Buffer Constrained Proactive Dynamic Voltage Scaling for Video Decoding Systems

Emrah Akyol, Mihaela van der Schaar
{eakyol, mihaela}@ee.ucla.edu
Electrical Engineering, University of California, Los Angeles, CA, USA

ABSTRACT

Significant power savings can be achieved on voltage/frequency configurable platforms by dynamically adapting the frequency and voltage according to the workload (complexity). Video decoding is one of the most complex tasks performed on such systems due to its computationally demanding operations like inverse filtering, interpolation, motion compensation and entropy decoding. Dynamically adapting the frequency and voltage for video decoding is attractive due to the time-varying workload and because the utility of decoding a frame is dependent only on decoding the frame before the display deadline. Our contribution in this paper is twofold. First, we adopt a complexity model that explicitly considers the video compression specifics to accurately predict execution times. Second, based on this complexity model, we propose a dynamic voltage scaling algorithm that changes effective deadlines of frame decoding jobs. We pose our problem as a buffer-constrained optimization and show that significant improvements can be achieved over the state-of-the-art dynamic voltage scaling techniques without any performance degradation.

Index Terms—*Video decoding, dynamic voltage scaling, complexity modeling, energy saving*

1. INTRODUCTION

Power-frequency reconfigurable processors are already available in wireless and portable devices. Recently, hardware components are being designed to support multiple power modes that enable trading off execution time for energy savings. For example, some mobile processors can change the speed (i.e., frequency) and energy consumption at runtime [1]. Significant energy savings can be achieved by adapting the voltage and processing speed for the tasks where early completion does not provide any gains. The energy spent on one process can be reduced by decreasing the voltage, which will correspondingly increase the delay [2].The main goal of all DVS algorithms is utilizing this energy-delay trade off for tasks whose jobs' completion times are immaterial as long as they are completed before their processing deadline An example is real-time video decoding, where an early completion of frame decoding does not provide any benefit as long as the display deadline for that frame is met. A DVS algorithm essentially assigns the operating level (i.e., power and frequency) for each job given the estimated cycle requirement (i.e., the required complexity) and the job deadline [2].

Recently, several researchers have addressed the problem of efficient DVS for multimedia applications. In [3], rather than using the worst case complexity, a worst case estimate satisfying a statistical guarantee determined based on the online profiled histogram of the previous frames is used for the frequency assignment of each job. Also, an intra-job DVS is proposed by gradually increasing the frequency (speed) within the job, while monitoring the instantaneous cycle demands. For other works on DVS, see [7] and the references there in.

For video decoding, a job represents conventionally the decoding of a frame. Based on the frame rate, there is a worst-case design parameter, $T_{\max}$ that denotes the amount of time available for processing a job. Depending on the time-varying characteristics of the video content, the deployed compression algorithm and encoding bit-rate, not every job needs the entire $T_{\max}$ to complete its execution.

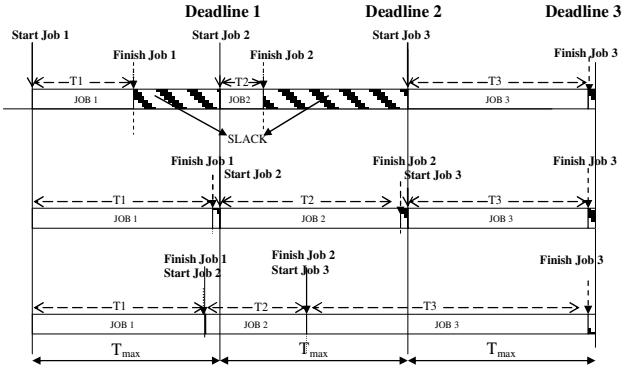

Figure 1: The top panel is no-DVS, middle panel is conventional DVS and the bottom panel is proposed DVS method.

As shown in the top panel of Figure 1, the first job needs T1 time units whereas the second job needs T2 time units. The key idea behind existing adaptive systems is to adapt the operating frequency such that the job is processed exactly in $T_{\max}$ time units (see the middle panel of Figure 1). Clearly, this shows the reactive and greedy nature of the existing adaptation process – the resources are optimized within a job for a fixed time allocation $T_{\max}$. However, the DVS gains can be substantially improved when the allocated times (T1, T2, and T3) and operating levels (power-frequency pairs) are optimized jointly (inter-job optimization) as shown in the bottom panel, Figure 1.

Assume we have three 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. With "no-DVS", processing is performed considering the worst case scenario: at the maximum frequency for each job $F_{\max}$ and corresponding maximum power $P_{\max}$. For "conventional DVS", frequencies are adjusted to finish each

job just-in-time, $f(1) = F_{\max}/2$, $f(2) = F_{\max}/4$, $f(3) = F_{\max}$. If we assume a power-frequency relationship of $P \propto f^3$ [2] then the power spent on the various jobs will be $p(1) = P_{\max}/8$, $p(2) = P_{\max}/64$, $p(3) = P_{\max}$. For the "proactive optimization" with modified deadlines, the total complexity is $c(1) + c(2) + c(3) = 7C_{\max}/4$, and frequencies are kept constant: $f(1) = f(2) = f(3) = 7F_{\max}/12$. The total energy spent on the group of jobs equals $E_{total} = \sum p(i).c(i)/f(i)$. Hence, the total energy spent for the "no-DVS" case is $E_{no-dvs} = 7E/4$, for the "conventional DVS" equals $E_{dvs} = 73E/64$ and for the "proactive DVS" case equals $E_{proactive\_dvs} = 343E/72$, where $E = P_{\max}.C_{\max}/F_{\max}$. In this example, conventional DVS provides 35% energy reduction with respect to no-DVS, whereas proactive DVS provides 66% energy reduction with respect to no-DVS.

As mentioned before, none of the previous studies consider *proactively* changing the time allocations and frequency; instead, they aim at adapting the frequency to *fixed time allocations* in a greedy fashion. We propose a novel DVS algorithm that adapts jobs deadlines by buffering the decoded frames before display. By utilizing this post-decoding buffer, we study the DVS problem into the context of the buffer constrained optimization problem, similar to well studied problems of rate-control with buffer constraints. We also propose an optimal and several low-complexity suboptimal solutions for the buffer-constrained DVS problem.

This paper is organized as follows: Section II describes the compression aware job definitions. The proposed DVS algorithms are explained in Section III. Section IV presents the comparative results and Section V concludes the paper.

2. COMPRESSION AWARE JOB DEFINITIONS

The state of the art video encoders deploy complex temporal predictions from both past and/or future frames which must be decoded much before their display deadline. Hence, each frame has a *decoding deadline* that is determined by the temporal decomposition structure (temporal dependencies). This deadline is different from the *play-back (display) deadline* determined by the frame rate *fr*. Let $\Re(n)$ be the set of frames for which frame *n* is used as a reference. Then, the display and decoding deadlines for frame *n* can be written: $deadline_{display}(n) = n/fr$,

$deadline_{decoding}(n) = \min\{deadline_{display}(i)\} : \forall i \in \Re(n)$.

Unlike previous work that considers the decoding of each individual frame as a task, we combine frames having the same *decoding deadline* (i.e., frames that are dependently encoded) into *one job* of the decoding task. We define every job based on three parameters: $job : \{deadline, complexity, size\}$, where

$deadline$: Decoding deadline of job *j*, $d(j)$

$complexity$: Estimated number of cycles that job *j* consumes on a specific platform, $c(j)$, see eg.[4] for details.

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

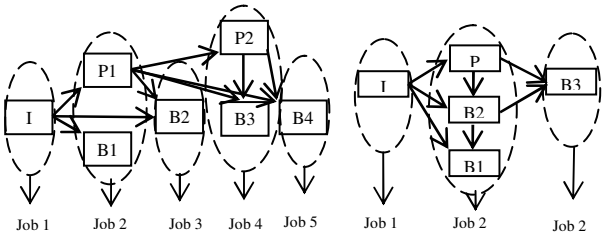

Figure 2: Directed acyclic dependence graphs for a) Dependencies between I-B1-B2-P1-B3-B4-P2 frames b) Hierarchical B Pictures, I-B1-B2-B3-P

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 and the hierarchical B pictures. Decoding frame I is a job and decoding frames P1 and B1 jointly represent another job. Prediction structures using hierarchical B pictures as in the H.264/AVC standard lead to the following sizes (s), complexities (c), and deadlines (d): the first job represents the decoding of the I-frame ($s(1) = 1, d(1) = \Delta t$) the second job consists of decoding frames P, B1 and B2 ($s(2) = 3, d(2) = 2\Delta t$) and the last job is the decoding of frame B3 ($s(3) = 1, d(3) = 4\Delta t$). It is important to notice that, both the second and the third job can be viewed from a high level perspective as decoding a B frame. However, the job parameters are substantially different, thereby highlighting the need for encoder-specific complexity estimation.

3. BUFFER CONSTRAINED DVS

Let us assume there is a discrete set of operating levels with corresponding frequency and power levels which can be used in our frequency/voltage adaptation. Each level has a different power consumption and different frequency $(\mathbf{P},\mathbf{F}) = \{(p_1,f_1),...,(p_N,f_N) \mid p_1 < ... < p_N; f_1 < ... < f_N\}$. Assume there are a total of M jobs with deadlines $D = \{d(1),...,d(M) \mid d(1) < .. < d(M)\}$, sizes $S = \{s(1),...,s(M)\}$ and complexity estimates $\mathbf{C} = \{c(1),...,c(M)\}$. The DVS problem attempts to find the set of operating levels (power and frequency tuple) for each job, as follows:

DVS Problem:

$$\text{find } (f^{opt}, p^{opt}) = \arg\min_{(P,F)} \sum_{i=1}^{M} E(i) \quad \text{(energy consumption)}$$

$$\text{s.t.} \sum_{m=1}^{j} c(m)/f(m) < d(j) \;\; \forall j : 1 \le j \le M \text{ (delay constraints)}$$

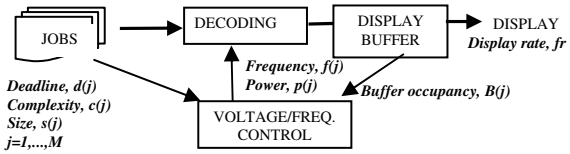

Figure 3: Proposed buffer controlled DVS

We propose to use a post-decoding buffer between the display device and the decoding platform as shown in Figure 3. The operating level of the job *j* (*p-f* tuple) is determined considering

the parameters of *M* jobs and buffer occupancy *B(j)* . For each job, the complexity estimates are updated and based on the buffer occupancy, a new operating level is assigned. We define the buffer occupancy for job j as $B(j)$ recursively

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

where *fr* denotes the frame rate, $B_{initial}$,the initial state of the buffer, depends on the initial playback delay(which may be zero if no delay is tolerable), and $t(j)$ is the time that job *j* takes which can be written as $t(j) = c(j) / f(j)$ .Then, the DVS problem becomes the problem of minimizing total energy under buffer constraints.

Buffer Constrained DVS Problem:

$$\text{find} \quad (f^{opt}, p^{opt}) = \arg\min_{(P,F)} \sum_{i=1}^{M} E(i) \quad \text{(energy consumption)}$$

$$\text{s.t. } 0 \le B(j) \le B_{\max} \quad \text{(buffer constraints)}$$

We need to guarantee the buffer never underflows, such that no frame freeze occurs. Also, the buffer state can not grow indefinitely because it is a finite physical buffer. If we assume the maximum buffer size is $B_{\max}$, we need to guarantee that the buffer occupancy at any state is lower than $B_{\max}$.

The optimal solution can be found by dynamic programming methods, which explicitly consider every possible frequency-power pair for each job and check the buffer state for overflow or underflow. A trellis is created with given complexity estimates of the job and possible power-frequency assignments. At every stage, the paths reaching the same buffer occupancy at a higher energy cost are pruned.

Although there is a finite number of operating levels, intermediate operating levels are achievable by changing the frequency/power within the job similar to the approach in [2] [3]. Figure 4 shows the energy-delay curve for jobs *j* and *j+1* Intermediate levels are achieved by frequency changes within the job. Greater values of the slope means greater energy spent with smaller delay yielding a higher frequency-power choice.

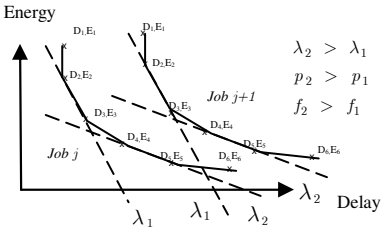

Figure 4: Operational Energy-Delay curve of jobs *j* and *j+1* with D,E pairs corresponds to different frequency-power levels of the processor.

Proposition I: If we neglect the transition cost from one frequency to another, the frequency change within the job corresponds to piece-wise bilinear interpolation power-frequency points as shown in Figure 4. **Proof:** See [7].

Proposition II: The slope between two E-D points only depends on the power/frequency values but not on complexities. **Proof:** See [7].

Proposition III: From a set of given power-frequency points, only the ones that generate a convex set of E-D points should be used for proactive DVS. **Proof:** See [7].

Hence, before any optimization, the power-frequency values which do not provide complex E-D points should be pruned, i.e., a convex hull of E-D points, from a possible set of E-D points should be generated. Since the slopes are identical for all jobs (Proposition II), this pruning is done only once for all jobs.

Proposition IV: The optimal operating level assignment will result in equal slopes for every job. **Proof:** See [7].

Buffer constrained DVS problem is analogous to buffer constrained R-D optimal bit allocation problem [5]. Hence, similar to rate (buffer)-control problems in video transmission, we aim to keep the buffer state in equilibrium ($B_{\max} / 2$) for a group of *W* jobs. Then, using Propositions II and IV, the optimal frequency considering a look-ahead window of *W* jobs:

$$f^{opt}(j) = \frac{fr. \sum_{m=j}^{j+W} c(m)}{B(j) - \frac{B_{\max}}{2} + \sum_{m=j}^{j+W} s(m)}$$

Algorithm 1 performs the delay-energy optimization for every job considering a look ahead window of *W* jobs.

Table I: Algorithm 1

1. Choose the set of frequency-power tuples which creates convex E-D points
2. For each job j $(1 \le j \le M)$, find the optimum frequency
3. Proceed to job $j+1$ (Step 2)

A fast extension of this algorithm, Algorithm 2, is to perform this optimization and assign the same frequency for W jobs and re-perform this optimization only when the buffer level gets lower or higher than the specified thresholds.

Table II: Algorithm 2

1. Choose the set of frequency-power tuples which creates convex E-D points
2. For each look-ahead window of size W
3. For job j within the window find the optimum frequency
4. Execute the job with the assigned frequency and check buffer state $B(j)$
5. If $B(j) \le \beta_1$ or $B(j) \ge \beta_2$ change the frequency (Step 3) else continue with same frequency (Step 4)

A least complex approximation, Algorithm 3, is changing the frequency only when there is a risk of overflow or underflow.

Table III: Algorithm 3

1. Choose the set of frequency-power tuples which creates convex E-D points.
2. For job j find the optimum frequency
3. Execute the job with the assigned frequency and check the buffer state
4. Set $j = j + 1$. If $B(j) \le \beta_1$ or $B(j) \ge \beta_2$ change the frequency (Step-2) else continue with same frequency (Step 3)

4. COMPARATIVE RESULTS

In this section we compare the proposed DVS method to conventional DVS methods. We used four different test sequences, *foreman, mobile, coastguard* and *silence,* 256 frames at CIF resolution and frame rate 30fps, encoded at two different compression specifications (low complexity and high complexity), and decoded at 2 different rates, 512kbps and 1024kbps. 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 are taken from [6] .

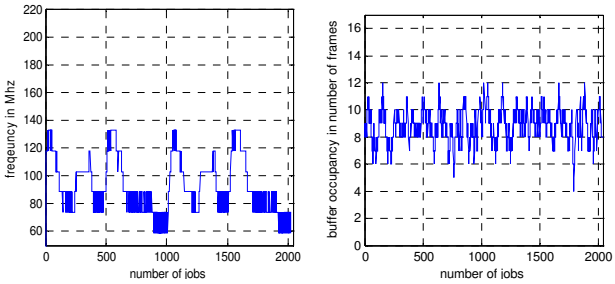

Figure 5 : Frequency assignment (a) and buffer fullness (b) for Algorithm 1

In the following, we compare the fast algorithms, Algorithm 1, 2, and 3 in terms of the buffer utilization, the number of frequency transitions and energy savings. The highest number of frequency changes occurs in Algorithm 1 compared to other algorithms whereas the least buffer level variation is observed, as shown in Figure 5. Algorithm 3 is the most conservative in terms of the number of frequency changes and as expected provides the least energy savings among the proposed three algorithms. The performance of Algorithm 2 is between that of Algorithm 1 and 3 in the sense of energy savings, buffer utilization, and the number of frequency changes.

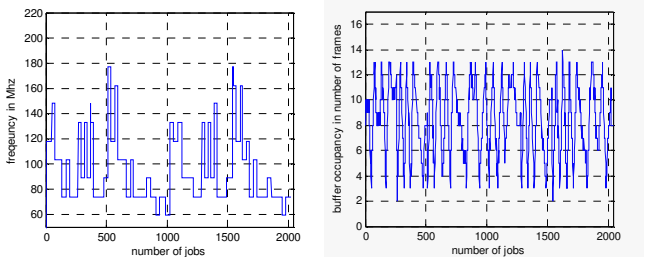

Figure 6: Frequency assignment (a) and buffer fullness (b) for Algorithm 3

As the buffer size and the look-ahead window size increases, energy savings of the proposed buffer controlled DVS algorithms increase as expected intuitively. The gap between the energy savings of Algorithm 1 and Algorithm 3 decrease as the buffer size gets large. This result shows the trade off between the buffer size, energy savings and number of frequency changes. If the buffer size is large, energy savings close to the savings of Algorithm 1 can be achieved with less frequency transitions using Algorithm 3. Conversely, when the buffer size is small, the difference in energy savings of Algorithm 3 and Algorithm 1 can be significant. We also simulated the conventional DVS method in two ways. The first method, utilizes a statistical worst case complexity estimate denoted here as *Conventional* [3]. Another method is the optimum conventional DVS for benchmark purposes which is named as *Conventional Lower Bound*. We assumed the complexities of each job are known apriori and find the optimum theoretical DVS gain achievable by conventional DVS methods. *Optimal Proactive* is the optimal dynamic programming based algorithm assuming an exact knowledge of complexities before the actual decoding is performed. Optimal Proactive DVS shows the limit of energy savings given the buffer size.

Table IV: Comparison of different DVS algorithms in terms of scaled energy spent and number of frequency changes per job. Benchmark (%100 energy spent case) is no-DVS.

	$B_{\max} = 8$ $W = 8$		$B_{\max} = 16$ $W = 16$	
	Scaled Energy	# freq. changes	Scaled Energy	#of freq. changes
Conv. Bound	45.21	1519.8	45.21	1519.8
Conventional	60.43	1688.5	60.43	1688.5
Optimal Proactive	36.93	1753.7	36.63	1747.3
Algorithm 1	40.36	655.91	40.23	545.67
Algorithm 2	40.60	161.0	40.58	82.75
Algorithm 3	42.35	121.17	41.66	57.92

As the results show, all of the proposed methods significantly out-perform the conventional algorithms in terms of energy savings and the number of frequency changes. Note that, the proposed method provides energy savings exceeding the upper bound of the conventional methods which is based on the exact knowledge of the complexity. This result clearly shows the superiority of the proposed proactive DVS method.

5. CONCLUSION

In this paper we proposed a novel DVS method for video decoding which achieves energy savings exceeding the upper bound of the conventional DVS methods. We explored the fundamental energy vs. delay trade-off for video decoding systems and proposed new trade-offs such as buffer size vs. energy savings and the number of frequency changes that the platform can handle.

6. REFERENCES

[1] Intel Inc, "Intel XScale Technology," Available online http://www.intel.com/ design/intelxscale

[2] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proceedings of Intern. Sym. on Low-Power Electronics and Design*. Monterey, CA 1998.

[3] W. Yuan et.al., "GRACE: Cross-layer adaptation for multimedia quality and battery energy," *IEEE Trans. on Mobile Comp.,*to appear

[4] 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.

[5] 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.

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

[7] E. Akyol, M. van der Schaar, "Complexity model based proactive dynamic voltage scaling for video decoding systems ", IEEE *Trans. on Multimedia*, to appear 2007