# Energy-aware feedback control for a H.264 video decoder

Sylvain Durand, Anne-Marie Alt, Daniel Simon, Nicolas Marchand

*International Journal of Systems Science*
Vol. 46, No. 8, 2015, 1432-1446

**RESEARCH ARTICLE**

# Energy-aware Feedback Control for a H.264 Video Decoder

Sylvain Durand[a], Anne-Marie Alt[a], Daniel Simon[a]* and Nicolas Marchand[b]

[a]INRIA Grenoble Rhône-Alpes, Inovallée, 38334 St Ismier Cedex, France;
[b]GIPSA-lab, 11 rue des Mathématiques BP 46, 38402 St Martin d'Hères, France

(*Received 00 Month 20xx; final version received 00 Month 20xx*)

Embedded devices using highly integrated chips must cope with conflicting constraints, while executing computationally demanding applications under limited energy storage. Automatic control and feedback loops appear to be an effective solution to simultaneously accommodate for performance uncertainties due to the tiny scale gates variability, varying and poorly predictable computing demands and limited energy storage constraints.

This paper presents the example of an embedded video decoder controlled by several feedback loops to carry out the trade-off between decoding quality and energy consumption, exploiting the frequency and voltage scaling capabilities of the chip. The inner loop controls the Dynamic Voltage and Frequency Scaling (DVFS) through a fast predictive control strategy. The outer loop computes the scheduling set-points needed by the inner loop to process frames decoding. The feedback loops have been implemented on a stock PC and experimental results are provided.

**Keywords:** Feedback scheduling, QoS, Energy-performance trade-off, Fast predictive control, H.264 video decoder

## 1. Framework and related works

The upcoming generations of embedded devices are integrating more and more multimedia and telecommunication applications, such as PDAs, mobile phones and tablets, thus requiring increasing on-board computing power. On the other hand, they become even more miniaturised (which means with limited energetic resources) while needing increased autonomy. These constraints are clearly conflicting, therefore technological evolutions are needed to improve their power consumption, computational efficiency and fabrication yield. In this paper, we propose the use of feedback control loops as a possible solution. The approach is applied to an embedded video decoding device whose requirements can be categorised in three main points :

**a) Energy saving requirements :** In current CMOS integrated chips, Dynamic Voltage and Frequency Scaling (DVFS) can be used to efficiently manage the energy consumption of a device. It results in simultaneously managing the frequency and the voltage (note that scaling down the voltage increases signals delays along the paths through electronic gates, thus needing to decrease its clock frequency as well, and inversely (Chandrakasan and Brodersen (1995), Flautner et al. (2004), Varma et al. (2003))). In many cases, the only performance requirement is that the tasks meet their deadline. Such cases create opportunities to run the processor at a lower computing level and achieve the same perceived performance while consuming less energy.A closed-loop DVFS approach is hence a good solution for energy saving. Classically, each task is considered independently, running at a constant voltage whose value is set to meet the deadline, and even selecting a small number of voltage levels leads to a drastic energy reduction (Ishihara and

*Corresponding author. Email: daniel.simon@inria.fr

ISSN: 0020-7721 print/ISSN 1464-5319 online

DOI: 10.1080/00207721.20xx.CATSid
http://www.informaworld.com

Yasuura (1998), Pouwelse et al. (2001)).

**b) Process variability requirements :** Systems on Chip (SoC) integrate extremely small scale CMOS manufacturing, e.g. silicon foundries currently target 32 $nm$ or even smaller (22 $nm$) gates. A nasty consequence of this very high integration is the variability in the silicon process : although a circuit is designed to run at a nominal clock frequency, the fabricated chip may vary by far from the expected performance (Romanescu et al. (2007)). Moreover, some cores may behave differently inside the same chip (Fesquet and Zakaria (2009)). Actually, this phenomenon is one of the leading causes for chip failures and delayed schedules at a sub-micrometric scale. To take full benefit of the potentially available computing power, hence it is needed that each computing node (or cluster of nodes) can be driven up to its maximal clock frequency. A common solution is the use of a Globally Asynchronous Locally Synchronous (GALS) architecture, e.g. Ferringer et al. (2006) : by removing the globally distributed clock, such circuits provide a cost effective solution for large SoC design. In practice, several nodes sharing a common frequency domain are gathered in a cluster, and clusters working at different frequencies are linked via an Asynchronous Network on Chip (ANoC) (Thonnart et al. (2009)). A GALS architecture can mitigate the impact of process and temperature variations, because a globally asynchronous system does not require that a global common frequency is dictated by the longest path delay of the whole chip, i.e. the critical path. In this case, the clock in each frequency domain is only determined by the slowest path in its domain. Furthermore, GALS techniques allow each locally synchronous island to be set independently, making DVFS more convenient than with the standard synchronous approach (Zakaria et al. (2010)).

**c) Quality of Service (QoS) requirements :** Highly integrated chips are expected to be used in many computing intensive fields, like in the present case of multimedia applications. Thanks to the important embedded computational power, many functions that were traditionally hardwired can now be implemented in software. This is for example the case for the radio communication receiving system, where components like filters, mixers and codecs can now be implemented with increased flexibility inside the programmable components of a Software Defined Radio (SDR) system. The extra computing power can also be used to decode video streams with high definition quality. However, high computing power has drawbacks in terms of energy consumption, especially for the case of mobile devices with limited embedded energy stored in battery. Consequently, trade-offs between measures of the multimedia application quality, available on-board energy and desired time to battery exhaustion must be managed, and may be translated into a control problem formulation. Trading performance and resources is relevant of QoS problems, which may be viewed as the management of some complex quality measures, assumed to provide an image of the satisfaction of application requirements. Indeed, QoS problem statements have been already used for energy-aware multimedia applications (Chiang et al. (2008)), most often focusing on networking load and communication management rather than on computing itself. An approach stating video processing under computation power limitations as a QoS problem is reported in Wurst et al. (2005). The QoS is evaluated via end-users perception criteria and enables tuning decoding parameters such as picture quality, deadline misses and quality level switches. Scalable processing is provided using several quality levels for frame decoding, each one associated with a corresponding computing cost.

### *Interaction between control and computing*

Control and real-time computing are used together in embedded control systems since decades. However it has been recognised only quite recently (e.g., Cervin et al. (2002) and Lu et al. (2002)) that more integrated co-design approaches between control and computing are likely to enhance both the performance and the robustness of real-time control systems. In particular it has been shown that digital objects, such as real-time schedulers and cyclic control tasks, can be efficiently managed through feedback loops. Therefore, they inherit the robustness and adaptivity

properties brought by closed-loop control. Moreover, digital processes can often be modelled through simple dynamics, e.g. using fluid modelling, therefore leading to simple controllers with very small execution overheads (Hellerstein et al. (2004), Malrait et al. (2011)). When applied to video codecs, feedback control has been mainly used for bit-rate management at the encoder side to accommodate the bandwidth of the communication link between the encoder and the decoder. For example, Sun and Ahmad (2004) combines prediction and PID control to simultaneously handle buffer occupancy and picture quality for an MPEG-4 video encoder. In other works, e.g. as reported in Brites et al. (2008), feedback channels are exploited in the coding process itself to enhance the compression rate and speed. Adaptive streaming can be post-processed on-the-fly, e.g. using a PI bit-rate controller in the server side as in De Cicco et al. (2011) or a Markov Decision Process (MDP) based algorithm as in Xiang et al. (2012). On the decoder side, using feedback is mainly motivated by energy consumption considerations, therefore considering the voltage and working frequency as control variables. For example, in Pouwelse et al. (2001) a feed-forward/feedback heuristics on-the-fly selects a H.263 decoder clock frequency, based on frame length modelling. Following the ideas of previously cited works for the control of web servers (Lu et al. (2002)), a PI controller is used in Lu et al. (2003) to control a MPEG decoder's speed and storage buffers occupancy to reduce the power consumption while preserving some real-time guarantees (but only simulation results are provided). Finally, in the already cited Wurst et al. (2005) reference, the QoS specification of a MPEG-2 decoder is implemented on a single fixed frequency CPU, using different quality levels as control variables driven by a MDP based controller.

***Guidelines***

The work presented here uses two low-levels control loops. The first one is based on fast Model Predictive Control and controls the chip's DVFS and computing speed. The second one controls the scheduling of the video frames through a simple Proportional controller. This control architecture is expected to be supplemented by a high-level QoS loop, whose design is out of the scope of the present paper. Anyway, its framework is required for designing the two other loops.

The proposed control architecture is able to manage several clusters. However, even when considering a single CPU with constant processing power, the approach shows a very effective and flexible decoding adaptation capability. Such a closed-loop control scheme follows several steps. Control design first needs a definition of the control objectives and an analysis and modelling of the process to be controlled. Basically, control uses an error signal between the desired and the measured (or estimated) state or output of the system. The output signals, which are significant for control purpose, must be identified and the corresponding sensors implemented. Then various control algorithms can be used to cyclically compute commands to be applied to the process via actuators, which must also be identified and implemented.

Using a GALS approach offers an easy integration of different functional units. Notably, it allows to slow down some parts of the circuit for better energy savings since each part can easily have its own independent clock frequency and voltage. Hence, such an architecture appears as naturally enabling distributed power management systems as well as local DVFS. This is even better not only in terms of power and performance, but also in terms of accommodation against variability (Marculescu and Talpes (2005)). As a result, a feedback control loop can be used to adapt the voltage/frequency of each part in order to respect some real-time constraints of the application and the allocated energy budget. Relying on this low-level layer, another control loop is able to lighten the computing load of functionals unit by dealing with the QoS at the application level, coping with the limitation of processing power and/or channel of communication and with some constraints in energy consumption. The setup to control the embedded video decoding mechanism of our case study is based on such an architecture.

The paper is organised as follows. In section 2, the closed-loop multi-layer architecture for

video decoding under computing resource and energy consumption constraints is sketched. The different control loops are then detailed in sections 3 for the DVFS layer and 4 for the frames rate controller. Some experimental results are finally given in section 5 using a short movie decoding as a support example, and a discussion concludes the paper.

## 2. Feedback scheduling setup for video decoding

Control design needs sensors and actuators to respectively monitor and drive the controlled process. In the particular case of feedback control of computing systems, sensors are provided by software probes used to build on-line indicators carrying out the processing activity. Actuators are provided by function calls, parameters tuning, or processing interruption/resume under control of an Operating System (OS). As the bitstream decoding quality is the object of control, models of the decoder quality (i.e. the control related model) as a function of various execution parameters (desired quality levels) are estimated from experiments. These models are further used to formalise the control objectives, e.g. using a QoS formulation. Besides data coming from the reported experiments, many ideas and assumptions are inspired by the work described in Wurst et al. (2005). Let first briefly introduce how works the decoding mechanism of the present study case before detailing the feedback multi-layer architecture.

### *2.1. Features of H.264/SVC*

H.264 is an international video coding standard, where a video sequence is made of three types of pictures : I pictures are reference pictures which are encoded independently of any other, P pictures are encoded using the previously encoded I picture, and B pictures are encoded using both the I and P previously encoded pictures. The order in which pictures are displayed is different from the order of the pictures encoding/decoding, hence there exist a systematic delay between decoding and display. For instance, if the display order is IBBBPBBBI then the encoding/decoding order will be IPBBBIBBB. The IPB pattern is defined at coding time and is invariant along all the video stream.

On the other hand, the Scalable Video Coding extension (SVC) has been defined to provide scalability capabilities in H.264 (Schwarz et al. (2007)), this is expected to provide the actuators needed for QoS control. Three types of scalability are allowed :

i) **Spatial scalability** enables to encode a video with several resolutions (i.e. number of pixels in a picture).
ii) **Temporal scalability** enables to encode all or a part of the frames of the original video with different rates.
iii) **Quality scalability** allows to encode the frames with several quantization steps which selectively cancels some information from the original video : its effect can be compared to a low-pass filter.

They can be combined to encode/decode a video stream. It is stated that the decoding process necessarily flows from lower to higher quality layers and, obviously, all the quality layers needed by the decoder must have been previously encoded by the decoder.

### *2.2. Control architecture*

The various control objectives and timing scales lead to define a hierarchy of three control loops to manage the decoding quality. They are depicted in Figure 1.

i) At high level, the **QoS controller** manages the application performance according to the

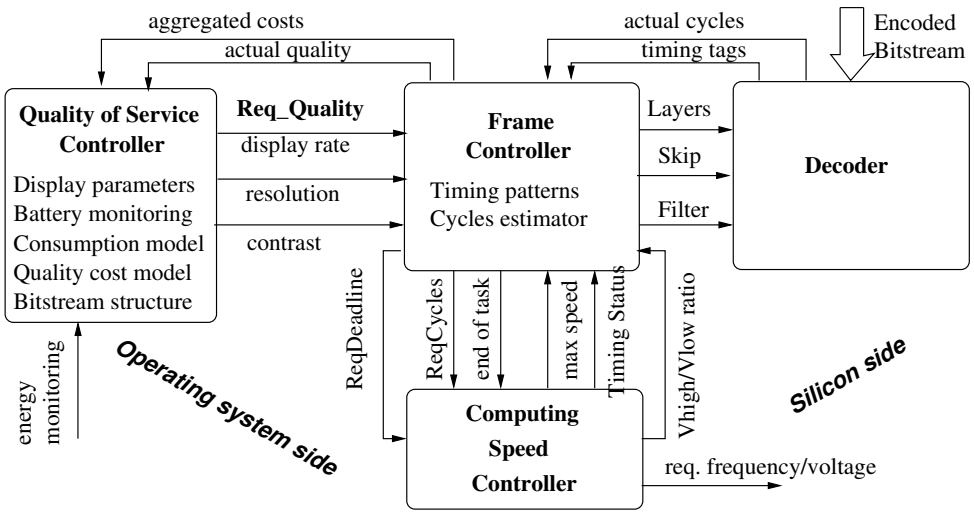



Figure 1. Proposed control architecture of an embedded video decoder.

available resources and end-user's requirements. Ideally, the goal of this controller is to maximise the quality of the displayed video stream under constraint of energy consumption. Hence, according to an available computing budget, the video must be decoded with the lower possible quantization step, higher resolution and maximal rate. The allowed computing budget itself depends on the available on-board energy storage and desired operating life. This high-level controller works with long term objectives with a time scale slow compared with the time scale of frame processing.

ii) At a lower level, decoding frames has basic deadlines related to the video display rate, typically $40\,ms$ for frames displayed at standard television rate. However, the decoding computing load is subject to fluctuations due to the varying content-dependent computation duty of the successive frames. Therefore an on-line adaptation of the decoding parameters (quality layers) can be associated with the frequency scaling capabilities of the cluster to meet the requested video rate. The **frame controller** works at the picture stream time scale and tightly cooperates with the speed controller integrated in each cluster.

iii) The **computing speed controller** manages the energy-performance trade-off in each cluster. It is integrated in the cluster's silicon, together with the voltage and frequency controllers which drive the power supply variables.

Whereas the design of the QoS control loop is only sketched in this paper, the two lower levels controllers are detailed from bottom to top in the following sections.

## 3. Control of the energy-performance trade-off

The computing speed controller aims at minimising the energy consumption while ensuring the frames computational inside given deadlines. This is possible by scaling the supply power taking into account some time constraints. In the present case, a closed-loop controller monitors the activity of the processor (its computational speed, shortly denoted speed in the sequel, e.g. in number of instructions per second) and adapts its voltage and frequency levels (afterwards denoted $V_{level}$ and $f_{level}$ respectively) with respect to a computational load. The suggested control strategy (see Durand and Marchand (2011) for further details) consists in dynamically calculating an energy-efficient speed set-point that the system will have to track in order to satisfy the control objective. This set-point is based on the number of instructions $\Omega_i$ and the

deadline $\Delta_i$ (both provided by the frame controller), to process task execution $T_i$ which decodes video frame $i$. $\Lambda_i$ denotes the laxity of $T_i$, i.e the remaining time to achieve the deadline of $T_i$.

We consider here a voltage scalable device with two possible voltage values $V_{high}$ and $V_{low}$. Let $\omega^{max}$ and $\omega_{max}$ respectively denote the maximal computational speeds when the system is running at $V_{high}$ and $V_{low}$ voltage. It follows that the high voltage level is necessary as soon as the average speed set-point of a task is higher than $\omega_{max}$ in order to avoid missing a given deadline. An intuitive method consists in building the average speed set-point of each task (that is the ratio $\Omega_i/\Delta_i$) in such a way that the number of instructions to do has been achieved at the end of the task (as depicted in Figure 2(a)). However, this method needs an infinite number of voltage and/or frequency levels (which is not possible in practice) and, anyway, is not energy-efficient since a whole task execution would be computed with the penalising high supply voltage, as highlighted in Figure 2(a) by $t_{V_{high}}$ for task execution $T_2$.

Nevertheless, a solution consists in splitting the task executions into two parts as represented in Figure 2(b). Firstly, the chip begins to run at high voltage (if required) with the maximal available frequency in order to achieve the maximal possible speed $\omega^{max}$, hence running faster than the average as for $T_2$ from time $t_2$ to $k$. Then, the execution finishes at low voltage at a speed lower than $\omega_{max}$ which, consequently, reduces the energy consumption. We propose to use only one possible frequency $F_{high}$ when running at $V_{high}$ whereas several frequency levels are possible at $V_{low}$ ($F_{low_1}$ and $F_{low_2}$ in the present study case with $F_{low_1} > F_{low_2}$). The degree of freedom on the frequency allows to approach as much as possible the deadline. A key point in the control strategy is that the switching time to go from $V_{high}$ to $V_{low}$ has to be suitably calculated to ensure a good computational performance. However, $k$ is not a priori known and therefore a predictive control law is used to dynamically calculate the switching time.

The presence of deadlines and input constraints to compute repetitive tasks naturally leads to a predictive control specification. Predictive control consists in finding a certain control profile over some time horizon to achieve a given control objective. The predictive issue can be formulated as a constrained optimisation problem : for each image $i$ to decode, find the computational speed set-point which minimises the high voltage running time $t_{V_{high}}$ while guaranteeing that the executed instruction number fits the number of instructions to do, that is

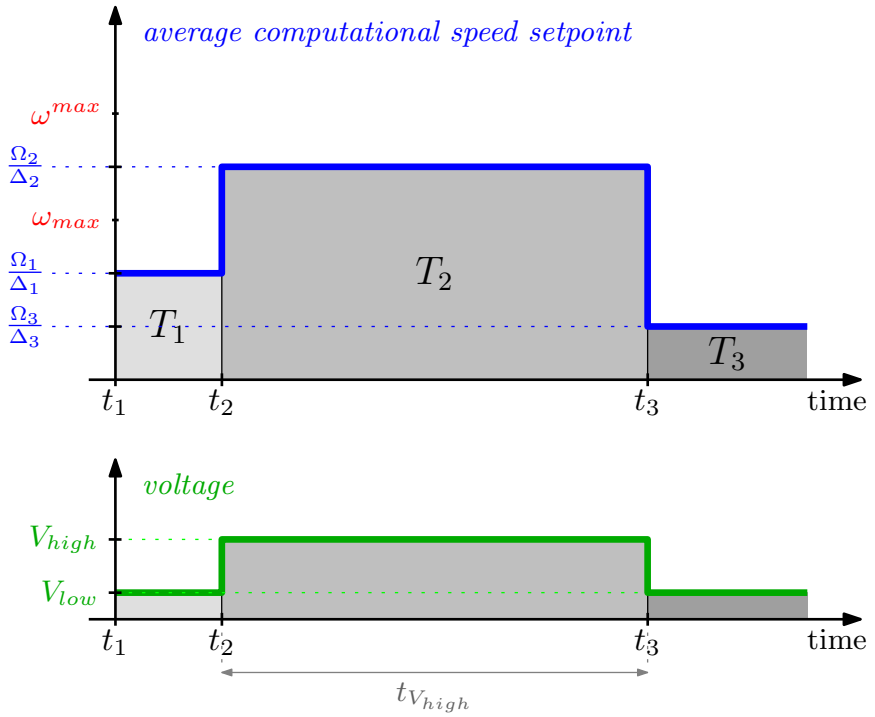
$$\min\ t_{V_{high}} \qquad \text{s.t.} \quad \int_{\Delta_i(t)} \omega(t)\ dt = \Omega_i \tag{1}$$

where $\int \omega dt$ corresponds to the executed number of instructions for the current image. In this particular case, the horizon is contractive : this means that the time to achieve the objective decreases with the laxity, hence the horizon is updated for each image to decode. However, this optimal criteria is associated to high computational cost, which is not acceptable in embedded systems with limited resources. The strategy adopted here is called fast predictive control and consists in taking advantage of the structure of the dynamical system to fasten the finding of the control profile (Alamir (2006)).

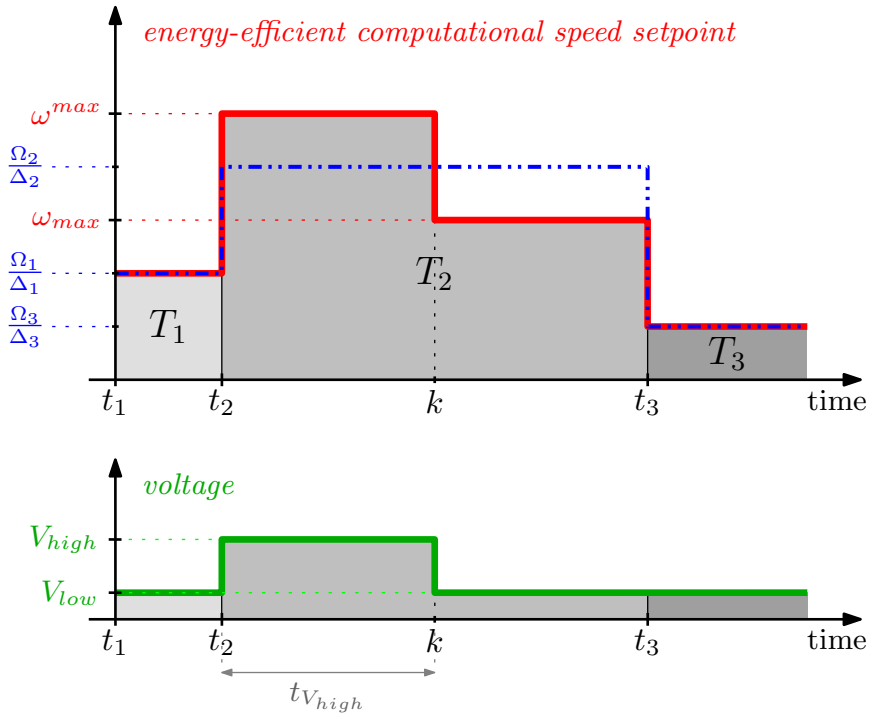
Indeed, the closed-loop solution yields an easier and faster algorithm since only two parameters need to be known, **i)** the computational load to be processed $\Omega_i$ and **ii)** the remaining time to achieve it $\Lambda_i$.

The speed required to meet the deadline (denoted the predicted speed $\upsilon$) is dynamically calculated at each sampling instant $k$ (where the sampling interval is much smaller than the decoding duration of an image)

$$\begin{aligned} \Omega(t_k) &= \Omega(t_{k-1}) + T_s\omega(t_k) \\ \upsilon(t_{k+1}) &= \frac{\Omega_i - \Omega(t_k)}{\Lambda_i(t_k)} \end{aligned} \tag{2}$$




(a) Intuitive average speed set-point.




(b) Energy-efficient speed set-point.

Figure 2. Different computational speed set-point buildings.

where $\Omega$ is the sum of the computational speed $\omega$ over past sampling instants, $T_s$ is the sampling period and $t_k$, $t_{k+1}$ and $t_{k-1}$ are the current, next and previous sampling times respectively. For sake of simplicity, $\Omega(\cdot)$, $\omega(\cdot)$ and $\upsilon(\cdot)$ are not indexed by the task execution number $i$. Note that here the processing load needed $\Omega_i$ for the task execution number $i$ is assumed constant during $T_i$ decoding, nevertheless it might be updated at any time during the execution of $T_i$ by the

frame controller for the case where decoding milestones can be identified and used on the fly.

The energy-efficient speed set-point is then directly deduced from the value of the predicted speed and so are the voltage and frequency levels. Indeed, the system has to run at $V_{high}$ and $F_{high}$ when the required speed is higher than the maximal speed at low voltage, otherwise the speed at low voltage is fast enough to finish to decode the image on time. Finally, the control decisions are

$$\left| \begin{array}{l} V_{level}(t_{k+1}) = V_{high} \\ f_{level}(t_{k+1}) = F_{high} \end{array} \right. \quad \text{if} \quad \upsilon(t_{k+1}) > \omega_{max}$$

$$\left| \begin{array}{l} V_{level}(t_{k+1}) = V_{low} \\ f_{level}(t_{k+1}) = F_{low_n} \end{array} \right. \quad \text{otherwise}$$

A frequency control strategy is needed at low voltage to determine $n$, but the principle remains the same. Also, a last control decision is possible "deactivating" the clock of the circuit. This is called the clock-gating principle (Kuzmicz et al. (2007)). In this case, the processor runs with the low voltage and a null frequency, which can be cost-effective when an image is decoded before its deadline. Furthermore, in order to be robust to manufacturing uncertainties, the maximal speeds $\omega^{max}$ and $\omega_{max}$ which are required in the control decision are estimated using a weighted average of the measured speed, details can be found in Durand and Marchand (2011).

The processing time $t_{V_{high}}$ at the highest voltage $V_{high}$ is minimum since the decoding of an image always starts with the penalising high voltage level (by construction of the predictive control law), ensuring that the low level will not be applied while the remaining computational load is large enough (higher than the maximum possible speed at $V_{low}$). The result is therefore structurally optimal. Furthermore, even if the voltage/frequency levels discretely vary, the set-point to track is always higher (or equal) than required by construction.

The stability of the control scheme follows the Lyapunov approach stating that, if an energy function of the system continuously decreases near an equilibrium point, then this point is a stable equilibrium for the system. Here we consider the decrease of the amount of instructions to be processed for $T_i$ as $x(t_k) = \Omega_i - \Omega(t_k)$. A classical Lyapunov function candidate would be of the form $V = x^T x$. $V$ is continuously decreasing as the processor's speed can be only positive, therefore ensuring the decoding control system stability.

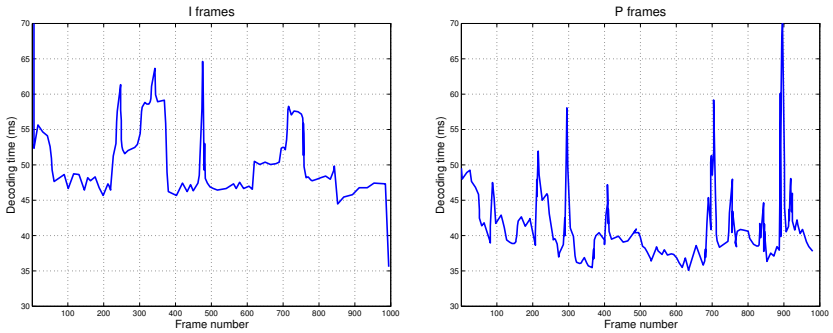
## 4. Frame control

The second control loop feeds the computing speed controller with estimations of the amount of computations to be performed (i.e. $\Omega_i$) within an associated deadline (i.e. $\Delta_i$) for each image $i$ to decode, as previously explained in section 3. This is developed in the sequel whereas a damping buffer is firstly introduced to store unprocessed data in case of missed deadlines.

### *4.0.1. Damping buffer provisioning*

Typically, deadlines in video decoding are associated with the video rate, e.g. $40\,ms$ are allocated to fully decode and display one image. However, even if the display video rate must be respected as far as possible, there are no strong synchronous constraints between the video source capture, encoding, decoding and display : latencies equivalent to several frames are allowed, hence there is room for scheduling flexibility at decoding time. Recall that the displayed order is different than the decoding order due to dependencies between images of different types I, P and B, as explained in subsection 2.1. Therefore, the displayed flow is inevitably delayed w.r.t. the incoming bitstream. Following the ideas in Wurst et al. (2005), an additional buffer is hence added to the frame decoding queue in such a way that decoding is performed several frames ahead of display. This added buffer is used to give space and accommodate for the vary-

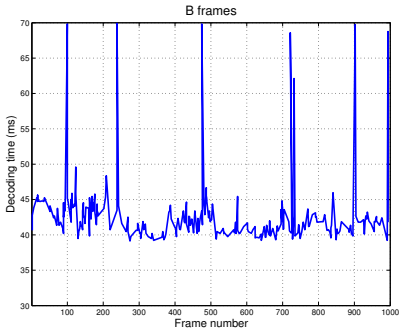
ing computing loads between frames. Measurements of decoding execution times were made to evaluate the profile and amplitude of computing load variations along a movie. Execution times measured from a 1000-frame long movie have been sorted according to the frame type (I, P or B) in Figure 3, where the bitstream has a unique layer with $624 \times 352$ pixel resolution and quantization step $Q_p = 28$ (note that larger is the quantization step, from 0 to 51, lower is the quality). This video sequence contains a mix of quiet and action plans. It can be observed, especially for the reference I frames, that the decoding times are almost constants for quite long intervals, with abrupt changes between flat areas, and isolated high values. The observation of constant intervals enables to estimate the number of cycles with sufficient accuracy. The maximal value for these isolated peaks suggests that a three-frame deep control buffer would be able to damp most of the computing load variations.




(a) Frame I.




(b) Frame P.




(c) Frame B.

Figure 3. Decoding times for I, P and B frames.

### *4.0.2. Deadline and computing load control*

The main goal of the frame controller is to provide the underlying computing speed controller with estimates of the number of cycles $\hat{\Omega}_{i+1}$ to be processed for the next frame within a requested deadline $\Delta_{r_{i+1}}$. According to Wurst et al. (2005) and to the measures plotted in Figure 3, an estimate of $\hat{\Omega}_{i+1}$ can be often taken as the last $\Omega_i$ reported by the computing speed controller, or by filtered values of the last executions, such as

$$\hat{\Omega}_{i+1} = \alpha\hat{\Omega}_{i-1} + (1-\alpha)\Omega_i \tag{3}$$

where $0 \leq \alpha \leq 1$ is a weight. Furthermore, considering a fixed ideal schedule $\{..., \tau_{i-1}, \tau_i, \tau_{i+1}, ...\}$, e.g. with equidistant intervals of $40\,ms$, a first basic feedback loop is aimed to regulate the requested deadlines $\Delta_{r_i}$ to their ideal value $\tau_i$. However, due to the rough prediction of the computing load $\hat{\Omega}_i$, the actual (measured) deadline deviates from its requested value, which yields $\Delta_i = \Delta_{r_i} + \delta_i$. Assuming that the computing load is almost constant, the computing

overshoot can then be driven to 0 if $\delta_{i+1} = (1-\beta)\delta_i$ with $0 < \beta \leq 1$, leading to the elementary deadline controller

$$\Delta_{i+1} = \tau_{i+1} + (1-\beta)\delta_i \tag{4}$$

The resulting control loop may accommodate for short term variations of the computing load for each frame. From a stability point of view, system (4) has no dynamics and (for a constant load $\Omega$) the successive values of $\delta$ results from a numerical suite which converges to 0 for $0 < \beta < 2$ (in practice $\beta \leq 1$ is chosen to avoid overshoots) : larger is $\beta$, faster is the decay of the deadline deviations $\delta$, and larger is the noise.

The capabilities of this controller can be exhausted in case of several successive peak loads able to overflow the three-frame ahead buffer. In that case, only the I frames will be fully decoded up-to the requested quality level, while the depending P and B frame decoding can be truncated up-to recovering enough buffer space. Thanks to the signals that are fed back by the decoder and by the computing speed controller, several control decisions can be considered, sorted by their expected increasing impact on the displayed quality :

i) Truncation of the decoding process at a quality layer lower than the set-point can be done at any point for B and P frames.
ii) A comparison between a reference decoding timing pattern and actual tags inserted in the decoder may help to anticipate overloads and abort useless steps rather than awaiting a deadline miss.
iii) In case of accumulated overload peaks running beyond nominal control actions, skipping or aborting a frame decoding may be taken as an emergency action, allowing to reset the decoding stack. This action must be as far as possible avoided especially for I frames.

## 5. Experimental results

In this last section, we implement and test in practice our proposal for decoding a short video.

### *5.1. Implementation*

A prototype of a H.264 decoding controller has been developed under the Linux operating system. It runs on a stock Linux kernel tuned for enhancing its real-time capabilities, i.e. compiled with the tick-less feature, high resolution timers and preemptive kernel options. In particular, these options ensure low latencies during task switching and allow for the very precise measurements of execution times (down to the hardware resolution) which are necessary to feed the feedback loops.

The tests have been executed on a standard Dell E6400 laptop, using a DuoCore2 processor. The test-bed uses the Symmetric Multi-Processing (SMP) capability of the kernel to allocate specified parts of the software to dedicated CPUs, e.g. the controllers run on one CPU and the decoder run on the other. To emulate the speed controller (which is assumed to be integrated in silicon in a future chip) the frequency scaling capabilities of the chipset have been used. At this end, the predictive controller of section 3 is implemented as a custom module of the cpufrequtils package, allowing to control the CPU frequency between 800 MHz and 2.535 GHz in several steps. On the other hand, the frame controller and decoder are encoded in Posix pthreads using the features of the Real-Time Posix library. The decoder itself is based on the reference implementation of the H.264 standard, namely the free and open source JSVM[1] software. This

[1] http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm

version is far to be optimised (compared with available versions such as the ffmpeg package) but it both implements all the features of the H.264/SVC and is open source, so that the source code can be easily instrumented to integrate the control features described in the previous sections. The counterpart w.r.t. optimised parallel versions is that only low resolution decoding can be achieve in real-time, anyway the set-up is able to demonstrate the efficiency of the approach.

### *5.2. Experimental results*

The following decoding experiments use a 1000-frame video encoded with a IBBBPBBBI structure. Due to the low performance of the JSVM decoder only low resolution ($624 \times 352$ pixels) is used. Two quality layers were considered, with or without post-processing filter. Indeed, an optional final processing is possible using a filter which, if applied, improves the final quality of the picture. The CPU frequency is controlled in three steps, with $F_1 = 2535\,\text{MHz}$, $F_2 = 1600\,\text{MHz}$ and $F_3 = 800\,\text{MHz}$. For this particular video sequence and CPU, the average decoding time for one frame is around $50\,ms$ at the highest quality (including filter). The decoder has been tested with various requested frame deadline values between 40 and $60\,ms$. Therefore, the decoding process is stressed for the shorter requested values and it is expected that the control strategy is able to keep a high decoding quality despite the lack of computing resources. Also, for the slower requested decoding rates it is expected that the control strategy allows to save CPU cycles and energy compared with an uncontrolled decoder.

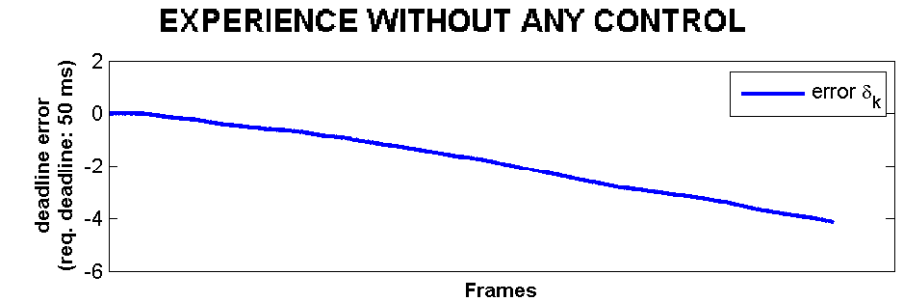
#### *5.2.1. Frame deadline control*

The first experimental results show the efficiency of the deadline controller while keeping the CPU frequency constant, using the extra three-frame buffer to damp the decoding overshoots with filtering gain $\alpha = 0.1$ and damping gain $\beta = 0.3$. Figure 4 plots the evolution of the deadline overshoot $\delta$ for a requested frame deadline of $50\,ms$. In Figure 4(a), this is done without any control. The plot exhibits a continuously growing drift in the decoding overshoot, as deadline errors accumulate with time. Conversely, using the deadline controller of subsection 4.0.2 allows to quickly absorb the deadline overshoots due to peaks in decoding time and to keep the real decoding schedule close to the theoretic one, as shown in Figure 4(b).

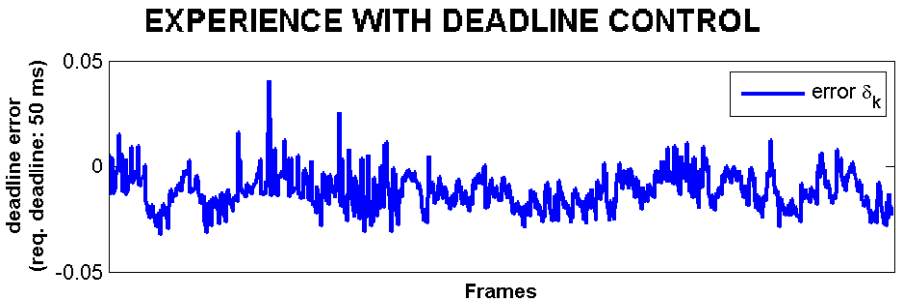
#### *5.2.2. Control over various requested deadlines*

The following plots report experiments where the frame decoding deadlines are fixed with requested values ranging from 40 to $65\,ms$, by steps of $5\,ms$. The expected frame deadlines are represented in green, the observed deadlines are plotted in red and the CPU frequency is highlighted in blue.

i) In Figure 5, no feedback controller is active and, consequently, the CPU frequency is always at the maximal value. This is clearly neither energy nor computing efficient since the highest frequency as well as decoding quality are always applied even for large requested deadlines.
ii) In Figure 6, the speed controller introduced in section 3 is active and the deadlines are fixed. For the shortest requested deadlines, the frequency is always maximal but, anyway, the deadlines cannot be reached due to the performance of the particular system used for these experiments (recall that the average decoding time for one frame is $50\,ms$). On the other hand, the decoding is finished on time for the larger ones and low frequency levels are used (which means the energy consumption can be reduced, as explained in the sequel) especially when the deadline constraint is weak.
iii) In Figure 7, the speed and frame controllers of section 4 are both active. In that case, the requested deadlines are always achieved thanks to the QoS control and damping buffer. Indeed, the final filter is activated only when possible (the filter is activated/deactivated




(a) Without any control.




(b) With deadline control.

Figure 4. Deadline errors when keeping the CPU frequency constant.

when the plot is respectively 1/0 in Figure 7). As expected, this filter is most often used for weak decoding constraints.

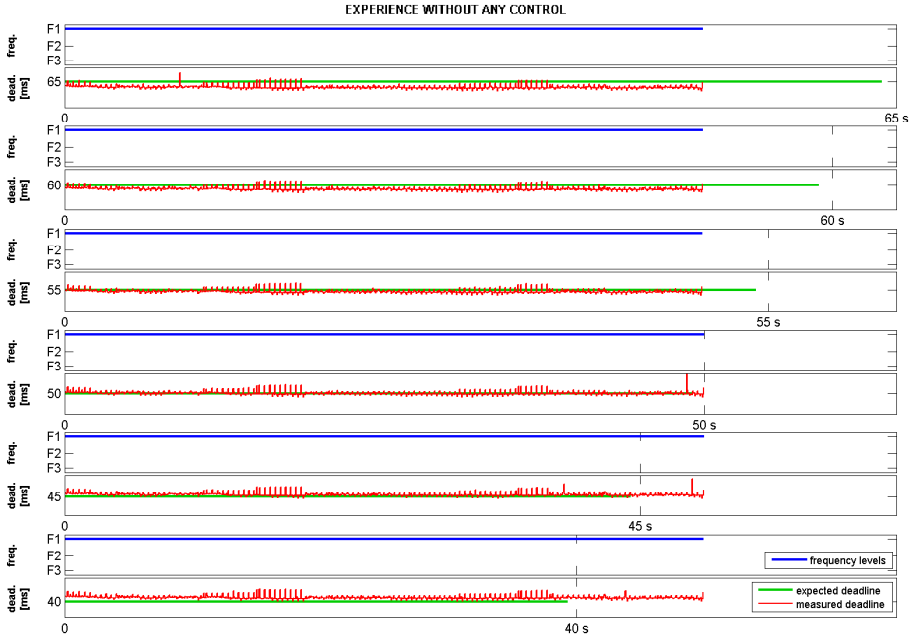



Figure 5. Control over various requested deadlines without any feedback controller.

### *5.2.3. Energy consumption*

The energy specifically spent for decoding could not be directly measured with the laptop used for the experiments, as it is not possible to spot out the energy cost of the CPUs from the

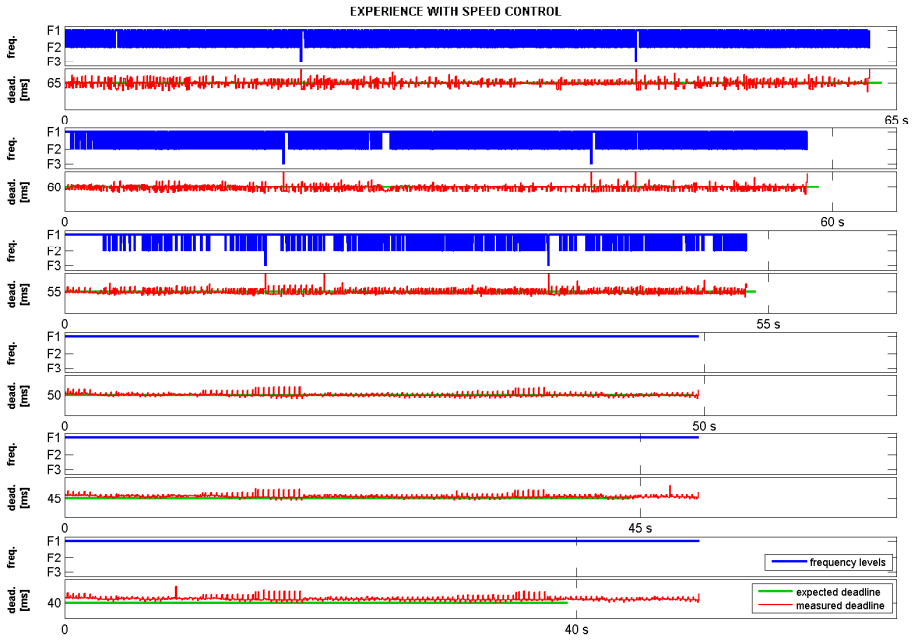



Figure 6. Control over various requested deadlines with only speed control (the deadlines are constant).


Figure 7. Control over various requested deadlines with speed and deadlines control (the computation load varies w.r.t. quantization and/or resolution layers).

energy spent by the peripherals (e.g. the GPU). Nevertheless, using the process knowledge given in section 3, energy cost estimations can be carried out from the physical models.

First, there are two voltage levels with only one frequency level possible when the system is running at the maximal one and two possible levels for the lower voltage. Secondly, the low voltage level is defined as the half of the higher one (for simplicity). As a result, the energy consumption of the previous scenarii can be estimated as follows :

- When running at 2535 MHz, the supply voltage is $V$.

- When running at 1600 MHz, the supply voltage is $V/2$.
- When running at 800 MHz, the supply voltage is $V/2$.

Moreover, it is assumed that the energy used for decoding is proportional to $f_{clk} \times V_{dd}^2$, as explained in Chandrakasan and Brodersen (1995). Thus, the estimated energy consumption for a video sequence where the average frame decoding time (with the highest quality) is 50 $ms$ can be summarised in Table 1 (normalised for each requested deadline with the consumption obtained without any control). These results call for several remarks :

i) For deadlines shorter than the average, the two controllers should be active. In particular, the frame deadline controller is able to toggle the quality level and avoids using permanently the highest frequency level, thus saving energy.
ii) For requested deadlines larger than the average, it is unlikely that a deadline can be missed and the speed controller alone is most often enough to decode on time the incoming frames. In that case, the frame deadline controller uselessly anticipates future overshoots and induces useless costly CPU cycles at high speed.

Table 1. Energy consumption estimation.

| **Deadline** | **Normalised energy consumption** | |
|---|---|---|
| | speed control only | speed and frame control |
| 40 | 1 | 0.76 |
| 45 | 1 | 0.81 |
| 50 | 1 | 0.86 |
| 55 | 0.92 | 1.01 |
| 60 | 0.78 | 0.91 |
| 65 | 0.65 | 0.77 |

### *5.2.4. Decoding quality degradation*

Finally, a penalty criterion based on video expertise (borrowed from Wurst et al. (2005)) has been applied to assess for the decoding quality degradation due to stressed decoding conditions. The following penalties are applied and accumulated along the decoding process of the test sequence :

- Skipping a frame : -10000
- Deadline achieved without filter : +5
- Deadline achieved with filter : +10
- Increasing quality level : -10
- Decreasing quality level : -50

The results are plot in Figure 8 only for the stressing deadlines (40, 45 and 50 $ms$), since longer deadlines are always achieved with no quality loss. Indeed, with the uncontrolled case, the decoding overshoots accumulate and finally induce quality level switches and even frame skips. The plot shows that using simple and cheap feedback controllers allows to increase the decoding quality while highly decreasing the computing and energetic costs.

## 6. Conclusion

In this paper a survey of problems related with very small scale integrated chips, running high computations demanding process on mobile devices, was first given. It appears that feedback control may provide solutions to fight against the silicon process variability, and to optimise a trade-off between the computation load and related energy cost in one hand and the application quality in the other hand.

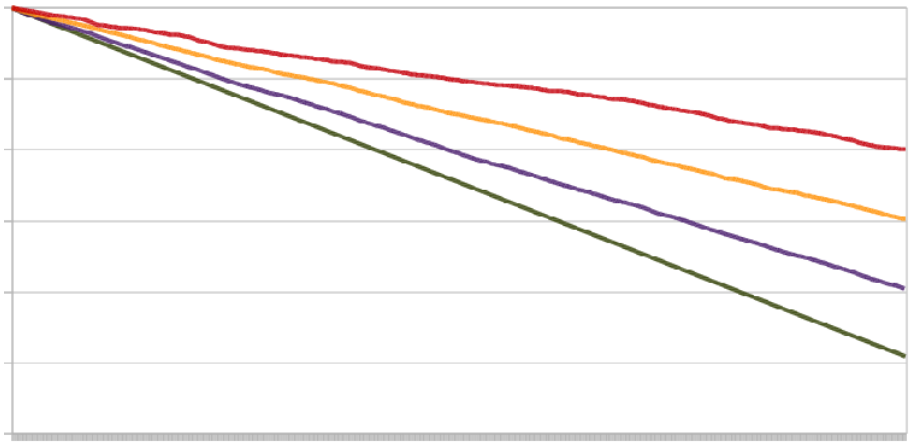


Figure 8. Decoding quality.

To support this statement a control architecture was proposed to target the particular case of an embedded video decoder. Several controllers are nested to meet the end-user's requirements in terms of quality of service and energy saving. The whole system finally runs minimising the energy consumption induced by computations while ensuring specified performances, which means that the video is decoded with a required resolution and rate while minimising the CPU consumption. An experimental validation of the proposal was set-up on a system using a single processor with different power modes for decoding. The results clearly show that up to 25 % energy saving can be achieved while keeping a specified video quality. As for other examples of integrated feedback control and computing systems (e.g. as in Lu et al. (2002) and Hellerstein et al. (2004)), it is shown again that feedback loops can easily enhance the performance and robustness of a computing system for very moderate programming efforts and execution overheads.

## Acknowledgements

This research has been supported by ARAVIS, a Minalogic project gathering STMicroelectronics with academic partners, namely TIMA and CEA-LETI for micro-electronics, INRIA/LIG for operating system and INRIA/Gipsa-lab for control. The aim of the project is to overcome the barrier of sub-scale technologies (45nm and smaller).

## References

Alamir, M., *Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes: A Parametrized Approach for Fast Systems*, Vol. 339, Lecture Notes in Control and Information Sciences, Springer-Verlag (2006).

Brites, C., Ascenso, J., Pedro, J.Q., and Pereira, F. (2008), "Evaluating a feedback channel based transform domain Wyner-Ziv video codec," *Signal Processing: Image Communication*, 23(4), 269–297.

Cervin, A., Eker, J., Bernhardsson, B., and Arzen, K.E. (2002), "Feedback-Feedforward Scheduling of Control Tasks," *Real-Time Systems*, 23(1–2), 25–53.

Chandrakasan, A.P., and Brodersen, R.W. (1995), "Minimizing Power Consumption in Digital CMOS Circuits," *Proceedings of the IEEE*, 83(4), 498–523.

Chiang, J.C., Lo, H.F., and T., L.W. (2008), "Scalable Video Coding of H.264/AVC Video

Streaming with QoS-based Active Dropping in 802.16e Networks," in *22nd Int. Conf. on Advanced Information Networking and Applications*, Okinawa, Japan, pp. 1450–1455.

De Cicco, L., Mascolo, S., and Palmisano, V. (2011), "Feedback Control for Adaptive Live Video Streaming," in *Multimedia Systems Conference MMSys 11*, San Jose, CA, USA: ACM, pp. 145 –156.

Durand, S., and Marchand, N. (2011), "Fully Discrete Control Scheme of the Energy-Performance Tradeoff in Embedded Electronic Devices," in *18th IFAC World Congress*, Milano, Italy, pp. 3298–3303.

Ferringer, M., Fuchs, G., Steininger, A., and Kempf, G. (2006), "VLSI Implementation of a Fault-Tolerant Distributed Clock Generation," in *21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'06)*, Arlington, USA, pp. 563–571.

Fesquet, L., and Zakaria, H. (2009), "Controlling Energy and Process Variability in System-on-Chips: Needs for Control Theory," in *3rd IEEE Multi-conference on Systems and Control - 18th IEEE International Conference on Control Applications*, Saint Petersburg, Russia, pp. 302–307.

Flautner, K., Flynn, D., Roberts, D., and Patel, D.I. (2004), "An Energy Efficient SoC with Dynamic Voltage Scaling," in *Conference on Design, Automation and Test in Europe DATE'04*, Vol. 3, Paris, France, pp. 30324–30327.

Hellerstein, J., Diao, Y., Parekh, S., and Tilbury, D., *Feedback Control of Computing Systems*, New York: Wiley-IEEE Press (2004).

Ishihara, T., and Yasuura, H. (1998), "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," in *International Symposium on Low Power Electronics and Design*, Monterey, CA, USA, pp. 197–202.

Kuzmicz, W., Piwowarska, E., Pfitzner, A., and Kasprowicz, D. (2007), "Static Power Consumption in Nano-CMOS Circuits : Physics and Modelling," in *14th International Conference Mixed Design of Integrated Circuits and Systems MIXDES'07*, Ciechocinek, Poland, pp. 163–168.

Lu, C., Stankovic, J., Son, S., and Tao, G. (2002), "Feedback Control Real-Time Scheduling: Framework, Modeling and Algorithms," *Real-Time Systems Journal*, 23(1/2), 85–126.

Lu, Z., Lach, J., Stan, M., and Skadron, K. (2003), "Reducing multimedia decode power using feedback control," in *21st International Conference on Computer Design ICCD'03*, San Jose, CA, USA, pp. 489–496.

Malrait, L., Bouchenak, S., and Marchand, N. (2011), "Experience with CONSER: A System for Server Control through Fluid Modeling," *IEEE Transactions on Computers*, 60(7), 951–963.

Marculescu, D., and Talpes, E. (2005), "Energy Awareness and Uncertainty in Microarchitecture-Level Design," *IEEE Micro*, 25, 64–76.

Pouwelse, J., Langendoen, K., Lagendijk, R., and Sips, H. (2001), "Power aware video decoding," in *22nd Picture Coding Symposium*, Seoul, Korea, pp. 303–306.

Pouwelse, J., Langendoen, K., and Sips, H. (2001), "Dynamic Voltage Scaling on a Low-Power Microprocessor," in *7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, pp. 251–259.

Romanescu, B.F., Bauer, M.E., Sorin, D.J., and Ozev, S. (2007), "Reducing the Impact of Process Variability with Prefetching and Criticality-Based Resource Allocation," in *16th International Conference on Parallel Architecture and Compilation Techniques*, Brasov, Romania, p. 424.

Schwarz, H., Marpe, D., and Wiegand, T. (2007), "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. on Circuits and Systems for Video Technology*, 17(9), 1103–1120.

Sun, Y., and Ahmad, I. (2004), "A Robust and Adaptive Rate Control Algorithm for Object-Based Video Coding," *IEEE Transactions on circuits and systems for video technology*, 14(10), 1167–1182.

Thonnart, Y., Beigne, E., and Vivet, P. (2009), "Design and Implementation of a GALS Adapter for ANoC Based Architectures," in *15th IEEE Symposium on Asynchronous Circuits and*

*Systems (ASYNC'09)*, Chapel Hill, USA, pp. 13–22.

Varma, A., Ganesh, B., Sen, M., Choudhury, S.R., Srinivasan, L., and Bruce, J. (2003), "A Control-Theoretic Approach to Dynamic Voltage Scheduling," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems CASES'03*, New York, USA, pp. 255–266.

Wurst, C.C., Steffens, L., Verhaegh, W.F., Bril, R.J., and Hentschel, C. (2005), "QoS Control Strategies for High-Quality Video Processing," *Real Time Systems*, 30(1), 7–29.

Xiang, S., Cai, L., and Pan, J. (2012), "Adaptive scalable video streaming in wireless networks," in *3rd Multimedia Systems Conference*, Chapel Hill, North Carolina, MMSys '12, New York, NY, USA: ACM, pp. 167–172.

Zakaria, H., Durand, S., Fesquet, L., and Marchand, N. (2010), "Integrated Asynchronous Regulation for Nanometric Technologies," in *First European Workshops on CMOS Variability VARI'10*, Montpellier, France, pp. 86–91.