

一种面向分组密码的指令扩展方法

刘 恺¹, 敖天勇^{1,2}, 饶金理¹, 戴 葵¹, 邹雪城¹

(1 华中科技大学 光学与电子信息学院, 湖北 武汉 430074;

2 河南大学 物理与电子学院, 河南 开封 475004)

摘 要: 针对信息安全领域广泛存在的分组密码运算需求, 提出一种面向分组密码的指令扩展方法. 通过统计分析 47 种分组密码算法的运算特点, 发现了四种需要加速的基本运算, 并设计了四个密码运算单元对这四种基本运算进行加速. 将这四个密码运算单元设计成为一个数据触发单元植入微处理器中, 从而实现了分组密码运算的加速. 该方法具有实现简单、灵活性高等优点. 评估结果显示扩展后的微处理器对于常见分组密码算法的加速比为 2.4~9.3, 且硬件开销仅为原微处理器的 1.3 倍.

关键词: 分组密码; 指令扩展; 微处理器; 数据触发

中图分类号: TP309

文献标识码: A

文章编号: 1000-7180(2015)11-0087-05

An Instruction Set Extension Method for Block Cipher

LIU Kai¹, AO Tian-yong^{1,2}, RAO Jin-li¹,

DAI Kui¹, ZOU Xue-cheng¹

(1 School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan 430074, China; 2 School of Physics and Electronics, Henan University, Kaifeng 475004, China)

Abstract: In order to meet the high-performance block cipher processing need existed in the information security field, an instruction set extension method for block cipher calculation is proposed. Based on the analysis of 47 kinds of block cipher algorithms, four basic operations that should be accelerated were found. Therefore, four crypto-operation units to accelerate these operations are designed. Moreover, a data trigger unit which includes the four crypto-operation units is designed and added to the microprocessor to accelerate the block cipher operations. This method has the advantages of briefness, flexibility and so on. The evaluation result shows that the modified microprocessor's speed-up ratios of common block cipher algorithms are about 2.4~9.3 and the hardware cost is only 1.3 times as large as the original microprocessor.

Key words: block cipher; instruction set extension; microprocessor; data trigger

1 引言

分组密码是一种常用的密码技术, 被广泛应用于 SSL、IPsec 等各种安全协议中. 分组密码有软件实现和硬件实现两种实现方式. 软件实现灵活性高, 但是性能和安全性较低. 而硬件实现性能和安全性较高, 但因为固化了算法导致灵活性较差. 一种折中的方法是指令扩展, 它是一种处理器增强技术, 通过

在处理器中增加运算单元、扩展指令集, 能够提高处理器处理某类运算的能力. 该技术已经广泛地运用于 Intel、AMD 等公司的处理器中. 面向分组密码进行指令扩展, 能够增加微处理器的密码运算能力, 极大地保障了手机等设备的信息安全.

关于面向分组密码的指令扩展, 国内外已经展开了广泛的研究. 文献[1]中设计了 sbbox4s/isbox4s/sbox4r 指令和 mixcol4s/imixcol4s 指令, 分

收稿日期: 2014-12-11; 修回日期: 2015-01-16

基金项目: 国家自然科学基金(61376031); 湖北省自然科学基金(ZRZ0051); 河南省重点科技攻关(152102210055)

别处理 AES 算法中的字节替换和列混合运算. 文献[2]设计了比特置换指令来加速含有比特置换运算的密码算法. 上述指令扩展的研究成果主要是针对一种算法或几种算法共有的一种操作, 适用面比较窄. 近年来出现了一些面向多种分组密码算法的指令扩展方法. 文献[3]基于 MIPS 微处理器, 扩展了移位运算、S 盒、比特置换等指令, 对多种分组密码算法提供了支持. 文献[3]虽然面向了多种分组密码算法, 但是需要对微处理器进行较大改动. 同时, 该方法缺乏对密码运算特点的深入分析和对硬件开销的考虑. 因此, 提出了一种面向多种分组密码算法的指令扩展方法.

2 分组密码的运算特点分析

2.1 分组密码的基本运算

结合文献[4]统计了分组密码算法的基本运算在 47 种算法中出现的频率, 结果如表 1 所示.

表 1 分组密码算法基本运算的使用频率

基本运算	使用频率/%	基本运算	使用频率/%
逻辑	97.9	比特置换	25.5
移位	85.1	有限域乘法	17.0
查找表	72.3	模乘	17.0
模加/减	46.8	模乘逆	2.1

其中逻辑运算和移位运算属于微处理器标准指令类型. 模加/减运算通常可以通过加/减法和移位指令实现. 模乘逆运算使用频率低, 硬件开销大. 因此, 本文不考虑上述四种运算. 对于剩下的查找表运算、有限域乘法运算、比特置换运算和模乘运算, 结合文献[5]的工作, 统计它们在常用的分组密码算法的运算时间中占的比例, 如表 2 所示. 由表 2 可知, 以上 4 种基本运算是分组密码算法中比较耗时的运算. 通过以上统计分析可以得出, 查找表运算、有限域乘法运算、比特置换运算和模乘运算在分组密码算法中常见且耗时. 因此本文将扩展微处理器的指令集, 加速这四种基本运算.

表 2 基本运算的时间比例

基本运算	AES/%	SM4/%	DES/%	IDEA/%	Twofish/%
查找表	72	63	35	0	24
有限域乘法	0	0	0	0	54
比特置换	0	0	32	0	0
模乘	0	0	0	73	0

2.2 扩展的基本运算特点分析

本文对需要扩展的 4 种基本运算进行深入分析, 研究它们的常见形式.

(1) 查找表运算

查找表运算可以分为使用内容固定的表格和使用内容可变的表格. 在 AES 等 34 种具有查找表运算的算法中, 仅有 3 种算法使用可变表格, 因此, 本文只考虑固定表格的查找表运算. 查找表运算有不同的类型, 比如 8×8 、 8×32 等, 其中前一个数字表示输入的位宽, 后一个数字表示输出的位宽. 通过对 31 种算法进行分析, 得出各种类型查找表的使用频率如表 3 所示. 由表 3 可知, 查找表的类型主要是 8×8 , 而且 4×4 、 6×4 、 3×3 和 6×2 的查找表可以基于 8×8 的查找表实现. 因此, 本文分析 8×8 的查找表.

表 3 各种类型查找表的使用频率

类型	频率/%	类型	频率/%	类型	频率/%
8×8	38.7	12×8	3.2	3×64	3.2
8×32	19.4	13×8	3.2	6×2	3.2
4×4	12.9	10×8	3.2	3×3	3.2
6×4	6.5	8×64	3.2		

在分组密码算法中, 一次轮变换可能用到多个查找表, 本文将一次轮变换用到的查找表个数称为“并行度”. 某些算法中可能会出现多个内容不同的查找表, 本文将内容不同的查找表个数称为“种类”. 在 47 种算法中, 有 12 种具有 8×8 的查找表, 本文对这些算法中查找表的并行度和种类进行分析, 统计结果如表 4 所示. 由表 4 可知, 分组密码算法中常见的并行度是 16, 种类是 1. 因此本文讨论并行度是 16, 种类是 1 的情况.

表 4 8×8 查找表的并行度和种类

算法	并行度	种类	算法	并行度	种类
AES	16	1	E2	8	1
square	16	1	SHARK	8	1
CRYPTON	16	2	FOX128/256	8	1
SAFER++	16	2	SM4	4	1
ARIA	16	2	FOX64/128	4	1
FROG	16	8	Skipjack	2	1

(2) 有限域乘法运算

通过对 AES 等 8 种具有有限域乘法运算的分组密码算法进行分析, 发现其中 6 种算法中的有限

域乘法是在 $GF(2^8)$ 上的. 因此, 本文讨论 $GF(2^8)$ 上的有限域乘法. 有限域乘法通常以矩阵乘的形式出现, 即一个常数矩阵左乘一个状态矩阵. 通过对 6 种算法的常数矩阵和状态矩阵的规模进行分析, 得出统计结果如表 5 所示. 由表 5 可知, 常数矩阵通常是 4×4 , 状态矩阵通常是 4×1 . 因此本文讨论常数矩阵是 4×4 , 状态矩阵是 4×1 的情况.

表 5 常数矩阵及状态矩阵的规模

算法	常数矩阵规模	状态矩阵规模
AES	4×4	4×4
CLEFIA	4×4	4×1
square	4×4	4×1
Twofish	4×4	4×1
FOX64/128	4×4	4×1
FOX128/256	8×8	8×1

(3) 比特置换运算

比特置换运算分为 64-64、32-32 等不同的规模, 前一个数字表示输入数据的位宽, 后一个数字表示输出的位宽. 通过对 47 种算法中出现的 14 种比特置换运算进行分析, 发现其中有 11 种能够通过 64-64 的比特置换实现. 因此, 本文讨论 64-64 的比特置换. 另外, 在 DES、LOKI91 等算法中存在一种输入位宽小于输出位宽的比特置换, 最典型的是 32-48 的比特置换, 该比特置换难以用一般的方法加速. 本文将其称为扩展置换, 并予以特别考虑.

(4) 模乘运算

大部分算法中的模乘运算是对 2^{32} 取模, 可以通过现有微处理器指令实现. 但在运用广泛的 IDEA 算法中, 存在一种对 $2^{16} + 1$ 取模的模乘运算, 难以用现有指令实现. 因此, 本文对该模乘运算进行加速.

3 指令扩展的实现

3.1 数据触发的指令扩展结构

本文提出的指令扩展体系结构如图 1 所示. 在微处理器中增加了一个数据触发单元, 数据触发单元中包含控制逻辑、RS(源寄存器)、RD(目的寄存器)和查找表、有限域乘等 4 个密码运算单元. 图中的加粗箭头表示数据流. 密码运算单元使用 RS 中的数据进行计算, 将计算结果存放在 RD 中. 微处理器的 REG BANK(寄存器堆)和 MEM(存储器)中的待计算数据可以传输给数据触发单元中

的 RS. 数据触发单元中 RD 内的计算结果可以通过写回逻辑传输给微处理器的 REG BANK. 通过写一个属于 RS 的特殊的触发寄存器(命名为 RT), 就可以触发密码运算单元进行运算, 并更新 RD 的值, 不需要额外的启动运算指令. 使用该结构进行指令扩展, 只需要扩展 MOVC 和 STOREC 两条指令就能够支持多种密码运算. MOVC 指令执行的操作如式(1)所示, STOREC 指令执行的操作如式(2)所示.

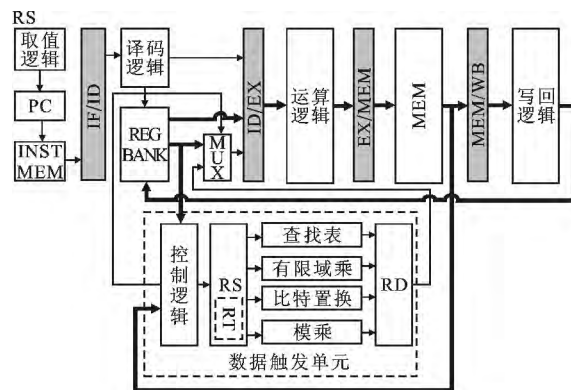


图 1 数据触发的指令扩展结构

$$RS \leftarrow \text{REGBANK}; ID/EX. A \leftarrow RD \quad (1)$$

$$RS \leftarrow \text{MEM} \quad (2)$$

可以看出, 指令扩展主要是通过增加选择器、改变数据流来实现的, 因而该结构不需要对微处理器进行复杂的修改, 具有很高的设计可靠性. 同时, 我们只需要增加密码运算单元就可以增加支持的运算类型, 不需要增加扩展指令的数目, 具有很高的灵活性和可扩展性. 另外, 密码运算单元与微处理器的流水线之间是相对独立的, 它们可以并行工作, 一方面减小了多周期运算的指令扩展难度, 另一方面提高了性能. 分组密码算法需要扩展的运算类型很多, 同时有部分运算是多周期运算, 因此特别适合使用该方法进行指令扩展.

3.2 密码运算单元的设计

使用图 1 所示的结构进行指令扩展, 核心在于密码运算单元的设计. 对于各个密码运算单元, 本文提出了多种实现方案, 并比较了各方案的性能和硬件开销, 从而选择了最有性价比的方案. 在评价硬件开销时, 本文使用 UMC 的 $0.18 \mu\text{m}$ 工艺.

(1) 查找表运算单元

查找表运算单元的硬件实现有三个方案, 分别是使用 1 个 256×8 的单端口 SRAM, 1 个 256×8 的双端口 SRAM 和 2 个 256×8 的双端口 SRAM.

这三个方案分别能同时完成 1、2、4 个 8×8 的查找表运算. 各个方案的完成运算所需时钟周期数、面积和面积时间积(命名为 AT)如表 6 所示, 其中面积时间积等于面积乘以周期数, 数值越小则性价比越高. 由表 6 可知, 方案二性价比最高, 因此将其作为最终方案.

表 6 查找表运算单元的性能、面积和面积时间积

方案	周期数	面积/等效门数	AT
无扩展	80	—	—
方案一	21	4 420	92 820
方案二	13	6 941	90 233
方案三	9	13 251	119 259

(2) 有限域乘运算单元

本文将 $GF(2^8)$ 上的有限域乘法器称为 GF-MUL, 对图 2 所示三个方案进行分析. 方案一只有 1 个 GF-MUL, 1 个时钟周期只能完成 1 个有限域乘法运算. 方案二含有 4 个 GF-MUL, 1 个时钟周期可以完成 4 个有限域乘法运算. 方案三相对于方案二将 32 位的 RS0 换成了 128 位的常数矩阵寄存器, 不必在运算前将常数矩阵的某一行从存储器中装载到密码运算单元中.

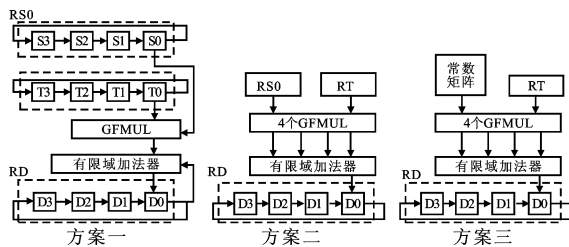


图 2 有限域乘运算单元硬件方案的结构框图

各个方案的性能、面积和面积时间积如表 7 所示. 无指令扩展时, 运算需要的时钟周期数与常数矩阵有关, 一般大于 44. 由表 7 可知, 方案三性价比最高, 因此将其作为最终方案.

表 7 有限域乘运算单元的性能、面积和面积时间积

方案	周期数	面积/等效门数	AT
无扩展	>44	—	—
方案一	21	1 273	26 733
方案二	9	2 523	22 707
方案三	5	3 460	17 300

(3) 比特置换运算单元

文献[6]中使用 BENES 网络实现比特置换运

算, 但是 BENES 网络不能很好地支持扩展置换. 另一种比特置换的方法是使用多个 64 选 1 的选择器, 这种方法对于扩展置换提供了较好的支持. 本文共提出 2 种方案进行比较, 方案一和方案二的结构如图 3 所示. 方案一使用 64-64 的 BENES 网络, 配置寄存器为 352 位, 其中低 32 位为触发寄存器. 载入完配置信息后, 将 RS0 中的源数据进行比特置换, 结果存在 RD 中. 方案二使用 5 个 64 选 1 的选择器, 配置寄存器为 30 位. 每装载一次配置信息, 对 5 位数据进行比特置换, 循环 13 次, 最终完成 64-64 的比特置换.

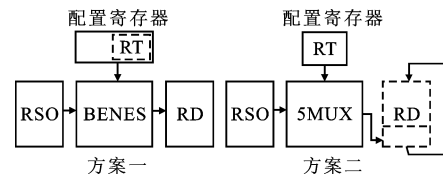


图 3 比特置换运算单元硬件方案的结构框图

各个方案实现 64-64 的比特置换和 32-48 的比特置换所需要的时钟周期数和面积如表 8 所示. 由表 8 可知, 方案二的性价比最高, 因此将其作为最终方案.

表 8 比特置换运算单元的性能和面积

方案	64-64	32-48	面积/等效门数
无扩展	191	143	—
方案一	18	59	5 749
方案二	20	18	2 161

(4) 模乘运算单元

针对 IDEA 算法中的模乘运算, 本文设计了相应的模乘运算单元. 无指令扩展和有指令扩展的性能和面积如表 9 所示. 由表 9 可知, 增加模乘运算单元, 并没有增加多大的硬件开销, 但是能极大地提高模乘速度, 因此本文在数据触发单元中放置了一个模乘运算单元.

表 9 模乘运算单元的性能和面积

方案	周期数	面积/等效门数
无扩展	16	—
方案一	4	2 509

4 实现结果与分析

本文基于 MIPS 微处理器实现了指令扩展, 同时在扩展后的微处理器上实现了 AES、SM4、DES 和 IDEA 四种密码算法. 表 10 给出了完成这些算法所需

要的时钟周期数,同时列举了相关文献中的设计结果.其中MIPS是指令扩展前的MIPS微处理器.

表10 时钟周期数对比

算法	文献[3]	ARMv4 ^[7]	MIPS	本文
AES	283	2 044	2 066	357
SM4	—	1 176	1 304	549
DES	116	1 974	1 996	742
IDEA	234	2 380	2 387	256

表11 硬件开销对比

硬件开销种类	文献[3]	ARMv4 ^[7]	MIPS	本文
逻辑资源开销	7074 ALUTs	—	2752 ALUTs	3966 ALUTs
存储器资源开销	>15360 bit	—	0 bit	2048 bit

可以看出,本文所提出的结构对于AES、SM4、DES和IDEA这4种分组密码算法的加速比分别为5.8、2.4、2.7和9.3,同时硬件开销仅相当于原微处理器的1.3倍.相比于文献[3]的设计,本文的性能虽然略显不足,但是硬件开销要低44%以上.

5 结束语

本文提出了一种面向分组密码的指令扩展方法并进行了实现.通过在微处理器中增加一个含有四个密码运算单元的数据触发单元,并扩展两条与数据触发单元进行数据交互的指令,可以增强微处理器的分组密码运算能力.该方法能够加速多种分组密码算法,具有实现简单、灵活、可扩展性强等优点.同时本文针对各个密码运算单元提出了多种设计方案并进行了实现,给出了各个方案的性能和硬件开销.对扩展后微处理器的评估结果显示,对于常见分组密码算法加速比可以达到2.4~9.3,同时等效门数仅为扩展前微处理器的1.3倍,是一种优秀的指令扩展方法.

参考文献:

- [1] Tillich S, Großschädl J. Instruction set extensions for efficient AES implementation on 32-bit processors[C]

使用UMC的0.18 μm 工艺对扩展后的微处理器进行了综合,其等效门数为111 216,相当于扩展前微处理器的1.3倍.为了与其他文献进行对比,将本设计在FPGA上进行实现,结果如表11所示,其中逻辑资源开销指整个微处理器消耗的逻辑资源,而存储器资源开销指扩展后的微处理器相对于扩展前的微处理器增加的存储器资源.文献[3]、MIPS及本文使用Stratix系列中的EP2S60F1020C3芯片作为目标器件.

// Cryptographic Hardware and Embedded Systems—CHES 2006. Berlin: Springer, volume 4249 of Lecture Notes in Computer Science, 2006: 270–284.

- [2] Shi Zhijie. Bit permutation instruction architecture implementation and cryptographic properties[D]. New York: Princeton University, 2004.
- [3] 张琰. RISC结构专用指令密码处理器研究与设计[D]. 郑州: 中国人民解放军信息工程大学, 2008.
- [4] 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.
- [5] Fiskiran A M, Lee R B. Performance scaling of cryptography operations in servers and mobile clients[C]// Proceedings of the Workshop on Building Block Engine Architectures for Computer Networks (BEACON). New York, 2004.
- [6] 向楠, 戴紫彬, 徐劲松. 一种基于BENES网络的可重构比特置换系统设计[J]. 计算机工程, 2007, 33(22): 178–180.
- [7] 倪晓强. 通用并行向量密码处理器研究[D]. 长沙: 国防科技大学, 2005.

作者简介:

刘 恺 男, (1990–), 硕士研究生. 研究方向为数字集成电路设计. E-mail: u200818077@gmail.com.