Feedback Thermal Control of Real-time Systems on Multicore Processors

Yong Fu[1], Nicholas Kottenstette[2], Chenyang Lu[1], Xenofon D. Koutsoukos [2]
[1]Dept. of CSE, Washington University, St. Louis, MO, {fuy, lu}@cse.wustl.edu
[2]Dept. of EECS, Vanderbilt University, Nashville, TN, {nkottens, Xenofon.Koutsoukos}@vanderbilt.edu

ABSTRACT

Embedded real-time systems face significant challenges in thermal management. While earlier research on feedback thermal control has shown promise in dealing with the uncertainty in thermal characteristics, multicore processors introduce new challenges that cannot be handled by previous solutions designed for single-core processors. Multicore processors require the temperature and real-time performance of *multiple* cores be controlled simultaneously, leading to multi-input-multi-output control problems with inter-core thermal coupling. Furthermore, current Dynamic Voltage and Frequency Scaling (DVFS) mechanisms only support a finite set of states, leading to *discrete* control variables that cannot be handled by standard linear control techniques. This paper presents *Real-Time Multicore Thermal Control (RT-MTC)*, a novel feedback thermal control framework specifically designed for multicore real-time systems. RT-MTC dynamically enforces both the desired temperature set point and the schedulable CPU utilization bound of a multicore processor through DVFS. RT-MTC employs a rigorously designed, efficient controller that can achieve effective thermal control with the small number of frequencies commonly supported by current processors. The robustness and advantages of RT-MTC over existing thermal control approaches are demonstrated through both experiments on an Intel Core 2 Duo processor and simulations under a wide range of uncertainties in power consumption.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Real-time and embedded systems; H3.4 [**Information Systems**]: System and Software—*performance evaluation*

General Terms

Algorithm

EMSOFT'12, October 7-12, 2012, Tampere, Finland.

Keywords

real-time systems, multicore, thermal control

1. INTRODUCTION

Embedded real-time systems face significant challenges in thermal management as they adopt modern computing platforms with increasing power density. While traditional embedded real-time systems typically run on single-core low-power microcontrollers, the increasing complexity of real-time applications demands the adoption of modern multicore microprocessors to leverage their computing power. Such systems must avoid processor overheating while maintaining desired real-time performance. The need to enforce temperature bounds can conflict with the need to meet real-time performance requirements, because thermal management mechanisms such as Dynamic Voltage and Frequency Scaling (DVFS) reduce processor speed resulting in prolonged execution times for real-time tasks. While modern processors usually rely on hardware throttling mechanisms to prevent overheating, such mechanisms can cause severe performance degradation unacceptable to real-time applications. Moreover, modern processors can exhibit significant *uncertainties* in their power and thermal characteristics. For instance, the power consumption of a processor may vary significantly when running different applications due to the different sets of instructions executed [18].

In recent years, control-theoretic thermal management approaches have shown promise in [8,11,12,21,34,35,37] handling uncertainties in thermal characteristics. In contrast to heuristic-based design relying on trial-and-error, control-theoretic approaches provide a scientific framework for systematic design and analysis of thermal control algorithms. However, previous research on feedback thermal control for embedded real-time systems focused on single-core processors and cannot handle the practical limitations of multicore processors. Thermal management mechanisms such as DVFS only support a finite set of states, leading to *discrete* control variables that cannot be handled by standard linear control techniques. Moreover, multicore processors require the temperatures and real-time performance of *multiple* cores to be controlled simultaneously, leading to multi-input-multi-output (MIMO) control problems with inter-core thermal coupling.

We present *Real-Time Multicore Thermal Control (RT-MTC)*, a novel feedback thermal control algorithm specifically designed to meet the challenges posed by multicore processors. RT-MTC employs a feedback control loop that enforces the desired temperature and CPU utilization bounds

of embedded real-time systems through DVFS. RT-MTC employs an efficient and robust control design that integrates three components.

- a robust nonlinear proportional controller that deals with uncertainties in power consumption;
- a saturation block for the controller output that enforces the schedulable utilization bound;
- a Pulse Width Modulation (PWM) component that achieves desired control input by dynamically switching between discrete voltage/frequency levels.

RT-MTC combines a control-theoretic approach and a practical design. In contrast to heuristics-based solutions relying on extensive testing and hand tuning, we provide control-theoretic analysis of the stability and robustness of RT-MTC under uncertainties in power consumption. At the same time, RT-MTC employs a simple and efficient control algorithm suitable for run-time execution. Moreover, RT-MTC can be easily implemented in the user space without modification to the OS kernel which is usually required by traditional thermal-aware real-time scheduling approaches. The robustness and advantages of RT-MTC over existing thermal control approaches are demonstrated through implementation on Linux and experiments on an Intel Core 2 Dual processor as well as extensive simulations with varying power consumption.

2. RELATED WORKS

There has been significant work on thermal aware real-time scheduling for both single-core processors [7, 33] and multicore processors [5, 6, 9]. Those algorithms rely on accurate models about the thermal characteristics of the processors, and hence cannot effectively deal with uncertainties in thermal characteristics such as power consumption and ambient temperature. Moreover, they usually require fine-grained scheduling decisions that require kernel-level implementations. In contrast, our feedback control approach is implemented in user space without modifications to the kernel and therefore can be easily deployed in existing systems.

Control-theoretic thermal management has been explored for non-real-time systems. Donald and Martonosi present a general framework of dynamic thermal management for multicore processors [8]. Essentially, the proposed framework is a hierarchical feedback control loop with PI controllers, but it does not provide real-time performance guarantees. Several papers [26, 34–37] have adopted model predictive control or online convex optimization for dynamic thermal management. None of these works is concerned with maintaining real-time performance. In addition, control approaches based on model predictive control and convex optimization has higher computation complexity than our efficient proportional control approach. Moreover, our approach deals with *discrete* voltage/frequency levels, a practical issue associated with DVFS which is ignored by the aforementioned control solutions [26, 35, 37].

Control-theoretic approaches have recently been proposed for thermal management of real-time systems [12, 21]. Our previous work [12] proposed a feedback control algorithm that enforces thermal and real-time constraints simultaneously. That work adjusts the rate of periodic real-time tasks as the control knob, whereas RT-MTC employs DVFS that does not require applications to support variable rates. Lindberg [21] proposed a feedback control framework to manage both temperature and media performance. Both algorithms [12, 21] are designed for single-core processors and cannot deal with multicore processors as they are not cognizant of inter-core thermal coupling in multicore processors.

Different from prior research handle thermal management on hardware level [4, 16, 30, 31], RT-MTC mainly focus on system level thermal management of multicore processors. Two aspects differentiate hardware and system level thermal management. First, thermal dynamics on hardware level is faster, with time constant at milliseconds [4]. In contrast at system level thermal dynamics of the processor is relative slow and with time constant in seconds [15]. Second, hardware thermal management usually adopt low level control knobs, e.g., clock gating or pipeline throttling, which can not be exposed as system level interfaces. In contrast, system level thermal management employs high-level knobs, e.g., DVFS, that are supported by most operating systems.

3. PROBLEM FORMULATION

We assume a common embedded real-time system model where the workload consists of real-time tasks released periodically. A embedded real-time system comprises a set of periodic real-time tasks running on a multicore processor with m homogeneous cores. The processor supports Dynamic Voltage and Frequency Scaling (DVFS). We assume two common characteristics of DVFS in mainstream multicore processors (e.g., Intel Core2, i5, i7 and Atom). First, the frequency and voltage of all the cores can only be scaled *uniformly*, i.e., all cores always share the *same* frequency and voltage. Second, the processor only supports a *discrete* set of frequencies. New challenges are posed by The dicretization and nonlinearity introduced by both assumptions pose key challenges to thermal control design that were not addressed in previous works [26, 34–37].

We assume *partitioned* multicore real-time scheduling, under which tasks are statically partitioned and bound to processor cores. There is a real-time tasks set $\mathbb{S}$ with n independent, periodic real-time tasks for the processor. For core l, there is a task set $\mathbb{S}_l \subseteq \mathbb{S}$ with n_l real-time tasks. Each task s_i in the task set $\mathbb{S}_l$ has a period p_i, a soft deadline d_i, and a worst-case execution time c_i. The utilization of an individual core l is thus $U_l = \sum_{s_j \in \mathbb{S}_l} \frac{c_j}{p_j}$.

We assume the tasks on a core are scheduled locally based on a real-time scheduling policy with a known schedulable utilization bound U_b, e.g., Rate Monotonic (RM) or Earliest Deadline First (EDF) under certain conditions [22]. The tasks on a core l meet their deadlines if $U_l \leq U_b$. The system can therefore guarantee the schedulability of all the tasks on a core by enforcing the schedulable utilization bound. [1]

Given a embedded real-time system running on a multicore processor, our problem is to control the temperature of the processor such that the *maximum* temperature among all the cores tracks a temperature set point, y_s, subject to the constraint of utilization bound U_b on each processor core. The temperature set point y_s is the desired temperature below the maximum temperature tolerable by the proces-

[1] Our approach can be extended to support a mixed task set containing periodic and soft real-time aperiodic tasks via well known aperiodic server mechanisms [23] by enforcing appropriate schedulable utilization bounds.

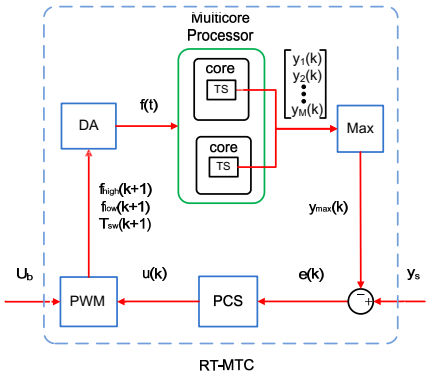


Figure 1: Feedback Control Loop of RT-MTC

sor. Our control problem formulation therefore aims to meet both the thermal and real-time performance requirements of a embedded real-time system.

4. OVERVIEW OF RT-MTC

The feedback control loop of RT-MTC, shown in Fig. 1, consists of a Temperature Sensor (TS) for each core, a Proportional Controller with Saturation (PCS), Pulse Width Modulation (PWM), and a DVFS Actuator (DA). The user input to RT-MTC is the desired temperature set point y_s and the utilization bound U_b. The feedback control loop is invoked periodically at the end of every sampling period. Specifically, at the end of k^{th} sampling period, RT-MTC performs the following operations:

1. The TS on each core measures the temperature of the core i, $y_i(k)$. The Max function calculates the maximum temperature among all cores and feeds the maximum temperature $y_{\max}(k)$ among all the cores to the PCS.

2. The PCS computes the controller output $u(k)$ as follows:

$$u(k) = \begin{cases} 1, \text{ if } k_p e(k) > 1, \\ -1, \text{ if } k_p e(k) < -1, \\ k_p e(k), \text{ otherwise;} \end{cases} \quad (1)$$

   where k_p is the coefficient of proportional control and $e(k) = y_s - y_{\max}(k)$. The output of the controller is limited to the range $[-1, 1]$. The PCS design is discussed in more details in Section 6.1.

3. The PWM receives the controller output $u(k)$ and calculates a pair of frequencies $f_{high}(k+1), f_{low}(k+1)$ and the switching time $T_{sw}(k+1)$. Details of calculating $f_{high}(k+1), f_{low}(k+1), T_{sw}(k+1)$ are presented in Section 5.2.

4. The DA adjusts the frequency of the multicore processor via the DVFS interface according to the $(f_{high}(k+1), f_{low}(k+1), T_{sw}(k+1))$ input from the PWM. Specifically, at $T_{sw}(k+1)$ seconds after the beginning of the current sampling period, the processor switches its frequency from $f_{high}(k+1)$ to $f_{low}(k+1)$. The implementation of DA is detailed in Section 7.

5. THERMAL DYNAMIC MODEL

As the first step of control design and analysis, we now present a difference equation model to characterize the relationship between the frequency and the temperature. We construct the model in three steps. We first caputre the power consumption. Based on a well known power model, we then characterize the impact of PWM on the power consumption model. Finally, we complete the system model by incorporating a widely used thermal RC model that characterizes the relationship between power consumption and temperature.

We note that our system model is necessarily a simplification of the actual system's thermal behavior for the purpose of control-theoretic design and analysis. The inherent robustness of feedback control enables our system to handle considerable modeling errors in model parameters, as demonstrated in our evaluation (Sec. 8.1.2).

5.1 Power Model

As shown in [12], the average power $\bar{P}(k)$ of a core in the k^{th} sampling period can be modeled as

$$\bar{P}(k) = U(k)P_{\text{act}}(k) + (1 - U(k))P_{\text{idle}}(k)$$

where $U(k)$ is the CPU utilization of the core, $P_{\text{act}}(k)$ is the active power, and $P_{\text{idle}}(k)$ is the idle power in k^{th} sampling period. $P_{\text{idle}}(k)$ can be approximated by a piecewise linear model $P_{\text{idle}} = (C_0(V(k)) + C_1(V(k))y(k))V(k)$ [28]. A well-known model of the active power is $P_{\text{act}}(k) = C_2V^3(k)$, where C_2 is a constant coefficient and $V(k)$ is the supply voltage [29].

We can rewrite the average power as

$$\bar{P}(k) = \bar{P}_a(k) + C_y y(k) \quad (2)$$

where $\bar{P}_a(k) = U(k)C_2V^3(k) + C_0(V(k))V(k)$ and $C_y = C_1(V(k))$. $\bar{P}_a(k)$ and C_y can be expressed in terms of the frequency, based on the relationship between supply voltage and frequency, $V(k) = Kf(k) + V_{\text{th}}$ [20] and $\frac{U(k)}{f(k)} = \frac{U_0}{f_0}$ where U_0 and f_0 are the initial CPU utilization and frequency. Note we assume that the processor utilization scales proportionally with the frequency which usually hold for those CPU bound applications.

5.2 Pulse Width Modulation (PWM)

As each core of the multicore processor runs under a discrete set of frequencies, the power $\bar{P}_a(k)$ in equation (2) can only switch between discrete levels. To track the temperature set point closely, PWM is employed to map desired average power in each sampling period to the discrete frequency levels supported by the processor.

The continuous input to the PWM in the k^{th} sampling period is $u(k) \in [-1, 1]$. The PWM computes $(f_{high}(k+1), f_{low}(k+1), T_{sw}(k+1))$ based on $u(k)$. The upper limit of the output corresponds to the maximum frequency supported by the processor. The lower limit of the output corresponds to the lowest frequency that satisfies the utilization bound or the minimum frequency, whichever is higher. Let the frequency corresponding to the upper and lower limit of $u(k)$ be $f_{\max}, f_{\min}$, and let $f_u(k) = f_{\min} + (f_{\max} - f_{\min})\frac{u(k)+1}{2}$. To minimize the change in CPU speed, PWM first chooses a pair of *consecutive* frequency levels f_i and f_{i+1} which satisfy $f_i \le f_u(k) \le f_{i+1}$ from the supported discrete frequency set; these are designated $f_{low}(k+1)$ and

$f_{high}(k+1)$ respectively. The time to switch from $f_{high}(k+1)$ to $f_{low}(k+1)$ is computed as

$$T_{sw} = \frac{f_u(k) - f_{low}(k+1)}{f_{high}(k+1) - f_{low}(k+1)} T_s,$$

where T_s is the sampling period. Note if $f_u(k)$ equals any frequency in the supported frequency set, both $f_{high}(k+1), f_{low}(k+1)$ will exactly equals that frequency and $T_{sw} = 0$.

Let $\bar{P}_{a,max}$, $\bar{P}_{a,min}$ be the upper and lower bound of $\bar{P}_a$, which are the average power consumption at $f_{\max}$ and $f_{\min}$, respectively. We can rewrite the power model to incorporate PWM based on (2) as

$$\bar{P}(k) = G_p(P_{ap}u(k) + P_{am}) + C_y y(k) \tag{3}$$

where $P_{ap} = (\bar{P}_{a,max} - \bar{P}_{a,min})/2$, $P_{am} = (\bar{P}_{a,max} + \bar{P}_{a,min})/2$, and G_p is the gain to represent the uncertainty caused by power variation.

The power consumption model (3) approximates the power behavior of the processor, since it derives the average power rather than actual power. However, as we shown in our stability analysis (Section 6.1) and experiments (Section 8.1.2), the inherent robustness of our feedback control design can tolerate considerable modeling error without compromising system stability.

5.3 Thermal Dynamic Model

Our control design is based on a well-established thermal RC model for multicore processors with M cores and a heat sink [9]. Compared to architecture-level thermal models such as Hotspot [17], the model presented here is simpler but more suitable for control design of thermal management. The effectiveness of the model has been validated in [9, 29].

Symbol	Meaning
$R_i, R_h, R_a, R_{i,j}$	thermal resistance of the core i, the heat sink, environment and thermal resistance between the core i and j
C_i, C_h	thermal capacitance of the core i and the heat sink
y_0, y_i, y_h	temperature of environment, the core i and the heat sink
P_i	power of the core i
$\mathbb{N}_i$	the set of cores adjacent the core i

Table 1: Symbols in Thermal Dynamic Model

Based on the symbols listed in Tab. 1, the thermal dynamic model of the multicore processor can be written in the following compact form:

$$\dot{\mathbf{Y}}(t) = A\mathbf{Y}(t) + B_P\mathbf{P}(t) + B_y y_0 \tag{4}$$

where $\mathbf{Y}(t) = [y_1(t), \ldots, y_M(t), y_h(t)]^\intercal \in \mathbb{R}^{M+1}$, $\mathbf{P}(t) = [P_1(t), \ldots, P_M(t)]^\intercal \in \mathbb{R}^M$ and y_0 is the ambient temperature, $A \in \mathbb{R}^{(M+1)\times(M+1)}$, $B_P \in \mathbb{R}^{(M+1)\times M}$ and $B_y \in \mathbb{R}^{(M+1)}$. The matrices A, B_P and B_y are computed as follows:

$$A(i,j) = \begin{cases} \frac{-1}{C_i}\left(\frac{1}{R_i} + \sum_{m\in\mathbb{N}_i}\frac{1}{R_{i,m}}\right), \text{ if } i = j \neq (M+1) \\ \frac{1}{R_{i,j}C_i}, \text{ if } j \in \mathbb{N}_i \\ \frac{1}{R_iC_i}, \text{ if } i \neq (M+1) \text{ and } j = (M+1) \\ \frac{1}{R_jC_h}, \text{ if } i = (M+1) \text{ and } j \neq i \\ \frac{-1}{C_h}\left(\frac{1}{R_a+R_h} + \sum_{m=1}^{M}\frac{1}{R_m}\right) \text{ if } i = j = (M+1) \\ 0, \text{ otherwise.} \end{cases},$$

$$B_P(i,j) = \begin{cases} \frac{1}{C_i}, \text{ if } i = j \\ 0, \text{ otherwise.} \end{cases},$$

$$B_y(i) = \begin{cases} \frac{1}{C_h(R_a+R_h)}, \text{ if } i = M+1 \\ 0, \text{ otherwise.} \end{cases}.$$

We use a Zero Order Hold (ZOH) equivalent model [10] in which the average power-model for $\bar{P}(k)$ is assumed to be held constant and the average environmental temperature is $y_0(k) = \frac{1}{T_s}\int_{kT_s}^{(k+1)T_s} y_0(t)dt$ during the k^{th} sampling period. The ZOH equivalent of (4) is

$$\mathbf{Y}(k+1) = \Phi_o\mathbf{Y}(k) + \Psi_P\bar{\mathbf{P}}(k) + \Psi_y y_0(k) \tag{5}$$

where $\Phi_o = e^{AT_s}$, $\Psi_P = \left(\int_0^{T_s} e^{A\tau}d\tau\right)B_P$, $\Psi_y = \left(\int_0^{T_s} e^{A\tau}d\tau\right)B_y$ and $\bar{\mathbf{P}}(k) = [\bar{P}_1(k), \ldots, \bar{P}_M(k)]^\intercal \in \mathbb{R}^M$. Substituting the power model (3) for $\bar{P}(k)$ in (5) results in:

$$\mathbf{Y}(k+1) = \Phi\mathbf{Y}(k) + P_{ap}\Psi_P G_p u(k) + \Psi_y y_0(k) + P_{am}\Psi_P G_p \tag{6}$$

in which $\Phi = \left(\Phi_o + C_y\Psi_P\begin{bmatrix} I_M & 0 \end{bmatrix}\right)$ where $\begin{bmatrix} I_M & 0 \end{bmatrix} \in \mathbb{R}^{M\times(M+1)}$ and $I_M \in \mathbb{R}^{M\times M}$ denotes the identity matrix. The term involving $y_0(k)$ relates how environmental temperature changes can perturb the system. The last term represents a fixed-disturbance due to the mean active power resulting from our proposed modulation approach.

In practice the model parameters can be estimated using well-known system identification method. Essentially, there are two methods to acquire the parameters of the compact thermal model. We can either extract the parameters based on fine grain thermal RC models, for example Hotspot [17] or estimate the parameters using realistic operational data, which is also the method we used in this paper. The detailed description of model identification is presented in Section 8.1.1.

6. CONTROL DESIGN

We propose a low-complexity controller to tackle the problem of thermal management of real-time systems on multicore processors. Our control design ensures that the maximum temperature of the cores tracks the thermal set-point without violating the utilization constraints. Although the control structure shown in Fig. 1 only has single input, the PCS must control the temperature of multiple cores simultaneously. Previous approaches to thermal control for the single core processor [12] is not suitable to multicore thermal control because their control design do not handle the interaction among the thermal dynamics of different cores. In this section we present a control design which can handle not only thermal coupling among cores but also other nonlinearities induced by the multicore processors.

6.1 Stability Analysis and Control Design

The PCS is designed based on passivity [27] and can accommodate the nonlinearities induced by the *Max* function and the saturation. There are various precise mathematical definitions for passive systems that essentially state that the output energy must be bounded so that the system does not produce more energy than was initially stored. Under certain technical conditions, strictly input and strictly output passive systems are Lyapunov stable [32]. In this case, passivity offers advantages for computing a Lyapunov function that is used to prove stability of the closed-loop system.

In order to analyze the stability of RT-MTC, we assume that the set-point $T_b = 0$ and we consider the unperturbed system where $y_0 = 0$, $\Psi_P G_P = 0$ in (6). We provide sufficient conditions that ensure the existence of a Lyapunov function for the closed loop system, and thus, stability of the RT-MTC. A detailed proof can be found in [13]. The disturbance in the power model arises because of (1) the ambient temperature that can change but is measurable and (2) the mean active power introduced by the PWM. We can minimize the steady-state error by taking into account these terms in the set-point T_b (the detailed derivation of T_b can be found in [13]).

THEOREM 1. *Consider the closed-loop system shown in Fig. 1 with $T_b = 0$ and assume that the power model of the multicore processor is described by (6) with $y_0(k) = 0$ and $P_{am}\Psi_P G_P = 0$. If there exists a matrix $P = P^\mathsf{T} > 0$ and $-\infty < \delta < 0$ such that the following LMI is satisfied:*

$$\begin{bmatrix} \Phi^\mathsf{T} P \Phi - P & \Phi^\mathsf{T} P P_{ap} \Psi_P G_p - \frac{1}{2} C_l^\mathsf{T} \\ \left(\Phi^\mathsf{T} P P_{ap} \Psi_P G_p - \frac{1}{2} C_l^\mathsf{T}\right)^\mathsf{T} & \delta + P_{ap}^2 G_p^\mathsf{T} \Psi_P^\mathsf{T} P \Psi_P G_p \end{bmatrix} \leq 0 \tag{7}$$

for all $l \in \{1, \ldots, M\}$, where C_l is the coefficient for the measured temperature of the core l, then the closed-loop system is passive and stable.

By exploring the solution of the LMI (7) given in Theorem 1, we can acquire the stability condition of the system under modeling error. Specifically, for items in the search space of power gain, thermal resistance and capacitance, we can check whether the LMI is solvable and then decide whether the closed-loop system is stable with the parameters. Accordingly we derive robustness of the system in terms of the range of uncertain parameters, power gain and thermal related parameters resulting in stable systems.

The above theorem can also be used for designing the controller. This is achieved by finding the smallest value of δ that satisfies the LMI (7), The controller gain of the PCS (equation (6)) is defined as $k = -\frac{1}{\delta}$. This is the highest proportional gain that guarantees stability of the closed-loop system. In general, higher controller gain improves control performance. If there is deviation from the set point, high gain controller ensures that the system will converge to the set-point as fast as possible. The LMI shown in the theorem can be solved efficiently using standard LMI tools such as the Matlab LMI toolbox and the Scilab lmitool.

7. IMPLEMENTATION OF RT-MTC

We have implemented RT-MTC on top of Linux, using a combination of Python, MATLAB, and C. The PCS, PWM, DVFS Actuator, and Max components shown in Fig. 1 are written in Python.

All the components in the feedback control loop are implemented in one process assigned the highest real-time process priority so that RT-MTC can be executed periodically with minimum interference from real-time tasks.

Thermal Sensor: Most modern multicore processors are equipped with hardware thermal sensors for each individual core, which are supported by the operating system or third-party libraries. For example, in Linux, the temperature of cores can be read from the interface provided by *lmsensor* [3] via the coretemp driver (/sys/bus/platform/drivers/coretemp/). The thermal information can also be acquired from standard ACPI interfaces. For those multicore processors without thermal sensors on each core, such as those used in embedded systems, soft thermal sensors [19] can be employed to estimate the temperature of a single core.

PCS and PWM: The implementations of PCS and PWM are straightforward, based on the description in Sec. 6 and Sec. 5.2.

DVFS Actuator: We implemented the DVFS Actuator using the *signal* mechanism provided by POSIX interface. First, an alarm is set to be fired at the switching time T_{sw} by using the POSIX *alarm* function. When the alarm expires, a *SIGALRM* signal is sent to the process's signal handler set by the function *sigact*. The signal handler calls a procedure to switch the frequency of the multicore processor from the high level f_{high} to the low level f_{low} via a interface which can access the processor's DVFS function, for examples, ACPI, lmsensor or Machine Specific Register. The delay between PWM output switching time T_{sw} and the time that the frequency is actually switched relies on the resolution of clock interrupt of the underlying operating system. For example, the Linux kernel uses a configurable time resolution (known as *jiffy*) which ranges from $1ms$ to $10ms$. Even at a resolution of $10ms$, the delay has negligible effect on the control performance, since it is comparatively much shorter than the sampling period. We choose $10s$ as the sampling period in our implementation because it is short enough to control the thermal behavior of the processor , which has time constant greater than $100s$, without imposing singnificant overhead from frequency switching and computation.

8. EVALUATION

We first evaluate RT-MTC through experiments based on above implementation and then perform extensive simulations with parameters acquired from model identification experiments. An Intel Core 2 Duo two core processor is used to run the experiments and be the target of simulations as it provides discrete DVFS mechanism. Moreover thermal parameters, especially thermal capacitance, of Intel Core2 Duo are acquired directly as shown later. The simulations complement experimental results by allowing us to examine RT-MTC's performance under stress-test conditions (such as fan failure) which are difficult or dangerous to run on real hardware.

8.1 Experiments

The hardware platform used for the experiments is a Lenovo W500 laptop with an Intel T9400 Core 2 Duo dual core processor and the Linux kernel 2.6.32 distributed with Fedora 12.[2] The T9400 processor has 2 digital thermal sensors lo-

[2]Although we only present the results of experiments for a dual core processor, the methodology and implementation can be ex-

cated on each core and supports processor-wide DVFS, that is, the two cores' frequencies must be set uniformly. The DVFS frequencies and the thermal properties of the T9400 are listed in Table 2.

Frequency	2.53, 1.6, 0.8 GHz
Voltage	1.175, 1.00, 0.900 V
T_{junc}	$105^{\circ}C$
Thermal Design Power (TDP)	35W

Table 2: Frequencies and Thermal Properties of the T9400 Processor

8.1.1 Model Identification

To acquire the parameters of the thermal RC model, we first run a set of real-time workloads to profile the processor's thermal behavior. Then the thermal parameters is identified from the experiments results by Matlab Model Identification Toolbox. The real-time workloads used for model identification involves two micro benchmarks, CRC and Bzip2. CRC is a data verification application chosen from Mibench [14], a test suite for embedded systems. Bzip2 is a data compression tool chosen from SPEC CPU 2006 [2], a standard benchmarks suite. We implement three kinds of workloads: CRC alone, Bzip2 alone and a Mixed workload containing both microbenchmarks. The workload for each core is identical and involves 5 periodic tasks which are either CRC or Bzip2 according to the type of the workload. The deadlines of the tasks are set to the same as their periods. The periods and execution time of the tasks are listed in Table 3.

	Task 1	Task 2	Task 3	Task 4	Task 5
Period	250	300	450	500	1000
Exe. Time	23	27	41	45	90

Table 3: Workload Tasks Period and Execution Time@$2.53GHz$ (ms)

To capture the comprehensive thermal behavior for different frequencies, we employ a pseudo-sequence of frequency as input, where frequency switches between $2.53GHz$ and $0.8GHz$. Considering the large time constant of the processor's thermal behavior, we run each workload for 5400s. Table 4 shows the results of the model identification via Matlab Model Identification Toolbox. Fig. 2 illustrates the temperature and frequency of the Mixed workload; the other two workloads are omitted here due to space constraints.

There are two important observations from Table 4. First, it indicates the efficacy of the thermal dynamic model, as the estimated model parameters result in fitness levels above 80% for all three workloads. Second, the model parameters estimated under different workload differ considerably. This entails that thermal control must be robust against uncertainties of model parameters caused by different workloads since it is unrealistic to expect users to re-estimate the parameters via system identification for every workload. Such robustness against modeling errors is an important advantage of RT-MTC, as shown in both the empirical results and the simulation study presented below.

tended to the processor with more than two cores easily since control design proposed in this paper is based on a general multicore processor model.

Thermal Parameters (Mixed, Fit*: 82%)					
$R_1(\Omega)$	1.61	$C_h(F)$	216.74	$R_{12}(\Omega)$	16.16
$R_2(\Omega)$	1.46	$C_2(F)$	1.25	$C_1(F)$	1.25
$R_a + R_h$	1.05				
Thermal Parameters (Bzip2, Fit:83%)					
$R_1(\Omega)$	1.35	$C_h(F)$	263.02	$R_{12}(\Omega)$	15.23
$R_2(\Omega)$	1.13	$C_2(F)$	1.61	$C_1(F)$	1.61
$R_a + R_h$	1.35				
Thermal Parameters (CRC, Fit: 81%)					
$R_1(\Omega)$	1.78	$C_h(F)$	242.23	$R_{12}(\Omega)$	16.83
$R_2(\Omega)$	1.56	$C_2(F)$	1.35	$C_1(F)$	1.35
$R_a + R_h$	1.08				

*: the accuracy index in Matlab Model Identification Toolbox.

Table 4: Results of Model Identification

8.1.2 Experiment Results

In this section we present the experimental results of RT-MTC on the real hardware platform. We run RT-MTC under the workload of the CRC and the Mixed for 10 minutes each. The controller parameters of RT-MTC are computed using the thermal RC model parameters of the Mixed workload. In this experiment we choose the temperature set point as $60^{\circ}C$ to ensure that internal thermal throttling circuit is not activated even when there is overshoot during temperature adjustment.

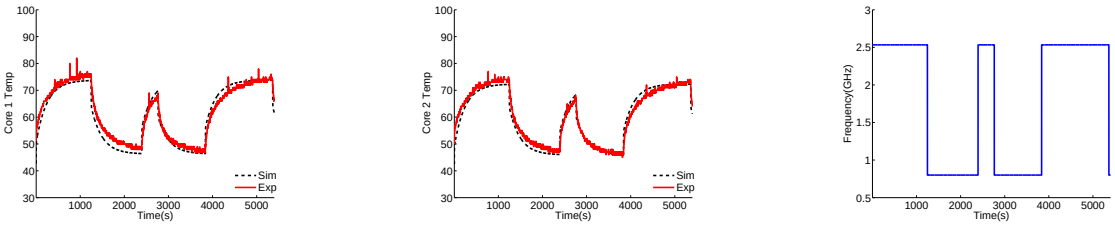
Two important observations can be made from the results plotted in Fig. 3. First, RT-MTC enforces both the temperature set point and the utilization bound. As seen in Fig. 3(b), after $280s$ the temperature is steady at the temperature set point, $60^{\circ}C$. The average upper limit of the utilization is 74%, which is below the utilization bound. Second, RT-MTC (with the same control parameters) can control the thermal behavior of the processor effectively under *both* test workloads. As shown in Table 4, there is difference between the parameters identified by the Mixed and the CRC workloads, which induces modeling error. Ensuring temperature set point in both cases shows RT-MTC robustness against modeling error induced by different workloads. Although there are spikes in temperature during the CRC workload caused by background services (which cannot be manipulated by our user-space implementation), RT-MTC quickly counteracts these spikes.

8.2 Simulation

We perform extensive simulations based on the model parameters identified from the experiments presented in Sec. 8.1.1 . Although we wish to explore the performance of RT-MTC in extreme scenarios, it is often impractical to carry such experiments out on real hardware. For example, an experiment int RT-MTC's performance in the face of fan failure would be likely to damage the processor. For this reason, we stress-test the performance of RT-MTC under simulation, as discussed in this section.

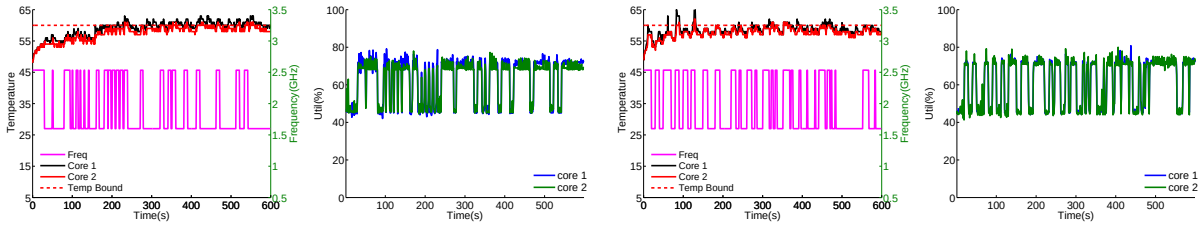
8.2.1 Simulation Setup

There are two components in our simulation environment:



(a) Core 1 Temperature (b) Core 2 Temperature (c) Frequency

Figure 2: Model Identification Data (Mixed Workload)



(a) Mixed Workload (b) CRC

Figure 3: Experimental Results of RT-MTC

an event driven simulator implemented in C++ and a Simulink module implemented in MATLAB (R2008a). The C++ simulator simulates embedded real-time systems over multicore processors and calculates the processor utilization according to the frequency output by the controller. The Simulink module performs the controller's computation. And the Simulink module also calculates the temperatures of multicore processors based on the utilization generated by the C++ simulator. The C++ simulator and the Simulink module communicate with each other through a TCP connection.

The target multicore processor in our simulation is the dual core processor, Intel Core 2 Duo T7200 [1]. The power and thermal related parameters of T7200 are shown in Table 5. The parameters of the leakage power model are acquired by linear approximation of an accurate leakage power model [24]. The active power and available frequencies are obtained from Intel T7200 data sheet [1]. Note that although the evaluation is only preformed on the dual-core processor, our approach for thermal management is developed for general multicore processors and therefore can handle the processors with more cores.

We use the same methodology and tools for model identification as described in Sec. 8.1.1. The acquired thermal parameters are listed in Table 5. As thermal design is different between manufacturers, it is reasonable that these parameters identified vary significantly from those identified for the T9400.

In the simulations we use a fine-grained workload which runs 10 periodic soft real-time tasks on each core. We assume partitioned scheduling for the multicore embedded real-time systems. The Rate Monotonic (RM) scheduling algorithm [22] is employed to schedule all tasks on each core. The utilization bound is set to 0.71. At the beginning of the experiment, the period of each task T_i is randomly generated in the range $[100ms, 200ms]$. The execution time of each task is generated to keep each task's utilization nearly equal and the sum of all tasks' utilization at 0.7, just below the utilization bound.

Power Parameters					
f(GHz)	0.8,	1.2,	1.6,	2.0	
C_0	-0.3638,	-0.3687,	0.1071,	2.3367	
C_1	0.0191,	0.0342,	0.0608,	0.1066	
C_2	7.7378				
Thermal Parameters					
$R_1(\Omega)$	0.53	$C_h(F)$	390	$R_{12}(\Omega)$	5.5
$R_2(\Omega)$	0.57	$C_2(F)$	39.14	$C_1(F)$	50.38
$R_a + R_h$	0.2				

Table 5: Simulation Parameters

In the following simulations, we set the temperature bound to $60°C$, below the temperature achieved by the Thermal Design Power (TDP) of T7200 so as not to activate the internal hardware thermal regulation. Note that the effectiveness of our approach does not rely on the specific temperature bound.

We compare RT-MTC against four other baseline algorithms, OPEN, Reactive, MPC-QUAN and MPC-PWM. OPEN statically sets the processors' frequency at beginning of the simulation and does not change it while the simulation runs.

MPC-QUAN and MPC-PWM are control-theoretic approaches and based on the algorithm proposed in [35]. The control algorithms of both baselines are the solutions of the following constraint optimizing problem with the optimizing objective as follows:

$$J(k) = \sum_{i=1}^{H_p} |y_{max}(k+i) - y_s|^2 \quad (8)$$

where H_p is the prediction horizon and y_s is the temperature set point. The solution of the optimizing problem also needs to satisfy the constraints of the utilization bound, the thermal bound, and the frequency limit. Note that $T(k)$ must

follow the thermal model (5). The solution of the constraint optimizing problem (8) is a vector with length of H_p. The first element of the solution is employed as control output. The pulse width modulation transforms the control output of the power to the duty cycle of the power signal. MPC-QUAN rounds off the control output, aforementioned as the final output while MPC-PWM employs a PWM mechanism described in the previous section to approximate the control output.

The baseline Reactive (Reactive Thermal Control) is a modified version of reactive speed control of embedded real-time systems [33]. The key design point of Reactive is that whenever the thermal threshold is hit, the frequency corresponding to equilibrium temperature (thermal bound in our case) is applied. Otherwise, the highest available frequency is applied. The original version of reactive speed control works at the level of tasks, that is, the frequency changes during the duration of one task running. Reactive, however, only changes frequency at the end of a sampling period. If all the parameters, both power and thermal related, are accurate, Reactive can enforce the thermal threshold effectively. However if there are uncertainties of parameters, the equilibrium temperature cannot precisely enforce the temperature bound.

8.2.2 Constant Power Variation

This set of simulations is designed to evaluate the performance of RT-MTC when there is constant deviation between the estimated and the real tasks power. In these simulations, we compare RT-MTC to the other baselines when the power ratio of all tasks running on the target multicore processor is 4.0, that is, the real power of the tasks is 4 times that of the estimated power. The value of power ratio is chosen intentionally to show the capability of RT-MTC to counteract heavy disturbances, a major benefit of control-theoretic thermal control. In this simulation, we expect RT-MTC to work resiliently under constant power variation.

Fig. 4 compares the performance of RT-MTC, Reactive, MPC-QUAN, and MPC-PWM when the power ratio is 4. We exclude OPEN from the comparison intentionally because it violates the thermal bound during the experiment. Without thermal management, the processor cannot handle the thermal bound violation, and the steady temperature of the two cores reaches $84°C$; this significantly exceeds the $60°C$ temperature threshold and likely to trigger the internal hardware thermal control.

As shown in the top figure in Fig. 4(a), the temperature under RT-MTC converges to the temperature set point $60°C$. The slight oscillation in converged temperature, which can be seen in Fig. 4(d), is caused by the sampling period. If the temperature surpasses the bound within the sampling period ($10s$ in this experiment) RT-MTC cannot respond to enforce the thermal bound. Meanwhile, we also observe the frequency switches between 3 levels guided by PWM according to RT-MTC's output.

The bottom half of Fig. 4(a) shows the utilization of the multicore processor. As seen in the figure, the utilization is always below the utilization bound, validating that RT-MTC can enforce the real-time utilization bound. Because of RT-MTC saturation component, the frequency never switches to the lowest level, which confines utilization under the real-time bound.

Fig. 4(b) illustrates the simulation results under Reactive. After two frequency switches, Reactive forces the frequency to stay at $1.6GHz$ even though the temperature violates the thermal bound. Recall the algorithm of Reactive: if the thermal bound is hit, the frequency will change to the predefined level to enforce the equilibrium temperature, which, otherwise, is calculated based on the nominal model. In this case, the predefined frequency level is $1.6GHz$. However, in this simulation, the power ratio is 4.0 rather than 1.0 used by Reactive. Hence, at the same frequency, more power is generated and the predefined frequency level in Reactive cannot prohibit the temperature from surpassing the bound. This experiment shows clearly that Reactive is not able to handle thermal management accurately under power uncertainty.

Compared to Reactive, RT-MTC follows the temperature set point more precisely under power uncertainty. When the power generated by the processor is overestimated, the processor runs at higher frequency in RT-MTC than Reactive, so that throughput of the systems is improved. When the power is underestimated, likewise, RT-MTC adjusts the processor frequency to consume less power than Reactive, which can not only save power consumption of the workload but also reduce power consumed by the cooling system. Moreover, in this case, Reactive is more likely to trigger internal thermal throttling.

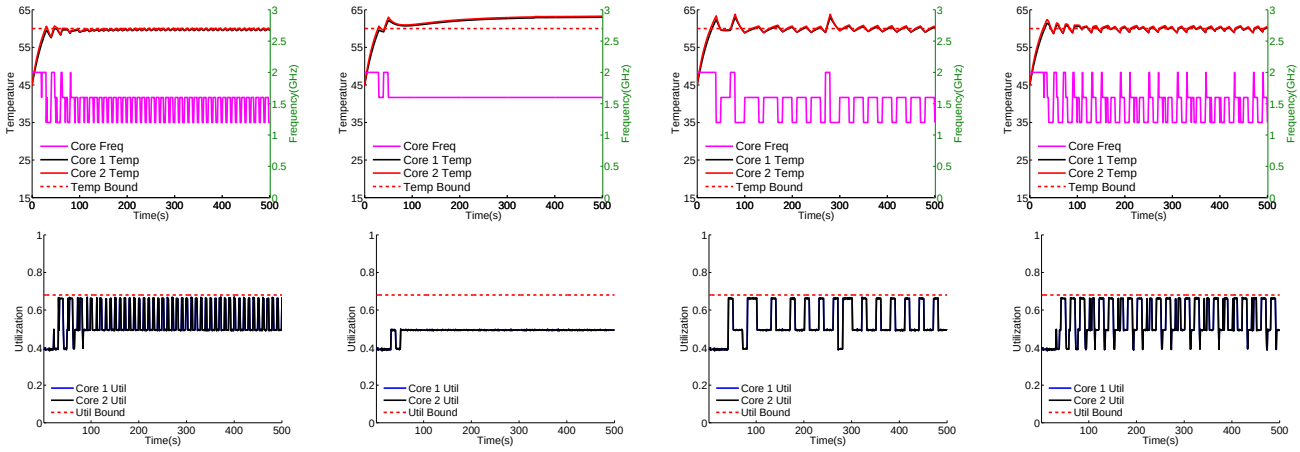
Fig. 4(c) and 4(d) show the simulation results of MPC-QUAN and MPC-PWM. Both baselines can ensure the temperature set point. However, there is oscillation in both cases. For MPC-QUAN, because of the effect of quantization, the temperature frequently violates the bound slightly. Although MPC-PWM can alleviate the effect of quantization by PWM, the sampling period that we analyzed in RT-MTC also induce oscillation around the thermal bound. Moreover, since MPC works on the margin of constraints, it behaves in a complex, nonlinear way. That makes the oscillation of MPC-PWM greater than that of RT-MTC. On the other hand, MPC can handle effectively the real-time constraints embedded in the constrain optimizing problem (8), which then enforces the real-time constraints, that is, the utilization bound.

The major advantage of RT-MTC over MPC-like methods is the reduction of running overhead and implementation complexity. When employing MPC, the controller must solve online the constrained optimization problem, which is notably computation intensive [25]. In contrast, RT-MTC only involves computation of a linear function. Moreover, although there are a few of commercial or open source optimization solver, porting them to solve MPC is still a difficult task.

8.2.3 Dynamic Power Variation

This set of simulations is designed to evaluate the case when the power ratio of tasks deviate from the estimation dynamically. Since tasks often experience different stages of processing, the power of tasks changes frequently. Thus, dynamic power variation is a common source of uncertainty for thermal management. In this simulation, we also assume asymmetric power ratio variation: that is, cores consuming different power when running. For the simulations in this section, we assume the power ratio of Core 1 rises to 4.0 at $200s$ and then decreases to 0.5 at $300s$ while Core 2 keeps the power unchanged.

Similarly to the case of constant power variation, OPEN violates the thermal bound under dynamic power variation.



(a) RT-MTC (b) Reactive (c) MPC-QUAN (d) MPC-PWM

Figure 4: Constant Power Variation (Power Ratio = 4)

However, since only the power of core 1 increases, the temperature of both cores rises less than if the power of both cores varied.

Fig. 5 shows the simulation results of different algorithms under dynamic power variation. Fig 5(a) shows that the temperature of core 1 is below the temperature bound under RT-MTC, validating that RT-MTC is able to ensure the thermal bound under dynamic power variation. We observe that RT-MTC responds to the abrupt temperature increase from 200s to 300s. So when power decreases, the temperature is still able to stay near the temperature bound.

Unlike the previous experiments, Reactive has no steady temperature error in the simulation, as shown in Fig. 5(b). As only one core's power rises, the heat generated by the processor is less than that when both cores' power rise; hence the predefined frequency level can enforce the thermal bound. However, we observes spikes in temperature which violates the thermal bound. These spikes occur because the reactive mechanism only responds to thermal violation passively, compared to RT-MTC where the feedback controller is designed intentionally to accommodate a temperature variation so as to offset thermal violation.

Fig. 5(c) and 5(d) show the results under MPC-QUAN and MPC-PWM, respectively. When subjected to dynamic power variation, both MPC baselines can keep the temperature around the thermal bound. But similarly to the case of constant power variation, quantization and nonlinear control behavior cause oscillation.

To explore the limits of robustness of RT-MTC, we also perform additional simulation experiments under wider uncertainty than the two simulations discussed here. The results also indicate that RT-MTC is more robust than other algorithms when subjected to uncertainties. More details on these experiments may be found in [13].

9. CONCLUSION

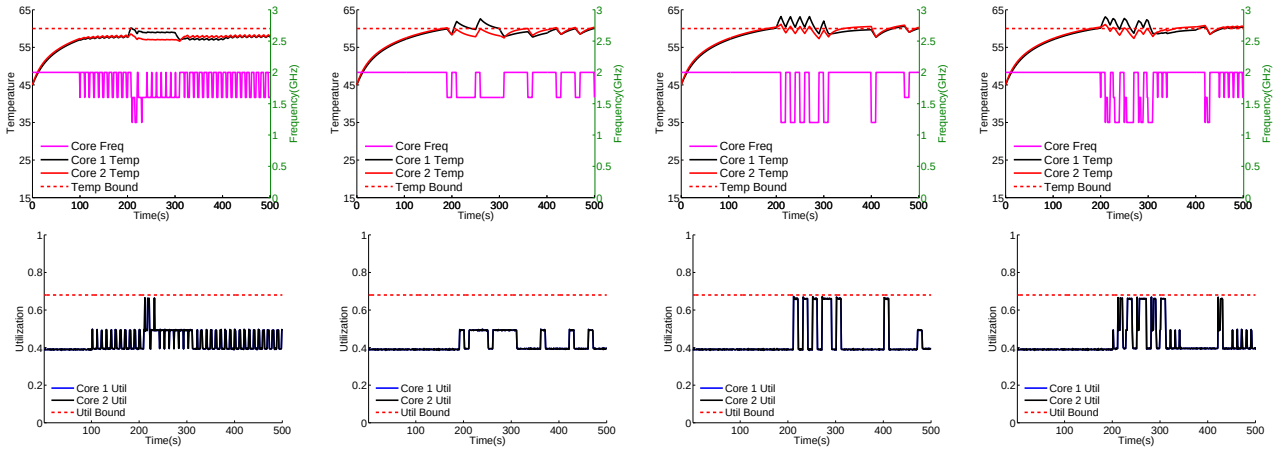
Embedded real-time systems face significant challenges in thermal management with their adoption of multicore processors of increasing power density. Such systems require the temperatures and real-time performance of *multiple* cores to be controlled simultaneously, leading to multi-input-multi-output control problems with inter-core thermal coupling. This paper presents *Real-Time Multicore Thermal Control (RT-MTC)*, the first feedback thermal control algorithm specifically designed for multicore embedded real-time systems. RT-MTC dynamically enforces both the temperature and the CPU utilization bounds of a multicore processor through DVFS. The strength of RT-MTC lies in both its control-theoretic approach and its practical design. RT-MTC employs a highly efficient controller that integrates saturation and proportional control components rigorously designed to enforce the desired core temperature and CPU utilization bounds. Moreover, It handles discrete frequencies through Pulse Width Modulation (PWM) that enables RT-MTC to achieve effective thermal control with only a small number of frequencies typical in current processors. The robustness and advantages of RT-MTC over existing thermal control approaches are demonstrated through extensive simulations under a wide range of power consumptions.

10. ACKNOWLEDGMENTS

The research of Yong Fu and Chenyang Lu is supported in part by NSF CAREER Award CNS-0448554 and CNS-1035773 (CPS). The research of Nicholas Kottenstette and Xenofon Koutsoukos is supported in part by NSF Award NSF-CCF-0820088.

11. REFERENCES

[1] http://ark.intel.com/Product.aspx?id=27255.
[2] http://www.spec.org/.
[3] www.lm-sensor.org.
[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.
[5] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. In *DATE*, 2008.
[6] J.-J. Chen, C.-M. Hung, and T.-W. Kuo. On the minimization fo the instantaneous temperature for periodic real-time tasks. In *RTAS*, 2007.
[7] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for real-time tasks under thermal constraints. In *RTAS*, 2009.
[8] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. *SIGARCH Comput. Archit. News*, 34(2):78–88, 2006.



(a) RT-MTC (b) Reactive (c) MPC-QUAN (d) MPC-PWM

Figure 5: Dynamic Power Variation

[9] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. Thermal-aware global real-time scheduling on multicore systems. In *RTAS*, 2009.

[10] G. F. Franklin, J. D. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., 1998.

[11] X. Fu, X. Wang, and E. Puster. Dynamic thermal and timeliness guarantees for distributed real-time embedded systems. In *RTCSA*, 2009.

[12] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang. Feedback Thermal Control for Real-time Systems. *RTAS*, 2010.

[13] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang. Feedback Thermal Control of Real-time Systems on Multicore Processors. Technical Report WUCSE-2011-3, Washington University in St. Louis, 2011.

[14] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *WWC*, 2001.

[15] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and freon: temperature emulation and management for server systems. In *ASPLOS*, 2006.

[16] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas. A framework for dynamic energy efficiency and temperature management. In *MICRO*, 2000.

[17] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. Compact thermal modeling for temperature-aware design. In *DAC*, 2004.

[18] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *MICRO*, 2003.

[19] J. S. Lee, K. Skadron, and S. W. Chung. Predictive temperature-aware DVFS. *IEEE Transactions on Computers*, 59(1):127–133, 2010.

[20] W. Liao, L. He, and K. M. Lepak. Temperature and supply voltage aware performance and power modeling at microarchitecture level. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 24(7):1042–1053, 2005.

[21] M. Lindberg and K.-E. Årzén. Feedback control of cyber-physical systems with multi resource dependencies and model uncertainties. In *RTSS*, 2010.

[22] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *JACM*, 20(1):46–61, 1973.

[23] J. Liu. *Real-Time systems*. Prentice Hall, 2000.

[24] Y. Liu, H. Yang, R. P. Dick, H. Wang, and L. Shang. Thermal vs energy optimization for dvfs-enabled processors in embedded systems. In *ISQED*, 2007.

[25] J. M. Maciejowski. *Predictive Control with Constraints*. Pearson Eduction Limited, Edinburg Gate, England, 2002.

[26] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta. Processor speed control with thermal constraints. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(9):1994–2008, 2009.

[27] R. Ortega, A. J. Van Der Schaft, I. Mareels, B. Maschke, and L. G. Y. Supelec. Putting energy back in control. *Control Systems Magazine, IEEE*, 21(2):18–33, 2001.

[28] G. Quan and Y. Zhang. Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks. In *ECRTS*, 2009.

[29] R. Rao, S. B. K. Vrudhula, and C. Chakrabarti. Throughput of multi-core processors under thermal constraints. In *ISLPED*, pages 201–206, 2007.

[30] K. Skadron, T. F. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *HPCA*, 2002.

[31] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *ICS*, 2003.

[32] A. J. van der Schaft. *L2-Gain and Passivity in Nonlinear Control*. New York:Springer-Verlag, 1999.

[33] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. *Real-Time Systems*, 39(1-3):73–95, 2008.

[34] Y. Wang, K. Ma, and X. Wang. Temperature-constrained power control for chip multiprocessors with online model estimation. *SIGARCH Comput. Archit. News*, 37(3):314–324, 2009.

[35] F. Zanini, D. Atienza, L. Benini, and G. De Micheli. Multicore Thermal Management with Model Predictive Control. In *ECCTD 2009*, 2009.

[36] F. Zanini, D. Atienza, and G. De Micheli. A control theory approach for thermal balancing of MPSoC. In *ASP-DAC*, pages 37–42, 2009.

[37] F. Zanini, C. N. Jones, D. Atienza, and G. De Micheli. Multicore thermal management using approximate explicit Model Predictive Control. In *ISCAS*, 2010.