A Thermal-Aware Approach for DVFS-enabled Multi-core Architectures

Javier Pérez Rodríguez*
Patrick Meumeu Yomsi*
Pavel Zaykov

*CISTER Research Centre

CISTER-TR-221205

2023

A Thermal-Aware Approach for DVFS-enabled Multi-core Architectures

Javier Pérez Rodríguez*, Patrick Meumeu Yomsi*, Pavel Zaykov

*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, pavel.zaykov@honeywell.com

https://www.cister-labs.pt

Abstract

Reducing thermal dissipation is vital for modern multi-core architectures to meet increasing computational demands. In this paper, we consider non-preemptive periodic tasks at two importance levels - Safety-Critical tasks (SC-tasks) and Best-Effort tasks (BE-tasks) - executing on a homogeneous multi-core processor with DVFS capabilities under thermal-aware design. We assume that tasks are scheduled according to a fully partitioned Fixed-Task-Priority (FTP) scheduler. Then, we propose two main contributions: (1) a new scheduling scheme, called NP-SafeSC, which reduces the responsiveness of SC-tasks as much as possible by procrastinating the execution of some BE-tasks; and (2) a new framework, called NP-ThermCare, that allows controlling both processor activity and the triggering of the cooling mechanism so that timing and thermal requirements are met. We provide a thorough analysis of our solutions, validate the results and evaluate their performance against a real-world use-case and intensive simulations. Our approach shows a consistent improvement of NP-SafeSC over NP-COIN in the responsiveness of all SC-tasks on each core. In particular, the improvement for the SC-task with the lowest priority reaches 39.53% for the real-world use-case and 45.17% for the simulations.

A Thermal-Aware Approach for DVFS-enabled Multi-core Architectures

Javier Pérez Rodríguez
CISTER, ISEP, IPP, Porto, Portugal
perez@isep.ipp.pt

Patrick Meumeu Yomsi
CISTER, ISEP, IPP, Porto, Portugal
pmy@isep.ipp.pt

Pavel Zaykov
Honeywell Aerospace, Brno, Czechia
pavel.zaykov@honeywell.com

***Abstract*—Reducing thermal dissipation is vital for modern multi-core architectures to meet increasing computational demands. In this paper, we consider non-preemptive periodic tasks at two importance levels – Safety-Critical tasks (*SC-tasks*) and Best-Effort tasks (*BE-tasks*) – executing on a homogeneous multi-core processor with DVFS capabilities under thermal-aware design. We assume that tasks are scheduled according to a fully partitioned Fixed-Task-Priority (FTP) scheduler. Then, we propose two main contributions: (1) a new scheduling scheme, called** NP-SafeSC**, which reduces the responsiveness of SC-tasks as much as possible by procrastinating the execution of some BE-tasks; and (2) a new framework, called** NP-ThermCare**, that allows controlling both processor activity and the triggering of the cooling mechanism so that timing and thermal requirements are met. We provide a thorough analysis of our solutions, validate the results and evaluate their performance against a real-world use-case and intensive simulations. Our approach shows a consistent improvement of** NP-SafeSC **over** NP-COIN **in the responsiveness of all SC-tasks on each core. In particular, the improvement for the SC-task with the lowest priority reaches** 39.53% **for the real-world use-case and** 45.17% **for the simulations.**

***Index Terms*—thermal-aware scheduling, non-preemptive schedulers, DVFS, multi-core platforms, safety-critical systems.**

I. Introduction

In recent decades, thermal management has become an important issue in the development of safety-critical computing systems geared by multi-core processors. In short, sub-micron technological advances have exponentially increased the power density of the chip, resulting in higher heat dissipation and thermal hotspots than ever before. In this context, malfunction, low reliability and even permanent damage can occur. Furthermore, in application domains such as avionics, which is our target area, it has become common to run tasks of multiple levels of importance on the same multi-core processor. If this is not carefully managed during run-time, undesirable interactions may occur that can affect system safety, performance, and reliability. Typically, we need to pay special attention to the timing behavior of important tasks, since their eventual failures (i.e., missed deadlines) are severe. In this work, we consider systems consisting of tasks at two importance levels: (1) safety-critical tasks (*SC-tasks*), where missed deadlines are forbidden and can lead to catastrophic consequences; and (2) best-effort tasks (*BE-tasks*), where occasional missed deadlines are undesirable but acceptable. Therefore, in order to adequately mitigate the interaction between tasks, efficient and cost-effective thermal management solutions are of paramount importance as they help system designers maintain temperature at acceptable levels (i.e., within predefined limits) while ensuring that all SC-tasks meet their timing and thermal requirements. We propose a model-based formalism and careful scheduling considerations that do not depend on ad-hoc techniques to enable a feasible system implementation. We build on Rodríguez and Yomsi [1], who propose an efficient proactive thermal-aware scheduler for DVFS-enabled single-cores, called NP-COIN. Therein, tasks are executed non-preemptively[1] (a common execution mode in avionics [2], [3]) to: (1) avoid unpredictable timing interference between tasks; (2) achieve a higher degree of predictability; and (3) avoid any potential issues associated with preemption. In addition, the scheduler (i) introduces cooling periods during run-time only when absolutely necessary to keep the temperature within specified range; and (ii) allows controlling processor activity with the least possible impact on performance. In summary, we design a thermal-aware scheduler, called NP-SafeSC, to reduce the responsiveness of SC-tasks as much as possible by procrastinating some BE-tasks, in a multi-core context.

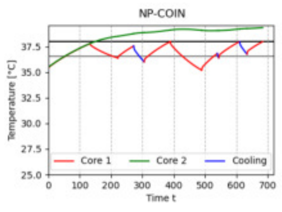
$\triangleright$ **This research.** We propose (1) a new scheduling scheme, called NP-SafeSC, that reduces the responsiveness of SC-tasks as much as possible by procrastinating the execution of some BE-tasks; and (2) a new framework, called NP-ThermCare, that allows controlling both the processor activity and the triggering of the cooling mechanism for multi-core platforms. We restrict our focus to a *dual-core* platform architecture with *inter-task* DVFS-enabled[2] homogeneous cores[3] for sake of readability. At this point, it should be emphasized that generalizing to architectures with a higher number of cores $(m > 2)$ does not add a major conceptual and/or technical issue to the problem. It merely requires knowledge of the underlying platform topology (i.e., the arrangement of cores on the chip) and additional computing power. We validate our theoretical results and evaluate the performance of our solution by using a real-world use-case (see Section VI-A) and intensive simulations (see Section VI-B).

II. Motivation

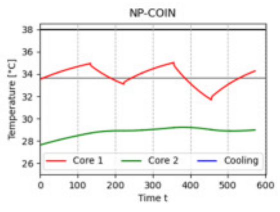
Despite the observed efficiency of the NP-COIN scheduler for single-core processors, its extension to multi-core architec-

[1]Once a job starts executing, it cannot be interrupted prior to its completion.
[2]Task's speed is not changed at run-time until it is preempted or completed.
[3]The cores have the same computing capabilities and are interchangeable.

tures is neither simple nor trivial. The very fact that multiple cores in a multi-core processor share the same die greatly complicates the problem of controlling timing and thermal constraints during run-time. This is because the temperature of each core changes dynamically and is affected by many factors, including the local workload, the temperature of neighboring logic cells, the material, the ambient temperature, and the cooling mechanisms on the chip. In addition, if SC-tasks and BE-tasks share the same core, then BE-tasks may cause SC-tasks to fail due to the blocking time associated with the non-preemptive execution mode.



(a) Both cores are active



(b) Core$_2$ is switched-off.

Fig. 1: Dual-core thermal behavior.

Figure 1 shows the thermal run-time behavior of a dual-core platform executing a real-world avionics use-case, namely "*Flight Management System (FMS) simulation of F-16 fighter aircraft*" [4]. The minimum thermal constraint of the chip is $T_{\min} = 25°C$ and the maximum thermal constraint is set to $T_{\max} = 38°C$. The two cores – Core$_1$ (see *red* curve) and Core$_2$ (see *green* curve) – are DVFS-enabled and operate at speeds of $[0.6; 0.9; 1.2]$ GHz each. The details of the use-case and task-to-core mapping are given in Table I, Section VI-A. Each core executes SC-tasks and BE-tasks together, see rows highlighted in white and gray, respectively. In Figure 1a, we assume the scenario where Core$_1$ executes its local workload by following a NP-COIN scheduler (here Rate Monotonic) and Core$_2$ is always busy executing its local workload at the highest available speed (i.e., at 1.2 GHz). As can be observed, the maximum thermal threshold is always met on Core$_1$ thanks to the introduction of cooling periods (see the *blue* segments) in the schedule of its local workload, but the situation is out of control on Core$_2$, where the maximum thermal threshold is violated. This situation is clearly due to the inter-core thermal interference during run-time. To illustrate this claim, let us look at the picture when Core$_2$ is switched-off (see Figure 1b). Here, only Core$_1$ is executing its local workload, and we can see that no cooling window was required in the schedule. However, we observe thermal activity on Core$_2$ with a peak temperature of $29.23°C - 25°C = 4.23°C$, i.e., 32.53% of the admissible thermal range $[25°C; 38°C]$.

III. Model of Execution

▷ **Task model.** We consider a set $\tau \overset{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n recurring *independent* tasks where each task is either (1) a *safety critical* (SC) task or (2) a *best effort* (BE) task (designated by the system designer to specify its "criticality" or "degree of importance"). We denote by τ_{SC} and τ_{BE} the subset of SC-tasks and BE-tasks, respectively. Every $\tau_i \in \tau$ is modeled by a *constrained-deadline periodic* task characterized by a 5-tuple (O_i, C_i, D_i, T_i, L_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; T_i is the *exact* inter-arrival time between two consecutive releases of task τ_i; and finally $L_i \in \{\text{SC}, \text{BE}\}$ is the criticality level. These parameters are given with the interpretation that during the execution of the system, task τ_i generates a (potentially infinite) number of successive jobs $\tau_{i,k}$, where $k = 1, \ldots, \infty$ is the job index. Specifically, job $\tau_{i,k}$: (1) is released at time $r_{i,k} \overset{\text{def}}{=} O_i + (k-1) \cdot T_i$, where $r_{i,1} \overset{\text{def}}{=} O_i$ (the release time of the first job of τ_i); (2) has an execution requirement of C_i; and (3) must complete within $[r_{i,k}, d_{i,k}]$, where $d_{i,k} \overset{\text{def}}{=} r_{i,k} + D_i$.

▷ **Platform model.** Without any loss of generality, we consider a *dual-core* platform $\pi \overset{\text{def}}{=} [\pi_1, \pi_2]$ composed of two homogeneous processing elements. We assume that every core $\pi_p \in \pi$ (with $p = 1, 2$) is DVFS-enabled and equipped with a set of $\ell \geq 1$ discrete speeds $s_{\pi_p} = \{s_1, \ldots, s_j, \ldots, s_\ell\}$, where $s_{\max} = s_1 \geq \ldots \geq s_j \geq \ldots \geq s_\ell = s_{\min}$ holds true for sake of easy presentation. Each π_p supports one inactive mode (when the core is switched-off) and ℓ active modes that are characterized by their operating speeds s_j. Tasks can be executed on a core only if it is in the active mode, otherwise the core is idle or cooling-down. In the latter case, no task is allowed to execute, even if ready and pending.

▷ **Scheduler model.** We assume that all the tasks are executed on the platform π in a non-preemptive manner by following a Fully Partitioned Fixed-Task-Priority scheduler, i.e., (1) each task is assigned to a specific core at design time; (2) each core executes its subset of tasks non-preemptively by following a classical FTP scheduler; and finally (3) migrations among cores are forbidden at run-time. We have $\tau = \tau(\pi_1) \cup \tau(\pi_2)$ and $\tau(\pi_1) \cap \tau(\pi_2) = \emptyset$, where $\tau(\pi_p)$ denotes the subset of tasks assigned to core π_p. We recall that an FTP scheduler assigns a constant priority to every task that is passed on to all its jobs. We assume that all the jobs generated from a task are executed at the same *constant* speed. Thus, 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. Such a restriction is common in the literature [4], [5]. We denote by $hp(\tau_i)$ (resp., $lp(\tau_i)$) the subset of tasks of higher (resp., lower) priority than τ_i. At a higher level of abstraction, this means that each task τ_i mapped to core, say π_p, may suffer a maximum blocking time given by Equation 1, where $s(\tau_j)$ is the speed at which task τ_j is executed.

$$B_i \overset{\text{def}}{=} \max_{\substack{\tau_j \in lp(\tau_i) \\ \tau_j \in \tau(\pi_p)}} \left\{ \frac{C_j}{s(\tau_j)} \right\} \tag{1}$$

At any point in time, if two jobs are ready and compete for execution, it is the job coming from the task with the highest priority that is executed first. This implicitly assumes that all the tasks have distinct priorities and there is at most one job per task that is ready at any time instant. The latter is guaranteed by the adopted task model, since $D_i \leq T_i$ for all τ_i. We enforce that *all SC-tasks have a higher priority than all*

BE-tasks and missing a deadline on a BE-tasks has no critical consequences. Any FTP scheduler can be used to define the priority of the tasks within each task type.

▷ **Thermal model.** We consider a Resistance-Capacitance (RC) thermal network [6]. Here, different parts of the chip and cooling solution are represented by thermal nodes and there are at least as many thermal nodes as blocks in the floorplan. Then, the temperatures at any instant in time are modeled as a function of three factors, namely: (i) the ambient temperature, (ii) the power consumption, and finally, (iii) the heat transfer among neighboring elements. For a system with N thermal nodes and by applying Kirchoff's first law [6], Equation 2 yields the resulting system of first-order differential equations

$$AT' + BT = P + T_{amb}G \tag{2}$$

where (i) matrix $A = [a_{i,j}]_{N\times N}$ contains the thermal capacitance values; (ii) matrix $B = [b_{i,j}]_{N\times N}$ contains the thermal conductance values between vertical and lateral neighboring nodes; (iii) column vector $T = [T_i(t)]_{N\times 1}$ represents the temperatures on the thermal nodes; (iv) column vector $T' = [T'_i(t)]_{N\times 1}$ accounts for the first-order derivative of the temperature on each thermal node with respect to time; (v) column vector $P = [P_i]_{N\times 1}$ contains the values of the power consumption on every node. Assuming that thermal node i is operating at speed s_j then we modeled it consumption as $P_i(s_j) = \beta_0 \cdot s_j^{\alpha} + \beta_1 \cdot s_j + \beta_2$ where α, β_0, β_1, β_2 are processor specific constants. This expression has proven to closely model the average power consumption on a core [7]. We consider: $\alpha = 3$, $\beta_0 = 12.5$, $\beta_1 = 1.5625$, $\beta_2 = 1.5869$, obtained from the *NXP i.MX8 QuadMax* datasheet [8]. We set $T_{amb} = 25°C$; and (vi) column vector $G = [b_{amb_i}]_{N\times 1}$ contains the thermal conductances between each node and the ambient temperature.

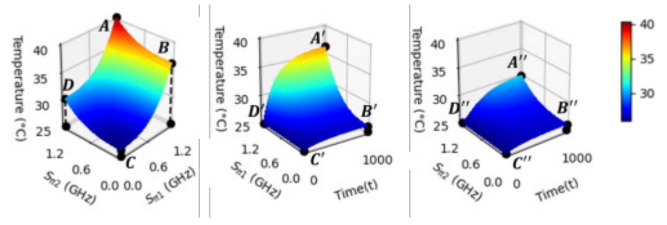
To solve this system of first-order differential equations, we assume that Core$_1$ and Core$_2$ operate at reference speeds s_{π_1} and s_{π_2} at a specific time instant. Note that $s_{\pi_1}, s_{\pi_2} \in \{s_1, s_2, \ldots, s_\ell\}$. Then, we use the well-established multi-dimensional *Laplace transform* technique. At any time instant t, the general shape of the solution $T_i^{(s_{\pi_1}, s_{\pi_2})}(t)$ for thermal node i is given by Equation 3.

$$T_i^{(s_{\pi_1}, s_{\pi_2})}(t) = T_i^{\infty}(s_{\pi_1}, s_{\pi_2}) + \sum_{\zeta=0}^{N} f_{i,\zeta}(s_{\pi_1}, s_{\pi_2}) \cdot e^{-\lambda_\zeta \cdot t} \tag{3}$$

In Equation 3: (1) T_i^{∞} is the *asymptote* of thermal node i w.r.t. time or the temperature to which node i will eventually converge to; (2) $f_{i,\zeta}$ are polynomial functions; and (3) λ_ζ are non-negative real numbers. Note that T_i^{∞} and $f_{i,\zeta}$ are functions of the core speeds s_{π_1} and s_{π_2}. The above-mentioned parameters; and matrices A, B, P and G set as later in Section VI-A.

This thermal model is able to address various thermal effects like the *transient heat exchange* between neighboring cores and the *permanent heat dissipation* associated with the static and/or dynamic power of the idle system. The run-time activity of Core π_2 on Core π_1 is materialized by the presence of its speed in the heating function of this core. Also, it follows from Equation 4 that $T_1^{\infty}(s_{\pi_1}, s_{\pi_2}) = \lim_{t\to+\infty} T_1^{(s_{\pi_1}, s_{\pi_2})}(t)$.

Assuming that the speeds s_{π_1} and s_{π_2} varies from 0 (when the corresponding core is switched-off) to 1.2 GHz (i.e., when the core executes at the maximum speed available on the chip), Figure 2 illustrates the actual curves for $T_1^{\infty}(s_{\pi_1}, s_{\pi_2})$, i.e., the thermal behavior of the system in its steady state (see Figure 2a); $T_1^{(s_{\pi_1},0)}(t)$, i.e., the thermal behavior when π_1 is the only active core and the neighboring core (here π_2) is inactive (see Figure 2b); and finally $T_1^{(0,s_{\pi_2})}(t)$, i.e., the thermal behavior when π_1 is inactive and π_2 is the only active core (see Figure 2c).

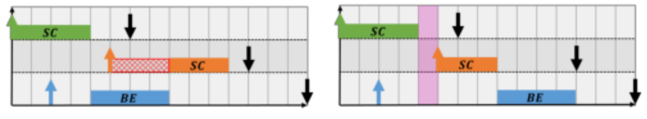


(a) Steady state (b) Intra-core (c) Inter-core

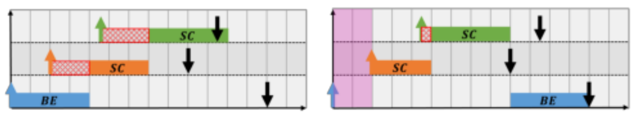
Fig. 2: Dual-core thermal behaviors from the thermal model.

IV. NP-SafeSC specification

To comprehend the specifics of the NP-SafeSC scheduler, it is important to describe the main intuition behind the scheduling policy. NP-SafeSC is designed as a proactive scheduler [1] that computes the length of any eventual cooling period (inserted into the schedule of the local workload) before dispatching the corresponding task. The scheduler goal is to reduce the responsiveness of the SC-tasks as much as possible. To this end, it commands the procrastination of BE-tasks each time their execution would insert a blocking time into the execution of a SC-task. Figure 3 illustrates this idea with a task-set consisting of two SC-tasks (see *green rectangle* and *orange rectangle*); and one BE-task (see *blue rectangle*). The tasks are ordered by their priorities as follows: the "green task" has the highest priority; followed by the "orange task"; and finally the "blue task" has the lowest priority. An upward arrow indicates the release of a job generated from the corresponding task, and a downward arrow indicates the deadline.



(a) Blocking in NP-COIN. (b) Blocking avoided in NP-SafeSC.



(c) Cascade-Blocking in NP-COIN. (d) CB avoided in NP-SafeSC.

Fig. 3: NP-COIN vs. NP-SafeSC w.r.t. blocking times.

In Figure 3a, the "orange task", a SC-task, is released during the execution of the "blue task", a BE-task, and experiences a blocking time associated with the execution of the latter under NP-COIN. The blocking can clearly cause a deadline miss, unfortunately. This would be the case if another SC-task, say with 2 time units of execution and a higher priority than the "orange task", was also released during the execution window of the BE-task. In Figure 3b, the blocking of the "orange task" by the BE-task is avoided under NP-SafeSC. This is achieved by forcing an idle period (see the *purple* band) between the completion time of the "green task" and the release time of the "orange task" and by procrastinating the execution of the BE-task only after the completion time of the latter.

Figure 3c depicts yet another phenomenon. Here, the completion time of the BE-task triggers a "cascade-blocking" (CB) effect in the execution of SC-tasks, resulting in a deadline miss of the "green task", a SC-task, which is unacceptable. In Figure 3d, this CB is avoided by enforcing an idle period (see the *purple* band) between the release time of the "blue task" and the release time of the "orange task" and by procrastinating the execution of the BE-task after the completion time of all the SC-tasks released in their original execution window under NP-COIN. This way, each SC-task can be blocked at *most once at run-time and this blocking will only be due to the execution of another SC-task.* No BE-task can block the execution of a SC-task under NP-SafeSC. Consequently, the maximum blocking time B_i^{SC} that any SC-task, say $\tau_i \in \tau_{\text{SC}}$, mapped to core, say π_p, would suffer reduces to Equation 4 under NP-SafeSC, where $s(\tau_j)$ is the speed of task τ_j.

$$B_i^{\text{SC}} \overset{\text{def}}{=} \max_{\substack{\tau_j \in lp(\tau_i) \\ \tau_j \in \tau(\pi_p) \cap \tau_{\text{SC}}}} \left\{ \frac{C_j}{s(\tau_j)} \right\} \tag{4}$$

From this equation, it is obvious that $B_i^{\text{SC}} \leq B_i$ (as defined in Equation 1), since $\tau(\pi_p) \cap \tau_{\text{SC}} \subseteq \tau(\pi_p)$, thus improving the responsiveness of each SC-task.

▷ NP-SafeSC **pseudo-code.** Algorithm 1 provides the pseudo-code of the NP-SafeSC scheduler. We assume that tasks in the ready queue are re-indexed according to their priority upon the task- to-core mapping. We designate the pending ready task with the highest priority in the queue (ReadyQ) as τ_{next} at any given time. This algorithm receives as inputs (1) the taskset mapped to the core under analysis, here $\tau(\pi_p)$; (2) the specific task $\tau_i \in \tau(\pi_p)$ to be scheduled; (3) the set of SC-tasks $\tau_{\text{SC}} \cap \tau(\pi_p)$ on this core; (4) the factors A; B; P and G used to feed the thermal model; (5) the thermal boundaries $T_{\min}$ and $T_{\max}$; and finally (6) the task- to-speed mapping. Note that CheckReleases() checks the releases of all tasks within a time interval, while CheckSCReleases() only checks the releases of SC-tasks.

Algorithm 1: NP-SafeSC scheduler.

```
Data: $\tau(\pi_p)$; $\tau_i \in \tau(\pi_p)$; $\tau_{\text{SC}}$; $A$; $B$; $P$; $G$; $T_{\max}$; $T_{\min}$; Task-to-speed
    mapping.
Result: NP-SafeSC schedule.
ReadyQ $= \emptyset$;
Record all the releases at $t = 0$;
Update ReadyQ w.r.t. tasks criticality levels;
if $\tau_i \in \tau_{\text{SC}}$ then
    Compute $B_i^{\text{SC}}$ (see Equation 4);
else
    Compute $B_i$ (see Equation 1);
end
Execute($\tau_{B_i^{\text{SC}}}$) or Execute($\tau_{B_i}$);
while ReadyQ $\neq \emptyset$ do
    $\tau_{\text{next}}$ = NextExec(ReadyQ);
    if $\tau_{\text{next}} \notin \tau_{\text{SC}}$ then
        while CheckSCReleases($\tau_{\text{next}}$) do
            Update ReadyQ;
            $\tau_{\text{next}}$ = NextExec(ReadyQ);
            SC$_{\text{flag}}$ = TRUE;
        end
        if SC$_{\text{flag}}$ == TRUE then
            CoolUntilRelease($\tau_{\text{next}}$);
        end
    end
    if Temperature($\tau_{\text{next}}$) $> T_{\max}$ then
        cooling = ComputeCooling($\tau_{\text{next}}$);
        while CheckReleases(cooling) do
            Update ReadyQ;
            $\tau_{\text{next}}$ = NextExec(ReadyQ);
            cooling = ComputeCooling($\tau_{\text{next}}$);
            if NoFit(cooling, $\tau_{\text{next}}$) then
                cooling = ComputeCooling($\tau_{\text{next}}$);
            end
        end
        CoolDownAndExecute(cooling, $\tau_{\text{next}}$);
    else
        Execute($\tau_{\text{next}}$);
    end
    Remove($\tau_{\text{next}}$, ReadyQ);
    CheckReleases($\tau_{\text{next}}$);
    Update ReadyQ;
end
```

V. NP-ThermCare specification

From a purely holistic multi-core perspective, it is important to re-iterate at this point that the temperature of the individual cores changes dynamically at run-time. This means that it is practically infeasible, if not impossible, to predict the exact timing behavior of the system with high resolution under thermal-aware design. The situation exacerbates when all cores execute tasks in parallel (see Figure 1a). Consequently, the only option is to assume a worst-case situation in which each core is permanently under stress, both from a timing and thermal point of view, so as not to jeopardize the actual predictability of the entire system. The goal of the NP-ThermCare framework is to configure the system in such a way that no thermal or timing constraint is violated.

From the preceding discussion, the timing behavior of any task $\tau_i \in \tau$ assigned to a core, say π_p, can be affected by two factors from a thermal standpoint: (1) the "*intra-core thermal interference*", which is related *solely* to the execution of other tasks assigned to π_p when it runs in isolation, i.e., assuming all cores in the vicinity of π_p are switched-off (see Figure 2b); and (2) the "*inter-core thermal interference*", which is due *solely* to the execution of cores in the vicinity of π_p, i.e., irrespective of the actual activity on π_p (see Figure 2c). To fully grasp the essence of a per-core based solution, such as NP-ThermCare, it is essencial to understand the difference between these two components. The main challenge is to derive provably safe and tight upper-bounds based on an estimation of the maximum thermal intra- and inter-core interference.

▷ **Specifics of** NP-ThermCare**.** This framework allows us to control both intra- and inter-core thermal interference for a given task-to-core mapping strategy to maintain the system within a safe operating range from a timing and thermal viewpoint. Assuming that each core runs either a NP-COIN

or a NP-SafeSC scheduler, our methodology consists in the following four steps to be applied in a chronological manner:
Step 1: *Computation of the Thermal Profile (TP) of each core.* To profile each Core π_p, we configure the execution environment such that all other cores but π_p are switched-off (i.e., Core π_p operates in *full isolation*). This way, the inter-core thermal interference is completely removed from the equation, and all the heat generated from the platform is exclusively produced by the activity on Core π_p. In this configuration, we execute the workload assigned to Core π_p and compute its TP as the surface below the curve upon the execution of the task with the lowest priority in $\tau(\pi_p)$. We recall that $\tau(\pi_p)$ is the subset of tasks assigned to π_p upon the task-to-core mapping. This process is achieved by assuming the worst-case scenario (see [1] for further details).
Step 2: *Ranking each core*. Upon the completion of Step 1, we determine the order in which our per-core schedulability analysis will be performed, since the higher the TP of a core, the lower is its available room temperature and the higher is its probability to trigger additional cooling periods due to inter-core thermal interference. To this end, we sort the cores in a non-decreasing order of their TP and assign a rank to each of them. The core with the lowest TP is assigned the highest rank. In case two cores share the TP, then the tie is broken in an arbitrary manner. At the end of this process, we re-indexed the cores according to their ranks.
Step 3: *Signature speed of each core*. The signature speed $s^{\text{sig}}_{\pi_p}$ of Core π_p is used to reach two objectives: (1) to profile the core runtime activity; and (2) to ensure that the temperature of the core at any time instant falls below $T_{\max}$ (the maximum admissible temperature) upon the execution of its workload, i.e., $\tau(\pi_p)$. Specifically, $s^{\text{sig}}_{\pi_p}$ is the maximum operating speed that sets the average temperature $\bar{T}(\tau(\pi_p))$ of π_p — *computed in a similar manner as the* TP *of Core* π_p[4] — as the thermal asymptote upon the execution of $\tau(\pi_p)$ on a constant core speed. Formally, $s^{\text{sig}}_{\pi_p}$ is obtained for π_p by solving Equation 5, where the only variable is s_{π_p} and s_{π_j} (with $j \neq p$) is constant.

$$T_p^{\infty}(s_{\pi_1}, s_{\pi_2}) = \bar{T}(\tau(\pi_p)) \qquad (5)$$

Step 4: *"Schedulability analysis"*. Upon the completion of Step 2, the schedulability analysis of π_1 is achieved by using its heating function as obtained from the thermal model and the principles of the NP-COIN or NP-SafeSC scheduler on each core. Here, in the heating function, we assume that all neighboring cores are consistently executing at their maximum available speed (i.e., in conditions of "extreme" inter-core thermal interference). Once the schedulability of π_1 is guaranteed, we proceed with that of the next core (here π_2) in a similar manner, but this time assuming that π_1 operates at its signature speed $s^{\text{sig}}_{\pi_1}$ as defined in Step 3. If all the cores are schedulable, then the system is deemed as "schedulable" for the predefined task-to-core and task-to-speed mapping. Otherwise, it is deemed "not schedulable" and a new task-to-core and/or task-to-speed mapping scheme must be assessed.

[4]By computing the surface below the curve obtained upon the execution of the lowest priority task in $\tau(\pi_p)$, assuming the worst-case scenario.

▷ **Pseudo-code for the** NP-ThermCare **schedulability test.** This is provided by Algorithm 2. This algorithm receives as inputs (1) the task-to-speed mapping; (2) the task-to-core mapping; (3) the factors A; B; P and G to feed the thermal model; (4) the thermal boundaries $T_{\min}$ and $T_{\max}$; and finally (5) the per-core schedulers. Without loss of generality, we assume that the cores are re-indexed w.r.t their rank. This means that the first core to be analyzed is π_1, the second is π_2 and so on. We also assume that the tasks mapped to each core π_p are re-indexed w.r.t. their criticality level and τ_1 is initially the first task in the queue. Line 10 checks the schedulability of task τ_i on core π_p by following the input scheduler (e.g., NP-COIN or NP-SafeSC).

Algorithm 2: NP-ThermCare schedulability test.

```
Data: Task-to-speed mapping; Task-to-core mapping; $A$; $B$; $P$; $G$; $T_{\max}$;
      $T_{\min}$; Schedulers.
Result: NP-ThermCare schedulability test.
1  foreach $\pi_p \in \pi$ do
2      Compute the thermal profile of $\pi_p$;
3  end
4  Rank all cores in $\pi$;
5  Re-index all cores w.r.t. their ranks;
6  $s^{\text{sig}}_{\pi_j\,(\text{with } j \neq p)} = \max(\text{speeds})$;
7  $\text{Schedulability}_{\text{Flag}} = \text{TRUE}$;
8  foreach $\pi_p \in \pi$ do
9      foreach $\tau_i \in \tau(\pi_p)$ do
10         $\text{Schedulability}_{\text{Flag}}$ = analysis of scheduler($\tau(\pi_p)$) for $\tau_i$;
11         if $\text{Schedulability}_{\text{Flag}}$ == FALSE then
12             return $\tau(\pi_p)$ is not schedulable on $\pi$;
13         end
14     end
15     Compute $s^{\text{sig}}_{\pi_p}$ ;
16 end
17 return $\tau$ is schedulable on $\pi$;
```

▷ **Potential limitations.** The following limitations can be reported for this approach: (1) *A dependency on the number of cores*. The higher the number of computing elements, the higher the number of times we need to run the NP-COIN or NP-SafeSC scheduler before we can assess the system schedulability; and (2) *A dependency on the adopted task-to-core and task-to-frequency mapping strategies*. Since the task-to-core and task-to-frequency mapping strategies are inputs to this approach, they directly impact the rank of each core, and this in turn may impact the schedulability analysis.

VI. Experimental results

A. Application to a real-world use-case

In this section, we apply our methodology to the FMS use-case [4] in order to show the importance of each step and the comparison differences between NP-COIN and NP-SafeSC. The parameters of the adopted task sets upon the task-to-core mapping are provided in Table I. Here, the assumed local NP-COIN scheduler or local NP-SafeSC scheduler are Rate Monotonic per task type. We recall that $T_{\max} = 38°C$ and $T_{\min} = 25°C$. Finally, we consider the characteristics of the NXP i.MX8 QuadMax [8] to build the power and thermal models (see Section III). Matrices A, B and G are set as follows.

$$\underbrace{\begin{bmatrix} 83.063 & 0 & 0 & 0 \\ 0 & 83.063 & 0 & 0 \\ 0 & 0 & 305.102 & 0 \\ 0 & 0 & 0 & 305.102 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 56.112 & -0.200 & -55.912 & 0 \\ -0.200 & 56.112 & 0 & -55.912 \\ -55.912 & 0 & 58.467 & -0.939 \\ 0 & -55.912 & -0.939 & 58.467 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1.616 \\ 1.616 \end{bmatrix}}_{G}$$

TABLE I: Task set parameters

	Core 1				Core 2			
Tasks	C_i	$D_i = T_i$	s_i	u_i	C_i	$D_i = T_i$	s_i	u_i
τ_1	60	200	1.2	0.25	500	1000	0.9	0.017
τ_2	100	1000	1.2	0.083	100	5000	1.2	0.555
τ_3	80	1000	0.9	0.089	100	5000	1.2	0.042
τ_4	100	1000	1.2	0.083	500	10000	1.2	0.017
τ_5	60	1000	0.6	0.1				
τ_6	80	5000	1.2	0.013				
Total				**0.618**				**0.630**

Table II shows the worst-case response time (WCRT) and the time gain/loss[5] incurred by each SC-task and BE-task under the same settings (i.e., Core$_1$ executes its workload by following a NP-COIN or a NP-SafeSC scheduler and Core$_2$ is always busy executing its workload at 1.2 GHz).

TABLE II: Core$_1$ under NP-COIN vs. NP-SafeSC.

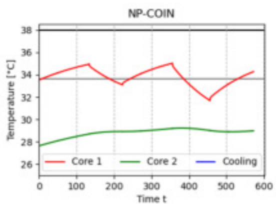
	WCRT					
	SC-tasks			BE-tasks		
Tasks	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6
NP-COIN	150.0	233.33	372.22	455.55	667.48	610.92
NP-SafeSC	138.88	222.22	222.22	557.86	750.00	716.66
Gain/Loss	7,41%	4.76%	4.76%	-22.45%	-12.36%	-17.30%

Figure 4 illustrates the WCRT of the lowest priority task on each core, for both schedulers, when it runs in isolation. In this figure, the execution of tasks is plotted in “red” on Core$_1$ and “green” on Core$_2$ under NP-COIN on each core (see Figure 4a and Figure 4b). Here, there is no distinction between the execution of SC-tasks and BE-tasks because NP-COIN is completely agnostic to the criticality levels of the tasks and their execution rely solely on their predefined priorities. In contrast, under NP-SafeSC on each core (see Figure 4c and Figure 4d), the execution of SC-tasks is plotted in “red” on Core$_1$ and “green” on Core$_2$; while the execution of BE-tasks is plotted in “gray” on both cores. Here, the enforced idle periods inserted in the schedule to reduce the responsiveness of SC-tasks are plotted in “purple”.

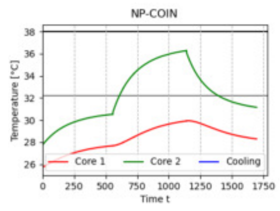
Upon completion of Step 1, we have: (1) $\text{TP}_{\pi_1} = 33.65°C$ and $\text{TP}_{\pi_2} = 32.22°C$ under NP-COIN; whereas we have (2) $\text{TP}_{\pi_1} = 33.09°C$ and $\text{TP}_{\pi_2} = 30.97°C$, under NP-SafeSC. From these results, $Rank(\pi_1) = 1$ and $Rank(\pi_2) = 2$ (see Step 2). This means that our per-core schedulability analysis, in both cases, will start by Core$_1$, assuming a constant maximum inter-core thermal interference from Core$_2$; followed by the schedulability analysis of Core$_2$, assuming that Core$_1$ consistently operates at its signature speed $s^{\text{sig}}_{\pi_1}$. With this configuration, Figure 5 displays the result of the schedule of the lowest priority task on π_1.

By applying the procedure presented in Step 3, we obtain $s^{\text{sig}}_{\pi_1} = 0.994$ GHz under NP-COIN on each core; and $s^{\text{sig}}_{\pi_1} = 0.986$ GHz under NP-SafeSC on each core. By plugging these values in the heating function of π_2 and by performing the schedulability analysis of this core, Figure 6 displays the result of the schedule of the lowest priority task on π_2.

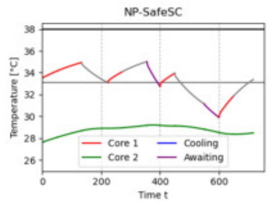
[5]A positive percentage means an improvement in the WCRT; whereas a negative percentage means a procrastination of the corresponding task.



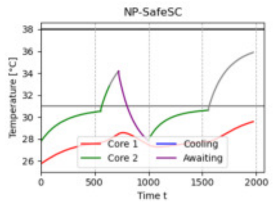
(a) Only Core$_1$ is active.



(b) Only Core$_2$ is active.

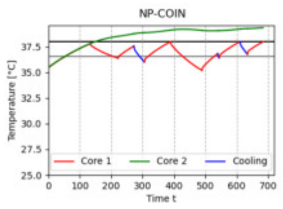


(c) Only Core$_1$ is active.

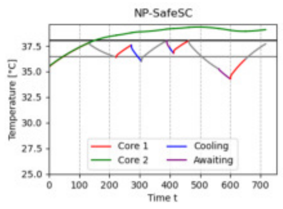


(d) Only Core$_2$ is active.

Fig. 4: Extraction of the TP for each core.



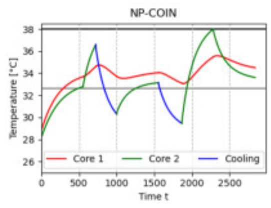
(a) Both cores active.



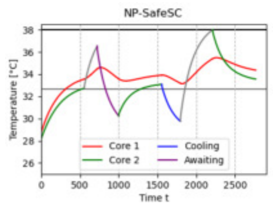
(b) Both cores active.

Fig. 5: Schedule of π_1 with π_2 operating at 1.2 GHz.

In contrast to Figure 5a and Figure 5b, we observed that all constraints are now satisfied for both cores (see Figure 6a and Figure 6b). The peak temperature reached on the platform is now $38.00°C$ on π_2 (i.e., $T_{\max}$), in contrast to $39.37°C$ and $39.34°$ in Figure 5a and in Figure 5b under NP-COIN and NP-SafeSC, respectively. This represents a gain of $\sim 3.48\%$ and $\sim 3.40\%$ in the peak temperatures. In addition, the thermal profiling of π_1 during the schedulability analysis of π_2 shows a peak temperature at $35.58°C$ against $29.93°C$ under NP-COIN and $29.58°C$ under NP-SafeSC observed when π_2 was executing in isolation (see Figure 4b and Figure 4d). This represents an increase of $\sim 15,87\%$ and $\sim 16.86\%$ above the maximum inter-core thermal interference when the core is running in isolation. This trend suggests that our approach is safe as this behavior must be coupled with the activity of the core. Table III reports the WCRT and the timing gain/loss incurred by each SC-task and BE-task in Core$_2$ after performing NP-ThermCare. Here, we compare the results when both cores run either NP-COIN or NP-SafeSC.



(a) Both cores active.



(b) Both cores active.

Fig. 6: Schedule of π_2 with π_1 operating at $s^{\text{sig}}_{\pi_1}$.

TABLE III: Core_2 under NP-COIN vs. NP-SafeSC.

	WCRT			
	SC-tasks		BE-tasks	
Tasks	τ_1	τ_2	τ_3	τ_4
NP-COIN	973.19	1056.52	1695.41	2284.25
NP-SafeSC	775.45	638.88	1694.44	2212.59
Gain/Loss	20.31%	39.53%	0.05%	3.13%

B. Application to synthetic workloads

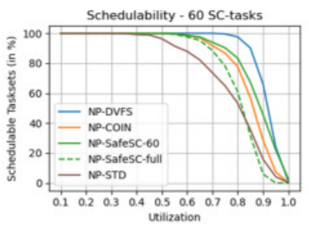
In this section we apply the proposed methodology to a high number of generated synthetic test cases.

▷ **Task set generation.** We generate $1,000$ task sets per-core with the target utilization varying from 0.1 to 1 by step of 0.05. Each core is equipped with three operating speeds $[0.6, 0.9, 1.2]$ GHz. The utilization u_i of each task τ_i is computed as $u_i \overset{\text{def}}{=} C_i/(T_i \cdot s_i)$. The longest admissible execution time ΔC on each core (computed as in [1]) is set at $\Delta C = 321.83$ and the task parameters are generated as follow: (1) the execution times C_i are uniformly distributed within $[\Delta C/4; 3 \cdot \Delta C/4]$; (2) the periods $T_i \in [1500; 162000]$ are generated by using the hyper-period limitation technique proposed in [9]; and (3) the deadlines D_i are randomly generated within $[0, 8 \cdot T_i; T_i]$. The task-to-speed mapping is performed in a random manner and the ratio of SC-tasks in each task set is chosen between 60%; 80% and 100% of the total number of tasks. The remainder are BE-tasks. Finally, the tasks are scheduled by following *Deadline Monotonic*.

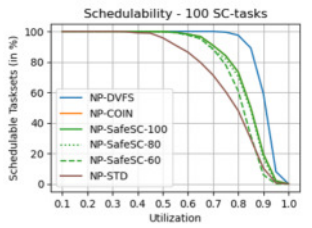
▷ **Evaluation metrics.** We validate the performance of our NP-ThermCare framework by running local executions of four different schedulers on each core. We compared (1) the *classical* DVFS, where only time matters while thermal constraints are ignored; (2) the proactive non-preemptive thermal-aware NP-COIN scheduler [1], which introduces cooling periods during run-time only when absolutely necessary to keep the temperature within specified parameters; (3) our proactive NP-SafeSC scheduler which aims to reduce the responsiveness of the SC-tasks as much as possible by procrastinating the execution of some BE-tasks; and finally (4) an adaptation to non-preemptive workloads of the so-called *Simple Time Derivative (STD)* proactive scheduler [10], which uses time derivatives on each core to predict future temperature rise, resulting in low temporal variability. Note that, the classical DVFS defines an upper bound for all possible solutions when other variables (here temperature) are considered in addition to time. Thus, the closer the behavior to DVFS, the better.

▷ **Interpretation of the results.** Figure 7 illustrates the system schedulability ratio when 60% of the tasks (see Figure 7a) and 100% of the tasks (see Figure 7b) in each core are SC-tasks. We observe that NP-DVFS always outperforms all other schedulers and schedules task sets even at high utilizations (see "blue" curves). This is because it ignores all thermal constraints. In Figure 7a, our NP-SafeSC-60 (see "solid green" curve) clearly dominates NP-COIN (see "orange" curve) and NP-STD (see "brown" curve) when both the timing and thermal constraints are checked *only* for SC-tasks. If these constraints are also checked for BE-tasks upon the procrastination of their execution (see "dash green" curve), then NP-COIN dominates NP-SafeSC-full and the behavior against NP-STD is contrasted. NP-STD dominates NP-SafeSC-full at high utilizations per core (here, from 85%).

From a numerical viewpoint, NP-SafeSC-60 can schedule at least 83.3% of the task-sets until 80% vs. 97.6% for classical DVFS (i.e., a performance loss of only 14.65%); 77.4% for NP-COIN (i.e., a performance gain of 7.08%); and 53.2% for NP-STD (i.e., a performance gain of 36.13%). The loss against classical DVFS decreases with an increase in the system utilization; whereas the gain over NP-COIN and NP-STD increases. Figure 7b illustrates the schedulability ratio with only of SC-tasks. Figure 8 compares the [Minimum-Average-Maximum] performances of the three schemes. The improvement in the responsiveness for the SC-task with the lowest priority – when 60% of the workload are SC-tasks – reaches 45.17%, obtained at 65% of utilization per core.



(a) (SC-tasks, BE-tasks) = (60, 40)%.



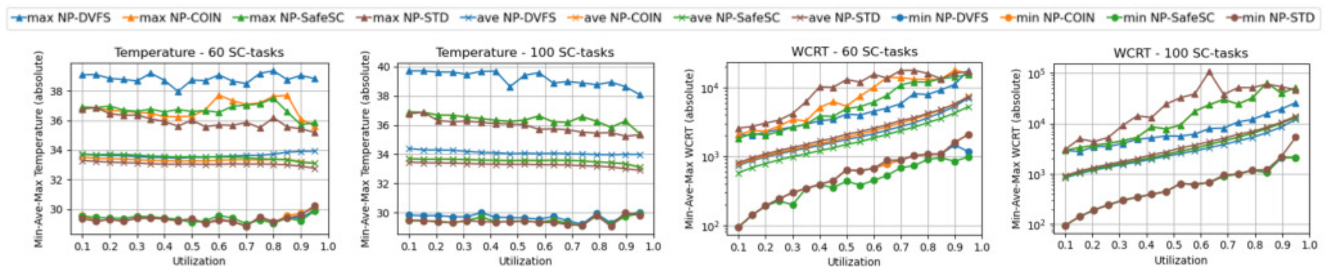
(b) (SC-tasks, BE-tasks) = (100, 0)%.

Fig. 7: Schedulability ratios.

VII. State of the Art

Targeting only the peak temperature. Fisher et al. [11] proposed a thermal-aware scheduling for sporadic real-time tasks to minimize peak temperature by deriving a preferred speed for each core. Chantem et al. [12] presented a Mixed-Integer Linear Programming (MILP) formulation to assign and schedule tasks with hard real-time constraints to minimize peak temperature. Schor et al. [13] proposed an analytical method to compute an upper bound on the worst-case peak temperature for a real-time system under all possible scenarios of task executions. Ahmed et al. [14] derived necessary and sufficient conditions for thermal feasibility of periodic task-sets. In [15], [16], the authors developed a task assignment heuristic that minimizes dynamic energy consumption and hence temperature. In [17], a multi-task look-ahead approach for managing power peaks and maximum temperatures is developed to ensure that dynamic slacks are allocated to tasks that have the greatest impact on the system. Recently, Safari et al. [18] proposed a thermal-aware scheduling scheme for fault-tolerant Mixed-Criticality Systems, called TherMa-MiCs, that satisfies the temperature and timing constraints of high-criticality tasks while attempting to maximize the QoS of low-criticality tasks. Unfortunately, all of these works ignore the thermal maximums of individual cores, which can significantly affect the resulting peak temperatures.

Inclusion of transient thermal behavior. In [19], the authors presented a convex optimization-based method for determining the operating frequencies of the underlying processor. Fu et al. [20] proposed a feedback thermal control



(a) Min-Ave-Max thermal variation. (b) Min-Ave-Max thermal variation. (c) Min-Ave-Max WCRT variation. (d) Min-Ave-Max WCRT variation.

Fig. 8: NP-COIN vs. NP-SafeSC vs. NP-STD vs. DVFS performances.

loop to enforce the desired temperature and utilization bounds. Pagani et al. [21] presented a method to calculate transient temperature for arbitrary time resolution without loss of accuracy, but we are not aware of any thermal-aware control technique. D'Souza and Rajkumar [22] proposed two algorithms to enable a thermally efficient sleep schedule. In [23], the authors proposed a thermal-aware server framework that considers the effect of continuously varying ambient temperature to limit the maximum operating temperature of mixed-criticality multicore systems. Recently, Ansari et al. [24] developed a thermal-aware technique for standby-sparing that aims to maximize the Quality of Service of soft real-time tasks. Nevertheless, none of these contributions have succeeded in accurately capturing transient and peak temperatures in the same framework while providing temporal guarantees. This work fills this gap.

VIII. Conclusion

This paper specifies an efficient thermal-aware resource management scheduler (NP-SafeSC) and a new framework (NP-ThermCare) for workloads running on DVFS-enabled multi-core platforms in a non-preemptive manner. NP-SafeSC reduced the responsiveness of Safety-Critical tasks, while procrastinating the execution of Best-Effort tasks. We validated the run-time behavior of our solution using a real-world use-case as well as intensive simulations. An interesting future work is to perform experiments on a real platform to expose the differences due to model abstraction.

Acknowledgment

This work was supported by the CISTER Research Unit (UIDP/UIDB/04234/2020) financed by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology); by project VALU3S (ECSEL/0016/2019 - JU grant nr. 876852), financed by national funds through FCT and European funds through the EU ECSEL JU. (The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey - Disclaimer: This document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains).

References

[1] J. P. Rodriguez and P. M. Yomsi, "An efficient proactive thermal-aware scheduler for dvfs-enabled single-core processors," in *RTNS*, 2021.
[2] ——, "Thermal-aware schedulability analysis for fixed-priority non-preemptive real-time systems," in *RTSS*, 2019, pp. 154–166.
[3] O. Benedikt, M. Sojka, P. Zaykov, D. Hornof, M. Kafka, P. Šůcha, and Z. Hanzálek, "Thermal-aware scheduling for mpsoc in the avionics domain: Tooling and initial results," in *RTCSA*, 2021, pp. 159–168.
[4] S. K. Roy, A. Sarkar, and R. Gangopadhyay, "Processor and bus co-scheduling strategies for real-time tasks with multiple service-levels," in *RTCSA*, 2021, pp. 21–30.
[5] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *ICCAD*, 2007, pp. 281–288.
[6] J. P. Rodriguez and P. M. Yomsi, "WiP: Towards a fine-grain thermal model for uniform multi-core processors," in *RTSS*, 2020, pp. 403–406.
[7] S. Pagani, "Power, energy, and thermal management for clustered many-cores," Ph.D. dissertation, Karlsruher Institut für Technologie, 2016.
[8] *i.MX 8QuadPlus Automotive and Infotainment Applications Processors*, NXP Semiconductors, 12 2020, rev. 1.
[9] V. Nelis, P. M. Yomsi, and J. Goossens, "Feasibility intervals for homogeneous multicores, asynchronous periodic tasks, and FJP schedulers," in *RTNS*, 2013, pp. 277–286.
[10] S. Shaik and S. Baskiyar, "Proactive thermal aware scheduling," in *IGSC*, 2017, pp. 1–6.
[11] N. Fisher, J. Chen, S. Wang, and L. Thiele, "Thermal-aware global real-time scheduling on multicore systems," in *RTAS*, 2009, pp. 131–140.
[12] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," *Trans. on VLSI Systems*, vol. 10, no. 10, pp. 1884–1897, 2010.
[13] L. Schor, I. Bacivarov, H. Yang, and L. Thiele, "Worst-case temperature guarantees for real-time applications on multi-core systems," in *RTAS*, 2012, pp. 87–96.
[14] R. Ahmed, P. Ramanathan, and K. Saluja, "Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks," in *ECRTS*, 2014, pp. 243–252.
[15] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Cooling-aware node-level task allocation for next-generation green hpc systems," in *HPCS*, 2016, pp. 690–696.
[16] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems," *TCAD*, vol. 35, no. 8, pp. 1269–1282, 2016.
[17] B. Ranjbar, T. D. A. Nguyen, A. Ejlali, and A. Kumar, "Online peak power and maximum temperature management in multi-core mixed-criticality embedded systems," in *DSD*, 2019, pp. 546–553.
[18] S. Safari, H. Khdr, P. Gohari, M. Ansari, S. Hessabi, and J. Henkel, "Therma-mics: Thermal-aware scheduling for fault-tolerant mixed-criticality systems," *TPDS*, vol. 33, pp. 1678–1694, 2022.
[19] A. Murali, S.and Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli, "Temperature control of high-performance multi-core platforms using convex optimization," in *DATE*, 2008, pp. 110–115.
[20] Y. Fu, N. Kottenstette, C. Lu, and X. D. Koutsoukos, "Feedback thermal control of real-time systems on multicore processors," in *EMSOFT*, 2012, p. 113–122.
[21] S. Pagani, J. Chen, M. Shafique, and J. Henkel, "Matex: Efficient transient and peak temperature computation for compact thermal models," in *DATE*, 2015, p. 1515–1520.
[22] S. M. D'souza and R. Rajkumar, "Thermal Implications of Energy-Saving Schedulers," in *ECRTS*, vol. 76, 2017, pp. 21:1–21:23.
[23] S. Hossein, A. Ghahremannezhad, and H. Kim, "On dynamic thermal conditions in mixed-criticality systems," in *RTAS*, 2020, pp. 336–349.
[24] M. Ansari, S. Safari, S. Yari, P. Gohari, H. Khdr, M. Shafique, J. Henkel, and A. Ejlali, "Thermal-aware standby-sparing technique on heterogeneous real-time embedded systems," *TETC*, pp. 1–1, 2021.