粗粒度可重构密码逻辑阵列智能映射算法研究

杜怡然[1],杨 萱[2],戴紫彬[1],南龙梅[1],李 伟[1]
(1. 解放军信息工程大学,河南郑州 450000;2. 江南计算技术研究所,江苏无锡 214083)

摘 要: 针对粗粒度可重构密码逻辑阵列密码算法映射周期长且性能不高的问题,该文通过构建粗粒度可重构密码逻辑阵列参数化模型,以密码算法映射时间及实现性能为目标,结合本文构建的粗粒度可重构密码逻辑阵列结构特征,提出了一种算法数据流图划分算法. 通过将密码算法数据流图中节点聚集成簇并以簇为最小映射粒度进行映射,降低算法映射复杂度;该文借鉴机器学习过程,构建了具备学习能力的智慧蚁群模型,提出了智慧蚁群优化算法,通过对训练样本的映射学习,持续优化初始化信息素浓度矩阵,提升算法映射收敛速度,以已知算法映射指导未知算法映射,实现密码算法映射的智能化. 实验结果表明,本文提出的映射方法能够平均降低编译时间 37.9% 并实现密码算法映射性能最大,同时,以算法数据流图作为映射输入,自动化的生成密码算法映射流,提升了密码算法映射的直观性与便捷性.

关键词: 粗粒度; 密码; 阵列; 智能映射; 机器学习
中图分类号: TP301.6 **文献标识码:** A **文章编号:** 0372-2112 (2020)01-0101-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.01.012

Research on Coarse-Grained Reconfigurable Cryptographic Logic Array Intelligent Mapping Algorithm

DU Yi-ran[1], YANG Xuan[2], DAI Zi-bin[1], NAN Long-mei[1], LI Wei[1]
(1. *Zhengzhou Institute of Information Science and Technology, Zhengzhou, Henan* 450000, *China*;
2. *Jiangnan Computing Technology Institute, Wuxi, Jiangsu* 214083, *China*)

Abstract: This paper presents algorithms for cryptographic algorithms mapping based on the coarse-grained reconfigurable cryptographic logic array. Due to a long mapping period and low performance for current algorithms mapping, we propose two methods to improve it. First, combine with the structural characteristics of the coarse-grained reconfigurable cryptographic logic array and cryptographic algorithms, an algorithm for data flow graph partitioning is proposed. By integrating the nodes into clusters to reduce the mapping complexity. Second, refer to the idea of machine learning, a smart ant colony optimization algorithm is proposed. By learning the training samples, the pheromone concentration matrix is continuously optimized and realizes the intelligentization of cryptographic algorithm mapping. The experimental results show that the proposed mapping method can reduce the compilation time by 37.9% and achieve the best performance. At the same time, the algorithm data flow graph is used as the mapping input, and the cryptographic algorithm map stream is automatically generated, which improves the cryptographic algorithm mapping more intuitive and convenient.

Key words: coarse-grained; cryptographic; array; intelligent mapping; machine learning

1 引言

粗粒度可重构密码逻辑阵列是一种基于数据流的高效密码运算结构,它结合了专用集成电路高性能与通用处理器高灵活性的特点,能有效实现对密码算法的加速. 丰富的计算与互连资源是粗粒度可重构密码逻辑阵列取得算法高速实现性能的基础,但同时也造成了算法映射复杂度的提升. 现阶段,基于粗粒度可重构密码逻辑阵列的密码算法映射常采用手工映射进行辅助,算法映射效率不高,映射过程复杂. 因此,在确保密码算法实现性能最大的前提下,如何实现基于粗粒度可重构密码逻辑阵列的自动化高效算法映射,是亟

收稿日期:2018-10-31;修回日期:2019-08-19;责任编辑:李勇锋
基金项目:国家自然科学基金(No.61404175)

待解决的重要问题.

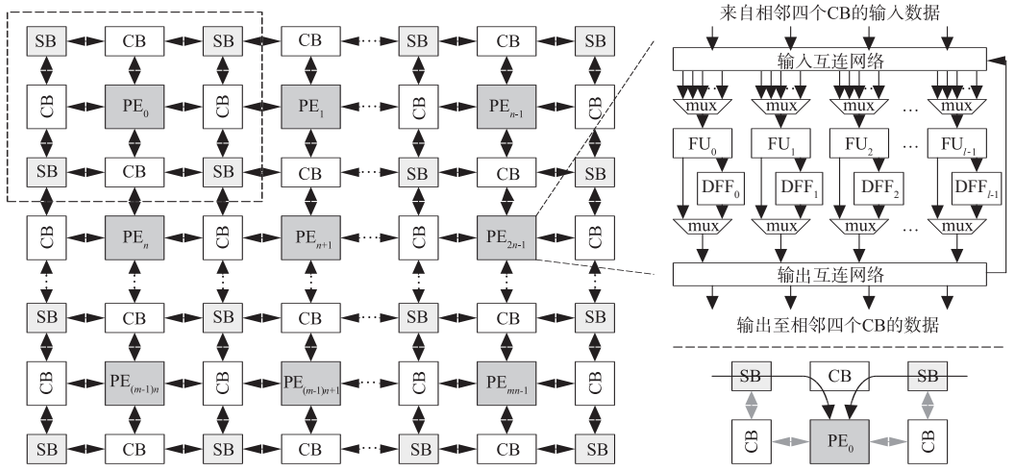
针对基于粗粒度可重构阵列的算法映射主要围绕以下方式展开研究. 文献[1]提出了一种全局高效的启发式算法 EPIMap,有效缩短了映射搜索空间;文献[2]提出了一种 SPKM 方法,提升映射算法在实际芯片中映射的可用性;文献[3]基于多面体模型并针对算法中的循环结构,提出了一种循环转化及映射算法 PolyMAP,提升算法收敛速度;文献[4]借鉴深度神经网络思想,构建了面向算法数据流图映射的神经网络学习结构 DFGNet,通过大量样本学习,提升对实际算法映射的效率;文献[5]提出了一种 2 步蚁群算法,将布局与布线阶段分离,提升了算法成功率,降低了算法映射时间. 上述文献主要针对多媒体算法及相应粗粒度可重构阵列结构展开研究,由于密码算法结构的特殊性,在基于粗粒度可重构密码逻辑阵列的密码算法映射时,上述方法均不能达到较好的映射效果.

本文针对基于粗粒度可重构密码逻辑阵列的高效算法数据流图映射问题进行研究,深入挖掘了粗粒度可重构密码逻辑阵列结构与密码算法数据流图之间存在的内在联系. 结合粗粒度可重构密码逻辑阵列硬件结构特点,提出了基于压缩映射的数据流图划分算法,以划分簇为算法映射的最小粒度,有效降低了算法映射的复杂度. 结合机器学习及启发式算法思想,本文提出了一种智慧蚁群算法,通过对训练样本映射过程的学习,提升目标算法的收敛速度,在实现密码算法高性能映射同时使算法映射更加便捷、高效.

2 可重构密码逻辑阵列参数化模型

课题组前期针对粗粒度可重构密码逻辑阵列展开研究,基于 55nm 工艺设计了一款粗粒度可重构密码逻辑阵列并完成流片验证. 基于前期研究成果,本文构建了规模为 $m \times n$ 的粗粒度可重构密码逻辑阵列参数化模型,其结构如图 1(a)所示,共包含 m 行 n 列个同构的运算元素(Processing Element,PE). 每个 PE 均与上下左右四个方向上的连接器(Connect Box,CB)直接相连,CB 又通过开关盒(Switch Box,SB)相互连接,从而实现了 PE 间数据的互连互通. 由于 CB、SB 及 PE 间均采用双向 32 比特数据线连接,无法实现如图 1(b)所示的同向数据并行.

PE 内部由输入互连网络、输出互连网络以及 l 个密码运算功能单元(Functional Unit,FU)组成. 其中 FU 的输出结果可选择是否进行一级寄存并输出至输出互连网络,输出互连网络能够将运算结果输出至相邻的四个 CB 或反馈至当前 PE 的输入互连网络,输入互连网络接收来自相邻四个 CB 的输入数据及当前 PE 输出互连网络的反馈.



(a)粗粒度可重构阵列参数化模型结构 (b)一种错误数据映射流

图1

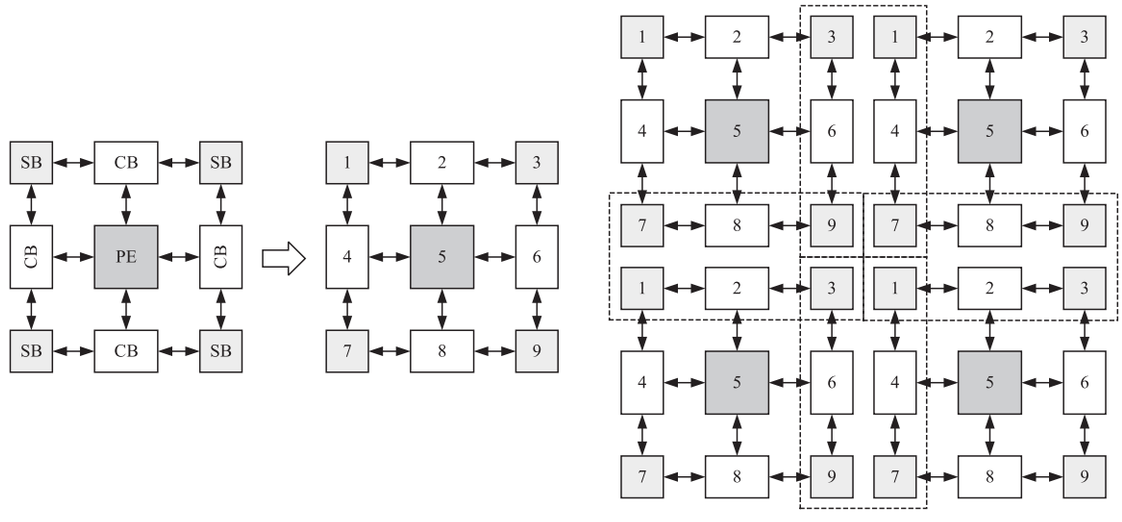
基于粗粒度可重构密码逻辑阵列的密码算法映射是将密码算法转换为粗粒度可重构密码逻辑阵列配置流的过程,考虑到密码算法表示方法的多样性以及实现的可行性,本文采用算法数据流图作为密码算法映射的输入,因此,将算法数据流图中节点进行如下表示:

$$\text{Node} \wedge Tag(x) \wedge Func(function) \wedge InputList(nodelist) \wedge OutputList(nodelist)$$

其中 $Tag(x)$ 表示算法数据流图中节点的标号,$Func(function)$ 表示该节点所执行的功能为 $function$,$InputList(nodelist)$ 及 $OutputList(nodelist)$ 表示与该节点相连的输入节点列表以及输出节点列表,通过对算法数据流图中的节点进行表示即可确定算法数据流图的细节特征.

为了使粗粒度可重构密码逻辑阵列与密码算法数据流图的描述方法更加统一,仿照上述算法数据流图节点的描述方法,对粗粒度可重构密码逻辑阵列参数化模型进行表示.

由于粗粒度可重构密码逻辑阵列中所包含的模块数目较为庞大,为了精确描述粗粒度可重构密码逻辑阵列中各模块的位置,本文定义了集群(Cluster)的概念. 一个集群包含了一个 PE 以及与该 PE 直接相连的四个 CB,同时还包含了与 CB 连接的四个 SB. 集群中各模块的相对位置如图 2(*a*)所示,因此各模块在粗粒度可重构密码逻辑阵列中的位置可表示为(*Cluster*(*x*),*y*),其中 *x* 等于集群中 PE 的下标,*y* 等于集群中的相对位置.



(*a*)集群及其内部各模块相对位置标识　　(*b*)相邻集群中的模块重叠

图2

由于相邻集群间存在如图 2(*b*)所示的模块重叠的情况,因此粗粒度可重构密码逻辑阵列中的同一模块可能存在多种表述方法. 为了更好的对算法映射过程进行描述,本文将粗粒度可重构密码逻辑阵列进行参数化表示,对算法映射状态进行如下定义.

① PE 的初始化状态

¬ PE ∧ *At*(*Cluster*(*x*),*time*(0)) ∧ *Func*(*empty*)

其中¬ PE 表示在时刻 0 位于 *Cluster*(*x*)的 PE 未被映射占用,同时功能链表 *Func*( )为空.

②CB 的初始化状态

¬ CB ∧ *At*((*Cluster*(*x*),*y*),*time*(0)) ∧ *Dire*(*empty*)

其中¬ CB 表示在时刻 0 位于(*Cluster*(*x*),*y*)的 CB 未被映射占用,同时数据流向链表 *Dire*( )为空.

③ SB 的初始化状态

¬ SB ∧ *At*((*Cluster*(*x*),*y*),*time*(0)) ∧ *Dire*(*empty*)

其中¬ SB 表示在时刻 0 位于(*Cluster*(*x*),*y*)的 SB 未被映射占用,同时数据流向链表 *Dire*( )为空.

④ PE 被映射状态

PE ∧ *At*(*Cluster*(*x*),*time*(*t*)) ∧ *Func*((FU$_0$, ¬ DFF$_0$,*IN*(*a*)),(FU$_1$,DFF$_1$,*IN*(*b*)),…,(¬ FU$_{l-1}$, ¬ DFF$_{l-1}$,*IN*(*empty*)))

其中 PE 表示在时刻 *t* 位于 *Cluster*(*x*)的 PE 被映射占用,同时功能链表 *Func*( )表示了 PE 中 FU 的映射情况,即 FU$_j$表示 FU$_j$被映射占用,¬ FU$_j$表示 FU$_j$未被映射占用,DFF$_j$表示 FU$_j$的运算结果经过一级寄存后输出,¬ DFF$_j$表示 FU$_j$的运算结果不经寄存直接输出,*IN*( )表示 PE 中 FU$_j$的输入来源,若¬ FU$_j$则 *IN*( )为空.

⑤ CB 被映射状态

CB ∧ *At*((*Cluster*(*x*),*y*),*time*(*t*)) ∧ *Dire*((*W*,*S*,*IN*(*a*)),(*N*,*E*,*IN*(*b*)))

其中 CB 表示在时刻 *t* 位于(*Cluster*(*x*),*y*)的 CB 被映射占用,且 CB 被映射为由 West 至 South 和由 North 至 East 的两条通路.

⑥ SB 被映射状态

SB ∧ *At*((*Cluster*(*x*),*y*),*time*(*t*)) ∧ *Dire*((*W*,*S*),(*N*,*E*))

其中 SB 表示在时刻 *t* 位于(*Cluster*(*x*),*y*)的 SB 被映射占用,且 SB 被映射为由 West 至 South 和由 North 至 East 的两条通路.

综上所述,本节实现了对密码算法数据流图及粗粒度可重构密码逻辑阵列的参数化描述,解决了映射算法的输入及输出表示问题,为基于粗粒度可重构密码逻辑阵列的密码算法映射奠定了基础.

3　密码算法数据流图划分及智能映射

受粗粒度可重构密码逻辑阵列计算及互连资源的

限制,密码算法数据流图的映射要遵循一定规则. 考虑到密码算法数据流图中算子的数目众多,连接关系复杂,因此本节首先对密码算法数据流图进行划分. 以初始化信息素矩阵为启发因子,通过对训练样本映射过程的学习,优化启发因子,使密码算法快速收敛于最优映射方案,以“已知”映射指导“未知”映射.

3.1 基于压缩映射的数据流图划分算法

密码算法数据流图划分是实现密码算法映射的关键,能够有效降低密码算法映射的复杂度,提升密码算法映射速度,是密码算法映射的预处理过程.

密码算法数据流图与粗粒度可重构密码逻辑阵列结构之间存在密切的对应关系,其中密码算法数据流图中的节点对应粗粒度可重构密码逻辑阵列的 FU,节点间的连线对应粗粒度可重构密码逻辑阵列的互连网络. 受粗粒度可重构密码逻辑阵列计算及连线资源的限制,算法数据流图的划分应遵从一定的规则.

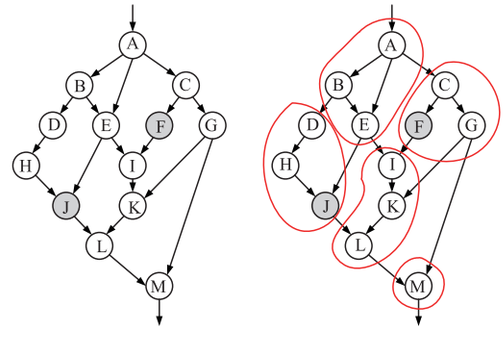
由于粗粒度可重构密码逻辑阵列 PE 内部采用 Crossbar 全互连网络结构设计,因此在映射时,无需考虑 PE 内部 FU 的互连关系,即将密码算法数据流图中的节点依照运算类型互斥的原则划分为簇,并定义划分后的簇为密码算法数据流图映射的最小粒度,与图 2 中粗粒度可重构密码逻辑阵列中的集群相对应. 对于划分好的运算簇,定义其入度及出度分别为簇的输入及输出数目,对应于 PE 的输入及输出端口数目,受粗粒度可重构密码逻辑阵列互连资源的限制,PE 的输入及输出均通过 CB 实现交互,因此,划分簇的入度及出度应小于等于与 PE 相连接的 CB 数目. 若存在簇 A 的输入来自于簇 B 的输出,且簇 B 的输入来自于簇 A 的输出,则定义簇间存在环路. 簇间数据环路将影响数据流图的数据流向,造成数据引用失序等错误. 因此,在进行密码算法数据流图的划分时应避免簇间数据流图的产生. 定义密码算法数据流图中节点的跨度为该节点到包含系统输入节点间有向跨越节点数目的最大值,节点间跨度连续确保了划分后算法数据流图的紧凑性,降低了因簇间过长连线导致映射失败的风险.

基于上述密码算法数据流图的划分规则,本文提出了一种基于压缩映射的数据流图划分算法,算法描述如下.

算法 1 基于压缩映射的数据流图划分算法

```
输入:节点列表 node_list、待映射簇列表 cluster_list
输出:划分后的数据流图
for i in node_list:
ArrangeDFG(i,node_list) #按连接关系构成有向数据流图
for i in node_list:
for j in cluster_list:#划分的簇节点应具有连接关系且节点间跨度连续
    if (LinkTest(i,j) == "Legal") and (SpaceTest(i,j) == "Legal"):
        return_value = MapCluster(i,j) #测试节点映射的 FU 类型
        if return_value == "MapToFU0":#映射至 FU0
        j. FU0_occupy_flag = True #置位簇 j 中 FU0 被映射标志位
            NodeMerge(i,j) #将节点 i 划分至簇 j
        进行簇的出入度及数据环路检测
        if (j. each_port_num < =4) and (CrossTest(j) == "Legal"):
            break
        else:
            j. FU0_occupy_flag = False #清除簇 j 中 FU0 被映射标志位
            NodeSeparation(i,j) #从簇 j 中删除节点 i
    if return_value == "MapToFU1":#映射至 FU1
    if return_value == "MapToFU2":#映射至 FU2
    if return_value == "MapToFU3":#映射至 FU3
    if return_value == "MapToNextCluster":
    pass #映射至下一个簇
else:
    pass
        else:
            pass
```

依据算法 1,对图 3(*a*)所示密码算法数据流图进行划分,其中节点 A ~ M 分别对应不同的运算类型,假定 FU_0 能够映射节点 A、D、G、I,FU_1 能够映射节点 B、C、H、L,FU_2 能够映射节点 E、K、M,FU_3 能够映射节点 F、J.



(*a*)密码算法数据流图 (*b*)划分簇后的密码算法数据流图

图3

图 3(*a*)的划分结果如图 3(*b*),将算法数据流图中的 13 个节点划分为了 5 个簇,其中簇 1 包含节点 A、B、E,簇 2 包含节点 C、F、G,簇 3 包含节点 D、H、J,簇 4 包含节点 I、K、L,簇 5 包含节点 M. 将待映射目标数目由 13 降为了 5,有效降低了密码算法映射的复杂度,提升了密码算法映射效率.

3.2 面向划分簇的智能映射算法

面向划分簇的密码算法映射是基于已划分完毕的密码算法数据流图进行映射,由于密码算法数据流图中的一个簇对应于粗粒度可重构密码逻辑阵列结构上的一个 PE,且无论簇内各节点的连接关系如何,均能够通过 PE 内部的全互连网络实现映射,因此在密码算法

映射时,本文将映射的最小粒度定义为数据流图划分后的簇,从而有效降低基于粗粒度可重构密码逻辑阵列映射的复杂度.

受资源及后端布局布线难度限制,粗粒度可重构密码逻辑阵列的 PE 间无法采用全互连结构进行连接,因此,相较于粗粒度可重构密码逻辑阵列中丰富的计算资源,其互连资源相对匮乏.

对于任意密码算法,假设算法中包含 n 个算子,分别对应于粗粒度可重构密码逻辑阵列中的 $FU_0 \sim FU_{n-1}$,且各 FU 的关键路径延迟分别为 $tcp_0 \sim tcp_{n-1}$;各相邻 FU 通过 x 个 CB 与 y 个 SB 连接($x,y \geqslant 0$),且 CB 的关键路径延迟为 tcp_{CB},SB 的关键路径延迟为 tcp_{SB}. 因此,密码算法映射时各步的关键路径延迟可以表示为:

$$tcp_0, x_1 tcp_{CB} + y_1 tcp_{SB}, tcp_1, \cdots, x_{n-1} tcp_{CB} + y_{n-1} tcp_{SB}, tcp_{n-1} \tag{1}$$

将 $x_1 tcp_{CB} + y_1 tcp_{SB} + tcp_1$ 表示为 tcp'_1,$x_2 tcp_{CB} + y_2 tcp_{SB} + tcp_2$ 表示为 tcp'_2,以此类推,则式(1)可改写为:

$$tcp_0, tcp'_1, tcp'_2, \cdots, tcp'_{n-1} \tag{2}$$

对密码算法进行操作合并,合并后操作的关键路径延迟可视为合并前各操作的关键路径延迟之和. 假设算法被合并为 m 步,则密码算法的数据吞吐率可以表示为:

$$\text{Throughput} = \frac{L}{m \times t_{\max}} \tag{3}$$

其中 L 为 $m \times t_{\max}$ 时间内经粗粒度可重构密码逻辑阵列处理的数据量,$t_{\max}$ 为算法映射 m 步中最长的关键路径延迟. 考虑到 CB、SB 由数据选择器电路构成,因此其关键路径延迟远小于算子的关键路径延迟. 根据式(3)可得,当数据量 L 及算法的拆分方式确定后,密码算法的数据吞吐率仅和算法映射时的关键路径延迟 $t_{\max}$ 相关,$t_{\max}$ 越小则密码算法的数据吞吐率越大.

$m=1$ 时:

$$\left.\begin{aligned} t_{\max} &= tcp_0 + tcp_1 + \cdots + tcp_{n-1} \\ t'_{\max} &= tcp_0 + tcp'_1 + \cdots + tcp'_{n-1} \end{aligned}\right\} \Rightarrow t_{\max} \leqslant t'_{\max}$$

$m=2$ 时:

$$t_{\max} = \max\{tcp_0 + tcp_1 + \cdots + tcp_{j-1}, tcp_j + tcp_{j+1} + \cdots + tcp_{n-1}\}$$

$$t'_{\max} = \max\{tcp_0 + tcp'_1 + \cdots + tcp'_{j-1}, tcp'_j + tcp'_{j+1} + \cdots + tcp'_{n-1}\}$$

假设

$$tcp_0 + tcp_1 + \cdots + tcp_{j-1} \geqslant tcp_j + tcp_{j+1} + \cdots + tcp_{n-1}$$

因为

$$t'_{\max} \geqslant tcp_0 + tcp'_1 + \cdots + tcp'_{j-1} \geqslant tcp_0 + tcp_1 + \cdots + tcp_{j-1} = t_{\max}$$

所以 $t_{\max} \leqslant t'_{\max}$

同理可证,m 为任意小于等于 n 的正整数时,$t_{\max} \leqslant t'_{\max}$ 均成立,因此,采用同样的方式对密码算法中算子进行合并,算子间经过的 CB、SB 越少,则密码算法的最终实现性能越大. 由于算子在 PE 内部采用全互连的方式连接,因此只需考虑 PE 间的连接,即被映射 PE 应尽可能集中分布,以减少连线长度,降低因互连资源不足而导致映射失败的风险.

根据上述研究可知,簇间互连线的长短直接影响了密码算法映射的成功率以及密码算法映射的性能,同时,考虑到粗粒度可重构密码逻辑阵列中各 PE 功能的同构性及位置的差异性,因此本文提出了一种智慧蚁群优化算法(Smart Ant Colony Optimization,SACO),即通过具有学习能力的蚂蚁,根据对已有映射知识储备及待映射密码算法数据流图特征的相似性,迅速确定起始簇的映射位置及各簇的相对位置,提升映射效率.

密码算法数据流图特征的相似性决定了经 SACO 算法收敛实现的映射结果存在相似性,因此,加入初始影响因子使得算法在预收敛路径上能够更快实现收敛,对于 SACO 算法即通过建立初始信息素浓度矩阵 $\boldsymbol{\Theta}$,提升算法在初始信息素浓度矩阵标识的路径上转移的概率. 每完成一次算法映射的同时更新初始信息素浓度矩阵,从而强化“学习”指导“实践”的映射过程,其正反馈过程如图 4 所示.

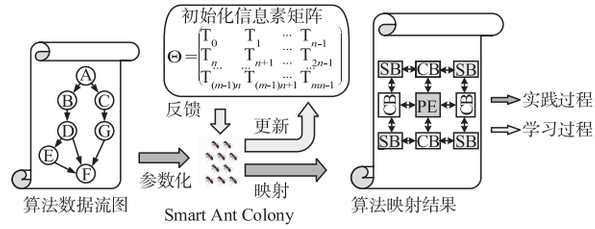


图4　SACO算法的学习与实践过程

SACO 算法步骤描述如下.

步骤 1:将密码算法数据流图划分为多个运算簇并进行参数化表示.

步骤 2:初始化信息素矩阵并依照初始化信息素矩阵中的参数概率性选择粗粒度可重构密码逻辑阵列映射起点,并将若干只蚂蚁置于起点位置.

步骤 3:依据广度优先搜索算法遍历划分后的密码算法数据流图,当完成某个簇的映射后,依据转移概率进行下一个簇的映射,转移概率计算方法为

$$p_{i,j}^{k}(t) = \begin{cases} \dfrac{[\tau_{i,j}(t) + \tau_{i,j}^{0}]^{\alpha} [\eta_{i,j}(t)]^{\beta}}{\sum\limits_{s \in J_k(i)} [\tau_{i,s}(t) + \tau_{i,s}^{0}]^{\alpha} [\eta_{i,s}(t)]^{\beta}}, & \text{如果} j \in J_k(i) \\ 0, & \text{否则} \end{cases} \tag{4}$$

其中 $J_k(i)$ 为下一步映射选择 PE 的集合,$\tau_{i,j}(t)$ 为 t 时刻 PE_i 与 PE_j 间信息素浓度,并以固定比例 ρ 进行挥发,$\eta_{i,j}(t)$ 为 t 时刻 PE_i 与 PE_j 间启发因子,反比与 PE_i 与 PE_j

间的最短距离(CB、SB 数目之和),α 为信息素的相对重要程度,β 为启发因子的相对重要程度.

定义规模为 $m \times n$ 的粗粒度可重构密码逻辑阵列初始化信息素浓度矩阵为

$$\Theta = \begin{pmatrix} T_0 & T_1 & \cdots & T_{n-1} \\ T_n & T_{n+1} & \cdots & T_{2n-1} \\ \cdots & \cdots & \cdots & \cdots \\ T_{(m-1)n} & T_{(m-1)n+1} & \cdots & T_{mn-1} \end{pmatrix} \tag{5}$$

其中

$$T_i = \begin{pmatrix} \tau^0_{i,0} & \tau^0_{i,1} & \cdots & \tau^0_{i,n-1} \\ \tau^0_{i,n} & \tau^0_{i,n+1} & \cdots & \tau^0_{i,2n-1} \\ \cdots & \cdots & \cdots & \cdots \\ \tau^0_{i,(m-1)n} & \tau^0_{i,(m-1)n+1} & \cdots & \tau^0_{i,mn-1} \end{pmatrix}, (0 \leqslant i \leqslant mn-1) \tag{6}$$

$\tau^0_{i,j}$ 为 PE_i 到 PE_j 间路径映射的初始化信息素浓度,其初值为 0,每完成一次密码算法映射,均需对初始化信息素浓度矩阵进行更新,更新规则如下:

$$\tau^0_{i,j} = \frac{\mu N^{\theta}_{i,j}}{N^{\theta}_{i,j} + \lambda}, (0 \leqslant i \leqslant mn-1, 0 \leqslant j \leqslant mn-1) \tag{7}$$

$$N_{i,j} = \begin{cases} \overline{N}_{i,j} + 1, & \text{不存在 } PE_i \text{ 到 } PE_j \text{ 的映射} \\ \overline{N}_{i,j} - 1, & \text{存在 } PE_i \text{ 到 } PE_j \text{ 的映射且} \overline{N}_{i,j} \geqslant 1 \\ 0, & \text{存在 } PE_i \text{ 到 } PE_j \text{ 的映射且} \overline{N}_{i,j} = 0 \end{cases} \tag{8}$$

其中 $\overline{N}_{i,j}$ 为 $N_{i,j}$ 的旧值,通过 $N_{i,j}$ 的更新来更新初始化信息素浓度矩阵.

步骤 4:在互连资源允许的条件下,采用最短路径法完成 PE 间互连,若最短路径已被占用,产生冲突,则在剩余可行解中继续寻找最短路径,若无任一路径可达,则当前蚂蚁死亡.

步骤 5:重复步骤 3 -4,同时更新 PE 集合的禁忌表,直至各蚂蚁均遍历完算法数据流图中的所有划分簇或死亡.

步骤 6:更新 PE_i 与 PE_j 间信息素浓度增量及信息素量,记录本次迭代路径,更新当前最优路径并清空 PE 集合的禁忌表.

步骤 7:判断是否达到预定迭代步数或出现停滞现象,若是,算法结束并输出最优路径,否则,跳转至步骤 2 进行下一轮迭代.

式(4)修正了传统蚁群算法中转移概率的计算方法,强化了已有映射过程对未知密码算法数据流图映射的影响,即提升了蚁群的记忆与学习能力. 当完成的密码算法映射量足够多,智慧蚁群具有足够大的"知识储备",则 SACO 算法能够实现快速的收敛,即能够更"有经验"完成基于粗粒度可重构密码逻辑阵列的算法映射. 采用 SACO 算法对图 3 所示密码算法数据流图进行映射,结果如图 5.

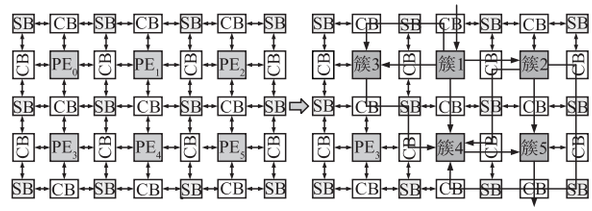


图5 基于粗粒度可重构密码逻辑阵列的算法映射

经过一定数量的迭代,映射方案趋于稳定,此时,密码算法映射性能可以用式(3)进行表示,且式(2)中的各个变量 $tcp_0, tcp'_1, tcp'_2, \cdots, tcp'_{n-1}$ 随映射方案的确定而确定. 由式(3)可知,当处理的数据量确定时,$m \times t_{max}$ 值越小,算法映射性能越高. 对于粗粒度可重构密码逻辑阵列,由于 PE 内部各 FU 所面向的操作不同,其设计复杂度差异较大,各 FU 的关键路径延迟不同. 因此,结合前期工作[6],在对密码算法数据流图进行划分簇时,依据映射目标,优化各 FU 输出寄存器的配置状态,从而实现密码算法映射的再优化.

4 实验与分析

为了验证本文提出的 SACO 算法的高效性以及其对密码算法高性能映射支持的有效性,本节基于课题组前期 55nm 工艺流片的粗粒度可重构密码逻辑阵列芯片,搭建了密码算法仿真验证测试平台,采用 Xilinx 公司 ZC7X020 FPGA 芯片并通过串口实现数据收发,结合典型密码算法 Blowfish、SM4、AES、DES、IDEA 及 MISTY 映射展开实验.

其中,粗粒度可重构密码逻辑阵列结构如图 6,包含 16 个同构运算元素 $PE_0 \sim PE_{15}$ 及 4 个异构运算元素 PE_S_0、PE_K_0、PE_S_1、PE_K_1. 其中 $PE_0 \sim PE_{15}$ 均包含 5 种功能单元,分别为算数类可重构运算单元(FU_AL)、置换类可重构运算单元(FU_BP)、逻辑类可重构运算单元(FU_LG)、非线性类可重构运算单元(FU_NF)以及空操作单元(FU_E).

采用 Intel Core i7-6650U@2.20GHz CPU 及 16GB 内存实验平台进行算法映射,工程代码基于 Python 3.5 版本编写. 首先清空初始化信息素矩阵并采用 SACO 算法,分别对上述 6 种典型密码算法进行映射,记录密码算法映射时间. 然后,对本文提出的 SACO 算法进行训练,为了生成更为贴合密码算法映射的初始化信息素矩阵,通过研究密码算法数据流图特点[6],随机产生用于映射的算法数据流图训练集. 实验结果如图 7.

实验结果表明随着训练组数的增加,算法映射编译时间呈整体下降趋势,且当训练组数大于3000组时,编译时间下降不再明显,即SACO算法达到成熟. 图中所示曲线产生的随机上升波动是由于所训练的映射算法与目标映射算法数据流图结构差异较大,因此对算法产生了一定干扰,但从整体来看,样本映射集的训练对缩短目标算法的编译时间起到了积极作用,其编译时间平均减少了约18.7%.

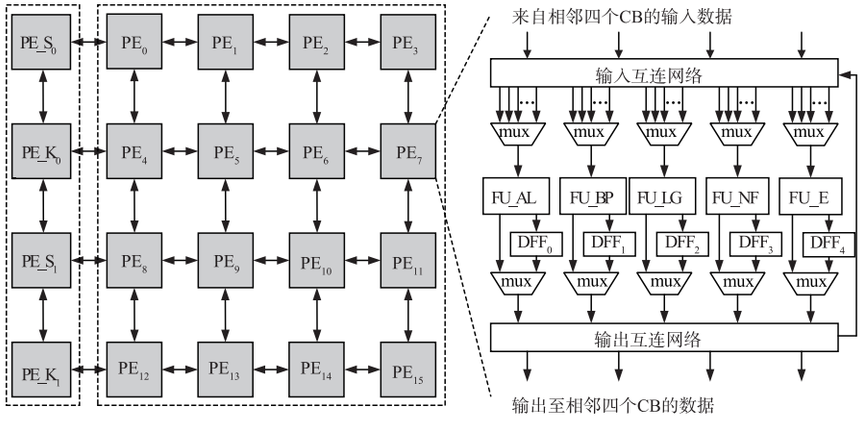


图6　粗粒度可重构密码逻辑阵列结构

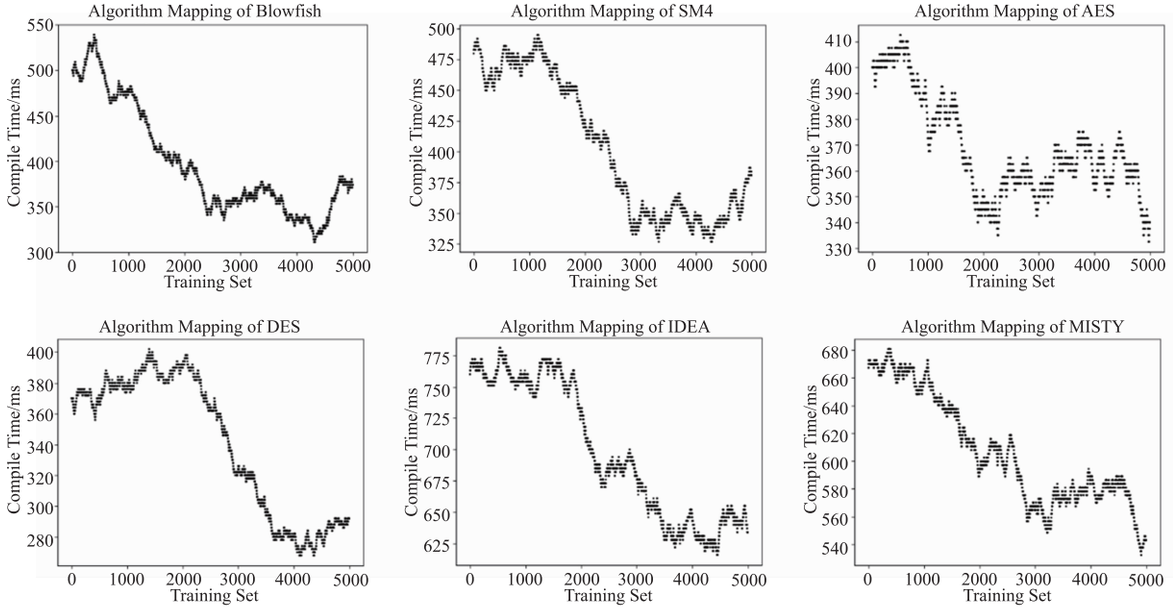


图7　基于SACO的算法映射

为了验证本文所提出算法的高效性,基于密码算法仿真验证测试平台,采用文献[1~5]所示映射算法对六种密码算法进行映射,由于文献中无算法源码,因此本文根据文献所示方法进行编程验证,考虑到代码实现的差异,因此对文献[1~5]中映射算法所消耗的编译时间进行了10%的优化,结果如图8所示.

由于本文所提出的SACO算法深入挖掘了粗粒度可重构密码逻辑阵列与密码算法数据流图之间的内在联系,并对密码算法数据流图进行了预处理优化,同时,采用大量训练集对算法映射过程进行训练,因此相

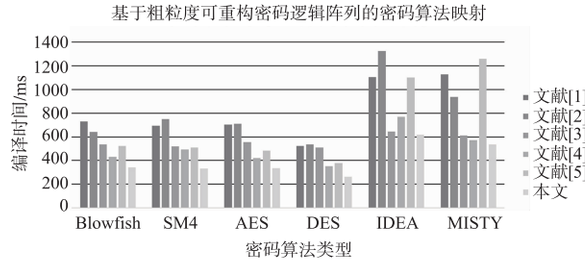


图8　基于粗粒度可重构密码逻辑阵列的密码算法映射

较于文献[1~5],本文所提出的SACO算法在编译时间上平均减少了约37.9%.

编译时间是衡量映射算法优劣的标志,但对于基于粗粒度可重构密码逻辑阵列的密码算法数据流图映射而言,其密码算法实现性能及功耗是衡量其映射实现的关键. 因此,基于密码算法仿真验证测试平台,分别采用SACO 算法及文献[1 ~5]所示算法对六种典型密码算法进行映射. 由于密文分组连接(Cipher Block Chaining, CBC)模式下算法实现性能是评价算法映射结果的重要指标,同时,基于前期研究可以得到,对于同一硬件平台,不同映射方法造成的功耗数值变化较小,因此,表 1 仅展示了 CBC 模式下的典型密码算法映射结果.

实验结果表明本文提出的 SACO 算法在平均降低编译时间 37. 9% 的同时能够使密码算法映射性能提升不低于文献[1 ~5]算法的映射结果,从而满足基于可重构密码逻辑阵列的密码算法高效映射.

表 1 典型密码算法映射结果

密码算法类型	映射方法											
	EPIMap[1]		SPKM[2]		PolyMAP[3]		DFGNet[4]		2-Step ACO[5]		SACO	
	编译时间 ms	实现性能 Mbps	编译时间 ms	实现性能 Mbps	编译时间 ms	实现性能 Mbps	编译时间 ms	实现性能 Mbps	编译时间 ms	实现性能 Mbps	编译时间 ms	实现性能 Mbps
Blowfish[7]	731	437	642	450	535	450	431	419	522	433	343	465
SM4	694	175	749	183	521	195	493	173	511	190	331	200
AES	703	403	711	421	555	440	420	415	485	380	335	440
DES	522	240	535	217	510	220	353	237	377	245	265	245
IDEA[8]	1105	157	1325	163	645	142	770	161	1101	163	617	176
MISTY[9]	1128	99	937	135	610	127	571	115	1258	126	535	138

5 结束语

为了实现基于粗粒度可重构密码逻辑阵列的密码算法数据流图高效映射,本文在对粗粒度可重构密码逻辑阵列结构进行参数化描述以及对密码算法数据流图进行划分的基础上,研究粗粒度可重构密码逻辑阵列结构与密码算法数据流图的内在关系,降低密码算法数据流图映射的复杂度. 结合机器学习及启发式算法,提出了基于以“已知”指导“未知”的算法映射思想,以降低算法编译时间,提高密码算法映射性能为目标,实现密码算法映射. 本文研究主要面向密码算法领域,但由于粗粒度可重构密码逻辑阵列是粗粒度可重构阵列在密码方向上的推广,因此本文研究工作对指导基于粗粒度可重构阵列的算法映射仍具有重要意义,下一步,将结合通用粗粒度可重构阵列及算法进行研究,优化智慧蚁群算法,提升算法的通用性.

参考文献

[1] MAHDI HAMZEH, AVIRAL SHRIVASTAVA, SARMA VRUDHULA. EPIMap: using epimorphism to map applications on CGRAs[A]. Proceedings of the 49th Annual Design Automation Conference[C]. San Francisco: IEEE, 2012. 1284 – 1291.

[2] JONGHEE W YOON, AVIRAL SHRIVASTAVA, SANGHYUM PARK, et al. SPKM: A novel graph drawing based algorithm for application mapping onto coarse-grained reconfigurable architectures[A]. 2008 Asia and South Pacific Design Automation Conference[C]. Seoul: IEEE, 2008. 776 – 782.

[3] LIU Dajiang, YIN Shouyi, LIU Leibo, et al. Polyhedral model based mapping optimization of loop nests for CGRAs[A]. Proceedings of the 50th Annual Design Automation Conference [C]. Austin: IEEE, 2013. 1 – 8.

[4] YIN Shouyi, LIU Dajiang, SUN Lifeng, et al. DFGNet: Mapping dataflow graph onto CGRA by a deep learning approach[A]. IEEE International Symposium on Circuits and Systems[C]. Baltimore: IEEE, 2017. 1 – 4.

[5] ZHOU Li, ZHANG Jianfeng, LIU Hengzhu. Ant colony algorithm for steiner tree problem in CGRA mapping[A]. International Conference on Information Science & Control Engineering[C]. Changsha: IEEE, 2017. 198 – 202.

[6] 杜怡然,南龙梅,戴紫彬. 可重构分组密码逻辑阵列加权度量模型及高能效映射算法[J]. 电子学报, 2019, 47(1): 82 – 91.
DU Yi-ran, NAN Long-mei, DAI Zi-bin. Reconfigurable block cryptographic logic array weighted metric model and high energy-efficient mapping algorithm[J]. Acta Electronica Sinica, 2019, 47(1): 82 – 91. (in Chinese)

[7] CANNIERE, CHRISTOPHE DE. Blowfish[A]. Encyclopedia of Cryptography & Security[C]. Springer, 2005. 48 – 49.

[8] 金晨辉,郑浩然. 密码学[M]. 北京: 高等教育出版社, 2009. 146 – 231.
JIN Chenhui, ZHENG Haoran. Cryptography[M]. Beijing: Higher Education Press, 2009. 146 – 231. (in Chinese)

[9] MATSUI MITSURU. New block encryption algorithm MISTY[J]. Lecture Notes in Computer Science, 1997, 1267(1267):54 – 68.

[10] LI Wei, ZENG Xiaoyang, NAN Longmei, et al. A reconfigurable block cryptographic processor based on VLIW architecture[J]. China Communications, 2016, 13(1):91 – 99.

[11] 冯晓,李伟,戴紫彬. 面向分组密码的可重构异构多核并行处理架构[J]. 电子学报, 2017, 45(6): 1311 – 1320.
FENG Xiao, LI Wei, DAI Zi-bin. Reconfigurable asymmetrical multi-core architecture for block cipher[J]. Acta Electronica Sinica, 2017, 45(6): 1311 – 1320. (in Chinese)

[12] HAN Jun, LI Yang, YU Zhiyi, et al. A 65 nm cryptographic processor for high speed pairing computation[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015, 23(4):692 – 701.

[13] YIN Shouyi, ZHOU Pengcheng, LIU Leibo, et al. Trigger-centric loop mapping on CGRAs[J]. IEEE Transactions on Very Large Scale Integration Systems, 2016, 24(5): 1998 – 2002.

作者简介

杜怡然　男. 1991 年 4 月出生,河南郑州人. 解放军信息工程大学计算机科学与技术专业博士研究生,从事安全专用芯片设计等有关研究.
E-mail:yrdu_ieu@163.com

杨　萱　女. 1984 年 11 月出生,陕西西安人. 江南计算技术研究所工程师,从事专用集成电路设计等有关研究.

戴紫彬　男. 1966 年 5 月出生,河南商丘人. 解放军信息工程大学教授,博士生导师,从事专用集成电路设计、芯片安全防护、信息安全芯片技术等有关研究.

南龙梅　女. 1981 年 11 月出生,陕西乾县人. 解放军信息工程大学讲师,从事安全芯片设计、集成电路技术等有关研究.

李　伟(通信作者)　男. 1983 年 11 月出生,天津人. 解放军信息工程大学副教授,从事体系结构、安全芯片设计、集成电路技术等有关研究.
E-mail:try_1118@163.com