Energy-Efficient, Reliable and QoS-Aware Task Mapping on Cyber-Physical Systems

Lei Mo, Angeliki Kritikakou, Xinmei Li

Energy-Efficient, Reliable and QoS-Aware Task Mapping on Cyber-Physical Systems

Lei Mo[1], Angeliki Kritikakou[2], and Xinmei Li[1]
[1] School of Automation, Southeast University, Nanjing 210096, China.
E-mails: lmo@seu.edu.cn, lixinmei1999@foxmail.com
[2] University of Rennes, INRIA, IRISA, CNRS, Rennes 35042, France.
E-mail: angeliki.kritikakou@irisa.fr

Abstract

Cyber-Physical Systems (CPS) usually consist of a set of embedded systems (CPS nodes) connected through wireless communication, providing multiple functionalities that support different types of applications. During CPS deployment, application tasks are mapped on the CPS nodes with the objective of enhancing real-time performance, energy efficiency, and execution reliability. To satisfy these requirements, effective task mapping approaches should be designed based on different types of tasks, platforms, application and system requirements. In this paper, we provide a comprehensive survey regarding the task mapping methods in CPS.

1 Introduction

Embedded systems can support diverse functionalities on a tiny platform, e.g., data collection and processing, wireless communication. With the characteristics of simple structure, high degree of customization, low cost and low power consumption, embedded systems have became the critical part of networked systems, such as Cyber-Physical Systems (CPS). They are widely used in the wireless nodes, such as sensors, actuators or controllers. The system applications, including control, sensing, data processing and date transmission, contain multiple dependent tasks. These tasks can be mapped on the wireless nodes so as to achieve the desired system performance. Following the "Fog/Edge-computing" model, instead of collecting and sending all data to a remote Base Station (BS), a part of the data processing is done on the wireless nodes, and thus, only a small part of pre-processed data is sent to the BS. As a result, the use of system resources can be optimized.

During the task mapping process, i.e., task allocation and task scheduling, the constraints related to energy consumption and real-time execution should be taken into account. This is because, most of the wireless nodes have limited energy budget, especially for the energy-harvesting or battery-powered devices. In addition, real-time responsiveness is required by many applications, e.g., mobile target tracking, as missing task deadline can have serious consequences. With a proper task mapping scheme, the tasks can be executed in parallel on various nodes so as to improve the real-time execution. On this basis, by employing energy efficiency methods, such as Imprecise Computation (IC), Dynamic Voltage and Frequency Scaling (DVFS), and Dynamic Power Management (DPM), the time and the energy consumed to execute the tasks can be optimized. However, DVFS will influence the reliability of task execution. Usually, the higher is the frequency to execute the tasks, the higher is the task reliability.

Energy efficiency, task deadline and reliability are important, but these are conflicting objectives, since enhancing real-time execution and task reliability often require to consume more energy. Several methods have been proposed to balance these requirements. In the rest of the paper, we will first provide the main preliminaries for system model used in task mapping, and then, give a comprehensive survey of the state-of-the-art.

2 System Model

In this section, we describe the typical characteristics of the system model used for task mapping. Table 5 links the task and platform model with the corresponding variables used in the task mapping methods.

Table 5: Link of main task and platform characteristics with task mapping variables.

Variables	*Task model*		*Platform model*					
	IC	Dependent	Multi-core	DVFS	DPM	Reliability	Migration	Network
Task frequency				√	√	√		
Task allocation			√					√
Task sequence		√						
Task start time					√			
Task adjustment	√							
Task duplication						√		
Task partition							√	
Node communication								√

2.1 Task Model

Tasks can be modeled as imprecise computation (IC) tasks and precise computation tasks, based on their characteristics [1]. In the IC model, a task can be logically divided into: 1) a *mandatory* subtask, which guarantees the basic QoS, and 2) an *optional* subtask, which further improves QoS. Both the mandatory subtask and the optional subtask must be completed before the task's deadline to generate a correct and in-time result, and the optional subtask is executed after the mandatory subtask. By executing the mandatory subtask, we obtain the basic Quality of Service (QoS). When the system resources are available, the optional subtask can be executed. The longer the optional subtask is executed, the better is the QoS of the result. Compared with the IC tasks, the precise computation tasks can be considered as a special case of IC tasks, where the complete task corresponds to the mandatory subtasks, while the optional subtask is empty.

The characteristics of an IC task τ_i can be described by a tuple $\{o_i, M_i, O_i, t_i^s, D_i, T_i\}$ [2], where M_i is the mandatory subtask, o_i is the optional subtask and it has an upper bound O_i, i.e., $0 \leq o_i \leq O_i$. M_i and O_i are measured in Worst Case Execution Cycles (WCECs). t_i^s, D_i and T_i are the start time, the deadline and the period of task τ_i, respectively. During the task mapping process, o_i and t_i^s are the optimization variables since they influence task scheduling decision. A real-time application is usually consisting of a set of N dependent tasks $\{\tau_1, \ldots, \tau_i, \ldots, \tau_N\}$. They can be described by a Directed Acyclic Graph (DAG) $G(\mathscr{V}, \mathscr{E})$, where vertexes $\mathscr{V}$ represent the set of tasks to be executed, while edges $\mathscr{E}$ represent the data dependencies between the tasks. The dependency between the tasks can be further described by an $N \times N$ binary matrix $\boldsymbol{S}$, where $s_{ij} = 1$ represents task τ_i precedes task τ_j, i.e., τ_j starts after the end time of τ_i, otherwise, $s_{ij} = 0$. Since the tasks are dependent, the adjustment of task start time t_i^s and t_j^s is restricted by the task sequence s_{ij}.

Based on the IC task, the QoS function provides the obtained QoS based on the number of execution cycles of the optional subtask. Usually, it can be formulated as: 1) *Linear* function, e.g., $\sum_{i=1}^{N}(a_i o_i + b_i)$ [3]; or 2) *Concave* function, e.g., $\sum_{i=1}^{N}(\alpha_i o_i + \beta_i \sqrt{o_i} + \sqrt{3}\gamma_i o_i)$ [4]. The linear function models the case where the system QoS increases uniformly during the optional subtask execution, while the concave function addresses the case where the increase of QoS exhibits a continuously nondecreasing rate as the optional subtask execution goes on. Linear and general concave functions are considered as the most realistic and typical QoS representation in the literature [5], since they adequately capture the behavior of many application areas, such as image and speech processing, control engineering, and automatic target recognition.

2.2 Platform Model

With the increasing requirements of high performance computation, low energy consumption and low task execution delay, *single-core* embedded systems are insufficient for data intensive applications, such as multimedia. High

performance computation and low task execution delay usually require more energy consumption. To balance these *contradictory* requirements, *multi-core* embedded systems are used. Multi-core platforms allow the computations to be split and assigned to multiple processors, and each processor can run at a lower voltage and frequency. Compared to a single-core system, this result has a higher energy and time efficiency. For example, the fire detection wireless node MiLive-v2 [6] is equipped with three processor types: 1) a 8-bit low-power AVR processor (ATmega128rfa1), which can be used to run simple tasks; 2) a 32-bit powerful ARM processor (ARM1176JZF), which can be used to run more complicated tasks; and 3) a Digital Signal Processor (DSP) unit, which can be specially used for image processing.

Dynamic Voltage and Frequency Scaling (DVFS) and *Dynamic Power Management* (DPM) are two effective methods to improve energy efficiency of task execution [7]. As long as the resource and application constraints (e.g., task deadline) allow, the methods with DVFS/DPM achieve significant energy reductions. DVFS is able to adjust the supply voltage/frequency of the processor during the task execution process, and thus, the time and the energy required to execute the tasks can be optimized. The power consumption of a processor θ_k is expressed as $P_k^c = P_k^s + P_k^d$ [7], where $P_k^s = C_k^s v_k^{\rho_k}$ is the *static* power of the processor ready to execute (being either on the active or idle mode), $P_k^d = C_k^d f_k v_k^2$ is the *dynamic* power of task execution. C_k^s, ρ_k and C_k^d are constants depending on the type of processor. By lowering the supply voltage/frequency (v_k, f_k), quadratic savings in energy consumption can be achieved. Based on the adjustment manner, DVFS can be classified as: 1) continuous DVFS [4]: the voltage can be changed within the range $[V_{\min}, V_{\max}]$; and 2) discrete DVFS [2]: the processor can select L different voltage/frequency levels $\{(v_1, f_1), \ldots, (v_l, f_l), \ldots, (v_L, f_L)\}$. On the other hand, based on the length of duration, DVFS can be classified as: 1) inter-task DVFS [2]: the voltage/frequency of processor stays constant during the execution of a task; 2) intra-task DVFS [8]: the voltage/frequency of processor can be changed during the execution of a task.

When the assigned tasks are finished, the processor will switch from the *active* mode to the *idle* mode. In addition, when the idle interval of the processor is longer than a certain threshold T_{th} (called break-even time), the processor will turn into the *sleep* mode. The transition time and energy overhead is very small compared to the time and energy required to complete a task. Such overheads are typically incorporated into the execution time and energy of the task [9]. According to the start time t_i^s and the end time t_i^e of each task τ_i, i.e., to adjust the idle interval, the processor can directly switch from the active model to the sleep mode through the DPM. Since sleep mode consumes less energy than idle mode, DPM can further reduce the energy consumption of task execution.

Although the energy efficiency of task execution can be enhanced through the DVFS, DVFS has a negative impact on reliability, mainly due to the increased transient fault rates at low supply voltage/frequency levels. Usually, the reliability follows a Poisson distribution model [10]. When a processor θ_k uses voltage/frequency level (v_k, f_k) to execute a task τ_i with C_i cycles, the reliability of task execution is

$$R_{ik} = e^{-\lambda \times 10^{\frac{d(f_{\max} - f_k)}{f_{\max} - f_{\min}}} \times \frac{C_i}{f_k}}, \tag{16}$$

where $f_{\max} = \max\{f_1, \ldots, f_L\}$ and $f_{\min} = \min\{f_1, \ldots, f_L\}$, λ and d are the constants related to fault rate and sensitivity. Eq. (16) shows that the higher frequency used to execute the tasks, the higher reliability can be obtained. To improve task reliability, besides DVFS, task replication can be also used. For instance, by applying selective task duplication, task τ_i is duplicated when its execution reliability is lower than a given threshold R_{th}. Then, task τ_i and its duplicated task are executed on a different processor, since it is unlikely that the execution of both original and duplicated tasks on different processors fails [11]. Therefore, if the reliabilities of original and duplicated tasks are R_{im} and R_{in}, respectively, the total reliability of task τ_i becomes $R_i = 1 - [1 - R_{im}][1 - R_{in}]$.

Based on the processor's characteristics, multi-core embedded systems can be divided into: 1) *homogeneous* platform, and 2) *heterogeneous* platform. For the homogeneous platform, the processors are the same, and thus, they have the same frequency characteristics (e.g., minimal, maximal and operating frequencies). However, for the heterogeneous platform, the processors are divided into several clusters, where each cluster consists of a set of symmetric processors that have the same frequency characteristics. Since the processors of heterogeneous platform have different voltage/frequency levels, a task execution efficiency $\lambda_{ik} \in (0, 1]$ is usually introduced to describe the task execution efficiency (i.e., the heterogeneity) [9]. Correspondingly, the Worst Case Execution Time (WCET) of task τ_i, when it is executed on processor θ_k, is calculated as $\frac{C_i}{f_k \lambda_{ik}}$. In addition, some heterogeneous platforms,

e.g., ARM big.LITTLE platform [12, 13], can support task migration, which means a task can migrate from one cluster (e.g., big cluster) to another cluster (e.g., LITTLE cluster) during task execution, and thus, the efficiency and the schedulability of task execution can be further enhanced. The methods mentioned above and the corresponding optimization variables are summarized in Table 5.

For the networked systems, since the nodes are connected with each other wirelessly, when dependent tasks are assigned to different nodes for execution, the nodes will spend time and energy for data communication [14]. Since the communication range of each node is limited, the task allocation decision will influence the communication cost of the nodes. Hence, we also need to optimize task-to-node allocation. As the task is time sensitive and the energy budget of the node is limited, this impact should be formulated. To achieve that, we can introduce an energy matrix $\boldsymbol{E} = [e_{\beta\gamma k}]_{M\times M\times M}$ and a time matrix $\boldsymbol{T} = [t_{\beta\gamma}]_{M\times M}$, where M is the number of the nodes. $e_{\beta\gamma k}$ is the energy consumed by a node θ_k when relaying unit of data from node θ_β to node θ_γ, and $t_{\beta\gamma}$ is the time required to transmit unit of data from node θ_β to node θ_γ. Therefore, based on the matrices $\boldsymbol{E}$ and $\boldsymbol{T}$, we obtain the corresponding communication cost under the given task allocation decision [15].

3 Task Mapping Methods

The basic task mapping contains two steps: 1) task allocation: determines on which processor/node should the task be executed, and 2) task scheduling: determines when a task starts and ends its execution. Table 6 classifies the relevant state-of-the-art methods, presented in this section, based on 1) the task model (Imprecise, Precise), 2) the target platform (Embedded, Networked), 3) the constraints (Energy, Real-Time, Reliability), 4) the objective (Minimize Energy, Balance Energy, Maximize Reliability, Maximize QoS), and 5) the achieved solutions (Heuristic, Optimal) of task mapping problem under study.

Table 6: Task Mapping Methods

Ref.	Task		Platform		Constraints			Objective				Solution	
	Imprecise	Precise	Embedded	Networked	Energy	Real-time	Reliability	MinE.	BalE.	MaxR.	MaxQoS	H.	O.
[16]		√	√			√		√				√	
[17]		√	√			√		√				√	
[9]		√	√			√		√				√	
[7]		√	√			√		√				√	
[18]		√	√			√		√				√	
[19]		√	√			√		√				√	
[20]		√	√		√	√				√		√	
[21]		√	√		√	√				√		√	√
[22]		√	√		√	√				√		√	
[23]		√	√			√	√	√				√	
[10]		√	√			√	√	√				√	
[11]		√	√			√	√	√				√	
[24]		√	√			√	√	√					√
[4]	√		√		√	√					√	√	√
[25]	√		√		√	√					√	√	
[26]	√		√		√	√					√	√	
[5]	√		√		√	√					√		√
[3]	√		√		√	√					√	√	
[27]	√		√		√	√					√	√	
[28]	√		√		√	√					√	√	√
[29]		√		√	√	√			√			√	
[30]		√		√	√	√			√				√
[31]		√		√	√	√		√				√	
[32]		√		√	√	√		√					√
[33]		√		√	√	√		√					√
[14]		√		√	√	√		√	√			√	√

3.1 Task Mapping on Embedded Systems

Existing task mapping methods can be classified as *Energy-aware* task mapping and *QoS-aware* task mapping.

3.1.1 Energy-Aware Task Mapping

Energy-aware task mapping problem usually considers the precise computation task model and the aim to minimize the energy consumption under energy, real-time and reliability constraints.

When the voltage/frequency level is *discrete*, the corresponding task mapping problem is usually formulated as Integer Programming (IP), e.g., [16, 17, 9]. To efficiently solve the task mapping problem, a hybrid Genetic Algorithm (GA) is proposed in [16], a polynomial-time two-step heuristic is designed in [17], and the IP problem is relaxed to a Linear Programming (LP) in [9]. Combining DVFS and DPM, a Mixed-Integer Linear Programming (MILP)-based task mapping problem is considered in [7] and the problem is solved by CPLEX solver.

When the voltage/frequency level is *continuous*, a convex task mapping problem is proposed in [18] and the problem can be solved by using polynomial-time methods. In [19], Mixed-Integer Non-Linear Programming (MINLP) is used to formulate the task mapping problem. The problem is relaxed to an MILP by linear approximation and is solved by Branch and Bound (B&B) method.

If multiple system requirements are taken into account, the complex coupling between the optimization variables makes the problem difficult to solve, especially when the coupling is non-linear and non-convex. The common methods to deal with the nonlinear items include: 1) linear approximation [19], and 2) variables replacement [7].

Taking the task reliability into account, existing methods include *reliability-optimized* task mapping [20, 21, 22] and *energy-optimized* task mapping [23, 10, 11]. The aim of reliability-optimized task mapping is to maximize task reliability under system resource and application constraints. Regarding energy-optimized task mapping, the aim is to minimize energy consumption, under energy supply, task reliability and real-time constraints.

To maximize the reliability of task execution, as well as to meet energy supply, task dependency and task deadline constraints, the dynamic and static methods are proposed in [20] and [21] to allocate and schedule dependent tasks on the multi-core platforms. The multi-objective task mapping problem is consider in [22], where the aim is to simultaneously maximize the reliability and the lifetime of tasks.

DVFS is applied in [23] to meet task reliability constraint. Since task duplication is not taken into account, higher voltage/frequency level may require to execute the tasks. In [10], full replication is used to meet reliability constraint, and thus, each task is replicated once at least. Although more tasks being duplicated, higher reliability is achieved while task redundancy is incurred, more energy and time are required to execute the tasks. DVFS and task duplication are combined in [11] and [24], where the only partial tasks are duplicated.

3.1.2 QoS-Aware Task Mapping

Existing works consider the QoS-aware task mapping problem using the IC task model and having a goal to maximize the QoS under a set of realtime and/or energy supply constraints.

The target platforms studied in [4] and [25] are single-core platforms. Therefore, there is no need to consider task allocation decision. Although some works target at multi-core platforms, e.g., [26, 5, 3, 27, 28], they focus on different contexts. For example, the task-to-processor allocation is fixed and given in advance for all the tasks in [26], each processor has a predefined frequency in [5, 3], the tasks are independent in [28], and the multi-objective task mapping is consider in [27] with the aim is to maximize the QoS as well as to minimize the energy consumption. For tractability reasons, the variable, optional subtask adjustment, is usually considered as continuous variable in the above studies. When task mapping problem is solved, the result is rounded down. As the tasks execute typically hundreds of thousands of cycles, this impact is negligible [4].

The QoS-aware task mapping problem is a well-known NP-hard problem. Hence, finding an optimal solution satisfying all the given constraints (e.g., energy efficiency, deadline, QoS, task dependency, and DVFS) is very difficult and time consuming. The methods that used to solve the aforementioned problems can be classified into two main classes. The first class includes the methods based on heuristics, e.g., [25, 26, 3, 27]. The second class includes the methods that always produce an optimal solution, e.g., [4, 5, 28].

The heuristic methods mentioned above usually adopted a multi-step optimization, i.e., to decouple the variables and to determine their values in sequence. For instance, a two-step heuristic is proposed in [3]. The aim of the first step is to find a proper task-to-processor allocation, such that the energy consumption is minimized. With the given task allocation decision, the energy consumed to execute one task is proportional to the length of its optional subtask.

Based on the given task allocation decision, the aim of the second step is to adjust the optional subtasks so as to maximize OoS under the energy supply constraint. Although the heuristic methods are able to find feasible solutions in a short amount of time, they do not provide the bounds on the solution quality. In addition, they are sensitive to the problem structure, i.e., when new assumptions or constraints are taken into account they must be redeveloped.

On the other hand, to find the optimal solution, the common methods include: 1) convex optimization [4, 5], and 2) Benders Decomposition (BD) [28]. Instead of solving the binary and the continuous variables of MILP problem simultaneously, BD technique decomposes the original problem into two smaller problems with less variables and constraints: an ILP-based *Master* Problem (MP) and a LP-based *Slave* Problem (SP). Then, it solves the subproblems by utilizing the solution of one in the other. By doing so, the computation time can be significantly reduced. In each iteration, the current MP is solved to determine a lower bound for the original problem along with the temporary values of the binary variables. And then the SP is solved to obtain an upper bound by utilizing the solution of MP. The bounds are updated if the stopping criterion, i.e., the gap between the upper and lower bounds is smaller than a predefined threshold, is not met, and a new constraint (i.e., *Benders cut*) is generated by using the solution of SP and is added to MP in next iteration. As the BD method runs in an iterative way, the stopping criteria can serve as the controllable parameters to trade-off the quality of the solution (i.e., system QoS) and the computational complexity (i.e., computing time). In addition, as the computational complexity of BD method is dominated by the cost of solving the ILP-based MP, an accelerated BD method is proposed in [2], without violating optimality of the solution. This method replaces the optimal solution of MP with the feasible solution and uses it for the iteration between the MP and the SP. The structure of BD algorithm is shown in Fig. 1.

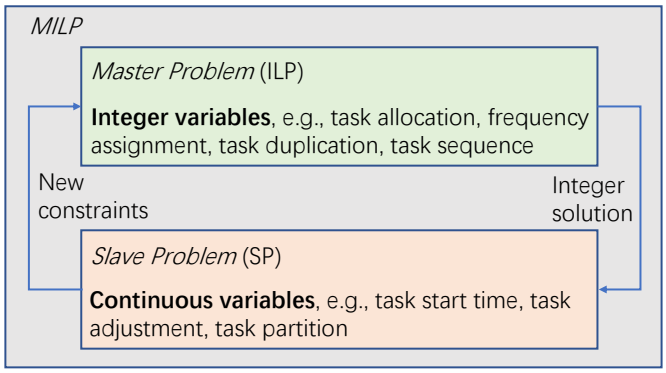

Figure 1: The structure of BD algorithm.

3.2 Task Mapping on Networked Systems

In the networked systems, there are various approaches to map dependent tasks on the wireless nodes, e.g., [29, 30, 31, 32, 33, 14]. In [29], the objective of task mapping is to minimize the energy consumption of the nodes with low energy level, and thus, the system lifetime can be enhanced. To solve this task mapping problem a multi-step heuristic method is designed. Similar task mapping problem is studied in [30], while a game theory approach is proposed to find the solution. However, the voltage/frequency levels of the processors are fixed in these approaches. The problems of task mapping and DVFS are jointly addressed in [31] and [32], based on the idea of problem decomposition, a heuristic method [31] and an optimal method [32] are presented to solve complex optimization problems. In [33], by using evolutionary algorithms (ant colony and bee colony) to perform task mapping, the energy consumption of the nodes for data communication and task execution can be minimized. The above methods assume that one node transmits data to another node through a fixed path. Since the wireless nodes are connected with each other through a mesh network, the communication between the nodes can be performed through multiple routing paths. The multi-path data routing is considered in [14], where the aim to enhance system lifetime, i.e., to balance the energy consumption of the nodes. The task mapping problem is first formulated as an Integer Non-Linear Programming (INLP) and then is solved by greedy algorithm.

4 Conclusion

This survey provides a current view of task mapping problem in CPS. We follow a three-step approach to summarize the recently published papers in the relevant area. More precisely, a task classification is obtained regarding he characteristics of task models, including the task dependency and the adjustment of execution cycles. A platform classification is proposed based on the platform type and the functions of the processor/node. Finally, the task mapping methods are classified, based on task and system under study, using task and system classifications.

References

[1] J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung, "Imprecise computations," *Proc. IEEE*, vol. 82, no. 1, pp. 83–94, 1994.

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

[3] 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 *Proc. IEEE DATE*, 2017, pp. 1402–1407.

[4] L. A. Cortes, P. Eles, and Z. Peng, "Quasi-static assignment of voltages and optional cycles in imprecise-computation systems with energy considerations," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 10, pp. 1117–1129, 2006.

[5] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, "Optimal reward-based scheduling for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 111–130, 2001.

[6] X. Liu, H. Zhou, J. Xiang, S. Xiong, K. M. Hou, C. de Vaulx, H. Wang, T. Shen, and Q. Wang, "Energy and delay optimization of heterogeneous multicore wireless multimedia sensor nodes by adaptive genetic-simulated annealing algorithm," *Wirel. Commun. Mob. Comput.*, vol. 2018, pp. 1–13, 2018.

[7] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3, pp. 1–21, 2014.

[8] K. Huang, K. Wang, D. Zheng, X. Jiang, X. Zhang, R. Yan, and X. Yan, "Expected energy optimization for real-time multiprocessor SoCs running periodic tasks with uncertain execution time," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2018.

[9] D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 810–823, 2015.

[10] M. A. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 813–825, 2017.

[11] C. Gou, A. Benoit, M. Chen, L. Marchal, and T. Wei, "Reliability-aware energy optimization for throughput-constrained applications on MPSoCs," in *International Conference on Parallel and Distributed Systems*, 2018, pp. 1–10.

[12] H. S. Chwa, J. Seo, H. Yoo, J. Lee, and I. Shin, "Energy and feasibility optimal scheduling on big.LITTLE platforms," in *Proc. IEEE RTSOPS*, 2013, pp. 770–777.

[13] L. Mo, A. Kritikakou, and O. Sentieys, "Approximation-aware task deployment on asymmetric multicore processors," in *Proc. ACM/IEEE DATE*, 2019, pp. 1513–1518.

[14] A. Pathak and V. K. Prasanna, "Energy-efficient task mapping for data-driven sensor network macroprogramming," *IEEE Trans. Comput.*, vol. 59, no. 7, pp. 955–968, 2010.

[15] L. Mo, A. Kritikakou, O. Sentieys, and X. Cao, "Real-time imprecise computation tasks mapping for DVFS-enabled networked systems," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8246–8258, 2021.

[16] A. Mahmood, S. A. Khan, F. Albalooshi, and N. Awwad, "Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm," *Electron.*, vol. 6, no. 2, 2017.

[17] H. Xu, F. Kong, and Q. Deng, "Energy minimizing for parallel real-time tasks based on level-packing," in *Proc. IEEE RTCSA*, 2012, pp. 98–103.

[18] F. Kong, W. Yi, and Q. Deng, "Energy-efficient scheduling of real-time tasks on cluster-based multicores," in *Proc. ACM/IEEE DATE*, 2011, pp. 1–6.

[19] L. F. Leung, C. Y. Tsui, and W. H. Ki, "Simultaneous task allocation, scheduling and voltage assignment for multiple-processors-core systems using mixed integer nonlinear programming," in *Proc. IEEE ISCAS*, 2003, pp. 309–312.

[20] Y. Ma, T. Chantem, R. P. Dick, S. Wang, and X. S. Hu, "An on-line framework for improving reliability of real-time systems on "big-LITTLE" type MPSoCs," in *Proc. ACM/IEEE DATE*, 2017, pp. 446–451.

[21] B. Zhao, H. Aydin, and D. Zhu, "On maximizing reliability of real-time embedded applications under hard energy constraint," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 316–328, 2010.

[22] J. Zhou, J. Sun, X. Zhou, T. Wei, M. Chen, S. Hu, and X. S. Hu, "Resource management for improving soft-error and lifetime reliability of real-time MPSoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 12, pp. 2215–2228, 2019.

[23] G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li, and K. Li, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1629–1640, 2017.

[24] M. Cui, A. Kritikakou, L. Mo, and E. Casseau, "Fault-tolerant mapping of real-time parallel applications under multiple DVFS schemes," in *Proc. IEEE RTAS*, 2021, pp. 387–399.

[25] H. Yu, B. Veeravalli, and Y. Ha, "Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems," in *Proc. IEEE ASP-DAC*, 2008, pp. 452–455.

[26] H. Yu, B. Veeravalli, Y. Ha, and S. Luo, "Dynamic scheduling of imprecise-computation tasks on real-time embedded multiprocessors," in *Proc. IEEE CSE*, 2013, pp. 770–777.

[27] M. Isabel, O. Javier, S. Rodrigo, and Z. Paula, "Energy-aware scheduling mandatory/optional tasks in multicore real-time systems," *Intl. Trans. in Op. Res.*, vol. 24, no. 12, pp. 173–198, 2017.

[28] L. Mo, A. Kritikakou, and O. Sentieys, "Energy-quality-time optimized task mapping on DVFS-enabled multicores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2428–2439, 2018.

[29] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, and L. Pirmez, "Efficient allocation of resources in multiple heterogeneous wireless sensor networks," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1775–1788, 2014.

[30] N. Edalat, C. Tham, and W. Xiao, "An auction-based strategy for distributed task allocation in wireless sensor networks," *Computer Communications*, vol. 35, no. 8, pp. 916–928, 2012.

[31] Y. Tian and E. Ekici, "Cross-layer collaborative in-network processing in multihop wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 3, pp. 297–310, 2007.

[32] L. Mo and A. Kritikakou, "Mapping imprecise computation tasks on cyber-physical systems," *Peer-to-Peer Networking and Applications*, vol. 12, no. 6, pp. 1726–1740, 2019.

[33] W. Zhang, B. Song, and E. Bai, "A trusted real-time scheduling model for wireless sensor networks," *Journal of Sensors*, vol. 2016, pp. 1–8, 2016.