CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

An Efficient Proactive Thermal-Aware Scheduler for DVFS-enabled Single-Core Processors

Javier Pérez Rodríguez
Patrick Meumeu Yomsi

CISTER-TR-210301

2021/04/07

An Efficient Proactive Thermal-Aware Scheduler for DVFS-enabled Single-Core Processors

Javier Pérez Rodríguez, Patrick Meumeu Yomsi

CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: perez@isep.ipp.pt, pmy@isep.ipp.pt

https://www.cister-labs.pt

Abstract

For decades now, thermal rise has been spotted as one of the major constraints of performance for high-end safety-critical processors. In this context, Dynamic Voltage and Frequency Scaling (DVFS) based solutions have proven to be effective to manage the chip temperature. In this paper, we consider the scheduling problem of non-preemptive periodic tasks on a single-core processor with DVFS-enabled capabilities under thermal-aware design. We assume that the tasks are scheduled by following any Fixed-Task-Priority (FTP) scheduler such as the traditional Rate Monotonic (RM) and Deadline Monotonic (DM). Then, we propose a new scheduling scheme, referred to as NP-COIN, which makes it possible to control both the processor activity and the triggering of the cooling mechanism with as little impact on performance as possible. We provide a thorough theoretical analysis of our solution, in terms of average temperature gain and timing penalty, against the classical DVFS schedule. Finally, we validate our theoretical results and assess the performance of our solution through a real-world use-case study from the avionics domain and through intensive simulations by using synthetic test cases.

An Efficient Proactive Thermal-Aware Scheduler for DVFS-enabled Single-Core Processors

Javier Pérez Rodríguez*
CISTER, ISEP, Polytechnic Institute of Porto
Porto, Portugal
perez@isep.ipp.pt

Patrick Meumeu Yomsi*
CISTER, ISEP, Polytechnic Institute of Porto
Porto, Portugal
pmy@isep.ipp.pt

ABSTRACT

For decades now, thermal rise has been spotted as one of the major constraints of performance for high-end safety-critical processors. In this context, Dynamic Voltage and Frequency Scaling (DVFS) based solutions have proven to be effective to manage the chip temperature. In this paper, we consider the scheduling problem of non-preemptive periodic tasks on a single-core processor with DVFS-enabled capabilities under thermal-aware design. We assume that the tasks are scheduled by following any Fixed-Task-Priority (FTP) scheduler such as the traditional Rate Monotonic (RM) and Deadline Monotonic (DM). Then, we propose a new scheduling scheme, referred to as NP-COIN, which makes it possible to control both the processor activity and the triggering of the cooling mechanism with as little impact on performance as possible. We provide a thorough theoretical analysis of our solution, in terms of average temperature gain and timing penalty, against the classical DVFS schedule. Finally, we validate our theoretical results and assess the performance of our solution through a real-world use-case study from the avionics domain and through intensive simulations by using synthetic test cases.

CCS CONCEPTS

• **Computer systems organization** → Real-time systems; • **Hardware** → Thermal issues; • **Theory of computation** → Design and analysis of algorithms.

KEYWORDS

thermal-aware scheduling, non-preemptive schedulers, dynamic voltage/frequency scaling, single-core platforms.

ACM Reference Format:
Javier Pérez Rodríguez and Patrick Meumeu Yomsi. 2021. An Efficient Proactive Thermal-Aware Scheduler for DVFS-enabled Single-Core Processors. In *29th International Conference on Real-Time Networks and Systems (RTNS'2021), April 7–9, 2021, NANTES, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3453417.3453430

*Both authors contributed equally to this research.

RTNS'2021, April 7–9, 2021, NANTES, France

ACM ISBN 978-1-4503-9001-9/21/04…$15.00
https://doi.org/10.1145/3453417.3453430

1 INTRODUCTION

Over the past few decades, thermal dissipation has become one of the most challenging issues in the design of modern safety critical computing systems. As a consequence of this, the demand for efficient thermal-aware techniques has gained momentum in the real-time research community to become one of the primary factors driving the growth of the microprocessor market. Indeed, excessive processor activity during a long period of time may force the platform to reach a high and undesirable thermal level, which in turn exposes the system to a malfunctioning or even a stall. Consequently, system designers not only have to grapple with timing constraints for safety critical systems; a suitable thermal management solution is also of utmost importance as it would help avoid hot spots and keep the device temperature at acceptable levels. Among the most promising solutions, scheduling mechanisms based on Dynamic Voltage and Frequency Scaling (DVFS) principles [35, 36, 39, 40] to reduce the power consumption and consequently the average thermal dissipation of the underlying processor, have proven to be viable, adaptive and efficient under different conditions. But, in their original specifications, their reactive nature and lack of thermal modeling make them unsuitable for providing predictability and/or performance. In this paper, we circumvent this hurdle in a rather elegant manner by proposing a novel scheduling scheme, referred to as NP-COIN, which targets performance optimization under thermal constraints. We consider a single processor platform with *inter-task* DVFS-enabled capabilities and we assume the periodic fixed priority *non-preemptive* constrained-deadline task model for describing the recurring processes that occur in critical real-time systems. Then, we simulate the execution of the tasks by following NP-COIN within the *feasibility interval* [9, 23].

▹ **On considering an inter-task** DVFS**.** Roughly speaking, DVFS solutions can be grouped into two main categories:

(1) *inter-task* DVFS [3, 16, 33], where the processor speed is settled at task-level, i.e., once a task is selected for execution, the speed is not changed until it is preempted or completed; and,
(2) *intra-task* DVFS [4, 20, 32], where the speed is adjusted at run-time during the execution of the task.

Comparing the two categories, the latter usually requires some degree of compiler support to insert power management points into the application code and to explicitly call Operating System services for speed reduction. The former category does not involve such changes and thus are more practical.

▹ **On considering a non-preemptive scheme.** In non-preemptive scheduling, the processor is allocated to each task until it terminates. The main drawback of these schedulers [14, 15] is that they can lead

to large response times due to the introduction of additional blocking times caused by low priority tasks, so reducing schedulability. On the positive side, they present several advantages w.r.t. their preemptive counterpart. To name a few examples, non-preemptive schemes: (1) offer a low scheduling overhead due to the lack of preemption; (2) provide a high degree of predictability; (3) need low computational resources for scheduling which may not be the case with a preemptive scheme; (4) ensure program locality since the data is fetched from the main memory and allocated in the cache; and finally (5) by construction, naturally guarantee the exclusive access to shared resources. For all these reasons and because these schemes are widely used in real-word applications (e.g., in the avionic domain), we opted for disabling preemption completely.

▹ **Our contribution.** This paper proposes a novel *proactive* thermal-aware scheduler, referred to as NP-COIN, together with its associated schedulability analysis. The basic idea is to introduce cooling periods only when it is *compulsory* to do so during run-time to keep the processor temperature within specified parameters. NP-COIN makes it possible to control both the processor activity and the triggering of the cooling mechanism *prior* to executing the corresponding workload with as little impact on performance as possible. To the best of our knowledge, this is the first contribution to address the thermal-aware schedulability analysis problem of non-preemptive real-time tasks on DVFS-enabled single-cores. Finally, we validate our theoretical results and assess the performance of our solution by using a Mission Control Computer (MCC) use-case study from the avionics domain as well as intensive simulations through synthetic test cases.

▹ **Paper organization.** The remainder of the paper is organized as follows. We present the adopted model of execution in Section 2. In Section 3, we present the challenge and provide the reader with the main intuition behind our proposed solution. Section 4 summarizes the required state-of-the-art basics and preliminary results. Our main contribution is reported in Section 6, where we specify our thermal-aware scheduler NP-COIN in details. Section 7 reports on the experiments conducted to validate our theoretical results and Section 8 reviews existing related works. Finally, Section 9 concludes the paper and provides future research directions.

2 MODEL OF EXECUTION

2.1 Task, platform and scheduler models

We consider a set $\tau \stackrel{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n recurring *independent* tasks to be executed on a single processor platform $\pi = [s_1, s_2, \ldots, s_m]$, where s_p (with $p \in [1, m]$) represent the $m \geq 1$ execution speed levels available on π. We assume throughout this paper that the smaller the index of a speed, the higher its value. In other words, this means that s_1 denotes the highest speed and s_m the slowest speed available on π. The tasks are scheduled by following any *non-preemptive* Fixed-Task-Priority (FTP) scheduler[1]. We assume that the smaller the index of a task, the higher its priority. Every $\tau_i \in \tau$ is modeled by a *constrained-deadline periodic* task characterized by a 4-tuple (O_i, C_i, D_i, T_i), where O_i is the offset; C_i is the worst-case execution time (WCET); $D_i \leq T_i$ is the relative deadline; and finally T_i is the *exact* inter-arrival time between two consecutive releases of task τ_i. We call each release of a task a "job", denoted by $\tau_{i,k}$, where $k = 1, \ldots, \infty$ is the job index. This means that job $\tau_{i,k}$: (1) is released at time $r_{i,k} \stackrel{\text{def}}{=} O_i + (k-1) \cdot T_i$; (2) has an execution requirement of at most C_i; and finally (3) must complete within $[r_{i,k}, d_{i,k}]$, where $d_{i,k} \stackrel{\text{def}}{=} r_{i,k} + D_i$. We recall that for FTP schedulers, every job generated from a task inherits the priority assigned to this task. We denote by "$\text{hp}(\tau_i)$" (resp., "$\text{lp}(\tau_i)$") the set of tasks with a higher (resp., a lower) priority than τ_i and by "$\text{hep}(\tau_i)$" (resp., "$\text{lep}(\tau_i)$") the set $\text{hp}(\tau_i) \cup \{\tau_i\}$ (resp., $\text{lp}(\tau_i) \cup \{\tau_i\}$). All the jobs generated from a task are executed at the same *constant* speed. This means that for a given reference speed, say $s > 0$ time units per seconds, it takes at most $\frac{C_i}{s}$ seconds to execute each job of τ_i. By following this rule, a low speed yields a long execution time, whereas a high speed yields a fast execution. Without any loss of generality, we neglect the overhead associated to both switching between speeds and gating the clock in this paper. As a matter of fact, these overheads can be included in the C_i for each task τ_i in a non-preemptive execution environment. For the schedulability analysis, we assume that each task always executes for its WCET. This means that the proposed analysis will be conservative, and will not be exact since even without considering the thermal aspect, the addressed problem is NP-hard in the strong sense. The task-to-speed mapping strategy can use the criticality level of the tasks. For example: "*the higher the criticality of a task, the higher the speed at which it must be executed*". Obviously, sophisticated strategies could be designed to address this specific point in order to control the incurred switching time and related overhead costs (from one speed to another and clock gating), but the actual task-to-speed mapping problem is left out of the scope of this work, as we assume that it is the responsibility of the system designer for a given application. Because speed selection has a substantial impact on thermal efficiency, our main objective in this paper is to design a methodology to mitigate its effects at run-time.

2.2 Thermal model

The adopted thermal model uses an RC circuit similar to the one described in [5], where: (*i*) R denotes a thermal resistance; (*ii*) C a thermal capacitance; (*iii*) T_A the ambient temperature[2]; (*iv*) $T(t)$ the processor temperature at time $t > 0$; and (*v*) $P(t)$ the heating phase at time $t > 0$. We assume that the processor may be in only one of the following two possible states at any time instant: (1) *active* (i.e., heating) during which tasks may be executing at a specific speed or (2) *inactive* (i.e., cooling) during which tasks are not allowed to execute (see blue curve in Figure 1). For sake of readability, only s_2 and s_3 are reported along the y-axis to represent the corresponding temperature $a \cdot s_p^{\alpha}/b$, with $p = 2, 3$, in all figures.

The heating phase can be modeled by the current $P(t) \stackrel{\text{def}}{=} P_D(t) + P_L(t)$ passing through the thermal resistance R. Here, $P_D(t) \stackrel{\text{def}}{=} \beta_0 \cdot s^{\alpha}$ represents the *dynamic current*; $P_L(t) \stackrel{\text{def}}{=} \beta_1 \cdot T(t) + \beta_2$ the *leakage current*; and α, β_0, β_1 and β_2 are processor specific constants. The derivative of the system temperature w.r.t. time can be calculated by solving the following linear differential equation.

[1] Popular FTP schedulers include Rate Monotonic and Deadline Monotonic.

[2] Here we assume the ambient temperature as constant.

$$T'(t) + \frac{T(t) - T_A}{R \cdot C} = \frac{P(t)}{C} \tag{1}$$

In Equation 1, by setting $T(t)$ to be $T(t) - \frac{R \cdot \beta_2 - T_A}{R \cdot \beta_1 - 1}$ to shift T_A to 0, we obtain the following classical differential equation.

$$T'(t) + b \cdot T(t) = a \tag{2}$$

where, parameters a and b are defined as in Equation 3.

$$a \stackrel{\text{def}}{=} \frac{\beta_0 \cdot s^\alpha}{C} \quad \text{and} \quad b \stackrel{\text{def}}{=} \frac{1}{R \cdot C} - \frac{\beta_1}{C} \tag{3}$$

Since the processor speed may vary from the execution of one job to another at run-time by assumption, it is important remark that parameter "a" is not constant over time, while parameter "b" is. As a matter of fact, $a = a_0 \cdot s^\alpha$, where $a_0 \stackrel{\text{def}}{=} \frac{\beta_0}{C}$ is constant. Throughout this paper, we consider $a_0 = 8$; $b \approx 0.228$ and $\alpha = 3$. These values are typical settings for a silicon chip [5, 34]. Hereafter, **we drop the index of constant "a_0" to avoid heavy notations** and to make the discussion clearer for the reader. We denote the temperature during the heating and cooling phases as $T_h(t)$ and $T_c(t)$.

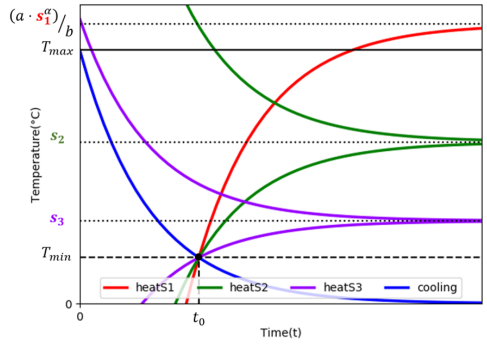


Figure 1: Typical thermal behavior over time w.r.t. speeds.

⋈ **Heating Model:** We assume that heating comes mainly from the processor activity and the heating produced by the other components has a negligible impact on the processor global thermal behavior. Hence, the solution to Equation 2, describing the heating function, is given by Equation 4.

$$T_{h_s}(t) = \frac{a \cdot s^\alpha}{b} + \left(T(t_0) - \frac{a \cdot s^\alpha}{b}\right) e^{-b \cdot (t - t_0)} \tag{4}$$

In Equation 4: (1) "s" is the reference execution speed, (2) "t_0" is the time required to cool-down the processor from $T_{\max}$ to $T_{\min}$; and (3) $T(t_0) = T_{\min}$ (see Figure 1). In this figure, the heating function is illustrated assuming three reference execution speeds, namely: (*i*) speed s_1 (see the "red" curve); (*ii*) speed s_2 (see the "green" curves); and (*iii*) speed s_3 (see the "purple" curves). In Case (*i*), we assume that the asymptote (here, $\frac{a \cdot s_1^\alpha}{b}$) exceeds $T_{\max}$, whereas this is not the case in Case (*ii*) and Case (*iii*). In the latter two cases, it is worth mentioning the change of dynamism in the curve representing the heating function over time. Indeed, while the heating function always evolves towards its asymptote (i.e., $\frac{a \cdot s_p^\alpha}{b}$ at execution speed s_p), it appears that the curve shapes outward, i.e., it is *concave*, if the current temperature falls below this value. If the current temperature is above the asymptotic value, then the curve shapes inward, i.e., it is *convex*.

⋈ **Cooling Model:** During this phase, we assume that no workload is allowed to be executed, so the processor remains inactive for certain period of time. This assumption is materialized by considering $s = 0$ and hence Equation 5 holds to describe the cooling function.

$$T_c(t) = T(t_0) \cdot e^{-b \cdot (t - t_0)} \tag{5}$$

This function is illustrated in Figure 1 (see the blue curve). From this figure and Equation 5, we have $T_{\max} = T_{\min} \cdot e^{b \cdot t_0}$, which implies that $t_0 = \frac{1}{b} \cdot \ln\left(\frac{T_{\max}}{T_{\min}}\right)$. A summary of key parameters is provided at the end of this paper (see Table 2).

3 CHALLENGE AND MAIN INTUITION BEHIND THE PROPOSED SOLUTION

In the traditional real-time scheduling theory, the attention of system designers is usually drawn to guaranteeing that all timing constraints are met. A good share of contributions cover other aspects like the processor temperature [6, 11, 18, 24, 28]. However, only a few falls in the scope of this work [1, 5, 25, 30]. In this section, we take the thermal dimension into account and provide the big-picture look at our proposed thermal-aware scheduling strategy for single-core processors.

From a thermal view point, it is worth noticing that tasks executed at a low speed dissipate less heat than those executed at a higher speed in the same time window (see Figure 1). In Figure 2, the processor operates at three different speeds s_1, s_2 and s_3 (with $s_1 \geq s_2 \geq s_3$) and a typical job execution sequence generated from the task sequence $\mathcal{T}_\star = \langle \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7 \rangle$ is illustrated under the classical inter-task DVFS scheme. As it is obvious from the plot, the platform temperature is always kept within the predefined thermal boundaries, $T_{\min}$ and $T_{\max}$. Here, tasks τ_1, τ_4 and τ_7 are executed at speed s_1 (see the "red" segments); tasks τ_3 and τ_5 are executed at speed s_2 (see the "green" segments); and finally tasks τ_2 and τ_6 are executed at speed s_3 (see the "purple" segments). It is important to remark the change of dynamism (*concave* vs. *convex*) in the execution of tasks τ_3 and τ_5 despite the fact that they are executed at the same speed. While the completion of τ_3 contributes to an increase in the processor temperature, it is the opposite for task τ_5. This change is due to: (1) the thermal point from which these tasks started their respective execution; and (2) the asymptotic value corresponding to the execution at speed s_2 (see Figure 1).

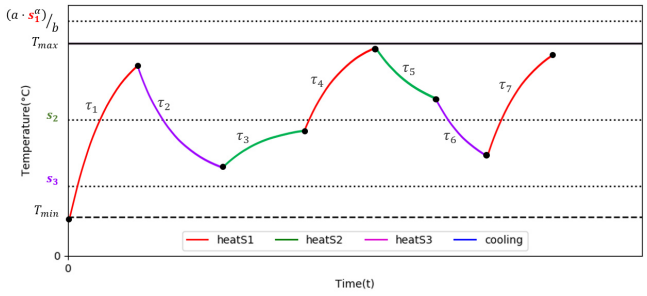


Figure 2: Typical job execution sequence under DVFS.

Now, considering the same task-set and attributes for each task, the picture may change drastically if the job execution sequence is different (see Figure 3).

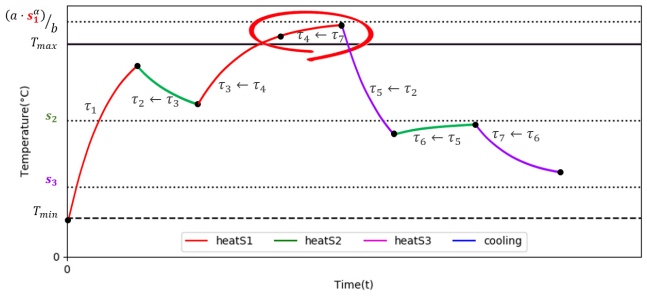


Figure 3: Influence of the tasks' execution sequence on the thermal behavior.

In Figure 3, $\mathcal{T}_{\star\star} = \langle \tau_1, \tau_3, \tau_4, \tau_7, \tau_2, \tau_5, \tau_6 \rangle$ is the assumed execution sequence. This is illustrated by $\tau_i \leftarrow \tau_j$, where task τ_j updates τ_i. Here, we observe a violation of the maximum thermal threshold $T_{\max}$, thus exposing the limitations of the classical inter-task DVFS scheduler. This thermal violation is due to the execution of tasks τ_4 and τ_7 in $\mathcal{T}_{\star\star}$. In order to get around this hurdle, actions must be taken prior to the execution of these tasks since the adopted scheduler is non-preemptive by assumption. This is precisely the strategy promoted in the design of NP-COIN. Specifically, we introduce a number of *cooling windows* denoted by $[cw_1; cw_2; cw_3; \ldots]$ during run-time *prior* to executing some specific workloads in order to keep the processor temperature within $T_{\min}$ and $T_{\max}$. These cooling windows also define a sequence of *execution windows* referred to as $[ew_1; ew_2; ew_3; \ldots]$, wherein the processor is continuously busy executing jobs, while meeting both the thermal and timing constraints (see Figure 4). A similar approach was recently adopted in the scope of multi-core platforms by using a slack distribution policy [41], but the proposed solution was targeting only peak temperatures, unfortunately. Our solution captures both the transient and permanent temperatures at run-time.

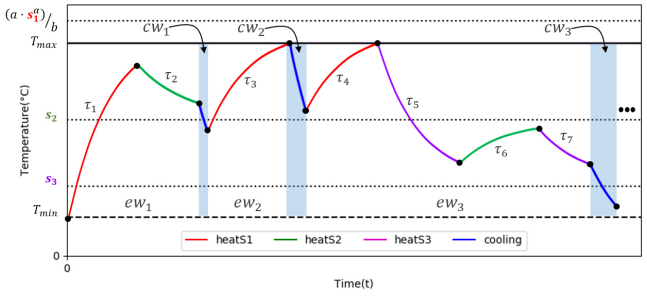


Figure 4: Typical job execution sequence under NP-COIN.

Rodríguez and Yomsi [25] showed that it is beneficial from a thermal viewpoint to trigger the cooling mechanism for several short intervals than a long cooling interval at once. In this paper, we acknowledge this result and build upon it. However, we adopt such a strategy only when it is mandatory to do so in order not to miss any thermal constraint and to keep the processor temperature within $T_{\min}$ and $T_{\max}$. As a matter of fact, each inserted cooling window causes a timing penalty in the completion time of the pending tasks. Put all together, the cumulative timing penalty may render the entire system not schedulable.

4 PREREQUISITES AND PRELIMINARY RESULTS

This section introduces a number of notations, basic properties and provides key concepts and/or preliminary results for a smooth understanding of the rest of this paper. Specifically, we recall the concepts of *level-i busy period*; *longest admissible execution time*; *shifted heating function* and *shifted cooling function* for a task τ_i, etc. They will constitute the backbone for the specification of our NP-COIN scheduler.

Definition 4.1 (level-i busy period [19]). A level-i busy period is a time interval $[\sigma, \mu]$ within which only jobs belonging to $\text{hep}(i)$ (except the first job, which is generated from task $\tau_j \in \text{lp}(\tau_i)$ with the longest WCET) are executed throughout $[\sigma, \mu]$, but no jobs belonging to $\text{hep}(i)$ are executed in $[\sigma - \epsilon, \sigma)$ or $(\mu, \mu + \epsilon]$ for any arbitrary small $\epsilon > 0$.

From Definition 4.1, a level-i busy period may consist of several consecutive *execution* and *cooling* windows and, in such a time interval, there is no time instant at which a pending ready job is not executed. In other words, the response time of every job $\tau_{i,k}$ (with $i \in [1, n]$ and $k \geq 1$), i.e., the time elapsed between its release and completion times, is part of a level-i busy period. As such, we will need to identify the phasing of offsets $O_1, O_2, \ldots, O_i$ that creates the level-i busy period wherein we find the longest response time for each task τ_i. While this is a rather simple exercise when the only considered metric is time, as in the classical non-preemptive scheduling theory, the clear-blue sky becomes cloudy-gray when thermal constraints are also taken into account. This is the case in this paper and thus more care is required. Since tasks may be executed at different speeds, the blocking term B_i of any τ_i admissible for execution on π can be stated as in Lemma 4.2.

Lemma 4.2 (Blocking term B_i). *Assuming that task $\tau_j \in \text{lp}(\tau_i)$ in Definition 4.1 is executed at speed $s_{p_j} > 0$ (with $p_j \in [1, m]$), then the blocking term B_i for the execution of task τ_i is defined by Equation 6.*

$$B_i = \begin{cases} \max\limits_{\tau_j \in \text{lp}(\tau_i)} \left\{ \dfrac{C_j}{s_{p_j}} \right\}; & \text{if } i \in [1, n-1] \\ 0; & \text{otherwise} \end{cases} \tag{6}$$

Considering that *all* tasks are executed at the same reference speed $s_{\text{ref}} = 1$ and the heating function $T_{h_{s_{\text{ref}}}}(t)$ crosses the maximum thermal threshold $T_{\max}$, Rodríguez and Yomsi (see [25], Lemma 1) showed that the longest admissible execution time (ΔC_{ref}) for any task on π is given by Equation 7.

$$\Delta C_{\text{ref}} = -\frac{1}{b} \cdot \ln\left(\frac{T_{\max} - a/b}{T_{\min} - a/b} \right) \tag{7}$$

Lemma 4.3 (Updated ΔC). *Assuming that the heating function $T_{h_s}(t)$ at the maximum speed available on π (here, s_1) crosses the*

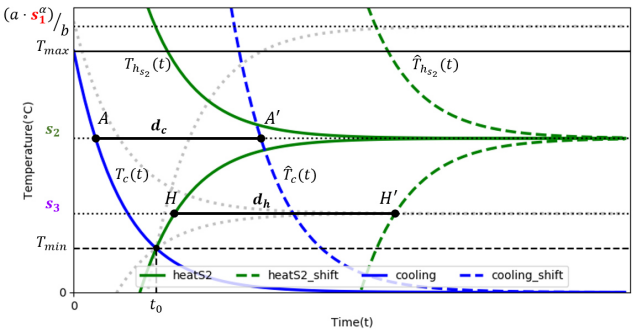


Figure 5: Shifted heating and cooling functions.

maximum thermal threshold $T_{\max}$, then the longest admissible execution time for any task (i.e., the longest execution time of any task that would not violate any thermal constraint) is given by Equation 8.

$$\Delta C \stackrel{\text{def}}{=} -\frac{s_1}{b} \cdot \ln\left(\frac{T_{\max} - a \cdot s_1^\alpha / b}{T_{\min} - a \cdot s_1^\alpha / b}\right) \tag{8}$$

Proof. The proof is similar to the one in [25]. We will not repeat it here due to space limitation. It is sufficient to note that ΔC is normalized by using an execution at the maximum speed. □

Definition 4.4 (Speed categorization s_{high} and s_{low}). Let $T(s_p)$ be the maximum achievable temperature when processor π executes from T_{min} at speed level p (with $p \in [1, m]$). Then, we categorize p as *high* if $T(s_p) \geq T_{max}$; otherwise, we categorize p as *low*. We denote by s_{high} (resp., s_{low}) the set of all high (resp., low) speeds.

To illustrate Definition 4.4, we consider the thermal behaviors in Figure 1, then $s_{\text{high}} = \{s_1\}$ and $s_{\text{low}} = \{s_2, s_3\}$. In the rest of this paper, we assume that τ_i (with $i \in [1, n]$) is always executed at speed s_{p_i} or simply "s_i" to keep the same index as the task.

Definition 4.5 (Shifted heating function ($\hat{T}_{h_s}(t)$) and Shifted cooling function ($\hat{T}_c(t)$)). Let $T_{h_s}(t)$ and $T_c(t)$ be the heating function and the cooling function as defined in Equations 4 and 5, respectively. Then, the *shifted* heating function $\hat{T}_{h_s}(t)$ at distance d_h from $T_{h_s}(t)$ and the *shifted* cooling function $\hat{T}_c(t)$ at distance d_c from $T_c(t)$ are derived as follows.

$$\hat{T}_{h_s}(t) \stackrel{\text{def}}{=} \frac{a \cdot s^\alpha}{b} + \left(T_{\min} - \frac{a \cdot s^\alpha}{b}\right) e^{-b \cdot (t - t_0 + d_h)} \tag{9}$$

and

$$\hat{T}_c(t) \stackrel{\text{def}}{=} T_{\min} \cdot e^{-b \cdot (t - t_0 + d_c)} \tag{10}$$

Assuming the thermal behaviors in Figure 1, the functions $\hat{T}_{h_{s_2}}(t)$ at distance d_h and $\hat{T}_c(t)$ at distance d_c are illustrated in Figure 5. Points H and A are shifted to H' and A'.

Lemma 4.6 (Heating function of a sequence $W_{h_{s_\ell}}(t)$). *Let $\mathcal{T}_\ell = \langle \tau_1, \tau_2, \ldots, \tau_\ell \rangle$ (with $\ell \geq 1$) denote an already re-indexed job execution sequence from time instant t_0 on platform π (see for example Figure 2). Then, the heating function governing the execution of task τ_ℓ is given by Equation 11.*

$$W_{h_{s_\ell}}(t) \stackrel{\text{def}}{=} \frac{a \cdot s_\ell^\alpha}{b} + e^{-b \cdot (t - t_0)} \left\{T_{\min} - \frac{a}{b}\left(s_1^\alpha - \Lambda_{\ell-1}\right)\right\} \tag{11}$$

where, parameter Λ_φ (with $\varphi \geq 0$) is given by Equation 12.

$$\Lambda_\varphi \stackrel{\text{def}}{=} \begin{cases} \sum_{p=1}^{\varphi} \left[\left(s_p^\alpha - s_{p+1}^\alpha\right) e^{b \cdot \left(\sum_{\gamma=1}^{p} \frac{C_\gamma}{s_\gamma}\right)}\right]; & \text{if } \varphi \geq 1 \\ 0; & \text{Otherwise} \end{cases} \tag{12}$$

Proof. (By induction on ℓ)

Base step $\ell = 1$: The job execution sequence is $\mathcal{T}_1 = \langle \tau_1 \rangle$ and we have:

$$\begin{aligned} W_{h_{s_1}}(t) &= T_{h_{s_1}}(t); \text{ i.e., the heating function for } \tau_1 \\ &= \frac{a \cdot s_1^\alpha}{b} + e^{-b \cdot (t - t_0)} \left(T_{\min} - \frac{a}{b} \cdot s_1^\alpha\right) \\ &= \frac{a \cdot s_1^\alpha}{b} + e^{-b \cdot (t - t_0)} \left(T_{\min} - \frac{a}{b} \cdot (s_1^\alpha - \Lambda_0)\right) \end{aligned}$$

So the lemma holds when $\ell = 1$.

Inductive hypothesis: Suppose the lemma holds for all ℓ up to some $q \geq 1$.

Inductive step: Let $\ell = q + 1$. Then $\mathcal{T}_{q+1} = \langle \tau_1, \ldots, \tau_q, \tau_{q+1} \rangle$ and we have:

$$\begin{aligned} W_{h_{s_{q+1}}}(t) &= \hat{T}_{h_{s_{q+1}}}(t); \text{ by Definition 4.5, where } d_h \text{ is unknown.} \\ &= \frac{a \cdot s_{q+1}^\alpha}{b} + \left(T_{\min} - \frac{a \cdot s_{q+1}^\alpha}{b}\right) e^{-b \cdot (t - t_0 + d_h)} \\ &= \frac{a \cdot s_{q+1}^\alpha}{b} + e^{-b \cdot (t - t_0)} \left(T_{\min} - \frac{a \cdot s_{q+1}^\alpha}{b}\right) e^{-b \cdot d_h} \end{aligned} \tag{13}$$

Since the tasks are executed in non-preemptively and τ_1 is executed from t_0, then the completion time of $\mathcal{T}_q = \langle \tau_1, \tau_2, \ldots, \tau_q \rangle$ is given by $\zeta \stackrel{\text{def}}{=} t_0 + \sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}$. By using the previous equality (LHS of Equation 14) and the induction hypothesis (RHS of Equation 14) upon the completion of τ_q (at time instant ζ) we have:

$$\begin{aligned} \frac{a \cdot s_{q+1}^\alpha}{b} + e^{-b \cdot (\zeta - t_0)} \left(T_{\min} - \frac{a \cdot s_{q+1}^\alpha}{b}\right) e^{-b \cdot d_h} &= \frac{a \cdot s_q^\alpha}{b} + e^{-b \cdot (\zeta - t_0)} \left\{T_{\min} - \frac{a}{b}\left(s_1^\alpha - \Lambda_{q-1}\right)\right\} \qquad (14) \\ \text{i.e.,} \quad \frac{a \cdot s_{q+1}^\alpha}{b} + e^{-b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} \left(T_{\min} - \frac{a \cdot s_{q+1}^\alpha}{b}\right) e^{-b \cdot d_h} &= \frac{a \cdot s_q^\alpha}{b} + e^{-b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} \left\{T_{\min} - \frac{a}{b}\left(s_1^\alpha - \Lambda_{q-1}\right)\right\} \\ \text{i.e.,} \quad e^{-b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} \left(T_{\min} - \frac{a \cdot s_{q+1}^\alpha}{b}\right) e^{-b \cdot d_h} &= \frac{a}{b}\left(s_q^\alpha - s_{q+1}^\alpha\right) + e^{-b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} \left\{T_{\min} - \frac{a}{b} \cdot \left(s_1^\alpha - \Lambda_{q-1}\right)\right\} \\ \text{i.e.,} \quad \left(T_{\min} - \frac{a \cdot s_{q+1}^\alpha}{b}\right) e^{-b \cdot d_h} &= \frac{\frac{a}{b}\left(s_q^\alpha - s_{q+1}^\alpha\right) + e^{-b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} \left\{T_{\min} - \frac{a}{b}\left(s_1^\alpha - \Lambda_{q-1}\right)\right\}}{e^{-b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)}} \end{aligned}$$

$$\text{i.e.,} \quad \left(T_{\min} - \frac{a \cdot s_{q+1}^{\alpha}}{b}\right) e^{-b \cdot d_h} = \frac{a}{b}\left(s_q^{\alpha} - s_{q+1}^{\alpha}\right) e^{b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} + \left\{T_{\min} - \frac{a}{b}\left(s_1^{\alpha} - \Lambda_{q-1}\right)\right\}$$

By substituting this last equality in Equation 13, we obtain:

$$\begin{aligned} W_{h s_{q+1}}(t) &= \frac{a \cdot s_{q+1}^{\alpha}}{b} + e^{-b \cdot (t-t_0)} \left\{ \frac{a}{b}\left(s_q^{\alpha} - s_{q+1}^{\alpha}\right) e^{b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} + \left[T_{\min} - \frac{a}{b}\left(s_1^{\alpha} - \Lambda_{q-1}\right)\right] \right\} \\ &= \frac{a \cdot s_{q+1}^{\alpha}}{b} + e^{-b \cdot (t-t_0)} \left\{ T_{\min} - \frac{a}{b}\left[s_1^{\alpha} - \left(\Lambda_{q-1} + \left(s_q^{\alpha} - s_{q+1}^{\alpha}\right) e^{b \cdot \left(\sum_{\gamma=1}^{q} \frac{C_\gamma}{s_\gamma}\right)} \right) \right] \right\} \\ &= \frac{a \cdot s_{q+1}^{\alpha}}{b} + e^{-b \cdot (t-t_0)} \left\{ T_{\min} - \frac{a}{b}\left(s_1^{\alpha} - \Lambda_q\right) \right\} \end{aligned}$$

and the lemma follows. □

Upon the completion of a job execution sequence, say $\mathcal{T}_\ell = \langle \tau_1, \ldots, \tau_\ell \rangle$, if the ready queue is not empty, then it could be the case that the execution of the next task, say $\tau_{\ell+1}$, requires the processor to cool-down for some time period, say $x_{\ell+1} > 0$, in order not to miss its thermal constraint (see for example Figure 4). When this is the case, the insertion of the cooling windows is mandatory and it defines successive execution windows. The computation of $x_{\ell+1}$ depends on the processor characteristics and (1) the thermal point reached after the *previous* execution window (on the left) *before* $\tau_{\ell+1}$ is executed; and (2) the length of the execution requirement of $\tau_{\ell+1}$ (on the right) in order not to miss its thermal constraint. To this end, we consider a "*fake task*", say τ_ℓ^{fake}, with an execution requirement C_ℓ^{fake} that would allow the processor to reach the same thermal point as $\mathcal{T}_\ell = \langle \tau_1, \ldots, \tau_\ell \rangle$ when executed in isolation, i.e., from $T_{\min}$ without interference from any other task (see the red-dashes in Figure 6). We assume that τ_ℓ^{fake} executes at $s_\ell^{\text{fake}} \in s_{\text{high}}$.

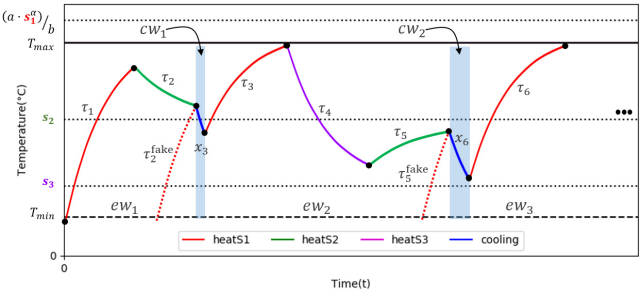


Figure 6: Cooling phase computation

By using the same intuitive idea as [25], Corollary 2, we compute $x_{\ell+1}$ in such a manner that the temperature of the processor is exactly $T_{\max}$ upon the completion of $\tau_{\ell+1}$. This is meant to reduce the length of the cooling period as much as possible. In addition, this approach induces the minimum timing penalty on the responsiveness of $\tau_{\ell+1}$. Formally, $x_{\ell+1}$ is defined as in Equation 15.

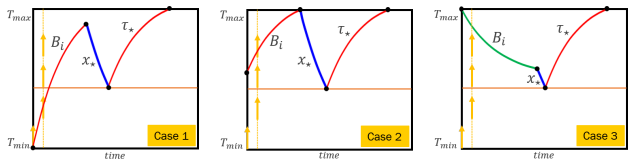
$$x_{\ell+1} \stackrel{\text{def}}{=} \frac{1}{b} \cdot \ln \left[\frac{T_{\min} + \frac{a \cdot s_\ell^{\text{fake}}}{b} \cdot \left(e^{b \cdot \frac{C_\ell^{\text{fake}}}{s_\ell^{\text{fake}}}} - 1\right)}{T_{\max} + \frac{a \cdot s_{\ell+1}}{b} \cdot \left(e^{-b \cdot \frac{C_{\ell+1}}{s_{\ell+1}}} - 1\right)} \right] - \left(\frac{C_\ell^{\text{fake}}}{s_\ell^{\text{fake}}} + \frac{C_{\ell+1}}{s_{\ell+1}} \right) \quad (15)$$

5 KEY PROPERTIES AND OBSERVATIONS

As mentioned in Section 2, the schedulability analysis is conducted by assuming that each task always executes for its WCET.

Lemma 5.1 (Longest thermal-aware level-i busy window). *The scenario that generates the longest thermal-aware level-i busy window for any task τ_i occurs when the following three conditions are satisfied:*

1. *Tasks $\tau_1, \tau_2, \ldots, \tau_i$ release a job at the same time instant (say, at time 0);*
2. *Task $\tau_k \in \text{lp}(\tau_i)$ with the maximum $\frac{C_k}{s_k}$ releases a job at an arbitrary small $\epsilon > 0$ time units before τ_i (this factor helps in computing the traditional blocking term B_i). If several tasks in $\tau_k \in \text{lp}(\tau_i)$ lead to the same ratio maximum $\frac{C_k}{s_k}$, then the tie is broken by selecting the task which terminates at the highest thermal level between $T_{\min}$ and $T_{\max}$.*
3. *The initial temperature $T_{\text{init}} \in [T_{\min}, T_{\max}]$ of the processor at time 0 is at a level that requires the insertion of a cooling window upon the completion of B_i for the execution of the next task in order not to violate the thermal constraint. Specifically, these situations are captured in Figure 7.*



(a) $T_{\text{init}} = T_{\min}$. (b) $T_{\text{init}} \in]T_{\min}, T_{\max}[$. (c) $T_{\text{init}} = T_{\max}$.

Figure 7: Longest level-i busy window.

The main intuition behind Lemma 5.1 is as follows. The worst-case thermal-aware scenario for every task τ_i occurs whenever it is requested simultaneously with requests of all higher priority tasks and the thermal level is as close as possible to $T_{\max}$ upon the execution of the blocking term B_i.

Lemma 5.2 (Worst-case thermal-aware response time R_i^{TA}). *The worst-case thermal-aware response time R_i^{TA} for any task $\tau_i \in \tau$ occurs in the longest level-i busy window as specified in Lemma 5.1.*

Lemma 5.2 is inspired directly from the well-established result in the classical real-time scheduling theory for non-preemptive tasks. Consequently, we will skip the proof in this paper. The worst-case thermal-aware response time R_i^{TA} for any task $\tau_i \in \tau$ will thus be computed by scrutinizing all its jobs in the longest thermal-aware level-i busy window and by extracting the maximum thermal-aware response times. If $R_i^{\text{TA}} \leq D_i$, then τ_i is schedulable. Otherwise, τ_i is not schedulable and so is the entire system. Note that R_i^{TA} is always greater than or equal to the traditional worst-case response time R_i of τ_i, when thermal constraints are ignored.

6 NP-COIN SCHEDULER

Our thermal-aware scheduler NP-COIN behaves like the classical DVFS unless the predefined thermal boundaries are about to be violated. In a nutshell, it is a proactive scheduler that computes the length of each cooling phase, say $x_\star$, prior to the execution of the corresponding task, say $\tau_\star$, during the next heating phase (see Section 4, Equation 15). However, what happens during the cooling period may be a "game changer" for the next task to be executed since the processor is inactive in this window. Indeed, another task, say $\tau_{\star\star}$, may release a new job, which requires an update of the ready queue and changes the horizon of the schedule (see Figure 8).

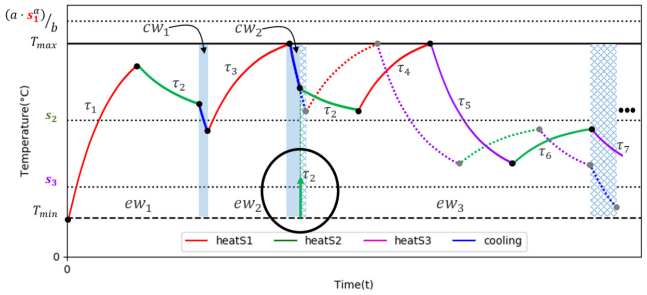


Figure 8: Releases during the cooling under NP-COIN.

In this figure, τ_2 releases a new job during the cooling window and the original schedule (see the dash-lines) is modified from that point onward (see the solid lines). Consequently, we must distinguish the three cases (see Figure 9) to compute the actual length, say x, of the cooling phase.

6.1 On the computation of "x"

▹ **Case 1 (Figure 9(a)): No task or an LP-task (here $\tau_{\star\star}$) releases a job.** Here, Equation 16 returns the value of "x".

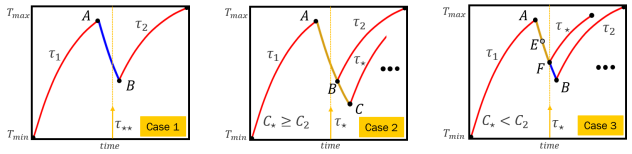
$$x \stackrel{\text{def}}{=} \frac{1}{b} \cdot \ln\left[\frac{T_{\min} + \frac{a \cdot s_1^\alpha}{b} \cdot (e^{b \cdot \frac{C_1}{s_1}} - 1)}{T_{\max} + \frac{a \cdot s_2^\alpha}{b} \cdot (e^{-b \cdot \frac{C_2}{s_2}} - 1)}\right] - \left(\frac{C_1}{s_1} + \frac{C_2}{s_2}\right) \tag{16}$$

▹ **Case 2 (Figure 9(b)): An HP-task (here $\tau_\star$) releases a job, say at time $r_\star$, and the cooling length is sufficient.** Here, $r_\star$ is reached after the computation of $x_\star$ (e.g., Point C) and $\tau_\star$ becomes active. Equation 17 returns the value of "x".

$$x \stackrel{\text{def}}{=} \frac{1}{b} \cdot \ln\left[\frac{T_{\min} + \frac{a \cdot s_1^\alpha}{b} \cdot (e^{b \cdot \frac{C_1}{s_1}} - 1)}{T_{\max} + \frac{a \cdot s_\star^\alpha}{b} \cdot (e^{-b \cdot \frac{C_\star}{s_\star}} - 1)}\right] - \left(\frac{C_1}{s_1} + \frac{C_\star}{s_\star}\right) \tag{17}$$

▹ **Case 3 (Figure 9(c)): An HP-task (here $\tau_\star$) releases a job, say at time $r_\star$, but the cooling length is not sufficient.** Here, $r_\star$ is *not* reached after the computation of $x_\star$ (e.g., Point E) and $\tau_\star$ remains *inactive*. We enforce an additional cooling period (from Point E to F) in order to avoid a priority inversion. Equation 18 returns the value of "x", where t_{crt} is the current absolute time instant in the schedule.

$$x \stackrel{\text{def}}{=} r_\star - t_{\text{crt}} \tag{18}$$



(a) No task or an LP-task is released during the cooling period.

(b) An HP-task is released and the cooling length is sufficient.

(c) An HP-task is released, but the cooling length is unsufficient.

Figure 9: Support for the computation of the cooling lengths.

7 EXPERIMENTAL RESULTS

We consider a single-core from an ARM-Cortex-A53 processor in Raspberry Pi 3 and set $T_{\max} = 55°C$ and $T_{\min} = 10°C$ to picture the conditions at high altitudes. The temperature of the standard atmosphere is smaller than or equal to $9.1°C$ starting from 3.000 feet upwards in the avionics domain (see [10], Chapter 4). Then, we consider: $a_0 = 8$, $b = 0.228$ and $\alpha = 3$ [34]. We assume three operational speeds $[0.8; 1.0; 1.2]$ (in GHz). The utilization for task τ_i is given by $u_i \stackrel{\text{def}}{=} C_i/(T_i \cdot s_i)$ and the tasks are scheduled by following the traditional *Deadline Monotonic* scheduler. From these inputs, $\Delta C = 11.5588$ and $t_0 = 7.4769$. In order to prove the usability of our algorithm, we apply the proposed methodology to the following two contexts:

(a) a real-world MCC use-case from the avionics domain;
(b) a high number of generated synthetic test cases.

▹ **Evaluation metrics:** We compare the performance of three schedulers: (1) DVFS, where time is the only evaluation metric and thermal constraints are ignored; (2) *thermal*-DVFS, where a task-set is not schedulable as soon as any task violates a thermal boundary; and finally (3) NP-COIN, where both thermal and timing constraints are taken into account. By following these definitions, it worth noticing that the timing and thermal behaviors of a task-set are similar under DVFS and *thermal*-DVFS. However, a task-set can be deemed schedulable under DVFS (this is the case when all the timing requirements are met), but unschedulable under *thermal*-DVFS (this is the case as soon as any thermal boundary is violated). Note that, the reverse is not true.

Despite the fact that DVFS and *thermal*-DVFS are intrinsically naive schedulers per se, it is worth noticing that they define the envelope of the solution space when both timing and thermal constraints are taken into account. Therefore, the closer the behavior of our NP-COIN solution is to DVFS, the more efficient it is. On the contrary, the closer our NP-COIN solution is to *thermal*-DVFS the poorer is its design.

7.1 Application to a real-world MCC use-case.

The adopted real-world use-case is characterized by a set of sensors and actuators (e.g., weapons, mission control devices, communication devices, etc.) interconnected on a data bus (e.g., MIL-STD-1553B). The timing requirements are written by personnel from IBM's Federal Sector Division, the Naval Weapons Center, and the Software Engineering Institute and are replicated here for sake of completeness (see Table 1). A detailed description of the use-case can be

found in [22, 27]. In a nutshell, the software integrates different subsystems to ensure that the aircraft is able to complete a mission and these subsystems are given with the following interpretation.

- *Display*: updates the screen information.
- *Radar Warning Receiver*: provides threat information.
- *Radar Control*: returns target position.
- *Navigation*: computes aircraft position, attitude, and rates.
- *Tracking*: updates target information.
- *Weapon Control*: updates weapon ballistics.
- *Built-in-Test*: determines equipment status.
- *Data Bus*: performs communication between MCC and devices external to the MCC.

From this description, we perform the task-to-speed mapping based on educated guesses to have all the parameters for the experiment and we assume the following classification without any loss of generality (see Table 1).

(1) The "Radar Warning Receiver (RWR)", "Radar Control (Radar)" and "Navigation (NAV)" are high-criticality tasks, which must be executed at speed $s_1 = 1.2$;
(2) The "Display" and "Tracking" are medium-criticality tasks, which must be executed at speed $s_2 = 1.0$;
(3) Finally, the remaining systems (here "Weapon", "Built-in-Test" (BIT) and "Data Bus") are low-criticality tasks, which must be executed at speed $s_3 = 0.8$.

Note that other classifications are possible and would not have any impact on the applied methodology by any mean.

Table 1: MCC timing requirements

System	Subsystem	C_i	$D_i = T_i$	s_i	$u_i(\%)$
Display	Status Update	3	200	1.0	1.500
	Keyset	1	200	1.0	0.500
	Hook Update	2	80	1.0	2.500
	Graphic Display	9	80	1.0	11.250
	Stores Update	1	200	1.0	0.500
RWR	Contact Mgmt.	5	25	1.2	16.666
Radar	Target Update	5	50	1.2	8.333
	Tracking Filter	2	25	1.2	6.666
NAV	Nav Update	8	59	1.2	11.300
	Steering Cmds.	3	200	1.2	1.250
	Nav Status	1	1000	1.2	0.080
Tracking	Target Update	5	100	1.0	5.000
Weapon	Weapon Protocol	1	200	0.8	0.625
	Weapon Release	3	200	0.8	1.875
	Weapon Aim	3	50	0.8	7.500
BIT	Equ. Status Update	1	1000	0.8	0.125
Data Bus	Poll Bus Devices	1	40	0.8	3.125
System utilization					**78.795**

Figure 10 illustrates the execution sequence corresponding to the lowest priority task (here, BIT) by following the three schedulers – for DVFS and *thermal*-DVFS (see the dash execution sequence) and for NP-COIN (see the plain execution sequence) – and assuming the released scenario described in Lemma 5.1, i.e., the scenario leading to the longest busy period.

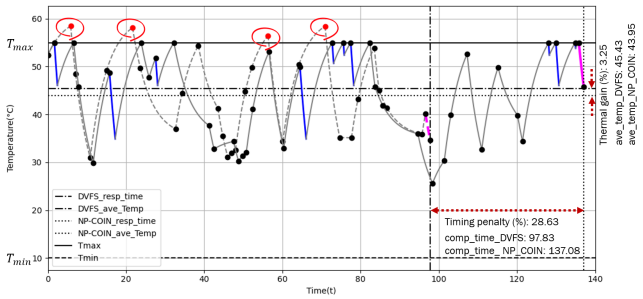


Figure 10: MCC (DVFS vs. NP-COIN) execution.

▹ **Interpretation of the results:** In Figure 10, we observe that the system is "schedulable" by following both NP-COIN and DVFS; whereas it is "not schedulable" under *Thermal*-DVFS, unfortunately. In the latter case, the thermal threshold $T_{\max}$ has been violated four times (see the red dots) during the execution of the tasks. We recall that DVFS follows the exact same behavior as *Thermal*-DVFS, but ignores all thermal constraints. The NP-COIN scheduler, in contrast, allows us to circumvent this hurdle by introducing a number of cooling periods (see the blue curves) in the resulting schedule before the execution of some workloads. Nonetheless, this procedure incurs some timing penalty in the responsiveness of the task under analysis. The completion time of the BIT task under DVFS is 97.83 time units, whereas it is 137.08 time units under NP-COIN. This means a timing penalty of 28.63%. From a thermal viewpoint, the average temperature under DVFS (i.e., the surface below the curve defined by the execution sequence until the execution of the BIT task) is $45.43^{\circ}C$; whereas it is $43.95^{\circ}C$ under NP-COIN. This means an average temperature drop of 3.25%, which corroborates the intended behavior in the design of this scheduler.

7.2 Application to synthetic test cases.

In this context, we generate 20,000 synthetic periodic constrained-deadline task-sets, uniformly distributed (1000 per system utilization) from 0.1 to 1, with a step of 0.05. We generate each task parameters as follows: (1) the execution times C_i are uniformly distributed within $[\Delta C/2; \Delta C]$; (2) the periods $T_i \in [30; 900]$ are generated by using the hyper-period limitation technique proposed in [12, 23]; and (3) the deadlines D_i are randomly generated within $[0, 8 \cdot T_i; T_i]$. The task-to-speed mapping is assumed to be random. We recall that the actual task-to-speed mapping problem is left out of the scope of this work, as it is assumed to be the responsibility of the system designer. It is worth mentioning that while some systems can continue to be operational when speed and voltage is changing[3] [13, 26], other systems stop during steep changes. Figure 11 illustrates the schedulability ratio for the three schedulers.

▹ **Interpretation of the results:** In Figure 11, we observe that *Thermal*-DVFS performs poorly even at low system utilization (at most 54.7% of the generated task-sets are schedulable) and the schedulability degrades as the system utilization increases. Conversely, NP-COIN performs very well in general. Its efficiency is illustrated by its close-knit relationship with DVFS. Until 80% of

[3]The frequency continues to vary during the transition period.

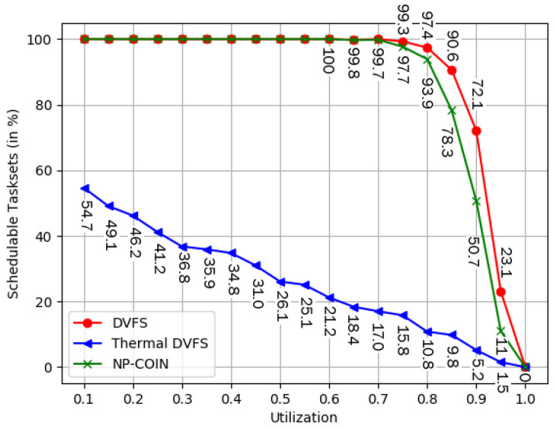


Figure 11: Schedulability ratio.

system utilization, at least 93.9% of the generated task-sets are schedulable vs. 97.4% for DVFS. This means a performance loss of only 3.5% against a scheduler that ignores all thermal constraints. This loss increases to 13.5% at 85%, to 29.6% at 90% and to 52.3% at 95% of system utilization. This trend is explained by the fact that there are lesser and lesser idle times left in the schedule to trigger any cooling. On another front, it is worth mentioning the huge gap between *Thermal*-DVFS and NP-COIN. This represents the task-sets ratio dragged from being not schedulable to schedulable by following the NP-COIN strategy. This gap reaches 83.1% at 80% of system utilization.

From a timing perspective, Figure 12 compares the (*Minimum - Average - Maximum*) worst-case response time for the lowest priority task under DVFS and NP-COIN. We notice a maximum deviation of (9.22% - 10.02% - 36.88%) in terms of timing penalty of NP-COIN against DVFS; whereas the minimum deviation is consistently kept at zero.

From a thermal standpoint, Figure 13 compares the (*Minimum - Average - Maximum*) temperature deviation for the lowest priority task under DVFS and NP-COIN. Here, we notice a maximum deviation of ($0.00°C$ - $1.23°C$ - $11.24°C$) in terms of thermal gain for NP-COIN over DVFS. We recall that the objective of NP-COIN is not to reduce the average temperature, but to guarantee that both the timing and thermal requirements are met for all tasks.

8 STATE OF THE ART

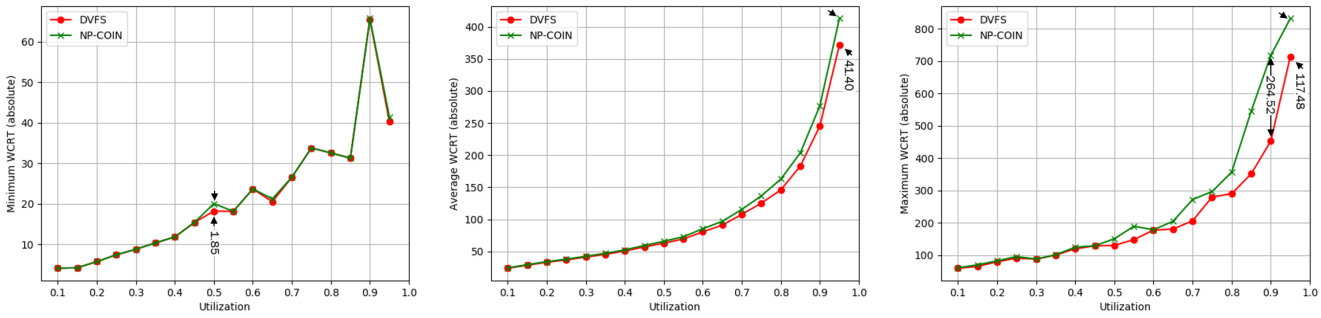
This section gives a brief overview on the existing works dealing with thermal-aware scheduling techniques upon single processor platforms. The list of presented contributions is by no means exhaustive, but is representative. Our thermal-aware scheduler (NP-COIN) intends to merge the most attractive properties of *clock-gating* and *DVFS-based* schedulers.

On clock-gating based techniques: Thermal-aware real-time systems have actively been addressed by the research community. Nonetheless, only a handful of contributions addresses clock-gating based schedulers. Along these lines, Chandarli et al. [5] adapted energy harvesting techniques [1] to thermal-aware ones and proposed worst-case response-time upper bounds for fully preemptive priority-ordered sporadic and independent tasks. However, they did so for classical non-concrete real-time task-sets. Ahmed et al. [2] proposed an alternative solution to the same problem. They established a necessary and sufficient condition for the thermal feasibility of periodic task-sets. Recently, Rodríguez and Yomsi [25] developed two schedulers that maintain the processor temperature within two thermal thresholds for non-preemptive tasks: (*i*) a reactive scheduler that cools-down the processor as much as possible in order to avoid high temperatures; and (*ii*) a proactive scheduler that cools-down the processor just for the needed amount to execute the pending workload. However, they did so only for non-DVFS-enabled platforms.

On DVFS-based techniques: Most of the contributions in this direction in the literature are reactive and rely on scaling-down the processor speed (through a combination of voltage and frequency scaling) to reduce its power consumption, and thereby its temperature [7, 8, 29–31, 37, 38]. Here, actions are taken only when temperature gets above a certain threshold and/or below through requests issued by task/scheduler. By using such an approach, Chantem et al. [7] determine the speed that maximizes the workload completed under a maximum temperature constraint. They propose an optimal DVFS control policy with two speed levels, but with non-negligible transition overheads. Chen et al. [8] explored thermal-constrained speed scheduling, but they did so for a set of frame-based real-time tasks, i.e., with the same period. They developed a proactive speed scheduling by utilizing dynamic voltage/speed scaling (DVS). Quan et al. [29] derived thermal feasibility checks and formulated constructive speed scheduling algorithms for periodic tasks under thermal constraints. Shaik and Baskiyar [31] propose a proactive software-based thermal aware scheduler, referred to as Simple Time Derivative (STD), which uses the time derivative of the core temperature as a predictor to lower its temperature and its temperature fluctuations. The approach requires to identify the so-called "hot" tasks [17, 21], which will be put to sleep for a short duration, if the time derivative goes above an empirically defined threshold. However, the metrics used in this procedure may not be accurate and can be questionable, unfortunately. Wang et al. [37] placed an emphasis on processors with discrete speed levels. Here, the processor runs at the highest speed until the threshold temperature is reached and then the equilibrium speed is used to keep the temperature just below its constraint. Unfortunately, the equilibrium speed might not always be available.

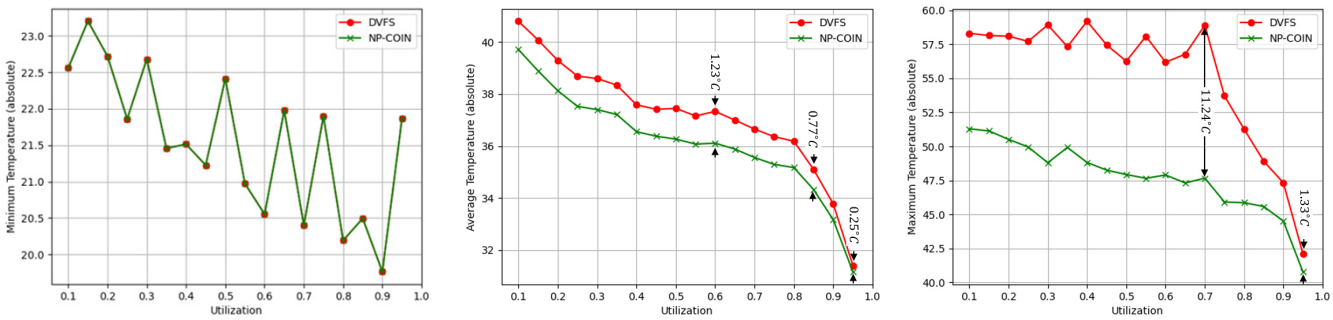
9 CONCLUSION AND FUTURE WORK

This paper considered the thermal-aware schedulability analysis of non-preemptive real-time tasks on a single processor platform. We captured both the thermal and timing behaviors of the system in the same framework by proposing a novel proactive scheduler, called NP-COIN, together with its associated schedulability analysis. Our solution captured not only the peak temperatures but also, the transient temperatures at run-time. We validated the run-time behavior of our solution through a real-world Mission Control Computer use-case from the avionics domain as well as through intensive simulations by using the typical thermal specifications of a single-core from an ARM-Cortex-A53 processor.



(a) Minimum. (b) Average. (c) Maximum.

Figure 12: [Minimum - Average - Maximum] Worst-case response time of the lowest priority task.



(a) Minimum. (b) Average. (c) Maximum.

Figure 13: [Minimum - Average - Maximum] Temperature of the lowest priority task.

Table 2: Summary of key parameters

Parameter	Description
τ	Set of tasks
π	Platform with the set of operating speeds
T_A	Ambient temperature
$T_{\min}$	Minimum thermal threshold
$T_{\max}$	Maximum thermal threshold
$P_D(t)$	Dynamic current
$P_L(t)$	Leakage current
a, b, α	Hardware dependent parameters
s	Operating speed
t_0	Time to cool-down from $T_{\max}$ to $T_{\min}$
$T_{h_s}(t)$	Heating function
$T_c(t)$	Cooling function
B_i	Blocking term
ΔC	Longest admissible execution time
$\hat{T}_{h_s}(t)$	Shifted heating function
$\hat{T}_c(t)$	Shifted cooling function
$W_{h_{s_\ell}}(t)$	Heating function of a sequence
x_ℓ	Cooling time required for the execution of τ_ℓ
R_i^{TA}	Worst case thermal-aware response time

As future work, there is a substantial number of interesting problems lying ahead of us. However, we plan to start by tackling the following ones, given in a chronological order:

(*i*) **Running experiments:** perform experiments on real platforms to expose the differences due to model abstraction of our approach;

(*ii*) **Multi-core problem:** address the multi-core problem under thermal-aware design by assuming a fully partitioned and/or a semi-global scheduler; and

(*iii*) **Mapping problem:** explore and assess the performance of various 2-phase mapping strategies, namely: *task-to-core* mapping and *task-to-speed* mapping on each individual core for an optimization of the overall heat dissipated at run-time.

ACKNOWLEDGMENTS

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020); also by the European Union through the Clean Sky 2 Joint Undertaking, under the H2020 Framework Programme (H2020-CS2-CFP08-2018-01), grant agreement nr. 832011 (THERMAC).

REFERENCES

[1] Y. Abdeddaïm, Y. Chandarli, and D. Masson. 2013. The Optimality of PFPasap Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems. In *25th Euromicro Conf. on Real-Time Systems*. 47–56. https://doi.org/10.1109/ECRTS.2013.16

[2] R. Ahmed, P. Ramanathan, and K. K. Saluja. 2014. Necessary and Sufficient Conditions for Thermal Schedulability of Periodic Real-Time Tasks. In *26th Euromicro Conference on Real-Time Systems*. 243–252. https://doi.org/10.1109/ECRTS.2014.15

[3] Hakan Aydin, Rami Melhem, Daniel Mossé, and Pedro Mejía-Alvarez. 2004. Power-Aware Scheduling for Periodic Real-Time Tasks. *IEEE Trans. Comput.* 53, 5 (May 2004), 584–600. https://doi.org/10.1109/TC.2004.1275298

[4] T. D. Burd and R. W. Brodersen. 2000. Design issues for Dynamic Voltage Scaling. In *ISLPED'00: Proc. of the International Symposium on Low Power Electronics and Design (Cat. No.00TH8514)*. 9–14. https://doi.org/10.1145/344166.344181

[5] Y. Chandarli, N. Fisher, and D. Masson. 2015. Response Time Analysis for Thermal-Aware Real-Time Systems under Fixed-Priority Scheduling. In *IEEE 18th Int. Symp. on Real-Time Dist. Computing*. 84–93. https://doi.org/10.1109/ISORC.2015.34

[6] T. Chantem, R. P. Dick, and X. S. Hu. 2008. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *2008 Design, Automation and Test in Europe*. 288–293. https://doi.org/10.1109/DATE.2008.4484694

[7] Thidapat Chantem, X. Sharon Hu, and Robert P. Dick. 2009. Online Work Maximization Under a Peak Temperature Constraint. In *Int. Symp. on Low Power Electronics and Design* (San Fancisco, CA, USA). ACM, New York, NY, USA, 105–110. https://doi.org/10.1145/1594233.1594257

[8] J. Chen, S. Wang, and L. Thiele. 2009. Proactive Speed Scheduling for Real-Time Tasks under Thermal Constraints. In *15th IEEE Real-Time and Embedded Technology and App. Symp.* 141–150. https://doi.org/10.1109/RTAS.2009.30

[9] L. Cucu-Grosjean and J. Goossens. 2011. Exact Schedulability Tests for Real-time Scheduling of Periodic Tasks on Unrelated Multiprocessor Platforms. *J. Syst. Archit.* 57, 5 (May 2011), 561–569. https://doi.org/10.1016/j.sysarc.2011.02.007

[10] John S Duncan. 2016. Pilot's handbook of aeronautical knowledge. *Oklahoma City: United States United States Department of Transportation, Federal Aviation Administration, Airman Testing Standards Branch* (2016).

[11] N. Fisher, J.-J. Chen, S. Wang, and L. Thiele. 2011. Thermal-Aware Global Real-Time Scheduling and Analysis on Multicore Systems. *JSA* 57, 5 (2011), 547–560. https://doi.org/10.1016/j.sysarc.2010.09.010

[12] Joel Goossens and Christophe Macq. 2001. Limitation of the Hyper-Period in Real-Time Periodic Task Set Generation. In *In Proceedings of the RTS Embedded System (RTS)*. 133–147.

[13] Inki Hong, Gang Qu, M. Potkonjak, and M. B. Srivastavas. 1998. Synthesis techniques for low-power hard real-time systems on variable voltage processors. In *Proc. 19th IEEE Real-Time Systems Symposium*. 178–187. https://doi.org/10.1109/REAL.1998.739744

[14] K. Jeffay, D. F. Stanat, and C. U. Martel. 1991. On non-preemptive scheduling of period and sporadic tasks. In *Proceedings Twelfth Real-Time Systems Symposium*. 129–139. https://doi.org/10.1109/REAL.1991.160366

[15] R. Jejurikar and R. Gupta. 2005. Energy aware non-preemptive scheduling for hard real-time systems. In *17th Euromicro Conference on Real-Time Systems (ECRTS'05)*. 21–30. https://doi.org/10.1109/ECRTS.2005.13

[16] W. Kim, J. Kim, and S. Min. 2002. A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '02)*. IEEE Computer Society, USA, 788. https://doi.org/10.5555/882452.874497

[17] A. Kumar, L. Shang, L. Peh, and N. K. Jha. 2008. System-Level Dynamic Thermal Management for High-Performance Microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 1 (2008), 96–108. https://doi.org/10.1109/TCAD.2007.907062

[18] P. Kumar and L. Thiele. 2011. Thermally Optimal Stop-Go Scheduling of Task Graphs with Real-Time Constraints. In *Asia and South Pacific Design Automation Conf., ASP-DAC*. IEEE. https://doi.org/10.1109/ASPDAC.2011.5722170

[19] J. P. Lehoczky. 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *11th Real-Time Syst. Symp.* 201–209. https://doi.org/10.1109/REAL.1990.128748

[20] Minming Li and F. Yao. 2005. An Efficient Algorithm for Computing Optimal Discrete Voltage Schedules, In Mathematical Foundations of Computer Science. *SIAM J. Comput.* 35, 652–663. https://doi.org/10.1007/11549345_56

[21] W. Liu and A. Nannarelli. 2010. Temperature aware power optimization for multicore floating-point units. In *Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*. 1134–1138. https://doi.org/10.1109/ACSSC.2010.5757581

[22] C. D. Locke, D. R. Vogel, and T. J. Mesler. 1991. Building a predictable avionics platform in Ada: a case study. In *Proc. Twelfth Real-Time Systems Symposium*. 181–189. https://doi.org/10.1109/REAL.1991.160372

[23] V. Nelis, P. Meumeu Yomsi, and J. Goossens. 2013. Feasibility Intervals for Homogeneous Multicores, Asynchronous Periodic Tasks, and FJP Schedulers. In *21st Int. Conference on Real-Time Networks and Systems* (Sophia Antipolis, France). ACM, NY, USA, 277–286. https://doi.org/10.1145/2516821.2516848

[24] Santiago Pagani. 2016. *Power, Energy, and Thermal Management for Clustered Manycores*. Ph.D. Dissertation. Karlsruher Institut für Technologie. https://doi.org/10.5445/IR/1000063307

[25] J. Perez Rodriguez and P. Meumeu Yomsi. 2019. Thermal-Aware Schedulability Analysis for Fixed-Priority Non-preemptive Real-Time Systems. In *IEEE Real-Time Systems Symposium (RTSS)*. 154–166. https://doi.org/10.1109/RTSS46320.2019.00024

[26] T. Pering, T. Burd, and R. Brodersen. 2000. Voltage scheduling in the IpARM microprocessor system. In *Proc. International Symposium on Low Power Electronics and Design*. 96–101. https://doi.org/10.1145/344166.344530

[27] Bhanuchander Reddy Poreddy and Steven Corns. 2011. Arguing Security of Generic Avionic Mission Control Computer System (MCC) using Assurance Cases. *Procedia Computer Science* 6 (2011), 499 – 504. https://doi.org/10.1016/j.procs.2011.08.092 Complex adaptive sysytems.

[28] X. Qin, W. Wang, and P. Mishra. 2012. TCEC: Temperature and Energy-Constrained Scheduling in Real-Time Multitasking Systems. *TCAD* 31, 8 (2012), 1159–1168. https://doi.org/10.1109/TCAD.2012.2190824

[29] G. Quan and Y. Zhang. 2009. Leakage Aware Feasibility Analysis for Temperature-Constrained Hard Real-Time Periodic Tasks. In *21st Euromicro Conference on Real-Time Systems*. 207–216. https://doi.org/10.1109/ECRTS.2009.28

[30] Deepak Rajan and Philip S. Yu. 2007. On Temperature-Aware Scheduling for Single-Processor Systems. In *High Performance Computing*, Srinivas Aluru, Manish Parashar, Ramamurthy Badrinath, and Viktor K. Prasanna (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 342–355. https://doi.org/10.1007/978-3-540-77220-0_33

[31] S. Shaik and S. Baskiyar. 2017. Proactive thermal aware scheduling. In *18th Int. Green and Sustainable Comp. Conf. (IGSC)*. 1–6. https://doi.org/10.1109/IGCC.2017.8323604

[32] D. Shin, J. Kim, and S. Lee. 2001. Intra-task voltage scheduling for low-energy hard real-time applications. *IEEE Design Test of Computers* 18, 2 (March 2001), 20–30. https://doi.org/10.1109/54.914596

[33] Y. Shin, K. Choi, and T. Sakurai. 2000. Power optimization of real-time embedded systems on variable speed processors. In *IEEE/ACM Int. Conf. on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Tech. Papers (Cat. No.00CH37140)*. 365–368. https://doi.org/10.1109/ICCAD.2000.896499

[34] K. Skadron, M. R. Stan, W. Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and D. Tarjan. 2003. Temperature-aware microarchitecture. In *30th Annual Int. Symp. on Computer Architecture*. 2–13. https://doi.org/10.1109/ISCA.2003.1206984

[35] G. M. Tchamgoue, K. H. Kim, and Y. Jun. 2012. Dynamic Voltage Scaling for Power-aware Hierarchical Real-Time Scheduling Framework. In *IEEE 15th International Conference on Computational Science and Engineering*. 540–547. https://doi.org/10.1109/ICCSE.2012.80

[36] G. Terzopoulos and H. D. Karatza. 2013. Dynamic Voltage Scaling Scheduling on Power-Aware Clusters under Power Constraints. In *IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications*. 72–78. https://doi.org/10.1109/DS-RT.2013.16

[37] Shengquan Wang, Youngwoo Ahn, and Riccardo Bettati. 2010. Schedulability analysis in hard real-time systems under thermal constraints. *Real-Time Systems* 46, 2 (Oct. 2010), 160–188. https://doi.org/10.1007/s11241-010-9104-7

[38] Shengquan Wang and R. Bettati. 2006. Reactive speed control in temperature-constrained real-time systems. In *18th Euromicro Conference on Real-Time Systems*.

[39] Baoxian Zhao, Hakan Aydin, and Dakai Zhu. 2013. Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 18 (03 2013). https://doi.org/10.1145/2442087.2442094

[40] X. Zhong and C. Xu. 2007. Energy-Aware Modeling and Scheduling for Dynamic Voltage Scaling with Statistical Real-Time Guarantee. *IEEE Trans. on Computers* 56, 3 (March 2007), 358–372. https://doi.org/10.1109/TC.2007.48

[41] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma. 2016. Thermal-Aware Task Scheduling for Energy Minimization in Heterogeneous Real-Time MPSoC Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 8 (2016), 1269–1282. https://doi.org/10.1109/TCAD.2015.2501286