Energy-Aware HEVC Software Decoding On Mobile Heterogeneous Multi-Cores Architectures

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

Energy-Aware HEVC Software Decoding On Mobile Heterogeneous Multi-Cores Architectures

Mohammed Bey Ahmed Khernache
Univ. Bretagne-Sud, UMR 6285, Lab-STICC, France

Jalil Boukhobza
Lab-STICC UMR CNRS 6285, ENSTA Bretagne, France

Yahia Benmoussa
Univ. M'hamed Bougara, LMSS, Algeria

Daniel Menard
INSA de Rennes, UMR CNRS 6164 IETR Image Group, France

Abstract

Video content is becoming increasingly omnipresent on mobile platforms thanks to advances in mobile heterogeneous architectures. These platforms typically include limited rechargeable batteries which do not improve as fast as video content. Most state-of-the-art studies proposed solutions based on parallelism to exploit the GPP heterogeneity and DVFS to scale up/down the GPP frequency based on the video workload. However, some studies assume to have information about the workload before to start decoding. Others do not exploit the asymmetry character of recent mobile architectures. To address these two challenges, we propose a solution based on classification and frequency scaling. First, a model to classify frames based on their type and size is built during design-time. Second, this model is applied for each frame to decide which GPP cores will decode it. Third, the frequency of the chosen GPP cores is dynamically adjusted based on the output buffer size. Experiments on real-world mobile platforms show that the proposed solution can save more than 20% of energy (mJ/Frame) compared to the Ondemand Linux governor with less than 5% of miss-rate. Moreover, it needs less than one second of decoding to enter the stable state and the overhead represents less than 1% of the frame decoding time.

2012 ACM Subject Classification Hardware → Platform power issues; Hardware → Chip-level power issues; Computing methodologies → Classification and regression trees; Computer systems organization → Multicore architectures

Keywords and phrases energy consumption, mobile platform, heterogeneous architecture, software video decoding, hardware video decoding, HEVC

Digital Object Identifier 10.4230/OASIcs.PARMA-DITAM.2022.4

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

1 Introduction

Mobile video content will generate nearly four-fifths of mobile data traffic by 2022, according to Cisco [2]. Smartphones, tablets, and media players are the favored and most frequently-used tools to consume this multimedia content. This proliferation can be explained by the omnipresence of mobile devices which made the consumption of these data easier. Also, the Covid-19 crisis has soared the use of video content, e.g., video-conferencing [10].

This context made the energy efficiency one of the most important factors in modern mobile platforms design, in particular for video decoding applications. To reduce video decoding energy consumption while delivering high performance, one proposed solution is the use of hardware (HW) video decoding performed by a HW decoder Intellectual Property (HDIP), such as the HEVC decoder [37, 11]. Actually, in a state-of-the-art work, dedicated

13th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and 11th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM 2022).
Editors: Francesca Palumbo, João Bispo, and Stefano Cherubin; Article No. 4; pp. 4:1–4:13
OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

processors outperform general purpose processors (GPPs) by around 1000× in terms of energy efficiency [23]. As a consequence, most modern smartphones are equipped with an HDIP [39, 28]. However, HDIPs are not flexible and are costly to implement, which generate a long time-to-market for new video codecs [19].

Heterogeneous multi-cores GPPs embedded in mobile platforms are composed of cores of different performances, e.g., ARM big.LITTLE architecture processors. They offer a great opportunity to enhance both performance and energy efficiency of software (SW) video decoding using parallelism and frequency scaling. In particular, HEVC is a parallel-friendly video codec as it supports different parallelism schemes [36]. Furthermore, GPPs are flexible as they allow developing and rapidly deploying new codecs. Therefore, if well exploited, GPPs energy consumption may be as close as possible to that of the HDIP of the target platform while satisfying the real-time decoding constraint.

To exploit the heterogeneity of modern GPPs, one should balance the video frames workload among GPP cores carefully. However, video frames, in a same video sequence, have different complexities. In addition, predicting the complexity of a frame is a challenging task as the information about it are normally not known before to start the decoding process, except at the crude level of whether a frame is of type I/P/B and its size.

Dynamic voltage and frequency scaling (DVFS) is a technique that addresses the variability of the video workload to reduce the consumed energy. It is enabled by scaling down/up the voltage (and frequency) based on the frame complexity. However, despite the possibility of decoding each video frame at a different frequency, Jensen's inequality [24] shows that decoding several frames at the average frequency gives better energy efficiency than decoding each frame at a different frequency.

The research questions (RQ) addressed in this paper are as follows:

1. RQ1: How to balance the video frames among the heterogeneous GPP cores, based on a little amount of information on the video frame to decode ?
2. RQ2: Once the GPP cores are selected, how to adjust their frequency in order to reduce the energy consumption ?

In this paper, we propose a solution composed of three phases:

1. Modeling of frame complexity: to establish a model able to classify video frames into two groups: (i) most complex frames, and (ii) 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. This phase solves RQ1.
3. Frequency scaling using feedback control (PI controller) with DVFS: to monitor the output buffer size in order to adjust the GPP frequency, here we reused the work in [27]. This phase solves RQ2.

The results show that, on the tested platforms, the proposed solution can save on average more than 20% of energy (mJ/Frame) compared to the Ondemand Linux governor. Moreover, the classification allows to exploit the heterogeneity of ARM big.LITTLE architecture by limiting the miss-rate to less than 5% of decoded frames. Finally, the proposed solution is very light as it represents on overage less than 1% of the frame decoding time.

Section 2 gives some background. Then, Section 3 reviews some related work. Our contribution is described in Section 4 with experimental results in Section 5. Finally, we conclude in Section 6.

2 Background

2.1 HW video decoding

HDIPs are massively parallel. Their architectures have been optimized for such parallelism by eliminating the power consumption related to instruction decoding and control logic characterizing GPPs [22]. For example, they integrate extreme multi-threading HW or specific data handling and memory access optimization HW [3]. The main advantage of such accelerators is their energy efficiency. For that, video decoding functions, in general, exhibit massive data parallelism thanks to some schemes proposed by video codecs.

The GPP generally communicates with the HDIP as an input/output (I/O) operation. This inter processor communication (IPC) may generate some energy overhead [21, 26]. 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 energy overhead.

2.2 SW parallel processing

To understand how architectural strategies can provide high processing performance at low power levels, it is necessary to look at the CMOS circuit dynamic power consumption equation. The dynamic energy of a CMOS circuit can be formulated as:

$$E_{\text{dyn}} = P_{\text{dyn}} * t \quad (1)$$

$$P_{\text{dyn}} = K.C_{\text{eff}}.f.V_{dd}^2 \quad (2)$$

where P_{dyn} is the dynamic power, K is a constant, C_{eff} is the circuit effective capacitance, f is the circuit clock frequency, and V_{dd} is the circuit voltage [16].

For instance, running a process using 2 GPP cores clocked at $\frac{f}{2}$ can save 2× of the consumed energy compared to using 2 GPP cores clocked at f. More energy can be saved in the case where the voltage V_{dd} is scaled with the frequency f. Therefore, by decreasing the frequency to the lowest level that provides the required performance, one can significantly reduce the consumed energy.

3 Related work

The studies conducted on video decoding energy consumption can be grouped according to the decoding parallelism scheme: (i) tiling, (ii) wavefront parallel processing (WPP), and (iii) frame-by-frame.

Tiling parallelism scheme

This scheme is supported by HEVC. In [41, 35, 12], the proposed solutions consist in scheduling frame tiles among heterogeneous cores, in a mobile asymmetric multi-cores architecture, e.g., ARM big.LITTLE. The scheduling is based on the tile complexity and the performance ratio between big and LITTLE cores. The tile complexity can be estimated by its resolution, the number of PUs that it incorporates, or the number of bits encoded in each CTU[1] of the tile.

[1] PU (Prediction Unit) and CTU (Coding Tree Unit) are sub-parts of a tile. CTU is the basic processing unit of HEVC decoding process (conceptually corresponding to a Macro-block in prior standards) [32].

WPP parallelism scheme

HEVC supports also the WPP parallelism scheme. In [34], the authors developed a strategy based on task migration between big and LITTLE cores such that all cores are busy all the time. In [17], 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.

Frame-by-frame parallelism scheme

Most state-of-the-art studies exploited the frame-by-frame parallelism scheme to reduce energy consumption of SW video decoding. For instance, in [29], the authors proposed a solution that dynamically adapts the processing frequency to the video frames characteristics. The complexity of the $(L+1)^{th}$ frame is estimated by the average decoding time of the last L decoded frames, L being a parameter.

In [31], the authors introduced a method 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. In [20, 18], the authors exploited the SIMD and multi-threading to decode multiple frames in parallel, in addition to low-power states that reduce the active and idle powers.

Approximate computing can also be used to save energy when performing video decoding [30]. It consists in skipping some modules or replacing them by others of lower complexity.

In [27], the authors proposed a solution to reduce the energy consumption of video decoding using a PI controller. The proposed solution controls the GPP frequency 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. This technique were reused in our contribution.

Discussion

All the aforementioned studies suffer from at least one of the following drawbacks. First, they do not take into account the multi-cores GPPs heterogeneity of mobile platforms. Second, they rely on detailed information to predict the complexity of a frame, e.g., number of bits contained in a CTU. However, these information are not available before to start decoding except if they are collected at the encoder side and are standardized. Third, they modify the decoding algorithm which makes it not complaint to the standard.

In this paper, a solution based on parallelism and DVFS is proposed to save energy when performing video decoding. It is compliant to the HEVC standard. The proposed solution is composed of three phases: (1) modeling of frame complexity, (2) classification of video frames for an adaptive assignment, and (3) frequency scaling using feedback control with DVFS (as introduced in [27]).

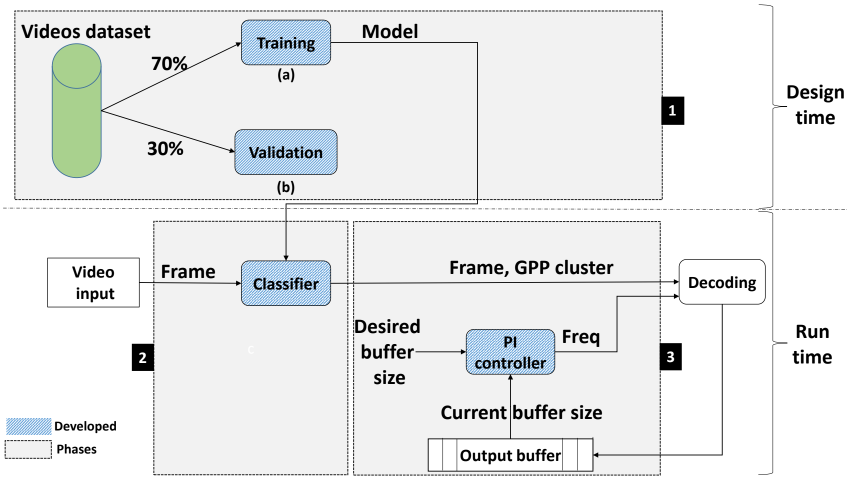

Figure 1 The proposed solution overview.

4 Contribution

In this paper, we propose a solution to reduce the energy consumption of HEVC SW decoding on mobile platforms. It aims at diminishing the HEVC SW decoding energy consumption to be as close as possible to that of the HEVC HDIP of the target platform while satisfying the real-time decoding constraint. This is achieved by two mechanisms: parallelism and DVFS.

Our proposed solution includes three different phases: (1) modeling of frame complexity, (2) assignment of frames to appropriate GPP cores using classification, and (3) frequency scaling using a PI controller with DVFS. Fig.1 depicts an overview of the proposed solution. 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. Phase 2 solves RQ1 and phase 3 solves RQ2.

4.1 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. Indeed, this work focuses on heterogeneous processors containing high performance cores and energy-efficient ones (such as ARM big.LITTLE processors). For that, any given video frame is classified 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.

To build the model of frame complexity, as illustrated in Fig.1, 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. In the second step, the model is applied on the remaining data (30%) to evaluate its accuracy.

Fig.2 depicts the inputs and output of the established model. The output is: (i) the group of most complex frames, or (ii) the group of least complex frames. The inputs are the independent variables (a.k.a. features): frame type and frame size. In a previous work [13],

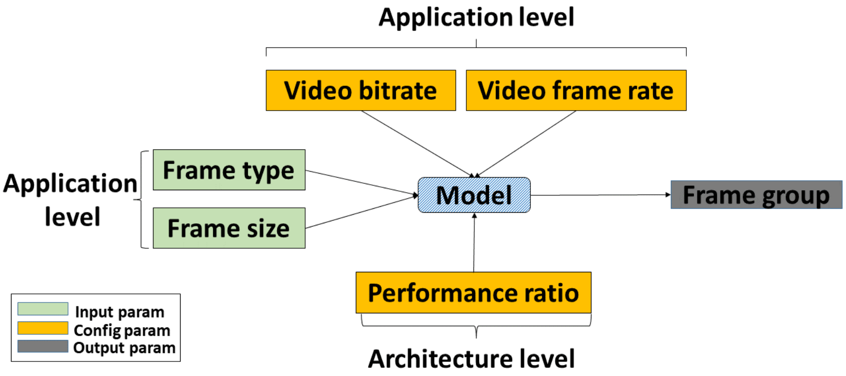

Figure 2 The proposed solution: phase 1 (Modeling of frame complexity).

it has been shown that these two parameters are very correlated to the frame complexity in MPEG video codec. However, in case of HEVC, our experiments revealed a weak correlation ($R^2 = 0.55$). As a result, we added three configuration parameters to improve the accuracy of the model.

The added parameters are: (i) video bitrate, (ii) video frame rate, and (iii) the performance ratio between the high performance and energy-efficient GPP cores. First, it has been shown that the video bitrate is correlated with the video decoding energy consumption in case of HEVC SW decoding [14]. Second, the video frame rate is used to determine the frame decoding deadline. Finally, the performance ratio is used as the GPP cores offer heterogeneous performances.

Finally, the regression model used to train the video frames data is the logistic regression [25]. The reason of this choice is its simplicity of implementation and its efficiency to take a binary decision (most complex or least complex frames) [25].

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

$$y = \frac{1}{1 + e^{-p(x1,x2,x3)}} \tag{3}$$

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

The linear function is formulated as follows:

$$p(v_{\text{bitrate}}, f_{\text{type}}, f_{\text{size}}) = \frac{w_0 + w_1 * v_{\text{bitrate}} + w_2 * f_{\text{type}} + w_3 * f_{\text{size}}}{ratio\_performance} \tag{4}$$

where w_0 is the intercept (a constant), 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.

4.2 Phase 2: GPP cores assignment using classification

The objective of the second phase is to decide online which GPP cores (high performance or energy-efficient) will decode the next frame. For that, the model built in the previous (offline) phase classifies the frames into two groups: (i) most complex frames, and (ii) least

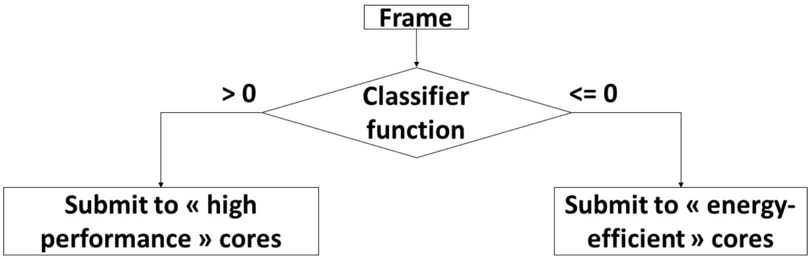
Frame
Classifier function
> 0
<= 0
Submit to « high performance » cores
Submit to « energy-efficient » cores

Figure 3 The proposed solution: phase 2 (Assignment using classification).

complex frames. The first group is 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 as illustrated in Fig.1. It is applied for each frame of the input video.

The classification is realized using Equation (4). Fig.3 depicts the algorithm of classification in a graphical representation. For each frame, the classifier function using Equation (4) is applied. If the result is positive, the frame is submitted to the high performance GPP cores; otherwise, it is submitted to the energy-efficient GPP cores. Note that if two or more consecutive frames are submitted to the same GPP cores type, they are stored in an input buffer.

4.3 Phase 3: Frequency scaling using PI controller [27]

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 [27] is adopted. This controller monitors the output buffer size in order to maintain it at a desired value (set point) which is an input parameter of the controller. To set the desired value, one can follow the guidelines given in the literature review, such as [27, 18]. This phase is performed during run-time, as illustrated in Fig.1. It is applied for each frame of the input video.

The PI controller has two inputs: (i) a set point, i.e., the desired output buffer, and (ii) the current output buffer size. Then, according to the output buffer size, the PI controller adjusts the GPP frequency, using DVFS, so that the current buffer size meets the set point. Note that the controller is engaged only when the difference between the set point and the current output buffer size is not zero.

To speed up or slow down the GPP cores, the GPP frequency, *gpp_cores_freq*, is calculated by multiplying the highest supported GPP cores frequency, $GPP\_\mathrm{cores}_{\mathrm{max\_freq}}$, by a scaling factor, r.

$$gpp\_cores\_freq = GPP\_\mathrm{cores}_{\mathrm{max\_freq}} * r \tag{5}$$

The scaling factor, r, is, in turn, decomposed into two components, as illustrated by the following formula:

$$r(n) = r_{\mathrm{e}}(n) + r_{\mathrm{c}}(n) \tag{6}$$

where r_{e} is the scaling factor estimation based on the history of the decoded frames, r_{c} is the output of the PI controller which is considered as an adjustment of r_{e} to compensate the missed deadlines in the past, 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.

5 Performance evaluation

5.1 Evaluation methodology & setup

This section describes the evaluation methodology.

5.1.1 Experimentation setups

The HW and SW experimental setups as well as datasets (33 video sequences) on which the proposed solution was applied are presented and summarized in Table 1. First, the experiments were carried out on two different platforms (Snapdragon 810 and Odroid-xu3 for HW and SW video decoding, respectively). Then, the same experiments were performed on a single platform (RB3) for both HW and SW video decoding.

Table 1 Experimental setups.

	Snapdragon 810 [1]	**Odroid-xu3** [7]	**RB3** [8]
HW setup			
HEVC HW	Supported	Not supported	Supported
HEVC SW	Not supported	Supported	Supported
Power measurement	N6705A DC Power Analyzer [14]		
SW setup			
OS	Android 6.0 Linux kernel 3.10.84	Ubuntu 16.04 Linux kernel 4.14.176+	Linaro Linux 10.3 Linux kernel 5.4.0
HEVC HW decoder	Android application + Mediacodec API	–	Open-HEVC [5]
HEVC SW decoder	–	Open-HEVC [5]	Ffmpeg [4] + v4l2 library
Video sequences datasets : 33 video sequences			
JCT-VC [15], Jellyfish [6], and some well-known video sequences on the web, e.g., [38]. Resolution: 1080p. Frame rate: 25, 30, and 50 fps. Mode: Random Access. Profile: Main			

To build the model of phase 1 of the proposed solution, sickit-learn framework [9] was used via Python programming language.

5.1.2 Methodology

We evaluated our solution in four steps.

First, the proposed solution is compared to 5 state-of-the-art solutions, as summarized in Table 2. In the first one, no DVFS is applied (Performance governor), the second uses the Ondemand governor [33]. The characterization method is not a real strategy. We have extracted the best configuration (number of GPP cores and their frequency) by testing all possible configurations offline (whereas the proposed solution selects it dynamically without such an effort). We also compared to the solution based only on PI controller [27] to show the impact of the classification we proposed. We finally compared the energy consumption with the one of the HDIP to evaluate how far is our solution from it.

Second, the accuracy of the model built in phase 1 is evaluated.

Table 2 The proposed video decoding energy consumption optimization summary.

Proposed solution	Classification + DVFS (PI controller)
State-of-the-art work	No DVFS (Performance Linux governor) [33]
	DVFS (Ondemand Linux governor) [33]
	Characterization proposed in [14]
	DVFS (PI controller) [27]
	HW decoding (HDIP) of the target platform

Table 3 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)	Characterization work [14]	Open-HEVC + DVFS (PI controller) [27]
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

Third, the stability of the output buffer, which is its occupation variation, is studied. We consider a system stable when the cores frequency does not change more than once in a second since this period is usually used to make group of pictures (GoP), e.g., for streaming applications.

Fourth, the overhead of the proposed solution (in percentage) is evaluated using the following formula:

$$ratio\_overhead = \frac{ps\_time}{frame\_dec\_time} * 100 \tag{7}$$

where *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.

5.2 Results and discussion

In this section, the results of the HEVC SW decoding energy consumption optimization are described and analyzed.

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

In case of Snapdragon 810 and Odroid-xu3 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 suitable GPP cluster and its clock frequency, in contrast to the characterization solution [14], but also can save up to 7% of energy (mJ/Frame). The classification technique, phase 2, brings on average 20% of energy saving as compared to [27]. Finally, the ratio of energy between the proposed solution and the HW video decoding is 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 PI controller solution, respectively. Concerning the HW video decoding, our solution consumes on average 4× more energy.

On all tested platforms, the miss-rate represents on average less than 5% of any given video sequence. 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 [40].

Table 3 summarizes the energy saving percentage of the proposed solution over state-of-the-art work.

5.2.2 Accuracy of the model

The model of frame complexity was trained with 70% of video frames dataset extracted from 33 videos sequences representing different durations and scenarios. The remaining 30% was used for validation. 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. This can be corrected by the PI controller of phase 3 (the miss rate was smaller then this proportion).

5.2.3 Stability of the output buffer

At the beginning of the decoding process, 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 in less than one second of decoding. The GPPs cores frequency changes at most once in a second of decoding. The fluctuation of output buffer size is due to the frames complexity which changes from one frame to another.

5.2.4 Overhead of the proposed solution

The overhead of the proposed solution is evaluated here. It is calculated using Equation (7). 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.

6 Conclusions & future work

This paper presents a solution to reduce the energy consumption of HEVC decoding on a heterogeneous mobile platform. The proposed solution is split into three phases: (1) modeling of frame complexity, (2) assignment of frames to appropriate GPP cores using classification, and (3) frequency scaling using a PI controller with DVFS. Phases 2 and 3 solves RQ1 and RQ2, respectively.

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. 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 the overhead, the proposed solution is very slight as it represents on average less than 1% of the frame decoding time. Finally, it should be noted that the HW video decoding presents the best trade-off between performance and energy consumption at the system level point of view.

In our future work, the aim is to apply our methodology to the successor of the HEVC standard, versatile video coding (VVC).

References

1 APQ8094 | Qualcomm. URL: https://www.qualcomm.com/products/apq8094.

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.

3 Development of the VPU | Jon Peddie Research. URL: https://www.jonpeddie.com/blog/development-of-the-vpu/.

4 Download FFmpeg. URL: https://ffmpeg.org/download.html.

5 GitHub - OpenHEVC/openHEVC at ffmpeg_update. URL: https://github.com/OpenHEVC/openHEVC/tree/ffmpeg_update.

6 Jellyfish Bitrate Test Files. URL: http://jell.yfish.us/.

7 Odroid-xu3 – odroid. URL: https://www.hardkernel.com/shop/odroid-xu3/.

8 Qualcomm® Robotics RB3 Development Kit. URL: https://www.qualcomm.com/products/qualcomm-robotics-rb3-platform.

9 scikit-learn: machine learning in python – scikit-learn 0.24.2 documentation. URL: https://scikit-learn.org/stable/.

10 Video usage is soaring. will it last? URL: https://newsroom.cisco.com/feature-content?type=webcontent&articleId=2080343.

11 F. Amish and E.B. Bourennane. Fully pipelined real time hardware solution for high efficiency video coding (hevc) intra prediction. *Journal of Systems Architecture*, 64:133–147, 2016. Real-Time Signal Processing in Embedded Systems.

12 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)*, pages 4298–4302, 2015. doi:10.1109/ICIP.2015.7351617.

13 A.C. Bavier and A.B. Montzand L.L. Peterson. Predicting mpeg execution times. *SIGMETRICS Perform. Eval. Rev.*, 26(1):131–140, June 1998.

14 M. Bey Ahmed Khernache, Y. Benmoussa, J. Boukhobza, and D. Menard. Hevc hardware vs software decoding: An objective energy consumption analysis and comparison. *Journal of Systems Architecture*, 115:102004, 2021.

15 F. Bossen. Common test conditions and software reference configurations. *JCTVC-L1100*, 12, 2013.

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*, volume 1, pages 288–297 vol.1, January 1995. doi:10.1109/HICSS.1995.375385.

17 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. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1827–1838, 2012. doi:10.1109/TCSVT.2012.2223056.

18 C.C. Chi, M. Alvarez-Mesa, and B. Juurlink. Low-power high-efficiency video decoding using general-purpose processors. *ACM Trans. Archit. Code Optim.*, 11(4), January 2015.

19 K. Choi and E.S. Jang. Leveraging parallel computing in modern video coding standards. *IEEE MultiMedia*, 19(3):7–11, July 2012. doi:10.1109/MMUL.2012.36.

20 Y. Duan, J. Sun, L. Yan, K. Chen, and Z. Guo. Novel efficient hevc decoding solution on general-purpose processors. *IEEE Transactions on Multimedia*, 16(7):1915–1928, 2014. doi:10.1109/TMM.2014.2337834.

21 J. Golston, S. Arora, and R. Reddy. Optimized video decoder architecture for TMS320C64x DSP generation. In Bhaskaran Vasudev, T. Russell Hsing, Andrew G. Tescher, and Touradj Ebrahimi, editors, *Image and Video Communications and Processing 2003*, volume 5022, pages 719–726. International Society for Optics and Photonics, SPIE, 2003.

22 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. *SIGARCH Comput. Archit. News*, 38(3):37–47, June 2010.

23 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)*, pages 10–14, February 2014.

24 C. Im, S. Ha, and H. Kim. Dynamic voltage scheduling with buffers in low-power multimedia applications. *ACM Trans. Embed. Comput. Syst.*, 3(4):686–705, November 2004.

25 N.A. Kudryashov. Logistic function as solution of many nonlinear differential equations. *Applied Mathematical Modelling*, 39(18):5733–5742, 2015.

26 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. *Journal of Systems Architecture*, 93:1–19, 2019.

27 Z. Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. In *Proceedings 21st International Conference on Computer Design*, pages 489–496, 2003. doi:10.1109/ICCD.2003.1240945.

28 B. Moyer and Y. Watanabe. Chapter 13 – hardware accelerators. In Bryon Moyer, editor, *Real World Multicore Embedded Systems*, pages 447–480. Newnes, Oxford, 2013.

29 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)*, pages 1–6, 2015. doi:10.1109/ICME.2015.7177406.

30 E. Nogues, D. Menard, and M. Pelcat. Algorithmic-level approximate computing applied to energy efficient hevc decoding. *IEEE Transactions on Emerging Topics in Computing*, 7(1):5–17, 2019. doi:10.1109/TETC.2016.2593644.

31 E. Nogues, A. Mercat, F. Arrestier, M. Pelcat, and D. Menard. Convex energy optimization of streaming applications for mpsocs. *ICASSP 2019 – 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1557–1561, 2019. doi:10.1109/ICASSP.2019.8682317.

32 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). *IEEE Transactions on Circuits and Systems for Video Technology*, 22:1669–1684, 2012.

33 Venkatesh Pallipadi and Alexey Starikovskiy. The ondemand governor. In *Proceedings of the linux symposium*, volume 2(00216), pages 215–230, 2006.

34 R. Rodríguez-Sánchez and E.S. Quintana-Ortí. Architecture-aware optimization of an hevc decoder on asymmetric multicore processors. *Journal of Real-Time Image Processing*, 13:25–38, March 2017.

35 H.J. Roh, S.W. Han, and E.S. Ryu. Prediction complexity-based hevc parallel processing for asymmetric multicores. *Multimedia Tools and Applications*, 76:25271–25284, December 2017.

36 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. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1858–1870, December 2012. doi:10.1109/TCSVT.2012.2223052.

37 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. *IEEE Journal of Solid-State Circuits*, 49(1):61–72, January 2014.

38 Xiph.org. Xiph.org :: Derf's Test Media Collection. URL: http://media.xiph.org/video/derf/.

39 K. Xu, T.M. Liu, J.I. Guo, and C.S. Choy. Methods for power/throughput/area optimization of h.264/avc decoding. *Journal of Signal Processing Systems*, 60(1):131–145, July 2010. doi:10.1007/s11265-009-0408-6.

40 F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 374–382, 1995. doi:10.1109/SFCS.1995.492493.

41 S. Yoo and E.S. Ryu. Parallel hevc decoding with asymmetric mobile multicores. *Multimedia Tools and Applications*, 76:17337–17352, August 2017.