Contention and Reliability-Aware Energy Efficiency Task Mapping on NoC-Based MPSoCs

Lei Mo, Xinmei Li, Angeliki Kritikakou, Xiaojun Zhai

Contention and Reliability-Aware Energy Efficiency Task Mapping on NoC-Based MPSoCs

Lei Mo*, *Member, IEEE,* Xinmei Li*, *Student Member, IEEE,* Angeliki Kritikakou†, *Member, IEEE,* and Xiaojun Zhai‡, *Senior Member, IEEE*

***Abstract*—Recently, Network-on-Chip (NoC)-based Multi-Processor System-on-Chips (MPSoCs) have become popular computing platforms for real-time applications due to high communication performance and energy efficiency over traditional bus-based MPSoCs. Due to the nature of network structures, network congestion along with transient faults, can significantly affect communication efficiency and system reliability. Most existing works have rarely focused on the concurrent optimization of network contention, reliability, and energy consumption. Here, we study the problem of contention and reliability-aware task mapping under real-time constraints for dynamic voltage and frequency scaling-enabled NoC. The problem entails optimizing voltage/frequency on cores and links to reduce energy consumption and ensure system reliability, while task mapping and slack time are adopted to alleviate network contention and reduce latency. We aim to minimize computation and communication energy and balance workload. This problem is formulated as a mixed-integer nonlinear programming, and we present an effective linearization scheme that equivalently transforms it into a mixed-integer linear programming to find the optimal solution. To reduce computation time, we propose a three-step heuristic, including task allocation, frequency scaling and edge scheduling, and communication contention management. Finally, we perform extensive simulations to evaluate the proposed method. The results show we can achieve 31.6% and 21.7% energy savings, with 95.5% and 98.6% less contention than the existing methods.**

***Index Terms*—Contention-aware, Reliability-aware, Energy Management, Task mapping, NoC.**

I. Introduction

With the increasing demand for high computing performances, e.g., Internet of Things (IoT) [1], big data processes [2], cloud centers [3] and blockchain systems [4], Multi-Processor System-on-Chip (MPSoC) architectures integrate multiple cores into a single platform. The increase in cores leads to many issues and challenges for traditional shared bus-based multicore platforms, such as poor scalability and low communication efficiency due to the limited bus bandwidth [5]. To address the scalability and bandwidth issues, Network-on-Chip (NoC) has been proposed as an interconnection architecture solution with much better scalability, flexibility, parallelism, and communication efficiency, compared to the traditional bus-based multicore platforms [6]. In the NoC-based MPSoCs, e.g., the Ethereal NoC [7], the Versal NoC [8], and the Arteris NoC [9], the Operating System (OS) can run at each core, responsible for task scheduling, task loading, packet assembly/disassembly, and Direct Memory Access (DMA) management, enabling the platforms to meet multiple constraints.

Furthermore, energy consumption has become an essential issue for the NoC-based MPSoCs as the number of cores increases. To achieve an effective system, appropriate task mapping methods are needed. Most energy-aware task mapping methods in bus-based MPSoCs mainly focus on the energy consumption of computation, e.g., [10]–[12]. To reduce both computation and communication energy consumption, the NoC-based MPSoCs are enhanced with Dynamic Voltage and Frequency Scheduling (DVFS) for cores, routers, and links [13]–[16]. The energy consumption can be reduced by lowering the Voltage/Frequency (V/F) level. However, the decrease in operating V/F levels impacts the transient fault rate on cores, routers, and links, thereby decreasing the system reliability [17]. Therefore, the trade-off between energy consumption and system reliability should be considered during task mapping on NoC-based MPSoCs. Some existing works [18]–[21] have studied the joint design problem of saving energy and enhancing system reliability. For instance, transient faults are considered on links and routers [18], [19] or cores [20], [21]. As transient faults may occur during task execution and communication, it is necessary to consider that they can occur on cores, routers, and links when optimizing task mapping for energy consumption and system reliability on NoC-based platforms.

Due to the nature of the network-based structure, the task mapping strategy on NoC-based platforms may cause network contention among the task data communication over the shared communication resource, i.e., routers and links. If a link is occupied due to the transmission of a message, any other messages requiring the same link will wait until the link becomes available [6], [14]. Network congestion degrades system performance due to the increased transmission delay, thereby affecting the execution time of applications, which is of paramount importance, especially for real-time applications [6]. For the NoC-based platforms, it is necessary to consider communication contention during task mapping. However, most existing works focused on energy and (or) reliability improvement but ignored the influence of communication contention [18], [21]. Other works alleviated the network congestion by introducing fixed slack time into the task scheduling [13]. Note that load imbalance may cause overloading and local network congestion on NoC. Overloading decreases power efficiency and system reliability [22]–[24] since the cores with high computation loads

*L. Mo and X. Li are with the School of Automation, Southeast University, 210096 Nanjing, China. E-mails: lmo@seu.edu.cn, xinmeili@seu.edu.cn.

†A. Kritikakou is with the University of Rennes, INRIA, IRISA, CNRS, 35042 Rennes, France, and also with the Institut Universitaire de France, 75005 Paris, France. E-mail: angeliki.kritikakou@irisa.fr.

‡X. Zhai is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K., E-mail: xzhai@essex.ac.uk.

stay active most of the time while the remaining cores are idle. To obtain an efficient system, we must simultaneously alleviate network contention, balance workload, and improve energy efficiency and reliability.

To address the above issues, we propose a novel task mapping method to jointly optimize energy efficiency through DVFS while balancing the workload and mitigating the communication contention on the NoC-based platform under real-time and reliability constraints. Therefore, our method is suitable for applications with energy efficiency, reliability, and timeliness requirements, such as autonomous drones [25], and video processing systems [26]. The main contributions of this paper are summarized as follows:

(1) We holistically model the problem of reducing energy consumption and balancing the workload for NoC-based platforms. DVFS adjusts the V/F levels of cores and links, reducing the total energy consumption under real-time and reliability constraints. Communication contention is considered, where flexible slack time alleviates communication contention, reducing latency.
(2) Based on the system model, the task mapping problem is formulated as a mixed-integer nonlinear programming (MINLP) problem, with the aim to minimize energy consumption and balance workload under multiple constraints. We apply the linearization method by introducing auxiliary variables and additional constraints and then equivalently transform the original problem into a mixed-integer linear programming (MILP) problem to find the optimal solution.
(3) Taking advantage of the structure of the task mapping problem, we propose a novel three-step heuristic to find the solutions. First, a dynamic programming-based task allocation heuristic is applied to find the task-to-core decision. Then, the improved feasibility-pump heuristic is used to determine the frequency assignment and the edge scheduling. Finally, we calculate the precise contention slack time and adjust frequencies on cores within the task deadlines to reduce energy.
(4) We perform simulations on randomly generated task sets and realistic application task sets to evaluate the performance of our method in terms of energy consumption, task reliability, and communication contention. Compared to the state-of-the-art methods [6] and [21], our method achieves 31.6% and 21.7% energy reduction and has 95.5% and 98.6% lower average contention, while balancing workload and satisfying reliability constraints. In addition, compared to the optimal solution, our method has a polynomial computation time, but with only 24.8% lower energy efficiency.

The remainder of this paper is organized as follows. Section II discusses the related work. Section III presents a contention- and reliability-aware task mapping problem. Section IV describes the proposed three-step heuristic. Section V shows the simulation results, and Section VI concludes this paper.

II. Related Work

To improve system efficiency, such as achieving energy and communication efficiency and reliability, task mapping on NoC-based MPSoCs has gained significant attention in recent years.

TABLE I
Comparison of Related Works

Ref.	Optimization objectives	RA	CA	LB
[27]	Energy	×	×	×
[16]	Energy	×	×	×
[21]	Energy	√	×	×
[28]	Task Replication Overhead	√	×	×
[20]	Energy	√	×	×
[29]	Energy + Temperature	√	×	×
[6]	Latency	×	√	×
[30]	Latency	×	√	×
[31]	Makespan	×	√	×
[13]	Makespan	×	√	×
[32]	Energy + Latency	×	√	×
[15]	Energy + Latency	×	√	×
[14]	Energy	×	√	×
[18]	Energy + Reliability + Latency	√	×	×
[19]	Energy + Reliability + Latency	√	√	×
[22]	Communication Efficiency + Workload	×	×	√
[33]	Energy + Execution Time + Workload	√	×	√
[34]	Energy + Makespan + Contention	×	√	√
Pro.	Minimize Energy + Balance Workload	√	√	√

Table I summarizes some representative works of task mapping on NoC-based MPSoCs, which include Reliability Awareness (RA), Contention Awareness (CA), and Load Balance (LB).

Energy efficiency is essential in NoC-based MPSoCs, especially for battery-powered mobile embedded systems [16]. Several works have studied energy-aware task scheduling on NoC-based MPSoCs [16], [27] to minimize the total energy of computation and communication under real-time constraints. However, using low operating frequency to execute tasks may impact task reliability, increasing the appearance of transient faults. To address this issue, some approaches introduce reliability constraints into energy-aware task mapping problem [20], [21]. For transient faults occurring on cores, a fixed fault-tolerant policy is applied [21], where tasks are duplicated, and frequencies are adjusted simultaneously. To mitigate the transient faults while improving real-time response, a fault-tolerant method selects from several policies, i.e., checkpointing, task replication, and checkpointing with task replication techniques [20]. A fault-tolerant mechanism based on the backup copy that simultaneously optimizes temperature and energy consumption is proposed in [29]. The primary task and backup copy are executed on the primary and backup cores, respectively, and the backup cores are used only when failures exist in the primary cores. However, task replication introduces replicas, which increases computation and communication costs. To minimize task replicas and latency, [28] jointly addresses transient faults of cores, links, and routers on NoCs.

Communication latency is another critical issue, as it affects the execution time of mapped applications, especially for real-time applications [6]. To minimize the communication latency, approaches focus on the link contention between data transfers over NoC [6], [30], [31]. In [6], [30], contention metrics are introduced to identify the links of NoC with high loads, which are prone to suffer congestion. The latency can be minimized by bypassing the congested links. The communications with potential contentions are prioritized, and priorities are used to reduce the makespan [31]. However, the above methods only consider optimizing communication efficiency, whereas energy

efficiency and reliability are not considered.

Considering network contention, several works [13]–[15], [32], [34] jointly optimize energy and latency. Few works also take into account reliability [18], [19]. Considering energy- and contention-aware task mapping on NoCs, the communication latency is optimized in [13], [32] by adding a fixed latency time into task scheduling, where the fixed latency time is calculated by the worst congestion estimation to alleviate the impact caused by link congestion. Energy consumption can be further reduced through task mapping and DVFS. However, introducing a fixed latency time may postpone task completion time. Therefore, optimizing task assignment and operation frequency can help in meeting task deadlines. Communications with contentions are prioritized to minimize energy consumption [14], [15], [31]. Compared to [14] that only considers DVFS, [15] integrates DVFS and DPM (Dynamic Power Management) techniques together to reduce further energy consumption. The optimization of the communication energy, reliability, and latency on NoCs is proposed in [18] by assigning the tasks onto a redundant core to avoid faulty links. Remapping and wait mechanisms are applied to solve the faults on cores, links, and routers while introducing fixed waiting time due to congestion into the scheduling [19]. However, the methods mentioned above only consider communication energy, more overhead is introduced for task remapping, they require redundant NoC hardware, and they omit the workload balance.

Regarding workload balance, it is optimized to improve the system [22], [33], [34]. For instance, a trade-off task mapping method balances workload and communication efficiency to reduce task execution time in [22]. Considering faults on routers, task reliability is tackled simultaneously considering energy consumption, execution time, and load balancing [33], but without network contention. An optimization method considering energy, makespan, and contention is proposed to avoid spatial and temporal contentions in [34]. However, the frequency adjustment and real-time constraints are not taken into account. Compared to the state-of-the-art methods in Table I, our work minimizes energy consumption and balances workload under real-time and reliability constraints while alleviating the network contention on the NoC-based platform.

III. System Model and Problem Formulation

A. System Model

1) NoC Model: The target platform is a 2D mesh NoC-based homogeneous MPSoCs, as it is widely used in current NoC architecture [13], [14], [27]. The platform has a set $\mathcal{P} \triangleq \{P_1, \ldots, P_M\}$ of M cores, and a set $\mathcal{R} \triangleq \{R_1, \ldots, R_M\}$ of M corresponding routers. For a 2D mesh NoC topology, as shown in Fig. 1(a), each router R_s has five I/O ports, where four ports are linked to the neighboring routers, while one is associated with the corresponding core P_s [13]. In addition, each port has a link and a buffer to transmit and store the data, respectively. All the cores and links in the NoC platform are DVFS-enabled. Each core can operate on a set $\{(v_{p,1}, f_{p,1}), \ldots, (v_{p,N_p}, f_{p,N_p})\}$ of N_p discrete V/F levels, where $(v_{p,k}, f_{p,k})$ denotes the k^{th} V/F level of the core $(1 \leq k \leq N_p)$. Similarly, each link can operate at a set $\{(v_{l,1}, f_{l,1}), \ldots, (v_{l,N_l}, f_{l,N_l})\}$ of N_l discrete V/F levels, and $(v_{l,g}, f_{l,g})$ denotes the g^{th} V/F level of the link $(1 \leq g \leq N_l)$. We adopt the deterministic XY routing, a popular routing method on 2D mesh NoC [13], [14], [27]. The routing set of physical links between cores P_s and P_d is $\mathcal{L}_{s,d} \triangleq \{l_{s,i}, \ldots, l_{j,d}\}$, where $l_{s,i}$ (or $l_{j,d}$) denotes the physical link between cores P_s and P_i (or cores P_j and P_d), and each link $l_{s,d}$ is full duplex with a bandwidth b_ω. The distance between P_s and P_d represents the number of links in the set $\mathcal{L}_{s,d}$, and it can be measured by the Manhattan distance $h_{s,d} = |x_s - x_d| + |y_s - y_d|$, where (x_s, y_s) and (x_d, y_d) are the coordinates of cores P_s and P_d in a mesh-based NoC, respectively, as the example shown in Fig. 1(a).

2) Task Model: A real-time application with N dependent tasks is modeled as a Directed Acyclic Graph (DAG) with 6-tuple elements $\{\mathcal{T}, \mathcal{E}, \mathcal{S}, \mathcal{W}, \mathcal{D}, H\}$. $\mathcal{T} \triangleq \{\tau_1, \ldots, \tau_N\}$ is a real-time task set with N tasks. $\mathcal{E}$ denotes the set of edges, where each edge $e_{i,j}$ represents the dependency between tasks τ_i and τ_j. Each edge has a weight $s_{i,j} \in \mathcal{S}$, representing the size of communication data between τ_i and τ_j. Each task τ_i has $\omega_i \in \mathcal{W}$ cycles, measured by Worst-Case Execution Cycles (WCEC). Task τ_i is released at time 0 and has an individual deadline $D_i \in \mathcal{D}$. H is the scheduling horizon, which is also the period of task τ_i. Due to the dependency, each task τ_i cannot start execution until all its predecessors' input data has arrived. At the same time, the output data is concurrently available for transmission to all its successors only when it completes execution.

We consider the transmission time over links for task data communication since the extra overheads (e.g., data copy between buffers and inter-router delay) are negligible compared with the transmission time [13]. Besides, we consider that the transmission time is increased with the routing distance. For the edge $e_{i,j}$, if the data with the size $s_{i,j}$ is transmitted from τ_i to τ_j by one unit link, the link transmission time with the V/F level $(v_{l,g}, f_{l,g})$ is calculated as

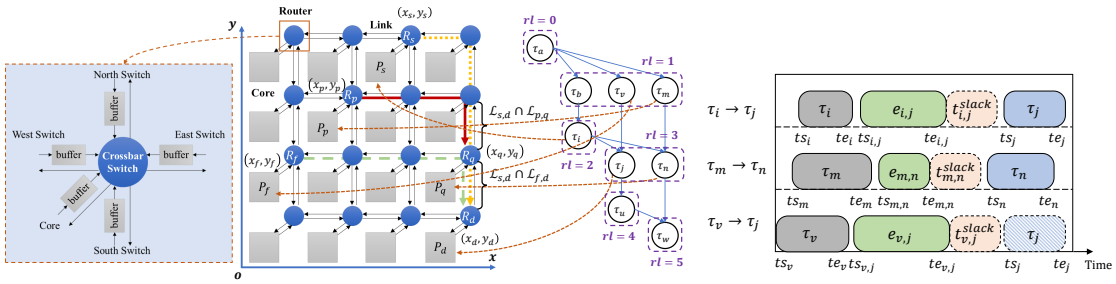
$$t^{UL}_{i,j,g} = s_{i,j}/(b_\omega f_{l,g}),\ \forall e_{i,j} \in \mathcal{E},\ 1 \leq g \leq N_l. \tag{1}$$

If the routing distance is more than one unit, e.g., tasks τ_i and τ_j are assigned to different cores P_s and P_d, and the routing distance associated with the edge $e_{i,j}$ is $h_{s,d}$, the transmission time with the V/F level $(v_{l,g}, f_{l,g})$ is computed as $t^{L}_{i,j,g} = h_{s,d} t^{UL}_{i,j,g}$. If two tasks are assigned to the same core, the communication cost between them is assumed to be zero.

3) Energy Model: Since each core and link can operate on different V/F levels, switching brings extra overhead to energy consumption and execution time. Compared with the energy and time required for task execution and data communication, the switching overhead is very small, especially when the switching rate is low. Hence, we omit the overhead of frequency switching [14], [16], [27]. We consider the power dissipation by computation and communication in the power model. Given the V/F level $(v_{p,k}, f_{p,k})$, the computation power P^{comp}_k dissipated in executing tasks is computed as follows [27]:

$$P^{comp}_k = \underbrace{C_{eff} v^2_{p,k} f_{p,k}}_{\text{dynamic power}} + \underbrace{L_g\,(v_{p,k} I_{sub} + |v_{bs}| I_j)}_{\text{static power}}, \tag{2}$$

where C_{eff}, L_g, I_{sub}, v_{bs}, and I_j denotes the effective switching capacitance, the number of devices in the circuit, the subthreshold leakage current, the body bias voltage, and the reverse



(a) Task mapping of Gauss elimination on a 4×4 NoC with XY routing. (b) Task execution and communication time.

Fig. 1. An example of task mapping with communication contention.

bias junction current, respectively. Besides the V/F level, the operating temperature also influences the static power. In this paper, we did not consider temperature, as it will introduce another dimension of variables into the task mapping problem, making it more difficult to solve. Instead, we mainly focus on optimizing energy efficiency, reliability, and communication contention, similar to the existing works [16], [20], [21].

The following link power model is adopted from [27]:

$$P_g^{link} = s_a C_{rep} v_{l,g}^2 f_{l,g} + L_g \left(v_{l,g} K_3 e^{K_4 v_{l,g}} e^{K_5 v_{bs}} + |v_{bs}| I_j \right), \tag{3}$$

where s_a is the average switching activity, C_{rep} is capacity load of a repeater, and K_3, K_4 and K_5 are hardware parameters.

Note that the routers and the links on NoC mainly consume communication energy. The communication energy for edge $e_{i,j}$ with the data size $s_{i,j}$ from core P_s to P_d, if it is operated at the V/F level $(v_{l,g}, f_{l,g})$ [14], [27], is given by:

$$E_g^{comm} = s_{i,j} \left[(h_{s,d} + 1) E_{Rbit} + h_{s,d} E_{Lbit,g} \right], \tag{4}$$

where E_{Rbit} is the energy consumption of transmitting a bit data on a router, and $E_{Lbit,g} = P_g^{link} / b_\omega f_{l,g}$ is the energy consumed by transmitting a bit data over a link with the frequency $f_{l,g}$. Since $h_{s,d}$ is the number of the links that a bit data passes during the communication, $h_{s,d} + 1$ is the number of routers that a bit data passes. On this basis, $(h_{s,d} + 1) E_{Rbit}$ is the energy consumption of transmitting a bit data for routes, while $h_{s,d} E_{Lbit,g}$ is the energy consumption of transmitting a bit data with the V/F $(v_{l,g}, f_{l,g})$ on links.

4) Communication Contention Model: In NoC, any physical link can only transmit one message at the same time; otherwise, communication contention occurs, leading to data transmission blocking and system performance degradation. The existence of communication contention should satisfy the time and space constraints. For instance, we assume that tasks τ_i, τ_j, τ_m, τ_n, and τ_v are assigned to cores P_s, P_d, P_p, P_q, and P_f, respectively. For these dependent tasks, the routing sets of the communication for the edges $e_{i,j}$, $e_{m,n}$, and $e_{v,j}$ are $\mathcal{L}_{s,d}$, $\mathcal{L}_{p,q}$, and $\mathcal{L}_{f,d}$, as shown in Fig. 1(a).

The communication contention in time is defined as the overlap of communication time for different edges, such as edges $e_{i,j}$ and $e_{m,n}$ in Fig. 1(b). If there is a communication time overlapping among $e_{i,j}$ and $e_{m,n}$, the contention constraints in time can be expressed as $ts_{i,j} < te_{m,n}$ and $ts_{m,n} < te_{i,j}$, where $ts_{i,j}$ and $ts_{m,n}$ ($te_{i,j}$ and $te_{m,n}$) denote the start time (end time) of data transmission for edge $e_{i,j}$ and $e_{m,n}$.

To indicate the sequence between $e_{i,j}$ and $e_{m,n}$, we introduce a binary variable $\eta_{i,j,m,n}$, where $\eta_{i,j,m,n} = 1$ denotes that edge $e_{i,j}$ begins communication before $e_{m,n}$ completes, otherwise, $\eta_{i,j,m,n} = 0$. Note that the value of $\eta_{i,j,m,n}$ is influenced by the start time and the end time of the edges $e_{i,j}$ and $e_{m,n}$, the relationship between them can be represented as

$$(te_{m,n} - ts_{i,j})/Z \le \eta_{i,j,m,n} \le (te_{m,n} - ts_{i,j})/Z + 1, \tag{5}$$

$$(te_{i,j} - ts_{m,n})/Z \le \eta_{m,n,i,j} \le (te_{i,j} - ts_{m,n})/Z + 1, \tag{6}$$

$$e_{i,j} \ne e_{m,n} \in \mathcal{E},\ \eta_{i,j,m,n}, \eta_{m,n,i,j} \in \{0, 1\},$$

where Z is a large positive integer. When $ts_{i,j} < te_{m,n}$ holds, (5) is relaxed to $\eta_{i,j,m,n} = 1$. Similarly, when $ts_{m,n} < te_{i,j}$ holds, (6) can be transformed into $\eta_{m,n,i,j} = 1$. As the example shown in Fig. 1(b), since there is a time overlap between the edges $e_{i,j}$ and $e_{m,n}$, we have $\eta_{i,j,m,n} = \eta_{m,n,i,j} = 1$.

The communication contention in space is defined that the routing sets of the edges have at least one shared link, e.g., in Fig. 1(a), for the edges $e_{i,j}$ and $e_{m,n}$, we have $\mathcal{L}_{s,d} \bigcap \mathcal{L}_{p,q} \ne \emptyset$, where $\mathcal{L}_{s,d} \bigcap \mathcal{L}_{p,q}$ denotes the common links between the routing sets of the edges $e_{i,j}$ and $e_{m,n}$. Hence, the communication contention occurs between the edges $e_{i,j}$ and $e_{m,n}$.

To alleviate the contention, we add flexible slack time into the communication process to ensure that the congested data can be transmitted sequentially. The slack time with a fixed value will generate additional idle time on cores, increasing energy consumption [13], [32]. In this paper, we set slack time as an adaptive value, which depends on the communication cost of overlapping links.

The slack time $t_{i,j}^{slack}$ and $t_{m,n}^{slack}$ for the edges $e_{i,j}$ and $e_{m,n}$ can be given by:

$$e_{i,j} \ne e_{m,n} \in \mathcal{E},\ P_s \ne P_d, P_p \ne P_q \in \mathcal{P},\ t_{i,j}^{slack}, t_{m,n}^{slack} \ge 0,$$
$$t_{i,j}^{slack}, t_{m,n}^{slack} \ge |\mathcal{L}_{s,d} \cap \mathcal{L}_{p,q}| \cdot \max \left\{ \overline{t_{i,j}^{UL}}, \overline{t_{m,n}^{UL}} \right\} -$$
$$Z(6 - x_{i,s} - x_{j,d} - x_{m,p} - x_{n,q} - \eta_{i,j,m,n} - \eta_{m,n,i,j}), \tag{7}$$

where $|\mathcal{L}_{s,d} \cap \mathcal{L}_{p,q}|$ denotes the number of overlapping links between the edges $e_{i,j}$ and $e_{m,n}$; binary variable $x_{i,s} = 1$ denotes task τ_i is assigned to core P_s, otherwise, $x_{i,s} = 0$; binary variable $c_{i,j,g}^L = 1$ denotes the message for edge $e_{i,j}$ is transmitted with $(v_{l,g}, f_{l,g})$, otherwise, $c_{i,j,g}^L = 0$; $\overline{t_{i,j}^{UL}} = \sum_{g \in \mathcal{N}_l} c_{i,j,g}^L t_{i,j,g}^{UL}$ is the transmission time on one unit link for the edge $e_{i,j}$. Similarly, for the edge $e_{m,n}$ we have $\overline{t_{m,n}^{UL}} = \sum_{g \in \mathcal{N}_l} c_{m,n,g}^L t_{m,n,g}^{UL}$.

As the example shown in Fig. 1, we have $x_{i,s} = x_{j,d} = x_{m,p} = x_{n,q} = x_{v,f} = 1$. When contention constraints in time and space are satisfied, i.e., $\eta_{i,j,m,n} = \eta_{m,n,i,j} = 1$

and $\mathcal{L}_{s,d} \bigcap \mathcal{L}_{p,q} \neq \emptyset$, the communication contention happens between the edges $e_{i,j}$ and $e_{m,n}$. Therefore, (7) is relaxed to

$$t_{i,j}^{slack}, t_{m,n}^{slack} \geq |\mathcal{L}_{s,d} \cap \mathcal{L}_{p,q}| \times \max\left\{\overline{t_{i,j}^{UL}}, \overline{t_{m,n}^{UL}}\right\}. \quad (8)$$

Note that the edges can operate at different V/F levels, and communication contention could happen over multiple edges simultaneously. For edges $e_{i,j}$ and $e_{m,n}$, the maximum communication time on overlapping links is $|\mathcal{L}_{s,d} \cap \mathcal{L}_{p,q}| \cdot \max\{\overline{t_{i,j}^{UL}}, \overline{t_{m,n}^{UL}}\}$. If there is no communication contention, i.e., contention constraints regarding time and space are not satisfied, since $\eta_{i,j,m,n} + \eta_{m,n,i,j} \leq 1$ and $\mathcal{L}_{s,d} \bigcap \mathcal{L}_{p,q} = \emptyset$, we have $-Z(6 - x_{i,s} - x_{j,d} - x_{m,p} - x_{m,q} - \eta_{i,j,m,n} - \eta_{m,n,i,j}) \ll 0$ and $|\mathcal{L}_{s,d} \cap \mathcal{L}_{p,q}| = 0$, and thus, (7) is relaxed to $t_{i,j}^{slack}, t_{m,n}^{slack} \geq 0$, which can be ignored since the lower bound of continuous variables $t_{i,j}^{slack}$ and $t_{m,n}^{slack}$ is 0. Similarly, for the contention between $e_{i,j}$ and $e_{v,j}$, we can get

$$t_{i,j}^{slack}, t_{v,j}^{slack} \geq |\mathcal{L}_{s,d} \cap \mathcal{L}_{f,d}| \times \max\left\{\overline{t_{i,j}^{UL}}, \overline{t_{v,j}^{UL}}\right\}. \quad (9)$$

When the contention happens over multiple edges concurrently, e.g., $e_{i,j}$, $e_{m,n}$, and $e_{v,j}$ in Fig. 1(b), the lower bound of slack time $t_{i,j}^{slack}$ should be updated according to (8) and (9), since $t_{i,j}^{slack}$ is influenced by $t_{m,n}^{slack}$ and $t_{v,j}^{slack}$ at the same time.

5) Fault-tolerant Model: During the application execution process, several faults can affect the system reliability, e.g., transient faults, intermittent faults, and permanent faults. We mainly focus on transient faults on cores, routers, and links since transient faults have higher rates than intermittent and permanent faults [29]. In addition, a fault on a router may affect the links connected to this router. In the worst case, a fault in any part of a router can be treated as a fault occurring in all the links related to the faulty router [18]. Therefore, we assume that the faults on routers are included in the faults over the links.

We consider that the average rate caused by transient faults is followed by the Poisson distribution [17], [35], [36]. Besides, the fault rate is affected by the voltage the system is run at [17]. Hence, the fault rate with a scaled frequency f can be modeled as $\lambda(f) = \lambda_0 \times 10^{\frac{d(f_{\max} - f)}{f_{\max} - f_{\min}}}$, where λ_0 is the average fault rate at $f_{\max}$, and d is a positive constant, representing the sensitivity of faults caused by transient faults to DVFS. Thus, the probability of success for a task on a hardware component with fault rate $\lambda(f)$ within the time interval t is $e^{-\lambda(f) \times t}$.

Considering the homogeneity of the NoC platform, we set the fault rates of cores and links as $\lambda_P(f)$ and $\lambda_L(f)$, respectively. The probability of success for task τ_i with the frequency $f_{p,k}$ on core P_s can be calculated as $r_{i,k}^P = e^{-\lambda_P(f_{p,k}) \times t_{i,k}^P}$, where $t_{i,k}^P = \frac{\omega_i}{f_{p,k}}$ denotes the execution time of task τ_i with V/F level $(v_{p,k}, f_{p,k})$. Similarly, the probability of successful transmission over one unit link for edge $e_{i,j}$ with the frequency $f_{l,g}$ can be given by $r_{i,j,g}^{UL} = e^{-\lambda_L(f_{l,g}) \times t_{i,j,g}^{UL}}$. In Fig. 1(a), when tasks τ_i and τ_j are assigned to cores P_s and P_d, respectively, the probability of success for edge $e_{i,j}$ along route $\mathcal{L}_{s,d}$ with the frequency $f_{l,g}$ is $r_{i,j,g}^L = \sum_{s \in \mathcal{M}} \sum_{d \in \mathcal{M}} x_{i,s} x_{j,d} (r_{i,j,g}^{UL})^{h_{s,d}}$, where $(r_{i,j,g}^{UL})^{h_{s,d}}$ represents the reliability probability of route $\mathcal{L}_{s,d}$ containing $h_{s,d}$ links. On this basis, the joint reliability probability of task τ_j can be calculated as

$$R_j = r_{j,k}^P \times r_{i,j,g}^L \times r_{v,j,g}^L = r_{j,k}^P \prod\nolimits_{e_{u,j} \in \mathcal{E}} r_{u,j,g}^L, \quad (10)$$

where $\prod_{e_{u,j} \in \mathcal{E}} r_{u,j,g}^L$ denotes the probability of successful transmission between task τ_j and its predecessors with the frequency $f_{l,g}$. If a task does not have any predecessors, e.g., task τ_a in Fig. 1(a), we consider only the faults on cores, and thus, the reliability probability of task τ_a is $R_a = r_{a,k}^P$. To maintain the desired system reliability, each task τ_j must satisfy its reliability constraints [21], which means the task reliability R_j is greater than the reliability threshold R_{th}.

B. Problem Formulation

This section presents the Mixed-Integer Nonlinear Programming (MINLP) formulation of the contention- and reliability-aware mapping problem. Our goal is to schedule and route each task such that real-time and reliability constraints are satisfied and the system performance (i.e., communication efficiency, energy consumption, and core utilization) is improved. Therefore, we must determine 1) task allocation, 2) frequency assignment, 3) edge scheduling, and 4) communication contention management. To formulate the task mapping problem, we introduce a set of binary and continuous variables, as summarized in Table II. For the sake of paper presentation, let $\mathcal{N} \triangleq \{1, \ldots, N\}$, $\mathcal{M} \triangleq \{1, \ldots, M\}$, $\mathcal{N}_p \triangleq \{1, \ldots, N_p\}$, $\mathcal{N}_l \triangleq \{1, \ldots, N_l\}$. The constraint descriptions are as follows:

1) Task Allocation Constraints: The task allocation variable $x_{i,s}$ is bounded by

$$\sum\nolimits_{s \in \mathcal{M}} x_{i,s} = 1, \ \forall i \in \mathcal{N}, \quad (11)$$

where (11) ensures that each task τ_i is assigned to a core.

2) Frequency Assignment Constraints: Since all cores and links in the target platform support DVFS, for each core and link, we have

$$\sum\nolimits_{k \in \mathcal{N}_p} c_{i,k}^P = 1, \ \forall i \in \mathcal{N}, \quad (12)$$

$$\sum\nolimits_{g \in \mathcal{N}_l} c_{i,j,g}^L = 1, \ \forall e_{i,j} \in \mathcal{E}, \quad (13)$$

where (12) ensures that a task τ_i is executed with one V/F core level $(v_{p,k}, f_{p,k})$, and (13) restricts that the message of edge $e_{i,j}$ is transmitted with one V/F link level $(v_{l,g}, f_{l,g})$.

3) Communication Start Time Constraints: The message transmission for edges should consider the precedence relations. For example, in Fig. 1(b), when the message for edge $e_{i,j}$ is transmitted from task τ_i to τ_j, the communication start time $ts_{i,j}$ is not earlier than the finish time of the predecessor τ_i. Therefore, we have

$$te_i \leq ts_{i,j}, \ \forall e_{i,j} \in \mathcal{E}. \quad (14)$$

4) Communication End Time Constraints: Taking task allocation and DVFS on links during task scheduling into account, the communication time of message $e_{i,j}$ is represented as

$$t_{i,j}^L = \left(\sum\nolimits_{s \in \mathcal{M}} \sum\nolimits_{d \in \mathcal{M}} h_{s,d} x_{i,s} x_{j,d}\right) \sum\nolimits_{g \in \mathcal{N}_l} c_{i,j,g}^L t_{i,j,g}^{UL}, \quad (15)$$

where $\sum_{g \in \mathcal{N}_l} c_{i,j,g}^L t_{i,j,g}^{UL}$ represents the transmission time over one unit link with DVFS, and $\sum_{s \in \mathcal{M}} \sum_{d \in \mathcal{M}} h_{s,d} x_{i,s} x_{j,d}$ is the number of links when transmitting the message for edge $e_{i,j}$.

If the message transmits from τ_i to τ_j through $e_{i,j}$, the communication end time $te_{i,j}$ is equal to the sum of communication

TABLE II
MAIN NOTATIONS USED IN PROBLEM FORMULATIONS

Related Parameters	
N	number of tasks in set $\mathcal{T} \triangleq \{\tau_i, \ldots, \tau_N\}$.
M	number of cores in set $\mathcal{P} \triangleq \{P_1, \ldots, P_M\}$.
$\mathcal{R}$	router sets, where $\mathcal{R} \triangleq \{R_1, \ldots, R_M\}$.
$l_{s,d}$	physical link between cores P_s and P_d.
$\mathcal{L}_{s,d}$	routing set between cores P_s and P_d.
$h_{s,d}$	Manhattan distance between core P_s and P_d.
$(v_{p,k}, f_{p,k})$	the k^{th} V/F level of core.
$(v_{l,g}, f_{l,g})$	the g^{th} V/F level of link.
N_p (N_l)	number of V/F levels in cores (links).
b_ω	link bandwidth.
$e_{i,j}$	dependency between tasks τ_i and τ_j, where $e_{i,j} \in \mathcal{E}$.
$s_{i,j}$	size of communication data between tasks τ_i and τ_j.
ω_i	execution cycles of task τ_i.
D_i	task deadline of task τ_i.
α_i	$= 1$ if task τ_i has no predecessors, else $\alpha_i = 0$.
λ_0, d	average fault rate and transient fault sensitivity.
R_{th}	reliability threshold.
μ	weight in the objective function.
Binary Variables	
$x_{i,s}$	$= 1$ if task τ_i is assigned to core P_s, else $x_{i,s} = 0$.
$c^P_{i,k}$	$= 1$ if task τ_i is executed with V/F level $(v_{p,k}, f_{p,k})$, else $c^P_{i,k} = 0$.
$c^L_{i,j,g}$	$= 1$ if the message for edge $e_{i,j}$ is transmitted with the level V/F $(v_{l,g}, f_{l,g})$, else $c^L_{i,j,g} = 0$.
$o_{i,j}$	$= 1$ if task τ_i starts execution before τ_j is completed, else $o_{i,j} = 0$.
$\eta_{i,j,m,n}$	$= 1$ if edge $e_{i,j}$ starts transmitting messages before $e_{m,n}$ completes, else $\eta_{i,j,m,n} = 0$.
Continuous Variables	
ts_i (te_i)	start (end) time of executing τ_i.
$ts_{i,j}$ $(te_{i,j})$	start (end) time of message transmission for edge $e_{i,j}$.
$t^{slack}_{i,j}$	slack time for edge $e_{i,j}$.
$U_{\max}$	maximum core utilization rate.
Auxiliary Variables	
$\beta_{i,s,k}$	binary variable, and $\beta_{i,s,k} = x_{i,s}c^P_{i,k}$.
$\gamma_{i,j,s,d}$	binary variable, and $\gamma_{i,j,s,d} = x_{i,s}x_{j,d}$.
$\delta_{i,j,s,d,g}$	binary variable, and $\delta_{i,j,s,d,g} = \gamma_{i,j,s,d}c^L_{i,j,g}$.

start time $ts_{i,j}$, communication time $t^L_{i,j}$, and slack time $t^{slack}_{i,j}$. The value of $te_{i,j}$ is not earlier than the start time of the successor τ_j. Thus, we have

$$te_{i,j} = ts_{i,j} + t^L_{i,j} + t^{slack}_{i,j} \leq ts_j, \ \forall e_{i,j} \in \mathcal{E}. \tag{16}$$

5) Real-time Constraints: For the real-time tasks, since each task τ_j must be completed within the deadline D_j, we have

$$te_j = ts_j + \sum\nolimits_{k \in \mathcal{N}_p} c^P_{j,k} t^P_{j,k} \leq D_j, \ \forall j \in \mathcal{N}, \tag{17}$$

where $\sum_{k \in \mathcal{N}_p} c^P_{j,k} t^P_{j,k}$ is the execution time of τ_j with DVFS.

6) Task Non-overlapping Constraints: If tasks are assigned to the same core, their execution sequence should be determined since one core cannot execute multiple tasks at the same time. Hence, we have the following constraints:

$$(te_j - ts_i)/Z \leq o_{i,j} \leq (te_j - ts_i)/Z + 1, \tag{18}$$

$$o_{i,j} + o_{j,i} + x_{i,s} + x_{j,s} \leq 3, \ \forall e_{i,j} \notin \mathcal{E}, \ \forall s \in \mathcal{M}. \tag{19}$$

If task τ_i starts the execution before the completion of task τ_j, i.e., $ts_i \leq te_j$, (18) is transformed to $o_{i,j} = 1$, otherwise, we get $o_{i,j} = 0$. If the execution time of τ_i and τ_j overlaps, and there is no dependency between τ_i and τ_j, $ts_j \leq te_i$ and $ts_i \leq te_j$ hold. We get $o_{i,j} = o_{j,i} = 1$ from (18). Therefore, (19) is transformed to $x_{i,s} + x_{j,s} \leq 1$, which means τ_i and τ_j cannot be assigned to the same core.

7) Reliability Constraints: Note that we consider transient faults on both cores and links. On the one hand, we consider only faults on cores for the task without predecessors, as there is no communication between this task and its predecessors. Let parameter $\alpha_j = 1$ denote task τ_j does not have predecessors, otherwise, $\alpha_j = 0$. With $\alpha_j = 1$ and DVFS, the reliability probability of task τ_j is given by

$$R_j = \sum\nolimits_{k \in \mathcal{N}_p} c^P_{j,k} r^P_{j,k}, \ \forall j \in \mathcal{N}, \ \alpha_j = 1. \tag{20}$$

On the other hand, if task τ_j has predecessors, i.e., $\alpha_j = 0$, we consider the reliability on both cores and links. Since the reliability of each link is related to the V/F, the reliability probability of links transmitting $e_{i,j}$ with DVFS is $\sum_{g \in \mathcal{N}_l} c^L_{i,j,g} r^L_{i,j,g}$. Therefore, the reliability probability of task τ_j is calculated as

$$R_j = \left(\sum\nolimits_{k \in \mathcal{N}_p} c^P_{j,k} r^P_{j,k}\right) \prod\nolimits_{e_{i,j} \in \mathcal{E}} \left(\sum\nolimits_{g \in \mathcal{N}_l} c^L_{i,j,g} r^L_{i,j,g}\right), \quad \forall j \in \mathcal{N}, \ \alpha_j = 0. \tag{21}$$

Combining (20) and (21), the reliability probability of task τ_j, including computation and communication, is formulated as:

$$R_j = \begin{cases} \sum_{k \in \mathcal{N}_p} c^P_{j,k} r^P_{j,k}, \ \forall j \in \mathcal{N}, \ \alpha_j = 1, \\ \left(\sum_{k \in \mathcal{N}_p} c^P_{j,k} r^P_{j,k}\right) \prod_{e_{i,j} \in \mathcal{E}} \left(\sum_{g \in \mathcal{N}_l} c^L_{i,j,g} r^L_{i,j,g}\right), \\ \forall j \in \mathcal{N}, \ \alpha_j = 0. \end{cases} \tag{22}$$

To run applications successfully, we must consider the reliability requirements of scheduling and routing for all tasks. Thus, the joint reliability probability of each task τ_j should be larger than reliability threshold R_{th}, and we have

$$R_j \geq R_{th}, \ \forall j \in \mathcal{N}. \tag{23}$$

8) Utilization Balance Constraints: Imbalanced mapping of tasks harms system reliability since the cores with heavy tasks assigned have high circuit activities, which raises the average fault rate λ_0 and decreases system reliability [22]. Moreover, load imbalance may cause local congestion in NoC, and the increased transmission delay will degrade communication efficiency. To improve system performance, we consider a load-balanced task mapping to minimize the maximum core utilization rate $U_{\max}$. To restrict the maximum core utilization rate among the cores, we have the following constraints:

$$U_s = \underbrace{\left(\sum\nolimits_{i \in \mathcal{N}} \sum\nolimits_{k \in \mathcal{N}_p} x_{i,s} c^P_{i,k} t^P_{i,k}\right)}_{\text{(a)}} / H \leq U_{\max}, \ \forall s \in \mathcal{M}, \tag{24}$$

where U_s denotes the core utilization rate of P_s, given by the ratio of the total task execution time on core P_s (i.e., (24)-(a)) to the scheduling horizon H.

9) Objective Function: We aim to minimize the weighted core utilization rate and total energy consumption. The objective function can be formulated as follows:

$$\min \ \mu E_{total}/E_{\max} + (1-\mu)U_{\max}. \tag{25}$$

where $\mu \in [0,1]$ is a weighted factor, E_{total} is the total energy consumption, and $E_{\max}$ is an upper bound of E_{total}.

$$E_{total} = \sum\nolimits_{e_{i,j}\in\mathcal{E}} \sum\nolimits_{s\in\mathcal{M}} \sum\nolimits_{d\in\mathcal{M}} x_{i,s}x_{j,d}s_{i,j}[(h_{s,d}+1)E_{Rbit} + h_{s,d}\sum\nolimits_{g\in\mathcal{N}_l} c^L_{i,j,g}E_{Lbit,g}] + \sum\nolimits_{i\in\mathcal{N}}\sum\nolimits_{k\in\mathcal{N}_p} c^P_{i,k}P^{comp}_k t^P_{i,k}. \tag{26}$$

10) Primal Problem: Based on the above constraints and objective function, the contention- and reliability-aware task mapping problem is formulated as follows:

$$\mathbf{PP}: \min_{\substack{\boldsymbol{x},\boldsymbol{c^P},\boldsymbol{c^L},\boldsymbol{o},\boldsymbol{\eta},\\ \boldsymbol{ts^P},\boldsymbol{ts^L},\boldsymbol{t},u}} \mu E_{total}/E_{max} + (1-\mu)U_{max} \tag{27}$$

$$\text{s.t.} \begin{cases} (5)-(7),(11)-(19),(22)-(24), \\ x_{i,s}, c^P_{i,k}, c^L_{i,j,g}, o_{i,j}, \eta_{i,j,m,n} \in \{0,1\},\ 0 \le ts_i, ts_{i,j} \le H, \\ \forall i,j,m,n \in \mathcal{N},\ \forall s \in \mathcal{M},\ \forall k \in \mathcal{N}_p,\ \forall g \in \mathcal{N}_l, \end{cases}$$

where we have $\boldsymbol{x} = [x_{i,s}]_{N\times M}$, $\boldsymbol{c^P} = [c^P_{i,k}]_{N\times N_p}$, $\boldsymbol{c^L} = [c^L_{i,j,g}]_{N\times N\times N_l}$, $\boldsymbol{o} = [o_{i,j}]_{N\times N}$, $\boldsymbol{\eta} = [\eta_{i,j,m,n}]_{N\times N\times N\times N}$, $\boldsymbol{ts^P} = [ts_i]_{1\times N}$, $\boldsymbol{ts^L} = [ts_{i,j}]_{N\times N}$, $\boldsymbol{t} = [t^{slack}_{i,j}]_{N\times N}$, and $u = U_{\max}$. Note that $\boldsymbol{x}$, $\boldsymbol{c^P}$, $\boldsymbol{c^L}$, $\boldsymbol{o}$, and $\boldsymbol{\eta}$ are binary variables, while $\boldsymbol{ts^P}$, $\boldsymbol{ts^L}$, $\boldsymbol{t}$, and $\boldsymbol{u}$ are continuous variables. Moreover, the nonlinear terms $e^{-\lambda t}$, $x_{i,s}c^P_{i,k}$, and $x_{i,s}x_{j,d}c^L_{i,j,g}$ are included in (15), (23), (24) and (26). Thus, (27) is an MINLP problem.

Remark 3.1: DVFS switching overheads are mainly influenced by the V/F levels before and after transition [37]. The time overhead is $t^{switch}(f_s,f_e) = t_{trans}\times(f_e-f_s)/f_e$, and the energy overhead is $E^{switch}(f_s,f_e) = P^{switch}_s \times t^{switch}(f_s,f_e)$, where t_{trans} is a time transition parameter, f_s and f_e are the frequencies before and after transition, respectively, and P^{switch}_s is the power before frequency transition. Based on the parameters of the NoC platform, we can get switching overheads regarding the links and the cores, e.g., $t^{Lswitch}_{i,j}$ and $E^{Lswitch}_{i,j}$ for data transmission among the edge $e_{i,j}$, and $t^{Pswitch}_i$ and $E^{Pswitch}_i$ for execution of task τ_i. Considering the DVFS switching overhead in PP, the extra time ($t^{Lswitch}_{i,j}$ and $t^{Pswitch}_i$) and energy ($E^{Lswitch}_{i,j}$ and $E^{Pswitch}_i$) should be added to constraints (16), (17), and (26).

Remark 3.2: In (2), the leakage current I_{sub} is affected by temperature T. The relationship between them can be described by a complex nonlinear function [38]. To reduce complexity, the leakage current can be linearized with the temperature within a given acceptable accuracy [16], [20], [27]. Specifically, the static power working at the V/F level $(v_{p,k}, f_{p,k})$ can be modeled as $P^{comp}_{sta,k} = (\gamma_s + \eta_s T)\times v_{p,k}$, where γ_s and η_s are constants depending on core P_s. Let $T(t)$ denote the temperature at time t. In (26), the energy consumed by the static power can be updated to $\sum_{s\in\mathcal{M}}\sum_{i\in\mathcal{N}}\sum_{k\in\mathcal{N}_p} x_{i,s}c^P_{i,k}t^P_{i,k}v_{p,k}(\gamma_s + \eta_s\int_0^H T(t)dt)$.

Remark 3.3: The above problem can be extended to heterogeneous platforms by modifying the constraints (12), (17), (22), (24), and (26). In addition, we can replace the frequency assignment variable $c^P_{i,k}$ with $c^P_{i,s,k}$, where $c^P_{i,s,k} = 1$ denotes that task τ_i assigned to core P_s is executed with V/F level $(v^P_{s,k}, f^P_{s,k})$. Similarly, the execution time of τ_i on P_s with $(v^P_{s,k}, f^P_{s,k})$ is updated as $t^P_{i,s,k} = \omega_i/f^P_{s,k}$, and the reliability of executing τ_i on P_s with $(v^P_{s,k}, f^P_{s,k})$ is represented as $r^P_{i,s,k} = e^{-\lambda_P \times f^P_{s,k}\times t^P_{i,s,k}}$.

C. Problem Linearization

The MINLP problem has been proven to be an NP-Hard [39], which is hard to solve within polynomial time. To find the optimal solution to PP, we propose a linearization method, which can equivalently convert PP into an MILP problem.

1) Nonlinear terms caused by exponential terms: We introduce the following lemma to linearize the product of binary variables and exponential terms in (23).

Lemma 3.1: Assume that z_i is a binary variable, and the constant $a_i > 0$ for each $i \in \mathcal{N}$. When $\sum_{i\in\mathcal{N}} z_i = 1$, the equation $\ln(\sum_{i\in\mathcal{N}} a_i z_i) = \sum_{i\in\mathcal{N}} z_i \ln a_i$ holds.

Proof: Considering the case when binary variable $z_m = 1$, while the other binary variable $z_n = 0$ ($\forall n \ne m \in \mathcal{N}$) due to $\sum_{i\in\mathcal{N}} z_i = 1$. Therefore, we have $\ln(\sum_{i\in\mathcal{N}} a_i z_i) = \ln(a_1 \times 0 + \cdots + a_m z_m + \cdots + a_N \times 0) = \ln a_m z_m = \ln a_m$. On the other hand, we get $\sum_{i\in\mathcal{N}} z_i \ln a_i = 0\times\ln a_1 + \cdots + z_m \ln a_m + \cdots + 0 \times a_N = z_m \ln a_m = \ln a_m$. Therefore, it is proved that $\ln(\sum_{i\in\mathcal{N}} a_i z_i) = \sum_{i\in\mathcal{N}} z_i \ln a_i$ when $\sum_{i\in\mathcal{N}} z_i = 1$. ■

Since logarithm is a monotonically increasing function, the relation between left and right parts in (23) remains unchanged when logarithm operation is performed on both sides of (23). According to *Lemma* 3.1, (23) can be linearized as follows:

$$\ln R_j \ge \ln R_{th},\ \forall j \in \mathcal{N}, \tag{28}$$

where

$$\ln R_j = \begin{cases} \sum_{k\in\mathcal{N}_p} c^P_{j,k}\ln r^P_{j,k},\ \alpha_j = 1, \\ \sum_{e_{i,j}\in\mathcal{E}}\sum_{g\in\mathcal{N}_l} c^L_{i,j,g}\ln r^L_{i,j,g} + \sum_{k\in\mathcal{N}_p} c^P_{j,k}\ln r^P_{j,k},\ \alpha_j = 0, \end{cases}$$

$$\ln r^P_{j,k} = \ln e^{-\lambda_P(f_{p,k})\times t^P_{j,k}} = -\frac{\omega_j \lambda_P(f_{p,k})}{f_{p,k}},$$

$$\ln r^L_{i,j,g} = \sum\nolimits_{s\in\mathcal{M}}\sum\nolimits_{d\in\mathcal{M}} x_{i,s}x_{j,d}h_{s,d}\ln r^{UL}_{i,j,g} = -\sum\nolimits_{s\in\mathcal{M}}\sum\nolimits_{d\in\mathcal{M}} x_{i,s}x_{j,d}\frac{h_{s,d}s_{i,j}\lambda_L(f_{l,g})}{b_\omega f_{l,g}}.$$

Taking the value of α_j in the above equations into account, (28) can be reformulated as follows:

$$\ln R_j = (1-\alpha_j)\sum\nolimits_{e_{i,j}\in\mathcal{E}}\sum\nolimits_{g\in\mathcal{N}_l} c^L_{i,j,g}\ln r^L_{i,j,g} + \sum\nolimits_{k\in\mathcal{N}_p} c^P_{j,k}\ln r^P_{j,k} \ge \ln R_{th},\ \forall j \in \mathcal{N}. \tag{29}$$

2) Nonlinear terms caused by the products of binary variables: We propose the following lemma to tackle the nonlinear items $x_{i,s}c^P_{i,k}$ and $x_{i,s}x_{j,d}c^L_{i,j,g}$ in (15), (24), (26), and (29).

Lemma 3.2: Assume that x_1 and x_2 are binary variables. The nonlinear term x_1x_2 can be replaced by an auxiliary (binary) variable y, where $y = x_1x_2$, and the additional constraints $y \le x_1$, $y \le x_2$ and $y \ge x_1 + x_2 - 1$.

Proof: When binary variables $x_1 = 1$ and $x_2 = 1$, the additional constraint is transformed to $1 \le y \le 1$, i.e., $y = 1$

holds. Similarly, for the cases 1) $x_1 = 0$ and $x_2 = 0$; 2) $x_1 = 1$ and $x_2 = 0$; 3) $x_1 = 0$ and $x_2 = 1$, we have $y = x_1 x_2 = 0$. ■

Based on *Lemma* 3.2, we introduce auxiliary (binary) variables $\beta_{i,s,k}$, $\gamma_{i,j,s,d}$ and $\delta_{i,j,s,d,g}$ to replace the nonlinear items, where $\beta_{i,s,k} = x_{i,s} c^P_{i,k}$, $\gamma_{i,j,s,d} = x_{i,s} x_{j,d}$, and $\delta_{i,j,s,d,g} = \gamma_{i,j,s,d} c^L_{i,j,g}$, and add the following constraints into (27).

$$\forall i, j \in \mathcal{N},\ \forall s, d \in \mathcal{M},\ \forall k \in \mathcal{N}_p,\ \forall g \in \mathcal{N}_l,$$
$$\beta_{i,s,k} \le x_{i,s}, \beta_{i,s,k} \le c^P_{i,k}, \beta_{i,s,k} \ge x_{i,s} + c^P_{i,k} - 1, \quad (30)$$
$$\gamma_{i,j,s,d} \le x_{i,s}, \gamma_{i,j,s,d} \le x_{j,d}, \gamma_{i,j,s,d} \ge x_{i,s} + x_{j,d} - 1, \quad (31)$$
$$\delta_{i,j,s,d,g} \le \gamma_{i,j,s,d}, \delta_{i,j,s,d,g} \le c^L_{i,j,g},$$
$$\delta_{i,j,s,d,g} \ge \gamma_{i,j,s,d} + c^L_{i,j,g} - 1. \quad (32)$$

Thus, the primal problem (27) can be converted into the following MILP problem through the above linearization method.

$$\mathbf{PP1}: \min_{\substack{\boldsymbol{x}, \boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}, \boldsymbol{\eta}, \boldsymbol{ts^P}, \\ \boldsymbol{ts^L}, \boldsymbol{t}, u, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}}} \mu E_{total}/E_{max} + (1-\mu) U_{max} \quad (33)$$
$$\text{s.t.} \begin{cases} (5)-(7),\ (11)-(14), (16)-(19), (29)-(32), \\ t^L_{i,j} = \sum_{s\in\mathcal{M}} \sum_{d\in\mathcal{M}} \sum_{g\in\mathcal{N}_l} h_{s,d} \delta_{i,j,s,d,g} t^{UL}_{i,j,g}, \\ U_s = \left(\sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{N}_p} \beta_{i,s,k} t^P_{i,k} \right) / H \le U_{\max}, \\ x_{i,s}, c^P_{i,k}, c^L_{i,j,g}, o_{i,j}, \eta_{i,j,m,n}, \beta_{i,s,k}, \gamma_{i,j,s,d}, \\ \delta_{i,j,s,d,g} \in \{0,1\},\ 0 \le ts_i, ts_{i,j} \le H, \\ \forall i, j, m, n \in \mathcal{N},\ \forall s, d \in \mathcal{M},\ \forall k \in \mathcal{N}_p,\ \forall g \in \mathcal{N}_l, \end{cases}$$

where we set $\boldsymbol{\beta} = [\beta_{i,s,k}]_{N\times M\times N_p}$, $\boldsymbol{\gamma} = [\gamma_{i,j,s,d}]_{N\times N\times M\times M}$, and $\boldsymbol{\delta} = [\delta_{i,j,s,d,g}]_{N\times N\times M\times M\times N_l}$, and the total energy consumption E_{total} in the objective function of (33) is given by $E_{total} = \sum_{e_{i,j}\in\mathcal{E}} \sum_{s\in\mathcal{M}} \sum_{d\in\mathcal{M}} s_{i,j}[(h_{s,d}+1)\gamma_{i,j,s,d} E_{Rbit} + h_{s,d} \sum_{g\in\mathcal{N}_l} \delta_{i,j,s,d,g} E_{Lbit,g}] + \sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{N}_p} c^P_{i,k} P^{comp}_k t^P_{i,k}$.

IV. Heuristic Task Mapping Approach

The proposed contention- and reliability-aware task mapping problem can be transformed into an MILP problem using the linearization methods presented in Section III-C. However, only small-scale problems can be solved efficiently by optimization solvers, such as Gurobi and CPLEX. The problem with large scale will lead to excessive time consumption and memory overflow problems. Hence, we proposed a novel three-step Contention- and Reliability-Aware Task Mapping and Scheduling (CRATMS) heuristic to perform 1) task-to-core assignment, 2) frequency scaling and edge scheduling, and 3) communication contention management, so as to provide better system performance. The structure of CRATMS is shown in Fig. 2.

A. Task Allocation Scheme

Since the tasks are dependent, the task-to-core assignment directly affects task scheduling and communication on NoC. In addition, the task allocation variable $\boldsymbol{x}$ exists in all the constraints of PP except the frequency assignment constraints (12) and (13), and thus, we consider $\boldsymbol{x}$ at the first step. The optimization processes are summarized in Algorithm 1.

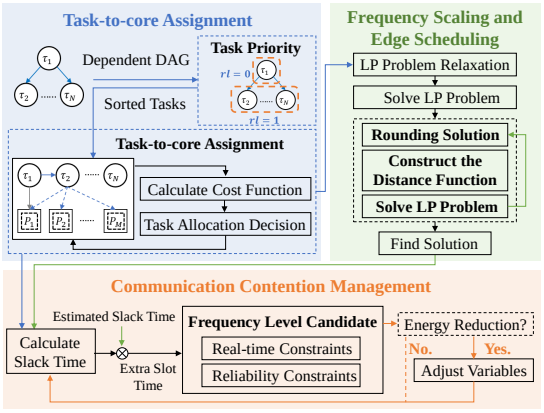


Fig. 2. The structure of the proposed CRATMS scheme.

1) Task Priority: We introduce the concept of task level to sort the dependent tasks in DAG. The task level of task τ_i is defined as the largest number of edges from itself to the entry node: $rl(\tau_j) = \max_{\forall e_{i,j}\in\mathcal{E}}\{rl(\tau_i)\} + 1$, and we set the task level of entry nodes as zero. For example, in Fig. 1(a), we have $rl(\tau_a) = 0$, $rl(\tau_b) = 1$, $rl(\tau_v) = 1$, $rl(\tau_m) = 1$, $rl(\tau_i) = 2$, $rl(\tau_j) = 3$, $rl(\tau_n) = 3$, $rl(\tau_u) = 4$, and $rl(\tau_w) = 5$. Therefore, the tasks in the set $\mathcal{T}$ can be sorted in ascending order according to their task levels, and we can get a new task set $\widehat{\mathcal{T}}$ (Line 2).

2) Task-to-core Assignment: To reduce the routing costs while balancing the task workloads among the cores, we consider the cost function of task τ_j assigned to core P_d as follows:

$$F(\tau_j, P_d) = \underbrace{\left(\mu \sum\nolimits_{e_{i,j}\in\mathcal{E}} \sum\nolimits_{s\in\mathcal{M}} x_{i,s} h_{s,d} \right) / h_{\max}}_{\text{(a)}} + \underbrace{(1-\mu)\left(U_d + t^P_{j,\min}/H\right)}_{\text{(b)}} \quad (34)$$

where $h_{\max}$ is the maximum Manhattan distance on NoC through the XY routing, and $t^P_{j,\min}$ is the execution time of task τ_j on core P_d with the minimum frequency $f_{p,\min}$. Hence, (34)-(a) is the ratio of routing distances between the cores executing tasks τ_i and τ_j to the maximum distance $h_{\max}$, and (34)-(b) is the core utilization of P_d. To reduce the cost of task assignment, for each non-entry task τ_j, they are assigned to the core P_{d^*} with the minimum cost function, i.e., $d^* = \arg\min_{d\in\mathcal{M}} F(\tau_j, P_d)$ (Line 5). However, since the entry tasks do not have predecessors, we assume that these tasks are assigned to core P_1, i.e., $d^* = 1$ (Line 7). After completing the assignment of task τ_j, we can update the task assignment decision $x_{j,d^*} = 1$ (Line 9) and core utilization rate $U_d = U_d + t^P_{j,\min}/H$ (Line 10). The decision regarding task allocation $\boldsymbol{x}^*$ is determined by applying the above method for each task.

B. Frequency Scaling and Edge Scheduling

Under the given task allocation decision, the frequencies of cores and links relate to energy consumption, task reliability, and real-time constraints, affecting the starting and ending time of task execution and edge communication. Therefore, the variables regarding core frequency assignment $\boldsymbol{c^P}$, link frequency assignment $\boldsymbol{c^L}$, task sequence $\boldsymbol{o}$, start time of task execution

Algorithm 1: Task Allocation Scheme

Input : Task graph TG and 2D-mesh NoC with M cores
Output: Task allocation variable $\boldsymbol{x}^*$

Initialize $x_{j,d} = 0$ and $U_d = 0$ ($\forall j \in \mathcal{N}$, $\forall d \in \mathcal{M}$) ;
Sort tasks in $\mathcal{T}$ by ascending task levels, and get $\widehat{\mathcal{T}}$;
for *each task* τ_j *in* $\widehat{\mathcal{T}}$ **do**
  if $rl(\tau_j) \neq 0$ **then**
    Get $d^* = \arg\min_{d \in \mathcal{M}} F(\tau_j, P_d)$ through (34) ;
  else
    Set $d^* = 1$;
  end
  Assign task τ_j to core P_{d^*}, and set $x_{j,d^*} = 1$;
  Update $U_{d^*} = U_{d^*} + t^P_{j,min}/H$;
end

$\boldsymbol{ts^P}$ and start time of message transmission $\boldsymbol{ts^L}$ are determined at this step concurrently. However, according to (7), the coupling between the slack time $\boldsymbol{t}$ and other variables, e.g., $\boldsymbol{x}$, $\boldsymbol{c^L}$, $\boldsymbol{ts^P}$, and $\boldsymbol{ts^L}$ makes it challenging to solve the problem of frequency scaling and edge scheduling. To handle this problem, we assume that the slack time $\boldsymbol{t}$ has a predetermined value $\hat{\boldsymbol{t}}$, which is calculated from (7) in the worst case. In this case, the communication contention constraints for any two edges, e.g., $e_{i,j}$ and $e_{m,n}$ in time are satisfied ($\eta_{i,j,m,n} = \eta_{m,n,i,j} = 1$), and the link frequencies are set to the minimum frequency $f_{l,\min}$. Moreover, the objective function of (27) can be transformed to minimize the energy consumption on NoC since the core utilization rate has been determined in the first step. Thus, substituting $x^*_{i,s}$ and $\hat{t}^{slack}_{i,j}$ into (27), the problem of frequency scaling and edge scheduling is given by

$$\textbf{SP1}: \min_{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}, \boldsymbol{ts^P}, \boldsymbol{ts^L}} E_{total} \tag{35}$$
$$\text{s.t.} \begin{cases} (12)-(19), (29), \\ c^P_{i,k}, c^L_{i,j,g}, o_{i,j} \in \{0,1\},\ 0 \leq ts_i, ts_{i,j} \leq H, \\ \forall i,j \in \mathcal{N},\ \forall k \in \mathcal{N}_p,\ \forall g \in \mathcal{N}_l. \end{cases}$$

Since (35) is an MILP problem with a smaller scale than (27), based on the structure of the problem, we proposed a novel Feasibility-Pump (FP) based method to solve it. The implementation details are summarized in Algorithm 2.

1) Let $\Phi = \{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\}$ denote the set of the variables in SP1, and we set $\mathcal{I} = \{j | \phi_j \in \{0,1\}\}$, where ϕ_j is the j^{th} element in the set Φ. To make problem (35) easier to solve, we relax it into a Linear Program (LP) problem SP2 by replacing the binary variables $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ with the continuous variables within the range $[0, 1]$ (Line 1). When SP2 is solved, we get the initial solution $\Phi^{(0)^*} = \{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\} \in \mathcal{Z}$ (Line 1), where $\mathcal{Z}$ denotes the set of the feasible region of SP2.

2) We initialize the iteration number $n = 0$ (Line 2). Since the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ in $\Phi^{(0)^*}$ may not be integer, we round the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ to the nearest integers (Line 3). When considering the task-level DVFS, each task τ_i is executed with only one V/F level on each core. Based on the i^{th} row of $\boldsymbol{c^P}$, we can find the maximum one among the values $\{c^P_{i,1}, \ldots, c^P_{i,N_p}\}$, and denote it as $k^* = \arg\max_{k \in \mathcal{N}_p}\{c^P_{i,k}\}$. For task τ_i, we assign the V/F level (v_{p,k^*}, f_{p,k^*}) to execute this task. Therefore, we have $\hat{c}^P_{i,k} = 1$ ($k = k^*$) and $\hat{c}^P_{i,k} = 0$ ($\forall k \neq k^* \in \mathcal{N}_p$). Similarly, we can handle the $\boldsymbol{c^L}$ and $\boldsymbol{o}$.

Algorithm 2: Frequency Scaling and Edge Scheduling

Input : Task allocation $\boldsymbol{x}$ and the predetermined slack time $\hat{\boldsymbol{t}}$
Output: Frequency assignment $\boldsymbol{c^P}^*$ and $\boldsymbol{c^L}^*$, and the start time of execution (communication) $\boldsymbol{ts^P}^*$ ($\boldsymbol{ts^L}^*$)

Relax SP1 to a LP problem and get the initial solution $\Phi^{(0)^*}$;
Initialize iteration number $n = 0$;
Round $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ in $\Phi^{(0)^*}$ to $\{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{L}}, \hat{\boldsymbol{o}}\}$, and update the solution $\hat{\Phi}^{(0)} = \{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{L}}, \hat{\boldsymbol{o}}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\}$;
while $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) > 0$ *and* $n < n_{\max}$ **do**
  Solve $\Phi^{(n)^*} = \arg\min_{\Phi^{(n)}}\{\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) : \Phi^{(n)} \in \mathcal{Z}\}$;
  if $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) = 0$ **then**
    Find the solution to SP1 and return $\Phi^{(n)^*}$;
  else
    Round $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ in $\Phi^{(n)^*}$ to $\{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{L}}, \hat{\boldsymbol{o}}\}$, and get $\hat{\Phi}^{(n)*} = \{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{L}}, \hat{\boldsymbol{o}}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\}$;
    Set $\hat{\Phi}^{(n+1)} = \hat{\Phi}^{(n)^*}$ for the next iteration;
    if $\hat{\Phi}^{(n+1)} = \hat{\Phi}^{(n)}$ **then**
      Select n_f variables $\hat{\phi}_j$ ($j \in \mathcal{I}$) in $\hat{\Phi}^{(n+1)}$ to flip their values, and update $\hat{\Phi}^{(n+1)}$;
    end
    Update iteration number $n \to n+1$;
  end
end
if $n = n_{\max} + 1$ *and* $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) > 0$ **then**
  Solve SP1 after substituting $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ with the corresponding values in $\hat{\Phi}^{(n)}$, and get $\Phi^{(n+1)^*}$;
  while $\Phi^{(n+1)^*} = \emptyset$ **do**
    Update frequency levels with $c^P_{i,k} = 1 \to c^P_{i,k+1} = 1$ and $c^L_{i,j,g} = 1 \to c^L_{i,j,g+1} = 1$;
    Solve SP1 after substituting $\{\boldsymbol{c^P}, \boldsymbol{c^L}\}$ with the updated values, and update $\Phi^{(n+1)^*}$;
  end
end

Specially, the rounding of $o_{i,j}$ is given by $\hat{o}_{i,j} = [o_{i,j}]$, where $[\cdot]$ represents the rounding operation. Therefore, the rounding processes of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ are as follows:

$$\begin{cases} \hat{c}^P_{i,k} = 1, \text{if } k = \arg\max\{c^P_{i,k}\}, \text{else},\ \hat{c}^P_{i,k} = 0, \\ \hat{c}^L_{i,j,g} = 1, \text{if } g = \arg\max\{c^L_{i,j,g}\}, \text{else},\ \hat{c}^L_{i,j,g} = 0, \\ \hat{o}_{i,j} = [o_{i,j}],\ \forall i,j \in \mathcal{N},\ \forall k \in \mathcal{N}_p,\ \forall g \in \mathcal{N}_l. \end{cases} \tag{36}$$

According to (36), the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ are transformed to $\{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{L}}, \hat{\boldsymbol{o}}\}$, and the rounding result is $\hat{\Phi} = \{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{o}}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\}$ (Line 3). Note that there is no need to round the values of $\boldsymbol{ts^P}$ and $\boldsymbol{ts^L}$ as they are continuous. Therefore, we obtain $\hat{\Phi}^{(0)} = \{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{o}}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\}$.

3) Note that $\Phi^{(0)^*} = \{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}, \boldsymbol{ts^P}, \boldsymbol{ts^L}\}$ is the solution to LP-based SP2. However, the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ may not be integer. Although the values of $\{\hat{\boldsymbol{c}}^{\boldsymbol{P}}, \hat{\boldsymbol{c}}^{\boldsymbol{L}}, \hat{\boldsymbol{o}}\}$ in $\hat{\Phi}^{(0)}$ are integer, $\hat{\Phi}^{(0)}$ may not be the solution to SP2 as the constraints may be violated. Since SP2 is relaxed from SP1, they have the same set of constraints. If the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ in $\Phi^{(0)^*}$ are integer, SP1 and SP2 have the same solution. For the solution to SP1, besides the constraints in SP2 must be satisfied, the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ should be the integer. To find the solution to SP1, based on the given rounding solution $\hat{\Phi}$, we construct a L1-norm distance between the solutions $\Phi^* \in \mathcal{Z}$ and $\hat{\Phi}$, and we define it as $\Delta(\Phi^*, \hat{\Phi}) = \sum_{j \in \mathcal{I}, \hat{\phi}_j = 1}(1 - \phi^*_j) + \sum_{j \in \mathcal{I}, \hat{\phi}_j = 0} \phi^*_j$, where $\hat{\phi}_j$ and ϕ^*_j are the j^{th} elements in $\hat{\Phi}$ and Φ^*, respectively. Since $\Delta(\Phi^*, \hat{\Phi}) \geq 0$, we have the following two cases:

Algorithm 3: Communication Contention Management

Input : The predetermined slack time $\hat{\boldsymbol{t}}$ and the outputs of Algorithms 1 and 2
Output: Core frequency assignment $\boldsymbol{c^P}^*$, the start time of executing tasks $\boldsymbol{ts^P}^*$, and the precise slack time $\boldsymbol{t}^*$

1 Update slack time $t_{i,j}^{slack*}$ ($\forall e_{i,j} \in \mathcal{E}$) based on (5)-(7);
2 Calculate time slot $\Delta t_{i,j} = \hat{t}_{i,j}^{slack} - t_{i,j}^{slack*}$ ($\forall e_{i,j} \in \mathcal{E}$);
3 **for** *each edge* $e_{i,j}$ *in* $\mathcal{E}$ **do**
4 Initialize $k_j^{init} = k$ for task τ_j if ${c_{j,k}^P}^* = 1$;
5 Calculate feasible frequency level $\hat{k}_j = \max\{k_j^R, k_j^T\}$ through (38) and (39) for task τ_j;
6 **if** $E_j^{comp}(f_{p,\hat{k}_j}) < E_j^{comp}(f_{p,k_j^{init}})$ **then**
7 Update ${c_{j,k}^P}^* = 1$ ($k = \hat{k}_j$) and ${c_{j,k}^P}^* = 0$ ($k \neq \hat{k}_j \in \mathcal{N}_p$);
8 **end**
9 Calculate task start time ts_j^* by (17);
10 **end**

a). If $\Delta(\Phi^*, \hat{\Phi}) = 0$, ϕ_j^* ($\forall j \in \mathcal{I}$) is integer and $\phi_j^* = \hat{\phi}_j$, and thus, Φ^* is a solution to SP1.

b). If $\Delta(\Phi^*, \hat{\Phi}) > 0$, we round Φ^* and update $\hat{\Phi}$ with the rounding result. On this basis, to reduce the distance $\Delta(\Phi^*, \hat{\Phi})$, we update the values of Φ^* and $\hat{\Phi}$ iteratively until the distance $\Delta(\Phi^*, \hat{\Phi}) = 0$.

In the n^{th} iteration, to minimize $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)})$ through $\Phi^{(n)}$, we construct a new problem SP3 as follows:

$$\textbf{SP3}: \min_{\Phi^{(n)}} \Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) \tag{37}$$
$$\text{s.t.} \begin{cases} (12)-(19), (29), \\ c_{i,k}^P, c_{i,j,g}^L, o_{i,j} \in [0,1],\ 0 \leq ts_i, ts_{i,j} \leq H, \\ \forall i,j \in \mathcal{N},\ \forall k \in \mathcal{N}_p,\ \forall g \in \mathcal{N}_l, \end{cases}$$

where $\Phi^{(n)}$ and $\hat{\Phi}^{(n)}$ denote the sets of variables and the rounding solution in the n^{th} iteration, respectively. Note that SP3 has the same form as SP2 except for the objective function. In the n^{th} iteration, we fix $\hat{\Phi}^{(n)}$ and solve SP3 to obtain the solution ${\Phi^{(n)}}^* \in \mathcal{Z}$, so as to reduce the current distance $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)})$ (Line 5). If $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) = 0$, ${\Phi^{(n)}}^*$ is the solution to SP1, and the iteration stops (Lines 6-7). Otherwise, we round the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ in ${\Phi^{(n)}}^*$ through (36), and get a rounding solution $\hat{\Phi}^{(n)^*}$, which is the given solution $\hat{\Phi}^{(n+1)} = \hat{\Phi}^{(n)^*}$ in the $(n+1)^{th}$ iteration (Lines 9-10).

Specially, if $\hat{\Phi}^{(n+1)} = \hat{\Phi}^{(n)}$, the solution to SP3 stays unchanged in the $(n+1)^{th}$ iteration, which means the distance $\Delta(\Phi^{(n+1)}, \hat{\Phi}^{(n+1)})$ is unchanged as well. To avoid endless loop, we randomly select n_f elements, e.g., $\hat{\phi}_j \in \hat{\Phi}^{(n+1)}$ ($j \in \mathcal{I}$), and flip their values (Lines 11-13), where $n_f \in [N_F/2, 3N_F/4]$, and N_F is a positive integer related to flipping number. During the flipping process, the value of $\hat{\phi}_j$ is reset to 1 if $\hat{\phi}_j = 0$, otherwise, $\hat{\phi}_j$ is reset to 0. Therefore, we can update $\hat{\Phi}^{(n+1)}$ to avoid cycling issue (Line 12). Then, we update the iteration number $n \to n+1$ (Line 14), and the iteration continues if $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) > 0$ or $n < n_{\max}$.

After iteration stops, we have $n = n_{\max} + 1$. If the distance $\Delta(\Phi^{(n)}, \hat{\Phi}^{(n)}) > 0$, the solution to SP1 is not founded, and we only get the rounding solution $\hat{\Phi}^{(n)}$. This solution cannot satisfy all constraints in SP1, especially the real-time and reliability constraints. Note that frequency assignment variables $\{\boldsymbol{c^P}, \boldsymbol{c^L}\}$ have an influence on real-time constraints (17) and reliability constraints (29). We iteratively adjust variables $\{\boldsymbol{c^P}, \boldsymbol{c^L}\}$ based on the rounding solution $\hat{\Phi}^{(n)}$ until the constraints in SP1 are satisfied (Lines 17-23). The details are as follows.

a). Since the rounding solution $\hat{\Phi}^{(n)}$ cannot satisfy the constraints in SP1, we solve SP1 by substituting $\{\boldsymbol{c^P}, \boldsymbol{c^L}, \boldsymbol{o}\}$ with the corresponding values in $\hat{\Phi}^{(n)}$. If this problem is feasible, we get a feasible solution ${\Phi^{(n+1)}}^*$ (Line 18), i.e., the solution to SP1 is found, and the algorithm stops. Otherwise, we set the solution to SP1 as ${\Phi^{(n+1)}}^* = \emptyset$.

b). When SP1 does not have a feasible solution, we increase the V/F levels of each core and link to find a feasible solution, e.g., $c_{i,k}^P = 1 \to c_{i,k+1}^P = 1$ and $c_{i,j,g}^L = 1 \to c_{i,j,g+1}^L = 1$ (Line 20). That is because increased operation frequencies on cores and links can reduce task completion time and improve task reliability to satisfy real-time and reliability constraints.

c). We update the values of $\{\boldsymbol{c^P}, \boldsymbol{c^L}\}$ in SP1, and solve this problem to find the feasible solution ${\Phi^{(n+1)}}^*$ (Line 21). Similarly, if SP1 dose not have a feasible solution, the solution ${\Phi^{(n+1)}}^*$ is set to $\emptyset$.

d). We check whether ${\Phi^{(n+1)}}^* = \emptyset$, in which case there is no feasible solution to SP1. We repeat Step b). and c). to adjust the frequencies on cores and links until the feasible solution ${\Phi^{(n+1)}}^*$ is found, i.e., ${\Phi^{(n+1)}}^* \neq \emptyset$ (Lines 19-22).

By the above FP method, we get the solution to SP1, i.e., $\{\boldsymbol{c^P}^*, \boldsymbol{c^L}^*, \boldsymbol{o}^*, \boldsymbol{ts^P}^*, \boldsymbol{ts^L}^*\}$, but slack time $\hat{\boldsymbol{t}}$ is predetermined.

C. Communication Contention Management

We apply Algorithm 2 to solve the problem of frequency scaling and edge scheduling and get the decisions regarding the core frequency assignment $\boldsymbol{c^P}^*$, link frequency assignment $\boldsymbol{c^L}^*$, task sequence $\boldsymbol{o}^*$, task scheduling time $\boldsymbol{ts^P}^*$, and edge scheduling time $\boldsymbol{ts^L}^*$. In the worst case, they are based on the given slack time $\hat{\boldsymbol{t}}$. During this process, we assume that the communication contention constraints in time are always satisfied, and the slack time $\hat{\boldsymbol{t}}$ is calculated with the minimum link frequency $f_{\min}$. Based on $\{\boldsymbol{x}^*, \boldsymbol{c^L}^*, \boldsymbol{ts^L}^*\}$, we can update the slack time $\boldsymbol{t}^*$ through (5)-(7), and get the extra time slot $\Delta t_{i,j} = \hat{t}_{i,j}^{slack} - t_{i,j}^{slack*}$, ($\hat{t}_{i,j}^{slack} \in \hat{\boldsymbol{t}}, t_{i,j}^{slack*} \in \boldsymbol{t}^*$). Since $\Delta t_{i,j}$ is a non-negative value, this time slot can execute task τ_j or transmit data for the edge $e_{i,j}$. Lowering the operating frequency of the core/link, the system energy consumption can be further reduced. However, when $\Delta t_{i,j}$ is used to transmit task data for the edge $e_{i,j}$, the start and end time of communication, i.e., $ts_{i,j}$ and $te_{i,j}$ will be changed. Thus, the slack time $t_{i,j}^{slack}$ is changed as well due to (5)-(7), which makes the problem difficult to solve. For problem-solving, we use time slot $\Delta t_{i,j}$ for task execution. The details are summarized in Algorithm 3.

a). We calculate the flexible slack time $t_{i,j}^{slack*}$ according to $\{\boldsymbol{x}^*, \boldsymbol{c^L}^*, \boldsymbol{ts^L}^*\}$ for each edge $e_{i,j} \in \mathcal{E}$ through (5)-(7) (Line 1). Based on the calculated slack time $t_{i,j}^{slack*}$ and the predetermined slack time $\hat{t}_{i,j}^{slack}$, we can use extra slot time $\Delta t_{i,j} = \hat{t}_{i,j}^{slack} - t_{i,j}^{slack*}$ to execute task τ_j (Line 2).

b). After completing the communication for edge $e_{i,j}$, task τ_j starts its execution. The initial frequency level used to execute τ_j is set to $k_j^{init} = k$ since ${c_{j,k}^P}^* = 1$ (Line 4).

c). Since $\Delta t_{i,j} \geq 0$ is only used for task execution, the execution time of τ_j is extended from t^P_{j,k^{init}_j} to $t^P_{j,k^{init}_j} + \Delta t_{i,j}$. To satisfy the real-time constraints, the task start time ts_j can be ahead using the extra slot time $\Delta t_{i,j}$, while the task end time te_j remains unchanged. Considering the constraint of execution time for task τ_j, the candidate of frequency level can be given by

$$k^T_j = \arg\max_{k \in \mathcal{N}_p} \{t^P_{j,k} : t^P_{j,k} \leq t^P_{j,k^{init}_j} + \Delta t_{i,j}\}, \quad (38)$$

where t^P_{j,k^{init}_j} denotes the execution time of task τ_j with the frequency f_{p,k^{init}_j}. Besides task execution time, operated frequencies also influence task reliability due to (29). To satisfy the reliability constraints, the candidate frequency level can be set to

$$k^R_j = \arg\min_{k \in \mathcal{N}_p} \{\ln r^P_{j,k} : \ln r^P_{j,k} + \ln R^L_j \geq \ln R_{th}\}, \quad (39)$$

where $R^L_j = \prod_{e_{i,j} \in \mathcal{E}} (\sum_{g \in \mathcal{N}_l} c^L_{i,j,g}{}^* r^L_{i,j,g})$ denotes the reliability probability of communication between τ_j and its predecessors, and $r^P_{j,k}$ denotes the reliability probability of executing τ_j with the frequency $f_{p,k}$. Therefore, to concurrently satisfy the constraints of task execution time and task reliability, the feasible frequency level for task τ_j is set to $\hat{k}_j = \max\{k^R_j, k^T_j\}$ (Line 5).

d). To minimize energy consumption, we determine whether the update of execution frequency is necessary (Line 6). Under the given frequency $f_{p,k}$, the computation energy is $E^{comp}_j(f_{p,k}) = P^{comp}_k t^P_{j,k}$. For task τ_j, if $E^{comp}_j(f_{p,\hat{k}_j}) < E^{comp}_j(f_{p,k^{init}_j})$, we set the execution frequency level as $\hat{k}_j$. Therefore, we update $c^P_{j,k}{}^* = 1$ ($k = \hat{k}_j$) and $c^P_{j,k}{}^* = 0$ ($k \neq \hat{k}_j \in \mathcal{N}_p$) (Line 7). On this basis, we can calculate task start time ts^*_j based on (17) (Line 9). However, if $E^{comp}_j(f_{p,\hat{k}_j}) \geq E^{comp}_j(f_{p,k^{init}_j})$, the frequency of τ_j is unchanged since we aim to reduce energy consumption.

e). We apply the above method for each edge $e_{i,j} \in \mathcal{E}$ to update task execution frequency. Therefore, we can obtain the core frequency assignment $\boldsymbol{c^P}^*$, the start time of executing tasks $\boldsymbol{ts^P}^*$, and the flexible slack time $\boldsymbol{t}^*$.

To sum up, through the above three algorithms: Task Allocation Algorithm 1, Frequency Scaling and Edge Scheduling Algorithm 2, and Communication Contention Algorithm 3, we can solve this task mapping and scheduling problem and obtain the solution $\{\boldsymbol{x}^*, \boldsymbol{c^P}^*, \boldsymbol{c^L}^*, \boldsymbol{o}^*, \boldsymbol{ts^P}^*, \boldsymbol{ts^L}^*, \boldsymbol{t}^*\}$.

D. Algorithm Complexity

CRATMS contains Algorithm 1-Algorithm 3, and the details of their computational complexities are as follows.

a). *Task Allocation*: A heuristic based on dynamic programming is applied in Algorithm 1 to solve the task allocation problem. Firstly, tasks are sorted according to their dependencies through the bubble method. The time complexity is $O(N^2)$ when sorting N tasks [40]. Then, task-to-core assignment is conducted by the dynamic programming method. The number of sub-problems influences the time complexity. Since N tasks are assigned to processors for execution, the task assignment requires N iterations, and the time complexity is $O(N)$. Thus, the time complexity of Algorithm 1 is $O(N^2 + N)$.

b). *Frequency Scaling and Edge Scheduling*: Algorithm 2 performs rounding operations iteratively to improve SP1. The rounding operation is based on the FP method. In the worst case, it needs $n_{\max}$ iterations. If no feasible solution is found within the iterations, we adjust the solution of frequency assignment to obtain a feasible solution. Considering the dimensions of variables in frequency assignment $\boldsymbol{c^P} = [c^P_{i,k}]_{N \times N_p}$ and $\boldsymbol{c^L} = [c^L_{i,j,g}]_{N \times N \times N_l}$, the adjustments of frequency assignment require at most $N_p N + N_l N^2$ times. Hence, the time complexity of Algorithm 2 is $O(n_{\max} + N_p N + N_l N^2)$.

c). *Communication Contention Management*: Algorithm 3 is applied to calculate the precise slack time for each edge (task communication). Since the number of edges is $|\mathcal{E}|$, the slack time calculation needs to perform $|\mathcal{E}|$ times, and thus, the time complexity of Algorithm 3 is $|\mathcal{E}|$.

Therefore, the time complexity of CRATMS is $O((N_l+1)N^2 + (N_p+1)N + n_{\max} + |\mathcal{E}|)$. Note that task number N is usually much larger than the numbers of V/F levels for cores and links (N_p and N_l), i.e., $N \gg N_p$ and $N \gg N_l$. The total time complexity can be simplified as $O(N^2 + N + n_{\max} + |\mathcal{E}|)$.

TABLE III
EXPERIMENTAL SET-UP

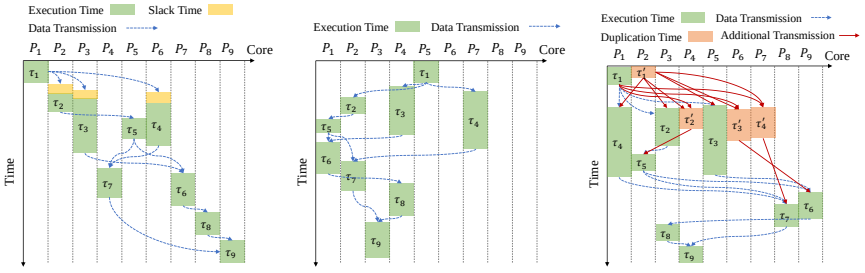
Platform Model	
Platform Size M: $2 \times 2 \rightarrow 5 \times 5$	V/F Levels: $N_p = 5$, $N_l = 5$
Processor Parameters	
Voltage $v_{p,k}$ (V)	0.75, 1.0, 1.3, 1.6, 1.8
Frequency $f_{p,k}$ (MHz)	150, 400, 600, 800, 1000
Power P^{comp}_k (mW)	80, 170, 400, 900, 1600
Link Parameters	
Frequency $f_{l,k}$ (MHz)	200, 400, 600, 800, 1000
Power P^{link}_k (mW)	160, 180, 520, 880, 1600
Bandwidth $b_\omega = 32$ bps	Router Energy Consumption $E_{Rbit} = 0.01$ nJ

Task Model	
Task Set	LU, GE, FFT, LE, MW, MDC, and randomly generated task sets
Task Number $N \in [5, 100]$	Execution Cycles $\omega_i \in [4 \times 10^7, 6 \times 10^8]$
Scheduling Horizon H	Deadline $D_i = \delta(D_{\max} - D_{\min,i}) + D_{\min,i}$
$D_{\max} = H$	$D_{\min,i} = \frac{\omega_i}{f_{p,\max}} + \sum_{e_{j,i} \in \mathcal{E}} \frac{s_{j,i}}{b_\omega f_{l,\max}}$

Tuned Parameters		
Task Reliability: $d = \{2, 3, 4, 5, 6, 7, 8\}$, $R_{th} = \{0.9, 0.95, 0.99, 0.9999\}$		
	Task Deadline Factor δ	Weight Factor μ
Min/Max/Step	0/1.2/0.1	0/1/0.1

V. SIMULATION RESULTS

In this section, we perform extensive experiments to evaluate the performance of the proposed heuristics (CRATMS), compared to the state-of-the-art methods, such as Contention and Energy-aware Task Mapping and Edge Scheduling scheme (CA-TMES) [13], ILP-based heuristics (ILP-HEU) [27], Link Contention-aware Allocation and Scheduling (LCAS) scheme [6], and Task Duplication and Path Selection-based (TDPS) scheme [21]. Specifically, CA-TMES minimizes the makespan and considers the network contention on NoC but ignores the system reliability. ILP-HEU is an energy-aware task mapping scheme without considering system reliability and network contention. LCAS is a link-contention-aware heuristic,



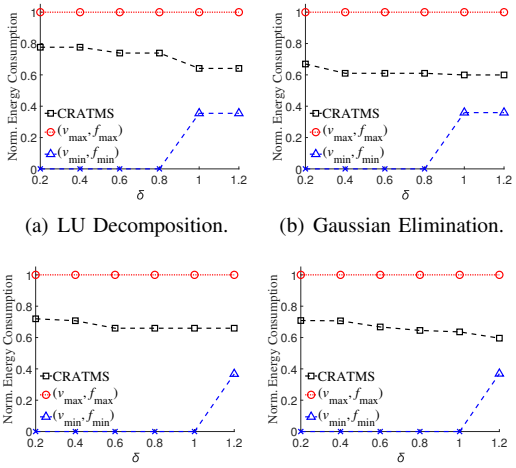
(a) Task mapping of CRATMS. (b) Task mapping of LCAS. (c) Task mapping of TDPS.

Fig. 3. Task allocation and scheduling of LU application when using CRATMS, LCAS, and TDPS in a 3×3 NoC.

which minimizes the contention metrics to reduce communication latency through task mapping, and TDPS is a task duplication-based reliability-aware task deployment scheme.

1) Simulation set-up: We deploy the proposed CRATMS and the other aforementioned methods on a 2D mesh NoC platform with DVFS technology. The NoC parameters are adopted from [14]. We conduct simulations on both randomly generated DAGs (the number of tasks N is changed within the range $[5, 100]$) and realistic application DAGs, such as LU Decomposition (LU) [41], Gaussian Elimination (GE) [41], Fast Fourier Transform (FFT) [41], Laplace Equation (LE) [42], Montage Workflows (MW) [43], and Molecular Dynamic Code (MDC) [41]. For the above DAGs, the number of the worst-case execution cycles ω_i for each task τ_i is assumed within the range $[4 \times 10^7, 6 \times 10^8]$ [44]. In addition, the system parameters during the simulations are considered as follows: the size of NoC platforms (M) is changed from 2×2 to 5×5; the weight of the objective function in problem (27) is set to $\mu \in [0, 1]$; the task deadline is $D_i = \delta(D_{\max} - D_{\min,i}) + D_{\min,i}$ [45], where $D_{\max} = H$ is the maximum task deadline (scheduling horizon), $D_{\min,i}$ is the minimum execution and communication time of task τ_i, and $\delta \in [0, 1.2]$ is a tuned deadline factor; the transient fault sensitivity and the reliable threshold are set to $d = \{2, 3, 4, 5, 6, 7, 8\}$ and $R_{th} = \{0.9, 0.95, 0.99, 0.9999\}$, respectively; the fault-tolerant model follows Poisson Distribution with the average fault rate $\lambda_0 = 10^{-6}$ [17], [35], [36]. The system parameters are summarized in Table III. Note that the values of these parameters do not affect the structure of the task mapping problem. Therefore, our proposed method is still applicable to other system parameters. The simulation is carried out on a computer with an Intel i7 processor operating at 2.30 GHz, and the algorithms are implemented in Matlab R2021b.

2) Simulation Results: Firstly, we illustrate the effectiveness of our method in terms of the joint optimization of energy, reliability, and network contention, compared with LCAS and TDPS. We set $\delta = 0.5$, $\mu = 0.5$, $d = 6$, and $R_{th} = 0.99$. Fig. 3 shows the task mapping results of CRATMS, LCAS, and TDPS when executing a realistic application, LU Decomposition ($N = 9$), on a 3×3 NoC platform. From it, we can see that under the real-time and reliability constraints, the total energy consumption of CRATMS ($E_{total} = 3.9\ J$) is less than LCAS and TDPS, which are $5.1\ J$ and $4.6\ J$, respectively. This is because LCAS sets all the cores/routers that operate at the same frequency level, while TDPS duplicates all the tasks to improve reliability, and thus, more energy is consumed to execute the tasks. As the results shown in Fig. 3(a) and Fig. 3(b), there is



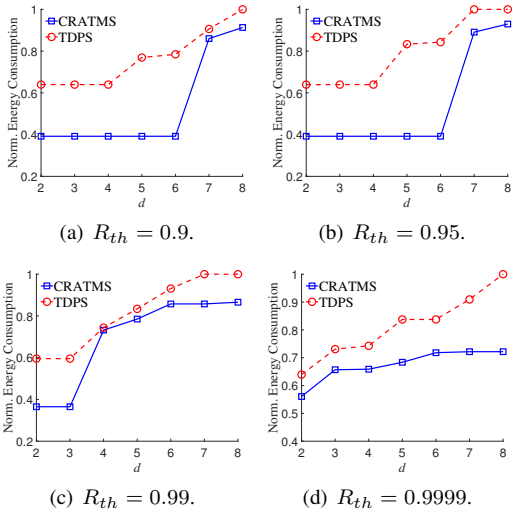
(a) LU Decomposition. (b) Gaussian Elimination.

(c) Laplace Equation. (d) Montage Workflow.

Fig. 4. Evaluations of CRATMS with flexible and fixed V/F levels: effects of deadline parameter δ.

no network contention using CRATMS and LCAS. To alleviate network contention, CRATMS considers both flexible slack time $t_{i,j}^{slack}$ in (7) and task mapping from the aspects of time and space to optimize task and edge scheduling. However, LCAS only uses task mapping to reduce network contention, and thus, it consumes more energy than CRATMS. On the other hand, Fig. 3(c) shows that there exists network contention during the task mapping process of TDPS, which is due to task duplication, e.g., the contentions among the tasks $\tau_1 \rightarrow \tau_2$, $\tau_1 \rightarrow \tau_2'$, $\tau_1 \rightarrow \tau_3$, $\tau_1 \rightarrow \tau_3'$, $\tau_1 \rightarrow \tau_4$, and $\tau_1 \rightarrow \tau_4'$, where τ_i' represents the duplication of τ_i. The above results show that the proposed CRATMS can simultaneously reduce energy consumption and network contention under reliability and real-time constraints.

Fig. 4 evaluates the influence of time factor δ on energy consumption and problem feasibility using the CRATMS method. We schedule the LU ($N = 9$), GE ($N = 14$), LE ($N = 16$), and MW ($N = 24$) applications with flexible and fixed V/F levels, such as $(v_{\max}, f_{\max})$ and $(v_{\min}, f_{\min})$, on a 3×3 NoC platform. In addition, the parameters are set to $\mu = 0.5$, $d = 6$, $R_{th} = 0.99$, and δ is varied from 0.2 to 1.2 with the step of 0.2. As shown in Fig. 4, CRATMS with flexible V/F level can lower the operation frequency to reduce energy consumption while still satisfying the real-time and reliability constraints. Therefore, under the given time factor δ, it has a smaller energy consumption, compared to CRATMS with the fixed V/F level $(v_{\max}, f_{\max})$. In addition, with the value of δ increasing, the energy consumption of CRATMS decreases. This is because tasks with tight deadlines need to be executed



(a) $R_{th} = 0.9$. (b) $R_{th} = 0.95$. (c) $R_{th} = 0.99$. (d) $R_{th} = 0.9999$.

Fig. 5. Evaluation of CRATMS and TDPS under the Montage Workflow application: effects of reliability parameters d and R_{th}.

with a high V/F level to satisfy deadline constraints, which increases energy consumption. In Fig. 4, the symbol $\times$ denotes no feasible solution to the task mapping problem, and the energy consumption is set to zero. When the V/F levels are set to small values, CRATMS cannot meet real-time constraints for all tasks.

Fig. 5 and Fig. 6 compare the energy consumption of CRATMS and TDPS with the different transient fault sensitivities d and reliability thresholds R_{th}. In Fig. 5, we execute the MW application ($N = 24$) on a 3×3 NoC platform, and we set $\delta = 0.5$, $\mu = 0.5$, $d = \{2, 3, 4, 5, 6, 7, 8\}$, and $R_{th} = \{0.9, 0.95, 0.99, 0.9999\}$. Note that with parameter d increasing, the reliabilities of task execution and data transmission decrease. Therefore, the NoC platform needs to increase the V/F level to enhance task reliability with the high values of d and R_{th}, increasing energy consumption. In Fig. 6, we compare CRATMS and TDPS under different d and R_{th} parameters. During this process, we randomly generate 50 sets of DAGs, where the number of tasks N is varied within the range $[5, 50]$. The energy saving ratio is defined as $(E_{total}^{TDPS} - E_{total}^{CRATMS})/E_{total}^{TDPS} \times 100\%$, where E_{total}^{TDPS} and E_{total}^{CRATMS} are the total energy consumptions of TDPS and CRATMS, respectively. Since TDPS mainly uses task duplication to improve reliability, compared to TDPS, CRATMS has 52.17%, 48.23%, 39.43%, and 18.55% average lower energy consumption with $d = 2, 4, 6, 8$. When the value of parameter d is small, the system reliability is high. Therefore, CRATMS applies a low V/F level to satisfy reliability constraints and reduce energy consumption. However, with the values of parameters d and R_{th} increasing, the system reliability will become lower and the reliability requirement will increase. To satisfy the reliability constraints, CRATMS needs to increase the V/F levels, leading to a decline in the energy-saving ratio.

Fig. 7 evaluates the effectiveness of reducing network contention using CRATMS under different NoC scales. We consider GE ($N = 14$), LE ($N = 16$), FFT ($N = 15$), MW ($N = 24$), and MDC ($N = 41$) applications, and the NoC size is changed from 2 to 5×5. Moreover, we set $\delta = 0.5$, $\mu = 0.5$, $d = 6$, and $R_{th} = 0.99$. To evaluate the communication contention on the target platform, we introduce an average

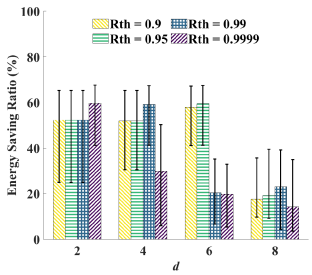
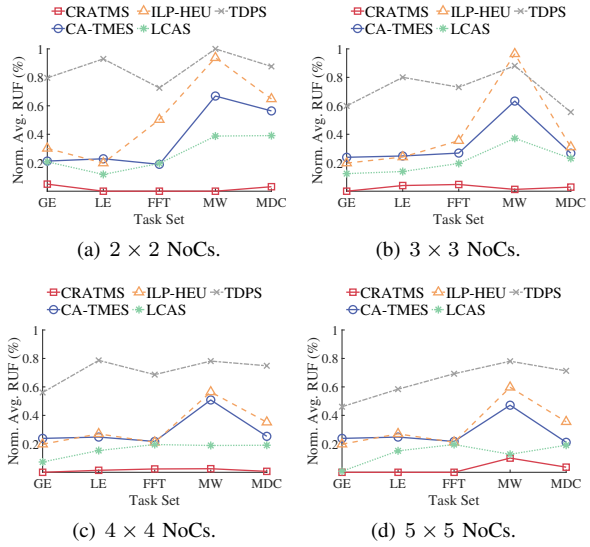


Fig. 6. Average energy saving ratio comparison of CRATMS and TDPS under different reliability parameters d and R_{th}.

Route Utilization Factor (RUF) for the tasks [6]. For the edge $e_{i,j}$ between tasks τ_i and τ_j, the RUF is defined as $RUF(e_{i,j}) = \frac{1}{|\mathcal{L}_{s,d}|} \sum_{l_{m,n} \in \mathcal{L}_{s,d}} \frac{|LOT_{l_{m,n}}^{e_{i,j}} \cap Z_{l_{m,n}}|}{|LOT_{l_{m,n}}^{e_{i,j}}|}$, where $LOT_{l_{m,n}}^{e_{i,j}}$ is the occupation time of link $l_{m,n}$ during the data transferring of the edge $e_{i,j}$, and $Z_{l_{m,n}}$ is the set of time slots when the link $l_{m,n}$ is occupied during the communication. Note that $LOT_{l_{m,n}}^{e_{i,j}} \cap Z_{l_{m,n}}$ indicates the slot of contention time of the link $l_{m,n}$, when data transmission of edge $e_{i,j}$ conflicts with other edges. $RUF(e_{i,j})$ represents the average ratio of the total contention time to the transmission time for the edge $e_{i,j}$. The lower the RUF, the less contention of the edge $e_{i,j}$. As the results depicted in Fig. 7, under the given application and NoC platform, the average RUF of CRATMS is the minimum and is close to zero. This is because CRATMS introduces flexible slack time and assigns the tasks to the cores with less route overlap, leading to reducing communication contention as much as possible. In contrast, ILP-HEU and TDPS omit communication contention. Moreover, CA-TMES mainly focuses on fixed slack time, which increases the communication time for the edges and makes real-time constraints hard to satisfy. At the same time, LCAS did not take slack time into account and only used task mapping to reduce task overlapping on routing links.

Fig. 8 compares the solution quality (e.g., objective function value, energy consumption, and computation time) achieved by the CRATMS and optimal method. Since task mapping problem (27) can be transformed into an MILP problem, the optimization solver, such as Gurobi, can be used to solve this problem and get the optimal solution. The parameters are set to $\delta = 0.5$, $\mu = 0.5$, $d = 6$, and $R_{th} = 0.99$. In Fig. 8, we perform LU ($N = 9$), GE ($N = 14$), FFT ($N = 15$), LE ($N = 16$) and MW ($N = 24$) applications on a 3×3 NoC platform. The results show that compared to the optimal solution, CRATMS has 24.8% higher energy consumption and 18.7% higher objective function value on average. However, the computation time of CRATMS is much less than that of the optimal method and almost varies linearly with the number of tasks. Therefore, CRATMS can achieve a trade-off between result quality and runtime.

Fig. 9(a)–Fig. 9(c) evaluate the energy consumption, reliability, and network contention of CRATMS, compared to ILP-HEU, TDPS, CA-TMES, and LCAS. We randomly generate ten DAGs, where the number of tasks N is changed from 10 to 100 with a step of 10. Compared to ILP-HEU and CA-TMES, CRATMS has an average 140% and 138% higher energy consumption, respectively. However, the reliabilities of ILP-HEU and CA-TMES are lower than the threshold $R_{th} = 0.99$, and ILP-HEU and CA-TMES have higher average RUF than



(a) 2×2 NoCs. (b) 3×3 NoCs.

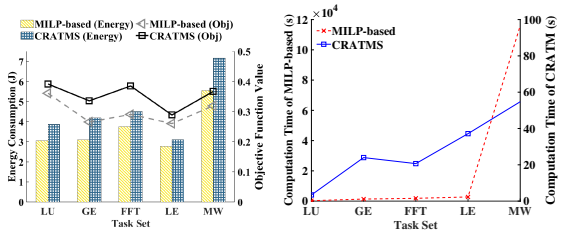
(c) 4×4 NoCs. (d) 5×5 NoCs.

Fig. 7. Comparison of the average RUF of different task mapping methods with the different NoC platform scale.

CRATMS. This is because ILP-HEU did not consider system reliability and network contention, while CA-TMES uses fixed slack time to alleviate the contention and ignore system reliability. Moreover, compared to LCAS and TDPS, where the reliability constraints are satisfied, CRATMS has 31.6% and 21.7% lower energy consumption, and 95.5% and 98.6% lower RUF on average, respectively. This is because LCAS minimizes the contention metric RUF through task-to-core allocation. It only considers the overlapping of routing links and omits the improvement of energy efficiency. In addition, TDPS uses task duplication to improve system reliability, leading to increased energy consumption and network contention. Since the flexible slack time and DVFS are optimized in task mapping, CRATMS has higher energy efficiency and lower network contention. Therefore, it is more suitable for large-scale and complex task scenarios. Fig. 9(d) evaluates the performances of CRATMS with and without DVFS switching overheads, and the parameters of switching overheads are adopted from [14]. The increment ratio of energy consumption is $(E_{total}^{switch} - E_{total})/E_{total}$, where E_{total}^{switch} and E_{total} are the total energy consumption with and without switching overhead, respectively. Similarly, we can get the increment ratio of the objective function value in PP (27). Fig. 9(d) shows that the increment ratios of energy consumption and objective function value are $0.1‰$ and $0.09‰$ on average, respectively. Moreover, the ratio of switching time to task execution and communication time is $0.09‰$ on average. Therefore, the energy and time overheads of frequency switching are much smaller than that of task execution and communication. This result is in line with the overhead analysis in [6], [29], [34]. Thus, the switching overhead can be omitted.

VI. Conclusion

To minimize energy consumption, while balancing workload under real-time and reliability constraints, we formulate the task mapping problem on the NoC-based MPSoCs platform as an MINLP problem. On the one hand, the frequency scalings of both cores and links are considered simultaneously to improve task reliability and energy efficiency. On the other hand, load balancing and flexible slack time are introduced into the task allocation and scheduling process to alleviate network contention and increase communication efficiency while satisfying real-time constraints. Based on the structure of the task mapping problem, we propose three-step heuristics (CRATMS), including task allocation, frequency scaling and edge scheduling, and network contention management, to solve this problem efficiently. Finally, we compare CRATMS with the existing contention-aware and reliability-aware methods, and the results show that the proposed method can reduce energy consumption and network contention under multiple constraints. Therefore, CRATMS can be applied to applications with high energy efficiency, fault-tolerant, and real-time requirements. In future work, we will extend our method to heterogeneous NoC platforms considering both permanent and transient faults.



(a) Energy consumption and objective function value. (b) Computation time.

Fig. 8. Energy consumption, objective function value, and running time comparison of MILP-based method and CRATMS.

Acknowledgment

This research was partly supported by the National Key Research and Development Program of China under Grant 2022YFF0902800, and the National Natural Science Foundation of China under Grants 61973163, 62303109, and 62374031.

References

[1] Z. Liao, Y. Ma, J. Huang, J. Wang, and J. Wang, "HOTSPOT: A UAV-assisted dynamic mobility-aware offloading for mobile-edge computing in 3-D space," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10 940–10 952, 2021.

[2] J. Wang, Y. Yang, T. Wang, R. S. Sherratt, and J. Zhang, "Big data service architecture: A survey," *J. Internet Technol.*, vol. 21, no. 2, pp. 393–405, 2020.

[3] C. Liu, K. Li, K. Li, and R. Buyya, "A new service mechanism for profit optimizations of a cloud provider and its users," *IEEE Trans. on Cloud Comput.*, vol. 9, no. 1, pp. 14–26, 2017.

[4] J. Zhang, S. Zhong, T. Wang, H.-C. Chao, and J. Wang, "Blockchain-based systems and applications: a survey," *J. Internet Technol.*, vol. 21, no. 1, pp. 1–14, 2020.

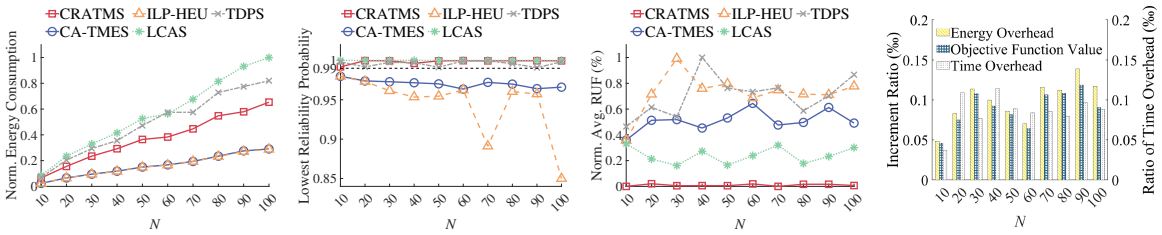
[5] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–38, 2013.

[6] S. Paul, N. Chatterjee, and P. Ghosal, "Dynamic task allocation and scheduling with contention-awareness for Network-on-Chip based multicore systems," *J. Syst. Archit.*, vol. 115, pp. 1–16, 2021.

[7] K. Goossens, J. Dielissen, and A. Radulescu, "Aethereal network on chip: concepts, architectures, and implementations," *IEEE Des. Test.*, vol. 22, no. 5, pp. 414–421, 2005.

[8] I. Swarbrick, D. Gaitonde, S. Ahmad, B. Gaide, and Y. Arbel, "Network-on-chip programmable platform in VersalTM ACAP architecture," in *ACM International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 212–221.

[9] J. Han, M. Choi, and Y. Kwon, "40-TFLOPS artificial intelligence processor with function-safe programmable many-cores for ISO26262 ASIL-D," *ETRI J.*, vol. 42, no. 4, pp. 468–479, 2020.



(a) Energy comparison. (b) Task reliability comparison. (c) Average RUF comparison. (d) Overhead comparison.

Fig. 9. Energy consumption, task reliability, average RUF, and switching overhead comparisons of various task mapping methods with different task DAGs.

[10] S. Moulik, Z. Das, R. Devaraj, and S. Chakraborty, "SEAMERS: A semi-partitioned energy-aware scheduler for heterogeneous multicore real-time systems," *J. Syst. Archit.*, vol. 114, pp. 101 953:1–101 953:11, 2021.

[11] Y. Sharma and S. Moulik, "CETAS: a cluster based energy and temperature efficient real-time scheduler for heterogeneous platforms," in *ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 501–509.

[12] Y. Sharma, S. Chakraborty, and S. Moulik, "ETA-HP: an energy and temperature-aware real-time scheduler for heterogeneous platforms," *J. Supercomput.*, vol. 78, no. 8, pp. 1–25, 2022.

[13] J. Han, M. Lin, D. Zhu, and L. T. Yang, "Contention-aware energy management scheme for NoC-based multicore real-time systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 691–701, 2014.

[14] D. Li and J. Wu, "Energy-efficient contention-aware application mapping and scheduling on NoC-based MPSoCs," *J. Parallel Distrib. Comput.*, vol. 96, pp. 1–11, 2016.

[15] H. Ali, U. U. Tariq, Y. Zheng, X. Zhai, and L. Liu, "Contention & energy-aware real-time task mapping on NoC based heterogeneous MPSoCs," *IEEE Access*, vol. 6, pp. 75 110–75 123, 2018.

[16] U. U. Tariq, H. Wu, and S. Abd Ishak, "Energy-aware scheduling of conditional task graphs on NoC-based MPSoCs," in *International Conference on System Sciences*, 2018, p. 5707–5716.

[17] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *IEEE/ACM International Conference on Computer-Aided Design*, 2004, pp. 35–40.

[18] C. Wu, C. Deng, L. Liu, J. Han, J. Chen, S. Yin, and S. Wei, "An efficient application mapping approach for the co-optimization of reliability, energy, and performance in reconfigurable NoC architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 8, pp. 1264–1277, 2015.

[19] C. Wu, C. Deng, L. Liu, J. Han, J. Chen, S. Yin, and S. Wei, "A multi-objective model oriented mapping approach for NoC-based computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 662–676, 2016.

[20] N. Chatterjee, S. Paul, and S. Chattopadhyay, "Task mapping and scheduling for network-on-chip based multi-core platform with transient faults," *J. Syst. Archit.*, vol. 83, pp. 34–56, 2018.

[21] L. Mo, Q. Zhou, A. Kritikakou, and J. Liu, "Energy efficient, real-time and reliable task deployment on NoC-based multicores with DVFS," in *IEEE Design, Automation and Test in Europe*, 2022, pp. 1347–1352.

[22] M. Mandelli, L. Ost, G. Sassatelli, and F. Moraes, "Trading-off system load and communication in mapping heuristics for improving NoC-based MPSoCs reliability," in *International Symposium on Quality Electronic Design*, 2015, pp. 392–396.

[23] C. Liu, K. Li, and K. Li, "A game approach to multi-servers load balancing with load-dependent server availability consideration," *IEEE Trans. on Cloud Comput.*, vol. 9, no. 1, pp. 1–13, 2018.

[24] K. Li, W. Yang, and K. Li, "Performance analysis and optimization for SpMV on GPU using probabilistic modeling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 196–205, 2014.

[25] J. He, Y. Xiao, C. Bogdan, S. Nazarian, and P. Bogdan, "A design methodology for energy-aware processing in unmanned aerial vehicles," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 27, no. 1, pp. 1–20, 2021.

[26] H. R. Mendis, N. C. Audsley, and L. S. Indrusiak, "Dynamic and static task allocation for hard real-time video stream decoding on NoCs," *Leibniz Trans. Embed. Comput. Syst.*, vol. 4, no. 2, pp. 01:1–01:25, 2017.

[27] S. Abd Ishak, H. Wu, and U. U. Tariq, "Energy-aware task scheduling on heterogeneous NoC-based MPSoCs," in *IEEE International Conference on Computer Design*, 2017, pp. 165–168.

[28] A. Namazi, M. Abdollahi, S. Safari, and S. Mohammadi, "A majority-based reliability-aware task mapping in high-performance homogenous NoC architectures," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 1, pp. 1–31, 2017.

[29] Y. Sharma and S. Moulik, "FATS-2TC: A fault tolerant real-time scheduler for energy and temperature aware heterogeneous platforms with two types of cores," *Microprocess. Microsyst.*, vol. 96, pp. 104 744:1–104 744:9, 2023.

[30] T. Maqsood, K. Bilal, and S. A. Madani, "Congestion-aware core mapping for network-on-chip based systems using betweenness centrality," *Future Gener. Comput. Syst.*, vol. 82, pp. 459–471, 2018.

[31] L. Yang, W. Liu, W. Jiang, M. Li, J. Yi, and E. H.-M. Sha, "Application mapping and scheduling for network-on-chip-based multiprocessor system-on-chip with fine-grain communication optimization," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 24, no. 10, pp. 3027–3040, 2016.

[32] O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan, "UNISM: Unified scheduling and mapping for general networks on chip," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 20, no. 8, pp. 1496–1509, 2011.

[33] Q. Le, G. Yang, W. N. Hung, X. Song, and F. Fan, "Performance-driven assignment and mapping for reliable networks-on-chips," *J. Zhejiang Univ. - Sci. C*, vol. 15, no. 11, pp. 1009–1020, 2014.

[34] R. Yan, Y. Zhou, A. Cai, C. Li, Y. Yan, and M. Yin, "Contention-aware mapping and scheduling optimization for NoC-based mpsocs," in *International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 305–313.

[35] P. Pop, K. H. Poulsen, V. Izosimov, and P. Eles, "Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems," in *IEEE/ACM International Conference on Hardware/software Codesign and System Synthesis*, 2007, pp. 233–238.

[36] Y. Guo, D. Zhu, and H. Aydin, "Reliability-aware power management for parallel real-time applications with precedence constraints," in *International Green Computing Conference and Workshops*, 2011, pp. 1–8.

[37] B. Acun, K. Chandrasekar, and L. V. Kale, "Fine-grained energy efficiency using per-core DVFS with an adaptive runtime system," in *International Green and Sustainable Computing Conference*, 2019, pp. 1–8.

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

[39] L. Mo, A. Kritikakou, and O. Sentieys, "Controllable QoS for imprecise computation tasks on DVFS multicores with time and energy constraints," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 8, no. 4, pp. 708–721, 2018.

[40] O. Astrachan, "Bubble sort: an archaeological algorithmic analysis," *ACM SIGCSE bulletin*, vol. 35, no. 1, pp. 1–5, 2003.

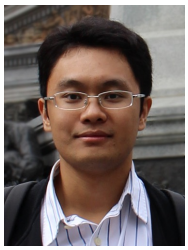
[41] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, 2002.

[42] 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, 2013.

[43] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Sci. Program.*, vol. 13, no. 3, pp. 219–237, 2005.

[44] J. Zhou, J. Yan, T. Wei, M. Chen, and X. S. Hu, "Energy-adaptive scheduling of imprecise computation tasks for QoS optimization in real-time MPSoC systems," in *IEEE Design, Automation and Test in Europe*, 2017, pp. 1402–1407.

[45] X. Li, L. Mo, A. Kritikakou, and O. Sentieys, "Approximation-aware task deployment on heterogeneous multicore platforms with DVFS," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 7, pp. 2108–2121, 2023.



Lei Mo (S'13–M'17) is currently an associate professor with the School of Automation, Southeast University, Nanjing, China. He received the B.S. degree from the College of Telecom Engineering and Information Engineering, Lanzhou University of Technology, Lanzhou, China, in 2007, and the Ph.D. degree from the College of Automation Science and Engineering, South China University of Technology, Guangzhou, China, in 2013. From 2013 to 2015, he was a research fellow with the Department of Control Science and Engineering, Zhejiang University, China. From 2015 to 2017, he was a research fellow with INRIA Nancy–Grand Est, France. From 2017 to 2019, he was a research fellow with INRIA Rennes–Bretagne Atlantique, France. His current research interests include networked estimation and control in wireless sensor and actuator networks, cyber-physical systems, task mapping and resource allocation in embedded systems. He serves as an Associate Editor for KSII Transactions on Internet and Information Systems, and International Journal of Ad Hoc and Ubiquitous Computing, and a TPC Member for several international conferences.

Xinmei Li received her B.S. degree in automation engineering from Sichuan University, Chengdu, China, in 2021. She is currently working towards the M.S. degree in Control Science and Engineering at Southeast University, Nanjing, China. Her current research interests include embedded and real-time systems, and network-on-chip.

Angeliki Kritikakou is currently an Associate Professor at University of Rennes 1 and IRISA - INRIA Rennes research center. She received her Ph.D. in 2013 from the Department of Electrical and Computer Engineering at University of Patras, Greece and in collaboration with IMEC Research Center, Belgium. She worked for one year as a Postdoctoral Research Fellow at the Department of Modelling and Information Processing (DTIM) at ONERA in collaboration with Laboratory of Analysis and Architecture of Systems (LAAS) and the University of Toulouse, France. Her research interests include embedded systems, real-time systems, mixed-critical systems, hardware/software co-design, mapping methodologies, design space exploration methodologies, memory management methodologies, low power design, and fault tolerance.

Xiaojun Zhai (SM'21) is currently a Reader in the Embedded Intelligent Systems Laboratory at the University of Essex. He has authored/co-authored over 140 scientific papers in international journals and conference proceedings. His research interests mainly include the design and implementation of the digital image and signal processing algorithms, custom computing using FPGAs, embedded systems, and hardware/software co-design.