绪论

* 1. 研究背景
     1. 可重构计算概述

随着科技水平不断发展，人们生活的需求越来越复杂，各种新型应用层出不穷。它们普遍具有运算复杂度高、处理数据量大等特点，在性能、功耗、灵活性、集成度和成本等方面对嵌入式移动终端提出了十分苛刻的要求。为了实现这些计算密集型的应用，主要采用两种方法：一种是在专用集成电路（Application Specific Integrated Circuit，ASIC）上用硬件实现，另一种是在通用处理器（General Purpose Processor，GPP）上用软件实现。ASIC具有高的能量效率、面积效率，在性能、功耗、芯片面积等方面能够进行很好的优化，但是它只能针对某一特定应用或者某一特定算法设计，灵活性低下，很难满足更多的新型应用需求，难以升级，重用性差；GPP则具有很强的灵活性，可适用于各种不同的应用领域，但指令流驱动的执行方式导致其整体性能和功耗并不理想，并且运行速度慢、效率低，难以满足应用的性能要求。

在这种背景之下，可重构计算技术应运而生。可重构计算同时兼顾了通用处理器与ASIC的优点，既保留了通用处理器的灵活性，也具有ASIC的高效性，能够比较好地满足众多复杂应用的计算需求。早在20世纪60年代，美国加州大学的Gerald Estrin就提出了可重构计算的概念[1]，将一个通用处理器连接一个计算阵列，通用处理器负责控制计算阵列运行，计算阵列可以接收配置信息执行特定的功能，两者结合起来，通过通用处理器对计算阵列的配置信息进行管理，就可以让计算阵列执行不同的任务，从而灵活性大大提高。限制于工艺水平，直到上世纪90年代可重构计算才重新获得重视，成为学术界和产业界的热点。1999年加州大学伯克利分校可重构技术研究中心的Wawrzynek和Dehon提出可重构计算的另外一种定义[2]，它将其视为一类计算机组织结构，有区别于其他组织结构的两类突出特点：制造后芯片的定制能力，即硅实现以后计算功能依旧可以按需改变，区别于传统的专用集成电路；能针对很多算法完成计算引擎的空间映射（区别于传统的指令驱动处理器）。

按照重构粒度划分，可以分为细粒度和粗粒度，这里的粒度是指可重构计算数据通路中运算单元的数据位宽。一般情况下，重构粒度越大可重构计算处理所需的配置信息就越少，重构的速度就越快，相应的功能灵活性也越低。粗粒度可重构架构适用于计算密集型的应用，如通信应用[3][4]、密码处理应用[5]和多媒体应用[6][7]等；近些年来，已经有很多国内外的公司和科研机构提出了多种不同的可重构架构，如REMAR[8]、PipeRench[9]、ADRE[10]、MorphySy[11]、XPP[12]、Zippy[13]等。

* + 1. 密码算法综述

随着计算机技术和网络通信技术的发展，信息安全问题也逐渐成为人们关注的社会问题，密码技术是保证信息的可用性、机密性和安全性等安全要求的基本手段。密码算法是各种安全应用的基础，也是信息系统安全性的根本所在，高效灵活的密码算法是各种高性能信息系统的重要指标和基本保障，因此成为信息安全领域的重要课题。

密码算法又可分为公钥密码、私钥密码和哈希函数三大类，其中，公钥密码主要是大数的模幂操作，私钥密码和哈希函数则以逻辑运算、移位、置换、替换为基础的迭代操作。密码算法按照密钥特点又被分为对称密码和非对称密码，对称密码在加密和解密时都使用相同的密钥，也称为传统密码，是20世纪70年代在公钥密码产生之前唯一的加密类型，迄今为止，在两种加密类型中仍然是使用最为广泛的加密类型，对称密码主要分为流密码算法和分组密码算法。非对称密码在加密和解密时则会使用不同的密钥，并且很难从一个推出另一个，主要有基于离散对数问题的ECC算法、基于大数分解的RSA算法、杂凑函数（散列算法）等。

密码算法应用常常需要处理较大的信息量，或具有较大的计算强度，往往是各种通信系统中计算密集型环节，影响整个系统的吞吐率。常规的GPP无法满足其速度要求，安全性也不如专用硬件，因此目前国内外对密码处理专用硬件的研究和开发十分活跃。密码算法的硬件加速方式可分为三类：密码算法处理器、特定密码算法ASIC和可重构密码处理结构。

随着移动互联网的飞速发展，对系统安全性的要求也越来越迫切，同时随着互联网大数据通信时代的到来，对安全处理器的性能要求也越来越高，保障系统安全所需投入的处理资源将越来越多，安全应用的范围也会越来越广，密码算法与可重构技术的结合，可以满足性能和安全方面的需求，具体地，可重构密码系统架构在这一应用领域的优势体现在以下几点[14]：

（1）可重构系统可以根据实际需求实现不同的密码算法，具有很大的算法灵活性；

（2）可重构系统的计算能力能够满足密码算法的高性能需求，结构能够根据特定的算法集定制硬件，使算法执行更加高效；

（3）可重构系统具有扩展性，能够适应不断被提出新的更安全的算法，同时支持随时修改密钥，满足某些特殊情况下的白片需求。

* 1. 国内外研究现状
     1. 可重构密码架构综述

近年来，能够同时满足密码算法应用的高性能、高灵活性和低成本需求的可重构密码处理器已成为研究热点。密码算法的实现结果的考核可以以面积效率为标准，由于密码算法的数据依赖性较低，和媒体、通信算法不同，密码算法的实现可以通过增加硬件开销来获得算法实现性能的提升。已有的面向密码的可重构架构在处理单元（Process Element，PE）设计上或多或少存在一些设计上的不足，导致整个架构变得臃肿，存在比较多的计算资源浪费。已有研究为了提高面积效率进行不懈努力，但与专用硬件实现的结果相比仍有显著差距。

COBRA[15]是一款面向对称密钥算法提出的指令级分布式可重构处理器，通过对多种对称密钥算法的映射实现进行验证，AES算法的实现面积效率为0.216Gbps/Mgates。为了保证充分的灵活性和并行计算能力，COBRA架构在每个处理单元中包含了各种算法所需要的所有算子，包括置换单元、S盒等资源开销很大的操作，从而造成了极大的硬件冗余；处理单元中的很多算子被串行组织，最长的路径上串联了13个算子单元，这使处理单元的电路延迟很大，整个处理器的主频非常低，很难实现高性能。

Celator[16]是由艾克斯—马赛大学研发的面向分组密码算法和哈希函数的可重构架构，其计算阵列基于脉动结构设计，采用二维互联结构完成计算单元间的数据传输，每个计算单元支持逻辑操作和算术操作，通过有限状态机控制计算阵列的数据访问和计算操作，有效控制了整体架构的硬件资源开销，DES算法的实现面积效率为0.25Gbps/mm2。但是该架构中处理单元的算子功能很少，支持的算法有限；并且基于指令的设计使得每一个操作都需要重新获取指令进行指令译码再进行数据处理，也无法实现算法的流水设计，算法实现性能不高。

Cyptoraptor[17]是由德州大学奥斯汀分校研发的一款高性能、低功耗、高灵活的密码处理器，面向分组密码、流密码、哈希函数三类百余种对称密钥算法进行架构探索，是目前研究支持算法最多的一款，DES和AES算法的实现面积效率分别为6.75Gbps/mm2和20.25Gbps/mm2。该架构通过灵活的计算单元和阵列互联模块，大幅提升了阵列的流水效率，同时对计算单元支持的算子操作进行组合，有效缩短了关键路径时延和流水级数，从而提升了计算性能。然而，基于该架构映射的对称密钥算法中，比较多的算法实现硬件利用率不高，影响了面积效率，其原因在于未能考虑不同算法间的特征差异，导致硬件架构存在过度设计。

ProDFA[18]是由国防科技大学提出的一种基于可编程数据流计算的体系结构框架，DES和AES算法的实现面积效率分别为6.09Gbps/mm2和5.83Gbps/mm2。通过分析架构可编程性和数据流计算特性与控制逻辑属性的关系在一个处理单元中通过多周期运算，实现完整的算法计算。迭代结构可以节省电路开销，多个处理单元集成可以实现高吞吐率。在处理单元设计时，每一个算子都单独设计，没有考虑与其它算子进行串行组合，这导致完成算法需要很多个处理周期，因此单个处理单元完成算法的性能很低，要想实现高性能需要很多的处理单元集成。

RCPA[19]是由解放军信息工程大学研究提出的一款高性能和高灵活的密码处理器，该架构利用处理单元阵列和阵列间的灵活互连，支持分组加密算法的流水展开映射实现，同时也考虑了算子单元之间的简单组合，缩短映射的流水级数，从而提升了计算性能。与Cyptoraptor架构类似，处理单元中算子单元的过度设计使算法实现时存在很多的闲置单元，硬件利用率不高，影响了面积效率。

综上所述，现有的面向密码算法的可重构架构在设计处理单元时，在算子组合上存在过度组合导致处理单元延迟过大的问题，或者不进行算子组合，存在映射周期过多的问题；处理单元中包含了算法所需的所有算子单元，算法映射时大部分算子单元被闲置，存在硬件利用率不高的问题。算子组合关系到处理器性能，算子利用率关系到芯片面积，因此建立一个统一的算法模型，充分挖掘算法的算子组合特征和算子位置特征，根据提取的特征，提出一套更合理的处理单元设计方案，优化算子组合，提高算子利用率，从而提升整个架构的面积效率。

* + 1. 可重构密码PE概述

可重构密码PE作为可重构架构的核心功能部件，是可重构密码架构设计中的关键环节。架构要完成的所有的运算都会在PE上执行，架构中的其它模块都只是使这些PE能正常高效工作的保障。PE的功能决定了架构的功能，PE的性能决定了架构的性能。当前的几类可重构密码PE方案在PE功能包含上对算法的分析不充分，导致PE在功能设计上过于臃肿，硬件利用率很低，影响了整个架构的面积效率。

可重构密码PE内部包含了完成密码算法所必须的各种功能单元，这些功能单元按照组合的方式不同可以分为串行组合、并行组合、串并混合。在串行组合方式中，各可重构功能单元按照一定的次序串行连接，对不需要执行运算的进行旁路；并行连接方式中，各可重构功能单元并行连接，通过数据选择器选择相应的功能单元参与运算。

早期的可重构密码处理器大多采用串行组合，这种组合方式在设计中要结合密码算法的处理特点，对算法的处理顺序进行分析并找出最佳的连接顺序；这种组合方式每次可以依次执行多种运算，提高了执行效率。但串行组合设计使路径延迟达到了最大，降低了整体的最高运算时钟频率，架构的整体性能很低；同时串联组合设计需要与算法中算子的组合紧密关联，但是没有一种组合策略可以兼顾所有算法，过度的串行组合会使算子利用率变得很低。

并行组合方式中，各个功能单元并行放置，设计简单且易于实现，并且使路径延迟最小，整个处理器可以有很高的主频。这种结构的缺点是每次只能执行一种的运算，完成算法的完整流程需要很多PE参与，因此这种组合方式一般出现在可重构密码阵列架构中，用更多的PE数量来弥补单个PE功能组合上的缺陷，通过增大面积开销来保证很高的性能。同样，并行组合方式并没有减少资源冗余，它用牺牲面积来换取更高的性能。

串并混合则是根据两种连接方式的优缺点，采取的一种折衷连接方式。将相关度低的功能单元并行组合，将相关度高的功能单元串行组合，同时考虑功能单元之间的平衡延迟。在并连时，通过数据选择器选择所需的运算的结果，串连时则对不需要执行运算的功能单元进行旁路。该方案在电路内部既有较小的路径延迟，又具有较高的执行效率。

表1- 1 不同架构的功能单元利用率

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 架构  名称 | PE方案 | PE占整个架构面积比例 | 功能单元利用率 | | | | | | | | |
| 算  法 | 功能单元 | | | | | | | |
| 算术  单元 | 移位  单元 | 置换  单元 | 逻辑  单元 | S盒 | 有限域乘法 | 模乘  单元 | 平均利用率 |
| COBRA | 串行组合 | 40% | AES | 0% | 0% | 0% | 17% | 100% | 50% | 0% | 14% |
| DES | 0% | 0% | 13% | 4% | 25% | 0% | 0% | 3% |
| RC5 | 13% | 8% | 0% | 8% | 0% | 0% | 3% | 7% |
| RCPA | 并行混合 | 90% | AES | 0% | 0% | 0% | 25% | 25% | 50% | 0% | 19% |
| DES | 0% | 0% | 17% | 8% | 8% | 0% | 0% | 6% |
| RC5 | 13% | 13% | 0% | 6% | 0% | 0% | 0% | 5% |
| Cyptoraptor | 串并混合 | 72% | AES | 0% | 0% | 0% | 50% | 14% | / | / | 14% |
| DES | 0% | 0% | 17% | 17% | 2% | / | / | 5% |
| RC5 | 13% | 13% | 0% | 13% | 0% | / | / | 3% |

表1-1列出了三种PE方案的代表架构中PE面积占整个架构的面积比例以及在映射不同算法时各种功能单元的利用率。

从表的前三列可以看出，对于可重构密码架构，PE占据了整个架构的大部分电路面积，特别是对于并行组合和串并混合的这种阵列架构，占比更是达到了90%和72%，因此PE设计成为了追求更高的面积效率的关键问题。表还列出了三种代表架构在实现三种典型密码算法时各种功能单元的利用率。可以发现，三种架构的功能单元利用率很低，AES的平均利用率达到15%左右，其它两个算法的的平均利用率只有5%左右。

在架构设计时，为了兼顾所有的算法，一定的功能单元冗余是必然的；当前的几类可重构密码架构在运行各单一算法时冗余比例达到了85%-95%，也就是说在运行某一个算法时，整个芯片有70%-85%的面积是一直被闲置的，这些冗余严重影响了算法的面积效率。

* 1. 论文研究内容及意义

本文针对分组密码算法的可重构实现提出了一套新的PE设计方案，用于减小功能单元的冗余，提高资源利用率，从而提高整个可重构平台的面积效率。在研究面向分组密码算法的PE方案的过程中，首先选定目标算法集，建立了一个统一的图模型来描述算法，在这个模型上提取分组密码算法算法算子的参数特征、组合特征和位置特征。然后根据算法集中提取的参数特征、组合特征和位置特征分别确定PE中功能单元的功能参数、功能组合和功能分布，确定一个初始的架构平台。接着对 这个初始建构建立图模型，开发一套算法映射工具，完成算法图到架构图的映射分析，并根据映射结果进行设计反馈，优化初始架构。最后对优化后的架构进行RTL实现验证。



图1- 1 研究内容

本文会对算法集合中的几十种分组密码算法进行映射分析，最终实验结果如表1-2所示，功能单元利用率相对于其它架构平台平均提高50%以上，面积效率相对于其它架构平台平均提高30%以上。

表1-2 考核指标

|  |  |
| --- | --- |
| 考核内容 | 指标 |
| 功能单元利用率 | 功能单元利用率相对于其它架构平台平均提高50%以上 |
| 整体性能 | 面积效率相对于其它架构平台平均提高30%以上 |

本文的研究方案如图1-1所示，具体研究内容包括：

（1）算法图建模和算法特征提取

PE方案的确定首先要分析足够多分组加密算法的特征，算法特征分析得越彻底，根据特征确定的架构才能与算法越契合，对应的功能单元利用率也会越高。有别于其它方案中对算法特征的人工提取，本文对分组加密算法建立了一个统一的图模型，便于采用自动化工具对大量算法的特征进行提取；这个图模型同样也会被用到后面的算法映射中。

（2）根据算法特征确定PE初始设计方案

根据算法集中提取的算子参数特征、组合特征和位置特征分别确定PE方案中功能单元的功能参数、功能组合和功能分布。这样就确定了一个初始的PE方案，这个方案并不是最优的，进一步的优化会在算法映射反馈之后。

（3）架构图建模、算法映射分析和反馈优化

本文中的算法映射基于图模型，首先用一个图模型来描述初始架构，算法和架构在相同的模型下采用自动化工具完成映射。自动化映射工具基于子图同构理论[]，采用改进的VF2[]算法完成图匹配，定义一组资源成本约束函数从成百上千的匹配中选取最优映射结果。对算法集的映射结果进行统计分析，找出初始架构中存在的冗余单元，对初始方案进行进一步的优化。

本文工作中主要有以下创新点：一是对架构和算法建立了统一的模型，基于两个模型，本文采用自动化工具对算法进行特征分析和算法映射，这有别于其它论文中的手工操作，自动化工具可以突破手工操作的局限性，排除很多人工干扰，同时可以对更多的目标算法进行分析；二是PE方案不只是考虑到分组密码算法中算子的参数特征和组合特征，本文关注到了不同算子在算法轮中的位置特征，根据位置特征来决定PE中的功能分布，其它架构对算子进行了全功能分布，造成了很大的资源浪费；三是对架构进行了映射分析，并根据映射结果对方案进行反馈设计，实现进一步优化。

论文的研究意义在于：以分组密码算法的高面积效率为目标，提出了一套设计粗粒度可重构PE方案的研究方法，并对方案进行了验证，在一定程度上解决了可重构密码架构平台功能利用率低下的问题，提升整个系统的面积效率，充分发挥粗粒度可重构系统在以密码算法为代表的数据密集计算领域的独特优势。

* 1. 论文组织结构

本文共分为7章，各章的主要内容具体如下：

第一章为绪论，从可重构计算、密码算法角度介绍本文的相关研究背景，介绍了可重构密码架构和可重构密码PE的国内外研究现状以及阐述了论文的研究内容和意义；

第二章介绍了分组密码算法的三种一般结构：Feistel、SP、LM和ARX，并分别介绍了三种结构的代表算法DES、AES、IDEA和SPECK。本章还给出了面向分组密码算法的可重构架构的结构介绍，重点介绍了可重构系统中计算引擎部分的结构，包括计算阵列、通用寄存器堆、输入输出通道等；

第三章建立了分组密码算法的统一图模型，在这个图模型上采用自动化工具提取分组密码算法算子的参数特征、组合特征和位置特征；

第四章根据提取的算子参数特征、组合特征和位置特征分别确定PE方案中功能单元的功能参数、功能组合和功能分布，提出PE初始设计方案；

第五章主要针对配置时间的减少设计了部分重构和流水调度的配置策略，结合分组密码算法的配置特征说明了部分重构和流水调度针对的问题和实现原理，同时给出了方案的工作机制和实现方式，最后通过实现对方案的有效性进行了验证；

第六章对本文的PE设计进行了验证，分析了方案在功能单元利用率和面积效率方面的实验结果，并与其它多种可重构系统在功能单元利用率和面积效率上进行对比分析；

第七章对本文的工作进行了总结，并对未来进一步的工作进行了展望。

算法映射分析与PE方案优化

可重构架构映射概述

应用映射是可重构设计的一个主要难题，在可重构架构上处理数据密集型应用要求较高的吞吐量和并行性，采用手动的方法将应用映射到可重构架构上不仅费时并且容易出错，当算法集合增大时，大量的映射变得非常繁琐。手工映射需要映射人员对架构有清晰的认识，不容易掌握，而且在架构进行微调时，所有的映射要重新进行，出现大量的重复性工作。另一方面，拥有高效的自动化映射工具允许架构设计人员对架构进行大规模的应用样本映射测试，根据映射结果反馈，对架构进行优化调整。因而设计高效的自动化工具是可重构架构设计的一个关键问题。

问题模型

应用可用数据流图（Data Flow Graph，DFG）表示，在DFG中每个节点表示应用中的某一个运算操作，有向边表示两个操作间的数据依赖关系和数据流向。应用到可重构阵列上的映射包括三个部分：DFG中的节点到可重构阵列上的PE的映射；DFG中的有向边到PE间互连的映射；数据到局部存储的映射。图xxx给出了一个应用到可重构阵列上的映射，图中的R表示该PE不进行运算，作为路由使用。



图5- 1

通常将应用映射到可重构阵列上需要考虑体系结构上的几点制约因素：

* 阵列

阵列的规模，即阵列上的资源数量，直接决定应用操作的并行度和吞吐量，较大的应用映射到资源数较少的阵列上时，需要根据资源数量对应用算法进行划分，在不同划分里的运算需要对可重构阵列进行配置切换才能继续执行。另一个关键因素是阵列的拓扑结构，包括阵列的横向执行单元的个数，这决定了应用可以在阵列上映射的数据宽度；阵列的纵向深度，这在应用进行流水展开时决定了可以映射的流水级数。

* 处理单元

每一个都有多个功能单元资源，功能资源包括单操作的功能单元例如移位器和支持多种操作的功能单元如一些功能单元组合操作。在异构的可重构阵列中，一些面积开销大的功能单元如S盒替代单元只存在阵列中的部分处理单元中，因而不是所有的操作都能映射到任意的处理单元上，部分操作只能由特定的处理单元完成。

* 局部存储器

通常上多个通过总线来共享一个局部存储器，总线的带宽有限，不能在一个周期来完成所有PE的数据获取，因此在映射时要考虑这种总线带宽限制。

* 互连网络

可重构阵列中PE间的互连资源是有限的，不是所有的PE间都可以直接互连；在某些时候还需要将某一个PE作为路由使用。互连网络结构是应用映射时通信路径选择的主要约束因素。

本文提出的可重构密码处理器上应用映射的特点

1. 可重构密码处理器阵列的互连是有导向的，数据从阵列的第一行输入，第一行的输出结果给第二行，数据经过整个阵列从最后一行输出。因此在映射时从阵列的第一行出发，数据流向和阵列的互连导向一致。
2. 可重构密码处理器阵列上的映射属于空间映射，密码算法会被循环展开，在架构上进行流水展开映射。
3. 行间的互连不是硬连线，而是数据选择。可以选择某条路径到达下一行的任何位置，但不可以同时到达所有的位置。也就是从某一个点出发的所有连线是互斥的，不可约同时存在，每一次只能互连到一个位置。
4. PE内部功能更复杂，不只是单个运算到单个PE的映；PE内部有很多组合功能设计，应用中的多个组合的功能会被同时映射到一个PE中。
5. 单个PE的输入不唯一、功能不唯一，也就是说在输入允许的情况下，横向上并行的多个功能可以被映射到同一个PE。
6. 可重构密码处理器阵列被设计成行间异构、列间异构。这样的不一致性结构在架构描述、映射算法上都有特异性。

研究现状

将循环映射到时，根据映射目标架构的不同，有两种常用的映射方法：时间映射（temporal mapping）和空间映射（spatial mapping）。

时间映射适用于处理单元只有一个或者一行的可重构处理器的架构。时间映射的方法将整个循环体映射到一个PE或一行PE上，每PE个或每行执行不同迭代次数的循环。PE要顺序执行多个操作，每执行完循环的一次操作就要进行一次重构，动态配置下一级的操作功能。时间映射方法的特点有：

* 由于循环的所有操作都在一个PE或一行内完成，不需要考虑PE间的数据互连；
* 可以直接利用传统的编程编译方法完成循环的映射；

这类CGRA映射问题[ V1-V3]的方案来自于VLIW架构的编译技术，它利用了VLIW中的时间流水模调度算法和VLIW中的存储共享特征。

空间映射的方法适用于可重构阵列架构，它将循环中的每个操作一次性映射到CGRA的PE阵列上，每个PE只绑定循环中的一个操作，只执行一种固定的操作，整个循环执行的过程中只用配置一次，不需要在经过重新的配置执行其他操作。空间映射的方法有以下特点：

* 能够充分利用CGRA计算资源和并行运算的能力，并行执行循环体的多次循环；
* PE只需要完成一个固定的操作，重构的开销小，整个循环的过程不需要进行重构；
* 操作节点的布局要充分考虑各个操作间的数据依赖关系和CGRA的互连资源，布局复杂，并且要使用较多的互连资源；

这类CGRA映射问题的研究更为广泛，最开始的方案灵感来自FPGA[v4-v5]综合里的布局布线技术，但是CGRA本身和FPGA有很大不同的，具体来说，CGRA中的互连线大部分是固定的，而FPGA中是可配置的；因此基于FPGA综合里的布局布线技术在固定的互连在寻找路由变得很困难。后来的一些映射方法主要来自于图论领域，[V6 ]利用子图同构算法获取DFG图到架构图之间的映射候选集；SPKM [V7] 引入分裂和外扩的方式 [V8] ，它将应用看成一个集合，每一次向外扩展一个点，余下的点作为新的扩展集合，一直到集合中的点全部外扩完毕。类似的图论方案还有如[V9]中的子图同胚[V10(134)]以及EPIMap[V10(56)]中的图满射技术。

本文中的可重构密码处理器是架构中包含PE阵列，因为映射时将密码算法循环展开以流水的形式映射到阵列上，因此本文中的应用映射问题为空间映射。根据上小节总结的架构特征，本文选用适用性更好的子图同构算法作为映射的依据，子图同构算法能够在异构的不规整架构中找出所有的可能映射方案，然后再通过一定的成本规则找出资源使用最少的方案作为最终映射方案。

基于子图同构的映射方案

子图同构基本概念

* 基本定义

**定义5.1（图）：**一个图是一个四元组，其中，

1. 称为G的顶点集，其元素称为顶点或结点；
2. 称为G的边集，其元素称为边；
3. 为图G的顶点标记函数，说明顶点与其标记的对应关系；
4. 为图G的边标记函数，说明边与其标记的对应关系。

**定义5.2（子图）：**一个图 为图的一个子图，记为，如果有：

1. ；
2. )；
3. ；
4. 。

此时，我们也称图为图的一个超图。

**定义5.3（图同构）：**图和图是同构的，记为，如果存在一个双射函数，使得：

* 1. ；
  2. 。

这样的函数也称为图与的图同构。例如，图xxx中A和B两个图同构。



图5- 2图同构的示例

**定义5.4（子图同构）：**给定图与的一个图同构以及另一个图，如果，则为图与的一个子图同构。图xxx是一个子图同构的示例，S是G的一个子图，它包含了G中顶点集的一个子集，而且S中顶点的连接方式与G中相同，所以称S是G的一个同构的子图。



图5- 3子图同构的示例

VF2子图同构算法

图的同构判定是图论学科的基本同题之一，文献[x-x]对这个问题进行了充分的研究。文献[x]提出了Ullmann算法，它是一个可前向剪枝的带有回溯的树搜索过程；文献[x]提出的VF2算法同样基于搜索方法，它利用一个快速计算的启发式规则进行剪枝，这使其性能得到显著提升。

表5- 1 Ullmann算法和VF2算法对比



VF2算法的核心思想是搜索加剪枝，重点就在于如何剪枝。状态s存储搜索过程中的部分匹配，以及算法需要的其它数据。代表中间状态，和表示当前状态s的部分匹配中图和图中的点，整个算法流程如图xxx所示。

|  |
| --- |
| 1) PROCEDURE Match(s)  2) 输入：中间状态，初始状态满足  3) 输出：两个图之间的所有匹配  4) IF M(s) 覆盖了的所有结点  5) 输出M(s)  6) ELSE  7) 计算可以被包含到M(s)中的候选点对集合P(s)  8) FOREACH P(s)中的点对p  9) IF 候选点对满足加入M(s)的所有约束规则  10) 将p加入M(s)，计算  11) CALL Match()  12) END IF  13) END FOREACH  14) 恢复数据状态  15) END IF  16) END PROCEDURE Match |

图5- 4 VF2子图同构算法流程

初始化时状态是，是空集，即还没有任何匹配；之后递归的进行搜索。如果当前状态s代表的部分匹配包含了中的所有节点，则已经找到了在中同构的子图，搜索结束；否则，在当前的局部匹配基础上，再匹配一个点。找出所以可能进行匹配点对集合；对于每一个匹配对p，检查加入匹配p是否可行，即加入p后，两个图还是否同构，以及加入p之后，是否还有扩展的可能性；如果加入匹配p可行，则将p加入s，递归调用Match()，继续搜索。如果若干次调用Match()后都没有找到同构的子图，则说明当前状态不可能扩展出可行的子图同构匹配；所以将生成新状态时加入的两点匹配p从s中删除，回溯到上一个状态。

在上述算法流程中，新加入的匹配p，我们要检验其加入的可行性，从而对搜索空间进行剪枝，来提高算法的效率。

先约定几个符号：和表示图和图中的点集。n和m分别表示图和图中的点。表示点n在图G中的前驱，表示点n在图G中的后继。和表示状态s在图和图中，指向当前已经匹配的点集的所有边的源点集合。和表示状态s在图和图中，从当前已经匹配的点集出发的所有边的终点集合。，即当前状态s在图中已经匹配的点集的所有一步邻居。，即图中，除了s中已经匹配的点，和这些点的一步邻居以外的点。

新加入的点p需要同时满足五个规则：-











和保证加入新的匹配对p后，两个子图仍然是同构的。设新加入的匹配对是(n,m)，则对于n在图中的所有前驱（或后继），必须能在图中m的前驱（或后继）里有相应的点与之对应。同样，对于m在图中的所有前驱（或后继），也必须能在图中n的前驱（或后继）里有相应的点与之对应。

、和都是剪枝策略。其中表示求集合中元素的个数。和表示n在（或）中的前驱（或后继）的数目，必须大于等于m在（或）中的前驱（或后继）的数目。如果不满足，则说明对于图中新匹配的点m，其邻居个数是大于图中n的邻居个数的，最终必然无法完全匹配图中所有的点。

跟和思想类似，只不过考虑的两步邻居。具体是，和中的考虑的邻居是和中的邻居。这些点既跟n相邻，又跟当前匹配中的其他点相邻。而考虑的邻居，是只跟n相邻，跟当前匹配中其他店不相邻的邻居。这样细粒度的考虑的好处是可以更细粒度的剪枝，从而提高剪枝效率。

算法的搜索空间主要由候选点对集合P(s)决定，在没一个状态s下都对应不同的搜索空间。候选点对的规则为：

1. 如果和都不为空，则取这两个集合中的所有点两两组合，生成候选匹配对集合；
2. 如果和两个集合都为空，若和都不为空，则取这两个集合中的所有点两两组合，生成候选匹配对集合；
3. 如果上面四个集合都为空（对于非连通图会出现这种情况）。则只能找两个图中所有没有匹配的点两两组合，生成候选匹配对集合。

细粒度的分类讨论，可以尽量减少单次生成的候选匹配对的数量。否则如果每次都按上面第三种方式生成，则每次递归都会生成很多之前生成过的匹配对，造成重复计算。

Conte, Donatello, et al. "Thirty years of graph matching in pattern recognition."*International journal of pattern recognition and artificial intelligence* 18.03 (2004): 265-298.

Ullmann, Julian R. "An algorithm for subgraph isomorphism." *Journal of the ACM (JACM)* 23.1 (1976): 31-42.

Cordella, Luigi P., et al. "A (sub) graph isomorphism algorithm for matching large graphs." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.10 (2004): 1367-1372.

架构图建模

算法映射基于图模型，因此需要对架构建立一个统一的图模型。第三章中采用有向图对算法进行建模，利用图的节点描述算法中算子，图中的边描述数据依赖。完成算法的图建模后就可以利用图论算法来提取算法的某些关键特征为架构设计提供依据。在可重构架构中，利用节点描述PE，边描述架构中的互连。这样算法和架构有了相同的基于图的描述方式，利用图论中的匹配算法可以完成算法到图的匹配映射[1][2]。

**定义5.5（架构图）：**一个架构图是一个二元组ARG=(V,E)，其中V是节点集，表示架构中的PE；E是边集，表示架构中的数据互连。

和算法图相比，架构图中的节点具有更复杂的功能属性和位置属性，架构图中的边网络也更加庞大。

初始架构中定义了5类PE，因此架构图中的节点有五类不同的功能属性，在架构图中用字符串进行描述。如表4.xx所示，PE\_type1的功能呢属性的字符串描述了第一类PE的19中功能，在进行算法图节点和架构图节点进行匹配时，算法图节点的功能属性必须是架构图节点功能属性的子集，也就是架构图节点对应的PE能够提供算法节点所需的功能，在属性比较时，算法图节点属性字符串是架构图节点属性字符串的子串。架构图中的（x，y）位置属性标识了该节点对应的PE在架构中的位置。在完成算法映射匹配时根据这些位置信息从多个匹配中选择出占用最少行、列资源的结果。

表5- 3架构图结点属性

|  |  |  |  |
| --- | --- | --- | --- |
| PE（节点） | x | y | 功能属性 |
| PE\_type1 | 1 | 1,2 | au\_xorau\_auxor\_xorauxor\_xorxorau\_xorxorauxor\_sh\_xorsh\_shxor\_xorshxor\_xor\_xorxor\_xorxorxor\_or\_and\_bn\_xorbn\_bnxor\_xorbnxor\_ |
| PE\_type2 | 1 | 3,4 | au\_xorau\_auxor\_xorauxor\_xorxorau\_xorxorauxor\_sh\_xorsh\_shxor\_xorshxor\_xor\_xorxor\_xorxorxor\_or\_and\_ |
| PE\_type3 | 2 | 1,2,3,4 | au\_xorau\_auxor\_xorauxor\_xorxorau\_xorxorauxor\_sh\_xorsh\_shxor\_xorshxor\_xor\_xorxor\_xorxorxor\_or\_and\_lut\_xorlut\_lutxor\_xorlutxor\_ |
| PE\_type4 | 3 | 1,2 | au\_xorau\_auxor\_xorauxor\_xorxorau\_xorxorauxor\_sh\_xorsh\_shxor\_xorshxor\_xor\_xorxor\_xorxorxor\_or\_and\_bn\_xorbn\_bnxor\_xorbnxor\_gfm\_xorgfm\_gfmxor\_xorgfmxor\_ |
| PE\_type5 | 4 | 3,4 | au\_xorau\_auxor\_xorauxor\_xorxorau\_xorxorauxor\_sh\_xorsh\_shxor\_xorshxor\_xor\_xorxor\_xorxorxor\_or\_and\_gfm\_xorgfm\_gfmxor\_xorgfmxor\_ |

如图xxx所示为一个异构组的图模型，每一行的4个PE被抽象成四个点，每个点有与PE对应的功能属性，行与行之间的互连被抽象成从上一行到下一行的有向边，架构中的行间互连支持数据从上一行的PE到下一行的任意一个PE，因此在图模型中上一行的每一个点都有一条有向边到下一行的任意一个点。



图5- 6异构组的图模型

[1]沈来信, 曾国荪, and 王伟. "基于分层超图的可重构体系结构模型." *计算机科学*40.4 (2013): 26-30.

[2]沈来信, and 王伟. "基于算粒感知的可重构体系结构." *Computer Engineering*39.9 (2013).

基于VF2算法的映射方案

算法映射问题与子图同构问题有两点不同：一是子图同构是指在模板图中存在一个和查询图同构的子图，算法映射的目标是从架构图中找出一个图，这个图包含算法图，同构是一个双射关系，包含是一个单射关系，因此需要在相应的匹配算法上放宽约束；二是子图同构是找出同构子图，它没有更多的物理意义，算法映射是要在架构中找到使用资源最少的一种匹配，因此，它需要从多个匹配中找到最优的解。

VF2算法适应性调整

子图同构是指在模板图中存在一个和查询图同构的子图，根据公式xxx可知同构是一个双射关系。在算法图和架构图匹配中，目标是从架构图中找出一个图，这个图包含算法图，但不要求同构，包含是一个从算法图到架构图的单射关系。因此在约束上，映射匹配的约束比子图同构弱。

图xxx描述了架构图和算法图的包含关系，映射{(v1+v2, a1), (v3, a5), (v4, a6), (v5, a7), (v6+v7+v8, a9), (v9, a10)}中架构图的子图{a1, a5 ,a6, a7, a9, a10}和对应的算法图不同构，子图中的多条边在算法图中没有对应的边存在，这只满足单射关系。

VF2算法的和规则对图同构的双射关系进行了约束，因此对于映射算法必须对这两个放宽这两个规则，新的规则如下：





在新的规则中，只要求图（架构图）中包含图（算法图）对应的前驱和后继，反向则不再约束。



图5- 7架构图和算法图的包含关系

单纯的VF2算法不考虑点的属性问题，规则和只保证了对应点的前驱和后继的对应关系，它不保证架构图中的点能提供算法图对应的点所包含的操作属性。在进行算法图和架构图建模时，用字符串来描述图中节点的功能属性，因此架构图节点功能属性字符串要包含算法图的操作属性字符串，增加一个属性约束规则如式xxx所示，其中attr(x)表示节点x的功能属性。



最优匹配约束规则

架构图对算法图满足包含关系的子图可能成千上万，这些子图都满足基本的映射要求，但并不都是我们所需要的。映射要考虑行资源、列资源、互联资源的成本，还要考虑映射聚集等因素。在本文提出的映射方案中主要考虑了5类约束，满足这5类约束的最终结果满足使用最少的行、列资源，最简单的数据互连，以及尽可能往架构的右上聚集。

对算法图中的节点加入拓扑信息a，v.a表示算法图中的点v出现在横向上的位置；对架构图中的节点加入拓扑信息（x, y），其中v.x表示架构图中点v代表的PE出现在纵向上的位置，v.y表示架构图中点v代表的PE出现在横向上的位置。S表示匹配算法找出的所有可能的匹配集合， 为S中的一个元素，表示一个有效的匹配；(,)表示一个有效匹配中的一个匹配点对。











[1]Clark, Nathan, et al. "Scalable subgraph mapping for acyclic computation accelerators." Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems. ACM, 2006.

[2]Hamzeh, Mahdi, Aviral Shrivastava, and Sarma Vrudhula. "EPIMap: using epimorphism to map applications on CGRAs." Proceedings of the 49th Annual Design Automation Conference. ACM, 2012.

[3]Tuhin, Mohammed Ashraful Alam, and Theodore S. Norvell. "Compiling parallel applications to coarse-grained reconfigurable architectures." Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on. IEEE, 2008.

[4]Brenner, Janina A., Sándor P. Fekete, and Jan C. Van Der Veen. "A minimization version of a directed subgraph homeomorphism problem."Mathematical Methods of Operations Research 69.2 (2009): 281-296.

映射结果分析与PE方案优化

算法映射结果分析

统计架构中每一个功能单元在映射完算法集合的算法后的使用情况，分析架构中各个功能单元的使用频率，和对应的映射算法；根据使用频率对功能单元进行优化。

基于映射结果的PE方案优化

没有被使用的功能：直接删除

使用频率低的功能：删除，对使用的算法进行重新映射，如果算法能够不同，且不增加硬件开销（通过走其它路线），那么将功能删除，否则保留。