2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Energy-Efficient Scheduling Algorithms with Reliability Goal on Heterogeneous Embedded Systems

1st Yu Han
Wuhan University of Science and Technology
Wuhan, China
hanyu_1025@163.com

2nd Wei Hu*
Wuhan University of Science and Technology
Wuhan, China
huwei@wust.edu.cn

3rd Jing Liu
Wuhan University of Science and Technology
Wuhan, China
liujing@wust.edu.cn

4th Yu Gan
Wuhan University of Science and Technology
Wuhan, China
rorchach2k@gmail.com

***Abstract*—In this paper, we are trying to find a algorithm for scheduling DAG (Directed Acyclic Graph) tasks in heterogeneous embedded systems to minimize energy consumption while meeting the reliability requirement. Like many traditional algorithms, we divide the task scheduling algorithm into two phases, the task priority calculation phase and the task allocation phase. In the task priority calculation phase, we proposes a priority calculation algorithm IOD based on the difference in task's input and output data. In the task allocation stage, we proposes a task allocation algorithm based on fault-tolerant technology of task replication and DVFS technology. Combining the two phase, we get three scheduling algorithms, IODS, IODQ and IODR. In the experimental part, we compare the performance of the algorithm proposed in this paper with existing research algorithms (EFSRG algorithm and HRRM algorithm). The analysis of experimental results shows that the IODS algorithm is a better choice.**

***Index Terms*—reliability, heterogeneous, energy, scheduling algorithms**

I. Introduction

A. Background

As an important part of embedded system design, task scheduling algorithm has been studied and applied widely. In recent years of research, as a measure of the performance of scheduling algorithms, reliability and energy consumption have proved to be related issues. The task scheduling problem on heterogeneous platforms with high reliability and low energy consumption proved to be an NP-hard problem [1]. Qiu et al. had proposed a novel heuristic algorithm to solve this problem for embedded computer systems [2] in 2009. The goal of most heuristic algorithms is to find a feasible solution rather than an optimal solution. Early scheduling algorithms set the shortest scheduling length as the goal, with the development of hardware and the complication of computing tasks, research on energy saving and reliability has gradually emerged [3] [4].

This paper was supported by the Key Project of Scientific Research Plan of Hubei Provincial Department of Education (Grant No. D20201102)

After the DVFS (Dynamic Voltage and Frequency Scaling) [8] technology was proposed, many scheduling algorithms proposed energy-saving optimization based on DVFS technology by reducing the operating frequency of the processor, it is indeed possible to reduce the energy consumption of task execution [5]. In those scheduling algorithms that focus on reliability, the importance of tasks can be quantified as reliability requirements according to their character [6]. Existing studies have proposed many algorithms to meet task's reliability, such as task replication [9], primary-backup, checkpoint and other fault-tolerant means [7] [11] [12]. However, it has been proven in the literature that reducing the execution frequency of the processor may lead to an increase in the instantaneous failure rate of the processor, then resulting in task execution failure. According to the relationship between task scheduling reliability and energy consumption, finding a high-reliability and low-energy scheduling algorithms is essential for the development of embedded heterogeneous multi-core systems [13] [14].

B. Motivation

Considering the scheduling length, energy consumption or reliability separately, many excellent scheduling algorithms can be found in the existing research [10]. But it is impractical that the scheduling algorithm that considers the scheduling length, energy consumption and reliability performance alone does not consider the connection of the three [15]. Random algorithms has superior performance, but the algorithm complexity is too high and the calculation cost is uncontrollable.

C. Contribution

In this paper, we study the scheduling problem of DAG graph task sets in heterogeneous embedded systems, comprehensively consider energy consumption and reliability, and aim

978-0-7381-2646-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00082

to find a heuristic scheduling algorithm that reduces energy consumption while meeting the reliability constraints of tasks.

1. We explained the model assumptions used in the scheduling algorithm, and described the scheduling problem to be solved in this paper in detail.
2. In the task priority calculation stage, we propose a task priority calculation algorithm IOD based on the difference between the input and output of the task data.
3. In the task allocation stage, we proposed three traversal rules based on the fault-tolerant technology of task replication and DVFS technology.
4. Designed an experiment to compare the algorithm proposed in this paper with the EFSRG algorithm [16] and the HRRM algorithm [17], and analyze the experimental results.

Outline: The structure of this paper is as follows. Section 1 is the introduction of this paper, which introduces the research background and research motivation. Section 2 introduces related research work. Section 3 introduces related models and problem statement. Section 4 and Section 5 describe the details of the algorithm implementation, experiments, and the analysis of experimental results. Section 6 is the conclusion.

II. Related Work

The scheduling research of embedded systems is classified into two categories according to the processor system, single-core scheduling and multi-core scheduling. Early research mainly discussed the single-core scheduling [18] [19] [20]. With the development of multi-core systems, the research on multi-core scheduling has been paid more and more attention [21] [22] [23]. Multi-core systems are divided into homogeneous and heterogeneous [24]. With the advent of the big data era, scheduling algorithms that reduce energy consumption and improve reliability have become research hot-spots. Qiu et al. had done many important works in this area [25] [26] [27].

Among the existing scheduling algorithms, some papers [28] [29] [30] focus on improving the reliability, while some papers [31] [32] [33] focus on reducing the energy consumption.

On reliability, the goal is to reduce the impact of transient and permanent failures of the processor system. There are many factors that cause processor system failures, including high temperature, hardware failure, etc. [34] [35]. Since permanent faults account for a low proportion of all faults during task execution, most studies on reliability scheduling mainly consider transient faults [36].

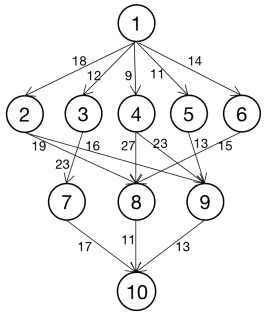
Regarding the discussion of energy consumption, the purpose is to reduce the energy consumption of application. As we all know, many embedded systems base on DVFS technology to achieve the purpose of energy saving, so most studies on energy-saving scheduling also adopt this technology [7].

III. System Model and Problem Formulation

A. Task Model

An application could be represented by a DAG (Directed Acyclic Graph), and each DAG is represented as $G = \{N, E\}$, N represents a set of nodes in G, and each node $\tau_i \in N$ represents a task, there is $T = \{\tau_1, \tau_2, ..., \tau_n\}$, n is the number of task. E represents the directed edge set of the DAG, which is the communication cost between tasks. $e_{i,j} \in E$ is the computation cost between task τ_i and task τ_j, there is $E = \{e_{i,j} | i, j \leq n, i \neq j\}$. The execution cost of the task node on each processor at the highest frequency is represented by W, $w_{i,k} \in W$ represents the τ_i execute on the processor u_k at the maximum frequency f_k^{max}(About the processor core model we will introduce in the next subsection), there is $W = \{w_{i,k} | i \leq n, k \leq m\}$.

We have an example of DAG task set as follow, show in Figure 1, the reliability goal is $R_{req} = 0.95$.



(a) an example of DAG task set with 10 tasks

	u_1	u_2	u_3
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

(b) the execution cost of the task node on each processor at the highest frequency

Fig. 1. An example DAG task

B. Processor Model

The processor model is composed of a set of heterogeneous fully connected processor, where each processor can independently adjust voltage and frequency based on DVFS technology, the processors set could be represented by $U = \{u_1, u_2, ..., u_m\}$, m is the number of processor. Each processor u_k can executed with a discrete set of frequency $F_k = \{f_k^{min}, f_k^{max}, f_k^{low}\}$, f_k^{min}, f_k^{max} is the lowest frequency and the highest frequency that u_k can reach in terms of hardware performance, f_k^{low} is the actual frequency lower limit, the value depends on the actual task set data and processor system parameters.

C. Reliability Model

The execution of task may fail due to hardware failure, high temperature and other uncertain factors. This paper establish a reliability model based on transient failure probability refers to the existing reliability scheduling research. The model uses the exponential function based on execution time and fault rate to express mission reliability.

When the task is executed, the fault rate can be expressed by λ, and the fault rate is only related to the hardware parameter. The fault rate when the processor is executed at the frequency f can be expressed as,

$$\lambda(k,f) = \lambda_k \times 10^{\frac{d_k(1-f)}{1-f_k^{min}}} \tag{1}$$

Among them, λ_k is the fault rate executed at the frequency f_k^{max} of the processor u_k, d_k is a static parameter related to the hardware, reflecting the sensitivity of the probability of failure to frequency changes, and f_k^{min} is the minimum frequency of u_k.

Further, the reliability of the task replica can be expressed by r, and the reliability of task τ_i executed at frequency $f_{i,k}$ on the processor u_k is can be expressed as:

$$r(i,k,f_{i,k}) = e^{-\lambda(k,f_{i,k})\times w_{i,k}\times\frac{f_k^{max}}{f_{i,k}}} \tag{2}$$

In particular, when $f_{i,k} = 0$, it means that the task has no replica on this processor, $r(i,k,f_{i,k}) = 0$.

The reliability of task τ_i is calculated by the reliability of each replica of the task as:

$$r_{actual}(i) = 1 - \prod_{k=1}^{m}(1 - r(i,k,f_{i,k})) \tag{3}$$

So the reliability of the task set could represent as:

$$R_{total} = \prod_{i=1}^{n} r_{actual}(i) \tag{4}$$

Our target is to let

$$R_{total} \geq R_{req} \tag{5}$$

Since the reliability R_{total} given by the task set represents the reliability of the entire task set, and each task is dynamically allocated processor and operating frequencies during the scheduling process, R_{total} needs to be decomposed into task reliability requirements $r_{req}(i)$ as:

$$\begin{aligned} & r_{req}(i) \\ &= \frac{R_{req}}{\prod_{j\in allocated(i)} r_{actual}(j) \times \prod_{j\in unallocated(i)} r_{req}(j)} \\ &= \frac{R_{req}}{\prod_{j\in allocated(i)} r_{actual}(j) \times \prod_{j\in unallocated(i)} \sqrt[n]{R_{req}}} \end{aligned} \tag{6}$$

$allocated(i)$ represents the set of tasks that have been allocated before task τ_i in the priority queue, and $unallocated(i)$ represents the set of tasks that have not been allocated after task τ_i in the priority queue.

Then, if each task τ_i in the task set satisfies $r_{actual}(i) \geq r_{req}(i)$, then the inequality (5) can be satisfied, that is, the reliability requirement of the task set can be satisfied.

D. Power Model

The power consumption of the processor is mainly composed of frequency-related dynamic consumption, frequency-independent dynamic consumption and static consumption. Among them, the frequency-related dynamic power consumption is the main component, and the total power of the processor is represented by P, and there is

$$\begin{aligned} P(k,f) &= P_{s,k} + g(P_{ind,k} + P_{d,k}) \\ &= P_{s,k} + g(P_{ind,k} + c_k \times f^{\alpha(k)}) \end{aligned} \tag{7}$$

$P_{s,k}$ means frequency-independent static power, $P_{ind,k}$ means frequency-independent dynamic power, $P_{d,k}$ means frequency-dependent dynamic power, g means system state, $g = 0$ when system is in sleep, $g = 1$ when system is running, and c_k means the switching capacitance of processor, its a hardware parameter, α_k represents the dynamic power exponent.

Further, the dynamic energy consumption of the replica of task τ_i executed on the processor u_k at frequency $f_{i,k}$ can be expressed as:

$$E_d(i,k,f_{i,k}) = (P_{ind,k} + c_k \times f^{\alpha_k}) \times w_{i,k} \times \frac{f_k^{max}}{f_{i,k}} \tag{8}$$

The scheduling dynamic energy consumption of task τ_i is calculated by the dynamic energy consumption of each replica of the task.

$$E_{d\_actual}(i) = \sum_{k=1}^{m} E_d(i,k,f_{i,k}) \tag{9}$$

The static energy consumption of the task set is related to the $SL(schedule \quad length)$ of the task.

$$E_s = SL \times P_s \tag{10}$$

The total energy consumption of the task set can be expressed as:

$$E_{total} = \sum_{i=1}^{n} E_{d\_actual}(i) + E_s \tag{11};$$

In particular, when $f_{i,k} = 0$, it means that the task has no replica on this processor, $E_d(i,k,f_{i,k}) = 0$.

E. Problem Formulation

1) Problem Description: According to the task model, processor model, reliability model and energy consumption model, the problem model is obtained as follows, given task set $G = \{N, E\}$, processor set $U = \{u_1, u_2, ..., u_m\}$, task set reliability requirement R_{req}, our purpose is to obtain a scheduling plan according to the scheduling algorithm, to make the task set meets the reliability requirements, and the total energy consumption for execution is as low as possible.

$$minE_{total}$$

$$R_{total} - R_{req} \geq 0$$

The scheduling scheme is expressed as $plan_{G,U} = \{priority, frequency\}$, $priority$ represents the task node priority queue, and $frequency$ represents the task node execution frequency $n \times m$ matrix. According to the equation(7), for a certain processor, $P_{ind,k}, c_k, \alpha_k, P_{s,k}$ are constants, then

$$\begin{aligned} & \tfrac{\partial E(i,k,f)}{\partial f} = \\ & \frac{f_{max}}{f} w_{i,k}(c_k \times \alpha_k \times f^{\alpha_k - 1} - \frac{1}{f}(P_{ind,k} + c_k \times f^{\alpha_k})) \end{aligned}$$

Let the above formula be equal to 0, then,

$$f_k^{ee} = \sqrt[\alpha_k]{\frac{P_{ind,k}}{(\alpha_k - 1)c_k}} \tag{12}$$

It can be concluded that when the frequency is lower than f_k^{ee} in the processor u_k, the energy consumption increases as the frequency decreases. Therefore, for u_k,

$$f_k^{low} = max\{f_k^{ee}, f_k^{min}\} \tag{13}$$

According to the above problem description and model assumptions, our scheduling algorithm needs to complete three things:

- Determine the scheduling priority of task nodes;
- According to the scheduling priority of the task node, the execution frequency of the task replica on each processor is determined under the requirement of reliability. If the frequency is 0, it means that the task has no replica on this processor;
- Output the scheduling plan and calculate the related performance of the scheduling plan.

2) Scheduling Rules: The scheduling algorithm proposed in this paper is a scheduling algorithm based on task replication that satisfies reliability constraints, so the scheduling process needs to follow the following rules [16]:

- Task cannot be preempted during execution;
- For each task, there is at most one replica on each processor;
- When all replica of the task is successfully executed, its child nodes can start to receive the data of the task.

3) Schedulable Judgment: According to (2), (3), (4), it can be concluded that the highest reliability value that can be achieved for a given task set on a heterogeneous multi-core processor system with given parameters for task-based replication scheduling is R_{total_max}

$$R_{total\_max} = \prod_{i=1}^{n}(1 - \prod_{k=1}^{m}(1 - r(i,k,f_k^{max}))) \tag{14}$$

That is, each task has a replica that is executed at the highest frequency on all processors.

When $R_{total\_max} > R_{req}$, it means that the scheduling of the task set in the processor system can meet the reliability goal, otherwise, the reliability goal can't be met.

IV. Algorithm Implementation

A. Task's Priority Calculation

1) IOD (Input and Output Data) Algorithm Description: The purpose of this stage is to obtain the priority queue of task scheduling in the DAG. The tasks enter the next stage for processing according to the priority queue, and the replica of each task has the same priority as the task. The scheduling priority calculation of classic algorithms such as HEFT[13] generally takes the maximum time required by the task to the exit node as the rank value, so as to determine the task priority according to the rank value. In reliability-energy scheduling, the execution frequency of tasks on each processor is determined dynamically, so the execution time of tasks on each processor is also uncertain. The priority calculation method of classic algorithms is not suitable for the scheduling model of this paper. This paper proposes a priority calculation method IOD based on the difference between input and output data. The maximum input data of task determines the earliest execution time of the task, and the maximum output data of task determines the earliest execution time of the child task. We want the task with the largest difference between the maximum output data and the maximum input data to be executed first. This can effectively avoid the waiting of other tasks. First, the task set represented by the DAG diagram is layered and scheduled by layer. Within each layer, calculate the maximum output data of the task, the maximum input data, and use the difference between the two as the rank value. The algorithm is as follows,

Algorithm 1 IOD

Input: $G = \{N, E\}, W$
Output: $priority[]$
for each task τ_i **do**
  $level(\tau_i) = \max_{\tau_j \in succ(\tau_i)}\{level(\tau_j)\} + 1;$
  $rank(\tau_i) = max\{data_{output} - data_{input}\};$
end for
$now \leftarrow 1;$
for each $level$ **do**
  sort all tasks in this $level$ by $rank(\tau_i)$;
end for
for each $level$ **do**
  for each task τ_i in this $level$ **do**
    $priority[\tau_i] \leftarrow now;$
    $now++;$
  end for
end for
return $priority[\ ]$.

For example of task set shown in Figure 1, we can use the IOD algorithm to get the priority of the task as Table 1. Then the order of task scheduling in the case is $\tau_1, \tau_4, \tau_3, \tau_5, \tau_2, \tau_6, \tau_7, \tau_9, \tau_8, \tau_{10}$.

TABLE I
Priority Allocate

task	level	max input	max output	priority
τ_1	1	0	18	1
τ_2	2	18	19	5
τ_3	2	12	23	3
τ_4	2	9	27	2
τ_5	2	11	13	4
τ_6	2	14	15	6
τ_7	3	23	17	7
τ_8	3	27	11	9
τ_9	3	23	13	8
τ_{10}	4	17	0	10

2) Time complexity of IOD Algorithm: The time complexity of calculating the level and the out_data of each node is $O(n+e)$, where e is the total number of data connections, that is, the

total number of elements in the set E. Subsequently, assigning the priority of each node is equivalent to sorting n nodes, The time complexity is $O(n^2)$, so the total time complexity is $O(n^2)$.

B. Task's Allocation

The purpose of this stage is to determine the execution frequency of task replicas on each processor. If the execution frequency of a certain task on a certain processor is 0, it means that the task does not have a replica on that processor. The goal of determining the execution frequency of the task on each processor is to perform the energy consumption as low as possible under the constraint of meeting the reliability requirements. Generally, the task is traversed through each frequency selection of each processor to obtain a feasible solution. Adopt different traversal rules, and get different feasible solutions.

1) Traversal Rules: We propose the following three traversal rules:

- R: based on the purpose of first meeting reliability requirements, propose the traversal frequency selection in ascending order based on task reliability.
- S: based on the relationship between reliability and energy consumption, it is proposed to traverse the frequency in ascending order according to the ratio s of energy consumption and reliability, and $S = \frac{E}{R}$.
- Q: based on the interrelationship between reliability and energy consumption, referring to the [37], it is proposed to traverse the frequency according, and $Q = -\frac{E}{\log_{10}(1-R)}$.

2) Algorithm Description: The algorithm of the task allocation stage is divided into three steps, as shown below:

1 Take out the taskτ_i at the head of the queue from the priority queue, and calculate the reliability requirement $r_{req}(\tau_i)$ of the task by equation(6).

2 Follow the Traversal Rules to traverse the frequency selection of the task on each processor until the $r_{req}(\tau_i)$ is met. If the highest frequency traversed to each processor still cannot meet the task reliability requirements, it will jump out of the loop and report the scheduling failure.

3 Output the processing frequency of the task on each processor, and delete the task node from the priority queue. If there is no task in the priority queue, end the task allocation phase, otherwise return to step 1.

According to the above rules, the algorithm of the task allocation stage can be expressed as algorithm 2, where the X could be R, S or Q according to different traversal rules.

3) Example: The various parameters of the processor are given as Table 2, and the tasks in Figure 1 can be scheduled for demonstration according to the following parameters.

For the task set in Figure 1, taking task τ_1 as an example of IOD algorithm and choosing traversal rules S (It can be called IODS algorithm), the algorithm has the following process, calculate $r_{req}(1) = \sqrt[10]{0.95} \approx 0.994884$, the traversal process of frequency is as follows Table 3.

Algorithm 2 Task's Allocation

Input: $G = \{N, E\}, W, U, R_{req}$
Output: $frequency[\ ][\ ]$

```
for each task τ_i (by the priority base on rank) do
    r_actual(i) ← 0;
    f_k^temp ← f_k^low;
    calculate r_req(i);
    while r_actual(i) ≤ r_req(i) do
        for each processor u_k do
            find f_min^temp with min{X};
            frequency[i][min] ← f_min^temp;
            f_min^temp ← f_min^temp + 0.01;
            calculate r_actual(i);
        end for
    end while
end for
return frequency[ ][ ].
```

TABLE II
PROCESSOR'S PARAMETERS

Processor parameters								
	$P_{ind,k}$	$P_{s,k}$	d_k	α_k	c_k	λ_k	f^{low}	f^{max}
u_1	0.03	0.005	2.0	2.9	0.8	0.0002	0.2583	1
u_2	0.03	0.005	2.0	2.9	0.9	0.00013	0.2480	1
u_3	0.03	0.005	2.0	3.0	1.0	0.0005	0.2466	1

First calculate the value of f_k^{ee}, $f_1^{temp} = f_1^{ee} = 0.2583$, $f_2^{temp} = f_2^{ee} = 0.2480$, $f_3^{temp} = f_3^{ee} = 0.2466$, and then use f_k^{temp} calculate the S of task τ_1 on each processor, we found that the value of S_3 is smallest (IODR is to find biggest R, IODQ is to find smallest Q), so let $f_{1,3} = 0.2466$ and let $f_3^{temp} = 0.2566$, continue the calculation in the previous step until the reliability requirements are met.

As result, $f_{1,1} = 0.4283$, $f_{1,2} = 0$, $f_{1,3} = 0.5666$, $r_{actual}(1) = 0.9951$ and $E_{actual}(1) = 6.5832$.

According to the above steps, using IOD algorithm and choosing the traversal rules as S (It can be called IODS algorithm) we can get the result of all tasks shown in Table 4.

Similarly, we can also get the results of other algorithms as shown in the Table 5. From the table, the energy consumption of IODS algorithm is lowest, and the scheduling length of HRRM algorithm is shortest. But this is just the result of an example, for evaluating the performance of all algorithms, we need to design complex experiments.

4) Time complexity IODX Algorithm:

- The time complexity of schedulability is $O(n \times m)$;
- The time complexity of selecting the execution frequency of the task on each processor is $O(n \times m \times l)$, l is the max number of frequency chooses of each processor;

Therefore, the total time complexity is $O(n \times m \times l)$.

V. EXPERIMENT

In the previous part of this paper, we proposed an algorithm IOD for calculating task priority and three traversal rules, which can be combined to obtain three heuristic scheduling

TABLE III
FREQUENCY CHOOSING

	u_1	u_2	u_3	r_{actual}	r_{req}
f_k^{temp}	0.2583	0.2480	0.2466		
s	3.4514	3.8598	**2.9508**	0.5565	0.994884
$f_{i,k}$	0	0	0.2466		
f_k^{temp}	0.2583	0.2480	0.2566		
s	3.4514	3.8598	**2.8165**	0.5840	0.994884
$f_{i,k}$	0	0	0.2566		
		... ...			
f_k^{temp}	0.2583	0.2480	0.5466		
s	3.4514	3.8598	3.4017	0.9357	0.994884
$f_{i,k}$	0	0	0.5466		
f_k^{temp}	0.2583	0.2480	0.5566		
s	**3.4514**	3.8598	3.4839	0.9819	0.994884
$f_{i,k}$	0.2583	0	0.5466		
		... ...			
f_k^{temp}	0.4283	0.2480	0.5666		
s	**3.5234**	3.8598	3.5686	0.9948	0.994884
$f_{i,k}$	0.4283	0	0.5566		
f_k^{temp}	0.4383	0.2480	0.5666		
s	3.5869	3.8598	**3.5686**	0.9951	0.994884
$f_{i,k}$	0.4283	0	0.5666		

TABLE IV
IODS ALGORITHM'S SCHEDULING RESULT

IODS						
task	u_1	u_2	u_3	r_{req}	r_{actual}	E_{actual}
τ_1	0.4283	0.0000	0.5666	0.994884	0.995066	6.58320
τ_4	0.2683	0.5980	0.0000	0.994702	0.997298	5.41979
τ_3	0.4783	0.2780	0.0000	0.992294	0.992298	5.28806
τ_5	0.4483	0.3780	0.0000	0.994880	0.995049	5.76829
τ_2	0.6183	0.2780	0.0000	0.994714	0.994730	8.35525
τ_6	0.4483	0.0000	0.5366	0.994868	0.995149	6.22966
τ_7	0.7583	0.2580	0.0000	0.994603	0.998848	6.36078
τ_9	0.2683	0.6180	0.0000	0.990656	0.995302	8.10623
τ_8	0.6183	0.0000	0.0000	0.990240	0.990665	1.84721
τ_{10}	0.0000	0.6980	0.0000	0.994457	0.994777	3.48296
$R_{total} = 0.9503, E_{total} = 57.44 + 2.92 = 60.36, SLR = 286.3476$						

algorithms IODS, IODR and IODQ. All three algorithms can solve the scheduling problem proposed in this paper. In order to find out which algorithm has the better performance, this section tests the performance of the algorithm through simulation experiments. And in order to make the experimental results more objective, we introduced the EFSRG algorithm and HRRM algorithm as the reference group in the experiment.

A. Comparative experiment based on FFT and GE

- FFT, Fast Fourier Transform is an efficient algorithm for calculating discrete Fourier Transform in a computer. The calculation of N is as follows, ρ is used as the size parameter of the Fast Fourier Transform application.

$$N = (2 \times 2^{\rho} - 1) + 2^{\rho} \times log(2^{\rho})$$

$$= (2 + \rho) \times 2^{\rho} - 1, \rho \in Z^{+}$$

- GE Gaussian Elimination method is mainly used to solve linear equations, it can also find the rank of matrix and the inverse of matrix. It is an important algorithm in linear algebraic programming. The calculation of N is as follows, ρ is used as the matrix size parameter of the Gaussian Elimination application.

$$N = (\rho^2 + \rho - 2)/2, \rho \in Z^{+}$$

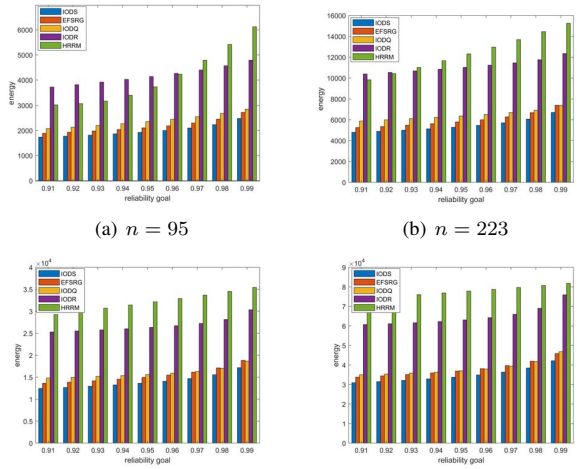
TABLE V
ALGORITHMS' SCHEDULING RESULT

$Algorithm$	E_{actual}	r_{actual}	SL
IODS	57.44+2.92=60.36	0.9503	286.35
IODQ	63.29+1.31=64.60	0.9502	141.62
IODR	68.51+1.14=69.65	0.9502	124.00
EFSRG	61.06+3.75=64.81	0.9503	316.22
HRRM	106.68+1.14=107.82	0.9842	123.00

The parameter settings of processor and application set refer to the existing literature as follows: $10ms \leq w_{i,k} \leq 100ms$, $10ms \leq e_{i,k} \leq 100ms$, $0.03 \leq P_{ind,k} \leq 0.07$, $1 \leq d_k \leq 3$, $0.8 \leq c_k \leq 1.2$, $2.5 \leq \alpha_k \leq 3.0$, $f_k^{max} = 1$, $0.00001 \leq \lambda_k \leq 0.0001$. We simulated a heterogeneous system with 8 processors. The frequency of each processor can be dynamically adjusted and the precision is 0.01.

Experiment 1 Compare the scheduling results of IODS, IODR, IODQ, EFSRG and HRRM algorithms for the FFT application DAG task set when ρ is equal to 4, 5, 6, 7 (the number of nodes N is 95, 223, 511, 1151), the scheduling results are reflected in the energy consumption with the given task set reliability goal (reliability requirements from 0.90 to 0.99, step size 0.01). For each node's number, randomly generate 100 FFT task sets, and compare the average energy consumption of various algorithms for scheduling task sets with various reliability goal, and the result is shown in the figure 2.

Experiment 2 Compare the scheduling results of IODS, IODR, IODQ, EFSRG and HRRM algorithms for the GE application DAG task set when ρ is equal to 13, 21, 32, 48 (corresponding to the number of nodes N is 90, 230, 527, 1175), the scheduling results are reflected in the energy consumption with the given task set reliability goal (reliability requirements from 0.90 to 0.99,step size 0.01). For each node's number, randomly generate 100 GE task sets, and compare the average energy consumption of various algorithms for scheduling task sets with various reliability goal, and the result is shown in the figure 3.

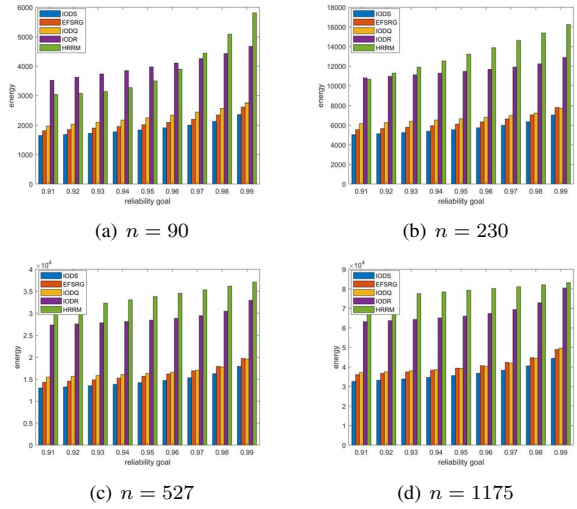


(a) $n = 95$ (b) $n = 223$

(c) $n = 511$ (d) $n = 1151$

Fig. 2. energy consumption of FFT task sets under different reliability goal

Analyzing the results of the experiment 1 and experiment 2 shows that: to schedule GE or FFT task sets with different numbers of nodes under the constraints of satisfying reliability requirements, the scheduling energy consumption of the IODS scheduling algorithm is smaller.



(a) $n = 90$ (b) $n = 230$

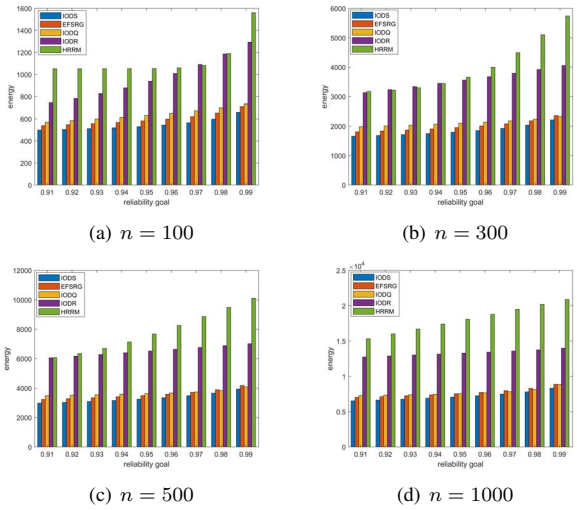
(c) $n = 527$ (d) $n = 1175$

Fig. 3. energy consumption of GE task sets under different reliability goal

B. Comparative experiment based on random tasks

The random generation task set refers to the parameter setting method used in the paper of Topcuolu HR [21] [38], number of tasks in the graph $n = 100, 300, 500, 1000$, range percentage of computation cost on processor $h = 0.75$, shape parameter of the graph $\epsilon = 2$, communication to computation ratio $CCR = 5$.

Experiment 3 Compare the scheduling results of IODS, IODR, IODQ, EFSRG and HRRM algorithms for the random DAG task set when n is equal to 100, 300, 500, 1000. Same with FFT and GE, for each node's number, randomly generate 100 random task sets, and compare the average energy consumption of various algorithms for scheduling task sets with various reliability goal, and the result is shown in the figure 4.

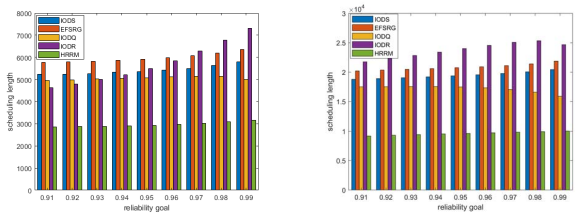


(a) $n = 100$ (b) $n = 300$

(c) $n = 500$ (d) $n = 1000$

Fig. 4. energy consumption of random task sets under different reliability goal

Experiment 4 Compare the scheduling results of the IODS, IODR, IODQ, EFSRG and HRRM algorithms for random DAG task sets with n being 300, 1000, given the reliability requirement of the task set is 0.95, compare the scheduling length under different number of nodes. For each n, 100 task sets are randomly generated, then calculated the average scheduling length of the 100 task sets. The results are shown in the figure 5.



(a) $n = 300$ (b) $n = 1000$

Fig. 5. scheduling length of different reliability goal

Analyzing the results of the experiment 3 and experiment 4 shows that: scheduling randomly generated task sets with different numbers of nodes under the constraints of satisfying reliability requirements. The scheduling energy consumption of the IODS scheduling algorithm is smaller. The scheduling length of the HRRM scheduling algorithm is smaller, but its energy consumption cost is high. In general, both IODS and IODQ are good choices. IODS has better energy-saving effects, and the scheduling length of IODQ is shorter. According

to the requirements of the problem formulation of this paper, so IODS is the best choice.

VI. Conclusions

In this paper, we focus on the scheduling of parallel application in heterogeneous embedded systems, we proposed an energy-saving scheduling algorithm that satisfies the reliability goal of task sets. The algorithm is based on task replication technology and DVFS technology to adjust the reliability and energy consumption of tasks, both considered energy consumption and reliability.

Our work is divided into two parts. In the task priority calculation phase, we proposes a priority calculation algorithm IOD based on the difference in task data input and output. In the task allocation stage, we proposes a task allocation algorithm based on fault-tolerant technology of task replication and DVFS technology. Combining the two phase, we get three scheduling algorithms IODS, IODQ and IODR.

Through the comparative experiments of EFSRG algorithm, HRRM algorithm, IODS algorithm, IODQ algorithm and IODR algorithm we find that the IODS is a better choice to solve the DAG task set scheduling problem with reliability requirements in heterogeneous embedded systems.

References

[1] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman (1983).

[2] M. Qiu and E. Sha, Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems, ACM Transactions on Design Automation of Electronic Systems (TODAES), 14(2), 1-30, 2009

[3] M. Qiu, J.W. Niu, L.T. Yang, X. Qin, S. Zhang, B. Wang, Energy-aware loop parallelism maximization for multi-core DSP architectures, IEEE/ACM Int'l Conf. on Green Computing and Comm., 2010

[4] M. Qiu and J. Li, Real-time embedded systems: optimization, synthesis, and networking, CRC Press

[5] M. Weiser, et al., Scheduling for Reduced CPU Energy. Springer, 1994.

[6] Ernst, D. , et al., Razor: circuit-level correction of timing errors for low-power operation, IEEE Micro 24.6(2004):10-20.

[7] Kumar, N., J. Mayank , and A. Mondal, Reliability Aware Energy Optimized Scheduling of Non-Preemptive Periodic Real-Time Tasks on Heterogeneous Multiprocessor System, IEEE Transactions on Parallel and Distributed Systems 31.4(2020):871-885.

[8] M. Qiu, C. Xue, Z. Shao, E.H.-M. Sha, Energy minimization with soft real-time and DVS for uniprocessor and multiprocessor embedded systems, ACM/IEEE Design, Automation & Test in Europe Conference (DATE), pp. 1-6, 2007

[9] M. Qiu, J. Liu, J. Li, Z. Fei, Z. Ming, E.H.-M. Sha, A novel energy-aware fault tolerance mechanism for wireless sensor networks, IEEE/ACM Int'l Conf. on Green Computing and Communications, 2011

[10] M. Qiu, H. Li, E.H.-M. Sha, Heterogeneous real-time embedded software optimization considering hardware platform, ACM symposium on Applied Computing, pp. 1637-1641, 2009

[11] Y. Guo, et al. Exploiting primary/backup mechanism for energy efficiency in dependable real-time systems, Journal of Systems Architecture 78(2017):68-80..

[12] Salehi, M., et al., Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems, IEEE Transactions on Very Large Scale Integration Systems, 24.7(2016):2426-2437.

[13] M. Qiu, K. Zhang, M. Huang, Usability in mobile interface browsing, Web Intelligence and Agent Systems: An International Journal, 4(1), pp. 43-59, 2006

[14] J. Niu, Y. Gao, M. Qiu, Z. Ming, Selecting proper wireless network interfaces for user experience enhancement with guaranteed probability, Journal of Parallel and Distributed Computing, 72(12), pp. 1565-1575, 2012

[15] Fan, X. , C. S. Ellis , and A. R. Lebeck, The Synergy Between Power-Aware Memory Systems and Processor Voltage Scaling, Int'l Workshop on Power-Aware Computer Systems, Springer, 2003.

[16] Xie, G. , et al. Energy-efficient Fault-tolerant Scheduling of Reliable Parallel Applications on Heterogeneous Distributed Embedded Systems. IEEE Transactions on Sustainable Computing, (2017):1-1.

[17] Xie, Guoqi , et al. Minimizing Redundancy to Satisfy Reliability Requirement for a Parallel Application on Heterogeneous Service-oriented Systems. IEEE Transactions on Services Computing, (2017):1-1.

[18] Liu, C. L. , and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. Journal of the ACM, 1973.

[19] Burchard, A. , et al. A linear-time online task assignment scheme for multiprocessor systems. IEEE Workshop on Real-Time Operating Systems and Software, 2002.

[20] M. Shreedhar and G. Varghese, Efficient fair queuing using deficit round-robin, in IEEE/ACM Transactions on Networking, vol. 4, no. 3, pp. 375-385, 1996.

[21] Topcuoglu, H. , S. Hariri , and M. Y. Wu . Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems (2002).

[22] Arabnejad, H. , and J. G. Barbosa . List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table, IEEE Transactions on Parallel & Distributed Systems 25.3(2014):682-694.

[23] Bittencourt, L. F. , R. Sakellariou , and E. Madeira . DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm. IEEE 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010.

[24] T. Mitra, Heterogeneous Multi-core Architectures. IPSJ Transactions on System LSI Design Methodology 8(2015):51-62.

[25] M. Qiu, Z. Chen, M. Liu, Low-power low-latency data allocation for hybrid scratch-pad memory, IEEE Embedded Systems Letters, 6(4), 69-72, 2014

[26] Y. Guo, Q. Zhuge, J. Hu, M. Qiu, E.H.-M. Sha, Optimal data allocation for scratch-pad memory on embedded multi-core systems, IEEE Int'l Conf. on Parallel Processing (ICPP), pp.464-471, 2011

[27] L. Zhang, M. Qiu, W.C. Tseng, E.H.-M. Sha, Variable partitioning and scheduling for MPSoC with virtually shared scratch pad memory, Journal of Signal Processing Systems, Vol.58(2), pp. 247-265, 2010

[28] V. Izosimov, P. Pop, P. Eles and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant distributed embedded systems," Design, Automation and Test in Europe, 2005, pp. 864-869.

[29] Srinivasan, S. and N. K. Jha . Safety and reliability driven task allocation in distributed systems, IEEE Transactions on Parallel & Distributed Systems, 10.3(1999):238-251.

[30] Kartik, S. and R. M. C. Siva, Task allocation algorithms for maximizing reliability of distributed computing systems, IEEE Transactions on Computers 46.6(2002):719-724.

[31] Lin, M., and S. M. Ng, Energy Aware Scheduling for Heterogeneous Real-Time Embedded Systems Using Genetics Algorithms, 2005.

[32] Xie G, Gang Z, Li R, et al., Energy-Aware Processor Merging Algorithms for Deadline Constrained Parallel Applications in Heterogeneous Cloud Computing. IEEE Transactions on Sustainable Computing, 2017, 2(2):62-75.

[33] J. Niu, C. Liu, Y. Gao, M. Qiu, Energy efficient task assignment with guaranteed probability satisfying timing constraints for embedded systems, IEEE TPDS, 25(8), 2043-2052, 2013

[34] Iyer, R. K., D. J. Rossetti, and M. C. Hsueh. Measurement and modeling of computer reliability as affected by system activity. ACM Transactions on Computer Systems 4.3(1986):214-237.

[35] Castillo, et al. Derivation and Calibration of a Transient Error Reliability Model. Computers, IEEE Transactions on C-31.7(1982):658-671.

[36] Shatz, S. M. and J. P. Wang . Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. IEEE Transactions on Reliability, 38.1(2002):16-27.

[37] Han, L., et al. Energy-aware strategies for reliability-oriented real-time task allocation on heterogeneous platforms. ICPP '20.

[38] Canon, Louis Claude, M. E. Sayah, and Héam, Pierre-Cyrille. A Markov Chain Monte Carlo Approach to Cost Matrix Generation for Scheduling Performance Evaluation, Int'l Conf. on HPCS, 2018, pp. 460-467.