基于FPGA的SM4算法高效实现方案

张宏科[1,2,3]，袁浩楠[3,4]，丁文秀[1,3]，闫峥[1,3,4]，李斌[2]，梁栋[1,2]

（1.西安电子科技大学网络与信息安全学院，陕西 西安 710126；2.中国电子科技集团公司第五十四研究所，河北 石家庄 050299；
3.西安电子科技大学空天地一体化综合业务网全国重点实验室，陕西 西安 710071；
4.西安电子科技大学杭州研究院，浙江 杭州 311200）

摘　要：针对SM4算法的FPGA实现方案存在数据处理速度不够高和逻辑资源占用过高的问题，提出了基于现场可编程门阵列（FPGA）的高性能、低资源消耗的SM4算法实现方案。所提方案采用循环密钥扩展与32级流水线加解密相结合的架构，循环密钥扩展的方式降低了逻辑资源消耗，32级流水线加解密的方式提高了数据吞吐率。同时，所提方案采用代数式S盒并通过合并线性运算以及在不可约多项式的合并矩阵中筛选最优矩阵运算的方式进一步减少S盒变换的运算量，从而达到降低逻辑资源占用与提高工程数据吞吐率的目的。测试结果显示，该方案比现有最佳方案在数据吞吐率上提升了43%，且资源占用率降低了10%。

关键词：SM4算法；FPGA实现；流水线架构；代数式S盒

中图分类号：TN92

文献标志码：A

DOI：10.11959/j.issn.1000–436x.2024053

Efficient implementation scheme of SM4 algorithm based on FPGA

ZHANG Hongke[1,2,3], YUAN Haonan[3,4], DING Wenxiu[1,3], YAN Zheng[1,3,4], LI Bin[2], LIANG Dong[1,2]

1. School of Cyber Engineering, Xidian University, Xi'an 710126, China
2. The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang 050299, China
3. State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
4. Hangzhou Institute of Technology, Xidian University, Hangzhou 311200, China

Abstract: To address the inefficient data processing performance and excessive resource utilization issues that field-programmable gate array (FPGA)-based SM4 implementations faced, an implementation scheme that adopted both iteration and pipeline in order to reduce resource consumption and improve throughput was proposed. A combination of cyclic key extension and 32 bit pipeline encryption and decryption architecture was adopted by the proposed scheme. The cyclic key extension reduced logical resource consumption, while the 32 bit pipeline encryption and decryption improved data throughput. Additionally, an algebraic S-box that combined linear operations to select an optimal matrix from those generated by different irreducible polynomials was employed. Resource usage and computation overhead was further minimized, thus achieving an increased engineering frequency. Experimental results demonstrate a 43% throughput improvement and a 10% reduction in resource usage compared to the current best scheme.

Keywords: SM4 algorithm, FPGA implementation, pipeline architecture, algebraic S-box

---

收稿日期：2023–11–01；修回日期：2024–02–03

通信作者：闫峥，zyan@xidian.edu.cn

基金项目：国家自然科学基金资助项目（No.U23A20300, No.62072351）；陕西省自然科学基础研究计划重点基金资助项目（No.2023-JC-ZD-35）；陕西省自然科学基础研究计划基金资助项目（No.2023-JC-YB-500）

Foundation Items：The National Natural Science Foundation of China (No.U23A20300, No.62072351), The Key Research Project of Shaanxi Natural Science Foundation (No.2023-JC-ZD-35), The Natural Science Basic Research Plan in Shaanxi Province of China (No.2023-JC-YB-500)

0　引言

数字化、网络化和智能化是信息化时代的主要特征。物联网和云计算在发展过程中产生了大量数据，它们将数据保存在各类存储设备中。这些存储设备携带着大量的机密信息和重要数据，一旦丢失、被窃取或未经授权使用将给政府、企业和个人造成巨大的损失，情形严重的甚至会影响国家安全。存储前对数据进行加密是防范存储设备数据泄露的重要手段。数据加密的核心技术是数据加密算法。SM4算法是国家密码管理局认证的商用分组密码算法，已经成为一项国际标准，适用于存储系统中对大量数据进行加解密处理。

SM4算法主要有3种实现方法。第一种是用处理器运行软件实现。文献[1]中给出了一种基于复合域的SM4算法快速软件实现方法。该方法在Intel i5、Intel i7和AMD R7环境下均能显著提升SM4算法的计算效率。但处理器为串行工作模式，且用软件实现SM4算法的方式在多任务场景下无法保证数据吞吐率的稳定性。第二种是用专用加密芯片实现，但该类方法流片费用高、实现周期长且更改难度大。第三种是用现场可编程门阵列（FPGA, field programmable gate array）实现。FPGA是一种内部电路可重构的芯片，可随时改变其内部连接关系，使之可以模拟处理器、显卡或其他并行计算模块。通过编程，FPGA内部可以实现SM4算法的硬件电路，既可避免处理器串行工作导致的效率低的问题及不同场景下性能不稳定的问题，又可避免采用专用芯片带来的流片费用高且流片后不可更改的问题。通常来讲，第三种方案在运行速率方面较第一种方案高，在灵活性方面较第二种方案高。鉴于FPGA的硬件化实现方式及可编程可重构易迭代的特性，采用FPGA实现SM4算法是一种高效可行的方式。

1　SM4算法现状分析

本文主要调研了基于FPGA实现SM4算法的方案。经过对类似方案的调研和归纳总结，通过FPGA实现SM4算法的设计方案大致可以分为2种类型。第一类是循环，这类设计适用于资源受限场景，吞吐量低。第二类是流水线，适用于性能优先场景，吞吐量高，此方案往往通过多级流水线架构实现，占用资源较多。

1.1　循环方案

Jin等[2]在Xilinx Virtex-4 FPGA设备上实现了SM4加密算法。在该方案中，加密模块每36个时钟完成一次数据加密操作，其中4个时钟用于缓存数据，32个时钟循环用于完成轮操作，该方案共消耗380个逻辑单元和4个随机存取存储器。Gao等[3]采用反馈循环架构，使用Moore型状态机控制整个系统的操作流程，对数据完成迭代转换。该方案使用寄存器存储预计算的轮密钥，减少了密钥扩展所需的时间和计算资源；采用反馈循环架构占用资源较少，但吞吐量较低。Guan等[4]使用Moore型状态机控制数据迭代过程，加密模块和密钥扩展模块同时运行，状态机在每个时钟内转换状态，控制加密模块和密钥扩展模块的运行。轮加密模块使用的密钥由前一状态的密钥扩展模块提供。文献[5]使用超大规模集成（VLSI, very large-scale integration）电路实现了循环架构。文献[6]使用流水线结构和乒乓协议优化了SM4密码算法的硬件实现。在何诗洋等[7]设计的方案中，轮密钥生成模块与加密模块并行执行。轮密钥生成模块产生的新一轮密钥传递给加密模块进行迭代计算，不需要额外的寄存器存储轮密钥，节约了硬件资源。此外，2个模块并行执行，当密钥更新后，轮密钥生成模块可实时扩展最新的轮密钥，加密模块无须等待。

1.2　流水线方案

Jin等[2]设计轮电路和电路采用了128个双端口块随机存取存储器（BRAM, block random access memory）存储32级流水线加密架构中所需的128套查找表式S盒，降低了查找表（LUT, look-up table）的消耗。Gao等[3]设计了流水线寄存器，实现了32级流水线加密架构。该方案中的轮密钥提前生成并存储在寄存器中。流水线架构可在一个时钟内处理32个数据块，使吞吐量最大化。Guan等[4]将32级流水线架构中流水线的规模缩减一半，将数据进行2轮迭代运算完成32次轮操作。在第1轮迭代中，明文输入流水线加密模块完成16次轮操作，并将流水线加密模块的输出作为输入，在第2轮迭代后再次完成16次轮操作，以完成数据加密过程。该方案通过减少流水线级数，降低了资源消耗。此外，该方案实现了Balance-4、Balance-8和Balance-16不同流水线级数的流水线架构，比较了不同流水线架构的资源消耗和性能表现。何诗洋等[7]提出了

3种不同的流水线设计架构：①基于LUT的流水线设计架构；②基于BRAM的流水线设计架构；③基于BRAM+REGISTER的流水线设计架构。第一种设计架构基于LUT实现，设计中使用了较大数量的LUT（7 466个），取得了较高的效率。第二种设计架构能够在BRAM用量增加不多且性能相差不大的前提下，大幅降低LUT资源的使用。第三种设计架构采用带有输出寄存器的BRAM，在增加部分寄存器和32个额外时钟时延的代价下，优化关键路径，提高运行频率和工程性能。文献[8]使用流水线结构实现SM4密码算法，并在方案中引入了混沌算法，增强了方案的安全性。文献[9]采用了流水线设计思想，在结构上对方案进行了优化。该方案采用多模组并行逻辑，调整单模组流水线步骤，使用先进先出（FIFO, first in first out）队列作中间缓存，增设流水线调度模块，支持模组全并行、分组并行和分时并行。文献[10]采用高层次综合（HLS, high-level synthesis）工具，将C语言综合作为底层的硬件设计，提出了3种优化方案（循环展开、数组优化和流水线优化），对SM4算法硬件进行实现与优化。

文献[2-4,7]均实现了循环和流水线架构。循环架构使用单个轮操作模块完成32次轮操作，占用资源少但数据吞吐率低，适用于资源受限的场景；流水线架构使用并行的32个轮操作模块完成32次轮操作，吞吐量高，但对资源的要求高，适用于对性能要求高且硬件资源充足的场景。多级流水线方案通常采用32级流水线架构，如图1所示。本文对文献[2-4,7]方案进行了对比分析，其优缺点对比如表1所示。

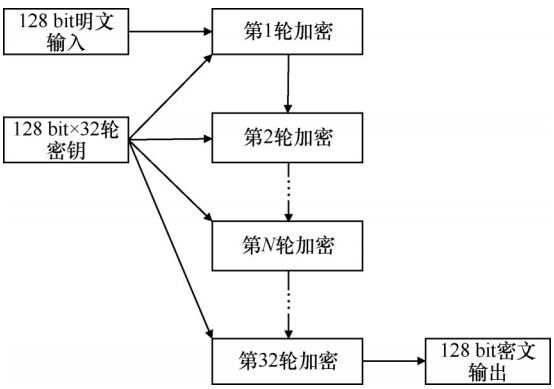


图1　32级流水线架构

表1　　文献[2-4,7]方案的优缺点对比

类型	优点	缺点
循环	资源消耗低	性能低
流水线	性能高	资源消耗高

由上述对比可知，基于FPGA实现SM4算法仍存在不足，难以实现效率和资源消耗的平衡：循环SM4算法数据处理效率低，而流水线线型SM4算法资源占用率过高。为此，本文采用优化后的代数式S盒和平衡型流水线架构，设计了基于FPGA的高性能、低资源消耗SM4加解密算法实现方案。首先，采用循环密钥扩展，对32级流水线进行加解密，降低了资源消耗，提高了数据处理速度；其次，采用代数式S盒，将线性运算合并，从不同不可约多项式生成的合并矩阵中筛选出最优矩阵以减少组合逻辑，进一步降低了资源消耗和运算量，提高了工程频率。测试结果显示，该方案比现有最佳方案在数据吞吐率上提升了43%，且资源占用率降低了10%。

2　SM4算法介绍

2.1　SM4算法组成

SM4算法是一种分组密码算法。数据分组长度为128 bit，密钥长度为128 bit。加密算法与密钥扩展算法都采用32轮迭代结构。SM4算法以字节（8位）和字（32位）为单位进行数据处理。SM4算法是对称运算，解密算法与加密算法类似，但轮密钥的使用顺序相反，解密算法的轮密钥是加密算法轮密钥的逆序[11-13]。

1) S盒变换

S盒变换是SM4加密流程的一个步骤，它是一种以字节为单位的非线性代替变换，其密码学的作用是起到一种非线性的混淆。S盒的输入和输出都是8位的字节，它是8位的非线性置换。设输入字节为a，输出字节为b，则S盒的运算可表示为b=Sbox(a)。S盒查找表如表2所示。假设S盒的输入为EF，则S盒的输出为表2中第E行与第F列交点处的值84，即Sbox(EF)=84。

2) 非线性变换τ

SM4算法的非线性变换（记作τ变换）是一种以字（32位）为单位的非线性代替变换。设输入字为A=(a_0,a_1,a_2,a_3)，输出字为B=(b_0,b_1,b_2,b_3)，则

$$B=\tau(A)=(\mathrm{Sbox}(a_0),\mathrm{Sbox}(a_1),\mathrm{Sbox}(a_2),\mathrm{Sbox}(a_3)) \tag{1}$$

3) 线性变换L

线性变换L以字为处理单位，其输入和输出都是32位的字。其密码学的作用是扩散。设L的输入字为B，输出字为C，则

$$C=L(B)=B\oplus(B<<<2)\oplus(B<<<10)\oplus(B<<<18)\oplus(B<<<24) \tag{2}$$

4) 合成变换T

合成变换T由非线性变换τ和线性变换L复合而成，数据处理的单位是字。设合成变换的输入为X，先对X进行τ变换，再进行L变换。

$$T(X)=L(\tau(X)) \tag{3}$$

合成变换T是非线性变换τ和线性变换L的复合，它起到了混淆和扩散的作用，提高了密码的安全性。

5) 轮加密函数

SM4算法采用对基本轮函数进行迭代的结构。SM4算法的轮函数是一种以字为处理单位的密码函数。设轮函数F的输入为(X_0, X_1, X_2, X_3)，4个32位的字，共128位。轮密钥为RK，RK也是一个32位的字。轮函数F的运算如下。

$$F(X_0,X_1,X_2,X_3,\mathrm{RK})=X_0\oplus T(X_1\oplus X_2\oplus X_3\oplus \mathrm{RK}) \tag{4}$$

2.2　密钥扩展算法

SM4 密码算法使用128位的加密密钥，并采用32轮迭代加密结构，每一轮加密使用一个32位的轮密钥，共使用32个轮密钥。轮密钥由主密钥经过密钥扩展得到。密钥扩展过程中所使用的系统参数如下：FK_0= (A3B1BAC6)，FK_1= (56AA3350)，FK_2= (677D9197)，FK_3= (B27022DC)。固定参数CK_0~CK_{31}为0x00070E15，0x1C232A31，0x383F464D，0x545B6269，0x70777E85，0x8C939AA1，0xA8AFB6BD，0xC4CBD2D9，0xE0E7EEF5，0xFC030A11，0x181F262D，0x343B4249，0x50575E65，0x6C737A81，0x888F969D，0xA4ABB2B9，0xC0C7EED5，0xDCE3EAF1，0xF8FF060D，0x141B2229，0x30373E45，0x4C535A61，0x686F767D，0x848B9299，0xA0A7AEB5，0xBCC3CAD1，0xD8DFE6ED，0xF4FB0209，0x10171E25，0x2C333A41，0x484F565D，0x646B7279。

设主密钥为MK(MK_0,MK_1,MK_2,MK_3)，输出轮密钥为RK_i，i=0,1,…,30,31，中间数据为K_i，i = 0,1,…,34,35，则密钥扩展算法可描述如下。

1) $(K_0,K_1,K_2,K_3)=(\mathrm{MK}_0\oplus \mathrm{FK}_0,\mathrm{MK}_1\oplus \mathrm{FK}_1,\mathrm{MK}_2\oplus \mathrm{FK}_2,\mathrm{MK}_3\oplus \mathrm{FK}_3)$

2) for i=0,1,…,30,31 do RK=$K_i\oplus T'(K_{i+1},K_{i+2},K_{i+3},\mathrm{CK}_i)$

其中，T'变换与加密算法轮加密函数中的T基本相同，只将其中的线性变换L修改为以下的L'

$$L'(B)=B\oplus(B<<<13)\oplus(B<<<23) \tag{5}$$

表2　　S盒查找表

输入高4位	输入低4位															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D6	90	E9	FE	CC	E1	3D	B7	16	B6	14	C2	28	FB	2C	05
1	2B	67	9A	76	2A	BE	04	C3	AA	44	13	26	49	86	06	99
2	9C	42	50	F4	91	EF	98	7A	33	54	0B	43	ED	CF	AC	62
3	E4	B3	1C	A9	C9	08	E8	95	80	DF	94	FA	75	8F	3F	A6
4	47	07	A7	FC	F3	73	17	BA	83	59	3C	19	E6	85	4F	A8
5	68	6B	81	B2	71	64	DA	8B	F8	EB	0F	4B	70	56	9D	35
6	1E	24	0E	5E	63	58	D1	A2	25	22	7C	3B	01	21	78	87
7	D4	00	46	57	9F	D3	27	52	4C	36	02	E7	A0	C4	C8	9E
8	EA	BF	8A	D2	40	C7	38	B5	A3	F7	F2	CE	F9	61	15	A1
9	E0	AE	5D	A4	9B	34	1A	55	AD	93	32	30	F5	8C	B1	E3
A	1D	F6	E2	2E	82	66	CA	60	C0	29	23	AB	0D	53	4E	6F
B	D5	DB	37	45	DE	FD	8E	2F	03	FF	6A	72	6D	6C	5B	51
C	8D	1B	AF	92	BB	DD	BC	7F	11	D9	5C	41	1F	10	5A	D8
D	0A	C1	31	88	A5	CD	7B	BD	2D	74	D0	12	B8	E5	B4	B0
E	89	69	97	4A	0C	96	77	7E	65	B9	F1	09	C5	6E	C6	84
F	18	F0	7D	EC	3A	DC	4D	20	79	EE	5F	3E	D7	CB	39	48

2.3 SM4 加密算法设计

SM4 加密算法采用 32 轮迭代结构，每轮使用一个轮密钥。设输入明文为(X_0,X_1,X_2,X_3)，共 4 个字，128 位。输入轮密钥为 RK_i，$i=0,1,\cdots,31$，共 32 个字。输出密文为(Y_0,Y_1,Y_2,Y_3)，共 4 个字，128 位。加密算法可描述为 $X_{i+4}=F(X_i,X_{i+1},X_{i+2},X_{i+3},\mathrm{RK}_i)=X_i\oplus T(X_{i+1}\oplus X_{i+2}\oplus X_{i+3}\oplus \mathrm{RK}_i)$，$i=0,1,\cdots,31$。之后完成反序变换 R：$R(Y_0,Y_1,Y_2,Y_3)=(X_{35},X_{34},X_{33},X_{32})$，得到密文。SM4 算法加密过程如图 2 所示。

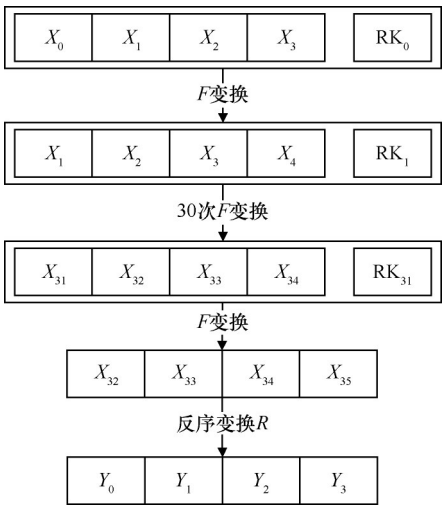


图 2 SM4 算法加密过程

2.4 SM4 解密算法设计

SM4 算法是对称运算，解密算法与加密算法步骤相同，但轮密钥的使用顺序相反，解密轮密钥是加密轮密钥的逆序。设输入密文为(Y_0,Y_1,Y_2,Y_3)，输入轮密钥为 RK_i，$i=31,30,\cdots1,0$，输出明文为(M_0,M_1,M_2,M_3)，为了方便与加密算法对照，在解密算法中仍然采用 X_i 表示密文。于是可得到如下的解密算法 $X=F(X_{i+4},X_{i+3},X_{i+2},X_{i+1},\mathrm{RK}_i)=X_{i+4}\oplus T(X_{i+3}\oplus X_{i+2}\oplus X_{i+1}\oplus \mathrm{RK}_i)$，$i=31,30,\cdots,1,0$，之后进行反序变换 R：$R(M_0,M_1,M_2,M_3)=(X_3,X_2,X_1,X_0)$，得到密文。

3 高性能方案设计

SM4 算法可拆分为不同层次的模块在 FPGA 中进行实现，并通过设计控制器对运行状态、顺序进行监控，完成密钥扩展和加解密操作。加解密模块中 S 盒置换的部分采用代数式 S 盒设计，整体设计架构采用多级流水线平衡型架构。

3.1 代数式 S 盒设计

本文采用复合域代数式 S 盒，并通过合并线性变换的方法减少代数式 S 盒的计算量，从而降低了 S 盒的资源消耗，通过在不同运算模块之间设置寄存器形成流水线的方式提高 S 盒的运行频率。

1) 复合域代数式 S 盒设计

基于 Liu 等[14]提出的基于有限域求逆构造 S 盒的理论，本文将其 S 盒构造方法融入用 FPGA 实现 SM4 算法的方案中，从而提高方案的整体效率。复合域代数式 S 盒计算过程如图 3 所示。该过程由求逆仿射变换和仿射变换组成，其代数结构为

$$\mathrm{Sbox}(a)=I(a\boldsymbol{A}_1+\boldsymbol{C}_1)\boldsymbol{A}_2+\boldsymbol{C}_2 \tag{6}$$

其中，a 为 S 盒的 8 bit 输入，I 为 $\mathrm{GF}(2^8)$ 上的乘法求逆，在复合域上进行乘法求逆，可以简化运算复杂度。$I(x)$ 对应的 8 次不可约多项式为 $f(x)=x^8+x^7+x^6+x^5+x^4+x^2+1$。循环矩阵 $\boldsymbol{A}_1,\boldsymbol{A}_2\in\mathrm{GL}(8,2)$，行向量 $\boldsymbol{C}_1$，$\boldsymbol{C}_2\in\mathrm{GF}(2^8)$。循环矩阵 $\boldsymbol{A}_1,\boldsymbol{A}_2$ 和向量 $\boldsymbol{C}_1,\boldsymbol{C}_2$ 分别为

$$\boldsymbol{A}_1=\boldsymbol{A}_2=\begin{bmatrix}1&1&1&0&0&1&0&1\\1&1&1&1&0&0&1&0\\0&1&1&1&1&0&0&1\\1&0&1&1&1&1&0&0\\0&1&0&1&1&1&1&0\\0&0&1&0&1&1&1&1\\1&0&0&1&0&1&1&1\\1&1&0&0&1&0&1&1\end{bmatrix}$$

$$\boldsymbol{C}_1=\boldsymbol{C}_2=(1,1,0,0,1,0,1,1)$$

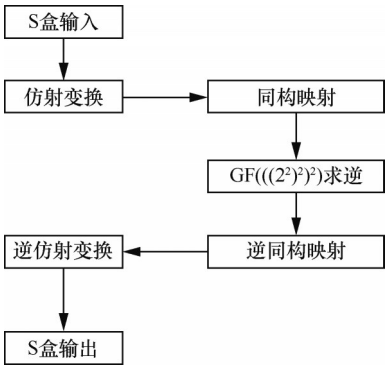


图 3 复合域代数式 S 盒计算过程

仿射变换容易实现，因此主要计算难度在于 $\mathrm{GF}(2^8)$ 的求逆问题。传统的方法是利用扩展欧几里得算法进行计算，但这种方式复杂且计算量巨大，不适用于硬件电路实现。使用复合域求逆在运算量和硬件资源消耗上少，适合用于硬件实现。本文根据有限域性质，通过同构变换，将 $\mathrm{GF}(2^8)$ 的有限域运算转换到复合域 $\mathrm{GF}(((2^2)^2)^2)$ 上进行。图 4 展示了

复合域乘法求逆过程：①将GF(2^8)上的元素经过同构映射矩阵映射到GF((((2^2)^2)^2)^2)；②在GF((((2^2)^2)^2)域上进行求逆运算；③求逆的结果经过①中逆矩阵射回。

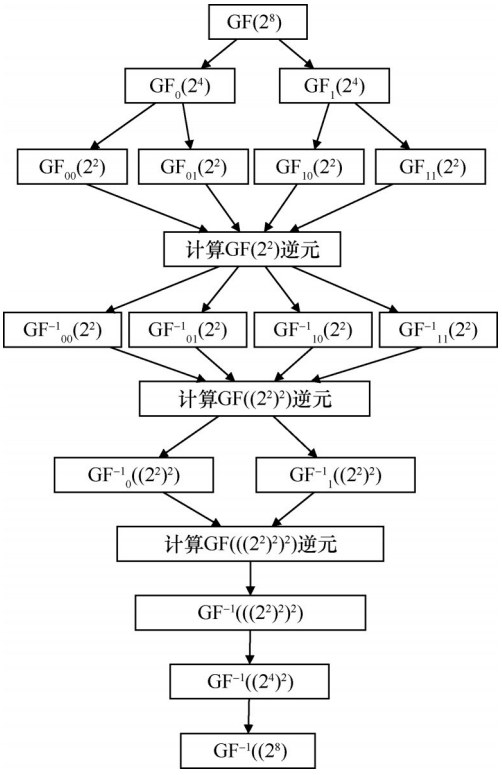


图4　复合域乘法求逆过程

2) 同构映射和仿射变换矩阵的合并

代数式S盒的计算过程由2次线性仿射变换和一次非线性的有限域求逆组成，其代数结构如式(6)所示。同构映射和仿射变换进行合并，逆同构映射和逆仿射变换进行合并，进一步优化计算效率以及降低资源的消耗。将同构映射矩阵和逆同构映射矩阵分别定义为$\boldsymbol{T}_1$和$\boldsymbol{T}_2$，则有Sbox(a)=I(($a\boldsymbol{A}_1$+$\boldsymbol{C}_1$)$\boldsymbol{T}_1$)$\boldsymbol{T}_2\boldsymbol{A}_2$+$\boldsymbol{C}_2$，继而有Sbox($a$)=$I$($a\boldsymbol{A}_1\boldsymbol{T}_1$+$\boldsymbol{C}_1\boldsymbol{T}_1$)$\boldsymbol{T}_2\boldsymbol{A}_2$+$\boldsymbol{C}_2$。将$\boldsymbol{A}_1\boldsymbol{T}_1$记为矩阵$\boldsymbol{M}_1$，$\boldsymbol{C}_1\boldsymbol{T}_1$记为$\boldsymbol{N}_1$，$\boldsymbol{T}_2\boldsymbol{A}_2$记为$\boldsymbol{M}_2$，则有

$$\mathrm{Sbox}(a) = I(a\boldsymbol{A}_1\boldsymbol{T}_1 + \boldsymbol{C}_1\boldsymbol{T}_1)\boldsymbol{T}_2\boldsymbol{A}_2 + \boldsymbol{C}_2 \quad (7)$$

记$a\boldsymbol{M}_1$+$\boldsymbol{N}_1$为合并运算1，即同构映射和仿射变换的合并，$a\boldsymbol{M}_2$+$\boldsymbol{C}_2$为合并运算2，即逆同构映射和逆仿射变换的合并，则有优化后的代数式S盒运算过程如图5所示。

3) 筛选最优合并矩阵

同构映射的过程中，使用正则基可将GF(2^8)上的元素表示为GF(2^4)上的一次线性多项式：$g(y)$=(a_1Y^{16}+a_0Y)，此时乘法运算需要模不可约多项式$r(y)$=y^2+τy+η，[Y^{16},Y]称为该域下的一组正则基，[Y^{16},Y]为$r(y)$=0的2个根。GF(2^4)上的元素可表示为GF(2^2)上的一次线性多项式：$a(z)$=(b_1Z^4+b_0Z)，此时的乘法运算需要模不可约多项式$t(z)$=z^2+μz+ρ。[Z^4,Z]为$t(z)$=0的2个根。

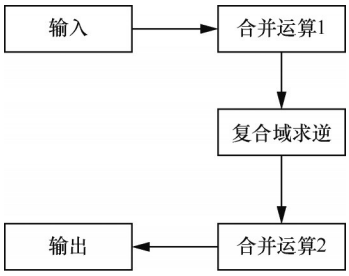


图5　优化后的代数式S盒运算过程

GF(2^2)上的元素可表示为GF(2)上的一次线性多项式：$b(w)$=(c_1W^2+c_0W)，其乘法需要模不可约多项式$s(w)$=w^2+w+1。[W^2,W]为$s(w)$=0的2个根。从GF(8)到GF((((2^2)^2)^2)的逆同构映射矩阵为[$Y^{16}Z^4W^2$, $Y^{16}Z^4W$, $Y^{16}ZW^2$, $Y^{16}ZW$, YZ^4W^2, YZ^4W, YZW^2, YZW]。根据不可约多项式$r(y)$=y^2+τy+η，$t(z)$=z^2+μz+ρ，$s(w)$=w^2+w+1所选参数不同，可生成多个不同的根，有不同的同构映射矩阵以及逆同构映射矩阵，故有不同的合并矩阵$\boldsymbol{M}_1$、$\boldsymbol{N}_1$、$\boldsymbol{M}_2$及不同的合并运算1和合并运算2。

将优化后的代数式S盒在FPGA上实现后，根据不同的合并运算1和合并运算2实现代数式S盒，有不同的硬件资源消耗。本文使用Python统计了所有可正确输出S盒的结果参数组合，得到了8种参数组合。将8种可输出正确结果的组合在FPGA上做了实现，分析了不同组合最后实现所需的硬件资源消耗，如表3所示。当参数[W^2,W]、[Z^4,Z]、[Y^{16},Y]为[0X5C,0X5D]、[0XC,0XD]、[0XBF,0XBE]时，资源消耗最少，为32个LUT。由此得到的合并运算1为

$$\begin{bmatrix} 0&0&0&0&1&0&0&1\\ 0&0&0&1&0&1&0&1\\ 1&1&0&1&0&0&1&1\\ 1&0&1&0&1&0&1&1\\ 0&0&0&0&1&1&1&0\\ 1&1&1&0&0&1&1&0\\ 1&0&1&1&1&0&1&0\\ 0&0&0&1&0&0&0&0 \end{bmatrix}$$

表 3 S 盒资源消耗

W^2,W	Z^4,Z	Y^{16},Y	资源占用/个 LUT
0x5C,0x5D	0xC,0xD	0xBE,0xBF	35
0x5C,0x5D	0xC,0xD	0xBF,0xBE	32
0x5C,0x5D	0xD,0xC	0xEE,0xEF	86
0x5C,0x5D	0xD,0xC	0xEF,0xEE	86
0x5D,0x5C	0x50,0x51	0x94,0x95	33
0x5D,0x5C	0x50,0x51	0x95,0x94	35
0x5D,0x5C	0x51,0x50	0x98,0x99	85
0x5D,0x5C	0x51,0x50	0x99,0x98	86

合并运算 2 为

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

最优的代数式 S 盒相比于查找表式 S 盒 LUT 消耗（40 个，参考 EDA 软件综合结果），占用的硬件资源减少 20%。根据该参数组合在 FPGA 实现的代数式 S 盒可减少运算量，减少基于 FPGA 实现 SM4 算法的方案整体的硬件资源消耗以及提高方案整体的效率。

4) 流水线 S 盒设计

单个时钟内进行大量的逻辑运算会导致系统时钟频率较低。在组合逻辑较为复杂的时候，最大计算时延路径限制了工程的最大时钟频率，降低了单个时钟内运算量可以提高电路的时钟频率。将代数式 S 盒内的运算单元进行拆解后插入寄存器以建立流水线，进而构造流水线代数式 S 盒，可以提高代数式 S 盒运算模块的时钟频率。

流水线代数式 S 盒结构如图 6 所示。将代数式 S 盒的运算分为三大模块，包括合并运算 1 模块、复合域求逆运算模块和合并运算 2 模块。通过在不同计算模块之间插入寄存器形成流水线，缩短了整体电路的关键路径，减少了因为复杂的组合逻辑设计造成的电路时延，降低了电路中每个时钟内的运算量，提高了本文方案的整体频率。

5) 与查找表式 S 盒对比

从工程易用性、资源占用和运行效率 3 个方面对查找表式 S 盒和复合域代数式 S 盒进行比较，结果如表 4 所示。在工程易用性方面，查找表式 S 盒只进行一次查找计算，原理简单，便于理解与工程集成；复合域代数式 S 盒计算复杂且涉及复合域相关知识，工程易用性一般。在资源占用方面，查找表式 S 盒占用 40 个 LUT，复合域代数式 S 盒只占用 32 个 LUT，因此复合域代数式 S 盒更节省逻辑资源，相比于查找表式 S 盒逻辑资源节省 20%。在运行效率方面，查找表式 S 盒运算量少，可运行在较高频率，满足大多数场景的频率要求；复合域代数式 S 盒计算复杂，但由于计算透明，与轮加密其他逻辑计算结合，选择合理位置进行寄存器插入，可使整体工程运行在更高频率。因此在工程应用上，如果没有明确约束，为了追求工程的简易性，查找表式 S 盒是常用的方式，如果资源占用和运行效率要求较高，复合域代数式 S 盒则具有明显优势。

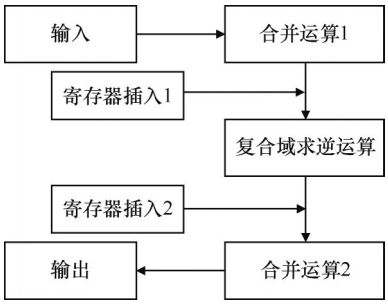


图 6 流水线代数式 S 盒结构

表 4 2 种 S 盒对比

S 盒类型	工程易用性	资源占用/个 LUT	运行效率/MHz
查找表式 S 盒	高	40	283
复合域代数式 S 盒	中	32	342

3.2 流水线架构设计

为了减少硬件资源消耗，本文在密钥扩展阶段采用了循环密钥扩展方式。为了提升本文方案的性能，本文进行了流水线架构的设计。SM4 算法的加解密过程是由 32 轮完全相同的轮函数组成，通过采用流水线结构来提高系统的数据吞吐率。本文方案采取循环密钥扩展架构和多级流水线加解密架构相结合的方式，设计了一个多级流水线平衡架构，如图 7 所示。设计该架构的原因如下。

循环架构面向资源节约优化。密钥扩展模块采用循环架构设计，使用一个模块迭代使用 32 次进

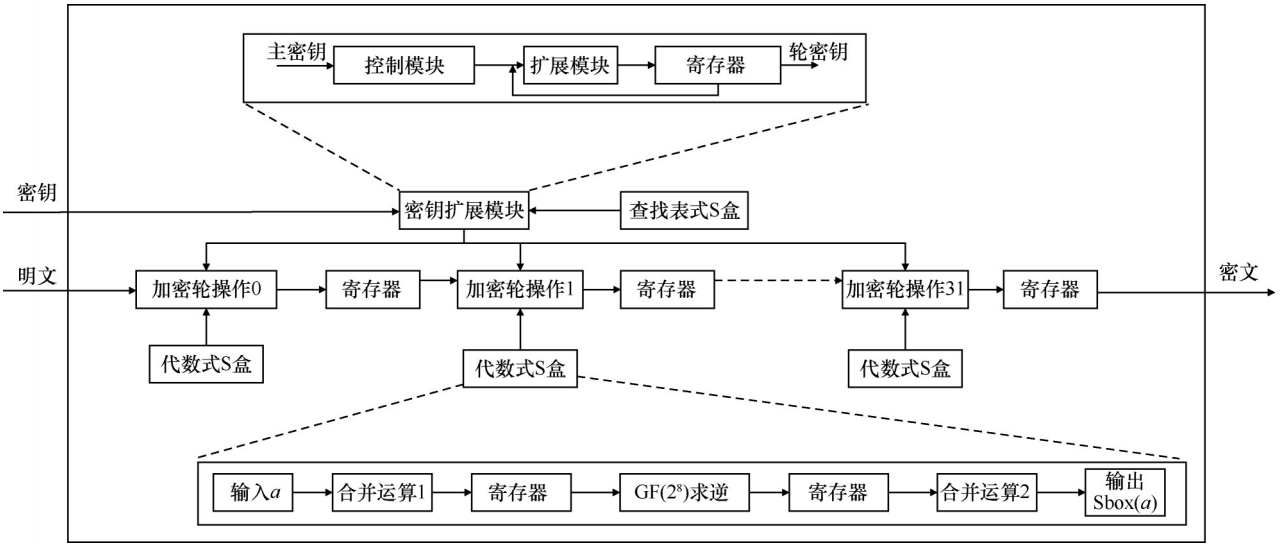


图7　多级流水线平衡架构

行32次轮操作获得32个轮密钥，减少硬件资源的使用。其中S盒置换部分采用查找表式S盒实现，减少密钥扩展需要的时钟个数，降低密钥扩展阶段所带来的时延。

流水线架构面向性能优化，主要体现在：1) 增加了轮密钥寄存器，轮密钥扩展模块将32个轮密钥存储在寄存器中，每个时钟轮密钥扩展模块都可以向加密模块输入当前轮密钥，保证流水线加密不间断；2) 流水线加密模块实例化32个轮加密模块，每个轮加密模块间插入寄存器以形成32级流水线加密架构；3) 在S盒置换模块中，采用流水线代数式S盒，提高方案整体的频率。

查找表式S盒硬件资源消耗多，完成S盒置换消耗时钟少，1个时钟可完成S盒置换。流水线代数式S盒消耗硬件资源低，频率高，但完成S盒置换需要消耗2个时钟。实际生活中，密钥扩展功能使用次数少，加密功能使用次数多，需要减少密钥扩展的时延，提高加密的效率。本文实现密钥扩展模块时，采用循环架构结合查找表式S盒的设计，快速完成密钥扩展，减少硬件资源消耗。实现加密模块时，本文采用32级流水线加密架构结合流水线代数式S盒的设计，提高了加密频率和数据吞吐率。

通过循环密钥扩展架构结合查找表式S盒，流水线加密架构结合流水线代数式S盒的方法，本文提高了整体方案的加密频率和吞吐量，降低了整体方案的硬件资源消耗。

4　FPGA方案实现

4.1　顶层架构设计

在本文方案中，硬件平台采用当下主流的FPGA器件，型号为Xilinx Zynq 7100，综合工具软件选用了配套的Xilinx Vivado 2019.1版本工具。高性能SM4加密模块顶层设计如图8所示。输入信号包括Clk_sys、Rst_n、Din[127:0]、Key[127:0]、De_sel、En_mode等信号。其中，Clk_sys为系统工作时钟，Rst_n为异步释放的同步复位信号，Din[127:0]为输入数据，Key[127:0]为输入密钥，De_sel为信号加密解密选择信号，En_mode为加密模式选择信号，支持电码本（ECB, electronic codebook book）、密码块链接（CBC, cipher block chaining）模式。输出信号包括Data_en、Dout[127:0]信号。其中，Data_en为数据使能标志位，1表示数据加密有效；Dout[127:0]为加密输入信号，配合Data_en信号使用。

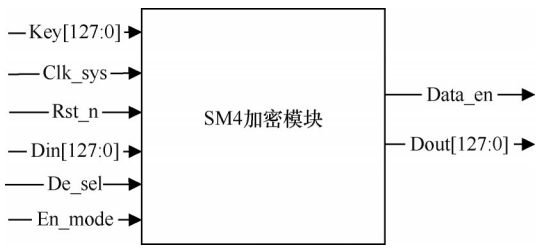


图8　SM4加密模块顶层设计

4.2　内部结构设计

本文方案的顶层内部设计包括密钥扩展模块和

32级流水线结构，顶层代码结构如图9所示。每一级轮加密运算模块内部有代数式S盒模块，代数式S盒代码结构如图10所示。S盒的内部设计包括合并运算和复合域求逆。复合域求逆又进行了相应的功能分解与实现。

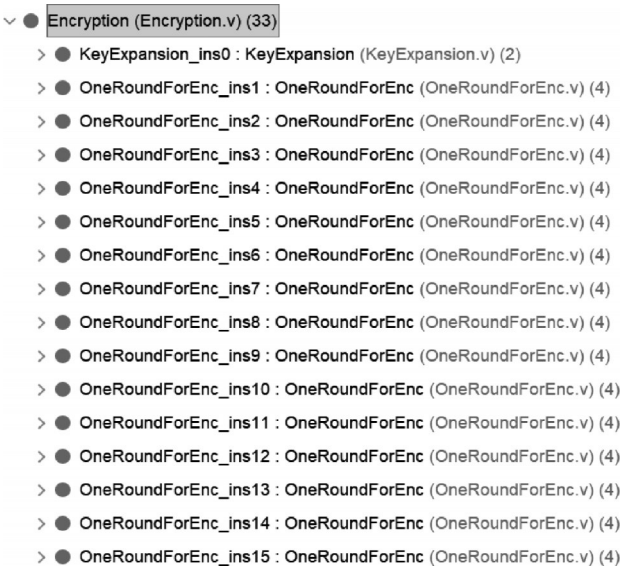


图9　顶层代码结构

图10　代数式S盒代码结构

5　结果测试与分析

在本文方案的测试分析中，本文采用Vivado 2019.1作为综合与实现工具。硬件平台采用Xilinx Zynq 7100硬件组成。

5.1　流水线S盒测试结果

首先，本文测试分析了寄存器插入位置对代数式S盒频率的影响。不插入寄存器时，由于单个时钟内的运算逻辑量较大，最大频率只能运行在190 MHz。本文对图7所示的架构中不同位置插入寄存器进行了测试，结果如表5所示。在插入两级寄存器的情况下，最大频率可以达到390 MHz。由于S盒最终会运行在整体工程中，结合工程内其他逻辑部分的运行进行分析，考虑资源占用问题，342 MHz可作为最大的系统运行频率。

表5　寄存器插入结果测试

寄存器插入位置	最大工作频率/MHz	逻辑资源消耗/个LUT
不插入寄存器	190	6 610
合并运算1模块后	230	7 250
GF(2^8)上求逆模块后	280	6 480
合并运算2模块后	230	7 377
合并运算1模块后、GF(2^8)上求逆模块后	390	8 709
GF(2^8)上求逆模块后、合并运算2模块后	280	6 738
合并运算1模块后、合并运算2模块后	280	7 250

5.2　系统测试结果

基于5.1节的系统工作频率测试结果，本文采用最优的代数式S盒进一步评估了本文方案性能并与其他已有方案进行了对比。图11为Vivado软件下基于本文方案开发的工程仿真波形，图中显示在循环密钥扩展完成后，32级流水线加密模块完成流水线初始化后输出第一组密文，此后进入连续加密状态，后续每个时钟均可输出一组密文。

为了更好地分析本文方案的性能，现选择文

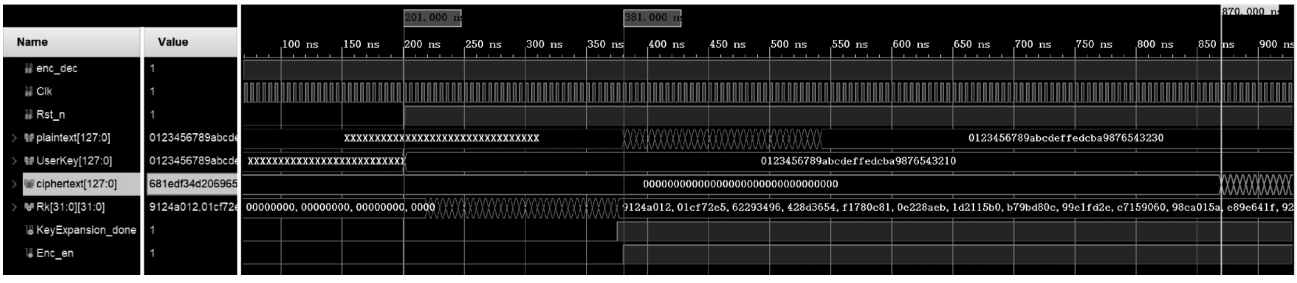


图11　Vivado仿真波形

表6　　方案整体测试结果

方案		资源占用/个LUT	系统频率/MHz	数据吞吐率/(Gbit·s^{-1})
文献[4]	流水线架构	7 667	212	27.15
	循环架构	687	210	0.82
文献[7]	流水线架构	7 466	250	30.52
	循环架构	697	333	1.27
本文方案	效率优先	8 709	390	49.92
	资源优先	6 740	342	43.77

献[4]和文献[7]进行对比。文献[4]和文献[7]中的2个方案均分别设计了循环架构实现方案和流水线架构实现方案。从表6所示的对比结果可以发现，在资源占用方面，本文方案表现良好，相较于传统的流水线架构方案[4,7]节省10%的LUT；在系统频率方面，本文方案明显优于传统的流水线架构方案[4,7]；在数据吞吐率方面，本文方案较文献[4]和文献[7]中的方案优势显著，比其最优方案提升43%。

综合来看，本文方案在逻辑资源消耗和性能方面取得了较好的平衡，逻辑资源消耗低，同时系统频率和数据吞吐率均得到了明显提升。

6　结束语

针对当前SM4算法实现效率低、资源消耗大的问题，本文提出了一个高性能、低资源消耗的基于FPGA的SM4算法实现方案。该方案采用循环密钥扩展，32级流水线进行加解密，降低了资源消耗，提高了数据处理速度。另外，采用代数式S盒，将线性运算合并，从不同不可约多项式生成的合并矩阵中筛选出最优矩阵以减少组合逻辑，进一步降低了资源消耗和运算量，提高了工程频率。基于主流的硬件平台Xilinx Zynq 7100，配合Xilinx Vivado 2019.1版本工具，对本文方案进行了验证。相较于已有方案，本文方案在数据吞吐率上提升了43%，资源占用率降低了10%，展示了先进性。

参考文献：

[1] 陈晨, 郭华, 王闯, 等. 一种基于复合域的国密SM4算法快速软件实现方法[J]. 密码学报, 2023, 10(2): 289-305.
CHEN C, GUO H, WANG C, et al. A fast software implementation of SM4 based on composite fields[J]. Journal of Cryptologic Research, 2023, 10(2): 289-305.

[2] JIN Y E, SHEN H B, YOU R Q. Implementation of SMS4 block cipher on FPGA[C]//Proceedings of the 2006 First International Conference on Communications and Networking in China. Piscataway: IEEE Press, 2006: 1-4.

[3] GAO X W, LU E H, XIAN L Q, et al. FPGA implementation of the SMS4 block cipher in the Chinese WAPI standard[C]//Proceedings of the 2008 International Conference on Embedded Software and Systems Symposia. Piscataway: IEEE Press, 2008: 104-106.

[4] GUAN Z Y, LI Y H, SHANG T, et al. Implementation of SM4 on FPGA: trade-off analysis between area and speed[C]//Proceedings of the 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR). Piscataway: IEEE Press, 2018: 192-197.

[5] YAN W W, YOU K D, HAN J, et al. Low-cost reconfigurable VLSI implementation of the SMS4 and AES algorithms[C]//Proceedings of the 2009 IEEE 8th International Conference on ASIC. Piscataway: IEEE Press, 2009: 135-138.

[6] CHENG H, ZHAI S, FANG L, et al. Improvements of SM4 algorithm and application in Ethernet encryption system based on FPGA[J]. Journal of Information Hiding and Multimedia Signal Processing, 2014, 5(3): 518-526.

[7] 何诗洋, 李晖, 李凤华. SM4算法的FPGA优化实现方法[J]. 西安电子科技大学学报, 2021, 48(3): 155-162.
HE S Y, LI H, LI F H. Optimization and implementation of the SM4 on FPGA[J]. Journal of Xidian University, 2021, 48(3): 155-162.

[8] 窦玉超. SM4算法优化及其密钥扩展算法的设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2021.
DOU Y C. Design and implementation of SM4 algorithm optimization and key expansion algorithm[D]. Harbin: Harbin Institute of Technology, 2021.

[9] 王凯, 刘凯, 李拓, 等. 可重构高速数据加密系统设计和实现[J]. 电子测量技术, 2021, 44(19): 8-15.
WANG K, LIU K, LI T, et al. Design and implementation of reconfigurable high-speed data encryption system[J]. Electronic Measurement Technology, 2021, 44(19): 8-15.

[10] 申懿鑫, 韩跃平, 唐道光. 高层次综合的SM4算法硬件实现与优化[J]. 单片机与嵌入式系统应用, 2023, 23(8): 11-14.
SHEN Y X, HAN Y P, TANG D G. Hardware implementation and optimization of SM4 algorithm based on high-level synthesis[J]. Microcontrollers & Embedded Systems, 2023, 23(8): 11-14.

[11] 中华人民共和国国家互联网信息办公室. 中华人民共和国密码法[EB/OL]. (2019-10-26) [2021-03-20].

The State Internet Information Office of the People's Republic of China. Cryptographic law of the People's Republic of China[EB/OL]. (2019-10-26)[2021-03-20].

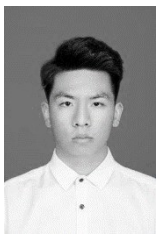
[12] 国家密码管理局. 无线局域网产品使用的SMS4密码算法[EB/OL]. (2016-11-18)[2021-03-20].
State Cryptography Administration. SMS4 cryptographic algorithm for wireless LAN products [EB/OL]. (2016-11-18)[2021-03-20].

[13] ISO. ISO/IEC 18033-3:2010/AMD 1:2021 Information technology security techniques—encryption algorithms—part 3: block ciphers-amendment 1:SM4[S]. 2021.

[14] LIU F, JI W, HU L, et al. Analysis of the SMS4 block cipher[C]//Australasian Conference on Information Security and Privacy. Berlin: Springer, 2007: 158-170.

[作者简介]

张宏科（1988– ），男，河北石家庄人，西安电子科技大学博士生、中国电子科技集团公司第五十四研究所高级工程师，主要研究方向为网络空间安全、嵌入式与通信系统、通信内生安全等。



袁浩楠（1999– ），男，河南郑州人，西安电子科技大学硕士生，主要研究方向为网络空间安全、区块链等。

丁文秀（1989– ），女，湖北仙桃人，博士，西安电子科技大学副教授、硕士生导师，主要研究方向为云计算安全、隐私保护、访问控制等。

闫峥（1972– ），女，安徽亳州人，博士，西安电子科技大学教授、博士生导师，主要研究方向为信任管理、可信AI、隐私保护、网络安全等。

李斌（1972– ），男，河北邢台人，中国电子科技集团公司第五十四研究所研究员级高级工程师、硕士生导师，主要研究方向为通信集成电路设计、硅基集成电路设计等。

梁栋（1986– ），男，山东泰安人，中国电子科技集团公司第五十四研究所高级工程师，主要研究方向为人工智能芯片、微系统集成、卫星通信等。