文章编号:1000-5862(2016)03-0307-05

基于离散粒子群优化的 MPSoC 节能调度算法

于国龙,崔忠伟,左 羽
(贵州师范学院数学与计算机科学学院,贵州省高校工业物联网工程技术研究中心,贵州 贵阳 550018)

摘要:为了降低多核片上系统 MPSoC 在应用中的能耗,在 MPSoC 上提出了基于优化离散粒子群算法的节能任务调度算法. 通过比例选择算子生成初始种群,以任务在 MPSoC 上不同内核执行的能耗作为解空间,粒子群在整个解空间上搜索最低能耗调度方案,并在算法中优化了粒子群算法的局部早熟问题,使算法性能进一步提升. 仿真实验表明:基于优化离散粒子群算法的节能调度算法与常用的 3 种调度算法相比,能耗得到了降低,且算法的截止期错失率并没有升高,保证了算法的整体性能.

关键词:粒子群算法;多处理器片上系统;节能;调度算法
中图分类号:TP 302.1 文献标志码:A DOI:10.16357/j.cnki.issn1000-5862.2016.03.19

0 引言

多处理器系统级芯片 MPSoC 目前已广泛应用于嵌入式系统,由于体积和功能的需要,系统能量一般非常有限,硬件能耗控制往往在技术上是有一定极限的,那么在应用时芯片上的任务调度算法节能性能成为 MPSoC 节能的另一个重要考虑的研究方向. 如文献[1]提出针对多核处理器上具有依赖关系的周期性硬实时任务,设计了一种基于动态电压调节的节能任务调度方法;该方法先用 RDAG 算法将任务独立化,然后以功耗最低为原则,用遗传算法确定任务映射. 文献[2]提出了一种新的能量感知的调度算法周期节约动态电压缩放(偶发任务 cc-dvsst),它是 dvsst 算法的改进;文献[3]提出一种基于分组的自适应任务调度算法,它能根据能量收集单元,由于能量输出的不确定性而造成的非能量约束情况和能量约束情况,自适应地选择任务调度算法[4-5].

上述节能调度算法都起到了一定的 MPSoC 节能的作用,但在将复杂的系统任务映射到多核心处理器上,并且满足功耗,任务截止期错失率,任务最后期限等约束条件,这时就是一个 NP 完全问题,是组合优化问题中的一种,没有从组合优化这个本质角度来设计算法. 另外,在节能的前提下,整体性能没有充分考虑,如算法的任务截止期错失率过高. 在应用中,由于粒子群(PSO)算法具有较好的并行处理解决问题的特性,可以在较大状态空间随机高效地采样和搜索,并快速收敛到最优或近似最优解,能较好地解决组合优化的 NP 问题[6-9]. 因此,本文提出基于优化离散粒子群算法的 MPSoC 节能调度算法,实验表明算法节能效果要优于常用的 3 种基于能耗的调度算法,并且算法的任务截止期错失率并没有升高,保证了算法整体性能.

1 MPSoC 节能调度算法设计

1.1 基于混沌搜索的离散粒子群算法优化

Kennedy 等于 1997 年率先提出了一种针对 0-1 规划问题的二进制 PSO(binary PSO,BPSO)算法,在 BPSO 中,一个二进制的空间就表示一个超立方体空间,每个粒子用一个二进制变量来表示,可以通过该二进制变量的某些位在{0,1}之间的翻转来实现粒子在这个超立方体空间中的移动[10-12]. 算法中粒子的速度则用于表示二进制变量的翻转概率,这样离散粒子群算法和基本粒子群算法就较相似,它们最大的区别在于离散粒子群算法的速度和位置更新稍有不同,具体的公式为[13-15] $v_i(k+1) = \omega v_i(k) + c_1 r_1(x_{p_i}(k) - x_i(k)) + c_2 r_2(x_{p_g}(k) - x_i(k))$,
其中 x_i 为粒子的位置,v_i 为粒子位置的变化率,ω 为

收稿日期:2016-02-13
基金项目:贵州省 2014 年省级本科教学工程项目(黔教高发〔2014〕378)和 2015 年省级本科教学工程建设项目(黔教高发〔2015〕337)和卓越工程师教育培养计划(黔教高发〔2013〕446)资助项目.
作者简介:于国龙(1981-),男,辽宁东港人,讲师,主要从事物联网和大数据的研究.

惯性权重 c_1 和 c_2 的学习因子,也称加速常数,r_1 和 r_2 为 [0,1] 范围内的均匀随机数,而 p_i 和 p_g 分别表示粒子的局部最优位置和全局最优位置,x_i、p_i 和 p_g 只能是 0 或 1,v_i 表示概率,取值在 [0,1] 之间,粒子位置的更新公式为

$$x_i(k+1) = \begin{cases} 1, rand(\cdot) < sig(v_i^{k+1}), \\ 0, 其他 \end{cases}$$

其中 $sig(v_i) = 1/(1+\exp(-v_i^k))$,函数 $sig(v_i^{k+1})$ 是一个转换限制函数,使 x_i 的每一个分量都限制在 [0,1] 之间,而 $rand(\cdot)$ 表示一个 [0,1] 之间的随机数,v_i 值越大,粒子的 x_i 选 1 的概率越大,v_i 值越小,x_i 选 0 的概率则越大.

但在实际应用中,离散粒子群算法在多次迭代后,由于各粒子向最优解靠拢时会趋向于同一性,导致离散粒子群算法会陷入局部早熟,通过混沌搜索生成一个混沌序列来替代早熟的粒子,使早熟粒子能及时跳出局部最优,尽快地搜索到最优解[16].

当粒子群中某粒子陷入局部最优时,首先混沌搜索就随机生成一个初始 D 维空间,$\boldsymbol{u}_0 = (u_{01}, u_{02}, \cdots, u_{0D})^{\mathrm{T}}$,$u_{0d} \in [0,1]$,再通过

$$u_{(i+1)d} = \begin{cases} 2u_{id} + rand(0,1)/50, & u_{id} \in [0,0.5], \\ 2 - 2u_{id} + rand(0,1)/50, & u_{id} \in [0.5,1], \end{cases}$$

将该个 D 维空间映射成一个混沌粒子序列,其中 $i = 0,1,2\cdots,N$,$d = 1,2\cdots,D$. 在映射后的粒子序列中,适应度最优的粒子将替换处于局部最优粒子,以达到使粒子跳出局部最优的目的.

1.2 任务能耗模型

通常一个 MPSoC 处理器包含 3 种主要的能量消耗: 动态功耗(P_{dynamic})、静态功耗(P_{static}) 和短路功耗(P_{short}).

动态功耗是由处理器运行任务时,CMOS 管充放电消耗的功耗,周期 T 内一次翻转功耗

$$P_{\mathrm{dynamic}} = K\frac{1}{2T}\left(\int_0^{T/2} V_0(-C_A \mathrm{d}V_0/\mathrm{d}t)\,\mathrm{d}t + \int_{T/2}^{T}(V_{DD}-V_0)\cdot(-c\mathrm{d}V_0/\mathrm{d}t)\,\mathrm{d}t\right),$$

解得 $P_{\mathrm{dynamic}} = KfC_A V_{DD}^2/2$,其中 K 为转换因子,由晶体管的物理特性决定,C_A 为晶体管的装载电容,f 为时钟频率,V_{DD} 为供能电压.

P_{static} 是指来自寄生二极管在加反向电压时,晶体管内出现的漏电现象产生的功率消耗为

$$P_{\mathrm{static}} = V_{dd}I_{ds} + |V_{bs}|I_{\mathrm{slc}}, P_{\mathrm{static}} = V_{dd}n\mu C_{OX}(W/L)\cdot(kT_h/q)^2 \mathrm{e}^{(V_{gs}-V_T)q/nkT_h}(1-\mathrm{e}^{-V_{ds}q/kT_h}) + |V_{bs}|I_s(T_0)2^{((T_h-T_0)/10)}(\mathrm{e}^{V_{dd}q/kT_h}-1),$$

其中 I_{ds} 为亚阈值电流,V_{bs} 为体偏电压,I_{slc} 为源漏电电流,n 为亚阈值斜率因子,μ 为载流子的迁移率,C_{OX} 为单位面积栅氧化层电容,W/L 为 MOS 管的宽长比,k 为玻耳兹曼常数,T_h 为热力学温度,q 为电子电荷量,V_{gs} 为漏极电压,V_T 为阈值电压,$I_s(T_0)$ 为参考温度点的反向饱和电流[17].

短路功耗是指所有在元件内部消耗的功率,例如 CMOS 晶体管在翻转过程中的短暂时间内,P 管和 N 管同时导通而形成电源和地之间短路电流造成的能耗,可以表示为 $P_{\mathrm{short}} = L\tau(V_{dd}-2V_T)^3 N_C f$,其中 L 为与管子大小和工艺有关的常数,τ 为信号上升或下降时间,V_T 为阈值电压,N_C 为反相器输出的平均管子数,f 为时钟频率.

处理器的总功耗 P_{total} 为 $P_{\mathrm{total}} = P_{\mathrm{dynamic}} + P_{\mathrm{static}} + P_{\mathrm{short}}$,其中动态功耗是主要的. 对 CMOS 电路的功耗估计和分析来说,最重要的任务是降低动态功耗. 但是在深亚微米工艺下,漏电流功耗逐渐成为主要功耗源,各种手持设备芯片的设计必须充分考虑漏电功耗的影响.

1.3 算法设计

在一个已知系统中,假设有 p 个处理器,定义处理器 $C = \{C_1, C_2, \cdots, C_p\}$,系统任务集 U_k 中有 m 个任务,则 $U_k = \{s_1, s_2, \cdots, s_m\}$,每个任务 $s_i(1 \leqslant i \leqslant m)$ 对应的有一个最终执行时间 $t_i(1 \leqslant i \leqslant m)$ 和截止时间 $TD_i(1 \leqslant i \leqslant m)$,则执行时间 $t = \{t_1, t_2, \cdots, t_m\}$,截止时间 $TD = \{TD_1, TD_2, \cdots, TD_m\}$,其中 $t_i \leqslant TD_i$,为了建模方便,用 $TD_i - t_i$ 表示包括任务 s_i 等待其他任务的时间在内的,所有处理器 $C_j(1 \leqslant j \leqslant p)$ 的空闲时间.

用 E_{ij} 为任务 s_i 在处理器 C_j 上执行所消耗的能量,$P_{\mathrm{dynamic\_}j}$,$P_{\mathrm{static\_}j}$,$P_{\mathrm{short\_}j}$ 分别表示任务 s_i 在处理器 C_j 上执行的动态功耗、静态功耗和短路功耗 $E_{ij} = P_{\mathrm{dynamic\_}j}t_i + P_{\mathrm{static\_}j}(TD_i - t_i) + P_{\mathrm{short\_}j}t_i$,则系统中任务的能耗用矩阵 $\boldsymbol{E}$ 表示为

$$\boldsymbol{E} = \begin{pmatrix} E_{11} & E_{12} & \cdots & E_{1p} \\ E_{21} & E_{22} & \cdots & E_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ E_{m1} & E_{m2} & \cdots & E_{mp} \end{pmatrix}.$$

目标函数 $f(x_1, x_2, \cdots, x_m) = \min \sum_{i=1}^{m} E_{ix_i}$,$x_i \in \{1,2,\cdots,p\}$. 从目标函数可以看出,每个任务能耗越小越好,故采用每个任务的能耗和的倒数作为适应度函数 F 为 $F(x_1, x_2, \cdots, x_m) = 1/f(x_1, x_2, \cdots, x_m) = 1/\min\left(\sum_{i=1}^{m} E_{ix_i}\right)$,$x_i \in \{1,2,\cdots,p\}$.

通过粒子迭代后可得到最优解$\{E_{1x_1},E_{2x_2},\cdots,E_{mx_m}\}$即对应一个耗能最少的调度方案.

1.4　算法实现步骤

(i) 根据目标函数的适应度,设置算法终止条件,同时设置 PSO 算法的 ω、c_1 和 c_2 参数.

(ii) 通过混沌映射对粒子的速度 $v_i(k+1)$ 及位置 $x_i(k+1)$ 进行初始化.

(iii) 计算粒子适应度值,找出最优值,与此前最优值比较,确定粒子是否陷入局部最优.

(iv) 若陷入局部最优,则生成混沌序列 $u_0=\{u_{01},u_{02},\cdots,u_{0D}\}$,替代陷入局部最优的粒子.

(v) 局部极值与全局极值进行比较,优于则替换全局极值.

(vi) 更新粒子的速度 $v_i(k+1)$ 及位置 $x_i(k+1)$.

(vii) 判断是否符合终止条件,符合则停止搜索,否则返回(iii).

2　实验及结果分析

为了验证节能调度算法的性能,基于 TGFF 建立任务实验,TGFF 是软硬件协同设计领域常用的实验数据生成工具,可以生成实验中所需要的任务实验数据. 在实验中处理器参照文献[18]采用 Intel PXA255 的功耗模型,相关参数如表 1 所示,多个处理器通过分布式内存进行通信.

表 1　Intel PXA255 功耗模型

频率 /MHz	电压 /V	功耗 /mW
13(空闲)	0.85	44.2
104	0.90	115.0
208	1.15	279.0
312	1.25	390.0
416	1.35	570.0
520	1.45	747.0
624	1.55	925.0

在上述功耗模型下,通过 TGFF 随机产生 5 个任务集,分别命名为任务集 $U_1 \sim U_5$,如表 2 所示. 其结点个数在 20 至 80 之间变动,而通讯边数在 30 至 120 之间变动,任务周期数在 500 至 2 700 之间,试验中选 $c_1=c_2=2$,最大迭代次数 300 次. 实验通过将本文算法与 Moser 提出的 LSA 算法、Abdeddaïm 提出的 ASAP 算法和 Pillai 提出的 LAEDF 算法等 3 个基于能耗考虑的常用调度算法进行能耗比较,对 5 个任务集的实验比较结果如图 1 和图 2 所示.

表 2　任务集特征表

任务集名称	节点数	边数	周期数
U_1	21	38	571
U_2	26	44	895
U_3	35	51	1 321
U_4	47	67	1 733
U_5	56	84	2 125

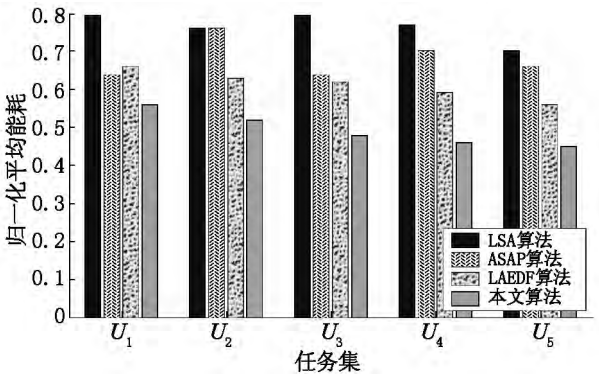

图 1　2 核 MPSoC 情况下的能耗对比

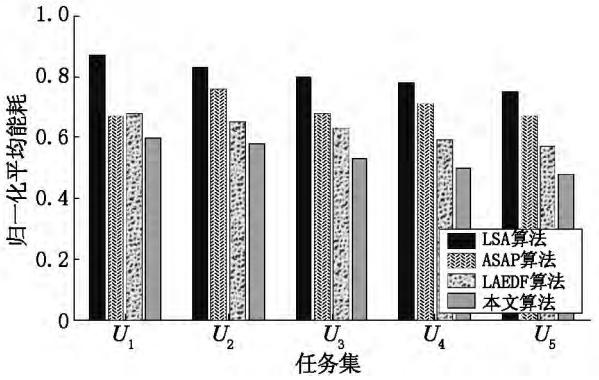

图 2　4 核 MPSoC 情况下的能耗对比

为了便于比较,将任务集的能耗归一化处理,图 1 是比较了 2 核 MPSoC 在执行任务集 $U_1 \sim U_5$ 时的能耗,图 2 是比较了 4 核 MPSoC 在执行任务集 $U_1 \sim U_5$ 时的能耗. 从图 1 和图 2 可以看出本文提出的节能调度算法的节能效果要优于其他 3 种常用基于能耗考虑的调度算法. 这是由于采用了混沌搜索优化的离散粒子群算法,使得任务集中的任务能基于能耗合理的得到调度. 并且还可以从实验中得出,任务集越大相对其他 3 种算法的调度节能效果越好,即图 1 和图 2 中的归一化后的本文算法和其他 3 种算法的能耗比例越来越小.

MPSoC 调度算法的任务截止期错失率是衡量算法性能最重要的指标之一,在上面 2 核和 4 核

MPSoC 运行任务集 $U_1 \sim U_5$ 时的能耗比较的基础上,下面通过比较随机的任务集,在本文算法和其他 3 种算法上运行的截止期错失率,来检验本文算法的性能. 其中任务集中任务的执行时间在0.05 ms ~ 0.50 ms 之间随机选择,实验结果如图 3 和图 4 所示.

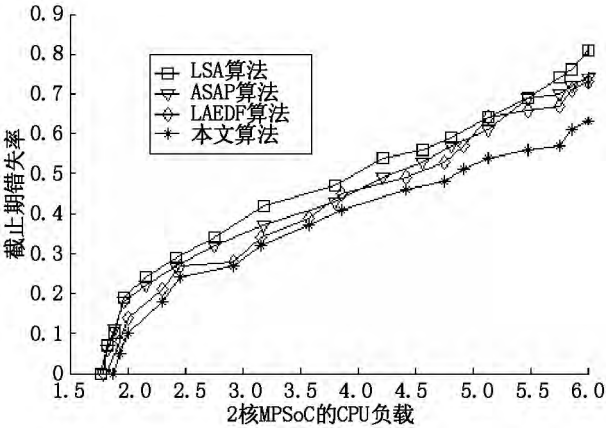

图 3 2 核 MPSoC 情况下的截止期错失率对比

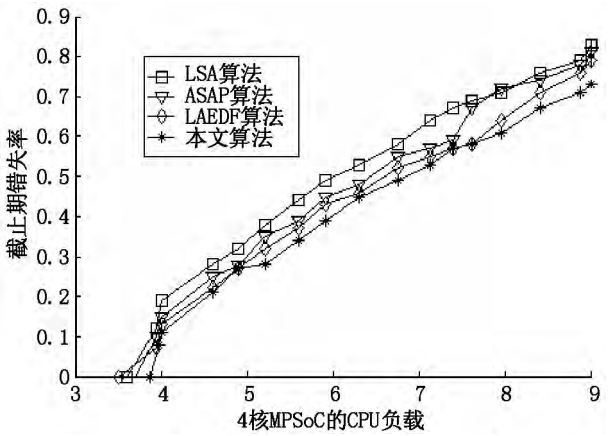

图 4 4 核 MPSoC 情况下的截止期错失率对比

从实验结果可以看出,2 核和 4 核 MPSoC 在运行任务集时,随着 CPU 负载的增加,其他 3 种调度算法和本文算法的任务截止期错失率都在增加,但本文调度算法的任务截止期错失率总体趋势上都没有超过其他 3 种算法,总体上要比其他 3 种算法截止期错失率低. 这就保证了该调度算法,在实现节能调度的前提下,没有降低算法的基本性能.

3 结论

通过局部早熟问题优化的离散粒子群算法来实现多核 MPSoC 任务的节能调度,实验结果表明调度算法的能耗比常用的基于能耗的调度算法要低,起到了一定的节能效果,并且任务集中的任务数越多,节能效果越好. 同时,本文算法在保证节能调度的前提下,也保证了算法的整体性能,如调度算法的任务截止期错失率,相对于常用的基于能耗的调度算法,并没有升高. 该调度算法特别适合于系统能量有限,任务数量多,且对低能耗要求较高的 MPSoC 嵌入式系统中.

4 参考文献

[1] 叶常华,左朝树. 基于多核处理器的节能任务调度方法[J]. 中国电子科学研究院学报,2012,7(2):204-207.

[2] Mei Jing,Li Kenli,Hu Jingtong,et al. Energy-aware preemptive scheduling algorithm for sporadic tasks on DVS platform[J]. Microprocessors and Microsystems,2013,37(1):99-112.

[3] 葛永琪,董云卫,张健,等. 一种能量收集嵌入式系统自适应调度算法[J]. 软件学报,2015,26(4):819-834.

[4] Huang Kai,Wolfgang Haid,Iuliana Bacivarov,et al. Embedding formal performance analysis into the design cycle of MPSoCs for real-time streaming applications[J]. ACM Transactions on Embedded Computing Systems,2012(1):135-151.

[5] Tavakkoli-Moghaddam R,Azarkish M,Sadeghnejad Barkousaraie A. A new hybrid multi-objective pareto archive PSO algorithm for a bi-objective job shop scheduling problem[J]. Expert Systems With Applications,2011,38(9):10812-10821.

[6] 成洪甲,杨雨,孙寅萍. 一种基于随机粒子群的变差函数优化方法[J]. 科学技术与工程,2012,12(31):8374-8378.

[7] Gao Xiang,Chen Yunji,Wang Huandong,et al. System architecture of Godson-3 multi-core processors[J]. Journal of Computer Science and Technology,2010,25(2):181-191.

[8] Zhang Guojun,Liu Min,Li Jian,et al. Multi-objective optimization for surface grinding process using a hybrid particle swarm optimization algorithm[J]. The International Journal of Advanced Manufacturing Technology,2014,71(9/12):1861-1872.

[9] Fariborz Jolai,Reza Tavakkoli-Moghaddam,Mohammad Taghipour. A multi-objective particle swarm optimisation algorithm for unequal sized dynamic facility layout problem with pickup/drop-off locations[J]. International Journal of Production Research,2012,50(15):4279-4293.

[10] Liao Chingjong,Evi Tjandradjaja,Chung Tsui Ping. An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem[J]. Applied Soft Computing Journal,2012,12(6):1755-

1764.

[11] Li Jia, Cheng Dashuai, Chiu Minsen. Pareto-optimal solutions based multi-objective particle swarm optimization control for batch processes [J]. Neural Computing and Applications, 2012, 21(6): 1107-1116.

[12] Hasan Hosseini-Nasab, Leila Emami. A hybrid particle swarm optimisation for dynamic facility layout problem [J]. International Journal of Production Research, 2013 (14): 4325-4335.

[13] Shieh Wann Yun, Pong Chin Ching. Energy and transition-aware runtime task scheduling for multicore processors [J]. Journal of Parallel and Distributed Computing, 2013, 73(9): 1225-1238.

[14] Boris Shnits. Multi-criteria optimisation-based dynamic scheduling for controlling FMS [J]. International Journal of Production Research, 2012, 50(21): 1-11.

[15] 赵振江. 基于量子粒子群优化算法的 PID 参数控制 [J]. 科学技术与工程, 2012, 12(22): 5490-5492.

[16] Krishnamoorthy M, Ernst A T, Baatar D. Algorithms for large scale shift minimisation personnel task scheduling problems [J]. European Journal of Operational Research, 2011, 219(1): 34-48.

[17] Huang Kai, Santinelli L, Chen Jianjia, et al. Applying real-time interface and calculus for dynamic power management in hard real-time systems [J]. Real Time Systems, 2011, 47(2): 163-193.

[18] Abusayeed Saifullah, Li Jing, Kunal Agrawal, et al. Multi-core real-time scheduling for generalized parallel task models [J]. Real Time Systems, 2013, 49(4): 404-435.

MPSoC Energy Saving Scheduling Algorithm Based on Discrete Particle Swarm Optimization

YU Guolong, CUI Zhongwei, ZUO Yu

(School of Mathematics and Computer Science, Guizhou Province University Industrial Networking Engineering Technology Research Center, Guizhou Normal College, Guiyang Guizhou 550018, China)

Abstract: In order to reduce the energy consumption of the MPSoC in the application, a energy saving task scheduling algorithm based on the optimization of discrete particle swarm optimization is proposed in MPSoC. The algorithm generates the initial population by the proportional selection operator, and takes the energy consumption of different cores on the MPSoC as the solution space, search for the lowest energy consumption scheduling scheme in the whole solution space and optimize local premature problem of the particle swarm optimization algorithm. Through these improve the performance of the algorithm. The simulation experiments show that the energy of saving scheduling algorithm based on the optimized discrete particle swarm optimization algorithm is lower than that of the common three algorithms, and the deadline miss rate of the algorithm does not increase, that ensure the overall performance of the algorithm.

Key words: particle swarm optimization; MPSoC; energy saving; scheduling algorithm

(责任编辑: 冉小晓)