Microprocessors and Microsystems 52 (2017) 312–324

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Reliability-aware low energy scheduling in real time systems with shared resources

Yi-wen Zhang*, Hui-zhen Zhang, Cheng Wang

College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China

ARTICLE INFO

Article history:
Received 30 October 2016
Revised 7 April 2017
Accepted 30 June 2017
Available online 4 July 2017

Keywords:
Real-time systems
Dynamic voltage scaling
Real-time scheduling
Shared resources

ABSTRACT

Dynamic voltage scaling (DVS) is a technique which is widely used to save energy in a real time system. Recent research shows that it has a negative impact on the system reliability. In this paper, we consider the problem of the system reliability and focus on a periodic task set that the task instance shares resources. Firstly, we present a static low power scheduling algorithm for periodic tasks with shared resources called SLPSR which ignores the system reliability. Secondly, we prove that the problem of the reliability-aware low power scheduling for periodic tasks with shared resources is NP-hard and present two heuristic algorithms called SPF and LPF respectively. Finally, we present a dynamic low power scheduling algorithm for periodic tasks with shared resources called DLPSR to reclaim the dynamic slack time to save energy while preserving the system reliability. Experimental results show that the presented algorithm can reduce the energy consumption while improving the system reliability.

1. Introduction

Energy management is an important in an embedded real-time system, especially for PDA and cell phones. Low energy consumption can reduce the cost of cooling and extend the lifetime of the battery. Dynamic voltage scaling (DVS) which uses the slack time to adjust the processor speed is a simple and effective technique to decrease the energy consumption [1].

There are many researches that focus on the energy management while all tasks will meet their deadlines in different real-time task models [2–6]. But these researches only take the energy consumption into consideration and ignore the system reliability. In order to improve the system reliability, many researchers [7–9] have studied the method of minimizing energy consumption while improving the system reliability. These researches use the checkpoint technique to improve the system reliability and ignore that the processor speed has effected on the system reliability. Zhu et al [10] show that the DVS technique has a negative impact on the system reliability i.e. it can increase transient fault rates at low supply voltage and frequency. As a result, the application of the DVS technique must be carefully, especially for the mission-critical real-time embedded applications. For this reason, a number of recent research articles focus on the system reliability while taking the energy consumption into consideration and promote the so-called reliability-aware power management (RA-PM) framework [11–16]. Zhu and Aydin [11] have first proposed RA-PM framework that minimizing the energy consumption while preserving the system reliability. Furthermore, Zhu [12] has extended the work in [11] and proposed schemes that can use the dynamic slack time to dynamically schedule an additional recovery task to recuperate the reliability loss while considering the energy consumption. In addition, Zhu and Aydin [13] have proved that a static RA-PM framework for periodic real time tasks is NP-hard and proposed two task-level utilization-based heuristics. Moreover, Zhao et al. [15] have proposed a more efficient shared recovery technique called SHR. The SHR can achieve more energy savings while preserving the system reliability.

The above researches assume that there are no dependencies between the tasks composing the considered task set. Sha et al. [17] have studied the priority inversion problem and proposed two priority inheritance protocols i.e. the basic priority inheritance protocol and the priority ceiling protocols. Jeffay [18] has proposed a new scheduling scheme called EDF/DDM to solve the problem of scheduling the sporadic tasks with shared resources. These researches don't take the energy consumption into consideration. Moreover, Horng et al. [19] have proposed a new technique that extends the work in [18] and considers the energy consumption while meeting task's deadlines. But this technique ignores the static power and assumes that each task instance with its worst case execution time. Furthermore, Zhang and Guo [5] have extended the work in [19] and proposed a more efficient algorithm, called DSTSASR. It can reclaim the dynamic slack time to save energy and take the general power model into consideration. In

* Corresponding author.
E-mail address: zyw@hqu.edu.cn (Y.-w. Zhang).

http://dx.doi.org/10.1016/j.micpro.2017.06.020
0141-9331/

addition, it combines the DVS with dynamic power management (DPM).

In this paper, we focus on the reliability-aware energy management for a periodic real time task set with shared resources. Firstly, we present a static low power scheduling algorithm for periodic tasks with shared resources called SLPSR which ignores the system reliability. Secondly, we prove that the problem of the reliability-aware low power scheduling for periodic tasks with shared resources is NP-hard and present two heuristic algorithms called SPF and LPF respectively. Finally, we present a dynamic low power scheduling algorithm for periodic tasks with shared resources called DLPSR to reclaim the dynamic slack time to save energy while preserving system reliability.

The remainder of the paper is organized as follows: the model and problem descriptions are presented in Section 2. A static solution for reliability-aware low power scheduling is presented in Section 3. In Section 4, we present a dynamic solution for reliability-aware low power scheduling. The simulation results and discussions are presented in Section 5 and conclusions are presented in Section 6.

2. Models and problem description

2.1. Processor and task model

In this paper, we assume that the DVS-enabled processor can provide a continuous speed, that is, the processor can be operated from the minimum processor speed $S_{\min}$ to the maximum processor speed $S_{\max}$. For simplicity, the speeds are normalized with respect to $S_{\max}$, i.e. $S_{\max} = 1$. This assumption is reasonable, because the discrete speed levels can be solved by the method of two adjacent speeds [2] or the next available higher speed. In addition, we ignore the speed transition overheads.

We focus on the real time system in an uni-processor that consists of the periodic task set T and a serially reusable resource set R. The periodic task set T has n periodic tasks, represented as $T = \{T_1, T_2, \cdots, T_n\}$ and the reusable resource set R has m reusable resources, represented as $R = \{R_1, R_2, \cdots, R_m\}$. According to [18], the resource is a software object, e.g. a data structure. It must be accessed in a mutually exclusive manner. A periodic task T_i can be described by (e_i, r_i, p_i), where e_i is the worst case execution time (WCET) of T_i under the maximum processor $S_{\max}$. r_i is the resource requirement of the task T_i and it can be represented by an integer ($1 \le r_i \le m$). If $r_i \neq 0$, the task T_i has a resource requirement and other tasks which access to the resource R_{r_i} will be blocked. If $r_i = 0$, the T_i is a task without resource requirement. p_i is the period of T_i. In addition, we assume that the relative deadline of T_i is equal to its period and that the task should access at most one resource at a time. We arrange the period of T_i in the non-decreasing order, i.e. $p_1 \le p_2 \le \cdots \le p_n$. Let $T_{i,j}$ be the j^{th} instance of the task T_i and $rt_{i,j}$ be the release time of the task T_i. We denote u_i as the utilization of the task T_i and it can be expressed by $u_i = e_i/p_i$. The system utilization U_{tot} can be expressed by $U_{tot} = \sum_{i=1}^{n} u_i$ and we assume that $U_{tot} < 1$. Furthermore, we assume that the WCET of T_i scales linearly with the processor speed, i.e. the WCET of T_i is equal to $e_i \cdot S_{\max}/S_i$ with the processor speed S_i.

The periodic task set T is scheduled by the EDF/DDM policy [18]. The EDF/DDM policy is based on the earliest deadline first (EDF) policy. Note that when the deadline is the same, the early released task has a higher priority and that both deadlines and the released time are the same, the lower index task has a higher priority. There are two kinds of deadlines for each task instance $T_{i,j}$ in the EDF/DDM policy. One is the initial deadline ($ID_{i,j}$), the other is the execution deadline ($ED_{i,j}$) which is determined by the time that $T_{i,j}$ begins to execute. Let P_i be the shortest period of the task that needs a resource R_i. It can be expressed by $P_i = \min_{1 \le j \le n} \{p_j | r_j = i\}$. In addition, let $t_{i,j}$ be the commences execution time of the task instance $T_{i,j}$. The initial deadline is equal to the execution deadline for the task without resource requirements. As for the task instance $T_{i,j}$ with resource requirement, the initial deadline $ID_{i,j}$ is equal to $rt_{i,j} + p_i$ and the execution deadline $ED_{i,j}$ is equal to $\min\{rt_{i,j} + p_i, \lfloor t_{i,j} \rfloor + 1 + P_i\}$ [15].

2.2. Energy model

In this paper, we adopt a simple and general system-level power model. It is first proposed in [10] and then used in [3–5]. Therefore, the power P is given by:

$$P = P_s + h(P_{ind} + P_{dep}) = P_s + h(P_{ind} + C_{ef} S^m) \quad (1)$$

Here, P_s is static power which maintains basic circuits and keeps the clock running. It can be treated as 0 by turning off the system. P_{ind} is a speed-independent power and can be treated as 0 when the processor is in idle status. P_{dep} is a speed-dependent power and can be expressed by $P_{dep} = C_{ef} \cdot S^m$. Here, C_{ef} stands for effective switching capacitance, m stands for system dependent constants ($2 \le m \le 3$) and S stands for the running speed of the processor. In addition, h stands for a constant coefficient. If the processor is in idle status, $h = 0$; else, $h = 1$.

For energy efficiency, the critical speed S_{crit} is proposed in [5,11], which is a minimum energy-efficient speed and it can be expressed by $S_{crit} = \sqrt[m]{\frac{P_{ind}}{C_{ef} \cdot (m-1)}}$. Therefore, all tasks will execute with the speed larger or equal to S_{crit}.

2.3. Fault and reliability models

There are both permanent and transient faults that occur during the task execution in the embedded real time systems. According to [11,12,20,21], the transient faults occur much more frequently than permanent faults, especially with the continued scaling of CMOS technologies and reduced design margins. Therefore, we focus on the transient faults and use the re-execution task or time-redundancy method to eliminate the effect of transient faults. In addition, we assume that transient faults can be detected by consistency checks or sanity [22] at the end of task's execution and assume that the overhead of the detection can be incorporated into WCET of the task.

We adopt the fault arrival rate model in [11,13,14]. It can be expressed as following:

$$\lambda(S) = \lambda_0 g(S) \quad (2)$$

Where λ_0 is the average fault rate under the maximum processor speed $S_{\max}$, $g(S_{\max}) = 1$and S is the running speed of the processor. Furthermore, we use the same exponential fault rate model as given in [11], $g(S)$ can be expressed as following:

$$g(S) = 10^{\frac{d(1-S)}{1-S_{\min}}} \quad (3)$$

Where d is a system constant ($d > 0$). From this model, we find that the lower processor speed, the higher fault arrival rate.

The reliability of a task is defined as the probability of the task which completes its execution successfully. According to [12,14,15], the reliability of a task T_i with its WCET under the running speed S_i can be expressed as following:

$$R_i(S_i) = e^{-\lambda(S_i) * \frac{e_i}{S_i}} \quad (4)$$

Where $\lambda(S_i)$ is given in the Eq. (2). Moreover, the original reliability of a task T_i (R_i^0) is defined as the task executes with the maximum processor speed $S_{\max}$ and it can be expressed by $R_i^0 = R_i(S_{\max}) = e^{-\lambda_0 \cdot e_i}$. As a result, the system original reliability of the periodic task set T is given by $\prod_i^n R_i^0$.