An Analysis of System Level Power Management Algorithms and Their Effects on Latency

Dinesh Ramanathan, Sandra Irani, and Rajesh K. Gupta

***Abstract*—The problem of power management for an embedded system is to reduce system level power dissipation by shutting off parts of the system when they are not being used and turning them back on when requests have to be serviced. Algorithms for this problem are *online* in nature; the algorithm must operate only with access to data that it has seen so far and without access to the complete data set or its characteristics. In this paper, we present online algorithms to manage power for embedded systems and discuss their effects on system latency.**

We introduce competitive analysis as a formal framework for the evaluation of various power management algorithms. Competitive analysis does not depend on the distribution of interarrival times of requests. In this context, we present a nonadaptive online algorithm, analyze its behavior, and show that it is optimal. In this paper, we also present a lower bound on the competitiveness of any adaptive algorithm. We show that no adaptive online algorithm can dissipate less than about 1.6 times the power dissipated by the optimal offline algorithm in the worst case. We also show that in order for any online algorithm to achieve this lower bound, it may have to maintain a complete history of the interarrival times of the requests in the input sequence. Since this is not practical, we present a simple algorithm that uses only the last interarrival time to predict the arrival of the next request. We show that this algorithm performs as well as previously proposed heuristics for the problem; however, we can bound its worst case performance. In all our formal analysis, we do not model the service time for a request, i.e., we assume that requests are services instantaneously.

To test the performance of all the proposed algorithms and compare their performance against previously propose heuristics, we use the disk drive of a laptop computer as an embedded system. In our experiments, we model service times, i.e., we assume that the time to service requests is proportional to the size of the request. Under these conditions, we observed that in some cases a simpler algorithm that shuts down the system whenever it encounters an idle period performs better than the proposed adaptive algorithms. Another contribution of this paper is an analytical explanation of this observation. The final contribution of this paper is the presentatation of an analytical proof that upper bounds the latency incurred by a subsystem, which employs a shutdown power management policy. This allows system designers to effectively tradeoff the savings in power with the increase in the system latency due to aggressive shutdown power management schemes.

***Index Terms*—Competitive ratio, embedded systems, latency, online algorithms, power management algorithms, service times, system level power.**

Manuscript received September 4, 2000; revised September 10, 2001. This paper was recommended by Associate Editor M. Pedram. The work of S. Irani was supported in part by NSF under Grant CCR-9625844 and Grant CCR-0105498 and in part by ONR under Award N00014-00-1-0617. The work of R. K. Gupta and D. Ramanathan was supported in part by NSF MIP, in part by ARPA/ITO PACC under Grant F33615-00-C-1632, and in part by SRC under Grant 2001-HJ-899.

The authors are with the Department of Information and Computer Science, University of California, Irvine, CA 92697 USA (e-mail: dinesh@ics.uci.edu; irani@ics.uci.edu; rgupta@ics.uci.edu).

Publisher Item Identifier S 0278-0070(02)01780-3.

I. Introduction

POWER dissipation in a very large scale integration (VLSI) system is a primary design consideration. In the design of portable computing devices, greater attention has to be paid to power estimation and management techniques. Over the past few years, methods to estimate and minimize power in the design of circuits have been reported. Several excellent reviews of power minimization techniques are presented by Pedram [9], Devadas and Malik [10], Chandrakasan and Brodersen [14], Najm [11] and Luca [26].

Low power VLSI design can be achieved at various levels of abstraction during the design process. These include the system level, behavioral level, the register transfer level (RTL), and the gate level. Most techniques in the literature are focused at minimizing power at the RTL level. This paper focuses on the problem at the system level.

A. System Model

Our model of the system is a reactive real-time embedded system that continually reacts to the stimuli coming from its environment and performs this interaction under timing constraints. This interaction causes the system to dissipate power in order to service the request. The interarrival time between requests is typically unknown and may not fall into any pattern. The requests typically arrive unpredictably and generally do not fall into well-known probability distributions [1], [2], [4], [5]. Therefore, a good power management strategy would selectively turn on and turn off the system to minimize the overall power consumption based on the arrival of the requests. In particular, the optimal power dissipation will be by an algorithm that knows the interarrival times between requests ahead of time. Further, during system level design, the internals of the system under consideration are not known. In order to determine an effective power management strategy for such a system, we assume that at least one power metric of the system is known: the ratio of the idle and the startup power dissipation. In this paper, we discuss strategies that selectively shutdown subsystems and turn them back on when needed.

One side effect of shutting down subsystems and restarting them when required is an increase in the subsystems' latency. When a subsystem is shut down and needs to be restarted, it incurs some delay overhead before the subsystem is initialized and is ready to service input requests. Since power management is needed for subsystems that are timing critical, the effect of latency arising from the power management scheme is crucial to the design of a system that complies with its timing requirements. This paper analyzes the effects of power management on the latency of the system. The tradeoff between latency and

power management is that the more aggressive the power management strategy, the more often the system is powered down, which in turn increases the latency of the system. The latency of the system also effects the power management strategy. When requests for service are delayed, the intervals of time in which the system is idle can change, which can effect the relative performance of power management algorithms. In this paper, we discuss lower bounds on the latency of subsystems in the presence of power management algorithms.

B. Background and Previous Work

There are essentially two kinds of power management strategies: nonadaptive and adaptive. Typically, power management creates sleep states with various levels of power savings and delay overheads which can be externally controlled. Typically, power management algorithms are carried out based on the following strategy: go to sleep after the system has been idle for a period of time. Algorithms that statically compute the sleep states based on the properties of the system and do not change when the requests are received are called nonadaptive algorithms. Adaptive power management algorithms dynamically determine the time after which the system shuts down as the system is servicing requests. Intuitively, we would expect adaptive algorithms to perform better than nonadaptive ones. Using heuristic algorithms, all previous works have experimentally shown that this intuition holds.

This paper builds upon earlier works on power management strategies [1], [2], [6], [24]. Srivastava *et al.* [6] conducted an extensive analysis on different system shutdown approaches. The authors have proposed an adaptive shutdown algorithm for power saving of event-driven systems. The authors first collected sample traces of on–off activity on an X-server, then they proposed two adaptive shutdown formulas based on the analysis of the sample traces: one using a general regression-analysis technique to correlate the length of the upcoming idle period and the second based on on–off activity behavior. These results demonstrated *experimentally* that adaptive shutdowns can reduce power dissipation in systems.

This result was followed up by Hwang and Wu [1]. Their analysis adapted the exponential-average approach [12] used in the CPU scheduling problem for the prediction of idle periods. They proposed an algorithm using two new strategies: prediction-miss correlation and prewakeup. The most significant contribution of this work is that these methods were independent of the traces obtained for the system under consideration. However, in the absence of analysis of the algorithm, it is not clear how close their results are to the optimal solution. Further, there is significant cost associated with a hardware implementation of their strategy.

More recently, Paleologo *et al.* [2] proposed adaptive power management algorithms for embedded systems by modeling the problem as a stochastic optimization problem. They use a laptop's disk drive as an embedded system [27] and using the Auspex file traces [25] they generate a Markov model using an exponential distribution as the base distribution. However, their assumption that the interarrival times between requests is exponentially distributed is hard to justify and may not always hold in view of significant correlation between accesses. As a result, their model is only as good as: a) the distribution they assume the traces to fall into and b) the traces themselves. There has also been work on modeling the dynamic power management problem using stochastic methods [15]–[17]. These papers also assume that the interarrival times between requests are exponentially distributed which eases the analysis but seldom occurs in real-life applications. There has also been some work on experimentally comparing the performance of various power management algorithms by building a framework for power management for notebook computers [17].

An important issue when designing power management algorithms is its effect on latency. None of these papers model the effects of power management on the latency of the system. In [1] the authors indicate that latency incurred due to the power management algorithm is a critical design parameter; however, they do not model it in their analysis. The authors experimentally determine the latency that is incurred by software systems for which power management strategies have been defined. However, their results cannot be generalized and applied to other systems, especiallly hardware systems where latency is a critical design parameter. Also, most of the previous work done in this area does not account for the service time of a request while modeling the problem.

C. Contributions

One of the problems with existing heuristics is the lack of indicators on how close these heuristics are to the optimal solutions, whether an optimal solution exists, and an understanding of why some heuristics perform better than others. Another problem is that these heuristics are dependent on the input distributions: they have been generated by examining trace patterns of several experimental examples. In this paper, we attempt to develop a framework within which these heuristics can be analyzed and propose algorithms that are independent of the input distribution. This framework is based on the notion of *competitive analysis*.

Competitive analysis has been used as a technique to analyze various online problems [18]–[23]. In [7] and [8], the authors analyze the spin-block problem which is similar to the power management problem without any latency consideration. The algorithms and proofs presented in this paper have been adapted from those works. There are significant changes in the problem formuation with latency consideration which the authors in [7] and [8] have not addressed.

On the whole, this paper makes five major contributions. The first contribution is the introduction of a formal analysis technique called competitive analysis to the power management problem. Competitive analysis as applied to power management does not depend on trace patterns and can be used to formally analyze the performance of existing heuristics. For this analysis, we do not model the service times for a given request, rather, we assume that the requests are serviced instantaneously. Using this technique, we can prove bounds on the power dissipation achieved by a power management algorithm. The second contribution is the presentation of an optimal nonadaptive online algorithm. The third contribution is a lower bound for any adaptive online algorithm. We show that no adaptive online algorithm can dissipate less than about

1.6 times the power dissipated by the optimal offline algorithm in the worst case unless it keeps a complete history of all interarrival times seen so far. We also show that in order for any online algorithm to achieve this lower bound, it may have to maintain a *complete* history of the interarrival times of the requests in the input sequence seen so far. Since this is not practical, we present a simple algorithm that uses only the last interarrival time to predict the arrival of the next request. We show that this algorithm performs as well as the heuristics, but we can bound its worst case performance and resource usage. Futher, our analysis shows that in order to design aggressive power management strategies, system designers will have to budget greater resources toward the power management.

The fourth contribution of this paper is the observation that *when service times for a request are modeled*, a simple algorithm that shuts down the system whenever it encounters an idle period performs better than the proposed adaptive algorithms in some cases and especially when the revival time is small. We analytically explain these observations. The final contribution of this paper is the presentatation of an analytical proof that upper bounds the latency incurred by a subsystem for which a shutdown power management policy is devised. We show that this bound is achieved if the system is shut down even once, thereby proving that this bound is very realistic and will occur in practice. We believe that latency is a critical design parameter that the system designer must account for while designing a system that has strict timing requirements. The system designer has to manage latency as well as dissipate as little power as possible.

To test the performance of all the proposed algorithms and compare their performance against previously proposed heuristics, we use the disk drive [27] of a laptop computer as an embedded system. We compared our algorithms with Hwang and Wu's heuristic algorithm [1] which is the most comprehensive algorithm that does not depend on the input sequence patterns.

II. Problem Definition

Our model of the problem consists of a reactive real-time embedded system that receives service requests online; the requests are not known in advance and do not necessarily fall into any well-known distribution pattern. Each service request j comes with a service time d_j that is also not known in advance and could potentially vary from request to request. Requests are handled in a first-come/first-served manner. The system cannot be idle as long as there are requests waiting in the system. If there are no requests waiting for the system, the system can be shut down during this idle period. If there are no requests waiting for the system and the system is off then when a request arrives, the system must be turned on immediately in order to service the newly arrived request. Our goal is to design an effective power management strategy for the system taking into account system performance (latency) and resources needed to implement such a strategy.

In this paper, we only consider a two-state shutdown system: the system can be in one of two shutdown states, i.e., shutdown or idling. In [3] we have examined the multistage shutdown policies and presented algorithms that work under various state-to-state transitions policies.

The system dissipates E_r units of energy during revival. It takes the system T_r units of time, called the *revival time*. During this time, the system expends energy at a rate of P_r, called the *revival power*, to go from the shutoff state to a state where it can start servicing requests. Let us assume that the average power dissipated by the system when it is idle is P_i. Let us assume that $P_r > P_i$. This implies that turning the machine on requires more power per time unit than leaving the machine in the "on" state. We denote

$$t_k = \frac{E_r}{P_i}.$$

t_k is called the *shutdown threshold*.

Example 1: Consider the IBM disk drive [27] that we model in our experiments. It has $T_r = 4$ s, $P_r = 4.5$ watts, and $P_i = 0.85$ watts which gives us a *shutdown threshold*, $t_k = (4^*4.5)/0.85 = 21.17$ s. This implies that after the disk has been idle for 21.17 s, it is advantageous to shutdown the system.

In our analysis, t_k is an important metric; it is the length of time such that if the system is idle for time t_k, the energy that is expended is the same as the energy required to revive the system from the powered down state. Note that if latency considerations are ignored, then if a given idle period is less than t_k, the optimal algorithm will remain powered up for the duration of the idle period. If the idle period is greater than t_k, the optimal algorithm will power down at the beginning of the idle period. Of course an algorithm working with requests as they come in will not know the length of an idle period when it begins since it will not know the time of arrival of the next request. Such an algorithm is called an online algorithm and is explained next.

We say that an algorithm is *online* if it operates without knowledge of the arrival time or service requirements of future requests. In other words, it does not depend on the complete data set to make its decisions: decisions are made based on data that has already arrived and the decision-making process may be changed as the historical data changes.

Our goal will be to minimize the total power consumption. At the same time, we would like to guarantee an upper bound on the latency of any request which enters the system. Further, the power management strategy should reflect the effects of implementations in hardware as well as software.

III. Competitive Analysis

Consider decisions that depend on future events of which the decision maker has only partial knowledge. As there is no certain method to determine the future, such decisions are taken *online*. A common approach to solve an online problem is to assume that the future events are distributed according to some, known or unknown, probability distribution and to devise a decision mechanism whose expected (average) performance is maximized subject to these assumptions. The following are some examples of online problems: 1) replace algorithms for maintaining a cache; 2) load balancing in a distributed environment; and 3) buying and selling stock in the stock market to maximize gains.

Under the assumption that there is some underlying probability distribution which governs future events, one might try

to "learn" these distributions so as to improve the decisions, as time goes by. This approach assumes that the real-life problems fall into some distribution that can be learned. An alternative approach to solving *online* problems, called *competitive analysis*, has been considered by computer scientists in recent years. Sleator and Tarjan [23] introduced the idea in the context of dynamic data structures. The competitive analysis approach assumes that the input to the problem is generated by an adversary. The performance of the *online strategy* (algorithm), which has no knowledge of the future events, is compared with that of an *optimal offline strategy* (adversary), which has complete knowledge of the future and operates optimally based on this information. Since its introduction, competitive analysis has been successfully used to analyze online algorithms in various areas of computer science: scheduling, graph coloring, robot motion planning, computational finance, matching, k-server, and file access and allocation in distributed systems. In this paper, we adapt competitive analysis to estimate the power dissipation in an embedded system.

Consider a fixed sequence σ of inputs to an embedded system. Let $P_{\rm opt}(\sigma)$ denote the minimum power that an offline algorithm (adversary) can dissipate on this sequence and let $P_A(\sigma)$ denote the power dissipated by an online algorithm A on the same sequence. The algorithm A is said to achieve the *competitive ratio* r, if for all values of inputs

$$P_A(\sigma) \leq r \cdot P_{\rm opt}(\sigma).$$

We are interested in finding the algorithm A that minimizes the competitive ratio. The competitive ratio as a measure for online algorithms was first introduced in [23]. We assume that the adversary generates the input sequence to the algorithm A, based only on the description of the problem. Intuitively, for our application, if an algorithm A that measures power dissipation is said to be r-competitive against an oblivious adversary, it means that in the worst case the algorithm would dissipate r times as much power as the optimal offline algorithm.

In order to simplify our analysis for the adaptive algorithms, first we assume that the requests are serviced instantaneously. This implies that for request j, the service time d_j is zero. In Section VI, we show that relaxing this assumption causes the algorithms based on this assumption to perform poorly. We present a theoretical analysis of these results.

IV. An Optimal Nonadaptive Power Management Algorithm

In this section, we present a nonadaptive online algorithm for power management. For nonadaptive algorithms, the service times for a particular job is of no consequence and is not modeled here. The main strategy is to wait for a particular threshold of idle time before shutting down the system. We establish a lower bound on the competitive ratio for any nonadaptive online algorithm, then show that the proposed algorithm achieves this lower bound thereby proving that our algorithm is optimal. We note that this problem is identical to the ski-rental problem discussed in [13] and its solution is typically presented as an introduction to online algorithms and competitive analysis.

```
Algorithm NonAdaptive:
(1) if (idle_state) then
(2)     idle_intervals = idle_intervals + 1;
(3) if (idle_intervals ≥ k) then
(4)     shutdown
```

Fig. 1. The optimal nonadaptive online algorithm. The algorithms enters the idle state if it has no request to service during a basic unit of time.

Before we present the algorithm, we must indicate that the *shutdown threshold* t_k is usually measured using a base unit of time t. This base unit of time can be as granular as provided by the system under consideration. As a result, we now define the number of these base time units that when elapsed provide us the shutdown thresold. We define the *shutdown-threshold estimate* as an integer

$$k = \left\lceil \frac{t_k}{t} \right\rceil.$$

Example 2: Consider the IBM disk drive in Example 1. Its shutdown threshold is 21.176 s. Let us suppose that the finest granularity of time it can measure is 1 μs. Hence, its shutdown-threshold estimate is $\lceil 21\,176\,472.3 \rceil = 21\,176\,473$ ticks of a 1-μs clock. This means that we now count the number of clock ticks (during which the system is idle) that expire after which the system is shutdown. This is typically the case in the design of most hardware systems.

On the other hand, if the finest granularity the disk could measure was 1 ms, its shutdown-threshold estimate would be $\lceil 21\,176.4723 \rceil = 21\,177$ ticks of a 1-ms clock.

Throughtout the rest of this paper, we will use the shutdown-threshold estimate as the parameter to examine the performance of all proposed algorithms.

A. The Algorithm

E_r is the energy dissipated during revival, T_r is the revival latency, $P_r = E_r/T_r$ is the revival power, and P_i is the power dissipated when the system is idle. Using

$$k = \left\lceil \frac{E_r}{P_i \cdot t} \right\rceil \tag{1}$$

$$k \geq \frac{E_r}{P_i \cdot t} \tag{2}$$

$$E_r \leq k \cdot P_i \cdot t \tag{3}$$

the shutoff strategy is simple: wait for $k-1$ idle time units and then shut down the system. If a service request arrives before $k-1$ time units, the system is not shut down; otherwise it is shut down after $k-1$ time units. Let us denote this deterministic online strategy by NONADAPTIVE as shown in Fig. 1. It says, when the system is in the idle state (variable `idle_state` is true), it counts the number of idle intervals. When the number of idle intervals is greater than or equal to k, the system shuts off.

Fig. 2 shows the behavior of the algorithm NONADAPTIVE as well as the behavior of the optimal offline adversary on a sequence of input requests.

B. Analysis of Algorithm

Lemma 1: Algorithm NONADAPTIVE is $2-1/k$ competitive.

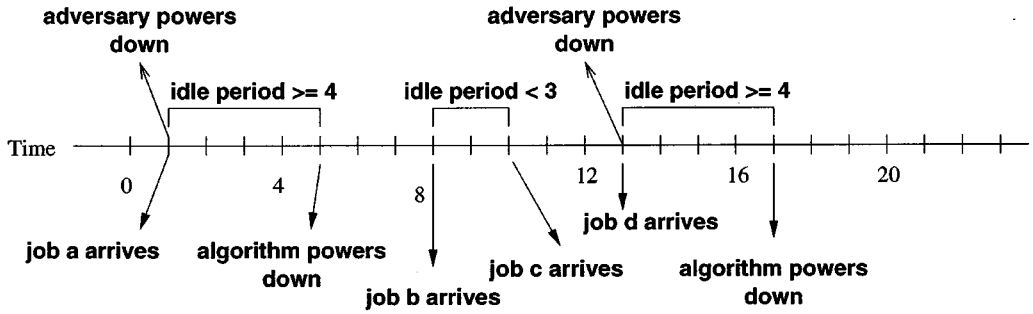

Fig. 2. The behavior of the algorithm and the optimal adversary on an input sequence. We assume that the first request (job) which arrives at time 0 is serviced instantaneously. The algorithm waits for four idle periods before it powers the system down, whereas the adversary powers it down immediately. The adversary powers the system immediately, since the adversary knows when the next request will arrive, in this case at time 8 and there are more than four idle intervals. On the other hand, when the next request (job) arrives at time 8, it is serviced instantaneously—neither the adversary nor the algorithm power down the system since the next request arrives at time 10. The adversary can power down immediately when the next request is at least k (in this case we have chosen $k = 4$) units away. The algorithm, on the other hand, has to wait in the idle state until it becomes advantageous to power down. We have assumed here for the sake of simplicity that the requests (jobs) are serviced instantaneously.

Proof of Lemma 1: Let us assume that n time units expire between two adjacent requests. This information is not known to algorithm NONADAPTIVE. We merely use it for the analysis. However, since the adversary controls the sequence of interarrival times, n is determined by the adversary. If the system is not shut down ($n \leq k-1$), then the offline algorithm and the online algorithm dissipate the same amount of power since they both remain idle until the arrival of the next request.

If the system is shut down ($n > k-1$), then the algorithm NONADAPTIVE dissipates

$$((k-1)\cdot t\cdot P_i)+E_r$$

energy.

The first term comes from the fact that the system was idle for $(k-1)$ time units of length t and dissipated P_i units of power in each time unit. The second term comes from the fact that the system was shut down and will have to be restarted to service requests (revival energy).

Now using (3)

$$((k-1)\cdot t\cdot P_i)+E_r \leq ((k-1)\cdot t\cdot P_i)+k\cdot t\cdot P_i = (2k-1)P_i\cdot t.$$

If the system is shut down ($n \geq k-1$), the *offline adversary* has two choices.

1) The adversary shuts the system down after some $k' < n$ units of time. In this case, k' might as well be 1 since any value of $k' > 1$ will imply that the adversary dissipates more power than required. Therefore, the adversary shuts down the system as soon as the first idle period has been encountered, provided the adversary knows that this idle period will be greater than k time units. Note that the adversary knows when the next request will arrive and can therefore make this choice when it is advantageous to do so. In this case, the adversary dissipates

$$E_r \leq k\cdot t\cdot P_i \tag{4}$$

units of energy.

2) The adversary does not shut down the system at all. It remains idle until the arrival of the next request, in which case it dissipates

$$n\cdot t\cdot P_i \tag{5}$$

units of energy.

The adversary tries to maximize the competitive ratio; it maximizes the algorithms power dissipation and minimizes its power dissipation. The adversary dissipates minimum power only when the power dissipated by the two choices it has are equal, which gives us $n = k-1$.

Thus, using (4) and (5), algorithm NONADAPTIVE attains competitive ratio

$$\min_{n>k-1}\frac{(((k-1)\cdot P_i)+P_r)\cdot t}{n\cdot t\cdot P_i}\frac{((k-1)\cdot P_i)+P_r}{k\cdot P_i} = \frac{((k-1)\cdot P_i)+k\cdot P_i}{k\cdot P_i} = 2-\frac{1}{k}.$$

■

The adversarial strategy depends on the value of n, the interarrival time between the two requests. Since the adversary picks the input sequence, the adversary has control over n. The adversary chooses n such that it causes the online algorithm to power down, forcing it to incur the additional power dissipated in powering up. Hence, the adversary picks n to be k. This implies that the adversary picks a sequence where the requests come to the algorithm as soon as it has powered down forcing it to dissipate the extra energy in powering up. On this sequence, the adversary shuts off as soon as an idle sequence is encountered. Therefore, the adversary picks n to be k and dissipates $k\cdot t\cdot P_i$ units of energy. This leads us to the lower bound argument presented in Section IV-C.

The factor $1/k$ is a consequence of the quantization of time based on the granularity of the clock used to measure time. It is easy to see that if we assume that time is continuous, the competitive ratio will tend to two since k will tend to infinity. Based on the type of embedded system, hardware or software, we can make some assumptions about the discreteness of time which will effect the competitive ratio. We do not consider the increase in latency of the system whose power we were are trying to

manage in this analysis; we use the experimental results to reach our conclusions.

C. Lower Bound Proof

In this section, we will show that algorithm NONADAPTIVE is optimal. This involves proving a lower bound for the competitive ratio of the problem.

Lemma 2: There exists no nonadaptive deterministic algorithm that can attain a competitive ratio less than $2 - 1/k$.

Proof of Lemma 2: First, notice that any deterministic strategy for this problem is a "threshold" strategy. A solution can be completely characterized by specifying a threshold after which the system shuts down. Let S_m be any arbitrary deterministic strategy where the "threshold" is m. There are two possible cases for analysis.

1) $m < k$: S_m attains a competitive ratio of

$$\frac{(m-1)\cdot t\cdot P_i + E_r}{m\cdot t\cdot P_i} \leq 1 + \frac{k-1}{m} \geq 2. \tag{6}$$

2) $m \geq k$: S_m attains a competitive ratio of

$$\frac{(m-1)\cdot t\cdot P_i + E_r}{E_r} \geq \tag{7}$$

$$\frac{(k-1)\cdot t\cdot P_i + E_r}{E_r} = 2 - \frac{1}{k}. \tag{8}$$

Therefore, from (6) and (8) we see that S_m consumes more power than algorithm NONADAPTIVE. Since the choice of S_m was arbitrary, Lemma 2 is proved. ■

Theorem 1: Algorithm NONADAPTIVE is optimal.

Proof of Theorem 1: Lemmas 2 and 1 prove that under the assumption that time has been discretized, strategy NONADAPTIVE is optimal. The same analysis applies under the assumption that time is continuous but the bound is 2.

D. Effects of Discretizing Time

The term $1/k$ occurs in the competitive ratio since we have assumed that time is discrete and broken into k intervals of time t, each after which we shutdown the system. If we assume that time is continuous, it is easy to see that the algorithm NONADAPTIVE will be 2-competitive. The discreteness of time has implementation implications for hardware and software systems and are addressed in this section.

In hardware systems, the system clock specifies the granularity of time for that system. Let us denote the period of this clock by t. In order to estimate t_k, we will have to wait some number of clock cycles of period t. Depending on whether t is a multiple of k or not, there will be an error in the estimation. We have used the ceiling operator in our analysis.

Moreover, hardware systems can be shut off after k ticks of the system clock. In software systems, however, there is no guarantee that the system (e.g., an X-server) can be shut off after k time units. Software systems get shutoff interrupts whose arrival times could differ based on the load on the systems. As a result, software systems could potentially dissipate more power than hardware systems even though they both use the same algorithms.

V. Adaptive Power Management Algorithms

Adaptive power management algorithms change the length of the interval after which shutdown occurs dynamically based on past performance. Such algorithms typically have improved performance due to their ability to base decisions on the nature of the data and dynamic system behavior. To begin with, we assume that each interarrival time is independently chosen from the same distribution and adaptive algorithms attempt to "learn" this distribution. Further, we also assume that the service time for each request is negligible. In Section VI, we relax this assumption and present experimental results that show the effect of modeling service times on the performance of various algorithms.

We show analytically that the lower bound on the competitive ratio for adaptive algorithms is better than the competitive ratio for nonadaptive algorithms. Learning the *true* distribution that generates the interarrival times would be very memory intensive since the algorithm would have to keep track of a significant part of the history of the sequence. Since this is not practical, we present an algorithm that uses the last input sequence to predict the next event. We also show that this algorithm is 3-competitive against an all knowing adversary. The analysis in this section builds upon the work of Karlin *et al.* [8] and specifically their formulation of the spin block problem [7]. We augment their theoretical work with experimental results. However, the authors in [7] and [8] do not model service times in their analysis.

A. Lower Bound for any Adaptive Algorithm

In the case of adaptive algorithms, assume that each interarrival time is independently chosen from some distribution, say $\Pi(t)$. Then, technically it is just a matter of "learning" this distribution and being able to pick the shutdown time using the estimator distribution. This implies that there are two subproblems to solve.

- Learn the *true distribution* from which the interarrival times are being generated.
- Pick a shutdown time, say τ.

Let us begin by assuming that $\Pi(t)$ has been learned by the algorithm. In order to distinguish between the *true distribution* and the learned distribution, we denote the learned distribution by $\pi(t)$. We also assume that time is continuous for the purpose of analysis.

Let σ denote an arbitrary interarrival time generated from $\Pi(t)$. Now, let algorithm A shut down the system after τ units of time in interval σ.

Theorem 2: No adaptive algorithm can achieve a competitive ratio better than $e/(e-1)$.

Proof of Theorem 2: We prove the lower bound by showing that algorithm A can be no better than $e/(e-1)$-competitive. Let t_k be a real number such that $t_k = E_r/P_i$. Recall that this is the same definition we have for t_k in Section II. If we assume that $Eng_A(\sigma)$ is the energy dissipated by algorithm

A when it encounters σ, then the expected energy dissipated by A is

$$E\left[Eng_A(\sigma)\right] = \int_0^{\tau} t \cdot \pi(t) \cdot P_i dt + \int_{\tau}^{\infty} (\tau \cdot P_i + E_r) \cdot \pi(t) dt. \tag{9}$$

The first term within the integral is the energy dissipated when the interarrival time is less than the shutdown threshold τ picked by algorithm A. The second term is the energy dissipated when the interarrival time is greater than the shutdown threshold. In this case the system gets shutdown: the system dissipates $\tau \cdot P_i$ before shutdown and E_r energy for revival.

Therefore

$$E\left[Eng_A(\sigma)\right] = P_i \left(\int_0^{\tau} t \cdot \pi(t) \cdot P_i dt + \int_{\tau}^{\infty} \left(\tau \cdot P_i + \frac{E_r}{P_i} \right) \cdot \pi(t) dt \right) \tag{10}$$

$$E\left[Eng_A(\sigma)\right] = P_i \left(\int_0^{\tau} t \cdot \pi(t) dt + \int_{\tau}^{\infty} (t_k + \tau) \cdot \pi(t) dt \right). \tag{11}$$

Algorithm A has to choose values for τ based on $\pi(t)$ such that the expected energy dissipated is minimized. The optimal offline algorithm (adversary) has

$$Eng_{\text{opt}}(\sigma) = \begin{cases} t \cdot P_i, & t \le t_k \\ E_r, & t > t_k. \end{cases}$$

In other words

$$E\left[Eng_{\text{opt}}(\sigma)\right] = \int_0^{t_k} t \cdot P_i \cdot \pi(t) dt + \int_{t_k}^{\infty} E_r \cdot \pi(t) dt$$

$$E\left[Eng_{\text{opt}}(\sigma)\right] = P_i \left(\int_0^{t_k} t \cdot \pi(t) dt + \int_{t_k}^{\infty} \frac{E_r}{P_i} \cdot \pi(t) dt \right)$$

$$E\left[Eng_{\text{opt}}(\sigma)\right] = P_i \left(\int_0^{t_k} t \cdot \pi(t) dt + \int_{t_k}^{\infty} t_k \cdot \pi(t) dt \right).$$

The adversary picks the distribution $\Pi(t)$ in order to maximize the competitive ratio and the expected energy dissipated by the algorithm. The algorithm A picks τ to minimize the competitive ratio.

Solving the differential equations obtained by differentiating twice with respect to τ, we obtain

$$\pi(t) = \begin{cases} \frac{1}{(e-1)t_k} e^{t/t_k}, & 0 \le t \le t_k \\ 0, & \text{otherwise.} \end{cases}$$

By setting $\int_0^{\infty} \pi(t) dt = 1$, the algorithm A yields a competitive ratio of

$$\frac{e}{e-1} \approx 1.581\,976\,7 < 1.6. \tag{12}$$

Since the choice of σ was arbitrary, the analysis applies to any interval in the request sequence. This shows that there is no algorithm that can achieve a competitive ratio better than $e/(e-1)$. ■

In this solution, we have given the algorithm the advantage of estimating $\Pi(t)$ exactly. However, if this is not the case, the algorithm will not be able to attain the bound proved here. To estimate $\Pi(t)$ accurately, the optimal algorithm may have to maintain accurate statistics by keeping track of the history of the interarrival times of requests. A practical alternative to this algorithm is to keep track of the last few interarrival times in order to determine what to do the next time after a request has been serviced. Interestingly, a different version of this algorithm is presented by Hwang and Wu [1]. The authors arrive at their algorithm by examining trace patterns.

```
Algorithm ADAPT:
(1) τ = curr_arrival_time - prev_arrival_time;
(2) if (idle_state) then
(3)     idle_intervals = idle_intervals + 1;
(4) if (τ ≥ k) then
(5)     shutdown
(6) else
(7)     if (idle_intervals ≥ k) then
(8)         shutdown
```

Fig. 3. The 3-competitive nonadaptive algorithm for power management.

B. A 3-Competitive Adaptive Algorithm

Fig. 3 outlines the algorithm. τ is computed dynamically and determines when the system will be shutdown.

We will now show that this algorithm is 3-competitive. There are two cases to consider: a) $\tau < k$: This case is identical to the deterministic case and we have shown that this approach achieves a competitive ratio of 2. b) $\tau \ge k$: In this case, the previous interarrival time was greater than k and as a result in the previous interval, the adversary dissipated at least $k \cdot P_i$ energy, whereas we dissipated P_r (to revive the system), since we shut down the system as soon as it was idle. Hence, we can add the power dissipated in this interval to the power dissipated by the adversary in the previous interval thereby amortizing the power dissipated in this interval with the power dissipated in the previous interval. This yields an overall competitive ratio of 3.

This algorithm is similar to the one presented by Hwang and Wu [1]. Their algorithm is used for software systems and computes τ as the cumulative weighted average of the previous interarrival times. This computation uses significant hardware resources. Therefore, this strategy is not suitable for implementation into hardware. Since they use a larger history of interarrival times to compute their shutdown threshold, their algorithm will be more competitive than ADAPT, but only marginally so. The 3-competitive algorithm, ADAPT is simple and can be used in both in hardware and software systems. Also, the analysis allows system designers to identify power critical subsystems and incorporate aggressive power management techniques for these systems by committing to more resources upfront in the system design process.

VI. MODELING SERVICE TIMES

Adaptive power management algorithms change the length of the threshold after which shutdown occurs dynamically based on past performance. Such algorithms typically have improved performance due to their ability to base decisions on the nature of the data and dynamic system behavior. Some power management algorithms have assumed that each idle period can be considered to be a random variable and the length of each idle

period is independently and identically distributed. These algorithms attempt to learn the distribution of idle periods as time progresses and optimize the threshold based on the estimate of this distribution [7], [8]. We have also presented such an adaptive algorithm in this paper.

Other algorithms have assumed that there is significant correlation between the length of recent idle periods and the length of idle periods in the near future. These strategies use recent idle periods to predict whether the upcoming idle period is likely to be more or less than the parameter k derived in Section II. Recall that k is the length of time such that if the system is idle for time k, the energy that is expended is the same as the energy required to revive the system from the powered down state. If an online algorithm knew whether the upcoming idle period was going to be shorter or longer than k, it could perform optimally. If the interval is less than k, it will stay powered up, otherwise power down at the beginning of the idle period. Thus, given an estimate of the length of the upcoming idle period, if the estimate is less than k, stay powered up; if it is greater than k then power down immediately. If, however, the algorithm chooses to stay powered up and finds that the idle period has lasted time k, the algorithm then powers down. This latter step is important in case the idle period is extremely long. These adaptive schemes are very effective if the arrivals are very bursty, a property which characterizes many request arrival sequences. That is, the predictions are accurate if short idle periods tend to be followed by short idle periods and long idle periods tend to be followed by long idle periods.

The ADAPT algorithm shown in Fig. 3, uses the length of the last idle period to predict the length of the next idle period. The algorithm presented by Huang and Wu [1] predicts the length of the next idle period by an exponentially weighted average of the lengths of the previous idle periods. None of the previous studies incorporates service time or power-up latency into their models. That is, they assume that these values are zero.

Interarrival time is defined as the time between two consecutive arrivals in a sequence. We denote interarrival time between requests j and $j+1$ as $I_j = a_{j+1} - a_j$. Interservice time is defined as the time between the end of one request and the arrival of the beginning of the next request. The adaptive algorithms described here were all introduced in the context where all service times and power-up times are assumed to be zero. In this case, the length of the last idle period is the same as the last interservice time which is also the same as the last interarrival time. Thus, it is not obvious which value to use with the adaptive schemes when introducing timing considerations into the model. Although we experimented with all three values, the results we present here use interservice time since that value seems to capitalize the most on correlations in the arrival sequence.

VII. EXPERIMENTAL RESULTS

We use the disk drive [27] in a laptop as an embedded system. We have obtained traces for the use of an Auspex File Server [25] from Berkeley's NOW project and apply these traces as stimuli to the laptop's disk drive. This drive and the traces were also used in the experiments performed by [2]. The data is all the requests that were made to an Auspex file server maintained by the University of California at Berkeley. We sieved the data so that we look at requests made by a single user to the file server. The original data was a union of requests obtained from all users that accessed data from the file server over a period of 7 days, 24 hours a day. In this paper, we picked one day from these 7 days at random—this was day number 6. It so happens that there were 12 hosts that made requests to the Auspex server on that day and so we have 12 different traces. Trace t6.H1064 represents the requests made by host 1064 to the Auspex server on day number 6. We believe that the data obtained from the NOW project is representative of a typical sequence of requests that the user of a laptop would generate which includes read, write, and update requests.

The disk drive has the following power characteristics. Its internal clock operates at 10 microseconds. The average power dissipated in servicing any request is denoted by $P_i = 0.85$ watts. The revival power dissipated is $P_r = 4.5$ watts. The revival latency is $T_r = 4$ s. Using these parameters, we modeled and simulated the following scenarios to compute the power dissipated.

1) *Algorithm Optimal*: This algorithm is assumed to know the sequence of arrival of requests and their service time, *all in advance*. As a result, it can make decisions based on knowledge of the future. The algorithm will shut down the system immediately if the next idle period is greater than k time units. An oracle will keep the system idle if the idle period is less than k time units. Any proposed algorithm tries to compete with this oracle strategy. This oracle is also referred to as the optimal adversary since every algorithm is trying to achieve the results obtained by an algorithm that has knowledge of the future.
2) *Algorithm* IMMEDIATE: This algorithm shuts down the system whenever it sees an idle period, however small, and incurs the overhead cost of revival.
3) *Algorithm Adapt*: This algorithm uses the previous interservice time, τ, to predict the next idle period. If this value is greater than k it shuts down immediately assuming that the next idle time will be greater than k time units. If τ is less than k, it keeps the system idle for a period of k unless a new request arrives.
4) *Algorithm HwangWu*: This algorithm is an adaptation of Hwang and Wu's algorithm **[1]**. It uses a cumulative average τ of the interservice times to predict the next idle period. If τ is greater than k, it shuts the system off as soon as an idle period is seen, otherwise it keeps the system idle for k time units before shutting down the system.

A. Discussion

Figs. 4 and 5 show the performance of the four power management algorithms in terms of power efficiency and latency. The latency of the system as a result of power management does not increase any more than 4 s, which is the analytical upper bound on the increase in system latency. Algorithm OPTIMAL is the optimal performance one can expect from any algorithm since OPTIMAL has complete knowledge of the future and hence makes optimal decisions. We note that IMMEDIATE outperforms the other algorithms in terms of power dissipation on two

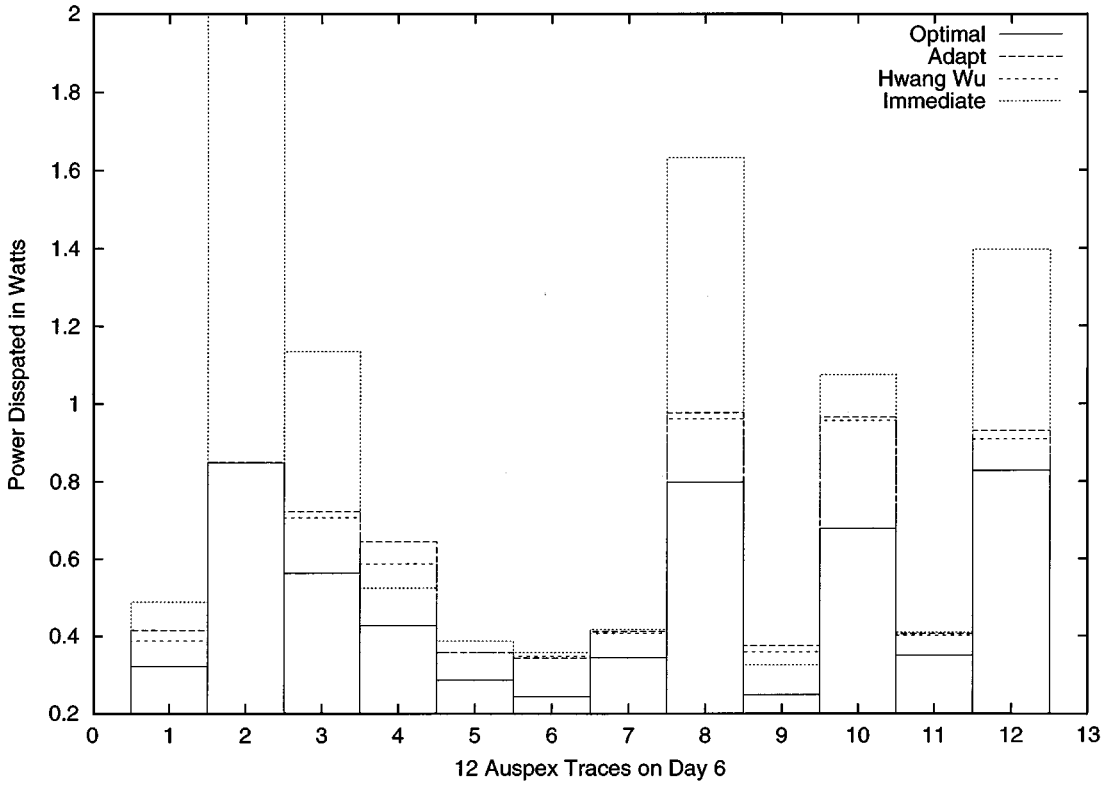

Fig. 4. Power dissipation in watts of the laptop's disk drive when the Auspex server traces are applied to it. Each column shows the power dissipated when that particular power management algorithm is used for the disk drive. Immediate's power dissipation for trace 4 and trace 9 is less than ADAPT.

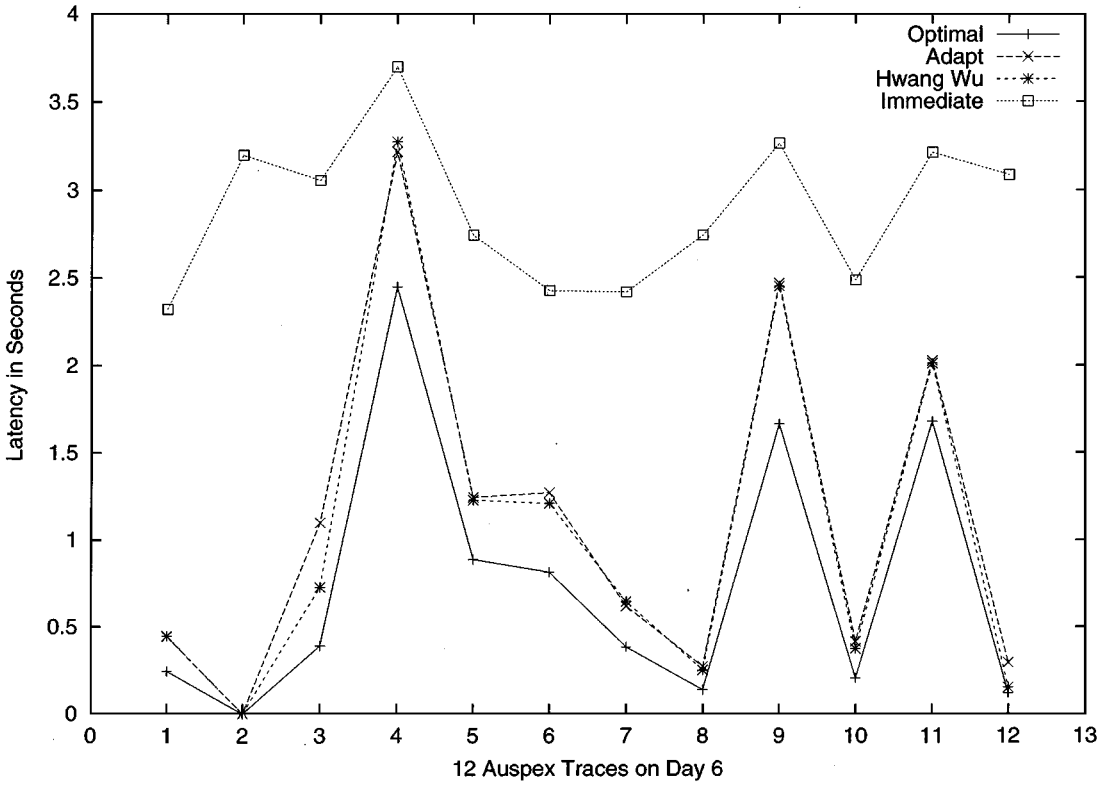

Fig. 5. Increase in latency (in seconds) of the laptop's disk drive when the Auspex server traces are applied to it. Each column shows the increase in latency when that particular power management algorithm is used for the disk drive.

traces. However, its effect on latency is worse than OPTIMAL and ADAPT. Further, we hypothesized that the peformance of IMMEDIATE would get better as T_r decreased. In order to test this hypothesis, we ran all the experiements varying T_r from 4 s, to 0.4 s and 4 ms. The results of our experiments are shown in Table V and they validate our hypothesis. *It is not unreasonable*

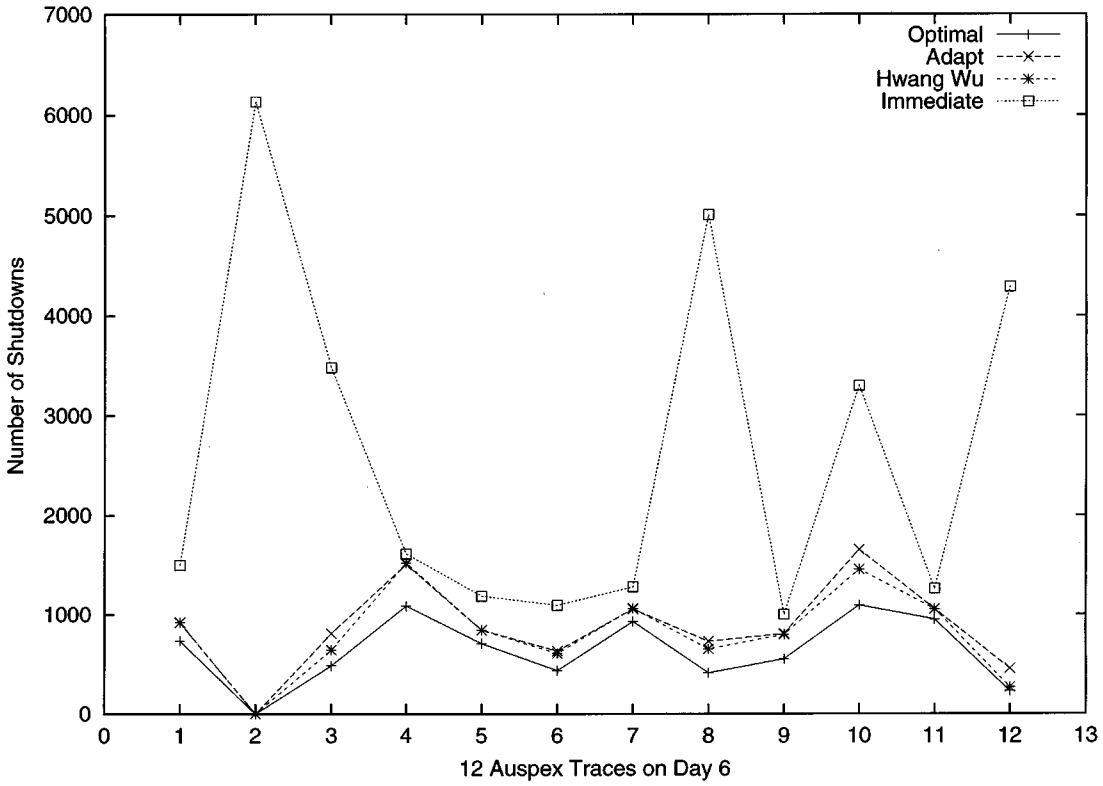

Fig. 6. Number of shutdowns of the laptop's disk drive when the Auspex server traces are applied to it. Each column shows the number of shutdowns when that particular power management algorithm is used for the disk drive.

for us to assume that T_r can assume these smaller values since the IBM disk [27] has these as the revival times for some intermediate low power idle states. The penalty of being in these low power idle states is that the power dissipated in these states is not zero, rather less than the power dissipated in the idle state.

There are two primary reasons for the success of IMMEDIATE over the other adaptive algorithms on the two traces. The first is that the patterns in the request arrival sequence which the adaptive algorithms were exploiting are no longer present when service time and power-up time are incorporated into the model. Fig. 7 shows the the sequences that ADAPT tries to exploit to reduce power and the behavior of IMMEDIATE on these sequences. Secondly, Fig. 6 shows the number of shutdowns for each of the four algorithms. Although ADAPT is making fewer mistakes than IMMEDIATE in predicting whether the upcoming idle period is greater than or smaller than k, the mistakes made by ADAPT are on average more costly than the mistakes made by IMMEDIATE and these mistakes are more expensive as T_r decreases. We discuss these two issues in more detail below.

In bursty arrival patterns, there is a sequence of very short interarrival times followed by a sequence of longer interarrival times. However, we found that when the service times and power-up times are incorporated into the model, many of the shorter interarrival times do not translate into idle times, because during bursts, requests get queued up in the system and executed one right after the other. The queue finally empties out when the burst is over and there is a longer interarrival time. *This tends to result in less correlation between the previous service time (or previous idle time) and the next idle time.*

In order to test this hypothesis, we ran ADAPT and IMMEDIATE on the same data, except that we artificially set service time and power-up time to be zero. The results are shown in Table III. Algorithm ADAPT does better on all traces. The reason that ADAPT is not doing better on the two traces when service times are modeled is explained below.

Table I shows the number of "mistakes" made by ADAPT on all the traces. That is, it shows the number of times ADAPT predicted that the next idle period would last less than time k when it actually lasted longer. In addition, it shows the number of times ADAPT predicted the next idle period would last longer than time k, but it actually lasted less time. Similar results for IMMEDIATE are shown in Table II. IMMEDIATE has only a one-sided error: by shutting down immediately, it is always in effect predicting that the idle period will be longer than k. The average wasted energy for each mistake is also given. This is the extra amount of energy expended because the algorithm predicted incorrectly. Note that the most costly kind of mistake is when the algorithm predicts that the idle period will be short when it is in fact long. Since only ADAPT makes this kind of error, it is paying a much higher price for each mistake even though it is making fewer mistakes.

Suppose an algorithm stays powered up (thinking the idle period will be short) and then powers down when the idle period has lasted k time units. Its cost is $k \cdot P_i + E_r = 2E_r$. If the algorithm had powered down, it would have expended only E_r units of energy. Thus, the wasted energy is E_r. On the other hand, if it powers down immediately and the length of the idle period is $t < k$, then the algorithm spends $E_r = k \cdot P_i$ but could have only spent $t \cdot P_i$. The extra energy expended is $(k-t)P_i$ which is

Guess \ Actual	Actual inter–arrival time is short i.e. t < k	Actual inter–arrival time is long i.e. t > k
Guess that the inter–arrival time is short i.e. t < k	Immediate dissipates 0 power before shutdown and Pr power for revival	Immediate dissipates 0 power before shutdown and Pr power for revival
	Adapt dissipates Pi till next request and is not shutdown	Adapt dissipates (k Pi) before shutdown and Pr power for revival
Guess that the inter–arrival time is long i.e. t > k	Immediate dissipates 0 power before shutdown and Pr power for revival	Immediate dissipates 0 power before shutdown and Pr power for revival
	Adapt dissipates 0 power before shutdown and Pr power for revival	Adapt dissipates 0 power before shutdown and Pr power for revival

Fig. 7. ADAPT makes guesses about the next interarrival times. Its guess is either correct or wrong. In this figure, we show the power dissipated by ADAPT when it guesses correctly as well as when it guesses incorrectly. We also compare the power dissipated by IMMEDIATE on the decisions made by ADAPT. In this figure, ADAPT tries to exploit the sequences where it guesses that the interarrival times are short. On the other hand, it dissipates a large amount of power when this guess is incorrect. As we show, when we begin modeling service times, in some sequences short interarrival times are replaced with longer interarrival times. This causes ADAPT to perform poorly as compared with IMMEDIATE on some sequences and especially when k is small.

TABLE I
STATISTICS THAT SHOW THE NUMBER OF TIMES ADAPT PREDICTED THE NEXT IDLE INTERVAL TO BE GREATER THAN k AND LESS THAN k. THE TABLE SHOWS THE NUMBER OF TIMES ADAPT WAS WRONG IN BOTH CASES AND THE ENERGY DISSIPATED DUE TO THESE WRONG DECISIONS. THE LAST COLUMN DISPLAYS THE AVERAGE POWER DISSIPATED PER WRONG DECISION MADE BY ADAPT. IN THIS CASE, T_r (REVIVAL TIME) IS ASSUMED TO BE 4 s

	Error Statistics for ADAPT					
	Predicted idle period $< k$		Predicted idle period $> k$			
Traces	Wrong	Energy Wasted in Joules	Wrong	Energy Wasted in Joules	Total Wrong Decisions	Avg Waste in Joules per wrong decision
t6.H1062	188	3534.400000	188	1461.362552	376	13.286603
t6.H1074	0	0.000000	1	15.396678	1	15.396678
t6.H2012	343	6448.400000	344	2552.818586	687	13.102210
t6.H2014	427	8027.600000	428	3250.438041	855	13.190688
t6.H2149	209	3929.200000	209	1672.747489	418	13.401788
t6.H3069	350	6580.000000	351	3554.703435	701	14.457494
t6.H3073	261	4906.800000	261	1844.375596	522	12.933287
t6.H3113	635	11938.000000	635	4269.980183	1270	12.762189
t6.H4060	111	2086.800000	111	971.663189	222	13.776861
t6.H4119	227	4267.600000	227	1652.387344	454	13.039620
t6.H4127	142	2669.600000	142	1307.189201	284	14.002779
t6.H4181	132	2481.600000	132	1159.949774	264	13.793749

typically only a fraction of E_r. This explains why the first kind of error is more costly than the second kind of error.

In Section V, we present a detailed theoretical analysis of this phenomenon under the assumption that the length of each idle period is generated independently by identical distributions. We have also discussed the details relevant to power management in an earlier section. Let T be the random variable which denotes the length of the idle time. Let $P[T = t]$ be the probability that $T = t$. For simplicity, we will assume that time is discretized. Then, if the algorithm's threshold is τ, its expected amount of energy dissipated is

$$\sum_{t=1}^{\tau} P[T=t] \cdot t \cdot P_i + \sum_{t=\tau+1}^{\infty} P[T=t](\tau \cdot P_i + E_r).$$

If the distribution is known, then the optimal online algorithm will use the equation above and choose τ to minimize the expected cost.

In order to understand why IMMEDIATE performed as well as it did, we determined the distribution based on the statistics for the idle periods of each sequence and then determined the optimal τ for each sequence. The results are given in Table IV. Each threshold is expressed as a fraction of the time interval k. What is noticeable here is that in all but a very few cases, the optimal threshold is small which indicates that IMMEDIATE is using close to the optimal threshold (traces 4 and 9). Thus, if we are restricting our attention to algorithms which use a fixed threshold, then for the data used in our study, a threshold of zero is closer to the optimal. Further, as T_r decreases, the optimal threshold decreases as well, which means that IMMEDIATE is

TABLE II
STATISTICS THAT SHOW THE NUMBER OF TIMES IMMEDIATE PREDICTED THE NEXT IDLE INTERVAL TO BE GREATER THAN k. THE TABLE SHOWS THE NUMBER OF TIMES THE DISK WAS SHUTDOWN AND THE NUMBER OF TIMES THIS DECISION WAS WRONG. THE TABLE ALSO SHOWS THE POWER DISSIPATED DUE TO THE WRONG DECISION. IN THIS CASE, T_r (REVIVAL TIME) IS ASSUMED TO BE 4 s

	Error Statistics for IMMEDIATE		
Traces	Total shutdowns	Wrong decisions	Energy Wasted in Joules per wrong decision
t6.H1062	1501	776	11.584568
t6.H1074	6136	6136	10.803558
t6.H2012	3482	3084	10.284557
t6.H2014	1612	536	7.805614
t6.H2149	1095	704	8.315458
t6.H3069	5011	4782	9.778153
t6.H3073	1002	470	7.646821
t6.H3113	3301	2313	8.912100
t6.H4060	1258	323	9.558162
t6.H4119	4289	4207	7.632398
t6.H4127	1187	494	10.914968
t6.H4181	1280	361	10.358660

TABLE III
STATISTICS THAT SHOW THE POWER DISSIPATED BY ADAPT AND IMMEDIATE WHEN SERVICE TIME AND LATENCY ARE NOT MODELED, I.E., THEY ARE CONSIDERED TO BE ZERO. WE ASSUME THAT T_r (REVIVAL TIME) IS 4 s

	Power Dissipation with Zero Service Time and Latency			
	ADAPT		IMMEDIATE	
Trace	Power dissipated	Shutdowns	Power dissipated	Shutdowns
t6.H1062	0.457607	981	10.961039	33656
t6.H1074	0.850326	1	21.971276	67470
t6.H2012	0.859521	995	10.827513	33236
t6.H2014	1.053709	2108	2.360304	7246
t6.H2149	0.512088	865	3.398084	10395
t6.H3069	1.094657	849	22.698990	69710
t6.H3073	0.579440	1090	1.585281	4868
t6.H3113	1.311975	2154	13.803080	42391
t6.H4060	0.971151	1909	2.932487	9003
t6.H4119	0.996386	535	5.144162	15797
t6.H4127	0.761427	1443	5.565346	17090
t6.H4181	0.443369	1098	4.333740	13306

using close to the optimal threshold and hence performs better than ADAPT.

Clearly, it is not always the case that IMMEDIATE outperforms ADAPT since that will depend on the parameter k relative to the input sequence. However, we have demonstrated that timing considerations can be important and should be incorporated into any model used to assess power management schemes. We have also demonstrated the effect of system parameters on the power management schemes. In order to demonstrate the effects of the parameter k on the performance of the algorithms, we varied the revival times from 4 s to 0.4 s and to 4 ms and computed the power dissipated by the various algorithms. This result is shown in Fig. 5. We see that the performance of algorithm IMMEDIATE becomes better as T_r decreases (k increases). In general, a successful power management scheme will depend on typical arrival patterns, service times, as well as system parameters.

VIII. EFFECT OF POWER MANAGEMENT ON LATENCY

In this section, we present a proof that bounds the maximum wait time for any request that enters the system. This gives us an upper bound on the latency of the system in the presence of a power management scheme for the system. Such an upper bound allows the system designer to account for latencies in the system that arise due to power management strategies for specific timing critical subsystems.

TABLE IV
BEST SHUTDOWN THRESHOLD COMPUTED FROM ALL POSSIBLE THRESHOLDS AVAILABLE FOR A PARTICULAR TRACE. THE THIRD COLUMN SHOWS THIS THRESHOLD AS A FRACTION OF k. THE LAST COLUMN PRESENTS THE POWER DISSIPATED IF THIS THRESHOLD WERE USED INSTEAD OF k AS THE SHUTDOWN THRESHOLD. IN THIS CASE, WE ASSUME THAT T_r (REVIVAL TIME) IS 4 s

	Best Shutdown Threshold		
Trace	Best Threshold in clock ticks	Best Threshold as a fraction of k	Power Dissipated by using threshold for shutdown (milliwatts)
t6.H1062	439841	0.198864	0.828485
t6.H1074	2119139	0.958121	0.732872
t6.H2012	5676651	2.566571	1.363959
t6.H2014	35938	0.016249	10.675742
t6.H2149	3013671	1.362564	2.764495
t6.H3069	6029685	2.726187	0.716727
t6.H3073	166408	0.075238	7.497255
t6.H3113	6002734	2.714002	1.116283
t6.H4060	244893	0.110723	4.559279
t6.H4119	4628491	2.092669	3.162420
t6.H4127	350080	0.158281	1.684208
t6.H4181	200443	0.090626	2.317981

A. Bound the Maximum Wait Time

Let σ be any sequence of arrivals of n requests into the system ordered according to arrival time. Since the service of these requests is on a first-come-first-served basis, we number the requests from 1 to n. Let a_j be the arrival time of request j. Let d_j denote the service time for request j. If the system is kept powered up the entire time, then the latency of each request is minimized. Suppose the following algorithm is adopted. Consider the arrival of request j at time a_j. If the system is idle at time a_j, then request j is begun immediately and incurs no latency. Alternatively, suppose that the system is busy when request j arrives and suppose that the last idle time ended with the arrival of request k. Then the system will be free to begin request j at time $a_k + \sum_{l=k}^{j-1} d_l$. The system starts request k at time a_k and must complete requests k through $j-1$ before starting request j. This means that the latency of request j denoted by $W(\sigma, j)$ is

$$W(\sigma, j) = \left(a_k + \sum_{l=k}^{j-1} d_l \right) - a_j. \tag{13}$$

Note that for any $i \neq k$ such that $i \leq j$

$$\left(a_i + \sum_{l=i}^{j-1} d_l \right) - a_j \leq \left(a_k + \sum_{l=k}^{j-1} d_l \right) - a_j. \tag{14}$$

Thus, the wait time of request j is

$$W(\sigma, j) = \max_{i \leq j} \left(a_i + \sum_{l=i}^{j-1} d_l \right) - a_j. \tag{15}$$

We call this the *inherent wait time* of request j since it is the smallest possible wait time achievable for job j by any

TABLE V

PERFORMANCE OF THE THREE ALGORITHMS AS k IS VARIED. THE VARIATION OF k IS DONE BY DECREASING T_r, THE REVIVAL TIME FROM 4 s DOWN TO 4 MS IN INTERVALS OF 0.1 s. THE PARAMETER k CAN BE REDUCED BY DECREASING E_r AS WELL, BUT WE HAVE KEPT THAT FIXED IN THIS CASE. THIS TABLE SHOWS THE CASES WHEN THE ALGORITHM IMMEDIATE PERFORMS BETTER THAN ALGORITHM ADAPT. THESE CASES ARE MORE PROMINENT WHEN k IS SMALL AND ARE INDICATED BY THE BOLD TYPEFACE. NOTICE THAT WHEN $k = 1800$, THE ALGORITHM IMMEDIATE COMPLETELY OUTPERFORMS ADAPT. THIS TABLE SHOWS THAT THE PERFORMANCE OF THE ALGORITHMS CAN VARY DEPENDING ON THE VALUE OF k: SIMPLE ALGORITHMS CAN OUTPERFORM COMPLICATED ALGORITHMS. AS A RESULT k NOW BECOMES AN IMPORTANT PARAMETER IN THE DESIGN OF THE SYSTEM SINCE IT INFLUENCES THE PERFORMANCE OF SHUTDOWN ALGORITHMS WHEN THEY ARE USED TO MANAGE POWER DISSIPATION

		Power dissipated by algorithms over varied k			
Trace	Algorithms	$T_r = 4$ secs	$T_r = 0.4$ secs	$T_r = 0.04$ secs	$T_r = 4$ msecs
		$k = 1800000$	$k = 180000$	$k = 18000$	$k = 1800$
	ADAPT	0.415675	0.076902	0.026423	**0.010249**
t6.H1062	HWANGWU	0.387899	0.075314	0.026575	0.009963
	IMMEDIATE	0.489422	0.129385	0.051770	**0.009799**
	ADAPT	0.850523	**0.345165**	0.090420	**0.021633**
t6.H1074	HWANGWU	0.850523	0.328085	0.103810	0.019517
	IMMEDIATE	1.998442	**0.332986**	0.117072	**0.019341**
	ADAPT	0.723009	**0.189156**	0.029796	0.009817
t6.H2012	HWANGWU	0.707956	0.188612	0.031543	0.009981
	IMMEDIATE	1.135051	**0.179658**	0.044123	0.010028
	ADAPT	**0.644899**	**0.058420**	0.007886	**0.001949**
t6.H2014	HWANGWU	0.588159	0.057169	0.007826	0.001808
	IMMEDIATE	**0.525061**	**0.058020**	0.008580	**0.001786**
	ADAPT	0.357596	0.060482	0.017480	**0.006314**
t6.H4127	HWANGWU	0.357999	0.060197	0.017659	0.005824
	IMMEDIATE	0.387226	0.079644	0.026648	**0.005555**
	ADAPT	0.344162	**0.058943**	0.010101	**0.002814**
t6.H2149	HWANGWU	0.348560	0.057288	0.010081	0.002681
	IMMEDIATE	0.358022	**0.057139**	0.012452	**0.002610**
	ADAPT	0.412118	0.054168	0.008870	**0.002999**
t6.H4181	HWANGWU	0.407473	0.053066	0.008862	0.002928
	IMMEDIATE	0.416951	0.061512	0.014951	**0.002748**
	ADAPT	0.977732	0.291366	0.065186	**0.020864**
t6.H3069	HWANGWU	0.962464	0.281681	0.066539	0.019689
	IMMEDIATE	1.632077	0.324674	0.093377	**0.018982**
	ADAPT	**0.375958**	**0.045307**	**0.008469**	**0.001818**
t6.H3073	HWANGWU	0.359632	0.043941	0.007542	0.001671
	IMMEDIATE	**0.326550**	**0.044165**	**0.008266**	**0.001642**
	ADAPT	0.965757	0.163703	0.041242	**0.015232**
t6.H3113	HWANGWU	0.957095	0.159952	0.041993	0.014283
	IMMEDIATE	1.076102	0.218932	0.068350	**0.013656**
	ADAPT	0.405936	0.055853	0.009058	**0.002668**
t6.H4060	HWANGWU	0.402309	0.055254	0.009252	0.002582
	IMMEDIATE	0.409773	0.057108	0.011852	**0.002536**
	ADAPT	0.930934	0.177891	0.023093	**0.005275**
t6.H4119	HWANGWU	0.909331	0.172831	0.022870	0.004845
	IMMEDIATE	1.396658	0.178235	0.026991	**0.004745**

power management scheme. The maximum inherent wait time incurred by any request in σ will be called *the inherent wait time of sequence* σ and will be denoted by $W(\sigma)$.

Part of the problem definition states that any algorithm must keep the machine on as long as there are requests to service. Therefore, we consider only the algorithms that consider powering down the system only during an idle period.

Lemma 3: The wait time of any power management algorithm on any sequence σ is at most $W(\sigma) + T_r$. T_r is called the revival time for the system.

Proof of Lemma 3: Fix an arbitrary algorithm A. Consider request j in σ. Suppose that when j arrives, the previous idle period ended with the arrival of request k. If the algorithm A did not power down during this period, then the amount of time that request j waits is just

$$\left(a_k + \sum_{l=k}^{j-1} d_l\right) - a_j = W(\sigma, j). \tag{16}$$

If the algorithm A did power down during this time period, then the amount of time that request j waits is

$$T_r + \left(a_k + \sum_{l=k}^{j-1} d_l\right) - a_j \leq T_r + W(\sigma, j). \tag{17}$$

Since this is true for every request j, it follows that no request waits longer than $W(\sigma) + T_r$. ■

Next we prove that under a worst case analysis, an algorithm can always be **forced** to have a request which must wait an additional time T_r as shown in the following lemma.

Lemma 4: Let A be any deterministic power management algorithm whose energy dissipation is some finite though variable value. Given a sequence σ with an inherent wait time W, when A is used on σ, there is a sequence σ' whose inherent wait time is W for which A has a request which is delayed by $W + T_r$.

Proof of Lemma 4: Pick an arbitrary algorithm A. Suppose that after a period of time τ, the algorithm will shut down the system if no requests have arrived. (τ cannot be infinite since the energy dissipation is finite). Pick any sequence σ such that $W(\sigma) = W$. Let j be a request such that $W(\sigma, j) = W$. Suppose that under the algorithm which nevers turns the system off the idle period previous to j's arrival ends with the arrival of request k. Make a new sequence σ' of $j - k + 1$ requests. The arrival time of request i in the new sequence is denoted by a'_i and will be $\tau + a_{i+k-1} - a_k$ from the old sequence. The duration of request i in the new sequence is denoted by d'_i and is d_{i+k-1} from the old sequence. Since the algorithm A powers down after time τ, the delay of request $j - k + 1$ in σ' will be

$$\left(T_r + \sum_{l=1}^{j-k+1} d'_j\right) - a'_{j-k+1} \tag{18}$$

$$= T_r + a_k + \sum_{l=k}^{j} d_l - a_j \tag{19}$$

$$= T_r + W(\sigma, j) = T_r + W(\sigma). \tag{20}$$

Thus we have constructed a new sequence σ', from an existing sequence σ, such that σ' has an inherent wait time of W and at least one request in σ' will be delayed by $W + T_r$. ■

Theorem 3: In the presence of a power management algorithm A, that shuts down a system when it is idle and revives it when needed for service, no request in any arrival sequence σ will be delayed by more than $W + T_r$ time units. T_r is the time it takes to revive the system from a shutoff state to a state where it can service the request and W is the inherent wait time of σ on A.

Proof of Theorem 3: Lemmas 3 and 4 prove this theorem. ■

IX. Conclusion

We have used competitive analysis to analyze previous "heuristic" algorithms for power management of embedded systems. We present a simple nonadaptive algorithm that is 2-competitive and optimal. We have proved that the lower bound on the competitive ratio for any adaptive online algorithm attempting to solve the power management problem is 1.582. We concluded that the simple 3-competitive adaptive algorithm can be used in both hardware and software systems and its results are guaranteed. The analysis of adaptive algorithms demonstrates that designers have to commit greater resources to power critical subsystems so that aggressive power management algorithms can be used for them.

In this paper, we have presented an upper bound on the increase in latency of the system in the presence of a power management algorithm. This upper bound allows a system designer to get a handle on the worst case increase in latency he may encounter when determining a power management system for timing critical subsystems. Since this latency is a function of the revival time, the system designers can make design decisions that can shorten the revival time for subsystems for which aggressive power management may be required.

We have also presented a performance comparison between two algorithms, ADAPT and IMMEDIATE, on a set of trace data [25] applied to the disk of a hard drive [27]. Our experimental results show that if service times of arriving requests are modeled, the relative performance of algorithms can change. This change is even more prominent when the revival time changes as well. We show that the simple algorithm that shuts down the system whenever it encounters an idle period surprisingly performs better than adaptive algorithms suggested in the literature on the data used here, especially when k is small. We provide an analytical explanation for this empirical result. However, IMMEDIATE's better performance comes at the cost of increased system latency. This paper demonstrates the power–latency tradeoff and provides insight into the effects of modeling service time on the power dissipation algorithm.

Note that it is not always the case that timing considerations will have such an important impact on the performance of power management strategies. This will depend on the distribution of interarrival times of requests as well as service times. In addition, IMMEDIATE outperforms ADAPT as the parameter k decreases relative to the input sequence. However, we have demonstrated that timing considerations can be important and should be incorporated into any model used to assess power management schemes. We have also demonstrated the effect of system parameters on the power management schemes. In general, a successful power management scheme will depend on typical arrival patterns, service times, as well as system parameters.

This paper leaves many open questions as well, for instance, the conditions where adaptive strategies are likely to be beneficial. We plan to explore shutdown strategies in other embedded applications, including real-time operating systems (at the task level), web servers, networked embedded systems, etc.

References

[1] C.-H. Hwang and A. C.-H. Wu, "A predictive system shutdown method for energy saving of event-driven computation," in *IEEE/ACM Int. Conf. Computer Aided Design*, Nov. 1997, pp. 28–32.

[2] G. A. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli, "Policy optimization for dynamic power management," in *Proc. 35th Design Automation Conf.*, June 1998, pp. 182–187.

[3] D. Ramanathan, "High-Level timing and power analysis for embedded systems ," Ph.D. dissertation, Univ. California, Irvine, 2000.

[4] D. Ramanathan and R. Gupta, "System level online power management algorithms," in *Proc. Design Automation and Test in Europe Conf.*, Paris, France, Mar. 2000.

[5] D. Ramanathan, S. Irani, and R. Gupta, "Latency effects of system level power management algorithms," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, Nov. 2000.

[6] M. B. Srivastava, A. P. Chandrakasan, and R. W. Broderson, "Predictive shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Trans. Very Large Integration Syst.*, vol. 4, pp. 42–54, Mar. 1996.

[7] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, "Competitive randomized algorithms for nonuniform problems," *Algorithmica*, vol. 11, no. 6, pp. 542–571, June 1994.
[8] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator, "Competitive snoopy caching," *Algorithmica*, vol. 3, no. 1, pp. 70–119, 1988.
[9] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Trans. Design Automation Electron. Syst.*, vol. 1, no. 1, pp. 3–56, Jan. 1996.
[10] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proc. 32nd Design Automation Conf.*, 1995, pp. 242–247.
[11] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 2, pp. 446–455, Dec. 1994.
[12] J. L. Peterson and A. Silberchatz, *Operating Systems Concepts*, 2nd ed. Reading, MA: Addison-Wesley, pp. 118–120.
[13] R. El-Yaniv, R. Kaniel, and N. Linial, On the equipment rental problem.
[14] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*. New York: Kluwer, 1995.
[15] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," in *Proc. Design Automation Conf.*, June 1999, pp. 555–561.
[16] Q. Qui, Q. Wu, and M. Pedram, "Stochastic modeling of a power-managed system: Construction and optimization," in *Proc. Int. Symp. Low Power Electronics and Design*, 1999.
[17] Y. H. Lu, E. Y. Chung, T. Simunic, L. Benini, and G. De Micheli, "Quantitative comparison of power management algorithms," in *Proc. Design Automation and Test in Europe (DATE 2000)*, Mar. 2000, pp. 20–26.
[18] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive algorithms for distributed data management," in *Proc. 24th Symp. Theory of Computation*, 1992, pp. 39–48.
[19] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, "Competitive analysis of financial games," in *Proc. 33rd Ann. Symp. Foundations Computer Sci.*, 1992, pp. 327–333.
[20] R. El-Yaniv and R. M. Karp, "The mortgage problem," in *2nd Israel Symp. Theory Computing Syst.*, June 1993.
[21] P. Raghavan, "A statistical adversary for on-line algorithms," *DIMACS Series in Discrete Math. Theoretical Computer Sci.*, vol. 7, pp. 79–83, 1991.
[22] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, pp. 202–208, Feb. 1985.
[23] ——, "Self-adjusting binary search trees," *J. ACM*, vol. 32, no. 3, pp. 652–686, July 1985.
[24] S. Udani and J. Smith, "The Power Broker: Intelligent power management for mobile computing," Dept. Computer and Information Science, Univ. Pennsylvania, Technical Report MS-CIS-96-12, 1996.
[25] (1993) Auspex file traces from the NOW project. [Online]. Available: http://now.cs.berkeley.edu/Xfs/AuspexTraces/auspex.html
[26] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. New York: Kluwer, 1997.
[27] (1996) Technical specifications of hard drive IBM travelstar VP 2.5 inch. [Online]. Available: http://www.storage.ibm.com/storage/oem/data/travvp.htm

Dinesh Ramanathan received the B.E degree in computer science and the M.S. degree in mathematics from the Birla Institute of Technology and Science, Pilani, India. He received the M.S. and Ph.D degrees in information and computer science from the University of California, Irvine, in 2000.

He is Director of Business Management at Raza Foundries, a venture capital company. He oversees the management of various portfolio companies building integrated circuits for the internet infrastructure. Prior to Raza Foundries, he was Director of Engineering at CynApps Inc., responsible for designing state-of-the-art synthesis compilers and verification tools using C++ as the hardware and system description language. Prior to joining CynApps, he spent three years at Synopsys Inc., designing and implementating Verilog and C compilers. As a graduate student, he spent a summer at the NASA Goddard Space Flight Center in Maryland within the Space Data and Computing Division. His research interests include timing and power optimizations for embedded systems, VLSI CAD algorithms, and on-line algorithms.

Sandra Irani received the BSE degree in electrical engineering at Princeton University in 1986. In 1991, she received the Ph.D. degree in computer science from the University of California, Berkeley, under the supervision of Richard Karp.

In the 1991/1992 academic year, she was a University of California President's Postdoctoral Fellow at the University of California, San Diego. She also spent part of the year at Princeton University as a postdoc at the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS). In autumn of 1992, she joined the faculty of University of California, Irvine, where she is currently an Associate Professor. Her research intertests include design and analysis of algorithms, in particular, on-line algorithms and their applications to scheduling and resource allocation.

Rajesh K. Gupta received the M.S. degree from Stanford University and the Ph. D. degree from the Unversity of Calfornia, Berkeley.

He is an Associate Professor of Information and Computer Science at University of California, Irvine. He also maintains an active interest in broad-band communication systems. He worked as an Assistant Professor at University of Illinois, Urbana-Champaign, from 1994 through 1996. Prior to that he was at Intel Corporation in Santa Clara, CA, where he worked as a member on a number of processor design teams. He is co-author of three patents. He is author of the book *Co-synthesis of Hardware and Software for Digital Embedded Systems* (New York: Kluwer, 1995). At UCI, he leads an effort on adaptive memory system architectures and co-leads an effort on compiler-controlled power/performance management. His research interests include system-level design for embedded and portable systems, VLSI design, and adaptive system architectures.

Dr. Gupta is a recipient of the UCI Chancellor's Award for excellence in undergraduate research, a National Science Foundation Career Award, two Departmental Achievement Awards, and a Components Research Team Award at Intel. He serves as Chair of the CANDE technical committee and on the board of governors of the IEEE Circuits and Systems Society. He also serves as Associate Editor-in-Chief of IEEE DESIGN AND TEST and on the editorial board of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN. He is a distinguished lecturer for the ACM/SIGDA and the IEEE CAS Society.