

CRYSTAL-KYBER 硬件设计优化空间探索

穆嘉楠^{1,2}, 赵艺璇^{1,2}, 严寒^{1,2}, 宋金峰^{1,2}, 叶靖^{1,2}, 李华伟^{1,2}, 李晓维^{1,2}

¹中国科学院计算技术研究所体系结构国家重点实验室 北京 中国 100094

²中国科学院大学计算机学院 北京 中国 101408

摘要 公钥密码学对全球数字信息系统的安全起着至关重要的作用。然而,随着量子计算机研究的发展和 Shor 算法等的出现,公钥密码学的安全性受到了潜在的极大的威胁。因此,能够抵抗量子计算机攻击的密码算法开始受到密码学界的关注,美国国家标准与技术研究院(National Institute of Standards and Technology, NIST)发起了后量子密码(Post-quantum cryptography, PQC)算法标准全球征集竞赛。在参选的算法中,基于格的算法在安全性、公钥私钥尺寸和运算速度中达到了较好的权衡,因此是最有潜力的后量子加密算法体制。而 CRYSTALS-KYBER 作为基于格的密钥封装算法(Key encapsulation mechanism, KEM),通过了该全球征集竞赛的三轮遴选。对于后量子密码算法,算法的硬件实现效率是一个重要评价指标。因此,本文使用高层次综合工具(High-level synthesis, HLS),针对 CRYSTALS-KYBER 的三个主模块(密钥生成,密钥封装和密钥解封装),在不同参数集下探索了硬件设计的实现和优化空间。作为一种快速便捷的电路设计方法,HLS 可以用来对不同算法的硬件实现进行高效和便捷的探索。本文利用该工具,对 CRYSTALS-KYBER 的软件代码进行了分析,并尝试不同的组合策略来优化 HLS 硬件实现结果,并最终获得了最优化的电路结构。同时,本文编写了 tcl-perl 协同脚本,以自动化地搜索最优优化策略,获得最优电路结构。实验结果表明,适度优化循环和时序约束可以大大提高 HLS 综合得到的 KYBER 电路性能。与已有的软件实现相比,本文具有明显的性能优势。与 HLS 实现工作相比,本文对 Kyber-512 的优化使得封装算法的性能提高了 75%,解封算法的性能提高了 55.1%。与基准数据相比,密钥生成算法的性能提高了 44.2%。对于 CRYSTALS-KYBER 的另外两个参数集(Kyber-768 和 Kyber-1024),本文也获得了类似的优化效果。

关键词 公钥密码学; 后量子密码学; CRYSTALS-KYBER; 高层次综合; 优化设计
中图法分类号 TP309.7 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.11.05

Optimization Space Exploration of Hardware Design for CRYSTAL-KYBER

MU Jianan^{1,2}, ZHAO Yixuan^{1,2}, YAN Han^{1,2}, SONG Jinfeng^{1,2}, YE Jing^{1,2}, LI Huawei^{1,2}, LI Xiaowei^{1,2}

¹State Key Laboratory of Computer Architecture, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing 100094, China

²School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 101408, China

Abstract Public key cryptography plays a vital role in the security of nowadays global digital information systems. However, with the development of quantum computing and the emergence of Shor's algorithm, the security of public key cryptography has been potentially greatly threatened. Therefore, cryptographic algorithms that can resist the attack from an adversary even has access to a quantum computer have begun to attract the attention of the cryptography community. The National Institute of Standards and Technology (NIST) has launched a global solicitation for the post-quantum cryptography algorithms standard. Among the participating algorithms, the lattice-based algorithm scheme achieves a good trade-off in security, key size, and operation speed, so it is the most potential post-quantum encryption algorithm scheme. CRYSTALS-KYBER, as a lattice-based Key Encapsulation Mechanism (KEM) algorithm, passed three rounds of the global solicitation for post-quantum cryptography algorithms standard. For post-quantum cryptographic algorithms, the hardware implementation efficiency of the algorithm is an important evaluation index. Therefore, this article explores the realization and optimization space of hardware design for the three main modules of CRYSTALS-KYBER (Key generation, key encapsulation, and key decapsulation) under different parameter sets, using the high-level synthesis tools (High-level synthesis, HLS). As a high-level hardware design method, HLS can be used to efficiently and conveniently explore the hardware implementation of different algorithms. This paper uses the HLS tools to analyze the software implementation of CRYSTALS-KYBER, and try different combination strategies to optimize the HLS hardware implementation results, and finally obtain the most optimized hardware structure. At the same time, this paper provides a tcl-perl collaboration script to automatically search for the optimal optimization strategy and

通讯作者: 叶靖, 博士, 副研究员, Email: yejing@ict.ac.cn。

本文得到了国家自然科学基金(NSFC)区域创新发展联合基金(No. U20A20202)的支持。

收稿日期: 2021-09-03; 修改日期: 2021-10-09; 定稿日期: 2021-10-21

obtain the optimal hardware structure. The experimental results show that the performance of the obtained hardware can be greatly improved by moderately optimizing the loops and timing constraints. In comparison with the state-of-the-art software implementation, this paper shows an obvious performance advantage. In comparison with the state-of-the-art HLS implementation, our optimizations of Kyber-512 improve the performance by up to 75% for key encapsulation algorithm and 55.1% for key decapsulation algorithm. And compared with the baseline, the performance was improved by 44.2% in the key generation algorithm. For the other two parameter sets (Kyber-768 and Kyber-1024), the same optimization effect is obtained.

Key words public key cryptography; post-quantum cryptography; CRYSTALS-KYBER; high-level synthesis; optimization

1 引言

公钥密码学为全球数字通信系统的构建发挥了重要作用^[1]。当前常用的密钥通信协议主要依赖三个核心密码功能: 公钥加密、密钥交换和数字签名。目前, 这些功能主要通过 Diffie-Hellman 密钥交换、RSA 密码体制和 ECC 密码体制等密码算法来实现。它的安全性通常取决于其背后的数学问题的计算困难度。

近年来, 量子计算算法^[2-3]和量子计算机^[4]的研究不断得到推进和进步。1994 年, Peter Shor 证明使用量子计算机可以有效地解决这些密码系统的底层数学问题。Shor 给出了一种指数加速算法, 可用于解决经典的困难数论问题, 例如大数分解和离散对数求解问题。而这些问题是现代密码系统安全的数学理论基础。

Grover 搜索算法可以双倍速度求解非结构化搜索问题, 大大提高密钥空间的搜索效率, 加速对称密码和散列加密系统的碰撞破解。虽然它不会完全破坏当前的密码系统, 但这使得我们需要提高密钥大小来抵抗其攻击。

同时, 滑铁卢大学量子计算学院的联合创始人米歇尔·莫斯卡 (Michele Mosca) 表示, 到 2026 年, 量子计算机将有可能可能性攻破 RSA-2048, 而到 2031 年, 莫斯卡认为成功攻击的概率将达到 50%^[5]。

因此, 量子计算机强大的计算能力对现代加密技术的安全性和隐私性构成的威胁成为亟待考虑和解决的新问题^[2]。对于掌握量子计算能力的攻击者, 现有对称密码的安全性将降低一半, 而现有非对称密码将被彻底攻破, 因此我们需要寻找全新的面向量子计算威胁安全的密码算法。

作为密码系统的重要研究方向, 后量子密码学的研究正在快速发展。和基于量子计算机和量子通信环境的量子密码学不同, 后量子密码学基于运行在经典计算机(如非量子计算机)上的密码系统, 足以

抵抗传统计算机和量子计算机的攻击, 从而可以和现有的网络与通信协议交互工作。

美国国家标准技术研究所在全球范围内开展了后量子算法的征集工作, 2019 年 1 月 30 日, 开展了第二轮征集工作, 选拔出 26 个后量子密码机制。其中, 关注度较高的算法可以分为以下几类: 基于格、基于编码、基于哈希、基于多变量密码学和基于同源, 这些算法有望解决后量子加密的问题。上述各种后量子加密算法之间有很大区别: 开发时间不同, 算法结构各有特点, 性能各有优劣。在这些算法中, 基于格的算法在安全性、公钥私钥尺寸和运算速度中达到了较好的权衡, 因此是最有潜力的后量子加密算法之一^[1,6]。而且在 2020 年公布的第三轮征集结果中, 作为一种基于格的密钥封装算法, CRYSTALS-KYBER 成功入选^[7]。

尽管 CRYSTALS-KYBER 的安全性得到承认, 但它的性能仍然有待探索^[8-9]。本论文根据 CRYSTALS-KYBER 算法进行了硬件设计的优化空间探索。本论文的主要贡献点如下:

(1) 针对 KYBER 中的循环设计, 本文在硬件设计中进行了优化。本论文分析了 CRYSTALS-KYBER 的 C 语言代码, 探索了整个函数和关键循环的调用关系。

(2) 使用 HLS 综合算法的 C 语言代码, 通过循环展开和循环流水进行设计空间探索。同时, 时钟约束的影响也将纳入考虑范围。

(3) 为了方便进行设计空间探索, 本论文设计了 tcl-perl 脚本程序, 用以自动运行 HLS。该程序还可以用于其他后量子算法的设计空间探索。

(4) 密钥生成、封装、解封装的优化均取得了积极效果。在 Kyber-512 中, 三个模块分别实现了 44.2%、55.7% 和 50.2% 的优化效率。对于封装和解封装模块, 与已发表的最新成果相比, 性能分别达到了 75% 和 55.1% 的提升。此外, 本论文还在电路面积和速度之间实现了更好的权衡。

(5) 对于不同的参数集, 优化策略的选择对优化效率的影响大致相同。

本论文的组织结构如下: 第二部分介绍了后量子加密和 CRYSTALS-KYBER, 包括算法功能介绍; 第三部分描述并分析了设计空间探索过程; 第四部分展示了优化效果; 第五部分总结全文。

2 PQC 与 CRYSTAL-KYBER

量子计算机与 Shor 算法的出现极大地提升了计算机的计算能力^[2]。尽管对称加密算法通过扩展密钥大小仍能保证自身的安全性, 但公钥加密算法在很大程度上遭受了不可逆转的损失^[1,10]。解决该问题的关键是, 发展一种新的后量子公钥加密体系。

NIST 收集的 PQC 算法主要包含两种类型: 密钥封装算法(也称为密钥封装机制, KEM)和数字签名算法^[7,11]。本论文主要关注前一种。PQC 中使用的 KEM 是一种抵抗量子攻击的有效手段。不同于使用公钥-私钥对直接传输信息, 密钥封装机制在传输过程中添加了一个共享秘密, 使用公钥和私钥可对共享秘密完成封装和解封装。不能被量子计算机攻破的密钥生成函数(例如哈希函数), 将被应用在加密封

装和解密封装的进程中。接着被用来完成对消息的加密和解密^[12-14]。因此, KEM 的主函数包括密钥生成, 密钥封装和密钥解封装^[14]。

(1) 密钥生成部分负责生成公钥 pk 和私钥 sk 。

(2) 密钥封装部分接收公钥 pk , 并通过对 pk 的加密封装, 输出共享秘密 ss 以及密文 ct 。

(3) 密钥解封装部分使用接收得到的私钥 sk 和密文 ct , 恢复出共享秘密 ss 。

在 PQC 算法中, 基于格的密钥建立算法相对简单、高效且并行度高, 所以受到了广泛关注和认可^[1]。

CRYSTALS-KYBER(或简称 Kyber)是一种基于格的 IND-CCA2 安全 KEM 算法, 其安全性建立在模格中解决 learning-with-error 问题的困难性上。Kyber 有三种参数集, Kyber-512, Kyber-768, Kyber-1024, 分别对应安全等级 1, 3, 5^[12-14]。在本论文的探索和实验过程中, 三种参数集的实验结果都会被展出。

3 设计空间探索

为提高电路性能, 本论文使用了图 1 所示的流程探索设计空间。

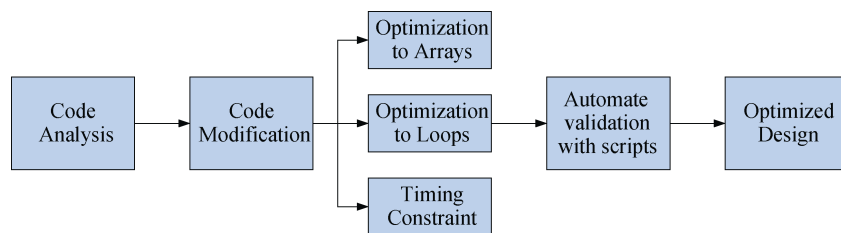


图 1 探索设计空间的流程

Figure 1 The process of exploring the design space

首先, 本论文对 C 语言代码的结构进行详细的分析, 寻找可优化的位置。接着, HLS 支持 RTL 综合, 同时可对 C 语言编写的模块进行验证。并且 HLS 可以自动生成 Verilog 代码和 IP 核。与人工完成 C 语言代码到 Verilog 代码的映射相比, HLS 为早期阶段的探索提供了一种即方便又快捷的方式。同时这也是一种实现和优化 PQC 算法硬件设计的良好尝试, 为本文提供了对比分析的基础指标。然而, 一些 C 语言结构不被 HLS 支持^[15-16], 所以本论文修改了原始的 Kyber 代码以适应 HLS 的规则。最后, 依托于基准方案(例, 未进行优化), 本论文考虑了三种不同方式的优化方案。HLS 提供对数组(或接口)和循环结构的优化。对不同的循环结构可应用不同的优化策略, 来达到更好的效果。另外, 时序约束条件也是 RTL 设计中的一个重要因素。本文中也展示出不同时序

约束条件下设计方案的探索 and 对比。基于上述分析, 本论文编写了脚本程序, 自动对得到的结果进行验证, 来获得最终的优化设计方案。

3.1 C 语言代码分析

为了完善后续的设计空间探索, Kyber 提供的算法^[12]和 C 语言代码需要被分析。其中, 本论文关注循环结构进和接口。

与 KEM 的主函数一致, Kyber 也拥有三个主函数: `crypto_kem_keypair`, `crypto_kem_enc`, `crypto_kem_dec`。图 2~3 分别展示了 Kyber-512 的三个主函数及循环结构的调用图示。其他参数集的 Kyber 算法, 调用关系基本与 Kyber-512 一致。红色块代表两个主函数, 黄色块代表函数以及指示它来自哪个文件, 蓝色块代表循环结构。为简化调用关系图, 重复的函数用灰色块表示, 且不再展开。水平箭头指向函

**Table 1** The list of third round candidates

在分析代码结构后,本文发现在函数 `keccak_absorb`、`keccak_squeezeblocks` 和 `gen_matrix` 中有多个复杂循环,它们之间也有调用关系。因此,我们将重点关注这一部分的优化策略选择。这些复杂循环也是导致高延迟的重要循环。

表 2 Kyber-512 的输入输出接口及其长度

Table 2 Input and output interfaces and their lengths of Kyber-512

Function	Input / Size for Kyber-512, 768, 1024 (bytes)	Output / Size for Kyber-512, 768, 1024 (bytes)
crypto_kem_keypair		pk / 800, 1184, 1568
		sk / 1632, 2400, 3168
crypto_kem_enc	pk / 800, 1184, 1568	ct / 736, 1088, 1568
		ss / 32, 32, 32
crypto_kem_dec	sk / 1632, 2400, 3168	ss / 32, 32, 32
	ct / 736, 1088, 1568	

表 3 HLS 中对于数组的优化方法

Table 3 Optimization methods for arrays in HLS

优化方法		具体阐述
数组分割		将数组分割为多个块, 用以增加输入输出的并行度, 降低延迟。
数组映射	数组映射	将多个小数组合并为一个数组, 可以减少使用的资源。
	横向	横向映射将多个数组直接连接成一个更长的数组。
	纵向	纵向映射对于不同数组同一位的元素进行位拼接。
数组重组		数组分割和纵向映射的结合, 分割数组后进行位拼接。可以降低资源消耗, 提高并行度。

表 4 HLS 中循环优化策略

Table 4 Optimization strategy for loops in HLS

循环优化策略	具体阐述
循环流水	通过允许并行执行循环迭代, 大大缩短了初始化间隔。下一次迭代不需要等到当前迭代运算结束就可以开始。通过流水线, 空闲模块可以被下一次迭代调用。
循环展开	将相同电路复制多份, 多次迭代可以同时独立进行。

表 5 Kyber 中循环的分类

Table 5 Loops in Kyber

简单循环	内部没有其他函数被调用, 或者循环内被调用的函数内部不包含其他循环。
复杂循环	这类循环中会调用包含循环的其他函数。

循环优化的策略选择是本文的讨论重点。通过对优化策略的解读和循环分析, 我们得到以下优化规则, 并根据这些规则指导实际的优化:

首先, Kyber 算法中有很多单层简单循环, 它们的功能是进行简单的算术运算和数组赋值。

对于数组边界小的这些循环, 以及 Kyber 中频繁调用的函数, 使用循环展开、创造多个独立操作将得到较好的优化效果。虽然会消耗电路面积, 但对这些循环进行完全展开会取得更好的综合效果。例如, *store64* 和 *load64* 的循环边界是 8, 它们适合使用循环展开来减少延迟。

但是, 我们认为在以下情况中, 流水线会在电路面积和延迟之间实现更好的权衡: 当数组边界增加时, 比如 *sha3_256* 函数的边界为 64; 或者函数中的循环较少调用, 例如 *pack_pk* 和 *unpack_pk*; 或者循环包含非常复杂的算数或赋值运算, 例如 *KeccakF600_StatePermute*。本文认为这些情况更适合

使用流水线。

此外, 有一些循环的内部计算过于复杂, 很少被调用。我们认为最好不要优化这种类型的循环。它可能会花费太多不值得的硬件成本。

其次, 对于嵌套的简单循环, 流水线优化对象本质上是外部循环和内部循环或函数。外层的流水线导致内层的展开, 这将大大增加电路面积, 例如 *cbd* 函数。因此, 内部循环的流水线优化将消耗更少的硬件资源。外层循环的流水线优化将显著增加需要调度的操作数量, 从而需要更多的硬件资源, 但它将导致延迟方面的性能更高的设计。

复杂循环和其他函数之间有调用关系。和嵌套简单循环相似, 如果对其进行流水线优化, 会出现内部函数完全展开的情况, 造成硬件资源的浪费。同时, 被调用函数内部的循环也可以被优化。

所以如何权衡以获得最佳效果, 是我们的研究重点。我们将在实验部分展示我们的研究结果。

3.5 时序约束

时序约束的设置会影响 HLS 的设计。为了使得电路的频率更高, 往往会付出延迟和面积开销增加的代价。文献[14]分析了 NIST 的 7 种密钥封装算法和 4 种签名算法的 HLS 实现。而在文献[14]提出的方案中, Kyber 的 HLS 设计选择了的时钟周期为 15 ns。由于频率与时钟周期有关, 我们在优化中尝试不同的时序约束来分析性能。

3.6 用脚本自动验证

通过以上分析, 我们得到了选择优化策略的基本原则。为确保结果正确, 还需要人工验证和数据比对。这个过程需要大量的操作和分析。由于 HLS 支持 tcl 脚本控制, 为了简化其验证过程并扩展到其他 PQC 算法, 我们编写了 tcl-perl 协同脚本程序来实现相应的操作。

该程序包含如表 6 所示的以下功能:

表 6 tcl-perl 协同脚本程序功能
Table 6 The function of tcl-perl collaborative script program

功能	具体阐述
C 代码分析	包括函数和循环体识别。在循环体中添加标签, 方便后续循环优化的执行。
脚本文件生成	由于 HLS 在执行优化选项时需要执行特定规范的 tcl 脚本, 因此根据第一步中确定的函数和标签以及不同的优化策略生成相应的脚本文件。
驱动 HLS 程序运行	首先自动打开 HLS 程序并驱动其仿真过程, 然后为每个优化策略分别保存其结果。在该步中, 可设置目标硬件平台、顶层功能和时序约束。在程序运行过程中, 偶尔会出现由于优化操作过于复杂, HLS 无法安排正常调度的情况。它会导致程序无法终止。我们设置了超时机制来解决此问题。
获取仿真结果报告	读取所有结果的关键信息, 包括硬件资源(BRAM、DSP、FF、LUT)、延迟和时钟拍数。且数据可以以表格形式存储, 便于比较。
可扩展功能	可以应用于任何其他 PQC 算法的 HLS 探索。只需要修改 C 代码, 去掉 HLS 不支持的操作, 就可以通过 HLS 的仿真验证。未来, 我们还将探索其他 PQC 算法的优化空间。

4 结果分析

本文实验使用了 Vivado HLS 2018.2 工具, 并以 Virtex-7 VC709 作为目标硬件平台。除讨论时钟周期的影响的实验, 其他设计的时钟均设置为 10 ns。

同时, 本文将 Kyber-512 作为主要分析对象, 对各种优化策略进行单独分析。其他参数集的优化设计将在最后独立给出并进行比较。

对于性能指标, 我们有以下几点考虑: 首先, 对于硬件设计, 我们需要考虑它的性能和硬件资源消耗。时钟拍数可以用来衡量性能, FF 和 LUT 的数量可以用来衡量硬件资源开销。此外, 性能和资源之间的均衡性是一个更有意义的指标。根据已有工作, 我们可以采用时延面积乘积 $LAP = (FF+LUT) \times clock\ cycles$ 作为一个度量。LAP 越小, 性能越好。在循环优化中, 我们加入了不同的时钟约束, 然而 $clock\ cycle$ 不能完全确定在不同时钟约束下所需的计算时间。因此本文进一步使用总运行时间与面积乘积 $TAP=LAP/f_{clk}(\text{时钟频率})=LAP \times clk(\text{时钟周期})$ 的方式对电路性能/资源的均衡性进行度量。

4.1 数组优化

由于三个顶层函数的优化结果有相同的趋势,

在这里以 `crypto_kem_enc` 函数为例进行说明。

如图 4 所示, 对一些被其他函数调用的内部模块的接口进行数组分割。而其他优化策略增加了外部端口的端口数或端口位宽(端口是整个模块的输入和输出), 在图 4 中, 横坐标展示了不同的优化策略。柱状图的值对应左边的纵坐标, 展示触发器、查找表的数量以及延迟。折线图对应右边的纵坐标, 展示延迟-面积乘积 LAP。

对于内部模块的局部接口, 数组分割能通过增大硬件资源开销降低延迟, 从而获得正向优化的效果。但是, 这带来的正向优化效果非常小, 只有 0.128%。

同时, 如果端口的优化增大硬件资源开销, 却没有相应地降低延迟, 将带来负向影响。这也会导致 LAP 的增加。其中, 当端口数量增加到 6 个以上时, LAP 增加得最多, 达到 67.2%。因此, 端口数量越多, LAP 增加越多。

从结果来看, 数组的优化无法达到理想的优化效果。所以, 在最终的优化设计中将不再考虑数组的优化。

4.2 对循环结构的优化

本论文在这里展示循环结构优化后, 三个顶层函数的结果和分析。

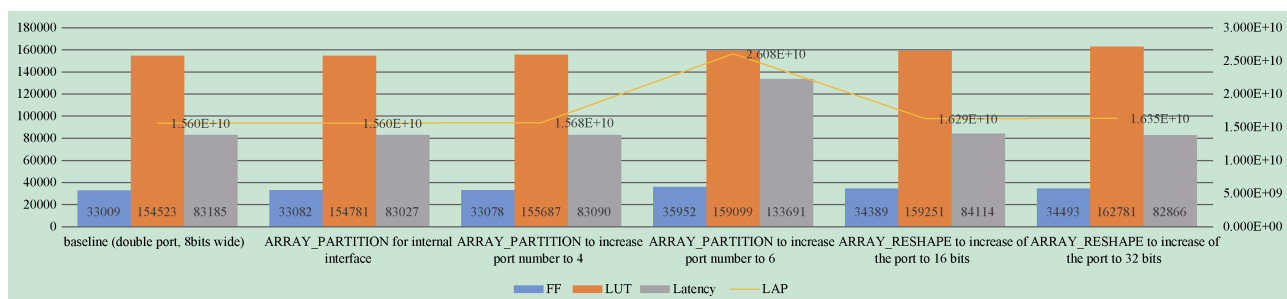


图 4 Kyber-512 接口和端口的优化(时钟周期为 10ns)

Figure 4 Optimization for interfaces and ports of Kyber-512 with clk=10ns

首先, 如图 5~7 所示, 对于时钟周期为 10ns 条件下的三个函数, 简单循环和复杂循环都获得了正向优化。相较于未优化方案, 密钥生成算法的简单循环和复杂循环优化方案 LAP 优化效率分别达到 41.3% 和 16.5%。对于封装算法, LAP 优化效率分别达到 45.5% 和 26.9%。对于解封装算法, LAP 的优化效率分别达到 41.8% 和 19.7%。这意味着, 经过对循环结构进行相关优化, 模块的性能可得到大幅提升。

表 7~8 展示了对三种循环结构添加循环流水和循环展开的数量。

其中, 对简单单层循环结构的优化结果符合由

代码结构分析得到的规律。

对于简单嵌套的循环结构, 优化规律基本和简单单层循环结构的一致。在对 Kyber 算法某个特定情况的探索过程中, 本论文发现, 对一个嵌套的简单循环结构中的最外层(Kyber 算法的最复杂嵌套结构包含 3 层)或一个子循环中的循环使用循环流水, 可以获得更好的效果增长, 例如 polyvec_compress 函数。这种嵌套的循环结构的特征是, 最内层的循环结构通常是一种简单的运算或赋值操作, 同时拥有较小的循环边界。这和本文对简单单层循环的分析一致。

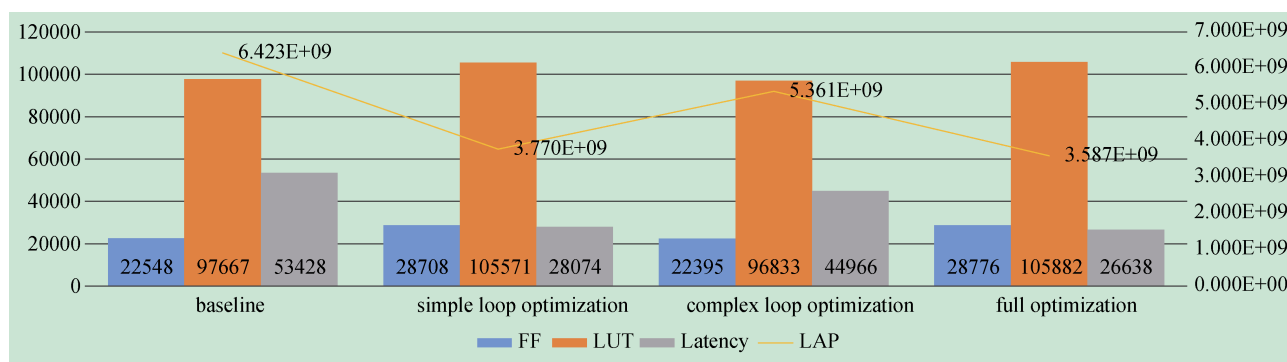


图 5 时钟周期 10ns 下, 对 Kyber-512 密钥生成算法的循环结构优化

Figure 5 Optimization for loops of key generation algorithm of Kyber-512 with clk=10ns

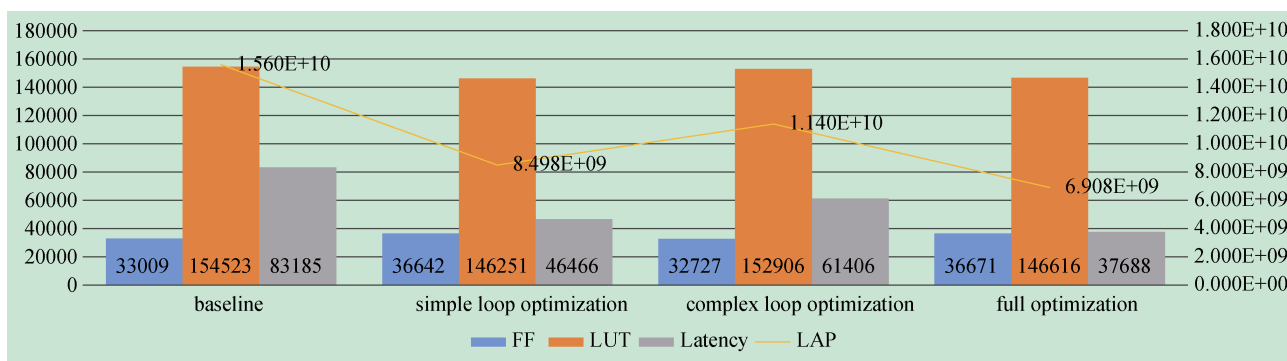


图 6 时钟周期 10ns 下, 对 Kyber-512 密钥封装算法的循环结构优化

Figure 6 Optimization for loops of encapsulation algorithm of Kyber-512 with clk=10ns

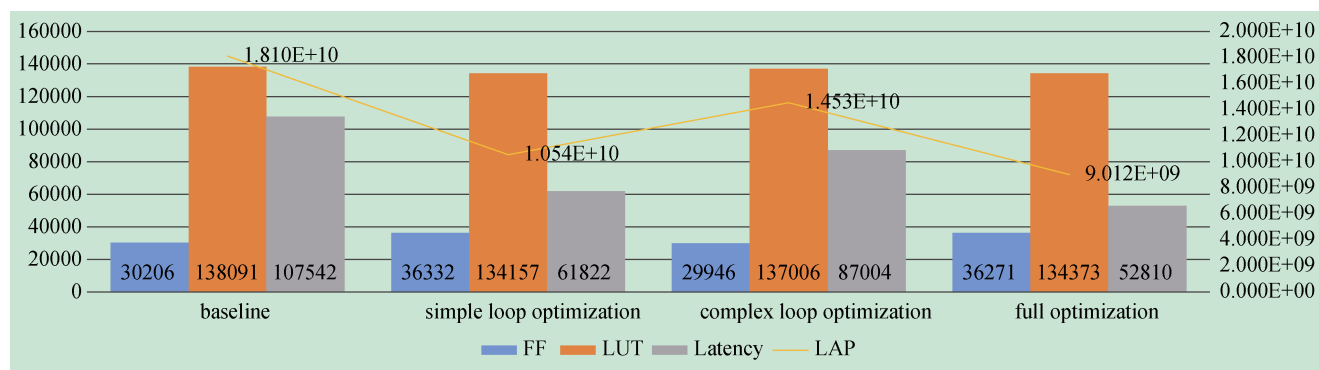


图 7 时钟周期 10ns 下, 对 Kyber-512 密钥解封装算法的循环结构优化

Figure 7 Optimization for loops of decapsulation algorithm of Kyber-512 with clk=10ns

表 7 对简单循环结构添加的循环流水与循环展开的数量

Table 7 The number of pipelining and unrolling added to simple loops

Function	Total number of loops	Unrolling	Pipelining
crypto_kem_keypair	26	5	18
crypto_kem_enc	34	10	19
crypto_kem_dec	45	13	26

表 8 对复杂循环结构添加的循环流水与循环展开的数量

Table 8 The number of pipelining and unrolling added to complex loop

Function	Total number of loops	Unrolling	Pipelining
crypto_kem_keypair	16	2	3
crypto_kem_enc	18	3	3
crypto_kem_dec	18	3	3

因此, 对大多数简单循环, 从过往分析中得到的规律是能够适用的。根据 Kyber 算法的特征, 多数循环结构都适合使用循环流水。循环流水与循环展开的应用比例为 2:1。对简单循环的优化可带来良好效果。

通过对复杂循环的探索, 本论文发现其优化策略的选择大致上与简单循环的选择一致。对于操作简单的内层调用函数, 外层复杂循环可以根据简单循环的分析选择合适的优化, 叠加对内部函数的优化。然而, 如果内层调用函数和其循环结构是复杂的, 例如 KeccakF600_StatePermute, 或者内层函数与其他函数也有复杂的后续调用关系, 例如 polyvec_pointwise_acc_montgomery, 此时不推荐对外层循环进行优化。例如, 对于 keccak_absorb 的循环 2 和循环 3, 循环 2 不做优化, 循环 3 使用循环流水才能达到最佳效果。

因此, 对于复杂循环, 因其复杂的结构, HLS 对

它们的调度和安排能力有限, 可选优化策略的范围很小。正如之前提到的 gen_matrix 函数的优化, HLS 不能提供恰当的调度策略, 因为 keccak_absorb 函数和 keccak_squeezeblocks 函数包含大量的复杂循环和后续调用。所以对 gen_matrix 函数不做优化是最好的选择。

同时, 与简单循环的优化相比, 循环展开的比例增加了。因为循环流水将展开所有低于当前循环级别的函数和循环。随着低级函数或循环数量的增加, 硬件资源会大量消耗。虽然可以优化的复杂循环数量很少, 但优化的部分都提供了显著的性能改进。硬件资源和延时都低于基准方案。

此外, 通过组合所有优化, 即完全优化, LAP 达到最低值。与基准方案相比, LAP 的优化效率可分别达到 44.2%、55.7%和 50.2%。硬件资源消耗略有上下波动, 但延时明显降低。

4.3 时序约束的影响

此外, 本论文还考虑了时钟的影响。文献[14]同样使用 Virtex-7 作为目标硬件平台, 它提供了在时钟周期为 15ns 时封装算法和解封装算法的基准数据, 对封装算法采用流水线策略优化和展开策略优化的数据, 以及对解封装算法采用展开策略优化的数据。在我们的实验中, 本论文尝试了不同的时序约束, 时钟周期设置包括 10ns 和 7.16ns 两种。

如图 8~10 所示, 随着时钟拍数的减少, 硬件消耗和时钟周期都有不同幅度的增加, 从而导致 LAP 增加。但是, 模块计算所需的总时间面积乘积 TAP 大大减少。对于密钥生成算法, 与时钟周期为 10ns 的基准实现相比, 时钟周期为 7.16ns 的基准实现的 TAP 降低了 17.9%。而对于封装和解封装算法, 分别下降了 15.3% 和 12.2%。此外, 与时钟周期为 10ns 相比, 三种算法的时钟周期设置为 7.16ns, 且采用全部优化的实现, TAP 分别降低了 17.2%、17.0%和

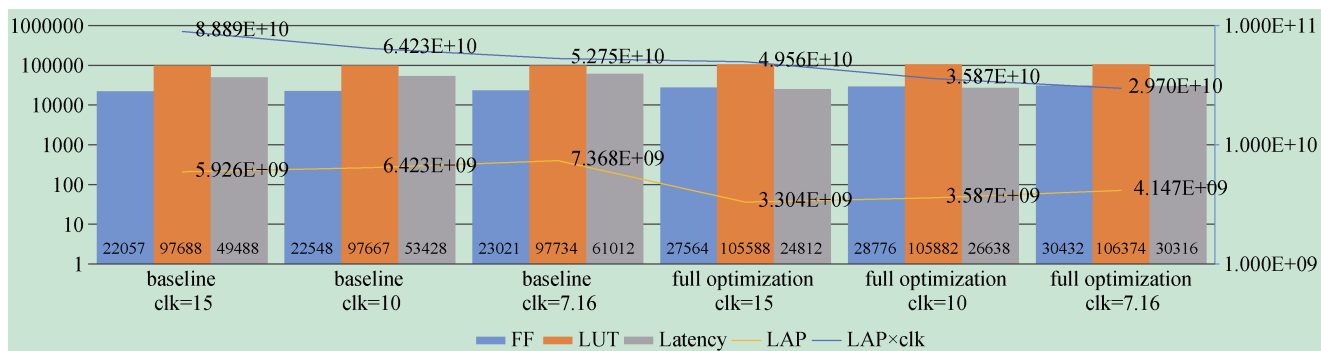


图 8 不同时钟周期下密钥生成算法的优化

Figure 8 Optimization for key generation algorithm with different clock

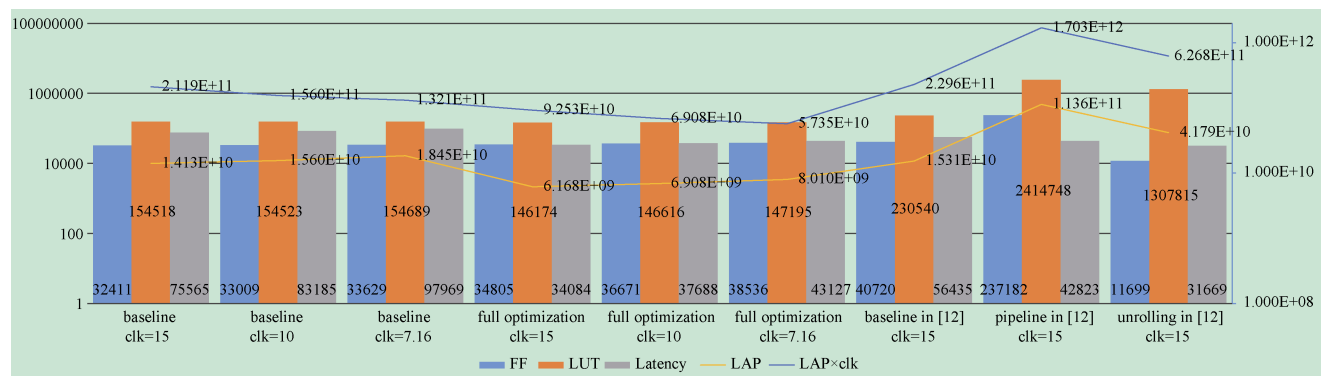


图 9 不同时钟周期下密钥封装算法的优化

Figure 9 Optimization for encapsulation algorithm with different clock

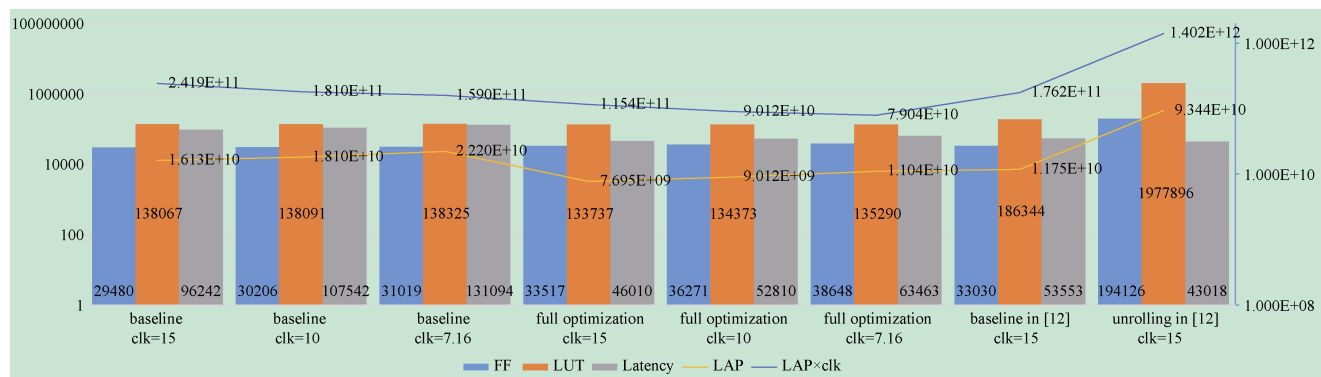


图 10 不同时钟周期下密钥解封算法的优化

Figure 10 Optimization for decapsulation algorithm with different clock

12.3%。因此,在时序约束中降低时钟周期可以进一步优化设计。在这个设计中它最低可以降至 7.16ns。

与文献[14]相比,无论采用哪种时钟周期和优化策略设置,本论文的实验结果都有更明显的优势。对于封装算法,文献[14]中的 LAP 和 TAP 最优结果对应的是基准设计。与文献[14]的最优结果相比,当时钟周期为 15ns 时,我们无优化的设计 LAP 降低了 7.7%。在时钟周期为 15ns 的设置下,采用全面优化策略的设计, LAP 降低了 59.7%。当时钟周期为 7.16ns 时,完全优化的设计 TAP 降低了 75%。对于

解封算法,文献[14]的最优结果也对应基准设计。而在本论文中对时钟周期为 15ns 的全面优化中, LAP 降低了 34.5%。当时钟周期为 7.16ns 时,完全优化的 TAP 降低了 55.1%。可以看到,我们的优化取得了显着的提高。

4.4 不同参数集的优化设计

图 11~13 展示了不同参数集的设计和优化结果。此处设置的时序约束为:时钟周期取 7.16ns 以获得最佳结果。从结果可以看出,不同参数集下的算法基本使用相同的模块,因此每个参数集之间的硬

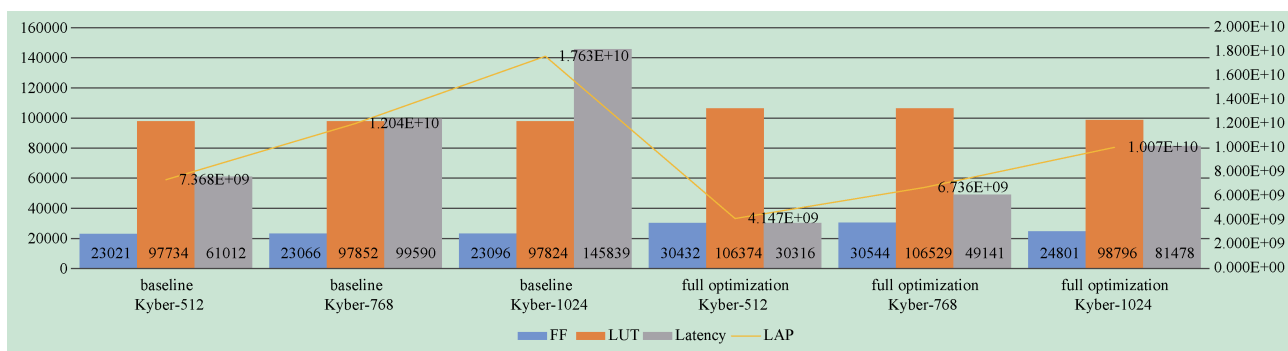


图 11 不同参数集的密钥生成算法设计

Figure 11 Design for key generation algorithm with different parameter sets

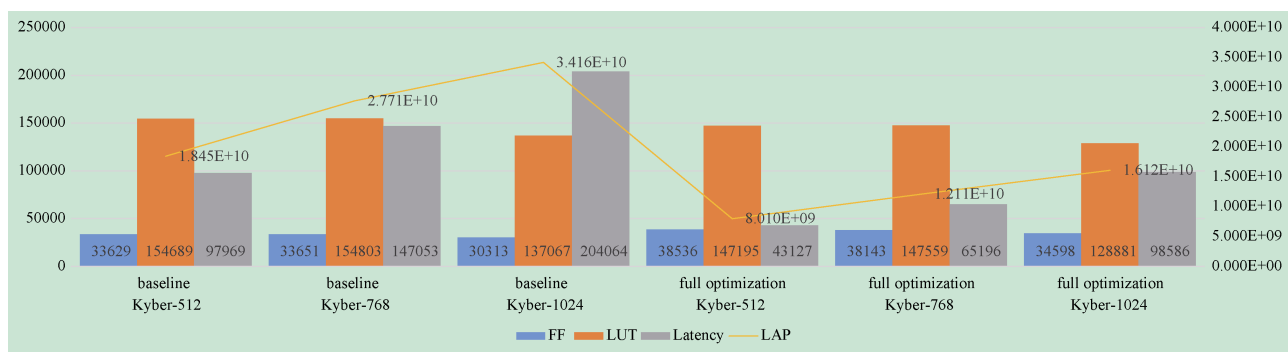


图 12 不同参数集的密钥封装算法设计

Figure 12 Design for encapsulation algorithm with different parameter sets

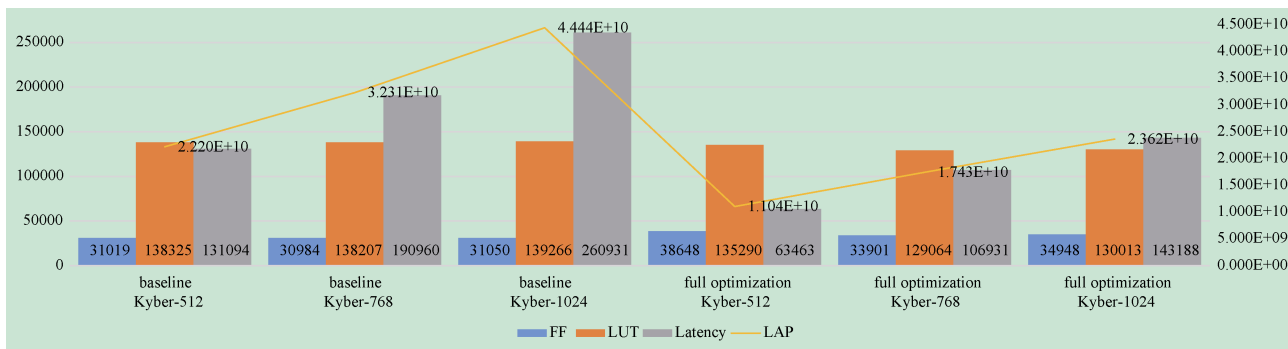


图 13 不同参数集的密钥解封算法设计

Figure 13 Design for decapsulation algorithm with different parameter sets

件资源消耗差异不大。但是, 由于要计算的数据量不同, 这导致总的时钟拍数变化显著, 时钟拍数与表 2 所展示的输入和输出长度呈线性增长趋势。

对于优化后的设计, 三个参数集采用了大致相同的优化策略。在极少数模块中, 由于处理的数据量不同, 实验中需要扩大循环边界或扩大计算规模, 从而影响优化策略的选择。最终, 对于 Kyber-512, 三种算法的优化效率分别为 43.7%、56.6% 和 50.3%。Kyber-768 的优化效率分别达到了 44.1%、56.3% 和 46.1%。而对于 Kyber-1024, 优化效率分别达到了 42.9%、52.8% 和 46.8%。

4.5 与已有工作的对比

表 9 展示了本文与在不同平台上实现的 Kyber512 的运行时钟周期数的对比。实验结果显示: 本文的 HLS 实现与硬件设计^[17]相比, 性能相近; 而本文的 HLS 实现与软件实现^[18-19]相比, 具有明显的性能优势。

表 10~11 展示了对于 Kyber512 的密钥封装模块和密钥解封模块的 HLS 实现对比。表 10~11 中的本文数据为采取完全优化策略且时钟周期 clk 选取 7.16ns 的数据。实验结果显示, 相比已有 HLS 实现^[14], 本文具有更小的硬件开销, 更短的运行时间。

表 9 在不同平台上实现的 Kyber512 的运行时钟周期对比
Table 9 Clock cycles of Kyber512 implemented on different platforms

Implementation platforms	Enc [cycles]	Dec [cycles]	Enc Time Reduction Percentage [%]	Dec Time Reduction Percentage [%]
This work	43127	63463	—	—
FPGA ^[18]	49015	68815	12.0	7.8
Haswell ^[19]	161440	190206	73.3	66.6
Cortex-M4 ^[20]	634000	597000	93.2	89.4

表 10 Kyber512 密钥封装模块 HLS 实现硬件资源开销及性能对比

Table 10 Comparison of hardware resource overhead and performance of Kyber512 Encapsulation module

	LUTs	FFs	Time(ms)	TAP
This work	147195	38536	0.309	57353
文献[14]	1307825	11699	0.475	626773

表 11 Kyber512 密钥解封装模块 HLS 实现硬件资源开销及性能对比

Table 11 Comparison of hardware resource overhead and performance of Kyber512 Decapsulation module

	LUTs	FFs	Time(ms)	TAP
This work	135290	38648	0.454	79037
文献[14]	1977896	194126	0.645	1401605

5 总结

目前, PQC 算法的硬件实现是一个崭新的、具有重大意义的研究领域。HLS 为探索 PQC 算法的硬件设计提供了方便快捷的工具。为了对 PQC KEM 算法 CRYSTALS-KYBER 进行研究, 本文对其硬件设计的优化空间进行了探索。本文对 Kyber 的软件架构进行了详细分析。此外, 本文编写了 tcl-perl 协同脚本以使得探索过程能够自动化。本文使用 Virtex-7 VC709 作为目标硬件平台。实验结果表明, 循环流水或循环展开的方法可以提高 Kyber 中循环的性能, 并且能够帮助实现硬件资源和速度之间的有效均衡。实验结果表明不同的循环适用不同的优化策略。其中, 在 Kyber-512 中, 三个算法模块的最优化设计的优化效率分别可以达到 44.2%、55.7% 和 50.2%。与已有工作^[14]相比, 本文的优化取得了积极的效果。封装模块的运行总时间与面积乘积 TAP 降低了 75%。解封装模块的运行总时间与面积乘积 TAP 减少了 55.1%。实验结果显示对于 Kyber 的不同参数集, 最优优化策略基本一致, 优化效率保持在同一水平。在未来的工作中, 一方面计划进一步探

索后量子加密算法的硬件实现, 并针对其中一些关键模块提出具体的硬件优化方案, 以实现效率更高的后量子加密算法专用电路; 另一方面, 对于后量子密码算法实现, 是否能够抵抗侧信道攻击也是一个重要的研究问题, 计划将对 Kyber 算法实现的侧信道攻防问题展开研究。

致 谢 本文得到了国家自然科学基金(NSFC)区域创新发展联合基金(No. U20A20202)的支持。通讯作者为叶靖。

参考文献

- [1] Chen L, Jordan S, Liu Y K, et al. Report on Post-Quantum Cryptography[R]. National Institute of Standards and Technology, 2016.
 - [2] Shor P W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer[J]. *SIAM Review*, 1999, 41(2): 303-332.
 - [3] Long G L. Grover Algorithm with Zero Theoretical Failure Rate[J]. *Physical Review A*, 2001, 64(2): 022307.
 - [4] Ladd T D, Jelezko F, Laflamme R, et al. Quantum Computers[J]. *Nature*, 2010, 464(7285): 45-53.
 - [5] Mailloux L O, Lewis II C D, Riggs C, et al. Post-Quantum Cryptography: What Advancements In Quantum Computing Mean for IT Professionals[J]. *IT Professional*, 2016, 18(5): 42-47.
 - [6] Micciancio D, Regev O. Lattice-Based Cryptography[M]. Post-Quantum Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019: 147-191.
 - [7] NIST. "Round-3-submissions". post-quantum-cryptography. 30 July. 2020: <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions>.
 - [8] Liu W Q, Fan S L, Khalid A, et al. Optimized Schoolbook Polynomial Multiplication for Compact Lattice-Based Cryptography on FPGA[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019, 27(10): 2459-2463.
 - [9] Rui K K, Wang C H, Fan S L, et al. High Performance Hardware Architecture of Lattice-Based Cryptography and Its FPGA Implementation[J]. *Journal of Data Acquisition and Processing*, 2019, 34(4): 689-696.
- (芮康康, 王成华, 范赛龙, 等. 一种高性能 R-LWE 格加密算法

- 的电路结构及其 FPGA 实现[J]. *数据采集与处理*, 2019, 34(4): 689-696.)
- [10] Bernstein D J, Lange T. Post-Quantum Cryptography[J]. *Nature*, 2017, 549(7671): 188-194.
- [11] NIST. Round-2-submissions. Post-Quantum Cryptography. 30 January. 2019: <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-2-submissions>.
- [12] R. Avanzi, J. Bos, et. al., CRYSTALS-Kyber algorithm specifications and supporting documentation(version 2.0). NIST PQC Round 2, 2019.
- [13] Bos J, Ducas L, Kiltz E, et al. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM[C]. *2018 IEEE European Symposium on Security and Privacy*, 2018: 353-367.
- [14] Basu K, Soni D, Nabeel M, et al. NIST Post-Quantum Cryptography-A Hardware Evaluation Study[J]. *IACR Cryptol. ePrint Arch.*, 2019, 2019: 47.
- [15] Vivado design suite user guide - High-Level synthesis. Xil-inx. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug902-vivado-high-level-synthesis.pdf. May.2021
- [16] Vivado Design Suite Tutorial - High-Level Synthesis. Xilinx. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug871-vivado-high-level-synthesis-tutorial.pdf. May. 2019.
- [17] Huang Y M, Huang M Q, Lei Z K, et al. A Pure Hardware Implementation of CRYSTALS-KYBER PQC Algorithm through Resource Reuse[J]. *IEICE Electronics Express*, 2020, 17(17): 20200234.
- [18] Avanzi R, Bos J, Ducas L, et al. CRYSTALS-Kyber algorithm specifications and supporting documentation[J]. *NIST PQC Round*, 2017, 2(4): 25-30.
- [19] Botros L, Kannwischer M J, Schwabe P. Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4[M]. *Progress in Cryptology – AFRICACRYPT 2019*. Cham: Springer International Publishing, 2019: 209-228.



穆嘉楠 于 2019 年在北京大学电子科学与技术专业获得学士学位。现在中科院计算所体系结构专业攻读硕士学位。研究领域为数字集成电路测试与安全。研究兴趣包括：后量子加密、隐私计算等。Email: mujianan19s@ict.ac.cn



赵艺璇 于 2019 年在北京邮电大学电信工程及管理专业获得学士学位。现在中科院计算所体系结构专业攻读硕士学位。研究领域为数字集成电路测试与安全。研究兴趣包括：后量子加密等。Email: zhaoyixuan19s@ict.ac.cn



宋金峰 于 2021 年在天津大学集成电路设计与集成系统专业获得学士学位。现在中科院计算所体系结构专业攻读硕士学位。研究领域为数字集成电路测试与安全。研究兴趣包括：后量子加密等。Email: songjinfeng21@mails.ucas.ac.cn



严寒 于 2021 年在天津大学集成电路设计与集成系统专业获得学士学位。现在中科院计算所体系结构专业攻读硕士学位。研究领域为数字集成电路测试与安全。研究兴趣包括：后量子加密等。Email: yanhan21@mails.ucas.ac.cn



叶靖 于 2014 年在中国科学院计算技术研究所系统结构专业获得博士学位。现任中科院计算所计算机体系结构国家重点实验室副研究员。研究领域为数字集成电路测试与安全。研究兴趣包括：DFT、ATPG、PUF、AI 安全等。Email: yejing@ict.ac.cn



李华伟 于 2001 年在中国科学院计算技术研究所获得博士学位。现任中科院计算所计算机体系结构国家重点实验室研究员。研究领域为集成电路设计自动化、智能计算、近似计算、容错计算、设计验证与测试。研究兴趣包括：集成电路设计自动化与智能计算。Email: lihuawei@ict.ac.cn



李晓维 于 1991 年在中国科学院计算技术研究所获得博士学位。现任中科院计算所计算机体系结构国家重点实验室研究员。研究领域为集成电路测试, EDA, 容错计算, 硬件安全。研究兴趣包括: VLSI 测试、容错计算、可靠设计、硬件安全等。Email: lxw@ict.ac.cn