Energy Efficient, Real-time and Reliable Task Deployment on NoC-based Multicores with DVFS

Lei Mo, Qi Zhou, Angeliki Kritikakou, Ji Liu

Energy Efficient, Real-time and Reliable Task Deployment on NoC-based Multicores with DVFS

Lei Mo*, Qi Zhou†, Angeliki Kritikakou‡, and Ji Liu§
*School of Automation, Southeast University, Nanjing 210096, China, Email: lmo@seu.edu.cn
†School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China, Email: 220184505@seu.edu.cn
‡Univ Rennes, INRIA, CNRS, IRISA, Rennes 35042, France, Email: angeliki.kritikakou@irisa.fr
§Big Data Laboratory, Baidu Research, Beijing 100085, China, Email: liuji04@baidu.com

***Abstract*—Task deployment plays an important role in the overall system performance, especially for complex architectures, including several cores with Dynamic Voltage and Frequency Scaling (DVFS) and Network-on-Chips (NoC). Task deployment affects not only the energy consumption but also the real-time response and reliability of the system. In this work, a task deployment approach is proposed to optimize the overall system energy consumption, including computation of the cores and communication of the NoC, under task reliability and real-time constraints. More precisely, the task deployment approach combines task allocation and scheduling, frequency assignment, task duplication, and multi-path data routing. The task deployment problem is formulated using mixed-integer non-linear programming. To find the optimal solution, the original problem is equivalently transformed to mixed-integer linear programming, and solved by state-of-the-art solvers. Furthermore, a decomposition-based heuristic, with low computational complexity, is proposed to deal with scalability. Finally, extended simulations evaluate the proposed methods.**

I. Introduction

Multicore architectures integrate multiple processors on a single chip, leading to platforms with low supply frequency, high data throughput and better energy efficiency [1]. With the development of nanoscale technologies, various processors communicate via the Network-on-Chips (NoC), instead of traditional, non-scalable, data buses [2]. The NoC uses its underlying micro-network components, i.e., routers, for the communication among the processors. The routers are usually connected by a mesh network, one of the most effective NoC topologies, due to its regularity, high bandwidth and short interconnections [3].

The inter-processor communication cost (time and energy) over NoC is not negligible compared to the cost of computation, occurring at the processors [4]. Furthermore, the communication cost depends on both task mapping and routing path decision. When dependent tasks are allocated and executed on different processors, data must be transmitted. Furthermore, multiple routing paths for the data transmission can exist in NoC, such as in the mesh network [2]. To improve the overall system performance, it is necessary to consider inter-processor communication, task mapping and routing path selection as an integral task deployment process.

Meanwhile, critical applications have the real-time and reliability constraints [5], whereas system energy consumption is essential. Although energy efficiency, real-time and reliable task execution are both important design objectives, they are usually conflicting. Enhancing real-time execution and reliability often requires more energy consumption. Dynamic Voltage and Frequency Scaling (DVFS) [6] has been introduced into modern multicore platforms to improve the energy efficiency, as it adjusts the time and the energy required to execute the tasks. However, DVFS has a negative impact on task execution and reliability. Lower frequencies lead to longer execution times and increased transient fault rates [7]. To increase the reliability, task duplication is applied [8]. However, task duplication has a significant impact both on energy and time; more tasks are executed, thus, more energy and time is required for the task computation and their communication over the NoC. Therefore, task duplication and frequency assignment should be also considered during the deployment process [9].

TABLE I
Classification of representative task deployment approaches

Ref.	*Task*			*Multicore platform*				*Solution*	
	All.	Sch.	Dup.	VF.	Reli.	Com.	MP.	Opt.	Heur.
[6]	√	√		√					√
[8]	√	√	√	√	√				√
[10]	√	√		√				√	√
[11]	√	√	√	√	√				√
[4]	√	√				√			√
[12]	√	√				√			√
[3]	√	√		√		√			√
[13]	√	√		√		√			√
[2]	√	√			√	√	√		√
[5]	√	√	√		√	√		√	√
[14]	√	√	√		√	√			√
Our	√	√	√	√	√	√	√	√	√

The task deployment problem has already been studied in embedded systems. Table I categories representative papers from the literature, performing task allocation (All.), scheduling (Sch.) and duplication (Dup.) on multicores with DVFS (VF.), considering multi-path routing (MP.), task reliability (Reli.) and communication cost (Com.). The solutions are given by optimal (Opt.) and heuristic (Heur.) algorithms. Several approaches exist to map dependent/independent tasks on the multi-core/single-core embedded platforms under multiple constraints, such as energy, real-time, and reliability [6], [8], [10], [11]. These works consider processors typically connected with a high-speed data bus. Thus, the communication cost between any two processors is usually ignored, as it's much smaller compared to task execution cost. However, for the platforms based on NoC, the communication cost becomes important. To reduce the communication cost on NoC-based platforms, the common methods include mapping tasks to processors [4], [12] and adjusting the operating voltage of the routers [3], [13]. However, task reliability is not taken into account, especially when DVFS is available. Existing approaches, to enhance NoC reliability,

include spare processors [2] and task duplication [5], [14]. However, no DVFS is considered [5], [14].

Compared with the state-of-the-art, we propose a novel task deployment process that includes simultaneous optimization of task allocation and scheduling, frequency assignment, task duplication and path selection, in order to balance the energy consumption of NoC-based multicore platforms, while meeting the system requirements regarding real-time execution and reliability. The main contributions of this work are:

1) An MINLP formulation for energy efficient task deployment problem on NoC-based multicores. The original problem is equivalently converted to an MILP problem, by adding auxiliary variables and constraints, to be optimally solved.
2) A decomposition-based heuristic to solve the above problem, which divides the original problem into three subproblems, having a simpler structure with less constraints and variables.
3) Extensive experimental results to demonstrate the advantages of the proposed approach, and evaluate the impact of parameters on task deployment. Compared with the optimal method, the proposed heuristic has negligible computation time, with an average cost of 26.5% in energy savings.

Next, Section II introduces the system model and the problem formulation. Section III designs the heuristic method. Section IV presents the evaluation and Section V the conclusion.

II. System Model and Problem Formulation

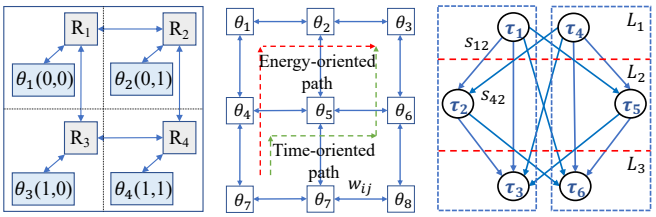
A. System Model

1) Task Set: The task set consists of M periodic tasks $\{\tau_1, \ldots, \tau_M\}$, released at time 0, sharing a common scheduling horizon H. Each task τ_i is described by a tuple $\{C_i, D_i, p_{ij}, s_{ij}\}$, where C_i is the Worst Case Execution Cycle (WCEC), D_i is the relative deadline, and p_{ij} is the $(i,j)^{th}$ element of task dependency matrix $\boldsymbol{p} = [p_{ij}]_{M\times M}$. If tasks τ_i and τ_j are dependent, and τ_i is a predecessor of τ_j, $p_{ij} = 1$, else, $p_{ij} = 0$. When τ_i finishes, it will generate a set of data with sizes s_{ij} for its successor τ_j ($p_{ij} = 1$).

2) Platform: The target platform consists of N processors $\{\theta_1, \ldots \theta_N\}$ and N routers (R), connected through a 2D-mesh network, as a 2×2 example shown in Fig. 1(a). Each router communicate with the neighbor routers through a pair of links. Data can be transmitted through multiple paths.

The processors support DVFS and have the same Instruction Set Architecture (ISA). A processor has L different Voltage/Frequency (V/F) levels $\{(v_1, f_1), \ldots, (v_L, f_L)\}$. A typical power model [13] is considered: the processor power with (v_l, f_l) is $P_l = P_l^s + P_l^d$. The static power is $P_l^s = L_g(v_l K_1 e^{K_2 v_l} e^{K_3 v_b} + |v_b| I_b)$. The dynamic power is $P_l^d = C_e v_l^2 f_l$. C_e is the average switched capacitance. L_g is the number of logic gates. K_1, K_2 and K_3 are parameters depending on the processor type. v_b and I_b are the body-bias voltage and body junction leakage current. The computation energy of task τ_i is $e_i^{comp} = P_l t_i^{comp}$, where t_i^{comp} its computation time.

The NoC topology is described by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the vertexes represent each processor $\theta_k \in \mathcal{V}$, while the edge $l_{ij} \in \mathcal{E}$ represents a direct communication link between the routers of processors θ_i and θ_j. Based on the NoC communication model and the Manhattan distance between the source and destination processors, a positive weight w_{ij} is associated with the edge l_{ij}. The goal of the task deployment is energy reduction under real-time constraints, we consider both energy-oriented and time-oriented paths as available options to transmit data, as shown in Fig. 1(b). Note that, the path with minimal energy can be different from the path with minimal latency [4]. Our approach explores these different paths.



(a) Platform (b) Multi-path routing (c) Task duplication

Fig. 1. An example of NoC-based multicore system.

For the energy (time)-oriented path, the weight w_{ij} represents the energy (time) required for transmitting and receiving a unit of data between processors θ_i and θ_j. Hence, the aim of energy (time)-oriented routing is to find the shortest path, e.g., according to Dijkstra's algorithm. Based on the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we obtain an energy matrix $\boldsymbol{e} = [e_{\beta\gamma k\rho}]_{N\times N\times N\times P}$ and a time matrix $\boldsymbol{t} = [t_{\beta\gamma\rho}]_{N\times N\times P}$, where $e_{\beta\gamma k\rho}$ represents the energy consumed at processor θ_k, if a unit of data is routed from θ_β to θ_γ through the ρ^{th} path, while $t_{\beta\gamma\rho}$ denotes the time required to transmit a unit of data from θ_β to θ_γ through the ρ^{th} path. When two dependent tasks are mapped on the same processor, the communication cost (time and energy) is zero [12]. The communication energy of a router, including the communications between this router and its neighbor routers and associated processor, is incorporated into the energy consumption of the processor for convenience.

3) Reliability: This work focuses on transient faults and adopts the Poisson fault probability model [8]. If a processor uses (v_l, f_l) to execute the task τ_i with C_i cycles, the task reliability is $r_{il} = e^{-\lambda \times 10^{\frac{d(f_{\max}-f_l)}{f_{\max}-f_{\min}}} \times \frac{C_i}{f_l}}$, where λ is the maximum failure, d is a constant that indicates the sensitivity of the failure rate corresponding to frequency scaling, $f_{\max} = \max_{\forall l}\{f_l\}$ and $f_{\min} = \min_{\forall l}\{f_l\}$. When the reliability of task τ_i is lower than its threshold R_{th}, to enhance the task reliability, task τ_i is duplicated (see (4) and (5)), considering that it is unlikely to have faults occurring concurrently in both copies [7].

Note that the task duplication affects the task model, and thus, the computation and communication cost. For instance, tasks τ_1, τ_2 and τ_3 are the original tasks, and tasks τ_4, τ_5 and τ_6 are their copies in Fig. 1(c). By task duplication, the task dependencies change, e.g., due to the dependency of τ_1 and τ_2, τ_4 and τ_2 become dependent and τ_4 generates data to τ_2.

B. Problem Formulation

The goal is to deploy tasks to balance the energy consumption of the overall system under real-time and reliability constraints. To achieve that, the following decisions are taken: 1) task frequency assignment, 2) task duplication, 3) path selection, 4) task allocation and 5) task scheduling. To formulate the task deployment problem, we introduce the following variables: 1) binary variable $y_{il} = 1$ if task τ_i is executed with frequency

f_l, otherwise, $y_{il} = 0$; 2) binary variable $h_i = 1$ if task τ_i exists, otherwise, $h_i = 0$; 3) binary variable $c_{\beta\gamma\rho} = 1$ if data is transmitted from θ_β to θ_γ through the ρ^{th} path, otherwise, $c_{\beta\gamma\rho} = 0$; 4) binary variable $x_{ik} = 1$ if task τ_i is allocated to processor θ_k, otherwise, $x_{ik} = 0$; 5) binary variable $u_{ij} = 1$ if task τ_i proceeds τ_j, otherwise, $u_{ij} = 0$; 6) continuous variable t_i^s represents the start time of task τ_i. For the sake of paper presentation, let $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{M} = \{1, \ldots, M\}$, $\mathcal{M}' = \{1, \ldots, 2M\}$, $\mathcal{L} = \{1, \ldots, L\}$ and $\mathcal{P} = \{1, 2\}$.

1) Task Allocation Constraints: Each task is executed on a single processor [6], without task migration:

$$\sum\nolimits_{k\in\mathcal{N}} x_{ik} = 1,\ \forall i \in \mathcal{M}'. \tag{1}$$

2) Path Selection Constraints: One data routing path is selected between two processors [2]:

$$\sum\nolimits_{\rho\in\mathcal{P}} c_{\beta\gamma\rho} = 1,\ \forall \beta \neq \gamma \in \mathcal{N}. \tag{2}$$

3) Frequency Assignment Constraints: The task is executed with one V/F level [10]:

$$\sum\nolimits_{l\in\mathcal{L}} y_{il} = 1,\ \forall i \in \mathcal{M}'. \tag{3}$$

4) Task Reliability Constraints: Let τ_i and τ_{i+M} ($\forall i \in \mathcal{M}$) denote the original and duplicated tasks, respectively, where τ_i and τ_{i+M} have the same execution cycles. Taking r_{il} and y_{il} into account, the reliability of task τ_i ($\forall i \in \mathcal{M}$), without duplication, is $r_i = \sum_l y_{il} r_{il}$. Note that whether a duplicated task τ_{i+M} exists, it dependents on the task reliability r_i. To indicate that, we introduce a binary variable h_i. Since the original tasks always exist, $h_i = 1$ ($\forall i \in \mathcal{M}$). If $r_i \geq R_{th}$, there is no need to duplicate task τ_i ($h_{i+M} = 0$); otherwise, task τ_i is duplicated ($h_{i+M} = 1$). We alternatively rewrite the above comparison in a linearly manner by the following lemma.

Lemma 2.1: [1] Assume that b is a binary variable, while x is a variable bounded by $0 \leq x \leq s$. The comparison: 1) $x \geq s_1 \Rightarrow b = 0$; 2) $x < s_1 \Rightarrow b = 1$ can be described by $\frac{x-(s_1-\sigma)}{s} \leq 1 - b \leq \frac{x}{s_1}$, where $s_1 \leq 1$ is a constant, and σ is a positive small enough value.

Based on *Lemma* 2.1, let $\sigma = \min_{\forall i,l}\{|r_{il} - R_{th}|\}$. The relationship between r_i and h_{i+M} can be described as:

$$\frac{r_i - (R_{th} - \sigma)}{\max_{\forall i,l}\{r_{il}\}} \leq 1 - h_{i+M} \leq \frac{r_i}{R_{th}},\ \forall i \in \mathcal{M}. \tag{4}$$

By duplicating the task τ_i ($\forall i \in \mathcal{M}$), its reliability is given by $r_i' = 1 - (1 - h_i r_i)(1 - h_{i+M} r_{i+M})$. Therefore, to satisfy the reliability constraint, we have

$$r_i' \geq R_{th},\ \forall i \in \mathcal{M}. \tag{5}$$

5) Task Sequence Constraint: A task τ_i cannot start execution until the input data from all its predecessors has arrived. When a task completes execution, its output data is available for transmission to all its successors. A router propagates the received task data, from other routers, towards its processor in sequence, and thus, the time spent for receiving the data required for the task execution is $t_i^{comm} = \sum_j \sum_\beta \sum_\gamma \sum_\rho p_{ji} h_i h_j x_{i\beta} x_{j\gamma} c_{\beta\gamma\rho} t_{\beta\gamma\rho}$.

[1]Due to the page limit, the proofs of the lemmas are omitted.

Let t_i^s and t_i^e denote the start time and the end time of task τ_i, respectively, where $0 \leq t_i^s \leq t_i^e \leq H$. The task execution time $t_i^{comp} = t_i^e - t_i^s = h_i \sum_l y_{il} \frac{C_i}{f_l}$. For the dependent tasks τ_i and τ_j, their start time and end time are bounded by

$$t_j^s + (1 - p_{ij})H \geq t_i^s + p_{ij} t_i^{comp} + t_j^{comm},\ \forall i \neq j \in \mathcal{M}'. \tag{6}$$

If $p_{ij} = 1$, τ_i precedes τ_j and τ_j is the closest task of τ_i, we get $t_j^s \geq t_i^e + t_j^{comp}$, else, (6) is always satisfied.

6) Task Non-Overlapping Constraint: When independent tasks τ_i and τ_j (i.e., $p_{ij} = 0$), are allocated to the same processor, their execution sequence must be determined, since the processor executes only one task at a time instance:

$$t_i^e \leq t_j^s + (2 - x_{ik} - x_{jk})H + (1 - u_{ij})H,\ \forall i \neq j \in \mathcal{M}',\ \forall k \in \mathcal{N}. \tag{7}$$

If τ_i and τ_j are assigned to the same processor (e.g., $x_{ik} = x_{jk} = 1$), (7) is meaningful, else, (7) is always true. With $x_{ik} = x_{jk} = 1$, if $u_{ij} = 1$ (i.e., τ_i precedes τ_j), we have $t_i^e \leq t_j^s$, else (i.e., $u_{ij} = 0$), (7) is always satisfied.

7) Real-Time Constraints: Since task τ_i should be finished within the scheduling horizon H and its execution time should be smaller than the relative deadline D_i, we have

$$t_i^{comp} \leq D_i,\ \forall i \in \mathcal{M}', \tag{8}$$

$$t_i^e \leq H,\ \forall i \in \mathcal{M}'. \tag{9}$$

8) Objective Function: Regarding the communication energy, if dependent tasks τ_i and τ_j are allocated to different processors, e.g., θ_β and θ_γ, the energy consumed on processor θ_k to transmit task data, with size s_{ij} from θ_β to θ_γ, through the ρ^{th} path, is $e_{ij\beta\gamma k\rho}^{comm} = p_{ij} s_{ij} x_{i\beta} x_{j\gamma} e_{\beta\gamma k\rho}$. On this basis, taking task duplication h_i into account, the communication energy consumed by processor θ_k is $E_k^{comm} = \sum_i \sum_j \sum_\beta \sum_\gamma \sum_\rho h_i h_j e_{ij\beta\gamma k\rho}^{comm}$.

Regarding the computation energy, to execute task τ_i with (v_l, f_l), the required energy is $e_i^{comp} = h_i \sum_l y_{il} \frac{C_i}{f_l} P_l$. Taking task allocation x_{ik} into account, the computation energy consumed by processor θ_k is $E_k^{comp} = \sum_i x_{ik} e_i^{comp}$.

To balance the energy consumption of the processors [15], the task deployment problem is formulated as follows:

$$\mathbf{P1}: \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{h},\boldsymbol{c},\boldsymbol{u},\boldsymbol{t^s}} (\max_{\forall k}\{E_k^{comm} + E_k^{comp}\}) \tag{10}$$
$$\text{s.t. } (1) - (8).$$

9) Problem Linearization: Since the nonlinear items $h_i y_{il}$, $x_{ik} h_i y_{il}$ and $h_i h_j x_{i\beta} x_{j\gamma}$ are included in the constraints (5)–(8), problem (10) is an MINLP, which is difficult to solve directly. To simplify the structure of the problem, we propose a linearization method to deal with nonlinear items.

Lemma 2.2: Assume that x, y and z are the binary variables. The nonlinear item $z = xy$ can be replaced by the following constraints: $z - x \leq 0$, $z - y \leq 0$ and $x + y - z \leq 1$.

The objective function with the constraints completes the MILP formulation, which can be optimally solved by the existing methods, such as branch-and-bound or Gurobi solver. However, significantly large amount of CPU time and resources is required for the optimal solution, especially with large problem sizes. This limitation reduces the applicability of complex, but more realistic, task deployments approaches in real systems.

Algorithm 1 Frequency Assignment and Task Duplication

1: **Input**: (v_l, f_l) $(\forall l \in \mathcal{L})$, C_i, D_i $(\forall i \in \mathcal{M}')$;
2: **Output**: y_{il} and h_i;
3: **Initialize**: $S[i] = -1$ $(\forall i \in \mathcal{M}')$, $h_i = 1$, $h_{i+M} = -1$ $(\forall i \in \mathcal{M})$;
4: **for** $\forall i \in \mathcal{M}$ **do**
5: $eMinMaxComp = \infty$;
6: **for** $\forall l \in \mathcal{L}$ **do**
7: **if** $\frac{C_i}{f_l} > D_i$ **then**
8: Continue;
9: **else**
10: $eMaxComp = \max_{\forall i}\{\frac{C_i}{f_{S[i]}} P_{S[i]}\}$ $(S[i] \neq -1)$;
11: **if** $eMaxComp < eMinMaxComp$ **then**
12: $eMinMaxComp = eMaxComp$;
13: $S[i] = l$;
14: **else**
15: Continue;
16: **end if**
17: **end if**
18: **end for**
19: Calculate y_{il} according to $S[i] = l$;
20: Calculate h_{i+M} according to (4);
21: Calculate $y_{(i+M)l}$ according to (5);
22: **end for**

III. Heuristic algorithm

Therefore, to improve the scalability of the proposed task deployment approach, we propose a novel heuristic to efficiently solve problem (10), which contains the following three phases.

1) Frequency Assignment and Task Duplication: The processors have the same ISA and V/F levels. Therefore, if a frequency is assigned to each task, and then, the tasks are allocated to the processors or different paths are selected for data transmission, the decision regarding the frequency assignment is still valid. In addition, as task duplication h_i is determined by frequency assignment y_{il}, according to (4), y_{il} and h_i should be still jointly optimized. Since y_{il} and h_i mainly influence the time t_i^{comp} and the energy e_i^{comp} of task computation, to balance the computation energy, the frequency assignment and task duplication problem is formulated as:

$$\mathbf{P2}: \min_{\boldsymbol{y},\boldsymbol{h}} (\max_{\forall i}\{e_i^{comp}\}) \quad \text{s.t. } (3), (4), (5), (8). \tag{11}$$

Minimizing the maximum energy consumption of each task execution helps to balance the energy consumption of the processors during task allocation occurring in phase 2. Since task allocation x_{ik} and path selection $c_{\beta\gamma\rho}$ are currently unknown, the communication cost (time and energy) is not considered in (11). Note that P2 is an INLP problem, as the binary variables y_{il} and h_i are coupled nonlinearly in (5) and (8). To solve this problem, a heuristic is proposed (described in Algorithm 1) based on the Greedy Algorithm (GA) [16]. The steps are:

a. An index $S[i]$ is introduced for frequency assignment per task τ_i $(\forall i \in \mathcal{M}')$, and it is initialized as $S[i] = -1$ (Line 3). If (v_l, f_l) is used to execute task τ_i, then $S[i] = l$.
b. Algorithm 1 follows the sequence $\{\tau_1, \ldots, \tau_M\}$ to iteratively assign the frequencies $\{f_1, \ldots, f_L\}$ for each task τ_i, with the aim to minimize the increase of energy consumption among the tasks that have already been assigned a frequency, i.e., $\min(\max_{\forall i}\{e_i^{comp}\})$ $(S[i] \neq -1)$ (Line 10). If the real-time constraint (8) cannot be satisfied, the frequency assignment $y_{il} = 1$ is excluded (Line 7).
c. When the frequency assignment y_{il} regarding the original task τ_i is known, the existence of the duplicated task h_{i+M} can be computed through (4). A similar method is used to assign a frequency to the duplicated tasks. The main difference is that, when assigning frequency to the duplicated tasks, the selected frequency should satisfy the reliability constraint (5) while providing the minimum energy increase.

Algorithm 2 Task Allocation and Scheduling

1: **Input**: y_{il} and h_i;
2: **Output**: x_{ik}, u_{ij} and t_i^s;
3: **Initialize**: $O[i] = -1$ $(\forall i \in \mathcal{M}')$;
4: Sort tasks according to their in-degree and out-degree;
5: **for** $\forall i \in \mathcal{M}'$ $(h_i = 1)$ **do**
6: $MinMaxEng = \infty$;
7: **for** $\forall k \in \mathcal{N}$ **do**
8: Calculate E_k^{comm} and E_k^{comp};
9: $MaxEng = \max_{\forall k}\{E_k^{comm} + E_k^{comp}\}$;
10: **if** $MaxEng < MinMaxEng$ **then**
11: $MinMaxEng = MaxEng$;
12: $O[i] = k$;
13: **else**
14: Continue;
15: **end if**
16: **end for**
17: Calculate x_{ik} according to $O[i] = k$;
18: Calculate u_{ij} and t_i^s according to x_{ik}, t_i^{comm} and t_i^{comp};
19: **end for**

2) Task Allocation and Scheduling: With the frequency assignment y_{il} and task duplication h_i, the computation time and energy of task τ_i are $t_i^{comp} = h_i \sum_l y_{il} \frac{C_i}{f_l}$ and $e_i^{comp} = h_i \sum_l y_{il} \frac{C_i}{f_l} P_l$, respectively. The next step is to determine the task allocation x_{ik}, task sequence u_{ij} and task start time t_i^s. To balance the energy consumption of the processors under the task sequence and the task non-overlapping constraints, the task allocation and scheduling problem is given by:

$$\mathbf{P3}: \min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{t^s}} (\max_{\forall k}\{E_k^{comm} + E_k^{comp}\}) \quad \text{s.t. } (1), (6), (7). \tag{12}$$

Note that the communication energy E_k^{comm} and communication time t_i^{comm} are influenced by the path selection $c_{\beta\gamma\rho}$. However, $c_{\beta\gamma\rho}$ is unknown at the current step. To formulate the problem (12), we fix the values of E_k^{comm} and t_i^{comm} to the average communication time of task τ_i and average communication energy of processor θ_k, respectively, i.e., $t_i^{comm} = M_1(\max_{\forall\beta,\gamma,\rho}\{t_{\beta\gamma\rho}\} + \min_{\forall\beta,\gamma,\rho}\{t_{\beta\gamma\rho}\})/2$ and $E_k^{comm} = M_2(\max_{\forall\beta,\gamma}\{e_{\beta\gamma k1}\} + \min_{\forall\beta,\gamma}\{e_{\beta\gamma k2}\})/2$. M_1 is the number of predecessors of τ_i, while M_2 is the number of original and duplicated tasks. Once the path selection $c_{\beta\gamma\rho}$ is determined, the value of t_i^{comm} and E_k^{comm} is updated accordingly. Based on the structure of MINLP problem (12), we design Algorithm 2:

a. A task allocation index $O[i]$ is introduced for each task τ_i $(h_i = 1)$ and it is initialized as $O[i] = -1$. If task τ_i is allocated to processor θ_k, we have $O[i] = k$.
b. The in- and out-degrees of all tasks are calculated and the tasks are divided into layers. For instance, in Fig. 1(c), tasks (τ_1, τ_4), (τ_2, τ_5) and (τ_3, τ_6) are assigned to the first (L_1), the second (L_2) and the third (L_3) layers, respectively. Tasks in the same layer are sorted in a descending order based on their execution cycles. If the tasks in same layer have same execution cycles, they are ordered randomly.
c. Algorithm 2 follows the sequence provided in the previous step to perform task allocation. With this sequence,

Algorithm 3 Path selection

1: **Input**: y_{il}, h_i, x_{ik}, u_{ij} and t_i^s;
2: **Output**: $c_{\beta\gamma\rho}$ and t_i^s;
3: **Initialize**: $Q[\beta][\gamma] = -1$ $(\forall \beta, \gamma \in \mathcal{N})$;
4: **for** $\forall(\beta, \gamma) \in \mathcal{N}$ **do**
5: $MinMaxCost = \infty$;
6: **for** $\forall \rho \in \mathcal{P}$ **do**
7: $Q[\beta][\gamma] = \rho$;
8: Calculate t_i^s, t_i^{comm} and t_i^e $(x_{i\gamma} = 1)$;
9: $tMax = \max_{\forall i}\{t_i^e\}$;
10: **if** $tMax > H$ **then**
11: Continue;
12: **else**
13: Calculate E_k^{comm};
14: $MinMaxCost = \max_{\forall k}\{E_k^{comm} + E_k^{comp}\}$;
15: **if** $MaxCost < MinMaxCost$ **then**
16: $MinMaxCost = MaxCost$;
17: $flag = \rho$;
18: **else**
19: Continue;
20: **end if**
21: **end if**
22: **end for**
23: $Q[\beta][\gamma] = flag$;
24: **end for**

the task sequence constraint (6) and task non-overlapping constraint (7) are handled at the same time.

d. Algorithm 2 iteratively allocates a task τ_i ($h_i = 1$) to a processor θ_k in order to balance the energy consumption of processors, i.e., minimize the maximum computation and communication energy of the processors (Lines 8–15).

3) Multi-path Selection: The final phase determines the path selection $c_{\beta\gamma\rho}$. Note that $c_{\beta\gamma\rho}$ does not influence the computation energy E_k^{comp} and time T_i^{comp}, only the communication energy E_k^{comm} and time T_i^{comm}. To balance the energy consumption of the processors, while meeting the real-time constraints, the path selection problem is formulated as:

$$\mathbf{P4}: \min_{\boldsymbol{c}}(\max_{\forall k}\{E_k^{comp} + E_k^{comm}\}) \quad (13)$$
$$\text{s.t. } (2), (9).$$

To solve the above ILP problem, we propose Algorithm 3.

a. A path selection index $Q[\beta][\gamma]$ is used for each processor pair $(\theta_\beta, \theta_\gamma)$, and initialized as $Q[\beta][\gamma] = -1$. $Q[\beta][\gamma] = \rho$ is the data transmission from θ_β to θ_γ, through the ρ^{th} path.

b. The routing path is determined iteratively for each pair of processors $(\theta_\beta, \theta_\gamma)$. The aim is to find a path for θ_β and θ_γ, that causes the minimum increase of communication and computation energy among these processors, i.e., $\min(\max_{\forall k}\{E_k^{comp} + E_k^{comm}\})$ (Lines 13–20). During this process, the real-time constraint (9) should be satisfied.

IV. Evaluation

The evaluation is performed considering a 4×4 2D-mesh NoC. The modeling of the energy consumed by processors and routers is based on [3]. The following parameters are considered in our experiments: the number of processors (N), the number of the tasks (M), the number of V/F levels (L).

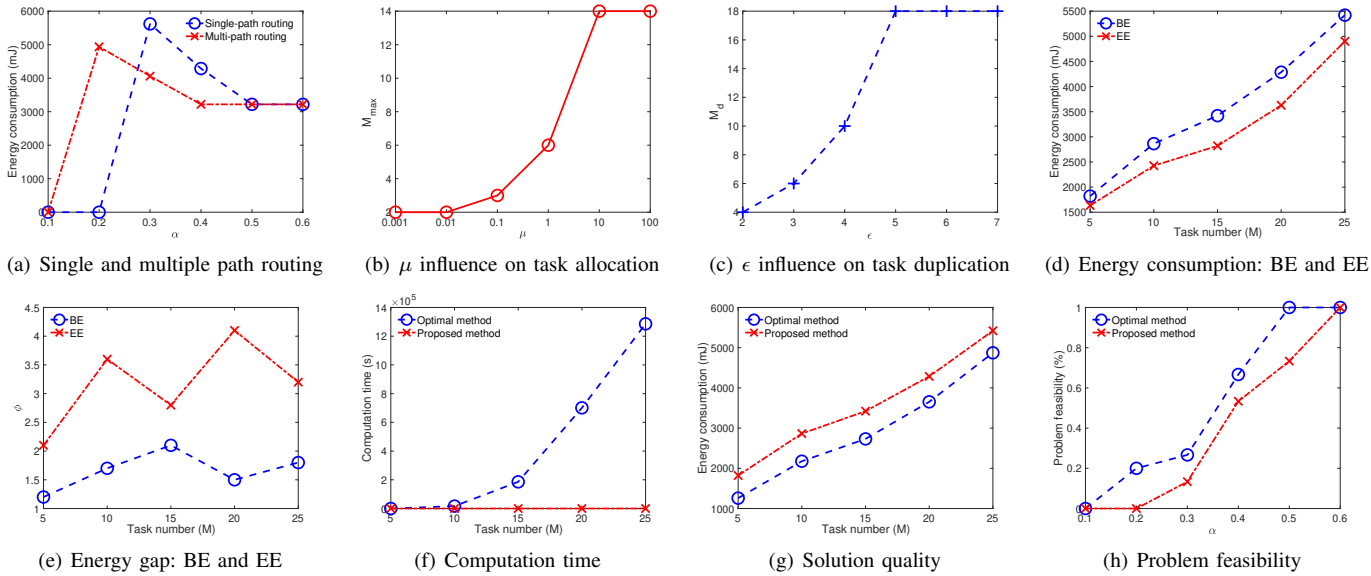
Fig. 2(a) compares the energy consumption and the feasibility, considering multi-path routing (described by (10)) and single-path routing. Compared to multi-path routing, the path selection $c_{\beta\gamma\rho}$ is fixed in single-path routing. Both task deployment problems are optimally solved by Gurobi. We set $N = 16$, $M = 20$, $L = 6$ and $H = \alpha \sum_{i \in \mathcal{C}}(t_{i,ave}^{comp} + t_{i,ave}^{comm})$, where $\mathcal{C}$ is the set of tasks belonging to the critical path. $t_{i,ave}^{comp} = (\max_{\forall l}\{\frac{C_i}{f_l}P_l\} + \min_{\forall l}\{\frac{C_i}{f_l}P_l\})/2$ and $t_{i,ave}^{comm} = M_1(\max_{\forall \beta,\gamma,\rho}\{t_{\beta\gamma\rho}\} + \min_{\forall \beta,\gamma,\rho}\{t_{\beta\gamma\rho}\})/2$ are the average computation time and communication time of task τ_i, respectively. Fig. 2(a) shows that with small α, e.g., $\alpha = 0.1$ or $\alpha = 0.2$, the problem is infeasible, since the constraints are hard to satisfy. The problem feasibility increases with α; the larger the value of α, the smaller the energy consumption. This is because the feasibility region of the problem enlarges with α, and the task deployment is a minimization problem. Fig. 2(a) also shows that under the same value of α, multi-path routing has a higher problem feasibility than single-path routing. Multi-path routing achieves a lower energy consumption because the path selection $c_{\beta\gamma\rho}$ is considered in the optimization. Thus, it can find a better path selection, further improving the energy efficiency, compared with single-path routing.

Fig. 2(b) shows the influence of processor parameters on task allocation decision x_{ik}. We introduce $M_{\max} = \max_{\forall k}\{M_k\}$, where M_k is the number of tasks allocated to processor θ_k, and $\mu = e_k^{comm}/e_k^{comp}$, where $e_k^{comm} = \max_{\forall \beta,\gamma,k,\rho}\{e_{\beta\gamma k\rho}\}$ and $e_k^{comp} = \max_{\forall i,l}\{\frac{C_i}{f_l}P_l\}$ are the processor parameters regarding communication and computation energy. The larger the value of μ, the more energy is consumed in data transmission than task execution. Fig. 2(b) shows that $M_{\max}$ increases with μ; when tasks have a large communication energy, the dependent tasks are allocated to the same processor to reduce the cost.

Fig. 2(c) explores the influence of processor parameters on task duplication decision h_i, where M_d represents the number of duplicated tasks for M original tasks, and $\epsilon = \max_{\forall l}\{\frac{P_l}{f_l}\}/\min_{\forall l}\{\frac{P_l}{f_l}\}$ is an index that represents the gap regarding task execution energy. Note that $t_i^{comp} = \sum_{l \in \mathcal{L}} y_{il}\frac{C_i}{f_l}P_l$ is the energy required to execute task τ_i. The larger the value of ϵ, the more energy is consumed because of task execution with high frequency, compared to task execution with low frequency. Fig. 2(c) shows that M_d increases with ϵ. When ϵ is small, the execution of one task (original task) with high frequency is more energy efficient than executing two tasks (original and duplicated tasks) with low frequency. With ϵ increasing, executing two tasks with low frequency becomes more efficient than executing one task with high frequency.

Fig. 2(d) and Fig. 2(e) compare the energy consumption of the proposed task deployment scheme, with the goal of Balancing the Energy consumption (BE), and the task deployment scheme with the goal of Minimizing the Energy consumption (ME), i.e., $\min \sum_{k \in \mathcal{N}} E_k^{all}$, where $E_k^{all} = E_k^{comm} + E_k^{comp}$ is the total energy of processor θ_k. To evaluate the performances of these schemes, we introduce an index $\phi = \max_{\forall k}\{E_k^{all}\}/\min_{\forall k}\{E_k^{all}\}$, where $E_k^{all} \neq 0$. The smaller the value of ϕ, the more balance is achieved regarding energy consumption of all processors. Fig. 2(d) and Fig. 2(e) show that the total energy consumption of ME is lower than BE (average 13.62%). However, the value of ϕ for BE is smaller than ME. This is because ME allocates the tasks to the same processors to reduce communication energy. Therefore, some processors will consume more energy than others. However, with BE this trend can be avoided in order to achieve energy balance.

Fig. 2(f) and Fig. 2(g) compare the solutions of problem (10)



(a) Single and multiple path routing (b) μ influence on task allocation (c) ϵ influence on task duplication (d) Energy consumption: BE and EE

(e) Energy gap: BE and EE (f) Computation time (g) Solution quality (h) Problem feasibility

Fig. 2. System performance with different parameters varying.

achieved by the proposed heuristic and the optimal solution, obtained by the Gurobi solver. From Fig. 2(f), we observe that the algorithm computation time increases with task number M, as more variables and constraints are involved. On the contrary, the proposed heuristic has a negligible computation time, since it divides the problem into three subproblems, i.e., P2, P3 and P4, which are solved in sequence. From Fig. 2(g), we observe that the solution of the heuristic has a higher, but still acceptable, energy consumption (average 26.05%) than the optimal solution, since it only provides a feasible solution.

Fig. 2(h) shows the feasibility in solving the task deployment problem (10) for the optimal and the heuristic methods. The experiments have been repeated $n_a = 30$ times with different task graphs. The used metric is the problem feasible ratio $\delta = n_f/n_a$, where n_f is the number of experiments with feasible solutions. Fig. 2(h) shows that δ increases with α. The constraints are relaxed with α, thus the feasible region of problem is enlarged. The optimal feasibility is higher than the heuristic, since it optimizes the variables concurrently, whereas the heuristic optimizes the variables step by step.

V. Conclusion

This work proposes a task deployment process for NoC-based multicore platforms with DVFS, that balances the overall energy consumption, under reliability and real-time constraints. The deployment process is formulated as an MINLP problem and equivalently transformed to an MILP problem. Our formulation jointly optimizes frequency assignment, task allocation, task scheduling, task duplication and path selection. Moreover, a novel heuristic method is proposed to enhance scalability, achieving good solutions with low computation time.

Acknowledgment

This work was partially supported by Fundamental Research Funds for the Central Universities (Grants 2242021R10113 and MCCSE2021B02), and by Southeast University "Zhishan Scholars" (Grant 2242021R40003).

References

[1] H. Ali, U. U. Tariq, and J. H. et al., "A survey on system level energy optimisation for MPSoCs in IoT and consumer electronics," *Computer Science Review*, vol. 41, p. 100416, 2021.

[2] P. V. Bhanu, P. V. Kulkarni, and J. Soumya, "Fault-tolerant network-on-chip design with flexible spare core placement," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 1, 2019.

[3] 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, 2015.

[4] 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. Syst.*, vol. 20, no. 8, pp. 1496–1509, 2012.

[5] 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, 2017.

[6] 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.

[7] C. Gou, A. Benoit, M. Chen, L. Marchal, and T. Wei, "Reliability-aware energy optimization for throughput-constrained applications on MPSoCs," in *IEEE ICPADS*, 2018, pp. 1–10.

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

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

[10] 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.

[11] 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.

[12] M. N. S. M. Sayuti and L. S. Indrusiak, "Real-time low-power task mapping in networks-on-chip," in *IEEE ISVLSI*, 2013, pp. 14–19.

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

[14] N. Chatterjee, S. Paul, and S. Chattopadhyay, "Fault-tolerant dynamic task mapping and scheduling for network-on-chip-based multicore platform," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 4, 2017.

[15] J. Joven, A. Bagdia, F. Angiolini, P. Strid, D. Castells-Rufas, E. Fernandez-Alonso, J. Carrabina, and G. De Micheli, "QoS-driven reconfigurable parallel computing for NoC-based clustered MPSoCs," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1613–1624, 2013.

[16] 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.