Latency Effects of System Level Power Management Algorithms*

Dinesh Ramanathan Sandy Irani Rajesh Gupta
Department of Information and Computer Science
University of California
Irvine, CA 92697
{dinesh,irani,rgupta}@ics.uci.edu

Abstract

A power management algorithm for an embedded system reduces system level power dissipation by shutting off parts of the system when they are not being used and turning them back on when they are required. Algorithms for this problem are online in nature since they must operate without knowledge of the arrival time or service requirements of future requests. In this paper, we present online algorithms to manage power for embedded systems. We perform an empirical analysis of these algorithms and give theoretical justification for the empirical results. Effective power management strategies have an adverse impact on the latency of the system for which the strategy is designed. Typically, the more aggressive the power management scheme, the greater the increase in the latency of the system. In this paper, we prove an upper bound on the additional latency of the system introduced by power management strategies. Moreover, we show that this upper bound occurs each time the system is shutdown and hence is an important system design parameter.

In addition, service time and latencies have an effect on power management strategies since they alter the length and occurrences of idle periods which. We study this phenomenon experimentally, by modeling the disk drive of a laptop computer as an embedded system. The results show that if service times of arriving requests are modeled, the relative performance of algorithms can change leading to non-adaptive algorithms performing better than adaptive ones. We compare the performance of adaptive and non-adaptive power management algorithms. In particular, our experimental results show that an "immediate" shutdown strategy that shuts down the system whenever it encounters an idle period performs surprising better than sophisticated adaptive algorithms suggested in the literature. We provide an analytical explanation for the effectiveness of power management strategies.

1 Introduction

Power dissipation in a 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 [6], Devadas and Malik [7] and Najm [8].

Low power VLSI design can be achieved at various levels of abstraction during the design process. These include the system level, behavioral level, the RTL level, and the gate level. Most the techniques in the literature are focused at the RTL level. This paper focuses on the system level where little prior work has been done. Furthermore, there has been little work to measure how latency and power management schemes effect each other. The notion of system level design is described next.

An embedded system (the system for short) is typically reactive and real-time in nature: it 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 inter-arrival time between requests is unknown to the algorithm. In order to determine the optimal power management policy, it is necessary to know the sequence of inter-arrival times of requests, the time to service each request and some internal parameters of the system. During system level design, this information is not known. However, 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 the strategies that selectively shutdown subsystems and turn them back on when needed as well as the effect of such strategies on the latency of the system. Typically, power management is needed for subsystems that are timing critical. As a result, 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.

This paper makes three contributions. The first is an analytical upper bound on the increase in the latency of the system due to power management. We prove that the maximum latency of the system is not increased by more than the time it takes to revive the system from the shut off state. Furthermore, this upper bound is always achieved whenever the system is powered down. As the second contribution, we introduce service times and latency into the model for power management algorithms. Most previous studies ignore these factors. We find that when service times are incorporated into the model, a simple algorithm, called IMMEDIATE, does better than previously proposed adaptive algorithms on the data used in our study. Algorithm IMMEDIATE simply shuts off the system whenever an idle period is encountered. The third contribution is an analysis of this behavior which provides theoretical justification for why IMMEDIATE performs better in our study than previously suggested algorithms. This leads to the conclusion that non-adaptive algorithms perform better on distributions whose inter-arrival times are locally uniform. We model a laptop computer's hard drive as an embedded system and apply previously proposed adaptive algorithms [12, 1] as well as our simple algorithm to manage power dissipation for this system.

1.1 Background and Previous Work

A power management scheme is specified by a series of thresholds: one for each power level, except the powered-down state. Each threshold determines how long the system

*Sandy Irani would like to acknowledge support provided by NSF grant CCR-9309456. Dinesh Ramanathan and Rajesh Gupta would like to acknowledge support provided by DARPA/ITO PAC/C Program Contract Number F33615-00-C-1632.

will remain in that state before it transitions to the next lowest power level or until a new request for service arrives. In this paper, we only consider two levels of power where the system is either powered-up or powered-down. Thus, for each idle period, the algorithm is governed by a single threshold. However, for systems that can *sequentially* move from a threshold with higher power dissipation to one with lower power dissipation, the same algorithms can be applied by changing the system parameters. On the other hand, for systems that can move between any arbitary threshold state, we believe that a randomized algorithm will be more useful and we leave that problem open in this paper.

There are essentially two kinds of power management strategies: non-adaptive and adaptive. For a non-adaptive algorithm, the threshold is statically computed based on the properties of the system and does not change when the requests are received. In adaptive algorithms, the time after which the system shuts down is determined dynamically as the system is servicing requests. Intuitively, we would expect adaptive algorithms to perform better than non-adaptive ones since they have the ability to learn and exploit patterns in the arrival sequence of requests. Using heuristic algorithms, all previous work has experimentally shown that this intuition holds.

This paper builds upon earlier works on power management strategies [3, 1, 2, 12]. Srivastava et al. [3] conducted an extensive analysis on different system shutdown approaches. They have proposed an adaptive shutdown algorithm for power saving of event-driven systems. They 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 [9] used in the CPU scheduling problem for the prediction of idle periods. They proposed an algorithm using two new strategies: prediction-miss correlation and pre-wakeup. 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. Furthermore, their algorithm may not be adapted to hardware systems since it requires expensive computation resources that may not be available.

More recently, Paleologo, Benini 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 [16] and using the Auspex file traces [14], they generate a Markov model using an exponential distribution for the arrival of the requests and to model the service times for each request. This work does incorporate service time into the model. Moreover, their assumption that the inputs are exponentially distributed is not fully justified and may not always hold in view of significant correlation between disk 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.

None of these papers, analyze the effects of power management on the latency of the system. The authors in [1] consider latency to be a metric that their algorithm optimizes. However, they only provide experimental results on the latency of the system they use to illustrate their algorithms. We believe that the effects of latency on the system's performance is a crucial decision that the system designer must make 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.

The authors in [12] have adapted the results from [5, 4] to the power management problem. They present an optimal non-adaptive algorithm and an adaptive algorithm that in the worst case, dissipates three times the power dissipated by the optimal offline algorithm which has complete knowledge of all future requests for service. However, here too, the authors do not model service time for a request. As a result, their experiments show that simple adaptive algorithms perform as well as complex adaptive algorithms presented by Hwang and Wu [1]. We have modified the adaptive algorithms presented in [12] to model service time for a given request. We show that this alters the way the adaptive algorithms perceive the input sequence and degrades their performance. As a result, we present a simpler non-adaptive algorithm that performs better than the adaptive algorithms.

2 Problem Definition

Given an embedded system, design a power management strategy for the system. The system 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. Requests are handled in a first-come-first-served manner. The system must be working on some request 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.

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 shut off 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 $c = \frac{E_r}{P_i}$. For the sake of simplicity we assume that $c = \lceil \frac{E_r}{P_i} \rceil$ The effects of this assumption are explained in detail in [13].

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 decisions making process is 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.

3 Effect of Power Management on Latency

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

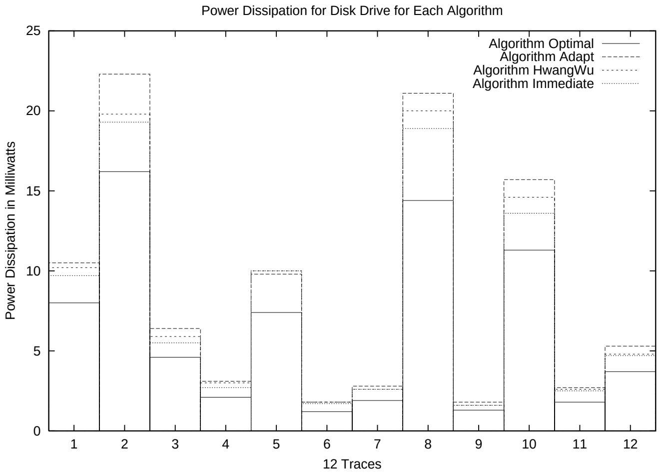

Figure 1: *Power dissipation in watts of the laptop's disk drive when the Auspex server traces are applied to it. Each column shows the power dissipated when that particular power management algorithm is used for the disk drive. Immediate's power dissipation for trace 5 is more than Adapt, whereas it is lower on traces 7 and 9. On traces 5. 7 and 9, HwangWu coincides with Immediate. We present these since the differences between the power dissipation is quite small and maynot be clear on the graph.*

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

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

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

$$\left(a_i + \sum_{l=i}^{j-1} d_l \right) - a_j \leq \left(a_k + \sum_{l=k}^{j-1} d_l \right) - a_j. \quad (2)$$

Thus, the wait time of request j is

$$W(\sigma, j) = \max_{i \leq j} \left(a_i + \sum_{l=i}^{j-1} d_l \right) - a_j. \quad (3)$$

We call this the *inherent wait time* of request j since it is the smallest possible wait time achievable for job j by any 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 1 *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.*

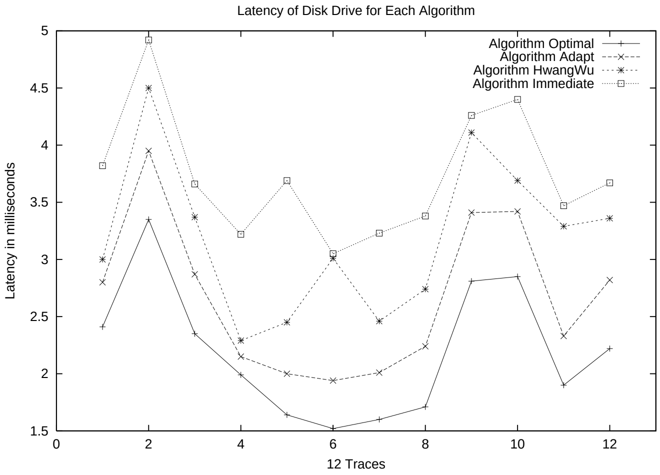

Figure 2: *Increase in latency (in secs) of the laptop's disk drive when the Auspex server traces are applied to it. Each column shows the increase in latency when that particular power management algorithm is used for the disk drive.*

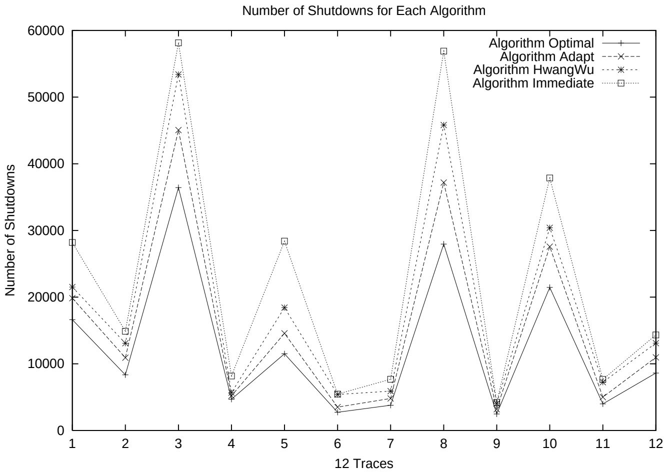

Figure 3: *Number of shutdowns of the laptop's disk drive when the Auspex server traces are applied to it. Each column shows the number of shutdowns when that particular power management algorithm is used for the disk drive.*

Proof of Lemma 1. 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

$$\left(a_k + \sum_{l=k}^{j-1} d_l \right) - a_j = W(\sigma, j) \quad (4)$$

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

$$T_r + \left(a_k + \sum_{l=k}^{j-1} d_l \right) - a_j \leq T_r + W(\sigma, j) \quad (5)$$

Since this is true for every request j, it follows that no requests 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 2 *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 2. 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 denote 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

$$\left(T_r + \sum_{l=1}^{j-k+1} d'_j\right) - a'_{j-k+1} \tag{6}$$

$$= T_r + a_k + \sum_{l=k}^{j} d_l - a_j \tag{7}$$

$$= T_r + W(\sigma, j) = T_r + W(\sigma) \tag{8}$$

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 1 *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 shut off state to a state where it can service the request and W is the inherent wait time of σ on A.*

Proof of Theorem 1. Lemma 1 and Lemma 2 prove this theorem. ∎

This result may seem intuitive, however, this is the first formal proof presented to validate the intuition. In addition, this proof is constructive in that it specifies how a worst case sequence could be generated that would force any shutdown algorithm to incur the latency. Further, this worst case bound is not esoteric, it occurs whenever the system is shutdown and therefore provides a metric that system designers should consider while designing the system.

4 Adaptive Power Management Algorithms

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 have attempted to learn the distribution of idle periods as time progresses and optimize the threshold based on the estimate of this distribution [5, 4].

```
Algorithm ADAPT:
τ = curr_arrival_time - prev_arrival_time
              + service_time;
if (idle_state) then
    idle_intervals = idle_intervals + 1;
if (τ ≥ c) then
    shutdown
else
    if (idle_intervals ≥ c) then
        shutdown
```

Figure 4: *A Simple Adaptive Algorithm For Power Management. This algorithm uses the previous inter-service time to predict the next idle interval.*

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 c derived in Section 2. Recall that c is the length of time such that if the system is idle for time c, 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 were going to be shorter or longer than c, it could perform optimally: if the interval is less than c, 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 c, stay powered up, if it is greater than c then power down immediately. If however, the algorithm chooses to stay powered up and finds that the idle period has lasted time c, the algorithm then powers down. This latter step is important in case that 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 Figure 4, uses the length of the last idle period to predict the length of the next idle period. This algorithm was introduced in [5, 4] and adapted in [12] for power management in embedded systems. The algorithm presented by Huang and Wu [1] predicts that the length of the next idle period is an exponentially weighted average of the lengths of the previous idle periods. None of these studies incorporates service time or power-up latency into their models. That is, they assume that these values are zero.

Inter-arrival time is defined as the time between two consecutive arrivals in a sequence. We denote inter-arrival time between requests j and $j+1$ as $I_j = a_{j+1} - a_j$. Inter-service 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 zero. In this case, the length of the last idle period is the same as the last inter-service time which is also the same as the last inter-arrival 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 inter-service time since that value seems to capitalize the most on correlations in the arrival sequence.

Traces	Error Statistics for IMMEDIATE	
	Wrong decisions	Energy Wasted in microjoules per wrong decision
t6.H1062	12289	7067.65
t6.H1074	22311	6479.40
t6.H2012	18205	7070.09
t6.H2014	2762	9347.30
t6.H2149	4065	8161.47
t6.H3069	29888	7450.14
t6.H3073	1803	7400.29
t6.H3113	17762	5897.85
t6.H4060	3731	9861.81
t6.H4119	5992	7719.42
t6.H4127	6921	5522.84
t6.H4181	3686	7686.84

Table 1: *Statistics that show the number of times* IMMEDIATE *predicted the next idle interval to be greater than c. The table shows the number of times it was correct and wrong. The table also shows the power dissipated due to the wrong decision.*

5 Experimental Results

We use the disk drive [16] in a laptop as an embedded system. We have obtained traces for the use of an Auspex File Server [14] 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 disk drive has the following power characteristics. Its internal clock works 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 $T_r = 4$ milliseconds. Using these parameters, we modeled and simulated the following scenarios to compute the power dissipated:

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 c time units. An oracle will keep the system idle if the idle period is less than c 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.

Algorithm IMMEDIATE : This algorithm shuts down the system whenever it sees an idle period, however small and incurs the overhead cost of revival.

Algorithm Adapt : This algorithm uses the previous inter-service, τ, to predict the next idle period. If this value is greater than c it shuts down immediately assuming that the next idle time will be greater than c time units. If τ is less than c, it keeps the system idle for a period of c unless a new request arrives.

Algorithm HwangWu : This algorithm is an adaptation of Hwang and Wu's algorithm [1]. It uses a cumulative average, τ, of the inter-service times to predict the next idle period. If τ is greater than c, it shuts the system off as soon as an idle period is seen, otherwise it keeps the system idle for c time units before shutting down the system.

5.1 Discussion

Figure 1 and Figure 2 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 milliseconds which is the analytical upper bound on the increase in system latency. We note that IMMEDIATE outperforms the other algorithms in terms of power dissipation. However, its effect on latency is worse than OPTIMAL and ADAPT.

There are two primary reasons for the success of IMMEDIATE over the other adaptive algorithms. This 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. Secondly, Figure 3 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 c, the mistakes made by ADAPT are on average more costly than the mistakes made by IMMEDIATE. We discuss these two issues in more detail below.

In bursty arrival patterns, there are a sequence of very short inter-arrival times followed by a sequence of longer inter-arrival times. However, we found that when the service times and power-up times are incorporated into the model, many of the shorter inter-arrival 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 inter-arrival 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 3. Although ADAPT does not beat IMMEDIATE on all trace files, it uses less power on five out of the twelve traces (instead of only one out of the twelve traces when timing considerations are incorporated). The reason that ADAPT is not doing even better is explained below.

Table 2 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 c when it actually lasted longer. In addition, it shows the number of times ADAPT predicted the next idle period would last longer than time c, but it actually lasted less time. Similar results for IMMEDIATE are shown in Table 1. IMMEDIATE has only one-sided error: by shutting down immediately, it is always in effect predicting that the idle period will be longer than c. 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 c time units. Its cost is $c \cdot P_i + E_r = 2E_r$. 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 $t < c$, then the algorithm spends $E_r = c \cdot P_i$ but could have only spent $t \cdot P_i$. The extra energy expended is $(c-t)P_i$ which is typically only a fraction of E_r. This explains why the first kind of error is more costly than the second kind of error.

The authors in [5, 4] give a more detailed analysis of this phenomenon under the assumption that the length of each idle period is generated independently by identical distributions. 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

$$\sum_{t=1}^{\tau} P[T = t] \cdot t \cdot P_i \; + \sum_{t=\tau+1}^{\infty} P[T = t](\tau \cdot P_i + E_r) \qquad (9)$$

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 4. Each threshold is expressed as a fraction of the time interval c. What is noticeable here is that in all but a very few cases (where immediate does not perform well), the optimal threshold is very small which indicates that IMMEDIATE is using close to the optimal threshold. 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.

6 Conclusions

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 system designer to get a handle on the worst case increase in latency they 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 [14] applied to the disk of a hard drive [16]. Our experimental results show that if service times of arriving requests are modeled, the relative performance of algorithms can change. 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. 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 inter-arrival times of requests as well as service times. In addition, it will not always be the case that IMMEDIATE will outperform ADAPT since that will depend on the parameter c 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] Chi-Hong Hwang, Allen C.-H. Wu. A Predictive System Shutdown Method For Energy Saving of Event-Driven Computation. *IEEE/ACM International Conference on Computer Aided Design*, Nov 1997, pages 28-32.

[2] G. A. Paleologo, L. Benini, A. Bogliolo, G. De Micheli. Policy Optimization for Dynamic Power Management. *Proc. of 35th Design Automation Conference*, pp.182-187, June 1998

[3] M. B Srivastava, A. P. Chandrakasan, R. W. Broderson. Predictive Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation. *IEEE Trans. on VLSI Systems*, vol. 4, no. 1, pp.42-54, M arch 1996

[4] Karlin A. R., Manasse M.S., McGeoch L.A., Owicki S. Competitive Randomized Algorithms for Nonuniform Problems. *Algorithmica*, vol. 11, no 6, pp 542-571, June 1994

[5] A. R. Karlin, M. S. Manasse, L. Rudolph, and D.D. Sleator. Competitive Snoopy Caching. *Algorithmica*, 3(1):70–119, 1988.

[6] M. Pedram. Power Minimization in IC Design: Principles and Applications. *ACM Trans. on Design Automation of Electronic Systems*, vol 1, no. 1, pages 3-56, January 1996

[7] S. Devadas, S. Malik. A Survey of Optimization Techniques Targeting Low Power VLSI Circuits. Proc. of the 32nd Design Automation Conference, pages 242-247 , 1995

[8] F. N. Najm. A Survey of Power Estimation Techniques in VLSI Circuits. *IEEE Trans. of VLSI Systems*, vol 2, no 4, pp 446-455, D ecember 1994

[9] J. L. Peterson, A. Silberchatz. Operating Systems Concepts. 2nd Ed, pp.118-120, Addision-Wesley Publishing Co. Inc.

[10] R. El-Yaniv, R. Kaniel, N. Linial. On the Equipment Rental Problem. Manuscript.

[11] D.D. Sleator, R.E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, Vol. 32, No. 3, pages 652–686, Jul y 1985.

[12] D. Ramanathan and R. Gupta System Level Online Power Management Algorithms In Proceedings of the *Design Automation and Test in Europe* Conference Paris, March 2000

[13] D. Ramanathan and R. Gupta On System Level Online Power Management Algorithms *Technical Report*, University of California, Irvine, 2000.

[14] Auspex File Traces from the NOW project, available at http://now.cs.berkeley.edu/Xfs/ AuspexTraces/auspex.html (1993)

[15] L. Benini and G. De Micheli. Dynamic Power Management: Design Techniques and CAD Tools, Kluwer, 1997

[16] Technical specifications of hard drive IBM Travelstar VP 2.5inch, available at http://www.storage.ibm.com/storage/oem/data/travvp.htm (1996)

	Error Statistics for ADAPT					
	Predicted idle period $< c$		Predicted idle period $> c$			
Traces	Wrong	Energy Wasted in Joules	Wrong	Energy Wasted in Joules	Total Wrong Decisions	Avg Waste in **microjoules** per wrong decision
t6.H1062	4793	90.10	4793	30.96	9586	12630.06
t6.H1074	11460	215.44	11460	81.27	22920	12945.87
t6.H2012	4920	92.49	4920	34.17	9840	12872.95
t6.H2014	1315	24.72	1315	11.93	2630	13938.85
t6.H2149	1755	32.99	1755	14.38	3510	13498.89
t6.H3069	13494	253.68	13494	98.29	26988	13042.22
t6.H3073	966	18.16	966	6.68	1932	12860.41
t6.H3113	8433	158.54	8434	50.01	16867	12364.60
t6.H4060	1675	31.49	1676	16.69	3351	14379.95
t6.H4119	3008	56.55	3008	25.33	6016	13611.62
t6.H4127	3533	66.42	3534	20.17	7067	12253.19
t6.H4181	1647	30.96	1647	13.69	3294	13558.34

Table 2: *Statistics that show the number of times Adapt predicted the next idle interval to be greater than c and less than c. The table shows the number of times it was wrong in both cases and the power dissipated due to this wrong decision. The last column displays the average power dissipated due to a mistake performed by Adapt 1.*

	Power Dissipation with Zero Service Time and Latency			
	ADAPT		IMMEDIATE	
Trace	Power dissipated	Shutdowns	Power dissipated	Shutdowns
t6.H1062	**10.72**	**23145**	**10.96**	**33656**
t6.H1074	24.16	51736	21.97	67470
t6.H2012	**10.00**	**18640**	**10.82**	**33236**
t6.H2014	**02.10**	**4145**	**02.36**	**7246**
t6.H2149	**03.05**	**5947**	**03.39**	**10395**
t6.H3069	23.17	45402	22.69	69710
t6.H3073	01.70	3671	01.58	4868
t6.H3113	15.13	32547	13.80	42391
t6.H4060	02.96	5998	02.93	9003
t6.H4119	05.51	12028	05.14	15797
t6.H4127	06.04	12855	05.56	17090
t6.H4181	**03.59**	**7174**	**04.33**	**13306**

Table 3: *Statistics that show the power dissipated in milliwatts by* ADAPT *and* IMMEDIATE *when service time and latency are not modeled. This implies that they are considered to be zero.* **Notice that ADAPT performs better than IMMEDIATE only in 5 of the traces. These cases are highlighted.**

	Best Shutdown Threshold		
Trace	Best Threshold in clock ticks	Best Threshold as a fraction of c	Power Dissipated by using threshold for shutdown (milliwatts)
t6.H1062	15	0.006782	18.881
t6.H1074	18	0.008138	18.899
t6.H2012	1956	0.884362	18.085
t6.H2014	787	0.355824	15.470
t6.H2149	787	0.355824	16.834
t6.H3069	391	0.176782	18.541
t6.H3073	58	0.026223	19.231
t6.H3113	10	0.004521	18.870
t6.H4060	397	0.179495	17.029
t6.H4119	28	0.012660	19.016
t6.H4127	25	0.011303	18.774
t6.H4181	391	0.176782	15.907

Table 4: *The best shutdown threshold computed from all possible thresholds available for a particular trace. The third column shows this threshold as a fraction of c. The last column presents the power dissipated if this threshold were used instead of c as the shutdown threshold.*