计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200700198

一种面向动态部分可重构片上系统的列表式软硬件划分算法

郭 彪[1,2] 唐 麒[2] 文智敏[3] 傅 娟[4] 王 玲[1] 魏急波[2]
1 湖南大学电气与信息工程学院 长沙 410082
2 国防科技大学电子科学学院 长沙 410073
3 长沙轨道交通运营有限公司车辆部 长沙 410000
4 军事科学院系统工程研究院 北京 100101
(guobiao@hnu.edu.cn)

摘 要 并行计算是提高系统资源利用率的重要手段,越来越多的多处理器片上系统通过集成具有不同功能特点的处理器来满足不同计算任务的需求。具备动态部分可重构特性的异构多处理器片上系统(Dynamic Partial Reconfiguration-Heterogeneous Multiprocessor Systems-on-Chip,DPR-HMPSoC)因其并行性好、计算效率高而被广泛使用,而低复杂度和高求解性能的软硬件划分算法是充分发挥其计算性能优势的重要保证。已有的相关软硬件划分算法时间复杂度高,且对DPR-HMPSoC平台的支撑不足。针对上述问题,首先提出了一种列表启发式软硬件划分与调度算法,其通过构建基于任务优先级的调度列表,完成任务的调度、映射、FPGA动态部分可重构区域划分等一系列操作;接着给出了软件应用建模、计算平台建模及所提算法的详细设计方案。仿真实验结果表明,所提算法与混合整数线性规划(Mixed Integral Linear Programming,MILP)和蚁群优化(Ant Colony Optimization,ACO)算法相比,可有效减少求解时间,且时间优势与任务规模成正比;在调度长度方面,所提算法的平均性能提升了约10%。

关键词:软硬件划分;列表启发式;动态部分可重构;现场可编程逻辑门阵列;调度
中图法分类号 TP302

List-based Software and Hardware Partitioning Algorithm for Dynamic Partial Reconfigurable System-on-Chip

GUO Biao[1,2],TANG Qi[2],WEN Zhi-min[3],FU Juan[4],WANG Ling[1] and WEI Ji-bo[2]
1 College of Electrical and Information Engineering,Hunan University,Changsha 410082,China
2 School of Electronic Science and Technology,National University of Defense Technology,Changsha 410073,China
3 Vehicle Department of Changsha Rail Transit Operation Co.,Ltd,Changsha 410000,China
4 Institute of System Engineering,Academy of Chinese PLA Military Science,Beijing 100101,China

Abstract Parallel computing is an important means to improve the utilization rate of system resources. More and more systems on multi-processor chip meet the requirements of different computing tasks by integrating processors with different functional characteristics. A heterogeneous multiprocessor system-on-chip (DPR-HMPSoC) with dynamic partial reconfigurability is widely used because of its good parallelism and high computing efficiency,while the software/hardware partitioning algorithm with low complexity and high solving performance is an important guarantee for giving full play to its computational performance advantages. The existing related software/hardware partitioning algorithms have high time complexity and insufficient support for the DPR-HMPSoC platform. In response to the above problems,this paper proposes a list heuristic software/hardware partitioning and scheduling algorithm. By constructing a scheduling list based on task priority,a series of operations such as task scheduling,mapping and FPGA dynamic partial reconfigurable area partitioning are completed. It introduces software application mode-ling,computing platform modeling,and the detailed design scheme of the proposed algorithm. The simulation experiment results show that the proposed algorithm can effectively reduce the solution time compared with the MILP and ACO algorithms,and the time advantage is proportional to the task scale. In terms of scheduling length,the average performance of the proposed algorithm is improved by about 10%.

Keywords Software/hardware partitioning,List-based heuristics,Dynamic partial reconfigurable,FPGA,Scheduling

到稿日期:2020-07-30 返修日期:2020-08-20 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
通信作者:唐麒(q.tang.andy@qq.com)

1 引言

随着5G、机器学习及云计算技术的发展，大量计算密集型应用对硬件设备的算力需求越来越大。在嵌入式计算领域，软件执行性能与硬件计算资源的变化规律往往呈正相关。截至目前，世界量产的最先进的芯片制造工艺为5nm级，芯片制造商台积电、三星和Intel的3nm工艺也即将商用，在芯片制程逐渐逼近原子尺寸的同时，其引发的原子间相互作用更加复杂，摩尔定律或陷入停滞，以增加单位面积的晶体管数量来提升芯片计算能力的模式将面临瓶颈。为解决上述问题，各大芯片设计厂商在并行计算、多核处理、硬件可重构及软硬件协同上进行架构创新和硬件集成。以Xilinx的Zynq-7000为例，其推出了集成ARM内核的SoC，如Zynq-7000 SoC，Zynq UltraScale＋MPSoC和Zynq-7000 UltraScale＋RFSoC，将软件可编程与硬件可编程完美结合，同时支持FPGA的动态部分可重构(Dynamic Partial Reconfiguration，DPR)，即允许在FPGA运行时更改部分硬件逻辑单元的配置而不影响或者中止其他正在执行的计算单元。本文将上述集成了DPR FPGA和CPU的计算平台称为DPR-HMPSoC。

软件应用由多个具有相互依赖关系的子任务组成，在DPR-HMPSoC平台上执行任务时，不同的任务执行顺序和映射方式会对计算资源利用率及软件执行结束时间产生影响。此外，任务在硬件上的执行速度一般是CPU上执行速度的3～5倍[1]。动态部分可重构提升了系统的灵活性，让应用可以更加充分地利用不同处理器资源，缩短了应用的总执行时间[2]。软硬件协同设计和应用的软硬件划分与调度是提升系统综合性能的关键。本文面向DPR-HMPSoC计算平台，对如何在满足多项物理约束的前提下实现高效的软硬件划分与调度进行研究。

面向DPR-HMPSoC计算平台的软硬件划分与调度问题属于组合优化问题，是一种典型的NP-难问题。针对上述问题，目前的算法主要有两类：一是规划类求解算法，如整数线性规划(Integral Linear Programming，ILP)[3-4]、MILP[1]，其通过建立约束方程使用数学求解器得出问题的最优解。二是启发式算法，包括基于任务优先级的列表启发式和元启发式两种。基于任务优先级的列表启发式算法依据产生的任务优先级列表进行调度[5]；元启发式算法通过模拟自然界中的自然现象将随机算法与局部搜索算法相结合，主要有禁忌搜索算法(Tabu Search，TS)、模拟退火算法(Simulated Annealing，SA)[6-7]、遗传算法(Genetic Algorithm，GA)[8-10]、蚁群优化算法(ACO)[11-13]等。

现有的规划类求解算法研究中，基于ILP与MILP的研究最为广泛，即约束方程中变量全为整数或包含部分实数，通过数学求解器求解问题的最优解，常用的数学求解器有Lindo/Lingo，Gurobi，Cplex，Glpk，lpsolve等。文献[3]建立了ILP与MILP两种约束方程模型来解决RCSP(Resource Constrained Scheduling Problem)[14]调度问题，并与其他4种基于ILP的算法进行了对比，实验结果证明其提出的约束方程模型较好，但其只提出了任务的调度问题模型，没有将其应用到特定的计算平台。文献[15]针对FPGA提出了基于ILP的流水线调度算法。文献[16]利用ILP算法解决了多处理器片上网络系统中的任务调度问题。文献[1]在文献[3]的基础上提出了针对DPR-HMPSoC计算平台的MILP算法，得到了软硬件划分的最优解，但是模型求解的时间复杂度很大，且文献[1]是通过对任务进行分簇来确定任务的划分方式(即提前约束任务在硬件或软件上执行)，对于实际情况中任务在软硬件上都可执行的情况，该模型存在支撑不足的问题。总的来说，现有的规划类求解算法可以求解出软硬件划分与调度问题的最优解，但算法的时间复杂度较大，随着任务规模的增大，求解时间呈指数级增长，一个中等规模的应用求解需要数十小时，难以满足实际情况中的求解时间要求。

文献[12]采用ACO算法解决了GPP，DSP，FPGA等异构多处理器片上系统(Multiprocessor Systems-on-Chip，MPSoC)中任务的调度、映射问题，但是没有建模DPR FPGA。文献[13]基于文献[12]的算法，以Xilinx Virtex-II Pro为计算平台建模，在MPSoC中增加了DPR FPGA的硬件结构，采用ACO算法进行软硬件划分与调度，与KLFM算法[2]相比其平均性能提升了16.5%。但文献[13]使用的DPR FPGA是一维重构结构，重构方式有很大的局限性且缺少灵活性。文献[6-7]提出了一个针对DPR FPGA的综合框架，综合考虑了软硬件划分与调度，并使用模拟退火算法来进行求解和优化。文献[8]采用遗传算法进行针对多处理器的任务调度，与文献[12]相同，计算平台中没有建模DPR FPGA。文献[9]使用遗传算法进行软硬件划分与调度，并将异构通信时延作为算法设计的重要参数。总的来说，基于ACO，GA，SA等元启发式的随机搜索算法通过随机搜索寻找软硬件划分与调度的较优解，但在大规模应用时，搜索时间较长，且容易出现局部最优解。

DPR-HMPSoC是未来计算平台发展的趋势，而面向该平台的软硬件划分与调度算法是充分实现其并行、异构计算和硬件加速优势的重要保证。在现有的研究中，应用调度长度(Scheduling Length，SL)和求解时间是评价软硬件划分与调度算法的两个性能指标。针对现有的软硬件划分算法存在的问题和不足，本文以DPR-HMPSoC为计算平台进行建模，将任务调度、任务映射及FPGA动态部分可重构区域划分进行协同设计，提出了一种基于贪心策略的任务列表启发式软硬件划分算法。由对比实验结果可知，所提算法在大幅减少求解时间的同时，获得了优于对比算法的软硬件划分与调度结果。

2 问题描述

2.1 软件应用模型

在应用调度理论研究中应用的建模方法很多，如DAG(Directed Acyclic Graph)[13,17-21]，SDFG(Synchronous Dataflow Graph)[22]，CSDFG(Cyclo-Static Dataflow Graph)[22]，SADFG(Scenario-Aware Dataflow Graph)[22]等。与文献[13，

17-21]相同，本文用有向无环图(Directed Acyclic Graph，DAG)来建模应用。一个 DAG 可表示为 $G=(T,E)$。其中，$T=\{t_0,t_1,\cdots,t_{n-1},t_n\}$，表示组成应用的任务集合；$E=\{e_0,e_1,\cdots,e_{m-1},e_m\}$ 为边的集合，各边表示任务之间的数据依赖关系。

任务 $t_i\in T$ 可表示为 $t_i=(hw_t,sw_t,rc_t,clb_{num})$，各元素分别表示任务在 FPGA 和 CPU 上的执行时间、在 FPGA 上的重构时间以及在 FPGA 上执行所需的 CLB 资源数量。

边 $e_i\in E$ 表示节点间的数据依赖关系，可表示为 $e_i=(parent_i,child_i,cost_i)$，各元素分别表示该依赖边指向的父任务、子任务以及父任务与子任务之间的通信时间。图 1 所示为一个 DAG 模型示例，该 DAG 的任务参数如表 1 所列。

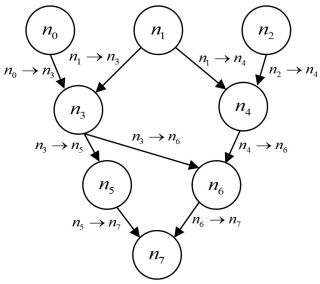


图 1　DAG 图示例

Fig. 1　Example of DAG graph

如图 1 所示，该 DAG 由 8 个任务和 9 条边组成。表 1 所列为任务参数信息，包括任务名、任务的软件执行时间、任务的硬件执行时间以及在 FPGA 上执行所需的 CLB 资源数量。

表 1　示例应用任务参数

Table 1　Task parameters of sample application

Task name	Sw execution time	Hw execution time	CLB num
n_0	26	13	9
n_1	9	4	4
n_2	10	4	2
n_3	23	6	2
n_4	6	1	1
n_5	28	11	3
n_6	24	2	1
n_7	2	1	1

2.2　系统计算平台模型

本文将 DPR-HMPSoC 作为研究平台。以赛灵思 Zynq-series SoC 为例，其计算平台可抽象为由一片 CPU 与 DPR FPGA 组成的异构系统，表示为 $K=(P,H)$。P 代表 CPU；H 代表 FPGA，可表示为 $H=\{PR_0,PR_1,\cdots,PR_K\}$，$PR_i$ 表示 FPGA 的一个可重构区域。本文研究的 DPR-HMPSoC 计算平台抽象模型如图 2 所示。该系统由一个 CPU 和一个支持二维重构的 DPR FPGA 组成。在 FPGA 的二维重构方式中，FPGA 的 CLB，DSP，BRAM 等逻辑资源呈列状排列。二维重构方式允许 FPGA 在有限范围内以任意矩形的大小进行重构，具有较大的灵活性，减少了不必要的资源重构开销，提高了资源利用率。图 2 中，FPGA 被划分为 3 个动态部分可重构子区域和一个静态子区域，每个区域可以通过 ICAP (Internal Configuration Access Port)端口下载不同的配置文件来更改区域的逻辑功能，但不影响或中止其他区域上的任务执行。FPGA 与 CPU 之间通过共享内存的方式来完成数据通信。

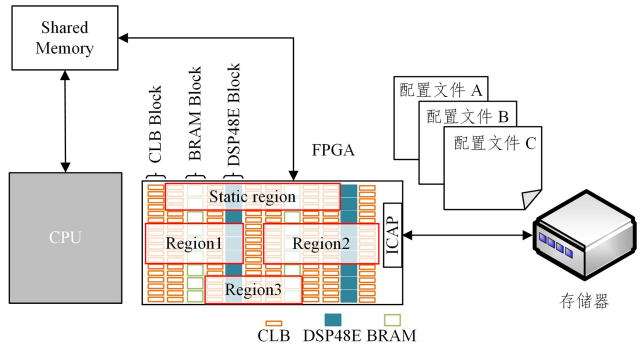


图 2　DPR-HMPSoC 系统平台的抽象模型

Fig. 2　Abstract model of DPR-HMPSoC system platform

2.3　划分、映射与调度

软硬件划分与调度是指确定应用所有任务的执行方式(软件执行或硬件执行)和执行时间(任务开始时间和结束时间)。在上述应用模型和平台模型上，软硬件划分问题可分解为映射、调度和划分 3 个子问题：

(1)划分问题指确定 FPGA 可重构区域的数量及各重构区域的资源数量(区域大小)；

(2)映射问题指确定任务的执行方式($t_i\to K$)，即建立 $t_i\to P$ 或 $t_i\to PR_j$ 的映射关系；

(3)调度问题指确定所有任务的执行顺序以及各任务的执行开始时间、执行结束时间和重构开始时间。

值得注意的是，由于系统硬件平台的物理条件限制，在进行映射、调度和划分时应遵循如下约束条件：

约束 1　任务映射时只能映射到一种类型的计算单元(P 或 H)，且只能映射一次。

约束 2　具有数据依赖关系的任务需要等其所有父任务执行结束后才能执行(若父任务与其在同一计算单元，此时由于传输速率较大，通信时延可忽略不计；若不在同一计算单元，则需等待通信完成后再执行)。

约束 3　划分到 P 上的任务只能串行执行。

约束 4　划分到 H 上的任务需要先重构再执行。

约束 5　任务在 PR_j 上执行时，PR_j 的资源数量应不小于任务执行所需的资源数量。

约束 6　同一时间上 H 执行的所有任务的资源总量不大于 FPGA 的总资源数量。

约束 7　H 的重构端口不能复用，即同一时间只能重构一个任务。

图 3 为示例应用的软硬件划分与调度结果示意图，包含一个 CPU 和 FPGA 的两个可重构区域 PR_0 和 PR_1。其中，Sw 表示在 CPU 上执行的任务，Hw 表示在 FPGA 上执行的任务；深灰色矩形代表任务的重构时间，浅灰色矩形代表任务的执行时间。

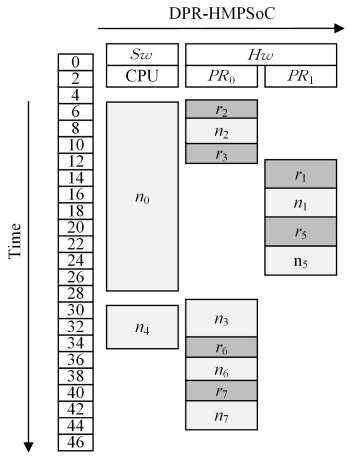


图 3 示例应用软硬件划分与调度结果图

Fig. 3 Hardware and software partitioning results of sample application

3 列表式软硬件划分与调度算法

本文提出一种针对 DPR-HMPSoC 的列表启发式软硬件划分与调度算法，其具体过程包括构建任务优先级列表、进行任务插入、动态划分 FPGA 可重构区域等；同时，使用贪心策略优化任务的最早完成时间。

我们将应用的总完成时间（调度长度）定义为 SL，因此软硬件划分与调度算法的目标函数为：

$$\min SL \tag{1}$$

3.1 构建基于任务优先级的调度列表

实现本文算法需首先构建一个基于任务优先级的调度列表。该列表包含了 DAG 中的所有任务，所有任务根据任务优先级递减的方式排列。计算任务优先级时，任务 t_i 的平均执行时间 $\bar{\varphi}_i$、t_i 与任务 t_j 之间的数据传输时延（通信时延）$cost_{i,j}$、t_i 的任务优先级大小（权值）$blevel(i)$ 分别为：

$$\bar{\varphi}_i=\frac{hw_t+sw_t}{2} \tag{2}$$

$$cost_{i,j}=\frac{data_{i,j}}{B} \tag{3}$$

$$blevel(i)=\begin{cases}\bar{\varphi}_i, & i\in exits\\ \bar{\varphi}_i+\max\limits_{j\in childs(i)}(blevel(j)+cost_{i,j}), & \text{otherwise}\end{cases} \tag{4}$$

其中，hw_t 表示任务的硬件执行时间，sw_t 表示任务的软件执行时间；$data_{i,j}$ 为两个节点之间通信传输的数据量大小，B 表示传输速率；$childs(i)$ 表示任务 t_i 的子任务集合；$exits$ 表示 DAG 中没有子任务的节点集合，如图 1 中的任务 n_7。

通过计算 DAG 中所有任务的 $blevel$，并按照其值的大小递减排序，得到待调度的任务拓扑排序列表。

3.2 任务调度与映射

任务调度时按照 3.1 节任务拓扑排序列表中的排序依次进行。由于列表是拓扑排序，故保证了父任务一定会在子任务之前调度，满足了节点之间的数据依赖关系。本文使用 $DRT_{i,k}$ 表示任务 t_i 在计算单元 k 上的数据就绪时间，即任务在 k 上最早开始执行时间的理论值。

$$DRT_{i,k}=\max_{j\in parents(i)}(tft(j)+cost_{i,j}),k\in P \text{ or } k\in H \tag{5}$$

其中，$tft(j)$ 为任务 t_i 的父任务 t_j 的执行结束时间，$parents(i)$ 为任务 t_i 的父任务集合，$cost_{i,j}$ 为任务 t_i 与其父任务 t_j 的通信时延。若其父任务映射到相同计算单元，此时由于传输速率较大，$cost_{i,j}$ 可忽略不计。

$DRT_{i,k}$ 是任务 t_i 在计算单元 k 上最早开始执行时间的理论值，实际上任务在计算单元 k 上开始执行的最早时间还应考虑计算单元是否有其他任务正在执行，若 $k\in H$，还应判断是否有足够的硬件资源满足任务的执行需求。故令 $EST_{i,k}$ 为任务在 k 上的实际最早开始执行时间（Earliest Start Time，EST）。

$$EST_{i,k}=\max(valid(k),DRT_{i,k}),k\in P \text{ or } k\in H \tag{6}$$

其中，$valid(k)$ 表示计算单元 k 的最早有效可用时间，其计算方式如 3.3 节中的式(8)所示。由式(6)可以得出任务在每个计算单元 k 上的实际最早开始执行时间。为了获得整个应用的最早结束时间，我们使用贪心策略选择当前任务在所有计算单元上的开始执行时间的最小值，如下所示：

$$\min EST_k,k\in P \text{ or } k\in H \tag{7}$$

通过式(7)得到任务的最早开始执行时间，并将取得该值的计算单元 k 作为任务的执行单元，使当前任务的映射和调度得以完成。

3.3 划分问题与插入策略

3.3.1 划分问题动态求解

本文提出了一种针对划分问题的动态求解方案，即在任务调度和映射的同时确定 FPGA 的重构区域数量和大小。对于重构区域的数量，通常设置一个上限值，重构区域的大小则根据任务使用资源量的不同进行动态调整。为了方便描述，我们将 cur_{clb} 定义为当前 FPGA 剩余的 CLB 资源数量，将 M 定义为重构区域数量的上限值，clb_{num} 为任务执行所需的 CLB 资源数量。当任务映射到 FPGA 上执行时，FPGA 的划分问题包括以下两种情况：

(1) $cur_{clb}\geqslant clb_{num}$，FPGA 中的剩余可用资源满足任务的执行需求。此时若 FPGA 中的区域数量小于 M，则划分一块新的区域供任务执行，重构区域大小即为任务执行所需的资源量的大小；若 FPGA 中的区域数量等于 M，则映射到最早结束当前任务执行的区域，重构区域的大小调整为任务执行所需的资源量。

(2) $cur_{clb}<clb_{num}$，FPGA 中的剩余可用资源不满足任务的执行需求。此时需等待其他区域的任务执行完毕，直到 $cur_{clb}\geqslant clb_{num}$，将任务映射到重构区域执行，重构区域的大小调整为任务执行所需的资源量。

因此，对于 3.2 节式(6)中的重要参数 $valid(k)$，当 $k\in P$ 时求解较简单，$valid(k)$ 为当前 CPU 上最后一个任务执行的结束时间。当 $k\in H$ 时，$valid(k)$ 的求解与 DPR FPGA 的划分问题有关，可以建立如下函数关系：

$$\underset{k\in H}{valid(k)}=\begin{cases}DRT_{i,k}, & cur_{clb}\geqslant clb_{num}\\ finishtime, & cur_{clb}<clb_{num}\end{cases} \tag{8}$$

其中，当 $cur_{clb}<clb_{num}$ 时，需要等待 FPGA 其他区域上的任务执行完毕，直到满足资源约束条件时任务才能开始执行。因此，$finishtime$ 的大小为满足 $cur_{clb}\geqslant clb_{num}$ 条件时 FPGA 其他重构区域任务执行的结束时间。通过式(8)可以解决 DPR FPGA 的动态区域划分问题，实现了整个算法的闭环设计。

3.3.2 插入策略

本文采取插入策略进一步优化算法设计，减小 SL，提升

算法的求解性能。插入策略的示意图如图 4 所示。

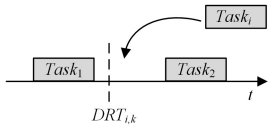


图 4　插入策略示意图

Fig. 4　Schematic diagram of insertion strategy

任务插入策略可以提高计算单元的时间复用性。若待部署的任务满足相应的限制条件，便可插入到已部署的两个任务之间的空闲时间段执行，从而提高计算资源的使用率。以图 4 为例，其限制条件为：

(1)$k\in P$ 时，若 $DRT_{i,k}$ 介于 $Task_1$ 的结束时间和 $Task_2$ 的开始时间之间，且 $Task_1$ 与 $Task_2$ 时间间隔大于 $Task_i$ 的 sw_t，即可将 $Task_i$ 映射到 P 上执行。

(2)$k\in H$ 时，除了满足(1)中 $DRT_{i,k}$ 的限制条件外，$Task_1$ 与 $Task_2$ 的时间间隔应大于 $Task_i$ 的 hw_t+rc_t 且在时间间隔内 FPGA 的 cur_{clb} 大于或等于 clb_{num}。

4　实验结果分析

4.1　实验参数设置

为了评估本文提出的软硬件划分算法的性能，我们使用 JPEG encoder，Parallel Gauss Elimination，LU decomposition，Parallel Tiled QR factorization，Gauss Elimination，Channel Equalizer，Gauss Jordan，Quadratic Equation Solver，TD-SCDMA，FFT，Laplace Equation，Parallel MVA，Ferret，Cyber Shake，Epigenomics，Montage，LIGO，WLAN 802.11a Receiver，MP3 Decoder Block Parallelism，SIPHT，Molecular Dynamics，Modem 等 22 个实际应用进行测试[23-24]。应用的节点数(规模)和依赖边的数量如表 2 所列。

为了避免特殊性和偶然性，本文选择的软硬件划分与调度的对比算法有两类：1)规划类算法中的 MILP 算法；2)启发式类算法中的 ACO 算法。同时，针对表 2 中的应用进行实验测试。其中 MILP 算法求解时间的 timeout 设置为 1 800 s，当 MILP 在求解时间达到 1 800 s 却没有找到全局最优解时，则返回当前已获得的最优解，MILP 算法中的求解器为 LINDO API 11.0[25]；ACO 算法[13]中蚂蚁数量设置为 5，迭代次数为 2 000，全局信息素挥发控制因子为 1，局部信息素挥发控制因子为 1，信息素挥发因子为 0.9。本文中的所有算法均采用 C++代码实现。

表 2　测试应用基本参数

Table 2　Basic parameters of test application

App Name	Task Num	Edge Num
JPEG encoder	8	9
Parallel Gauss Elimination	12	17
LU decomposition	14	19
Parallel Tiled QR factorization	14	21
Gauss Elimination	14	19
Channel Equalizer	14	21
Gauss Jordan	15	20
Quadratic Equation Solver	15	15
TD-SCDMA	16	20
FFT	16	20
Laplace Equation	16	24
Parallel MVA	16	24
Ferret	20	19
Cyber Shake	20	32
Epigenomics	20	22
Montage	20	30
LIGO	22	25
WLAN 802.11a Receiver	24	28
MP3 Decoder Block Parallelism	27	46
SIPHT	31	34
Molecular Dynamics	41	71
Modem	50	86

4.2　实验结果分析

表 3 列出了所有算法的仿真实验结果。在实验对象相同的前提下，本文将算法求解的调度长度 SL 和求解时间作为评价指标。为了方便分析不同算法的求解结果，引入调度长度变化比例 SLR 来衡量不同算法求解的调度长度相对差异。

表 3　仿真实验结果

Table 3　Simulation experiment results

App Name	Task Num	eHEFT		MILP			ACO		
		SL	Time	SL	Time	SLR/%	SL	Time	SLR/%
JPEG encoder	8	52	0.004	45	0.28	15.56	47	26.544	10.64
Parallel Gauss Elimination	12	52	0.006	50	2.27	4.00	53	47.35	−1.89
LU decomposition	14	82	0.007	81	157.4	1.23	84	54.757	−2.38
Parallel Tiled QR factorization	14	74	0.011	67	165.99	10.45	74	56.003	0.00
Gauss Elimination	14	70	0.006	69	12.79	1.45	70	55.039	0.00
Channel Equalizer	14	82	0.007	72	177.64	13.89	82	65.221	0.00
Gauss Jordan	15	65	0.007	72	timeout	−9.72	81	62.146	−19.75
Quadratic Equation Solver	15	48	0.012	53	timeout	−9.43	56	65.973	−14.29
TD-SCDMA	16	104	0.007	85	294.57	22.35	101	68.722	2.97
FFT	16	46	0.007	55	timeout	−16.36	59	66.264	−22.03
Laplace Equation	16	102	0.008	101	1 755.83	0.99	107	63.737	−4.67
Parallel MVA	16	76	0.007	73	timeout	4.11	83	64.413	−8.43
Ferret	20	70	0.01	76	timeout	−7.89	76	81.511	−7.89
Cyber Shake	20	76	0.01	108	timeout	−29.63	94	89.551	−19.15
Epigenomics	20	94	0.009	90	timeout	4.44	101	82.243	−6.93
Montage	20	115	0.01	113	timeout	1.77	129	86.203	−10.85
LIGO	22	86	0.01	108	timeout	−20.37	107	101.786	−19.63
WLAN 802.11a Receiver	24	173	0.012	172	timeout	0.58	175	98.373	−1.14
MP3 Decoder Block Parallelism	27	114	0.016	149	timeout	−23.49	138	124.125	−17.39
SIPHT	31	105	0.014	195	timeout	−46.15	129	160.727	−18.60
Molecular Dynamics	41	159	0.027	436	timeout	−63.53	190	235.283	−16.32
Modem	50	234	0.038	509	timeout	−54.03	249	304.263	−6.02

$$SLR=\frac{SL_{eHEFT}-SL_{contrast}}{SL_{contrast}}*100\% \tag{9}$$

其中，SL_{eHEFT} 为本文算法的求解结果，$SL_{contrast}$ 为对比算法的求解结果。若 SLR 等于 0 表示两种算法的求解结果相同，若 SLR 大于 0 表示本文算法的结果次于对比算法，若 SLR 小于 0 表示本文算法优于对比算法。

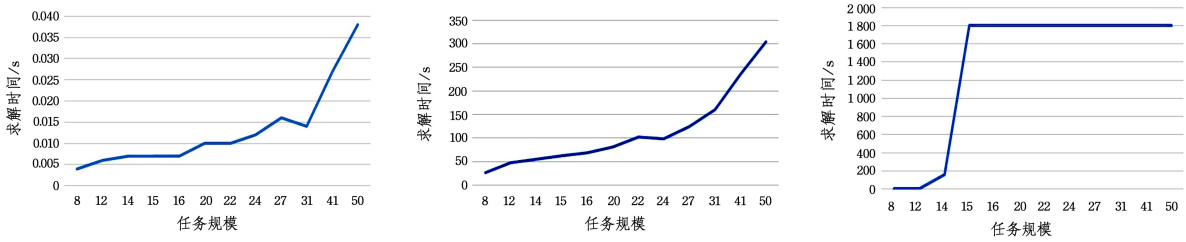
分析实验数据可得，本文所提算法在求解结果上优于 MILP 与 ACO 算法，平均性能优于 MILP 约 9.08%，优于 ACO 约 8.3%。从本文算法与 ACO 算法的对比实验中可以发现，本文算法除了在 JPEG encoder 和 TD-SCDMA 两个应用的求解结果上较差(平均 SLR 较小，约为 6%)外，在其他应用上均优于 ACO 算法且求解时间远远少于 ACO 算法，整体性能有较大优势。

在本文算法与 MILP 算法的对比实验中可以发现，应用规模小于 15 时，MILP 算法的求解结果优于本文算法；但应用规模大于 15 时本文算法的求解优势开始显现，且随着任务规模的增大，本文算法的优势逐渐明显，当任务规模到达 30 左右时，本文算法性能已经优于 MILP 约 50%。实验中，在求解时间上，本文算法耗时最大不超过 0.04 s，MILP 最小求解时间为 0.28 s 且在任务规模大于 15 时求解已经超时，达到了 1 800 s。

总的来说，在算法的求解性能方面，当任务规模较小时，MILP 算法＞本文算法＞ACO 算法；但随着任务规模的增大，本文算法的求解结果均优于 MILP 和 ACO 算法。在算法的求解时间方面，MILP 算法与 ACO 算法的求解时间均随任务规模的增大而急剧增加，本文算法远远优于对比算法。可以看出，在面临中大规模应用时，本文算法在求解性能和求解时间上均有较大的优势，更满足实际情况中的应用需求。

本文算法的时间复杂度主要由 3 个部分构成：构建任务优先级列表、任务调度和映射、FPGA 的划分。本文算法的时间复杂度约为 $O(v^2\times p)$，ACO 算法的时间复杂度约为 $O(v^2\times N\times m\times p)$，其中 v 为任务的数量，p 为计算单元的数量，N 为迭代次数，m 为蚂蚁数量。MILP 算法的时间复杂度与约束方程和建模方式有关，且呈指数级增长。可以看出，随着任务规模的增大，MILP 的时间复杂度最高，增长速率和增长幅度都最大，ACO 算法与本文算法的时间复杂度、增长速率基本相同，但 ACO 算法的增长幅度远远大于本文算法。

为了更加明显地看出算法求解时间的变化趋势，图 5 分析了本文算法、ACO 算法和 MILP 算法在面对不同规模应用时，求解时间与任务数量的变化关系。由图5(a)和图 5(b)可以明显看出，在算法的求解时间增长速率方面，本文算法与 ACO 算法基本相同，但在增长幅度方面，ACO 算法约为本文算法的 1 000 倍。图 5(c)中，MILP 算法求解十几个任务的时间已经超时(大于 1 800 s)，远远多于本文算法的求解时间。综上分析，在算法时间复杂度方面，本文算法表现更好。



(a)本文算法 (b)ACO (c)MILP

图 5 任务规模与求解时间的变化关系

Fig. 5 Relationship between task scale and solution time

结束语 本文提出了基于贪心策略的列表式软硬件划分算法，详细介绍了应用建模、计算平台建模和所提算法的流程，并采用了一组实际应用对所提算法进行性能评估。实验结果表明，在平均求解性能和时间复杂度上本文所提算法均优于对比算法，且随着任务规模的增大，所提算法的优势明显增加，在云计算、大数据等领域具有更加广阔的应用前景。在下一步的研究中，考虑将本文算法与元启发式算法相结合，进一步优化求解性能。

参 考 文 献

[1] ZHU L H, WANG L, TANG Q, et al. Efficient MILP Model for HW/SW Partitioning of Dynamic Partial Reconfigurable SoC [J]. Computer Science, 2020, 47(4): 18-24.

[2] BANERJEE S, BOZORGZADEH E, DUTT N D. Integrating physical constraints in HW-SW partitioning for architectures with partial dynamic reconfiguration[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2006, 14(11): 1189-1202.

[3] CHAKRABORTTY R K, SARKER R A, ESSAM D L. A Comparative Study of Different Integer Linear Programming Approaches for Resource Constrained Project Scheduling Problems [M]. Lecture Notes in Management and Industrial Engineering, 2018.

[4] REDAELLI F, SANTAMBROGIO M D, OGRENCI M S. An ILP Formulation for the Task Graph Scheduling Problem Tailored to Bi-dimensional Reconfigurable Architectures[C]// International Conference on Reconfigurable Computing and Fpgas, 2008. IEEE, 2008: 97-102.

[5] TOPCUOGLU H, HARIRI S, WU M Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing[J/OL]. IEEE Transactions on Parallel and Distributed Systems. https://ieeexplore.ieee.org/document/993206.

[6] WANG Z, TANG Q, WANG L, et al. A Joint Optimization Algorithm for Partition-Scheduling of Dynamic Partial Reconfigurable Systems Based on Simulated Annealing [J]. Computer Science, 2020, 47(8): 26-31.

[7] CHEN S,HUANG J,XU X,et al. Integrated Optimization of Partitioning,Scheduling,and Floorplanning for Partially Dynamically Reconfigurable Systems[J]. Physical Review B,2018,17(10):4114-4116.

[8] HOU E S H,ANSARI N. A genetic algorithm for multiprocessor scheduling[J]. IEEE Transactions on Parallel and Distributed Systems,1994,5(2):113-120.

[9] ABDALLAH F,TANOUGAST C,KACEM I,et al. Genetic algorithms for scheduling in a CPU/FPGA architecture with heterogeneous communication delays[J/OL]. Computers & Industrial Engineering. https://www.sciencedirect.com/science/article/abs/pii/S0360835219304644.

[10] HOU N,HE F Z. Hybrid parallel genetic strategy with two-step adjustment for HW/SW partitioning[J]. Huazhong Univ. of Sci. & Tech. (Natural Science Edition),2017,45(12):39-45.

[11] CATTANEO R,BELLINI R,DURELLI G,et al. PaRA-Sched:A Reconfiguration-Aware Scheduler for Reconfigurable Architectures[C]// 2014 IEEE International Parallel & Distributed Processing Symposium Workshops (IPDPSW). IEEE,2014.

[12] TUMEO A,PILATO C,FERRANDI F,et al. Ant colony optimization for mapping and scheduling in heterogeneous multiprocessor systems[C]// Embedded Computer Systems:Architectures,Modeling,and Simulation. IEEE,2008.

[13] FERRANDI F,LANZI P L,PILATO C,et al. Ant Colony Optimization for mapping,scheduling and placing in reconfigurable systems[C]//2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS). IEEE,2013.

[14] BRUCKER P,DREXL A,MOEHRI R,et al. Resource-constrained project scheduling:Notation,classification,models,and methods[J]. European Journal of Operational Research,1999,112(1):3-41.

[15] DHAR A,YU M,ZUO W,et al. Leveraging Dynamic Partial Reconfiguration with Scalable ILP Based Task Scheduling[C]// 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID). 2020.

[16] TANG Q,WU S F,SHI J W,et al. Optimization of Duplication-Based Schedules on Network-on-Chip Based Multi-Processor System-on-Chips[J]. IEEE Transactions on Parallel & Distributed Systems,2017(3):1-1.

[17] BIONDI A,BALSINI A,PAGANI M,et al. A Framework for Supporting Real-Time Applications on Dynamic Reconfigurable FPGAs[C]// 2016 IEEE Real-Time Systems Symposium (RTSS). IEEE,2016.

[18] MA Y,LIU J,ZHANG C,et al. HW/SW partitioning for region-based dynamic partial reconfigurable FPGAs[C]// IEEE International Conference on Computer Design. IEEE,2014:470-476.

[19] VIPIN K,FAHMY S A. FPGA Dynamic and Partial Reconfiguration:A Survey of Architectures,Methods,and Applications[J]. ACM Computing Surveys,2018,51(4):1-39.

[20] XIAO X,LI Z. Chemical Reaction Multi-Objective Optimization for Cloud Task DAG Scheduling[J]. IEEE Access,2019(99):1-1.

[21] XU J R,ZHU H J. Multi-objective scheduling algorithm of DAG tasks in cloud computing[J]. Application Research of Computers. http://en.cnki.com.cn/Article_en/CJFDTotal-JSYJ201901008.htm.

[22] STUIJK S,GEILEN M,BASTEN T. SDF3:SDF For Free[C]// 6th International Conference on Application of Concurrency to System Design,ACSD 2006,Proceedings. IEEE,2006:276-278.

[23] TANG Q,BASTEN T,GEILEN M,et al. Mapping of synchronous dataflow graphs on MPSoCs based on parallelism enhancement[J]. Journal of Parallel & Distributed Computing,2017,101(MAR.):79-91.

[24] CANON L C,SAYAH M E,PIERRE-CYRILLE H. A Comparison of Random Task Graph Generation Methods for Scheduling Problems[J]. arXiv:1902.05808,2019.

[25] LINDO API 11[EB/OL]. https://www.lindo.com/index.php/products/lindo-api-for-custom-optimiz.

GUO Biao, born in 1995, postgraduate. His main research interests include software defined radio and dynamic partially reconfiguration technology of FPGA.

TANG Qi, born in 1986, Ph.D, assistant professor. His main research interests include reconfigurable computing, embedded parallel computing, algorithm designand software defined system.