DOI:10.19651/j.cnki.emt.2519822

基于软硬协同的 Paillier 同态加密算法加速方案设计*

阮笃钧 周 骅
(贵州大学大数据与信息工程学院 贵阳 550025)

摘 要:本文针对 Paillier 同态加密算法在加密及同态运算中存在的计算效率低及灵活性不足的问题,设计并实现了一种 Paillier 同态加密算法加速方案。该方案通过软硬协同设计技术,高效处理算法预计算、数据交互以及算法操作解析要求,有效提升了方案灵活性并降低了资源占用量。并且,方案通过定制化设计并使用了双高基数蒙哥马利模乘核心,显著提升了方案计算吞吐量及实时性。测试结果表明:方案对算法关键计算步骤的加速效果显著。在 1 024 位计算宽度下,模乘与模幂的计算的平均时延约为 0.523、667.42 μs。相较于 Intel I9-13900HX 处理器,其时延分别降低约 68.74%与 42.76%(对应加速 3.20 倍与 1.75 倍)。所提出的方案能够为安全多方计算、联邦学习提供高效隐私计算支持。

关键词:软硬协同;Paillier 算法;蒙哥马利算法;模运算
中图分类号:TN918.91 **文献标识码**:A **国家标准学科分类代码**:520.60

Design of a Paillier encryption acceleration system based on software-hardware co-design

Ruan Dujun Zhou Hua
(College of Big Data and Information Engineering, Guizhou University, Guiyang 550025, China)

Abstract: This paper addresses the issues of low computational efficiency and insufficient flexibility encountered during the encryption and homomorphic operations of the Paillier homomorphic encryption algorithm. We design and implement an acceleration scheme for the Paillier algorithm. Utilizing software-hardware Co-design technology, this scheme efficiently processes algorithmic precomputation, data interaction, and the requirements for parsing algorithm operations, thereby effectively enhancing its flexibility and reducing resource consumption. Furthermore, significant improvements in computational throughput and real-time performance are achieved through the customized design and implementation of a dual-high-radix Montgomery modular multiplication core. Test results demonstrate a significant acceleration effect on the algorithm's critical computational steps. Under a 1 024-bit computational width, the average latencies for modular multiplication and modular exponentiation are approximately 0.523 and 667.42 μs, respectively. Compared to an Intel Core i9-13900HX processor, these latencies are reduced by approximately 68.74% and 42.76% (corresponding to speedups of 3.20× and 1.75×). The proposed scheme is capable of providing efficient privacy computation support for secure multi-party computation and federated learning.

Keywords: software-hardware co-design; Paillier algorithm; Montgomery algorithm; modular arithmetic

0 引 言

Paillier 同态加密算法允许多个参与方在不泄露隐私数据的前提下协同完成计算任务[1]。其在安全多方计算、联邦学习以及医疗联合分析等领域具有巨大的实际应用需求[2-4]。然而由于 Paillier 算法存在高位宽的模计算操作,使得传统软件计算存在显著的效率瓶颈,难以满足高并发、低延迟的实际部署需求[5-7]。因此,如何在保障多方数据安全的同时,确保其计算效率和可用性,成为当前研究的核心问题。

现今的加速方案主要有硬件逻辑设计以及软硬协同设计两大类,区别在于是否使用内核处理器以提升方案整体灵活性。而在算法层面,国内外常使用对硬件实现友好的蒙哥马利算法[8-9]。文献[10]提出一种专用 ASIC 硬件架

收稿日期:2025-09-09
*基金项目:国家自然科学基金(62272123)项目资助

构搭配 256 基蒙哥马利乘法器的设计，取得了极佳的加速效果。但该设计受限于架构固化且成本高昂使其只能应用于特定场景。文献[11]提出一种基于现场可编程门阵列(field-programmable gate array，FPGA)的加速方案，通过高层次综合(high-level synthesis，HLS)构建了紧凑的 32 基蒙哥马利模乘单元。在保障设计灵活性的同时通过部署并行加密核心在 FPGA 上实现了较高吞吐量的计算操作。但受限于单个核心的资源量较少，其单次操作的最低延迟没有得到太大的提升。文献[12]提出使用一种基于软硬协同(Software-Hardware Co-design)的加速方案，通过构建多核微码驱动的 72 基蒙哥马利模乘单元最终实现了优异的面积效率与数据吞吐量能力。但由于微码驱动开发复杂反而失去了软硬协同计算具有的灵活性优势。且由于核间存储瓶颈存在，导致计算时延的提升有限。文献[13]基于 FPGA 提出了一种 2 基蒙哥马利算法的方案，并设计了一种低延迟的压缩加法器结构。显著降低了模乘运算的面积时间积(area-time product，ATP)。在资源占用方面具有极佳的优势但在数据吞吐量与操作延迟上存在一定劣势。

综上所述，文献[10]的性能较好、资源消耗一般但灵活性较差，文献[11]的灵活性极佳、资源消耗少但计算性能有限，文献[12]的资源消耗量较少、性能较为理想但灵活性不足，文献[13]具有最佳的资源效率但性能与灵活性均较差。以上文献的计算性能均未有突破之处。故而方案旨在设计一种基于软硬协同的 Paillier 同态加密算法加速设计方案，以解决安全多方计算等场景下的高速率、强时效计算需求。力图在保障足够灵活性与合理资源使用量的前提下，进一步加强性能方面的提升。

方案采用计算单元并行化计算技术和双核高基数蒙哥马利模乘模块提高了模数计算性能。对于计算时效性需求，通过逻辑循环展开以及计算路径复用等方法减少了计算延迟，提高了计算响应速度与时序裕量。通过软硬协同控制与计算，降低了软件逻辑控制复杂度，并去除了轻量级参数计算对硬件资源的占用。使得实现复杂度降低的同时，增强了方案的灵活性与计算效率。为 Paillier 算法应用提供了兼具高性能与灵活性的加速支撑。

1　基础知识

1.1　Paillier 同态加密算法

Paillier 同态加密算法是一种加性同态加密算法，完整的 Paillier 加密算法有密钥生成、加密、解密 3 个过程。用户首先需要生成公钥 (n,g) 与私钥 (λ,μ)。首先，使用随机数筛选出两个不同的大素数 (p,q)，并令公钥 $n=p\cdot q$。再随机选取整数 $g\in\mathbb{Z}_{n^2}^{*}$ 作为另一个公钥，或是选择 $g=n+1$ 用于简化计算。私钥 λ 可由 $(p-1)$ 与 $(q-1)$ 的最小公倍数计算得出：

$$\lambda=lcm(p-1,q-1) \tag{1}$$

私钥 μ 通过公钥对 (n,g) 计算得出：

$$\mu=L(g^{\lambda}\ mod\ n^2)^{-1}mod\ n \tag{2}$$

$$L(x)=\frac{(x-1)}{n} \tag{3}$$

使用公钥对与随机数 $r(0<r<n)$ 即可生成密文 c，$c=[\![m]\!]=g^{m}r^{n}\ mod\ n^2$，而要将密文 c 解密则使用私钥对 (λ,μ) 进行解析：

$$m=\mu\cdot L(c^{\lambda}\ mod\ n^2)mod\ n \tag{4}$$

密文 $([\![m_1]\!],[\![m_2]\!])$ 间进行同态加的过程用 $\oplus$ 表示，密文与标量 $([\![m_1,a]\!])$ 相乘的过程则用 $\odot$ 表示。其公式如下：

$$[\![m_1]\!]\oplus[\![m_2]\!]=([\![m_1]\!]\cdot[\![m_2]\!])\ mod\ n^2=[\![m_1+m_2]\!] \tag{5}$$

$$[\![m_1]\!]\odot a=([\![m_1^{a}]\!])\ mod\ n^2=[\![m_1\cdot a]\!] \tag{6}$$

Paillier 算法以位操作量作为时间复杂度的基本单位，而空间复杂度以操作数长度 k －bit 为基本单位，对应加密、解密、同态计算的复杂度如所示。表 1 所示为 Paillier 算法的复杂度总结。Paillier 算法的加密和解密操作时间复杂度均为 $O(k^3)$，其核心瓶颈在于 g^{m} 和 r^{n} 对模数 n^2 的模幂运算。同理，标量乘法的核心瓶颈为 $[\![m_1^{a}]\!]$ 对 n^2 的模幂计算。而同态加法仅需 $O(k^2)$ 时间复杂度，这是因其本质为单次模乘操作。所有操作的空间复杂度均为线性，源于密文固定膨胀为 $2k$ －bit 且无需超线性存储。

表 1　Paillier 算法复杂度总结

Table 1　Summary of Paillier algorithm complexity

操作	时间复杂度(位操作)	空间复杂度/bit
加密	$O(k^3)$	$O(k)$
解密	$O(k^3)$	$O(k)$
同态加法	$O(k^2)$	$O(k)$
标量乘法	$O(k^3)$	$O(k)$

1.2　蒙哥马利算法

蒙哥马利算法是一种高效模运算方法。该算法的核心思想是通过将数据转换到蒙哥马利域中，把原本耗时的模除法操作转化为更快的乘法和移位运算，如算法 1 所示。其中数据 A 的蒙哥马利形式表示为 $\bar{A}=AR(MODN)$，R 是计算基数，通常取 2 的幂次以便于通过移位进行高效计算。

算法 1

输入：$\overline{A},\overline{B},N,R(R=2^{w},w=1,2,3\cdots)$，
$N'(N'=-N^{-1}(MOD\ R),N<R)$

输出：$T=ABR(MOD\ N)$

a) $T=\overline{A}\cdot\overline{B}$

b) $T=(T+(T\cdot N'(MOD\ R))\cdot N)/R$

c) *if* $T>N$ 输出 $T-N$

d) *else* 输出 T

在蒙哥马利算法中,由于计算基数 R 为 2 的幂次,所以任意对 R 的取模计算与除法计算的计算结果在硬件中均可以用右移 R 次获得。这规避了模运算中计算复杂度最高的除法计算。但仍存在高位宽的乘加计算,这对硬件的资源占用与布局是毁灭性的。所以如算法 2 所示,需要对基数 R 进行拆分,以规约迭代的方式进行乘加计算与移位操作。本方案中将基数 R 设定为 $2^8 = 256$。

算法 2

输入:$\overline{A}(\overline{A}_1,\overline{A}_2,\cdots,\overline{A}_{num})$,$\overline{B}(\overline{B}_1,\overline{B}_2,\cdots,\overline{B}_{num})$,
$N(N_1,N_2,\cdots,N_{num})$,$N'$,$R(R=256)$

输出:$T = ABR(MOD\ N)$

a) 变量 $C \leftarrow 0$,$S \leftarrow 0$,$M \leftarrow 0$,$T \leftarrow 0$
b) *For* $i = 0$ *to* $num - 1$
c) $C \leftarrow 0$
d) *For* $j = 0$ *to* $num - 1$
e) $(C,S) \leftarrow T[j] + \overline{A}[j] * \overline{B}[i] + C$
f) $T[j] \leftarrow S$
g) $(C,S) \leftarrow T[num] + C$
h) $T[num] \leftarrow S$
i) $T[num] \leftarrow C$
j) $C \leftarrow 0$
k) $M \leftarrow T[0] * N'(MOD\ R)$
l) $(C,S) \leftarrow t[0] + M * N[0]$
m) *For* $j = 1$ *to* $num - 1$
n) $(C,S) \leftarrow T[j] + M * N[j] + C$
o) $T[j-1] \leftarrow S$
p) $(C,S) \leftarrow T[s] + C$
q) $T[s-1] \leftarrow S$
r) $T[s] \leftarrow T[s+1] + C$

而蒙哥马利的时、空间复杂度与传统模乘算法相当,时间复杂度为 $O(k^2)$ 来源于算法中的两层循环模约减,其核心优势在于更少的常数因子参与计算,避免了传统模计算中反复计算商的操作。空间复杂度为 $O(k)$,但需额外存储中间计算结果。蒙哥马利算法用常数时间的加、乘法以及移位操作替代了昂贵的除法计算,这不仅显著的提升了大数模乘的计算效率,还由于其算法流程规整,无显式除法/复杂分支,天然适配恒定时间实现。若严格避免由 $T > N$ 带来的减法修正数据依赖分支,则可有效抵抗时序攻击(timing attack, TA)和简单功耗分析(simple power consumption analysis,SPA)。为了避免结果数据与模数 N 的条件减法分支并保持常数时间,方案采取将结果保持在 $[0,R+N-1]$ 的区间内,则有:

$$T = (T + mN)/R = (AB + mN)/R \tag{7}$$

$$T < (R^2 + RN)/R = R + N \tag{8}$$

此时,数据与模数 N 的条件减法分支变为与 R 的单次整数比较,当结果小于 R 时可执行与 0 的减法,两个因素共同作用下是的算法计算开销为常数时间,保障了方案抗测信道攻击的能力。

2 软硬协同加速方案设计与实现

2.1 方案总体设计框架

本方案提出的 Paillier 同态加密硬件架构如图 1 所示,方案总体由基于 FPGA 的可编程逻辑(programmable logic, PL)以及基于 ARM 内核的处理系统(processing system, PS)两部分组成。PS 端负责接收、存储输入数据与算法指令。指令由算法解析模块处理,解析出具体算法步骤;而预计算模块与随机数生成模块二者由接收的数据生成结果并将其存储于存储模块中。前述指令与处理后的数据通过高速可扩展接口(advanced extensible interface, AXI)传输至 PL 端。PL 端通过 AXI 接口接收并存储来自 PS 的指令与数据,随后,状态机控制模块根据指令需求,调度存储模块中的数据输入至计算模块进行计算。该计算模块核心包含蒙哥马利模乘、模幂与数学计算单元。计算任务完成后,状态机控制模块驱动计算结果由计算单元返回至 PL 端存储模块,供 PS 后续读取使用。

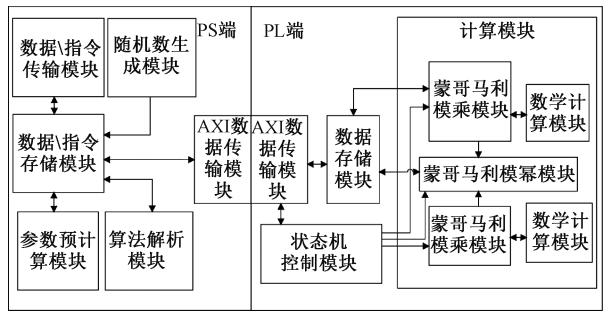

图 1 方案架构

Fig. 1 Solution architecture

2.2 方案工作流程

方案的工作流程如图 2 所示,首先,PS 端通过安全信道接收上级终端的公私钥,以此进行密钥更新和蒙哥马利参数预计算。随后,PS 端等待接收计算指令和计算数据。在完成算法操作解析后,通过 AXI 将数据传输至 PL 端的存储器中,同时指令状态接口电平维持在正确状态。接下来,PL 端在读取数据的同时,依据指令状态接口电平决定计算任务类型并执行加速计算,具体包括模幂、模乘和除法等计算密集型任务。最后,计算结果通过输出逻辑回传至 PL 端存储器中,供后续 PS 端计算或读取使用。整个软硬件协同流程中,轻量级逻辑、计算任务(如密钥更新、参数预计算、算法解析和数据收发)由 PS 端处理,计算密集型任务则由 PL 端硬件加速完成,方案通过 AXI 接口实现高效的数据交互与任务调度。

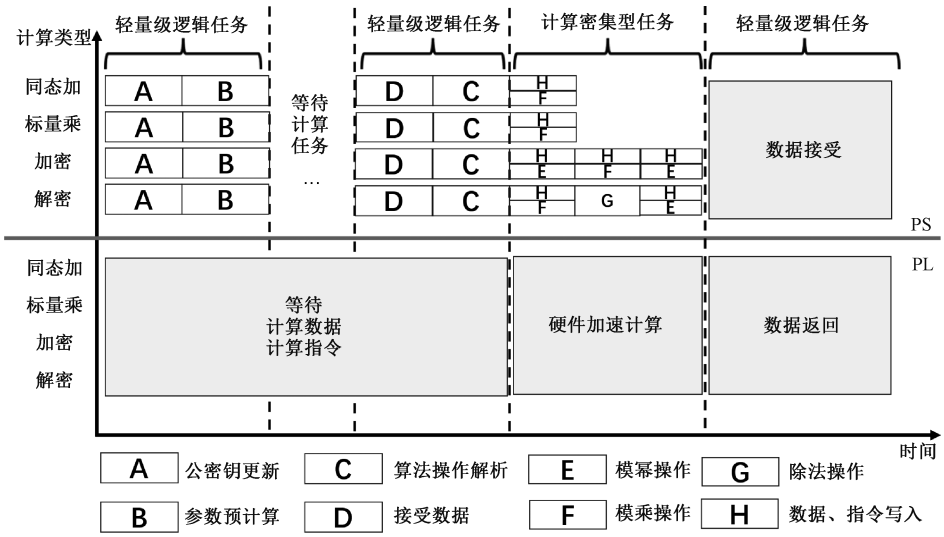

图 2　方案工作流程图

Fig. 2　Workflow diagram of the plan

方案中于 PS 端执行公密钥更新以及蒙哥马利参数预计算任务在整个密钥生命周期内仅需执行一次，而其他操作则可根据算法任务需求反复执行。

2.3　蒙哥马利模乘模块实现

蒙哥马利模乘算法在计算过程中存在高位宽的乘加计算，这对硬件资源与数据路径时序收敛提出了较高的要求。FPGA 的 PL 资源中设计有用于数学计算的硬核数字信号处理器(digital signal processor，DSP)知识产权核心(intellectual property core，IP Core，)。但本设计使用的 VIVADO 的 DSP48E IP 最高仅能处理 27×18＋47＋进位的乘加计算或是 48＋48＋进位的加法计算，需要将 IP 级联以处理更高位宽的输入数据。32 b 乘法器最小处理单元如图 3 所示，将 32 比特的输入数据以高低 16 比特拆分依次输入到寄存器中。寄存器在数据稳定后又将数据输入到 DSP48E IP 中。在经过带权乘加计算后，一级乘加单元会将结果最低 16 b 数据稳定输出并分离出高 16 b 数据参与下一级乘加计算，而其余三级乘加单元会依次与前一级的高位数据以及进位数据进行累加。加法树的最终结果会将结果数据与溢出数据分别输出到后续寄存器与其他处理单元中，以此构建高位宽乘法模块。

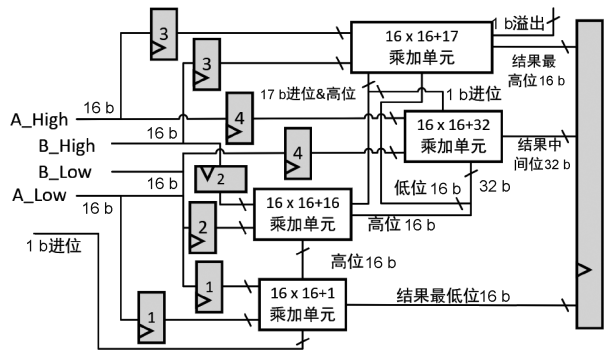

图 3　32 比特乘法器结构

Fig. 3　The structure of a 32-bit multiplier

高位宽乘加单元的内部结构如图 4 所示，设计以 128×1 024 b 乘法单元作为第二级计算单元。在该计算单元中本文以 32 b 乘法、加法单元为基础，通过带权重加法单元对每个乘法单元的结果求和构造了 32×1 024 b 乘法单元。再通过由 32 b 加法单元级联组成的 1 056 b、1 024 b 加法器构造加法树。为降低数据输出延迟，在加法树最后一级中仅使用 1 024 b 加法器对中间位部分和进行计算。对最高位 32 b 数据与次高位 32 b 数据采取使用查找表(look-up table，LUT)求进位结果和，这能使得 DSP 资源使用量轻微减少，节省了片上资源使用量并减轻了时序约束收敛压力。设计以同样的架构思路设计了 256×1 024＋1 280 b 乘加计算单元作为第 3 级计算单元以供蒙哥马利模乘模块使用。

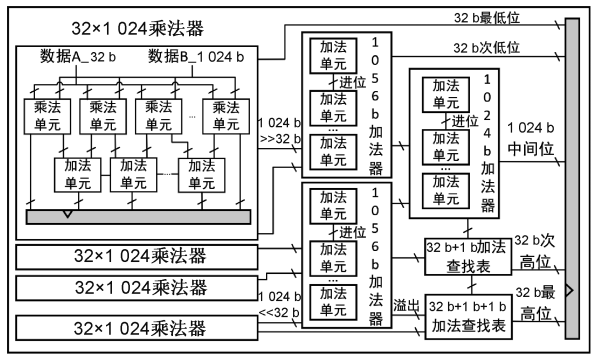

图 4　128×1 024 乘法器结构

Fig. 4　The structure of a 128×1 024 multiplier

如图 5 所示是蒙哥马利模乘模块状态机的顶层结构图。状态机系统初始处于空闲状态，在经外部标志位激励触发后，控制内部逻辑从随机存取内存块(block random

access memory, BRAM)中读取待模乘数据 A、B;模数 N 以及由软件预计算的逆元 N',待完成数据加载任务后将状态转移至 A、B 部分积计算状态,在此状态下,系统基于蒙哥马利算法流程与模约减循环计数器决定是单纯计算 A、B 部分积还是模约减计算并存入结果数据寄存器 P 中。随后,状态机进入参数 M 计算状态,根据当前结果数据 P 计算参数 M。参数 M 使用的 256×256 b 高位截断乘法器是特化用于专门计算参数 M 的乘法器,其与 A、B 部分积乘加单元的区别是去除了高位部分的乘法器结构与加法树结构,由于基数 $R = 2^8 = 256$,因此高位部分并不参与模约减计算,这在一定程度上缓解了片上资源与时序压力。待参数 M 准备外部以供模约减计算状态使用后,系统复用同一乘加单元计算当前部分积的模约减结果。通过模约减循环计数器决定是否继续部分积的计算。最后,蒙哥马利算法需要把把计算结果和模数 N 比较,以保证结果范围为[0,2N−1]内,以供下一次蒙哥马利模乘计算。所以,最后需要判断结果数据是否超过 1 024 b 的约束条件,以此来决定是否对计算结果 P 减去模数 N 或是 0 的减法计算,来保证系统计算过程为常数时间,从而避免潜在的侧信道攻击。当减法计算完成或是空计算后,状态机进入数据输出状态。将模乘结果输入 BRAM 中供其余计算单元使用或是供 PS 端读取调用。

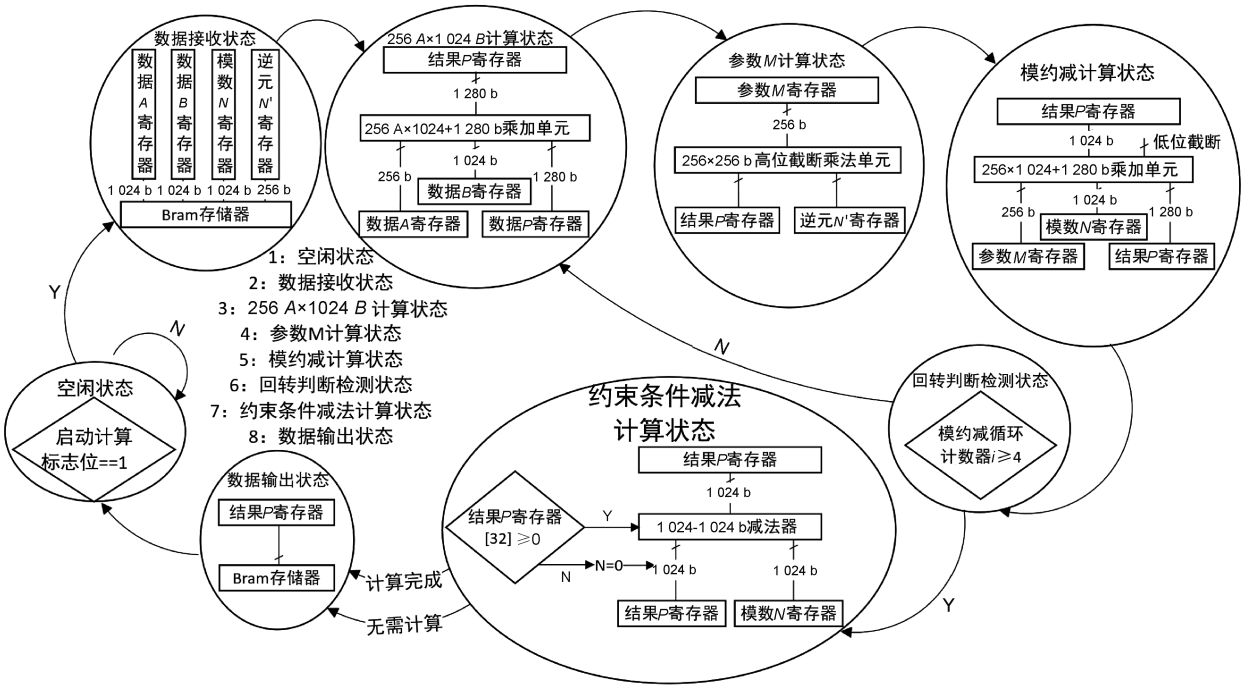

图 5 蒙哥马利模乘状态机结构

Fig. 5 Montgomery modular multiplication state machine structure

2.4 蒙哥马利模幂 L-R 模块实现

蒙哥马利模幂从左至右(left-right, L-R)计算是依托蒙哥马利模乘算法与重复平方乘算法构建的。其算法流程如算法 3 所示。首先,将数据 A 转化成蒙哥马利域形式,并将数值 1 也转换到蒙哥马利域并存入变量 X 中。然后,以数据 E 的 2 进制位数值从左至右做判断循环,当该位 2 进制值为 1 时,对蒙哥马利域数据 A 与 X 做模乘计算,否则对数据 A 做模平方计算。最后,当计数器值抵达数据 E 最高位后,对蒙哥马利域结果数据 X 做与数据 1 的蒙哥马利模乘使之退出蒙哥马利域,得到最终模幂计算结果。

蒙哥马利模幂模块的状态机结构如图 6 所示,该结构采用包含 6 个核心状态的有限状态机(finite state machine, FSM)设计。与模乘模块相同,状态机系统初始处于空闲状态等待启动信号。之后,系统在数据接收状态

算法 3

输入:$A, E(E_1, E_2, \cdots, E_j)_2, N, N'$

输出:$X = A^E (MOD\ N)$

a) $X \leftarrow 1$

b) $\overline{A} \leftarrow \mathrm{MontMult}(A, R^2(\mathrm{MOD}\ N), N, N')$

c) $\overline{X} \leftarrow \mathrm{MontMult}(X, R^2(\mathrm{MOD}\ N), N, N')$

d) *For* $i = 0$ *to* $j - 1$

e) if($E_i == 1$)

f) $\overline{X} \leftarrow \mathrm{MontMult}(\overline{X}, \overline{A}, N, N')$

g) $\overline{A} \leftarrow \mathrm{MontMult}(\overline{A}, \overline{A}, N, N')$

h) $X \leftarrow \mathrm{MontMult}(\overline{X}, 1)$

接收底数 A、指数 E、模数 N、预计算参数 $R^2(\mathrm{MOD}\ N)$以及软件预先调用模乘模块计算得到的数据 1 的蒙哥马利

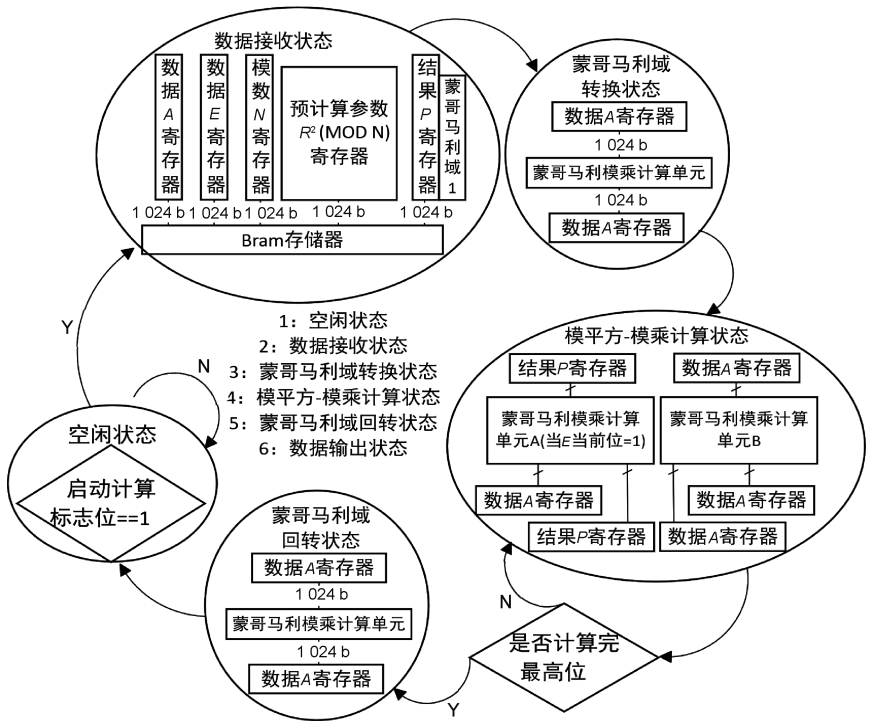

图 6　蒙哥马利模幂状态机结构

Fig. 6　Montgomery modular exponentiation state machine structure

域形式数据。然后，系统进入蒙哥马利域转换状态，将底数 A 与预计算参数 R^2（MOD N）输入到蒙哥马利模乘模块中，并接收转换后的数据。当数据准备完成后系统就进入模平方算法计算模块。在算法 3 中，由于模平方计算与模乘计算之间不存在结果数据与模平方数据间的数据依赖，故而在模幂模块中设计有两块能够独立计算的蒙哥马利模乘模块，用于同时处理算法 3 中的模乘与模平方计算。通过双核并行计算能够将循环计算的延迟降低一倍，而对应的资源消耗量也将提升。

2.5　PS 端设计与实现

PS 端的顶层结构如图 7 所示。包含板载双倍速率同步动态随机存储器（double data rate synchronous dynamic random access memory，DDR SDRAM）（简称 DDR 存储器）、算法解析模块、预计算模块、随机数生成模块、以太网传输模块和 AXI 传输逻辑模块等核心组件。其中，内外数据通信模块负责从外部设备接收计算需求与数据到板载 DDR 中同时也负责传输计算结果到外部设备存储器。而板载 DDR 会将计算需求与计算数据传输至算法解析模块与 AXI 传输逻辑模块。预计算模块生成必要的预计算数据，随机数生成模块提供密码学安全的随机数，两者同样通过 AXI 总线传输至 PL 端供计算。当 PL 端开始计算时会拉低扩展多用途输入输出口（extended multi-purpose input/output，EMIO）的电平信号，完成计算后则将其重新拉高，以测试 PL 端实际计算时间。

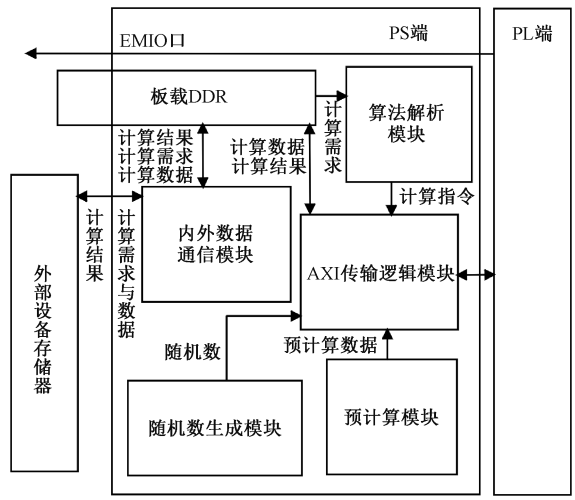

图 7　PS 端顶层结构

Fig. 7　PS structure top-level

3　加速方案测试

3.1　测试设置

本文使用的硬件平台为搭载了 Xilinx 的 ZYNQ XCZU15EG-FFVB1156-2-i 型号芯片的开发板，该开发板搭载型号芯片，该芯片提供丰富的硬核 DSP48E IP 和可编程逻辑块（configurable logic block，CLB）资源。上位测试系统使用 window11 25H2 版本操作系统用以模拟顶端

设备产生密钥与计算数据。性能测试数据集基于 Python3 使用 PHE 库生成随机密钥与随机计算数据,包含 Paillier 加密计算与独立的模乘、模幂计算。系统硬件部分使用 Verilog HDL 设计,并通过 Vivado 2024.1 工具完成仿真、综合和布局布线的完整实现流程。

加速系统支持的密钥位宽为 512 b,可以处理 512 b 位宽数据的加密、同态加、标量乘法计算。并可独立处理 1 024 b 宽度的模乘、模幂计算。使用随机数据测试下,系统在 512 b 密钥下加密计算吞吐量为 3 649.2 OPS,模乘计算吞吐量为 1 875.8 KOPS 而模幂计算吞吐量为 1 498.3 OPS。

3.2 测试结果

如图 8 所示是搭建的测试系统,计算机使用自建的测试软件通过交换机与测试平台的内外数据通信模块进行数据、指令通讯传输。示波器通过探针检测测试 EMIO 口输出的计算状态响应信号,以此得到实际计算时间。

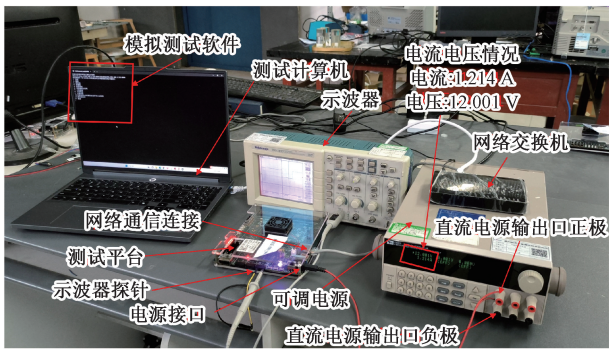

图 8 测试系统

Fig. 8 Test system

本文设计的加速方案,其 PL 端核心电路的工作频率为 287 MHz,在满足计算电路时序约束要求下,加速系统的资源消耗情况如表 2 所示。其中查找表、触发器(flip flop, FF)、以及 DSP 块的资源占用分别为 57.42%、29.28%、74.69,这些资源用于构成逻辑控制、部分数据暂存、以及组成乘加计算单元。

表 2 系统资源占用率

Table 2 System resource utilization rate

FPGA 资源	可用	已用	占比/%
LUT	341 280	195 955	57.42
FF	682 560	199 844	29.28
DSP	3 528	2 635	74.69

为分析功耗与性能实际情况,表 3 展示了本文加速系统与中央处理器(central processing unit, CPU)在模乘、模幂运算上的计算时延和功耗对比结果。两者同样使用蒙哥马利算法,本文设计的系统运行于 287 MHz 工作频率下,测试使用的英特尔 CPU 在运行测试程序时的睿频加速(turbo boost, TB)工作频率约为 4.4 GHz。本文加速系统与 Intel I9-13900HX CPU 相比,在模乘运算时延上加速约 3.20 倍,在模幂运算时延上降低约 1.75 倍。

表 3 与 CPU 的时延对比

Table 3 Comparison with CPU latency

平台	频率/GHz	模乘时延/μs	模幂时延/μs
Intel I9-13900HX	4.400	1.673	1 165.91
本文 ZU15EG	0.287	0.523	667.42

而本文设计的加速系统功耗情况如图 9 所示,本文加速系统的功耗约为 Intel I9-13900 基础功耗 55 W 的 12.4%。

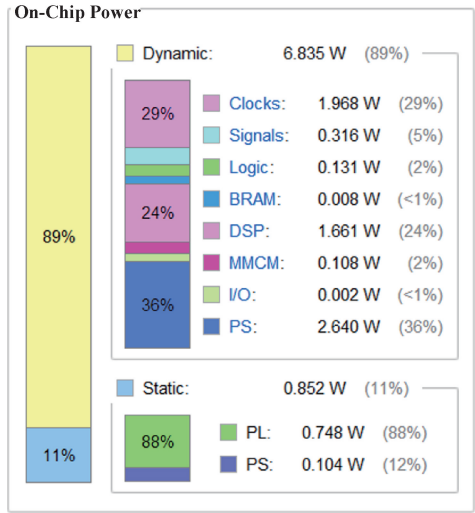

图 9 加速系统功耗图

Fig. 9 Power consumption graph of the acceleration system

表 4 展示了本文加速系统与其他 FPGA 加速器的对比结果。由于不同平台之间存在架构差异,因此本文采用基础逻辑资源块对等转换的方式进行对比。例如,各文献中所使用的不同系列的硬件逻辑资源块统一用含有 8 个 6 输入 LUT、16 个 FF 的 CLB 表示。以反映硬件资源消耗量。无法等效转换的硬件资源则不做转换。而在文献中没有写明的资源用无做标识。文献[10]、文献[11]与文献[12]分别存在平台类型不同;无具体模乘、模幂吞吐量的问题,难以比较,故不列入表 3。表中所有算法指标均在 1 024 位计算场景下,其中模乘与模幂 ATP 指标用以综合考量算法的资源占用量与计算速度,计算方法为 LUT 数量×时间(微秒)×10^{-3}。

文献[13]依托其创新改进的 5-2CSA 加法器降低了关键路径的延迟。并且其仅使用了逻辑资源块作为消耗,不依赖于 DSP 等硬核资源,在不同平台移植上具有极大优势。同时,其专注于优化资源使用量的减少,在 ATP 上同样有优势。但在计算吞吐量上的很难有优势。文献[14]

表 4　加速方案性能对比

Table 4　Performance comparison of acceleration schemes

加速方案	频率/MHz	CLB	DSP	模乘吞吐量(KOPS)	模幂吞吐量(OPS)	Paillier 加密吞吐量(OP/S)	模乘 ATP	模幂 ATP
文献[13]	278	1 027	0	180.8	无	无	45.45	无
文献[14]	无	1 272	0	992.41	484.57	956.69	10.26	21 000.45
文献[15]	211/201	510/1 530	0/56	719.4/无	无/657.9	无/无	5.67/无	18 604.64
文献[16]	386	1 530	45	无	无	680.3	无	无
文献[17]	257	59 098	461	1 863.2	3 593.7	7 029.9	253.88	131 574.37
本文	287	35 478	2 609	1 875.8	1 498.3	3 649.2	151.28	189 430.10

采用了基于快速傅里叶变换(fast Fourier transform, FFT)的麦克劳林模乘无条件选择版蒙哥马利模乘算法,消除了原算法中的所有条件分支选择操作,提高了一定的安全性与效率并搭配双 FFT 乘法器架构实现了有较高灵活性的 1 024～7 680 b 高吞吐量模乘计算。但由于其架构较复杂且资源消耗量少,故而在计算能力上较弱。文献[15]采用高基数蒙哥马利模乘与流水线 Karatsuba 大数乘法器架构,实现了 512～1 024 b 的模乘计算,文献[16]以文献[15]为基础,实现了 512～2 048 b 的模幂计算。两篇文献的共同特点是使用了流水线、并行执行、资源复用与操作交织设计,使得资源使用量与计算吞吐量比值具有一定优势。对比 CPU 在一定程度提高了计算吞吐量,但提升有限仅在平行计算使用时具有优势。文献[17]使用其独特的 Pipe FL 架构、硬件感知蒙哥马利算法与软硬协同计算,实现了极高吞吐量的模乘、模幂与 Paillier 加密计算。但该设计在较低端 FPGA 平台使用时的资源总量、时序约束压力很大。且为掩盖流水线处理延迟需求,对批处理数据量要求也很大,对小批量数据场景加速效果较弱,在文献中的应用场景并未对这一点有苛刻的要求。

总体而言,与文献[13]、文献[14]、文献[15]、文献[17]中的设计相比,本文在模乘吞吐量上分别提升约 937.59%、88.99%、160.92%、0.68%。与文献[14]、文献[15]相比在 1 024 b 模幂吞吐量上分别提升约 209.33%、127.74%。与文献[14]、文献[16]相比在 Paillier 加密计算吞吐量上分别提升约 281.38%、436.34%。与文献[17]相比,本文在模乘吞吐量上仅有微弱优势,但通过比较 ATP 可知,本文的模乘单元设计具备更小的面积时间积,在计算时间和资源消耗上达到了更好的平衡。而模幂吞吐量则约为其 50%,这是由于设计平台资源总量限制因素,无法解决模平方计算中的数据依赖问题。但方案在计算吞吐量与计算时延上对比其他文献仍有一定的优势。这使得加速方案在处理低批量、高吞吐量与快速响应需求时展现出显著的能力。

4　结　　论

本文针对 Paillier 半同态加密算法在复杂模幂运算中计算效率低的问题,提出了一种基于软硬件协同的加速架构。通过分析强时效场景的需求,从算法与硬件两个层面优化计算过程:在算法层,采用蒙哥马利模乘优化技术实现大整数模运算的并行处理与数据复用,降低运算复杂度;在硬件层,设计定制化运算单元,提升模乘与模幂的计算速度。同时,在软件层实现参数预计算与任务解析功能,并通过 AXI 总线完成软硬件间的数据交互与控制。测试结果表明,在 1 024 bit 位宽下,该方案相比传统软件在模乘、模幂上的性能分别提升了 256.4%、74.7%,能够为安全多方计算与联邦学习等应用提供高效的隐私计算支持。

参考文献

[1] PAILLIER P. Public-key cryptosystems based on composite degree residuosity classes[C]. International Conference on the Theory and Applications of Cryptographic Techniques, 1999: 223-238.

[2] SAHINBAS K, CATAK F O. Secure multi-party computation-based privacy-preserving data analysis in healthcare IoT systems [M]. Berlin: Spinger International Publishing, 2023: 57-72.

[3] 郭显,王典冬,冯涛,等.基于同态加密的可验证隐私保护联邦学习方案[J].电子与信息学报,2025,47(4):1113-1125.
GUO X, WANG D D, FENG T, et al. Design of privacy-preserving federated learning scheme based on homomorphic encryption[J]. Journal of Electronics & Information Technology, 2025, 47(4): 1113-1125.

[4] KU H CH, SUSILO W, ZHANG Y D, et al. Privacy-preserving federated learning in medical diagnosis with homomorphic re-encryption [J]. Computer Standards & Interfaces, 2022, 80: 103583.

[5] SHI G, LI Y, WANG X, et al. PHEP: Paillier homomorphic encryption processors for privacy-preserving applications in cloud computing[C]. 2023 IEEE Hot Chips 35 Symposium (HCS). Palo Alto, CA: IEEE, 2023: 1-20.

[6] LIN D, CAO H, TIAN C, et al. The fast Paillier decryption with Montgomery modular multiplication based on OpenMP[C]. 2022 IEEE 13th International Symposium on Parallel Architectures, Algorithms and Programming(PAAP). Beijing: IEEE, 2022: 1-6.

[7] TENG Y, LIU C. Research on GPU-based parallel processing for the Paillier algorithm [C]. 2023 International Conference on Blockchain Technology and Information Security(ICBCTIS),2023: 101-106.

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

[9] KOC C K, ACAR T, KALISKI B S. Analyzing and comparing Montgomery multiplication algorithms[J]. IEEE Micro, 1996, 16(3): 26-33.

[10] CAI C, AWANO H, IKEDA M. High-speed ASIC implementation of Paillier cryptosystem with homomorphism[C]. 2019 IEEE 13th International Conference on ASIC(ASICON), 2019: 1-4.

[11] YANG Z, HU S, CHEN K. FPGA-based hardware accelerator of homomorphic encryption for efficient federated learning[J]. ArXiv preprint arXiv: 2007.10560, 2020.

[12] BAHADORI M, JÄRVINEN K. A programmable SoC-based accelerator for privacy-enhancing technologies and functional encryption [J]. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 2020, 28(10): 2182-2195.

[13] 任仕伟, 王华阳, 郝越, 等. 蒙哥马利模乘算法改进及硬件实现[J]. 北京理工大学学报自然版, 2024, 44(3): 306-311.
REN SH W, WANG H Y, HAO Y, et al. An improved Montgomery modular multiplication algorithm and its hardware implementation [J]. Journal of Beijing Institute of Technology, 2024, 44(3): 306-311.

[14] DAI W, CHEN D, CHEUNG R C C, et al. FFT-based McLaughlin′s montgomery exponentiation without conditional selections[J]. IEEE Transactions on Computers, 2018, 67(9): 1301-1314.

[15] SAN I, AT N. Improving the computational efficiency of modular operations for embedded systems[J]. Journal of Systems Architecture, 2014, 60(5): 440-451.

[16] SAN I, AT N, YAKUT I, et al. Efficient paillier cryptoprocessor for privacy-preserving data mining [J]. Security and Communication Networks, 2016, 9(11): 1535-1546.

[17] WANG Z, CHE B, GUO L, et al. PipeFL: Hardware/software co-design of an FPGA accelerator for federated learning[J]. IEEE Access, 2022, 10: 98649-98661.

作者简介

阮笃钧,硕士,主要研究方向为嵌入式系统开发、算法硬件化等。

E-mail:18275179646@163.com

周骅(通信作者),博士,副教授,主要研究方向为数字系统设计、信息安全机制研究以及物联网嵌入式系统设计。

E-mail:zhouhua97@gmail.com