

Characterizing and Exploiting Task-Load Variability and Correlation for Energy Management in Multi-Core Systems

Soner Yaldiz Alper Demir Serdar Tasiran

{syaldiz,aldemir,stasyran}@ku.edu.tr

Koç University
Istanbul, Turkey

Paolo Ienne Yusuf Leblebici

{paolo.ienne,yusuf.leblebici}@epfl.ch

Swiss Federal Institute of Technology (EPFL)
Lausanne, Switzerland

Abstract— We present a hybrid energy management technique that exploits the variability of and correlations among the computational loads of tasks in a real-time application with soft timing constraints. In our technique, task load variability and correlations are captured in stochastic models that incorporate certain salient features and essential characteristics of the application. We use the stochastic models in formulating and solving the energy management problem for applications with soft timing constraints running on multiprocessor systems with dynamic voltage scaling (DVS). We present a novel optimization formulation for minimizing average energy consumption while providing a probabilistic guarantee for satisfying timing constraints. We compare our stochastic models and energy management scheme with other models and schemes that do not capture/exploit either the variability of or the correlations among the computational loads of tasks.

I. INTRODUCTION

Minimization of energy consumption is a primary concern in the design of complex embedded systems, such as those implemented with multiprocessors and multicore systems. These systems can be found in a wide range of products and applications ranging from mobile consumer electronics to high-performance computing. With the advent of *Systems-on-Chip* (SOC), *Networks-on-Chip* (NOC), platform-based design, and programmable and reconfigurable architectures, almost all of the complex systems that will be conceived in the future will arguably have such multicore architecture. Hardware and operating-system (OS) support for implementing intelligent energy management schemes for these kinds of systems are currently being developed [1]. Energy consumption is increasingly an issue not only for battery operated devices; even if unlimited power is available, a large number of components tightly packed onto a chip and a large number of chips packed into a small volume pose cooling and reliability problems.

We address the energy management problem for real-time applications that have been decomposed into concurrent, interdependent tasks and that are intended to run on communicating multi-processor systems. We consider applications with repetitive behaviour—as in systems for image, speech and video processing—and with soft real-time constraints—that is, with some *tolerance to deadline misses*. We represent these applications with task graphs with precedence relationships and timing constraints [2]. We assume that applications will run on a multi-core platform with *Dynamic Voltage Scaling* (DVS) enabled *Processing Elements* (PE) and adaptive bandwidth *Communication Resources* (CR) where the system architecture has been already determined. Construction and exploitation of stochastic models—that capture correlations—for energy management are the major contributions of this work. In this paper, we do not concentrate on the details of the hardware platform, and we assume that the tasks composing the application have been already scheduled and assigned to processing elements.

This work has two key contributions:

- We demonstrate, using the important multimedia application of MPEG video decoding as an example, that there is significant variability in and *correlations* among the workloads of concurrent tasks that constitute the application. We illustrate on this example how energy management ignorant of task load variability and/or

correlation can be significantly sub-optimal or not properly fulfill timing constraints. We introduce *stochastic application models* that capture workload variations and correlations and show that these models predict well the workloads of tasks in any MPEG execution.

- We present a novel optimization formulation which results in processor *Voltage Settings* (VS's) that minimize the *average* energy consumption of the system while providing a probabilistic guarantee for satisfying timing constraints. This formulation *fully exploits* time-variability and correlation information captured by the stochastic application models. The optimization problem is solved off-line and its results stored in a look-up table. The optimized energy management policy is realized during run-time performing only look-ups of the stored off-line optimization results. We also describe a simple on-line heuristic addition to our standard optimized energy management scheme that achieves further energy savings by performing simple on-line computations and table look-ups.

The rest of the paper is organised as follows: In the rest of this section we present a simple, contrived example and conclude with a review of previous work. In Section II, we present our stochastic application models and demonstrate that significant variabilities and correlations exist in actual application runs. We then, in Section III, present our novel optimization formulation, its off-line solution, and a simple on-line heuristic that achieves additional energy savings. Experimental results on a multimedia application and comparison with other models and schemes are presented in Section IV. Conclusions and future directions are presented in Section V.

A. Motivating example

When there are significant correlations among the *Computational Demands* (CD's) of different tasks composing an application, energy management based on the assumption of statistical independence may be (i) incorrect—i.e., may not achieve the required probability of satisfaction of the timing constraints—or (ii) sub-optimal. To illustrate this point, consider two tasks u_1 and u_2 running in tandem on a processor. Suppose that each can have a computational load of either l or h ($l < h$) with a probability of 0.5 each. Consider the following three scenarios:

(a) u_1 and u_2 are independent

(b) u_1 and u_2 are fully positively correlated, i.e., either both of them have a workload of h , or both have a workload of l .

(c) u_1 and u_2 are fully negatively correlated, i.e., when one has a workload of l , the other one has a workload of h .

Scenario	P($2l$)	P($l+h$)	P($2h$)
(a) Indep.	0.25	0.50	0.25
(b) Full pos.	0.50	0	0.50
(c) Full neg.	0	1	0

To see how ignoring correlations may lead to miscomputation of the timing deadline satisfaction probability, consider the following situation. Suppose that (b) is the case and assume that u_1 and u_2 are required to meet a deadline with a minimum probability of 0.75. Consider an energy management policy that assumes that u_1 and u_2 are independent, and slows down the processor to a constant speed so that the deadline is met exactly when the computational

demand is $l+h$. At this speed the deadline is made if the demand is $2l$ or $l+h$ but is missed when it is $2h$. Since tasks are correlated, this policy meets the deadline 50% of the time instead of the required 75%.

To see how ignoring correlations may lead to suboptimal energy savings, suppose now that (c) is the case and assume that u_1 and u_2 are required to meet a deadline with 100% probability. If an energy management policy operates assuming that the tasks are independent, at least 25% of the time it will decide to run the processor fast enough to finish a computational load of $2h$ within the deadline. However, since the true probability of this worst case is 0, this policy is sub-optimal.

We would also like to point out that the total worst-case execution time in cases (a) and (b) is equal to $2h$, whereas it is equal to $l+h$ in case (c). In general, when tasks are considered independent, the total worst-case execution time will always be equal to the sum of the individual worst-case execution times of the tasks. However, with correlated tasks, the total worst-case execution time may be (considerably) smaller than the sum of the individual worst-cases. This key observation tells us that one could exploit correlation information (but not load variability) for energy savings, even for applications with hard real-time deadlines.

B. State of the art

Most of the previous work in this area concentrates on real-time applications with *hard* timing constraints ([3], [4], [5], among others). Systems with hard real-time deadlines are forced to operate under the assumption of worst-case execution times. However, for a large class of applications running on power-sensitive portable systems, the constraints on real-time behavior are not as strict. For instance, for streaming multi-media applications, some degradation in quality and performance is acceptable if significant energy savings are achieved in return. Such savings are possible when worst-case computational demand occurs rarely and is significantly different from the average case [6], [7], [8], as we demonstrate in this paper for a streaming multimedia application. Yet, little work has been done on energy minimization that can exploit the time-variability in the computational demands of the tasks that constitute an application.

Two notable exceptions are [9] and [10]. In [9], authors describe a heuristic scheme that provides probabilistic completion ratio guarantees while trying to minimize energy consumption. They employ probability distributions for execution times of tasks, but the execution times of different tasks are assumed to be independent of each other. In Section IV, we will compare the results we obtain with our energy management scheme with the one proposed in [9]. In a similar way to our work, in [10] authors present and use for power management a stochastic model for prediction of execution times for streaming multimedia applications. They fit the parameters of their *analytical* model to experimental frame inter-arrival time data but do not model the breakdown of the load to concurrent tasks running on a multi-core system.

We will demonstrate with our results in Section II and Section IV that one can exploit correlation information for energy savings even for applications with hard real-time deadlines operating under the assumption of worst-case execution times. In Section IV, we also compare our results with the ones obtained with the energy management scheme proposed in [3] that is based on worst-case execution times assuming independence.

II. STOCHASTIC MODELING OF APPLICATIONS

A. Preliminaries

This section is concerned with the stochastic modeling of the *computational demand* (CD) of a task, a term we use to refer to a normalized measure of the workload of a task that is independent of processor speed. For this purpose, we use the number of clock cycles required to execute the task on a given processor. For many

applications, the CD and its breakdown to individual tasks exhibit significant variability over time [6], [7], [8] – a fact that can be exploited to reduce energy consumption when probabilistic timing constraints apply [11], [9].

We represent the variability of task CD's and their correlation by a joint probability distribution function *dist*. For the representation and manipulation of *dist* to be manageable, we quantize the CD for each task u and represent it using a relatively small number of discrete values $Q_u = \{0, 1, 2, \dots, q_u\}$. We have used values of q_u ranging from 16 to 256. *dist* is a probability distribution over a discrete space, i.e., $dist(L_1, \dots, L_n) \in [0, 1]$ where each $L_i \in \{0, 1, 2, \dots, q_i\}$ is the quantized CD for task i . *dist* is obtained from experimental data. We run the application on a set of inputs and record at each period the number of cycles spent on executing each task. To minimize the effects of context switches, cache misses, and branch mispredictions on the measured number of execution cycles, we run the application several times on the same input and average the results. We divide the range between zero and the maximum CD for task u uniformly into q_u intervals. The demand of each task in each period is converted to a discrete quantity using this quantization. During each run, at each period p , the quantized tuple of CD's $(L_1^{(p)}, L_2^{(p)}, \dots, L_N^{(p)})$ constitutes one “observation” in the space of observations $Q_1 \times Q_2 \times \dots \times Q_N$. After running the application on a set of inputs and obtaining $|O|$ observations, $dist(L_1^{(p)}, L_2^{(p)}, \dots, L_N^{(p)})$ is approximated by dividing by $|O|$ the number of periods that $(L_1^{(p)}, L_2^{(p)}, \dots, L_N^{(p)})$ is observed. The averaging over different inputs and time is implicit here.

B. Stochastic modeling of MPEG2 decoding

As our case study, we focused on the widely-used and computationally intensive MPEG2 video decoding application. We used the slice-based task decomposition and assignment to four processors that was found to yield the best performance in [12] with a frame rate of 24 per second. The implementation of [12] divides the computation associated with each slice into two separate tasks: variable-length decoding (VLD) and motion compensation (MC). The task graph representing this decomposition and task assignment is given in Figure 1. Numbers within ovals correspond to

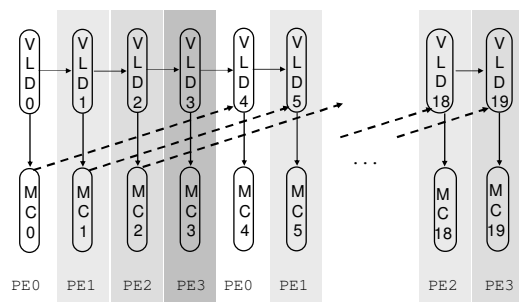


Fig. 1. Task graph for MPEG2

the slice number and dashed arrows represent precedence relations that are due to tasks being assigned to the same processor.

We collected experimental data using a set of ten MPEG2 movie segments from feature movies. Each segment was of approximately ten minutes long, i.e., consisted of roughly 14.5 thousand frames with 19 slices each. Figure 2 presents CD histograms for a slice chosen from the middle of the screen. The bottom pair of histograms correspond to data collected from a single segment, the top pair is obtained using the entire collection of movies. The bottom histograms are qualitatively representative of all task demand histograms obtained from MPEG2. Tasks' CD distributions, while concentrated around their means, have long tails. The shapes of the distributions for all MC tasks appear similar. The same is true for

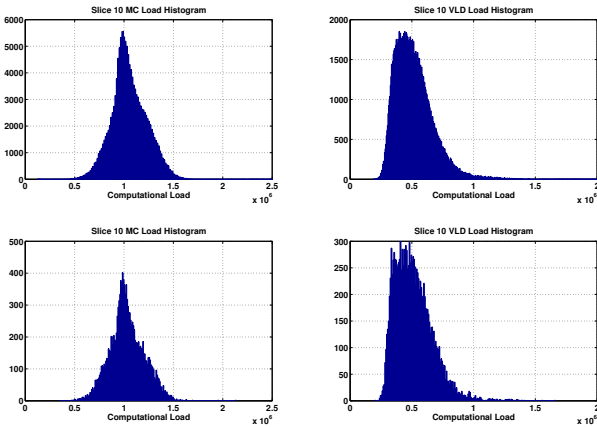


Fig. 2. Task workload histograms

VLD tasks, while the shapes for MC tasks and VLD tasks appear to be qualitatively different from each other.

Figure 3 presents the correlation coefficients of the workload for tasks of the center slice (VLD10 and MC10) with the workloads of other VLD and MC tasks as a function of the distance of the slices from the center slice. The dotted lines show data obtained from

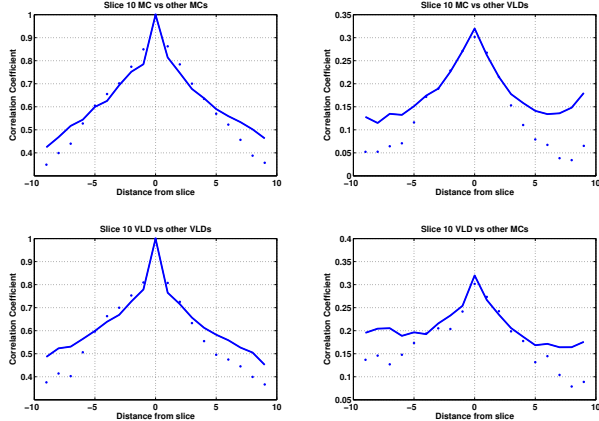


Fig. 3. Task workload correlations

particular movie segment, whereas the solid lines represent aggregate statistics. Correlation graphs obtained from other input files are very similar. Two facts emerge distinctly from this data: (i) the tasks of slices close to each other are highly correlated, and (ii) the MC and VLD tasks of a particular slice are highly correlated with each other.

If all processors in Figure 1 are run at the same speed, the data collected from the MPEG2 runs indicates that the critical path through the task graph is VLD0 \rightarrow VLD1 \rightarrow VLD2 \rightarrow ... \rightarrow VLD19 \rightarrow MC19. The sum of workloads of tasks along this path, in the worst-case in all of our MPEG2 executions was around 27.7 million cycles. If we were to assume that the tasks' workloads were independent, the worst-case demand along this path would have been the sum of worst-case loads for each task, amounting to around 63.8 million cycles. This significant difference demonstrates that ignoring task correlations for MPEG2 decoding would result in far from optimal energy management decisions, even when operating under the assumption of worst-case execution times.

III. ENERGY MINIMIZATION WITH STOCHASTIC MODELS

A. Preliminaries and definitions

Our problem formulation starts from an application that has been decomposed into a set of concurrent tasks as described in the previ-

ous section for MPEG2 decoding. The set of tasks and their precedence dependencies are represented using a directed acyclic graph called the *task graph* [2], [3]. Each node u in the graph represents a task and each edge (u, v) indicates that task v can only start executing after u finishes. We also take as given (i) the assignment of tasks to processing elements (PEs) and (ii) the order in which the tasks on a given PE are executed, i.e., the task scheduling. Edges are added to the task graph to represent precedence relationships due to the scheduling of tasks that run on the same PE.

Each task can have a deadline constraint D_u on its finish time F_u , and an additional global deadline constraint D_{DAG} for the whole task graph that restricts all tasks' finish time. Tasks in the task graph may represent communication events as well as computational tasks. Likewise, the "PE's" are not necessarily processors but may represent communication resources such as a bus, on-chip network or ethernet connection. For simplicity of exposition, we assume that the power supply voltages for the DVS enabled PE's are continuously adjustable.

Our first goal is to determine, by an off-line computation, fixed voltages for each task that minimize the *average* energy consumption while providing a probabilistic guarantee for the satisfaction of the timing constraints.

B. Energy consumption model and averaging

The goal of this section is to describe how average energy consumption is computed using the stochastic model for the application. Let L_u be a stochastic variable representing the *computational demand* associated with a task during one period. The stochastic application model discussed in Section II provides a joint probability density *dist* for the vector of computational demands \mathbf{L}_u of all tasks.

We use L_u and the parameters of the processor to compute average energies. The cycle-time (CT) for a PE while it is running task u is a function of V_{ddu} , the voltage setting for task u .

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

where k_{ct} and a_{ct} are technology related constants and V_{th} is the threshold voltage [3]. The execution time t_u for a task u is then given by

$$t_u = L_u CT = L_u k_{ct} \frac{V_{ddu}}{(V_{ddu} - V_{th})^{a_{ct}}} \quad (2)$$

The dynamic energy consumed by a PE to execute a task u is given by

$$E_u^{dyn} = L_u C_u V_{ddu}^2 \quad (3)$$

where the "effective switching capacitance" C_u is a measure of the utilization of the PE by the task u [3]. The leakage energy and the voltage transition overhead energy and time penalty can be easily also included in the formulation, however we do not consider them here for the sake of simplicity [5], [13].

The total energy consumed in all PEs during one period of the application can then be computed as follows

$$E^{tot} = \sum_{u=1}^N E_u^{dyn} \quad (4)$$

Since L_u is a randomly varying quantity, the total energy consumed in the PE's during one period of the application is also a random quantity. Our energy management scheme minimizes the *average* (expected) energy consumption

$$\mathbf{E} [E^{tot}] = \sum_{u=1}^N \mathbf{E} [E_u^{dyn}] \quad (5)$$

where $\mathbb{E}[\cdot]$ represents expectation according to the joint probability distribution $dist$ of the computational demands L_u . In (5), we have

$$\mathbb{E}[E_u^{dyn}] = \mathbb{E}[L_u] C_u V_{ddu}^2 \quad (6)$$

The average energy consumption in (5) is the objective function that our formulation aims to minimize. Given a voltage setting for all the tasks, the average energy in (5) can easily be computed using the average computational demands $\mathbb{E}[L_u]$ for the tasks. $\mathbb{E}[L_u]$ can be computed using the marginal probability densities for L_u which can be deduced from $dist$.

C. Formulation of probabilistic performance guarantee

We achieve minimized energy consumption while providing probabilistic performance guarantees for the satisfaction of timing constraints by formulating and solving a constrained optimization problem with the objective function in (5) and a constraint expressed as

$$P_{SAT} = \mathbb{P}(F_u \leq D_u, u = 1, 2, \dots, N) \geq P_{CON} \quad (7)$$

where F_u and D_u are the finish time and deadline for task u , and $\mathbb{P}(\cdot)$ is the probability measure on the joint probability space defined by $dist$. In words, the probability that all timing constraints (i.e., task deadline constraints and the global deadline constraint) are met, P_{SAT} , should be at least P_{CON} . Here $0 \leq P_{CON} \leq 1$ is a measure of the tolerance of the application to deadline misses.

P_{SAT} is computed by an explicit enumeration of the points in the joint probability space. The number of points in this joint probability space with nonzero probability is bounded by the number of data points in the application run trace that was used to construct $dist$. This number is much smaller than the size of the entire probability space given by $Q_1 \times Q_2 \times \dots \times Q_N$. We exploit this by using a sparse representation for $dist$. The computational complexity of the evaluation of (7) is $O((N+G)K)$ where N is the number of tasks, G is the number of edges in the task graph, and K is the number of points in the probability space with nonzero $dist$.

For all of the examples we present in this paper, the evaluation of (7) was fast enough for the optimization to be performed in at most tens of minutes of CPU time. We can afford much more CPU time to solve the optimization problem, since this computation is done off-line. Still, to be able to handle very large task graphs and/or very long data traces efficiently, in our future research, we plan to investigate the use of efficient, implicit graph representations [14] of the joint probability space and density that are able to exploit special properties such as conditional independence as well as sparsity.

D. Optimization problem formulation for computing optimal fixed voltage settings and its off-line solution

We formulate the energy minimization problem as a nonlinear constrained optimization problem as follows

$$\begin{aligned} \min \quad & \mathbb{E}[E^{tot}] = \sum_{u=1}^N \mathbb{E}[E_u^{dyn}] \\ \text{subject to} \quad & \mathbb{P}(F_u \leq D_u, u = 1, 2, \dots, N) \geq P_{CON} \end{aligned} \quad (8)$$

The variables in the optimization formulation above are the V_{ddu} 's for the tasks. We also impose upper and lower bounds on the voltage settings V_{ddu} for the tasks in solving (8). In the MPEG2 decoding task graph, there are 38 tasks, 2 tasks for each of the 19 slices. Hence, there are 38 variables.

We currently use an SQP (sequential quadratic programming) optimizer to solve (8), which works much more effectively when

analytical gradients of the objective function and the constraints are supplied. The analytical gradient of the objective function in (5) can easily be computed. Unfortunately, the gradient for the nonlinear constraint in (7) is not analytically computable. We compute it approximately using finite differences. For most of the problems we tried, the aforementioned SQP implementation quickly converges to the optimal solution. The optimization problem described above is solved off-line, not during run-time.

The optimal fixed voltage settings for the task set produced by the off-line solution of the optimization problem above are stored in a table for run-time look-up. For the rest of our treatment, we will refer to the energy management scheme that uses these fixed voltage settings as OFLN. This name was chosen to emphasize the fact that OFLN does not perform any on-line computations. During run-time, the voltage settings of each PE change from task to task but the voltage for each task is fixed for all periods and does not need to be computed on-line.

E. Run-time adjustments to voltage settings

While the V_{ddu} 's obtained as the off-line solution of the optimization problem described above are the optimal *fixed* voltages, it may be possible to achieve further energy savings by allowing run-time adjustments to task voltages. Intuitively, the goal is to lower the task voltage below V_{ddu} for periods during which we detect relatively low computational demand while preserving our probabilistic performance guarantee. If in a particular period, right before a task u is executed, we realize that task u would still have slack even if worst-case total demand is experienced for the remaining tasks in the period, we take advantage of this opportunity by lowering the voltage for u to a value that is smaller than V_{ddu} read from the table.

Our optimization formulation selects a subset $\mathbf{Q}_{SUB} \subseteq Q_1 \times \dots \times Q_N$ of the probability space for which setting the task voltages to V_{ddu} guarantees that the deadlines will be met. For each task u , let us define the following quantities:

- $t_{wc}(u)$: The time it takes for the PE that runs u to execute the worst-case demand for u (within \mathbf{Q}_{SUB}) at the voltage setting V_{ddu} .
- $t_{latest}(u)$: Assuming that all tasks v , starting with u until the end of the period are run at a voltage setting of V_{ddv} , the latest time that u has to start so that for all points in \mathbf{Q}_{SUB} , the deadlines are still met.

$t_{wc}(u)$ and $t_{latest}(u)$ are computed off-line for each task u using $dist$ and a reverse-topological traversal of the task graph. This computation takes linear time in the size of the task graph and the number of points in \mathbf{Q}_{SUB} . The off-line computed $t_{wc}(u)$ and $t_{latest}(u)$ are also stored in the same table containing the fixed optimal voltage setting V_{ddu} for each task.

Suppose that during a particular period, right before task u starts execution, t units of time have elapsed since the beginning of the period. We apply the following heuristic for updating the voltage setting of task u . If $\Delta t = t_{latest}(u) - t > 0$, lower the voltage for task u such that the worst case load for u at the new voltage V_{ddu}' takes $t_{wc}(u) + \Delta t$. Observe that the run-time computations required for the online update is quite cheap: A few table look-ups, a few arithmetic operations and a comparison.

We will refer to the version of our energy management scheme that performs run-time adjustments to task voltage settings as ONLN for the rest of the paper.

IV. EXPERIMENTS AND DISCUSSION

This section reports experimental results using our energy management schemes and comparisons with other approaches. The following is a list of the approaches compared:

- I: The energy management scheme in [3]. This scheme minimizes energy consumption under the assumption of worst-case compu-

TABLE I
COMPARISON OF ENERGY MANAGEMENT SCHEMES I, II, OFLN AND ONLN (RELAXED DEADLINE)

		$P_{CON} = 0.90$				$P_{CON} = 0.95$				$P_{CON} = 0.99$			
Movie No:		I	II	OFLN	ONLN	I	II	OFLN	ONLN	I	II	OFLN	ONLN
1	E	857.3	153.2	100.0	98.1	830.5	146.2	100.0	96.7	762.4	129.0	100.0	91.4
	Pr	1.000	1.000	0.905	0.905	1.000	1.000	0.947	0.947	1.000	1.000	0.989	0.989
2	E	859.8	156.3	100.0	98.2	837.4	151.5	100.0	97.3	774.2	135.2	100.0	92.8
	Pr	1.000	1.000	0.775	0.775	1.000	1.000	0.879	0.879	1.000	1.000	0.985	0.985
3*	E	864.4	154.5	100.0	98.1	838.8	148.1	100.0	96.9	768.7	129.8	100.0	91.5
	Pr	1.000	1.000	0.894	0.894	1.000	1.000	0.956	0.956	1.000	1.000	0.991	0.991
4	E	862.1	152.6	100.0	98.0	834.7	145.4	100.0	96.6	763.7	127.4	100.0	90.9
	Pr	1.000	1.000	0.931	0.931	1.000	1.000	0.970	0.970	1.000	1.000	0.991	0.991
5	E	867.1	152.6	100.0	98.2	837.9	145.0	100.0	96.8	765.6	126.8	100.0	91.1
	Pr	1.000	1.000	0.953	0.953	1.000	1.000	0.978	0.978	1.000	1.000	0.992	0.992
6*	E	851.9	152.9	100.0	98.0	824.5	145.5	100.0	96.6	756.2	128.0	100.0	91.0
	Pr	1.000	1.000	0.923	0.923	1.000	1.000	0.954	0.954	1.000	1.000	0.988	0.988
7	E	861.1	153.2	100.0	97.9	834.4	146.4	100.0	96.5	764.3	128.5	100.0	91.0
	Pr	1.000	1.000	0.919	0.919	1.000	1.000	0.964	0.964	1.000	1.000	0.993	0.993
8	E	864.2	151.4	100.0	98.0	835.2	143.9	100.0	96.5	763.2	126.0	100.0	90.5
	Pr	1.000	1.000	0.960	0.960	1.000	1.000	0.983	0.983	1.000	1.000	0.997	0.997
9	E	856.5	158.1	100.0	98.3	830.7	150.6	100.0	97.1	762.6	132.0	100.0	91.9
	Pr	1.000	1.000	0.894	0.894	1.000	1.000	0.939	0.939	1.000	1.000	0.978	0.978
10	E	853.5	155.8	100.0	98.3	828.5	149.2	100.0	97.1	761.0	131.2	100.0	91.9
	Pr	1.000	1.000	0.871	0.871	1.000	1.000	0.941	0.941	1.000	1.000	0.994	0.994
Average Energy		859.8	154.1	100.0	98.1	833.3	147.2	100.0	96.8	764.2	129.4	100.0	91.4

tational loads for tasks. The overall worst-case load for a task graph is computed by summing the individual worst-case loads for the tasks assuming independence. Hard timing constraints are assumed.

II: The energy management scheme in [9]. Load variability information is used in the form of marginal load distributions for tasks. Task loads are assumed to be independent of each other. Energy minimization is done using a two-stage approach, with off-line heuristics and run-time voltage selection. This approach can handle soft timing constraints and provides probabilistic completion ratio guarantees under the assumption that task loads are independent.

OFLN: Our energy management scheme that uses optimal fixed voltage settings and performs only table look-ups during run-time, with no on-line computations or adjustments to voltage settings. Please see Section III-D for a detailed description.

ONLN: Our energy management scheme that augments OFLN with run-time adjustments for further energy savings at the very small run-time expense of additional table look-ups, a few arithmetic operations and a comparison. Please see Section III-E for a detailed description.

GOD: This is a non-causal, hypothetical energy management scheme that is not implementable in reality. This scheme has *a priori* knowledge of all task loads during an execution, and, for selecting voltage settings, it has unlimited run-time computational resources at no energy cost. Because task loads are different for each period, before each individual period (each MPEG2 frame) is executed, it solves a separate optimization problem on-line to compute the optimal voltage settings for that period.

All of the schemes above were implemented and simulated in MATLAB.

Table I presents the results of a set of experiments that contrasts the essential features of the first four techniques listed above. Several technicalities need to be explained before the results can be interpreted properly.

The task decomposition, assignment and scheduling for the tasks were done as described in Section II. All probability distributions and probabilistic parameters for all techniques were derived from a set of eight 10-minute segments from a feature movie encoded in MPEG2. We call this set of movie segments the “training set”. Movies marked by an asterisk (movies 3 and 6) were intentionally left out of the training set in order to test how well the stochastic model for MPEG2 can predict the characteristics of an

unknown movie.

In each set of columns marked by $P_{CON} = 0.90$, $P_{CON} = 0.95$, and $P_{CON} = 0.99$, all techniques that can make use of soft timing constraints were given the *target* constraint satisfaction probability of 0.90, 0.95 and 0.99, respectively. $P_{CON} = 0.99$ models the “hard deadline” case, while $P_{CON} = 0.9$ and $P_{CON} = 0.95$ are two different choices for deadline miss tolerance.

For all techniques, the *actual* percentage of frames completed successfully by the deadline differs from the target satisfaction probability and is listed for each movie segment in a row marked “ P_{ACH} ”. For **II**, the difference between the target and accomplished completion rates is very pronounced and is because (i) **II** uses an overly-conservative method to guarantee the satisfaction probability, (ii) **II** ignores task correlations, and (iii) a *dist* obtained from the training set may not model the load distribution of a particular movie segment perfectly accurately. Factor (iii) applies to **ONLN** and **OFLN** as well, but we have found that it causes only a minor difference between the target and accomplished completion ratios. This indicates that an application-specific (but movie independent) stochastic model captures quite accurately the load demand of a new, unknown movie. Movie 2 is an exception due to the fact that its workload statistics differ from the training set’s. This exception has inspired a refinement/generalization of the stochastic application model to incorporate an “application state”, which is part of our current/future work.

The energy consumption that each technique achieves for each movie segment is listed in rows marked “Engy”. For ease of comparison, for each movie segment, the energy quantities are normalized to the energy consumption achieved by **OFLN**. To make the comparison fair, and not biased by the facts that **II** overshoots the target completion probability and **I** does not make use of probability distributions, the total energy is computed using *the same subset of frames* for each technique – those that **OFLN** finishes by the deadline. Even after factoring out the overachievement of completion ratios by other techniques, **OFLN** and **ONLN** achieve significant energy savings over **II** and perform almost an order of magnitude better than **I**. Average results over all movie segments are presented in the last row of Table I.

It is noteworthy that **ONLN** and **OFLN** consume a lot less energy than **II** and **I** even for $P_{CON} = 0.99$, the “hard deadline” case. This stems from the fact that the correlated worst-case computational demand of the tasks in this application (more precisely, those on the critical paths of the task graph) differs significantly from the

TABLE II
COMPARISON OF OFLN AND ONLN (TIGHT DEADLINE)

		$P_{CON} = 0.90$		$P_{CON} = 0.95$		$P_{CON} = 0.99$	
Movie No:		OFLN	ONLN	OFLN	ONLN	OFLN	ONLN
1	E	100.0	95.6	100.0	90.9	100.0	70.3
	Pr	0.896	0.896	0.947	0.947	0.989	0.989
2	E	100.0	96.1	100.0	92.7	100.0	75.1
	Pr	0.756	0.756	0.876	0.876	0.985	0.985
3*	E	100.0	95.9	100.0	91.3	100.0	70.5
	Pr	0.905	0.905	0.959	0.959	0.991	0.991
4	E	100.0	95.4	100.0	90.3	100.0	68.3
	Pr	0.936	0.936	0.971	0.971	0.991	0.991
5	E	100.0	95.9	100.0	90.5	100.0	68.4
	Pr	0.957	0.957	0.979	0.979	0.992	0.992
6*	E	100.0	95.9	100.0	90.8	100.0	69.5
	Pr	0.925	0.925	0.956	0.956	0.988	0.988
7	E	100.0	95.2	100.0	90.1	100.0	68.9
	Pr	0.920	0.920	0.964	0.964	0.993	0.993
8	E	100.0	95.5	100.0	89.8	100.0	66.9
	Pr	0.965	0.965	0.984	0.984	0.997	0.997
9	E	100.0	96.9	100.0	92.3	100.0	72.7
	Pr	0.903	0.903	0.943	0.943	0.978	0.978
10	E	100.0	97.1	100.0	92.7	100.0	72.0
	Pr	0.867	0.867	0.936	0.936	0.994	0.994
Average Energy		100.0	95.9	100.0	91.1	100.0	70.3

demand computed by ignoring their correlations. OFLN and ONLN are able to make use of the more accurate, correlated worst-case load.

To contrast the performance of energy management techniques when the (soft) deadlines that are relatively “relaxed” vs. “tight”, Table II presents the results of the same experiments in Table I ran using a tighter deadline. We chose deadlines that intuitively reflect the two following scenarios on a multi-processor system running a number of applications concurrently: (i) all tasks can be completed at full V_{dd} even if independent worst-case demand was experienced by each MPEG2 task, and (ii) the multi-processor system is experiencing high load, therefore, the resources that can be allotted to MPEG2 tasks is only 80% of the (correlated) worst-case time at full V_{dd} . The results in Table II are run with a deadline that corresponds to (ii).

There are two noteworthy differences between the results in Table I and Table II. First, no results with **II** are reported in Table II. This is because **II** is not able to handle the tighter deadline although it is achievable by OFLN and ONLN. **II** declares the required deadline as infeasible because of its assumption that task loads are independent. Second, the on-line heuristic ONLN improves upon OFLN quite a bit when the deadline is tight, while it only caused a modest improvement upon OFLN when the deadline was relaxed.

Table III compares the results achieved by different techniques for Movie 4 using a *dist* obtained from Movie 4 itself. The goal of this experiment is to remove the inaccuracy in stochastic modeling and to determine how close to GOD different techniques come if they are given accurate, exact statistics. Results show that OFLN performs better than **II**, and ONLN improves upon OFLN. There is also indication that there is room for improvement using a more refined stochastic model, which is part of our current work.

The following conclusions can be derived from the experimental results presented in this section and the discussions above:

- The stochastic model we propose captures the essential characteristics of applications, and is able to predict well actual task workload distributions and correlations for other runs not contained in the training data set.
- The stochastic model of the application serves successfully as the basis of energy optimization and timing constraint fulfillment prediction, and our optimization scheme can properly exploit the flexibility for a given value of required completion ratio.
- Energy management schemes taking both load variability and correlations into account yield larger energy savings compared with approaches operating under the worst-case execution time assumptions or other schemes that assume independent task load dis-

TABLE III
COMPARISON OF **II**, OFLN, ONLN AND GOD

Prob	II	OFLN	ONLN	GOD
0.99	FAIL	100	65.81	52.00
0.95	100.65	100	86.40	71.84
0.90	108.23	100	92.32	76.60

tributions.

- OFLN, our energy management scheme using fixed optimal voltage settings computed off-line as the solution of an optimization problem that takes correlations into account, without any run-time adjustments and on-line computations, already achieves larger energy savings compared with other approaches that ignore correlations, even when these approaches perform run-time adjustments and on-line computations. ONLN, with the addition of very low-cost run-time adjustments to our energy management scheme, achieves further, sometimes significant energy savings over schemes that ignore correlations.

V. CONCLUSIONS AND FUTURE WORK

We presented a novel off-line optimization formulation, associated stochastic models, and an on-line heuristic management scheme for the energy management of real-time applications on multiprocessor systems, with soft timing constraints. We demonstrated on a streaming multimedia application that significant variabilities and correlations exist for the computational demands of concurrent tasks, and must be taken into account. Our results indicate that there is significant potential in intelligent energy management that can exploit application characteristics as captured by our stochastic models.

In our future work, we plan to explore generalizations and additions to the stochastic application models, energy management scheme and the optimization formulation described in this work. In particular, we will study formulations with stochastic models that have state in order to capture correlations over time and the input dependency of task workloads.

REFERENCES

- [1] K. Flautner, D. Flynn, and M. Rives. A combined hardware-software approach for low-power SoCs: Applying adaptive voltage scaling and intelligent energy management software. In *DesignCon*, 2003.
- [2] Robert P. Dick. Multiobjective synthesis of low-power real-time distributed embedded systems, Ph.D. Dissertation, Princeton University November 2002.
- [3] Y. Zhang, X. Sharon Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *Proceedings of the 39th Design Automation Conference*, New Orleans, La., June 2002.
- [4] G. Varatkar and R. Marculescu. Communication-aware task scheduling and voltage selection for total systems energy minimization. In *ACM/IEEE International Conference on Computer-Aided Design*, November 2003.
- [5] A. Andrei, M. Schmitz, P. Eles, Z. Peng, and B.M. Al-Hashimi. Overhead-conscious voltage selection for dynamic and leakage energy reduction of time-constrained systems. In *Design Automation and Test in Europe Conference*, February 2004.
- [6] Y-T. S. Li and S. Malik. Performance analysis of embedded software using implicit path enumeration. In *Proceedings of the 32nd Design Automation Conference*, June 1995.
- [7] Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th Design Automation Conference*, New Orleans, La., June 1999.
- [8] A. Maxiaguine, S. Künzli, and L. Thiele. Workload characterization model for tasks with variable execution demand. In *Design Automation and Test in Europe*, February 2004.
- [9] S. Hua, G. Qu, and S. Bhattacharyya. Energy efficient multi-processor implementation of embedded software. In *International Workshop on Embedded Software*, October 2003.
- [10] Tajana Simunic, Luca Benini, Andrea Acquaviva, Peter Glynn, and Giovanni De Micheli. Dynamic voltage scaling and power management for portable systems. In *Proceedings of the 38th Design Automation Conference*, Las Vegas, Nev., June 2001.
- [11] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.W.-S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In *IEEE Real Time Technology and Applications Symposium*, 1995.
- [12] Eiji Iwata and Kunle Olukotun. Exploiting coarse-grain parallelism in the mpeg-2 algorithm. Stanford University Computer Systems Lab Technical Report CSL-TR-98-771 September 1998.
- [13] 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 International Conference on Computer-Aided Design*, November 2002.
- [14] M. Bozga and O. Maler. On the representation of probabilities over structured domains. In *Proc. CAV'99*. Springer, 1999.