面向SM2和SM3算法的RISC-V密码协处理器设计

李秀滢，鄂佳言，武秀云，杨亚涛，李兆斌
（北京电子科技学院电子与通信工程系，北京 100070）

摘要：针对资源受限的嵌入式平台下国密数字签名算法运算开销大、执行效率低的问题，设计并实现了一种面向数字签名中SM2、SM3算法软硬协同加速处理的RISC-V密码协处理器。该处理器采用自定义指令扩展机制，将SM2标量乘中的模乘、模加减等有限域关键运算以及SM3扩展压缩过程映射为协处理器指令，并设计了模加减、专用素数域模乘、通用Montgomery模乘和SM3扩展压缩等硬件单元。实验结果表明，标量乘时间最低为1.14 ms，AT值为2.69；SM3扩展压缩单元完成单个消息分组处理仅需64个时钟周期，吞吐率达到1 240 Mbit/s；完整数字签名和验签时间分别为2.92 ms和3.40 ms，对应AT值分别为11.77和13.70。所提方案提高了SM2标量乘、SM3扩展压缩以及数字签名验签过程的执行效率，可为资源受限场景下国密数字签名算法的工程实现提供参考。

关键词：RISC-V；数字签名；SM2算法；SM3算法；协处理器
中图分类号：TP332；TP309
文献标志码：A
doi：10.11959/j.issn.1000–436x.TXXB260187

Design of RISC-V cryptographic coprocessor for the SM2 and SM3 algorithms

Li Xiuying, E Jiayan, Wu Xiuyun, Yang Yatao, Li Zhaobin
Department of Electronic and Communication Engineering, Beijing Electronic Science and Technology Institute, Beijing 100070, China

Abstract: To reduce the computational cost of digital signature algorithms on resource-constrained embedded platforms, a RISC-V cryptographic coprocessor was designed and implemented. A hardware-software co-design method was developed. The key finite-field operations in SM2 scalar multiplication and the SM3 expansion-compression process were mapped to custom coprocessor instructions. Several hardware units were designed, including a modular add/sub unit, a dedicated prime-field modular multiplication unit, a general Montgomery modular multiplication unit, and an SM3 expansion-compression unit. Experimental results showed that a minimum scalar multiplication time achieves 1.14 ms , with an AT value of 2.69. A single message block was processed by the SM3 expansion-compression unit in only 64 clock cycles, and a throughput of 1240 Mbit/s was achieved. Complete digital signature generation and verification were completed in 2.92 ms and 3.40 ms, respectively, with corresponding AT values of 11.77 and 13.70. The proposed scheme improves the execution efficiency of SM2 scalar multiplication, SM3 expansion-compression, and digital signature generation and verification, and provides a practical reference for the engineering implementation of Chinese national cryptographic digital signature algorithms in resource-constrained scenarios.

Key words: RISC-V, digital signature, SM2 algorithm, SM3 algorithm, coprocessor

收稿日期：2026–04–08；**修回日期：**2026–06–27
通信作者：杨亚涛，yy2008@163.com
基金项目：国家自然科学基金资助项目（No.62476013）；北京自然科学基金资助项目（No.L251067）；国家档案局科技基金资助项目（No.2025-Z-009）；中央高校基本科研业务费专项资金资助项目（No.3282025045, No.3282025039）
Foundation Items: The National Natural Science Foundation of China (No.62476013), The Natural Science Foundation of Beijing (No.L251067), Science and Technology Project of the National Archives Administration of China (No.2025-Z-009), The Fundamental Research Funds for Central Universities (No.3282025045, No.3282025039)

0 引言

在普适计算时代，数字签名技术提供的身份认证、完整性校验和抗抵赖性，成为保护个人隐私与企业机密的重要手段，广泛应用于网上交易、智能终端接入、车联网通信和工业控制等场景。在我国商用密码体系中，SM2和SM3算法是构成国密数字签名的核心算法，其现有实现方式包括：纯软件优化、专用硬件加速和软硬件协同设计。纯软件实现灵活性较高，但在资源受限平台上执行效率较低；专用硬件实现性能较好，但通用性、灵活性和扩展性不足。相比之下，软硬件协同设计能够在灵活性、性能和资源开销之间取得较好的平衡，适合轻量级和可重构的密码计算加速场景。

RISC-V指令集架构的开放性和可扩展性，为密码算法软硬件结合的定制化加速提供了优选的体系基础。文献[1]的NICE（nuclei instruction co-unit extension）通信接口和Rocket体系[2]的RoCC通信接口均支持自定义协处理器扩展，便于将专用密码协处理器接入主处理器系统中。现有RISC-V体系中扩展的密码算法专用指令集[3]主要面向简单密码操作，缺乏针对SM2素数域运算的专用指令，且针对SM3扩展的指令粒度较细，导致对数字签名方案的整体加速效果有限。

针对上述问题，本文面向SM2和SM3算法设计了一种RISC-V密码协处理器，用于在较低资源开销下提升国密数字签名相关运算的执行效率。本文的主要贡献如下。

（1）设计了一种基于RISC-V软硬件协同的SM2和SM3加速架构。该架构将模乘、模加减和SM3扩展压缩等高频耗时操作映射为硬件算子，将协议控制、标量乘、群运算和模逆等流程保留在软件层执行，从而避免全硬件实现带来的资源开销。

（2）构建了基于RISC-V的密码协处理器。面向SM2素数域运算，设计了模加减、专用快速模乘和通用Montgomery模乘指令硬件单元；结合256 bit大数寄存器组，设计了数据加载、写回、搬运和交换指令。面向消息扩展与压缩过程，设计了SM3扩展压缩指令硬件单元，以提高SM3计算效率。

（3）在Xilinx Artix-7 FPGA平台上完成协处理器的实现与验证。实验结果表明，基于本文协处理器实现的Montgomery Ladder标量乘和滑动窗口标量乘时间分别为2.04 ms和1.14 ms，资源面积与花费时间的乘积值（area-time product, AT）[4]分别为4.81和2.69；SM3扩展压缩单元处理单个消息分组仅需64个时钟周期，吞吐率达到1 240 Mbit/s；完整数字签名和验签AT值分别为11.77和13.70，体现出较好的面积性能综合效率。

1 相关研究

针对SM2和SM3算法高效实现的方法主要包括纯软件优化、纯硬件加速和软硬件协同优化。

在软件优化方面，乔晗等[5]面向GmSSL密码库，采用快速模约减、NEON并行点运算和NAF标量乘等方法，提高了SM2签名与标量乘的性能。杨先伟等[6]则针对SM3压缩函数和消息扩展过程进行软件优化。此类方法灵活性高，但受限于通用处理器的指令宽度、寄存器资源和访存开销，在资源受限平台上难以实现高性能。

在硬件加速方面，现有研究多通过优化硬件实现的模运算、点运算和杂凑压缩计算提升核心操作效率。张盛仕等[7]面向SM2软硬件协同系统，优化了素数域标量乘与模运算硬件结构。Kong等[8]采用Karatsuba-Ofman乘法、快速模约减和wNAF方法设计了高性能SM2硬件算法模块。Zheng等[9]围绕SM3压缩迭代过程设计了高效低功耗硬件模块。此类方案计算性能较好，但硬件结构通常与特定算法流程绑定，资源开销高，通用性和复用性有限。

近年来，基于自定义扩展指令的软硬协同加速方案，逐渐成为轻量级密码实现和算法可重构的研究热点。王腾飞等[10]设计了SM2专用指令协处理器，通过专用指令和流水线结构提升标量乘性能。王明登等[11]基于E203处理器实现了SM2、SM3和SM4协同加速，验证了RISC-V自定义指令在国密算法中的可行性。研究表明，自定义指令能够缩短密码运算执行路径，并在性能与资源开销间取得平衡。但现有方案多围绕特定算法流程展开，对不同标量乘算法和上层软件调用的适配能力仍有不足。

考虑上述问题，本文面向资源受限场景下的数字签名应用需求，针对SM2和SM3进行软硬件协同加速，设计可复用的底层硬件算子，构造专用密码协处理器。不同的标量乘算法（如Montgomery Ladder、滑动窗口，以及其他由点加和倍点构成的标量乘算法）均可在软件层通过调用同一套底层扩展密码指令实现，从而提高协处理器的通用性和可扩展性，提升密码算法的可重构能力。

2　关键技术基础

2.1　RISC-V自定义指令

RISC-V在指令编码中预留了用户自定义区域，便于在保持基础指令体系结构的同时，扩展面向特定算法的功能指令。对于32位RISC-V指令，规定指令[6:0]位为操作码字段，当该字段为0x0B、0x2B、0x5B和0x7B编码时，代表用户自定义指令[12]。RISC-V的指令编码格式如图1所示。

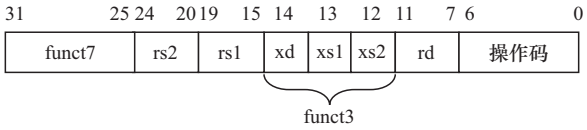


图1　RISC-V的指令编码格式

图1中，rd、rs1、rs2字段分别表示目的寄存器、源操作数1、源操作数2，funct7和funct3字段用于区分不同类型的指令。本文主处理器和协处理器交互接口为蜂鸟E203处理器的NICE接口。当主处理器译码到自定义指令后，通过NICE接口将操作请求发送给协处理器处理。协处理器指令对funct3字段进行了重新定义，拆分为xs1位、xs2位和xd位，用于指示指令是否需要读取源操作数rs1和rs2，以及是否需要写回目的寄存器rd。

2.2　处理器与协处理器交互机制

与将硬件加速模块作为独立外设挂接于系统总线上的方式不同，协处理器能够直接参与主处理器指令流中的特定操作。该方式通常不需要像外设调用那样额外经过寄存器配置、总线事务发起和轮询等待等控制过程，适合SM2、SM3这类包含大量高频基础运算的密码算法加速场景。主处理器与协处理器交互示意如图2所示。

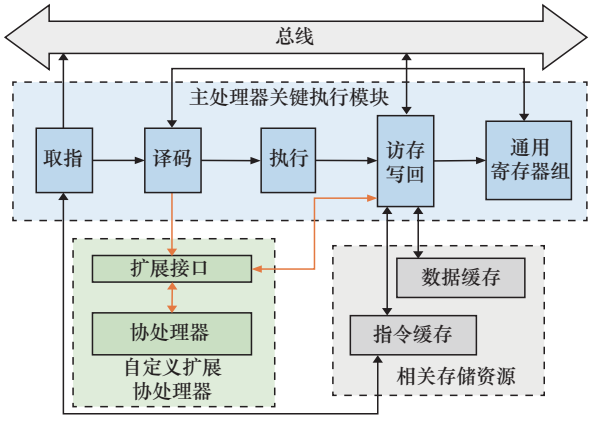


图2　主处理器与协处理器交互示意

执行指令时，主处理器完成取指和译码，并判断当前指令由主处理器执行还是由协处理器执行。通用指令由主处理器执行部件完成，自定义扩展指令则通过接口发送给协处理器，由协处理器启动相应硬件单元完成计算，并将结果写回通用寄存器或目标存储单元。

2.3　数字签名算法

国密数字签名算法核心为SM2和SM3算法。

2.3.1　SM2算法解析

SM2建立在有限域椭圆曲线运算基础之上。SM2支持素数域F_p和二元扩域F_{2^m}两类椭圆曲线，其中素数域方案具有结构清晰、应用广泛且便于与大整数模运算硬件结合等特点，本文选取素数域椭圆曲线进行实现。SM2算法运算层次如图3所示，将SM2算法拆解为3层运算结构，最上层为标量乘，中间层为椭圆曲线群运算（含点加和倍点），最下层为有限域基本运算（含模加、模减、模乘和模逆）。

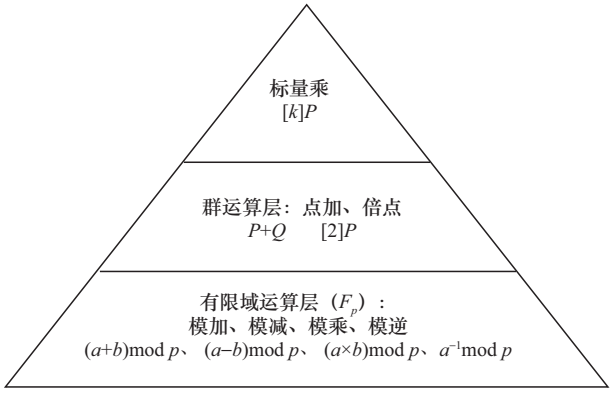


图3　SM2算法运算层次

2.3.2　SM3算法解析

国密的消息摘要算法SM3，用于完整性保护、数字签名和各类密码协议中。SM3算法流程如图4所示，主要由消息填充、消息扩展和压缩迭代3个部分构成。

SM3算法先对消息M填充处理，并划分为若干512位分组，每分组拆为16个初始消息字，再扩展消息字供压缩迭代函数使用。设初始向量为$V^{(0)}$，压缩迭代函数以当前链接变量$V^{(i)}$和该消息分组为输入，生成下一轮链接变量$V^{(i+1)}$，全部消息分组经过以上迭代处理后，最终得到输出杂凑值。

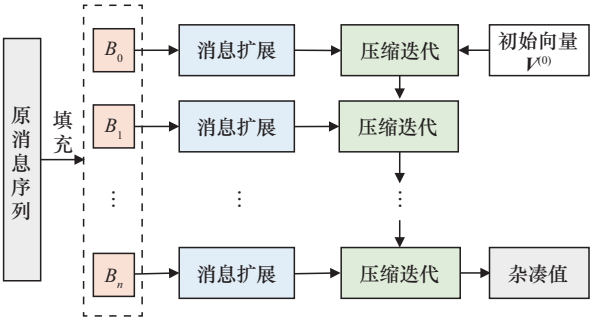


图4　SM3 算法流程

2.3.3　数字签名算法解析

数字签名算法主要包括签名和验签两个过程，国密数字签名基本流程如图 5 所示。由图 5 可以看出，SM3 用于生成用户杂凑值和消息摘要，SM2 则基于摘要结果完成签名生成或签名验证。

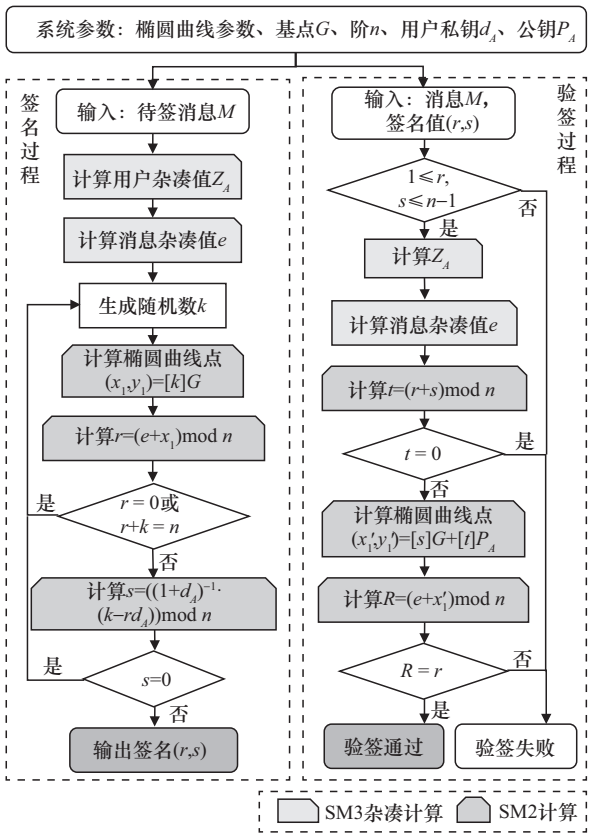


图5　国密数字签名基本流程

签名阶段主要包括消息摘要计算、一次基点标量乘 $[k]G$、若干次模加减、模乘以及一次模逆运算；验签阶段不再涉及私钥相关模逆，但需完成消息杂凑，以及两个标量乘 $[s]G$、$[t]P_A$ 和一次点加运算。其中时间开销主要集中在标量乘、群运算和底层模运算，SM3 摘要计算也是重要耗时部分。

3　算法实现的优化方法

3.1　软硬件实现结构划分

本文针对 SM2 和 SM3 算法设计的软硬件实现单元结构划分如图 6 所示。硬件层主要实现调用频繁、计算开销大的模加、模减和模乘，以及消息扩展与压缩迭代过程。软件层负责模逆、点加、倍点、标量乘运算以及消息填充，通过自定义指令调用硬件单元完成关键计算。

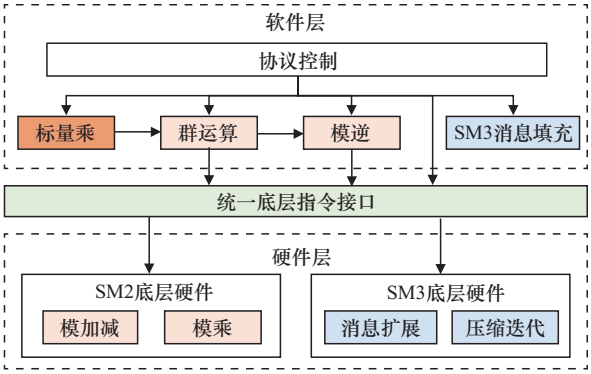


图6　软硬件实现单元结构划分

3.2　SM2 算法实现的优化设计

3.2.1　有限域运算层实现的优化方法

模加和模减运算单元极简，算法上不需做额外优化，直接由硬件实现；模乘是性能瓶颈，本文设计了快速模乘算子用于高频 SM2 标准素数域运算，同时设计通用 Montgomery 模乘算子以支持不同算法需求，两种模乘算子都由硬件实现；模逆算子调用频率低，基于快速模乘算子软件实现，在保证性能的同时减少硬件资源开销。

（1）快速模乘算子优化

模乘是 SM2 素数域的核心运算，其性能直接影响点加、倍点和标量乘等上层计算。对于 256 bit 大整数模乘，先整数相乘再模除会带来高硬件开销和计算时延，因此本文基于优化的大整数乘法和模约减运算，实现了快速模乘算子。

① 大整数乘运算优化策略

Karatsuba 算法[13]与 Comba 算法[14]是两类典型的大整数乘法实现方法。文献[15]的实现结果表明，前者通过分解操作数减少乘法次数，控制逻辑较复杂；后者部分积累加路径较长。本文结合两者优点设计了针对 256 bit 大整数的行级并行分段乘法，单个时钟周期内，将乘数分段输入并行乘法器阵列，与被乘数所有分段同时计算，从而降低迭代次数。

② 模约减运算优化策略

Montgomery[16]、Barrett[17]和 Solinas[18]是较常见的三类模约减方法。Montgomery模约减适用于一般奇模数，支持连续模乘，但需域转换；Barrett通过预计算常数近似求商避免除法，但增加了存储和额外乘法开销；Solinas针对伪梅森模数，将高位折叠为移位与加减组合，实现低开销快速约减。SM2素数域模数具有典型的伪梅森稀疏特性，由模运算等价关系得到：

$$2^{256} = 2^{224} + 2^{96} - 2^{64} + 1 (\mathrm{mod}\, P) \tag{1}$$

对于超过256 bit的中间结果，可将高于256 bit的部分按照式(1)折叠到低位，用有限次移位与加减运算替代一般模除过程。基于此，本文采用Solinas快速约减方法进一步提高模乘运算的执行效率。

③ 快速模乘算子设计

在①和②优化策略的基础上，本文进一步采用并行交织型模乘架构，实现大整数乘法与快速模约减运算交替融合计算。在每轮分段乘积生成后，当前部分积被及时并入累加结果，并对高位溢出部分进行折叠约减，从而限制中间数据宽度，减少寄存器和缓冲资源开销。并行交织模乘算子优化实现方法如算法1所示。

算法1　面向SM2素域的并行交织模乘算法

输入　$A,B \in [0,P-1], d, s = \left\lceil \frac{256}{d} \right\rceil$

输出　$T = A \cdot B \bmod P$

1) $T \leftarrow 0$

2) for j from 0 to $s-1$ do

3)　for all $i = 0,1,\cdots,s-1$ do

4)　　$p_i \leftarrow a_i b_j$

5)　end for

6)　$V_{\mathrm{prod}} \leftarrow \sum_{i=0}^{s-1} 2^{di} p_i$

7)　$T \leftarrow T \ll d + V_{\mathrm{prod}}$

8)　$H \leftarrow T \gg 256$; $L \leftarrow T \bmod 2^{256}$

9)　$T \leftarrow L + H + 2^{96}H + 2^{224}H - 2^{64}H$

10) end for

11) if $T \geqslant P$ then

12)　$T \leftarrow T - P$

13) end if

算法1中，$A = \sum_{i=0}^{s-1} a_i 2^{di}$，并按扫描顺序表示 $B = \sum_{j=0}^{s-1} \tilde{b}_j 2^{d(s-1-j)}$，其中 $\tilde{b}_0$ 为最高段。

（2）Montgomery模乘算子优化

为支持SM2数字签名过程中可能涉及的一般奇模数模乘，本文设计了通用Montgomery模乘算子。与专用P模乘算子不同，该算子不依赖SM2标准素数的特殊形式，具有更好的通用性。

Montgomery模乘通过引入预计算参数 $N' = -N^{-1} (\mathrm{mod}\, R)$，可将中间结果与模数 N 的特定倍数相结合，使约减过程转化为移位与加法操作。本文将Montgomery约减拆解并交替分配到乘法迭代过程中，使部分积累加与约减同步进行，从而形成面向通用奇模数的交织型Montgomery模乘实现结构。具体算子优化实现方法如算法2所示。

算法2　基于字长级扫描的交织Montgomery模乘算法

输入　$N(k\text{位}), A, B \in [0,N-1], d, s = \left\lceil \frac{k}{d} \right\rceil$

输出　$T = A \cdot B \cdot R^{-1} \bmod N$，其中 $R = 2^{sd}$

1) $T \leftarrow 0$

2) for i from 0 to $s-1$ do

3)　$q \leftarrow (T[0] + A[i] \cdot B[0]) N' (\mathrm{mod}\, 2^d)$

4)　$T \leftarrow T + A[i] \cdot B + qN$

5)　$T \leftarrow T \gg d$

6) end for

7) if $T \geqslant N$ then

8)　$T \leftarrow T - N$

9) end if

算法2中，$A = \sum_{i=0}^{s-1} A[i] 2^{di}$，$B = \sum_{j=0}^{s-1} B[j] 2^{dj}$。

（3）模逆算子优化

模逆在各类模运算中开销最大，其实现对硬件结构影响显著。现有方法包括二进制扩展欧几里得算法[19]和基于费马小定理的算法[20]。前者效率高但需复杂逻辑，硬件开销大；后者将模逆转为模乘序列，可复用现有模乘算子，无需额外硬件。本文采用基于费马小定理实现模逆算子，其中模逆计算中用到的模乘算子为本文优化实现的。

3.2.2　群运算层实现的优化方法

在椭圆曲线密码体制中，点加和倍点这两种群运算效率与底层坐标系的选取密切相关。本文采用Jacobian坐标系描述椭圆曲线点，群运算过程中可

将求逆操作推迟到最终结果转换阶段，进而将大部分计算转化为模乘、模平方、模加和模减，提高整体运算效率。设仿射坐标点$P=(x,y)$在Jacobian坐标系表示为$P=(X:Y:Z)$，则二者满足：

$$\begin{cases} x=\dfrac{X}{Z^2} \\ y=\dfrac{Y}{Z^3} \end{cases} \tag{2}$$

其中，$Z\neq 0$时表示普通曲线点，$Z=0$表示无穷远点。

设椭圆曲线满足方程$y^2=x^3+ax+b$，对点$P=(X_1:Y_1:Z_1)$进行倍点运算，记结果为$2P=(X_3:Y_3:Z_3)$。由仿射坐标变换到Jacobian坐标系后的倍点计算表示为：

$$\begin{cases} A=X_1^2,B=Y_1^2,C=B^2 \\ D=2[(X_1+B)^2-A-C] \\ E=3A+aZ_1^4,F=E^2 \\ X_3=F-2D \\ Y_3=E(D-X_3)-8C \\ Z_3=2Y_1Z_1 \end{cases} \tag{3}$$

对于点加运算，设$P_1=(X_1:Y_1:Z_1)$，$P_2=(X_2:Y_2:Z_2)$，结果为$P_3=(X_3:Y_3:Z_3)=P_1+P_2$，则变换的点加计算为：

$$\begin{cases} U_1=X_1Z_2^2,U_2=X_2Z_1^2 \\ S_1=Y_1Z_2^3,S_2=Y_2Z_1^3 \\ H=U_2-U_1,R=S_2-S_1 \\ X_3=R^2-H^3-2U_1H^2 \\ Y_3=R(U_1H^2-X_3)-S_1H^3 \\ Z_3=HZ_1Z_2 \end{cases} \tag{4}$$

3.2.3 标量乘运算层实现的优化方法

本文设计的协处理器并非面向单一算法流程，而是希望在提升运算效率基础上，兼顾算法可重构性，因此优化实现了两种标量乘算法。

（1）滑动窗口算法

通过预计算和窗口重编码减少点加次数，发挥底层模乘、模加和模减指令的运算性能，显著降低标量乘的总计算开销。Jacobian坐标下的滑动窗口算法实现如算法3所示。

算法3 Jacobian坐标下的滑动窗口算法

输入 $k=(k_{l-1}...k_1k_0)_2,P=(x_p,y_p),w$

输出 $Q=k\cdot P=(x_Q,y_Q)$

1) 预计算奇数倍点表$T[1],\cdots,T[2^w-1]$
2) $Q\leftarrow 0,i\leftarrow l-1$
3) while $i\geqslant 0$ do
4) if $k_i=0$ then
5) $Q\leftarrow$ PointDouble(Q)
6) $i\leftarrow i-1$
7) else
8) 选取以k_i结尾的最长非零窗口L
9) 计算L次PointDouble(Q)
10) $Q\leftarrow$ PointAdd$(Q,T[L]),i\leftarrow i-L$
11) end if
12) end while
13) $Q\leftarrow$Jacobian坐标转仿射坐标

（2）Montgomery Ladder算法

该算法采用规则迭代结构，控制流程规整，具有一定的抗简单功耗分析（simple power analysis, SPA）能力[21]，适用于对实现安全性和操作规律性有要求的场景。Jacobian坐标下的Montgomery Ladder算法实现如算法4所示。

算法4 Jacobian坐标下的Montgomery Ladder算法

输入 $k=(k_{t-1}\cdots k_1k_0)_2,P=(x_P,y_P)$

输出 $Q=k\cdot P=(x_Q,y_Q)$

1) $R_0\leftarrow 0,R_1\leftarrow P,b_{\text{prev}}\leftarrow 0$
2) for i from 255 down to 0 do
3) $b_{\text{cur}}\leftarrow k_i$
4) swap_signal $\leftarrow b_{\text{cur}}\oplus b_{\text{prev}}$
5) 根据swap_signal交换R_0,R_1
6) $R_1\leftarrow$ PointAdd(R_0,R_1)
7) $R_0\leftarrow$ PointDouble(R_0)
8) $b_{\text{prev}}\leftarrow b_{\text{cur}}$
9) end for
10) 根据b_{prev}交换R_0,R_1
11) $Q\leftarrow$Jacobian坐标转仿射坐标

3.3 SM3算法实现的优化设计

SM3算法处理长度小于2^{64} bit的消息M，输出256 bit摘要，其运算流程包括消息填充、消息扩展和压缩迭代。消息扩展和压缩迭代需要针对每个消息分组重复执行，是SM3算法的主要计算开销来源。本文将消息扩展与压缩迭代过程进行并发执行结构设计，将寄存器依赖从68个字大幅缩减至16个字。具体实现方法如算法5所示。

算法5　SM3扩展压缩实现算法

输入　$B_i = W_0,W_1,\cdots,W_{15}$，$V^{(i)} = (A,B,C,D,E,F,G,H)$

输出　$V^{(i+1)}$

1) $(A,B,C,D,E,F,G,H) \leftarrow V^{(i)}$

2) 将$W_0,\cdots,W_{15}$写入窗口寄存器$W[0\cdots15]$

3) for j from 0 to 63 do

4)　依据当前窗口寄存器生成W_j和W_j'

5)　更新A,B,C,D,E,F,G,H

6)　if $j < 52$ then

7)　　生成扩展消息字，更新窗口寄存器

8)　end if

9) end for

10) $V^{(i+1)} \leftarrow V^{(i)} \oplus (A,B,C,D,E,F,G,H)$

4　RISC-V密码协处理器设计

4.1　RISC-V自定义指令设计

指令作为指挥硬件工作的接口，是算法软硬件运算单元协同的关键。本文结合SM2、SM3算法的实际运算设计了4条SM2相关运算指令、1条SM3专用指令以及4条数据传送指令。所有自定义指令均采用RISC-V架构中的custom3编码空间，对应操作码为0x7B；指令格式为标准*R*类型。本文设计的自定义指令如表1所示。

ld_cp与st_cp指令用于完成内存与协处理器之间的数据交换，实现32 bit数据与256 bit大数之间的装载和回写。mov_cp和swap_cp指令用于协处理器内部寄存器之间的数据搬运和条件交换，以减少大数实际移动带来的开销。

SM2相关指令包括专用*P*模乘指令sm2_p_mmul、Montgomery模乘指令sm2_mgm_mmul、模加指令sm2_madd和模减指令sm2_msub。点加、倍点和标量乘等高层运算仍由软件调度完成，并通过调用上述底层模运算指令实现加速。SM3专用指令sm3_ce用于完成消息扩展与压缩迭代，其结果写回协处理器内部寄存器堆。

4.2　协处理器设计

4.2.1　协处理器硬件结构设计

基于图2的RISC-V处理器结构，本文设计了面向国密数字签名的密码加速协处理器结构，如图7所示。

主处理器发出的自定义指令经请求通道进入协处理器后，由译码单元解析操作类型及源、目的寄存器信息，并结合资源占用和数据相关状态判断指令是否可以发射。为减少256 bit大数搬移开销，协处理器内部维护逻辑寄存器到物理寄存器的映射表。指令访问寄存器时先完成映射转换，再访问对应物理寄存器；数据搬运和条件交换操作只更新映射关系，无需实际移动数据。状态控制单元负责模块忙闲判断、指令发射和结果返回。

协处理器设置32×256 bit、1写2读寄存器堆，用于存放SM2域元素、点运算中间变量及SM3中间结果，并通过寄存器访问控制模块协调读出、写回和访存回写。由于外部存储通路为32 bit字宽，而内部运算以256 bit数据块为粒度，数据加载写回单元负责完成32 bit数据与256 bit大数之间的拆分和拼接。

运算执行单元是协处理器的核心。专用模乘单元面向SM2素数域模数特点设计，用于支持点加、

表1　自定义指令

指令助记符	funct7	rs2	rs1	rd	功能描述
ld_cp	0000101	传输字数	内存基址	目的索引	内存数据批量加载至内部大数寄存器
st_cp	0000110	传输字数	内存基址	目的索引	内部大数寄存器数据批量写回内存
mov_cp	0001001	—	源操作数索引	目的索引	在内部大数寄存器之间执行数据搬运
swap_cp	000011x	寄存器索引2	—	寄存器索引1	交换两个内部大数寄存器的映射关系
sm2_p_mmul	0000000	乘数*B*索引	乘数*A*索引	结果索引	执行一次SM2素数域模*P*乘法
sm2_mgm_mmul	0000001	乘数*B*索引	乘数*A*索引	结果索引	执行一次Montgomery模乘运算
sm2_madd	0000010	加数索引	被加数索引	结果索引	执行一次模加运算
sm2_msub	0000011	减数索引	被减数索引	结果索引	执行一次模减运算
sm3_ce	0000100	消息块索引	IV状态	更新索引	执行一次SM3扩展压缩运算

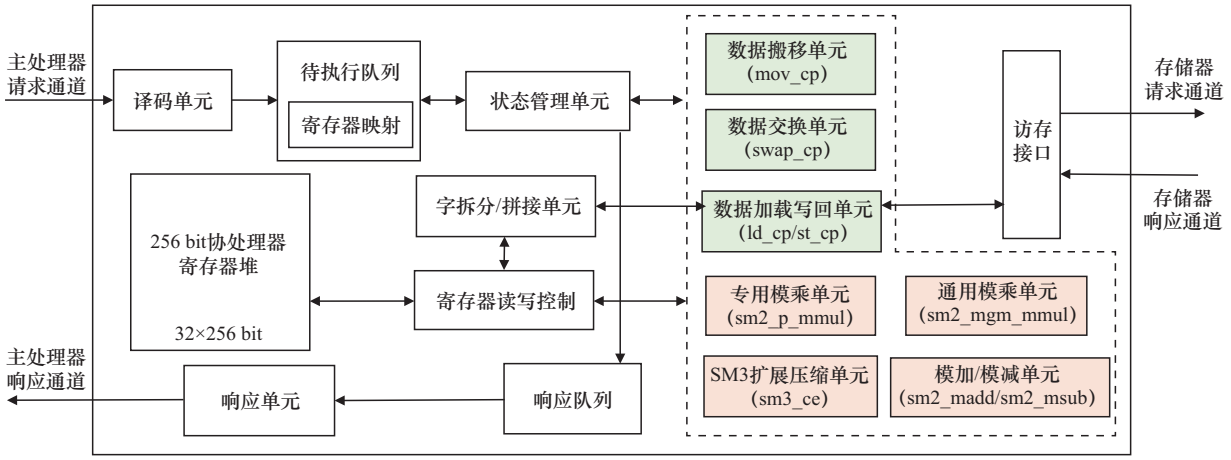


图 7　面向国密数字签名的密码加速协处理器结构

倍点中的素数域乘法；通用模乘单元用于上层协议的通用模乘计算；模加/模减单元用于有限域基本加减运算；SM3 扩展压缩单元实现消息扩展和压缩计算，支撑签名验签过程中完整的杂凑运算。

4.2.2　SM2 专用指令硬件实现

（1）模加/模减单元

本文将模加与模减运算集成在同一硬件单元中，对应 sm2_madd 和 sm2_msub 指令。此设计是为节省硬件面积，减少模加与模减分别实现时重复数据通路带来的资源开销。该单元通过统一的控制逻辑支持不同模数下的运算，并采用固定运算流程，每次运算时间恒定，具备一定的 SPA 抵抗能力。

（2）模乘单元

本文对底层乘法数据通路位宽进行了比较。位宽较小时，硬件开销较低，但迭代轮次增加，运算速度下降；位宽较大时，虽然可减少迭代次数，但不利于主频提升。通过实验分析，3.2.1 节中阐述的算法 1 和算法 2 在位宽 $d = 64$ 时，硬件开销和计算速度之间取得较好平衡。

① 专用模乘单元

针对 SM2 标准素数 P，本文设计了专用模乘硬件单元，对应 sm2_p_mmul 指令，核心硬件实现结构如图 8 所示。

该模乘单元采用 4 路并行的 64×64 bit 乘法单元。每个时钟周期内，4 个乘法单元分别计算乘数 A 的 4 个 64 bit 分段与当前 B 分段 B_i 的乘积，并结合上一轮累加结果完成迭代更新。针对中间结果超过 256 bit 的问题，单元利用 SM2 素数模数的特殊形式进行快速约减，将高位溢出项折叠到低位，并通过移位、加减和条件减法完成模约减。考虑到折叠后仍可能存在残余溢出，中间迭代阶段仅进行一次模减校验，在最后修正阶段统一约减。

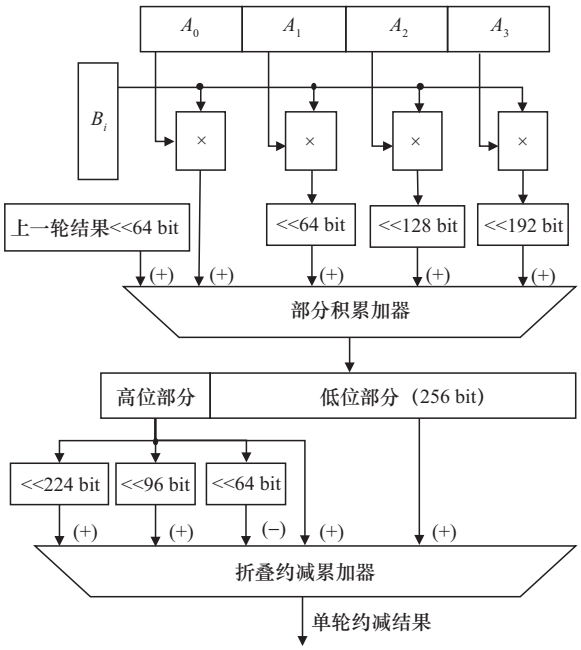


图 8　SM2 标准素数 P 专用模乘核心硬件实现结构

② 通用 Montgomery 模乘单元

基于算法 2，本文设计了 256 bit 的 Montgomery 模乘硬件单元，对应 sm2_mgm_mmul 指令。考虑到数字签名中通用模乘相对专用模乘调用频率低很多，在设计中更侧重资源开销控制，因此仅使用一个 64×64 bit 的乘法器来实现通用模乘单元。输入操作数 A 和 B 均按 64 bit 拆分为 4 个字，模数 N 也按同样方式分解为 n_0,n_1,n_2,n_3，通过 4 轮迭代完成乘加、约减以及移位操作。单轮 Montgomery 乘加及约减时序如图 9 所示。

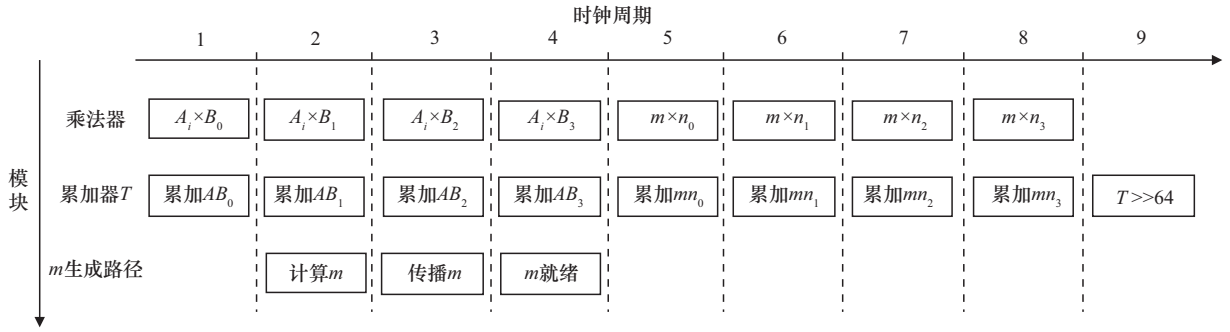


图9　单轮Montgomery乘加及约减时序

图9的每轮迭代中，单元先完成部分积累加，再根据最低字计算约减因子，并将约减过程与乘法迭代交织执行。由于Montgomery约减能够将除法转化为移位和加法操作，该结构避免了一般模除带来的高硬件开销。经过全部迭代后，单元执行一次条件减法，得到最终模乘结果。

4.2.3　SM3专用指令硬件实现

针对SM3算法，本文设计了基于消息扩展与压缩迭代协同执行的硬件单元，对应sm3_ce指令，其硬件实现结构如图10所示。

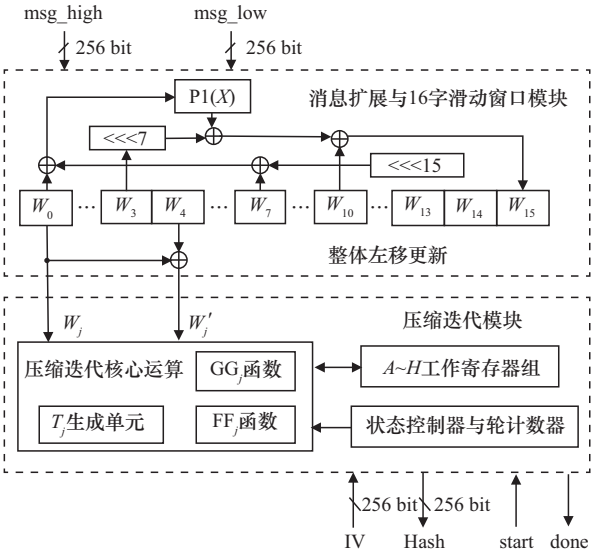


图10　SM3扩展压缩单元硬件实现结构

框架执行单元内部仅保留一个深度为16的移位寄存器阵列W。每个时钟周期内，消息扩展模块根据当前窗口数据生成新消息字，并且寄存器组整体移位，将新消息字移入窗口末端。消息扩展模块并发输出W_j和W_j'给压缩迭代模块，进而实现单周期内同时完成一次消息扩展与压缩迭代。

为降低硬件开销，T_j采用寄存器滚动方式生成，避免额外查表逻辑。针对单轮压缩中多操作数累加路径较长的问题，本文采用并行加法树优化关键路径，例如将TT1的计算FF+D+SS2+W_j'优化成(FF + D)和(SS2 + W_j)，再由末级加法器对两路中间结果进行汇总得到TT1；同时对底层布尔函数进行等价化简，FF函数由原始定义的$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$等价变换为$(X \wedge Y) \vee (Z \wedge (X \vee Y))$；GG函数重写为$Z \oplus (X \wedge (Y \oplus Z))$。

5　实验验证与分析

5.1　实验环境配置

本文基于Nuclei DDR200T开发板完成实验验证，其核心芯片为Xilinx Artix-7（xc7a200tfbg484-2）FPGA。软件开发环境基于HBird SDK搭建；PC端配置Nuclei RISC-V嵌入式工具链，用于C语言测试程序的编译。硬件实现方面，基于E203处理器软核，采用Verilog HDL语言实现本文的密码加速协处理器，并通过Vivado完成综合实现与比特流文件生成，随后将其下载至开发板。在运行阶段，借助Nuclei OpenOCD通过make upload命令完成测试程序烧录，经串口输出测试结果，国密数字签名算法板上验证结果如图11所示。

5.2　实验比较分析

5.2.1　协处理器各指令硬件单元测试

本文在Vivado平台下完成综合实现，测得协处理器最大工作频率为68 MHz。协处理器各单元实现情况如表2所示。

由表2可以看出，协处理器的硬件开销主要集中在模乘相关单元。专用P模乘单元占用3 502 LUT和64 DSP，Montgomery模乘单元占用2 417 LUT和26 DSP，是资源消耗的主要部分。模加减单元和SM3扩展压缩单元均未使用DSP，整体资源开销较低。

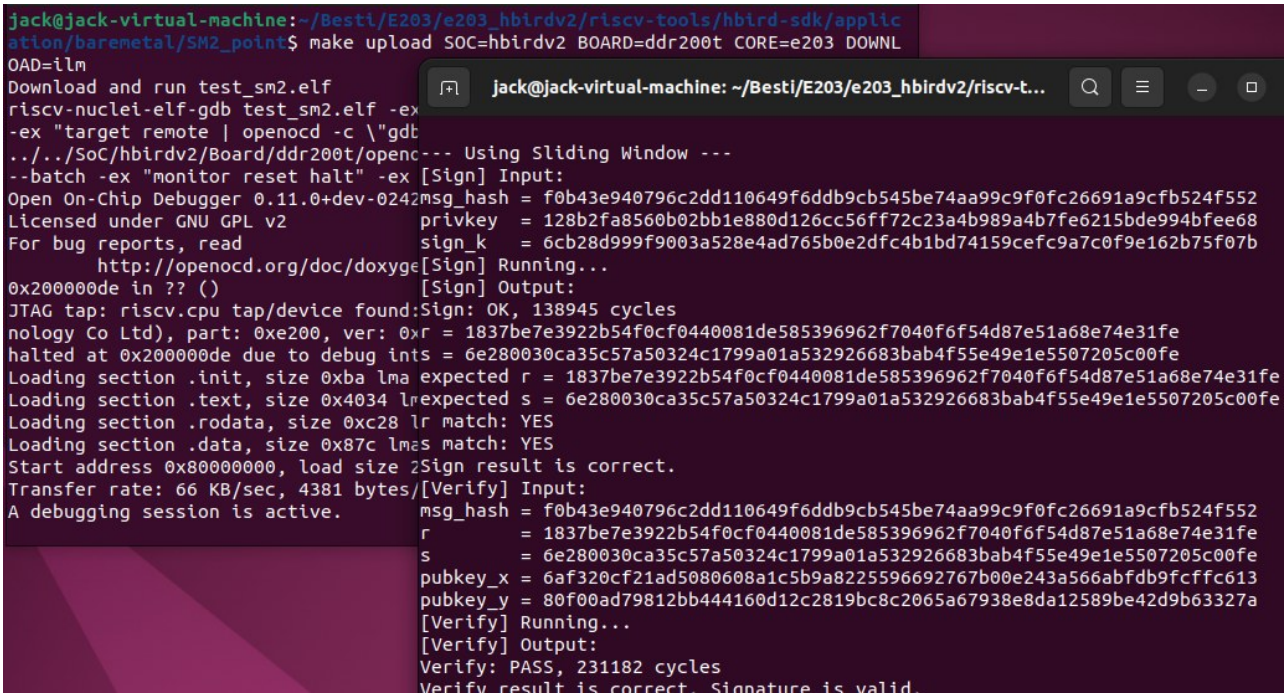


图11 国密数字签名算法板上验证结果

表2 协处理器各单元实现情况

硬件单元	LUT	DSP	FF	周期/cycle
模加减	2 705	0	257	1
专用P模乘	3 502	64	1 873	14
Montgomery模乘	2 417	26	3 275	58
SM3扩展压缩	1 323	0	1 360	64
其他	473	0	2 921	—

5.2.2 标量乘实现方案比较

为全面评价本文协处理器在运算速度与资源开销之间的综合表现，引入AT作为评估指标。AT定义为资源面积与执行时间的乘积，本文将slice的占用量计为资源面积的计算值，执行时间以ms为单位。AT值越小，说明在给定硬件面积下能够获得更高的计算效率，更适合资源受限场景。

本文基于算法4（Montgomery Ladder）实现的方案1，能够抵抗SPA分析；基于算法3（$w=4$滑动窗口）实现的方案2，运算性能高。本文方案与其他方案标量乘对比如表3所示。由表3可以看出，本文方案2的AT值仅为2.69，在所有方案中综合效率最优；本文方案1的AT值为4.81，同样优于多数其他方案。虽然文献[24]和文献[25]在执行时间上更短，但其面积开销相对较大，AT值仍高于本文方案2。实验结果表明本文的协处理器在资源受限条件下能够较好提升标量乘性能。

表3 本文方案与其他方案标量乘对比

文献	器件	面积/kslice	时钟频率/MHz	时间/ms	AT
本文方案1	Artix-7	2.36	68	2.04	4.81
本文方案2	Artix-7	2.36	68	1.14	2.69
文献[22]	Virtex-4	20.79	60	6.9	143.45
文献[23]	Virtex-5	10.2	67	6.63	67.63
文献[24]	Virtex-4	22.24	21	0.68	15.12
文献[10]	Zynq-7	7.15	110	2.25	16.09
文献[25]	Zynq Ultra-Scale+ MPSoC	8.38	100	0.72	6.09

5.2.3 SM3算法实现方案比较

为比较不同SM3算法实现方案的性能，引入吞吐率作为评价指标，计算式为：

$$\text{吞吐率}=\frac{f\times L}{N} \tag{5}$$

其中，f表示电路工作频率，L表示单次处理的数据位宽，N表示完成一次运算所需的时钟周期数。

（1）与RISC-V扩展*K*指令集对比

针对SM2和SM3算法，RISC-V的*K*指令集（标准密码扩展指令集）中仅包含两条用于SM3实现的专用指令，分别为sm3p0和sm3p1指令。本文在同一器件上进行了基于*K*指令集与本文自定义指令的加速实现效果实验。本文方案与RISC-V扩展*K*指令集对比如表4所示。

表4　本文方案与RISC-V扩展*K*指令集对比

SM3核心部分实现	面积/slice	时钟频率/MHz	时钟周期/cycle	吞吐率/（Mbit·s^{-1}）
基于sm3p0和sm3p1指令	57	357	6 096	30
基于本文指令	374	155	64	1 240

由表4可知，标准*K*指令方案硬件开销较低，仅占用57 slice，但单次消息扩展与压缩迭代需6 096个时钟周期，吞吐率为30 Mbit/s。本文方案占用374 slice，单次运算周期降至64个周期，吞吐率达到1 240 Mbit/s，约为标准*K*指令方案的41.3倍，显著提高了SM3核心处理效率。在面积增加约5倍的情况下，实现40倍吞吐率的增长。

（2）与其他方案对比

本文方案与SM3算法其他实现方案对比如表5所示。由表5可以看出，本文方案在155 MHz工作频率下实现了1 240 Mbit/s的吞吐率，优于文献[26]、文献[27]、文献[11]的方案。本文方案面积为374 slice，与低资源开销方案效果持平。在低资源开销前提下，本文方案实现了较高吞吐率，能够满足SM2签名验签过程中对SM3压缩计算的加速需求，取得了较好的综合性能。

表5　本文方案与其他方案SM3算法对比

文献	器件	面积/slice	时钟频率/MHz	时钟周期/cycle	吞吐率/（Mbit·s^{-1}）
本文方案	Artix-7	374	155	64	1 240
文献[26]	EP1C20F324C6	1 562	89.3	64	714.4
文献[27]	ASIC 110nm	377	36	69	66.78
文献[28]	Cyclone IV	9.11×10^4	227	—	80.43×10^3
文献[11]	Artix-7	280	154	64	1 232

5.2.4　数字签名实现方案比较

本文方案与其他方案数字签名算法对比如表6所示。由表6可以看出，本文方案在签名与验签两种场景下均取得了较好的综合性能。在68 MHz工作频率下，本文方案签名和验签时间分别为2.92 ms和3.40 ms，对应AT值分别为11.77和13.70。与文献[10]相比，本文方案虽然工作频率较低，但面积仅为4.03 kslice，明显优于其7.15 kslice，且验签AT值由32.25降至13.70，综合效率更高。与文献[25]相比，本文方案在验签时间上仍有差距，但面积开销更小。本文方案在执行时间方面，略逊于文献[30]、文献[31]等方案，但资源消耗明显更低。总体来看，本文方案在面积资源执行性能之间取得了较好的平衡。

6　结束语

本文针对资源受限嵌入式平台下数字签名验签执行效率不高的问题，设计并实现了一种基于RISC-V自定义扩展指令的密码协处理器，用于软硬件协同加速国密数字签名算法。该协处理器将

表6　本文方案与其他方案数字签名算法对比

文献	器件	面积/kslice	时钟频率/MHz	时间/ms		AT	
				签名	验签	签名	验签
本文方案	Artix-7	4.03	68	2.92	3.40	11.77	13.70
文献[29]	Virtex-5	3.42	80	131.45	134.46	449.56	459.85
文献[27]	Zynq-7	6.30	50	12.84	26.23	80.892	165.25
文献[10]	Zynq-7	7.15	110	2.28	4.51	16.30	32.25
文献[30]	Kintex UltraScale XCKU060	63.37	27	0.14	0.28	8.87	17.74
文献[25]	Zynq UltraScale+ MPSoC	8.38	60	—	2.08	—	17.43
文献[31]	Kintex-7	8.63	50	0.98	1.74	8.46	15.02

SM2 中的有限域关键运算和 SM3 扩展压缩过程映射为指令，由主处理器与协处理器共同完成计算。实验结果表明，本文方案可有效提升 SM2 标量乘、SM3 扩展压缩以及签名验签的执行效率，优于已有实现方案。本文方案在运算性能、资源开销和实现灵活性之间取得了较好的平衡。

参考文献：

[1] Nuclei. Hummingbirdv2 E203 Core and SoC[EB]. (2022-10-27)[2026-04-08].

[2] Asanović K, Avizienis R, Bachrach J, et al. The rocket chip generator[R]. Berkeley: EECS Department, University of California, Berkeley, 2016.

[3] Marshall B, Saarinen M J O. RISC-V cryptography extensions volume I: scalar & entropy source instructions (version 1.0.0)[R]. RISC-V International, 2021.

[4] Kabin I, Dyka Z, Klann D, et al. Resistance of the Montgomery ladder against simple SCA: theory and practice[J]. Journal of Electronic Testing, 2021, 37(3): 289-303.

[5] 乔晗, 王安, 王博, 等. 面向GmSSL密码库的SM2算法快速优化实现[J]. 计算机学报, 2025, 48(2): 463-476.
Qiao H, Wang A, Wang B, et al. Optimized implementation of the SM2 algorithm on the GmSSL cryptographic library[J]. Chinese Journal of Computers, 2025, 48(2): 463-476.

[6] 杨先伟, 康红娟. SM3 杂凑算法的软件快速实现研究[J]. 智能系统学报, 2015, 10(6): 954-959.
Yang X W, Kang H J. Fast software implementation of SM3 Hash algorithm[J]. CAAI Transactions on Intelligent Systems, 2015, 10(6): 954-959.

[7] 张盛仕, 胡湘宏, 熊晓明. 基于国密算法 SM2 软硬件协同系统的 FPGA 架构[J]. 单片机与嵌入式系统应用, 2019, 19(7): 15-19.
Zhang S S, Hu X H, Xiong X M. FPGA architecture of software and hardware co-design based on national secret algorithm SM2[J]. Microcontrollers & Embedded Systems, 2019, 19(7): 15-19.

[8] Kong T J. FPGA implementation of a high-performance SM2 coprocessor[C]//Proceedings of the 2023 3rd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI). Piscataway: IEEE Press, 2023: 174-179.

[9] Zheng X, Hu X H, Zhang J L, et al. An efficient and low-power design of the SM3 hash algorithm for IoT[J]. Electronics, 2019, 8(9): 1033.

[10] 王腾飞, 张海峰, 许森. SM2 专用指令协处理器设计与实现[J]. 计算机工程与应用, 2022, 58(2): 102-109.
Wang T F, Zhang H F, Xu S. Design and implementation of SM2 coprocessor with specific instructions[J]. Computer Engineering and Applications, 2022, 58(2): 102-109.

[11] 王明登, 严迎建, 郭朋飞, 等. 基于RISC-V指令扩展方式的国密算法SM2、SM3和SM4的高效实现[J]. 电子学报, 2024, 52(8): 2850-2865.
Wang M D, Yan Y J, Guo P F, et al. Efficient implementation of national security algorithms SM2, SM3 and SM4 based on RISC-V instruction extension method[J]. Acta Electronica Sinica, 2024, 52(8): 2850-2865.

[12] 胡振波. 手把手教你RISC-V CPU处理器设计(上)[M]. 北京: 人民邮电出版社, 2021.
Hu Z B. Step-by-step guide to RISC-V CPU (Volume I): processor design[M]. Beijing: Posts & Telecom Press, 2021.

[13] Karatsuba A A, Ofman Y P. Multiplication of multidigit numbers on automata[J]. Soviet Physics Doklady, 1963, 7: 595-596.

[14] Comba P G. Exponentiation cryptosystems on the IBM PC[J]. IBM Systems Journal, 1990, 29(4): 526-538.

[15] Rafferty C, O'Neill M, Hanley N. Evaluation of large integer multiplication methods on hardware[J]. IEEE Transactions on Computers, 2017, 66(8): 1369-1382.

[16] Montgomery P L. Modular multiplication without trial division[J]. Mathematics of Computation, 1985, 44(170): 519-521.

[17] Barrett P. Implementing the rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor[C]// Advances in Cryptology—CRYPTO'86. Berlin: Springer, 1987: 311-323.

[18] Solinas J A. Generalized mersenne prime[M]//Tilborg H C A V, Jajodia S. Encyclopedia of cryptography and security. Berlin: Springer, 2011: 509-510.

[19] Hars L. Modular inverse algorithms without multiplications for cryptographic applications[J]. EURASIP Journal on Embedded Systems, 2006(1): 1-13.

[20] Ahmadi H R, Sim M L, Hwang K, et al. A low-power and low-energy flexible GF(p) elliptic-curve crypto-processor design based on prime-field arithmetic[J]. Journal of Zhejiang University-SCIENCE C, 2010, 11(11): 909-922.

[21] Randolph M, Diehl W. Power side-channel attack analysis: a review of 20 years of study for the layman[J]. Cryptography, 2020, 4(2): 15.

[22] Ananyi K, Alrimeih H, Rakhmatov D. Flexible hardware processor for elliptic curve cryptography over NIST prime fields[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2009, 17(8): 1099-1112.

[23] Marzouqi H, Al-Qutayri M, Salah K. An FPGA implementation of NIST 256 prime field ECC processor[C]//Proceedings of the 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS). Piscataway: IEEE Press, 2013: 493-496.

[24] Hu X H, Cai S T, Zhan R D, et al. High performance SM2 elliptic curve cryptographic processor over GF(p)[C]//Proceedings of the 2019 Chinese Control Conference (CCC). Piscataway: IEEE Press, 2019: 8904-8908.

[25] 魏文鹏. 基于软硬协同的双域SM2算法加速器设计与实现[D]. 武汉: 华中科技大学, 2024.
Wei W P. Design and implementation of a dual-domain SM2 accelerator based on hardware-software co-design[D]. Wuhan: Huazhong University of Science and Technology, 2024.

[26] 于永鹏, 严迎建, 李伟. SM3 算法高速 ASIC 设计及实现[J]. 微电子学与计算机, 2016, 33(4): 21-26.
Yu Y P, Yan Y J, Li W. High-speed ASIC design and implementation of SM3 algorithm[J]. Microelectronics & Computer, 2016, 33(4): 21-26.

[27] Zheng X, Xu C Y, Hu X H, 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.

[28] 方轶, 丛林虎, 邓建球, 等. 基于FPGA的SM3算法快速实现方案[J]. 计算机应用与软件, 2020, 37(6): 259-262.
Fang Y, Cong L H, Deng J Q, et al. Fast implementation of SM3 algorithm based on FPGA [J]. Computer Applications and Software, 2020, 37(6): 259-262.

[29] Liu Y H, Guo W, Tan Y, et al. An efficient scheme for implementation of SM2 digital signature over GF(p)[C]//Contemporary Research on E-business Technology and Strategy. Berlin: Springer, 2012: 250-258.

[30] 李斌, 周清雷, 陈晓杰, 等. 可重构的素域SM2算法优化方法[J]. 通信学报, 2022, 43(3): 30-41.
Li B, Zhou Q L, Chen X J, et al. Optimization of reconfigurable SM2 algorithm over prime field[J]. Journal on Communications, 2022, 43(3): 30-41.

[31] 孔团结, 郑昉昱, 郭润, 等. 低功耗软硬结合SM2算法实现[J]. 密码学报(中英文), 2024, 11(6): 1354-1369.
Kong T J, Zheng F Y, Guo R, et al. Implementation of low-power software and hardware combination for SM2 algorithm[J]. Journal of Cryptologic Research, 2024, 11(6): 1354-1369.

[作者简介]



李秀滢（1975– ），女，北京电子科技学院副教授、硕士生导师，主要研究方向为密码芯片设计、密码工程、智能系统安全。



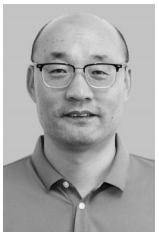
鄂佳言（2001– ），女，北京电子科技学院硕士生，主要研究方向为密码芯片设计及安全防护技术。



武秀云（2000– ），女，北京电子科技学院硕士生，主要研究方向为信息安全、计算机视觉。



杨亚涛（1978– ），男，博士，北京电子科技学院教授、博士生导师，西安电子科技大学硕士生导师，主要研究方向为密码协议和算法、后量子密码、全同态加密、信息安全等。



李兆斌（1977– ），男，博士，北京电子科技学院教授、硕士生导师，主要研究方向为密码算法实现与测评、下一代网络安全等。