Journal of Systems Architecture 83 (2018) 12–22

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Energy aware fixed priority scheduling for real time sporadic task with task synchronization

Yi-wen Zhang*, Cheng Wang, Jin Liu

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

ARTICLE INFO

Keywords:
Sporadic task
Fixed priority
Shared resource
Energy management

ABSTRACT

Energy management is one of the key issues in the real-time embedded systems. The slowdown method based on the dynamic voltage scaling (DVS) and the shutdown method based on the dynamic power management (DPM) can effectively reduce energy consumption. The previous algorithm which uses the static speed to deal with the blocking situation is too conservative. We present a static task synchronization for sporadic tasks scheduling (STSST) algorithm. The dynamic speed is computed to minimize energy consumption while satisfying the time constraints in the STSST algorithm. But, the STSST algorithm assumes that each task executes with its worst case execution time. As the actual execution time of the task is often lower than its worst case execution time. A dynamic task synchronization for sporadic task (DTSST) algorithm which can reclaim the slack time generated from the early completion task is proposed. It combines the DVS technique and the DPM technique to reduce energy consumption. Furthermore, the feasibility conditions are given and proved. The experimental results show that the DTSST algorithm can reduce energy consumption up to 20.57%～67.66% over the STSST algorithm and it consumes 43.79%～67.66% less energy than that of the DS algorithm.

1. Introduction

Energy consumption is one of the key issues in the real-time embedded systems, especially for the portable mobile systems operated with a limited battery capacity. The development speed of battery technique is much lower than the growth rate of system energy consumption [1]. The processor is one of the major sources of energy consumption [2]. There are two effective techniques to reduce the processor energy consumption. One is to slowdown the processor speed by the dynamic voltage scaling (DVS) technique [3]. The other is to shut down the processor by dynamic power management (DPM) technique [4]. The different slowdown speeds will result in different energy consumptions. Therefore, how to choose proper speeds is very important for minimizing energy consumption while satisfying the system timing constraint.

In past decades, power aware scheduling algorithms for real-time tasks in DVS-enabled system have been widely studied [3–9]. Pillai and Shin [6] have investigated the independent periodic task energy efficient scheduling problem and proposed a cycle-conserving EDF (CCEDF) algorithm based on the task actual utilization. But it ignores the processor static power. Chen and Kuo [7] have considered the general power model and proposed the procrastination determination algorithm to replace the greedy procrastination algorithm. The procrastination determination algorithm can effectively reduce the processor leakage current power consumption. For energy efficiency, Zhang and Guo [8] have proposed a novel algorithm which takes the general power model into consideration and can use the slack time generated from the completed higher priority task and lower priority task to reduce energy consumption. In addition, Niu and Li [9] have proposed a fixed priority scheduling policy low power algorithm which combines a critical strategy and shut down strategy to reduce energy consumption.

The above researches assume that there are no dependencies between the tasks composing the considered task set. In fact, the tasks are usually dependent due to shared resources. There are some protocols such as priority inheritance protocol (PIP) [10], priority ceiling protocol (PCP) [10] and stack resource policy (SRP) [11] to enforce mutually exclusive access to shared resources. The energy efficient scheduling problem for periodic tasks with non-preemptive sections has been studied in [12]. The dual speed (DS) algorithm has been proposed to solve this problem and it is based on the SRP protocol and the RM scheduling policy. The task begins to execute at static low speed S_L without considering blocking time. When the task is blocked by the lower priority task, the processor switches to the static high speed S_H which takes the blocking time into consideration. To achieve further energy savings, Jejurikar [5] has studied the problem energy efficient

* Corresponding author.
E-mail addresses: zyw@hqu.edu.cn (Y.-w. Zhang), wangcheng@hqu.edu.cn (C. Wang), geneleo@hqu.edu.cn (J. Liu).

https://doi.org/10.1016/j.sysarc.2017.11.004
Received 2 November 2016; Received in revised form 7 May 2017; Accepted 17 November 2017
Available online 21 November 2017

scheduling of non-preemptive tasks and proposed the stack based slowdown (SBS) algorithm, which builds upon the optimal feasibility test for non-preemptive systems and based on the EDF scheduling policy. The SBS algorithm can minimize the transitions to a higher speed by computing different slowdown factors based on the blocking task. Later, Lee et al [2] have studied the same problem and proposed a multi-speed algorithm that exploits various speed levels depending on specific blocking situation to minimize energy consumption. Note that, the above researches overestimate preemption times on non-preemptive tasks which result in the speed of the task higher than necessary. To further reduce energy consumption, Li et al [13] have proposed a novel individual speed algorithm (ISA) that computes one speed for each individual task in a non-preemptive task set without jeopardizing any task deadlines.

There are some works [14–16] discussing the task synchronization to enforce mutually exclusive access to shared resources. Jejurikar and Gupta [14] have proposed dual mode (DM) algorithm based on the RM scheduling policy. The DM algorithm computes static slowdown factors in the presence of task synchronization to minimize energy consumption of the system. In addition, it introduces the frequency inheritance which bounds the blocking time to guarantee the task deadlines. Furthermore, the same problem based on EDF scheduling policy is studied in [15]. The above task synchronization algorithm uses the maximum blocking time to compute the processor speed. To further reduce energy consumption, Wu [16] proposed blocking aware two-speed (BATS) algorithm based on the stack resource policy to synchronization the tasks with shared resources. The BATS algorithm borrows the idea of the DS algorithm and it is based on the EDF scheduling policy. The BATS algorithm has recomputed the high speed based on the actual blocking time to reduce energy consumption. In addition, the BATS algorithm first schedules tasks with the static low speed, the processor switches to the high speed when the block occurs.

Note that the above researches focus on the dependent periodic tasks with shared resources. Few researches focus on the dependent sporadic tasks with shared resources. Horng et al. [17] have studied the problem of scheduling dependent sporadic tasks with shared resources and proposed a novel algorithm, called DVSSR. But it ignores the static power of the processor and assumes that each task executes with its worst case execution time and that each task will require access to at most one resource at a time. Zhang and Guo [18] have extended this work and have proposed an energy efficient algorithm, which reclaims the slack time from the early completion task to reduce energy consumption and takes the general power model into account. But it also assumes that each task will require access to at most one resource at a time, and it is based on the EDF scheduling policy.

In this paper, we focus on the dependent sporadic tasks with shared resources model based on the fixed priority scheduling policy. We first present a static task synchronization for sporadic tasks scheduling (STSST) algorithm assuming that each task executes with its worst case execution time. The dynamic speed is computed to minimize energy consumption while satisfying the time constraints in the STSST algorithm. In addition, the STSST algorithm uses the SRP protocol to enforce mutually exclusive access to shared resources. To achieve further energy savings, we present a dynamic task synchronization for sporadic task (DTSST) algorithm which can reclaim the slack time generated from the early completion task. It combines the DVS technique and the DPM technique to reduce energy consumption.

The rest of the paper is organized as follows. In Section 2 the preliminaries are introduced. We describe a static task synchronization for sporadic task algorithm, called STSST, in Section 3. We present a dynamic task synchronization for sporadic task algorithm, called DTSST, in Section 4. The experimental results are presented in Section 5 and conclude with the summary in Section 6.

Table 1
Summary of major notations for the task model.

Notation	Description
T_i	A sporadic real time task
P_i	The minimum separation period between the release of two consecutive instances of the task T_i
D_i	The relative deadline of the task T_i
C_i	The worst case execution time of the sporadic task T_i
WCET	Worst case execution time
Z_i	A list of critical section of the task T_i
$Z_{i,\ j}$	The jth critical section of the task T_i.
$T_{i,\ k}$	The kth invocation of the task T_i
U_{tot}	The system utilization
E_O	The energy overhead from the dormant mode to the active mode
P_{idle}	The power consumption of the processor when it is in idle modes
t_o	The break-even time
B_i	The worst case blocking time of the task T_i
S_L	The low speed
S_H	The high speed
$LLB(n)$	The lower bound of the system utilization
B_j'	The actual blocking time from which the task T_j can block the task T_i
$P(T_i)$	The priority of the task T_i
t'	The earliest time that a task will miss its deadlines
t''	The latest time point that no task is released before t'' and has a deadline at or before t'
$D_{t,t'}$	The total processor demand in $[t'', t']$
t_d	The absolute deadline of the task T_i
t_h	The time that the task T_i is blocked
β	A set of the tasks which are released at or after t'' and have deadline at or before t'
$U_i^F(t)$	The total unused time in the FRT-list at time t that can used by the task T_i
$U_i^{rem}(t)$	The available time of the task T_i at time t
$W_i^{rem}(t)$	The worst case residue execution time of the task T_i under the maximum speed at time t.
S_{temp}	A dynamic slowdown factor
t_m	The earliest time that a task with priority level $P(T_i)$ will miss its deadline
$_l$	The latest time point that no task arrives before t_l or not any time budget in FRT-list with priority level higher or equal to $P(T_i)$
t_2	The arrive time of the task $T_k(t_2 < t_1)$
t_1	The latest time point before t_m that the time budget of the task with priority level lower than $P(T_i)$ is consumed

2. Preliminaries

2.1. System model

The system consists of a task set which has n canonical sporadic real time tasks, represented as $T = \{T_1, T_2, ..., T_n\}$. Each task T_i is described by a 4-tuple (P_i, D_i, C_i, Z_i).We assume that the relative deadline of the sporadic task is equal to its minimum separation period ($D_i = P_i$, for each task T_i). The sporadic task is sorted by ascending order of their minimum separation periods i.e. ($P_1 \le P_2, \cdots, \le P_n$). The major notations and associated descriptions used in this paper are summarized in Table 1.

The system utilization U_{tot} is equal to $\sum_{i=1}^{n} \frac{C_i}{P_i}$ and the sporadic task set is scheduled by the RM policy. Under RM policy, the shorter the period (minimum separation period), the higher the priority and the higher priority task executes firstly. The task set which is scheduled by the RM policy is feasible [19] if

$$U_{tot} \le n\left(2^{\frac{1}{n}} - 1\right) \tag{1}$$

Where n is the number of tasks. We assume that $U_{tot} \le LLB(n)$ and that the context's switch overhead is incorporated in the task WCET.

We assume that the sporadic task set has a set of m shared resources which are represented as $R = \{R_1, R_2, \cdots, R_m\}$ and are accessed by the tasks in a mutually exclusive manner. The semaphores, locks and monitors are the common method to synchronization [20]. The semaphores are used for task synchronization in this paper. When a task can