HEVC hardware vs software decoding: An objective energy consumption analysis and comparison

Mohammed Bey Ahmed Khernache, Yahia Benmoussa, Jalil Boukhobza, Daniel Menard

HEVC hardware vs software decoding: an objective energy consumption analysis and comparison

Mohammed Bey Ahmed Khernache*, Yahia Benmoussa†, Jalil Boukhobza‡, and Daniel Menard§
*Univ. Bretagne-Sud, UMR 6285, Lab-STICC, Lorient, France
mohammed.bey-ahmed-khernache@univ-ubs.fr
†Univ. M'hamed Bougara, LMSS, Boumerdes, Algeria
yahia.benmoussa@gmail.com
‡Lab-STICC UMR CNRS 6285, ENSTA Bretagne, Brest, France
jalil.boukhobza@ensta-bretagne.fr
§INSA de Rennes, UMR CNRS 6164 IETR Image Group, Rennes, France
daniel.menard@insa-rennes.fr

***Abstract*—Web data are experiencing a proliferation of video content for mobile platforms. This is accompanied by new advances in heterogeneous general purpose processor (GPP) cores embedded in mobile devices which offer a great opportunity to enhance both performance and energy efficiency of software (SW) video decoding. On the other hand, hardware (HW) video accelerators are more energy-efficient but are not flexible and their time-to-market is significant. In this context, this paper proposes a characterization methodology to investigate the performance and power consumption of two video decoding approaches on mobile platforms. The first one uses a HW decoder intellectual property (HDIP) in addition to a GPP (for the control). The second one is SW-based and uses only a heterogeneous multi-core GPP. The objective is to study the behavior of both video decoding approaches by comparing them and to understand why and in which case it is worth relying on the GPP rather than the HDIP. We also derive the optimal GPP configuration (number of cores and their frequency) that minimizes the energy consumption for a given video bit-stream on a given platform. The proposed methodology was applied on the HEVC video codec standard. In some state-of-the-art work figures, the SW video decoding consumes up to $1000\times$ more energy than HDIPs. Our results show that, for video resolutions of 1080p and lower and at the operating system perspective point of view, the HEVC SW decoding consumes on average less than $4\times$ more energy (mJ/Frame) than the HW one. Then, the more we scale up the resolution, the more we get the advantage of using the HW video decoding. Furthermore, the HEVC HW and SW decoders consume effectively less than 30% and 50% of the global power consumption of the tested platforms, respectively.**

***Index Terms*—power consumption, mobile platform, HEVC, software video decoding, hardware video decoding, heterogeneous architecture**

I. Introduction

By 2022, online videos will make up more than 82% of all consumer internet traffic [1]. Smartphones, tablets, and media players are the major consumers of this multimedia content. 75% of all video plays are on mobile devices [2] of which 80% are equipped with full high definition (1080p) or lower screen resolutions [3].

In response to the growing need for video applications, such as Internet streaming, videoconferencing, digital storage media, and television broadcasting, Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) jointly developed the High Efficiency Video Coding (HEVC) codec standard in 2013. HEVC was released as the successor to the predominant recommendation ITU-T Advanced Video Coding (AVC/H.264) [4]. The main purpose of HEVC is to drop the bandwidth and storage requirements by roughly 50% with respect to its predecessor. HEVC presents high performance and efficient coding techniques that are capable of reducing the bit-rate for ultrahigh-definition (2160p) resolution videos [5].

These new trends in video application usage combined with the market explosion of multimedia consumer electronics raised new challenges for mobile device architecture designers. Indeed, to fit the important processing requirements of video applications, such as real-time constraint, processing resources embedded in these devices 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. For instance, in [6], measurements show that the smartphone's battery can entirely be consumed when decoding a 1080p video sequence in real-time. The video decoding process is run in GPP by applying the state-of-the-art HEVC codec for $\sim$4h. The measurements do not consider peripherals energy such as that of the display system. This leads to a forceful decrease in mobile devices autonomy as the increased power demand could not be compensated by the improvements in battery technology [7].

For those reasons, energy efficiency has become one of the most important factors in modern microprocessor design, especially for video decoding. One proposed solution is the use of HW video decoding run by a HW decoder intellectual property (HDIP), such as the HEVC decoder [8] [9]. In state-of-the-art work, it is widely accepted that the dedicated processors outperform the general purpose processors (GPPs) by around $1000\times$ in terms of energy efficiency [10]. This is achieved thanks to the use of specialized processing units [11] which eliminate the power consumption related to instruction decoding and control logic characterizing GPP [12]. As a consequence, most modern smartphones are equipped with an

HDIP that consumes less energy with respect to the real-time video decoding constraint [13] [14]. However, HDIPs are not flexible and are costly to implement, which generate a long time-to-market for new video codecs [15].

New advances in multi-core GPPs embedded in mobile devices offer a great opportunity to enhance both the 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 [16] [17]. In case of HEVC decoding, this is enabled by virtue of advanced parallelism schemes supported by this codec [18]. Furthermore, GPPs allow developing and rapidly deploying a broad range of applications, in particular new video codecs.

With the complex and costly design and integration of HDIPs into mobile platforms and the advances in energy efficiency and flexibility of GPPs, we are willing to investigate to what extent do HW video decoders outperform SW ones. We also question about their relevance with regards to the parameters impacting the video decoding process.

Numerous research studies have investigated the HEVC codec. Some of them studied its complexity without investigating the impact of the architecture on which the video decoding process is run [19]. Some others focused on either HW video decoding [20] or SW video decoding [21]. To the best of our knowledge, none of these studies tried to shed some light on the behavior of both video decoding approaches by comparing them and to understand why and in which case it is worth relying on the GPP rather than the HDIP.

In this paper, we propose a methodology that allows to characterize and evaluate the performance and energy efficiency of the two aforementioned approaches of HEVC decoding. This methodology intervenes at both operating system and application levels. Hence, for both HW and SW video decoding, we identify the relevant operating system and application parameters that impact the energy consumption properties. Then, we evaluate and compare different video decoding configurations. In addition, our methodology helps to infer the optimal configuration (number of GPP cores, clock frequency) that gives the best trade-off between performance and energy consumption to decode a video sequence on a given platform.

The obtained experimental results revealed that, on the tested platforms, from the operating system level point of view, and for 1080p video resolution and lower, SW video decoding consumes on average less than $4\times$ more power as compared to HW video decoding. This gap is not as large as expected, according to the state of the art [10], thanks to SW parallel processing. This is not true for high video data rate (bitrate, frame rate, and resolution) where the HW video decoding offers the best energy efficiency (e.g., a ratio of more than $10\times$ for video resolution). Thus, the ratio between the SW and HW HEVC decoding energy as a function of video resolution is not linear.

The remainder of this paper is organized as follows: Section II introduces the necessary background on video decoding energy consumption. Our HEVC decoding performance and power consumption characterization methodology is described in Section III. Section IV discusses the experimental evaluation results. Section V reviews studies related to the energy efficiency of HW and SW video decoding. Finally, conclusions are given in Section VI.

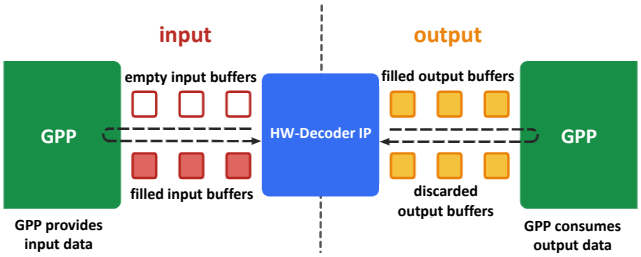


Fig. 1: HW video decoding diagram [22]

II. Background

This section provides the necessary knowledge to help understand our contribution. The following topics are mainly related to energy consumption and parallel video decoding.

For better readability of the paper, all the notations used are summarized in Table VIII.

A. HW video decoding

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:

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

where $video\_frame\_rate$ is the frame rate of the decoded video sequence. Then, at the beginning of each period, as shown in Fig.1, 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 [22].

Furthermore, HDIPs are massively parallel and depend, for their performance, on the fact that the decoders output video data in 2D arrays[1]. 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

[1]2D array: a structure organized as lines (frame height) and columns (frame width).

extreme multi-threading HW or specific data handling and memory access optimization HW [23]. 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 inter processor communication (IPC) may generate some overhead [24] [25]. The IPC also includes all other elements involved in 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.

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

B. SW parallel processing

In this section, we describe how the parallelism is exploited by GPPs, at both architectural and operating system levels, to improve their energy efficiency. It should be noted that GPPs are also designed with other parallelism techniques, such as pipeline and single instruction and multiple data (SIMD) instruction sets. These techniques are used but not included in the study scope of this paper.

To understand how architectural strategies can provide high processing performance at low power levels, it is necessary to look at the complementary metal oxide semiconductor (CMOS) circuit dynamic power consumption equation given by:

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

where C_{eff} represents the circuit effective capacitance and V the supply voltage associated to the clock frequency f [16].

Reducing the supply voltage increases the transistor propagation time and thus requires lowering the processing frequency to keep the circuit in a functional stage.

Assuming that the processing time doubles at the frequency $\frac{f}{2}$, let E_{v_1} and E_{v_2} be the amounts of energy consumed by a circuit at the frequencies $\frac{f}{2}$ and f, respectively. The ratio between these two amounts is:

$$\frac{E_{v_1}}{E_{v_2}} = \frac{C_{\text{eff}}.V_1^2.\frac{f}{2}.2.t}{C_{\text{eff}}.V_2^2.f.t} = (\frac{V_1}{V_2})^2 \leqslant 1$$

where V_1 and V_2 are the necessary voltages supplied at the frequencies $\frac{f}{2}$ and f, respectively. The energy saving is equal to $(100 - (\frac{V_1}{V_2})^2 * 100)\%$.

Therefore, by reducing the power supply voltage to the lowest level that provides the required performance, we can significantly reduce energy consumption. This way, dynamic voltage-frequency scaling (DVFS) is designed to optimize dynamic energy consumption by scaling down the frequency when the operating system observes less load to handle by the GPP.

In order not to decrease the performance when scaling down the frequency, one may replicate HW, e.g., clocked at $\frac{f}{2}$, at the architecture level. This allows to save energy, but at the expense of an additional circuit area. This type of parallelism has shown good performance and energy efficiency for multimedia applications [26]. This generates an additional energy overhead due to the duplication of the processing circuits and the use of multiplexing/demultiplexing circuits in the architecture [27].

III. Methodology & setup

This section describes our proposed video decoding performance and power consumption characterization methodology. Then, the HW and SW experimental setups as well as datasets on which the proposed methodology is applied are presented.

A. Overall power model

In our experiments, we measured power to study and compare the HW and SW video decoding energy consumption. The power consumption is formulated in Equation (3):

$$P_{\text{glob\_dyn}} = P_{\text{glob\_idle}} + \hat{P}_{\text{glob\_active}} \tag{3}$$

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. Each power consumption (idle or active) is composed of two power components as shown in the following equations[2]:

$$P_{\text{glob\_idle}} = \hat{P}_{\text{GPP\_idle}} + \hat{P}_{\text{Other\_idle}} \tag{4}$$

$$\hat{P}_{\text{glob\_active}} = \hat{P}_{\text{GPP\_active}} + \hat{P}_{\text{Other\_active}} \tag{5}$$

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

Regarding the dynamic power consumption of the GPP, $\hat{P}_{\text{GPP\_dyn}}$, it is the sum of the active and idle powers. It is formulated as a function of the number of cores that are in idle and active states, as given by Equation (6).

$$\begin{aligned}\hat{P}_{\text{GPP\_dyn}} = {} & nb_{\text{active}} * (\hat{P}_{\text{GPP\_1\_core\_idle}} + \hat{P}_{\text{GPP\_1\_core\_active}}) \\ & + nb_{\text{idle}} * \hat{P}_{\text{GPP\_1\_core\_idle}}\end{aligned} \tag{6}$$

where nb_{idle} and nb_{active} are the number of cores which are in idle and active states, respectively, whereas $\hat{P}_{\text{GPP\_1\_core\_idle}}$ and $\hat{P}_{\text{GPP\_1\_core\_active}}$ are the power consumption of one GPP core in idle and active states, respectively[3].

Then, the power consumption of one GPP core in idle state, $\hat{P}_{\text{GPP\_1\_core\_idle}}$, is given by Equation (7). It is deduced by subtracting the global power consumption when turning on one GPP core from that when turning on two GPP cores, both in idle state.

$$\hat{P}_{\text{GPP\_1\_core\_idle}} \approx P_{\text{glob\_idle\_2\_cores}} - P_{\text{glob\_idle\_1\_core}} \tag{7}$$

where $P_{\text{glob\_idle\_2\_cores}}$ and $P_{\text{glob\_idle\_1\_core}}$ are the global power consumption of the platform in idle state when only two

[2]Throughout the paper, a symbol with *circumflex* indicates a calculated power, i.e., deduced from other measured powers.

[3]In case of a heterogeneous architecture, such as ARM big.LITTLE, nb_{idle}, nb_{active}, $\hat{P}_{\text{GPP\_1\_core\_idle}}$, and $\hat{P}_{\text{GPP\_1\_core\_active}}$ are, in turn, split into components related to big and LITTLE cores, respectively.

GPP cores and one GPP core are turned on, respectively. We suppose that $\hat{P}_{\text{Other\_idle\_1\_core}}$ and $\hat{P}_{\text{Other\_idle\_2\_cores}}$ are equivalent and thus the difference is rounded to zero.

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 the 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 is negligible.

$\hat{P}_{\text{Other\_idle}}$ ratio, $\hat{r}_{p\_other\_idle}$, is given by Equation (8). It is the proportion of the power consumption of all but GPP with respect to the global dynamic power, $P_{\text{glob\_dyn}}$.

$$\hat{r}_{p\_other\_idle} = \frac{\hat{P}_{\text{Other\_idle}}}{P_{\text{glob\_dyn}}} * 100 \tag{8}$$

Next, the effective amount of energy consumed for the video decoding process, $\hat{E}_{\text{glob\_active}}$, is given by Equation (10). To evaluate it, we first calculate the video decoding power consumption, $\hat{P}_{\text{glob\_active}}$.

$$\hat{P}_{\text{glob\_active}} = P_{\text{glob\_dyn}} - P_{\text{glob\_idle}} \tag{9}$$

$$\hat{E}_{\text{glob\_active}} = \sum(\hat{P}_{\text{glob\_active}} * \Delta t) \tag{10}$$

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, $\hat{r}_{\text{sw/hw}}$, is calculated using the following equation:

$$\hat{r}_{\text{sw/hw}} = \frac{\hat{E}_{\text{glob\_active\_sw}}}{\hat{E}_{\text{glob\_active\_hw}}} * 100 \tag{11}$$

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

B. 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. It is worth mentioning that the video decoding performance and power consumption characterization is done independently from the application design and its intricacies[4]. Also, the decoded frames are not displayed[5].

Fig.2 depicts an overview of our methodology. The inputs are the parameters for which the impact is estimated and the output is the consumed power measured. Our objective is to study and compare, by measurement, the energy consumption of two video decoding approaches: HW and SW[6]. For that purpose, we vary and study the impact of several parameters triggered at two hierarchical levels: (i) operating system, and (ii) application.

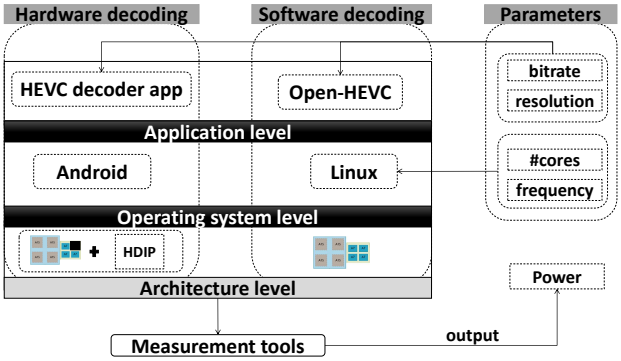


Fig. 2: The proposed characterization methodology overview

We note that the architecture is the same throughout the experiments: an HDIP in addition to a GPP (referred to as GPP–HDIP, in this paper), for the HW video decoding, and a heterogeneous GPP for the SW one.

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 operating system 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 power consumption of the video decoding at the application level. The measured video decoding power is given by Equation (3). The study at this level can be divided into the three following steps.

a) Idle state: The objective of this study is to assess the power when the two video decoding platforms are in idle state. This step is mandatory since the two video decoding approaches are run on two different processor types.

In case of HW video decoding platform, only one GPP core is used. On this platform, we show how much the GPP one core idle power, $\hat{P}_{\text{GPP\_1\_core\_idle}}$ of Equation (7), represents against that of the global platform, $P_{\text{glob\_idle}}$ of Equation (4), by comparing them. The GPP core frequency is fixed and clocked at the frequency that allows it to launch the target application (video decoding application)[7]. 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

[4]This means that the study we conducted does not take into account the details related to HEVC decoding kernels (ex: Entropy decoding, Motion compensation, etc). We consider the HEVC decoding process as a black box. 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 kernels.

[5]The display system study is beyond the scope of this paper.

[6]Throughout the paper, the two video decoding approaches refer to: HW and SW video decoding.

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

power, $\hat{P}_{\text{GPP\_idle}}$ of Equation (6), represents against that of the global platform, $P_{\text{glob\_idle}}$ of Equation (4), 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 minimum and then its maximum frequency.

The measurements done in idle state are used to deduce the effective video decoding energy consumption, according to Equation (10).

b) 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[8] and when they are performing the video decoding process. Here, $P_{\text{glob\_dyn}}$ is evaluated and compared to $P_{\text{glob\_idle}}$, according to Equation (3). For that, a set of video sequences are decoded to analyze the HW video decoding power consumption by zooming in one second of decoding time[9]. This time duration is chosen to evaluate the video bitrate and frame rate which are quality of service (QoS) metrics in video decoding applications. 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 (11), by comparing them. Third, we investigate the $\hat{P}_{\text{Other}}$ ratio, using Equation (8), which includes the IPC of the video decoding process.

c) 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, we investigate $\hat{E}_{\text{glob\_active}}$ which is calculated using Equation (10). 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 maximum value supported by every core is reached. Beyond this frequency, the high-performance core frequencies continue to be scaled until their maximum supported value, whereas the remaining cores are clocked at their maximum 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 (6). For instance, in a big.LITTLE SoC of eight GPP cores ($4\times$big + $4\times$LITTLE), using five active cores gives four different combinations that can be evaluated, see Table I.

[8]doze mode: idle state concept used in Android OS

[9]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 I
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

Next, for each combination of the number of cores, only the one which consumes the minimum energy (mJ/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.

After that, the $\hat{P}_{\text{Other}}$ ratio is evaluated using Equation (8). $\hat{P}_{\text{Other}}$ 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 (11).

2) Application level: The objective of this level is to explore how the video parameters (bitrate, frame rate, and resolution) affect the video decoding energy consumption. For that, a set of video sequences encoded at different bitrates, frame rates, and resolutions, from 720p to 2160p through 1080p and 1600p, are decoded. Then, the overall energy consumption, $\hat{E}_{\text{glob\_active}}$, is calculated using Equation (10). Finally, this value is divided by the video sequence number of frames to obtain the average energy/frame. After that, 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).

C. Experimental setup

1) HW setup: In our work, we characterized the HEVC decoding performance and power consumption on three mobile platforms: Snapdragon 810 development board (APQ 8094) [28], Odroid-xu3 [29], and Qualcomm Robotics RB3 [30]. 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. We describe below the HW and SW setups used in our experiments.

a) Snapdragon 810 development board: 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 ($4\times$ Cortex-A57 high performance cores and $4\times$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 HW decoder. Android debug bridge (adb) tool was used to communicate with this platform via a USB connection.

b) Odroid-xu3: has a smart-phone-like 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 ($4\times$ Cortex-A15 high performance out-of-order cores and $4\times$ 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.

c) Qualcomm Robotics RB3 development 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: $4\times$ high performance cores and $4\times$ 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.

2) Power consumption evaluation: In our experiments, N6705A DC Power Analyzer is used to measure power. The power measured by this tool is given by Equation (3). 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.

In the case where the platform has on-board sensors, such as the SW video decoding platform (Odroid-xu3), Open-PEOPLE [31] is used as it measures $P_{\text{GPP\_idle}}$ and $P_{\text{GPP\_active}}$, from Equation (3), separately from the rest of the components power[10].

Then, to measure the power consumption of the video decoding process, we first start the power consumption data-logging, followed by putting 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:

$$T_{\text{M}} = T_{\text{idle\_1}} + (T_{\text{L}} + T_{\text{D}}) + T_{\text{idle\_2}} \tag{12}$$

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 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 software, 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.

[10] $P_{\text{GPP\_idle}}$ 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 II
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

3) SW setup: As operating system (OS), Android 6.0 (Marshmallow, under Linux kernel 3.10.84) for Snapdragon 810, as the only supported OS on this platform, Ubuntu 16.04 (Linux kernel 4.14.176+) distribution for Odroid-xu3, and Debian-based Linaro Linux 10.3 (Linux kernel 5.4.0) for RB3 were used. The kernel of Odroid-xu3 OS 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 governor was also enabled on the Snapdragon 810 platform.

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 software [32] is compiled on Odroid-xu3 and RB3 with NEON optimizations enabled. Ffmpeg software [33] was used with v4l2 library to leverage the HEVC HDIP available on the RB3 platform.

The raw video sequences were encoded using ffmpeg software. 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 cache memory was flushed at the beginning of each experiment in order to clean the memory from the temporary data used in previous experiments.

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) [34] 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 [35] dataset and others [36]. The common characteristics of these video sequences are summed up in Table II.

In our work, we studied only the common parallelism scheme for all videos which is frame-by-frame. For other parallelism schemes, Wavefront Parallel Processing (WPP) and Tiling, the decoding process depends on the coding parameters.

5) Methodology summary: Finally, the proposed video decoding performance and power consumption characterization methodology is summarized in Table III.

TABLE III
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	
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
Measured powers	$P_{\text{glob\_idle}}$ and $P_{\text{glob\_dyn}}$ of Equation (3)	
Calculated powers/energies	$\hat{P}_{\text{glob\_active}}$ of Equation (3) $\hat{P}_{\text{GPP\_dyn}}$ of Equation (6) $\hat{P}_{\text{GPP\_1\_core\_idle}}$ of Equation (7) $\hat{E}_{\text{glob\_active}}$ of Equation (10)	
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 II	

IV. Results & analysis

In this section, we describe and analyze the obtained results of the HEVC decoding performance and power consumption characterization. According to our proposed methodology, the results are presented in two sections, operating system level and then application level.

A. Operating system level

1) Idle state: The first step of this level aims at studying the idle power consumption on two tested platforms: (i) the HW video decoding platform (Snapdragon 810), and (ii) the SW video decoding platform (Odroid-xu3). For the former, we show the GPP–HDIP power consumption and then compare it to that of the global platform. Regarding the latter, we show the GPP power consumption and then compare it to that of the global platform. In addition, the GPP is clocked at its minimum and then its maximum frequency. Finally, we compare the idle power of the two platforms. These results are shown in Fig.3.

a) 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 Equation (3). Periodically, the system exits doze mode for a short time to let the applications complete their deferred activities [37]. This is illustrated by the regular peaks in both HW video decoding platform doze power consumption curves in Fig.3.

Fig.3 additionally plots the 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). Compared to the global platform doze power, one can observe that the $\hat{P}_{\text{Other\_idle}}$ component, which is the difference between $P_{\text{glob\_idle}}$ and $\hat{P}_{\text{GPP\_1\_core\_idle}}$ in Fig.3, constitutes a huge part of the global power consumption. Indeed, it is about 88%.

b) SW video decoding platform idle power consumption: On Linux-like systems (on Odroid-xu3 platform in our case), 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

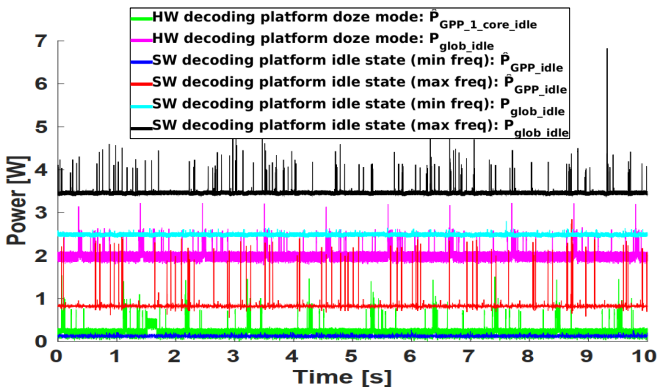


Fig. 3: HW video decoding platform (Snapdragon 810) versus SW video decoding platform (Odroid-xu3) idle power

state, the SW video decoding platform consumes $P_{\text{glob\_idle}}$, according to Equation (3). This results in on average 2.49 Watts and 3.46 Watts for the minimum and maximum GPP clock frequency, respectively. It should be noted that the WFI power consumption is different from the GPP static power [38].

Fig.3 additionally plots the 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 minimum and maximum GPP clock frequency, respectively, according to Equation (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.3, constitutes a huge part of the global power consumption. Actually, it is about 94% and 76% for the minimum and maximum GPP clock frequency, respectively.

As explained in Section II-B, idling at low clock frequency allows the processor to consume significantly less power. For instance, on the SW video decoding platform (Odroid-xu3), clocking the GPP cores at their minimum frequency would allow to gain, at the operating system level, approximately 84% of power compared to the state where the cores are clocked at their maximum frequency, see Fig.3. 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), whereas $P_{\text{glob\_idle}}$ represents on average 2.01 Watts, according to Equation (3). Therefore, $\hat{P}_{\text{Other\_idle}}$ constitutes on average 88% of the global platform power.

On the SW video decoding platform (Odroid-xu3), according to Equation (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 minimum and maximum GPP clock frequency, respectively. Thus, we can save around 84% of power by scaling down the GPP frequency from maximum to minimum. On the other hand, $\hat{P}_{\text{Other\_idle}}$ constitutes about 90% and 85% of the global platform power for the minimum and maximum GPP clock frequency, respectively.

2) HW video decoding: We describe here the HEVC HW decoding power consumption using the GPP–HDIP on the Snapdragon 810 platform. We investigate $P_{\text{glob\_dyn}}$ and $P_{\text{glob\_idle}}$ values of Equation (3). We then assess 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 (8). For that, *Kimono* video test sequence, which possesses the characteristics given in Table II (resolution is 1080p), is decoded as an example. The video decoding process is done in real-time. Other tested video sequences showed similar behaviors.

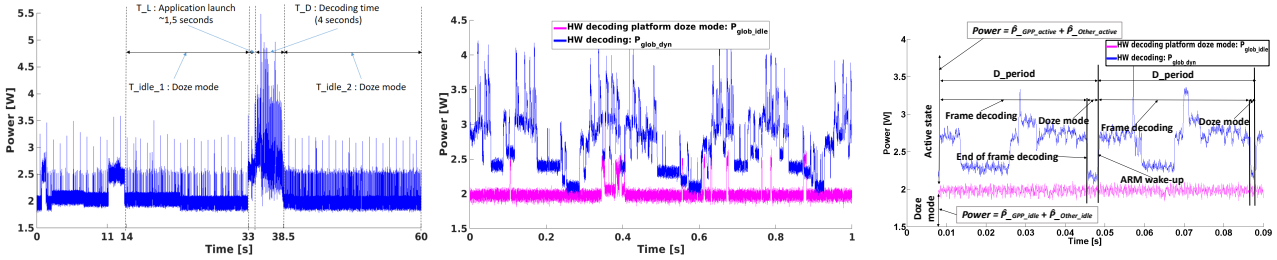
Fig.4a shows the complete data-logging of the HW video decoding power consumption. It includes the different parts of Equation (12). We noted that the launching time is not negligible in case of the Android platform.

Fig.4b depicts the HW video decoding power consumption, $P_{\text{glob\_dyn}}$, compared to doze power, $P_{\text{glob\_idle}}$, using Equation (3), during one second. Given that the video example (Kimono) duration is 4 seconds, the position of the snapshot shown in Fig.4b is from 1 to 2 seconds of the video. Fig.4c 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 example, on average 2.75 Watts while the doze power ($P_{\text{glob\_idle}}$) represents on average 2.01 Watts.

In Fig.4b, one can observe that the HDIP enters doze mode whenever it finishes decoding before the end of the video decoding period, see Section II-A, 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.4c, we can observe two doze intervals of 7.5% and 2.5% of the video decoding period, respectively. 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.4c 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, 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 [39]. Therefore, in case of HW video decoding, $\hat{P}_{\text{HDIP}}$ is never equal to zero, according to Equation (3).

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



(a) HW video decoding power consumption complete data-logging

(b) Zoom on 1 second of HW video decoding versus doze mode power consumption

(c) Zoom on 2 frames HW video decoding versus doze mode power consumption

Fig. 4: HEVC HW decoding power consumption (platform: Snapdragon 810, video sequence: Kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

elements involved in the video decoding process. The overhead leads to extra performance and energy costs. Indeed, according to Equation (8), $\hat{P}_{\text{Other}}$ 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 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.

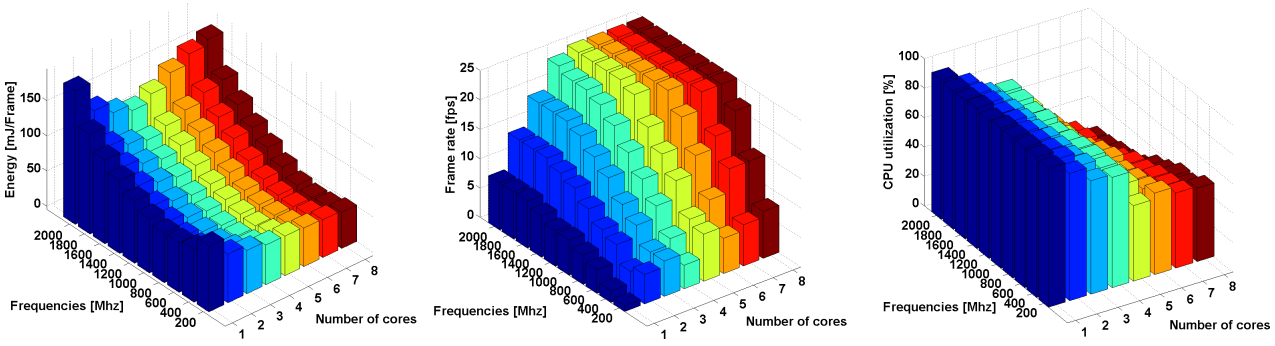
3) SW video decoding: In this step, we analyze the HEVC SW decoding energy consumption using a heterogeneous GPP on the Odroid-xu3 platform. We investigate $\hat{E}_{\text{glob\_active}}$ given by Equation (10) by deducing $P_{\text{GPP\_active}}$, see Footnote (10), from Equation (3). We vary the number of cores and their operating frequencies. We then assess 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 using Equation (8). Finally, we compare the SW video decoding energy consumption to that of the HW video decoding by calculating the ratio between them, $\hat{r}_{\text{sw/hw}}$, using Equation (11). For that, *Kimono* video test sequence, which possesses the characteristics given in Table II (resolution is 1080p), is decoded as an example. The decoding process is done in real-time, so, the fps is 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.5a, Fig.5b, and Fig.5c 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 (10) and is given in mJ/Frame. At first sight, one may notice that the energy does not always grow as the number of cores and their clock frequencies increase as explained in this section.

The curve, in Fig.5a, 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.5c. Then, the more we add GPP cores, the less we occupy them and the 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.

In the presented results, following our methodology described in Section III-B1c, 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.5b. We notice 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.

Different five GPP cores configurations (in a heterogeneous big.LITTLE architecture), clocked at 1.2 GHz, give 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 we take off one big core and replace it with one LITTLE core, the energy consumption (mJ/Frame) decreases, see Fig.6. 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 I: using



(a) Energy as a function of the number of active cores and the processing frequencies

(b) Frame rate as a function of the number of active cores and the processing frequencies

(c) GPP utilization as a function of the number of active cores and the processing frequencies

Fig. 5: HEVC SW decoding evaluation (platform: Odroid-xu3, video sequence: kimono, bitrate: 1831 Kbps, frame rate: 25 Hz, resolution: 1080p)

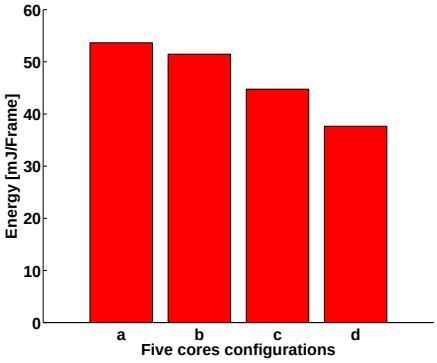


Fig. 6: Five cores configurations energy consumption

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, at the operating system level, 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.

We recall that energy is given by Equation (10). Using one to four GPP cores, the fps increases (so the video decoding time decreases) as we increase the number of active cores, see Fig.5b (one to four cores). On the other hand, as it is stated in Section II-B, 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.5a. There are two things which can 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 more and more every time we add a big core. 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.

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 function. 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, $\hat{P}_{\text{Other\_idle}}$ represents, in this study example, on average 2.33 Watts, deduced using Equation (7), while the SW video decoding global power represents on average 3.91 Watts, according to Equation (3). So, $\hat{P}_{\text{Other}}$ 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.

The objective of the last step 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 IV presents the energy consumption of both HEVC HW decoding, using GPP–HDIP, and HEVC SW decoding, using different GPP configurations (one to five GPP cores) on the big.LITTLE heterogeneous architecture. 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. 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 IV show that varying the number of cores can save the energy consumption considerably. Using one core not only consumes the largest amount of energy (a ratio of more than $5\times$) but also the real-time constraint is not satisfied. Then, when two big cores are decoding, the video is

TABLE IV
HEVC HW (on the Snapdragon 810 platform) versus SW (on the Odroid-xu3 platform) video decoding energy consumption

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

decoded without any missed frame, i.e., in real-time. However, it still drains the battery autonomy. 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 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 less than $2\times$). Beyond five cores, the energy (mJ/Frame) increases as explained earlier in this section.

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 (mJ/Frame) is using five GPP cores at 1.2 GHz. Moreover, decoding at this configuration would save, at the operating system level, 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 [40]. Finally, to do SW video decoding, the GPP consumes effectively on average less than 50% of the SW video decoding global power consumption. This overhead includes the inter processor communication between the GPP and other components involved in video decoding, such as memory.

B. Application level

At this level, we evaluate the influence of the video bitrate, frame rate, and resolution on the HEVC decoding energy consumption executed on mobile platforms. The video decoding consumed energy is calculated using Equation (10) 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 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, and (ii) in HW using the GPP–HDIP (on the Snapdragon 810 platform).

1) Varying video bitrate: Fig.7a 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 (10).

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.

In case of SW video decoding, the energy (mJ/Frame) increases as the video bitrate grows up. By definition, the bitrate is the amount of data required to decode a single 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.7a, decoding the 9283 Kbps video sequence would increase the energy consumption (mJ/Frame) on average by $3.08\times$ compared to the 1434 Kbps video sequence.

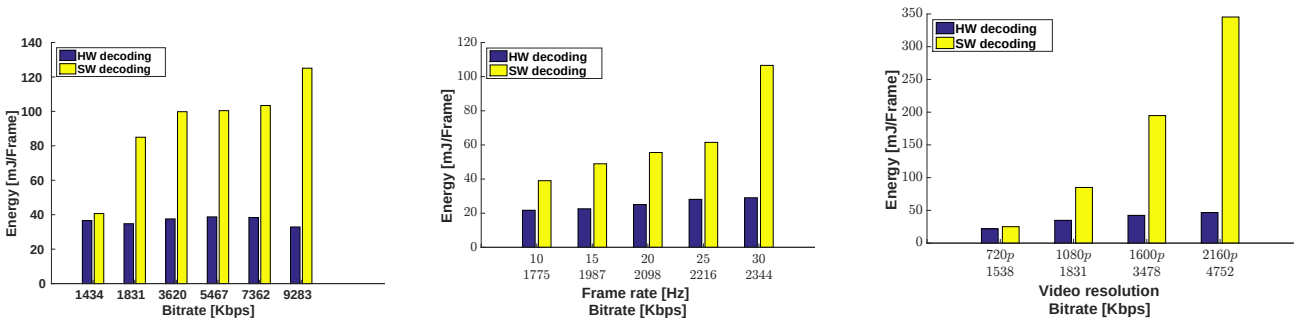
2) Varying video frame rate: Fig.7b depicts the impact of the video frame rate parameter on the HW and SW video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (10).

First of all, when changing the video frame rate, the resulting video bitrate changes accordingly, i.e., the video bitrate increases with the frame rate.

In case of HW video decoding, the energy consumption (mJ/Frame) increases every time we increase the video frame rate. 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 we get. Therefore, more consumed energy (mJ/Frame).

In the presented example, the increase is very slight. This shows that the HDIP is not scalable, i.e., the increase in energy (mJ/Frame) is not remarkable. Indeed, in the study example shown in Fig.7b, the video sequence encoded at 30 Hz would need on average $1.34\times$ more energy (mJ/Frame) to be decoded than another one encoded at 10 Hz. The reason is that the HDIP is designed, i.e., optimized for high data rate (4K content) in terms of energy efficiency (mJ/Frame).

In case of SW video decoding, the energy (mJ/Frame) increases as the video frame rate grows up. This increase can be explained by the high data rate resulting when we increase the video frame rate. Actually, increasing the video frame rate means that the GPP should do more job in the same unit of time, i.e., the number of frames decoded per second. In addition, to do more job, the GPP number of cores and frequencies should be scaled accordingly. Therefore, the GPP consumes more power. For instance, in Fig.7b, decoding the video sequence encoded at 30 Hz would increase the energy



(a) HEVC decoding energy when varying video bitrate (frame rate: 25 Hz, resolution: 1080p)

(b) HEVC decoding energy when varying video frame rate (resolution: 1080p)

(c) HEVC decoding energy when varying video resolution (frame rate: 25 Hz)

Fig. 7: HEVC HW vs SW video decoding energy consumption (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, platforms: Snapdragon 810 and Odroid-xu3)

TABLE V
HW video decoding power consumption percentage, $(1 - \hat{r}_{p\_other})$ using Equation (8)

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

TABLE VI
Energy consumption ratio as a function of video resolution, see Equation (11), on Snapdragon 810 and Odroid-xu3

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

consumption (mJ/Frame) on average by $2.72\times$ compared to that encoded at 10 Hz. The more we increase the frame rate, the more energy (mJ/Frame) is needed for GPP to decode a video sequence.

3) Varying video resolution: Fig.7c 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 (10).

In case of HW video decoding, scaling the video resolution from 720p to 2160p through 1080p and 1600p would consume only $1.43\times$, $1.94\times$ and $2.14\times$ more energy (mJ/Frame), respectively. This is because the HDIP is designed for 2160p (a.k.a. 4K) video content. In addition, this shows that, as we concluded, the HDIP did not 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 23% of the global power consumption for 1080p resolution and lower, it is getting more significant when increasing the video resolution. Table V 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 more the video resolution the more we get the advantage of the HDIP power efficiency. As the HDIP 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 $1.64\times$ more energy (mJ/Frame) than a 720p video sequence. Above this resolution, the ratio increases drastically: $7.76\times$ and $13.76\times$ for 1600p and 2160p resolutions, respectively.

Table VI depicts the ratio between the HEVC SW and HW video decoding energy, calculated using Equation (11), as a function of the video resolution. Compared to the state-of-the-art study where it is widely accepted that the dedicated processors outperform the GPPs by around $1000\times$ in terms of energy efficiency, for low video resolutions (720p and 1080p), from the operating system level point of view, the gap is not as high as that of the state-of-the-art work (a ratio of less than $3\times$). However, for high video resolutions, the GPP–HDIP offer the best performance in terms of energy efficiency (a ratio of more than $7\times$). The more we increase the video resolution, the more we get the advantage of the HW video decoding energy efficiency.

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 $1.43\times$, $1.94\times$ and $2.14\times$

in case of the former and $1.64\times$, $7.76\times$ and $13.76\times$ in case of the latter. Therefore, the more we increase video resolution, the more we get the advantage of the HW video decoding.

Bitrate, frame rate, and resolution 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 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 decoder a high priority to utilize the GPP. It is applicable to all processing resources required to perform the video decoding: memory (DRAM, cache, and HDD), buses, etc, as well.

The video frame rate and resolution parameters are highly correlated with the video bitrate. For instance, when we change the resolution of a given video sequence, 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.7a, Fig.7b, and Fig.7c show similar behaviors.

Finally, we concluded that, 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 we want to model the energy consumption of the video decoding process.

C. Results generalization

In this section, we present the results obtained when performing experiments, at the application level, on the same platform (RB3) for both HW and SW video decoding.

Fig.8a, Fig.8b, and Fig.8c show the video decoding energy consumption (mJ/Frame), $\hat{E}_{\text{glob\_active}}$ of Equation (10), when varying the video bitrate, frame rate, and resolution parameters, respectively. Actually, Fig.8 presents the results of the same measures as Fig.7 but on a different platform (RB3). It plots the results of HW and SW video decoding energy consumption (mJ/Frame) on the same platform (RB3).

Compared to the results obtained in Fig.7, one can note that they show a similar trend of the energy consumption (mJ/Frame). This can also be confirmed by Table VII which presents the results of the same measures as Table VI but on a different platform (RB3). We think that the difference of ratios 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 [41]. Unfortunately, our measurement system does not allow to estimate this power.

TABLE VII
Energy consumption ratio as a function of video resolution, see Equation (11), on RB3

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

V. Related work

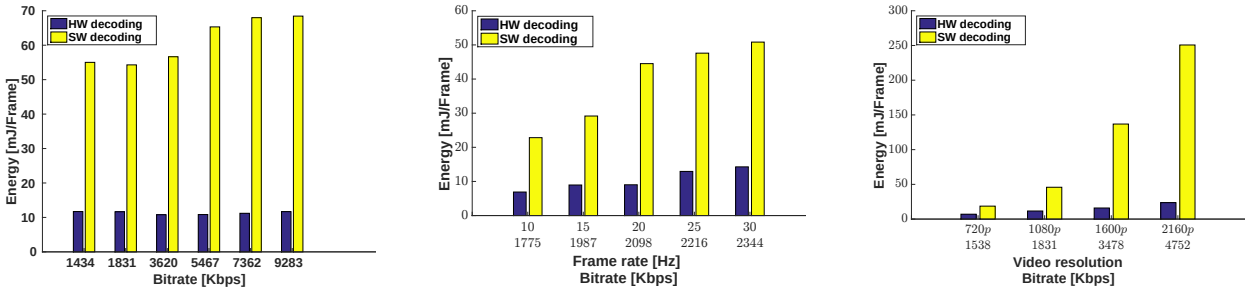
Various studies have been conducted on video decoding power consumption for previous and current codec standards. A large proportion of these studies addressed the earlier video codec standard H.264 (a.k.a. AVC). The authors in [39] gave a comprehensive and comparative study of the performance and energy consumption of AVC decoding applications on embedded heterogeneous platforms containing a GPP and a DSP. In [27], the authors showed how using pipelines and parallelism in CMOS circuits enhances energy efficiency.

The HEVC decoding process has been the interest of multiple studies on different platforms. On a heterogeneous ARM big.LITTLE architecture, Raffin et al. [42] proposed strategies which make profit of data and task-level parallelism as well as a new frequency control system. These strategies aim at optimizing the HEVC SW decoding power consumption. In addition, HEVC has been profiled on both ARM and x86 architectures to study its complexity [19]. It was shown that 480p30 decoding on mobile devices is feasible within reasonable bitrate ranges. Unfortunately, the impact on power consumption was not investigated. Furthermore, in [9], M. Tikekar et al. proposed architectural optimizations for an HEVC HW decoder on an application-specific integrated circuit (ASIC) test chip. These optimizations showed very low power consumption. However, the study did not investigate SW video decoding. In addition, HEVC has also been optimized on DSPs and then compared to the GPP [43]. Nonetheless, on the one hand, the video decoding power consumption was not investigated. On the other hand, the DSP processor is not relevant in our study. In [20], Christian Herglotz and André Kaup investigated the energy an HEVC HW decoder needs for decoding and displaying an HEVC-coded bit-stream. The results are used to establish a model capable of estimating the HEVC HW decoding energy consumption. In [21], the authors studied the complexity of different modules of HEVC and their energy consumption. The authors proposed some techniques for the HEVC decoding process in order to optimize its energy consumption on a mobile device. The authors did not compare the energy efficiency to that of a dedicated HW decoder.

In this context, our study comes to investigate the HEVC decoding performance and power consumption on mobile devices. We analyzed two approaches of video decoding: HW using the GPP–HDIP, and SW using a heterogeneous GPP. 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.

VI. Conclusions & future work

In this paper, we proposed a characterization methodology to evaluate the HEVC decoding performance and energy consumption. We investigated, by measurement, the HEVC



(a) HEVC decoding energy when varying video bitrate (frame rate: 25 Hz, resolution: 1080p)

(b) HEVC decoding energy when varying video frame rate (resolution: 1080p)

(c) HEVC decoding energy when varying video resolution (frame rate: 25 Hz)

Fig. 8: HEVC HW vs SW video decoding energy consumption (Video sequence: kimono, QPs: 21 to 37, PSNRs: 22.97 to 43.8, platform: RB3)

decoding in both implementation approaches: HW, using the GPP–HDIP, and SW, using only a heterogeneous GPP processor. We studied the performance as well as the power/energy consumption of HEVC decoding 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 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, we 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 $4\times$ 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. The more we increase the video data rate, the more we get the advantage of the HW video decoding energy efficiency.

Finally, on the tested platforms, from the operating system level 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.

We aim, in our future work, at proposing an online HEVC decoding scheduling. One may also expect to further reduce the gap between the HW and SW HEVC decoding if more advanced optimizations are used.

ACKNOWLEDGMENT

This work was supported by BPI France, Cap Digital, and Région Ile de France through the French project EFIGI.

REFERENCES

[1] *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).

[2] *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).

[3] *Mobile Screen Resolution Stats Worldwide | StatCounter Global Stats*. URL: https://gs.statcounter.com/screen-resolution-stats/mobile/worldwide (visited on 09/30/2019).

[4] 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.

[5] ITU-T Study Group et al. "Series H: Audiovisual and multimedia systems: Infrastructure of audiovisual services–coding of moving video". In: *ITU-T Study Group,"Series H: Audiovisual and multimedia systems: Infrastructure of audiovisual services–coding of moving video," in General Secretariat and Telecom Radiocommunication (ITU-R) Standardization (ITU-T), sec. H* 264 (2005). revised: 02/2018.

[6] 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.

[7] 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.

[8] 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.

[9] Farouk Amish and El-Bay 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] *MediaCodec | Android Developers*. URL: https://developer.android.com/reference/android/media/MediaCodec (visited on 01/09/2019).

[23] *Development of the VPU | Jon Peddie Research*. URL: https://www.jonpeddie.com/blog/development-of-the-vpu/ (visited on 09/12/2019).

[24] 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 Photon-

ics. SPIE, 2003, pp. 719–726. DOI: 10.1117/12.476330. URL: https://doi.org/10.1117/12.476330.

[25] 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.

[26] 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.

[27] D. Markovic, V. Stojanovic, B. Nikolic, M.A. Horowitz, and R.W. Brodersen. "Methods for true energy-performance optimization". In: *IEEE Journal of Solid-State Circuits* 39.8 (Aug. 2004), pp. 1282–1293. ISSN: 1558-173X. DOI: 10.1109/JSSC.2004.831796.

[28] *APQ8094 | Qualcomm*. URL: https://www.qualcomm.com/products/apq8094 (visited on 05/03/2018).

[29] *Development of the VPU | Jon Peddie Research*. URL: https://www.jonpeddie.com/blog/development-of-the-vpu/ (visited on 12/01/2017).

[30] "Qualcomm Robotics RB3". In: (). URL: https://www.qualcomm.com/media/documents/files/robotics-rb3-platform-product-brief.pdf.

[31] Y. Benmoussa, E. Senn, J. Boukhobza, M. Lanoe, and D. Benazzouz. "Open-PEOPLE, 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.

[32] *GitHub - OpenHEVC/openHEVC at ffmpeg_update*. URL: https://github.com/OpenHEVC/openHEVC/tree/ffmpeg_update (visited on 07/07/2018).

[33] *Download FFmpeg*. URL: https://ffmpeg.org/download.html (visited on 03/01/2018).

[34] F. Bossen. "Common test conditions and software reference configurations". In: *JCTVC-L1100* 12 (2013).

[35] *Jellyfish Bitrate Test Files*. URL: http://jell.yfish.us/ (visited on 03/01/2018).

[36] Xiph.org. *Xiph.org :: Derf's Test Media Collection*. 2015. URL: http://media.xiph.org/video/derf/ (visited on 11/28/2017).

[37] *Optimize for Doze and App Standby*. URL: https://developer.android.com/training/monitoring-device-state/doze-standby (visited on 10/01/2018).

[38] 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/.

[39] 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.

[40] E. Nogues, A. Mercat, F. Arrestier, M. Pelcat, and D. Menard. "Convex Energy Optimization of Streaming Applications for MPSoCs". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019, pp. 1557–1561. DOI: 10.1109/ICASSP.2019.8682317.

[41] Peter 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.

[42] 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.

[43] M. Chavarrías, F. Pescador, M.J. Garrido, E. Juárez, and C. Sanz. "A multicore DSP HEVC decoder using an actorbased dataflow model and OpenMP". In: *IEEE Transactions on Consumer Electronics* 61.2 (May 2015), pp. 236–244. ISSN: 1558-4127. DOI: 10.1109/TCE.2015.7150599.

TABLE VIII
Annex: Notations used in the paper

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	-