A Model Checking Approach to Evaluating System Level Dynamic Power Management Policies for Embedded Systems

Sandeep K. Shukla Rajesh K. Gupta

Center for Embedded Computer Systems,
Department of Information and Computer Science,
University of California at Irvine,
Irvine, CA 92697
E-mail: {skshukla,rgupta}@ics.uci.edu

Abstract

System Level Power Management policies are typically based on moving the system to various power management states, in order to achieve minimum wastage of power. The major challenge in devising such strategies is that the input task arrival rates to a system is usually unpredictable, and hence the power management strategies have to be designed as on-line algorithms. These algorithms are aimed at optimizing wasted power in the face of nondeterministic task arrivals. Previous works on evaluating power management strategies for optimality, have used trace driven simulations, and competitive analysis. In this work, we build upon the competitive analysis based paradigm. Our work views a power management strategy as a winning strategy in a two player game, between the power management algorithm, and a non-deterministic adversary. With the power of non-determinism, we can generate the worst possible scenarios in terms of possible traces of tasks. Such scenarios not only disprove conjectured bounds on the optimality of a power management strategy, but also guides the designer towards a better policy. One could also prove such bounds automatically. To achieve these, we exploit model checkers used in formal verification. However, specific tools which are focused mainly on this kind of power management strategies are under development, which would alleviate some of the state explosion problems inherent in model checking techniques.

1. Introduction

For a range of embedded systems, as well as conventional systems such as servers, work stations, and laptops, controlling power dissipation is an important system design issue [2]. This is either because of limited energy supply in portable devices, or excessive power consumption due to ever increasing micro-electronic integration that has adverse cost implications. Over the past few years, various methods to estimate and minimize the power in the design of systems, at various levels of design abstractions have been proposed. However, in this work, we concentrate on the system-level power management only. In the early stages of system design, one needs to be able to evaluate and estimate impacts of power management policies enforced at the system level. In [3], a system modeling approach for dynamic power management strategy evaluation has been proposed. Their approach is based on creating finite state models, with power states, and state transitions, with a cumulative power dissipation model. However, as stated in [3], assessing the impact in a quantitative manner, of various alternative power management policies on power and performance, is still an open problem. For example, the system level power models in [3] are geared towards a stochastic simulation based evaluation of the power management strategies. As a result, no analytical bounds are obtained, and also due to the usage of stochastic simulations, the quantitative assessments, are as valid as the assumptions made on the probablity distribution of arrival times and durations of systems tasks. In this work, we take a more formal approach, which is amenable to algorithmic analysis using a model checking techniques [5], by modeling systems under design as finite state Kripke structures [5].

2. Dynamic Power Management Strategies

One widely used approach to designing low power embedded systems, is dynamic power management [3]. However, building a system with Dynamic power management is difficult and error prone [3]. Simulations based on system traces, are time consuming, and often slow, especially when

there is a demand for fast time to market. Recent works in system level dynamic power management [4, 11, 14, 19], have mostly used predictive techniques for deciding system shutdown policies, or voltage scaling policies. In order to evaluate a predictive strategy, one has to do extensive stochastic simulation to predict the proximity of the strategy to a theoretically optimal strategy, and predict the effects on system performance. A more analytical approach based on Competitive Analysis [1] was taken in [15, 17, 16]. In this approach, no assumptions were made about the probability distribution of the task arrival rate, or duration of the tasks. Worst case competitive analysis was used to evaluate the policy optimization strategies, to obtain competitive bounds on the optimality of the strategies. In other words, this approach could provide a factor f, such that the energy dissipated when using a specific policy, is no worse than f times the energy dissipated had there been an oracle to know the task arrival and durations before hand. However, in that approach the correctness of bounds were proven by combinatorial arguments, and there was no automatic method for generating scenarios that would falsify conjectured bounds. Automated proof of bounds, and automated generation of scenarios that disproves conjectured bound, would help designers to evaluate new strategies, for their effectiveness, or change the algorithms to meet the required bounds.

2.1 Dynamic Power Management as a Game

In this work, the problem we want to tackle is as follows: We view the dynamic power management problem as a two player game between a power management policy and a malicious *adversary*. The system under design is modeled as a finite state system, that receives service requests, and request arrival times are unknown a priori. We do not assume any probability distribution of the inter-arrival time, because, usually such distributions need to be validated using simulation, and *test of hypothesis*, and even validated assumptions are only specific to particular work loads. The first player, being the power management strategy, takes the system into various power states (modes), in order to save or reduce power dissipation, based on system idle time, and other parameters. The other player in the game is a malicious adversary that generates service requests, at the most inopportune manner for the power management policy. The adversary, being the generator of the requests, can take the system to appropriate power states, and hence represents the theoretically optimal power managemet policy. We want to optimize our power management policy so that the adversary can never make the system dissipate more than a required multiple of the power dissipated by the theoretically optimal policy. If we do not make any assumptions on the behavior of the adversary, we obtain generic quantitative bounds on the ratio of power dissipated by a particular management policy with respect to a theoretically optimal policy. However, if we want, we could make assumptions of various kinds, including probability distribution on the request inter-arrival time, on the adversary, and obtain more specific kind of bounds. However, in this paper we only concentrate on the generic bounds, to show the power of our techniques. In the high level design context, we might also need to automatically construct scenarios which lead to more power dissipation than expected by a required bound fixed apriori. Ability to generate such scenarios will give feed back, in improving the policy, and once a policy is improved, we want to prove that the required bound still holds. A natural extension of this problem is to introduce resource constraints (such as latency constraints), and by modeling those, we can obtain similar bounds on power dissipation, which maintains the required constraints.

3. A Model Checking Approach

In this work, we approach this problem of evaluating the strategies, using a formal verification like approach using a model checker. For proof of concept, we used a off-the-shelf model checker [6], namely SMV [10]. However, since our models are of a specific kind, an off-the-shelf model checker soon gets into the well known state explosion problem [6], which may be avoided if we design a model checker that can handle this class of models more efficiently without exploding in space complexity.

However, for the sake of introducing the techniques, we consider an example of a simple threshold based non-adaptive dynamic power management strategy [15]. In this example, there are three power management states of a system. When the system is active on a *task*, it is said to be in *busy* state. Since, we do not consider, voltage scaling, we assume that at *busy* state, all management policies will have to dissipate the same amount of energy, and hence, it does not contribute to wasted idle energy, which is our major concern. We also model a discrete time view of the system, and when a system is busy with activity, we assume that event *tsk*, is detected. When the system has no task to execute, we assume that an event *ntsk* is detected by the power manager. As a result, the sequence of job arrivals and executions, are modeled as a sequence over the event set $\{tsk, ntsk\}$. When the system has no task to work on, it immediately transitions into an *idle* state. The energy dissipated from this state to come back to *busy* state, upon arrival of a task is small. However, to stay in this state, the system has to dissipate, and hence waste energy. The third state, is a *sleep* state, which the system transitions to, if it does not want to dissipate any power. However, transitioning too early to a *sleep* state, may lead to wastage of power, because the energy required to come back to *busy* state from *sleep* state is usually high. As a result, a thresh-

old based power management strategy, usually waits at the *idle* state for a threshold amount of clock cycles, after which it decides to go to a sleep state. Figure 1, shows the state diagram of the system, showing the transitions taken by our power management policy (as given in [15]), shown using solid arrows. The assumption in [15], is that, the ratio of the energy dissipated to move from *sleep* to *busy* state, to the energy dissipated per cycle at *idle* state is *k*, which is a fixed constant. The power management policy, hence uses a stopwatch, every time, it starts idling. When the stopwatch ticks *k*, the system is taken to *sleep* state. The transition shown by *dashed* arrows, are the transitions that an adversary would take. Notice that, the adversary can only differ from the policy, as to when to transition from the *idle* to *sleep* state, because, the adversary, being all powerful, knows when the next task will arrive. So, if the adversary knows that the next task is imminent, it would stay in the *idle* state, rather than moving to *sleep* state, just because the clock ticks at *k*. The way to model, this power of the adversary, is by nondeterministically choosing this transition, with the help of a nondeterministic variable *random*. This is where, the state space exploration based tools, present their power. In a simulation environment, this activity of an adversary has to be modeled programmatically, by setting a random number, using randomization. Here, that is not necessary, because a state space exploration tool, explores all possible paths through the state space, and therefore perfectly models the most arbitrary behavior of the adversary.

Now, suppose, we want to prove a bound on the amount of energy dissipated while idling, by the policy, with respect to an all knowing adversary, we have to model the power dissipation, as extra variables in the system, which is not known in the figure. If we call them *policy_power*, and *adversary_power*, then we would ask a model checker to prove the following property:

$$assert(G(policy\_power <= f * adversary\_power))$$

. However, if the property, holds, then we have proven a competitive bound, fully automatically. Note, that this property is a simple *safety* [6] property, in Linear Time Temporal Logic. However, more interesting is the case, when the bound does not hold. Then we obtain a trace of a counter example, which shows exactly the steps the adversary will take, to falsify the bound. This allows for a debugging by the system designer to tune the policy to make the prespecified bounds.

However, the finite state model checker SMV used in our first experiments, had to be also bounded on the number of steps, that one needs to explore, before, the bound is proven or falsified. The soundness of such a proof crucially depends on a *small model* theorem, which says that if there is a counter example, there is a small enough one, so that bounding the number of steps of the system is not unsound.

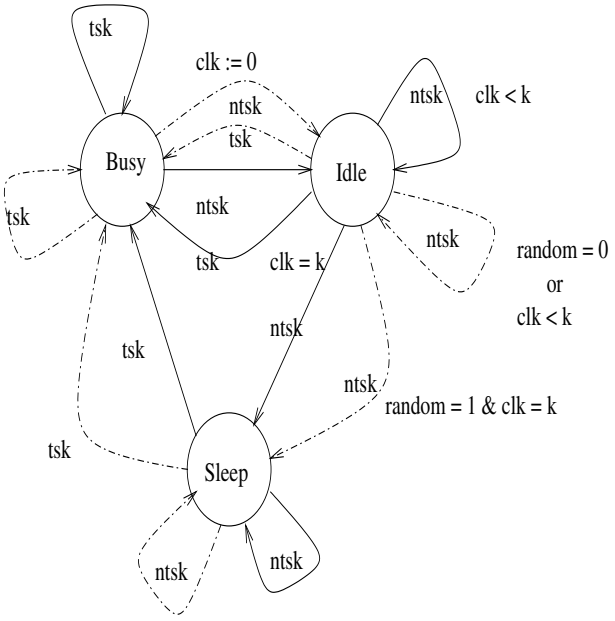

Figure 1. Single Sleep State Case

We do not have a generic small model theorem for dynamic power management, but for the given example, we have a small model theorem stated below. Due to lack of space, we are not giving detailed proof, but since this example is small and straight forward, we expect, astute readers to see the correctness of the theorem.

Theorem 3.1 *Let*

- n_{ij}, *and* m_{ij}, *where* $i, j \in \{busy, idle, sleep\}$, *and subscript* ij *refers to a transition from state* i, *to state* j, *be the number of* i, *to* j *transitions by the policy, and the adversary respectively,*
- k *be the ratio of the energy dissipation in moving from* $sleep$ *state to busy state, to the power dissipation per cycle while remaining* $idle$,
- *the power dissipation in* $sleep$ *state be* 0, *and power dissipation in busy state be counted as* 0,
- f *be a desired bound on the ratio of energy dissipation by the policy, to that by the adversary.*

If the bound f *does not hold for all sequences of* $\{tsk, ntsk\}$, *then there are constants* $c_{ij}(f, k)$ *dependent on* f *and* k, *such that a sequence of* $\{tsk, ntsk\}$ *exists for which* $n_{ij} \leq c_{ij}(f, k)$, *and* f *is falsified on that sequence.*

Theorem 3.1 shows, that by making an assumption on the number of steps of the system, we do not necessarily compromise the soundness. How to compute this bound for this case, can be expressed in another theorem, omitted due to lack of space. However, it must be mentioned, that we do not yet have any generic bounds, or methods to find bounds for all dynamic power management policies. Theorem 3.1 applies to this specific power management algorithm.

In summary, the novelty of our approach lies in two aspects of this technique:

1. Simulation based techniques can be avoided, and more generic optimizations can be achieved, in place of workload specific bounds
2. We can construct worst case traces, automatically, which can guide us better in policy optimization.

For the second aspect, we use counter example generation capability of model checkers [6, 18].

3.1 An Example of a Multiple Power down states

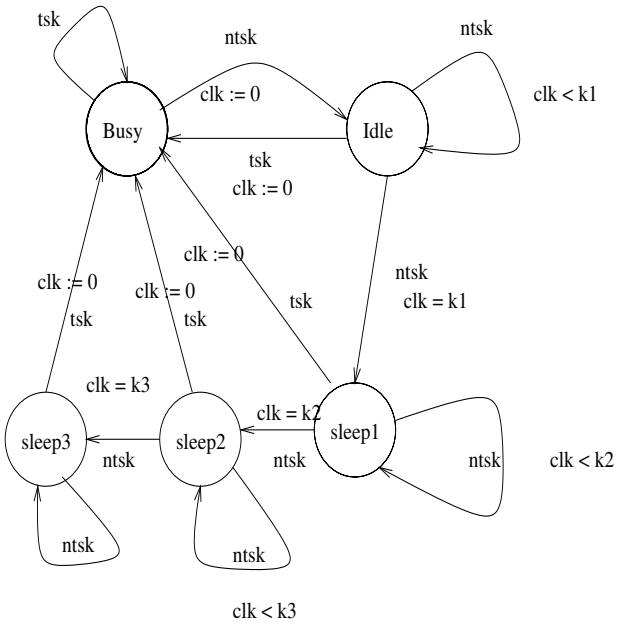

Figure 2. Multiple Sleep States Example

Figure 2, illustrates a case with multiple *sleep* states, each having different power dissipation rate, and different latency and energy consumption to go back to the *busy* state. The power management strategy, fixes multiple threshold values for the clock, such that the system is taken through the subsequent *sleep* states, when the idle time reaches the subsequent threshold values. Figure 2 shows an abstract view of how we model such a system within our framework, and the proof of the competitive bound related properties is very similar to the previous example. In Figure 2, we only show the actions of the power management policy, and not that of the adversary. Also, there is a similar small model theorem, provable in this case.

We also have proven competitive bounds on examples of *adaptive* algorithms for power management. However, the modeling of such algorithm is very similar, except, that, adaptive algorithms are *history* based. In such algorithms, the past inter-arrival times for tasks, are remembered, and according to some functions of that history, the next threshold is set. Since a regular model checker gets easily into state-explosion problem, we cannot model very deep history, however, clever encoding could be used to model the recording of shallow history of past few intervals.

4. Conclusions, Limitations, and Future Work

The results reported in this paper, are very preliminary, and we need to extend this framework, to experiment with other existing dynamic power management policies. We also, plan to carry out experiments with other hybrid [8], and probabilistic [12] model checkers. We plan to experiment with theorem provers such as PVS [13], as well. We believe that, our approach will create a uniform framework for the design and evaluation of dynamic power management policies, which can encompass all kinds of techniques, including probabilistic modeling techniques.

Limitations: In the current implementations, we have proven a number of theoretically proven bounds for Adaptive and Non-Adpative policies described in [15], using the SMV model checker. However, these policies are parameterized with respect to certain algorithm parameters, such as the constant k in the first example in Section 3. We have proven bounds for specific values of k, but we need an induction scheme [9] that would prove these bounds for all values of these parameters.

We also assumed a *small model theorm* that states, that depending on the parameters of the policies, if there are counter examples to a conjectured bound, the size of the smallest counter example is suitably small. We have proven such small model theorems for the examples presented here. However, we need a generic small model theorem, to make this approach even more legitimate. Since the model checkers often get into state explosion problems, we have to limit the number of system steps, we run the proof on.

Current Status and Future work: We have been pushing the model checker SMV to limits. However, unfortunately, some of the arithmetic that we use in the model to accumulate power statistics, for example, enters in an

SMV model as a part of the state variables. In a more custom made model checker tailored for this particular kind of models, we would have arithmetic built into the system, and expressible as rewards/penalties on transitions, rather than on states. We are planning on building such a tool. We also need model checkers that would be able to express probability distribution on the inputs, in order to extend this work to evaluating policies that assumes a specific probability distribution on the inputs. In our on going work, we have modeled and proved some of the policies using a hybrid automaton based model checker HyTech [8], which can model the power dissipation as an analog variable, and the model checking uses polyhedral techniques [7] to handle such analog variables. We also have modeled a probabilistic power management policy using a probabilistic modeling framework, called PRISM [12]. We will report the results of these modeling and verification activities in a future paper.

5. Acknowledgement

This work was supported by SRC, and DARPA/ITO supported PADS project under the PAC/C programme.

References

[1] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. In *Proceedings of the 24th Symposium on Theory of Computation*, pages 39–48, 1992.

[2] L. Benini, G. De Micheli, and E. Macii. Designing low-power circuits: practical recipes. *IEEE Circuits and Systems Magazine*, 1(1):6–25, Mar. 2001.

[3] L. Benini and G. D. Micheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Publications, 1998.

[4] Chi-Hong Hwang, C. Allen and H. Wu. A Predictive System Shutdown Method For Energy Saving of Event-Driven Computation. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pages 28–32, Nov. 1997.

[5] E. Clarke and E. Emerson. Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic. In D. Kozen, editor, *Proceedings of the Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, Yorktown Heights, New York, May 1981. Springer-Verlag.

[6] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[7] N. Halbwachs, P. Raymond, and Y. E. Proy. Verification of linear hybrid systems by means of convex approximation. In *Proceedings of the SAS94: Static Analysis Symposium, LNCS 864*, pages 223–237, 1994.

[8] T. Henzinger. A user guide for hytech. In *http://www.eecs.berkeley.edu/ tah*. 2001.

[9] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[10] K. L. McMillan. Cadence smv website. In *http://www-cad.eecs.berkeley.edu/~kenmcmil/smv/*. 2000.

[11] G. A. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli. Policy Optimization for Dynamic Power Management. In *Proceedings of the 35th Design Automation Conference*, pages 182–187, June 1998.

[12] PRISM. Probabilistic symbolic model checker: Prism. In *http://www.cs.bham.ac.uk/~dxp/prism*. 2001.

[13] PVS. Pvs theorem procer website. In *http://pvs.csl.sri.com*. 1996.

[14] Q. Qiu and M. Pedram. Dynamic Power Management Based on Continuous-Time Markov Decision Processes. In *Proceedings of the Design Automation Conference*, pages 555–561, June 1999.

[15] D. Ramanathan. *High-Level Timing and Power Analysis for Embedded Systems*. PhD thesis, University of California at Irvine, Sept. 2000.

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

[17] D. Ramanathan, S. Irani, and R. Gupta. Latency Effects of System Level Power Management Algorithms. In *Proceedings of the International Conference on Computer Aided Design*, Nov. 2000.

[18] S. K. Shukla, H. B. Hunt III, and D. J. Rosenkrantz. Hornsat, model checking, verification and games. *In Proceedings of CAV'96, LNCS 1102*, pages 99–110, 1996.

[19] M. B. Srivastava, A. P. Chandrakasan, and R. W. Broderson. Predictive Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation. *IEEE Trans. on VLSI Systems*, 4(1):42–54, Mar. 1996.