On the Energy Efficiency of Parallel Multi-core *vs* Hardware Accelerated HD Video Decoding

Yahia Benmoussa Univ. Bretagne Occidentale, UMR6285, Lab-STICC, France. Univ. M'hamed Bougara, LIMOSE, Algeria
yahia.benmoussa@univ-brest.fr

Jalil Boukhobza
Univ. Bretagne Occidentale, UMR6285, Lab-STICC, F29200 Brest, France
jalil.boukhobza@univ-brest.fr

Eric Senn
Univ. Bretagne Sud, UMR6285, Lab-STICC, F56100 Lorient, France
eric.senn@univ-ubs.fr

Djamel Benazzouz
Univ. M'hamed Bougara, LMSS, Boumerdes, Algeria
dbenazzouz@yahoo.fr

ABSTRACT

Hardware video accelerators are used on mobile devices to provide support for energy efficient real time High definition (HD) video decoding. Recently, the rise of multi-core architectures on those devices increased their performances and make real time HD video decoding possible using parallel processing on the GPP cores only. What is even more interesting to know is the level of energy efficiency these kind of multi-core General Purpuse Processor (GPP) can achieve as compared to hardware video accelerators. In this paper, we propose an experimental evaluation of the energy efficiency of the two video decoding approaches. An accurate energy measurement was achieved on a recent low-power 40 nm mobile SoC containing a quad-core ARM processors and a video hardware accelerator. The results show that parallel multi-core HD decoding enhances both the performance and the energy efficiency as compared to the use of a single core. However, the hardware accelerated decoding is about three times more energy efficient. Based on the experimental observations, some challenges for enhancing parallel multi-core video decoding energy efficiency are pointed out.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems; D.4.8 [**Operating Systems**]: Performance; C.3 [**Special Purpose and Application Based Systems**]: Real-time and embedded systems

Keywords

Parallel video decoding, Energy efficiency, multi-core SoC

EWiLi'14, 13-14 November 2014, Lisbon, Portugal.

1. INTRODUCTION

Video decoding is both processing intensive and real time application. To fulfill these constraints, the processor equipping the mobile devices may need to run at more and more high frequency especially in the context of an increasing demand on HD videos.

However, due to the thermal and power issues faced in the design of modern microprocessors, it is no longer possible to increase continuously the clock frequency. In fact, using high frequencies leads to a drastic increase in the thermal dissipation and the energy consumption due to the quadratic relation between the dynamic power consumption and the clock frequency. This is more critical in the case of energy constrained mobile devices such as smartphones and tablets. To overcome this issue, modern embedded processor architectures use the parallelism to increase the performance without the need to increase the frequency [9].

In the field of video decoding, the parallelism can enhance the energy efficiency on the energy constrained device. It can be implemented in a specialized hardware video accelerators whose energy efficiency is well established [18]. However, the hardware accelerators are a proprietary solutions and lack of flexibility. In fact, they are not open and their use depend on some API provided by the vendor. Moreover, it may take a long time to implement a new video standard on hardware circuits unlike the software based solutions running on GPP. For example, the latest mobile device still does not support hardware accelerator for the new HEVC standard.

Recently, the new SoC equipping mobile devices include more and more GPP cores. For example, the latest ARM big.LITTLE architecture contains four Cortex A7 and four Cortex A15 processors [3]. What is even more interesting to know is the level of performance and energy efficiency these kind of multi-core GPP processors can achieve as compared to hardware video accelerators. The objective is to provide a video decoding solution that conciliates both the energy efficiency and the flexibility of video decoding.

In this study, we investigate the performance and energy efficiency of parallel multi-core video decoding as compared to the hardware accelerator based approach. For this purpose, we propose an experimental methodology based on power consumption measurement achieved on an embedded platform containing four GPP cores and a video hardware accelerator. The obtained results showed that the hardware accelerator is three time more energy efficient than the optimal parallel multi-core video decoding. Moreover, they allowed to point out some challenges to enhance the energy efficiency of the parallel multi-core video decoding.

The remainder of this paper is organized as follows : Related works on energy consideration of parallel video decoding are discussed in section 2. In section 3, some background material regarding the power consumption and the energy efficiency of parallel video decoding is presented. The experimental methodology and the obtained results are described in section 4 and 5 respectively. Finally, the conclusions and some future work perspectives are given in section 6.

2. RELATED WORKS

In [10], is introduced some architecture design basis of hardware accelerated H.264/AVC HD video decoding. The advantages, in terms of performance and energy consumption, of H.264/AVC video decoding using hardware accelerator are highlighted in [18]. In the same way, a more general study [12] investigates the reasons of energy inefficiency of GPPs and proposes guidelines to reduce the energy breakdown as compared to video hardware accelerator. The energy efficiency of hardware accelerated video decoding is well established, however, the video standards evolve quickly and hardware vidoe accelerator does not provide the flexibility to adapt to those changes [16].

DSP-based solutions aim to conciliate the flexibility of GPPs and the energy efficiency of hardware accelerator. In [9], the authors focus on the performance and energy efficiency of DSPs due to the use of pipeline and parallelism in CMOS circuits. In [6, 5], the authors compare the performance and the energy efficiency of GPP and DSP. However, the HD video decoding was not considered in these studies. Moreover, DSP-based video decoding seems to be abandoned by mobile device manufacturer in favor of hardware video accelerators which are more energy efficient.

With the rise of modern SoC integrating more and more processor cores, many studies investigated the performance and the energy efficiency of parallel video decoding on these architectures. In [13], they compared different video decoding parallelism levels (MB, slice and frame) on multi-core architecture. In [4], the authors focus on the energy efficiency of parallel H.264/AVC decoding on multi-core processor. They have evaluated the energy saving as compared to mono-core decoding. However, they have not considered hardware acceleration in their study.

In this study, we propose a comprehensive experimental methodology to investigate the energy efficiency of parallel multi-core video decoding as compared to that based on hardware video accelerators. As far as we know, no prior work provided a clear evaluation data of the two approaches.

3. BACKGROUND

We describe hereafter some elementary background related to the energy consumption in electronic -circuits and the role of parallelism in reducing the energy consumption especially in case of video decoding.

3.1 Energy consumption

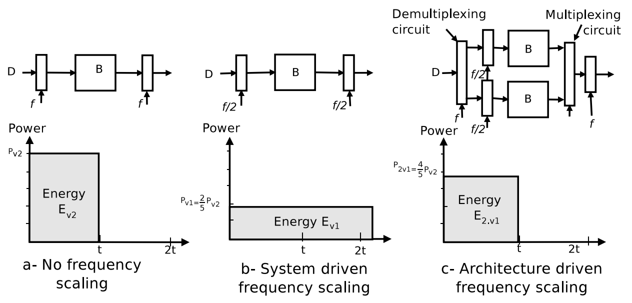


Figure 1: Architecture driven frequency scaling

In CMOS digital circuits, the total power consumption is the sum of the static and dynamic power :

$$P_{tot} = P_{static} + P_{dyn} \quad (1)$$

where P_{static} and P_{dyn} are defined as :

$$P_{static} = I_{leak}.V \quad (2) \qquad P_{dyn} = C_{eff}.V^2.f \quad (3)$$

I_{leak} is the leakage current, V is the supply voltage associated to the clock frequency f and C_{eff} is the circuit effective capacitance [9].

The static power is related to the circuit fabrication technology and does not depend on its activity. Below 65-nm circuits feature size, it becomes significant and poses new low-power design challenges [14]. On the other hand, the dynamic power is related to the circuit activity. For example, in case of a microprocessor, the dynamic power depends on the type of instructions executed and on the data accessed. In equation 3, this is represented by the C_{eff} parameter defined as $C_{eff} = A.C$, where C is the circuit capacitance and A is the the activity factor.

Figure-1 illustrates a simplified representation of a CMOS circuit which processes a set of sequential data D (encoded video frames) using a block B (video decoder). The block B operates at frequencies $\frac{f}{2}$ and f corresponding to the supply voltage levels $V_1 = 0.925V$ and $V_2 = 1.15V$ respectively[1]. If t is the processing time when B operates at a frequency f (Figure 1-a), then the energy consumption is $E_{V_2} = P_{V_2}.t$ where $P_{V_2} = C_{eff}.V_2^2.f$. If we suppose the processing time at frequency $\frac{f}{2}$ (Figure-1-b) is doubled, then the ratio between the energy E_{V_1} consumed by the circuit at the frequency $\frac{f}{2}$ with $V_1 = 1.06V$, and E_{V_2} is :

$$\frac{E_{V_1}}{E_{V_2}} = \frac{C_{eff}.V_1^2.\frac{f}{2}.2.t}{C_{eff}.V_2^2.f.t} = (\frac{V_1}{V_2})^2 \simeq 65\%$$

In this case, scaling down the voltage and the frequency decreases the power consumption to $P_{V_1} = C_{eff}.V_1^2.\frac{f}{2} \simeq \frac{2}{5}.P_{V_2}$ which leads to 35% energy saving at the cost of a decreased performance. This may represent a scenario where the operating system scales down dynamically the processor frequency at run time when it detects a load decrease. This illustrates a system-driven voltage scaling.

In order to save energy without sacrificing performance, an architectural-driven voltage scaling [9] can be achieved by using two B blocks which are both clocked at a frequency $\frac{f}{2}$ and supplied with a voltage V_1 as described in Figure-1-c.

[1]V_1 and V_2 are the associated to the frequencies 800 and 400 MHz of the Cortex A9 processor used in our experiments.

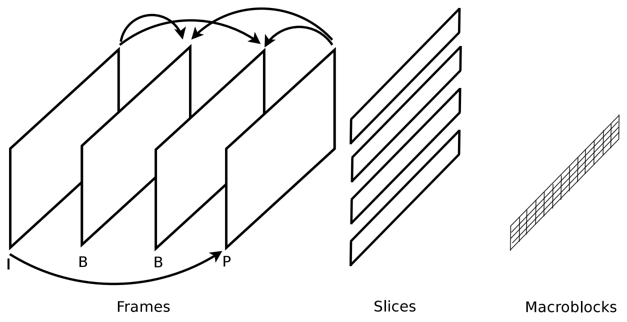


Figure 2: Video decoding parallelism levels

$P_{2.V_1}$ and $E_{2.V_1}$ refer to the power and the energy consumption associated to this configuration. Since the two blocks are operating in parallel, the execution time does not decrease and the ratio between $E_{2.V_1}$ and E_{V_2} is :

$$\frac{E_{2.V_1}}{E_{V_1}} = \frac{C_{eff}.V_1^2.\frac{f}{2}.t + C_{eff}.V_1^2.\frac{f}{2}.t}{C_{eff}.V_2^2.f.t} = (\frac{V_1}{V_2})^2 \simeq 65\%$$

In this configuration, the total power consumption $P_{2.V_1}$ is the sum of the power consumptions of the two blocks, which is equal to $2.C_{eff}.V_1^2.\frac{f}{2} = \frac{4}{5}.P_{V_2}$. The energy saving is equal to 35% without sacrificing the performance but at a cost of an additional circuit area and static power.

3.2 Parallel video decoding

As illustrated in Figure-2, a H.264/AVC video sequence is composed of a set of frames. Each frame may contain several slices and each slice contains several macroblocks (MB = 16 x 16 pixels). The H.264 standard defines three main types of slices: I, P, and B. An I slice uses intra prediction and is independent of the slices in other frames. In intra prediction a MB is predicted based on adjacent blocks. A P-slice uses motion estimation and intra prediction and depends on one or more slices in a previous frames, either I, P or B. Motion estimation is used to exploit temporal correlation between slices. Finally, B-slices use bidirectional motion estimation and depend on slices from previous and future frames. Each slice can be decoded independently of the slices within the same frame whatever its type.

The parallelism of video decoding can be achieved at a frame, slice or MB levels [15]. At a frame level, the frames may be decoded in parallel on different processing units. The drawback of such an approach is that it does not scale very well because the number of independent slices is limited at a given time. On the other hand, a higher scalability is possible at a slice level. However, this depends on the encoder setting to enable multi-slice frames. At MB level, a very good scalability can be achieved when the decoding is implemented on hardware codecs. On the other hand, parallel MB decoding on many core processors is not efficient due to a considerable inter-processor and synchronization overhead [2].

In this study, we compare slice-based parallelism on multi-core ARM processors and hardware accelerated video decoding using MB level parallelism.

4. METHODOLOGY

The proposed experimental methodology aims to compare between the energy efficiency of parallel multi-core and hardware accelerated HD video decoding. It is based on power consumption measurement on a real embedded platform containing a multi-core processor and a hardware video codec. We describe hereafter the used hardware and software, then the performance and energy consumption measurement methodology.

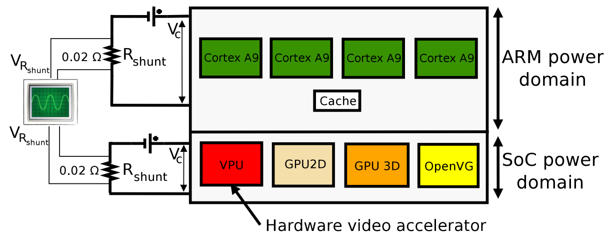


Figure 3: i.MX6 SoC power domains

Applications	Videos	Test sequences	Big Buck Bunny
		Rate	24 Frames/s
		Resolution	720p (1280x720)
		Bit-rate (Kb/s)	2Mb/s
	GStreamer	ARM codec plug-in	ffdec_h264
		HW codec plug-in	vpudec
	Linux	Kernel version	Linux 3.10.17
		DVFS driver	cpufreq
Hardware	HW codec	Model	Proprietary (Freescale)
		Frequencies (MHz)	264
	ARM	Model	Quad-core Cortex A9 + NEON
		Frequencies (MHz)	400, 800, 1000
	SoC	Model	Freescale i.MX6 quad
		Technology	40 nm

Table 1: Hardware and software setup

4.1 Hardware Setup

We use in our experiment the *SABRE* development board containing the low-power i.MX6 Quad-core SoC. This SoC consists of a Quad Cortex A9 ARM cores and a set of specialized processing units such as a Graphical Processing Unit (GPU) and a Video Processing Unit (VPU) (See Figure-3. Each Cortex A9 processor supports 3 frequencies : 400 MHz, 800 MHz and 1000 MHz. The VPU is a hardware accelerator implementing H.264/AVC encoding/decoding standard. It is clocked at 264 MHz and supports full HD video decoding up to 60 Hz rate. In what follows, the VPU term serves to designate the video hardware accelerator.

4.2 Software Setup

On this hardware platform, the Linux operating system version 3.0.17 was used with *cpufreq* enabled to drive the ARM cores frequency scaling. The *userspace* governor was activated to allow the control of the clock frequency at the application level. The H.264/AVC video decoding was achieved using GStreamer [11], a multimedia development framework. The ARM decoding, was performed using *ffdec_h264*, an open-source plug-in based on the widely used *ffmpeg/libavcodec* library compiled with the support of NEON SIMD instructions set. For the hardware accelerated decoding, we used *vpudec*, a proprietary GStreamer H.264/AVC plug-in provided by *Freescale*. As a test video, we use the well known *Big Buck Bunny* sequence. We encode it in 720p resolution (1280x720), 2Mb/s bit-rate and 24Hz rate using *x264* encoder. We configured the encoder to set the number of slice per frame to 4 by means of the *–slices* option. The objective is to fully exploit the 4 available ARM cores on the

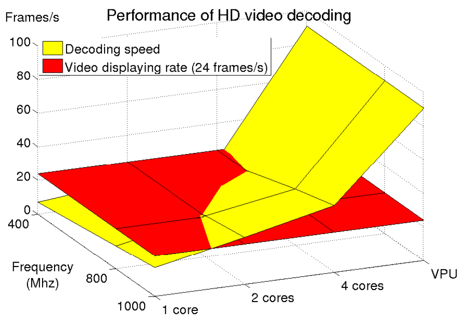


Figure 4: Performance of video decoding

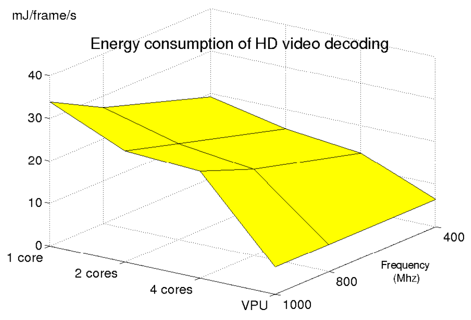


Figure 6: Energy consumption of video decoding

	400 MHz	800 MHz	1000 MHz
1 core	7,16	13,71	17,03
2 cores	12,30 (x 1.71)	24,55 (x 1.79)	28,02 (x 1.64)
4 cores	18,35 (x 2.56)	33,36 (x 2.43)	39,80 (x 2.33)
VPU	90,57 (x 12.64)	90,61 (x 6.60)	91,05 (x 5.34)

Table 2: Video decoding performances (fps)

	400 MHz	800 MHz	1000 MHz
1 core	19.16	27.24	33.85
2 cores	15.46 (x 0.80)	22.57 (x 0.82)	26.16 (x 0.77)
4 cores	13.55 (x 0.70)	20.30 (x 0.74)	25.12 (x 0.74)
VPU	6.41 (x 0.33)	6.53 (x 0.23)	6.61 (x 0.18)

Table 3: Video decoding energy (mJ/frame)

i.MX6 SoC while decoding the video. Table-1 summarizes the used hardware and software setup.

4.3 Performance measurement

We started by measuring the performance of video decoding using a single core, dual-core, quad-core decoding at all the available clock frequencies (400, 800 and 1000 MHz) and the VPU decoding. The number of cores used for decoding the video is selected by setting the value of *max_threads* parameter of the *ffdec_h264* plug-in. The VPU and multi-core video decoding is selected by choosing the corresponding *GStreamer* plug-in : (*ffdec_h264* or *vpudec*). For each configuration, we calculated the number of decoded frame per second (fps). The *libavcodec* library supports both slice and frame multi-threaded decoding. However, the *ffdec_h264* plug-in does not allow to select explicitly which method to use and the automatic selection mechanism tends to select systematically the frame-level multi-threading. To fix this issue, the plug-in was forced to use the slice-level method by setting *active_thread_type* = FF_THREAD_SLICE in the *pthread.c* source file.

4.4 Energy consumption measurement

The used *SABRE* board has two power domains which can be measured separately. The ARM power domain include the 4 ARM cores plus the cache memory and the SoC power domain include the VPU, 2DGPU, 3DGPU and the OpenVG [1]. At each power domain was inserted R_{shunt}, a 0.02 Ω shunt resistor (See Figure-3).

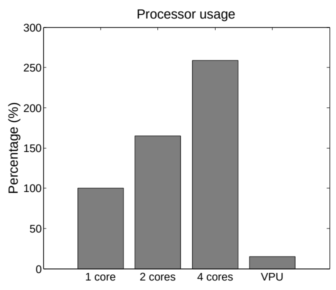


Figure 5: Video decoding processor usage

5. EXPERIMENTAL RESULTS

5.1 Performances

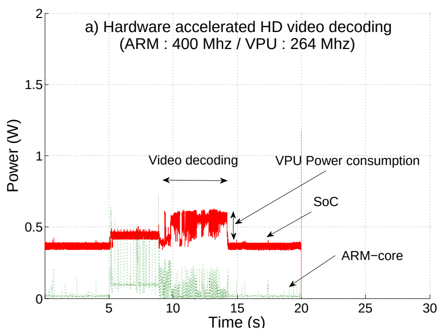
The power consumptions is then measured using the Open-PEOPLE framework [7], a multi-user and multi-target power and energy optimization platform and estimator. It includes the NI-PXI-4472 digitizer allowing up to a 100 KHz sampling resolution. At a given time, the power consumption is $P = \frac{V_c . V_{shunt}}{R_{shunt}}$. The energy consumption is obtained by summing the elementary power consumption obtained using 1 KHz sampling rate multiplied by the sampling duration.

In case of multi-core ARM video decoding, only the ARM power domain consumption is measured. On the other hand, the sum of the ARM power domain and the SoC power are measured in case of VPU decoding since both the ARM cores the the VPU are involved in the decoding process.

Table-2 shows the performances results of the video decoding. One can observe that in case of multi-core decoding, the decoding speed is higher than the displaying rate using 2 cores or 4 cores starting from 800 MHz clock frequency. In case of VPU video decoding, the decoding speed is (x 3.75) higher than the displaying rate regardless of the ARM cores frequency[2]. This is illustrated in Figure-4 where the flat red surface represents the displaying rate (24 fps).

The values between the parenthesis in Table-2 represent the performance scaling factor as compared to mono-core video decoding. One can observe that using four ARM cores allows only x2.4 performance increase. This is mainly due to the unbalanced workload. In fact, the video encoder divides each frame into equal-size slices. However, the decoding workload depends on the slice scene complexity. Thus, a decoding thread assigned to a given slice may terminate be-

[2]The frequency of the VPU frequency (264 MHz) remains constant when varying the frequency of the ARM cores



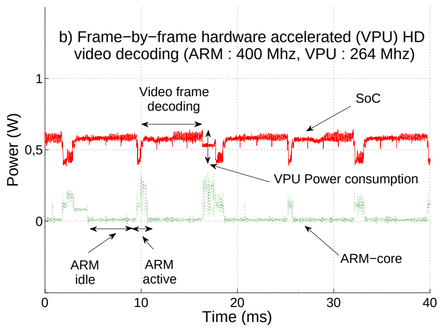


Figure 7: VPU Video decoding power consumption

fore the other ones. During this time, it goes into a blocked status waiting the other threads to terminate.

On the other hand, the scaling factor is much more higher (from x5 to x12) in case of VPU decoding. This is due to MB level parallelism implemented in the VPU.

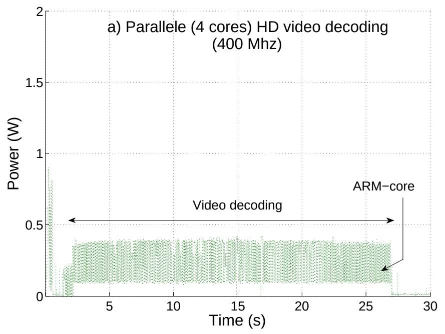
The measured processor usages[3] illustrated in Figure-5 confirm these observations. In fact, in case of single-core video decoding (one thread), the processor usage is 100% which means that the decoding thread is all time in active state. However, it is around 160% and 260% in case of dual-core and quad-core decoding respectively. On the other hand, when using the VPU, the processor usage is about 15% because the ARM cores are almost time in idle mode waiting for the frame to be decoded by the VPU.

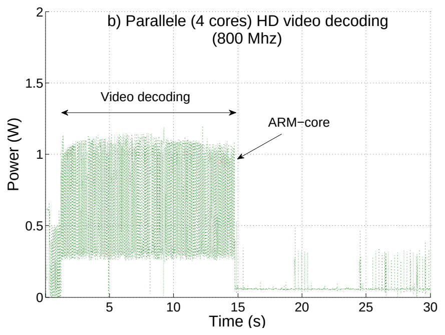
5.2 Energy consumption

Table-3 shows the energy consumption of video decoding using the ARM cores and the VPU. The values between parenthesis in Table-2 represent the energy reduction factor as compared to single core decoding. As expected, for a given clock frequency, increasing the number of cores allows to reduce the energy consumption (See Figure-6). For example, as compared to mono-core decoding, the optimal multi-core configuration (4 cores, 800 MHz) deceases the energy by a factor of x0.74 while increasing the performance by a factor of x2.43.

On the other hand, the energy saving is much more important in case of VPU video decoding (0.23 scaling factor) as compared to mono-core decoding at 800 MHz and x0.36 as compared to the optimal multi-core video decoding (4 cores, 800 MHz). This can be explained by both a high decoding performance and a very low power consumption

[3]*processor usage* $= (\sum_i T_i)/T_{exe}$ where T_i is the time that the i^{th} thread got a processor core (active time), T_{exe} is the decoding time.





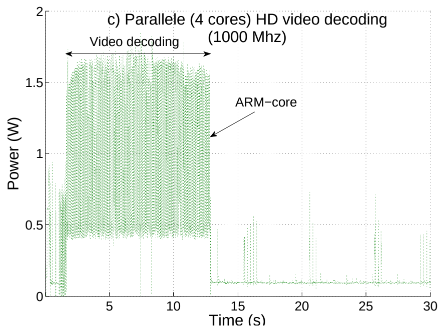


Figure 8: HD Video decoding energy consumption

of the VPU. As illustrated in Figure-7-a, one can observe that the decoding time of the 480 video frames terminated in almost 5 seconds. During this decoding phase, the power consumption of the SoC power domain increases with only 0.2 W which correspond to the VPU power consumption. This low value can be explained by the low frequency (264 MHz) of the VPU. During this time, the ARM cores power consumption is negligible. In fact, as illustrated in Figure-7-b showing the frame-by-frame power consumption variation, the ARM cores are almost time in idle state waiting the VPU to decode a video frames. In the idle state, the ARM cores execute the WFI (Wait For Interrupt) instruction were almost the processor clocks are gated to reduce the power consumption.

Unlike the VPU decoding, multi-core video decoding can not conciliate the performance and the energy efficiency. As illustrated in Figure-8, at 400 MHz frequency (See Figure-8-a), the power consumption is low ($\backsim$ 0.3 mW), but the decoding time is very long. On the other hand, at the higher frequencies, the decoding time is reduced but the power consumption increases considerably (See Figure-8-b and c).

One can highlight that the unbalanced workload over the processor cores may be source of energy inefficiency. In fact, during a thread waiting time, the processor core continues to consume energy while doing nothing. One approach to fix this issue is to set the clock frequency of each core depending on the slice decoding workload or to transit a processor core to low power mode during its inactivity using Dynamic Power Management (DPM) as proposed in [17]. However, this is not possible in case of the used *i.MX6* SoC since it does not support a per-core DVFS/DPM.

6. CONCLUSION

This paper is a use case study based on the *i.MX6* SoC. The experimental results showed that multi-core video decoding allows to enhance both the performance and the energy efficiency of HD video decoding as compared to single core decoding. However, the hardware video accelerator is three time more energy efficient than multi-core optimal multi-core video decoding.

Although, these results may be different on other architecture, the obtained data allows to have a general idea the the energy consumption levels of HD video decoding on a recent heterogeneous SoC.

According to the rapid evolution of the SoC which tend to integrate more and more cores [8], one may expect that the energy efficiency of multi-core video decoding can be enhanced if a larger number of cores are used [19]. Moreover, as pointed out in the results discussion, the energy efficiency of multi-core video decoding may also be enhanced if it is combined with per-core DVFS/DPM strategies. The objective is to avoid wasting the energy due to *idling* the cores. We plan to investigate these issues in a future works using the Exynos5 SoC containing 8 cores supporting a per-core DVFS/DPM.

Acknowledgment

This work was supported by BPI France, Région Ile-de-France, Région Bretagne and Rennes Métropole through the French Project GreenVideo.

7. REFERENCES

[1] *i.MX 6Dual/6Quad Power Consumption Measurement*, Freescale Semiconductor, 2012.

[2] M. Álvarez Mesa, A. Ramírez, A. Azevedo, C. Meenderinck, B. Juurlink, and M. Valero. Scalability of macroblock-level parallelism for h. 264 decoding. In *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, pages 236–243. IEEE, 2009.

[3] ARM. big.little processing. http://www.arm.com/products/processors/technologies/biglittleprocessing.php, 2014.

[4] E. Baaklini, S. Rethinagiri, H. Sbeity, and S. Niar. Scalable row-based parallel h.264 decoder on embedded multicore processors. *Signal, Image and Video Processing*, pages 1–15, 2014.

[5] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz. Energy consumption modeling of h.264/avc video decoding for gpp and dsp. *in Proceedings of 16th Euromicro Conference on Digital System Design*, 2013.

[6] Y. Benmoussa, J. Boukhobza, E. Senn, and D. Benazzouz. GPP vs DSP: A performance/energy characterization and evaluation of video decoding. *in Proceedings of the IEEE 21st International Symposium On Modeling, Analysis And Simulation Of Computer And Telecommunication Systems*, 2013.

[7] 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 Proceedings of the IEEE 22nd International Symposium On Modeling, Analysis And Simulation Of Computer And Telecommunication Systems*, 2014.

[8] S. Borkar. Thousand core chips: A technology perspective. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 746–749. ACM, 2007.

[9] A. Chandrakasan, S. Sheng, and R. Brodersen. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473 –484, 1992.

[10] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen. Analysis and architecture design of an hdtv720p 30 frames/s h. 264/avc encoder. *Circuits and Systems for Video Technology, IEEE Trans. on*, pages 673–688, 2006.

[11] C. M. Don Darling and B. Singh. Gstreamer on texas instruments OMAP35x processors. *Proceedings of the Ottawa Linux Symposium*, pages 69–78, 2009.

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

[13] D. Kiliçarslan, C. G. Gürler, Ö. Özkasap, and A. M. Tekalp. Energy efficient video decoding on multi-core devices. In *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, pages 63–66. ACM, 2011.

[14] N. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, 2003.

[15] C. Meenderinck, A. Azevedo, M. Alvarez, B. Juurlink, and A. Ramirez. Parallel scalability of h. 264. In *Proceedings of the first Workshop on Programmability Issues for Multi-Core Computers*, 2008.

[16] G. J. Smit, A. B. Kokkeler, P. T. Wolkotte, and M. D. van de Burgwal. Multi-core architectures and streaming applications. In *Proceedings of the 2008 international workshop on System level interconnect prediction*, SLIP '08, pages 35–42. ACM, 2008.

[17] Y.-H. Wei, C.-Y. Yang, T.-W. Kuo, S.-H. Hung, and Y.-H. Chu. Energy-efficient real-time scheduling of multimedia tasks on multi-core processors. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 258–262. ACM, 2010.

[18] 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, 2010.

[19] S. Zhu, Z. Yu, S. Cui, Z. Yu, and X. Zeng. H.264 video parallel decoder on a 24-core processor. In *ASIC (ASICON), 2013 IEEE 10th International Conference on*, pages 1–4, 2013.