Thermal Aware System-Wide Reliability Optimization for Automotive Distributed Computing Applications

Ajinkya S. Bankar, Shi Sha, *Senior Member, IEEE*, Janki Bhimani, *Member, IEEE*, Vivek Chaturvedi, *Member, IEEE*, and Gang Quan, *Senior Member, IEEE*

***Abstract*—As the automotive industry is shifting the paradigm towards autonomous driving, safety guarantee has become a paramount consideration. Temperature plays a key role in the system-wide reliability of the electronic control systems (ECS) used in the automotive. A vehicle is usually subjected to harsh temperature conditions from its operating environment. The increasing power density of IC chips in the ECS further exacerbates the operating temperature and thermal gradient condition on the chip, thereby significantly impacting the vehicle's reliability. In this paper, we study how to map a periodic distributed automotive application on a heterogeneous multiple-core processing architecture with temperature and system-level reliability issues in check. We first present a mathematical programming model to bound the peak operating temperature for the ECS. Then we propose a more sophisticated approach based on the genetic algorithm to effectively bound the peak temperature and optimize the system-wide reliability of the ECS by maximizing its mean-time-to-failure (MTTF). To this end, we present an algorithm to guarantee the peak temperature for periodic applications with variable execution times to ensure our approach's effectiveness. We also present several computationally efficient techniques for system-wide MTTF computation, which show several-order-of-magnitude speed-up over the state-of-the-art method when tested using synthetic cases and practical benchmarks.**

***Index Terms*—Automotive Reliability, System-Level MTTF, Thermal Aware, ECU, ISO 26262.**

I. Introduction

THE mainstream innovation in the automotive industry is driven by electronic systems that have transformed automobiles from a mechanical-only system to a sophisticated network of embedded systems. Utilizing innovative electronics technologies provide safer integration systems, which reduce human failure in driving, e.g., from traditional GPS positioning to inter-vehicle communication, from the traditional CAN bus to Automotive Ethernet and Media Oriented Serial Transport [1]. Meanwhile, consolidating auto-electronics and enhancing ECS reliability becomes more important as endorsed by ISO-26262 standard [2] because more and more life-critical control decisions are made electronically on the roads.

Temperature plays a key role in terms of reliability for automotive systems. Among various failure mechanisms in an automotive ECS, the temperature-induced fault can reach as high as 55% [3]. The aggressive scaling of transistor sizes increases the chip power density and runtime temperature, and the ever-increasing computing complexity dramatically exaggerates the chip thermal gradient, which could harm the ECS reliability [4]. Moreover, the vehicles undergo stringent environmental conditions with the high temperature of ECUs reaching from 90 °C to 150 °C [5], which accelerates the aging or wear-out due to thermal-induced failure phenomena such as Electromigration (EM), Time-Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI), and Thermal Cycling (TC) [6]. Also, exceeding the temperature threshold may cause the ECU to enter the thermal emergency situation and be shut down automatically during runtime [7], which may cause catastrophic consequences. Therefore, the runtime temperature should be carefully managed for reliability concerns to meet the safety requirements for the automotive, e.g., the international standard ISO 26262.

Designers may use various active and passive cooling methods to restrict the ECU temperature, such as convection heat sinks and spreaders in low-end ECUs [8] or passive cooling for high-end ECUs, e.g., Tesla Model 3 [9]. However, relying solely on a more powerful mechanical cooling system cannot solve the thermal crisis in vehicle ECS as they are ineffective in dealing with the localized thermal hotspots due to non-uniform power densities on these ICs. In particular, for the emerging 3D ICs that are a promising alternative for addressing the rapidly growing computation/storage needs of Artificial Intelligence applications [10]–[12], the thermal problems become even more substantial, and the existing active and passive cooling techniques utilized in modern automotives are powerless [13], [14]. Moreover, the design of the mechanical cooling system requires

Manuscript received 19 February 2021; revised 15 October 2021 and 7 March 2022; accepted 2 June 2022. Date of publication 24 June 2022; date of current version 17 October 2022. This work was supported by the Dissertation Year Fellowship of the University Graduate School, Florida International University. The review of this article was coordinated by Prof. Sohel Anwar. *(Corresponding author: Ajinkya S. Bankar.)*

Ajinkya S. Bankar and Gang Quan are with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33172 USA (e-mail: abank013@fiu.edu; gaquan@fiu.edu).

Shi Sha is with the College of Business and Engineering, Wilkes University, Wilkes-Barre, PA 18766 USA (e-mail: shi.sha@wilkes.edu).

Janki Bhimani is with the Knight Foundation School of Computing and Information Sciences, Florida International University, Miami, FL 33199 USA (e-mail: janki.bhimani@fiu.edu).

Vivek Chaturvedi is with the Department of Computer Science And Engineering, Indian Institute of Technology, Palakkad, Kerala 678557, India (e-mail: vivek@iitpkd.ac.in).

Digital Object Identifier 10.1109/TVT.2022.3185978

thermal characteristics of the ECS under different workloads and environments to be incorporated in the design process for better design trade-offs. Hence, thermal-aware resource management is a critical part of the solution to minimize the temperature impact for ECS.

In this paper, we study the problem of how to map periodic distributed automotive applications on a heterogeneous ECU architecture to mitigate thermal-induced system-level reliability degradations. In particular, we aim at electromigration-aware MTTF maximization and latency minimization without violating the temperature threshold on distributed heterogeneous ECUs. Our proposed optimization framework can also be readily utilized to improve other failure types. Overall, we have made the following contributions:

1) We develop a simple thermal-aware mathematical programming-based method to bound the peak temperature when mapping periodic vehicle applications on a heterogeneous ECS architecture.
2) We find that existing approaches cannot safely bound the peak temperature when tasks' actual execution times vary from their Worst-Case Execution Times (WCETs). To this end, we propose a computation-efficient thermal bounding method that can be safely adopted in practical cases with execution time variance. Further, we formally prove a series of supportive lemmas and theorems in this work to ensure its effectiveness.
3) We improve the system-level MTTF computing efficiency. Our proposed formula can be used for problems with higher design complexity and more optimization dimensions than [15], [16]. Further, we extend our MTTF calculation approach to the more general case (when different ECUs may have different aging rates).
4) We incorporate our thermal bounding method and system-level MTTF estimation into a genetic algorithm-based approach to map the periodic vehicle applications on ECS. Based on both synthesized test cases and real-life benchmarks, the experimental results show that the proposed approach can achieve $54\times$ to $110\times$ speed-up over the state-of-the-art approach [16].

II. Related Work

An automotive ECS carries safety-critical applications, which demands both stringent real-time responsiveness and high-reliability requirements. In this paper, we improve the optimization framework that can simultaneously deal with ECS temperature limitations, real-time constraints, and reliability requirements.

The thermal safeness ensured in this work is essential for achieving a sustainable running ECS, which prevents automatically triggered power gating, clock throttling, or shutting down ECUs when thermal emergency occurs. Some reliability optimization approaches have been proposed, with no issues of temperature on computing hardware taken into consideration, e.g. [17]–[22]. So, they are not always feasible in thermal-constrained platforms. For example, Huang *et al.* [23] optimized the energy, reliability, and makespan jointly on directed acyclic graphs (DAGs); however, they did not consider the thermal effects on frequency-dependent transient faults in the reliability framework. Other existing approaches employ simplified power/thermal models, which can either cause thermal threshold violation during runtime or pessimistically predict the system performance. For example, Lee *et al.* [24] utilized constant maximum powers for peak temperature identification on consecutive execution phases, so its predicted peak temperature can overly constrain its actual resource utilization. Moreover, the solution in [24] does not optimize the reliability.

To facilitate more rigorous analytical thermal analyses, some more sophisticated algorithms have been developed to identify the peak temperature [25], [26]. However, their computational cost can be prohibitively high when scaling the application complexity and processing platforms [25]–[27]. To reduce the peak temperature identification complexity, thermal bounding approaches [28], [29] were developed to estimate the highest possible peak temperature for given task sets with different mappings. However, as shown later in this paper, these approaches [28], [29] cannot effectively bound the peak temperature when the task execution time varies. Other works, e.g. [30], [31], use the regression-based model and practical set-up, respectively, to estimate the peak chip temperature. These solutions cannot guarantee that the actual peak temperature stays below the estimated results and are also difficult to be incorporated into an optimization framework. Hence, there is a need to develop more effective and efficient temperature-constrained methodologies that can be safely applied on ECS with additional optimization factors, such as reliability enhancement and makespan minimization.

Many works consider temperature when establishing reliability optimization frameworks [32]. Zhou *et al.* [32] studied the task scheduling problem on a heterogeneous computing platform for latency minimization problem under the peak temperature and reliability constraint. However, their reliability model, i.e., the transient fault model, is independent of temperature. They also ignored the dependency of task executions and assumed that the temperature can reach stable status instantaneously. Ergun *et al.* [33] studied the system reliability maximization problem on IoT systems without considering the timing constraints; thus, their solutions become infeasible for the latency-critical automotive applications. These methodologies fall short of effectively dealing with the thermal challenges and their impacts on system reliability and fulfill critical ECU requirements in automobile system design.

To incorporate system reliability optimization into our work, we adopt the widely used Mean-Time-To-Failure (MTTF) to determine a system's lifetime reliability and develop a fault-tolerant mechanism based on processor/system state [34]. A recent study extended the validation scope of MTTF from Electromigration to Thermomigration and stress-migration induced failure [35]. However, one primary concern is that the computational cost of MTTF calculation can be prohibitively high [15], [16], [36], [37] to obtain high accuracy.

The rest of the paper is organized as follows. We introduce the system models and formally define the problem in Section III. In Section IV, a linear mathematical programming model for

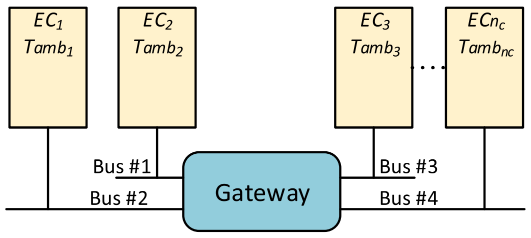


Fig. 1. The heterogeneous ECS Architecture.

a simple thermal approach is presented. In Section V, a more accurate peak temperature bounding method is proposed. In Section VII, our genetic algorithm details, experiment set-up, and results are described. At last, we conclude in Section VIII.

III. Preliminary

In this section, we first discuss the architecture and system models used in this paper. Later, we define the system-level thermal and reliability models.

A. Architecture and Application Models

In this study, we consider integrating multiple ECUs as an ECS targeting safety-critical applications. Assume an ECS consists n_c heterogeneous ECUs interconnected by a bus network through a central gateway [18], [19], [38] as $\mathbf{EC} = \{EC_1, EC_2, \ldots, EC_{n_c}\}$ in Fig. 1. Each ECU can be interfaced with various sensors and actuators. As the ECUs have different ambient temperatures as per their mounting locations in the vehicle [5], we assume different ambient temperatures for each ECU, $\mathbf{T_{amb}} = \{T_{amb_1}, T_{amb_2}, \ldots, T_{amb_{n_c}}\}$. This ambient temperature accounts for the cumulative effect of the environment temperature and the surrounding electronics/mechanical device heat transfer.

We consider periodic automotive applications as in the existing works [39]–[41], and use the Directed Acyclic Graph (DAG), $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ to model their behavior, which will be mapped on the heterogeneous architecture described above. We assume that the node-set $\mathcal{V}$ consists of total n_v nodes for sensing/control tasks in an ECS as $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{n_v}\}$. Each task may require different computation times due to the heterogeneity of the ECUs. Therefore, we construct a 2-D matrix for worst-case execution time (WCET) of the task set on different ECUs as,

$$\mathbf{W}_{n_v \times n_c} = \{w_{ik}, i = 1, \ldots, n_v;\ k = 1, \ldots, n_c\}, \tag{1}$$

where w_{ik} represents the worst-case execution time of the i-th task ($\mathcal{V}_i$) executed on the k-th ECU (EC_k). We assume that the overhead for administering task executions can be reckoned by calibrating each task's execution times, and the WCET of the tasks are not affected by the thermal profile. The edge set $\mathcal{E} = \{e_{ij}, \text{with } \mathcal{V}_i, \mathcal{V}_j \in \mathcal{V}\}$ represents the worst-case communication cost between different task nodes in a system.

B. Power and Thermal Models

The total power consumption of an ECU is a combination of dynamic power (P_d) and leakage power (P_s) [42]. The dynamic power does not depend on the transient temperature [43], but each task in an automotive system has different dynamic power based on the switching activity [24]. Therefore, to exploit this phenomenon, we consider the dynamic power of task $\mathcal{V}_i$ as, $P_d(i) = \alpha_i.V_{dd}^3$. Where α_i is the activity factor with $\alpha_i = 0$ for the idle task, and V_{dd} is the supply voltage of the ECU.

We assume that the leakage power of an ECU is linearly dependent on the thermal state [43], i.e., $P_s = (\phi_1.T + \phi_0)$, where T is the temperature of the ECU, ϕ_0, and ϕ_1 are ECU-dependent constants. Therefore, the all-inclusive total power consumption (P_{total}) of an ECU while running the task $\mathcal{V}_i$ can be expressed as

$$P_{total} = P_d(i) + P_s = V_{dd}^3 \cdot \alpha_i + (\phi_1 \cdot T + \phi_0). \tag{2}$$

We adopt the widely used RC-thermal model [24], [29], [43]–[45] to capture the ECU temperature dynamics. $T(t)$ and $P(t)$ are the transient temperature (°C) and its corresponding power consumption ($Watt$) at time instant t, respectively. Then, the transient temperature $T(t)$ follows

$$R_{th} \cdot C_{th} \cdot \frac{dT(t)}{dt} + T(t) - R_{th} \cdot P_{total}(t) = T_{amb_k}, \tag{3}$$

where R_{th}, C_{th} represent thermal-resistance (°C$/W$) and thermal-capacitance ($J/$°C). Given the equation (3), for task $\mathcal{V}_i$, we can easily identify its ending temperature $T(t_2)$ of the interval $[t_1, t_2]$ with $T(t_1)$ being the initial temperature as

$$T(t_2) = \frac{A(i)}{B} + \left(T(t_1) - T_{amb_k} - \frac{A(i)}{B}\right) e^{-B(t_2 - t_1)} + T_{amb_k}, \tag{4}$$

where $A(i) = (\phi_0 + V_{dd}^3 \cdot \alpha_i)/C_{th}$ and $B = 1/(R_{th} \cdot C_{th}) - \phi_1/C_{th}$. If the ECU executes the periodic task profile, then the stable status temperature of the ECU can be given by the following theorem (Similar proof is described in Quan *et al.* [43] and hence omitted.)

Theorem 1: Let an ECU, i.e., EC_k runs a periodic schedule with the period of t_p, starting from the ambient temperature (T_{amb_k}). Let T_L be the ending temperature of the first period and let T_{ss} be the temperature when it reaches a stable status. Then,

$$T_{ss} = T_{amb_k} + \frac{T_L - T_{amb_k}}{1 - \mathbb{K}}, \tag{5}$$

where $\mathbb{K} = exp(-B \cdot t_p)$.

C. Lifetime Reliability Model

In this paper, we focus on temperature sensitive *Electromigration* (EM) wear-out failure mechanism and estimate the system-level reliability of the ECUs as a Mean Time to Failure (MTTF). At the higher temperature and current density, the Electromigration causes displacement of the mass in the conductors of inadequate cross-section, thereby leading to the vacancies in the chip geometry, which can not be recovered. This can be correlated with the mean time to failure as [46],

$$MTTF(T) \propto A_c \cdot J^{-n} \cdot e^{\frac{E_a}{K \cdot T}}, \tag{6}$$

where A_c is a cross-section area of the conductor related constant; J is the current density in Amp/cm^2; E_a is the activation energy in *electron-volts*; K is the Boltzmann's constant; T is the *Kelvin* temperature, and $n = 2$ unless otherwise specified. From equation (6), the lifetime reliability of the device is inversely dependent on the peak temperature, and it would be improved if the peak temperature is reduced.

D. Problem Formulation

For the given automotive DAG application set $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, we seek mapping strategies on the heterogeneous architecture $\mathbf{EC}$ such that the latency is reduced (by satisfying the deadline) and system-level lifetime reliability is maximized (by optimizing the operating temperatures for ECUs judiciously). Simultaneously, we intend to bound the peak temperature to avoid the thermal emergency situation as stated before.

Problem 1: Given $\mathbf{EC}$ and DAG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, allocate all task nodes in $\mathcal{G}$ to $\mathbf{EC}$ such that (1) the peak temperature of the design (T_{peak}) is lower than the given temperature threshold ($T_{\max}$); (2) the makespan ($C_{\max}$), peak temperature, and system-level lifetime reliability ($MTTF$) of the design are optimized.

IV. Mathematical Programming Approach

Clearly, Problem 1 in Section III-D is NP-hard, and different approaches can be applied to solve it, such as mathematical programming [47], simulated annealing [16], genetic algorithm [48], etc. In what follows, we first deal with Problem 1 using the mathematical program approach, as it is a common approach that can produce the optimal solution for an optimization problem.

To satisfy the peak temperature constraint of the $\mathbf{EC}$ after partitioning task set $\mathcal{V}$, we define the decision variables x_{ik} as,

$$x_{ik} = \begin{cases} 1, & \text{if } \mathcal{V}_i \text{ is assigned to } EC_k; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Each task $\mathcal{V}_i$ must be allocated to only one processor as constrained by,

$$\sum_{k=1}^{n_c} x_{ik} = 1, \quad \forall \mathcal{V}_i \in \mathcal{V}, \forall EC_k \in \mathbf{EC}. \tag{8}$$

Let us define another decision variable σ_i as the starting time of task $\mathcal{V}_i$, then the makespan of the system is defined to be,

$$\sigma_i + \sum_{k=1}^{n_c} w_{ik} \cdot x_{ik} \le C_{\max}, \quad \forall \mathcal{V}_i \in \mathcal{V}, \forall EC_k \in \mathbf{EC}. \tag{9}$$

In the meantime, if predecessor-successor task pairs are allocated to different ECUs, then the data dependencies among them are managed by the following equation,

$$\sigma_i + w_{ik} \cdot x_{ik} + e_{ij} \cdot (x_{jl} + x_{ik} - 1) \le \sigma_j, \tag{10}$$

where EC_k and $EC_l \in \mathbf{EC}, \mathcal{V}_i$ and $\mathcal{V}_j \in \mathcal{V}, \forall e_{ij} \in \mathcal{E}$. On the other hand, the following constraints ensure the executions of two tasks allocated on the same ECU will never overlap,

$$\begin{gathered} \sigma_i + w_{ik} - \sigma_j \le \mathcal{M} \cdot (2 - x_{ik} - x_{jk}) \\ \text{OR} \\ \sigma_j + w_{jk} - \sigma_i \le \mathcal{M} \cdot (2 - x_{ik} - x_{jk}), \end{gathered} \tag{11}$$

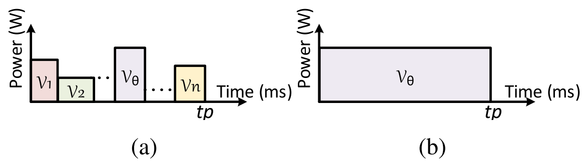


Fig. 2. (a) The WCET schedule with period t_p. (b) A hypothetical schedule with only task $\mathcal{V}_\theta$ for the entire period t_p.

where $\forall \mathcal{V}_i \ne \mathcal{V}_j \in \mathcal{V}, \ \forall EC_k \in \mathbf{EC}$, and $\mathcal{M}$ is a large positive constant.

Note that the constraints imposed by the equations (8–11) ensure the minimal makespan and task precedence, but they do not contemplate the thermal behavior in the mathematical model. From equations (3) and (4), we can observe that the temperature is a non-linear function of t and hence unsolvable by linear solvers. Therefore, we use a simple thermal approach in which each task has a constant stable status temperature irrespective of the starting temperature and thermal capacitance of the ECU [29], [49]. In particular, for any task $\mathcal{V}_j \in \mathcal{V}$, we can set $\frac{dT(t)}{dt} = 0$ in equation (3) and obtain the constant stable status temperature as $T = T_{amb_k} + R_{th} \cdot P_{total}$. With this knowledge, we can find the stable status temperature for any $\mathcal{V}_i \in \mathcal{V}$ on EC_k as

$$T_k^*(i) = \frac{A(i)}{B} + T_{amb_k}, \quad \forall T_{amb_k} \in \mathbf{T_{amb}}. \tag{12}$$

Therefore, the highest system-wide temperature (denoted as T^*) can be formulated as

$$T^* = \max_{i,k}(T_k^*(i)), \quad \forall \mathcal{V}_i \text{ allocated on } EC_k. \tag{13}$$

Further, we have the following theorem to ensure that T^* bounds the peak temperature for any resultant task allocation.

Theorem 2: For any mapping of $\mathcal{V}$ to $\mathbf{EC}$, with periodic execution, the resultant highest system temperature never exceeds T^*.

Proof: Suppose EC_k periodically executes n tasks $\{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ at a period t_p, whose dynamic powers are $\{P_d(1), \ldots, P_d(n)\}$ as shown in Fig. 2(a). Let task $\mathcal{V}_\theta$ consume the highest dynamic power among all tasks allocated on EC_k, i.e., $P_d(\theta) = max\{P_d(1), \ldots, P_d(n)\}$. Then, for all $\mathcal{V}_i$ allocated on EC_k, since $P_d(\theta) \ge P_d(i)$, we have

$$T_k^*(\theta) \ge T_k^*(i). \tag{14}$$

Using a hypothetical schedule that keeps at a constant power consumption $P_d(\theta)$ in one period as Fig. 2(b) can bound the peak temperature of the schedule shown in Fig. 2(a) because both the execution time and power consumption for all $\mathcal{V}_i$ are not larger than $\mathcal{V}_\theta$ on EC_k in Fig. 2(b). When applying equation (14) on all ECUs, $T^* = \max_{i,k}(T_k^*(i))$ can effectively bound the system-wide peak temperature. □

With Theorem 2, we can readily add a peak temperature-related constraint as follows,

$$T^* \le T_{\max}. \tag{15}$$

Also, we need to incorporate the minimization of peak temperature into the design objective. Note that to reduce the makespan and to reduce the peak temperature are two conflicting design objectives. How to deal with multi-criteria optimization in a more sophisticated manner is not the focus of this paper. Instead, we use a weighted sum as our optimization objective as follows,

$$Max : \mathbb{W}_1 \cdot \frac{C_s - C_{\max}}{C_s} + \mathbb{W}_2 \cdot \frac{T_{\max} - T^*}{T_{\max}}, \quad (16)$$

where $\mathbb{W}_1, \mathbb{W}_2$ are the weights and $\mathbb{W}_1 + \mathbb{W}_2 = 1$. C_s is the minimal makespan that a task set can complete on a given ECS without considering its temperature constraint, reliability optimization, etc. The mathematical program minimizes the makespan of the application, but it can be readily adopted to use makespan as a strict constraint according to the application's real-time requirement.

The mathematical programming approach presented in this section has three challenges. First, it is well known that the proposed method is NP-hard in nature. Second, to identify the peak temperature, this approach adopts a strategy that assumes an ECU can immediately reach its stable status [49], which is pessimistic especially when the task execution time is very short (from tens of microseconds to several milliseconds [29]). Third, we can't incorporate system-level lifetime reliability in the mathematical programming framework due to the non-linearity. Therefore, we resort to the meta-heuristic genetic algorithm [48] to solve the Problem 1. In the following sections, we first study how to capture the peak temperature under a task mapping configuration, and then, we develop efficient MTTF calculations accordingly.

V. Bound the Worst-Case Peak Temperature

A key to solve Problem 1 in Section III-D is to estimate an accurate peak temperature for a given task/ECU mapping configuration. Several approaches have been proposed to estimate the peak temperature of a processor. For example, the numerical method splits each execution interval into short fragments and attempts to find the peak temperature by checking each small stretch (e.g. [50]–[52]). Also, an epoch-based peak temperature detection methodology has been proposed using a mix of analytical and numerical methods in [44] or by solving the first-order derivative on each processing core via the Newton-Raphson method in [45]. Another approach greedily searches the worst-case task arrival cases to bound the runtime peak temperature [26]. However, these approaches have very high computation complexity (e.g. [26], [50]–[52]), which makes them ineffective during the design space exploration.

For more computationally efficient approaches, besides the *simple thermal approach* stated in Section IV, Chaturvedi *et al.* [28] proposed a so-called "hypothetical step-up schedule" to bound the peak temperature for a periodic schedule. However, as shown in what follows, this approach works under the assumptions of each task instance always takes its worst-case execution times. It may fail when the task's execution time vary in runtime, which is quite normal in practical automotive scenarios [53].

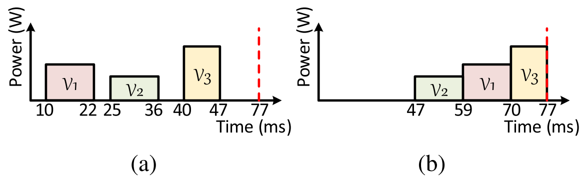


Fig. 3. An example of the step-up schedule.

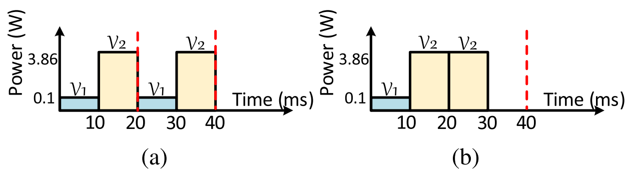


Fig. 4. A motivational example.

A. Motivation Examples

In real-time computing, it is a common practice to use the worst-case execution times to bound the longest completion time of a task set under a scheduling policy. However, we find that using the worst-case execution time-based real-time schedules (e.g. [50]–[52]) can not always identify the highest possible peak temperature during runtime. To put our discussions into perspective, assume a task set contains three periodic tasks with the same period, and one period of its schedule (based on the worst-case execution times) is depicted in Fig. 3(a). Fig. 3(b) shows its corresponding *step-up* schedule, with the same interval lengths but organized so that the dynamic power consumptions are monotonically increasing from the first interval to the last. It has been formally proved that the peak temperature of a schedule is bounded by its corresponding "step-up" schedule [28], [29]. For example, if $w_{1k} =$ 12 ms, $w_{2k} = 11$ ms, $w_{3k} = 7$ ms and $P_d(1) = 2.204$ W, $P_d(2) = 0.962$ W, $P_d(3) = 2.924$ W with a period 77 ms, using the experimental set-up in Section VII-A, the peak temperature of the step-up schedule (Fig. 3(b)) is 51.97 °C, which bounds the actual peak temperature of 51.69 °C for the schedule in Fig. 3(a). It is much smaller than the peak temperature of 125.45 °C using the simple peak temperature prediction approach of Section IV.

The example shown in Fig. 3 demonstrates that the step-up schedule can guarantee peak temperature only when all the tasks run with their worst-case execution times (WCETs). However, the temperature bound given by a step-up schedule may be violated when the tasks run with execution times lower than their WCETs, as shown in the following motivational example. Fig. 4(a) shows a step-up schedule with two tasks $\mathcal{V}_1$ and $\mathcal{V}_2$. Assume the dynamic power consumptions and the WCETs for $\mathcal{V}_1$ ($\mathcal{V}_2$ resp.) are 100 mW (3.86 W resp.) and 10 ms (10 ms resp.), with a period of 20 ms. When tasks $\mathcal{V}_1$ and $\mathcal{V}_2$ take their WCETs as Fig. 4(a) and execute long enough to reach their thermal stable status, the peak temperature is 93.61 °C. However, the actual task execution time may vary with the concurrent workload on one chip [54]. Without losing the generality, assume in a period of the stable status, task $\mathcal{V}_1$ changes its execution time between 0 and 10 ms as shown in Fig. 4(b). Then, the highest peak temperature can reach 94.02 °C in Fig. 4(b) and violate the peak temperature predicted by Fig. 4(a). This motivation example clearly shows that the step-up schedule can not effectively bound the peak temperature for a periodic schedule when task execution times are variable. Although it is possible to greedily search all possibilities of execution time combinations (e.g., in [26]) for the highest possible temperature, the computational cost could be prohibitively high when applying to multiple-ECU cases with tens to hundreds of applications. Therefore how to *efficiently* bound the worst-case peak temperature remains a problem.

In what follows, we propose a time-efficient algorithm to bound the peak temperature for variable execution time scenarios.

B. Accurate Peak Temperature Identification Algorithm

We begin with several lemmas to support the algorithm that tightly bounds the peak temperature for variable execution time scenarios and later prove a theorem to validate the proposed algorithm's effectiveness.

Lemma 1: Let EC_k run n tasks $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ consecutively with a t_p period. Let $T_m(q \cdot t_p)$ be the temperature at $t = q \cdot t_p$ when all tasks are executed with their WCETs. Then, we have $T_m(q \cdot t_p) \geq T(q \cdot t_p)$, if $T(q \cdot t_p)$ is the temperature at $t = q \cdot t_p$ when at least one task instance from $t \in [0, q \cdot t_p]$ runs with execution time less than its WCET.

Lemma 2: Let EC_k run $\mathcal{V}_i$ during an interval $t \in [t_1, t_2]$ and let EC_k's temperature at t_1 be T_1. Then EC_k's temperature monotonically increases (decreases *resp.*) within interval t, if $T_k^*(i) \geq T_1$ ($T_k^*(i) \leq T_1$ *resp.*).

As per Lemma 1, if at least one of the tasks on ECU executes shorter than its WCET, then the temperature at the end of q-th period is no larger than its WCET counterpart. From Lemma 2, we deduce that if the individual task stable status temperature is higher (lower resp.) than the starting point temperature in an interval, the ECU temperature monotonically increases (decreases resp.) when running in a constant execution mode during the interval. The detailed proofs of the lemmas are described in Appendices A and B, respectively.

With Lemma 1 and 2, an algorithm is developed to identify the peak temperature when an ECU runs a set of tasks that considers task completion time variation during the runtime.

Algorithm 1 first calculates the stable status temperature at the end of the application period, assuming that each task takes its WCET (line 1). Then, lines 2-8 iteratively identify the highest temperature of each interval in the schedule. Specifically, if a task's stable state temperature is higher than the current temperature bound (T_m), the task is taken into account to bound the possible higher peak temperature. Otherwise, the current interval is not considered in determining the peak temperature bound (T_m). Finally, the ending temperature for the last interval is output as the peak temperature (line 9). The complexity of Algorithm 1 is $O(n)$, where n is the number of tasks allocated to an ECU. Also, we formulate Theorem 3 as follows to ensure the guarantee of the peak temperature identified through Algorithm 1.

Theorem 3: Let tasks $\{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ be executed on ECU EC_k periodically with variable execution times. Then, the highest possible temperature during the execution is no more than T_m output from Algorithm 1.

Algorithm 1: Bounding the Peak Temperature with Execution Time Variance.

```
   Inputs: Task mapping based on WCET schedule of
    $EC_k$; Power/thermal parameters; Timing parameter
    matrix $\mathbf{W}_{n_v \times n_c}$;
   Output: Temperature bound $T_m$ of $EC_k$.
1: Let $T_m = T_{ss}$ in Theorem 1 for the WCET schedule;
2: for each task $\mathcal{V}_i$ allocated on $EC_k$ do
3:    Calculate $T_k^*(i)$ based on equation (12);
4:    if $T_k^*(i) > T_m$ then
5:       $T'_m$ = the ending temperature of running $\mathcal{V}_i$ with
         initial temperature $T_m$;
6:       $T_m = T'_m$;
7:    end if
8: end for
9: return $T_m$
```

This theorem is proved with the help of Lemma 1 and Lemma 2 in Appendix C. It substantiates the effectiveness of the peak temperature output from the proposed Algorithm 1. Next, we discuss our new methods to compute the system-level MTTF.

VI. Computationally Efficient System-Level MTTF Formulation

In this section, we first briefly introduce the relevant background for the state-of-the-art method that accurately calculates the system-level MTTF [16]. We then discuss the limitations of the method in [16] and propose our computationally efficient approaches to estimate the system-level MTTF.

Without losing the generality, we adopt the Weibull distribution to model the wear-out effects at the system-level [16]. The reliability of a single ECU at time instant t with temperature T is

$$\mathcal{R}(t, T) = e^{-\left(\frac{t}{\eta(T)}\right)^{\beta}}, \tag{17}$$

where $\eta(T)$ and β represent scale and slope parameters of the Weibull distribution, respectively. Then, we have

$$\eta(T) = \frac{MTTF(T)}{\Gamma\left(1 + \frac{1}{\beta}\right)}. \tag{18}$$

When executing periodic tasks, the reliability of an ECU in one period (t_p) is a function of the temperature and duration for each sub-interval ($a = 0, \ldots, s-1$) [16], which can be formulated as

$$\mathcal{R}(t_p) = e^{-\left(\sum_{a=0}^{s-1} \frac{\Delta t_i}{\eta(T_i)}\right)^{\beta}}. \tag{19}$$

Let the aging factor of the ECU [16] be

$$\mathcal{A} = \sum_{a=0}^{s-1} \frac{\Delta t_i}{\eta(T_i)}. \tag{20}$$

Then, the reliability of the ECU for q consecutive periods is

$$\mathcal{R}(q \cdot t_p) = e^{-\left(\sum_{a=0}^{q\cdot(s-1)} \frac{\Delta t_a}{\eta(T_a)}\right)^{\beta}} = e^{-\left(q\cdot\sum_{a=0}^{s-1} \frac{\Delta t_a}{\eta(T_a)}\right)^{\beta}} = e^{-(q\cdot\mathcal{A})^{\beta}}.$$

Therefore, the reliability of the system with n_c ECUs can be expressed as

$$\mathcal{R}_{system}(q \cdot t_p) = e^{-\sum_{b=1}^{n_c}(q\cdot\mathcal{A}_b)^{\beta_b}}. \tag{21}$$

As MTTF $\gg t_p$, the system-level $MTTF$ approximation can be calculated at the end of each period as

$$MTTF_{system} = \sum_{a=0}^{\infty} e^{-\sum_{b=1}^{n_c}(a\cdot\mathcal{A}_b)^{\beta_b}} \cdot t_p. \tag{22}$$

To reduce the computational cost of (22), a more efficient method, Speedup Technique-I is proposed in [16] by assuming the reliability of an ECU keeps the same for every v periods. The estimated system-level MTTF is

$$MTTF_{system}^{speed\_up-I} = \sum_{a=0}^{\infty} e^{-\sum_{b=1}^{n_c}(a.\mathcal{A}_b.v)^{\beta_b}} .t_p.v. \tag{23}$$

The accuracy of the state-of-the-art system-level MTTF in (23) depends on three parameters, i.e., the highest value of a, period t_p, and the number of periods v used to expedite the approximation. It is advisable to reach a high value of a until the MTTF is saturated, which drains a significant amount of computation time, as later observed in Section VII-B. To achieve higher computational efficiency, it is necessary to find an alternative strategy to estimate the system-level MTTF efficiently, reducing the computation time without compromising the MTTF estimation accuracy significantly.

A. A Computing Efficient Approach for MTTF Calculation

When optimizing the system design goal such as MTTF using meta-heuristics like a genetic algorithm or simulated annealing, high computation efficiency of the design objective function can enable a more effective design space exploration process. As mentioned before, the approach based on (23) is very time-consuming, which severely limits its searching scope and efficiency. Therefore, it is highly desirable that a more computing efficient approach for the MTTF calculation can be utilized in the search engine during the design space exploration process.

As the automotive applications have task interval lengths in ms to μs [18], [55], an ECU's temperature dynamics can be safely treated as a constant value in a period. This fact helps to simplify the system-level formulation of MTTF. Let $\eta_1, \eta_2, \ldots, \eta_{n_c}$ be the scale parameters of the $\mathbf{EC}$ given by (18). We assume that they depend on a constant temperature over the period, which can be calculated using the average temperature of the period in the stable status

$$T_{avg,b} = \frac{\sum_{a=0}^{s-1} \int_{l_a}^{l_{a+1}} \frac{A(a)}{B} + \left(T(t_a) - T_{amb_k} - \frac{A(a)}{B}\right)e^{-B\cdot\Delta_a} + T_{amb_k}}{t_p}, \tag{24}$$

where s represents the number of intervals in a period, $\Delta_a = (l_{a+1} - l_a)$ is the length of each interval, t_p is the period, and b is the ECU index from 1 to n_c.

To further improve the computational efficiency and safely bound the MTTF in an ECS, we take advantage of using a uniform slope parameter of the Weibull distribution on all the ECUs to expedite MTTF computations by adopting the worst-case wear-out rate on each unit. The ECS system reliability can then be formulated as

$$\mathcal{R}_{system} = e^{-\sum_{b=1}^{n_c}\left(\frac{t}{\eta_b}\right)^{\beta}}. \tag{25}$$

Based on equation (25), we can formulate the system-level MTTF as

$$MTTF_{system}^{Fast-same} = \int_0^{\infty} e^{-\sum_{b=1}^{n_c}\left(\frac{t}{\eta_b}\right)^{\beta}} dt. \tag{26}$$

By substituting $x = \sum_{b=1}^{n_c}\left(\frac{t}{\eta_b}\right)^{\beta}$ in (26), we have

$$x = t^{\beta} \cdot \sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right). \tag{27}$$

$$x^{\frac{1}{\beta}} = t \cdot \left(\sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}. \tag{28}$$

Further, we have

$$t = \frac{x^{\frac{1}{\beta}}}{\left(\sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}}, \tag{29}$$

$$dt = \frac{x^{\frac{1}{\beta}-1} \cdot dx}{\beta \cdot \left(\sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}}. \tag{30}$$

Note that $t \to 0$ leads to $x \to 0$ and $t \to \infty$ leads to $x \to \infty$. Now, we can factor out (26) and obtain our fast MTTF formula as

$$MTTF_{system}^{Fast-same} = \frac{1}{\beta \cdot \left(\sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}} \int_0^{\infty} e^{-x} \cdot x^{\frac{1}{\beta}-1} \cdot dx, \tag{31}$$

which converges to the following simple form [56],

$$MTTF_{system}^{Fast-same} = \frac{\Gamma(\frac{1}{\beta})}{\beta \cdot \left(\sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}}. \tag{32}$$

Equation (32) is independent of compute-expensive integration operations, and $\Gamma(\frac{1}{\beta})$ can be readily calculated. So, our proposed MTTF estimation method in (32) is highly computation efficient and can tightly bound the MTTF for a heterogeneous ECS. As shown in Section VII-A(B), the proposed MTTF formula in (32) can speed-up the computing efficiency of (23) by 12× for synthetic test cases and 164× for the *Real-Life* benchmark. Moreover, the proposed approximation approach can achieve an average error below 0.1% for the estimation of system-level MTTF.

B. Fast System-Level MTTF Calculation for ECUs With Different Degradation Rates

The MTTF formula proposed in Section VI-A can accurately estimate a system-level MTTF, when (i) the temperature fluctuation in one period is small enough to be considered as a constant value, e.g., when applications' periods on an ECS are as short as several μs or ms; (ii) the ECU's wear-out rate has a negligible variance compared to the worst-case wear-out rate in an ECS, e.g., when all ECUs in an ECS have similar operational status and ambient temperatures. However, according to [5], the ambient temperature of different ECUs in one vehicle could vary as much as $90°C - 150\,°\mathrm{C}$ due to different mounting locations, which may potentially cause a large wear-out rate variance within an ECS. Therefore, equation (32) could result in an inaccurate system-level MTTF estimation without an adequately justified wear-out rate.

When adopting the uniform wear-out rate MTTF formula in equation (32), results using the same slope parameter (β) of the Weibull distribution can help to save several magnitudes of the computational cost, but they may also deviate from the practical heterogeneous ECU settings significantly. Although the state-of-the-art method for system-wide MTTF calculation in equation (23) can be used to address the heterogeneity of the ECU wear-out rates, its computational cost could be prohibitively high. To this end, we use heterogeneous thermal wear-out rates of all the ECUs to construct system-wide statistical wear-out rate value, which can be safely implanted into equation (32). Specifically, let $\mathfrak{B} = \{\beta_1, \ldots, \beta_{n_c}\}$ be the slope parameters of the Weibull distribution for ECUs. Then, we implant four different statistical parameters in (32), including maximal, minimal, average, and geometric mean values, as follows:

1) Let $\beta = \beta_{\max}$, where

$$\beta_{\max} = max\{\beta_1, \ldots, \beta_{n_c}\}. \tag{33}$$

2) Let $\beta = \beta_{\min}$, where

$$\beta_{\min} = min\{\beta_1, \ldots, \beta_{n_c}\}. \tag{34}$$

3) Let $\beta = \beta_{\text{avg}}$, where

$$\beta_{\text{avg}} = average\{\beta_1, \ldots, \beta_{n_c}\} = \frac{\beta_1 + \beta_2 + \cdots + \beta_{n_c}}{n_c}. \tag{35}$$

4) Let $\beta = \beta_{geo}$, where

$$\beta_{geo} = \textit{geo mean}\ \{\beta_1, \ldots, \beta_{n_c}\} = \sqrt[n_c]{\beta_1 \cdot \beta_2 \cdot \ldots \cdot \beta_{n_c}}. \tag{36}$$

With different statistical wear-out rate approximations given by (33)-(36), we can obtain an effective slope parameter for each ECU as,

$$\eta_b = \frac{MTTF(T_{avg,b})}{\Gamma\left(1 + \frac{1}{B}\right)}, \tag{37}$$

where $b = 1, \ldots, n_c$ and $B \in \{\beta_{\max}, \beta_{\min}, \beta_{\text{avg}}, \beta_{geo}\}$. With the help of (32) we estimate the computationally efficient system-level MTTF for different wear-out rates of the $\mathbf{EC}$ as,

$$MTTF_{system}^{Fast-diff} = \frac{\Gamma(\frac{1}{B})}{B \cdot \left(\sum_{b=1}^{n_c} \left(\frac{1}{\eta_b^B}\right)\right)^{\frac{1}{B}}}. \tag{38}$$

Using the same parameter settings in Section VII-A(A), we compare the MTTF accuracy of (38) with state-of-the-art equation (23). We observe that the geometric mean ($\beta = \beta_{geo}$) generates the lowest error in average ($\approx 2\%$) among all the candidates for both synthetic test cases and practical benchmarks. Therefore, the rest of the paper uses $\beta = \beta_{geo}$ for fast system-level MTTF calculation for ECUs with different degradation rates with minimal accuracy degradation.

With a safer peak temperature bound and a more accurate and efficient MTTF formulation, we are ready to employ a genetic algorithm for Problem 1. Our genetic algorithm implementation is similar to that presented in [48], and its set-up is briefly described in the following section.

VII. Experiment Results

In this section, we present our experiments and results to validate the performance of our proposed approaches with different parameter settings.

A. Experimental Set-Up

Our genetic algorithm is set-up as follows:

- *Chromosome design:* Each chromosome represents a feasible solution to the task scheduling problem. Chromosome comprises the first part as task mapping and later as the task schedules. We randomly initialized the task mappings on the given ECUs, and initial task schedules were generated with respect to the precedence constraints of the DAG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The pool of the initial population was randomly generated with the chromosomes for the given size.
- *Fitness function:* The formulation of the fitness function is crucial to refine the ECU design alternatives, and it also impacts the solution's superiority using a metaheuristic approach. To solve Problem 1, we need to co-optimize the makespan, peak temperature, and MTTF. First, we calculated the makespan similar to that in [48]. Then, we adopted our proposed peak temperature bound in Algorithm 1 and fast system-level MTTF with the same wear-out rate together. In another experiment, we endorsed the proposed fast system-level MTTF formulation's effectiveness with different wear-out rates. Accordingly, our fitness functions are:

1) The peak temperature for both Genetic Algorithms (GAs) is calculated using our proposed Algorithm 1. The system-level MTTF values are estimated by (23) and (32) for both GAs to compare their performance. We normalized the makespan, temperature, system-level MTTF and defined the weighted fitness function as,

$$f_1 = \mathbb{P}_1 \frac{C_l - C_{\max}}{C_l} + \mathbb{P}_2 \frac{T_{\max} - T_m}{T_{\max}} + \mathbb{P}_3 \frac{MTTF_{system}}{MTTF_{\max}}, \quad (39)$$

where $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ are the weights having $\mathbb{P}_1 + \mathbb{P}_2 + \mathbb{P}_3 = 1$, and C_l is the smallest possible makespan derived from the solutions of the first generation. $MTTF_{\max}$ is a system-level maximum MTTF estimated by assuming the ambient temperature for each ECU.

2) Our experimental study also examines the efficacy of the proposed system-wide MTTF calculation for ECUs with different wear-out rates by implementing two separate GAs with the system-level MTTF calculation by (23) and (38), respectively. We used a normalized weighted sum of the makespan and system-level MTTF in the objective function as

$$f_2 = \mathbb{R}_1 \frac{C_l - C_{\max}}{C_l} + \mathbb{R}_2 \frac{MTTF_{system}}{MTTF_{\max}}, \quad (40)$$

where $\mathbb{R}_1, \mathbb{R}_2$ are the weights with $\mathbb{R}_1 + \mathbb{R}_2 = 1$ and other parameters have the same significance as in the above case-(1). The fitness functions given by equations (39) and (40) explore the design space to search the solution with minimum makespan of the application by satisfying other design constraints.

- *Parent/survivor selection:* Due to multi-objective fitness functions, we maintained the Pareto optimal front during the evolution process from generations-to-generations and used the tournament selection policy to choose the survivors for the next generation [48]. In the tournament selection, we randomly chose 10 chromosomes, sorted them based on their fitness, and picked the top 5 for crossover and mutation. The remaining 5 elements act as the non-evolved chromosomes for the next generation.
- *Crossover/mutation operators:* We used an adaptive crossover operator on the task mapping part of the survivor chromosomes. For randomly chosen two chromosomes, a crossover point is randomly selected between 1 to n_v, and the parts of the two chromosomes after it are exchanged to produce two descendants. Afterward, the crossover descendants are mutated, where any of the pre-mapped tasks in a chromosome is randomly selected, and its mapping is randomly altered to another ECU. (see [48] for more details)

We utilize both synthetic test cases and practical benchmarks to verify the performance of the proposed approaches in terms of temperature bound and system-level MTTF formulations. The parameters for the ECUs used in our experiments are described in Table I.For synthetic test cases, the task graphs were randomly generated using a Task Graph Generator [18] with *average computation cost* = 15 ms, *communication to computation ratio* = 1, and the *shape* parameter varied in the range of $[0.3, 0.9]$ to cover a wide gamut of the test cases. The dynamic power for all task nodes was chosen to be uniformly distributed in the range of $[0.1W, 3.86W]$ adhering to practical automotive systems [24]. Apart from synthetic tests, we verified our proposed formulations on two practical benchmarks, *Real-Life* [18] and *Automotive Cruise Controller (ACC)* [55], with their parameters listed in Table II.

TABLE I
ECU PARAMETERS FOR THE EXPERIMENT [24], [57]

ϕ_0(A)	ϕ_1(A/°C)	C_{th}(J/°C)	R_{th}(°C/W)	max. Power(W)
0.035	0.016	0.0454	22	3.86

TABLE II
BENCHMARK PARAMETERS

Benchmark	n_c	n_v	$\mathbf{P_d(task)}$	$\mathbf{W}_{n_v \times n_c}, \mathcal{E}$	$\mathbf{T_{amb}}$
Real-Life[‡]	16	31	[0.1W, 3.86W][¶]	[100μs, 400μs][‡]	[35°C, 45°C]
ACC[§]	4	20	[0.1W, 3.86W][¶]	Specified for each task/edge[§]	[35°C, 45°C]

[‡]described in [18], [§]described in [55], [¶]described in [24].

To achieve about 10 years of a lifetime by assuming a drive time of 2.5 hours per day for an ECU at the temperature of 80 °C [42], we used $A_c = 10^{-7} cm^2$, $J = 5.7 * 10^5$ Amp/cm^2, and $E_a = 0.48$ eV to estimate the electromigration-induced MTTF [46]. For the existing model of system-level reliability, we used $v = 100$ in equation (23) [16]. The ambient temperatures were randomly chosen in the interval [35 °C, 45 °C] to account for the variation of the ECU mounting locations if not otherwise specified.

Our experiments were lined-up in the following two major categories:

a) *Verification of different approaches to our target research problem:* For this set of experiments, we compare the effectiveness and efficiency of different approaches to our research problem, i.e., Problem 1 in Section III-D. Our experiments were conducted based both on synthetic test cases as well as practical real-life benchmarks. Specifically, for synthetic test cases, we adopted an ECS containing 3 ECUs with 10, 15, 20, and 25 tasks to simplify tracing parameter trade-offs. A sufficiently high-temperature threshold $T_{\max}$ of 150 °C was set because the *simple thermal approach* can lead to a very high peak temperature and the mathematical solver is unable to produce a feasible solution under a tightly constrained $T_{\max}$ otherwise.
In our genetic algorithm implementation, the initial solution pool was chosen to be 500 random samples, and the genetic algorithm was progressed for 400 generations, thereby terminating with Pareto optimal front solution in the end. The uniform worst-case slope parameter of the Weibull distribution was chosen to be $\beta = 2$ [16]. For both synthetic test cases and benchmarks, without losing

TABLE III
AVERAGE CPU TIME

Num. of tasks	10	20	30	40	50	Real-Life	ACC
Avg. CPU time (ms) for MTTF method in [16]	44.21	48.59	50.46	52.96	56.40	685.156	49.84
Avg. CPU time (ms) for proposed MTTF method	3.609	3.812	3.906	4.063	4.188	4.17	3.859

TABLE IV
AVERAGE MTTF ERROR PERCENTAGE

Num. of tasks	10	20	30	40	50	Real-Life	ACC
Avg. error (%)	0.023	0.032	0.050	0.067	0.084	0.00033	0.077

generality, we set the weight combinations as $\mathbb{W}_1 = \mathbb{W}_2 = 1/2$ and $\mathbb{P}_1 = \mathbb{P}_2 = \mathbb{P}_3 = 1/3$, which commits an equal influence to each parameter.

The four approaches tested in this category are stated below:

- *Temperature oblivious approach (M-TO):* The mathematical programming model that minimizes the application makespan but does not consider thermal behavior as the state-of-the-art integer linear programming method in [47].
- *Simple thermal aware approach (M-STA):* The mathematical programming approach explained in Section-IV that assumes an ECU reaches its stable status temperature immediately [29], [49].
- *Accurate peak temperature identification with fast MTTF calculation (G-APTI-FMC):* The genetic algorithm-based approach bounds the peak temperature accurately with the proposed Algorithm 1 and proposed fast system-level MTTF formulation using a uniform system-wide wear-out rate as described in Section-VI-A.
- *Accurate peak temperature identification with traditional MTTF calculation (G-APTI-TMC):* The genetic algorithm-based approach that identifies the peak temperature accurately using the proposed Algorithm 1 and the state-of-the-art system-level MTTF formulation in equation (23) [16].

b) *Verification of the proposed approaches for fast system-level MTTF calculation:* For this set of experiments, we focus our study on the efficacy of the proposed approaches for fast MTTF calculation. To study the performance of our fast MTTF calculation based on the same degradation rate, i.e., based on equation (32), we assumed that all tasks were assigned to a single ECU in our generated synthesized test cases and real-life benchmarks. We compared this approach with the traditional approach (i.e., [16]), and the results for the average CPU times and error percentages by these two approaches were collected and listed in Table III and Table IV, respectively.

We then extended our experiments to more sophistic scenarios when different ECUs may have different degradation rates. To this end, we assumed the system contains 100 tasks and the number of ECUs to be 5, 10, 15, and 20. We randomly chose the slope parameter β for each ECU in the range [2, 3], which signifies the early lifetime wear-out [58]. We used the equal weights as, $\mathbb{R}_1 = \mathbb{R}_2 = 0.5$ in this set-up. The following approaches were implemented and tested:

- *Traditional MTTF calculation with different degradation rates (G-TMC-DDR):* It is a genetic programming-based approach with the objective of makespan minimization and system-level MTTF maximization, which is estimated by the existing method in equation (23) [16]. Here we used the GA with the initial solution space of 100 samples and evolution over 200 generations.
- *Fast MTTF calculation with different degradation rates for small size of GA (G-FMC-DDR-S):* It is a genetic algorithm-based approach that minimizes the makespan and maximizes the system-level MTTF estimated by the proposed formulation in Section-VI-B with an initial population pool of 100 samples, and we evolve it for 200 generations.
- *Fast MTTF calculation with different degradation rates for large size of GA (G-FMC-DDR-L):* It is also a genetic algorithm-based approach that minimizes the makespan and maximizes the system-level MTTF estimated by the proposed formulation in Section-VI-B with an initial population pool of 500 samples and evolved for 1000 generations.

The industrial-grade linear solver CPLEX 12.10.0 was used to solve the formulated mathematical programming problems of M-TO and M-STA. Whereas G-APTI-FMC, G-APTI-TMC, G-FMC-DDR, G-TMC-DDR-S, and G-TMC-DDR-L were implemented using Matlab R2020a, running on an HP Z800 server with 24 cores and 32 GB memory.

B. Experimental Results and Discussions

This section presents the experimental results for the set-up described in the previous section and discusses our findings in detail.

a) *Verification of different approaches to our target research problem:* Fig. 5 shows the results (average peak temperature, average system-level MTTF, average makespan, and average CPU time) by four different approaches, i.e., *M-TO*, *M-STA*, *G-APTI-FMC*, and *G-APTI-TMC*, as introduced above.

From Fig. 5(c), we can see that *M-TO* has the smallest makespan among all the approaches, but Fig. 5(a) and Fig. 5(b) show that *M-TO* has the highest peak temperature and the lowest system-level MTTF. This result clearly indicates that constraining the peak temperature in an automotive ECS is necessary for hardware protection and system-wide lifetime reliability enhancement. When thermal behavior is considered, the makespans for other approaches become longer since the temperature constraint limits the computational throughput on an ECU. Also, with increased thermal prediction accuracy, *G-APTI-FMC* and *G-APTI-TMC* allow to have a longer makespan but

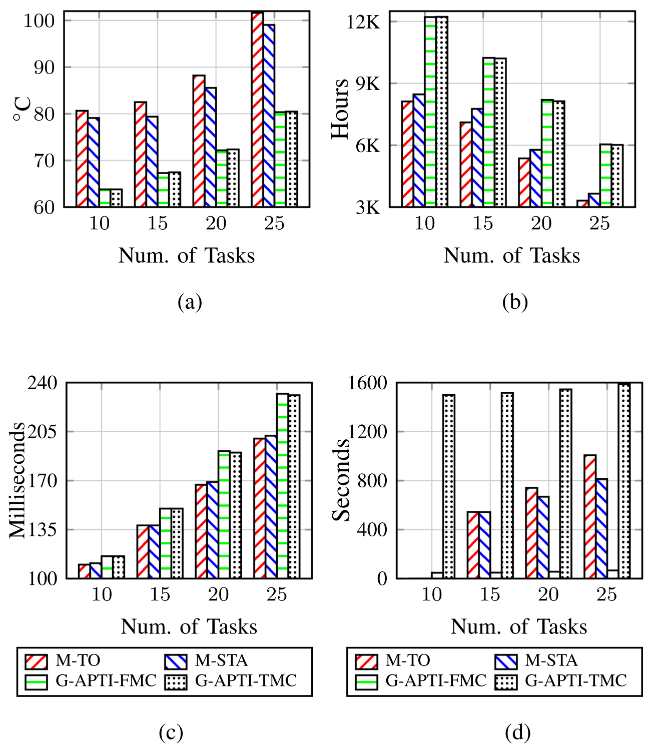


Fig. 5. Experimental results for 3-Processor system. (a) Avg. system peak-temperature. (b) Estimated avg. system-level MTTF. (c) Avg. makespan. (d) Avg. CPU time.

a much lower peak temperature and higher system-level MTTF, as clearly shown in Fig. 5(a) and 5(b).

Meanwhile, from Fig. 5(a), we observe that the average peak temperatures for *G-APTI-FMC* and *G-APTI-TMC* differ approximately in the range of 0.1 °C to 0.3 °C. Also, in Fig. 5(c), their makespans differ by 0 ms to 1 ms. Similarly, the estimated average system-level MTTF differ by 0.1%, 0.26%, 0.77%, and 0.44% respectively from 10 to 25 tasks as depicted in Fig. 5(b). But, as observed in Fig. 5(d), there is a significant difference in CPU times between the *G-APTI-TMC* and *G-APTI-FMC*. Thanks to our fast MTTF calculation methods, our approach (*G-APTI-FMC*) can achieve similar performance results (average peak temperature, average makespan, and overall average system-level MTTF) with the CPU times about 23 to 31 times lower than *G-APTI-TMC*.

In terms of peak temperature identification accuracy, if we compare the results by *G-APTI-TMC* with *M-STA* in Fig. 5(a), we can see that *M-STA* is quite pessimistic, i.e., by 15.3 °C, 11.9 °C, 13.2 °C, and 18.5 °C as the task numbers increase from 10 to 25. These results demonstrate that our proposed temperature identification methods, as shown in Algorithm 1, can greatly help *G-APTI-TMC* to obtain the solution with significantly reduced peak temperature and much improved system-level MTTF over *M-STA*, as much as 44.38%, 31.49%, 40.78%, and 64.61% as observed in Fig. 5(b).

From Fig. 5(d), we can also see that the computational cost for *M-TO* and *M-STA* increases rapidly due to the NP-hard nature of the mathematical programming technique.

TABLE V
RESULTS OF REAL-LIFE BENCHMARK

	M-TO	**M-STA**	**G-APTI-FMC**	**G-APTI-TMC**
Avg. Makespan (μs)	518	532	1167	1169
Avg. Sys. T_{peak} ($°C$)	116.1	114.6	74.5	74.4
Avg. Sys. MTTF (Hr)	1336	1389	5005	5006
Avg. CPU time (s)	237.8	683.3	135.5	26001.2

TABLE VI
RESULTS OF ACC BENCHMARK

	M-TO	**M-STA**	**G-APTI-FMC**	**G-APTI-TMC**
Avg. Makespan (ms)	147	150	184	188
Avg. Sys. T_{peak} ($°C$)	107.4	102.7	85.3	84.9
Avg. Sys. MTTF (Hr)	2219	2209	4020	4095
CPU time (s)	259.51	123.93	79.34	1370.5

Even so, their CPU times are still lower than that by *G-APTI-TMC*. Besides their ineffectiveness in dealing with temperature issues, the results clearly show that *M-TO* and *M-STA* cannot be used for large-scale systems when the task and ECU numbers continue to grow. In the meantime, the results also highlight the need to speed-up the system-wide MTTF calculation in design space exploration.

Similar findings are observed for the two automotive benchmarks. For a practical *Real-Life* benchmark, as shown in Table V, there is an average makespan difference of $2\mu s$, an average peak temperature difference of 0.1 °C, and an MTTF difference of 1 Hr between *G-APTI-FMC* and *G-APTI-TMC*, but *G-APTI-FMC* can speed-up the system-level MTTF calculations by 191 times than *G-APTI-TMC*. On the other hand, *M-STA* is 40.1 °C higher in peak temperature and 3616 Hr lower in system-level MTTF than *G-APTI-FMC*, which underlines the important contribution of the proposed *G-APTI-FMC* for the practical systems.

In the meantime, for the *ACC* benchmark, in Table VI, we can see that our proposed approach *G-APTI-FMC* can improve the computation efficiency of *G-APTI-TMC* by over 17 times with degradation in the peak temperature and system-level MTTF no more than 0.4 °C and 75 Hr, respectively. Compared with *M-TO* (*M-STA*, resp.), *G-APTI-FMC* can improve the peak temperature and system-level MTTF with 22.1 °C (17.4 °C, resp.) and 1801 Hr (1811 Hr, resp.), respectively.

b) *Verification of the proposed approaches for fast system-level MTTF calculation:* For single ECU or ECUs with similar degradation rates, as shown in Tables III and IV, our proposed fast MTTF calculation can be extremely efficient and effective. As shown in Table III, the proposed MTTF method can speed-up the computations about 12× for synthetic test cases and 164× for the Real-Life benchmark compared with the state-of-the-art method (23). In the meantime, from Table IV, we can see that the average error is less than 0.1% for both synthetic task cases and practical automotive benchmarks.

To consider the case when different ECUs may have different degradation rates, we propose four candidate

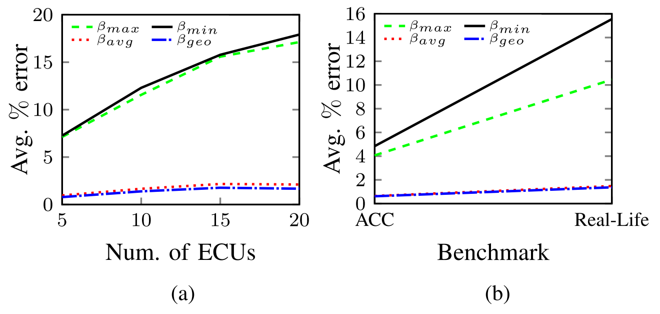


Fig. 6. Average error percentage. (a) Synthetic Test Cases. (b) Practical Benchmarks.

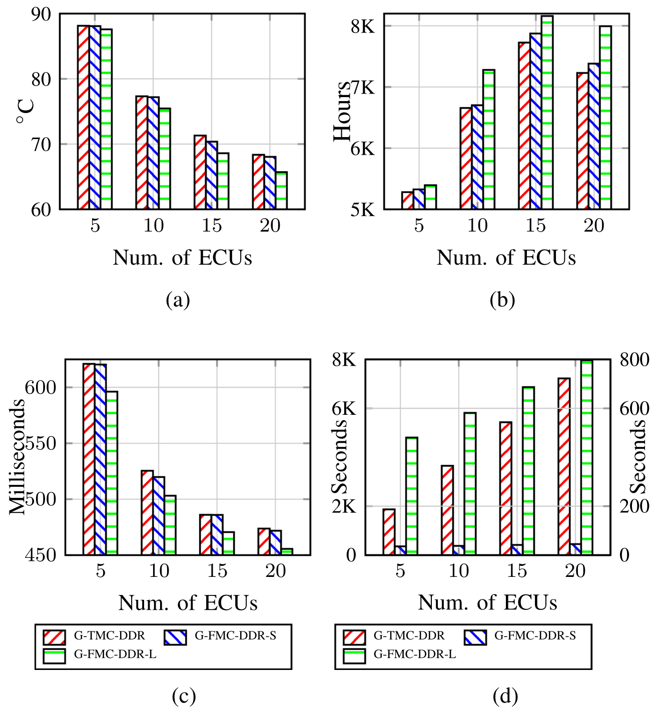


Fig. 7. Experimental results for 100-Tasks system. (a) Avg. system peak-temperature. (b) Est. avg. system-level MTTF. (c) Avg. makespan. (d) Avg. CPU time.

approaches, i.e., (33)-(36), to be used in (38). We tested these four approaches by mapping 100 tasks on 5, 10, 15, 20 ECUs in the synthesized test cases and our two practical benchmarks. We compare their performance in MTTF calculations by collecting the estimation errors normalized by the traditional one [16] (equation (23)). These results are denoted as $\beta_{\max}, \beta_{\min}, \beta_{\text{avg}}$, and β_{geo}, respectively, and shown in Fig. 6.

As shown in Fig. 6(a) and 6(b), we can see that the approach with β_{geo} has less than a 2% average error compared to the existing system-level MTTF equation (23) for both synthetic test cases and practical benchmarks. Further, we analyze the proposed methodology to compute the system-level MTTF with different degradation rates. We randomly generated the synthetic test cases for the configurations of 100-Tasks and 5, 10, 15, 20 ECUs and collected the 100 feasible test results, together with their peak temperatures, system-level MTTFs, makespans, and CPU times of the solutions, shown in Fig. 7.

In design space exploration, e.g., in a genetic algorithm or simulated annealing, evaluating the design objective function accurately and quickly is vital for the algorithm's effectiveness. To accurately evaluate the fitness of a design alternative helps to direct the search in the right direction, while the computation efficiency in fitness evaluation enables an extensive design space exploration. With the similar size of design space by *G-TMC-DDR* and *G-FMC-DDR-S*, as shown in Fig. 7(a), we observe that the average peak temperatures differ by 0.1 °C, 0.2 °C, 0.9 °C, 0.3 °C when ECU numbers increase from 5 to 20. Accordingly, their average system-level MTTFs differ by 0.8%, 0.6%, 1.9%, and 2.1%, as shown in Fig. 7(b) and their average makespans differ by 0 ms, 5 ms, 1 ms, 2 ms in Fig. 7(c). However, as observed in Fig. 7(d) (left Y-axis for the *G-TMC-DDR*, right Y-axis for the *G-FMC-DDR-S* and *G-FMC-DDR-L*), the CPU time of the *G-TMC-DDR* is 54, 95, 110, 108 times higher than that of *G-FMC-DDR-S*, respectively, as the ECU number increase from 5 to 20. Hence, with our fast MTTF calculation as formulated in (38), we can achieve a similar (a little inferior) performance in terms of the peak temperature, system-level MTTF, and makespan, but significant CPU time improvement.

The highly efficient MTTF calculation enables us to search a much larger solution space. In *G-FMC-DDR-L*, we increase both the population size and the reproduction generations in the genetic algorithm, with the total size of design space increased by 25 times. As observed in Fig. 7(d), with increased search space, the *G-FMC-DDR-L* approach requires more CPU time than *G-FMC-DDR-S*, but it helps to yield better results. For *G-FMC-DDR-L*, in Fig. 7(a), we see that the average peak temperature is reduced by 0.5 °C, 1.8 °C, 2.7 °C, 2.6 °C when ECU numbers increase from 5 to 20. Accordingly, their average system-level MTTFs improve by 113 Hr, 622 Hr, 434 Hr, 764 Hr as shown in Fig. 7(b), and Fig. 7(c), the average makespans reduced by 25 ms, 23 ms, 16 ms, 18 ms compared with *G-TMC-DDR*. In the meantime, it is worth mentioning that, by increasing the size of the initial population and number of generations for evolution in *G-FMC-DDR-L*, we can obtain solutions with better performance metrics (i.e., average peak temperature, average system-level MTTF, and average makespans). Simultaneously consuming much less CPU times, i.e., 3, 6, 7, 8 times less than the *G-TMC-DDR* according to Fig. 7(d).

We can see similar results from the two practical automotive benchmarks. As shown in Table VII, comparing *G-TMC-DDR* and *G-FMC-DDR-S*, there is not much change in the average makespan. Their peak temperatures differ slightly by 0.33 °C, and average MTTF also differ slightly by 15 Hr. However, *G-FMC-DDR-S* is faster than *G-TMC-DDR* by 100 times. In the meantime, as

TABLE VII
RESULTS OF ACC BENCHMARK FOR DIFF DEGRADATION RATE

	G-TMC-DDR	G-FMC-DDR-S	G-FMC-DDR-L
Avg. Makespan (ms)	188	188	180
Avg. Sys. T_{peak} $(°C)$	89.69	90.02	88.34
Avg. Sys. MTTF (Hr)	4149	4164	4264
Avg. CPU time (s)	1567.9	15.58	178.92

TABLE VIII
RESULTS OF REAL-LIFE BENCHMARK FOR DIFF DEGRADATION RATE

	G-TMC-DDR	G-FMC-DDR-S	G-FMC-DDR-L
Avg. Makespan (μs)	1489	1490	1463
Avg. Sys. T_{peak} $(°C)$	69.70	68.94	66.06
Avg. Sys. MTTF (Hr)	8173	8195	8867
Avg. CPU time (s)	32245.2	12.42	260.1

shown in Table VIII, the makespan, peak temperature, and MTTF estimation for *G-FMC-DDR-L* outperform *G-TMC-DDR* significantly, i.e., by 26 μs, 3.64 °C, and 694 Hr, respectively, with CPU time 124 times lower. The prominence of our fast system-level MTTF calculation helps to achieve better design space exploration due to low timing complexity in the practical systems and offers better results.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we study how to map a periodic automotive application on ECU with temperature issues taken into consideration. We first propose a mathematical programming approach to meet the peak temperature constraint while reducing the application latency. We further propose a more sophisticated genetic algorithm-based approach to deal with the peak temperature constraint and system-wide reliability optimization. To this end, we develop two key algorithms, i.e., guaranteeing the peak temperature for periodic tasks with variable execution times and the analytical approach for system-wide MTTF calculation, which can speed-up the process by several orders of magnitudes. Experimental results for two practical automotive benchmarks show that the proposed peak temperature algorithm is accurate by 17 °C and 40 °C, which results in 81% and 260% more accurate MTTF estimation. At the same time, the proposed MTTF formulation is faster by 17× and 191× with an accuracy loss of less than 0.1%, which validates the efficiency of our proposed approach.

As the automotive industry is set to be transformed into autonomous cars, the temperature will play an increasingly critical role in the safety and reliability of automotive applications, especially as more advanced Artificial Intelligence (AI)/Deep Learning technologies are incorporated into autonomous driving systems. The future work of this research includes extending the thermal awareness to more advanced AI/Deep Learning applications and more advanced hardware architecture, such as those that incorporate GPU and dedicated NPU hardware units that can accommodate such applications.

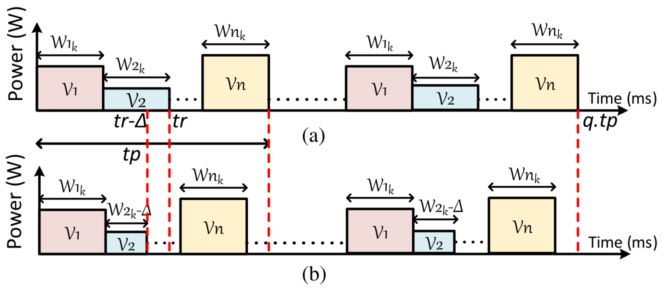


Fig. 8. (a) The WCET schedule with an ending temperature of $T_m(q \cdot t_p)$ in the stable status. (b) The schedule with a varied execution time with an ending temperature of $T(q \cdot t_p)$ in the stable status.

APPENDIX A
PROOF OF LEMMA 1

Lemma 1: Let an ECU EC_k run n tasks $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n$ consecutively with a period of t_p. Let $T_m(q \cdot t_p)$ be the temperature at $t = q \cdot t_p$ when all tasks are executed with their WCETs. Then, $T_m(q \cdot t_p) \geq T(q \cdot t_p)$, if $T(q \cdot t_p)$ is the temperature at $t = q \cdot t_p$ when at least one task instance from $t \in [0, q \cdot t_p]$ runs with execution time less than its WCET.

Proof: Let's consider an arbitrary schedule with n tasks and period t_p as shown in Fig. 8. Let $T_m(q \cdot t_p)$ and $T(q \cdot t_p)$ be the temperature at $t = q \cdot t_p$ in Fig. 8 a and Fig. 8 b, respectively. In Fig. 8 a, all the tasks take their respective WCETs. In Fig. 8 b, the execution time of task $\mathcal{V}_2$ is reduced by Δ compared to its WCET. Therefore, the starting time of the remaining tasks in the scheduling sequence remains the same or is shifted to the left by Δ amount of time. From these two figures, we indeed find that both schedules are identical up-to-time instant $(t_r - \Delta)$, so as their temperature, $T(t_r - \Delta) = T'(t_r - \Delta)$. Let $T(t_r)$ and $T'(t_r)$ be the starting temperatures for the remaining part of the schedules, respectively. The schedule in Fig. 8 a has a higher accumulated computing time at $(q \cdot t_p)$ than the schedule in Fig. 8 b, and the workload in Fig. 8 b is moved away from $t = q \cdot t_p$. Based on *Lemma 3* of Schor *et al.* [59], we conclude that $T_m(q \cdot t_p) \geq T(q \cdot t_p)$. □

APPENDIX B
PROOF OF LEMMA 2

Lemma 2: Let an ECU EC_k run $\mathcal{V}_i$ during an interval $t \in [t_1, t_2]$ and let EC_k's temperature at t_1 be T_1. Then, EC_k's temperature monotonically increases (decreases *resp.*) within interval t, if $T_k^*(i) \geq T_1$ ($T_k^*(i) \leq T_1$ *resp.*).

Proof: From equation (3), we can infer that when EC_k reaches its stable state temperature, equation $T_k^*(i) = R_{th} \cdot P_{total}(i) + T_{amb_k}$ holds, since $\frac{dT(t)}{dt} = 0$. From equation (3), we can also infer that

$$\begin{aligned} \frac{dT_1}{dt} &= \frac{-T_1 + R_{th} \cdot P_{total}(i) + T_{amb_k}}{R_{th} \cdot C_{th}} \\ &= \frac{-T_1 + T_k^*(i)}{R_{th} \cdot C_{th}}. \end{aligned} \tag{41}$$

Since $T_k^*(i) \geq T_1$, we have $\frac{dT_1}{dt} \geq 0$. Then, EC_k's temperature monotonically increases and vice-versa. □

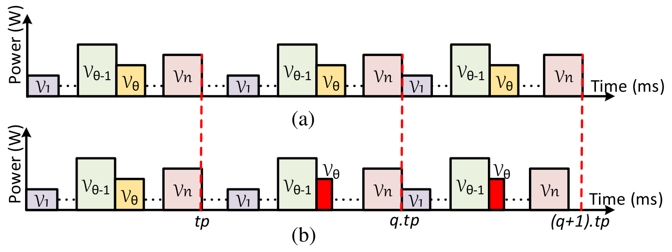


Fig. 9. (a) The WCET schedule. (b) The schedule with a varied execution time of tasks $\mathcal{V}_\theta$ in a stable state.

Appendix C Proof of Theorem 3

Theorem 3: Let tasks $\{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ be executed on ECU EC_k periodically with variable execution times. Then, the highest possible temperature during the execution is no more than T_m output from Algorithm 1.

Proof: Let a baseline schedule contain n tasks $\{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ with period t_p and each task takes its WCET, as shown in Fig. 9 a. Without losing generality, we assume Fig. 9 b contains all the same intervals as Fig. 9 a, except task $\mathcal{V}_\theta$ whose execution time is reduced by Δ in the q-th period.

Let $T_W(t)$ and $T_A(t)$ be the temperatures at the instance of t for Fig. 9 a and Fig. 9 b, respectively. Based on Lemma 1, at $t = (q \cdot t_p)$, we can infer that the temperature of Fig. 9 b is no larger than Fig. 9 a, so we have

$$T_W(q \cdot t_p) \geq T_A(q \cdot t_p). \tag{42}$$

Now, consider a period from $t = (q \cdot t_p)$ to $t = ((q+1) \cdot t_p)$. Starting from $t = (q \cdot t_p)$ let the temperature output based on Algorithm 1 be $\widetilde{T}_W(i)$ after going through task $\mathcal{V}_i$ in the loop (line 2-7) in Algorithm 1. Then, based on Lemma 2, $\widetilde{T}_W(i)$ must be equal or higher than the temperature when completing $\mathcal{V}_i$ with the schedule in Fig. 9 b. Meanwhile, the starting temperature at each interval considered in Algorithm 1 is no lower than that of $T_A(q \cdot t_p)$. Therefore, $\widetilde{T}_W(i)$ monotonically increases to T_m in each valid interval as per Algorithm 1.

On the other hand, for the schedule in Fig. 9 b, if task $\mathcal{V}_i$ has the stable state temperature lower than $T_A(q \cdot t_p)$, the temperature will be lowered. Therefore, we certainly find that at any time instant, the temperature predicted by Algorithm 1 (i.e., T_m) is higher than its periodic counterpart with varied execution times of the tasks. □

Acknowledgment

The authors would like to thank anonymous reviewers for their comments and suggestions which contributed significantly to improving quality of this paper.

References

[1] J. Scheuring and U. Bernhard, "Time-sensitive networking in the automotive industry," *ATZelectronics Worldwide*, vol. 15, no. 9, pp. 54–58, 2020.
[2] *Road Vehicles-Functional Safety*, ISO Standard 26262, 2018.
[3] ZVEI-German Electrical and Electronic Manufacturers' Association, "Handbook for robustness validation of automotive electrical/electronic modules", Frankfurt, Germany, Tech. Rep., 2013.
[4] S. Hamdioui, D. Gizopoulos, G. Guido, M. Nicolaidis, A. Grasset, and P. Bonnot, "Reliability challenges of real-time systems in forthcoming technology nodes," in *Proc. IEEE Des., Automat. Test Europe Conf. Exhib.*, DATE, 2013, pp. 129–134.
[5] M. Krüger, S. Straube, A. Middendorf, D. Hahn, T. Dobs, and K. D. Lang, "Requirements for the application of ECUs in e-mobility originally qualified for gasoline cars," *Microelectronics Rel.*, vol. 64, pp. 140–144, 2016.
[6] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. Rosing, "WARM: Workload-aware reliability management in linux/android," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1557–1570, Sep. 2017.
[7] S. Pagani, P. D. Manoj, A. Jantsch, and J. Henkel, "Machine learning for power, energy, and thermal management on multicore processors: A survey," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 1, pp. 101–116, Jan. 2020.
[8] H. Martin, G. Przemyslaw, W. Bernhard, and R. Sven, "Affordable and safe high performance vehicle computers with ultra-fast on-board ethernet for automated driving," *Adv. Microsystems Automot. Appl.*, pp. 56–68, 2019.
[9] J. Korta, P. Skruch, and K. Holon, "Reliability of automotive multidomain controllers: Advancements in electronics cooling technologies," *IEEE Veh. Technol. Mag.*, vol. 16, no. 2, pp. 86–94, Jun. 2021.
[10] W. Lu, P.-T. Huang, H.-M. Chen, and W. Hwang, "An energy-efficient 3D cross-ring accelerator with 3D-SRAM cubes for hybrid deep neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 4, pp. 776–788, Dec. 2021.
[11] K. Ueyoshi et al., "QUEST: Multi-purpose log-quantized DNN inference engine stacked on 96-MB 3-D SRAM using inductive coupling technology in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 186–196, Jan. 2019.
[12] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 380–392, Jun. 2016.
[13] J.-H. Han, K. Torres-Castro, R. E. West, N. Swami, and M. Stan, "Thermal analysis of microfluidic cooling in processing-in-3D-stacked memory," in *Proc. IEEE 22nd Int. Conf. Thermal, Mech. Multi-Physics Simul. Experiments Microelectronics Microsystems*, EuroSimE, 2021, pp. 1–6.
[14] S. Wang, Y. Yin, C. Hu, and P. Rezai, "3D integrated circuit cooling with microfluidics," *Micromachines*, vol. 9, no. 6, 2018, Art no. 287.
[15] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu, "Improving system-level lifetime reliability of multicore soft real-time systems," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 25, no. 6, pp. 1895–1905, Jun. 2017.
[16] H. Lin, Y. Feng, and X. Qiang, "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," in *Proc. IEEE Des., Automat. Test Europe Conf. Exhib.*, 2009, pp. 51–56.
[17] L. Niu, "Reliability-aware energy-efficient scheduling for (m,k)-Constrained real-time systems through shared time slots," *Microprocessors Microsystems*, vol. 77, 2020, Art. no. 103110.
[18] G. Xie, G. Zeng, Y. Liu, J. Zhou, R. Li, and K. Li, "Fast functional safety verification for distributed automotive applications during early design phase," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4378–4391, May 2018.
[19] G. Xie et al., "Reliability enhancement toward functional safety goal assurance in energy-aware automotive cyber-physical systems," *IEEE Trans. Ind. Inform.*, vol. 14, no. 12, pp. 5447–5462, Dec. 2018.
[20] A. Taherin, M. Salehi, and A. Ejlali, "Reliability-aware energy management in mixed-criticality systems," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 3, pp. 195–208, Jul.–Sep. 2018.
[21] M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 813–825, Mar. 2017.
[22] D. Zhu, "Reliability-aware dynamic energy management in dependable embedded real-time systems," *ACM Trans. Embedded Comput. Syst.*, vol. 10, no. 2, pp. 1–27, 2010.
[23] J. Huang, R. Li, X. Jiao, Y. Jiang, and W. Chang, "Dynamic DAG scheduling on multiprocessor systems: Reliability, energy, and makespan," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3336–3347, Nov. 2020.
[24] Y. Lee, H. S. Chwa, K. G. Shin, and S. Wang, "Thermal-aware resource management for embedded real-time systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2857–2868, Nov. 2018.

[25] I. Ukhov, P. Eles, and Z. Peng, "Temperature-centric reliability analysis and optimization of electronic systems under process variation," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 23, no. 11, pp. 2417–2430, Nov. 2015.

[26] L. Schor, I. Bacivarov, H. Yang, and L. Thiele, "Worst-case temperature guarantees for real-time applications on multi-core systems," in *Proc. IEEE 18th Real-Time Embedded Technol. Appl. Symp.*, 2012, pp. 87–96.

[27] A. L. Da Silva, A. L. Del Mestre Martins, and F. G. Moraes, "Fine-grain temperature monitoring for many-core systems," in *Proc. 32nd Symp. Integr. Circuits Syst. Des.*, 2019, pp. 1–6.

[28] V. Chaturvedi, H. Huang, S. Ren, and G. Quan, "On the fundamentals of leakage aware real-time DVS scheduling for peak temperature minimization," *J. Syst. Architecture*, vol. 58, no. 10, pp. 387–397, 2012.

[29] S. Sha, A. S. Bankar, X. Yang, W. Wen, and G. Quan, "On fundamental principles for thermal-aware design on periodic real-time multi-core systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, no. 2, pp. 1–23, Feb. 2020.

[30] K. C. Chen, H. W. Tang, Y. H. Liao, and Y. C. Yang, "Temperature tracking and management with number-limited thermal sensors for thermal-aware NoC systems," *IEEE Sensors J.*, vol. 20, no. 21, pp. 13018–13028, Nov. 2020.

[31] M. Sojka, O. Benedikt, Z. Hanzalek, and P. Zaykov, "Testbed for thermal and performance analysis in MPSoC systems," in *Proc. IEEE 15th Conf. Comput. Sci. Inf. Syst.*, FedCSIS, 2020, pp. 683–692.

[32] J. Zhou et al., "Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms," *J. Syst. Softw.*, vol. 133, pp. 1–16, 2017.

[33] K. Ergun et al., *RelIoT: Reliability Simulator for IoT Networks*. W. Song, K. Lee, Z. Yan, L.-J. Zhang, and H. Chen, Eds., Cham, Switzerland:Springer International Publishing, 2020.

[34] M. I. Bin Bandan, S. Bhattacharjee, S. K. Jali, and D. K. Pradhan, "Instantaneous mean-time-to-failure (MTTF) estimation for checkpoint interval computation at run time," *Microelectronics Rel.*, vol. 98, pp. 69–77, 2019.

[35] K. N. Tu and A. M. Gusak, "A unified model of mean-time-to-failure for electromigration, thermomigration, and stress-migration based on entropy production," *J. Appl. Phys.*, vol. 126, no. 7, 2019, Art. no. 075109.

[36] S. S. Sahoo, B. Veeravalli, and A. Kumar, "CL(R) early: An early-stage DSE methodology for cross-layer reliability-aware heterogeneous embedded systems," in *Proc. 57th ACM/IEEE Des. Automat. Conf.*, 2020, pp. 1–6.

[37] H. Salamy, "Energy-aware schedules under chip reliability constraint for multi-processor systems-on-a-chip," *J. Circuits, Syst. Comput.*, vol. 29, no. 9, 2020, Art. no. 2050135.

[38] G. Xie, K. Yang, H. Luo, R. Li, and S. Hu, "Reliability and confidentiality co-verification for parallel applications in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1353–1368, Jun. 2021.

[39] K.-J. Lee, J.-W. Kim, H.-J. Chang, and H.-S. Ahn, "Mixed harmonic runnable scheduling for automotive software on multi-core processors," *Int. J. Automot. Technol.*, vol. 19, pp. 323–330, 2018.

[40] S. Bateni and C. Liu, "Predictable data-driven resource management: An implementation using autoware on autonomous platforms," in *Proc. IEEE Real-Time Syst. Symp.*, RTSS, 2019, pp. 339–352.

[41] M. Li, J. Gao, L. Zhao, and X. Shen, "Adaptive computing scheduling for edge-assisted autonomous driving," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5318–5331, Jun. 2021.

[42] A. S. Bankar, S. Sha, V. Chaturvedi, and G. Quan, "Thermal aware lifetime reliability optimization for automotive distributed computing applications," in *Proc. IEEE 38th Int. Conf. Comput. Des.*, ICCD, Hartford, CT, USA, 2020, pp. 498–505.

[43] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Trans. Ind. Inform.*, vol. 6, no. 3, pp. 329–339, Aug. 2010.

[44] Q. Han, M. Fan, O. Bai, S. Ren, and G. Quan, "Temperature-constrained feasibility analysis for multicore scheduling," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 12, pp. 2082–2092, Mar. 2016.

[45] S. Pagani, J. J. Chen, M. Shafique, and J. Henkel, "MatEx: Efficient transient and peak temperature computation for compact thermal models," in *Proc. IEEE Des., Automat. Test Europe Conf. Exhib., DATE*, 2015, pp. 1515–1520.

[46] J. R. Black, "Electromigration—A brief survey and some recent results," *IEEE Trans. Electron Devices*, vol. 16, no. 4, pp. 338–347, Apr. 1969.

[47] A. Ait El Cadi, O. Souissi, R. Ben Atitallah, N. Belanger, and A. Artiba, "Mathematical programming models for scheduling in a CPU/FPGA architecture with heterogeneous communication delays," *J. Intell. Manuf.*, vol. 29, no. 3, pp. 629–640, 2018.

[48] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *J. Parallel Distrib. Comput.*, vol. 70, no. 1, pp. 13–22, 2010.

[49] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta, "Processor speed control with thermal constraints," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 56, no. 9, pp. 1994–2008, Sep. 2009.

[50] K. Skadron, K. Sankaranarayanan, S. Velusamy, D. Tarjan, M. R. Stan, and W. Huang, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Architecture Code Optim.*, vol. 1, no. 1, pp. 94–125, 2004.

[51] S. Sharifi, D. Krishnaswamy, and T. S. Rosing, "PROMETHEUS: A. proactive method for thermal management of heterogeneous MPSoCs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 7, pp. 1110–1123, Jul. 2013.

[52] R. Rao and S. Vrudhula, "Fast and accurate prediction of the steady-state throughput of multicore processors under thermal constraints," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1559–1572, Oct. 2009.

[53] M. Faizan and A. S. Pillai, "Dynamic task allocation and scheduling for multicore electronics control unit (ECU)," in *Proc. IEEE 3rd Int. Conf. Electron., Commun. Aerosp. Technol.*, ICECA, 2019, pp. 821–826.

[54] B. Andersson, H. Kim, D. D. Niz, M. Klein, R. R. Rajkumar, and J. Lehoczky, "Schedulability analysis of tasks with corunner-dependent execution times," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 3, May 2018, Art. no. 2050135.

[55] S. K. Roy, R. Devaraj, A. Sarkar, S. Sinha, and K. Maji, "Optimal scheduling of precedence-constrained task graphs on heterogeneous distributed systems with shared buses," in *Proc. IEEE 22nd Int. Symp. Real-Time Distrib. Comput.*, ISORC, 2019, pp. 185–192.

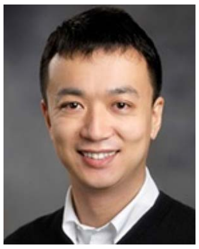
[56] F. Qi and C. P. Chen, "A complete monotonicity property of the gamma function," *J. Math. Anal. Appl.*, vol. 296, no. 2, pp. 603–607, 2004.

[57] Y. Lee, E. Kim, and K. G. Shin, "Efficient thermoelectric cooling for mobile devices," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Des.*, ISLPED, 2017, pp. 1–6.

[58] S. Speaks, "Reliability and MTBF overview" Vicor Reliability Engineering, Tech. Rep., 2014.

[59] L. Schor, H. Yang, I. Bacivarov, and L. Thiele, "Worst-case temperature analysis for different resource models," *IET Circuits, Devices Syst.*, vol. 6, no. 5, pp. 297–307, 2012.

Ajinkya S. Bankar received the B.E. and M.E. degrees from Savitribai Phule Pune University, Baramati, India, in 2010 and 2013, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA, in 2022. From 2013 to 2018, he was an Assistant Professor with the Vidya Pratishthan's Institute of Engineering and Technology, Savitribai Phule Pune University. He is currently a Data Scientist with Crowley, Jacksonville, FL, USA. His research interests include advanced real-time computing system design, power/thermal aware design, sensitivity analysis of deep neural networks, and machine learning.



Shi Sha (Senior Member, IEEE) received the B.S. degree in electrical engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, the M.S. degree in telecommunications systems management from Murray State University, Murray, KY, USA, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA. He is currently an Assistant Professor with the College of Business and Engineering, Wilkes University, Wilkes-Barre, PA, USA. His research interests include embedded machine learning, real-time scheduling, and high-performance/low-power computing.

Janki Bhimani (Member, IEEE) is an Assistant Professor at Florida International University, Miami. She received her Ph.D. and M.S. degrees in Computer Engineering from the Northeastern University, Boston. Her B.Tech. is from Gitam University, India in Electrical and Electronics Engineering. Her current research interests are performance modeling, resource management, and capacity planning for various emerging inter-disciplinary memory and storage research domains.

Vivek Chaturvedi (Member, IEEE) received the M.S. degree from Syracuse University, Syracuse, NY, USA, and the Ph.D. from Florida International University, Miami, FL, USA. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Indian Institute of Technology, Palakkad, Palakkad, India. He was a Research Scientist with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His current research interests include power and thermal optimization in multi/many core processors, including both 2D and 3D architectures. He is also actively working in the areas of security and reliability of embedded systems.

Gang Quan (Senior Member, IEEE) received and B.S. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, the M.S. degree from the Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree from the University of Notre Dame, Notre Dame, IN, USA. He is currently a Professor with the Department of Electrical and Computer Engineering, Florida International University. His research interests and expertise include real-time computing, smart IoT design, power-/thermal-aware computing, advanced computer architecture, and cloud and reconfigurable computing.