

学校代号 10532

学 号 G09246062

分 类 号 TP301.6

密 级 普通



湖南大学
HUNAN UNIVERSITY

工程硕士学位论文

一种改进的动态可重构片上系统任务 调度算法研究

学位申请人姓名 杨煜星

培 养 单 位 信息科学与工程学院

导师姓名及职称 刘彦 讲师 陈猛胜 高工

学 科 专 业 软件工程

研 究 方 向 通信与网络软件

论文提交日期 2013 年 3 月 18 日

学校代号：10532

学 号：G09246062

密 级：普通

湖南大学工程硕士学位论文

一种改进的动态可重构片上系统 任务调度算法研究

学位申请人姓名： 杨煜星

导师姓名及职称： 刘彦 讲师 陈猛胜 高工

培 养 单 位： 信息科学与工程学院

专 业 名 称： 软件工程

论文提交日期： 2013 年 3 月 18 日

论文答辩日期： 2013 年 4 月 6 日

答辩委员会主席： 廖波 教授

The Research on Online Hardware Task Schedule Algorithm Base-on
Partial Reconfigurable Computing System

by

YANG Yuxin

B.E.(Central South University)2003

A thesis submitted in partial satisfaction of the

Requirements for the degree of

Master of Engineering

in

Software Engineering

in the

Graduate School

of

Hunan University

Supervisor

Assistant Professor LIU Yan

Senior Engineer CHEN Mengsheng

March, 2013

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的
研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或
集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均
已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：

日期： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保
留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。
本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检
索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密☐，在_____年解密后适用本授权书。

2、不保密☐。

(请在以上相应方框内打“√”)

作者签名：

日期： 年 月 日

导师签名：

日期： 年 月 日

摘 要

可重构计算技术兼备定制化芯片的高性能和通用 CPU 的灵活性而日益受到学术界和产业界的关注。随着电子技术的飞速发展，目前的可重构器件已经可以支持部分动态可重构，可以满足现代高性能技术、专用加速计算和各种普通应用等更为灵活的应用场景对计算芯片的需求。与此同时，也就为可重构硬件任务的管理提出了新的要求。

硬件任务调度作为可重构操作系统的核心功能，是充分发挥和利用好可重构计算技术的关键。为了提高可重构器件的任务调度成功率和资源利用率，并进一步将现有调度算法相关研究工作推向实用，本文在前人研究基础上针对可重构器件异构结构上的硬件放置问题展开研究，提出了改进的任务调度算法。论文的具体研究工作如下：

为了能更好的描述和说明可重构器件资源的异构性，构建了新的可重构硬件系统调度框架和硬件任务抽象模型。在硬件任务模型中增加了任务配置时间、任务依赖关系和通信需求等因素，从而有助于后续调度算法的展开研究，也为相关研究成果进一步接近实用奠定基础；

针对硬件任务的通信需求，在前人工作基础上提出了一种改进的总线优先的任务放置策略。当具有通信需求的硬件任务到来时，放置器将优先将其放置在离可重构器件片上固定总线资源较近的地方，从而降低通信延迟和面积开销；

针对任务间依赖关系和通信需求，提出了改进的 PreSPSA 算法。该算法根据可重构器件异构资源的分布和特点，首先对预留任务队列中的硬件任务进行分簇管理，将有依赖关系和通信需求的任务尽量放置在相同的可重构区域中并尽早开始。模拟实验表明，综合上述技术的改进的调度算法在不明显增加系统开销的情况下，提高了可重构计算系统的调度成功率。

综上所述，本文围绕可重构系统中的实时硬件任务调度问题展开了深入研究，特别针对具有优先依赖和通信需求的硬件人在更为实际的异构可重构器件上的调度和管理问题提出了改进的算法，并进行了模拟实验，搭建了部分可重构原型系统，取得了一定的成果。

关键词：可重构计算；在线硬件任务调度；硬件资源管理；实时系统

Abstract

Reconfigurable computing technologies compose of customized chips of high performance and the flexibility of general purpose CPU, and then it is increasingly attracting the attention of academia and industry. With the rapid development of electronic technology, the current reconfigurable devices can support partial dynamic reconfigurable, and can meet the computing chips' requirements of modern high technology, dedicated to accelerate calculation, a variety of common applications and more flexible application scenario. At the same time, it puts forward new demands for reconfigurable hardware task management.

As the core function of reconfigurable operating system, hardware task scheduling is the key technique to make good use of reconfigurable computing technology. In order to improve the success ratio of task scheduling and resource utilization in reconfigurable devices, and further to manage the existing scheduling algorithm research applied to the practical work, this paper put forward an improved task scheduling algorithm basis of previous studies, which for the research of the question of where to place the reconfigurable devices. The specific research work is as follows:

In order to describe and explain the heterogeneity of reconfigurable device resources better, I construct a new framework of reconfigurable hardware system scheduling and hardware task abstract model. I increase the factors of task configuration time, task dependencies and communication cost described in hardware model, which would be helpful to the study of subsequent task scheduling algorithm, and lay a foundation for the related research results to be more closed to practical;

With the demand of hardware task communication, I have proposed an improved bus priority task placement strategy. When it has the communication requirements of hardware task coming, this algorithm will give the priority to put the task closer to the bus to reduce the communication delay and area overhead;

According to the dependencies between tasks and communication requirements, the improved PreSPSA algorithm is proposed. Based on the reconfigurable device distribution and the characteristics of heterogeneous resources, this algorithm firstly deal with the hardware tasks in task queue, manage these task into clusters, the task with dependency or communication will in the same area of the reconfigurable as far

as possible and start as early as possible. Simulation experiments show that the improved scheduling algorithm with these methods improve the scheduling success ratio of reconfigurable computing systems without significantly increase system overhead.

In conclusion, consider with the hardware real-time task scheduling problem of reconfigurable system, especially for the priority depends and communication demand hardware, this paper presented an improved scheduling algorithm for a more practical scheduling on heterogeneous reconfigurable devices and management problems, I have proposed simulated experiment, set up a reconfigurable prototype system, and made a certain extent achievement.

Key words: reconfigurable computing; online hardware task scheduling; hardware resources management; real-time systems

目 录

学位论文原创性声明和学位论文版权使用授权书	I
摘 要	II
Abstract.....	III
插图索引.....	VII
第 1 章 绪 论	1
1.1 研究背景及意义	1
1.2 国内外发展现状	2
1.3 论文的主要工作	4
1.4 论文结构组织	4
第 2 章 可重构理论阐述	5
2.1 可重构硬件基础	5
2.1.1 细粒度可重构单元	5
2.1.2 粗粒度可重构单元	7
2.2 可重构计算体系结构	8
2.2.1 可重构系统分类	8
2.2.2 可重构配置方式	10
2.3 可重构任务调度与管理	11
2.3.1 硬件在线任务调度	11
2.3.2 可重构资源管理	12
2.3.3 任务放置策略	13
2.4 小结	14
第 3 章 一种改进的硬件任务调度算法	15
3.1 硬件调度问题描述	15
3.1.1 硬件任务建模	15
3.1.2 调度系统框架	16
3.1.3 硬件任务调度	18
3.2 资源分配策略	20
3.2.1 算法改进思路	20
3.2.2 总线优先放置策略	21
3.3 考虑任务间通信的调度算法	23
3.3.1 算法改进思路	23

3.3.2 通信优先可抢占调度算法	24
3.4 片上资源管理	28
3.5 小结	30
第 4 章 算法模拟与评价	31
4.1 模拟系统简介	31
4.1.1 RCSSimulator 介绍	31
4.1.2 测试数据集生成	33
4.2 算法评估与分析	34
4.3 原型系统搭建	36
4.4 小结	40
结 论	41
参考文献	43
致 谢	48

插图索引

图 2.1 3-输入查找表	6
图 2.2 Xilinx 可重构单元	6
图 2.3 Altera DSP 块	7
图 2.4 松耦合结构	8
图 2.5 紧松耦合结构	9
图 2.6 片上系统结构	9
图 2.7 可重构片上系统结构	10
图 2.8 任务放置策略示意图	13
图 3.1 可重构资源模型	16
图 3.2 可重构片上系统调度模型	17
图 3.3 可重构资源模型	17
图 3.4 资源分配策略流程图	22
图 3.5 任务调度过程示意图	23
图 3.6 算法 3.4 流程图	27
图 3.7 相关矩阵抽象模型	29
图 4.1 RCSSimulator 架构	32
图 4.2 RCSSimulator 图形界面	33
图 4.3 调度模拟统计结果	35
图 4.4 资源利用率统计结果	36
图 4.5 Virtex-II Pro 开发板	36
图 4.6 模块动态重构开销	40

第 1 章 绪 论

本课题来源于国家 863 计划项目“面向可重构片上系统的过程级动态软硬件划分研究”(批准号:2007AA01Z104)及国家自然科学基金项目“异构多核片上系统自适应实时任务调度机制及算法研究”(批准号:60973030),着重于探索具有动态可重构特性的可重构片上系统的实时任务管理问题,研究硬件任务实时调度算法。

1.1 研究背景及意义

可重构计算结合了通用处理器和ASIC两者的优点,兼具硬件的高效性和软件的可编程性^[1]。随着集成电路容量的不断增加及自动设计技术的迅速发展,结合了可重构计算技术的可重构片上系统相关技术日臻成熟,成为了业界的研究热点。

摩尔定律多年来成功预测了芯片晶体管密度提升的规律并依然有效,处理器性能的增长速率也一再验证了这条定律。一般来说,计算机中的处理器性能提升取决于两个主要因素:第一是处理器运行的主频,这个方面的因素主要受制于半导体技术和具体制造工艺;第二个方面就是计算机系统结构的设计及优化,这成为一直以来计算机学术界持续奋斗的目标。在此前的几十年中,除了少数指令级并行开发等技术外,处理器性能的提升主要是通过采用复杂的设计工艺来不断提升晶体管密度和主频来实现的。进入 21 世纪后,由于半导体制造工艺的不断发展,目前已经快接近硅材料的物理极限。传统的处理器通过提升晶体管密度和主频来增加处理器性能的方法遭遇了前所未有的瓶颈,主要表现在“存储墙”、“功耗墙”等方面,难以进一步挖掘指令级的并行性。摆在研究者面前的一个现实问题是:如何能够通过利用单芯片上不断提升的晶体管密度更进一步的提升处理器性能?引入新的体系结构以及设计方法是解决问题的办法之一。

信息化社会和数字化的生活使得信息产品丰富多样、应用类型层出不穷,对处理器的设计要求也截然不同。传统的科学和工程方面的计算需求主要是浮点计算能力,随着多媒体、网络和移动设备应用的普及,对处理器的性能需求转向了实时性以及流式数据类型的响应,并且需要支持数据级、线程级等不同层次的并行。为了适应市场变化中新的应用需求,也需要设计新的处理器结构,突破传统处理器所面临的技术瓶颈。因此,片上多处理器(Chip Multi-Processor, CMP)架构在学术界以及工业界迅速得到了普遍的关注和认可。CMP是在单一芯片上集成多个较为简单的处理器核,从设计角度来说可以简化芯片设计过程,从应用角度来说可以高效的适应不同类型的需求,具有广阔的发展前途。

从多处理器片上系统的系统结构设计来说, 可以从增加单芯片上处理器内核的数量以及提升各个处理器单元之间的异构性两个方面入手。从目前来看, 主要的多处理器产品一般是通过增加片上处理器内核数量的方式来提升系统性能, 如Intel公司的 80 核处理器原型^[2], 及Tilera在 2007 年推出了 64 核处理器产品, 在 2009 年推出了 100 核处理器产品^[3]。但是当片上处理器内核的数量达到一定程度之后, 此方法所带来的性能提升非常有限, 反而更加加重了芯片在通信、同步和功耗等方面的开销。增加芯片的异构性可以更好地处理应用需求与计算机系统结构之间的结构性矛盾, 使得处理器资源更好地与实际应用相匹配。

由于大量计算密集型的应用, 如无线通信等日益增长的灵活性需求(如对于不断演化的各种标准和操作环境的适应), 器件必须在运行时具有高度的自适应性。另一方面, 这些应用也必须使用非常高效的实现方式, 特别是它们在开发中使用的资源、功耗等都极大的影响了产品的质量。对于这些典型的应用领域, 传统的指令集处理器和ASIC无法解决她们在设计灵活性和实现需求方面的矛盾。而可重构硬件则是解决这个问题一个非常有趣的方法。

当然也还有一些原因使得在SoC设计中更愿意使用可重构资源。ASIC在设计 and 流片上的巨大成本使得开发者更愿意在不同的应用中使用同样的SoC计算平台, 从而能更好的降低单芯片的成本。可重构资源的出现, 正好可以使得芯片具备根据不同的产品或者产品的不同变化做出及时的调整, 但是代价较低。同样的, 未来设计的复杂性使得在设计流程中的错误和bug可能会需要花费很大的成本、费时的去重新设计芯片。可重构资源一般都是同构的阵列, 它们都是可以原先验证无误的, 从而最小化设计时候出现错误的可能。此外, 后端可编程性也有助于修复和更改产品的失误, 这也是固定的硬件所不具备的能力。

当今的Field Programmable Gate Array, 即FPGA由于提供了灵活的硬件可编程能力和丰富的商业产品, 进一步增加了嵌入式系统设计的灵活性。具有FPGA设计能力的芯片厂商将可编程资源与各种处理器核相结合, 从而出现了将FPGA和MPSoC相结合的芯片, 成为了一类重要的异构MPSoC。Xilinx于 2011 年 3 月发布了Zynq-7000 系列平台将两个ARMCortex-A9 MPCore处理器内核与可编程逻辑及各种硬IP核结合从而提供快速开发应用系统能力。Alter推出了异构SoC FPGA平台, 包含ARM Cortex-A9 内核和Cyclone V和Arria V 系列FPGA资源的集成芯片。在以Xilinx、Alter、Intel为代表的半导体厂商的大力推动下, 具有重构特征的多处理器芯片必将在嵌入式领域得到极大地推广, 并将广泛应用到传输、能源、工业控制、医疗以及军事等多种领域。

1.2 国内外发展现状

可重构计算的主要目标是希望通过硬件可编程来满足计算任务的需求, 使硬

件自适应计算任务需求的变化,以达到最佳的计算性能。这种结构可变的特点填补了软/硬件之间的鸿沟,很好地适应了实际应用中的多元化需求。

20 世纪 60 年代,美国加州大学计算机专家 Gerald Estrin 提出了一个由通用处理器和数字逻辑部件两部分组成^[4]的可重构原型系统。限于当时的技术条件,Estrin 研制的系统虽然只是其理论设计的粗略近似,但这种结构奠定了以后可重构计算系统的核心基础^[5]。到 20 世纪 70 年代末,前苏联计算机专家 Suetlana P. kartashev 和 Steven L. kartashev 博士提出了动态可重构系统结构的概念。

1986 年 Xilinx 公司第一款基于 SRAM 技术的 FPGA 的问世,标志着现代可重构计算的开端^[6]。这些基于 SRAM 技术的可编程器件由细粒度的可编程逻辑块和可编程开关相互连接,因此可以快速地对整个 FPGA 重新编程。这一特性极大地推动了可重构计算技术的发展,并在容错计算、并行处理、集成电路等方面得到了广泛地应用。跨入 21 世纪以后,基于动态可重构芯片的动态可重构计算技术受到了越来越广泛的关注。

自 21 世纪以来,人们对动态可重构计算的研究力度逐渐加大。2002 年,Xilinx 公司推出第一款支持一维动态可重构 FPGA Virtex-2 系列,把部分可重构计算的研究热度提到了空前高度。2004 年,Xilinx 又推出了支持二维可重构 FPGA Virtex-4 系列,2002 年,Compton K 在一篇关于可重构计算系统与软件的调查报告^[7]中,详细探讨了可重构计算系统的硬件模式。其中包括从单片架构系统到多片架构系统、内部结构和外部耦合;也关注了可重构计算系统上的软件架构,即如何把计算任务通过辅助工具规划到可重构器件中去;同时还涉及到了在可重构系统运行时的高复杂性情况下如何重复利用可重构资源等问题。这是较早对可重构计算进行全方位调查研究的报告,它较系统的阐述了当时可重构计算的最前沿技术和方案,并预测了可重构计算的未来。正是因为 Compton K 等人的工作,给其后的可重构计算研究工作者提供了一个可靠模型。

Walder H, Steiger C, Platzner M 等人较早的对部分可重构系统的任务调度放置算法进行了探索研究。2003 年,他们发表在 IEEE Computer Society 上的文献^[11]中,奠定了局部动态可重构系统的资源抽象模型基础,并为该模型设计了一个简单任务调度算法。同年,在同一刊物上,他们还提出了一种划分空闲资源空间成矩形,并通过 Hash 矩阵管理这些矩形,并建立了基础此的放置策略。这个算法具备当时最好的资源管理效率和较高的任务调度成功率。2004 年,他们发表了一篇关于研究可重构嵌入式平台操作系统的论文^[10],引起了业内的广泛关注。文中舍弃了他们先前采用平面搜索空闲资源的算法,而改用 Stuffing 算法,把时间加入到资源管理算法中,使任务调度的平均成功率提升到了 80%,但时间代价太高,并且仍然可能拒绝实际上能成功调度的任务。

这些研究者的工作在可重构计算的研究工作中起了里程碑的作用,不单为部

分可重构计算系统建立了基本模型，同时他们提出的划分矩形的资源管理方式为系统为可重构资源管理研究开辟了新的方向。他们的成果引起了业内的广泛关注，他们提出的 Stuffing 算法也成为了后来研究者评估自己算法性能的一个主要参考算法。其后的很多国内外研究者在他们的模型的基础上对他们的算法进行了改进，更多的研究者针对他们建立的模型提出了更好的算法。

1.3 论文的主要工作

第一，在传统基于矩形的可重构系统建模方法的基础上，考虑实际FPGA商用产品和芯片的特性，构建一个更为实用化的模型，充分考虑可重构片上系统在通信、外设接口及动态可重构资源配置等方面的实际情况。

第二，在上述较为贴近实际应用的动态可重构系统模型中研究硬件任务实时管理问题。任务放置算法的优劣影响着动态可重构片上的芯片利用率和计算性能，在保证资源利用率的同时提高任务放置质量，是本论文研究的重点。

1.4 论文结构组织

本论文共分四章，其组织结构为：

第1章：绪论。首先介绍本文工作的项目来源，然后阐述了论文的的目的及意义、拟解决的关键问题和论文组织结构。

第2章：相关研究。简要介绍了可重构计算相关基础知识，分析了国内可重构操作系统及硬件任务调度算法相关领域的研究现状。

第3章：一种改进的硬件任务调度算法。构建了一个新的实用化可重构片上系统模型，并给出硬件任务调度问题描述。在此基础之上，阐述了本文主要的算法改进思路和具体方案。

第4章：算法模拟与评价。在前人研究基础上，改进了算法模拟系统。对本文提出的改进算法进行模拟实验和评估。所提算法从一定程度上提高了任务调度成功率和资源利用率，其更重要的意义在于可更进一步使得调度算法更接近实际使用平台。最后搭建了一个具有部分动态可重构能力的原型系统，进行了验证性实验。

最后，总结本人在整个算法研究过程中的心得，对下一步工作做出展望。

第 2 章 可重构理论阐述

可重构计算指系统与某种形式的硬件可定制资源一起协同工作。这些可定制的硬件资源可以周期性的改变从而达到使用同样的硬件执行不同的应用程序的目的。可重构硬件在系统实现的有效性和灵活性上提供了一个良好的平衡，从而与传统的指令集处理器上时序顺序计算模型相比具有显著的效率优势。

2.1 可重构硬件基础

当前各种类型的应用对于计算平台提出了诸多要求，特别是对计算部件的性能和功耗要求特别苛刻，如流媒体视频应用、图像与模式识别和高端人家交互应用等都对计算硬件提出了新的要求。同时，芯片功耗和生产的成本依然是一个产品成百的关键，这使得计算平台在结构设计上颇具挑战。一般来说有三种主要的计算结构思路，即高性能微处理器结构、专用集成电路ASIC结构和可重构计算结构。使用微处理器结构，可以方便的从市场上买个各种类型的高性能处理器并易于使用。但是由于其通用性的设计使得其在单个领域应用时计算效率和能效均无法令人满意，并且即使摩尔定律依然有效，处理器性能的提升依然难以满足个别高性能计算领域的需求；使用专用集成电路ASIC则提供了与微处理器完全相反的解决方案，它通过全定制化的硬件设计可专门针对某一个或者某一类型的应用进行优化，充分利用程序的并行性从而获得性能和能效上的最佳，但与此同时也就失去了系统开发的灵活性，而且开发成本较为昂贵；可重构计算作为上述两种方式的折中方案，既保持了微处理器结构可编程的特性，又可充分领域ASIC方案中所体现出来的硬件特性。一般来说，可重构计算系统包含有一个或者多个处理器和可重构资源。处理器内核可以用来执行顺序指令及非关键性代码，而计算密集型任务和关键代码则可以在可重构资源上用硬件实现，从而获得更好的整体系统性能。

可重构系统最早诞生于 20 世纪 60 年代，其飞速发展开始于 20 世纪 80 年代 FPGA（Filed Programmable Gate Aarry）的出现。由于FPGA具有硬件可编程特点，运用FPGA的可重构计算（Reconfigurable Computer, RC）迅速发展起来，可重构片上系统的相关研究也逐渐增加，并已经成功将其应用于数据压缩和模式识别以及高性能计算等领域。

2.1.1 细粒度可重构单元

一般来说可重构功能单元可以分为细粒度和粗粒度两种。细粒度可重构单元

最典型的构成方式就是使用较少的位来实现最为基本的功能组件。当前使用最为广泛的细粒度可重构单元是查找表，大量在商用可重构器件和芯片中使用。

很多的可重构计算系统使用商业可重构芯片作为基本单元，这些商用的FPGA芯片一般包含有 3 输入或者 6 输入的查找表，这就构成了一种非常标准的细粒度可重构单元。如图 2.1 所示为一个 3 输入查找表的基本结构图。只要输入正确的配置信息，即bit-stream，该查找表可以实现任何 3 个输入真值表所代表的功能。并且对查找表的输入进行扩展以实现更复杂的功能也十分容易。在本文使用的Xilinx可重构计算平台中也是基于查找表的方式来实现硬件可重构的。

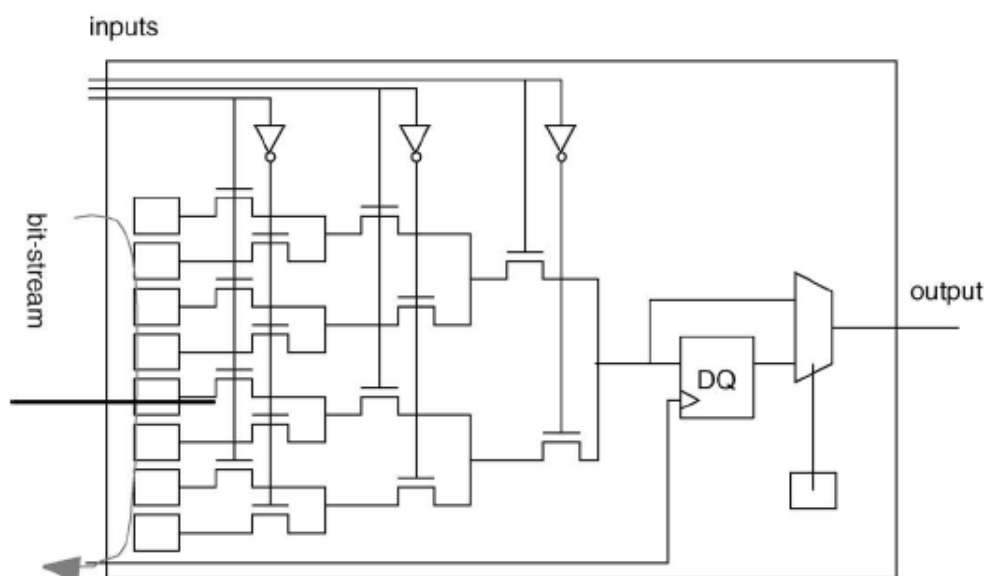


图 2.1 3-输入查找表

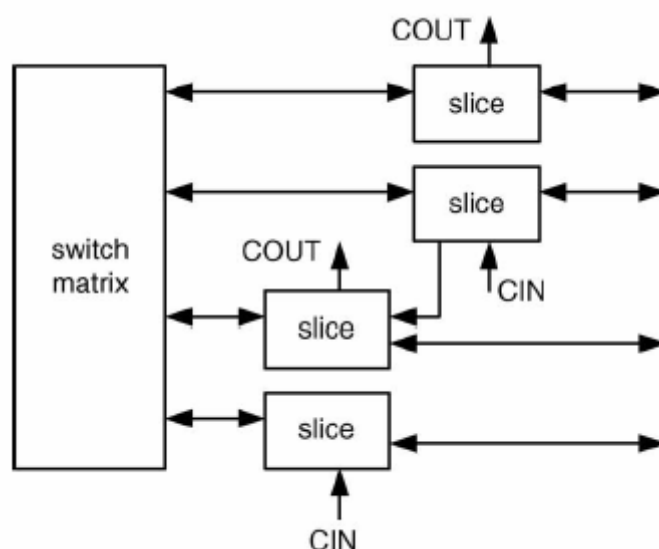


图 2.2 Xilinx 可重构单元

其基本的细粒度可重构单元如图 2.2 所示。在Xilinx的文档描述中称图 2.2 所示结构为可配置逻辑块（Configurable Logic Blocks, CLBs），其中每个“slice”包

含有 2 个查找表。细粒度可重构单元最为显著的优势是灵活性，可以实现任何逻辑电路。但与此同时，与粗粒度可重构单元相比，灵活性的优势也就导致了在功耗、速度、器件面积和延迟方面的劣势。由于大多数的商业FPGA主要是用来对计算密集型的任务进行单独的加速处理，因此用于实现各种类型的算术单元是细粒度可重构单元的主要用途。因此，在实际的Xilinx芯片内部，还在查找表的基础上进行了进位优化和基本门结构配置，从而进一步提升可重构单元硬件部件实现的效率。

2.1.2 粗粒度可重构单元

粗粒度可重构单元则与上述细粒度可重构单元相反，其电路规模较大，由多个算术或者逻辑单元部件组合而成，可以实现较为独立的功能，甚至是实现大容量的存储单元。目前使用的主要商业FPGA芯片中也存在粗粒度可重构单元。它们一般是由细粒度可重构单元组合而成，结合其他特殊部件可以完成特定类型的任务。例如实验室的Xilinx Virtex可重构器件就在单芯片内部具有独立的乘法器模块。一般来说，商用FPGA芯片大规模应用于通信领域，因此芯片专门设计了多个粗粒度可重构的乘法器模块，特别适合于数字信号处理类型的运算硬件实现。在基于Xilinx的FPGA进行通信设备、滤波算法等开发时，灵活的使用芯片内部提供的乘法器模块，将可以显著的提升系统的效率和性能，节省硬件面积。作为全世界主要的FPGA供应商之一的Altera公司的Stratix系列芯片则包含功能更为完善的粗粒度可重构模块，称为DSP块，如图 2.3 所示。

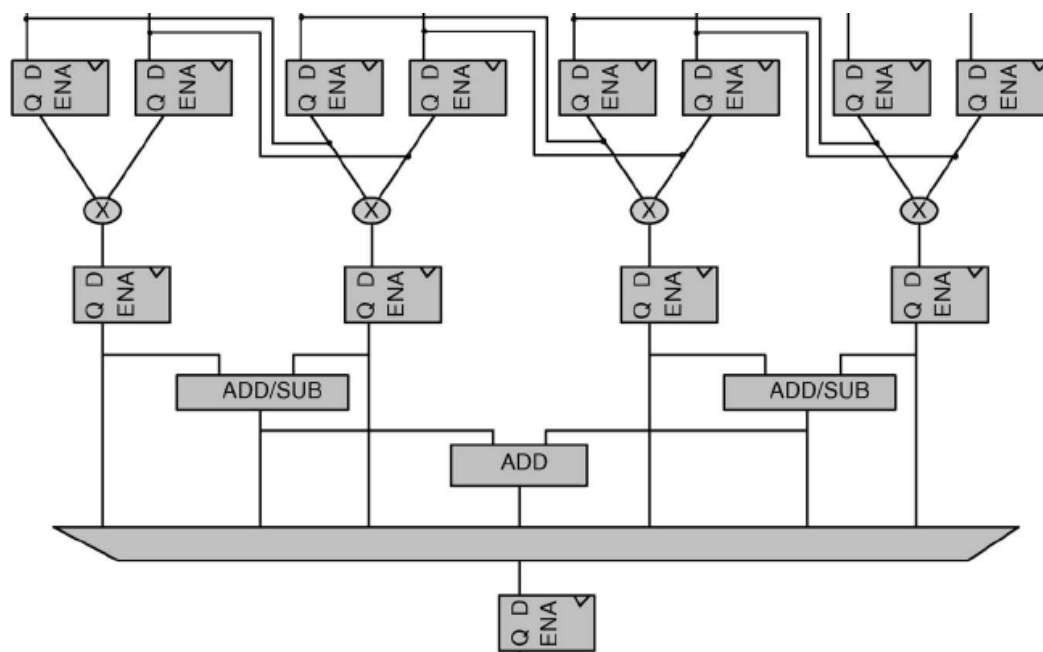


图 2.3 Altera DSP 块

每个DSP模块可以执行“乘-加”操作，这在数字信号处理特别是滤波器的设计中具有很重要的作用。与上述Xilinx公司的Virtex芯片上的粗粒度可重构单元相

比，DSP块具有更好的灵活性和适用性，可以更加好的满足具体应用设计的需求，但同时DSP块在芯片面积、数据处理速度和功耗上则比不上Xilinx的乘法器模块。

与商业FPGA相对应的还有一种类型的可重构芯片，称为定制化可重构芯片。它们多数是科研机构或者少数大企业为了某一个特定的应用专门进行的定制化设计和生产出来的器件。其可重构基础单元、配置和开发工具及内部体系结构均具有独特性，其内部一般广泛采用了粗粒度可重构单元进行面向应用的优化设计。由于此类型平台价格昂贵、开发和应用灵活性有限，不在本文详细讨论之列。

2.2 可重构计算体系结构

2.2.1 可重构系统分类

典型的可重构计算系统由一个或者多个处理器、一个或者多个可重构资源和存储模块组成。依据可重构系统中可重构计算资源与传统CPU的耦合程度一般可以分成以下几类。

第一类是耦合度最松的一类，也是可重构计算早期发展时候常用作系统加速部件时使用的结构，如图 2.4 所示。

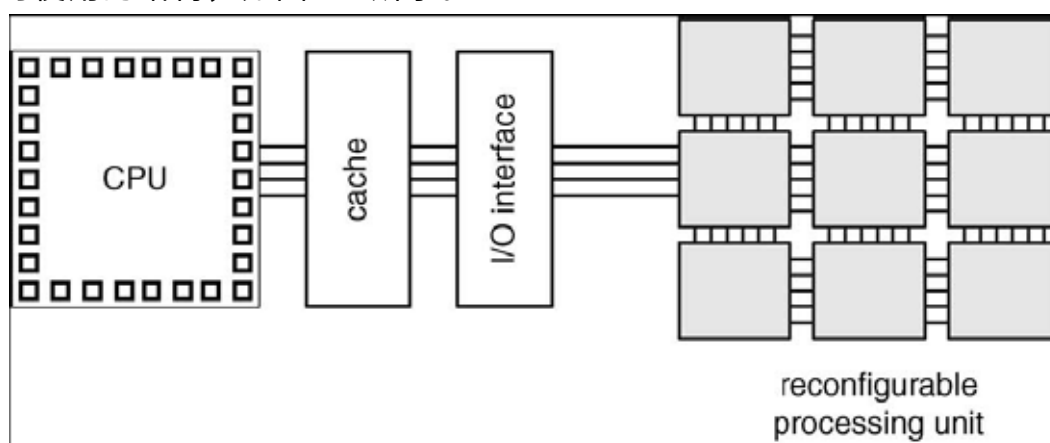


图 2.4 松耦合结构

在此类可重构计算系统中，可重构处理单元完全是作为相对于传统CPU单元较为独立的部件存在。他们之间通过CPU计算系统的外部设备接口进行通信。使用此种结构的一个重要的缺陷在于可重构处理单元与CPU计算部分之间的通信依赖于传统I/O，其通信速率成为了系统性能瓶颈。一般来说，此种结构的计算平台适合于运行类似仿真系统软件、芯片设计模拟等应用程序，因为此类应用启动以后需要CPU进行的干预比较少，可以尽可能的发挥松耦合结构的优势，回避缺点。

第二类是紧耦合结构，示意图如图 2.5 所示。在紧耦合结构中，可重构处理单元相比于松耦合结构中的位置，要更加靠近CPU。一般来说，既可以将可重构处理单元至于cache外围，也可以将可重构处理单元至于CPU与cache之间，这

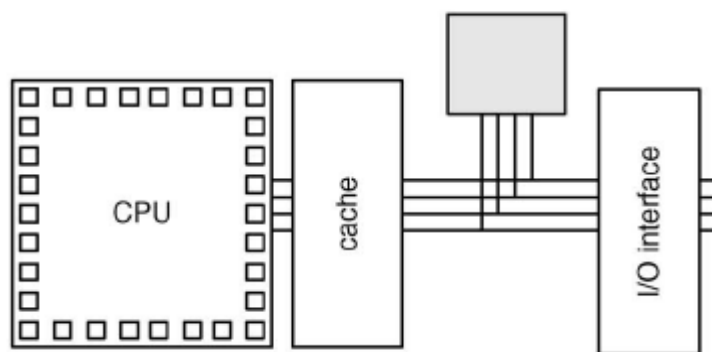


图 2.5 紧松耦合结构

取决于你的应用系统对于存储系统设计的具体需求。紧耦合结构与松耦合结构最大的不同在于，可重构处理单元与CPU之间的通信开销显著降低了，CPU可以更加快速和高效的对在可重构处理单元上运行的各种程序进行干预。与此同时，带来的代价就是作为加速部件的可重构处理单元挂载到CPU架构和存储系统的内部总线上，它本身的行为和数据访问将会同样给CPU的程序运行带来很大的影响。

第三类可以称为片上系统结构，示意图如图 2.6 所示。此种结构是一种非常紧密耦合的结构，它将可重构处理单元与CPU集成到一块芯片之上。从芯片外部

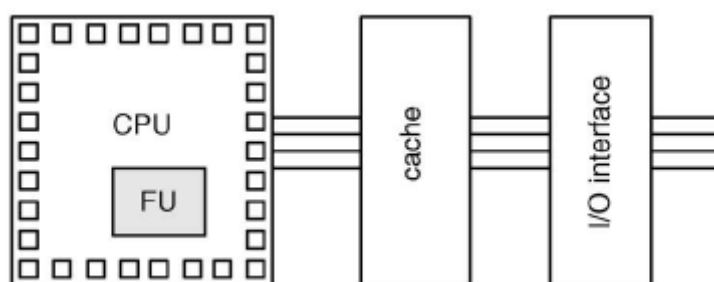


图 2.6 片上系统结构

接口来看，此芯片与传统的顺序执行CPU没有区别，但是再其内部确有单独的可重构资源用于进行单独的数据加速处理。一般来说，此种结构的芯片十分灵活。在保持传统串行CPU设计方法和板级系统的情况下，可以使用可重构处理器单元进行定制化的开发，包括定制开发加速部件、算法硬件实现单元，甚至可以对传统CPU的指令集进行面向应用的扩展。提供灵活性的同时，当然也要求开发人员对于软件、硬件编程都具备一定的了解。

随着 21 世纪微电子技术的飞速发展,另外一种有趣的可重构计算器件逐渐在市场上获得认同。它们一改传统计算平台以串行CPU为主、FPGA或者可重构计算资源为辅的计算模式，而是提供一个超大容量的、具有硬件可编程能力的FPGA芯片，并且可以在这个芯片内部构件一个或者多个传统串行CPU内核，也组成了一种较为典型的可重构片上系统，如图 2.7 所示。显然，此种芯片的推广要数

Altera/Xilinx这个两个FPGA商业巨头最为重视。此种芯片特别满足于现在数字通信技术的需求。在传统通信领域，FPGA已经发挥了不可替代的作用，在核心路由器、通信协议实现、交换机等重要产品中均有广泛的使用。随着各类增值业务的持续拓展，若能在保持FPGA传统优势的同时使得单芯片中亦可执行网络服务、控制与界面管理等传统串行CPU所负担的工作，那么对于产品开发人员来说是有利的技术选择。

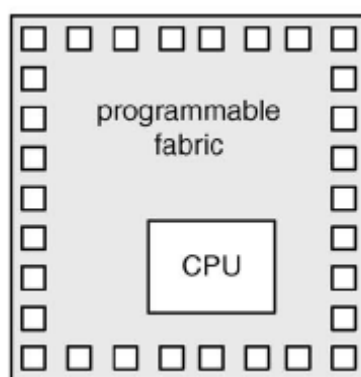


图 2.7 可重构片上系统结构

从可重构计算系统结构的研究范畴来说，还有诸如路由与互连网络、配置与优化等核心内容，但由于本文所涉主题与其联系不多，在此不再赘述。

2.2.2 可重构配置方式

一般来说，可重构资源的配置方式可以分为静态重构和动态重构两种。

静态可重构（或者称为编译时可重构）是现在可重构计算系统实际应用中进行的硬件程序配置最为简单、使用最为广泛的一种方法。使用静态可重构配置方法一般来说较为适合硬件改进或者升级频率较低的系统，特别，一个完整的应用或者程序可以由一个配置完成时使用静态配置策略最为合适。其主要的目的就是提高系统配置的性能，节省配置时间。

这种配置方式的特点就是由一个系统级的配置组成。在开始操作之前，可重构资源将装载他们各自的配置文件。一旦操作开始执行，可重构资源将在应用操作的整个做成中间保持这些配置不变。这样的话，硬件资源在整个设计或者应用的生命周期里都是静态的。相比较于纯软件方式（如微处理器）有更好的性能，相比较与ASIC方式有更好的成本优势，并且获得较为成熟的CAD工具的支持，就是静态可重构方式的优点。

静态可重构在应用运行之前分配了资源，而动态可重构（或者称为运行时可重构）使用动态分配的策略，可以在运行时对硬件资源进行重新分配。动态重构某种意义上是作为一种更为灵活的“时间-空间计算”的平衡的高级技术。他可以通过使用在系统执行期间动态的装载和卸载高度优化的硬件资源来提高系统的性

能。在动态可重构模式之下，系统的灵活性得以保持，但是功能密度更高了。

动态可重构是基于虚拟硬件的概念的，类似于虚拟内存的模型。在此，实际上的硬件资源要小于所有配置所需要的硬件资源之和。因此，为了不减少实际上可配置的应用的数目，可以采取在实际硬件资源上根据他们的需要换进换出的使用硬件。Xilinx公司目前推出了部分动态可重构器件用于支持高级应用特性的开发。部分动态可重构器件就是在保有动态可重构器件运行时可配置的特点基础上，允许只对芯片的部分区域进行重配置，而其他的区域依然正常运行。实际上，此项部分动态可重构技术也就成为本文硬件任务调度与管理算法具体应用实现的基本技术模型和基础。

2.3 可重构任务调度与管理

2.3.1 硬件在线任务调度

由于当前的FPGA等商用可重构计算平台在技术上具备了部分动态可重构的能力，即可在保持芯片上其他区域正常运行的情况下，对给定区域进行重新配置、改变功能。由此就有了针对硬件任务进行在线调度的可能性。一般来说，系统中采用在线任务调度方式进行任务管理，调度器将以先来先服务的方式逐个处理到达的任务，而调度器对于后续的任务相关信息是无法预先获知的。在我的研究课题中，硬件任务的上下文切换时间尚未到达理想的水平^[8]，因此假定已经配置到芯片上的硬件任务一旦开始启动执行，在其未完成之前，是不可以被换出芯片的，即不可剥夺。研究工作主要包括两个方面：其一，针对部分可重构器件中的空闲资源管理进行研究；其二是针对任务的放置策略进行研究。

2003年，Walder. H等发表的文章中^[11]，首先提出了局部动态可重构系统的资源抽象模型，并设计实现了一个简单的任务调度算法。2004年，他们将研究结果整合后在IEEE Transactions on Computers发表了关于可重构操作系统的论文^[10]，成为了相关研究领域的奠基文章。论文中对可重构资源模型进行了归纳和总结，并提出了Stuffing算法，任务调度的平均成功率提升到了80%，但开销较大，并且仍然可能拒绝实际上能成功调度的任务。Stuffing算法也成为同类算法评价的重要基础。

对于部分可重构系统来说，随着系统运行时间的增长，可重构器件资源必然逐步被各种计算任务“填充”，它们遵循着先到先运行、运行完后释放资源的原则。相比较于传统的串行处理器任务调度，硬件任务调度具有一个显著的特点就是，任务启动时间和就绪队列的任务调度安排必须要结合该任务在可重构器件上所放置的位置以及可重构器件上空闲的资源情况来决定。这是硬件任务调度与传统软件任务调度最为显著的区别。现有操作系统无需考虑任务的放置问题，只要考虑

资源的分配问题。因为，研究者一般使用矩形任务模型进行硬件任务调度研究，随着任务就绪队列中任务的增加，可重构器件上将会频繁的出现任务换入换出的情况，这时就会在可重构器件的表面留下“碎片”，类似文件系统的碎片问题^[44]。

一维可重构器件上的硬件任务碎片较易发现和管理^[9~11]，类似于传统软件环境下的内存碎片管理问题。对于目前研究得更为丰富的二维部分动态可重构器件而言，最为重要的方法是在运行时任务管理部件上设计优化的硬件任务放置策略从而降低碎片的产生^[12]，显然这比后续查找和消除碎片要积极和重要。对此有些研究工作者采用二维（2D）方式来管理一维（1D）可重构资源^[13,14]，或者使用技术手段跟踪可用的空闲区域，以便获得一个更有效的管理方案^[8,17~22]。

2.3.2 可重构资源管理

由于一般将可重构芯片表面的可重构资源划分为矩形区域。一般来说，可以使用一个数据结构管理和维护矩形空闲区域。当有新的硬件任务需要进行放置时就从这个空闲区域列表中进行查找和匹配。此种方法的优点在于实现相对比较简单，思路比较明确和清晰，但使用此种方式做到完全可识别性则开销比较大^[17]。相关的研究如文献[10,17,45]为了降低硬件任务放置时的开销从而满足系统对于实时性的要求，就只能管理小数量级的空闲举行空间，也同时失去了完全可识别性。

此外，可以通过将每个可重构单元是否可用的状态映射成为一个状态矩阵，然后基于状态矩阵开发相关的搜索和管理算法^[23~25]。这种方法的主要优点是可以较为明确的完成硬件任务调度的完全可识别性，即如果可重构器件表面的区域存在可以放置硬件任务的区域，则该算法必定能够找到。但时间运行开销较大依然是改类型算法投入实用的巨大挑战。

实际上，硬件任务放置问题抽象之后就是在平面上寻找一个空区域的问题，属于计算几何学中的一个传统问题^[26~28]。对于部分动态可重构系统中经常使用到的二维可重构系统模型，文献[8]把空闲矩形区域划分为可重叠的区域来管理，文献[17]则将其划分为不重叠的矩形来管理。但使用互补重叠的矩形进行资源管理有可能得到不是最大空间的矩形，无法满足完全可识别性的要求。实验数据表明，使用可重叠划分的方法对矩形区域进行管理可以提升放置质量约 8%^[8]。从提高可重构计算系统的运行效率来看，在线任务调度算应该尽可能的具备完全可识别的特性，因此主要的研究工作集中在优化算法、降低算法开销等方面^[20,29]。

文献[20]中通过对比可重叠举行划分和非重叠矩形划分方法发现，前者所得到的矩形数目多一些，因此也将会更加占用存储空间、消耗更多的搜索时间。因此，作者的主要工作是采用一种新的数据结构来记录划分的矩形，并提出了 Staircase 算法，并最终证明文中所提方法一定能找到任何一个最大的空闲矩形，

把最坏情况下的时间复杂度降低到 $O(W \times H)$ 。

Yi Lu等在DATE08上发表了一篇重要的论文，报告了目前效果最好一种基于可重叠矩形划分的硬件任务资源管理算法。文中的算法成功的将资源管理的时间复杂度降低为 $O(2N)$ ，其中 N 为当前可重构器件上已放置任务的数目。但实际实现时整个算法的效率并没有多大提高，且不能很好的与任务放置策略相结合。

2.3.3 任务放置策略

任务放置策略主要是指，当硬件任务到达后调度器需要在一系列可用的空闲矩形中寻找最为合适的一个区域进行放置（因为可以进行放置的区域可能有多个）。任务调度必须满足任务的实时性和面积约束要求。如图 2.8 所示。

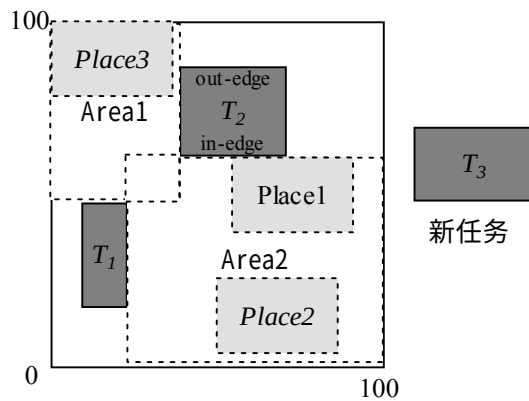


图 2.8 任务放置策略示意图

如图 2.8 中，可重构器件上已经有两个硬件任务在运行，分别是 T_1 、 T_2 。当有新的任务 T_3 到来时，显然器件中有多个位置可以用于放置运行。实际运行中的调度器将会通过资源管理算法发现，有图 2.8 中的Area1 和Area2 两个矩形区域可以用于放置新来的任务（区域是重叠的），并且硬件任务在区域内的具体放置地点也是需要进行决策的。实际上，矩形区域内的放置问题是NP难问题^[30]。

华中科技大学的黄文奇^[31~33]等针对上述问题提出了一些启发式的调度算法，对离线调度方法取得了较好的成果，包括使用了拟任的基于占角的布局算法。早期可重构系统任务调度的相关研究中较多使用的是FirstFit (FF) 策略，即最先找到合适的矩形区域即进行硬件任务的放置。后续一系列工作希望找到最为合适的代价函数来衡量任务放置的效果^[9,14,17-22,24,44,46]，其中比较重要的工作是使用任务面积之差来评价放置质量，但该方法对于碎片产生的控制和管理效率不高。

此后，齐骥等人^[34,46]提出了一种基于被占用区域连续度作为代价函数的机制。芯片上已占用区域连续度(Degree of Occupied Area Concatenation, DOAC)是当前被占用可重构资源连续度的量化值，也叫任务投影连续度DTSC(Degree of TS Concatenation)。它把一维可重构计算的任务调度成功率提高到了 77.3%。周学功等人在文献[45]中提出了新的代价函数，并进行了大量的数据仿真，取得了目前

为止最好的调度效果。

2.4 小结

本章首先简要介绍了可重构计算所以来的硬件基础，即基本的可重构单元的结构。然后，对可重构计算体系结构进行了分类，并说明了各种架构设计的特点。最后，对可重构硬件任务调度与管理相关领域的研究现状进行了综述和评价，为论文后续工作奠定基础。

第 3 章 一种改进的硬件任务调度算法

从前文的分析可知，目前正对可重构计算系统硬件任务调度的研究主要有几个方面，其一是在目前所建立的二维矩阵模型的基础上，继续改进硬件任务在线调度算法的效率，这可以主要从两个方面入手，即一方面提升可重构器件空闲资源的管理效率，另一方面着重提升硬件任务进行实际放置时的策略。与此同时，随着高性能商用FPGA芯片的推出，目前以Xilinx公司为代表的FPGA产品已经可以提供二维动态可重构的能力，但是其在实际使用时还存在很多限制，与传统上可重构硬件任务调度抽象模型有很大的不同。上述两个方面的工作将是本文在课题组现有研究基础之上进一步提升动态可重构硬件任务在线调度器及其算法设计的主要思路。

3.1 硬件调度问题描述

动态可重构片上系统的硬件任务调度问题与传统CPU架构下的串行计算系统的任务调度问题有显著的不同，因此本节首先介绍部分动态可重构硬件任务调度问题的抽象模型描述。

3.1.1 硬件任务建模

一般来说，认为使用图形或者硬件描述语言进行输入后电路系统设计，经过与实际可重构平台相关的工具进行编译和综合后，即可得到能够在可重构器件进行布置的位流文件(Bitstream)。对于这个过程一般来说使用二维矩形区域来建模。因为“硬件任务”实际上是编译完成之后的实际电路系统，对于可重构器件来说，其最为重要的属性就是占用的晶体管数量，或者说“面积”以及这个电路本身的“形状”。因此，将可重构器件资源建模成由一系列的可重构资源组成的矩形表面区域，那么“硬件任务”即依据它所需要的可重构资源编译成为相应的矩形块。系统的调度器就需要将“硬件任务”的矩形块放置在可重构器件的表面，从而完成硬件资源的配置。虽然使用简单的矩形模型来对硬件任务和可重构资源进行建模会带来显著的碎片管理问题，但依然是目前主要的建模方式，简单高效，并且与当前工业界流行的主流EDA工具想配合，有助于理论研究工作未来进一步的走向实用。硬件任务最为重要的属性是系统执行硬件任务时的时钟频率、硬件任务执行的时间，已经改硬件任务实际占用的可重构器件资源的多少。

现在对常用的矩形硬件任务模型做一下简单的介绍。如图 3.1 所示，常用的矩形硬件任务模型有一维模型和二维模型两种。抽象模型中，动态可重构器件的

硬件资源是由可重构单元（RCUs）组成的矩形来描述。图 3.1（a）所示为一维模型，矩形的硬件任务块可以沿着水平方向放置，而期间表面的资源在垂直方向上不再进行分配。从模型中可以看出，一维模型可以大大简化硬件任务的放置问题，但是同时也带来了较为明显的碎片问题，而且其对于空间的利用率难以让人满意。由于硬件任务块的水平方向宽度不可预知，由此就会产生如图 3.1（a）中所示的“碎片 1”，此外，垂直方向上剩余的可重构资源没法再配置，则使得“碎片 2”成为器件上可重构资源无法充分利用的主要原因。

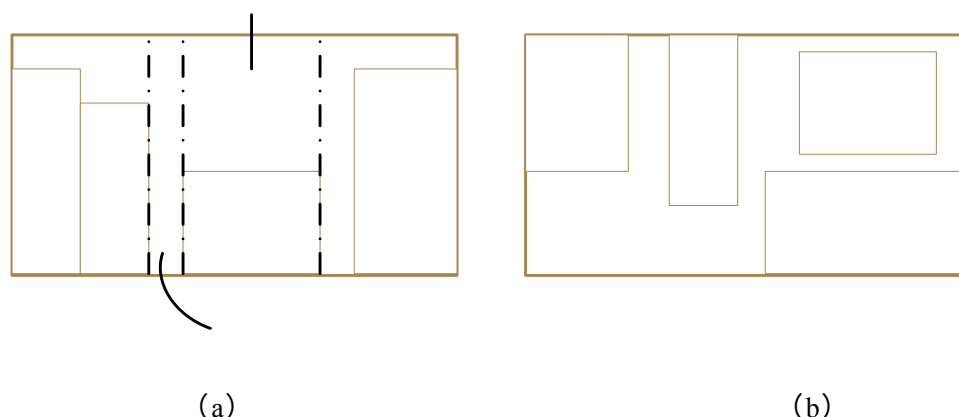


图 3.1 可重构资源模型

图 3.1（b）中展示的是动态部分可重构器件资源的二维模型。在上述模型之下，硬件任务可以在水平和垂直两个方向上任意放置。这样的放置策略显然将可以获得更高的调度成功率和可重构器件资源的利用率，也有利于进一步的降低碎片的产生。但同时也带来了硬件任务调度和放置问题解决的复杂性。

3.1.2 调度系统框架

虽然目前对于商用FPGA上部分动态可重构特性的深度应用和开发尚未取得大规模的应用，但是相关的理论和工具研究已经引起了广泛的关注，而且Xilinx公司推出的Virtex系列FPGA已经能很好的支持部分动态可重构系统开发了。本节将参考相关文献，描述本文所研究的硬件任务调度问题的整体系统框架。

基于可重构片上系统SoC的硬件任务在线调度系统的整体框架如图 3.2 所示。整个可重构计算系统可能是有一片商用FPGA芯片组成，也可以有多个FPGA和传统CPU芯片组成，从抽象模型的角度来说，可以描述为一个由互联网路负责通信的异构多处理器系统。可重构SoC中传统CPU负责执行串行控制类型的任务等，可重构资源则主要用来实现计算密集型任务。对于新到来的任务，将有SoC芯片上运行的具备可重构硬件管理功能的操作系统进行调度和管理。其主要的功能部件有调度器、放置器和装载器。编译完成后的硬件任务模块到达时，可以存放在任务就绪队列中等待进一步的处理；就绪任务队列进入调度器后，由其运行调度

算法为每个硬件任务分配具体的开始时间；此时，硬件任务已经完成了传统意义上的调度工作，等待放置器进一步为其在可重构器件表面寻找合适的位置进行布置。需要特别指出的是，此处调度器和放置器需要协同工作以共同确定一个硬件任务模块的任务启动时间和放置位置，两者孤立无法更好的解决问题。硬件任务运行完成之后，系统可以收回其所占用的这部分可重构资源，并用于重新分配，进入空闲资源进行管理。调度器对于硬件任务的处理结果即为任务调度成功和失败两种。目前任务调度成功率是衡量硬件任务调度算法性能的主要指标。

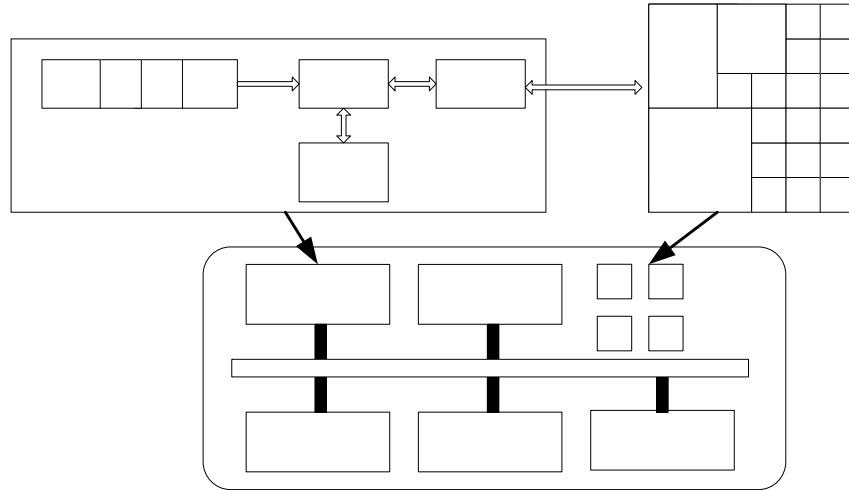


图 3.2 可重构片上系统调度模型

本文在上述调度模型系统框架之上,对可重构器件资源模型做进一步的细化,充分考虑器件在实际应用时的情况,在系统建模时引入组件的异构性、任务间通信及外部设备接口等与实际器件紧密相关的内容,从而使得硬件任务调度研究更加接近实用水平。图 3.3 所示即为新的实用化可重构资源模型^[24]。

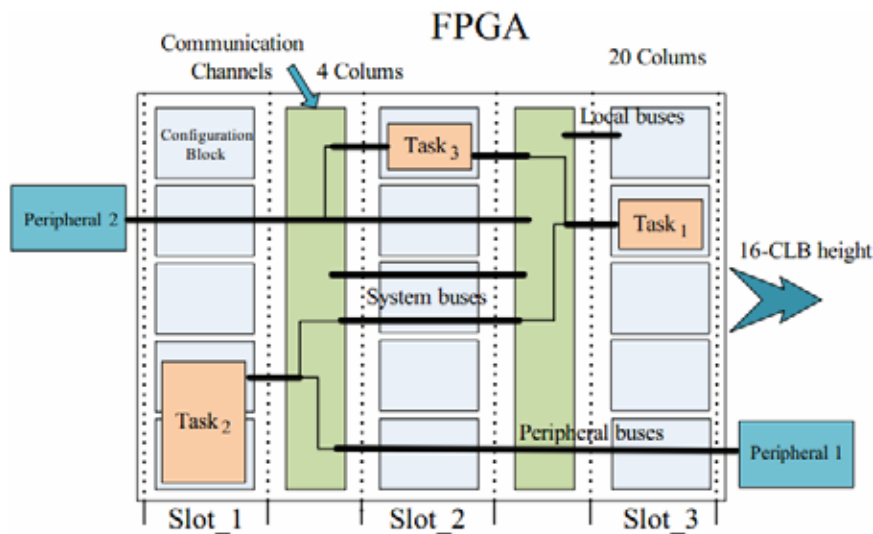


图 3.3 可重构资源模型

对基于FPGA的动态实时调度模型添加新的约束，使模型更接近于真实任务单元

务之间及任务与外部设备的数据通信引入了一个新的主要时间约束：任务数据通信时间。如果不考虑通信时间，在一个应用执行期间，调度器不能保证任务之间、任务与外部设备之间正确的数据交换。图 3.3 所示的新的资源模型中，FPGA被划分为槽(slot)，每个槽再进一步被划分为块(block)。块即是配置块，是最小的配置单元。每一个任务仅允许放置在同一个槽中的相邻配置块。通信结构包括通信信道、局部总线、系统总线和外围总线。每一个配置块通过局部总线连接到相邻的通信信道。在相邻的槽之间，增加通信信道，这样所有的总线都可以连接。系统总线连接所有的通信信道从而水平扩展FPGA，使得非邻接槽中的任务也能够通信。与系统总线不同的是，外围总线连接的是通信信道和外围设备。

3.1.3 硬件任务调度

传统的硬件任务调度研究主要如图 3.1 所示，硬件任务可以描述为具有矩形的形状和一定的大小。任务的大小使用执行该任务需要占用的可重构单元的数量来表示，而任务的形状定为矩形。例如，一个任务 T_i 被建模成面积为任务所需占用的RCU区域的宽度和高度($w_i \times h_i$)的矩形空间。

本文的研究重点在于考虑更为实用的环境下，结合实际FPGA商业器件平台的特点进行硬件任务调度研究。有前文的分析可知原有的硬件任务调度问题主要有以下几个方面的关键因素：

其一，硬件任务在可重构器件表面使用占用一定数量RCU的矩形块来表达；

其二，硬件任务主要的时间属性是任务启动时间、任务执行时间和任务期限；

因此任务调度问题就可以描述为，在FPGA器件的表面，为每一个到来的硬件任务安排一个具体的放置位置，使得其能在任务期限到来之前顺利执行完毕。如果结合图 3.3 所示的更为实用的可重构片上系统结构模型，则调度问题需要增加更多更为细节的考虑。

首先，任务间的通信是硬件任务实际在可重构片上系统或者FPGA器件上运行需要解决的首要问题。这一点在传统的任务调度模型中忽略了。

其次，任务间有通信的需求，同样也就使得任务见的优先性约束成为制约硬件任务调度的一个重要问题。传统的任务调度模型假设硬件任务相互之间是无关的，这个假设显然很大程度上与实际应用差距较大。

因此考虑到增加任务间通信和任务间优先性约束的条件之后，可以对新的更加实用的硬件任务调度问题进行如下描述。

硬件任务形式化描述：在更为实用的硬件任务调度模型中，实时的硬件任务一般可以用一个七元组的集合来表示 $T(w, h, a, e, d, p)$ ，其中 w 、 h 所描述的是硬件任务在FPGA器件表面进行配置时所占用区域的可重构单元RCU的数量，由其构成的矩形的宽度和高度； a 表示硬件任务到达系统进行计算处理的时间； e 表示硬

件任务本身在可重构计算平台上执行完成所需的时间, d 与传统实时任务调度中的任务截止时间含义一致, p 则是表示硬件任务从配置CPU或者memory中开始放置, 知道此配置过程完成所需时间, 即为任务配置开销。更进一步, 为了在此基础模型之上可以清楚地表达硬件任务模块的优先性约束和通信时间开销, 使用一个矩阵 $E(i, j)$ 记录类似DAG图中的边及其权值, 用于描述硬件任务 T_i 和 T_j 之间的优先关系及通信时间, 权值为 c_{i, j_0}

与前期研究者不同的, 本文的研究在任务调度基础模型上进行了改进和调整, 使得可重构硬件任务的描述更为接近现实需求。从而, 依据上述改进后的硬件任务调度模型, 对于可重构硬件任务实时调度问题可以描述如下:

一组硬件任务集合使用七元组 $T(w, h, a, e, d, p)$ 表示。对于任意一个调度成功的硬件任务使用 $T(x_1, y_1, x_2, y_2, s, f)$ 来描述, 由 (x_1, y_1) 、 (x_2, y_2) 两组笛卡尔坐标系值来表示硬件任务在FPGA可重构器件表面成功放置区域的对角线定点坐标; 可重构系统调度器为硬件任务调度决策的任务启动时间则表示为 s , 任务在器件上执行结束的时间为 f , $f = s + p + e$ 。调度器和放置器相结合, 目标就是为了要找奥一个符合下述规则的、可以被系统所接受的有效硬件任务调度:

对于一个被调度的任务 $T(w, h, a, e, d, p)$, 如果对它的一个调度 $T(x_1, y_1, x_2, y_2, s, f)$ 满足以下四个条件, 则称其为任务 T 的一个有效调度。

- 1) $x_2 = x_1 + w - 1 \cap y_2 = y_1 + h - 1$
 $\cap 1 \leq x_1 \leq W \cap 1 \leq y_1 \leq H$
 $\cap x_1 \leq x_2 \leq W \cap y_1 \leq y_2 \leq H$ (放置位置不超出边界)
- 2) $s \geq a \cap s \geq \text{Pred}(\text{all})$ (启动时间在到达时间之后, 并且前趋任务已完成)
- 3) $s + p + e \leq d$ (保证截止时间前完成)
- 4) $\forall$ 已调度任务 $B(x'_1, y'_1, x'_2, y'_2, s', f')$:

$$x_1 > x'_2 \cup x_2 < x'_1 \cup y_1 > y'_2 \cup y_2 < y'_1 \quad (\text{空间不重叠})$$

$$\cup s \geq f' \cup f \leq s' \quad (\text{时间不重叠})$$

其中 (x_1, y_1, x_2, y_2) 为调度器执行后为每一个具体硬件任务分配的用于放置的矩形位置的对焦定点坐标, 其中 (x_1, y_1) 为靠近原点的顶点坐标, (x_2, y_2) 为 (x_1, y_1) 对角的顶点坐标。在此假设, 对于可重构片上系统的任务调度模型, 一旦任务调度器为待分配的硬件任务找到至少一个满足上述约束的具体放置位置和任务启动时间 (即有效调度), 则硬件任务称为被接受; 否则硬件任务将被拒绝。从系统运行效率和实用角度来说, 硬件任务的调度成功率将直接影响可重构资源的利用率和整个可重构片上系统的并行性能。

一般来说, 在实现算法是将用到以下几种主要的数据结构:

1. ExTasklist——硬件任务执行链表。该数据结构用于保存正在执行的任务的集合。

2. ResTasklist——预调度任务队列。该数据结构用于保存调度成功后、尚未实际进行任务布置的预留任务的集合。

3. FreeResMap——用于表示可用的器件表面资源分布，该数据结构用于描述FPGA器件上的可重构资源使用状态。

使用上述三种基本的数据结构类型，即可将动态可重构系统的硬件任务调度问题进行较为清晰的简化和规范化。在本文中，我的工作希望可以根据动态可重构片上系统的实时使用情况，包括器件表面各种异构资源的使用情况在线调整任务调度结果。不是一般性，可以认为动态可重构片上系统的表面资源是一个二维矩形的区域。这种近似可以大大简化对问题的描述，但不影响逼近核心问题。因此，在使用的FreeResMap数据结构中，可以使用一个 $W \times H$ 的区域来表示行为 W 、列为 H 宽度的可重构器件资源。在实际的商业化FPGA器件中，其可重构资源也是以此种类似的方式进行组织和管理。

3.2 资源分配策略

3.2.1 算法改进思路

以Xilinx公司为代表的部分动态可重构计算系统最为重要的特点就是快速、方便的对局部区域在不影响其他部分系统运行的情况下进行重新配置。这使得可重构器件的资源利用率可以显著地提升。可重构资源的分配策略问题需要解决的是：当使用可重构硬件任务在线调度算法在可重构器件表面找到了针对当前任务放置需求可用的多个矩形区域的时候，调度器选择哪个矩形区域进行放置最为合理和高效。从数学模型上来说，针对二维矩形抽象模型的硬件任务放置问题类似一个二维的Packing问题，是为NP难问题^[30]。针对传统研究该问题时未考虑硬件任务实时性和通信等，本文工作所关注的问题的解决更为复杂。

相关研究中与资源分配策略相关的工作主要是以下集中策略为主：

第一种策略属于最为简单的方式，即FirstFit策略。一旦放置器为当前的硬件任务找到一个有效的放置位置，则立即使用该区域进行硬件任务放置。不再继续寻求其他更为合理的放置位置。

第二种类型的策略是参照了传统实时任务调度方法，使用各种任务间的属性作为任务调度效率的评价指标，从而给出不同的放置结果，主要有最早完成时间Diff-Finish Time策略，即选择与相邻区域已布置的硬件任务结束时间尽量相同的区域优先放置。这样一来，当相邻的硬件任务执行完毕后，将可以得到面积更大的空闲区域从而有利于新到来的硬件任务运行。

第三种类型是使用一些传统经验规则,试图使矩形块在放置时尽可能的紧密。涉及到了二维邻接策略(2D-Adjacency)、三维邻接策略(3D-Adjacency)等。

一般认为第一类方式时间效率最佳,算法简单易行、策略明了易懂,但无法取得最好的硬件任务放置效果。第二种类型的放置策略借鉴了诸多传统任务调度研究成果,将硬件任务调度时间方面的属性增加到放置策略的考虑中来,显然放置的效果会比第一种方式好,但时间开销会显著增加。第三类策略则是充分利用硬件任务矩形块填充时的特点,使用一些启发式的策略尽量使各个举行任务放置得更为紧密,同时在时间开销和放置效率上取得较好的平衡。

考虑到本文主要的研究工作是在课题组已有工作的基础上进一步增加硬件任务调度模型的实用性,使得相关的技术成果更加接近实用水平。我将在前人工作基础上,提出一种考虑总线和接口优先的放置策略,并利用任务邻接边数最大作为辅助原则,从而尽可能提升任务调度的实际效果。

3.2.2 总线优先放置策略

在课题组前期研究工作中提出了亚可抢占调度算法^[38],以邻接边为任务放置的主要考虑。但结合图 3.3 所示实用化可重构器件结构,在各个可用于布置的可重构资源block之间有专门的通信总线信道。因此,本文所提总线优先放置策略在进行任务放置策略选择时优先考虑尽可能将有通信需求的硬件任务靠近总线放置,再根据最小邻接边算法对放置效果进行评估。

使用一个整数数据DistBus用于记录硬件任务放置过程中与片上总线块资源之间的距离,DistList用于记录可用距离优先位置。由于图 3.3 系统模型中,总线通信资源在分布在从底至上的一个较长的区域内,因此通过计算硬件任务与位置固定的总线资源之间的x轴向距离来评价两者的靠近程度。

在 3.3.1 小节对邻接边进行了详细介绍,某区域的邻接边数包括任务与其他任务的共享边数加上占用的可重构资源边界边数。根据可重构器件资源的相关矩阵,可以很容易的到某区域的邻接边数。具体过程描述如下。

算法 3.1 *GetBusVal(x1,y1,x2,y2)*

1. **Init** DistBus;
2. **IF** Comm=true **then**
2. **for** x2 from i=0 to W **do**
3. DistBus(i)=Bus.x-x2
4. **End for**

DistList=ValidTest(min(Sort(DistBus)))

算法 3.1 主要是计算每个需要放置的硬件任务块与可重构器件表面固定位置的通信总线之间的距离。在这里首先需要尝试在一系列可用于放置的位置,哪个

离总线位置近。位置最近的一系列可用位置存于 $DistList$ 列表中。其中 $ValidTest$ 函数用于对每个候选的放置位置进行有效检测，即该硬件任务周围是否有其他位置被已布置任务占据。从系统实现上来说，这个函数的部分检测功能在后续算法 2 的实现中也有完成，但为表示完整性在此也做有效性检测。

算法 3.2 则用于计算一系列可用于放置的位置其周围邻接边的长度。算法扫描区域 $(x1, y1, x2, y2)$ 四周RCU的权值，如果权值为负，代表该RCU被其他任务占用，因此邻接值 $value$ 加 1。

算法 3.2 $GetAdjVal(x1, y1, x2, y2)$:

1. $value \leftarrow 0$
2. **for** x *from* $x1$ *to* $x2$ **do**
3. $scanadj$;
4. $value \leftarrow value + 1$
5. **end if**
6. **for** y *from* $y1$ *to* $y2$ **do**
7. $scanadj$;
8. $value \leftarrow value + 1$
9. **end for**
10. **return** $value$

因此，通过对于放置时每个具体位置的重新评估，可以使得有通信任务的硬件任务在布置时尽可能的靠近通信信道，如图 3.4 所示。

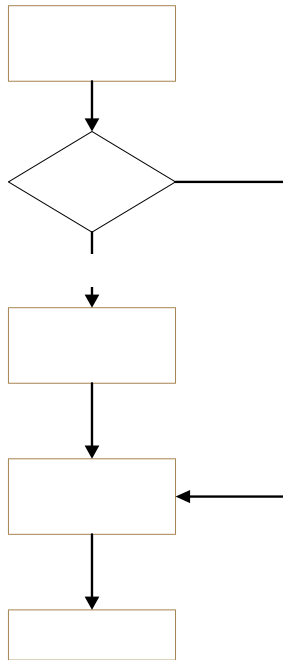


图 3.4 资源分配策略流程图

对于等待找到合适放置位置的硬件任务，首先判断其是否具有任务间通信需求，如果有通信任务则在放置策略选择时优先考虑靠近可重构器件上的总线通信资源；如果没有任务间通信需求，则可直接使用最小邻接边放置算法。任务间通信需求可从硬件任务抽象模型中获得，当实时任务到来时，可以通过查询矩阵 $E(i,j)$ 中边的权值，从而获得硬件任务 T_i 和 T_j 之间的优先关系及通信时间，权值为 $c_{i,j}$ 。

3.3 考虑任务间通信的调度算法

3.3.1 算法改进思路

现有的实时操作系统RTOS在进行实时任务管理时，多数会采用基于任务优先级的管理方法，对实时任务进行可抢占式的实时调度。这种算法主要的目标是保证高优先级的任务能够尽快的得到处理。与此相关的研究非常多，在此不再赘述。与传统处理器软件任务管理不同，运行于可重构片上系统之上的硬件任务考虑目前的实现技术有一个非常现实的难题，即任务切换代价比较大。这实际上意味着，一旦硬件任务由调度器确定还开始时间并已经配置到可重构芯片后，任务的执行就不可逆转了，不能执行中途被抢占或者在任务间进行切换。

与传统CPU执行软件任务切换相比，硬件任务除了要保持相同的寄存器等相关现场数据外，还需要对该硬件任务所占用的可重构区域进行管理，这使得硬件任务切换的开销系统难以承受。现有的研究主要集中在优化硬件任务配置时间等方面，但仍无法有效地降低硬件任务切换总体时间开销^[36-37,45]。

在课题组前期研究中^[38]，可重构硬件任务调度器在任务调度管理过程中，会保留有一个预留任务队列，即一系列已经通过调度器调度成功，但尚未来得及通过放置器配置到可重构器件上实际执行的硬件任务。对于这部分预留任务进行任务切换或者其他更新的调度算法改进，既可以避免实际硬件任务切换的开销，又可有机会改进调度效果。用一个实例进行说明^[38]如下：

如图所示，在 t 时刻，成功调度任务 T_5 。而到 $t+1$ 时刻， T_6 到来时，发现 T_6 找不到有效调度区域。此时， T_5 还未加载到可重构资源上执行，系统剥夺 T_5 所占用的资源，对 T_5 、 T_6 进行统一调度，通过算法成功调度 T_5 、 T_6 。

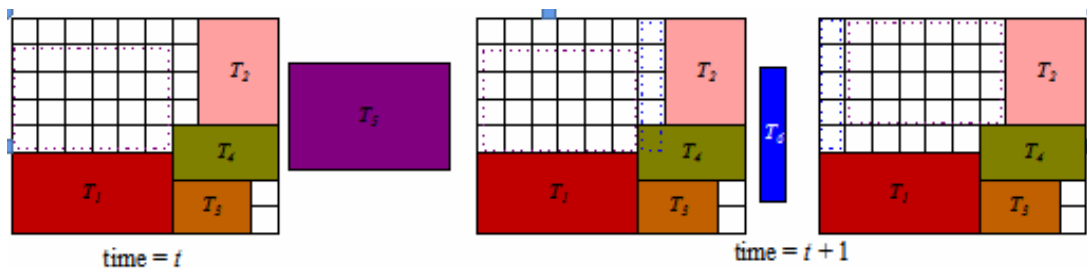


图 3.5 任务调度过程示意图

从图 3.5 中可以看出，对于预留任务队列中的硬件任务进行一定程度上的

统筹安排,可以更为有效的利用可重构器件的资源。由于任务 T_5 先于任务 T_6 到达,因此调度器一般情况下将会先给任务 T_5 找到一个可行的调度。即如图 3.4 中 $\text{time}=t$ 时刻状态所示。单个从 T_5 的调度情况来看,此方案并无不妥。但当任务 T_6 到来时则发现目前可重构器件表面已经没有可以使用的矩形块了。如果在预留任务列表中,能将任务 T_5 和 T_6 的调度顺序进行调整,则可能获得更为良好的调度效果。

考虑到 3.1.3 节的本文所关注的硬件任务调度问题形式化描述,在预留任务列表中使用可抢占的原则针对预留任务进行“二次处理”,即在原有可抢占策略的基础上增加对于任务间通信的考虑。由图 3.3 所示结构模型可知,硬件任务配置完成后,其任务间通信需要通过专用的通信信道和总线完成,而具有通信需求的硬件任务在进行放置时必须考虑这一点,否则任务间通信所带来的面积开销将会大大影响可重构片上系统的实际应用效果。此外,通过对已经进入预留任务队列表的硬件任务优先关系进行分析,调度器也可以在预留队列任务管理时考虑将具有任务间通信需求的硬件任务尽可能的放置在一起并在时间上有连续性。上述约束均是考虑硬件任务实时调度更为接近实用化研究时的新问题和新情况,亦是本文研究的重点。

3.3.2 通信优先可抢占调度算法

虽然考虑的是实时任务调度,任务在到达系统之前具有任务属性不可预知的特点,但是由于可重构片上系统中任务调度设计的特殊性,在预留任务列表中的任务实际上已经可以完全了解其运行信息了,此外新到来的任务若无法立即安排放置,则其任务属性亦可知。在本文的研究中,针对预留任务队列中的已知任务属性的硬件任务使用离线调度算法进行启动时间、放置位置的规划是合理的,但仍需重点考虑调度时间和实时性。对离线任务采用本文提出的总线优先放置策略,对于硬件任务是否具有通信需求区别对待。

可重构计算系统一般用于数据密集型计算,或者作为专用加速单元使用。因此,一般来说可重构计算系统上运行的硬件具有较为重要的实时性要求。在实际的系统实现中,由于商业FPGA芯片进行重构时保存现成、进行任务切换的开销比较大,一般实用系统中未采用抢占式内核来进行硬件任务的调度。在课题组原有工作基础上^[38],本文的工作主要针对任务间通信的需求及可重构器件物理位置的特点进行了进一步的优化。下面简要介绍亚抢占调度算法的基本思路^[38]:该算法考虑到,若新到任务无法找到有效调度时,可以对预调度任务队列中的任务所占用的可重构器件表面资源进行重新分配,从而可以基于一个更为全局和全面的信息对所到达的任务进行调度规划。

首先,如图 3.3 所示,由于实用化的可重构器件表面是典型的异构结构。其可重构资源不再是均匀分布在所有器件表面,其中有诸如外设接口、总线结构、

存储器等各种不同的其他定制化器件。因此，但下层器件结构模型发生了变化，上层调度软件也需要做出相对应的改变。主要有以下几个方面：

第一，对于到达系统的硬件任务，将首先根据任务间的通信需求进行分簇，将有通信任务和优先依赖的任务放在一个可重构配置区域中，即“位置靠近”；

第二，对于到达系统的硬件任务，将根据任务间通信量的大小安排关键任务先执行，即“时间靠近”。

算法的详细描述如下：首先，对于新到达和已经进行有效调度但尚未进行实际布置的预留任务，在预留任务列表中进行信息收集和整合，根据 3.1.3 节本文所提硬件任务形式化描述的属性进行整理。由此可以获得硬件任务的面积属性、各项时间属性、任务间优先关系和通信需求。可以形成类似传统任务DAG抽象模型的数据结构。

本文使用结合了硬件任务执行时间与任务之间的依赖关系对其进行分簇。对节点进行初始化分簇，从当前最后一个硬件任务（虚拟结束节点）开始，依次朝前寻找能与该硬件任务在同一个可重构资源块中执行时间相对较优的硬件任务，直到所有硬件任务的节点或者所有的前趋硬件任务节点都已经被分至其它的簇，递归执行，直至所有的节点都被分至各簇。算法的详细描述如下：

算法 3.3 *Cluster (T,E)* :

1. **For** all Nodes Initial;
2. **From** T_e to T_f
3. **If** (!visited[V_i]) then
4. NewCluster[k], add T_i to Cluster[k];
5. Visited [T_i] = true, $k++$, cur_pos = i;
6. Cluster[k].favo_proc = favo_proc[T_i]
7. **End If**
8. **While** ($T_{current}.Indegree$)
9. **If** not all of Prep ($T_{current}$) have been visited
10. **Chose** T_j from Prep($T_{current}$) **that**
11. T_j (favo_proc) **set** $T_{current}$ execute time minimize;
12. Vistied[T_j] = ture; cur_pos = T_j ;
13. add $T_{current}$ to Cluster[k];
14. udpatd Cluster[k].favo_proc;
15. **End While**

上述分簇算法虽然在初始化的时候需要遍历整个任务队列，时间复杂度为 $O(n \times p^2)$ ，但由于预留任务队列中硬件任务数量非常有限，时间上的开销尚可以承受。

使用算法 3.3 对预留任务队列中的硬件任务进行分簇处理，可以使得具有任务间通信需求的任务尽可能的放在一个可重构资源区域中，避免跨区域的长距离通信现象出现。

通过不同的调度策略，原有亚可抢占调度算法具有一些优势。该算法可以与现有操作系统中广泛采用的其他多种调度策略进行结合。例如，在硬件任务调度时使用考虑任务执行时间最短策略，则类似与传统操作系统中的最短剩余时间有限的调度策略；若使用类似EDF策略进行硬件任务管理，则可以在更为强调任务实时性的情况下，保证更为紧迫的任务优先执行。

与传统的操作系统不同，可重构片上系统的硬件任务管理还必须考虑硬件任务的放置策略。使用更为合理和科学的放置策略，有利于提升硬件任务在线调度的成功率（可以提供更多的空余资源），从而实际上进一步提升了可重构计算系统的并行性和计算效率。但是由于放置问题的难解性，考虑到在线任务调度对于算法运行时开销的限制和要求，本文为了降低可抢占调度的时间开销，采用最大邻接边的调度方式。具体算法过程如下：

算法 3.4 *PreSPSA(T)* :

1. **Cluster**(ResTaskList, E)
2. taskList $\leftarrow$ copyResTask(ResTaskList)
3. TasksCount $\leftarrow$ taskList.addTask(T)
4. maxValu $\leftarrow$ 0, count $\leftarrow$ 0
5. bestTask $\leftarrow$ NULL, newResTaskList $\leftarrow$ NULL
6. **while** tasksCount – count > 0 **do**
7. **for** task **in** taskList **do**
8. value $\leftarrow$ *FindArea*(task)
9. **if** value > maxValu **then**
10. maxValu $\leftarrow$ value, bestTask $\leftarrow$ task
11. **else if** value < 0 **then** count $\leftarrow$ count + 1
12. **end if**
13. **end for**
14. newResTaskList .acceptTask(bestTask)
15. taskList.deleteTask(bestTask)
16. taskCount $\leftarrow$ taskCount – 1
17. **end while**
18. **if** isBetter(taskList, T) return False **then return**
19. **end if**
20. ResTaskList $\leftarrow$ newResTaskList

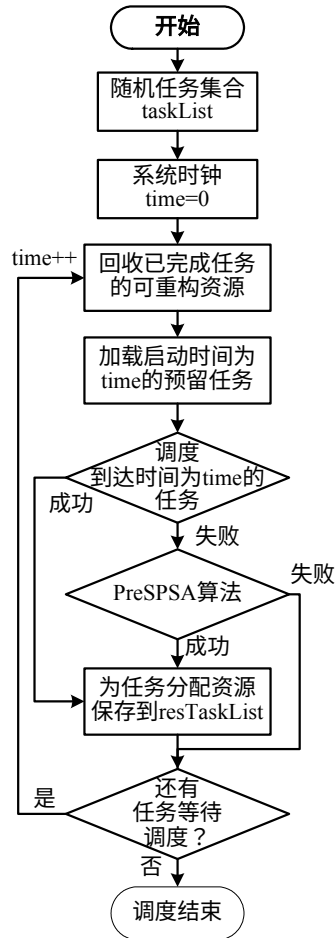


图 3.6 算法 3.4 流程图

PreSPSA算法在原有算法^[38]基础上,首先对预留任务队列中的任务进行分簇,然后再将各个簇内的硬件任务分别使用传统的最大邻接边调度算法进行处理。新到来的任务与预留任务队列中调度成功尚未放置的任务一起分簇后通过copyResTask函数复制到tasklist;然后系统对tasklist中的硬件任务进行调度,使用FindArea()函数对调度的结果进行评价。在此过程中,硬件任务的启动时间将与其具体放置位置协同考虑,并调用本文所提总线优先调度策略进行放置。此时,系统自动根据调度算法的结果进行调整。若PreSPSA算法的调度效果明显好于现存预调度任务队列的调度效果(该算法主要是调整预调度任务队列任务优先顺序)则更新该队列,反之则保持原有调度优先顺序不做变化。

对于调度效果的评价机制,可以根据不同的需求进行设计。本文主要追求可重构资源的最大利用率,在考虑器件面积碎裂程度的情况下,可以通过比较新到来任务所占用面积和布置后的可重构器件表面剩余空间面积之和来判定。若新到来任务所占用面积大于未能成功调度的任务面积之和,则认为这次调度是合理的,效果较好。

分析算法 3.4, PreSPSA与已有的SPSA算法相比,主要是在新任务到达时需要使用分簇策略对所有预留任务列表中的任务进行一次遍历。虽然这个步骤比较耗时,调度完所有任务所需要调度的次数为 $N + (N - 1) + \dots + 1$ 。时间复杂度为 $O(N^2)$,其中 N 为ResTaskList中的任务数加上新任务数。

3.4 片上资源管理

片上资源管理问题,即为针对可重构片上系统中可重构资源模型(如图 3.1所示),需要通过有效的片上资源管理方法对硬件任务的放置进行规划,其目标是降低可重构系统的碎片,提高系统利用率。例如,通过合理的放置规划,使得碎片更少、空闲区域的形状更有利于后续硬件任务的放置。

由于本文的研究目标平台是二维局部可重构系统,因此对于其片上可重构资源的管理主要的方式就是如何更为合理的“拼凑”尽可能多或者尽可能大的矩形。一般来说,现有的可重构矩形资源管理方式主要分为基于矩形的方法、基于相关矩阵的方法和直接扫描法。三种方法共同的技术手段都是需要对由大量可重构单元构成的可重构器件表面进行扫描,从而得到器件表面使用情况的信息。

以往关于硬件任务管理的相关研究中,基于划分最大矩形的方法在进行硬件任务放置策略的制定时候有较大的灵活性。因为在进行任务放置时,只要调度器发现放置器找到了一个比待布置硬件任务面积大并合适的区域即可确定其放置位置。但此方法需要付出找到可重构器件上所有可能的最大矩形的时间开销和代价,否则就无非达到系统的完全可识别性。如果使用相关矩阵的方式对器件资源进行管理,则具有实现简单、鲁棒性高的特点,时间开销处于可以接受的合理范围之内。文献[30]对划分最大可重叠矩形区域空间的方法进行了研究,提出了时间复杂度为 $O(E)$ 的算法,其中 E 为当前执行任务和预留任务数量之和,但实际算法运行过程中由于可重构区域扫描算法的开销使得整体算法开销进一步增加。本文使用课题组前期研究的成果SA算法,其结合相关矩阵与文献[45]中的算法,通过相关矩阵查找可重构资源,采用分割扫描区域的方法查找获得可用于放置的矩形区域。

下面对本文使用的课题组前期研究成果SA算法进行简要介绍^[38]。

首先,可以依照可重构器件的表面可重构单元的分布对应的建立一个映射矩阵。针对二维动态可重构器件,这个映射矩阵也是二维的,其中每一个元素都与可重构器件表面的每个RCU一一对应。当可重构器件表面的RCU资源被占用(即放置了硬件任务),则与其相对应的矩阵中相应位置的值设定为负值;若RCU为空闲资源则矩阵中对应位置的值设定为正值。设定RCU对应的矩阵元素的值为整数。其中,可重构器件表明已经被占用区域,其映射矩阵的值最大为-1;同时,按照 x 轴向使用由低到高,与此被占用区域相邻的、已进行放置的区域则映射矩阵的权

值相差 1；类似的方法，可重构器件表面未被使用的空闲区域相对应的映射矩阵值最小为 1，按y轴向由大到小，彼此相邻的、未被使用的空闲可重构资源映射矩阵值之间相差 1。如图 3.7 所示为一个 8×8 个RCU的相关矩阵抽象模型^[38]。

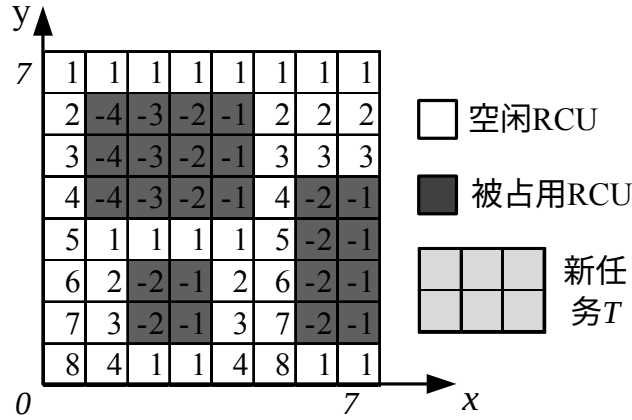


图 3.7 相关矩阵抽象模型

SA算法简单描述如下：

从几点 $O(x, y)$ 开始扫描。在映射矩阵的区域 $(x, y, x + w - 1, y + h - 1)$ 内通过对比矩阵值查找是否存在可重构资源被占用情况。整个矩形区域只要有一个可重构单元被占用，该矩形即为不合格。若此次搜索的结果表明不存在占用情况，则可将本次进行扫描的基点坐标加入到当前硬件任务的有效放置位置列表中。并在 $(x1, y1, x2 - w + 1, y2 - h + 1)$ 区域内，继续沿着x轴和y轴方向逐个进行扫描，若依然存在可重构单元被占用的情况即转第 2 步；若扫描完成后即转第 3 步；

(1) 定位到该矩形区域内被占用的可重构资源块所使用的硬件任务。因为可重构资源块被占用就说明已经有一个硬件任务放置到该区域了，并与当前正在扫描的区域有重叠。因此，找到该硬件任务的右上角坐标，建立新的扫描区域，这样就可以避开该硬件任务已经占用的区域，缩短扫描时间；

(2) 如果可供扫描的区域为空，则整个扫描过程完成；否则继续回到第 1 步。

通过本章的分析，一个较为完整的面向可重构片上系统的硬件任务调度与放置框架基本成型。总的来说，一个硬件任务具备了在可重构系统上执行的条件，则进入就绪任务队列。然后，队列中的硬件任务接受调度器和放置器的协同工作并给出调度结果。其中，调度算法使用本文所提出的PreSPSA算法，而硬件任务放置策略则使用总线优先策略。由此，则可以为每个进入可重构系统的硬件任务安排相对应的启动时间、放置位置等信息，并进入就绪队列。可重构器件本身的配置电路则会将就绪任务队列中的硬件任务依次配置到FPGA芯片执行。

3.5 小结

本章首先介绍了在传统可重构计算系统任务调度模型基础上，增加对于商业可重构器件实际情况的考虑后的一种更为实用的建模方法。并在此基础上提出了改进的可重构硬件任务实时调度算法。首先，在放置策略方面，结合具体可重构器件的通信架构提出了总线优先放置策略，在原有的最小邻接边放置算法的基础上考虑硬件任务与器件总线结构的距离，优化放置实际效果；其次，根据预留硬件任务列表中的硬件任务的优先性和通信约束，根据器件可重构资源区域进行分簇处理，提升调度算法的实际可用性，提高器件利用率。

第 4 章 算法模拟与评价

前面一章主要阐述了在课题组前期研究成果的基础上，本文提出的实时硬件任务调度算法的改进思路和具体算法，主要是针对当前商用FPGA平台的具体情况修改了可重构调度系统模型，在硬件任务调度时增加了对任务间通信、可重构器件总线资源等约束，提出了新的调度算法和可重构资源管理策略。本章以第三章的算法为基础，基于课题组调度模拟平台进行改进，验证和评估所提算法。

4.1 模拟系统简介

课题组通过多年积累，搭建了一个用于可重构片上系统硬件任务调度的模拟平台^[38]，可以专门针对动态可重构硬件任务调度、可重构操作系统以及动态软硬件划分数值仿真进行相关研究。

4.1.1 RCSSimulator介绍

可重构器件的开发具有很多方面的特殊性，例如需要专门学习硬件描述语言，单独熟悉各个器件生产商提供的开发工具和平台等，这些给普通的软件开发人员和团队带来不少困惑。为了能够方便对运行于可重构计算系统之上的调度算法进行评估，我们开发了RCSSimulator。可以通过CPU纯软件方式模拟整个硬件任务的运行状态，虽然与实际FPGA平台级验证和测试有一定差距，但依然可以为前期算法的研究和评估提供良好的平台。

在原有的工作基础上，在模拟器中增加了第 3 章中本文提到的改进的硬件任务调度模型。同时，依然动态可重构片上系统的任务调度流程分为三个部分来管理，即任务调度器、放置器和加载器。其中调度器和放置器一起为硬件任务决定何时启动执行以及在可重构器件表面具体的放置位置，而加载器则负责模拟不同可重构平台进行硬件任务配置时的开销。

与原有系统不同的是，在放置器的具体实现部分增加了本文提出的总线优先放置原则，从而可以在进行硬件任务调度时优先考虑硬件任务的通信需求。此外对调度器中运行的调度算法进行了改进，使得其充分考虑预留任务队列中硬件任务之间的依赖关系和通信约束，并对硬件任务进行分簇管理，从而适应商业FPGA芯片的可重构器件资源的分布。RCSSimulator核心设计框架如图 4.1 所示^[38]。

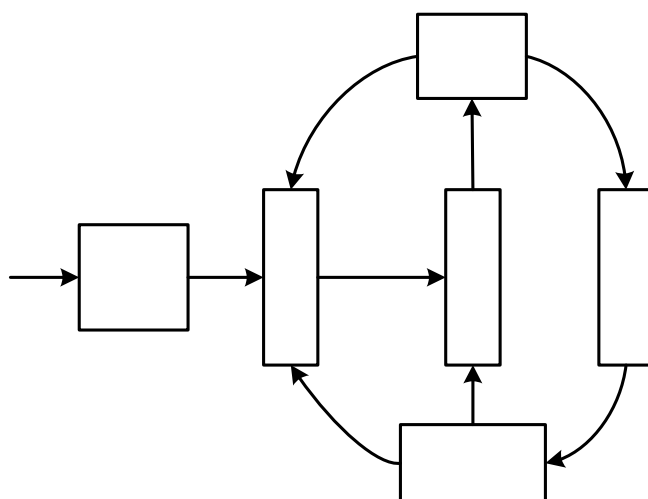


图 4.1 RCSSimulator 架构

RCSSimulator基本运行原理如下：

1. 模拟器的主要目的是对任务调度算法和策略进行模拟和验证，因此既没有真正的硬件任务，也不需要实际的可重构平台。硬件任务将使用随机方法产生并记录到任务列表TaskList中。

2. 模拟器按时间节拍推进。随着系统时间 t 的流动驱动任务列表中的相关任务。例如，随着系统时间推进到 t_1 ，则此时任务列表中硬件任务到达时间 $a = t_1$ 的任务进入系统准备运行，即可将任务传送给调度器进行调度操作。

3. 调度器与放置器协同合作，对到达的硬件任务做进一步的处理。他们将综合器件资源信息、预留任务列表情况、硬件任务间依赖关系和通信需求等约束，寻找该硬件任务的有效调度。

4. 放置器将调度成功的硬件任务根据放置策略放置到可重构器件资源表面。这是使用的是总线优先的放置策略。当硬件任务找到合适的放置位置后，该硬件任务的一次调度过程基本完成，将就入就绪任务列表等待装载执行。

5. 为了充分模拟商业FPGA器件对硬件任务配置的过程，专门设计了装载机模拟FPGA配置过程。在实际的系统模拟过程中，依据系统平台的特性，装载器的数量可以设置为多于1个。

6. “资源模拟管理器”负责对可重构器件的建模。在课题组前期工作基础上，不仅维护了可重构资源映射矩阵，同时还建立了总线资源模型、通信信道模型和外设接口模型及其对应的映射矩阵。

通过使用PreSPSA调度算法，可以对于具有优先性约束和通信依赖的硬件任务进行优化调度，并在放置策略选择时一方面考虑将具有通信需求的硬件任务分簇到一个组从而在同一个可重构区域中执行，另一方面也通过将有通信需求的硬件任务尽可能的靠近可重构器件固定总线资源放置，从而优化通信开销。

在模拟实验中，为了提高任务调度的效率，在新到来的任务可以找到合理调度位置时，可以通过设置选择不启动使用PreSPSA算法进一步优化预留任务队列。使用这个可选优化选项将有利于进一步分析预留任务队列与硬件任务放置之间的关系，提高任务调度成功率。

为了方便调度过程在PC平台上演示，课题组开发了专门的图形用户界面便于直观的了解任务调度过程，如图 4.2 所示。

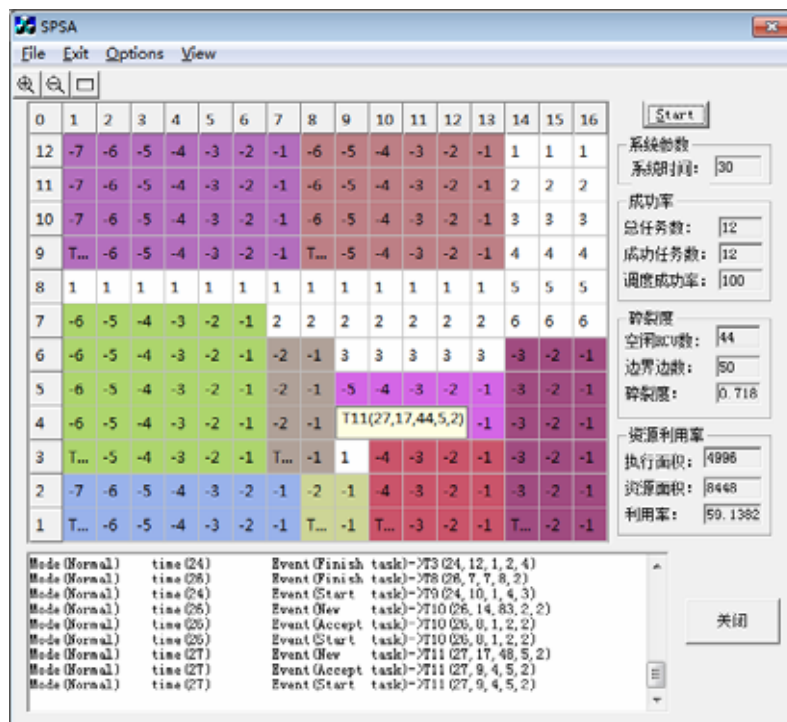


图 4.2 RCSSimulator 图形界面

其中表格部分为资源管理器界面，每个格子的文本内容为相关矩阵中的值。任务占用区域的第一个格子的文本内容为该任务的信息，用一个 5 元组表示 $T_i(a, e, f, w, h, p)$ ，右侧将调度信息实时显示出来。

4.1.2 测试数据集生成

本文中RCSSimulator和测试和数据的生成所使用的PC配置如下：

- CPU：Intel® Core(TM) i3 CPU550 @3.2GHz；
- Memory：2.00 GB；
- Operating System：Windows XP；
- 开发环境：MFC + VC++；

目前在实验中选择多个目标器件进行模拟，较为典型的有Xilinx Virtex XCV1000 型FPGA。其内部的可重构单元可以简单的抽象为具有 64 行×96 列=6114 个RCU。与此同时，具通信总线专用模块和固定外设IO接口。正是由于软件

模拟平台的优越性,通过简单的方式就可以匹配多种型号、复杂的商业FPGA芯片。在本文中,使用与文献[10-11,45]类似的测试任务集产生方法。该测试集可随机生成的硬件模拟任务分为T30、T40和T50三类。 T_n 表示任务的宽度和高度在 $[5, n]$ 内均匀分布,这与现有的可重构IP核所需求的可重构面积基本符合^[45]。任务的运行时间在 $[5, 50]$ 个时间单元内均匀分布。任务的松弛时间($d-a-e$)在 $[1, 100]$ 个时间单元内均匀分布。

与此同时,也使用TGFF随机产生多种不同类型的硬件任务DAG图,并利用随机生成的数据对预留任务队列里的硬件任务 $E(i, j)$ 进行初始化。一般生成的任务图主要按照通信时间与执行时间的比对随机产生的DAG任务分成为CCR=0.1, CCR=1, CCR=10的三大类。具体测试数据生成使用类似使用的方法如下简述,本文主要是在其测试集基础之上增加由TGFF工具产生的任务间随机依赖性和通信关系,并将其应用到预留任务列表的数据中。具体来说就是针对C30、C40、C50测试集随机选取部分任务增加TGFF任务属性。

由于可重构计算系统一般来说主要用于完成面向特定应用的、定制化的专用计算活动,因此其总体的负载不会很高,一般运行其上的任务均是经过较好的规划和组织的。因此,测试集中每组测试任务所需要的可重构资源的总数也是有所控制的。同样使用文献[11,47]的方法,利用负载率计算公式分别产生负载率为0.3、0.5、0.7、1.0、1.5和2.0的测试集进行模拟。

模拟数据采样则是每次运行RCSSimulator进行调度一组测试集,任务数量为1000个,并统计调度结果和资源利用率。不同面积和负载率各生成100组任务进行上述模拟过程,最后取整体平均值用于分析。实验对象选取了Stuffing算法^[10]、KTVS算法^[25]和CR^[45]算法进行对比。

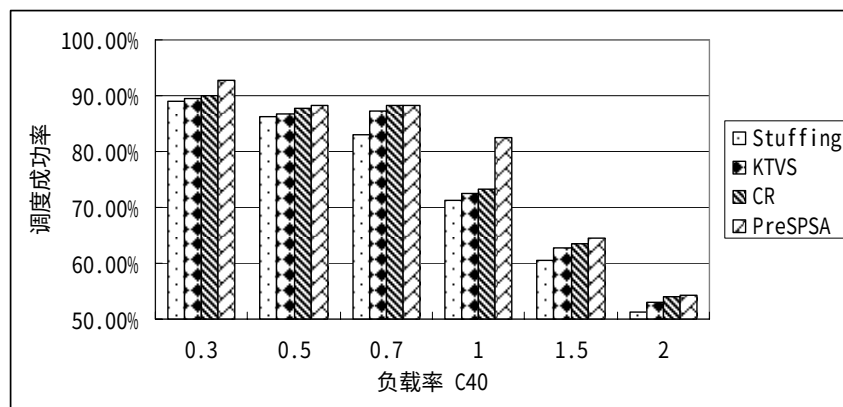
4.2 算法评估与分析

首先,将本文所提出来的算法与上述实验对象的算法进行了比较。考虑到上述算法在原有设计过程中,有部分并未充分考虑通信等因素,因此对其算法进行了改写,再与本文所提算法进行比较。

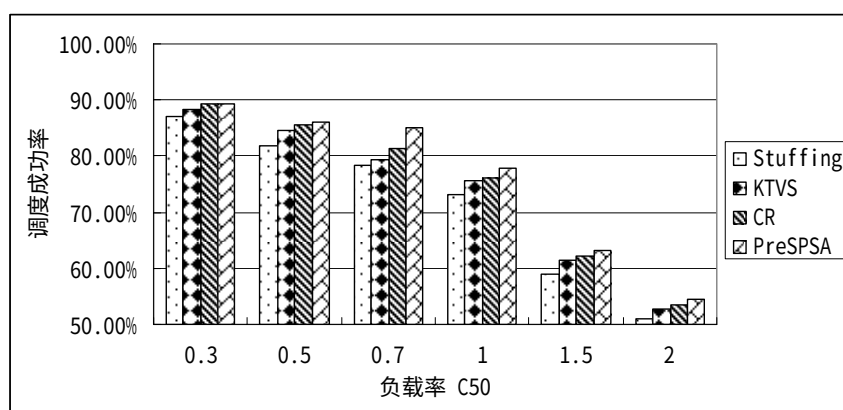
图4.3所示为模拟平台运行RCSSimulator系统及按照4.1.2节原则随机生成的测试任务集进行测试的统计结果。从调度结果的数据统计上来看,本文所提PreSPSA算法相比较Stuffing算法、KTVS算法和CR算法都具有一定的优势。主要的原因在于标准的Stuffing算法的放置策略比较简单,可重构器件表面的碎片难于管理;KTVS算法具有一定的碎片管理能力,但是其算法未有考虑通信等因素,因此在具有通信和任务优先约束的任务集到来时其均作为独立的任务处理,其任务拒绝率会提升;CR算法是目前标准硬件任务抽象模型下调度效果最好的算法,但同样未有考虑任务间通信的特殊性及放置策略。本文所提算法较为接近CR算法

的调度效果，但可以更好的适应实用化的应用场景。

在实际的硬件任务处理时，如果放置策略满足完全可识别性，则总体上来说其对调度成功率影响就不大了，主要是看其算法的执行效率是否满足系统实时性的要求。总的说来，PreSPSA算法可以取得较好的调度效果，主要是因为系统在出现不可接受的任务时，它能进一步使用可抢占的原则对预留任务列表中的任务进行调度，并且在此时考虑任务的通信约束和优先性约束。



(a)



(b)

图 4.3 调度模拟统计结果

此外，对于可重构系统来说，器件的资源利用率也是十分重要的指标。本文对器件的利用率做了粗略的统计，如下图 4.4 所示。

本文提出的PreSPSA算法可以在很大程度上维持可重构器件的资源利用率，因为算法本身的特色之一就是在任务放置时使用最小邻接边的原则，并同时预留任务队列进行可抢占性的调度。这等于同时在调度时间安排和放置位置安排上尽可能快速找到一个次优的方案。但从改进算法提升的效果来看，PreSPSA算法对于可重构器件资源利用率的提升不及调度成功率。

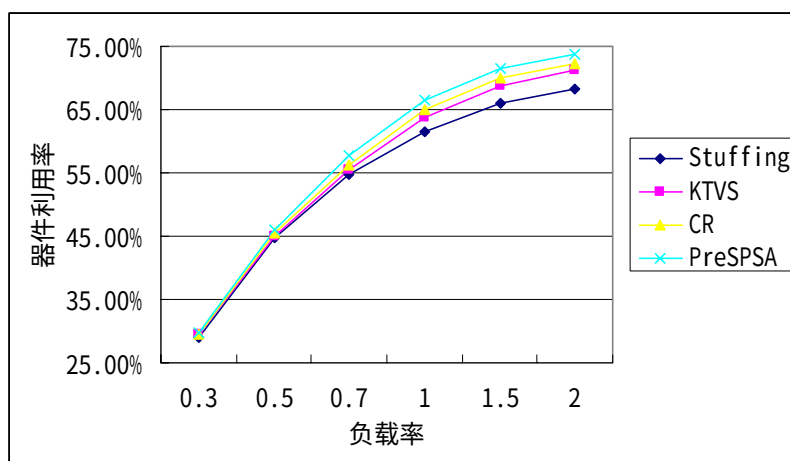


图 4.4 资源利用率统计结果

从上述算法执行时间来分析,使用可重叠最大矩形的算法将会随着硬件任务的增加而运行时间显著上升,适合于大面积、粗粒度硬件任务的管理;KTVS在扫面映射矩阵时只记录放置顶点位置,在被测试算法中执行效率最高,但其不具备完全可识别性;本文所提PreSPSA算法可在相对稳定的执行时间内,处理负载较多的任务,资源利用率和管理方案具有优势。

4.3 原型系统搭建

考虑到本文研究的主要目的就是为了让硬件任务实时调度算法推进到实用,因此基于Xilinx公司的商业FPGA搭建了原型系统,对于动态可重构技术进行了初步的探索和实践。本文研究动态部分重构技术所使用的硬件平台是Xilinx公司提供的Virtex-II Pro开发系统,如图 4.5 所示。所使用到的开发工具主要有Xilinx EDK9.1, Xilinx ISE9.1, ModelSim6.1。

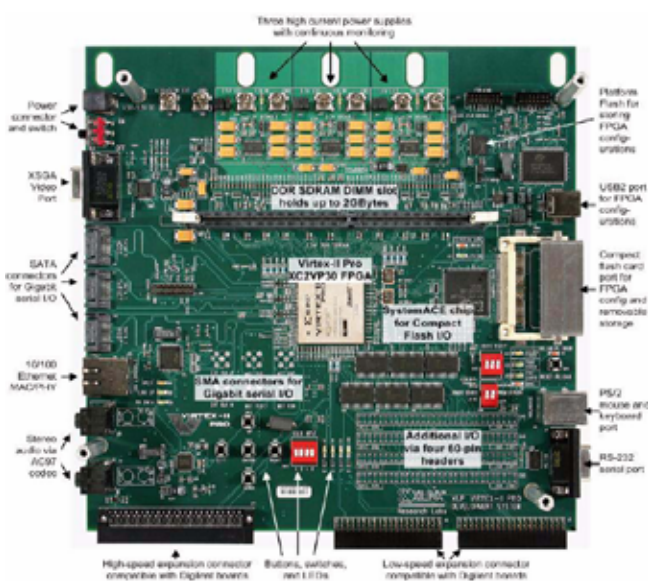


图 4.5 Virtex-II Pro 开发板

开发板集合Xilinx Virtex-II Pro系列的XC2VP30 型号FPGA以及环绕FPGA的各种外设组件，专门用于各种应用系统的评估和快速开发与设计。XC2VP30 FPGA是Xilinx第一代推出的支持部分动态可重构的产品。由于近年来Xilinx公司对部分可重构技术采取了预约与授权机制，因此XC2VP30 平台是目前所能获得的最为方便的硬件原型平台。该平台具有丰富的外围接口，并且在FPGA资源的内部可以通过软件配置两个标准的PowerPC内核，从而实现多核架构，为基于IP核的高端SoC设计提供快速原型系统。从编程方式上来说，该系列的FPGA芯片可以提供良好的图形用户界面，为芯片引脚的配置和使用提供基于脚本的编辑方式，从底层器件层到顶层设计都可以在一个IDE界面完成。与此同时，对于原型板上芯片资源的配置可以灵活的采用多种方式进行。一方面可以通过USB电缆与上位PC机连接，通过软件的边界扫描功能进行芯片数据下载；另一方面也可以通过对板上FLASH芯片的配置、将下载数据写入到FLASH中，从而使得每次上电运行时均可以自动由FLASH中读入配置文件进行重构操作。相关具体数据和操作细节可以参考其数据手册，在此不再赘述。

在部分动态可重构系统中，一般来说是对固定的区域进行重构。在现有技术条件下尚未实现对任意区域的任意配置。在本文的实现案例中，将部分用于演示的核心算法使用标准库进行封装，从而形成相对固定的IP核；然后，可以将其挂载到OPB总线上。通过ISE集成开发环境提供的部分动态可重构工具，可以通过软件的方式对挂载在OPB总线上的固定的IP核进行重构操作。在进行此项相关配置的时候，并不会影响系统其他部分的正常运行和使用。实际上，是通过在芯片内部的PowerPC芯片上编写程序，通过从外部CF存储器上读取用于动态可重构配置的配置数据文件（一般来说，用于动态可重构配置的数据文件较大，均存储在外部存储芯片中），然后使用系统提供的ICAP接口将可重构配置数据下载到FPGA中来完成重构模块间的功能置换。因此这里仅仅简单介绍主要涉及的几个核心期间和技术。

1. PowerPC405 硬微处理器

本文中使用的FPGA平台现在支持两种方式的嵌入式处理器。一种是硬核处理器，一种是软核处理器。对于XC2VP30 系列的原型系统平台来说，Xilinx公司提供了基于PowerPC架构的硬核，可以支持 32 位的高性能嵌入式应用。这是目前业界主流的嵌入式处理器硬核。此外，该平台还支持命名为MicroBlaze的软核处理器。该软核处理器可以通过ISE集成开发环境进行配置，对处理器指令集、总线结构和外设进行优化。显然，硬核可以提供更好的性能但是没有软核具有的灵活性。在本文的具体实现中，使用硬核处理器PowerPC405。

PowerPC405 硬微处理器是一种利用先进的IP内核植入技术开发的、面向嵌入式高性能应用的PSoC核心处理器。利用IP内核植入技术，可以将PowerPC内核放

置在Virtex-II系列FPGA中的任意位置从而为更为灵活、便利的使用此 32 位CPU提供机会。这种灵活的放置并不会带来更多的负担，PowerPC处理器可以很好的与周围的硬件资源、接口和专用计算模块互联。与此同时，由于均使用来自于Xilinx的集成技术，使用PowerPC处理器作为配置管理的核心单元，可以更为方便的对FPGA内部的总线及动态重构资源进行配置，从而进一步提高可重构计算平台的性能。

2. ICAP

一般来说，对FPGA芯片进行配置数据下载、重写或者更新均通过类似于JTAG接口这样的并行传送端口。这样的接口一般是独立于芯片之外，需要在FPGA原型系统平台上配合设计相应的辅助功能芯片才能实现。如现有Altera公司的FPGA就通过专用的MasterBlaze下载电缆来实现对FGPA芯片的配置和下载。但此种方式由于受到原型系统板设计、下载数据电缆设计等多方面的影响在数据传输效率、使用的方便性和灵活性等方面均有极大的限制。最为重要的是，以此种方式进行FPGA芯片配置每次都需要对整个芯片进行重写，无法实现部分动态可重构配置的目标。

针对这一系列问题，特别是为了支持Xilinx公司芯片的动态部分可重构技术，该公司从Virtex-II系列芯片开始，在芯片内部设计了ICAP模块专门用于提供芯片动态可重构数据加载技术支持。通过使用ICAP工具，用户可以通过FPGA芯片上的内置处理器内核对该芯片其他部分的可重构资源进行配置和修改。这实际上正是进行部分动态可重构的基础。从具体实现上来说，ICAP在FPGA内部以硬件原语的方式使用，允许用户实现FPGA的配置以及配置数据的回读。使用ICAP要遵循以下原则^[38,39]：

(1) 如果对FPGA进行配置后，SelectMAP不能与ICAP同时使用。

若PERSIST位置为 1，则表示选择使用SelectMAP方式配置数据，引脚DONE的信号为高电平，芯片的ICAP将处于不可使用状态。

若PERSIST位置为 0，则表示选择使用SelectMAP方式配置数据，SelectMAP的引脚D[0:7]、RDWR_B、CS_B、BUSY和INIT_B等多个端口将表示为通用用户I/O状态。芯片的ICAP功能处于工作状态。

(2) 如果同时使用JTAG和ICAP两种方式对FPGA芯片的配置数据进行操作，则需要注意以下几个情况：

当在使用ICAP进行FPGA芯片内部可重构资源配置或者数据读写操作时，JTAG操作相关的信号和指令不能影响到当前FPGA的运行，比如，JTAG CFG_IN、CFG_OUT、JSTART、JSHUTDOWN等指令不能通过下载电缆写入到FPGA。

当使用JTAG端口进行配置或者配置数据回读时，需要在完成相关操作后，若需使用ICAP原语则需要与JTAG的命令保持同步；反之，使用ICAP接口完成配置

或者回读配置数据操作后，在使用JTAG接口进行任何操作以前配置逻辑也需保持同步。

3. System ACE

由于ICAP在使用上还是存在诸多限制，因此在实现本文所提的部分动态可重构系统时，需要进行初始配置文件下载。初始配置文件的下载和保存有两种方式，既可以保存在芯片内部 PROM 中，也可以保存在外部存储设备中。对于实验性质的开发或者验证，可以将较小的配置文件保存在 PROM 中使用，但是考虑到本文所涉及的系统较为复杂，因此，需要使用 System ACE 解决方案来进行片外存储器配置文件保存方法。在系统的实际实现中，可以使用 System ACE 机制将配置数据保存在外接的 Flash 存储卡中（容量足够保存大多数应用配置文件）。然后，用户可以控制 FPGA 内核，通过 System ACE 控制器把数据写入到 HWICAP IP 核的缓冲存储空间中。System ACE CF 存储设备包括 Xilinx 的 ACE Flash 卡或者其他厂家的 Compact Flash 卡。

在所开发的部分动态可重构原型系统上，以 128 位 AES 和 DES 算法为研究对象，实验和分析硬件任务实时调度的性能和可实现性。首先，需要将上述加密算法模块进行封装，让其以 RTL 级实现方式封装成 IP 模块，并挂到 OPB 总线。系统用户，可以通过运行于 PowerPC405 芯片上的程序，通过调用对应的 IP 模块（硬件任务）和对应的硬件函数来实现各种数据加密功能。在的系统实现中，与上述两个算法相对应的硬件函数分别为 OPB_DES() 和 OPB_AES()。在 PowerPC 上运行的应用程序或者系统进程，需要使用到 AES 或者 DES 加密算法时，可以调用对应的硬件函数。调用前先检查相关的 IP 模块是否已经配置到 FPGA 中并且挂载到 OPB 总线上。如果相关操作已经完成，则可以直接运行；如果配置工作尚未完成则需先从外部 FLASH 卡中读取配置文件进行配置，并通过 ICAP 完成部分重构数据的重构。实际上，当有新的硬件函数实现需要更新的时候，均可使用 ICAP 工具在不影响当前系统运行的情况下完成关键硬件算法的更新。

在使用部分动态可重构的过程中，本文所考虑的两种加密算法可以在 PowerPC 处理器上运行的硬件任务调度进程的管理下选择利用动态重构区域的资源来实现功能的切换。这实际上等于使用较少的资源实现了多种复杂功能的动态切换，从根本上提高了可重构资源的利用率。使用此种方式所付出的开销就是时间，也就是“时间换空间”。此处的时间开销即包括进行部分动态可重构配置所消耗的配置时间，也包括由于不能同时运行两个加解密算法所影响的应用程序开销。如果应用程序规模比较大，需要处理的数据比较多，则重构配置时间会显著增加，此时应该尽可能的让每一个配置运行时间长一些，从而减少配置开销对整体效率的影响。

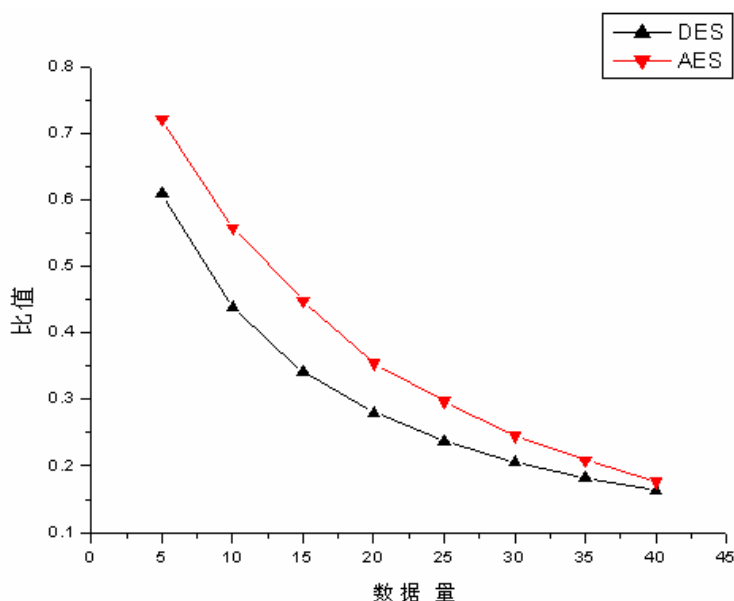


图 4.6 模块动态重构开销

考虑到配置时间的重要性，在上述平台上对实际配置开销与数据量的关系进行测试。实验中，XC2VP30 FPGA 上，AES 的配置需时为 2.14 秒，DES 的配置需时为 1.22 秒。图 4.6 中所示为 AES 和 DES 的重构配置开销随着数据量大小变化所占的比值图，数据量表示需要处理的数据的个数，其中 AES 每个数据大小为 128bit，DES 每个数据大小为 64bit。随着数据量的增大，重构配置开销在系统的总的运行时间比变小。

4.4 小结

本章以第三章的算法为基础，在模拟器 RCSSimulator 中增加了改进的硬件任务调度模型。同时，依然用动态可重构片上系统的任务调度流程来管理，但在放置器的具体实现部分增加了本文提出的总线优先放置原则。在模拟实验中，为了提高任务调度的效率，在新到来的任务可以找到合理调度位置时，可以通过设置选择不启动使用 PreSPSA 算法进一步优化预留任务队列。得出 PreSPSA 算法可以在很大程度上维持可重构器件的资源利用率，因为算法本身的特色之一就是在任务放置时使用最小邻接边的原则，并同时预留任务队列进行可抢占性的调度。

结 论

虽然动态部分可重构技术相关研究已经引起了学术界和工业界的广泛重视，但有关可重构操作系统和实时硬件任务管理方面的研究成果离实际应用还有很长的路要走。作为可重构操作系统的核心能力部件——硬件任务调度器及其实现的相关工作才起步十余年，伴随着可重构器件硬件技术的飞速发展，必将需要研究人员在此方向上持续付出更多努力。

本文以提高可重构片上系统中实时硬件任务在线调度算法的实用性为目标，结合课题组前期研究工作，面向实际商业FPGA芯片的结构特点，有针对性的改进了硬件任务调度系统抽象模型，并以此为基础提出了改进的调度算法。本文完成的主要工作如下：

1. 根据更为实用的可重构器件资源模型，提出改进的可重构硬件任务形式化模型和调度系统模型。对于可重构器件进行建模的方法主要是将可重构资源抽象为一个可分配的矩形，此种方式可以降低硬件任务调度研究的难度，以便在充分考虑任务调度特性的同时，尽可能的减少底层具体硬件细节对于上层调度的影响。本文在对可重构器件资源建模时考虑到新的可重构器件资源的异构性，在矩形抽象模型的基础上，增加对于可重构器件总线及接口资源的描述，从而可以在更为准确的系统模型上进行可重构硬件任务实时调度研究，为后续相关研究奠定基础；

2. 在前人研究基础之上，提出PreSPSA调度算法。原有调度算法主要是基于传统的矩形可重构系统资源模型设计，未能考虑硬件任务间通信的情况。而在实际应用中一般均存在不同程度的任务间通信。所提算法针对上述问题重点考虑任务间通信对于实时调度和分配的影响，在可重构资源分配时使用新的基于总线优先的分配策略，在任务调度时使用考虑任务间通信的可抢占调度算法对预留任务队列进行规划，从而提升任务调度效率。实验结果表明，该算法可以有效的提升可重构系统的调度成功率。

3. 论文在上述两项理论改进工作的指导下，改进了可重构硬件任务调度模拟系统RCSSimulator，增加了其中系统模型部分关于异构器件资源的表达。进一步利用Xilinx可重构计算平台构建了部分动态可重构计算系统。基于该系统使用部分动态可重构技术运行了AES和DES算法实例，验证了部分动态可重构技术的优势。

文中提出的PreSPSA调度算法主要思路依然是在实时调度的场景下，针对预留任务队列使用部分离线任务调度的方式对硬件任务进行重新的规划，以期获得更好的调度效果。随着部分可重构技术的进一步发展，以开发实用性为主的异构

动态可重构片上系统核心技术为目标，未来还有很多工作可以持续探索：

首先，可重构逻辑器件上运行的硬件任务之间进行切换时带来的开销一直是困扰其进一步推广应用的主要因素之一。随着微电子技术的进一步发展，硬件任务切换开销有望降低到一个较为可以接受的水平。因此，未来考虑切换开销情况下的、以可抢占技术为基础的动态硬件任务调度算法的研究是重点研究内容之一。

其次，适合于异构可重构片上系统使用的操作系统是研究的最终目标，但实际系统的实现均依赖于实际可重构器件平台。因此，在研究更为接近实际的可重构器件模型和调度算法的基础上，未来的主要工作是以实际商用可重构器件为基础逐步构建具有实时调度与分配、动态可重构和切换的调度器及内核，从而构建以FPGA器件为实际运行平台的验证系统。进一步推进更为实用的技术的研究与开发既是本文工作的初衷，亦是未来该领域的研究重点。

参考文献

- [1] 段然, 樊晓桢, 高德远等. 可重构计算技术及其发展趋势. 计算机应用研究, 2004, (8): 14-17
- [2] Hartenstein R. Trends in reconfigurable logic and reconfigurable computing. proceedings of the Electronics, Circuits and Systems, 2002 9th International Conference on, 2002, 801-808
- [3] PACT Corporation[EB/OL]. <http://www.pactcorp.com>, 2008, 8
- [4] Estrin G, Bussell B, Turn R, et al. Parallel Processing in a Restructurable Computer System. Electronic Computers, IEEE Transactions on, 1963, EC-12(6): 747-755
- [5] 贾英江, 高欣宝. 可重构计算机简介. 计算机应用研究, 1999, 16(5): 4-6
- [6] Xilinx Corporaton[EB/OL]. <http://www.xilinx.com>, 2008, 8
- [7] Compton K, Hauck S. Reconfigurable computing: a survey of systems and software. ACM Comput Surv, 2002, 34(2): 171-210
- [8] Walder H, Platzner M. Non-preemptive multitasking on FPGAs: Task placement and footprint transform. proceedings of the Proceedings of the 2nd International Conference on Engineering of Reconfigurable Systems and Architectures (ERSA), 2002, 24-30
- [9] Ahmadinia A, Bobda C, Teich J, et al. Temporal task clustering for online placement on reconfigurable hardware. proceedings of the 2003 IEEE International Conference on Field-Programmable Technology (FPT), 2003 Proceedings, 2003. 359-362
- [10] Steiger C, Walder H, Platzner M, et al. Operating systems for reconfigurable embedded platforms: Online scheduling of real-time tasks. IEEE Transactions on Computers, 2004, 53(11): 1393-1407
- [11] Steiger C, Walder H, Platzner M, et al. Online scheduling and placement of real-time tasks to partially reconfigurable devices. proceedings of the 24th IEEE Real-Time Systems Symposium, 2003 RTSS 2003, 2003, 224-225
- [12] Brebner G, Diessel O. Chip-based reconfigurable task management. proceedings of the Field-Programmable Logic and Applications, 2001, 182-191
- [13] Hubner M, Schuck C, Becker J, et al. Elementary block based 2-dimensional dynamic and partial reconfiguration for Virtex-II FPGAs. proceedings of the

- Parallel and Distributed Processing Symposium, 2006 IPDPS 2006 20th International, 2006, 8
- [14] Van D V, Fekete S, Majer M, et al. Defragmenting the module layout of a partially reconfigurable device. Arxiv preprint csAR/0505005, 2005
- [15] Xilinx I. Virtex-4 Configuration Guide[EB/OL]. <http://www.xilinx.com>, 2009, 3
- [16] Xilinx I. Virtex-5 Configuration User Guide[EB/OL]. <http://www.xilinx.com>, 2009, 3
- [17] Bazargan K, Kastner R, Sarrafzadeh M, et al. Fast template placement for reconfigurable computing systems. IEEE Design & Test of Computers, 2000, 17(1): 68-83
- [18] Walder H, Steiger C, Platzner M, et al. Fast online task placement on FPGAs: free space partitioning and 2D-hashing. proceedings of the Parallel and Distributed Processing Symposium, 2003 Proceedings International, 2003, 8
- [19] Ahmadiania A, Bobda C, Bednara M, et al. A new approach for on-line placement on reconfigurable devices. proceedings of the Parallel and Distributed Processing Symposium, 2004 Proceedings 18th International, 2004
- [20] Handa M, Vemuri R. An efficient algorithm for finding empty space for online FPGA placement. proceedings of the Proceedings of the 41st annual Design Automation Conference, 2004, 960-965
- [21] Tabero J, Septién J, Mecha H, et al. A low fragmentation heuristic for task placement in 2d rtr hw management. Field Programmable Logic and Application, 241-250
- [22] Septién J, Mecha H, Mozos D, et al. 2D defragmentation heuristics for hardware multitasking on reconfigurable devices. proceedings of the Parallel and Distributed Processing Symposium, 2006 IPDPS 2006 20th International, 2006, 7
- [23] Zhu Y. Efficient processor allocation strategies for mesh-connected parallel computers. Journal of Parallel and Distributed Computing, 1992, 16(4): 328-337
- [24] Chiu G, Chen S. An efficient submesh allocation scheme for two-dimensional meshes with little overhead. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(5): 471-486
- [25] Wu F, Hsu C, Chou L, et al. Processor allocation in the mesh multiprocessors using the leapfrog method. IEEE Transactions on Parallel and Distributed Systems, 2003, 14(3): 276-289
- [26] Healy P, Creavin M, Kuusik A, et al. An optimal algorithm for rectangle placement. Operations Research Letters, 1999, 24(1-2): 73-80

- [27] Orlowski M. A new algorithm for the largest empty rectangle problem. *Algorithmica*, 1990, 5(1): 65-73
- [28] Edmonds J, Gryz J, Liang D, et al. Mining for empty spaces in large data sets. *Theoretical Computer Science*, 2003, 296(3): 435-452
- [29] Lu Y, Marconi T, Gaydadjiev G, et al. An efficient algorithm for free resources management on the FPGA. *proceedings of the Proceedings of the conference on Design, automation and test in Europe*, 2008, 1095-1098
- [30] Joseph Y, Tam T, Wong C, et al. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 1990, 10(3): 271-275
- [31] 黄文奇, 朱虹, 许向阳等. 求解方格packing问题的启发式算法. *计算机学报*, 1993, 16(11): 829-836
- [32] 陈端兵, 黄文奇. 一种求解矩形块装填问题的拟人算法. *计算机科学*, 2006, 33(5): 234-237
- [33] 黄文奇, 刘景发. 基于欧氏距离的矩形Packing问题的确定性启发式求解算法. *计算机学报*, 2006, 29(5): 734-739
- [34] 齐骥, 李曦, 于海晨等. 一种面向动态可重构计算的调度算法. *计算机研究与发展*, 2007, 44(8): 1439-1447
- [35] Huang W, Cao X, Wang B, et al. An Online Scheduling Algorithm for Reconfigurable Tasks Based on Dynamic Planning Preemptive Threshold. *proceedings of the CiSE 2009 International Conference on*, 2009, 1-4
- [36] Redaelli F, Santambrogio M, Sciuto D, et al. Task scheduling with configuration prefetching and anti-fragmentation techniques on dynamically reconfigurable systems. *proceedings of the Proceedings of the conference on Design, automation and test in Europe*, 2008, 519-522
- [37] Redaelli F, Santambrogio M, Memik S, et al. An ilp formulation for the task graph scheduling problem tailored to bi-dimensional reconfigurable architectures. *International Journal of Reconfigurable Computing*, 2009, 1-15
- [38] 许新达. 基于局部可重构计算的在线硬件任务调度算法研究: [湖南大学硕士学位论文]. 湖南: 计算机与通信学院, 2010, 33-40
- [39] 赵远宁. 基于Xilinx Virtex-II Pro的过程级动态部分可重构系统设计与实现: [湖南大学硕士学位论文]. 湖南: 计算机与通信学院, 2008, 35-45
- [40] 梁克, 张凯龙, 周兴社等. 可重构硬件操作系统技术. *小型微型计算机系统*, 2007, (11): 2047-2050
- [41] Tessier R, Burleson W. Reconfigurable Computing for Digital Signal Processing: A Survey. *J VLSI Signal Process Syst*, 2001, 28(1/2): 7-27

- [42] 曲英杰, 王沁, 王昭顺等. 可重构体系结构的特征及应用. 计算机工程与应用, 2001, 17(8): 19-21+41
- [43] Septién J, Mozos D, Mecha H, et al. Perimeter quadrature-based metric for estimating FPGA fragmentation in 2D HW multitasking. proceedings of the Parallel and Distributed Processing, IEEE international Symposium on, 2008, 1-8
- [44] 周学功, 梁樑, 黄勋章等. 可重构系统中的实时任务在线调度与放置算法. 计算机学报, 2007, 30(11): 1901-1909
- [45] 齐骥, 李曦, 胡楠等. 基于硬件任务顶点的可重构系统资源管理算法. 电子学报, 2006, (11): 2094-2098
- [46] Abbott Al, Athanas Pm, Chen L, et al. Finding lines and building pyramids with SPLASH 2. proceedings of the FPGAs for Custom Computing Machines, 1994 Proceedings IEEE Workshop on, 1994, 155-163
- [47] Buell D, Arnold J, Kleinfelder W, et al. Splash 2: FPGAs in a custom computing machine. IEEE Computer Society, 1996
- [48] 李丽, 辛勤, 杨乐平等. 基于FPGA的可重构系统的应用. 微处理机, 2001, (3): 11-13
- [49] Hauser J, Wawrzynek J. Garp: A MIPS processor with a reconfigurable coprocessor. proceedings of the FPGAs for Custom Computing Machines, 1997 Proceedings, The 5th Annual IEEE Symposium on, 1997, 12-21
- [50] Singh H, Lee M, Lu G, et al. MorphoSys: An integrated re-configurable architecture. proceedings of the Proceedings of the NATO RTO Symp on System Concepts and Integration, Monterey, CA, USA, 1998, 20-22
- [51] Singh H, Lee M, Lu G, et al. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications. IEEE Transactions on Computers, 2000, 49(5): 465-481
- [52] PACT corporation[EB/OL]. <http://www.pactcorp.com>, 2009, 1
- [53] 邹祎. 支持动态可重构硬件透明编程操作系统的任务调度研究: [湖南大学硕士学位论文], 2008
- [54] 南希, 龚龙庆, 田卫等. 基于FPGA的动态可重构系统设计与实现. 现代电子技术, 2009, 32(6): 4-7+11
- [55] Noguera J, Badia R. HW/SW codesign techniques for dynamically reconfigurable architectures. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2002, 10(4): 399-415
- [56] Todman T, Constantinides G, Wilton S, et al. Reconfigurable computing: architectures and design methods. IEE Proceedings-Computers and Digital

- Techniques, 2005, 152(2): 193-207
- [57] Wigley G, Kearney D. Research issues in operating systems for reconfigurable computing. proceedings of the proceedings of the International Conference on Engineering of Reconfigurable System and Algorithms (ERSA), 2002, 10-16
- [58] 蔡一茂, 赵元富, 兰利东等. 基于国产SoPC平台的外部总线的设计与实现. 微电子学与计算机, 2012, 29(5): 15-19
- [59] 赵宏阳, 丁晓明. 基于FPGA的高速多路视频数据采集系统. 单片机与嵌入式系统应用, 2012, 12(7): 123-135
- [60] 唐柳, 黄樟钦, 侯义斌等. 计算机工程与应用. 可重构多核片上系统体系结构综述, 2012, 48(20): 56-62
- [61] Greenbaum J. Reconfigurable logic in SoC systems. proceedings of the Custom Integrated Circuits Conference, 2002 Proceedings of the IEEE 2002, 2002, 5-8

致 谢

三年的研究生生活即将结束。回顾这三年的求学经历，一路走来，无限感慨。由衷的感谢给予我关心和呵护、支持和鼓励的所有人。

首先感谢的是我的导师，是一位品德高尚、待人诚恳的老师。论文从选题、开题，到研究路线和最终的撰写，无不凝聚着导师的大量心血。导师渊博的知识、对待工作的热情以及孜孜以求的工作作风，是我学习的榜样，也必定会让我受益终身。在此，我谨向您表示衷心的感谢！

感谢湖南大学计算机与通信学院的老师和工作人员，是你们提供给了我良好的学习、科研环境，是湖南大学栽培了我。

感谢与我日夜相伴的师兄弟，是你们的陪伴才让我在学习、科研过程中得到更大的乐趣。

感谢我的爸爸妈妈，是你们给我有利的学习环境。感谢我的爱人对我工作的支持帮助，帮助我翻阅文献，你的默默关心让我狂燥的心情得已平复。

最后，再一次感谢所有帮助我的人和我爱的人，是你们给了我勇往直前的信念！谢谢你们！

2013 年 3 月