Energy-Efficient Functional Safety Design Methodology Using ASIL Decomposition for Automotive Cyber-Physical Systems

Guoqi Xie, *Senior Member, IEEE*, Hao Peng, Jing Huang, Renfa Li, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Automotive cyber-physical systems (ACPS) are typical cyber-physical systems because of the joint interaction between the cyber part and physical part. Functional safety requirement (including response time and reliability requirements) for an ACPS function must be assured for safe driving. Auto industry is cost-sensitive, power-sensitive, and environment-friendly. Energy consumption affects the development efficiency of the ACPS and the living environment of people. This paper solves the problem of optimizing the energy consumption for an ACPS function while assuring its functional safety requirement (i.e., energy-efficient functional safety for ACPS). However, implementing minimum response time, maximum reliability, and minimum energy consumption is a conflicting problem. Consequently, solving the problem is a challenge. In this paper, we propose a three-stage design process toward energy-efficient functional safety for ACPS. The topic problem is divided into three sub-problems, namely, response time requirement verification (first stage), functional safety requirement verification (second stage), and functional safety-critical energy consumption optimization (third stage). The proposed energy-efficient functional safety design methodology is implemented by using automotive safety integrity level decomposition, which is defined in the ACPS functional safety standard ISO 26262. Experiments with real-life and synthetic ACPS functions reveal the advantages of the proposed design methodology toward energy-efficient functional safety for ACPS compared with state-of-the-art algorithms.**

***Index Terms*—Automotive safety integrity level (ASIL) decomposition, automotive cyber-physical systems (ACPS), energy-efficient, functional safety, ISO 26262.**

Manuscript received August 29, 2018; revised December 21, 2018 and April 8, 2019; accepted May 6, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61702172 and Grant 61672217, in part by the National Technical Committee of Auto Standardization Research Foundation of China under Grant BZ201908, in part by the Natural Science Foundation of Hunan Province under Grant 2018JJ3076, in part by the Open Research Project of the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, China, under Grant PAL-N201803, and in part by the Fundamental Research Funds for the Central Universities. Associate Editor: F. Belli. *(Corresponding author: Guoqi Xie.)*

G. Xie, H. Peng, J. Huang, and R. Li are with the Department of Computer Engineering, College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China, and also with the Key Laboratory for Embedded and Cyber-Physical Systems of Hunan Province, Changsha 410082, China (e-mail: xgqman@hnu.edu.cn; hao_peng@hnu.edu.cn; jingh@hnu.edu.cn; lirenfa@hnu.edu.cn).

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/TR.2019.2915818

NOMENCLATURE

Acronyms and Abbreviations

ACPS	Automotive cyber-physical systems.
AFT	Actual finish time.
ASIL	Automotive safety integrity level.
AST	Actual start time.
BSH	Bicriteria scheduling heuristic.
CAN	Controller area network.
CPS	Cyber-physical systems.
DAG	Directed acyclic graph.
DEWTS	DVFS-enabled energy-efficient workflow task scheduling.
DUECM	Downward and upward energy consumption minimization.
DVFS	Dynamic voltage and frequency scaling.
ECU	Electronic control unit.
E/E	Electrical and electronic.
EES	Energy-efficient scheduling.
EFSRG	Energy-efficient fault-tolerant scheduling with a reliability goal.
EST	Earliest start time.
EPM	Energy-aware processor merging.
EFT	Earliest finish time.
FSEO	Functional safety-critical energy consumption optimization.
FSRV	Functional safety requirement verification.
HARA	Hazard analysis and risk assessment.
HEFT	Heterogeneous earliest finish time.
IAL	Insertable ASIL level.
HRRM	Heuristic replication for redundancy minimization.
LFT	Latest finish time.
MRCRG	Minimizing resource cost with reliability goal.
OCT	Optimistic cost table.
QEPM	Quick energy-aware processor merging.
QM	Quality management.

QoS	Quality of service.
STAMP	Systems-theoretic accident modeling and processes.
WCET	Worst case execution time.
WCRT	Worst case response time.

Notations

$\|U\|$	Size of set U.
pred(n_i)	Immediate predecessor task set of task n_i.
succ(n_i)	Immediate successor task set of task n_i.
$w_{i,k,h}$	WCET of task n_i in ECU u_k and ASIL L_h.
$c_{i,j}$	WCRT of message $m_{i,j}$ from tasks n_i to n_j.
$\text{rank}_\text{u}(n_i)$	Upward rank value of task n_i.
λ_k	Failure rate of ECU u_k.
$\lambda_{k,v}$	Failure rate of ECU u_k with frequency $f_{k,v}$.
EFT $(n_i, scheme_g)$	Earliest finish time of task n_i with scheme $scheme_g$.
AFT(n_i)	Actual finish time of task n_i.
EFT $\left(n_i^\beta, u_k, L_h\right)$	Earliest finish time of replica n_i^β in ECU u_k and ASIL L_h.
EST(n_i, u_k)	Earliest start time of task n_i in ECU u_k.
LFT(n_i, u_k)	Latest finish time of task n_i in ECU u_k.
$R\left(n_i, u_k, L_h\right)$	Reliability of task n_i in ECU u_k and ASIL L_h.
$R\left(n_i, u_k, L_h, f_{k,v}\right)$	Reliability of task n_i in ECU u_k, ASIL L_h with frequency $f_{k,v}$.
$R(n_i)$	Reliability of task n_i.
$R_\text{req}(n_i)$	Reliability requirement of task n_i.
$R(G)$	Reliability of function G.
$R_\text{req}(G)$	Reliability requirement of function G.
RT(G)	Response time of function G.
$D(G)$	Real-time requirement (i.e., deadline) of function G.
$E\left(n_i, u_k, L_h, f_{k,v}\right)$	Energy consumption of task n_i in ECU u_k and ASIL L_h with frequency $f_{k,v}$.
$E(G)$	Energy consumption of function G.
IAL(n_i, u_k)	Insertable ASIL level of task n_i in ECU u_k.

I. Introduction

A. Motivation

CYBER-PHYSICAL systems (CPSs) are engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components [1]–[3]. Automotive CPSs (ACPS) are typical CPSs because of the joint interaction between the cyber part and physical part [4]–[7]. In an ACPS, electronic control units (ECUs) are mounted on communication buses and exchange data on buses by a message passing manner. ACPS functions (i.e., applications) are usually distributed and parallel because they exhibit evident data dependencies and precedence constraints among tasks (e.g., brake-by-wire function) [4], [7], [8]. Considering the above characteristics, such an ACPS function has been represented by a directed acyclic graph (DAG) [4], [8].

Different from general networked systems, safety is the consistent theme for safe driving and life protection in the ACPS [5], [6], [9]. To further strengthen the safety development of ACPS, the road vehicles-functional safety standard (i.e., ISO 26262) for ACPS was issued in 2011 [10]. In ISO 26262, functional safety is formally defined as the absence of unreasonable risks owing to hazards caused by malfunctioning behavior of electrical and electronic systems [10]. ISO 26262 has become the preferential direction of the ACPS development. The response time and reliability are two functional safety properties in ISO 26262. Response time requirement (also named deadline constraint or real-time requirement [11]) and reliability requirement (also named reliability goal) must be assured simultaneously for safe driving. However, simultaneously implementing minimum response time and maximum reliability is a conflicting problem for a DAG function [12]–[16], such that these two requirements may not be assured simultaneously. Ensuring the functional safety requirement for an ACPS function is a challenge.

Auto industry is cost-sensitive, power-sensitive, and environment-friendly because almost every automaker is looking forward to achieving good quality recognition in the mass market competition. Energy consumption affects the development efficiency of the ACPS and the living environment of people [17]. At the same time, to increase the driving mileage of battery electrical vehicles, it is necessary to carry out comprehensive and effective energy management for ACPS. Therefore, energy consumption must be optimized as much as possible to improve the development efficiency of ACPS after ensuring its functional safety.

There are some methods to reduce energy consumption in ACPS from different levels, including dynamic power management [18], dynamic voltage and frequency scaling (DVFS) [19], [20], dynamic programming [21], and model predictive control [22]. In this paper, we use DVFS to optimize energy consumption from a system-level perspective. DVFS simultaneously scales the voltage and frequency of a processor (i.e., ECU) to manage the energy consumption [23]–[27] and is employed to reduce the energy consumption for the ACPS [19], [20], [28]. However, implementing minimum energy consumption and minimum response time is a conflicting problem. The reason is that when the voltage of an ECU is scaled down, the execution time of the task in this ECU will increase. In addition, implementing minimum energy consumption and maximum reliability is also a conflicting problem. The reason is that when the voltage of an ECU is scaled down, the transient failures of the ECU will rise. As a result, the reliability of the function is weakened [23]–[27].

This paper aims to solve the problem of optimizing the energy consumption for an ACPS function while assuring its functional safety requirement (i.e., energy-efficient functional safety for ACPS). Considering the mutual conflict among response time minimization, reliability maximization, and energy consumption minimization, solving this problem is a challenge, especially in a complex ACPS.

B. Overview of This Paper

A life cycle of a safety-critical system development process usually involves the analysis (concept), design, implementation,

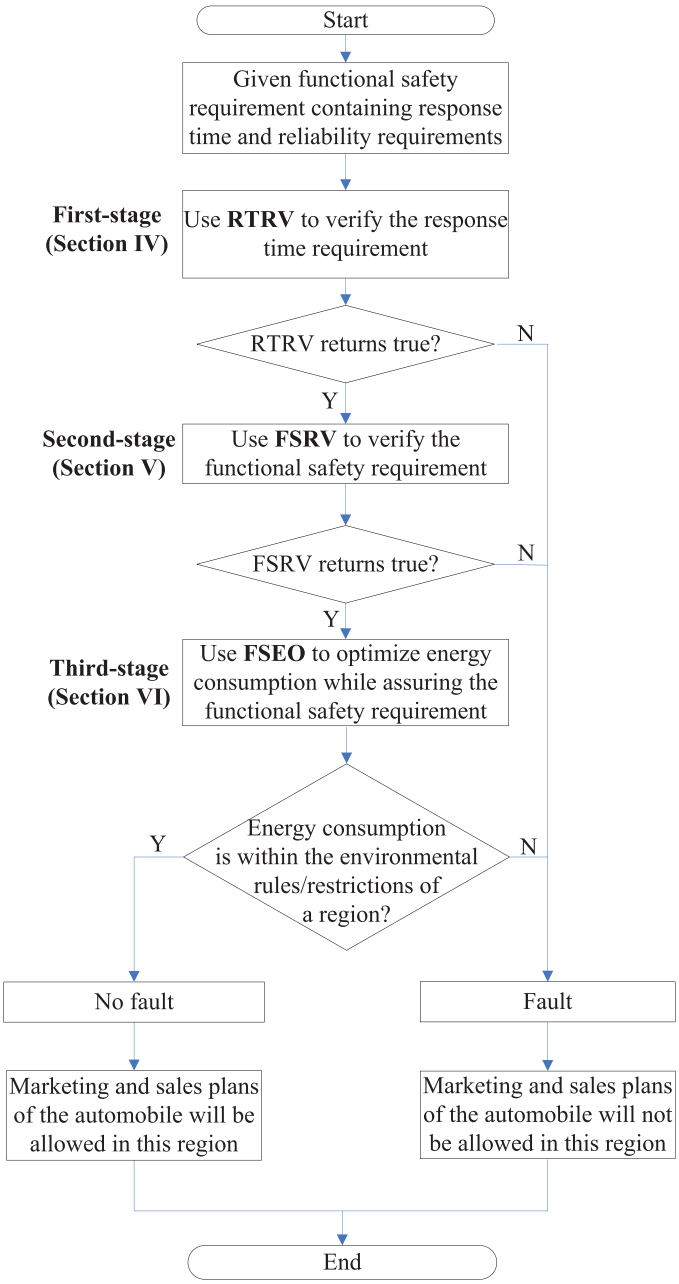


Fig. 1. Flowchart of the energy-efficient functional safety design methodology for an ACPS.

testing, and runtime phases [13], [29]. We focus on the design phase in this paper. Considering that directly implementing the energy-efficient functional safety for an ACPS is complex, we propose a three-stage design process toward energy-efficient functional safety for the ACPS. The problem is divided into three sub-problems, namely, response time requirement verification (RTRV), functional safety requirement verification (FSRV), and functional safety-critical energy consumption optimization (FSEO). Each stage solves the corresponding sub-problem.

Fig. 1 shows the flow chart of the energy-efficient functional safety design methodology for the ACPS in this paper.

1) In the first-stage, we solve the first sub-problem about RTRV for an ACPS function in Section IV, in which we present the RTRV algorithm to obtain the ACPS function's response time. If the response time of the function is less than or equal to its response time requirement, then the verification is passed; otherwise, it is not passed. The above process is shown in the first stage of Fig. 1.
2) If the RTRV is passed in Section IV, then the second-stage solves the second sub-problem about FSRV for the ACPS function in Section V, in which we present the FSRV algorithm to obtain the ACPS function's reliability while assuring its response time requirement. If the reliability is larger than or equal to the reliability requirement, then the verification is passed; otherwise, it is not passed. The above process is shown in the second stage of Fig. 1.
3) If the FSRV is passed in Section V, then the third stage solves the third sub-problem (i.e., real objective problem) about FSEO for the ACPS function in Section VI, in which we present the FSEO algorithm to optimize the energy consumption while assuring its response time and reliability requirements. If the energy consumption is within the environmental rules/restrictions in a designated region, then the energy-efficient functional safety design methodology is valid, and the marketing sales plans of the automobile will be allowed in this region; otherwise, the design methodology is invalid, and the marketing sales plans of the automobile will not be allowed in this region. The above process is shown in the third stage of Fig. 1.

C. Contributions of This Paper

1) The contribution of the RTRV algorithm proposed in the first stage is that it implements the objective of verifying the response time of the ACPS function by using an automotive safety integrity level (ASIL) decomposition according to the guide of ISO 26262.
2) The contribution of the FSRV algorithm proposed in the second stage is that it implements the objective of enhancing the reliability of the ACPS function while assuring its response time requirement by using the ASIL decomposition further according to the guide of ISO 26262.
3) The contribution of the FSEO algorithm proposed in the third stage is that it implements the objective of optimizing energy consumption of the ACPS function while assuring its response time and reliability requirements. That is, for given functional safety requirement containing response time and reliability requirements, FSEO employs a DVFS-enabled energy consumption optimization technique to implement energy-efficient functional safety design methodology for ACPS.
4) Experiments with real-life and synthetic functions reveal the advantages of the proposed design methodology toward energy-efficient functional safety for ACPS compared with state-of-the-art algorithms.

II. Related Work

In [30], Sheikh and Ahmad summarized the evaluation of performance, energy, and temperature for diverse task assignment techniques. In [31], Assayad *et al.* explored the tradeoff among

reliability, power consumption, and execution time. The correlation among the performance, reliability, and energy consumption of a non-DAG function has been studied in cloud systems [32], [33]. We mainly review the response time, reliability, and energy consumption for a DAG function. The related works reviewed in this section are organized as follows.

1) *Response Time and Reliability:* It was pointed out in ISO 26262 that random hardware failures for a hardware element are unpredictable. However, such failures could obey a probability distribution during the life cycle [10], and the Poisson distribution has been widely adopted for transient failures in many studies [12]–[14], [34]–[36]. The problem of simultaneously optimizing response time and reliability is considered a typical bicriteria optima or Pareto optima problem [12]–[14], [36]–[39]. The bicriteria scheduling heuristic (BSH) algorithm proposed in [12] generates an approximate Pareto curve of nondominated solutions, by which the designers can find the points that simultaneously assure the reliability and response time requirements. However, BSH has high time complexity and does not meet the time control of the development life cycle of ACPS. The union fast functional safety verification (UFFSV) method proposed in [39] implements low time complexity by minimizing response time while assuring the reliability requirement or maximizing reliability, assuring the response time requirement. However, UFFSV does not adopt fault-tolerance, such that it may not be able to assure high reliability requirement. The MaxRe algorithm proposed in [34] aims to optimize the resource cost for a DAG function while assuring its reliability requirement by fault-tolerance. MaxRe is implemented by transferring the reliability requirement of a DAG function to that of each task through setting all tasks to equal reliability requirement in an average manner. The RR algorithm proposed in [35] extends the MaxRe algorithm by considering the previous assignments of tasks when assigning the current task. The minimizing resource cost with reliability goal (MRCRG) algorithm proposed in [13] solves the same problem as MaxRe and RR without using fault-tolerance. When using MRCRG, each unassigned task is preassigned to the ECU with maximum reliability. The heuristic replication for redundancy minimization algorithm proposed in [14] employs fault-tolerance and takes the upper bound on reliability as the preassigned value for each unassigned task to optimize redundancy. However, all these algorithms only consider the reliability requirement and disregard the response time requirement.
2) *Response Time and Energy Consumption:* One of the objectives of DVFS-enabled design techniques is to optimize the energy consumption for a DAG function while assuring its response time requirement. The energy-efficient scheduling (EES) algorithm proposed in [40] addresses the problem in heterogeneous distributed systems. EES reclaims the slack time for each task in its fixed assigned processor [40]. Considering that EES is merely an upward approach, the downward and upward energy consumption minimization algorithm proposed in [41] solves the same problem as that solved in [40] by using downward and upward approaches together. Considering that using the DVFS technique alone is insufficient, a combined EES approach of non-DVFS and global DVFS was proposed in [42]. Turning OFF partial processors to optimize energy consumption is a feasible method, particularly in cloud-based distributed systems [43], [44]. The DVFS-enabled energy-efficient workflow task scheduling (DEWTS) algorithm proposed in [43] aims to optimize energy consumption and realize EES-based slack time reclamation by turning OFF inefficient processors. Given that DEWTS is not energy-aware, then energy-aware processor merging (EPM) and quick EPM algorithms were proposed in [44]. In addition, optimizing the response time for a DAG function while assuring its energy constraint was also studied in [45]. However, all these algorithms do not consider the reliability requirement.
3) *Response Time, Reliability, and Energy Consumption:* Energy consumption is related to reliability in DVFS-enabled processors. A model based on the relationship between energy consumption and reliability was established in [23]. The problem of enhancing the reliability for a DAG function while assuring its response time requirement and energy consumption constraint was solved in a uniprocessor [25] and ACPS [7], respectively. On the basis of [25], the problem of optimizing the energy consumption for a DAG function while assuring the response time and reliability requirements in a uniprocessor was solved in [26]. However, heterogeneous ECUs have been employed in ACPS. The energy-efficient fault-tolerant scheduling with a reliability goal (EFSRG) algorithm proposed in [27] addresses the problem of optimizing the energy consumption for a DAG function while assuring its reliability requirement. However, EFSRG merely assures the reliability requirement and does not consider the response time requirement. As pointed out in the Introduction section, this paper aims to address the problem of optimizing the energy consumption for an ACPS function while assuring its functional safety requirement, including response time and reliability requirements.

III. Preliminaries

This section introduces the preliminaries of the ACPS architecture and ASIL decomposition in ISO 26262.

A. ACPS Architecture

A typical ACPS architecture is shown in Fig. 2. A controller area network (CAN) bus connects to several ECUs, and ECUs connect to several sensors and actuators in the CAN bus [42]. The CAN bus is an event-triggered, nondestructive, strictly deterministic medium arbitration bus and is ideally suited for ACPS. The end-to-end computation and communication process for an ACPS function is from sensors to actuators [4].

A task executed completely in one ECU sends messages to all its successor tasks, which may be located in different ECUs. For example, task n_1 is executed in ECU u_1. It sends a message

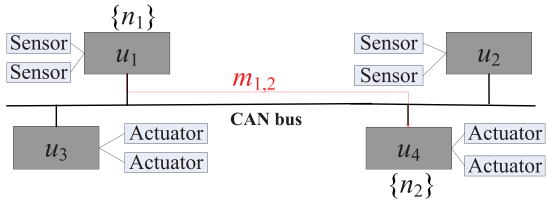


Fig. 2. Typical ACPS architecture.

TABLE I
SEVERITY, EXPOSURE, AND CONTROLLABILITY CLASSIFICATIONS PROVIDED BY ISO 26262 [10], [46]

Severity		Exposure		Controllability	
S0	No injuries	E0	Incredibly unlikely	C0	Controllable in general
S1	Light to moderate injuries	E1	Very low probability	C1	Simply controllable
S2	Severe to life-threatening injuries	E2	Low probability	C2	Normally controllable
S3	Life-threatening to fatal injuries	E3	Medium probability	C3	Difficult to control or uncontrollable
		E4	High probability		

$m_{1,2}$ to its successor task n_2 located in u_4 (see Fig. 2). The ECUs are heterogeneous, and the ECU set is represented by $U = \{u_1, u_2, \ldots, u_{|U|}\}$. Notice that we employ $|U|$ to represent the size of set U. This paper employs $|X|$ to represent the size of set X.

B. ASIL Determination

ISO 26262 defines three mutually orthotropic functional safety attributes, namely, severity, exposure, and controllability, to construct four criticality levels denoted by ASIL (i.e., A, B, C, and D) [10]. The severity, exposure, and controllability classifications provided by ISO 26262 are shown in Table I.

1) Severity refers to the extent of harm to an individual in a specific situation, and it has four levels—S0, S1, S2, and S3 [10]. Severity is related to systemic failures of hazardous events. Severity cannot be changed for a given ACPS function because it has been determined by hazard analysis and risk assessment (HARA) during the analysis phase.
2) Exposure refers to the relative expected frequency of the operational conditions, in which hazardous events may occur and cause hazards and injuries. It has five levels—E0, E1, E2, E3, and E4. Exposure is related to random hardware failures [10]. Exposure can be understood by the inverse expression of reliability (i.e., exposure $= 1 -$ reliability).
3) Controllability refers to avoiding a specified harm or damage through the timely reactions of the involved persons, and it has four levels—C0, C1, C2, and C3 [10]. Controllability is obtained based on driver's current state, such that it can be dynamically changed during the runtime phase. In this paper, we set the controllability to a fixed level of C3 (i.e., uncontrollable) for the ACPS function during the design phase. This arrangement is feasible because ACPS design is usually conservative for safety needs.

TABLE II
ASIL DETERMINATION BASED ON SEVERITY, EXPOSURE, AND CONTROLLABILITY PROVIDED BY ISO 26262 [10], [46]

Severity	Exposure	Controllability		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

The ASIL determinations are shown in Table II. ASIL A and ASIL D represent the lowest and highest ASILs, respectively. ASIL is a manifestation of the risk degree. The higher the ASIL, the greater the risk and the effort required to reduce the risk. QM means quality management and is not related with the safety design for an ACPS function. Each ASIL is the combination of specific severity, exposure, and controllability values. For example, ASIL D is only the combination of S3, E4, and C3 according to Table II. The ASIL is determined by HARA and assigned to corresponding safety requirements during the analysis phase. Moreover, the ASIL has been known before making the detailed design, and some techniques can be used for hazard analysis, such as hazard and operability analysis [47], failure modes and effects analysis [48], fault tree analysis [49], event tree analysis [50], goal structuring notation [51], and systems theoretic process analysis [52]. Considering that this paper focuses on the design phase, we assume that the ASIL of an ACPS function has been known by a certain HARA technique during the analysis phase.

Considering that severity and controllability are fixed at S3 and C3, respectively, in this paper, the exposures of E1, E2, E3, and E4 correspond to ASIL A, ASIL B, ASIL C, and ASIL D. Therefore, we let $\{L_{\mathrm{A}}, L_{\mathrm{B}}, L_{\mathrm{C}}, L_{\mathrm{D}}\}$ represent the ASIL level corresponding to exposure set of $\{\mathrm{E1}, \mathrm{E2}, \mathrm{E3}, \mathrm{E4}\}$ in this paper.

C. ASIL Decomposition Principle

Although ASIL is determined during the analysis phase by HARA, it is possible to decompose the safety requirement of an ACPS function to redundant safety requirements. Those redundant safety requirements are then assigned to tasks of the ACPS function if and only if these tasks are independent. ASIL decomposition is an important functional safety assurance method in ISO 26262 and it means apportioning redundant safety requirements to sufficiently independent elements with the objective of reducing the ASIL of these elements [9], [10].

1) *Redundancy Paradigm:* Two types of redundancies exist: homogenous and heterogeneous.
   a) Homogeneous redundancy is formed by the duplication of elements (e.g., duplicated hardware devices or software tasks) in ASIL decomposition.

b) Heterogeneous redundancy is formed by the combination of hardware devices and software tasks in ASIL decomposition.

2) *Primary-Backup Replication Paradigm:* Two types of primary-backup replications exist [14], [27], [34], [35], [53]: passive and active.
   a) Passive replication means that whenever an ECU fails, the task executed in that ECU is rescheduled to proceed in a backup ECU.
   b) Active replication means that each task has multiple replicas simultaneously executed in multiple different ECUs. As long as at least one replica is executed successfully, the task will be successfully completed. Active replication does not allow two replicas (including primary replica and backups) of the same task to be allocated to the same ECU [14], [27], [34], [35].

In this paper, we use the homogeneous redundancy and the active replication to implement ASIL decomposition because this combination can implement the independence among the software tasks.

D. ASIL Decomposition Guide

For an ACPS function with ASIL D, the highest ASIL for the ECUs to execute the tasks of this function must be ASIL D. For an ACPS function with ASIL C, the highest ASILs for the ECUs to execute the tasks of this function can be ASIL C or ASIL D. The ACPS function in this paper is a safety-critical function with ASIL D. As explained in Section III-B, ASIL is a manifestation of the risk degree. The higher the ASIL, the greater the risk and the greater the effort required to reduce the risk. To support the execution of the function with ASIL D, the highest ASIL for the ECUs to execute the tasks of this function must be ASIL D; otherwise, the function with ASIL D cannot be initialized. If the ASIL of the function being studied is ASIL C or ASIL B, there are similar solutions by referencing to the algorithms proposed in the paper.

The guide of the ASIL decomposition scheme is provided in ISO 26262, Part 9, Section V and is shown in Fig. 3 [10]. ASIL decomposition is defined in ISO 26262 and complies with the ACPS functional safety design specification for efficient development among different automakers. Using ASIL decomposition is necessary because it can enhance the reliability of the task by adding redundancy and executing the task in lower ASILs with shorter WCETs for lower cost. For example, task n_1 is set to ASIL D. n_1 can be decomposed to two redundant replicas n_1^1 and n_1^2 based on Fig. 3(d). n_1^1 can be set to ASIL B and n_1^2 can be set to ASIL B. After this decomposition, n_1 changes from one replica to two replicas, thereby enhancing the reliability.

Then, each task's replicas can be executed in all four ASILs by ASIL decomposition. All possible decomposition schemes of ASIL D are obtained as follows.

1) There are three ASIL decomposition schemes for ASIL D, including ASIL C + ASIL A, ASIL B+ ASIL B, and ASIL D + QM, as shown in Fig. 3(d). Notice that QM is not related with the safety design for an ACPS function as pointed out earlier. Therefore, the feasible decomposition schemes for ASIL D are ASIL C + ASIL A and ASIL B + ASIL B.
2) There are two ASIL decomposition schemes for ASIL C, including ASIL B + ASIL A and ASIL C + QM, as shown in Fig. 3(c). The feasible decomposition scheme for ASIL C is only ASIL B + ASIL A. Therefore, the decomposition scheme ASIL C + ASIL A generated by ASIL D can be further decomposed to ASIL B + 2 × ASIL A, as shown by the arrow from Fig. 4(a)–(d).
3) There are two ASIL decomposition schemes for ASIL B, including ASIL A + ASIL A and ASIL A + QM, as shown in Fig. 3(b). The feasible decomposition scheme for ASIL B is only ASIL A + ASIL A. Therefore, the decomposition scheme ASIL B + ASIL B generated by ASIL D can also be further decomposed to ASIL B + 2 × ASIL A, as shown by the arrow from Fig. 4(b)–(d).

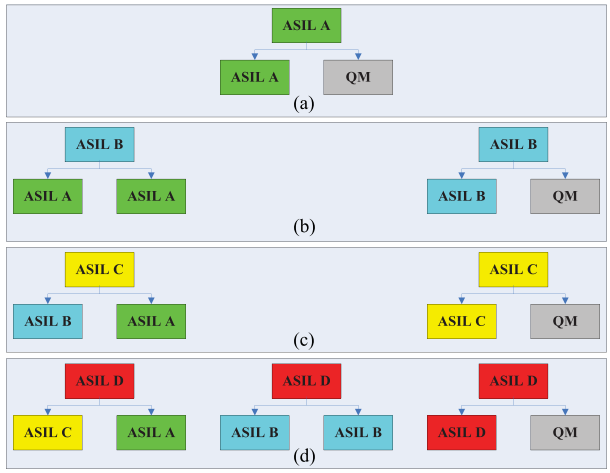


Fig. 3. Guide of ASIL decomposition scheme in ISO 26262 [10].

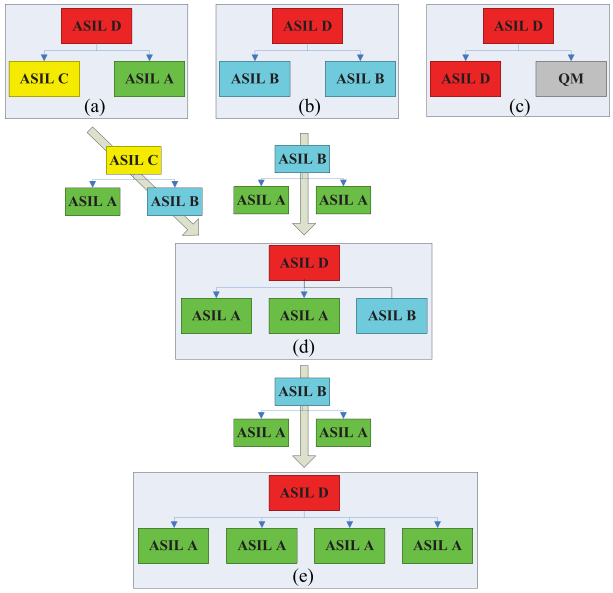


Fig. 4. Five ASIL decomposition schemes for ASIL D.

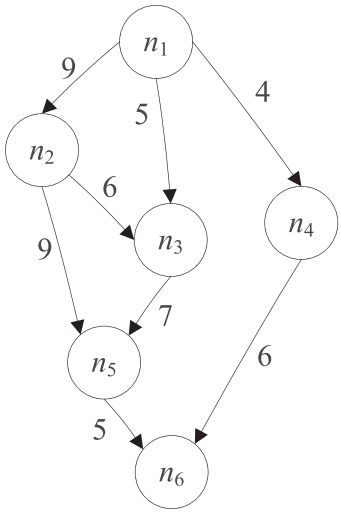


Fig. 5. Motivational ACPS function in this paper.

4) Considering that the feasible decomposition scheme for ASIL B is only ASIL A + ASIL A [shown in Fig. 3(b)], ASIL B + 2 × ASIL A can be further decomposed to 4 × ASIL A, as shown by the arrow from Fig. 4(d)–(e).

Finally, there are five ASIL decomposition schemes for ASIL D, as shown in Fig. 4. These five schemes are as follows: ASIL C + ASIL A (scheme$_1$), ASIL B + ASIL B (scheme$_2$), ASIL D (scheme$_3$), ASIL B + 2 × ASIL A (scheme$_4$), and 4 × ASIL A (scheme$_5$).

IV. Response Time Requirement Verification

This section solves the first sub-problem of RTRV for an ACPS function in the first stage.

A. ACPS Function Model

A DAG $G = (N, W, M, C)$ is employed to represent an ACPS function with an end-to-end computation and communication process [4], [7], [13]. The detailed parameters are explained as follows.

1) N is a node set of the graph. Each node $n_i \in N$ denotes a task. Fig. 5 shows a motivational ACPS function consisting of six tasks. pred(n_i) denotes the set of immediate predecessor tasks of n_i; considering the example in Fig. 5, we have pred$(n_5) = \{n_2, n_3\}$. succ(n_i) denotes the set of immediate successor tasks of n_i; for example, we have succ$(n_5) = \{n_6\}$. The task without any predecessor task is called n_{entry} (i.e., n_1 in Fig. 5), and the task without any successor task is called n_{exit} (i.e., n_6 in Fig. 5). An ACPS function may be released by receiving physical data from multiple sensors and be finished through sending the execution command to multiple actuators. For example, the brake-by-wire function shown in Fig. 6 has one entry task and two exit tasks [7], [54]. Therefore, multiple n_{entry} or multiple n_{exit} tasks may exist. If an ACPS function has multiple n_{entry} or multiple n_{exit} tasks, then a dummy entry or exit task with zero-weight dependency should be added to the graph to balance it, especially when implementing the proposed algorithms in a programming language.

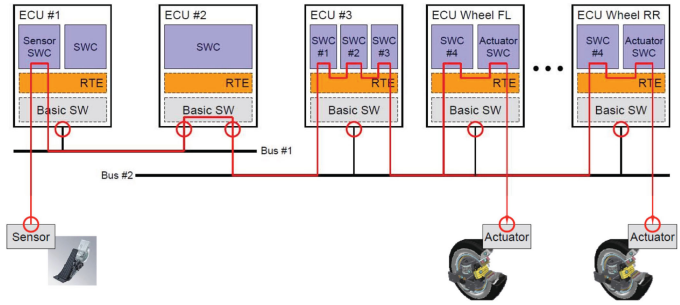


Fig. 6. Brake-by-wire function [7], [54].

TABLE III
WCETs of Tasks in Different ECUs and ASILs

	n_1				n_2				n_3				n_4				n_5				n_6			
	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4	u_1	u_2	u_3	u_4
L_{A}	4	7	5	8	10	9	4	7	15	8	13	16	14	11	10	6	14	12	16	8	4	6	4	8
L_{B}	6	9	7	10	12	11	6	9	17	10	14	18	16	13	12	14	16	14	18	10	5	8	5	10
L_{C}	8	11	9	12	14	13	8	11	19	12	16	20	21	23	18	20	18	16	20	12	6	10	7	12
L_{D}	10	13	11	14	16	15	10	13	23	14	18	24	27	28	25	22	20	18	22	14	7	12	9	14

2) W represents an $|N| \times |U| \times 4$ cube, where $w_{i,k,h}$ denotes the worst case execution time (WCET) of n_i in ECU u_k and ASIL L_h. Considering partial ECUs can only execute specific tasks, we let $w_{i,k,h} = +\infty$, which means that assigning n_i to u_k is meaningless. Considering that we focus on the design phase, we assume that all the WCETs of the tasks are known and determined through the WCET analysis during the analysis phase [55].
A high-ASIL task has larger WCET than (or equal WCET to) a low-ASIL task in the same ECU, because a task's critical level represents the degree of conservatism for the WCET estimation. The higher the critical level of the task is, the more conservative the WCET estimation is and the larger the WCET is [56]. ACPS are typical mixed-criticality systems through using ASIL to represent the criticality, and thus we should illustrate that the WCET of any functional task depends on its ASIL level [57], [58]. Each task has individual WCET values in individual ECUs even in the same ASIL because ECUs are heterogeneous. Table III lists the WCETs of each task in four ECUs $\{u_1, u_2, u_3, u_4\}$ and four ASILs $\{L_{\text{A}}, L_{\text{B}}, L_{\text{C}}, L_{\text{D}}\}$ for the motivational ACPS function. The weight 7 of n_1, u_2, and L_{A} in Table III denotes the WCET of n_1 in u_2 and L_{A}, and is expressed by $w_{1,2,\text{A}} = 7$.
3) The communication from a task in an ECU to another task in another ECU must be implemented by passing a message in the bus. We let M be a set of communication edge in the motivational ACPS function. The edge $m_{i,j} \in M$ denotes the communication message from n_i to n_j. Accordingly, $c_{i,j} \in C$ denotes the worst case response time (WCRT) of message $m_{i,j}$. For example, the weight 9 of the edge from n_1 to n_2 in Fig. 5 denotes the WCRT of message $m_{1,2}$, which is denoted by $c_{1,2} = 9$.

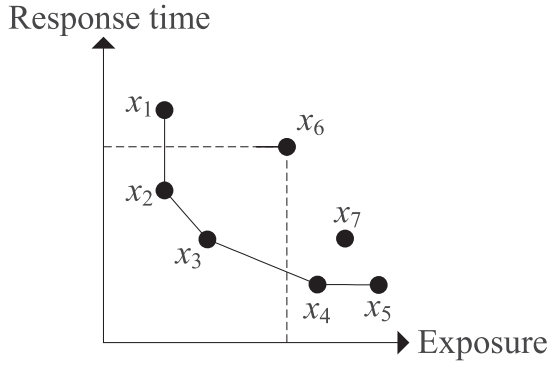


Fig. 7. Bicriteria between response time minimization and exposure minimization [12], [39].

Similar to WCET, we also assume that all the WCRTs of the messages are known and determined through the WCRT analysis during the analysis phase [59]. Related WCRT analysis methods can be refer to [59]–[61], such as the end-to-end WCRT analysis for CAN messages [59], the exact WCRT analysis for single-core automotive gateways [60], and approximate WCRT analysis for multicore automotive gateways [61]. Considering the shared memory mechanism for the same ECU, $c_{i,j}$ should be 0 if n_i and n_j are assigned to the same ECU.

The scheduling strategies can be either preemptive or nonpreemptive in ACPS [4], [62]. We employ nonpreemptive task scheduling in ECUs to keep consistent with the nonpreemptive CAN message scheduling in the network.

The motivational ACPS function will be employed to explain the proposed three-stage design process in the paper. For simplicity, all units of all parameters are ignored in the example. All the data employed in the motivational ACPS function is intended to give readers a better understanding of the methodology proposed in this paper and does not reflect the real deployment of the ACPS.

B. Bicriteria Between Response Time Minimization and Exposure Minimization

Fig. 7 shows the bicriteria between response time minimization and exposure minimization (i.e., reliability maximization) for an end-to-end ACPS function [12], [39].

In Fig. 7, each point x^1–x^7 represents a solution of a bicriteria minimization problem. The points x^1, x^2, x^3, x^4, and x^5 are Pareto optima; the points x^1 and x^5 are weak optima, whereas the points x^2, x^3, and x^4 are strong optima. The set of all the Pareto optima is called a Pareto curve.

The topic of this paper is to solve the problem of optimizing the energy consumption for an ACPS function while assuring its functional safety requirement. An intuitive method is to assure both the response time and reliability requirement simultaneously and then optimize the energy consumption. To simultaneously assure the response time and reliability requirement, a possible strategy is to find strong optimal points by using the BSH algorithm proposed in [12]; an iterative approach is then adopted to find the optimal point among the strong optimal points. However, considering that the time complexity of BSH reach as high as $O(|N| \times 2^{|U|})$, simultaneously assuring the response time and reliability requirements is time consuming and may exceed the time control of the development life cycle of the ACPS. On the basis of the aforementioned explanations, the following sections propose a three-stage design process, which first assures response time requirement, then assures reliability requirement, and finally optimizes energy consumption.

C. RTRV Method

In this section, we aim to find the ECU and ASIL assignments for all the tasks to reduce the ACPS function's response time

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

where $\mathrm{RT}(G)$ and $\mathrm{AFT}(n_{\mathrm{exit}})$ represent the response time of the ACPS function and actual finish time (AFT) of the exit task, respectively. If

$$\mathrm{RT}(G) \leqslant D(G)$$

where $D(G)$ is the response time requirement (i.e., deadline) of the ACPS function, then the RTRV is passed; otherwise, it is not passed.

Scheduling tasks with quality of service requirement for optimality in multiple ECUs is an NP-hard optimization problem [42], [63]. Considering that the auto industry is cost-sensitive, shortening the ACPS function's development life cycle to obtain high development efficiency is crucial. Hence, we adopt list scheduling to find the ECU and ASIL assignments.

1) Task Prioritization: Task prioritization is the first step in list scheduling. Upward rank value [64] and optimistic cost table [65] are available task prioritization standards. No task prioritization standard is better than other standards due to uncertain WCET and WCRT values and the DAG's topologies. However, prioritizing tasks by the descending order of upward rank values (i.e., $\mathrm{rank_u}$) using (1) is considered a well-defined task prioritization standard because it has been well employed for the energy-efficient design [40]–[42], [44], [66] and reliability-aware design [13], [14], [27], [34], [35]. We employ the highest level ASIL D to determine $\mathrm{rank_u}$ because the ASIL of the ACPS function is ASIL D

$$\mathrm{rank_u}(n_i) = \overline{w_{i,\mathrm{D}}} + \max_{n_j \in \mathrm{succ}(n_i)} \{c_{i,j} + \mathrm{rank_u}(n_j)\}. \quad (1)$$

$\overline{w_{i,\mathrm{D}}}$ denotes the average WCET of task n_i in ASIL D

$$\overline{w_{i,\mathrm{D}}} = \left(\sum_{k=1}^{|U|} w_{i,k,\mathrm{D}} \right) \Big/ |U|. \quad (2)$$

The upward rank values of all the tasks of the motivational ACPS function are shown in Table IV. The task priorities in this section are $\{n_1, n_2, n_3, n_5, n_4, n_6\}$.

2) Response Time Calculation: There are two types of fault-tolerant schedules [14], [27], [67], [68]: strict and general. For a strict schedule, n_i should wait for the finish of all the replicas of n_i's predecessors before starting its execution. Therefore, a strict schedule is a compile-time schedule. For a general schedule,

TABLE IV
UPWARD RANK VALUES OF THE TASKS IN THE MOTIVATIONAL ACPS FUNCTION

Task	n_1	n_2	n_3	n_4	n_5	n_6
$rank_{\mathrm{u}}(n_i)$	117	96	67	38	42	10

n_i's execution can be started as soon as one replica of n_i's each predecessor has been successfully finished. Therefore, a general schedule is a run-time schedule. We adopt the strict schedule in this paper because this paper focuses on the design phase, which belongs to the static mapping.

Considering that ISO 26262 employs ASIL decomposition to implement fault-tolerance, we let num_i ($\mathrm{num}_i \leqslant |U|$) be the number of replicas for n_i after using ASIL decomposition. Thus, the replica set of n_i is $\{n_i^1, n_i^2, \ldots, n_i^{\mathrm{num}_i}\}$, where n_i^1 is the primary and the others are the backups.

Similar to the heterogeneous earliest finish time algorithm [64], we assign n_i to the ECU with the minimum earliest finish time (EFT) because it can assure the local optimal of each task. For an ACPS function with ASIL D, five ASIL decomposition schemes exist for each task, and each scheme has one–four replicas, then we choose the ASIL decomposition scheme with the minimum EFT. That is, the assigned ASIL decomposition scheme $\mathrm{scheme}_{sc(i)}$ for n_i is determined by

$$\mathrm{AFT}(n_i) = \mathrm{EFT}(n_i, \mathrm{scheme}_{sc(i)}) = \min_{g\in[1,5]} \mathrm{EFT}(n_i, \mathrm{scheme}_g). \tag{3}$$

The remaining work involves calculating the EFT of each scheme $\mathrm{EFT}(n_i, \mathrm{scheme}_g)$. scheme_g may have different replicas with different ASILs and need to prioritize these replicas. There are two typical solutions for prioritizing replicas: ascending order of ASILs and descending order of ASILs. Through a large number of experiments, we found a very interesting phenomenon that these two solutions always generate the same response time for the same ACPS function. The root reason is that the ASIL decomposition employs active replication in this paper. Active replication does not allow two replicas of the same task to be assigned to the same ECU as pointed out in Section III-C. Therefore, the two aforementioned solutions are both feasible. In this paper, we merely employ the first solution to explain the example.

We let $\mathrm{EFT}\left(n_i^\beta, u_k, L_h\right)$ represent the EFT of replica n_i^β in ECU u_k and ASIL L_h. Given that the strict schedule is employed, EFT is calculated by

$$\begin{cases} \mathrm{EFT}(n_1^\beta, u_k, L_h) = \mathrm{EST}(n_1^\beta, u_k, L_h) + w_{1,k,h} \\ \mathrm{EFT}(n_i^\beta, u_k, L_h) = \max \\ \quad \left\{ \begin{array}{l} \mathrm{avail}[k], \\ \max\limits_{n_x\in\mathrm{pred}(n_i),\alpha\in[1,num_x]} \left\{\mathrm{AFT}(n_x^\alpha) + c_{x,i}^{'}\right\} \end{array} \right\} + w_{i,k,h}. \end{cases} \tag{4}$$

$\mathrm{EST}(n_i^\beta, u_k, L_h)$ represents earliest start time (EST) of n_i^β in ECU u_k and ASIL L_h. $\mathrm{avail}[k]$ is the earliest available time of ECU u_k. $\mathrm{AFT}(n_x^\alpha)$ is the AFT of replica n_x^α. $c_{x,i}^{'}$ represents the WCRT of message $m_{x,i}$ between n_x^α and n_i^β. If n_x^α and n_i^β are assigned to the same ECU, then $c_{x,i}^{'}$ is 0; otherwise, $c_{x,i}^{'}$ is equal to $c_{x,i}$. $w_{i,k,h}$ denotes the WCET of task n_i in ECU u_k and ASIL L_h.

Considering the example of scheme_1 [see Fig. 3(a)], ASIL D is decomposed to one ASIL A and one ASIL C. Thus, n_1 executed in ASIL A has the following options:

$$\begin{cases} \mathrm{EFT}(n_1, u_1, L_\mathrm{A}) = 4 \\ \mathrm{EFT}(n_1, u_2, L_\mathrm{A}) = 7 \\ \mathrm{EFT}(n_1, u_3, L_\mathrm{A}) = 5 \\ \mathrm{EFT}(n_1, u_4, L_\mathrm{A}) = 8 \end{cases}. \tag{5}$$

Considering that $\mathrm{EFT}(n_1, u_1, L_\mathrm{A}) = 4$ is the minimum value in (5), n_1 is assigned to u_1 and L_A. Then, n_1 with ASIL C may be executed in the following ECUs:

$$\begin{cases} \mathrm{EFT}(n_1, u_2, L_\mathrm{C}) = 11 \\ \mathrm{EFT}(n_1, u_3, L_\mathrm{C}) = 9 \\ \mathrm{EFT}(n_1, u_4, L_\mathrm{C}) = 12 \end{cases}. \tag{6}$$

Considering that $\mathrm{EFT}(n_1, u_3, L_\mathrm{C}) = 9$ is the minimum value in (6), n_1 is further assigned to u_3 and L_C. u_1 is not involved in (6) because n_1 has been assigned to it in (5). Assigning n_1 to u_1 again is not allowed using active replication. Based on (5) and (6), we have

$$\mathrm{EFT}(n_1, \mathrm{scheme}_1) = 9. \tag{7}$$

$\mathrm{EFT}(n_i, \mathrm{scheme}_g)$ is calculated by

$$\mathrm{EFT}(n_i, \mathrm{scheme}_g) = \min_{u_k\in U} \mathrm{EFT}(n_i, u_k, L_{\mathrm{last}(\mathrm{scheme}_g)}) \tag{8}$$

where $L_{\mathrm{last}(\mathrm{scheme}_g)}$ represents the last ASIL that is used to calculate EFT. The last ASIL is L_C for scheme_1.

The remaining ASIL decomposition schemes employ the same pattern as scheme_1. Finally, all the EFTs of n_1 in the five schemes are as follows:

$$\begin{cases} \mathrm{EFT}(n_1, \mathrm{scheme}_1) = 9 \\ \mathrm{EFT}(n_1, \mathrm{scheme}_2) = 7 \\ \mathrm{EFT}(n_1, \mathrm{scheme}_3) = 10 \\ \mathrm{EFT}(n_1, \mathrm{scheme}_4) = 9 \\ \mathrm{EFT}(n_1, \mathrm{scheme}_5) = 8 \end{cases}. \tag{9}$$

We choose $\mathrm{EFT}(n_1, \mathrm{scheme}_2) = 7$ because it has the minimum EFT. In other words, the n_1-chosen ASIL decomposition scheme is scheme_2, and its two replicas are assigned to u_1 and u_3, as shown in Fig. 8.

The remaining tasks employ the same pattern as n_1. Given that all the nonexit tasks are the predecessors of the exit task, the response time of the ACPS function is the AFT of the exit task and is calculated by

$$\mathrm{RT}(G) = \mathrm{AFT}(n_{\mathrm{exit}}) = \min_{u_k\in U} \mathrm{EFT}(n_{\mathrm{exit}}, u_k, L_{\mathrm{last}(\mathrm{scheme}_g)}). \tag{10}$$

The task reassignments of the motivational ACPS function are shown in Fig. 8. The tasks with specific colors are executed

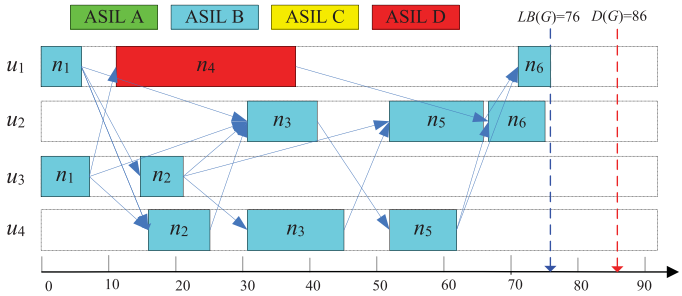


Fig. 8. RTRV-generated task reassignment of the motivational ACPS function.

in specific ASILs. In this paper, ASIL A, ASIL B, ASIL C, and ASIL D are specified by the colors of green, blue, yellow, and red, respectively. The response time of the ACPS function is 76.

3) Response Time Requirement Verification: The RTRV is operated as follows.

A known response time requirement $D(G)$ of the ACPS function is provided on the basis of the actual physical time requirement after HARA. If

$$\mathrm{RT}(G) \leqslant D(G) \tag{11}$$

then, the RTRV is passed; otherwise, it is not passed. For the motivational ACPS function, we let the response time requirement be $D(G) = 86$, which can pass the RTRV, because

$$76 \leqslant 86$$

according to (11).

D. RTRV Algorithm

An algorithm called RTRV is presented to verify the response time requirement of the ACPS function based on the aforementioned analysis. The RTRV algorithm is described in Algorithm 1.

RTRV chooses the ASIL decomposition scheme, which has the minimum EFT for each task among five schemes according to the task prioritization standard of upward rank value. After implementing the ASIL decomposition scheme for the exit task n_{exit}, the response time of n_{exit} is then obtained. Considering the response time of n_{exit} represents the response time $\mathrm{RT}(G)$ of the ACPS function G, $\mathrm{RT}(G)$ can be compared with the given response time requirement $D(G)$ to verify whether the response time requirement can be assured. The main details are as follows.

1) In Line 1, RTRV prioritizes tasks by the descending order of $\mathrm{rank_u}(n_i, u_k)$ values using (1) (from the entry to the exit tasks).
2) In Lines 2–13, RTRV chooses the ASIL decomposition scheme with the minimum EFT among five schemes for each task. Particularly, each task's EFT in individual ECU and ASIL is obtained in Line 7, and their response time values in individual ASIL decomposition schemes are obtained in Line 10.
3) In Line 14, RTRV calculates the ACPS function's response time $\mathrm{RT}(G)$.

Algorithm 1: The RTRV Algorithm.

Input: $U = \{u_1, u_2, \ldots, u_{|U|}\}$, $\{L_\mathrm{A}, L_\mathrm{B}, L_\mathrm{C}, L_\mathrm{D}\}$, G, $D(G)$
Output: $RT(G)$ and related values

1: Prioritize the tasks in a list ranku_task_list by the descending order of $\mathrm{rank_u}(n_i)$ values using Eq. (1) (from the entry to the exit tasks);
2: **while** (there are tasks in ranku_task_list) **do**
3: $n_i \leftarrow$ ranku_task_list.out();
4: **for** ($g \leftarrow 1; g \leq 5; g$++) **do**
5: **for** (each ASIL $L_h \in \mathrm{scheme}_g$) **do**
6: **for** (each ECU $u_k \in U$) **do**
7: Calculate $\mathrm{EFT}(n_i, u_k, L_h)$ using (4);
8: **end for**
9: **end for**
10: Calculate $\mathrm{EFT}(n_i, \mathrm{scheme}_g)$ using (8);
11: **end for**
12: Choose the ASIL decomposition scheme with the minimum EFT among five schemes of n_i;
13: **end while**
14: Calculate the $\mathrm{RT}(G)$ using (10);
15: **if** ($\mathrm{RT}(G) \leqslant D(G)$) **then**
16: Pass the response time requirement verification;
17: **return**;
18: **else**
19: Do not pass the response time requirement verification;
20: **return**;
21: **end if**

4) In Lines 15–21, RTRV compares the ACPS function's response time $\mathrm{RT}(G)$ with its response time requirement $D(G)$ to verify whether the response time requirement can be assured.

The time complexity of the RTRV algorithm is $O(|N|^2 \times |U|)$. The reason is that traversing all tasks needs $O(|N|)$ time (Lines 2–13), and calculating $\mathrm{EFT}(n_i, u_k, L_h)$ needs $O(|N| \times |U|)$ time (Lines 4–11).

V. Functional Safety Requirement Verification

If the RTRV is passed, then we begin to solve the second sub-problem of FSRV for the ACPS function in the second. We present the reliability requirement based on ISO 26262 and then introduce the reliability requirement verification method.

A. Reliability Requirement and Reliability Model

Table V shows the duration/probability of exposure provided by ISO 26262 [10], [46]. Although ISO 26262 is a functional safety standard based on random hardware failures, the reliability requirement is not defined. Fortunately, the reliability requirement for a specific exposure level can be deduced [46]. As can be seen in Table V, E2 has a fixed reliability requirement value of 0.99, while the reliability requirements of other exposures (i.e., E1, E3, and E4) are only given the reliability

TABLE V
EXPOSURE CLASSIFICATIONS REGARDING THE PROBABILITY OF EXPOSURE PROVIDED BY ISO 26262, AND RELIABILITY REQUIREMENT DEDUCED BY THE PROBABILITY OF EXPOSURE [10], [46]

Exposure	Probability of exposure	Reliability requirement	Specified reliability requirement in this paper
E1 Very low probability	Not specified	At least exceeds 0.99	0.9999
E2 Low probability	$< 1\%$	0.99	0.99
E3 Medium probability	[1%, 10%]	>0.9	0.95
E4 High probability	$> 10\%$	<=0.9	0.9

requirement ranges according to the deductive process in [46]. In other words, the reliability requirements of E1, E3, and E4 are flexible. The advantage of such flexibility is that the automakers can specify reliability values according to individual production demands, as long as the reliability values fall within the exposure scopes deduced by the exposures defined in ISO 26262. For example, this paper assumes that the reliability requirement values are 0.9999, 0.95, and 0.9 corresponding to exposures E1, E3, and E4, respectively.

There are two types of random hardware failures [12]–[14], [27], [34], [35]: transient and permanent. A transient failure is a failure that occurs and subsequently disappears. Soft errors (such as bit flips) are examples of such failures [69], [70]. Permanent failures are failures that occur and remain (open, short of a connection). This paper considers the transient failure of ECUs with the Poisson distribution [7], [12]–[14], [34], [35], [71]. We employ λ_k to denote the failure rate of ECU u_k. The reliability value of n_i executed in u_k and ASIL L_h is denoted by

$$R\left(n_i, u_k, L_h\right) = e^{-\lambda_k \times w_{i,k,h}} \tag{12}$$

and the failure probability for n_i without using replication is

$$1 - R\left(n_i, u_k, L_h\right) = 1 - e^{-\lambda_k w_{i,k,h}}. \tag{13}$$

Given that active replication is employed, the reliability of n_i is updated to

$$R\left(n_i\right) = 1 - \prod_{\beta=1}^{\text{num}_i} \left(1 - R\left(n_i^\beta, u_{pr(n_i^\beta)}, L_{(n_i^\beta)}\right)\right) \tag{14}$$

where $u_{pr(n_i^\beta)}$ and $L_{(n_i^\beta)}$ represent the assigned ECU and ASIL, respectively, of replica n_i^β. Thus, the reliability value of the ACPS function is

$$R(G) = \prod_{n_i \in N} R(n_i). \tag{15}$$

B. FSRV Method

In this section, we aim to find the ECU and ASIL reassignments for all the tasks to enhance the ACPS function's reliability

$$R(G) = \prod_{n_i \in N} R(n_i) \tag{16}$$

while assuring its response time requirement

$$\text{RT}(G) \leqslant D(G). \tag{17}$$

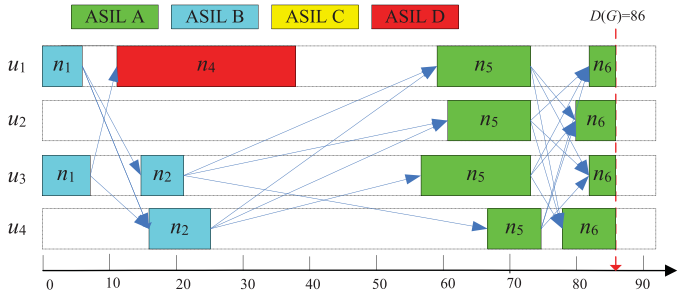


Fig. 9. Task reassignments of n_6 and n_5 using FSRV.

We let $R_{\text{req}}(G)$ represent the reliability requirement of ACPS function G. If

$$R(G) \geqslant R_{\text{req}}(G)$$

then the FSRV is passed; otherwise, it is not passed. In this section, we also employ list scheduling to solve the subject problem.

1) Task Prioritization: Considering that this section aims to enhance reliability while assuring response time requirement, we can sufficiently employ the slack of $D(G) - RT(G)$ (see Fig. 8) as much as possible to enhance reliability. Therefore, we reassign the tasks from the exit to the entry tasks to maximize $R(G)$ while assuring the constraint of $\text{RT}(G) \leq D(G)$.

Let u_l represent the assigned ECU of n_y^β. Then, the latest finish time (LFT) of the current task is defined by

$$\begin{cases} \text{LFT}(n_{\text{exit}}, u_k) = D(G) \\ \text{LFT}(n_i, u_k) = \min\limits_{n_y \in succ(n_i), \beta \in [1, \text{num}_y]} \left\{ \text{AST}(n_y^\beta) - c_{i,y}^{k,l} \right\} \end{cases}. \tag{18}$$

$c_{i,y}^{k,l}$ represents the WCRT of $m_{i,y}$ assuming that n_i is assigned to u_k and n_y is assigned to u_l. $\text{AST}(n_y^\beta)$ represents the actual start time (AST) of n_y^β. Notably, n_i has different $\text{LFT}(n_i, u_k)$ values depending on the assignments of its successor tasks because $c_{i,y}^{k,l}$ is not a fixed value

$$c_{i,y}^{k,l} = \begin{cases} c_{i,y} & k \neq l \\ 0 & k = l \end{cases}. \tag{19}$$

The task prioritization in this section involves prioritizing tasks by the descending order of LFTs using (20) from the exit to the entry tasks

$$\text{LFT}(n_i) = \max_{u_k \in U} \text{LFT}(n_i, u_k). \tag{20}$$

Obviously, the task prioritization for the motivational ACPS function in this section is $\{n_6, n_5, n_3, n_4, n_2, n_1\}$ according to Fig. 8.

2) Response Time Requirement of the Task: The core thought is to transfer the response time requirement of the ACPS function to that of each task. We assume that n_6 and n_5 have been reassigned using the proposed FSRV algorithm shown in Fig. 9. We consider the third task n_3 to be reassigned. In the following discussion, n_3 is employed as an example to explain and calculate the response time requirement of a task.

1) The RTRV-generated task assignments of the current task n_3 is cleared to achieve reassignment and reliability enhancement. When the new reliability value of the current task is calculated, the task assignments of the other tasks cannot be changed to avoid violating the precedence constraints among tasks.
2) Similar to $\text{LFT}(n_i, u_k)$, we obtain $\text{EST}(n_i, u_k)$ of task n_i in u_k caused by the precedence constraints with its predecessors. The purpose of calculating $\text{LFT}(n_i, u_k)$ and $\text{EST}(n_i, u_k)$ is to obtain available slacks. Let u_l represent the assigned ECU of n_x^α. Then, each task has individual $\text{EST}(n_i, u_k)$ in different ECUs as follows:

$$\begin{cases} \text{EST}(n_{\text{entry}}, u_k) = 0 \\ \text{EST}(n_i, u_k) = \max\limits_{n_x \in \text{pred}(n_i), \alpha \in [1, \text{num}_x]} \left\{ \text{AFT}(n_x^\alpha) + c_{x,i}^{l,k} \right\}. \end{cases} \tag{21}$$

That is, the response time requirement of n_i is the combination of its $\text{EST}(n_i, u_k)$ and $\text{LFT}(n_i, u_k)$.
n_3's ESTs and LFTs in all ECUs (see Fig. 9) according to (21) and (18) are as follows:

$$\begin{cases} \text{EST}(n_3, u_1) = 31 \\ \text{EST}(n_3, u_2) = 31 \\ \text{EST}(n_3, u_3) = 31 \\ \text{EST}(n_3, u_4) = 27 \end{cases} \quad \begin{cases} \text{LFT}(n_3, u_1) = 50 \\ \text{LFT}(n_3, u_2) = 50 \\ \text{LFT}(n_3, u_3) = 52 \\ \text{LFT}(n_3, u_4) = 50 \end{cases}. \tag{22}$$

3) Even when EST and LFT have been obtained in each ECU for each task, task reassignment must be further constrained because each ECU is not always available. Considering that some slacks remain in each ECU shown in Fig. 9, task reassignment is actually task insertion. The slack set in ECU u_k for n_i is defined as follows:

$$S_{i,k} = \{S_{i,k,1}, S_{i,k,2}, \ldots, S_{i,k,|S_k|}\} \tag{23}$$

where $S_{i,k,1}$ represents the first slack in u_k for n_i. We define the yth slack $S_{i,k,y}$ as follows:

$$S_{i,k,y} = [t_{\text{s}}(S_{i,k,y}), t_{\text{e}}(S_{i,k,y})]$$

where $t_{\text{s}}(S_{i,k,y})$ and $t_{\text{e}}(S_{i,k,y})$ represent start time and end time of the slack $S_{i,k,y}$, respectively. For example, the slacks of n_3 in four ECUs are as follows:

$$\begin{cases} S_{3,1,1} = [6, 11], S_{3,1,2} = [38, 59], S_{3,1,3} = [73, 82] \\ S_{3,2,1} = [0, 61], S_{3,2,2} = [73, 80] \\ S_{3,3,1} = [7, 15], S_{3,3,2} = [21, 57], S_{3,3,3} = [73, 82] \\ S_{3,4,1} = [0, 16], S_{3,4,2} = [25, 67], S_{3,4,3} = [75, 78] \end{cases}. \tag{24}$$

To avoid violating the precedence constraints among tasks, the current task n_i should be assigned to the available slacks to assure the new EST and LFT constraints, as follows:

$$\text{EST}(n_i, u_k) = \max\{\text{EST}(n_i, u_k), t_{\text{s}}(S_{i,k,y})\} \tag{25}$$

and

$$\text{LFT}(n_i, u_k) = \min\{\text{LFT}(n_i, u_k), t_{\text{e}}(S_{i,k,y})\}. \tag{26}$$

For example, the new EST and LFT values of n_3 in all ECUs are as follows:

$$\begin{cases} \text{EST}(n_3, u_1) = 38 \\ \text{EST}(n_3, u_2) = 31 \\ \text{EST}(n_3, u_3) = 31 \\ \text{EST}(n_3, u_4) = 27 \end{cases} \quad \begin{cases} \text{LFT}(n_3, u_1) = 50 \\ \text{LFT}(n_3, u_2) = 50 \\ \text{LFT}(n_3, u_3) = 52 \\ \text{LFT}(n_3, u_4) = 50 \end{cases}. \tag{27}$$

Compared with (22) and (27), we can see that $\text{EST}(n_3, u_1)$ has been changed from 31 to 38.

3) Reliability Calculation of the ASIL Decomposition Scheme: Considering that the new response time requirement of n_i has been obtained, the reliability value of each ASIL decomposition scheme for n_i is calculated. We consider the example of n_3 with scheme_1 in Fig. 4(d) (L_{C} + L_{A}) to explain the reliability calculation process.

1) We calculate the insertable ASIL level (IAL) in each ECU for each task to explore the possible ASIL decomposition scheme assignment. For example, the IAL values of n_3 in all ECUs are as follows:

$$\begin{cases} \text{IAL}(n_3, u_1) = \{\text{NULL}\} \\ \text{IAL}(n_3, u_2) = \{L_{\text{A}}, L_{\text{B}}, L_{\text{C}}, L_{\text{D}}\} \\ \text{IAL}(n_3, u_3) = \{L_{\text{A}}, L_{\text{B}}, L_{\text{C}}, L_{\text{D}}\} \\ \text{IAL}(n_3, u_4) = \{L_{\text{A}}, L_{\text{B}}, L_{\text{C}}\} \end{cases}. \tag{28}$$

$\text{IAL}(n_3, u_1)$ is NULL because the WCET $w_{3,1,\text{A}}$ is 15 (see Table III), which exceeds its maximum available slack of 12 ($\text{LFT}(n_3, u_1) - \text{EST}(n_3, u_1) = 50 - 38 = 12$). There is no L_{D} for $\text{IAL}(n_3, u_4)$ because the WCET $w_{3,4,\text{D}}$ is 24 (see Table III), which exceeds its maximum available slack of 23 ($\text{LFT}(n_3, u_4) - \text{EST}(n_3, u_4) = 50 - 27 = 23$).
2) Based on IAL, the reliability of each task in each ECU and ASIL is calculated. The failure rates of four ECUs are assumed to be 0.02, 0.025, 0.03, and 0.035, and the results are as follows:

$$\begin{cases} R(n_3, u_2, L_{\text{A}}) = 0.818731,\ R(n_3, u_2, L_{\text{B}}) = 0.778801 \\ R(n_3, u_2, L_{\text{C}}) = 0.740818,\ R(n_3, u_2, L_{\text{D}}) = 0.704688 \\ R(n_3, u_3, L_{\text{A}}) = 0.677057,\ R(n_3, u_3, L_{\text{B}}) = 0.657047 \\ R(n_3, u_3, L_{\text{C}}) = 0.618783,\ R(n_3, u_3, L_{\text{D}}) = 0.582748 \\ R(n_3, u_4, L_{\text{A}}) = 0.571209,\ R(n_3, u_4, L_{\text{B}}) = 0.532592 \\ R(n_3, u_4, L_{\text{C}}) = 0.496585 \end{cases}. \tag{29}$$

3) In the following, we calculate the reliability value of n_3 with scheme_1. Considering that scheme_1 consists of L_{A} + L_{C}, there are two solution based on two orders of (L_{A}, L_{C}) and (L_{C}, L_{A}).

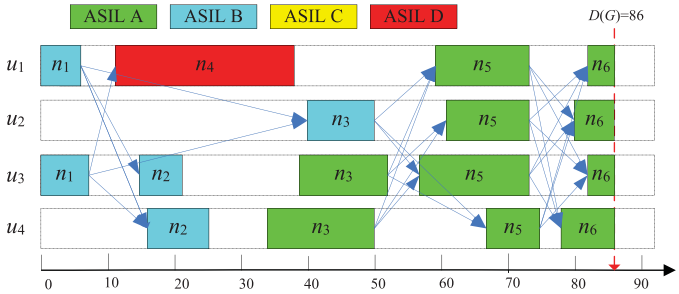


Fig. 10. Reassignment of task n_3 using FSRV.

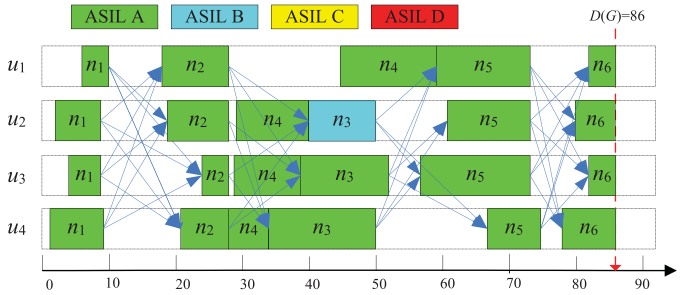


Fig. 11. Task reassignments using FSRV.

a) For the first solution solution_1, we first choose $R(n_3, u_2, L_{\text{A}}) = 0.818731$ because it has the maximum reliability among all L_{A} candidates. We further choose $R(n_3, u_3, L_{\text{C}}) = 0.618783$ among all L_{C} candidates. Note that $R(n_3, u_2, L_{\text{C}}) = 0.740818$ cannot be chosen in this step because u_2 has been occupied by L_{A}. Therefore, the reliability of n_3 with scheme_1 and solution_1 is $R(n_3, \text{scheme}_1, \text{solution}_1) = 0.930897$ calculated by (14).

b) For the second solution solution_2, we first choose $R(n_3, u_2, L_{\text{C}}) = 0.740818$ because it has the maximum reliability among all L_{C} candidates. We further choose $R(n_3, u_3, L_{\text{A}}) = 0.677057$ among all L_{A} candidates. Note that $R(n_3, u_2, L_{\text{A}}) = 0.818731$ cannot be assigned in this step because u_2 has been occupied by L_{C}. Therefore, the reliability of n_3 with scheme_1 and solution_2 is $R(n_3, \text{scheme}_1, \text{solution}_2) = 0.916299$ calculated by (14).

c) The final reliability of n_3 with scheme_1 is

$$\begin{aligned} &R(n_3, \text{scheme}_1) \\ &= \max_{t\in(1,2)} \left(R(n_3, \text{scheme}_1, \text{solution}_t)\right) \\ &= \max(0.930897, 0.916299) \\ &= 0.930897. \end{aligned} \tag{30}$$

4) Choosing the ASIL Decomposition Scheme: Similar to scheme_1, we calculate n_3's reliability values of all the ASIL decomposition schemes as follows:

$$\begin{cases} R(n_3, \text{scheme}_1) = 0.930897 \\ R(n_3, \text{scheme}_2) = 0.924139 \\ R(n_3, \text{scheme}_3) = 0.704688\ . \\ R(n_3, \text{scheme}_4) = 0.972638 \\ R(n_3, \text{scheme}_5) = \text{NULL} \end{cases} \tag{31}$$

$R(n_3, \text{scheme}_5) = \text{NULL}$ because scheme_5 needs four L_{A} and four ECU assignments, but there are only three available ECUs for IAL, as shown in (28).

We choose scheme_4 because it has the maximum reliability among the five values above. Notably, scheme_4 [see Fig. 4(d)] has three replicas with two ASIL A and one ASIL B, and the replicas assignments are shown in Fig. 10.

The remaining tasks n_4, n_2, and n_1 employ the same regular pattern as n_3. The task reassignments of the motivational ACPS function are shown in Fig. 11. The response time is 86. The new reliability of the ACPS function is 0.942343 calculated by (15), and it exceeds the RTRV-generated reliability of 0.451705.

5) Functional Safety Requirement Verification: The FSRV can be operated as follows.

A known reliability requirement $R_{\text{req}}(G)$ of the ACPS function is provided on the basis of the duration/probability of exposure in Table V. If the actual reliability $R(G)$ acquired by FSRV is larger than or equal to the reliability requirement $R_{\text{req}}(G)$

$$R(G) \geqslant R_{\text{req}}(G) \tag{32}$$

then the functional safety requirement is passed; otherwise, it cannot be passed. For this motivational ACPS function, we let the reliability requirement be $R_{\text{req}}(G) = 0.9$, which is the maximum reliability requirement of E4 according to Table V. $R(G) = 0.942343$ can pass the functional safety requirement because

$$0.942343 \geqslant 0.9$$

according to (32).

C. FSRV Algorithm

On the basis of the aforementioned analysis, an algorithm called FSRV is presented to verify the functional safety requirement (described in Algorithm 2).

FSRV transfers the response time requirement of the ACPS function to that of each task. Then, each task chooses the ASIL decomposition scheme with the maximum reliability value among five schemes without violating its response time requirement. In other words, FSRV employs a reliability enhancement technique to verify the functional safety requirement for ACPS. The main details are as follows.

1) In Line 1, FSRV prioritizes tasks by the descending order of LFTs using (20) from the exit to the entry tasks.
2) In Lines 2–13, FSRV chooses the ASIL decomposition scheme with the maximum reliability among five schemes for each task without violating its response time requirement. Particularly, each task's response time requirement is obtained in Lines 6 and 7, and its reliability is updated (enhanced) in Line 12.
3) In Line 14, FSRV calculates the ACPS function's reliability $R(G)$.

Algorithm 2: The FSRV Algorithm.

Input: $U = \{u_1, u_2, \ldots, u_{|U|}\}$, $\{L_{\rm A}, L_{\rm B}, L_{\rm C}, L_{\rm D}\}$, G, $D(G)$, RTRV-generated assignments.
Output: RT(G), $R(G)$, and related values

1: Prioritize the tasks in a list lft_task_list by the descending order of LFTs using (20) from the exit to the entry tasks;
2: **while** (there are tasks in lft_task_list) **do**
3: $n_i \leftarrow$ lft_task_list.out();
4: **for** $(g \leftarrow 1; g \leqslant 5; g++)$ **do**
5: **for** (each ECU $u_k \in U$) **do**
6: Calculate LFT(n_i, u_k) using (26);
7: Calculate EST(n_i, u_k) using (25);
8: **end for**
9: Calculate $R(n_i, \text{scheme}_g)$ according to the "(3) Reliability calculation of the ASIL decomposition scheme" in Section IV-B;
10: **end for**
11: Choose the ASIL decomposition scheme with the maximum reliability among five schemes of n_i;
12: Update $R(n_i)$ using (14);
13: **end while**
14: Update the $R(G)$ using (15);
15: **if** $(R(G) \geqslant R_{\rm req}(G))$ **then**
16: Pass the functional safety requirement verification;
17: **return**;
18: **else**
19: Do not pass the functional safety requirement verification;
20: **return**;
21: **end if**

4) In Lines 15–21, FSRV compares the ACPS function's reliability $R(G)$ with its reliability requirement $R_{\rm req}(G)$ to verify the functional safety requirement.

Similar to RTRV, the time complexity of the FSRV algorithm is also $O(|N|^2 \times |U|)$. The reason is that traversing all tasks needs $O(|N|)$ time (Lines 2–13), and calculating the EST(n_i, u_k) and LFT(n_i, u_k) needs $O(|N| \times |U|)$ time (Lines 5–8).

VI. Functional Safety-Critical Energy Consumption Reduction

If the FSRV is passed, then we begin to solve the third subproblem of FSEO in the third stage.

A. Power and Energy Models

Given a DVFS-capable system, the system-level power model proposed in [23] and employed in [24]–[27], [41], [42], [72] is adopted in this paper. In this power model, the power at frequency f is expressed by

$$P(f) = P_{\rm s} + \hbar(P_{\rm ind} + P_{\rm d}) = P_{\rm s} + \hbar(P_{\rm ind} + C_{\rm ef} f^m). \quad (33)$$

P_s represents the static power and can be removed only by powering OFF the ACPS. $P_{\rm ind}$ represents the frequency-independent dynamic power and can be removed by switching the state of ACPS to a sleep state. $P_{\rm d}$ represents the frequency-dependent dynamic power. $\hbar$ represents the ACPS states and indicates whether dynamic powers are currently being consumed in the ACPS. When the ACPS is active, $\hbar = 1$; otherwise, $\hbar = 0$. $C_{\rm ef}$ represents the effective switching capacitance. m represents the dynamic power exponent, which should not be smaller than 2. Both $C_{\rm ef}$ and m are ECU-dependent constants.

$P_{\rm s}$ is always consumed and unmanageable except for powering OFF the engine of ACPS. Considering that the functional safety is based on the automobile being driving, this paper only manages the dynamic power (i.e., $P_{\rm ind}$ and $P_{\rm d}$). Reducing $P_{\rm d}$ does not directly lead to a reduction in energy consumption due to the existence of $P_{\rm ind}$. The energy-efficient frequency $f_{\rm ee}$ was found in [24] and employed in [25]–[27], [41], [42], [72]

$$f_{\rm ee} = \sqrt[m]{\frac{P_{\rm ind}}{(m-1)C_{\rm ef}}}. \quad (34)$$

If an ECU's frequency varies from a minimum available frequency $f_{\rm min}$ to the maximum frequency $f_{\rm max}$, then the lowest energy-efficient frequency to execute a task should be

$$f_{\rm low} = \max(f_{\rm min}, f_{\rm ee}) \quad (35)$$

according to [24]–[27], [41], [42], [72]. Therefore, the actual effective frequency f_v is larger than or equal to $f_{\rm low}$ and less than or equal to $f_{\rm max}$.

Given that ECUs are heterogeneous in ACPS, each ECU should have individual power parameters [27], [41], [42]. The frequency-independent dynamic power set is defined as

$$\{P_{1,\rm ind}, P_{2,\rm ind}, \ldots, P_{|U|,\rm ind}\}. \quad (36)$$

The set of frequency-dependent dynamic powers is defined as

$$\{P_{1,\rm d}, P_{2,\rm d}, \ldots, P_{|U|,\rm d}\}. \quad (37)$$

The set of effective switching capacitances is defined as

$$\{C_{1,\rm ef}, C_{2,\rm ef}, \ldots, C_{|U|,\rm ef}\}. \quad (38)$$

The set of dynamic power exponents is defined as

$$\{m_1, m_2, \ldots, m_{|U|}\}. \quad (39)$$

The set of lowest energy-efficient frequency is defined as

$$\{f_{1,\rm low}, f_{2,\rm low}, \ldots, f_{|U|,\rm low}\} \quad (40)$$

and the set of actual effective frequency is defined as

$$\left\{ \begin{array}{l} \{f_{1,\rm low}, f_{1,a}, f_{1,b}, \ldots, f_{1,\rm max}\}, \\ \{f_{2,\rm low}, f_{2,a}, f_{2,b}, \ldots, f_{2,\rm max}\}, \\ \vdots \\ \{f_{|U|,\rm low}, f_{|U|,a}, f_{|U|,b}, \ldots, f_{|U|,\rm max}\} \end{array} \right\}. \quad (41)$$

Let $E(n_i, u_k, L_h, f_{k,v})$ denote the ECU-generated energy consumption of task n_i in ECU u_k and ASIL L_h with frequency

$f_{k,v}$

$$E(n_i, u_k, L_h, f_{k,v}) = (P_{k,\text{ind}} + C_{k,\text{ef}} \times (f_{k,v})^{m_k}) \times w_{i,k,h} \times \frac{f_{k,\max}}{f_{k,v}}. \tag{42}$$

n_i's energy consumption is the sum of those of its all replicas as follows:

$$E(n_i) = \sum_{\beta=1}^{\text{num}_i} E(n_i^\beta) = \sum_{\beta=1}^{\text{num}_i} E(n_i^\beta, u_{pr(n_i^\beta)}, L_{pr(n_i^\beta)}, f_{pr(n_i^\beta),hz(n_i^\beta)}) \tag{43}$$

where $u_{pr(n_i^\beta)}$, $L_{pr(n_i^\beta)}$, and $f_{pr(n_i^\beta),hz(n_i^\beta)}$ represent the assigned ECU, ASIL, and frequency, respectively, of replica n_i^β. The ACPS function's energy consumption is the sum of those of all its tasks as follows:

$$E(G) = \sum_{i=1}^{|N|} E(n_i). \tag{44}$$

B. Energy-Enabled Reliability Model

Different frequencies cause different failure rates in ACPS [7], [23], [25]–[27]. Let $\lambda_{k,\max}$ denote the failure rate of ECU u_k with the maximum frequency. Then, we let the failure rate of u_k with frequency $f_{k,v}$ be $\lambda_{k,v}$, which is calculated by [7], [23], [25]–[27]

$$\lambda_{k,v} = \lambda_{k,\max} \times 10^{\frac{d(f_{k,\max} - f_{k,v})}{f_{k,\max} - f_{k,\min}}}. \tag{45}$$

d represents the sensitivity of failure rates to voltage scaling and is a constant. The reliability of task n_i executed in ECU u_k and ASIL L_h with frequency $f_{k,v}$ is calculated by

$$R(n_i, u_k, L_h, f_{k,v}) = e^{-\lambda_{k,v} \times \frac{w_{i,k,h} \times f_{k,\max}}{f_{k,v}}} = e^{-\lambda_{k,\max} \times 10^{\frac{d(f_{k,\max} - f_{k,v})}{f_{k,\max} - f_{k,\min}}} \times \frac{w_{i,k,h} \times f_{k,\max}}{f_{k,v}}}. \tag{46}$$

In (46), reliability and frequency are simultaneously increasing in the same ECU. Hence, scaling down the frequency to reduce energy consumption can lead to low reliability. In a DVFS-capable ACPS, the reliability of n_i is updated to

$$R(n_i) = 1 - \prod_{\beta=1}^{\text{num}_i} \left(1 - R\left(n_i^\beta, u_{pr(n_i^\beta)}, L_{(n_i^\beta)}, f_{pr(n_i^\beta),hz(n_i^\beta)}\right)\right). \tag{47}$$

C. Functional Safety-Critical Energy Consumption Reduction Method

In this section, we aim to find the ECU and ASIL assignments of all the tasks to optimize the ACPS function's energy consumption

$$E(G) = \sum_{i=1}^{|N|} E(n_i) \tag{48}$$

while assuring its functional safety requirement

$$\text{RT}(G) \leqslant D(G) \tag{49}$$

and

$$R(G) \geqslant R_{\text{req}}(G). \tag{50}$$

The list scheduling is still used in this section. Our core strategy is to transfer the functional safety requirement of the ACPS function to that of each task.

1) Task Prioritization: We still employ the task prioritization standard of LFT, namely, prioritizing tasks by the descending order of LFTs using (20) from the exit to the entry tasks. Then, we assume that the current task to be assigned is $n_{s(y)}$, which denotes the yth assigned task. $\{n_{s(1)}, n_{s(2)}, \ldots, n_{s(y-1)}\}$ denotes the assigned task set using the FSRV algorithm proposed in the previous section. $\{n_{s(y+1)}, n_{s(y+2)}, \ldots, n_{s(|N|)}\}$ denotes the assigned task set using the FSEO algorithm proposed in this section.

2) Reliability Requirement of Task: When assigning $n_{s(y)}$, the reliability of the ACPS function is calculated by

$$R(G) = \prod_{x=1}^{y-1} R_{\text{FSRV}}(n_{s(x)}) \times R(n_{s(y)}) \times \prod_{z=y+1}^{|N|} R_{\text{FSEO}}(n_{s(z)}) \tag{51}$$

where $R_{\text{FSRV}}(n_{s(x)})$ denotes the FSRV-generated reliability of $n_{s(x)}$ and $R_{\text{FSEO}}(n_{s(z)})$ denotes the FSEO-generated reliability of $n_{s(z)}$.

The ACPS function's reliability value $R(G)$ must be larger than or equal to the reliability requirement $R_{\text{req}}(G)$ according to the problem statement. That is, we have

$$\prod_{x=1}^{y-1} R_{\text{FSRV}}(n_{s(x)}) \times R(n_{s(y)}) \times \prod_{z=y+1}^{|N|} R_{\text{FSEO}}(n_{s(z)}) \geqslant R_{\text{req}}(G). \tag{52}$$

Hence, the following constraint is made for task $n_{s(y)}$.

$$R(n_{s(y)}) \geqslant \frac{R_{\text{req}}(G)}{\prod_{x=1}^{y-1} R_{\text{FSRV}}(n_{s(x)}) \times \prod_{z=y+1}^{|N|} R_{\text{FSEO}}(n_{s(z)})}.$$

We let the reliability requirement for current task $n_{s(y)}$ be

$$R_{\text{req}}(n_{s(y)}) = \frac{R_{\text{req}}(G)}{\prod_{x=1}^{y-1} R_{\text{FSRV}}(n_{s(x)}) \times \prod_{z=y+1}^{|N|} R_{\text{FSEO}}(n_{s(z)})}. \tag{53}$$

Hence, the reliability requirement of the ACPS function is transferred to each task. The reliability requirement of the ACPS function can be assured, as long as the reliability requirement of each task in (54) is assured

$$R(n_{s(y)}) \geqslant R_{\text{req}}(n_{s(y)}). \tag{54}$$

3) Response Time Requirement of the Task: The response time requirement of the task is the combination of its ESTs [see (25)] and LFTs [see (26)]. We still assume that n_6 has been reassigned using the FSEO algorithm. We consider n_5 as the second task to be reassigned. In the following discussion, we employ n_5 as the example to explain the process of calculating the response time requirement of the current task. As shown in

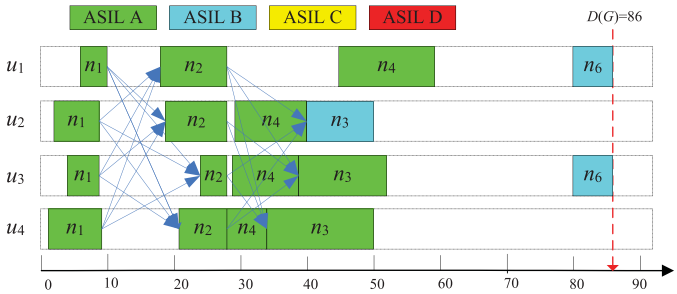


Fig. 12. n_6 has been reassigned using FSEO, and n_5 is ready to be assigned using FSEO.

TABLE VI
POWER PARAMETERS OF ECUS (u_1, u_2, u_3, AND u_4)

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	0.7	2.5	0.27	1.0
u_3	0.07	1.0	2.8	0.31	1.0
u_4	0.07	1.0	2.7	0.31	1.0

Fig. 12, the new EST and LFT values of n_5 in all ECUs are as follows:

$$\begin{cases} \text{EST}(n_5,u_1)=59 \\ \text{EST}(n_5,u_2)=59 \\ \text{EST}(n_5,u_3)=57 \\ \text{EST}(n_5,u_4)=59 \end{cases} \quad \begin{cases} \text{LFT}(n_5,u_1)=75 \\ \text{LFT}(n_5,u_2)=75 \\ \text{LFT}(n_5,u_3)=75 \\ \text{LFT}(n_5,u_4)=75 \end{cases}. \tag{55}$$

4) Energy Consumption of the Task: After determining the response time and reliability requirements of the current task n_i (i.e., $n_{s(y)}$), we choose the ASIL decomposition scheme that has the minimum energy consumption for n_i while assuring its response time and reliability requirements. In the following text, we explain how to calculate the energy consumption of n_5.

1) The reliability requirement of n_5 is 0.963953 calculated by (53).
2) The response requirement of n_5 is shown in (55).
3) The IAL values of n_5 in all ECUs are as follows:

$$\begin{cases} \text{IAL}(n_5,u_1)=\{L_\text{A},L_\text{B}\} \\ \text{IAL}(n_5,u_2)=\{L_\text{A},L_\text{B},L_\text{C}\} \\ \text{IAL}(n_5,u_3)=\{L_\text{A},L_\text{B}\} \\ \text{IAL}(n_5,u_4)=\{L_\text{A},L_\text{B},L_\text{C},L_\text{D}\} \end{cases} \tag{56}$$

such that all the five ASIL decomposition schemes may be chosen.

4) We assume that the power parameters for all ECUs are known (shown in Table VI), where the maximum frequency $f_{k,\text{max}}$ for each ECU is 1 and the frequency resolution is 0.01. All the parameter units are omitted in this example as pointed out earlier. The lowest energy-efficient frequency $f_{k,\text{low}}$ for each ECU is obtained according to (35) shown in Table VI.

Considering that L_A can be available in four ECUs, as shown in (56), we can list all the possible combination assignments of ECUs, ASILs, and frequencies for n_5. scheme$_4$ is complex because it contains two L_A and one L_B. Therefore, we employ n_5 with scheme$_4$ to explain the process. Given that the lower frequency in $[f_{k,\text{low}}, f_{k,\text{max}}]$ means lower energy consumption and lower reliability, we prioritize these combination assignments by the ascending order of $R(n_5,u_k,L_h,f_{k,h})$, as shown in Table VII, where each row represents a possible combination assignment.

TABLE VII
POSSIBLE COMBINATION ASSIGNMENTS OF ECUS, ASILS, AND FREQUENCIES FOR n_5 WITH SCHEME$_4$

u_k	L_h	$f_{k,v}$	$R(n_5,u_k,L_h,f_{k,v})$	$E(n_5,u_k,L_h,f_{k,v})$
u_4	L_A	0.5	0.051279	3.58
u_4	L_A	0.51	0.059806	3.64
			.....	
u_4	L_B	0.79	0.409478	7.58
u_2	L_A	0.75	0.414747	6.10
			.....	
u_3	L_A	0.89	0.459084	14.23
u_4	L_B	0.82	0.459205	7.99
u_2	L_A	0.78	0.463090	6.40
			.....	
u_4	L_A	**0.92**	**0.672013**	**7.55**
u_2	L_B	**0.97**	**0.672579**	**9.94**
u_1	L_A	**0.92**	**0.676805**	**10.02**

a) We choose the first combination assignment $<u_4, L_\text{A}, 0.5>$. In this assignment, the reliability and energy consumptions are 0.051279 and 3.58, respectively. Obviously, 0.051279 cannot assure n_5's reliability requirement of 0.963953. Therefore, we must further explore the next combination assignments.
b) The second combination assignment is considered. This combination is also assigned to u_4. Considering that u_4 can only be assigned once due to the use of active replication, we must delete the previous first combination assignment and choose the current assignment by nonstop frequency replacement until $<u_4, L_\text{B}, 0.79>$. Therefore, the reliability of n_5 is changed to 0.409478, which still cannot assure its reliability requirement of 0.963953. Therefore, we must explore the next assignments again.
c) The next possible combination assignment is moved to u_2, where the combination assignment is $<u_2, L_\text{A}, 0.75>$. In this case, the assignments $<u_4, L_\text{B}, 0.79>$ and $<u_2, L_\text{A}, 0.75>$ are chosen, and the reliability of n_5 under these two assignments reaches 0.654395 calculated by (47). However, the reliability requirement of 0.963953 is still not assured, so the next assignments must be evaluated.
d) The process is continued until the assignments $<u_3, L_\text{A}, 0.89>$, $<u_4, L_\text{B}, 0.82>$, and $<u_2, L_\text{A}, 0.78>$ are chosen. These assignments comply with the requirement that scheme$_4$ contains two L_A and one L_B. In this case, the reliability of n_5 is changed up to 0.842941, which still cannot assure the reliability requirement of 0.963953.
e) Finally, the combination assignments denoted in bold in Table VII are chosen because the reliability of n_5 reaches 0.965292, which exceeds the reliability requirement of 0.963953. The energy consumption of n_5 is 27.51. The

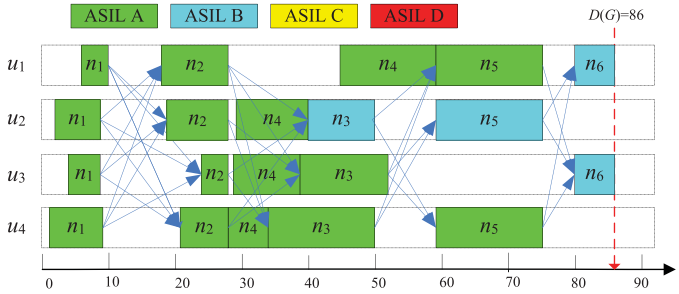


Fig. 13. Replica assignment of n_5 using FSEO.

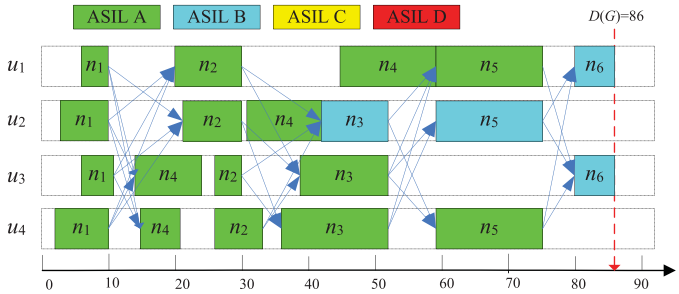


Fig. 14. Task assignments of the motivational ACPS function using FSEO.

remaining assignments can be skipped because the reliability requirement of n_5 has been assured.

5) Choosing ASIL Decomposition Scheme for the Task: Similar to scheme_4, the remaining ASIL decomposition schemes of n_5 also quantitatively choose the combinations of ECUs, ASILs, and frequencies; the final reliability and energy consumptions are shown in (57). We choose the ASIL decomposition scheme with the minimum energy consumption for n_5 while assuring its functional safety requirement. The results in (57) show that only scheme_4 and scheme_5 can assure the reliability requirement of n_5 (the response time requirement has been assured by limiting EST and LFT), whereas the other schemes cannot (the values are denoted as NULLs). Given that scheme_4 has less energy consumption than scheme_5, scheme_4 is chosen for n_5

$$\begin{cases} R(n_5, \text{scheme}_1) = \text{NULL}, E(n_5, \text{scheme}_1) = \text{NULL} \\ R(n_5, \text{scheme}_2) = \text{NULL}, E(n_5, \text{scheme}_2) = \text{NULL} \\ R(n_5, \text{scheme}_3) = \text{NULL}, E(n_5, \text{scheme}_3) = \text{NULL} \\ R(n_5, \text{scheme}_4) = 0.965292, E(n_5, \text{scheme}_4) = 27.51 \\ R(n_5, \text{scheme}_5) = 0.979769, E(n_5, \text{scheme}_5) = 41.05 \end{cases} . \tag{57}$$

6) Energy Consumption of the ACPS Function: The replica assignment of n_5 is shown in Fig. 13.

Similar to n_5, the remaining tasks n_4, n_3, n_2, and n_1 choose corresponding ASIL decomposition schemes with ECU, ASIL, and frequency combination assignments, as shown in Fig. 14, where the actual reliability, response time, and energy consumption of the ACPS function are $R(G) = 0.900001$, $RT(G) = 86$, and $E(G) = 159.41$ calculated by (15), (10), and (44), respectively.

Algorithm 3: The FSEO Algorithm.

Input: $U = \{u_1, u_2, \ldots, u_{|U|}\}$, $\{L_\text{A}, L_\text{B}, L_\text{C}, L_\text{D}\}$, G, $D(G)$, $R_\text{req}(G)$, FSRV-generated assignments.
Output: $\text{RT}(G)$, $R(G)$, $E(G)$, and related values

Prioritize the tasks in a list lft_task_list by the descending order of LFTs using (20) from the exit to the entry tasks;
while (there are tasks in lft_task_list) **do**
  $n_i \leftarrow$ lft_task_list.out();
  for (each ECU $u_k \in U$) **do**
    Calculate n_i's response time requirement $\text{EST}(n_i, u_k)$ and $\text{LFT}(n_i, u_k)$ using (25) and (26), respectively;
  end for
  Calculate n_i's reliability requirement $R_\text{req}(n_i)$ using (53);
  for ($g \leftarrow 1; g \leqslant 5; g++$) **do**
    for (each ASIL $L_h \in \text{scheme}_g$) **do**
      for (each ECU $u_k \in U$) **do**
        for (each frequency $f_{k,v}$ in from $f_{k,\text{low}}$ and $f_{k,\text{max}}$) **do**
          Calculate $R(n_i, u_k, L_h, f_{k,v})$ using (46);
          Calculate $E(n_i, u_k, L_h, f_{k,v})$ using (42);
        end for
      end for
    end for
    Obtain the ECU, ASIL, and frequency combination assignments of n_i according to the "(4) Energy consumption of the task" in Section V-C;
    Calculate $R(n_i, \text{scheme}_g)$ using (47) based on obtained ECU, ASIL, and frequency combination assignments;
    Calculate $E(n_i, scheme_g)$ using (43) based on obtained ECU, ASIL, and frequency combination assignments;
  end for
  Choose the ASIL decomposition scheme with minimum energy consumption while assuring $R(n_i, scheme_g) \geqslant R_\text{req}(n_i)$
end while
$RT(G) \leftarrow D(G)$;
Calculate $E(G)$ using Eq. (44);
Calculate $R(G)$ using Eq. (15);
return.

D. FSEO Algorithm

We propose an algorithm FSEO (described in Algorithm 3) to optimize the energy consumption of the ACPS function while assuring its functional safety requirement.

FSEO transfers both the response time and reliability requirements of the ACPS function to those of each task. Then, each task chooses the ASIL decomposition scheme with the minimum energy consumption among five schemes without violating its

TABLE VIII
RESULTS OF THE MOTIVATIONAL ACPS FUNCTION USING FSRV AND FSEO ALGORITHMS

	$D(G)$	$R_{\text{req}}(G)$	$R(G)$	$E(G)$
FSRV	86	0.9	0.959330	350.12
FSEO	86	0.9	0.900002	159.41

response time and reliability requirements. That is, FSEO employs a DVFS-enabled energy consumption optimization technique to implement energy-efficient functional safety for ACPS. The main details are as follows.

1) In Line 1, FSEO prioritizes tasks by the descending order of LFTs using (20) from the exit to the entry tasks.
2) In Lines 2–22, FSEO chooses the ASIL decomposition scheme with the minimum energy consumption among five schemes for each task without violating its response time and reliability requirements. Particularly, each task's response time is obtained in Lines 4–6, each task's reliability requirement is obtained in Line 7, and each task's reliability and energy consumption in each ASIL scheme are obtained in Lines 18 and 19.
3) In Lines 23–25, FSEO calculates the ACPS function's response time RT(G), reliability $R(G)$, and energy consumption $E(G)$.

The time complexity of the FSEO algorithm is $O(|N|^2 \times |U|+|N| \times |U| \times |F|)$. The detailed process is below.

1) Traversing all tasks needs $O(|N|)$ time (Lines 2–22).
2) Calculating the EST(n_i, u_k) and LFT(n_i, u_k) needs $O(|N| \times |U|)$ time (Lines 4–6).
3) Calculating $R\left(n_i, u_k, L_h, f_{k,v}\right)$ and $E\left(n_i, u_k, L_h, f_{k,v}\right)$ should traverse all ECUs and frequencies and thus needs $O(|U| \times |F|)$ time (Lines 8–16).
4) Calculating $R(n_i, \text{scheme}_g)$ and $E(n_i, \text{scheme}_g)$ also needs $O(|U| \times |F|)$ time (Lines 17–19). Notice that that (2), (3), and (4) are not nested.

Table VIII shows the results of the motivational ACPS function using FSRV and FSEO algorithms. Compared with the FSRV-generated energy consumption of 350.12, FSEO reduces the energy consumption by about half. Although the reliability obtained by FSEO is reduced from 0.959330 to 0.900002, it is still larger than the reliability requirement of 0.9.

VII. EXPERIMENTAL EVALUATION

We employ the actual response time $RT(G)$ [(10), reliability $R(G)$ ((15)], and the energy consumption $E(G)$ [see (44)] of the ACPS function as evaluation metrics because the energy-efficient functional safety for ACPS in this paper is about response time, reliability, and energy consumption. To the best of our knowledge, the most relevant algorithm for this paper is the EFSRG algorithm proposed in [27]. However, EFSRG differs from the proposed FSEO algorithm in the following three points.

1) EFSRG aims to optimize energy consumption of a DAG function while assuring its reliability requirement, whereas FSEO aims to optimize energy consumption of a DAG function while assuring its reliability and response time requirements.
2) EFSRG only has one criticality level for task execution, whereas FSEO has four criticality levels (i.e., ASIL A–ASIL D) based on the ISO 26262 standard.
3) EFSRG can have $|U|$ replicas for each task, whereas FSEO at most has four replicas for each task according to ASIL decomposition in ISO 26262.

TABLE IX
PARAMETER VALUES OF EXPERIMENTS

Parameters	Scopes
WCETs of ECU tasks in ASIL A $w_{i,k,\text{A}}$	100–400 μs
WCETs of ECU tasks in ASIL B $w_{i,k,\text{B}}$	500–800 μs
WCETs of ECU tasks in ASIL C $w_{i,k,\text{C}}$	900–1200 μs
WCETs of ECU tasks in ASIL D $w_{i,k,\text{D}}$	1300–1600 μs
WCRTs of CAN messages $c_{i,j}$	100–400 μs
Failure rates of ECUs λ_k	0.005/–0.0001/μs

To achieve a sufficient comparison, we extend EFSRG to four ASIL levels and implement replication in accordance with ASIL decomposition explained in Section III-C. We name this new algorithm EFSRG_ASIL. Therefore, the algorithms compared with the proposed RTRV, FSRV, and FSEO algorithms are EFSRG and EFSRG_ASIL.

We employ the parameter values of real ACPS as experimental data, as summarized in Table IX.

The number of ECUs in the ACPS is 16. All ECUs have the fixed values as below. The effective switching capacitance $C_{k,\text{ef}}$ is 1, the dynamic power exponent m_k is 3, the maximum frequency is $f_{k,\max} = 1$ GHz, and the discrete frequency resolution is 0.01 GHz. The frequency-dependent dynamic power $P_{k,\text{d}}$ is different for different ECUs and is calculated by $C_{k,\text{ef}} \times {f_{k,h}}^{m_k}$. Different ECUs have different frequency-independent dynamic powers, the scope of which is 0.03 W $\leqslant P_{k,\text{ind}} \leqslant$ 0.07 W. In summary, the heterogeneity of ECU is reflected in the frequency-independent dynamic power, the failure rate, and WCET. All the algorithms are implemented by employing Java. The experimental results are obtained by running the algorithms based on given parameter values.

A. Real-Life Function

The real-life ACPS function shown in Fig. 15 is adopted from [7] and [13]. This ACPS function contains six functional blocks: engine controller consisting of n_1–n_7, automatic gear box consisting of n_8–n_{11}, antilocking brake system consisting of n_{12}–n_{17}, wheel angle sensor consisting of n_{18}–n_{19}, suspension controller consisting of n_{20}–n_{24}, and body work consisting of n_{25}–n_{31}.

Experiment 1: Considering that the reliability requirement for exposure E3 (medium probability) is larger than 0.9 and that E2 (low probability) is equal to 0.9 according to Table V, we change the reliability requirement from 0.9 to 0.99 with a 0.01 increment. This experiment aims to observe the energy consumptions of a real-life ACPS function under varying reliability requirements. As the lower bound of the ACPS function is 4269 μs using RTRV, the response time requirement is given

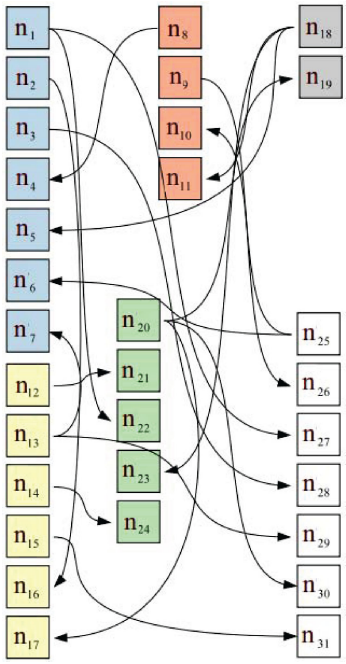


Fig. 15. Real-life ACPS function.

TABLE X
RESPONSE TIME VALUES (UNIT: μS) OF THE REAL-LIFE ACPS FUNCTION UNDER VARYING RELIABILITY REQUIREMENTS

$R_{\text{req}}(G)$ $=x$	x=0.9	x=0.91	x=0.92	x=0.93	x=0.94	x= 0.95	x=0.96	x=0.97	x=0.98	x=0.99
RTRV	4,269	4,269	4,269	4,269	4,269	4,269	4,269	4,269	4,269	4,269
FSRV	5,269	5,269	5,269	5,269	5,269	5,269	5,269	5,269	5,269	5,269
FSEO	5,269	5,269	-	-	-	-	-	-	-	-
EFSRG	17,239	18,181	17,039	16,965	16,673	16,624	16,388	17,151	16,869	15,623
EFSRG _ASIL	9,791	9,708	-	-	-	-	-	-	-	-

TABLE XI
RELIABILITY VALUES OF THE REAL-LIFE ACPS FUNCTION UNDER VARYING RELIABILITY REQUIREMENTS

$R_{\text{req}}(G)$ $=x$	x=0.9	x=0.91	x=0.92	x=0.93	x=0.94	x=0.95	x=0.96	x=0.97	x=0.98	x=0.99
RTRV	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001
FSRV	0.9185	0.9185	0.9185	0.9185	0.9185	0.9185	0.9185	0.9185	0.9185	0.9185
FSEO	0.9001	0.9101	-	-	-	-	-	-	-	-
EFSRG	0.9001	0.9101	0.9201	0.9301	0.9401	0.9501	0.9601	0.9701	0.9801	0.9901
EFSRG _ASIL	0.9001	0.9101	-	-	-	-	-	-	-	-

by the lower bound plus 1000 μs. That is, the response time requirement is fixed at 5269 μs.

Tables X–XII show the response time values, reliability values, and energy consumptions of the real-life ACPS function under varying reliability requirements.

1) Tables X–XII show that the RTRV-generated response time values, reliability values, and energy consumptions are fixed because RTRV is merely employed for response time verification and does not involve reliability requirements.
2) Table X shows that FSRV-generated response time values are fixed and equal to the fixed response time requirement of 5269 μs because FSRV aims to enhance the reliability while assuring the response time requirement. Therefore, FSRV-generated reliability values are also fixed at 0.9185 as shown in Table XI.
3) Tables X–XII show that only when the reliability requirements are 0.9 and 0.91, can FSEO obtain two valid values, whereas the eight other reliability requirements (0.92–0.99) denoted by "-" are invalid. The reason is that when verifying the reliability requirement using FSRV, only 0.9 and 0.91 are passed (i.e., return true), and the other reliability requirements return false.
4) Table XI shows that EFSRG can always assure corresponding reliability requirements in all cases, but it cannot assure the response time requirement of 5269 μs, as shown in Table X. The reason is that each task has as many as $|N|$ replicas, and all these replicas are executed at the lowest ASIL. In other words, EFSRG easily assures the reliability requirement without considering the response time requirement.
5) Tables X–XII show that EFSRG_ASIL merely obtains two valid values. The reason is that EFSRG_ASIL employs ASIL decomposition, and each task has at most five schemes. Thus, high reliability requirements are difficult to assure in this case.
6) We focus on the energy consumptions in Table XII. For the valid results when the reliability requirement is 0.9 and 0.91, EFSRG_ASIL generates the minimum energy consumption, followed by FSEO, EFSRG, FSRV, and RTRV. Although EFSRG_ASIL is better than FSEO in optimizing energy consumption, its response time requirement cannot be assured, as shown in Table X. Therefore, FSEO is the best from the perspective of energy-efficient functional safety for ACPS. In addition, FSEO outperforms EFSRG and FSRV by 59% and 86%, respectively; such an advantage indicates that using the DVFS-enabled energy consumption optimization technique is necessary. The results obtained by EFSRG in Table XII also confirm that a high reliability can result in large energy consumption of an ACPS function, and reliability maximization and energy consumption minimization are conflicting.

TABLE XII
ENERGY CONSUMPTIONS (UNIT: WμS) OF THE REAL-LIFE ACPS FUNCTION UNDER VARYING RELIABILITY REQUIREMENTS

$D(G)$ $=x$	x=0.9	x=0.91	x=0.92	x= 0.93	x=0.94	x=0.95	x=0.96	x=0.97	x=0.98	x=0.99
RTRV	189,679	189,679	189,679	189,679	189,679	189,679	189,679	189,679	189,679	189,679
FSRV	180,580	180,580	180,580	180,580	180,580	180,580	180,580	180,580	180,580	180,580
FSEO	25,724	25,556	-	-	-	-	-	-	-	-
EFSRG	62,404	63,406	64,440	65,357	66,329	67,557	68,901	71,078	73,720	78,134
EFSRG _ASIL	20,233	20,641	-	-	-	-	-	-	-	-

Experiment 2: The response time requirement should be increased because the function safety requirement of the ACPS function cannot be assured using FSEO when the reliability requirement is larger than or equal to 0.92 in Experiment 1. In this experiment, we let the response requirement be increased from 5,269 μs to larger values with 500 μs increments to determine the point that the reliability requirement can be assured.

TABLE XIII
RESPONSE TIME VALUES (UNIT: μS) OF THE REAL-LIFE ACPS FUNCTION UNDER VARYING RESPONSE TIME REQUIREMENTS

$D(G)$ =x	x= 4,269	x= 4,769	x= 5,269	x= 5,769	x= 6,269	x= 6,769	x= 7,269	x= 7,769	x= 8,269	x= 8,789
RTRV	4,269	4,269	4,269	4,269	4,269	4,269	4,269	4,269	4,269	4,269
FSRV	4,269	4,769	5,269	5,769	6,269	6,769	7,269	7,769	8,269	8,769
FSEO	-	-	-	-	6,269	6,769	7,269	7,769	8,269	8,769
EFSRG	16,624	16,624	16,624	16,624	16,624	16,624	16,624	16,624	16,624	16,624
EFSRG _ASIL	-	-	-	-	-	-	-	-	-	-

TABLE XIV
RELIABILITY VALUES OF THE REAL-LIFE ACPS FUNCTION UNDER VARYING RESPONSE TIME REQUIREMENTS

$D(G)$ =x	x= 4,269	x= 4,769	x= 5,269	x= 5,769	x= 6,269	x= 6,769	x= 7,269	x= 7,769	x= 8,269	x= 8,789
RTRV	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001	0.6001
FSRV	0.8837	0.9081	0.9125	0.9277	0.9539	0.9577	0.9577	0.9577	0.9577	0.9577
FSEO	-	-	-	-	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501
EFSRG	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501
EFSRG _ASIL	-	-	-	-	-	-	-	-	-	-

TABLE XV
ENERGY CONSUMPTIONS (UNIT: WμS) OF THE REAL-LIFE ACPS FUNCTION UNDER VARYING RESPONSE TIME REQUIREMENTS

$D(G)$ =x	x= 4,269	x= 4,769	x= 5,269	x= 5,769	x= 6,269	x= 6,769	x= 7,269	x= 7,769	x= 8,269	x= 8,789
RTRV	189,679	189,679	189,679	189,679	189,679	189,679	189,679	189,679	189,679	189,679
FSRV	180,334	180,663	180,580	181,632	180,614	180,520	180,520	180,520	180,520	180,520
FSEO	-	-	-	-	26,004	26,004	26,004	26,004	26,004	26,004
EFSRG	67,557	67,557	67,557	67,557	67,557	67,557	67,557	67,557	67,557	67,557
EFSRG _ASIL	-	-	-	-	-	-	-	-	-	-

To reduce complexity and observe the results easily, the reliability requirement is fixed at 0.95, which falls in the reliability requirement scope of exposure E3 in Table V.

Tables XIII–XV show the response time values, reliability values, and energy consumptions of the real-life ACPS function under varying response time requirements.

1) The results in Tables XIII–XV show that when the response time requirement reaches 6269 μs, the functional safety requirement is assured (i.e., valid results are shown) using FSEO. These results confirm that a long response time can result in a high reliability of an ACPS function, and implementing minimum response time and maximum reliability is conflicting.
2) A strange finding is that EFSRG_ASIL does not obtain valid results in all the cases. The reason is that merely using ASIL decomposition to assure the reliability requirement is insufficient, and using the reliability enhancement technique can improve the opportunity of assuring the reliability requirement. For example, the reliability is enhanced from 0.6001 (using RTRV) to 0.9539 (using FSRV) when the response time requirement is 6269 μs. On the basis of the reliability enhancement technique, FSEO can optimize energy consumption while assuring its functional safety requirement.
3) Table XV shows that FSEO generates the minimum energy consumption and outperforms EFSRG, FSRV, and RTRV by 61.5%, 85.6%, and 86.3%, respectively, according to the effective results. The results further indicate that FSEO is an energy-efficient algorithm for ACPS functional safety.

TABLE XVI
RESPONSE TIME VALUES (UNIT: μS) OF SYNTHETIC ACPS FUNCTIONS UNDER VARYING NUMBERS OF TASKS

$\|N\|$ =x	x=50	x=60	x=70	x=80	x=90	x= 100
RTRV	9,944	12,001	13,718	15,511	14,667	15,259
FSRV	16,000	16,000	16,000	16,000	16,000	16,000
FSEO	16,000	16,000	16,000	16,000	-	-
EFSRG	22,759	27,999	30,884	35,051	35,073	39,574
EFSRG_ASIL	-	-	-	-	-	-

TABLE XVII
RELIABILITY VALUES OF SYNTHETIC ACPS FUNCTIONS UNDER VARYING NUMBERS OF TASKS

$\|N\|$=x	x=50	x=60	x=70	x=80	x=90	x= 100
RTRV	0.5268	0.6123	0.5740	0.5609	0.4903	0.4341
FSRV	0.9807	0.9773	0.9731	0.9733	0.7619	0.7116
FSEO	0.9501	0.9501	0.9501	0.9501	-	-
EFSRG	0.9501	0.9501	0.9501	0.9501	0.9501	0.9501
EFSRG_ASIL	-	-	-	-	-	-

TABLE XVIII
ENERGY CONSUMPTIONS (UNIT: WμS) OF SYNTHETIC ACPS FUNCTIONS UNDER VARYING NUMBERS OF TASKS

$\|N\|$=x	x=50	x=60	x=70	x=80	x=90	x= 100
RTRV	294,186	343,050	409,532	493,353	525,117	595,095
FSRV	268,749	328,927	39,0850	450,470	515,027	570,584
FSEO	36,183	41,182	52,084	55,064	-	-
EFSRG	86,372	99,184	124,362	137,545	160,468	176,642
EFSRG_ASIL	-	-	-	-	-	-

B. Synthetic Functions

Experiment 3: We employ the task graph generator download from [73] to generate random ACPS functions. The parameters of synthetic ACPS functions are similar to those of real-life ACPS functions. Three important parameters affect the structures of randomly generated ACPS functions.

1) The communication-to-computation ratio (CCR) is chosen from $\{0.1, 0.5, 1.0, 2.0, 5.0\}$; the smaller the CCR is, the lower the communication time is, and vice versa. We set it to 1 in the experiment.
2) The shape parameter refers to the density of the graph and belongs to the range of [0.35, $\sqrt{|N|/3}$]. We set it to 1 in the experiment.
3) The heterogeneity factor values belong to the scope of 0 to 1 in the task graph generator, where 0 and 1 represent the lowest and highest heterogeneity factors, respectively. For fairness, we set the heterogeneity factor to 0.5 in the experiment.

The task numbers of the ACPS functions are changed from 50 to 100 with ten increments. The response time and reliability requirements of all ACPS functions are fixed at 16 000 μs and 0.95, respectively. The results are shown in Tables XVI–XVIII.

1) The results in Tables XVI–XVIII show that when the task number reach 90, the functional safety requirement is no longer assured for the proposed algorithms. Even when the reliability enhancement technique is employed with FSRV,

the actual reliability values are merely 0.7619 and 0.7116 when the task numbers are 90 and 100, respectively.

2) The results in Tables XVI–XVIII further show that EFSRG_ASIL does not obtain a valid result in all the cases. Therefore, EFSRG_ASIL is not a feasible approach toward energy-efficient functional safety for ACPS.
3) As expected, the energy consumptions in Table XVIII show that FSEO still generates the minimum energy consumptions in all the valid cases. FSEO outperforms FSRV by 86.5–87.8% in energy consumption optimization. In other words, the advantage of FSEO is significant toward energy-efficient functional safety for ACPS. FSEO outperforms EFSRG by 58.1–60.0% in energy consumption optimization, and EFSRG cannot assure its response time requirement. Therefore, only FSEO is useful toward energy-efficient functional safety for ACPS among the five algorithms.

VIII. Conclusion

In this paper, we solved the problem of energy-efficient functional safety design for ACPS by proposing a three-stage design process. The first stage was to verify the response time requirement of the ACPS function by using the proposed RTRV algorithm. The second stage was to verify the reliability requirement of the ACPS function while assuring its response time requirement by using the proposed FSRV algorithm. The third stage was to optimize energy consumption while assuring functional safety requirement by using the proposed FSEO algorithm.

The biggest contribution of this paper is that different stages of our three-stage design process can solve the different performance optimization problems with different requirements; moreover, the proposed algorithms in different stages follow the ISO 26262 standard and use different ASIL decomposition schemes: 1) RTRV chooses the ASIL decomposition scheme that has the minimum EFT; 2) FSRV chooses the ASIL decomposition scheme with the maximum reliability among five schemes without violating its response time requirement; and 3) FSEO chooses the ASIL decomposition scheme that has the minimum energy consumption among five schemes without violating its response time and reliability requirements.

Experiments showed that the FSEO algorithm achieved less energy consumptions than the FSRV algorithm and the state-of-the-art EFSRG algorithm. Our three-stage design process not only can optimize energy consumption with a better advantage, but also can shorten the ACPS function's development life cycle. Our energy-efficient functional safety design methodology could be useful for the cost-sensitive, power-sensitive, and environment-friendly auto industry. As cyber part dynamically interacts with the physical part, and dynamics is the inherent property of CPS, the future work could study the energy-efficient functional safety design for dynamic ACPS.

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] NSF, "Cyber-physical systems (CPS). Program solicitation NSF 18-538," 2018. [Online]. Available: https://www.nsf.gov/pubs/2018/nsf18538/nsf18538.htm
[2] G. Xie, G. Zeng, J. Jiang, C. Fan, R. Li, and K. Li, "Energy management for multiple real-time workflows on cyber–physical cloud systems," *Future Gener. Comput. Syst.*, May 2017, doi: 10.1016/j.future.2017.05.033.
[3] L. Liu, G. Xie, and R. Li, "Synchronization stability analysis of medical cyber-physical cloud system considering multi-closed-loops," *J. Circuits, Syst., Comput.*, Dec. 2018, doi: 10.1142/S0218126619501986.
[4] G. Xie, G. Zeng, Z. Li, R. Li, and K. Li, "Adaptive dynamic scheduling on multi-functional mixed-criticality automotive cyber-physical systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6676–6692, Aug. 2017.
[5] C. Schmittner, Z. Ma, E. Schoitsch, and T. Gruber, "A case study of FMVEA and chassis as safety and security co-analysis method for automotive cyber-physical systems," in *Proc. 1st ACM Workshop Cyber-Phys. Syst. Security*, 2015, pp. 69–80.
[6] G. Macher, E. Armengaud, D. Schneider, E. Brenner, and C. Kreiner, "Towards dependability engineering of cooperative automotive cyber-physical systems," in *Proc. Eur. Conf. Softw. Process Improvement* 2017, pp. 205–215.
[7] G. Xie *et al.*, "Reliability enhancement towards functional safety goal assurance in energy-aware automotive cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 12, pp. 5447–5462, Dec. 2018.
[8] H. Zeng, M. Di Natale, P. Giusto, and A. Sangiovanni-Vincentelli, "Stochastic analysis of can-based real-time automotive systems," *IEEE Trans. Ind. Informat.*, vol. 5, no. 4, pp. 388–401, Sep. 2009.
[9] A. Munir, "Safety assessment and design of dependable cybercars: For today and the future," *IEEE Consum. Electron. Mag.*, vol. 6, no. 2, pp. 69–77, Apr. 2017.
[10] ISO, *Road Vehicles-Functional Safety*, ISO 26262-1: 2011, 2011.
[11] D. Gizopoulos, "Online periodic self-test scheduling for real-time processor-based systems dependability enhancement," *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 2, pp. 152–158, Apr. 2009.
[12] 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. 2009.
[13] 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. Informat.*, vol. 13, no. 4, pp. 1629–1640, Aug. 2017.
[14] G. Xie *et al.*, "Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems," *IEEE Trans. Serv. Comput.*, to be published, doi: 10.1109/TSC.2017.2665552.
[15] 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. Informat.*, vol. 14, no. 6, pp. 2418–2431, Jun. 2018.
[16] G. Xie, W. Ma, H. Peng, R. Li, and K. Li, "Price performance-driven hardware cost optimization under functional safety requirement in large-scale heterogeneous distributed embedded systems," *IEEE Trans. Ind. Electron.*, to be published, doi: 10.1109/TIE.2019.2905815.
[17] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 3, pp. 318–326, Mar. 2014.
[18] S. J. Moura, H. K. Fathy, D. S. Callaway, and J. L. Stein, "A stochastic optimal control approach for power management in plug-in hybrid electric vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 545–555, May 2011.
[19] P. Dziurzanski, A. K. Singh, and L. S. Indrusiak, "Energy-aware resource allocation in multi-mode automotive applications with hard real-time constraints," in *Proc. IEEE 19th Int. Symp. Real-Time Distrib. Comput.*, 2016, pp. 100–107.
[20] K. Nii *et al.*, "A dynamic/static SRAM power management scheme for DVFS and AVS in advanced automotive infotainment SOCs," in *Proc. IEEE Symp. VLSI Technol.*, 2016, pp. 1–2.
[21] Z. Chen, C. C. Mi, J. Xu, X. Gong, and C. You, "Energy management for a power-split plug-in hybrid electric vehicle based on dynamic programming and neural networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 4, pp. 1567–1580, May 2014.
[22] B. Hredzak, V. G. Agelidis, and M. Jang, "A model predictive control system for a hybrid battery-ultracapacitor power source," *IEEE Trans. Power Electron.*, vol. 29, no. 3, pp. 1469–1479, Mar. 2014.

[23] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2004, pp. 35–40.

[24] D. Zhu and H. Aydin, "Energy management for real-time embedded systems with reliability requirements," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2006, pp. 528–534.

[25] 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.

[26] B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, pp. 99–109, Mar. 2013.

[27] G. Xie, Y. Chen, X. Xiao, Y. Chen, R. Li, and K. Li, "Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 3, pp. 167–181, Jul./Sep. 2018.

[28] P. Heinrich and C. Prehofer, "Network-wide energy optimization for adaptive embedded systems," *ACM SIGBED Rev.*, vol. 10, no. 1, pp. 33–36, 2013.

[29] V. Vyatkin, "Software engineering in industrial automation: State-of-the-art review," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1234–1249, Aug. 2013.

[30] H. F. Sheikh and I. Ahmad, "Sixteen heuristics for joint optimization of performance, energy, and temperature in allocating tasks to multi-cores," *ACM Trans. Parallel Comput.*, vol. 3, no. 2, Aug. 2016, Art. no. 9.

[31] I. Assayad, A. Girault, and H. Kalla, "Tradeoff exploration between reliability, power consumption, and execution time," in *Proc. Int. Conf. Comput. Safety, Rel., Sec.*, 2011, pp. 437–451.

[32] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, "A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 3, pp. 401–412, Mar. 2016.

[33] P. Sun, Y. Dai, and X. Qiu, "Optimal scheduling and management on correlating reliability, performance, and energy consumption for multiagent cloud systems," *IEEE Trans. Rel.*, vol. 66, no. 2, pp. 547–558, Jun. 2017.

[34] L. Zhao, Y. Ren, Y. Xiang, and K. Sakurai, "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems," in *Proc. 12th IEEE Int. Conf. High Perform. Comput. Commun.*, 2010, pp. 434–441.

[35] L. Zhao, Y. Ren, and K. Sakurai, "Reliable workflow scheduling with less resource redundancy," *Parallel Comput.*, vol. 39, no. 10, pp. 567–585, Jul. 2013.

[36] G. Xie, G. Zeng, R. Li, and K. Li, "Quantitative fault-tolerance for reliable workflows on heterogeneous IaaS clouds," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2017.2780098.

[37] M. Hakem and F. Butelle, "A bi-objective algorithm for scheduling parallel applications on heterogeneous systems subject to failures," in *Proc. 17th emes Rencontres Francophones du Paralllisme*, 2006, pp. 25–35.

[38] A. Dogan and F. Ozguner, "Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 308–323, Mar. 2002.

[39] G. Xie, G. Zeng, Y. Liu, J. Zhou, R. Li, and K. Li, "Fast functional safety verification for distributed automotive applications during early design phase," *IEEE Trans. Ind. Electron.*, vol. 65, no. 5, pp. 4378–4391, May 2018.

[40] Q. Huang, S. Su, J. Li, P. Xu, K. Shuang, and X. Huang, "Enhanced energy-efficient scheduling for parallel applications in cloud," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2012, pp. 781–786.

[41] G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1067–1078, Jun. 2017.

[42] G. Xie, G. Zeng, X. Xiao, R. Li, and K. Li, "Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3426–3442, Dec. 2017.

[43] 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," *J. Grid Comput.*, vol. 14, no. 1, pp. 55–74, 2016.

[44] G. Xie, G. Zeng, R. Li, and K. Li, "Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 62–75, Jun. 2017.

[45] J. Li, G. Xie, K. Li, and Z. Tang, "Enhanced parallel application scheduling algorithm with energy consumption constraint in heterogeneous distributed systems," *J. Circuits, Syst., Comput.*, Dec. 2018, doi: 10.1142/S0218126619501901.

[46] G. Xie, Y. Chen, Y. Liu, R. Li, and K. Li, "Minimizing development cost with reliability goal for automotive functional safety during design phase," *IEEE Trans. Rel.*, vol. 67, no. 1, pp. 196–211, Mar. 2018.

[47] P. Baybutt, "A critique of the hazard and operability (hazop) study," *J. Loss Prevention Process Ind.*, vol. 33, pp. 52–58, Jan. 2015.

[48] H.-C. Liu, J.-X. You, X.-Y. You, and M.-M. Shan, "A novel approach for failure mode and effects analysis using combination weighting and fuzzy vikor method," *Appl. Soft Comput.*, vol. 28, pp. 579–588, Mar. 2015.

[49] P. Liu, L. Yang, Z. Gao, S. Li, and Y. Gao, "Fault tree analysis combined with quantitative analysis for high-speed railway accidents," *Safety Sci.*, vol. 79, pp. 344–357, Nov. 2015.

[50] J. D. Andrews and S. J. Dunnett, "Event-tree analysis using binary decision diagrams," *IEEE Trans. Rel.*, vol. 49, no. 2, pp. 230–238, Jun. 2000.

[51] T. Kelly and R. Weaver, "The goal structuring notation—A safety argument notation," in *Proc. Depend. Syst. Netw. Workshop Assurance Cases*, 2004, pp. 1–6.

[52] H. S. Mahajan, T. Bradley, and S. Pasricha, "Application of systems theoretic process analysis to a lane keeping assist system," *Rel. Eng. Syst. Safety*, vol. 167, pp. 177–183, 2017.

[53] Q. Zheng, B. Veeravalli, and C.-K. Tham, "On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs," *IEEE Trans. Comput.*, vol. 58, no. 3, pp. 380–393, Mar. 2009.

[54] E. Rolf, "Formal performance analysis in automotive systems design—A rocky ride to new grounds," in *Proc. 23rd Int. Conf. Comput. Aided Verification*, 2011, pp. 1–31. [Online]. Available: http://101.110.118.63/formal verification.cs.utah.edu/cav2011/content/presentations/CAV2011V3.pdf

[55] G. Bernat, A. Colin, and S. M. Petters, "Wcet analysis of probabilistic hard real-time systems," in *Proc. 23rd IEEE Real-Time Syst. Symp.*, 2002, pp. 279–288.

[56] A. D. Burns and R. Davis, "Mixed-criticality systems: A review (eighth edition)," 2016. [Online]. Available: http://www-users.cs.york.ac.uk/burns/review.pdf

[57] Q. Zhao, Z. Gu, and H. Zeng, "Resource synchronization and preemption thresholds within mixed-criticality scheduling," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 4, p. 81, Dec. 2015.

[58] Q. Zhao, Z. Gu, and H. Zeng, "Design optimization for autosar models with preemption thresholds and mixed-criticality scheduling," *J. Syst. Archit.*, vol. 72, pp. 61–68, Jan. 2017.

[59] 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, Nov. 2017.

[60] G. Xie, G. Zeng, R. Kurachi, H. Takada, R. Li, and K. Li, "Exact WCRT analysis of message-processing tasks on gateway-integrated in-vehicle can clusters," *ACM Trans. Embedded Comput. Syst.*, vol. 17, no. 6, Jan. 2019, Art. no. 95.

[61] G. Xie *et al.*, "WCRT analysis and evaluation for sporadic message-processing tasks in multicore automotive gateways," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 2, pp. 281–294, Feb. 2019.

[62] M. Zeller, C. Prehofer, G. Weiss, D. Eilers, and R. Knorr, "Towards self-adaptation in real-time, networked systems: Efficient solving of system constraints for automotive embedded systems," in *Proc. 5th IEEE Int. Conf. Self-Adaptive Self-Organizing Syst.*, 2011, pp. 79–88.

[63] J. D. Ullman, "Np-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, Jun. 1975.

[64] 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.

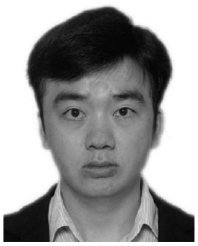
[65] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.

[66] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1374–1381, Aug. 2011.

[67] A. Benoit, L.-C. Canon, E. Jeannot, and Y. Robert, "Reliability of task graph schedules with transient and fail-stop failures: complexity and algorithms," *J. Scheduling*, vol. 15, no. 5, pp. 615–627, Oct. 2012.

[68] A. Girault, E. Saule, and D. Trystram, "Reliability versus performance for critical applications," *J. Parallel Distrib. Comput.*, vol. 69, no. 3, pp. 326–336, Mar. 2009.

[69] J. Zhou *et al.*, "Resource management for improving soft-error and lifetime reliability of real-time MPSoCs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, to be published, doi: 10.1109/TCAD.2018.2883993.
[70] Y. Ma, J. Zhou, T. Chantem, R. P. Dick, S. Wang, and X. S. Hu, "On-line resource management for improving reliability of real-time systems on big–little type MPSoCs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, to be published, doi: 10.1109/TCAD.2018.2883990.
[71] 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.
[72] D. Zhu and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1382–1397, Oct. 2009.
[73] Aug. 2015. [Online]. Available: https://sourceforge.net/projects/taskgraphgen/



Guoqi Xie (M'15–SM'19) received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2014.

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

Dr. Xie received the Best Paper Award at IEEE ISPA, in 2016 and IEEE TCSC Award for Excellence (Early Career Researcher), in 2018. He is currently serving on the editorial boards of *Journal of Systems Architecture*, *Journal of Circuits, Systems and Computers*, *Microprocessors and Microsystems*, and IEEE ACCESS. He is an ACM Senior Member.

Hao Peng is currently working toward the M.S. degree in computer science and engineering with Hunan University, Changsha, China.

His current research interests include automotive cyber-physical systems and automotive functional safety.

Jing Huang received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2018.

His research interests include energy-efficient computing, cyber-physical systems, and machine learning.

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 with the Department of Computer Engineering, College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. He is the Director of the Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha, China. His major interests include computer architectures, embedded computing systems, cyber-physical systems, and Internet of Things.

Mr. Li is a Member of the council of CCF and a Senior Member of ACM.

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 SUNY Distinguished Professor of Computer Science with the State University of New York, New York, NY, USA. He is also a Distinguished Professor with Hunan University, Changsha, China. He has published more than 640 journal articles, book chapters, and refereed conference papers. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, and intelligent and soft computing.

Mr. Li is currently or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. He has received several Best Paper Awards.