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—*video compression, buffer control, power optimization for multimedia systems, dynamic voltage scaling.*

I. Introduction

Dynamic voltage scaling (DVS) algorithms were proposed for dynamically adapting the operating frequency and voltage. The main goal of existing 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 [1]. 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 multimedia decoding applications. In [2], a state-of-the-art DVS algorithm for video decoding was presented. 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 seconds. 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. Also, none of the prior research 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. [3] "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".

Idle (sleep) states are especially important when idle state can consume nearly as much power as active state [3]-[6], which necessitates the joint optimization of the idle state scheduling and DVS. Note that, while idle state scheduling for sensor networks [4][6] and leakage-aware (i.e., passive power aware) DVS for real-time applications [3] 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 post decoding buffer to mitigate the highly varying complexity profile of video decoding, not only to be able to decode high complexity frames [7], but also to perform efficient DVS by relaxing the hard job deadlines to soft buffer overflow/underflow constraints. A post decoding buffer is crucial for video decoding applications since bursts of frames should be jointly decoded till 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 as follows. Section 2 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 3. Comparative results are presented in Section 4 and Section 5 concludes the paper.

II. Dynamic Voltage Scaling 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 Figure 1. Let us assume, we have $M = 3$ jobs with complexities $c(1) = \frac{C_{\max}}{2}, c(2) = \frac{C_{\max}}{4}, c(3) = C_{\max}$. From now on, we use the term complexity to represent the number of execution cycles. With no-DVS, 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, the frequencies are adjusted to finish each job "just-in-time", prior to its delay deadline, i.e., $f(1) = \frac{F_{\max}}{2}, f(2) = \frac{F_{\max}}{4}, f(3) = F_{\max}$. In this case, the remaining idle times are neglected. Let us assume an active power-frequency relationship of $P \propto f^3$ [1][2]. Then, the active power will be $p(1) = \frac{P_{\max}}{8}, p(2) = \frac{P_{\max}}{64}, p(3) = P_{\max}$. The total "active" energy spent on the group of jobs can be found as

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

Hence, the total active energy spent for the no-DVS case is $E^{active}_{no\_DVS} = 1.75E$ and for the conventional DVS $E^{active}_{DVS} = 1.14E$, where $E = \frac{P_{\max}.C_{\max}}{F_{\max}}$. 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 non-negligible passive power, we should consider the total energy , i.e., $E^{total} = E^{active} + E^{passive}$. The passive energy can be written as

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

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. The transition energy is the energy spent by going into the active state from the idle state [3][6] . When the idle time t_{idle} is zero, there is no transition energy, captured with the sign function in Eq.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., $E_{trans} = P_{idle} = 0$. Then, if we assume $P_{pass} = P_{\max}$ (which is typical for several devices [3] [6]), the total energies corresponding to no-DVS and conventional DVS are $E^{total}_{no\_DVS} = 3.5E$ and $E^{total}_{DVS} = 4.14E$. 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.

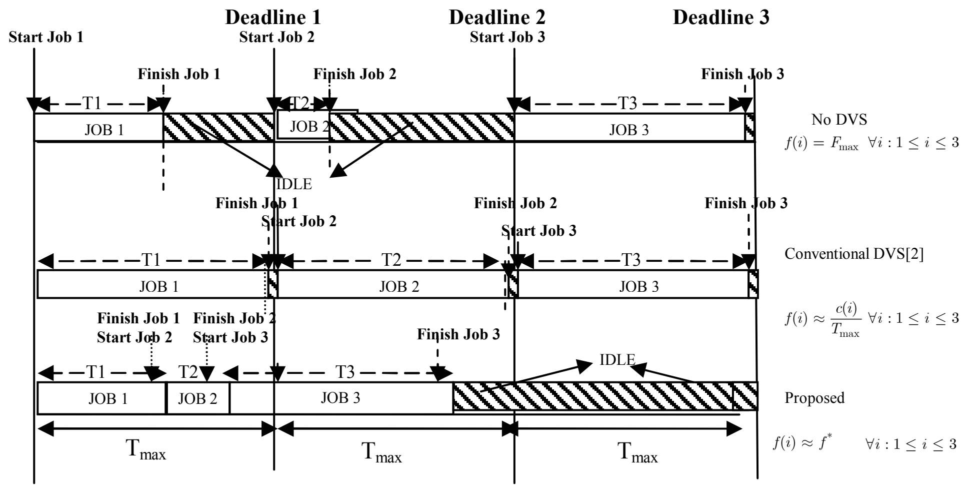


Figure 1: DVS Strategies: 1) No DVS (top) 2) conventional DVS [2] (middle) 3) Proposed DVS with sleep states (bottom)

The bottom panel 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 $E^{total}_{DVS+state\_sch.} = 3.31E$, 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 it 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 $P_{idle} = 0$ but the transition energy E_{trans} is high, the "idle" state where $E_{trans} = 0$ but P_{idle} is high, and the "stand-by" state that has nonzero power consumption and transition energy [6]. In Section 3, 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 [2] 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, $job : \{deadline, complexity, size\}$

$deadline$: 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 eg. [8].

$size$: Number of decoded *original* frames when job j finishes, $s(j)$.

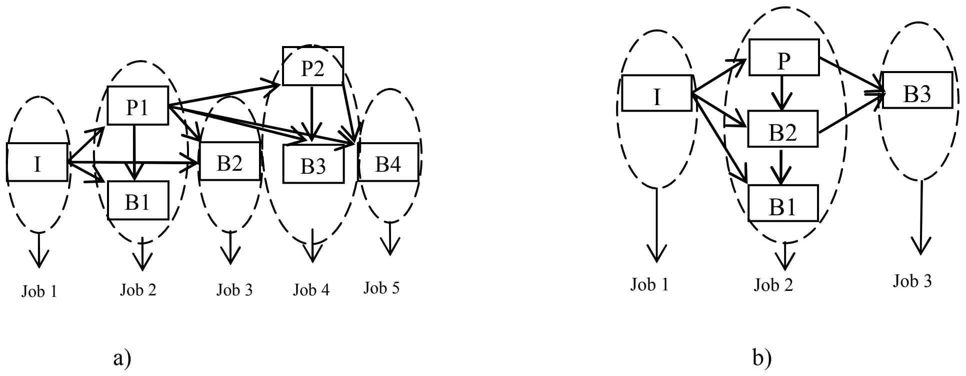


Figure 2: DAGs and job definitions for a) conventional I-B1-B2-P1-B3-B4-P2 frames b) Hierarchical B Pictures, I-B1-B2-B3-P

In predictive coding, frames are encoded with interdependencies that can be represented by a directed acyclic dependence graph (DAG). Examples are shown in Figure 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 Figure 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 above illustrative examples).

Another video compression specific feature of the proposed DVS framework is the usage of a post-decoding (display) buffer between the display device and the decoding platform. This post decoding buffer helps to mitigate complexity estimate mismatches and converts hard job deadlines to soft buffer underflow/overflow constrains. We neglect the overhead of buffer management energy cost throughout the paper.

Figure 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 and 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.

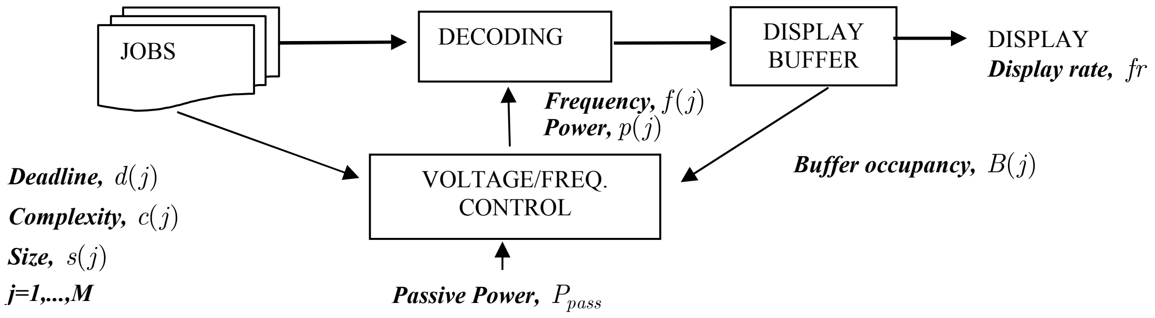


Figure 3: Buffer controlled DVS for video decoding

III. Proposed DVS Algorithm

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. Let us assume that there is a discrete set of operating levels with corresponding frequency and power levels, which can be used for voltage/frequency adaptation $\mathbf{L} = \{l_j : (p_j, f_j), 1 \le j \le N \mid l_1,..,l_N\}$. Each level has a different power consumption and different frequency, $(\mathbf{P},\mathbf{F}) = \{(p_1, f_1),...,(p_N, f_N) \mid p_1 < ... < p_N; f_1 < ... < f_N\}$, 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 $\mathbf{D} = \{d(1),...,d(M) \mid d(1) < ... < d(M)\}$ and sizes $\mathbf{S} = \{s(1),...,s(M)\}$. Then, the dynamic voltage scaling problem attempts to find the set of operating level (frequency and thus power) for each job $\mathbf{l}^{\mathbf{opt}} = \{l(1),...,l(M) \mid \forall l \in \mathbf{L}\}$. Throughout the paper, we assume the passive power is independent of the frequency [1] [2], P_{pass}. Thus, the investigated DVS optimization can be formulated as follows.

Buffer Constrained DVS Optimization considering Passive Power:

$$\mathbf{l}^{\mathbf{opt}} = \arg\min_{l \in L} \sum_{j=1}^{M} \{ p(j)t(j) + P_{pass}t(j) \} \qquad \text{(energy consumption)}$$

$$\text{subject to: } 0 \le B(j) \le B_{\max}\ ;\ \forall j : 1 \le j \le M \ \text{ (buffer overflow/underflow constraint)} \qquad (3)$$

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

$$B(j) = \max(B(j-1) + s(j) - t(j).fr, 0) \text{ and } B(0) = B_{initial} \tag{4}$$

where fr denotes the frame rate, $B_{initial}$ 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 processing time can be defined as:

$$t(j) = \frac{c(j)}{f(j)}. \tag{5}$$

The optimal frequency $f^*(j)$ is defined as: $$f^*(j) = \arg\min_{f \in F} E(j), \tag{6}$$

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

$f^*(j)$ can be determined by solving the differential equation: $$\left.\frac{\partial E(j)}{\partial f(j)}\right|_{f^*(j)} = 0. \tag{8}$$

By replacing Eq. 5, and Eq. 7 in Eq. 8, we obtain the optimal frequency as the frequency that satisfies the differential equation:

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

The optimization strategy depends on the value of the set of operating frequencies with respect to $f^*(j)$. As can be seen from Equation (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 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: a) $f_{\min} \le f^*$ (case-1) and b) $f_{\min} > f^*$ (case-2). In our subsequent derivations, we assume a power-frequency function as in [1][2]: $p(f) = \alpha.f^k$ 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:

$$\frac{\partial E}{\partial f} = \frac{c}{f^2}(\alpha(k-1)f^k - P_{pass}) \tag{10}$$

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

From Eq.9, we obtain that the optimal frequency is $f^* = \sqrt[k]{\frac{P_{pass}}{\alpha(k-1)}}$ (12)

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

Case-1 ($f_{\min} \leq f^*$): 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 [9]. Also, the processor can go into different type of idle (sleep) states, such as the sleep-1 that consumes negligible power at idle state (P^1_{idle}) but results in a significant energy consumption in state transitions(E^1_{trans}), and sleep-2 that costs significant power at idle states but has less transition cost. Let us assume sleep-1 has energy transition cost E^1_{trans} , but no steady state cost. Sleep-2 has transition cost E^2_{trans} ($E^2_{trans} < E^1_{trans}$) and idle state cost P^2_{idle} , where $P^1_{sleep} < P^2_{sleep}$. Let $t_{idle}(j)$ be the time that 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 following optimization problem.

Buffer Constrained DVS Optimization with Passive Power and Sleep States:

$$\mathbf{l^{opt}}, \mathbf{t}_{idle}, \mathbf{state} = \arg\min_{l \in L} \sum_{j=1}^{M} \left\{ p(j)t(j) + P_{pass}t(j) + P^{state}_{idle} t_{idle}(j) + E^{state}_{trans} \mathrm{sign}(t_{idle}(j)) \right\}$$

subject to: $0 \leq B(j) \leq B_{\max}$; $\forall j : 1 \leq j \leq M$, where $B(j) = \max(B(j-1) + s(j) - (t(j) + t_{sleep}(j)).fr, 0)$ (13)

In the above formulation, $\mathbf{l^{opt}}, \mathbf{t}_{idle}, \mathbf{state}$ 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.

The optimal solution to these types of constrained problems can be found by deploying dynamic programming based algorithms which are impractical due to their high complexity. Hence, 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 sleep state till 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 [9]. Idle (sleep) state type (sleep-1 or sleep-2) is determined according to the buffer occupancy and complexities of the jobs.

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 energy spent in the idle state-1 and idle state-2 (transition energy+ passive energy in idle state) for one buffer depletion time are $E_1 = E_{trans}^1 + \frac{(\beta_2 - \beta_1)}{fr} P_{idle}^1$ and $E_2 = E_{trans}^2 + \frac{(\beta_2 - \beta_1)}{fr} P_{idle}^2$, 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., $\frac{\left(E_{trans}^1 - E_{trans}^2\right)}{(\beta_2 - \beta_1)} fr < P_{idle}^2 - P_{idle}^1$. Sleep-2 should be chosen when $\frac{\left(E_{trans}^1 - E_{trans}^2\right)}{(\beta_2 - \beta_1)} fr > P_{idle}^2 - P_{idle}^1$.

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

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

Since we keep the frequency constant for all jobs, we can write $\sum_{j=1}^{M} t(j) = \frac{\sum_{j=1}^{M} c(j)}{f}$,

(15)

and

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

The number of transitions is identical to the number of buffer depletions, which is $\frac{\sum_{j=1}^{M} t_{idle}(j)}{(\beta_2 - \beta_1)} fr$. Hence, the total energy can be written as:

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

Similar to Eq.9, the operating f_{opt} frequency can be found as the frequency that satisfies the differential equation

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

As can be seen from the operating frequency depends on 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. The algorithm is given in Table I.

Table I: Low Complexity Algorithm

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

Case-2 ($f_{\min} > f^*$): Since all 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. [9]). Hence, the interested reader is referred to existing literature [9] for details on finding the solution of this problem.

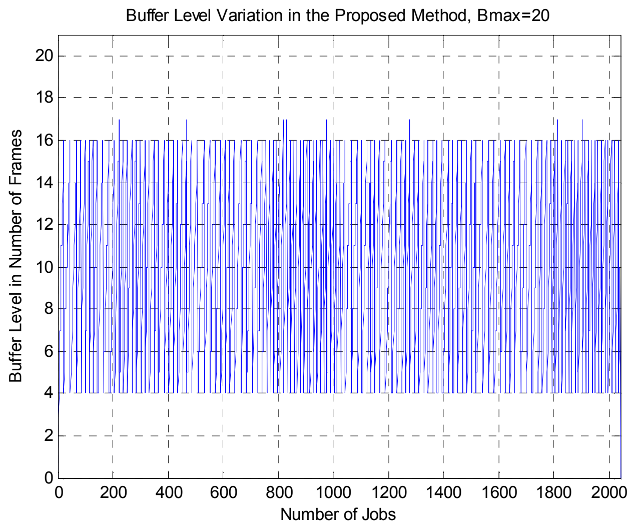
IV. Results

In this section, we compare the proposed DVS method to the conventional DVS method. In our experiments we used four different test sequences, *foreman, mobile, coastguard* and *silence,* at CIF resolution and at a frame rate of 30fps. We used a wavelet video coder that utilizes motion-compensated temporal filtering with 4 level temporal and 5 level spatial decompositions. We generated two sets of decoded video with high and low complexity at two rates, 512 kbps and 1024kbps. High complexity compression includes adaptive update and quarter pixel motion compensation whereas the update step is skipped and only 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. Power and frequency values that we used are shown in Table II, which are reported for the Intel StrongARM processor [10] in [11].

Table II: Frequency power values of Strong-Arm processor analyzed in [11]

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 a passive power $P_{pass} = P_{\max}$, and transition energy $E_{trans} = P_{\max} / fr$ in the experiments. Two buffer sizes are tested, we set the buffer size to 20 decoded frames, $B_{\max} = 20$, for Proposed DVS-1 and $B_{\max} = 50$ for Proposed DVS-2. We set $\beta_1 = 0.2B_{\max}$, $\beta_2 = 0.8B_{\max}$ and $P_{idle} = P_{\max} / 10$ for the experiments.



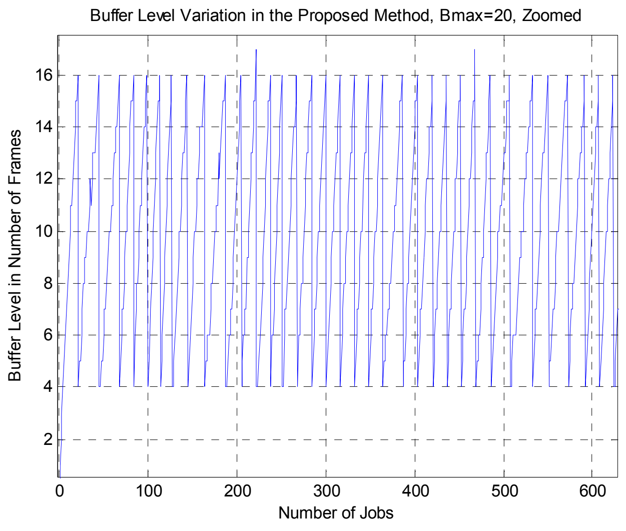


Figure 4: Buffer level variation for one realization of test sequences

Figure 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 $\beta_1 = 4$. The comparative results are given in Table III.

Table III: Comparative (scaled) results on active, passive and total energy consumption

	Active Energy	Passive Energy	Idle Energy	Trans. Energy	Total Energy
No DVS	E	E	$0.11E$	$1.08E$	$3.19E$
Conventional DVS	$0.44E$	$1.99E$	-	-	$2.43E$
Proposed DVS-1	$0.68E$	$1.07E$	$0.03E$	$0.01E$	$1.88E$
Proposed DVS-2	$0.68E$	$1.07E$	$0.03E$	$0.003E$	$1.87E$

Our results show the superiority of the proposed DVS method over other methods when passive power is significant. As the buffer size increases the transition energy decreases significantly. Hence, increasing the buffer size may help decreasing the total energy when the transition energy is significant. There is no transition and idle energy cost assumed for Conventional DVS, since this technique assumes to finish jobs just in time, i.e., exact complexities are assumed to be known beforehand only for Conventional DVS.

V. Conclusion

We proposed a novel state/frequency scheduling method for video decoding systems. The proposed method utilizes video encoding specific job definitions and a post-decoding buffer. We experimentally show the benefit of the proposed method in systems where passive power is significant compared to active power.

References

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

[2] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets, “GRACE: Cross-layer adaptation for multimedia quality and battery energy,” to appear in *IEEE Transactions on Mobile Computing*

[3] R. Jejurikar, C. Pereira, and R. Gupta. Leakage Aware Dynamic Voltage Scaling for Real-Time Embedded Systems. in *Proc. Conf. on Design Automation*, pages 275–280, 2004.

[4] 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 Communications Magazine* , pp. 28-39, August 2002

[5] A. Naveh et al,“Power and Thermal Management in Intel Core-Duo Processor”, *Intel Technology Journal,* Volume 10, Issue 2, 2006

[6] T. Simunic, “Energy efficient system design and utilization”, Ph.D. Thesis, Stanford University, 2001

[7] S. Regunathan, P. A. Chou, and J. Ribas-Corbera, “A generalized video complexity verifier for flexible decoding,” *Proc. IEEE International Conference on Image Processing*, vol. 3, pp. 289-292, Sept. 2003.

[8] M. van der Schaar and Y. Andreopoulos, “Rate-distortion-complexity modeling for network and receiver aware adaptation,” *IEEE Trans. on Multimedia*, vol. 7, no. 3, pp. 471-479, June 2005

[9] A. Ortega, K. Ramchandran, and M. Vetterli, “Optimal trellis-based buffered compression and fast approximations,” *IEEE Trans. on Image Processing*, Vol. 3, No. 1, Jan. 1994.

[10] Intel Inc, “Intel StrongARM processors,” Available: http://developer.intel.com/ design/strong/

[11] A. Sinha and A. P. Chandrakasan, “Jouletrack: A web based tool for software energy profiling,” in *Proc. IEEE/ACM DAC*, 2001.