

## Design and Implementation of Low-Cost SM4 Algorithm Based on RISC-V Instruction Set Extension\*

CHEN Rui<sup>1\*</sup>, LI Bing<sup>2</sup>, LIU Xiangdong<sup>1</sup>

(1. School of Computer and Software Engineer, Nanjing Vocational University of Industry Technology, Nanjing Jiangsu 210012, China;

2. School of Microelectronics, Southeast University, Nanjing Jiangsu 210035, China)

**Abstract:** Considering constrained resource and low cost of Industrial Internet of Things (IIoT) devices, in order to protect the confidentiality of data collected by the IIoT devices, hardware/software co-design of SM4 algorithm is proposed to balance resource overhead, performance and delay. Based on open-source RISC-V instruction set architecture (ISA), two customized instructions are added to realize round functions of encryption/decryption and key expansion, and a low-cost hardware architecture of SM4 instruction function unit is designed and implemented. According to cycle-accurate simulation results, the latency is reduced by 81.72% and throughput is improved 4.47 times compared with implementation without custom instructions. The synthesis results under SMIC 180 nm technology demonstrate that, compared with related works, the hardware resource overhead is reduced by 38.9% at least.

**Key words:** industrial IoT; SM4; RISC-V; instruction set extension; encryption

EEACC: 7220

doi: 10.3969/j.issn.1005-9490.2021.01.021

## 基于 RISC-V 指令扩展的低开销 SM4 算法设计与实现\*

陈锐<sup>1\*</sup>, 李冰<sup>2</sup>, 刘向东<sup>1</sup>

(1. 南京工业职业技术大学计算机与软件学院, 江苏 南京 210012; 2. 东南大学微电子学院, 江苏 南京 210035)

**摘要:** 为了保障工业物联网采集数据的机密性, 同时考虑到物联网终端设备资源受限与成本低廉的特点, 提出以软硬件协同设计的方式实现 SM4 算法, 以平衡资源开销、性能和延时。在开源 RISC-V 指令集的基础上, 增加了两条自定义指令以实现密钥扩展和解密算法的轮函数, 设计了一款低开销的 SM4 指令功能单元硬件电路结构。从时钟周期精确的仿真结果来看, 与无扩展指令的实现相比, 延时缩减 81.72%, 吞吐率提升 4.47 倍。从 SMIC 180 nm 工艺下综合结果来看, SM4 指令功能单元仅占用了 1684 门, 与参考文献相比, 资源开销至少降低 38.9%。

**关键词:** 工业物联网; SM4; RISC-V; 指令扩展; 加密

中图分类号: TP309.7

文献标识码: A

文章编号: 1005-9490(2021)01-0108-06

近年来, 物联网 (Internet of Things, IoT) 已被广泛应用于各个领域, 特别是在工业领域<sup>[1]</sup>。越来越多的物联网设备被安装在车间、厂房、机械装置和工业装备上<sup>[2]</sup>。工业物联网不仅提高了生产效率, 而且降低了生产成本<sup>[3]</sup>。随着联网设备的增多, 物联网面临的安全问题也越来越严峻<sup>[4]</sup>。造成这一问题的主要原因是三个。首先, 出于成本方面的考虑,

物联网设备资源受限, 无法提供先进的安全技术保障<sup>[5]</sup>。其次, 物联网设备数量的增多, 导致探索潜在安全漏洞的机会增多。最后, 物联网设备产生、处理和交换大量对公共安全至关重要的数据以及对隐私敏感的信息<sup>[6]</sup>, 这些数据和信息对攻击者具有一定的吸引力。一般而言, 物联网设备采集的数据会通过网络传递到云端服务器进行分析和处理<sup>[7]</sup>。

**项目来源:** 江苏省工业软件工程技术研究开发中心开放基金重点项目 (ZK19-04-03)

**收稿日期:** 2020-06-28 **修改日期:** 2020-07-30

这些数据在传输过程中,可能被中间人篡改、删除,从而破坏了数据的完整性、可靠性和机密性。考虑到这些数据可能对公共安全至关重要或者对隐私敏感,应该在传输数据之前,对这些数据进行加密处理。

对称密码算法可用于数据加密。常见的对称密码算法有美国的 AES 标准、中国的 SM4 标准等。SM4 (也称为 SMS4) 已成为中国国家标准,与 AES 相比,SM4 具有以下特点,使其更适用于资源受限的环境:(1) SM4 的安全特性等效于 AES-128<sup>[8]</sup>;(2) 加密和解密的结构相同;(2) 用于加密和解密的 Sbox 相同;(4) 轮函数仅需要 4 个 Sbox (每个具有 256×8 位),而在 AES 中则需要 16 个。SM4 算法的实现可以通过软件、硬件或者软硬协同的方式。软件实现的性能较低,特别是对于性能和延时比较敏感的工业场景,软件执行引入的延时是难以满足工业场景对于性能和延时的需求。因此,相关研究工作主要关注于硬件电路实现。虽然 SM4 算法本身已具备适合于资源受限环境的众多优势,但是考虑到物联网设备成本方面的因素,需要尽可能地降低 SM4 算法硬件实现时带来的资源开销。目前已经有不少参考文献致力于降低 SM4 算法的硬件电路资源开销,比如基于复合域算术实现 Sbox<sup>[8-9]</sup>,降低了 Sbox 导致的资源开销;基于异步多米诺逻辑实现 SM4<sup>[10]</sup>,降低功耗的同时也降低了面积开销。然而,从物联网设备成本方面考虑,SM4 算法硬件电路资源开销还需进一步地降低。

为了降低 SM4 硬件资源开销,本文提出以软硬件协同设计的方式实现 SM4 算法。首先,分析了 SM4 软件实现的性能瓶颈。其次,基于软件分析结果,在开源指令集 RISC-V<sup>[11]</sup>的基础上,提出了两条自定义指令,分别用于实现 SM4 算法的加解密算法和密钥扩展算法的轮函数。最后,提出一种复用 RISC-V 处理器寄存器资源的方法,以减少 SM4 存储资源开销,并设计了一款低开销的 SM4 指令功能单元电路结构。实验结果显示,相对于软件实现的性能,本文方法能够将吞吐率提升 4.47 倍,延时缩短 81.72%。在 SMIC 180nm 工艺下的综合结果显示,与参考文献相比,本文方法的硬件资源开销至少降低 38.9%。

## 1 SM4 算法简介

该算法由加解密算法(如图 1(a)所示算法 1)和密钥扩展算法(如图 1(b)所示算法 2)两部分组成,分组长度和密钥长度均为 128 bit。加解密算法

与密钥扩展算法均采用 32 轮非线性迭代结构。从算法 1 可以看出,加密算法和密钥扩展算法都需要 32 轮的迭代才能得到最终结果,因而每轮计算所消耗的时间决定了整个算法所消耗的时间。为了评估 SM4 软件实现的性能,本文以 Linux 内核中的 SM4 算法源代码为基础,在一款商业级的开源 RISC-V 处理器 SCRI<sup>[12]</sup>上运行,以获取时钟周期精确的仿真结果,然后对其进行了性能剖析。

算法 1: SM4 加密算法

```

输入: 128-bit 明文  $P = (P_0, P_1, P_2, P_3) \in (\mathbb{Z}_2^{32})^4$ , 轮密钥  $rk_i \in \mathbb{Z}_2^{32}, i = 0, 1, \dots, 31$ 
输出: 128-bit 密文  $C = (C_0, C_1, C_2, C_3) \in (\mathbb{Z}_2^{32})^4$ 
1  $B \in (\mathbb{Z}_2^8)^4$ ;
2  $S, L \in (\mathbb{Z}_2^{32})^4$ ;
3 for  $i = 0; i < 32; i++$  do
4    $B = (b_0, b_1, b_2, b_3) = P_{i+1} \oplus P_{i+2} \oplus P_{i+3} \oplus rk_i$ ;
5    $S = \tau(B) = (Sbox(b_0), Sbox(b_1), Sbox(b_2), Sbox(b_3))$ ;
6    $L = L(S) = S \oplus (S \ll 2) \oplus (S \ll 10) \oplus (S \ll 18) \oplus (S \ll 24)$ ;
7    $P_{i+4} = P_i \oplus L$ ;
8 end
9  $C = (C_0, C_1, C_2, C_3) = R(P_{32}, P_{33}, P_{34}, P_{35}) = (P_{35}, P_{34}, P_{33}, P_{32})$ ;
10 return C

```

(a) SM4 加密算法

算法 2: SM4 密钥扩展算法

```

输入: 128-bit 密钥  $MK = (MK_0, MK_1, MK_2, MK_3) \in (\mathbb{Z}_2^{32})^4$ 
输出: 轮密钥  $rk_i \in \mathbb{Z}_2^{32}, i = 0, 1, \dots, 31$ 
1  $(K_0, K_1, K_2, K_3) = (FK_0 \oplus MK_0, FK_1 \oplus MK_1, FK_2 \oplus MK_2, FK_3 \oplus MK_3)$ ;
2 for  $i = 0; i < 32; i++$  do
3    $CK_i = (ck_{i,0}, ck_{i,1}, ck_{i,2}, ck_{i,3}), ck_{i,j} = (4i + j) \times 7 \pmod{256}$ ;
4    $B = (b_0, b_1, b_2, b_3) = K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i$ ;
5    $S = \tau(B) = (Sbox(b_0), Sbox(b_1), Sbox(b_2), Sbox(b_3))$ ;
6    $L = L(S) = S \oplus (S \ll 13) \oplus (S \ll 23)$ ;
7    $rk_i = K_i \oplus L$ ;
8 end
9 return  $rk_i$ ;

```

(b) SM4 密钥扩展算法

图 1 SM4 加密算法和密钥扩展算法

分析结果如图 2 所示,仅通过 Sbox 完成一个字节的非线性变换就消耗了 6 个时钟周期,而密钥扩展和加密算法轮函数的单次迭代分别需要 41 和 49 个时钟周期。值得注意的是,完成 32 次轮函数迭代,密钥扩展和加解密分别需要 1 802 和 2 178 个时

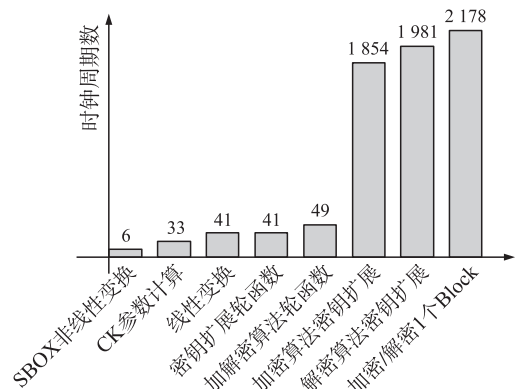


图 2 SM4 算法子函数性能分析

钟周期,而这两个值并不等于  $41 \times 32$  和  $49 \times 32$ 。造成这一现象的主要原因是,每次轮函数的迭代都需要通过多个访存指令从存储器中将数据载入到寄存器中。基于上述的分析,得出的结论是,为了消除 SM4 软件性能瓶颈,提升 SM4 性能,降低时延,必须做到:(1)降低 Sbox 非线性变换消耗时间;(2)降低轮函数单次迭代消耗时间;(3)减少轮函数中的访存指令以减少访问存储器消耗时间。

从算法 1 中的 4~7 行和算法 2 的 3~7 行可以发现,轮函数中包含的运算较多,如果能将这些运算通过一条指令完成,则可以达到降低轮函数单次迭代消耗时间的目的,而 Sbox 非线性变换已然包含在轮函数中了,因此上述结论(1)也可以通过扩展指令来消除或掩盖。考虑到开源指令 RISC-V 指令集定义了 32 个通用寄存器,如果能够借用部分通用寄存器用于存放加解密或者密钥扩展算法运算过程中产生的中间数据,则可以减少不必要的存储器访问从而消除存储器访问消耗时间。

## 2 RISC-V 指令扩展

基于上述的分析,依据开源 RISC-V 指令规范,本文定义了两条自定义指令,SM4.KEY.RF 和 SM4.ENC.RF。扩展指令 SM4.KEY.RF 用于实现密钥扩展算法轮函数单次迭代中所有运算,包括异或、Sbox 非线性变换、线性变换,如算法 2 中 4~7 行所示。扩展指令 SM4.ENC.RF 用于实现加解密算法轮

函数单次迭代中的所有运算,如算法 1 中 4~7 行所示。二者指令编码格式如图 3 所示。依据 RISC-V 指令规范,选取 7 位二进制数 0001011 作为 SM4 扩展指令的操作码,以便于与其他类型指令进行区分。在扩展指令 SM4.KEY.RF 中,目标寄存器由编译器决定,可以为任意寄存器,源寄存器只需要 1 个,用于载入 CK 参数。在扩展指令 SM4.ENC.RF 中,将指定目标寄存器为 x28,以减少不必要的存储器访问,而源寄存器用于载入密钥扩展算法生成的轮密钥。

|            |     |    |    |    |      |    |      |   |       |   |         |
|------------|-----|----|----|----|------|----|------|---|-------|---|---------|
|            | 31  | 20 | 19 | 15 | 14   | 12 | 11   | 7 | 6     | 0 |         |
| SM4.KEY.RF | 0   |    |    |    | rs1  |    | 001  |   | rd    |   | 0001011 |
| SM4.ENC.RF | 0   |    |    |    | rs1  |    | 010  |   | 11100 |   | 0001011 |
|            | 保留位 |    |    |    | 源寄存器 |    | 指令功能 |   | 目标寄存器 |   | 操作码     |

图 3 本文提出的两条 RISC-V 扩展指令

## 3 低开销 SM4 指令功能单元结构设计

依据两条扩展指令的功能,本文设计了一款低开销的 SM4 指令功能单元,并对 RISC-V 处理器的寄存器堆的电路结构进行了修改,具体结构如图 4 所示。从图 4 可以看出,SM4 指令功能单元有 6 个输入和 1 个输出端口,其中 5 个输入用于传递轮函数单次迭代所需的数据,而这 5 个输入直接从寄存器堆引入。需要注意的是,这 5 个从寄存器堆直接引入的输入,有 4 个是固定连接到某一个寄存器,剩余 1 个由指令传递的 rs1 选取。之所以这样设计,主要原因有 3 个:(1)由于轮函数单次迭代需要 5

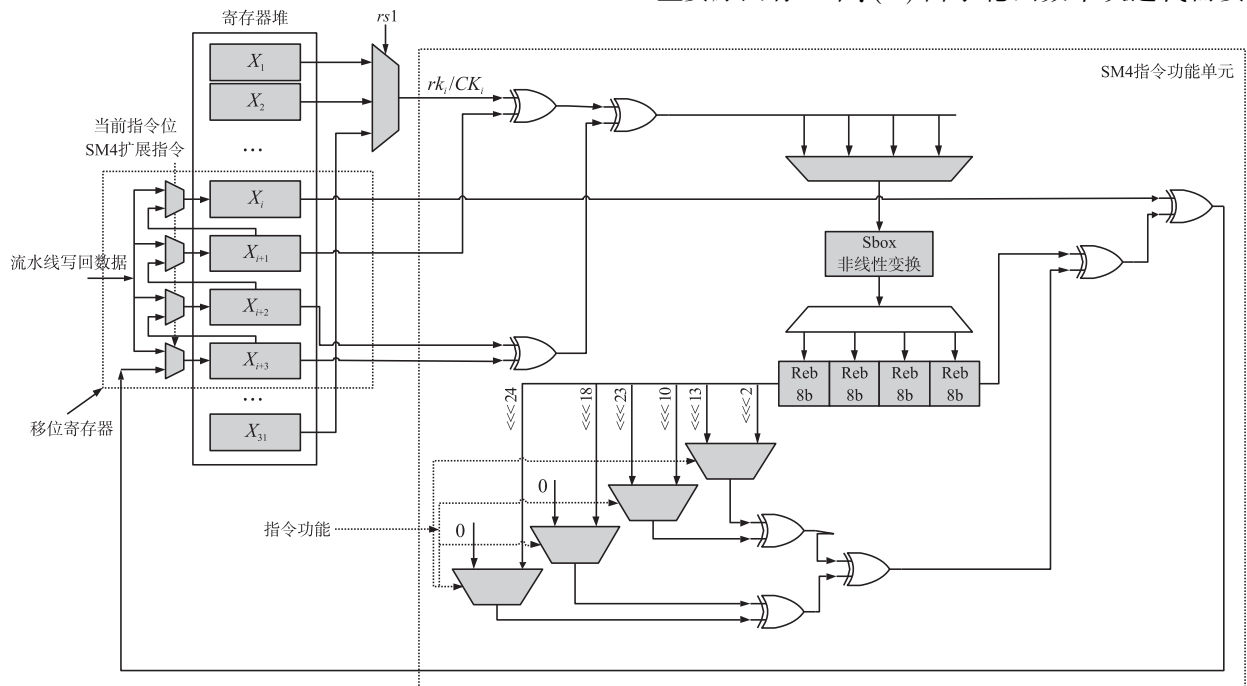


图 4 本文提出的 SM4 指令功能单元

个32位的数据,固定连接寄存器后,数据直接从寄存器中读取,无需等待,提升指令执行效率;(2)借用处理器的4个通用寄存器存放轮函数每次迭代产生的临时数据,无需再通过存储器访问指令从存储器中载入数据,减少访问存储器次数,缩短算法延时;(3)轮函数计算结果直接写入固定寄存器,无需编译器指定存放位置,省去读取寄存器时间,提升指令执行效率。

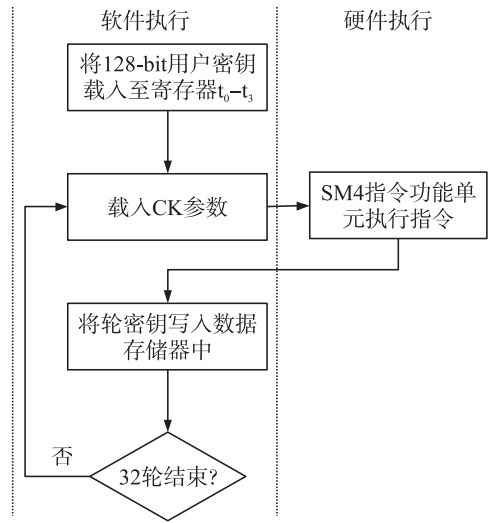
如图4所示,当处理器执行到SM4.ENC.RF或者SM4.KEY.RF指令时,寄存器堆中的寄存器 $t_0-t_3$ 构成移位寄存器,数据可以从 $t_3$ 移入,然后通过移位从 $t_0$ 移出。之所以这样设计,是为了在轮函数单次迭代结束之后,能够自动进行移位,为下一次的迭代准备数据,从而不需要再通过指令从其他位置加载数据,因而可以提升效率。

### 3.1 软硬件协同工作

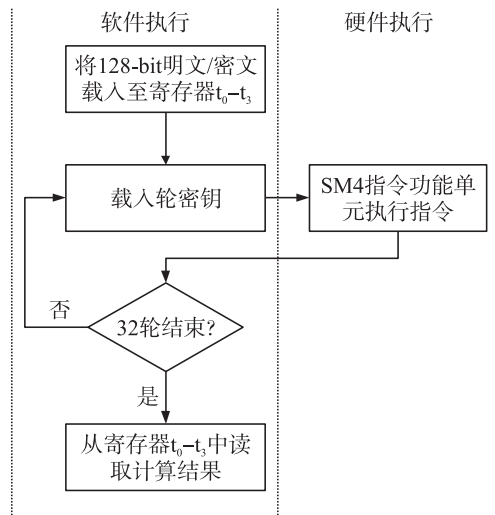
SM4指令功能单元仅完成轮函数单次迭代计算,完成整个加密算法或者密钥扩展算法,还需要软件的配合,软件主要负责算法流程的控制。为了节省扩展密钥所占用的资源开销,本文不采用在线密钥扩展,而是在加解密之前预先执行密钥扩展算法,并将生成的32个32位的扩展密钥存放到数据存储器中。如图5(a)所示,在调用SM4.KEY.RF指令进行密钥扩展之前,需要先将128位的用户密钥存放到指定的4个寄存器中,然后才能开始循环迭代。每次迭代都会生成一个32位的轮密钥,这些密钥交由编译器指定存放位置。在32个轮密钥生成完毕之后,可以开始数据加解密。

如图5(b)所示,在调用SM4.ENC.RF指令进行加解密之前,需要先将128位的明文或者密文存放到指定的4个寄存器中,然后才能开始循环迭代计算。每次迭代均会先载入一个轮密钥,然后执行

SM4.ENC.RF,每次迭代计算的结果直接写入到寄存器 $t_3(x28)$ ,32次迭代结束之后,直接从寄存器 $t_0-t_3$ 读取数据即可。



(a) 密钥扩展软硬件协同工作流程



(b) 加解密软硬件协同工作流程

图5 SM4算法软硬件协同工作流程

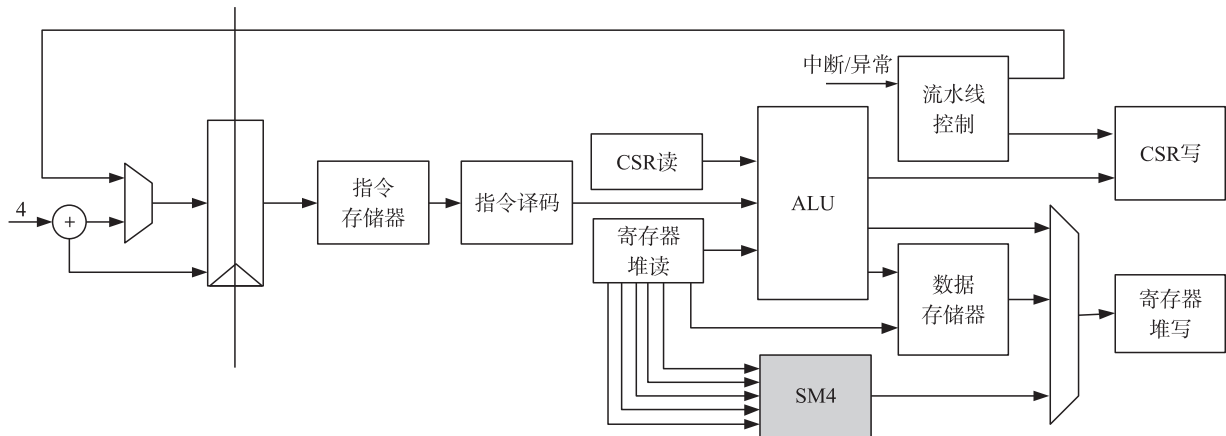


图6 SM4指令功能单元嵌入到开源RISC-V处理器SCR1中



## 4 实验与结果分析

为了验证扩展指令的功能,评估指令功能单元的资源开销,本文采用 Verilog HDL 对其进行了描述,然后将其集成到商业级开源 RISC-V 处理器 SCR1 中。如图 6 所示,SCR1 配置成了二级流水线结构,而 SM4 指令功能单元嵌入到流水线的第二级,与算术运算单元 ALU 处于同一个流水级。

### 4.1 面积开销

为了评估资源开销,本文将未修改的 SCR1 以及修改之后的 SCR1 分别以 100 MHz 的时钟频率在 SMIC 180 nm 工艺下,通过 Synopsys Design Compiler 进行综合,综合结果如表 1 所示。从表 1 可以看出,SM4 引入的硬件资源开销只有 1684 等效门。

表 1 SMIC 180nm 工艺下的 100MHz 综合结果

|          | 综合面积/ $\mu\text{m}^2$ | 等效门数   |
|----------|-----------------------|--------|
| SCR1     | 215 965               | 24 597 |
| SCR1+SM4 | 230 750               | 26 281 |
| SM4 Only | 14 785                | 1 684  |

注:等效门=综合面积/具有最小面积 NAND2 的面积  $8.78 \mu\text{m}^2$ 。

### 4.2 性能表现

为了评估性能,本文将 SM4 算法分别采用有指令集扩展和没有指令集扩展的方式进行软件实现,并通过 RISC-V 处理器提供的 CYCLE 计数器对算法消耗的时钟周期数进行计数。软件实现通过修改之后的 RISC-V GCC 交叉编译器进行编译,然后将编译输出的 Hex 文件以测试激励的方式载入到 Synopsys VCS 仿真平台,以获得时钟周期精确的仿真结果。如表 2 所示,添加指令集扩展之后,密钥扩展、加密和解密所需的时钟周期分别缩减了 81.72%、87.69%、82.40%,而吞吐率分别提升了 4.47 倍、7.12 倍和 4.68 倍。

表 2 有/无扩展指令延时(时钟周期)对比

|      | 无扩展指令 | 有扩展指令 | 缩减比率   |
|------|-------|-------|--------|
| 密钥扩展 | 1854  | 339   | 81.72% |
| 加密   | 2161  | 266   | 87.69% |
| 解密   | 2262  | 398   | 82.40% |

注:缩减比率=(无扩展指令-有扩展指令)/无扩展指令。

表 3 有/无扩展指令 100 MHz 时钟频率下的吞吐率对比

|      | 无扩展指令<br>/(Mbit/s) | 有扩展指令<br>/(Mbit/s) | 提升比率/倍 |
|------|--------------------|--------------------|--------|
| 密钥扩展 | 6.90               | 37.76              | 4.47   |
| 加密   | 5.92               | 48.12              | 7.12   |
| 解密   | 5.66               | 32.16              | 4.68   |

注:提升比率=(有扩展指令-无扩展指令)/有扩展指令。

### 4.3 与参考文献的对比

表 4 罗列了本文与参考文献的比较结果。虽然本文在延时和吞吐率方面的优势都不明显,但是面积和等效门数优势突出,造成这一现象的主要原因是各个设计的设计目标可能不一致,比如文献[13]瞄准的是高吞吐率,而本文的设计目标是低开销。从表 4 中数据可以看出,与参考文献相比,面积开销(等效门数)至少降低 38.9%。

表 4 与参考文献资源开销的比较结果

| 文献   | 工艺<br>/nm | 面积<br>/ $\mu\text{m}^2$ | 等效<br>门数 | 延时(时<br>钟周期)    | 吞吐率<br>/(Mbyte/s)     |
|------|-----------|-------------------------|----------|-----------------|-----------------------|
| [13] | 180       | ~271 000                | —        | 32              | 14 647                |
| [14] | 110       | 635 812                 | 124 860  | 35              | 13.4                  |
| [15] | 14        | —                       | 10 200   | 32              | 4040                  |
| [16] | 180       | —                       | 2 756    | 164             | 242                   |
| [17] | 180       | —                       | 3 060    | 128             | 185                   |
| [18] | 180       | 649 516                 | —        | 16              | 1 790                 |
| 本文   | 180       | 14 785                  | 1 684    | 339/266/<br>398 | 37.76/48.12/<br>32.16 |

## 5 总结

面向低成本物联网终端领域数据加密需求,针对 SM4 算法,本文基于开源 RISC-V 指令集,提出了两条 SM4 扩展指令,设计了一款低开销的 SM4 指令功能单元硬件电路结构,以软硬件协同工作的方式实现 SM4 密钥扩展算法和加解密算法,并在性能和硬件资源开销之间取得平衡。本文提出 2 条扩展指令分别用于实现 SM4 密钥扩展算法和加密算法轮函数中的所有运算。从时钟周期精确的仿真结果来看,与无扩展指令的实现方式相比,延时至少降低 81.72%,吞吐率至少提升 4.47 倍。从 SMIC 180 nm 工艺下的综合结果来看,与参考文献相比,硬件资源开销至少降低 38.9%。本文提出的方法能够兼顾性能和资源开销,因而较为适合于资源受限成本低廉的物联网场景。

### 参考文献:

- [1] Yu X, Guo H. A Survey on IIoT Security[C]//2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS). Singapore:IEEE,2019:1-5.
- [2] Chen B, Wan J, Shu L, et al. Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges[J]. IEEE Access, 2018,6:6505-6519.
- [3] He D, Ma M, Zeadally S, et al. Certificateless Public Key Authenticated Encryption with Keyword Search for Industrial Internet of Things[J]. IEEE Transactions on Industrial Informatics, 2018,14(8):3618-3627.

- [4] Zhou W, Jia Y, Peng A, et al. The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved [J]. IEEE Internet of Things Journal, 2019, 6(2): 1606–1616.
- [5] Zhou J, Cao Z, Dong X, et al. Security and Privacy for Cloud-Based IoT: Challenges [J]. IEEE Communications Magazine, 2017, 55(1): 26–33.
- [6] Sadeghi C, Wachsmann M. Security and Privacy Challenges in Industrial Internet of Things [C]//2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). San Francisco, CA, USA: IEEE, 2015: 1–6.
- [7] Aceto G, Persico V, Pescapé A. A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges [J]. IEEE Communications Surveys & Tutorials, 2019, 21(4): 3467–3501.
- [8] Satpathy S, Mathew S, Suresh V, et al. 250 mV~950 mV 1.1 Tbps/W Double-Affine Mapped Sbox Based Composite-Field SMS4 Encrypt/Decrypt Accelerator in 14 nm Tri-Gate CMOS [C]//2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits). Honolulu, HI, USA: IEEE, 2016: 1–2.
- [9] Fu H, Bai G, Wu X. Low-Cost Hardware Implementation of SM4 Based on Composite Field [C]//2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference. Chongqing, China: IEEE, 2016: 260–264.
- [10] Li Y, Wu X, Bai G. Implementation of SM4 Algorithm Based on Asynchronous Dual-Rail Low-Power Design [C]//2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT). Qingdao, China: IEEE, 2018: 1–3.
- [11] Waterman A, Asanovic K. The RISC-V Instruction Set Manual Volume I: Unprivileged ISA [EB/OL]. <https://content.riscv.org/wp-content/uploads/2019/12/riscv-spec-20191213.pdf>, 2020–12–13.
- [12] Syntacore. SCR1 [EB/OL]. <https://github.com/syntacore/scr1>, 2019–12–01.
- [13] 王泽芳, 唐中剑. SM4 算法 CTR 模式的高吞吐率 ASIC 实现 [J]. 电子器件, 2019, 42(1): 173–177.
- [14] Zheng X, Xu C, Hu X, et al. The Software/Hardware Co-Design and Implementation of SM2/3/4 Encryption/Decryption and Digital Signature System [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(10): 2055–2066.
- [15] Satpathy S, Suresh V, Mathew S, et al. 220 MV ~ 900 MV 794/584/754 GBPS/W Reconfigurable GF(24)2 AES/SMS4/Camellia Symmetric-Key Cipher Accelerator in 14NM Tri-Gate CMOS [C]//2018 IEEE Symposium on VLSI Circuits. Honolulu, HI, USA: IEEE, 2018: 175–176.
- [16] Zhu K, Zhang L, Dai Z, et al. Design and Implementation of Low-Cost SM4 for Consumer Electronic Product [C]//2016 IEEE International Conference on Consumer Electronics-China (ICCE-China). Guangzhou, China: IEEE, 2017: 1–5.
- [17] Shang M, Zhang Q, Liu Z, et al. An Ultra-Compact Hardware Implementation of SMS4 [C]//2014 IIAI 3rd International Conference on Advanced Applied Informatics. Kitakyushu, Japan: IEEE, 2014: 86–90.
- [18] 符天枢, 李树国. SM4 算法 CBC 模式的高吞吐率 ASIC 实现 [J]. 微电子学与计算机, 2016: 13–18.



陈锐 (1986—), 男, 副教授, 工学博士, 主要研究方向为密码算法 IP 及处理器架构设计;



李冰 (1968—), 男, 教授、博士生导师, 工学博士, 主要研究方向为网络安全算法与芯片设计;



刘向东 (1971—), 男, 副教授, 工学博士, 主要研究方向为计算机应用技术。