

Stochastic Modeling and Optimization for Energy Management in Multi-Core Systems: A Video Decoding Case Study

Soner Yaldiz, *Student Member, IEEE*, Alper Demir, *Member, IEEE*, and Serdar Tasiran, *Member, IEEE*

Abstract— We address the problem of energy minimization in multi-core systems running soft real-time applications with tolerance to deadline misses. We consider applications that have been decomposed into a set of concurrent and interdependent tasks. We present a novel stochastic modeling and optimization framework which leads to effective energy management policies. At the heart of this framework lie stochastic application models which capture the variability of and the spatial and temporal correlations among the workloads of tasks. We use these stochastic models in novel mathematical formulations to obtain optimal energy management policies. These policies minimize the average energy consumption using dynamic voltage scaling, provide a probabilistic guarantee for satisfying the timing constraints and introduce negligible overhead at runtime. Experimental results on MPEG2 video decoding show that our policies achieve significant energy savings, often close to the theoretical upper bound.

Index Terms— Multi-core processors, real-time systems, energy management, dynamic voltage scaling, stochastic programming

I. INTRODUCTION

An energy management technique commonly used in modern systems is dynamic voltage scaling (DVS). DVS enables processors to slow down and save energy when the computational demand is low. Using this capability for real-time applications poses several challenges. Real-time applications are subject to timing constraints called “soft” or “hard” depending on whether or not the application can tolerate the occasional violation of the constraints. Many real-time applications from various domains have a periodic nature and process a varying amount of input data at regular time intervals called *periods*. The computational demand in each period depends on the processed input data, varies over time and is unknown a priori. This *workload variability* makes the use of dynamic techniques difficult, because one does not know ahead of time how much to slow down the hardware component without violating the timing constraints. Our work focuses on soft real-time applications, for which significant energy savings are possible because deadline misses can be tolerated for a fraction of the high workload cases. The framework we develop in this paper centers on the formal, stochastic characterization of workload variability, and using these characterizations to make optimal use of DVS for energy management on multi-core processors. In Section II, we present an overview of issues in energy management and relevant previous work most closely related to the techniques described in this paper.

We consider applications that have been decomposed into a set of concurrent tasks in order to take advantage of the multi-core platform. Tasks in such applications can be executed concurrently on different processors running at different

voltages – a key flexibility that makes multi-core processors attractive energy-efficient platforms for computationally demanding multimedia applications. However, exploiting this flexibility in an optimal way is significantly more challenging than modeling and exploiting workload variability for an application running on a single processor. Because of data dependencies and other precedence relationships among tasks, and timing constraints posed on the entire application, one can neither simply add up task workloads nor make completely independent DVS decisions for different tasks. The formal model for a concurrent application running on a DVS-capable multi-core platform must incorporate the platform’s DVS characteristics and a *joint* characterization of workload variability for interdependent tasks. This adds several more dimensions to the energy optimization problem for multi-core processors. We discuss the details of the software model and the DVS-capable hardware platform in Section III.

Workloads of tasks in decomposed multi-media applications exhibit significant correlations. Some aspects of these correlations had been observed in the literature while others are among the novel contributions of our paper. We classify workload correlations in a real-time application into two: *temporal* and *spatial* correlations. By temporal correlation, we refer to the correlation among the workloads of tasks executed in different periods. This phenomenon has been studied in the literature. By spatial correlation, we refer to the correlation among the workloads of concurrent tasks *within the same period*. Our experience with MPEG2 decoding [1] indicates that there can be significant spatial correlations. Spatial correlations had not been investigated previously in the context of energy management. Spatial correlations arise and are particularly important for multi-core systems as they require that coordinated DVS decisions be made for the correlated tasks within a period. The timing constraints are typically given for the entire set of tasks, while the workload dependencies as well as the precedence relationships among tasks need to be taken into account in order to make accurate DVS decisions. The observation that there are significant such correlations in multimedia applications, the development of stochastic models of varying sophistication and power for capturing these correlations, and the use of these models for optimal energy management on multi-core processors are some of the key contributions of our work. In Section IV, we present motivating examples highlighting the importance of spatial and temporal correlations for energy management.

Inspired by our experience with MPEG2 video decoding, we propose four stochastic application models for capturing

the statistical characteristics of concurrent task workloads. The first model captures both workload variability and spatial correlations using general joint probability density functions (PDF). The other three models employ *states* to capture application-specific features and temporal correlations in addition to variability and spatial correlations. These models treat the periodic execution of an application as a discrete-time stochastic process and are derived using application-specific features. For MPEG2 decoding, these features include frame type, compressed frame data size, and the computational workload decoding a frame. The parameters for the stateful models such as state transition probabilities and joint task workload PDF's are obtained from a statistical characterization of the task workloads using long representative runs of the application. We observe that for MPEG2 decoding run on different inputs, certain probabilistic quantities related to the probability mass at the tails of PDF's vary relatively little over time spans relevant for energy management. In Section V, we describe our stochastic models and their use for energy management. We illustrate the construction of stochastic models on the MPEG2 video decoding application in Section VI.

For each stochastic model we propose, we develop a novel optimization formulation which results in an optimal energy management scheme, described in Section VII. Each scheme minimizes the average energy consumption of the system using DVS and satisfies the timing constraints according to the application's tolerance to deadline misses. We also show that energy management problems can be formulated as activity network problems which have been studied in the literature [2]. For solving the formulated optimization problems, we present in Section VIII two efficient solution techniques, one of which employs a heuristic approach previously proposed for activity network problems. Since the optimization problems presented in this paper are solved off-line and their results are stored in a look-up table, our energy management schemes introduce negligible overhead at runtime. In Section IX, we present the experimental setup and our results on MPEG2 decoding. Our energy management schemes achieve energy savings higher than previous approaches and come close to the (unrealizable) theoretical optimum.

II. RELATED WORK

The use of DVS for minimizing energy consumption, and stochastic modeling of applications in general and MPEG2 decoding in particular have been investigated intensively in the literature. Many of the individual aspects and components of the integrated framework we developed appear in different contexts in previous work. We structure the discussion of related work and the comparison with our approach around the following issues:

- (i) whether single-core or multi-core platforms are targeted,
- (ii) whether hard or soft real-time applications are considered,
- (iii) whether the energy management policy is computed offline, online at runtime, or a combination of both,
- (iv) the stochastic models and optimizations methods used, whether they are exact or heuristic, and
- (v) whether the stochastic workload models used take into account task correlations and variations over time.

A. Issues

Single vs. Multi-core. Because workloads of tasks serially executed on a single processor can simply be added up, a group of tasks on a single-core system can be treated as a single task. In this case, optimal energy consumption can be accomplished by selecting a single voltage used for all tasks throughout the period. This simplifies the energy management problem considerably since the total workload of all tasks in a period can be modeled by a one-dimensional PDF. Some previous work considers particular simple processor interconnection topologies such as a few cores organized as a pipeline or two cores with a buffer in between (e.g. [3], [4]). The formalization and solution of the energy management or other resource optimization problems are significantly simplified for such restricted topologies. However, the solutions found are specific to a hardware configuration, and the full power-performance flexibility of a general multi-processor platform is not exploited. Finally, relatively little work focuses on general-purpose multiprocessor platforms.

Hard vs. Soft Real-Time Applications. For soft real-time applications, the statistical distribution of workload, especially the part of the tail of the distribution for which the energy management policy will choose to miss the deadline becomes the most important factor.

Off-line vs. online approaches. A DVS policy can be computed off-line, before the application runs on the platform. Alternatively, it can be computed as the application is running, typically by performing observations about the execution and applying an adaptive algorithm to adjust voltages. Several approaches follow a hybrid strategy. We follow off-line and hybrid approaches in our work.

Stochastic models and optimization methods. The idea behind DVS-based energy management for soft real-time applications is to characterize the workload and predict and decide for which one of the most computationally-demanding periods to not meet the timing constraints. The stochastic modeling, the selection of the cases in which deadlines will be missed, and the voltage settings for implementing these choices can each be carried out formally or in an ad-hoc/heuristic way. Formal stochastic models range from single-dimensional PDF's that disregard task correlations to ones that model joint distributions, to models that use states or history to encapsulate variation of task workloads over time.

Workload variation over time and temporal correlations. The variation of workload over time (in different periods) is another dimension in the stochastic modeling issue. Some approaches explicitly model this variation while others use aggregate statistics averaged over time. We first present a model of the latter kind (a "stateless" model) and then refine it to several "stateful" models that capture task workload variations and correlations over time. Similarly, some models capture spatial correlations while others ignore it. All of our stochastic models explicitly incorporate spatial correlations.

B. Previous Work

The following recent studies focus on hard real-time applications running on single processors, yet, they are able to

save energy by exploiting slacks at runtime when worst-case loads are not encountered. Gruian [5] considers a set of independent tasks running on a single core system where task voltage decisions are made heuristically. Leung and Hu [6] allocate the available execution time to tasks by formulating and solving an energy optimization problem which still guarantees the worst-case scenario. Andrei et al. [7] compute a set of start time and voltage setting pairs for each task using the best, average and the worst-case workloads. At runtime, their algorithm selects one of the precomputed voltage settings depending on the time within the period the task actually starts at. Scordino and Bini [8] present a DVS scheme for minimizing the expected energy consumption for a single task running on a single processor. Xu et al. [9] present a hybrid, offline-online approach for the voltage scaling of a set of tasks with independent PDF's. Liu et al. [4] use a mathematical model that removes from consideration the ϵ percent probability mass at the tail of a one-dimensional PDF. This allows them to work with worst-case assumptions for the remaining probability mass and to provide guarantees in all but ϵ percent of the cases. Hong et al. [10] present an offline approach in which the execution time is allocated to a set of fine-grain tasks. They capture the workload variability using PDF's while ignoring the correlations among tasks.

Since multi-core platforms are relatively new, there are relatively few applications decomposed and optimized for such platforms, and there is much less work on energy management for them. Zhang et al. [11] present a mathematical formulation which leads to an optimal energy management scheme for hard real-time applications. In this formulation, the worst-case task workloads are used to ensure that the timing constraints are always satisfied. They formulate and solve an optimization problem in which the spatial correlations among task workloads are ignored and worst-case workload is assumed for each task. Hua et al. [12] present a heuristic hybrid approach for soft real-time applications running on multicore systems. Their policies include runtime voltage adjustments to make use of slacks caused by tasks finishing earlier than expected.

A line of research on energy minimization uses workload estimators at runtime to predict the workload at the beginning of each period. These estimators are derived empirically using either application-specific features or past workload history. In some work, estimated workload for the current period is expressed as a function of easily-observable, application-specific parameters [13], [14]. In others, workload for the current period is predicted based on workload history [13], [15]–[17]. The key idea behind this category of approaches is the conjecture that the workload in the current period is likely to be similar to the workload in recent periods. Both categories of techniques are vulnerable to some extent to sudden fluctuations in workload [18]. Some studies (e.g. [17] for a single processor) adaptively correct for the prediction error using a feedback mechanism. Depending on the complexity of the workload estimator and the frequency with which it is updated, computing the estimator may require high runtime overhead. Some approaches capture offline the statistical behavior of workload using probabilistic models [4]–[9], [19], [20]. More precisely, they construct probability distributions for or oth-

erwise statistically characterize particular applications. These approaches to energy minimization are able to provide tighter performance bounds and can be shown to be optimal if the underlying workload models are accurate.

Stateful models have been previously proposed for energy management in the literature [20]–[23]. In most previous work, a queuing network formalism is used where the emphasis is on streaming applications and the arrival rate of tasks rather than workload variability. Other work has considered these two factors together. In some of these approaches (e.g. [20], [23]), the task workloads are assumed to be exponentially distributed. Other studies associate a specific unique workload with each particular state (e.g. [24]). When the number of possible workload values is high due to high variability, and when concurrent tasks have correlated workload PDF's, these models become inadequate. In our approach, we define novel stateful models. Each state is defined based on an application-specific feature, and, instead of a single aggregate multi-dimensional workload PDF describing the entire application, we obtain one conditional PDF per state that describes the workload variation when the application is in that particular state.

Workload variability for parallelized MPEG2 decoding.

This paper presents a general stochastic modeling and optimization framework and uses MPEG2 decoding only as a case study. Previous work on characterizing MPEG2 workload variability is therefore relevant but not central to our work. To the best of our knowledge, there is no statistical characterization work that targets spatial correlations or the particular parallel MPEG2 decoding implementation we worked on. For other video decoding applications broken into small tasks (that are still run on a uniprocessor), [10] presents a task workload model. [25] characterizes worst-case workloads for MPEG2 decoding on a uniprocessor. [14] builds and empirically fits parameters to a model for predicting the workload of decoding MPEG2 frames. The workload prediction is stateless (apart from the use of frame types) and does not model workload correlations over time. This model uses very low-level, application-specific features. Workload characteristics of MPEG2 decoding has also been studied in the context of benchmarking suite design (e.g. [26]).

III. BACKGROUND

A. The Software Model

On a multi-core processor, there are precedence relations among tasks of a parallel application due to data dependencies or the fact that two tasks are assigned to run on the same core. These precedence relations are represented by a task graph [11], [27]. A task graph (e.g. Fig. 2) is a directed acyclic graph $G = (V, E)$ in which nodes represent tasks and each edge $(u, v) \in E$ represents the fact that task u must finish before task v can start.

We assume that the tasks have been scheduled and assigned to multiple cores with DVS capability, and our formulations guarantee optimal voltage selection for this fixed scheduling and task assignment. For a different scheduling and/or task assignment, the optimum found by our approach will be different but the application does not need to be re-characterized

statistically. The effects of task scheduling and assignment are entirely captured by the task graph. For a different scheduling and assignment, the new optimal voltage selection problem can be formulated automatically. Thus, our methodology can be used in the inner-loop of an algorithm (e.g. [28]) which considers all of these problems jointly.

We represent the execution time of task u by the variable t_u . In a given period, given the values of t_u for each $u \in V$, the finishing time of each task u (denoted by F_u) and the completion time of all tasks (denoted by F_{DAG}) can be found by a topological traversal of the task graph.

Since the application processes different input data in each period, the execution times of tasks, and consequently, F_u 's and F_{DAG} are different in each period. For such an application, timing constraints can be expressed either as deadlines on the finishing times of individual tasks (D_u on F_u) or a global deadline on the completion time of all tasks (D_{DAG} on F_{DAG}). In a soft real-time application, it is sufficient to satisfy the timing constraints with a probability of P_{CON} , called the probabilistic performance constraint.

B. Hardware Architecture, Dynamic Voltage Scaling

In this paper, we consider a homogeneous, shared-memory system architecture in which each processing core has a local cache. We assume that each core in the multi-core system has independent DVS capability. The speed of a core is quantified by the cycle-time (CT), which is a function of the voltage setting as follows

$$CT(V_{ddu}) = k_{ct} \frac{V_{ddu}}{(V_{ddu} - V_{th})^{a_{ct}}} \quad (1)$$

in which V_{ddu} is the core's voltage setting for task u , k_{ct} , a_{ct} and V_{th} are technology-related constants [11]. For our experiments, we obtained the values of these parameters from a commercial processor with DVS capability, in which the core voltage can range from 0.85 (V_{min}) to 1.55 (V_{max}) volts [29]. We assume that the voltage settings of the cores are continuously adjustable. By not constraining the voltages to discrete values, we are able to quantify the maximum energy savings that can be obtained using DVS. The energy-management policies presented in this paper can be applied to processors with discrete operating modes in one of two ways. One can solve the optimization problems presented in Section VII assuming continuously-adjustable voltages and then round up each task voltage in the optimal solution to the closest available higher voltage level. This approach preserves the probabilistic guarantees provided by the problem formulation but sacrifices some energy savings. Alternatively, to obtain the exact optimum for a processor with discrete modes, the problem formulations in Section VII can be interpreted as integer programming problems, and, at the cost of more complex optimization algorithms, exact optimum solutions can be computed.

When a task u is executed at a voltage setting V_{ddu} , its execution time becomes $t_u = CT(V_{ddu}) L_u$ in which L_u represents the workload of the task u , i.e., the number of clock cycles spent to execute a task. The dynamic energy consumed by the CMOS circuitry in a core executing a task

with workload L_u and at a voltage setting of V_{ddu} is computed by $E_{dyn_u} = C_u L_u V_{ddu}^2$, where C_u denotes the effective switching capacitance that is a measure of the core utilization by task u . We consider only dynamic energy in this work. The energy consumption model above can be easily modified to include additional consumption due to leakage currents and/or the overhead due to voltage transitions [30], [31], and our formulations can handle them with little or no modification. It is important to note that our approach can be easily adapted to model multi-core systems with heterogeneous cores or components such as the communication interface between cores, disk drives or display devices.

C. Definitions

We define an *action* (\mathcal{V}) as a tuple $(V_{dd1}, \dots, V_{dd|V|})$ that specifies a voltage setting V_{ddu} for each task $u \in V$ in a task graph $G = (V, E)$. We use the term *policy* (β) to refer to a method that chooses an action deterministically at each period. A policy can use either a single action or a set of actions.

We first define the problem of energy management with deterministic workload (EMD). In this problem, the workloads of tasks are known before their execution and the goal is to find the optimal action. The solution of the EMD problem is used to obtain the theoretical bound for maximum achievable energy savings, as we explain in Section VII-A.

Definition 1: Given a task graph (V, E) and a set of hard timing constraints, energy management with deterministic workload (EMD) is the problem of finding an action which minimizes the total energy consumption subject to these constraints for given task workloads $(l_1, \dots, l_{|V|})$.

In this paper, we address the problem of energy management with stochastic workload (EMS) in which the joint distribution of task workloads is known and the timing constraints can be violated probabilistically. The EMS problem is formally defined as follows:

Definition 2: Given a task graph (V, E) , a set of timing constraints and a probabilistic performance constraint P_{CON} , energy management with stochastic workload (EMS) is the problem of finding an optimal policy β which minimizes the expected total energy consumption and satisfies the timing constraints with a probability of P_{CON} for randomly varying task workloads $(L_1, \dots, L_{|V|})$.

IV. MOTIVATING EXAMPLES

We consider a simple application composed of two tasks running on a single-core system. Tasks have to be finished within a deadline of 2 seconds, and each task can have a workload of 2 or 10 instructions per period with equal probability. The processing core has two operating modes where it can execute 6 (10) instructions-per-second in the slow (fast) mode.

Spatial correlations. When there are significant spatial correlations among the workloads of tasks executed in the same period, energy management based on the assumption of statistical independence may either not achieve the desired probabilistic performance or be suboptimal. Table I shows three extreme cases in which task workloads are independent, fully positively and fully negatively correlated. For example,

TABLE I
PROBABILITY DISTRIBUTIONS FOR THE MOTIVATING EXAMPLE

(L_1, L_2)	Independent	Positive	Negative
(2,2)	0.25	0.5	0
(2,10)	0.25	0	0.5
(10,2)	0.25	0	0.5
(10,10)	0.25	0.5	0

in the third column, task workloads are fully negatively correlated, and consequently, the probabilities of the workload combinations (2, 2) and (10, 10) are zero. In all of the three cases, each task still has low or high workload with equal (marginal) probability. In the following two scenarios, suppose, for the sake of illustration, that we are unaware of spatial correlations and assume erroneously that task workloads are independent, represented by the first column in Table I.

In the first scenario, suppose that, in reality, the task workloads are positively correlated, and that the probabilistic performance constraint is 0.75, which means that the deadline can be violated in one-fourth of all periods. Given this performance constraint, we switch the core to the slow mode in which it can execute 12 instructions within 2 seconds. Since we are under the impression that the deadline is met for all workload combinations except for (10, 10) in this mode, we presume that the desired probabilistic performance is achieved. However, in reality, this decision would achieve a completion probability of 0.50 instead of the required 0.75, since the true probability of (10, 10) is higher than its expected value of 0.25 due to positive correlation. Therefore, if we ignore spatial correlations when they actually exist, we are unable to achieve the desired probabilistic performance.

In the second scenario, suppose that the probabilistic performance constraint is 1.0 (a hard timing constraint) and that task workloads are negatively correlated. Given this performance constraint, we switch the core to the fast mode in which it can execute 20 instructions within 2 seconds to meet the deadline in all workload combinations. However, this decision results in excessive energy consumption, since the workload combination of (10, 10) never occurs in reality due to negative correlation. This also illustrates that spatial correlations can be exploited even for applications with hard timing constraints, because tasks do not necessarily have their worst-case workload in the same period.

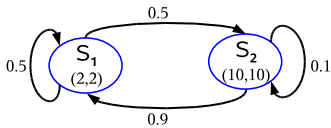


Fig. 1. Stateful stochastic model for the motivating example

Temporal correlations. For an illustration of how application features and temporal correlations, in addition to spatial correlations, can be exploited for energy management, consider the same two-task application running on the single-core system described above. Let task workloads be fully positively correlated spatially, i.e., only the workload combinations (2, 2) and (10, 10) have a nonzero probability of occurrence. Suppose that these workload combinations

occur according to the stochastic process shown in Fig. 1, corresponding to a stateful, Markov chain model. With this model, the steady-state occurrence probabilities for the states S_1 and S_2 are 9/14 and 5/14 respectively. Suppose that the desired completion probability is 0.95. If we disregard the stateful stochastic nature of the application we would be constructing an energy management policy based on only the knowledge of the steady-state (marginal) probabilities given above. In this case, we need to operate the core in the fast mode all the time in order to meet the desired completion probability, saving no energy. We can improve this policy slightly by using a randomized scheme which operates the core in the fast mode with a probability of 0.95, still meeting the completion probability constraint. However, it is possible to construct more energy efficient policies by exploiting the stateful nature of the application. A policy which switches the core to the fast (slow) mode whenever the application is in S_1 (S_2) workload state in the previous period is guaranteed to meet the deadline with a probability of $1 - 0.1 \times 5/14 \approx 0.96$, and achieves significant energy savings by operating the core in the fast mode with a probability of only $9/14 \approx 0.64$. We call a policy of this sort *previous-state observable*, because policy decisions are based on observations of the state of the application in the previous period for which execution has been completed. Now, suppose that the application has an identifiable feature which enables us to observe the state of the application at the beginning of each period before execution. A policy which switches the core to the slow mode when the application is in state S_1 in the current period and to the fast mode when it is in S_2 will achieve even more energy savings by operating the core in the fast mode with a probability of only $5/14 \approx 0.36$. This policy also guarantees that the desired completion probability is met. By analogy, a policy of this sort is called *current-state observable*.

V. STOCHASTIC APPLICATION MODELS

A. Stateless Stochastic Model (SSM)

In the stateless model, task workloads (L_u 's) are treated as discrete-valued random variables with arbitrary distributions, and as a discrete-time stochastic process. Time discretization is done at period boundaries. The distribution of task workloads in a given period is represented by a joint probability mass function called p_{joint} which is defined over a multidimensional sample space \mathbf{L}_{space} as follows

$$p_{joint}(l_1, \dots, l_{|V|}) = \mathbf{prob}\{L_u = l_u, u = 1, \dots, |V|\} \quad (2)$$

Each point in \mathbf{L}_{space} is a unique $|V|$ -tuple of task workloads occurring with a nonzero probability. The points in \mathbf{L}_{space} are essentially the collection of unique workload combinations encountered in the training set used for the statistical characterization of the application. More specifically, \mathbf{L}_{space} can be represented by a matrix in which each column represents a task and each row corresponds to the collection of actual task workload values for a particular period in the training set. The size of \mathbf{L}_{space} grows linearly both in the number of tasks and the number of periods in the training set. Since our optimization problems are solved off-line, this linear increase in the size of the sample space \mathbf{L}_{space} is not a major concern.

If we had assumed that the task workloads are independent of each other, then the sampling points in \mathbf{L}_{space} would have been obtained with a Cartesian product of the sampling point sets for workloads of individual tasks. In such a case, the dimension of \mathbf{L}_{space} would have been exponential in the number of tasks.

As an example, the p_{joint} of the joint distribution shown in the second column of Table I can be described as follows

$$\begin{aligned} p_{joint}(2, 2) &= 0.5 & p_{joint}(2, 10) &= 0 \\ p_{joint}(10, 2) &= 0 & p_{joint}(10, 10) &= 0.5 \end{aligned}$$

As this example shows, p_{joint} captures both workload variability and spatial correlations (in a given period) among the task loads. However, it can not capture workload variation over time and temporal correlations, since it does not include timing information. The energy management policy β_{SSM} which uses the same action in each period at runtime optimally solves the problem expressed by the SSM model.

B. Stateful Stochastic Models

In all of our stateful models, states $\{s_1, \dots, s_m\}$ are defined using application-specific features or extracted using generic techniques such as hidden Markov modeling. When moving to a stateful model, the task decomposition and the task graph for the application are not changed. State transitions are allowed to occur only at period boundaries and each task in the task graph is executed exactly once per period. In other words, during one entire period, i.e., one pass over the task graph, the application remains in the same state¹ but each task may be run at a different voltage setting. It is possible in principle to allow the application state to change more frequently, e.g., after each task, but this increases the complexity of the optimization problem and DVS overhead. In our setting, the probabilistic performance of a policy can be formulated analytically using the parameters of the stateful models as will be shown in Section VII-B.2 and VII-B.3.

In the stateful models, the state concept is used to obtain conditional workload distributions from the overall, aggregate multi-dimensional workload distribution used in our stateless model. The conditional distribution in each s_i is described by $p_{joint|i}$. The aggregate workload distribution p_{joint} can be expressed in terms of $p_{joint|i}$'s using Bayes' formula as follows

$$p_{joint}(l_1, \dots, l_{|V|}) = \sum_{i=1}^m \pi_i p_{joint|i}(l_1, \dots, l_{|V|}) \quad (3)$$

in which π_i is the limiting, steady-state probability of s_i . The key distinction between the first two stateful models to be described is whether the application state can be observed at the beginning of a period, before tasks are executed, or at the end of a period, after tasks are executed.

The temporal correlations between task workloads can be captured with a generalized multi-dimensional PDF for task workloads of several consecutive periods considered jointly.

¹Note that the sense in which we use the term "state" here refers to a mode or feature of the application. This is different from "processor state" which refers to a voltage setting or different sleep modes made available by processor cores.

Such a joint PDF would be defined over a multi-dimensional space generated by repeated Cartesian products of \mathbf{L}_{space} with itself. Even though it is very general, this model becomes quite unwieldy when more than a few consecutive periods need to be jointly considered. On the other hand, a stateful model with an associated state transition structure and state-conditioned workload distributions captures temporal correlations in an indirect but compact and structured manner.

1) *Current-State Observable (CSO) Model:* In the CSO model, the application state is observable at the beginning of each period, before the tasks are executed. Therefore, the conditional distribution of task workloads in each period is known before execution ($p_{joint|i}$ if the state of the period is s_i). For example, a CSO model can be constructed for MPEG2 video decoding in which states are defined according to the frame type information. Since the type of a frame is known before it is decoded, the state of the current period will be observable before the tasks are executed.

The CSO model leads to an energy management policy β_{CSO} which uses a unique action \mathcal{V}_i for each s_i . At runtime, action \mathcal{V}_i is used whenever the current state is s_i . Each action is optimized according to the conditional joint workload distribution in the associated state. The improvement brought by β_{CSO} over β_{SSM} depends on the dissimilarity of conditional joint workload distributions in different states. If task workloads have exactly the same distribution in each state, a CSO model is indistinguishable from an SSM model.

2) *Previous-State Observable (PSO) Model:* In the PSO model, the application state becomes observable (identifiable) only at the end of a period rather than at the beginning. For example, a PSO model can be constructed for MPEG2 video decoding in which states are defined based on the actual workload of decoding a frame, as described in Section VI. Since the total actual workload is known only after a frame is completely decoded, only the state of the previous period is observable at the beginning of the current period.

In the PSO model, it is assumed that state transitions possess the Markov property, and consequently, a PSO model corresponds to a Markov Chain [32]. The state transitions are represented with a transition probability matrix A in which a_{ij} is the probability at s_i of taking a transition from state s_i to s_j . The state of the current period depends on the state of the previous period and its outgoing transition probabilities.

The PSO model leads to an energy management policy β_{PSO} that chooses action \mathcal{V}_i whenever the state in the previous period is s_i . As a result, each action \mathcal{V}_i needs to be optimized based on the weighted workload distribution in all states reachable from s_i in a single transition. This joint workload distribution of tasks experienced by \mathcal{V}_i is represented with p_{joint}^i and is computed by conditioning on reachable states from state s_i in a single transition as follows

$$p_{joint}^i(l_1, \dots, l_{|V|}) = \sum_{j=1}^m a_{ij} p_{joint|j}(l_1, \dots, l_{|V|}) \quad (4)$$

3) *Hybrid-State Observable (HSO) Model:* In the HSO model, the application state is defined based on a combination of features some of which are identifiable at the beginning of a period, and some observable only at the end of a period after

execution. For example, an HSO model can be constructed for MPEG2 video decoding in which states are defined according to the combination of frame type for the frame about to be decoded in the current period and the amount of total actual workload for the frame whose decoding has just been completed, as described in Section VI. The HSO model leads to an energy management policy β_{HSO} similar in structure to β_{PSO} . However, the inclusion of a feature that is identifiable at the beginning of a period in the state definition can potentially help refine the workload PDF in the current period. This extra information is likely to result in more effective energy management than β_{PSO} .

VI. CASE STUDY: MPEG2 VIDEO DECODING

A. Preliminaries

For our case study, we focused on MPEG2 decoding where a compressed still image (a frame) is decoded in each period. These frames can be of three types: intra (I), predicted (P) and bidirectionally predicted (B). Each frame contains a number of horizontal slices, which are further partitioned into macroblocks.

We used the decomposed MPEG2 video decoding implementation by Olukotun et al. [33]. We worked with a particular slice-based parallel task decomposition that was found to yield the best performance in terms of load balancing, communication and synchronization overhead among tasks.

The workload of each slice in an MPEG2 frame is divided into two separate tasks denoted by VLD (variable-length decoding and inverse quantization) and MC (motion compensation and inverse discrete cosine transform). The task graph representing this decomposition for a frame with 19 slices is shown in Fig. 2. In this figure, tasks are assigned to four processing cores denoted by $C_1 - C_4$. The numbers within ovals correspond to slice numbers, and the dashed arrows represent precedence relations among tasks assigned to the same core. The execution of each MC task has to wait for the completion of the VLD task of the same slice, while the MC and VLD tasks of separate slices can be executed concurrently.

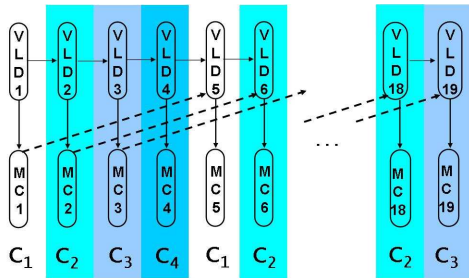


Fig. 2. The task graph of MPEG2 video decoding

In this work, we used two movies (Girl With a Pearl Earring and Spiderman 2) for training and evaluation data. We obtained six 30-minute clips from these two movies each of which is encoded at a frame rate of 25 fps and a resolution of 544×304 . For the construction of the stochastic models for MPEG2 video decoding, two of the six clips are used as training data in the construction of the joint workload distribution and associated statistical measures. These two clips have a total of some 90,000 frames. We use the term

training set to refer to these two clips. The remaining four clips are intentionally left out of the training set to determine how accurately stochastic application models can predict the workload in unknown movies.

In order to obtain the joint workload distribution of tasks (p_{joint}), the application is first executed on the training set. The workloads of tasks in each period are measured using hardware performance counters for various types of low level events in a core. To access these performance counters, the low level interface of PAPI [34] is used. PAPI offers increased accuracy and introduces negligible overhead compared with other approaches to workload measurement. The workload data collection was done on a computer with a single-core processor (Intel Pentium M (Dathon)) operating at its maximum clock frequency (1.5 GHz) and 512 MB of main memory. The processor has a cache system which includes an L1 instruction (32 KB), an L1 data (32KB) and a unified L2 cache (2048 KB) on the chip. The computer ran Linux (kernel v.2.6.16), and the Linux kernel was patched with perfctr (v.2.6.21), a driver required by PAPI (v.3.2.1) to access the hardware performance counters. We used this platform as an approximation since an instrumented, true multi-core platform where each core's voltage is independently adjustable and where task assignment to processors is under the application control was not available.

Workload data collection was repeated several times for each input data in order to be able to filter out effects of context switches. In order to account for cache misses, we also investigated and collected experimental data for a more detailed workload model which considers the breakdown of workload into its computation and memory access components. For MPEG2, our experiments showed that memory operations constitute less than 10% of the total workload and that the major source of workload variation is the computation component. Thus, for the platform we considered, we use a simple workload model with only the computation component for the results we present in this paper. For other platforms, memory bandwidth may be a bottleneck and energy management may have to explicitly consider the memory access component of the workload.

The values of task workloads measured in each period are stored as separate observations in a trace file denoted by Q . The collection of all unique observations in Q constitutes the sample space denoted by L_{space} . The probability of each unique observation $(l_1, \dots, l_{|V|})$ in L_{space} is given by

$$p_{joint}(l_1, \dots, l_{|V|}) = \frac{\# \text{ of periods } (l_1, \dots, l_{|V|}) \text{ was observed in } Q}{|Q|} \quad (5)$$

in which $|Q|$ denotes the total number of observations in Q . For stateful models, Q is partitioned into a set of disjoint subsets where Q_i contains the observations labeled as state s_i . The conditional probability $p_{joint|i}(l_1, \dots, l_{|V|})$ of each unique observation $(l_1, \dots, l_{|V|})$ in s_i is computed by replacing Q with Q_i in the equation above. The occurrence frequency (steady-state probability) of each state is computed as $\pi_i = \frac{|Q_i|}{|Q|}$.

The state transition matrix A is also computed using the trace file where each a_{ij} is computed as follows

$$a_{ij} = \frac{\# \text{ of times state changes from } i \text{ to } j}{|Q_i|} \quad (6)$$

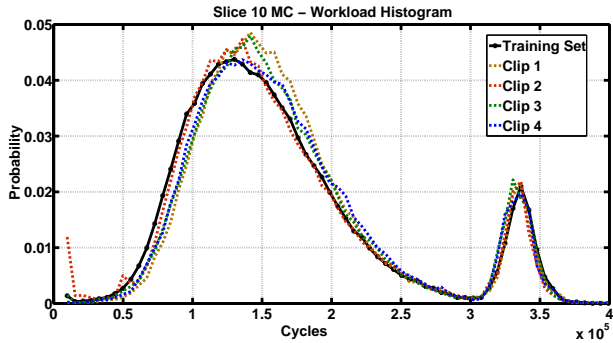


Fig. 3. Workload distribution of MC for the center slice

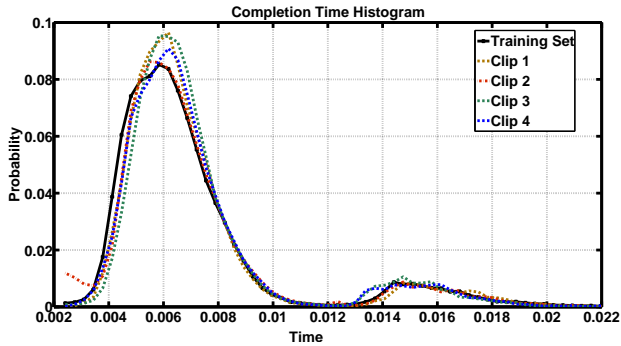


Fig. 4. The completion time histogram at maximum voltage

B. Stateless Stochastic Modeling of MPEG2

This section presents MPEG2 workload statistics extracted from the two clips in the training set and the other four movie clips. In all figures, the solid line represents the training set, while the dashed lines represent the other four clips.

Fig. 3 shows the workload distribution of the MC tasks of the slice in the center of each frame. The MC tasks of other slices also have similar workload distributions. It was shown in [1] that for the VLD task, the shape of the curve is different from the curve for MC, but the VLD workload curves for different slices are very similar. For both MC and VLD tasks, the maximum, average and the minimum workload values are significantly different from each other. The very long tails of these PDF's imply that treating MPEG2 decoding as a soft rather than hard real-time application offers the potential for significant energy savings.

Fig. 4 shows the completion time distribution of the task graph when all tasks are executed at the maximum voltages of the cores. We observe that there is a significant difference among the maximum, average and the minimum completion times, the ratio of maximum to minimum and maximum to mean are around 10 and 3 respectively. Moreover, the maximum completion time in Fig. 4, which we call the *correlated completion time*, is about 50% lower than the *uncorrelated completion time* computed by a topological traversal of the task graph in which each task has its individual worst-case workload observed in the training set. This difference means that tasks never have their worst-case workloads in the same period. As a result, for MPEG2 decoding, the assumption of statistical independence of task workloads will lead to pessimistic and suboptimal energy management.

We observed that the aggregate probability distributions of task workloads can be quite similar for movie clips which

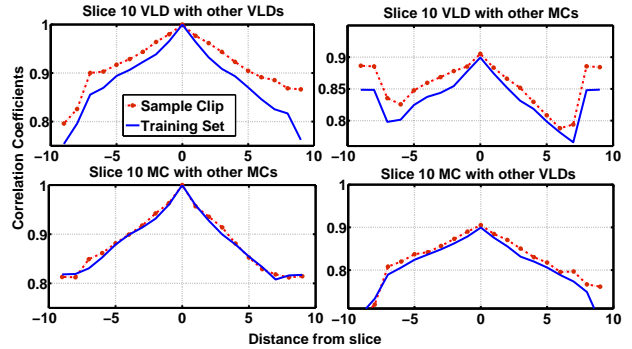


Fig. 5. Correlation coefficients representing spatial correlations

contain more than ten thousand frames and multiple scenes. For example, Fig. 3 and Fig. 4 show the distribution of a randomly selected task's workload and the completion time in all of the clips. In these figures, the overall shapes of the histograms are similar for different clips even though the minimum (best-case), mean and maximum (worst-case) workload values still vary from clip to clip. When an application has tolerance to deadline misses, the probability mass past a certain workload (the tail mass of the workload distribution) becomes the determining factor for energy management rather than workload values of extreme, corner cases. The similarity of the histograms referred to above implies that even though the extreme workload values depend on the input data, the aggregate distributions of task workloads can be very similar for two different inputs each containing multiple scenes and thousands of frames. Therefore, the probability distribution assembled from a training set appears to represent well the long-term statistics of inputs not contained in the training set. This observation is experimentally confirmed in Section IX.

Finally, Fig. 5 shows the correlation coefficients of the workloads of the VLD and MC tasks of the center slice with the workloads of other VLD and MC tasks as a function of the distance of the slices from the center slice. Dotted lines show data obtained from a sample clip and solid lines represent aggregate statistics over all six clips. This figure shows that the tasks of slices close to each other are highly correlated. Moreover, the MC and VLD tasks of a particular slice are also highly correlated with each other.

C. Stateful Stochastic Modeling of MPEG2

We now describe four stateful models for MPEG2. All models are constructed using the workload data in the training set. In these models, states are defined according to the following features, which have also been used in previous approaches [13], [15], [16]:

- Frame Type: I, P, or B.
- Frame Load: Total workload for decoding a frame (period).
- Frame Size: Compressed data size for a frame.

In the first CSO-type stateful model of MPEG2, states are defined based on frame type, which is observable before a frame is decoded. We denote this model CSO^T and the energy management policy based on this model by β_{CSO}^T . This model has three states.

In our second model of CSO-type, we define states based on both frame type and frame data size. In order to define states, the frame data size observed in the training set is quantized

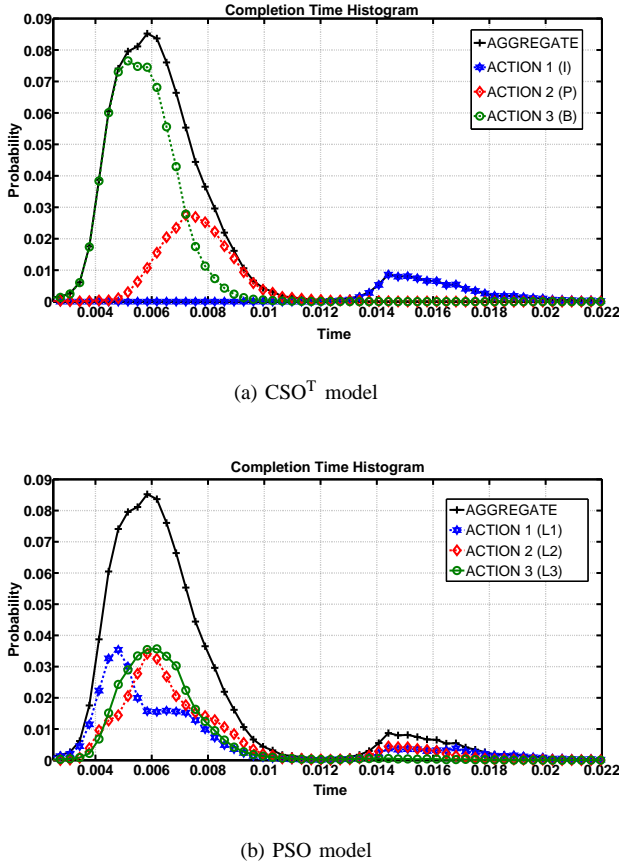


Fig. 6. The completion time histograms for the stateful models

into three bins per frame type with equal number of frames falling into each bin. As a result, this model has a total of $3 \times 3 = 9$ states. We denote this model CSO^{TS} and the energy management policy based on this model by $\beta_{\text{CSO}}^{\text{TS}}$.

In the third stateful model for MPEG2, of PSO-type, states are defined based on total frame workload, which becomes available only after a frame is decoded. In order to define states, the total frame workload observed in the training set is also quantized into three bins, resulting in three states. Our final and fourth stateful model is a hybrid one, of HSO-type, where state definition is based on frame type (of the current frame) and frame load (of the previous frame) quantized into three bins, resulting in a total number of nine states. The energy management policies based on the PSO and HSO models are denoted by β_{PSO} and β_{HSO} respectively.

Fig. 6 shows the conditional distribution of the completion time for the CSO^{T} and PSO models at maximum voltage, with the completion time distribution in each state s_i (represented by $p_{\text{joint}|i}$) of the CSO^{T} model in Fig. 6(a), and the completion time distribution associated with each action \mathcal{V}_i (represented by p_{joint}^i) of the PSO model in Fig. 6(b). In other words, each histogram in Fig. 6(b) is the conditional workload distribution for the next state if the previous state is s_i . Each action in $\beta_{\text{CSO}}^{\text{T}}$ and β_{PSO} will be optimized using the distributions in Fig. 6(a) and 6(b) respectively. In both figures, the completion time distribution obtained using the overall, aggregate data for all of the states is also shown.

Fig. 6(a) shows that defining states according to frame type results in dissimilar conditional workload distributions in the states of the CSO^{T} model. Moreover, this model can capture the hump on the right as a distinct state shown by the solid line in Fig. 6(a). On the other hand, defining states according to only frame load in the pure PSO model results in less dissimilar workload distributions for different actions as shown in Fig. 6(b). This means that the temporal correlation between the workloads of tasks in consecutive frames is not very significant when the frame type is disregarded. Therefore, it is expected that the policy optimized using the PSO model, i.e., β_{PSO} , will be less effective in saving energy than $\beta_{\text{CSO}}^{\text{T}}$ in the case of MPEG2 video decoding. However, the energy management policy based on the HSO model described above is guaranteed to perform better than both $\beta_{\text{CSO}}^{\text{T}}$ and β_{PSO} , because it is constructed by merging the two and can jointly exploit the dependence of workload on frame type and workload correlation between consecutive periods.

VII. ENERGY MINIMIZATION PROBLEM FORMULATIONS

A. Energy Management with Deterministic Workloads

The optimization problem for the EMD problem (defined in Section III-C) had been formalized in earlier work [1], [11]. If an EMD problem is solved separately for every period in a given input stream of an application, then the theoretical bound for energy savings can be found. This energy management policy, which we call β_{BND} , is not realizable since task workloads are not known in advance. In Section IX, we use β_{BND} to evaluate the effectiveness of the energy minimization policies presented in the following sections.

B. Energy Management with Stochastic Workloads

1) *Formulation for SSM*: The objective of a β_{SSM} policy, based on a stateless stochastic model described in Section V-A, is to minimize the average energy consumption in the long run while meeting a probabilistic performance constraint. The solution of this problem results in an energy management policy which uses the same optimized action $\mathcal{V} = (V_{dd1}, \dots, V_{dd|V})$ in every period. We denote this problem by EMS_{SSM} . A formulation for EMS_{SSM} was presented earlier in [1].

2) *Formulation for CSO*: The energy management policy β_{CSO} , based on a CSO model, has a unique action for each state: It uses action \mathcal{V}_i whenever it observes that the application is in state s_i in the current period. The optimal parameters for the actions of this policy can be computed by solving a two-level optimization problem which resembles the stateful structure of the underlying stochastic application model. In this two-level formulation, the optimization variables of the upper level are the conditional completion probabilities for actions (states): P_{CON}^i for \mathcal{V}_i . When the application is at state s_i , the timing constraint has to be satisfied at least P_{CON}^i percent of the time. The goal of the lower level is to find the optimal voltage settings (for minimum average energy consumption in state s_i) which achieve a conditional completion probability handed down by the upper level in the associated state. In a sense, the upper level acts as an optimal *mediator* in between the lower level optimizations performed disjointly for each state. This two-level formulation can be expressed as follows

Upper Level

$$\begin{aligned} & \text{find} && P_{CON}^i \text{ for each action } \mathcal{V}_i \\ & \text{minimize} && \sum_{u=1}^m \pi_i E_{tot}^i \\ & \text{subject to} && \sum_{u=1}^m \pi_i P_{CON}^i \geq P_{CON} \end{aligned}$$

Lower Level

$$\begin{aligned} & \text{find} && V_{dd_{u,i}} \in [V_{min}, V_{max}] \text{ for each task } u \\ & \text{minimize} && E_{tot}^i = \sum_{u=1}^{|V|} C_u \mathbf{E} [L_u | s_i] V_{dd_{u,i}}^2 \\ & \text{subject to} && P_{SAT_i}(V_{dd_{1,i}}, \dots, V_{dd_{|V|,i}}) \geq P_{CON}^i \end{aligned}$$

where $V_{dd_{u,i}}$ is the voltage setting for task u in action \mathcal{V}_i . E_{tot}^i , the objective function of the lower level problem, represents the conditional expected energy consumption conditioned on s_i . $\mathbf{E} [L_u | s_i]$ represents the conditional expected workload of task u in s_i and can be computed using $p_{joint|i}$. Finally, $P_{SAT_i}(V_{dd_{1,i}}, \dots, V_{dd_{|V|,i}})$, denoted compactly by $P_{SAT_i}(\mathcal{V}_i)$, is the conditional completion probability achieved by action \mathcal{V}_i in state s_i . Each one of the lower level problems is an instance of the EMS_{SSM} problem discussed in the previous section.

3) *Formulation for PSO*: The optimal energy management policy β_{PSO} also has a unique action for each state. Since the state of the period becomes observable only after the tasks are executed, β_{PSO} chooses action \mathcal{V}_i whenever it observes that the application was in state s_i in the previous period. This policy can also be constructed by solving a two-level optimization problem that incorporates the stateful structure of the underlying model. When selecting voltages for the current state, β_{PSO} only has the knowledge that the application was in state s_i in the previous period. Thus, differently from β_{CSO} , it has to base the selection of \mathcal{V}_i on a conditional probability distribution describing what the current state can be given that the application was in s_i in the last period. For this, a joint conditional PDF is obtained for the task workloads in the current period, and \mathcal{V}_i is optimally selected accordingly. The formalization of this problem is very similar in form to the β_{CSO} problem and is omitted because of space limitations.

4) *Formulation for HSO*: In an HSO model, the state definition is partially based on a feature that is identifiable before execution. Hence, this feature has the same value at all of the states reachable (in a single transition) from a particular state in the HSO model. For instance, in the HSO model we described in Section VI-C for MPEG2 that is based on the frame type (of the current frame) and the frame load (of the previous frame) quantized into three bins, there are only three single-step reachable states from every one of the nine states, corresponding to the three bins of frame workload and the identified frame type. State transition probabilities, a_{ij} 's, are extracted accordingly from the training set. Apart from this distinction, an HSO model has the same structure as a PSO model. Thus, the same optimization formulation can be used.

C. Multi-Core Energy Management and Activity Networks

We observed that multi-core energy management can also be formulated as an optimal activity network problem [2]. An activity network (AN) contains a set of activities and precedence relations among them represented by a directed acyclic graph. Each activity u has duration d_u , which is a function of a design variable x_u . The finishing time of all activities for a given set of d_u 's is called the makespan.

In a deterministic activity network (DAN), activity durations are deterministic functions of the design variables. A typical

objective in an optimal DAN problem is to find x_u 's that minimize the makespan subject to some timing and resource constraints. In a stochastic activity network (SAN), design variables determine the probability distributions of activity durations. Therefore, the makespan of a SAN is also a random quantity. A typical objective in an optimal SAN problem is to minimize the average value of the makespan or to keep it below a threshold value with a specified probability, subject to constraints on resources.

The task graph representation of a parallelized application is analogous to an activity network where the execution time t_u (voltage setting V_{dd_u}) of a task corresponds to the duration d_u (design variable x_u) of an activity. The completion time defined for a task graph corresponds to the makespan of an activity network. Consequently, EMD and EMS_{SSM} problems can be converted into optimal DAN and SAN problems.

VIII. SOLVING ENERGY MINIMIZATION PROBLEMS**A. Energy Management with Deterministic Workloads**

The optimization problem formulated for EMD in Section VII-A has a convex objective function. Since cycle-time is also a convex function of the voltage setting as expressed in (1), an EMD problem is a convex programming problem. Through several transformations, an EMD problem can be exactly formulated as a geometric program (GP) [35]. A GP is a convex optimization problem with a special form. As a result of its special structure, a GP can be solved efficiently and globally [35]. We formulate each EMD problem as a GP and solve it using a public domain software package [36].

B. Energy Management with Stochastic Workloads

Each EMS formulation becomes a constrained nonlinear optimization problem for which we use a sequential quadratic programming (SQP) solver (*fmincon* in MATLAB [37]). The probabilistic constraint function and its gradient are computed by explicit enumeration and finite differences respectively as had been presented in [1]. The SQP-based solution technique is used for the optimization of β_{SSM} . For the optimization of β_{CSO} , β_{PSO} and β_{HSO} , we use the solution technique described in the next section.

In solving EMS problems, repeated explicit traversals of the points in \mathbf{L}_{space} are necessary to compute completion probabilities under certain conditions. In our experiments, \mathbf{L}_{space} had a maximum size of 90000 which made it comparable to the largest MPEG2 workload datasets we have encountered in the literature. We found that the time taken for traversing \mathbf{L}_{space} did not constitute a bottleneck in the offline optimization runs we carried out. If much longer application runs are used to obtain \mathbf{L}_{space} , binning may be needed.

C. Stochastic Energy Management: SAN Heuristic

Our experiments have shown that the SQP-based solution technique described in the previous section is not robust when it is used to compute optimal policies for stateful models. This is due to the fact that the probabilistic constraint function is not numerically well-behaved in these cases. We devised a new, robust solution technique to overcome this difficulty. In this solution technique, the two-level formulations are replaced with another two-level formulation in which the EMS_{SSM}

problem in the lower level is approximately transformed into an EMD problem.

Kim et al. [2] proposed a heuristic approach for an optimal SAN problem that arises in statistical digital circuit design. The key idea behind this heuristic is to add increasing *margins* to the expected values of activity durations until the constraints of a given optimal SAN problem are satisfied. In this approach, a SAN problem is approximated by a DAN problem in which each stochastic activity duration is replaced with a deterministic quantity expressed in terms of its mean and standard deviation as follows

$$d_u'(x_u) = \mu(d_u(x_u)) + \kappa_u \sigma(d_u(x_u)) \quad (7)$$

where $\mu(d_u(x_u))$ and $\sigma(d_u(x_u))$ denote the mean and standard deviation of $d_u(x_u)$ respectively. κ_u , the *margin coefficient*, is used to account for the statistical variations of activity durations. For a given set of margin coefficients, an approximated DAN problem is solved to obtain optimal values of the design variables, x_u 's. These optimal x_u 's are then used to evaluate the probabilistic constraints to check/verify whether they are satisfied or not. In order to determine the optimal margin coefficients that guarantee the satisfaction of these constraints, several schemes have been discussed in [2] that are based on iterative or exhaustive search.

We incorporate the heuristic approach described above into the lower level of the two-level formulations described previously. We transform the EMS_{SSM} problems in the lower level into an EMD problem by using a state dependent margin coefficient for all tasks as follows

$$l_{(u,i)} = \mu(L_{ui}) + \kappa_i \sigma(L_{ui}) \quad (8)$$

For β_{CSO} , L_{ui} in (8) represents the workload of task u in state s_i . Therefore, $\mu(L_{ui})$ and $\sigma(L_{ui})$ are calculated using the conditional workload distribution $p_{joint|i}$. For β_{PSO} and β_{HSO} , L_{ui} represents the workload of task u in all states reachable with a single transition from s_i . Thus, $\mu(L_{ui})$ and $\sigma(L_{ui})$ are calculated using the workload distribution p_{joint}^i defined in Section V-B.2. Then, we change the optimization variables of the upper level from completion probabilities to margin coefficients specified for every action: κ_i for \mathcal{V}_i . In this way, we formulate the search for optimal margin coefficients as an optimization problem in the upper level. The new two-level formulation for all stateful models can be expressed as follows

$$\begin{array}{ll} \text{Upper Level} & \\ \text{find} & \kappa_i \text{ for each action } \mathcal{V}_i \\ \text{minimize} & \sum_{u=1}^m \pi_i E_{tot}^{\kappa_i} \\ \text{subject to} & \sum_{u=1}^m \pi_i P_{SAT}^{\kappa_i} \geq P_{CON} \\ \text{Lower Level} & \\ \text{find} & V_{dd_{u,i}} \in [V_{min}, V_{max}] \text{ for each task } u \\ \text{minimizing} & E_{tot}^{\kappa_i} = \sum_{u=1}^{|V|} C_u \mu(L_{ui}) V_{dd_{u,i}}^2 \\ \text{subject to} & \\ & t_u - CT(V_{dd_{u,i}}) l_{(u,i)} = 0 \\ & t_u - F_u \leq 0 \\ & F_u + t_v - F_v \leq 0 \\ & F_u - D_u \leq 0 \\ & F_u - D_{DAG} \leq 0 \end{array}$$

In the above formulation, the variables of the upper level are the margin coefficients, while the variables of the lower

level are the voltage settings in \mathcal{V}_i . $P_{SAT}^{\kappa_i}$ is the conditional completion probability achieved by action \mathcal{V}_i in a state.

The new two-level formulation for the EMS problem has two key advantages. First, $P_{SAT}^{\kappa_i}$ (as a function of κ_i 's) is smoother than $P_{SAT_i}(\mathcal{V}_i)$ (defined in Section VII-B.2), because $P_{SAT}^{\kappa_i}$ increases monotonically with κ_i . This smoothness eliminates the inaccuracy experienced due to the discontinuities of $P_{SAT_i}(\mathcal{V}_i)$. Second, the lower level problem can be solved much more efficiently since it is an EMD problem that can be formulated as a GP.

We solve the upper level of the new two-level formulation using SQP. We compute the gradients of the objective and the constraint functions for this level by finite differences. We solve each EMD problem in the lower level by formulating it as a GP problem as described in Section VIII-A. The only drawback of this solution technique is the fact that spatial correlations among task workloads are no longer explicitly exploited, since the variation of task workloads are taken into account using the same margin coefficient for all tasks. In Section IX, we present results which show that the spatial correlations are implicitly exploited by this formulation to a degree, and the sub-optimality due to the heuristic approximation employed is not practically significant.

In the two-level formulation for the stateful models, the number of optimization variables for the upper level increases linearly with the number of states. The computational cost of solving the upper level problem and the number of lower-level problems that need to be solved in each upper level iteration depend on the number of states. In our experiments, we have found that the total CPU time required for solving two-level problems grows linearly with the number of states when the size of \mathbf{L}_{space} is kept fixed. In all problems, some 100 states can be handled with reasonable resource consumption. We believe that it is unlikely that an application will give rise to a much larger number of identifiable states with sufficiently dissimilar probability distributions.

IX. RESULTS ON MPEG2 CASE STUDY

For all of the results we present, the training and evaluation data sets described in Section VI-A were used. Technical issues related to our experimental setup are listed below to help interpret our results.

- All energy management policies are optimized for the set of completion probabilities $P_{CON} = \{0.85, 0.90, 0.95, 0.99\}$. $P_{CON} = 0.99$ models an (almost) hard timing constraint, while other completion probabilities reflect different choices for the deadline miss tolerance.
- All energy management schemes are optimized using stochastic models compiled from the workload data collected from the first and the fourth clips (the training set). The other four clips are used for evaluation only.
- Two different global deadlines are used as the timing constraint. The *relaxed deadline* refers to the uncorrelated completion time described in Section VI-B and it is computed based on the assumption that task workloads are independent. We use this deadline to be able to compare our schemes with previous approaches. The *tight deadline* corresponds to the

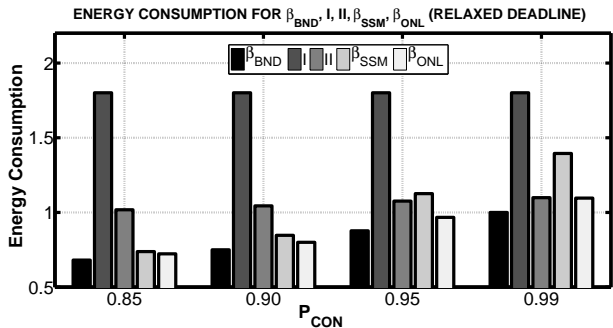


Fig. 7. Energy consumption comparison for the relaxed deadline case.

correlated completion time described in Section VI-B. The tight deadline is computed based on a more realistic scenario since tasks do not have their worst-case workload in the same period in MPEG2 video decoding.

- The reported energy values are the total energy consumption for the frames that are finished before the deadline. This is done to enable a fair comparison between all approaches. All reported energy values are normalized to the theoretical bound achieved by β_{BND} (described in Section VII-A) for the maximum completion probability.

In the rest of this section, the discussion focuses mainly on the energy savings achieved by different techniques. The CPU times required to solve the optimization problems were reasonable in all cases, considering that these problems are solved offline once per platform. On an Intel Pentium D CPU 3.20GHz machine with 2 GB of RAM running Linux Suse 9.3, a typical optimization problem for EMD (Section VIII-B) took 4 CPU seconds, with the GP formulation. For the stateless stochastic model, a typical optimization run took about 15 CPU minutes with 90,000 points in \mathbf{L}_{space} . The CPU times needed for computing optimal solutions to stateful models grow roughly linearly with the number of states. With 90,000 states in \mathbf{L}_{space} , for 3, 9, 27 and 81 states in stateful models, computing optimal solutions took 10, 35, 103, and 255 CPU minutes respectively.

A. Comparisons with Previous Approaches

The following four schemes are compared:

- \mathbf{I} is the energy management scheme proposed in [11] for hard real-time applications on multi-core systems.
- \mathbf{II} is the energy management scheme proposed in [12] for soft real-time applications on multi-core systems. This is a hybrid, heuristic scheme with off-line computations and on-line adjustments.
- β_{SSM} is the energy management policy described in Section V-A. The optimized set of voltage settings are stored in a look-up table and used without any runtime adjustments.
- β_{ONL} is the energy management policy which uses the optimized voltage settings of β_{SSM} above, and makes runtime adjustments to reclaim timing slacks whenever possible while still guaranteeing the completion probability achieved by β_{SSM} as explained in [1].

We compare our policy β_{SSM} with \mathbf{I} (which was proposed for applications with hard timing constraints) to reveal how much energy savings can be achieved when the application

has some tolerance to deadline misses. We compare β_{SSM} with \mathbf{II} , since it addresses the same problem treated in this work. There are two noteworthy differences between β_{SSM} and \mathbf{II} : First, \mathbf{II} ignores the spatial correlations among task workloads. Second, \mathbf{II} is based on greedy heuristics. Since \mathbf{II} performs runtime computations to further exploit timing slacks, we also compare it with β_{ONL} , which is a runtime heuristic based on our policy β_{SSM} . For a fair comparison, we do not include our policies based on stateful models in these comparisons, because the previous approaches \mathbf{I} and \mathbf{II} do not employ states.

All energy management schemes are trained, optimized, and simulated with the relaxed deadline due to the fact that \mathbf{I} and \mathbf{II} fail to work with the tight deadline. Since \mathbf{I} and \mathbf{II} assume the statistical independence of task workloads, they require that the deadline be greater than or equal to the relaxed deadline.

Because of space constraints, we do not report the completion probabilities achieved by the above energy management schemes, but we provide a summary of the results obtained. \mathbf{I} achieves full completion all the time since it is intended for hard real-time applications. \mathbf{II} behaves conservatively and overshoots the completion probability by 3% to 13% for $P_{CON} < 0.99$. On the other hand, β_{SSM} and β_{ONL} achieve the desired completion probability with a maximum percentage error of 1.2 for all of the clips. These results show that our stateless stochastic model predicts accurately the workloads of clips which are not included in its training set, while ignoring correlations leads to pessimistic completion. β_{ONL} is able to preserve the completion probability achieved by β_{SSM} not only for the two clips in the training set, but also for the other four clips used for evaluation.

Figure 7 presents the energy consumptions achieved by the considered energy management schemes. The energy consumption of β_{BND} decreases significantly as the completion probability constraint is lowered. This shows that introducing a small amount of tolerance to deadline misses results in a considerable potential for energy savings. β_{SSM} is able to exploit this tolerance and it reduces the energy consumption by up to a factor of 2.44 when compared to \mathbf{I} . For $P_{CON} < 0.95$, β_{SSM} consumes less energy than \mathbf{II} even though \mathbf{II} is a scheme that exploits runtime slacks unlike β_{SSM} . These results show that considering spatial correlations leads to more effective energy management. β_{ONL} , which is based on β_{SSM} with the additional capability to reclaim slacks detected at runtime, achieves higher energy savings than \mathbf{I} and \mathbf{II} . The energy consumption of β_{ONL} is 10% to 40% lower than \mathbf{II} . It is also remarkable that the energy consumption of β_{ONL} is within 10% of the theoretical bound β_{BND} .

B. Comparison of Stateless and Stateful Models

We compare β_{SSM} , which is based on a stateless model, with the following energy management policies based on stateful models:

- β_{PSO} is based on a PSO model with three states which are defined based on the workload of decoding a frame.
- β_{CSO}^T is based on a CSO model with three states which are defined based on frame type.

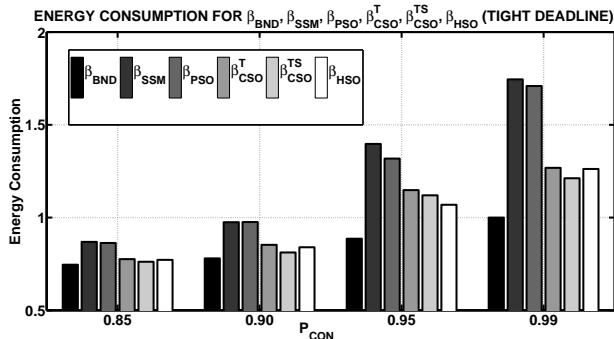


Fig. 8. Energy consumption comparison for the tight deadline case.

- β_{CSO}^{TS} is based on a CSO model with nine states, defined using both frame type and frame data size.
- β_{HSO} is based on an HSO model with nine states, using both frame type and frame load to define states.

The stochastic models and the associated energy management schemes are trained, optimized and simulated using the tight deadline. Because of space constraints, we do not report detailed results on the completion probabilities achieved by the schemes above. However, all of the stochastic models predict accurately the workloads for inputs that are not contained in the training set, and all of the energy management schemes attain the desired completion probability precisely on the average, never under- or over-achieving by more than 1%.

Figure 8 presents the energy consumptions of the policies listed above². The following conclusions can be derived based on the results obtained. All management policies based on stateful models, with the exception of β_{PSO} , perform significantly better than β_{SSM} , demonstrating the effectiveness of using stateful stochastic models for energy management. β_{PSO} 's performance is comparable to that of β_{SSM} . All stateful models except for β_{PSO} incorporate frame type into the state definition. The relatively poor performance of β_{PSO} compared with the other stateful policies suggests that the type of a frame to be processed is a better indicator of task workloads than just the workload of the previous frame. While it is intuitively clear that some observable information about the current period helps obtain better results than pure PSO models, when this is not possible, we conjecture that β_{PSO} can improve upon β_{SSM} for applications in which workload variation is relatively slow. Further evidence for this conjecture is provided by the good performance of β_{HSO} as explained below.

The policies β_{CSO}^{TS} and β_{HSO} both exhibit improved performance over β_{CSO}^T . This reconfirms within the context of our stateful models the observation (e.g., [16], [17]) that data size for the frame to be processed (β_{CSO}^{TS}) or the actual load of the previous frame (β_{HSO}) are good workload predictors for the current frame and shows that when this information is used in addition to frame type, it leads to more effective energy management. For β_{HSO} , this means that information about the previous frame is useful for refining the conditional workload PDF based on the type of the current frame. When we compare β_{CSO}^{TS} with β_{HSO} , we observe that β_{HSO} exhibits

²The normalized energy values in Figure 7 and 8 should not be compared directly since the global deadlines used are different.

TABLE II

ON-TIME FRAME DISPLAY PROBABILITY WITH A FRAME BUFFER

Clip No:	$P_{CON} = 0.85$		$P_{CON} = 0.90$	
	β_{SSM}	β_{ONL}	β_{SSM}	β_{ONL}
Buffer=2	0.970	0.968	0.997	0.995
Buffer=3	0.997	0.995	1.000	0.999

slightly better performance than β_{CSO}^{TS} at $P_{CON} = 0.95$, while β_{CSO}^{TS} is slightly better at all other values of P_{CON} . Thus, there is no clear winner between the actual workload of the previous frame and the (compressed) data size for the current frame to be decoded as a better indicator for the current frame workload. Finally, it is remarkable that the energy consumptions of all of the stateful policies (except for β_{PSO}) are within 10% of the theoretical bound for $P_{CON} < 0.95$, and even within 2% for β_{CSO}^{TS} at $P_{CON} = 0.85$.

C. Performance with a Frame Buffer

Our energy management schemes do not drop any frames even when the decoding of a frame takes longer than the deadline. In video decoding applications, the output of the decoder is usually buffered. As a result, a missed deadline for a frame can be compensated by earlier or later frames that finish before the deadline, and the performance quality perceived by the viewer (i.e. the fraction of frames displayed on time) is higher than the specified (on-time) completion probability. We present results in Table II for on-time frame display probability that we obtained by re-evaluating the performance of the decoder in the presence of a frame buffer. Even when $P_{CON} = 0.85$ (meaning that for 15% of the frames, the decoding time is larger than the specified deadline), the actual number of frames that are displayed late are less than 4% with a frame buffer with space for two frames, and less than 1% with a buffer of size three. This demonstrates that the P_{CON} values we experimented with are all acceptable viewing quality-energy trade-off points.

D. Effectiveness and Performance of SAN Heuristic

We compare the exact optimization formulation and solution method described in Section VIII-B to the one based on a heuristic SAN formalism described in Section VIII-C. Only the former technique exploits spatial correlations explicitly. Both methods have been applied to the EMS_{SSM} problem with the tight deadline, and the resulting policies are denoted by β_{SSM} and β_{SSM}^{SAN} respectively. Figure 9 shows the energy consumption of the two policies. The energy consumption of

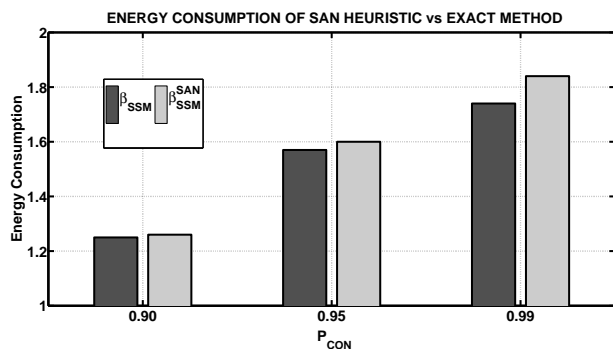


Fig. 9. Energy consumption of SAN Heuristic versus Exact Method

β_{SSM}^{SAN} is within 5% of β_{SSM} . Therefore, the SAN heuristic-based technique used in the optimization of policies based on stateful models works fairly well for the stochastic energy minimization problem. Figure 9 also shows that capturing spatial correlations explicitly and precisely becomes more important for the effectiveness of an optimization method as the desired completion probability increases.

X. CONCLUSIONS AND FUTURE WORK

We demonstrated that stochastic models and optimization formulations based on them can be used successfully for energy management on multi-core systems. Our work can be extended in the following directions: First, one can study a more flexible stateful model for applications for which a natural state definition is not immediate. In such a case, Hidden Markov Models can be used to extract the states in an application. Second, the explicit use of spatial correlations in the two-level optimization formulation based on the SAN heuristic can be investigated. Task scheduling, assignment and energy minimization problems should be studied jointly in a more comprehensive framework. As new parallelized applications are created for the emerging many-core systems, the modeling and optimization framework proposed in this paper can be developed further for practical use.

REFERENCES

- [1] S. Yaldiz, A. Demir, S. Tasiran, P. Jenne, and Y. Leblebici, "Characterizing and exploiting task-load variability and correlation for energy management in multi-core systems," in *ESTIMEDIA'05: Proc. IEEE 3rd Workshop on Embedded Systems for Real-Time Multimedia*, 2005, pp. 135–140.
- [2] S. Kim, S. Boyd, S. Yun, D. Patil, and M. Horowitz, "A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing, 2005," *Manuscript. Available from www.stanford.edu/boyd/heur_san_opt.html*.
- [3] Y. Liu, A. Maxiaguine, S. Chakraborty, and W. T. Ooi, "Processor frequency selection for soc platforms for multimedia applications," in *RTSS '04: Proc. 25th IEEE Intl. Real-Time Systems Symposium (RTSS'04)*, pp. 336–345.
- [4] Y. Liu, S. Chakraborty, and W. T. Ooi, "Approximate vccs: a new characterization of multimedia workloads for system-level mpoc design," in *DAC '05: Proc. 42nd ACM/IEEE conf. on Design automation*, 2005, pp. 248–253.
- [5] F. Gruian, "Hard real-time scheduling for low-energy using stochastic data and dvs processors," in *ISLPED '01: Proc. 2001 intl. symp. on Low power electronics and design*, pp. 46–51.
- [6] L.-F. Leung, C.-Y. Tsui, and X. S. Hu, "Exploiting dynamic workload variation in low energy preemptive task scheduling," in *DATE '05: Proc. conf. on Design, Automation and Test in Europe*, pp. 634–639.
- [7] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. A. Hashimi, "Quasi-static voltage scaling for energy minimization with time constraints," in *DATE '05: Proc. conf. on Design, Automation and Test in Europe*, Washington, DC, USA, 2005, pp. 514–519.
- [8] C. Scordino and E. Bini, "Optimal speed assignment for probabilistic execution times," in *2nd Workshop on Power-Aware Real-Time Computing (PARC05)*, NJ, Sept, 2005.
- [9] R. Xu, D. Mosse, and R. Melhem, "Minimizing expected energy in real-time embedded systems," in *EMSOFT '05: Proc. 5th ACM intl. conf. on Embedded software*, pp. 251–254.
- [10] S. Hong, S. Yoo, H. Jin, K.-M. Choi, J.-T. Kong, and S.-K. Eo, "Runtime distribution-aware dynamic voltage scaling," in *ICCAD '06: Proc. 2006 IEEE/ACM intl. conf. on Computer-aided design*, pp. 587–594.
- [11] Y. Zhang, X. S. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *DAC '02: Proc. 39th conf. on Design automation*, pp. 183–188.
- [12] S. Hua, G. Qu, and S. S. Bhattacharyya, "Energy-efficient multi-processor implementation of embedded software," *Lecture Notes in Computer Science*, vol. 2855, 2003.
- [13] A. C. Bavier, A. B. Montz, and L. L. Peterson, "Predicting mpeg execution times," in *SIGMETRICS '98: Proc. 1998 ACM SIGMETRICS joint intl. conf. on Measurement and modeling of computer systems*, 1998, pp. 131–140.
- [14] Y. Tan, P. Malani, Q. Qiu, and QingWu, "Workload prediction and dynamic voltage scaling for mpeg decoding," in *Design Automation, 2006. Asia and South Pacific Conf. on*, 2006.
- [15] D. Son, C. Yu, and H. Kim, "Dynamic voltage scaling on MPEG decoding," *Intl. Conf. of Parallel and Distributed Sys. (ICPADS)*, 2001.
- [16] K. Choi, K. Dantu, W.-C. Cheng, and M. Pedram, "Frame-based dynamic voltage and frequency scaling for a mpeg decoder," in *ICCAD '02: Proc. 2002 IEEE/ACM intl. conf. on Computer-aided design*, 2002, pp. 732–737.
- [17] L. Abeni, T. Cucinotta, G. Lipari, L. Marzario, and L. Palopoli, "Qos management through adaptive reservations," *Real-Time Syst.*, vol. 29, no. 2-3, pp. 131–155, 2005.
- [18] V. Venkatchalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 195–237, 2005.
- [19] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli, "Dynamic power management for portable systems," in *MobiCom '00: Proc. 6th intl. conf. on Mobile computing and networking*, 2000, pp. 11–19.
- [20] A. Nandi and R. Marculescu, "System-level power/performance analysis for embedded systems design," in *DAC '01: Proc. 38th ACM/IEEE conf. on Design automation*, 2001, pp. 599–604.
- [21] G. A. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli, "Policy optimization for dynamic power management," in *DAC '98: Proc. 35th annual conf. on Design automation*, 1998, pp. 182–187.
- [22] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time markov decision processes," in *DAC '99: Proc. 36th ACM/IEEE conf. on Design automation*, 1999, pp. 555–561.
- [23] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli, "Dynamic voltage scaling and power management for portable systems," in *DAC '01: Proc. 38th conf. on Design automation*, 2001, pp. 524–529.
- [24] L. Benini, A. Bogliolo, G. A. Paleologo, and G. D. Micheli, "Policy optimization for dynamic power management," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, June 1999.
- [25] L. O. Burchard and P. Altenbernd, "Estimating decoding times of mpeg-2 video streams," in *Proc. Intl. Conf. on Image Processing (ICIP '00)*, 2000.
- [26] A. Maxiaguine, S. Künzli, and L. Thiele, "Workload characterization model for tasks with variable execution demand," in *Design Automation and Test in Europe*, Feb. 2004.
- [27] R. P. Dick, "Multiobjective synthesis of low-power real-time distributed embedded systems," Ph.D. dissertation, November 2002.
- [28] Y. Jin, N. Satish, K. Ravindran, and K. Keutzer, "An automated exploration framework for fpga-based soft multiprocessor systems," in *CODES+ISSS '05: Proc. 3rd IEEE/ACM/IFIP intl. conf. on Hardware/software codesign and system synthesis*, 2005, pp. 273–278.
- [29] Intel PXA270 Processor, "Electrical, mechanical, and thermal specification," Available from <http://www.intel.com/design/pca/products/pxa27x/techdocs.htm>.
- [30] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B. Al-Hashimi, "Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems," in *Design Automation and Test in Europe Conf.*, February 2004.
- [31] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *ACM/IEEE Intl. Conf. on Computer-Aided Design*, November 2002.
- [32] S. M. Ross, "Introduction to probability models," January 2003.
- [33] E. Iwata and K. Olukotun, "Exploiting coarse-grain parallelism in the mpeg-2 algorithm," 1998.
- [34] PAPI, "Performance application programming interface," Available from <http://icl.cs.utk.edu/papi/>.
- [35] S. Boyd, S. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Revised for Optimization and Engineering*. Available from www.stanford.edu/boyd/gp-tutorial.html.
- [36] A. Mutapcic, K. Koh, S. Kim, and S. Boyd, "ggplab version 0.9, A Matlab Toolbox for Geometric Programming," January 2006.
- [37] fmincon, "Optimization toolbox, MATLAB, Mathworks, Inc."