Dynamic Workload Peak Detection for Slack Management

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 an analytical study on dynamism and possibilities on slack exploitation by dynamic power management is presented. We introduce a specific workload decomposition method for work required for (streaming) application processing data tokens (e.g. video frames) with work behaviour patterns as a mix of periodic and aperiodic patterns. It offers efficient and computationally light method for speculation on considerable work variations and its exploitation in energy saving techniques. It is used by a dynamic power management policy which has low overhead and reduces both requirements for buffering space, and deadline misses (increase QoS). We evaluate our policy in experiments on MPEG4 decoding of several different input sequences and present results.**

I. Introduction

One of the main imperatives in design of nomadic devices such as mobile phones is to increase the battery life time. Low power is also important for tethered devices such as set-top boxes to increase their life time e.g. through reduced thermal stress. This demands effective power management to lower both the power and energy consumption. In a previous paper [1] we performed an analytical study of slack (spare capacity) in a SOC, and how it can be used by dynamic-voltage-and-frequency-based power-management policies. We varied the granularity (frequency) of power management and observed the energy and quality (number of deadline misses) impact of the policies on soft and hard real-time applications. In this paper we extend that work with an adaptive power-management policy. We observe the workload of an application as a number of cycles required to process of input data tokens (e.g. video frames) for several input sequences. The workload varies and the difference between worst and average work is considerable (Figure 1). We distinguish fast and slow changes in workload, as well as frames that differ significantly from the average case (peaks). Energy-optimal power management would schedule an averaged frequency for as many frames as possible, and the peak frames are limitations to averaging. In general, averaging over longer periods increases the number of misses or/and requires larger buffers. We propose a speculative policy with peak and phase detector that utilizes and exploits patterns in workload to save power and reduce buffer space. We dynamically detect periodic and aperiodic modes in workload variation. Also, peak amplitude and their variable inter-arrival distances are dynamically detected. Based on this, we apply adaptive power management by reducing operating voltage and frequency. On top of that, we pro-actively generate slack as an additional slack margin to reduce number of misses that might happen due to this speculative (non-conservative) approach.

In Section II, we define the architecture and applications in scope, as well as energy model and power management and concepts of work and slack. In Section III, we give an extended characterization of application workload. Section IV discusses the theoretical energy-optimal policy, followed by Section V with the description of our peak and phase detection and the resulting policy in Section VI. Section VII describes experiments and discusses the results. After reviewing related work in Section VIII, we conclude the paper in Section IX.

II. Scope

A. System architecture and 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 [2], 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.

We consider *soft real-time streaming applications*. In general, such applications operate on sequences of tokens that each has a deadline by which they should be processed. Soft real-time applications allow a limited number of deadline misses, but at the cost of an undesirable quality degradation. In our case, tokens are compressed video frames, and the deadlines define when they should be uncompressed and 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 VII we present the buffer utilisation within the application and the benefits of our approach. By assuming no constraints on input data arrival times and output buffers sizes, we exclude an important part of the slack, which is a result of any irregularity in input/output operations. The reason for this is because it would be out of the scope of this paper, since it is not observable within a single tile, but rather on the system as a whole. It is actually considerable part of the overall slack in today's SoC with can be exploited for power management as presented in [2].

B. Energy model and power management

In common with many other power management strategies, we use slack, i.e. unused capacity, 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 power, and we can only affect the *dynamic energy*, which is dominant in a SOC while it operates.

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, work w is the number of cycles executed 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 $v(f_{Min})$ ($v(f_{Max})$). Further details of the used energy model can be found in our previous work [1].

Dynamic voltage and frequency scaling (DVFS) utilizes number of voltage-frequency operating points through power modes at run time

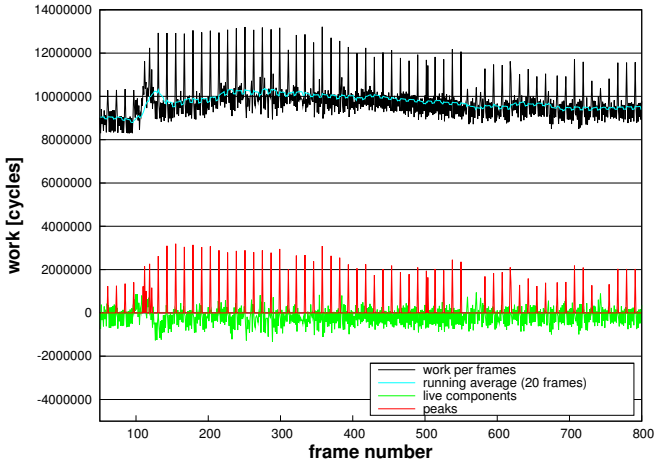

Fig. 1. MPEG4 sequence decoding (part): work per frame and its components.

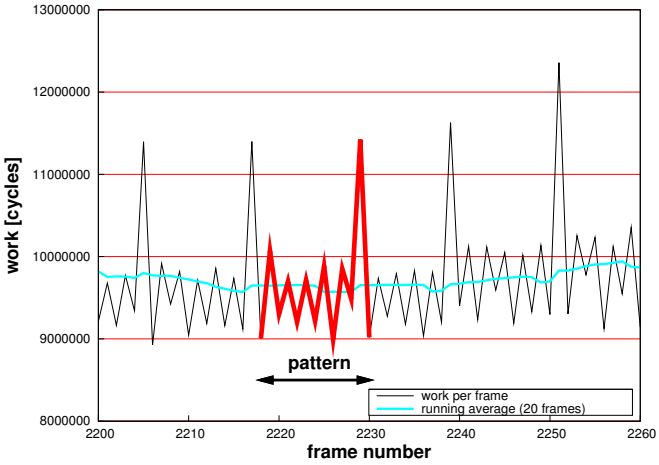

Fig. 2. Work per frame (detail of Figure 1) with repeating pattern.

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. We define the *granularity* of a policy as the shortest time between successive transitions.

C. Work and slack concepts

The *work* w_i of a frame i is the number of processor cycles required to fetch, process, and store it. We assume that work depends only on the input token(s), and that is independent of the operating point of the processor. This holds when the input and output tokens of a task, as well as its instructions, are stored in the local memories of the tile. The application should also not be affected by other applications, which holds in systems such as CompSOC [3].

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* $acet_i = w_i/f_i$. In order not to miss any deadline it must be less than the frame rate: $acet_i \leq T = 1/f_{FR}$. The *absolute deadline* of a frame f_i is the absolute time at which it must be produced (displayed). The *absolute slack* is defined by: $s_i = (i+1)T - \sum_{j=0}^{i} acet_j$. When a deadline is not met, it is a *miss*. In a previous paper we introduced the *perfect-predictor policy* that perfectly speculates on the work of future frames and schedules the average frequency for them, and *proven-slack policy* that uses only available proven slack and assumes the worst-case work for each of coming frames.

III. Workload Characterization

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 many factors, for example on whether it is an MPEG4 I or P frame. I frames require considerably more work than B and P frames. The perfect-predictor policy sets the frequency equal to the cumulative work of a group of frames divided by the allotted time (T times group size). This ensures that the last frame in the group finishes on time. However, all other frames could potentially miss their deadlines. In general, a frame with more work than the average work will result in a negative slack, thus in a miss, and a frame with work less than the average will contribute in a positive slack. But the order within a group, as well as work amount, also impacts the number of misses. The worst case is when the frames in a group have *decreasing* work,

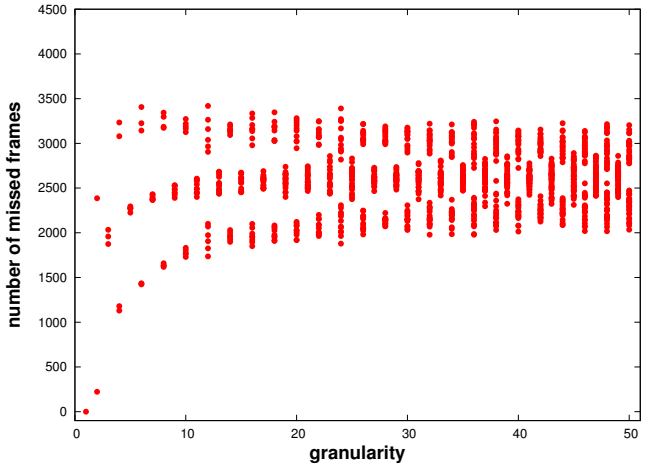

Fig. 3. Number of misses vs. granularity from 1 to 50 for different phases.

in which case all but the last frame miss their deadline. The quality is then $1/(N-1)$, for a group length (granularity) of N. The opposite, non-decreasing order is the best case, with no missed deadlines and a maximum quality of 1. In practice, it is usually a mix (Figure 2).

Based on these observations, the key idea of this paper is to determine groups (granularities) of the right size. Additionally, the phase of groups must be selected such that, ideally, the work of successive frames in a group increases. In this way, we maximise the quality.

To confirm our intuition, we did experiments using the perfect-prediction policy for the granularity range from 1 to 50. The number of misses increases with the granularity (converging to the number of misses when the entire sequence is run at the average frequency). There are two different trends for even and for odd granularities, explained below. This experiment was repeated with varying starting phase (offset of the first transition): $0, 1, ...N-1$ for the granularity N. The result is shown in Figure 3. The difference between adjacent odd and even phases can be up to 50% of the total number of frames. To improve clarity of illustration, Figure 4 shows results only for granularities from 1 to 15 and the points of the same granularity are connected with a line. The conclusion is that the granularity and phase of the power management policy both influences the number of misses. This means that in work per frame distribution there has

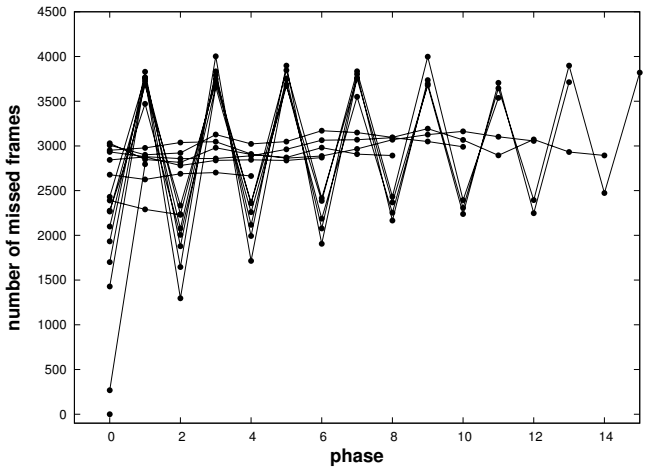

Fig. 4. Number of misses vs. phase for a granularity range 1-15.

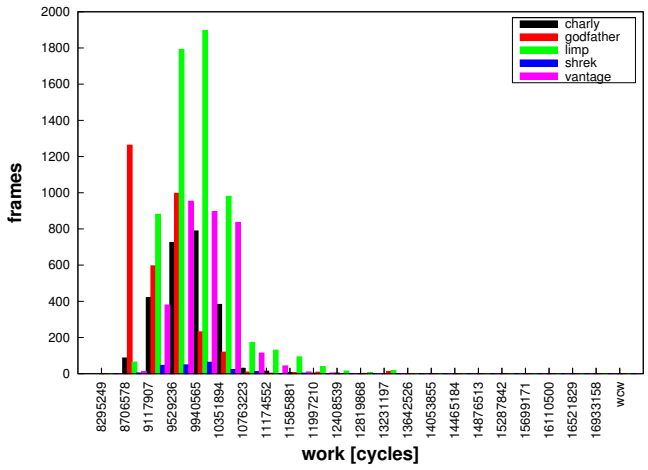

Fig. 5. Work histogram for 5 different input sequences.

a certain periodicity.

According to the traditional spectral analysis, work per frames can be decomposed into components consisting of low and high frequencies. The first, we denote as *running average* since it is calculated as running average over sufficiently long number of frames and, the second, we denote as *alternating component* calculated as a difference from the original amount of work and the running average. Successive samples of the alternating component are comparable in work one to the others, but with changed sign for almost the whole sequence. Also, the average value of the alternating component is small compared to running average, so almost every two consecutive samples are averaged in pairs. However, alternating component contains *peaks* that are much bigger than the rest. An example of work per frame decomposed in such a way is shown by the running average, alternating component, and peaks in Figure 1.

Observation of the work over time suggests that peaks appear with a certain periodicity. This periodical behaviour, in case it is regular, could be beneficial to system design. In particular, they could be the basis for groups of frames that we assign the same operating point and phase. For that reason, we focus our approach on determining a reliable speculation on work of future frames to exploit the described features of work signal for energy saving purposes.

Depending on work variation, we can distinguish following cases of work over time a) slowly changing, when the running average is the dominant component with almost no peaks; b) fast changing, when the alternating component with peaks is dominant; and c) hybrid, when on top of the running average there are peaks, distributed in a regular or irregular way. In this paper, we focus on a hybrid case with the intention to find peaks, which allows the determination of periods with the right phase.

The traditional and most common approach in system dimensioning would preferably follow the running average component as a requirement and then provision enough system resources to satisfy this trend. That works well unless there are peak frames that would excess provisioned resources and therefore results in deadline misses. However, there are methods for improvement, as for example, resource over-provisioning, or frames skipping or dropping. They always come at a certain cost, in extra energy spent while generating so-called over-provisioning slack, or require more space for buffering the results. Even worse, conservative approaches for the system design will rely on *wcw* in order to guaranty that all deadlines will be met. This results in even more expensive design.

Another approach would be to speculate on the work for future frames and set the operating point accordingly. On Figure 1, it can be seen that major part of the frames are close to the average work, and just some of them are considerably bigger than that trend. Almost all of them are much lower than the *wcw* of the whole trace. The distribution of work of frames of 5 different sequences is illustrated in Figure 5, showing that the majority of frames are concentrated in the lower bins of the histogram, much lower than *wcw*. This can lead to slack, which in turn can be used to save energy.

IV. Energy-optimal policy and its applicability

Yao et al. [4] introduce an off-line algorithm to determine the energy-optimal DVFS schedule for a set of real-time tasks, assuming active power model as a convex function of processor frequency. Applying that algorithm to a single task will result in a single average frequency equal to task's work divided by the duration of time until its deadline. For a set of tasks it gives a schedule that always has the average frequency for processing work over the time intervals bounded by critical deadlines that limit further averaging. Part of transition overhead is reduced by this policy too, since the number of transitions is minimised.

Streaming applications like MPEG4 have equidistant deadlines, and a convenient assumption would be that the moments when input data for frames are also equidistant. This holds, in general for the streams that require the same order of processing and displaying. This could be ensured by an additional initial delay from the arrival moment to the moment processing starts. Then, the critical intervals are bounded by the frames that require more work than the others, i.e. the peak frames.

The main limitation for applicability of this energy optimal policy at run-time is that the work of future frames has to be known a priori. Since that is almost never possible for real systems, speculation has to be applied. If we can predict the length and work of the next critical interval, then we can apply the method of [4] at run time.

V. Phase and peak detection

As explained in Section III, the work sequence can be decomposed in 3 major components: running average (corresponds to a static or a DC component), variable alternating component, and peaks that can be regular or not. Our major concern now is to detect and speculate on

peaks distribution within the workload, their frequency and difference from the running average.

The algorithm of peak and phase detection is given in Algorithm 1. After each frame is processed, its work w_i is saved in buffer *frames_buff* of recent M frames (the enqueue operation is denoted by an arrow). If w_i is greater than a *threshold*, that frame is detected as a peak and its w_i is saved in buffer *peaks_buff* buffer. We also keep track of the distance between successive peaks in buffer *dist_buff* and if all of the most recent peaks are equidistant, that is an indication of the sequence part where peaks regularly repeat. This mode is denoted as *periodic*, in contrast to the *aperiodic* mode. If a peak is expected but does not appear in the periodical mode, a frame is declared as a peak but not saved in *peaks_buff* and *dist* is not reset. But, when *PERIODICITY_MARGIN* successive peaks do not appear when expected, then the mode changes from the periodic to the aperiodic.

Algorithm 1: Peak and phase detection algorithm.

Input: work of the last frame $\{w_i\}$
Output: period N, isPeak, mode, w_{avg}
// After processing of frame i;
frames_buff $\leftarrow w_i$;
$w_{avg} := \sum_{1 \le j \le M} w_j / M$;
$\Delta w := w_i - w_{avg}$;
if $\Delta w \ge$ *treshold* **then**
  isPeak = *true*;
  peaks_buff $\leftarrow \Delta w$;
  treshold := *THRESHOLD_RATIO* $\cdot$ (*min* Δw *in peaks_buff*);
  dist_buff $\leftarrow$ *dist*;
  dist := 0;
  if *all* Δn_k *in dist_buff are equal* **then**
    mode := *periodic*;
    $N := \Delta n_k$;
else
  dist := *dist* + 1;
  if *dist* $\ge N \cdot$ *PERIODICITY_MARGIN* **then**
    isPeak := *false*;
    mode := *aperiodic*;
    N := *DEFAULT_N*;
  else
    isPeak := *mod*(*dist*, N) = 0;

However, as the running average varies over time, slow variations (or drift) can affect the detection of peaks and therefore, the period and phase detection. To prevent this, the peak threshold is dynamically adapted according to the size of recent peaks. It is calculated as a ratio of minimal work stored in the *peak_buff* buffer, but it will not go below a default value. The reason for this is the fact that any lower difference between a peak and the current running average will not necessarily bring any benefit to power management.

VI. Power management policy based on peak and phase detection

When peak and phase information is available, they can be used by power management. The basic observation is that peak work is significantly larger than the average work. If the operating frequency is lower than required for the peak frame (which is our aim), then a peak has negative slack. In other words, it consumes slack that has been built up by one or more frames that used less than the average work. If the accumulated slack is insufficient, it results in a deadline miss. We minimise power by averaging the operating frequency over as many frames as possible, subject to frame deadlines.

Our power management policy uses the period, or time between peaks, determined by Algorithm 1, and sets a single operating point for the entire group of frames in that period. The operating point is equal to the average frequency, such that the last frame in the group is guaranteed to meet its deadline, but intermediate frames may miss theirs. However, and crucially, it also aligns the group such that the peak frame is at the end of the period. The work of the preceding frames is therefore (on average) smaller than the average work in the period, and they build up slack, which is then used by the peak frame. Positioning the peak frame at another place in the group will likely result in a deadline miss for the peak frame (and perhaps even some following frames).

In more detail: after completing processing a peak frame, we change the operating point, according to

$$f_{new} = \frac{N \cdot w_{avg}}{N \cdot T + (s_{i-1} - s_{margin})}, \quad (1)$$

where N is the number of frames corresponding to granularity (length of the period), w_{avg} is running average work, $T = 1/f_{FR}$ is the time interval between two successive deadlines, s_{i-1} is current absolute slack (remaining slack after all previous frames up to frame i have been processed), and s_{margin} is *slack margin* that is not being immediate reused, preventing some misses on frames that come immediately after a peak. The slack margin is the amount of slack conservatively preventing occasional misses that might happen due to this speculative approach. Note that the slack margin is a fixed offset, and only causes a higher operating point once: if it is not used, it is carried over to the next period. (Essentially, all deadlines are met by a margin.)

A free interpretation of the previous formula would be that current proven slack is being re-used in coming period of N frames ending with the next peak expected. Depending on the mode of the power manager, the period will be either a default period in the aperiodic, or the detected period in periodic mode. $N \cdot w_{avg}$ is the speculation on the work in coming period ending with the first next peak, including the work of the peak. Since the expectation for work of each of the frames is equal and their deadlines are equidistant, the deadline of the last frame in period is the most critical among them, and therefore the whole work of a period has to be finished by the end of the period. The energy optimal frequency for this period is the average frequency over this period, like f_{new} is calculated. Not shown in Equation 1, f_{new} is maximized at f_{Max} and minimized at f_{Min}. Similarly, f_{new} is set to f_{Max} if the result of Equation 1 is negative, which means that the deadline of the last frame was missed, and processing needs to catch up as soon as possible.

Our heuristic power management policy can lead to misses for several reasons. First, the cumulative work of the new period (minus the slack carried over from the previous period) is larger than forecast. This happens when the running average increases, e.g. due to a scene change, as can be observed in Figure 1, around frame 120. Or, the peak is larger than expected, and the accumulated slack is insufficient. Second, the period is shorter than expected, i.e. the peak occurs earlier than forecast. This happens infrequently in our traces. The use of previously generated slack and the slack margin reduce or even eliminate the impact of the mispredictions. It may be possible to improve the power management policy to deal with these effect, e.g. by dynamical tuning of the other parameters in the peak and

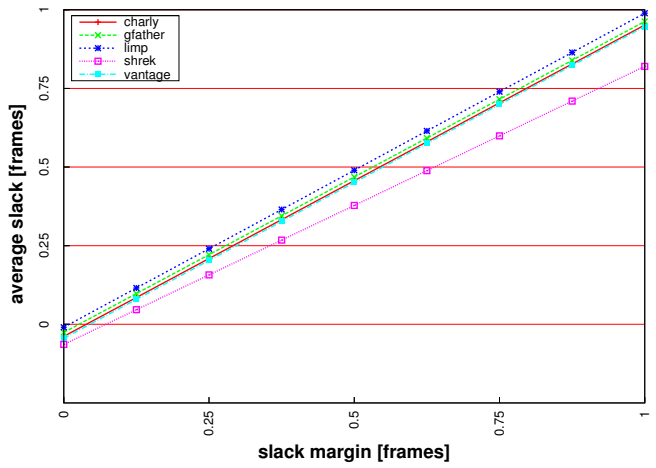

Fig. 6. Remaining slack at transition vs. given slack margin.

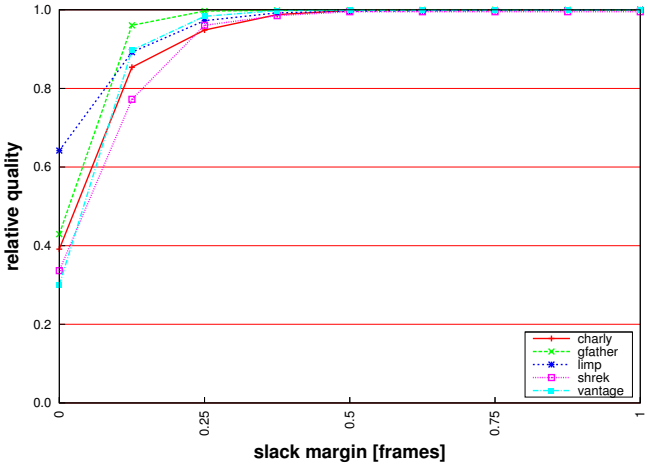

Fig. 7. Relative quality vs. given slack margin.

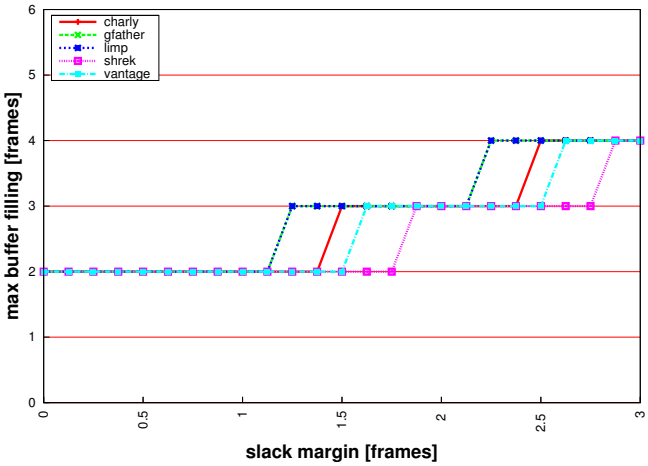

Fig. 8. Maximal buffer filling vs. given slack margin.

phase detector.

VII. Experimental results

We benchmark our approach on an MPEG4 decoder running on an ARM946 processor operating at 86 MHz. Input streams are sequences containing I and P frames of 176x144 pixels resolution, with a frame rate of 25 frames per second. The work sampling frequency is 32.57 kHz providing work samples further transformed into number of processor cycles grouped per frame. Based on this, we analytically evaluate the effects of our power management policies. We use 5 different input streams with different characteristics (e.g. scene change dynamism, complexity, level of compression). To model the overhead of transitions between operating points and policy execution, we include 20 μsec of inactive period, and 1 msec of execution at every power manager invocation. The power manager runs at the current operating point at the time of its invocation. Since peak and phase detection algorithm is neither complex nor computationally expensive, compared to the application, we add no overhead for it. Comparing to the state-of-the-art techniques, the overhead assumption of the power management overhead is quite pessimistic and larger than in a real system.

The configuration of the peak and phase detector follows: the capacity of buffers are 20 for *frames_buff*, 3 for *peaks_buff* and *dist_buff*. *THRESHOLD_RATIO* is 60%, *PERIODICITY_MARGIN* is 5, and *DEFAULT_N* is 5.

For different input sequences, Figure 6 shows the average amount of slack available at the transition moments for different values of the slack margin. Both slack and slack margin are expressed in time in units of $T = 1/f_{FR}$. Once the targeted amount of slack is generated, the system runs at the frequency that adapts to the work, and tries to keep that slack. The slack does not accumulate further and thus extra energy is not consumed.

Figure 7 shows the number of frames decoded on time, normalized to the total number of frames in the input sequence. For a slack margin of $0.5T$ or bigger, there are just few misses, while for lower values the number of misses is between 50% and 90%.

Maximum buffer fillings are shown in Figure 8. These values exclude any buffer filling at the start of the sequence, before the power manager has started. The values are low and differ up to at most two frames from the given slack margin. From Figure 6 and 8 we conclude that the maximal and average buffer fillings are very close. This confirms that presented policy keeps the system operating frequency very close to the current work requirements, without generating extra slack, while keeping the number of misses low.

In the experimental sequences, when in a periodic mode, the phase and period detector detects 12 frames as the most common granularity. The average energy consumption per frame for all traces is 2.43 J, which is around 30% lower than energy consumption running without any power management, including its negligible energy overhead.

VIII. Related work

A wide range of heuristics and power management policies have been developed based on history and pattern matching of workload of an application. One of them is presented in [5], but since it was matching very complex patterns, in general it does not perform well. Our pattern is simpler and performs well if applied on appropriate use-case. The speed of DVFS infrastructure is increasing [6], enabling power management at very fine granularity. This was the motivation for the previous and this work. Azevedo [7] 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 [8] presents the collaboration between compiler and operating system and by inserting instrumentation code into the program code to vary the granularity.

The same authors propose theoretical solution for choosing the optimal granularity in [9]. Choi [10] presents DVFS technique for an MPEG decoder with sub-frame granularity by differentiating between invariable and variable parts of a decoder. Son [11] proposes weighted average to speculate on future work of an MPEG decoder, very similarly to a PID-based control system. Gheorghita et al. in [12] proposed application scenarios to be used with power management. That method can be combined with our peak detector if there is suitable workload pattern within a scenario and different settings can be given to the peak detector depending on a specific scenario.

IX. Conclusions

In this paper we introduce a dynamic power management policy that removes slack in the system and uses it to reduce power and energy consumption for soft real-time applications. We also present the analysis and the characteristics of the workload of an MPEG4 decoder. Analysis results in workload decomposition into slow and fast-changing components, and with periodic peaks. We have developed a phase and period detector that observes application workload and detects periodical patterns and speculates on the future work. It also dynamically aligns the application workload peaks and the power management transitions.

This power management policy lowers the energy consumption up to 30% on average, in comparison to the system execution without applying any power management. The result is comparable with the results of static and conservative approach published before [1]. Additionally, it reduces the maximal and average buffer space needed for processed frames before displaying, as well as the amount of the slack in the system. These values are parameters in our policy, and they can be changed to trade the number of misses and energy consumption.

Acknowledgements

Authors would like to thank Albert Molderink of the University of Twente for the collaboration and for providing us with the MPEG4 sequences on which this work is based.

References

[1] A. Milutinovic, K. Goossens, and G. Smit, "Impact of power-management granularity on the energy-quality trade-off for soft and hard real-time applications," *System-on-Chip, 2008. SOC 2008. International Symposium on*, pp. 1–4, Nov. 2008.

[2] A. Molnos and K. Goossens, "Conservative dynamic energy management for real-time dataflow applications mapped on multiple processors," *Digital System Design, DSD*, August 2009.

[3] A. Hansson, K. Goossens, M. Bekooij, and J. Huisken, "CoMPSoC: A template for composable and predictable multi-processor system on chips," *Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, 2009.

[4] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, 1995.

[5] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU," *Proceedings of the First ACM International Conference on Mobile Computing and Networking*, 1995.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] D. Son, C. Yu, and H. Kim, "Dynamic Voltage Scaling on MPEG Decoding," *Proceedings of the Eighth International Conference on Parallel and Distributed Systems, ICPADS 2001.*, pp. 633–640, 2001.

[12] S. Gheorghita, T. Basten, and H. Corporaal, "Application scenarios in streaming-oriented embedded system design," pp. 1–4, 2006.