Compression-Aware Energy Optimization for Video Decoding Systems With Passive Power

Emrah Akyol and Mihaela van der Schaar

***Abstract*—The objective of dynamic voltage scaling (DVS) is to adapt the frequency and voltage for configurable platforms to obtain energy savings. DVS is especially attractive for video decoding systems due to their time-varying and highly complex workload and because the utility of decoding a frame is solely depending on the frame being decoded before its display deadline. Several DVS algorithms have been proposed for multimedia applications. However, the prior work did not take into account the video compression algorithm specifics, such as considering the temporal dependencies among frames and the required display buffer. Moreover, the effect of the passive (leakage) power when performing DVS for multimedia systems was not explicitly considered. In this paper, we determine the optimal scheduling of the active and passive states to minimize the total energy for video decoding systems. We pose our problem as a buffer-constrained optimization problem with a novel, compression-aware definition of processing jobs. We propose low-complexity algorithms to solve the optimization problem and show through simulations that significant improvements can be achieved over state-of-the art DVS algorithms that aim to minimize only the active power.**

***Index Terms*—Buffer control, dynamic voltage scaling (DVS), power optimization for multimedia systems, video compression.**

I. Introduction

RECENTLY, the area of energy saving in multimedia applications has gained interest due to high-energy consumption of multimedia applications on devices with limited power, such as mobile devices [1]–[5]. One method of energy savings is dynamic voltage scaling (DVS) where the operating frequency and voltage are dynamically adapted to the workload. The main goal of the DVS algorithms is to utilize the energy–delay tradeoff for tasks whose jobs' completion times are immaterial as long as they are completed before their deadline [6]. An example of such a task is real-time video decoding, where early completion of frame decoding does not provide any benefit as long as the display deadline is met. DVS algorithms assign the operating level (i.e., power and frequency) for each job given the estimated cycle requirement (i.e., complexity) and the completion deadline of the job. Several DVS algorithms have been proposed for delay sensitive applications such as video decoding [4], [5], [7], [8]. Most DVS algorithms perform optimization over the current job in a greedy manner considering the worst case complexity estimate [7], [8] or average complexity estimates [4]. In [7], a state-of-the-art DVS algorithm for video decoding was presented, where worst case complexity estimates are generated on the fly, just based on the complexities of the previous frames with the same frame type. Then, for each frame, based on the worst case complexity estimate, frequency is adjusted just to finish the frame decoding job in time. Subsequently, this algorithm is referred to as "conventional DVS." Conventional DVS algorithms aim to match frequency/voltage to fixed time intervals, e.g., for the decoding of 30-Hz video, voltage/frequency is controlled such that each frame is decoded in 1/30 s. Hence, these methods do not consider the deployed compression structure, which requires a different number of frames to be decoded at different time instances depending on the adopted temporal prediction structure. In [16], we proposed a new method of DVS for multimedia where accurate complexity modeling is utilized that considers compression specifications. None of the prior research, including our prior work, on DVS for multimedia has explicitly addressed the effect of the passive (leakage) power, which becomes increasingly important given the recent developments in decreasing active power. As mentioned in, e.g., [8] "the static power consumption is comparable to the dynamic power dissipation and projected to surpass it if measures are not taken to minimize leakage current." Hence, in this work, we consider the effect of passive power for DVS for multimedia applications.

Idle (sleep) states are especially important when the idle state can consume nearly as much power as the active state [8]–[11], which necessitates the joint optimization of the idle-state scheduling and DVS. Note that, while idle-state scheduling for sensor networks [9], [11] and leakage-aware (i.e., passive power aware) DVS for real-time applications [8] were separately considered in prior research, their joint optimization in the context of DVS for the video decoding applications becomes important due to the dynamic behavior and relatively high complexity of these applications.

In this paper, we explicitly consider the video compression specifics to redefine conventionally used job definitions for video decoding and provide a solution for energy minimization (that takes into account the effect of leakage power) by jointly considering DVS and idle-state scheduling. We also utilize a postdecoding buffer to mitigate the highly varying complexity profile of video decoding, not only to be able to decode high-complexity frames [12], but also to perform efficient DVS by relaxing the hard job deadlines to soft buffer overflow/underflow constraints. A postdecoding buffer is crucial for video decoding applications since bursts of frames should be jointly decoded until a given deadline. Hence, we investigate the optimal scheduling of active and passive states together with the optimal frequency assignment using a buffer-controlled DVS framework for video decoding systems. The letter is organized

Manuscript received January 18, 2007; revised August 2, 2007. First published July 25,2008; current version published October 8, 2008. This paper was recommended by Associate Editor I. Ahmad.

E. Akyol is with the Electrical and Computer Engineering Department, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: eakyol@ece.usb.edu).

M. van der Schaar is with the Electrical Engineering Department, University of California, Los Angeles, CA 90095-1594 USA (e-mail: mihaela@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2008.928886

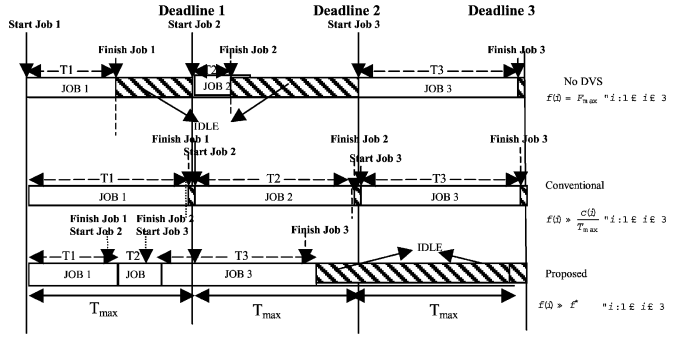

Fig. 1. DVS strategies. (1) No DVS (top panel). (2) conventional DVS [2] (middle panel). (3) Proposed DVS with sleep states (bottom panel).

as follows. Section II describes the effects of passive power on DVS algorithms and presents the novel job definitions specific to video decoding. The extension of buffered DVS for systems with passive power and low complexity suboptimal solutions are presented in Section III. Comparative results are presented in Sections IV and V concludes the paper.

II. DVS With Passive Power

A. Effect of Passive Power on DVS

In the following, we illustrate the various DVS methods and the effect of passive power using a simple example, shown in Fig. 1. Let us assume we have $M = 3$ jobs with complexities $c(1) = (C_{\max}/2)$, $c(2) = (C_{\max}/4)$, $c(3) = C_{\max}$. From now on, we use the term complexity to represent the number of execution cycles. The top panel in Fig. 1 illustrates a no-DVS scenario: the processing is performed at the maximum frequency available $F_{\max}$ and the corresponding $P_{\max}$ (maximum active power) for each job. When the job finishes, the processor goes into an idle state. For conventional DVS, as shown in the middle panel of Fig. 1, the frequencies are adjusted to finish each job "just-in-time" prior to its delay deadline, i.e., $f(1) = (F_{\max}/2)$, $f(2) = (F_{\max}/4)$, $f(3) = F_{\max}$. In this case, the remaining idle times are neglected. Let us assume an active power-frequency relationship of $P \propto f^3$ [1], [7]. Then, the active power will be $p(1) = (P_{\max}/8)$, $p(2) = (P_{\max}/64)$, $p(3) = P_{\max}$. The total "active" energy spent on the group of jobs can be found as

$$E^{\text{active}} = \sum_{i=1}^{3} \frac{p(i)c(i)}{f(i)}. \tag{1}$$

Hence, the total active energy spent for the no-DVS case is $E^{\text{active}}_{no\_DVS} = 1.75E$ and for the conventional DVS $E^{\text{active}}_{DVS} = 1.14E$, where $E = (P_{\max} \cdot C_{\max}/F_{\max})$. As can be seen from this simple example, if we neglect the effect of passive power, the conventional DVS algorithm decreases the energy consumption significantly. However, in the cases of nonnegligible passive power, we should consider the total energy, i.e., $E^{\text{total}} = E^{\text{active}} + E^{\text{passive}}$. The passive energy can be written as

$$E^{\text{passive}} = \sum_{i=1}^{3} P_{\text{pass}} \cdot \frac{c(i)}{f(i)} + P_{\text{idle}} \cdot t_{\text{idle}} + E_{\text{trans}} \cdot \text{sign}(t_{\text{idle}}) \tag{2}$$

where P_{pass} is the passive power, P_{idle} is the power in idle (sleep) mode, t_{idle} is the idle (sleep) time, and E_{trans} is the transition energy [8], [11]. The transition energy is the energy spent by going into the active state from the idle state. When the idle time t_{idle} is zero, there is no transition energy, captured with the sign function in (2). We assume that the passive power is independent of frequency/voltage changes throughout the paper and there is no transition cost and no power dissipation in idle state for this simple example, i.e., $E_{\text{trans}} = P_{\text{idle}} = 0$. Then, if we assume $P_{\text{pass}} = P_{\max}$ (which is typical for several devices [8], [11]), the total energies corresponding to no-DVS and conventional DVS are $E^{\text{total}}_{\text{no\_DVS}} = 3.5E$ and $E^{\text{total}}_{\text{DVS}} = 4.14E$. DVS algorithms that merely minimize "active" energy spent are suboptimal when passive power is significant. As can be seen from this example, while conventional DVS decreases active energy, total energy spent in conventional DVS may be even higher than no-DVS scenario since processing time, thus passive energy spent, also increases in conventional DVS.

The bottom panel of Fig. 1 shows the proposed joint optimization of DVS and sleep states. If the frequency is kept constant at $f(1) = f(2) = f(3) = 0.79F_{\max}$, then the total energy equals $E^{\text{total}}_{\text{DVS+state\_sch.}} = 3.31E$, which is significantly less than conventional DVS approaches. The effect of the transitions is neglected for the simplicity of the example, but significant savings can be obtained by optimizing idle states with respect to buffer sizes, as will be shown later in the paper. Most processors have different idle states with different power consumption P_{idle} and transition energy E_{trans} levels. For example, there are three different idle states in most processors such as "shut down," where $P_{\text{idle}} = 0$ but the transition energy E_{trans} is high, the "idle" state where $E_{\text{trans}} = 0$ but P_{idle} is high, and the "stand-by" state that has nonzero power consumption and transition energy [11]. In Section III, we analyze the optimal choice of idle state in addition to optimal DVS and idle state scheduling.

B. Video-Compression-Specific DVS

In the current highly adaptive video compression algorithms, the current frame is predicted from both past and future frames. Some reference frames should be decoded before their display deadline to be used as reference. The frames that are jointly encoded share the same decoding deadline. Hence, unlike previous work on multimedia DVS [7] which assumes a job as decoding a single frame, we combine the frames with the same *decoding deadline* and define decoding this collection of dependent frames as *one job* of the decoding task. The proposed new job definition is especially important for new video coding methods, where multiple frames are jointly filtered temporally. In general, we define every job with three parameters as follows:

job: : {deadline, complexity, size}

deadline	decoding deadline of the job j, $d(j)$;
complexity	estimated number of cycles that job j consumes on a specific platform, $c(j)$; these estimates can be achieved with several methods, see, e.g., [13];
size	number of decoded *original* frames by the job j, $s(j)$.

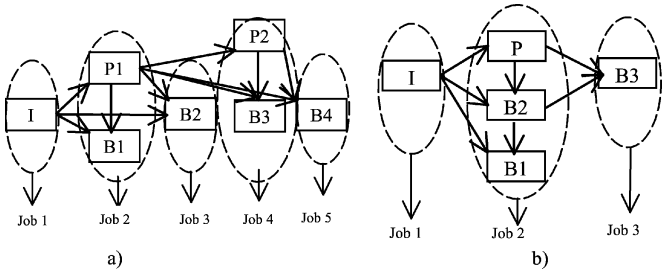

Fig. 2. DAGs and job definitions for (a) conventional I-B1-B2-P1-B3-B4-P2 frames and (b) hierarchical B pictures, I-B1-B2-B3-P.

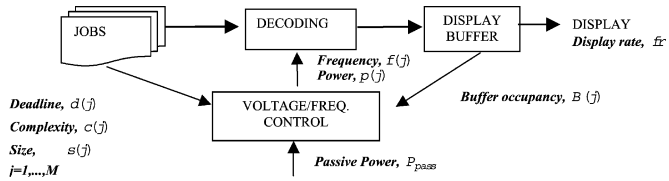

Fig. 3. Buffer-constrained DVS scheme.

In predictive coding, frames are encoded with interdependencies that can be represented by a directed acyclic dependence graph (DAG). Examples are shown in Fig. 2 for two different GOP structures: the conventional I-B-B-P-B-B-P GOP structure and hierarchical B pictures. DAGs can represent the distinction of a job and frame decoding clearly, for this example; decoding frame I and decoding frames P1 and B1 represent different jobs as shown in Fig. 2. Hierarchical B pictures structure, which is based on flexible reference frame selection in the latest compression standard H.264/AVC, is more interesting in terms of jobs with varying sizes, complexities, and deadlines. The first job is decoding frame I $(s(1) = 1, d(1) = \Delta t)$, where the second job is decoding frames P, B1, and B2 $(s(2) = 3, d(2) = 2\Delta t)$, and the last job is decoding frame B3 $(s(3) = 1, d(3) = 4\Delta t)$. Also, note that the first job, decoding the I-frame, is composed of only the texture related complexities (i.e., entropy coding and inverse transform), whereas the second job includes several bi-directional motion compensation operations, thereby showing that not only the complexity is varying per job, but also the type of the complexity (entropy coding, inverse transform, motion compensation, or interpolation) is changing. Thus, while decoding two B-frames is the same from a high-level perspective, the job parameters are substantially different showing the superiority of the delay-aware job definitions over conventional job definition. The proposed job definitions do not require any complexity overhead as they can be readily deduced from the encoding structure (see the illustrative examples given above).

III. Proposed DVS Algorithm

Let us first formally define the general DVS problem. Assume that there is a discrete set of operating levels with corresponding frequency and power levels, which can be used for voltage/frequency adaptation $\mathbf{L} = \{l_j : (p_j, f_j), 1 \leq j \leq N | l_1, .., l_N\}$. Each level has a different power consumption and different frequency, $(\mathbf{P}, \mathbf{F}) = \{(p_1, f_1), ..., (p_N, f_N) | p_1 < ... < p_N; f_1 < ... < f_N\}$, where the power is an increasing function of the frequency. Assume that there are a total of M jobs with complexity estimates $\mathbf{C} = \{c(1), ..., c(M)\}$, deadlines $\mathbf{D} = \{d(1), ..., d(M) | d(1) < ... < d(M)\}$, and sizes $\mathbf{S} = \{s(1), ..., s(M)\}$. Assume job j has a processing time, which can be simply found by $t(j) = c(j)/f(j)$, and total energy consumption $E(j)$. Then, the DVS problem attempts to find the set of operating level (frequency and thus power) for each job $\mathbf{l^{opt}} = \{l(1), ..., l(M) | \forall l \in \mathbf{L}\}$ that minimizes the total energy. Thus, the general DVS optimization can be formulated as shown in (3) at the bottom of the page [7], [8].

Another video compression-specific feature of the proposed DVS framework is the usage of a postdecoding (display) buffer between the display device and the decoding platform. This postdecoding buffer helps to mitigate complexity estimate mismatches and converts hard job deadlines to soft buffer underflow/overflow constrains. Note that, in the formulation above, there are M distinct constraints for the processing times. By utilizing this buffer, these constraints are converted to buffer overflow/underflow constraints.

Fig. 3 shows the general setup for the buffer-controlled DVS for video decoding. Frequency for job j is determined by considering the parameters of total M jobs, buffer occupancy $B(j)$, and passive power P_{pass}. For each job, complexity estimates are updated, the buffer occupancy is checked, and the frequency for the job is assigned. The DVS optimization problem described below is aimed at minimizing the total energy by adapting the frequency/voltage level over time, based on the complexity of the tasks, such that the buffer overflow/underflow constraints are fulfilled at all times. Note that, in the proposed optimization, all jobs are completed before their decoding deadline and hence there is no video distortion incurred due to DVS. Throughout the paper, we assume the passive power P_{pass} is independent of the frequency [1], [7]. We neglect the overhead of buffer management energy cost throughout the paper, and we assume that

$$\textbf{General DVS Problem}:$$
$$\text{find } \mathbf{l^{opt}} = \arg\min_{l \in \mathbf{L}} \sum_{j=1}^{M} E(j) \text{ (energy consumption)}$$
$$\text{subject to}: \sum_{m=1}^{j} t(m) < d(j); \quad \forall\, j : 1 \leq j \leq M \text{ (delay constraints)} \tag{3}$$

the entire encoded bitstream is available before decoding. See (4), shown at the bottom of the page.

In the optimization above, we define the buffer occupancy for job j as $B(j)$ based on the following recursion:

$$B(j) = \max(B(j-1) + s(j) - t(j)\cdot fr, 0)$$
$$B(0) = B_{\text{initial}} \tag{5}$$

where fr denotes the frame rate and B_{initial} is the initial state of the buffer that depends on the initial playback delay (which may be zero if any delay is not tolerable). $B(j)$ is defined in terms of number of frames.

The optimal frequency $f^*(j)$ is defined as

$$f^*(j) = \arg\min_{f\in F} E(j) \tag{6}$$

where

$$E(j) = p(j)t(j) + P_{\text{pass}}t(j). \tag{7}$$

$f^*(j)$ can be determined by solving the differential equation

$$\left.\frac{\partial E(j)}{\partial f(j)}\right|_{f^*(j)} = 0. \tag{8}$$

By replacing (5) and substituting (7) into (8), we obtain the optimal frequency as the frequency that satisfies the differential equation

$$f(j)\frac{\partial p(f(j))}{\partial f(j)} - p(f(j)) = P_{\text{pass}}. \tag{9}$$

The optimization strategy depends on the value of the set of operating frequencies with respect to $f^*(j)$. As can be seen from (9), the optimal frequency $f^*(j)$ does not depend on the job index, i.e., irrespective of the complexity of the job, the optimal frequency is identical and only depends on the power-frequency relationship. Hence, we drop j in the following derivations. Let $f_{\min}$ be the minimum frequency that the processor can operate at, while satisfying buffer underflow constraints. Note that $f_{\min}$ depends on both the processor's frequency-power relationships and the novel job definitions. Depending on the temporal encoding structure, $f_{\min}$ can take different values. Let us define two distinct cases: 1) $f_{\min} \le f^*$ (case-1) and 2) $f_{\min} > f^*$ (case-2). In our subsequent derivations, we assume a power-frequency function as in [1] and [7]: $p(f) = \alpha\cdot f^k$ with $k > 2$. Based on this, we can determine the optimal solutions for different frequency regions.

Proposition: If $f_{\min} > f^*$ (i.e., case-2), then the total energy is a convex function of frequency for all f greater or equal to $f_{\min}$.

Proof: To prove that the energy is convex increasing in the frequency, it suffices to show that the first and second derivatives are both positive as

$$\frac{\partial E}{\partial f} = \frac{c}{f^2}\left(\alpha(k-1)f^k - P_{\text{pass}}\right) \tag{10}$$
$$\frac{\partial^2 E}{\partial f^2} = \frac{c}{f^3}\left(\alpha(k-1)(k-2)f^k + 2P_{\text{pass}}\right). \tag{11}$$

From (9), we obtain that the optimal frequency is

$$f^* = \sqrt[k]{\frac{P_{\text{pass}}}{\alpha(k-1)}}. \tag{12}$$

From (12), $f_{\min} > f^*$ indicates $\alpha(k-1)f^k > P_{\text{pass}}$ for $\forall f \ge f_{\min}$ which makes $(\partial E/\partial f) > 0$ for $\forall f \ge f_{\min}$. $(\partial^2 E/\partial f^2) > 0$, for $\forall\, f > f_{\min}$ when $k > 2$ since both terms $(\alpha(k-1)(k-2)f^k$ and $2P_{\text{pass}})$ are positive in (11). Hence, $(\partial E/\partial f) > 0$ and $(\partial^2 E/\partial f^2) > 0$, showing that $E(f)$ is convex for $\forall f \ge f_{\min} > f^*$. ■

1) Case-1 ($f_{\min} \le f^*$): The optimum frequency is in the range of operating frequencies of the processor; hence, this frequency (which does not depend on the complexity) can be applied to all jobs. However, this assignment may not guarantee to satisfy the buffer overflow constraint. Since $E(f)$ is not convex in that region, slope-based (i.e., Lagrangian optimization type) techniques cannot be applied [14]. Also, the processor can go into different type of idle (sleep) states, such as sleep-1 that consumes negligible power at idle state $\left(P^1_{\text{idle}}\right)$ but results in a significant energy consumption in state transitions $\left(E^1_{\text{trans}}\right)$ and sleep-2 that costs significant power at idle states but has less transition cost. Let us assume that sleep-1 has energy transition cost E^1_{trans}, but no steady state cost. Sleep-2 has transition cost $E^2_{\text{trans}}\left(E^2_{\text{trans}} < E^1_{\text{trans}}\right)$ and idle state cost P^2_{idle}, where $P^1_{\text{sleep}} < P^2_{\text{sleep}}$. Let $t_{\text{idle}}(j)$ be the time that the processor sleeps after job j. Then, we need to minimize the energy by determining the joint optimal scheduling of the sleep states and sleep times in addition to the optimal frequency selection, as formulated in the optimization problem given in (13), shown at the bottom of the next page.

In the above formulation, $\mathbf{l^{opt}}$, $\mathbf{t}_{\text{idle}}$, and $\mathbf{state}$ are $M\times 1$ vectors that represent the optimal operating level (frequency/power), optimal idle time and idle state type (sleep-1 or sleep-2) for M jobs.

Buffer Constrained DVS Optimization considering Passive Power :

$$\mathbf{l^{opt}} = \arg\min_{l\in L}\sum_{j=1}^{M}\{p(j)t(j) + P_{\text{pass}}t(j)\}\ (\text{energy consumption})$$
$$\text{subject to}: 0 \le B(j) < B_{\max};\quad \forall\, j: 1\le j\le M\ \left(\tfrac{\text{buffer overflow}}{\text{underflow constraint}}\right) \tag{4}$$

The optimal solution to these types of constrained problems can be found by deploying deterministic dynamic programming-based algorithms such as the Viterbi algorithm for which, given the initial conditions, the best solution resulting to each possible states are found. Note that, in our problem, there is a limited number of states (due to the limited buffer size), hence, we can use a dynamic programming algorithm. Basically, we can create a trellis to represent all viable parameters (state type, idle time, and operating level) at each instant, and then, given the buffer constraints, we can prune out the branches that fail to satisfy the buffer constraints. Also, for the same buffer occupancy, only the branch that has the lowest total energy is kept while others are pruned out. Here, buffer constraints are considered relying on the complexity estimates, hence we inherently assume complexity estimates are accurate enough such that, when the estimates satisfies/violates the buffer constraints, the real complexity values should also satisfy/violate the buffer constraints. Then, each surviving branch generates a path (feasible parameter choice) with an associated energy cost. Finally, the path with minimum total energy cost is chosen as the optimal parameter choice for M jobs. The complexity of this optimal solution increases linearly with the number of operating levels (N), number of jobs (M), and the possible choices for idle time. The possible choice for idle time is limited by the finite size of the buffer $B_{\max}$. Hence, the order of the worst case complexity of the dynamic programming based solution is about $O(N \cdot B_{\max} \cdot M)$. Since the dynamic programming-based optimization can be impractical due to its high complexity, we propose a suboptimal low-complexity algorithm to solve this problem. The solution is based on aggregating the jobs (i.e., decoded frames) until the buffer is close to overflow $(B(j) > \beta_1)$ and going into the sleep state to avoid overflow and stay in the sleep state until the buffer is close to underflow $(B(j) < \beta_2)$. The thresholds β_1 and β_2 can be determined using complexity estimates or can be set heuristically, depending on the buffer size [14]. Idle (sleep) state type (sleep-1 or sleep-2) is determined according to the buffer occupancy and complexities of the jobs. The worst case complexity of this suboptimal fast algorithm is very low $(O(1))$, since the operating level (i.e., voltage and frequency) is set just once, before all of the jobs are processed, and the idle time is chosen as the depletion time of the buffer when it is full. The performance degradation, compared with the optimal high-complexity algorithm, of this suboptimal low-complexity algorithm is experimentally shown in the Section IV.

To determine the sleep-state type, we need to compare the expected energy consumption at the sleep state for one buffer depletion time. The total energies spent in the idle state-1 and idle state-2 (transition energy + passive energy in idle state) for one buffer depletion time are $E_1 = E^1_{\text{trans}} + ((\beta_2 - \beta_1)/fr)\, P^1_{\text{idle}}$ and $E_2 = E^2_{\text{trans}} + ((\beta_2 - \beta_1)/fr)\, P^2_{\text{idle}}$, respectively. The optimal sleep-state type depends on the buffer size (β_1 and β_2) and transition and idle energy spent. Sleep-1 should be preferred when $E_1 < E_2$, i.e., $\left(\left(E^1_{\text{trans}} - E^2_{\text{trans}}\right)/(\beta_2 - \beta_1)\right) fr < P^2_{\text{idle}} - P^1_{\text{idle}}$. Sleep-2 should be chosen when $\left(\left(E^1_{\text{trans}} - E^2_{\text{trans}}\right)/(\beta_2 - \beta_1)\right) fr > P^2_{\text{idle}} - P^1_{\text{idle}}$.

To find the operating frequency, we modify the optimal frequency found in (10), considering the power in sleep mode and transition energy. We note that the sum of idle and active time, i.e., total time is constant for M jobs corresponding to a total of N frames

$$\sum_{j=1}^{M} \{t(j) + t_{\text{idle}}(j)\} = \frac{N}{fr}. \tag{14}$$

Since we keep the frequency constant for all jobs, we can write

$$\sum_{j=1}^{M} t(j) = \frac{\sum_{j=1}^{M} c(j)}{f} \tag{15}$$

$$\sum_{j=1}^{M} t_{\text{idle}}(j) = \frac{N}{fr} - \frac{\sum_{j=1}^{M} c(j)}{f}. \tag{16}$$

The number of transitions is identical to the number of buffer depletions, which is $\left(\sum_{j=1}^{M} t_{\text{idle}}(j)/(\beta_2 - \beta_1)\right) fr$. Hence, the total energy can be written as shown in (17) at the bottom of the next page. Similar to (9), the operating f_{opt} frequency can be found as the frequency that satisfies the differential equation

$$f\frac{\partial p(f)}{\partial f} - p(f) = \left(P_{\text{pass}} - P_{\text{idle}} - \frac{E_{\text{trans}}}{\beta_2 - \beta_1} fr\right). \tag{18}$$

As can be seen, the operating frequency depends on the active power-frequency function, which characterizes the video decoding device $p(f)$, passive power P_{pass}, power in sleep (idle) mode P_{idle}, and the transition energy E_{trans} from idle state to active state. It also depends on the buffer size $(\beta_2 - \beta_1)$, since, as the buffer size increases, the number of transitions decreases. Note that the operating frequency does not depend on the buffer occupancy or the complexity of the job, such that evaluating this frequency takes place only once for the process, hence the proposed algorithm has very low complexity overhead. Also note that, since there is a limited number of discrete frequencies available, the available frequency closest to the optimal frequency f_{opt} is selected. The algorithm is given in Table I.

Buffer Constrained DVS Optimization with Passive Power and Sleep States:

$$\mathbf{l^{opt}}, \mathbf{t}_{\text{idle}}, \mathbf{state} = \arg\min_{l \in L} \sum_{j=1}^{M} \{p(j)t(j) + P_{\text{pass}}t(j) + P^{\text{state}}_{\text{idle}} t_{\text{idle}}(j) + E^{\text{state}}_{\text{trans}} \text{sign}(t_{\text{idle}}(j))\}$$
$$\text{subject to}: 0 \le B(j) < B_{\max}; \quad \forall j : 1 \le j \le M, \text{ where } B(j) = \max(B(j-1) + s(j) - (t(j) + t_{\text{idle}}(j)) \cdot fr, 0) \tag{13}$$

TABLE I
PROPOSED LOW-COMPLEXITY SUBOPTIMAL ALGORITHM

1. Find the operating frequency closest to f_{opt} according to Equation-18.
2. For each job $j; 1 \le j \le M$,
3. Execute the job with the assigned frequency (f_{opt}) and check the buffer occupancy $B(j)$.
4. If $B(j) \ge \beta_2$, go into the sleep state and wait until $B(j) \le \beta_1$
   Else, continue with same frequency (i.e., go to Step 4), and proceed to next job, i.e., set $j = j+1$.
5.

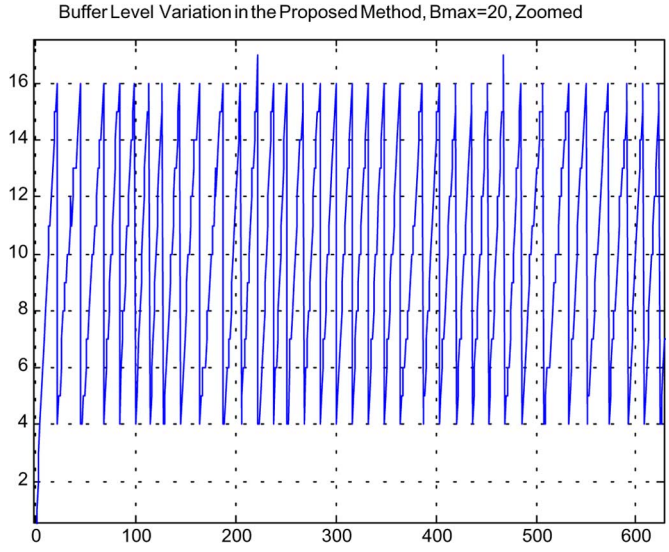

Fig. 4. Buffer fullness example for one realization of the proposed algorithm.

Fig. 4 illustrates the buffer-level variation for one realization. The buffer level increases as the jobs are processed faster than the required display speed until the buffer is close to overflow (i.e., $\beta_2 = 16$). Subsequently, the processor goes in the sleep mode until the buffer level decreases to $\beta_1 = 4$.

2) Case-2 ($f_{\min} > f^*$): Since all of the frequencies are in the convex region, there should be no sleep states and the processor should operate at the minimum frequency that satisfies buffer (delay) constraints. The new job definitions are especially important for this case where the optimization should take the complexities of each job into account. This problem is analogous to the conventional rate control problem with buffer constraints in video coding (see, e.g., [14]). Hence, the interested reader is referred to existing literature [14], [17] for details on finding the solution of this problem.

IV. RESULTS

Here, we compare the proposed DVS methods with the conventional DVS method for platforms with significant power. In our experiments we used four different test sequences, *foreman, mobile, coastguard*, and *silence*, at CIF resolution and at a frame rate of 30 fps. We used a wavelet video coder [18] that utilizes motion-compensated temporal filtering with four-level temporal and five-level spatial decompositions. We generated two sets of decoded video with high and low complexity at two rates, 512 and 1024 kbps. High-complexity compression includes adaptive update and quarter-pixel motion compensation whereas the update step is skipped and only a half-pixel MC is used for the low-complexity case. To obtain statistically meaningful results, we concatenated the resulting 16 videos in 12 different orders, resulting in a set of 12 long videos with 3072 frames each. We present the average results of 12 videos with different decoding traces. The complexity values are obtained from actual measurements whereas the power and frequency values that we used are shown in Table II, and they are reported for the Intel Strong ARM processor [15] in [16].

TABLE II
FREQUENCY POWER VALUES OF STRONG-ARM PROCESSOR ANALYZED IN [16]

Frequency (MHz)	59	74	89	103	118	133	148	177	192	206
Power(mW)	33.2	42.0	54.0	71.2	91.8	115.5	149.5	221.0	280.0	360.0

We assume different values for the passive power and two sleep states to obtain meaningful results for a broad class of video decoding systems. We assume a passive power $P_{\text{pass}} = P_{\max}$ or $P_{\text{pass}} = P_{\max}/2$ and two sleep (idle) states. Sleep (idle) state-1 has the transition energy $E_{\text{trans}} = P_{\max}/fr$ and idle power $P_{\text{idle}} = 0$, while sleep (idle) state-2 has $E_{\text{trans}} = 0$ and $P_{\text{idle}} = P_{\max}/10$ [11]. Hence, we have four different passive power–sleep state combinations. β_1 and β_2 can be set heuristically as a linear function of the maximum buffer size as done in the solution of the well known rate control problem in [9]. Alternatively, they can be determined based on the complexity estimates and the expected error in complexity estimates, such as done in our previous work [17]. For the experiments, we fix the buffer size $B_{\max} = 20$ and $\beta_1 = 0.2B_{\max}$, $\beta_2 = 0.8B_{\max}$ heuristically. The comparative results are presented in the following.

Table III shows the results for the case where $P_{\text{pass}} = P_{\max}$ and sleep (idle) state -1, i.e., the transition energy is assumed as $E_{\text{trans}} = P_{\max}/fr$ and there is no idle power $P_{\text{idle}} = 0$. No DVS, as illustrated in the top panel of Fig. 1, denotes the case where there is no change in frequency and the processor is run at the highest maximum power/frequency ($f_{\max}$, $P_{\max}$) until the job is finished. Then, it goes into idle (sleep) state until the next job. The active energy spent in this case is assumed as the unit energy (E) and all other energies are scaled according to E. Since $P_{\text{pass}} = P_{\max}$, the passive energy in the no-DVS scenario is also E. There is no idle energy in every scenario since $P_{\text{idle}} = 0$. There is transition at the end of every job, resulting in the highest transition energy. The conventional DVS scenario

$$E_{\text{total}} = \underbrace{p(f)\frac{\sum_{j=1}^{M} c(j)}{f}}_{\text{Active energy}} + \underbrace{P_{\text{pass}}\frac{\sum_{j=1}^{M} c(j)}{f}}_{\text{Passive energy}} + \underbrace{P_{\text{idle}}\left[\left(\frac{N}{fr}\right) - \frac{\sum_{j=1}^{M} c(j)}{f}\right]}_{\text{Idle energy}} + \underbrace{\frac{\left(\frac{N}{fr}\right) - \frac{\sum_{j=1}^{M} c(j)}{f}}{\beta_2 - \beta_1} \cdot fr \cdot E_{\text{trans}}}_{\text{Transition energy}} \tag{17}$$

TABLE III
COMPARATIVE (SCALED) RESULTS ON ACTIVE, PASSIVE, AND TOTAL ENERGY CONSUMPTION, THE CASE WHERE $P_{\text{pass}} = P_{\max}$ AND SLEEP (IDLE) STATE-1

	Active Energy	Passive Energy	Idle Energy	Trans. Energy	Total Energy
No DVS	E	E	-	$1.08E$	$3.19E$
Conventional DVS	$0.44E$	$1.99E$	-	$0.01E$	$2.43E$
Proposed	$0.68E$	$1.07E$	-	$0.08E$	$1.83E$
Optimal	$0.55E$	$1.03E$	-	$0.08E$	$1.58E$

TABLE IV
COMPARATIVE RESULTS ON ACTIVE, PASSIVE, AND TOTAL ENERGY CONSUMPTION, THE CASE WHERE $P_{\text{pass}} = P_{\max}/2$ AND SLEEP (IDLE) STATE-1

	Active Energy	Passive Energy	Idle Energy	Trans. Energy	Total Energy
No DVS	E	$0.5E$	-	$1.08E$	$2.58E$
Conventional DVS	$0.44E$	E	-	-	$1.44E$
Proposed	$0.49E$	$0.74E$	-	$0.05E$	$1.28E$
Optimal	$0.38E$	$0.71E$	-	$0.05E$	$1.14E$

TABLE V
COMPARATIVE RESULTS ON ACTIVE, PASSIVE, AND TOTAL ENERGY CONSUMPTION, THE CASE WHERE $P_{\text{pass}} = P_{\max}$ AND SLEEP (IDLE) STATE-2

	Active Energy	Passive Energy	Idle Energy	Trans. Energy	Total Energy
No DVS	E	E	$0.11E$	-	$3.19E$
Conventional DVS	$0.44E$	$1.99E$	-	-	$2.43E$
Proposed	$0.69E$	$1.08E$	$0.20E$	-	$1.97E$
Optimal	$0.54E$	$1.03E$	$0.18E$		$1.75E$

does not take the passive power into account or the buffer utilization. It just considers the convex power-frequency relationship and it runs at the smallest frequency/power level that satisfies the job deadlines. This results in the lowest active energy consumption but since the passive power is significant, conventional DVS has the highest passive energy among all others. For conventional DVS, the exact real complexities are assumed to be known beforehand and the jobs are assumed to be finished just in time. Hence, there is not transition to sleep state and hence no transition energy or sleep (idle) energy in conventional DVS scenario. The optimal DVS is, as explained in Section III, based on the dynamic programming solution.

Table IV shows the results for the case where $P_{\text{pass}} = P_{\max}/2$ and sleep (idle) state -1, i.e., the transition energy is assumed as $E_{\text{trans}} = P_{\max}/fr$ and there is no idle power $P_{\text{idle}} = 0$. There is no change in the operation of the conventional DVS and the no-DVS scenario, since they do not adapt to the passive power. However, the proposed method changes the operating frequency and, hence, active and passive time spent.

Table V shows the results for the case where $P_{\text{pass}} = P_{\max}$ and sleep (idle) state -2, i.e., the transition energy is assumed as $E_{\text{trans}} = 0$ and there is no idle power $P_{\text{idle}} = P_{\max}/10$. There is no transition energy in any case.

Table VI shows the results for the case where $P_{\text{pass}} = P_{\max}/2$ and sleep (idle) state -2, i.e., the transition energy is assumed as $E_{\text{trans}} = 0$ and there is no idle power, $P_{\text{idle}} = P_{\max}/10$.

Our results show the superiority of the proposed DVS method over other methods when passive power is significant for different video decoding system specifications, i.e., different passive power and sleep (idle) state cases. The buffer size is set to 20 frames for all of the experiments, which is a reasonable for most devices with power and storage constraints such as mobile devices. Increasing the buffer size only decreases the total transition energy in the proposed method, since the frequency of the transitions decrease with the buffer size.

TABLE VI
COMPARATIVE RESULTS ON ACTIVE, PASSIVE, AND TOTAL ENERGY CONSUMPTION, THE CASE WHERE $P_{\text{pass}} = P_{\max}/2$ AND SLEEP (IDLE) STATE-2

	Active Energy	Passive Energy	Idle Energy	Trans. Energy	Total Energy
No DVS	E	$0.5E$	$0.11E$	-	$3.19E$
Conventional DVS	$0.44E$	E	-	-	$1.44E$
Proposed	$0.46E$	$0.85E$	$0.09E$	-	$1.4E$
Optimal	$0.52E$	$0.73E$	$0.08E$	-	$1.33E$

V. CONCLUSION

We proposed a novel state/frequency scheduling method for video decoding systems with significant passive power. The proposed method utilizes video encoding-specific job definitions and a postdecoding buffer, as the video compression specifications. The optimal solution and the suboptimal low-complexity heuristics are proposed to solve the DVS problem with passive power. We experimentally show the benefit of the proposed method in systems where passive power is significant compared with active power, with several passive power and sleep (idle) state levels.

REFERENCES

[1] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraint," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 645–658, May 2005.

[2] X. Lu, E. Erkip, Y. Wang, and D. G. Goodman, "Power-efficient multimedia communication over wireless channels," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 10, Dec. 2003.

[3] Y. Liang, I. Ahmad, and X. Wei, "Adaptive techniques for simultaneous optimization of visual quality and battery power in video encoding sensors," in *Proc. Int. Conf. Image Process.*, Atlanta, GA, Oct. 2006, pp. 2477–2480.

[4] K. Choi, K. Dantu, W. Cheng, and M. Pedram, "Frame-based dynamic voltage and frequency scaling for a MPEG decoder," in *Proc. ICCAD*, 2002, pp. 732–737.

[5] M. Mesarina and Y. Turner, "Reduced energy decoding of MPEG streams," *Multimedia Syst.*, vol. 9, no. 2, pp. 202–213, 2003.

[6] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Norwell, MA: Kluwer, 1997.

[7] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets, "GRACE: Cross-layer adaptation for multimedia quality and battery energy," *IEEE Trans. Mobile Computing*, vol. 5, no. 7, pp. 799–815, Jul. 2006.

[8] R. Jejurikar, C. Pereira, and R. Gupta, "Leakage aware dynamic voltage scaling for real-time embedded systems," in *Proc. Conf. Design Automation*, 2004, pp. 275–280.

[9] R. Min, M. Bhardwaj, S. Cho, N. Ickes, E. Shih, A. Sinha, A. Wang, and A. P. Chandrakasan, "Energy-centric enabling technologies for wireless sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 28–39, Aug. 2002.

[10] A. Naveh *et al.*, "Power and thermal management in Intel core-duo processor," *Intel Technol. J.*, vol. 10, no. 2, pp. 109–122, 2006.

[11] T. Simunic, "Energy efficient system design and utilization," Ph.D. dissertation, Elect. Eng. Dept., Stanford Univ., Stanford, CA, 2001.

[12] S. Regunathan, P. A. Chou, and J. Ribas-Corbera, "A generalized video complexity verifier for flexible decoding," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2003, vol. 3, pp. 289–292.

[13] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Trans. Multimedia*, vol. 7, no. 3, pp. 471–479, Jun. 2005.

[14] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," *IEEE Trans. Image Process.*, vol. 3, no. 1, pp. 26–40, Jan. 1994.

[15] Intel StrongARM Processors Intel Inc [Online]. Available: http://developer.intel.com/ design/strong/

[16] A. Sinha and A. P. Chandrakasan, "Jouletrack: A web based tool for software energy profiling," in *Proc. IEEE/ACM DAC*, 2001, pp. 220–225.

[17] E. Akyol and M. van der Schaar, "Complexity model based proactive dynamic voltage scaling for video decoding systems," *IEEE Trans. Multimedia*, vol. 9, no. , pp. 1475–1492, Nov. 2007.

[18] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. van der Schaar, J. Cornelius, and P. Schelkens, "In-band motion compensated temporal filtering," *Signal Process.: Image Commun.*, vol. 19, no. 7, pp. 653–673, Aug. 2004.