计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20160670
54(12):2833-2842,2017

# 基于流体系架构的分组密码处理器设计

李功丽[1,2] 戴紫彬[1] 徐进辉[1] 王寿成[1] 朱玉飞[1] 冯 晓[1]
[1](解放军信息工程大学 郑州 450001)
[2](河南师范大学计算机与信息工程学院 河南新乡 453002)
(ligl522@163.com)

# Design of Block Cipher Processor Based on Stream Architecture

Li Gongli[1,2], Dai Zibin[1], Xu Jinhui[1], Wang Shoucheng[1], Zhu Yufei[1], and Feng Xiao[1]
[1](*PLA Information Engineering University*, *Zhengzhou* 450001)
[2](*College of Computer & Information Engineering*, *Henan Normal University*, *Xinxiang*, *Henan* 453002)

**Abstract** To improve the performance of cipher processor, the performance model of cipher processor is proposed. And based on this model, three ways for enhancing cipher processor's performance are presented, which are sharing multi-level resources, binding operations of pre-xor or post-xor and maximizing parallelism of block cipher algorithms. According to these improvement methods, the type and amount of cryptographic function units are determined. However, the function units are not only numerous but also different in operation data width and latency, so how to organize these function units efficiently becomes a key problem. The stream processor architecture can integrate a large number of function units to obtain high performance. Then, we design and implement a reconfigurable block cipher processor prototype which is based on stream processor architecture, and in order to organize the numerous function units effectively, the function units are divided into basic units, inter-bank-shared units and inter-cluster-shared units respectively according to their processing width. The prototype is synthesized in 65nm CMOS process and several typical block cipher algorithms are mapped on it. The evaluation results show that the processor prototype is area-efficient and its performance is not only beyond that of international application specific instruction cipher processors, but also higher than that of the domestic reconfigurable array processors.

**Key words** block cipher; stream processor; performance model; reconfigurable; cipher processor

**摘 要** 为提升密码处理器性能,构建了密码处理器性能模型.基于该模型,提出多级资源共享、绑定前/后异或操作、最大化算法并行度等处理器性能提升技术,并根据性能提升技术确定了功能单元的种类和数量.然而功能单元不仅数量较多,而且在操作位宽和操作延迟方面均有较大差异,如何有效组织这些功能单元成为了一个关键问题.利用流体系结构可以高效集成大量功能单元的特点,设计并实现了基于流体系结构的可重构分组密码处理器原型,并通过把功能单元划分为基本处理单元,bank 间共享单元和簇间共享单元 3 个层次来解决功能单元处理位宽和操作延迟的差异.在 65 nm CMOS 工艺下对处理器原型进行综合,并在该结构上映射了典型的分组密码算法.实验结果证明:该处理器以较小的面积获得了较高的性能,对典型分组密码算法的处理速度,不仅超越了国际上的密码专用指令处理器,而且高于国内可重构阵列结构密码处理器.

**收稿日期**:2016-08-31;**修回日期**:2016-12-09
**基金项目**:国家自然科学基金项目(61404175)
This work was supported by the National Natural Science Foundation of China (61404175).

**关键词** 分组密码;流处理器;性能模型;可重构;密码处理器

**中图法分类号** TP309.7

最近几十年,信息安全已经逐步深入到政治、经济、军事以及人们日常生活的方方面面,并在上述领域发挥着越来越重要的作用.分组密码是保证信息安全的有效措施.随着信息安全需求的与日俱增,对分组密码处理的要求也越来越高,吞吐率一直是密码处理中首要考虑的性能指标.而随着密码应用场景的多样化和复杂化,还要求密码系统能够灵活适应不同算法,以应对不断发展的攻击手段和密码系统升级压力,所以密码系统的灵活性也变得越来越重要.上述这些要求都给密码处理器的设计提出了新的挑战.

可重构分组密码处理器面向分组密码处理进行优化,运算单元可重构使密码处理具有了较高的灵活性[1].同时,通过重复设置大量的功能单元来提高性能,但是功能单元种类的选择和功能单元数量的确定缺乏有效的理论分析作为基础,所以在性能提升的同时,往往导致面积和功耗的急剧增加,这使得开发出来的密码处理器因为面积和功耗的问题,在实际应用时受到诸多限制.本文以 Amdahl 定律[2]为基础,结合分组密码处理的特点,首先构建了密码处理性能模型,然后通过该性能模型确定功能单元种类及数量,从而在有效提升密码处理性能的同时,有效控制处理器的面积.

## 1 相关研究

国内外针对可重构分组密码处理器的实现技术展开了一系列研究,主要可归纳为可重构分组密码专用指令处理器结构和可重构分组密码阵列处理器结构.

可重构分组密码专用指令处理器的指令及运算单元面向密码应用进行优化,运算单元可重构,具有数据处理位宽大、处理并行性高、控制简单、开发便捷等优点.孟涛和戴紫彬[3]提出的可重构分簇分组式分组密码处理架构(reconfigurable clustered block cipher processor, RCBCP),可灵活地重构为 4 个 32 b,2 个 64 b 和 1 个 128 b 的数据通路,设计并实现了 5 级流水线以及运算单元内流水结构.Gaspar 等人[4]提出的密码处理器 HCrypt 设计了分离的数据/密钥路径,集成了独立的加密模块和解密模块.Li 等人[5]基于 VLIW 指令结构开发了 4 路 32 b 并行结构来提高可重构密码处理器的性能.相比于其他实现模式,专用指令处理器结构工作频率较低、密码处理性能有待进一步提升.

阵列结构可重构分组密码处理系统设计了面向分组密码的可重构运算单元,大幅提升了密码处理的性能.杨晓辉等人[6]提出的可重构分组密码处理模型(reconfigurable cipher processing architecture, RCPA),采用粗粒度可重构密码单元,能够在横向和纵向上开发分组密码并行特征.陈韬等人[7]提出了一种基于流处理架构的可重构分组密码阵列结构(stream based reconfigurable clustered block cipher processing array, S-RCCPA),通过设计基于 Crossbar 的分级互连网络连接多个粗粒度可重构的功能单元,构造了分簇式的密码处理阵列.Sayilar 和 Chiou[8]提出的可重构密码处理结构 Cryptoraptor,通过集成 80 个基本重构单元实现了较高的性能.阵列结构的单元一般为同构结构,为确保处理单元能够广泛支持分组密码操作,处理单元设计比较复杂,面积和功耗较大,存在资源利用率低、用户开发困难等问题.

流处理器[9-10]作为近年来兴起的一种高性能处理器,主要针对计算密集型应用,通过集成大量的功能单元提供丰富的计算能力,从而获得了较高的性能.如 Imagine 流处理器在 250 MHz 下的峰值性能达到 10GFLOPS,而 Merrimac 在 1 GHz 下的峰值性能达到 128GFLOPS,且在实际应用中达到了 117.3GFLOPS.流处理器以其对计算密集型应用所显示出的强大性能优势而成为了当前高性能计算领域的研究热点.

分组密码处理具有典型的计算密集性特征[11],是适合于流体系结构的应用.所以本文首先基于 Amdahl 定律构建了密码处理器的性能模型,然后在该模型的指导下,确定了密码处理单元的种类和数量,并基于流体系结构有效组织功能单元,设计了面向分组密码的可重构密码流处理器原型,最后通过综合实验验证了该原型的性能及面积优势.

## 2 基于 Amdahl 定律的密码处理性能模型

根据 Amdahl 定律,密码算法可以分为串行部分和并行部分,分别设为 $N_s$ 和 $N_p$.对于并行度为 $i$

的任务,使用 1 个运算单元串行处理所需时间为 $t_p$,则采用 $i$ 个相同的处理单元并行加速后执行时间为 $t_p/i$. 考虑到密码算法并行部分的并行度并不是固定不变的,设密码算法最大并行度为 $M$,并行度为 $i$ 的任务串行执行时间为 $t_{pi}(1\leqslant i\leqslant M)$. 设处理器可实现的最大并行度为 $I$,则对并行部分加速后算法的执行时间 $t'_p$ 为

$$t'_p = \sum_{i=1}^{M}\left(\frac{t_{pi}}{i}\times\left\lceil\frac{i}{I}\right\rceil\right). \tag{1}$$

从式(1)可以得出,当 $I\geqslant i$ 时,无论 $I$ 如何增加,都无法再进一步减少执行时间. 即通过设置更多的硬件资源所能提高的并行度受限于算法本身的并行性.

令 $t_{pi}$ 对应的指令条数为 $N_{pi}$,则并行加速后所需指令条数为

$$N'_p = \sum_{i=1}^{M}\left(\frac{N_{pi}}{i}\times\left\lceil\frac{i}{I}\right\rceil\right). \tag{2}$$

而对于串行部分,假设完成串行部分的指令共为 $N_s$ 条,分别记为 $n_1,n_2,\cdots,n_{N_s}$. 如果指令 $n_i$,$n_{i+1},\cdots,n_{i+k}$ 可以在 1 个时钟周期内完成,则原指令可合并为 1 条新指令,记作 $n_{ik}$. 经过合并后完成串行部分所需指令条数变为

$$N'_s = \sum_{i=1}^{N_s}\sum_{k=i}^{N_s}(p(i,k)\times n_{ik}), \tag{3}$$

其中,$p(i,k)=\begin{cases}0, & n_{ik}\text{不存在.}\\ 1, & n_{ik}\text{存在.}\end{cases}$

串行部分减少的指令条数为

$$N_d = \sum_{i=1}^{N_s}\sum_{k=i}^{N_s}(p(i,k)\times(k-i)). \tag{4}$$

通过对并行性部分的并行加速和串行部分的合并加速后,处理器完成密码运算所需时间为

$$T' = \frac{CPI\times\left((N_s-N_d)+\sum_{i=1}^{M}\left(\frac{N_{pi}}{i}\times\left\lceil\frac{i}{I}\right\rceil\right)\right)}{f_{clk}}, \tag{5}$$

其中,$CPI$ 为执行一条指令的平均时钟周期数,$f_{clk}$ 为系统主频(单位为 MHz). 此时,假设待处理的数据长度为 $L$(单位为 b),则吞吐率 $P$ 为

$$P = \frac{L\times f_{clk}}{CPI\times\left((N_s-N_d)+\sum_{i=1}^{M}\left(\frac{N_{pi}}{i}\times\left\lceil\frac{i}{I}\right\rceil\right)\right)}. \tag{6}$$

从式(6)可知,为了提高系统的吞吐率 $P$,可通过提高系统频率 $f_{clk}$、减少指令的平均执行周期 $CPI$、增加合并的串行指令数目 $N_d$ 以及提供足够的并行处理资源 $I$ 这 4 种途径提升密码处理器性能.

## 3 密码处理性能提升策略

下面将结合分组密码结构特征及基本单元的运算特征,分别针对 $f_{clk}$,$CPI$,$N_d$,$I$ 这 4 个参数来讨论密码处理器性能提升策略.

### 3.1 $f_{clk}$ 与 $CPI$——多级资源共享机制

根据前面的分析可知,提高系统频率 $f_{clk}$ 能够显著提升密码处理器性能,而频率取决于系统的关键路径.

首先选取国际上典型的分组密码算法作为样本进行研究. 算法包括数据加密标准 DES、高级加密标准 AES, ISO/IEC 分组密码标准(Camellia, MISTY1,CAST-128,TDEA,SEED)、欧洲 NESSIE 计划的分组密码、日本 CRYPYTEC 计划的分组密码、韩国标准 ARIA 以及行业标准 IDEA,SMS4, CLEFIA,FOX,Skipjack 等算法. 选取的算法均有比较广泛的应用,结构也具有一定的代表性,基本可以代表当前分组密码的设计特点.

通过对所选 21 种算法的分解发现,首先分组密码算法整体结构主要有 FEISTEL,SP,L-M 等,基于相同或相似结构的分组密码具有相似的运算单元,涉及的操作类型有较大的交集,可归纳为 S 盒替代、移位、有限域乘法、模乘、模加/减、比特置换和基本逻辑等;其次操作的数据位宽集中于 2 类:

1) 32 b 以内小位宽操作,特点是位宽小、并行度高;

2) 128 b 以上大位宽操作,多是异或、置换和移位操作.

为了确定系统的关键路径,对上述算法集合中使用的运算进行统计,并采用 65 nm CMOS 工艺标准单元库,对 8 b,16 b,32 b,64 b,128 b 等不同操作位宽的基本密码运算单元进行逻辑综合,结果如表 1 所示.

由综合结果可知,不同运算单元的延时差异较大,运算单元的选取将会直接影响系统频率.

同时,影响系统性能的另一个因子——指令的平均执行周期 $CPI$:

$$CPI = \sum_{i=1}^{n}(CPI_i\times p_i), \tag{7}$$

其中,$CPI_i$,$p_i$ 分别为第 $i$ 类指令的执行周期数和使用频率. 由式(7)可知,指令的使用频率对系统 $CPI$ 有较大影响. 假设系统使用频率最高的是指令 $i$,当 $p_j\ll p_i$ 时,即使 $CPI_j$ 为多个时钟周期,对 $CPI$ 的

**Table 1 Synthesization Results of Cryptographic Units (Frequency & Area)**

**表 1 密码运算单元综合结果(频率及面积)**

| Unit | Delay Time/ns | Usage | Area/$\mu m^2$ | Unit | Delay Time/ns | Usage | Area/$\mu m^2$ |
|---|---|---|---|---|---|---|---|
| S box(8 b-8 b) | 0.553 | High | 11217.9 | GF(32 b) | 1.22 | High | 18090 |
| xor(8 b) | 0.04 | High | 23 | shift(8 b) | 0.39 | Low | 128 |
| logic(8 b) | 0.22 | High | 143.52 | shift(16 b) | 0.52 | Low | 394 |
| mod add/sub(8 b) | 0.31 | Low | 113 | shift(32 b) | 0.61 | High | 998 |
| mod add/sub(16 b) | 0.42 | Medium | 212 | shift(64 b) | 0.7 | Medium | 2424 |
| mod add/sub(32 b) | 0.5 | High | 399.7 | shift(128 b) | 0.79 | High | 5575 |
| mod add/sub(64 b) | 0.95 | Medium | 7582 | bit perm(8 b) | 0.78 | Low | 256 |
| mod mult(8 b) | 0.98 | Low | 349 | bit perm(16 b) | 1.04 | Low | 788 |
| mod mult(16 b) | 1.25 | Low | 1538.4 | bit perm(32 b) | 1.22 | Medium | 1996 |
| mod mult(32 b) | 1.41 | High | 6765 | bit perm(64 b) | 1.4 | Medium | 4848 |
| arithemetic mult(8 b) | 1.28 | Low | 734.4 | bit perm(128 b) | 1.58 | High | 11150 |
| arithemetic mult(16 b) | 1.3 | Low | 3713.3 | byte perm(128 b) | 1.08 | Medium | 6304 |
| arithemetic mult(32 b) | 1.43 | Medium | 18051.8 | byte mult(32 b) | 1.22 | Low | 747 |

影响也十分有限,所以选择功能单元的目标是尽量减少使用频率高的指令的执行周期,从而降低 *CPI*.

根据上述分析,为使用频率高的指令设置专用的功能单元,使指令可以在 1 周期或 2 周期内完成,而对于使用频率较低的指令,综合考虑其重构代价和组合延时由这些基本单元重构或组合实现.

通过表 1 可知,使用频率高的 8 b-8 b 的 S 盒、8 b 逻辑操作、32 b 模加/减等延时都低于 1 ns,不会影响处理器关键路径.而使用频率高的 32 b 有限域乘法、32 b 模乘、128 b 置换等单元的延时较大.集成这些大延时的单元会使系统频率降为原来的 1/2 或 1/3,若不集成这些单元,一旦算法中包括这类操作,往往需要几个甚至十几个操作才能组合完成,从而使算法性能急剧变差.所以综合考虑系统频率及 *CPI*,根据单元延时设置这些大延时的单元为 2 周期或是 3 周期,从而避免大延时的单元降低系统频率.

进一步考虑不同位宽单元的重构开销,确定构成系统的运算单元包括 8 b-8 b 的 S 盒、8 b 逻辑单元和模加/减单元,32 b 的模乘单元和有限域乘法单元以及 128 b 的移位和比特置换单元.8 b 的模加/减单元虽然使用频率不高,但是通过设置进位链[12]可以并行实现 16 b 或 32 b 的模加/减操作,所以把 8 b 的模加/减单元作为基本运算单元.因为选择的运算单元位宽变化较大,所以如何有效地组织这些单元,既能使各个单元有效配合,高效实现密码算法,又能使单元之间的数据通路易于实现,是下一步要解决的关键问题.

首先选择 8 b-8 b 的 S 盒替代、逻辑运算和模加/减构成一个可重构密码处理 bank(reconfigurable cipher processing bank, RCPB)作为系统的基本运算核心,RCPB 的基本数据位宽为 8 b.因为密码操作中,以 8 b 和 32 b 操作居多,所以把 4 个 RCPB 组成一个可重构密码簇(reconfigurable cipher processing cluster, RCPC).同一个簇中各 RCPB 的功能单元不是独立的,它们可以重构或组合实现 16 b 或 32 b 的操作.

有限域乘法和模乘单元均为 32 b,组成 bank 间共享单元(inter-bank-shared unit, BSU),一个 BSU 对应 4 个 RCPB.128 b 的移位和置换组成簇间共享单元(inter-cluster-shared unit, CSU),每个 CSU 对应 4 个簇,可以同时与 4 个簇进行数据交互.

通过把使用频率高的指令用专用单元实现,使指令可以在 1 周期或 2 周期完成,从而有效降低 *CPI*;把延时较长的指令设置为多周期,可以避免大延时单元降低系统频率;然后根据单元的位宽和重构效率等,把 8 b 位宽的运算单元组织成基本运算核心,大位宽的单元由多个基本核心共享.这种小位宽的运算核心加大位宽的共享资源,既能够让多个小位宽单元并行执行,又可以减少大位宽运算单元的数量,从而在提升处理器性能的同时,提高资源利用率并减少处理器面积.

### 3.2 $N_d$——绑定前/后异或

异或操作是分组密码中使用频率最高的运算,而且异或操作的延时和面积都非常小,通过将异或操作与其他密码操作组合实现复合操作,可以有效减少算法所需指令条数.以 SMS4 算法为例,在无绑

定操作情况下,轮函数需使用 13 条指令,在支持指令绑定前/后异或的情况下,轮函数仅需使用 7 条指令,从而使指令条数减少了 46.15%.通过对算法集合中的运算进行分析统计,异或运算绑定使用频率如表 2 所示:

**Table 2 Usage of Binding Xor Operation**

**表 2 绑定异或操作使用频率**

| Operation | Usage/% | Delay Time/ns |
|---|---|---|
| xor-S box | 77.59 | 0.6 |
| S box-xor | 76.10 | 0.6 |
| xor-shift | 56.40 | 0.83 |
| shift-xor | 50.87 | 0.83 |
| mod add/sub-xor | 49.63 | 0.35 |
| Xor-mod add/sub | 47.20 | 0.35 |
| logic with three inputs | 50.97 | 0.45 |

绑定有 2 种实现方式:

1) 利用执行过程中流水线的译码和写回段延时较小,把异或操作绑定在寄存器取数和写回段,这种方法不会影响系统的工作频率,但是会产生严重的数据相关问题,特别是无法实现数据旁路,而在密码运算中有大量的中间运算结果需要旁路,所以使用该方法会造成严重的流水线停顿;

2) 直接把密码操作与异或绑定,但是这可能会影响系统的关键路径,从而影响频率.通过分析异或操作与其他操作的使用频率及延时,选择使用频率高且对关键路径不影响或是影响较小的操作进行绑定.根据表 2 的统计结果,确定绑定异或的功能单元有 S 盒前后绑定、逻辑后绑定、模加后绑定、移位后绑定.

上述 4 个操作绑定后,延时均未超过关键路径.绑定异或操作前后,算法轮函数所需要的指令条数如表 3 所示:

**Table 3 Number of Round Function Instruction Pre-binding and Post-binding**

**表 3 绑定操作前后轮函数的指令条数**

| Algorithm | Instruction Number | |
|---|---|---|
| | Pre-binding | Post-binding |
| DES | 5 | 4 |
| AES | 4 | 3 |
| CAST-128* | 7 | 6/5 |
| Camellia | 8 | 7 |
| SEED | 30 | 24 |
| SHACAL2 | 9 | 7 |
| SC2000 | 29 | 21 |

**Continued(Table 3)**

| Algorithm | Instruction Number | |
|---|---|---|
| | Pre-binding | Post-binding |
| IDEA | 9 | 9 |
| CLEFIA | 5 | 4 |
| SkipJack* | 14/12 | 5/4 |
| SAFER++ | 19 | 18 |
| Sperpent | 10 | 9 |
| MARS | 88 | 78 |
| Twofish | 8 | 5 |
| FOX-64 | 10 | 7 |
| RC6 | 11 | 10 |
| MISTY1 | 38 | 36 |
| SMS4 | 13 | 7 |
| FOX-128 | 12 | 9 |
| KASUMI | 21 | 13 |
| ARIA | 9 | 8 |

Note: * has more than one round functions

在绑定异或操作后,95%以上的密码算法轮函数所需要的指令条数均有所减少,平均减少了 19.3%.但是也有算法在绑定后,指令条数并未减少,如 IDEA.这是因为 IDEA 算法中的异或操作出现在延时较大的模乘操作之后,为了不影响关键路径,未对模乘操作绑定异或.

### 3.3 *I*——算法并行度分析

通过多路并行执行能够提升密码算法实现性能.但是对于密码算法来说,可实现的最大并行度受限于算法本身的并行性.基于 3.1 节所确定的运算单元,对算法集合中各算法的分组内最大并行度 $i$ 进行分析,结果如表 4 所示:

**Table 4 Max Parallelism in Block**

**表 4 分组内最大并行度**

| Algorithm | Parallelism | Algorithm | Parallelism |
|---|---|---|---|
| DES | 8 | Sperpent | 16 |
| AES | 16 | MARS | 16 |
| CAST-128 | 4 | Twofish | 8 |
| Camellia | 8 | FOX-64 | 4 |
| SEED | 8 | RC6 | 16 |
| SHACAL2 | 16 | MISTY1 | 4 |
| SC2000 | 8 | SMS4 | 4 |
| IDEA | 4 | FOX-128 | 8 |
| CLEFIA | 16 | KASUMI | 8 |
| Skipjack | 4 | ARIA | 16 |
| SAFER++ | 16 | | |

通过表 4 可以发现,算法集合中的密码算法分

组内最大并行度为 16,且所占比例较高(38.1%),所以系统集成 16 个 RCPB 可以最大限度地开发大部分算法的分组内并行性,使式(7)中的 $\left\lceil \frac{i}{I} \right\rceil$ 因子达到最小值 1.

以 AES 算法为例,16 个 RCPB 可以实现 AES 分组内最大并行. 同时,在 ECB 模式下,这 16 个 RCPB 还可以实现多个分组的纵向流水执行. 因为有限域乘法均需要 2 个周期,成为了流水线的瓶颈,所以把有限域乘法单元设置为 2 段流水,从而使 AES 算法在 ECB 模式下可以流水执行 4 个分组.

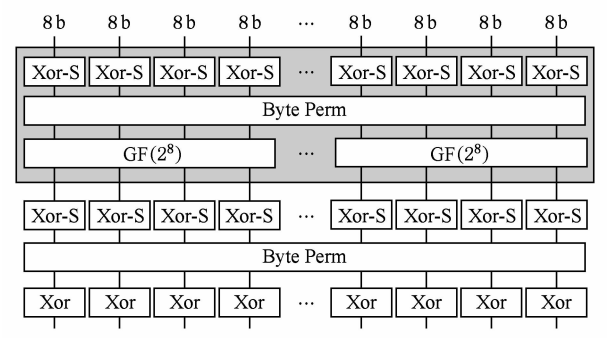


Fig. 1 AES mapping result

图 1 AES 算法映射图

而对于并行度小于 16 的密码算法,在 ECB 模式下或是 CBC 模式下通过交错技术[13],可以在多个 RCPB 上运行不同的分组,实现多个分组的横向并行. 例如 SMS4 算法,执行 1 个 128 b 的分组需要 4 个 RCPB,16 个 8 b 的数据在 4 个 RCPB 上的映射结果如图 2 所示. 如果使用全部 16 个 RCPB,可以实现 4 个分组的并行执行.

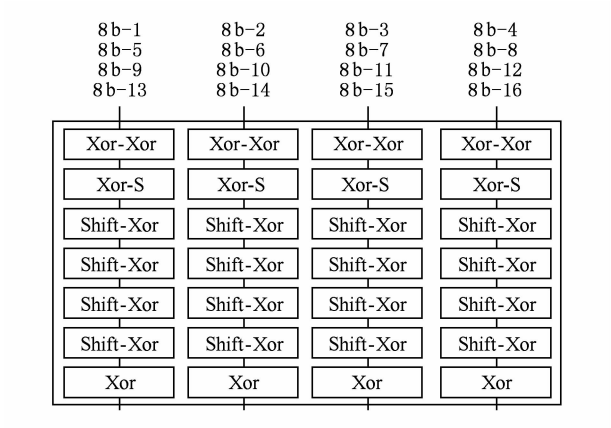


Fig. 2 SMS4 mapping result

图 2 SMS4 算法映射图

多个 RCPB 既可以通过流水线技术实现多个分组的纵向并行,同时对于并行度小于 16 的算法,还可以实现多个分组的横向并行. 2 个维度上的并行性开发可以大大提高密码处理器的吞吐率以及资源利用率.

## 4 可重构分组密码流处理器原型

为了提高系统的时钟频率和减少指令的平均执行周期,选择使用频率高的功能单元构成系统的运算部件,并根据其位宽分别组成基本运算核心 RCPB 和多级共享单元 BSU 及 CSU;为了减少算法的指令条数,在不影响关键路径的前提下绑定异或操作;通过对典型算法的并行性分析发现,设置 16 个 RCPB 时,可以最大限度地实现密码算法的分组内并行. 基于上述分析,以流处理器框架为基础,设计了可重构

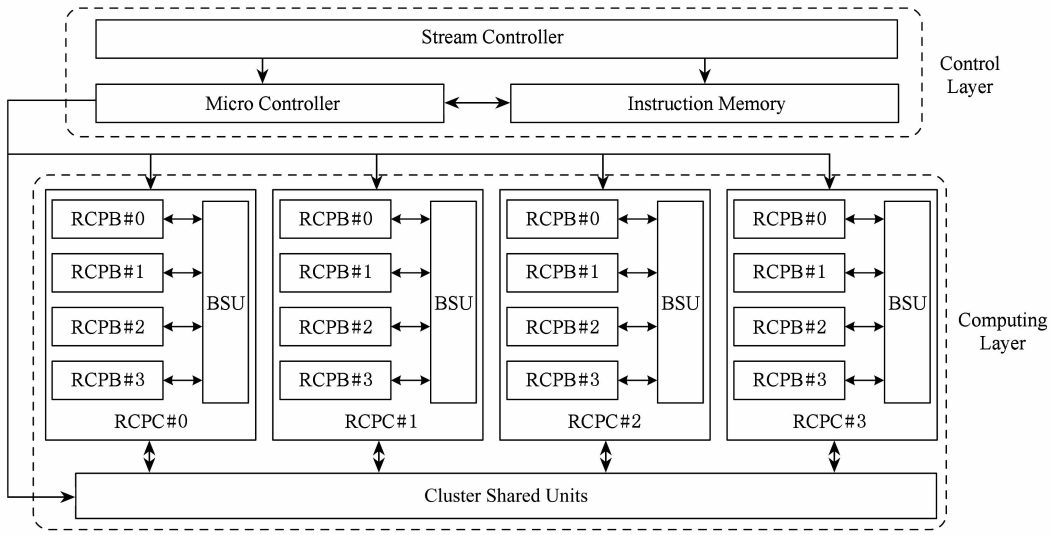


Fig. 3 General architecture of RCSP

图 3 RCSP 结构图

密码流处理器(reconfigurable cipher stream processor, RCSP),结构如图 3 所示.

整个 RCSP 的结构分为控制层和运算层 2 个部分. 控制层由流控制器和微控制器两级控制组成. 当需要执行密码算法时,首先由流控制器加载算法指令和数据到 RCSP 中,指令存储在指令存储器中,指令存储器的容量为 512×128 b,数据通过 IO 接口送入到与每个功能单元直接相连的局部寄存器(local register file, LRF)中. 指令执行时由微控制器读取指令和译码,并把指令信息广播到对应的功能单元. 同时,微控制器还负责执行循环跳转等控制指令. RCSP 采用 VLIW 指令结构,所以它的控制部分相对简单,从而可以把更多的片上面积用于运算部件.

RCSP 的运算层由基本运算核心 RCPB,bank 间共享单元 BSU 和簇间共享单元 CSU 组成. 处于不同层次中的功能单元之间的连接方式如图 4 所示,其中白色表示的是 RCPB 内部的功能单元,浅灰色表示的是 BSU 单元,深灰色表示的是 CSU 单元.

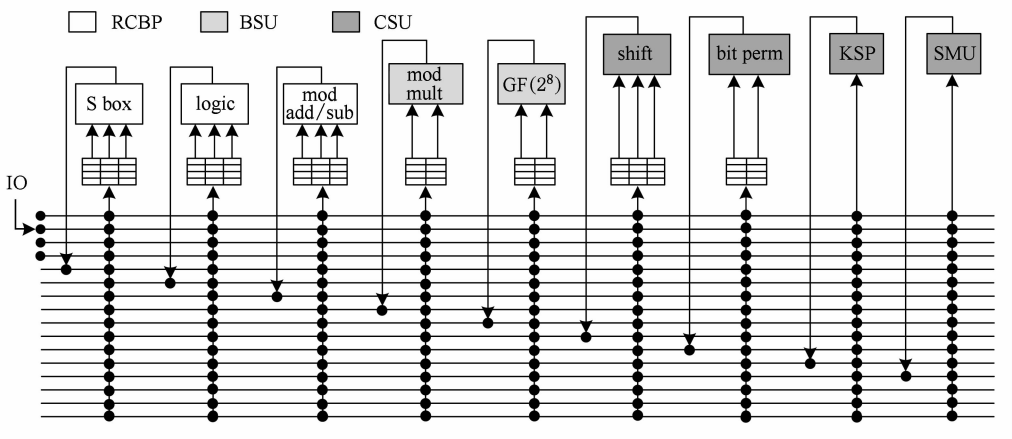


Fig. 4 Architecture of operation layer

图 4 运算层结构图

为了实现各个功能单元之间的快速数据交换,通过交叉互联开关把各个功能单元连接起来. 每个功能单元在一个 RCPB 中都对应一个 8 b 位宽的结果输出总线和若干个输入寄存器. 输入寄存器中包括了绑定操作时前/后绑定的操作数.

从逻辑结构上看,因为 BSU 和 CSU 中的共享单元在每个 RCPB 内通过交叉互连网络与 RCPB 内的功能单元连接,相当于共享单元在每个 RCPB 中均有一个 8 b 位宽的功能单元,是独立属于每个 RCPB 的,即每个 RCPB 中均有 7 个 8 b 位宽的功能单元,它们之间的数据交换通过交叉互连网络在单周期内实现. RCPB 与共享单元这种逻辑结构上相互独立的设计,可以使密码算法映射变得简单.

因为密码算法有大量的中间运算结果,需要频繁进行中间结果的写回与读取操作,所以中间结果的写回与读取时间对系统的执行速度有较大影响. 设置与每个功能单元直接相连的 LRF 保存中间结果,每次运算单元所需要的操作数都是直接从与它对应的 LRF 中取出,当操作完成,结果通过交叉互连网络写入下一个要使用该结果的功能单元所对应的 LRF 中. 同时,LRF 支持写穿透[9]. 中间结果保存到寄存器的同时传送给需要该结果的功能单元,避免因为数据写后读相关造成的流水线停顿. 因为功能单元都是从自己对应的 LRF 中取操作数,相比于集中式的寄存器,这种分布式的结构可以有效减少操作延迟和寄存器面积.

密钥寄存器(key scratch pad, KSP)设置为 32×128 b,在每个 bank 中对应的位宽是 8 b,即每个 bank 一次可以向 KSP 中读/写一个 8 b 的数据.

每个 RCPB 中都有一个共享存储单元(shared memory unit, SMU),各个 bank 的 SMU 通过字节置换单元连接,实现单周期内多个 RCPB 之间的数据交换.

## 5 实验及性能对比

使用 Verilog 语言对 RCSP 进行描述后,通过 65 nm CMOS 工艺标准单元库及相应负载模型对设计的处理器原型进行逻辑综合,综合结果如表 5 所示. 其中 Slack 为正值表示实际延迟时间满足约束条件,且其值越大表示实际时钟频率可以综合得更高;而 Slack 为负值表示实际延迟时间超过了约束条件,DC 综合工具已经无法再继续优化. 根据综合

结果,RCSP 时钟频率最高可达 826 MHz,为保证工作稳定性,将 RCSP 工作频率确定为 800 MHz. 后文讨论均为 RCSP 在 800 MHz 工作频率下的运算性能.

**Table 5 Result Based on ASIC Implementation**

**表 5 基于 ASIC 实现的结果**

| Delay Time/ns | Area/$\mu m^2$ | Slack/ns | Frequency/MHz |
|---|---|---|---|
| 1.25 | 1893731.5 | 0.02 | 800 |
| 1.23 | 1893731.5 | 0.01 | 813 |
| 1.22 | 1914053.4 | 0 | 820 |
| 1.21 | 1936873.7 | −0.08 | 826 |

对选取的算法集合进行适配,结果表明 RCSP 可以高效处理 SP 结构、Feistel 结构、LM 结构的分组密码算法. 由于目前密码处理器文献中,一般仅列出 AES,DES,IDEA 的实现结果,考虑到 MISTY 结构算法具有一定代表性,增加了 MISTY 结构的代表算法 KUSAMI. RCSP 与其他密码处理器性能对比情况如表 6 所示. 其中,Cryptonite[14]、CCProc[15]、RCBCP[3]、文献[5]和 MCCP[16]为可重构密码专用指令处理器;Celator[17],RCPA[6],S-RCCPA[7],Cryptorapto[8],RCPC[18]为阵列结构可重构密码处理器.

**Table 6 Performance Comparison with Other Cryptographic Processors**

**表 6 不同密码处理器性能对比**

| Processor | Process/nm | Area/$mm^2$ | Frequency/MHz | Throughput/Gbps | Algorithm | Reconfigurable |
|---|---|---|---|---|---|---|
| Cryptonite | XCV2P30 | | 400 | 0.73<br>0.73<br>0.57 | AES<br>DES<br>IDEA | N |
| CCProc | 130 | 5.3 | 250 | 0.41 | AES | N |
| MCCP | XCV4SX35 | | 192 | 0.85/1.90 | AES | Y |
| Celator | 130 | 0.1 | 190 | 0.05<br>0.03 | AES<br>DES | Y |
| RCBCP | 180 | | 312.5 | 0.69<br>0.4<br>0.42 | AES<br>DES<br>IDEA | Y |
| RCPA | 180 | 14.9 | 180 | 0.79<br>0.46<br>0.4 | AES<br>DES<br>IDEA | Y |
| S-RCCPA | 180 | 13.4 | 243.9 | 2.8<br>0.9<br>1.6 | AES<br>DES<br>IDEA | Y |
| Cryptoraptor | 45 | 6.32 | 1000 | 6.4/128<br>2.67/42.67<br>1/16 | AES<br>DES<br>KASUMI | Y |
| RCPC | 65 | 4.28 | 400 | 2.16<br>2.72<br>6.3 | AES<br>DES<br>SMS4 | Y |
| Literature 5 | 180 | 18 | 350 | 1.55<br>0.45<br>0.79 | AES<br>DES<br>IDEA | Y |
| RCSP | 65 | 1.89 | 800 | 2.63/10.24<br>0.52/1.7<br>0.52/3.76<br>0.49/0.98<br>0.46/1.83 | AES<br>DES<br>IDEA<br>KASUMI<br>SMS4 | Y |

根据综合结果,RCSP 以较小的面积代价获得了较高的性能. RCSP 对 AES 的实现性能在 CBC 和 ECB 模式下分别达到了 2.63 Gbps 和 10.24 Gbps. 而对于 DES 和 IDEA,在 CBC 模式下吞吐率均为 0.52 Gbps,在 ECB 模式下,通过纵向流水和横向并行,吞吐率分别达到了 1.7 Gbps 和 3.76 Gbps. 通过性能对比发现,RCSP 对密码算法的实现性能明显高于其他专用指令处理器,大大提升了可重构密码处理器的实现性能.

与阵列结构的可重构密码处理器比较,RCSP 实现性能仍高于 Celator,RCPA,S-RCCPA 处理架构. 由此可见,通过设置 8 b 的基本处理单元和多级共享单元、绑定前/后异或以及多维度开发并行化等性能提升策略,可重构分组密码流处理器 RCSP 获得了与当前阵列结构可重构密码处理器相当的密码处理性能.

考虑到资源消耗及单元使用频率,RCSP 中未设计大位宽查找表单元,对于 9-9 的 S 盒操作需使用多条指令组合实现. 由于 KASUMI 结构使用了大量的 9-9 的 S 盒,RCSP 对 KASUMI 算法实现性能较低. Cryptoraptor 设计了 10-32 的查找表,对 AES,DES,KASUMI 的实现性能均高于 RCSP,但 Cryptoraptor 结构未设计乘法单元,对使用范围较广的 IDEA 算法支持效果差,Cryptoraptor 作者在性能对比中回避了 IDEA 算法,因此无法作出比较,同时 Cryptoraptor 资源消耗大约是 RCSP 的 5 倍(考虑到工艺差距),且 Cryptoraptor 是阵列结构,应用开发难度较大.

## 6 结束语

设计高效、灵活的可重构密码处理器是信息安全领域的热点研究课题. 密码专用指令处理器具有配置信息少、灵活便捷的优点,但处理性能不高,成为限制其进一步发展的主要因素.

针对上述问题,根据分组密码算法特征,以提高分组密码处理器性能为目标,构建了分组密码处理器性能模型,并基于该模型研究密码处理器性能提升技术. 提出了通过提升系统工作频率和减少指令平均周期的多级资源共享技术、提升单周期密码操作数目的绑定前/后异或技术、开发分组密码并行特征的并行处理技术. 利用流处理器可以集成大量功能单元、控制结构简单的特点,在性能提升技术的指导下,设计并实现了面向分组密码的可重构密码流处理器原型 RCSP. 在 65 nm CMOS 工艺下完成了 RCSP 的逻辑综合,通过典型分组密码算法的映射结果表明,RCSP 的密码实现性能均高于其他专用指令处理器,而且高于 Celator,RCPA,S-RCCPA 等阵列结构可重构密码处理结构,具有很好的应用场景.

由于缺少大位宽 S 盒运算单元,RCSP 对 MSITY,KUSAMI 等算法的实现性能较低,为提升密码算法实现性能,下一步可考虑集成 9-9 的查找表.

## 参 考 文 献


[1] Compton K, Hauck S. Reconfigurable computing: A survey of systems and software [J]. ACM Computing Surveys, 2002, 34(2): 171-210

[2] Chen Shuming, Chen Shenggang, Yin Yaming. Revisiting Amdahl's law in the hierarchical chip multicore processors [J]. Journal of Computer Research and Development, 2012, 49(1): 83-92 (in Chinese)
(陈书明, 陈胜刚, 尹亚明. Amdahl 定律在层次化片上多核处理器中的扩展[J]. 计算机研究与发展, 2012, 49(1): 83-92)

[3] Meng Tao, Dai Zibin. Reconfigurable clustered architecture of block cipher processor [J]. Journal of Electronics & Information Technology, 2009, 31(2): 453-456 (in Chinese)
(孟涛, 戴紫彬. 分组密码处理器的可重构分簇式架构[J]. 电子与信息学报, 2009, 31(2): 453-456)

[4] Gaspar L, Fisher V, Bernard F, et al. HCrypt: A novel concept of crypto-processor with secured key management [C] //Proc of Int Conf on Reconfigurable Computing and FPGAs(RECONFIG 2010). Piscataway, NJ: IEEE, 2010: 280-285

[5] 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

[6] Yang Xiaohui, Dai Zibin, Zhang Yongfu. Research and design of reconfigurable computing targeted at block cipher processing [J]. Journal of Computer Research and Development, 2009, 46(6): 962-967 (in Chinese)
(杨晓辉, 戴紫彬, 张永福. 可重构分组密码处理结构模型研究与设计[J]. 计算机研究与发展, 2009, 46(6): 962-967)

[7] Chen Tao, Luo Xingguo, Li Xiaonan, et al. An architecture of stream based reconfigurable clustered block cipher processing array [J]. Journal of Electronics & Information Technology, 2014, 36(12): 3027-3034 (in Chinese)
(陈韬, 罗兴国, 李校南, 等. 一种基于流处理框架的可重构分簇式分组密码处理结构模型[J]. 电子与信息学报, 2014, 36(12): 3027-3034)

[8] Sayilar G, Chiou D. Cryptoraptor: High throughput reconfigurable cryptographic processor [C] //Proc of IEEE/ACM Int Conf on Computer-Aided Design (ICCAD 2014). Piscataway, NJ: IEEE, 2014: 155-161

[9] Rixner M. Stream Processor Architecture [M]. Amsterdam, Netherlands: Kluwer Academic Publishers, 2001

[10] Zhang Chunyuan, Wen Mei, Wu Nan, et al. Research and Design of Stream Processor [M]. Beijing: Publishing House of Electronics Industry, 2009 (in Chinese)
(张春元, 文梅, 伍楠, 等. 流处理器研究与设计[M]. 北京: 电子工业出版社, 2009)

[11] Wang Yuliang. Research and design on coarse-grain reconfigurable architecture for cryptographic algorithms [D]. Zhengzhou: PLA Information Engineering University (in Chinese)
(王玉良. 面向密码算法的粗粒度可重构结构研究与设计[D]. 郑州: 解放军信息工程大学, 2010)

[12] Jarvinen K, Dimitrov V, Azarderakhsh R. A generalization of addition chains and fast inversions in binary fields [J]. IEEE Trans on Computers, 2015, 64(9): 2421-2432

[13] Desai A, Ankalgi K, Yamanur H, et al. Parallelization of AES algorithm for disk encryption using CBC and ICBC modes [C] //Proc of the 4th Int Conf on Computer Communication and Networking Technology. Piscataway, NJ: IEEE, 2013: 1-7

[14] Buchty R. Cryptonite—a programmable crypto processor architecture for high-bandwidth applications [D]. München, Bavaria, Germany: Technische University München, 2002

[15] Theodoropoulos D, Papaefstathiou I, Pnevmatikatos D N. CCproc: An efficient cryptographic coprocessor [C] //Proc of the 16th IFIP/IEEE Int Conf on Very Large Scale Integration (VLSI'08). Berlin: Springer, 2008: 160-163

[16] Bossuet L, Grand M, Gaspar L, et al. Architectures of flexible symmetric key crypto engines-a survey: From hardware coprocessor to multi-crypto-processor system on chip [J]. ACM Computing Surveys, 2013, 45(4): 115-123

[17] Fronte D, Perez A, Payrat E. Celator: A multi-algorithm cryptographic co-processor [C] //Proc of Int Conf on Reconfigurable Computing and FPGAs (RECONFIG'08). Piscataway, NJ: IEEE, 2008: 438-443

[18] Wang Bo, Liu Leibo. A flexible and energy-efficient reconfigurable architecture for symmetric chipper processing [C] //Proc of IEEE Int Symp on Circuits and Systems (ISCAS 2015). Piscataway, NJ: IEEE, 2015: 1182-1185

**Li Gongli**, born in 1981. PhD candidate of the PLA Information Engineering University. Her main research interests include cryptographic arithmetic, computer architecture and high-performance cryptographic processor.

**Dai Zibin**, born in 1966. Professor and PhD supervisor in the PLA Information Engineering University. His main research interests include cryptographic arithmetic, VLSI design of crypto-ICs, and SoC platform for security applications.

**Xu Jinhui**, born in 1978. PhD. His main research interests include reconfigurable computing, information security and SoC system design.

**Wang Shoucheng**, born in 1992. Master. His main research interests include reconfigurable computing and SoC system design.

**Zhu Yufei**, born in 1990. Master. His main research interests include VLSI designs for security applications.

**Feng Xiao**, born in 1987. PhD. Her main research interests include high performance multi-core processor, design of SoC platform for security applications.