System Level Online Power Management Algorithms [*]

Dinesh Ramanathan [†] Rajesh Gupta
Center for Embedded Computer Systems
Department of Information and Computer Science
University of California
Irvine, CA 92697
{dinesh,rgupta}@ics.uci.edu

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 they are required. Algorithms for this problem are online in nature where the algorithm must operate without access to the complete data set or its characteristics. In this paper, we present online algorithms to manage power for embedded systems and provide experimental analysis to back up the theoretical results.

Specifically, this paper makes four contributions. We propose an optimal online algorithm for power management. We present an analysis of algorithmic efficiency using a technique called competitive analysis which is particularly suitable for online algorithms. Using this analysis technique, we develop a lower bound for the non-adaptive version of the power management problem and show that our algorithm achieves this lower bound. Next, we explore adaptive algorithms that try to shut down the system based on historical data. We provide a lower bound for any algorithm that uses adaptive methods to manage power. We also propose an algorithm that is independent of the input data distribution, practical and usable in both hardware and software systems with guaranteed performance. Finally, we compare these algorithms with previously proposed heuristics both theoretically and experimentally. For the experiments, we model the disk drive of a laptop computer as an embedded system. The results show that the proposed algorithms perform well in practice with guaranteed bounds on their performance. Further, this paper conclusively demonstrates that to implement aggressive power management techniques for power critical subsystems, designers will have to commit greater resources such as dedicated registers and ALU units.

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].

[*]The authors would like to acknowledge support from National Science Foundation award numbers MIP 95-01615 (CAREER) and CCR-9806898, and from DARPA grant DABT63-98-C-0045.

[†]Parts of this work was done while this author was working for CynApps Inc., Santa Clara, CA 94043.

Low power VLSI design can be achieved at various levels of abstractions 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. 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 and does not fall into any pattern. The requests typically arrive unpredictably and are not drawn from any well known probability distribution. 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 inter-arrival times ahead of time. 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 the strategies that selectively shutdown subsystems and turn them back on when needed.

1.1 Background and Previous Work

There are essentially two kinds of power management strategies: non-adaptive and adaptive. Typically, power management creates sleep states with various levels of power savings and delay overheads which can be externally controlled. The sleep states are statically computed based on the properties of the system and do not change when the requests are received. In non-adaptive power management, the length of idle periods after which the system shuts off is statically determined. In adaptive power management, the system automatically detects idle periods, which change dynamically, and disables the system.

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.* In non-adaptive algorithms, the idle period after which the system shuts down is predetermined. 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. Using heuristic algorithms, all previous works have experimentally shown that this intuition holds.

This paper builds upon earlier works on power management strategies [3, 1, 2, 13]. 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. Further, their algorithm may not be adapted to hardware systems since it requires expensive computation resources that may not be available.

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 [16] and using the Auspex file traces [14] they generate a Markov model using an exponential distribution as the base distribution. However, their assumption that the inputs are exponentially distributed is not fully justified 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.

1.2 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 arrived at by examining trace patterns of several experimental examples. In this paper, we attempt to develop a yardstick within which these heuristics can be analyzed and propose algorithms that are independent of the input distribution.

Competitive analysis has been used earlier as a technique to analyze problems similar to the power management problem in the theoretical computer science community. The algorithms and proofs presented in this paper have been adapted for the power management problems from the works of [4] and [5]. In [4], the authors solve the spin-block problem which is similar to the power management problem and as a result the proofs and algorithms for both problems are almost the same.

This paper has four 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 existing heuristics. 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 non-adaptive 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. We also show that in order for any online algorithm to achieve this lower bound, it has to maintain a *complete* history of the inter-arrival times of the requests in the input sequence. Since this is not practical, we present a simple algorithm that needs only the last inter-arrival time. We show that this algorithm performs as well as the heuristics, but we can bound its performance in the worst case.

To test the performance of these algorithms, we use the disk drive [16] of a laptop as an embedded system. We compared our algorithms with Hwang and Wu's heuristic algorithm [1] which is the most comprehensive algorithm and does not depend on the input sequence patterns. Our results show that the algorithms presented in this paper perform as well as the heuristics, are simpler to implement and their performance is guaranteed.

2 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: (a) replace algorithms for maintaining a cache, (b) load balancing in a distributed environment, (c) 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 learnt. An alternative approach to analyzing *online* problems, called *competitive analysis* has been considered by computer scientists in recent years. Sleator and Tarjan [12] introduced it while dealing with 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, matching, $k-$server, and file access and allocation in distributed systems. In this paper, we adapt competitive analysis to estimate the power dissipation in embedded systems.

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

We are interested in picking the algorithm S that minimizes the competitive ratio. Competitive ratio as a measure for online algorithms was first introduced in [12]. We assume that the adversary generates the input sequence to the

```
Algorithm NonAdaptive:
(1) if (idle_state) then
(2)     idle_intervals = idle_intervals + 1;
(3) if (idle_interval $\geq$ k) then
(4)     shutdown
```

Figure 1: *The optimal non-adaptive online algorithm.*

algorithm S, based only on the description of the problem. Intuitively, for our application, if an algorithm S 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.

2.1 Online Power Management

Consider a system that is interacting with its environment by servicing requests. A power management strategy for such a system dictates when to shutdown after the requests from the environment cease to arrive. The arrival of requests is online and the problem of determining when the system should shut itself down is an online problem. Further, after a system shutdown, it may takes a time interval called *revival time*, L, to come back to its normal operating mode. The power dissipated during revival is called *revival power* and is denoted by P_r. The power dissipated by the system when it is idle is called *idle power* and is denoted by P_i.

3 An Optimal Non-Adaptive Power Management Algorithm

In this section, we present a non-adaptive online algorithm for power management. We also establish a lower bound on any non-adaptive online algorithm, thereby proving that our algorithm is optimal. We note that this problem is identical to the ski-rental problem discussed in [10] and its solution is typically presented as an introduction to online algorithms and competitive analysis.

3.1 The Algorithm

To simplify the analysis, we normalize the revival power so that P_r is dissipated in a single time unit. Hence, P_r is the power as well as the energy dissipated during revival. Now, let us assume that $P_r = k \cdot P_i$ for some integer $k \geq 1$. As a result, we have discretized time has into integer units. Now, the shut off 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 Figure 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 k, the system shuts off.

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

3.2 Analysis and Proof of Optimality

We will now show that algorithm NonAdaptive is $2 - \frac{1}{k}$ competitive. Let us assume that n time units expire between two adjacent requests. This information is not known to

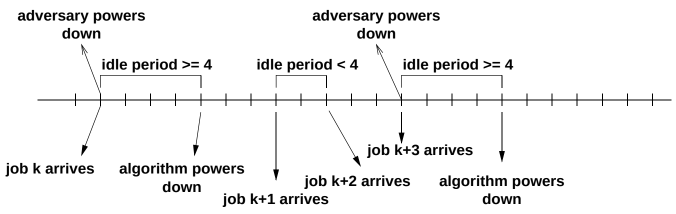

Figure 2: *The behavior of the algorithm and the optimal adversary on an input sequence. Since the adversary knows the input sequence ahead of time, the adversary can power down immediately when the next request is at least k (in this case we have chosen k = 4) units away. The algorithm on the other hand, has to wait in the idle state till it becomes advantageous to power down. We assume for the sake of simplicity that jobs are serviced instantaneously.*

algorithm NonAdaptive. We merely use it for the analysis. However, since the adversary controls the sequence of inter-arrival times, n is determined by the adversary. If the system is not shut down ($n < k-1$), then the off-line algorithm and the online algorithm dissipate the same amount of power since they both remain idle till the arrival of the next request.

If the system is shut down ($n \geq k-1$), then the algorithm NonAdaptive dissipates

$$((k-1) \cdot P_i) + P_r$$

energy.

The first term comes from the fact that the system was idle for $(k-1)$ time units 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 power).

Now,

$$((k-1) \cdot P_i) + P_r = ((k-1) \cdot P_i) + k \cdot P_i = (2k-1)P_i$$

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 that 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
$$P_r = k \cdot P_i$$
units of energy.
2. The adversary does not shut down the system at all. It remains idle till the arrival of the next request, in which case it dissipates
$$n \cdot P_i$$
units of energy.

The adversarial strategy depends on the value of n, the inter-arrival 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 $k \cdot P_i$ units of energy.

Alternatively, 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 $n = k$.

Thus, algorithm NonAdaptive attains competitive ratio

$$\frac{((k-1)\cdot P_i) + P_r}{k \cdot P_i} = \frac{((k-1)\cdot P_i) + k \cdot P_i}{k \cdot P_i} = 2 - \frac{1}{k}$$

The factor $\frac{1}{k}$ is a consequence of the quantization of time. It is easy to see that if we assume that time is continuous, the competitive ratio will tend to 2 since k will tend to infinity. Based on the kind of embedded system, hardware or software, we can make some assumptions about the discreteness of time which will effect the competitive ratio. In this paper, we do not consider the increase in latency of the system whose power we were are trying to manage in the analysis: we use the experimental results to reach our conclusions. We refer the reader to [11] for a detailed analytical discussion of the effects of power management on latency of the system.

Next, we will show that algorithm NonAdaptive is optimal. This involves proving a lower bound for competitive ratio of the problem. 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_t be any arbitrary deterministic strategy where the "threshold" $t \neq k$. There are two possible cases for analysis:

1. $t < k$: S_t attains a competitive ratio of

$$\frac{(t-1)\cdot P_i + P_r}{t \cdot P_i} = 1 + \frac{k-1}{t} \geq 2$$

2. $t > k$: S_t attains a competitive ratio of

$$\frac{(t-1)\cdot P_i + P_r}{P_r} \geq \frac{(k-1)\cdot P_i + P_r}{P_r} = 2 - \frac{1}{k}$$

Therefore, S_t consumes more power than P_k. This proves that under the assumption that time has been discretized, strategy NonAdaptive is optimal. The same analysis applies under the assumption that time is continuous.

The term $\frac{1}{k}$ occurs in the competitive ratio since we have assumed that time is discrete. 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. We refer the reader to [11] for a comprehensive discussion of these issues..

4 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. We can also show analytically that the lower bound on the competitive ratio for adaptive algorithms is better than the competitive ratio for non-adaptive algorithms. Let us assume that each inter-arrival time is independently chosen from the same distribution and adaptive algorithms attempt to "learn" this distribution. In practice, for any adaptive algorithm to achieve the lower bound, it would have to keep track of the entire history of input sequences in order to estimate the distribution from which the input sequences arise. 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 oblivious adversary. The analysis in this section builds upon the work of Karlin, Manasse, Rudolph and Sleator [5], and specifically their formulation of the spin block problem [4]. We augment their theoretical work with experimental results.

4.1 Lower Bound for any Adaptive Algorithm

In the case of adaptive algorithms, assume that each inter-arrival 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. Let σ denote an arbitrary inter-arrival time from an arbitrary sequence of requests. Let algorithm S use $\pi(t)$ as an estimation function for the true distribution. Now, let algorithm S shut down the system after τ units of time in interval σ. Note that τ is generated by S using $\pi(t)$ to estimate it for the interval σ. Now, we prove the lower bound by showing that algorithm S that uses an estimation function $\pi(t)$ to estimate $\Pi(t)$ will be $\frac{e}{e-1}$-competitive. Then, we show that any algorithm that deviates from the correct estimation of the $\Pi(t)$.

Intuitively, if $\Pi(t)$ was: there are only 2 inter-arrival lengths of time to pick from, say 2 and 100. As a result, there is a 50% probability of getting an inter-arrival time of 2 units and 50% of getting one of 100 units. As a result, S would try to learn $\Pi(t)$. When it has learnt $\Pi(t)$ completely, it will shut the system down after it has been idle for 2 units of time since it now has learnt that the inter-arrival can be only 2 or 100.

Let k be a real number such that $k = \frac{P_r}{P_i}$. Let us assume that the system dissipated power proportional to the time for which it is servicing the request and power proportional to k when it is waking up from a shutdown. Then the expected power dissipated by S is

$$E[C_S(\sigma)] = \int_0^{\tau} t \cdot \pi(t)dt + \int_{\tau}^{\infty} (k+\tau)\cdot \pi(t)dt.$$

Algorithm S has to choose values for τ based on $\pi(t)$ such that the expected energy dissipated is minimized.

$$E[C_{opt}(\sigma)] = \int_0^{k} t \cdot \pi(t)dt + \int_{k}^{\infty} k \cdot \pi(t)dt.$$

The optimal off-line algorithm (adversary) has

$$E[C_{opt}(\sigma)] = \begin{cases} \tau & t \leq k \\ k & t > k \end{cases}$$

In other words,

$$E[C_{opt}(\sigma)] = \int_0^{k} t \cdot \pi(t)dt + \int_{k}^{\infty} k \cdot \pi(t)dt.$$

	Power dissipation in watts				Latency in milliseconds			
	Algorithm				Algorithm			
Traces	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
t6.H1062	0.8504	0.0802	0.0424	0.0409	0.449	0.590	0.630	0.647
t6.H1074	0.8503	0.3381	0.3121	0.3248	0.209	0.576	0.912	0.884
t6.H2012	0.8505	0.2005	0.0897	0.1083	0.265	0.688	0.843	0.887
t6.H2014	0.8500	0.0753	0.0491	0.0430	0.001	0.905	1.330	2.311
t6.H2149	0.8501	0.0613	0.0419	0.0278	0.144	0.603	0.849	1.075
t6.H3069	0.8503	0.2807	0.2334	0.2197	0.095	0.376	0.594	0.615
t6.H3073	0.8502	0.0509	0.0363	0.0230	1.256	2.138	2.670	3.434
t6.H3113	0.8508	0.1739	0.1173	0.0723	0.661	0.932	1.074	1.106
t6.H4060	0.8500	0.0626	0.0586	0.0197	0.026	0.552	1.059	1.712
t6.H4119	0.8501	0.2148	0.1092	0.0761	0.058	1.165	1.639	1.942
t6.H4127	0.8505	0.0628	0.0611	0.0341	1.167	1.456	1.804	1.888
t6.H4181	0.8501	0.0627	0.0197	0.0178	0.185	0.565	0.650	0.663

Table 1: *The average power dissipation and latency for (a) no power management scheme, (b) the simple non-adaptive algorithm, (c) the simple adaptive 3-competitive algorithm and (d) Hwang and Wu's adaptive algorithm where the shutoff threshold is a cumulative average of all inter-arrival times.*

The adversary picks the distribution, $\Pi(t)$ and the expected energy dissipated by the algorithm, S is maximized. Therefore,

$$E[C_S(\sigma)] \leq \quad (1+\alpha)E[C_{opt}(\sigma)]$$

for some $\alpha > 0$ and α is minimized.

Setting the preceding inequality to equality and solving the differential equations obtained by differentiating twice with respect to τ, we obtain

$$\pi(t) = \begin{cases} \frac{1}{(e-1)k}e^{t/k} & 0 \leq t \leq k \\ 0 & \text{otherwise} \end{cases}$$

Solving for α by setting $\int_0^\infty \pi(t)dt = 1$, we get $\alpha = \frac{1}{e-1}$. Therefore, the algorithm S yields a competitive ratio of

$$1 + \alpha = \frac{e}{e-1} \approx 1.5819767 < 1.6$$

Since the choice of σ was arbitrary, the analysis applies to any interval in the request sequence. This shows that there is an adaptive algorithm whose competitive ratio is $\frac{e}{e-1}$. Now, we have estimated $\pi(t)$ to the performance of the optimal offline algorithm. Any estimator $\gamma(t)$ that does not estimate the optimal offline exactly will not be as competitive as S. Due to lack of space, we skip the details of this proof.

This shows that the optimal algorithm will have to maintain accurate statistics by keeping track of the entire history of the inter-arrival times of requests. A practical alternative to this algorithm is to keep track of the last few inter-arrival 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.

4.2 A 3-Competitive Adaptive Algorithm

Figure 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) $\tau \geq k$: In this case, the previous inter-arrival 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, yielding 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 inter-arrival times. This computation uses significant hardware resources. Therefore, this strategy is not suitable for implementation into hardware. Since they use a larger history of inter-arrival times to compute their shut down threshold, their algorithm will be more competitive than ADAPTIVE, but only marginally so. The 3-competitive algorithm, ADAPTIVE, 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.

5 Experimental Results and Discussions

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. $P_i = 0.85$ watts,

```
Algorithm ADAPTIVE:
(1) $\tau$ = curr_arrival_time - prev_arrival_time;
(2) if (idle_state) then
(3)     idle_intervals = idle_intervals + 1;
(4) if ($\tau \geq k$) then
(5)     shutdown
(6) else
(7)     if (idle_intervals $\geq$ k) then
(8)         shutdown
```

Figure 3: *The 3-competitive adaptive algorithm for power management that governs the shut down of an embedded system.*

$P_r = 4.5$ watts, $L = 4$ milliseconds and average power dissipated to service a request is 2.3 watts. Using these parameters, we modeled and simulated the following scenarios to compute the power dissipated: (a) no power management scheme at work, (b) the simple non-adaptive strategy, (c) the simple adaptive 3-competitive algorithm and (d) Hwang and Wu's adaptive algorithm where the shutoff threshold is a cumulative average of all the inter-arrival times.

We observed, as we had predicted, that the 3-competitive strategy does well under most conditions. Hwang and Wu's algorithm does marginally better than algorithm ADAPTIVE on some traces. We also find that algorithm ADAPTIVE is much better than algorithm NONADAPTIVE and so is Hwang and Wu's algorithm. We have run several experiments on the modeled disk drive. Table 1 presents the results obtained from the experiments. The traces are named based on the day they were obtained and the host from which they emanated. These results show that the algorithms perform as well in practice as the heuristics; however, the algorithms are very simple and can be used in both hardware and software systems with their performance guaranteed.

We also note that the adaptive algorithms are a lot more aggressive in shutting down the system than the non-adaptive algorithm. This saves power, but trades off with the latency of the system. We believe that the system designer has to make design tradeoffs between power dissipation and latency.

Though we have shown that the worst case competitive ratio for algorithm ADAPTIVE is 3, the experimental results show that the worst case input sequence does not occur often over a large time interval. However, the worst case for algorithm NONADAPTIVE is encountered often enough that it dominates the power dissipated. Therefore, algorithm ADAPTIVE performs better than algorithm NONADAPTIVE in practice and as shown by our experimental results.

The analysis we present outlines the worst input sequence for Algorithm NONADAPTIVE and in practice, the probability of the occurrance of this sequence over a large interval is substantial and therefore the lower bound dominates the power dissipated by algorithm NONADAPTIVE. However, the behavior of algorithm ADAPTIVE is different on algorithm NONADAPTIVE 's worst case input sequence. Algorithm ADAPTIVE predicts the next inter-arrival time based on the previous one. Therefore, its worst case sequence becomes one where every other inter-arrival time is larger than k and in practice, the probability of this sequence occurring over a large period of time is very small. Hence, we see that algorithm ADAPTIVE performs significantly better than algorithm NONADAPTIVE. In contrast, we do not know what the worst case bounds for the previously proposed heuristics for this problem are nor do we know what the worst case distribution looks like.

6 Conclusions

We have used competitive analysis to analyze previous "heuristic" algorithms for power management of embedded systems. We present a simple non-adaptive 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.

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, March 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, December 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. Ramanathan, S. Irani, R. Gupta. System Level Power Management and Its Effects on Latency *Technical Report*, Center for Embedded Computer Systems. University of California, Irvine (1999).

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

[13] S. Udani, J. Smith. The Power Broker: Intelligent Power Management for Mobile Computing. *Technical Report MS-CIS-96-12, Department of Computer and Information Science, University of Pennsylvania (1996).*

[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)