Energy and Computing Ressource Aware Feedback Control Strategies for H.264 Video Decoding

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

Feedback Control Strategies for H.264 Video Decoding under Computing Resource and Energy Consumption Constraints

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

Abstract

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 accommodate for both the performance uncertainties due to the tiny scale gates variability, varying and poorly predictable computing demands and limited energy storage constraints.

This paper presents the practical example of an embedded video decoder controlled by several cascaded 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 to adapt the computing speed of the chip to the demands of the video flow decoder. The outer loop is fed back with measures coming from the current frame decoding execution, and computes the scheduling set-points needed by the inner loop to process the next frame decoding. A user-level control system provides Quality of Service parameters to the inner and outer loops to meet end-user requirements gathering decoding quality and energy consumption related constraints.

The feedback loops have been implemented on a stock PC and some experimental results are provided. It is shown that a noticeable reduction of the energy consumption can be achieved while preserving a requested decoding quality, and that the robustness of feedback loops accommodates for the uncertainty coming both from the chip's variability and from the demanded computing power.

Index Terms

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

INTRODUCTION

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

S. Durand, A-M. Alt and D. Simon are with the NeCS project-team, an INRIA Grenoble Rhône-Alpes and GIPSA-lab joint team, Grenoble, France (sylvain.durand@inria.fr, daniel.simon@inria.fr)

N. Marchand is with the SysCO team, Control Department of GIPSA-lab, Grenoble University, Grenoble, France (nicolas.marchand@gipsa-lab.fr)

other hand, they become even more miniaturized (which means limited energetic resources) while needing increased autonomy. These constraints are clearly conflicting, therefore technological evolutions are needed to improve both 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 categorized in three main points:

a) Energy saving requirements: In current CMOS integrated chips, the energy reduction is a quadratic function of the supply voltage and a linear function of the clock frequency [1]. As a result, Dynamic Voltage Scaling (DVS) can be used to efficiently manage the energy consumption of a device [2]. The supply voltage can be reduced , but one has to take care that scaling down the voltage of a microprocessor increases signals delays along the paths through electronic gates, thus needing to decrease its clock frequency as well. Therefore, adapting the supply voltage is very interesting when possible but implies the use of Dynamic Frequency Scaling (DFS) to keep correct the system behavior. The addition of DFS to DVS is called Dynamic Voltage and Frequency Scaling (DVFS) and results in simultaneously managing the frequency and the voltage. 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. Decreasing the processor clock frequency reduces power consumption but simply spreads the computation out over time, thereby consuming the same total amount of energy as before. Finally, reducing the voltage level as well as the clock frequency achieves the desired goal of reduced energy consumption at an appropriate performance level [3]. Furthermore, several behaviors are known to minimize the energy consumption while guaranteeing good computational performance (see [4] for further details). Classically, each task is considered independently, running at a constant voltage whose value is set to meet the deadline. If only a limited number of discrete voltage levels are available, the voltage values closest to the optimal one are the best choices to lower the energy consumption. Selecting some of these levels leads to a drastic energy reduction even if the number of levels is very small. At the end, the voltage has to be reduced as much as possible and the frequency adapted to the computational load [5]. A closed-loop DVFS approach is hence a good solution for energy saving.

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 implementation may vary far from this expected performance [6]. Moreover, some cores may behave differently into the same chip. In particular, different cores will be capable of different maximal clock frequencies even if coming from the same design [7]. 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, it is needed that each computing node (or group of nodes) can be driven up to its maximal clock frequency. A common solution is the use of the Globally Asynchronous Locally Synchronous (GALS) paradigm. By removing the globally distributed clock, such circuits provide a promising solution for 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). A GALS architecture can mitigate the impact of process and temperature variations, because a globally asynchronous system does not require that the global frequency is dictated by the longest path delay of the whole chip, i.e. the critical path. In this case, each clock domain frequency is only determined by the slowest path in its domain. Furthermore, GALS techniques allow each locally synchronous islands to be set independently, making voltage and frequency scaling more convenient than with the standard synchronous approach [8].

c) Quality of Service (QoS) requirements: Highly integrated chips are expected to be used in many computing intensive fields. Typical ones are multimedia applications, such as receiving, decoding and displaying high definition television streams on mobile devices. This particular case depicts the study described in the present paper. 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 were primarily stated and studied in the framework of networking. More generally, and beyond the initial (network related) meaning, QoS 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 [9], most often focusing on networking load and communication management rather than on computing itself. However an approach stating video processing as a QoS problem is reported in [10]. 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. Video processing is known to be subject to fluctuations and content-dependent processing times: at run-time QoS management is made adaptive and robust against the incoming stream uncertainties by an active closed-loop control, where measures of the system's outputs (e.g. actual CPU load and deadlines miss) are fed back to the decoder's input to chose the next frame decoding quality level. Even if considering a single CPU with constant processing power, the approach shows a very effective and flexible decoding adaptation capability. Such closed-loop control scheme follows several steps. Control design first needs a definition of the control objectives and an analysis and modeling 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 must be implemented. Then various control algorithms can be used to cyclically compute commands to be applied to the process via actuators, which must be also identified and implemented.

Using a GALS approach offers an easy integration of different functional units. Moreover, 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 natural enablers for distributed power management systems as well as for local DVFS. This is even better not only in terms of power and performance, but also in terms of variability [11]. 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. Furthermore, another control could lighten the computing load of a given functional unit by dealing with the QoS at the application level, with the limitation of processing power and/or channel of communication and with some constraints in energy consumption. This latter loop also manages the different voltage/frequency domains of the chip with their own performance. A practical example of such an architecture is part of the ARAVIS[1] project, which aims at providing architectural solutions for manufacturing circuits in nanometric technologies with strong technological uncertainties. The suggested solution consists in using a closed-loop scheme to improve performance and several overlapped loops were suggested in order to manage the energy and activity in a chip in advanced technology. The setup to control the embedded video decoding mechanism of our case study is then based on this architecture.

Structure of the paper

The paper is organized as follows. In section I, we give an overview of the video decoding process. In particular, we introduce the H.264/SVC standard with the different allowed scalabilities as well as the decoding process, in order to see how the quality of such a system can be controlled. Then, in section II we detail a closed-loop architecture for video decoding under computing resource and energy consumption constraints and develop the different proposed control strategies. Finally, some experimental results are presented in section III when decoding a short movie as an example.

I. H.264/SVC CODEC OVERVIEW

H.264 (also known as MPEG-4/AVC) is an international video coding standard for Advanced Video Coding proposed by the Joint Video Team: it was first approved in 2003 [12]. It is intended to be used in many multimedia applications such as downloading and streaming via Internet, software defined radio, multimedia for mobile devices and high definition television. More recently the Scalable Video Coding extension (SVC) has been defined to provide scalability capabilities, e.g. enabling multiple resolutions and various quality levels in compressed bitstreams. Therefore this extension is expected to provide the actuators needed by a QoS control loop working on top of the decoder.

[1]Advanced Reconfigurable and Asynchronous Architecture for Video and Software radio Integrated on chip http://www.minalogic.org/TPL_CODE/TPL_PROJET/PAR_TPL_IDENTIFIANT/903/99-embedded-electronics-nanoelectronics-technology.htm

A. Features of H.264/SVC

The H.264/SVC reference software, which implement the features of the Joint Scalable Video Model (JSVM[2]) algorithm, has been elected for preliminary experiments. Although it is not optimized, this software implements all the features defined for the H.264 standard and for the associated SVC extension which are detailed in [13].

Basically SVC enables to decode only some selected parts of the incoming compressed bitstream. First, the raw input video flow is encoded to obtain a compressed bitstream. At coding time SVC allows for encoding the input video with combinations of different temporal rates, spatial resolutions and quantization steps. At the output of the encoder, the bitstream which contains several quality layers is sent to the decoder via a communication medium. Before decoding, selected partial bitstreams are extracted from the initial bitstream. Finally, only selected partial bitstreams are decoded, while switching between layers is possible in some cases. Three types of scalability are allowed by SVC:

i) ***Spatial scalability*** enables to encode a video with several resolutions (i.e. number of pixels in a picture). The original high resolution video is down-converted to new video streams with lower resolutions. The final bitstream contains the video with all the encoded resolutions. The decoder decodes first the picture with the lowest resolution, then pictures with higher resolutions if needed.

ii) ***Temporal scalability*** enables to encode all or a part of the frames of the original video with different rates. For example it can be chosen to encode only half of frames to save computing power or networking bandwidth: in that case the displayed video only contains 15 frames per second rather than 30 frames per second in the original video stream.

iii) ***Quality scalability*** allows to encode the frames with several quantization steps. The quantization step selectively cancels some information from the original video: its effect can be compared to a low-pass filter. Indeed, the human eye is more sensible to the low frequencies than high frequencies, so that canceling high contrasts can be done progressively with a moderate impact on the visual perception. High quantization steps lead to lower quality pictures but also to lower computing costs. For a frame containing several quantization based quality layers, the decoder must process first the lowest quality layer (with the highest quantization step), then layers of increasing quality.

These different scalability properties can be combined to encode/decode a video stream as depicted in Figure 1. It is stated that the decoding process necessarily flows from lower to higher quality layers. Obviously all the quality layers needed by the decoder must have been previously encoded and transmitted over a communication channel to the decoder.

B. Video bitstream structure

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

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

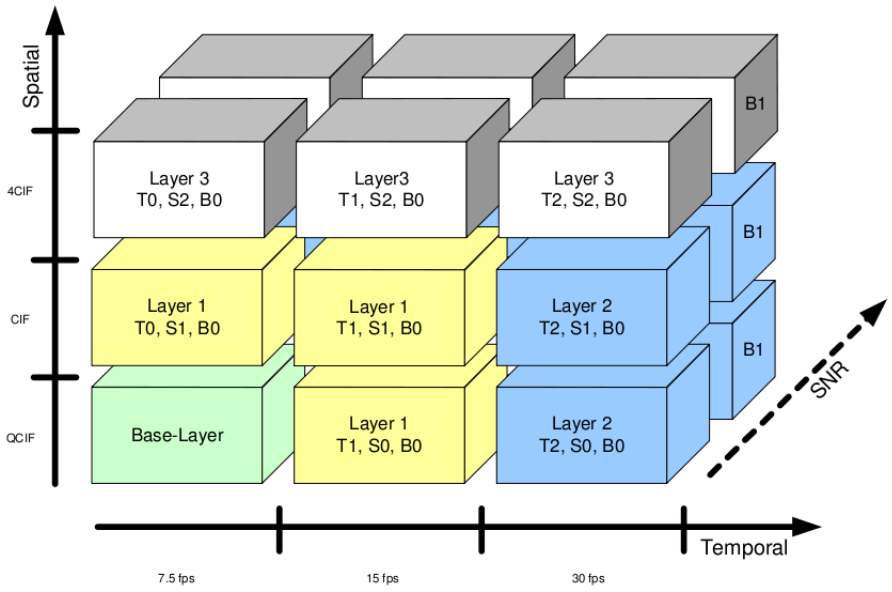


Fig. 1. Bitstream with spatial, temporal and quality layers.

encoded using both the I and P pictures of there patterns. Hence the order in which pictures are displayed is different from the order of the pictures encoding/decoding. 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. The constant interval between successive I pictures is the *intra period* of the pattern and the number of P pictures between two I is also constant and known as the *group of pictures*.

The bitstream is made of so-called access units (which contain one picture each). Each access unit is divided in NAL (Network Abstraction Layer) units of two types: *slices* (or Video Coding Layer units) which contain pixels related information, and non-VCL units which contain access unit structure related information, such as the slice parameters. The number, size and shape of the slices inside access units are defined at coding time and constant over all the bitstream. Slices are themselves divided in macro-blocks.

C. Decoding process

The present work only focuses on the decoder which is now detailed through an example. Let us decode a bitstream which contains a video with two resolution levels: 352x288 pixels and 704x576 pixels. Each resolution layer of the video contains two quantization layers (Q_p) 30 and 20 (recall that higher is the quantization step, lower is the quality). Therefore the bitstream contains 4 layers, sorted from the worst to the best quality:

- Layer 0: resolution of $352x288$ pixels and $Q_p = 30$,
- Layer 1: resolution of $352x288$ pixels and $Q_p = 20$,
- Layer 2: resolution of $704x576$ pixels and $Q_p = 30$,
- Layer 3: resolution of $704x576$ pixels and $Q_p = 20$.

The bitstream is necessarily decoded in this order, from the worst to the best quality layers. Switching between resolution layers is only possible when decoding I pictures, and the P and B pictures can only be decoded with the decoded resolution and quantization layers decoded for their I reference picture (but they can be decoded with a quantization lower than their I reference) [13]. The decoding process for each layer and for each slice follows the following steps, which are identical for all slices:

1) Initialization of the slice and decoding parameters.
2) Slice parsing, analysis of the bitstream and entropy decoding. Entropy decoding consists in converting the binary information in decimal coefficients corresponding to each macro-block.
3) Slice decoding: this step reconstructs the picture thanks to the decimal coefficients computed in the entropy decoding.
4) Optional final processing using a loop filter. This filter is applied to improve the final quality of the picture and is executed at the end of the total frame decoding process.

II. Feedback scheduling setup for video decoding

er 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 frames processing. At the lower level, decoding frames has basic deadlines related to video display, 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. These various control objectives and timing scales lead to define a hiera rchy of two control loops to manage the decoding quality. At high level, a ***QoS controller*** manages the application performance according to the available resources and end-user's requirements. At a lower level, the ***frame controller*** works at the pictures stream time scale and tightly cooperates with the ***processing speed controller*** integrated in each cluster. This latter controller is a third control loop needed for energy-performance management. From top (application software and long terms objectives) to bottom (silicon level and high control rate), the control hierarchy is as depicted in Figure 2:

Control design needs sensors to observe the controlled process and actuators to modify its state and output. 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 activation, parameter tuning, or processing suspend and 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 will be further used to formalize 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 [10]. Various control objectives

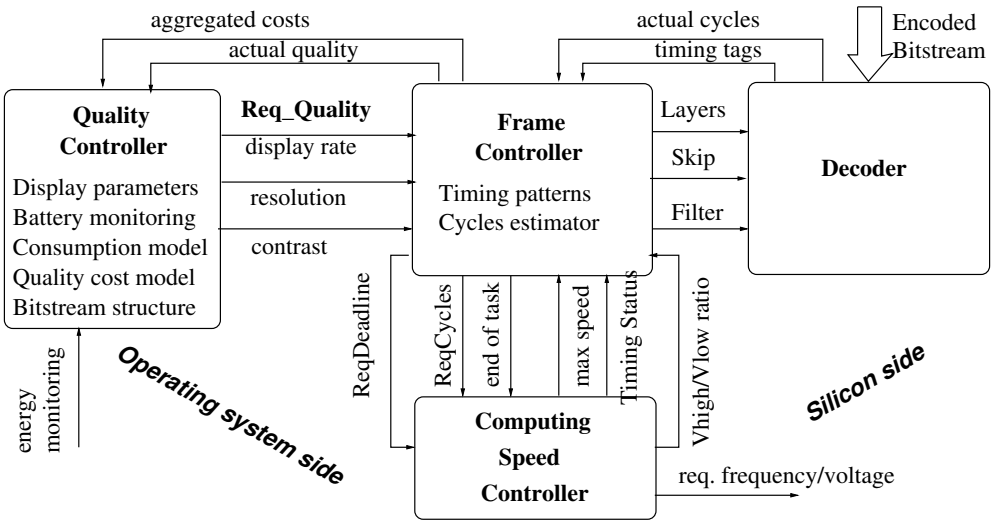


Fig. 2. Proposed control architecture of an embedded video decoding mechanism.

and timing scales lead to define a hierarchy of three control loops to manage the decoding quality as well as the energy consumption. From top (application software and long terms objectives) to bottom (silicon level and high control rate), the control hierarchy is depicted in Figure 2:

- The ***quality controller*** software runs in a master processor on top of the OS. It communicates with the clusters via an ANoC. 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 maximize 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.
- The ***frame controller*** software runs in one of the nodes of the considered cluster as a high priority task. Decoding frames has basic deadlines related to video display, 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.
- The ***computing speed controller*** is integrated in the cluster's silicon, together with the voltage and frequency controllers. It manages the energy-performance trade-off in each cluster.

These different controllers are finally detailed from bottom to top in the following subsections.

A. Control of the energy-performance trade-off

The computing speed controller aims at minimizing the energy consumption while ensuring some good computational performance. 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 needed computational load. The suggested control strategy (see [14] 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 Ω_i and the deadline Δ_i, provided by the frame controller, to process task execution T_i which decodes video frame i. Λ_i denotes the laxity of T_i, i.e the time to deadline of T_i.

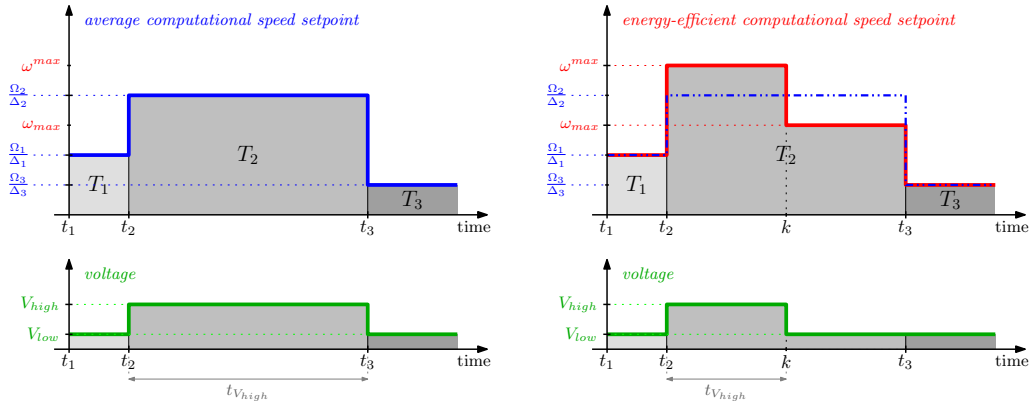
We consider here a voltage scalable device with two possible voltage values V_{high} and V_{low}. Let ω^{max} and ω_{max} respectively denote the maximal computational speeds when the system is running at V_{high} and V_{low} voltage. It follows that the high voltage level is necessary as soon as the average speed set-point of a task is higher than ω_{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 Ω_i/Δ_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 3(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 penalizing high supply voltage, as highlighted in Figure 3(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 3(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 ω^{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 ω_{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.

Indeed, the presence of deadlines and input constraints to compute repetitive tasks naturally leads to predictive control. Predictive control consist 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 optimization problem : for each image i to decode, find the computational speed set-point which minimizes 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



(a) Intuitive average speed set-point.

(b) Energy-efficient speed set-point.

Fig. 3. Different computational speed set-point buildings.

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 [15].

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* Ω_i and ***ii)*** *the remaining time to achieve it* Λ_i.

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

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

where Ω is the sum of the computational 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 needed Ω_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 [16]. In this case, the processor runs with the low voltage and a null frequency, which can be useful when an image is decoded before its deadline. Furthermore, in order to be robust to manufacturing uncertainties, the maximal speeds ω^{max} and ω_{max} which are required in the control decision are estimated using a weighted average of the measured speed, details can be found in [14].

The processing time $t_{V_{high}}$ at the highest voltage V_{high} is minimum since the decoding of an image always starts with the penalizing 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.

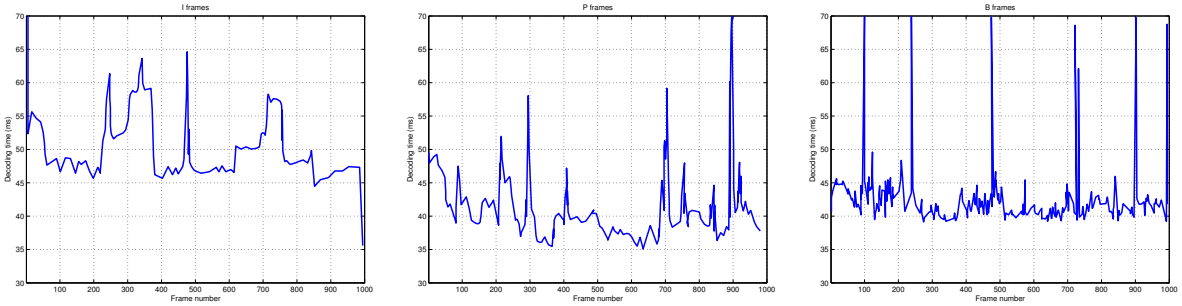
The stability of the control scheme follows the Lyapunov approach stating that, if an energy function of the system continuously decreases near an equilibrium point, then this point is a stable equilibrium for the system. Here we consider the decrease of the amount of instructions to be processed for 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.

B. Frame control

The second control loop feeds the computing speed controller with estimations 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, as previously explained in subsection II-A. This is developed in the sequel whereas a damping buffer is firstly introduced. It allows to miss some deadlines.

1) Frame buffer: 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 I-B. Therefore, the displayed flow is inevitably delayed w.r.t. the incoming bitstream. Following the ideas in [10], 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 varying 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 4, where the bitstream has a unique layer with 624×352 pixel resolution and quantization step $Q_p = 28$. 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.

Fig. 4. Decoding times for I, P and B frames.

2) Decoding execution time modeling: Additionally, the on-line observation of the decoding progress may help to anticipate for computing overloads and deadline misses. As a result, some tags can be inserted in the decoder to point out remarkable milestones. These tags are inserted at the transition between the structural steps of the video decoder previously described in subsection I-C. The distribution of decoding steps is very similar for each type of frame. Moreover, according to various experiments, the timing pattern associated to a particular slice position and to a particular type of frame seems to be stable enough to provide a usable on-the-fly estimation of the computing load needed by the frame being decoded and to anticipate for some control actions, such as filtering skips for instance.

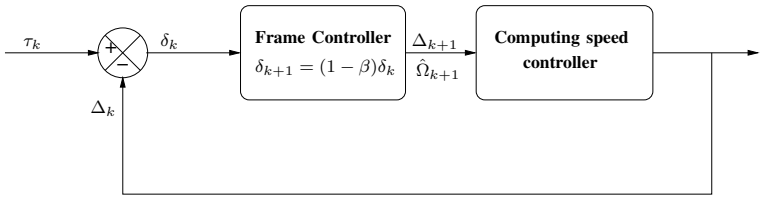


Fig. 5. Frame controller basics.

3) 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 [10] and to the measures plotted in Figure 4, an estimate of $\hat{\Omega}_{i+1}$ can be often taken as the last Ω_i reported by the computing speed controller, or by filtered values of the last executions, such that

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

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 Δ_{r_i} to their ideal value τ_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 is depicted in Figure 5. It 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 Ω) the successive values of δ results from an numerical suite which converges to 0 for $0 < \beta < 2$ (in practice $\beta \leq 1$ is chosen to avoid overshoots), larger is β faster is the decay of the deadline deviations δ, 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:

- 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.
- 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. In particular, the final filtering action can be skipped.
- 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.

C. Control of the Quality of Service

The latter controller manages long terms and user's defined goals. Informally, the control objective consists in trading-off the decoded bitstream quality and the energy storage lifetime, with an average energy consumption level

in mind. Quality parameters are expressed in terms of display requirements, e.g. high definition or standard mode, display rate and screen resolution. The end-user's may select different weights between these parameters, e.g. by imposing a high definition display whatever the cost, or asking for a mandatory lifetime before refill. Using a rough model between the quality layer and energy costs, this loop aims at setting the current requested quality level to be decoded. The on-line monitoring of the battery level and voltage decaying rate are fed back for on-line estimation and correction of this quality set-point. Thanks to the computing speed controller, the relations between the cluster's computational speed and the electrical power needed to feed the cluster can be approximated by monotonic functions. In other words, higher is the demanded computation burden higher is the energy consumption. It is expected that such monotonic cost functions lead to a simple control design, even if a formal statement for this control problem remains to be done.

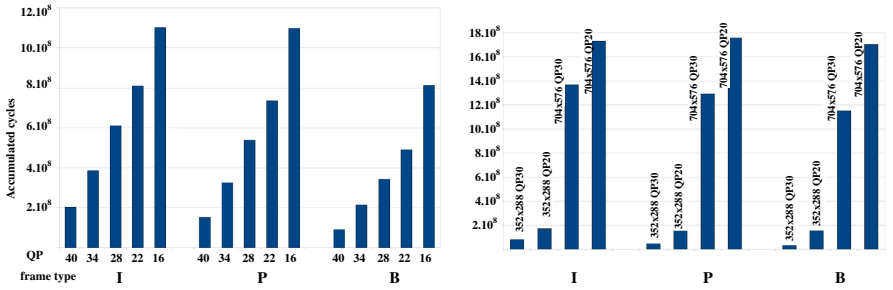
1) Quality layers and computing costs: After decoding the first frames, the structures of the bitstream (number of quantization/resolution layers, IPB structure, picture rate and slice map) are known and can be used to actually set the decoding parameters. Some of the video parameters are constrained by the incoming bitstream and by the display mode:

- The display rate constraints the average deadline for each picture (e.g. $40\,ms$ for standard TV rate).
- The quantization and resolution layers in the decoded bitstream must be encoded in the source stream.
- B and P frames cannot be decoded at a quality higher than the one of their reference I frame.

Switching on-line the display rate should be avoided as far as possible due to the visible effect on the display. Hence the usual choice for the variable decoding parameters, able to handle varying computing capabilities, is the requested quality layer to be decoded. A correct estimation of the quality set-point needs the knowledge of a model (cost function) to link up the quality layers and computing loads (and at least to understand their respective variations). Figure 6(a) plots the average number of cycles required to decode the five quantization layers of a particular bitstream. The quantization steps are here equidistant and set to $Q_p = 40$, 34, 28, 22, and 16 (the quantization step can vary between 0 and 51). The processor works at the fixed frequency $2.534\,$GHz. From left to right, the plot shows the average cycle number for I, P and B frames. One could note it is assumed that measures taken from a fixed frequency CPU provide a good image of the computing load in terms of statements to be executed. On the other hand, Figure 6(b) shows the number of cycles required for a bitstream made of a mix of resolution and quantization layers. The different parameters used for these experiments were already given in subsection I-C.

The results show that the choice of the quantization and/or resolution layers has a significant impact on the computational load, and that switching resolution has a rough influence while quantization may provide fine control. Therefore, decoding only the lower quality layers appears to be an effective actuator to reduce the decoding cost and the related energy consumption, and to manage the trade-off between the displayed quality and the energy consumption constraints. Other preliminary experiments, using an elementary controller to skip high quality quantization steps in case of overload, showed that switching between quantization layers has a very moderated impact on the viewer's perception, while efficiently saving execution cycles and avoiding deadline overshoots. Recall that the quality layers contained in the incoming bitstream must be decoded in sequence, with the low quality layers

first. Observing the latter experimental results in Figure 6, it appears that increasing the allocated computing load monotonically increases the decoding quality. Once again, this nice property is expected to help the design of the quality controller.



(a) Computation load for quantization layers.

(b) Computation load for combined resolution-quantization layers.

Fig. 6. Variation of the computation load w.r.t. quantization and/or resolution layers.

III. Experimental results

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

A. Implementation of the control architecture

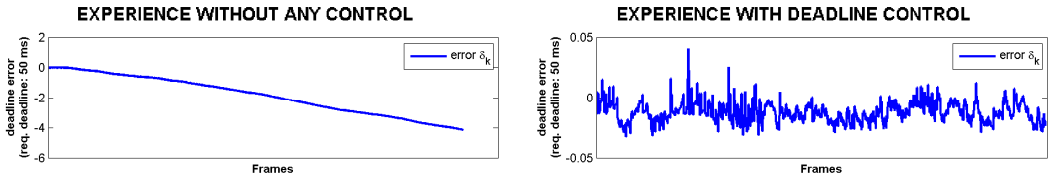
A prototype of a H.264 decoding controller has been developed under the Linux operating system. It runs on a stock Linux kernel tuned for enhancing its real-time capabilities, i.e. compiled with the tickles feature, high resolution timers and preemptive kernel options. In particular, these options ensure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

The tests have been executed on a stock Dell E6400 laptop, using a DuoCore2 processor. The testbed uses the Symmetric Multi-Processing (SMP) capability of the kernel to allocate specified parts of the software to dedicated CPUs, e.g. the controllers on one CPU and the decoder on the other. To emulate the speed controller (which is assumed to be integrated in silicon) the frequency scaling capabilities of the chipset have been used. At the end, the predictive controller of subsection II-A is implemented as a new module of the *cpufrequtils* package, allowing to control the CPU frequency between 800 MHz and 2.53 GHz in several steps. On the other hand, the frame controller and decoder are based on Posix *pthreads* and on 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 software. This version is far to be optimized (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 adapted to integrate the control features described in the previous sections.

B. Experimental results

The following decoding control experiments use a 1000-frame video with a IBBBPBBBI encoded in a structure. Due to the low performance of the JSVM decoder only low resolution ($624x352$ pixels) has been used. Two quality layers were considered, with or without post-processing filter. As the video was decoded on a single CPU, the flow was encoded with only one slice. The CPU frequency can be 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 maximal 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.

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. Figure 7 plots the evolution of the deadline overshoot δ for a requested frame deadline of $50\,ms$. In Figure 7(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 II-B3 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 7(b).



(a) Without any control.

(b) With deadline control.

Fig. 7. Deadline errors when keeping the CPU frequency constant.

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.

- In Figure 8, 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.
- In Figure 9, the speed controller introduced in subsection II-A 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.

- In Figure 10, both speed and frame controllers are active. Note that the frame controller was developed in section section:feedback,frame. In that case, the requested deadlines are always achieved, notably thanks to the QoS control and damping buffer. Indeed, the final filter is activated only when possible (the filter is activated and deactivated when the plot is 1 or 0 respectively in Figure 10). As expected, this filter is most often used for weak decoding constraints.

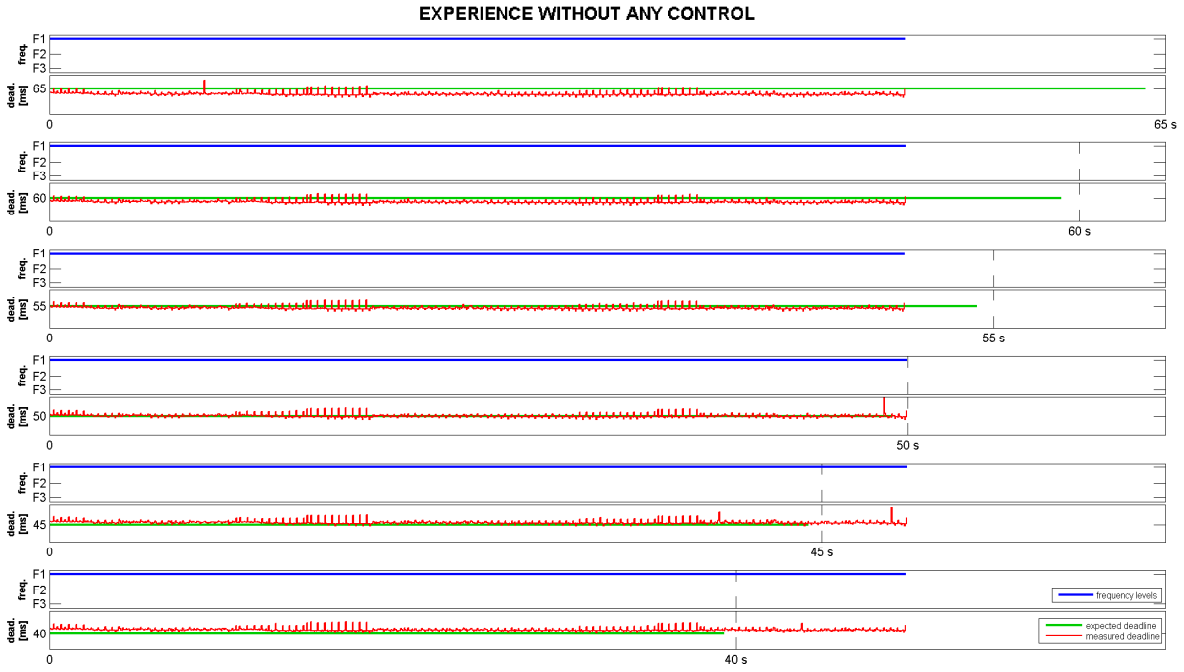


Fig. 8. Control over various requested deadlines without any feedback controller.

3) Energy consumption: The energy consumed for the various experiments could not be directly measured with the laptop used for the experiments. Nevertheless, we make some assumptions. First, we have 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 $2535\,\mathrm{MHz}$, the supply voltage is V.
- When running at $1600\,\mathrm{MHz}$, the supply voltage is $V/2$.
- When running at $800\,\mathrm{MHz}$, the supply voltage is $V/2$.

Moreover, we also assume that the energy used for decoding is proportional to $f_{clk} \times V_{dd}^2$, as explained in the introduction. Thus, the estimated energy consumption for a video sequence where the average frame decoding time (with the highest quality) is $50\,ms$ can be summarized in Table I (normalized for each requested deadline with the

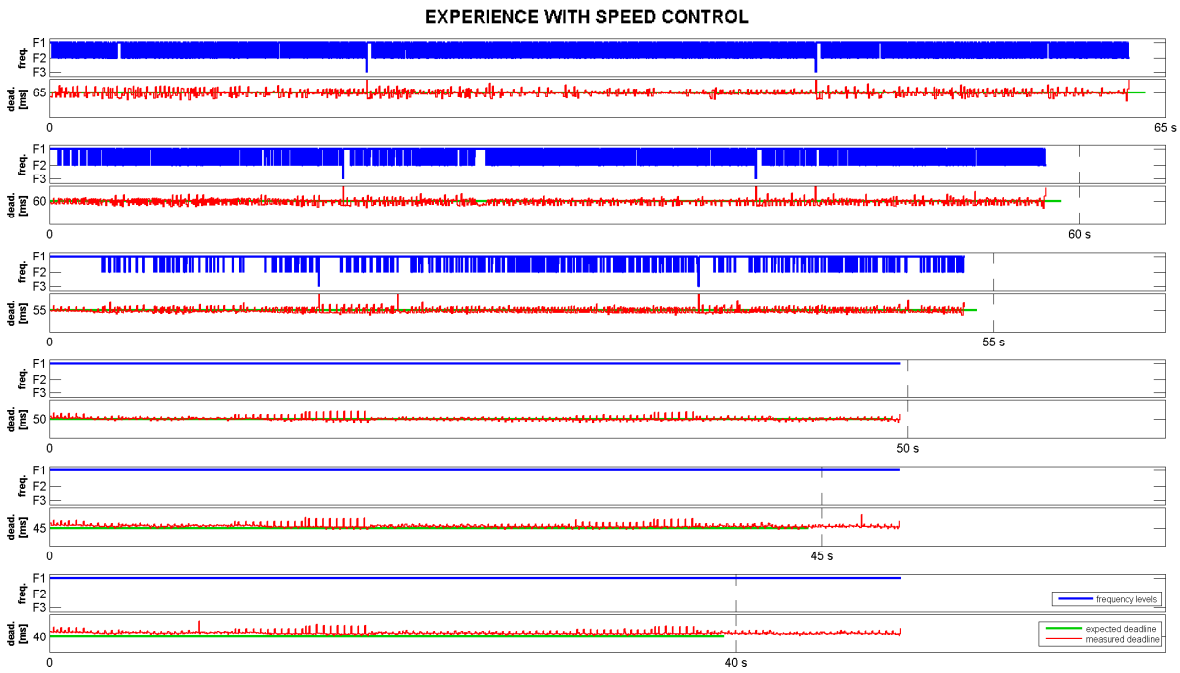


Fig. 9. Control over various requested deadlines with only speed control (the deadlines are constant).

consumption obtained without any control). These results call for several remarks:

- 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.
- 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 I

ENERGY CONSUMPTION ESTIMATION.

Deadline	**Normalized energy consumption**	
	computing speed control	computing speed & 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

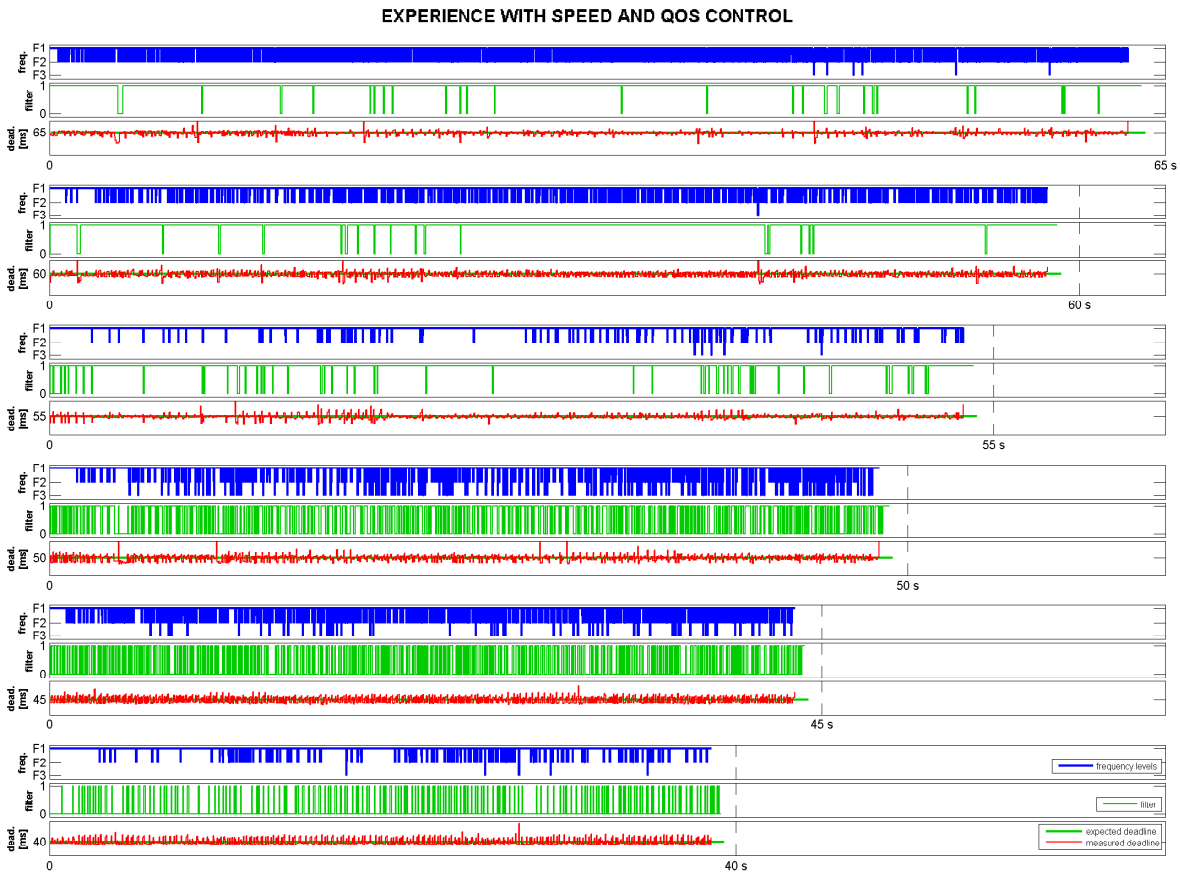


Fig. 10. Control over various requested deadlines with speed and QoS control (the computation load varies w.r.t. quantization and/or resolution layers).

4) Decoding quality degradation: Finally, a penalty criterion (borrowed from [10]) has been applied to assess for the decoding quality degradation due to stressed decoding conditions. The following penalties are thus applied:

- 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 11 only for the stressing deadlines (40, 45 and $50\,ms$), since longer deadlines are always achieved with no quality loss. Therefore, using simple and cheap feedback controllers allows to increase the decoding quality while highly decreasing the computing and energetic costs (compared with the uncontrolled case, where the deadline drift – see Figure 7 – finally induces quality level switches or even frame skips).

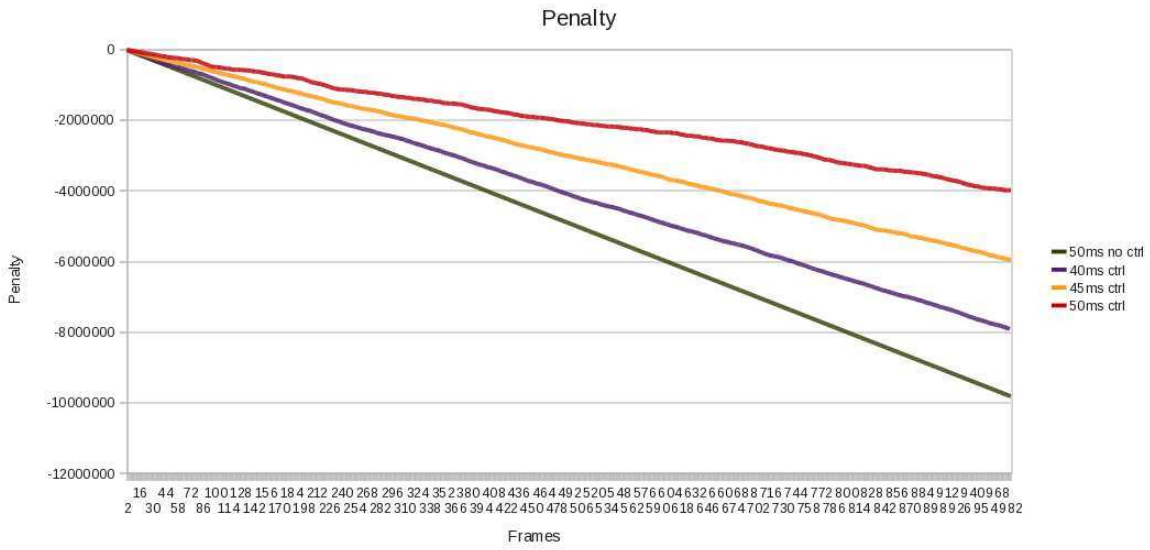


Fig. 11. Decoding quality.

CONCLUSION AND FUTURE WORK

In this paper a survey of different problems facing highly computing applications on embedded integrated chips was firstly presented. A closed-loop scheme clearly appeared as a solution in such systems and an architecture was hence proposed in this way for a particular case study, that is video decoding mechanism. The idea consists in nesting different controllers in different implementation levels, in order to be able to meet some end-user requirements in terms of quality of service and energy saving. The whole system finally runs minimizing the energy consumption while ensuring some good performance, that means the video to decode could be seen with a good resolution while the system consumes as less as possible. A practical validation of the proposal was realized on a system using only one processor and different power modes. The results clearly show that a reduction of about $25\,\%$ of energy saving can be achieved while the deadlines are ensured for most of frames to decode. Through this example, we hence demonstrated that feedback loops can highly enhance performance of a computing system. The next step is to implement this control scheme on a multicore platform and more fine-grain architectures.

ACKNOWLEDGMENTS

This research has been supported by the ARAVIS project, 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

[1] A.P. Chandrakasan and R.W. Brodersen. Minimizing power consumption in digital CMOS circuits. *Proceedings of the IEEE*, 83(4):498–523, 1995.

[2] K. Flautner, D. Flynn, D. Roberts, and D.I. Patel. An energy efficient soc with dynamic voltage scaling. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2004.

[3] A. Varma, B. Ganesh, M. Sen, S.R. Choudhury, L. Srinivasan, and J. Bruce. A control-theoretic approach to dynamic voltage scheduling. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2003.

[4] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings of the International Sympsonium on Low Power Electronics and Design*, 1998.

[5] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, 2001.

[6] B.F. Romanescu, M.E. Bauer, D.J. Sorin, and S Ozev. 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*, 2007.

[7] L. Fesquet and H. Zakaria. 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*, 2009.

[8] H. Zakaria, S. Durand, L. Fesquet, and N. Marchand. Integrated asynchronous regulation for nanometric technologies. In *VARI 2010: first European Workshops on CMOS Variability*, 2010.

[9] J.-C. Chiang, H.-F. Lo, and Lee. W.T. 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*, 2008.

[10] Clemens C. Wurst, Liesbeth Steffens, Reinder J. Verhaegh, Wim F.J.and Bril, and Christian Hentschel. Qos control strategies for high-quality video processing. *Real Time Systems*, 30(1), 2005.

[11] D. Marculescu and E. Talpes. Energy awareness and uncertainty in microarchitecture-level design. *IEEE Micro*, 25:64–76, 2005.

[12] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. on Circuits and Systems for Video Technology*, 13(7):560–576, 2003.

[13] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 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, 2007.

[14] S. Durand and N. Marchand. Fully discrete control scheme of the energy-performance tradeoff in embedded electronic devices. In *Proceedings of the 18th World Congress of IFAC*, 2011.

[15] M. Alamir. *Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes: A Parametrized Approach for Fast Systems*, volume 339. Springer-Verlag, 2006.

[16] W. Kuzmicz, E. Piwowarska, A. Pfitzner, and D. Kasprowicz. Static power consumption in nano-cmos circuits: Physics and modelling. In *Proceeding of the 14th International Conference Mixed Design of Integrated Circuits and Systems*, 2007.