Impact of Power-Management Granularity on The Energy-Quality Trade-off for Soft And Hard Real-Time Applications

Aleksandar Milutinović
University of Twente,
The Netherlands
Email: a.milutinovic@utwente.nl

Kees Goossens
NXP Semiconductors & Delft University of Technology
The Netherlands,
Email: kees.goossens@nxp.com

Gerard J.M. Smit
University of Twente,
The Netherlands,
Email: g.j.m.smit@utwente.nl

***Abstract*— In this paper we introduce the concepts of *work* of tokens (e.g. video frames) in an application, and *slack* arising from variations in work. Slack is used for dynamic voltage and frequency scaling in combination with a conservative power-management policy that never misses deadlines, for hard real-time applications, and with a non-conservative policy for soft real-time applications. We evaluate both policies for a number of different *granularities* (frequency of activation of the power manager) on an MPEG4 application, on *energy and quality* (deadline misses).**

We conclude that for soft real-time applications, there is a clear optimum in the energy, which depends on the work histogram of the application. The conservative policy has no deadline misses, and is only negligibly more expensive in terms of energy than the non-conservative policy. Finally, the granularity of both policies can be very coarse (128 frames) to reduce the power manager activation frequency, which has an insignificant energy cost.

I. Introduction and scope

Power management (here, energy minimisation) is imperative to increase the battery life time of nomadic devices such as mobile phones, but also for tethered devices such as set-top boxes to increase their life time e.g. through reduced thermal stress.

In this paper we perform an analytical study of slack (spare capacity) in a SOC, and how it can be used by several dynamic-voltage-and-frequency-based power-management policies. In addition, we vary the granularity (frequency) of power management. We consider the energy and quality (number of deadline misses) impact of the policies on soft and hard real-time applications, through an evaluation using an MPEG4 decoder mapped on an ARM processor.

In Section II, we introduce the applications of interest, work and slack, our energy model, and power management (policies). Section III introduces conservativeness of a policy (when it is safe to use for hard real-time applications) and its granularity. Section IV describes our experiments and their results, in particular the impact of the policies and their granularities on the energy-quality trade-off. After reviewing related work in Section V, we conclude in Section VI.

II. Model

A. Application model

In this paper we focus on power management of a single tile, consisting of a programmable *processor* with local memories and peripherals. Although our power management policies are compatible with multiple such tiles in a multi-processor SOC, we will not further consider inter-tile power management in the remainder. Each tile has its own frequency and voltage domain that can be set independently to a voltage-frequency *operating point* at run time. The benefits and costs of scaling are discussed below.

We consider soft and hard *real-time streaming applications*. In general, such applications operate on sequences of tokens that each have a deadline by which they should be produced. Hard real-time applications do not allow deadline misses (late production of tokens), whereas soft real-time applications allow a limited number of deadline misses, but at the cost of a quality degradation. In our case, tokens are compressed video frames, and the deadlines define when they should be displayed. The frame rate f_{FR} determines the regular spacing $T = 1/f_{FR}$ of deadlines in time. We assume that the input data and output space of the application are always available. In Section IV we comment on the buffer utilisation within the application.

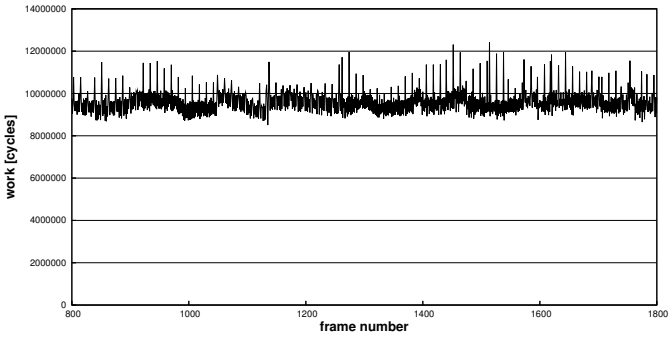

Fig. 1. Work per frame for part of an MPEG4 sequence.

B. Work and slack

The *work* w_i of a frame i is the number of processor cycles required to fetch, process, and store it. The *total work* of a sequence of frames is the sum of work of the individual frames. We assume that work depends only on the input token(s), and is independent of the operating point of the processor. This holds when the input and output tokens of task, as well as its instructions, are stored in the local memories of the tile [1].

Work for different input tokens may vary, e.g. the work for a frame depends on the complexity of its decoding, which strongly depends on whether it is an MPEG I or P frame. I frames require considerably more work than B and P frames, as we shall see later. The *worst-case work* of a sequence of frames is $wcw = Max_{j=0}^{\infty} w_j$. The time to finish the work of frame i at a frequency f_i is the *actual-case execution time* $acet_i = w_i/f_i$. Figure 1 shows work per frame of part of an MPEG4 sequency, further discussed in Section IV.

In order not to miss any deadline, f_i should be high enough. In fact, there are two different kinds of deadlines. A *relative deadline* requires that the actual execution time of a frame i is less than required by frame rate. With a regular execution, it must be less than the frame rate: $acet_i \leq T = 1/f_{FR}$. *Relative slack* r is the difference

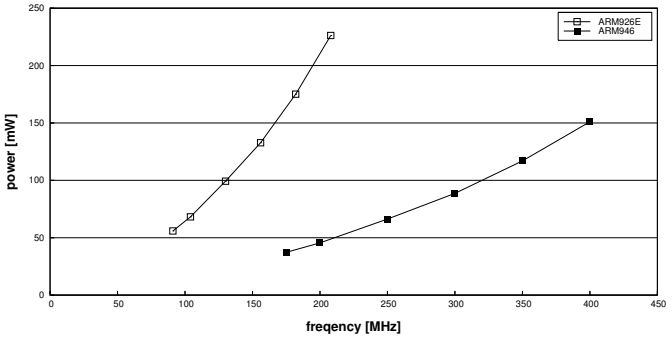

Fig. 2. Power Model: a) ARM 926E, b) ARM 946.

of the worst-case and the actual-case execution time of frame i at the maximum operating frequency f_{Max}: $r_i = T - acet_i$. The *absolute deadline* of a frame f_i is the absolute time at which it must be produced (displayed). The *absolute slack* is defined correspondingly: $s_i = (i+1)T - \sum_{j=0}^{i} acet_j$.

When a deadline is not met, it is a *miss*. There are fewer relative than absolute misses because a single hard frame can cause several successive frames to miss their absolute deadlines, even though they do not miss their relative deadlines. We focus on absolute deadlines in the sequel, because they are important for the user (e.g. frame rate), and are harder to ensure.

C. Energy model

In common with many other power management strategies, we use slack to reduce the operating point (frequency and voltage) of the processor, and thus save energy. In this paper we assume that the process technology used is optimised to minimise leakage, and we can only affect the *dynamic energy*, which is dominant in SOCs.

Dynamic *power* is given by $P_{dyn} = \alpha C V^2 f = \alpha C V^2 w/t$, where α is the switching activity, C is the switched capacitance, and V and f define the voltage-frequency operating point. Alternatively, w is the work performed in time t. The *energy* spent is then $E_{dyn} = P_{dyn} t = \alpha C V^2 w$. To minimise energy, the voltage must be scaled to the lowest value supporting the frequency required to meet a deadline.

A processor can run at a minimum (maximum) frequency f_{Min} (f_{Max}), requiring a minimum (maximum) voltage $v(f_{Min})$ ($v(f_{Max})$). The power model $P(f)$ used in this paper, related to the function $v(f)$ is computed as follows. Our starting point is frequency-power measurements of an ARM926E board, which are shown in Figure 2. The ARM926E is not powerful enough to execute our application in real time. The ARM946 is, but the public data on its power characteristics are insufficient. For this reason, we correlated the maximum operating point (peak power) of the ARM926E with the maximum operating point of an ARM946 as given in [2].

D. Dynamic power management and policies

We assume that the SOC under consideration has been dimensioned at design time to minimise the energy consumption, and focus on dynamic power management. Dynamic voltage and frequency scaling (DVFS) power management defines voltage-frequency operating points at run time according to a *policy* to trade processor performance for energy. A *transition* occurs whenever the operating point is changed, to increase the performance and energy, or decrease them, as required.

Dynamic power management, and DVFS in particular, has several costs, in terms of area of the DVFS infrastructure, and reduced processor performance (assuming the policy is implemented in software). Both result in a power cost. In some implementations, processors must be idle during transitions, again lowering the processor performance. Additionally, it takes time to change the voltage of a processor due to its capacitance, which means that transitions are not instantaneous, resulting a minor loss of performance or energy. Our experiments take these costs into account, as described in Section IV.

III. Policy conservativeness and granularity

A policy is *conservative* if it does not introduce any deadline misses (i.e. lowers the quality of the result) compared to operating at f_{Max}, and non-conservative otherwise. Conservative policies are required for hard real-time applications, whereas soft real-time applications can tolerate occasional deadline misses and could use non-conservative policies. We use DVFS with policies based on run-time observation of already available slack or prediction of future slack to reduce energy. The *proven slack* of a frame i is the cumulative slack of the frames before it, i.e. the absolute slack of frame $i-1$. Proven slack can be detected at run time. There may be future slack that is unproven at the start of frame i's work. We will use a (hypothetical) *perfect predictor* to compute future unproven slack for frame i before it is executed. This is a useful baseline for later comparisons because no real predictors can do any better.

We define the *granularity* of a policy as the shortest time between successive transitions. The aim of this paper is to investigate the impact of the granularity of the policy on the energy reduction, taking into account the transition overhead. We will scale the granularity from 1 frame to the length of the entire sequence of frames. Given a certain granularity N we use two policies: *perfect predictor* and *proven slack*. The former accurately predicts the cumulative amount of work of the next N frames and scales the performance of the processor to the average frequency for those frames ($f_{avg_i} = (\Sigma_{j=0}^{N-1} w_{i*N+j})/(NT)$ for group i). In other words, the last of the N frames will never miss its deadline, but preceding frames might. For $N > 1$ this policy is not conservative, therefore allowed for soft real-time applications, but not for hard real-time applications. The proven-slack policy assumes that the next N frames all require the worst-case work, but uses all the proven slack of previous group to reduce the frequency of the processor (but never scaling below f_{Min}): $f_{max_i} = (NMax_{j=0}^{\infty} w_j)/(NT + s_{i-1})$ for group i. The first N frames have no proven slack, and hence run at the frequency required for the worst-case of the entire sequence. Hence, all frames will meet their deadline, and this policy is always conservative, and suitable for both soft and hard real-time applications.

For $N = \infty$ (i.e. the length of the input sequence), the processor operates at f_{avg} with the perfect-predictor policy, and at f_{max} using the proven-slack policy. The former is the traditional minimum energy operating point, when deadlines are ignored (running a best-effort application at its average requirement). The latter is our baseline (no misses, no power management, no overhead) for later comparisons. The final point of interest is the lowest frequency f_{min}, the maximal frequency at which all deadlines are missed. When varying the granularity N for both policies, the total energy will vary, as will the number of misses. This is an instance of an *energy-quality* or *cost-performance* plot. In the results section we shall present this plot, and draw some conclusions on the relative performance of the policies.

When $N > 1$, frames may be produced early, in which case they are stored in memory. We do not increase the energy for the policy because only the time at which data (frames) are produced changes, but not their number (or size). Hence the energy consumed by the interconnect between processor and memories is unchanged.

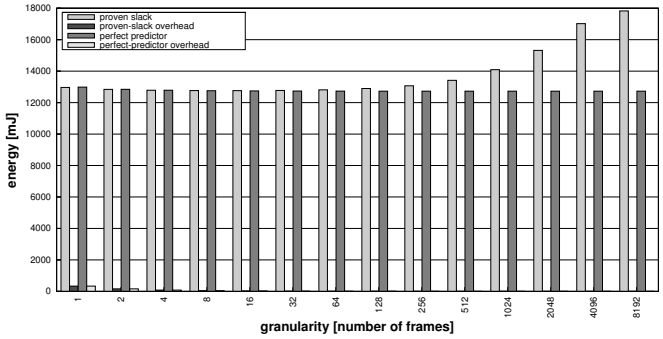

Fig. 3. Application & power-management energies, for different granularities.

Regarding the impact on memories, we assume that data is stored in sufficiently large buffers in either SRAM or DRAM. For DRAM we assume that unused banks are not switched off. As a result, the buffer filling (due to early or late data production) does not change the energy consumed by the memory.

IV. Experimental results

Our application is an MPEG4 decoder running on an ARM946 running at 86 MHz. It decodes an input stream of 207 seconds, with I and P frames, at 25 frames per second, and a resolution of 176x144 pixels. The measured number of cycles per frame were measured, and define the work per frame. On the basis of this, we analytically evaluate our two policies for a number of different granularities. A transition results in a 20 μsec inactive period, and a 1 msec execution of the power manager, with the associated energy cost.

Figure 3 shows the energy consumed by the application and the power manager for different granularities, for both policies. The rightmost bar corresponds to the entire sequence, i.e. f_{avg} for the perfect-predictor policy and f_{max} for the proven-slack policy. The energy savings w.r.t. operating at f_{max} are around 30% for 1-128 frames, at a cost of 2% for the power manager. Above 128 frames the proven-slack policy uses linearly more energy. The energy used by the perfect-predictor policy decreases lightly, but at the cost of increasingly missing deadlines, as we shall see below.

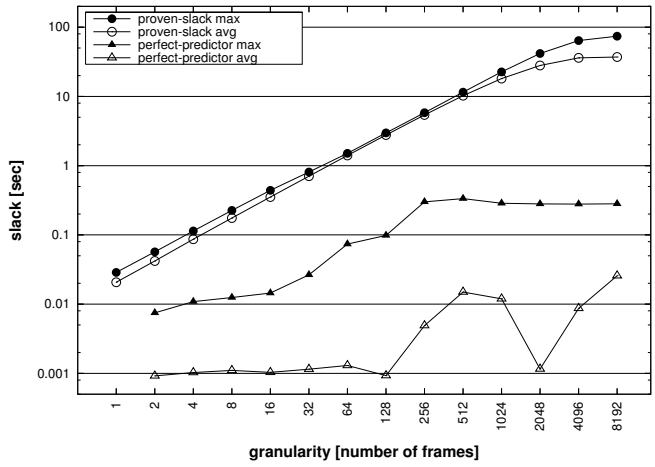

Fig. 4. Remained slack versus granularity.

Figure 4 shows the average slack ($(\sum_{i=0}^{S-1} s_i)/S$, for a sequence of S frames) and worst-case slack ($Max_{i=0}^{S-1} s_i$) for different granularities, for both policies. For the proven-slack policy the average slack saturates at the difference between worst-case work and average-case work, while maximum slack keeps increasing. This indicates that the policy cannot always exploit the accumulated slack (e.g. because the processor cannot be scaled below f_{Min}), which therefore reaches large values. The perfect-predictor policy, on the other hand, uses slack effectively because it never accumulates to large values.

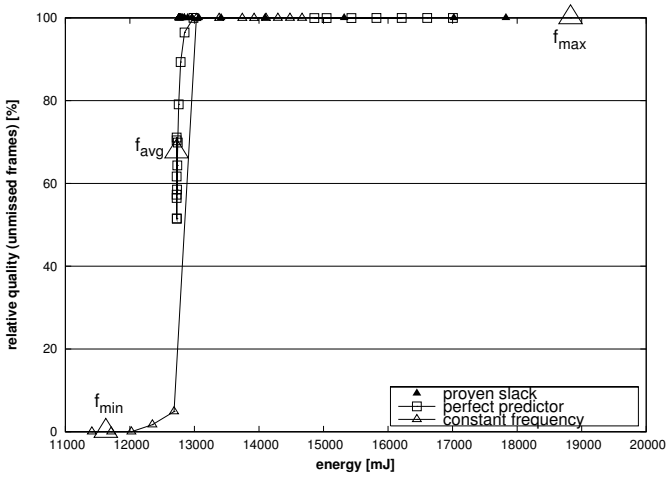

Fig. 5. Energy-quality trade-off for different policies and granularities.

However, the perfect-predictor policy obtains its lower energy at the cost of deadline misses. Figure 5 shows the energy versus the relative quality (number of frames that are produced on time). Both policies are plotted for various granularities. The three reference points are also indicated: the baseline f_{max} (always running at the frequency of the worst-case frame, i.e. no, misses, no power management, no overhead), f_{min} (the maximal frequency at which all deadlines are missed), f_{avg} (the frequency for minimum energy, when only the global deadline is met). The *constant-frequency* policy, which has no power management (overhead), connects these three points.

Note that the proven-slack policy is indeed conservative, because it provides 100% quality (no misses) for all energies (operating points). The perfect predictor, however, starts missing deadlines around 13000 mJ, and drops to 0% quality (f_{min}) at 11600 mJ. The transition from good to bad takes place in a very narrow band: a 95% quality improvement costs only 3% additional energy. This is positive, because it is clear where the optimum is (13000 mJ). In fact, this results from the distribution of work, as shown in the work histogram in Figure 6. Many frames can be processed in the range of 240-250 MHz, with a relatively small number of much larger frames. Hence the operating point for soft real-time applications can be close to the transition, but hard real-time applications must have a much higher operating frequency, when using the perfect-predictor policy.

For this reason, we compare the perfect-predictor and proven-slack policies in the transition range, shown in more detail in Figure 7. The 1-frame proven-slack policy is conservative, i.e. offers 100% quality, and uses only 0.3% more energy than the perfect predictor. Furthermore, increasing the granularity from 1 to 128 increases the energy of the proven-slack policy by only 2%. This is a positive result because it allows the power manager to run very infrequently, lowering its overhead.

Figure 8 shows that the buffer filling increases linearly with the granularity. As argued in Section III, however, storing data early does not cost any extra energy. If buffers are smaller than shown or data are always available at the input, then the application will stall, and restart when data arrive, like the race-to-idle policy. The result is a

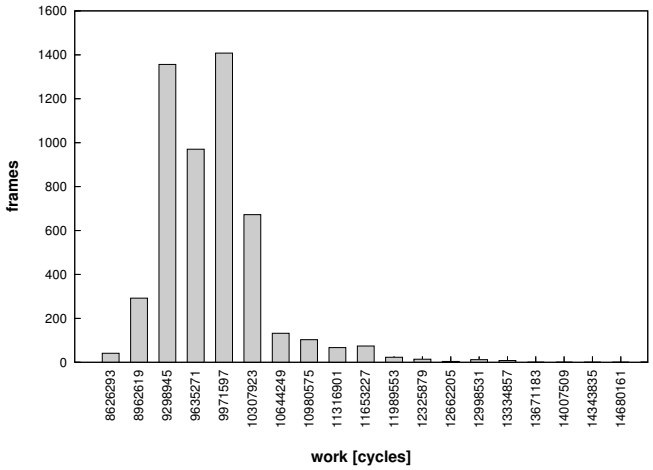

Fig. 6. Work histogram.

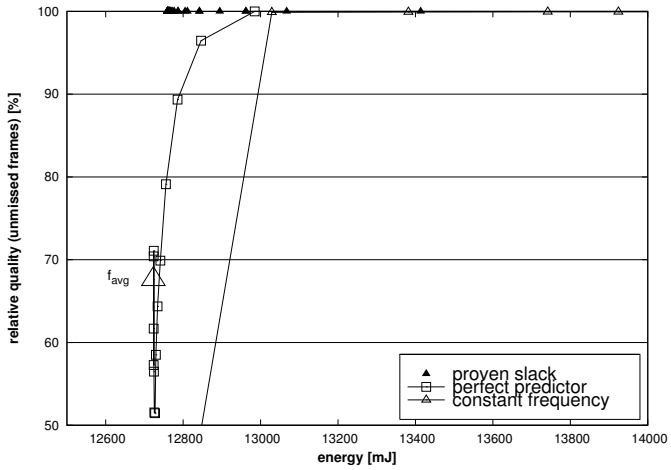

Fig. 7. Energy-quality trade-off for different policies and granularities.

conservative policy, although less energy efficient.

V. Related work

The speed of DVFS infrastructure is increasing [3], enabling power management at very fine granularity. This was the motivation for the study that we present. Azavedo [4] uses the compiler to place the checkpoints in program code at the boundaries of basic blocks, which represents fine granularity solution that uses variable granularity but in a limited range. AbouGhazaleh [5] presents the collaboration between compiler and operating system and by inserting instrumentation code into the program code to achieve to vary the granularity. The same authors propose theoretical solution for choosing the optimal granularity in [6]. Choi [7] presents DVFS technique for an MPEG decoder with sub-frame granularity by differentiating between invariable and variable parts of a decoder.

VI. Conclusions

In this paper we introduce the concepts of *work* of tokens in an application, and the difference between the worst case and actual case work (*slack*). We use slack for dynamic voltage and frequency scaling in combination with two policies: the perfect predictor and proven slack. The *proven-slack policy is conservative*, which means that it never misses deadlines (late completion of work), as required for hard real-time applications. The *perfect-predictor* policy, on the other hand, may occasionally miss a deadline, which is allowed for soft real-time applications. Both policies have been evaluated on an MPEG4 application, for a number of different *granularities*, i.e. frequency of operating point changes (power manager activations).

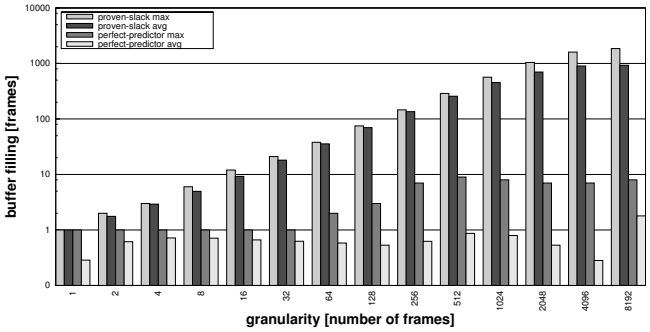

Fig. 8. Buffer fillings for different policies and granularities.

From the experiments we draw the following conclusions. 1) A long tail in the work distribution results in a steep quality improvement: from almost 0% to almost 100% at an additional energy cost of only 3%. This means that for soft real-time applications, there is a *clear optimum in the energy-quality trade-off*. The work distribution of many applications exhibits such a long tail, which is why hard real-time guarantees are usually considered expensive to offer. 2) The proven-slack policy offers 100% quality at only 0.3% more energy than the perfect-predictor policy, which is theoretical upper bound and hard to achieve in practice. Hence, *conservative power management for hard real-time guarantees, is only negligibly more expensive* than non-conservative power management for soft real-time guarantees. 3) The energy of the policies increases by only 2% when increasing the granularity to 128 frames. Hence, the *power manager can run very infrequently, at an insignificant energy cost*.

Acknowledgements

We would like to thank Albert Molderink of the University of Twente for the MPEG4 traces on which this work is based.

References

[1] M. Bekooij, O. Moreira, P. Poplavko, B. Mesman, M. Pastrnak, and J. van Meerbergen, "Predictable embedded multiprocessor system design," *Lecture notes in computer science*, vol. 3199, pp. 77–91, 2004.

[2] [Online]. Available: http://www.arm.com/products/CPUs/ARM946E-S.html

[3] M. Meijer, J. Pineda de Gyvez, and R. Otten, "On-chip digital power supply control for system-on-chip applications," in *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*. New York: ACM Press, 2005, pp. 311–314.

[4] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, and A. Nicolau, "Profile-based dynamic voltage scheduling using program checkpoints," *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, pp. 168–175, 2002.

[5] N. AbouGhazaleh, D. Mossé, B. Childers, and R. Melhem, "Collaborative operating system and compiler power management for real-time applications," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 5, no. 1, pp. 82–115, 2006.

[6] N. AbouGhazaleh, D. Mosse, B. Childers, and R. Melhem, "Toward the placement of power management points in real-time applications," *Compilers and operating systems for low power table of contents*, pp. 37–52, 2003.

[7] K. Choi, K. Dantu, W. Cheng, and M. Pedram, "Frame-based dynamic voltage and frequency scaling for a MPEG decoder," *Digest of technical papers- IEEE/ACM International Conference on Computer-Aided Design*, pp. 732–737, 2002.