一种密码专用可编程逻辑阵列的分组密码能效模型及其映射算法

李 伟* 高嘉浩 杜怡然 陈 韬

(战略支援部队信息工程大学 郑州 450001)

摘 要：密码专用可编程逻辑阵列(CSPLA)是一种数据流驱动的密码处理结构，该文针对不同规模的阵列结构和密码算法映射实现能效关系的问题，首先以CSPLA的特定硬件结构为基础，以分组密码的高能效实现为切入点，建立基于该结构的分组密码算法映射能效模型并分析影响能效的相关因素，然后进一步根据阵列结构上算法映射的基本过程提出映射算法，最后选取几种典型的分组密码算法分别在不同规模的阵列进行映射实验。结果表明越大的规模并不一定能够带来越高的能效，为取得映射的最佳能效，阵列的规模参数应当与具体的硬件资源限制和密码算法运算需求相匹配，CSPLA规模为4×4～4×6时映射取得最优能效，AES算法最优能效为33.68 Mbps/mW，对比其它密码处理结构，CSPLA具有较优的能效特性。

关键词：密码专用可编程逻辑阵列；分组密码；能效；映射

中图分类号：TP918.2 **文献标识码**：A **文章编号**：1009-5896(2021)05-1372-09

DOI: 10.11999/JEIT200079

Energy Efficiency Model and Mapping Algorithm of Block Cipher for Cipher Specific Programmable Logic Array

LI Wei GAO Jiahao DU Yiran CHEN Tao

(*Strategic Support Force Information Engineering University*, *Zhengzhou* 450001, *China*)

Abstract: Cipher Specific Programmable Logic Array (CSPLA) is a data stream-driven cryptographic processing structure. The relations between cryptographic mapping energy efficiency and array structures of different scales is considered in this paper. First, based on the specific hardware structure of CSPLA and block ciphers, an energy efficiency model of block cipher algorithm mapping based on this structure is established and related factors affecting energy efficiency are analyzed. Then the basic process of algorithm mapping on the array structure is discussed and a mapping algorithm is proposed. Finally, several typical block cipher algorithms are selected to perform mapping experiments on arrays of different scales. The results show that larger scale CSPLA does not necessarily bring higher energy efficiency. When the CSPLA scale is about 4×4～4×6 which achieves the best energy efficiency. In order to obtain the best energy efficiency, the scale parameter of the array should match the specific hardware resource constraints and cryptographic algorithm parameters. The optimal energy efficiency of AES algorithm is 33.68 Mbps/mW. CSPLA has better energy efficiency characteristics compared with other cryptographic processing structures.

Key words: Cipher Specific Programmable Logic Array (CSPLA); Block cipher; Energy efficiency; Mapping

1 引言

密码计算是一种典型的数据密集型计算形式，在信息安全领域发挥着重要作用，对其处理结构的计算性能有着一定的要求，同时随着不同类型密码算法的不断发展以及对密码处理结构攻击方法和手段的进步，对于密码处理结构还要求其具备一定的灵活性和安全性[1]。

密码专用可编程逻辑阵列(Cipher Specific Programmable Logic Array, CSPLA)作为一种新型的密码专用处理结构，迎合了密码计算的数据流处理特点，使其兼具一定的灵活性和安全性，因而对比密码专用电路、密码专用指令集架构处理器等结构更为适应当今密码处理任务[2]。CSPLA内部由专为密码处理而设计的运算单元规则排列而成，通过配置信息来形成特定的数据通路，从而完成特定的密码计算任务。

阵列内部越多的运算单元意味着更丰富的硬件资源，进而能够映射更大规模密码计算任务，达到更高的处理性能。随着阵列规模的不断扩大，整体

收稿日期：2020-01-19；改回日期：2020-06-15；网络出版：2020-07-24
*通信作者： 李伟 try_1118@163.com

结构的功耗和面积也将随之线性增长，若在映射过程中无法充分利用内部运算单元，将影响到整体的能效比和计算资源利用率。目前，国内外有很多相关的文献针对粗粒度可重构结构应用映射过程的相关算法、执行时间、实现性能等方面均有所深入研究[3–7]，但是密码计算任务在运算过程中和通用型计算任务相比而言在数据、存储等方面有较为明显的差异，相关模型并未针对密码阵列的具体结构进行优化，特别是基于不同规模的阵列结构的能效模型研究相对较少。

因此本文以分组密码算法为出发点，在设计的可重构密码处理单元基础上结合层次化的互连网络扩展了多种不同规模的CSPLA结构并构建了分组密码算法的能效模型，同时以尽可能提高映射时计算资源利用率为目标，提出了基于该模型的映射算法。选取几种典型分组密码算法在不同规模的阵列结构上进行映射实验，通过实验数据寻找到计算资源和整体能效比之间的平衡点，选取一个最优的计算资源规模以及排布结构，使得密码算法映射时达到最优的能效比。

2　密码专用可编程逻辑阵列

2.1　CSPLA基本结构

本文提出的密码专用可编程逻辑阵列CSPLA由计算系统、互连网络、共享存储类单元(Shared Store Unit, SSU)、控制器以及接口部分组成。其中作为主体部分的计算系统由可重构密码计算单元(Reconfigurable Cipher Process Element, RCPE)规整排列组成，RCPE内部由不同类型的密码算子单元、多级寄存网络、分布式控制器和内层互连网络组成，粒度为32 bit，主要包括算术类AG、逻辑类LG、置换类BP和非线性类NF4类单元，能够满足对称密码算法中的大部分运算操作。其结构如图1所示。

RCPE中内层互连网络可以灵活选择外部数据的来源以及去向，实现单个单元数据流循环处理，能够使得阵列内部硬件资源利用率一直处于一个较高的水平，有助于提升整体能效比。CSPLA中规整排列的相邻RCPE之间通过双向数据通路连接起来，同时将计算系统、SSU以及输入输出接口互相连通，灵活实现数据在阵列内部的传输、交换。SSU主要由寄存器堆和SRAM组成，完成密码算法中子密钥、常数、中间数据以及配置信息的存取；接口则由两个FIFO组成，分别对应数据的输入和输出缓存。

CSPLA中第1级RCPE为输入级单元，内部互连网络同INFIFO相连，用于接收待处理的输入数据；最后一级RCPE为输出级单元，内部互连网络同OUTFIFO相连，用于输出处理完成的数据。CSPLA同其他阵列结构一样，内部计算资源具备较为明显的可扩展性，通过互连网络，内部单元的排列规模可以进行灵活调整，以此适应不同场景下密码处理的需求。

2.2　CSPLA硬件资源模型

CSPLA作为实现密码算法的硬件资源平台，本文首先对阵列结构进行集合形式的抽象[8]，为之

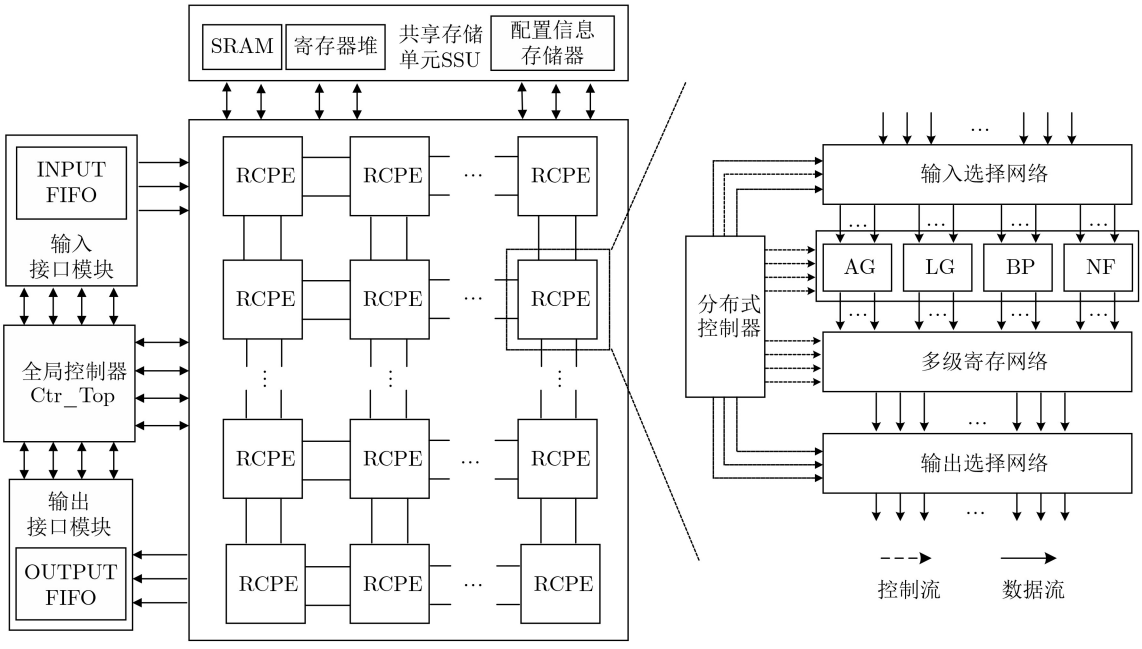


图 1 CSPLA整体结构及RCPE结构示意图

后的映射过程提供具体的依据，将一个$m\times n$规模的CSPLA的硬件资源定义如下：

(1) 将CSPLA中所有的硬件资源根据映射功能上的不同划分为计算、互连、存储、输入输出接口和控制5类元素，则CSPLA的硬件资源模型可以视为一个集合$A=\{\text{FUN},\text{CON},\text{CTR},\text{MEM},\text{IO}\}$；

(2) CSPLA中的计算资源主要由规则排布的可重构密码计算单元中的4类算子组成，其集合可表示为 $\text{FUN}=\{(\text{AG},\text{LG},\text{BP},\text{NF})_{i,j}|i=1,2,\cdots,m;j=1,2,\cdots,n\}$；

(3) CSPLA中的互连资源主要包括RCPE内部的数据传输网络和RCPE外部的互连网络，可表示为 $\text{CON}=\{(\text{Con\_in},\text{Con\_ex})_{i,j}|i=1,2,\cdots,m;j=1,2,\cdots,n\}$；

(4) CSPLA中的存储资源主要包括RCPE内部的寄存资源和共享存储单元SSU，可表示为$\text{MEM}=\{\text{SSU},(\text{Rst})_{i,j}|i=1,2,\cdots,m;j=1,2,\cdots,n\}$；

(5) CSPLA中的输入输出接口由输入级单元和输出级单元同外部FIFO的数据通路，每一路均为32 bit，可表示为 $\text{IO}=\{\text{Input}_{1,j},\text{Output}_{m,j}|j=1,2,\cdots,n\}$；

(6) CSPLA中的控制结构包括处于全局控制器和各可重构运算单元的分布式控制器，可表示为 $\text{CTR}=\{\text{Ctr\_top},(\text{Ctr\_cuc})_{i,j}|i=1,2,\cdots,m;j=1,2,\cdots,n\}$。

整个映射过程就是针对特定密码计算任务在硬件模型的集合内部限定的资源上进行数据的调度和分配，最终映射的结果为包含上述全部5类元素的配置信息。

CSPLA优势在于数据处理的多路并行性，从理论上CSPLA可以通过无限堆叠计算单元来扩充计算资源，实现更大规模的密码计算任务以提升吞吐率。随着规模的扩大，阵列结构的面积和功耗将显著增加。内部各类资源不一定能满足数据实时处理的需求，从而影响到整体能效。因此阵列结构规模并非越大越好，单纯以性能指标来评价其结构规模的优劣往往不足以说明问题。本文拟从密码处理的能效作为评价CSPLA结构的参数指标。对于阵列而言必然存在一个相对的规模，在处理特定密码任务时达到相对最优能效。

3 CSPLA能效模型

3.1 模型建立

针对模型需求，结合分组密码和阵列结构的基本特点，首先给出如下定义：

定义1 循环空间和非循环空间：将待映射的分组密码中需要反复执行多次的操作组成的集合称为循环空间C；算法过程中仅仅执行1次的操作构成的集合为非循环空间M。两者关系为$C\cap M=\varnothing$，且$C\cup M=\text{Block}$，即C和M交集为空，两者共同组成完整的待映射的分组密码算法。

定义2 空间映射：$C=(V,E)$和$M=(V,E)$是对应循环空间和非循环空间的数据流图，包括操作节点和数据依赖边，空间映射就是在阵列的硬件基础上生成一个由计算资源和互连资源组成的数据路径。

定义3 RCPE计算能力：对算子单元和循环空间中的运算操作归一化处理，RCPE能够完成循环空间中操作比例定义为RCPE的计算能力α，显然对于不同的算法或不同的映射方案α值不同且$0\le t\alpha\le 1$。

定义4 最大算子频次：将待映射算法中映射单分组独立操作所需最多的算子数目定义为最大算子频次$N_{\max}$，对于$m\times n$规模的阵列，映射的并行度受限于$N_{\max}$。

CSPLA最终能效比取决于密码算法实现的吞吐率和功耗。在实现过程中一方面要提高密码算法吞吐率，另一方面则尽可能降低阵列的整体功耗。对于一个$m\times n$规模的阵列，吞吐率可表示为

$$Tp_{m\times n}=D\times F \tag{1}$$

吞吐率表示每秒数据处理量，能够直观体现阵列的密码计算能力。F为阵列运行频率，显然阵列运行频率越高，吞吐率也越高。而D为阵列运行周期平均处理数据量，单位为bit，可表示为

$$D=\frac{Q\times W}{T_{\text{cycle}}} \tag{2}$$

其中，Q为数据分组数量，W为算法分组长度，T_{cycle}为完成数据处理所需的运算周期。由式(2)分析可知，对于特定的密码算法，W为固定值，若要提高阵列运行周期内数据处理量，一方面是尽可能增加内部并行处理的数据量；另一方面则是缩短运算的时钟周期。假设各分组内部映射并行度为i，不考虑数据依赖关系和其他非计算延时的情况下，算法的运算周期T_{cycle}可表示为

$$T_{\text{cycle}}=\sum_{i}^{i_{\max}}\left(\frac{W\times Q}{\alpha}\times\frac{1}{i}\right) \tag{3}$$

$m\times n$规模的阵列功耗可表示为式(4)，其中P_{FIFO}和P_{SSU}分别为FIFO和共享存储单元的功耗，P_{S}和P_{D}分别为单个RCPE的静态功耗和动态功耗。随着阵列规模的扩大，功耗的各项数值必将有一定幅度的增加。映射的最终能效模型可表示为式(5)

$$P_{m\times n}=P_{\mathrm{FIFO}}+P_{\mathrm{SSU}}+m\times n\times P_S+\sum P_D \tag{4}$$

$$\mathrm{EFF}\,(m,n,i,Q)=\frac{\mathrm{T}p_{m\times n}}{P_{m\times n}}=\frac{Q\times W\times F_{m\times n}}{T_{\mathrm{cycle}}\times P_{m\times n}} \tag{5}$$

3.2 模型参数分析

3.2.1 运算周期T_{cycle}

数据在运算过程中的传输可能受到互连资源和接口的限制而产生一定程度的延时，与此同时还需要占用阵列内部的互连资源进行数据传输。当内部处理的数据较多时，有限的互连资源无法满足多组数据同时传输的需求，阵列输入、输出FIFO也可能出现无法满足数据输入输出需求的情况，这时数据子块处理完成后产生阻塞，浪费单元计算性能的同时也产生了一定的传输延时。CSPLA的输入输出FIFO每个时钟周期内只能有1组32 bit数据进出，假设算法数据块以最大粒度进行处理，当计算资源对数据的实时处理效率超过了接口的输入输出效率，即

$$\frac{W\times Q}{32}>m\times n\times\alpha \tag{6}$$

RCPE必须等待接口的数据输入或输出才能继续执行运算任务，从而产生接口延时。在实际运算过程中，阵列内部的互连资源总是有限的，接口往往不是限制运算性能发挥的主要因素。一般在计算资源满足映射需求前提下只需考虑外部互连资源是否满足资源限制。RCPE间每一个传输通路都可以实现1组数据的传输，对于一个$m\times n$规模的阵列，RCPE外层的互连资源量可以表示为$(m-1)n+m(n-1)$。

从分组密码处理横向角度来看，每个时钟周期内各个数据子块在处理上很少有数据交互，彼此的运算过程较为独立，使用的互连资源相对较少；从纵向角度来看，分组密码采用的是迭代型结构，核心的轮运算过程中各个数据子块都与前一时钟周期中间数据存在着直接的数据交互，各个顺序排列的算子单元间存在着直接的数据交互，互连资源开销较大。假设数据运算过程中间不存在横向的数据传输需求，则此时运算过程额外产生的传输延时可表示为

$$T_{\mathrm{trans}}=\sum_{i}^{i_{\max}}\left(\frac{W\times Q}{32m}-\frac{W\times Q}{\alpha}\times\frac{1}{i}\right) \tag{7}$$

此外由于运算单元在运行过程中计算任务的改变而暂停计算，等待重新写入配置信息后再次进入运算状态也将产生一定的时钟延时。RCPE中完成动态重构的过程需要两个时钟周期，设β为RCPE的动态重构能力，分组内部数据重构信息量为R，则由完成运算任务而产生的动态重构延时可表示为

$$T_{\mathrm{recfg}}=2\times\sum_{i}^{i_{\max}}\left(\frac{R\times Q}{\beta}\times\frac{1}{i}\right) \tag{8}$$

当阵列内部单元需要进行配置重构时，可将待重构计算单元的任务暂停并转移至其它空闲的计算单元，如图2所示，通过运算任务的合理分配从而将重构的时钟延时隐藏在整体的计算任务周期中，在这种情况下重构产生的时钟延时对最终能效的影响可以通过计算资源合理调度得到有效削减[9]。假设互连资源能够完全满足任务转移需求，则产生的动态重构延时可表示为

$$T_{\mathrm{recfg}}=2\times\sum_{i}^{i_{\max}}\left[\frac{R\times Q}{\beta}\times\frac{1}{i}-\left(m\times n-\frac{Q\times W}{32\alpha}\right)\times\frac{1}{\beta}\times\frac{1}{i}\right] \tag{9}$$

当阵列内部多个分组数据并行处理时，处于空闲状态的计算单元减少，若没有充足的空闲单元承接转移的计算任务，待重构的计算单元只能暂停相应的计算任务，将中间数据写入存储资源，待重构完成再进入运算状态，由此在运算过程中引入了重

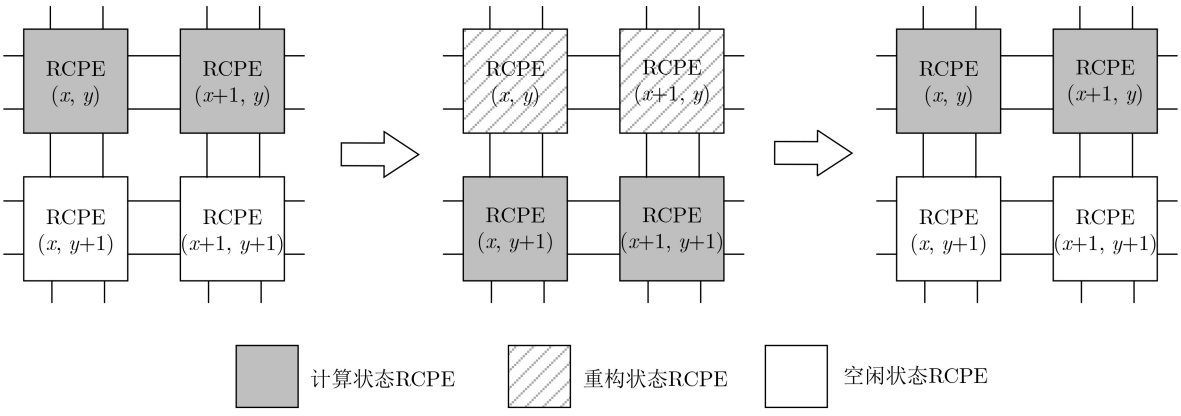


图 2 计算任务转移隐藏动态重构延时

构延时。任务处理周期将延长，导致阵列的吞吐率降低。

$$\begin{aligned}\mathrm{EFF}\,(m,n,i,Q) &= \frac{Tp_{m\times n}}{P_{m\times n}}\\ &= \frac{Q\times W\times F_{m\times n}}{(T_{\mathrm{cycle}}+T_{\mathrm{trans}}+T_{\mathrm{recfg}})\times P_{m\times n}}\end{aligned} \tag{10}$$

综合考虑运算过程中的非计算延时，映射的最终能效模型可以改写为式(10)。在实际映射过程中，数据在运算过程中不可能完全不存在横向的数据互连，同时当分组数量达到一定程度时，互连资源可能无法满足数据无阻塞传输的需求，从而成为限制能效提升的主要因素，由此必然使整个算法的处理周期延长，产生的非计算延时大小与具体的分组密码算法及映射方案有关。

3.2.2 整体功耗$P_{m\times n}$

CSPLA中RCPE组成的计算资源在整体资源中占据绝大部分比重。由上文分析可知，阵列整体功耗$P_{m\times n}$将随着阵列规模的扩大而增大。基于课题前期的研究，在规模参数m和n变化的过程中，输入输出FIFO 和SSU的功耗开销即P_{FIFO}和P_{SSU}相对稳定，仅有微小幅度增加，而静态功耗P_{S}变化幅度最为显著，与阵列规模参数呈现出线性变化的关系，且单个RCPE的静态功耗一般是其动态功耗的数10倍。

当阵列规模上升到一定程度后，$m\times n$个RCPE产生的静态功耗P_{S}成为整体功耗的主要来源，在$P_{m\times n}$开销中占据的比重远大于其它几项功耗开销，则CSPLA整体功耗的表达可简化为

$$P_{m\times n}=m\times n\times P_{\mathrm{S}} \tag{11}$$

由此可见，映射能效受到运行频率、运算周期、功耗等因素的共同影响。对于$m\times n$规模的阵列，映射过程中应当充分利用阵列内部的运算单元，提高硬件资源的利用率，以平衡随着规模增大而引入的功耗开销，使有限的运算单元尽可能处在运算状态中，能够使得映射达到相对最优能效值。

3.2.3 阵列规模参数

目前分组密码算法的分组长度大多为64 bit和128 bit，其中128 bit是更为主流的分组长度。相对集中的分布范围有利于选择CSPLA的规模，即m和n的取值。m和n的值越大，即阵列中RCPE数量越多，包含的各类资源也越丰富。理想情况下阵列映射的最大分组数量Q可表示为式(12)。

$$Q=\left\lfloor\frac{m\times n}{N_{\max}}\right\rfloor \tag{12}$$

从阵列的横向规模来看，RCPE处理粒度为32 bit，m取值应该在适应算法分组长度的基础上进行选择，当每一级设置4个RCPE时能够基本满足128 bit的单分组数据处理需求。与此同时第1级作为输入级，其内部寄存器与输入FIFO的接口直连，考虑到接口电路负载能力，横向RCPE数量应该控制在10以内，因此本文中将m的取值设定为4或者8。

从阵列的纵向规模来看，对于分组密码的工作模式而言，在ECB模式下不存在数据依赖关系时，将单周期内的操作在每一级RCPE上完全流水展开，达到较高的映射性能。但实际应用中分组密码更多的情况下使用的是CBC和OFB这类存在前后数据依赖的工作模式，无法实现算法的流水展开。纵向上的级数过大使得阵列整体的面积和功耗开销变得难以承受，从而限制其应用场景。因此本文中考虑CSPLA的自循环处理机制，结合分组密码的多个分组并行处理方法，考虑到SSU单元的两级扩展性，将n取值从2开始并进行递增。通过对不同规模CSPLA的映射分析，选取出相对最优能效的阵列规模结构。

3.3 映射算法

在CSPLA上进行算法的分组并行映射的基本步骤如下[10]：

(1) 根据待映射算法的操作划分对应的循环空间和非循环空间，生成对应的数据流图；

(2) 设定分组数量，多个数据子块的处理首先需要在满足阵列计算资源限制下选取合适的值；

(3) 为单分组的数据选取映射的可重构计算单元RCPE，以及内部具体的算子单元；

(4) 由数据流图，即根据映射过程中数据的传输关系完成互连配置信息的生成，搭建相应的数据路径，涵盖除控制之外的其他4类硬件资源；

(5) 根据数据路径和时序关系，生成控制配置信息，包括读写控制、动态重构控制等。

基于上文得到的能效模型，为得到尽可能高的能效值，在相关资源满足前提的情况下，针对CSPLA的自循环单分组并行映射，提出如表1所示的算法，输入为阵列的硬件资源模型和待映射的分组密码算法，输出为算法具体的配置信息，算法的约束条件为CSPLA包含的有限硬件资源。

首先对于待映射的分组密码算法，通过操作划分，确定算法的循环空间和非循环空间并生成对应的数据流图(行(1)、行(2))。循环空间中的运算操作一般利用RCPE的自循环运算实现，而非循环空间中的运算操作在单次运算完成后还需要考虑数据传输和配置重构的问题。其次根据最大算子频次初

表 1　分组密码的自循环单分组并行映射算法

输入：$A=\{\text{FUN},\text{CON},\text{CTR},\text{MEM},\text{IO}\}$, $\text{Block}=\{I,R,L\}$

输出：$\text{Map}=\{\text{Text}_{\text{FUN}},\text{Text}_{\text{CON}},\text{Text}_{\text{CTR}},\text{Text}_{\text{MEM}},\text{Text}_{\text{IO}}\}$

(1) $\text{Block}=\{C,M\}\leftarrow\text{Block}=\{I,R,L\}$

(2) generate $C=(V,E)$ and $M=(V,E)$

(3) initial $a=\left\lfloor\dfrac{m\times n}{N_{\max}}\right\rfloor$

(4) **while** $E(a\cdot C)\not\subset$ IO and $E(a\cdot C)\not\subset$ con_ex do

(5) $a=a-1$

(6) **end while**

(7) **for** each $\text{op}_i\in\text{Block}$ do

(8) $Q=a$

(9) $\{(\text{AG},\text{LG},\text{NF},\text{BP})_{i,j}|i=1,2,\cdots,m;j=1,2,\cdots,n\}\leftarrow\text{op}_k$,

(10) update FUN and MEM

(11) $\{(\text{Con\_in},\text{Con\_ex})_{i,j}|i=1,2,\cdots,m;j=1,2,\cdots,n\}\leftarrow$
$<\text{op}_{i-1},\text{op}_i>$

(12) update IO and CON

(13) **end for**

(14) $\text{FUN}_C\leftarrow V_n(C)$, $\text{Con\_ex}_C\leftarrow E_n(C)$

(15) $\text{FUN}_M\leftarrow V(M)$, $\text{Con\_ex}_M\leftarrow E(M)$

(16) generate CTR

(17) return Map

步设定映射并行度，同时需要考虑互连、接口资源是否符合约束条件，在计算和互连两类元素均满足约束条件的情况下，可以确定并行度并继续映射，否则应当调整Q值直至满足约束条件(行(3)～行(6))。在映射过程中针对每一个独立操作选择RCPE中合适的算子单元，并确定前后操作的互连关系，更新操作映射后的阵列计算、存储以及接口、互连资源信息(行(7)～行(13))。将循环空间和非循环空间的数据流图生成全部的计算和互连配置信息的集合，并生成与映射方案对应的控制配置信息(行(14)～行(16))，最终输出包含相关5类资源的全部配置信息。

4　实验及参数分析

4.1　映射实验

得益于CSPLA结构的可扩展性，通过对RCPE和SSU的数量、排列方式以及互连网络的调整可以较为灵活地搭建不同规格的阵列结构。使用Verilog HDL对不同规格的CSPLA设计进行描述，并利用EDA工具对设计进行映射的仿真分析和数据的验证测试。

本文选取了AES, SM4和DES 3个最为典型的分组密码算法，在不同规模的阵列上进行不同数量的单分组映射实验，得到的参数结果如表2所示，其单项指标参数随阵列规模的变化情况如图3所示。

4.2　实验结果分析

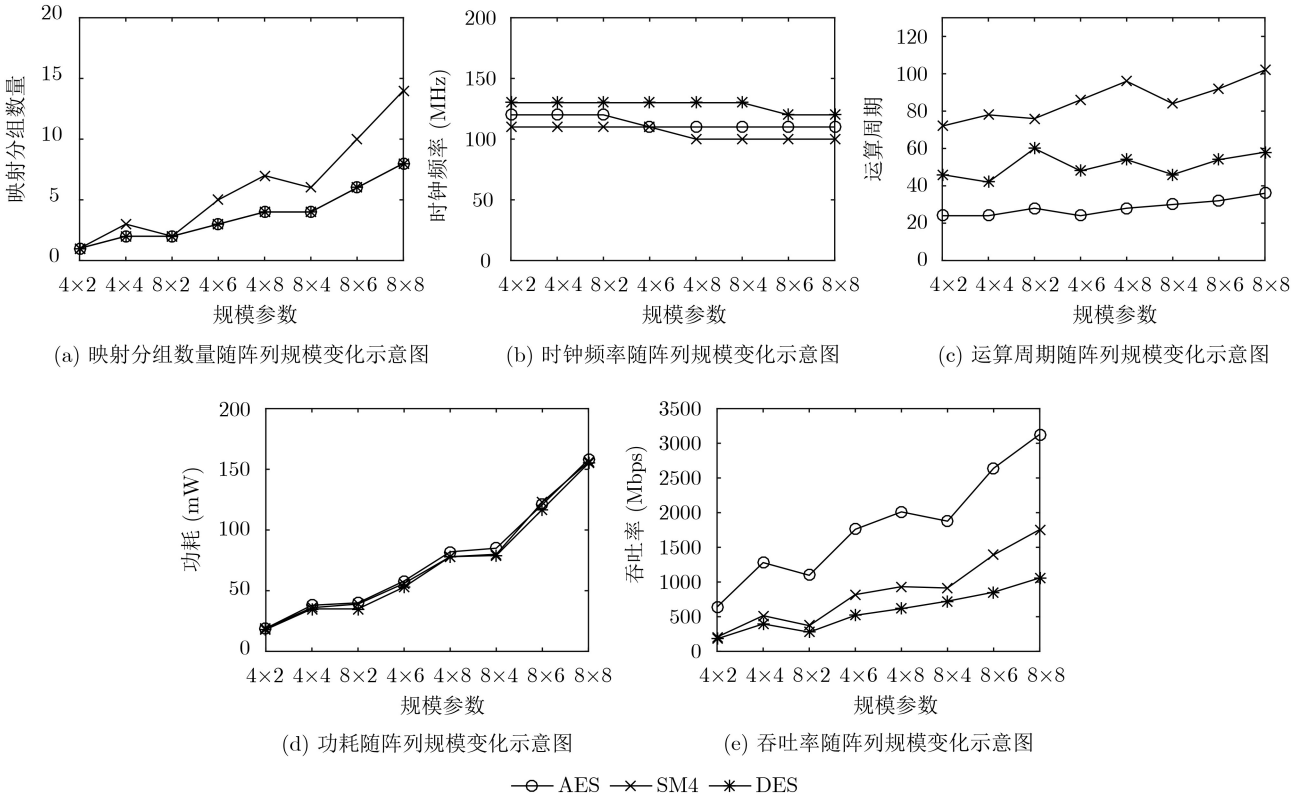
从表2中的数据分析可知，阵列规模的扩大意味着其内部可以容纳更多分组的数据的映射，并行映射分组数据数量不断增大，在性能上带来了吞吐率的显著提升，实际的映射并行度和最终的运算吞吐率则与具体算法有关。而同一算法在不同规模阵列上的运行频率随规模参数的变化并不明显，这是因为运行频率取决于映射方案中的关键路径，算法所映射的算子单元组成了内部数据路径，规模的变换并不会改变算法映射中关键路径的延时。

CSPLA整体功耗则与规模参数呈现出线性关系，规模越大则整体功耗开销越大。为了保持能效收益，在阵列上进行多组数据的并行映射，随着资源使用程度趋于饱和，产生的非计算延时导致平均运算周期变长，导致吞吐率偏低的同时能效随之降低。由此得出的不同CSPLA规模参数下的能效数据如图4所示。

当阵列结构为4×2时，内部的硬件资源较为紧张，而扩展为4×4时，阵列各类资源都更加充裕，例如在4×2规模的阵列上映射DES算法，在运算过程中需要对BP单元以及部分互连网络进行重配，

表 2　典型分组密码算法映射参数

参数	映射分组数量				运算周期				时钟频率(MHz)				功耗(mW)				吞吐率(Mbps)			
规模	4×2	4×4	4×6	4×8	4×2	4×4	4×6	4×8	4×2	4×4	4×6	4×8	4×2	4×4	4×6	4×8	4×2	4×4	4×6	4×8
AES	1	2	3	4	24	24	24	28	120	120	110	110	19	38	59	82	640	1280	1760	2011
SM4	1	3	5	7	72	78	86	96	110	110	110	100	19	36	57	78	207	515	819	933
DES	1	2	3	4	46	42	48	54	130	130	130	130	18	35	56	78	181	396	520	616
参数	映射分组数量				运算周期				时钟频率(MHz)				功耗(mW)				吞吐率(Mbps)			
规模	8×2	8×4	8×6	8×8	8×2	8×4	8×6	8×8	8×2	8×4	8×6	8×8	8×2	8×4	8×6	8×8	8×2	8×4	8×6	8×8
AES	2	4	6	8	28	30	32	36	120	110	110	110	40	82	121	168	1097	1877	2640	3129
SM4	2	6	10	14	76	84	92	102	110	100	100	100	39	80	123	166	371	914	1391	1756
DES	2	4	6	8	60	46	54	58	130	1300	120	120	35	79	117	155	277	724	853	1059



(a) 映射分组数量随阵列规模变化示意图 (b) 时钟频率随阵列规模变化示意图 (c) 运算周期随阵列规模变化示意图

(d) 功耗随阵列规模变化示意图 (e) 吞吐率随阵列规模变化示意图

图 3 各项参数随阵列规模的变化示意图

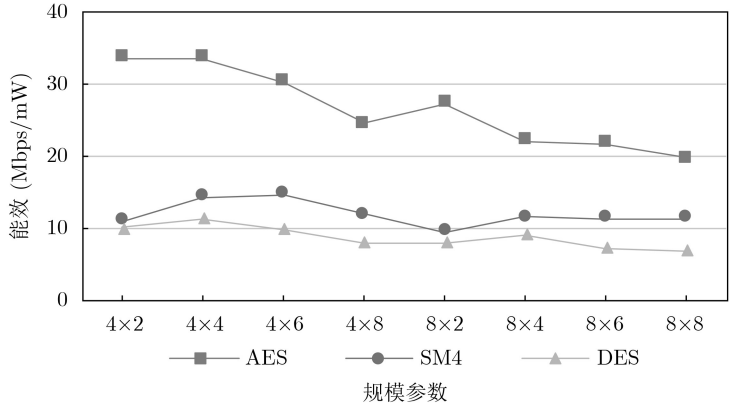


图 4 典型分组密码算法映射能效

由此产生了动态重构延时。而在4×4阵列上映射时则能够保证计算资源的充足，仅需对互连信息进行重配，运算周期反而降低。因此在保障基本运算能力前提下，阵列规模不宜过小。当阵列横向规模不变，纵向规模继续扩大时，即阵列结构为4×6或4×8时，多组数据并行映射带来资源使用的拥挤，产生的非计算延时特别是其中占主要部分的传输延时将使得运算周期延长2～10个时钟周期不等，同时伴随着整体功耗开销增大，从而几种典型算法的能效约有10%及以上不同幅度降低，对于横向参数为8时的变化趋势也几乎类似。

当纵向规模相同时，阵列的横向规模从4扩展至8时，CSPLA内部的硬件资源能够满足更多数据的并行映射，但是功耗也同步增加，且阵列接口FIFO的输出能力总是有限的，特别是当数据填充耗时高于分组数据在每一级处理周期时，就将存在接口延时导致计算资源闲置的情形。平均而言横向规模为8时单个分组数据所需的完整运算周期普遍多于规模为4的阵列结构，因而在运算周期延长4～6个时钟周期的情况下能效将受到一定程度的影响。其中不同算法由于自身相关参数的差异受影响的程度不同，最终几种典型分组算法的能效有5%～34%不同程度的降低。结合上述分析可知，AES和DES算法在4×4规模时分别取得相对最优映射能效33.68和11.31 Mbps/mW, SM4算法在4×6规模时取得最优能效14.63 Mbps/mW。

表 3　AES算法相关参数对比

处理结构	工艺(nm)	换算工艺(nm)	性能(Mbps)	功耗(mW)	能效(Mbps/mW)	等价能效(Mbps/mW)
CryptoManiac[11]	250	55	64	606	0.11	0.50
SophSEC[12]	130	55	654	325	2.01	4.76
文献[13]	180	55	1190	285	4.18	13.67
Cryptoraptor[14]	45	55	128000	6130	20.88	17.08
REMUS_LPP[15]	65	55	2840	103	27.57	32.59
本文(4×4)	55	55	1280	38	33.68	33.68
本文(4×8)	55	55	2011	82	24.52	24.52
本文(8×8)	55	55	3129	168	19.80	19.80

密码处理结构普遍采用AES算法实现的相关参数作为评价指标，因此本文同样选取AES算法在几种典型规模的CSPLA上的映射参数同其它结构进行比对，得到的结果如表3所示。

本文参考了文献[16]中的工艺换算方法能效进行了简单等价，该方法无法消除工艺偏差带来的影响，仅作为参考依据。对比表3中结果可以看出在55 nm工艺下，CSPLA(8×8)得益于数据流计算的优势，其整体结构的计算效率更高，在能效上对比文献[11]、文献[12]中的指令集密码处理结构有明显提升，而对比文献[14]中同为阵列类型的处理结构在能效上则只存在小幅度优势。根据本文所提能效模型及映射算法，通过适当控制阵列结构规模的同时兼顾较高的资源利用率，可以适当缩减运算过程中不必要的非计算性延时，虽然规模的降低导致了吞吐率的降低，但得益于功耗开销的显著降低，因而映射能效反而表现出一定提升。此时CSPLA(4×4)的映射能效相较于8×8规模时有明显提高，对比文献[14]中的结构能效提升了将近1倍，略优于文献[15]中所提出的结构。

综上所述，分组密码算法在CSPLA的映射能效并不会随规模的扩大而持续增加。CSPLA的横向规模参数应当和接口输入输出能力、算法运算位宽相适应，一方面在较短的时间内完成单级计算的数据处理填充，另一方面可以充分开发算法分组内的处理并行性；CSPLA的纵向规模参数应与算法计算周期、资源需求量相适应，以尽可能在提高资源利用率的同时缩短运算周期，从而达到最优的能效值。本文提出的能效模型及其映射算法能够有效地实现分组密码算法在CSPLA结构上的高能效映射。

5　结束语

本文提出了一种密码专用可编程逻辑阵列结构CSPLA的规模参数与映射能效的关系，在一种确定的阵列结构的基础上，结合分组密码算法单包并行处理的特点，从阵列的横向和纵向规模参数出发，分析了影响阵列最终映射能效的相关因素，建立了与硬件结构相匹配的能效模型并提出了对应的映射算法，结合典型分组密码算法进行了映射实验分析，结果表明阵列规模的持续扩大并不会带来密码算法映射能效的持续提升。在实现多个分组并行处理的情况下阵列引入的非计算延时会延长平均运算周期，降低整体的运算吞吐率，无法平衡整体增加的功耗开销，导致能效降低。在应用过程中应灵活确定阵列的规模参数，以适应不同情形下的指标需求。本文提出基于阵列结构的分组密码算法能效模型，对于序列密码算法、杂凑密码算法的高能效映射实现也具有一定的适用价值。

参 考 文 献

[1] LIU Leibo, WANG Bo, and WEI Shaojun. Reconfigurable Computing Cryptographic Processors[M]. Beijing: Publishing House of Science, 2018: 5–8.

[2] WANG Bo and LIU Leibo. Dynamically reconfigurable architecture for symmetric ciphers[J]. *Science China Information Sciences*, 2016, 59(4): 042403. doi: 10.1007/s11432-015-5381-z.

[3] ANSALONI G, TANIMURA K, POZZI L, *et al*. Integrated kernel partitioning and scheduling for coarse-grained reconfigurable arrays[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012, 31(12): 1803–1816. doi: 10.1109/TCAD.2012.2209886.

[4] 杨子煜, 严明, 王大伟, 等. 面向CGRA循环流水映射的数据并行优化[J]. 计算机学报, 2013, 36(6): 1280–1289. doi: 10.3724/SP.J.1016.2013.01280.
YANG Ziyu, YAN Ming, WANG Dawei, *et al*. Data parallelism optimization for the CGRA loop pipelining mapping[J]. *Chinese Journal of Computers*, 2013, 36(6): 1280–1289. doi: 10.3724/SP.J.1016.2013.01280.

[5] SHAO Shengjia, YIN Shouyi, LIU Leibo, *et al*. Map-reduce inspired loop parallelization on CGRA[C]. 2014 IEEE

International Symposium on Circuits and Systems (ISCAS), Melbourne, Australia, 2014: 1231–1234. doi: 10.1109/ISCAS.2014.6865364.

[6] 戴紫彬, 曲彤洲. 基于预配置和配置重用的粗粒度动态可重构系统任务调度技术[J]. 电子与信息学报, 2019, 41(6): 1458–1465. doi: 10.11999/JEIT180831.
DAI Zibin and QU Tongzhou. Task scheduling technology for coarse-grained dynamic reconfigurable system based on configuration prefetching and reuse[J]. *Journal of Electronics & Information Technology*, 2019, 41(6): 1458–1465. doi: 10.11999/JEIT180831.

[7] YIN Shouyi, LIU Dajiang, PENG Yu, *et al*. Improving nested loop pipelining on coarse-grained reconfigurable architectures[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016, 24(2): 507–520. doi: 10.1109/TVLSI.2015.2400219.

[8] 孙康. 可重构计算相关技术研究[D]. [博士论文], 浙江大学, 2007.
SUN Kang, Research on reconfigurable computing technologies[D]. [Ph. D. dissertation], Zhejiang University, 2007.

[9] WANG Yansheng, LIU Leibo, YIN Shouyi, *et al*. On-chip memory hierarchy in one coarse-grained reconfigurable architecture to compress memory space and to reduce reconfiguration time and data-reference time[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2014, 22(5): 983–994. doi: 10.1109/TVLSI.2013.2263155.

[10] 高嘉浩, 李伟, 陈韬. 基于密码逻辑阵列的分组密码高能效映射方法[J]. 电子技术应用, 2019, 45(11): 21–26, 31.
GAO Jiahao, LI Wei, and CHEN Tao. Block cipher energy efficient mapping method based on cipher logic array[J]. *Application of Electronic Technique*, 2019, 45(11): 21–26, 31.

[11] WU L, WEAVER C, and AUSTIN T. CryptoManiac: A fast flexible architecture for secure communication[C]. The 28th Annual International Symposium on Computer Architecture, Goteborg, Sweden, 2001: 110–119. doi: 10.1109/ISCA.2001.937439.

[12] HUANG Wei, HAN Jun, WANG Shuai, *et al*. A low-complexity heterogeneous multi-core platform for security soc[C]. 2010 IEEE Asian Solid-State Circuits Conference, Beijing, China, 2010: 1–4. doi: 10.1109/ASSCC.2010.5716621.

[13] WEI Li, ZENG Xiaoyang, DAI Zibin, *et al*. A high energy-efficient reconfigurable VLIW symmetric cryptographic processor with loop buffer structure and chain processing mechanism[J]. *Chinese Journal of Electronics*, 2017, 26(6): 1161–1167. doi: 10.1049/cje.2017.06.010.

[14] SAYILAR G and CHIOU D. Cryptoraptor: High throughput reconfigurable cryptographic processor[C]. 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, USA, 2014: 155–161. doi: 10.1109/ICCAD.2014.7001346.

[15] LIU Leibo, WANG Dong, ZHU Min, *et al*. An energy-efficient coarse-grained reconfigurable processing unit for multiple-standard video decoding[J]. *IEEE Transactions on Multimedia*, 2015, 17(10): 1706–1720. doi: 10.1109/TMM.2015.2463735.

[16] LIU Bin and BAAS B M. Parallel AES encryption engines for many-core processor arrays[J]. *IEEE Transactions on Computers*, 2013, 62(3): 536–547. doi: 10.1109/TC.2011.251.

李 伟：男，1983年生，副教授，博士生导师，研究方向为密码处理器设计，ASIC专用芯片设计.

高嘉浩：男，1995年生，硕士生，研究方向为可编程逻辑电路设计.

杜怡然：男，1991年生，讲师，研究方向为SoC与可重构设计，安全专用芯片设计.

陈 韬：男，1979年生，副教授，硕士生导师，研究方向为安全专用芯片设计.

责任编辑：马秀强