

基于分段查找表与动态CG-NTT复用的同态ReLU加速

廖颖龙¹, 罗羽乔², 李丹婕¹, 李灵武¹, 姚磊³, 蒋琳⁴, 方俊彬^{1,5}

(1. 暨南大学物理与光电工程学院, 广东 广州 510632; 2. 暨南大学智能科学与工程学院, 广东 珠海 519070;
3. 深圳云安宝科技有限公司, 广东 深圳 518055; 4. 哈尔滨工业大学(深圳) 计算机科学与技术学院, 广东 深圳 518055;
5. 教育部“光电混合集成电路”国际合作联合实验室, 广东 广州 510632)

摘要: 全同态加密不支持非线性同态计算, 导致隐私保护机器学习中的同态ReLU评估难以直接实现。现有采用高阶多项式近似ReLU激活函数的方案设计大位宽模乘与高阶多项式乘法运算, 使得系统吞吐量受限、资源利用率低。对此, 提出了一种CPU-FPGA异构协同的同态ReLU评估方案: 由CPU端执行ReLU的高阶多项式近似, FPGA端执行同态多项式评估加速。具体而言, 首先设计了分段查找表模运算方法, 借助双级分解映射机制将大位宽模乘转化为并行高低位查表与线性组合, 模乘时延较现有最优方案降低97%。其次, 提出动态资源复用CG-NTT架构优化多项式乘法效率, 依托固定访存模式实现动态表复用策略, 相较于传统NTT架构, 减少15%~37%计算单元, 降低45%~50.8%运算时延。最后, 构建CPU-FPGA异构加速框架, 通过相位调优接口与资源协调机制, 使同态ReLU评估获得21.4倍加速比。实验验证了该方案在加速密文推理中的可行性, 但最耗时的自举操作仍需在CPU端执行。因此, 未来研究将聚焦自举操作的硬件加速, 以进一步提升密文推理系统的实用性能。

关键词: 全同态加密; 查找表模运算; CG-NTT; ReLU激活函数; CPU-FPGA异构协同

中图分类号: TP309.2

文献标志码: A

DOI:10.11959/j.issn.2096-109x.AQ25232

Accelerating homomorphic ReLU via segmented lookup table and dynamic CG-NTT reuse

Liao Yinglong¹, Luo Yuqiao², Li Danjie¹, Li Lingwu¹, Yao Lei³, Jiang Lin⁴, Fang Junbin^{1,5}

1. Department of Optoelectronic Engineering, Jinan University, Guangzhou 510632, China
2. School of Intelligent Systems Science and Engineering, Jinan University, Zhuhai 519070, China
3. Shenzhen Yunanbao Technology Co., Ltd., Shenzhen 518055, China
4. School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
5. International Cooperation Joint Laboratory for Optoelectronic Hybrid Integrated Circuits, Guangzhou 510632, China

Abstract: Fully homomorphic encryption does not support nonlinear homomorphic computation, making it difficult to directly implement homomorphic ReLU evaluation in privacy-preserving machine learning. Existing schemes that employ high-degree polynomials to approximate the ReLU activation function involve large-bitwidth modular multiplication and high-degree polynomial multiplication operations, resulting in limited system throughput and low resource utilization. To address this, a CPU-FPGA heterogeneous collaborative architecture was proposed for homomorphic ReLU evaluation, with the CPU executing high-degree polynomial approximation and the

收稿日期: 2025-12-05; 修回日期: 2026-05-03

通信作者: 方俊彬, tjunbinfang@jnu.edu.cn

基金项目: 深圳市科技重大专项 (No.KJZD20240903104301003)

Foundation Item: Shenzhen Major Science and Technology Project (No.KJZD20240903104301003)

FPGA accelerating homomorphic polynomial evaluation. Specifically, a segmented lookup table modular reduction algorithm was first designed. By means of a two-level decomposition mapping mechanism, large-bitwidth modular multiplication was converted into parallel high/low-bit lookup table operations and linear combinations. As a result, the modular multiplication delay was reduced by 97% compared to the existing state-of-the-art scheme. Secondly, a dynamic resource-reusing CG-NTT architecture was proposed to optimize polynomial multiplication efficiency. Relying on a fixed memory access pattern, a dynamic table reuse strategy was implemented. Compared with traditional NTT architectures, 15%~37% of the computation units were reduced, and 45%~50.8% of the operation latency was decreased. Finally, a CPU-FPGA heterogeneous acceleration framework was constructed. Through a phase-tuning interface and a resource coordination mechanism, a speedup of 21.4 times was achieved for homomorphic ReLU evaluation. The feasibility of the proposed scheme in accelerating ciphertext inference was verified by experiments, although the most time-consuming bootstrapping operation still had to be executed on the CPU side. Therefore, future research will be focused on the hardware acceleration of the bootstrapping operation, in order to further improve the practical performance of the ciphertext inference system.

Key words: fully homomorphic encryption, lookup table modular reduction, CG-NTT, ReLU activation function, CPU-FPGA heterogeneous collaborative

0 引言

近年来,机器学习逐步落地产业化应用,覆盖医疗、金融、交通等多个领域。然而,机器学习的落地需要依托海量敏感数据,容易引发隐私泄露与信息安全风险。为了保护数据安全,业界陆续引入全同态加密(fully homomorphic encryption, FHE)、安全多方计算和差分隐私等隐私保护技术。其中, FHE 技术允许在加密数据上直接进行计算而无须解密,具有安全性高、交互开销低等优点,是现阶段发展前景突出的隐私计算技术之一。自 1978 年 Rivest、Adleman 和 Dertouzos 首次提出 FHE 构想^[1],到 2009 年 Gentry 基于理想格构造出首个可实现的 FHE 方案^[2], FHE 技术历经多轮迭代演进。BGV (brakerski-gentry-vaikuntanathan)^[3]、FV (fan-vercauteren)^[4]等第二代 FHE 方案借助模切换技术提升运算效率;以 CGGI (chillotti-gama-georgieva-izabachène)^[5]和 CKKS^[6-8] (cheon-kim-kim-song) 为代表的第三代全同态方案则通过近似计算和自举算法优化进一步降低计算开销。其中, CKKS 方案凭借高效的浮点数同态运算能力,已成为隐私保护机器学习的主流选择。

然而, CKKS 密文推理系统的实际部署受限与两大瓶颈。其一,环上高维多项式乘法与大位宽模乘运算会引发超线性计算时延。其二,整流线性单位(rectified linear unit, ReLU)函数这类

激活函数需要通过高阶多项式近似实现,其动态输入范围缩放与不规则内存访问模式,造成硬件资源利用率低下。为突破上述性能瓶颈,国内外研究者相继提出面向中央处理器(central processing unit, CPU)、图形处理器(graphics processing unit, GPU)、现场可编程门阵列(field programmable gate array, FPGA) 3 类硬件平台的加速优化方案^[9-20]。然而,现阶段面向 FPGA、适配 CKKS 密文推理系统的落地加速实现,大多仅集中在加速大位宽模乘运算和高维多项式乘法上^[19-21],忽视了 ReLU 运算过程中的数论变换(number theoretic transform, NTT)算子密集调用、模乘位宽失配等核心问题,严重制约了系统效率提升。

基于 CKKS 的密文 ReLU 方案在 FPGA 上的实现仍受片上资源效率与吞吐量的约束,其原因主要包括两点。一是, CKKS 的核心算子 NTT 需要动态生成地址,导致内存访问模式不规则^[19-21],加剧了片上存储与计算资源的竞争。二是, CKKS 中大量的模乘运算需处理大位宽模数,而现有的模乘器受限于逐位进位链,难以满足高吞吐需求^[22-24]。在单一硬件平台上实现同态 ReLU 评估时,这些性能瓶颈更为突出。纯 CPU 系统虽能灵活处理 ReLU 多项式阶数动态调整与输入归一化参数 v 的实时更新,但其串行架构难以高效执行 NTT 算子的高度并行化乘加操作,致使 ReLU 计算吞吐量受限。FPGA 虽可通过流

流水线加速 NTT 核心运算，却因硬件逻辑固化，无法适配 ReLU 计算中由 v 变化引发的访存模式随机跳变（如不同缩放因子下多项式系数的位宽变化）。

尽管已有研究在模运算优化^[22-24]、NTT 架构加速^[25-28]及 FHE 硬件加速^[35-36]等方面取得了显著进展，但这些工作均未系统性地解决 CKKS 密文 ReLU 评估中的模乘位宽失配、NTT 访存冗余与跨平台协同低效等问题。具体而言，现有查找表取模电路^[30]受限于存储爆炸与加法链过长，无法适应全同态加密的大模数约简；NTT 架构^[28]缺乏对密文层级动态性的资源适配机制。此外，现有的 FPGA 加速器^[35-36]都集中于全同态加密的单一算子加速。

针对上述挑战，本文在算法层面提出一种面向同态 ReLU 评估的分段查找表（segmented lookup table, S-LUT）模约减与动态几何固定数论变换（constant-geometry NTT, CG-NTT）资源复用方法。通过双级分解映射与子表激活策略，该方法解决了大位宽模乘存储爆炸与 NTT 访存冗余问题。基于上述算法，本文进一步实现了 CPU-FPGA 异构高效 ReLU 计算系统，通过并行查表架构、预取一计算协同流水线及相位调谐接口完成算法的硬件加速与系统协同，克服了 ReLU 非线性计算低效等核心问题。本文的主要贡献如下。

1) 设计分段查找表取模电路，采用双级分解映射机制将大位宽模乘运算转换为高位/低位并行查表与线性组合操作。这一设计使查找表所需的存储资源从 2^n 降至 $2^{n/2}$ ，线性组合的加法链长度缩短 50%，模乘运算时延较现有代表性方案最高降低 97%。

2) 提出动态资源复用 CG-NTT 架构，在固定访存模式的基础上，采用动态表复用策略降低内存地址计算次数，并根据密文层级激活对应的取模子表集合 $\{q_{j1}, q_{j2}, \dots\}$ 。通过部分重配置技术，将未激活的查找表临时分配给旋转因子存储单元，消除冗余表项占用。相比于原有的 NTT 架构，该设计减少了 15%~37% 的计算单元，从而降低了 45%~50.8% 的运算时延。

3) 构建基于 CPU-FPGA 异构协同的动态负

载均衡框架，利用相位调谐接口与资源协同机制提升数据传输与计算效率。在此基础上，在异构框架上实现密文 ReLU 计算 21.4 倍的加速比，并集成至现有密文推理系统，实现端到端性能提升。

1 预备知识

1.1 查找表

查找表（lookup table, LUT）是 FPGA 中基于预计算的核心组件，其本质是通过地址映射直接输出预存结果，以空间换时间来加速复杂运算。数学上，LUT 可视为离散函数 $f(x) = y$ 的硬件实现，输入 x 作为地址索引，输出 y 存储于对应位置。例如，在密码学中，NTT 运算的旋转因子可预计算存入 LUT，并通过索引快速调用。FPGA 的物理 LUT 资源（如 Xilinx UltraScale+ 架构的 6 输入 LUT）本质上是可配置逻辑单元，支持灵活重组为分布式随机存取存储器（random access memory, RAM），实现低时延的并行数据访问。与块随机存取存储器（block random access memory, BRAM）相比，LUT 构建的分布式 RAM 容量较小，但时延更低，适合高频访问场景；而 BRAM 容量大但时延较高，二者结合使用可以平衡效率与资源开销。

在大模数同态加密场景中，LUT 需存储的模数 Q 位宽可达 32~64 位，直接构建全量表（规模达 MB 至 GB 级）远超 FPGA 片上资源限制。为此，采用分块策略：将 Q 分解为多个小模数，通过中国剩余定理在查表后重组结果，以分块存储降低单表容量，但需额外模乘计算。进一步优化中，混合存储架构将高频数据映射至 LUT 分布式 RAM，低频数据存于 BRAM，结合算法特性压缩冗余预计算值，从而在存储效率与计算复杂度间实现帕累托最优。这种设计使得 FPGA 能在有限资源下高效支持大规模密码学运算，凸显其在实时性与能效上的优势。

1.2 几何固定快速数论变换(CG-NTT)

常用的 NTT 实现框架主要包括变几何 Cooley-Tukey NTT (CT-NTT) 和 CG-NTT。其中 CG-NTT 算法如算法 1 所示。CG-NTT 因其内存访问模式的一致性与数据流的可预测性，相较

于CT-NTT架构,在FPGA、专用集成电路(application specific integrated circuit, ASIC)等硬件上具有静态地址映射、参数无关设计等优势。

算法1 CG-NTT算法

输入 多项式系数 $a[0, \dots, n-1] \bmod q$ 、模数 q 、原根 $\psi[0, \dots, \frac{n}{2}-1]$ 和多项式阶数 $n = 2^m$

输出 CG-NTT 结果 $A[0, \dots, n-1] \bmod q$

- 1) for $s = 1:m$ do
- 2) $h = 2^{s-1}$
- 3) $\text{num}_{\text{groups}} = 2^{m-s}$
- 4) for $\text{group}_{\text{idx}} = 0:\text{num}_{\text{groups}} - 1$ do
- 5) $\text{base} = \text{group}_{\text{idx}} \times 2^s$
- 6) for $j = 0:h - 1$ do
- 7) $k = \text{base} + j$
- 8) $l = k + h$
- 9) $\omega = \psi[j \times \text{num}_{\text{groups}}]$
- 10) $t = (a[l] \times \omega) \bmod q$
- 11) $A[l] = (a[k] - t) \bmod q$
- 12) $A[k] = (a[k] + t) \bmod q$
- 13) end for
- 14) end for
- 15) end for

目前,已有许多文献通过实验验证了CG-NTT在吞吐量、能效比与资源利用率上均具有显著优势^[25-27]。

然而,CG-NTT的关键计算瓶颈在于其蝶形运算阶段的高计算时延。该阶段需对每一组系数执行模乘累加操作,运算表达式如下:

$$\begin{cases} t \leftarrow (a[l] \times \omega) \bmod q \\ A[l] \leftarrow (a[k] - t) \bmod q \\ A[k] \leftarrow (a[k] + t) \bmod q \end{cases} \quad (1)$$

实验数据显示,此类操作占CG-NTT总运算时间的90%以上,且其串行依赖性与密集模运算特性导致关键路径时延显著增加。尽管CG-NTT在吞吐量上表现优异,但高时延问题仍严重制约其在低时延实时任务中的部署效率,成为实际应用中亟待突破的关键挑战。

1.3 ReLU激活函数多项式近似

在基于CKKS的全同态加密推理系统中,ReLU激活函数的高效实现需解决其非线性运算

特性与同态运算约束之间的冲突。ReLU依靠阈值判断实现非线性映射(输入大于0时原样输出,小于或等于0时输出0),但在同态加密下直接比较密文大小会导致密文噪声急剧膨胀,破坏解密可行性。为此,主流方案采用低阶多项式拟合ReLU的输入输出映射关系,例如利用奇次多项式拟合正区间的线性特性,同时在负区间强制输出接近0的值。这种近似需平衡多项式阶数、逼近精度和同态乘法深度:高阶多项式虽能提升精度,但会导致计算复杂度呈指数级上升,而低阶多项式可能因拟合不足引入显著误差,尤其在0点附近。

硬件实现的关键挑战在于动态输入范围与固定多项式系数之间的矛盾。密文数据的动态范围要求多项式系数随缩放因子进行动态调整,频繁的参数加载操作会挤占片上带宽;受限片上存储容量,多余系数需要依赖高时延的片外存储器,由此带来高访存时延,形成系统带宽瓶颈。此外,多项式运算的数据串行依赖与同态乘法深度紧密相关,例如7阶多项式需至少3级乘法深度,导致硬件流水线难以充分展开。因此,ReLU优化的核心目标是在加密域内以最小计算代价逼近ReLU,同时满足噪声增长约束与硬件资源限制。

1.4 剩余数系统

剩余数系统(residue number system, RNS)通过中国剩余定理,将大模数 Q 分解为 K 个NTT友好模数 $\{Q_1, \dots, Q_k\}$ 的乘积,从而构建同构映射。

对于FPGA、ASIC等硬件加速平台而言,分解后的NTT友好模数 $\{Q_1, \dots, Q_k\}$ 可以通过流水线单元等并行技术加速,采用“空间换时间”策略实现计算吞吐量的线性扩展。本文采用RNS架构,通过构建 K 个小查找表 $\text{LUT}(Q_1), \dots, \text{LUT}(Q_k)$ 分解大查找表 $\text{LUT}(Q)$,并采用分段查找表技术对每个小查找表进行双级分解映射,从而最小化NTT运算所需要的硬件资源。在计算复杂度层面,小模数运算不仅规避了整数乘法器的高资源消耗特性,其同构结构还可将理论计算开销从 $O(M \log(N) \log(Q))$ 降至 $O(M \log(N) \log(Q_i))$ 的并行累加模式。

1.5 相关工作

为明确本文的技术贡献，本小节从问题场景、核心方法与优化目标3个维度，系统对比本文与已有前沿工作^[23-24,28,30,35-36,39]的差异。

现有取模电路主要包括迭代取模^[23-24]和查找表取模^[30,39]。前者需要多步执行，且受限于进位链时延；后者的存储开销随模数位宽呈指数级膨胀。本文从算法层面通过双级分解映射降低存储复杂度，并通过动态区间合并进一步压缩所需的存储空间，以实现低时延、低存储开销的取模运算。

现有工作验证了CG-NTT在静态访存与参数适配方面的优势^[25-28]，但未针对底层的模数链进行动态优化。为此，本文提出动态表复用策略，利用CG-NTT静态访存与RNS同构特性，仅激活当前需要的取模子表；并在具体实现时通过预取—计算流水线隐藏重配置开销，实现资源与时延的同步降低。

基于FPGA的加速平台工作^[35-36]聚焦在自举加速，尚未实现针对同态ReLU的加速。本文提出了基于CPU-FPGA的同态ReLU加速框架：CPU负责多项式近似与动态参数，FPGA负责同态评估，首次实现了高效的异构ReLU同态运算。

2 基于异构架构的同态ReLU加速

在基于CKKS^[6]的全同态加密推理系统中，

ReLU是机器学习模型中常用的非线性激活函数，其隐私保护需求推动了全同态加密实现方案的快速发展。由于全同态加密机制无法直接对非线性函数进行密文评估，非线性算子的同态计算通常分为两个阶段：非线性函数多项式近似和同态近似线性函数评估。本文提出将近似高阶多项式函数生成步骤部署于CPU端完成，同时将同态近似线性函数评估（即同态多项式评估）卸载至FPGA端并加速执行，并针对性设计CPU-FPGA异构传输接口，同态ReLU近似评估框架如图1所示。

针对同态ReLU近似评估的核心挑战——非线性特性需要通过高阶多项式近似实现，而动态输入范围缩放与密文噪声敏感性问题导致硬件加速面临“精度”“时延”“资源”三重矛盾——本文提出一种面向密文ReLU的异构协同加速架构。该架构以FPGA为核心构建ReLU专用计算引擎。首先，针对ReLU多项式近似中密集的大位宽模乘运算及其核心算子NTT因模乘运算复杂、系数存储爆炸导致的效率瓶颈，本文创新性地提出双级分解映射机制，将大位宽模运算拆解为高/低位并行查表与线性组合，结合剩余数系统的动态表复用策略，使NTT时延较原架构降低50%。该设计首次突破模运算资源效率的“超线性诅咒”，为大规模同态加密提供硬件级优化范式。在此基础上，针对密文推理中非线性函数的高噪声敏感性与计算不可逆性，本文提出奇对

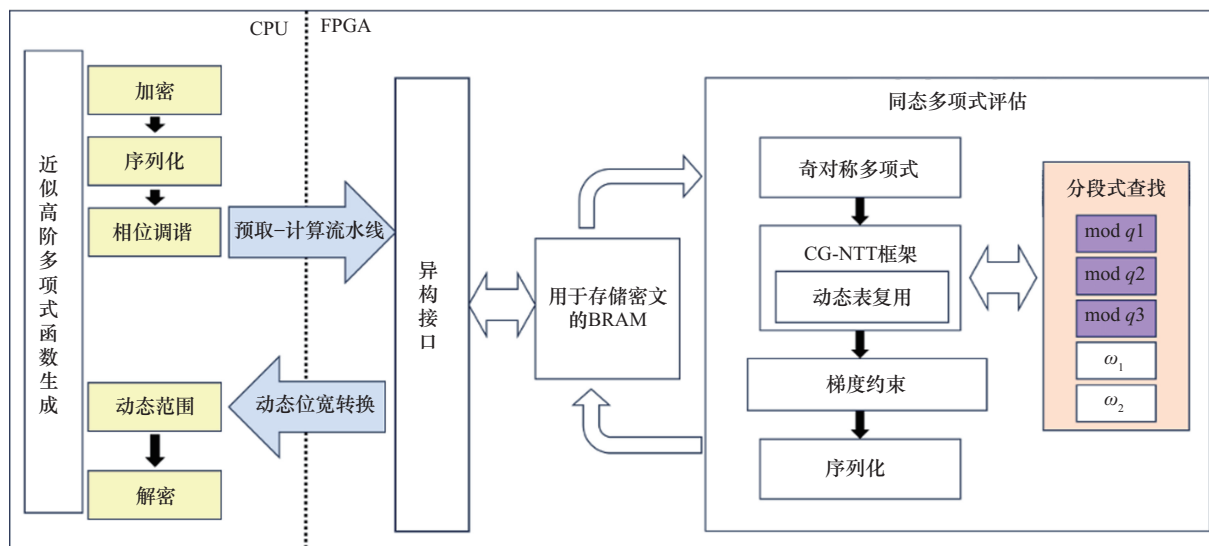


图1 同态ReLU近似评估框架

称约束多项式与零点邻域微分固化技术, 通过动态区间缩放与 FPGA 端深度流水线架构, 实现 ReLU 激活的 21.4 倍加速比 (相较于纯 CPU 方案)。最后, 针对密文数据洪流引发的跨平台传输阻塞问题, 本文创新性地设计了相位调谐接口与预取-计算协同流水线, 通过时间片轮转调度将密文传输时延完全覆盖于有效计算周期内。

实验表明, 本方案为隐私保护神经网络中密文 ReLU 实用化部署提供了首套兼顾效率与精度的端到端硬件解决方案。

2.1 查找表模运算的优化设计

本文提出基于查找表的模运算 (look up table-based modular arithmetic, LUT-MA), 通过预计算与查表机制替代传统复杂运算, 以突破硬件瓶颈。具体而言, 针对已知模数 q , 将二进制整数 $A = \sum_{i=0}^n A_i 2^i$ 的模运算分解为位级累加操作, 即通过计算 $A \bmod q = \left(\sum_{i=0}^n (A_i 2^i \bmod q) \right) \bmod q$ 得到 $2^i \bmod q$, 再通过预计算的值构建静态查找表, 从而将模运算简化为查表与累加过程。相较于依赖乘法器与复杂进位链的传统模乘方案, LUT-MA 仅需加法器链即可完成模约简, 在 FPGA 中实现零乘法器占用。

然而, 经典 LUT-MA 方案面临存储与时延的双重挑战: 查找表规模随模数位宽 n 呈指数级增长 $O(2^n)$, 例如, 当 $n = 32$ 位时, 表容量可达 4 GB, 远超 FPGA 片上存储容量; 同时, 位分解导致的加法链长度与 n 呈线性关系, 即 $O(n)$, 严重制约时序性能。上述局限性凸显了

传统方案在资源效率与实时性之间的失衡。为此, 本文进一步提出分段查找表架构与动态权重修正策略, 以实现指数级压缩存储开销并显著降低运算时延。

2.1.1 分段查找表架构设计

S-LUT 架构如图 2 所示。

将 n 位模数分解为低位宽 $n_1 = n/2$ 与高位宽 $n_2 = n/2$, 并分别构建低位子表 LUT_L 与高位子表 LUT_H , 使 $A \bmod q$ 满足式(2):

$$A \bmod q = \left(\sum_{i=0}^{n_1-1} (A_i \cdot 2^i \bmod q) + 2^{n_1} \sum_{j=0}^{n_2-1} (A_j 2^j \bmod q) \right) \bmod q \quad (2)$$

其中, i 为低位部分的位索引, j 为高位部分的局部位索引, A_i 表示低位部分第 i 位的二进制值, A_j 表示高位部分重新编号后第 j 位的二进制值。对于 n 位模数 q , 分段查找表的存储复杂度 C_{storage} 可表示为:

$$C_{\text{storage}} = 2 \cdot 2^{n/2} \log q \quad (3)$$

每个子表 LUT_H/LUT_L 存储 $2^{n/2}$ 个条目, 每个条目占用 $\log q$ 位。由于每个子表的模加运算可以并行执行, 因此加法链长度缩短为 $O(n/2)$, 进而时序路径时延降低 50%。

2.1.2 低位累加器的硬件简化

对于 LUT_L , 其计算过程可重构为:

$$\sum_{i=0}^{n_1-1} (A_i 2^i \bmod q) = \sum_{i=0}^{n_1-1} A_i 2^i - q \frac{\sum_{i=0}^{n_1-1} A_i 2^i}{q} \quad (4)$$

式(4)表明, 低位累加可通过移位-减法器

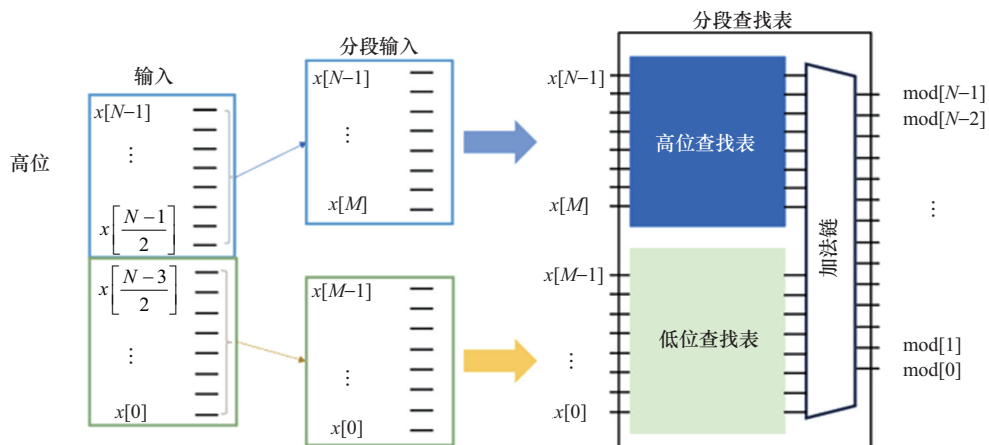


图2 S-LUT架构

链（结合移位寄存器与减法器的级联架构）直接实现，无须预存查找表，如图3所示。具体步骤为：输入比特流 A_i 首先进入移位累加器，根据位权 2^i 生成中间值 $T = \sum_{i=0}^{n_1-1} A_i 2^i$ ；随后，减法器通过并行比较器阵列实时计算 $\lfloor T/q \rfloor$ 值，并输出 $T' = T - q \cdot \lfloor T/q \rfloor$ 作为最终模运算结果。此架构的核心优势在于实现了“存储—计算置换”，相较于传统方案需预存 2^{n-1} 个 $\log q$ 位表项，本设计仅需配置可重构移位寄存器与组合逻辑电路，硬件资源消耗与位宽 n_1 呈线性关系而非指数级增长。

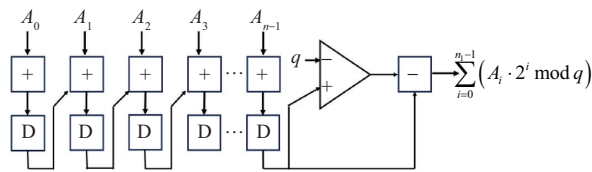


图3 移位-减法器链示意

在典型FHE参数（ $n_1 = 16, q = 32$ ）下，该方案可节省存储空间 $2^{n_1} \cdot \lfloor \log q \rfloor$ 。以 $n_1 = 16$ 为例，传统 LUT_L 需要 $2^{16} \times 32 = 2\,097\,152$ bit的存储单元，而优化后可以减少 $2^{16} \times \log_2 32 = 327\,680$ bit。

2.1.3 高位子表的基 $r = 2^k$ 并行查表优化

传统基于比特串行的查表机制在处理 n_2 位输入时需进行 n_2 次顺序访存，其关键路径时延与 n_2 呈线性正相关，这是NTT硬件性能发展的瓶颈。为此，本文借鉴巴雷特约简算法的基 r 分解思想^[28]，提出多粒度并行查表架构。

该架构通过设定并行粒度参数 k ，将输入值 T 按 k 比特位宽划分为 $\lceil n_2/k \rceil$ 个数据段，每段权重预处理为 $2^{jk} \bmod q$ 。 T 可表示为：

$$T = \sum_{j=0}^{n_2-1} A_j 2^j \quad (5)$$

此分解策略使得每周可并行处理 k 比特数据，相较于传统串行架构将访存次数压缩至 $\lceil n_2/k \rceil$ 个，理论时延降低为原方案的 $1/k$ 。

随后，针对每段值 $T_{\lfloor jk/k + k - 1:j/k \rfloor} \in [0, 2k - 1]$ ，预计算其模 q 贡献值：

$$A_j = \left(T_{\lfloor jk/k + k - 1:j/k \rfloor} \cdot 2^{jk} \bmod q \right) \quad (6)$$

为进一步优化存储效率，本文提出动态区间

合并策略：若连续 m 个数据段的贡献值均满足 $\Delta_j = \Delta_{j+1} = \dots = \Delta_{j+m}$ ，则将其合并为单一表项并标记区间 $[j, j+m]$ ，同时修正合并后的贡献值 Δ_{jj+m} 。

$$\Delta_{jj+m} = \Delta_j \sum_{t=0}^m 2^{(j+t)k} \bmod q \quad (7)$$

该合并操作由有限状态机仅在预计算阶段自动生成，无须额外硬件开销。改进架构采用两级流水线设计：第1级流水线并行读取 k 位数据段并匹配压缩查找表项，第2级流水线基于修正权重执行模加运算。

2.1.4 动态权重修正电路

基 r 分段的权值计算面临指数项 $2^{jk} \bmod q$ 随 j 动态变化的挑战，为此，设计基于预计算与循环移位协同的权重生成架构。该架构的核心在于预计算—增量更新机制，该机制首先离线计算初始权值表 $W_j = 2^{jk} \bmod q$ 并固化于FPGA片上只读存储器。在运行时，权值更新通过硬件循环左移实现，即：

$$W_{j+1} = (W_j 2^k) \bmod q \quad (8)$$

此递推关系源于模运算的乘法同态性，使得每次权值更新仅需一次模乘操作。在Xilinx系列FPGA中，该操作可映射至数字信号处理器（digital signal processor, DSP）模块，其内置的 25×18 位乘法器与48位累加器链可支持单周期完成模乘运算。

为降低存储开销，设计采用权值压缩策略：当检测到 $W_j 2^k < q$ 时，直接通过算术左移 k 位生成 W_{j+1} ，避免冗余的模运算；反之，则触发DSP的模减单元计算 $W_{j+1} = (W_j \ll k) - q$ 。该策略通过动态检测条件选择计算路径，理论上可显著减少预计算存储开销，同时结合硬件流水线优化设计，能够在低时延约束下完成权值更新。这一方法通过硬件逻辑与算法特性的协同，为大规模NTT计算提供了高效的动态权值生成解决方案，避免了全量存储方案的高资源消耗问题。

2.2 基于查找表模运算的CG-NTT架构

同态加密的核心运算（如NTT、外部积）中，模运算占据总计算时间的70%以上。以典型蝶形运算步骤为例，其表达式为：

$$A[i] = a[2i] + \omega a[2i + 1] \quad (9)$$

该过程涉及 1 次模加法与 1 次模乘法，其中模乘的硬件实现效率直接决定整体性能。传统优化方案聚焦于设计低时延模乘法器，但其依赖多级流水线与复杂进位链，导致硬件资源开销激增，难以满足同态加密的实时性需求。

本文创新性地采用查找表模运算重构 CG-NTT 计算流程。尽管 CG-NTT 的非原地特性要求存储空间翻倍，即存储输入与输出多项式各一份，但其静态访存模式与剩余数系统分解技术的结合，为查找表模运算的硬件资源优化提供了独特机遇。本节详述如何通过模数分解与表复用策略，规避传统大模数查找表模运算应用于 CG-NTT 带来的存储爆炸问题。

2.2.1 模数分解与子表优化

对于在 FHE 同态计算中常见的复合大模数 $Q = q_1 q_2 \cdots q_k$ ，为了提高并行效率，常用 RNS 技术将其分解为 k 个互质小模数 $\{q_i\}$ 。因此，可以基于分段查找表架构，为每个 q_i 独立构建优化子表。每个子模数对应的查找表，可独立构建且对存储规模有一定的限制，即采用基 $r = 2^4$ 分段策略将每个子表存储空间压缩到 $2^{n_i/2}$ 。因此，该方法的总存储空间仅为各子表压缩后的累加和：

$$\sum_{i=1}^k 2^{n_i/2} \quad (10)$$

与传统存储大模数 Q 的查找表所需的存储空间 2^n 相比，该方法能够显著减少所需存储空间。

此外，所有子表都可以通过统一的静态访存模式实现跨模数一致性控制。这是因为，在预计算过程中可以采用固定地址步长存储查找子表，并在使用时指定并行度参数，以此决定所读取的子表。因此，与单个大模数查找表相比，该方法并不需要加入额外的控制模块，即无须为不同小模数 q_i 定制控制逻辑。

2.2.2 动态表复用机制

在 CKKS 方案中，不同等级的密文所采用的模数各不相同，因此预计算中存储的子表并不一定在当前实际运算中被使用。为此，本文结合 CG-NTT 的静态访存特性，实现了支持不同等级的密文按需加载子表的功能，从而实现查找表复用。具体而言，对于特定等级的密文 CG-NTT 运算，仅需激活其 RNS 分解对应的子表集合：

$$\{q_{j1}, q_{j2}, \cdots, q_{jm}\} \subseteq \{q_1, q_2, \cdots, q_k\} \quad (11)$$

由于只有等级相等的两个密文（即两个密文的模数相同）才能进行运算，因此这两个密文的运算仅需激活一组子表集合。

针对未激活的余数系统子表资源，本文提出基于 FPGA 部分重配置技术的动态存储优化方法。在密文计算预处理阶段，系统根据当前运算需求选取并激活对应的子表集合，而未被调用的子表所占用的 BRAM 区块，通过动态重映射机制临时分配给旋转因子存储单元。由于 CG-NTT 所需的旋转因子可通过纯明文计算实时生成（如基于预存原根动态展开），其生成过程可嵌入流水线空闲周期，从而避免因动态配置引入的额外时延。同时，旋转因子仅在激活时占用重映射的 BRAM 空间，运算完成后立即释放，不影响下一次密文运算前模数子表生成。

本文采用基于 BRAM 区块的逻辑粒度进行部分重配置。将每个对应一个小模数的一组 BRAM 及其专用的查找表电路定义为一个可重配置分区，其大小一般为 2~4 个 36 kB 的 BRAM 块。在动态负载场景下，部分重配置的切换开销被严格约束于两级流水线间隙内完成。通过子表预取策略，重配置过程对系统吞吐量的影响可忽略不计。重配置时间取决于待切换子表的规模，但由于仅涉及 BRAM 区块元数据重置，实际时延低于传统部分重配置方案。这种轻量级的重配置开销，可以与 CG-NTT 流水线的固有气泡周期相匹配，从而保证动态资源复用不会打断计算主流线。结合第 2.2.3 小节的预取—计算协同流水线，以 1 024 位、4 个 PE 计算核心的 CG-NTT 为例，说明为何没有因动态配置引入额外时延。

2.2.3 预取—计算协同流水线

考虑到 FPGA 在 CG-NTT 中每次切换所需的取模子表会存在一定时延，为避免取表引入的时延，本文采用动态数据预载策略，设计了预取—计算协同流水线：在 CG-NTT 蝶形运算的间隙期，利用直接内存访问（direct memory access, DMA）控制器，提前将下一运算阶段所需的参数子表从片外存储迁移至片上高速缓存。这一设计充分利用了可编程逻辑器件的流水线特性，在

参数预载过程中同步执行当前运算任务，通过硬件资源的时间复用机制，将数据搬运开销完全覆盖在有效计算周期内。

以1 024位、4个PE计算核心的CG-NTT为例，具体说明如何借助该预取—计算协同流水线，利用运算间隙实现动态表的高效复用。在该参数配置下，NTT理论上需要1 280个周期完成计算^[38]。然而，在实际执行时，每一级运算都要等待前一级完成一组蝶形对的计算，加上不同PE核心之间的存储访问冲突，整条流水线会不可避免地出现空隙，最终总周期数比理想情况多出约30%，即约1 664个周期。本文正是将这些空隙周期充分利用起来，完成动态表复用和旋转因子的计算工作。具体流程如下：首先，在上一轮密文计算完成并开始序列化后，DMA控制器将下一轮NTT的密文参数传入高速缓存，系统依据该密文等级重新生成对应的模运算子表，并同时初始化旋转因子 ω 。以1 024位为例，此步骤中仅初始化 $\{\omega^0, \omega^1, \omega^2, \dots, \omega^{256}, \omega^{512}\}$ 。随后，在NTT不同阶段的空隙中，利用原根的递归特性生成旋转因子，每个旋转因子仅需一次简单乘法即可完成^[39]。

2.3 面向FPGA的ReLU激活函数多项式近似加速方案

ReLU激活函数的硬件加速面临两大核心挑战：其一，输入密文数据的超维特性（如高维张量）导致传统硬件资源分配策略失配，引发内存访问模式碎片化与计算单元利用率低下；其二，激活函数计算路径的动态性（如输入范围自适应缩放与分段近似策略）与固定硬件传输架构间存在时序冲突，造成流水线阻塞。本节提出一种动态可重构的硬件加速架构，基于符号函数解耦与零点微分约束，消除数据语义与硬件逻辑之间的鸿沟，实现ReLU计算流的高效硬件映射；通过CPU-FPGA的资源弹性分配机制与数据通路优化协同创新，最终构建计算与数据传输深度耦合的异步流水线。

2.3.1 基于符号函数解耦与微分约束的ReLU复合多项式近似优化

本文提出一种面向FPGA的ReLU激活函数多项式近似加速方案，以攻克传统ReLU近似方

案在FPGA部署中存在的近似精度和硬件资源之间不可调和的矛盾。

针对FPGA的定点数计算特性，本文将ReLU函数表示为 $y = x$ 与 $\text{sgn}(x)$ 的复合架构：

$$\text{ReLU}(x) = x(0.5 + 0.5\text{sgn}(x)) \quad (12)$$

该架构将原函数的非线性特征显式解耦为 $\text{sgn}(x)$ 与线性运算的复合操作，以便于后续多项式近似模块的硬件映射。

由于 $\text{sgn}(x)$ 存在阶跃奇异性，本文采用分段加权最小二乘法与Remez算法计算得到ReLU函数的近似多项式 $p(x)$ ，通过代入复合架构，导出乘法器-加法器友好型表达式：

$$\text{ReLU}(x) = x(0.5 + 0.5p(x)) \quad (13)$$

神经网络中每个ReLU输入范围的动态变化，都会导致硬件映射失配。因此，本文通过动态线性缩放将任意ReLU输入驯服至归一化区间 $[-1, 1]$ ，建立参数自适应的统一近似基核。实验证明，这一方法可显著降低片上BRAM资源占用，且无精度损失。

为了突破高阶多项式计算瓶颈，本文基于Minimax准则的复合多项式近似方法，结合FPGA流水线设计思想，将整体多项式分解为级联结构：

$$p = p_k \cdots p_1 \quad (14)$$

其中，每个子多项式 p_i 对应独立的算术逻辑单元，可通过深度流水实现低时延计算。

符号函数在零点不连续，造成Remez算法发散，本文采用区间裁剪策略：在近似区间 $[-1, -\varepsilon] \cup [\varepsilon, 1]$ 上， $p(x)$ 和 $\text{sgn}(x)$ 的差值始终小于 $2^{-\alpha}$ ，其中 ε 是一个接近0的小正数， α 表示预设的定点数小数位宽，即小数部分保留的二进制位数。该截断处理可有效规避FPGA定点数在0点的振荡现象，同时减少逻辑资源消耗。

基于 $\text{sgn}(x)$ 的奇对称性约束，本文严格采用奇数阶多项式进行保奇性逼近：

$$p(x) = -p(x), \forall x \in [-1, 1] \quad (15)$$

然而，目标多项式的连续可微特性与 $\text{sgn}(x)$ 在 $x = 0$ 处的阶跃不连续存在本质冲突，导致近似函数在0点邻域出现超调振荡，即近似多项式 $p(x)$ 在0点附近从-1急速上升到1。该现象如图4所示，可表示为：

$$\lim_{\epsilon \rightarrow 0^+} |p(\epsilon) - \text{sgn}(\epsilon)| = 1 \quad (16)$$

其中, ϵ 表示从 0 的右侧趋近于 0 的变量; 该式表明零点右邻域的逼近误差不会随 ϵ 趋近于 0 而消失。

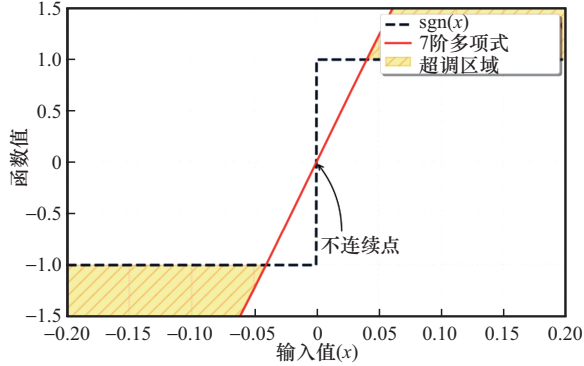


图4 针对ReLU多项式逼近在0点邻域的超调振荡特性

为此, 截断区间约束多项式逼近范围至 $[-1, -\epsilon] \cup [\epsilon, 1]$, 其中, ϵ 基于硬件友好型位宽 α , 可经验性设置为 $2^{-\alpha}$ 。在缩窄区间 $[-v, v]$ 内, 为了确保多项式能快速从接近 -1 的值过渡到接近 1 的值, $p(x)$ 需要满足下列条件:

$$p(v) \approx 1, p(-v) \approx -1 \quad (17)$$

根据 $p(x)$ 在 v 点的展开式, 代入边界条件可以得到式(18)。

$$a_1 \geq \frac{1}{v} (1 + Cv^3) \quad (18)$$

其中, a_1 是多项式 $p(x)$ 中 x 的系数, 即多项式在 0 点处的导数值; C 为多项式高阶系数。

为了强化收敛速度, 本文采用更严格的下界。当逼近 $|x| \leq v$ ($v \in (0, 1)$) 时, $p(x)$ 在零点处的导数值应当满足:

$$\left. \frac{dp}{dx} \right|_{x=0} = a_1 \geq \frac{2}{v} (1 + \mathcal{O}(v^3)) \quad (19)$$

该微分约束迫使近似多项式在缩窄区间 $[-v, v]$ 内加速收敛: 在 $x \in [0, v]$ 时, $p(x)$ 的上升速率与 $1/v$ 呈正相关。因此, 本文将近似区间从初始的 $[-1, 1]$ 进一步缩小为 $[-v, v]$ ($v < 1$)。随着区间变小, p_1 在 0 点的正邻域迅速趋于值 1, 从而使 ReLU 在 0 点附近的近似误差显著降低, 如图 5 所示。经优化后的区间缩放策略确保:

$$|p(x) - \text{sgn}(x)| \leq 2^{-\beta} \quad (20)$$

其中, β 是用于控制误差的超参数。

为了实现奇数阶多项式硬件资源压缩, 本文

进一步引入零点邻域微分约束硬件化机制, 即在 FPGA 端集成区间缩放因子 v 的计算模块, 使得 $\left. \frac{dP}{dx} \right|_{x=0} \geq 2/v$ 的数学约束可以直接固化为 BRAM 阵列的物理寻址规则。

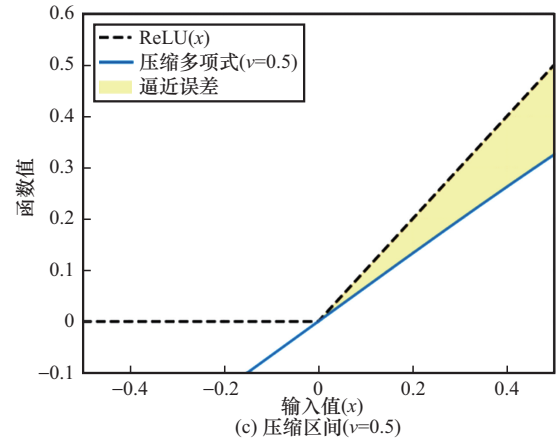
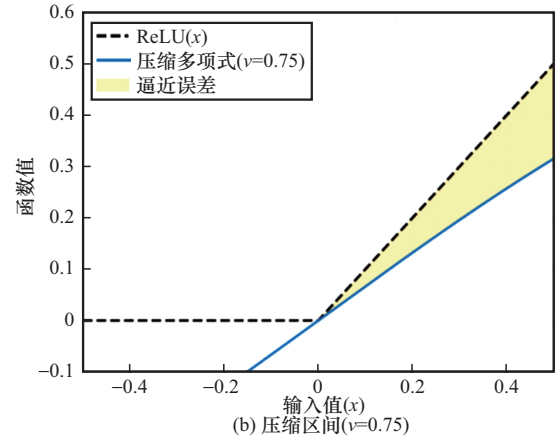
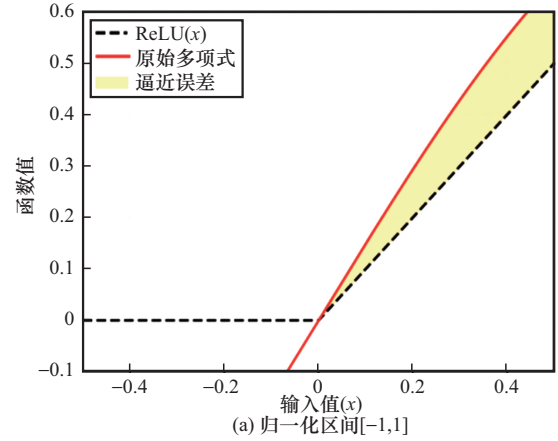


图5 0点邻域ReLU多项式逼近的动态区间缩放与收敛加速机制

为了将微分约束映射至硬件, 即计算区间缩放因子 v , 本文设计了 BRAM 物理寻址规则, 其

示意如图6所示。首先，在BRAM中预存储一系列 v 的候选取值，包括0.1、0.2、0.5等。然后，对于固定多项式的输入值 a_1 ，将其定点化后的高位字段直接作为BRAM的基地址；其低位字段经缩放后实时生成梯度偏移量，缩放倍数根据式(19)计算可得。最终，采用“基地址+截断偏移”的结构合成存放可用区间 v 值的物理地址。该设计能够确保满足算法约束与硬件逻辑的协同设计：一方面，计算所得的 v 满足梯度要求；另一方面，如果 v 的地址超出预存BRAM的地址范围，就会触发BRAM的地址溢出机制，提示需要修改近似多项式。这一机制能够彻底规避多项式逼近密文激活函数而导致的梯度爆炸风险。

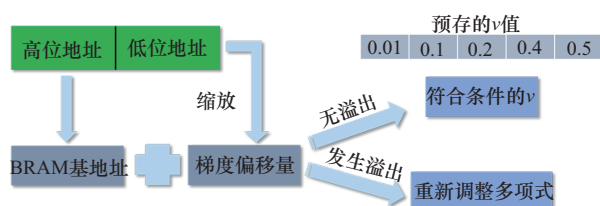


图6 BRAM物理寻址规则示意

2.3.2 异构资源协同机制

传统异构加速框架因资源分配滞后与跨架构协同低效，易陷入“木桶效应”困境：CPU端的静态任务规划难以匹配FPGA的动态负载波动，而粗粒度资源调度导致密文计算路径频繁阻塞。本文采用三大机制实现CPU-FPGA计算任务的动态负载均衡。

1) 基于微软SEAL同态加密软件库^[29]中的BatchEncoder构建NTT-RNS联合映射引擎，将原始输入编码为适配FPGA计算粒度的密文多项式，突破CPU-FPGA数据语义鸿沟。

2) 创新性设计利用多项式次数 d 与NTT规模 N 的乘积 $d \times N$ ，预估FPGA端ReLU函数计算时间窗口。任务调度器基于历史执行日志构建负载预测模型，避免两端长时间处于待机时间。

3) 在FPGA端植入BRAM-DDR4态势感知控制器。该控制器的核心规则基于阈值触发。本文设计了一个NTT活跃度硬件计数器，通过对每个NTT模块添加活跃标志及全局活跃度累加器实现。其中，每个NTT模块的活跃标志由以下条件共同决定：DSP是否正在执行蝶形运算、BRAM是否正在

被连续访问，以及该模块是否处于等待数据状态。

该计数器能实时监测当前正在计算的NTT模块数量，即NTT密度。实验发现，如果NTT的密度高于90%，反而会导致FPGA计算性能下降。因此，本文设置当NTT密度突破85%阈值时，触发计算路径切换：系统将旁路FPGA端的ReLU单元，由CPU端的单指令多数据（SIMD）指令集接管激活函数计算，彻底终结资源争用僵局。

CPU-FPGA异构框架负载—资源联合优化架构如图7所示。

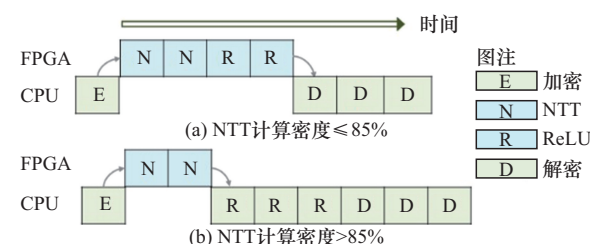


图7 CPU-FPGA异构框架负载—资源联合优化架构

2.3.3 异构接口单元

在CPU-FPGA异构系统中，CPU端专门用于加密数据的中继传输：通过PCIe Gen3×16总线，将经过全同态加密的ReLU输入密文（每个密文多项式含 2^{16} 个系数）传输至FPGA，其数据流本质为加密激活值的超维张量集合。然而，ReLU密文计算的动态特性（如零点邻域缩放因子 v 的实时更新）导致传输数据量呈雪崩式增长——除了密文多项式外，还需同步传输多项式近似参数（含阶数 d 、收敛区间 $[-v, v]$ 等动态配置项）。传统传输架构的线性编码机制因带宽过载而沦为性能瓶颈。

为突破CKKS方案中密文数据洪流对片上存储的冲击，本文提出一种基于异构接口的存储优化机制，即CPU-FPGA异构系统中的高维密文传输相位协调位宽自适应机制，如图8所示。在CPU端完成数据序列化编码处理后，通过PCIe Gen3×16总线建立主机与FPGA间的DMA传输通道。针对传统架构中BRAM端口位宽与密文系数粒度的适配问题，本文创新性地设计了具备相位调谐能力的接口控制器。该模块采用时分复用机制将64 B/cycle的原始数据流解耦为8个8 bit相位通道，每个时钟周期通过动态相位对齐技术完成512 bit到8×64 bit的量子化位宽转换。

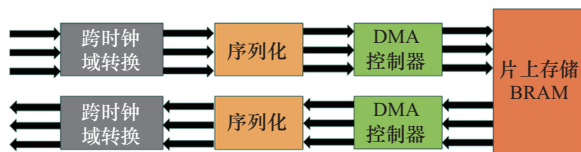


图8 CPU-FPGA异构系统中的高维密文传输相位协调位宽自适应机制

本文利用该异构接口单元和PCIe Gen3×16总线，在CPU端和FPGA端之间构建异步流水线：当FPGA端处于第 N 个Batch密文的多项式近似运算阶段（计算强度达1 024个周期）时，下一组包含 2^{16} 个系数的密文多项式及动态评估密钥参数，已通过预加载通道完成总线传输与反序列化操作。这种基于计算—传输窗口交叠的流水线设计，将传输时延完全吸收在计算时隙内，实现零额外传输开销的连续ReLU计算。

2.4 异构密文推理硬件加速系统

本文在上述ReLU加速框架基础上，提出一种客户端—服务器协同的密文推理系统，如图9所示，实现端到端隐私保护机器学习推理。客户端CPU对初始数据加密后，将密文传输至服务器端；服务器CPU执行ReLU函数的高阶多项式近似计算并生成动态参数，通过序列化接口将参数与密文打包传输至FPGA；FPGA端利用CG-NTT加速同态多项式评估，依次完成卷积、池化与全连接操作，其中ReLU激活函数通过奇对称多项式与零点梯度约束实现硬件友好型近似；最终，结果密文返回服务器CPU解密后，送回客户端。

该系统的核心在于将计算密集型的同态ReLU评估与多项式运算高效卸载至FPGA硬件加速器，并协同CPU完成动态参数生成与任务调度。为验证上述方案的性能，在第3.5节构建

了一个完整的密文推理系统原型，选用ResNet-20和CIFAR-10作为测试基准。该密文推理系统在服务器端全程采用密文计算，在保障数据隐私的前提下，提高了密文推理的实时性。因此，该系统能够为医疗影像分析、金融风控等隐私敏感场景提供一种潜在的解决方案。

然而，在如图9所示的异构密文推理硬件加速系统中，除了依赖前文提及的基于查找表的模运算、NTT运算等操作之外，密文推理还涉及自举操作。自举操作能够刷新密文噪声，是实现多次密文计算和保障密文推理准确率的关键。然而，目前已有的FPGA自举框架^[35-36]中，在FPGA上实现完整的自举操作需要占用60%以上LUT资源及80%以上的BRAM存储资源。因此，为了避免片上资源不足的问题，本文没有将ReLU激活函数和自举操作一并移植到FPGA上实现，而是选择采用已有的CPU实现^[32]。

尽管本文未能在FPGA上同时加速自举操作和ReLU激活函数，但本文所提的分段查找表架构（S-LUT）和动态CG-NTT，依然为自举操作的加速提供了可行的技术路径。其原因在于：自举操作在多项式环上进行运算，运算过程依赖大位宽模运算和多项式运算。此外，自举操作需要根据密文噪声等级动态切换模数链，而动态CG-NTT的表复用策略可以根据自举操作的模数需求，仅激活对应的RNS子模数集合，并将未激活子表所占用的BRAM资源临时分配给自举操作所需的旋转因子存储，从而减少外部存储资源的访问次数。

尽管S-LUT和动态CG-NTT为自举操作的加速提供了可行的技术路径，但要依赖这些算子在FPGA实现快速、有效的自举，仍面临诸多挑

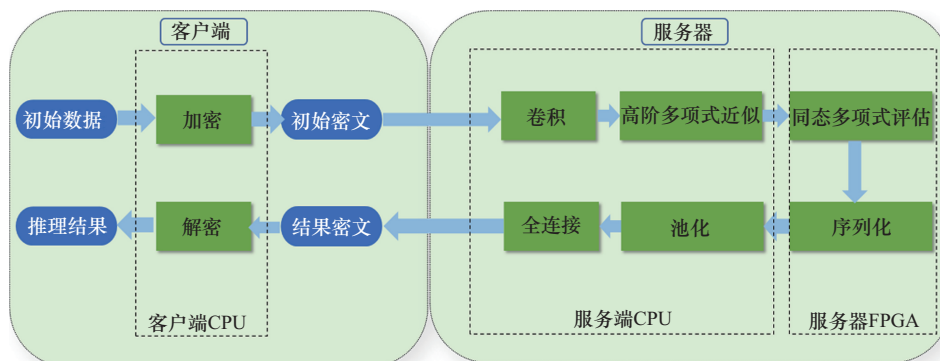


图9 密文推理系统

战, 包括: 如何在仅剩的40% LUT资源内实现S-LUT; 如何将具有强依赖性的自举子步骤进行拆分, 发挥FPGA的并行计算优势; 如何适配自举操作中频繁的模数切换, 避免查表时间空隙等。这些问题都需要在后续工作中进一步攻克。

3 实验

3.1 实验设置

本节基于Xilinx系列FPGA平台实现硬件加速器, 并同步完成软件算法验证。软件实现部分以提出的并行计算框架为基础, 在Ubuntu 22.04 LTS系统上完成部署。硬件实现则采用CPU-FPGA架构, 目标平台为Xilinx Virtex UltraScale+ XCVU9P系列FPGA及Intel Xeon Silver 4214 CPU。其中, FPGA平台最大工作频率为825 MHz, 可用LUT资源为1 182 240个, 可用DSP资源为6 840个, 可用BRAM块为2 160块。

基于环学习带错误 (ring learning with errors, RLWE) 假设的CKKS方案构建全同态加密架构, 其核心参数配置如下: 多项式环维度设为 $N = 2^{12} = 4\,096$, 设置于分圆环 $\mathbb{Z}[X]/(X^{4\,096} + 1)$; 模数链采用FHE中常用的3级动态调整策略 $q = \{q_0 = 39\text{ bit}, q_1 = 35\text{ bit}, q_2 = 35\text{ bit}\}$, 其中初始模数 $q_0 = 2^{39} - 2^{35} + 1$ 为支持快速数论变换的特殊素数, 用于高精度计算; 二级模数 q_1 、 q_2 用于模数切换技术。

3.2 查找表模运算的性能评估

为充分验证算法的先进性, 选取基于进位保留加法的蒙哥马利约简方法^[23]、查找表模约简

方法^[30]、基于混合约简技术的模运算方法^[39]以及本文提出的分段查找表模运算方法开展对比实验。分别采用128 bit、256 bit、512 bit和1 024 bit模数进行实验, 且所有实验均以批量处理 2^{12} 规模次取模操作为基准, 以模拟FHE实际运算需求, 所有运算均在Xilinx Virtex UltraScale+ XCVU9P平台上实现。

模运算性能对比见表1、模运算的优化时延—面积权衡如图10所示。实验结果表明, 分段查找表模运算方法在128 bit、256 bit、512 bit及1 024 bit宽场景下均取得了较好的性能表现。在XCVU9P平台上, 分段查找表模运算方法在不同位宽场景下的时延分别为8.63 ns、15.57 ns、26.79 ns和50.15 ns。相比基于进位保留加法的蒙哥马利约简和查找表模约简方法, 分段查找表模运算方法时延降幅达到96.8%~97.5%; 而混合约简模运算方法采用多级折叠约简平衡DSP和片上存储资源的使用, 但在256 bit、512 bit、1 024 bit位宽场景下, 其时延高于分段查找表模运算方法。

分段查找表模运算方法时延性能的提升主要来源于第2.1节的硬件架构设计: 将复杂的模运算电路转化为固定查表与简单的模加操作, 避免了常见方法中迭代计算带来的时序时延。此外, 分段查找表模运算方法在保持时延随位宽线性增长特性的同时, 其增长斜率也显著低于基于进位保留加法的蒙哥马利约简方法。比如, 从128 bit扩展到512 bit, 分段查找表模运算方法的时延增

表1 模运算性能对比

设计方法	位宽/bit	实现平台	时延/ns	面积
基于进位保留加法的蒙哥马利约简	128	GF 22nm	280.8	$9.7 \times 10^3 \mu\text{m}^2$
	256		573.92	$19.7 \times 10^3 \mu\text{m}^2$
	512		1 056.28	$39.7 \times 10^3 \mu\text{m}^2$
查找表模约简	128	Virtex UltraScale+ XCVU9P	11.85	24 120 LUTs, 81 DSPs
	256		17.55	87 622 LUTs, 289 DSPs
	512		25.68	329 868 LUTs, 1089 DSPs
基于混合约简技术的模运算	256	Agilex7	57	22 580 ALMs, 90 DSPs
	512		82	277 344 ALMs, 1710 DSPs
	1 024		104	142 972 ALMs, 975 DSPs
分段查找表模运算	128	Virtex UltraScale+ XCVU9P	8.63	12 165 LUTs, 85 DSPs
	256		15.57	50 064 LUTs, 280 DSPs
	512		26.79	178 631 LUTs, 517 DSPs
	1 024		50.15	736 581 LUTs, 809 DSPs

加了约 210.4%，基于进位保留加法的蒙哥马利约简方法则增加了约 276%。

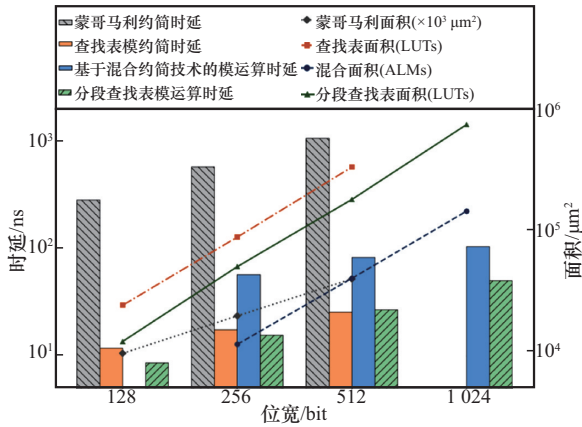


图 10 模运算的优化时延-面积权衡

在硬件资源消耗方面，分段查找表模运算方法优于传统的 LUT 方法，即查找表模约简方法。以 128 bit 实现为例，所需 LUT 数量从 24 120 个降至 12 165 个，降幅为约 49.6%，同时 DSP 资源使用量维持在相近水平。这种资源优势在更大位宽下更加明显：在 512 bit 位宽下，分段查找表模运算方法所需的 LUT 由 329 868 个降至 178 631 个，DSP 则从 1 089 个降至 517 个。与混合约简模运算相比，在 512 bit 位宽下，分段查找表模运算方法的 DSP 消耗减少 69.7%。这主要得益于通过查表将大部分计算转成 LUT 逻辑，进而减少了对 DSP 资源的依赖。

实验数据证实，分段查找表模运算方法通过精细化的查表分段策略与数据通路优化，在保证运算速度的同时实现了面积的优化。此外，在 1 024 bit 大位宽场景下，分段查找表模运算方法仍然可以在 Virtex 平台上实现，并未出现 LUT 资源不足的问题。这为大规模密码学运算单元的硬件实现提供了一种新的设计思路。

3.3 CG-NTT 的性能评估

为验证本文提出的基于分段查找表模运算方法与动态表复用机制的 CG-NTT 架构 (Ours) 的优化效果，基于 Xilinx Virtex UltraScale+ XCVU9P 平台，对比传统 CT-NTT 架构^[31]、现有 CG-NTT^[28] 架构及本文方案在吞吐率、时延等维度的性能差异。实验参数与第 3.1 节一致，多项式环维度 $N = 4\,096$ 个，每组测试包含 2^{12} 次 NTT 运算。

3 种架构在 LUT、BRAM、触发器 (flip-flop, FF) 等指标下的性能对比见表 2。本文方案仅需 17 个 DSP 单元，较 CT-NTT 与 CG-NTT 分别减少约 37% 与 15%，表明动态表复用策略显著降低了高精度运算的硬件依赖。此外，本文方案的面积时间积 (area-time product, ATP) 为 $1.00\times$ ，CT-NTT 与 CG-NTT 的 ATP 分别为本文方案的 6.17 倍和 1.14 倍，表明其在单位面积下的计算效率显著领先。此优势源于动态表复用策略对硬件资源浪费的有效规避。

表 2 3 种架构性能对比

架构	FF 数量/个	LUT 数量/个	DSP 数量/个	BRAM 数量/块	ATP
本文方案	798	3 542	17	17	1.00×
CT-NTT	1 995	22 316	27	11	6.17×
CG-NTT	1 817	3 324	20	14	1.14×

注：ATP 列数值以本文方案为基准，1.00× 代表基准值，其余数值为相对基准的倍数。

3 种架构的时延与吞吐量性能对比如图 11 所示，本文方案的时延较 CT-NTT 与 CG-NTT 方案降低约 50.8%，且吞吐量为 254 kops/s，较 CT-NTT 提升 117%，较 CG-NTT 提升 30.3%。此优势源于分段式 LUT-MA 对模乘关键路径的简化，以及预取-计算流水线对内存访问冲突的动态规避。

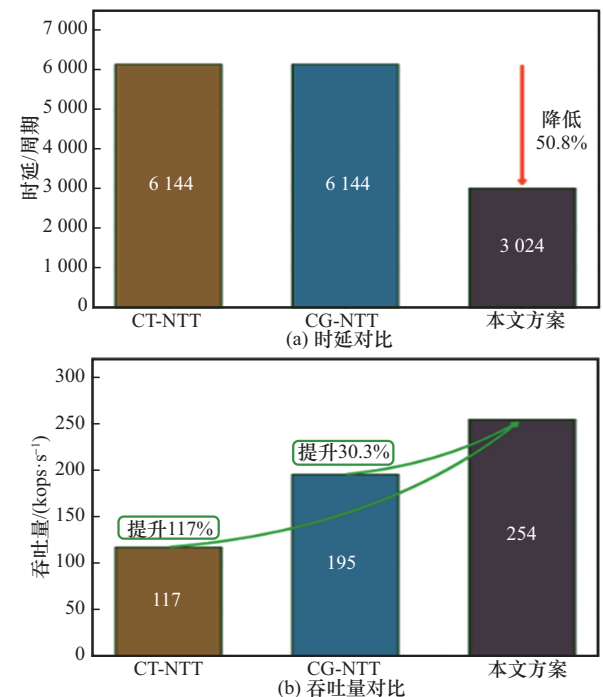


图 11 3 种架构时延与吞吐量性能对比

3.4 同态ReLU近似评估的性能评估

在全同态加密推理系统中，同态ReLU激活函数多项式近似评估的高效实现面临双重挑战：其非线性特性与同态加密计算流之间的语义断层、近似精度与硬件资源约束的动态平衡。本节针对CKKS方案中基于最小最大多项式的ReLU近似算法，在Xilinx Virtex UltraScale+ XCVU9P FPGA平台上，实现本文设计的ReLU激活函数多项式近似方案与基于单指令多数据流的CPU加速方案EvoReLU^[32]、AppReLU^[33]的性能对比。其中，对于EvoReLU^[32]，使用其官方GitHub仓库实现该方案，配置严格遵循原文献；对于AppReLU^[33]，采用其通信作者提供的非公开源代码，并进行了严格的交叉验证以确保对原始设计的忠实复现。两个对比方案均在Intel Xeon Silver 4214 CPU平台上实现，与其他实验中使用的CPU平台保持一致。测试基准设定为批量处理 2^{16} 个密文激活单元，采用的复合多项式阶数为{3,7}，本次实验中每个ReLU操作时延是连续100次计算的平均值。

同态ReLU近似评估方案性能对比见表3，ReLU激活函数多项式近似方案在计算效率上实现了显著突破。相较于AppReLU方案与EvoReLU方案，ReLU激活函数多项式近似方案将ReLU激活时延大幅降低至19.2 ms，较两者分别减少约95.3%与95.1%。这一提升的关键在于第2.2节和第2.3节所述的优化架构及本文提出的资源协同机制，而非单纯依赖FPGA的硬件并行性。第3.2节与第3.3节的实验结果表明，在相同FPGA平台下，ReLU激活函数多项式近似方案可降低模运算时延并减少NTT所需硬件资源，本节的同态ReLU近似评估框架正是基于上述架构搭建。由于目前公开文献中缺乏可直接对比的同类CPU-FPGA异构方案，为展示异构化的优势，本节选取纯CPU方案作为性能参照。

ReLU激活函数多项式近似方案通过软硬件协同设计实现了资源效率提升：FPGA端仅需部署多项式运算模块，LUT与DSP资源占用分别控制在21 973个和289个；通过基于BRAM-DDR4态势感知的动态分配策略，FPGA与CPU的资源利用率分别稳定在91.4%与90.2%，综合

占用率为90.8%，低于AppReLU方案和EvoReLU方案。与此同时，ReLU激活函数多项式近似方案摆脱了对固定指令集的依赖，通过动态路径切换灵活规避串行瓶颈。

表3 同态ReLU近似评估方案性能对比

指标	AppReLU ^[33]	本文方案	EvoReLU ^[32]
ReLU时延	410.8 ms	19.2 ms	394.8 ms
FPGA LUT 占用量/个	N/A	21 973	N/A
FPGA DSP 占用量/个	N/A	289	N/A
CPU占用率	93.4%	90.2%	95.4%
FPGA占用率	N/A	91.4%	N/A

此外，实验还测试了不同复合多项式阶数下的同态ReLU近似性能，包括密文乘法深度、时延以及对应的平均误差，结果见表4。实验结果表明，随着复合阶数从{13}增加至{7,15}，乘法深度也从5增至8，但时延仅从16.6 ms增至26.5 ms，仍远低于AppReLU和EvoReLU方案。这主要得益于本文所提系统方案实现了高效的模运算和CG-NTT模块，使得额外增加的多项式模乘运算能够较快完成。实验结果还显示，高阶多项式可以显著降低近似误差，有效提升ReLU近似的精度。然而，高阶多项式阶数会增加乘法深度，从而需要额外的自举操作次数。因此，在实际密文推理系统中，大多不采用高阶多项式。

表4 不同复合多项式阶数下同态ReLU近似的性能对比

复合多项式的阶数	乘法深度	时延/ms	平均误差
{13}	5	16.6	3.90e-02
{3,7}	6	19.2	1.26e-02
{7,7}	7	22.6	5.15e-04
{7,15}	8	26.5	1.33e-04

3.5 异构密文推理硬件加速系统的性能评估

为验证本文提出的CPU-FPGA异构协同加速方案在密文推理任务中的性能优势，基于Xilinx Virtex UltraScale+ XCVU9P FPGA平台与Intel Xeon Silver 4214 CPU搭建的异构推理系统开展实验。其中，CPU端负责逻辑调度与轻量计算；FPGA端基于本文提出的CG-NTT框架部署定制

化多项式加速单元, 以实现密文 ReLU 运算硬件加速。对比方案采用 ResNet-20 模型在 Cifar10 数据集上的密文多项式推理, 其中 LPFHE^[33]与 AutoFHE^[32]方案在 Intel Xeon Silver 4214 CPU 环境下以纯软件模式运行, A*FV^[34]则基于 3×NVIDIA P100 GPU 与 1×NVIDIA V100 GPU 的异构平台实现。

同态加密推理方案的性能对比见表 5。在 ResNet-20 模型的加密推理任务中, 本文方案展现出显著性能优势。实验数据显示: 基于 GPU 的 A*FV 方案耗时最长, 纯 CPU 实现的 LPFHE 和 AutoFHE 方案分别耗时 122 s 和 150 s, 而本文所提方案仅需 113 s 即可完成推理。具体而言, 本文方案相较 A*FV 实现 79.6% 的加速比, 较 LPFHE 和 AutoFHE 分别提升 7.5% 和 24.6% 的推理效率。

表 5 同态加密推理方案的性能对比

方案	硬件平台	耗时/s
A*FV	3×NVIDIA P100 1×NVIDIA V100 GPU	554
LPFHE	Intel Xeon Silver 4214 CPU	122
AutoFHE	Intel Xeon Silver 4214 CPU	150
本文方案	Intel Xeon Silver 4214 CPU Virtex UltraScale+ XCVU9P	113

同态加密推理方案的时延与准确率性能对比进一步呈现了各方案的推理耗时与分类准确率的综合表现, 如图 12 所示。除了 A*FV 因密文计算中的精度损失累积问题, 准确率仅为 77.6% 外, 其他方案的准确率均高于 90%。此外, 本文方案的准确率仅比 AutoFHE 方案低 1.6%, 其原因在于本文方案采用了相对低阶多项式近似 ReLU 带来的误差。与同样采用相对低阶多项式的 LPFHE 具有相同的准确率, 这也说明本文方案的硬件架构没有额外引入新的误差。结合图 12 的数据能看出, 本文方案在精度与现有纯 CPU 方案相当的前提下, 推理耗时更短。

此外, 本文方案在成本方面虽然要高于 CPU 方案, 但低于 A*FV 方案, 且本文加速模块仅需中端的 FPGA 即可实现, 成本相对较低且易于商业化。在功耗方面, 本文方案仅将部分核心单元

放置在 FPGA 上, 使 FPGA 和 CPU 均无需满负载运行, 从而在低功耗需求应用场景中展现出潜在优势。

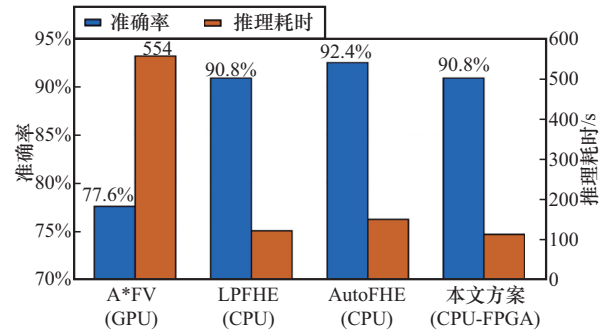


图 12 同态加密推理方案的时延与准确率性能对比

尽管本文提出的 CPU-FPGA 异构加速架构在密文推理任务中优于对比方案, 但相较于 LPFHE 方案仅提速 7.5%, 说明目前研究在实用化需求上仍有优化空间。分析表明, 当前方案的性能瓶颈主要源于未集成自举操作的加速优化。作为全同态加密中重置噪声、支持深层计算的关键步骤, 自举操作具有极高的计算复杂度且频繁触发, 在密文推理中占据 55% 以上耗时^[33-34]。在复杂模型或多层推理任务中, 自举操作将迅速成为端到端性能的主要开销。因此, 未加速的自举操作会显著制约系统整体性能, 特别是在需要多次自举的密文推理场景中, 其耗时会在一定程度上抵消本文框架的硬件加速收益。

然而, 如表 5 所示, 本文方案在未加速自举的前提下仍实现了不低于 7.5% 的推理加速, 其原因在于第 3.4 节中展示的快速同态 ReLU 评估。实验数据也表明, 未加速的自举操作并未完全抵消本文方案对 ReLU 的加速效果。此外, 本文所提密文推理系统的核心价值并不仅在于缩短推理耗时, 更在于构建了一套资源高效、可扩展性的 ReLU 同态评估硬件框架。相较于纯 CPU 方案, 本系统中的 ReLU 架构在保证推理精度的同时, 展现出更低的硬件资源消耗特性, 更适用于边缘端这类资源受限的部署场景^[40-41]。后续研究也将重点探索自举操作的定制化硬件加速策略, 以进一步提高端到端密文推理系统的性能。

4 结束语

本文针对全同态加密中ReLU激活函数的高效密文计算难题,提出了一种基于CPU-FPGA异构协同的硬件加速系统,通过算法与硬件的深度协同优化,实现了端到端性能的显著提升。首先,设计的分段查找表模运算采用双级分解映射机制,结合剩余数系统动态表复用策略,将大位宽模乘运算的存储资源从 2^n 压缩至平方根级 $2^{n/2}$,实验表明其模运算时延较现有方案降低97%,存储资源占用减少45.8%~52.7%。其次,提出的可重构CG-NTT架构通过动态表复用与预取一计算协同流水线技术,消除内存地址计算冗余,将计算单元减少15%~37%,运算时延降低50.8%。此外,针对密文推理中的非线性瓶颈,设计的ReLU激活函数多项式近似方案通过奇对称约束与零点邻域微分固化技术,在FPGA端实现了21.4倍加速。

本文方案的一个主要局限在于未能对全同态加密中重置噪声、支持深层计算的关键操作——自举操作进行硬件加速。自举操作计算复杂度极高且在密文推理中频繁调用,其未被加速会抵消部分性能收益,制约系统实用化。

然而,本文提出的分段查找表与动态资源复用CG-NTT架构,经过完善优化后可用于自举加速。其原因在于NTT是自举操作的关键模块,优化该架构能够提升自举操作性能^[35-36]。未来研究将重点实现自举操作加速,使得自举操作与现有异构架构深度融合,进一步优化密文推理系统性能。

参考文献:

- [1] Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms[J]. Foundations of secure computation, 1978, 4(11): 169-180.
- [2] Gentry C. A fully homomorphic encryption scheme[D]. Stanford: Stanford University, 2009.
- [3] Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping[J]. ACM Transactions on Computation Theory, 2014, 6(3): 1-36.
- [4] Fan J, Vercauteren F. Somewhat practical fully homomorphic encryption[EB]. 2012.
- [5] Chillotti I, Gama N, Georgieva M, et al. TFHE: fast fully homomorphic encryption over the torus[J]. Journal of Cryptology, 2020, 33(1): 34-91.
- [6] Cheon J H, Kim A, Kim M, et al. Homomorphic encryption for arithmetic of approximate numbers[C]//Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security. Cham: Springer, 2017: 409-437.
- [7] Cheon J H, Han K, Kim A, et al. A full RNS variant of approximate homomorphic encryption[M]//Selected Areas in Cryptography - SAC 2018. Cham: Springer, 2019: 347-368.
- [8] Kim T, Kwak H, Lee D, et al. Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition[C]//Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2023: 726-740.
- [9] Gilad-Bachrach R, Dowlin N, Laine K, et al. Cryptonets: applying neural networks to encrypted data with high throughput and accuracy[C]//Proceedings of the 33rd International Conference on International Conference on Machine Learning. JMLR, 2016: 201-210.
- [10] Brutzkus A, Gilad-Bachrach R, Elisha O. Low latency privacy preserving inference[C]//Proceedings of the 36th International Conference on Machine Learning. PMLR, 2019: 812-821.
- [11] Mishra P, Lehmkuhl R, Srinivasan A, et al. Delphi: a cryptographic inference system for neural networks[C]//Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice. New York: ACM, 2020: 27-30.
- [12] Lou Q, Jiang L. Homet: a homomorphic-encryption-friendly privacy-preserving mobile neural network architecture[C]//Proceedings of the 38th International Conference on Machine Learning. PMLR, 2021: 7102-7110.
- [13] Park J, Kim M J, Jung W, et al. AESPA: accuracy preserving low-degree polynomial activation for fast private inference[PP]. V2. arXiv (2022-02-18)[2025-09-05]. arXiv: 2201.06699.
- [14] Lee E, Lee J W, No J S, et al. Minimax approximation of sign function by composite polynomial for homomorphic comparison[J]. IEEE Transactions on Dependable and Secure Computing, 2022, 19(6): 3711-3727.
- [15] Huang Y, Gong X S, Kong X Y, et al. EFFACT: a highly efficient full-stack FHE acceleration platform[C]//Proceedings of the 2025 IEEE International Symposium on High Performance Computer Architecture (HPCA). Piscataway: IEEE Press, 2025: 1143-1157.
- [16] Cheon S, Lee Y, Kim D, et al. DaCapo: automatic bootstrapping management for efficient fully homomorphic encryption[C]//Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24). 2024: 6993-7010.
- [17] Ju J H, Park J, Kim J, et al. NeuJeans: private neural network inference with joint optimization of convolution and FHE bootstrapping[C]//Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2024: 4361-4375.
- [18] Yang Y H, Xu X C, Zhang H B, et al. Hydra: scale-out FHE accelerator architecture for secure deep learning on FPGA[C]//Proceedings of the 2025 IEEE International Symposium on High Performance Computer Architecture (HPCA). Piscataway: IEEE Press, 2025: 1174-1186.
- [19] Zhu Y L, Wang X Y, Ju L, et al. FxHENN: FPGA-based accelera-

- tion framework for homomorphic encrypted CNN inference[C]//Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE Press, 2023: 896-907.
- [20] Ye T, Kuppannagari S R, Kannan R, et al. Performance modeling and FPGA acceleration of homomorphic encrypted convolution[C]//Proceedings of the 2021 31st International Conference on Field-Programmable Logic and Applications (FPL). Piscataway: IEEE Press, 2021: 115-121.
- [21] Han M Q, Zhu Y L, Lou Q, et al. coxHE: a software-hardware co-design framework for FPGA acceleration of homomorphic computation[C]//Proceedings of the 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). Piscataway: IEEE Press, 2022: 1353-1358.
- [22] Yang Y H, Zhang H Z, Fan S Y, et al. Poseidon: practical homomorphic encryption accelerator[C]//Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE Press, 2023: 870-881.
- [23] Mazonka O, Chielle E, Soni D, et al. Fast and compact interleaved modular multiplication based on carry save addition[C]//Proceedings of the 2022 IEEE/ACM International Conference on Computer Aided Design (ICCAD). Piscataway: IEEE Press, 2023: 1-9.
- [24] Guan Z Y, Zhu Y Q, Huang Y C, et al. ESC-NTT: an elastic, seamless and compact architecture for multi-parameter NTT acceleration[C]//Proceedings of the 2024 Design, Automation & Test in Europe Conference & Exhibition (DATE). Piscataway: IEEE Press, 2024: 1-6.
- [25] Geng Y, Hu X, Li M H, et al. Rethinking parallel memory access pattern in number theoretic transform design[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2023, 70(5): 1689-1693.
- [26] Su Y, Yang B L, Yang C, et al. ReMCA: a reconfigurable multi-core architecture for full RNS variant of BFV homomorphic evaluation[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(7): 2857-2870.
- [27] Su Y, Yang B L, Yang C, et al. A highly unified reconfigurable multicore architecture to speed up NTT/INTT for homomorphic polynomial multiplication[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022, 30(8): 993-1006.
- [28] Ren X L, Chen Z H, Gu Z, et al. CHAM: a customized homomorphic encryption accelerator for fast matrix-vector product[C]//Proceedings of the 2023 60th ACM/IEEE Design Automation Conference (DAC). Piscataway: IEEE Press, 2023: 1-6.
- [29] Chen H, Laine K, Player R. Simple encrypted arithmetic library - SEAL v2.1[M]//Financial Cryptography and Data Security. Cham: Springer, 2017: 3-18.
- [30] Öztürk E. Design and implementation of a low-latency modular multiplication algorithm[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(6): 1902-1911.
- [31] Riazzi M S, Laine K, Pelton B, et al. HEAX: an architecture for computing on encrypted data[C]//Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 1295-1309.
- [32] Ao W, Boddeti V N. Automated adaption of CNNs for efficient evaluation over FHE[PP]. V1. arXiv(2023-10-12) [2025-09-05]. arXiv:2310.08012.
- [33] Wan J P, Li D J, Fang J B, et al. LPFHE: low-complexity polynomial CNNs for secure inference over FHE[M]//Computer Security - ESORICS 2024. Cham: Springer, 2024: 403-423.
- [34] Al Badawi A, Jin C, Lin J, et al. Towards the AlexNet moment for homomorphic encryption: HCNN, the first homomorphic CNN on encrypted data with GPUs[J]. IEEE Transactions on Emerging Topics in Computing, 2021, 9(3): 1330-1343.
- [35] Xu Z H, Ye T, Kannan R, et al. FAST: FPGA acceleration of fully homomorphic encryption with efficient bootstrapping[C]//Proceedings of the 2025 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. New York: ACM, 2025: 115-126.
- [36] Agrawal R, de Castro L, Yang G W, et al. FAB: an FPGA-based accelerator for bootstrappable fully homomorphic encryption[C]//Proceedings of the 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). Piscataway: IEEE Press, 2023: 882-895.
- [37] Meng X C, Jiang Z J, Lyu Y D. HF-NTT: hazard-free dataflow accelerator for number theoretic transform[PP]. arXiv(2024-10-07) [2025-09-05]. arXiv:2410.04805.
- [38] Im N, Yang H, Eom Y, et al. Efficient twiddle factor generators for NTT[J]. Electronics, 2024, 13(16): 3128.
- [39] Gribok S, Langhammer M, Pasca B. FPGA modular multipliers using hybrid reduction techniques[C]//Proceedings of the 2024 34th International Conference on Field-Programmable Logic and Applications (FPL). Piscataway: IEEE Press, 2024: 288-296.
- [40] 顾天晟, 曾福康, 邵思思, 等. 基于拟态防御和 Paillier 加密的智能物联网终端安全防护方法[J]. 网络与信息安全学报, 2024, 10(6): 164-176.
- Gu T S, Zeng F K, Shao S S, et al. Security protection method based on mimic defense and Paillier encryption for smart IoT terminals[J]. Chinese Journal of Network and Information Security, 2024, 10(6): 164-176.
- [41] 王圆圆, 韩益亮, 吴立强, 等. 智能电网中隐私保护的多维多函数数据聚合方案[J]. 网络与信息安全学报, 2025, 11(3): 163-174.
- Wang Y Y, Han Y L, Wu L Q, et al. Privacy-preserving multi-dimensional multi-function data aggregation scheme in smart grid[J]. Chinese Journal of Network and Information Security, 2025, 11(3): 163-174.

[作者简介]



廖颖龙 (2001-), 男, 暨南大学物理与光电工程学院硕士生, 主要研究方向为同态加密、硬件加速。



罗羽乔（2001-），女，暨南大学智能科学与工程学院硕士生，主要研究方向为人工智能、对抗攻击。



姚磊（1992-），男，硕士，深圳云安宝科技有限公司总经理，主要研究方向为隐私计算、数据安全等。



李丹婕（2000-），女，暨南大学物理与光电工程学院硕士生，主要研究方向为隐私计算、同态加密。



蒋琳（1980-），女，博士，哈尔滨工业大学（深圳）教授、博士生导师，主要研究方向为全同态加密、零知识证明、安全多方计算。



李灵武（1999-），男，暨南大学物理与光电工程学院博士生，主要研究方向为同态加密、硬件加速。



方俊彬（1981-），男，博士，暨南大学教授、博士生导师，主要研究方向为硬件加速、全同态加密。