

文章编号:1001-9081(2010)11-2870-03

基于放置代价的可重构系统任务统一调度算法

蔡富强¹, 郭兵¹, 沈艳², 王继禾¹, 伍元胜¹

(1. 四川大学 计算机学院, 成都 610065; 2. 成都信息工程学院 控制工程学院, 成都 610225)

(caiqiangscu@163.com)

摘要:高效的任务调度算法对可重构系统的性能有极大的影响。针对目前可重构系统任务在线调度算法的不足,提出了一种基于放置代价的调度算法。该算法考虑了3种代价,分别为:硬件任务在FPGA上的执行时间、占用的FPGA面积以及FPGA的碎片情况,并且也考虑了软硬件任务的统一调度。在调度过程中,当代价超过设定的阈值时,就拒绝其在FPGA上运行,并由CPU执行其软实现。通过合理地拒绝一些代价较大的任务,能够从整体上提高任务调度成功率。实验表明,同已有算法相比,该算法能够获得更高的任务截止保证率。

关键词:可重构系统;调度算法;放置算法;放置代价;硬件任务

中图分类号: TP301.6; TP316 **文献标志码:** A

Unified task scheduling algorithm of reconfigurable system based on placement cost

CAI Fu-qiang¹, GUO Bing¹, SHEN Yan², WANG Ji-he¹, WU Yuan-sheng¹

(1. College of Computer Science, Sichuan University, Chengdu Sichuan 610065, China;

2. School of Control Engineering, Chengdu University of Information Technology, Chengdu Sichuan 610225, China)

Abstract: High efficient task scheduling algorithms can greatly influence the performance of reconfigurable systems. This paper analyzed the disadvantages of some current on-line reconfigurable system task scheduling algorithms, and proposed a new scheduling algorithm based on placement cost. This scheduling algorithm considered three types of cost, including hardware task execution time in FPGA, occupied FPGA area and fragmentation situation of FPGA, and also considered the unified scheduling of hardware and software tasks. When a hardware task was in scheduling, if the placement cost exceeded the setting threshold value, the task would be rejected to run in FPGA, but its software implementation could run in CPU. By reasonably rejecting some high cost tasks, this algorithm could achieve a higher overall successful scheduling rate. The simulation results show that this scheduling algorithm can achieve a higher deadline-guarantee ratio than others.

Key words: reconfigurable system; scheduling algorithm; placement algorithm; placement cost; hardware task

0 引言

随着大规模集成电路的不断发展,兼具软件灵活性和硬件性能的现场可编程门阵列(Field Programmable Gate Array, FPGA)变得越来越强大。在嵌入式系统的设计中,计算任务既可以采用软件实现,也可以采用基于FPGA的硬件实现。硬件实现高效,易于满足实时性的要求,但成本较高;软件实现方便灵活,成本低,但速度较慢。具体采用何种实现需要根据需求而定。

目前,可重构FPGA主要采用二维结构,二维结构中的各个子区域可以独立配置,而互不影响;同时,硬件任务要能够在其上运行,需要占用一定的面积;另外,由于配置一个硬件任务运行需要很大的代价,硬件任务运行结束后才释放占用的FPGA资源。因此,硬件任务要能运行就会涉及到任务的放置策略问题。这与二维网格多处理器系统中的处理器分配非常类似。经过多年的发展,已产生了多种任务放置算法。如BL^[1]、QA^[2]、RBL^[3]、Bazargan^[4]、OTF^[5]等。到达的硬件任务被排队,由放置算法选择合适的FPGA区域运行。

到操作系统层面,就需要考虑硬件任务的调度问题,甚至是软硬件任务的统一调度。目前,硬件任务调度算法^[6-8]一般都建立在已有的放置算法之上(采用预约策略,考虑硬件任务对时间的需求,将二维放置算法扩展到三维)。但这些算法都有一个缺点:没有考虑硬件任务的放置代价,没有考虑软硬件任务的统一调度。针对这个问题,在已有算法的基础上,本文提出一种基于放置代价的软硬件任务统一调度算法。

1 系统模型

可重构FPGA由一定数量的可重构单元(Reconfigurable Units, RCU)组成。二维可重构FPGA含有 $W \times H$ 个RCU, W 表示宽度, H 表示高度。硬件任务运行时占用FPGA的一个子区域。本文的模型包含主CPU和可重构FPGA,具体的系统调度模型如图1所示。

主CPU用于运行调度算法和管理可重构FPGA资源。为每个硬件任务实现一个等价的软件任务,当硬件任务放置被拒绝时,可以考虑在CPU上执行其等价的软件实现,让更多的任务满足截止期的要求。到达的任务放入到达队列,由

收稿日期:2010-04-26;修回日期:2010-06-23。

基金项目:国家863计划项目(2008AA01Z105);四川省杰出青年科技基金资助项目(2010JQ0011)。

作者简介:蔡富强(1986-),男,四川绵阳人,硕士研究生,主要研究方向:嵌入式实时系统;郭兵(1970-),男,山东泰安人,教授,博士,主要研究方向:RTOS、嵌入式软件、SoC芯片;沈艳(1973-),女,湖南永州人,副教授,博士,主要研究方向:智能化网络化测控技术、智能仪器;王继禾(1986-),男,陕西西安人,博士研究生,主要研究方向:嵌入式实时系统;伍元胜(1986-),男,四川广安人,博士研究生,主要研究方向:嵌入式实时系统。

调度器调度放置器为任务预约所需的 FPGA 资源(放置位置,启动时间)。当不能满足截止期要求或者放置代价过大时,该任务被拒绝,放入软运行队列中由 CPU 执行;否则放入预约队列,由加载器在其启动时间载入 FPGA 中运行,并放入硬运行队列(加载器还负责运行结束任务的撤销)。硬件任务不抢占执行,按预约的启动时间执行;软件任务按最早截止时间优先进行调度。

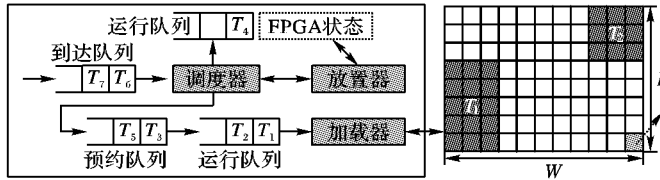


图1 可重构系统模型

定义 1 任务。一个任务可以定义为六元组: $T(w, h, a, rh, rs, d)$ 。其中: $w \times h$ 代表任务运行时占用的 FPGA 面积, a 表示任务的到达时间, rh 表示任务在 FPGA 上的运行时间, rs 表示任务在 CPU 上的运行时间, d 表示任务的截止时间。运行(预约)的硬件任务通过 $T(x_1, y_1, x_2, y_2, s, f)$ 表示, 其中 (x_1, y_1) 为任务放置基点, $x_2 = x_1 + w - 1, y_2 = y_1 + h - 1, s$ 为任务的启动时间, f 为任务的结束时间。

定义 2 任务截止保证率。满足截止期要求的任务数占总任务数的百分比。

定义 3 放置代价。以任务在 FPGA 上的运行时间、空间代价甚至是 FPGA 整体碎片程度作为衡量因素。放置代价较大的任务, 会极大地影响后续任务, 进而使任务截止保证率降低。

定义 4 任务 T 接受区域 A_T 。由于任务要占用一定的硬件面积, 使得 FPGA 的最右边和最上面的一些单元不能作为任务 T 的放置基点。 $A_T = (1, 1, W_A, H_A)$, 其中 $W_A = W - w + 1, H_A = H - h + 1$ 。图 2 左侧阴影部分为一 3×4 任务的接受区域。

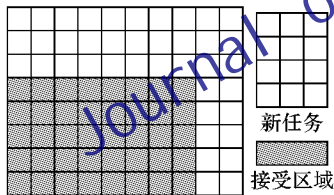


图2 任务接受区域

2 在线调度算法

2.1 相关工作

一般的放置算法均可以直接作为调度算法使用(有空闲区域就接受, 没有就拒绝), 但效果较差, 当前被拒绝的任务, 因未来某个运行任务的撤销而具有了空闲区域, 仍然有可能满足其截止期的要求。放置算法分为扫描类算法和空闲区管理算法。扫描类算法以 BL^[1]、QA^[2]、RBL^[3] 等为代表, 该类算法具有完全识别能力, 但早期的算法复杂度较高, 同 RCU 数量成正比, QA 算法比较好地解决了这一问题。空闲区管理算法以 Bazargan^[4]、OTF^[5] 等算法为代表, 这类算法采用一定的策略来管理 FPGA 空闲区域, 但要做到完全识别的代价很大。本文采用 QA 算法作为放置算法。

放置算法并不能高效地充当调度算法, 高效的调度算法需要考虑时间的流动。一种叫做预约的策略被提了出来, 该策略可描述如下: 在未来时间的流动中, 由于运行结束任务的撤销和预约任务的开始, FPGA 的使用情况将发生变化, 进而在某个时刻将有可能出现满足到达任务的放置区域, 该任务可以在该时刻运行。文献[6]提出一种紧凑预约(Compact

Reservation, CR)调度算法, 考虑所有的运行任务和预约任务, 计算到达任务的接受区域各 RCU 最早可用时间, 按照一定策略选择一个满足到达任务截止期要求的放置基点, 该算法能够在一定程度上提高任务截止期保证率。文献[7]提出的 Stuffing 调度算法通过模拟未来时刻任务的终止和启动来为到达任务预约放置区域, 它是一种比较通用的算法, 能够和现有的基于空闲区的放置算法融合。文献[8]是一种基于时间窗口的 Stuffing 调度算法, 使 Stuffing 算法可以同扫描类放置算法相结合。本文采用文献[8]中的算法作为任务的调度策略。

2.2 基于代价的调度算法

传统的操作系统为提高调度系统的吞吐量, 引入复杂的调度策略。目前针对硬件任务的调度主要还是采用先来先服务(First Come First Service, FCFS)的策略, 导致任务的截止期保证率还不高。本文在已有算法的基础上, 提出一种基于放置代价的软硬件任务统一调度算法。放置代价将从以下 3 个方面考虑: 1) 以硬件任务占用 FPGA 面积作为放置代价, 当硬件任务的面积超过一定阈值时, 就拒绝其在 FPGA 上运行, 将其软件实现放入软运行队列; 2) 以硬件任务在 FPGA 上的运行时间作为放置代价, 当硬件任务的运行时间超过一定阈值时, 就拒绝其在 FPGA 上运行, 将其软件实现放入软运行队列; 3) 以 FPGA 的碎片程度作为放置代价, 当因为任务的放置而使 FPGA 的碎片程度变化较大时, 就拒绝其在 FPGA 上运行, 将其软件实现放入软运行队列。算法调度任务运行的过程如下:

算法 1 schedule(T)。

```

1) if  $T$  is soft task then
2)   Add task to soft executing queue
3)   return
4)  $EL \leftarrow$  hard executing queue,  $RL \leftarrow$  hard reserving queue
5)  $ts \leftarrow a, tf \leftarrow ts + rh$ , accept  $\leftarrow$  false
6) while  $tf \leq d$  do
7)   for each task  $T_E(x_1, y_1, x_2, y_2, s, f)$  in  $EL$ 
8)     if  $f \leq ts$  remove  $T_E$  from  $EL$ 
9)   for each task  $T_R(x_1, y_1, x_2, y_2, s, f)$  in  $RL$ 
10)    if  $s < tf$  then
11)      remove  $T_R$  from  $RL$ 
12)    insert  $T_R$  into  $EL$ 
13)     $\langle x, y \rangle \leftarrow QA(EL, T)$ 
14)    accept  $\leftarrow$  PlaceCost( $x, y, T$ )
15)    if accept is true then
16)      AddReservation( $T, x, y, ts$ )
17)    return
18)     $ts \leftarrow$  next finishing time event from  $EL$ 
19)     $tf \leftarrow ts + rh$  //end while
20) if accept is false then
21)   Add task to soft executing queue

```

算法 1 中描述的算法以文献[8]中的调度算法为基础, 主循环模拟时间的流动, 采用高效的 QA 算法来为任务寻找放置基点(算法第 13 行), 以 FPGA 的碎片程度作为任务的放置代价(算法第 14 行), PlaceCost 函数以文献[9]中的方法计算 FPGA 的碎片程度, 当放置前后 FPGA 的碎片程度变化超过一定阈值时, 就返回 false。对于以硬件任务的占用面积或运行时间作为放置代价的实现更简单, 不需要在每次循环都计算代价, 代价的计算和任务拒绝与否在循环开始之前执行。

2.3 调度实例

假定 FPGA 含有 8×5 个可重构单元, 以任务占用的面积作为放置代价, 当 $w \times h > 28$ 时, 就拒绝任务。表 1 给出了一组任务实例。

按照本文的调度算法,任务 T_1 放置基点为(1,1),启动时间为1,结束时间为6;任务 T_2 放置基点为(5,1),启动时间为1,结束时间为11;任务 T_3 由于放置代价过大被拒绝,执行对应的软件任务,仍然满足截止期要求;任务 T_4 放置基点为(1,1),启动时间为6,结束时间为14。所有的任务均满足截止期的要求。而按照文献[6-8]的方法,没有考虑放置代价,对任务 T_3 将会被预约到11时刻开始执行(放置基点(1,1)),任务 T_4 由于不能满足截止期要求而被拒绝。

表1 任务实例

任务	w	h	a	rh	rs	d
T_1	4	3	1	5	10	20
T_2	2	3	1	10	25	30
T_3	6	5	2	10	25	30
T_4	3	3	3	8	24	20

3 模拟实验

在模拟实验中,采用 Xilinx 公司的芯片 vertex XCV1000 作为可重构 FPGA,该 FPGA 含有 96×64 个 RCU 单元。生成的模拟任务各参数随机分布,任务宽度取值范围为[20,90],高度取值范围为[10,60],任务在 FPGA 上执行时间范围[100,1000],软实现执行时间为对应硬实现的2~5倍,任务到达时间范围[1,100],任务的截止时间为: $a + rh + t, t \in [100,500]$ 。

3.1 阈值的选取

测试时模拟生成任务数量为1000,各代价的不同阈值均测试100次,结果取平均值。3种代价的阈值选取与任务截止保证率关系如图3(a)所示。对于时间和空间代价,横坐标取下面的值,图中给出的是比例值;对于碎片程度代价,横坐标取上面的值。从图中可以看出:阈值有最优值,时间代价最优值在 $0.8 \times \text{MAX}(rh)$ 附近,面积代价最优值在 $0.6 \times W \times H$ 附近,碎片程度代价最优值在2.0附近;低于最优值时,阈值的选取对任务截止保证率影响很大,高于最优值后阈值对任务截止保证率影响减小,趋近于没有考虑代价时的情形。下面进行的测试代价选最优值。

3.2 调度算法比较

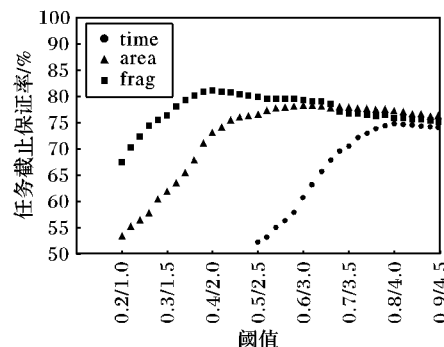
根据模拟生成任务的数量分为5个测试集: C_{600} 、 C_{800} 、 C_{1000} 、 C_{1200} 、 C_{1400} ,下标为任务数量。每个测试集测量100次,结果取平均值。参与比较的调度算法有:1)文献[8]中的算法(WS);2)以任务占用面积为代价的算法(EWS),拒绝阈值取最优;3)以任务在FPGA上运行时间为代价的算法(TWS),拒绝阈值取最优;4)以FPGA的碎片程度为代价的算法(FWS),拒绝阈值取最优。各算法的任务截止保证率如图3(b)所示。

可见,基于放置代价的调度算法能够获得更好的调度性能,这在任务较多时尤其明显。因为通过合理地拒绝一些代价较大的任务,使后续更多的任务获得执行的机会,进而提高任务截止保证率。而对于3种放置代价,基于FPGA碎片程度的效果最好,基于任务占用面积的次之,而基于任务执行时间的最差,稍微好于没有考虑放置代价的调度算法。

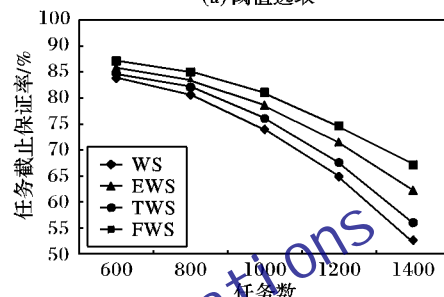
4 结语

在可重构系统中,好的放置算法和调度算法能极大地提高系统的性能。经实验证明,本文提出的基于代价的可重构系统软硬件任务统一调度算法能够获得更高的任务截止保证

率。从实验结果可以看出,3种放置代价的效果差别较大,因此,在未来的工作中,可以考虑把3种放置代价融合在一起,以进一步提高调度算法的效率。



(a) 阈值选取



(b) 调度算法比较

图3 任务截止保证率比较

参考文献:

- [1] SHARMA D D, PRADHAN D K. A fast and efficient strategy for submesh allocation in mesh-connected parallel computers [C]// Proceedings of the 5th IEEE Symposium on Parallel and Distributed Processing. Washington, DC: IEEE Computer Society, 1993: 682-689.
- [2] YOO S-M, YOUN H Y, SHIRAZI B. An efficient task allocation scheme for 2D mesh architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 8(9): 934-942.
- [3] CHIU C M, CHEN S K. An efficient submesh allocation scheme for two-dimensional meshes with little overhead[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(5): 471-486.
- [4] BAZARGAN K, KASTNER R, SARRAFZADEH M. Fast template placement for reconfigurable computing systems[J]. IEEE Design and Test of Computers, 2000, 17(1): 68-83.
- [5] WALDER H, STEIGER C, PLATZNER M. Fast online task placement on FPGAs: Free space partitioning and 2D-hashing [C]// Proceedings of 17th Parallel and Distributed Processing Symposium. Washington, DC: IEEE Computer Society, 2003: 178-185.
- [6] ZHOU XUEGANG, WANG YING, HUANG XINZHANG, et al. Fast on-line task placement and scheduling on reconfigurable devices [C]// PPL 2007: Proceedings of 2007 International Conference on Field Programmable Logic and Applications. Washington, DC: IEEE, 2007: 132-138.
- [7] STEIGER C, WALDER H, PLATZNER M. Online scheduling and placement of real-time tasks to partially reconfigurable devices [C]// Proceedings of the 24th IEEE International Real-Time Systems Symposium. Washington, DC: IEEE Computer Society, 2003: 224-235.
- [8] ZHOU XUEGANG, WANG YING, HUANG XINZHANG, et al. On-line scheduling of real-time tasks for reconfigurable computing system [C]// IEEE International Conference on Field Programmable Technology. Washington, DC: IEEE, 2006: 57-64.
- [9] AHMED A E, EL-BOGHADADI H M, SHAHEEN S I. Fragmentation aware placement in reconfigurable devices [C]// The 6th International Workshop on System-on-Chip for Real-Time Applications. Washington, DC: IEEE, 2006: 37-44.