PAPER

A Two-Phase Algorithm for Reliable and Energy-Efficient Heterogeneous Embedded Systems

Hongzhi XU[†a)] ***and*** **Binlian ZHANG**[†b)]**,** ***Nonmembers***

SUMMARY Reliability is an important figure of merit of the system and it must be satisfied in safety-critical applications. This paper considers parallel applications on heterogeneous embedded systems and proposes a two-phase algorithm framework to minimize energy consumption for satisfying applications' reliability requirement. The first phase is for initial assignment and the second phase is for either satisfying the reliability requirement or improving energy efficiency. Specifically, when the application's reliability requirement cannot be achieved via the initial assignment, an algorithm for enhancing the reliability of tasks is designed to satisfy the application's reliability requirement. Considering that the reliability of initial assignment may exceed the application's reliability requirement, an algorithm for reducing the execution frequency of tasks is designed to improve energy efficiency. The proposed algorithms are compared with existing algorithms by using real parallel applications. Experimental results demonstrate that the proposed algorithms consume less energy while satisfying the application's reliability requirements.

key words: *dynamic voltage and frequency scaling, energy consumption, heterogeneous embedded systems, parallel applications, reliability requirement*

1. Introduction

With the rapid development of integrated circuit and chip technology, the performance of the embedded system has been substantially improved. Yet along with such improvements also comes with more energy consumption. For energy conservation and environmental concerns, dynamic voltage and frequency scaling (DVFS) techniques have been widely used for low-power design. DVFS reduces energy consumption by scaling down the execution voltage and frequency of the processor at runtime [1]–[4]. At present, there are many mainstream processors, such as Intel, AMD, and ARM, that apply DVFS techniques to improve energy efficiency. However, the probability of the transient faults in processor will be significantly increased by scaling down the execution frequency, which will weaken the reliability of the applications [5]. The reliability of an application (task) is defined as the probability that the application (task) can be executed correctly [5], [6]. For many embedded systems, reliability is an important quality metric of the applications. There are many safety standards for reliability, such as ISO 26262 for automotive software systems and DO-178C for avionics software [6]. Hence, satisfying reliability requirement of the application is crucial for embedded systems, otherwise, it may lead to disastrous consequences [7].

In general, the enhancement of the systems' reliability may come with more energy consumption. Therefore, a tradeoff has to be made between the degree of reliability and energy consumption. Specifically, energy consumption should be as little as possible when the reliability requirement is satisfied in the system design phase. Many well-designed algorithms are proposed to improve energy efficiency while guaranteeing the reliability of the applications [5], [7]–[11]. However, these works are mainly based on single-processor systems. In recent years, with the advance of cyber-physical systems [12], [13], many energy-efficient algorithms with DVFS techniques have been proposed for heterogeneous systems [14], [15]. However, these algorithms do not consider the application's reliability. In order to make the application up to reliability requirement, many researchers focus on the problem of minimizing system resource (energy) consumption while satisfying the reliability requirement of the application. Some algorithms have been proposed for parallel applications on heterogeneous systems [6], [16]–[18], which first transfer the reliability requirement of the application to the reliability requirement of each task and then use the heuristic method to minimize the resources (energy) consumption. These methods may not be the most effective approach for improving energy efficiency. In this paper, a novel approach is proposed to minimize energy consumption while satisfying the reliability requirement of the parallel applications on heterogeneous embedded systems.

This paper focuses on the design phase of the system, the main contributions are as follows:

(1) This paper proposes a two-phase algorithm framework to satisfy applications' reliability requirements. The framework contains two phases. Specifically, for each application, the first phase initiates task assignment on processors with the minimum energy consumption so that the application's initial reliability can be acquired. After that, based on the actual reliability, the second phase satisfies the reliability requirement and improves the energy efficiency.

(2) This paper proposes an algorithm to improve an application's reliability if it is lower than the reliability requirement after the initial assignment so that the reliability requirement can be finally satisfied.

(3) This paper proposes an algorithm to reduce energy consumption through DVFS techniques if the initial reliability is already higher than the reliability requirement after the

Manuscript received December 11, 2023.
Manuscript revised April 11, 2024.
Manuscript publicized May 27, 2024.

[†]College of Computer Science and Engineering, Jishou University, Zhangjiajie 427000, China.
a) E-mail: xuhongzhi9@163.com
b) E-mail: zhangbinlian@163.com

DOI: 10.1587/transinf.2023EDP7262

initial assignment so that the application's energy efficiency can be improved.

(4) Experiments with real parallel applications are conducted in different scenarios. Experimental results confirm that the proposed algorithms consume less energy than other approaches.

2. Related Work

In order to reduce the energy consumption while guaranteeing reliability of the system, many techniques have been proposed. Zhu et al. [5] proposed a reliability-aware power management framework for periodic tasks. In [7], when the slack time is not enough to schedule a recovery for a whole task, checkpointing techniques are used to reliability-aware power management schemes. Zhao et al. [8] presented a shared-recovery based frequency assignment technique to minimize energy consumption while preserving the system reliability. Fan et al. [10] considered the single processor system with independent real-time tasks, proposed reliability-aware dynamic power management algorithm under reliability constraint to minimize the energy consumption. Xu et al. [11] designed several energy-efficient scheduling algorithms with reliability guarantee for a set of periodic real-time tasks on single processor system. Aforementioned techniques are based on single processor systems and DVFS is used to improve energy efficiency.

In recent years, there have been a lot of works on heterogeneous systems. Mei et al. [19] designed a fault-tolerant dynamic rescheduling algorithm for heterogeneous computing systems. Wang et al. [20] proposed a scheduling algorithm based on replication for maximizing system reliability. Zhang et al. [21] also investigated bi-objective genetic scheme to optimize energy efficient and reliability for a parallel application on heterogeneous computing systems. Zhao et al. [16] proposed MaxRe algorithm, which using a dynamic number of replicas for different tasks to satisfy the reliability requirements of the application. To further reduce the resources consumption, Zhao et al. [17] also presented a RR algorithm for workflow applications on heterogeneous systems, which consume fewer resources to satisfy the reliability requirements. Xie et al. [6] designed MRCRG algorithm for a DAG-based parallel application on heterogeneous embedded systems, the algorithm first calculates the reliability requirement of each task, and then assigns the task to the processor with the least resource consumption. Xie et al. [18] also introduced the ESRG algorithm, which uses DVFS technique to minimize energy consumption while satisfying the reliability requirement of a parallel application. Xu et al. [22] designed two algorithms to guarantee the reliability of applications. Kumar et al. [23] introduced the framework to satisfy the reliability of non-preemptive periodic tasks on heterogeneous systems. Liu et al. [24] proposed a task replication method to maximize the reliability of the application on heterogeneous systems. Peng et al. [25] proposed a reliability and performance-aware scheduling with energy constraint for parallel applications on heterogeneous systems. In addition, there are some surveys related to fault-tolerance techniques in [26].

The afoementioned MRCRG [6], MaxRe [16], RR [17], and ESRG [18] algorithms first transform the reliability requirements of parallel applications into those of their subtasks, subsequently employing heuristic algorithms to minimize energy (resource) consumption. However, such methods may not be the most effective approach for improving energy efficiency. Therefore, this paper proposes a novel two-phase algorithm framework that achieves lower energy consumption while satisfying the application's reliability requirements.

3. System Model and Problem Formulation

3.1 Heterogeneous System Model

The heterogeneous embedded system studied in this work consists of m processor nodes, which are represented as a set $PN = \{pn_1, pn_2, \ldots pn_m\}$. All processor nodes are connected via the bus and ignore data communication conflicts.

3.2 Application Model

From the system design perspective, parallel applications are composed of precedence-constrainted tasks, which are usually represented as directed acyclic graph (DAG) [4], [27], [28]. Similar to [6], [15], [18], [27], [29], the parallel application is modelled as DAG $A = (T, C)$, where T and C are explicated as follows:

$T = \{t_1, t_2, \ldots, t_n\}$ is a vertex set of DAG, which indicates that there are n tasks in application A. C is an edge set of DAG, which indicates the data communication relationship between tasks. The value of $c_{i,j}$ represents the time required for data transmission from t_i to t_j. If t_i and t_j are assigned to the same processor, then the data transmission time is 0. For conveniently describing the relationship between tasks in a parallel application, the direct predecessor tasks of t_i is denoted as $pred(t_i)$ and the direct successor tasks of t_i is denoted as $succ(t_i)$. An example of DAG is shown in Fig. 1, which consists of seven tsks and nine data transmission edges. The edge $c_{1,2} = 20$ indicates that if t_1 and t_2 are assigned to the different processor, then the data transmission time is 20; otherwise, the data transmission time is 0.

Because the processors are heterogeneous, the required execution time of the same task on different processor may be different. A two-dimensional array $n \times m$ is used to represent

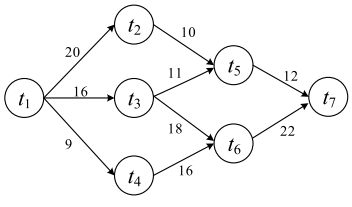


Fig. 1 An example of a DAG

Table 1 WCET of tasks on different processors of the motivation example

$task$	pn_1	pn_2	pn_3	$Rank$
t_1	16	12	18	107.7
t_2	12	13	17	59.3
t_3	13	11	16	76.3
t_4	10	15	12	73.3
t_5	13	10	14	35.3
t_6	11	16	9	45
t_7	15	7	11	11

the worst case execution time (WCET) of each task execution on different processor with maximum execution frequency. $w_{i,j}$ represents the WCET of the task t_i executes on processor pn_j. The WCET of ten tasks in Fig. 1 on three processor are shown in Table 1, where the WCET of t_1 on pn_1, pn_2, and pn_3 is 16, 12, and 18, respectively.

3.3 Energy Consumption Model

Based on [7], [9], [29], the power dissipation of the processor operates at frequency f is given by

$$\begin{aligned} P(f) &= P_{\text{sta}} + h(P_{\text{ind}} + P_{\text{dyn}}) \\ &= P_{\text{sta}} + h(P_{\text{ind}} + C_{\text{swi}} f^a). \end{aligned} \tag{1}$$

In (1), P_{sta} and P_{ind} represent the static power and the leakage power dissipation, respectively. P_{dyn} is the dynamic power dissipation, which is related to execution frequency of the processor. C_{swi} is the switching capacitance and a represents the dynamic energy exponent. When the processor state is active, $h = 1$; otherwise, $h = 0$.

According to [5], [29], the critical frequency which enables a minimum total energy consumption per execution cycle can be calculated by

$$f_{\text{cri}} = \sqrt[a]{\frac{P_{\text{ind}}}{C_{\text{swi}}(a-1)}}. \tag{2}$$

Assuming the available execution frequency of the processor is f_{min} to f_{max}, for improving energy efficiency, the lowest execution frequency of the processor should be

$$f_{\text{low}} = \max(f_{\text{cri}}, f_{\text{min}}). \tag{3}$$

Therefore, the discrete frequencies available for processor pn_j are $f_{j,\text{low}}$ to $f_{j,\text{max}}$.

When task t_i executes on processor pn_j with execution frequency $f_{j,k}$, the dynamic energy consumption $E_{\text{d}}(t_i, pn_j, f_{j,k})$ can be calculated by

$$\begin{aligned} &E_{\text{d}}(t_i, pn_j, f_{j,k}) \\ &= (P_{j,\text{ind}} + C_{j,\text{swi}} f^{a_j}) \times w_{i,j} \times \frac{f_{j,\text{max}}}{f_{j,k}} \end{aligned}, \tag{4}$$

where $P_{j,\text{ind}}$, $C_{j,\text{swi}}$, and a_j represent the leakage power dissipation, the switching capacitance, and the dynamic energy exponent of processor pn_j, respectively.

Considering that different tasks may be executed on different processors, the data transmission between two processors will consume energy. Assuming that the data transmission energy consumption is proportional to the transmission time, the energy consumption rate per unit time is expressed as ecr [6]. Therefore, when task t_i executes on processor pn_j, the data transmission energy consumption is given by

$$E_{\text{t}}(t_i, pn_j) = \sum\nolimits_{t_x \in pred(t_i)} ecr \times c_{x,i}. \tag{5}$$

In Eq. (5), if the task t_i and the direct predecessor task t_x are assigned to the same processor, then the value of $c'_{i,j}$ is 0; otherwise the value of $c'_{i,j}$ is $c_{i,j}$.

According to Eqs. (4) and (5), when the task t_i is assigned to a processor, the dynamic energy consumption can be calculated by

$$\begin{aligned} E_{\text{d}}(t_i) &= E_{\text{d}}(t_i, pn_{ap(i)}, f_{ap(i),af(i)}) \\ &\quad + E_{\text{t}}(t_i, pn_{ap(i)}), \end{aligned} \tag{6}$$

where $pn_{ap(i)}$ and $f_{ap(i),af(i)}$ represent the assigned processor and corresponding execution frequency of task t_i, respectively.

From the above analysis, when all tasks in application A are assigned to the processor, the total dynamic energy consumption can be calculated by

$$E_{\text{d}}(A) = \sum\nolimits_{i=1}^{n} E_{\text{d}}(t_i). \tag{7}$$

The static energy consumption of application A is denoted as $E_{\text{s}}(A)$, which can be calculated by

$$E_{\text{s}}(A) = \sum\nolimits_{j=1}^{m} P_{j,\text{sta}} \times SL(A), \tag{8}$$

where $SL(A)$ is scheduling length of application A which is the time from the beginning of the first task execution to the completion of the last task. According to Eqs. (7) and (8), the total energy consumption of the application can be given by

$$E(A) = E_{\text{d}}(A) + E_{\text{s}}(A). \tag{9}$$

3.4 Reliability Model

The faults might occur during the execution of an application, which can be divided into transient faults and permanent faults. Since transient faults occur more commonly than permanent faults [5], [7], only transient faults are considered in this study. Assume that the probability of transient faults during task execution follows the Poisson distribution [5]–[7], [18]. When task t_i is assigned to processor pn_j with the maximum execution frequency, the reliability of which is given by

$$R(t_i, pn_j, f_{j,\text{max}}) = e^{-\lambda_{j,\text{max}} \times w_{i,j}}, \tag{10}$$

where $\lambda_{j,\text{max}}$ indicates the transient fault rate per unit time of processor pn_j execution with the maximum frequency.

According to [5], [7], [9], [29], the transient faults rate $\lambda_{j,k}$ of the processor pn_j with execution frequency $f_{j,k}$ is given by

$$\lambda_{j,k} = \lambda_{j,\max} \times 10^{\frac{d_j \times (f_{j,\max} - f_{j,k})}{f_{j,\max} - f_{j,\min}}}, \tag{11}$$

where d_j represents the sensitivity fault rate to corresponding voltage and frequency scaling of the processor pn_j, which is greater than 0.

Based on Eqs. (10) and (11), when task t_i is assigned to the processor pn_j with execution frequency $f_{j,k}$, the reliability of task t_i is given by

$$\begin{aligned} R(t_i) &= R(t_i, pn_j, f_{j,k}) \\ &= e^{-\lambda_{j,\max} \times 10^{\frac{d_j \times (f_{j,\max} - f_{j,k})}{f_{j,\max} - f_{j,\min}}} \times \frac{w_{i,j} \times f_{j,\max}}{f_{j,k}}}. \end{aligned} \tag{12}$$

When all the tasks are assigned to the processor, the reliability of the application is given by

$$R(A) = \prod_{i=1}^{n} R(t_i, pn_{ap(t_i)}, f_{ap(t_i),af(t_i)}). \tag{13}$$

Based on Eq. (12), the obtainable maximum reliability of task t_i can be calculated by

$$R_{\max}(t_i) = \max_{pn_j \in PN} \{R(t_i, pn_j, f_{j,\max}))\}. \tag{14}$$

If all the tasks are assigned with the obtainable maximum reliability, the reliability of the application will reach the maximum, which can be calculated by

$$R_{\max}(A) = \prod_{i=1}^{n} R_{\max}(t_i). \tag{15}$$

3.5 Problem Description

Considering a parallel application A execution on a heterogeneous embedded system, the reliability requirement of application is $R_{\text{req}}(A) \leq R_{\max}(A)$. The problem of this work is to determine the processor and frequency assignments for each task in A, so that the total energy consumption can be minimized while the application's reliability requirement is satisfied. The formal description is to minimize

$$E(A) = E_{\text{d}}(A) + E_{\text{s}}(A) \tag{16}$$

subject to

$$R(A) \geq R_{\text{req}}(A). \tag{17}$$

4. The Proposed Algorithms

Because the reliability of the application is determined by the reliability of all the tasks within the application, the reliability requirement of the application can be transferred into the reliability requirement of each task, such as the MRCRG algorithm [6], the MaxRe algorithm [16], the RR algorithm [17], and the ESRG algorithm [18]. The proposed algorithm framework in this study is different from the algorithms mentioned above, which is divided into two phases to minimize energy consumption while satisfying the reliability requirement. The first phase is for initial assignment and the second phase is for either satisfying the reliability requirement or improving energy efficiency. Initial assignment is to make each task reach the highest reliability with a certain minimum energy consumption. In the second phase, when the reliability of the application obtained in the first phase is lower than the reliability requirement, then the algorithm must enhance the reliability of certain tasks until the reliability requirement of the application is satisfied; otherwise, the DVFS technique can be used to improve energy efficiency.

Similar to [6], [18], [27], the task topology order is generated using the *Rank* value, which is defined as

$$Rank(t_i) = \overline{w_i} + \max_{t_j \in succ(t_i)} \{c_{i,j} + Rank(t_j)\}. \tag{18}$$

In Eq. (18), the $\overline{w_i}$ is average WCET of the task t_i on each processor, which can be calculated by

$$\overline{w_i} = \left(\sum_{j=1}^{M} w_{i,j}\right) \Big/ m. \tag{19}$$

The *Rank* values of each task of application in motivational example are shown in Table 1, the task with a greater the *Rank* value has a higher priority. Therefore, before the application is executed, we sort the task in a non-increasing order of *Rank* values, which can make the task-execution in the correct order.

4.1 Initial Assignment

The algorithm of the first phase is named initial assignment with minimum energy consumption (IMEC). Based on aforementioned, the IMEC algorithm is shown in Algorithm 1.

The main objective of the IMEC algorithm is to achieve high reliability with relatively low energy consumption. For task t_i, IMEC first assigns it to the processor with the lowest energy consumption without applying DVFS, so that the minimum energy consumption e and corresponding reliability can be calculated. Then, IMEC traverses other processor and frequency combinations, searching whether task t_i can achieve higher reliability when the energy consumption is less than or equal to e.

The details of the IMEC algorithm are as follows. Line 1 initializes the scheduling queue qt, while lines 2-26 use a **while** loop to schedule each task. For task t_i, IMEC traverses all processors to find the minimum energy consumption e required to execute task t_i without using DVFS, as well as its corresponding reliability r (Lines 5-13). Lines 14-25 are double-layer **for** loops used to traverse the combination of processor and execution frequency. Line 16 calculates the energy consumption of task t_i. If the energy consumption is less than or equal to e and the reliability is higher than r (Lines 17-19), the processor and execution frequency for executing task t_i are updated (Line 21).

The time complexity of the IMEC algorithm is analyzed as follows. For each task t_i, IMEC calculates the minimum energy consumption e and the corresponding reliability r

Algorithm 1 IMEC

Require: $PN = \{pn_1, pn_2, ...pn_m\}$, A and $R_{req}(A)$
Ensure: $R(A)$
1: sort the tasks to queue qt by non-increasing order of $Rank$ value
2: **while** qt is not empty **do**
3: $t_i \leftarrow queuet.out()$
4: $e \leftarrow \infty$
5: **for** each processor $pn_j \in PN$ **do**
6: calculate $E_{\mathrm{d}}(t_i)$ using Eq. (6)
7: **if** $E_{\mathrm{d}}(t_i) < e$ **then**
8: $e \leftarrow E_{\mathrm{d}}(t_i)$
9: calculate $R(t_i)$ using Eq. (12)
10: $r \leftarrow R(t_i)$
11: $ap(i) \leftarrow j$
12: **end if**
13: **end for**
14: **for** each processor $pn_j \in PN$ and $j \neq ap(i)$ **do**
15: **for** frequency $f_{j,k}$ from $f_{j,\mathrm{low}}$ to $f_{j,\mathrm{max}}$ **do**
16: calculate $E_{\mathrm{d}}(t_i)$ using Eq. (6)
17: **if** $E_{\mathrm{d}}(t_i) \leq e$ **then**
18: calculate $R(t_i)$ using Eq. (12)
19: **if** $R(t_i) > r$ **then**
20: $r \leftarrow R(t_i)$
21: $ap(i) \leftarrow j$ and $af(i) \leftarrow k$
22: **end if**
23: **end if**
24: **end for**
25: **end for**
26: **end while**
27: calculate $R(A)$ using Eq. (13)

can be completed in O($n \times m$) time. Then, IMEC traverses other processors and frequency levels, which can be done in O($n \times m \times fs$) time, where fs represents the maximum number of available discrete frequencies of the processor. Therefore, the time complexity of IMEC is O($n^2 \times m \times fs$).

The IMEC algorithm obtains the initial reliability of the application. If the reliability requirement is higher than that obtained by IMEC, it is necessary to enhance reliability of certain tasks until the reliability requirement of the application is satisfied. In the following, we will enhance the reliability of some tasks to satisfy the reliability requirement of the application.

4.2 Enhancing the Reliability of Tasks

In order to enhance the reliability of the application, we first give the method for choosing tasks that need enhanced reliability.

(1) We choose tasks with relatively low reliability to enhance. This method can quickly enhance the reliability of the application.

(2) We choose tasks that have great potential for enhancing reliability. This method may not increase too much energy consumption when improving the reliability of the task, because there are more options for allocatable processors and frequency levels.

The following introduces how to choose tasks that have the potential to enhance reliability.

Define the current reliability ratio (CRR), which is the ratio of the current reliability of the task to the maximum reliability of the task.

$$CRR(t_i) = \frac{R_{\mathrm{cur}}(t_i)}{R_{\mathrm{max}}(t_i)} \tag{20}$$

In Eq. (20), $R_{\mathrm{cur}}(t_i)$ represents the current reliability of task t_i. Obviously, $CRR(t_i)$ is less than or equal to 1. $CRR(t_i) = 1$ indicates that task t_i has been assigned with the maximum reliability. If CRR values of all tasks equal to 1, the reliability of application will reach $R_{\mathrm{max}}(A)$; otherwise, the reliability of application is lower than $R_{\mathrm{max}}(A)$. It is not difficult to find that tasks with small CRR value has great potential to improve reliability.

When the IMEC algorithm is completed, the reliability and initial CRR value of all the tasks can be obtained. For example, we assume that the power and reliability parameters for all processors are known and shown in Table 2, where the maximum frequency $f_{j,\mathrm{max}}$ for each processor is normalized to 1.0. The maximum reliability of the application in motivational example is 0.9750174, which can be obtained by Eq. (15). Table 3 shows the initial CRR value of the tasks in motivation application when the IMEC algorithm is completed.

Table 2 Power and reliability parameters of processors

pn_j	$P_{j,\mathrm{sta}}$	$P_{j,\mathrm{ind}}$	$C_{j,\mathrm{swi}}$	a_j	$\lambda_{j,\mathrm{max}}$	d_j
pn_1	0.01	0.03	1.2	2.8	0.0003	2.1
pn_2	0.01	0.05	1.0	2.6	0.0004	2.0
pn_3	0.01	0.07	0.9	2.7	0.0006	2.2

Table 3 The initial CRR value of the tasks in motivation application when the IMEC algorithm is completed.

t_i	pn_j	Frequency	$R(t_i)$	$R_{\mathrm{max}}(t_i)$	CRR
t_1	2	1	0.9952115	0.9952115	1
t_3	2	1	0.9956097	0.9961076	0.9995001
t_4	1	0.9	0.9933712	0.9970045	0.9963558
t_2	2	1	0.9948135	0.9964065	0.9984013
t_6	3	1	0.9946146	0.9967054	0.9979022
t_5	2	1	0.9960080	0.9961076	0.9999
t_7	2	1	0.9972039	0.9972039	1

As shown in Table 3, the CRR value of the tasks t_2, t_3, t_4, t_5, and t_6 are less than 1, which indicates that the reliability of these tasks can be enhanced. When the reliability of a task is enhanced, the reliability of the application will be enhanced. Assuming that the reliability of task t_i is enhanced from $R_{\mathrm{cur}}(t_i)$ to $R_{\mathrm{new}}(t_i)$, the enhanced reliability of application can be calculated by

$$R(A) = \frac{R_{\mathrm{cur}}(A)}{R_{\mathrm{cur}}(t_i)} \times R_{\mathrm{new}}(t_i), \tag{21}$$

where $R_{\mathrm{cur}}(A)$ represents the current reliability of the application before task enhanced.

In addition, if task t_i is reassigned to a new processor, it will affect the communication energy consumption of directly successor tasks, which can be calculated by

$$E_{\mathrm{succ}}(t_i, pn_j) = \sum\nolimits_{t_x \in succ(t_i)} ecr \times c_{i,x}. \tag{22}$$

Based on the above analysis, the algorithm for enhancing the reliability of tasks (ERT) is described as Algorithm 2.

The main ideas of the ERT algorithm are as follows. When the reliability of the application is lower than the reliability requirement, ERT repeatedly chooses the task to enhance the reliability until the reliability requirement of the application is satisfied. The details of main idea are explained as follows.

(1) Line 2 chooses an appropriate task t_i that has relatively low reliability and great potential for enhancing reliability (call the LRGP algorithm).

(2) If task t_i is executed at a lower frequency, ERT will increase its execution frequency (Lines 3-4). Otherwise, ERT traverses all processes and corresponding frequency levels to search for combinations with reliability greater than $R(t_i)$ and minimum energy consumption (Lines 6-19).

(3) Line 21 updates $R(A)$.

Algorithm 2 ERT

Require: results of IMEC and $R_{\mathrm{req}}(A)$
Ensure: $R(A)$ and $E(A)$
1: **while** $R(A) < R_{\mathrm{req}}(A)$ **do**
2: call the LRGP algorithm to obtain the task t_i that need to enhance reliability
3: **if** t_i with the reduced execution frequency **then**
4: increase the execution frequency of t_i by one level
5: **else**
6: $e = \infty$
7: **for** each processor $pn_j \in PN$ and $j \neq ap(i)$ **do**
8: **for** frequency $f_{j,k}$ from $f_{j,\mathrm{low}}$ to $f_{j,\mathrm{max}}$ **do**
9: calculate $R_{\mathrm{new}}(t_i)$ using Eq. (12)
10: **if** $R_{\mathrm{new}}(t_i) > R(t_i)$ **then**
11: calculate $E_{\mathrm{new}}(t_i)$ using Eq. (6)
12: calculate $E_{\mathrm{succ}}(t_i)$ using Eq. (22)
13: **if** $E_{\mathrm{new}}(t_i) + E_{\mathrm{succ}}(t_i) < e$ **then**
14: $e = E_{\mathrm{new}}(t_i) + E_{\mathrm{succ}}(t_i)$
15: $ap(i) \leftarrow j$ and $af(i) \leftarrow k$
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: **end if**
21: update $R(A)$ using Eq. (21)
22: **end while**
23: calculate $E(A)$ using Eq. (9)

In the ERT algorithm, the **while** loop is executed for one iteration. Because each iteration will enhance the reliability of a task, the reliability of the application can be constantly enhanced until $R(A) \geq R_{\mathrm{req}}(A)$ by using iteration.

The time complexity of ERT is analyzed as follows. During each iteration, obtaining task t_i has time complexity O(n) (the LRGP algorithm). In Lines 7-19, ERT needs to traverse all the processors and all corresponding execution frequency levels to find the suitable processor with higher reliability and minimum energy consumption, which has time complexity O($n \times m \times fs$). Because there are n tasks and m processors, the maximum number of iterations is $n \times m \times fs$. Therefore, the worst case time complexity of the ERT algorithm is O($n^2 \times m^2 \times fs^2$).

Algorithm 3 LRGP

Require: results of current scheduling
Ensure: t_i
1: traverse the task queue to find the first task t_k whose CRR value is not equal to 1
2: $i \leftarrow k$
3: **for** $x \leftarrow k+1$ to n **do**
4: **if** $R(t_x) < R(t_i)$ and $CRR(t_x) < CRR(t_i)$ **then**
5: $i \leftarrow x$
6: **end if**
7: **end for**

4.3 Improving Energy Efficiency

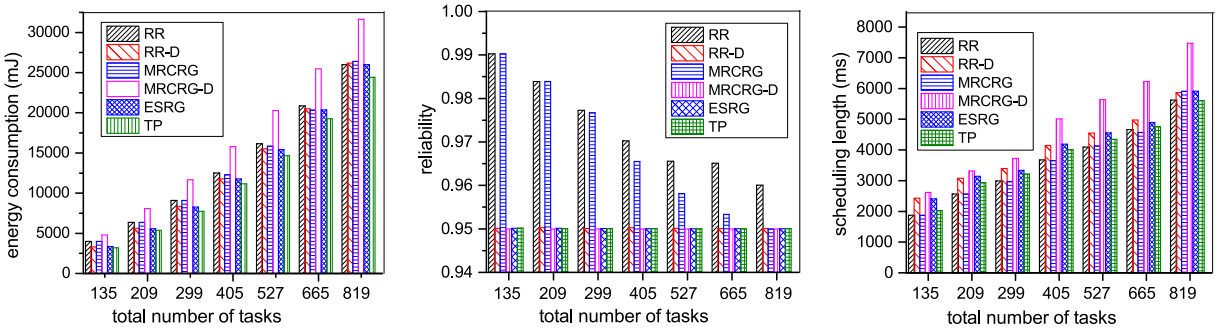
Sometimes, the reliability requirement of application may be lower than the reliability generated by the IMEC algorithm. In this case, we can reduce the frequency of the processor to improve energy efficiency. Due to the reduction of the execution frequency, the reliability of the task will be dramatically weakened. According to the idea of choosing tasks by ERT, in this algorithm, the tasks with the relatively high reliability and large CRR should be selected preferentially. Based on the above ideas, an algorithm for improving energy efficiency (IEE) is designed as Algorithm 4.

Algorithm 4 IEE

Require: results of current scheduling and $R_{\mathrm{req}}(A)$
Ensure: $R(A)$ and $E(A)$
1: $rcs \leftarrow$ reliability of current scheduling
2: **while** $R(A) > R_{\mathrm{req}}(A)$ **do**
3: choose the task t_i with the relatively high reliability and large CRR
4: calculate $R_{\mathrm{new}}(t_i)$ after reducing the one level execution frequency
5: $rcs \leftarrow \frac{rcs}{R(t_i)} \times R_{\mathrm{new}}(t_i)$
6: **if** $rcs > R_{\mathrm{req}}(A)$ **then**
7: reduce the execution frequency of task t_i by one level
8: $R(A) \leftarrow rcs$
9: **end if**
10: **end while**
11: calculate $E(A)$ using Eq. (9)

The IEE algorithm is mainly a while loop. When the current reliability of the application is higher than the reliability requirement, IEE repeatedly chooses the task with relatively high reliability and large CRR to reduce their execution frequency. Lines 4-5 calculate the reliability of the application after reducing the execution frequency of task t_i. Line 7 indicates that if the reliability of the application still satisfies the requirement, IEE will reduce the execution frequency of task t_i by one level. Line 8 updates the reliability of current scheduling.

The time complexity of the IEE algorithm is analyzed as follows. Fisrt, IEE chooses a suitable task, which has time complexity O(n). Then, IEE reduces the execution frequency of task and calculates the reliability, which has time complexity O(1). There are n tasks and fs level execution frequency, so the worst time complexity of IEE is O($n^2 \times fs$).



(a) The energy consumptions (b) The reliability values (c) The scheduling lengths

Fig. 2 GE applications for different total number of tasks (Experiment 1).

5. Experimental Performance Evaluation

5.1 Experimental Parameters

The C++ program is used to implement a scheduling simulator. The experimental parameters are mainly taken from [6], [7], [18] and listed as follows:

(1) The WCET of task t_i assigned to the processor pn_j is 10 ms $\leq w_{i,j} \leq$ 100 ms, the communication time between t_i and t_j is 10 ms $\leq c_{i,j} \leq$ 100 ms.

(2) For processor pn_j, $P_{j,\text{sta}} = 0.01$, $0.03 \leq P_{j,\text{ind}} \leq 0.07$, $0.8 \leq C_{j,\text{swi}} \leq 1.2$, and $2.5 \leq a_j \leq 3.0$. The maximum frequency is normalized to $f_{j,\max} = 1.0$ GHz and the lowest frequency is calculated by Eq. (3), the difference between adjacent frequency levels is 0.1 GHz.

(3) The communication energy consumption rate is $ecr = 0.2$ W.

(4) The transient faults rate of processor pn_j executes with maximum frequency is $0.000001 \leq \lambda_{j,\max} \leq 0.000009$, and the sensitivity faults rate of voltage/frequency scaling is $1.0 \leq d_j \leq 3.0$.

(5) Similar to [6], a simulated heterogeneous platform with 32 processors is constructed to execute applications.

Considering that ESRG [18] reduces energy consumption while satisfying the reliability requirement, the state-of-the-art RR [17] and MRCRG [6] consume fewer resources than MaxRe [16]. However, the RR and MRCRG do not reduce the execution frequency to improve energy efficiency. For that matters, we include DVFS techniques as further extension according to the ESRG algorithm framework, and these two extended algorithms are called as RR-D and MRCRG-D, respectively. Then, the proposed two-phase (TP) algorithm will be compared with the RR, RR-D, MRCRG, MRCRG-D, and ESRG algorithms.

Gaussian elimination (GE) and Fourier transform (FT) are widely used as DAG-based parallel applications for algorithm evaluation [6], [15], [18], [29], because GE and FT have the characteristics of low and high parallelism respectively. Therefore, these two applications are used to evaluate the algorithms, and their introduction is as follows.

GE application: A nonnegative integer s is used to describe the size of the GE application, the total number of tasks in application is $n = \frac{s^2+s-2}{2}$.

FT application: A nonnegative integer s is used to describe the size of the FT application, the total number of tasks in application is $n = (2 \times s - 1) + s \times \log_2^s$ with $s = 2^y$, where y is a nonnegative integer. The DAG structures of GE and FT applications can be found in Refs. [6], [15], [18], and [29].

5.2 Different Task Numbers

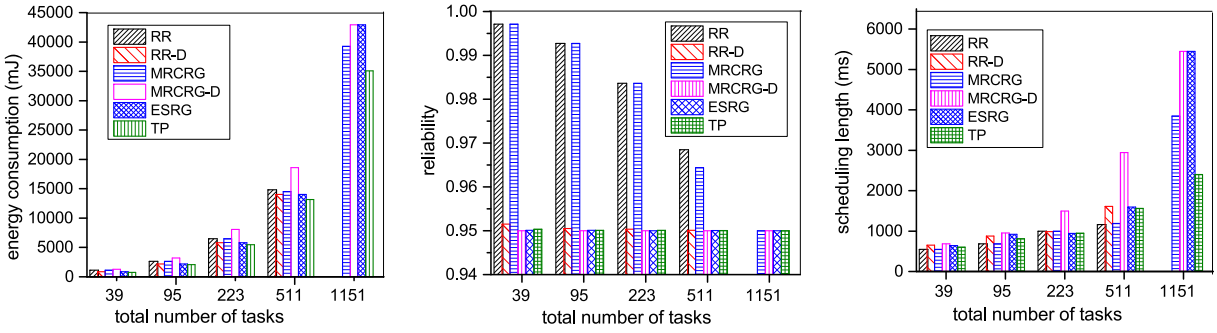
Experiment 1: This experiment uses GE application with different total number of tasks to compare the algorithms. The reliability requirement of the application is 0.95. Scale parameter s increases from 16 to 40 with increments of 4 (i.e., the total number of tasks is 135, 209, 299, 405, 527, 665, and 819 respectively). The energy consumption, obtained reliability, and the scheduling length with different algorithms are shown in Fig. 2.

Figure 2 (a) shows the energy consumptions of the different algorithms. As the total number of tasks increases, the energy consumptions of the six algorithms are increased. Overall, MRCRG-D generates the most energy consumption, and RR second. TP generates the least energy consumption. In details, the energy consumption generated by TP is 90.3% of RR, 94.1% of RR-D, 91.0% of MRCRG, 72.9% of MRCRG-D, and 94.6% of ESRG.

Figure 2 (b) shows the reliabilities of the different algorithms. The reliabilities obtained by RR-D, MRCRG-D, ESRG, and TP are almost the same, which are slightly higher than the reliability requirement. However, the reliabilities obtained by RR and MRCRG are much higher than the reliability requirement.

Figure 2 (c) shows the scheduling lengths of the different algorithms. The difference of scheduling length generated by RR, and MRCRG is not obvious, but the scheduling length generated by MRCRG-D is significantly larger than that of other algorithms. Whenever the scheduling length generated by TP is smaller than that generated by MRCRG-D and ESRG.

Experiment 2: This experiment uses FT application with different total number of tasks to compare the algorithms. The reliability requirement of the application is



(a) The energy consumptions (b) The reliability values (c) The scheduling lengths

Fig. 3 FT applications for different total number of tasks (Experiment 2).

0.95. Scale parameter s is 8, 16, 32, 64, and 128 respectively (i.e., the total number of tasks is 39, 95, 223, 511, and 1151 respectively). The energy consumption, obtained reliability, and the scheduling length of the algorithms are shown in Fig. 3.

As shown in Fig. 3 (a), as the total number of tasks increase, the energy consumption of the six algorithms are also increased. In details, the energy consumption generated by TP is 85.8% of RR, 93.8% of RR-D, 88.4% of MRCRG, 76.3% of MRCRG-D, and 85.9% of ESRG. Notice that when the total number of tasks is greater than or equal to 819, the reliability of the application can not be reached to 0.95 using RR and RR-D. In other words, when the number of tasks is equal to 1151, tasks will not be scheduled appropriately using RR and RR-D. Therefore, the schedule results at 1151 tasks using RR and RR-D are not plotted in the figure.

As shown in Fig. 3 (b), the reliabilities obtained by RR-D, MRCRG-D, ESRG, and TP are slightly higher than reliability requirement, and the reliabilities obtained by RR and MRCRG are much higher than reliability requirement. When the total number of tasks is greater than or equal to 1151, the reliability of the application can not be reached to 0.95 using RR and RR-D.

As shown in Fig. 3 (c), MRCRG-D generates the largest scheduling length, while other algorithms generate relatively small scheduling lengths when the total number of tasks is less than or equal to 511.

The result of Experiment 2 is similar to that of Experiment 1, The TP algorithm can always make the application satisfy the reliability requirement and generate the least energy consumption.

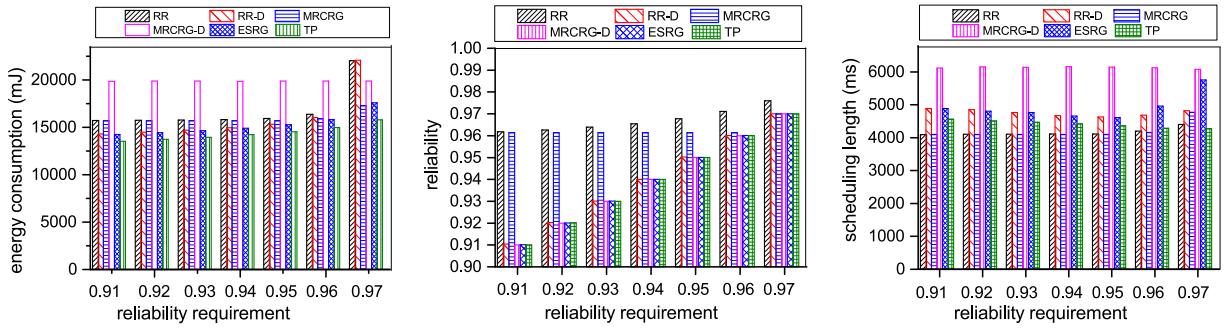
The main reasons for the above two experimental results can be explained as follows.

(1) RR and RR-D assume that the tasks that have not been assigned are with the same reliability requirement, ESRG also assumes the tasks that have not been assigned have the same reliability requirements $R_{\mathrm{ubrg}}(t_i) = \sqrt[n]{R_{\mathrm{req}}(A)}$. When the tasks are assigned to the processor, the actual reliability of tasks are little difference in ESRG and RR. However, the reliability requirements of all tasks in ESRG are more balanced than that in RR and RR-D. In RR and RR-D, the reliability of some tasks may be reduced too low (relative to ESRG), so the tasks that have not been assigned must be executed with higher reliability. As a result, sometimes, due to the task's reliability requirement is too higher, no processor can satisfy such requirement. Therefore, When the number of tasks becomes larger, RR and RR-D can not make the application to achieve the reliability requirement. In General, In RR, RR-D, and ESRG, the reliability requirement of tasks are not very different from each other. Hence, the application's energy consumption and the actual reliability generated by RR-D and ESRG is almost the same.

(2) When DVFS technique is used in the algorithms, The RR-D and MRCRG-D algorithm reduces the execution frequency as much as possible. In MRCRG-D, the reliability requirement of early assigned few tasks is too lower, thus more later-assigned tasks must be executed with the highest reliability, which takes more efforts to improve energy efficiency. Therefore, MRCRG-D generates more energy consumption than other algorithms.

(3) In TP, the task will be assigned to a processor with minimal energy consumption in the first phase using IMEC. When the reliability requirement of the application is higher than the reliability generated by IMEC, TP chooses the tasks that have relatively low reliability and great potential to enhance the reliability. Therefore, the tasks are assigned to the processor of higher energy efficiency more easily. When the reliability requirement of the application is lower than the reliability generated by IMEC, TP reduces the execution frequency of many tasks, other tasks are still assigned to processors with minimal energy consumption. Other algorithms first transform the reliability requirement of the application into the reliability requirement of each task, and then assign the tasks to the processor that satisfy the reliability requirements. However, some tasks may have relatively high reliability requirements, which are difficult to be allocated to high efficient processors resulting in significant energy overhead. Compared with these algorithms, the proposed TP consumes much less energy.

(4) When DVFS technique is used in the algorithms, in order to improve energy efficiency, the execution frequency of task is reduced as much as possible. As a result, the actual reliability of the assigned tasks will be slightly higher than the reliability requirement. In contrast, when the algorithm



(a) The energy consumptions (b) The reliability values (c) The scheduling lengths

Fig. 4 GE applications for different reliability requirements (Experiment 3).

without using DVFS technique, the actual reliability of the assigned tasks may be much higher than the reliability requirement. Therefore, the actual reliability generated by RR and MRCRG are much higher than reliability requirement, the actual reliability generated by RR-D, MRCRG-D, ESRG, and TP are slightly higher than reliability requirement.

(5) All algorithms do not consider scheduling length. In MRCRG-D, the tasks that are executed later must be executed with the highest reliability, which may lead some tasks that can be executed in parallel to be assigned to the same processor. MRCRG may do the same, but MRCRG does not reduce the execution frequency of task. Therefore, the scheduling length generated by MRCRG-D is longer than that of other algorithms. When the number of tasks is relatively large, the reliability requirement of each task is relatively high. In this case, TP first considers increasing the execution frequency of the tasks and generates the minimum scheduling length.

5.3 Different Reliability Requirements

Experiment 3: This experiment uses GE applications with different reliability requirements to compare the algorithms. Scale parameter $s = 32$ (i.e., the total number of tasks is 527). The reliability requirement of the application is changed from 0.91 to 0.97 with each increment of 0.01. The energy consumption, obtained reliability, and the scheduling length of the algorithms are shown in Fig. 4.

As shown in Fig. 4 (a), when the reliability requirement of the application is changed from 0.91 to 0.96, the change of energy consumption generated by all algorithms is small. When the reliability requirement of the application reaches 0.97, The energy consumption generated by all algorithms increases significantly. The main reasons can be explained as follows. When the reliability requirement of the application is lower than 0.96, due to the reliability requirement is not very high, all algorithms are more easily assign tasks to higher energy efficiency processor. On the contrary, when the reliability requirement of the application is higher than 0.96, the tasks are more difficult assign to higher energy efficiency processor. Overall, TP generates the least energy consumption. In details, the energy consumption generated by TP is 85.8% of RR, 90.1% of RR-D, 90.2% of MRCRG, 72.4% of MRCRG-D, and 94.2% of ESRG.

As shown in Fig. 4 (b), when the application is scheduled with RR-D, MRCRG-D, ESRG, and TP, the reliabilities of application obtained by these algorithms are about the same and slightly higher than reliability requirement. When the application is scheduled with RR and MRCRG, the reliability of the application is obviously higher than reliability requirement of the application.

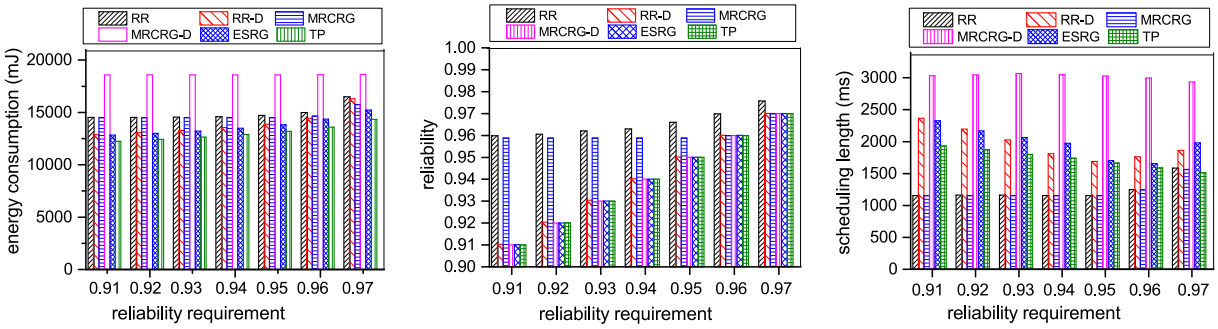
As shown in Fig. 4 (c), when the reliability requirement of the application is increased, the change of scheduling length generated by all algorithms is not significant, but the scheduling length generated by TP is always less than that generated by RR-D, MRCRG-D, and ESRG.

Experiment 4: This experiment uses FT applications with different reliability requirements to compare the algorithms. Scale parameter $s = 64$ (i.e., the total number of tasks is 511). The reliability requirement of the application is increased from 0.91 to 0.97 with each increment of 0.01. The energy consumption, obtained reliability, and the scheduling length of the algorithms are shown in Fig. 5.

As shown in Fig. 5 (a), when the reliability requirement is lower than 0.96, the energy consumption generated by all algorithms has little change. When the reliability requirement reaches 0.97, the energy consumption generated by all the algorithms increases significantly. Similar to the previous experimental results, the TP algorithm generates the least energy consumption.

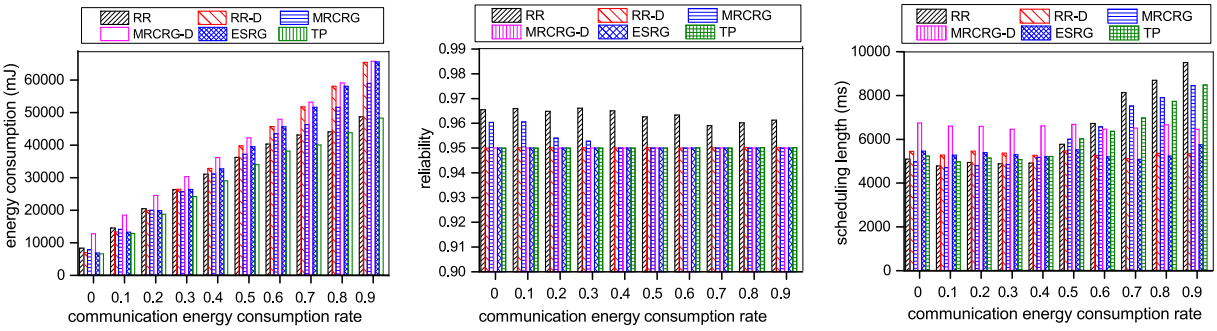
As shown in Fig. 5 (b), all algorithms can make the reliability of the application satisfy the requirements. RR-D, MRCRG-D, ESRG, and TP generate almost the same reliability.

As shown in Fig. 5 (c), whenever MRCRG-D generates the maximum scheduling length. with the reliability requirement of the application increases, the scheduling length generated by TP tends to decrease but is not obvious. When the reliability requirement increases from 0.91 to 0.95, the scheduling length generated by RR-D and ESRG also tends to decrease, but the scheduling length generated by RR and MRCRG almost remains unchanged. However, when the reliability requirement exceeds 0.95, the scheduling length generated by RR, RR-D MRCRG, and ESRG tends to increase.



(a) The energy consumptions (b) The reliability values (c) The scheduling lengths

Fig. 5 FT applications for different reliability requirements (Experiment 4).



(a) The energy consumptions (b) The reliability values (c) The scheduling lengths

Fig. 6 GE applications for different data transmission energy consumption rates (Experiment 5).

5.4 Different Data Transmission Energy Consumption Rates

Experiment 5: This experiment compares the algorithms for different data transmission energy consumption rates. GE applications with $s = 36$ (i.e., the total number of tasks is 665) are used in this experiment. The reliability requirement of the application is 0.95. The data transmission energy consumption rate *ecr* is increased from 0.1 to 0.9 with each increment of 0.1. The energy consumption, obtained reliability, and the scheduling length of the algorithms are shown in Fig. 6.

Figure 6 (a) shows the energy consumptions with different algorithms. According to the results, as *ecr* increases, the energy consumption generated by all algorithms also increases. It is worth noting that the differences in energy consumption for each algorithms will be greater. The main reasons are as follows. The same task may be assigned to different processors by different algorithms, which will lead to the difference in data transmission energy consumption becomes larger. Overall, the TP algorithm generates the lowest energy consumption than other algorithms.

As shown in Fig. 6 (b), the reliabilities obtained by RR-D, MRCRG-D, ESRG, and TP are slightly higher than reliability requirement, and the reliability obtained by RR is much higher than reliability requirement. The experimental results are similar to that of the previous four experiments.

As shown in Fig. 6 (c), when *ecr* increases from 0.1 to 0.5, the scheduling length of all algorithms has little changes, and the scheduling length of MRCRG-D is the longest. However, when *ecr* increases from 0.5 to 0.9, the scheduling length generated by RR, MRCRG and TP algorithms increases significantly. This happens because these three algorithms have more processors to choose when assigning tasks. When *ecr* is too high, these algorithms may choose processors with lower data transmission energy consumption to execute tasks, which will increase the scheduling length.

From the result of Experiment 1 to Experiment 5, it can be summarized as follows:

(1) Under different application scenarios, TP always generates the least energy consumption in all the algorithms.

(2) When the DVFS technique is used to reduce the execution frequency of some tasks, the total energy consumption may be increased. However TP can make good use of DVFS technique to improve energy efficiency.

(3) TP can generate relatively small scheduling length in most cases and the changes of application's reliability requirement have a little effect on the scheduling length.

6. Conclusions

This paper presents a two-phase algorithm framework to minimize energy consumption while satisfying reliability

requirement of the application. Experimental results demonstrate that the proposed algorithm consumes less energy than the state-of-the-art algorithms. We think that the proposed algorithm can be applied to the design phase of heterogeneous embedded systems. In future work, both scheduling length and reliability requirement will be considered. For instance, when the DVFS technique is used to improve energy efficiency, the values of slack time will be taken into account.

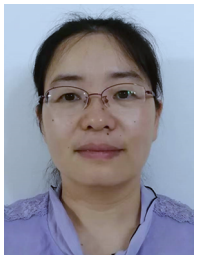
Acknowledgments

The research was partially funded by the National Natural Science Foundation of China (Grant Nos. 62062036).

References

[1] X. Zhu, C. He, K. Li, and X. Qin, "Adaptive energy-efficient scheduling for real-time tasks on dvs-enabled heterogeneous clusters," Journal of parallel and distributed computing, vol.72, no.6, pp.751–763, 2012.
[2] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," Parallel and Distributed Systems, IEEE Transactions on, vol.25, no.11, pp.2867–2876, 2014.
[3] K. Li, "Power and performance management for parallel computations in clouds and data centers," Journal of Computer and System Sciences, vol.82, no.2, pp.174–190, 2016.
[4] G. Xie, G. Zeng, X. Xiao, R. Li, and K. Li, "Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems," IEEE Trans. Parallel Distrib. Syst., vol.28, no.12, pp.3426–3442, Dec. 2017.
[5] D. Zhu and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," IEEE Trans. Comput., vol.58, no.10, pp.1382–1397, 2009.
[6] 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.
[7] D. Zhu, "Reliability-aware dynamic energy management in dependable embedded real-time systems," ACM Transactions on Embedded Computing Systems (TECS), vol.10, no.2, pp.1–27, 2010.
[8] B. Zhao, H. Aydin, and D. Zhu, "Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints," ACM Transactions on Design Automation of Electronic Systems, vol.18, no.2, pp.1–21, 2013.
[9] M. Lin, Y. Pan, L.T. Yang, M. Guo, and N. Zheng, "Scheduling co-design for reliability and energy in cyber-physical systems," IEEE Trans. Emerg. Topics Comput., vol.1, no.2, pp.353–365, 2013.
[10] M. Fan, Q. Han, and X. Yang, "Energy minimization for on-line real-time scheduling with reliability awareness," Journal of Systems and Software, vol.127, pp.168–176, 2017.
[11] H. Xu, R. Li, L. Zeng, K. Li, and C. Pan, "Energy-efficient scheduling with reliability guarantee in embedded real-time systems," Sustainable Computing: Informatics and Systems, vol.18, pp.137–148, 2018.
[12] P. Derler, E.A. Lee, and A.S. Vincentelli, "Modeling cyber–physical systems," Proc. IEEE, vol.100, no.1, pp.13–28, 2012.
[13] I. Lee, O. Sokolsky, S. Chen, J. Hatcliff, E. Jee, B. Kim, A. King, M. Mullen-Fortino, S. Park, A. Roederer, and K.K. Venkatasubramanian, "Challenges and research directions in medical cyber–physical systems," Proc. IEEE, vol.100, no.1, pp.75–90, 2012.
[14] K. Li, "Energy-efficient task scheduling on multiple heterogeneous computers: Algorithms, analysis, and performance evaluation," IEEE Transactions on Sustainable Computing, vol.1, no.1, pp.7–19, 2016.
[15] G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems," IEEE Trans. Ind. Informat., vol.13, no.3, pp.1068–1078, June 2017.
[16] L. Zhao, Y. Ren, Y. Xiang, and K. Sakurai, "Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems," 2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC), pp.434–441, IEEE, 2010.
[17] L. Zhao, Y. Ren, and K. Sakurai, "Reliable workflow scheduling with less resource redundancy," parallel computing, vol.39, no.10, pp.567–585, 2013.
[18] 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 Transactions on Sustainable Computing, vol.3, no.3, pp.167–181, 2018.
[19] J. Mei, K. Li, X. Zhou, and K. Li, "Fault-tolerant dynamic rescheduling for heterogeneous computing systems," Journal of Grid Computing, vol.13, no.4, pp.507–525, 2015.
[20] S. Wang, K. Li, J. Mei, G. Xiao, and K. Li, "A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems," Journal of Grid Computing, vol.15, no.1, pp.23–39, 2017.
[21] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," Information Sciences, vol.379, pp.241–256, 2017.
[22] H. Xu, R. Li, C. Pan, and K. Li, "Minimizing energy consumption with reliability goal on heterogeneous embedded systems," Journal of Parallel and Distributed Computing, vol.127, pp.44–57, 2019.
[23] N. Kumar, J. Mayank, and A. Mondal, "Reliability aware energy optimized scheduling of non-preemptive periodic real-time tasks on heterogeneous multiprocessor system," IEEE Trans. Parallel Distrib. Syst., vol.31, no.4, pp.871–885, 2019.
[24] J. Liu, Z. Zhu, and C. Deng, "A novel and adaptive transient fault-tolerant algorithm considering timing constraint on heterogeneous systems," IEEE Access, vol.8, pp.103047–103061, 2020.
[25] J. Peng, K. Li, J. Chen, and K. Li, "Reliability/performance-aware scheduling for parallel applications with energy constraints on heterogeneous computing systems," IEEE Transactions on Sustainable Computing, vol.7, no.3, pp.681–695, 2022.
[26] S. Safari, M. Ansari, H. Khdr, P. Gohari-Nazari, S. Yari-Karin, A. Yeganeh-Khaksar, S. Hessabi, A. Ejlali, and J. Henkel, "A survey of fault-tolerance techniques for embedded systems from the perspective of power, energy, and thermal issues," IEEE Access, vol.10, pp.12229–12251, 2022.
[27] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel Distrib. Syst., vol.13, no.3, pp.260–274, 2002.
[28] Z. Tang, L. Qi, Z. Cheng, K. Li, S.U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment," Journal of Grid Computing, vol.14, no.1, pp.55–74, 2016.
[29] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," Information Sciences, vol.319, pp.113–131, 2015.
[30] T. Mladenov, S. Nooshabadi, and K. Kim, "Implementation and evaluation of raptor codes on embedded systems," IEEE Trans. Comput., vol.60, no.12, pp.1678–1691, 2011.

Hongzhi Xu received the Ph.D. degree in computer science and engineering from Hunan University, Changsha, China, in 2018. He is a professor at Jishou University, Zhangjiajie, China. His research interests include heterogeneous computing systems and energy-efficient computing.



Binlian Zhang received the M.S. degree in computer science and engineering from Hunan Normal University, Changsha, China, in 2007. She is an associate professor at Jishou University, Zhangjiajie, China. Her research interests include heterogeneous computing systems and energy-efficient computing.