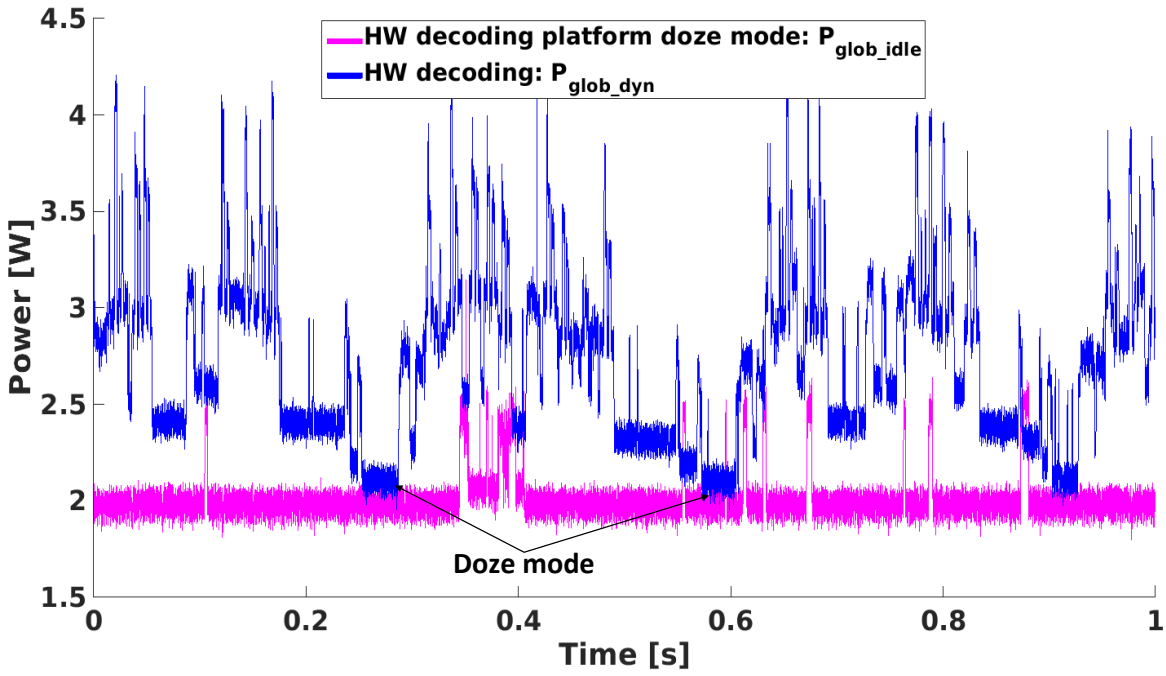


Figure 4.7 – Zoom on 1 second of HW video decoding versus doze mode power consumption (platform: Snapdragon 810, video sequence: Kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

the HDIP is still consuming power. This is because the HDIP is not shut off when it is waiting for the next frame to decode from the GPP. The doze mode interval, when it is present, is so short that entering in a deeper sleep mode would have hurt the performance [104].

The frame decoding process is done by the HDIP which is handled, by the GPP, as an I/O operation. This brings an overhead due to GPP–HDIP communication, e.g., HW interrupt handling, and the communication with other processing elements involved in the video decoding process. The overhead leads to extra performance and energy costs. Indeed, according to Equation (7.12), $\hat{P}_{\text{Other}}$ represents more than 75% of the global power. Therefore, the GPP–HDIP consumes, in this study example, about less than 25% of the HW video decoding global power consumption.

Summary: To sum up, as the HDIP (on the Snapdragon 810 platform) is designed for high data rate processing, in order to save power, it spends a proportion of time at doze mode whenever it finishes decoding a frame before the next video decoding period starts. The presence and the duration of doze mode after the frame decoding depend on the frame complexity. On average, it represents 5.8% of the video decoding period over all the

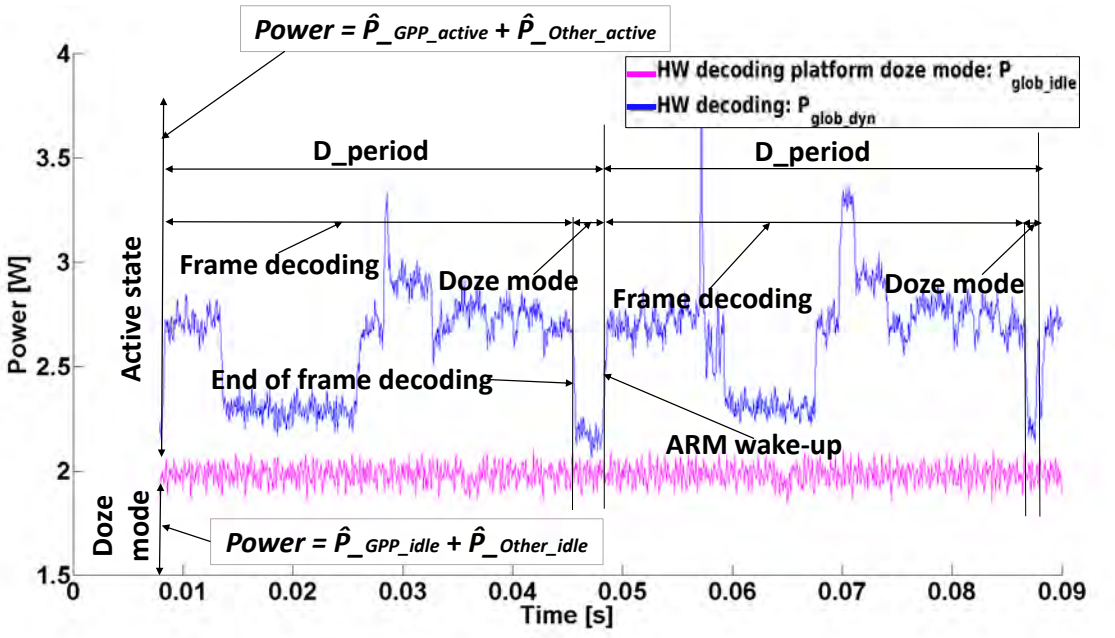


Figure 4.8 – Zoom on 2 frames HW video decoding versus doze mode power consumption (platform: Snapdragon 810, video sequence: Kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

decoded frames. Nevertheless, $\hat{P}_{\text{Other}}$ constitutes an enormous power consumption part of the HW video decoding global power, on average more than 70%. This part includes the IPC between the HDIP and other processing elements involved in video decoding, such as memory. This decreases substantially the energy efficiency of the HW video decoding. Indeed, on average less than 30% of the global power is consumed by the GPP–HDIP.

4.2.3 SW video decoding

In the third step of the study at the OS level, the HEVC SW decoding energy consumption using a heterogeneous GPP on the Odroid-xu3 platform is analyzed. First, $P_{\text{GPP\_active}}$ as a particular case of $\hat{E}_{\text{glob\_active}}$, which is given by Equation (3.11), see Footnote 3 (P.76), is investigated by varying the number of GPP cores and their operating frequencies. Then, the impact of $\hat{P}_{\text{Other}}$, which corresponds to the power consumed related to the remaining elements present in the platform, such as memory, on the global platform power consumption is assessed using Equation (7.12). Finally, the SW video decoding energy consumption is compared to that of the HW video decoding by calculating the

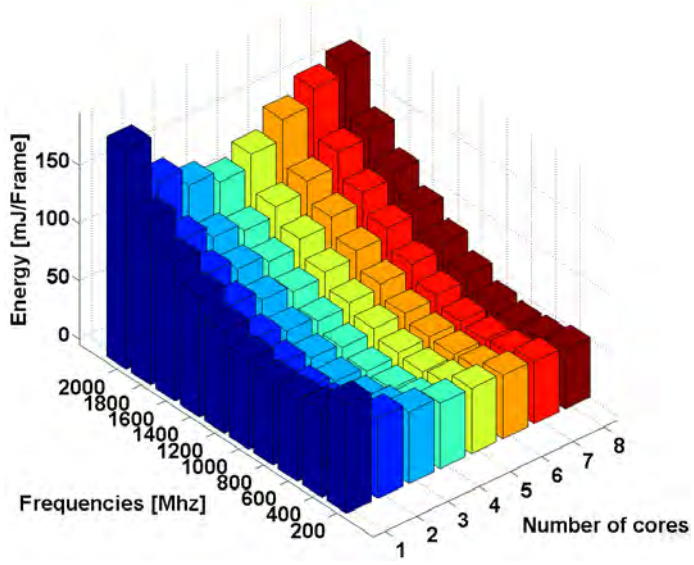


Figure 4.9 – Energy as a function of the number of active cores and the processing frequencies (platform: Odroid-xu3, video sequence: kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

ratio between them, $\hat{r}_{\mathrm{sw/hw}}$, using Equation (7.15). For that, *Kimono* video test sequence, which possesses the characteristics given in Table 4.2 (resolution is 1080p), was decoded as an example. The decoding process was done in real-time, so, the fps was bounded to the frame rate at which the video is encoded (25 Hz). The video sequence frames are decoded in parallel, thanks to the frame-by-frame parallelism scheme. The experiments were conducted on other video sequences as well and they all showed similar behaviors.

Fig.4.9, Fig.4.10, and Fig.4.11 show the energy, the fps, and the GPP utilization behaviors with respect to the number of processing cores and their operating frequencies, respectively. The energy is calculated using Equation (3.11) and is given in mJ/Frame. At first sight, one may notice that the energy does not always increase as the number of cores and their clock frequencies increase, as explained in this section.

GPP energy consumption vs number of cores

The curve, in Fig.4.9, revealed that the overall HEVC decoding energy consumption changes drastically as the number of GPP cores and their frequencies change. In fact, whatever the GPP clock frequency is, decoding with only one core (mono-threading) consumes the most energy (mJ/Frame) since the GPP usage percentage is more than 90%, see Fig.4.11. Then, the more GPP cores added, the less they are occupied and the

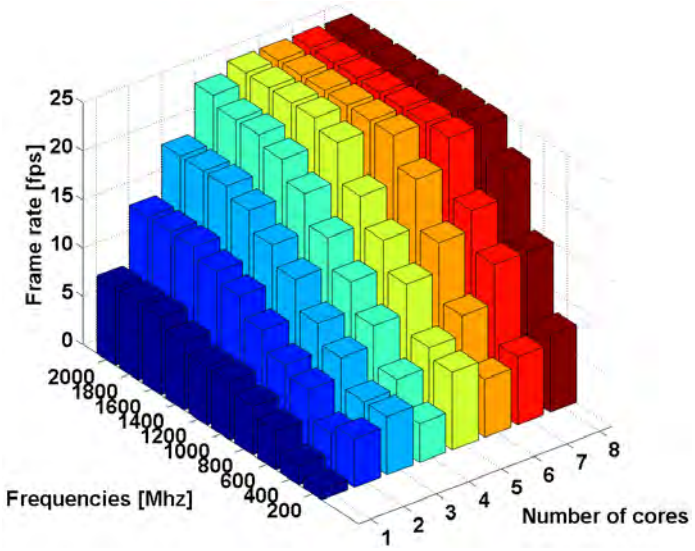


Figure 4.10 – Frame rate as a function of the number of active cores and the processing frequencies (platform: Odroid-xu3, video sequence: kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

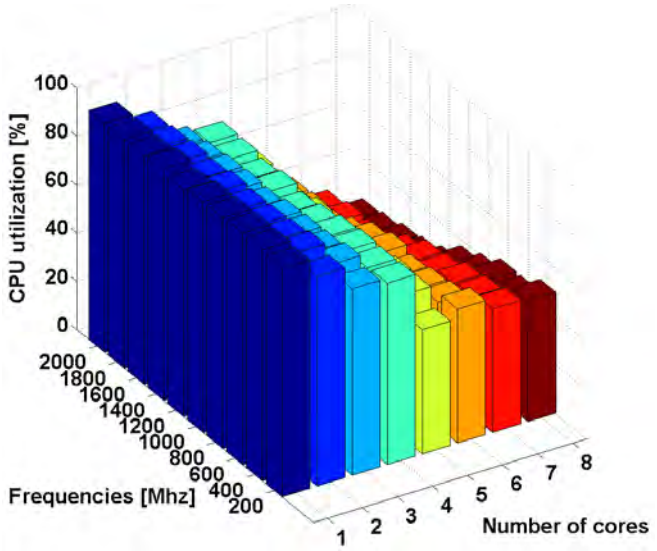


Figure 4.11 – GPP utilization as a function of the number of active cores and the processing frequencies (platform: Odroid-xu3, video sequence: kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

less energy (mJ/Frame) the video decoding process consumes. This is the case because the load is balanced among the GPP cores and so they have less load to deal with. In addition, the cores that are not involved in the video decoding process are put into idle state and thus consume a few amount of energy.

GPP performance vs number of cores

In the presented results, following our methodology described in Section 3.3.1, from one to four cores, only LITTLE cores configurations are selected. None of these configurations allows HEVC real-time decoding. One should add more GPP cores to satisfy this constraint. At this point, all configurations from five GPP cores at 1.2 GHz and above can perform HEVC real-time decoding, see Fig.4.10. It can be noticed that the configurations with two, three, and four big cores were able to reach the real-time capability. However, they were not selected because they consume a huge amount of energy (mJ/Frame) compared to the best configuration that will be discussed later.

GPP heterogeneity impact on energy consumption

Different five GPP cores configurations (in a heterogeneous big.LITTLE architecture), clocked at 1.2 GHz, gave different energy consumption results. Using four big cores and one LITTLE core would result in real-time video decoding but consume a huge amount of energy (mJ/Frame). Then, every time one big core is taken off and replaced with one LITTLE core, the energy consumption (mJ/Frame) decreases, see Fig.4.12. Therefore, the setup that allows to reach the real-time video decoding while consuming the least amount of energy (mJ/Frame) is the configuration (d) in Table 3.1: using one big core and four LITTLE cores at 1.2 GHz. This configuration is the optimal one, offering the best trade-off between performance and energy consumption (mJ/Frame). It would save, by taking into account $P_{\text{GPP\_active}}$, see Footnote 3 (P.76), around 29% of energy (mJ/Frame) compared to the first configuration (a).

This gain is not so high because of the GPP cores heterogeneity. One source of energy inefficiency appears when a big core finishes decoding a frame and waits a long time before a LITTLE core finishes its task. This creates a substantial overhead due to the inter-communication between heterogeneous GPP cores.

It should be recalled that energy is given by Equation (3.11). Using one to four GPP cores, the fps increases as the number of active cores is increased, see Fig.4.10 (one to four cores). So, the video decoding time decreases. On the other hand, as it is stated in Section 2.1.3, the parallelism decreases the consumed energy. This explains the decrease in the energy consumption (mJ/Frame) when decoding with one to four LITTLE cores. By adding more GPP cores, five to eight cores, the energy consumption (mJ/Frame) increases as the number of cores increases, see Fig.4.9. There are two things which can

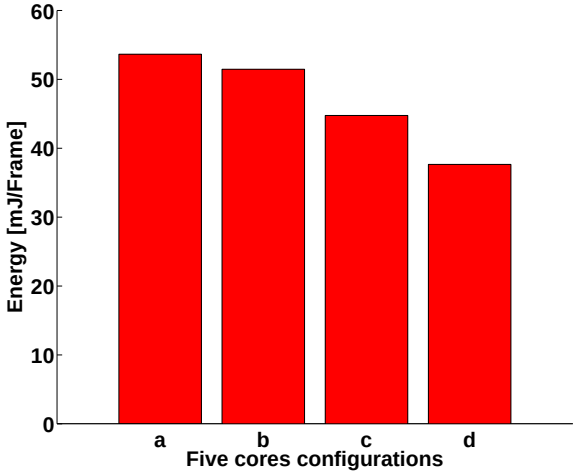


Figure 4.12 – Five cores configurations energy consumption (configurations a-d refer to those defined in Table 3.1)

explain that increase. First, the video decoding process time is constant since the real-time video decoding is reached using five cores. Second, the power is growing higher every time a big core is added. This results in an increase of the overall energy consumption. As a consequence, increasing the number of GPP cores does not always increase the energy efficiency.

$\hat{P}_{\mathbf{Other}}$ overhead

Furthermore, the frame decoding process is done by GPP which is situated in a SoC. The SoC, in turn, is connected to other elements in the device to work. This brings an overhead due to the GPP communication with other elements involved in video decoding. The overhead leads to extra performance and energy costs. Indeed, $\hat{P}_{\mathrm{Other\_idle}}$ represents, in this study example, on average 2.33 Watts, deduced from Equation (7.7), while the SW video decoding global power represents on average 3.91 Watts, according to Equation (3.2). So, $\hat{P}_{\mathrm{Other}}$ represents more than 59% of the global power. Therefore, the SW video decoding GPP power represents, in this study example, on average less than 41% of the SW video decoding global power consumption.

Summary: For energy-efficient SW video decoding design, under real-time constraints and the GPP cores heterogeneity, the parallelism should be used with precaution. In the presented example, the configuration which gives the best trade-off between performance

and energy consumption (mJ/Frame) is using five GPP cores, four LITTLE cores and one big core, at 1.2 GHz. Moreover, decoding at this configuration would save, at the GPP level point of view, around 29% of energy (mJ/Frame) compared to the configuration with four big cores and one LITTLE core.

Moreover, the results showed that the most energy-efficient point to do HEVC SW real-time decoding is none of the standard setting points: {freq_min, freq_max, nb_cores_min, nb_cores_max}. Furthermore, our experiments revealed that minimizing the processing frequency is not the most energy-efficient strategy because the execution time is increased. These results are also reported in [114]. Finally, to do SW video decoding, the GPP consumes effectively on average less than 50% of the SW video decoding global platform power consumption. This overhead includes the inter processor communication between the GPP and other components involved in video decoding, such as memory.

4.2.4 HEVC HW vs SW decoding energy consumption

The objective of the last step of the OS level is to highlight the impact of the parallelism on the SW video decoding energy consumption. It consists in summing up the results by comparing the HEVC SW to HW decoding in terms of energy efficiency.

Table 4.4 presents the ratio of the energy consumption between HEVC SW decoding, using different GPP configurations (one to five GPP cores) on the big.LITTLE heterogeneous architecture and HEVC HW decoding, using GPP–HDIP. For HEVC SW decoding, among all configurations, only those which ensure the real-time video decoding constraint are selected, except the first configuration (1 GPP core) which cannot, in any case, satisfy this constraint. Furthermore, the configurations six to eight GPP cores are excluded as the optimal configuration is reached using five GPP cores, as explained before.

The results in Table 4.4 show that varying the number of cores can save the energy consumption considerably. Using one GPP core not only consumes the largest amount of energy (a ratio of more than 5×) but also does not satisfy the real-time constraint. Then, when two big cores are decoding, the video was decoded without any missed frame, i.e., in real-time. However, it still drains the battery. Adding more big cores (using three or four cores) does not change drastically the energy consumption since the fps is bounded at the frame rate at which the video sequence is encoded (25 Hz) and is reached in the last setup (two big cores). Finally, the configuration with five cores offers the best trade-off between performance and energy efficiency. It actually diminishes the SW video decoding energy consumption and gets it the closest to that of the HW video decoding (a ratio of

Table 4.4
HEVC HW versus SW video decoding energy consumption (platforms: Snapdragon 810 (HW) and Odroid-xu3 (SW), video sequence: kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

Decoding	**SW (#big : #LITTLE)**					**HW**
Config	1 GPP core (0:1)	2 GPP cores (2:0)	3 GPP cores (3:0)	4 GPP cores (4:0)	5 GPP cores (1:4)	GPP–HDIP
$\hat{r}_{\text{sw/hw}}$	5.38	4.93	4.93	4.88	1.66	1

less than 2×). Beyond five cores, the energy (mJ/Frame) increases as explained earlier in this section.

4.3 Application level

At this level, the influence of the video bitrate, frame rate, and resolution on the HEVC decoding energy consumption executed on mobile platforms is evaluated. The video decoding consumed energy is calculated using Equation (3.11) and is given in mJ/Frame. *Kimono* video sequence is taken as an example to vary bitrate (at 1080p resolution), frame rate (at 1080p resolution), and resolution. Experiments were conducted on other video sequences as well and the curves show similar behaviors. Video sequences were decoded: (i) in HW using the GPP–HDIP (on the Snapdragon 810 platform), and (ii) in SW using a heterogeneous GPP (on the Odroid-xu3 platform) at the optimal configuration (number of cores, clock frequency) that offers the best trade-off between performance and energy consumption, as explained in Section 3.3.1.

4.3.1 Varying video bitrate

Fig.4.13 depicts the impact of the video bitrate parameter on the HW and SW video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (3.11).

The energy consumption (mJ/Frame) is almost steady in case of HW video decoding. Indeed, increasing the bitrate does not change a lot the energy consumption. The HDIP always tries to supply the necessary resources to avoid the energy over-consumption.

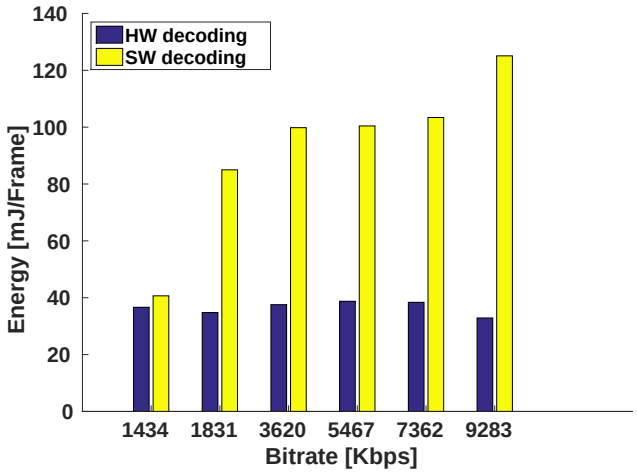


Figure 4.13 – HEVC decoding energy consumption when varying video bitrate (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, frame rate: 25 Hz, resolution: 1080p, platforms: Snapdragon 810 and Odroid-xu3)

In case of SW video decoding, the consumed energy (mJ/Frame) increases as the video bitrate grows high. By definition, as mentioned in Section 2.1.1, the bitrate is the amount of data required to decode one second of video. So, the higher the bitrate, the higher the video visual quality and thus the more bandwidth it requires. Therefore, this leads to more data volume to process and then more GPP usage. For instance, in Fig.4.13, decoding the 9283 Kbps video sequence would increase the energy consumption (mJ/Frame) on average by 3.08× compared to the same video sequence encoded at 1434 Kbps.

The SW video decoding energy consumption does not evolve linearly when varying video bitrate. The reason comes from the fact that every time the video bitrate is increased, the number of GPP cores also need to be increased and their frequencies accordingly. In addition, as explained in Section 4.2.3, the energy consumption of SW video decoding using a heterogeneous GPP does not scale linearly as the number of cores and their frequencies grow high. Therefore, there is not a linear relationship between the video bitrate and the energy consumption in case of SW video decoding.

4.3.2 Varying video frame rate

Fig.4.14 depicts the impact of the video frame rate parameter on the HW and SW video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (3.11).

First of all, when changing the video frame rate, the resulting video bitrate changes

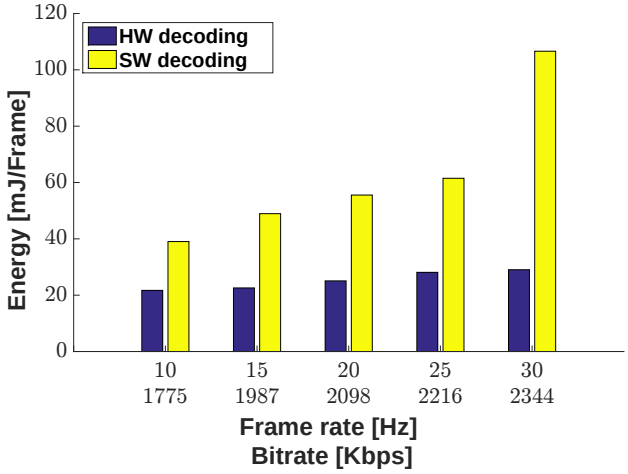


Figure 4.14 – HEVC decoding energy consumption when varying video frame rate (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, resolution: 1080p, platforms: Snapdragon 810 and Odroid-xu3)

accordingly. The video bitrate evolves with the frame rate. That is, increasing the number of frames per second implies increasing the number of bits per second.

In case of HW video decoding, the energy consumption (mJ/Frame) increases every time the video frame rate is increased. The reason is that the higher the frame rate, the more memory transfers are needed in order to copy encoded and decoded frames from and to memory. And so, the more I/O operations are needed. Therefore, more consumed energy (mJ/Frame) is observed.

In the presented example, the increase is very slight. This shows that the HDIP is not scalable, i.e., the HEVC HW decoding energy consumption (mJ/Frame) does not increase remarkably with the video frame rate. Indeed, in the study example shown in Fig.4.14, the video sequence encoded at 30 Hz would need on average 1.34× more energy (mJ/Frame) to be decoded compared to this same video sequence encoded at 10 Hz. This confirms that the HDIP is not scalable for low data rate.

In case of SW video decoding, the energy (mJ/Frame) increases as the video frame rate grows high. This increase can be explained by the high data rate resulting when the video frame rate is increased. Actually, increasing the video frame rate means that the GPP should do more job in the same unit of time in order to satisfy the real-time decoding constraint, i.e., the number of frames decoded per second. In addition, to do more job, the GPP number of cores and their frequencies should be scaled accordingly. Therefore,

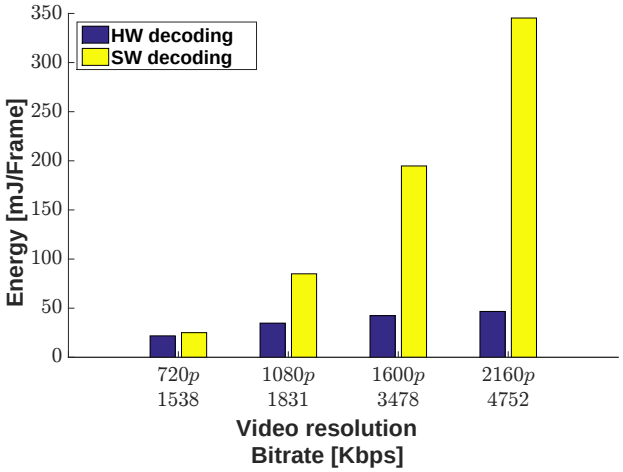


Figure 4.15 – HEVC decoding energy consumption when varying video resolution (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, frame rate: 25 Hz, platforms: Snapdragon 810 and Odroid-xu3)

Table 4.5
Video resolution scaling factors with respect to 720p

Resolution	720p	1080p	1600p	2160p
Scaling factor	1	2.25	4.44	9

the GPP consumes more power. For instance, in Fig.4.14, decoding the video sequence encoded at 30 Hz would increase the energy consumption (mJ/Frame) on average by 2.72× compared to that encoded at 10 Hz. The more the frame rate is increased, the more energy (mJ/Frame) is needed for GPP to decode a video sequence.

4.3.3 Varying video resolution

Fig.4.15 depicts the impact of the video resolution parameter on the HW and SW video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (3.11). The video resolution varies from 720p to 2160p. Table 4.5 summarizes the scaling factors of video resolutions with respect to the smallest one (720p).

In case of HW video decoding, scaling the video resolution from 720p to 2160p through 1080p and 1600p would consume only 1.43×, 1.94× and 2.14× more energy (mJ/Frame),

Table 4.6
HW video decoding power consumption percentage, $(100 - \hat{r}_{p\_other})$ using Equation (7.12)

Resolution	720p	1080p	1600p	2160p
HW video decoding (%)	23	24	33	41

respectively. Compared to the video resolution scaling factors shown in Table 4.5, the HEVC HW decoding energy consumption (mJ/Frame) does not scale as the video resolution does. This is because the HDIP, in the tested platform (Snapdragon 810), is designed for 2160p (a.k.a. 4K) video content. This shows that, the HDIP did not seem to implement mechanisms that adapt the energy consumption when decoding low resolution videos.

On the other hand, our experiments revealed that the video resolution impacts substantially the effective power consumed by the GPP–HDIP when performing the HW video decoding process. While this power represents on average less than 24% of the global power consumption for 1080p resolution and lower, it is getting more significant when increasing the video resolution. Table 4.6 sums up the GPP–HDIP HW video decoding power consumption percentage with respect to the global platform one as a function of video resolution.

Clearly, for the HW video decoding, the HDIP is more suited for high video resolutions in terms of power efficiency. As the HDIP of the tested platform (Snapdragon 810) is designed for 2160p video content, when processing less data, e.g., 720p content, there are fewer resources (memory size, number of parallel thread cores, etc) involved in the video decoding process. Furthermore, these unused resources are still consuming power, which leads to substantial energy overhead.

Things look different when it comes to the SW video decoding. As expected, the energy consumption (mJ/Frame) of a mobile device depends heavily on the video resolution. However, the scaling is not always linear. For instance, playing back a 1080p video sequence would consume on average 1.64× more energy (mJ/Frame) than a 720p video sequence. Above this resolution, the ratio increases drastically: 7.76× and 13.76× for 1600p and 2160p resolutions, respectively.

Table 7.5 depicts the ratio between the HEVC SW and HW decoding energy consumption, calculated using Equation (7.15), as a function of the video resolution. Compared to the state-of-the-art study [22] where it is stated that dedicated processors outperform

Table 4.7
Energy consumption ratio as a function of video resolution, see Equation (7.15), P.149, on Snapdragon 810 and Odroid-xu3

Resolution	720p	1080p	1600p	2160p
$\hat{r}_{\mathbf{sw/hw}}$	1.15	2.18	4.59	7.4

the GPPs by around 1000× in terms of energy efficiency, for low video resolutions (720p and 1080p), and from the system point of view [4], the gap is not as high as that of the state-of-the-art work (a ratio of less than 3×). However, for high video resolutions, the GPP–HDIP offer the best performance in terms of energy efficiency (a ratio of more than 7×).

Summary: As a conclusion, the video resolution impacts differently the HW and SW video decoding (on the Snapdragon 810 and Odroid-xu3 platforms, respectively). Scaling the video resolution from 720p to 2160p through 1080p and 1600p would increase the energy by 1.43×, 1.94× and 2.14× in case of the former and 1.64×, 7.76× and 13.76× in case of the latter. Therefore, the HW video decoding is more suited for high video resolution in terms of energy efficiency.

Then, bitrate and frame rate metrics represent a strong constraint for the video decoding process as it should be respected in a unit of time, e.g., bits per second, to guarantee a smooth video play-back. Therefore, the required processing resources should all be involved in order to satisfy this constraint even at the cost of energy over-consumption. This means that the OS should give the video decoding process a high priority to utilize the GPP and all required resources: memory (cache, DRAM, and HDD), buses, etc.

Next, the video frame rate and resolution parameters are highly correlated with the video bitrate. For instance, when the resolution of a given video sequence is changed, to maintain its frame rate, the bitrate should be scaled accordingly. The higher the video resolution, the higher the bitrate. Similarly, the higher the frame rate, the higher the bitrate. Therefore, Fig.4.13, Fig.4.14, and Fig.4.15 show similar behaviors.

Furthermore, to conclude, on the tested platforms, the frame rate parameter has a similar impact on the video decoding power consumption as that of the bitrate and resolution parameters. This allows, for instance, to interchange these parameters when one

4. System point of view: all the resources required to perform the video decoding process are considered.

wants to model the energy consumption of the video decoding process.

Finally, as explained in Section 4.1.1, the above presented results came from experiments which were conducted on two different platforms (Snapdragon 810 and Odroid-xu3). This can give an idea about the trends when comparing HW and SW video decoding energy consumption. To validate these trends, other experiments were conducted on a single platform (RB3) which will be presented later, in Section 4.4.

4.4 Results generalization

In this section, the influence of the video bitrate, frame rate, and resolution on the HEVC HW and SW decoding energy consumption is evaluated. In these experiments, both HEVC HW and SW decoding are decoded on a single platform (RB3). The objective of these results is to validate those obtained when running on two different platforms: Snapdragon 810 and Odroid-xu3.

The video decoding consumed energy is calculated using Equation (3.11) and is given in mJ/Frame. *Kimono* video sequence is taken as an example to vary bitrate (at 1080p resolution), frame rate (at 1080p resolution), and resolution. Experiments were conducted on other video sequences as well and the curves show similar behaviors. Video sequences were decoded: (i) in HW using the GPP–HDIP, and (ii) in SW using a heterogeneous GPP at the optimal configuration (number of cores and their clock frequency) that offers the best trade-off between performance and energy consumption, as explained in Section 3.3.1.

4.4.1 Varying video bitrate

Fig.4.16, shows the video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (3.11), when varying the video bitrate parameter. As a general note, the energy consumption values on RB3 are less than those on Snapdragon 810 and Odroid-xu3, see Fig.4.13. One cause of this behavior is that RB3 consumes less dynamic power. Indeed, it is equipped with a SoC fabricated at a lower technology process (10 nm) than that of the Snapdragon 810 and Odroid-xu3 platforms. Also, the curve shows similar trends as those shown in Fig.4.13.

In case of HEVC HW decoding, the energy consumption (mJ/Frame) is almost steady as observed on the Snapdragon 810 platform. This confirms that HDIP always tries to

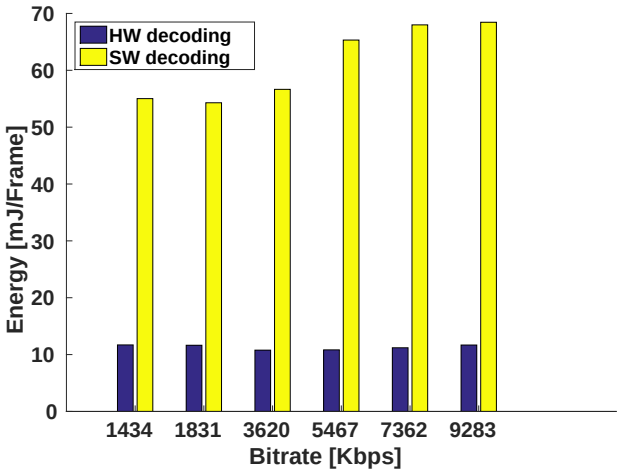


Figure 4.16 – HEVC decoding energy when varying video bitrate (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, frame rate: 25 Hz, resolution: 1080p, platform: RB3)

supply the necessary resources to avoid the energy over-consumption.

In case of HEVC SW decoding, globally, the consumed energy (mJ/Frame) increases as the video bitrate grows high. Compared to the results presented in Fig.4.13, the RB3 platform seems to be less scalable in terms of energy consumption. Indeed, in Fig.4.16, decoding the 9283 Kbps video sequence would increase the energy consumption (mJ/Frame) on average by 1.24× compared to the 1434 Kbps video sequence.

4.4.2 Varying video frame rate

Fig.4.17, shows the video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (3.11), when varying the video frame rate parameter. As a general note, again the energy consumption values on RB3 are less than those on Snapdragon 810 and Odroid-xu3, see Fig.4.14. Also, the curve shows similar trends as those shown in Fig.4.14.

In case of HEVC HW decoding, the energy consumption (mJ/Frame) increases every time the video frame rate is increased. As explained earlier, this can be caused by the fact that the higher the frame rate, the more memory transfers are needed in order to copy encoded and decoded frames from and to memory. And so, the more I/O operations are needed. Therefore, more consumed energy (mJ/Frame).

In case of HEVC SW decoding, the energy (mJ/Frame) increases as the video frame rate grows high. Contrary to the HEVC HW decoding energy consumption in Fig.4.17,

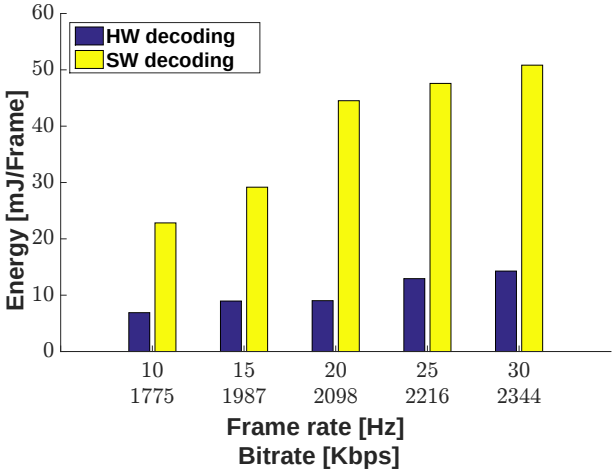


Figure 4.17 – HEVC decoding energy when varying video frame rate (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, resolution: 1080p, platform: RB3)

the HEVC SW decoding energy consumption is more scalable.

4.4.3 Varying video resolution

Fig.4.18, shows the video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (3.11), when varying the video frame rate parameter. The video resolution varies from 720p to 2160p. Table 7.6 summarizes the scaling factors of video resolutions with respect to the smallest one (720p). As a general note, once more again the energy consumption values on RB3 are less than those on Snapdragon 810 and Odroid-xu3, see Fig.4.15. Also, the curve shows similar trends as those shown in Fig.4.15.

In case of HEVC HW decoding, scaling the video resolution from 720p to 2160p through 1080p and 1600p would consume only $1.66\times, 2.27\times$ and $3.37\times$ more energy (mJ/Frame), respectively. These ratios are not remarkable compared to the ratios of pixels among those resolutions, see Table 4.5. This is because the HDIP, in the tested platform (RB3), is designed for 2160p (a.k.a. 4K) video content. In addition, this shows that, the HDIP did not seem to implement mechanisms that adapt the energy consumption when decoding low resolution videos.

In case of HEVC SW decoding, as expected, Fig.4.18 confirms that the energy consumption (mJ/Frame) of a mobile device depends heavily on the video resolution. However, the scaling is not always linear. For instance, playing back a 1080p video sequence

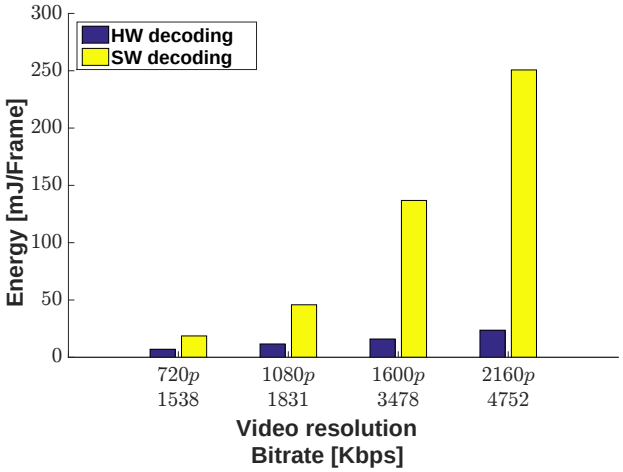


Figure 4.18 – HEVC decoding energy when varying video resolution (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, frame rate: 25 Hz, platform: RB3)

Table 4.8
Energy consumption ratio as a function of video resolution, see Equation (7.15). P.149, on RB3

Resolution	720p	1080p	1600p	2160p
$\hat{\mathbf{r}}_{\mathrm{sw/hw}}$	2.65	3.93	8.57	10.6

would consume on average 2.46× more energy (mJ/Frame) than playing back a 720p video sequence. Above this resolution, the ratio increases substantially: 7.34× and 13.44× for 1600p and 2160p resolutions, respectively. These ratios confirm those obtained when running on two different platforms (Snapdragon 810 and Odroid-xu3).

Table 7.6 depicts the ratio between the HEVC SW and HW decoding energy consumption, calculated using Equation (7.15), as a function of the video resolution. Compared to the state-of-the-art [22] study where it is stated that dedicated processors outperform the GPPs by around 1000× in terms of energy efficiency, for low video resolutions (720p and 1080p), from the system point of view, the gap is not as high as that of the state-of-the-art work (a ratio of less than 4×). However, for high video resolutions, the GPP–HDIP offer the best energy efficiency (a ratio of more than 8×).

Summary: As a conclusion, the results obtained on RB3 confirm those obtained on Snapdragon 810 and Odroid-xu3. first, the video resolution impacts differently the

HW and SW video decoding (on RB3). Scaling the video resolution from 720p to 2160p through 1080p and 1600p would increase the energy by 1.66×, 2.27× and 3.37× in case of the former and 2.46×, 7.34× and 13.44× in case of the latter. Therefore, the experiments on RB3 confirm that the HW video decoding is more suited for high video resolutions.

Then, as explained earlier, the video frame rate and resolution parameters are highly correlated with the video bitrate. For instance, when the resolution of a given video sequence is changed to keep its frame rate, the bitrate should be scaled accordingly. The higher the video resolution, the higher the bitrate. Similarly, the higher the frame rate, the higher the bitrate. Therefore, Fig.4.16, Fig.4.17, and Fig.4.18 show similar behaviors.

Finally, we think that the difference of ratios between the results obtained on Snapdragon 810 and Odroid-xu3 and on RB3 might be explained by the fact that RB3 consumes a substantial amount of static power as it is equipped with a SoC fabricated at a lower technology process (10 nm) than that of the Snapdragon 810 and Odroid-xu3 platforms [144]. Unfortunately, our measurement system does not allow to estimate this power.

4.5 Summary

Finally, the results of the proposed video decoding performance and power consumption characterization methodology is summarized in Table 4.9 and Table 4.10, for the operating system and application levels, respectively.

Table 4.9
The results summary of the proposed HEVC decoding performance and power consumption characterization methodology (video sequence example: *Kimono*) : OS level

<table>
<tr><td></td><td></td><td colspan="6">Tested mobile platforms</td></tr>
<tr><td></td><td></td><td colspan="2">Qualcomm Snapdragon 810</td><td colspan="4">Odroid-xu3</td><td>Qualcomm RB3</td></tr>
<tr><td rowspan="6">OS level</td><td rowspan="4">Idle state</td><td>Global platform</td><td>GPP one core</td><td colspan="2">Global platform</td><td colspan="2">GPP all cores</td><td rowspan="4">-</td></tr>
<tr><td colspan="2">At 1344 MHz</td><td>Min freq</td><td>Max freq</td><td>Min freq</td><td>Max freq</td></tr>
<tr><td>2.01 Watts</td><td>0.22 Watts</td><td>2.49 Watts</td><td>3.46 Watts</td><td>0.13 Watts</td><td>0.82 Watts</td></tr>
<tr><td colspan="2">$\hat{P}_{\text{Other\_idle}}$ represents 88% of the global platform power</td><td colspan="4">$\hat{P}_{\text{Other\_idle}}$ represents 94% and 76% of the global platform power for the min and max GPP frequency, respectively.</td></tr>
<tr><td colspan="2">-</td><td colspan="2">Saving 28% of power when scaling GPP freq from max to min</td><td colspan="2">Saving 84% of power when scaling GPP freq from max to min</td></tr>
<tr><td rowspan="3">HW dec</td><td colspan="2">$P_{\text{glob\_dyn}}$ represents 2.75 Watts</td><td colspan="4" rowspan="3">-</td><td rowspan="3">-</td></tr>
<tr><td colspan="2">$\hat{P}_{\text{Other}}$ represents on average more than 70% of the global platform power</td></tr>
<tr><td colspan="2">The duration of doze mode after the frame decoding represents on average 5.8% of the video decoding period</td></tr>
<tr><td rowspan="4">SW dec</td><td colspan="2" rowspan="4">-</td><td colspan="4">The optimal config is using 1 big core and 4 LITTLE cores, clocked at 1.2 GHz</td><td rowspan="4">-</td></tr>
<tr><td colspan="4">GPP heterogeneity impact: config (d) in Table 3.1 is 29% more energy efficient (mJ/Frame) than config (a)</td></tr>
<tr><td colspan="4">$P_{\text{glob\_dyn}}$ represents 3.91 Watts</td></tr>
<tr><td colspan="4">$\hat{P}_{\text{Other}}$ represents on average more than 50% of the global platform power</td></tr>
</table>

Table 4.10
The results summary of the proposed HEVC decoding performance and power consumption characterization methodology (video sequence example: *Kimono*) : application level

		Tested mobile platforms		
		Qualcomm Snapdragon 810 (HW dec)	Odroid-xu3 (SW dec)	Qualcomm RB3 (HW and SW dec)
Application level	Bitrate	The energy consumption (mJ/Frame) is almost steady. This confirms that the HDIP is not scalable for low data rate	The consumed energy (mJ/Frame) increases as the video bitrate grows high. However, the increase is not linear	In case of HEVC HW decoding, the energy consumption (mJ/Frame) is almost steady, whereas in case of HEVC SW decoding, the consumed energy (mJ/Frame) increases as the video bitrate grows high. However, the increase is very slight and is not linear
		-	Decoding the 9283 Kbps video sequence would increase the energy consumption (mJ/Frame) on average by 3.08× compared to the same video sequence encoded at 1434 Kbps	In case of HEVC SW decoding, decoding the 9283 Kbps video sequence would increase the energy consumption (mJ/Frame) only by 1.24× compared to the same video sequence encoded at 1434 Kbps
	Frame rate	The energy consumption (mJ/Frame) increases every time the video frame rate is increased. However, the increase is slight. This confirms that the HDIP is not scalable for low data rate	The consumed energy (mJ/Frame) increases as the video frame rate grows high	The energy (mJ/Frame) increases as the video frame rate grows high for both HEVC HW and SW decoding
		Decoding the video sequence encoded at 30 Hz would increase the energy consumption by only 1.34× compared to the same video sequence encoded at 10 Hz	Decoding the video sequence encoded at 30 Hz would increase the energy consumption by 2.72× compared to the same video sequence encoded at 10 Hz	
	Resolution	The energy consumption (mJ/Frame) increases every time the video resolution is increased. However, the increase is not the same as that of the resolution scaling factors, see Table 4.5	The consumed energy (mJ/Frame) increases as the video resolution grows high. However, the increase is not linear	The energy consumption (mJ/Frame) increases every time the video resolution is increased for both HEVC HW and SW decoding. However, the increase is not the same as that of the resolution scaling factors, see Table 4.5
		Decoding the video sequence of 2160p would increase the energy consumption by 2.14× compared to the same video sequence at lower resolution (720p)	Decoding the video sequence of 2160p would increase the energy consumption by 13.76× compared to the same video sequence at lower resolution (720p)	Decoding the video sequence of 2160p would increase the energy consumption by 3.37× and 13.44× compared to the same video sequence at lower resolution (720p) for HEVC HW and SW decoding, respectively
		For 720p video sequence, HEVC SW decoding consumes as energy as that of HEVC HW decoding (factor of 1.15×)		For 720p and 2160p video sequences, HEVC SW decoding consumes 2.65× and 10.6× more energy than what HEVC HW decoding does, respectively
		For 2160p video sequence, HEVC SW decoding consumes 7.4× more energy than what HEVC HW decoding does		

4.6 Conclusion

In this chapter, the results of the HEVC decoding performance and energy consumption characterization which answered to the research questions RQ1 and RQ2 stated in the problem statement are described and analyzed. The results are presented in two sections: (i) operating system level, and then (ii) application level. First, at the operating system level, the power is studied at idle state. Then, the power consumption of the HW video decoding, followed by that of the SW video decoding are analyzed. After that, the HW video decoding energy efficiency is compared to that of the SW one. Next, at the application level, the HW video decoding energy efficiency is compared to that of the SW one by studying the impact of video quality parameters: bitrate, frame rate, and resolution.

In short, a characterization methodology to evaluate the HEVC decoding performance and energy consumption is proposed. The HEVC decoding is investigated, by measurement, in both implementation approaches: HW, using the GPP–HDIP, and SW, using only a heterogeneous GPP processor. The performance as well as the power/energy consumption of HEVC decoding is studied by varying metrics triggered at operating system and application levels.

First, the proposed video decoding characterization methodology allows to find the best trade-off between performance and energy consumption.

Second, the energy consumption of a mobile device depends on the video quality parameters (bitrate, frame rate, and resolution). While in case of HW video decoding situation, the three parameters are less crucial. However, when it comes to the SW video decoding, we are being confronted with high energy consumption when scaling up the video quality parameters (bitrate, frame rate, and resolution).

Third, the results showed that, for low resolutions (720p and 1080p) and on the tested platforms, the SW video decoding consumes, at the operating system level, less than 4× more energy than that of the HW one. As a consequence, in more than 80% of mobile devices use cases that are equipped with 1080p or lower screen resolution, the SW video decoding can be an acceptable solution. This is because the HDIP seems very costly regarding the energy efficiency it presents for low video resolutions, in addition to its lack of flexibility. However, the HW video decoding stays more suited for high video bitrate, frame rate, and resolution parameters.

Finally, on the tested platforms, from the system point of view, the power required for the video decoding process does not dominate the global power consumption of the

platform. Actually, the video decoding process consumes effectively on average less than 30% and 50% of the global power consumption in case of HW and SW video decoding, respectively. The rest of the power includes the inter processor communication between the video decoder (GPP–HDIP or GPP), memory transfers, and other elements involved in video decoding. Therefore, the HDIP, when integrated in a SoC, is not so efficient and thus more efforts should be done to deal with this drawback.

In the next chapter, in continuity of the proposed performance and energy consumption characterization methodology, our proposed optimization is presented. It consists in reducing video decoding energy consumption dynamically, i.e., during run-time.

Chapter 5

Video decoding energy consumption optimization

Mobile video content is ubiquitous thanks to the omnipresence of mobile platforms. Besides, the mobile platforms are equipped with heterogeneous multi-cores architectures, such as ARM big.LITTLE, which have known great advances nowadays. In addition, built-in Linux governors to adjust the GPP frequency and thus save energy does not take into account the specificities of video decoding applications. In this context, we propose a solution that exploits these advances in order to reduce the SW video decoding energy consumption.

In this chapter, our proposed strategy solution to solve the problem statement (RQ3 [1] and RQ4 [2]) is presented. First, an overview of the proposed solution is described. Then, the solution is detailed by explaining different phases. For better readability of the manuscript, the proposed solution is summarized in Table 5.1. Next, the experimental methodology setup is given. Finally, the obtained results are presented and analyzed.

5.1 Video decoding energy consumption optimization overview

This section gives on overview of our proposed solution. It includes three different phases, as depicted in Fig.5.1: (i) modeling of frame complexity, (ii) assignment of frames to appropriate GPP cores, and (iii) frequency scaling [41]. The first phase is performed offline, i.e., during design time, whereas the two other phases are performed online, i.e., during the decoding process.

1. RQ3: How to balance the video content workload among the heterogeneous GPP cores, based on a little amount of information on the video to decode?
2. RQ4: Once the GPP cores are selected, how to adjust their frequency in order to reduce the energy consumption?

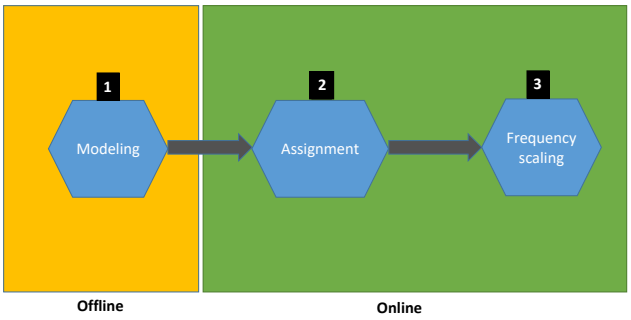


Figure 5.1 – The proposed solution overview

In the first phase, a model of frame complexity is established. The objective of this model is to balance frames in a heterogeneous mobile architecture composed of: (a) high performance GPP cores which consume a large amount of energy, and (b) low performance ones which consume much less energy (referred to as energy-efficient GPP cores). For that, the model classifies the given video frames into two groups, according to their complexity in terms of the number of GPP clock cycles required to decode them: (i) most complex frames, and (ii) least complex frames. This phase is performed during design-time, i.e., only once during the target platform life-time. Therefore, the model is applied to any new video frame which was not included in the test dataset.

The classification is enabled thanks to a logistic function [145]. The input parameters are: the frame type and the frame size. The configuration parameters are: the video bitrate, the video frame rate, and the ratio of performance between the heterogeneous GPP cores. The output of this model is the type of GPP cores (high performance or energy-efficient).

In the second phase, the classification is applied using the established model in the first phase. This phase is performed online, i.e., when decoding a video. For that, the next frame to decode is the input of the classifier. The classifier decides whether the frame should be submitted to the high performance or energy-efficient GPP cores.

In the third phase, the clock frequency of the selected GPP cores in the second phase is calculated using a feedback control technique. This phase is performed during run-time after the classification phase. For that, the solution proposed in [41] is used and integrated to the proposed one. The solution consists in using a PI controller. This latter monitors the size of the output buffer which contains the decoded frames waiting for display. Actually,

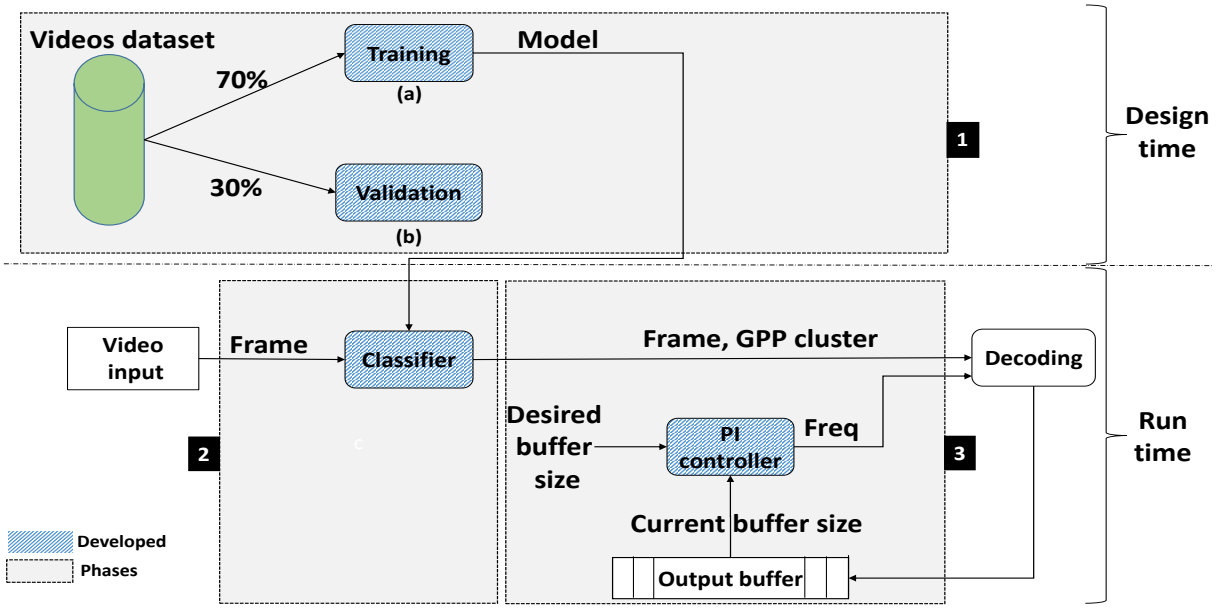


Figure 5.2 – The proposed solution overview

the PI controller has two inputs: (i) the set point, i.e., the desired output buffer size as an input, and (ii) the current output buffer size. Then, according to the output buffer size, the PI controller adjusts frequency of the GPP cores selected GPP in phase 2, so that the current buffer size meets the set point.

5.2 Video decoding energy consumption optimization

In this chapter, we propose a solution to reduce the energy consumption of HEVC SW decoding on mobile platforms. It aims at optimizing the HEVC SW decoding in order to approach the energy consumed by the HEVC HDIP of the target platform while satisfying the real-time decoding constraint. This is achieved by two mechanisms: parallelism and frequency scaling.

Fig.5.2 depicts an overall diagram of the proposed solution. It includes three different phases: (1) modeling of frame complexity, (2) assignment of frames to appropriate GPP cores (using classification), and (3) frequency scaling (using DVFS with feedback control). These phases are described in the following subsections.

5.2.1 Phase 1: Modeling of frame complexity

The objective of the first phase is to build a model of frame complexity which is able to balance the video frames workload between the high performance and energy-efficient GPP cores. This is made possible by classifying any given video frame into two groups: (i) most complex frames to be decoded by high performance GPP cores, and (ii) least complex frames to be decoded by energy-efficient GPP cores. The complexity is expressed as the number of GPP clock cycles required to decode a frame. Thanks to this classification, the first group will be submitted to the high performance GPP cores and the latter to the energy-efficient ones.

This phase is performed during design-time, i.e., offline. It is done only once in the whole target platform life-time. The output of this phase is a model which is used by phase 2 as illustrated in Fig.5.2.

To build the model of frame complexity, as illustrated in Fig.5.2, there are two steps: (a) training of the model, and (b) validation of the model. In the first step, data related to frames representing multiple video sequences are collected. Then, a part of them (70%) is injected to the model for training, i.e., the model takes those data to learn how to correlate the input parameters to the output one which is the frame complexity. After that, in the second step, the model is applied on the remaining data (30%) to measure its accuracy, and so to accept it or to reject it.

Fig.5.3 depicts the inputs and output of the established model. The output is the group of the frame: (i) most complex frames, or (ii) least complex frames. The inputs are the independent variables (a.k.a. features): frame type and frame size. In a previous work [146], it has been shown that these two parameters are very correlated to the frame complexity in MPEG video codec. Therefore, we kept them in our model. However, in HEVC, the correlation is not as high as what is stated in the state-of-the-art work. Indeed, in our experiments, the coefficient of determination was found to be weak ($R^2 = 0.55$). As a result, other features should be added to improve the accuracy of the model. We added three configuration parameters.

The added configuration parameters are: (i) video bitrate, (ii) video frame rate, and (iii) the performance ratio between the high performance and energy-efficient GPP cores. First, the video bitrate has a correlation with the video decoding energy consumption. Indeed, as explained in our characterization work, see Section 4.3, in case of HEVC SW decoding, the energy consumption increases with the video bitrate, and vice versa. Second, the video frame rate is used to determine the frame decoding deadline. Finally, the

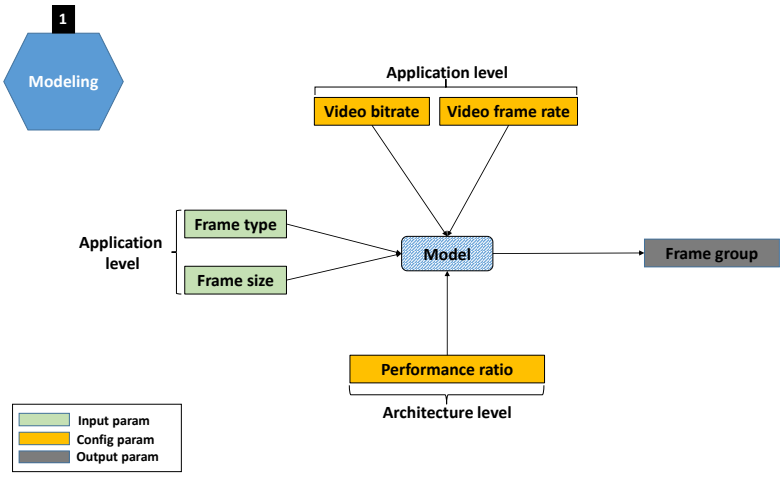


Figure 5.3 – The proposed solution: phase 1 (Modeling of frame complexity)

performance ratio between the high performance and energy-efficient GPP cores is used as the GPP cores offer heterogeneous performances.

Algorithm 1: Collecting frames data algorithm

```
1  for each video sequence do
2      for each frame do
3          submit_to_energy_efficient();
4          set_energy_efficient_gpp_freq_to_max(); // Set the GPP speed to max
5          dec_time = decode_the_frame(); // Decode the frame and calculate its decoding time
6          if dec_time <= D_period then
7              gpp_cores_type = energy_efficient;
8          else
9              gpp_cores_type = high_performance;
10         end
11         store(video_bitrate, frame_type, frame_size, gpp_cores_type); // in a log-file
12     end
13 end
```

Algorithm 1 describes the procedure used to collect data for training the model. First, every single frame is parsed. Then, in line 3, the frame is submitted to the energy-efficient GPP cores which are clocked at their highest supported frequency value, see line 4. The decoding of the frame is launched, in line 5. Next, the decoding time is compared to D_{period}, which represents the deadline of decoding, see Equation (2.10). If the decoding real-time constraint (deadline) is satisfied, the frame should be submitted to the energy-efficient GPP cores, in line 7; otherwise, it should be submitted to the high performance ones, in line 9. Lastly, in line 11, the parameters described previously are stored in a log file to feed the training process. Fig.5.4 depicts the collecting data process in a graphical

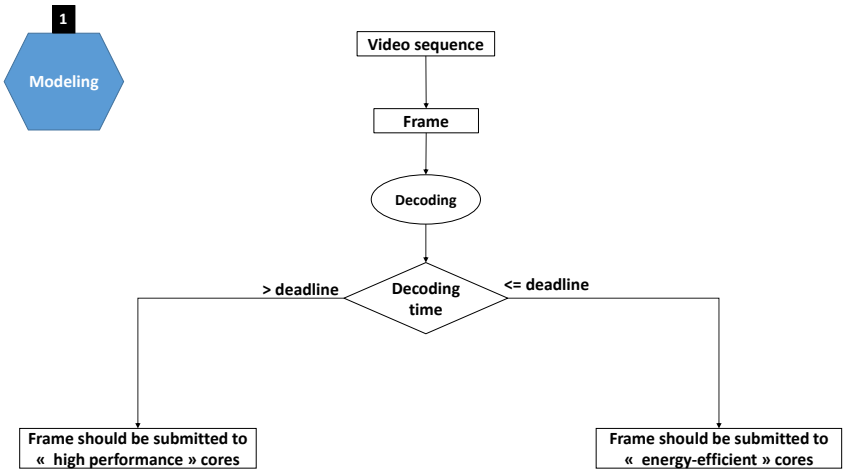


Figure 5.4 – The proposed solution: phase 1 (Modeling of frame complexity)

representation.

Finally, the regression model used to train the video frames data is the logistic regression. The reason of this choice is its simplicity of implementation and its efficiency to take a binary decision (most complex or least complex frames). Other regression models are not accurate enough to model the complexity of frames. For instance, the linear regression has an R^2 of 0.55, whereas the polynomial regression (degree = 2) has an R^2 of 0.79.

Logistic Regression is a statistical model used to study the relationship between a set of independent variables and a dependent variable. It is used when the dependent variable is categorical. In our case, the dependent variable is the frames group: (i) most complex frames, and (ii) least complex frames.

The model resulting from this phase is expressed by the following formula, using a logistic function:

$$y = \frac{1}{1 + e^{-\mathrm{p(x1,\ x2,\ x3)}}} \tag{5.1}$$

where y is the output of the model which will be used in phase 2. It takes values between 0 and 1, and $p(x1, x2, x3)$ is the linear function of the input and configuration parameters described above. It is a real number.

The linear function is expressed as follows:

$$p(v_{\mathrm{bitrate}}, f_{\mathrm{type}}, f_{\mathrm{size}}) = \frac{w_0 + w_1 * v_{\mathrm{bitrate}} + w_2 * f_{\mathrm{type}} + w_3 * f_{\mathrm{size}}}{ratio\_performance} \tag{5.2}$$

where w_0 is the intercept, v_{bitrate} is the the video bitrate, f_{type} and f_{size} are the type and size of the frame to decode, respectively, and $ratio\_performance$ is the ratio of performance between the high performance and energy-efficient GPP cores of the target platform. Finally, w_1, w_2, and w_3 are the coefficients of the model.

5.2.2 Phase 2: GPP cores assignment using classification

The objective of the second phase is to decide which GPP cores (high performance or energy-efficient) will decode the next frame. For that, the model built in the previous phase classifies the frames into two groups: (i) most complex frames, and (ii) least complex frames. The first group will be submitted to the high performance GPP cores and the latter to the energy-efficient ones. Then, the frame is decoded in parallel among the selected GPP cores via tiling or WPP parallelism scheme (depending on the coding configurations).

This phase is performed during run-time, i.e., while performing video decoding. It is applied for each frame of the input video. The output of this phase is the GPP cores type (high performance or energy-efficient) which is the input of phase 3 as illustrated in Fig.5.2.

Algorithm 2: Classifier module algorithm

```
1 init_config_params(); // Initialize: bitrate, frame rate, and performance ratio
2 gpp_cores_estim = p(v_bitrate, f_type, f_size); // Run the logistic function
3 if gpp_cores_estim > 0 then
4 |   submit_to_high_performance_cores();
5 else
6 |   submit_to_energy_efficient_cores();
7 end
```

The classification is realized using Equation (7.17). Algorithm 2 describes the procedure to perform the classification. First, in line 1, configuration parameters are initialized: the video bitrate and frame rate are obtained from the input video sequence, and the ratio of performance between the high performance and energy-efficient GPP cores of the target platform is obtained from its dedicated data-sheets. Then, in line 2, the estimated GPP cores, gpp_cores_estim, is calculated using Equation (7.17). If the result is positive, the frame is submitted to the high performance GPP cores, seen line 4; otherwise, it is submitted to the energy-efficient GPP cores, see line 6. Note that if two consecutive frames or more are submitted to the same GPP cores type, they are stored in an input buffer. Fig.5.5 depicts the algorithm of classification in a graphical representation.

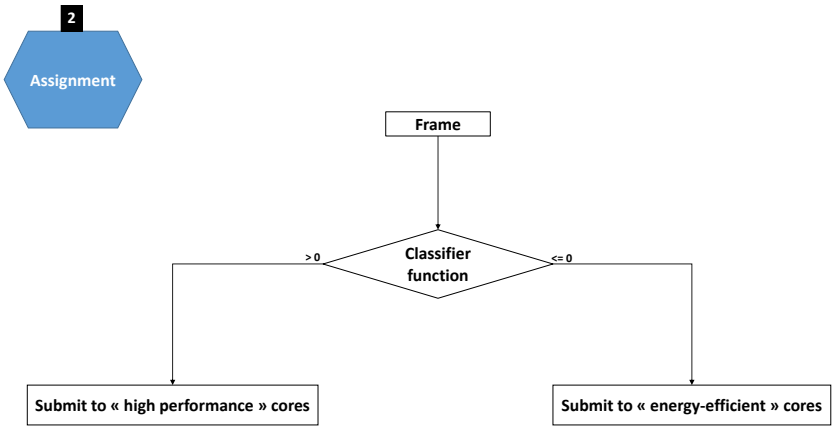


Figure 5.5 – The proposed solution: phase 2 (Assignment using classification)

5.2.3 Phase 3: Frequency scaling using PI controller [41]

The third phase is fully inspired from previous work [41]. The objective of this phase is to select the clock frequency at which the selected GPP cores will decode the current frame. For that, the Proportional Integral (PI) controller proposed in [41] is adopted. This controller monitors the output buffer size in order to maintain it at a desired value (set point) [3] which is an input parameter of the controller. The output buffer is a buffer used to store the decoded frames before they are sent to the display system. So, when the output buffer size exceeds the desired value, the controller slows down the GPP cores speed, and vice versa.

This phase is performed during run-time, i.e., while performing video decoding. It is applied for each frame of the input video. The output of this phase is the clock frequency of the GPP cores selected in phase 2, as illustrated in Fig.5.2.

Algorithm 3: PI controller algorithm

```
1 buffer_size = get_buffer_size();
2 if (buffer_size != set_point) then
3 |   error = set_point - buffer_size;
4 |   update_GPP_cores_freq(); // See Algorithm 4
5 else
6 |   do_nothing();
7 end
```

3. To set the desired value, one can follow the guidelines given in the literature review, such as [41][115].

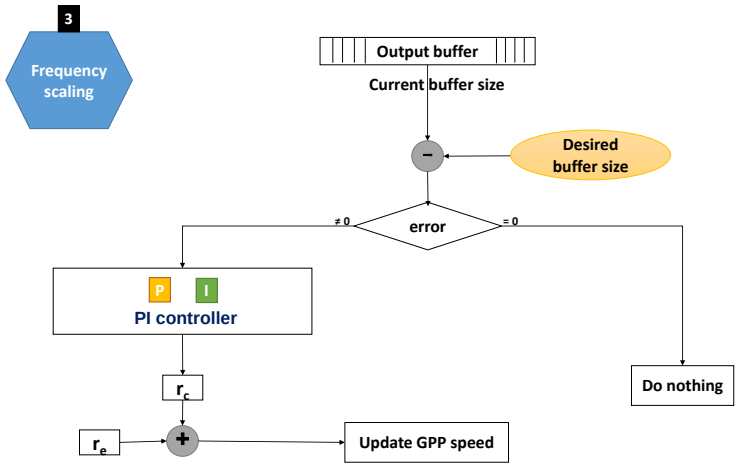


Figure 5.6 – The proposed solution: phase 3 (DVFS using PI controller)

Algorithm 3 describes the procedure of the PI controller to adjust the GPP cores frequency. First of all, the output buffer size is obtained. Then, the error is calculated which is the difference between the current output buffer size and the desired value. If the error is equal to zero, there is nothing to do. If the error is positive, the PI controller should speed up the GPP cores, whereas a negative error signifies that the GPP cores should be slowed down. Note that the desired output buffer size value (set point) is set at the beginning of the video decoding process, i.e., before starting this algorithm.

Updating the GPP cores frequency

To speed up or slow down the GPP cores, a scaling factor, r, is calculated and then multiplied by the highest supported GPP cores frequency.

$$gpp\_cores\_freq = GPP\_\text{cores}_{\text{max\_freq}} * r \tag{5.3}$$

where gpp_cores_freq is the output clock frequency of the GPP cores selected in phase 2, $GPP\_\text{cores}_{\text{max\_freq}}$ is their highest supported frequency value, and r is the scaling factor of that frequency. gpp_cores_freq is rounded to the higher supported frequency as the GPP cores support discrete frequency values. Rounding to higher frequency allows to avoid deadline misses. r takes values between 0 and 1. For instance, an r value of 1 implies that the GPP cores decode the current frame at its highest speed.

The scaling factor r is, in turn, decomposed into two components: one for an estimation

based on the history of the decoded frames and the second to adjust the estimation as it might be some missed deadlines in the past, as illustrated by the following formula:

$$r(n) = r_{\mathrm{e}}(n) + r_{\mathrm{c}}(n) \tag{5.4}$$

where r_{e} is the scaling factor estimation based on the previously decoded frames, r_{c} is the output of the PI controller which is considered as an adjustment of r_{e}, and n is the number of the next frame to decode. That is, a negative value of r_{c} indicates that the GPP cores should be slowed down, and vice versa.

After that, the scaling factor r_{e} is the optimal scaling factor in the case where the previously decoded frames were decoded in real-time, i.e., before their deadline. It is calculated by the following formula:

$$r_{\mathrm{e}}(n) = \frac{1}{D_{\mathrm{period}}} * \frac{\sum_{j=0}^{j=n-1} C_{\mathrm{j}}}{nb\_dec\_frames} \tag{5.5}$$

where D_{period} is the period of decoding. It represents the deadline before which the frame should be decoded. It is given by Equation (2.10), P.36. C_{n} is the decoding time of $frame_{\mathrm{n}}$, nb_dec_frames is the number of already decoded frames, and n is the number of the next frame to decode. n cannot be equal to zero as the controller is not engaged before to decode at least one frame. Also, it is bounded by a number, e.g., the number of frames in one minute of decoding.

Next, the output of the PI controller, r_{c}, is given by the following formula:

$$r_{\mathrm{c}}(n) = K_{\mathrm{p}} * error(n-1) + K_{\mathrm{i}} * \sum_{j=0}^{j=n-1} error(j) \tag{5.6}$$

where K_{p} and K_{i} are the proportional and integrated coefficients of the PI controller, respectively, $error(n-1)$ is the error resulting after decoding the last frame, and n is the number of the next frame to decode. That is, $\sum_{k=1}^{k=n-1} error(k)$ is reset to zero every one minute of decoding.

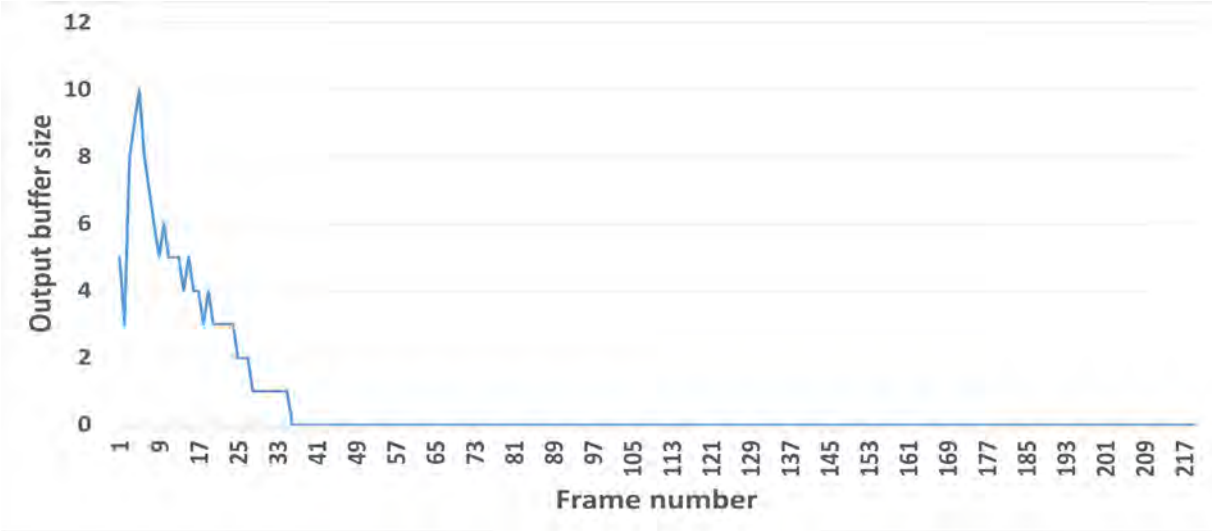


Figure 5.7 – Video decoding in open loop mode

Algorithm 4: update_GPP_cores_freq()

C_{n}: decoding time of $frame_{\mathrm{n}}$;
D_{period} : decoding deadline;
$GPP\_\mathrm{cores}_{\mathrm{max\_freq}}$: the highest freq supported by the selected GPP cores;
$r_{\mathrm{e}}(n) = \frac{1}{D_{\mathrm{period}}} . \frac{\sum C_{\mathrm{n}}}{nb\_dec\_frames}$; // Take into account the history of decoded frames
$r_{\mathrm{c}}(n) = K_{\mathrm{p}} * error(n-1) + K_{\mathrm{i}} * \sum error$; // Output of the PI controller
$r = r_{\mathrm{e}}(n) + r_{\mathrm{c}}(n)$;
$gpp\_cores\_freq = GPP\_\mathrm{cores}_{\mathrm{max\_freq}} * r$; // Output GPP frequency rounded to the nearest discrete supported frequency

Algorithm 4 summarizes the procedure to update the GPP cores frequency.

To tune the PI controller parameters (K_{p} and K_{i}), we used an empirical method. The reason is that the conditions to apply, e.g., both Ziegler–Nichols methods [147] are not satisfied in our experiments. For instance, Fig.5.7 shows the variation of the output buffer size when decoding a test video sequence in an open loop mode, i.e., without using any controller. Obviously, Ziegler-Nichols method designed for an open loop process is not suited for this curve. Similarly, in closed loop mode, the condition of regular oscillation is not satisfied in none of our experiments.

Tuning the PI controller coefficients

Tuning PI controller coefficients is performed in two steps: (i) tune initial values, and (ii) refine those values by running intensive experiments.

To tune the proportional parameter (K_p), let K_i be equal to zero. Then, let $r_\mathrm{e} = 0.5$. This means that 0.5 is the optimal ratio so that the decoded frames be decoded within their deadline. After that, suppose the case where there are already five frames which were decoded, and that there is no frame in the output buffer (the worst case). According to Algorithm 3, error = 5. In this case, the selected GPP cores should be run at their full speed, i.e., according to Equation (7.19), r = 1.

Next, according to Equation (7.19), $r_\mathrm{c} = 0.5$. By applying Equation (5.6), $K_\mathrm{p} = 0.1$.

To tune the initial value of the integral parameter (K_i), we suppose that it is one order of magnitude less with respect to K_p as K_i is multiplied by the sum of errors. By applying this to the example above, we get $K_\mathrm{i} = 0.01$.

Once the initial values of K_p and K_i are tuned, they are then refined by carrying out other benchmarks, i.e., decoding other video sequences. The most important criteria to tune those parameters is the stability time, i.e., how many frames should be decoded in order to get to the stability state of the controller where the GPP cores frequency does not need to be updated.

At the end of phase 3, as illustrated in Fig.5.2, the selected GPP cores in phase 2 start decoding the current frame at the frequency calculated in phase 3. Then, after decoding, the decoded frame is sent to the output buffer for display [4]. The size of this buffer is thus updated and the new value will be the input of the PI controller for the next frame to decode.

5.2.4 Contribution summary

In summary, our proposed solution consists of three phases:

1. Modeling of frame complexity: to establish a model able to classify video frames into two groups: (i) the most complex frames, and (ii) the least complex frames.
2. Assignment of frames using classification: to decide to which GPP cores (high performance or energy-efficient) a frame should be submitted to.
3. Frequency scaling using DVFS with feedback control (PI controller) [41]: to monitor the output buffer size in order to adjust the GPP frequency. Given a desired output buffer size, the PI controller speeds up or slows down the GPP speed in order to maintain the desired value.

4. The display system study is beyond the scope of this paper. It is simulated by a process that receives decoded frames at regular intervals.

Table 5.1
The proposed video decoding energy consumption optimization summary

Phase	Label	Description
Phase 1	Modeling	- To model the complexity of frames - Decode each frame using the energy-efficient GPP cores, clocked at their highest frequency value - If the frame is decoded within the deadline, so it should be submitted to the energy-efficient GPP cores - Otherwise, the frame should be submitted to the high performance GPP cores - Realized during design-time, i.e., once in the system life-time
Phase 2	Assignment	- Using classification (logistic function) - Input parameters: frame (type, size) - Configuration parameters: video (bitrate, frame rate), and performance ratio between the high performance and energy-efficient GPP cores of the target platform - Output: GPP cores type (high performance or energy-efficient) - Realized during run-time
Phase 3	Frequency scaling [41]	- Using DVFS with feedback control (PI controller) - To adjust the GPP frequency in order to maintain a certain value (desired output buffer size) - To monitor the output buffer size - Realized during run-time

5.3 Experimental methodology

In this section, all the environment setups used to conduct our experiments are described. First, the HW setups which include the description of the different platforms used in our experiments are presented. Then, the process to measure the power consumption when performing video decoding is explained. After that, the SW used to run video decoding and design the proposed solution are stated. Next, the video sequences dataset used to validate the proposed methodology are given. Finally, the methodology to validate the proposed solution is explained.

5.3.1 HW setup

In our work, we applied the proposed solution to the ARM big.LITTLE mobile architecture which is composed of two clusters: (i) big, which contains high performance cores, and (ii) LITTLE, which contains energy-efficient cores. Specifically, we optimized the HEVC decoding energy consumption on three mobile platforms: Snapdragon 810 development board (APQ 8094) [129], Odroid-xu3 [130], and Qualcomm Robotics RB3 [131]. The first one is used to perform the HW video decoding and the second one for the SW video decoding. A different platform is used for the SW video decoding because the first one does not allow to control the GPP frequency. Then, the last platform (RB3) supports both HW and SW video decoding. Below, the HW and SW setups used in our experiments are described. Details about these platforms are described in Section 4.1.1.

5.3.2 Power consumption evaluation

In our experiments, N6705A DC Power Analyzer was used to measure power. The power measured by this tool is given by Equation (3.2). Then, the overall energy consumption, $\hat{E}_{\text{glob\_active}}$, is calculated using Equation (3.11). More details about the power consumption evaluation are described in Section 4.1.2.

5.3.3 SW setup

The SW setup was the same as that described in Section 4.1.3. In addition, the proposed optimization work is based on Open-HEVC software [137].

To build the model of phase 1 of the proposed solution, sickit-learn framework [148] was used via Python programming language.

5.3.4 Video sequences dataset

In this contribution, the dataset described in Table 4.2 was mainly used. The main difference is that the proposed solution is focused on the 1080p video resolution, and tiling and WPP parallelism schemes. There are 33 video sequences in the tested dataset. The common characteristics of these video sequences are summarized in Table 5.2.

5.3.5 Methodology

In our experimental methodology, we proceeded in four steps:

Table 5.2
Video sequences dataset characteristics

Parameter	Value
Resolution	1080p
Frame rate	25 and 30 fps
Mode	Random Access
Profile	Main
Parallelism scheme	Tiling and WPP

— build the model of frame complexity,
— tune the PI controller coefficients used for frequency scaling,
— evaluate the overhead of the proposed solution,
— compare the proposed solution to those of the state of the art.

1- Classification model

The first step is to build the classification model of frame complexity (phase 1 of the proposed solution). For that, 70% of video sequences dataset described in Table 5.2 were used for training the model. Then, the remaining 30% was used for its validation. It will be question of calculating the accuracy of the resulting model.

2- Tuning the PI controller coefficients

The second step is to tune the PI controller parameters (phase 3 of the proposed solution). The objective is to find the best values which: (i) guarantee the stability of the output buffer size, and (ii) reach this stability as fast as possible. Then, the stability of the output buffer size using the PI controller is studied. After that, to validate the tuned coefficients, i.e., the reaction of the controller, we change the set point during the decoding process.

3- Evaluation of the proposed solution overhead

The third step is to evaluate the overhead of the proposed solution (in percentage). It is expressed as the ratio between the time necessary to run the proposed solution with respect to the time required to decode a frame.

$$ratio\_overhead = \frac{ps\_time}{frame\_dec\_time} * 100 \tag{5.7}$$

where *ratio_overhead* is the overhead ratio of the proposed solution, *ps_time* represents the time spent to run the proposed solution (phases 2 and 3), and $frame\_dec\_time$ represents the time required to decode a frame.

The overhead is calculated for each frame. Then, the ratios are averaged over all the decoded frames of the tested video sequences dataset.

4- Comparison to state-of-the-art work

The last step is to compare the proposed solution with state-of-the-art work. Since our work is realized upon Open-HEVC software, it is relevant making comparison with this same software without the proposed techniques (classification and PI controller). First, the comparison was performed with Open-HEVC without optimizations and using the Performance Linux governor. The objective of this comparison is to draw the upper bound of the energy consumed by HEVC decoding.

Second, the proposed solution was compared with Open-HEVC without our optimizations and using the Ondemand Linux governor. The objective of this comparison is to show the impact of using parameters related to video decoding to adjust GPP frequency [5].

Third, the proposed solution was compared with our first contribution. In this work, a static solution that selects the best configuration (number of GPP cores and their clock frequency) to minimize the energy consumption of video decoding was proposed. The selected configuration does not change during decoding.

Fourth, the proposed solution was analyzed without the classification technique (phase 2) and then compared with the complete version of the solution.

Finally, the proposed solution was compared with the HW HEVC decoding of the target platform in order to compare with the lower bound of energy consumption.

5. Ondemand uses parameters related to the GPP workload in general.

Table 5.3
The proposed video decoding energy consumption optimization summary

Proposed solution	**State-of-the-art work**				
Classification + DVFS (PI controller)	Performance Linux governor	Ondemand Linux governor	Best configuration according to our characterization work (Chapter 3)	PI controller [41]	HW decoding (HDIP)

All those comparisons were performed first on two different platforms: Odroid-xu3 (SW video decoding) and Snapdragon 810 (HW video decoding). Then, the same comparisons were carried out on RB3 (both HW and SW HEVC decoding) for validation.

Table 5.3 summarizes the comparison of the proposed solution with the state-of-the-art work.

5.4 Results & analysis

In this section, the results of the HEVC decoding energy consumption optimization are described and analyzed. According to our proposed methodology, the results are presented in three sections: (i) study of the accuracy of the model built in phase 1, (ii) study of the stability of the output buffer size using the PI controller, and (iii) comparison of the proposed solution with other ones as described in Table 5.3.

5.4.1 Accuracy of the model

The first step of our methodology is to study the model built in phase 1 and then used in phase 2 of the proposed solution. The first phase of our proposed solution consists in modeling the complexity of video frames in order to classify them into two groups: (i) most complex frames, and (ii) least complex frames. The resulting model is a logistic function.

In case of Odroid-xu3 platform, the accuracy of the model was 93%, whereas 98% was achieved in case of RB3 platform. This indicates that at most 7 frames over 100 are not decoded by the right GPP cluster, e.g., they are decoded by the GPP LITTLE cluster instead of the big one. The result is that these frames may not be decoded within the deadline. Similarly, in the case where a frame is decoded by the GPP big cluster instead

of the LITTLE one, this might induce some slight extra energy consumption.

Fortunately, our proposed solution is able to mitigate this issue. Indeed, the decoded frames are inserted to the output buffer. Then, when multiple frames miss their deadline, the output buffer occupancy decreases quickly, whereas when multiple frames are decoded before their deadline, the output buffer occupancy increases quickly. Accordingly, if the size of this buffer becomes greater than the desired size value, the PI controller scales up the GPP frequency, and vice versa, using Algorithm 4. Therefore, the bad decisions that can be made by the classifier (using this model) are mitigated by the PI controller.

Summary: To sum up, the model of frame complexity was trained with 70% of video frames dataset. The remaining 30% was used for validation. The resulting model is a logistic function as it is simple to implement and accurate. Indeed, the accuracy of this model is 93% and 98% for Odroid-xu3 and RB3 platforms, respectively.

5.4.2 Stability of the output buffer size

In this section, the stability of the output buffer size with regard to the desired size value set by the user at the beginning of the decoding process is analyzed. For that, *Blue-sky* test video sequence, which possesses the characteristics given in Table 5.2 (parallelism scheme is WPP), is decoded as an example. Other 12 tested video sequences showed similar behaviors.

Fig.5.8 plots the frames complexity of *Blue-sky* test video sequence. The complexity is expressed as the number of CPU clock cycles required to decode a frame, using *rtdsc()* function. In this test video sequence, there is only one "I" frame. Moreover, this latter is on average 4× more complex than "P" frames which are, in turn, 3× more complex than "B" frames.

Fig.5.9 depicts the results of phase 2 and phase 3 of the proposed solution when decoding *Blue-sky* test video sequence. Phase 2 is represented by the variation of GPP clusters (bottom left), whereas phase 3 is represented by the variation of output buffer size (top left), and the GPP clusters clock frequencies (right side). The PI controller coefficients were tuned following the empirical method described in Section 7: $K_{\mathrm{p}} = 0.01$ and $K_{\mathrm{i}} = 0.001$. Also, the desired output buffer is set to five.

First, at the beginning of the decoding process, both GPP clusters (big and LITTLE) were set to their highest supported frequency, 2.0 GHz and 1.5 GHz for big and LITTLE GPP clusters, respectively. Clocking at these values allows filling the output buffer as fast as possible. Note that the display process does not receive frames until the fifth frame is

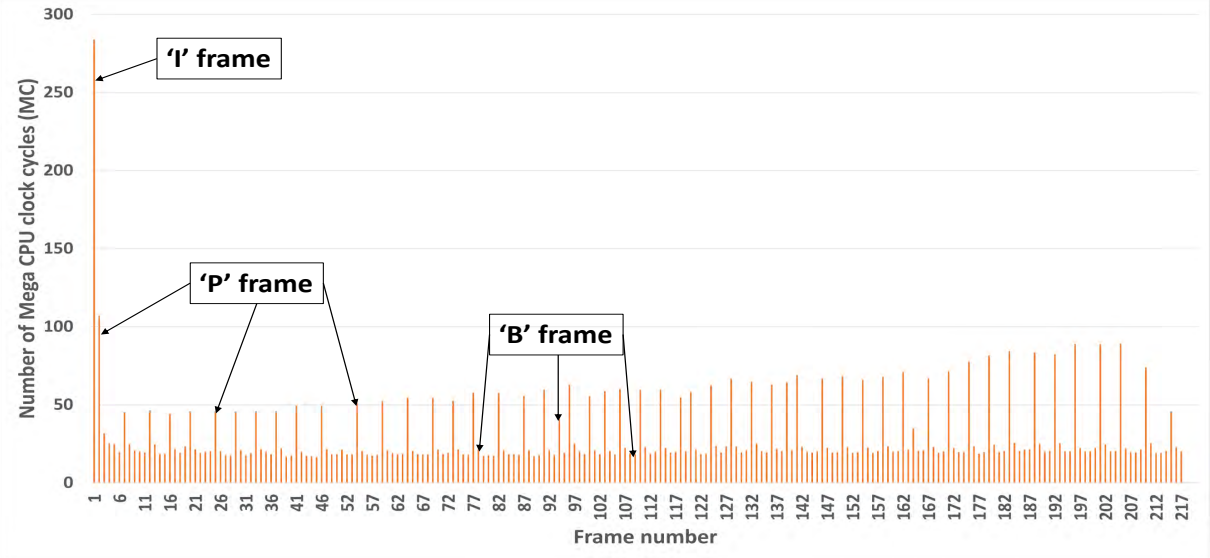


Figure 5.8 – "Blue-sky" video sequence frames complexity

decoded [6].

Then, after decoding five frames, the display process starts receiving frames, and thus the PI controller starts monitoring the output buffer size. The controller tries to maintain the output buffer size to the desired value (5 frames). Once the desired value was reached, the GPP clusters were slowed down to 1.3 GHz and 1.0 GHz for big and LITTLE GPP clusters, respectively. The time to switch between frequencies is assumed to be negligible. Some processors stall for as much as 50-100 µsec., but this is less than 1% of D_{period}. Newer processors may reduce or eliminate this latency [41].

Then, the output buffer size was almost stable between 4 and 5 frames, i.e., reached its stability after decoding around 17 frames, i.e., before one second of decoding. The fluctuation of size between 4 and 5 is due to the variation of frames complexity, see Fig.5.8. For instance, when a complex frame is decoded and takes more time than the deadline, the buffer size decreases to 4. Then, if a much less complex frame is decoded and takes a small amount of time (less than the deadline), the buffer size increases again to 5.

Next, after decoding around 135 frames, the output buffer size jumps to 7. Accordingly, the GPP clusters were sped up by increasing their frequency to 1.5 GHz and 1.1 GHz for

6. The number of frames to decode before to start sending to the display process is a configuration parameter. In this example, it is set to 5.

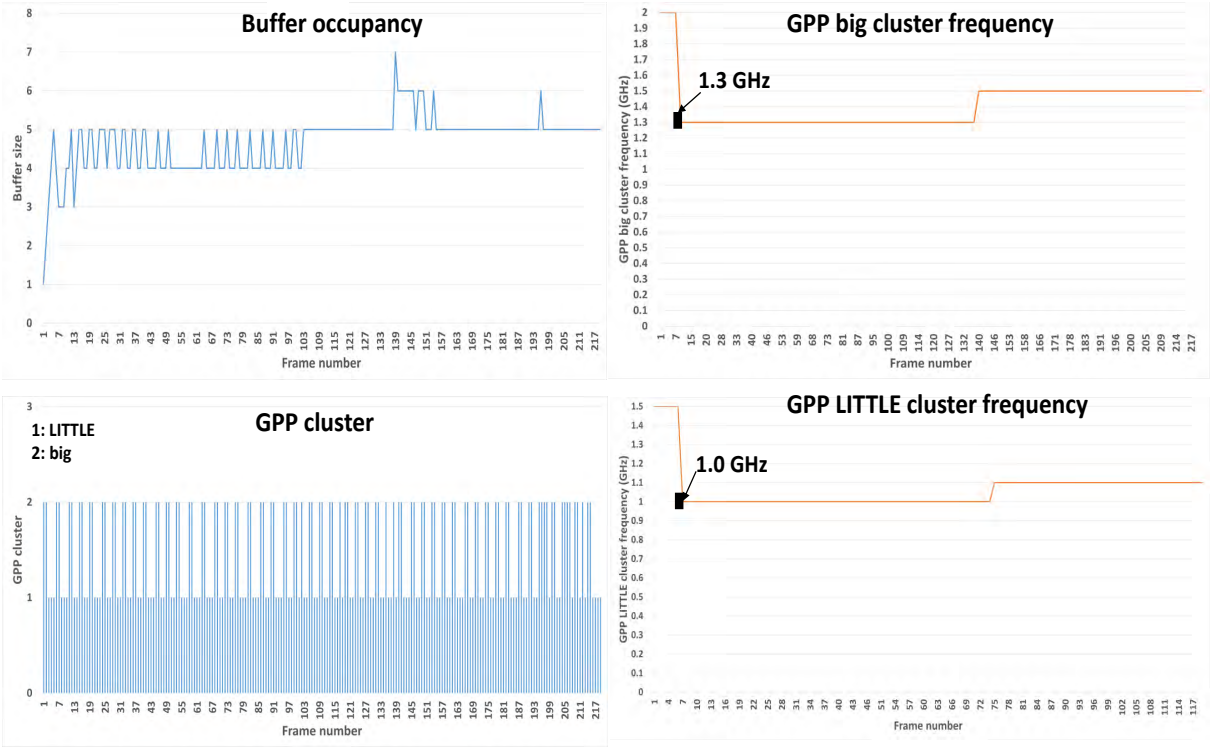


Figure 5.9 – PI controller analysis

big and LITTLE GPP clusters, respectively, in order to decrease the output buffer size to the desired value. From that point, the output buffer size was more stable until the end of decoding.

Fig.5.10 illustrates the PI controller behavior when changing the set point during the decoding process. The set point was changed after decoding 100 frames. At the beginning, it was set to 3, and then to 7. Other parameters were the same as those of Fig.5.9.

First, the GPP clusters started decoding at their highest speed in order to fill the output buffer. Once the frames started to be sent to the display process, the PI controller starts monitoring the output buffer size. For instance, at the 9^{th} frame, the big GPP cluster clock frequency was slowed down from 2.0 GHz to 1.6 GHz. As a consequence, the output buffer size decreased form 5 to 4. Then, it started to fluctuate between 4 and 3 and between 3 and 2. This fluctuation is due to the frames complexity variation, see Fig.5.8.

After decoding 100 frames, the set point was changed form 3 to 7. To fill the output buffer as fast as possible, the GPP clusters were sped up by clocking them to their highest supported frequency. The PI controller keeps filling the output buffer without sending the

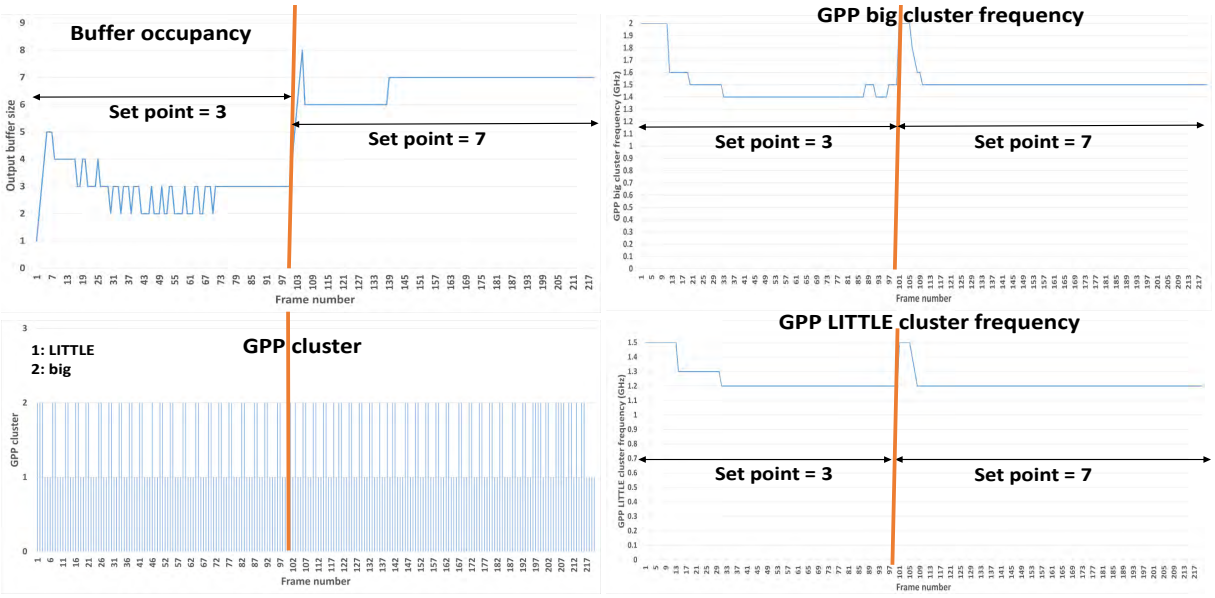


Figure 5.10 – PI controller stability when changing the set point

decoded frames to the display process momentarily.

Then, once the output buffer size overtook the set point, the GPP cluster were slowed down progressively until reaching the frequencies 1.5 GHz and 1.2 GHz for big and LITTLE clusters, respectively. One can note that the GPP clusters frequency were stable until the end of decoding which is a condition to save energy.

Summary: To sum up, at the beginning of decoding, the GPP clusters are clocked at their highest supported frequency values. This allows to fill the output buffer as fast as possible to reach the desired output buffer size. Then, the display process starts receiving frames, and thus the PI controller starts monitoring the output buffer size.

The stability of this latter was reached after decoding 17 frames, i.e., less than one second of decoding (video frame rate is 25 Hz). The fluctuation of output buffer size, in Fig.5.9, between 4 and 5 or between 5 and 6 is due to the variation of frames complexity. Therefore, the PI controller tries to maintain the output buffer size at the desired value as long as possible to reduce the energy consumption.

5.4.3 Overhead of the proposed solution

The overhead of the proposed solution is evaluated here. It is calculated using Equation (5.7) which is the ratio of the time necessary to execute phases 2 and 3 with respect to

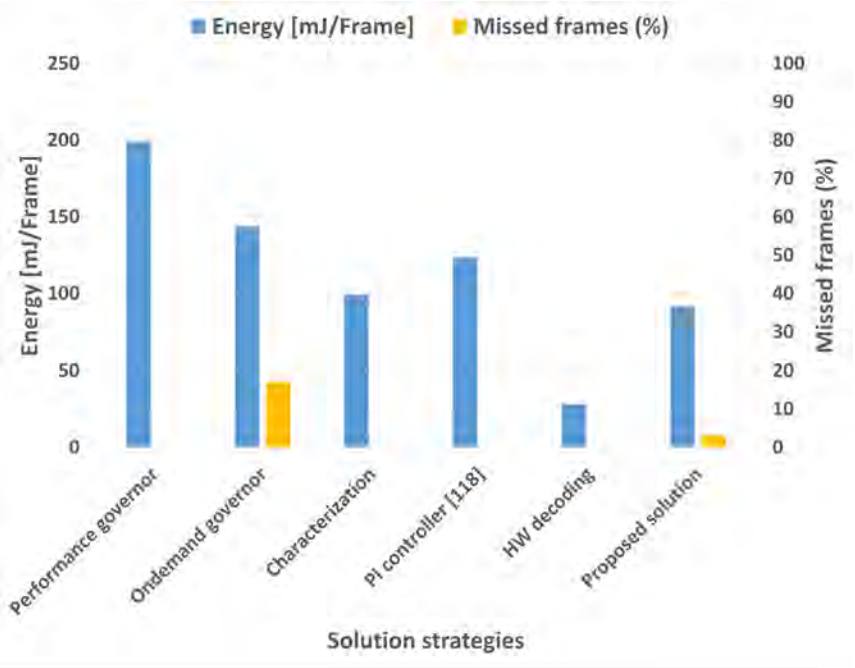


Figure 5.11 – The energy consumption (mJ/Frame) proposed solution vs the state-of-the-art work (on Snapdragon 810 and Odroid-xu3)

the time required to decode a frame.

The results show that the overhead represents on average less than 1% of the decoding time. That is, it is negligible compared to the gain of energy that the proposed solution permits to get.

5.4.4 Comparison with the state-of-the-art work

This section presents the results of comparison between the proposed solution and the state-of-the-art work, as illustrated in Table 5.3. The video decoding consumed energy is calculated using Equation (3.11) and is given in mJ/Frame. *Blue-sky* video sequence is taken as an example. Experiments were conducted on other video sequences as well and the curves show similar trends.

Fig.5.11 plots the results obtained on Snapdragon 810 (HW video decoding) and Odroid-xu3 (SW video decoding solution). First, setting the Performance Linux governor, the SW decoder consumes the most energy among the other presented solutions. Actually, this governor runs the GPP cores at their highest supported frequency values all the time. In addition, when a frame is decoded before the deadline, the GPP enters

idle state. Therefore, Performance governor introduces two causes of energy consumption: high processing frequency and GPP wake-up from idle state to active one to decode the next frame. The proposed solution mitigates these two drawbacks by adjusting the GPP frequency and eliminating the idle time. This enables around 53% of energy saving (mJ/Frame) when decoding *Blue-sky* video test sequence. On average, the proposed solution can save 40% of energy compared to the Performance Linux governor solution. Obviously, when the Performance Linux governor is set up, there is no miss-rate since the GPP cores perform decoding at their highest speed. Though the proposed solution adjusts the GPP frequency, it induces less than 1% miss-rate thanks to the classification and feedback control techniques.

Second, the Ondemand Linux governor tries to adjust the GPP frequency over the time. Updating the GPP frequency is based on information related to the GPP workload. In contrast, the proposed solution adjusts the GPP frequency based on information related to the target application: video decoding. This specificity permits to save up to 36% of energy (mJ/Frame) when decoding *Blue-sky* video test sequence. On average, the proposed solution can save 30% of energy compared to the Ondemand Linux governor solution. Then, regarding the frame miss-rate, when the Ondemand governor is set up, the miss-rate is on average more than 10%.

Third, the characterization solution selects statically the best configuration: number of cores and their operating frequencies. Statically means that one needs to test all configurations to select the best one which satisfies the decoding real-time constraint and consumes the least energy possible. However, once a frequency is selected, it remains constant during all the decoding process. This implies that, for instance, a simple frame may be decoded by the big GPP cluster at a higher frequency than what is required. The proposed solution dynamically tries to keep the GPP frequency as stable as possible while affecting the right GPP cluster according to the complexity of the frame to decode. This small advantage enables on average 7% of energy saving (mJ/Frame) over the characterization solution.

Fourth, the proposition solution was compared to the solution based only on the feedback control technique, i.e., without using the classification technique. The controller tries to adjust the GPP frequency based on the output buffer size. The GPP cluster is selected alternatively between the big and LITTLE ones. In contrast, the proposed solution selects the GPP cluster based on the complexity of the frame decode. This results in saving 25% of energy (mJ/Frame) when decoding *Blue-sky* video test sequence. On average, the

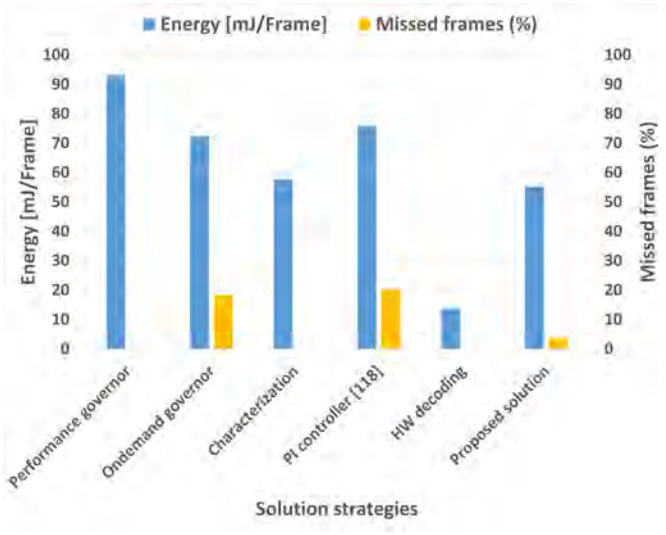


Figure 5.12 – The energy consumption (mJ/Frame) of the proposed solution vs the state-of-the-art work (on RB3)

proposed solution can save 20% of energy thanks to the classification technique.

Finally, the HW video decoding solution consumes the least energy among the presented results. Compared to a state-of-the-art study [22] where it is stated that dedicated processors outperform the GPPs by around 1000× in terms of energy efficiency, from the system level point of view, the proposed solution consumes on average only 3× more energy (mJ/Frame). This ratio confirms the results obtained in our characterization work.

The above presented results came from experiments which were conducted on two different platforms (Snapdragon 810 and Odroid-xu3). This can give an idea about the trends when comparing the proposed solution with the state-of-the-art work. To validate these trends, other experiments were conducted on a single platform (RB3), as depicted in Fig.5.12.

As a general note, the energy consumption values on RB3 are lower than those on Snapdragon 810 and Odroid-xu3. One cause of this behavior is that RB3 consumes less dynamic power. Indeed, it is equipped with a SoC fabricated at a lower technology process (10 nm) than that of the Snapdragon 810 and Odroid-xu3 platforms. Also, both curves (Fig.5.11 and Fig.5.12) show similar trends.

Fig.5.12 shows that, when decoding *Blue-sky* video test sequence, the proposed solution can save up to 40% and 23% of energy (mJ/Frame) compared to the Performance Linux governor, Ondemand Linux governor, respectively. Then, some 4% of energy can be saved compared to the characterization solution. The classification allows to save up to 27%

of energy (mJ/Frame) compared to the sate-of-the-art work which does not include this technique. Regarding the HW video decoding, the proposed solution consumes only 4× more energy (mJ/Frame).

On average, over the tested video sequences, the proposed solution can save 35% and 20% of energy (mJ/Frame) compared to the Performance Linux governor, Ondemand Linux governor, respectively. Then, some 4% of energy (mJ/Frame) can be saved compared to the characterization solution. The classification permits to save 23% of energy (mJ/Frame) compared to the sate-of-the-art work which does not include this technique. Concerning the HW video decoding, the proposed solution consumes on average 4× more energy (mJ/Frame).

Table 5.4 summarizes the energy saving percentage of the proposed solution over state-of-the-art work.

Summary: To sum up, in case of Snapdragon 810 and Odroid-xu3 tested platforms, the proposed solution can save on average 40% and 30% of energy (mJ/Frame) compared to the Performance and Ondemand Linux governors, respectively. Then, the proposed solution not only determines dynamically the right GPP cluster and its clock frequency, in contrast to the characterization solution, but also can save up to 7% of energy (mJ/Frame). The classification technique, phase 2, brings on average 20% of energy saving. Finally, the ratio of energy between the proposed solution and the HW video decoding is only about 3×.

To validate these results, experiments were conducted on RB3 platform which supports both HW and SW HEVC decoding. On this platform, the proposed solution can save up to 35%, 20%, 4%, and 23% of energy (mJ/Frame) compared to the Performance Linux governor, the Ondemand Linux governor, the characterization work, and the solution without the classification technique, respectively. Concerning the HW video decoding, the proposed solution consumes on average only 4× more energy (mJ/Frame).

On all tested platforms, the miss-rate represents on average less than 5%. In addition, our solution needs less than one second of decoding to enter the stable state of the output buffer size and thus the GPP frequency, as suggested by Jensen's inequality in [1]. Finally, regarding the overhead of the proposed solution, it represents on average less than 1% of the decoding time.

Table 5.4
The proposed solution energy saving (%) over state-of-the-art work

State-of-the-art work		Open-HEVC + DVFS (Performance Linux governor)	Open-HEVC + DVFS (Ondemand Linux governor)	Best configuration according to our characterization work (Chapter 3)	Open-HEVC + DVFS (PI controller) [41]
Average of the proposed solution energy saving (%)	On Snapdragon 810 and Odroid-xu3 platforms	40	30	7	20
	On RB3 platform	35	20	4	23

5.5 Conclusion

This chapter presents a solution to reduce the energy consumption of HEVC decoding on a heterogeneous mobile platform. The proposed solution is split into three phases: (i) modeling of frame complexity, (ii) assignment of frames to appropriate GPP cores (using classification), and (iii) frequency scaling using a PI controller with DVFS.

In the first phase, a model is built. The objective of this model is to classify any given video frame into two groups: (i) most complex frames, and (ii) least complex frames. The input parameters are: frame type and frame size. The configuration parameters are: video bitrate, video frame rate, and the ratio of performance between the big and LITTLE GPP cluster. In the second phase, the video frames are scheduled between big and LITTLE GPP cluster using the model built in phase 1. In the third phase, the frequency of the selected GPP cluster is adjusted using a PI controller. This latter monitors the output buffer size in order to maintain it in a certain value which is an input parameter of the controller.

The established model in phase 1 is more than 90% accurate. This accuracy permits to exploit efficiently the heterogeneous character of mobile architectures, such as ARM big.LITTLE.

The results show that the proposed solution can save up to 36% of energy (mJ/Frame) compared to the Ondemand Linux governor. Also, compared to the the characterization work, the proposed solution can save on average some 7% of energy. The main advantage over this solution is the dynamic character that adjusts the GPP frequency during run-

time. Actually, the characterization work selects the frequency by testing all possible configurations which is not realistic in real-world scenario. Finally, the proposed solution consumes on average only 4× more energy than the HW decoding.

Moreover, the classification has a great role to exploit the heterogeneity of ARM big.LITTLE architecture and limit the miss-rate. Actually, the proposed solution induces less than 5% of miss-rate, whereas 10% of miss-rate is observed when the Ondemand Linux governor is set up.

In terms of output buffer size stability, the proposed solution needs less than one second of decoding to enter in a stable state. Indeed, once the stable state is reached, the GPP frequency changed only twice.

Finally, the proposed solution is very slight as it represents on average less than 1% of the decoding time.

Chapter 6

Conclusions & future directions

6.1 Conclusions

This section concludes the thesis. It briefly reminds the context in which this work was conducted. Then, the research questions that were addressed are presented, followed by their answers.

The present thesis addressed the issue of video decoding energy consumption on low power heterogeneous mobile SoCs. It was motivated, on the one hand, by the context of the ever growing demand for mobile video content and what this implies in terms of computational demands, and, on the other hand, by the limited battery life-time which does not evolve as fast as video content does. Also, the HW video decoding (using a dedicated processor, e.g., an HDIP) and the SW video decoding (using a heterogeneous multi-cores GPP) are in a rude competition where the former lacks flexibility whereas the latter gains this feature but still suffers from energy budget constraints.

Specifically, we addressed four research questions described below:

1. RQ1: How relevant are the video decoders (HW and SW) with regards to the parameters impacting the video decoding process?
2. RQ2: To what extent do HW video decoders (HDIPs) outperform SW ones?
3. RQ3: How to balance the video content workload among the heterogeneous GPP cores, based on a little amount of information on the video to decode?
4. RQ4: Finally, once the GPP cores are selected, how to adjust their frequency in order to reduce the energy consumption?

The answers of the above research questions led to two main contributions. First, the thesis consisted in understanding the performance and energy consumption of the HW and SW video decoding by characterizing them. Second, it consisted in proposing mechanisms and strategies for optimizing the energy consumption of SW video decoding.

In the first contribution, the research questions (RQ1 and RQ2) were addressed. The proposed methodology is based on extensive performance and energy consumption measurements. The experiments were carried out on real-world platforms representing different processing configurations, including: (a) mono core GPP processor, (b) multi-cores GPP processor, and (c) HW video decoder (HDIP). This allowed to reflect real life scenarios. In addition, experiments were conducted on three different platforms which represent mobile phones of different generations. Using different platforms allows to have a clearer understanding of energy consumption of video decoding. Also, it permits an objective comparison between the HW and SW video decoding energy consumption. Finally, it makes the obtained results generalizable to other platforms not tested in our work. The proposed characterization methodology is described in Chapter 3. Then, the results of the methodology applied on HEVC codec are presented and analyzed in Chapter 4.

RQ1 questioned about the parameters impacting the performance and energy consumption of video decoding in both implementations: HW and SW. The parameters are triggered at two levels: (i) operating system (OS) level, and (ii) application level. The answer is two-fold. First, at the OS level, the proposed characterization methodology tests all the possible configurations (number of GPP cores and their clock frequency) and measures the consumed energy for each one. Then, it selects the configuration which consumes the least energy. Second, at the application level, the video parameters: bitrate, frame rate, and resolution are studied.

The results show that, on the tested platforms, decoding with one big GPP core and four LITTLE GPP cores consumes on average 2× less energy (mJ/Frame) than decoding with two big GPP cores. Then, the studied parameters at the application level have been shown correlated with the performance and energy consumption of HEVC decoding. This confirms that HEVC decoding is similar to its predecessors in terms of energy consumption with respect to the above parameters. RQ1 is solved in Chapter 3 and the results are presented and analyzed in Chapter 4.

RQ2 questioned about the ratio of energy consumption between the HW and SW video decoding. The answer depends on the level at which the study is carried out as well as the parameters taken into account to make the comparison. Actually, comparing the HW and SW video decoding energy consumption at the processor level is different than the same comparison but at the system level. For instance, the system overhead was evaluated and its impact on the overall performance and energy consumption was analyzed.

The results show that, on the tested platforms, for low video resolutions (e.g., 720p)

and at the system level point of view, the SW video decoding consumes as much energy as the HW one. However, for high video resolutions (e.g., 2160p), the HW video decoding is more suited as it consumes less energy. Moreover, at the system level, the GPP when performing video decoding consumes at most on average 10× more energy than the HDIP whereas, in the state-of-the-art work, the ratio at the processor level is approximately 1000×. As a consequence, objective comparison should take into account multi-levels parameters in order to have a broaden understanding of the performance and energy consumption of video decoding on mobile platforms. RQ2 is solved in Chapter 3 and the results are presented and analyzed in Chapter 4.

In the second contribution, the research questions (RQ3 and RQ4) were addressed. The proposed optimization solution as well as the obtained results are described and analyzed in Chapter 5.

RQ3 questioned about the possibility of exploiting the heterogeneity of mobile GPP to balance video frames among its cores based on a small knowledge of these frames characteristics. Our proposed solution is based on the classification technique. Thanks to this technique, the frames can be classified into two groups: (i) most complex frames, and (ii) least complex frames, so that the first group of frames should be decoded by the high performance cores (e.g., big GPP cluster) and the second group by the energy-efficient cores (e.g., LITTLE GPP cluster). The classifier takes only two information about the frame to decode: its type and its size. RQ3 is solved in Section 5.2.2 of Chapter 5.

Once a GPP cluster is selected thanks to classification, RQ4 questioned about a mechanism that allows to adjust its clock frequency. The proposed solution reused a state-of-the-art work which is based on the feedback control technique (PI controller). The controller monitors the output buffer size so that it maintains a certain value which is a parameter (set by the user). This technique allowed to maintain the output buffer size and thus the GPP frequency stable which is a recommendation of ARM to save energy on big.LITTLE-based architecture SoCs. RQ4 is solved in Section 7.4.4 of Chapter 5.

The results show that, on the tested platforms, the proposed solution can save on average more than 20% of energy compared to the Ondemand Linux governor. In addition, the proposed solution consumes on average only 4× more energy (mJ/Frame) the HW video decoding which confirms the results found in our first contribution. Moreover, the classification allows to exploit the heterogeneity of ARM big.LITTLE architecture by limiting the miss-rate. Actually, the proposed solution induces less than 5% of miss-rate, whereas Ondemand Linux governor induces more than 10% of miss-rate. In terms of

buffer occupancy, the proposed solution needs less than one second of decoding to enter in a stable state. Finally, the proposed solution is very light as it represents on overage less than 1% of the frame decoding time. As a consequence, the answer of this question, i.e., the proposed solution, can easily be integrated into mobile platforms to optimize the battery life-time.

In short, the first contribution allows to reduce the energy consumption of video decoding while satisfying the real-time constraint in a static fashion, i.e., by testing all the possible configurations. This scenario is useful in a limited use cases, such as video-on-demand (VoD). The second contribution achieves the same purpose (performance and energy consumption) but in a dynamic fashion. It exploits the heterogeneous character of mobile platforms, e.g., ARM big.LITTLE, using the classification technique.

In conclusion, although the video decoding energy consumption issue is still actively considered by the scientific community (in academia and industry), this thesis contributed to this effort. It helped enhance the understanding pf the energy consumption of HW and SW video decoding on low power heterogeneous mobile SoCs as well as bridge the gap between these two approaches. Finally, this thesis raised open questions that will be discussed in the next section.

6.2 Future directions

The work presented in this thesis enhanced the understanding of HEVC HW and SW decoding performance and energy consumption. To further continue the research on this topic, several possible future research directions which need deeper investigations are proposed in this section.

6.2.1 Application on other video codecs

As explained in Chapter 3, the proposed methodology to characterize the performance and energy consumption of video decoding is independent of any specific video codec. In this thesis, the characterization methodology was applied on HEVC codec. Therefore, it can easily be extended to newer video codecs, e.g., VVC, and without any change. Indeed, this latter belongs to the MPEG video codecs family whose the main principles are explained in Section 2.1.1.

Regarding the optimization proposed in Chapter 5, it can also be applied to VVC.

In fact, despite the new improvements introduced into the new standard, the proposed optimization stays applicable since it considers the video codec as a black box.

6.2.2 Display system

The display system is considered as beyond the scope of this thesis. However, we think that this system should be investigated and then be included in any video application study. Indeed, there is no video decoded without being displayed. In the state of the art, many models exist to estimate the energy consumption of video decoding without taking into account the display system. However, one cannot apply these models in real-world scenarios as in this case the display system is present.

According to the state of the art [149], the energy consumption of displaying a video on mobile screen depends on parameters related to both the video sequence itself and the technology used in the fabrication of the screen. For instance, recent Active Matrix Organic Light Emitting Diode (AMOLED) technology consumes less energy per pixel than its predecessor LCD technology [150]. Moreover, the display is very dependent on the video to decode in terms of brightness, colors, etc. Therefore, we think that including the display system to the video application energy consumption study may give a complete video playback understanding.

6.2.3 Per-core DVFS

In the proposed work, the number of cores and their clock frequencies should be carefully configured in order to save energy. By doing so and exploiting load balancing among heterogeneous GPP cores, the results show great energy savings of SW video decoding allowing to reduce the gap with the HW one. It should be recalled that, in our work, all the cores of a GPP cluster are set to the same frequency.

In our opinion, a per-core DVFS scaling over heterogeneous GPP cores is a promising technique to save energy of parallel SW video decoding. Indeed, the partitions composing a frame (slices, tiles, or WPP) are decoded in parallel at the GPP core level, i.e., each GPP core decodes a specific partition. In addition, these partitions have different complexities. Therefore, it is relevant to adjust the GPP core frequency to correspond to the complexity of the partition to decode. Technically, recent platforms support this feature. This work is planed to be one of our next studies.

6.2.4 Analytical tuning of the PI controller coefficients

In the proposed solution to optimize the SW video decoding energy consumption, a PI controller is used to adjust the GPP frequency. The coefficients of the controller are tuned empirically, and before to start decoding. We think that those coefficients can be calculated during run-time using appropriate techniques, such as reinforcement learning. We think that this is feasible as the ratio gain/overhead of applying reinforcement learning on embedded systems has been shown to be acceptable in [151].

The reinforcement learning technique allows the controller to adjust its coefficients to correspond the best to the video frames workload, by trial and error, using feedback from its own actions and experiences. One more advantage of this technique is that it does not need a prior knowledge of the video frame to decode. The objective is to provide to the video decoder some auto-learning capabilities for predicting the upcoming workload based on the workload history.

6.2.5 Three-clusters architecture

The proposed optimization was applied on ARM big.LITTLE architecture. It classifies any given frame into two groups : (i) the most complex frames, and (ii) the least complex frames. Then, the frames of the first group are submitted to big cores whereas the others are submitted to LITTLE cores.

The proposed solution can also be applied on big.Medium.LITTLE architecture. In this case, any given video frame is classified to three groups : (i) the most complex frames, (ii) the medium complex frames, and (iii) the least complex frames. The frames of the first group are submitted to big cores, the frames of the second group to Medium cores, and the frames of the third group to LITTLE cores. This architecture shows an important advancement in terms of performance and energy consumption. The authors in [152] [153] show that the architecture composed of three clusters is more energy-efficient the 2-clusters one when the program is well designed to exploit the parallelism among the heterogeneous cores.

6.2.6 Programming models

According to [154], the programming models OpenMP or OmpSs have an impact on the performance and energy consumption on ARM big.LITTLE-based architecture.

Therefore, one should find and then tune the relevant parameters to obtain the best trade-off between performance and energy.

As explained in Section 2.1.1, HEVC codec is a parallel friendly process. We think that re-designing the parallelism of the video decoding framework (e.g., Open-HEVC) will allow to gain further energy consumption.

Chapter 7

Résumé substantiel en français

7.1 Introduction

Dans cette section, le contexte qui nous a motivé à travailler sur le problème de la consommation énergétique du décodage vidéo sur les plateformes mobiles est présenté. Ensuite, la problématique est posée. Dans la suite, la portée et le positionnement de notre travail par rapport à ceux de l'état de l'art sont définis. Finalement, le plan de ce résumé est décrit. À noter que la thèse s'intègre dans le cadre du projet FUI-23 EFIGI [1].

7.1.1 Contexte

Les vidéos mobiles sont en constante évolution en termes de volume et de facilité d'accès. En effet, selon Cisco [2], en 2022, les vidéos sur Internet représenteront plus de 82% de l'ensemble du trafic y circulant. 75% de ces vidéos sont vues sur des plateformes mobiles [3] dont 80% sont équipées d'écran de résolution 1080p ou moins [4].

Les auteurs de [6] expliquent cette prolifération par certains facteurs : partage de vidéos via les réseaux sociaux, activité solitaire immersive (e.g., dans les transports publics), etc. En outre, les spectateurs retiennent 95% du contenu d'un message lorsqu'ils le regardent sur une vidéo, contre 10% lorsqu'ils le lisent dans un texte [7].

En réponse au besoin croissant en applications vidéo, les groupes Moving Picture Experts Group (MPEG) et l'ITU-T Video Coding Experts Group (VCEG) ont conjointement développé plusieurs codecs au cours des deux dernières décennies : H.262/MPEG-2 [10], H.264/AVC [11][12], H.265/HEVC [13], et H.266/VVC [14]. Chaque codec vise à réduire le *bitrate*, et donc la taille du fichier vidéo, par rapport à son prédécesseur, tout en conservant la même qualité visuelle.

Un défi important résultant de ces nouvelles tendances dans l'architecture matérielle (HW) est la maîtrise de la consommation d'énergie. En fait, chaque nouveau codec utilise

1. EFIGI : Efficient Future & nEw Generation vIdeo coding

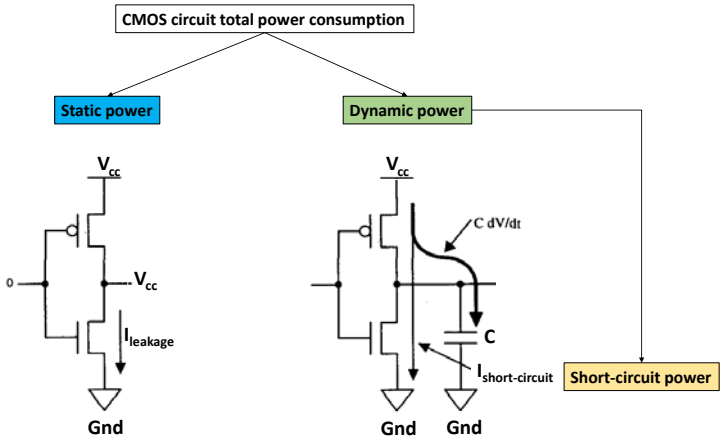


Figure 7.1 – Décomposition de la puissance consommée [67]

des outils plus avancés que son prédécesseur. Ces outils avancés, d'une part, augmentent l'efficacité du codage et diminuent ainsi la bande passante et, d'autre part, rendent les algorithmes de compression de plus en plus complexes et agressifs [17].

7.1.2 Consommation d'énergie dans les circuits électroniques CMOS

La consommation d'énergie (en joules) d'un circuit électronique CMOS est la quantité de puissance P consommée pendant un temps T.

$$E = \int_0^T P(t)\, dt \tag{7.1}$$

où P est la puissance consommée (en Watt) par le circuit à l'instant t.

La puissance totale d'un circuit (P) est la somme de la puissance statique (P_{static}), de la puissance dynamique (P_{dyn}), et de la puissance de court-circuit (P_{short}), voir la figure 7.1. Ceci peut être traduit par l'équation suivante :

$$P = P_{\text{static}} + P_{\text{dyn}} + P_{\text{short}} \tag{7.2}$$

Tout d'abord, la puissance statique est la puissance consommée à l'état où il n'y a aucune activité au niveau du circuit électronique CMOS [68]. La puissance statique

est donc consommée indépendamment de tout programme en cours d'exécution dans le système.

La puissance dynamique est la puissance consommée lorsqu'il y a une activité, par exemple, lorsque le GPP exécute un programme. Elle est calculée comme suit :

$$P_{\text{dyn}} = C_{\text{eff}} * V^2 * f \tag{7.3}$$

où C_{eff} représente la capacité effective du circuit et V la tension d'alimentation associée à la fréquence d'horloge f [29].

Enfin, la composante P_{short} de l'équation (7.2) capture la puissance résultant d'un courant de court-circuit. Elle est considérée comme une petite partie de la puissance dynamique totale car elle est liée à l'activité du circuit [67][72]. Par conséquent, elle est négligée dans le reste de ce manuscrit.

Dans la suite du manuscrit, la consommation de puissance totale d'un circuit CMOS est donnée par l'équation suivante :

$$P = P_{static} + P_{dyn} \tag{7.4}$$

7.1.3 Techniques de réduction de la consommation d'énergie du décodage vidéo

Pour réduire la consommation énergétique du décodage vidéo, deux mécanismes sont principalement utilisés : (i) le parallélisme, et (ii) l'adaptation du couple tension-fréquence (DVFS).

Parallélisme

Le parallélisme du décodage vidéo sur les processeurs multi-cœurs peut être réalisé au niveau de l'image (trame), de la tranche (slice), ou du macro-block (MB) [83]. Le codec HEVC propose aussi du parallélisme par tuiles et par front d'onde (WPP) [31].

Le parallélisme doit être exploité avec précaution. En effet, plus la taille des blocs à paralléliser est petite, plus le surcoût de communication est important.

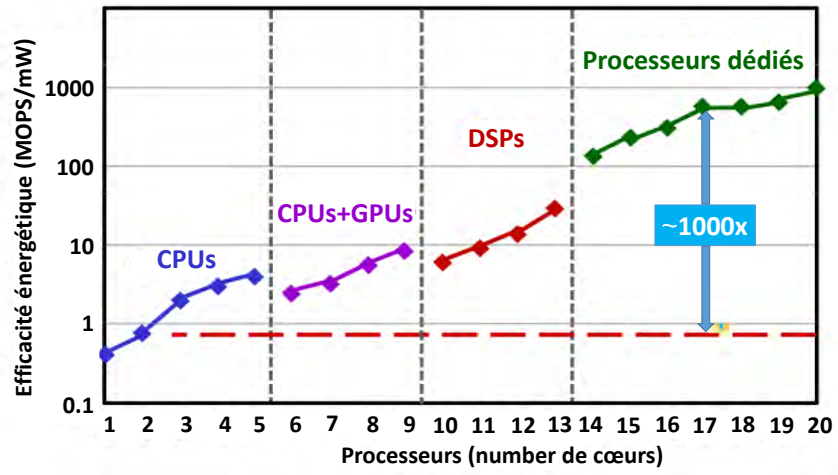


Figure 7.2 – Comparaison des processeurs en termes d'efficacité énergétique [22]

DVFS

Selon les équations (7.1) et (7.3), et, par souci de simplicité, la fréquence du processeur est considérée comme proportionnelle à la tension (c'est-à-dire $V \propto f$) comme supposé dans [73][74]. L'énergie dynamique est alors :

$$E_{dyn} = K * f^3 * t \tag{7.5}$$

où K est une constante, f est la fréquence du processeur, et t est le temps d'exécution. En utilisant cette équation, si, par exemple, la fréquence est divisée par 2, le temps d'exécution, t, peut être doublé mais l'énergie, E_{dyn}, est divisée par 4.

7.1.4 Décodage matériel (HW) vs décodage logiciel (SW)

La figure 7.2 illustre une comparaison entre différents types de processeurs : GPP, GPU, DSP, et les circuits dédiés, e.g., HW Decoder Intellectual Property (HDIP), en termes d'efficacité énergétique (millions d'opérations par seconde / mW). Le graphique montre deux points importants. Premièrement, les processeurs spécialisés contiennent plus de cœurs que les processeurs à usage général, ce qui les rend plus puissants. Deuxièmement, plus un processeur est spécialisé, plus il est économe en énergie. En outre, tous les types de processeurs comparés, dans la figure 7.2, sont capables d'effectuer le décodage vidéo, seuls ou avec l'aide du GPP.

Décodage HW

Une solution proposée pour réduire la consommation d'énergie du décodage vidéo tout en offrant des performances élevées consiste à utiliser un décodage vidéo matériel exécuté par un processeur dédié (désigné HDIP dans ce manuscrit). Les processeurs spécialisés offrent une combinaison optimale de performances et d'efficacité énergétique. Par exemple, dans l'état de l'art, un processeur spécialisé est dit être énergétiquement 1000× plus efficace qu'un processeur généraliste (désigné GPP dans ce manuscrit) [22]. Ceci est possible grâce à l'utilisation d'unités de traitement spécialisées [32] qui éliminent la consommation d'énergie liée au décodage des instructions et à la logique de contrôle caractérisant les GPPs [33]. Par conséquent, la plupart des smartphones modernes sont équipés d'un HDIP qui consomme moins d'énergie en satisfaisant la contrainte de décodage vidéo en temps réel [23][24].

En revanche, les HDIPs ne sont pas flexibles, c'est-à-dire qu'ils ne peuvent pas supporter d'autres solutions que celles déjà implémentées. De plus, le temps de mise sur le marché est plus long qu'une solution logicielle (SW). Enfin, un HDIP est considéré par le GPP comme un périphérique. De ce fait, une attention particulière doit être portée à la communication inter-processeurs (IPC en anglais) entre les deux types de processeurs.

Décodage SW

Le décodage SW s'effectue sur un GPP. Une caractéristique importante d'un GPP est qu'il permet de développer et de déployer rapidement une large gamme d'applications, notamment de nouveaux codecs vidéo.

Ensuite, les nouvelles avancées dans les architectures des GPP multi-cœurs hétérogènes intégrés dans les plateformes mobiles offrent une grande opportunité d'améliorer à la fois les performances et l'efficacité énergétique du décodage vidéo (SW). En fait, l'exploitation du traitement parallèle entre les cœurs disponibles permet de réduire la fréquence d'horloge de fonctionnement requise, ce qui diminue considérablement la puissance dynamique consommée [30]. Dans le cas du décodage vidéo, l'hétérogénéité du GPP peut être exploitée pour décoder en parallèle les trames (ou les parties des trames) selon les schémas de parallélisme supportés par chaque codec.

7.1.5 Problématique

Dans ce contexte, les questions de recherche (QR) auxquelles nous avons essayé de répondre dans cette thèse sont formulées comme suit :

1. QR1 : Quelle est la performance des décodages vidéo (HW et SW) par rapport aux paramètres ayant un impact sur le processus de décodage vidéo ?
2. QR2 : Dans quelle mesure le décodage vidéo HW est-il plus performant que le décodage SW ?
3. QR3 : Comment équilibrer la charge vidéo entre les cœurs GPP hétérogènes, sur la base d'une faible quantité d'informations sur la vidéo à décoder ?
4. QR4 : Enfin, une fois les cœurs GPP sélectionnés, comment ajuster leur fréquence afin de réduire la consommation d'énergie ?

7.1.6 Objectif et champ d'application

Les questions de recherche ont été traitées dans un certain cadre qui est décrit ci-dessous.

Tout d'abord, les questions de recherche ont été traitées sans prendre en compte les détails des modules de décodage vidéo. Autrement dit, le décodage vidéo est considéré comme une boîte noire.

Dans nos expérimentations, les architectures mobiles hétérogènes, par exemple l'architecture ARM big.LITTLE, sont ciblées, grâce à leur large intégration dans les plateformes mobiles récentes [38]. Dans le contexte d'une application de décodage vidéo, cette architecture s'avère particulièrement intéressante car la complexité des trames diffère considérablement d'une trame à l'autre.

Notre étude est réalisée à deux niveaux : (i) le niveau système d'exploitation (OS), et (ii) le niveau applicatif. L'architecture n'est pas variée puisque nous avons utilisé les deux architectures de microprocesseurs largement répandues sur les SoC mobiles : HDIP et GPP (ARM).

Grâce à la prédominance de HEVC sur AVC, il a été choisi comme codec principal sur lequel notre étude et nos expériences ont été menées. Il convient de noter que VVC n'a pas été choisi car il est sorti récemment en juillet 2020 [16], et les premières implémentations temps réel n'ont été disponibles qu'en 2021.Par conséquent, ce choix garantit l'actualité de notre travail.

7.1.7 Plan

Ce résumé reprend l'organisation des chapitres du corps de la thèse. La section 2 décrit la méthodologie proposée pour résoudre la problématique (QR1 et QR2). La section 3 est consacrée à la présentation des résultats de la caractérisation des performances et de la consommation d'énergie du décodage HEVC. La section 4 détaille la solution proposée pour optimiser la consommation énergétique du décodage HEVC. Enfin, la section 5 tire les conclusions de cette étude.

7.2 Caractérisation des performance et consommation énergétique du décodage vidéo

Dans cette section, nous traitons les questions de recherche (QR1 et QR2) de la problématique de la thèse. Premièrement, le modèle de puissance utilisé pour calculer la puissance/énergie consommée lors du décodage vidéo est défini. Deuxièmement, la méthodologie de caractérisation des performances et de la consommation d'énergie du décodage vidéo matériel et logiciel pour une analyse et comparaison objectives est détaillée.

7.2.1 Modèle de puissance

À partir de l'équation (2.5), qui formule la consommation totale de puissance d'un circuit CMOS, uniquement la puissance dynamique est évaluée. La raison est que, dans cette thèse, la consommation de puissance/énergie du décodage vidéo est étudiée et que la puissance statique est indépendante de toute activité dans le système.

Le modèle de puissance proposé se base sur une approche par décomposition. Cette approche permet d'évaluer séparément les différentes composantes de la puissance consommée.

Tout d'abord, nous considérons que la puissance dynamique d'un GPP est la somme des puissances active et inactive (*idle*), comme illustrée dans la figure 7.3. Ainsi, la consommation de puissance dynamique globale d'une plateforme, $P_{\text{glob\_dyn}}$, est formulée dans l'équation (7.6) [2] :

$$P_{\text{glob\_dyn}} = P_{\text{glob\_idle}} + \hat{P}_{\text{glob\_active}} \tag{7.6}$$

2. Dans le manuscrit, un symbole avec *circonflexe* indique une valeur (puissance, énergie, ou ratio) déduite d'autres puissances/énergies calculées qui sont obtenues à partir d'outils de mesure.

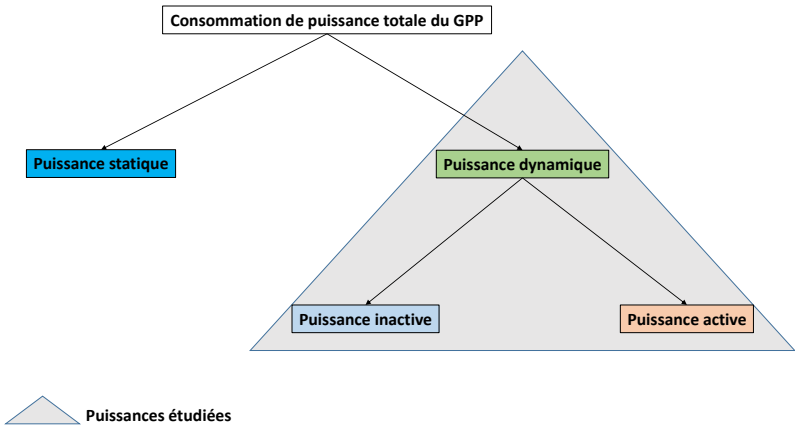


Figure 7.3 – Décomposition de puissance

où $P_{\text{glob\_idle}}$ correspond à la consommation de puissance globale de la plateforme à l'état de repos, et $\hat{P}_{\text{glob\_active}}$ est la consommation de puissance effective liée au décodage vidéo.

La puissance inactive est la puissance consommée à l'état où il n'y a pas de processus en cours d'exécution au niveau "utilisateur". Il convient de préciser que la puissance au repos est différente de la puissance statique.

En ce qui concerne la puissance active, il s'agit de la puissance consommée à l'état où il y a un ou plusieurs processus en cours d'exécution au niveau "utilisateur". Cette puissance est uniquement liée aux processus en cours d'exécution, c'est-à-dire sans tenir compte de la puissance inactive.

Chaque puissance (inactive ou active) est composée de deux éléments de puissance et formulée dans les équations suivantes :

$$P_{\text{glob\_idle}} = \hat{P}_{\text{GPP\_idle}} + \hat{P}_{\text{Other\_idle}} \tag{7.7}$$

$$\hat{P}_{\text{glob\_active}} = \hat{P}_{\text{GPP\_active}} + \hat{P}_{\text{Other\_active}} \tag{7.8}$$

où $\hat{P}_{\text{GPP}}$ et $\hat{P}_{\text{Other}}$ sont la consommation de puissance du GPP et des autres éléments présents dans la plateforme, respectivement.

Selon les équations (7.7) et (7.8), la puissance dynamique globale d'une plateforme peut être formulée comme la somme de la puissance dynamique du GPP, $\hat{P}_{\text{GPP\_dyn}}$, et celle des autres éléments présents dans la plateforme, $\hat{P}_{\text{Other\_dyn}}$. Ceci peut être traduit par l'équation suivante :

$$P_{\text{glob\_dyn}} = \hat{P}_{\text{GPP\_dyn}} + \hat{P}_{\text{Other\_dyn}} \tag{7.9}$$

Dans le cas d'une architecture GPP multi-cœurs hétérogènes, la consommation dynamique du GPP est formulée en fonction du nombre de cœurs qui exécutent un programme (par exemple, le décodage vidéo) et de ceux qui sont à l'état d'inactivité, pour chaque type de cœur. Par exemple, dans un SoC équipé d'une architecture ARM big.LITTLE, la consommation de puissance dynamique du GPP est donnée par l'équation suivante :

$$\begin{aligned}\hat{P}_{\text{GPP\_dyn}} = {} & nb_{\text{run\_big}} * (\hat{P}_{\text{GPP\_1\_core\_idle\_big}} + \hat{P}_{\text{GPP\_1\_core\_active\_big}}) \\ & + nb_{\text{not\_run\_big}} * \hat{P}_{\text{GPP\_1\_core\_idle\_big}} \\ & + nb_{\text{run\_LITTLE}} * (\hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}} + \hat{P}_{\text{GPP\_1\_core\_active\_LITTLE}}) \\ & + nb_{\text{not\_run\_LITTLE}} * \hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}}\end{aligned} \tag{7.10}$$

où $nb_{\text{not\_run\_big}}$, $nb_{\text{not\_run\_LITTLE}}$, $nb_{\text{run\_big}}$, et $nb_{\text{run\_LITTLE}}$ représentent le nombre de cœurs big et LITTLE à l'état de repos et actif, respectivement.
$\hat{P}_{\text{GPP\_1\_core\_idle\_big}}$, $\hat{P}_{\text{GPP\_1\_core\_idle\_LITTLE}}$, $\hat{P}_{\text{GPP\_1\_core\_active\_big}}$, et $\hat{P}_{\text{GPP\_1\_core\_active\_LITTLE}}$ représentent la consommation de puissance d'un cœur big et LITTLE à l'état de repos et actif, respectivement.

La figure 7.4 montre un exemple où $nb_{\text{run\_big}} = 1, nb_{\text{not\_run\_big}} = 3, nb_{\text{run\_LITTLE}} = 4$, and $nb_{\text{not\_run\_LITTLE}} = 0$.

Ensuite, la consommation de puissance d'un cœur GPP à l'état de repos, $\hat{P}_{\text{GPP\_1\_core\_idle}}$, est donnée par l'équation (7.11). Elle est déduite, à l'aide de l'équation (7.7), en soustrayant la consommation de puissance globale où un seul cœur GPP est allumé de celle où deux

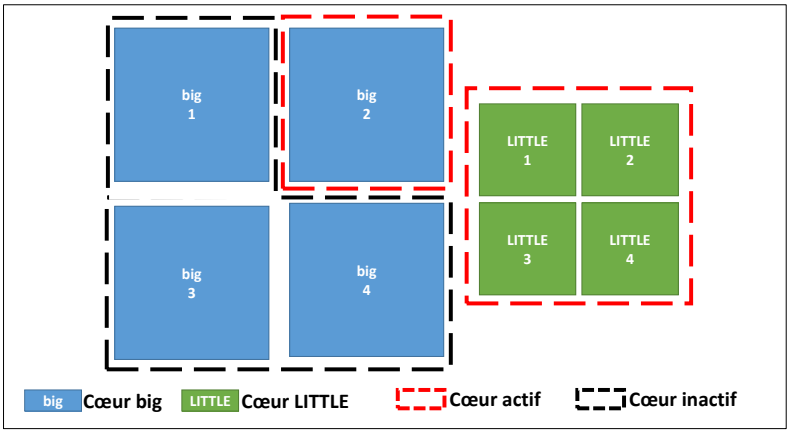


Figure 7.4 – Exemple d'une architecture GPP hétérogène (ARM big.LITTLE)

cœurs GPP sont allumés, tous à l'état de repos (idle).

$$\hat{P}_{\text{GPP\_1\_core\_idle}} \approx P_{\text{glob\_2\_cores\_idle}} - P_{\text{glob\_1\_core\_idle}} \tag{7.11}$$

où $P_{\text{glob\_2\_cores\_idle}}$ et $P_{\text{glob\_1\_core\_idle}}$ sont la consommation de puissance globale de la plateforme à l'état de repos lorsque seulement deux cœurs GPP et un cœur GPP sont allumés, respectivement. Nous supposons que $\hat{P}_{\text{Other\_1\_core\_idle}}$ et $\hat{P}_{\text{Other\_2\_cores\_idle}}$ sont équivalentes et la différence est donc arrondie à zéro. De plus, les cœurs GPP sont identiques et consomment donc la même quantité de puissance chacun.

En ce qui concerne $\hat{P}_{\text{Other}}$, de l'équation (7.9), elle inclut la puissance consommée par le HDIP. Elle inclut également la puissance due à la communication (IPC) lors du décodage vidéo matériel. En outre, elle inclut la puissance liée à la mémoire, par exemple, les transferts entre la mémoire vive (RAM) et le tampon interne du HDIP. Le terme $\hat{P}_{\text{Other}}$ comprend une petite quantité de puissance nécessaire pour communiquer avec un ordinateur personnel (PC) afin, par exemple, de recevoir des commandes de décodage vidéo. Nous considérons que cette quantité d'énergie est négligeable.

Le ratio $\hat{P}_{\text{Other\_idle}}$, $\hat{r}_{p\_other\_idle}$, est donné par l'équation (7.12). Il s'agit de la proportion de la consommation de puissance de tous les éléments sauf celle du GPP par rapport

à la puissance dynamique globale, $P_{\text{glob\_dyn}}$.

$$\hat{r}_{p\_other\_idle} = \frac{\hat{P}_{\text{Other\_idle}}}{P_{\text{glob\_dyn}}} * 100 \tag{7.12}$$

Ensuite, la quantité effective d'énergie consommée pour le processus de décodage vidéo, $\hat{E}_{\text{glob\_active}}$, est donnée par l'équation (7.14). Pour l'évaluer, la consommation de puissance du décodage vidéo, $\hat{P}_{\text{glob\_active}}$, est calculée.

$$\hat{P}_{\text{glob\_active}} = P_{\text{glob\_dyn}} - P_{\text{glob\_idle}} \tag{7.13}$$

Ensuite, l'équation (2.1) est appliquée pour calculer la consommation d'énergie du décodage vidéo.

$$\hat{E}_{\text{glob\_active}} = \sum(\hat{P}_{\text{glob\_active}} * \Delta t) \tag{7.14}$$

où $\hat{P}_{\text{glob\_active}}$ est la consommation de puissance du décodage vidéo à chaque période d'échantillonnage de l'appareil de mesure, et Δt est la durée de cette période. Notez que $\sum \Delta t$ est le temps de décodage d'une séquence vidéo.

Enfin, le ratio entre l'énergie de décodage vidéo logiciel et matériel, $\hat{r}_{\text{sw/hw}}$, est calculé à l'aide de l'équation suivante :

$$\hat{r}_{\text{sw/hw}} = \frac{\hat{E}_{\text{glob\_active\_sw}}}{\hat{E}_{\text{glob\_active\_hw}}} * 100 \tag{7.15}$$

où $\hat{E}_{\text{glob\_active\_sw}}$ et $\hat{E}_{\text{glob\_active\_hw}}$ représentent la consommation d'énergie des décodages vidéo logiciel et matériel, respectivement.

7.2.2 Méthodologie de la caractérisation du décodage vidéo

Dans cette section, nous expliquons la méthodologie proposée pour caractériser les performances et la consommation d'énergie des décodages vidéo matériel et logiciel.

La caractérisation du décodage vidéo permet de trouver les paramètres qui ont un

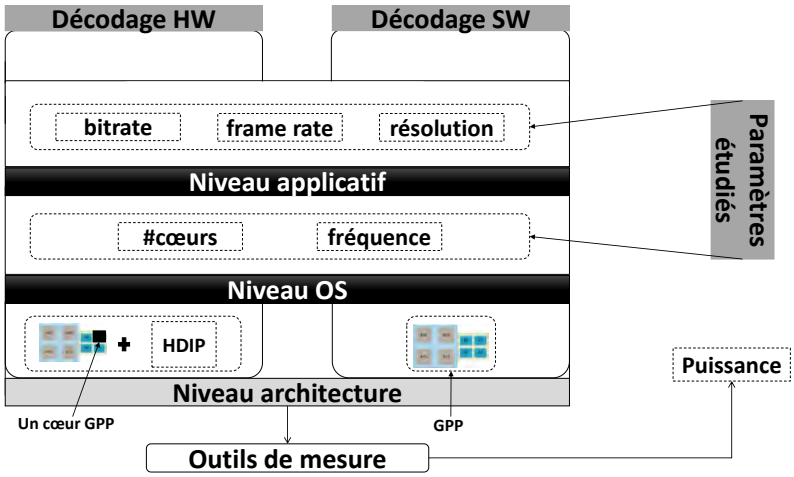


Figure 7.5 – Schéma global de la méthodologie proposée

impact sur les performances et la consommation énergétique du décodage vidéo. Notre approche est basée sur des mesures empiriques, en utilisant des outils de mesure réels, tout en faisant varier différents paramètres afin d'étudier leur pertinence. La méthodologie que nous proposons est conçue indépendamment de tout environnement expérimental sous-jacent afin de la rendre généralisable à plusieurs plateformes.

La figure 7.5 présente un aperçu global de la méthodologie proposée. Les entrées sont les paramètres pour lesquels l'impact sur la consommation d'énergie est estimé et la sortie est la puissance consommée.

La méthodologie proposée intervient à deux niveaux : (i) OS, et (ii) applicatif. L'architecture est la même dans toutes les expériences menées. Un HDIP en plus d'un GPP (appelé GPP–HDIP, dans ce manuscrit), sont utilisés pour le décodage vidéo HW, et un GPP hétérogène est utilisé pour le décodage vidéo SW.

Niveau OS

À ce niveau, nous procédons en trois étapes. Dans un premier temps, nous étudions la consommation de puissance des plateformes, sur lesquelles les décodages vidéo matériel et logiciel sont effectués, à l'état de repos. L'état de repos correspond à l'état où aucun programme n'est exécuté au niveau "utilisateur".

Dans un deuxième temps, nous étudions d'abord la puissance globale consommée par la plateforme, $P_{\text{glob\_dyn}}$, lors du décodage vidéo matériel. Ensuite, nous étudions $\hat{P}_{\text{Other}}$ qui comprend la puissance liée à l'IPC entre le HDIP et d'autres éléments de traitement

Table 7.1
Différentes combinaisons de cinq cœurs GPP hétérogènes

Combinaison	Numéro de cœurs "big"	Nombre de cœurs "LITTLE"
a	4	1
b	3	2
c	2	3
d	1	4

impliqués dans le décodage vidéo, par exemple la mémoire.

Dans un troisième temps, nous étudions d'abord l'énergie consommée par le GPP, un cas particulier de $\hat{E}_{\text{glob\_active}}$, lors du décodage vidéo logiciel. Cette étude consiste à faire varier des paramètres liés à un GPP multi-cœurs hétérogène : nombre de cœurs et leur fréquence d'horloge. Ensuite, nous étudions $\hat{P}_{\text{Other}}$ qui comprend la puissance correspondant aux autres éléments présents dans la plateforme.

Pour simplifier, tous les cœurs du GPP sont cadencés à la même fréquence jusqu'à ce que la valeur maximale supportée par chaque cœur soit atteinte. Au-delà de cette fréquence, les fréquences des cœurs hautes performances continuent d'être mises à l'échelle jusqu'à leur valeur maximale supportée, tandis que les autres cœurs sont cadencés à leur fréquence maximale supportée.

Ensuite, pour chaque fréquence d'horloge, le nombre de cœurs GPP varie également de un au nombre de cœurs disponibles en considérant toutes les différentes combinaisons. En effet, nous faisons varier le nombre de cœurs inactifs et actifs comme défini dans l'équation (3.6). Par exemple, dans une architecture ARM big.LITTLE SoC de huit cœurs GPP (4×big + 4×LITTLE), l'utilisation de cinq cœurs actifs donne quatre combinaisons différentes qui peuvent être évaluées, voir Tableau 7.1.

Ensuite, pour chaque combinaison du nombre de cœurs et de la fréquence, seule celle qui consomme le moins d'énergie (énergie/trame) est sélectionnée afin de la comparer à la consommation énergétique du décodage matériel. Enfin, l'énergie moyenne, le nombre de trames décodées par seconde (*fps*), et le taux d'utilisation du GPP sont calculés en fonction du nombre de cœurs GPP et de leur fréquence d'horloge.

7.2.3 Niveau applicatif

À ce niveau, nous évaluons l'impact des paramètres vidéo : *bitrate*, *frame rate*, et résolution sur la consommation énergétique du décodage vidéo implémenté en matériel

et en logiciel. Ces paramètres peuvent affecter soit la performance (temps de décodage), soit la consommation d'énergie. Nous comparons ensuite la consommation d'énergie des décodages vidéo matériel et logiciel, via le ratio $\hat{r}_{\mathrm{sw/hw}}$, en fonction de ces paramètres.

7.3 Résultats de la caractérisation appliquée sur HEVC

Dans cette section, les résultats obtenus de la caractérisation des performances et de la consommation d'énergie du décodage HEVC sont décrits et analysés. Ces résultats correspondent aux réponses aux questions de recherche (QR1 et QR2) de la problématique de la thèse.

Configurations expérimentales

Dans cette section, toutes les configurations d'environnement utilisées pour réaliser nos expériences sont décrites. Tout d'abord, les configurations matérielles et logicielles sont résumées dans tableau 7.2. Ensuite, les caractéristiques des séquences vidéo utilisées pour valider la méthodologie proposée sont résumées dans le tableau 7.3.

Table 7.2
Résumé des configurations expérimentales

	Décodage matériel (HW)	**Décodage logiciel (SW)**
Configurations matérielles	Snapdragon 810 : 4× big (Cortex-A57) + 4× LITTLE (Cortex-A53)	Odroid-xu3 : 4× big (Cortex-A15) + 4× LITTLE (Cortex-A7)
	Qualcomm RB3 : 4× big + 4× LITTLE (8 cœurs Qualcomm Kryo 385)	
Plateformes de mesure de puissance	N6705A Power Analyzer	Open-PEOPLE (uniquement pour la plateforme Odroid-xu3)
OS	Android 6.0 (Marshmallow)	Ubuntu 16.04
Configurations logicielles	Sur Snapdragon 810 : une application Android (API : Mediacodec)	Sur Odroid-xu3 et RB3 : Open-HEVC + des optimisations NEON
	Sur Qualcomm RB3 : ffmpeg (hevc_v4l2m2m)	
Jeu de vidéos	JCT-VC, Jellyfish, et des séquences vidéo connues sur le web. Leurs caractéristiques sont décrites dans le tableau 7.3	

Dans ce travail, nous avons étudié uniquement le schéma de parallélisme commun à toutes les séquences vidéo, à savoir le schéma trame-par-trame. Pour les autres schémas

Table 7.3
Caractéristiques des séquences vidéo utilisées

Parametètre	Valeur
Resolution	720p, 1080p, 1600p, and 2160p
Frame rate	10, 15, 20, 25, 30, and 50 fps
Mode	Random Access
Profile	Main

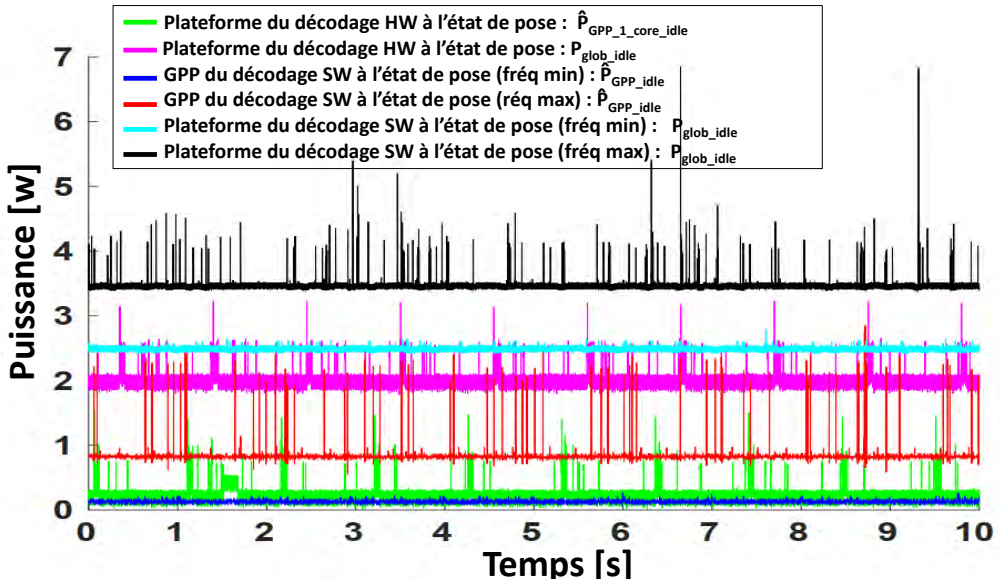


Figure 7.6 – Consommation de puissance des plateformes des décodages vidéo HW (Snapdragon 810) et SW (Odroid-xu3) à l'état de repos

de parallélisme, Wavefront Parallel Processing (WPP) et Tiling, le processus de décodage dépend des paramètres de codage.

Pour des raisons de précision, chaque expérience a été effectuée trois fois, puis la moyenne a été calculée. De plus, la mémoire cache a été vidée au début de chaque expérience afin de la nettoyer des données temporaires utilisées dans les expériences précédentes.

Dans les sections suivantes, les résultats obtenus en utilisant les configurations décrites ci-dessus sont présentés.

Niveau OS

État de repos

Les résultats de l'étude à l'état de repos sont présentés dans la figure 7.6.

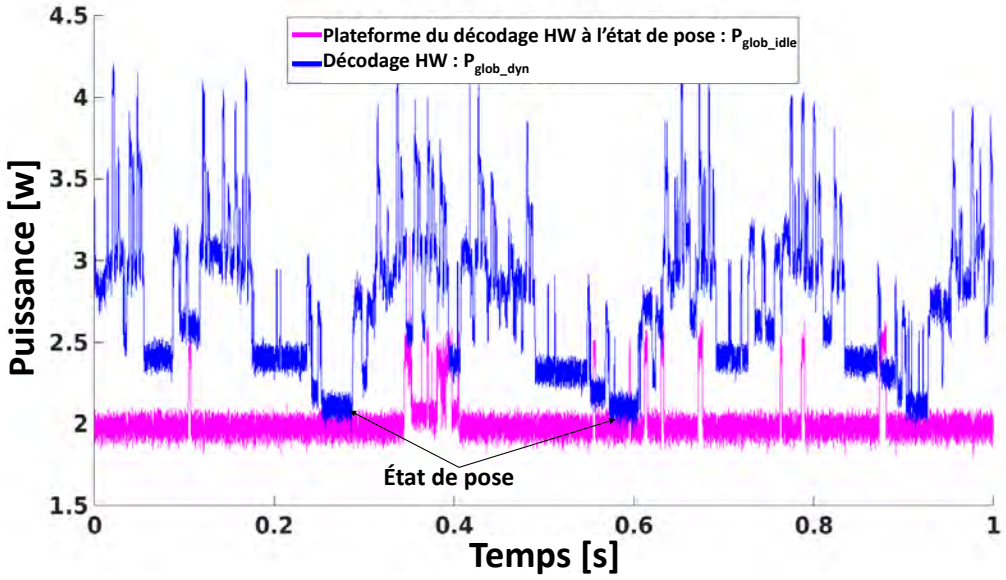


Figure 7.7 – Zoom sur 1 seconde du décodage vidéo HW

Dans le cas de la plateforme de décodage vidéo HW (Snapdragon 810), $\hat{P}_{\text{GPP\_1\_core\_idle}}$ représente en moyenne 0.22 Watts, selon l'équation (7.11), alors que $P_{\text{glob\_idle}}$ représente en moyenne 2.01 Watts. Par conséquent, $\hat{P}_{\text{Other\_idle}}$ constitue en moyenne 88% de la puissance globale de la plateforme.

Sur la plateforme de décodage vidéo SW (Odroid-xu3), l'intervalle des valeurs de puissance du GPP à l'état de repos est en moyenne compris entre 0.13 Watts et 0.82 Watts. Les limites de l'intervalle correspondent à la consommation de puissance aux fréquences d'horloge minimale et maximale du GPP, respectivement. Ainsi, nous pouvons économiser environ 84% de puissance du GPP en réduisant sa fréquence du maximum au minimum. Enfin, $\hat{P}_{\text{Other\_idle}}$ constitue environ 90% et 85% de la consommation de puissance globale de la plateforme pour les fréquences d'horloge du GPP minimale et maximale, respectivement.

Décodage vidéo HW

La figure 7.7 représente la consommation de puissance globale du décodage vidéo HW, $P_{\text{glob\_dyn}}$, comparée à la consommation de puissance globale à l'état de repos (doze pour l'Android), $P_{\text{glob\_idle}}$.

Le décodage vidéo HW ($P_{\text{glob\_dyn}}$) consomme, dans cet exemple, en moyenne 2.75 Watts alors que la consommation à l'état de repos ($P_{\text{glob\_idle}}$) représente en moyenne

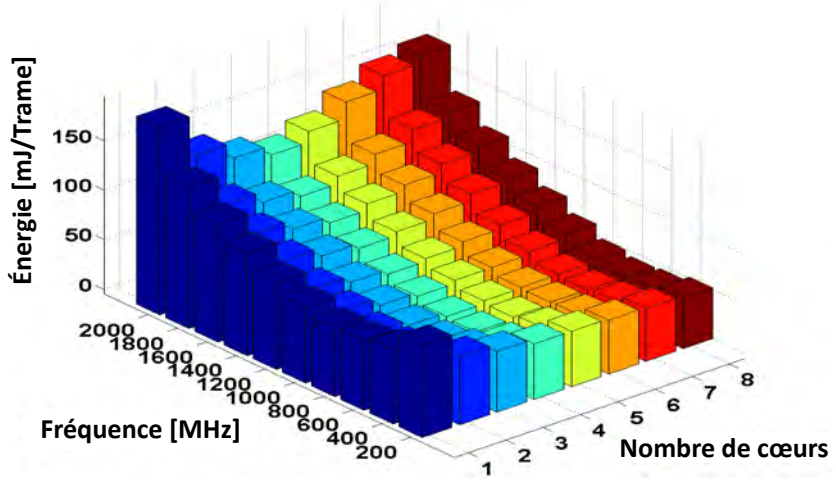


Figure 7.8 – Consommation d'énergie en fonction du nombre de cœurs GPP et de leur fréquence

2.01 Watts. Ainsi, $\hat{P}_{\text{Other}}$ constitue une énorme partie de la consommation de puissance du décodage vidéo HW, en moyenne plus de 70%. Cette partie comprend l'IPC entre le HDIP et le GPP, une opération d'entrée/sortie, ainsi que les autres composants impliqués dans le décodage vidéo, comme la mémoire.

Comme le HDIP (sur la plateforme Snapdragon 810) est conçu pour un traitement de données à haut débit, afin d'économiser de l'énergie, il passe une proportion de temps à l'état de repos chaque fois qu'il termine le décodage d'une trame avant que la période de décodage vidéo suivante ne commence. En moyenne, cette proportion représente environ 5.8% de la période de décodage vidéo.

Décodage vidéo SW

Pour la conception du décodage vidéo SW économe en énergie en exploitant l'hétérogénéité des cœurs GPP, sous la contrainte du temps réel, le parallélisme doit être utilisé avec précaution. Dans l'exemple présenté sur la figure 7.8, la configuration qui donne le meilleur compromis entre performances et énergie (mJ/trame) est l'utilisation de cinq cœurs GPP (un cœur big and quatre cœurs LITTLE) à 1.2 GHz. De plus, le décodage avec cette configuration ((d) dans le tableau 3.1) permettrait d'économiser, au niveau du système, environ 29% de l'énergie (mJ/trame) par rapport à la configuration avec quatre cœurs big et un cœur LITTLE ((a) dans le tableau 3.1).

En outre, pour effectuer le décodage vidéo SW, le GPP consomme effectivement en

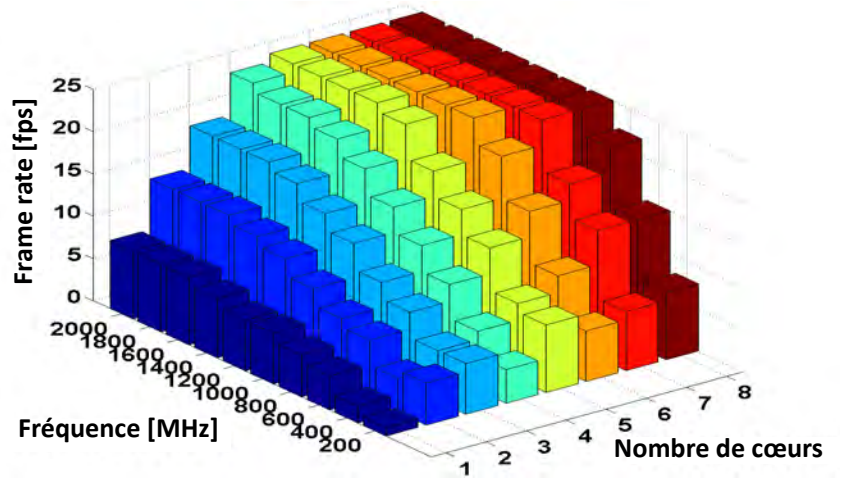


Figure 7.9 – FPS en fonction du nombre de cœurs GPP et de leur fréquence

Table 7.4
Consommation énergétique des décodages HEVC matériel (sur Snapdragon 810) et logiciel (sur Odroid-xu3 platform)

Décodage	**SW (#big : #LITTLE)**					**HW**
Config	1 cœur GPP (0 : 1)	2 cœurs GPP (2 : 0)	3 cœurs GPP (3 : 0)	4 cœurs GPP (4 : 0)	5 cœurs GPP (1 : 4)	GPP–HDIP
$\hat{r}_{\text{sw/hw}}$	5.38	4.93	4.93	4.88	1.66	1

moyenne moins de 50% de la consommation énergétique globale de la plateforme. Cette surcharge inclut la communication interprocesseur entre le GPP et les autres composants impliqués dans le décodage vidéo, comme la mémoire.

Les résultats présentés dans le tableau 7.4 montrent que le fait de varier le nombre de cœurs GPP permet de réduire considérablement la consommation d'énergie. L'utilisation d'un seul cœur consomme non seulement la plus grande quantité d'énergie (un ratio de plus de 5×) mais aussi la contrainte de temps réel n'est pas satisfaite. Ensuite, lorsque deux à quatre cœurs big sont sollicités, la vidéo est décodée en temps réel. Cependant, ces configurations consomment beaucoup d'énergie (ratios de plus de 4×). Enfin, la configuration avec cinq cœurs offre le meilleur compromis entre performances et efficacité énergétique. Elle diminue en effet la consommation d'énergie du décodage vidéo SW et la rapproche le plus de celle du décodage vidéo HW (un ratio inférieur à 2×).

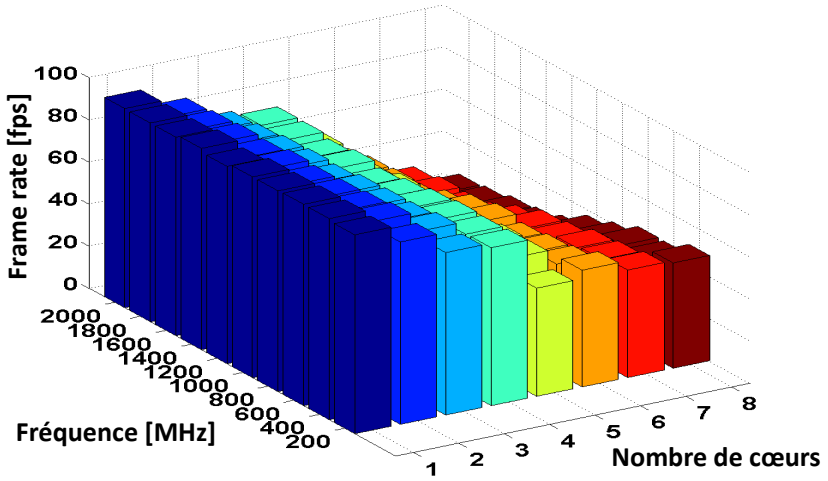
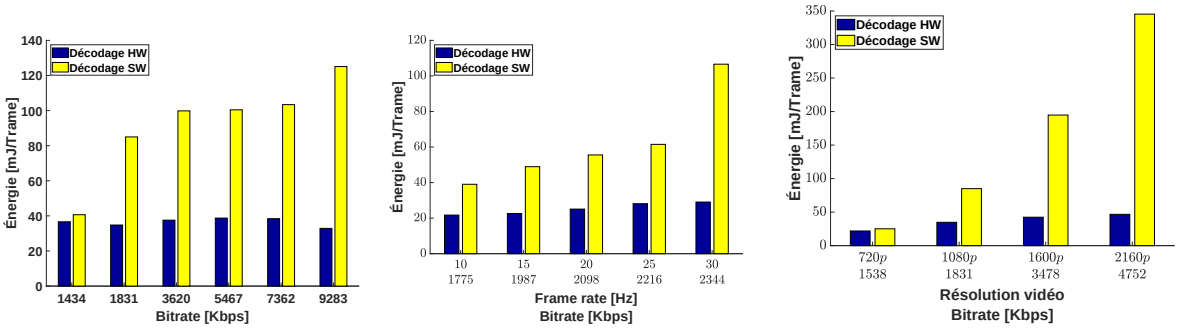


Figure 7.10 – Taux d'utilisation du GPP en fonction du nombre de cœurs GPP et de leur fréquence



(a) Variation du *bitrate* (*frame rate* : 25 Hz, résolution : 1080p)

(b) Variation du *frame rate* (résolution : 1080p)

(c) Variation de la résolution (*frame rate* : 25 Hz)

Figure 7.11 – Consommation énergétique des décodages HEVC HW vs SW (Séquence vidéo : kimono, QPs : 21 to 37, PSNRs : 22.97 à 43.8, plateforme : Snapdragon 810 et Odroid-xu3)

Niveau applicatif

Les figures 7.11a, 7.11b, et 7.11c illustrent l'impact du *bitrate*, du *frame rate*, et de la résolution sur la consommation énergétique des décodages vidéo HW et SW (mJ/trame), $\hat{E}_{\text{glob\_active}}$ de l'équation (7.14).

Les paramètres vidéo : *bitrate*, *frame rate*, et résolution impactent différemment la consommation énergétique des décodages vidéo HW et SW (sur les plateformes Snapdragon 810 et Odroid-xu3). Par exemple, le passage de la résolution de 720p à 2160p

Table 7.5
Ratio de consommation énergétique des décodages HEVC SW et HW en fonction de la résolution, voir l'équation (7.15), sur Snapdragon 810 et Odroid-xu3

Resolution	720p	1080p	1600p	2160p
$\hat{r}_{\mathbf{sw/hw}}$	1.15	2.18	4.59	7.4

en passant par 1080p et 1600p augmente la consommation d'énergie de 1.43×, 1.94×, et 2.14× dans le cas du HW et 1.64×, 7.76× et 13.76× dans celui du SW. Par conséquent, plus la résolution de la vidéo est grande, plus l'avantage de la consommation énergétique du décodage vidéo HW est remarquable.

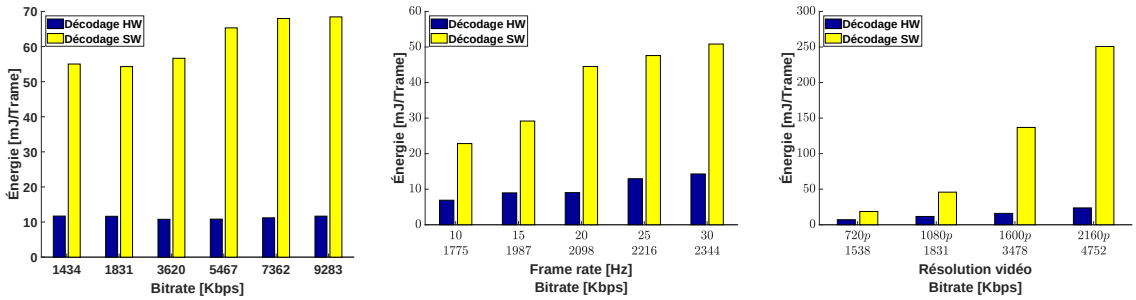
Les paramètres vidéo : *bitrate*, *frame rate*, et résolution représentent une forte contrainte pour le processus de décodage vidéo. En effet, ils doivent être respectés dans un certain délai, par exemple, nombre de bits par seconde, pour garantir une visualisation fluide de la vidéo. Par conséquent, toutes les ressources nécessaires doivent être allouées afin de satisfaire cette contrainte, même au prix d'une surconsommation d'énergie. Cela signifie que le système d'exploitation doit donner au décodeur vidéo une priorité élevée pour utiliser le GPP. Cela s'applique aussi à toutes les ressources nécessaires pour effectuer le décodage vidéo : mémoire (DRAM, cache et disque dur), bus, etc.

Les paramètres vidéo : *frame rate* et résolution sont fortement corrélés avec le *bitrate*. Par exemple, lorsque la résolution d'une séquence vidéo donnée est modifiée, pour conserver son *frame rate*, le *bitrate* doit être adapté en conséquence. Plus la résolution vidéo est élevée, plus le *bitrate* est élevé. De même, plus le *frame rate* est élevé, plus le *bitrate* est élevé. Par conséquent, les figures 7.11a, 7.11b, et 7.11c présentent des comportements similaires.

Le tableau 7.5 présente le ratio entre la consommation d'énergie du décodage HEVC SW et celle du HW, calculé à l'aide de l'équation (7.15), en fonction de la résolution vidéo. Pour les basses résolutions vidéo (720p et 1080p), d'un point de vue système, l'écart n'est pas aussi élevé que celui de l'état de l'art [22] (un ratio de moins de 3×). Cependant, pour les résolutions vidéo élevées, le GPP–HDIP offre le meilleur compromis entre les performances et l'efficacité énergétique (un ratio supérieur à 7×).

Généralisation des résultats

Afin de valider les résultats de la caractérisation obtenues pour deux plateformes de décodage HW et SW différentes (Snapdragon 810 et Odroid-xu3), les mêmes expérimen-



(a) Variation du *bitrate* (*frame rate* : 25 Hz, résolution : 1080p)

(b) Variation du *frame rate* (résolution : 1080p)

(c) Variation de la résolution (*frame rate* : 25 Hz)

Figure 7.12 – Consommation énergétique des décodages HEVC HW vs SW (Séquence vidéo : kimono, QPs : 21 to 37, PSNRs : 22.97 à 43.8, plateforme : RB3)

Table 7.6
Ratio de consommation énergétique des décodages HEVC SW et HW en fonction de la résolution, voir l'équation (7.15), sur RB3

Resolution	720p	1080p	1600p	2160p
$\hat{\mathbf{r}}_{\mathrm{sw/hw}}$	2.65	3.93	8.57	10.6

tations ont été réalisées sur une même plateforme (RB3). Les figures 7.12a, 7.12b, et 7.12c illustrent ces résultats.

Comparé aux résultats obtenus dans la figure 7.11, on peut noter qu'ils montrent une tendance similaire de la consommation d'énergie (mJ/trame). Cela peut également être confirmé par le tableau 7.6 qui présente les résultats des mêmes mesures que le tableau 7.5 mais sur une plateforme différente (RB3). Nous pensons que la différence de ratios pourrait s'expliquer par le fait que RB3 consomme une quantité importante de puissance statique car elle est équipée d'un SoC dotée d'une technologie de 10 nm [144].

7.4 Optimisation de la consommation énergétique du décodage vidéo

Dans cette section, nous traitons les questions de recherche (QR3 et QR4) de la problématique de la thèse. Premièrement, nous expliquons la solution proposée. Ensuite, les configurations expérimentales pour valider la solution proposée sont présentées. Finale-

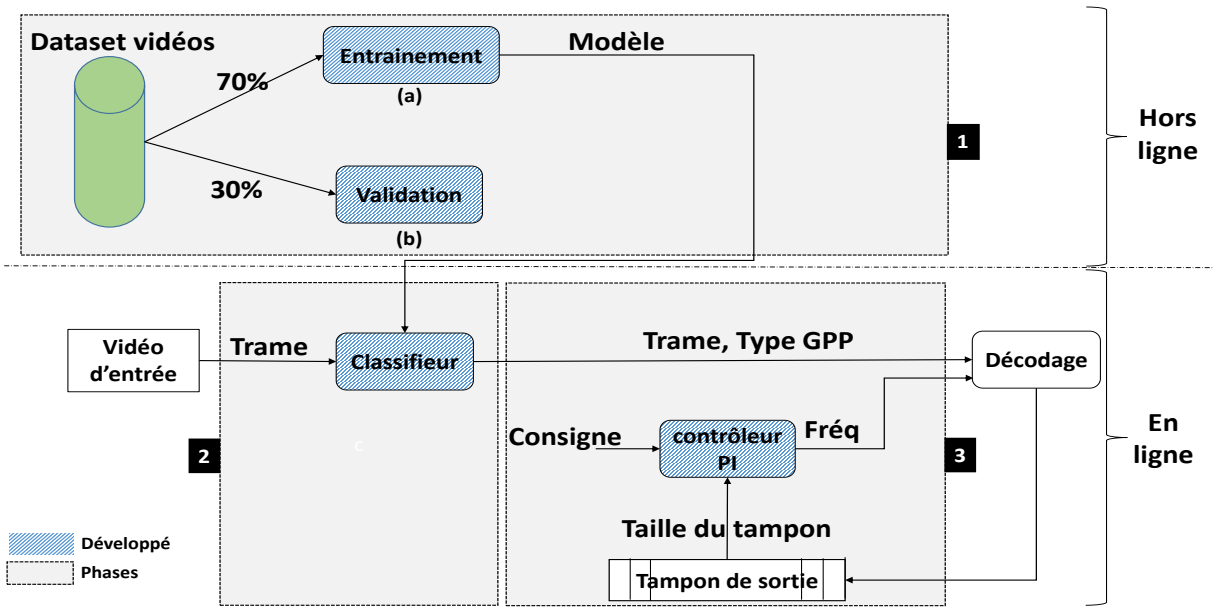


Figure 7.13 – Schéma global de la solution proposée

ment, les résultats obtenus sont présentés et analysés.

7.4.1 Optimisation de la consommation d'énergie du décodage vidéo SW

La solution proposée comprend trois phases : (i) modélisation de la complexité de trame, (ii) assignation, et (iii) mise à jour de la fréquence du GPP [41]. La figure 7.13 présente un schéma global de la solution proposée.

7.4.2 Phase 1 : Modélisation de la complexité de trame

L'objectif de la phase 1 est de construire un modèle capable d'équilibrer des trames vidéo entre les cœurs GPP haute performance et ceux à faible consommation d'énergie. Pour ce faitm le modèle classe chaque trame vidéo dans l'un des deux groupes : (i) les trames les plus complexes, et (ii) les trames les moins complexes. La complexité est exprimée par le nombre de cycles processeur nécessaires pour décoder une trame. Grâce à cette classification, le premier groupe sera soumis aux cœurs GPP haute performance, et le second aux cœurs GPP à faible consommation d'énergie.

Cette phase est réalisée pendant la conception, c'est-à-dire avant de lancer le processus du décodage. Elle n'est effectuée qu'une seule fois pendant toute la durée de vie de la plate-

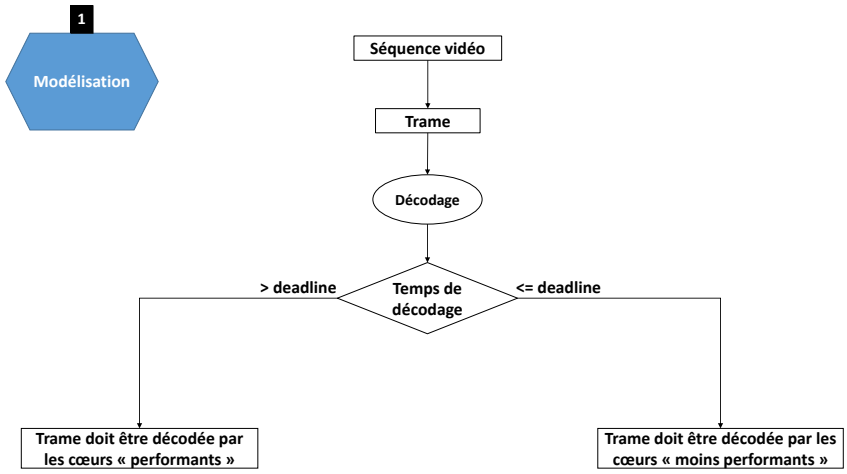


Figure 7.14 – La solution proposée : phase 1 (modélisation de la complexité de trame)

forme cible. Le résultat de cette phase est un modèle qui constitue l'entrée de la phase 2, comme l'illustre la figure.

Pour construire le modèle, comme illustré dans la figure 7.13, il y a deux étapes : (a) l'entrainement du modèle, et (b) la validation du modèle. Dans la première étape, les données relatives aux images représentant plusieurs séquences vidéo sont collectées. Ensuite, une partie d'entre elles (70%) est injectée dans le modèle pour l'apprentissage, c'est-à-dire que le modèle apprend à corréler les données d'entrée à la complexité des trames. Ensuite, dans la deuxième étape, le modèle est appliqué sur les données restantes (30%) pour mesurer sa précision, et ainsi l'accepter ou le rejeter.

La sortie du modèle est le type des cœurs GPP : haute performance ou à faible consommation d'énergie. Les variables indépendantes d'entrée sont : le type de la trame et sa taille. Les paramètres de configuration sont : (i) le *bitrate* vidéo, (ii) le *frame rate* vidéo, et (iii) le ratio de performances entre les cœurs GPP haute performance et ceux à faible consommation d'énergie.

La figure 7.14 illustre le processus de collecte des données pour l'entrainement du modèle. Pour chaque trame de l'ensemble des séquence vidéo du dataset, elle est décodée par les cœurs GPP moins performants. Ces derniers tournent à leur fréquence maximale. Si la frame est décodée dans les délais, elle doit être décodée par les cœurs GPP moins performants ; sinon elle doit être décodée par les cœurs GPP performants. Ces données sont enregistrée dans un fichier qui servira à l'entraînement du modèle de la complexité de trames.

Enfin, le modèle résultant de cette phase est exprimé par la formule suivante, en

utilisant une fonction logistique [145] :

$$y = \frac{1}{1 + e^{-\mathrm{p(x1,\ x2,\ x3)}}} \tag{7.16}$$

où y est la sortie du modèle qui sera utilisé dans la phase 2. Elle prend des valeurs comprises entre 0 et 1. $p(x1, x2, x3)$ est la fonction linéaire des paramètres d'entrée et de configuration décrits ci-dessus. Il s'agit d'un nombre réel.

La fonction linéaire est formulée comme suit :

$$p(v_{\mathrm{bitrate}}, f_{\mathrm{type}}, f_{\mathrm{size}}) = \frac{w_0 + w_1 * v_{\mathrm{bitrate}} + w_2 * f_{\mathrm{type}} + w_3 * f_{\mathrm{size}}}{ratio\_performance} \tag{7.17}$$

où w_0 est l'ordonnée à l'origine, v_{bitrate} est le débit binaire vidéo, f_{type} et f_{size} sont le type et la taille de la trame à décoder, respectivement, et $ratio\_performance$ est le rapport de performance entre les cœurs GPP haute performance et ceux à faible consommation d'énergie de la plateforme cible. Enfin, w_1, w_2 et w_3 sont les coefficients du modèle.

7.4.3 Phase 2 : Assignation par classification

L'objectif de la phase 2 est d'utiliser le modèle de la complexité de trames pour décider quels cœurs GPP (haute performance ou à faible consommation d'énergie) décoderont la trame à traiter. Pour cela, le classifieur prend le type et la taille de la trame à décoder et les injecte au modèle construit dans la phase 1. Ensuite, la trame est envoyée au cluster GPP sélectionné. Finalement, la trame est décodée en parallèle entre les cœurs du cluster GPP choisis via un schéma de parallélisme *tiling* ou WPP (selon les configurations de codage).

La figure 7.15 illustre l'algorithme de classification.

7.4.4 Phase 3 : Mise à jour de la fréquence du GPP en utilisant le contrôleur PI

La phase 3 est inspirée de l'état de l'art [41]. L'objectif de cette phase est de sélectionner la fréquence d'horloge à laquelle les cœurs GPP choisis décoderont la trame courante. Pour cela, un contrôleur Proportionnel Intégral (PI) est utilisé [41]. Ce contrôleur surveille la

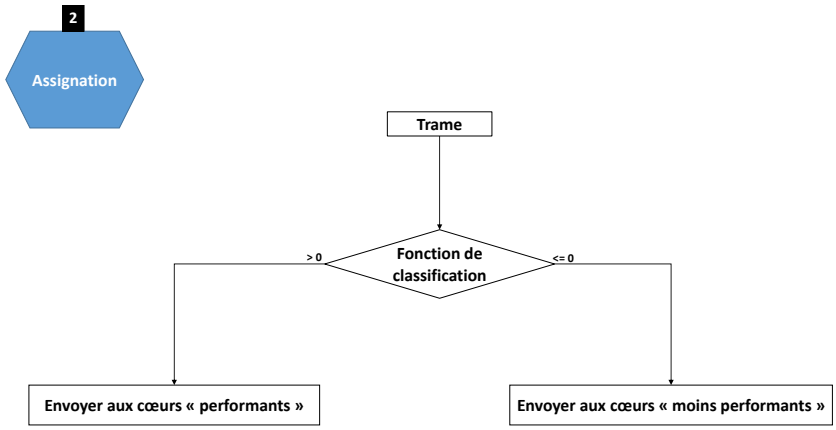


Figure 7.15 – La solution proposée : phase 2 (assignation en utilisant la classification)

taille du tampon de sortie afin de la maintenir à une valeur souhaitée (la consigne), qui est un paramètre du contrôleur [3]. Ainsi, lorsque la taille du tampon de sortie dépasse la consigne, le contrôleur ralentit les cœurs GPP en baissant leur fréquence, et vice versa.

La figure 7.16 illustre le diagramme du contrôleur PI utilisé dans la phase 3. Il y a deux entrées du contrôleur : (i) la taille actuelle du tampon de sortie, et (ii) la consigne.

Ensuite, pour accélérer ou ralentir un cluster GPP, un facteur, r, est calculé et ensuite multiplié par la fréquence maximale supportée par le cluster GPP choisi dans la phase 2.

$$gpp\_cluster\_freq = GPP\_cluster_{\text{max\_freq}} * r \tag{7.18}$$

où $gpp\_cluster\_freq$ est la fréquence d'horloge de sortie du cluster GPP choisi en phase 2, $GPP\_cluster_{\text{max\_freq}}$ est sa valeur de fréquence maximale prise en charge, et r est le facteur d'échelle de cette fréquence. La valeur de r est comprise entre 0 et 1. Par exemple, une valeur r de 1 signifie que le cluster GPP décode la trame actuelle à sa vitesse maximale.

Le facteur d'échelle r est, à son tour, décomposé en deux composantes, comme l'illustre la formule suivante :

$$r(n) = r_{\text{e}}(n) + r_{\text{c}}(n) \tag{7.19}$$

3. Pour définir la consigne, des études de l'état de l'art donnent des directives de choix, such as [41][115].

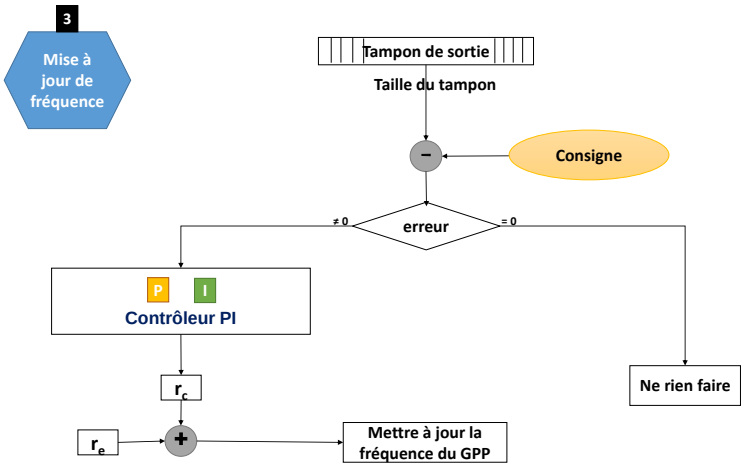


Figure 7.16 – La solution proposée : phase 3 (mise à jour de la fréquence du GPP en utilisant un contrôleur PI)

où r_e est l'estimation du facteur d'échelle basée sur les trames précédemment décodées, r_c est la sortie du contrôleur PI qui est considérée comme un ajustement de r_e, et n est le numéro de la prochaine trame à décoder. Autrement dit, une valeur négative de r_c indique que les cœurs GPP doivent être ralentis, et vice versa.

7.4.5 Configurations expérimentales

Les configurations expérimentales pour valider la solution proposée sont globalement les mêmes que celles décrites dans la Section 7.3. De plus, pour construire le modèle de la phase 1 de la solution proposée, le framework sickit-learn [148] a été utilisé via le langage de programmation Python. Concernant les séquences vidéo de test, la résolution 1080p a été visée car environ 80% des plateformes mobiles sont équipés d'écrans de résolution 1080p ou moins [4]. Dans le dataset de test, il y a 33 séquences vidéo.

7.4.6 Résultats & analyse

Précision du modèle

Le modèle résultant est une fonction logistique. La raison de ce choix est sa simplicité d'implémentation et son efficacité à prendre une décision binaire (trames les plus complexes ou les moins complexes).

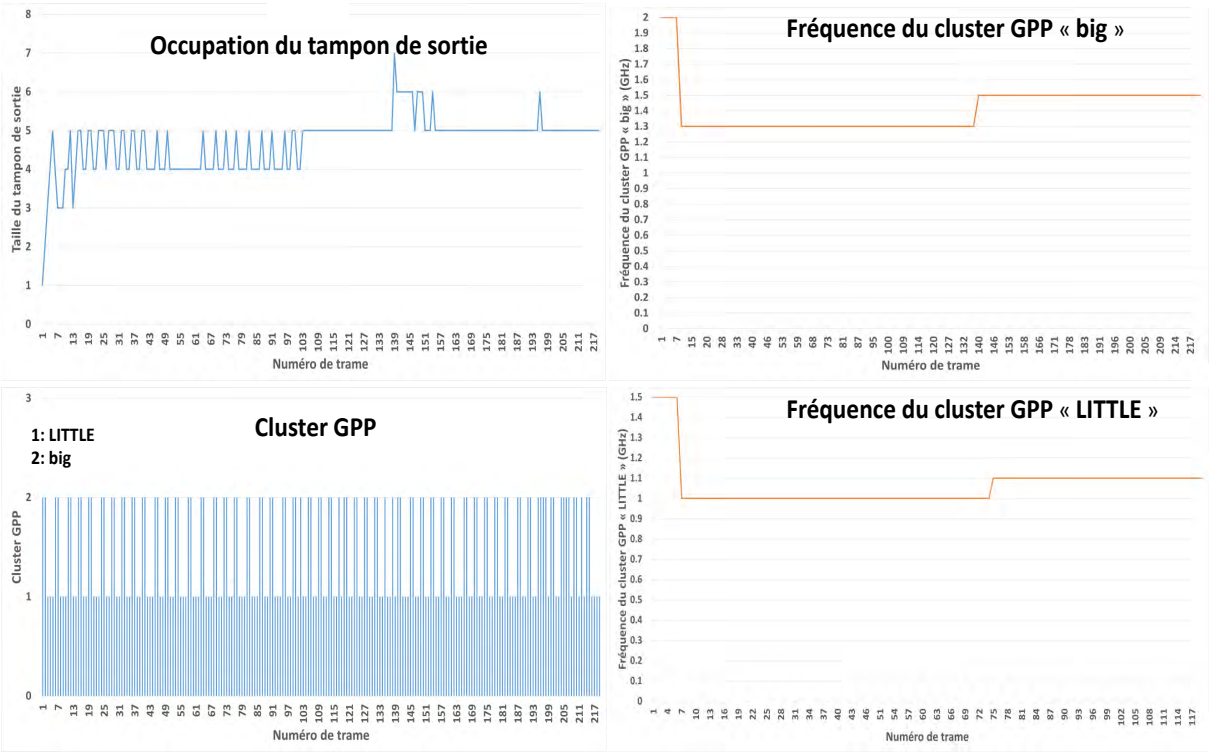


Figure 7.17 – Analyse du contrôleur PI

Le modèle a été entraîné avec 70% de trames vidéo. Les 30% de trames restantes ont été utilisées pour la validation. Le modèle résultant est une fonction logistique car il est simple à mettre en œuvre et précis. En effet, la précision de ce modèle est de 93% et 98% pour les plateformes Odroid-xu3 et RB3, respectivement.

Stabilité de la taille du tampon de sortie

La figure 7.17 présente les résultats de la phase 2 et de la phase 3 de la solution proposée lors du décodage de la séquence de test vidéo *Bluesky*. La phase 2 est représentée par la variation des clusters GPP (en bas à gauche), tandis que la phase 3 est représentée par la variation de la taille du tampon de sortie (en haut à gauche), et les fréquences d'horloge des clusters GPP (à droite). Les coefficients du contrôleur PI sont configurés comme suit : $K_\mathrm{p} = 0,01$ et $K_\mathrm{i} = 0,001$. De plus, la consigne a été fixée à 5.

Au début du décodage, les clusters GPP sont cadencés à leur fréquence maximale. Cela permet de remplir le tampon de sortie aussi rapidement que possible. Ensuite, le processus d'affichage commence à recevoir des trames, et donc le contrôleur PI commence à surveiller la taille du tampon de sortie.

Table 7.7
Gain d'énergie de la solution proposée (%) par rapport à l'état de l'art

Travaux de l'état de l'art		Gouverneur Linux "Performance"	gouverneur Linux "Ondemand"	Caractérisation (section 7.2)	Contrôleur PI [41]
Moyenne du gain en énergie (%)	Sur les plateformes Snapdragon 810 et Odroid-xu3	40	30	7	20
	Sur la plateforme RB3	35	20	4	23

La stabilité de cette dernière a été atteinte en moyenne en moins d'une seconde du décodage. Dans l'exemple de la figure 7.17, la taille du tampon s'est stabilisée après le décodage de 17 trames, soit moins d'une seconde de décodage (le *frame rate* est de 25 Hz). Le remplissage du tampon de sortie va entre 4 et 5 ou entre 5 et 6. Ceci est dû à la variation de la complexité des trames. La stabilité de la taille du tampon de sortie implique la stabilité de la fréquence GPP qui est une condition de la réduction de sa consommation d'énergie, comme le suggère l'inégalité de Jensen dans [1].

7.4.7 Coût de la solution proposée

Le coût de la solution proposée est évalué comme le ratio entre le temps nécessaire à l'exécution des phases 2 et 3 et le temps nécessaire au décodage d'une trame. Le ratio est calculé pour chaque trame. Ensuite, la moyenne des ratios est calculée et présentée ici.

Les résultats montrent que l'exécution de la solution proposée représente en moyenne moins de 1% du temps de décodage. Ce coût est négligeable par rapport au gain d'énergie que la solution proposée permet d'obtenir.

7.4.8 Comparaison avec l'état de l'art

La tableau 7.7 illustre les résultats de la comparaison entre la solution proposée et l'état de l'art. Les expérimentations ont été menées sur les plateformes Snapdragon 810 (décodage vidéo HW), Odroid-xu3 (solution de décodage vidéo SW), et RB3 (décodages HW et SW).

Sur les plateformes Snapdragon 810 et Odroid-xu3, la solution proposée peut économiser

en moyenne 40% et 30% d'énergie (mJ/Trame) par rapport aux gouverneurs Linux Performance et Ondemand, respectivement. Ensuite, la solution proposée non seulement détermine dynamiquement le bon cluster GPP et sa fréquence d'horloge, contrairement à la solution de caractérisation, mais aussi peut économiser en moyenne 7% d'énergie (mJ/trame) par rapport à cette dernière. En outre, l'efficité de la classification a été démontrée. En effet, la solution proposée permet d'économiser en moyenne 20% par rapport à une solution sans classification. Finalement, le ratio d'énergie entre la solution proposée et le décodage vidéo HW est en moyenne seulement d'un facteur égal à 3×.

Pour valider ces résultats, des expériences ont été menées sur la plateforme RB3 qui supporte à la fois les décodages HEVC HW et SW. Sur cette plateforme, la solution proposée permet d'économiser en moyenne 35%, 20%, 4%, et 23% d'énergie (mJ/trame) par rapport au gouverneur Linux Performance, au gouverneur Linux Ondemand, à la solution de caractérisation, et à la solution sans classification, respectivement. En ce qui concerne le décodage vidéo HW, la solution proposée consomme en moyenne seulement 4× plus d'énergie (mJ/trame).

La classification joue un rôle important pour exploiter l'hétérogénéité de l'architecture ARM big.LITTLE et limiter le taux d'échecs. En effet, la solution proposée induit moins de 5% de taux d'échecs, alors que plus de 10% de taux d'échecs sont observés lorsque le gouverneur Linux Ondemand est mis en place.

7.5 Conclusion & perspectives

7.5.1 Conclusions

La présente thèse aborde la question de la consommation énergétique du décodage vidéo sur des SoC mobiles hétérogènes à faible consommation. Elle intervient dans un contexte où, d'une part, la demande en contenu vidéo mobile est toujours croissante avec ce que cela implique en termes de QoS, et, d'autre part, la durée de vie des batteries reste limitée et n'évolue pas aussi vite que le contenu vidéo. De plus, les solutions de décodage vidéo HW et SW sont en rude concurrence, la première manquant de flexibilité alors que la seconde gagne cette caractéristique mais souffre toujours du budget énergétique.

Plus précisément, nous avons essayé de répondre à quatre questions de recherche énoncées ci-dessous.

1. QR1 : Quelle est la performance des décodages vidéo (HW et SW) par rapport aux

paramètres ayant un impact sur le processus de décodage vidéo ?

2. QR2 : Dans quelle mesure le décodage vidéo HW est-il plus performant que le décodage SW ?
3. QR3 : Comment équilibrer la charge vidéo entre les cœurs GPP hétérogènes, sur la base d'une faible quantité d'informations sur la vidéo à décoder ?
4. QR4 : Enfin, une fois les cœurs GPP sélectionnés, comment ajuster leur fréquence afin de réduire la consommation d'énergie ?

Les réponses aux questions de recherche ci-dessus ont conduit à deux contributions principales. Premièrement, la thèse a consisté à comprendre les performances et la consommation énergétique des décodages vidéo HW et SW en les caractérisant. Deuxièmement, elle a consisté à proposer des mécanismes et des stratégies pour optimiser la consommation énergétique du décodage vidéo SW.

Dans la première contribution, les questions de recherche (QR1 et QR2) ont été abordées. La méthodologie proposée est basée sur des mesures approfondies de performance et de consommation d'énergie. Les expériences ont été réalisées sur des plates-formes réelles représentant différentes configurations de traitement, y compris : (a) processeur mono-cœur GPP, (b) processeur multi-cœurs GPP, et (c) décodeur vidéo HW (HDIP). Cela a permis de refléter des scénarios réels. En outre, les expériences ont été menées sur trois plateformes différentes qui représentent des téléphones mobiles de différentes générations. L'utilisation de différentes plateformes permet d'avoir une compréhension plus claire de la consommation d'énergie du décodage vidéo. Elle permet également une comparaison objective entre la consommation d'énergie du décodage vidéo HW et SW. Enfin, elle rend les résultats obtenus généralisables à d'autres plateformes non testées dans notre travail. La méthodologie de caractérisation proposée est décrite dans la section 7.2. Ensuite, les résultats de la méthodologie appliquée au codec HEVC sont présentés et analysés dans la section 7.3.

QR1 s'est interrogée sur les paramètres ayant un impact sur les performances et la consommation d'énergie du décodage vidéo dans les deux implémentations : HW et SW. Les paramètres sont déclenchés à deux niveaux : (i) le niveau du système d'exploitation (OS), et (ii) le niveau de l'application. La réponse est double. Tout d'abord, au niveau OS, la méthodologie de caractérisation proposée teste toutes les configurations possibles (nombre de cœurs GPP et leur fréquence d'horloge) et mesure l'énergie consommée pour chacune d'elles. Ensuite, elle sélectionne la configuration qui consomme le moins d'énergie.

Deuxièmement, au niveau de l'application, les paramètres vidéo : bitrate, frame rate, et résolution sont étudiés.

Les résultats montrent que, sur les plateformes testées, le décodage avec un gros cœur GPP et quatre cœurs LITTLE GPP consomme en moyenne 2× moins d'énergie (mJ/Frame) que le décodage avec deux gros cœurs GPP. Ensuite, les paramètres étudiés au niveau de l'application ont été mis en corrélation avec les performances et la consommation d'énergie du décodage HEVC. Cela confirme que le décodage HEVC est similaire à ses prédécesseurs en termes de consommation d'énergie par rapport aux paramètres ci-dessus. QR1 est résolue dans la section 7.2 et les résultats sont présentés et analysés dans la section 7.3.

QR2 s'est interrogée sur le rapport de consommation d'énergie entre le HW et le SW décodage vidéo. La réponse dépend du niveau auquel l'étude est réalisée ainsi que des paramètres pris en compte pour effectuer la comparaison. En effet, la comparaison de la consommation énergétique du décodage vidéo HW et SW au niveau du processeur est différente de la même comparaison mais au niveau du système. L'étude comparative multi-niveaux de Par exemple, l'overhead du système a été évalué et son impact sur les performances globales et la consommation d'énergie a été analysé.

Les résultats montrent que, sur les plates-formes testées, pour les basses résolutions vidéo (par exemple, 720p) et du point de vue du système, le décodage vidéo SW consomme autant d'énergie que le HW. Cependant, pour les hautes résolutions vidéo (par exemple, 2160p), le décodage vidéo HW est plus adapté car il consomme moins d'énergie. De plus, au niveau du système, le GPP, lorsqu'il effectue le décodage vidéo, consomme en moyenne au maximum 10× de plus d'énergie que le HDIP alors que, dans l'état de l'art, le rapport au niveau du processeur est d'environ 1000×. Par conséquent, une comparaison objective devrait prendre en compte des paramètres à plusieurs niveaux afin d'avoir une compréhension plus large des performances et de la consommation d'énergie du décodage vidéo sur les plateformes mobiles. QR2 est résolue dans la section 7.2 et les résultats sont présentés et analysés dans la section 7.3.

Dans la deuxième contribution, les questions de recherche (QR3 et QR4) ont été traitées. La solution d'optimisation proposée ainsi que les résultats obtenus sont décrits et analysés dans la section 7.4.

QR3 s'interroge sur la possibilité d'exploiter l'hétérogénéité des GPP mobiles pour équilibrer les trames vidéo entre ses cœurs en se basant sur une petite connaissance des caractéristiques de ces trames. La solution que nous proposons est basée sur la technique

de classification. Grâce à cette technique, les trames peuvent être classées en deux groupes : (i) les trames les plus complexes, et (ii) les trames les moins complexes, de sorte que le premier groupe de trames doit être décodé par les cœurs à haute performance (par exemple, big GPP cluster) et le second groupe par les cœurs à faible consommation d'énergie (par exemple, LITTLE GPP cluster). Le classificateur ne prend que deux informations sur la trame à décoder : son type et sa taille. QR3 est résolue dans la section 7.4.

Une fois qu'un GPP cluster est sélectionné grâce à la classification, QR4 s'interroge sur un mécanisme permettant d'ajuster sa fréquence d'horloge. La solution proposée a réutilisé un travail de pointe qui est basé sur la technique de contrôle par rétroaction (contrôleur PI). Le contrôleur surveille la taille du tampon de sortie afin qu'elle maintienne une certaine valeur qui est un paramètre (fixé par l'utilisateur). Cette technique a permis de maintenir la taille du tampon de sortie et donc la fréquence GPP stable, ce qui est une recommandation d'ARM pour économiser l'énergie sur les SoCs de l'architecture basée sur big.LITTLE. QR4 est résolu dans la section 7.4.

Les résultats montrent que, sur les plateformes testées, la solution proposée permet d'économiser en moyenne plus de 20% d'énergie par rapport au gouverneur Linux Ondemand. De plus, la solution proposée consomme en moyenne seulement 4× plus d'énergie (mJ/Frame) que le décodage vidéo HW ce qui confirme les résultats trouvés dans notre première contribution. De plus, la classification permet d'exploiter l'hétérogénéité de l'architecture ARM big.LITTLE en limitant le taux d'échec. En effet, la solution proposée induit moins de 5% de taux d'échec, alors que le gouverneur Ondemand Linux induit plus de 10% de taux d'échec. En termes d'occupation de la mémoire tampon, la solution proposée nécessite moins d'une seconde de décodage pour entrer dans un état stable. Enfin, la solution proposée est très légère puisqu'elle représente sur l'overage moins de 1% du temps de décodage de la trame. Par conséquent, la réponse à cette question, c'est-à-dire la solution proposée, peut facilement être intégrée dans les plateformes mobiles pour optimiser la durée de vie de la batterie.

En résumé, la première contribution permet de réduire la consommation énergétique du décodage vidéo tout en satisfaisant la contrainte de temps réel de manière statique, c'est-à-dire en testant toutes les configurations possibles. Ce scénario est utile dans un nombre limité de cas d'utilisation, comme la vidéo à la demande (VoD). La deuxième contribution atteint le même objectif (performance et consommation d'énergie) mais de manière dynamique. Elle exploite le caractère hétérogène des plateformes mobiles, par exemple, ARM big.LITTLE, en utilisant la technique de classification.

En conclusion, bien que la question de la consommation d'énergie du décodage vidéo soit toujours activement considérée par la communauté scientifique (dans le milieu universitaire et l'industrie), cette thèse a contribué à cet effort. Elle a permis d'améliorer la compréhension de la consommation d'énergie des décodages vidéo HW et SW sur des SoC mobiles hétérogènes à faible consommation d'énergie ainsi que de combler le fossé entre ces deux approches. Enfin, cette thèse a soulevé des questions ouvertes qui seront discutées dans la section suivante.

7.5.2 Perspectives

Les travaux présentés dans cette thèse ont permis d'améliorer la compréhension des performances et de la consommation d'énergie des décodages HEVC HW et SW. Afin de poursuivre la recherche sur ce sujet, plusieurs directions possibles qui nécessitent des investigations plus approfondies sont proposées dans cette section.

Application sur d'autres codecs vidéo

Comme expliqué dans la section 7.2, la méthodologie proposée pour caractériser la performance et la consommation d'énergie du décodage vidép est indépendante de tout codec vidéo spécifique. Dans cette thèse, la méthodologie de caractérisation a été appliquée au codec HEVC. Par conséquent, elle peut facilement être étendue à des codecs vidéo plus récents, par exemple VVC, et ce sans aucune modification. En effet, les paramètres vidéo étudiés dans cette thèse (*bitrate*, *frame rate*, et résolution) sont maintenus dans le nouveau codec.

En ce qui concerne l'optimisation proposée dans la section 7.4, elle peut également être appliquée à VVC. En fait, malgré les nouvelles améliorations introduites dans la nouvelle norme, l'optimisation proposée reste applicable puisqu'elle considère le codec vidéo comme une boîte noire.

Système d'affichage

Le système d'affichage est considéré comme dépassant le cadre de cette thèse. Cependant, nous pensons que ce système devrait être étudié et ensuite être inclus dans toute étude d'application vidéo. En effet, il n'existe pas de vidéo décodée sans être affichée. Dans l'état de l'art, de nombreux modèles existent pour estimer la consommation énergétique du décodage vidéo sans prendre en compte le système d'affichage. Cependant, on ne peut

pas appliquer ces modèles dans des scénarios réels car dans ce cas le système d'affichage est présent.

Selon l'état de l'art [149], la consommation énergétique de l'affichage d'une vidéo sur un écran mobile dépend de paramètres liés à la fois à la séquence vidéo elle-même et à la technologie utilisée pour la fabrication de l'écran. Par exemple, la technologie récente des diodes électroluminescentes organiques à matrice active (AMOLED) consomme moins d'énergie par pixel que la technologie LCD qui l'a précédée [150]. De plus, l'écran est très dépendant de la vidéo à décoder en termes de luminosité, de couleurs, etc. Par conséquent, nous pensons que l'inclusion du système d'affichage dans l'étude de la consommation d'énergie des applications vidéo peut permettre une compréhension complète de la lecture vidéo.

DVFS par cœur

Dans le travail proposé, le nombre de cœurs et leurs fréquences d'horloge doivent être soigneusement configurés afin d'économiser de l'énergie. En procédant ainsi et en exploitant l'équilibrage de charge entre les cœurs GPP hétérogènes, les résultats montrent d'importantes économies d'énergie du décodage vidéo SW permettant de réduire l'écart avec la consommation énergétique du décodage vidéo HW. Il convient de rappeler que, dans notre travail, tous les cœurs d'un cluster GPP sont réglés à la même fréquence.

À notre avis, une mise à l'échelle par cœur DVFS sur des cœurs GPP hétérogènes est une technique prometteuse pour économiser l'énergie du décodage vidéo SW parallèle. En effet, les blocs composant une trame (*tranches*lice, *tile*, ou WPP) sont décodés en parallèle au niveau du cœur GPP, c'est-à-dire que chaque cœur GPP décode un bloc spécifique. En outre, ces blocs ont des complexités différentes. Il est donc pertinent d'ajuster la fréquence des GPP core pour correspondre à la complexité du bloc à décoder. Techniquement, les plateformes récentes supportent cette fonctionnalité. Ce travail est prévu pour être une de nos prochaines études.

Réglage analytique des coefficients du contrôleur PI

Dans la solution proposée pour optimiser la consommation énergétique du décodage vidéo SW, un contrôleur PI est utilisé pour ajuster la fréquence GPP. Les coefficients du contrôleur sont réglés empiriquement, et avant de commencer le décodage. Nous pensons que ces coefficients peuvent être calculés pendant l'exécution en utilisant des techniques

appropriées, comme l'apprentissage par renforcement. Nous pensons que cela est faisable car le rapport gain/coût de l'application de l'apprentissage par renforcement sur les systèmes embarqués s'est avéré acceptable [151].

La technique d'apprentissage par renforcement permet au contrôleur d'ajuster ses coefficients pour qu'ils correspondent le mieux possible à la charge de travail des images vidéo, par essais et erreurs, en utilisant le retour d'information de ses propres actions et expériences. Un autre avantage de cette technique est qu'elle ne nécessite pas une connaissance préalable de la trame vidéo à décoder. L'objectif est de fournir au décodeur vidéo des capacités d'apprentissage automatique pour prédire la charge de travail à venir sur la base de l'historique de la charge de travail.

Architecture à trois clusters

L'optimisation proposée a été appliquée sur l'architecture ARM big.LITTLE. Elle classe toute trame donnée en deux groupes : (i) les trames les plus complexes, et (ii) les trames les moins complexes. Ensuite, les trames du premier groupe sont soumises aux big cores tandis que les autres sont soumises aux LITTLE cores.

La solution proposée peut également être appliquée sur une architecture big.medium.LITTLE. Dans ce cas, toute image vidéo donnée est classée en trois groupes : (i) les images les plus complexes, (ii) les images moyennement complexes, et (iii) les images les moins complexes. Les trames du premier groupe sont soumises à de gros cœurs, les trames du deuxième groupe à des cœurs moyens et les trames du troisième groupe à de petits cœurs. Cette architecture montre une avancée importante en termes de performance et de consommation d'énergie. Les auteurs de [152] [153] montrent que l'architecture composée de trois clusters est plus économe en énergie que celle à deux clusters lorsque le programme est bien conçu pour exploiter le parallélisme entre les cœurs hétérogènes.

7.5.3 Modèles de programmation

Selon [154], les modèles de programmation OpenMP ou OmpSs ont un impact sur les performances et la consommation d'énergie sur l'architecture ARM big.LITTLE. Par conséquent, il convient de trouver et puis d'ajuster les paramètres pertinents pour obtenir le meilleur compromis entre performances et énergie.

Comme le codec HEVC est un processus parallèlisable, nous pensons qu'une nouvelle conception du parallélisme d'un framework du décodage vidéo (par exemple, Open-HEVC) permettra de gagner encore en consommation d'énergie.

Acronyms

AVC Advanced Video Coding. 4, 15, 17, 18, 41–44, 46, 53, 54, 144

CMOS Complementary Metal Oxide Semiconductor. vi, viii, x, xvi, xvii, 6, 15, 24–27, 34, 58, 140, 141, 145

DMR Deadline Miss Rate. 24

DPM Dynamic Power Management. 27, 28, 45

DSP Digital Signal Processor. x, 6, 10, 35, 38–40, 42, 43, 51, 142

DVFS Dynamic Voltage and Frequency Scaling. viii, x, 11, 14, 27–33, 44, 45, 47, 52–54, 107, 116, 117, 121, 130, 136, 141, 142, 172

FPS Frames Per Second. 23, 24

GPP General Purpose Processor. vii, viii, x–xvii, 5–7, 9–14, 25–29, 31–45, 47, 51, 53, 54, 57, 58, 60–64, 66–69, 71, 73, 74, 77–96, 99, 101, 103–109, 111–118, 120–127, 129–134, 136, 137, 141–145, 147, 148, 150, 151, 154–158, 160–170, 172, 194, 195

GPU Graphical Processing Unit. 6, 10, 35, 37–40, 51, 142

HDIP HardWare Intellectual Property. 5, 6, 9–12, 35–40, 43, 44, 51, 55, 57, 63, 64, 66, 67, 69, 71–74, 77, 78, 81–84, 89, 90, 92, 94–96, 98, 99, 102–104, 107, 132–134, 142–144, 148, 150, 155, 156, 168, 169

HEVC High Efficiency Video Coding. v–viii, xi–xv, 4–6, 10–17, 20, 21, 41–48, 52, 54, 71–82, 84–94, 96–104, 107, 108, 118, 120, 121, 129, 130, 133, 135, 138, 141, 144, 145, 152, 153, 155, 157–159, 167–169, 171, 173, 194

HW Hardware. v–viii, x, xi, xiii–xvii, 5, 7–9, 11–13, 33, 34, 36, 37, 40, 41, 43, 44, 55–58, 63, 64, 66–69, 71–75, 78–84, 89–104, 117, 118, 120, 121, 126, 128, 129, 131–136, 139, 142–144, 150, 152–159, 166–172, 194, 195

IPC Inter Processor Communication. x, 11, 12, 37–39, 41, 42, 51, 57, 63, 67, 81, 84, 143, 148, 150, 155

ISO/IEC International Standardization Organization / International Electrotechnical Commission. 16

MB Macro-block. 17–19, 21, 34, 41, 44, 49, 52–54, 141

MC Motion Compensation. 18, 37, 43, 45, 46

MPEG Moving Picture Experts Group. vi, x, 1, 11–13, 15–18, 20, 37, 40, 46, 49, 52, 54, 108, 135

OS Operating System. v, xiv, 1, 7, 8, 11, 40, 41, 51, 57, 64, 66, 73–75, 78, 81, 84, 89, 95, 101, 133, 144, 150, 152, 153, 168, 194, 195

PSNR Peak Signal to Noise Ratio. 23, 43, 51

QoS Quality of Service. 23, 43, 46, 67, 167

QP Quantization parameter. 19, 23, 42

SIMD Single Instruction Multiple Data. 38, 45, 46, 49

SoC System on Chip. 1, 11, 29, 33, 38, 43, 45, 48, 56, 61, 67, 73, 88, 96, 100, 132, 134, 135, 144, 147, 151, 159, 167, 170, 171, 194

SW Software. v–viii, xi, xiii–xvii, 6–9, 11–13, 28, 44, 45, 55–58, 64, 66–69, 71–73, 75–81, 84, 88–105, 107, 108, 117, 118, 121, 126, 129, 132–137, 142–144, 150, 152–160, 166–169, 171, 172, 194, 195

VIF Visual Information Fidelity. 23

VMAF Video Multi-method Assessment Method. 23

VMSSIM Video Structural Similarity Index Measure. 23

VVC Versatile Video Coding. 4, 6, 17, 135, 144, 171

WPP Wavefront Parallel Processing. x, 13, 20–22, 44, 48, 52–54, 78, 111, 118, 119, 122, 136, 141, 162, 172

BIBLIOGRAPHY

[1] F. Yao, A. Demers, and S. Shenker, « A scheduling model for reduced CPU energy », *in*: *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 1995, pp. 374–382, DOI: 10.1109/SFCS.1995.492493.

[2] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper - Cisco*, URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html (visited on 04/20/2019).

[3] *More Than 75% of Worldwide Video Viewing is Mobile eMarketer Newsroom*, URL: https://www.emarketer.com/newsroom/index.php/threequarters-video-viewing-mobile/ (visited on 04/20/2019).

[4] *Mobile Screen Resolution Stats Worldwide | StatCounter Global Stats*, URL: https://gs.statcounter.com/screen-resolution-stats/mobile/worldwide (visited on 09/30/2019).

[5] *Cisco public Cisco Visual Networking Index: Global Mobile Data Traffic The Cisco® Visual Networking Index (VNI) Global Mobile Data*, 2019.

[6] K. O'hara, A. Slayden Mitchell, and A. Vorbau, *Consuming Video on Mobile Devices*, 2007, ISBN: 9781595935939.

[7] *Mobile video viewing*, https://www.insivia.com/50-must-know-stats-about-video-animation-marketing-2013/, [Online; accessed: 20-April-2019].

[8] T. Stockhammer, « Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles », *in*: *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, San Jose, CA, USA: Association for Computing Machinery, 2011, pp. 133–144, ISBN: 9781450305181, DOI: 10.1145/1943552.1943572, URL: https://doi.org/10.1145/1943552.1943572.

[9] *How does the player figure out which video quality to request?*, URL: https://bitmovin.com/docs/player/faqs/how-does-the-player-figure-out-which-video-quality-to-request (visited on 06/15/2021).

[10] ITU-T Study Group et al., *Recommendation ITU-T H.262 (02/2000), MPEG-2*, ITU-T Study Group,"Series H: Audiovisual, multimedia systems: Infrastructure of audiovisual services–coding of moving video," in General Secretariat, and Telecom Radiocommunication (ITU-R) Standardization (ITU-T), sec.H.

[11] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, « Video coding with H.264/AVC: tools, performance, and complexity », *in*: *IEEE Circuits and Systems Magazine* 4.*1* (2004), pp. 7–28, ISSN: 1558-0830, DOI: 10.1109/MCAS.2004.1286980.

[12] R. Vani and M. Sangeetha, « Survey on H.264 standard », *in*: *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, vol. 86, Springer, Berlin, Heidelberg, 2012, pp. 397–410, ISBN: 9783642273162, DOI: 10.1007/978-3-642-27317-9_41, URL: https://link.springer.com/chapter/10.1007/978-3-642-27317-9%7B%5C_%7D41.

[13] ITU-T Study Group et al., *Recommendation ITU-T H.265 (02/2018), High Efficiency Video Coding*, ITU-T Study Group,"Series H: Audiovisual, multimedia systems: Infrastructure of audiovisual services–coding of moving video," in General Secretariat, and Telecom Radiocommunication (ITU-R) Standardization (ITU-T), sec.H.

[14] ITU-T Study Group et al., *Recommendation ITU-T H.266 (08/2020), High Efficiency Video Coding*, ITU-T Study Group,"Series H: Audiovisual, multimedia systems: Infrastructure of audiovisual services–coding of moving video," in General Secretariat, and Telecom Radiocommunication (ITU-R) Standardization (ITU-T), sec.H.

[15] *Growing Support of HEVC or H.265 Video on Mobile Devices | ScientiaMobile*, URL: https://www.scientiamobile.com/growing-support-of-hevc-or-h-265-video-on-mobile-devices/.

[16] B. Bross, J. Chen, J.R. Ohm, G.J. Sullivan, and Y.K. Wang, « Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC) », *in*: *Proceedings of the IEEE* (2021), pp. 1–31, DOI: 10.1109/JPROC.2020.3043399.

[17] J.R. Ohm, G.J. Sullivan, H. Schwarz, T.K. Tan, and T. Wiegand, « Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC) », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22 (12 2012), pp. 1669–1684, ISSN: 10518215, DOI: 10.1109/TCSVT.2012.2221192.

[18] C. Herglotz, D. Springer, M. Reichenbach, B. Stabernack, and A. Kaup, « Modeling the Energy Consumption of the HEVC Decoding Process », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 28.*1* (Jan. 2018), pp. 217–229, ISSN: 1558-2205, DOI: 10.1109/TCSVT.2016.2598705.

[19] M. Broussely and G. Archdale, « Li-ion batteries and portable power source prospects for the next 5–10 years », *in*: *Journal of Power Sources* 136.*2* (2004), Selected papers presented at the International Power Sources Symposium, pp. 386–394, ISSN: 0378-7753, DOI: https://doi.org/10.1016/j.jpowsour.2004.03.031, URL: http://www.sciencedirect.com/science/article/pii/S037877530400343X.

[20] M. Tikekar, C. Huang, C. Juvekar, V. Sze, and A.P. Chandrakasan, « A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications », *in*: *IEEE Journal of Solid-State Circuits* 49.*1* (Jan. 2014), pp. 61–72, ISSN: 1558-173X, DOI: 10.1109/JSSC.2013.2284362.

[21] F. Amish and E. Bourennane, « Fully pipelined real time hardware solution for High Efficiency Video Coding (HEVC) intra prediction », *in*: *Journal of Systems Architecture* 64 (2016), Real-Time Signal Processing in Embedded Systems, pp. 133–147, ISSN: 1383-7621, DOI: https://doi.org/10.1016/j.sysarc.2015.10.002, URL: http://www.sciencedirect.com/science/article/pii/S1383762115001162.

[22] M. Horowitz, « 1.1 Computing's energy problem (and what we can do about it) », *in*: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014, pp. 10–14, DOI: 10.1109/ISSCC.2014.6757323.

[23] K. Xu, T.M. Liu, J.I. Guo, and C.S. Choy, « Methods for Power/Throughput/Area Optimization of H.264/AVC Decoding », *in*: *Journal of Signal Processing Systems* 60.*1* (July 2010), pp. 131–145, ISSN: 1939-8115, DOI: 10.1007/s11265-009-0408-6, URL: https://doi.org/10.1007/s11265-009-0408-6.

[24] B. Moyer and Y. Watanabe, « Chapter 13 - Hardware Accelerators », *in*: *Real World Multicore Embedded Systems*, ed. by Bryon Moyer, Oxford: Newnes, 2013, pp. 447–480, ISBN: 978-0-12-416018-7, DOI: https://doi.org/10.1016/B978-0-12-416018-7.00013-4, URL: http://www.sciencedirect.com/science/article/pii/B9780124160187000134.

[25] K. Choi and E.S. Jang, « Leveraging Parallel Computing in Modern Video Coding Standards », *in*: *IEEE MultiMedia* 19.*3* (July 2012), pp. 7–11, ISSN: 1941-0166, DOI: 10.1109/MMUL.2012.36.

[26] M.P. Singh and M.K. Jain, *Evolution of Processor Architecture in Mobile Phones*, 2014, pp. 975–8887.

[27] D. Etiemble, *45-year CPU evolution: one law and two equations.*

[28] T. Kuroda, « CMOS design challenges to power wall », *in*: *Digest of Papers. Microprocesses and Nanotechnology 2001. 2001 International Microprocesses and Nanotechnology Conference (IEEE Cat. No.01EX468)*, 2001, pp. 6–7, DOI: 10.1109/IMNC.2001.984030.

[29] T.D. Burd and R.W. Brodersen, « Energy efficient CMOS microprocessor design », *in*: *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, vol. 1, Jan. 1995, 288–297 vol.1, DOI: 10.1109/HICSS.1995.375385.

[30] H. Blume, J. von Livonius, L. Rotenberg, T.G. Noll, H. Bothe, and J. Brakensiek, « OpenMP-based parallelization on an MPCore multiprocessor platform – A performance and power analysis », *in*: *Journal of Systems Architecture* 54.*11* (2008), Embedded Systems: Architectures, Modeling and Simulation, pp. 1019–1029, ISSN: 1383-7621, DOI: https://doi.org/10.1016/j.sysarc.2008.04.001, URL: http://www.sciencedirect.com/science/article/pii/S1383762108000568.

[31] R. Sjoberg, Y. Chen, A. Fujibayashi, M.M. Hannuksela, J. Samuelsson, T.K. Tan, Y. Wang, and S. Wenger, « Overview of HEVC High-Level Syntax and Reference Picture Management », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22.*12* (Dec. 2012), pp. 1858–1870, ISSN: 1558-2205, DOI: 10.1109/TCSVT.2012.2223052.

[32] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M.A. Horowitz, « Convolution Engine: Balancing Efficiency & Flexibility in Specialized Computing », *in*: *SIGARCH Comput. Archit. News* 41.*3* (June 2013), pp. 24–35, ISSN: 0163-5964, DOI: 10.1145/2508148.2485925, URL: http://doi.acm.org/10.1145/2508148.2485925.

[33] R. Hameed, W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B.C. Lee, S. Richardson, C. Kozyrakis, and M. Horowitz, « Understanding Sources of Inefficiency in General-purpose Chips », *in*: *SIGARCH Comput. Archit. News* 38.*3* (June 2010), pp. 37–47, ISSN: 0163-5964, DOI: 10.1145/1816038.1815968, URL: http://doi.acm.org/10.1145/1816038.1815968.

[34] M. Vuletic, *Unifying software and hardware of multithreaded reconfigurable applications within operating system processes*, tech. rep., EPFL, 2006.

[35] L. Perneel, H. Fayyad-Kazan, and M. Timmerman, « Can Android be used for real-time purposes? », *in*: *2012 International Conference on Computer Systems and Industrial Informatics*, IEEE, 2012, pp. 1–6.

[36] B. Ouni, C. Belleudy, S. Bilavarn, and E. Senn, « Embedded operating systems energy overhead », *in*: *Proceedings of the 2011 Conference on Design Architectures for Signal Image Processing (DASIP)*, 2011, pp. 1–6, DOI: 10.1109/DASIP.2011.6136853.

[37] Y. Benmoussa, « Performance and Energy Consumption Characterization and Modeling of Video Decoding on Multi-core Heterogenous SoC and their Applications », PhD thesis, Université de Bretagne Occidentale, 2015.

[38] S. Wang, G. Ananthanarayanan, Y. Zeng, N. Goel, A. Pathania, and T. Mitra, « High-Throughput CNN Inference on Embedded ARM Big.LITTLE Multicore Processors », *in*: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.*10* (2020), pp. 2254–2267, DOI: 10.1109/TCAD.2019.2944584.

[39] Hongsuk Chung, Munsik Kang, and Hyun-Duk Cho, *Heterogeneous Multi-Processing Solution of Exynos 5 Octa with ARM® big.LITTLE™ Technology.*

[40] S.K. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser, « Fast and accurate hybrid power estimation methodology for embedded systems », *in*: *Pro-*

ceedings of the 2011 Conference on Design Architectures for Signal Image Processing (DASIP), 2011, pp. 1–7, DOI: 10.1109/DASIP.2011.6136852.

[41] Z. Lu, J. Lach, M. Stan, and K. Skadron, « Reducing multimedia decode power using feedback control », *in*: *Proceedings 21st International Conference on Computer Design*, 2003, pp. 489–496, DOI: 10.1109/ICCD.2003.1240945.

[42] *About MPEG – MPEG*, URL: https://www.mpegstandards.org/about-mpeg/ (visited on 03/11/2021).

[43] D. Le Gall, « MPEG: A Video Compression Standard for Multimedia Applications », *in*: *Commun. ACM* 34.*4* (Apr. 1991), pp. 46–58, ISSN: 0001-0782, DOI: 10.1145/103085.103090, URL: https://doi.org/10.1145/103085.103090.

[44] *Bandwidth and Storage Calculator | StarDot Technologies*, URL: http://stardot.com/bandwidth-and-storage-calculator (visited on 06/01/2021).

[45] Y. Tew and K. Wong, « An Overview of Information Hiding in H.264/AVC Compressed Video », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 24.*2* (2014), pp. 305–319, DOI: 10.1109/TCSVT.2013.2276710.

[46] D.J. Le Gall, « The MPEG video compression algorithm », *in*: *Signal Processing: Image Communication* 4.*2* (1992), pp. 129–140, ISSN: 0923-5965, DOI: https://doi.org/10.1016/0923-5965(92)90019-C, URL: https://www.sciencedirect.com/science/article/pii/092359659290019C.

[47] T. Sikora, « MPEG digital video-coding standards », *in*: *IEEE Signal Processing Magazine* 14.*5* (1997), pp. 82–100, DOI: 10.1109/79.618010.

[48] N. Ahmed, T. Natarajan, and K.R. Rao, « Discrete Cosine Transform », *in*: *IEEE Transactions on Computers* C-23.*1* (1974), pp. 90–93, DOI: 10.1109/T-C.1974.223784.

[49] V. Sze and M. Budagavi, « A comparison of CABAC throughput for HEVC/H.265 VS. AVC/H.264 », *in*: *SiPS 2013 Proceedings*, 2013, pp. 165–170, DOI: 10.1109/SiPS.2013.6674499.

[50] E. Kalali, Y. Adibelli, and I. Hamzaoglu, « A high performance and low energy intra prediction hardware for HEVC video decoding », *in*: *Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing*, Oct. 2012, pp. 1–8.

[51] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, « An Overview of Tiles in HEVC », *in*: *IEEE Journal of Selected Topics in Signal Processing* 7.*6* (2013), pp. 969–977, DOI: 10.1109/JSTSP.2013.2271451.

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

[53] R. Koenen, « Profiles and levels in MPEG-4: Approach and overview », *in*: *Signal Processing: Image Communication* 15.*4* (2000), pp. 463–478, ISSN: 0923-5965, DOI: https://doi.org/10.1016/S0923-5965(99)00058-2, URL: https://www.sciencedirect.com/science/article/pii/S0923596599000582.

[54] D. Flynn, D. Marpe, M. Naccari, T. Nguyen, C. Rosewarne, K. Sharman, J. Sole, and J. Xu, « Overview of the Range Extensions for the HEVC Standard: Tools, Profiles, and Performance », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 26.*1* (2016), pp. 4–19, DOI: 10.1109/TCSVT.2015.2478707.

[55] M. Vranjes, S. Rimac-Drlje, and K. Grgic, « Locally averaged PSNR as a simple objective Video Quality Metric », *in*: *2008 50th International Symposium ELMAR*, vol. 1, 2008, pp. 17–20.

[56] M.O. Martínez-Rach, P. Piñol, O.M. López, M.P. Malumbres, J. Oliver, and C.T. Calafate, « On the Performance of Video Quality Assessment Metrics under Different Compression and Packet Loss Scenarios », *in*: (2014), DOI: 10.1155/2014/743604, URL: http://dx.doi.org/10.1155/2014/743604.

[57] M. Song, D. Tjondronegoro, and M. Docherty, « Saving Bitrate vs. Pleasing Users: Where is the Break-Even Point in Mobile Video Quality? », *in*: *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, Scottsdale, Arizona, USA: Association for Computing Machinery, 2011, pp. 403–412, ISBN: 9781450306164, DOI: 10.1145/2072298.2072351, URL: https://doi.org/10.1145/2072298.2072351.

[58] Video Quality Experts Group et al., « Final report from the video quality experts group on the validation of objective models of video quality assessment, phase II », *in*: *2003 VQEG* (2003).

[59] Q. Huynh-Thu and M. Ghanbari, « Scope of validity of PSNR in image/video quality assessment », *in*: *Electronics Letters* 44 (13 2008), p. 800, ISSN: 00135194, DOI: 10.1049/el:20080522, URL: https://digital-library.theiet.org/content/journals/10.1049/el_20080522.

[60] S. Winkler and P. Mohandas, « The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics », *in*: *IEEE Transactions on Broadcasting* 54.*3* (2008), pp. 660–668, DOI: 10.1109/TBC.2008.2000733.

[61] H.R. Sheikh and A.C. Bovik, « Image information and visual quality », *in*: *IEEE Transactions on Image Processing* 15.*2* (2006), pp. 430–444, DOI: 10.1109/TIP.2005.859378.

[62] Z. Wang, L. Lu, and A.C. Bovik, « Video quality assessment based on structural distortion measurement », *in*: *Signal Processing: Image Communication* 19.*2* (2004), pp. 121–132, ISSN: 0923-5965, DOI: https://doi.org/10.1016/S0923-5965(03)00076-6, URL: https://www.sciencedirect.com/science/article/pii/S0923596503000766.

[63] B. García, L. López-Fernández, F. Gortázar, and M. Gallego, « Practical evaluation of VMAF perceptual video quality for webRTC applications », *in*: *Electronics (Switzerland)* 8 (8 Aug. 2019), p. 854, ISSN: 20799292, DOI: 10.3390/electronics8080854, URL: https://github.com/Netflix/vmaf.

[64] C.C. Wüst, L. Steffens, W.F.J. Verhaegh, R.J. Bril, and C. Hentschel, « QoS control strategies for high-quality video processing », *in*: vol. 30, Springer, May 2005, pp. 7–29, DOI: 10.1007/s11241-005-0502-1, URL: https://link.springer.com/article/10.1007/s11241-005-0502-1.

[65] W. Yuan and K. Nahrstedt, « Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems », *in*: *SIGOPS Oper. Syst. Rev.* 37.*5* (Oct. 2003), pp. 149–163, ISSN: 0163-5980, DOI: 10.1145/1165389.945460, URL: https://doi.org/10.1145/1165389.945460.

[66] G. Büyüközkan and S. Güleryüz, « Multi Criteria Group Decision Making Approach for Smart Phone Selection Using Intuitionistic Fuzzy TOPSIS », *in*: *International Journal of Computational Intelligence Systems* 9.*4* (2016), pp. 709–725, DOI: 10.1080/18756891.2016.1204119, eprint: https://doi.org/10.1080/18756891.2016.1204119, URL: https://doi.org/10.1080/18756891.2016.1204119.

[67] J.A. Butts and G.S. Sohi, « A static power model for architects », *in*: *Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000*, 2000, pp. 191–201, DOI: 10.1109/MICRO.2000.898070.

[68] J.A. Butts and G.S. Sohi, « A static power model for architects », *in*: *Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000*, 2000, pp. 191–201, DOI: 10.1109/MICRO.2000.898070.

[69] N.S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan, *Leakage Current: Moore's Law Meets Static Power*, 2003, DOI: 10.1109/MC.2003.1250885.

[70] N.H.E. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*, Pearson Education India, 2015.

[71] K. Moiseev, A. Kolodny, and S. Wimer, « Timing-aware power-optimal ordering of signals », *in*: *ACM Transactions on Design Automation of Electronic Systems* 13 (4 Sept. 2008), pp. 1–17, ISSN: 10844309, DOI: 10.1145/1391962.1391973, URL: https://dl.acm.org/doi/10.1145/1391962.1391973.

[72] T. Mudge, « Power: a first-class architectural design constraint », *in*: *Computer* 34.*4* (2001), pp. 52–58, DOI: 10.1109/2.917539.

[73] J.R. Lorch and A.J. Smith, « Improving dynamic voltage scaling algorithms with PACE », *in*: vol. 29, Association for Computing Machinery (ACM), June 2001, pp. 50–61, DOI: 10.1145/384268.378429, URL: https://dl.acm.org/doi/10.1145/384268.378429.

[74] D. Meisnery, C.M. Sadlerz, L.A. Barrosoz, W.D. Weberz, and T.F. Wenischy, « Power management of Online Data-Intensive services », *in*: ACM Press, 2011, pp. 319–330, ISBN: 9781450304726, DOI: 10.1145/2000064.2000103, URL: http://portal.acm.org/citation.cfm?doid=2000064.2000103.

[75] T. Simunic, L. Benini, P. Glynn, and G.D. Micheli, « Dynamic power management for portable systems », *in*: ACM, 2000, pp. 11–19, DOI: 10.1145/345910.345914, URL: http://portal.acm.org/citation.cfm?doid=345910.345914.

[76] Y. Hwang, S. Ku, and K. Chung, « A predictive dynamic power management technique for embedded mobile devices », *in*: *IEEE Transactions on Consumer Electronics* 56.*2* (2010), pp. 713–719, DOI: 10.1109/TCE.2010.5505992.

[77] Y.H. Lu and G.D. Micheli, « Comparing system-level power management policies », *in*: *IEEE Design and Test of Computers* 18 (2 Mar. 2001), pp. 10–19, ISSN: 07407475, DOI: 10.1109/54.914592.

[78] D. Suleiman, M. Ibrahim, and I. Hamarash, « Dynamic voltage frequency scaling (DVFS) for microprocessors power and energy reduction », *in*: *4th International Conference on Electrical and Electronics Engineering*, vol. 12, 2005.

[79] V. Gutnik and A.P. Chandrakasan, « Embedded power supply for low-power DSP », *in*: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 5.*4* (1997), pp. 425–435, DOI: 10.1109/92.645069.

[80] K. Choi, K. Dantu, W.C. Cheng, and M. Pedram, « Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder », *in*: *Proceedings of the 2002 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '02, San Jose, California: Association for Computing Machinery, 2002, pp. 732–737, ISBN: 0780376072, DOI: 10.1145/774572.774680, URL: https://doi.org/10.1145/774572.774680.

[81] K. Choi, R. Soma, and M. Pedram, « Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times », *in*: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24.*1* (2005), pp. 18–28, DOI: 10.1109/TCAD.2004.839485.

[82] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, « Low-Power CMOS Digital Design », *in*: *Low-Power CMOS Design*, Wiley-IEEE Press, Jan. 1998, pp. 36–46, ISBN: 9780470545058, DOI: 10.1109/9780470545058.part1.

[83] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, and A. Ramirez, « Parallel scalability of H. 264 », *in*: *Proceedings of the first Workshop on Programmability Issues for Multi-Core Computers*, 2008.

[84] A.P. Chandrakasan and R.W. Brodersen, « Minimizing power consumption in digital CMOS circuits », *in*: *Proceedings of the IEEE* 83.*4* (1995), pp. 498–523, DOI: 10.1109/5.371964.

[85] M. Horowitz, E. Alon, D. Patil, S. Naffziger, Rajesh Kumar, and K. Bernstein, « Scaling, power, and the future of CMOS », *in*: *IEEE InternationalElectron De-*

vices Meeting, 2005. IEDM Technical Digest. 2005, 7 pp.–15, DOI: 10.1109/IEDM.2005.1609253.

[86] *MediaCodec | Android Developers*, URL: https://developer.android.com/reference/android/media/MediaCodec (visited on 01/09/2019).

[87] M. Wakin, J.N. Laska, M.F. Duarte, D. Baron, S. Sarvotham, D. Takhar, K.F. Kelly, and R.G. Baraniuk, « Compressive imaging for video representation and coding », *in*: *Picture Coding Symposium*, vol. 1, 13, 2006.

[88] *Development of the VPU | Jon Peddie Research*, URL: https://www.jonpeddie.com/blog/development-of-the-vpu/ (visited on 09/12/2019).

[89] J. Golston, S. Arora, and R. Reddy, « Optimized video decoder architecture for TMS320C64x DSP generation », *in*: *Image and Video Communications and Processing 2003*, ed. by Bhaskaran Vasudev, T. Russell Hsing, Andrew G. Tescher, and Touradj Ebrahimi, vol. 5022, International Society for Optics and Photonics, SPIE, 2003, pp. 719–726, DOI: 10.1117/12.476330, URL: https://doi.org/10.1117/12.476330.

[90] L. Li, C. Sau, T. Fanni, J. Li, T. Viitanen, F. Christophe, F. Palumbo, L. Raffo, H. Huttunen, J. Takala, and S.S. Bhattacharyya, « An integrated hardware/software design methodology for signal processing systems », *in*: *Journal of Systems Architecture* 93 (2019), pp. 1–19, ISSN: 1383-7621, DOI: https://doi.org/10.1016/j.sysarc.2018.12.010, URL: http://www.sciencedirect.com/science/article/pii/S1383762118301735.

[91] G.J.M. Smit, A.B.J. Kokkeler, P.T. Wolkotte, and M.D. van de Burgwal, « Multi-Core Architectures and Streaming Applications », *in*: *Proceedings of the 2008 International Workshop on System Level Interconnect Prediction*, SLIP '08, Newcastle, United Kingdom: Association for Computing Machinery, 2008, pp. 35–42, ISBN: 9781595939180, DOI: 10.1145/1353610.1353618, URL: https://doi.org/10.1145/1353610.1353618.

[92] M. Macedonia, « The GPU enters computing's mainstream », *in*: *Computer* 36.*10* (2003), pp. 106–108, DOI: 10.1109/MC.2003.1236476.

[93] G. Shen, G.P. Gao, S. Li, H.Y. Shum, and Y.Q. Zhang, « Accelerate video decoding with generic GPU », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 15.*5* (2005), pp. 685–693, DOI: 10.1109/TCSVT.2005.846440.

[94] B. Wang, D.F. de Souza, M. Alvarez-Mesa, C.C. Chi, B. Juurlink, A. Ilic, N. Roma, and L. Sousa, « Highly parallel HEVC decoding for heterogeneous systems with CPU and GPU », *in*: *Signal Processing: Image Communication* 62 (2018), pp. 93–105, ISSN: 0923-5965, DOI: https://doi.org/10.1016/j.image.2017.12.009, URL: https://www.sciencedirect.com/science/article/pii/S0923596517302631.

[95] D.F. de Souza, A. Ilic, N. Roma, and L. Sousa, « GPU-assisted HEVC intra decoder », *in*: *Journal of Real-Time Image Processing* 12 (2 Aug. 2016), pp. 531–547, ISSN: 18618200, DOI: 10.1007/s11554-015-0519-1.

[96] J. Eyre, « The digital signal processor Derby », *in*: *IEEE Spectrum* 38.*6* (2001), pp. 62–68, DOI: 10.1109/6.925269.

[97] Texas Instruments and Incorporated Spru, *TMS320 DSP/BIOS v5.42 (Rev. I)*, 2012, URL: http://processors.wiki.ti.com/index.php/CCSv5.

[98] S.L. Tsao and S.Y. Lee, « Performance evaluation of inter-processor communication for an embedded heterogeneous multi-core processor », *in*: *Journal of information science and engineering* 28.*3* (2012), pp. 537–554.

[99] M. Tikekar, C. Huang, C. Juvekar, V. Sze, and A.P. Chandrakasan, « A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications », *in*: *IEEE Journal of Solid-State Circuits* 49.*1* (Jan. 2014), pp. 61–72, ISSN: 1558-173X, DOI: 10.1109/JSSC.2013.2284362.

[100] C. Meenderinck, A. Azevedo, B. Juurlink, M. Alvarez Mesa, and A. Ramirezex, « Parallel scalability of video decoders », *in*: *Journal of Signal Processing Systems* 57 (2 Nov. 2009), pp. 173–194, ISSN: 19398018, DOI: 10.1007/s11265-008-0256-9, URL: https://link.springer.com/article/10.1007/s11265-008-0256-9.

[101] U.S. Gorgonio, H.R.B. Cunha, E.X. de L. Filho, S.O.D Luiz, A. Perkusich, and M.R.A. Morais, « Application Profiling in a Dual-Core Platform », *in*: *2008 Digest of Technical Papers - International Conference on Consumer Electronics*, 2008, pp. 1–2, DOI: 10.1109/ICCE.2008.4588103.

[102] C.N. Chiu, C.T. Tseng, and C.J. Tsai, « Tightly-coupled MPEG-4 video encoder framework on asymmetric dual-core platforms », *in*: *2005 IEEE International Symposium on Circuits and Systems*, 2005, 2132–2135 Vol. 3, DOI: 10.1109/ISCAS.2005.1465041.

[103] S.O. Dornellas Luiz, G. de Moura Vasconcelos, and L.D. da Silva, « Formal Specification of DSP Gateway for Data Transmission between Processor Cores of OMAP Platform », *in*: *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, Fortaleza, Ceara, Brazil: Association for Computing Machinery, 2008, pp. 1545–1549, ISBN: 9781595937537, DOI: 10.1145/1363686.1364046, URL: https://doi.org/10.1145/1363686.1364046.

[104] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz, « GPP vs DSP: A Performance/Energy Characterization and Evaluation of Video Decoding », *in*: *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug. 2013, pp. 273–282, DOI: 10.1109/MASCOTS.2013.35.

[105] M. Mody, « HEVC video encoder decoder architecture for multi-cores », *in*: *2014 International Conference on Signal Processing and Communications (SPCOM)*, 2014, pp. 1–5, DOI: 10.1109/SPCOM.2014.6983918.

[106] F. Bossen, B. Bross, K. Suhring, and D. Flynn, « HEVC Complexity and Implementation Analysis », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22.*12* (Dec. 2012), pp. 1685–1696, ISSN: 1558-2205, DOI: 10.1109/TCSVT.2012.2221255.

[107] N. Sidaty, J. Heulot, W. Hamidouche, E. Nogues, M. Pelcat, and D. Menard, « Reducing computational complexity in HEVC decoder for mobile energy saving », *in*: *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug. 2017, pp. 1026–1030, DOI: 10.23919/EUSIPCO.2017.8081363.

[108] C. Herglotz and A. Kaup, « Decoding Energy Estimation of an HEVC Hardware Decoder », *in*: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5, DOI: 10.1109/ISCAS.2018.8350964.

[109] M. Holliman and Y.K. Chen, « MPEG decoding workload characterization », *in*: *Proc. of Workshop on Computer Architecture Evaluation using Commercial Workloads*, 2003, pp. 23–34.

[110] O. Silvén and T. Rintaluoma, « Energy Efficiency of Video Decoder Implementations », *in*: *Mobile Phone Programming: Application to Wireless Networking*, ed. by Frank H. P. Fitzek and Frank Reichert, Dordrecht: Springer Netherlands, 2007, pp. 421–439, ISBN: 978-1-4020-5969-8, DOI: 10.1007/978-1-4020-5969-8_23, URL: https://doi.org/10.1007/978-1-4020-5969-8_23.

[111] E. Nogues, R. Berrada, M. Pelcat, D. Menard, and E. Raffin, « A DVFS based HEVC decoder for energy-efficient software implementation on embedded processors », *in*: *2015 IEEE International Conference on Multimedia and Expo (ICME)*, 2015, pp. 1–6, DOI: 10.1109/ICME.2015.7177406.

[112] E. Raffin, E. Nogues, W. Hamidouche, S. Tomperi, M. Pelcat, and D. Menard, « Low power HEVC software decoder for mobile devices », *in*: *Journal of Real-Time Image Processing* 12.*2* (Aug. 2016), pp. 495–507, ISSN: 1861-8219, DOI: 10.1007/s11554-015-0512-8, URL: https://doi.org/10.1007/s11554-015-0512-8.

[113] Y. Duan, J. Sun, L. Yan, K. Chen, and Z. Guo, « Novel Efficient HEVC Decoding Solution on General-Purpose Processors », *in*: *IEEE Transactions on Multimedia* 16.*7* (2014), pp. 1915–1928, DOI: 10.1109/TMM.2014.2337834.

[114] E. Nogues, A. Mercat, F. Arrestier, M. Pelcat, and D. Menard, « Convex Energy Optimization of Streaming Applications for MPSoCs », *in*: (2019), pp. 1557–1561, DOI: 10.1109/ICASSP.2019.8682317.

[115] C.C. Chi, M. Alvarez Mesa, and B. Juurlink, « Low-Power High-Efficiency Video Decoding Using General-Purpose Processors », *in*: *ACM Trans. Archit. Code Optim.* 11.*4* (Jan. 2015), ISSN: 1544-3566, DOI: 10.1145/2685551, URL: https://doi.org/10.1145/2685551.

[116] E. Nogues, S. Holmbacka, M. Pelcat, D. Menard, and J. Lilius, « Power-Aware HEVC Decoding with Tunable Image Quality », *in*: *IEEE International Workshop on Signal Processing Systems*, Belfast, United Kingdom, Oct. 2014, URL: https://hal.archives-ouvertes.fr/hal-01078566.

[117] E. Nogues, D. Menard, and M. Pelcat, « Algorithmic-Level Approximate Computing Applied to Energy Efficient HEVC Decoding », *in*: *IEEE Transactions on Emerging Topics in Computing* 7.*1* (2019), pp. 5–17, DOI: 10.1109/TETC.2016.2593644.

[118] E. Nogues, E. Raffin, M. Pelcat, and D. Menard, « A Modified HEVC Decoder for Low Power Decoding », *in*: *Proceedings of the 12th ACM International Conference on Computing Frontiers*, CF '15, Ischia, Italy: Association for Computing Machinery, 2015, ISBN: 9781450333580, DOI: 10.1145/2742854.2747284, URL: https://doi.org/10.1145/2742854.2747284.

[119] S. Yoo and E.S. Ryu, « Parallel HEVC decoding with asymmetric mobile multicores », *in*: *Multimedia Tools and Applications* 76 (16 Aug. 2017), pp. 17337–17352, ISSN: 1380-7501, DOI: 10.1007/s11042-016-4269-2, URL: http://link.springer.com/10.1007/s11042-016-4269-2.

[120] H.J. Roh, S.W. Han, and E.S. Ryu, « Prediction complexity-based HEVC parallel processing for asymmetric multicores », *in*: *Multimedia Tools and Applications* 76 (23 Dec. 2017), pp. 25271–25284, ISSN: 15737721, DOI: 10.1007/s11042-017-4413-7.

[121] H. Baik and H. Song, « A complexity-based adaptive tile partitioning algorithm for HEVC decoder parallelization », *in*: *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 4298–4302, DOI: 10.1109/ICIP.2015.7351617.

[122] R. Rodríguez-Sánchez and E.S. Quintana-Ortí, « Architecture-aware optimization of an HEVC decoder on asymmetric multicore processors », *in*: *Journal of Real-Time Image Processing* 13 (1 Mar. 2017), pp. 25–38, ISSN: 1861-8200, DOI: 10.1007/s11554-016-0606-y, URL: http://link.springer.com/10.1007/s11554-016-0606-y.

[123] C.C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, « Parallel Scalability and Efficiency of HEVC Parallelization Approaches », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22.*12* (2012), pp. 1827–1838, DOI: 10.1109/TCSVT.2012.2223056.

[124] M. Koziri, D. Zacharis, I. Katsavounidis, and N. Bellas, « Implementation of the AVS video decoder on a heterogeneous dual-core SIMD processor », *in*: *IEEE Transactions on Consumer Electronics* 57.*2* (2011), pp. 673–681, DOI: 10.1109/TCE.2011.5955207.

[125] Y. Tan, P. Malani, Q. Qiu, and QingWu, « Workload prediction and dynamic voltage scaling for MPEG decoding », *in*: *Asia and South Pacific Conference on Design Automation, 2006.* 2006, 6 pp.-, DOI: 10.1109/ASPDAC.2006.1594802.

[126] G.B. Dantzig and P. Wolfe, « Decomposition principle for linear programs », *in*: *Operations research* 8.*1* (1960), pp. 101–111.

[127] X. Li and D. Yeung, « Application-Level Correctness and its Impact on Fault Tolerance », *in*: *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, 2007, pp. 181–192, DOI: 10.1109/HPCA.2007.346196.

[128] T. Miyazaki, C. Sakamoto, M. Kuwayama, K. Saisho, and A. Fukuda, « Parallel Pthread library (PPL): user-level thread library with parallelism and portability », *in*: *Proceedings Eighteenth Annual International Computer Software and Applications Conference (COMPSAC 94)*, 1994, pp. 301–306, DOI: 10.1109/CMPSAC.1994.342788.

[129] *APQ8094 | Qualcomm*, URL: https://www.qualcomm.com/products/apq8094 (visited on 05/03/2018).

[130] *ODROID-XU3 – ODROID*, URL: https://www.hardkernel.com/shop/odroid-xu3/ (visited on 12/01/2017).

[131] *Qualcomm® Robotics RB3 Development Kit*, URL: https://www.qualcomm.com/products/qualcomm-robotics-rb3-platform (visited on 07/14/2020).

[132] *Intrinsyc Introduces Snapdragon 810 Powered Mobile and Tablet MDPs, DragonBoard Development Kit - CNX Software*, URL: https://www.cnx-software.com/2014/11/28/intrinsyc-introduces-snapdragon-810-powered-mobile-and-tablet-mdps-dragonboard-development-kit/ (visited on 06/22/2018).

[133] *ODROID-XU3 review, sort of*, URL: https://www.dedoimedo.com/computers/odroid-xu3-quick-review.html (visited on 06/22/2018).

[134] *Qualcomm® Robotics RB3 Development Kit - Qualcomm Developer Network*, URL: https://developer.qualcomm.com/qualcomm-robotics-rb3-kit (visited on 06/22/2018).

[135] *Keysight Technologies N6700 Modular Power System Family Low-Proile Mainframes for ATE: N6700B, N6701A, N6702A DC Power Analyzer Mainframe for RD: N6705B DC Power Modules: N6731B-N6784A Software for DC Power: 14585A*.

[136] Y. Benmoussa, E. Senn, J. Boukhobza, M. Lanoe, and D. Benazzouz, « OpenPEOPLE, A Collaborative Platform for Remote Accurate Measurement and Evaluation of Embedded Systems Power Consumption », *in*: *2014 IEEE 22nd International Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems*, Sept. 2014, pp. 498–501, DOI: 10.1109/MASCOTS.2014.72.

[137] *GitHub - OpenHEVC/openHEVC at ffmpeg_update*, URL: https://github.com/OpenHEVC/openHEVC/tree/ffmpeg_update (visited on 07/07/2018).

[138] *Download FFmpeg*, URL: https://ffmpeg.org/download.html (visited on 03/01/2018).

[139] F. Bossen, « Common test conditions and software reference configurations », *in*: *JCTVC-L1100* 12 (2013).

[140] *Jellyfish Bitrate Test Files*, URL: http://jell.yfish.us/ (visited on 03/01/2018).

[141] Xiph.org, *Xiph.org :: Derf's Test Media Collection*, 2015, URL: http://media.xiph.org/video/derf/ (visited on 11/28/2017).

[142] *Optimize for Doze and App Standby*, URL: https://developer.android.com/training/monitoring-device-state/doze-standby (visited on 10/01/2018).

[143] J.A. Butts and G.S. Sohi, « A static power model for architects », *in*: *Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000*, IEEE, pp. 191–201, ISBN: 0-7695-0924-X, DOI: 10.1109/MICRO.2000.898070, URL: http://ieeexplore.ieee.org/document/898070/.

[144] P. Nilsson, « Arithmetic reduction of the static power consumption in nanoscale CMOS », *in*: *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems*, 2006, pp. 656–659, ISBN: 1424403952, DOI: 10.1109/ICECS.2006.379874.

[145] N.A. Kudryashov, « Logistic function as solution of many nonlinear differential equations », *in*: *Applied Mathematical Modelling* 39.*18* (2015), pp. 5733–5742, ISSN: 0307-904X, DOI: https://doi.org/10.1016/j.apm.2015.01.048, URL: https://www.sciencedirect.com/science/article/pii/S0307904X15000517.

[146] A.C. Bavier, A.B. Montz, and L.L. Peterson, « Predicting MPEG Execution Times », *in*: *SIGMETRICS Perform. Eval. Rev.* 26.*1* (June 1998), pp. 131–140, ISSN: 0163-5999, DOI: 10.1145/277858.277892, URL: https://doi.org/10.1145/277858.277892.

[147] D. Valério and J.S. da Costa, « Tuning of fractional PID controllers with Ziegler–Nichols-type rules », *in*: *Signal Processing* 86.*10* (2006), Special Section: Fractional Calculus Applications in Signals and Systems, pp. 2771–2784, ISSN: 0165-1684, DOI: https://doi.org/10.1016/j.sigpro.2006.02.020, URL: https://www.sciencedirect.com/science/article/pii/S0165168406000624.

[148] *scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation*, URL: https://scikit-learn.org/stable/ (visited on 08/11/2021).

[149] S. Chakraborty and D.K.Y. Yau, « Predicting energy consumption of MPEG video playback on handhelds », *in*: *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1, 2002, 317–320 vol.1, DOI: 10.1109/ICME.2002.1035782.

[150] H.D. Kim, H.J. Chung, B.H. Berkeley, and S.S. Kim, « Emerging technologies for the commercialization of AMOLED TVs », *in*: *Information Display* 25.*9* (2009), pp. 18–22.

[151] A. Yeganeh-Khaksar, M. Ansari, S. Safari, S. Yari-Karin, and A. Ejlali, « Ring-DVFS: Reliability-Aware Reinforcement Learning-Based DVFS for Real-Time Embedded Systems », *in*: *IEEE Embedded Systems Letters* 13.*3* (2021), pp. 146–149, DOI: 10.1109/LES.2020.3033187.

[152] D. Novo, A. Nocua, F. Bruguier, A. Gamatie, and G. Sassatelli, « Evaluation of heterogeneous multicore cluster architectures designed for mobile computing », *in*: *2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, IEEE, 2018, pp. 1–8.

[153] A. Butko, F. Bruguier, D. Novo, A. Gamatié, and G. Sassatelli, « Exploration of performance and energy trade-offs for heterogeneous multicore architectures », *in*: *arXiv preprint arXiv:1902.02343* (2019).

[154] A. Butko, F. Bruguier, A. Gamatié, and G. Sassatelli, « Efficient programming for multicore processor heterogeneity: Openmp versus ompss », *in*: *OpenSuCo*, 2017.

Titre : Performance et efficacité énergétique du décodage des standards vidéo de nouvelle génération sur les architectures multi-cœurs basse consommation

Mot clés : Décodage vidéo, matériel, logiciel, consommation de puissance/énergie, architecture mobile multi-cœurs hétérogène, ARM big.LITTLE, classification

Résumé : Les exigences des applications vidéo en ressources de calcul ne cessent d'évoluer. Cette évolution a induit une grande pression sur l'autonomie des batteries des plateformes mobiles. De ce fait, des composantes matérielles dédiées sont intégrées au sein des systèmes-sur-puce (SoC) afin d'optimiser la consommation énergétique du décodage vidéo matériel (HW). Cependant, d'une part, ces décodeurs (HW) ne sont pas flexibles et prennent du temps pour être commercialisés. D'autre part, les processeurs généralistes (GPPs) multi-cœurs hétérogènes sont suffisamment puissants pour pouvoir faire le décodage vidéo logiciel (SW) en temps réel et sont flexibles. En revanche, ils consomment une grande quantité d'énergie. Dans ce contexte, comment optimiser le décodage SW sur les GPPs afin de se rapprocher le plus possible de l'efficacité énergétique du décodage HW ?

Le projet de thèse vient pour répondre à cette question en proposant des mécanismes et des stratégies d'optimisation de la consommation énergétique du décodage vidéo sur les plateformes mobiles hétérogènes. La thèse s'intègre dans le cadre du projet FUI-23 EFIGI.

La première phase de la thèse est la caractérisation. Elle consiste à investiguer, par la mesure, la consommation de puissance du décodage HEVC mis en œuvre en matériel (HW) et en logiciel (SW). Pour cela, nous avons proposé une méthodologie qui intervient à deux niveaux : (i) système d'exploitation (OS), et (ii) applicatif.

Au niveau OS, il s'agit de comprendre le comportement des deux types de décodage : HW et SW. Pour ce qui concerne le décodage HW, nous avons étudié non seulement le traitement sur le décodeur matériel, mais aussi sa communication avec le GPP. Concernant le décodage SW, nous avons étudié l'impact de la variation du nombre de cœurs GPP impliqués dans le décodage selon les fréquences de fonctionnement sur la performance et la consommation énergétique.

Au niveau applicatif, il s'agit de comprendre l'impact de certains paramètres, liés à la vidéo, sur la consommation énergétique du décodage vidéo. Ces paramètres sont : bitrate, frame rate, et résolution. Ensuite, il s'agit de comparer les deux types de décodage en faisant varier les paramètres susmentionnés.

La deuxième phase de la thèse est l'optimisation. Il s'agit de réduire la consommation énergétique du décodage SW en ayant pour objectif de se rapprocher le plus possible de l'efficacité énergétique du décodage HW. La solution proposée peut être décomposée en trois phases : (1) modélisation de la complexité de trames, (2) classification, et (3) mise à jour de la fréquence du GPP.

La phase de modélisation consiste à créer un modèle qui permet de décider sur quels cœurs une trame va être décodée, en connaissant sa taille et son type. Ensuite, la deuxième phase (classification) consiste à appliquer le modèle établi en phase 1. Il s'agit de diriger une trame, en fonction de sa taille et de son type, vers les cœurs performants ou ceux moins performants. Les différents blocs de cette trame sont décodés en parallèle entre les cœurs sélectionnés. Finalement, la troisième phase (mise à jour de la fréquence du GPP) consiste à ajuster la fréquence des cœurs sélectionnés en phase 2 en utilisant une technique du «feedback control». Pour cela, le contrôleur surveille la taille du buffer de sortie et essaye de la tenir à une valeur définie par l'utilisateur. C'est donc une solution en «Asservissement».

Title: Performance and energy efficiency of new generation video decoding standards on low power consumption-based multi-cores architectures

Keywords: Video decoding, hardware, software, power/energy consumption, heterogeneous multi-cores mobile architecture, ARM big.LITTLE, classification

Abstract:

The computational demands of video applications are constantly evolving. This evolution has put a great pressure on the battery life-time of mobile platforms. As a result, dedicated hardware components are integrated into system-on-chip (SoC) devices to optimize the energy consumption of hardware video decoding. However, on the one hand, these decoders (HW) are not flexible and take a long time-to-market. On the other hand, heterogeneous multi-core general purpose processors (GPPs) are powerful enough to do software video decoding (SW) in real time and are flexible. However, they consume a large amount of power. In this context, how to optimize the SW video decoding on GPPs to get as close as possible to the energy efficiency of the HW one?

The thesis project consists in proposing mechanisms and strategies to optimize the energy consumption of video decoding on heterogeneous mobile platforms. The thesis is part of the FUI-23 EFIGI project.

The first phase of the thesis is the characterization. It consists in investigating, by measurement, the power consumption of the decoding implemented in hardware (HW) and in software (SW). For this purpose, we proposed a methodology that intervenes at two levels: (i) operating system (OS), and (ii) application.

At the level of the operating system, the aim is to understand the behavior of the two types of decoding: HW and SW. Concerning the HW decoding, we studied not only the processing on the hardware decoder, but also its communication with the GPP. Concerning the decoding, we studied the impact of the variation of the number of cores involved in the decoding according to the operating frequencies on the performance and the energy consumption.

At the application level, the aim is to understand the impact of certain parameters, related to video, on the energy consumption of video decoding. These parameters are: bitrate, frame rate, and resolution. Then, we compare the two types of decoding by varying the above-mentioned parameters.

The second phase of the thesis is the optimization. The goal is to reduce the energy consumption of the decoding process by getting as close as possible to the energy efficiency of the decoding process. The proposed solution can be decomposed into three phases: (1) frame complexity modeling, (2) classification, and (3) GPP frequency scaling.

The modeling phase consists in creating a model that allows to decide on which cores a frame will be decoded, knowing its size and type. Then, the second phase (classification) consists in applying the model established in phase 1. It consists in directing a frame, according to its size and type, to the best performing cores or to the least performing ones. The different blocks of this frame are decoded in parallel between the selected cores. Finally, the third phase (updating the frequency of the GPP) consists in adjusting the frequency of the cores selected in phase 2 using a «feedback control» technique. To do so, the controller monitors the size of the output buffer and tries to keep it at a value defined by the user.