Optimality and Improvement of Dynamic Voltage Scaling Algorithms for Multimedia Applications

Zhen Cao, Brian Foo, Lei He *Senior Member, IEEE,* Mihaela van der Schaar, *Senior Member, IEEE*

***Abstract*—The high complexity and time-varying workload of emerging multimedia applications poses a major challenge for dynamic voltage scaling (DVS) algorithms. While many DVS algorithms have been proposed for real-time applications, an efficient method for evaluating the optimality of such DVS algorithms for multimedia applications does not yet exist. In this paper, we propose the first offline linear programming (LP) method to determine the minimum energy consumption for processing multimedia tasks under stringent delay deadlines. Based on the obtained energy lower bound, we evaluate the optimality of various existing DVS algorithms. Furthermore, we extend the LP formulation in order to construct an online DVS algorithm for real-time multimedia processing based on robust sequential linear programming. Simulation results obtained by decoding a wide range of video sequences show that, on average, our online algorithm provides a scheduling solution which requires less than 0.3% more energy than the optimal lower bound with only 0.03% miss rate. In comparison, a very recent algorithm consumes roughly 4% more energy than the optimal lower bound at the same miss rate.**

***Index Terms*—Dynamic voltage scaling, energy management, linear programming, multimedia communication, scheduling, system modeling.**

I. INTRODUCTION

DUE to the popularity of streaming multimedia applications on mobile and pervasive computing devices, computationally intensive multimedia applications must often be processed by energy-limited systems. Dynamic voltage scaling (DVS)-enabled processors are particularly attractive for such devices, since they can adapt their voltage level and associated clock frequency in real time to save energy while handling time-varying workloads and display deadlines [1][2]. In general, a DVS-enabled processor can conserve energy by reducing its voltage level; however, decreasing the voltage level will also slow the processor clock speed, thereby increasing the processing time, and hence the overall delay [2][3][4]. DVS algorithms attempt to find a dynamic balance between the operating level (i.e. power and frequency) of the processor, and the quality-of-service for multimedia applications in terms of meeting stringent delay deadlines.

A. Existing Works

A wide variety of DVS algorithms have been proposed for delay-sensitive applications [5] - [14]. Earlier DVS algorithms perform optimization over one or two tasks, considering either the worst case execution time (WCET), or the average case execution time (ACET) [6][9]. The performance of these approaches is limited because future tasks with imminent deadlines may require extremely high processing power to finish in time. Alternatively, a stochastic soft real-time scheduler was proposed to increase the voltage level adaptively, as long as the soft deadline is met in the worst case [7]. However, this is based upon the assumption that all jobs follow the same complexity distribution, which is rarely the case for multimedia applications. Hence, setting periodic soft deadlines and using the same complexity model for all jobs can be suboptimal.

Another category of DVS algorithms considers joint power scheduling based on multiple job deadlines. LaEDF [5] attempts to process tasks at the lowest frequencies and tries to defer jobs such that the minimum amount of work is done while ensuring that all future deadlines will still be met. Some approaches employ feedback control or adaptive linear prediction to estimate the complexity of future jobs [8][10]-[12], which take advantage of temporal-correlations and patterns inherent in multimedia jobs. Some DVS approaches also employ application-based feedback to the operating system instead of expected statistical behavior [29], and consider energy consumption for both microprocessor and memory devices [32] or the whole system [23][34]. Scalable scheduling approaches also exist [11][28], where the number of tasks released for execution (and hence, the number of deadlines to consider) can be controlled by adjusting various parameters, such as the "aggressiveness" factor in [11]. To improve the performance of application-aware DVS algorithms, in our prior work [13], we proposed the construction of stochastic multimedia complexity models, where different video frames and sequence types are classified into different sets of complexity distributions. The parameters of the distributions can be transmitted in advance and used to analytically approximate the delay for processing each frame at different processor operating levels, thus enabling the system to adapt the processor voltage in real-time. In [15], a technique combining intra- and inter-task voltage scheduling is proposed. However, the optimal voltage schedule solutions proposed are

This work was partially supported by NSF CCR-0306682, NSF CCF 0541453, and NSF CNS-0509522. Address comments to lhe@ee.ucla.edu.

Z. Cao, L. He and M. van der Schaar are with are with Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: caoz@ucla.edu, {lhe,mihaela}@ee.ucla.edu).

B. Foo, was with Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA. He is now with research department of Lockheed Martin, Sunnyvale, CA, USA.

only optimal statistically.

In the existing works, online DVS algorithms are evaluated by experimental comparisons with other online algorithms. However, there has not been a low-complexity approach to determine how far these algorithms are from the *optimal* power scheduling scheme. A few studies have provided methods for computing the optimal offline scheduling problem, such as solving an integer linear program (ILP) [21], or a dynamic programming problem [12]. However, in these works, the complexity grows super-polynomially with the number of jobs considered. This intractability results from certain assumptions, such as the voltage switch overhead being significant compared to the complexity required for processing each job, and thus voltage switch should not be used within a job. However, this assumption is not necessary if the multimedia job complexities are very high compared to the switch overhead, which is usually the case for state-of-the-art video coders.

Furthermore, leakage current in CMOS circuits today contributes a significant portion to total power consumption. Leakage current is expected to increase five-fold with each generation [16]. Hence, leakage power in DVS problems has been studied intensively [16] - [19]. When technologies such as power gating are used to reduce leakage power, the zero power and frequency of sleeping mode should be considered in a DVS algorithm and it is possible that the power-frequency function for processors could be non-convex. In this case, existing works [6][8][13][17] that attempt to minimize idle periods under the assumption of a convex power-frequency function will be no longer effective. Hence, adaptive DVS algorithms and efficient analysis of optimality for both convex and non-convex power-frequency functions are needed.

B. Contributions of This Paper

The contributions of this paper are as follows: first, we analyze the optimality of DVS algorithms by deriving a lower bound for energy consumption, subject to processing all jobs before their delay deadlines (i.e. zero miss rate). We propose a linear programming (LP) DVS solution to obtain the optimal offline scheduling solution for both convex and non-convex power-frequency functions. Unlike the integer programming formulation presented in [21] for temperature-aware DVS scheduling, we take advantage of the fact that the delay overhead of voltage switch is negligible compared to the high multimedia job complexities. Based on the workload traces collected during execution time, we solve the offline LP problem to obtain the lower energy bound for DVS algorithms. A thorough investigation of video decoding results (where many video sequences are decoded at many different bit rates) shows that, under the same zero miss rate, laEDF [5] consumes approximately 15% more energy than the optimal solution, and our prior queuing based algorithm in [13] consumes approximately 4% more than the optimal solution.

Second, based on the proposed LP formulation and accurate multimedia complexity modeling, we propose an online robust sequential linear programming approach to DVS, namely SLP/r, which outperforms existing DVS solutions. Experimental results from real-time video decoding (where workloads are highly time-varying) indicate that SLP/r consumes less than 0.3% more energy than the optimal DVS solution while dropping only 0.03% of decoding jobs. While the a very recent algorithm (the queuing-based algorithm 2 in [13] by coauthor of this paper) consumes roughly 4% more energy than the optimal at the same miss rate, our online approach has significantly reduced the gap between online algorithms and optimal solution from 4% to 0.3%. Also of note, the SLP/r algorithm has only a small overhead, since the time complexity of SLP/r mainly depends on the efficiency of the LP solver. The relative complexity of SLP/r will scale down when supporting increasingly computational applications (e.g. higher resolution multimedia decoding) in the future.

While we have used video decoding as an example in this paper for motivation and experiment, both the offline LP and online SLP/r approaches are applicable to the DVS problem concerning other delay-sensitive real-time applications with time-varying workloads, such as data mining and stream processing applications.

This paper extends our previous study in [27]. We extend the online algorithm SLP/r to support adjustable granularities of running sequential linear programming. Also, by studying and optimizing over the granularity and conservativeness of SLP/r, we further reduce the energy consumption gap between online algorithms and optimal solution by 3× (from 1% to 0.3%).

The rest of this paper is organized as follows: section II provides background on multimedia complexity and power modeling. Section III formally states the real-time DVS problem. Sections IV and V introduce the optimal offline LP solution and the online SLP/r algorithm, respectively. Section VI presents experimental results to validate our work. Section VII concludes our work.

II. BACKGROUND AND MODELING

A. Multimedia Complexity

State-of-the-art video coders (H.264, SVC etc.) often encode adjacent frames jointly in order to exploit the temporal correlation existing in the video and thereby reduce video transmission bit-rate. However, this leads to complicated group-of-pictures (GOP) structures, where particular video frames require the reconstruction of reference frames in order to be decoded, and other video frames require few or no such reference frame for their decoding. This results in significant workload variations between adjacent decoding jobs (Fig. 1). Moreover, the workload variations will also depend on the different characteristics exhibited by video sequences (e.g. different motion and texture characteristics etc.) [12][13].

In this work, to mitigate the detrimental effects of highly time-varying workloads on DVS algorithms, we adopt the application-aware model for the video coding complexity described in [13] for the proposed online algorithm. In our prior work [13], we showed that complexity statistics of decoding jobs can be decomposed into the sum of complexity metrics that follow simple, well-known distributions, such as Poisson distribution for entropy decoding. Hence, we can approximate each metric by i.i.d. random complexities, which sum up to approximate a Gaussian distribution by the central limit

theorem of probability. Hence, for experiments in our work, we assume that the complexity of jobs follows Gaussian distribution. However, our algorithms are applicable to other media complexity models (e.g., the ones used in [12]) or media compression tasks.

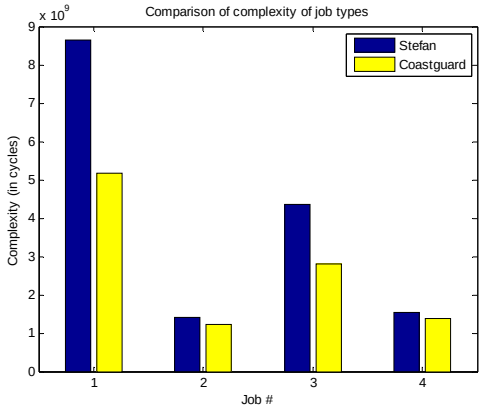


Fig. 1. Comparison of various decoding jobs for video sequences Stefan and Coastguard.

In our work, a job class is defined as a particular frame type in a GOP. For example, we may consider four job classes for a GOP structure in a 3 temporal level MCTF wavelet video coder, where each job involves decoding 2 video frames. Similarly, job classes can be determined for MPEG and H.264 coders based on I, B, and different P-frame types. To model the complexity within each class of jobs, offline training of decoding is used to obtain workload distributions of each job class in different video sequences, as shown in Fig. 2 for the MCTF wavelet coder. These distributions enable us to collect important information about the decoding complexity of each job class, such as the mean and standard deviation. Then, this metadata information can be sent by the encoder/server ahead of jobs with low transmission overhead whenever the sequence characteristics or coder parameters change [25]. Such information can be used by the proposed online DVS algorithm to achieve the tradeoff between energy consumption and quality-of-service.

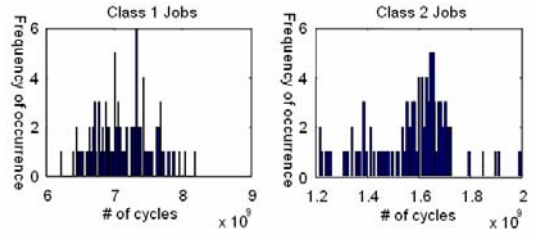


Fig. 2. The workload distribution within one class of decoding jobs.

Finally, note that the complexity of each video decoding job is on the order of a billion cycles. Hence, overheads associated with voltage switches, which are on the order of less than one hundred clock cycles [24], are negligible compared to the processing complexity of multimedia tasks. On the other hand, the number of voltage switches is the number of voltage levels adopted within the job (we can integrate the time allocations of each voltage level into one if more than one time allocation of a voltage level is scheduled). The largest number of voltage switches occurs for the job within which we utilize all different voltage levels. Hence, the number of voltage switches within a job is no more than the total number of voltage levels. Based on these observations, we assume that the voltage switch overhead can be ignored.

B. Dynamic and Leakage Power

In general, a processor consumes both dynamic and leakage power for a given V_{dd} level, and consumes no power when the V_{dd} level is zero, i.e. in the power gating or sleep mode. To evaluate our proposed algorithms, we adopt the power model proposed in [17] and used in [16][21] for real-time applications. However, the algorithms proposed in this paper can apply to any power model, regardless of whether the power-frequency function is convex or not. The dynamic power is:

$$P_d = CV_{dd}^2 F \quad (1)$$

where C is the effective switching capacitance, V_{dd} is the supply voltage and F is the clock frequency. We choose the leakage power model from [17], which includes the subthreshold and the reverse bias leakage power. For a given supply voltage V_{dd}, the leakage power P_s and subthreshold leakage current I_{sub} are:

$$P_s = L_g \left(V_{dd} I_{sub} + |V_{bs}| I_j \right) \quad (2)$$

$$I_{sub} = K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} \quad (3)$$

where L_g is the number of devices in the circuit, I_j is the reverse bias junction current, V_{bs} is the body bias voltage, K_3, K_4 and K_5 are constant fitting parameters. The clock frequency F and threshold voltage V_{th} are:

$$F = (V_{dd} - V_{th})^a / (L_d K) \quad (4)$$

$$V_{th} = V_{th1} - K_1 V_{dd} - K_2 V_{bs} \quad (5)$$

where L_d is the logic depth of the path, *a, K1, K2* and V_{th1} are technology constants. We adopt the constants for 70nm technology node from [16] in our experiment, shown in Table I.

TABLE I
70NM TECHNOLOGY CONSTANTS

Const	Value	Const	Value	Const	Value
K_1	0.063	K	5.26×10^{-12}	C	0.43×10^{-9}
K_2	0.153	V_{th1}	0.244	I_j	4.8×10^{-10}
K_3	5.38×10^{-7}	V_{bs}	-0.7	L_d	37
K_4	1.83	A	1.5	L_g	4×10^6
K_5	4.19				

C. Assumptions and Clarifications

In this paper, the DVS problem we are solving has certain attributes which must be considered: we consider a known workload for the offline problem and an uncertain workload for the online problem; we consider both inter- and intra-job scheduling, where we allow voltage switch to occur within a job as well as between jobs; similar to most DVS-enabled processors, the configurable voltage levels are discrete.

Furthermore, we assume that power is constant if the voltage and frequency level are set; this assumption is also adopted in many existing works [5] - [15]. Also, we assume that compared with multimedia decoding jobs, the voltage switch overhead is small enough to be ignored. For the offline problem, we assume that the complexity and arrival time for each decoding job are

known. This information can be obtained from the trace of the video decoding. For the online problem, we assume that the mean and standard deviation of complexities are obtained by off-line training and are transmitted to the decoder before decoding of these jobs start [13].

III. PROBLEM STATEMENT

For the DVS problem, we are given a sequence of decoding jobs. Each job has a given complexity (workload in unit of clock cycles), arrival time and display deadline. Because we are performing real-time media transmission and decoding, the arrival time can be influenced by the time-varying network characteristics [26]. Also, a voltage/frequency configurable system can switch the frequency of its processor by dynamically adapting its voltage level. Hence, we have a set of active operating levels with frequencies and corresponding powers (sum of leakage power and dynamic power). Furthermore, if power gating is enabled, we have an additional operating level for the sleep mode. The goal of a DVS algorithm is to find a scheduling solution to minimize the total energy consumption. The DVS problem is formalized below:

Problem Formulation 1

Given M decoding jobs with their associated complexities, arrival times and display deadlines, plus K voltage levels with the associated clock frequencies and power, the DVS problem is to find the voltage scheduling solution to minimize the energy consumption for the entire sequence of jobs under the following constraints: the decoder can only start a job after it arrives from the network and the decoder needs to finish each job before its deadline.

To write DVS problem in formulas, let $C = \{C_1,\ldots,C_M\}$, $T = \{T_1,\ldots,T_M\}$, and $D = \{D_1,\ldots,D_M\}$ be the complexity, arrival time and display deadline of each job, respectively. Let $F = \{F_0,\ldots,F_K\}$,and $P = \{P_0,\ldots,P_K\}$ (F_0 and P_0 for sleeping mode) be the associated clock frequencies and powers for each voltage level, respectively. The scheduling solution is $S = \{T_s, V_s, N\}$, where N is the number of voltage switches, $T_s = \{t_0,\ldots, t_N, t_{N+1}, t_0=0\}$ and $V_s = \{v_0,\ldots,v_N\}$ are the time (not including t_{N+1}) and voltage level for each switch; t_{N+1} is the time all jobs are finished.

Then, the DVS problem is:

$$\min \; E = \sum_{i=0}^{N} (t_{i+1} - t_i) P_{vi} \tag{6}$$

Subject to

$$L(t_{n+1}) \le \sum_{j=0}^{n} (F_{vj} \bullet (t_{i+1} - t_i)) \le U(t_{n+1}), \; for \; 0 \le n \le N \tag{7}$$

where equation (6) describes the total energy consumption, equation (7) describes the constraints: the decoder can only start decoding a job after it arrives, and each job should be finished before its display deadline. $U(t)$ and $L(t)$ are the upper and lower bound of cumulative decoding complexity at time t and will be defined precisely later in this section.

When the precise complexity of each job is known, the constraints for the problem are given by deterministic C_i and T_i. When uncertainties exist in the workload and transmission time, C_i and T_i can be viewed as stochastic variables and DVS scheduling algorithms cannot guarantee that all jobs will be decoded before their deadlines. Hence, in the stochastic case, the hard deadline constraint can be replaced with the constraint of keeping the miss rate for jobs within a tolerable range.

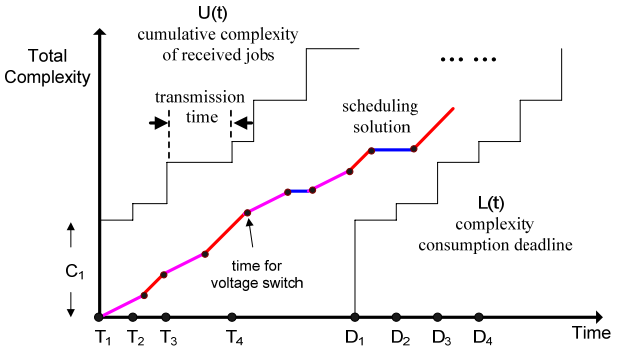


Fig. 3. DVS problem formulation in time-complexity space.

We further illustrate the DVS problem in the time-complexity space, as shown in Fig. 3. Here, the x-axis is time and the y-axis is the cumulative complexity of jobs. T_i indicates arrival time and D_i indicates deadline. C_i is the complexity of each job. The step function $U(t)$ is the cumulative complexity of jobs based on their arrival times. It indicates the maximal computation that can by done by time t. Step function $L(t)$ is the cumulative complexity of jobs based on their deadlines. It indicates the minimal computation that needs to be done by time t. So, $U(t)$ depends on T_i and C_i while $L(t)$ depends on D_i and C_i. $U(t)$ is not simply a shift of $L(t)$ over time, since $U(t)$ captures the transmission time of a job over a network. On the other hand, the display deadlines are deterministic and correspond to the video frame display times. The constraints are given by:

$$U(t) = \sum_{j=1}^{k} (C_j), for \; T_{k-1} < t \le T_k, \; 1 \le k \le M, \; T_0 = 0 \tag{8}$$

$$L(t) = \sum_{j=0}^{k-1} (C_j), for \; D_{k-1} \le t < D_k, \; 1 \le k \le M, C_0 = 0, D_0 = 0 \tag{9}$$

Since the decoder cannot start decoding a job before it is completely received from the network, and it must finish the job before its deadline, a valid DVS solution is a piecewise linear curve between $U(t)$ and $L(t)$. As shown in Fig. 3, the point connecting two segments indicates the time for a voltage switch while the slope of a segment indicates the clock frequency. We call this curve the *cumulative computation curve*, as described in (7).

IV. OPTIMAL OFFLINE SOLUTION

In this section, we show that the deadline-driven multimedia DVS problem can be mapped into a tractable LP problem. If we know the precise complexity and arrival time of each decoding job, we can obtain the optimal scheduling solution.

We define a *transition point* as the time when a new job arrives (i.e. any T_i), or when a job deadline is reached (i.e. any D_i). We also define an *adaptation interval* as the time period between two adjacent transition points. The adaptation intervals for sample $U(t)$ and $L(t)$ curves are marked in dotted lines in Fig. 4. We now prove an important theorem for DVS.

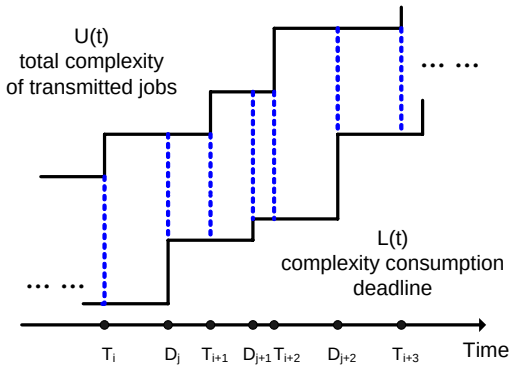


Fig. 4. Adaptation intervals.

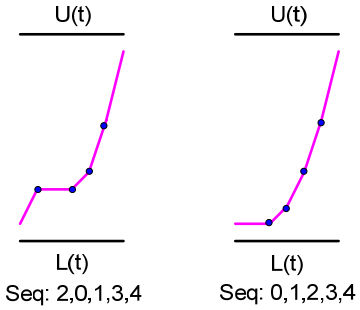


Fig. 5. Different voltage scheduling orderings.

Primary Theorem: Within an adaptation interval where *U(t)* and *L(t)* are constant, a feasible voltage scheduling can be expressed as the time allocation of each voltage level. Another voltage scheduling with the same allocation will have the same cumulative computation and the same amount of energy consumed by the end of the adaptation interval.

Proof: First, if the scheduling has more than one time allocation for a voltage level, we can integrate these allocations into one. The total energy consumption is the sum of each time allocation multiplied by the corresponding power, and the total computation consumption is the sum of each time allocation multiplied by the corresponding frequency. If the time allocation is fixed for all voltage levels, the energy consumption and cumulative computation are both fixed. Second, the cumulative computation curve will lie between *U(t)* and *L(t)*. If *U(t)* and *L(t)* are constant, the order of voltage levels will not affect the performance. Fig. 5 presents an example for two different orders (2,0,1,3,4) and (0,1,2,3,4) (the numbers refer to the slopes) with the same time allocation. ■

The *primary theorem* is the *key idea* to map the DVS problem to a tractable LP problem. Rather than finding the precise times for voltage switches, which would create an intractable integer linear programming (ILP) problem as in [21], we instead solve for the *percentage* of time for each voltage level within an adaptation interval. The LP problem is formulated as following.

Problem Formulation 2

The offline DVS problem is:

$$\min \; E = \sum_{i=1}^{N}\sum_{j=0}^{K}(A_{ij} \bullet P_j \bullet (I_i - I_{i-1})) \qquad (10)$$

Subject to

$$0 \le A_{ij} \le 1, \; for \; 0 \le j \le K \; and \sum_{j=0}^{K} A_{ij} = 1 \qquad (11)$$

$$L(I_n) \le \sum_{i=1}^{n}\sum_{j=0}^{K}(F_j \bullet A_{ij} \bullet (I_i - I_{i-1})) \le U(I_n), \; for \; 1 \le n \le L \qquad (12)$$

Here, we label the transition points as an ordered set $I = \{I_0,\ldots,I_L\}$, where I_0=0 and $I_L = T_{end}$, i.e. we have a total of L adaptation intervals. For these L intervals, we have voltage level allocation vectors given by $A = \{A_1,\ldots,A_L\}$, where $A_i = \{A_{i0},\ldots,A_{iK}\}$ and A_{ij} is the percentage of voltage level j in adaptation interval I_i to I_{i+1}. Then, the unknown is the voltage level allocation vectors given by A. The constraint in (12) is that the valid DVS solution should be between *U(t)* and *L(t)* defined in (6) and (7).

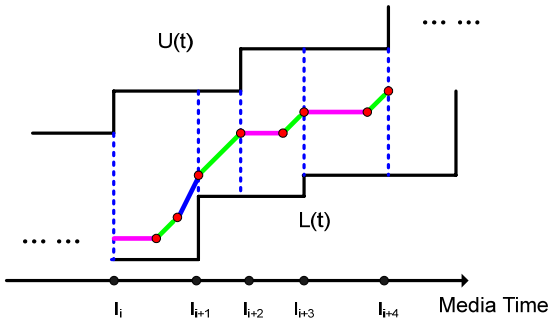


Fig. 6. Scheduling solution.

One can easily prove that the problem defined in (10) to (12) is a linear programming problem [30]. Hence, with this formulation, solving the LP problem leads to the optimal solution for the offline DVS problem. Once the optimal time allocation in each adaptation interval is obtained, we schedule the voltage from lowest to highest. We show an example with 3 voltage levels (including power gating) in Fig. 6. For the first adaptation interval, voltage level 0, 1, 2 occupy 50%, 25%, 25% of time respectively, for the second, the third and fourth intervals, the time allocation is (0%, 100%, 0%), (66%, 34%, 0%) and (75%, 25%, 0%). As shown in the figure, we start from the lowest voltage level with non-zero time allocation and we skip the unused voltage levels.

Note that this formulation is *pervasive*: the operating voltages can be of any discrete values, and there is no requirement for the power-frequency model. Furthermore, this formulation is also applicable to other delay-sensitive DVS problems for real-time applications.

The offline approach can be used to determine the operational lower bound for energy consumption, as well as whether the utilized online DVS algorithm operates close to the optimal scheme. In the next section, we will discuss an online adaptation of the proposed algorithm.

V. EFFECTIVE ONLINE ALGORITHM

For online multimedia applications, where jobs are received over a network, we often do not know the precise complexity and arrival times of each decoding job. Nevertheless, the idea of mapping DVS into a linear programming problem in section IV can still be used for online DVS. We solve the stochastic online DVS problem by sequentially solving a robust linear

program (rLP). We label our algorithm SLP/r.

There are three stages in each round of SLP/r: *prediction*, *solving rLP* and *commitment*. For prediction, we predict the stochastic complexity of decoding jobs in a future time window by using the linear combination of the mean and standard deviation of jobs. As discussed in section II.A, this information can be transmitted to the decoder before decoding start. Then we solve an rLP problem to obtain the scheduling solution for the predicted decoding jobs in the window. Finally, we commit one or more jobs based on the scheduling solution obtained from solving rLP. The committed number of jobs is defined as the *granularity* of rLP. It is smaller than the number of jobs predicted in prediction stages. After commitment, we move the window forward, predict the complexity in the new window, and repeat the rLP based on new statistics.

A. Consideration of Stochastic Complexity

The prediction of future decoding job complexities in the sliding window is crucial to our online solution. Using only the mean of each job class for prediction may lead to a high miss rate. To reduce the probability of misses, we incorporate the standard deviation of each job class with the mean to estimate the bounded "worst case" complexity in a probabilistic manner. In SLP/r, we adopt the linear combination of the mean and standard deviation for each job class to explicitly adjust *U(t)* and *L(t)*, and hence, to determine the miss rate probability. The adjustments are based on a conservativeness α. Note that for jobs far into the future of a prediction window, the cumulative standard deviation over jobs may be large. Therefore, a scaled coefficient α (possibly 0, such that only the mean is considered) can be used to guarantee feasibility of rLP. This does not necessarily increase the miss rate, because we only commit the imminent jobs and not all predicted jobs in commit stage.

Problem Formulation 3:

The rLP problem for a given prediction window is:

$$\min \ E = \sum_{i=1}^{W}\sum_{j=0}^{K}(A_{ij} \bullet P_j \bullet \varphi) \tag{13}$$

Subject to

$$0 \le A_{ij} \le 1, \ for \ 0 \le j \le K \ \ and \ \sum_{j=0}^{K} A_{ij} = 1 \tag{14}$$

$$L(I_n) \le \sum_{i=1}^{n}\sum_{j=0}^{K}(F_j \bullet A_{ij} \bullet \varphi) \le U(I_n), \ for \ 1 \le n \le W \tag{15}$$

where φ the display interval, and W is the prediction window size. Adaptation intervals *I*, *U(t)* and *L(t)* are defined as the following:

$$I = \{I_0,...,I_W\}, I_i = i \bullet \varphi \tag{16}$$

$$U(t) = \sum_{j=1}^{k}(\widetilde{C}_{W_0+j}), \ for \ I_{k-1} < t \le I_k, \ 1 \le k \le W \tag{17}$$

$$L(t) = \max(0, U(t - \theta \bullet \varphi)) \tag{18}$$

where W_0 is the current adaptation interval and $\widetilde{C}_i$ is the predicted stochastic complexity of job *i*. (17) and (18) show that *U(t)* and *L(t)* are the upper and lower bounds of the cumulative predicted complexity. Also, since we assume each job is released θ display intervals before the display deadline, *U(t)* is simply a time-shifted version of *L(t)*. Please refer to the detailed description in section V.B and V.C. Specifically, we have:

$$\widetilde{C}_{W_0+j} \le \rho_{W_0+j} + \alpha_j \bullet \sqrt{v_{W_0+j}} \tag{19}$$

$$\alpha_j = \max(0, \alpha \bullet (R - j + 1)/R), \ for \ 1 \le j \le W \tag{20}$$

where ρ_i and v_i are the mean and variance of stochastic complexity of job *i*, α is the conservativeness and R is a constant. Equation (20) indicates that the coefficient of standard deviation decreases between α and 0 over time. Note that a tradeoff between miss rate and energy consumption can be achieved by tuning α. For example, increasing α will make the bounds tighter and leads to a lower miss rate at the cost of higher average energy consumption.

One can show that the problem defined by equations (13) to (20) is an rLP problem [31]. Once we get the schedule solution, we schedule the voltage in the order from lowest to highest voltage level, identical to the offline problem. Note that with stochastic complexity model, the proposed online algorithm applies to other real-time applications although we only use video decoding as an example. After committing one or more jobs, we need to adjust *U(t)* and *L(t)* dynamically. The idea will be discussed and demonstrated in section V.C.

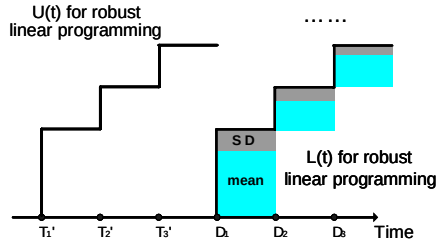
B. Extension to Unreliable Network

For SLP/r, another challenge is that we need to cope with the time-varying network characteristics, since we do not know the exact arrival time of a job. We assume that a network buffer at the decoding side collects packets and dispatches jobs to the decoder according to the display frame rate. Then, we predict the time when each job is ready to be decoded is θ display intervals before the deadline. This indicates that the adaptation intervals are divided by the display deadlines of each job, and the number of adaptation intervals is $M + \theta$, where M is the number of jobs. In this fashion, we can reduce the number of adaptation intervals from $2M$ to $M + \theta$ (hence the size of the rLP problem). In this case, the adaptation intervals *I*, *U(t)* and *L(t)* are defined as (16) to (18). If a job arrives before the scheduling time (i.e. the real *U(t)* is higher than the complexity consumption line), we determine the voltages as guided by rLP. If a job arrives late due to insufficient network bandwidth, power gating can be used to shut down the processor until this new job arrives, based on which *U(t)* and *L(t)* are adjusted for the next rLP.

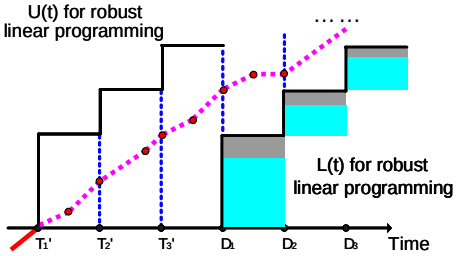
C. Illustration of SLP/r

We further illustrate SLP/r in the time-complexity space, as shown in Fig. 7. Fig. 7(a) shows the prediction stage. We predict the complexity of each job using the linear combination of mean and standard deviation (gray area). We predict the arrival time *U(t)* is ahead of *L(t)* by θ display intervals; then, *U(t)* is only a shift of *L(t)*. Please note that while we show a prediction of 3 jobs here, in our implementation we often predict 8 or 16 jobs. We then solve an rLP for jobs in the window, as shown in Fig. 7(b); the dotted line perpendicular to the x-axis indicates the adaptation intervals and the dotted piecewise linear curve indicates the scheduling solution from solving rLP. The solid curve in the bottom indicates the existing

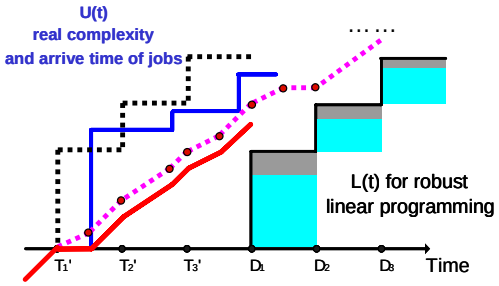
cumulative computation curve from the previous round.



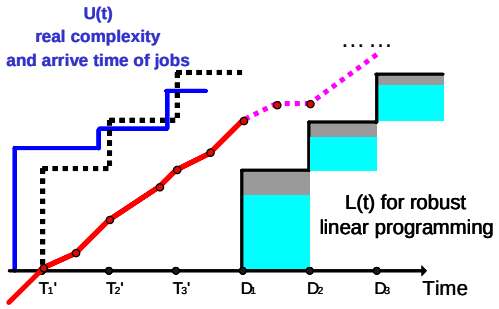
(a) Prediction



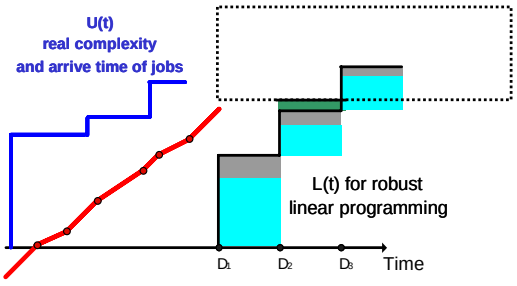
(b) Solving rLP



(c) Commitment, jobs arrive late



(d) Commitment, jobs arrive early



(e) Move the future window forward

Fig. 7. Detailed illustration of SLP/r.

The strategies for dealing with unreliable networks are shown in Fig. 7(c) and 7(d). Fig. 7(c) highlights the case when a job arrives late, while Fig. 7(d) highlights the case when a job arrives early. Here, the dotted step curve indicates *U(t)* for robust linear programming while the solid step curve indicates real *U(t)* (the same applies for Fig. 7(d)). In Fig. 7(c), the solid piecewise linear curve illustrates that we power gate over the delayed time period, and then commit a given number of jobs (the given number is the granularity of SLP/r). Because the unit of commitment is an adaptation interval, the granularity of rLP defines a lower bound on the number of jobs to be committed. If the decoder finishes decoding and has extra computation to be done in the last adaptation interval, we begin decoding the next job (and possibly more jobs if these jobs have arrived, and extra resources are available). As shown in Fig. 7(c), we also commit part of the second job because extra computation is done within the third adaptation interval. Fig. 7(d) indicates the case when jobs arrive earlier. In this case, we commit two jobs plus part of the third job. This is because the first job cannot be finished within the first two adaptation intervals, and in the third adaptation interval the second job and part of the third job are finished. Note that though the granularity set for this example is one job, it's possible to commit more jobs in each round of rLP, two and part of the third shown in this case. After commitment, we need to adjust the prediction for the third job in the next run of rLP, since part of the third job has been completed. As shown in Fig. 7(e), we reduce the predicted complexity of the third job as part of it has been finished. Also, we move the future window forward to start the next round, as indicated by the dotted rectangle. Then, we repeat this process until all jobs are finished.

VI. SIMULATIONS AND RESULTS

A. Experimental Setup

In our experiments, we adopted the power and frequency model for the 70nm technology node in [16][17]. We considered discrete voltage levels V_{dd} between 0.6V and 1.0V with voltage step sizes of 0.1V. The clock frequencies and power for different V_{dd} levels are presented in Table II.

We combined 10 video sequences with different characteristics into a long sequence, which was then decoded using a 4 temporal level MCTF coder[1]. We measured the complexity of each decoding job in terms of clock cycles of real computers and used the measurement for offline scheduling. We pre-trained the stochastic model using the measurement for the proposed online algorithm SLP/r as in [13].

TABLE II
FREQUENCY AND POWER FOR DIFFERENT VDD LEVELS

Vdd (V)	0.6	0.7	0.8	0.9	1.0
Frequency (GHz)	0.79	1.27	1.81	2.42	3.09
Dynamic Power (10-5W)	0.12	0.27	0.50	0.84	1.33
Leakage Power (10-5W)	0.21	0.29	0.40	0.54	0.72
Total Power (10-5W)	0.33	0.56	0.90	1.38	2.05

To simulate a real-time video decoding environment with sequences that have a frame rate of 30Hz, we fixed display

[1] We chose the MCTF coder since the workload variations are highly noticeable for the different sequences. Note that using a different coder would only lead to a different complexity trace for the decoding jobs, but would not affect the optimality of our offline algorithm.

deadlines for the application. We assumed that the frame arrivals from the network following the normal distribution as discussed in [25] to simulate a wireless network, and we applied the same generated arrival times of jobs for all algorithms in our experiments. For all algorithms, we calculated the energy using the same power model considering the leakage power. Since the actual value of energy is not important for comparison between the three methods, we report the normalized energy, given by the energy consumption ratio of online schemes to the optimal solution.

Furthermore, due to the stochastic nature of complexities and transmission delays, we present results based on a Monte Carlo simulation, where the Gaussian distribution of decoding complexities is from the trace of a real decoding system [13]. We also modeled the transmission delay using a normal distribution [25].

Two parameters need to be set by the user in SLP/r. The first one is the conservativeness (α in *Problem Formulation 3*) which decides the trade-off between miss rate and energy. The second one is the granularity of SLP/r. It is the number of jobs to commit before shifting the future time window. It decides the tradeoff between runtime and quality of solution. Intuitively, a large conservativeness and a small granularity may lead to higher energy consumption, while a low conservativeness and a large granularity may lead to a high miss rate. Our experiment in the next sub-section will study different combinations of conservativeness and granularity to verify whether the above intuition is correct.

B. Optimality Study

In our experiment, we extended laEDF [5] and the queuing based algorithms [13] to use the leakage-aware power model. Also we extended these algorithms to consider sleep mode for a fair comparison. For queuing based algorithms 1 and 2 in [13], we selected algorithm 2 for comparison as it outperforms algorithm 1 experimentally. We tuned the parameters to obtain different trade off points for energy and miss rate. For the queuing based algorithm, we tuned the delay sensitivity parameter ε, and for laEDF, we used different WCETs.

The results are presented in Fig. 8. The energy achieved by the optimal offline LP solution (e.g. the lower bound) is normalized to 1. Note that based on our formulation, the optimal solution always has zero miss rate. The result shows that for a zero miss rate, laEDF consumes approximately 15% more than the optimal and queuing based algorithm 2 consumes approximately 4% more than the optimal.

We also compared SLP/r with the optimal solution and existing algorithms. For this experiment, we set granularity as 1 job, and we tuned the conservativeness α to obtain different trade off points for energy and miss rate. The sliding window size of SLP/r is set to 16 jobs (2 GOPs). From Fig. 8, one can see that SLP/r has only about 0.6% more energy consumption than the optimal solution while keeping the miss rate below 0.1%. The queuing-based algorithm 2 consumes roughly 3.5% more energy than SLP/r under the same miss rate (0.1%), while laEDF consumes approximately 13% more than SLP/r. Though existing work in [13] is very close to optimal, SLP/r further explores the potential of online DVS algorithms and significantly reduces the gap between online algorithms and optimal solution. Also, note that the comparison is based on the result from SLP/r with granularity of 1 job. However, we can achieve an even better solution by changing other parameter settings, shown in the following section.

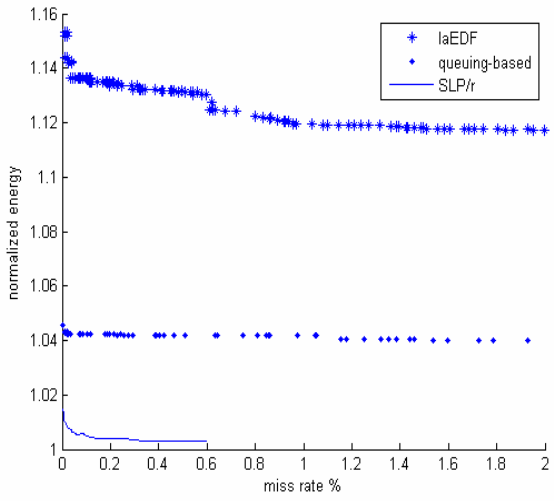


Fig. 8. Energy and miss rate.

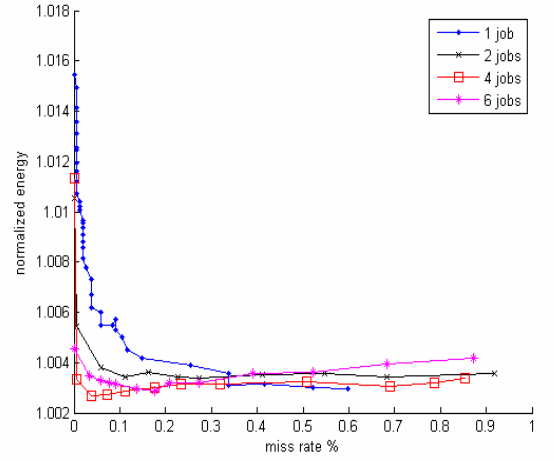


Fig. 9. Granularity versus solution quality.

C. Optimizing SLP/r

To study the impact of granularity on the decoding quality of the solution, we ran simulations for granularities from 1 job to 8 jobs and compare the lowest energy points. In Fig. 9, the simulation results for granularities 1, 2, 4 and 6 jobs are plotted. We found that for a granularity of 4 jobs, we achieved 0.03% miss rate with 0.3% more energy compared to the optimal offline solution, which outperforms all other granularities. Also, the increase of normalized energy with an increasing miss rate for large granularities is an interesting phenomenon. This is because, for large granularities, when the conservativeness is low, the predicted complexity bounds may be looser than the actual bounds, especially for jobs far in the window. The scheduling solution from the loose bounds will adopt lower voltage level than needed. Hence, when jobs are committed,

computation complete before deadline may be less than needed, thus cause a missed job. Meanwhile, computation that needs to be complete will be more for the next immediate job in next round of SLP/r. In this way, the voltage levels adopt will be higher for the next immediate job in the window and lower for the jobs far in the window. Hence, the overall energy consumption will be higher. For small granularities such as 1 job, the adjustment is faster. Hence, the energy consumption will not be higher.

To further study the impact of parameter settings, we applied different combinations of conservativeness (from 0 to 4) and granularities (from 1 job to 8 jobs). The corresponding results for energy and miss rate are presented in Fig. 10 and Fig. 11 respectively.

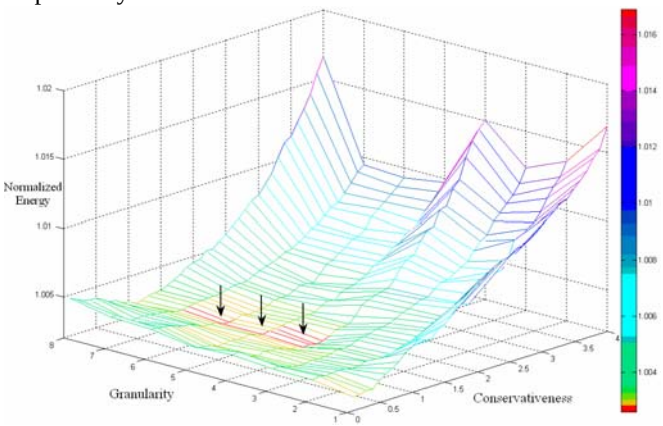


Fig. 10. Energy versus granularity/conservativeness.

The impact of parameters on energy is shown Fig. 10. One can see that, for a fixed granularity, larger conservativeness usually leads to higher energy consumption. Also, for conservativeness less than 1, energy consumption increases while conservativeness decreases. This trend is more distinct for larger granularities. The interpretation is that a large conservativeness leads to a larger prediction of job complexity in the window. Thus, the corresponding schedule solution tends to adopt a higher voltage level, which leads to higher energy consumption. A very small conservativeness on the other hand leads to a less than needed computation done. Hence, if the next job carries a large workload, the processor needs to operate at a high voltage level to compensate for lost time. For larger granularity, this phenomenon is more significant because the feedback and adjustment are slower. Another interesting phenomenon is that energy vibration appears in the large conservativeness region. For a large conservativeness, granularities 4 and 8 jobs consume less energy than others. This is because of the specific GOP structure adopted in our experiment. Granularities of 4 and 8 jobs always have jobs that contain I frames (large workload) as the immediate next job in the future time window. Due to the large α of the immediate next job (see equation (19) and (20) for details), the prediction will be very conservative. Hence, the prediction will result in higher energy consumption and lower miss rate. This phenomenon is more distinct for conservativeness 4 due to the higher energy consumption which results from a large conservativeness.

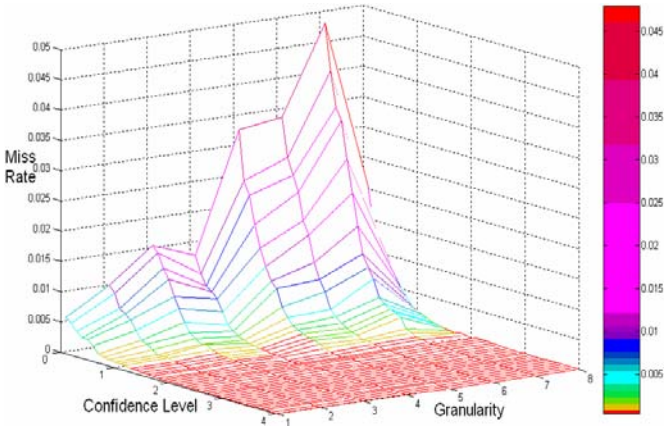


Fig. 11. Miss rate versus granularity/conservativeness.

The impact of parameters on miss rate is shown Fig. 11. We find that for conservativeness larger than 2, most granularities lead to a zero miss rate. When the conservativeness is small, granularities of 4 and 8 jobs have a lower miss rate. This phenomenon is again the result of the GOP structure used in our experiment.

To indentify the default parameters of SLP/r, we find from Fig. 10 that for granularity of 4 - 6 jobs and conservativeness 1.5, we can get the minimal energy consumption (marked by arrows). In Fig. 11, among these parameter settings, a granularity of 4 jobs and conservativeness 1.5 has a miss rate very close to zero. Therefore, for the decoder used, we determined that the combination of a 4 job granularity and conservativeness 1.5 is the approximate optimal parameter setting, and can be used as default parameters. The analysis is as following: for a small granularity, increasing the conservativeness will lead to lower miss rate but it will be too aggressive using a large conservativeness for each of them. Hence, a larger granularity will balance the conservativeness and miss rate better. However, too large of a granularity will lead to inaccurate predictions and lagged adjustments. Hence, there exists an approximate optimal combination of granularity and miss rate: 4 jobs for granularity and 1.5 for conservativeness, as shown from our experiment. It is important to note that the energy and miss rate do not change dramatically around the aforementioned setting. Therefore, it is a robust setting. This setting can be used in practice because we have considered decoding of different video types in our experiment.

D. Runtime

For a granularity of 4 jobs and conservativeness of 1.5, the total runtime of SLP/r for the combined 512s long video sequence is 18s, which indicates that the runtime overhead of the online scheduling algorithm is approximately 3.5% of the video decoding workload, which is acceptable. While the runtime existing laEDF and queuing base algorithms are less than 0.1%, we expect the relative runtime overhead of SLP/r to decrease in the future with more careful implementation. The associated energy overhead of scheduling will also decrease relatively to the more computationally intensive applications such as higher resolution video decoding.

VII. CONCLUSIONS

In this paper, we have analyzed the optimality of online DVS algorithms by formulating the optimal off-line DVS as a linear program. We show that at a zero miss rate, existing works consume 4% more energy than the optimal solution. We have also developed an effective online DVS algorithm using robust sequential linear programming, which significantly outperforms existing online DVS solutions and is merely 0.3% away from the optimal. Though existing work is close to optimal, we further reduce the gap between online algorithms and optimal solution from 4% to 0.3%.

To further improve the performance of these DVS solutions, we plan to develop solutions which can more precisely predict complexity of future jobs by exploiting the video sequence characteristics and the corresponding coding parameters used by state-of-the-art multimedia coding algorithms. In this way, we can reduce the runtime overhead of SLP/r by reducing the frequency of solving the rLP problem. Also, we plan to build a lookup table for scheduling solutions based on offline training to further reduce the runtime. Finally, we will apply our proposed formulation and algorithms to other real-time delay-sensitive applications with time-varying workloads.

REFERENCES

[1] L. Benini, and G. De Micheli. Dynamic power management: design techniques and CAD tools. Kluwer Academic Publishers, Norwell, MA, 1997.

[2] D. Marculescu. On the use of microarchitecture-driven dynamic voltage scaling. *Proceedings of the Workshop on Complexity-Effective Design*, 2000.

[3] J. Lorch, and A. Smith. PACE: a new approach to dynamic voltage scaling. *IEEE Trans. on Computers,* vol. 53, no. 7, pp. 856-869, Jul. 2004.

[4] T. Ishihara, and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *Proceedings of International Symposium on Low-Power Electronics and Design. Monterey*, 1998.

[5] P. Pillai, and K. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. *Proceedings of the 18th ACM symposium on Operating Systems*, 2001.

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

[7] W. Yuan, and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. *Proceedings of the 19th ACM Symposium on Operating System Principles*, 2003.

[8] Y. Zhu, and F. Mueller. Feedback EDF scheduling exploiting dynamic voltage scaling. *Proceedings of the 11th international conference on Computer Architecture*, 2004.

[9] K. Choi, K. Dantu, W. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a MPEG decoder. *Proceedings of ICCAD*, 2002.

[10] Y. Zhu, and F. Mueller. DVSleak: combining leakage reduction and voltage scaling in feedback EDF scheduling. *Proceedings of LCTES*, 2007.

[11] A. Maxiaguine, S. Chakraborty, and L. Thiele. DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs. *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2005.

[12] E. Akyol, and M. van der Schaar. Complexity model based proactive dynamic voltage scaling for video decoding systems. *IEEE Trans. Multimedia*, vol. 9, no. 7, pp. 1475-1492, Nov. 2007.

[13] B. Foo, and M. van der Schaar. A queuing theoretic approach to processor power adaptation for video decoding systems. *IEEE Trans. Signal Process*, vol. 56, no. 1, pp. 378-392, Jan. 2008.

[14] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Trans. Comput.*, vol. 53, no. 5, pp. 584-600, May 2004.

[15] C. Xian, Y.-H. Lu, and Z. Li. Dynamic voltage scaling for multitasking real-time systems with uncertain execution time. *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1467-1478, Aug. 2008.

[16] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. *Proceedings of DAC*, 2004.

[17] S. Martin, K. Flautner, T. Mudge, and D. Blaauw. Combined dynamic voltage scaling and adaptive body biasing for low power microprocessors under dynamic workloads. *Proceedings of ICCAD*, 2002.

[18] C. Kim, and K. Roy. Dynamic VTH Scaling Scheme for Active Leakage Power Reduction. *Proceedings of Design, Automation, and Test in Europe*, 2002.

[19] L. Yan, J. Luo, and N. K. Jha. Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems. *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 7, pp. 1030-1041, July 2005.

[20] S. Hong, S. Yoo, B. Bin, K-M. Choi, S-K. Eo, and T. Kim. Dynamic voltage scaling of supply and body bias exploiting software runtime distribution. Proceedings of Design, Automation, and Test in Europe, 2008.

[21] S. Zhang, and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. *Proceedings of ICCAD*, 2007.

[22] R. Jayaseelan, and T. Mitra. Temperature aware task sequencing and voltage scaling. *Proceedings of ICCAD*, 2008.

[23] S. Zhang, and K. Chatha. System-level thermal aware design of applications with uncertain execution time. *Proceedings of ICCAD*, 2008.

[24] J. Dunning, G. Garcia, J. Lundberg, and E. Nuckolls. An all-digital phase-locked loop with 50-cycle lock time suitable for high-performance microprocessors. *IEEE Journal of Solid-State Circuits*, vol. 30, no. 4, pp. 412 – 422, Apr. 1995.

[25] A. Adas. Traffic Models in Broadband Networks. *IEEE Communications Magazine*, vol. 35, no. 7, pp. 82-89, July 1997.

[26] 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, June 2005.

[27] Z. Cao, B. Foo, L. He, and M. van der Schaar. Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications. *Proceedings of DAC*, 2008.

[28] J. Pouwelse, K. Langendoen, and H. Sips. Application-directed voltage scaling. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 5, pp. 812-826, Oct. 2003.

[29] D. Biermann, E.G. Sirer, and R. Manohar. A rate matching-based approach to dynamic voltage scaling. *Proceedings of the First Watson Conference on the Interaction between Architecture, Circuits, and Compilers*, October 2004.

[30] A. Schrijver. Theory of linear and integer programming. John Wiley and Sons, 1986.

[31] S. Boyd, and L. Vandenberghe. Convex optimization. Cambridge University Press, 2003.

[32] Y. Cho, and N. Chang. Energy-Aware Clock-Frequency Assignment in Microprocessors and Memory Devices for Dynamic Voltage Scaling. *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 6, pp. 1030-1040, June. 2007.

[33] D. Ma. Automatic Substrate Switching Circuit for On-Chip Adaptive Power-Supply System. *IEEE Trans. On Circuits and Systems II*, vol. 54, no. 7, pp. 641-645, July 2007.

[34] X. Zhong, and C. Xu. System-wide energy minimization for real-time tasks: lower bound and approximation. *Proceedings of ICCAD*, 2006