Thermal Aware Lifetime Reliability Optimization for Automotive Distributed Computing Applications

Ajinkya S. Bankar[1], Shi Sha[2], Vivek Chaturvedi[3] and Gang Quan[1]
[1]Department of Electrical and Computer Engineering, Florida International University, Miami, Florida 33174, USA.
[2]Department of Electrical Engineering and Physics, Wilkes University, Wilkes-Barre, Pennsylvania 18766, USA.
[3]Department of Computer Science and Engineering, Indian Institute of Technology, Palakkad, Kerala 678557, India.
Emails: {abank013, ga.quan}@fiu.edu, shi.sha@wilkes.edu, vivek@iitpkd.ac.in

***Abstract*—As the automotive industry is moving towards electric and self-driving vehicles, how to ensure the high degree of reliability for electronic control systems (ECS) has emerged as a serious concern. Temperature plays a vital role in the reliability of ECS because vehicles are subjected to high chip temperature due to harsh operating conditions and high integrated circuit (IC) on-chip power density. In this paper, we study the problem on how to optimize the lifetime reliability and guarantee the chip's peak temperature of ECS by judiciously allocating the application to ECS. We first propose a simple mathematical programming based thermal aware approach, assuming temperature can reach a stable status immediately. We then present a genetic algorithm approach based on effective and computationally efficient methods for peak temperature identification and system-wide lifetime reliability calculation, by taking advantage of the periodicity of vehicle applications. Our experimental results, based on both synthetic test cases and practical benchmarks demonstrate the significant improvement in lifetime reliability and CPU time for automotive ECS achieved by our proposed algorithms compared to the state-of-the-art results.**

***Index Terms*—Automotive reliability, system-level MTTF, Thermal Aware, ECU, ISO 26262.**

I. Introduction

Automotive electronic systems are highly complex and safety-critical. Today, the auto industry is transforming towards autonomous and electric vehicles. This development has presented a tremendous challenge to ensure a much higher degree of reliability than before for the electronic control system (ECS). The ECS consists of a range of heterogeneous electronic units (ECUs), performing critical tasks such as engine control, driver assistance, and passenger safety, and hence are requisite to remain reliable throughout their operational lifetime. Reliability in automotive has become a first-class design constraint. International Standards Organization (ISO) emphasizes significantly on the reliability of electronic systems and has mandated a compliance ISO 26262 [1] for the automotive industry.

The lifetime reliability of an ECU is degraded when it encounters a permanent failure of its components due to aging or wear-out and temperature plays a primary role in this process. First of all, for a vehicle to work reliably in a harsh environment, the ECUs inside vehicles should operate under extreme temperature conditions, with the maximum temperature ranging from 90 °C to 155 °C [4]. Chip temperature higher than its limit can cause a processor to suspend its computation and potentially cause a catastrophic consequence. Besides, the progressive miniaturization of integrated circuits (ICs) and increased power density on the chip results in higher temperature gradients that lead to accelerated chip aging. In particular, ECUs are subjected to rapid 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) [5], [25]. As one of the main driving forces, high operating temperature accounts for 55% of failures in ECUs among all the factors [2].

In this paper, we study the problem of how to optimize the lifetime reliability, in terms of mean time to failure (MTTF), by judiciously allocating distributed automotive applications to a set of heterogeneous ECUs, under a given peak operating temperature constraint. Most of the existing work on the automotive task allocation problem (e.g. [6], [14], [15]) does not take temperature and temperature-dependent system reliability into consideration. For instance, Xie et al. developed a strategy for automotive task allocation to minimize the transient fault that varies with the supply voltage of the target ECU [3]. However, the approach fails to account for temperature when addressing the reliability issues. While Lee et al. [9] proposed an online *single* processor resource management strategy for real-time periodic automotive applications under a peak temperature constraint, they transformed the peak temperature constraint into a constant maximum power limit, which can be very pessimistic. Also, temperature-dependent reliability issues are not taken into consideration in their approach.

Mapping automotive applications on multiple ECUs is itself an NP-hard problem. With conventional approaches such as mathematical programming, evolution algorithms, etc., the key to success is to evaluate the design constraints and optimization goals, including peak temperature and system reliability, effectively and efficiently during the optimization process. A few approaches published (e.g., [20], [26]–[29]) to identify the peak temperature of a processor. Their high computation complexities make them infeasible for use in mathematical programming or meta-heuristic approach, such as the evolution algorithms [22]. Another approach simplifies the peak temperature calculation based on the so-called "step-up schedule" with linear computational complexity [8]. However, as we show later in the paper, using such an approach cannot ensure peak temperature guarantee in certain situations. There is also some work (such as [19]) that simplifies the calculation of peak temperature, assuming the temperature of a processor can immediately reach its stable status. As we show later in the paper, such an approach can lead to an extremely pessimistic design.

The MTTF estimation is a critical aspect of the vehicle design phase. An online method of system-level MTTF in [23] uses a Monte-Carlo simulation, with excessive computation time for high accuracy. Similarly, the system-level lifetime reliability technique for periodic schedules proposed in [25] can be used to estimate the lifetime of the ECUs. However, the computational cost increases rapidly with the numbers of ECUs and tasks in a system. High computational cost in estimating MTTF can significantly degrade the optimization performance during the design space exploration process.

In this paper, we address the lifetime reliability optimization problem through a series of contributions. First, we develop a mathematical programming based approach for allocating automotive applications on distributed ECUs by assuming that the temperature can reach a stable status immediately [18], [19]. This approach incorporates the temperature issues into well-known mathematical programming but can be pessimistic. Next, we show through an example that another state-of-the-art step-up schedule [8] can only guarantee the peak temperature of a periodic schedule when all tasks take their worst-case execution times. For automotive applications, when task execution times are variable [15], a higher peak temperature can be reached and potentially violate the maximum temperature constraint. To this end, we take advantage of the periodicity of vehicle applications [6], [9] and develop a computationally efficient peak temperature estimation method to accurately bound the peak temperature. We also develop a new effective method to estimate system-level MTTF for a periodic schedule with low computational cost than available in the literature [23], [25]. Based on these methods, we develop a genetic algorithm-based approach. Lastly, our experimental results based on synthetic test cases and practical benchmarks demonstrate the effectiveness of our proposed methodology compared to the state-of-the-art.

The rest of the paper is organized as follows. In Section-II, we introduce our system models and formally define our problem. A mathematical programming model for a simple thermal approach is shown in Section-III. A more accurate peak temperature estimation method is shown in Section-IV. In Section-V, we present a computationally efficient system-level MTTF formulation. We present our experiments and results in the Section-VI and conclude in Section-VII.

II. Preliminary

In this section, we first introduce the architecture and application models. We then discuss the thermal and reliability models and formulate our research problem.

A. Architecture and application models

We consider that the vehicle electronic system consists of n_c heterogeneous electronic control units (ECUs), i.e. $\mathbf{EC} = \{EC_1, EC_2, ..., EC_{n_c}\}$, which communicate through the communication network (e.g., CAN buses) connected with different sensors and actuators. Each ECU receives data from sensors or sends control signals to actuators. As the environment temperature for the ECU depends on its mounting location in an automotive [4], we assume different ambient temperatures for different ECUs, i.e., $\mathbf{T_{amb}} = \{T_{amb_1}, T_{amb_2}, ..., T_{amb_{n_c}}\}$, accounting for the cumulative effect of the surrounding environment temperature and possible thermal impacts from all the neighboring mechanical units and ECUs.

We adopt the commonly used directed acyclic graph (DAG), i.e., $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ to model the distributed automotive application, which runs periodically with a period of t_p. The node-set ($\mathcal{V}$) consists of n_v task nodes, i.e., $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_{n_v}\}$, representing tasks to be performed on ECUs for sensing and control. Due to the heterogeneity of ECUs, the same task may take different execution times on different ECUs. The worst-case execution times for all the tasks are thus formulated as a two-dimension matrix, i.e.,$\mathbf{W}_{n_v \times n_c} = \{w_{ik}, i = 1, ..., n_v;\ k = 1, ..., n_c\}$, where w_{ik} denotes the worst-case execution time when the task $\mathcal{V}_i$ is executed on ECU EC_k.

The edge set of the DAG ($\mathcal{E}$) represents the messages passing along communication buses among ECUs, i.e., $\mathcal{E} = \{e_{ij}$, with $\mathcal{V}_i, \mathcal{V}_j \in \mathcal{V}\}$. Each edge e_{ij} denotes the worst-case communication cost for a communication message from $\mathcal{V}_i$ to $\mathcal{V}_j$. We assume that the overhead for handling task executions can be accounted by adjusting the execution times for each task.

B. Power and thermal models

The total power consumption of an ECU is a combination of the dynamic power P_d and leakage power P_s. As shown in [9], in a distributed vehicle application, different tasks exhibit significant dynamic power variations. We thus adopt the task level switching activity α_i to capture the dynamic power differences of different tasks, i.e., $P_d(i) = \alpha_i.V_{dd}^3$, where $P_d(i)$ is the dynamic power consumption of task $\mathcal{V}_i$, α_i is the switching activity of task $\mathcal{V}_i$, V_{dd} is the supply voltage of the corresponding ECU. For idle tasks, there are no switching activities, so we have $\alpha_i = 0$. There are recent studies on thermal aware scheduling, which mostly deal with DVFS based techniques such as [17], [18], [20], [21], but they augment the transient failures thereby degrading lifetime reliability [7]. Therefore, we propose to use a constant voltage scheduling as automotive system demands high safety [1]. An ECU consumes not only dynamic power but also leakage power, which is temperature-dependent [9]. We assume that the leakage power of an ECU is linearly dependent to its temperature, i.e., $P_s = (\phi_1.T + \phi_0)$ [8], where T is the temperature of the ECU, ϕ_0 and ϕ_1 are technology-dependent constants. Therefore, the overall power consumption of an ECU (denoted as P_{total}) when running $\mathcal{V}_i$ can be formulated as, $P_{total} = P_d(i) + P_s = V_{dd}^3.\alpha_i + (\phi_1.T + \phi_0)$. We adopt the standard lumped RC model to represent the thermal behavior of an ECU, which has been widely used in [9], [16], [18], [26], [27]. Specifically, for a given ambient temperature T_{amb_k}, let $T(t)$ and $P(t)$ denote the temperature ($^\circ C$) and power consumption ($Watt$) at time instant t of an ECU. Then we have,

$$R_{th}.C_{th}.\frac{dT(t)}{dt} + T(t) - R_{th}.P_{total}(t) = T_{amb_k}, \quad (1)$$

where R_{th}, C_{th} denote thermal resistance ($^\circ C/W$), and thermal capacitance ($J/^\circ C$). From (1), when an ECU executes task $\mathcal{V}_i$ in time interval $[t_1, t_2]$, let the initial temperature be $T(t_1)$ then the ending temperature is given by,

$$T(t_2) = \frac{A(i)}{B} + \left(T(t_1) - T_{amb_k} - \frac{A(i)}{B}\right) e^{-B(t_2-t_1)} + T_{amb_k}, \quad (2)$$

where $A(i) = (\phi_0 + V_{dd}^3.\alpha_i)/C_{th}$ and $B = 1/(R_{th}.C_{th}) - \phi_1/C_{th}$. When an ECU repeats a mission profile periodically with tasks' worst-case execution times, the transient temperature in the thermal stable status can be quickly captured by the following theorem. (Similar proof can be found in Quan et al. [16] and is omitted due to page limit.)

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, $T_{ss} = T_{amb_k} + \frac{T_L - T_{amb_k}}{1-\mathbb{K}}$, where $\mathbb{K} = exp(-B \cdot t_p)$.

Note that, when an ECU runs a set of tasks periodically, given the dynamic power consumption and worst-case execution time of each task, the temperature at the end of the first period can be readily calculated using (2). Then, we can get the stable status temperature of the ECU using Theorem 1.

C. Lifetime Reliability model

In this paper, we focus on the *Electromigration* (EM) wear-out scheme and estimate the reliability of an ECU by Mean Time to Failure (MTTF) parameter. EM is a well known wear-out failure mechanism for semiconductor devices. It leads to the mass displacement in the conductors and permanent voids in the chip geometry, thereby causing non-repairable damage. The Joule heating and harsh thermal environment result in a soaring temperature profile of the ECU, which worsens the EM, and, thus, the MTTF simultaneously. Specifically, the MTTF is proportional to the EM [11] as $MTTF \propto A_c \cdot J^{-n} \cdot e^{\frac{E_a}{K \cdot T}}$, where A_c is a constant related to the cross-section area of the conductor, J is the current density in Amp/cm^2, E_a is the activation energy in electron-volts, K is the Boltzmann's constant, and T is the Kelvin temperature. We can observe that, the lifetime reliability is inversely proportional to the temperature and it can be improved if the run-time temperature of the ECU is lowered. In this paper, the typical electromigration-induced failure scaling factor n is 2 [11], if not otherwise specified.

D. Problem formulation

Given an $\mathbf{EC}$ and the DAG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, our goal is to map all task nodes in $\mathcal{G}$ to minimize the overall makespan (and thus to avoid the potential deadline miss), and maximize its lifetime reliability. In the meantime, we want to optimize the design by lowering down its peak temperature due to two reasons: (1) Many modern processors have a thermal protection mechanism to temporarily halt the processor once the temperature exceeds a certain temperature threshold, thereby degrading system performance [18]; (2) Reducing peak temperature helps to improve the reliability of an ECU. In summary, our problem can be formulated as follows.

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 no more than the given temperature threshold (T_{max}); (2) the makespan (C_{max}), peak temperature, and lifetime reliability ($MTTF$) of the design is optimized.

III. Mathematical Programming Approach

The design objective formulated in Problem 1 is an NP-hard problem and different design optimization methodologies have been proposed, such as convex optimization [5], [17], analytical approach [9], [16], mathematical programming [10], meta-heuristics approach [12], [25]. In this section we incorporate thermal behavior in mathematical programming.

To quickly identify the peak temperature for a given task set $\mathcal{V}$ mapping on $\mathbf{EC}$, let x_{ik} be the task assignment variable defined as:

$$x_{ik} = \begin{cases} 1, & \text{when } \mathcal{V}_i \text{ is assigned to } EC_k; \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Then, task $\mathcal{V}_i$ should be allocated to only one processor, where $\sum_{k=1}^{n_c} x_{ik} = 1, \forall \mathcal{V}_i \in \mathcal{V}$. If σ_i is the starting time of task $\mathcal{V}_i$, the maximum makespan C_{max} is

$$\sigma_i + \sum_{k=1}^{n_c} w_{ik} \cdot x_{ik} \leq C_{max}, \forall \mathcal{V}_i \in \mathcal{V}. \quad (4)$$

When incorporating the communication cost into the timing constraint, we have

$$\sigma_i + w_{ik} \cdot x_{ik} + e_{ij} \cdot (x_{jl} + x_{ik} - 1) \leq \sigma_j, \quad (5)$$

where EC_k and $EC_l \in \mathbf{EC}, \mathcal{V}_i$ and $\mathcal{V}_j \in \mathcal{V}, \forall e_{ij} \in \mathcal{E}$. Further, to avoid the time overlapping of different tasks on the same ECU, we have

$$\begin{aligned} &\sigma_i + w_{ik} - \sigma_j \leq \mathcal{M} \cdot (2 - x_{ik} - x_{jk}) \text{ or} \\ &\sigma_j + w_{jk} - \sigma_i \leq \mathcal{M} \cdot (2 - x_{ik} - x_{jk}), \end{aligned} \quad (6)$$

where $\mathcal{V}_i \neq \mathcal{V}_j \in \mathcal{V}$, $EC_k \in \mathbf{EC}$ and $\mathcal{M}$ is a large positive constant. More detailed discussions and explanations of these formulations can be found in [10]. Note that (3) - (6) identify the constraints for makespan and precedence constraints during the task assignment phase, which do not consider temperature impacts. From (1), temperature varies with t non-linearly. To incorporate temperature impacts into the mathematical programming formulation, a straightforward strategy is to consider that an ECU can immediately reach the temperature stable status when executing a task [18], [19]. Specifically, this assumption let $\frac{dT(t)}{dt} = 0$ in (1) for any $\mathcal{V}_i \in \mathcal{V}$, so each task can immediately reach its stable state temperature as $T = T_{amb} + R_{th}P_{total}$, regardless of the starting temperature and thermal capacitance of the ECU.

Therefore, when EC_k runs $\mathcal{V}_i$, the temperature is a constant value of $T_k^*(i)$ as follows

$$T_k^*(i) = \frac{A(i)}{B} + T_{amb_k}. \tag{7}$$

With the knowledge of the temperature constant for each task, the highest system-wide temperature (denoted as T^*) of **EC** when executing task set $\mathcal{V}$ is

$$T^* = \max_{i,k}(T_k^*(i)), \qquad \forall \mathcal{V}_i \text{ allocated on } EC_k. \tag{8}$$

Now, we formally bound the system peak temperature in the following theorem.

Theorem 2: For any mapping of $\mathcal{V}$ to **EC**, with periodic execution, the resultant highest system temperature never exceeds T^*.
This theorem can be easily proved by showing that the operating temperature of any task (e.g., $\mathcal{V}_i$) is no more than calculated by (7) and the proof is omitted due to the page limit. With Theorem 2, the peak temperature constraint must be guaranteed as long as $T^* \leq T_{max}$, for any mapping strategy.

The makespan minimization and the peak temperature minimization usually result in contradictory scheduling decisions. However, to satisfy the application demands under the physical limitations of an ECU, in this paper, we combine a weighted sum of these two design objectives, such that our optimization framework is general enough to be adopted in arbitrary and adjustable scenarios as follows

$$Max : \mathbb{W}_1 \cdot \frac{C_s - C_{max}}{C_s} + \mathbb{W}_2 \cdot \frac{T_{max} - T^*}{T_{max}}, \tag{9}$$

where $\mathbb{W}_1, \mathbb{W}_2$ are the weights and $\mathbb{W}_1 + \mathbb{W}_2 = 1$. C_s is a constant representing the typical vehicle application makespan without considering temperature effects.

Note that the peak temperature bound in Theorem 2 could be tight, when most of the tasks' actual execution times ($\forall \mathcal{V}_i \in \mathcal{V}$) are long enough and their dynamic power factors (α_i) are big enough to truly reach and keep at the highest temperature (T^*) in one period. Otherwise, due to the waste of temperature "headrooms", the feasibility could be hurt or squeeze the makespan minimization spaces.

To improve the quality of the solution, in what follows, we utilize a more sophisticated metaheuristic approach for thermal analyses and involve MTTF to our design goals.

IV. Bound the worst-case peak temperature

Several approaches have been proposed to estimate the peak temperature of a processor. For example, the numerical method splits each execution interval into short pieces and seeks the peak temperature by checking each small step length in [28]. Interval-wise peak temperature detection methodologies have been proposed by using an amalgamation of analytical and numerical method in [26] or by solving the first-order derivative on each processing unit via the Newton-Raphson method in [27]. A thermal-aware scheduling algorithm with power/temperature-driven priority assignment has been explored in [20]. However, some of the approaches do not focus on the periodic task executions (e.g. [27]) and may have very high computation complexity (e.g. [28], [29]), which makes them infeasible to be applied during the run-time design space exploration.

The state-of-the-art *hypothetical step-up schedule* theory in [18] can be used to bound the peak temperature with the advantages that: (1) the complexity to identify the peak temperature is linear (concerning the number of intervals); (2) the step-up schedule is unique as long as the global schedule is determined. However, one major problem is that using such a hypothetical schedule can only bound the peak temperature when the tasks' actual execution times take their worst-case execution times (WCETs). When task execution times vary from their WCETs, the actual execution trace's peak temperature may exceed the temperature bound in [18], as shown in the following motivational example. In Figure 1(a),

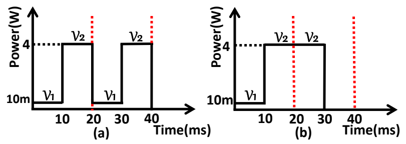


Fig. 1: A motivational example.

we construct a step-up schedule of two tasks $\mathcal{V}_1$ and $\mathcal{V}_2$ with power consumption of $100\ mW$ and $3.86\ W$, respectively. Let each task's WCET be $10\ ms$, the period be $20\ ms$, and let the schedule execution be repeated periodically. Detailed run-time power and thermal parameters can be seen in Section VI-A. When each task's actual execution time strictly follows its WCET, as shown in Figure 1(a), the *hypothetical step-up schedule* predicts that the peak temperature is $93.61\ °\text{C}$. Nevertheless, assume that task $\mathcal{V}_1$ spends a negligible execution time in its second occurrence, as shown in Figure 1(b), then, the peak temperature is $94.02\ °\text{C}$, which exceeds the predicted bounds using a *hypothetical step-up schedule*.

This motivation example clearly shows that the *hypothetical step-up schedule* theory can not effectively bound the peak temperature for a periodic schedule when task execution times vary from their WCETs.

Similar scenarios in practical ECU attract our attention that for the angular tasks associated with crankshaft rotation of the automobile engine, tasks usually have lower execution times than their WCETs when the engine is in steady-state [15]. Although the algorithm presented by Schor et al. [29] can bound the peak temperature for such cases by checking all possibilities of the task arrival and different execution patterns, its computational cost can be prohibitively high, especially when tens to hundreds of ECUs are embedded in one vehicle for executing large task-sets.

To present our new approach that accurately bound the peak temperature, we first introduce several lemmas. Then we present an algorithm for our approach and introduce a theorem that ensures the correctness of our algorithm.

Lemma 1: Let an ECU run n tasks $\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_n$ consecutively with a period of t_p. Let $T_m(k \cdot t_p)$ be the temperature at $t = k \cdot t_p$ when all tasks are executed with their WCETs.

Then $T_m(k \cdot t_p) \geq T(k \cdot t_p)$, if $T(k \cdot t_p)$ is the temperature at $t = k \cdot t_p$ when at least one task instance from $t \in [0, k \cdot t_p]$ runs with execution time less than its WCET.

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.*).

From Lemma 1, the stable status temperature at the starting point of a periodic schedule will always decrease if at least one task's execution time is less than its WCET. This lemma can be proved by using the known conclusions shown in previous literature (e.g., *Theorem 7* in [29]). Lemma 2 shows that when running a task within an interval, if the task's stable status temperature when executed alone is higher (lower *resp.*) than it's starting temperature, the temperature of ECU during the interval will monotonically increase (decrease *resp.*). The proofs of the lemmas are omitted due to the page limit.

Based on Lemma 1 and 2, we develop an algorithm to identify the peak temperature when an ECU runs a set of tasks periodically with possible variable execution times. Algorithm 1, calculates the stable status temperature at the end of the application period, assuming that each task takes its WCET in Line 1. From Line 2 to Line 8, we iteratively construct a task sequence resulting in the highest peak temperature for all tasks allocated on EC_k. At last, the peak temperature, considering variable execution time, is returned in Line 9. For each ECU, the complexity of Algorithm 1 is $O(n)$, with n being the number of tasks allocated to an ECU.

Algorithm 1: Peak temperature identification.

```
Inputs: Mapped tasks's WCET schedule on $EC_k$;
        Power/thermal parameters; $\mathbf{W}_{n_v \times n_c}$;
Output: Temperature bound $T_m$.

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 (7);
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 $T_m$
```

Moreover, we have the following theorem on the property of the peak temperature identified through Algorithm 1.

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

Without loss of generality, this theorem can be easily proved with the help of lemma 1 and lemma 2 but omitted due to page limitation. Theorem 3 validates the effectiveness of the peak temperature output from Algorithm 1.

In the following section, we introduce a new method to compute the system-level MTTF.

V. Computationally Efficient System-Level MTTF Formulation

In this section, we first present a state-of-the-art method that accurately calculates system-wide MTTF [25]. We then discuss the limitation of this method and propose a novel computationally efficient formulation to estimate the system-level MTTF. We also present a comparative study on the two methods for average relative error rate.

We use Weibull distribution to study the wear-out effect at the system-level [25]. The reliability of single ECU at time instant t with temperature T is $\mathcal{R}(t, T) = e^{-\left(\frac{t}{\eta(T)}\right)^{\beta}}$, where $\eta(T)$ and β represents the scale and slope parameters of the Weibull distribution, respectively. From the MTTF of Weibull distribution, we have

$$\eta(T) = \frac{MTTF(T)}{\Gamma\left(1 + \frac{1}{\beta}\right)}. \tag{10}$$

The system-level reliability with n_c ECUs for k periods [25] is $\mathcal{R}_{system}(k \cdot t_p) = e^{-\sum_{j=1}^{n_c} (k \cdot \mathcal{A}_j)^{\beta}}$, where the aging factor $\mathcal{A}$ is constant in the thermal stable status and calculated for s task intervals in a period as [25]

$$\mathcal{A} = \sum_{i=0}^{s-1} \frac{\Delta t_i}{\eta(T_i)}. \tag{11}$$

As $MTTF \gg t_p$, the system-level $MTTF$ can be calculated at each t_p and approximated [25] as

$$MTTF_{system} = \sum_{i=0}^{\infty} e^{-\sum_{j=1}^{n_c} (i \cdot \mathcal{A}_j)^{\beta}} \cdot t_p. \tag{12}$$

The implementation of (12) can be quite time-consuming. Since to get the close approximation of MTTF we need to calculate the sum on the right hand side of (12) when i-index reaches a large value until the MTTF is saturated, which can take a significant amount of computation time as later observed in Section-VI-B. Therefore, it is necessary to find an effective and efficient way to estimate the system-level MTTF which can reduce the computation time with a negligible compromise of accuracy.

According to (11), the aging factor is calculated by assuming a constant temperature for each task interval. On the other hand, according to practical automotive benchmarks as task intervals are in several μs to ms [3], [14], we can assume constant temperature over the period to simplify the system-level MTTF calculation. Specifically, we assume η_1,...,η_{n_c} are the scale parameters of the $\mathbf{EC}$ given by (10) which depend on the constant temperature over the period. We use average temperature of a period to find $MTTF(T)$ in stable status as

$$T_{avg} = \frac{\sum_{i=0}^{s-1} \int_{l_i}^{l_{i+1}} \frac{A(i)}{B} + \left(T(t_i) - T_{amb_k} - \frac{A(i)}{B}\right) e^{-B \cdot L_i} + T_{amb_k}}{t_p}.$$

TABLE I: Average MTTF error percentage

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

where s are the number of intervals in a period, l_i is the length of each interval, and t_p be the period. We assume that the slope parameter of the Weibull distribution is same for each ECU. The reliability of the system is $\mathcal{R}_{system} = e^{-\sum_{j=1}^{n_c}\left(\frac{t}{\eta_j}\right)^{\beta}}$. Therefore, the system-level MTTF is given by,

$$MTTF_{system}^{Fast} = \int_0^{\infty} e^{-\sum_{j=1}^{n_c}\left(\frac{t}{\eta_j}\right)^{\beta}} dt. \quad (13)$$

By substituting $x = \sum_{j=1}^{n_c}\left(\frac{t}{\eta_j}\right)^{\beta}$ in (13), we get $x = t^{\beta} \cdot \sum_{j=1}^{n_c}\left(\frac{1}{\eta_j^{\beta}}\right)$. Then, $x^{\frac{1}{\beta}} = t \cdot \left(\sum_{j=1}^{n_c}\left(\frac{1}{\eta_j^{\beta}}\right)\right)^{\frac{1}{\beta}}$. By re-arranging the terms, $t = \frac{x^{\frac{1}{\beta}}}{\left(\sum_{j=1}^{n_c}\left(\frac{1}{\eta_j^{\beta}}\right)\right)^{\frac{1}{\beta}}}$, and thus, $dt = \frac{x^{\frac{1}{\beta}-1}\cdot dx}{\beta\cdot\left(\sum_{j=1}^{n_c}\left(\frac{1}{\eta_j^{\beta}}\right)\right)^{\frac{1}{\beta}}}$. Note that when $t \to 0$, we have $x \to 0$ and $t \to \infty$, we have $x \to \infty$ respectively. Now, we can define (13) as, $MTTF_{system}^{Fast} = \frac{1}{\beta\cdot\left(\sum_{j=1}^{n_c}\left(\frac{1}{\eta_j^{\beta}}\right)\right)^{\frac{1}{\beta}}} \int_0^{\infty} e^{-x} \cdot x^{\frac{1}{\beta}-1} \cdot dx.$
which converges to the following simple form [24],

$$MTTF_{system}^{Fast} = \frac{\Gamma(\frac{1}{\beta})}{\beta \cdot \left(\sum_{j=1}^{n_c}\left(\frac{1}{\eta_j^{\beta}}\right)\right)^{\frac{1}{\beta}}}. \quad (14)$$

Since $\Gamma(\frac{1}{\beta})$ can be readily calculated, the MTTF value can be estimated very quickly. In the meantime, Table I shows the average percentage errors between MTTF values using (12) and (14) respectively. The results were obtained assuming that all tasks are assigned to a single processor. The parameter details for synthetic task cases and benchmarks are described in Section VI-A. We can observe that the average error percentage is below 0.1%.

With our new methods for the peak temperature identification and MTTF estimation, we are ready to utilize the genetic algorithm for Problem 1 which has the similar implementation as in [12], and its set-up is described in the following section.

VI. Experimental Results

In this section, we validate our design with different platform/parameter settings and compare their performances with the state-of-the-art techniques.

A. Experimental setup

Our genetic algorithm is constructed as follows:

TABLE II: ECU parameters for the experiment [9], [13]

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

- *Chromosome design*: Each chromosome has two parts. The first part defines the task mapping to the ECUs and the second part defines the task schedule that confirms to the precedence constraints of the DAG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$.
- *Fitness function*: The makespan is calculated similarly to that in [12], and Algorithm 1 is implemented to find the peak temperature for both GAs. We assume that the WCET schedule is used to bound the system-level MTTF for both the GAs. The system-level MTTF is calculated by (14) and (12) respectively in each GA to compare their effectiveness. The makespan, peak temperature, and MTTF are normalized, and their weighted sum is used as the fitness function given by,

$$f = \mathbb{P}_1\frac{C_s - C_{max}}{C_s} + \mathbb{P}_2\frac{T_{max} - T_m}{T_{max}} + \mathbb{P}_3\frac{MTTF_{system}}{MTTF_{max}}, \quad (15)$$

  where $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ are the weights with $\mathbb{P}_1 + \mathbb{P}_2 + \mathbb{P}_3 = 1$, and C_s is a constant for the smallest possible makespan of the first generation. $MTTF_{max}$ is a system-level maximum MTTF calculated assuming the ambient temperature for each ECU.
- *Parent/survivor selection*: A Pareto optimal front is maintained from generation to generation, and tournament selection strategy similar to that in [12] is used.
- *Crossover/mutation operators*: Operators mainly work on the first section of the chromosome (task mapping) and are designed to be adaptive (see [12] for more details).

The proposed methodologies were tested on heterogeneous ECU platforms with different numbers of processing units and task nodes. The typical power and thermal-related parameters are listed in Table II. Both synthetic test cases and practical benchmarks are used in our experiments. For the synthetic validation, we used a small set of 3 ECUs with 10, 15, 20, and 25 tasks to ease the presentation and tracing the parameter trade-offs. Later, we utilized more ECUs and tasks in benchmark verification. The ambient temperatures were 35 °C, 40 °C, and 45 °C to approximate the variety of ECU mounting environments. Also, the temperature threshold T_{max} was set to 150 °C. The task graphs were generated using a Task Graph Generator [3] with $average\ computation\ cost = 15\ ms$, $communication\ to\ computation\ ratio = 1$, and the $shape$ parameter shifted in the range of [0.3, 0.9] to cover a broad spectrum of the test cases. The dynamic power for task nodes is uniformly distributed [9] in the range of [0.1 W, 3.86 W]. Besides the synthetic tests, we verified our design by two practical benchmarks, *Real-Life* [3] and *Automotive Cruise Controller (ACC)* [14], whose parameters are listed in Table III. For both synthetic validations and benchmark verification, without losing generality, we set the weight com-

TABLE III: 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 [3], [§]described in [14], [¶]described in [9]

bination as $\mathbb{W}_1 = \mathbb{W}_2 = 1/2$ and $\mathbb{P}_1 = \mathbb{P}_2 = \mathbb{P}_3 = 1/3$ which gives an equal importance to each parameter.

To evaluate the electromigration-induced MTTF, we set $A_c = 10^{-7}\ cm^2$, $J = 5.7 * 10^5\ Amp/cm^2$, $E_a = 0.48\ eV$ [11]. These values ensure approximately 9000 hours of lifetime (equivalent to 10 years lifetime at approx. 2.5 hours per day) for an ECU at the temperature of 80 °C.

For both the genetic algorithms, the size of initial solution is chosen as 500, and the genetic algorithm is set to evolve for 400 generations, thereby terminating with Pareto optimal front solution. We compare the proposed Algorithm 1 with a state-of-the-art simple thermal approach. Also, we evaluate the efficiency of the proposed system-level MTTF computation with that of [25]. The four test approaches are stated below:

Temperature Oblivious Approach (M-TO): The mathematical programming approach that minimizes the overall makespan but does not consider temperature impacts. We have used the integer linear program used in [10] for implementation of this approach.

Simple Thermal Aware Approach (M-STA): The mathematical programming approach that assumes an ECU reaches its stable status temperature immediately [18], [19]. The formulations are explained in Section-III.

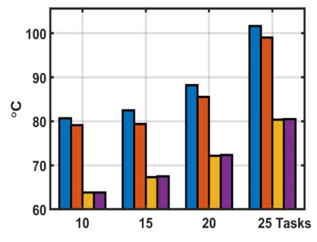
Accurate Peak Temperature Identification with Fast MTTF Calculation (G-APTI-FMC): The genetic programming approach which bounds the peak temperature with the proposed Algorithm 1 and proposed system-level MTTF formulations as described in Section-V.

Accurate Peak Temperature Identification with Traditional MTTF Calculation (G-APTI-TMC): The genetic programming approach that identifies the peak temperature accurately using the proposed Algorithm 1 and system-level MTTF formulations given in [25].

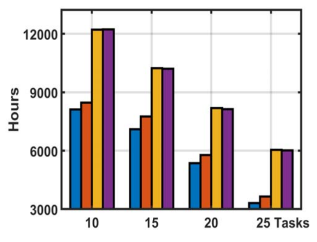
CPLEX 12.10.0 was used to solve the formulated mathematical programming problems of **M-TO** and **M-STA**. Whereas, **G-APTI-FMC** and **G-APTI-TMC** were implemented using Matlab R2020a, running on HP Z800 server with 24 cores and 32GB memory.

B. Experimental results and discussions

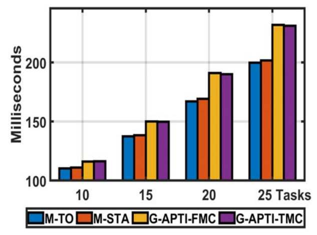
For each configuration of 3-ECUs, we generated random test cases and collected the first 100 feasible test case performance, including makespan, peak temperature, MTTF, and CPU times. From Fig. 2(c), we can see that without thermal awareness, *M-TO* has the smallest makespan, but it also leads to the lowest system-level MTTF. When taking temperature into account,



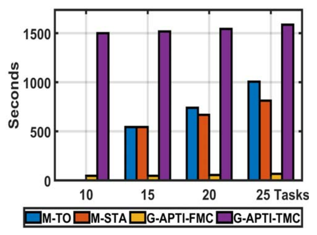
(a) Avg. system peak-temperature



(b) Estimated avg. system-MTTF[†]



(c) Avg. makespan[†]



(d) Avg. CPU time[†]

Fig. 2: Experimental results for 3-Processor system
[†]Numbers are rounded to the nearest integer value for the sake of presentation

the makespan from other approaches become longer. With increased thermal prediction accuracy, *G-APTI-FMC* and *G-APTI-TMC* produce the longest makespan.

In the meantime, from Fig. 2(a), we observe that the average peak temperature for *G-APTI-FMC* and *G-APTI-TMC* differs approximately in the range of 0.1 °C to 0.3 °C, while, 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. 2(b). But there is an increase of about 31 to 23 times in CPU time for the *G-APTI-TMC* as compared to the proposed *G-APTI-FMC*. So, with a similar peak temperature and MTTF performances, *G-APTI-FMC* outperforms *G-APTI-TMC* in terms of computational efficiency.

If we assume that the proposed *G-APTI-TMC* as a baseline performance, then from Fig. 2(a), we perceive that the state-of-the-art *M-STA* is pessimistic by 15.3 °C, 11.9 °C, 13.2 °C, and 18.5 °C as we increase the task numbers from 10 to 25. The accuracy of the peak temperature identification for *G-APTI-FMC* and *G-APTI-TMC* helps to estimate the system-level MTTF better than *M-STA*, as evident in Fig. 2(b).

Meanwhile, there is another interesting fact observed in the Fig. 2(d), as the number of tasks increases, T_{max} becomes the tighter constraint on the search space and prune the solutions, thereby, the CPU time of *M-STA* is lower than that of *M-TO*. Also, we can observe that the average CPU time increases exponentially for *M-TO* and *M-STA* as the number of tasks are increased due to mathematical programming models used for their implementation. Therefore, we could not use the test cases with a large number of tasks and ECUs in our experiments.

We observe similar phenomena for the two practical benchmarks. As shown in Table IV, there is a peak temperature difference of 0.1 °C and MTTF difference of 1 hour between *G-APTI-FMC* and *G-APTI-TMC*, but *G-APTI-FMC* has speedup the MTTF calculations by 191 times than *G-APTI-*

TABLE IV: 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 V: Results of ACC Benchmark

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

TMC. Whereas, the existing *M-STA* is pessimistic by 40.1 °C than *G-APTI-FMC* which highlights the important contribution of the proposed *G-APTI-FMC* for the practical systems.

From Table V, we can see that the peak temperature of *M-TO* is higher than that of *M-STA* by 4.7 °C but still estimates MTTF 10 hours higher than *M-STA*. It is because the system-level MTTF depends not only on the system's peak temperature but also on the individual temperatures of all the ECUs in a system.

VII. Conclusion

In this paper, we present different approaches to minimize the overall makespan and peak temperature under maximum temperature constraint and optimize the system-level MTTF by judiciously mapping automotive application on distributed heterogeneous ECUs. Besides a mathematical programming approach with thermal awareness, we develop effective and efficient strategies to bound the peak temperature and estimate MTTF quickly. Based on this, we develop a genetic algorithm for task mapping to enhance operational lifetime reliability. Our experimental results demonstrate the importance of thermal awareness on reliability enhancement for vehicle systems and validate the efficiency of our proposed approach.

References

[1] Road Vehicles-Functional Safety, ISO Std. 26262, 2018.

[2] ZVEI, German Electrical and Electronic Manufacturers' Association, Handbook for Robustness Validation of Automotive Electrical/Electronic Modules, Frankfurt, Germany, 2008.

[3] G. Xie, et. al., "Fast Functional Safety Verification for Distributed Automotive Applications During Early Design Phase," in *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4378-4391, May 2018.

[4] M. Kruger, et. al., "Requirements for the application of ECUs in e-mobility originally qualified for gasoline cars," *Proceedings of the 27th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis*, vol. 64, pp. 140-144, Sept. 2016.

[5] P. Mercati, et. al., "WARM: Workload-Aware Reliability Management in Linux/Android," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1557-1570, Sept. 2017.

[6] K. Lee, et. al., "Mixed harmonic runnable scheduling for Automotive Software on Multi-core processors," *Int. Journal of Automotive Technol.*, vol. 19, pp. 323-330, Apr. 2018.

[7] B. Zhao, et. al., "On Maximizing Reliability of Real-Time Embedded Applications Under Hard Energy Constraint," *IEEE Trans. Industrial Informatics*, vol. 6, no. 3, pp. 316-328, Aug. 2010.

[8] V. Chaturvedi, et. al., "On the fundamentals of leakage aware real-time DVS scheduling for peak temperature minimization," *Elsevier Journal of Systems Architecture*, vol. 58, no. 10, pp. 387-397, Nov. 2012.

[9] Y. Lee, et. al., "Thermal-Aware Resource Management for Embedded Real-Time Systems," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2857-2868, Nov. 2018.

[10] A. A. El Cadi, et. al., "Mathematical programming models for scheduling in a CPU/FPGA architecture with communication delay," *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)*, Rabat, 2013, pp. 1-6.

[11] J. R. Black, "Electromigration - a brief survey and some recent results," in *IEEE Trans. Electron Devices*, vol. 16, no. 4, pp. 338-347, Apr. 1969.

[12] F. Omara and M. Arafa, "Genetic algorithms for task scheduling problem," in *Elsevier Journal of Parallel Distrib. Comput.*, vol. 70, pp. 13-22, Oct. 2009.

[13] Y. Lee et. al., "Efficient thermoelectric cooling for mobile devices," *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Taipei, 2017, pp. 1-6.

[14] S. K. Roy, et. al., "Optimal Scheduling of Precedence-constrained Task Graphs on Heterogeneous Distributed Systems with Shared Buses," *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, Valencia, Spain, 2019, pp. 185-192.

[15] M. Faizan and A. S. Pillai, "Dynamic Task Allocation and Scheduling for Multicore Electronics Control Unit (ECU)," *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2019, pp. 821-826.

[16] G. Quan and V. Chaturvedi, "Feasibility Analysis for Temperature-Constraint Hard Real-Time Periodic Tasks," in *IEEE Trans. on Industrial Informatics*, vol. 6, no. 3, pp. 329-339, Aug. 2010.

[17] A. Iranfar, et. al., "TheSPoT: Thermal Stress-Aware Power and Temperature Management for Multiprocessor Systems-on-Chip," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1532-1545, Aug. 2018.

[18] S. Sha, et. al., "On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems," in *ACM Trans. on Design Automation of Electronic Systems*, vol. 25, no. 2, article 23 , Feb. 2020.

[19] A. Mutapcic, et. al., "Processor Speed Control With Thermal Constraints," in *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1994-2008, Sept. 2009.

[20] T. Li, et. al., "Temperature Minimization and Thermal-Driven Scheduling for Real-Time Periodic Tasks," in *Journal of Sign. Process. Syst. 91*, 685–700 (2019).

[21] K. Cao, et. al., "Lifetime-aware real-time task scheduling on fault-tolerant mixed-criticality embedded systems," in *Elsevier Journal of Future Generation Computer Systems*, vol. 100, 165-175, Nov. 2019.

[22] H. F. Sheikh, et. al., "Performance, Energy, and Temperature Enabled Task Scheduling using Evolutionary Techniques," in *Elsevier Journal of Sustainable Computing: Informatics and Systems*, vol. 22, pages 272-286, Jun. 2019.

[23] Y. Ma, et. al., "Improving System-Level Lifetime Reliability of Multicore Soft Real-Time Systems," in *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1895-1905, June 2017.

[24] F. Qi, and C. Chen, "A complete monotonicity property of the gamma function," in *Elsevier Journal. Math. Anal. Appl.* 296 (2004) 603–607.

[25] L. Huang, et. al., "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," *2009 Design, Automation & Test in Europe Conference & Exhibition*, Nice, 2009, pp. 51-56.

[26] Q. Han, et. al., "Temperature-Constrained Feasibility Analysis for Multicore Scheduling," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 2082-2092, 2016.

[27] S. Pagani, et. al., "MatEx: Efficient transient and peak temperature computation for compact thermal models," *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, 2015, pp. 1515-1520.

[28] S. Sharifi, et. al., "PROMETHEUS: A Proactive Method for Thermal Management of Heterogeneous MPSoCs," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 1110-1123, July 2013.

[29] L. Schor, et. al., "Worst-Case Temperature Guarantees for Real-Time Applications on Multi-core Systems," *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*, Beijing, 2012, pp. 87-96.