中国科学 : 信息科学 2025年 第 55 卷 第 1 期: 64–79
SCIENTIA SINICA Informationis
论文

面向窄带物联网的 ZUC 算法紧凑硬件实现

宋锐[1], 向泽军[1,2*], 张若琳[3], 张莎莎[1], 陈思维[1]

1. 湖北大学网络空间安全学院, 武汉 430062
2. 密码科学技术全国重点实验室, 北京 100878
3. 湖北大学数学与统计学学院, 武汉 430062
* 通信作者. E-mail: xiangzejun@hubu.edu.cn

收稿日期: 2024–10–10; 修回日期: 2024–11–16; 接受日期: 2024–11–18; 网络出版日期: 2024–12–31

国家自然科学基金 (批准号: 62272147)、湖北省自然科学基金 (批准号: 2024AFB573) 和国家重点研发计划青年科学家项目 (批准号: 2023YFA1011200) 资助

摘要 物联网 (Internet of Things, IoT) 领域当前正面临着无法回避且持续存在的网络安全威胁以及设备资源受限的双重挑战. 针对前述问题, 本文在 ASIC (application specific integrated circuit) 平台上, 利用时序复用与门控时钟技术, 设计了一种高效的低面积 ZUC 算法硬件实现电路. 此电路通过确保每个功能模块仅被实例化一次, 实现了电路面积的极小化. 在 S 盒的设计上, 本文借鉴了塔域分解的思想, 并提出了一种算法, 用于在有限域 $\mathbb{F}_{2^n}$ 到有限域 $\mathbb{F}'_{2^n}$ 之间搜索同构映射矩阵. 该算法旨在找到一种同构映射, 当它与 S 盒运算的仿射矩阵及其他相关矩阵相乘后, 能够以最少的异或逻辑门数实现映射. 基于上述两点, 本文所实现的 S1-box 在面积上与当前 AES 算法的 Sbox 相当. 在线性变换部分, 本文采用了最大距离可分 (maximum distance separable, MDS) 矩阵拆解的思想, 使得整个线性层的实现仅需 164 个异或门. 在加法链的设计上, 本文采用了进位存储加法器、32 比特加法器、单加数的 31 比特加法器与中间寄存器的组合. 这一设计使得线性反馈移位寄存器层与有限状态自动机层能够共享同一条加法链, 从而进一步优化了电路结构. 在 TSMC 90 nm 工艺下综合验证, 本文所提出的硬件实现方案在时钟频率为 250 MHz 时, 吞吐率可达 2 Gbps, 同时面积开销仅为 6.67 kGE. 与当前主流方案相比, 本设计在保持吞吐率不变的前提下, 面积开销降低了 44%.

关键词 ZUC, 面积优化, 复合域, 时序复用, 窄带物联网

1 引言

物联网秉持万物互联的核心理念, 正以前所未有的速度推动各行各业的数字化转型与智能化升级. 物联网 (Internet of Things, IoT) 的核心在于构建一个由多样化设备、精密传感器及智能对象紧密交织而成的互联网络体系, 该体系能够跨越互联网边界, 实现数据之间的无缝流通与高效交互. 从当前学术界与工业界的广泛视角审视, IoT 已在汽车制造、医疗健康、教育领域以及智能家居等多个垂直

引用格式: 宋锐, 向泽军, 张若琳, 等. 面向窄带物联网的 ZUC 算法紧凑硬件实现. 中国科学: 信息科学, 2025, 55: 64–79, doi: 10.1360/SSI-2024-0299
Song R, Xiang Z J, Zhang R L, et al. Compact hardware implementation of ZUC algorithm for NB-IoT. Sci Sin Inform, 2025, 55: 64–79, doi: 10.1360/SSI-2024-0299

行业内展现出举足轻重的作用与深远影响 [1~4]. 预计到 2033 年, 全球 IoT 设备部署将激增至惊人的 396 亿件, 广泛覆盖垂直行业及其他领域 [5]. 届时, IoT 可能对全球经济产生 5.5 万亿至 12.6 万亿美元的影响 [6]. 窄带物联网 (NarrowBand-IoT, NB-IoT) 是作为第三代合作伙伴计划 (the 3rd generation partnership project, 3GPP) 制定的一项低功耗广域网 (low-power wide-area network, LPWAN) 技术, 凭借其卓越的特性, 包括支持海量低速率设备的接入、对时延的低敏感度、设备成本的超低门槛以及低功耗表现, 展现出极为广阔的发展前景 [7].

在 5G 的背景下, 国际通信联盟为 5G 定义了三大应用场景: 增强型移动带宽、超可靠低时延通信、海量机器类通信 (massive machine type communication, mMTC) [8]. 而 NB-IoT 正是专为 mMTC 场景设计的 LPWAN 技术, 它强调实现端到端的安全性, 该需求亟须融入可信赖的安全保障与身份验证特性, 从而有效抵抗 IoT 在数据安全与传输可靠性上面临的风险 [9,10]. 而以 ZUC 算法为核心的机密性算法 EEA-3 和完整性算法 EIA-3 , 精准契合了上述安全需求. 在 IoT 的框架下, 用于保护感知数据的密码算法必须适应资源有限的嵌入式设备的需要. 故设计一种专注于面积优化的 ZUC 算法就显得尤为必要, 它更易于长期运行在资源有限的环境中. 然而, 当前关于 ZUC 算法的研究文献普遍倾向于对其时延和吞吐率等性能指标进行深入探讨. 例如, 文献 [11~13] 均采用了多级流水线技术, 这一策略虽显著提升了算法的工作频率, 但随之而来的是中间寄存器资源开销的大幅增加, 这无疑与 IoT 设备资源受限的特性相悖. 在 IoT 系统芯片的设计考量中, Cavo 等 [14] 和 Sharaf 等 [15] 利用 ZUC 算法与 SNOW3G 算法的相似性, 通过硬件重用技术减少了 ZUC 算法的面积开销, 但 S-box 并未采纳复合域实现的方案来进一步缩减面积.

基于上述因素, 本文对 ZUC 算法的核心组件进行了一定程度上的面积优化, 具体体现在以下几个方面: (1) 电路整体的极小化: 每个功能模块仅被实例化一次; (2) S 盒的高效实现: 本文提出了两种 S 盒的塔域分解方案, 该方案在复合域上实现了 S 盒功能, 并显著降低了所需的电路面积; (3) 线性变换 L 的精简: 本文以当前最少的异或门数量实现了线性变换 L 的功能. (4) 加法链的优化: 整个 ZUC 算法共享一条加法链, 并进一步缩短了有限状态自动机层的关键路径.

2　背景知识

2.1　符号说明

$\bullet_{\mathrm{H}}$: 任意比特长字符串 $\bullet$ 的高 16 位.

$\bullet_{\mathrm{L}}$: 任意比特长字符串 $\bullet$ 的低 16 位.

$\bullet_{[m:n]}$: 由任意比特长字符串 $\bullet$ 的第 n 到 m 位构成的新字符串.

$\oplus$: 异或操作运算符.

$S()$: S 盒变换.

$\lll n$: 循环左移 n 比特运算符.

$\gg n$: 右移 n 比特运算符.

$\boxplus$: 模 2^{32} 加运算符.

$\neg$: 比特翻转运算符.

Z: ZUC 算法输出的密钥流.

$\mathbb{F}_2$: 包含 0 和 1 两个元素的有限域.

$\mathbb{F}_{2^k}$: 包含 2^k 个元素的有限域.

2.2　ZUC 算法

ZUC-256 的输入为 256 比特长的密钥和 184 比特长的初始向量 (initial vector, IV), 输出为一串 32 比特长的密钥流. 如图 1 所示, 该算法由三部分组成. 第一部分是线性反馈移位寄存器 (linear

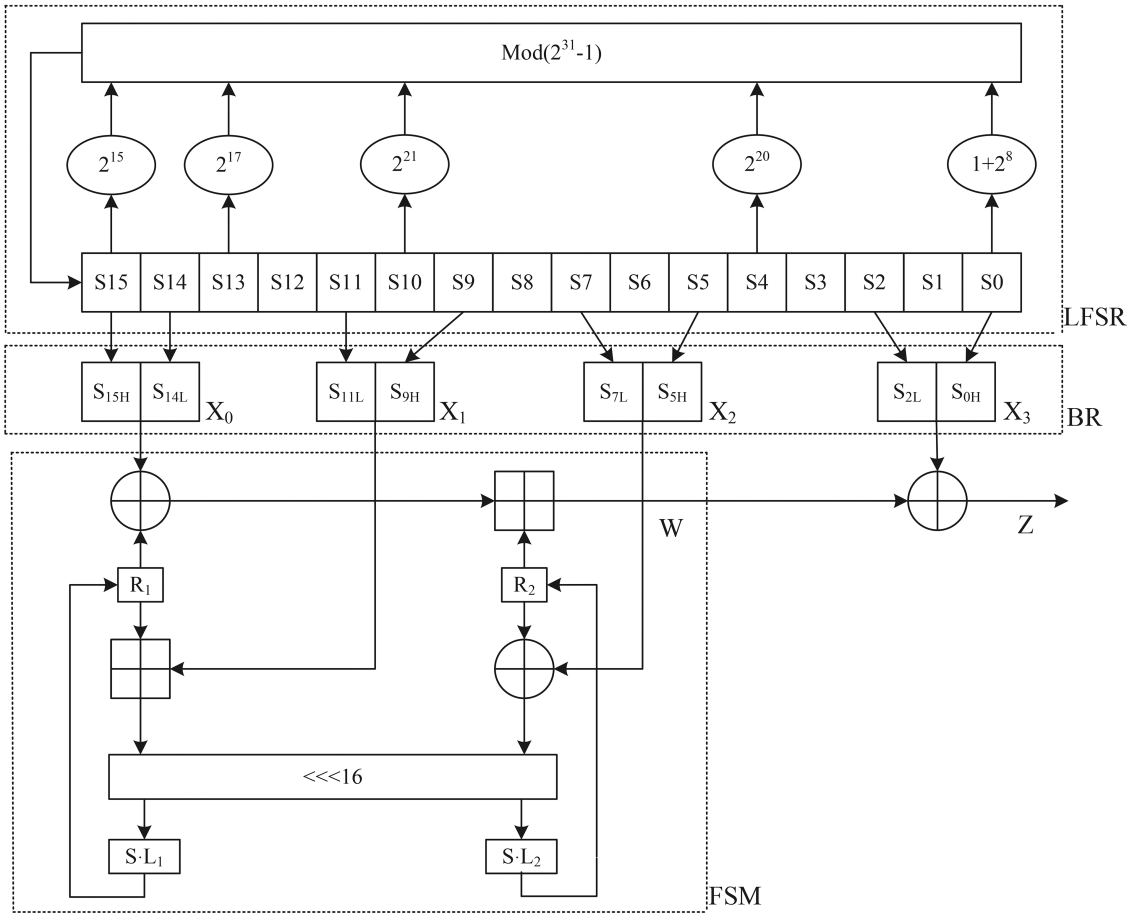

图 1　ZUC-256 流密码的工作模式.

Figure 1　Work mode of ZUC-256 stream cipher.

feedback shift register, LFSR), 它由 16 个 31 比特长的寄存器单元 $(S_{15}, S_{14}, \ldots, S_1, S_0)$ 组成, 其中存储的数据为 $(s_{15}, s_{14}, \ldots, s_1, s_0)$. 第二部分是比特重组 (bit reorganization, BR), 该部分主要从 LFSR 中取出特定的比特串后重组成 4 个 32 比特字 (X_0, X_1, X_2, X_3) 并将其用于后续的第三部分及密钥流的产生. 第三部分是有限状态自动机 (finite state machine, FSM), 该部分含有两个 32 比特字寄存器 R_1 与 R_2 以及模 2^{32} 加法运算、线性变换 L、S 盒变换等操作.

2.3　LFSR

根据工作模式的不同, LFSR 的更新步骤也存在一定的差异. 在初始化模式下, 生成 s_{16} 的步骤如式 (1) 所示, 在工作模式下, 该步骤如式 (2) 所示:

$$s_{16} = (u + 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)s_0) \mod (2^{31} - 1), \tag{1}$$

$$s_{16} = (2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)s_0) \mod (2^{31} - 1), \tag{2}$$

其中, u 表示 FSM 的输出 W 右移 1 位后得到的 31 位比特串, 即 $u = W \gg 1$.

若经过上述计算后, s_{16} 的取值恰好为 0, 则将其重置为 $2^{31} - 1$. 得到的 s_{16} 将用于 LFSR 的更新移位操作, 即 LFSR 中存储的上一轮的值 $(s_{15}, s_{14}, \ldots, s_1, s_0)$ 依次右移, 舍弃 s_0 并填补 s_{16}, 最终完成更新移位操作, 得到值 $(s_{16}, s_{15}, \ldots, s_2, s_1)$.

2.4　BR

BR 层从 LFSR 的 8 个寄存器单元 $(S_{15}, S_{14}, S_{11}, S_9, S_7, S_5, S_2, S_0)$ 中各抽取特定的 16 比特, 并将其重组为 4 个 32 比特字 (X_0, X_1, X_2, X_3). 其中 X_0, X_1 与 X_2 参与 FSM 部分的计算, 而 X_3 用于生成密钥流. BR 层的具体过程如下式所示:

$$\begin{aligned} X_0 &= s_{15\mathrm{H}}||s_{14\mathrm{L}}, \\ X_1 &= s_{11\mathrm{L}}||s_{9\mathrm{H}}, \\ X_2 &= s_{7\mathrm{L}}||s_{5\mathrm{H}}, \\ X_3 &= s_{2\mathrm{L}}||s_{0\mathrm{H}}. \end{aligned}$$

2.5　FSM

FSM, 即非线性函数部分, 该层对 BR 层产生的 X_0, X_1, X_2 进行异或、模加、比特串连接、线性变换以及 S 盒等操作, 以得到用于密钥流生成和 LFSR 输入的 32 比特长的字 W. 此外, FSM 层还负责迭代更新 R_1 和 R_2 的值. FSM 模块的工作步骤如下式所示:

$$\begin{aligned} W &= ((X_0 \oplus R_1) + R_2) \mod 2^{32}, \\ W_1 &= (R_1 + X_1) \mod 2^{32}, \\ W_2 &= R_2 \oplus X_2, \\ R_1 &= S(L_1(W_{1\mathrm{L}}||W_{2\mathrm{H}})), \\ R_2 &= S(L_2(W_{2\mathrm{L}}||W_{1\mathrm{H}})), \\ L_1(X) &= X \oplus (X \lll 2) \oplus (X \lll 10) \oplus (X \lll 18) \oplus (X \lll 24), \\ L_2(X) &= X \oplus (X \lll 8) \oplus (X \lll 14) \oplus (X \lll 22) \oplus (X \lll 30), \end{aligned}$$

其中 L_1, L_2 为线性变换, S 为 32×32 的 S 盒 (由 4 个小的 8×8 的 S 盒并置而成).

2.6　ZUC 算法的基本流程

ZUC 算法生成 32 位密钥流的过程包含以下关键步骤:

(1) 将待输入的密钥和初始向量进行特定的装载操作, 将其写入 LFSR 的 16 个存储单元. 同时, 确保 FSM 中的记忆单元 R_1, R_2 初始化为 0.

(2) 依次执行 BR 层、FSM 层和 LFSR 层初始化模式的相关步骤, 该过程需重复 32 轮. 在此阶段, FSM 的输出取高 31 位作为 LFSR 的输入.

(3) 执行一次 BR 层、FSM 层和 LFSR 层工作模式的相关步骤, 但在这一阶段忽略 FSM 的输出结果.

(4) 持续循环执行步骤 (3), 确保每次 FSM 产生输出 W 时, 都与 X_3 进行异或操作, 从而生成所需的密钥流.

2.7　有限域基础

定理1 (有限域的存在性与唯一性 [16])　对于任意素数 p, 任意正整数 n, 都存在阶为 p^n 的有限域. 任何阶为 p^n 的有限域都同构于多项式 $x^{p^n} - x$ 在素数域 $\mathbb{Z}_p$ 上的分裂域.

换言之, 相同阶的有限域都是同构的. AES 算法与 ZUC 算法在 S 盒变换中各自涉及的有限域实际上是同构的, 即它们可以通过同构映射相互转换.

定理2 (基于不可约多项式构造有限域 [16])　$f(x) \in \mathbb{F}_q[X]$ 是不可约多项式, 则 $\mathbb{F}_q[X]/(f(x))$ 是有限域.

定理3 (有限域的扩张 [16])　设有限域 $\mathbb{F}_q$, 其中 $q=p^r$ 且 p 为素数, $r\geqslant 1$. 如果 $\mathbb{F}_q[X]$ 的 n 次不可约多项式为 $m(x)=a_0+a_1x+\cdots+a_nx^n$, 那么 $\mathbb{F}_q[X]/(m(x))$ 是一个含有 q^n 个元素的有限域, 并且它的每一个元素可以唯一地表示成:

$$c_0+c_1z+\cdots+c_{n-1}z^{n-1},$$

其中 $c_i\in\mathbb{F}_q$, $i=0,1,\ldots,n-1$; z 是 $m(x)$ 在 $\mathbb{F}_q[X]/(m(x))$ 下的根.

定理 2 与定理 3 是利用商环构造复合域的前提条件.

3　算法的电路优化

3.1　S 盒实现

S 盒变换 (S-box) 是众多密码算法中不可或缺的非线性组件, 其设计使得自身具有较高的代数次数以及较优的非线性度和差分均匀度, 以此增强密码系统对代数攻击的抵抗力. 从硬件实现的角度来看, S-box 的实现策略通常涵盖三种主要方法: 查表实现 [12, 15, 17]、基于布尔函数表达式实现 [18]、基于复合域进行实现 [19]. 若以面积优化作为评估标准, 那么基于复合域的实现方式无疑是三者中最为高效的选择.

3.1.1　复合域 S1 盒的设计方案一: $\mathbb{F}_{2^8}\to\mathbb{F}_{((2^4)^2)}$

ZUC 算法的 S1-box 结合了仿射变换与有限域求逆, 具体实现如下式所示:

$$\text{S1-box}(x)=Mx^{-1}+B=\begin{pmatrix}0&1&1&1&1&0&0&1\\1&0&1&1&1&1&0&0\\1&1&0&1&0&1&1&0\\1&1&1&0&0&0&1&1\\0&1&1&1&1&1&1&0\\1&0&1&1&0&1&1&1\\1&1&0&1&1&0&1&1\\1&1&1&0&1&1&0&1\end{pmatrix}x^{-1}+\begin{pmatrix}0\\1\\0\\1\\0\\1\\0\\1\end{pmatrix},\tag{3}$$

其中 M 是一个 8×8 的矩阵, B 是一个 8×1 的常向量, x 为待输入的 8 比特向量, 其求逆运算定义在有限域 $\mathbb{F}_{2^8}$ 上, 该有限域的构造方式为

$$\mathbb{F}_{2^8}=\mathbb{F}_2/f(x)=\mathbb{F}_2/(x^8+x^7+x^3+x+1).$$

本文借助文献 [20] 中提出的同构矩阵搜索算法, 与文献 [21] 中给出的一类复合域多项式基的构造方法, 成功地将 $\mathbb{F}_{2^8}$ 上的求逆运算转换至复合域 $\mathbb{F}_{((2^4)^2)}$ 中. 其中复合域的构造方式如下:

$$\mathbb{F}_{((2^4)^2)}=[\mathbb{F}_2/(x^4+x^3+x^2+x+1)]/(x^2+0001x+0010).$$

则式 (3) 可以改写为

$$\text{S1-box}(x)=Mx^{-1}+B=MT^{-1}(Tx)^{-1}+B=K(Tx)^{-1},$$

其中 T 为同构矩阵, T^{-1} 为逆同构映射矩阵. 鉴于在大多数互补金属氧化物半导体工艺库 (complementary metal oxide semiconductor, CMOS) 下, 异或门与异或非门所占用的电路面积相同. 因此, 本文将

矩阵 M 与 T^{-1} 进行合并处理, 并将模 2 加 B 的运算以异或非的形式 (在矩阵中用红色的 1 表示) 嵌入到新构建的矩阵 K 中, 以此实现运算的等效转换与面积优化表达. 值得注意的是, 矩阵 T 与 K 还存在重复的公共项. 通过应用线性层优化工具 [22] 进行精细处理, 本文成功地将前者的实现成本降低至仅包含 12 个异或门, 而后者的实现则进一步优化为包含 7 个异或门、1 个 2 输入的同或门以及 3 个 3 输入的同或门, 显著提升了电路的紧凑性. 矩阵 T 存在 8 种不同的表现形式, 本文特别选取了使得矩阵 K 包含最少门电路数量的 T 作为设计基础. 其值如下:

$$T=\begin{pmatrix}1&1&0&0&0&0&0&0\\0&1&0&0&0&1&0&1\\0&0&1&0&0&0&0&1\\0&0&0&1&0&0&1&1\\0&0&1&1&0&0&1&1\\0&0&1&1&0&1&0&0\\0&1&1&0&0&1&1&1\\0&0&1&0&1&0&0&0\end{pmatrix},\quad K=\begin{pmatrix}1&1&1&1&0&0&0&0\\1&0&0&0&1&0&0&1\\1&1&0&1&0&1&1&1\\0&1&0&1&0&0&0&1\\1&0&0&0&1&1&0&0\\0&1&1&1&0&1&0&1\\0&0&1&0&0&1&0&1\\1&1&0&0&1&0&0&1\end{pmatrix}.$$

对于复合域 $\mathbb{F}_{((2^4)^2)}$ 上的各种计算, 文献 [21] 已经详细列出了乘法、平方等运算, 本文在此就不过多赘述. 值得一提的是, 求逆运算仍有优化空间. 本文使用 Jean 等 [23] 提出的专为非线性部件优化设计的 lighter 工具, 对 4 进 4 出的求逆运算进行了优化, 优化后的表达式呈现如下:

$$\begin{cases}t_1=\neg(\neg(h_2\&h_0)|h_3), & r_1=\neg(h_1\oplus t_1), & h_0^{-1}=r_2,\\ t_2=\neg(\neg(h_2\&h_3)\&r_1), & r_2=h_0\oplus t_2, & h_1^{-1}=r_5,\\ t_3=\neg(r_2\&h_3|r_1), & r_3=\neg(h_2\oplus t_3), & h_2^{-1}=r_4,\\ t_4=\neg(\neg(r_3|r_2)|r_1), & r_4=h_3\oplus t_4, & h_3^{-1}=\neg r_3.\\ t_5=\neg(r_3\&r_2\&r_4), & r_5=r_1\oplus t_5, & \end{cases}\tag{4}$$

式 (4) 的输入为 4 比特串 $h_3h_2h_1h_0$, 输出为它的逆 $h_3^{-1}h_2^{-1}h_1^{-1}h_0^{-1}$, 下标 0 表示最低有效位. 对于 $\mathbb{F}_{2^8}$ 中的元素 g, 它可以表示成一次多项式 $g=\gamma_1 y+\gamma_0$, 其中 γ_1,$\gamma_0, y\in\mathbb{F}(2^4)$. 设 g 的逆元为 $d=\delta_1 y+\delta_0$, 则有 $gd=1$. δ_1,δ_0 的计算过程由下式给出 [19]:

$$\begin{aligned}\delta_1&=(\gamma_1^2\nu+\gamma_1\gamma_0\tau+\gamma_0^2)^{-1}\gamma_1,\\ \delta_0&=(\gamma_1^2\nu+\gamma_1\gamma_0\tau+\gamma_0^2)^{-1}(\gamma_0+\gamma_1\tau),\end{aligned}$$

其中 ν 以及 τ 是二次不可约多项式 $(x^2+0001x+0010)$ 的范数 (norm) 和迹 (trace), 用二进制比特串可以表示为 $\nu=(0010)_b$ 以及 $\tau=(0001)_b$. 那么上式可以简写为

$$\begin{aligned}\delta_1&=(\gamma_1^2\nu+\gamma_1\gamma_0+\gamma_0^2)^{-1}\gamma_1,\\ \delta_0&=(\gamma_1^2\nu+\gamma_1\gamma_0+\gamma_0^2)^{-1}(\gamma_0+\gamma_1).\end{aligned}$$

特别地, 元素 0 的逆元被定义为其本身, 即 $S(0)$ 的输出为常向量 B.

$\mathbb{F}_{((2^4)^2)}$ 上的 S1-box 实现流程如下.

(1) 顶部: 将 S1-box 的 8 比特输入 g 通过矩阵 T 映射为域 $\mathbb{F}_{((2^4)^2)}$ 上的元素 y, 即

$$T(g_7g_6g_5g_4g_3g_2g_1g_0)=(y_7y_6y_5y_4y_3y_2y_1y_0).$$

令 $\gamma_1 = (y_7y_6y_5y_4)$, $\gamma_0 = (y_3y_2y_1y_0)$.

(2) 中部: 计算 δ_1 与 δ_0, 其中 δ_1 与 δ_0 内部的求逆运算可通过式 (4) 进行, 最终 $y^{-1} = (\delta_1, \delta_0)$.

(3) 底部: 将 y^{-1} 左乘矩阵 K 即可得到 S1-box 的输出.

3.1.2　复合域 S1 盒的设计方案二: $\mathbb{F}_{2^8} \to \mathbb{F}_{(((2^2)^2)^2)}$

ZUC 的 S1-box 设计与 AES 算法的 Sbox 在结构上展现出高度的相似性. 鉴于 AES 算法在复合域实现上的成熟度与优越性, 本文采用的方案二遵循了另一种优化策略: 首先, 将 ZUC S1-box 所基于的有限域 $\mathbb{F}_{2^8}$ 通过同构映射转换至与 AES Sbox 的同阶有限域上, 该域定义为 $\mathbb{F}'_{2^8} = \mathbb{F}_2/(x^8+x^4+x^3+x+1)$. 随后, 为了进一步优化资源消耗, 本文再次实施同构映射至一个塔域结构 $\mathbb{F}_{(((2^2)^2)^2)}$ [19]. 则 S1-box 的计算过程可由下式得到:

$$\text{S1-box}(x) = MT^{-1}D^{-1}(DTx)^{-1} + B,$$

其中, 矩阵 T 为域 $\mathbb{F}_{2^8}$ 至域 $\mathbb{F}'_{2^8}$ 的同构映射矩阵, 矩阵 D 为域 $\mathbb{F}'_{2^8}$ 至塔域 $\mathbb{F}_{(((2^2)^2)^2)}$ 的同构映射矩阵, T^{-1}, D^{-1} 分别为其逆矩阵. T 的搜索算法如算法 1 所示. 各矩阵的取值如下所示:

$$T = \begin{pmatrix} 0&0&0&0&1&1&0&0\\ 0&1&0&1&1&1&1&0\\ 1&1&0&1&0&0&1&0\\ 1&0&1&0&0&1&1&0\\ 1&0&1&1&1&1&0&0\\ 1&0&1&0&1&0&0&0\\ 0&0&0&1&0&0&1&0\\ 1&0&1&1&1&1&1&1 \end{pmatrix}, \quad D = \begin{pmatrix} 1&1&1&0&0&1&1&1\\ 0&1&1&1&0&0&0&1\\ 0&1&1&0&0&0&1&1\\ 1&1&1&0&0&0&0&1\\ 1&0&0&1&1&0&1&1\\ 0&0&0&0&0&0&0&1\\ 0&1&1&0&0&0&0&1\\ 0&1&0&0&1&1&1&1 \end{pmatrix}.$$

与第 3.1.1 小节中采用的处理方式相类似, 上述 S1-box 的表达式可以简化为

$$\text{S1-box}(x) = U(Px)^{-1},$$

$$P = \begin{pmatrix} 1&0&0&0&0&1&0&1\\ 1&0&0&1&0&1&0&1\\ 0&0&1&0&0&0&0&1\\ 0&0&1&1&1&1&1&1\\ 1&0&1&1&1&0&1&1\\ 1&0&1&1&1&1&1&1\\ 0&0&1&1&0&0&1&1\\ 1&1&1&0&0&1&1&1 \end{pmatrix}, \quad U = \begin{pmatrix} 0&0&1&0&0&1&0&1\\ 0&1&1&1&1&1&1&0\\ 1&1&1&0&0&0&0&1\\ 0&0&0&0&1&1&0&1\\ 0&1&0&1&0&0&1&1\\ 0&0&0&1&1&0&1&0\\ 0&1&1&0&1&0&1&1\\ 1&1&1&1&1&1&0&1 \end{pmatrix}.$$

精细处理后, 矩阵 P 的实现代价为 12 个异或门, 矩阵 U 的实现代价为 12 个异或门以及 4 个异或非门. 在求逆运算的实现上, 本文直接借鉴了文献 [19] 中提出的 GF_INV_8 求逆电路 (此处可去除多余的选择逻辑, 文献 [19] 需要选择逻辑来执行逆 Sbox 操作).

$\mathbb{F}_{(((2^2)^2)^2)}$ 上的 S1-box 实现流程如下.

(1) 顶部: 将 S1-box 的 8 比特输入 g 通过矩阵 P 映射为域 $\mathbb{F}_{(((2^2)^2)^2)}$ 上的元素 e, 即

$$P(g_7g_6g_5g_4g_3g_2g_1g_0) = (e_7e_6e_5e_4e_3e_2e_1e_0).$$

算法 1 同构矩阵搜索算法: $\mathbb{F}_{2^n} \to \mathbb{F}'_{2^n}$.

输入: $\mathbb{F}_{2^n}$ 与 $\mathbb{F}'_{2^n}$ 中的 n 次不可约多项式 $p(x)$ 与 $q(x)$、仿射矩阵 M 及相关矩阵 M' (若有);
遍历 $\mathbb{F}'_{2^n}$ 上的所有元素 α;
if $p(\alpha) = 0$ **then**
　for $i = n-1, i \geqslant 0, i--$ **do**
　　执行模幂操作: $\beta = \alpha^i \mod q(x)$;
　　将结果 β 放入待筛选矩阵 T_i 的第 $n-i-1$ 列 (最左侧的一列被标识为第 0 列);
　end for
else
　继续执行第一步;
end if
得到 n 个待筛选矩阵 $T_0 \cdots T_{n-1}$;
for $i = n-1, i \geqslant 0, i--$ **do**
　执行矩阵的模 2 乘法: $T'_i = M \otimes M' \otimes T_i$;
　计算矩阵中 1 的个数: $\mathrm{Cnt}_i = \mathrm{Count}(T'_i)$;
end for
取 1 个数最少的矩阵下标: $\min = \mathrm{FindMin}(\mathrm{Cnt}_0 \cdots \mathrm{Cnt}_{n-1})$;
输出: 含最少异或逻辑门的同构映射矩阵 $T = T_{\min}$.

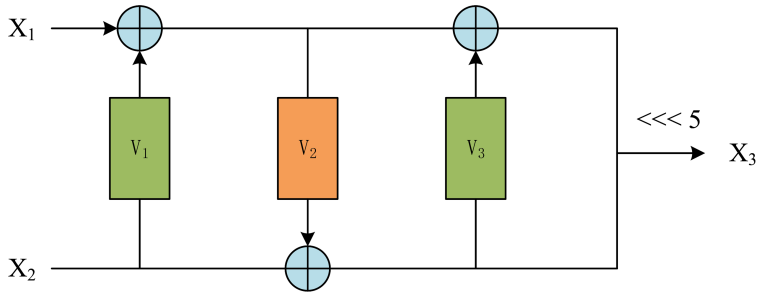

图 2　(网络版彩图) S0-box 结构.

Figure 2　(Color online) Structure of the S0-box.

(2) 中部: 利用 GF_INV_8 对元素 e 进行求逆.

(3) 底部: 将 e^{-1} 左乘矩阵 U 即可得到 S1-box 的输出.

3.1.3　组合逻辑 S0 盒的设计

ZUC 的 S0-box 在本质上遵循三层 Feistel 结构设计 [24], 该特性在图 2 中得到了直观的展示. 其中, S0-box 的输入包括一个高 4 比特串 X_1 和低 4 比特串 X_2. 变换 V_1, V_2, V_3 均定义在 $\mathbb{F}_{2^4}$ 上. 特别地, V_2 因其具备一一映射的特性而显得独特, 这使得本文可以利用 lighter 工具进行优化. 而 V_1 与 V_3 在经过公共项消除后, 可通过简单的组合逻辑实现. V_2 优化后的表达式如下所示:

$$
\begin{cases}
t_1 = \neg vi_2, & r_1 = \neg(t_2 \oplus vi_0), & \\
t_2 = \neg(t_1 \& vi_3), & r_2 = \neg(t_3 \oplus vi_3), & vo_0 = r_7, \\
t_3 = \neg(r_1 \& t_1), & r_3 = \neg(t_3 \oplus r_2), & vo_1 = r_6, \\
t_4 = \neg(r_3 | r_1), & r_4 = \neg(t_1 \oplus t_4), & vo_2 = \neg(r_3 \oplus t_7), \\
t_5 = \neg(\neg(r_3 | r_2) | r_4), & r_5 = r_1 \oplus t_5, & vo_3 = \neg r_5. \\
t_6 = \neg(r_3 \& r_6), & r_6 = r_2 \oplus r_4, & \\
t_7 = \neg(r_5 \& r_7), & r_7 = t_6 \oplus r_4, &
\end{cases}
$$

表 1　不同实现方案的 Sbox 对比.

Table 1　Comparison of Sbox with different schemes.

Implementation	Area (GE)	Latency (ns)	Technology (nm)
S1-box LUT	450.24	1.57	TSMC 90
S0-box LUT	440.49	1.81	TSMC 90
AES Sbox [25]	202	–	SMIC 130
Our S0-box	63.49	1.59	TSMC 90
Our S1-box Scheme 1	215.24	4.94	TSMC 90
Our S1-box Scheme 2	201.99	4.48	TSMC 90

其输入为 (vi_3, vi_2, vi_1, vi_0), 输出为 (vo_3, vo_2, vo_1, vo_0).

3.1.4　S 盒实现对比

基于上述设计方案, 本文在 Synopsys 公司的电路综合工具 Design Compiler (K2015.6) 上完成了综合. 表 1 对比了 ZUC 查找表实现、本文方案一、本文方案二以及现有的 AES Sbox [25] 实现之间的性能差异 (工艺角为 SS, 温度为 125°C, 电压 0.9 V). 本文所提出的 S0-box 实现相较于其传统的查找表实现, 在面积消耗上实现了显著的 86% 的缩减, 同时时延也降低了 13%, 展现出了卓越的性能优化. 进一步地, 本文提出的 S1-box 实现方案二在面积表现上为 201.99 GE (等效门, 二输入的与非门), 这一数值相较于其查找表实现缩减了 56%, 与当前先进的 AES Sbox 实现方案在面积上相当.

3.2　线性变换 L 的实现

ZUC 算法线性层 L 所采用的 MDS 矩阵由移位和异或构造, 因此对 L 的优化本质上即是对 MDS 的优化. 在孙壮等 [26] 的最新研究成果中, 线性变换 L_1 与 L_2 的实现共使用了 173 个异或门. 借助于 Lin 等 [27] 对 SM4 算法线性层的观察 (性质 1) 以及利用线性层优化工具 [22], 本文实现 L_1 与 L_2 仅需 164 个异或门, 相较于孙壮等的方案, 减少了 6% 的逻辑门消耗.

性质1 ([27])　在 SM4 轮函数中, 线性变换 L 所对应的矩阵可以表示为一个右循环矩阵:

$$\mathrm{Circ}_1(I_2, I_2, O_2, O_2, O_2, I_2, , O_2, O_2, O_2, I_2, O_2, O_2, I_2, , O_2, O_2, O_2),$$

其中 I_2 是 2 行 2 列的单位矩阵, O_2 是 2 行 2 列的零矩阵. ZUC 算法的线性变换 L_1 与 SM4 算法中线性变换 L 完全一致, L_2 同样能够以一种右循环矩阵 Circ_2 的形式来表达:

$$\mathrm{Circ}_2(I_2, O_2, O_2, O_2, I_2, O_2, , O_2, I_2, O_2, O_2, O_2, I_2, O_2, , O_2, O_2, I_2).$$

因此, L_2 可被简化为 $\mathbb{F}_2$ 上两个相同 16×16 矩阵 L_2'. 对于 L_2 的输入 $(li_{31}, li_{30}, \ldots, li_1, li_0)$, 其输出 $(lo_{31}, lo_{30}, \ldots, lo_1, lo_0)$ 的计算方式为

$$\begin{aligned}(lo_{31}, lo_{29}, \ldots, lo_3, lo_1)^{\mathrm{T}} &= L_2'(li_{31}, li_{29}, \ldots, li_3, li_1)^{\mathrm{T}},\\ (lo_{30}, lo_{28}, \ldots, lo_2, lo_0)^{\mathrm{T}} &= L_2'(li_{30}, li_{28}, \ldots, li_2, li_0)^{\mathrm{T}},\end{aligned}$$

$$L_2' = \mathrm{Circ}(1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,1).$$

借助文献 [22] 中提出的算法, L_2' 可由 41 个异或门实现 (L_1' 同样如此). 故整个 ZUC 算法的线性变换 L 的实现共需 $41 \times 4 = 164$ 个异或门.

表 2　本文例化的功能模块.

Table 2　Function modules instances in this paper.

Block	Enable signal (C_clk)	Block description
S1-box	–	Scheme 2 in this paper
S0-box	–	Combinational logic implementation of S0-box
L_1	{0}	Implemented by combining two L_1' blocks
L_2	{0}	Implemented by combining two L_2' blocks
CSA2_31	{0, 1, 2}	Implemented by cascading two 31-bit CSA adders
Adder32	{0, 2, 3}	32-bit adder
Adder31	{3}	31-bit adder for a single operand

3.3　ZUC 算法的整体实现

为了优化算法实现时的电路面积, 本文采取时序复用的策略 (详见 3.3.1 小节), 确保每个功能模块仅被实例化 1 次. 考虑到算法中的 Sbox 是由 4 个小型 Sbox 组合而成 (S0-box, S1-box, S0-box, S1-box), 且线性变换 L_1, L_2 的输出均需经过 Sbox 进行进一步处理, 故本文将原本单轮 LFSR 的更新操作分解为 4 个独立的时钟周期来执行. 此设计旨在提升资源利用效率, 详细例化的功能模块如表 2 所示. 其中, 模块使能信号为 C_clk, 其取值范围限定在集合 {0, 1, 2, 3} 内. 值得一提的是, 模块 Adder32 在 C_clk = 2 时, 其待输入的值会根据算法当前所处的阶段 (初始化阶段或工作阶段) 而有所不同. 此外, 由于 S1-box 和 S0-box 每轮需要跨越 4 个时钟周期完成其工作, 因此并未为它们单独设置使能信号.

电路的整体设计如图 3 所示. 为了清晰区分, 本文所采用的寄存器以橙色背景高亮显示, 而功能模块的时序复用部分则采用灰色填充. 为了简化算法的整体结构图示, 图中并未直接采用逻辑器件符号来描绘多路选择器, 而是以线路交汇处的黑色小圆点作为替代. 鉴于多路选择器的面积占用显著超过 31 位二输入异或门, 因此设计中未对异或门进行复用处理. 此外, 该设计引入了 32 位中间寄存器 FF_1 与 31 位中间寄存器 FF_2, 它们共同协作以分隔并管理 4 个独立的时钟周期 (stage0∼stage3). 尤为值得一提的是, FF_1 还肩负着存储预计算结果 $W_1 = R_1 \boxplus X_1$ 的任务, 此举有效地缩短了关键路径的长度. 当算法运行至稳定的工作状态时, 每经过 4 个时钟周期, Adder32 的输出 $W = (X_0 \oplus R_1) \boxplus R_2$ 将与 X_3 进行异或运算, 最终生成所需的密钥流.

3.3.1　加法链与中间寄存器的时序复用

模加运算体现在 ZUC 算法中的 LFSR 和 FSM 部分, 其中 LFSR 执行模 $2^{31}-1$ 加法运算, 而 FSM 执行模 2^{32} 加法运算. 为使得电路面积最小化, 本文并未针对两种模加运算结构进行专门定制的独特设计, 而是采用了 CSA 加法器、Adder32 以及 Adder31 来实现其功能.

为了清晰地阐述时序复用的逻辑流程, 本文以初始化阶段的首轮操作为具体实例进行说明, 如表 3 所示. 为简化说明过程, 预设 ZUC 算法在执行此轮操作前已成功装载了必要的初始密钥及向量.

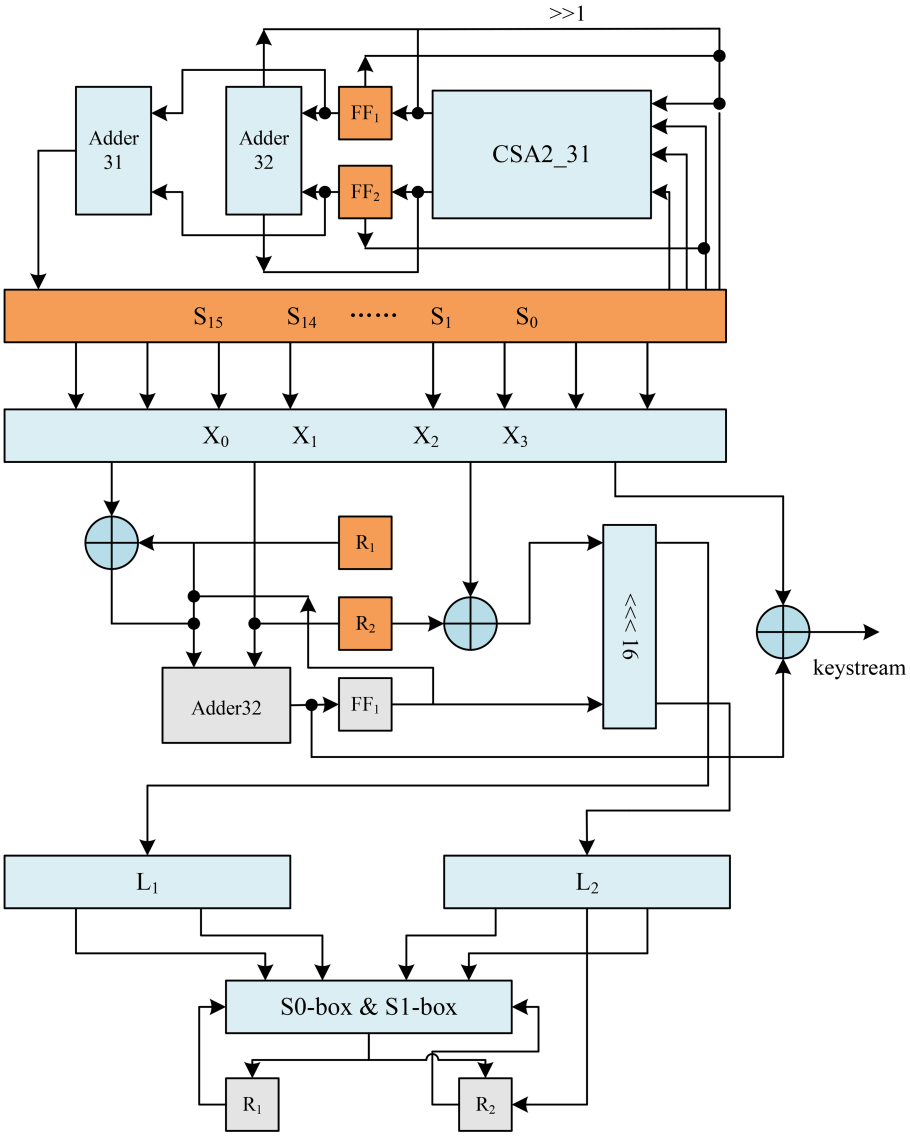

图 3　ZUC 整体结构.

Figure 3　Framework of ZUC.

表 3 中各值的计算方式如下, 其中 S_i^n 表示 S_i 处的值在模 $2^{31}-1$ 下的 n 次方:

$$\begin{cases} t_1 = S_0 \oplus S_0^8 \oplus S_4^{20}, & r_1 = S_0 \& S_0^8 | S_0 \& S_4^{20} | S_0^8 \& S_4^{20}, \\ t_2 = \mathrm{FF}_1 \oplus \mathrm{FF}_2 \oplus S_{10}^{21}, & r_2 = \mathrm{FF}_1 \& \mathrm{FF}_2 | \mathrm{FF}_1 \& S_{10}^{21} | \mathrm{FF}_2 \& S_{10}^{21}, \\ t_3 = \mathrm{FF}_1 \oplus \mathrm{FF}_2 \oplus S_{15}^{15}, & r_3 = \mathrm{FF}_1 \& \mathrm{FF}_2 | \mathrm{FF}_1 \& S_{15}^{15} | \mathrm{FF}_2 \& S_{15}^{15}, \\ N_1 = t_1 \oplus (r_{1[29:0]} || r_{1[30]}) \oplus u, & C_1 = t_1 \& (r_{1[29:0]} || r_{1[30]}) | t_1 \& u | (r_{1[29:0]} || r_{1[30]}) \& u, \\ N_2 = t_2 \oplus (r_{2[29:0]} || r_{2[30]}) \oplus S_{13}^{17}, & C_2 = t_2 \& (r_{2[29:0]} || r_{2[30]}) | t_2 \& S_{13}^{17} | (r_{2[29:0]} || r_{2[30]}) \& S_{13}^{17}, \\ N_3 = t_3 \boxplus (r_{3[29:0]} || r_{3[30]}). \end{cases}$$

在此设计中, FF_1 寄存器在前 3 个时钟周期内分别存储了 CSA 加法操作输出的和 N_1, N_2, N_3. 而在第 4 个时钟周期, 它则负责存储预计算的 W_1 值, 该值为下一轮 4 个时钟周期的运算做准备. FF_2 寄存器在前两个时钟周期内分别存储了 CSA 加法操作产生的进位部分 C_1 与 C_2, 后两个时钟周期则

表 3　时序复用的逻辑流程. ⊞: 模 2^{32} 加运算符.

Table 3　Logic flow of time-division multiplexing. ⊞ denotes the integer modular 2^{32} addition.

Register	Stage0	Stage1	Stage2	Stage3
FF_1	N_1	N_2	N_3	$R_1 \boxplus X_1$
FF_2	C_1	C_2	–	–
R_1	S0-box($L_{1[31:24]}$) S1-box($L_{1[23:16]}$) $L_{1[15:0]}$	S0-box($R_{1[15:8]}$) S1-box($R_{1[7:0]}$)	–	–
R_2	$L_{2[31:0]}$	–	S0-box($R_{2[31:24]}$) S1-box($R_{2[23:16]}$)	S0-box($R_{2[15:8]}$) S1-box($R_{2[7:0]}$)

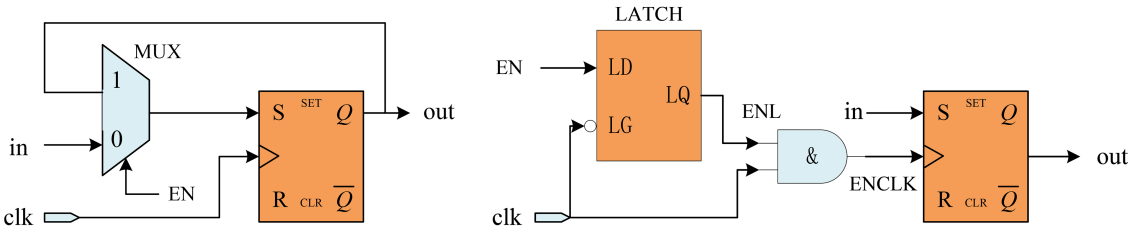

图 4　(网络版彩图) 门控时钟的应用.

Figure 4　(Color online) Gate-controlled clock application.

保持空闲状态. 值得注意的是, FF_1 及 FF_2 的初始值分别为 X_1 及 0. R_1 与 R_2 寄存器以错开时钟周期的方式使用 S0-box 与 S1-box. 在 stage3 阶段, Adder31 模块接收来自 FF_1 的值 N_3 (Adder32 计算得出的 31 比特和以及 1 比特进位信号), 随后执行加法运算, 从而得出 LFSR 的更新值 s_{16}. 该值存储在寄存器 S_{15} 中, 系统将寄存器组 $(S_{14}, \ldots, S_0)$ 的值依次向右移动, 确保 LFSR 的正确更新.

3.3.2　门控时钟的应用

由表 3 可知, 在某些时钟周期内, 特定功能模块处于闲置状态. 为了有效降低电路的总体功耗, 本文在 RTL (register transfer level) 层面引入了门控时钟技术. 图 4 直观地展示了该技术的优势: 左侧展示的是未采用门控时钟的传统触发器, 而右侧则是加入了门控时钟机制后的触发器. 在未应用门控时钟的传统架构中, 针对有源时钟信号 clk 和同步控制信号的触发器组, 综合工具依赖于复杂的反馈环路和选择器来实现连接. 当这些触发器在连续多个时钟周期内保持恒定值时, 尽管其输出未发生变化, 但它们仍不可避免地消耗着能量. 特别是在使能信号 EN 长时间保持为 0 的情况下, 触发器组及相关的时钟线和选择器会持续消耗不必要的能量, 这主要是由于时钟信号的频繁翻转触发了电路中的冗余开关动作, 从而增加了额外的功率损耗.

FF_2, R_1 以及 R_2 等寄存器即存在上述问题, 因此本文采用了图 4 右侧所示的结构来优化这些组件的性能. 引入门控时钟后, 新的时钟信号 ENCLK 由与门控制, 该与门的使能则源自使能信号 EN. 而 ENCLK 的翻转频率远低于源时钟信号 clk, 这一特性能极大地减少寄存器内部及时钟网络的冗余功耗. 换言之, 采用门控时钟技术的组件, 其时钟输入得到了精细化的管理, 确保仅在数据更新或状态转换的必要时刻才被激活. 此外, 由于门控时钟替代了原电路中的选择器, 这进一步实现了电路面积的缩减.

4　验证与分析

本文基于 ASIC (application specific integrated circuit) 平台, 对第 3 节所设计的电路进行了具体

表 4　本文成果与相关工作在硬件性能上的比较.

Table 4　Comparison of the hardware performance in this paper with related work.

Implementation	Area (kGE)	Frequency (MHz)	Total power (mW)	Throughout (Gbps)	Technology (nm)
AES [14]	7.1	100	–	0.05	65
ZUC [15]	11.76	–	0.371	–	TSMC 130
ZUC+SNOW3G [14]	16.6	100	0.81 (SNOW3G)	0.8	65
ZUC+SNOW3G [15]	13.5	–	–	2	UMC 65
ZUC (this paper)	6.6	250	1.89	2	TSMC 90 (slow)
ZUC (this paper)	7.0	1000	12.3	8	TSMC 90 (fast)

实现. 所有设计均通过综合工具 Design Compiler (K2015.6) 进行了全面综合, 随后在 Modelsim (10.5) 软件平台上完成了功能仿真测试, 以确保所提电路架构的可靠性. 为验证 ZUC 算法实现的准确性, 本文将 ZUC 算法国家标准文档中的测试用例与实现结果进行了逐一比对, 结果显示两者完全一致, 证明了本方案能够准确无误地生成密钥流. 值得注意的是, 由于本文中的 ZUC 算法每 4 个时钟周期才产生一个 32 比特的密钥流字 keystream, 故算法的吞吐率为工作频率的 8 倍. 为了进一步评估本文工作的性能与效率, 表 4 从多个关键维度, 包括面积占用、工作频率、吞吐率、功耗表现以及所采用的工艺库, 将本文的实验成果与当前已公开发表的研究成果进行了全面而深入的对比分析.

从表 4 中可观察到, 本文所实现的 ZUC 算法在 TSMC 90 nm 工艺库下, 针对 Slow 工艺条件 (SS 工艺角、温度 125°C、电压 0.9 V), 其面积占用仅为 6.6 kGE. 这一性能指标相较于文献 [14] 中提出的 AES 紧凑实现, 实现了 8% 的面积缩减; 而与文献 [15] 中 ZUC 算法架构相比, 更是实现了高达 44% 的面积优化. 值得注意的是, 文献 [14,15] 还探讨了将 ZUC 算法与 SNOW3G 算法集成于整体的设计, 由于两者算法流程的高度相似性, 使得它们能够共享 LFSR 的寄存器组, 从而在面积占用上展现出显著优势. 然而, 即便在如此高效的基准下, 本文提出的 ZUC 架构面积占用仍不到其 50%, 同时保持了相当的吞吐率, 这充分彰显了本文设计仍具有一定的优势. 此外, 本文还进一步在 Fast 工艺条件 (FF 工艺角、温度 −40°C、电压 1.1 V) 下对 ZUC 算法实现进行了详尽评估. 评估结果显示, 该设计在面积占用上略有增加, 增加至 7.0 kGE 的同时, 工作频率可高达 1 GHz, 吞吐率更是达到了 8 Gbps. 这一结果为其在多种应用场景下的高效运行提供了坚实支撑.

4.1　面积占用详情

为了清晰呈现本文的组织架构, 图 5 采用扇形图形式, 从各子组件面积占比的维度详细展示了具体数据. 其中, 以 $(S_{15},\ldots,S_0)$ 为核心的 LFSR 寄存器组, 以及 FF_1, FF_2, R_1, R_2 等寄存器, 共同构成了面积占用的最大份额, 达到了 4205.2 GE, 占比 63%. 其他逻辑则包括了用于初始向量以及密钥装载过程中消耗的或门、用于 FSM 的异或门、用于生成控制信号的选择逻辑以及用于状态判断的逻辑表达式等. 扇形图 6 从芯片面积分布角度列举了组合逻辑、非组合逻辑、缓冲器以及反向器 (buffer/inverter, BUF/INV) 的面积分布情况. 其中组合逻辑开销最大, 为 3624.74 GE, 占比 53%.

4.2　关键路径

本文在深入探讨面积消耗问题的同时, 亦未忽视对时延因素的充分考量. 如 3.3.1 小节所述, 本文对 FSM 部分中值 W_1 进行了预计算处理, 并将其存储在 FF_1 中. 故 FSM 的关键路径由 Path_1 缩减至更为高效的 Path_2, 即 32 比特加法器 Adder32 缩减为 32 比特的异或门.

$$\mathrm{Path}_1 : W_1 = R_1 \boxplus X_1 \to L \to S_1,$$
$$\mathrm{Path}_2 : W_2 = R_2 \oplus X_2 \to L \to S_1,$$

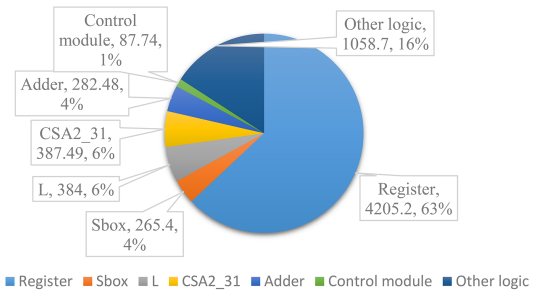

图 5　子部件面积占用.

Figure 5　Subcomponent area occupancy.

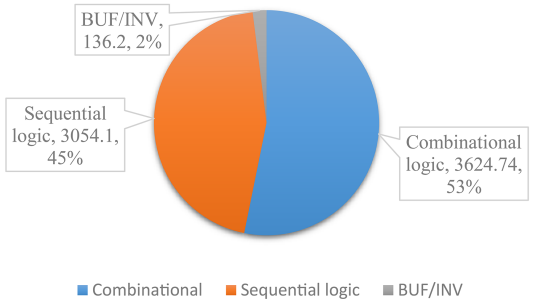

图 6　芯片面积分布.

Figure 6　Chip area distribution.

其中, "$\rightarrow$" 用于指示一个逻辑到另一个逻辑的方向. 由于加法链的复杂性, 本文 ZUC 算法实现的关键路径位于初始化阶段 stage0 的 LFSR 部分, 其路径 Path_3 可以被描述为以下过程:

$$\mathrm{Path}_3 : R_1 \oplus X_0 \rightarrow u = [(R_1 \oplus X_0) \boxplus R_2] \gg 1 \rightarrow \mathrm{CSA2\_31},$$

即关键路径由 1 个异或门、逻辑单元 Adder32 以及 CSA2_31 组成.

5　总结

本文针对窄带物联网这一特定应用场景, 设计并提出了 ZUC 算法的紧凑型硬件实现方案. 该方案通过综合考虑算法整体实现的时序复用设计、Sbox 实现的复合域设计、线性层 L 实现的拆分设计、寄存器的门控时钟控制以及加法链的共享策略, 实现了算法在面积和功耗上的双重优化. 基于上述优化措施, 本文的 ZUC 算法硬件实现在 TSMC 90 nm 工艺库下相较于现有研究成果具有更好的性能. 具体地, 在工艺角为 SS 的情况下, 本文的硬件实现面积最低仅为 6.6 kGE, 吞吐率和工作频率分别为 2 Gbps 和 250 MHz. 此外, 在工艺角为 FF 的情况下, 本文硬件实现在面积略增至 7.0 kGE 的条件下, 将吞吐率提升至 8 Gbps, 同时工作频率增至 1 GHz. 鉴于 NB-IoT 设备对低面积成本和低功耗的迫切需求, 以应对大规模设备部署的缺口及在电力基础设施薄弱的偏远地区部署的挑战, 本方案更加贴合 NB-IoT 等资源要求严苛的应用场景.

参考文献

1　Peter O, Pradhan A, Mbohwa C. Industrial Internet of Things (IIoT): opportunities, challenges, and requirements in manufacturing businesses in emerging economies. Procedia Comput Sci, 2023, 217: 856–865

2　Rejeb A, Rejeb K, Treiblmaier H, et al. The Internet of Things (IoT) in healthcare: taking stock and moving forward. Internet Things, 2023, 22: 100721

3　Badshah A, Ghani A, Daud A, et al. Towards smart education through Internet of Things: a survey. ACM Comput Surv, 2023, 56: 1–33

4　Alaa M, Zaidan A A, Zaidan B B, et al. A review of smart home applications based on Internet of Things. J Netw Comput Appl, 2017, 97: 48–65

5　Vailshery L S. Number of IoT connections worldwide 2022-2033, with forecasts to 2030. [2024-8-15]. https://www.statista. com/statistics/1183457/iot-connected-devices-worldwide/

6　Chui M, Collins M, Patel M. IoT value set to accelerate through 2030: where and how to capture it. [2024-8-15]. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it

7　Mahbub M. NB-IoT: applications and future prospects in perspective of Bangladesh. Int J Inf Technol, 2020, 12: 1183–1193

8 Popovski P, Trillingsgaard K F, Simeone O, et al. 5G wireless network slicing for eMBB, URLLC, and mMTC: a communication-theoretic view. IEEE Access, 2018, 6: 55765–55779

9 Narayanan S, Tsolkas D, Passas N, et al. NB-IoT: a candidate technology for massive IoT in the 5G era. In: Proceedings of IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, 2018. 1–6

10 Mentsiev A U, Magomaev T R. Security threats of NB-IoT and countermeasures. IOP Conf Ser-Mater Sci Eng, 2020, 862: 052033

11 Liu Z B, Zhang Q L, Ma C Q, et al. HPAZ: a high-throughput pipeline architecture of ZUC in hardware. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016. 269–272

12 Liu Y T, Shen Z S, Fang S, et al. A hardware design of ZUC-256 stream cipher of pipelining structure with high throughput. Acta Electron Sin, 2023, 51: 438 [刘云涛, 申泽生, 方硕, 等. 高吞吐率流水线结构的 ZUC-256 流密码硬件设计. 电子学报, 2023, 51: 438]

13 Xu A, Wu Y, Yang J, et al. A high-throughput hardware implementation of ZUC-256 stream cipher. In: Proceedings of the 4th International Conference on Communications, Information System and Computer Engineering (CISCE), 2022. 24–27

14 Cavo L, Fuhrmann S, Liu L. Design of an area efficient crypto processor for 3GPP-LTE NB-IoT devices. Microprocessors Microsyst, 2020, 72: 102899

15 Sharaf M A, AbdelBary E, Mostafa H, et al. Efficient ASIC implementation of a NB-IoT security co-processor. In: Proceedings of IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 2020. 695–698

16 Lidl R, Niederreiter H. Finite Fields. Cambridge: Cambridge University Press, 1997

17 Hulle N, Prathiba B, Khope S R. Hardware optimization and FPGA implementation of pipelined ZUC architecture. In: Proceedings of IEEE 6th International Conference on Computing, Communication and Automation (ICCCA), 2021. 63–69

18 Gligoroski D, Moe M E G. On deviations of the AES S-box when represented as vector valued Boolean function. Int J Comput Sci Network Secur, 2007, 7: 156–163

19 Canright D. A very compact S-box for AES. In: Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2005. 441–455

20 Chen C, Guo H, Wang C, et al. A fast software implementation of SM4 based on composite fields. J Cryptol Res, 2023, 10: 289–305 [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. 密码学报, 2023, 10: 289–305]

21 Li Y J, Zhang W G, Ge Y D, et al. Optimized realization of AES-like algorithm S-Box. J Cryptol Res, 2023, 10: 531–538 [李艳俊, 张伟国, 葛耀东, 等. 类 AES 算法 S 盒的优化实现. 密码学报, 2023, 10: 531–538]

22 Xiang Z J, Zeng X Y, Lin D, et al. Optimizing implementations of linear layers. IACR Trans Symmetric Cryptol, 2020, 2020: 120–145

23 Jean J, Peyrin T, Sim S M, et al. Optimizing implementations of lightweight building blocks. IACR Trans Symmetric Cryptol, 2017, 2017: 130–168

24 SAGE E. Specification of the 3GPP Confidentiality and Integrity Algorithms 128EEA3 & 128EIA3. Document 4: Design and Evaluation Report, 2011

25 Wei Z H, Sun S S, Hu L, et al. Searching the space of tower field implementations of the $\mathbb{F}_{2^8}$ inverter-with applications to AES, Camellia, and SM4. Int J Inf Comput Secur, 2023, 20: 1–26

26 Sun Z, Huang Z Y. Implementing quantum circuit of ZUC algorithm. J Cyber Secur, 2023 [孙壮, 黄震宇. ZUC 算法的量子电路实现. 信息安全学报, 2023] doi: 10.19363/J.cnki.cn10-1380/tn.2023.06

27 Lin D, Xiang Z, Xu R, et al. Quantum circuit implementations of SM4 block cipher based on different gate sets. Quantum Inf Process, 2023, 22: 282

Compact hardware implementation of ZUC algorithm for NB-IoT

Rui SONG[1], Zejun XIANG[1,2*], Ruolin ZHANG[3], Shasha ZHANG[1] & Siwei CHEN[1]

1. *School of Cyber Science and Technology, Hubei University, Wuhan 430062, China*;
2. *State Key Laboratory of Cryptology, Beijing 100878, China*;
3. *Faculty of Mathematics and Statistics, Hubei University, Wuhan 430062, China*
* Corresponding author. E-mail: xiangzejun@hubu.edu.cn

Abstract The Internet of Things (IoT) domain is currently facing unavoidable and persistent challenges in the form of cybersecurity threats and constrained device resources. To address the aforementioned issues, this paper designs an efficient low-area hardware implementation of the ZUC algorithm on an ASIC platform, utilizing time-multiplexing and clock gating techniques. This circuit minimizes area by ensuring that each functional module is instantiated only once. In the design of the S-box, this paper draws on the concept of tower field decomposition and proposes an algorithm to search for isomorphic mapping matrices between the finite field $\mathbb{F}_{2^n}$ and the finite field $\mathbb{F}'_{2^n}$. The algorithm aims to find an isomorphic mapping that, when multiplied by the affine matrix of the S-box operation and other related matrices, can achieve the mapping with the least number of XOR logic gates. Based on these two points, the implemented S1-box in this paper has an area comparable to the current AES algorithm's S-box. In the linear transformation part, the paper adopts the idea of maximum distance separable (MDS) matrix decomposition, allowing the entire linear layer to be implemented with only 164 XOR gates. For the design of the addition chain, the paper employs a carry-save adder, a 32-bit adder, a single operand 31-bit adder, and intermediate registers. This design enables the linear feedback shift register layer and the finite state machine layer to share the same addition chain, further optimizing the circuit structure. The proposed hardware implementation achieves a throughput of 2 Gbps at a clock frequency of 250 MHz, with an area overhead of only 6.67 kGE, under the TSMC 90 nm process. Compared to current mainstream designs, this paper reduces area overhead by 44% while maintaining the same throughput.

Keywords ZUC, area optimization, composite domain, time division multiplexing, NB-IoT