On Dynamic Thermal Conditions in Mixed-Criticality Systems

Seyedmehdi Hosseinimotlagh, Ali Ghahremannezhad, and Hyoseung Kim
University of California, Riverside
{shoss007, aghah001, hyoseung}@ucr.edu

***Abstract*—The rising demand for powerful embedded systems to support modern complex real-time applications signifies the on-chip temperature challenges. Heat conduction between CPU cores interferes in the execution time of tasks running on other cores. The violation of thermal constraints causes timing unpredictability to real-time tasks due to transient performance degradation or permanent system failure. Moreover, dynamic ambient temperature affects the operating temperature on multi-core systems significantly.**

In this paper, we propose a thermal-aware server framework to safely upper-bound the maximum operating temperature of multi-core mixed-criticality systems. With the proposed analysis on the impact of ambient temperature, our framework manages mixed-criticality tasks to satisfy both thermal and timing requirements. We present techniques to find the maximum ambient temperature for each criticality level to guarantee the safe operating temperature bound. We also analyze the minimum time required for a criticality mode change from one level to another. The thermal properties of our framework have been evaluated on a commercial embedded platform. A case study with real-world mixed-critical applications demonstrates the effectiveness of our framework in bounding operating temperature under dynamic ambient temperature changes.

I. Introduction

Ensuring continuous operation with high assurance in the physical environment remains a significant challenge to cyber-physical systems (CPS). This is particularly important for safety-critical applications with real-time mixed-criticality components, e.g., automotive, aerospace, manufacturing, and defense systems, where even occasional timing failures of high-criticality components can lead to catastrophic consequences. Various types of unexpected changes in the physical environment may affect the system behavior and contribute to the difficulty of this problem.

Ambient temperature is one of the key factors that affect many mixed-criticality CPS applications. For instance, in automobiles, a report from the National Renewable Energy Laboratory of the US Department of Energy [14] indicates that cabin air temperature can reach up to and 82°C in Phoenix, Arizona. The heat generated by the engine worsens the ambient temperature level of nearby electronic control units [35]. Another example is a fire-containment drone [44]. Even with a heat protection shield, the drone's computing system starts warning when the ambient temperature reaches 35°C and becomes nonoperational at 40°C. This also limits the minimum distance from the drone to a fire hazard.

While dynamic ambient temperature is an important problem, most thermal management schemes for real-time embedded systems [1, 2, 11, 22, 27, 28] assume fixed, room-level ambient temperature and focus only on the temperature increase caused by the computing system itself. CPS are expected to run in various physical environments; hence, the assumption of room temperature made by prior work limits their practical applicability. There is recent work [50] considering dynamic ambient temperature but it assumes a uni-processor single-criticality system.

Under harsh ambient temperature, a system may not be able to utilize 100% of CPU time even if the CPU runs at the minimum possible frequency with active/passive cooling packages. The operating temperature of the CPU may still reach the maximum thermal constraint, resulting in temporary system shutdown or permanent hardware damage. In such a condition, the only option left to ensure timing and thermal guarantees of more critical tasks is to secure cooling time by suspending less critical tasks. In other words, although DVFS [16, 32, 36, 39, 51] and cooling packages [8, 12, 18, 19] can help tolerate high ambient temperature, it is inevitable to consider only partial operations of the system.

In this work, we aim to design a system that offers different levels of assurance against ambient temperature changes. This is different from the well-known Vestal model [45], which focuses on varying assurance of execution time, but it shares the spirit of addressing uncertainties in real-time system design. To avoid confusion, we clarify our mixed-criticality model as follows:

Definition 1. Thermally mixed-criticality systems *are the systems that assure ambient temperature changes and heat dissipation from lower-criticality task execution do not adversely affect the real-time schedulability of higher-criticality tasks.*

In the thermally mixed-criticality model, ambient temperature plays a key role in determining the maximum amount of workload that can be executed on the CPU. The following questions still remain unanswered by the state-of-the-art:

- Up to what ambient temperature is the system fully or partially operational? Specifically, for a given criticality level of a mixed-criticality system, can we find the corresponding *critical ambient temperature*, under which real-time tasks with that or higher criticality are guaranteed to meet their deadline at the expense of lower-criticality tasks?
- Can we take into account the effect of dynamic ambient temperature along with heat conduction on a modern multi-core processor?

- If the system moves from a hot to a cold region, how long will it take for the system to cool down and safely resume the operation of low-criticality tasks, without violating the processor thermal constraint?

This paper presents a multi-core mixed-criticality scheduling framework with ambient temperature awareness. In our proposed framework, thermal-aware servers are used to bound heat generation at each criticality level and the criticality mode change is triggered by ambient temperature changes. This is the first work to address the aforementioned limitations and provides analytical guarantees on the timely execution of critical components in dynamic thermal conditions.

Contributions. The contributions of this paper are as follows:

- We show that the problem of thermal-aware real-time scheduling can be decomposed into thermal schedulability (how much CPU budget is usable under thermal constraints) and timing schedulability (if tasks are schedulable using given budget). Our thermal schedulability achieves the simplicity in timing analysis by ensuring that the budget is guaranteed to be made available for any execution patterns without violating thermal constraints.
- We extensively analyze the thermal safety of a multi-core system and bound the maximum operating temperature that the system can reach. At a specific ambient temperature level, we characterize the worst-case thermal behavior of a system and also determine the minimum time for the system to transition from one criticality level to a lower level.
- We introduce the notion of *idle thermal servers* that allow bounding the maximum operating temperature caused by multiple preemptive active servers scheduled dynamically on a multi-core processor for a given mixed-criticality taskset.
- We perform a case study on mixed-criticality applications running on an ODROID-XU4 embedded platform, and evaluate our framework and analysis in different ambient temperature levels.

II. Related Work

There exist extensive studies on bounding the maximum operating temperature in non-real-time multi-core systems [16, 17, 32, 36, 39, 51], most of which propose adjusting CPU clock speed. The authors of [16, 17] proposed a feedback loop that dynamically controls processor temperature by either adapting CPU utilization or scaling frequency to satisfy thermal constraints in varying ambient temperature environment. In [36] and [39], the authors proposed proactive frequency scheduling to improve overall system performance under different ambient temperatures. Although average-case performance degradation has been discussed in these papers, they cannot be directly applied to our problem.

The notion of *hot* and *cold* tasks has been introduced [8, 22, 24, 28, 50, 51] in both real-time and non-real-time systems. They propose scheduling algorithms to interleave hot and cold tasks, adjust the CPU frequency, and force idling time for the CPU to cool down after running hot tasks. In most of them, the scheduling problem is either to find task execution order or to reduce the size of lengthy hot tasks [22] in a fixed schedule. However, DVFS may cause a considerable reduction in system reliability over time [23, 30, 48] and may be unsupported in some embedded devices.

There exists extensive research on real-time uni-processor systems with stringent thermal constraints [1, 3, 4, 9, 10, 22, 24, 27, 46, 50]. In uni-processor systems, thermal dependency between workloads is only *temporal*. However, in multi-core systems, because of heat dissipation between cores, there also exists *spatial* thermal dependency between the execution of workload on one core and those on other cores. Due to this reason, the work on uni-processor systems cannot be used safely in multi-core real-time systems.

The notion of thermal servers (either by injecting of idle tasks or using thermal-aware servers explicitly) have been proposed in the literature of real-time systems for uni-processors [1, 22, 27, 28] and multi-core platforms [2, 11]. In particular, the authors of [11] introduced a novel technique for periodic tasks executing on multi-core platforms. This technique introduces an Energy Saving (ES) task that runs with the highest priority and captures the sleeping time of CPU cores. The technique can be seen as an alternative to a thermal-aware server because the ES task effectively models the budget-depleted duration of a thermal server. The authors of [2] proposed thermal-isolation servers that avoid the thermal interference among tasks in temporal and spatial domains with thermal composability. Despite their achievements in isolating the thermal-aware design from real-time schedulability analysis, these techniques assume a fixed schedule for idle task or periodic servers, and are inapplicable to dynamically-scheduled (e.g., priority-based) servers in multi-core platforms. Since priority-based preemptive servers are widely used in many real-time systems, such as real-time virtualization [25, 47], the restriction imposed by these prior thermal servers is a significant limiting factor.

Matrix exponential is a well-known approach to solve the first-order linear system of differential equation. In [34], the authors proposed a technique based on the numerical Newton–Raphson method to solve the thermal equations in the steady state for each power change. Based on this, the work in [7, 29, 37, 49] finds the worst-case peak temperature by exhaustively searching all possible patterns. Unlike prior work, the unique contribution of our work is that we solve the temperature equation for oscillating power signals analytically, by representing them as continuous functions, and analyze the worst-case peak temperature directly for multi-core platforms. It is worth noting that our work not only proves the worst case for peak temperature but also reduces computational complexity considerably.

There exists some research focusing on varying ambient temperature in real-time systems [22, 46, 50]. However, there is no discussion in these studies about mixed-criticality systems and the effect of harsh ambient temperature change on the schedulability of critical tasks.

To the best of our knowledge, there is no prior work on multi-core mixed-criticality systems with the consideration of the effect of ambient temperature change.

III. System Model

A. Power Model

The total power consumption of CMOS circuits is modeled as the summation of dynamic and static powers [6], i.e., $P(t) = P_S(t) + P_D(t)$. Static power P_S depends on the semiconductor technology and the operating temperature caused by current leakage. Hence, it can be modeled as: $P_S(t) = k_1\theta(t)+k_2$, where k_1 and k_2 are technology-dependent system constants, and $\theta(t)$ is the operating temperature [31]. Dynamic power $P_D(t)$ is the amount of power consumption due to the processor operating frequency f, modeled as $P_D = k_0 f^s$, where s and k_0 are system constants that depend on the semiconductor technology.

B. Thermal-aware Mixed-criticality Servers

We consider multiple preemptive thermal-aware servers for each CPU core. Each server is statically associated with one core and does not migrate to another core at run-time. A server v_i is modeled as $v_i = (C_i, T_i, L_i)$, where C_i is the maximum execution budget of the server v_i, T_i is its budget replenishment period, and L_i is its criticality level. Servers are ordered in an increasing order of priorities, i.e. $i < j$ implies that a server v_i has lower priority than a v_j. At the criticality level of l, only the servers with criticality level higher than or equal to l (i.e., $L_i \geq l$) are eligible to execute.

For budget replenishment policies, we consider *polling* [40], *deferrable* [42], and *sporadic servers* [41]. Let J_i denote the task release jitter relative to the server release. The value of J_i is T_i under the polling server policy and $T_i - C_i$ under the deferrable and sporadic server policies [5].

C. Task Model

This work considers sporadic tasks with implicit deadlines under *partitioned fixed-priority preemptive scheduling*, which is widely used in many practical systems. Each task τ_i is statically allocated to one thermal server (thus to the corresponding CPU core of that server) with a unique priority. In each server, tasks are labeled in an increasing order of priority, i.e., $i < j$ implies τ_i has lower priority than τ_j. There also exist non-real-time tasks running with the lowest priority level in the server and they execute only if there is no real-time task ready to execute and the server has remaining budget. A task τ_i is modeled as $\tau_i = (E_i, D_i)$ where E_i is the worst-case execution time (WCET) of task τ_i, D_i denotes the minimum inter-arrival time of τ_i which is implicit deadline of τ_i. It is worth mentioning that the simplicity in timing schedulability achieved by our work enables easy adoption of more complex task models. For instance, the analysis for tasks with critical sections under hierarchical scheduling [21, 26] is directly applicable to our work since we ensure periodic resource budget.

D. Criticality Model

In this work, there exists a set of m criticality levels $L = \{l_1, l_2, \ldots, l_m\}$. At criticality mode l, only the servers (and tasks within these servers) with criticality level higher than or equal to l are eligible to execute. Thus, for each criticality level l, there exists a subset of servers $V^l \subseteq V$ and a subset of taskset $\Gamma^l \subseteq \Gamma$ that execute.

Definition 2. Critical ambient temperature *of a criticality level l is the maximum ambient temperature that the system can execute eligible servers $v \in V^l$ without violating the system's maximum temperature constraint.*

Details on how criticality mode changes is given in Sec. IV.

IV. Framework Design

This section presents the overall design-time and run-time aspects of our framework and how the criticality mode changes. With our framework, all tasks in a system run within thermal-aware servers. A criticality mode change is triggered by ambient temperature, which can be obtained from an off-chip temperature sensor. If the criticality mode changes from a lower criticality level to a higher one, i.e., the critical ambient temperature has been reached, the framework terminates the lower-criticality servers and their tasks immediately. This design guarantees that the lower-criticality tasks have no effect on the *thermal* and *timing* schedulability of tasks running in servers with higher criticality levels. The timing schedulability of tasks refers to the ability to complete their execution by the deadline. Thermal schedulability is defined as follows.

Definition 3. Thermal schedulability *is to guarantee that under any task execution patterns, the CPU does not exceed the maximum temperature constraint.*

When the ambient temperature changes from a higher criticality level to a lower one, lower-criticality servers and their tasks do not resume immediately. The reasoning is that it takes time for the CPU to cool down to reach the safe temperature level that the increase in workload (due to resuming the lower-criticality tasks) will not lead to a temperature violation. Therefore, in the transition to a lower criticality level, only higher-criticality servers (and also their tasks) still run, and after reaching the safe temperature, then lower-criticality servers (and their tasks) resume. Let *shifting time* denote this delay. We will later determine this delay as a function of physical characteristics and system settings.

The state diagram of criticality mode changes is illustrated in Fig. 1. The criticality mode of the system is determined by the associated servers of the current state (e.g., V^l for level l), and the change of the state is triggered by the monitored ambient temperature. If the ambient temperature goes higher than the critical ambient temperature of the highest criticality mode l_m, the system will shutdown. An increase in the ambient temperature leads the system state to transition to a higher criticality mode immediately. However, a transition to a lower criticality mode involves a shifting state. Let S_i denote the shifting state from one criticality mode l_{i+1} to its

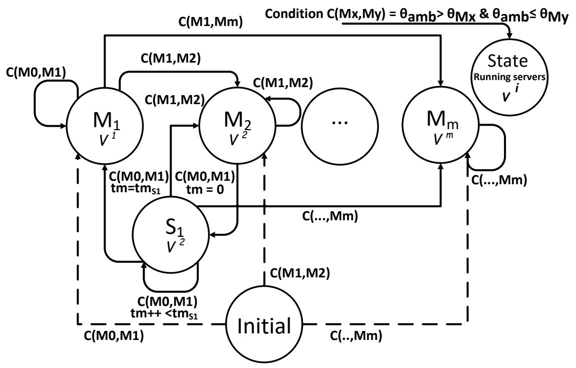


Figure 1: Criticality mode change diagram

immediate lower mode l_i, and M_i represents the state of the criticality mode l_i. After staying in S_i for tm_{Si} time units and the ambient temperature is still under the safe level, the system state will change to M_i.

In summary, the design-time analysis and the run-time support of our framework are as follows:

Design-time:

1) Check the timing schedulability of tasks for each criticality.
2) Find the parameters of thermal-aware servers that ensure thermal schedulability for each criticality.
3) Compute the corresponding critical ambient temperature for each criticality.
4) Compute the shifting time from each criticality level to its immediate lower one.

Run-time:

1) Monitor ambient temperature.
2) If ambient temperature exceeds the critical ambient temperature of the current criticality level l_i, a) switch to a higher criticality l_{i+1}, and b) terminate servers with criticality less than l_{i+1} immediately.
3) If ambient temperature decreases below the critical ambient temperature of one lower criticality level l_{i-1}, a) switch to the lower criticality l_{i-1}, b) wait for *shifting time*, and c) resume servers with criticality l_{i-1}.

V. Thermal Analysis

In this section, we first develop a generalized thermal model to represent the CPU temperature as a function of a generic input power signal. We show the relation of workload and ambient temperature level under the maximum thermal constraint in a steady state. The shifting time to a lower-criticality level will be discussed. Finally, we prove the worst-case scenario of task arrivals in invariant ambient temperature at a specific criticality level in multi-core platforms.

A. General thermal model for periodic power signal

The thermal behavior of the CPU is modeled when a generic periodic power signal generates heat dissipation, which results in temperature oscillations. The temperature function of the CPU is analytically extracted based on the ambient temperature, workload, physical and geometrical properties, and the thermal resistances between the CPU and surroundings.

A generic power signal is a periodic step function:

$$P(t) = \begin{cases} P_S & \textit{Sleeping time} \\ P_S + P_D & \textit{O.W.} \end{cases}$$

Fig. 2 illustrates this power signal function where t_{wk} and t_{slp} denote the execution time and the sleeping time of a periodic workload, u is the CPU utilization (i.e., $u = \frac{t_{wk}}{T}$), and ϕ is the release offset.

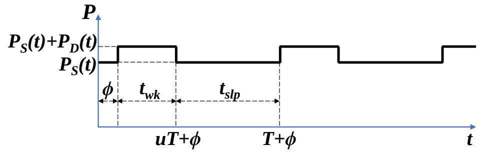


Figure 2: A generic periodic power signal.

In order to derive a continuous temperature function, here we represent the periodic step function of the power signal as the following Fourier series:

$$P(t) = P_S + P_D u + P_O \tag{1}$$

where

$$P_O = \sum_{k=1}^{\infty} \frac{2P_D \sin(uk\pi)}{k\pi} \cos\left(\frac{2k\pi}{T}\left(t - \frac{uT}{2} - \phi\right)\right)$$

Note that using this continuous representation of the power signal is advantageous in deriving a straightforward formula for a temperature equation, compared to iteratively applying the recursion relation with new initial conditions at every period. First, we define $\theta(t) = \Theta_{CPU}(t) - \Theta_{amb}(t)$ as the temperature difference of the CPU core and the ambient. Then, the rate of temperature change can be captured by the following differential equation [2, 11]:

$$\frac{d\theta(t)}{dt} = A\theta(t) + BP(t) \tag{2}$$

where A and B are scalar values determined based on the system inner thermal resistance and capacitance. Substituting $P(t)$ from Eq. 1, we solve Eq. 2 for the CPU temperature. Assuming the temperature difference at the initial time t_0 is θ_0, the temperature of the CPU can be written as:

$$\begin{aligned} \theta(t) = \theta_0 e^{A(t-t_0)} - \frac{B}{A}(P_S + P_D u)\left(1 - e^{A(t-t_0)}\right) + \\ B\left(S(A, P_D, u, T, \phi, t) - S(A, P_D, u, T, \phi, t_0)e^{A(t-t_0)}\right) \end{aligned} \tag{3}$$

where S is a function defined as:

$$S(\beta, P, u, T, \phi, t) = -\sum_{k=1}^{\infty} \frac{2PT\sin(uk\pi)}{k\pi\left(T^2\beta^2 + 4k^2\pi^2\right)} \times \left(-T\beta\cos\left(\frac{2k\pi}{T}(t-\psi)\right) + 2n\pi\sin\left(\frac{2k\pi}{T}(t-\psi)\right)\right)$$

with $\psi = \frac{uT}{2} + \phi$.

The parameters A and B in Eq. 3 are the system characteristics which depend on the thermal resistances, heat capacity, CPU mass, and thermal convection of the ambient. It is worth mentioning that the thermal response of a system with constant power dissipation can be derived as a special case of Eq. 3 by considering $u = 1$:

$$\theta(t) = \theta_0 e^{A(t-t_0)} - \frac{B}{A}(P_S + P_D)\left(1 - e^{A(t-t_0)}\right) \tag{4}$$

If $t_0 = 0$, the well-known expression for the constant power dissipation case can be obtained from Eq. 4:

$$\theta(t) = \alpha + (\theta_0 - \alpha)e^{\beta t} \tag{5}$$

where $\beta = A$, and $\alpha = -\frac{B}{A}(P_S + P_D)$.

For any u, the thermal response of the system with a constant power signal reaches the steady state. From Eq. 4:

$$\theta_s(t) = \alpha = -\frac{B}{A}\left(P_S + P_D\right). \tag{6}$$

For the case where the CPU utilization is not 100%, the temperature still oscillates in the steady state, but the oscillation stays within a certain range given by the minimum and the maximum steady-state temperatures (θ_L and θ_M). We can derive the steady state thermal response from Eq. 3:

$$\theta_s(t) = -\frac{B}{A}\left(P_S + P_D u\right) + BS(A, P_D, u, T, \phi, t) \tag{7}$$

where $S(A, P_D, u, T, \phi, t)$ represents the oscillation. Based on this general expressions, we will give details on the thermal behavior of multiple CPU cores under transient and steady conditions in Sec. V-B.

1) Workload, ambient, and maximum temperature relations: In the steady state condition, the relation between workload, ambient temperature Θ_{amb}, and the maximum operating temperature Θ_M can be determined based on the model developed above. The temperature difference θ_M can be written as:

$$\theta_M = \Theta_M - \Theta_{amb} = -\frac{B}{A}\left(P_S + P_D\frac{1 - e^{AuT}}{1 - e^{AT}}\right) \tag{8}$$

Therefore, if Θ_M is used as a thermal constraint, the maximum ambient temperature for a given utilization u is expressed as follows:

$$\Theta_{amb} = \Theta_M + \frac{B}{A}\left(P_S + P_D\frac{1 - e^{AuT}}{1 - e^{AT}}\right) \tag{9}$$

Also, another useful expression can be derived for the workload based on the constraint maximum temperature, ambient temperature, and the power signal:

$$u = \tfrac{1}{AT}\ln\left(1 + \tfrac{A}{B}\tfrac{\Theta_M - \Theta_{amb} + (B/A)P_S}{P_D}\left(1 - e^{AT}\right)\right) \tag{10}$$

2) Time shifting and transient analysis: We derive average steady-state temperature by removing the oscillations from the above equations. This is especially useful for transient phase analysis. From Eq. 3, the following can be derived:

$$\begin{aligned}\theta_{ave}(t) = \theta_0 e^{A(t-t_0)} - \frac{B}{A}\left(P_S + P_D u\right)\left(1 - e^{A(t-t_0)}\right) \\ -BS(A, P_D, u, T, \phi, t_0)e^{A(t-t_0)}\end{aligned} \tag{11}$$

$\theta_{ave}(t)$ can be approximated by taking the first few terms of the series for $S(A, P_D, u, T, \phi, t_0)$. It can be seen from Eq. 11 that the plateau of θ_{ave} in the steady state is $-\frac{B}{A}(P_S + P_D u)$.

Based on the expression of the $\theta_{ave}(t)$, shifting time can be calculated which is defined as the time it takes for the system to reach a new steady-state condition when system parameters are changed. We calculate the shifting time that the CPUs take to reach to 99% of the new steady-state condition from an initial temperature difference of θ_0. Assuming that S_k is the value of $S(t_0)$ taking the first k terms, we have:

$$t_{shift} = \frac{1}{A}\ln\left(\left|\frac{0.01\frac{B}{A}(P_S + P_D u)}{\theta_0 + \frac{B}{A}(P_S + P_D u) + BS_k}\right|\right) \tag{12}$$

Fig. 3 illustrates temperature profile $\theta(t)$, oscillating steady state temperature $\theta_s(t)$, and average temperature $\theta_{ave}(t)$.

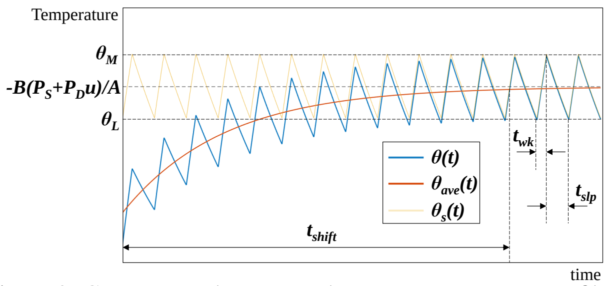


Figure 3: Current, steady state, and average temperature profiles.

B. General thermal model for multi-core platforms

Similar to the single-core case, a general thermal model can be developed for a multi-core platform with a periodic input power signal. In this subsection, we represent the power signal as a continuous function and show its benefit in simplifying the final solution. For a multi-core platform with n cores, we can have n eigenvalues that define the thermal response of the system. Therefore, we use matrix notations to solve the differential equations. After performing a thermal analysis, the rate of CPUs' temperature change can be denoted as:

$$\frac{d\boldsymbol{\theta}(t)}{dt} = \mathbf{A}\boldsymbol{\theta}(t) + \mathbf{B}\mathbf{P}(t) \tag{13}$$

where $\mathbf{A}$, an $n \times n$ matrix, and $\mathbf{B}$, a diagonal $n \times n$ matrix, are determined by the inner thermal resistance and capacitance of the system. $\boldsymbol{\theta}$ is an $n \times 1$ matrix of the CPU cores' temperature difference, and $\mathbf{P}(t)$ is the $n \times 1$ matrix of CPU cores' power signal functions. If $\boldsymbol{\theta_0}$ is the initial temperature difference matrix at t_0, the solution of Eq. 13 can be written as:

$$\boldsymbol{\theta}(t) = e^{(t-t_0)\mathbf{A}}\boldsymbol{\theta}_0 + \int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\mathbf{P}(s)ds \tag{14}$$

The first part is the homogeneous solution which is the thermal response due to the initial temperature difference from the ambient. The second part is the non-homogeneous solution caused by the input power signal. The temperature rise due to a power signal is independent of the initial condition. $e^{(t-t_0)\mathbf{A}}$ is the matrix exponential of $\mathbf{A}$. For a platform with n CPU cores, we denote the eigenvalues as β_1, β_2, ..., and β_n with $\beta_1 \geq \beta_2 \geq ... \geq \beta_n$. To solve Eq. 14, we first show that the solution of power signals can be obtained as the sum of the solutions of each signal.

Theorem 1. *(Superposition) Thermal response due to any combinations of power signals is equivalent to the sum of thermal responses caused by each of those power signals.*

Proof. Assume that $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are the thermal responses caused by executions $\mathbf{P}_1(t)$ and $\mathbf{P}_2(t)$. Also, $\boldsymbol{\theta}_3$ is the thermal response caused by $\mathbf{P}_3(t) = \mathbf{P}_1(t) + \mathbf{P}_2(t)$. From Eq. 13:

$$\frac{d\boldsymbol{\theta}_1(t)}{dt} = \mathbf{A}\boldsymbol{\theta}_1(t) + \mathbf{B}\mathbf{P}_1(t), \text{ and } \frac{d\boldsymbol{\theta}_2(t)}{dt} = \mathbf{A}\boldsymbol{\theta}_2(t) + \mathbf{B}\mathbf{P}_2(t).$$

By adding these two differential equations, we have:

$$\frac{d}{dt}\left(\boldsymbol{\theta}_1(t) + \boldsymbol{\theta}_2(t)\right) = \mathbf{A}\left(\boldsymbol{\theta}_1(t) + \boldsymbol{\theta}_2(t)\right) + \mathbf{B}\left(\mathbf{P}_1(t) + \mathbf{P}_2(t)\right)$$

This is equivalent to the differential equation of $\boldsymbol{\theta}_3$:

$$\frac{d\boldsymbol{\theta}_3(t)}{dt} = \mathbf{A}\boldsymbol{\theta}_3(t) + \mathbf{B}\mathbf{P}_3(t) = \mathbf{A}\boldsymbol{\theta}_3(t) + \mathbf{B}\left(\mathbf{P}_1(t) + \mathbf{P}_2(t)\right)$$

Therefore, $\boldsymbol{\theta}_3(t) = \boldsymbol{\theta}_1(t) + \boldsymbol{\theta}_2(t)$. ■

Let $\mathbf{V}$ denote an $n \times n$ matrix of eigenvectors of $\mathbf{A}$ where column i is the i^{th} eigenvector. Also, $\mathbf{D}$ is a diagonal $n \times n$ matrix in which the diagonal entries are the eigenvalues of $\mathbf{A}$. The solutions of the non-homogeneous part in Eq. 14 can be written as:

$$\begin{aligned}\int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\mathbf{P}(s)ds &= \int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\left(\mathbf{P}^{\infty} + \mathbf{P}_o(s)\right)ds \\ &= \int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\mathbf{P}^{\infty}ds + \int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\mathbf{P}_o(s)ds\end{aligned} \tag{15}$$

where $\mathbf{P}^{\infty}$ and $\mathbf{P}_o(s)$ are the constant and oscillating part of the power signal matrix, respectively.

$$\mathbf{P}^{\infty} = [P_j{}^{\infty}]_{n\times1} = [P_{S_j} + P_{D_j}u_j]_{n\times1}$$

$$\mathbf{P}_o = [P_{o_j}] = \left[\sum_{k=1}^{\infty} \frac{2P_{D_j}}{k\pi}\sin\left(u_j k\pi\right)\cos\left(\frac{2k\pi}{T_j}\left(s-\psi_j\right)\right)\right]_{n\times1}$$

with $\psi_j = \frac{u_j T_j}{2} + \phi_j$. Since $\mathbf{P}^{\infty}$ is constant,

$$\int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\mathbf{P}^{\infty}ds = -\mathbf{A}^{-1}\left(\mathbf{I} - e^{(t-t_0)\mathbf{A}}\right)\mathbf{B}\mathbf{P}^{\infty} \tag{16}$$

For the second integral we have:

$$\begin{aligned}\int_{t_0}^{t} e^{(t-s)\mathbf{A}}\mathbf{B}\mathbf{P}_o(s)ds &= \mathbf{V}e^{t\mathbf{D}}\int_{t_0}^{t} e^{-s\mathbf{D}}\mathbf{V}^{-1}\mathbf{B}\mathbf{P}_o(s)ds \\ &= \mathbf{V}e^{t\mathbf{D}}.\left[\sum_{i=1}^{n}\int_{t_0}^{t} e^{-\beta_j s}V_{ji}^{-1}B_{ii}P_{o_i}(s)\right]_{n\times1}\end{aligned} \tag{17}$$

Similar to the single core case, we can use S in Eq. 3 but in a matrix form. We define the matrix $\mathbf{S}$ as:

$$\mathbf{S}(t) = [S_{ij}(t)]_{n\times n} = \left[S(\beta_i, P_{D_j}, u_j, T_j, \phi_j, t)\right]_{n\times n}$$

We have $\int e^{-\beta_j s}P_{o_i}(s)ds = e^{-\beta_j s}S_{ji}(s)$. Therefore:

$$\begin{aligned}&\left[\sum_{i=1}^{n}\int_{t_0}^{t} e^{-\beta_j s}V_{ji}^{-1}B_{ii}P_{o_i}(s)ds\right]_{n\times1} \\ &= \left[\sum_{i=1}^{n} e^{-\beta_j t}V_{ji}^{-1}B_{ii}S_{ji}(t) - e^{-\beta_j t_0}V_{ji}^{-1}B_{ii}S_{ji}(t_0)\right]_{n\times1}\end{aligned}$$

To write this in a matrix notation, we define $\mathbf{B}' = diag(\mathbf{B})$ an $n \times 1$ matrix containing the diagonal entries of $\mathbf{B}$ such that $B'_j = B_{jj}$. Then,

$$\begin{aligned}&\left[\sum_{i=1}^{n} e^{-\beta_j t}V_{ji}^{-1}B_{ii}S_{ji}(t) - e^{-\beta_j t_0}V_{ji}^{-1}B_{ii}S_{ji}(t_0)\right]_{n\times1} \\ &= e^{-t\mathbf{D}}\left(\mathbf{V}^{-1}\circ\mathbf{S}(t)\right)\mathbf{B}' - e^{-t_0\mathbf{D}}\left(\mathbf{V}^{-1}\circ\mathbf{S}(t_0)\right)\mathbf{B}'\end{aligned}$$

where $\mathbf{V}^{-1}\circ\mathbf{S}(t)$ is the Hadamard product of matrices $\mathbf{V}^{-1}$ and $\mathbf{S}(t)$, which is a matrix of the same dimension of $n \times n$ where each element ij is the product of counterpart elements ij of $\mathbf{V}^{-1}$ and $\mathbf{S}(t)$: $\left(\mathbf{V}^{-1}\circ\mathbf{S}(t)\right)_{ij} = V_{ij}^{-1}S_{ij}(t)$. Substituting this in Eq. 17 gives:

$$\begin{aligned}&\mathbf{V}e^{tD}.\left[\sum_{i=1}^{n}\int_{t_0}^{t} e^{-\beta_j s}V_{ji}^{-1}B_{ii}P_{o_i}(s)\right]_{n\times1} \\ &= \mathbf{V}\left(\mathbf{V}^{-1}\circ\mathbf{S}(t) - e^{(t-t_0)\mathbf{D}}\left(\mathbf{V}^{-1}\circ\mathbf{S}(t_0)\right)\right)\mathbf{B}'\end{aligned} \tag{18}$$

Substituting Eq. 16 and Eq. 18 in Eq. 15 results in a general solution for the thermal response of a n-core CPU platform with distinct input power signals to each core:

$$\begin{aligned}\boldsymbol{\theta}(t) =& e^{(t-t_0)\mathbf{A}}\boldsymbol{\theta}_0 - \mathbf{A}^{-1}\left(\mathbf{I} - e^{(t-t_0)\mathbf{A}}\right)\mathbf{B}\mathbf{P}^{\infty} + \\ &\mathbf{V}\left(\mathbf{V}^{-1}\circ\mathbf{S}(t) - e^{(t-t_0)\mathbf{D}}\left(\mathbf{V}^{-1}\circ\mathbf{S}(t_0)\right)\right)\mathbf{B}'\end{aligned} \tag{19}$$

This expression of $\boldsymbol{\theta}(t)$, which contains simple matrix operations, can be easily used to compute the general solution for the transient temperature profile of multi-core CPUs.

C. Worst-case execution scenarios

It is important to know the workload execution patterns which cause the peak heat dissipation. In this section, based on the model developed in the previous sections, we explore and discuss the worst-case scenarios for workload execution.

1) Consecutive workload execution: First, we prove that the time to reach the maximum temperature is minimized if all workloads execute consecutively. Suppose that for any arbitrary execution of workloads in a period of T, the CPU executes some workload for t_1, sleeps for t_2, and then wakes up and executes for t_3 time units, and these working-sleeping switches occur n times. For the CPU idling time, Eq. 5 changes to $\theta(t) = \theta_0\ e^{\beta t}$ if the static power is ignored (i.e., $\alpha = 0$). Hence, the temperature change during a period is:

$$\begin{aligned}\theta_{t_0} &= \theta_L \\ \theta_{t_1} &= \alpha + (\theta_{t_0} - \alpha) * e^{\beta t_1} \\ \theta_{t_2} &= \theta_{t_1} * e^{\beta t_2} \\ &\vdots \\ \theta_{t_n} &= \theta_{t_{n-1}} * e^{\beta t_n}\end{aligned}$$

where $\sum t_i = T$ for all $n > 1$ and $t_i > 0$.

In the steady state, the temperature of the beginning and the end of each period is equal. Therefore, considering $\theta_{t_n} = \theta_L$,

$$\theta_L = \alpha\frac{e^{\beta(t_1+t_2+\cdots+t_{n-1}+t_n)} - e^{\beta(t_2+\cdots+t_{n-1}+t_n)} + \cdots - e^{\beta t_n}}{e^{\beta T} - 1}$$

where $t_{wk} = \sum_{i=1,\,i=i+2} t_i$ is the execution time and $t_{slp} = \sum_{i=2,\,i=i+2} t_i$ is the idle time, respectively. We prove that the amount of total execution time is minimized when there is only one execution time chunk. In other words, the worst-case thermal scenario happens when all workloads execute consecutively.

For clarification, suppose the following two scenarios for a given period under the maximum temperature constraint:

- All workloads execute consecutively to reach the maximum temperature for t_{wk} time units; then, the CPU sleeps until the beginning of the next period for t_{slp} time units (Fig. 4a).
- A portion of workloads executes for t_1 time units, then the CPU sleeps for t_2 time units, and the rest of the workloads run until the CPU reaches the maximum temperature at $t_1 + t_2 + t_3$; afterwards the CPU sleeps for t_4 time units (Fig. 4b).

The following shows that the budget of a thermal server should be bounded based on the first scenario.

Theorem 2. *The amount of waking time to reach the maximum temperature constraint is minimized when all workloads execute consecutively.*

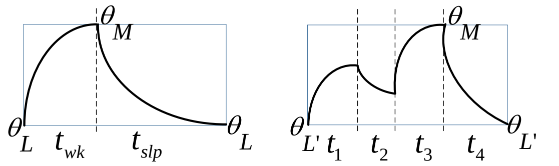


Figure 4: Temperature change in one period when the CPU operates a) for t_{wk} time units b) for t_1 and t_3 time units.

Proof. We are interested to prove that $t_w \le t_1 + t_3$. To prove the statement, we find the relation of the minimum steady state temperature in each scenario, individually. To calculate the budget for the first scenario, we first calculate the minimum steady-state temperature which is

$$\theta_L = \frac{e^{\beta\, t_{slp}}\left(\alpha - \alpha\, e^{\beta\, t_{wk}}\right)}{1 - e^{\beta\, t_{wk}}\, e^{\beta\, t_{slp}}}.$$

Accordingly, the maximum temperature in the steady state is

$$\theta_M = \alpha - e^{\beta\, t_{wk}}\left(\alpha + \frac{e^{\beta\, t_{slp}}\left(\alpha - \alpha\, e^{\beta\, t_{wk}}\right)}{e^{\beta\, t_{wk}}\, e^{\beta\, t_{slp}} - 1}\right) = \alpha\frac{e^{\beta\, t_{wk}} - 1}{e^{\beta\, T} - 1}.$$

For the second scenario, the maximum temperature is reached after $t_1 + t_2 + t_3$ time units. Therefore,

$$\theta_M = \alpha\frac{e^{\beta(t_1+t_2+t_3)} - e^{\beta(t_2+t_3)} + e^{\beta(t_3)} - 1}{e^{\beta T} - 1}$$

It is worth noting that although the minimum steady-state temperature can be different (θ_L vs. $\theta_{L'}$), in both scenarios the given maximum temperature is the same. Hence,

$$\theta_M = \alpha\frac{e^{\beta\, t_{wk}} - 1}{e^{\beta\, T} - 1} = \alpha\frac{e^{\beta(t_1+t_2+t_3)} - e^{\beta(t_2+t_3)} + e^{\beta(t_3)} - 1}{e^{\beta T} - 1}.$$

Therefore, we have $e^{\beta\, t_{wk}} = e^{\beta(t_1+t_2+t_3)} - e^{\beta(t_2+t_3)} + e^{\beta(t_3)}$. Now we want to prove that for any value of t_1, t_2 and t_3, $t_1 + t_3 \ge t_w$.

Contradiction: Suppose the hypothesis is not correct. Hence,

$$t_1 + t_3 < t_w \longrightarrow \beta\,(t_1 + t_3) > \beta\, t_w \longrightarrow e^{\beta\,(t_1+t_3)} \ge e^{\beta\, t_{wk}}$$

$$e^{\beta\,(t_1+t_3)} \ge e^{\beta\, t_{wk}} = e^{\beta(t_1+t_2+t_3)} - e^{\beta(t_2+t_3)} + e^{\beta(t_3)} \iff$$

$$e^{\beta\, t_1} \ge e^{\beta(t_1+t_2)} - e^{\beta(t_2)} + 1 \iff e^{\beta\, t_1} - 1 \ge e^{\beta(t_1+t_2)} - e^{\beta(t_2)}$$

$$\iff e^{\beta\, t_1} - 1 \ge e^{\beta(t_2)}(e^{\beta(t_1)} - 1) \iff e^{\beta(t_2)} \ge 1 ※$$

since $\beta < 0$ and $t_2 > 0$. ■

Corollary 2.1. *The maximum temperature reduces when the period and waking time are halved, since it is the special condition where $t_1 = t_3$ and $t_2 = t_4$.*

Server budget calculation under polling server budget replenishment policy. Now we calculate the "maximum" budget that a server can have while limiting the operating temperature not to exceed the given thermal constraint in a single-core platform. The worst case for the polling server happens when it exhausts all of its replenishment budget at the beginning of its period and then it sleeps until the beginning of the next replenishment period. For a server period T,

$$T = t_{wk} + t_{slp}. \quad (20)$$

In the steady state of the system, we are interested in bounding the server's maximum temperature. According to Eq. 5, $\alpha + (\theta_L - \alpha)e^{\beta t_{wk}} \le \theta_M$. By calculating the minimum temperature at the end of the period and substituting it in this formula, the maximum budget for a period T is given by:

$$t_{wk} <= \frac{1}{\beta}\ln\frac{\theta_M(e^{\beta T} - 1) + \alpha}{\alpha}. \quad (21)$$

D. Thermal back-to-back execution

Suppose that the CPU workload runs at the end of a period and the workload of the next period execute at the beginning of the period. It causes burst heat generation by back-to-back execution, which can potentially lead to thermal violation. It is noteworthy that this case has been shown in Corollary 2.1. The worst-case occurrence is when this phenomenon happens repetitively in the steady state.

Server budget calculation under deferrable server budget replenishment policy. Hereby, the maximum replenishment budget for the deferrable server is determined considering this phenomenon. One can model it as a periodic workload with doubled waking ($2t_{wk}$) and sleeping ($2t_{slp}$) times, and determine the maximum waking time.

Theorem 3. *The maximum waking time in thermal back-to-back execution is* $t_{wk} = \frac{1}{2\beta}\ln\frac{\theta_M(e^{2\beta T}-1)+\alpha}{\alpha}$.

Proof. Considering the period of workload as twice of the previous example, we have $2t_{wk} + 2t_{slp} = 2T$. The maximum temperature is reached after $2t_{wk}$ time units. So, $\theta_M = \alpha + (\theta_L - \alpha)e^{2\beta\, t_{wk}}$. Similarly to Eq. 21, we have $t_{wk} = \frac{1}{2\beta}\ln\frac{\theta_M - \alpha}{\theta_L - \alpha}$. Since the minimum temperature in the steady state is reached after $2t_{slp}$, $\theta_M e^{2\beta\, t_{slp}} = \theta_L$ which leads to $t_{slp} = \frac{1}{2\beta}\ln\frac{\theta_L}{\theta_M}$. Hence,

$$t_{slp} + t_{wk} = T \Longrightarrow \frac{1}{2\beta}\ln\frac{\theta_M - \alpha}{\theta_L - \alpha} + \frac{1}{2\beta}\ln\frac{\theta_L}{\theta_M} = T.$$

Therefore, the minimum temperature in the steady state is $\theta_L = \frac{\alpha\theta_M e^{2\beta\, T}}{\theta_M(e^{2\beta\, T}-1)+\alpha}$. By substituting the value of θ_L in t_{wk}, the maximum waking time for the period T is obtained. ■

Corollary 3.1. *The maximum achievable utilization of a server under the maximum temperature constraint θ_M is $\frac{\theta_M}{\alpha}$.*

Proof. Since the maximum utilization is obtained when the period converges to 0, the maximum utilization is

$$\lim_{T\to 0} u = \lim_{T\to 0}\frac{\frac{1}{2\beta}\theta_M 2\beta e^{2\beta T}}{\theta_M(e^{2\beta T} - 1) + \alpha} = \frac{\theta_M}{\alpha}.$$

The same approach applies to the polling servers. ■

Unlike the claim in [37, 49] which states that the worst-case peak temperature occurs when the system warms up by applying a periodic pattern to be in steady state following by a burst workload, we will show that by employing thermal-aware servers for mixed-criticality tasks, this will never happen.

Theorem 4. *The maximum temperature of a CPU for any workload execution pattern in the "steady-state" condition is less than the periodic back-to-back execution pattern.*

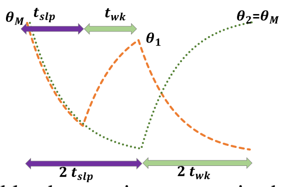


Figure 5: Workload execution pattern in the steady state.

Proof. To prove that, we follow up on the two scenarios illustrated in Fig. 5. Without loss of generality, it is assumed that the system has been already warmed up. The green line in Fig. 5 shows that back-to-back execution pattern continues in the steady-state phase. The orange line depicts the worst-case burst workload that can be executed (according to the Theorem 2 where all of the replenished budget is exhausted at the beginning of the period). Let θ_1 and θ_2 represent the maximum temperature of burst execution and normal back-to-back execution, respectively. Hence, $\theta_1 = \alpha + \theta_M\,\mathrm{e}^{T\,\beta} - \alpha\,\mathrm{e}^{\beta\,t_w}$ and $\theta_2 = \alpha + \theta_M\,\mathrm{e}^{2\,T\,\beta} - \alpha\,\mathrm{e}^{2\,\beta\,t_w}$. Let Δ_θ denote the temperature difference of these scenarios. Therefore,

$$\Delta_\theta = \theta_2 - \theta_1 = \theta_M\,\mathrm{e}^{2\,T\,\beta} - \theta_M\,\mathrm{e}^{T\,\beta} + \alpha\,\mathrm{e}^{\beta\,t_w} - \alpha\,\mathrm{e}^{2\,\beta\,t_w}.$$

By considering $u = \frac{\theta_M}{\alpha}$ and substituting the maximum waking time determined from Theorem 3, we show that $\Delta_\theta > 0$ which means

$$u\mathrm{e}^{2\,\beta\,T} - \mathrm{e}^{2\,\beta\,\frac{1}{2\beta}\,\ln\frac{\theta_M(e^{2\beta T}-1)+\alpha}{\alpha}} + \mathrm{e}^{\beta\,\frac{1}{2\beta}\,\ln\frac{\theta_M(e^{2\beta T}-1)+\alpha}{\alpha}} - u\mathrm{e}^{\beta\,T} > 0.$$

For any value of $u \in [0,1]$, since $(u-1)(1+\mathrm{e}^{\beta\,T})^2 < 0$, then we have $u + u\mathrm{e}^{2\,\beta\,T} + 2\mathrm{e}^{\beta\,T} < 1 + 2u\mathrm{e}^{\beta\,T} + \mathrm{e}^{2\,\beta\,T}$. By multiplying u in the inequality, we have

$$u^2 + 1 - 2u + u^2\mathrm{e}^{2\,\beta\,T} - 2(u-1)u\mathrm{e}^{\beta\,T} < u\mathrm{e}^{2\,\beta\,T} - u + 1$$

which leads to $u - 1 - u\mathrm{e}^{\beta\,T} > -(\frac{\theta_M\mathrm{e}^{2\,\beta\,T}-\theta_M+\alpha}{\alpha})^{\frac{1}{2}}$. Therefore, $u\mathrm{e}^{2\,\beta\,T} - \frac{\theta_M\mathrm{e}^{2\,\beta\,T}-\theta_M+\alpha}{\alpha} + (\frac{\theta_M\mathrm{e}^{2\,\beta\,T}-\theta_M+\alpha}{\alpha})^{\frac{1}{2}} - u\mathrm{e}^{\beta\,T} > 0.$ ■

E. Peak temperature analysis on multi-core platforms

Now, we extend our analysis to multi-core platforms where each CPU core consists only of one thermal-aware server. We will show that the minimum replenishment budget of polling servers on multi-core CPU happens when all of them exhaust their budget completely at the same time. Afterwards, the maximum available server budget will be determined.

Theorem 5. *The minimum value of waking time for a maximum temperature constraint is when the server on each core exhausts all its budget at once, simultaneously.*

Proof. Assume we have a dual-core CPU with the same physical characteristics and surrounding conditions. Both cores have the same periodic step power signal but with a phase difference of ϕ. For simplicity we assume P_S is zero and $P = P_D$. The input power of the first core $P_1(t)$ starts at $t = t_0$ while the input power of the second core $P_2(t)$ starts with a phase change at $t = t_0 + \phi$. We consider two cases: for case I, the phase change is zero ($\phi = 0$), and for case II it is positive ($\phi > 0$). The schematic of power signal for the two cases is depicted in Fig.6. The temperature profiles of in-phase and out-of-phase cases are compared in Fig.7.

Lemma. In a multi-core system, if the power signal of each core varies with time in the way described above, the maximum change rate magnitude of temperature is when the powers are in-phase ($\phi = 0$).

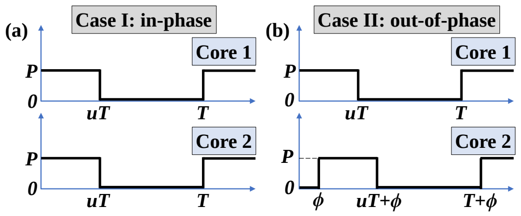


Figure 6: Power signal of the cores for (a) case I: in-phase, and (b) case II: out-of-phase.

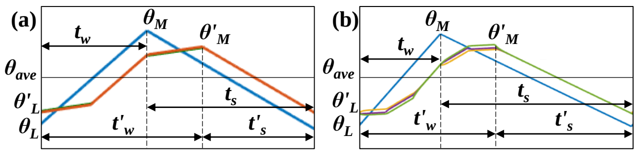


Figure 7: Temperature profiles of in-phase and out-of-phase cases for a (a) two-core and (b) three-core system. (Blue lines represent the temperature of the cores in in-phase and other colors represent temperature of the cores in out-of-phase states.)

For a dual-core CPU. According to the developed model, temperatures of the two cores between t_0 and t when the power P_1 and P_2 are constant are as follows:

$$\theta = \frac{1}{2}e^{\beta_1(t-t_0)}\begin{bmatrix}\theta_{01}+\theta_{02}\\ \theta_{01}+\theta_{02}\end{bmatrix} + \frac{1}{2}\,e^{\beta_2(t-t_0)}\begin{bmatrix}\theta_{01}-\theta_{02}\\ \theta_{02}-\theta_{01}\end{bmatrix} + \frac{B_1}{2\beta_1}\left(e^{\beta_1(t-t_0)}-1\right)\begin{bmatrix}P_1+P_2\\ P_1+P_2\end{bmatrix} + \frac{B_1}{2\beta_2}\left(e^{\beta_2(t-t_0)}-1\right)\begin{bmatrix}P_1-P_2\\ P_2-P_1\end{bmatrix}$$

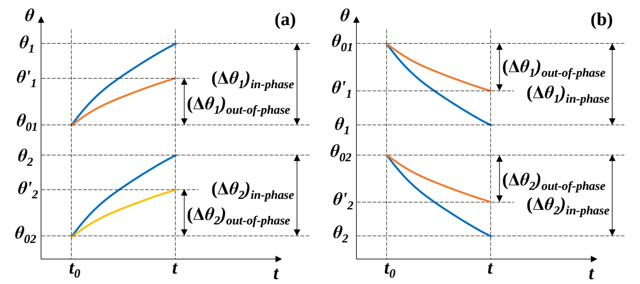


Figure 8: Comparison of temperature (a) increase and (b) decrease between in-phase and out-of-phase cases.

Assume that core 1 and 2 start from initial temperatures of θ_{01} and θ_{02} at time t_0 and get to the final temperatures of θ_1 and θ_2 at time t as shown in Fig.8 . We want to show that $|\Delta\theta_{in-phase}| > |\Delta\theta_{out-of-phase}|$ for any time slot between t_0 and t where the power remains constant. We calculate the derivative of temperature for the two cases. When there is a power for the in-phase case ($P \neq 0$), the temperate change rate is positive. For in-phase case $P_1 + P_2 = 2P$ and $P_1 - P_2 = 0$. For the out-of-phase case, $P_1 + P_2 = P$ and $P_1 - P_2 = -P$ because $P_1 = 0$ and $P_2 = P$ or vice versa. We have: (I represents in-phase and II represents out-of-phase)

$$\left(\frac{d\theta}{dt}\right)_I - \left(\frac{d\theta}{dt}\right)_{II} = \frac{B_1 P}{2}\begin{bmatrix}e^{\beta_1(t-t_0)}+e^{\beta_2(t-t_0)}\\ e^{\beta_1(t-t_0)}-e^{\beta_2(t-t_0)}\end{bmatrix}$$

Since $\beta_1 > \beta_2$ for both cores and B_1 and P are positive, then $\left(\frac{d\theta}{dt}\right)_I \geq \left(\frac{d\theta}{dt}\right)_{II}$. Same conclusion can be drawn if $P_1 = P$ and $P_2 = 0$. We can conclude that $\Delta\theta_{in-phase} \geq \Delta\theta_{out-of-phase}$ based on the fact that if $f \geq g$ in $[a, b]$, and f and g are integrable in $[a, b]$, then $\int_a^b f dx \geq \int_a^b g dx$. If the power in the in-phase case is zero ($P_1 + P_2 = 0$ and $P_1 - P_2 = 0$), the temperature change rate is negative, and for $P_1 = 0$ and $P_2 = P$ we have:

$$\left(\frac{d\theta}{dt}\right)_I - \left(\frac{d\theta}{dt}\right)_{II} = \frac{B_1 P}{2}\begin{bmatrix} -e^{\beta_1(t-t_0)} + e^{\beta_2(t-t_0)} \\ -e^{\beta_1(t-t_0)} - e^{\beta_2(t-t_0)} \end{bmatrix}$$

Since $\beta_1 > \beta_2$ for both cores $\left(\frac{d\theta}{dt}\right)_I \leq \left(\frac{d\theta}{dt}\right)_{II}$. Same conclusion can be drawn if $P_1 = P$ and $P_2 = 0$. Therefore, in case of an increase in temperature, the in-phase state has the highest temperature change rate, and in case of a decrease in temperature, the in-phase state has the lowest temperature change rate. In conclusion, $|\Delta\theta_{in-phase}| \geq |\Delta\theta_{out-of-phase}|$.

For a multi-core CPU. For a multi-core system when temperature is increasing, the difference between the temperature derivatives of the in-phase (I) and out-of-phase (II) cases between t_0 and t when the power signal does not change is as follows:

$$\left(\frac{d\theta}{dt}\right)_I - \left(\frac{d\theta}{dt}\right)_{II} = \frac{B_1}{n}e^{\beta_1(t-t_0)}\left(n - k_1\right)P\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1}$$

$$-\frac{B_1}{k_2}e^{\beta_2(t-t_0)}\begin{bmatrix} m_{21}P \\ m_{22}P \\ \vdots \\ m_{2n}P \end{bmatrix}_{n\times 1} - \ldots - \frac{B_1}{k_n}e^{\beta_n(t-t_0)}\begin{bmatrix} m_{n1}P \\ m_{n2}P \\ \vdots \\ m_{nn}P \end{bmatrix}_{n\times 1}$$

k_i, $i = 2, .., n$ are integers which satisfy $1 \leq k_i \leq n$, $i = 2, .., n$. For k_1 we have $1 \leq k_1 < n$ since there is at least one core which is idle when powers are in out-of-phase. m_{ij} can change based on how many cores are active and we have $m_{ij} < k_i$. Since $n > k_i$, the first term is positive, and since $\beta_1 \gg \beta_2, ..., \beta_n$ and all βs are negative, the other terms decrease exponentially to zero and can be neglected compared to the first term. Therefore, $\left(\frac{d\theta}{dt}\right)_I \geq \left(\frac{d\theta}{dt}\right)_{II}$. It can be shown in the same way that when there is no power in the in-phase case, the temperature is decreasing and $\left(\frac{d\theta}{dt}\right)_I \leq \left(\frac{d\theta}{dt}\right)_{II}$.

Lemma. For the described power signals, the maximum temperature for in-phase case is larger than that of the either core for the out-of-phase case ($\theta_M \geq \theta^{'}_M$).

First, let's point out that $\theta_{ave} = -\mathbf{A}^{-1}\mathbf{B}\mathbf{P}^{\infty}$ is the same for both cases in the steady state. Therefore, $\theta_M + \theta_L = \theta^{'}_M + \theta^{'}_L$. Assume that $\theta_M < \theta^{'}_M$, then $\theta_L > \theta^{'}_L$. Assume that the time it takes for the in-phase case temperature to get from θ_L to θ_M is t_w and the time it takes to get from θ_M to θ_L is t_s. Assume these times for the out-of-phase case is t'_w to get from θ'_L to θ'_M and t'_s to get from θ'_M to θ'_L. If $\theta_M < \theta^{'}_M$, and $\theta_L > \theta^{'}_L$, then t_w would be smaller than t'_w because the temperature increase rate is largest for the in-phase case. At the same time, $t_s < t^{'}_s$ since temperature decrease rate is largest for the in-phase case. Therefore, $t_w + t_s < t'_w + t'_s$. But the periods for the two cases are the same $t_w + t_s = t'_w + t'_s$. ■

Experimental Example To support our claim, we measure the temperature of big cores on the Exynos 5422 SoC [13] using IR camera FLIR A325sc [15] with the sampling rate of 60 frames per second. Four periodic computationally-intensive workloads are ran on four big cores of the board. In all cases, the CPU frequency is set to 1.4 GHz, and each workload executes every 10 seconds with the utilization of 40%. Let $\phi(i)$ denote the delay of starting time of a workload execution on core i. Table I shows the configurations of each case.

Table I: Descriptions of workload executions on big cores.

Name	description	workloads delay settings (seconds)
Case 1	all workloads execute at the same time	$\phi(1) = \phi(2) = 0$ $\phi(3) = \phi(4) = 0$
Case 2	workloads on two cores begin after those of other cores	$\phi(1) = \phi(2) = 0$ $\phi(3) = \phi(4) = 4$
Case 3	workloads on two cores begin 1 second before finishing those of on other cores	$\phi(1) = \phi(2) = 0$ $\phi(3) = \phi(4) = 3$
Case 4	workloads on each core execute with a 2-second overlap	$\phi(1) = 0$ $\phi(2) = 2$ $\phi(3) = 4$ $\phi(4) = 6$

Fig. 9 shows our observations of the maximum temperature of the SoC for the described cases in the steady state. As one can see, in Case 1 where all workloads execute synchronously, the maximum temperature of the chip reaches its highest value and its minimum temperature is the lowest among all cases. On contrary, in Case 4 where workloads of different cores have the least overlap, the maximum temperature of the chip fluctuates the least of other cases and it is close to the average temperature in the steady state. In Case 2 where there is no overlap between executions of two pairs of workloads, the rate of temperature rise is lower than the Case 3 where there is 1 second overlap within all workload executions.

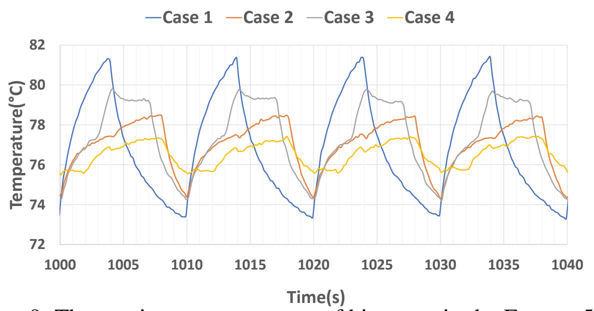


Figure 9: The maximum temperature of big cores in the Exynos 5422 captured by FLIR A325sc IR camera in the operating frequency of 1.4 GHz without heat sink.

VI. Multiple Server Analysis

In this section, we extend our analysis for multiple servers running on the multi-core CPU at each criticality level. We are interested to see if there are enough sleeping slacks between active times of servers to cool down the CPU. The cooling time must be large enough such that the CPU does not exceed the maximum temperature constraint under any circumstance. Hence, we should check if the amount of the sleeping time required for cooling is guaranteed for a given period of time.

We propose an *idle thermal server* technique in this regard. Unlike regular servers, the idle server does not execute;

instead, its budget represents the amount of time that the CPU core needs to be idle in the cooling phase. The reasoning behind this technique is to simplify the modeling of the resulting idle time from the execution of multiple regular servers as a single budget parameter. Hence, the idle server's budget can be determined such that the maximum CPU operating temperature caused by heat dissipation during the idle server's inactive time is the same as or higher than that of running regular servers under any task execution pattern. If such an idle server is schedulable, one can conclude that the given taskset is thermally schedulable. The thermal effect of multiple servers is analyzed by the complement signal of periodic idle-server execution, the worst-case behavior of which has been proved by Theorems 1-5.

After analytically finding the relation between the budget and period of the idle server, we check its schedulability.

Since the idle server does not actually exist on the CPU, it is considered as the lowest-priority server in the schedulability test. In our proposed framework, the CPU is not forced to sleep so that the proposed idle server has no effect on the timing schedulability of regular running servers.

The idle server utilization corresponds to the time during which all regular servers are deactivated. Based on this, we investigate the feasibility of the idle server with its minimum possible period within a valid range. In this work, we focus on designing homogeneous idle servers, i.e., all idle servers on different CPU cores share the same parameters at each criticality level. It is worth mentioning that this does not mean that sleeping time in different cores must happen at the same time. Finding the different idle server settings for each CPU core is beyond the scope of this paper.

Idle server design First, we compute the total utilization of the CPU core c at the criticality level l by $u_c^l = \sum_{\forall i \ \mathbb{P}(v_i)=c \wedge v_i \in V^l} \frac{C_i}{T_i}$, where $\mathbb{P}(v_i)$ is the CPU core assigned to v_i. The maximum per-core CPU utilization is then $u_{max}^l = \max_{\forall c} u_c^l$. Due to the homogeneity of idle servers on all cores, u_{max}^l is considered as the utilization of one core so that each core is supposed to be idle at least $1 - u_{max}^l$. As a result, we are looking for a server that can be schedulable with the amount of utilization of at least $1 - u_{max}^l$. Let u_{idle}^l denote this value. According to Theorem 3, the period of the idle server T_{idle}^l can be modeled as:

$$T_{idle}^l \leq \frac{\ln(1 + \alpha \frac{e^{1-u_{idle}^l + \frac{1}{2\beta}} - 1}{\theta_M})}{2\beta}. \tag{22}$$

It is worth noting that the period is determined based on the back-to-back execution under in presence of multiple servers for both polling and deferrable budget replenishment policies since servers can preempt each other and sleeping time does not happen in a contiguous manner. T_{idle}^l is an increasing function in terms of utilization. As one can figure out from Eq. 22, an increase in the workload on the CPU core (hence resulting in a decrease in u_{idle}^l) leads to a decrease in T_{idle}^l. In other words, under heavier workload, the CPU has to sleep more frequently but in a shorter duration, to satisfy the maximum temperature constraint.

Thermal schedulability Hereby, we present our proposed thermal schedulability test for a specific utilization of idle servers. The worst-case response time of each idle server of each core at each criticality level can be obtained as follows

$$R_{idle,c}^{n+1,l} = u_{idle}^l \times T_{idle}^l + \sum_{\forall i \ \mathbb{P}(v_i)=c \wedge v_i \in V^l} \left\lceil \frac{R_{idle,c}^{n,l}}{T_i} \right\rceil C_i \tag{23}$$

where $R_{idle,c}^{n+1,l}$ denotes the worst-case response time of the idle server on the core c at the criticality level l. As shown in Eq. 23, the idle server of each core can be preempted by all active servers at l. The only unknown parameter in above test is the idle server settings. Next, we discuss the optimal valid range of idle server's setting range to reduce the number of tests for each criticality level.

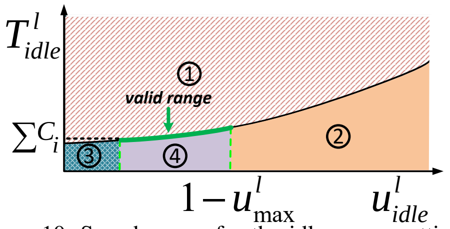


Figure 10: Search space for the idle server settings.

Optimal server setting range As discussed in Eq. 22, the period of the idle server is an increasing function in terms of its utilization. Fig. 10 plots the period with respect to utilization of the idle server. The area of the plot is divided to four regions. We discuss that only the highlighted area needs to be searched for finding the valid settings of idle servers.

- **region ①:** In this region, the CPU violates the maximum temperature constraint according to Eq. 22
- **region ②:** The utilization of idle server has to be less than $1-u_{max}^l$. Choosing a setting in this region causes the system to be unschedulabe at this criticality level.
- **region ③:** Since idle servers have the lowest priority, they cannot preempt other servers. According Eq. 23, in the worst case all servers preempt the idle servers.
- **region ④:** Valid settings can be found in this region, but it is unnecessary to search the entire of this region because there exists a valid setting with period of T', a solution is valid on the highlighted range with the same utilization value.

Therefore, the only period of $T_{idle} = \frac{\ln(1+\alpha \frac{e^{1-u_{idle}+\frac{1}{2\beta}}-1}{\theta_M})}{2\beta}$ within the optimal range of utilization, u_{idle}, needs to be checked. One may try to insert T_{idle} into Eq. 23 and assign T_{idle} to $R_{idle,c}^{n+1,l}$ to find a single optimum point. However, because of the ceiling operation in Eq. 23, finding the optimum u_{idle} would need an exhaustive search.

After finding the idle server settings, the critical ambient temperature for each criticality level is computed by using Eq. 9 with $u = 1 - u_{idle}^l$. The shifting time from criticality level $l + 1$ to l can be determined by applying Eq. 12.

Timing schedulability Due to our separate thermal schedulability analysis, we are able to use the existing response time test developed for independent tasks with no thermal constraints under hierarchical scheduling [38]:

$$W_i^{n+1,l} = E_i + \sum_{\substack{\tau_h \in \mathbb{V}^l(\tau_i)\\ h>i}} \left\lceil \frac{W_i^{n,l} + J_i + (W_h^l - E_h)}{D_h} \right\rceil E_h + \left\lceil \frac{W^{n,l} + C_j}{T_j} \right\rceil (T_j - C_j) \tag{24}$$

where $W_i^{n,l}$ is the worst-case response time of τ_i at criticality level l, $\mathbb{V}^l(\tau_i)$ is the server of τ_i, J_j is the jitter of a task running in a server v_j (see Sec III), and $W_i^{0,l} = E_i$.

VII. Evaluation

This section gives the experimental evaluation of our framework. First, we show the model validation by measuring the physical system parameters on a real platform. After the analysis of server characteristics in different ambient temperatures, we present our discussion with a case study.

A. Experimental platform

The experimental platform is an ODroid-XU4 development board [33] equipped with a Samsung Exynos5422 SoC. There exist two different CPU clusters of little Cortex-A7 and big Cortex-A15 cores, where each cluster consists of four homogeneous cores. Built-in sensors with the sampling rate of 10 Hz with the precision of 1°C are on each big CPU core to measure the chip temperature. Note that there are no temperature sensors on little cores since the power consumption and heat generation of the little cluster is considerably low. The DTM throttles the frequency of the big CPU cluster to 900 MHz when one of its cores reaches the pre-defined maximum temperature constraint of 95°C. During experiments, the CPU fan is turned off and the CPU is set to run at 1400 MHz.

B. Model Validation

According to Eq. 3 and 19, the characteristic matrices **A** and **B** can be determined by a utilization test at different CPU frequencies. A zero utilization (idle) at different ambient temperatures can reveal the range of the static and consequently the dynamic dissipation. After finding the thermal parameters of the system and model calibration, the analytical model is validated with the experimental results. For this purpose the CPU temperature is recorded at 90% utilization with a period of 1 s in Θ_{amb} of 23°C. Then utilization is decreased to 30% and the CPU is cooled down until reaching a steady state. Afterwards, the CPU working at 30% is placed in the furnace with Θ_{amb} of 42°C. The CPU temperature is recorded until the steady state. The same conditions are simulated using the developed model. Fig. 11 compares the CPU temperature recorded in the experiment at different stages of workloads and ambient temperature with the predicted values by the model. It can be seen that there is a good agreement between the model and the experimental results.

C. Workload, period, and ambient temperature relations

We investigate the effect of ambient temperature, server period, and utilization using Eq. 9 and Eq. 10. Assuming the temperature threshold of $\Theta_m = 95$°C, the maximum

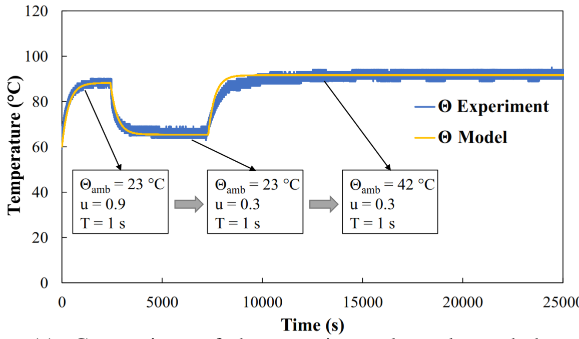


Figure 11: Comparison of the experimental results and the model prediction for CPU temperature at different workloads and ambient temperatures.

allowable utilization is plotted in Fig. 12a against period and ambient temperatures. It can be seen that for all considered periods, when the ambient temperature is increased from 23°C, the maximum workload decreases almost linearly with the ambient temperature. At higher ambient temperatures lower workloads can be used until the ambient temperature of 69°C where even an idle CPU usage will result in a working temperature equal to the threshold temperature. To better see the effect of period, the maximum workload is plotted against period at different ambient temperatures in Fig. 12b. The period has been changed from 10 ms to 30 s. It can be seen that the maximum workload decreases by increasing the value of the period in an almost linear manner. This has been discussed and confirmed in Theorem 2. Moreover, it can be concluded that the effect of ambient temperature on the maximum allowable workload is more prominent than the effect of period.

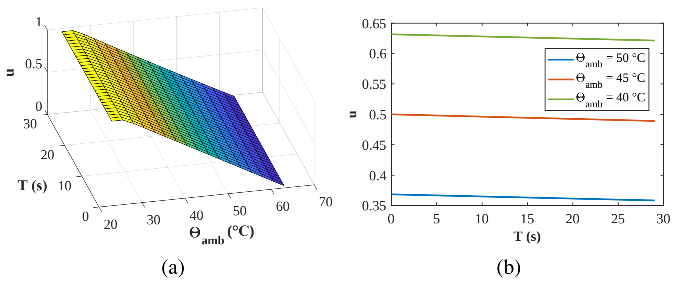
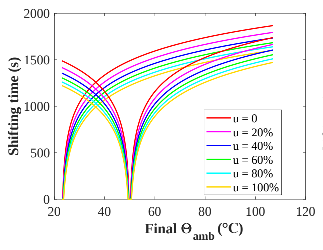


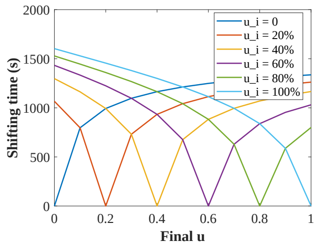
Figure 12: a) Utilization versus period and ambient temperature. b) Utilization versus period at different ambient temperatures.

D. Shifting time analysis

As mentioned before, any change in the working parameters of the system may cause a transient thermal behavior and change the steady state conditions. Here, we discuss the effects of changing workload and also ambient temperature on the thermal response of the CPU cores. In Fig. 13a. shifting times are plotted against the final ambient temperature at different workloads assuming the initial ambient temperature is 23°C or 50°C. It can be seen that at higher workloads it takes less time for the CPU to reach the steady state when ambient temperature is changed from Θ_{amb_i} to Θ_{amb_f}. Also, at all workloads, it takes more time for the CPU to reach the final ambient temperature Θ_{amb_f} if it starts from a lower initial



(a)



(b)

Figure 13: Shifting time a) from initial ambient temperature to different final ambient temperatures at various workloads, b) from different initial workloads to different final workloads at an ambient temperature of 23°C.

ambient temperature Θ_{amb_i}. Furthermore, for all utilizations, the time it takes for the CPU to reach the steady state when the ambient temperature goes up by $\Delta\Theta$ is almost equal to the time it takes when it goes down by the same amount.

In Fig. 13b. shifting times are plotted against the final workload u_f for different initial workloads u_i. It can be seen that it takes more time to reach the steady state of final u_f when starting from a lower workload u_i. Also, opposite to the case of ambient temperature, if $u_i < u_f$, it takes less time to reach the steady state when shifting from u_i to u_f (heating), compared to when shifting from u_f to u_i (cooling).

E. Case study

We emulate the mixed-criticality Flight Management System (FMS) application [2, 20] which comprises two criticality levels of H and L. The parameters of the executing real-time tasks are given in [2]. In our experiment, there exist one high-criticality and one low-criticality server on each CPU core. The budget replenishment period of each thermal-aware server is considered 50 ms under the deferrable budget replenishment policy. The budget for high-criticality and low-criticality servers are 15 ms and 27 ms, respectively. Tasks are assigned to CPU cores by using the worst-fit decreasing (WFD) heuristic for load balancing across cores and are scheduled by the Rate Monotonic (RM) policy. Since the amount of the workload in low-criticality level is insignificant to reach the maximum temperature, non-real-time tasks are also assigned to low-criticality servers with the lowest priority level.

Critical ambient temperatures have been determined by Eq. 9: 24°C for the low-criticality and 40°C for the high-criticality level. As shown in Fig. 14, the experiment has been performed in the furnace. Nordic Semiconductor Thingy:52™ IoT sensor development kit [43] is used to capture the ambient temperature with the sampling rate of 10 Hz.

Fig. 15 shows the experimental and model results for a case study with period of 50 ms. In step I, the CPU is idling for 1800 s, and then in step II, workload with $u = 95\%$ is applied at $\Theta_{amb} = 24$°C. The system is left to work with $u = 95\%$ until it reaches steady state conditions and keeps working for about 10000 s. The CPU temperature reaches to a value around 89°C. Afterwards, in step III, the CPU is placed in the furnace with $\Theta_{amb} = 40$°C and the workload is changed to 30% at

Figure 14: Experimental environment using furnace.

the same time. The temperature increases to 92°C and remains steady for about 4000 s. The CPU is then taken out of the furnace and left to work with $u = 30\%$ for a fast cooling. Finally, in step IV, the workload is set to 95% at ambient temperature $\Theta_{amb} = 24$°C. It can be seen that the developed model matches the experimental results with a good accuracy. The time step in the model is more accurate than the actual temperature sensor and temperature variations for each period can be captured by the developed model. Temperature curve from 6000 s to 6002 s is zoomed for a better comparison of the variations.

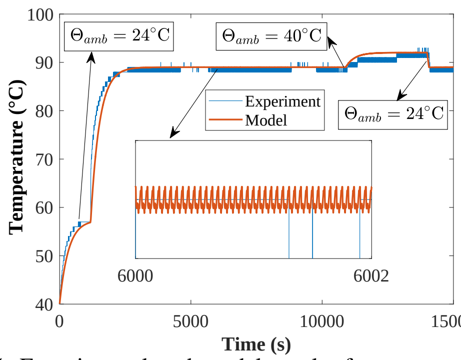


Figure 15: Experimental and model results for a case study with a period of 50 ms at two ambient temperatures and workloads.

VIII. Conclusion

In this paper, we proposed a novel mixed-criticality thermal-aware server framework to bound the maximum temperature of CPU cores in the presence of dynamic ambient temperature. In this framework, the server schedule is flexible – fully preemptive and priority-based. We investigated the thermal feasibility by analyzing the amount of slack between execution of preemptive thermal servers with the notion of idle servers. We presented a mechanism to optimally search the maximum ambient temperature for every criticality level. We provided analytical foundations to check thermal safety while temporal safety is guaranteed. Experimental results show that our proposed framework is effective in bounding the maximum temperature at every criticality level.

For future work, we plan to extend our analysis to allow different idle server settings per CPU core, which can improve the range of ambient temperature supported by mixed-criticality levels. In addition, we will study the thermal behavior of tasks to alleviate the pessimism of thermal schedulability. For instance, a task frequently accessing the system memory may generate much less heat than those being computationally-intensive. We will develop our analysis to capture the effect of cooling packages and forced heat convection.

Acknowledgment

The authors would like to thank Alexander A. Balandin and Fariborz Kargar for their help with experimental environment using the furnace. The authors are also indebted to Daniel Wong and Sheldon Tan for their help in IR thermal imaging.

References

[1] M. Ahmed, N. Fisher, S. Wang, and P. Hettiarachchi. Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources. *Sustainable Computing: Informatics and Systems*, 1(3):226–240, 2011.

[2] R. Ahmed, P. Huang, M. Millen, and L. Thiele. On the design and application of thermal isolation servers. *ACM Trans. Embed. Comput. Syst.*, 16(5s):165:1–165:19, Sept. 2017.

[3] R. Ahmed, P. Ramanathan, and K. K. Saluja. On thermal utilization of periodic task sets in uni-processor systems. In *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 267–276. IEEE, 2013.

[4] R. Ahmed, P. Ramanathan, and K. K. Saluja. Temperature minimization using power redistribution in embedded systems. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 264–269. IEEE, 2014.

[5] G. Bernat and A. Burns. New results on fixed priority aperiodic servers. In *IEEE Real-Time Systems Symposium*, 1999.

[6] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. In *2008 Design, Automation and Test in Europe*, pages 288–293, March 2008.

[7] T. Chantem, X. S. Hu, and R. P. Dick. Online work maximization under a peak temperature constraint. In *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, pages 105–110. ACM, 2009.

[8] T. Chantem, X. S. Hu, and R. P. Dick. Temperature-aware scheduling and assignment for hard real-time applications on mpsocs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(10):1884–1897, 2010.

[9] J.-J. Chen, C.-M. Hung, and T.-W. Kuo. On the minimization for the instantaneous temperature for periodic real-time tasks. In *13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'07)*, pages 236–248. IEEE, 2007.

[10] J.-J. Chen, S. Wang, and L. Thiele. Proactive speed scheduling for real-time tasks under thermal constraints. In *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 141–150. IEEE, 2009.

[11] S. M. D'Souza and R. Rajkumar. Thermal implications of energy-saving schedulers. In *ECRTS*, 2017.

[12] A. Elghool, F. Basrawi, T. K. Ibrahim, K. Habib, H. Ibrahim, and D. M. N. D. Idris. A review on heat sink for thermo-electric power generation: Classifications and parameters affecting performance. *Energy conversion and management*, 134:260–277, 2017.

[13] Exynoss 5422. https://www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-5-octa-5422, 2019.

[14] R. Farrington and J. Rugh. Impact of vehicle air-conditioning on fuel economy, tailpipe emissions, and electric vehicle range. In *Earth technologies forum*, pages 1–6. NREL Washington, DC, 2000.

[15] FLIR A325sc. https://www.flir.com/products/a325sc, 2019.

[16] Y. Fu, N. Kottenstette, Y. Chen, C. Lu, X. D. Koutsoukos, and H. Wang. Feedback thermal control for real-time systems. In *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 111–120. IEEE, 2010.

[17] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos. Feedback thermal control of real-time systems on multicore processors. In *Proceedings of the tenth ACM international conference on Embedded software*, pages 113–122. ACM, 2012.

[18] A. Ghahremannezhad and K. Vafai. Thermal and hydraulic performance enhancement of microchannel heat sinks utilizing porous substrates. *International Journal of Heat and Mass Transfer*, 122:1313–1326, 2018.

[19] A. Ghahremannezhad, H. Xu, M. A. Nazari, M. H. Ahmadi, and K. Vafai. Effect of porous substrates on thermohydraulic performance enhancement of double layer microchannel heat sinks. *International Journal of Heat and Mass Transfer*, 131:52–63, 2019.

[20] G. Giannopoulou, N. Stoimenov, P. Huang, and L. Thiele. Scheduling of mixed-criticality applications on resource-sharing multicore systems. In *Proceedings of the Eleventh ACM International Conference on Embedded Software*, page 17. IEEE Press, 2013.

[21] S. Hosseinimotlagh and H. Kim. Thermal-aware servers for real-time tasks on multi-core gpu-integrated embedded systems. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 254–266. IEEE, 2019.

[22] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu. Throughput maximization for periodic real-time systems under the maximal temperature constraint. *ACM Trans. Embed. Comput. Syst.*, 13(2s):70:1–70:22, Jan. 2014.

[23] A. Iranfar, M. Kamal, A. Afzali-Kusha, M. Pedram, and D. Atienza. Thespot: Thermal stress-aware power and temperature management for multiprocessor systems-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[24] R. Jayaseelan and T. Mitra. Temperature aware task sequencing and voltage scaling. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 618–623. IEEE Press, 2008.

[25] H. Kim and R. Rajkumar. Predictable shared cache management for multi-core real-time virtualization. *ACM Transactions on Embedded Computing Systems (TECS)*, 17(1):1–27, 2017.

[26] H. Kim, S. Wang, and R. Rajkumar. vMPCP: A synchronization framework for multi-core virtual machines. In *2014 IEEE Real-Time Systems Symposium*, pages 86–95, Dec 2014.

[27] P. Kumar and L. Thiele. Cool shapers: shaping real-time tasks for improved thermal guarantees. In *Proceedings of the 48th Design Automation Conference*, pages 468–473. ACM, 2011.

[28] P. Kumar and L. Thiele. System-level power and timing variability characterization to compute thermal guarantees. In *Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 179–188. ACM, 2011.

[29] P. Kumar and L. Thiele. Thermally optimal stop-go scheduling of task graphs with real-time constraints. In *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pages 123–128. IEEE, 2011.

[30] C. J. Lasance. Thermally driven reliability issues in microelectronic systems: status-quo and challenges. *Microelectronics Reliability*, 43(12):1969–1974, 2003.

[31] Y. Liu, R. P. Dick, L. Shang, and H. Yang. Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *2007 Design, Automation Test in Europe Conference Exhibition*, pages 1–6, April 2007.

[32] Y. Ma, T. Chantem, X. S. Hu, and R. P. Dick. Improving lifetime of multicore soft real-time systems through global utilization control. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 79–82. ACM, 2015.

[33] ODROID-XU4. http://www.hardkernel.com/, 2016.

[34] S. Pagani, J.-J. Chen, M. Shafique, and J. Henkel. Matex: Efficient transient and peak temperature computation for compact thermal models. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1515–1520. IEEE, 2015.

[35] S. Park, J.-J. Chen, D. Shin, Y. Kim, C.-L. Yang, and N. Chang. Dynamic thermal management for networked embedded systems under harsh ambient temperature variation. In *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, pages 289–294. IEEE, 2010.

[36] F. Paterna and T. S. Rosing. Modeling and mitigation of extra-soc thermal coupling effects and heat transfer variations in mobile devices. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 831–838, Nov 2015.

[37] D. Rai, H. Yang, I. Bacivarov, J.-J. Chen, and L. Thiele. Worst-case temperature analysis for real-time systems. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[38] S. Saewong, R. R. Rajkumar, J. P. Lehoczky, and M. H. Klein. Analysis of hierarchical fixed-priority scheduling. In *ECRTS*, 2002.

[39] O. Sahin and A. K. Coskun. Providing sustainable performance in thermally constrained mobile devices. In *2016 14th ACM/IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, pages 1–6, Oct 2016.

[40] L. Sha, J. Lehoczky, and R. Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. In *IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, 1986.

[41] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, 1(1):27–60, 1989.

[42] J. K. Strosnider, J. P. Lehoczky, and L. Sha. The deferrable server

algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1):73–91, 1995.

[43] Nordic Semiconductor Thingy:52 IoT Sensor Development Kit. https://www.mouser.com/new/nordicsemiconductor/nordic-thingy-52/, 2019.

[44] B. A. Toribio, C. D. Peterson, D. P. Rubenstein, and T. P. Neilan. Fire containment drone. Technical report, Worcester Polytechnic Institute, 2016.

[45] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, pages 239–243. IEEE, 2007.

[46] S. Wang and R. Bettati. Delay analysis in temperature-constrained hard real-time systems with general task arrivals. In *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, pages 323–334. IEEE, 2006.

[47] S. Xi, J. Wilson, C. Lu, and C. Gill. RT-Xen: towards real-time hypervisor scheduling in Xen. In *International Conference on Embedded Software (EMSOFT)*, 2011.

[48] Y. Xiang, T. Chantem, R. P. Dick, X. S. Hu, and L. Shang. System-level reliability modeling for mpsocs. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 297–306. ACM, 2010.

[49] H. Yang, I. Bacivarov, D. Rai, J.-J. Chen, and L. Thiele. Real-time worst-case temperature analysis with temperature-dependent parameters. *Real-Time Systems*, 49(6):730–762, 2013.

[50] K. G. S. Youngmoon Lee, Hoonsung Chwa and S. Wang. Thermal-aware resource management for embedded real-time systems. In *Embedded Software (EMSOFT), 2018 International Conference on*. IEEE, 2018.

[51] S. Zhang and K. S. Chatha. Thermal aware task sequencing on embedded processors. In *Proceedings of the 47th Design Automation Conference*, pages 585–590. ACM, 2010.