基于预配置和配置重用的粗粒度动态可重构系统任务调度技术

戴紫彬 曲彤洲*
(解放军信息工程大学 郑州 450001)

摘 要：配置时间过长是制约可重构系统整体性能提升的重要因素，而合理的任务调度技术可有效降低系统配置时间。该文针对粗粒度动态可重构系统(CGDRS)和具有数据依赖关系的流应用，提出了一种3维任务调度模型。首先基于该模型，设计了一种基于预配置策略的任务调度算法(CPSA)；然后根据任务间的配置重用性，提出了间隔配置重用与连续配置重用策略，并据此对CPSA算法进行改进。实验结果证明，CPSA算法能够有效解决调度死锁问题、降低流应用执行时间并提高调度成功率。与其它调度算法相比，对流应用执行时间的平均优化比例达到6.13%～19.53%。

关键词：粗粒度动态可重构系统；流应用；预配置；配置重用
中图分类号：TP309.7; TN492 **文献标识码**：A **文章编号**：1009-5896(2019)06-1458-08
DOI: 10.11999/JEIT180831

Task Scheduling Technology for Coarse-grained Dynamic Reconfigurable System Based on Configuration Prefetching and Reuse

DAI Zibin QU Tongzhou
(*The PLA Information Engineering University*, *Zhengzhou* 450001, *China*)

Abstract: Long configuration time is a significant factor which restricts the performance improvement of the reconfigurable system, and a reasonable task scheduling technology can effectively reduce the system configuration time. A three-dimensional task scheduling model for Coarse-Grain Dynamic Reconfigurable System (CGDRS) and flow applications with data dependencies is proposed. Firstly, based on this model, a Configuration Prefetching Schedule Algorithm (CPSA) applying pre-configured strategy is designed. Then, the interval and continuous configuration reuse strategy are proposed according to the configuration reusability between tasks, and the CPSA algorithm is improved accordingly. The experimental results show this algorithm can avoid scheduling deadlock, reduce the execution time of flow applications and improve scheduling success rate. The optimization ratio of total execution time of flow applications achieves 6.13%～19.53% averagely compared with other scheduling algorithms.

Key words: Coarse-Grain Dynamic Reconfigurable System (CGDRS); Flow applications; Pre-configured; Configuration reuse

1 引言

作为一种典型的并行计算平台，可重构处理器满足了大数据时代日益提高的系统性能需求。该平台能够通过更新配置信息实现硬件资源的重用，从而执行相对于处理器硬件规模大得多的应用[1]，目前，制约可重构计算发展的重要因素之一是系统配置时间相对任务计算时间过长；因此如何降低配置时间成为提高可重构处理器系统整体性能的关键问题[2]。主流可重构处理器可分为现场可编程门阵列(Field Programmable Gate-Array, FPGA)与粗粒度可重构阵列(Coarse-Grained Reconfigurable Array, CGRA)两种[3]。FPGA在片上只存储当前任务的配置信息，新任务执行前必须更新配置信息。而CGRA的片上存储容量大，具有多级配置缓存，也被称为多重配置页面，多个页面中可存储多份配置信息。因此在CGRA上，通过快速切换配置页面即可执行新的任务。

针对可重构系统配置延迟过长的问题，文献[4]提出了一种局部动态重构技术。该技术可使处理器在更改目标区域配置信息的同时维持其他区域的正常工作；但它并未给出具体的任务调度方案。文献[2]针对动态部分可重构FPGA模型提出了一种预配置列表调度算法，算法通过定义任务的预配置优

收稿日期：2018-08-24；改回日期：2019-02-02；网络出版：2019-02-26
*通信作者： 曲彤洲 qutongzhou@outlook.com

先级提高了可重构系统的调度成功率和运行效率。文献[5]则提出了一种基于配置重用策略的列表调度算法，进一步缩短了任务执行时间。文献[6–8]中也提出了类似的调度策略，但以上方案只针对FPGA设计，并未考虑到CGRA架构上多页面配置储存的特殊性。文献[9]提出了一种基于预测机制的CGRA任务调度算法，精准预测了可重构资源不足情景下的任务预取顺序。文献[10]与文献[11]中针对CGRA架构模型提出了一种支持复杂控制流映射的最优化调度算法，能够计算出近似最优的任务调度时间表。但以上方案并不适用于存在数据依赖关系的流应用。

为降低 CPU-CGRA 架构的配置延时，本文通过对任务进行优先级排序以及开发“配置-计算”流水线，提出了一种基于预配置机制的粗粒度动态可重构系统(Coarse-Grain Dynamic Reconfigurable System, CGDRS)任务调度算法 (Configuration Prefetching Schedule Algorithm, CPSA)。在此基础上，提出了两种配置重用策略，从而进一步优化了CPSA算法。实验表明，本文技术可解决调度死锁问题，有效提高算法的调度成功率，并且显著降低流应用的执行时间。

2　模型建立

2.1　CGRA计算资源模型

本文采用的CGDRS精简结构如图1(a)所示，由主处理器、片外配置存储和CGRA共同组成[12]，CGRA内又包含可重构计算资源和接口单元。主处理器内运行任务调度算法，根据每个任务的不同参数，控制片外配置存储将任务计算所需的配置信息传输至CGRA，并给出任务执行过程中所需的控制信号；CGRA的接口单元接收配置信息并将其发往可重构计算资源，如图1(b)。可重构计算资源可抽象为由粗粒度运算单元(Processing Element, PE)组成的2维矩阵[13]，**PE**可根据自身配置在确定时刻执行确定任务；每个**PE**拥有n_p个独享的配置页面，所以CGRA的存储资源可看作一个3维空间[14]。因此，CGRA计算资源模型可抽象成一个数组(H, W, P)，其中H和W分别代表**PE**矩阵的长和宽，P代表阵列上所有配置页面构成的集合

$$P=\{p_1, p_2, \cdots, p_{n_p}\} \tag{1}$$

任务可表示为一个6元组$t(x, y, w, h, c, e)$，其中(x, y)代表任务t所映射矩形区域的左下角顶点坐标，w和h分别代表矩形的长和宽，c和e分别代表任务执行所需的配置时间和计算时间。在本文中，流应用特指具有数据依赖关系的n_t个任务构成的集合。那么包含n_t个任务的流应用A可表示为

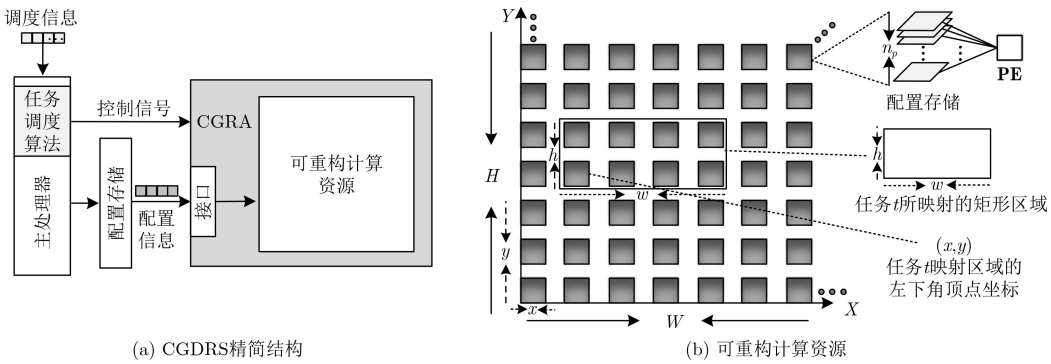
$$A=\{t_1, t_2, \cdots, t_{n_t}\} \tag{2}$$

建立如图1(b)所示的2维坐标系，x轴、y轴分别代表**PE**矩阵或配置矩阵的长、宽坐标。那么每个任务可映射在坐标系的确定2维区域内。为简化模型，规定任务的映射区域呈矩形。对于流应用A，通常采用有向无环图(Directed Acyclic Graph, DAG)描述任务之间的数据依赖关系[15]。将任一DAG记为集合$G=(V, E)$，节点$v_i \in V$代表流应用中的任务t_i; E代表G中所有的有向边集合，有向边$e(v_i, v_j)$表示任务t_j要在任务t_i计算完成后执行，并将任务t_i称为任务t_j的前驱任务，任务t_j称为任务t_i的后继任务。

2.2　CGDRS3维任务调度模型

CGRA具有多级配置页面，CGDRS上的任务调度问题分为配置与计算两个阶段。配置过程可视作为3维空间上的填充问题。由于配置信息传递速率受接口带宽的限制，任意时刻主处理器能够进行配置的任务数量存在上限。本文将每个**PE**的单个配置页面看作单位立方体，则CGRA的总配置存储空间可视为体积为$n_p \times H \times W$的立方体，而应用所需的配置空间可看作单位立方体堆积成的3维空间，如图2所示。

配置问题可进一步抽象为在$n_p \times H \times W$的立方体内，寻找空白区域来放置新任务的配置信息。假



(a) CGDRS精简结构　　(b) 可重构计算资源

图 1 CGDRS整体结构示意

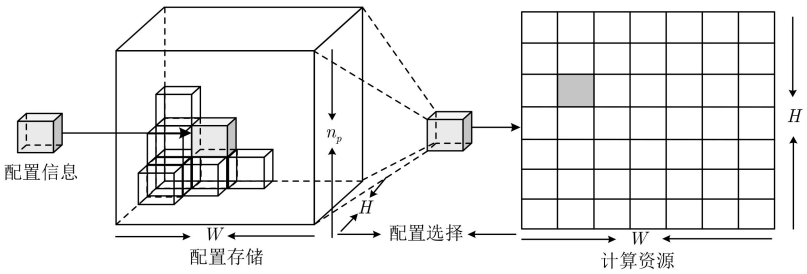


图 2 CGDRS任务配置与计算过程

设单任务所需的配置存储空间不大于CGRA单页面的存储容量，则配置过程存在以下几个约束条件：

(1) 配置存储约束：空闲配置存储空间必须大于或者等于任务所需配置空间，且任意两个任务无配置资源冲突问题。即$\exists p_m \in P$，在页面p_m所确定的平面坐标系中，对于$\forall 1 \le i,j \le k, i \ne j$，

$$\left.\begin{aligned}&\min\{x_i+w_i, x_j+w_j\} \le \max\{x_i, x_j\}\\ \cup&\min\{y_i+h_i, y_j+h_j\} \le \max\{y_i, y_j\}\end{aligned}\right\} \tag{3}$$

(2) 配置端口约束：任意时刻，可重构系统进行配置的任务数量不多于接口带宽W_i与任务处理位宽W_t之间的比值。定义fuc: $x_{s,i}\rightarrow(0,1)$为任务t_i的状态函数，$x_{s,i}$=1表示s时刻任务t_i正在配置，那么

$$\sum_{i=1}^{n_t} x_{s,i} \le W_i / W_t \tag{4}$$

任务在CGRA上计算时不受接口速率的限制，只要任务配置完成，所有的**PE**可并行计算。为提高可重构处理器的资源利用率，在配置存储非空的情况下，计算资源应保持计算状态。具体地，计算问题可抽象为在已缓存完毕且未执行的配置存储中，选择具有最高优先级的任务执行。计算过程存在以下约束条件：

(1) 计算资源约束：CGRA的空闲计算资源要大于或等于任务所需计算资源，且任意两个任务无计算资源冲突，在由**PE**阵列所确定的平面坐标系中，对于$\forall 1 \le i,j \le k, i \ne j$，仍应满足式(3)；

(2) 配置对计算的约束：任务计算阶段必须滞后于任务配置阶段，将任务t_i的配置开始时间记为s_{ci}，计算开始时间记为s_{ei}，则

$$s_{\mathrm{ci}} + c \le s_{\mathrm{ei}} \tag{5}$$

(3) 任务数据依赖关系约束：各任务之间需要满足DAG图所描述的数据依赖关系，所有任务只有在其所有前驱任务执行完毕后才能开始执行。设任务t_j的前驱任务构成集合F，那么$\forall t_i \in F$，需要满足

$$s_{\mathrm{ei}} + e \le s_{\mathrm{ej}},\ \forall t_i \in F \tag{6}$$

3 调度算法设计

3.1 CPSA任务调度算法

本文中，CGDRS上的任务调度目标是最小化任务集合的总执行时间。对于流应用，由于不同任务之间具有数据依赖关系，其计算过程必须是串行的；但配置过程不存在此约束。将任务在CGDRS上的执行过程划分为等待配置、配置、等待计算和计算4个阶段，并定义如下任务类型。

(1) 未配置任务：对于任意$t_i \in A$，若t_i未进入配置阶段，当前系统时间满足$T < s_{\mathrm{ci}}$，则t_i为未配置任务；

(2) 配置完成任务：对于任意$t_i \in A$，若t_i已经完成配置阶段但并未进入计算阶段，但t_i存在未计算完成的前驱任务，即当前的系统时间满足$s_{\mathrm{ci}} + c < T < s_{\mathrm{ei}}$，且$\exists e(v_x, v_i)$，使$T \le s_{\mathrm{ex}} + e$，则$t_i$为配置完成任务；

(3) 计算就绪任务：对于任意$t_i \in A$，若t_i为配置完成任务且其所有的前驱任务均已计算完毕，即当前的系统时间满足$s_{\mathrm{ci}} + c < T < s_{\mathrm{ei}}$，且$\forall e(v_x, v_i)$，使$T \ge s_{\mathrm{ex}} + e$，则$t_i$为计算就绪任务；

(4) 计算任务：对于任意$t_i \in A$，若t_i正在进行计算，即系统时间满足$s_{\mathrm{ei}} \le T \le s_{\mathrm{ei}} + e$，则$t_i$为计算任务。

CGDRS上的任务调度问题，关键在于确定未配置任务的配置优先级和配置完成任务的计算优先级。否则可能会出现后续任务配置完成而无法计算，前驱任务因缺少配置存储空间而无法执行的死锁状态。为避免死锁现象的发生，本文将未配置任务进一步划分为未配置任务与预配置任务。

(5)预配置任务：对于任意$t_i \in A$，若t_i为未配置任务且t_i的所有前驱任务均为配置完成任务，则t_i为预配置任务。

在本文所设计的任务调度算法中，只有预配置任务拥有配置权限，从而保证了所有任务在进行配置时，它们的前驱任务均已配置完成，避免了死锁状态的产生。图3(a)为由4个任务组成的DAG。未采用预配置机制时，各任务的配置与计算过程依次执行，如图3(b)所示；其中c代表任务配置阶段，

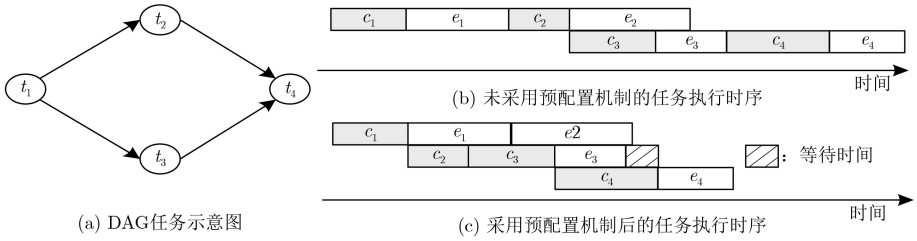


图 3 预配置机制对任务执行时序的影响

e代表执行阶段。采用预配置机制进行调度后，任务集合执行情况如图3(c)所示。当t_1配置完成后，t_2, t_3由未配置任务变为预配置任务；此时根据优先级先后对t_2, t_3进行配置，在满足资源约束的条件下，可隐藏部分配置时间。但需要注意的是，尽管任务的配置过程可以提前，但计算过程仍然要满足DAG所描述的数据依赖关系。因此，任务t_4需要等待t_2, t_3计算完成后开始计算。

任务的优先级按照文献[16]提出的DAG静态调度方案确定。3种典型的DAG节点优先级算法是ASAP(As Soon As Possible), ALAP(As Late As Possible)和 CPF(Critical Path First)算法。具体的计算方案如式(7)所示，其中l_i为任务t_i的优先级。任务t_i的tl值代表在DAG中由源节点出发到达t_i所属节点v_i的最长路径长度；bl值代表在DAG中由t_i所属节点v_i出发到达汇节点的最长路径长度。b_s代表从源节点到达汇节点的任务关键路径长度。

$$l_i=\begin{cases}\text{tl}, & \text{ASAP}\\ b_s-\text{bl}, & \text{ALAP}\\ b_s-\text{tl}-\text{bl}, & \text{CPF}\end{cases} \tag{7}$$

根据上文中定义的任务类型，在CPSA调度算法中设置5个任务队列分别存放未配置任务、预配置任务、配置完成任务、计算就绪任务和计算任务，分别表示为WTQ, PTQ, ATQ, RTQ, ETQ。

如图4所示，任务在各个队列之间的转换过程如下：

步骤 1　在系统未开始工作之前，由于源节点任务不存在前驱任务，所以直接进入PTQ队列，而其余任务全部进入WTQ队列；

步骤 2　当PTQ队列非空，且CGRA配置存储满足配置约束条件时，按照优先级进行任务的配置；

步骤 3　若有任务配置完成，遍历WTQ队列并从中选取预配置任务进入PTQ队列，同时将完成配置的任务由PTQ队列转移至ATQ队列，遍历ATQ队列并从中选取计算就绪任务进入RTQ队列；

步骤 4　当任务所需的计算资源为空闲时，CGRA将占用相同计算资源的计算就绪任务，按照优先级顺序先后执行；同时任务由RTQ队列进入ETQ队列；

步骤 5　任务计算阶段结束后，将其从ETQ队列中移出，并更新ATQ队列；

步骤 6　重复执行步骤2—步骤5直至 WTQ, PTQ, RTQ, ATQ, ETQ为空。

在任务队列中，任务按照优先级高低排序。其中t_0为源节点任务，t_M为某队列中优先级最高的任务，CM_List与CR_List分别代表CGRA的配置资源列表和计算资源列表。

3.2 配置重用策略

CPSA任务调度算法采用预配置策略减少了CGDRS上应用的整体执行时间，但并没有开发不同任务之间的配置重用性。配置可重用是指在应用

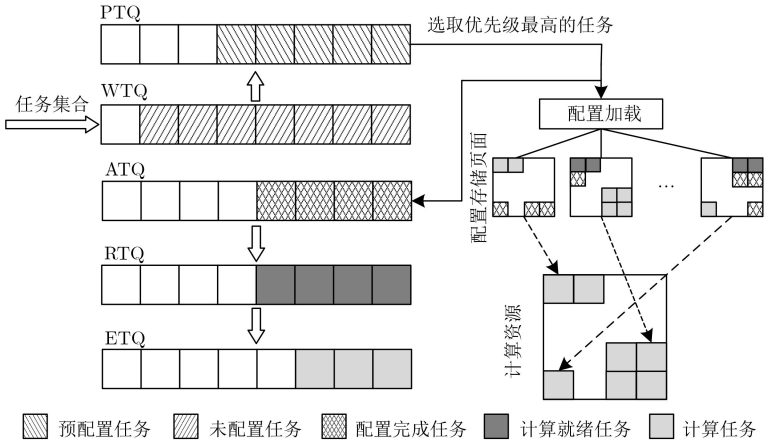
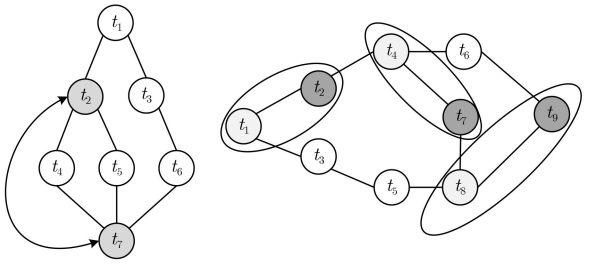


图 4 CPSA算法下任务在各个队列中的转换示意图

中存在不相同的两个任务，它们所处理的数据不同，但对数据的处理方式相同，任务所使用的**PE**相同，即配置信息完全相同。开发配置的可重用性可以显著降低应用的整体配置时间，同时降低配置存储空间的占用。如图5所示，配置信息存在两种重用方式：(1)配置信息间隔可重，这是指某任务的配置信息在一定数量的其他任务执行完毕后能够再次被利用；(2)配置信息连续可重用，这是指n个连续任务$t_i, \cdots, t_{i+n-1}$在执行完毕后，它们的配置信息仍然可以被直接重复利用m次。



(a) 间隔可重用配置信息 (b) 连续可重用配置信息

图 5 可重用配置信息示意

由于DAG图中不能表示循环操作，所以将配置重用表示为n个节点的重复出现。下面给出间隔重用任务和连续重用任务集合的具体定义。

(1) 间隔重用任务：假设任务t_i的配置信息表示为Ct^i, 对于$t_i \in A$, 若存在任务t_j使式(8)成立，则两个任务的配置信息间隔可重用，且任务t_j为t_i互为间隔重用任务。

$$\mathrm{Ct}^i=\mathrm{Ct}^j \cup j \notin [i-1,i+1] \tag{8}$$

本文对间隔重用任务提出了两种配置策略：一为配置保存策略：将任务t_i的配置信息保存在配置存储中，当t_i的配置重用任务出现时可以直接利用已有的配置信息实现重用，直到t_i的所有配置重用任务都执行完毕，该配置存储区域才可被覆盖。此方案能够显著降低可重用任务的配置时间，但需要长期占用配置资源，配置灵活性较低。其二是配置检测策略：该策略并不为任务保存专用的配置存储，而是按照正常的CPSA算法调度流程运行。当有新任务t_i进行配置时，遍历ATQ队列中的所有任务，若检测到存在t_i的间隔重用任务则省略配置过程直接将t_i加载至ATQ队列。配置检测策略相较于配置保存策略无需占用配置存储空间，具有更高的配置灵活性，因此本文采用配置检测策略调度间隔重用任务；

(2)连续重用任务集合：对于$t_i, \cdots, t_{i+n-1} \in A$，若存在任务$t_{i+n}, \cdots, t_{i+m\times n-1}$使式(9)成立，则称任务$\{t_i, \cdots, t_{i+n-1}\}$为连续重用任务集合，表示集合内任务的配置信息连续可重用。

$$\bigcup \sum_{k=i}^{i+n-1} \mathrm{Ct}^k=\mathrm{Ct}^{k+n}=\cdots=\mathrm{Ct}^{k+(m-1)n} \tag{9}$$

对于连续重用任务，本文提出了一种任务优先执行策略。当系统检测到任务$t_i, \cdots, t_{i+n-1}$构成连续重用任务集合时，仍然按照预配置优先级加载配置信息，当任务$t_i, \cdots, t_{i+n-1}$的配置信息均加载完毕后，系统不按照预先设定的优先级计算任务，而是先将任务集合重复计算m次。而后再计算其它任务。鉴于连续重用任务集合之间串行化的执行顺序，任务间的数据依赖关系不会被破坏。

3.3 基于预配置和配置重用策略的CPSA调度算法

为进一步减少流应用的整体执行时间，本节在CGDRS算法基础上，引入配置重用机制，对CPSA调度算法作出改进，相较于原算法的主要改变是添加了两个可重用任务的检测模块。检测模块分别作用于配置完成任务ATQ与计算就绪任务队列RTQ中，当检测到ATQ中存在当前正在配置任务的间隔重用任务时，直接将任务由PTQ加载至ATQ；当检测到RTQ中存在连续重用任务集合且该任务集合所需的计算资源可用，则直接将该任务集合执行m次，并将其加载至ETQ。改进后的CPSA算法运行流程如图6所示。

4 仿真实验

4.1 实验环境

仿真环境采用C++语言在Microsoft Visual Studio 2012环境中编写并测试通过。可重构计算资源的规模定义为16×16=256个**PE**，每个**PE**拥有8个配置页面。实验中具体的应用产生方法如下。设各应用包含的任务数量为[50, 150]。在各应用中设置$J(J\in[1, 30])$组间隔可重用任务，组内任务的配置相同；为简化实验，设每组内包含间隔可重用任务的数量为2。设置$L(L\in[1, 15])$组连续重用任务集合，每个集合中包含任务的数量为[1, 3]个；并设每个任务集合被重复执行2次。各应用中，除源节点和汇节点外，其他节点的入度和出度取值范围为[1, 10]。对于任务$t_i(x_i, y_i, w_i, h_i, c_i, e_i)$，任务高度和宽度在区间[1, 8]内随机生成。设定单位时间pt为10 ns，则任务配置时间$c_i=r\times w_i\times h_i\times \mathrm{pt}$。其中，$r$为配置系数，$r$在区间[0.5, 1.0]内随机取值。而任务的计算时间$e_i=k\times c_i, k\in[0.5, 2.0]$。

将J与L的取值空间进行3等分，分别得到3个子取值空间，记为$J10$, $J20$, $J30$和$L5$, $L10$, $L15$。对J与L的子取值空间进行自由组合，共产生9种子

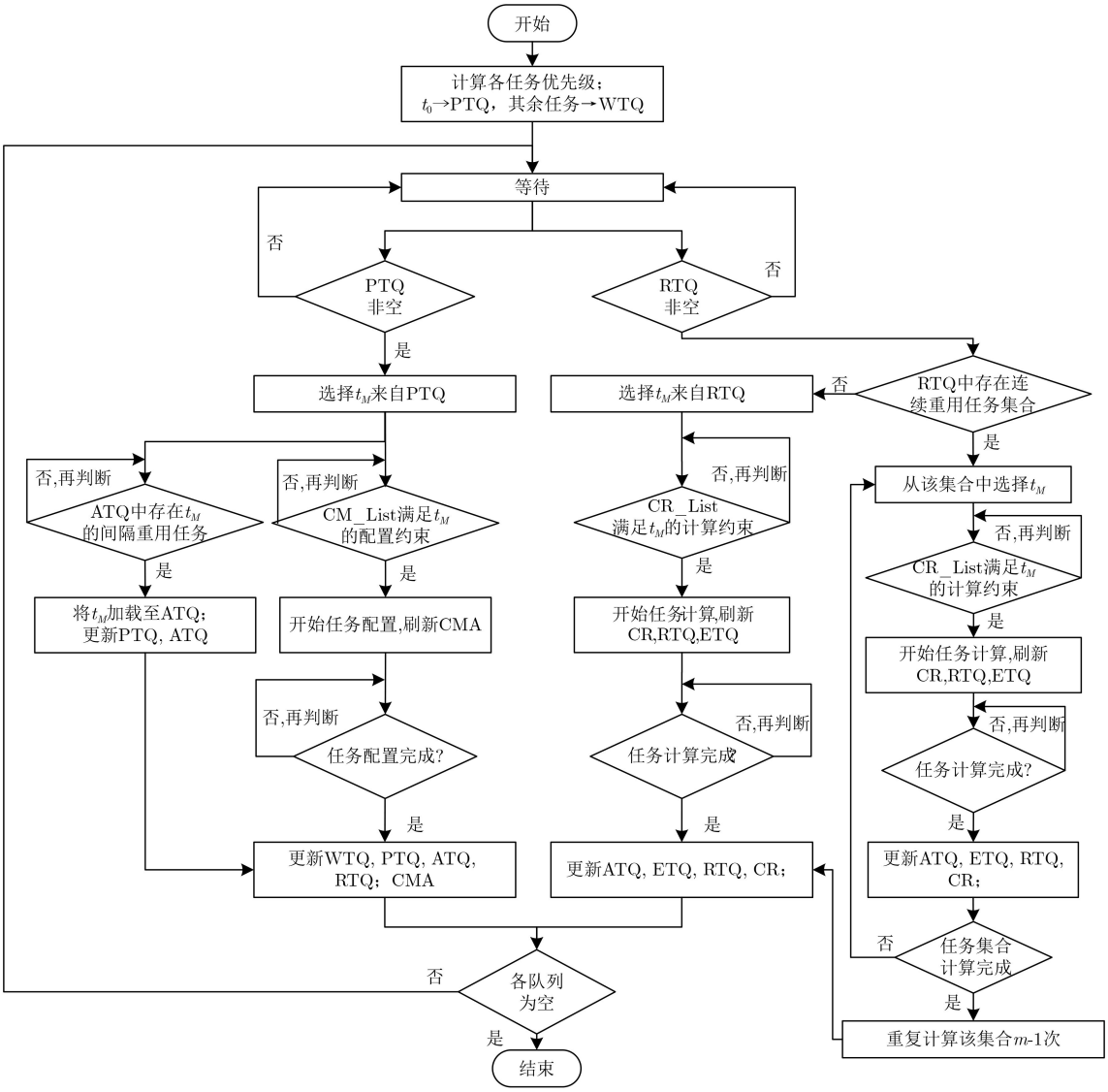


图 6 改进后的CPSA调度算法流程图

取值空间组合形式，如图7中的横坐标所示。实验采用ALAP算法确定任务的优先级，在不同的子取值空间组合下各产生500组测试集合，将所有应用的仿真数据平均值记为测试结果。

4.2 实验结果分析

4.2.1 可调度性评估

本文用调度成功率来衡量调度算法的可调度性。调度成功率即是指可以在约束时间内执行完毕的应用比例(因无调度死锁现象从而可以正确执行完毕的应用比例)。分别在9种不同的可重用任务条件下，将本文所提CPSA调度算法与Pre[2], CCF[3], RCP[8], BULB[9]进行调度成功率的对比，实验结果如图7所示。

从图7可以得出，CPSA算法与Pre算法均取得了100%的调度成功率，较CCF、RCP及BULB算法均有提高。这是因为二者均采用了预配置思想，它可以解决调度死锁问题，从而保证应用的正确执行。实验证明了预配置策略在可重构系统任务调度问题上的可行性。但调度成功率只能初步评估各调度算法的调度能力，为此实验进一步评估了调度算法的有效性。

4.2.2 有效性评估

本文用应用执行时间来评估应用的性能；应用执行时间越短，表明相应的调度算法越有效。为评估本文技术的有效性，将本文技术与None, Pre, CCF, RCP以及BULB5种算法进行比较。其中None代表未使用任何任务调度算法。各调度算法下，任务的执行时间如图8所示，时间单位为ns。

各调度算法相较于None来说，均在不同程度上降低了计算应用的总执行时间。Pre, CCF和RCP算法均采用了预配置策略；但使用CCF与CPSA算法时，应用执行时间更低。这是因为两种算法同时

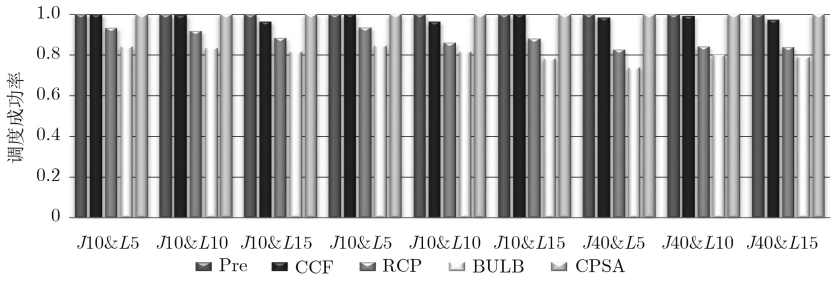


图 7 调度成功率对比图

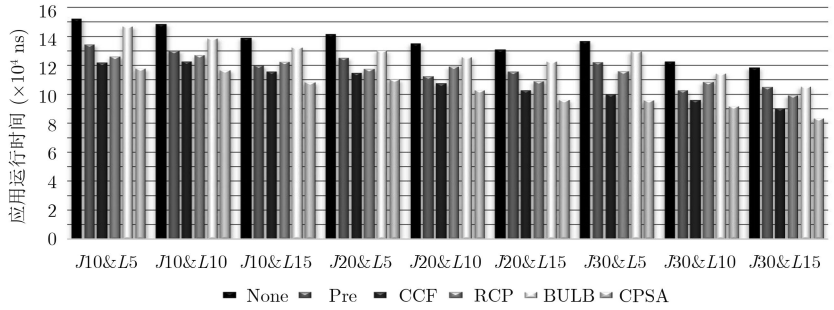


图 8 CPSA调度算法与其它算法调度结果对比

采用了配置重用策略，从而进一步减少了应用的配置时间。相较而言，BULB对于应用执行时间的优化效果最差，这主要是因为BULB算法的目标是提高可重构系统的资源利用率和降低应用执行时间，且侧重于前者。表1进一步量化了CPSA调度算法相较于其他算法对应用执行时间和调度时间开销的优化情况。将CPSA算法调度后的应用执行时间记为T_{after}、采用其他调度方案的应用执行时间记为T_{ago}，则应用执行时间优化比例为

$$O=(1-T_{\text{after}}/T_{\text{ago}})\times 100\% \tag{10}$$

由表1可知，在不同的任务重用条件下，相较于未使用预配置策略的BULB调度算法，CPSA算法对应用执行时间的优化比例均值达到19.53%；这说明了预配置策略在隐藏任务配置时间方面的优越性。相较于同样基于预配置策略的Pre和RCP算法，本文技术的优化比例平均约为13.67%和11.91%，这主要是由于CPSA调度算法采用了配置重用策略，能够进一步开发任务间的配置重用性。而相较于同样考虑任务配置重用性的CCF调度算法来说，CPSA算法依然取得了6.13%的优化比例；这主要因为CCF算法只开发了间隔重用任务的配置重用性，而CPSA算法进一步开发了连续重用任务的配置重用性，对任务的配置重用性的开发更为充分。

表 1 CPSA调度算法相较于其它调度算法对应用执行时间的优化比例(%)

应用	O(Pre)	O(CCF)	O(RCP)	O(BULB)
$J20\&L5$	12.30	4.38	6.50	19.79
$J20\&L10$	10.32	5.87	8.06	15.73
$J20\&L15$	9.66	7.33	11.48	18.18
$J30\&L5$	12.06	5.02	6.35	15.28
$J40\&L15$	20.75	8.63	16.10	20.90
$J30\&L10$	8.57	5.59	13.81	18.26
$J30\&L15$	16.95	7.55	11.93	21.63
$J40\&L5$	21.57	5.11	17.28	26.02
$J40\&L10$	10.89	5.68	15.72	19.97
平均值	13.67	6.13	11.91	19.53

综上所述，CPSA调度算法与其它可重构系统调度算法相比，通过定义预配置任务，可避免调度死锁现象；通过深度开发任务的配置重用性，可高效利用CGRA的配置存储空间。因此本文技术对CGRA体系结构和流应用具有更强的适应性。

5 结束语

本文研究了粗粒度动态可重构系统上的流任务调度问题。为解决配置死锁问题，本文提出一种基于预配置策略的任务调度算法CPSA。算法在配置排序问题上考虑了任务优先级，以提高调度精确性；并设计了“配置-计算”流水线以隐藏任务配置时间。然后分析了流应用中任务间存在的配置重用性，并定义了两种配置可重用的任务类型：间隔重用任务与连续重用任务；进而提出了间隔配置重用策略和连续配置重用策略。基于两种配置重用策略对CPSA调度算法进行了改进，进一步优化了应用的整体执行时间。实验结果表明，CPSA算法能有效解决调度死锁问题并减少流应用的执行时间；

相较于其它文献中提出的调度算法，对应用执行时间的优化比例达到了6.13%～19.53%。

参 考 文 献

[1] WANG Yansheng, LIU Leibo, YIN Shouyi, *et al*. On-chip memory hierarchy in one coarse-grained reconfigurable architecture to compress memory space and to reduce reconfiguration time and data-reference time[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2014, 22(5): 983–994. doi: 10.1109/TVLSI.2013.2263155.

[2] 梁楑, 周学功, 王颖, 等. 采用预配置策略的可重构混合任务调度算法[J]. 计算机辅助设计与图形学学报, 2007, 19(5): 635–641. doi: 10.3321/j.issn:1003-9775.2007.05.016.
LIANG Liang, ZHOU Xuegong, WANG Ying, *et al*. Pre-configuration based hybrid tasks scheduling in reconfigurable systems[J]. *Journal of Computer-Aided Design & Computer Graphics*, 2007, 19(5): 635–641. doi: 10.3321/j.issn:1003-9775.2007.05.016.

[3] 王延升. 粗粒度动态可重构处理器中的高能效关键配置技术研究[D]. [博士论文], 清华大学, 2014: 1–13.
WANG Yansheng. High energy-efficient key techniques in configurations for coarse-grained dynamically reconfigurable processor[D]. [Ph.D. dissertation], Tsinghua University, 2014: 1–13.

[4] LI Zhiyuan and HAUCK S. Configuration prefetching techniques for partial reconfigurable coprocessor with relocation and defragmentation[C]. Proceedings of 2002 ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays, Monterey, California, USA, 2002: 187–195. doi: 10.1145/503048.503076.

[5] 韩晓亚, 汪斌强, 黄万伟, 等. 采用配置完成优先策略的可重构任务调度算法[J]. 小型微型计算机系统, 2012, 33(3): 587–593. doi: 10.3969/j.issn.1000-1220.2012.03.027.
HAN Xiaoya, WANG Binqiang, HUANG Wanwei, *et al*. Scheduling algorithm for dependent reconfigurable tasks based on configuration completion first[J]. *Journal of Chinese Computer Systems*, 2012, 33(3): 587–593. doi: 10.3969/j.issn.1000-1220.2012.03.027.

[6] LIFA A, ELES P, and PENG Zebo. Minimization of average execution time based on speculative FPGA configuration prefetch[C]. Proceedings of 2012 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 2012: 1–8. doi: 10.1109/ReConFig.2012.6416761.

[7] LIFA A, ELES P, and PENG Zebo. A reconfigurable framework for performance enhancement with dynamic FPGA configuration prefetching[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016, 35(1): 100–113. doi: 10.1109/TCAD.2015.2448694.

[8] MORALES-VILLANUEVA A, KUMAR R, and GORDON-ROSS A. Configuration prefetching and reuse for preemptive hardware multitasking on partially reconfigurable FPGAs[C]. Proceedings of 2016 Conference on Design, Automation & Test in Europe, Dresden, Germany, 2016: 1505–1508.

[9] WU Binbin, YAN Like, WEN Yuan, *et al*. Run-time configuration prefetching to reduce the overhead of dynamically reconfiguration[C]. Proceedings of the 23rd IEEE International SOC Conference, Las Vegas, USA, 2010: 305–308. doi: 10.1109/SOCC.2010.5784651.

[10] RUSCHKE T, JUNG L J, and HOCHBERGER C. A near optimal integrated solution for resource constrained scheduling, binding and routing on CGRAs[C]. Proceedings of 2017 IEEE International Parallel and Distributed Processing Symposium Workshops, Lake Buena Vista, USA, 2017: 213–218. doi: 10.1109/IPDPSW.2017.99.

[11] RUSCHKE T, JUNG L J, WOLF D, *et al*. Scheduler for inhomogeneous and irregular CGRAs with support for complex control flow[C]. Proceedings of 2016 IEEE International Parallel and Distributed Processing Symposium Workshops, Chicago, USA, 2016: 198–207. doi: 10.1109/IPDPSW.2016.72.

[12] BONDALAPATI K and PRASANNA V K. Reconfigurable computing systems[J]. *Proceedings of the IEEE*, 2002, 90(7): 1201–1217. doi: 10.1109/JPROC.2002.801446.

[13] 陈锐, 杨海钢, 王飞, 等. 基于自路由互连网络的粗粒度可重构阵列结构[J]. 电子与信息学报, 2014, 36(9): 2251–2257. doi: 10.3724/SP.J.1146.2013.01646.
CHEN Rui, YANG Haigang, WANG Fei, *et al*. Coarse-grained reconfigurable array based on self-routing interconnection network[J]. *Journal of Electronics & Information Technology*, 2014, 36(9): 2251–2257. doi: 10.3724/SP.J.1146.2013.01646.

[14] 徐金甫, 刘露, 李伟, 等. 一种基于阵列配置加速比模型的无损压缩算法[J]. 电子与信息学报, 2018, 40(6): 1492–1498. doi: 10.11999/JEIT170900.
XU Jinfu, LIU Lu, LI Wei, *et al*. A new lossless compression algorithm based on array configuration speedup model[J]. *Journal of Electronics & Information Technology*, 2018, 40(6): 1492–1498. doi: 10.11999/JEIT170900.

[15] 徐晓东. 动态可重构系统中任务调度与布局算法研究[D]. [硕士论文], 中国科学技术大学, 2017: 35–48.
XU Xiaodong. Task scheduling and floorplanning algorithm in dynamically reconfigurable systems[D]. [Master dissertation], University of Science and Technology of China, 2017: 35–48.

[16] KWOK Y K and AHMAD I. Static scheduling algorithms for allocating directed task graphs to multiprocessors[J]. *ACM Computing Surveys*, 1999, 31(4): 406–471. doi: 10.1145/344588.344618.

戴紫彬：男，1966年生，教授，博士生导师，研究方向为可重构计算与安全专用芯片设计.

曲彤洲：男，1994年生，硕士生，研究方向为可重构计算与信息安全.