DyPS: Dynamic Processor Switching for Energy-Aware Video Decoding on Multi-core SoCs

Yahia Benmoussa
Univ. M'hamed Bougara, LMSS, Boumerdes, Algeria
Univ. Bretagne Occidentale, UMR6285, Lab-STICC, F29200 Brest, France
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

Yassine Hadjadj-Aoul
IRISA, Université de Rennes1
yassine.hadjadj-aoul@irisa.fr

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

ABSTRACT

In addition to General Purpose Processors (GPP), Multi-core SoCs equipping modern mobile devices contain specialized Digital Signal Processor designed with the aim to provide better performance and low energy consumption properties. However, the experimental measurements we have achieved revealed that system overhead, in case of DSP video decoding, causes drastic performances drop and energy efficiency as compared to the GPP decoding. This paper describes DyPS, a new approach for energy-aware processor switching (GPP or DSP) according to the video quality. We show the pertinence of our solution in the context of adaptive video decoding and describe an implementation on an embedded Linux operating system with the help of the GStreamer framework. A simple case study showed that DyPS achieves 30% energy saving while sustaining the decoding performance.

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

Adaptive Video Decoding, Energy, ARM, DSP, GStreamer, Embedded Linux.

1. INTRODUCTION

Nowadays, mobile devices such as smart-phones and tablets include more and more powerful hardware. For example, a hardware configuration including a processor clocked at more than 1 GHz frequency becomes common. However the use of high frequencies requires higher voltage levels and leads to an increase in energy consumption due to the quadratic relation between the dynamic power and the supplied voltage in CMOS circuits [6]. In a context where Lithium battery technologies are not evolving fast enough, the autonomy duration of those devices becomes a very critical issue [4] especially when using processor intensive applications such as video playback. In [7], it is shown that video playback is the most important energy intensive mobile application. This is due to the important use of the processing resources responsible of more than 60% of the consumed energy.

To overcome this issue, Digital Signal Processors (DSP) are a solution used to provide better performance-energy properties. Indeed, the use of parallelism in data processing increases the performance without requiring higher voltages and frequencies [16]. This makes them an energy-efficient choice in energy constrained devices [23] such as smart-phones and tablets where they are integrated in multi-core SoCs in addition to GPP [22].

When decoding a video stream, the use of the full processing capabilities of the hardware is not always necessary. For example, due to bandwidth limitation, the video may be coded in a low quality which leads to less decoding processing requirements [10]. In this case, in order to save energy, one might use dynamic voltage and frequency scaling feature provided by some low-power processors. This mechanism is used to scale down the voltage and the frequency in case of low processing workloads [17].

In addition to the above-stated energy considerations, the operation system overhead is an important parameter to consider especially in case of DSP video coding. In fact, the inter-processor communication generates a system overhead resulting from cache memory coherency maintenance, parameters passing, and I/O latency. This overhead is not negligible in case of decoding a low quality video requiring

EWiLi'13, August 26-27, 2013, Toulouse, FRANCE.

less processing power. In such case, as confirmed by experimental performance and energy consumption measurements [5], it was shown that a GPP video decoding can be the best choice in many cases as compared to the DSP decoding.

Accordingly, we propose in this paper an implementation of an energy-aware dynamic processing resources selection technique. This new approach allows a transparent processor switching (DSP/GPP) on a multi-core SoCs including a GPP and a DSP in a context of adaptive decoding of different video qualities.

The remainder of this paper is organized as follows : In section 2, the problem statement and context are given. In sections 3, the proposed solution is described. The implementation details, experimental evaluation and results are discussed in section 4 and 5 respectively. Related works on energy consideration of video decoding on Multi-core SoCs are discussed in section 6. Finally, conclusions and some future work perspectives are given in Section 7.

2. PROBLEM STATEMENT AND CONTEXT

2.1 Problem Statement

When decoding a video stream using a DSP, the inter-processor communication may generate a system overhead, and thus additional energy consumption [9, 2]. As an illustration, Fig. 1 describes the steps of a typical DSP video decoding process controlled by a GPP. The video frames are supposed in an input buffer in the memory.

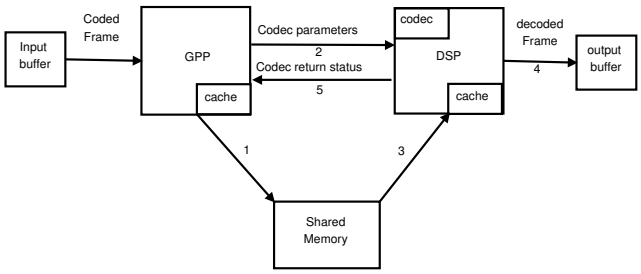

Figure 1: DSP video decoding

1. The GPP writes-back the frame located in its cache (as the frame may be located in the cache) to a shared memory so that the DSP can access it.
2. The GPP sends the parameters to the DSP codec via a GPP/DSP hardware bus.
3. The DSP invalidates the entries in its cache corresponding to a frame buffer in the shared memory.
4. The DSP decodes and transfers the frame to the output buffer.
5. The DSP sends the return status to the GPP.

In fact, both DSP and GPP have their proper cache memory and communicate using a shared memory. This imposes to manage cache coherency each time the DSP shares a data with the GPP. In addition, from the operating system level, The GPP/DSP communication is managed by a driver. A frame (a compressed picture) decoding is considered, from the GPP point of view, as an I/O operation generating a system latency caused by entering the *idle* state and handling of hardware interrupt. In addition, the GPP/DSP data transfers are generally achieved using Direct Memory Access (DMA) which offloads the processor from memory data transfers tasks but induces additional I/O and interrupt latency.

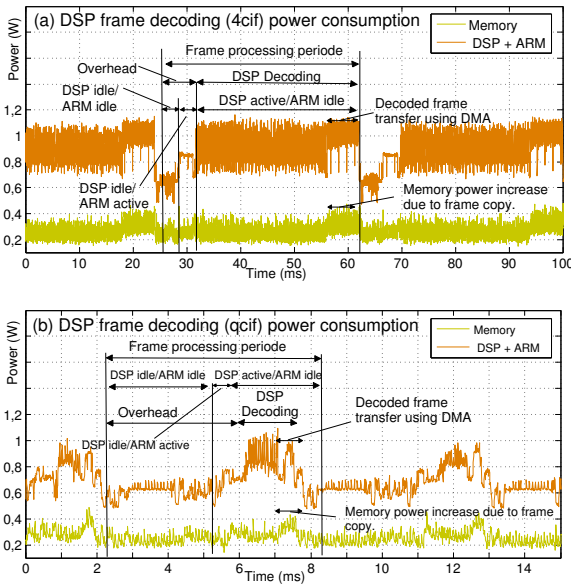

Figure 2: ARM and DSP frame decoding

A frame based DSP decoding analysis showed that the system overhead cannot be neglected in case of low video quality DSP decoding [5]. For example, Figure 2 shows the measured power consumption during the execution of the above steps on an OMAP3530 SoC containing a Cortex A8 ARM processor and a TMS320C64X DSP. Two video qualities are used : *4cif* (704x576) and qcif (176x144) resolution with 4 Mb/s and 256 Kb/s bit-rate respectively. The ARM processor and the DSP are clocked at 720 MHz and 520 MHz frequency respectively.

In Figure 2, the DSP frame decoding phase is represented by the strip varying between 0.7 W and 1.1 W corresponding to [32 ms, 62ms] and [6.2 ms, 7.5ms] intervals (*4cif* and *qcif* respectively). This phase is terminated by a burst of DMA transfers of the decoded frame macro-blocks from the DSP cache to the shared memory. This phase corresponds to the intervals [56 ms, 62 ms] and [7.2 ms, 7.5 ms] and is illustrated by an increase in memory power consumption. When the DSP terminates the frame decoding, it returns to the GPP (ARM Cortex A8) the execution status and enters the *idle* state. This event occurs, for example in Fig. 2-a (*4cif*) at 25 ms. The ARM wakeup latency is represented by the power level 0.66 W. The ARM wakeup event is represented by the power transition from 0.64 W to 0.85 W level.

A deeper performance and energy measurement showed that the processing which is not related to frame decoding (system overhead) represents 50% of the total processing time and 30% of the consumed energy in case of *qcif* resolution. This is not negligible and may have an impact on the overall performance and energy properties of video decoding we discuss hereafter.

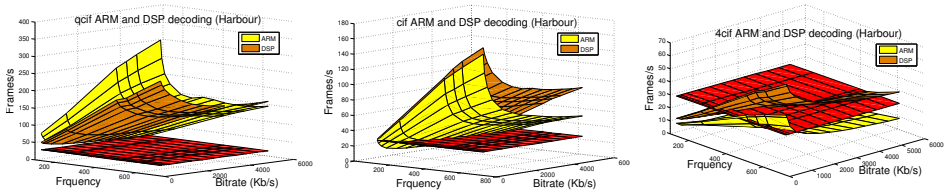

Figure 3: ARM *vs* DSP video decoding performance

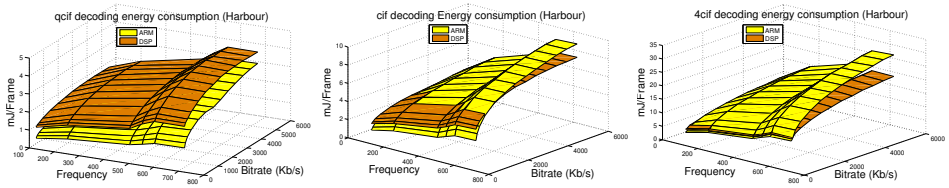

Figure 4: ARM vs DSP decoding energy consumption of H264/AVC video

2.1.1 Performances Impact

Figure 3 shows, a comparison between the measured performance (Decoded Frames/s) of GPP and DSP video decoding (Harbor sequence/30 Hz) according to the video bit-rate, resolution and clock frequency. The flat surface is the reference decoding speed corresponding to the display rate (30 Hz). It appears clearly that the DSP performances drops as compared to ARM decoding in case of *qcif* resolution.

2.1.2 Energy consumption Impact

Figure 4 shows a comparison between the energy consumption (mJ/Frame) of ARM and DSP video decoding for the same sequence according to the video bit-rate, resolution and processor frequency. One can observe that in case of *qcif* and low bit-rate *cif* resolution (352x288), the ARM video decoding is more energy-efficient than the DSP.

More information on performance and energy behavior of DSP decoding in term of video quality can be found in [5]. In the next section, we discuss the opportunity to exploit these results in a adaptive video decoding context.

2.2 Context

In mobile devices, a video content may be accessed using heterogeneous networks. Figure 5 illustrates an examples of some network technologies and their bandwidth capabilities which range from tens of Kbits to tens of Mbits per seconds. Consequently, the video quality should vary when the mobile device roams from a network to another. In addition, the bandwidth may fluctuate within the same network due to network congestion. In this context, novel streaming and video coding standards [21, 19] are designed to adapt dynamically the video content quality to fit with the available networks bandwidth. Based on the aforementioned observations, one can leverage such dynamic video adaptation by selecting the best suitable processing resource (GPP or DSP) available on multi-core SoCs for a better energy-efficiency.

3. PROPOSED SOLUTION

We designed here a solution that consists in selecting the best processing resource in the context of adaptive video decoding on multi-core SoC containing a GPP and DSP.

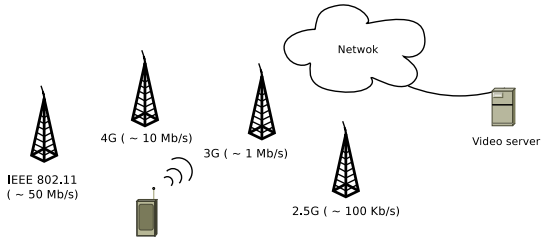

Figure 5: Video delivery via heterogeneous networks

3.1 Scope

Two video quality adaptation approaches exist: The scalable video coding [19] and adaptive streaming [21]. In the first one, a video content is coded in a self-contained file containing multiple layers : a base video quality layer and multiple enhancement layers. The enhancement layer consists of incremental data which allow to obtain a higher video quality starting from the base layer. In this case, there is a dependency between the different video qualities.

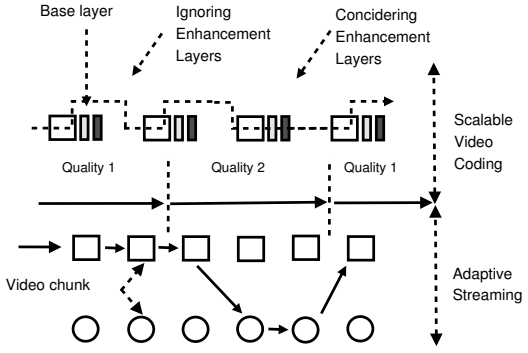

Figure 6: Scalable video coding and adaptive streaming

On the other hand, in adaptive streaming, a single video content is coded in independent streams having different qualities. each stream is divided into smaller units (video

sequences of few seconds) called chunks. The quality adaptation is achieved at a chunk granularity. Each chunk is decoded independently from the other chunks. Figure 6 illustrates these two approaches.

3.2 Adaptation logic

Based on the observations discussed in section 2, we propose to implement a processor switching policy according to the video resolution. The implemented video player switches to ARM video decoding in case of low video resolutions (*qcif*) and to DSP decoding for higher resolution (*cif*/*4cif*). By doing so, the video decoder would select the best processing resources achieving the decoding task using the least energy as illustrated in Figure 7.

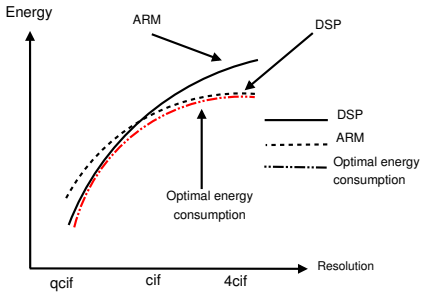

Figure 7: Optimal energy consumption according to resolution

We choose to implement such logic in case of dynamic adaptive streaming scenario because: 1) it is more used in real life as compared to scalable coding, 2) As discussed in section 3.1, the different video qualities are independent from each other (unlike scalable coding) which allows to one video chunk to be decoded in a GPP and the next one on the DSP without worrying about any dependency issue.

In what remains, we discuss the implementation details of the proposed dynamic processor switching technique in case of adaptive video streaming regardless of any existing video streaming technology and standard [21, 1, 3]. This technique is implemented on an embedded Linux operating systems using the *GStreamer* multimedia framework.

4. DYPS DESIGN AND IMPLEMENTATION

4.1 Hardware Setup

DyPS (Dynamic Processor Switching) was implemented on the OMAP3530 EVM board containing the low-power OMAP3530 *SoC* consisting of a Cortex A8 ARM processor and TMS320C64X DSP. The power consumptions of the DSP and the ARM processors are measured using the OpenPEOPLE framework [20], 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 rate.

4.2 Software Setup

On this hardware platform, the Linux operating system version 2.6.32 was used. The H264/AVC video decoding was achieved using GStreamer [8], a multimedia development framework. The ARM decoding, was performed using *ffdec_h264*, a plug-in based on the widely used *ffmpeg/libavcodec* library compiled with the support of NEON SIMD

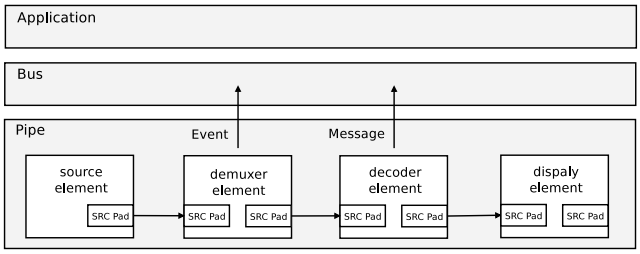

Figure 8: GStreamer framework

instructions set. For DSP decoding, we used *TIViddec2*, a proprietary H.264/AVC baseline profile plug-in provided by *Texas Instrument*.

GStreamer is a framework for creating streaming multimedia applications. It has a modular design based on plug-ins that provide various codecs and other functionalities. In *GStreamer*, an *Element* is the most important object. It has one specific function, which can be, for example, the reading of data from a file, decoding of this data or witting it to a display device. Many elements can be linked together to form a *Pipe* and let data flow through this chain. As illustrated in Figure 8, In a typical decoding chain, the Elements are connected thanks to *Pads* which are used to negotiate links and data flow between them. Data flows out of one element through one or more source *Pads*, and elements accept incoming data through one or more sink *Pads*. The types of these data are described as a *GstCaps* (Capabilities).

In addition, GStreamer provides powerful communication and synchronization mechanisms. Thus, different *Elements* can exchange various types of messages through a *Bus* and rise *Events* which can be handled synchronously by a dedicated handler. The above-described GStreamer features make it suitable to implement our proposed solution since we have to deal with dynamic events related to video quality adaptation decoding using two types of codecs targeting a ARM and DSP processor.

4.3 DyPS design

In the proposed solution, we reproduce a typical adaptive streaming scenario where a video content is coded in different qualities and divided into small chunks. Each chunk is coded using a video compression standard (in our case, H.264/AVC) and contained in an MP4 file format. Thus, a complete decoding pipe consists of the following elements:

- *filesrc* : for reading the video file.
- *qtdemux* : for extracting the video content (H.264/AVC coded data) from the MP4 files.
- *ffdec_h264* or *TIViddec2* : for decoding the coded H.264/AVC data using the ARM processor or the DSP.
- *xvimagesink* : for displaying the video content.

In DyPS, we suppose that different chunks are contained in one MP4 file. As illustrated in Fig. 9 : a video file is read using *filesrc* element (1) then its video content is extracted (2) using *qtdemux* demuxer. This element rises a "new pad" event (3) when it detects a new video chunk. According to the Pad type (in our case, the video resolution), a dedicated event handler plugs dynamically (4) the demuxer to

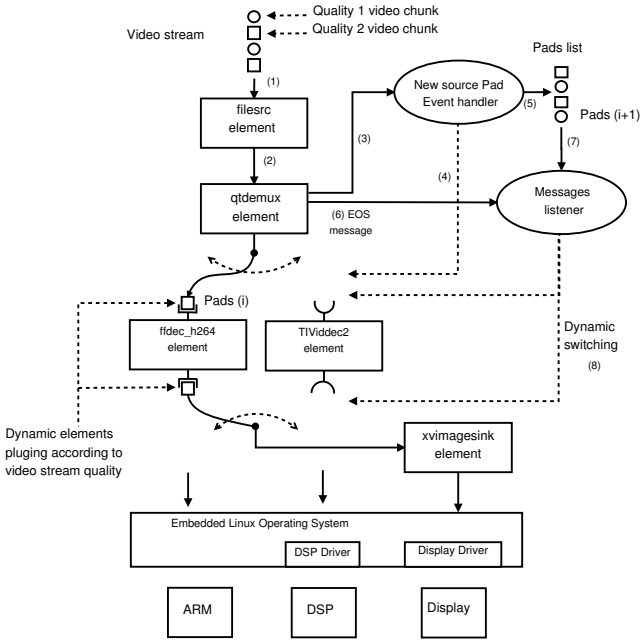

Figure 9: DyPS: Dynamic processor switching solution design using GStreamer

the *ffdec_h264* or *TIViddec2* decoder element. The next detected pads are queued in a list (5). When a video chunk is totally played, an "End Of Stream" (EOS) message is sent via the communication bus (6). Each time an EOS is sent, a message listener treats it by retrieving a pad from the list (7) and plugs it to a decoder element (ARM or DSP decoder) according to the video quality (8). The processor switching is achieved at this step. The selected decoder is then connected to the *xvimagesink* display element. All these functionalities are controlled from the application using an API provided by the *GStreamer* framework.

5. CASE STUDY

As discussed in section 3.2, we have configured DyPS to play *qcif* videos on ARM processor and higher resolution on DSP processor. 10 seconds H.264/AVC Harbor video chunks was grouped in a MP4 file as illustrated in Figure 10.

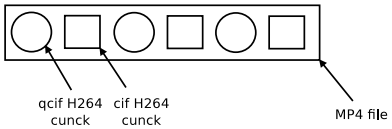

Figure 10: Video chunks in MP4 file

The player (with DyPS plugged) achieved a transparent processor switching according to the decoded video resolution. Figure 11 shows the power consumption plots resulting from decoding consecutive (*cif*, 4Mb/s) and (*qcif*, 512 Kb/s) chunks when disabling (Fig. 11-a) and enabling (Fig. 11-b) the processor switching. One can observe that switching to ARM decoding in case of *qcif* resolution allows to reduce to power consumption. A 30% energy saving is achieved in this example as compared to the DSP decoding when using DyPS without impact on the performances.

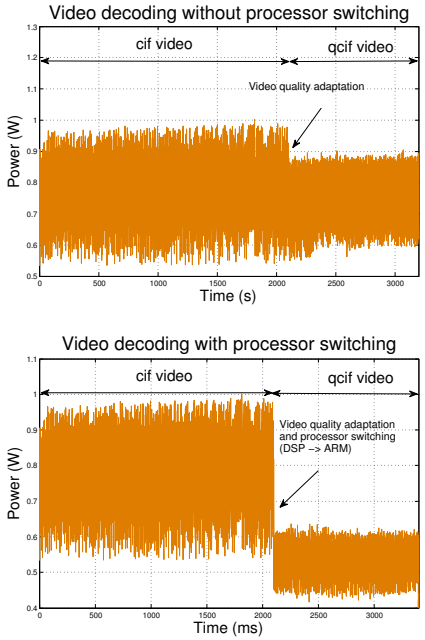

Figure 11: Dynamic processor switching impact on video decoding power consumption

6. RELATED WORKS

Video decoding performance and energy consumption optimization issue has been addressed by many works. In [16], the performance and energy consideration when using pipelines and parallelism in CMOS circuits is studied and it was shown why these architectures increase the energy efficiency. In [24], the particular case of H.264 video decoding is analyzed and an energy-aware architecture design methodology is proposed for energy efficient H.264/AVC decoding. At this level, the impact on the energy consumption of application and operating system layers are not considered.

At a higher level, In [10], H.264/AVC decoding performance is characterized on different GPP processor architectures at CPU cycle level. This approach is used in [15] for energy characterization and modeling of the different H.264/AVC decoder modules. The results were used to develop an energy-aware video decoding strategy for ARM processor supporting DVFS feature. The result of this study are generalized in [14] for considering the variation in video bit-rate. These studies was focusing only on GPP processor.

Many works studied the performance and energy consumption of DSP video decoding. In [18, 13, 9], performance consideration of DSP decoding are analyzed especially regarding cache coherency maintenance and DMA transfers. In [12], energy characterization of DSP processing is addressed in terms of memory access and DMA transfers. In [11], DSP video decoding energy consumption is analyzed in terms of different video coding qualities. Many of these studies highlight the performance and the energy efficiency of the DSP video decoding.

In this work a combined GPP/DSP decoding technique in a context of adaptive video decoding was proposed width the objective to save energy. As far as we know, no study proposed before such an approach.

7. CONCLUSION

In this paper, we described DyPS: a new technique for energy-aware dynamic processor switching based on energy consumption properties of GPP and DSP when decoding video. We have shown the benefit of using such a technique for energy saving in a context of dynamic video streaming. The feasibility of this technique was demonstrated by implementing it on the flexible and powerful GStreamer multimedia frameworks on embedded Linux platform. The processor switching criteria was based on the video resolution. This can be generalized to the video bit-rate. DyPS was validated on one hardware platform and a more thorough validation is to be performed on other platforms by characterizing the performance and energy consumption of GPPs and DSPs in order to confirm the optimal energy consumption configuration (see Figure 7).

A more elaborate policy can be envisaged for driving the processor switching technique. For example, the overall system load is an important criteria to be considered. In fact, even ARM decoding is more energy efficient for some video qualities, the use of the DSP offloads the ARM processor and lets the operating system or other application execute tasks without impacting the video playback. Thus, we can suggest to schedule a systematic DSP video decoding starting from a given system load threshold.

This work is a proof of concept and as a future works, we plan to implement this technique in a real adaptive streaming client and to extend the switching criteria to the bit-rate and the system load.

8. REFERENCES

[1] Apple HTTP live streaming: http://tools.ietf.org/id/draft-pantos-http-live-streaming-04.txt.

[2] Codec engine overhead, [online]. available: http://processors.wiki.ti.com/index.php/codec_engine_verhead.

[3] Microsoft smooth streaming: http://go.microsoft.com/?linkid=9682896.

[4] M. B and G. Archdale. Li-ion batteries and portable power source prospects for the next 5-10 years. *Journal of Power Sources*, 136(2):386–394, Oct. 2004.

[5] 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.

[6] T. Burd and R. Brodersen. Energy efficient CMOS microprocessor design. *System Sciences, Proceedings of the Twenty-Eighth Hawaii International Conference on*, 1:288–297, 1995.

[7] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, pages 21–21, 2010.

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

[9] J. Golston, S. Arora, and R. Reddy. Optimized video decoder architecture for TMS320C64x dsp generation. *Proc. SPIE 5022, Image and Video Communications and Processing*, pages 719–726, 2003.

[10] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/AVC baseline profile decoder complexity analysis. *Circuits and Systems for Video Technology, IEEE Trans on*, 13(7):704–716, 2003.

[11] E. Juarez, F. Pescador, P. J. Lobo, A. Groba, and C. Sanz. Distortion-energy analysis of an OMAP-Based H.264/SVC decoder. *Mobile Multimedia Communications*, (77):544–559, Jan. 2012.

[12] N. Julien, J. Laurent, E. Senn, and E. Martin. Power consumption modeling and characterization of the TI C6201. *IEEE Micro*, 23(5):40–49, Sept. 2003.

[13] S. Kant, U. Mithun, and P. S. S. B. K. Gupta. Real time H.264 video encoder implementation on a programmable dsp processor for videophone applications. *Consumer Electronics, 2006. ICCE '06. 2006 Digest of Technical Papers. International Conference on*, pages 93–94, 2006.

[14] X. Li, Z. Ma, and F. Fernandes. Modeling power consumption for video decoding on mobile platform and its application to power-rate constrained streaming. *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1 –6, 2012.

[15] Z. Ma, H. Hu, and Y. Wang. On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding. *IEEE Transactions on Multimedia*, 13(6):1240 –1255, Dec. 2011.

[16] D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, and R. Brodersen. Methods for true energy-performance optimization. *Solid-State Circuits, IEEE Journal of*, 39(8):1282–1293, 2004.

[17] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. *Proceedings of the 7th annual int conference on Mobile computing and networking*, pages 251–259, 2001.

[18] P. Ramachandra and M. R. Satish. H.264 main profile video decoding implementation techniques on OMAP3430IVA. *Signal Processing (ICSP), 2010 IEEE 10th Int Conference on*, pages 271–274, 2010.

[19] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9), Sept. 2007.

[20] E. Senn, D. Chillet, O. Zendra, C. Belleudy, S. Bilavarn, R. Atitallah, C. Samoyeau, and A. Fritsch. Open-people: Open power and energy optimization PLatform and estimator. *15th Euromicro Conference on Digital System Design (DSD)*, pages 668 –675, Sept. 2012.

[21] T. Stockhammer. Dynamic adaptive streaming over HTTP : standards and design principles. *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.

[22] C. H. K. Van Berkel. Multi-core for mobile phones. *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1260–1265, 2009.

[23] A. Wang and A. Chandrakasan. Energy-efficient DSPs for wireless sensor networks. *Signal Processing Magazine, IEEE*, 19(4):68–78, 2002.

[24] 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.