

A Control-Theoretic Approach to Power Management in Embedded Processors

Nagarajan Kandasamy and Sherif Abdelwahed

Institute for Software Integrated Systems
Vanderbilt University
2015 Terrace Place
Nashville, TN 37203, U.S.A.

Contact author: Nagarajan Kandasamy
Phone: +1 615 322 6973
FAX: +1 615 343 7440
nagarajan.kandasamy@vanderbilt.edu

Suggested Topics: Power management, Low-power embedded systems

ABSTRACT

Embedded processors must be operated in power-efficient fashion while satisfying application-specific quality-of-service (QoS) requirements. This paper addresses the power management of embedded processors via a control-theoretic method. A model-predictive controller is proposed where the corresponding control signals managing power consumption are generated via a mathematical processor model by minimizing a specified cost function. Using a past history of time-varying data arrival and processing rates, the controller predicts future processor behavior over a finite look-ahead interval. Based on this prediction, it then derives the lowest possible operating frequency necessary to process data arrivals by optimizing (multiple) QoS criteria. We describe the processor model, formulate the power management problem, and derive the necessary controller. The performance of the proposed controller is also evaluated via detailed experiments using synthetic workloads.

Key words: Low power, power management, embedded systems, model predictive control

1 Introduction

Embedded microprocessors have been built into a wide range of consumer devices to perform application-specific functions; the end user is often unaware of the processors' existence. Other emerging uses of such processors are in sensor networks to solve a diverse set of distributed sensing and control problems. Typically, the applications hosted by such networks fuse (combine) data from multiple processors to accurately assess and control their surrounding environments, and have both military and civilian uses, including target detection, tracking, environment and industrial-plant monitoring, development of smart buildings, and surveillance for law enforcement [8]. These embedded systems have several distinguishing characteristics, which collectively pose some new and important research challenges: (1) The processors have limited computing and power resources. (2) They must have long lifetimes requiring little (or no) maintenance. (3) They must satisfy application requirements while operating in unpredictable and dynamic environments.

The fore-mentioned processors must be operated in power-efficient fashion to maximize their lifetimes. Furthermore, they must also satisfy application-specific quality-of-service (QoS) requirements. This paper addresses efficient power management on embedded processors while satisfying application QoS requirements via a control-theoretic method. The proposed scheme uses *Model Predictive Control (MPC)* where the control signals managing power consumption are generated via a (mathematical) behavioral processor model by minimizing a specified objective function. The MPC approach, widely used for industrial process control [6], is well suited to handle the dynamic processor operating conditions.

Processor power is typically managed using an appropriate policy; for example, during periods of inactivity, the processor may be shut down or placed in a low-power state. Since such state transitions incur some time (power) overhead, predictive methods to determine the type of transition and when to perform it have been developed [4] [25]. Prediction considers both past data arrival history and application performance objectives. Also, since the embedded systems of interest handle a time-varying workload, continuous peak performance is not required during processor operation. Energy can be conserved via *dynamic voltage scaling (DVS)* which adapts both the supply voltage and operating frequency of the processor to suit current computational requirements [5]. Since power consumption relates quadratically to the supply voltage, the corresponding energy savings can be quite significant. Many low-power processors including Intel's StrongArm [20] support DVS.

The authors of [24] [18] use past workload history to predict the workload corresponding to a future time interval, and the operating frequency is set to a value capable of processing this workload. A number of authors have also proposed DVS-aware extensions to the well known rate-monotonic and earliest-dead-

line-first task scheduling algorithms [19] [9] [10] [21]. Typically, these approaches aim to schedule a set of periodic (aperiodic) tasks under the specified scheduling discipline while minimizing power consumption.

The problem of interest is to manage the limited power resources of embedded processors efficiently while meeting desired QoS objectives under unpredictable operating conditions. Recently, control-theoretic methods have been successfully applied to other resource management problems in computer systems including real-time processor scheduling [13] [7] [3], bandwidth allocation and QoS adaptation in web servers [2], load management in e-mail servers [17], and network flow control [15]. Classical control theory provides a systematic approach to resource management in general settings; if the underlying computer system is correctly modeled and its operating environment accurately estimated, the control actions required to maintain a certain QoS level and/or optimize a given utility (cost) function such as power consumption can be derived—as shown by Adelwahed et al. [3]. Moreover, control theory provides well-established mathematical techniques to analyze the correctness and performance of the proposed methods [16].

With few exceptions, however, there has been limited research into applying control-theoretic concepts to manage energy consumption in processors. The authors of [22] present a closed-loop controller which adapts the energy consumed by functional units in a processor in response to an exponentially distributed workload. In [14], a feedback controller addresses similar QoS concerns in multimedia applications by scaling the processor frequency appropriately in response to current throughput. The operating frequency, however, is scaled in the continuous domain or over a large number of discrete frequency settings. This assumption, however, may not always hold in practice; for example, both the AMD-K-2 [1] and StrongARM [20] processors offer only a limited number of frequency settings, eight and ten, respectively.

We develop an MPC-based approach to managing power in processors using DVS to process (environmental) data with time-varying arrival rate as per QoS requirements at the lowest possible operating frequency. Using a past history of data arrival and processing rates, a mathematical model of processor operation predicts its future evolution over a finite look-ahead duration. Based on the predicted system evolution, the control signals—frequency settings—required to operate the processor close to the desired point are calculated by optimizing the (multiple) QoS criteria. We describe the processor model, formulate the power management problem, and derive the necessary controller. The major features of the proposed approach are as follows:

- The processor model does not assume an *a priori* signal arrival distribution and is well suited to real-world operating conditions.
- Rather than assume worst-case processing times for the data items, the controller estimates the current (instantaneous) values using recently observed times.
- The proposed controller operates in a discrete frequency domain; thus, it is directly applicable to pro-

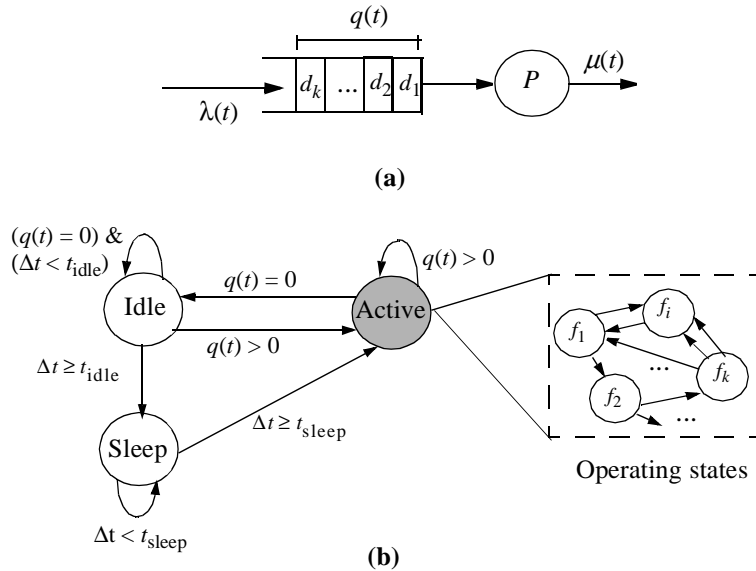


Figure 1. (a) A queuing model of the processor and (b) a state machine representation of typical processor operating modes

processors with limited operating frequencies.

- The controller handles the multi-variable optimization problems.
- Control signals are obtained via iterative optimization which may be interrupted anytime while still providing the current best solution. thereby trading-off execution time and solution quality. It can be used in real-time systems.

Finally, we evaluate the performance of our approach via detailed experiments using synthetic workloads.

The rest of this paper is organized as follows. Section 2 discusses system modelling assumptions while Section 3 presents the controller design. Section 4 evaluates the performance of the proposed algorithms. We summarize the results and conclude the paper with a discussion on future work in Section 5.

2 System Modeling Assumptions

This section discusses the processor behavior and energy models assumed in this paper and the online control problem for power management.

Figure 1(a), shows a standard queuing model for a processor P where $\lambda(t)$ and $\mu(t)$ denote the arrival and processing rates, respectively, of the data stream $\{d_i\}$, and $q(t)$ denotes the queue size at time instant t [11]; we do not assume an *a priori* arrival-rate distribution for $\{d_i\}$ and the processor does not idle whenever the queue contains data items. The queue utilization at time t is given by $U(t) = q(t)/q_{\max}$ where q_{\max} is the designer-specified maximum queue size.

Figure 1(b) shows a state machine representation of a typical power management scheme for processor P where transitions may be triggered by events or the passage of time. For example, when the queue is empty, P is idled to save power; when new events arrive, P is switched back to the active (shaded) state with little time overhead. If the processor stays idle beyond a threshold duration, it is placed in the sleep state for a specified time period. In this state, however, P does not register external events, and consequently, they are simply dropped. The processor transitions back to the active state at the end of the sleep period. Though the proposed controller design focuses on the active state, it can be readily integrated with the high-level power management state machine where predictive shutdown methods such as those proposed in [25] can be used to affect the necessary transitions.

The active state for processor P in Fig. 1(b) is a collection of several discrete sub-states, each with a specific frequency setting f_i ; in this state, power consumption can be minimized by scaling f_i appropriately. We assume a processor capable of operating within a limited number of $\{f_i\}$ settings. This differentiates our work from that of [14] where frequency is scaled in the continuous domain. We denote the time required to process d_i while operating at the maximum operating frequency f_{\max} by c_i . Then the corresponding processing time while operating at some instantaneous frequency $f(t) \in \{f_i\}$ is $c_i/\alpha(t)$ where $\alpha(t) = f(t)/f_{\max}$ is the appropriate scaling factor. Furthermore, in embedded systems such as sensor networks, it is reasonable to expect some correlation between the execution times corresponding to successive data signals $\{d_i\}$ since these signals are usually temporally correlated. This implies that the corresponding execution times may be predicted with some accuracy. The controller presented in Section 3 uses an autoregressive moving average (ARMA) filter to predict both data arrival rates and their execution times.

We use the model proposed by Sinha and Chandrakasan [23] to estimate the energy consumed by processor P ; the energy consumed during time period t is $[\alpha(t)]^2$ where $\alpha(t)$ is the corresponding scaling factor. This simple model provides reasonably accurate estimates and has been previously used in [14] to evaluate controller performance.

Finally, we describe the requirements of our controller informally; a more rigorous problem formulation follows in Section 3. During any given time interval t , the controller on processor P must simultaneously minimize both the queue utilization $U(t)$ and energy consumption $E(t)$; lower $U(t)$ values are desirable since the processing delay incurred by a newly arrived data item is inversely proportional to $1 - U(t)$. The controller, therefore, solves a multi-variable optimization problem involving $U(t)$ and $E(t)$, and having contradictory requirements. In such cases, the designer typically prioritizes these variables to achieve the desired controller behavior.

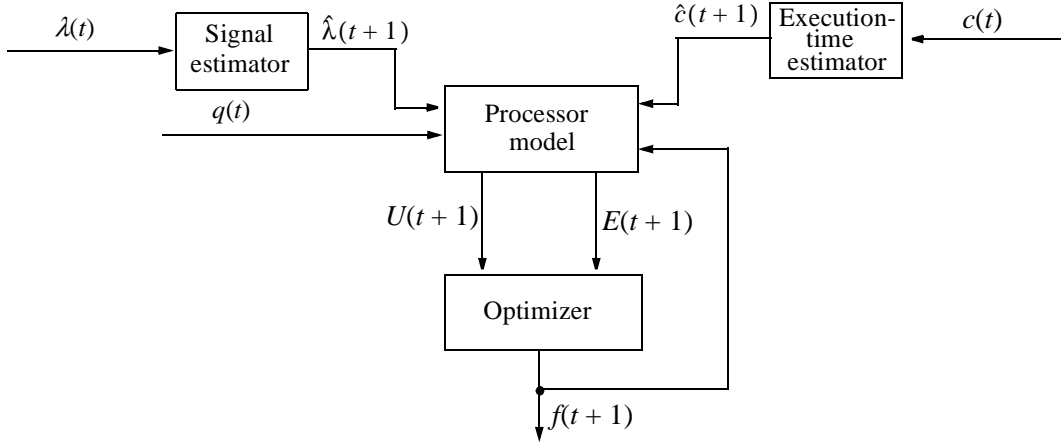


Figure 2. The basic structure of the model predictive controller

3 Controller Design

This section briefly describe the online control method and formulate the power management problem including the system model and cost function. Issues related to controller feasibility and stability are also discussed.

We propose a MPC-based (predictive) controller which uses a behavioral processor model to obtain the corresponding control signal(s) by minimizing a specified objective function [6]. The basic ideas behind the controller are as follows:

- The future processor outputs, in terms of $U(t+k)$ and $E(t+k)$, $k = 1 \dots N$, for a pre-determined prediction (or look ahead) horizon of N steps are obtained during each sampling instant t using the processor model. These predictions depend on known values (past inputs and outputs) up to the sampling instant t , and on the future control signals, in terms of $f(t+k)$, $k = 1 \dots N-1$, which are inputs to the processor and to be calculated.
- The set of future control signals at each step of the prediction horizon are calculated by optimizing a (multi-variable) cost function $F(U(t+k), E(t+k))$ to achieve the designer-specified behavior.
- The control signal $f(t)$ resulting in the best behavior, as determined by the cost function, over the prediction horizon is chosen and applied as input to the processor during sampling instant t ; the other inputs are rejected. During the next sampling instant $t+1$, $U(t+1)$ and $E(t+1)$ are known and the control method is repeated again. Note that the observed $U(t+1)$ and $E(t+1)$ values may be different from those predicted by the controller at time t .

Figure 2 shows the controller structure comprising the processor model, estimators, and the optimizer. The processor behavior of interest is described in terms of the dynamic queue utilization $U(t)$ and the cor-

responding energy consumption $E(t)$. We assume discrete sampling times with uniform interval sizes. If $q(t)$ denotes the queue length at time t , then the estimated queue length at time $t + 1$ is:

$$q(t+1) = q(t) + \left(\hat{\lambda}(t+1) - \frac{\alpha(t+1)}{\hat{c}(t+1)} \right)$$

where $\hat{\lambda}(t+1)$ and $\hat{c}(t+1)$ denote the estimated data arrival rate and execution time, respectively, and $\alpha(t+1) = \frac{f(t+1)}{f_{\max}}$ is the scaling factor; the execution time $\hat{c}(t+1)$ is obtained with respect to the maximum processor frequency f_{\max} . The corresponding queue utilization is:

$$U(t+1) = \frac{q(t+1)}{q_{\max}}$$

where q_{\max} , the maximum queue size, is constrained by the available memory. The energy consumed by the processor while operating at frequency $f(t+1)$ is:

$$E(t+1) = [\alpha(t+1)]^2$$

The optimizer then minimizes the following cost function to obtain the required operating frequency $f(t)$:

$$F(t) = w_u \cdot [U(t)]^2 + w_e \cdot [E(t)]^2$$

where w_u and w_e are designer-specified weights denoting the relative importance of variables $U(t)$ and $E(t)$, respectively.

A good estimator of future system inputs (outputs) is crucial to the predictive controller design discussed above. As shown in Fig. 2, we use ARMA estimators to predict the future data arrival rate $\hat{\lambda}(t+1)$ and corresponding execution time $\hat{c}(t+1)$; for example, given the arrival rate $\lambda(t)$ at sampling instant t and the mean $\bar{\lambda}$ of past observed rates (over a specified history window), the estimated rate $\hat{\lambda}(t+1)$ is:

$$\hat{\lambda}(t+1) = \beta \cdot \bar{\lambda} + (1 - \beta) \cdot \lambda(t)$$

where the gain β determines how the estimator tracks variations in the observed arrival rate; a low β biases the estimator towards the current observation while larger values favor past history. Rather than statically fix β , an adaptive estimator described in [12] can be used. It tracks large arrival-rate (execution time) changes quickly while remaining robust against small variations. When the estimated values match the observed ones, those estimates are given more weight with a higher β . If, however, the estimator does not accurately match the observed values, β is decreased to improve convergence. A second ARMA filter is used to adapt the gain $\beta(t)$ dynamically as follows:

$$\Delta(t) = \gamma \cdot \bar{\Delta} + (1 - \gamma) \cdot |\hat{\lambda}(t-1) - \lambda(t)|$$

where $\Delta(t)$ denotes the error between the observed and estimated arrival rates at time t and $\bar{\Delta}$ denotes the mean error over a certain history window, and γ is empirically determined. Then, $\beta(t) = 1 - \frac{\Delta(t)}{\Delta_{\max}}$ where Δ_{\max} denotes the largest error seen in the corresponding history window.

Given the processor model in terms of queue utilization and energy consumption, a cost function, and inputs $\{f_i\}$, at each sampling instant t , the online controller guides the processor through the corresponding

state space $x(t) = (U(t), E(t))$, $x(t) \in \mathfrak{R}^2$. From the current state $x(t)$, the online controller constructs the tree of (all) possible future states $x(t+k)$, $k = 1 \dots N$, upto the specified depth (horizon) N by applying input frequencies from $\{f_i\}$ —the search heuristic used by the controller is discussed in more detail later in the Section. The set of states minimizing the cost function at depth N is selected, and a state x_m is chosen from this set and traced back to $x(t)$ through the state space to identify the input sequence leading to x_m ; the first member of this sequence is then applied to the processor. The above control action is repeated at each sampling period.

Since control actions are typically taken after exploring only a limited number of states, we must guarantee the feasibility of the online controller at design time, i.e., it must maximize (minimize) the specified performance objectives in finite time even under observation (or measurement) errors. Abdelwahed et al. [3] discuss the necessary and sufficient conditions for an online control algorithm to be feasible, and we now discuss the intuition behind their proof. A discrete-event system such as ours is online controllable if for any state, it is always possible to find an input controlling the immediate system evolution by incrementing (decrementing) some state variables; In other words, the system must possess a sufficient degree of freedom to traverse the state space when an inputs is applied.

The controller must also guarantee stability by operating the processor within a predefined safe region—typically obtained by limiting the values of some (or all) system variables. In our system, both the operating frequency and queue size are bounded from above by f_{\max} and q_{\max} , respectively; data items are simply dropped if queue size exceeds q_{\max} . Therefore, the overall system is inherently stable.

Algorithm Design: As noted earlier, the predictive controller evaluates multiple system states within some finite look-ahead interval to determine the best input to apply at current time t . The number of possible states to be explored depends on the size of the input set $\{f_i\}$ and the specified horizon interval. The controller must therefore evaluate the nodes of a search tree having a specified depth. We now propose an efficient search heuristic suitable for online control.

During each sampling instant, the control algorithm accepts the current state $x(t)$, operating frequencies $\{f_i\}$, and search depth d_{\max} as input, and returns the best input $f(t)$ to be applied to $x(t)$. At each increasing depth, however, fewer state spaces may be explored. The intuition behind this strategy is the following. It is reasonable to expect the accuracy of the arrival-rate and execution times estimators to degrade with increasing tree depth, i.e., further down the prediction horizon. Also, since the best node, in terms of the cost function $F(t)$, is updated during each step of the search process, the search algorithm may be stopped at any point, especially for controllers executing within a fixed time budget. This implies a trade-off between control quality and timeliness.

Simulation parameter	Value
Max. queue size q_{\max}	100
Weight w_u	0.65
Weight w_e	0.45
Frequency inputs $\{f_j\}$	Between [200, 600] Mhz in 25 Mhz increments

Figure 3. Important system parameters used in the simulation experiments

4 Performance Evaluation

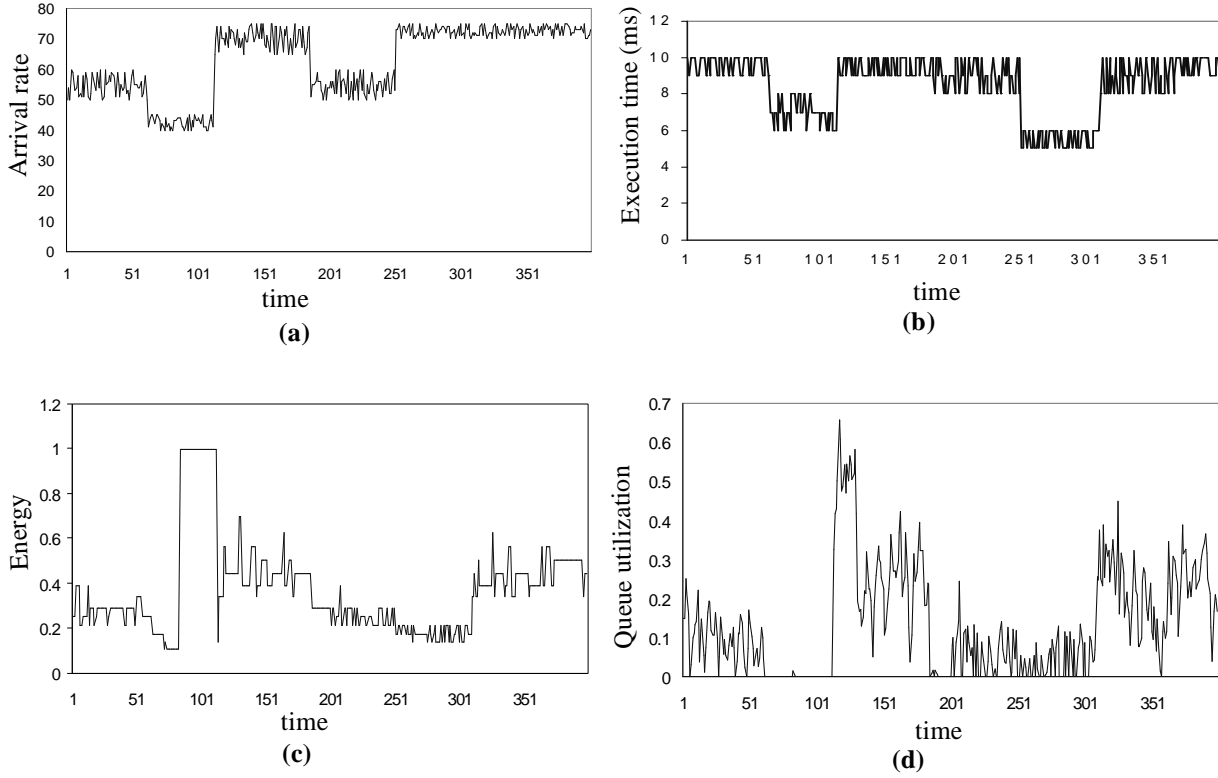
We have evaluated the performance of the proposed controller using synthetic workload. Figure 3 describes some of the simulation parameters assumed in our simulations. The weights w_u and w_e prioritize the energy consumption and queue utilization terms in the cost function (see Section 3) to be optimized, and are specified by the system designer.

Figure 4 summarizes the controller performance corresponding to one simulation run. Figures 4(a) and (b) show the generated data arrival rates and execution times; the distribution aims to simulate a somewhat predictable data stream which can be estimated with reasonable accuracy. Figures 4(c) and (d) show the behavior of the controller under this workload in terms of energy and queue utilization, respectively. Finally, Fig. 4(e) shows the impact of the search depth (or prediction horizon) d_{\max} on controller behavior. Increasing d_{\max} clearly results in better control—quantified by the cost of the obtained solutions in Column 5. However, the controller overhead increases correspondingly, as indicated by the number of explored states in Column 2.

Figure 5 shows another simulation run of the controller; the data arrival rates in Fig. 5(a) exhibit somewhat less predictability from those of Fig. 4(a). It can be seen from the controller performance in Figs. 5(c) and (d) that the estimators track the system behavior closely. Furthermore, increasing d_{\max} results in better control action.

5 Conclusions

Embedded processors must be operated in power-efficient fashion to maximize their lifetimes while satisfying application-specific QoS requirements. We have proposed a predictive controller to efficiently manage power consumption in such processors; the controller uses DVS to process (environmental) data with time-varying arrival rate and execution times as per specified QoS requirements at the lowest possible



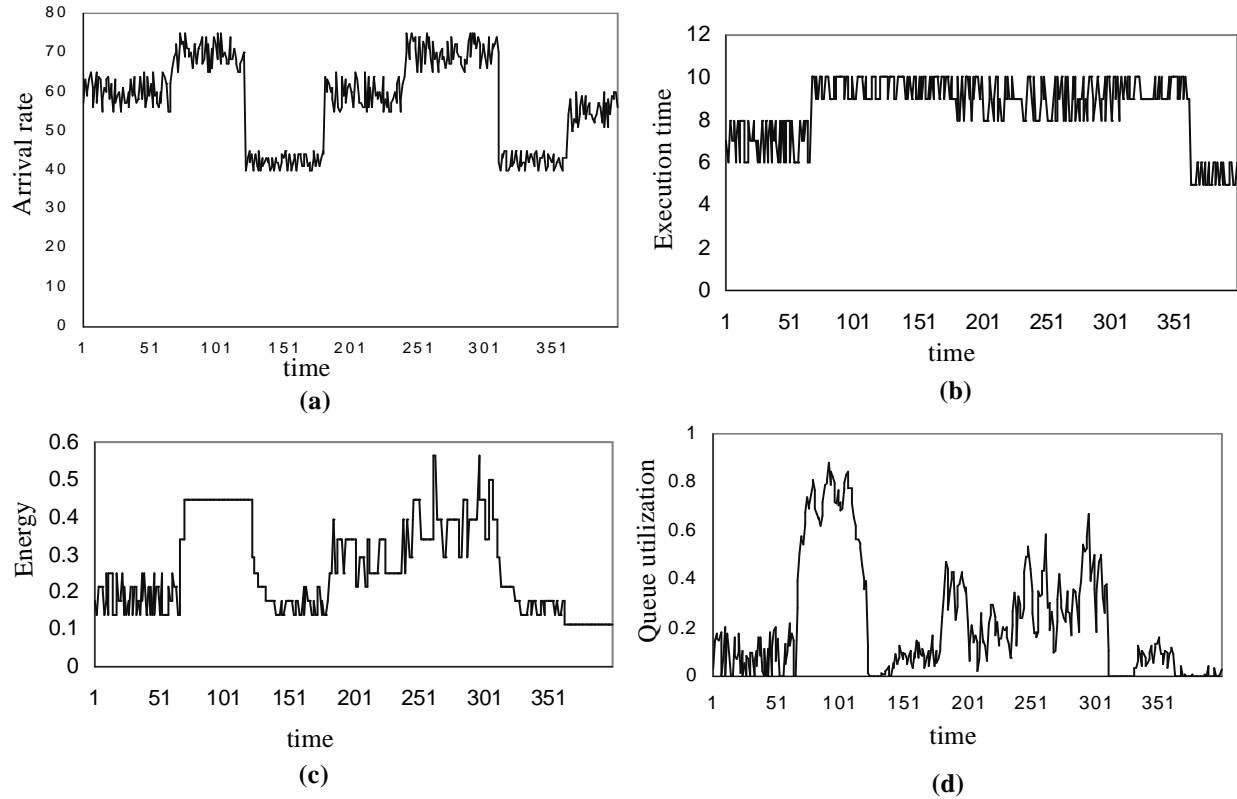
Depth d_{\max}	Number of explored states	Avg. Energy consumption	Avg. Queue Utilization	Solution cost
1	16	0.302	0.259	0.124
2	157	0.372	0.135	0.106
3	1151	0.396	0.051	0.101

(e)

Figure 4. (a) and (b) The data arrival rates and execution time, respectively, used in the simulation run; (c) processor energy consumption and (d) queue utilization; (e) Impact of d_{\max} on controller performance

operating frequency. We have described the processor model, formulated the power management problem, and derived the corresponding controller. Finally, the proposed controller was evaluated via detailed experiments using synthetic workloads and its performance was found to be satisfactory.

As part of future work, we will quantify the execution time overhead of the predictive controller on an actual embedded processor. We also plan to integrate the controller within the state machine described in Fig. 1(a) to achieve comprehensive power management of the processor.



Depth	Number of explored states	Avg. Energy consumption	Avg. Queue Utilization	Solution cost
1	16	0.262	0.220	0.104
2	142	0.300	0.107	0.070
3	994	0.321	0.050	0.068

(e)

Figure 5. The controller behavior corresponding to another simulation run; (a) and (b) data arrival rates and execution time, respectively; (c) and (d) energy consumption and queue utilization, respectively; (e) Controller performance under varying d_{\max}

6 References

- [1] Advanced Micro Devices Corp., Mobile AMD-K6-2+ Processor Data Sheet, Publication 23446, June 2000.
- [2] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, "Performance Guarantees for Web Server End-Systems: A Control Theoretic Approach," *IEEE Trans. Parallel & Distributed Syst.*, vol. 13, no. 1, pp. 80-96, January 2002.
- [3] S. Abdelwahed et al., "Online Safety Control of a Class of Hybrid Systems," *Proc. Conf. Decision & Control*, pp. 1988-1990, 2002
- [4] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, "Policy Optimization for Dynamic Power Management," *IEEE Trans. Computer-Aided Design Integrated Circuits & Syst.*, vol. 18, no. 6, pp. 813-833, June 1999.

- [5] T. D. Burd and R. W. Brodersen, "Energy Efficient CMOS Microprocessor Design," *Proc. Hawaii Int'l Conf. Syst. Sciences*, pp. 288-297, 1995.
- [6] E. F. Camacho and C. Bordons, *Model Predictive Control*, Springer-Verlag, London, 1999.
- [7] A. Cervin, J. Eker, B. Bernhardsson, and K. Arzen, "Feedback-Feedforward Scheduling of Control Tasks," *J. Real-Time Syst.*, vol. 23, no. 1-2, July/September 2002.
- [8] D. L. Hall, *Mathematical Techniques in Multisensor Data Fusion*, Artech House, Boston, 1992.
- [9] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-Line Scheduling of Hard Real-Time Tasks on Variable Voltage Processor," *Proc. Int'l Conf. Computer-Aided Design (ICCAD)*, pp. 653-656, 1998.
- [10] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Processors," *Proc. Int'l Symp. Low Power Electronics & Design (ISLPED)*, pp. 197-202, 1998.
- [11] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, New York, 1991.
- [12] M. Kim and B. Noble, "Mobile Network Estimation," *Proc. ACM Conf. Mobile Computing & Networking*, pp. 298-309, 2001.
- [13] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," *J. Real-Time Syst.*, vol. 23, no. 1-2, pp. 85-126, July/September 2002.
- [14] Z. Lu et al., "Control-Theoretic Dynamic Frequency and Voltage Scaling for Multimedia Workloads," *Proc. Int'l Conf. Compilers, Architectures, & Synthesis Embedded Syst. (CASES)*, pp. 156-163, 2002.
- [15] S. Mascolo, "Classical Control Theory for Congestion Avoidance in High-Speed Internet," *Proc. Conf. Decision & Control*, pp. 2709-2714, 1999.
- [16] K. Ogata, *Modern Control Engineering*, Prentice Hall, Englewood Cliffs, NJ, 1997.
- [17] S. Parekh et al., "Using Control Theory to Achieve Service level Objectives in Performance Management," *J. Real Time Systems*, vol. 23, no. 1-2, pp. 127-141, July/September 2002.
- [18] T. Pering, T. Burd, and R. W. Brodersen, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," *Proc. Int'l Symp. Low Power Electronics & Design (ISLPED)*, pp. 76-81, 1998.
- [19] P. Pillai and K. G. Shin, "Real-Time Voltage Scaling for Low-Power Embedded Operating Systems," *Proc. Operating Syst. Principles (SOSP)*, 2001.
- [20] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," *Proc. Conf. Mobile Computing & Networking (MOBICOM)*, pp. 251-259, 2001.
- [21] Y. Shin and K. Choi, "Power Conscious Fixed-Priority Scheduling for Hard Real-Time Systems," *Proc. Design Automation Conf.*, pp. 134-139, 1999.
- [22] T. Simunic and S. Boyd, "Managing Power Consumption in Networks on Chips," *Proc. Design, Automation, & Test Europe (DATE)*, pp. 110-116, 2002.
- [23] A. Sinha and A. P. Chandrakasan, "Energy Efficient Real-Time Scheduling," *Proc. Int'l Conf. Computer Aided Design (ICCAD)*, pp. 458-463, 2001.
- [24] A. Sinha and A. P. Chandrakasan, "Dynamic Power Management in Wireless Sensor Networks," *IEEE Design & Test Computers*, vol. 18, no. 2, pp. 62-74, March/April 2001.
- [25] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive System Shutdown and Other Architectural Techniques for Energy-Efficient Programmable Computation," *IEEE Trans. VLSI Syst.*, vol. 4, no. 1, pp. 42-55, March 1996.