第 37 卷 第 2 期
Vol.37 No.2

计 算 机 工 程
Computer Engineering

2011 年 1 月
January 2011

• 开发研究与设计技术 • 文章编号：1000—3428(2011)02—0242—03 文献标识码：A 中图分类号：TP316

一种面向部分可重构系统的实时调度算法

殷进勇，顾国昌，吴艳霞

(哈尔滨工程大学计算机科学与技术学院，哈尔滨 150001)

摘 要：针对当前可重构资源模型难以实现或资源利用率低等不足，提出一种新的资源模型。基于此模型，设计一种能够调度周期和非周期任务的混合实时任务调度算法。把周期任务分成若干组，在 FPGA 上为每组任务预留一个槽。当有非周期任务到达时，预先调度当前忙碌期内的所有周期任务，在保证当前忙碌期内周期任务满足截止期限且不影响下一个忙碌期内周期任务执行的情况下，把非周期任务调度到某个槽内执行。实验结果表明，该算法能够充分利用可重构资源，满足所有接收任务的截止期限。

关键词：操作系统；实时任务调度；硬件任务；资源模型

Real-time Schedule Algorithm for Partial Reconfigurable System

YIN Jin-yong, GU Guo-chang, WU Yan-xia

(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

【Abstract】 Because the proposed reconfigurable resource models are difficult to be implemented or can not make full use of the resource, a new resource model is proposed. Based on the model, a hybrid real-time schedule algorithm is proposed to schedule periodic and aperiodic tasks. The algorithm partitions periodic tasks into several groups and reserves one slot on FPGA for each group. When an aperiodic task arrives, the algorithm pre-schedules all the periodic tasks' instances in current busy period and schedules the aperiodic task guaranteeing the deadlines of periodic tasks in current busy period and without influences on the tasks' execution in the next busy period. Experimental results demonstrate that the algorithm can make full use of reconfigurable resource and meet all accepted tasks' deadlines.

【Key words】 operating system; real-time task schedule; hardware task; resource model

DOI: 10.3969/j.issn.1000-3428.2011.02.085

1 概述

在嵌入式系统的设计中，计算任务可以通过 ASIC 等硬件来实现，也可以利用微处理器通过软件实现。可重构计算填补了软、硬件之间的鸿沟，它不仅保持了硬件的高性能，而且具有接近于软件的灵活性[1]。

可重构资源模型分为一维模型和二维模型，由于二维模型的资源利用率高于一维模型，因此现有的文献多数基于二维模型[2-4]。根据任务的放置方式，二维模型又可分为任意型和分割型。在任意型中，只要 2 个任务不重叠，则可放置到 FPGA 的任意位置。这种放置方式仅考虑了任务所占用的计算资源，而没有考虑连线资源，如与芯片管脚的连线，因此难以实现，往往只是理论上的探讨。针对此问题，文献[1]提出了分割型资源模型，可重构资源被固定地分割成一系列的槽，每个槽是一个最小的资源分配单位。由于在同一时刻一个槽内只能运行一个任务，因此对于面积较小的任务来说，资源利用率较低。

本文提出了一种新的资源模型，称为约束型模型。在这种模型中，根据所调度的任务集，具体地确定每个槽的大小以及槽的个数，合理地布局每个槽的位置。基于此模型，提出了一种调度周期和非周期混合实时任务的算法，并解决了硬件任务在加载时的重定位问题。

2 任务和系统模型

系统调度的任务包括一组周期任务 $T=\{T_1,T_2,\cdots,T_n\}$和随机到达系统的非周期任务 $J=\{J_1,J_2,\cdots J_n\}$，每个任务 $T_i(J_i)$用一个 6 元组$(A_i,C_i,D_i,P_i,W_i,H_i)$表示。其中，$A_i$、$C_i$和 D_i分别表示任务的到达时间、执行时间、相对截止期限；W_i和 H_i分别表示任务的宽度和高度，即在 FPGA 上所占用的列数和行数；P_i表示任务的周期，非周期任务的周期 $P_i=\infty$。实时任务在每个周期内的一次执行称为一个实例，任务 T_i在第 $j(j\geq1)$个周期内的实例用 $T_{ij}(J_{ij})=(R_{ij},C_{ij},d_{ij},W_{ij},H_{ij})$表示。其中，$R_{ij}=A_i+(j-1)\times P_i$表示 $T_{ij}(J_{ij})$的释放时间；$d_{ij}=R_{ij}+D_i$表示 $T_{ij}(J_{ij})$的绝对截止期限；$C_{ij}=C_i$、$W_{ij}=W_i$、$H_{ij}=H_i$与 C_i、W_i和 H_i的意义相同。在下文中用 τ 表示周期或非周期任务的实例集合，在不混淆的情况下，任务实例有时也称为任务。

硬件平台为一个部分可重构 FPGA，如图 1 所示，包含 CPU、系统总线、总线接口等系统电路以及为硬件任务预留的槽 *Slot*；硬件任务动态地加载到槽内，通过接口连接到系统总线上。

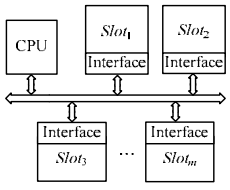


图 1 硬件平台架构

FPGA 由 $N=W\times H$ 个可重构单元构成，形成一个 W 列 H 行的二维矩形阵列，器件中的每个可重构单元都是相同的。槽集 $Slot=\{Slot_1,Slot_2,\cdots,Slot_m\}$表示在 FPGA 上为实时任务预

作者简介：殷进勇(1978－)，男，博士研究生，主研方向：实时系统，任务调度，可重构计算；顾国昌，教授、博士、博士生导师；吴艳霞，博士

收稿日期：2010-07-08 **E-mail：**yinjinyong@yahoo.com

留的空闲区域，每个 $Slot_i$ 可用(W_i,H_i)表示，W_i 和 H_i 分别表示空闲区域中可重构单元的列数和行数。

3　实时任务调度算法

定义 1 一个调度后的任务 τ_i，如果已开始运行但还没有结束，则称 τ_i 为运行任务；如果尚未开始运行，则称 τ_i 为预约任务。运行任务和预约任务统称已调度任务。

定义 2 用 ER_i 表示任务 τ_i 的最早可能执行时间。$EF_i=ER_i+C_i$ 表示任务 τ_i 的最早可能完成时间。

定义 3 如果任务 τ_i 在 t_i 时刻开始执行不会导致当前忙碌期内的已调度任务错失截止期限且不影响下一个忙碌期内任务的执行，那么称 $LR_i=\max\{t_i\}$ 为 τ_i 的最迟允许执行时间。$LF_i=LR_i+C_i$ 表示任务 τ_i 的最迟允许完成时间。

在某个槽内，假设当前忙碌期内的已调度任务 $\tau=(\tau_1,\tau_2,\cdots,\tau_n)$，按已调度的顺序排列。任务 τ_i 的最早可能执行时间和最迟允许执行时间计算如下：

$$ER_i=\begin{cases}R_i & i=1\\ EF_{i-1} & 1<i\le n\end{cases}$$
$$LR_i=\begin{cases}\min\{d_i,\ LR_{i+1}\}-C_i & 1<i\le n\\ \min\{d_i,\ NBS\}-C_i & i=n\end{cases}\quad(1)$$

其中，NBS 表示下一个忙碌期的开始时刻。

3.1　周期任务可调度性判定

把周期任务集 $T=\{T_1,T_2,\cdots,T_n\}$分成 m 组，每组对应一个槽。在一个槽上调度硬件实时任务与在单处理机上非抢占调度实时任务类似，可采用单处理机的任务可调度性判定方法判定一个槽上的实时任务是否可调度。文献[5]给出了周期任务在单处理机上被非抢占 EDF 算法调度的充分条件，采用本文的符号描述如下，即定理 1。

定理 1 周期任务集 $T=\{T_1,T_2,\cdots,T_n\}$按任务周期不降顺序排列，如果满足以下条件，那么任务集 T 可被非抢占 EDF 算法在槽 $Slot_k$ 上调度。

(1) $\forall\ T_i\in T$：$T_i.W_i\le Slot_k.W_k\wedge T_i.H_i\le Slot_k.H_k$；

(2) $\sum_{i=1}^{n}C_i/P_i\le 1$；

(3) $\forall\ i,1\le i\le n$；$\forall\ L,P_1\le L\le P_i$：$L\ge C_i+\sum_{j=1}^{i-1}\left\lfloor\frac{L-1}{P_j}\right\rfloor C_j$。

条件(1)表示任务 T_i 可放置到槽 $Slot_k$ 内；条件(2)保证槽 $Slot_k$ 在时间上的负载不能超过 1；条件(3)是对非抢占调度的一个约束，保证任务 T_i 可在截止期限内完成。

把一组槽放置到 FPGA 上是一个典型的正交切割下料问题。判定算法 PSchTest 首先确定周期任务集 $T=\{T_1,T_2,\cdots,T_n\}$ 所需要的槽集 $Slot$ 并把任务分配到每个槽上，然后采用 BF[6] 正交切割下料算法判定槽集 $Slot$ 能否放置到 FPGA 上，算法 PSchTest 描述如下：

(1)周期任务集 $T=\{T_1,T_2,\cdots,T_n\}$按任务的宽度从小到大，宽度相等的任务按高度从小到大顺序排序。

(2)令槽集 $Slot=\varphi$，已分配任务集 $SchTsk=\varphi$，$i=1$。

(3)如果 $T=\varphi$，转到(6)。

(4)如果 $SchTsk\cup\{T_i\}$满足定理 1 中的条件(2)与条件(3)，那么 $SchTsk=SchTsk\cup\{T_i\}$, $T=T-\{T_i\}$, $i=i+1$，转到(3)。

(5)创建槽 $Slot_k$，使得 $Slot_k.W_k=\max\{T_j.W_j\mid T_j\in SchTsk\}$, $Slot_k.H_k=\max\{T_j.H_j\mid T_j\in SchTsk\}$, $Slot=Slot\cup\{Slot_k\}$, $SchTsk=\varphi$，转到(3)。

(6)根据 BF 算法，如果槽集 $Slot$ 能放置到 FPGA 上，则任务集 T 可调度；否则不可调度，算法结束。

虽然 PSchTest 算法是一个伪多项式算法，但它仅在系统运行前执行一次，对系统的性能影响不大。

3.2　非周期任务可调度性判定

算法预先调度当前忙碌期内的所有周期任务，在当前忙碌期内判定非周期任务的可调度性，在下一个忙碌期开始前执行完所有已调度的周期和非周期任务[7]。

定理 2 当非周期任务 J_k 释放时，假设在槽 $Slot_j$ 上当前忙碌期内的已调度任务 $\tau=(\tau_1,\tau_2,\cdots,\tau_n)$，按已调度的顺序排列。如果满足以下条件，那么任务 J_k 可被非抢占 EDF 算法在 $Slot_j$ 上调度且不影响下一个忙碌期内任务的执行。

(1)$\tau=\varphi$：$\min\{NBS,d_k\}-R_k\ge C_k$；

(2)$\tau\ne\varphi$：$\exists\ 1<i\le n$, s.t. $\min\{LR_i,d_k\}-\max\{ER_i,R_k\}\ge C_k\vee\min\{NBS,d_k\}-\max\{EF_n,R_k\}\ge C_k$。

其中，NBS 表示下一个忙碌期的开始时刻。

证明：

如果满足条件(1)，那么任务 J_k 可在 $\min\{NBS, d_k\}$之前执行完毕，所以任务 τ_k 可被非抢占 EDF 算法在 $Slot_j$ 上调度且不影响下一个忙碌期内任务的执行。

如果满足条件(2)，那么在任务 τ_i 之前或 τ_n 之后存在足够的空闲时间使得任务 J_k 可被非抢占 EDF 算法在 $Slot_j$ 上调度且不影响下一个忙碌期内任务的执行。

证毕。

3.3　算法描述

在线调度算法 Schedule 首先在每个槽上调度当前忙碌期内的周期任务，当非周期任务到达时，算法按采用 First-fit 策略调度到第 1 个能够接收此任务的槽上。假设周期任务 $T=\{T_1,T_2,\cdots,T_n\}$所需要的槽集为 $Slot=\{Slot_1,Slot_2,\cdots,Slot_m\}$。Schedule 算法描述如下：

(1) $\forall\ Slot_i\in Slot$，如果 $C(t)\ge NBS\wedge SchFlag$=False，按以下步骤调度周期任务：

1)$Load=0$；

2)令 T_{ij} 为下一个释放时间最早的周期任务；

3)如果 $R_{ij}>C(t)+Load$，则转到 7)；

// T_{ij} 为下一个忙碌期的任务

4)$Load=Load+C_{ij}$；

5)令 $T_{xy}=\{T_{pq}\mid T_{pq}\in RdyQue_i\wedge ER_{pq}\le R_{ij}\le EF_{pq}\}$，按绝对截止期限升序把 T_{ij} 插入到以 T_{xy} 为头的队列中；

6)按式(1)计算 $RdyQue_i$ 中任务的最早可能执行时间，转到 2)；

7)按式(1)计算 $RdyQue_i$ 中任务的最迟允许结束时间；

8) $NBS=R_{ij}$，$SchFlag$=True。

(2)设 J_{ij} 为新到达非周期任务，根据定理 2，如果$\exists\ Slot_i\in Slot$ 可接收 J_{ij}，则按以下步骤调度任务 J_{ij}，否则转到(3)。

1)令 $\tau_{xy}=\{\tau_{pq}\mid\tau_{pq}\in RdyQue_i\wedge ER_{pq}\le R_{ij}\}$，按绝对截止期限升序把 J_{ij} 插入到以 τ_{xy} 为头的队列中；

2)按式(1)计算 $RdyQue_i$ 中任务的最早可能执行时间和最迟允许结束时间。

(3)算法结束。

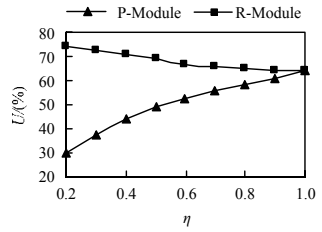
在 Schedule 算法中，$C(t)$表示当前时间；NBS 表示下一个忙碌期开始时刻；如果当前忙碌期内的周期任务已调度则 $SchFlag$=True，否则 $SchFlag$=False；$Load$ 表示已调度任务的执行时间总和；$RdyQue_i$ 表示槽 $Slot_i$ 上的已调度任务队列。

假设已调度任务个数为 N，那么调度周期任务时步骤 5)的时间复杂性为 $O(N^2)$，步骤 6)和步骤 7)的时间复杂性为

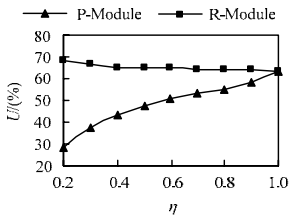
$O(N)$，其余步骤的时间复杂性为 $O(1)$。由于每个槽需要调度一次，共调度 m 次，因此调度周期任务的复杂性为 $O(mN^2)$。同理可分析调度非周期任务时间复杂性为 $O(mN^2)$，所以算法 Schedule 的时间复杂性为 $O(mN^2)$。

4 实验结果

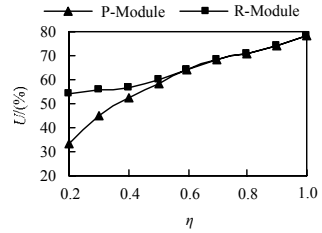
与软件任务利用率不同，硬件任务利用率包括时间利用率和空间利用率 2 个部分，定义综合利用率 $U=\frac{C_i\times W_i\times H_i}{P_i\times W\times H}$，其中，$W$ 和 H 表示 FPGA 的宽度和高度，对于非周期任务，P_i 等于模拟时间长度。可重构器件按照 Xilinx Virtex-5LX330 的规模定义，具有 240×108 个可重构单元。周期任务的周期在[100,500]上均匀分布，执行时间在[10,50]上均匀分布，分别在分割模型(P-Module)和约束模型(R-Module)上测试了算法的综合利用率，结果如图 2 所示。其中，$Cn(n=30,40,50)$ 表示任务的宽度和高度的最大值为 n，最大值与最小值的比为 η，任务的宽度和高度在$[\eta n, n]$均匀分布。从图 2 中可看出，在 $\eta=1$ 即所有任务的面积均相等时，两者的测试结果一致，在其他情况下分割模型的资源利用率均低于约束模型的资源利用率。



(a)C30



(b)C40



(c)C50

图 2 2 种模型资源利用率比较

在非周期任务的到达时间为强度 λ 的泊松流，相对截止期限和执行时间分别在[100,500]和[10,50]上均匀分布条件下测试了非周期任务的接收率 AR 以及相应的综合利用率 U。实验首先调度一组周期任务，使得每个槽上的时间利用率约为 50%，周期任务的综合利用率为 32.14%。测试结果如图 3 所示，结果表明在非周期任务综合利用率小于等于 28.5%的情况下，任务接收率在 95%以上。

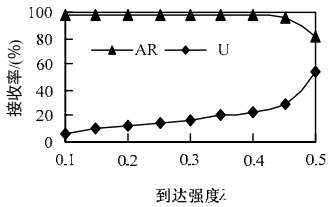


图 3 非周期任务接收率

5 结束语

可重构资源二维任意型模型难以实现，而二维分割型模型资源利用率较低，因此提出了约束资源模型。在约束资源模型上设计了一种调度周期和非周期混合实时任务调度算法。实验结果表明，该模型兼有任意型的高资源利用率和分割型的可实现性特点。

参考文献

[1] Compton K. Reconfigurable Computing: A Survey of Systems and Software[J]. ACM Computing Surveys, 2002, 34(2): 171-210.

[2] Steiger C, Walder H, Platzner M. Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-time Tasks[J]. IEEE Trans. on Computers, 2004, 53(11): 1393-1407.

[3] Pellizzoni R, Caccamo M. Real-time Management of Hardware and Software Tasks for FPGA-based Embedded Systems[J]. IEEE Trans. on Computers, 2007, 56(12): 1666-1680.

[4] 杨林楠, 李红刚, 张丽莲, 等. 基于 FPGA 的高速多路数据采集系统的设计[J]. 计算机工程, 2007, 33(7): 246-248.

[5] Yang Linnan, Li Honggang, Zhang Lilian, et al. Design of High Speed Multichannel Data Gathering System Based on FPGA[J]. Computer Engineering, 2007, 33(7): 246-248.

[6] Jeffay K, Stanat D F, Martel C U. On Non-preemptive Scheduling of Periodic and Sporadic Tasks[C]//Proc. of the 12th Real-time Systems Symposium. San Antonio, USA: [s. n.], 1991: 129-139.

[7] Burke E K, Kendall G, Whitwell G. A New Placement Heuristic for the Orthogonal Stock-cutting Problem[J]. Operations Research, 2004, 52(4): 655-671.

编辑 顾逸斐

(上接第 239 页)

参考文献

[1] Uzsoy R, Lee C Y, Martin-Vega L A. A Review of Production Planning and Scheduling Models in the Semiconductor Industry, Part I: System Characteristics, Performance Evaluation and Production Planning[J]. IIE Transactions, 1992, 24(4): 47-60.

[2] 李 飞, 伍乃骐. 基于 eM-Plant 的虚拟晶圆制造自动组合装置[J]. 计算机工程, 2009, 35(10): 232-234.

[3] Agrawal, G K, Heragu S S. A Survey of Automated Material Handling Systems in 300-mm Semiconductor Fabs[J]. IEEE Trans. on Semiconductor Manufacturing, 2006, 19(1): 112-120.

[4] Montoya-Torres J R. A Literature Survey on the Design Approaches and Operational Issues of Automated Wafer-transport Systems for Wafer Fabs[J]. Production Planning & Control, 2006, 17(7): 648-663.

[5] Bilge B, Esenduran G, Varol N, et al. Multi-attribute Responsive Dispatching Strategies for Automated Guided Vehicles[J]. International Journal of Production Economics, 2006, 100(1): 65- 75.

[6] Edmonds J, Karp R M. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems[J]. Journal of the Association for Computing Machinery, 1972, 19(2): 248-264.

编辑 金胡考