计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200500110

一种基于模拟退火的动态部分可重构系统划分-调度联合优化算法

王 喆[1,2] 唐 麒[2] 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
(wangzhe_hnu@qq.com)

摘 要 基于FPGA的动态部分可重构(Dynamically Partially Reconfigurable,DPR)技术因在处理效率、功耗等方面具有优势,在高性能计算领域得到广泛应用。DPR系统中的重构区域划分和任务调度决定了整个系统的性能,因此如何对DPR系统的逻辑资源划分和调度问题进行建模,并设计高效的求解算法是保证系统性能的关键。在建立划分和调度模型的基础上,设计了基于模拟退火(Simulated Annealing,SA)的DPR系统划分-调度联合优化算法,用于优化重构区域的划分方案和任务调度。文中提出了一种新型新解产生方法,可有效跳过不可行解及较差解,加快了对解空间的搜索并提高了算法的收敛速度。实验结果表明,与混合整数线性规划(Mixed Integral Linear Programming,MILP)和IS-k两种算法相比,提出的基于SA的算法的时间复杂度更低;且针对大规模应用,该算法能够在较短的时间内获得较好的划分与调度结果。

关键词:动态部分可重构;模拟退火;调度;划分;FPGA
中图法分类号 TP302

Joint Optimization Algorithm for Partition-Scheduling of Dynamic Partial Reconfigurable Systems Based on Simulated Annealing

WANG Zhe[1,2],TANG Qi[2],WANG Ling[1] and WEI Ji-bo[2]
1 College of Electrical and Information Engineering,Hunan University,Changsha 410082,China
2 College of Electronic Science and Technology,National University of Defense Technology,Changsha 410073,China

Abstract Dynamically partially reconfigurable (DPR) technology based on FPGA has many applications in the field of high-performance computing because of its advantages in processing efficiency and power consumption. In the DPR system,the partition of the reconfigurable region and task scheduling determine the performance of the entire system. Therefore,how to model the logic resource partition and scheduling of the DPR system and devising an efficient solution algorithm are the keys to ensure the performance of the system. Based on the establishment of the partition and scheduling model,a joint optimization algorithm of DPR system partition-scheduling based on simulated annealing (SA) is designed to optimize the reconfigurable region partitioning and task scheduling. A new method is proposed for skip infeasible solutions and poor solutions effectively which accelerates the search of solution space and increases the convergence speed of the SA algorithm. Experimental results show that,compared with mixed integer linear programming (MILP) and IS-k,the proposed algorithm based on SA has lower time complexity,and for the large-scale applications,it can solve better partition and scheduling results in a short time.

Keywords Dynamically partially reconfigurable,Simulated annealing,Scheduling,Partition,FPGA

1 引言

随着摩尔定律的终结,IC晶体管集成密度逐渐趋于临界值,短期内CPU的处理能力难以得到更大的提升。FPGA具有高性能、低能耗及可硬件编程的特点。FPGA并行处理的特性使其能够更快地执行任务,对于同一个计算任务,其处理速度一般比CPU快几十倍。因此,当前的FPGA架构能够在不同时刻动态地重新配置器件上的部分硬件资源,同时不影响其余逻辑电路的工作状态,从而实现器件上的不同逻辑电路区域各自独立工作,我们将其称为动态部分可重构[1]。DPR技术通过在FPGA上划分多个可重构区域,在同一时间分别执行不同任务,达到对FPGA资源在时间和空间上的复用。由于动态时分复用,有限的空间资源可在时域无限扩展,可重构系统能够在有限资源的片上系统中执行更大资源需求的计算应用。该技术能够将FPGA划分为多个可重构区域,使不同区域之间的执行过程互不影响,从而实现任务的并行处理。随着5G时代的到来,以及机器学习的广泛应用,CPU的处理能力面临巨大考验。FPGA的并行能力和可重构能力以及相比DSP,GPU等处理器功耗低的特点,使得它在高性能计算领域有着巨大的作用。

动态部分可重构技术提供了在应用运行期间更改硬件配置的能力。基于SRAM的FPGA,允许通过加载部分配置文

到稿日期:2020-05-22 返修日期:2020-07-01 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:王玲(wl_hunu@163.com)

件来修改 FPGA 的配置，从而在运行期间在不影响其他区域的执行、不损害应用程序的完整性的情况下，改变部分重构区域的配置。

动态部分可重构系统的划分与调度问题为 NP 问题，求解 NP 问题的方法主要包括构建 ILP/MILP 模型、分支定界、A* 算法、启发式搜索方法等。ILP 的求解时间较长，求解效率较低。启发式方法一般只能求解近似最优解，但是求解时间短，如果设计得当，得到的近似最优解与全局最优解近似，同时具有很强的可扩展性。

文献[2]研究了在部分可重构 FPGA 上周期性硬实时任务的调度，其假定 FPGA 静态划分为一组具有相同大小的矩形区域，并且可以将任何任务映射到每个矩形区域。文献[3]提出了基于 MILP 方法的异构系统的划分与调度问题，在 2 维架构 FPGA 器件上优化了 DPR 系统的功耗和调度长度。文献[4-5]提出了基于 MILP 模型的布局算法，并在重构区域上对布局做出了优化。文献[6]对文献[7]进行了改进，通过预分簇建立 MILP 模型，求解出任务在软硬件的映射方案，极大地降低了模型的复杂度，提高了求解结果与实际应用的符合度。文献[8]提出了一种基于遗传算法的多处理器流水线内存感知调度方法，利用启发式方法的高效性，在不缩短应用运行时间的情况下提高吞吐量。文献[9]提出的运行时间系统管理器 RTSM 用于在可用处理器和 FPGA 的可重构区域上调度任务。RTSM 采用任务重用策略来最小化重构开销，在区域之间移动任务以有效地管理 FPGA 区域，为将来的重构和执行操作保留任务，并支持配置预取。文献[10]提出的遗传算法集成了针对布局问题的本地搜索策略，能够优化不同的度量标准。文献[11]对重构区域进行了模块化划分，对重构时间和区域面积进行了联合优化，缩短了重构时间并减少了面积开销，但没有综合考虑整个应用的运行时间。文献[12]提出了一种蚁群优化方法，用于通过一维 DPRFPGA 和一个或多个指令处理器（例如微处理器或 DSP）在 SoC 上映射、调度和放置 DAG。

本文针对 DPR 系统应用调度和划分问题，提出了调度和划分模型，设计了基于模拟退火算法的求解方法对模型进行求解。通过降低模型复杂度来加快对调度和划分方案的求解，在有效缩短求解时间的同时，还可获得近似最优解。实验结果表明，本文所提算法能够高效求解出划分和调度方案，在较短时间内得到近似最优解。

2　问题描述

2.1　平台模型

本文面向基于动态部分可重构的 FPGA 的硬件平台，采用 Xilinx 的 Zynq 系列硬件平台模型，如图 1 所示。该平台由一个具有二维 DPR 能力的 FPGA 和一个控制 FPGA 的微处理器组成，FPGA 可以被虚拟化为一个静态区域和多个动态部分可重构区域[1]。内部配置访问端口（Internal Configuration Access Port，ICAP）是可以直接读写 FPGA 内部配置寄存器的专用接口。微控制器通过 ICAP 将用于配置重构区域的比特流文件加载到 FPGA，从而对动态部分可重构系统进行配置。由于硬件平台的限制，动态部分可重构系统的划分与调度过程需要遵循如下约束条件：1）重构区域的配置只能串行执行，单个时间点只能有一个重构区域被重构；2）任务在重构区域重构之后才可执行；3）某任务的执行须在所依赖的数据抵达后才可开始。

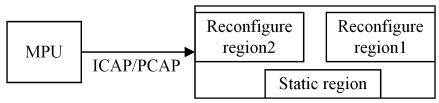

图 1　平台示意图

Fig. 1　Platform diagram

任务与重构区域之间需要建立映射关系，任务被映射到重构区域之后，需要对重构区域的面积进行划分。本文考虑了实际应用的资源需求和 FPGA 的大小之间的资源约束关系，同时对异构资源的种类及相应的重构时间进行了划分。具有动态部分可重构能力的 FPGA 需要划分出多个动态部分可重构区域 $PR=\{PR_0,PR_1,\cdots,PR_r\}$，各重构区域包含多种资源，包括 CLB（Configurable Logic Block），DSP（Digital Signal Processing），BRAM（BlockRAM）等。重构区域 PR_i 的重构时间 RT_i 与其资源量成正比，可表示为各类型资源重构时间的线性叠加，即 $RT=k_1*BRAM+k_2*DSP+k_3*CLB$（$k$ 为单位资源的重构时间）。

2.2　应用模型

本文采用 DAG 对应用进行建模，DAG 可表示为 $G(V,E)$，$V=\{n_0,n_1,\cdots,n_{|V|-1}\}$ 表示 DAG 包含的任务集，任务 n_i 包含两个属性：资源占用量和执行时间。有向边集 $E=\{e_0,e_1,\cdots,e_{|E|-1}\}$ 表示任务之间的数据依赖关系[13]。

任务 n_i 的父任务为其所需依赖数据的任务，表示为 $PN=\{pn_1,pn_2,\cdots,pn_a\}$；任务 n_i 的子任务为其生成数据的目标任务，表示为 $CN=\{cn_1,cn_2,\cdots,cn_b\}$。为简单起见，我们引入虚拟汇点 n_s 作为 DAG 中所有出度为 0 的任务的子任务，只包含重构顺序属性，n_s 的重构顺序为父任务集中的最大重构顺序加 1，表示依赖关系的有向边指向 n_s。

图 2 给出了一个示例应用的 DAG 模型，该 DAG 模型包含 8 个任务、9 条有向边。表 1 列出了任务的参数，包括任务在 FPGA 的执行时间 ET，每个任务所需的 CLB 资源数量、父任务集 PN、子任务集 CN。

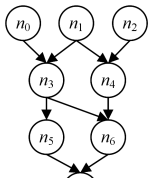

图 2　DAG 样例

Fig. 2　DAG diagram

表 1　DAG 参数

Table 1　DAG parameter

Task Name	ET	CLB Num	PN	CN
n_0	243	231	—	$\{n_3\}$
n_1	175	532	—	$\{n_3,n_4\}$
n_2	154	287	—	$\{n_4\}$
n_3	146	357	$\{n_0,n_1\}$	$\{n_5,n_6\}$
n_4	139	532	$\{n_1,n_2\}$	$\{n_5\}$
n_5	199	91	$\{n_3\}$	$\{n_7\}$
n_6	256	406	$\{n_3,n_4\}$	$\{n_7\}$
n_7	17	14	$\{n_5,n_6\}$	$\{n_s\}$

2.3 问题描述

划分与调度问题本质上是 NP 问题，二者互相影响。为更加详细地描述划分与调度问题，将该问题分解为几个子问题：虚拟化 FPGA 为多个可重构区域并确定可重构区域的数量和大小、任务的调度顺序和目标函数，进而对解进行量化。

将应用的任务会映射到 FPGA，需要确定重构区域的数量和每个重构区域大小。通常对重构数量设定一个上限值，在上限值内确定划分与调度的最优解，而每个重构区域的大小则要满足映射到该区域的任务的最大资源需求。在 DPR 系统中，任务映射到重构区域之后对应一个重构节点，每个重构节点的生命周期共分为两个阶段：重构阶段和执行阶段。单个重构节点对应于唯一的任务，具有多种属性，包括重构开始时间 Rs、重构结束时间 Re、执行开始时间 Es、执行结束时间 Ee。其中，重构结束时间为重构开始时间与重构节点包含任务的重构时间跨度之和，即 $Re=Rs+RT$，执行结束时间为执行开始时间与执行时间跨度之和，即 $Ee=Es+ET$。

重构区域的资源与任务的资源需求需要满足以下约束关系：1）单个重构区域的资源种类及资源量必须满足本重构区域上最大重构节点对资源的需求；2）各个重构区域的资源量总和不超过 FPGA 资源的总量。

由于硬件平台的约束，重构过程只能串行完成。不同重构节点对应于唯一的重构顺序，所有任务在重构区域上的重构顺序表示为 $order=[\cdots,n_i^h,\cdots,n_j^h,\cdots](i\neq j,0\leqslant h\leqslant|V|-1)$，每个元素在 $order$ 中从左往右依次递增，h 值为其重构顺序。

调度长度是同构或异构高性能计算系统中的重要参数，对于同一个计算应用而言，调度长度越短，系统的计算效率就越高。因此，为了量化划分与调度问题的解，我们定义一个目标函数 $SL=\max(Ee)-\min(Rs)$（其中，$\max(Ee)$为最后一个重构节点的执行结束时间，$\min(Rs)$为首个重构节点的重构开始时间）[14]，表示在可重构器件上整个应用从开始到结束运行的时间跨度，本文将 SL 作为衡量解的指标之一。

3 划分与调度算法

3.1 划分与调度流程

DPR 系统的划分和调度问题是将给定应用的 DAG 中的子任务按一定的重构顺序映射到重构区域，并确定重构区域的数量和大小。在满足平台约束和资源约束的前提下，本文设计了基于 SA 算法的 DPR 系统划分与调度的过程。

（1）求解出任务的重构顺序和任务到重构区域的映射关系，根据映射关系确定每个重构区域包含的资源类型及其相应的资源量。

（2）在已知每个任务在 FPGA 上的执行耗时以及每个重构区域大小的基础上求解调度结果，调度结果包括任务开始执行时间、开始重构时间以及 SL。

不同算法会有不同的求解过程，但最终都是求出可行的划分与调度方案，SL 是衡量一个可行解优劣的重要参数，SL 越小，解的质量越优，反之解的质量越差。在保证重构区域得到有效划分的情况下，以最小化应用 SL 为目标，其对应的解为所求最终解。在比较不同算法的性能时，SL 同样是比较的关键指标。为了更加准确地衡量不同算法的求解效率，求解时间也要作为衡量算法性能的关键指标。

3.2 算法结构

SA 算法是一种应用广泛的随机寻优算法，可以分解为解空间、目标函数、初始解 3 个部分。SA 算法在搜索解空间的过程中，可以将每个新解的目标函数值与当前解做比较，并以一定概率接受一个比当前解差的解，从而跳出局部最优得到全局最优解，其算法流程如图 3 所示。本文设计的基于 SA 的动态部分可重构系统划分-调度联合优化算法的求解过程如下：首先随机产生一个初始解，然后计算目标函数，产生新解，随后进行迭代优化。每次迭代产生新解时，只对前一次接受的解做出扰动。

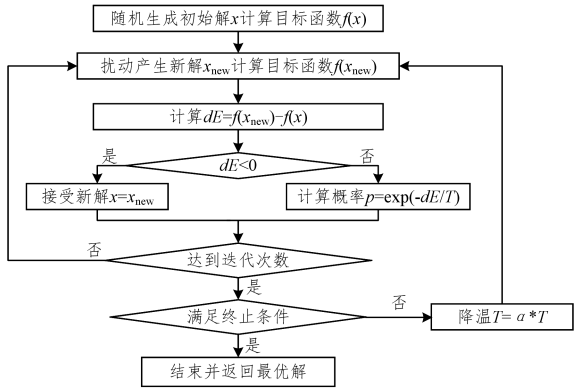

图 3 算法流程图

Fig. 3 Algorithm flow chart

在产生新解时，本文采用先删除后插入的方法，即对一个选定的任务所在的重构区域和重构顺序做变换。例如，选择一个待删除任务 n_{delete}，然后在重构顺序中将此任务插入到另一个任务 n_{insert} 之前，插入 n_{delete} 之后其在 $order$ 中的顺序为新的重构顺序。对 n_{delete} 选择新重构区域 PR_{new}，在原重构区域 PR_{old} 中删除 n_{delete} 并将结果插入 PR_{new}，具体步骤如下：

（1）随机选择一个待删除任务 n_{delete}；

（2）选择一个待插入任务 n_{insert} 和新重构区域 PR_{new}；

（3）在重构区域 PR_{old} 中删除 n_{delete}，在 PR_{new} 中插入 n_{delete}，在重构序列 $order$ 中将 n_{delete} 插入到 n_{insert} 之前；

（4）产生新解，计算目标函数值 SL，以 Metropolis 准则接受新解，继续模拟退火其他迭代步骤。

3.3 解的结构

本文所研究问题的解包含多个部分：任务到重构区域的映射、重构区域数量、重构区域大小、重构顺序。当完成求解之后，每个任务会被映射到重构区域，按照重构顺序对重构区域进行重构。在求解过程中算法需要对解空间进行搜索，如果重构顺序和任务与重构区域的映射不得当，则会导致任务在执行阶段的数据依赖关系无法满足，最终无法完成整个应用的执行。因此，每次迭代求解出的解可能是可行解或不可行解。下面将对解的可行性进行分析，并在算法中做出改进。

为了清楚地描述解的可行性问题，本文给出如下定义。

定义 1 任务 n_i 与其任意一个子任务或后代任务 n_{ci} 处于同一重构区域，且 n_i 的重构发生在 n_{ci} 之后，即为逆序。

定义 2 任务 n_i 与其所有子任务 n_{ci} 处于不同重构区域或 n_i 与其子任务处于同一重构区域时，n_i 的重构发生于其同

一重构区域的子任务之前，即为正序。

定义 3　包含逆序的解称为不可行解，不包含逆序的解称为可行解。

定义 4　在可行解中，重构顺序不满足所有任务的 $order_{pn}<order_{cn}$ 的解称为较差解。

如图 4 所示，DAG 重构序列为 $order=\{n_3^0,n_2^1,n_1^2,n_0^3\}$，图 5 给出了任务与重构区域的映射关系：$PR_0=\{n_3,n_0\}$，$PR_1=\{n_2\}$，$PR_1=\{n_1\}$。假设重构区域的资源满足任务的需求，根据数据的依赖关系，n_3 在执行时需要 n_1，n_2 执行完成后产生的数据，而 n_1，n_2 又依赖于 n_0 执行后产生的数据，因此无论 n_0 在 A，B，C 哪一个位置都无法给 n_1，n_2 传递所需的数据。因为只有在重构区域完成了正在进行的重构和执行阶段之后才可以开始 n_0 的重构阶段，而当前重构区域已经有其他任务等待执行，所以无法为 n_0 配置可重构区域，也就无法执行 n_0，从而无法产生 n_1，n_2 的依赖数据，此重构顺序称为逆序，对应的解称为不可行解。

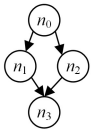

图 4　DAG 样例

Fig. 4　Example DAG

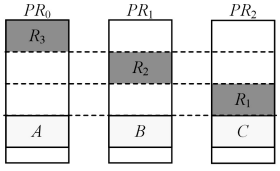

图 5　不可行解示意图

Fig. 5　Infeasible solution diagram

若将重构顺序改为 $order=\{n_3^0,n_0^1,n_1^2,n_2^3\}$，如图 6 所示，$n_0$ 虽然在 n_3 后发生重构，但 n_3 会在其重构完成后等待依赖数据，暂不进行执行阶段，n_0 所在 PR_1 的重构、执行操作相继完成后，n_1，n_2 可分别在 PR_2，PR_1 完成重构、执行阶段，将数据传递给 PR_0 的 n_3，因此可以在 n_1，n_2 执行完毕后，n_3 开始执行，此重构顺序为正序，但由于 n_3 的执行需要较长时间的等待，且在等待期间不能发生重构。在时间上，没能充分利用 DPR 系统的执行时间对重构时间的隐藏特性[15]，故此解为质量较差解。

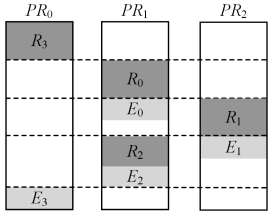

图 6　可行解示意图

Fig. 6　Feasible solution diagram

3.4　构建邻域解集

实验表明，在选择待插入任务 n_{insert} 时，随机选择 n_{insert} 产生的解收敛很慢且质量较差，产生的大量新解中会包含很多不可行解和较差解。除此之外，随机选择待插入任务还会影响最优解的质量，导致调度长度较大。因此，我们对删除插入算法的步骤 2 进行改进，以提高算法的有效性。根据 n_{delete} 的数据依赖关系，在其父任务集 PN 中找出 n_{delete} 的重构顺序值最大的父任务 pn，其重构顺序为 $order_{pn}$；在子任务集 CN 中找出 n_{delete} 的重构顺序值最小的子任务 cn，其重构顺序为 $order_{cn}$。$(order_{pn},order_{cn}]$ 区间为 n_{delete} 待插入点的正序，在此区间内选择待插入点以避免产生不可行解，同时也有效跳过了较差解。

引入的虚拟汇点 n_s 可使得 DAG 所有任务的待插入区间 $(order_{pn},order_{cn}]$ 有意义。区间内的值作为插入点候选集，每个 $order_i$ 对应的任务 n_i 作为待插入任务，将 n_{delete} 插入并计算调度长度，生成一个调度长度集，遍历调度长度集，以 Metropolis 准则返回比最小 SL 大的插入点，若均没有选中，则返回最小调度长度。算法具体描述如下：

(1)根据 $(order_{pn},order_{cn}]$ 产生待插入点集；

(2)将 n_{delete} 逐个插入并计算调度长度，得到调度长度集；

(3)遍历调度长度集，以 Metropolis 准则返回比最小 SL 大的插入点；

(4)否则，返回最小值。

返回值返回到主程序后，进行模拟退火算法的目标值选择部分。

4　实验结果与分析

本文采用一组测试应用[16-17]（具体参数如表 2 所列），对基于 SA 的划分与调度算法，MILP，IS-k 算法进行性能评估，利用 GUROBI[1] 对 MILP，IS-k 进行求解，以 SL 和求解时间为评价指标，对上述方法进行对比。

表 2　应用参数

Table 2　Application parameters

Application Name	TaskNum	EdgeNum
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

4.1　实验设置

本文提出的基于 SA 的求解方法兼容所有资源类型，为了便于分析，以 CLB 代表所有资源，以单位 CLB 重构时间为 1 个周期，因此 RT＝CLB。对于重构区域数量的划分，如果划分出的区域太多，会导致划分出的可重构区域粒度过细，使每个区域的资源量有限而无法执行应用，因此需要限定重构区域数量以划分出区域面积较大的重构区域，以满足应用的执行资源需求。本实验设置重构区域数量的上限为 3。

实验选择的任务规模不尽相同，$|V|\in[8,50]$，由于 FPGA 的总资源量与任务总资源量的大小关系决定着整个可重

构系统是否能够完成，为保证仿真与实际场景相符又不失一般性，我们设置 $resourceRatio=(\sum_{i=0}^{|V|-1} n_i\_CLB)/FPGA\_CLB=0.4$。本实验设置初始温度为 200，终止温度为 0.01，降温系数为 0.98，内循环迭代次数为 10。

4.2 性能分析

本文采用算法求解时间及 *SL* 作为算法性能指标。由于 MILP 和 IS-k 均是基于 ILP 的算法，对于任务规模较大的 DAG 有较高的时间复杂度，同时为便于在相同时间条件下比较各算法的求解效率，为求解器设置了求解时间上限 *Timeout*，当求解时间超过 *Timeout* 时终止求解过程，并把当前获得的最优调度长度作为返回值。

不同 DAG 求解出的 *SL* 不同，为了更加清晰地表示不同方法求解出的 *SL* 的相对大小，引入性能提升比例(Performance Improvement Ratio，PIR)来表示 *SL* 的相对差值：

$$PIR=\frac{SL_{\mathrm{SA}}-SL_{\mathrm{other}}}{SL_{\mathrm{other}}}$$

当 $PIR=0$ 时，基于 *SA* 的算法与其他算法求解出的调度长度相同；当 $PIR<0$ 时，基于 *SA* 的算法求解出的调度长度优于其他算法，$|PIR|$ 表示 SL_{SA} 优于 SL_{other} 的差值占 SL_{other} 的百分比；当 $PIR>0$ 时，基于 *SA* 的算法求解出的调度长度差于其他算法，$|PIR|$ 表示 SL_{SA} 差于 SL_{other} 的差值占 SL_{other} 的百分比。

为了比较相同时间内不同方法的求解结果，将 *Timeout* 设置为基于 *SA* 的算法的求解时间。通过比较表 3 中 *Timeout*=SA Time 的数据可以得知，在相同时间内，MILP *PIR* 中只有 JPEG Encoder 的性能提升比例为 0，IS-k *PIR* 中有 JPEGEncoder 和 Parallel Tiled QR factorization 的性能提升比例为 0，即 MILP 与 IS-k 求解出的 *SL* 与基于 *SA* 的算法相同；其余 App 的 *PIR* 均为负值，相差比例很大，其中 MILP PIR 的 Ferret 达到 60.3%，IS-k *PIR* 的 Quadratic Equation Solver 达到 29.1%，基于 *SA* 的算法求解得到的 *SL* 全部优于其他两种算法。

表 3 实验结果

Table 3 Experiment result

Application Name	SA Time	Timeout=SA Time				Timeout=1 800 s			
		MILP *PIR*/%	MILP Time	IS-k *PIR*/%	IS-k Time	MILP *PIR*/%	MILP Time	IS-k *PIR*/%	IS-k Time
JPEG encoder	0.639	0.0	0.632	0.0	0.360	0.0	0.39	0.0	0.360
Parallel Gauss Elimination	2.239	−22.0	timeout	−1.9	2.230	0.0	29.18	−1.9	2.100
LU decomposition	3.695	−15.2	timeout	−2.3	timeout	0.0	timeout	0.0	5.760
Parallel Tiled QR factorization	1.603	−26.4	timeout	0.0	timeout	0.0	1 109.05	−4.3	10.120
Gauss Elimination	1.413	−7.0	timeout	−5.3	timeout	0.0	timeout	0.0	7.420
Channel Equalizer	0.303	−54.5	timeout	−9.5	timeout	2.2	102.88	2.3	3.150
Gauss Jordan	0.858	−21.6	timeout	−14.9	timeout	0.0	timeout	−9.6	40.430
Quadratic Equation Solver	0.950	−40.8	timeout	−29.1	timeout	1.3	timeout	−0.3	4.490
TD-SCDMA	1.835	−10.3	timeout	−4.1	timeout	2.2	timeout	0.9	32.240
FFT	1.974	−19.2	timeout	−9.1	timeout	0.0	timeout	−0.6	115.390
Laplace Equation	1.992	−13.3	timeout	−5.7	timeout	0.7	timeout	−0.9	16.730
Parallel MVA	1.930	−21.8	timeout	−9.8	timeout	0.0	timeout	−4.8	56.640
Ferret	2.005	−60.3	timeout	−17.1	timeout	−1.5	timeout	−3.9	50.600
Cyber Shake	5.401	−37.7	timeout	−10.9	timeout	0.0	timeout	−1.4	1 475.250
Epigenomics	2.932	−47.5	timeout	−14.8	timeout	−0.1	timeout	−5.3	117.740
Montage	1.886	−48.3	timeout	−12.4	timeout	0.0	timeout	−1.5	43.830
LIGO	2.986	−40.4	timeout	−13.1	timeout	0.0	timeout	0.2	40.900
WLAN 802.11a Receiver	2.721	−27.6	timeout	−7.8	timeout	−1.8	timeout	−1.6	70.630
MP3 Decoder Block Parallelism	10.011	−36.8	timeout	−8.7	timeout	−1.3	timeout	−0.1	131.320
SIPHT	8.733	−43.4	timeout	−5.0	timeout	−0.4	timeout	−2.0	timeout
Molecular Dynamics	10.400	−40.6	timeout	−9.2	timeout	−8.7	timeout	−7.4	timeout
Modem	5.504	−58.5	timeout	−3.8	timeout	−34.9	timeout	0.7	268.300

为了进一步分析基于 SA 的算法的性能，我们将求解器的求解时间上限设置为 Timeout=1 800 s。从表 3 中 Timeout=1 800 s 条件下的数据可以看出，任务数量小于 20 的应用中，只有 4 个应用的 MILP 求得的 *SL* 优于基于 SA 的算法，*PIR* 为 2%左右，*SL* 差值较小；对于其余应用，基于 SA 的算法与 MILP 求得的 *SL* 均相等；在 IS-k 的求解结果中，虽然有 2 个应用的 *SL* 优于基于 SA 的算法，但百分比均较小，对应的 *PIR* 分别为 2.3%，0.9%；其余应用的 *SL* 等于或劣于基于 SA 的算法的求解结果。对于任务数量大于 20 个的应用，MILP 求解的全部应用在其求解过程中均超时，IS-k 的部分求解过程超时，在表格中求解时间记为 Timeout；然而，基于 SA 的算法在较短的时间内求解得到了更优的 *SL*，*SL* 全部优于或等于 MILP，IS-k 求解得到的 *SL*；此外，任务规模越大，越能体现出基于 SA 的算法的性能优势，MILP 的求解时间大约为基于 SA 的算法的几十倍。部分应用的 *PIR* 较大，与 MILP 的求解结果相比，基于 SA 的算法求解 Molecular Dynamics 的调度长度的性能提升比例为 8.7%，Modem 的性能提升比例为 34%。

结束语 本文提出了一种基于模拟退火算法的动态部分可重构系统划分-调度联合优化算法，介绍了从分析模型到设计算法的整个流程，最后通过仿真实验验证了模型和算法的高效性。随着任务数量的增多，该算法的求解时间同样受到

[1] http://www.gurobi.com

任务数量的影响而有所增加，但是相比 MILP 和 IS-k 算法，在较短时间内得到了近似最优解甚至是最优解，在求解效率方面有着极大的优势，能够在更短的时间内寻找到近似最优解。下一步将考虑更多的因素，包括功耗、布局等，进行求解优化。

参 考 文 献

[1] Xilinx，Inc. Vivado Design Suite User Guide Partial Reconfiguration[OL]. https://www. xilinx. com/support/documentation/sw _ manuals/xilinx2020 _ 1/ug909-vivado-partial-reconfiguration. pdf.

[2] SAHA S，SARKAR A，CHAKRABARTI A. Scheduling dynamic hard real-time task sets on fully and partially reconfigurable platforms[J]. IEEE Embedded Systems Letters，2015，7(1)：23-26.

[3] DEIANA E A，RABOZZI M，CATTANEO R，et al. A multiobjective reconfiguration-aware scheduler for FPGA-based heterogeneous architectures[C]// 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE，2015：1-6.

[4] RABOZZI M，LILLIS J，SANTAMBROGIO M D. Floorplanning for partially-reconfigurable fpga systems via mixed-integer linear programming[C]// 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines. IEEE，2014：186-193.

[5] SEYOUM B B，BIONDI A，BUTTAZZO G C. FLORA：Floorplan Optimizer for Reconfigurable Areas in FPGAs[J]. ACM Transactions on Embedded Computing Systems (TECS)，2019，18(S5)：1-20.

[6] 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.

[7] MA Y，LIU J，ZHANG C，et al. HW/SW partitioning for region-based dynamic partial reconfigurable FPGAs[C]// 2014 IEEE 32nd International Conference on Computer Design (ICCD). IEEE，2014：470-476.

[8] SALAMY H，ASLAN S. A genetic algorithm based approach to pipelined memory-aware scheduling on an MPSoC[C]// 2015 IEEE Dallas Circuits and Systems Conference (DCAS). IEEE，2015：1-4.

[9] CHARITOPOULOS G，KOIDIS I，PAPADIMITRIOU K，et al. Hardware task scheduling for partially reconfigurable FPGAs [C]// International Symposium on Applied Reconfigurable Computing. Springer，Cham，2015：487-498.

[10] RABOZZI M，DURELLI G C，MIELE A，et al. Floorplanning automation for partial-reconfigurable FPGAs via feasible placements generation[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems，2016，25(1)：151-164.

[11] XIE D，SONG L F，DONG Y P，et al. Efficient dynamic partial Reconfigurable Design of FPGA based on clustering Partition algorithm [J]. Electronics and Packaging，2018，18(9)：8，14.

[12] 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-2013). IEEE，2013：47-54.

[13] TANG Q，WU S，SHI J，et al. Modeling of schedule-aware synchronous dataflow[J]. Journal of National University of Defense Technology，2017(2)：19.

[14] 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 and distributed systems，2016，28(3)：826-837.

[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). IEEE，2020：201-206.

[16] CANON L C，SAYAH M E，PIERRE-CYRILLE H. A Comparison of Random Task Graph Generation Methods for Scheduling Problems[M]// Euro-Par 2019：Parallel Processing. 2019.

[17] 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.

WANG Zhe，born in 1995，postgraduate. His main research interests include software defined radio and reconfiguring computing.

WANG Ling，born in 1962，Ph.D，professor. Her main research interests include digital signal processing and radio communication.