Report Number: WUCSE-2011-3

2011

Feedback Thermal Control of Real-time Systems on Multicore Processors

Yong Fu, Nicholas Kottenstette, Chenyang Lu, and Xenofon D. Koutsoukos

Real-time systems face significant challenges in thermal management with their adoption of modern multicore processors. While earlier research on feedback thermal control has shown promise in dealing with the uncertainties in the thermal characteristics, multicore processors introduce new challenges that cannot be handled by previous solutions designed for single-core processors. 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. Furthermore, current Dynamic Voltage and Frequency Scaling (DVFS) mechanisms only support a finite set of states, leading to discrete control variables that cannot be... **Read complete abstract on page 2.**

Feedback Thermal Control of Real-time Systems on Multicore Processors

Yong Fu, Nicholas Kottenstette, Chenyang Lu, and Xenofon D. Koutsoukos

Complete Abstract:

Real-time systems face significant challenges in thermal management with their adoption of modern multicore processors. While earlier research on feedback thermal control has shown promise in dealing with the uncertainties in the thermal characteristics, multicore processors introduce new challenges that cannot be handled by previous solutions designed for single-core processors. 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. 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), the first feedback thermal control framework specifically designed for multicore real-time systems. RT-MTC dynamically enforces both the temperature and the CPU utilization bounds of a multicore processor through DVFS with discrete frequencies. 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. It handles discrete frequencies through a PulseWidth Modulation (PWM) that achieves effective thermal control by manipulating the dwelling time of discrete frequencies. As a result RT-MTC can achieve effective thermal control with only a small number of frequencies typical in current processors. The robustness and advantages of RTMTC over existing thermal control approaches are demonstrated through extensive simulations under a wide range of uncertainties in term of power consumption.

Feedback Thermal Control of Real-time Systems on Multicore Processors

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

***Abstract*—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 that the temperature and real-time performance of *multiple* cores are controlled simultaneously, leading to multi-input-multi-output (MIMO) 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)*, the first 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 integrates saturation, proportional control components and Pulse Width Modulation (PWM). RT-MTC 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 the Intel Core 2 Duo processor and extensive simulations under a wide range of uncertainties in power consumption.**

I. Introduction

The increasing complexity of real-time applications demands the adoption of multicore microprocessors to leverage their computing power [1], [2]. For these multicore real-time systems, processor overheating must be avoided while maintaining desired real-time performance. Moreover, as the heat dissipated by these real-time systems increases, there is a rise in the operational cost of cooling systems and accompanying environmental impacts (e.g. emission of greenhouse gases). Thus, balancing real-time performance and thermal management requirement can lead to a sustainable real-time computing model.

However, the need to enforce temperature bounds may conflict with the need to meet real-time performance requirements, because typical thermal management mechanisms such as Dynamic Voltage and Frequency Scaling (DVFS) reduce processor speed, resulting in prolonged response 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. In particular, the power consumption of a processor may vary significantly when running different applications, due to the different sets of instructions executed [3], [4].

In recent years, control-theoretic approaches have shown promise in feedback thermal control [5]–[12] to effectively deal with 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 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 both a desired temperature and the CPU utilization bounds of the real-time systems through DVFS. RT-MTC employs an efficient and robust control design that integrates the following 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 temperature by dynamically switching between discrete voltage/frequency levels.

RT-MTC combines both 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 that 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 under a wide variation in power consumption.

II. Related Works

There has been significant work on thermal aware real-time scheduling for both single-core processors [13], [14] and multicore processors [15]–[17]. Those algorithms rely on accurate models about the thermal characteristics of the processors, and hence cannot effectively deal with significant 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 can be implemented in user space without modifications to the kernel (as presented in Section VII). Our solution can therefore be easily deployed in existing systems.

Control-theoretic thermal management has been explored for non-real-time systems. As a comparative study of different control strategies and mechanisms for thermal management of multicore processors, [10] presents a general framework of dynamic thermal management for multicore processors. Essentially, the proposed framework is a hierarchical feedback control loop with PI controllers. However, the authors do not provide real-time performance guarantees. Several papers [6]–[8], [18]–[20] have adopted model predictive control or online convex optimization for dynamic thermal management. None of these works are 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 [7], [8], [18].

Control-theoretic approaches have recently been proposed for thermal management of real-time systems [5], [9]. [5] 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. [9] proposed a feedback control framework to manage both temperature and media performance Both [9] and [5] 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.

III. Problem Formulation

We assume a common real-time system model where the workload consists of real-time tasks released periodically (e.g., hybrid testing [1], video, avionics mission computing and process control). A real-time system comprises a set of periodic real-time tasks running on a multicore processor

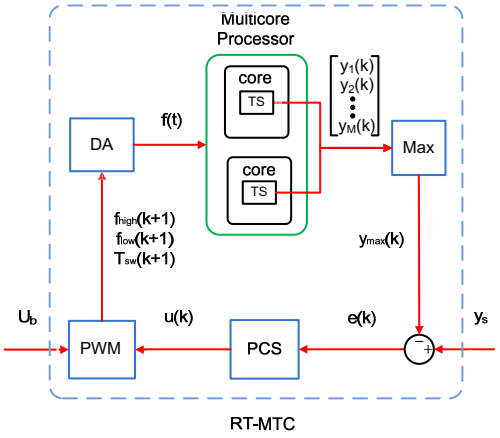


Fig. 1: Feedback Control Loop of RT-MTC

with $M \in \mathbb{N}$ homogeneous cores. The processor supports Dynamic Voltage and Frequency Scaling (DVFS). We assume two common characteristics of DVFS in mainstream multicore processors. 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.

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 the 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 bounded to its period, 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 [21]. 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 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. The temperature set point y_s is the desired temperature below the maximum temperature tolerable by the processor. Our control problem formulation therefore aims to meet both the thermal and real-time performance requirements of a real-time system.

IV. 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

[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 [22] by enforcing appropriate schedulable utilization bounds.

Proportional Controller with Saturation (PCS), a Pulse Width Modulation (PWM), and a DVFS Actuator (DA). The Max function calculates the maximum temperature among all cores, as measured by TS. The user input to RT-MTC is the desired temperature set point y_s. 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, $y_i(k)$, 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 in Section VI-A.
3) The PWM receives the controller output $u(k)$ and calculates a pair of frequencies $f_{high}(k+1), f_{low}(k+1)$ as well as 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 V-B.
4) The DA adjusts the frequency of the multicore processor via the DVFS interface according to the $(f_{high}(k+1), f_{low}(k+1), T_{sw}(k+1))$ output 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 VII.

V. Thermal Dynamic Model

As a foundation for 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 use a well-known model of the power consumption as a function of the frequency. 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 and approximation 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 demostrated in our evaluation (Sec. VIII-A2).

A. Power Model

As shown in [23], 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 accurately by a piecewise linear model $P_{\text{idle}} = (C_0(V(k)) + C_1(V(k))y(k))V(k)$ [24]. 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 [25].

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, $V(k) = Kf(k) + V_{\text{th}}$ [26], .

B. Pulse Width Modulation (PWM)

As each core of the multicore processor only runs under a discrete set of frequencies, the power $\bar{P}_a(k)$ in equation (2) can only switch between discrete levels given a utilization $U(k)$. To track the temperature set point closely, PWM is employed mapping desired average power in each sampling period to the discrete power level 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 $(f_{high}(k+1), f_{low}(k+1), T_{sw}(k+1))$ from $u(k)$. The upper limit of the output corresponds to the maximum frequency supported by hardware of the processor. The lower limit of the output corresponds to the lowest frequency which does not violate the real-time schedulable utilization bound or the processor's minimum frequency, if the processor's frequency cannot be set that low. 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 $\bar{P}_a$ at $f_{\max}, f_{\min}$ respectively. We can rewrite the power model to incorporate PWM based on (2) as

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

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 power uncertainty induced by the lower and upper bound average power. Note that the uncertainty caused by power variation is also represented by G_p.

The power consumption model (3) captures power behavior of the processor approximately, since it derives the average

power rather than actual power, and the limit on $\bar{P}_a$ is employed as parameters. However, as we shown in our stability analysis (Section VI-A) and experiments (Section VIII-A2), inherent robustness of RT-MTC feedback control design can tolerate some degree of modeling error without compromising system stability.

C. 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 [17]. Compared to architecture-level thermal models such as Hotspot [27], the model presented here is simpler but more suitable for control design of thermal management. The effectiveness of the model has been validated in [17], [25].

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 I: Symbols in Thermal Dynamic Model

Based on the symbols listed in Tab. I, 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 $\mathbf{Y}(t) = [y_1(t), \ldots, y_M(t), y_h(t)]^\mathsf{T} \in \mathbb{R}^{M+1}$, $\mathbf{P}(t) = [P_1(t), \ldots, P_M(t)]^\mathsf{T} \in \mathbb{R}^M$ 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 [28] 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 $\bar{\mathbf{P}}(k) = [\bar{P}_1(k), \ldots, \bar{P}_M(k)]^\mathsf{T} \in \mathbb{R}^M$. Substituting the power model (3) for $\bar{P}(k)$ in (5) results in:

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

in which $\Phi = \left(\Phi_o + C_y\Psi_P\begin{bmatrix} I_M & 0 \end{bmatrix}\right)$ where $\begin{bmatrix} I_M & 0 \end{bmatrix} \in \mathbb{R}^{M\times(M+1)}$, $\mathbf{u}(k) = [u_1(k), \ldots, u_M(k)]^\mathsf{T} \in \mathbb{R}^M$, 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 There are two methods to acquire the parameters of the compact thermal model. We can extract the parameters from a fine grain thermal RC model, such as Hotspot. Alternatively the parameters can be estimated using realistic operational data, which is also the method we used in this paper. The detailed description of model identification is presented in Section VIII-A1.

VI. 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 and the utilization constraints. As shown in Equation (1), the PCS is a proportional controller with saturation. Saturation is necessary to take into account the minimum and maximum frequency supported by the processor.

A. Control Design

The PCS is designed based on passivity [29] 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 [30]. In this case, passivity offers advantages for computing a Lyapunov function that is used to prove stability of the closed-loop system.

B. Stability

Before the stability analysis, the architecture of the closed-loop system from control theory perspective is shown in Fig. 2.

For simplicity of discussion we will first consider the internal-stability of the system depicted in Fig. 2 in which $y_0(k) = 0$, $P_{am}\Phi_P G_p = 0$ and $y_b = 0$. These terms will later be considered as external-disturbances and corresponding

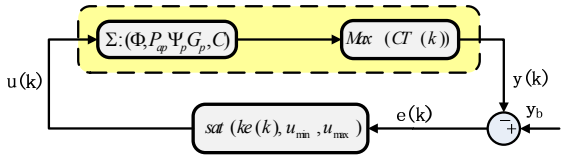


Fig. 2: Passivity based control architecture.

reference in order to determine the steady-state operating ranges of the system.

The following theorem provides the sufficient conditions for the system depicted in Fig. 2 to be asymptotically stable.

Theorem 1: Assume that the system $\Sigma : \{\Phi, P_{ap}\Psi_P G_p, C\}$ depicted in Fig. 2 is unperturbed such that:

$$y(k+1) = \Phi y(k) + P_{ap}\Psi_P G_p u(k)$$
$$y_m(k) = \max\{Cy(k)\}.$$

where $Cy(k) = [y_1(k), \ldots, y_M(k)]^{\mathsf{T}}$ is the output vector for each l^{th}-core temperature. If there i) exists a symmetric positive definite matrix P ($P = P^{\mathsf{T}} > 0$) such that $V(k) = y^{\mathsf{T}}(k)Py(k) > 0$, $\Delta V(k) = V(k+1) - V(k) \leq y_m(k)u(k) - \delta u^2(k)$ ($\delta < 0$); and ii) $k < \frac{-1}{\delta}$; then the closed-loop system is globally asymptotically stable.

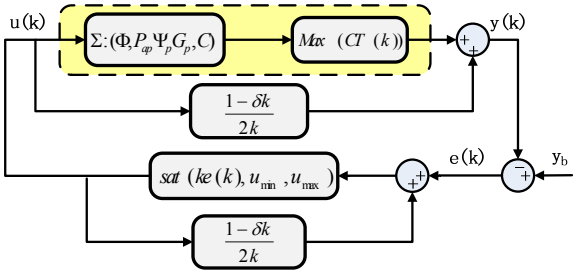


Fig. 3: Equivalent control architecture (no disturbance).

Proof: We use the following equivalent loop-transformation depicted in Fig. 3 so that substituting $y_m(k) = \bar{y}(k) + \frac{\delta k - 1}{2k}u(k)$ into $\Delta V(k) \leq y_m(k)u(k) - \delta u^2(k)$ results in the strictly-input passive relationship

$$\Delta V(k) \leq \bar{y}(k)u(k) - \epsilon u^2(k) \tag{7}$$

in which $\epsilon = \frac{1+\delta k}{2k} > 0$.

Next, we observe that the saturation function is also strictly-output passive since

$$u(k)e(k) \geq \frac{1}{k}u^2(k).$$

Observing that the closed-loop is well posed since $k|\delta| < 1$ and $T_b = 0$, we substitute $e(k) = \left(-\bar{y}(k) + \frac{1-\delta k}{2k}u(k)\right)$ which results in the following relationship:

$$-u(k)\bar{y}(k) \geq \epsilon u^2(k)$$
$$\bar{y}(k)u(k) - \epsilon u^2(k) \leq -2\epsilon u^2(k). \tag{8}$$

Replacing the right-hand side (RHS) of (8) with the RHS of (9) results in the following expression:

$$\Delta V(k) \leq -2\epsilon u^2(k) \tag{9}$$

Since Σ is an asymptotically stable system with finite-gain then there exists a $P_s = P_s^{\mathsf{T}} > 0$ in which $V_s(k) = y^{\mathsf{T}}(k)P_s y(k) > 0$, and $\gamma > 0$ s.t.

$$\Delta V_s(k) \leq \gamma^2 u^2(k) - \bar{y}^2(k) \tag{10}$$

Therefore we can choose a Lyapunov function $V_p(k) = V(k) + \frac{2\epsilon}{\gamma^2}V_s(k) > 0$. Next, we multiply both sides of (11) by $\frac{2\epsilon}{\gamma^2}$ and add the respective sides to those in (10) such that

$$\Delta V_p(k) \leq -\frac{2\epsilon}{\gamma^2}\bar{y}^2(k). \tag{11}$$

Therefore, $V_p(k)$ is a valid *Lyapunov function* [32] for the combined feedback system in which $\Delta V_p(k) < 0$ for all $y_l(k) \neq 0$ $l \in \{1, \ldots, M\}$ and it is obvious from our ZOH equivalent RC-Thermal model that the largest invariant set $\mathcal{M}$ in which $\Delta V_p(k) = 0$ and $(I - \Phi)[0, \ldots, 0, y_{M+1}]^{\mathsf{T}}(k) = 0$ ($\Phi(M+1, M+1) \neq 1$) is when $T_{M+1} = 0$ ($\mathcal{M} = \{0\}$) therefore satisfying the conditions in Theorem 4 (see Appendix A) for the closed-loop system to be globally asymptotically stable. ■

All that remains is to determine a linear-matrix inequality test to compute a $P = P^{\mathsf{T}} > 0$ s.t. $\Delta V(k) = V(k+1) - V(k) \leq y_m(k)u(k) - \delta u^2(k)$ ($\delta < 0$).

Lemma 1: Denote $V(k) = y^{\mathsf{T}}(k)Py(k)$. If there exists a matrix $P = P^{\mathsf{T}} > 0$ and $\delta < 0$ s.t.

$$\Delta V(k) \leq C_l y(k)u(k) - \delta u^2(k), \ \forall\ l \in \{1, \ldots, M\} \tag{12}$$

then $\Delta V(k) = V(k+1) - V(k) \leq y_m(k)u(k) - \delta u^2(k)$.

Proof: Since $y_m(k) = \max\{Cy(k)\} = \max\{C_1 y(k), ..., C_M y(k)\}$ it is obvious that if (13) holds $\forall\ l \in \{1, \ldots, M\}$ then $\Delta V(k) = V(k+1) - V(k) \leq y_m(k)u(k) - \delta u^2(k)$. ■

If we define the output $y_{pl}(k) = y_l(k) - \delta u(k)$ then we can solve for δ by solving the following linear-matrix inequalities which determine if each sub-system with input $u(k)$ and output $y_{pl}(k)$ is passive.

Theorem 2: If there exists a matrix $P = P^{\mathsf{T}} > 0$ and $-\infty < \delta < 0$ s.t. the following LMI's are satisfied:

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

for all $l \in \{1, \ldots, M\}$ then $\Delta V(k) = V(k+1) - V(k) \leq y_m(k)u(k) - \delta u^2(k)$.

Proof: The above LMI test is a well known necessary and sufficient test [33, Corollary 2] to solve for (13) in Lemma 1 in terms of each passive output $y_{pl}(k)$ $l \in \{1, \ldots, M\}$. ■

The value for δ can be maximized w.r.t. the above LMI test using the MATLAB function `mincx()` to minimizes $-\delta$ [2] [34].

[2] mincx() is part of the Robust Control Toolbox.

With asymptotic stability established, we can now determine the lowest possible temperature, y_{low} the processor cores and heat-sink can achieve when $u(k) = \frac{-1}{2}$. In particular by solving (6) for the case when $y(k+1) = y(k) = y_{\text{low}}$ when $u(k) = \frac{-1}{2}$ and the environmental temperature is fixed at $y_0(k) = y_0$ we have

$$y_{\text{low}} = U_{\max} \left(I - \Phi\right)^{-1} \left[P_a(f_{\text{low}})\Psi_P G_p + \Psi_y y_0\right].$$

As with a proportional model-predictive control approach some steady-state tracking error $E = (y_b - C_{l\max})$ (in which $l\max \in \{1, \ldots, M\}$ denotes the appropriate index relating to the hottest core temperature) will result due to a non-zero environmental temperature y_0 and non-zero P_{am} resulting from our pulse-width modulation approach. Denoting $\bar{\Psi}_{ap} = kP_{ap}\Psi_P G_p$ we have the following relationship to determine E

$$\begin{aligned} E = & y_b - C_{l\max}\left(I - \Phi + \bar{\Psi}_{ap} C_{l\max}\right)^{-1} \\ & \left[\bar{\Psi}_{ap} y_b + \Psi_y y_0 + P_{am}\Psi_P G_p\right]. \end{aligned}$$

This steady-state error can be mitigated for the nominal case by choosing to let $y_b = (\bar{y}_{\max} + y_{\text{ff}})$ and substituting in the above equation for $E = (\bar{y}_{\max} + y_{\text{ff}} - C_{l\max} y)$ and then solving for y_{ff} such that $(\bar{y}_{\max} - C_{l\max} y) = 0$. For simplicity, we will assume $C_{l\max} = C_1$, denote $R = \left(I - \Phi + \bar{\Psi}_{ap} C_1\right)$ and compute y_{ff} as follows:

$$y_{\text{ff}} = \frac{(1 - C_1 R^{-1}\bar{\Psi}_{ap})\bar{y}_{\max} - C_1 R^{-1}\left[\Psi_y y_0 + P_{am}\Psi_P G_p\right]}{C_1 R^{-1}\bar{\Psi}_{ap}}.$$

VII. 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 may be read using interface provided by the operating system or a third-party library. For example, in Linux, the temperature of cores can be read from the interface provided by *lmsensor* [35] via the coretemp driver (/sys/bus/platform/drivers/coretemp/). The thermal information can also be acquired from standard ACPI interfaces [36]. For those multicore processors without thermal sensors on each core, such as those used in embedded systems, soft thermal sensors [37] 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. VI and Sec. V-B.

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} using some interfaces to access the processor's DVFS function (ACPI, lmsensor, or Machine Specific Register). The delay from PWM output switching time T_{sw} to the time that the frequency is actually changed 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. We choose the period because $10s$ is short enough to control the thermal behavior of the processor (its time constant is greater than $100s$) without imposing undue overhead from frequency switching and computation.

VIII. Evaluation

We first evaluate RT-MTC through experiments based on the above implementation, using a laptop equipped with an Intel Core 2 Duo processor. We then perform extensive simulations based on parameters acquired through model identification experiments on the laptop. The simulations complement experiments results by allowing us to examine RT-MTC's performance under stress-test conditions (such as fan failure) which are difficult to run on real hardware.

A. Experiments

The hardware platform used for the experiments is a Lenovo W500 laptop equipped with an Intel T9400 Core 2 Duo processor, Fedora Linux 12, and the Linux kernel 2.6.32 distributed with Fedora 12. The T9400 processor has 2 digital thermal sensors located on each core and supports processor-wide DVFS: that is, the two cores' frequencies must be set to the same level. The DVFS frequencies and the thermal properties of the T9400 are listed in Table II.

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

TABLE II: Frequencies and Thermal Properties of the T9400 Processor

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. Matlab Model Identification Toolbox is then employed to estimate the parameters. The real-time workloads used for model identification are based on Mibench [38], a test suite for embedded systems which includes common automotive, consumer electrics, security, and office applications, and SPEC CPU 2006 [39],

a standard benchmarks suite widely used in industry and academia. We choose two micro benchmarks, CRC and Bzip2, from Mibench and SPEC CPU 2006 respectively. The CRC is a data verification application and the Bzip2 is a data compression tool. Then we implement three kinds of workloads: CRC alone, Bzip2 alone and a Mixed workload containing both benchmarks. Mixed workload includes both benchmarks. Each workload involves 10 periodic tasks, equally distributed between the two cores. The period and execution time of the tasks are listed in Table III.

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

TABLE III: Workload Tasks Period and Execution Time@$2.53GHz$ (ms)

Thermal Parameters (Mixed, Fit*: 82%)					
Parameters	Value	Parameters	Value	Parameters	Value
$R_1(\Omega)$	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%)					
Parameters	Value	Parameters	Value	Parameters	Value
$R_1(\Omega)$	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%)					
Parameters	Value	Parameters	Value	Parameters	Value
$R_1(\Omega)$	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				

*: Fit is the accuracy index in Matlab Model Identification Toolbox.

TABLE IV: Results of Model Identification

To stimulate the thermal behavior in different frequency patterns, 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 IV shows the results of the model identification via Matlab Model Identification Toolbox. Fig. 4 illustrates the temperature and frequency of the Mixed workload; the other two workloads are omitted here due to space constraints.

Two important observations can be drawn from Table IV. 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 indicates the importance for thermal control to be robust against uncertainties in model parameters caused by different workloads, as it is unrealistic to expect users to re-profile the system through 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.

2) Experiment Results: In this section we show the results of experiments on the hardware platform. We run RT-MTC under the CRC and the Mixed workloads for 10 minutes each. The controller parameters of RT-MTC are computed using the thermal RC model parameters of the Mixed workload.

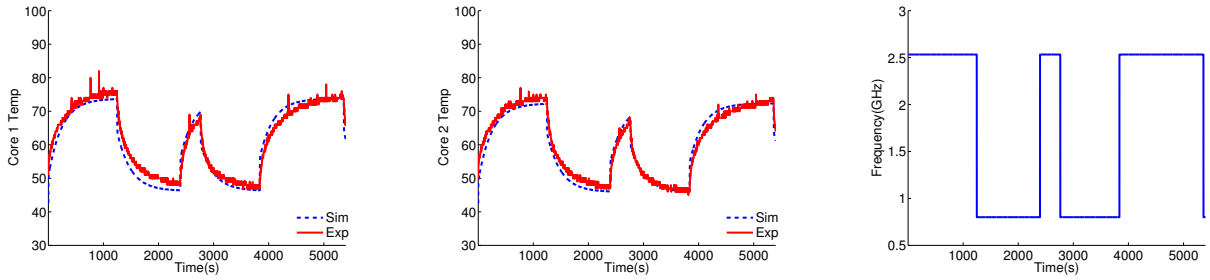
Two important observations may be made from the results, plotted in Fig. 5. First, RT-MTC can maintain the temperature set point while enforcing the utilization bound. As seen in Fig. 5(b), after $280s$ the temperature is steady at the temperature set point, $60°C$. The average upper limit of the utilization 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 IV, 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.

B. Simulation

We perform extensive simulations based on the model parameters identified through the experiments presented in the last subsection. 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.

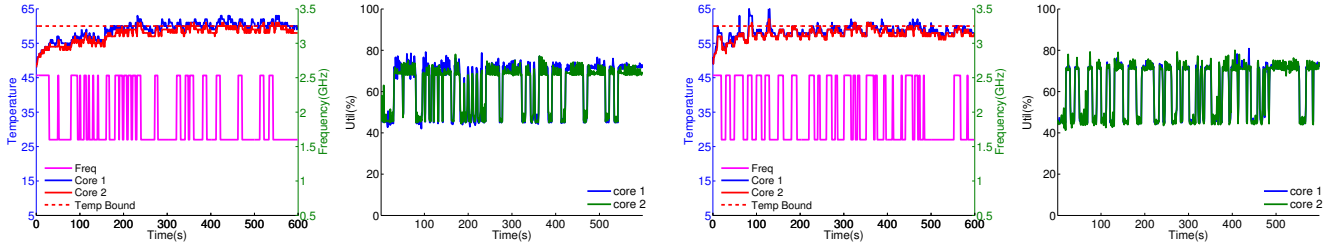
1) Simulation Setup: There are two components in our simulation environment: an event driven simulator implemented in C++ and a Simulink module implemented in MATLAB (R2008a). The C++ simulator simulates real-time systems over multicore processors and calculates the processor utilization according to the frequency output by the controller. The Simulink module performs the controller's computation. 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 [40]. The power and thermal related parameters of T7200 are shown in Table V. The parameters of the leakage power model are acquired by linear approximation of an accurate leakage power model [41]. The active power and available frequencies are obtained from Intel T7200 data sheet [40]. 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.



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

Fig. 4: Model Identification Data (Mixed Workload)



(a) Mixed Workload (b) CRC

Fig. 5: Experimental Results of RT-MTC

We use the same methodology and tools for model identification as described in Sec. VIII-A1. The acquired Thermal parameters are listed in Table V. As thermal design is different between manufacturers, it is reasonable that these parameters identified vary significantly from those identified for the T9400.

Power Parameters					
Parameters	Value				
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					
Parameters	Value	Parameters	Value	Parameters	Value
$R_1(\Omega)$	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 V: Simulation Parameters

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 real-time systems. The Rate Monotonic (RM) scheduling algorithm [21] 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 $[100ms, 200ms]$. Based on the tasks' period, the execution time of each task is chosen to generate nearly equal utilization for each task while keeping the sum of all tasks' utilization at 0.7, just below the utilization bound.

In the following simulations, we set the temperature bound to $60°C$, slightly 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 [7]. The control algorithms of both baselines are the solutions of the following constraint optimizing problem with the optimizing objective as follows:

$$J(k) = \sum_{i=1}^{H_p} |y_{max}(k+i) - y_s|^2 \tag{13}$$

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 (14) 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 real-time systems [13]. 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 is uncertainty of parameters, the equilibrium temperature cannot precisely enforce the temperature bound.

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. 6 compares the performance of RT-MTC, Reactive, MPC-QUAN, and MPC-PWM when the power ratio is 4. We exclude OPEN from the comparison 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.

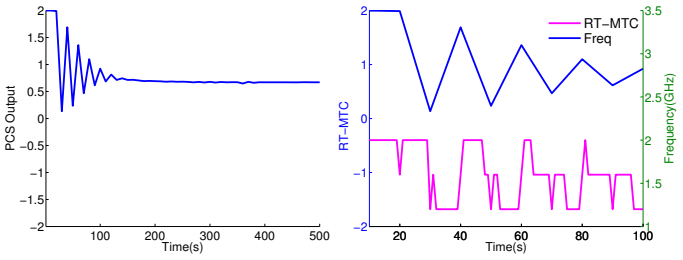
As shown in the top figure in Fig. 6(a), the temperature under RT-MTC converges to the temperature set-point $60°C$. The slight oscillation in converged temperature, which can be seen in Fig. 6(d), is caused by the sampling period. If the temperature surpasses the bound within the sampling period ($10s$ 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. 6(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. 6(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.

To closely examine the mechanism of PWM, Fig. 7 provides a zoom view of mapping between RT-MTC output and the frequency. From the zoom view, we can see the higher the controller output is, the longer the frequency stays on high level. On the other hand, when the RT-MTC output changes, the duty cycle of the frequency changes accordingly. Fig. 7 intuitively verifies the mechanism of PWM presented in Sec. V-B.



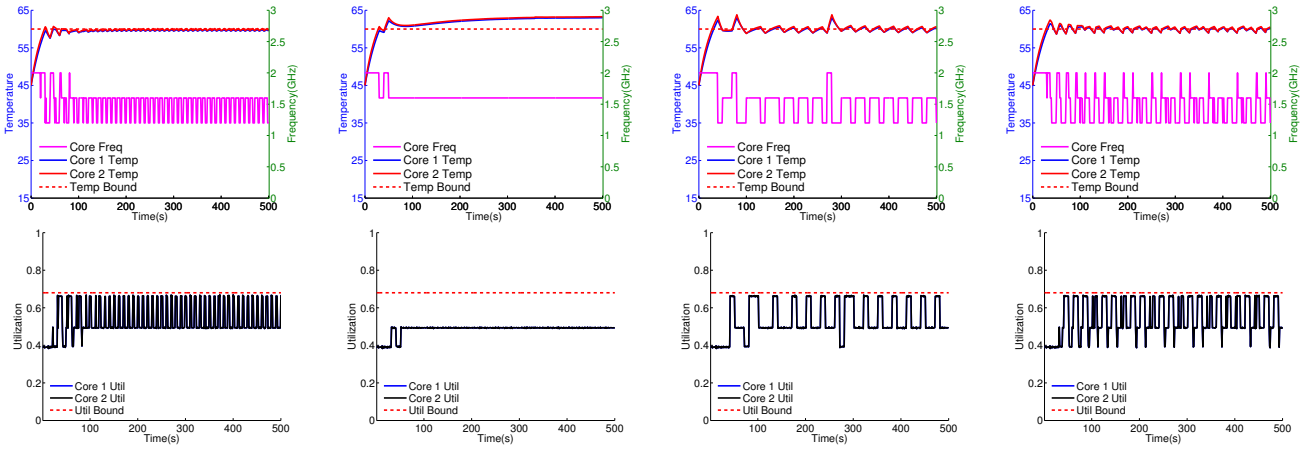
(a) RT-MTC Output

(b) Mapping between RT-MTC Output and Frequency (Zoom View)

Fig. 7: Mapping between RT-MTC Output and Frequency

Fig. 6(c) and 6(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 the real-time constraints embedded in the constrain optimizing problem (14), 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



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

Fig. 6: Constant Power Variation (Power Ratio = 4)

complexity. When employing MPC, the controller must solve online the constrained optimization problem, which is notably computation intensive [42]. 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.

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. 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. 8 shows the simulation results of different algorithms under dynamic power variation. Fig 8(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. 8(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.

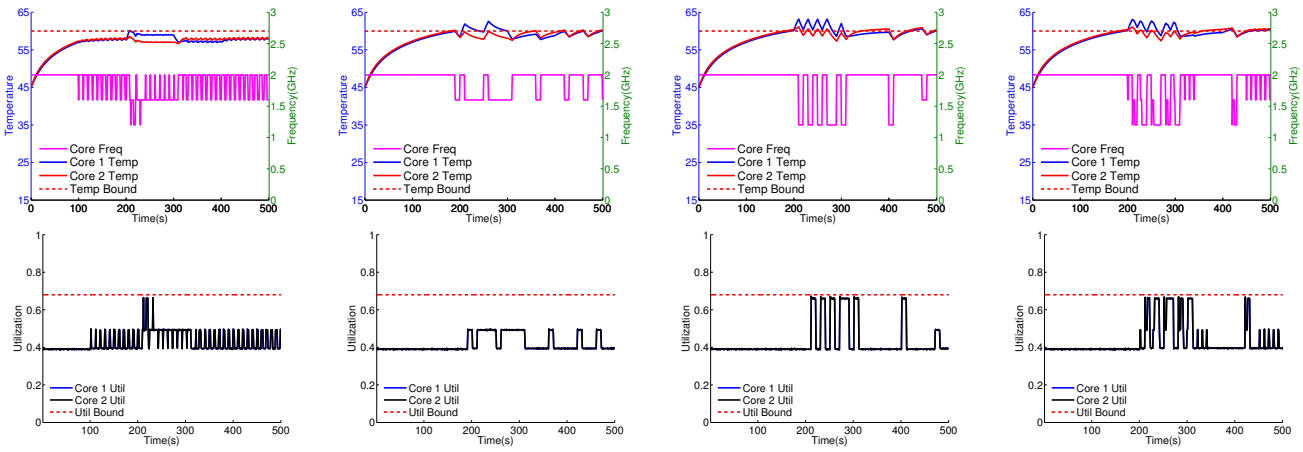
Fig. 8(c) and 8(d) show the results under MPC-QUAN and MPC-PWM, respectively. When subjected to dynamic power variation, both MPC baselines can keep the temperature around the thermal bound. But similarly to the case of constant power variation, quantization and nonlinear control behavior cause oscillation.

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 power ratio uncertainty. More details on these experiments may be found in [31].

C. Robustness

To evaluated the robustness of RT-MTC against power ratio variation, we perform a stress test on RT-MTC over a range of power ratios, specifically, $[0.5, 6]$ with interval 0.5. we ignore the cases when power ratio is greater than 6 since it is impossible in practice. In this simulation we still assume asymmetric and constant power variation. We run the simulation over $1000s$. The power ratio of core 1 is set as 4.0 while that of core 2 as 1.0. Only the temperature of core 1 in last $500s$ is recorded since the temperature of core 2 can not violated the threshold in these cases.

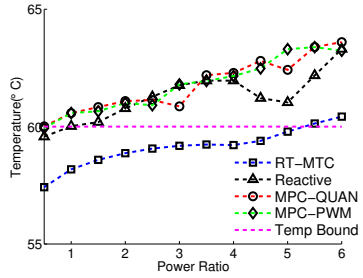
As seen in Fig. 9(a) and 9(b), the maximum and average temperature over varied power ratio of different algorithms are compared. In Fig. 9(a), behavior of RT-MTC is distinct to other algorithms. We observer that, under most of power variation cases (except power ratio 5.5 and 6), the maximum temperature of RT-MTC is below the temperature bound. That means RT-MTC can enforce thermal constraints effectively. In contrast, Reactive, MPC-QUAN and MPC-PWM can only ensure thermal bound when power ratio is less than 1. Beyond



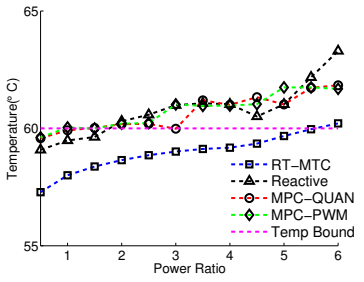
(a) RT-MTC (b) Reactive (c) MPC-QUAN (d) MPC-PWM

Fig. 8: Dynamic Power Variation

that power ratio, the error between maximum temperature and thermal bound increase. A useful practice of thermal management is not only to enforce the thermal constraint but also to maintain throughput of the processor. We notice that in Fig. 9(a) when power ratio is between $[3, 5]$, RT-MTC can maintain the temperature near the bound which implies the processor works under the maximum speed which is allowed thermally. Fig. 9(a) hints that for RT-MTC parameters tuning we can estimate the power of workload conservatively so as to maintain better trade-off between thermal constraints and processing capability under real workload. The the results of average temperature, shown in Fig. 9(b), can apply the same analysis of maximum temperature case.



(a) Maximum Temperature



(b) Average Temperature

Fig. 9: Robustness of RT-MTC Against Power Ratio

IX. Conclusion

The increase of complexity of modern real-time applications has accelerated their adoption of multicore processors. Sustainable real-time computing requires effective thermal control to reduce cooling cost without violating the real-time performance requirement. Multicore real-time 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 real-time systems. RT-MTC dynamically enforces both a desired temperature set point and the schedulable utilization bounds of a multicore processor through DVFS. The strength of RT-MTC lies in both in its control-theoretic approach and in 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 the small number of frequencies typical of current processors.

A direction of future work is to extend our control-theoretic approach to provide thermal and load control for performance-sensitive systems with more sophisticated workload models as well as distributed systems settings.

References

[1] H.-M. Huang, T. Tidwell, C. Gill, C. Lu, X. Gao, and S. Dyke, "Cyber-physical systems for real-time hybrid structural testing: a case study," in *ICCPS*, 2010.

[2] C. Kenna, J. Herman, B. Brandenburg, A. Mills, and J. Anderson, "Soft real-time on multiprocessors: Are analysis-based schedulers really worth it?" in *RTSS*, 2011.

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

[4] ——, "Identifying program power phase behavior using power vectors," in *Proceedings of IEEE International Workshop on Workload Characterization*, 2003.

[5] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang, "Feedback Thermal Control for Real-time Systems," *RTAS*, 2010.
[6] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 314–324, 2009.
[7] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore Thermal Management with Model Predictive Control," in *ECCTD 2009*, 2009.
[8] F. Zanini, C. N. Jones, D. Atienza, and G. De Micheli, "Multicore thermal management using approximate explicit Model Predictive Control," in *ISCAS 2010*, 2010.
[9] M. Lindberg and K.-E. Årzén, "Feedback control of cyber-physical systems with multi resource dependencies and model uncertainties," in *RTSS*, 2010.
[10] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," *SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 78–88, 2006.
[11] J. Srinivasan and S. V. Adve, "Predictive dynamic thermal management for multimedia applications," in *ICS*, 2003.
[12] X. Fu, X. Wang, and E. Puster, "Dynamic thermal and timeliness guarantees for distributed real-time embedded systems," in *RTCSA*, 2009.
[13] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," *Real-Time Systems*, vol. 39, no. 1-3, pp. 73–95, 2008.
[14] J.-J. Chen, , S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints," in *RTAS*, 2009.
[15] J.-J. Chen, C.-M. Hung, and T.-W. Kuo, "On the minimization fo the instantaneous temperature for periodic real-time tasks," in *RTAS*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 236–248.
[16] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," in *DATE*, 2008.
[17] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *RTAS*, april 2009, pp. 131 –140.
[18] 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*, vol. 56, no. 9, pp. 1994–2008, 2009.
[19] F. Zanini, D. Atienza, and G. De Micheli, "A control theory approach for thermal balancing of mpsoc," in *ASP-DAC*, 2009, pp. 37–42.
[20] Y. Fu, C. Lu, and H. Wang, "Control-theoretic thermal balancing for clusters," in *IPDPS*, 2010.
[21] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *JACM*, vol. 20, no. 1, pp. 46–61, 1973.
[22] J. Liu, *Real-Time systems*. Prentice Hall, 2000.
[23] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. Koutsoukos, and H. Wang, "Feedback thermal control for real-time systems," Department of Computer Science and Engineering, Wahsington University in St. Louis, Tech. Rep. 2009-17, 2009.
[24] G. Quan and Y. Zhang, "Leakage aware feasibility analysis for temperature-constrained hard real-time periodic tasks," in *ECRTS*, 2009.
[25] R. Rao, S. B. K. Vrudhula, and C. Chakrabarti, "Throughput of multicore processors under thermal constraints," in *ISLPED*, 2007, pp. 201–206.
[26] W. Liao, L. He, and K. M. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1042–1053, 2005.
[27] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam, "Compact thermal modeling for temperature-aware design," in *DAC*, 2004.
[28] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison Wesley Longman, Inc., 1998.
[29] 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*, vol. 21, no. 2, pp. 18–33, 2001.
[30] A. J. van der Schaft, *L2-Gain and Passivity in Nonlinear Control*. New York:Springer-Verlag, 1999.
[31] Y. Fu, N. Kottenstette, C. Lu, and X. Koutsoukos, "Feedback thermal control of real-time systems on multicore processors," Department of Computer Science and Engineering, Wahsington University in St. Louis, Tech. Rep. WUCSE-2011-3, 2011. [Online]. Available: http://cse.wustl.edu/Research/Lists/Technical/tbmc.pdf
[32] M. Vidyasagar, *Nonlinear systems analysis*. Englewood Cliffs, NJ: Prentice Hall, 1992.
[33] N. Kottenstette and P. Antsaklis, "Relationships between positive real, passive dissipative, & positive systems," *ACC*, 2010.
[34] G. Balas, R. Chiang, A. Packard, and M. Safonov, "Robust control toolbox," *MATLAB*, vol. 3, 2005.
[35] "www.lm-sensor.org."
[36] L. Brown, A. Keshavamurthy, R. M. D.S. Li, and . L. Y. V. Pallipadi, "Acpi in linux," in *Linux Symposium*, 2005, pp. 51–68.
[37] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware dvfs," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 127–133, 2010.
[38] 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 *IEEE 4th Annual Workshop on Workload Characterization*, 2001.
[39] "http://www.spec.org/."
[40] "http://ark.intel.com/Product.aspx?id=27255."
[41] 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.
[42] J. M. Maciejowski, *Predictive Control with Constraints*. Edinburg Gate, England: Pearson Eduction Limited, 2002.
[43] W. Haddad and V. Chellaboina, *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton Univ Pr, 2008.

APPENDIX A
DT INVARIANT SET THEOREM

Unlike the continuous-time invariant set theorem used to prove asymptotic stability, it is difficult to find the discrete-time (DT) version listed in the literature. Therefore, we shall recall its formulation which allows the conditions of a discrete Lyapunov function $V(x)$ to be weakened in order to prove that a system is globally asymptotically stable. In particular, we wish to consider a system which is characterised by the following finite-difference equation:

$$x(k+1) = f(x(k)),\ x(0) = x_0,\ k \in \{0, 1, \dots\}. \quad (14)$$

in which the state-vector $x(k) \in \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous-function in $\mathbb{R}^n$ and $f(0) = 0$.

Definition 1: A set $\mathcal{M} \subset \mathcal{D} \subseteq \mathbb{R}^n$ is a *positively invariant* set for the nonlinear dynamical system (15) if $s_k(\mathcal{M}) \subseteq \mathcal{M}$, for all $k \in \bar{\mathbb{Z}}_+$, where $s_k(\mathcal{M}) \stackrel{\triangle}{=} \{s_k(x) : x \in \mathcal{M}\}$. A set $\mathcal{M} \subseteq \mathcal{D} \subseteq \mathbb{R}^n$ is an *invariant set* for the dynamical system (15) if $s_k(\mathcal{M}) = \mathcal{M}$ for all $k \in \bar{\mathbb{Z}}_+$.

Theorem 3: [43, Theorem 13.5] Consider the nonlinear dynamical system (15) and assume that there exists a continuous function $V : \mathbb{R}^n \to \mathbb{R}$ such that

$$V(0) = 0 \quad (15)$$

$$V(x) > 0,\ x \in \mathbb{R}^n,\ x \neq 0, \quad (16)$$

$$\Delta V(x) = V(f(x)) - V(x) \leq 0,\ x \in \mathbb{R}^n, \quad (17)$$

$$V(x) \to \infty \text{ as } \sqrt{x^{\mathsf{T}} x} \to \infty. \quad (18)$$

Furthermore, assume that the set $\mathcal{R} \stackrel{\triangle}{=} \{x \in \mathbb{R}^n : \Delta V(x) = 0\}$ contains no other *invariant set* $\mathcal{M}$ other than the set $\{0\}$. Then the zero solution $x(k) = 0$ to (15) is globally asymptotically stable [3].

[3]Thanks to Andrew Teel for pointing out a misquote on the initial presentation of Theorem 13.5