Microprocessors and Microsystems 45 (2016) 164–175

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Low power fixed priority scheduling sporadic task with shared resources in hard real time systems

Yi-wen Zhang[a,*], Chu-gui Xu[b]

[a] College of Computer Science and Technology, Huaqiao University, XiaMen, China, 361021
[b] School of Information Engineering, Sanming University, Sanming, China 365004

ARTICLE INFO

Article history:
Received 21 January 2016
Accepted 29 April 2016
Available online 3 May 2016

Keywords:
Sporadic task
Resource-sharing
Energy management
Fixed priority scheme

ABSTRACT

Dynamic voltage scaling (DVS) and dynamic power management (DPM) are two effective techniques in a real time system. In this paper, we address the problem of the canonical sporadic task scheduling based on a fixed-priority scheduling scheme and take a generalized power model into account. The sporadic tasks share a set serially reusable, single-unit resources. First, we present a rate monotonic with dual priority scheduling policy, called RM/DPP, to solve the sporadic tasks shared resources scheduling problem and discuss the feasibility of the RM/DPP algorithm. Second, a static fixed-priority sporadic tasks scheduling algorithm with shared resources, called SFPSASR, has been put forward, which considers the off-chip workload and assumes that each task executes with its worst case execution time. Third, for energy efficiency, a dynamic fixed-priority sporadic tasks scheduling algorithm with shared resources, called DFPSASR, has been put forward, which considers the speed transition overhead and combines the DVS and the DPM technology. The experimental results show that the proposed SFPSASR algorithm can reduce the energy consumption by 42.14%~51.73% over the RM/DPP algorithm and the DFPSASR algorithm can reduce the energy consumption by 79.37%~82.94% over the SFPSASR algorithm.

1. Introduction

Low energy consumption has become one of the major goals in an embedded real time system design. This is especially important for battery-operated systems, such as mobile phones and portable medical devices. The reason for this is that the low energy consumption can extend the lifetime of the battery and decrease the heat dissipation of the devices. The major energy consumption of the processor comes from the dynamic power due to switching activity and the static power consumption due to the leakage current [1]. Dynamic power management (DPM) [2] and dynamic voltage scaling (DVS) [3–10] have been widely used to reduce the energy consumption. DPM switches the processor to a sleep mode or turns off the system component that is not currently in using. DVS aims to reduce the dynamic power by scaling down the processor frequency when the processor is not fully loaded.

Existing studies on a real-time system for energy management focus on the periodic task model or the mixed task model [3–9]. In this paper, we investigate the canonical sporadic tasks that share a set serially reusable, single-unit resources and take a generalized power model into account. The resource is a software object, for example, a data structure. The tasks share this resource that must be accessed in a mutually exclusive manner. The main contribution of this paper can be summarized as follows:

(1) A new rate monotonic with dual priority scheduling policy, called, RM/DPP, is put forward. The RM/DPP algorithm which borrows some ideas about preemption threshold from [11,12] is based on rate-monotonic (RM) scheduling policy and it can ensure that the resource can be accessed in a mutually exclusive manner. In addition, we discuss the feasibility of the RM/DPP algorithm.
(2) We present a static fixed-priority sporadic tasks scheduling algorithm with shared resources, called SFPSASR, which considers the off-chip workload and assumes that each task executes with its worst case execution time. The SFPSASR algorithm combines the static sporadic tasks low power scheduling algorithm (SSTLPSA) presented in [13] with the RM/DPP algorithm. In addition, we discuss the feasibility of the SFPSASR algorithm.
(3) We present a dynamic fixed-priority sporadic tasks scheduling algorithm with shared resources, called DFPSASR, which considers the speed transition overhead and combines the DVS and the DPM technology. In addition, the DFPSASR algorithm can reclaim the slack time from the early completion task.

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

http://dx.doi.org/10.1016/j.micpro.2016.04.010
0141-9331/

The remainder of the paper is organized as follows. Section 2 reviews the literature about the lower power scheduling algorithm. The preliminaries are introduced in Section 3. Section 4 presents a new rate monotonic with dual priority scheduling policy, called, RM/DPP. Section 5 presents a static scheduling algorithm, called SFPSASR. We present a dynamic algorithm, called DFPSASR, in Section 6. The simulation experiments are presented in Section 7 and Section 8 concludes our work.

2. Related work

Many researchers have explored DVS in real-time systems [3–9]. The above researches focus on the periodic task model or mixed task model. First, we focus on algorithms which consider the canonical sporadic task model. The works focus on the sporadic task model presented in [13–16]. Qadi et al. [14] have proposed a DVS algorithm, called DVSST, which is based on the earliest deadline first (EDF) scheduling policy and only considers the dynamic power. Zhong and Xu [15] have proposed an algorithm, called TV-DVS, which is more energy efficient than that of the DVSST algorithm [14], but the task will not meet the deadline constraints. Zhang and Guo [16] have proposed a more efficient algorithm, called DSTSA, which considers a generalized power model and reclaims the dynamic slack to reduce the energy consumption while all tasks will meet their deadlines. In addition, Zhang and Guo [13] have used a RM scheme to solve the canonical sporadic task low power scheduling problem and proposed a DVS algorithm, called DSTLPSA, which takes a generalized power model into account and combines DVS with DPM.

Note that all of above researches assume that each task is independent, i.e. the task doesn't share resources. The resource shared problem in real time system has been studied in [11,17–19]. Sha et al. [17] have proposed two protocols, i.e. the basic priority inheritance protocol (PIP) and the priority ceiling protocol (PCP), to solve the shared resource problem which results in a priority inversion problem. Baker [18] has extended the PCP policy to dynamic priority schemes and proposed the stack resource policy (SRP). In addition, Jeffay [19] has proposed a new scheduling policy, called EDF/DDM, which fits to the dynamic priority scheduling algorithm. Moreover, Saksena and Wang [11] have proposed a more flexible scheme, called preemption threshold, to solve a priority inversion problem. We use the basic idea of preemption threshold in this paper.

Although, the above works focus on the shared resource problem, these works don't take the energy consumption into consideration. The following works have taken the shared resource problem and the energy consumption into consideration [20–23]. The algorithms presented in [20] and in [22] have focused on the periodic task model. Although, The algorithm presented in [21] has focused on the sporadic task model. It only takes the dynamic power into account and ignores the static power. Moreover, it assumes that the execution time of a task scales linearly with the processing speed and that each task executes with its worst case execution time. The algorithm presented in [23] is the most closely related to our work. But it is based on the EDF scheduling policy.

The algorithm presented in [15] considers the sporadic task model, but it ignores the resource share problem and the task will miss its deadline. The algorithm presented in [20] considers the resource share problem, but it focuses on the periodic task model and assumes that the linear relationship between the speed and the execution time of the task. In addition, it schedules the task by the EDF scheduling policy. The algorithm presented in [22] considers the resource share problem and uses the RM scheduling policy, but it focuses on the periodic task model and assumes that the linear relationship between the speed and the execution time of the task. To the best of our knowledge, this is a first work to consider the shared resource for sporadic task model which is based on the RM scheduling policy. In addition, it takes the speed transition overhead into consideration, and it considers the non-linear relationship between the speed and the execution time of the task. Most importantly, our algorithm can meet deadlines constraints.

3. Preliminaries

3.1. System model

We consider a sporadic task set in hard real time systems that shares a set of serially reusable, single-unit resources. This model comes from [19]. A set serially reusable, single-unit resource is a software object, e.g., a data structure. It is shared among a group of tasks, and it must be accessed in a mutually exclusive manner [19]. For example, within the context of a concurrent programming language in which shared data is encapsulated within a monitor [24], a resource would be an individual monitor. A sporadic task set and a reusable resource set are represented as $T = \{T_1, T_2, \ldots, T_n\}$ and $R = \{R_1, R_2, \cdots, R_m\}$, respectively. A sporadic task T_i can be described by a 3-tuple (e_i, r_i, p_i) [19,21], where e_i is the worst case execution time (WCET) of task T_i under the maximum processor speed. r_i is the resource requirement of the task T_i. It can be represented by an integer ($1 \le r_i \le m$). If $r_i \neq 0$, the task T_i needs a resource R_{r_i} and other task which needs the resource R_i will be pended. This can ensure that the resource can be mutually exclusive accessed. If $r_i = 0$, then the task T_i doesn't have resource requirements. p_i is the minimum time interval between the release of two consecutive instances of a task. The minimum time interval p_i is arranged in the non-decreasing order, i.e. $p_1 \le p_2 \le, \ldots, \le p_n$. We assume that the relative deadline of a task is equal to its minimum time interval and that each task can access at most one resource at a time. In addition, we ignore the scheduling overhead, such as context-switching overhead, task selection overhead in this paper.

Let P_j be the shortest minimum time interval task that uses resource R_j ($1 \le j \le m$) and it can be expressed $P_j = \min_{1 \le i \le n} (p_i | r_i = j)$. Let LR_i be the last release time of task T_i. We denote $T_{i,j}$ as j^{th} instance of task T_i and assume that a variable speed processor can be operated at a set of continuous speed $[S_{\min}, S_{\max}]$, where $S_{\min}$ and $S_{\max}$ are the minimum speed and the maximum speed of the processor, respectively. We normalize the speed with the $S_{\max}$, i.e. $S_{\max} = 1.0$. According to [25], there are time overheads resulting from the speed-switching. We denote TO_{ij} as the time overhead of the processor speed changes from S_i to S_j. TO_{ij} can be expressed [25] as follows

$$TO_{ij} = K \cdot \left| S_i - S_j \right| \tag{1}$$

Where K is a constant factor.

The worst case execution time can be divided into two parts [4], one is the frequency-dependent execution time (on-chip), the other is frequency-independent execution time (off-chip). The frequency-independent execution time contains the I/O device access latency and the main memory access time. The worst case execution time of task T_i with the speed of S_i can be expressed as follows:

$$e_i = \frac{x_i}{S_i} + y_i \tag{2}$$

where x_i is the frequency-dependent execution time, y_i is frequency-independent execution time. We denote ρ as the off-chip workload and it can be expressed by $\rho = \frac{y_i}{e_i}$. Let U_{tot} be the system utilization and it can be expressed by $U_{tot} = \sum_{i=1}^{n} \frac{e_i}{p_i}$.