基于 Koblitz 曲线的高能效与低时延椭圆曲线点乘架构

张金磊[1,2,3]，孙唯添[4]，蒋宇杰[1]，王安[1]，张靖奇[1]，杨先明[5]，郝越[6,7*]

（1. 北京理工大学网络空间安全学院 北京 100081；
2. 中车南京浦镇车辆有限公司 南京 210031；
3. 中车科技创新（北京）有限公司 北京 100036；
4. 西安交通大学电子科学与技术学院 西安 710049；
5. 飞腾信息技术有限公司 天津 300450；
6. 北京大学集成电路学院 北京 100871；
7. 北京微电子技术研究所 北京 100076）

摘要：针对 Koblitz 曲线在椭圆曲线密码（elliptic curve cryptography，ECC）系统中的运算效率问题，对点乘操作的硬件加速架构进行了研究。首先，提出了一种优化的 τNAF 标量转换算法及其配套硬件结构，用于缩短预处理阶段的时延。在此基础上，设计了两种兼顾流水线效率的计算架构：一是采用单乘法器和紧凑型四级流水线设计的面积高效架构，旨在提升电路资源利用率；二是针对不同二进制域（GF(2^{163})、GF(2^{283})、GF(2^{571})）分别采用两级和三级流水线设计的低时延架构，通过双乘法器并行处理以减少点加运算的时钟周期。基于 Virtex-7 FPGA 平台的实验结果表明，所提架构在 3 个典型域上均实现了显著的时延优化。相比于现有最优研究成果，面积高效架构在 GF(2^{571}) 域下的时延降低了 58.111%，低时延架构在 GF(2^{163}) 域下的时延降低了 43.952%。研究结论证实，通过优化流水线调度与算法映射，能够有效平衡椭圆曲线密码协处理器的计算性能与资源消耗，为高性能密码硬件设计提供了参考方案。

关键词：椭圆曲线密码；FPGA；Koblitz 曲线；点乘；硬件架构；流水线调度

中图分类号：TP309.7　　**文献标志码：**A

DOI: 10.20172/j.issn.2097-3136.260606

Resource-efficient and latency-optimized hardware accelerator for Koblitz-curve point multiplication

Zhang Jinlei[1,2,3], Sun Weitian[4], Jiang Yujie[1], Wang An[1],
Zhang Jingqi[1], Yang Xianming[5], Hao Yue[6,7*]

（1. School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China；
2. CRRC Nanjing Puzhen Co., Ltd., Nanjing 210031, China；
3. CRRC Science and Technology Innovation (Beijing) Co., Ltd., Beijing 100036, China；
4. School of Electronic Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China；
5. FeiTeng Information Technology Co., Ltd., Tianjin 300450, China；
6. School of Integrated Circuits, Peking University, Beijing 100871, China；
7. Beijing Microelectronics Technology Institute, Beijing 100076, China）

*** 通信作者：**E-mail: haoy@pku.edu.cn

基金项目：天津市科技计划项目（24ZGZNGX00020）

引用格式：张金磊，孙唯添，蒋宇杰，等. 基于 Koblitz 曲线的高能效与低时延椭圆曲线点乘架构 [J]. 网络空间安全科学学报，2026，4（3）：64 - 79.

Citation Format: Zhang J L，Sun W T，Jiang Y J，et al. Resource-efficient and latency-optimized hardware accelerator for Koblitz-curve point multiplication[J]. Journal of Cybersecurity，2026，4（3）：64 - 79.

Abstract: This paper investigates hardware acceleration architectures for point multiplication to address the computational efficiency issues of Koblitz curves in elliptic curve cryptography (ECC) systems. An optimized τNAF scalar conversion algorithm and its corresponding hardware structure are first proposed to reduce latency in the pre-computation phase. On this basis, two computation architectures with optimized pipeline efficiency are designed. The area-efficient architecture employs a compact four-stage pipeline with a single multiplier to improve hardware resource utilization. For different binary fields (GF(2^{163}), GF(2^{283}), and GF(2^{571})), the low-latency architecture adopts two-stage and three-stage pipeline designs respectively, and leverages dual parallel multipliers to reduce the clock cycles required for point addition. Experimental results on the Virtex-7 FPGA platform demonstrate that both architectures achieve substantial latency improvements across all three fields. Compared with state-of-the-art designs, the area-efficient architecture reduces latency by 58.111% over GF(2^{571}), while the low-latency architecture reduces latency by 43.952% over GF(2^{163}). The results demonstrate that optimizing pipeline scheduling and algorithm mapping can effectively balance the computational performance and resource consumption of ECC coprocessors, providing a practical reference for the design of high-performance cryptographic hardware.

Keywords: elliptic curve cryptography; FPGA; Koblitz curves; point multiplication; hardware architecture; pipeline scheduling

0　引言

信息安全在数字化时代的重要性日益凸显，各类加密算法在数据保护中发挥着核心作用。随着物联网（internet of things，IoT）的快速普及，在保障设备安全性能的同时实现硬件轻量化，已成为学术界与工业界的研究焦点[1]。面向物联网的认证密钥协商协议研究表明，椭圆曲线密码（elliptic curve cryptography，ECC）已被广泛用作构建安全通信信道的核心密码学原语[2]，这进一步凸显了在资源受限设备上实现高效 ECC 运算的工程价值。

椭圆曲线密码自 1985 年被 Koblitz[3] 和 Miller[4] 提出以来，凭借其能以更短的密钥提供同等安全强度的优势，在执行效率与资源消耗方面表现出显著优于其他加密算法的特性。尽管量子计算的兴起给传统加密算法带来了挑战，但由于后量子密码（post-quantum cryptography，PQC）算法目前存在公钥尺寸较大、运算效率较低等问题，ECC 在实际应用中仍是不可或缺的主流密码方案。此外，已有研究将 PQC 与 ECC 结合用于安全凭证管理系统[5]，进一步证明了 ECC 的研究意义与实践价值。在 ECC 运算体系中，点乘作为最核心的底层操作，其执行效率直接决定了整个加密系统的综合性能。

Koblitz 曲线[6] 是一类特殊的二进制域椭圆曲线，其核心特性为可利用 Frobenius 映射替代复杂的倍点运算，大幅简化点乘的计算流程。将 Koblitz 曲线的代数优势与二进制域算术运算的简洁性相结合，对于资源受限的轻量级物联网设备具有极高的工程应用价值。Frobenius 映射的计算开销极小，仅涉及二进制域内的平方运算。然而，如何高效地进行标量转换并优化硬件调度，仍是提升点乘性能的关键挑战。

针对上述挑战，通过对 Koblitz 曲线点乘运算的算法流程与硬件架构进行深度分析，提出并实现了一套高性能的点乘解决方案。本文主要工作及贡献如下。

1）算法优化层面：针对标量转换过程，提出了改进的混合输出双字标量约减算法与混合输入 τNAF（τ non-adjacent form）双字生成算法。该算法通过优化运算逻辑，消除了冗余加法操作，同时在硬件层面设计了适配的高效标量转换器架构。

2）架构设计层面：针对点加运算过程，通过精细化的数据依赖分析，设计了两种典型硬件架构：面积高效架构采用四级流水线与单乘法器的紧凑设计，实现了 100% 的硬件资源利用率；低时延架构则利用双并行乘法器及多级流水线技术，最大限度缩短了运算时钟周期。

3）实验对比层面：在 Virtex-7 FPGA 平台上完成了不同二进制域宽度的电路实现。实验结果表明，所提方案在计算时延与面积时间积（area-time product，ATP）等核心性能指标上均显著优于现有先进设计，展现出极高的工程实用价值。

1　相关工作

国内外学者针对 Koblitz 曲线点乘运算优化开展了广泛研究[7-14]。在并行计算方面，Järvinen 等[7] 与 Azarderakhsh 等[8] 通过点运算交替执行或修改计算公式，利用多个并行乘法器打破数据依赖、减少计算轮数，但引入了较大的硬件面积开销。在标量编码与预计算方面，Roy 等[9] 与 Wang 等[10] 通过扫描多位标量序列或预计算编码组合提升并行度；Järvinen 等[11] 与 Vuillaume 等[12] 则引入窗口技术或动态生成辅助点以减少运算周期，但上述方案均需占用额外硬件资源，用于预计算数据的存储与调度管理。在流水线调度方面，Li 等[13] 利用单乘法器流水线将 1-bit 序列

的计算压缩至 9 个周期；Zode 等[14]探讨了基于约束的调度方法，但该研究未引入流水线优化设计，整体运算时钟周期数偏高。

此外，由于点乘架构的底层运算逻辑具有通用性，通用曲线的优化方案[15-21]也具有重要的参考意义。在乘法器优化方面，Alharbi 等[15]采用数字并行乘法器将大位宽乘法分解为多个并行小位宽乘法，以缩短关键路径；Rashid 等[16]采用位并行 Karatsuba-Ofman 乘法器（Karatsuba-Ofman multiplier，KOM）有效缩减乘法运算时钟周期；Heidarpur 等[17]提出无重叠乘法（overlap-free Karatsuba multiplier，OFKM）策略，通过按操作数奇偶性分割数据消除递归重叠问题，进一步缩短关键路径；Thirumoorthi 等[18]与 Zeghid 等[19]则分别在 OFKM 多段划分策略及其与交错乘法技术的融合上做出了改进。然而，受限于乘法器自身的硬件实现复杂度，仅依靠底层运算单元优化对点乘整体性能的提升效果较为有限。Kumar 等[20]与 Nadikuda 等[21]则从数据调度角度出发，通过优化乘法、加法等运算单元的调度方案来提升并行度，降低计算时延。

综上所述，现有研究大多仅侧重于单一维度的性能优化，在降低计算时延的同时，通常会引入较高的硬件面积开销。现有研究针对不同乘法器配置规模与流水线级数，缺乏系统性的数据依赖分析与最优调度方案探索。因此，构建一种能充分利用流水线资源、兼顾面积效率与计算性能的点乘架构具有重要的研究价值。

2 预备知识

2.1 二进制域椭圆曲线算术

椭圆曲线密码系统的运算架构通常分为 4 个层级：最顶层为加密协议层，其次为点乘算法层，再次为点加与倍点运算层，最底层为有限域基本运算层。其中，点乘运算是实现各类密码协议的核心，其执行效率主要取决于点加与倍点运算的性能，而这些运算最终由底层的模加、模减、模乘及模逆等有限域操作完成。

本文重点研究二进制域 GF(2^m) 上的椭圆曲线标量乘法实现。在有限域元素的常用表示形式中，多项式基因可有效降低模乘运算的硬件实现复杂度，被广泛应用于 ECC 硬件设计。在多项式基表示下，各底层运算的特性如下。

1）模加与模减：在二进制域中，模加与模减运算是等价的，均可通过对两个元素的二进制系数进行按位异或操作高效实现。

2）模乘与模平方：模乘运算分为多项式乘法与模约减两个阶段。首先计算两个 m 位多项式的乘积，得到 $2m-1$ 位的中间多项式结果，再基于约减多项式 f(x) 完成模约减运算。当两个运算操作数完全一致时，多项式乘法可简化为模平方运算；在多项式基表示下，模平方运算仅需在原始系数间补零即可实现，计算开销远低于常规模乘运算。

3）模逆：这是有限域中复杂度最高、耗时最长的操作。在本文设计中，模逆运算基于费马小定理并结合改进型塔形算法（improved tower algorithm，ITA）实现。

在坐标系选择方面，仿射坐标下的每一次点加、倍点运算均包含模逆操作，会产生较大的计算时延。为了规避这一瓶颈，通常采用投影坐标进行点乘计算。在投影坐标系下，整个点乘迭代过程仅需执行模乘与模平方运算，仅在运算结束、将结果还原为仿射坐标时执行一次模逆运算，可显著提升系统整体运算性能。

2.2 Koblitz 曲线与 Frobenius 映射

Koblitz 曲线的曲线方程可定义为：

$$E_a: y^2+xy=x^3+ax^2+1, a\in\{0,1\}. \quad (1)$$

该曲线最显著的代数特性在于其 Frobenius 映射性质：若点 $P(x,y)$位于曲线 $E_a(\mathrm{GF}(2^m))$上，则点 (x^2,y^2)同样位于该曲线上。基于该性质，Frobenius 映射τ可定义为$(x,y)=(x^2,y^2)$，该映射在二进制域下仅需完成两次模平方运算，硬件实现的时间与资源开销极低。

在 Koblitz 曲线计算体系中，Frobenius 映射可等效为点 P 与复数算子τ的乘法运算，其中，τ满足特征方程$\tau^2+2=\mu\tau$，且$\mu=(-1)^{1-a}$。这一特性使得点乘运算 $Q=kp$中的整数标量k可转换为以τ为基的表达式 $\sum_{i=0}^{l-1}k_i\tau^i$（$k\in\{0,1\}$），进而将传统点乘运算重构为 $\sum_{i=0}^{l-1}k_i\tau^i(P)$的计算形式。通过该转换，点乘运算中复杂度较高的倍点操作被轻量化的 Frobenius 映射替代，整体计算复杂度转变为对多组 Frobenius 映射生成点的点加累加运算。由于 Frobenius 映射相比倍点运算在逻辑实现上更为简单，这种替代方案显著降低了点乘运算的整体复杂度，为高性能硬件架构的设计奠定了基础。

3 改进的标量算法运算与硬件架构实现

3.1 标量转换理论

在 Koblitz 曲线点乘运算中，为利用低开销的

Frobenius 映射代替高复杂度的倍点运算，需先将整数标量 k 转换成以 τ 为基的表达式。点乘运算包含 l 次 Frobenius 映射操作与若干次点加操作，其中点加运算的硬件复杂度远高于 Frobenius 映射操作。因此，优化标量转换的目标是得到长度更短且非零元素更少的序列。

Solinas 等[22]提出了 τNAF 方法，这种方法使用带符号的数字使序列变得稀疏，从而减少了运算中的非零项。然而，直接对标量连续除以 τ 生成的序列长度约为二进制序列的两倍，无法实质性降低整体计算量。

为缩短序列长度，需利用 Koblitz 曲线在 $GF(2^m)$ 有限域上的特殊代数性质。Meier 和 Staffelbach[23]的研究表明，曲线上的点 P 满足如下公式：

$$\begin{aligned} &P=(x,y)=(x^{2^m},y^{2^m})=\tau^m P\\ &(\tau^m-1)P=\mathcal{O}\end{aligned} \tag{2}$$

基于此性质，如果标量 γ 满足：

$$\gamma=\kappa\ (\mathrm{mod}\tau^m-1) \tag{3}$$

那么使用 γ 代替 κ 进行点乘的结果是完全相同的：

$$\kappa P=\lambda(\tau^m-1)P+\gamma P=\mathcal{O}+\gamma P=\gamma P \tag{4}$$

经约减后的标量 γ 生成的 τNAF 序列长度缩短至 m 左右，且非零元素比例保持在 1/3，这有效降低了点乘的复杂度。

目前最有效的约减方法是 Brumley 和 Jarvinen 提出的时延约减法[24]。该方法通过对 κ 连续除以 τ 共 m 次，最后通过一次加法得到约减结果 γ，计算过程如下：

$$\begin{aligned} &\kappa=\lambda\tau^m+(\gamma-\lambda)\\ &\gamma=\lambda+(\gamma-\lambda)\end{aligned} \tag{5}$$

在实际应用中，为了进一步提升转换速度，常采用 Adikari[25]提出的双字时延约减算法。该算法的核心是在每次迭代中直接除以 τ^2，从而将原本需要的 m 次除法运算缩减至约 m^2 次。这种改进使标量转换的时钟周期缩减了一半，显著降低了计算时延。本文后续提出的硬件架构正是基于这一双字处理逻辑进行的优化。

尽管双字处理框架已有相关研究先例，但现有实现方案均需在标量约减阶段末尾执行一次显式累加操作，将规范商与余数合并为统一数据格式后，再输入至 τNAF 生成模块。这一操作看似简单，却在硬件层面引入了额外的加法器资源，并形成了跨模块的数据依赖，限制了关键路径的进一步压缩。本文的核心改进为重新设计两个运算阶段的数据交互接口：标量约减阶段以“规范商+二进制余数序列”的混合形式直接输出数据，τNAF 生成阶段同步适配输入逻辑以接收该混合数据格式，在保障计算结果准确的前提下，彻底消除两阶段间的显式累加操作。这一改进与 Adikari 等[25]工作的本质区别在于：后者优化的是迭代次数，而本文优化的是阶段间的数据表示与接口设计，二者在优化维度上可以叠加使用。

3.2　标量转换的硬件架构与实现

本节详细介绍标量转换器的硬件实现方案。为降低硬件开销，本文对双字时延约减算法与 τNAF 生成算法进行优化，核心改进思路为重构算法数据表示形式，通过引入混合输出与混合输入机制，去除原有算法必需的累加运算步骤，有效精简电路计算单元，降低硬件面积开销并提升整体运算性能。

在标量约减阶段，算法首先执行一次除以 τ 的运算。此时，商 d_0 和 d_1 可按照 $(\mu d_0/2,-d_0/2)$ 直接更新，无需引入额外的加法操作。除法过程中产生的余数不以规范形式表示，而是直接以二进制序列的形式记录，无需进行累加。在后续每次迭代中，依次执行除以 τ^2 的运算，每轮迭代完成后，将对应的余数位 ε_0 和 ε_1 移入移位寄存器 w 缓存存储。算法最终输出的结果由规范形式的商 (d_0,d_1) 与二进制序列形式的余数 w 共同组成，两者无需相加合并，可直接作为后续 τNAF 生成阶段的输入数据。

在序列生成阶段，以约减阶段输出的规范形式商 $\kappa=d_0+d_1\tau$ 与二进制余数序列 ω 作为联合输入，每次迭代同时处理两位数据，在每轮循环中依据当前的 (c_0,c_1) 与余数序列 w 的低位联合计算输出符号位 (u_0,u_1)，并相应更新 (c_0,c_1) 与 w，逐步生成 τNAF 序列。由于在约减阶段已采用混合表示省去了额外的加法运算，序列生成过程中同样减少了所需的加法操作数量，从而有效缩短了关键路径并降低了面积开销。

基于上述改进算法，本文设计了对应的标量转换器硬件电路，整体架构如图 1 所示。

该电路的核心组成包括 4 个加减法器、3 个寄存器、若干多路选择器以及一个控制单元。控制单元产生一个 2 位控制信号 s，用于切换电路工作阶段，共包含 4 种工作状态：当 $s=0$ 时，电路处于初始化状态；当 $s=1$ 时，电路执行除以 τ 的懒惰约减运算；当 $s=2$ 时，电路执行除以 τ^2 的懒惰约减运算；当 $s=3$ 时，电路进入 τNAF 生成状态。在以下描述中，+表示逻辑或，·表示逻辑与，⊕表示异或，$a_{i,j}$ 表示 a_i 的第 j 位，以此类推。

在时延约减状态下，余数ε_0和ε_1的计算公式如下：

$$\varepsilon_0 = d_{0,0}, \varepsilon_1 = d_{0,1} \oplus d_{1,0} \tag{6}$$

商d_0和d_1的更新遵循除以τ（公式 7）和除以τ^2（公式 8）的逻辑。计算公式如下：

$$(d_0, d_1) \leftarrow (d_1 + \mu[d_0/2], -[d_0/2]) \tag{7}$$

$$(d_0, d_1) \leftarrow ((-d_0 + 2\mu d_1)/4, -(\mu d_0 + 2d_1)/4) \tag{8}$$

在如图 1 所示的架构中，电路左侧部分负责懒惰约减状态下d_0和d_1的迭代计算，右下方电路将每轮迭代产生的ε_0和ε_1依次写入移位寄存器w中存储。

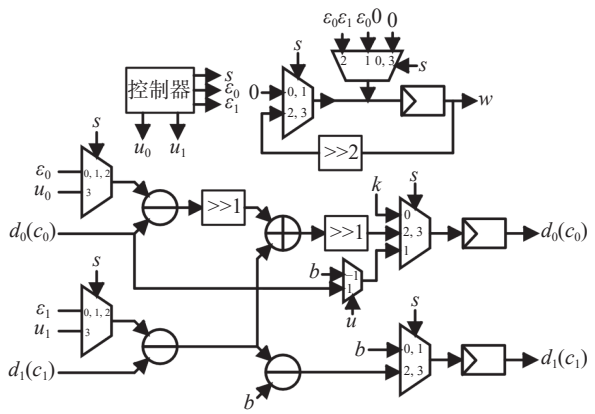

图 1 标量转换器的硬件架构

Fig. 1 Hardware architecture of the scalar converter

在τNAF 生成状态下，控制逻辑单元综合利用c_0、c_1和w计算当前的输出符号(u_0, u_1)，具体逻辑表达式如下：

$$\begin{aligned}
u_{0,0} &= c_{0,0} \oplus w_0 \\
u_{0,1} &= (c_{0,0} \oplus w_0) \cdot (c_{0,1} \oplus c_{1,0} \oplus w_1) \\
f &= \begin{cases} w_0 \cdot (c_{0,1} + \overline{c_{1,0}}) + c_{0,1} \cdot \overline{c_{1,0}}, & \mu = 1 \\ w_0 \cdot (c_{0,1} + c_{1,0}) + c_{0,1} \cdot c_{1,0}, & \mu = -1 \end{cases} \\
u_{1,0} &= \overline{c_{0,0} \oplus w_0} \cdot (c_{0,1} \oplus c_{1,0} \oplus w_1 \oplus w_0) \\
u_{1,1} &= u_{1,0} \cdot (c_{0,2} \oplus c_{1,1} \oplus w_2 \oplus f)
\end{aligned} \tag{9}$$

此时c_0和c_1的迭代更新公式为：

$$\begin{aligned}
&c_0 \leftarrow c_0 + w_0 - u_0, c_1 \leftarrow c_1 + w_1 - u_1 \\
&(c_0, c_1) \leftarrow \left(\frac{-c_0 + 2\mu c_1}{4}, \frac{-(\mu c_0 + 2c_1)}{4} \right)
\end{aligned} \tag{10}$$

寄存器w每个时钟周期右移两位，为下一轮u_0和u_1的计算提供更新后的低位输入。从电路复用的角度来看，τNAF 生成状态与懒惰约减状态共用图 1 左侧的c_0、c_1迭代电路，二者的区别仅在于顶部两个多路选择器的选通信号不同：懒惰约减状态下选通信号为ε，对应执行$c - \varepsilon$运算；τNAF 生成状态下选通信号切换为u，对应执行$c - u$运算。这一设计实现了电路资源的有效复用，降低了整体面积开销。

在 GF(2^m) 域上，上述标量转换架构的总转换时延约为$m+4$个时钟周期，具体构成如下：1 个初始化周期、1 个除以τ的约减周期、$(m-1)/2$个除以τ^2的约减周期，以及$(m+4)/2$个 τNAF 生成周期。表 1 给出了 Virtex-7 标量转换架构在 XC7VX690TFFG1157-3 芯片上的实现结果。

表 1 基于 Virtex-7 的标量转换实现结果

Table 1 Implementation result of scalar converter on Virtex-7

m	频率/MHz	CC	时延/μs	#LUT	#FF	#Slice	ATP
163	213	84	0.394	1096	503	299	118
283	162	144	0.889	1823	863	506	450
571	138	288	2.087	3550	1738	986	2058

4 点乘硬件架构设计与优化

本节围绕 Koblitz 曲线点乘硬件架构的设计与优化展开：首先分析点乘运算流程及其数据依赖关系，随后针对单乘法器与双乘法器两种配置，系统探讨不同流水线级数下的最优调度方案，并在此基础上提出面积高效架构与低时延架构两种实现方案，最后给出整体硬件架构设计。

4.1 Koblitz 曲线点乘运算流程分析

Koblitz 曲线点乘核心是将标量转换为 τNAF 形式后，以 Frobenius 映射替代传统椭圆曲线的倍点运算，仅保留点加运算作为主要计算开销，整体流程采用从最低有效位到最高有效位的扫描方式，大幅降低运算复杂度。

完成标量转换得到 τNAF 序列后，点乘运算由 Frobenius 映射与点加两类操作构成。其中，Frobenius 映射仅需完成二元域上的两次平方运算，计算开销极低；点加则是决定点乘整体时延的关键操作。为规避仿射坐标下点加需频繁执行模逆的问题，本文采用混合坐标实现点乘：点加运算在射影坐标下完成，Frobenius 映射仍在仿射坐标下执行，兼顾运算效率与实现简便性。

本文采用的混合坐标点加公式包含 8 次乘法、9 次加法、5 次平方，依据乘法操作可划分为 8 个子计算步骤，并按结果输出分为三组：第一组完成Z_3的计算，第二组完成X_3的计算，第三组完成Y_3的计算，各组间存在明确的数据依赖关系，对应的混合坐标点加运算表达式为

$$
\begin{aligned}
&\mathcal{Z}_0: E = x_2 Z_1 \\
&\mathcal{Z}_1: \begin{cases} C = Z_1(E + X_1) \\ F = aC + (E + X_1)^2 \end{cases} \\
&\mathcal{X}_0: G = y_2 Z_1^2 \\
&\mathcal{X}_1: X_3 = C(F + G + Y_1) + (G + Y_1)^2 \\
&\mathcal{Y}_0: H = C(G + Y_1) + Z_3 \\
&\mathcal{Y}_1: D = x_2 Z_3 \\
&\mathcal{Y}_3: Y_3 = H(D + X_3) + J
\end{aligned}
\tag{11}
$$

Koblitz 曲线点乘运算先将输入标量完成规约与 τNAF 生成，随后从低位到高位逐位扫描 τNAF 序列，当位非零时执行点加操作，每轮迭代均对当前点执行 Frobenius 映射，最终通过一次模逆将射影坐标结果转换回仿射坐标输出。

采用从右至左的计算方式，标量转换中的规约完成后即可逐位输出 τNAF 序列，点乘可与 τNAF 生成并行执行，大幅压缩整体时延。同时，Frobenius 映射与点加可并发处理：当 τNAF 位为 0 时，仅执行 Frobenius 映射；当位非零时，存储当前点用于点加，同时持续执行后续 Frobenius 映射，仅在点加未完成且新非零位到来时暂停映射，待点加结束后恢复。由于点加时延远大于 Frobenius 映射，后者带来的时钟开销可忽略不计，点乘总时延完全由点加运算决定。

4.2　流水线调度与架构分析

由 4.1 节的分析可知，点乘时延主要取决于点加运算的计算效率，而乘法运算是点乘中最复杂、占用资源最多的操作。本文采用流水线 KOM，通过在乘法器内部插入寄存器，将复杂的组合逻辑均匀分割，以缩短关键路径、提升时钟频率。在此基础上，本文系统分析了单乘法器与双乘法器架构在不同流水线级数下的点乘性能，通过重新设计点乘数据调度方案，确定各架构配置下点乘所需的最小时钟周期数。

4.2.1　单乘法器架构分析

对于单乘法器（1M）架构，每轮点乘理论上至少需要 8 个时钟周期来完成 8 次乘法运算。然而由于点乘各子计算单元之间存在数据依赖关系，实际所需周期数可能超过理论下界。具体而言，Z_1中C的计算依赖Z_0中E的结果，而下一轮Z_0中E的计算又依赖当前轮Z_1输出的Z_3，因此Z_0与Z_1之间存在严格的先后约束，这一依赖关系决定了每轮点乘实际消耗的时钟周期数。经分析，在流水线级数不超过 4 级时（即单次乘法所需周期不超过 4 个），流水线点乘仍可达到理论最小周期数 8；而当流水线达到 5 级时，受Z_0与Z_1数据依赖的制约，每轮点乘所需周期增至 10 个，流水线中出现气泡，总周期数随之上升。此后随流水线级数继续增加，所需周期数将线性增长。

在 2 级、3 级和 4 级流水线设计中，Z_0与Z_1消耗的总时钟周期均不超过理论最小值 8，点乘每轮需 8 个时钟周期，流水线中无气泡产生。在 GF(2^m) 域上，点乘过程中点乘次数约为$m/3$，因此点乘所需总时钟周期约为$8m/3$。各子计算单元中，Z_0、Z_1和X_0需优先执行，X_0的计算不受这两步结果约束，可与之重叠。在 2 级流水线设计中，由于流水线级数较少，前一轮点乘对当前轮的影响较小，X_0可插入Z_0与Z_1之间；而当流水线级数增加时，为避免与前一轮产生冲突，X_0需移至Z_1之后执行。X_1、Y_0、Y_1和Y_2须在Z_1与X_0均完成后执行，Y_3则须在X_1、Y_0和Y_1均完成后执行。

4.2.2　双乘法器架构分析

对于双乘法器（2M）架构，由于两个乘法器并行运行，每轮点乘的理论最小周期数降至 4。Z_0与Z_1之间的数据依赖关系仍然存在，其执行顺序保持不变，但X_0可与Z_1并行计算。

在 2 级流水线设计中，X_0 在 Z_1 完成后启动，随后计算 Y_1，X_1 与 Y_0 在 X_0 完成后并行执行，最后计算 Y_2 和 Y_3，每轮点乘可达到理论最小周期数 4。在 3 级和 4 级流水线设计中，X_0 与 Z_1 并行执行，Y_0 在 X_0 完成后启动，X_1 与 Y_1 并行计算，最后依次执行 Y_2 和 Y_3；但受 Z_0 与 Z_1 数据依赖的制约，3 级和 4 级设计每轮点乘所需周期分别增至 6 和 8，无法达到理论最小值。在 GF(2^m) 域上，2M 架构 3 种流水线设计所需的点乘总时钟周期分别约为 $4m/3$、$2m$ 和 $8m/3$。

随着乘法器数量的进一步增加，流水线紧凑性将难以保证，面积开销也将急剧上升，因此本文仅考虑单乘法器与双乘法器两种架构。

4.2.3　实验对比与架构选择

本节将上述 7 种点乘方案在 XC7VX690TFFG1157-3 芯片上进行综合实现，结果如表 2 所示。

在单乘法器架构中，随流水线级数增加，系统时钟频率提升，当级数不超过 4 时，点乘总周期数不变，故总时延随级数增加而降低；当级数达到 5 级时，虽时钟频率继续上升，但总周期数增大，导致时延反而增加，时延最低点出现在 4 级流水线设计。在双乘法器架构中，时钟频率同样随流水线级数增加而提升，2 级设计可达到理论最小周期数，而 3 级和 4 级设计的周期数逐渐增大。对于 GF(2^{163}) 域，最低时延出现在 2 级流水线设计；对于 GF(2^{283}) 和 GF(2^{571}) 域，最低时延则出现在 3 级流水线设计。

图 2 展示了所有方案在 3 个二元域上的时延、面积（Slice 数量）和 ATP 的变化趋势。

总体而言，双乘法器架构与单乘法器架构的时钟频率差异不显著，但双乘法器架构的总时钟周期

表 2 基于 Virtex-7 的点乘实现结果

Table 2 Implementation result of point multiplication on VIRTEX-7

域长	架构	流水线	频率/MHz	周期数	时延/μs	LUT	Slice	ATP
163	1M	2S	192.2	435	2.26	11007	3154	7137
		3S	293.3	435	1.483	11639	3379	5012
		4S	349.7	435	1.244	11791	3365	4186
		5S	400.0	543	1.358	11798	3297	4476
	2M	2S	203.4	217	1.067	19448	5390	5750
		3S	291.4	326	1.119	20474	5702	6380
		4S	350.9	435	1.240	20560	5762	7143
283	1M	2S	162.3	755	4.651	22637	6886	32025
		3S	245.7	755	3.073	23656	6622	20348
		4S	294.1	755	2.567	23445	6628	17014
		5S	337.8	943	2.791	25065	7258	20259
	2M	2S	157.2	377	2.398	42737	11619	27859
		3S	238.1	566	2.377	45685	12610	29976
		4S	305.8	755	2.469	44447	11961	29530
571	1M	2S	115.7	1523	13.159	61635	16716	219961
		3S	196.9	1523	7.737	65348	19401	150102
		4S	227.3	1523	6.701	63842	18850	126318
		5S	264.6	1903	7.193	65612	19998	143852
	2M	2S	114.2	761	6.666	125526	33891	225930
		3S	180.5	1142	6.327	124262	33982	214993
		4S	238.1	1523	6.397	123415	35921	229772

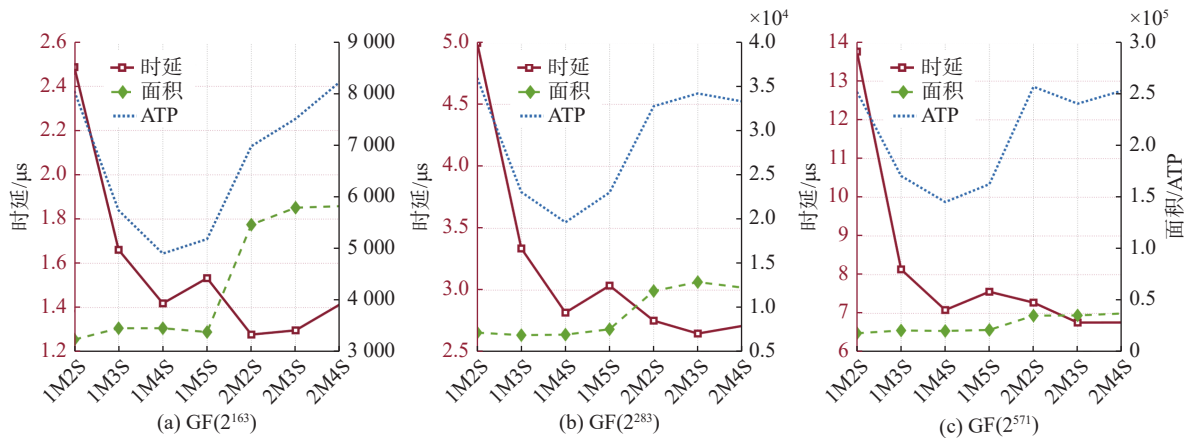

图 2 点乘实验结果

Fig. 2 Implementation results in point multiplication architectures

数普遍低于单乘法器架构，因此整体最低时延出现在双乘法器架构中。在面积方面，双乘法器架构因额外引入一个乘法器，切片资源占用显著大于单乘法器架构。采用 ATP 衡量时延与面积之间的折中关系，由于流水线级数对面积影响较小，ATP 的变化趋势与时延基本一致。尽管双乘法器架构具有更低的时延，但其面积大幅增加，导致 ATP 最小值仍出现在单乘法器架构中。

4.2.4　面积高效架构与低时延架构

基于上述分析，本文提出两种点乘架构：面积高效架构与低时延架构。面积高效架构以最低 ATP 为目标，采用单乘法器结合 4 级流水线设计，在降低计算时延的同时保持较低的面积占用，具有较高的面积利用效率。低时延架构以最低点乘时延为目标，采用双乘法器并行设计，以面积资源换取更低的计算时延；对于 GF(2^{163}) 域采用 2 级流水线，对于 GF(2^{283}) 和 GF(2^{571}) 域则采用 3 级流水线。

面积高效型架构的算子调度方案如图 3 所示，其中，红色高亮部分标记点乘的计算结果。

4 级流水线设计以单个乘法单元、两个加法单元和一个平方单元，在 8 个时钟周期内完成点乘所需的 8 次乘法、10 次加法和 5 次平方运算。需要额外说明的是，由于数据依赖和关键路径的约束，实现中引入了一次额外加法操作：$F+G+Y_1$结果需在第 10 个时钟周期才被使用，而F在第 8 周期可用、G在第 9 周期可用，因此需在第 8 周期先计算$F+Y_1$，在第 9 周期再计算$(F+Y_1)+G$；又因后续计算还需$G+Y_1$，实际共需 3 次加法，比理论最小值多 1 次。此外，Frobenius 映射计算还需额外 2 个平方单元，一次完整点乘共需 19 个时钟周期，X_3、Y_3和Z_3分别在第 13、19 和 8 个时钟周期输出。4 级流水线乘法单元将在输入数据后的最后 3 个时钟周期内输出结果，乘法结果将存入对应寄存器或直接送入加法单元。

面积高效架构的关键路径出现在点乘计算过程中，涉及一个多路选择器和已流水化的乘法器。通过合理为各计算单元（乘法器、加法器等）分配寄存器，可减小多路选择器的输入规模，进一步优化关键路径。表 3 给出了面积高效型架构点乘部分的寄存器级调度方案。

点乘单元共需 9 个寄存器，乘法器输入端的多路选择器为 5 选 1。基于上述算子调度与寄存器分配方案，面积高效型硬件架构如图 4 所示。图中寄存器组用于存储计算过程中的中间结果和最终结果，同时作为流水线架构的第一级流水寄存器使用。

低时延架构由 2 级和 3 级两种流水线的双乘法器架构构成，2 级设计适用于 GF(2^{163}) 域，3 级设计适用于 GF(2^{283}) 和 GF(2^{571}) 域，其算子调度方案分别如图 5 和图 6 所示，红色高亮部分同样标记点乘的计算结果。

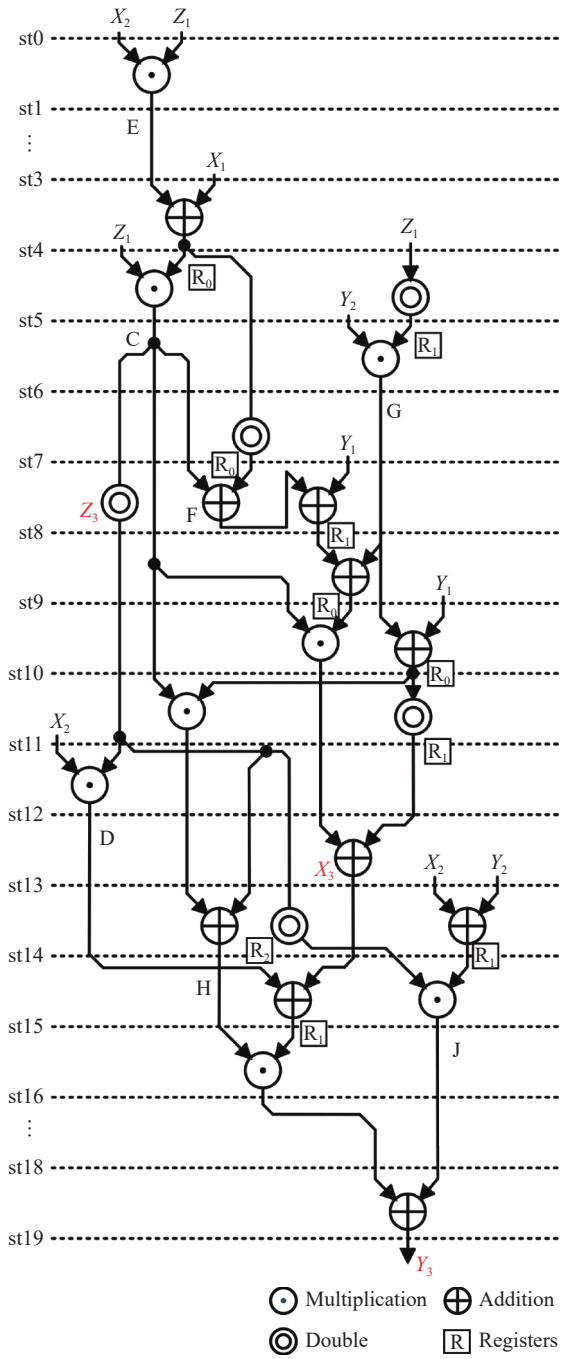

图 3　面积高效型架构的算子调度方案

Fig. 3　Operator scheduling scheme of the area-efficient architecture

2 级流水线设计以两个乘法单元、3 个加法单元和两个平方单元，在 4 个时钟周期内完成点乘所需的 8 次乘法、9 次加法和 5 次平方运算，并额外配置

表 3 面积高效型架构点乘部分寄存器级调度方案

Table 3 Register-level scheduling scheme of dot product for area-efficient architecture

时钟周期	乘法器		加法器				平方器
	MUL_0	MUL_1	ADD0_0	ADD0_1	ADD1_0	ADD1_1	SQR
1	$x_2^{(0)}$	$Z_1^{(0)}$	$R_1^{(-1)}$	$G^{(-1)}$	—	—	—
2	$\mathrm{RCH}^{(-1)}$	$R_0^{(-1)}$	$G^{(-1)}$	$Y_1^{(-1)}$	—	—	—
3	$\mathrm{RCH}^{(-1)}$	$R_0^{(-1)}$	$\mathrm{MUL\_r}^{(-2)}$	$J^{(-2)}$	—	—	$R_0^{(-1)}$
4	$x_2^{(-1)}$	$Z_3^{(-1)}$	$\mathrm{MUL\_r}^{(0)}$	$X_1^{(0)}$	—	—	—
5	$Z_1^{(0)}$	$R_0^{(0)}$	$\mathrm{MUL\_r}^{(-1)}$	$R_1^{(-1)}$	—	—	$Z_1^{(0)}$
6	$y_2^{(0)}$	$R_1^{(0)}$	$\mathrm{MUL\_r}^{(-1)}$	$R_1^{(-1)}$	$x_2^{(-1)}$	$y_2^{(-1)}$	$Z_3^{(-1)}$
7	$R_2^{(-1)}$	$R_1^{(-1)}$	$\mathrm{MUL\_r}^{(-1)}$	$X_3^{(-1)}$	—	—	$R_0^{(0)}$
8	$\mathrm{RCH}^{(-1)}$	$R_1^{(-1)}$	$\mathrm{MUL\_r}^{(0)}$	$Y_1^{(0)}$	$\mathrm{RCH}^{(0)}$	$R_0^{(0)}$	$\mathrm{RCH}^{(0)}$
9	$x_2^{(1)}$	$Z_1^{(1)}$	$R_1^{(0)}$	$G^{(0)}$	—	—	—
10	$\mathrm{RCH}^{(0)}$	$R_0^{(0)}$	$G^{(0)}$	$Y_1^{(0)}$	—	—	—
11	$\mathrm{RCH}^{(0)}$	$R_0^{(0)}$	$\mathrm{MUL\_r}^{(-1)}$	$J^{(-1)}$	—	—	$R_0^{(0)}$
12	$x_2^{(0)}$	$Z_3^{(0)}$	$\mathrm{MUL\_r}^{(1)}$	$X_1^{(1)}$	—	—	—
13	$Z_1^{(1)}$	$R_0^{(1)}$	$\mathrm{MUL\_r}^{(0)}$	$R_1^{(0)}$	—	—	$Z_1^{(1)}$
14	$y_2^{(1)}$	$R_1^{(1)}$	$\mathrm{MUL\_r}^{(0)}$	$R_1^{(0)}$	$x_2^{(0)}$	$y_2^{(0)}$	$Z_3^{(0)}$
15	$R_2^{(0)}$	$R_1^{(0)}$	$\mathrm{MUL\_r}^{(0)}$	$X_3^{(0)}$	—	—	$R_0^{(1)}$
16	$\mathrm{RCH}^{(0)}$	$R_1^{(0)}$	$\mathrm{MUL\_r}^{(1)}$	$Y_1^{(1)}$	$\mathrm{RCH}^{(1)}$	$R_0^{(1)}$	$\mathrm{RCH}^{(1)}$
17	$x_2^{(2)}$	$Z_1^{(2)}$	$R_1^{(1)}$	$G^{(1)}$	—	—	—
18	$\mathrm{RCH}^{(1)}$	$R_0^{(1)}$	$G^{(1)}$	$Y_1^{(1)}$	—	—	—
19	$\mathrm{RCH}^{(1)}$	$R_0^{(1)}$	$\mathrm{MUL\_r}^{(0)}$	$J^{(0)}$	—	—	$R_0^{(1)}$

$^{(-2)}$表示上上一轮；$^{(-1)}$表示上一轮；$^{(0)}$表示当前轮；$^{(1)}$表示下一轮；$^{(2)}$表示下下一轮。

2 个平方单元用于 Frobenius 映射计算。一次完整点乘需 9 个时钟周期，X_3、Y_3 和 Z_3 分别在第 7、9 和 5 个时钟周期输出；2 级流水线乘法单元在接收输入数据后的下一个时钟周期即可输出结果。3 级流水线设计以两个乘法单元、两个加法单元和两个平方单元，在 6 个时钟周期内完成相应的乘法、加法和平方运算，同样配置 2 个额外平方单元用于 Frobenius 映射。一次完整点乘需 13 个时钟周期，X_3、Y_3 和 Z_3 分别在第 10、13 和 6 个时钟周期输出；3 级流水线乘法单元在接收输入数据后的两个时钟周期内输出结果。

2 级流水线点乘单元共需 12 个寄存器，两个乘法器输入端的多路选择器均为 4 选 1；3 级流水线点乘单元共需 9 个寄存器，两个乘法器输入端的多路选择器均为 3 选 1。两种低时延硬件架构分别如图 7 和图 8 所示。

4.3 整体硬件架构

本文所提 Koblitz 曲线点乘运算的整体硬件架构如图 9 所示，主要分为标量转换和点乘计算两部分。

标量转换部分实现了将标量 k 转换为以 τ 为基的 τNAF 表示的标量转换算法；点乘计算部分主要由 ALU 模块构成，包含用于 Frobenius 映射、点乘及模

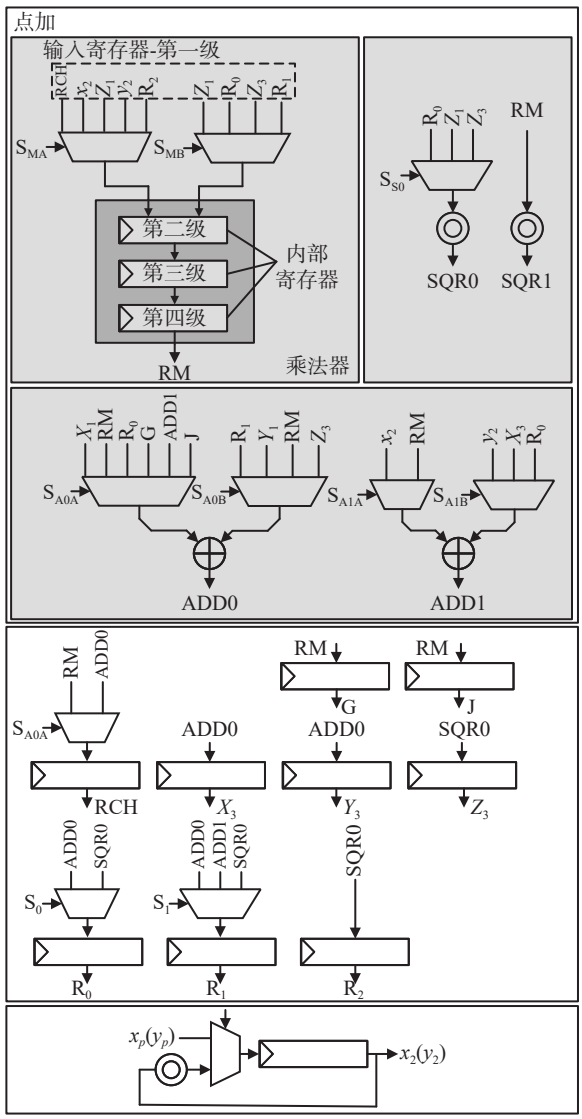

图 4　面积高效型的硬件架构

Fig. 4　Hardware architecture diagram of the area-efficient architecture

逆运算的乘法、加法等基本运算单元，以及负责管理点乘流程控制逻辑的控制单元和存储计算结果的寄存器组。由于标量转换与点乘计算两部分的时钟频率不同，采用 FIFO 进行跨时钟域转换。标量转换模块完成规约后，以每周期两位的速率生成 τNAF 序列并写入 FIFO；点乘计算模块检测到 FIFO 非空后即开始读取序列并启动计算。从实现结果来看，标量转换模块的时钟频率略低于点乘计算模块，但由于标量转换采用双字算法，其数据输出速率高于后续点乘计算的需求，因此不会影响后续运算的正常推进。

5　实验结果与对比分析

本节对所提两种 Koblitz 曲线点乘硬件架构进行

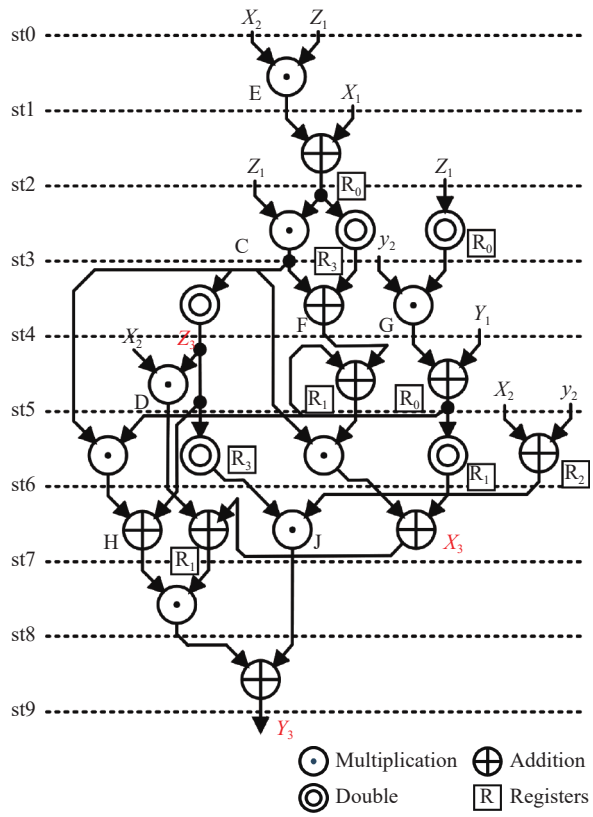

图 5　2 级流水线的低时延架构算子调度方案

Fig. 5　Operator scheduling scheme of the low-latency architecture with two-stage pipeline

FPGA 实现与性能评估，给出完整的时延、资源占用、ATP 及功耗数据，并与当前主流设计进行横向对比，验证所提架构的技术优势。

5.1　评估指标与实现平台

Koblitz 曲线点乘总时延由 τNAF 标量转换与点乘计算两部分组成，分别对应不同时钟频率与周期数，总时延计算公式为：

$$\text{Latency} = T_\tau \times \text{CC}_\tau + T_{\text{KP}} \times \text{CC}_{\text{KP}} \quad (12)$$

其中，T_τ、CC_τ为标量转换的时钟周期与周期数；T_{KP}、CC_{KP}为点乘计算的时钟周期与周期数。

在 GF(2^m)域中，本文 τNAF 算法总周期数约为 $m+5$，可与点乘并行执行，实际有效周期约为 $(m+5)/2$。点乘周期主要由点加决定，单乘法器架构每轮点加 8 周期，双乘法器架构在 GF(2^{163})为 4 周期，在 GF(2^{283})、GF(2^{571})为 6 周期，最终加上模逆周期得到总周期数。

硬件资源占用以 FPGA Slice 数量衡量，综合性能采用 ATP 评估，计算公式为：

$$\text{ATP} = \text{Slice} \times \text{Latency} \quad (13)$$

本文采用 Xilinx Virtex-7 FPGA 平台，以 Vivado

2 019.1 完成综合与实现，分别使用性能优化策略与额外时序优化策略提升电路频率与时序表现。为便于与其他结构进行对比，同时在 Virtex-5、Virtex-4 器件上进行移植实现。实验结果对比见表 4。

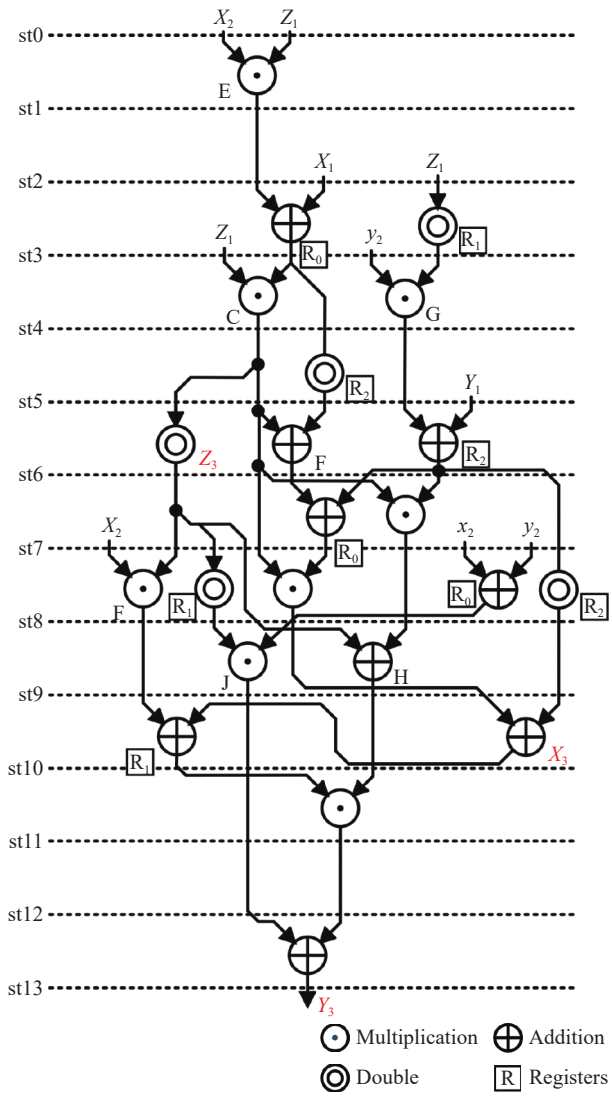

图 6 3 级流水线的低时延架构算子调度方案

Fig. 6 Operator scheduling scheme of the low-latency architecture with three-stage pipeline

5.2 实现结果

面积高效架构采用单乘法器 4 级流水线，在 GF(2^{163})、GF(2^{283})、GF(2^{571})3 个典型二元域上的点乘总时延分别为 1.683 μs、3.455 μs、7.511 μs，对应 Slice资源占用为 3 631、7 867、20 612。

低时延架构采用双乘法器并行结构，在 GF(2^{163})使用 2 级流水线，在GF(2^{283})、GF(2^{571})使用 3 级流水线，3 个域上时延分别为 1.347 μs、3.279 μs、7.071 μs，对应Slice占用为 6 026、14 246、38 515。

5.3 对比分析

将本文所提两种架构与近年来面向 Koblitz 曲线的点乘设计、面向常规椭圆曲线的高性能点乘设计进行横向对比，对比对象包括文献 [13, 19-20, 26-28] 等主流方案，对比维度覆盖时延、资源占用、面积–时延积等关键指标。

在 Koblitz 曲线点乘相关设计中，文献 [13] 与文献 [26] 均包含完整 τNAF 标量转换与点乘运算，其中，文献 [13] 在同类设计中具有相对更优的时延与资源表现，其在 Virtex-5 平台上实现的GF(2^{163})域点乘时延为 2.505 μs。与该结果相比，本文面积高效架构时延降低 21.158%，面积占用增加 18.856%，面积–时延积降低 6.287%；低时延架构时延降低 43.952%，面积占用增加 88.474%，面积–时延积增加 5.646%。

在GF(2^{163})域上，本文面积高效架构时延较现有最优结果降低 21.158%，低时延架构时延降低 43.952%；在 GF(2^{283})域上，面积高效架构时延降低 38.985%，低时延架构时延降低 42.459%；在 GF(2^{571})域上，面积高效架构时延降低 58.111%，低时延架构时延降低 59.720%，在 3 个典型域上均取得显著的时延优化效果。

文献 [19] 与文献 [27] 在时延指标上同样具有较强竞争力，其中文献 [19] 在 Virtex-7 平台上 GF(2^{163})域时延为 2.450 μs，文献 [27] 为 2.068 μs，但上述两种设计的硬件资源占用量更大，导致面积–时延积显著高于本文所提架构。整体对比结果表明，本文面积高效架构在面积、时延与 ATP 之间实现最优平衡，适用于资源受限的物联网终端设备；低时延架构实现全域最低计算时延，适用于对实时性要求严苛的高速加密场景。

与面向常规椭圆曲线的面积高效设计相比，例如，文献 [20] 在 Virtex-7 平台上实现的 GF(2^{163})域点乘时延为 3.710 μs，占用 2 437 个 Slice，ATP为 9 041。本文面积高效架构时延降低 54.636%，面积占用增加 48.995%，ATP降低 32.419%；低时延架构时延降低 63.693%，面积占用增加 147.271%，ATP降低 10.242%。与文献 [28] 相比，其在 Virtex-7 平台上 GF(2^{163})域时延为 2.510 μs，占用 3 422 个 Slice，ATP为 8 591。本文面积高效架构时延降低 32.948%，面积占用增加 6.108%，ATP降低 28.879%；低时延架构时延降低 46.335%，面积占用增加 76.096%，ATP降低 5.541%。

实验同时记录了硬件功耗，如表 5 所示。随着域阶数增大，系统总功耗上升；相同域下，单乘法器面积高效架构的资源利用率更高、功耗更低。

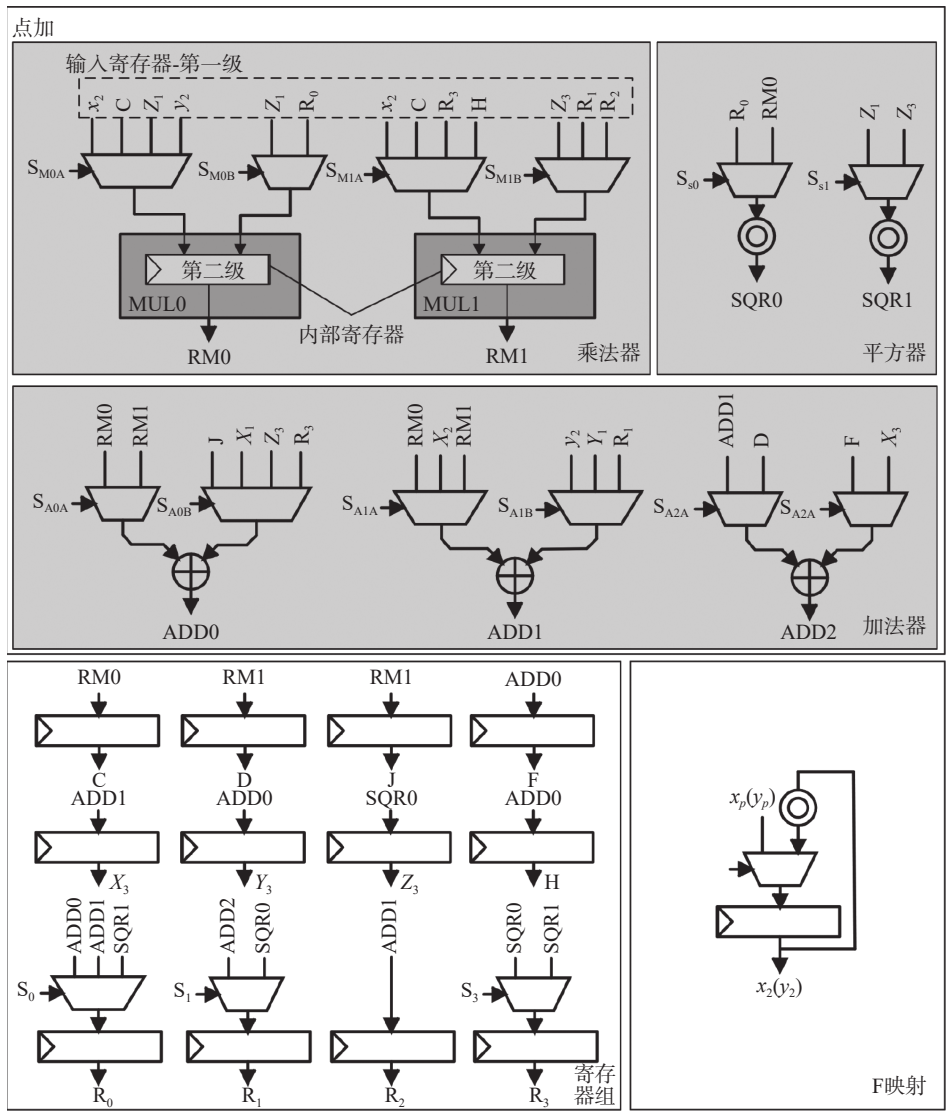

图 7　2 级流水线的低时延硬件架构

Fig. 7　Hardware architecture diagram of the low-latency architecture with two-stage pipeline

此外，部分对比文献未单独标注标量转换模块的时延与资源占用，相关开销已计入整体系统的总时延与面积，本文在对比过程中统一采用完整系统指标，保证对比结果的公平性与有效性。

6　结束语

本文针对 Koblitz 曲线点乘运算的硬件实现开展了系统性研究。在算法层面，提出了基于混合表示的双字懒惰规约与 τNAF 生成算法，通过消除冗余加法操作降低了关键路径时延并缩减了电路面积。在架构层面，对单乘法器与双乘法器在不同流水线级数下的调度方案进行了系统分析，确定了各配置下的最小时钟周期数，并据此提出面积高效架构与低时延架构两种实现方案：前者采用单乘法器 4 级流水线，流水线效率达 100%，在面积与时延之间实现最优折中；后者采用双乘法器设计，在文中 3 个典型二进制域上均取得最优计算时延。上述成果验证了通过流水线调度与算法协同优化能够有效平衡密码协处理器的计算性能与资源消耗，为资源受限的物联网设备与高速密码硬件设计提供了参考方案。

在未来研究中，可重构架构设计是重要的拓展方向，通过支持多域参数的灵活配置，可进一步提升架构通用性、拓宽适用范围。与此同时，在保持高性能的前提下引入抗侧信道攻击防护机制，如随机化中间值与均衡功耗等防护手段，相关优化设计是满足实际安全部署需求的关键课题。此外，随着后量子密码与 ECC 混合应用场景的不断拓展，面向两类算法的协同加速架构设计也值得深入探索。

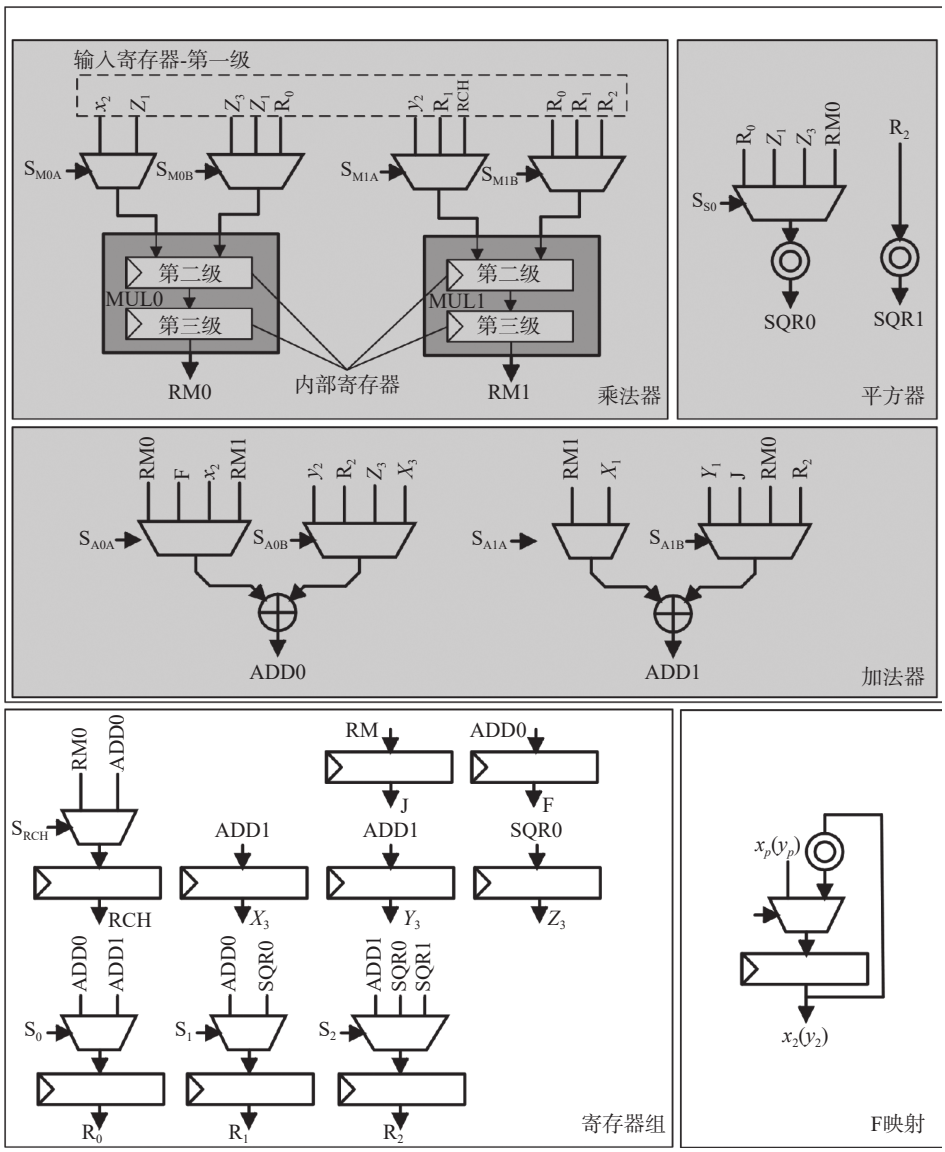

图 8 3 级流水线的低时延架构硬件架构

Fig. 8 Hardware architecture diagram of the low-latency architecture with three-stage pipeline

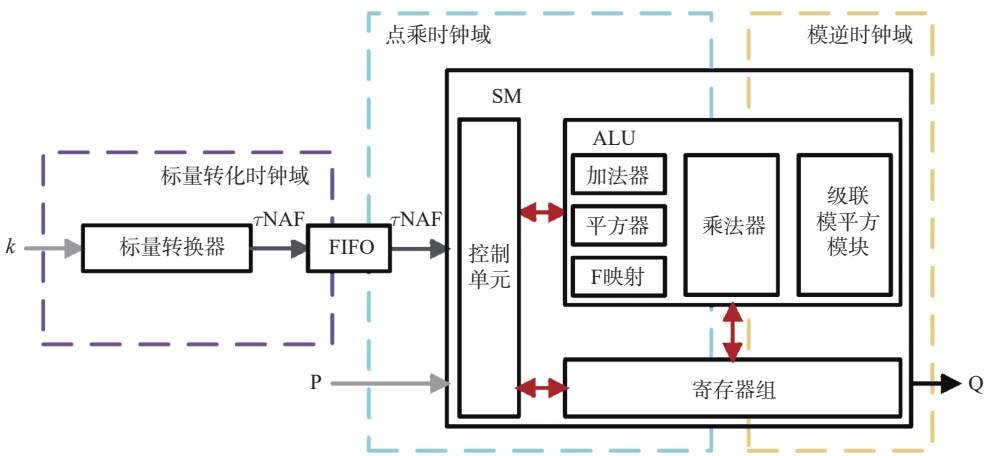

图 9 Koblitz 曲线点乘运算的整体硬件架构

Fig. 9 Overall hardware architecture of point multiplication on Koblitz curve

表 4　实验结果对比

Table 4　Comparison of experimental results

曲线	文献	域(m)	架构	设备	总计		
					时延/μs	#Slice	ATP
Koblitz曲线	[13]	163	—	V5	2.505	3670	9193
			—	V4	3.402	7732	26302
		283	—	V5	5.815	7738	44993
		571	—	V5	18.513	20291	375642
	[26]	163	—	V5	5.262	23975	126158
	本文	163	1M	V7	1.683	3631	6110
				V5	1.975	4362	8615
				V4	2.955	5571	16462
			2M	V7	1.347	6026	8115
				V5	1.404	6917	9712
				V4	2.092	9019	18868
		283	1M	V7	3.455	7867	27184
				V5	3.548	9027	32029
				V4	5.500	12379	68086
			2M	V7	3.279	14246	46713
				V5	3.346	14807	49544
				V4	5.408	19597	105974
		571	1M	V7	7.511	20612	154821
				V5	7.755	24625	190962
				V4	12.199	32630	398065
			2M	V7	7.071	38515	272325
				V5	7.457	42827	319380
				V4	11.349	55726	632429
General曲线	[19]	163	d=4	V7	2.930	4435	12995
			d=6	V7	2.450	5705	13977
		283	d=4	V7	8.240	7096	58471
			d=6	V7	6.750	8951	60419
		571	d=4	V7	28.800	13789	397123
			d=6	V7	23.020	16359	376584
	[27]	163	—	V7	2.068	8762	18122
		283	—	V7	4.095	20451	83739
		571	—	V7	9.719	41974	407950
	[20]	163	—	V7	3.710	2437	9041
				V5	5.220	2502	13062
		283	—	V7	7.690	5493	42240
				V5	10.790	5640	60859
	[28]	163	∞	V7	2.510	3422	8591
			1	V7	4.832	3422	16536
		283	∞	V7	4.932	7983	39374
			1	V7	9.331	7983	74487
		571	∞	V7	10.851	20158	218732
			1	V7	20.377	20158	410763
	[29]	163	—	V5	3.900	3590	14002

表 5 本文架构在 Virtex-7 上的功耗

Table 5 Power consumption of the proposed architectures on Virtex-7

域（m）	架构	τNAF的功率/W	KP的功率/W	总功率/W
163	1M	0.383	1.620	2.003
	2M	0.383	2.693	3.076
283	1M	0.399	2.979	3.378
	2M	0.399	8.679	9.078
571	1M	0.416	8.938	9.354
	2M	0.416	19.188	19.604

参考文献：

[1] Garg S，Kaur K，Kaddoum G，et al. Toward secure and provable authentication for internet of things：realizing industry 4.0[J]. IEEE Internet of Things Journal，2020，7（5）：4598-4606.

[2] 王凯，董建阔，肖甫，等. 面向物联网的认证密钥协商协议研究综述 [J]. 网络空间安全科学学报，2024，2（5）：2-16.

Wang K，Dong J K，Xiao F，et al. Review of research on authentication key agreement protocols for internet of things[J]. Journal of Cybersecurity，2024，2（5）：2-16.

[3] Koblitz N. Elliptic curve cryptosystems[J]. Mathematics of Computation，1987，48（177）：203-209.

[4] Miller V S. Use of elliptic curves in cryptography[M]//Advances in Cryptology — CRYPTO '85 Proceedings. Berlin，HeidelbergSpringer2007：417-426.

[5] Chen A C H，Lin B Y. Hybrid scheme of post-quantum cryptography and elliptic-curve cryptography for certificates — a case study of security credential management system in vehicle-to-everything communications[C]//Proceedings of the 2024 7th International Conference on Circuit Power and Computing Technologies（ICCPCT）. Piscataway：IEEE Press，2024：426-430.

[6] Koblitz N. CM-curves with good cryptographic properties[M]//Advances in Cryptology — CRYPTO '91. Berlin，HeidelbergSpringer，2007：279-287.

[7] Järvinen K U，Skyttä J O. Fast point multiplication on Koblitz curves：parallelization method and implementations[J]. Microprocessors and Microsystems，2009，33（2）：106-116.

[8] Azarderakhsh R，Reyhani M A. High-performance implementation of point multiplication on Koblitz curves[J]. IEEE Transactions on Circuits and Systems II：Express Briefs，2013，60（1）：41-45.

[9] Roy S S，Rebeiro C，Mukhopadhyay D. A parallel architecture for Koblitz curve scalar multiplications on FPGA platforms[C]//Proceedings of the 2012 15th Euromicro Conference on Digital System Design. Piscataway：IEEE Press，2012：553-559.

[10] Wang T，Liu T T. ECC processor over the Koblitz curves with τ-NAF converter and square-square-add algorithm[C]//Proceedings of the 2020 IEEE Asia Pacific Conference on Circuits and Systems（APCCAS）. Piscataway：IEEE Press，2020：23-26.

[11] Järvinen K U，Skyttä J O. High-speed elliptic curve cryptography accelerator for Koblitz curves[C]//Proceedings of the 2008 16th International Symposium on Field-Programmable Custom Computing Machines. Piscataway：IEEE Press，2008：109-118.

[12] Vuillaume C，Okeya K，Takagi T. Short-memory scalar multiplication for Koblitz curves[J]. IEEE Transactions on Computers，2008，57（4）：481-489.

[13] Li L J，Li S G. High-performance pipelined architecture of point multiplication on Koblitz curves[J]. IEEE Transactions on Circuits and Systems II：Express Briefs，2018，65（11）：1723-1727.

[14] Zode P，Deshmukh R. Optimization of elliptic curve scalar multiplication using constraint based scheduling[J]. Journal of Parallel and Distributed Computing，2022，167：232-239.

[15] Alharbi A R，Hazzazi M M，Jamal S S，et al. DCryp-unit：crypto hardware accelerator unit design for elliptic curve point multiplication[J]. IEEE Access，2024，12：17823-17835.

[16] Rashid M，Sonbul O S，Zia M Y I，et al. Throughput/area-efficient accelerator of elliptic curve point multiplication over GF（2^{233}） on FPGA[J]. Electronics，2023，12（17）：3611.

[17] Heidarpur M，Mirhassani M. An efficient and high-speed overlap-free karatsuba-based finite-field multiplier for FGPA implementation[J]. IEEE Transactions on Very Large Scale Integration（VLSI）Systems，2021，29（4）：667-676.

[18] Thirumoorthi M，Leigh A J，Heidarpur M，et al. Novel formulations of M-term overlap-free karatsuba binary polynomial multipliers and their hardware implementations[J]. IEEE Transactions on Very Large Scale Integration（VLSI）Systems，2023，31（10）：1509-1522.

[19] Zeghid M，Ahmed H Y，Chehri A，et al. Speed/area-efficient ECC processor implementation over GF（2^m） on FPGA via novel algorithm-architecture co-design[J]. IEEE Transactions on Very Large Scale Integration（VLSI）Systems，2023，31（8）：1192-1203.

[20] Kumar N R，Shirisha C. ECC architecture over GF（2^m） for resource-constrained applications[J]. AEU - International Journal of Electronics and Communications，2020，125：153383.

[21] Nadikuda P K G，Boppana L. An area-time efficient point-multiplication architecture for ECC over GF（2^m） using polynomial basis[J]. Microprocessors and Microsystems，2022，91：104525.

[22] Solinas J A. Efficient arithmetic on Koblitz curves[M]// Towards a Quarter-Century of Public Key Cryptography. Boston，MA：Springer US，2000：125-179.

[23] Meier W，Staffelbach O. Efficient multiplication on certain nonsupersingular elliptic curves[C]//Advances in Cryptology — CRYPTO' 92. Berlin，Heidelberg：Springer，1993：333-344.

[24] Brumley B B，Jarvinen K U. Conversion algorithms and implementations for Koblitz curve cryptography[J]. IEEE Transactions on Computers，2010，59（1）：81-92.

[25] Adikari J，Dimitrov V S，Jarvinen K U. A fast hardware architecture for integer to τNAF conversion for Koblitz curves[J]. IEEE Transactions on Computers，2012，61（5）：732-737.

[26] Tian X M，Ding R，Wu X J，et al. Hardware implementation of a cryptographically secure pseudo-random number generators based on Koblitz elliptic curves[C]//Proceedings of the 2020 IEEE 3rd International Conference on Electronics Technology（ICET）. Piscataway：IEEE Press，2020：91-94.

[27] Zhang J Q，Chen Z M，Ma M Z，et al. High-performance elliptic curve scalar multiplication architecture based on interleaved mechanism[J]. IEEE Transactions on Very Large Scale Integration（VLSI）Systems，2025，33（3）：757-770.

[28] Zhang J Q，Chen Z M，Ma M Z，et al. High-performance ECC scalar multiplication architecture based on comb method and low-latency window recoding algorithm[J]. IEEE Transactions on Very Large Scale Integration（VLSI）Systems，2024，32（2）：382-395.

[29] Nadikuda P K G，Boppana L. Low area-time complexity point multiplication architecture for ECC over GF（2^m） using polynomial basis[J]. Journal of Cryptographic Engineering，2023，13（1）：107-123.