计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210100080

基于动态优先级设备低能耗调度算法

张忆文[1]　林铭炜[2]
1 华侨大学计算机科学与技术学院　福建 厦门　361021
2 福建师范大学数学与信息学院　福州 350117

摘　要　现有的嵌入式周期任务低能耗调度算法只考虑相互独立的任务模型,且仅仅用动态电压频率调节技术来降低能耗。针对这些不足,提出能够支持资源受限的周期任务模型,且同时利用动态电压频率调节技术和动态功耗管理技术来降低系统能耗的算法。所提算法包括设备调度和任务调度两部分。在设备调度阶段,利用动态功耗管理技术降低设备能耗。在任务调度阶段,利用最早截止期限优先策略调度任务,以及利用栈资源协议实现共享资源的互斥访问;任务开始以低速度执行,若其在执行过程中被阻塞,将以高速度执行,这样能够有效地降低处理器的能耗。实验结果表明,所提算法能够有效地降低系统能耗。

关键词:能耗管理;实时调度;设备;动态优先级
中图法分类号　TP316.2

Devices Low Energy Consumption Scheduling Algorithm Based on Dynamic Priority

ZHANG Yi-wen[1] and LIN Ming-wei[2]
1 College of Computer Science and Technology,Huaqiao University,Xiamen,Fujian 361021,China
2 College of Mathematics and Informatics,Fujian Normal University,Fuzhou 350117,China

Abstract　Previous studies consider independent periodic task model and only apply dynamic voltage frequency scaling (DVFS) to reduce energy consumption. An algorithm that can support preemptive periodic tasks with non-preemptive shared resources is proposed to overcome this shortcoming. It combines DVFS and dynamic power management (DPM) techniques to reduce energy consumption. It consists of device scheduling and job scheduling. In device scheduling,DPM technique is used to reduce the energy consumption of IO devices. In job scheduling,the earliest deadline first policy is used to schedule tasks and the stack resource protocol is used as synchronization protocol for shared resources. In addition,the task executes at low speed without blocking and switches to high speed with blocking to reduce the energy consumption of the processor. The experimental result shows that the proposed algorithm can yield significantly energy savings with respect to the existing algorithm.

Keywords　Energy management,Real-time scheduling,Devices,Dynamic priority

1　引言

大多数嵌入式系统都是采用电池供电,例如手机、笔记本电脑、掌上电脑等,这些设备由于体积和重量的限制,它们都有能耗的限制。因此,低能耗是嵌入式系统的主要目标。嵌入式系统的能耗主要来源于CPU、内存和IO设备等。动态电压频率调节[1](DVFS)技术和动态功耗管理[2](DPM)技术是降低系统能耗的主要技术。

DVFS技术根据系统的负载,动态地调节处理器电压和频率,降低了处理器的能耗。很多研究工作[3-8]将DVFS技术与实时调度理论结合起来降低处理器的能耗。这些研究都有一个共同的特点,就是在满足硬实时任务截止期限的前提下,根据系统产生的空闲时间,利用DVFS技术动态地调节处理器速度,降低处理器的能耗。

DPM技术根据设备的空闲时间来决定是否将设备切换到低功耗状态,以降低设备能耗。目前,有不少研究者针对嵌入式系统IO设备能耗问题展开研究[9-12]。这些研究工作都有一个共同点,也就是在满足硬实时任务截止期限的前提下,根据设备的空闲时间,利用DPM技术将处于空闲状态的设备切换到低功耗状态以降低设备能耗。

上述研究要么仅利用DVFS技术降低处理器能耗,要么仅利用DPM技术降低IO设备能耗。而对于嵌入式系统而言,它不仅包括CPU,还包括IO设备、内存等组件,而这些组件也会产生能耗。将DVFS技术与DPM技术结合起来,同时降低CPU和IO设备能耗的研究相对较少[13-14]。文献[13]针对相互独立的周期任务模型,提出了设备禁止区域单调速率(DFR-RMS)算法。该算法利用单调速率策略调度任务,通过建立设备禁止区域来延长设备的空闲时间,再利用DPM技术将设备切换到低功耗状态降低能耗。此外,DFR-RMS还可以利用DVFS技术降低处理器能耗。然而,DFR-RMS算法只能适用于采用固定优先级策略的系统。文献[14]针对采用动态优先级策略的嵌入式系统的CPU和IO设备能耗问

基金项目:福建省自然科学基金(2019J01080);厦门市青年创新基金(3502Z20206012);国家自然科学基金(61872086)
This work was supported by the Natural Science Foundation of Fujian Province of China(2019J01080),Youth Innovation Fund Projects of Xiamen City(3502Z20206012) and Natural Science Foundation of of China(61872086).
通信作者:张忆文(zyw@hqu.edu.cn)

题开展研究，提出了设备禁止区域最早截止期限优先(DFR-ED)算法。该算法利用最早截止期限优先策略调度任务，通过设备禁止区域，利用 DVFS 技术与 DPM 技术来降低能耗。

但这些研究只能支持相互独立的周期任务模型，不能支持资源受限的周期任务模型。针对现有研究工作的不足，提出了基于动态优先级设备低能耗调度算法。该算法利用最早截止期限优先策略调度任务，同时结合 DVFS 技术与 DPM 技术，能够支持资源受限的周期任务模型。

2 系统模型

2.1 任务模型

在单处理器系统中，考虑有 n 个资源受限周期任务的周期任务集 $T=\{T_1,T_2,\cdots,T_n\}$。周期任务 T_i 用四元组 $(P(T_i),W(T_i),Dev(T_i),Rs(T_i))$[10] 表示，$P(T_i)$ 是 T_i 的周期，$W(T_i)$ 是 T_i 在最坏情况下的执行时间，$Dev(T_i)$ $(Dev(T_i)=\{D_1,D_2,\cdots,D_m\})$ 是 T_i 执行时所用到的设备集合，$Rs(T_i)$ $(Rs(T_i)=\{R_1,R_2,\cdots,R_k\})$ 是 T_i 执行时所用到的非抢占资源集合。非抢占的设备可以被认为是非抢占的资源。不同的任务使用同一设备的时间不同。可抢占的设备属于集合 $Dev(T_i)$，但不属于集合 $Rs(T_i)$。T_i 的第 j^{th} 个实例用 $T_{i,j}$ 表示，$T_{i,j}$ 所需的设备集合用 $Dev(T_{i,j})$ 表示，且 $Dev(T_{i,j})=Dev(T_i)$。T_i 的利用率 $U_i=\dfrac{W(T_i)}{P(T_i)}$，周期任务集 T 的利用率 $U_{tot}=\sum_{i=1}^{n}U_i$。$T_i$ 的优先级用 $P_r(T_i)$ 表示，利用最早截止期限优先[15](EDF)策略调度周期任务集 T。EDF 策略根据任务的绝对截止期限分配任务的优先级，任务的绝对截止期限越近，其优先级就越高；任务的绝对截止期限越远，其优先级就越低。利用栈资源协议[16](SRP)来确保资源之间的互斥访问。

2.2 设备模型和能耗模型

每个设备存在休眠状态和活跃状态。采用任务间的设备调度[14]，即任务 T_i 开始执行时，其所需的所有设备必须处于活跃状态，设备间的状态转化存在着时间开销与能耗开销。设备 D_i 处于活跃状态和休眠状态的功耗分别为 P_i^a 和 P_i^s；设备 D_i 从活跃状态切换到休眠状态和从休眠状态切换到活跃状态的时间开销分别为 t_i^{as} 和 t_i^{sa}；设备 D_i 从活跃状态切换到休眠状态和从休眠状态切换到活跃状态的能耗开销分别为 E_i^{as} 和 E_i^{sa}。设备 D_i 的临界时间 B_i 如式(1)所示：

$$B_i=\max\{t_i^{as}+t_i^{sa},\frac{E_i^{as}+E_i^{sa}-[P_i^s\cdot(t_i^{as}+t_i^{sa})]}{P_i^a-P_i^s}\} \tag{1}$$

设备 D_i 的能耗开销 E_i^{dev} 的计算方法如下：

$$E_i^{dev}=E_i^{act}+E_i^{tran} \tag{2}$$

其中，E_i^{act} 是设备 D_i 在活跃状态的能耗，E_i^{tran} 是设备 D_i 的状态转化的能耗。大多数在线的 DPM 算法都是通过预测设备的空闲时间，来决定是否将设备切换到低功耗状态，以降低能耗。

系统能耗主要来自 CPU 和设备，其计算方法如下：

$$E_{tot}=E_{cpu}+E_{dev}=E_{cpu}+\sum_{i=1}^{m}E_i^{dev} \tag{3}$$

其中，E_{cpu} 是 CPU 在执行周期任务集 T 时所消耗的能耗，E_{dev} 是所有设备的能耗。CPU 的功耗模型可以近似为 $P(S)=\alpha S^3$，其中 α 是转化电容，S 是归一化的处理器速度。假设处理器能够提供连续的归一化速度[14]，即 $S\in[S_{\min},1.0]$，其中 $S_{\min}$ 是处理器最小的速度。

3 动态优先级调度算法

3.1 研究动机

文献[12]研究了设备低能耗调度问题，提出了基于 EDF 策略的 DFR-EDF 算法。DFR-EDF 算法通过利用设备禁止区域来延长设备空闲时间以降低能耗。D_i 的设备禁止区域用 FR_i 表示。FR_i 可以通过二元组 (Δ_i,Π_i) 描述，其中 Δ_i 是 FR_i 的长度且 $\Delta_i>B_i$，Π_i 是两个连续 FR_i 的最小间隔，即两个连续的 FR_i 之间的时间至少为 Π_i。FR_i 存在活跃状态和休眠状态。当 FR_i 处于活跃状态时，使用 D_i 的任务将被阻塞直到 FR_i 结束(变为休眠状态)。Δ_i 由式(4)计算：

$$B_i<\Delta_i\leqslant\min_{T_j\in\gamma_i}(P(T_j)-W(T_j)) \tag{4}$$

其中，γ_i 需要使用 D_i 的任务集合。Π_i 由式(5)计算：

$$\frac{\Delta_i}{1-UD_i}\leqslant\Pi_i\leqslant\max_{T_j\in\gamma_i}\{P(T_j)\} \tag{5}$$

其中，UD_i 是任务集合 γ_i 的总利用率。

定理 1[14] 按照周期进行非降序排序的周期任务集 T 和设备禁止区域集合 $\Gamma=\{(\Delta_1,\Pi_1)\cdots(\Delta_m,\Pi_m)\}$，DFR-EDF 算法调度周期任务集 T 是可行的，必须满足以下条件：

$$\sum_{i\in\gamma_k}\left(\frac{\Delta_i}{\Pi_i}+\frac{\Delta_i}{P(T_j)}\right)+\sum_{j=1}^{k}\frac{W(T_j)}{P(T_j)}\leqslant 1,\forall k(k=1,\cdots,n) \tag{6}$$

其中，γ_k 代表 k 个最小周期任务使用的设备禁止区域的集合。

通过下文的任务集来解释 DFR-EDF 算法存在的不足。考虑存在 3 个周期任务的任务集 $T=\{T_1,T_2,T_3\}$，周期任务的具体参数信息如下：

$T_1=(120,30,D_1,R_1)$，$T_2=(150,20,\varnothing,\varnothing)$，$T_3=(600,90,\varnothing,R_1)$

其中，T_1 和 T_3 共享非抢占的资源 R_1，T_1 在执行过程中需要使用 D_1。假设 D_1 的临界时间是 25($B_1=25$)，其设备禁止区域的长度为 30 ($\Delta_1=30$)，两个连续的设备禁止区域的最小间隔为 120 ($\Pi_1=120$)。此外，假设 T_1 和 T_2 的第一个实例在时刻 0 释放，T_3 的第一个实例在时刻 110 释放。根据定理 1，DFR-EDF 算法调度周期任务集 T 是可行的。DFR-EDF 算法在区间[0，600]调度周期任务集 T 的调度结果如图 1 所示。

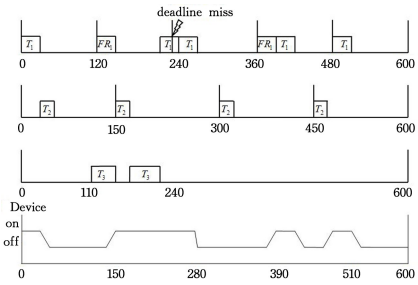

图 1 DFR-EDF 算法调度周期任务集 T

Fig. 1 DFR-EDF algorithm schedules periodic task set T

图 1 中，T_1 在时刻 0 开始执行；在时刻 30，T_1 完成执行且 T_2 开始执行；在时刻 50，T_2 完成执行；在时刻 110，T_3 开始执行，且其在时刻 150 被 T_2 抢占执行。尽管 T_1 在时刻 120 释放了一个任务实例，但由于其执行所需的资源被 T_3 占用，因此它被阻塞。当 T_2 完成执行时，T_3 恢复执行，且其在时刻 220 完成执行。T_1 恢复执行，但其在时刻 240 错过截止期限。其他任务以相似的方式进行调度，这里不再赘述。为

了弥补 DFR-EDF 算法存在的不能满足任务截止期限的不足，将提出新的算法。

3.2 计算任务的执行速度

由于任务共享非抢占资源，高优先任务可能会被低优先级任务阻塞。借用文献[17]的思想，当没有阻塞发生时，任务以低速度 S_L 执行；当有阻塞发生时，任务以高速度执行。为了有效地降低 CPU 能耗，高速度的计算方法与文献[17]中的方法不同。

定理 2[15] n 个相互独立的周期任务按照其周期进行非降序排序，EDF 算法调度可行的条件为：

$$\sum_{i=1}^{n}\frac{W(T_i)}{P(T_i)}\leqslant 1 \tag{7}$$

根据定理 2，低速度 S_L 由式(8)计算：

$$S_L\geqslant\sum_{i=1}^{n}\frac{W(T_i)}{P(T_i)} \tag{8}$$

定理 3[17] 资源受限的周期任务集 T 按照周期进行非降序排列，EDF 算法调度 T 且 T_i 以高速度 S_{H_i} 执行可行的条件为：

$$\frac{1}{S_{H_i}}(\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}\left\lfloor\frac{t}{P(T_k)}\right\rfloor\cdot W(T_k))\leqslant t, P(T_1)\leqslant t\leqslant P(T_i), \text{且 } S_L\leqslant S_{H_i} \tag{9}$$

其中，G_j 是 T_j 的最大阻塞时间 $(1\leqslant j\leqslant n)$。

根据定理 3，任务 T_i 的高速度$(1\leqslant j\leqslant n)$由式(10)计算：

$$\frac{(\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}\left\lfloor\frac{t}{P(T_k)}\right\rfloor\cdot W(T_k))}{t}\leqslant S_{H_i}, P(T_1)\leqslant t\leqslant P(T_i), \text{且 } S_L\leqslant S_{H_i} \tag{10}$$

3.3 设备调度

对于任务间的设备调度，T_i 准备执行时，其所需要的设备必须都处于活跃状态。因此，不同的 DPM 策略将导致不同的调度结果。为了简单起见，我们使用简单的在线 DPM 策略(DPM-EDF)来调度设备。因为设备状态之间的转化存在时间开销和能耗开销，按照固定的周期将设备切换到休眠状态的节能效果并不理想。

DPM-EDF 根据设备的临界时间来决定是否将设备切换到休眠状态。当设备空闲时间长于其临界时间时，将设备切换到休眠状态以降低设备能耗，因此最重要的就是预测设备空闲时间。没有完成执行的任务需要使用 D_i 的最早时间用 $N(D_i)$表示。对于任务而言，只有其释放实例之后才能使用设备。因此，$N(D_i)$可以由式(11)计算：

$$N(D_i)=\min(RT(T_{i,j})) \tag{11}$$

其中，$T_{i,j}$ 没有完成执行且需要使用设备 D_i，$RT(T_{i,j})$是 $T_{i,j}$ 的释放时间。DPM-EDF 在任务的调度点对设备的状态进行切换。所谓任务调度点指任务释放实例、完成执行或者离开关键区的时刻。当 $N(D_i)-t>B_i$ 时，可以将 D_i 切换到休眠状态。处于休眠状态的设备 D_i 被重新激活的时刻用 $Up(D_i)$表示。DPM-EDF 的详细描述如算法 1 所示。

算法 1 DPM_EDF

```
1. 在调度点 t 的设备调度
2.   If(任务实例 T_{i,j}完成执行)
3.   If(∃D_i,D_i=active and N(D_i)-t>B_i) //active 表示设备处于活跃状态
4.     D_i=sleep;// sleep 表示设备处于休眠状态
5.         Up(D_i)=N(D_i)-t_i^sa;
6.     End
7. End
8.   If (∃D_i,D_i=sleep and Up(D_i)=t)
9.     D_i=active;
10.       Up(D_i)=0;
11. End
```

3.4 算法

所提出的基于动态优先级设备低能耗调度算法（LPSA）包括两个部分：任务调度和设备调度。设备调度采用 DPM-EDF，任务调度通过 SRP 协议来确保资源的互斥访问。当没有阻塞发生时，任务以低速度 S_L 执行；当高优先级任务被低优先级任务阻塞时，低优先级任务将以高速度执行。LPSA 算法的详细描述如算法 2 所示。

算法 2 LPSA 算法

```
1. 初始条件：计算 S_L 和 S_Hi (S_Hi>1，设置 S_Hi=1)
2. 调度点 t 的调度
3. If (T_i释放之前处理器处于空闲状态) Then，任务 T_i 以低速度 S_L
   执行；
4. End
5. If (P_r(T_i)<P_r(T_j))
6.       If (任务 T_i 抢占任务 T_j)
7.       Then，任务 T_i 以低速度 S_L 执行；
8.   Else /* 任务 T_i 被任务 T_j 阻塞 */
9.     Then，任务 T_j 以高速度 S_Hi 执行；
10. End
11. If (任务 T_j完成执行)
12.  Then，被阻塞的任务 T_i 以高速度 S_Hi 执行
13. End
14.  利用 DPM-EDF 调度设备
```

计算低速度 S_L 的时间复杂度为 $O(n)$，高速度 S_{H_i} 的时间复杂度为 $O(n^2)$；DPM-EDF 策略调度设备的时间复杂度是 $O(n)$，使用 EDF 和 SRP 协议调度任务的时间复杂度是 $O(n)$；因此 LPSA 算法的时间复杂度为 $O(n^2)$。

3.5 可行性

因为任务准备执行时，其所使用的设备都必须处于活跃状态，所以 DPM-EDF 策略不会影响任务调度的可行性。若要证明 LPSA 算法是可行的，只需要证明任务以低速度或者高速度执行，所有的任务都能够在其截止期限内完成执行且能够确保互斥地使用资源，接下来的定理将证明 LPSA 算法是可行的。

定理 4 资源受限的周期任务 T，按照任务的周期进行非降序排序，LPSA 算法调度 T 可行，式(8)和式(10)成立。

证明：参考文献[18]中的方法，假设 LPSA 算法调度周期任务集 T，存在任务错过截止期限。t'是任务错过截止期限的最早时刻，t''是 t'之前最近的时间点，即没有任务在 t''之前释放且没有任务的截止期限在 t'之前。假如 t''不存在，令 $t''=0$。因此，在区间$(t'',t']$中不存在空闲时间。将区间$(t'',t']$的任务分成两种情况：第一种情况，任务都在 t''之后释放且其截止期限在 t'之前，用 β 表示满足第一种情况的所有任务的集合；第二种情况，在 t''之前处理器处于空闲状态或者存在 T_j 在 t''之前释放且截止期限在 t'之后。

第一种情况：在区间$(t'',t']$没有阻塞发生

在第一种情况中，只有任务集 β 中的任务在区间$(t'',t']$执行，且所有的任务都以低速度 S_L 执行。在区间$(t'',t']$中的处理器需求用 $D_{t,t'}$ 表示，其值为 $\sum_{i=1}^{n}\frac{W(T_i)}{S_L}\cdot\left\lfloor\frac{t'-t''}{P(T_i)}\right\rfloor$。因为

在区间$(t'',t']$不存在空闲时间，且有任务在t'错过截止期限，所以有：

$$\sum_{i=1}^{n}\frac{W(T_i)}{S_L}\cdot\left\lfloor\frac{t'-t''}{P(T_i)}\right\rfloor>t'-t'' \tag{12}$$

进而推出：

$$\sum_{i=1}^{n}\frac{W(T_i)}{P(T_i)}>S_L \tag{13}$$

这与式(12)矛盾。

第二种情况：在区间$(t'',t']$有阻塞发生。

假设T_j在t''之前已经开始执行。首先，讨论T_j在区间$(t'',t']$只会引起一次阻塞。假设T_i在区间$(t'',t']$被T_j阻塞，很显然T_i是集合β中优先级最高的任务。T_i的截止期限和被阻塞的时刻分别为t_d和t_h。T_j开始是以低速度S_L执行，当阻塞发生时，T_j以高速度S_{H_i}执行直到其完成执行。T_i以高速度S_{H_i}恢复执行直到其完成执行。在区间$(t'',t']$的处理器需求可以分为3部分：

(1)在区间$(t'',t_h]$，任务以低速度执行，其处理器需求为$\sum_{k=1}^{i}\frac{W(T_k)}{S_L}\left\lfloor\frac{t_h-t''}{P(T_k)}\right\rfloor$；

(2)在t_h，T_i被T_j阻塞，T_j以高速度执行，因此，在区间$[t_h,t_d]$的处理器需求为$\frac{\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}W(T_k)\cdot\left\lfloor\frac{t_d-t_h}{P(T_k)}\right\rfloor}{S_{H_i}}$，其中$G_j$是$T_j$的最大阻塞时间；

(3)在区间$[t_d,t']$没有阻塞发生，任务以低速度S_L执行，处理器需求为$\sum_{k=1}^{i}\frac{W(T_k)}{S_L}\cdot\left\lfloor\frac{t'-t_d}{P(T_k)}\right\rfloor$。

由于有任务在t'错过截止期限，因此在区间$(t'',t']$的处理器需求大于区间$(t'',t']$的长度，故式(14)、式(15)成立。

$$\sum_{k=1}^{i}\frac{W(T_k)}{S_L}\cdot\left\lfloor\frac{t_h-t''}{P(T_k)}\right\rfloor+\sum_{k=1}^{i}\frac{W(T_k)}{S_L}\cdot\left\lfloor\frac{t'-t_d}{P(T_k)}\right\rfloor+\frac{\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}W(T_k)\cdot\left\lfloor\frac{t_d-t_h}{P(T_k)}\right\rfloor}{S_{H_i}}>t'-t'' \tag{14}$$

$$\Rightarrow\frac{\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}W(T_k)\cdot\left\lfloor\frac{t_d-t_h}{P(T_k)}\right\rfloor}{S_{H_i}}+\sum_{k=1}^{i}\frac{W(T_k)}{P(T_k)\cdot S_L}\cdot(t_h-t''+t'-t_d)>t'-t'' \tag{15}$$

根据式(10)，进而推出：

$$\Rightarrow\frac{\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}W(T_k)\cdot\left\lfloor\frac{t_d-t_h}{P(T_k)}\right\rfloor}{S_{H_i}}>t'-t''-(t_h-t'')-(t'-t_d) \tag{16}$$

$$\Rightarrow\frac{\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}W(T_k)\cdot\left\lfloor\frac{t_d-t_h}{P(T_k)}\right\rfloor}{S_{H_i}}>t_d-t_h \tag{17}$$

令$t=t_d-t_h$，且进行不等式变化：

$$\Rightarrow\frac{\max\{G_j\mid P(T_i)<P(T_j)\}+\sum_{k=1}^{i}W(T_k)\cdot\left\lfloor\frac{t}{P(T_k)}\right\rfloor}{S_{H_i}}>t \tag{18}$$

可以推出式(18)与式(10)矛盾。

接下来讨论在区间$(t'',t']$不止发生一次阻塞，因为T_j是唯一一个释放时间在t''之前且在t'之后完成执行的任务，所以T_j阻塞或者被集合β中的任务阻塞。当任务被T_j阻塞，这种情况已经在计算高速度时进行了考虑。当任务被集合β中的任务阻塞，任务的阻塞时间是集合β中任务执行时间的一部分，这并不会影响处理器需求的增加。因此，定理得证。

3.6 算法实例

使用LPSA算法重新调度周期任务集T，通过计算可知$S_L=0.53$，$S_{H_1}=1.0$。LPSA算法在区间$[0,600]$调度任务的结果如图2所示。

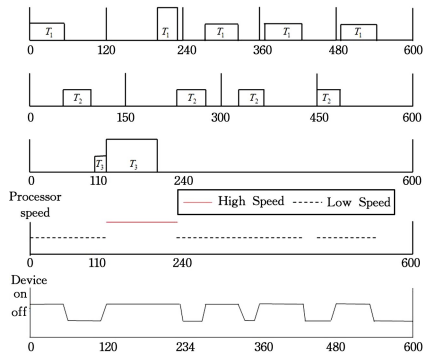

图2 LPSA算法调度周期任务集T

Fig. 2 LPSA algorithm schedules periodic task set T

如图4所示，T_1以低速度S_L执行，在时刻57完成执行。在时刻57，T_2以速度S_L执行，且其在时刻95完成执行。在时刻110，T_3以低速度S_L执行；在时刻120，T_1被T_3阻塞，此时T_3以高速度S_{H_1}执行，且其在时刻204完成执行。在时刻204，T_1以高速度S_{H_1}执行，且其在时刻234完成执行。其它任务以相似的方法调度，这里不在赘述。从图4可以看出，所有的任务都能够在其截止期限内完成执行。

4 仿真实验

为了评价LPSA算法的性能，用C语言实现一个周期任务调度的仿真器。该仿真器基于PXA 270[4]处理器。该处理器的功耗模型可以近似为$P=0.08+1.52*S^3$，其可提供的归一化速度范围为$[0.15,1.0]$。随机产生100个周期任务集，每个周期任务集包含15个周期任务。T_i的周期[14]在区间$[25\text{ ms},1\,300\text{ ms}]$中随机选择；$T_i$的最坏情况下执行时间在1到其周期之间随机选择。通过调节任务最坏情况下的执行时间，确保系统利用率不超过给定的值。T_i从SST Flash SST39LF020，SimpleTech Flash Card，and Realtek Ethernet Chip等设备中随机选择0或者1个设备，设备的具体参数请参见文献[12]。T_i需要的设备在实验中被认为是非抢占的，因此T_i的资源集合和设备集合相同。

在周期任务调度仿真器中实现3种算法：1)LPSA算法；2)DPM_EDF，任务始终以最大处理器速度执行，利用DPM技术降低设备能耗；3)DS算法[17]，没有利用DPM技术节能，所有设备处于活跃状态且任务以低速度或者高速度执行。由于文献[13]和文献[14]提出的算法没有考虑资源共享问题，所有在仿真实验中没有将它们作为对比算法。利用归一化节约能耗来评价算法的性能，节约能耗指没有使用节能技术的算法相对于能耗感知算法的能耗差值。归一化节约能耗由式(19)计算：

$$\text{归一化节约能耗}=1-\frac{\text{能耗感知算法的能耗}}{\text{没有使用节能技术算法的能耗}} \tag{19}$$

将实验的仿真时间设置为 1 000 000 个时间片，系统利用率从 0.15 变化到 0.8，步长为 0.05，考察系统利用率对归一化节约能耗的影响。实验结果如图 3 所示。

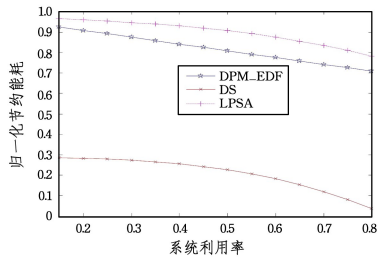

图 3 归一化能耗

Fig. 3 Normalized energy consumption

从图 5 可以看出，所提算法节约能耗都受到了系统利用率的影响。随着系统利用率的增加，所有算法的归一化节约能耗都降低。这是因为，系统利用率增加，系统可利用的空闲时间缩短，进而导致 DVFS 或者 DPM 技术被用来降低能耗的机会减少。相比其他算法，LPSA 算法节约的能耗更多。LPSA 算法与 DPM_EDF 和 DS 算法相比分别可以多节约大约 8.10%和 69.17%的能耗。其主要原因在于，LPSA 不仅可以利用 DVFS 技术降低处理器能耗，而且可以利用 DPM 技术降低设备能耗。此外，DPM_EDF 算法比 DS 算法可以多节约大约 61.07%的能耗。这主要是因为系统的能耗主要来自设备的能耗，而 DS 算法中的设备一直处于活跃状态。

结束语 所提出的 LPSA 算法能够支持资源受限的周期任务模型，且能够同时利用 DVFS 技术和 DPM 技术降低系统能耗。LPSA 算法在设备调度阶段利用 DPM 技术降低设备能耗。在任务调度阶段，利用 EDF 策略调度任务，利用 SRP 协议确保资源的互斥使用，任务开始以低速度执行，当其在执行过程中被阻塞时，将以高速度执行，有效地降低了处理器能耗。此外，利用时间需求分析的方法，证明 LPSA 算法是可行的。仿真实验表明，LPSA 算法能够有效地降低系统能耗。LPSA 算法考虑单处理任务模型，忽略了处理器速度切换开销；接下来将研究多处理器任务模型且考虑速度切换开销的能耗调度问题。

参 考 文 献

[1] TAHERIN A, SALEHI M, EJLALI A. Reliability-Aware Energy Management in Mixed-Criticality Systems[J]. IEEE Trans on Sustainable Computing, 2018, 3(3): 195-208.

[2] WEI J, PAN X, JIANG K, et al. Energy-Aware Design of Stochastic Applications With Statistical Deadline and Reliability Guarantees[J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 2019, 38(8): 1413-1426.

[3] ZHANG Y W, LI H B. Energy aware mixed tasks scheduling in real-time systems [J]. Sustainable Computing-Informatics & Systems, 2019, 23: 38-48.

[4] ZHANG Y W. Energy-aware mixed partitioning scheduling in standby-sparing systems[J]. Computer Standards & Interfaces, 2019, 61: 129-136.

[5] ZHANG Y W. System level fixed priority energy management algorithm for embedded real time application[J]. Microprocessors and Microsystems, 2019, 64: 170-177.

[6] ZHANG Y W. Energy-aware Mixed-criticality Sporadic Task Scheduling Algorithm[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021, 40(1): 78-86.

[7] ZHANG Y W, WANG C, GUO R F. Resource constrained periodic task low power scheduling algorithm[J]. Journal of Chinese Computer Systems, 2017, 38(5): 1076-1080

[8] ZHANG Y W, WANG C, LIN J. Energy aware fixed priority scheduling for real time sporadic task with task synchronization [J]. Journal of Systems Architecture, 2018, 83: 12-22.

[9] SWAMINATHAN V, CHAKRABARTY K. Energy-conscious, deterministic I/O device scheduling in hard real-time systems [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(7): 847-858.

[10] SWAMINATHAN V, CHAKRABARTY K. Pruning-based, energy-optimal, deterministic I/O device scheduling for hard real-time systems[J]. ACM Transactions on Embedded Computing Systems, 2005, 4(4): 141-167.

[11] CHENG H, GODDARD S. Online Energy-Aware I/O Device Scheduling for Hard Real-Time Systems[C]// Proc. of Design, Automation and Test in Europe (DATE'06). 2006: 1- 6.

[12] CHENG H, GODDARD S. EEDS_NR: an online energy-efficient I/O device scheduling algorithm for hard real-time systems with non-preemptible resources[C]// Proc. of 18th Euromicro Conference on Real-Time Systems (ECRTS'06). 2006: 251-260.

[13] DEVADAS V, AYDIN H. Real-Time Dynamic Power Management through Device Forbidden Regions[C]// Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium. 2008: 34-44.

[14] DEVADAS V, AYDIN H. DFR-EDF: A Unified Energy Management Framework for Real-Time Systems[C]// Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium. 2010: 121-130.

[15] LIU C L, LAYLAND J W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment[J]. In Journal of the Association for Computing Machinery, 1973, 20(1): 46-61.

[16] BAKER T P. Stack-Based Scheduling of Real time Processes [J]. The Journal of Real-Time Systems, 1991, 3(1): 67-99.

[17] ZHANG F, CHANSON S T. Blocking-aware processor voltage scheduling for real-time tasks[J]. ACM Trans on Embedded Computing Systems, 2004, 3(2): 307-335.

[18] ZHANG Y W, WU W J, GUO R F. Resource Constrained Periodic Task Dual Speed Scheduling Algorithm[J]. Journal of Chinese Computer Systems, 2018, 39(9): 2119-2123.

ZHANG Yi-wen, born in 1988, Ph.D, associate professor, is a member of China Computer Federation. His main research interests include real-time system and low-power scheduling.

LIN Ming-wei, born in 1985, Ph.D, professor. His main research interests include fuzzy decision-making analysis and storage performance optimization.