Reliability Enhancement Toward Functional Safety Goal Assurance in Energy-Aware Automotive Cyber-Physical Systems

Guoqi Xie, *Member, IEEE*, Hao Peng, Zhetao Li, *Member, IEEE*, Jinlin Song, Yong Xie, *Member, IEEE*, Renfa Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Automotive cyber-physical systems are energy-aware and safety-critical systems where energy consumption should be controlled from a perspective of design constraints and reliability should be enhanced toward functional safety goal assurance. In this paper, we solve the problem of reliability enhancement of an automotive function (i.e., functionality or application) under energy and response-time constraints based on the dynamic voltage and frequency scaling technique. The problem is solved by a two-stage solution, namely, response-time reduction under energy constraint and reliability enhancement under energy and response-time constraints. The first stage is solved by proposing average energy preallocation, and the second stage is solved by proposing a reliability-enhancement technique based on the first stage. Examples and experiments show that the proposed solution can not only assure energy and response-time constraints, but also enhances reliability as much as 16.66% compared with its counterpart.**

***Index Terms*—Automotive cyber-physical systems (ACPS), energy aware, functional safety, response-time constraint, reliability enhancement.**

Manuscript received March 28, 2018; revised June 4, 2018; accepted July 6, 2018. Date of publication July 10, 2018; date of current version December 3, 2018. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0202901, Grant 2017YFB0202905, in part by the National Natural Science Foundation of China under Grant 61702172, Grant 61672217, in part by the Natural Science Foundation of Hunan Province under Grant 2018JJ3076, in part by the CCF-Venustech Open Research Fund under Grant CCF-VenustechRP2017012, in part by the Open Research Project of the State Key Laboratory of Synthetical Automation for Process Industries (SAPI), Northeastern University, China under Grant PAL-N201803, in part by the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China under Grant ICT1800399, and in part by the Fundamental Research Funds for the Central Universities. Paper no.-TII-18-0775. *(Corresponding author: Renfa Li.)*

G. Xie, H. Peng, J. Song, and R. Li are with the College of Computer Science and Electronic Engineering, Hunan University, Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha Hunan 410082, China (e-mail: xgqman@hnu.edu.cn; hao_peng@hnu.edu.cn; songjinlin@hnu.edu.cn; lirenfa@hnu.edu.cn).

Z. Li is with the College of Information Engineering, Xiangtan University, Hunan 411105, China (e-mail: liztchina@hotmail.com).

K. Li is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2018.2854762

I. Introduction

A. Background

AUTOMOBILES are important ground-mobility vehicles in intelligent transportation systems and they are having a paradigm shift toward more efficient and green transportation [1], [2]. Several automotive functions (i.e., functionalities or applications), such as brake-by-wire, lane departure warning, and collision avoidance, depend on the interaction, feedback, and coordination of multiple electronic control units (ECUs) through networks by collecting and transferring physical world data from 360° sensors to actuators [3], [4]. For example, the brake-by-wire function is released by receiving collected data from a sensor, executing the data in multiple ECUs, transmitting them in two controller area network (CAN) buses, and completing the process by sending the performing action to two actuators [5]. Considering that this process is directed and acyclic, a directed acyclic graph (DAG) has been used to represent a distributed automotive function, which consists of multiple tasks with precedence constraints [4], [6]. Automotive systems have been studied as automotive cyber-physical systems (ACPS) because of the joint and tight interactions between automotive functions and the physical world [4]. CPS are engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components [7]. The automotive industry is power-sensitive and environment-friendly for green environmental protection [8]. Due to the particularity of the application environments, ACPS are energy-aware systems (i.e., lifetime-aware systems), which are usually fuel-powered or battery-powered, such that the energy supply is limited and energy consumption should be controlled from a perspective of design constraint [9], [10].

In addition to energy consumption control, ACPS are safety-critical systems, in which any small failures may result in serious risks, such as unexpected acceleration, deceleration and steering of the vehicle, abnormally opening airbag, sudden opening of the door when the vehicle is traveling at a high speed, and exception of the break-by-brake. These risks will affect driving safety, resulting in various casualty accidents such as death, injury, and damage. To cope with the increasing safety risks of automobiles, the road vehicles functional safety standard ISO 26262 was officially released in 2011 to address the automotive functional safety problem [11]. Functional safety refers to the

absence of unreasonable risks due to hazards caused by malfunctioning behavior resulting from systematic failures and random hardware failures [6], [11]. Missing response-time constraint (i.e., real-time constraint, timing requirement, and deadline) is considered a typical systematic failure, and the function must be correctly finished in its hard deadline. Abnormal execution is considered a typical random hardware failure, and it is related to reliability, which is understood as the probability of the function surviving for a given period of time [6]. However, increased failure rates of chips affect the reliability due to the increase of high-density chip integration, high temperature, and other electromagnetic interferences and radiations [12], [13].

B. Motivation

Response-time constraint and reliability goal must be simultaneously satisfied to assure automotive functional safety goal. However, they may not be satisfied simultaneously in practice because increasing reliability intuitively increases the response time of a DAG-based distributed function [6], [14]–[16]. Response-time minimization and reliability maximization are conflicting, such that assuring functional safety goal is a bicriteria optima problem [6], [14], [15], [17]. For this reason, we aim to propose a method to find the solution with the maximum reliability under energy and response-time constraints. By solving this problem, we can implement a reliability-enhancement (i.e., making the reliability value higher) technique to assure high-reliability goal of a function, thereby assuring high functional safety goal.

Dynamic voltage and frequency scaling (DVFS) is a popular energy-optimization technique in embedded systems by dynamically scaling down the voltage or frequency of a processor (ECU) [15], [18]–[20]. However, DVFS could lead to two negative impacts: 1) increase in the execution time of tasks, consequently increasing the response time of the function, thereby missing the response-time constraint of the function; and 2) rise in transient failures of the ECU, consequently weakening the reliability of the function [18]. In other words, reliability, response time, and energy consumption will affect each other. For this reason, Zhao *et al.* [18] studied the same problem as this paper of reliability enhancement of a distributed function under energy and response-time constraints in DFVS-enabled systems; however, the objective platform was a uniprocessor system. ACPS are heterogeneous distributed embedded systems, where several heterogeneous ECUs with different performance are distributed in communication buses, and communication among ECUs is performed through message passing over buses [4], [6]. That is, we need to restudy this problem to adopt the heterogeneity and distribution of ACPS. Considering that the heterogeneity of ECUs can lead to uncertain task allocation, and the interaction among tasks depends on the communication in the bus, solving the topic problem in ACPS is much more complicated than in a uniprocessor system.

C. Overview and Contributions

We propose a two-stage solution because the problem to be solved is complicated. The first stage solves the basic problem of response-time reduction under energy constraint (RREC). On the basis of the first stage, the second stage solves the topic problem of reliability enhancement under energy and response-time constraints (REREC). A flow chart of the proposed two-stage solution is shown in Fig. 1.

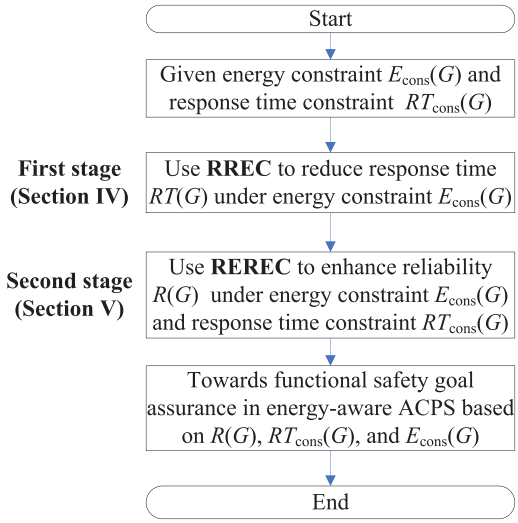


Fig. 1. Flow chart of the proposed two-stage solution.

The contributions of this paper are summarized as follows.

1) The first stage is solved by transferring the energy constraint of the function to each task by using average energy preallocation. Then, each task is allocated to the ECU with the minimum earliest finish time (EFT) under its energy constraint. Compared with the latest similar algorithm in [21] that uses a pessimistic energy preallocation, the proposed RREC algorithm (Algorithm 1) uses average energy preallocation to reduce pessimism and thereby greatly reducing the response time of function under its energy constraint.
2) The second stage is solved by transferring the response-time and energy constraints of the function to each task. Then, each task is reallocated to the ECU with the maximum reliability under its response-time and energy constraints to implement the reliability-enhancement technique. Compared with the latest algorithm in [6] that merely ensures the response-time constraint, the proposed REREC algorithm (Algorithm 2) can ensure the energy and response-time constraints of function and further enhance its reliability compared with RREC.
3) Examples and experiments show that the proposed solution can effectively enhance reliability compared with its counterparts and confirm the feasibility toward functional safety goal assurance in energy-aware ACPS.

II. Related Work

Considering that the problem to be solved in this paper involves a two-stage solution, namely, reducing response time under energy constraint and reliability enhancement under energy and response-time constraints, we review recent related research according to two stages.

1) Reducing response time under energy constraint: The algorithms proposed in [15] and [22] address the problem of reducing energy consumption under response-time constraint, which is the opposite of the first stage in this paper. For example, the "upward" energy-efficient scheduling (EES) algorithm proposed in [22] and used in [23] reduces energy consumption by reclaiming the slack time of each task in the same processor, and "upward" refers to optimizing energy from exit to entry tasks. Considering the limited energy saving when using EES, Xie *et al.* [15] presented a new method called "downward" and "upward" energy consumption minimization (DUECM), where "downward" means optimizing energy from entry to exit tasks. The minimizing schedule length with energy consumption constraint (MSLECC) algorithm proposed in [21] is identical to the first stage in this paper. MSLECC transfers the energy constraint of the function to each task by using minimum energy preallocation. Then, each task is allocated to the processor with the minimum earliest finish time (EFT) under its energy constraint. However, minimum energy preallocation is too pessimistic toward a limited reduction in response time (refer to Section IV-C for details about MSLECC).

2) Enhancing reliability under energy constraint: As stated in ISO 26262, random hardware failures occur unpredictably during the life cycle of a hardware element, but they follow a probability distribution. Traditionally, transient failures were modeled through Poisson distribution with average arrival rate λ [17], [24], [25]. Considering the effects of voltage and frequency scaling on transient failures [26], the average rate depends on system processing frequency and supply voltage. Zhu *et al.* [26] built a relationship model between energy and reliability. Based on this relationship model, Zhang *et al.* [27] solved the problem of maximizing reliability of a distributed function under its energy consumption constraint in heterogeneous systems using the reliability maximum energy conservative method. Xiao *et al.* [28] improved the proposed method in [27] by using the minimum energy preallocation.

3) Enhancing reliability under energy and response-time constraints: Besides enhancing reliability under energy constraint, enhancing reliability under response-time constraint is also studied. Xie *et al.* [6] proposed a fast functional safety verification (FFSV) method called maximizing reliability under response-time requirement (MRRR) to verify whether the functional safety goal of a distributed automotive function can be assured. However, [6] does not take the energy constraint into consideration, such that it is not suitable for energy-aware ACPS. Zhao *et al.* [18] solved the problem of reliability enhancement of a distributed function under energy and response-time constraints in a uniprocessor. However, the system platform in our study is ACPS, which are heterogeneous distributed embedded systems.

Through the above review, we have the following summary. 1) For heterogeneous systems, the problems of reducing response time under energy constraint, enhancing reliability under energy constraint, and enhancing reliability under response-time constraint for a distributed function have been studied. 2) The problem of enhancing reliability under energy and response-time constraint has been studied in [18], but only for uniprocessor. In the following, we will solve the problem of enhancing reliability under energy and response-time constraint for a distributed function in ACPS. We would like to provide a new design reference for system designers and developers in this area.

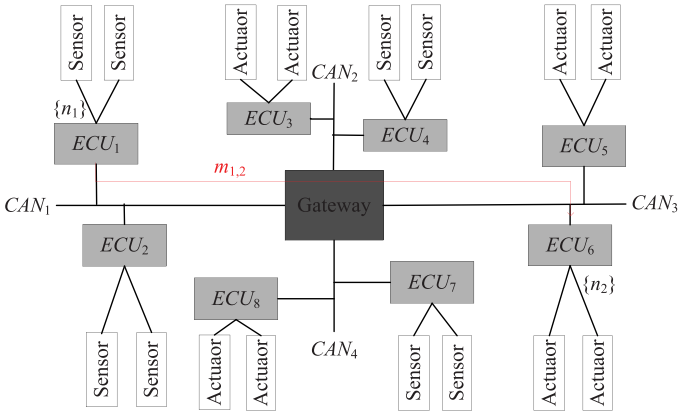


Fig. 2. ACPS architecture [4].

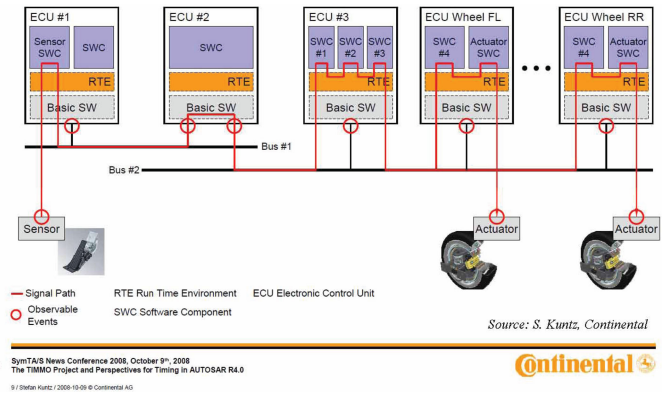


Fig. 3. Brake-by-wire function [5].

III. Models

A. ACPS Architecture

A new generation ACPS architecture is shown in Fig. 2, which is an integrated architecture, where four CAN buses are integrated by the central gateway [4]. In this architecture, several ECUs connect to several sensors, and other ECUs connect to several actuators because physical processes are composed of many parallel processes [4]. Partial ECUs can release the function by receiving collected data from sensors, and other partial ECUs can complete the function by sending the performing action to the actuators [4]. Fig. 3 shows the end-to-end execution process of the brake-by-wire function, which involves one sensor, two CAN buses, multiple ECUs, and two auctors [5].

As pointed out in Section I-A, the brake-by-wire function in Fig. 3 is represented by a DAG. Fig. 2 illustrates the simple execution process of an automotive function: ECU_1 receives the data from the sensor to trigger the first task (i.e., the entry task of the function), which is represented as n_1 in DAG. After n_1 is executed completely in ECU_1, a message $m_{1,2}$ is sent from

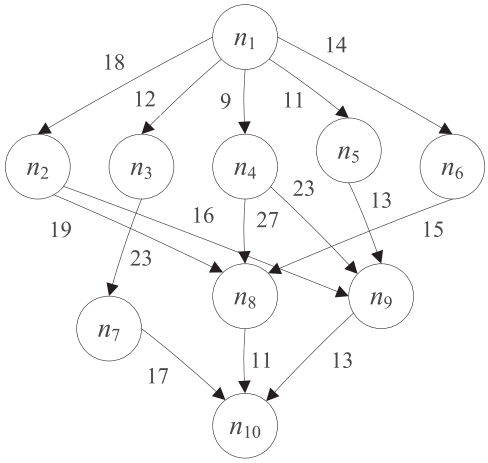


Fig. 4. Motivational distributed function [6].

n_1 to its successor task n_2, which will be executed in ECU_6. $m_{1,2}$ is transmitted in two CAN buses CAN_1 and CAN_3. After triggering a series of tasks with precedence constraints, the final task (i.e., the exit task) is allocated to an ECU, and completes the process by sending the performing action to actuators.

B. Function Model

On the basis of the architecture and execution process analysis, we let $U = \{u_1, u_2, \ldots, u_{|U|}\}$ represent a set of heterogeneous ECUs in the platform, where $|U|$ represents the size of set U. For any set X, this paper uses $|X|$ to denote its size. Meanwhile, a motivating example of distributed automotive function is shown in Fig. 4, and the DAG parameters $G = (N, W, M, C)$ are explained as follows:

1) N represents a set of nodes in G, and each node $n_i \in N$ represents a task. Fig. 4 shows a motivational function with 10 tasks. $\text{pred}(n_i)$ represents the set of the immediate predecessor tasks of n_i; considering the example in Fig. 4, we have $\text{pred}(n_8) = \{n_2, n_4, n_6\}$. $\text{succ}(n_i)$ represents the set of the immediate successor tasks of n_i; for example, we have $\text{succ}(n_8) = \{n_{10}\}$. The task that has no predecessor tasks is denoted as n_{entry} (i.e., n_1 in the example), and the task that has no successor tasks is denoted as n_{exit} (i.e., n_{10} in the example). If there are multiple entry or exit tasks in the function, then we add an extra entry or exit task with the edge value of 0 into it.
2) W is an $|N| \times |U|$ matrix, where $w_{i,k}$ denotes the worst case execution time (WCET) of n_i running in u_k with the maximum frequency. Each task $n_i \in N$ has different WCET values in different ECUs due to the heterogeneity of ECUs [6]. The WCET of a task is the maximum execution time among all possible real execution time values when the task is executed in a specific ECU. If task n_i can not be allocated in u_k because some ECUs can only execute partial tasks, it is assumed that $w_{i,k}$ is infinity (i.e., $w_{i,k} = +\infty$), such that n_i will not be allocated to u_k. Through such treatment, we can solve the problem that certain tasks can only be allocated to particular ECUs. In this paper, we assume that all WCETs of the tasks are known and determined through the WCET analysis [29]. Table I lists the WCETs of each task in 3 ECUs $\{u_1, u_2, u_3\}$. The weight 16 of n_1 and u_2 in Table I represents the WCET of n_1 in u_2, and is denoted by $w_{1,2} = 16$.
3) Communication between tasks mapped to different ECUs is performed through message passing over the bus [6]. M is a set of communication edges, and each edge $m_{i,j} \in M$ represents communication message from n_i to n_j. Accordingly, $c_{i,j} \in C$ represents the worst case response time (WCRT) of $m_{i,j}$. The WCRT of a message is the maximum response time among all possible real response-time values when the message is transmitted in a specific bus. The WCRTs in this paper are theoretical upper bounds. The reason is that finding the exact WCRT is a combinatorial explosion problem. Therefore, the general WCRT analysis is to estimate a tight WCRT upper bound, which is larger than or equal to the exact WCRT, within a pseudo-polynomial computational time [30]. We omit the WCRT calculation of the message, but directly assume that it is known in this paper for simplicity. Note that if n_i and n_j are allocated to the same ECU, then $c_{i,j}$ is set to 0 because of the shared memory mechanism in the same ECU. For example, the weight 18 of the edge between n_1 and n_2 in Fig. 4 represents the WCRT of $m_{1,2}$ is denoted by $c_{1,2} = 18$ if n_1 and n_2 are not allocated to the same ECU.
4) Scheduling can be either preemptive or nonpreemptive according to the AUTOSAR standard [4], [6]. This study uses the non-preemptive scheduling to maintain consistency with the message scheduling in the CAN buses.

TABLE I
WCETs OF TASKS IN DIFFERENT ECUs OF THE MOTIVATIONAL DISTRIBUTED FUNCTION [6].

Task	u_1	u_2	u_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11
n_8	5	11	14
n_9	18	12	20
n_{10}	21	7	16

IV. Response-Time Reduction Under Energy Constraint

This section refers to the first stage and solves the problem of response-time reduction under energy constraint.

A. Power and Energy

We can change (scale) down the frequency to change down the voltage (or change up the voltage to change up the frequency) in CMOS circuits because there is a nearly linear relationship between frequency and voltage [15]. The lower the frequency

is, the lower the voltage becomes. There is a nearly linear relationship between voltage and power. The lower the voltage is, the lower the power becomes. Hence, we can change the frequency to change the power, thereby changing the energy consumption. In this work, we adopt the system-level power model proposed in [26] and then used in [15], [18], and [21], where power consumption with frequency f is given by

$$P(f) = P_s + h(P_{\text{ind}} + P_d) = P_s + h\left(P_{\text{ind}} + C_{\text{ef}} f^m\right). \quad (1)$$

P_s is the static power, and the dynamic power is composed of two parts: P_{ind} and P_d. P_{ind} is the frequency-independent dynamic power that is a constant. P_d is the frequency-dependent dynamic power that varies according to the frequency. h represents the system state, where $h = 1$ means the system is active and $h = 0$ means the system is turned OFF. C_{ef} is effective switching capacitance, and m is the dynamic power exponent and is no smaller than 2. The static power is unmanageable and is not the main factor. We disregarded it in this paper and mainly considered the dynamic part. When the processing frequency is slowed down, the execution time of tasks increases, such that low frequency may not result in low energy consumption. Thus, a minimum energy efficient frequency f_{ee} exists [15], [18], [21], [26], and it is denoted by

$$f_{\text{ee}} = \sqrt[m]{\frac{P_{\text{ind}}}{(m-1)C_{\text{ef}}}}. \quad (2)$$

Assuming that the frequency of an ECU varies from minimum available frequency f_{min} to maximum frequency f_{max}, the lowest energy-efficient frequency to execute a task is

$$f_{\text{low}} = \max(f_{\text{min}}, f_{\text{ee}}). \quad (3)$$

Therefore, any actual effective frequency f_h should belong to the scope of $f_{\text{low}} \leqslant f_h \leqslant f_{\text{max}}$.

Considering that the number of ECUs is $|U|$ in ACPS and these ECUs are completely heterogeneous, each ECU should have individual power parameters [15], [21], and we define them as follows. The frequency-independent dynamic power set is

$$\{P_{1,\text{ind}}, P_{2,\text{ind}}, \ldots, P_{|U|,\text{ind}}\}.$$

The frequency-dependent dynamic power set is

$$\{P_{1,d}, P_{2,d}, \ldots, P_{|U|,d}\}.$$

The effective switching capacitance set is

$$\{C_{1,\text{ef}}, C_{2,\text{ef}}, \ldots, C_{|U|,\text{ef}}\}.$$

The dynamic power exponent set is

$$\{m_1, m_2, \ldots, m_{|U|}\}.$$

The lowest energy-efficient frequency set is

$$\{f_{1,\text{low}}, f_{2,\text{low}}, \ldots, f_{|U|,\text{low}}\}$$

and the actual effective frequency set is

$$\left\{ \begin{array}{c} \{f_{1,\text{low}}, f_{1,\alpha}, f_{1,\beta}, \ldots, f_{1,\text{max}}\}, \\ \{f_{2,\text{low}}, f_{2,\alpha}, f_{2,\beta}, \ldots, f_{2,\text{max}}\}, \\ \cdots, \\ \{f_{|U|,\text{low}}, f_{|U|,\alpha}, f_{|U|,\beta}, \ldots, f_{|U|,\text{max}}\} \end{array} \right\}.$$

We let $E(n_i, u_k, f_{k,v})$ represent the energy consumption of task n_i in ECU u_k with frequency $f_{k,v}$ and calculate it as

$$E(n_i, u_k, f_{k,v}) = (P_{k,\text{ind}} + C_{k,\text{ef}} \times {f_{k,v}}^{m_k}) \times w_{i,k,v} \quad (4)$$

where

$$w_{i,k,v} = w_{i,k} \times \frac{f_{k,\text{max}}}{f_{k,v}}.$$

Finally, the energy consumption of the function is the sum of the energy consumptions of all tasks.

$$E(G) = \sum_{i=1}^{|N|} E(n_i) = \sum_{i=1}^{|N|} E(n_i, u_{\text{pr}(i)}, f_{\text{pr}(i),\text{hz}(i)}) \quad (5)$$

where $u_{\text{pr}(i)}$ and $f_{\text{pr}(i),\text{hz}(i)}$ represent the allocated ECU and frequency of n_i, respectively.

Given that the energy consumption of function G is the sum of that of each task, we can obtain the minimum and maximum energy consumption values of function G as

$$E_{\min}(G) = \sum_{i=1}^{|N|} E_{\min}(n_i) = \sum_{i=1}^{|N|} \min_{u_k \in U} E(n_i, u_k, f_{\text{k,low}}) \quad (6)$$

and

$$E_{\max}(G) = \sum_{i=1}^{|N|} E_{\max}(n_i) = \sum_{i=1}^{|N|} \max_{u_k \in U} E(n_i, u_k, f_{\text{k,max}}) \quad (7)$$

respectively. $E_{\min}(n_i)$ and $E_{\max}(n_i)$ represent the minimum and maximum energy consumption values, respectively, and can be obtained by traversing all the ECUs with the lowest and maximum energy-efficient frequencies, respectively.

We assume that the energy constraint of function G is $E_{\text{cons}}(G)$. Then, it should be larger than or equal to $E_{\min}(G)$; otherwise, $E_{\text{cons}}(G)$ is always not assured. Meanwhile, $E_{\text{cons}}(G)$ should be less than or equal to $E_{\max}(G)$; otherwise, $E_{\text{cons}}(G)$ is always assured. Hence, this paper assumes that $E_{\text{cons}}(G)$ belongs to the scope of $E_{\min}(G)$ and $E_{\max}(G)$, namely

$$E_{\min}(G) \leqslant E_{\text{cons}}(G) \leqslant E_{\max}(G).$$

B. Problem Statement

Consider a distributed automotive function with known energy constraint in ACPS with heterogeneous ECUs, which support different frequencies by DVFS. Then, the problem to be addressed in this section is to allocate an ECU with a proper frequency to each task to reduce the response time of the function under its energy constraint. Let AFT(n_{exit}) represent the actual finish time (AFT) of the exit task. Considering that the AFT of the n_{exit} is response time of the function G, the formal

TABLE II
UPWARD RANK VALUES OF TASKS OF THE MOTIVATIONAL DISTRIBUTED FUNCTION IN FIG. 4 [6], [15], [21], [32]

Task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
$rank_u(n_i)$	108	77	80	80	69	63.3	42.7	35.7	44.3	14.7

description is to find the ECU and frequency allocation of all tasks to reduce the response time of the function

$$\mathrm{RT}(G) = \mathrm{AFT}(n_{\mathrm{exit}})$$

subject to its energy constraint

$$E(G) = \sum_{i=1}^{|N|} E(n_i) = \sum_{i=1}^{|N|} E(n_i, u_{\mathrm{pr}(i)}, f_{\mathrm{pr}(i),\mathrm{hz}(i)}) \leqslant E_{\mathrm{cons}}(G)$$

and the frequency constraint

$$f_{\mathrm{pr}(i),\mathrm{low}} \leqslant f_{\mathrm{pr}(i),\mathrm{hz}(i)} \leqslant f_{\mathrm{pr}(i),\max}$$

for all $i : 1 \leqslant i \leqslant |N|, u_{\mathrm{pr}(i)} \in URT(G)$ represents the response time of the function.

C. Minimum Energy Preallocation

The MSLECC algorithm is identical to the first stage in this paper. Considering that scheduling tasks with a quality of service constraint for optimality in multiprocessors is an NP-hard optimization problem [6], [21], [31], MSLECC implements heuristic list scheduling.

MSLECC transfers the energy constraint of the function to each task based on the following strategy.

Assuming that the task to be allocated is $n_{s(y)}$, where $n_{s(y)}$ represents the yth allocated task, then $\{n_{s(1)}, n_{s(2)}, \ldots, n_{s(y-1)}\}$ represents the task set where the tasks have been allocated, and $\{n_{s(y+1)}, n_{s(y+2)}, \ldots, n_{s(|N|)}\}$ represents the task set where the tasks have not been allocated. Initially, all tasks of the function are unallocated. Tasks are ordered according to the decreasing order of the upward rank value ($\mathrm{rank_u}$) [32]

$$\mathrm{rank_u}(n_i) = \overline{w_i} + \max_{n_j \in \mathrm{succ}(n_i)} \{c_{i,j} + \mathrm{rank_u}(n_j)\} \tag{8}$$

which is considered the *de facto* prioritizing task criterion for DAG-based list scheduling in heterogeneous systems because it is widely used in reliability-aware and energy-aware scheduling [6], [15], [21], [22]. Table II shows the upward rank value of tasks for the motivational function (Fig. 4), and we find that the task allocation order is $\{n_1, n_3, n_4, n_2, n_5, n_6, n_9, n_7, n_8, n_{10}\}$.

To ensure that the energy constraint of the function is assured in each task allocation, the MSLECC algorithm proposed in [21] presents the preallocation of each unallocated task $n_{s(z)}$ in $\{n_{s(y+1)}, n_{s(y+2)}, \ldots, n_{s(|N|)}\}$ for all tasks.

In MSLECC, the energy preallocation value of $n_{s(z)}$ is $E_{\min}(n_{s(z)})$. Therefore, when allocating $n_{s(y)}$, the energy consumption value of the function is expressed as

$$E(G) = \sum_{x=1}^{y-1} E(n_{s(x)}) + E(n_{s(y)}) + \sum_{z=y+1}^{|N|} E_{\min}(n_{s(z)}).$$

Then, $E(G)$ must be less than or equal to $E_{\mathrm{cons}}(G)$ according to the problem statement. We have

$$\sum_{x=1}^{y-1} E(n_{s(x)}) + E(n_{s(y)}) + \sum_{z=y+1}^{|N|} E_{\min}(n_{s(z)}) \leqslant E_{\mathrm{cons}}(G).$$

Therefore, the actual energy consumption value of the task $n_{s(y)}$ should have the following constraint:

$$E(n_{s(y)}) \leqslant E_{\mathrm{cons}}(G) - \sum_{x=1}^{y-1} E(n_{s(x)}) - \sum_{z=y+1}^{|N|} E_{\min}(n_{s(z)}).$$

The energy constraint of task $n_{s(y)}$ is

$$E_{\mathrm{cons}}(n_{s(y)}) = E_{\mathrm{cons}}(G) - \sum_{x=1}^{y-1} E(n_{s(x)}) - \sum_{z=y+1}^{|N|} E_{\min}(n_{s(z)}). \tag{9}$$

Eq. (9) indicates that the energy constraint of the function is transferred to each task. As long as each task assures its energy constraint of

$$E(n_{s(y)}) \leqslant E_{\mathrm{cons}}(n_{s(y)})$$

the energy constraint of the function can also be assured.

The limitation is that the unallocated low-priority tasks are preallocated with the minimum energy, such that the energy constraints of the high-priority tasks become large. In other words, high-priority tasks consume massive energy, thereby causing low-priority tasks to select ECUs with minimum energy but with long execution time. Consequently, low-priority tasks can prolong the response time of the function. Therefore, MSLECC is severely pessimistic toward unfair energy usage among tasks and thus results in limited reduction of the response time. Therefore, a more objective and effective algorithm is required.

D. Average Energy Preallocation

Given the limitation of pessimistic energy preallocation to unallocated tasks using MSLECC, this section presents an improved energy preallocation to reduce the response time by adopting average energy preallocation. For convenience, we first define available energy.

Definition 1: (*Available Energy*). Available energy is the difference between the energy constraint and the minimum energy consumption of function G

$$\Delta E_{\mathrm{ae}}(G) = E_{\mathrm{cons}}(G) - E_{\min}(G). \tag{10}$$

An average energy preallocation of $n_{s(z)}$ is described as

$$E_{\mathrm{avg}}(n_{s(z)}) = E_{\min}(n_{s(z)}) + \frac{\Delta E_{\mathrm{ae}}(G)}{|N|}. \tag{11}$$

The main improvement is that the energy preallocation of $n_{s(z)}$ is changed from $E_{\min}(n_{s(z)})$ to $E_{\mathrm{avg}}(n_{s(z)})$; thus, $E_{\min}(n_{s(z)})$ is changed to $E_{\mathrm{avg}}(n_{s(z)})$ when calculating

Algorithm 1: The RREC Algorithm.

Input: $G = (N, W, M, C)$, U, $E_{\text{cons}}(G)$
Output: $E(G)$, $\text{RT}(G)$ and its related values
1: Sort the tasks by descending order of rank_u values;
2: Calculate $E_{\text{min}}(G)$ using (6);
3: Calculate $E_{\text{ae}}(G)$ using (10);
4: **for** $(y \leftarrow 1; y \leq |N|; y{+}{+})$ **do**
5: Calculate $E_{\text{cons}}(n_{s(y)})$ using (12);
6: **for** (each ECU $u_k \in U$) **do**
7: **for** (each frequency $f_{k,v}$ in $f_{k,\text{max}}$ and $f_{k,\text{low}}$) **do**
8: Calculate $E\left(n_{s(y)}, u_k, f_{k,v}\right)$ using (4);
9: Calculate $\text{EFT}(n_{s(y)}, u_k, f_{k,v})$ using (14);
10: **end for**
11: **end for**
12: Allocate the current task $n_{s(y)}$ to the ECU that has the minimum EFT under the energy constraint of $E(n_{s(y)}, u_k, f_{k,v}) \leqslant E_{\text{cons}}(n_{s(y)})$;
13: **end for**
14: Calculate $E(G)$ using (5);
15: $\text{RT}(G) \leftarrow \text{AFT}(n_{\text{exit}})$;

$E_{\text{cons}}(n_{s(y)})$

$$E_{\text{cons}}(n_{s(y)}) = E_{\text{cons}}(G) - \sum_{x=1}^{y-1} E(n_{s(x)}) - \sum_{z=y+1}^{|N|} E_{\text{avg}}(n_{s(z)}). \tag{12}$$

That is, we distribute the available energy evenly to each unallocated task. The final energy constraint of $n_{s(y)}$ is shown in (12). In this manner, the energy constraint of the function can still be transferred to each task. We prove the correctness in Theorem 1.

E. Response-time Reduction

After transferring the energy constraint of the function to each task, the task is allocated to the ECU with the minimum EFT under the energy constraint of each task.

We let $\text{EST}(n_{s(y)}, u_k)$ represent the earliest start time (EST) of task $n_{s(y)}$ in ECU u_k:

$$\begin{cases} \text{EST}(n_{\text{entry}}, u_k) = 0 \\ \text{EST}(n_{s(y)}, u_k) = \\ \quad \max\left\{\text{avail}[k], \max\limits_{n_h \in \text{pred}(n_{s(y)})} \left\{\text{AFT}(n_h) + c'_{h,s(y)}\right\}\right\} \end{cases} \tag{13}$$

where $\text{avail}[k]$ is the earliest available time when ECU u_k is ready for task execution. For the entry task, $\text{EST}(n_{s(y)}, u_k) = 0$. $\text{AFT}(n_h)$ is the AFT of task n_h, and $c'_{h,s(y)}$ represents the communication time between n_h and $n_{s(y)}$. $c'_{h,s(y)} = c_{h,s(y)}$ only when n_h and $n_{s(y)}$ are allocated to different ECUs; otherwise, $c'_{h,s(y)} = 0$. Then, EFT is calculated as

$$\text{EFT}(n_{s(y)}, u_k, f_{k,v}) = \text{EST}(n_{s(y)}, u_k, f_{k,v}) + w_{s(y),k,v}. \tag{14}$$

The value of $\text{EFT}(n_{s(y)}, u_k, f_{k,v})$ is the task allocation criterion because it can meet the local optimal of $n_{s(y)}$.

TABLE III
POWER AND FREQUENCY PARAMETERS OF THE ECUS

u_k	$P_{k,\text{ind}}$	$C_{k,\text{ef}}$	m_k	$f_{k,\text{low}}$	$f_{k,\text{max}}$
u_1	0.03	0.8	2.9	0.26	1.0
u_2	0.04	0.8	2.5	0.26	1.0
u_3	0.07	1.0	2.5	0.29	1.0

The strategy of response-time reduction under energy constraint is as follows: we simply need to traverse all the ECUs and frequencies to allocate the current task $n_{s(y)}$ to the ECU that has the minimum EFT under the energy constraint.

$$\text{EFT}(n_{s(y)}, u_{\text{pr}(s(y))}, f_{\text{pr}(s(y)),\text{hz}(s(y))}) = \min_{\substack{u_k \in U,\ f_{k,v} \in [f_{k,\text{low}}, f_{k,\text{max}}], \\ E(n_{s(y)}, u_k, f_{k,v}) \leqslant E_{\text{cons}}(n_{s(y)})}} \left\{\text{EFT}(n_{s(y)}, u_k, f_{k,v})\right\}. \tag{15}$$

F. RREC Algorithm

In the analysis of Sections IV-D and IV-E, the RREC algorithm is presented to reduce the response time of a function under its energy constraint, as shown in Algorithm 1.

The main idea of RREC is that the energy constraint of the function is transferred to each task by average energy preallocation. Then, each task merely selects the ECU and frequency with the minimum EFT under its energy constraint. The main advantages are as follows.

1) In Line 1, RREC sorts tasks in a descending order of rank_u values.
2) In Lines 4–13, RREC iteratively schedules each task of the function according to the priority of each task.
3) In Line 5, RREC obtains the energy constraint of the current task by using (12) before it is allocated.
4) In Lines 6–11, RREC traverses all ECU and frequency combinations to select the combination with the minimum EFT under its energy constraint.
5) In Lines 14 and 15, RREC calculates the actual energy consumption and response-time values of the function, respectively.

The time complexity of RREC is analyzed as follows: 1) traversing all tasks requires $O(|N|)$ time (Lines 4–13); and 2) calculating $\text{EFT}(n_i, u_k, f_{k,v})$ requires $O(|N| \times |U| \times |F|)$, where $|F|$ represents the maximum number of discrete frequencies from the lowest to the maximum frequencies (Lines 6–11). Therefore, the time complexity of the RREC algorithm is $O(|N|^2 \times |U| \times |F|)$. That is, RREC is a low complexity algorithm.

G. Task Mapping Based on RREC

This section provides an example to show task mapping using the RREC algorithm. Considering the motivational distributed function in Fig. 4, the parameters of each ECU are shown in Table III. In this example, the maximum frequency $f_{k,\text{max}}$ for each ECU is 1.0, and the frequency resolution is set to 0.01. All the parameter units are omitted for simplicity in this example.

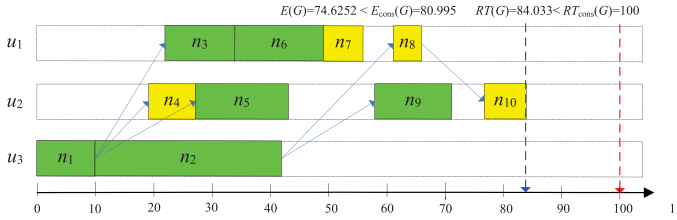


Fig. 5. RREC-generated task mapping of the motivational function in Fig. 4.

TABLE IV
SCHEDULING RESULTS OF THE MOTIVATIONAL FUNCTION USING MSLECC AND RREC

Algorithm	$E(G)$	$RT(G)$
MSLECC	80.9939	129.3660
RREC	74.6252	84.0330

We calculate the minimum and maximum energy consumption values as $E_{\min}(G) = 20.31$ and $E_{\max}(G) = 161.99$ according to (6) and (7), respectively. We set the energy constraint of G as $E_{\text{cons}}(G) = 0.5 \times E_{\max}(G) = 80.995$. Then, we obtain the mapping results shown in Fig. 5 by using RREC. The actual energy consumption of the function is 74.6252, which is lower than $E_{\text{cons}}(G)$. Meanwhile, the response time is 84.033.

Table IV shows the actual energy consumption and final response time obtained by using RREC and MSLECC algorithms, respectively. Both RREC and MSLECC can assure the energy constraint, but RREC has a shorter response time than MSLECC.

V. RELIABILITY ENHANCEMENT UNDER ENERGY AND RESPONSE-TIME CONSTRAINTS

This section refers to the second stage and solves the problem of reliability enhancement under energy and response-time constraints.

A. Reliability Model

As stated in the automotive functional safety standard ISO 26262, random hardware failures occur unpredictably during the life cycle of a hardware element but follow a probability distribution. Exposure refers to the relative expected frequency of operational conditions in which hazardous events may occur [11]. In ISO 26262, exposure involves five levels, namely, E0, E1, E2, E3, and E4, where E0 represents the lowest level (i.e., incredibly unlikely) and E4 represents the highest level (i.e., high probability; injury could occur under most operating conditions) [11]. Exposure illustrates the possibility of hazardous events occurring and can affect reliability, which is inversely expressed as exposure (i.e., reliability = 1 – exposure) [6]. Correspondingly, the reliability goals for exposures are shown in Table V [6]. To reduce the possibility of hazardous events caused by exposure, we must enhance reliability to assure the reliability goal, thereby assuring the functional safety goal.

TABLE V
CLASSES OF PROBABILITY OF EXPOSURE REGARDING DURATION/PROBABILITY OF EXPOSURE IN ISO 26262 [6], [11]

Exposure level		Probability of exposure	Reliability goal
E1	Very low probability	Not specified	At least exceeds 0.99
E2	Low probability	$< 1\%$	0.99
E3	Medium probability	[1%, 10%]	>0.9
E4	High probability	$> 10\%$	<=0.9

In a DVFS-capable ACPS, different frequencies possess different failure rates according to relevant research summaries [18]. Hence, we let $\lambda_{k,\max}$ represent the failure rate of ECU u_k with the maximum frequency. Given that the failure rate follows the Poisson distribution, the failure rate $\lambda_{k,v}$ of u_k with frequency $f_{k,v}$ is calculated by [18]

$$\lambda_{k,v} = \lambda_{k,\max} 10^{\frac{d(f_{k,\max} - f_{k,v})}{f_{k,\max} - f_{k,\min}}} \tag{16}$$

where d is a constant that represents the sensitivity of failure rates to voltage scaling.

We then build the relationship between reliability and frequency according to (16), namely, the reliability of task n_i executed in ECU u_k with frequency $f_{k,v}$ calculated as

$$\begin{aligned} R(n_i, u_k, f_{k,v}) &= e^{-\lambda_{k,v} \times \frac{w_{i,k} \times f_{k,\max}}{f_{k,v}}} \\ &= e^{-\lambda_{k,\max} 10^{\frac{d(f_{k,\max} - f_{k,v})}{f_{k,\max} - f_{k,\min}}} \times \frac{w_{i,k} \times f_{k,\max}}{f_{k,v}}}. \end{aligned} \tag{17}$$

Eq. (17) indicates that the relationship between reliability and frequency is monotonically increasing in the same ECU. The function's reliability is correspondingly adjusted as [6], [18]

$$R(G) = \prod_{n_i \in N} R(n_i) = \prod_{n_i \in N} R\left(n_i, u_{\text{pr}(i)}, f_{\text{pr}(i),\text{hz}(i)}\right) \tag{18}$$

where $u_{\text{pr}(i)}$ and $f_{\text{pr}(i),\text{hz}(i)}$ represent the allocated ECU and frequency of n_i, respectively.

B. Problem Statement

Consider a distributed automotive function with known energy constraint in ACPS with heterogeneous ECUs, which support different frequencies by DVFS. The problem to be addressed in this section is to allocate an ECU with a proper frequency to each task to enhance the reliability of the function under its energy and response-time constraints. The formal description is to find the ECU and frequency allocation of all tasks to enhance the reliability of the function

$$R(G) = \prod_{n_i \in N} R\left(n_i\right) = \prod_{n_i \in N} R\left(n_i, u_{\text{pr}(i)}, f_{\text{pr}(i),hz(i)}\right)$$

under its energy constraint

$$E(G) = \sum_{i=1}^{|N|} E(n_i) \leqslant E_{\text{cons}}(G),$$

its response-time constraint

$$RT(G) \leqslant RT_{\text{cons}}(G)$$

and the frequency constraint

$$f_{\text{pr}(i),\text{low}} \leqslant f_{\text{pr}(i),\text{hz}(i)} \leqslant f_{\text{pr}(i),\max}$$

for all $i : 1 \leqslant i \leqslant |N|, u_{\text{pr}(i)} \in U$.

C. Energy Constraint of the Task

Considering that this section tackles the enhancement of reliability under energy and response-time constraints, we make full use of the slack of $\text{RT}_{\text{cons}}(G)$-$\text{RT}(G)$ shown in Fig. 5 to enhance reliability. We implement the objective by reallocating the tasks from the exit task to the entry task.

When allocating $n_{s(y)}$, we express the energy consumption of the function as follows:

$$E(G) = \sum_{x=1}^{y-1} E_{\text{RREC}}(n_{s(x)}) + E(n_{s(y)}) + \sum_{z=y+1}^{|N|} E_{\text{REREC}}(n_{s(z)})$$

where $E_{\text{RREC}}(n_{s(x)})$ represents the RREC-generated energy of $n_{s(x)}$ and $E_{\text{REREC}}(n_{s(z)})$ represents the REREC-generated energy of $n_{s(z)}$. The REREC algorithm will be proposed in the following text.

The actual $E(G)$ must be less than or equal to $E_{\text{cons}}(G)$ according to the problem statement. We have

$$\sum_{x=1}^{y-1} E_{\text{RREC}}(n_{s(x)}) + E(n_{s(y)}) + \sum_{z=y+1}^{|N|} E_{\text{REREC}}(n_{s(z)}) \leqslant E_{\text{cons}}(G).$$

Therefore, the actual energy for task $n_{s(y)}$ should have the following constraint:

$$E(n_{s(y)}) \leqslant E_{\text{cons}}(G) - \sum_{x=1}^{y-1} E_{\text{RREC}}(n_{s(x)}) - \sum_{z=y+1}^{|N|} E_{\text{REREC}}(n_{s(z)}).$$

We let the energy constraint for current task $n_{s(y)}$ be

$$E_{\text{cons}}(n_{s(y)}) = E_{\text{cons}}(G) - \sum_{x=1}^{y-1} E_{\text{RREC}}(n_{s(x)}) - \sum_{z=y+1}^{|N|} E_{\text{REREC}}(n_{s(z)}). \quad (19)$$

Therefore, the energy constraint of the function is transferred to each task. As long as each task assures its energy constraint in

$$E(n_{s(y)}) \leqslant E_{\text{cons}}(n_{s(y)}) \quad (20)$$

then the energy constraint of the function can also be assured.

D. Response-Time Constraint of the Task

Similar to the energy constraint, the response-time constraint of the function is also transferred to each task.

1) We clear the RREC-generated task allocation of the current task n_{10} to implement reallocation and enhance reliability. To avoid violating the response-time constraint of the function, we do not change the task allocations of other tasks when calculating the new reliability value of the current task.

2) We define the latest finish time (LFT) of the current task. LFT is restricted by its successor tasks due to the precedence constraints among tasks and is calculated as follows:

$$\begin{cases} \text{LFT}(n_{\text{exit}}, u_k) = \text{RT}_{\text{cons}}(G) \\ \text{LFT}(n_i, u_k) = \\ \quad \min\left\{ \min_{n_j \in \text{succ}(n_i)} \left\{ AST(n_j) - c'_{i,j} \right\}, AS_{\text{end}}(n_i, u_k) \right\} \end{cases}. \quad (21)$$

$\text{AST}(n_j)$ represents the actual start time (AST) of n_j. $AS_{\text{end}}(n_i, u_k)$ represents the end instant of available slack of n_i in u_k. Notably, n_i has individual $\text{LFT}(n_i, u_k)$ in different ECUs.

3) Similar to $\text{LFT}(n_i, u_k)$, we need to obtain $\text{EST}(n_i, u_k)$ due to the precedence constraints with its predecessors. Each task has individual $\text{EST}(n_i, u_k)$ in different ECUs as follows:

$$\begin{cases} EST(n_{\text{entry}}, u_k) = 0 \\ EST(n_i, u_k) = \\ \quad \max\left\{ \max_{n_h \in \text{pred}(n_i)} \left\{ AFT(n_h) + c'_{h,i} \right\}, AS_{\text{start}}(n_i, u_k) \right\} \end{cases}. \quad (22)$$

$AS_{\text{start}}(n_i, u_k)$ represents the start instant of available slack of n_i in u_k. In other words, the response-time constraint of the task is the combination of $\text{EST}(n_i, u_k)$ and $\text{LFT}(n_i, u_k)$. The ESTs and LFTs of n_{10} in all ECUs in Fig. 5 are as follows:

$$\begin{cases} EST(n_{10}, u_1) = 55.8343 \\ EST(n_{10}, u_2) = 66.033 \\ EST(n_{10}, u_3) = 83.7989 \end{cases} \quad \begin{cases} LFT(n_{10}, u_1) = 100 \\ LFT(n_{10}, u_2) = 100 \\ LFT(n_{10}, u_3) = 100 \end{cases}.$$

E. Reliability Enhancement

After deciding the energy and response-time constraints of each task, we reallocate task $n_{s(y)}$ to the ECU with the maximum reliability under its energy and response-time constraints. That is,

$$R(n_{s(y)}) = R\left(n_{s(y)}, u_{\text{pr}(s(y))}, f_{\text{pr}(s(y)),\text{hz}(s(y)))}\right) = \max_{\substack{u_k \in U, f_{k,\text{low}} \leqslant f_{k,v} \leqslant f_{k,\max} \\ E(n_{s(y)}, u_k, f_{k,v}) \leqslant E_{\text{cons}}(n_{s(y)}) \\ w_{s(y),k,v} \leqslant \text{MET}(n_{s(y)}, u_k)}} \{R(n_{s(y)}, u_k)\} \quad (23)$$

where $\text{MET}(n_{s(y)}, u_k)$ represents the maximum execution time of $n_{s(y)}$ in u_k and is expressed as

$$\text{MET}(n_{s(y)}, u_k) = \text{LFT}(n_{s(y)}, u_k) - \text{EST}(n_{s(y)}, u_k)). \quad (24)$$

The energy constraint is

$$E(n_{s(y)}, u_k) \leqslant E_{\text{cons}}(n_{s(y)})$$

and the response-time constraint is

$$w_{s(y),k} \leqslant \left(\text{LFT}(n_{s(y)}, u_k) - \text{EST}(n_{s(y)}, u_k)\right).$$

Algorithm 2: The REREC Algorithm.

Input: $G = (N, W, M, C)$, U, $E_{\text{cons}}(G)$, $RT_{\text{cons}}(G)$, RREC-generated mapping
Output: $E(G)$, $R(G)$

1: Sort the tasks by ascending order of rank_u values.
2: **for** $(y \leftarrow |N|; y \geq 1; y--)$ **do**
3: Calculate $E_{\text{cons}}(n_{s(y)})$ using (19);
4: **for** (each ECU $u_k \in U$) **do**
5: Calculate $\text{LFT}(n_{s(y)}, u_k)$ using (21);
6: Calculate $\text{EST}(n_{s(y)}, u_k)$ using (22);
7: Calculate $\text{MET}(n_{s(y)}, u_k)$ using (24);
8: **for** (each frequency $f_{k,v}$ in $f_{k,\max}$ and $f_{k,\text{low}}$) **do**
9: Calculate $E(n_{s(y)}, u_k, f_{k,v})$ using (4);
10: Calculate $R(n_{s(y)}, u_k, f_{k,v})$ using (17);
11: **end for**
12: **end for**
13: Reallocate the task $n_{s(y)}$ to the ECU with the maximum reliability under its energy and response-time constraints;
14: **end for**
15: Calculate the actual energy consumption $E(G)$ using (5);
16: Calculate $R(G)$ using (18).

F. REREC Algorithm

Inspired by the above analysis, we propose the REREC algorithm described in Algorithm 2.

The main idea of REREC is that both the energy and response-time constraints of the function are transferred to each task. Each task merely selects the ECU and frequency combination with the maximum reliability under its energy and response-time constraints. The main advantages are explained as follows.

1) In Line 1, REREC sorts tasks in an ascending order of rank_u values.
2) In Lines 2–14, REREC iteratively schedules each task of the function according to the priority of each task.
3) In Line 3, REREC obtains the energy constraint of the current task by using (19) before it is allocated.
4) In Lines 4–13, REREC traverses all ECU and frequency combinations to select the combination with the maximum reliability values under the energy and response-time constraints.
5) In Lines 15 and 16, REREC calculates the actual energy consumption and reliability value of the function, respectively.

REREC has the same time complexity of $O(|N|^2 \times |U| \times |F|)$ as RREC. In other words, REREC is also a low complexity algorithm.

Finally, we combine RREC and REREC to form a two-stage solution. The flow chart has been presented in Fig. 1. As RREC improves previous pessimistic energy preallocation, it can greatly reduce the response time of function under its energy constraint. Based on RREC, REREC can further enhance the reliability by making full use of the slack between RREC-generated response time and response-time constraint. REREC can provide a solution toward functional safety goal assurance in energy-aware ACPS.

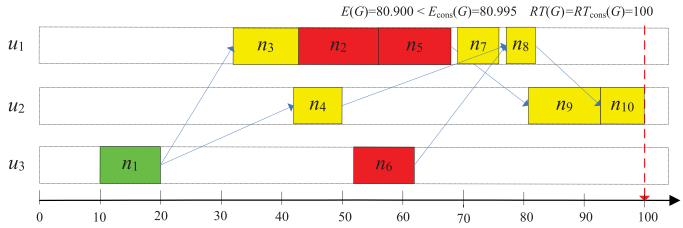


Fig. 6. REREC-generated task mapping of the motivational function in Fig. 4.

G. Task Mapping Based on REREC

This section provides an example to show the mapping results using the REREC algorithm. We assume that the power parameters for all ECUs are those shown in Table III. The failure rates for three ECUs are $\lambda_1 = 0.00025$, $\lambda_2 = 0.00030$, and $\lambda_3 = 0.00035$. The sensitivity of failure rates to voltage scaling d of each ECU is 1. Fig. 6 also shows the task mapping of motivational distributed function G using REREC, where the response time is equal to the response-time constraint, namely, $\text{RT}(G) = \text{RT}_{\text{cons}}(G) = 100$. Evidently, n_3, n_6, and n_7 (marked with red color) are reallocated to other ECUs. The actual energy of the function is $E(G) = 80.900$, which is less than but close to $E_{\text{cons}}(G) = 80.995$. The final reliability is $R(G) = 0.9729$, which is much higher than 0.9214 generated by RREC. This example confirms that REREC can implement reliability enhancement compared with RREC. Note that the AST of n_1 is 10.32 rather than 0, such that the actual response of the function is $100 - 10.32 = 89.68$.

VI. Experiments

A. Experimental Metrics

Considering that we propose a two-stage solution in this paper, it is better that each stage has similar algorithm for comparison.

1) The first stage solves the problem of response-time reduction under energy constraint by proposing the RREC algorithm. The compared algorithm is MSLECC [21] because it is the state-of-the-art algorithm to minimize the response time under energy constraint of a distributed function in heterogeneous distributed systems. In other words, MSLECC and RREC actually solve the same problem and they are worth being used to compare.
2) The second stage solves the problem of reliability enhancement under response-time and energy constraints by proposing the REREC algorithm. As pointed out in Section II, this problem is only for uniprocessor [18]. Fortunately, the MRRR (named FFSV2 in [6]) algorithm is the state-of-the-art algorithm to maximize the reliability under response-time constraint of a distributed function in heterogeneous automotive systems. Although MRRR

does not consider energy constraint, we can still analyze some results through experimental comparisons.

Given that this paper focuses on the design phase, the function parameters used in this phase are selected based on the real deployment. We use the parameter values of real ACPS as experimental data. The failure rate of each task falls within the range from 10^{-6} to 9×10^{-6} in the time unit of 1 μs, and the WCETs of the tasks and the WCRTs of the messages fall within the range of 100–400 μs. The ECU number of ACPS is 16. The maximum frequency of each ECU is $f_{k,\max} = 1$ GHz. All frequencies are discrete, and the frequency solution is 0.01 GHz. The effective switching capacitance $C_{k,\text{ef}}$ of each ECU is 1. The dynamic power exponent set m_k of each ECU is 3. The sensitivity of failure rates to voltage scaling d of each ECU is 1. The sole difference among ECUs is that they have different frequency-independent dynamic powers, which belong to the scope of 0.03 W $\leqslant P_{k,\text{ind}} \leqslant$ 0.07 W. Frequency-dependent dynamic power $P_{k,\text{d}}$ depends on frequency and is calculated by $C_{k,\text{ef}} \times {f_{k,h}}^{m_k}$. We use Java to implement and run all the algorithms based on the given parameter values to obtain experimental results.

The experimental values are obtained by executing once run for one function. Many tests with the same parameter values and scales are preformed and show the same regular pattern and relatively stable results. In other words, the experiments are repeatable and do not affect the consistency of the results.

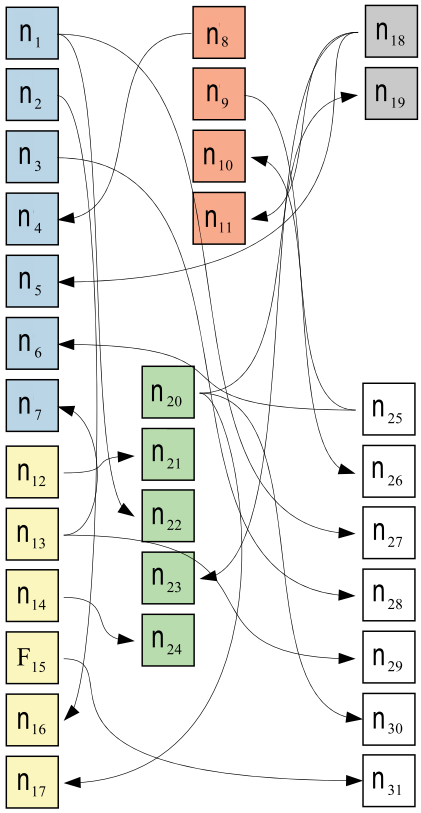


Fig. 7. Real-life automotive function.

B. Real-Life Automotive Function

We use the real-life automotive function adopted in [6], as shown in Fig. 7. This function consists of six functional blocks: engine controller with seven tasks (n_1–n_7), automatic gear box with four tasks (n_8–n_{11}), antilocking brake system with six tasks (n_{12}–n_{17}), wheel angle sensor with two tasks (n_{18}–n_{19}), suspension controller with five tasks (n_{20}–n_{24}), and body work with seven tasks (n_{25}–n_{31}). The real-life automotive function is generated based on the above parameter values.

The minimum and maximum energy consumption values of the function are 825.94 and 12 440.24 Wμs by using (6) and (7), respectively. The heterogeneous EFT (HEFT) generated energy consumption value is 5382.36 Wμs, which is considered the maximum meaningful energy consumption value because HEFT is a well-studied and commonly used DAG-based scheduling algorithm for reducing the response time without using DVFS [32]. Therefore, we change the energy constraints from 825.94 Wμs to 5382.36 Wμs with an increment of 1139.1 Wμs and observe the results. We fix the response-time constraint at 5000 μs. The results are shown in Figs. 8–10.

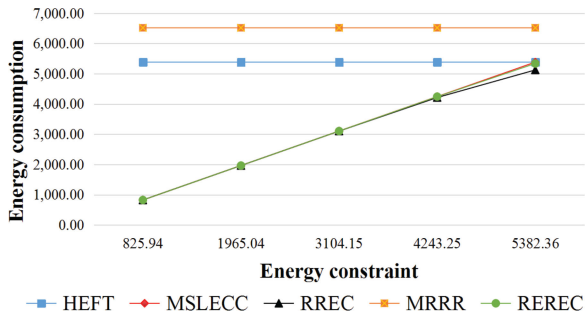


Fig. 8. Energy consumption values (unit: Wμs) of a real-life automotive function for different energy constraints (unit: Wμs).

Fig. 8 shows the actual energy consumption values of a real-life automotive function for different energy constraints.

1) The curves of MSLECC, RREC, and REREC in Fig. 8 are basically coincident, such that they can hardly be distinguished. This phenomenon reflects that the actual energy consumption values generated by MSLECC, RREC, and REREC are always smaller than and close to the corresponding energy constraints.
2) Fig. 8 shows that none of the energy consumption values generated by MRRR assures the energy constraints, but is far greater than the corresponding energy constraints. The reason is that MRRR does not involve the energy consideration but focuses on reliability enhancement under response-time constraint.
3) Based on the curves in Fig. 8 and the aforementioned analysis, we confirm that MSLECC, RREC, and REREC are designed for energy constraints, whereas HEFT and MRRR are not.

Fig. 9 shows the actual response-time values of a real-life automotive function for different energy constraints.

1) Considering that the response-time constraint is fixed at 5000 μs, the HEFT, MSLECC, RREC, MRRR, and

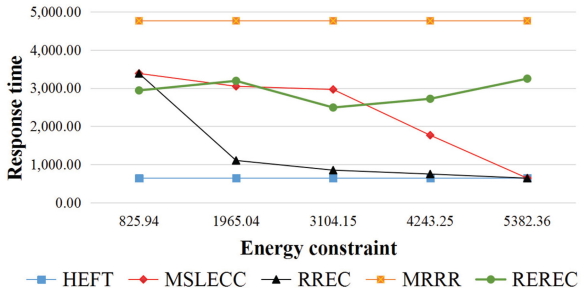


Fig. 9. Response-time values (unit: μs) of a real-life automotive function for different energy constraints (unit: Wμs).

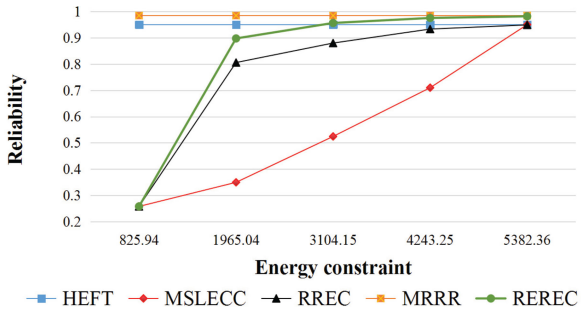


Fig. 10. Reliability values of a real-life automotive function for different energy constraints (unit: Wμs).

REREC algorithms can always assure the response-time constraint. HEFT obtains the minimum response-time value followed by RREC in all the cases. Particularly, HEFT-generated and MRRR-generated response-time values are fixed at 640 μs and 4764 μs, respectively. The reason is that HEFT and MRRR do not involve energy consideration, such that the results do not change for different energy constraints.

2) The curves show that MSLECC and RREC obtain approximate equal response values in the two ends because two extreme energy constraints limit the play of RREC. RREC generates smaller response-time values than MSLECC because MSLECC adopts pessimistic energy preallocation, whereas RREC uses a relative average energy preallocation to improve the limitation of MSLECC. These results and analyses indicate that RREC effectively addresses the problem of reducing the response time under its energy constraint.
3) We know that HEFT is merely for reducing response time, RREC is for reducing response time under energy constraint, MRRR is for enhancing reliability under response-time constraint, and REREC is for enhancing reliability under energy and response-time constraints. Considering that response time, energy, and reliability are mutually exclusive, we find that the more constraints there are, the worse the performance is.

Fig. 10 shows the reliability values of a real-life automotive function for different energy constraints.

1) The reliability values obtained by HEFT are fixed at 0.9501. MRRR obtains the highest reliability value 0.9847 in all the cases, whereas REREC obtains relatively low reliability values when the energy constraint is small. The reason is that MRRR is for enhancing reliability under only one constraint, whereas REREC is for enhancing reliability under two constraints. The limitation of MRRR is that it cannot assure the energy constraints, as shown in Fig. 8.
2) Although HEFT and MSLECC also obtain relatively high-reliability values in all the cases, they have individual limitations. HEFT cannot always assure the energy constraints (Fig. 8), whereas MSLECC generates longer response time in most cases (Fig. 9).
3) REREC obtains higher reliability values as much as 10.22% than RREC when the energy constraint is 1965.04 Wμs because REREC implements reliability-enhancement technology based on RREC. If we let the reliability goal be 0.94 (i.e., the reliability must be larger than or equal to 0.94), then RREC has only one case (the energy constraint is 5382.36 Wμs) that assures the reliability goal of 0.94, whereas REREC has three cases (except for the case where the energy constraint is 825.94 Wμs and 1965.04 Wμs). In summary, REREC effectively enhances the reliability to assure the high-reliability goal compared with RREC.

C. Synthetic Automotive Functions

Given the increasing complexity of ACPS, future automotive functions are likely to reach 100 tasks [6]. To further validate the effectiveness of the proposed approach, we use additional synthetic functions with the same actual parameter values as the real-life function to observe the results. Randomly generated functions can be obtained by using the task graph generator [33], which can be used to develop task graphs that are needed for research works in areas of task scheduling. The following parameters are set to generate the functions [33].

1) The communication to computation ratio (CCR) set is $\{0.1, 0.5, 1.0, 2.0, 5.0\}$.
2) The ECUs' heterogeneous factor set is $\{0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$; the larger the factor is, the higher the heterogeneity is.
3) The shape factor is a random number, whose range of values is $[0.35, \sqrt{|N|/3}]$.

In this experiment, we set CCR to 1, heterogeneous factor to 1.0, and shape factor to $\sqrt{|N|/3}$. We change the number of tasks of function from 50 to 100 with 10 increments. The response-time constraint is fixed at 5000 μs for functions because they have more tasks than the real-life function in Fig. 7. Table VI shows the HEFT-generated energy values and corresponding energy constraints of synthetic automotive functions for different numbers of tasks.

1) Fig. 11 shows the energy consumption values of synthetic automotive functions for different numbers of tasks. Similar to Fig. 8, the curves of MSLECC, RREC, and REREC are also basically coincident, such that they are still

TABLE VI
HEFT-GENERATED ENERGY VALUES (UNIT: WμS) AND CORRESPONDING ENERGY CONSTRAINTS OF SYNTHETIC AUTOMOTIVE FUNCTIONS FOR DIFFERENT NUMBERS OF TASKS

$\|N\|$	50	60	70	80	90	100
$E_{\text{HEFT}}(G)$	1275.23	1836.95	1679.99	2368.69	2094.62	2711.66
$E_{\text{cons}}(G)$	637.62	918.47	839.99	1184.34	1047.31	1355.83

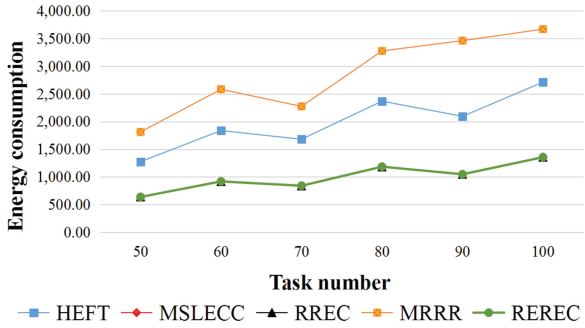


Fig. 11. Energy consumption values (unit: Wμs) of synthetic automotive functions for different numbers of tasks.

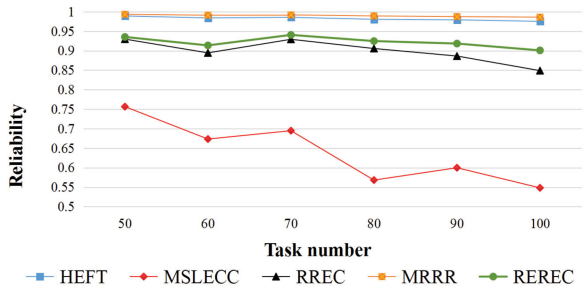


Fig. 12. Response-time values (unit: μs) of synthetic automotive functions for different numbers of tasks.

hardly distinguished. This phenomenon further confirms that the actual energy consumption values generated by MSLECC, RREC, and REREC are always smaller than and close to the corresponding energy constraints. On the contrary, HEFT and MRRR-generated energy consumption values are still far greater than the corresponding energy constraint values.

2) Fig. 12 shows the response-time values of synthetic automotive functions for different numbers of tasks. We can see that RREC can still generate smaller response-time values than MSLECC in all the cases. Although HEFT and MRRR have shorter response-time values than RREC in most cases, none of them can assure the energy constraints combing with the data results in Fig. 11 and Table VI.
3) Fig. 13 shows the reliability values of synthetic automotive functions for different numbers of tasks. REREC can generate higher reliability values than RREC in all the cases. These results further indicate that REREC's reliability-enhancement technique is effective toward automotive functional safety goal assurance. Although HEFT and MRRR have higher reliability values than

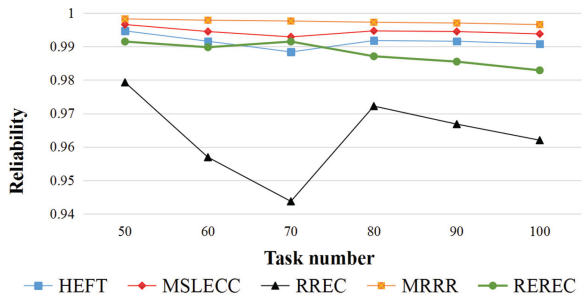


Fig. 13. Reliability values of synthetic automotive functions for different numbers of tasks.

TABLE VII
PARAMETERS OF PLATFORM

CPU	OS	Memory	DVFS Tool
Allwinner A20 (Dual-Core ARM Cortex-A7)	Debian	1 GB	CPUfreq
Frequency (unit: GHz)	1.01, 0.960, 0.912, 0.864, 0.816, 0.768, 0.744, 0.720, 0.696, 0.672, 0.648, 0.600, 0.528, 0.480, 0.408, 0.384, 0.360, 0.336		

REREC, they either do not assure the energy consumption constraints.

D. Hardware Platform Experiment

We setup a hardware platform with six Allwinner A20 by a small experiment. The detail parameters of the hardware platform are shown in Table VII.

The core's voltage is 5 V and the platform changes power by changing the current. We test that the idle power P_{idle} (including P_{ind} and P_s, i.e., $P_{\text{idle}} = P_{\text{ind}} + P_s$) depends on frequency change. Under the condition that there is no task to execute, we test that the idle powers are different when CPU operates at different frequencies. For example, the actual idle power is 1.35 W (0.27 A $\times$ 5 V) when frequency is 336 MHz, 1.45 W (0.29 A $\times$ 5 V) when frequency is 866 MHz, and 1.5 W (0.3 A $\times$ 5 V) when frequency is 1.01 GHz.

Besides P_d, P_{ind}, and P_s, other power P_{other} exists due to the using of OS and other components: 1) $P_d = C_{\text{ef}} f^m$, where $C_{\text{ef}} = 1$ and $m = 3$ using data fitting; 2) P_s and P_{other} are considered the basic static power together because these two powers are hard to separate [i.e., $P_s \leftarrow (P_s + P_{\text{other}})$]; the result shows that P_s is 1.35 W when the core is idle in the minimum available frequency of $f_{\min} = 0.336$ GHz; 3) $P_{\text{idle}} = 1.5$ W when the core is idle in the maximum frequency of $f_{\max} = 1.01$ GHz. Therefore, P_{ind} is 0.15 W($P_{\text{idle}} = 1.5 - 1.35 = 0.15$ W); 4) f_{ee} is $\sqrt[3]{\frac{0.15}{(3-1)\times 1}} = 0.422$ GHz calculated by (3). As 0.422 GHz does not exist in the discrete frequency set, f_{ee} is set to 0.480 GHz because it is closest to 0.422 GHz.

By recording the actual number of assembly instructions and corresponding execution times, we are able to obtain the instructions per second (IPS) as 0.67 G in the hardware platform. The real-life benchmark in Fig. 7 is created by setting 0.67×10^{-4} G instructions for each task. Therefore, the WCET for each task with 1.01 GHz is 100 μs.

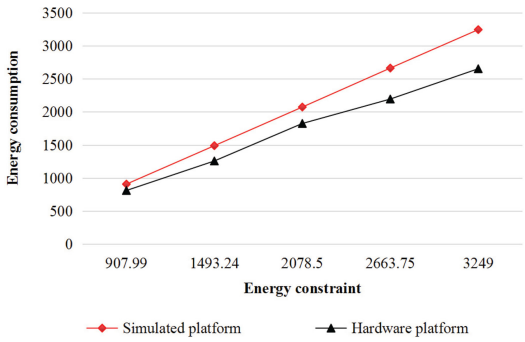


Fig. 14. Energy consumption values (unit: Wμs) for different energy constraint values (unit: Wμs) on simulated and hardware platforms.

We execute the function in the hardware platform according to the allocated core, start time, and end time of each task generated by RREC in the simulated platform. In other words, the task allocations under the hardware platform is exactly the same as under the simulated platform. Our objective is to verify whether the energy consumption of the function is within its energy constraint in the hardware platform.

The minimum and HEFT-generated energy values of the function are 907.99 and 3249 Wμs, respectively. The energy constraint of the function is changed from 907.99 to 3249 Wμs with the increment of 585.25 Wμs. The results are shown in Fig. 14, where we can see that the energy consumption values generated in the hardware platform are always less than those generated in the simulated platform. That is, the energy consumption of the function is within its energy constraint in the hardware platform. With the increment of energy constraint, the differences are increasingly large. The reason for the above phenomenon is that P_{ind} decreases as frequency decreases in the hardware platform, whereas P_{ind} is fixed in the simulated platform. For example in the hardware platform, we have the following records: when the frequency is 1.01 GHz, P_{ind} is 0.15 W, and when the frequency is 0.408 GHz, P_{ind} is deceased to 0.035 W. However, P_{ind} is fixed at 0.15 W without considering the frequency change in the simulated platform. For the reason as to why P_{ind} decreases as frequency decreases in the hardware platform, we analyze that similar to P_{s}, P_{ind} also includes partial powers of OS and other components, which are frequency dependent and is hard to separate.

E. Summary of Experiments

We made the following summarizations based on the experiments:

1) MSLECC, RREC, and REREC can always assure energy constraints, and HEFT and MRRR do not assure energy constraints.
2) RREC is more effective than MSLECC in reducing the response time under energy constraint.
3) MRRR can enhance the reliability but does not assure the response-time constraints.
4) REREC obtains higher reliability values than RREC and implements reliability enhancement based on RREC.

Through the above summary, we confirm that the proposed two-stage solution (i.e., RREC and REREC) show feasibility toward functional safety goal assurance in energy-aware ACPS.

VII. Conclusion

We solved the problem of reliability enhancement of a distributed automotive function under energy and response-time constraints in ACPS by proposing the RREC and REREC algorithms to form a two-stage solution. RREC improves the pessimistic energy preallocation of the existing method, thereby significantly reducing response time. REREC implements reliability enhancement based on the first stage and demonstrates the effectiveness of reliability enhancement by using real-life and synthetic automotive functions. According to the ISO 26262 standard, RREC can only assure the reliability goal corresponding to exposure E3, whereas REREC can assure the high-reliability goal corresponding to exposure E2 by reliability-enhancement technique based on RREC. RREC and REREC show feasibility toward functional safety goal assurance in energy-aware ACPS.

Appendix

Theorem 1: Each task $n_{s(y)}$ in distributed function G can always find an ECU to be allocated to assure

$$E(G) = \sum_{x=1}^{y-1} E(n_{s(x)}) + E(n_{s(y)}) + \sum_{z=y+1}^{|N|} E_{\text{avg}}(n_{s(z)}) \leqslant E_{\text{cons}}(G).$$

Proof: Considering that average energy preallocation $E_{\text{avg}}(n_{s(z)})$ in (11) is considered the minimum energy of the current task $n_{s(z)}$, if we can prove that the sum of the average energy preallocations of all tasks is less than or equal to the energy constraint of the function, then the theorem is proven.

We let the sum of the average energy preallocations of all tasks be

$$E_{\text{avg}}(G) = \sum_{z=1}^{|N|} E_{\text{avg}}(n_{s(z)}). \tag{25}$$

Then, substituting (11) into (25) yields

$$\begin{aligned}
&\sum_{z=1}^{|N|} \left(E_{\min}(n_{s(z)}) + \frac{\Delta E_{\text{ae}}(G)}{|N|} \right) \\
&= \sum_{z=1}^{|N|} \left(E_{\min}(n_{s(z)}) + \frac{E_{\text{cons}}(G) - E_{\min}(G)}{|N|} \right) \\
&= \sum_{z=1}^{|N|} E_{\min}(n_{s(z)}) + \sum_{z=1}^{|N|} \left(\frac{E_{\text{cons}}(G) - E_{\min}(G)}{|N|} \right) \\
&= E_{\min}(G) + (E_{\text{cons}}(G) - E_{\min}(G)) = E_{\text{cons}}(G).
\end{aligned}$$

Given that $E_{\text{avg}}(G)$ is equal to $E_{\text{cons}}(G)$ under the average energy preallocation for tasks, we can find allocated ECUs to assure $E_{\text{cons}}(G)$. Thus, Theorem 1 is proven. ■

ACKNOWLEDGMENT

The authors would like to express their gratitude to the Associate Editor and four anonymous reviewers for their constructive comments which have helped to improve the quality of the paper.

REFERENCES

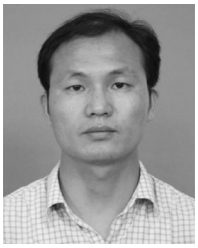
[1] "Cyber-physical systems in green transportation," 2017. [Online]. Available: http://www.ieee-ies.org/images/files/tii/ss/2017/CfP-Cyber-Physical_Sys tems_in_Green_Transportation_2017-6-27.pdf
[2] H. Dong, S. Gao, and B. Ning, "Cooperative control synthesis and stability analysis of multiple trains under moving signaling systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2730–2738, Oct. 2016.
[3] M. Di Natale and A. L. Sangiovanni-Vincentelli, "Moving from federated to integrated architectures in automotive: The role of standards, methods and tools," *Proc. IEEE*, vol. 98, no. 4, pp. 603–620, Apr. 2010.
[4] G. Xie, G. Zeng, Z. Li, R. Li, and K. Li, "Adaptive dynamic scheduling on multifunctional mixed-criticality automotive cyber-physical systems," *IEEE Trans. Veh. Technol.*, vol. 66, pp. 6676–6692, Aug. 2017.
[5] E. Rolf, "Formal performance analysis in automotive systems design - a rocky ride to new grounds," in *Proc. 23rd Int. Conf. Computer Aided Verification*. Springer, 2011. [Online]. Available: http://101.110.118.63/formalverification.cs.utah.edu/cav2011/content/presentations/CAV2011V3.pdf
[6] 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, pp. 4378–4391, May 2018.
[7] NSF, "Cyber-physical systems (cps), program solicitation nsf 17-529," pp. 1–21, 2017. [Online]. Available: https://www.nsf.gov/pubs/2017/nsf17529/nsf17529.htm
[8] X. Hu, H. Wang, and X. Tang, "Cyber-physical control for energy-saving vehicle following with connectivity," *IEEE Trans. Ind. Electronics*, vol. 64, no. 1, pp. 8578–8587, Nov. 2017.
[9] K. Vatanparvar and M. A. Al Faruque, "Battery lifetime-aware automotive climate control for electric vehicles," in *Prof. 52nd ACM/EDAC/IEEE Design Automat. Conf. (DAC)*. IEEE, 2015, pp. 1–6.
[10] P. Mercati, V. Hanumaiah, J. Kulkarni, S. Bloch, and T. Rosing, "Blast: Battery lifetime-constrained adaptation with selected target in mobile devices," in *Proc. 12th EAI Int. Conf. Mobile Ubiquitous Systems: Computing, Netw. Services*. ICST, 2015, pp. 80–89.
[11] ISO, "Iso 26262–road vehicles-functional safety," *Int. Org. Standardization ISO 26262*, 2011.
[12] Y. Wan, H. Huang, D. Das, and M. Pecht, "Thermal reliability prediction and analysis for high-density electronic systems based on the markov process," *Microelectronics Rel.*, vol. 56, no. 5, pp. 182–188, May 2016.
[13] S. Gao, H. Dong, and B. Ning, "Neural adaptive dynamic surface control for uncertain strict-feedback nonlinear systems with nonlinear output and virtual feedback errors," *Nonlinear Dynamics*, vol. 90, no. 4, pp. 2851–2867, Dec. 2017.
[14] A. Girault and H. Kalla, "A novel bicriteria scheduling heuristics providing a guaranteed global system failure rate," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 4, pp. 241–254, Oct.-Dec. 2009.
[15] G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications on heterogeneous systems," *IEEE Trans. Ind. Informat.*, vol. PP, pp. 1–1, Mar. 2017.
[16] G. Xie, Y. Chen, R. Li, and K. Li, "Hardware cost design optimization for functional safety-critical parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Ind. Inform.*, vol. PP, no. 99, pp. 1–14, Nov. 2017.
[17] G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li, and K. Li, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," *IEEE Trans. Ind. Informatics*, vol. 13, no. 4, pp. 1629–1640, Aug. 2017.
[18] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 316–328, Aug. 2010.
[19] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2867–2876, Nov. 2014.
[20] J. Zhou, T. Wei, M. Chen, J. Yan, X. S. Hu, and Y. Ma, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time mpsoc systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1269–1282, Aug. 2016.
[21] X. Xiao, G. Xie, R. Li, and K. Li, "Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems," in *Proc. 14th IEEE Int. Symp. Parallel Distrib. Process. Appl*. IEEE Computer Society, 2016, pp. 1471–1476.
[22] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proc. 2012 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput*. IEEE Computer Society, 2012, pp. 781–786.
[23] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment," *Journal of Grid Computing*, vol. 14, no. 1, pp. 55–74, Mar. 2016.
[24] S. M. Shatz and J. P. Wang, "Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems," *IEEE Trans. Rel.*, vol. 38, no. 1, pp. 16–27, Apr. 1989.
[25] 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, Nov. 2017.
[26] D. Zhu, R. Melhem, and D. Mosse, "The effects of energy management on reliability in real-time embedded systems," in *Proc. Ieee/acm Int. Conf. Comput.-Aided Design*, 2004, pp. 35–40.
[27] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," *Inf. Sci.*, vol. 319, no. C, pp. 113–131, Oct. 2015.
[28] X. Xiao, G. Xie, C. Xu, C. Fan, R. Li, and K. Li, "Maximizing reliability of energy constrained parallel applications on heterogeneous distributed systems," *J. Comput. Sci.*, Mar. 2017.
[29] G. Bernat, A. Colin, and S. M. Petters, "Wcet analysis of probabilistic hard real-time systems," in *Proc. 23rd IEEE Real-Time Syst. Symp.* IEEE, 2002, pp. 279–288.
[30] G. Xie *et al.*, "Wcrt analysis of can messages in gateway-integrated in-vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 9623–9637, Dec. 2017.
[31] J. D. Ullman, "Np-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, Jun. 1975.
[32] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[33] "https://sourceforge.net/projects/taskgraphgen/," 2015.

Guoqi Xie (M'15) received a Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2014.

He was a Postdoctoral Research Fellow with Hunan University from 2015 to 2017, and with Nagoya University, Japan, from 2014 to 2015. Since 2017, he has been an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University. His current research interests include design automation of automotive cyber-physical systems, embedded and cyber-physical systems, and parallel and distributed systems.

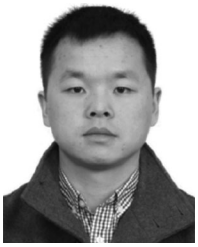
He received the best paper award at ISPA 2016. He is currently serving on the editorial boards of *Journal of Systems Architecture*, *Journal of Circuits*, and *Systems and Computers*.

Hao Peng is currently working toward the M.S. degree with Hunan University, Changsha, China. His current research interests include automotive cyber-physical systems.



Zhetao Li (M'17) received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2010. He is a Professor with Xiangtan University, Xiangtan, China. His current research interests include cyber-physical systems and Internet of Things (IOT).

Jinlin Song received the M.S. degree in computer science and engineering from Hunan University, Changsha, China, in 2017. His current research interests include automotive cyber-physical systems.



Yong Xie (M'18) received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2013.

He is currently an Associate Professor with the Xiamen University of Technology, Xiamen, China. He is currently a Postdoctoral Researcher with the National Laboratory for Parallel and Distributed Processing, National University of Defensive Technology, Changsha, China. His major interests include in-vehicle networks, embedded real-time systems, and cyber-physical systems.

Renfa Li (M'05–SM'10) received the Ph.D. degree in electronic engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2002.

He is a Professor of computer science and electronic engineering with Hunan University, Changsha, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, Hunan University. His current research interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of Things.

Keqin Li (M'90–SM'96–F'15) received the Ph.D. degree in computer science from the University of Houston, Houston, TX USA, in 1990.

He is a Distinguished Professor of computer science with the State University of New York, New York, NY, USA. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of Things, and cyber-physical systems.

He is currently or has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL and DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He is an IEEE Fellow.