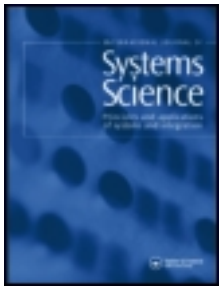

International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tsys20

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

Sylvain Durand [a], Anne-Marie Alt [a], Daniel Simon [a] & 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
Published online: 20 Aug 2013.

To cite this article: Sylvain Durand, Anne-Marie Alt, Daniel Simon & Nicolas Marchand (2013): Energy-aware feedback control for a H.264 video decoder, International Journal of Systems Science, DOI: 10.1080/00207721.2013.822607

To link to this article: http://dx.doi.org/10.1080/00207721.2013.822607

International Journal of Systems Science, 2013
http://dx.doi.org/10.1080/00207721.2013.822607

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 16 October 2012; final version received 8 June 2013*)

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 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. Introduction

The upcoming generations of embedded devices are integrating an increasing number of multimedia and telecommunication applications, such as PDAs, mobile phones and tablets, and require increasing on-board computing power. They become highly miniaturised with limited embedded energetic resources, while needing increased autonomy. These constraints are clearly conflicting, and technological evolutions are needed to improve their power consumption, computational efficiency and fabrication yield. The use of feedback control loops is expected to provide solutions, and such an approach is applied here for an embedded video decoding device.

Energy availability is one of the main limiting factors for mobile platforms powered by batteries. Power management techniques are used to minimise the energy consumption while reaching computation deadlines and overall application performance. Actually, the dynamic power consumption depends both on the clock frequency and on the supply voltage. As a consequence, the power consumption can be minimised with an appropriate setting of these parameters, using so-called dynamic voltage and frequency scaling (DVFS) methods. Note that scaling down the voltage increases signals delays along the paths through electronic gates, thus needing to decrease accordingly its clock frequency, and conversely (Chandrakasan and Brodersen 1995; Varma et al. 2003; Flautner, Flynn, Roberts, and Patel 2004). A closed-loop DVFS approach is hence a good solution for energy saving (Ishihara and Yasuura 1998; Pouwelse, Langendoen, and Sips 2001).

On the other hand, one of the leading causes for chip failures and delayed schedules in circuits at a submicrometric scale is the *process variability*. 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 actual reachable frequency may vary by far from the expected performance (Romanescu, Bauer, Sorin, and Ozev 2007). Moreover, the different cores of the same chip may behave differently (Fesquet and Zakaria 2009). Hence, to fully benefit from the potentially available computing power, it is needed that each computing node (or cluster of nodes) can be driven up to its maximal clock frequency. A solution is the use of a globally asynchronous locally synchronous (GALS) architecture, e.g. Ferringer, Fuchs, Steininger, and Kempf (2006), which additionally provides a cost-effective solution for large SoC design by removing the globally distributed clock circuitry. 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, Beigne, and Vivet 2009). In such a GALS architecture, the fastest clock in each frequency domain is only determined by the slowest path of the domain, i.e. its critical path, therefore mitigating the impact of process and temperature variations on clocks compared with the usual global clock implementation. Finally, GALS techniques allow each locally synchronous island to be set independently,

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

making multiple local DVFS more convenient than with the standard synchronous approach (Zakaria, Durand, Fesquet, and Marchand 2010).

From the applications perspective, such highly integrated chips are expected to be used in many computing intensive fields, in particular multimedia applications. Thanks to the high computing power, many functions that were traditionally hardwired can now be implemented in software. This is for instance the case for the radio communication receiving system, where components, e.g. filters, mixers and codecs, can now be implemented with increased flexibility in a programmable software defined radio (SDR) layer. 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. Consequently, trade-offs between measures of the multimedia application quality, available on-board energy and desired time to battery exhaustion must be managed, and can be translated into a control problem formulation. Trading performance and resources is relevant to quality of service (QoS) problems, which may be viewed as the management of some complex quality measures providing an image of the satisfaction of application requirements. Indeed, QoS problem statements have been already used for energy-aware multimedia applications (Chiang, Lo, and Lee 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, Steffens, Verhaegh, Bril, and Hentschel (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.

In this paper, we propose to use feedback control loops to control the QoS of an embedded video decoding device. An inner loop controls the DVFS mechanism while an outer loop computes the scheduling set-points needed by the inner loop to process frames decoding. The rest of the paper is organised as follows. Section 2 provides a context and the problem statement. In Section 3, the closed-loop multi-layer architecture for video decoding under energy constraints is sketched. The control loops are then detailed in Section 4 for the DVFS layer and in Section 5 for the frames rate controller. Some experimental results are given in Section 6 using a short movie decoding as a support example. A discussion concludes the paper.

2. Context and problem statement

An original aspect of the present work is to propose a control/computing co-design strategy for video decoding under energy consumption constraints. This is notably possible by means control methods based on feedback.

2.1. *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 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 (e.g. Cervin, Eker, Bernhardsson, and Arzen 2002; Lu, Stankovic, Son, and Tao 2002). 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, Diao, Parekh, and Tilbury 2004; Malrait, Bouchenak, and Marchand 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) combine 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, Ascenso, Pedro, and Pereira (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, Mascolo, and Palmisano (2011) or a Markov Decision Process (MDP) based algorithm as in Xiang, Cai, and Pan (2012). On the decoder side, using feedback is mainly motivated by energy consumption considerations, and the voltage and working frequency are used as control variables. For example, in Pouwelse, Langendoen, Lagendijk, and Sips (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, Lach, Stan, and Skadron (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.

2.2. *DVFS in GALS architectures*

The work presented here uses two low-levels control loops. The first one is based on fast predictive control and manages the chip's DVFS and computing speed. The second one controls the scheduling of the video frames through a

simple loop. 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. However, this framework underlies the design the two lower-level 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 the modelling of the process to be controlled. Basically, control uses an error signal between the desired goal 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 frequency domain is equipped with its own independent timing source. 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 to accommodate to both real-time constraints and allocated energy budget. These well controlled DVFS devices can thus be used as the actuators of the higher level QoS aware loops.

The setup to control the embedded video decoding mechanism of our case study is based on such a GALS-DVFS architecture.

3. 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 building online indicators to carry 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.

3.1. 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 exists 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.

Beyond the basic protocol, the scalable video coding (SVC) extension has been defined to provide scalability capabilities in H.264 (Schwarz, Marpe, and Wiegand 2007). It 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 quantisation steps which selectively cancels some information from the original video (as if it passed through a low-pass filter).

They can be combined to encode/decode a video stream. 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.

3.2. Control architecture

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

(i) At high level, the QoS controller manages the application performance according to the 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 quantisation step, higher resolution and maximal rate. The allowed computing

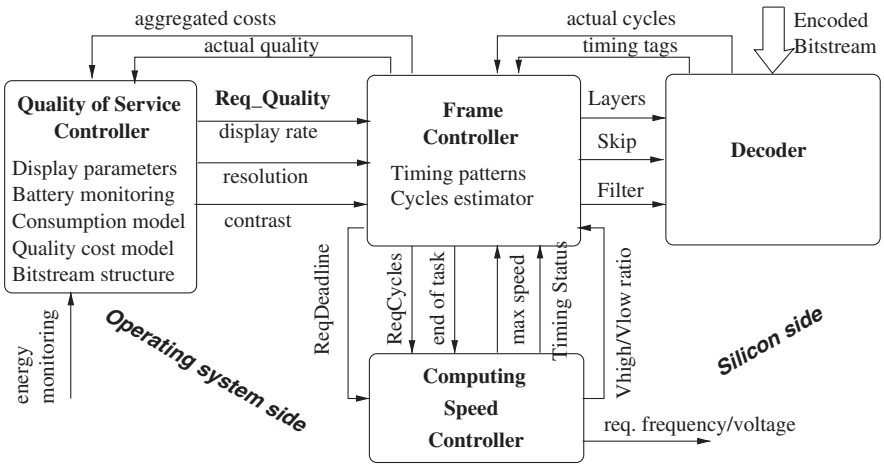

Figure 1. Proposed control architecture of an embedded video decoder.

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 slower time scale than compared with the time scale of frame processing.

(ii) At medium level, frames decoding 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 online 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 *computing 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. It is sampled at a high frequency (e.g. 25 MHz when integrated in a dedicated chip) and provides robustness against the frequency gain uncertainty due the variability of the silicon process.

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.

4. Control of the energy-performance trade-off

The computing speed controller aims at minimising the CPU energy consumption while decoding the frames inside requested deadlines. This is possible by online scaling the supply power according to the varying working load.

4.1. *Energy saving policies*

In current CMOS integrated circuits, the average power consumption P_{avg} and the energy dissipation E are dominated by the dynamic power that arises from electrical gate switching (see Chandrakasan and Brodersen 1995):

$$P_{avg}(t) \propto f_{clk}(t)V_{dd}(t)^2$$
$$E(t) \propto V_{dd}(t)^2, \quad (1)$$

where V_{dd} is the supply voltage and f_{clk} is the clock frequency. These equations show that lowering the supply voltage and slowing the clock can reduce the power and energy consumption. This is the goal of the computing speed controller.

The energy reduction is a quadratic function of the voltage V_{dd} and a linear function of the frequency f_{clk}. However, the supply voltage and clock frequency cannot be set independently. Reducing the supply voltage induces larger delays in the signal propagation through silicon gates. The propagation time along the critical path of the chip must be kept smaller than the clock interval to avoid missing events. Therefore the clock frequency must be also lowered according to the voltage reduction (Zhai, Blaauw, Sylvester, and Flautner 2005). In practice, the supply voltage and clock frequency are coherently managed by a DVFS device.

From the decoding process viewpoint, the objective is that the real-time tasks used to decode frames must be completed by their deadline, while minimising the energy used for the computation. Several policies are already known to cope with this objective, e.g. detailed in Ishihara and Yasuura (1998). As the energy cost is proportional to V_{dd}^2, the computation must be performed using the lower possible

supply voltage compliant with the deadline. Ideally, if both the supply voltage and clock frequency were available as *continuous* variables, the energy optimal policy consists in running the task at the minimum voltage (and associated maximal frequency) needed to exactly meet the deadline.

In practice, the voltage supplied by the DVFS is quantized in few discrete values, e.g. only in two levels for the presented case study. In that case, the optimal policy needs to run the CPU using the two immediate neighbours of the ideal continuous voltage level. Obviously the time spent using the expensive high voltage level must be minimised. Moreover, as the voltage transitions are expensive, their number must be minimised (Miermont, Vivet, and Renaudin 2007).

In other words, during a task execution, the chip should be switched exactly once from the fast mode to the slow mode, so that the task exactly meets its deadline. Note that, for this case, the energy expended to compute the task would be higher than the optimal value found for the ideal case using continuous levels for the voltage and frequency. In that sense, using a limited number of supply voltage levels only allows for sub-optimal solutions compared with the ideal continuous case.

However, both the computation loads and the silicon process are subject to uncertainties. Therefore a feedback controller designed to track in real-time the optimal switching point is expected to perform more robustly than an off-line implementation of the sub-optimal policy.

4.2. Control statement

Assume that the chip can be operated via the DVFS with two voltage levels and their associated frequency levels, denoted V_{level} and f_{level} in the sequel. The computing activity is characterised by the computation speed ω of the processor (e.g. in number of instructions executed every second), shortly denoted speed in the sequel. Its model is a linear static function with unknown parameters

$$\omega = \alpha(V_{dd}) f_{clk} + \beta(V_{dd}), \tag{2}$$

where $\alpha(\cdot)$ and $\beta(\cdot)$ can be identified at some point, but highly vary over time and thus prevents any efficient off-line implementation.

The control strategy consists in a dynamic calculation of the energy-efficient switching point between high and low voltage, i.e. V^{high} and V_{low}, able to satisfy the control objective. The computation of this set-point assumes that the number of instructions Ω_i and the deadline Δ_i needed to process the next task T_i (decoding frame i) are known. Indeed these quantities are estimated by the higher-lever frame controller (this is discussed in the sequel).

Let ω^{high} and ω_{low} denote the maximal computing speeds when the system is running at V^{high} and V_{low}, respectively. In the ideal case with continuous voltage levels, computing the task at the average speed $\bar{\omega}_i = \Omega_i/\Delta_i$ would exhaust all the instructions by the deadline (assuming that one instruction is executed for each clock cycle). Note that if $\bar{\omega}_i > \omega^{\text{high}}$ the execution of task T_i is not feasible by the desired deadline.

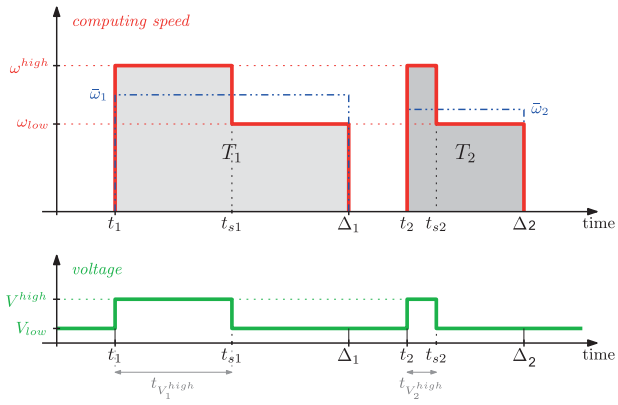

Figure 2. Speed profile for a task execution.

With only two voltage levels, computing a feasible task while spending the minimum cost consists in splitting its execution in two parts as depicted in Figure 2. The task first runs at high frequency and voltage (if required), thus achieving the maximal possible computing speed ω^{high} and running faster than the average (as for T_1 from time t_1 to t_{s1}). Then, the execution finishes at the low voltage and speed (as from t_{s1} to Δ_1) which consequently reduces the energy consumption. Note that beginning by the high speed level allows for on-the-fly corrections to meet the deadline even if ω^{high} or Ω_i were not correctly estimated, whereas their values are re-evaluated during the task execution.

To summarise, minimising the energy consumption of the working CPU during the execution of a task T_i requires to minimise the time spent at $V^{\text{high}}/F^{\text{high}}$. However, due to varying working loads and silicon performance, the time t_{si} at which the supply voltage must be switched from V^{high} to V_{low} cannot be *a priori* known. A feedback controller, based on a predictive control law, is designed in the next section to compute the switching time t_{si} in real-time, according to online measurements available from the decoding process.

4.3. Fast predictive control law

The presence of deadlines and input constraints to compute repetitive tasks naturally leads to a predictive control specification. Predictive control consists in finding an open-loop control profile over some time horizon and in applying it until the next time instant. The control problem is then reconsidered using the new state variables and a new control profile is generated. This finally yields a closed-loop control and the stability relies in the way the open-loop control is chosen.

The time horizon can be constant, infinite or less classically *contractive* as in the present case, where it corresponds

to the task deadline Δ_i. This means that the time to achieve the objective decreases with the laxity (i.e. the remaining time before the end of the task processing), and the horizon is updated for each image i to decode.

The key point is the choice of the open-loop strategy. Here, the problem is to find the computing speed profiles which minimise the high voltage running time $t_{V_i^{\text{high}}}$ while guaranteeing that the T_i-task processing meets its deadline. The predictive control problem can be formulated as a constrained energy-performance trade-off optimisation problem, that is mathematically expressed by

$$\min\ t_{V_i^{\text{high}}} \qquad \text{s.t.} \quad \int_{\Delta_i} \omega(t)\, dt = \Omega_i, \tag{3}$$

where $\int \omega dt$ is the current number of executed instructions for the image being processed.

Predictive control is known to be a robust approach, able to easily handle many kinds nonlinearities and constraints. It is also known to often suffer from high computational costs which are hardly compliant with real-time computing. However, the simplicity of system (2) allows for considering a cheap and real-time compliant *fast predictive control* design. It consists in taking advantage of the structure of the dynamical system to speed up the finding of the open-loop control (see Alamir 2006).

Actually, the closed-loop solution yields a fast algorithm since only two parameters need to be known, (i) the computational load to be processed Ω_i and (ii) the remaining time to achieve it, i.e. the laxity Λ_i. The computing speed required to meet the deadline (denoted the *predicted speed* $\upsilon(t_k)$) is dynamically calculated at each sampling instant k (where the sampling interval is much smaller than the decoding duration of an image)

$$\begin{aligned} \Omega(t_k) &= \Omega(t_{k-1}) + T_s \omega(t_k) \\ \upsilon(t_{k+1}) &= \frac{\Omega_i - \Omega(t_k)}{\Lambda_i(t_k)}, \end{aligned} \tag{4}$$

where Ω is the sum of the computing speed ω 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 Ω_i for the task execution number i is assumed constant during T_i decoding. Nevertheless, it might be updated by the frame controller at any time during the execution of T_i for the case where decoding milestones can be identified and used on the fly.

The energy-efficient computing 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} as long as the required computing speed υ is higher than speed at low voltage ω_{low}. Otherwise, the computing speed achieved at V_{low} is fast enough to finish to decode the image on time. Finally, the control decisions are

$$\begin{cases} V_{\text{level}}(t_{k+1}) = V^{\text{high}} \\ f_{\text{level}}(t_{k+1}) = F^{\text{high}} \end{cases} \quad \text{if} \quad \upsilon(t_{k+1}) > \omega_{\text{low}} \tag{5}$$

$$\begin{cases} V_{\text{level}}(t_{k+1}) = V_{\text{low}} \\ f_{\text{level}}(t_{k+1}) = F_{\text{low}} \end{cases} \quad \text{otherwise.}$$

In practice, the frequency associated to each voltage level is usually the maximal possible one for this voltage, in order to minimise the running time. However, it frequently happens that several frequency levels are associated with the lower voltage level, thus providing added flexibility when approaching weak computing activity. A last level is given by a 'clock gating' capability, where the clock of an idling processing unit can be disabled, thus even reducing the energy consumption down to the leaks. These additional degrees of freedom can be used with the same control algorithm (4)-(5), following the principle of Section 4.1 stating that the optimal computing speed profile uses the two immediate neighbours of the ideal constant speed $\bar{\omega}_i$. The design of such controllers is detailed in Durand and Marchand (2011), and implemented for the experiment of Section 6 with a chip allowing for two voltage levels and three frequency levels, one at V^{high} and two at V_{low}.

Even if the proposed algorithm (4)-(5) is very simple, it belongs to the family of predictive control in the sense that the control profile is calculated for a given time horizon, i.e. the laxity Λ_i remaining to execute the current task T_i. It is structurally optimal as it implements an optimal policy stated in Ishihara and Yasuura (1998) to manage the computing speed/energy trade-off in chips controlled by a DVFS (Section 4.2). Obviously, the optimal trade-off would be reached only if all components were perfect or predictable. Compared with an off-line implementation, it is asserted that using feedback loops allows for automatically and robustly approach the optimal policy despite of system uncertainties and variability.

4.4. *Stability and robustness*

The stability of the control scheme follows from 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 in the contractive time horizon of the task T_i as

$$V(t) = \Omega_i - \int_{\Delta_i} \omega(t) dt. \tag{6}$$

This expression comes from Equation (3). A discrete version can also be obtained from Equation (4) for the proposed

control algorithm, which leads to the same results. Classically, in the predictive control framework, Equation (6) represents the integral of a cost function over the prediction horizon and can be considered as a candidate Lyapunov function. V is continuously decreasing, as the computing speed ω can be only positive, which ensures the stability of the decoding controlled system.

Robustness is also an important issue. It deals with the conservation of the stability property when things do not behave as expected. Unexpected behaviours can be of two types: a wrong estimation of the number of instructions to process Ω_i, or the presence of process variability.

In the first case, if Ω_i is overestimated, the task will be completed before its deadline but V will remain decreasing. The clock-gating mechanism can then be used. If Ω_i is underestimated, the deadline will be missed but the damping buffer and deadline controller will anyway process the incoming data without loss (see Section 5.1). Nonetheless, V remains decreasing during the whole task execution.

In case of process variability, the performance of the system is affected, as the real computing speed becomes

$$\omega_{\text{real}}(t) = \kappa\, \omega(t), \tag{7}$$

where $\kappa \geq 0$ is the unknown process variability factor. $\kappa = 1$ means that the real process behaves ideally according to model (2). $\kappa < 1$ means that the performance of the system is weaker than expected, which is a problematic case leading to systematic deadlines misses if no online corrective actions are given. $\kappa > 1$ means that the system is faster than expected, leading to idle times of the processor. Nonetheless, V in Equation (6) becomes

$$V(t) = \Omega_i - \kappa \int_{\Delta_i} \omega(t)dt \tag{8}$$

and so the stability is still ensured for all $\kappa > 0$. The convergence rate is reduced for $0 < \kappa < 1$ and increased for $\kappa > 1$. As a result, the system runs a longer (respectively shorter) time at the penalising high supply voltage (by construction of the control law) to compensate the weak (respectively strong) performance. The only unstable case is for $\kappa = 0$, which means that the processor does not compute at all. In that case, the operating system must not allocate tasks to this part of the chip.

Especially, note that the computing speeds ω_{low} and ω^{high} required by the control algorithm (4)-(5) are estimated in real-time from the actual measured speed (as detailed in Durand and Marchand 2011), so that corrective actions can be computed at the sampling rate of the controller. Hence, the switching point computation does not rely on any wrong off-line estimation of the computing speeds, and the tasks will meet their deadline as far as the processor is able to execute the computational load by the deadline. The condition for a task to be feasible is then $\bar{\omega}_i \leq \kappa\, \omega^{\text{high}}$.

5. Frame control layer

The second control layer feeds the computing speed controller with estimates of the amount of computations to be performed (i.e. Ω_i) within an associated deadline (i.e. Δ_i) for each image i to decode. This is developed in the sequel whereas a damping buffer is first introduced to store unprocessed data in case of missed deadlines.

5.1. *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 from the decoding order due to dependencies between images of different types I, P and B. This was explained in Section 3.1. Therefore, the displayed flow is inevitably delayed with respect to the incoming bitstream.

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 quantisation step $Q_p = 28$ (note that larger is the quantisation 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 constant for quite long intervals, with abrupt changes between flat areas and isolated high values. Note that Wurst et al. (2005) recorded similar load profiles, although using a MPEG2 decoder with a different codec scheme.

Hence, following the ideas in Wurst et al. (2005), an additional buffer is added to the frame decoding queue in such a way that decoding is performed several frames ahead of display. This damping buffer is used to store data in excess due to unusually complex frames and gives space and accommodation for the varying computing loads between frames. As the recorded worst case values of the isolated peaks are about three times larger than the average, it is induced that a buffer of three-frames depth is sufficient to damp most of the computing load variations, while avoiding data loss due to buffering overflow.

5.2. *Computation load estimation*

The goal of the frame controller is to provide the inner computing speed controller with estimates of the

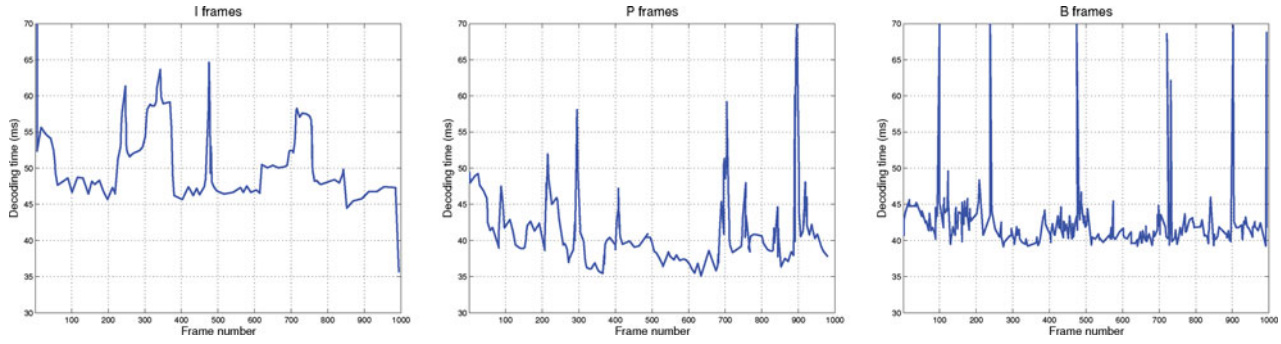

Figure 3. Decoding times for I, P and B frames: (a) frame I, (b) frame P and (c) frame B.

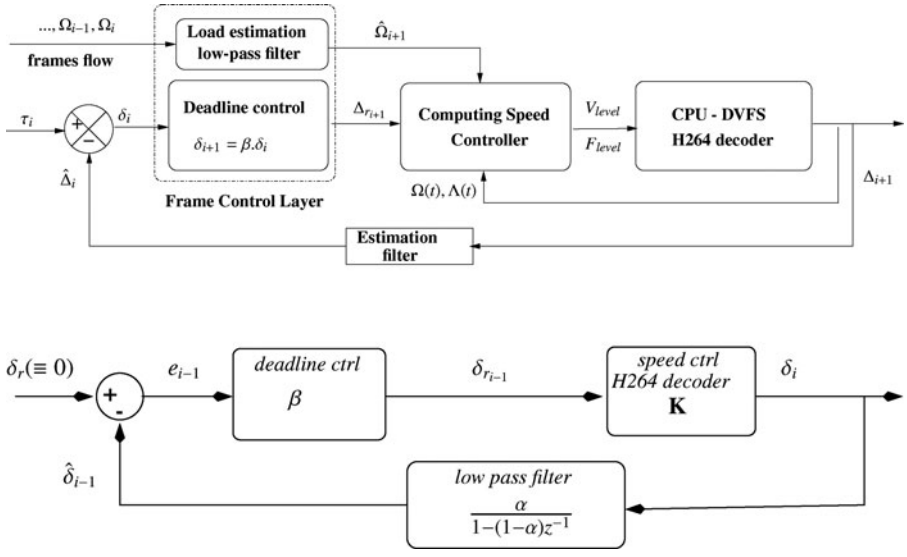

Figure 4. Deadline control architecture: (a) deadline and computing speed cascade loops and (b) deadline control loop.

number of cycles to be processed for the next frame ($\hat{\Omega}_{i+1}$), and the allocated deadline to decode the frame ($\Delta_{r_{i+1}}$) (see Figure 4).

The measurements plotted in Figure 3 show that the decoding time profiles gather stochastic and structural components, so that the computing load needed to decode a particular frame cannot be precisely inferred from the previous ones.

Therefore, rather than trying to compute any prediction, the estimation of the next frame computation load $\hat{\Omega}_{i+1}$ can be simply taken equal to the last one, i.e. Ω_i, recorded by the instrumentation inserted in the H.264 decoder). Even better, it can be provided by smoothed past values through a low-pass filter:

$$\hat{\Omega}_{i+1} = \alpha\hat{\Omega}_{i-1} + (1-\alpha)\Omega_i, \tag{9}$$

where $0 \leq \alpha < 1$ controls the filter damping.

5.3. *Deadlines dynamic allocation*

The goal of the deadline controller is to allocate the deadline of the next frame decoding. Considering a fixed ideal schedule $\{..., \tau_{i-1}, \tau_i, \tau_{i+1}, ...\}$, e.g. with equidistant intervals of 40 ms, the feedback loop is aimed to regulate the suite of requested deadlines Δ_{r_i} towards their ideal values τ_i. However, due to the fluctuations of the computing load, the actual (measured) end of decoding time Δ_i for frame i deviates from its requested value, which yields

$$\Delta_i = \tau_i + \delta_i, \tag{10}$$

where δ_i is the deadline error. Remember that in case of overshoot ($\delta_i > 0$), the unprocessed data are temporarily stored in the damping buffer and that Δ_i is only known at the end of the frame decoding.

To avoid stressing the speed controller and to provide a smooth control of the deadline despite the unpredicted

fluctuations decoding times, the control objective states that the deadline error should exponentially decay to 0 (for a constant input load):

$$\delta_{i+1} = \beta\, \delta_i, \tag{11}$$

where β controls the decay rate, with $0 < \beta \leq 1$. Gathering Equations (10) and (11) leads to compute the ideal end-of-computing time for the next frame, which is chosen as the target deadline:

$$\Delta_{r_{i+1}} = \tau_{i+1} + \beta\, \delta_i. \tag{12}$$

Indeed the control architecture depicted by Figure 4(a) is a standard cascade controller as depicted in textbooks, e.g. Skogestad and Postlethwaite (2005) among many others. The inner loop (i.e. the speed controller) isolates the frame controller from the effects of the actuator's imperfections, namely the dispersion of working frequencies and gains due to the variability of the silicon process. Note that the bandwidth of the inner loop is by far faster than the one of frame controller which is sampled at the video rate. Therefore, its dynamics can be neglected for the frame controller stability analysis, and its transfer is approximated by a gain K between the actual and the requested deadline $K = \frac{\Delta_{i+1}}{\Delta_{r_{i+1}}}$. Thanks to the speed controller and associated DVFS, this gain is most often close to 1 ($K \simeq 1$), but may have transiently larger values when peaks of incoming data lead to an under-evaluation of the expected processing load Ω. Note that the decoder introduces a one frame computation delay: the deadline error δ_i actually observed after processing frame i ending at time Δ_i is the result of the requested deadline $\delta_{r_{i-1}}$ computed at time τ_{i-1}.

As usual when designing feedback control for computing devices, the loop dynamics is mainly due to the measurement filters (Cervin et al. 2002). The controller design is restated in the framework of deadline errors. Considering a one sample delay for the decoding process and an optional low-pass filter (9) in the return path (see Figure 4(b)), the z-domain transfer function of the deadline error is given by

$$\frac{\delta}{\delta_r} = \frac{\beta K[1-(1-\alpha)z^{-1}]}{z-1+\alpha(1+\beta K)}, \tag{13}$$

which is stable if its unique pole is inside the unit circle. Therefore, the deadline control loop is stable for

$$\alpha(\beta K + 1) < 2, \tag{14}$$

where $0 < \alpha \leq 1$ and $0 < \beta \leq 1$ to provide adequate damping. If the measuring filter is omitted (as in the experiment of Section 6), $\alpha = 0$ and the stability condition simply strips down to $\beta K < 1$. Note that small values for the control gain β compensate for large values of K, thus providing robustness with respect to the uncertainties of the computing speed control loop.

The capabilities of this controller can be exhausted if successive peaks of processing load overflow the three-frames ahead buffer. In that case, only the I frames has to be fully decoded up-to the requested quality level, while the depending P and B frame decoding can be truncated up-to recovering enough buffering 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.

6. Experimental results

6.1. Implementation

The proposed control architecture has been implemented and experimented with video samples. 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 real-time threads switching and allow for the very precise measurements of execution times (down to the hardware resolution) which are necessary to feed back the control 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 decoders run on the other. To emulate the speed controller (which should be integrated in silicon in a dedicated chip) the frequency scaling capabilities of the chipset have been used. At this end, the predictive controller of Section 4 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 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. Therefore, the source code can be easily instrumented to integrate the control features described in the previous sections. The counterpart with respect to optimised parallel versions is that only low resolution decoding can be achieved in real-time. Nonetheless, the setup is able to demonstrate the feasibility and efficiency of the approach.

6.2. 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 × 352 pixels) is used. Two quality layers are 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 $F_1 = 2535$ MHz, $F_2 = 1600$ MHz and $F_3 = 800$ 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 65 ms. Therefore, the decoding process is stressed for the shorter requested values, but 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.

6.2.1. Frame deadline control

The first experiments only use the deadline controller while keeping the CPU frequency constant. A buffer of three-frames depth is implemented to damp the decoding overshoots. The filtering gain for the load estimate (9) is set to $\alpha = 0.1$ and the damping gain for the deadline controller (12) is set to $\beta = 0.3$. Preliminary tests showed that using a low-pass filter on the deadline measurement is useless for this case study. Figure 5 plots the evolution of the deadline overshoot δ for deadlines scheduled every 50 ms. In Figure 5(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, when using the deadline controller of Section 5.3, the deadline overshoots due to peaks in decoding time are quickly absorbed and the real decoding schedule is kept close to the theoretic one, as shown in Figure 5(b).

6.2.2. 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 6, 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 when not needed, i.e. for large requested deadlines.

(ii) In Figure 7, the speed controller introduced in Section 4 is active and the deadlines are fixed. For the shortest requested deadlines, even when the frequency is always maximal the deadlines cannot be reached due to the poor performance of the JSVN decoder 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. Thus the energy consumption is actually reduced when the deadline constraint is weak.

(iii) In Figure 8, the speed and frame controllers of Sections 4 and 5 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 when the plot is respectively 1/0 in Figure 8). As expected, this filter is all the more activated as the decoding constraints are weak.

6.2.3. 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 energy spent by the peripherals (e.g. the GPU).

Nevertheless, available models, e.g. Chandrakasan and Brodersen (1995), state that the energy consumption of a chip behaves as Equation (1). This relation is used to compare and sort the control strategies used along the experiments, according to a normalised energy consumption.

The used chipset and associated clock driver can be operated using the following couples of parameters:

- $f_{clk} = 2535$ MHz, $V_{dd} = V$;
- $f_{clk} = 1600$ MHz, $V_{dd} = V/2$;
- $f_{clk} = 800$ MHz, $V_{dd} = V/2$.

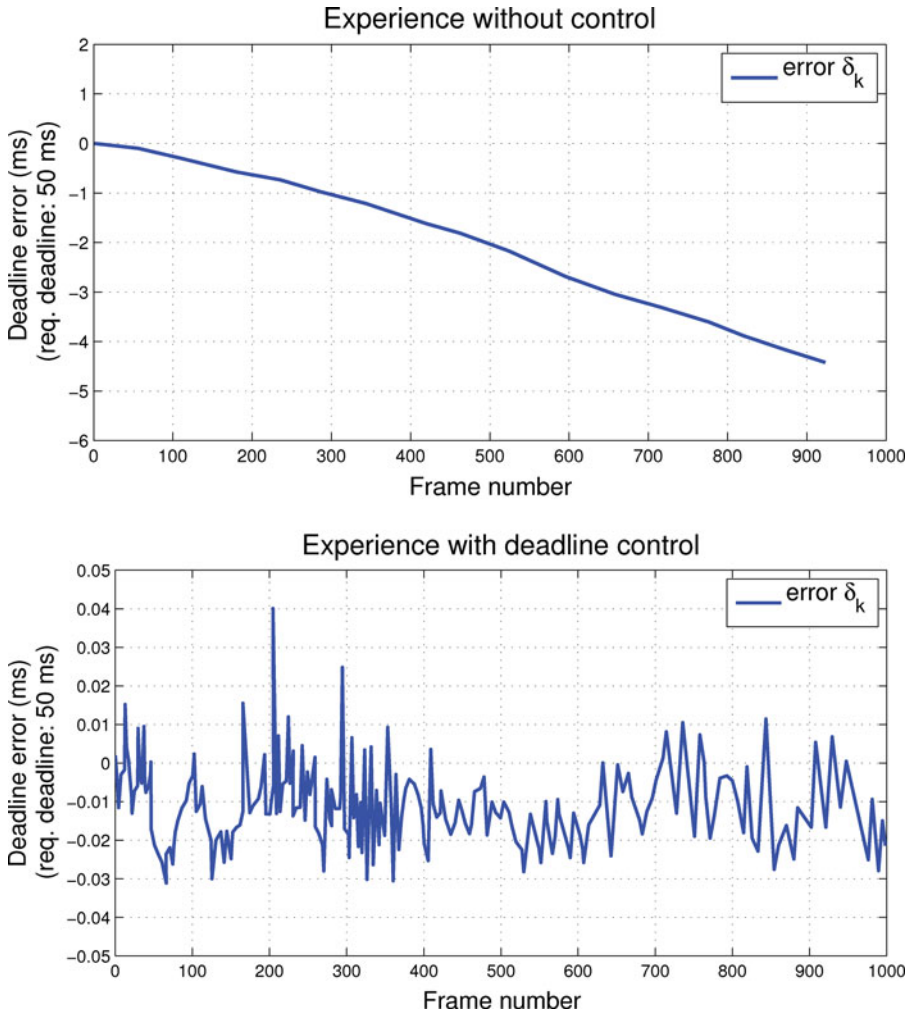

Figure 5. Deadline errors when keeping the CPU frequency constant: (a) without any control and (b) with deadline control.

Table 1. Energy consumption estimation.

	Normalised energy consumption	
Deadline	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

Table 1 summarises the normalised energy consumption for the already used video sequence, for which the average frame decoding time at the highest quality is 50 ms.

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.

6.2.4. *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:

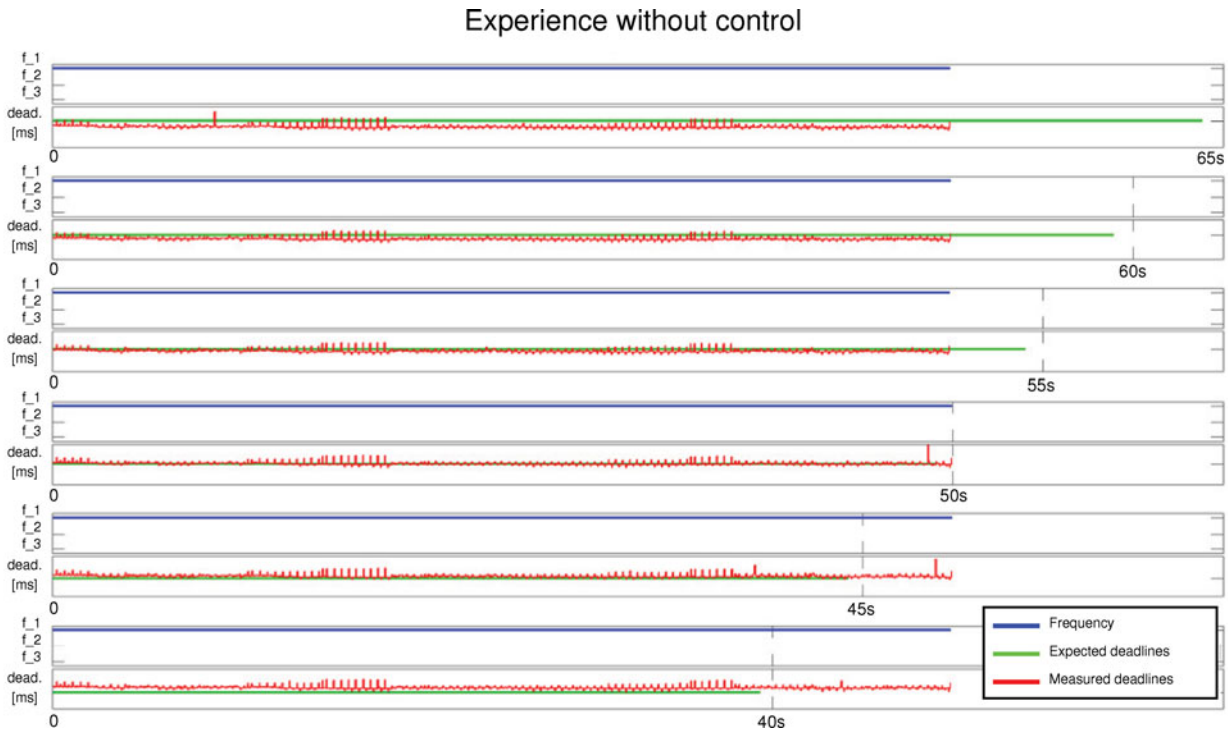

Figure 6. Decoding without any feedback control.

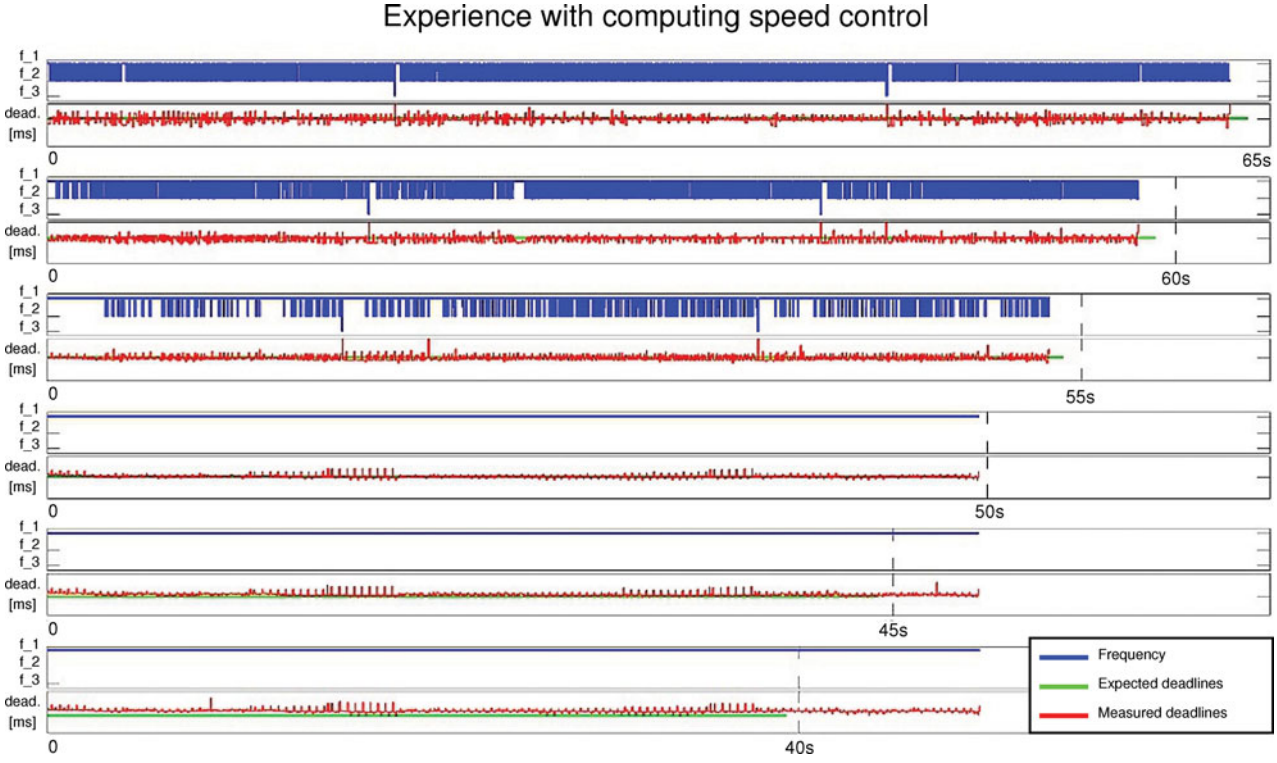

Figure 7. Decoding with computing speed control only.

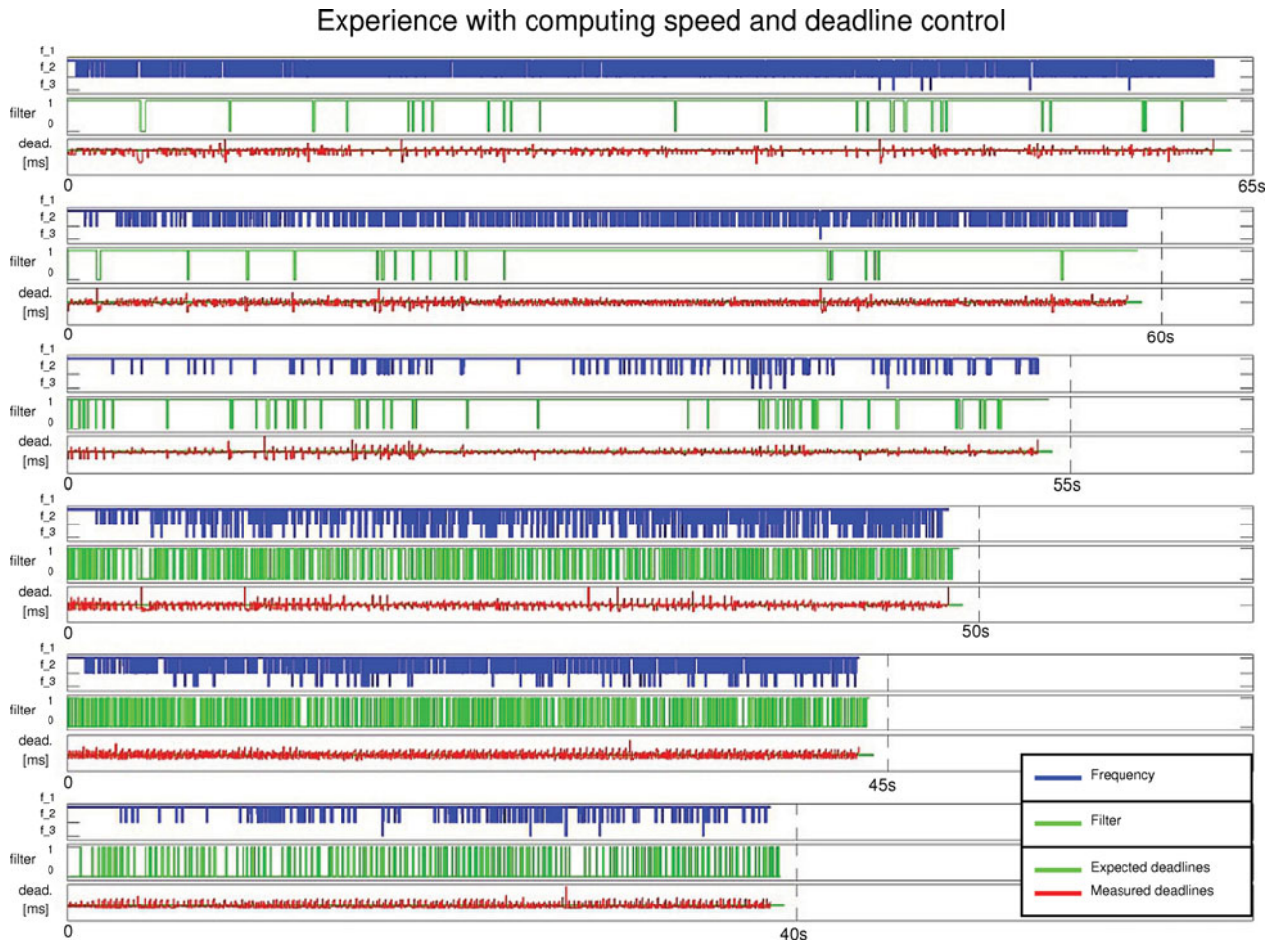

Figure 8. Decoding with computing speed and deadline controllers.

- 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 plotted in Figure 9 only for the stressing deadlines (40, 45 and 50 ms), since longer deadlines are always achieved without 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 substantially decreasing the computing and energetic costs.

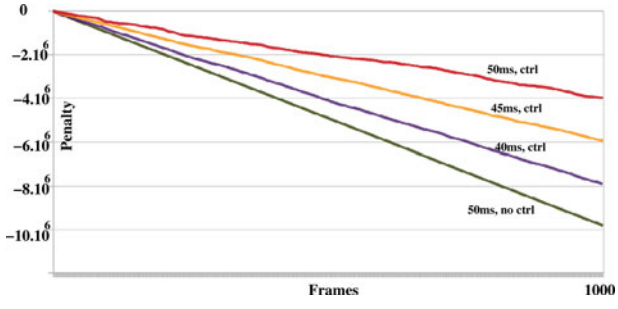

Figure 9. Decoding quality and penalties.

7. 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 robustly manage a trade-off between the computation load and related energy cost in one hand and the application quality in the other hand.

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.

A low level controller relies on the frequency and voltage scaling capabilities of the chip. At a higher level, the decoding scheduling parameters, namely the estimated computation loads and associated deadlines, are provided by a frame control layer dedicated to the video application. Both controllers are simple enough to be run with very small execution overheads.

An experimental validation of the decoder and associated controllers has been setup on a single core system, using different power modes for decoding under various conditions and constraints. The results show that, under tight operating constraints, a specified video quality can be achieved while saving up to 25 % of energy compared with the uncontrolled case. Thanks to feedback loops, adaptive decoding can be achieved using very simple models of the process.

Indeed, this is once again an example of the ability of feedback control to robustly handle systems with uncertainties and variability, without needing a detailed knowledge of the plant. The application of feedback control integrated in computing devices is still unusual, but it is raising a growing interest, both for ground applications and for mobile devices.

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 (45 nm and smaller).

Note

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

Notes on contributors

Sylvain Durand received MEng degree in Control and Embedded Systems from ESISAR (Grenoble-INP), Valence, France, in 2007, and PhD degree in Systems and Control Theory from Grenoble University, Grenoble, France, in 2011. After a PhD at the INRIA/GIPSA-lab, Grenoble, France, he held a post-doctoral position in 2011 at the Integrative Research Center (CRI PILSI) in a strong collaboration with the CEA-Leti, Grenoble, France. He is currently a post-doctoral researcher at the UMI LAFMIA (CNRS, CINVESTAV), Mexico City, Mexico. His work is dedicated to the control of embedded and networked cyber-physical systems with limited resources. In particular, he is investigating non-uniform sampling using event-based and self-triggered control techniques.

Anne-Marie Alt received her Engineering degree in Mechatronics from ENSIAME and the Research Master's degree in Electronics and Telecommunication from the University of Valenciennes in France in 2008. She worked as a Software Engineer in Automatics and Embedded Systems in INRIA in Grenoble, France for one and a half year. Since 2010, she has been working as Hardware and Software Engineer in KROHNE in Romans sur Isere, France in the Research and Development Department. She is currently developing and testing the level meters based on time domain reflectometry and radar technologies, taking care of high accuracy and low power consumption.

Daniel Simon graduated as a Engineer (1976) and Doctor-Engineer (1980) from the School of Mechanics and Microtechnologies (ENSMM), Besançon, France. He also received the 'habilitation à diriger des recherches' from University of Nice Sophia-Antipolis in 1998. He was a research scientist with INRIA since 1980, and is now member of the DEMAR team shared by INRIA and LIRMM in Montpellier, France. His main interests are with co-design methodologies for feedback control and real-time computing, with current applications in robotics and bio-medical engineering.

Nicolas Marchand was born in Suresnes, France. He graduated in Electrotechnical and Control Engineering from the Grenoble Institute of Technology (Grenoble INP), France (1995) and received a PhD in Nonlinear Control (2000) at Grenoble INP. After one year at the University of Lyon, France, he held for two years an Assistant Prof. position at the University of Paris XI in Gif-sur-Yvette, France, and is currently a CNRS fellow at the GIPSA-lab, Grenoble, France. He is currently the leader of the Control Systems Department and deputy director of GIPSA-lab. His research interests are on modelling and control for computer systems and biomimetic micro unmanned aerial vehicles (MAVs).

References

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

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 Lee, W.T. (2008), 'Scalable Video Coding of H.264/AVC Video Streaming with QoS-based Active Dropping in 802.16e Networks', in *Proceedings of the 22nd International Conference 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 *ACM Multimedia Systems Conference (MMSys 11)*, San Jose, CA, USA, pp. 145–156.

Durand, S., and Marchand, N. (2011), 'Fully Discrete Control Scheme of the Energy-Performance Tradeoff in Embedded Electronic Devices', in *Proceedings of the 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 *Proceedings of the 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 *Proceedings of the 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)*, Paris, France, Vol. 3, pp. 30324–30327.

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

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.

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 *Proceedings of the 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.

Miermont, S., Vivet, P., and Renaudin, M. (2007), 'A Power Supply Selector for Energy- and Area -Efficient Local Dynamic Voltage Scaling', in *PATMOS'07: 17th International Workshop on Power and Timing Modeling, Optimization and Simulation*, Gothenburg, Sweden, pp. 556–565.

Pouwelse, J., Langendoen, K., Lagendijk, R., and Sips, H. (2001), 'Power Aware Video Decoding', in *Proceedings of the 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 *Proceedings of the 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 *Proceedings of the 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 Transactions on Circuits and Systems for Video Technology*, 17(9), 1103–1120.

Skogestad, S., and Postlethwaite, I. (2005), *Multivariable Feedback Control – Analysis and Design*, Chichester, UK: Wiley.

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 *Proceedings of the 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 *Proceedings of the 3rd Multimedia Systems Conference*(*MMSys'12*), Chapel Hill, North Carolina, pp. 167–172, New York, NY: ACM.

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.

Zhai, B., Blaauw, D., Sylvester, D., and Flautner, K. (2005), 'The Limit of Dynamic Voltage Scaling and Insomniac Dynamic Voltage Scaling', *IEEE Transactions on Very Large Scale Integrated Systems*, 13, 1239–1252.