软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.007389]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 FPGA 的格基数字签名算法硬件优化实现 *

胡跃 *赵旭阳 * 王威 袁谦 郑婕妤 杨亚芳

(复旦大学 计算机科学技术学院,上海 200433)

通讯作者: 赵旭阳, E-mail: xuyangzhao21@m.fudan.edu.cn

摘 要: 数字签名算法对于网络安全基础设施有至关重要的作用,目前的数字签名方案大多是基于 Rivest-Shamir-Adleman（RSA）和椭圆曲线密码学（ECC）实现的.随着量子计算技术的快速发展,基于传统的公钥密码体系的数字签名方案将面临安全性风险,研究和部署能够抵抗量子攻击的新型密码方案成为了重要的研究方向.经过多轮评估分析,美国国家标准研究院(National Institute of Standards and Technology, NIST)于 2024 年 8 月公布了后量子数字签名标准方案 ML-DSA,其核心算法是 Dilithium.本文针对格基数字签名算法 Dilithium 高维多项式矩阵运算的特点,基于 FPGA 平台提出了多种优化实现方法,具体包括可配置参数的多功能脉动阵列运算单元、专用型多项式并行采样模块、针对多参数集的可重构存储单元设计、针对复杂多模块的高并行度时序状态机,旨在突破性能瓶颈以实现更高的签名运算效率,并最终实现了可同时支持三种安全等级的数字签名硬件架构.本文的设计方案在 Xilinx Artix-7 FPGA 平台上进行了实际的部署和运行,并且和已有的同类型工作进行了对比.结果表明,与最新的文献相比,本文的设计方案在三种安全等级下的签名运算效率分别提升了 7.4 倍、8.3 倍和 5.6 倍,为抗量子安全的数字签名运算服务提供了性能基础,并且对于推进格密码方案的工程化和实用化进程提供了一定的借鉴意义和参考价值.

关键词: 后量子密码；格基密码；数字签名算法；FPGA；硬件实现;

中图法分类号: TP309.2

中文引用格式: 胡跃,赵旭阳,王威,袁谦,郑婕妤,杨亚芳. 基于 FPGA 的格基数字签名算法硬件优化实现. 软件学报. http://www.jos.org.cn/1000-9825/7389.htm

英文引用格式: Hu Y, Zhao XY, Wang W, Yuan Q, Zheng JY, Yang YF. Hardware Optimization Implementation of Lattice Digital Signature Algorithm Based on FPGA. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7389.htm

Hardware Optimization Implementation of Lattice Digital Signature Algorithm Based on FPGA

HU Yue, ZHAO Xu-Yang, WANG Wei, YUAN Qian, YANG Ya-Fang, ZHENG Jie-Yu, YANG Ya-Fang

(Fudan University, School of Computer Science, Shanghai 200433)

Abstract: Digital signature algorithms play a vital role in network security infrastructure. Most of the current digital signature schemes are based on RSA and ECC . With the rapid development of quantum computing technology, digital signature schemes based on traditional public key cryptography will face security risks. Researching and deploying new cryptographic schemes that can resist quantum attacks has become an important research direction. After several rounds of evaluation and analysis, NIST announced the post-quantum digital signature standard ML-DSA in August 2024, and its core algorithm is Dilithium. In view of the characteristics of the high-dimensional

* 基金项目:国家重点研发计划基金资助项目(No.2022YFB2701601); 上海市协同创新基金资助项目(No.XTCX-KJ-2023-54);上海市科委区块链关键技术攻关专项基金资助项目(No.23511100300)

*作者胡跃、作者赵旭阳对论文有相同的贡献，为本论文共同一作

收稿时间: 2024-06-30; 修改时间: 2024-09-05; 采用时间: 2024-12-30; jos 在线出版时间: 2025-01-20

polynomial matrix operation of Dilithum, this paper proposes a variety of optimization implementation methods based on the FPGA platform, including multifunctional systolic array operation units with configurable parameters, dedicated polynomial parallel sampling modules, reconfigurable storage unit design for multiple parameter sets, and high-parallel timing state machines for complex multi-modules, aiming to break through performance bottlenecks and achieve higher signature operation efficiency, and finally realize a digital signature hardware architecture that can support three security levels at the same time. Our hardware architecture was actually deployed on the Xilinx Artix-7 FPGA platform, and compared with existing similar works. The results show that, our design has improved the signature operation efficiency by 7.4 times, 8.3 times and 5.6 times at three security levels, respectively, which will provide the performance foundation for quantum-resistant digital signature services, and provide meaningful application value and reference significance for the relevant research about lattice cryptography engineering and practicality.

Key words: Post-quantum cryptography; Lattice-based cryptography; Digital signature algorithm; FPGA; Hardware implementation

公钥密码算法体系是保障现代社会信息安全的基石,RSA 和 ECC 作为其中的最重要的两种算法[1],基于其构建的密钥封装机制、数字签名协议和同态加密等高级密码学应用,可以保证通信双方核心数据要素的保密性、完整性和不可抵赖性,因此在云计算、车联网、区块链等新兴计算平台上得到了广泛的应用.随着量子计算技术的快速发展,传统的基于大数分解和离散对数的传统密码学方案将有可能被运行量子算法的计算机系统在多项式时间内破解,其中最著名的量子算法是基于傅里叶变换的 Shor 算法[2,3]和 Grover 提出的关于无序数据库的搜索算法[4].其中,Grover 算法常被用于对称算法(如 AES 算法)的分析和破解,而 Shor 算法主要用于攻击基于椭圆曲线构建的密码学方案.为了应对量子计算技术对现有的公钥密码基础设施的安全性威胁,后量子密码技术(Post-quantum cryptography, PQC)受到了学术界和工业界的广泛关注和重点研究[5,6].

美国国家标准技术研究院从 2016 年开始向全球征集后量子密码算法的标准化方案[5,7],共包括基于哈希(hash-based)[8]、基于编码(code-based)[9]、基于多变量(multivariate-based)[10]、基于同源(isogeny-based)[11]和基于格(lattice-based)[12]五种基础构造方法,由于基于格的技术路线可以更好的平衡安全性、公私钥尺寸和实现效率三方面的因素,因此得到了重点的研究和关注.根据 NIST 在 2022 年 7 月发布的后量子标准项目的评选结果,第一批的 4 个标准化算法中有 3 个为格基密码方案,其中 CRYSTALS-Kyber[13]被确定为密钥封装机制的标准化方案,CRYSTALS-Dilithium[14]和 Falcon[15]为数字签名算法.在最新的研究报告中,NIST 官方在 ARM-CortexM4 平台上对两种数字签名算法进行了全方位的性能对比[16],由于 Falcon 的方案设计中存在大量的浮点运算,这使得签名方案的实例化部署过程更加复杂.结果表明 Dilithium 的签名运算效率要远高于 Falcon,并且占用的存储资源更少.从安全性层面来看,Dilithium 方案可以使用唯一的公钥和明文信息对应给定的签名,即签名结果存在强不可伪造性.因此 NSIT 官方在公布的报告中明确指出,推荐 Dilithium 为后量子数字签名算法的标准化方案,最终的标准化报告于 2024 年 8 月份发布[17], 并将其命名为为数字签名标准 ML-DSA.

Dilithium 是一种经典的基于格理论的数字签名方案,其安全性基于 MLWE(Module learning with errors)问题和 MSIS(Module short integer solution)问题[18].Dilithium 的大多数操作为多项式环上的乘法运算,可以使用数论变换（Number-Theoretic Transform,NTT）技术实现运算过程的加速,这样的设计结构非常适合软硬件平台进行优化实现和实际部署,因此具备了较好的工程化和实用化前景.此外,Dilithium 方案采用了 Fiat-Shamir with Aborts 结构,签名过程包括一系列条件检查和拒绝运算,这个过程确保生成的签名不泄露任何私钥信息.考虑到量子计算技术的快速发展以及数字签名技术在网络通信安全过程中发挥着不可替代的作用,同时类似于区块链和云计算这种新型的计算平台对于算法部署的灵活性和安全标准的多样性有较高的要求,因此设计并实现专用化、可重构、轻耦合的后量子数字签名算法集成电路,从而在多维应用场景下为核心数据提供高性能的、抗量子安全的身份验证、数据完整性保护、数据抗抵赖性、认证授权和隐私保护等服务,成为 PQC 领域一个重要的研究方向.本文以 Artix-7 系列 FPGA 平台为基础提出一种新的针对 Dilithium 算法最新参数集的完整硬件实现方案,通过对功能模块进行针对性的优化,并充分利用 FPGA 模块级并行的特点进一步突破签名运算的性能瓶颈,旨在为 Dilithium 算法的实用化进程提供有价值的硬件设计参考方案,具体工作如下：

（1）针对签名算法运算数据量较大这一特点,本文设计了专用化的高吞吐量脉动阵列单元作为多项式运

算模块的核心,将数论变换的八级运算以硬件资源的形式进行固化,并统一实现八并行的 NTT 运算、INTT 运算、点乘运算和模加运算.这样的设计方式不仅极大提高了关键过程的运算效率,并且将不同类型运算之间的时序差异进行规整化,有利于整体的流水线运算架构的设计与优化.

(2)研究并分析签名运算中的多项式系数特点和采样运算类型,并针对性的优化 Keccak 核内部的寄存器结构,进而提出了通用型和采样型两种哈希运算模块,在提高采样运算并行度的前提下有效节约了硬件资源消耗,同时可获得更高的时钟频率.

(3)针对三种安全等级下的签名算法参数集,研究并分析其中的多项式系数类型和向量存储深度的变化规律,基于最高安全等级对应的参数内容进行存储阵列的设计与实现,并根据顶层的片选信号实现存储单元的重构,在满足多维度存储需求的条件下最大化的提升了硬件资源的利用率.

(4)深入分析核心运算步骤对应的时钟周期消耗数量,同时充分考虑脉动阵列运算单元的高吞吐率的运行特点,在此基础上对签名算法的关键运算步骤进行细粒度的分解和重组,以向量为尺度实现核心运算功能,进而提出了全新的分段式时序状态控制逻辑设计方案,进一步提高了算法的整体运行效率.

1 签名算法软硬件实现相关工作

基于各种软硬件平台进行算法的优化实现是密码工程领域的一个重要研究方向,这对于推动后量子密码算法的工程化应用和实例化部署具有重要的现实意义.软件优化实现一般通过SIMD指令集对格密码系统中的复杂且耗时的运算过程进行并行加速,主要包括基于Intel处理器的AVX2/AVX 512指令集和基于ARM处理器的 NEON 指令集两大类.由于在后量子密码标准征集过程中,AVX2 是各提交方案的必要参考实现之一,同时NIST 也将 ARMv7 架构下的 ARM Cortex-M4 处理器选定为了后量子密码标准化竞赛的性能评估设备之一[19],因此,目前已有较多关于 Dilithium 签名算法的纯软件优化方案[20,21].考虑到指令集仅可以实现寄存器层面的加速运算,这阻碍了软件实现方案在算法整体性能上的进一步提升.基于可编程的基础逻辑资源设计的纯硬件方案能够实现更高的运算性能，对于推动 Dilithium 签名算法的实用化进程具有重要作用，同样是后量子密码工程领域的重要研究方向之一[22-27].此外，软硬件结合方案的核心思路是将签名算法中较为耗时的运算过程以硬件加速器的形式实现,软件部分主要负责数据调度和逻辑处理,综合考虑了软件实现的灵活性和硬件实现的高效性,因此也得到了较多的研究和关注[30-35].其中,Banerjee 等人[30]基于 RSIC-V 处理器首次实现了包括 Dilithium 在内的第二轮评选的所有候选算法,是一种较为通用的设计方案,与基于 ARM 平台实现的签名算法相比在性能上有较大的提升.Zhao 等人[32]同样基于RSIC-V处理器提出了一种新的面向MLWE-LBC的高性能设计方案,论文综合考虑了RLWE结构中数据层面的并行化和MLWE结构中指令层面的并行化,以Xin等人[33]提出的向量化处理器的思路为基础设计高并行的 NTT 处理器架构,进一步提升了签名运算的实现效率.Wang 等人[25,34]首先基于 Zynq-7000 平台提出了一种针对于 Dilithium 算法的纯硬件设计方案,并以此为基础提出了基于 PolarFir 平台的可同时支持 Dilithium 和 Kyber 两种算法的设计方案.根据文献给出的测试结果,随着硬件化程度的提高,整体的实现性能也随之提升.

相比于纯软件和软硬结合两种实现方式,Dilithium 算法的纯硬件实现的研究相对较少.Ricci 等人[23]基于VHDL 语言第一次完整实现了第二轮的 Dilithium 签名算法,利用并行化的设计思想有效提升了算法的运行效率.Land 等人[24]基于 Xilinx Artix-7 平台提出了一种高效紧凑的第三轮 Dilithium 算法的纯硬件设计方案,可同时支持三组参数集.上文介绍软硬结合设计时提到,Wang 等人[25]提出了一种基于 Zynq-7000 平台的专用型 Dilithium 算法的实现方案,考虑到文献提出的设计思路中，签名算法的所有核心运算过程均以硬件模块的方式实现,软件部分仅负责 packing/unpacking 操作以及输入输出端的使能信号,因此也可视为一种纯硬件实现方案.由于硬件实现涵盖了签名算法的所有流程,因此与软硬结合类的设计方案相比,最终的实现性能也得到了较大的提升.Zhao 等人[26]提出一种新的针对第三轮 Dilithium 算法的紧凑、高性能的硬件架构,首先针对性的设计了多个功能模块,并且提出了一种高效的流水线运算模式,与已有的纯硬件设计方案相比,在运算效率和资源消耗两个方面均获得了最优的结果.

此外,考虑到在格基密码方案中多项式乘法运算过程较为复杂且耗时,基于 FPGA 平台设计专用的多项式运算加速引擎同样是硬件实现的重要研究方向之一.其中 Mert 等人[36,37,38]提出了一系列的专用 NTT 运算模块的设计方案,其核心思想是通过设计无数据冲突的访存逻辑确保运算的正确性,进而提升 NTT 运算的并行化程度.Zhao 等人[39]通过深入研究蝴蝶运算中的操作数变化规律,确定了模数 q 对应的数据路径上所有寄存器的位宽上限,从而设计了一种紧凑型低延迟的蝴蝶运算单元.在此基础上,作者设计了三种并行模式的数论变换模块并进行对比测试,结果显示与单结构相比,二并行和四并行结构在运算效率增加两倍和四倍的同时,硬件资源仅提高了 1.45 倍和 2.58 倍,主要原因是在例化过程中存在资源复用的情况,这对于高性能运算模块的设计提供了有意义的参考.Xing 等人[40]提出了一种新的蝴蝶运算单元的设计思路,其核心思想可概括为: 将格密码方案中的多种不同类型的运算过程抽象化为一系列基础运算的排列组合,进而将上述运算过程在硬件层面同构设计成为一个复合功能模块.这种设计模式能够有效提升硬件资源的利用率,在 Hu 和 Zhao 等人[41,32]工作中也体现出类似的设计思路.需要注意的是,这种高度同构的设计模式可能会造成流水线中的运算级延迟时间变长,从而影响整个模块的最高频率,因此应根据具体的应用需求综合考虑设计架构.除了上述经典的迭代结构的多项式运算模块,还有一种针对高性能运算的流水线型设计架构,其核心思想是将数论变换（或类似的算法）的多级运算过程中的操作数匹配规则用硬件资源进行固化,在提升输出端口并行度的同时,以运算级深度为基准例化整个数据流通路径,最终形成二维结构的蝴蝶单元排列范式.上述设计思路首先在 Mook 等人提出的 FFT 算法硬件架构设计过程中得以体现[42],并进一步在上文介绍的 Zhao 等人的工作中[26,32]中针对 Dilithium 算法的核心运算过程进行了针对性的优化和扩展.这种设计架构只需将操作数不断地送入运算单元输入端即可得到最终的结果,因此具有较高的吞吐率,非常适合大量数据且单一运算的应用场景.

随着 NIST 最终的标准方案的公布,对于 PQC 的迁移应用的研究将进入到实质性的阶段.从实例化部署和工程化应用的角度分析,与另外两种实现方案相比,纯硬件的设计方案有如下优势：

1）性能提升明显.通常在独立的硬件资源空间内针对特定的算法设计专用型的电路结构,并使用时序状态机确保功能的正确实现,无需调用软件资源实现逻辑控制,这意味着方案设计更加关注算法本身的特点而非多平台之间的交互规则,特别是对于单一算法进行优化实现时,软硬结合方案的通用性优势难以很好的体现.同时,硬件设计本质是一种比特层面的实现方案,可以从功能模块和整体架构两个维度实现运算的并行性.由于存在上述两方面的特点,纯硬件设计方案通常可以实现更高的性能提升.

2）体系结构更加成熟.经过数十年的研究与发展,硬件设计的理念在学术界与工业界均取得了丰富的科研成果和工具支持.其中,寄存器传输级（Register-Transfer Level, RTL）设计方法已成为主流[43].具体地,硬件系统可抽象成为数据路径与控制网络两个部分,并分别由不同的硬件单元组成.数据路径通常涉及中等规模的组件,如加法器、乘法器、寄存器、多路复用器等,主要负责操作数的运算与存储.而控制网络则对应时序状态机设计,其核心是由计数器、选择器、触发器、控制信号等基础的逻辑单元构成的复杂硬件结构.设计者能够直接或间接地规定每个时钟周期内电路的行为,进而控制数据路径的工作模式,并实现所需的计算流程.此外,基础的 RTL 设计指导原则包括速度与面积互换、流水线设计和乒乓操作三种.具体地,第一种通常视为整体性的设计原则,通过明确延迟时间、时钟周期、面积以及功耗四个指标优先级差异,将硬件设计分为高性能和轻量级两种解决方案,从而根据算法结构特点和应用场景需求提出更加适合的硬件设计思路,并且有利于完成进一步的实例化部署和应用.流水线设计在微观层面的实施方式是在一个组合逻辑电路中插入寄存器,从而降低最长路径的延迟时间.如果将插入的寄存器视为一系列简单的运算单元,也可将流水线思想推广到宏观层面,用于分析并实现较为复杂的功能模块,如本文 3.2 节所示.乒乓操作是使用 FIFO 或寄存器资源实现数据的缓存,通常用于实现流水线结构的数据对齐或不同模块输入输出端的数据规整化.

3）产业化成本可控.考虑到国内和国际的 PQC 标准化方案仍存在进一步优化和发展的空间,新的公钥加密体系尚未建立,因此目前的抗量子安全迁移进程正处于重要的过渡阶段.这意味着现有的后量子密码算法的软硬件实现方案不仅要考虑综合性能,还需要关注实际的部署成本,为未来实现基于后量子密码算法的专用化密码学芯片做好技术储备.如上文所说,FPGA 芯片经过数十年的发展,已经具备了成熟的技术体系和完善的产

业生态,结合已有的密码板卡设计模式,能够快速形成低成本的产品研发和技术迭代.更重要的是,已有的、经过产业实践检验的纯硬件设计成果能够以 IP 核形态挂载至总线结构上,从而高效且平滑的构建专用密码学芯片的设计方案.这是其他设计方案所不具备的产业优势,非常适合过渡阶段的低成本工程化特点.

根据上文的分析,对于 Dilithium 算法的纯硬件设计与实现方案的研究具有重要的价值,同时考虑到签名运算高维多项式向量的设计特点,在方案设计过程中进一步提升算法运算效率应具备更高的优先级.现有的纯硬件实现方案有进一步优化的空间: Soni 等人[22]基于 HLS 方法完成签名算法的硬件设计，Ricci 等人[23]将算法流程直接使用大量的硬件资源进行例化,两个工作均未提出更好的硬件优化技术. Land 等人[24]使用 DSP 单元对多项式运算模块进行优化设计,有效提升了签名算法的运行效率,但整体硬件架构的并行度和资源复用率较低.Wang 等人[25,34]基于不同的技术路径分别实现了两种四并行的多项式运算模块,在提升运算效率的同时也在一定程度上节约了 DSP 资源的消耗.但对于签名算法来说,迭代型数论变换结构难以进一步地提升高维多项式向量的乘法运算效率.与之相比, Zhao 等人[26]设计了流水线结构多项式运算模块,使用 4 个蝴蝶运算单元分别完成数论变换过程中的相邻两级运算,辅以合适的时序状态机设计,有效提升了签名算法的整体运行效率.但是设计的运算模块每时钟周期仅输入输出单个多项式系数,完成k次数论变换需要$256 \times \mathrm{k} + 296$个时钟周期,其中的有效运算时间比例较低.此外,文献采用级联的方式设计 Keccak 核,虽然可以将 SHA3 函数运算的效率加倍,但过于复杂的组合逻辑运算限制了最高时钟频率.基于上述两点分析,Zhao 等人的工作[26]在运算效率方面具有进一步提升的可能性.

2 预备知识

本章节将给出与本文具体工作相关的预备知识,首先给出格困难问题的基本定义,之后详细分析 Dilithium 算法的核心运算流程,为之后的硬件优化设计提供理论基础.

2.1 基本定义和符号说明

首先给出多项式环的基本定义,设 n 和 q 为正整数,其中 n 为 2 的幂次,$\mathbb{Z}$和$\mathbb{R}$分别为有理数集合和实数集合.令集合$\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z} = \{0,1,..,\mathrm{q}-1\}$,集合$\mathbb{R}_q = \mathbb{R}/\mathrm{q}$,则$\mathcal{R} = \mathbb{Z}[x]/(x^n+1)$为多项式环,$\mathcal{R}$中元素为 n 次整系数多项式.定义$\mathcal{R}_q = \mathbb{Z}_q/(x^n+1)$,其中的多项式系数均在$\mathbb{Z}_q$中.$\mathcal{R}_q$中的多项式$f$可表示为$a = \sum_{i=0}^{n-1} a_i x^i$,其中$a_i \in \mathbb{Z}_q$,多项式用向量形式可表示为$a = (a_0, a_1, \ldots, a_{n-1})$.进一步,环$\mathcal{R}/\mathcal{R}_q$上向量可表示为$\boldsymbol{a}$,矩阵可表示为$\boldsymbol{A}$,同时令$\boldsymbol{a}^T$和$\boldsymbol{A}^T$分别表示其转置.在此基础上,对于元素集合$\boldsymbol{a} = \sum_{i=0}^{n-1} w_i x^i \in \mathcal{R}$,其无穷范数$l_\infty$可以表示为$\| \boldsymbol{a} \|_\infty := max\|a_i\|_\infty$,同样向量$\boldsymbol{a} = [\boldsymbol{a_i}]_j \in \mathcal{R}^k$的无穷范数可表示为$\| \boldsymbol{a} \|_\infty := \max\|\boldsymbol{a_i}\|_\infty$.此外,定义专用运算符$S_\eta \subseteq \mathcal{R}$表示元素集合满足$\| \mathbf{w} \|_\infty \leq \eta$.

2.2 MLWE和MSIS定义

LWE 问题是构建格基密码方案的底层数学原理之一.定义分布$(\boldsymbol{a}, b) = \boldsymbol{a}^T s + e$,其中向量$\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$,噪声$e \leftarrow \psi_\alpha$.令秘密$s$服从$\mathbb{Z}_q$上的均匀分布,则计算型 LWE 问题为给定分布$(\boldsymbol{a}, b)$,计算得到秘密$s$; 判定型 LWE 问题为区分给定的$(\boldsymbol{a}, b)$是否为$\mathbb{Z}_q^n \times \mathbb{Z}_q$上的均匀分布.由于经典的 LWE 问题所需的采样空间和运算量较大,因此不适合直接用于构建密码学方案.RLWE 通过引入一个额外的多项式环,并利用负循环移位运算生成矩阵元素,有效降低了采样难度,但同时更加稳定的代数结构特征也在一定程度上降低了安全性,因此相关研究进一步提出了基于模格的 MLWE 问题.定义运算$\boldsymbol{b} := \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$,其中矩阵$\boldsymbol{A} \leftarrow \mathcal{R}_q^{k \times l}$,参数$k = \mathrm{poly}(\lambda)$,向量$(\boldsymbol{s}, \boldsymbol{e}) \leftarrow S_\eta^l \times S_\eta^k$.可以看出 MLWE 问题是有多个 RLWE 实例组合而成,兼顾了 LWE 的安全性和 RLWE 的运算效率,因此多数格基密码方案选择基于 MLWE 问题进行构建.

给定矩阵$\boldsymbol{A} \leftarrow \mathcal{R}_q^{k \times l}$,其中$k = \mathrm{poly}(\lambda)$,MSIS 问题在$\beta > 0$时可被参数化为在由$\boldsymbol{A}$确定的多维格中寻找非零最短原相$\vec{\mathbf{x}} \in \mathcal{R}_q^l$,满足公式$\boldsymbol{A} \cdot \vec{\mathbf{x}} = \mathbf{0}$且$\| \vec{\mathbf{x}} \| \leq \beta$.根据 MSIS 在$l_p (p \in [1, \infty])$范数中的假设,在参数选择合适的条件下,不存在多项式时间算法能够以不可忽略的概率寻找到正确的原像$\vec{\mathbf{x}}$.格基数字签名算法 Dilithium 的安全性同时基于 MLWE 问题和 MSIS 问题,并且更加关注无穷范数下的 MSIS 假设.

2.3 CRYSTALS-Dilithium算法介绍

CRYSTALS-Dilithium 由三个概率多项式时间算法(KeyGen, Sign, Verify)组成,其中KeyGen(1^{λ})为密钥生成算法,输入安全参数1^{λ},输出公私钥对(pk, sk).Sign(sk, M)为签名算法,输入私钥sk和待签名消息M,随后执行循环签名运算,直到产生满足安全边界条件的签名σ.Verify (pk, M, σ)为签名验证算法,输入公钥pk、待验证消息M及其签名σ,输出$b \in \{0,1\}$,用于判断是否为正确的签名.

如算法 1 所示,Dilithium 算法中的核心运算过程包括$\hat{\mathbf{A}} \circ \hat{\boldsymbol{y}}, \hat{c} \circ \hat{\mathbf{s}}_1, \hat{c} \circ \hat{\mathbf{s}}_2, \hat{c} \cdot \hat{\boldsymbol{t}}_0$等环上的多项式乘法运算,通常使用数论变换技术加速其运算效率.由于 Dilithium 的环参数(n,q)的选择刚好满足公式$q \equiv 1(mod 2n)$,因此可以使用经典数论变换加速多项式运算的过程.此时,在循环群$\mathbb{Z}_q^*$上存在 512 次本原单元根α,满足公式$x^n + 1 \cong (x-\alpha)(x-\alpha^3)(x-\alpha^{511})$.根据中国剩余定理可知,每一个多项式$\boldsymbol{a} \in \mathcal{R}_q = \mathbb{Z}_q[x]/(x^n+1)$可以被唯一地表示为$[a(\alpha), a(\alpha^3), \ldots, a(\alpha^{2n-1})]^T \in \mathbb{P}_q^n$.因此多项式乘法运算可以分解为子多项式乘法的形式,并使用迭代的思想最终分解为系数点乘运算,在 256 维多项式系数的数论变换过程中共包含 8 级运算,进而将环上的多项式乘法运算的复杂度由$O(n^2)$降低为$O(n\log n)$.此外多项式系数采样同样是签名算法中关键且较为耗时的运算过程,在 Dilithium 算法中使用了两种 XOF 计算实例:运算过程 ExpandS、ExapndMask、SampleInBall 对应计算实例 SHAKE-256,而运算过程 ExpandA 对应计算实例 SHAKE128,其中耗时最多的两个运算过程分别为$\mathrm{H}(\rho \parallel \boldsymbol{t}_1)$和矩阵$\boldsymbol{A}$的采样运算 ExpandA.在进行硬件实例化设计时,需要重点考虑上述核心运算步骤的优化实现方案.

算法 1 Dilthium 算法核心运算过程

KeyGen():

$(\rho, \rho', K) \in \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} \Leftarrow \mathrm{H}(\zeta||k||l)$ //$\zeta \Leftarrow \{0,1\}^{256}$

$(s_1, s_2) \in S_\eta^l \times S_\eta^\kappa \Leftarrow \mathrm{ExpandS}(\rho')$

$\hat{\mathbf{A}} \in R_q^{k \times l} \Leftarrow \mathrm{Expand\,A}(\rho)$

$\boldsymbol{t} \Leftarrow \mathrm{NTT}^{-1}(\hat{\mathbf{A}} \circ \hat{\mathbf{s}}_1) + s_2$ //$\hat{\mathbf{s}}_1 = \mathrm{NTT}(\mathbf{s}_1)$

$(\boldsymbol{t}_1, \boldsymbol{t}_0) \Leftarrow \mathrm{Power2\,Round}_q(\boldsymbol{t}, d)$

$\mathrm{tr} \in \{0,1\}^{256} \Leftarrow \mathrm{H}(\rho \parallel \boldsymbol{t}_1)$

return $\left(pk = (\rho, \boldsymbol{t}_1), sk = (\rho, K, tr, \boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{t}_0)\right)$

Sign(*sk*,*M*)

$\hat{\mathbf{s}}_1 = \mathrm{NTT}(\mathbf{s}_1), \hat{\mathbf{s}}_2 = \mathrm{NTT}(\mathbf{s}_2), \hat{\mathbf{t}}_0 = \mathrm{NTT}(\mathbf{t}_0)$

$\hat{\mathbf{A}} \in R_q^{k \times l} \Leftarrow \mathrm{Expand\,A}(\rho)$

$\mu \in \{0,1\}^{512} \Leftarrow \mathrm{H}(\mathrm{tr} \parallel M),\ \rho' \in \{0,1\}^{512} \Leftarrow \mathrm{H}(K \parallel rnd \parallel \mu)$

$\kappa \Leftarrow 0, (\mathbf{z}, \mathbf{h}) \Leftarrow \perp$

while (z, h)=⊥ do

$\boldsymbol{y} \in S_{\gamma_1}^l \Leftarrow \mathrm{ExpandMask}(\rho', \kappa)$

$\boldsymbol{w} \Leftarrow \mathrm{NTT}^{-1}(\hat{\mathbf{A}} \circ \hat{\boldsymbol{y}})$ //$\hat{\boldsymbol{y}} = \mathrm{NTT}(\boldsymbol{y})$

$\boldsymbol{w_1} \Leftarrow \mathrm{HighBits}(\boldsymbol{w})$

$\tilde{c} \in \{0,1\}^{256} \Leftarrow \mathrm{H}(\mu \parallel \boldsymbol{w}_1)$

$c \in R_q \Leftarrow \mathrm{SamplelnBall}\ (\tilde{c})$

$\hat{c} := \mathrm{NTT(c)}$

$\boldsymbol{z} \Leftarrow \boldsymbol{y} + \langle cs_1 \rangle$ //$\langle cs_1 \rangle = \mathrm{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_1)$

$\boldsymbol{r}_0 \Leftarrow \mathrm{LowBits}(\boldsymbol{w} - \langle cs_2 \rangle)$ //$\langle cs_2 \rangle = \mathrm{NTT}^{-1}(\hat{c} \circ \hat{\mathbf{s}}_2)$

if $\| \boldsymbol{z} \|_\infty \geq \gamma_1 - \beta$ **and** $\|r_0\|_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) \Leftarrow \perp$

else

$\langle ct_0 \rangle = \mathrm{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{t}}_0)$

$\boldsymbol{h} \Leftarrow \mathrm{MakeHint}(-\langle ct_0 \rangle, \boldsymbol{w} - \langle cs_2 \rangle + \langle cs_1 \rangle)$

if $\|ct_0\|_\infty \geq \gamma_2$ **or** [[# of 1 s in $\boldsymbol{h}$ is $> \omega$]] **then**$(\mathbf{z}, \mathbf{h}) \Leftarrow \perp$

$\kappa \Leftarrow \kappa + l$

return $\sigma = (\tilde{c}, \boldsymbol{z}, \boldsymbol{h})$

Verify(*pk,M', σ*)

$\hat{\mathbf{A}} \in R_q^{k \times l} \Leftarrow \mathrm{Expand\,A}(\rho)$

$\mathrm{tr} \Leftarrow \mathrm{H}(pk, 64)$

$\mu \in \{0,1\}^{512} \Leftarrow \left(\mathrm{H}(tr \parallel M')\right)$

3: $c \in R_q \Leftarrow \text{SamplelnBall}\ (\tilde{c})$
4: $\boldsymbol{w}_1' \Leftarrow \text{UseHint}_q(\boldsymbol{h}, \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{z}) - \text{NTT}(c) \circ \text{NTT}(\mathbf{t}_1 \circ 2^d)), 2\gamma_2)$
5: **return** $[\| \boldsymbol{z} \|_\infty < \gamma_1 - \beta]$ and $[\tilde{c} = \text{H}(\mu \| \boldsymbol{w}_1')]$

3 Dilithium 算法硬件实现方案

从硬件实现的层面分析算法 1 可得到如下结论：首先,Dilithium 算法中存在大量高维度的多项式向量和多项式矩阵,这对应大量的多项式乘法运算过程.其次,签名算法中的子运算类型较多,主要包含了三种核心的采样算法及十余种辅助运算.考虑到上述两个关键的运算瓶颈,在进行硬件实现时需要针对性的设计专用功能模块,同时合理规划数据调度、资源复用、时钟周期等多种因素的约束,进而设计并实现高效的时序状态控制逻辑,从顶层进一步的提升整体的签名运算效率.根据上述分析,本文设计的顶层架构主要由四个部分组成：

（1）基于延迟转向型硬件架构、并满足多参数集的多项式运算模块,采用八级流水的蝴蝶单元阵列满足多种类型的系数运算需求,包括 NTT 运算、逆 NTT 运算、累加模运算和模乘运算.更重要的是,针对 Dilithium 算法高维向量/矩阵的特点,流水线型的设计架构可以连续执行运算操作,极大提升了整个模块的数据吞吐量.

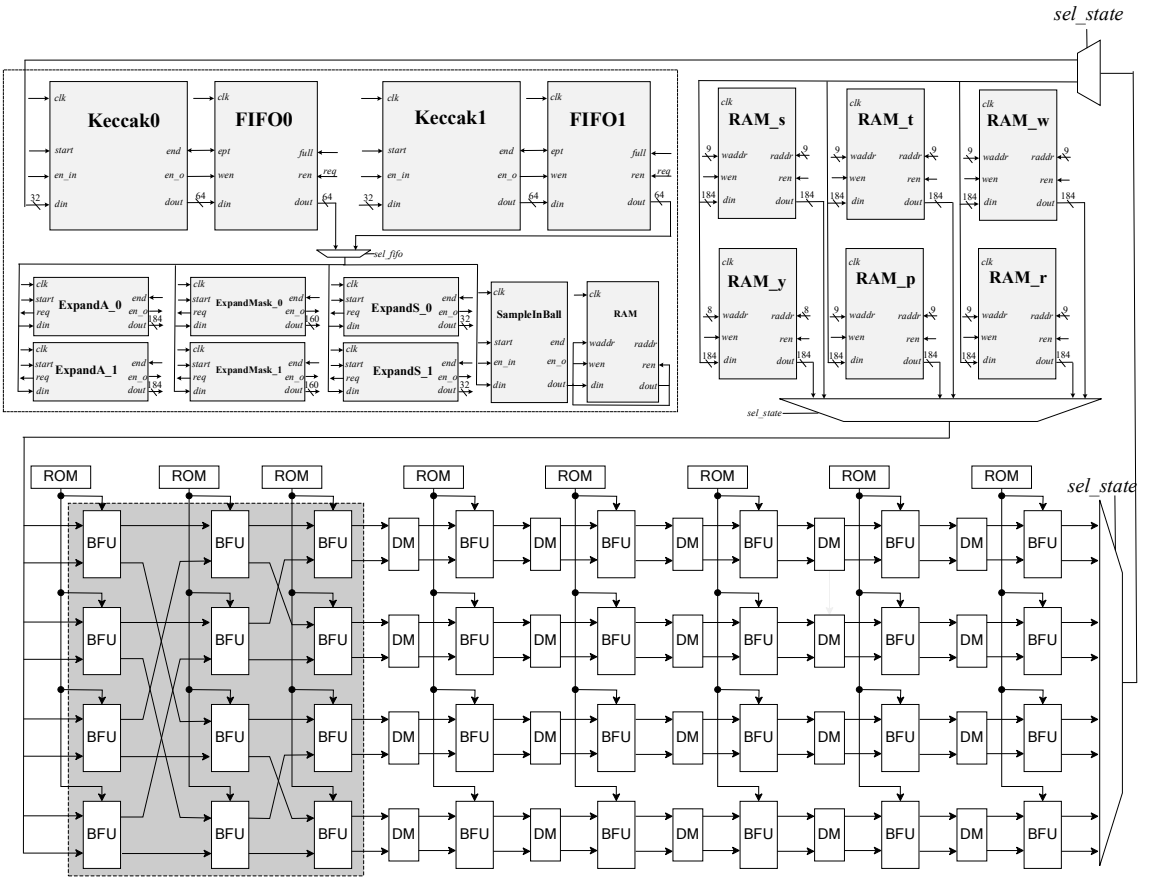

图 1　签名算法硬件设计方案顶层架构

（2）针对 Dilithium 算法中核心采样运算和哈希运算的输入输出数据的具体特点,设计了两种不同的采样模块,并针对专用于采样运算的功能模块进行了合理的优化,辅以用于数据规整化的数据编解码子功能模块,以更低的资源消耗实现了采样运算效率的加倍.

（3）以安全等级 5 对应的参数集为基准设计统一的存储单元阵列,并根据顶层模块输入的片选信号决定

多项式系数的组合规则和存储深度,基于可重构的设计思想,以更高的资源利用率实现可满足全参数集条件下的多项式向量/矩阵存储需求.

（4）通过分析签名过程核心运算步骤的子运算类型,根据细粒度重组的设计思想对其进行分解和组合,有效提升单次输入运算阵列的多项式系数总量,从硬件逻辑上有效降低单个系数对应的无效延迟时间.其次,将顶层的采样模块和运算阵列模块的并行度尽可能地提高,在不产生数据冲突的前提下进一步提升整体硬件架构的时钟周期资源的利用率以及签名运算效率.

3.1 多项式运算模块设计

如图 1 所示,多项式运算模块以延迟转向架构为基础设计实现,其核心部分为基 2 蝴蝶单元（Bufferfly Uint, BFU）构成的运算阵列,辅以 40 个移位寄存器和 20 个多路选择器（Delay Register and Multiplex Selector,DM）,因此也可称为脉动阵列运算单元.如上文所述,在 Dilithium 算法中的数论变换过程为标准的八级迭代运算,与硬件模块的运算阵列一一对应,同时相邻阵列在寄存器和选择器的辅助下实现正确的操作数匹配,相当于将 NTT/INTT 运算的每一级运算过程使用硬件资源进行固化.因此,运算模块输入端可以持续填充数据,每个时钟周期最多可送入 8 个参与运算的操作数,在多级流水的设计结构下可实现对高维多项式向量/矩阵的持续运算.根据上述运算模块的硬件结构可知,总时间消耗由两部分组成:输入端填充数据的时钟周期,这部分时间与单次参与运算的数据总量成正比;运算过程的延迟时周期数,这部分时间与输入数据在运算模块内部的流通路径和运算级深度相关.对于一种特定的运算类型,延迟时间由参与系数运算的蝴蝶单元及其延迟转向单元的数量确定,因此这部分延迟时间是一个固定值.在参数 n 为 256 的前提下,一个完整的多项式系数可在 32 个时钟周期内完成输入,假设对于 K 个多项式系数进行某种类型的运算,其延迟时间为 D,因此时钟周期的利用率可计算为 $32\mathrm{K}/(32\mathrm{K}+\mathrm{D})$.根据分析易知,随着参与运算的多项式数量 K 的增大,有效时间利用率将会向 1 逐步收敛,在硬件层面此过程可解释为：当参与运算的多项式数量增加后,固定长度的延迟时间在更大的数据量上被有效稀释,因此单个参与运算的数据对应的延迟时间自然减低,同时也意味着模块的吞吐率提高.需要注意的是,上述的多项式运算模块的硬件结构特征,在提升具体的多项式运算效率的同时,从宏观层面对于最终的顶层硬件设计方案的选择也有着实质性的影响,这一点将在 3.7 小节的时序控制逻辑部分详细介绍.

3.2 紧凑型蝴蝶单元设计与实现

图 2 给出了多项式运算模块的核心—蝴蝶运算单元的硬件架构,采用了兼顾 CT 和 GS 蝴蝶操作的双路设计结构,设置两级加法器、一个乘法器和一个约减单元,并包含一个延迟深度为 7 个时钟周期的专用寄存器,用于规整化双路输入数据流在硬件层面的时序差异.本文设计的低延迟、紧凑型蝴蝶运算单元可满足多种类型的运算需求,根据片选信号确定本次对应的运算实例.具体地,当片选信号设置为 1 时,对应的实例为 INTT 运算,输入端口从上到下依次为 $(\mathrm{coff0},\mathrm{coff1},\mathrm{tw})$,此时两个输出端口的值分别为 $(\mathrm{coff0}+\mathrm{coff1})/2(modq)$ 和 $(\mathrm{coff0}-\mathrm{coff1})w/2(modq)$.当片选信号设置为 0 时,可对应三种运算实例,分别为模加运算、模乘运算和正向 NTT 运算.需要注意的是,多项式的模加运算和模乘运算仅使用到脉动阵列前两列的 8 个蝴蝶单元,因此在不考虑硬件延迟的情况下完成一组运算消耗的时钟周期数为 32,与数论变换运算对应的输入端填充时间相同.这样的设计架构为之后顶层的时序优化提供了硬件基础.

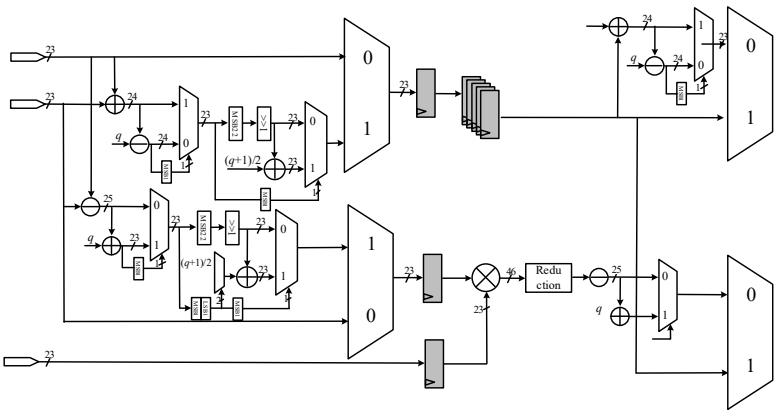

图 2 紧凑型蝴蝶运算单元（脉动阵列的基本运算单元）

在运算过程中需要确保最终的输出结果被规约到范围 $[0, q-1]$中,常用的约减算法有 Montgomery 算法和 Barrett 算法两种[28,29],在本文提出的硬件设计方案中,相比之下 Barrett 约减算法有两点重要的优势：首先,Montgomery 算法的约减结果存在区间$(-q, q)$中,在硬件层面意味着每个系数都要多一比特的符号位,而 Barrett 算法的约减结果存在区间$(0, q)$中,不存在负数的情况,更加适合在硬件设计中统一使用无符号数的方式.其次,Barrett 约减算法的结果同时包括了约减结果 *res* 和商值 *quo*.考虑到在 Dilithium 算法中还存在运算过程$\text{Decompose}_q(r, \alpha)$,其核心功能为计算$r$的高位和低位,即运算过程$r_0 := r mod^{\pm} \alpha$和$r_1 := (r - r_0)/\alpha$,结果刚好与余数 *res* 和商值 *quo* 分别对应.综合上述两点分析,本文选择基于 Barrett 算法设计约减模块.如图 3 所示,根据延迟寄存器的布局结构可知,约减单元存在三级流水运算,在考虑输入输出端的延迟条件下,一次完整的约减过程需要 5 个时钟周期,这同时也是蝴蝶单元上路数据流需要设置延迟寄存器进行时序规约的主要原因.在第一级运算中,根据最低汉明重量原则分解 Dilithium 算法的模值 q 可得$2^{23} - 2^{13} + 1$,在此基础上将 Barrett 算法中的两次乘法操作使用加减法代替,从而节约了 DSP 资源的消耗.第二级运算核心是将初步的约减结果进行修正,修正依据是约减结果的前三比特数据,共对应八种情况,修正的结果存在于一个较大的范围$[0, 2^{24} - 1]$中.在最后的第三级运算中,对于范围$[2^{23}, 2^{24} - 1]$中的数据再进行一次模减运算,可得到最终正确的约运算的结果.如上文所述,运算过程$\text{Decompose}_q(r, \alpha)$对应的功能模块同构于图 3 所示的三级运算的硬件架构,区别在于此时存在 190466 和 523776 两种模值的情况,对应的分解结果分别为$2^{18} - 2^{16} - 2^{13} + 2^{11}$和$2^{19} - 2^9$,同时为了节约硬件资源,在实现 8380417 对应的约减模块时,采用不带除的设计模式.

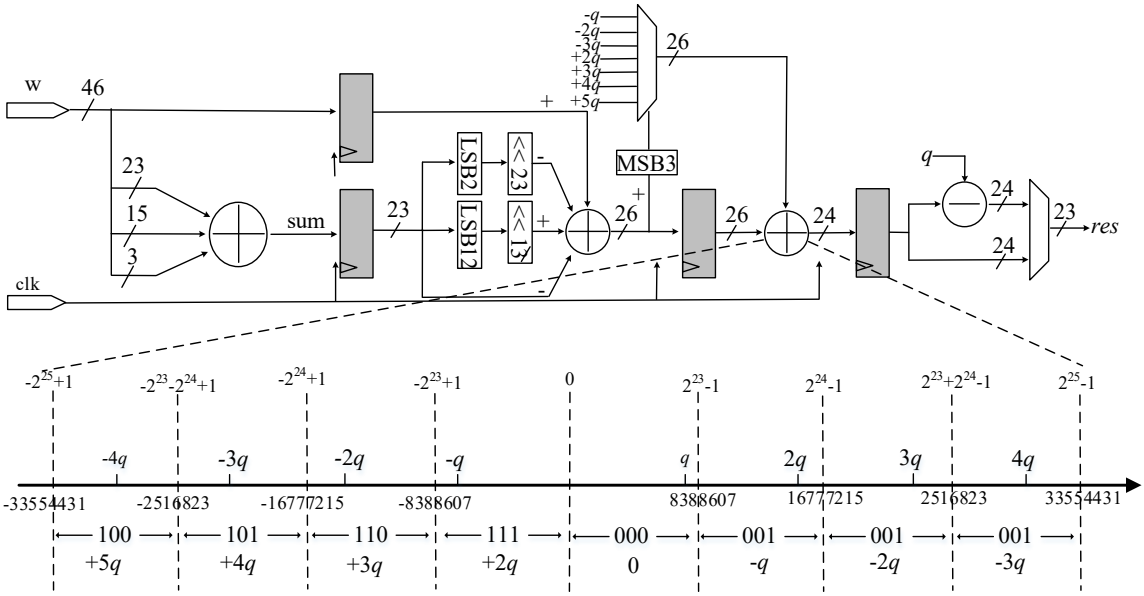

图 3 基于 Barrett 算法的改进型约减运算硬件设计（包含修正逻辑）

3.3 多功能置换网络设计

根据上文分析可知,脉动阵列运算单元的输入端具有顺序敏感性的特点,即运算结果的正确性与输入数据的匹配具有强依赖关系,同时采样得到的多项式系数初始顺序、参与运算的输入顺序、运算完成后的输出结果顺序三者不完全一致.因此,为了确保多项式运算的正确性,需要对于脉动阵列单元的输入或输出端的数据匹配关系进行重排序操作.本文将采样得到的多项式系数排序顺序设定为初始顺序,基于核心的多项式运算类型的特点,设计并实现了包含四种工作模式的多功能置换网络,如图 4 所示,其中系数x的下标使用十六进制计数方式,用来表示多项式中的系数索引值.具体地,第一种置换模式将初始顺序转化为NTT运算的输入顺序,如图 4(a)所示,每组正确的输入系数分布在 8 个不同的地址位,因此每 8 个时钟周期可得到一组正确的操作数组.第二种置换模式将 NTT 运算的输出顺序转化为初始顺序,并在乘法累加运算开始之前完成数据重排,如图 4(b)所示,初始顺序的前 8 个数据分布在四个 NTT 运算的数据结果中,因此每 4 个时钟周期可得到一组正确的操作数组.在向量 $\boldsymbol{t}$ 的运算过程中,INTT($\boldsymbol{As_1}$)运算的输出结果需要与向量$\boldsymbol{s_2}$中的对应系数进行模加操作,考虑到向量 $\boldsymbol{t}$ 中的多项式系数需要以初始顺序参与下一步的H($\rho \parallel \boldsymbol{t_1}$)运算,因此本文选择将INTT运算的输出顺序转化为初始顺序,并将其定义为第三种置换模式,如图4(c)所示.此外,由于点乘运算($\boldsymbol{cs_1}, \boldsymbol{cs_2}, \boldsymbol{ct_0}$)的结果存在于NTT域内,需要在参与下一步运算之前需要将 NTT 运算的输出顺序转化为 INTT 的输入顺序,此过程定义为第四种置换模式,如图 4(d)所示.

由于第一种置换模式所需的延迟周期数最多,因此以第一种情况为基础进行多功能置换网络的设计,所需的硬件资源可同时兼容其余三种情况.根据上述分析,本文的重排序网络主要由两组寄存器构成,每组包含八个位宽为 184 比特的长寄存器,可用于脉动阵列单元输入输出端的数据缓存.同时,将暂存的操作数按照寄存器的索引值依次进行延迟对齐操作,在经过固定数量的时钟周期之后,可确保产生 8 个满足运算需求的系数组.使用两组寄存器依次交替进行上述操作,最终可实现多项式运算及对应的重排序操作的流水线运行模式.

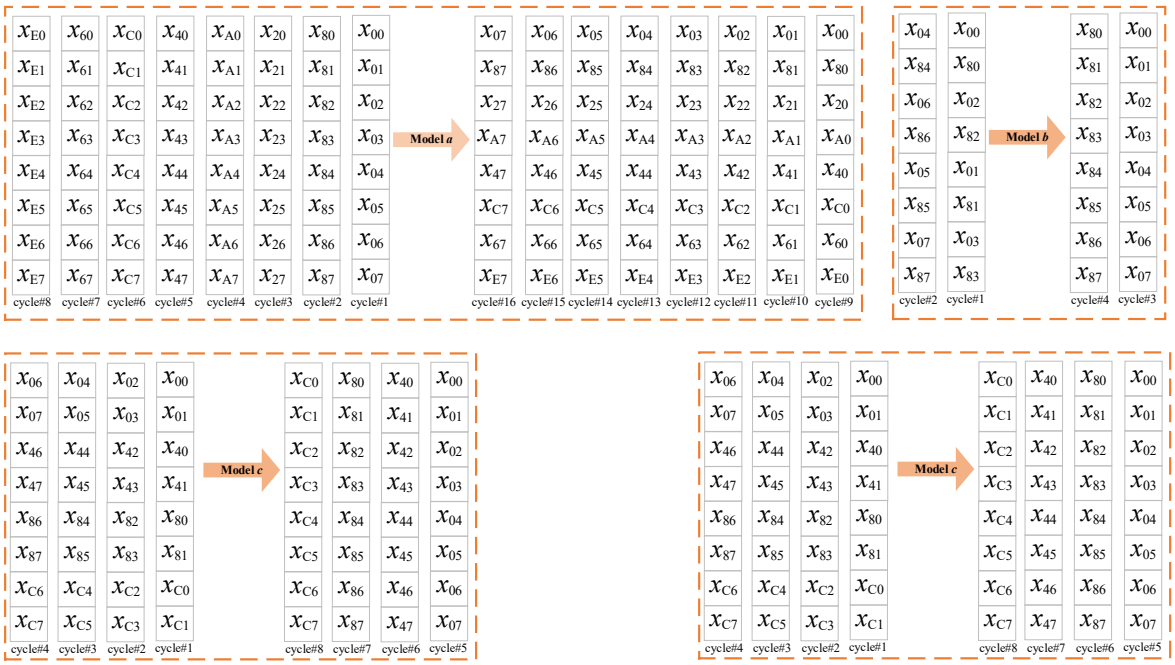

图 4　置换网络工作模式

3.4　双核哈希模块设计

生成随机比特流是采样算法的前置运算,其性能的差异将直接影响后续的采样模块的运行效率.如上文所述,在 Dilithum 算法中使用 SHAKE128 和 SHAKE256 两种计算实例生成随机比特流,其中最核心的三个采样运算为ExpandS、ExpandMask 和ExpandA,分别对应向量$\boldsymbol{s}$、向量$\boldsymbol{y}$和矩阵$\boldsymbol{A}$的多项式系数采样过程,其余的计算过程主要负责生成数据摘要或生成随机种子.本文深入研究并总结了这两类运算的特点,并得到如下结论：采样算法ExpandS、ExpandA和 ExpandMask 的输入参数长度均小于 XOF 计算实例输入数据的分组长度,这意味着在进行系数采样时,哈希模块仅在开始阶段需要输入数据,而在后续的迭代步骤中,不存在将上一阶段运算结果与新输入数据进行异或操作的运算过程,这为哈希模块的优化提供了基础条件.根据上述分析,本文综合考虑了硬件资源和算法效率两个因素,设计了两种不同的专用哈希模块,分别为通用型哈希模块和采样型哈希模块,如图 5 所示.

通用型哈希模块的硬件架构主要由三个长度为 1600 比特的长寄存器构成,输入输出端口的长度设定为 64 比特,可满足签名运算中所有的 XOF 实例的运算需求.与之相对,采样型哈希模块的硬件架构中仅包含两个长寄存器,专门用于多项式向量$\boldsymbol{s}$、向量$\boldsymbol{y}$和矩阵$\boldsymbol{A}$中的系数采样运算.两者区别在于,通用型哈希模块每完成 24 次轮函数迭代之后,运算结果在输出的同时暂存入第三个长寄存器,并与第一个长寄存器中存放的输入数据进行异或运算,从而生成下一轮迭代运算的初始状态数据,此运算过程对应H()函数.由于采样算法ExpandS、ExpandA和 ExpandMask 的输入数据位宽远小于 XOF 计算实例单次输入的最大长度,因此采样型哈希模块中无需第三个长寄存器的存在即可完成所需要的运算功能,在实现层面节约了较多的硬件资源.此外,基于时钟复用的思路设计上述两个运算模块,核心的长寄存器分别对应 XOF 实例运算过程中不同的子运算过程,且彼此之间共享时钟周期资源.在具体的测试过程中,仅需 120 个时钟周期即可完成两个矩阵$\boldsymbol{A}$或向量$\boldsymbol{y}$中的多项式系数的采样运算,而向量$\boldsymbol{s}$的多项式系数采样运算的对应数据为 48 个时钟周期.

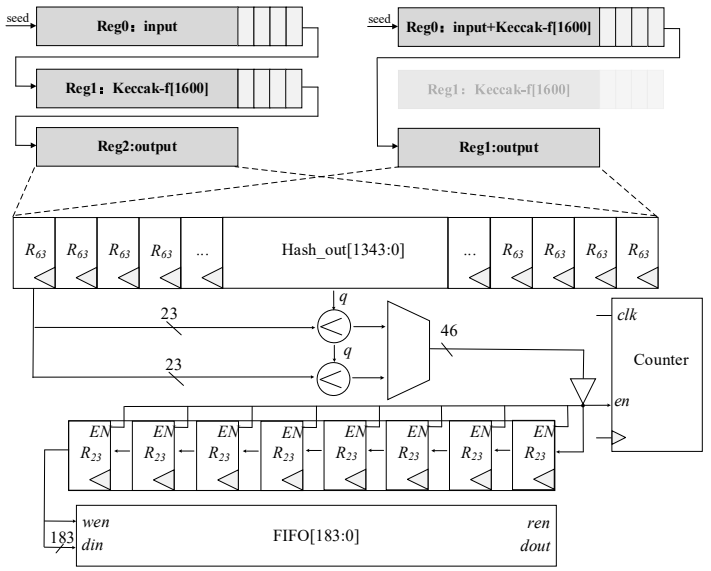

图 5 基于双 Keccak 核的拒绝采样硬件设计

完成采样运算之后,需要在硬件层面对得到的多项式系数进行规整化操作,进而满足多项式采样和多项式运算的顶层流水线设计需求.本文在上述两个采样模块的输出端分别设置了一个深度为32、位宽为184的FIFO单元用于多项式系数的缓存.考虑到每两个 256 维的多项式系数是同步生成的,因此设计了专用的、以单比特指示信号 sel_fifo 为核心的功能控制逻辑,旨在确定两个 FIFO 单元对应的读信号置高的时序位置,从而确保采样得到的多项式系数按照正确的顺序依次参与下一步的脉动阵列运算.具体地,当向量或矩阵中的多项式个数为偶数时,FIFO0 和 FIFO1 的读信号分别依次置高,且最后参与运算的多项式系数来自于 FIFO1 单元中的暂存数据,如向量$\boldsymbol{s_2}$所示.当向量或矩阵中的多项式个数为奇数时,FIFO 单元的读信号时序逻辑不变,但最后参与运算的多项式系数来自于 FIFO0 单元,且 FIFO1 单元中的采样数据将被同步释放.由于采样模块得到新的多形式系数最快需要 48 个时钟周期,远大于 FIFO 单元中暂存数据的完整输出时间,因此不会出现时序冲突和数据覆盖.本文以更少的硬件资源消耗实现了二并行的采样运算,同时在顶层的硬件架构上保持了较高的时钟频率,有效提高了整体的签名运算效率.

3.5 可重构存储阵列设计

根据签名运算过程中多项式系数的类型差异,在硬件层面可将存储空间分为三种情况:(1)与公私钥相关的多项式系数需要长期存储并参与所有的运算过程,应分配独立的存储空间.(2) 矩阵$\boldsymbol{A}$中多项式系数仅参与特定的运算过程,且总数据量较大,应基于 on-the-fly 的思想设计合适的时序逻辑,使采样得到的系数直接参与运算,不占用额外的存储空间.(3)作为运算过程中间值的多项式系数,应与后续的运算输出结果共享统一的存储空间.根据上述思路进行存储资源最小化的设计实现,具体如图 6 所示.考虑到存储单元与前后的采样单元和运算单元在硬件层面具有一定的依赖关系,参考缓存FIFO单元的设定参数,存储单元位宽同样为184比特,即单地址位可存储 8 个多项式系数.为了优化存储空间的设计,需进一步考虑 FPGA 芯片的 IP 核设计特点.在进行实例化的过程中以 36 比特为基准对应 0.5 个 BRAM 单元,如果将单地址上的 8 个多项式系数分别存储,需要消耗至少 4 个 BRAM 单元,存储资源利用率为 63%.本文将单地址位分解为四个 36 比特和一个 40 比特,共对应 5 个部分、2.5 个 BRAM 单元即可实现多项式系数的存储需求.

根据上述分析,以安全等级 5 对应的参数集为基准设计可重构的存储单元阵列,共由六个部分组成.其中,向量$\boldsymbol{s_1}$、向量$\boldsymbol{s_2}$和向量$\boldsymbol{t}$中的多项式系数统一长期存储在图 6 中的 BRAM_s 和 BRAM_t 单元中,其设置深度分别为 480 和 256,最多可存储 23 个多项式系数.向量$\boldsymbol{w}$中的多项式系数存储在 BRAM_w 单元中,设置深度为 256,对应 8 个多项式系数的存放位置,并且与运算过程$\boldsymbol{w-cs_2+ct_0}$的中间值和输出结果共享存储空间,对应上文

所述的第二种情况.此外,向量$\boldsymbol{y}$的多项式系数有两种存在形式：运算过程$\boldsymbol{Ay}$中系数在 NTT 域内,而运算过程$\mathbf{z}=\boldsymbol{y}+\boldsymbol{cs}_1$中系数在自然数域内.因此,同样需要设置独立的存储空间用于存储向量中系数的初始值和数论变换结果,对应 BRAM_y 单元,并与运算结果$\mathbf{z}$共享存储空间.

BRAM_PWM 单元用于存放乘法运算的中间值,由于三个关键的乘法运算过程$\boldsymbol{cs}_1$、$\boldsymbol{cs}_2$ 和 $\boldsymbol{ct}_0$中多项式系数均在NTT域内,可直接参与点乘运算,因此硬件层面的存储空间应与维度最高的向量对应,其深度设定为256.在乘法累加运算过程中,仅使用到了 BRAM_PWM 单元的前 32 个存储地址位,具体地：运算过程$\boldsymbol{A}_{ji}\boldsymbol{y}_i+\boldsymbol{A}_{j(i-1)}\boldsymbol{y}_{i-1}$对应的乘法操作和累加操作同步执行,此过程仅对应一个多项式的存储空间,迭代l次可得到向量$\boldsymbol{w}$中一个完整的多项式系数.由于向量$\boldsymbol{w}$已经分配了独立的存储空间,此时 BRAM_PWM 单元的前 32 个地址位空间可以被释放.考虑到流水线运算的连续性,在完成一轮运算后使用 0 值覆盖 BRAM_PWM 单元前 32 个地址位中的数据,此时每轮迭代的第一次乘法累加可解释为运算过程$\mathbf{0}+\boldsymbol{A}_{j0}\boldsymbol{y}_0$.将上述迭代过程进行$k$次即可得到运算的完整输出结果$\boldsymbol{w}=\boldsymbol{Ay}$.此外,在实现 3.1 节中介绍的脉动阵列单元对应的置换网络具体功能时,需要专用的存储空间暂存多项式系数并完成排列顺序的转换,对应 BRAM_RO 单元,存储深度同样与维度最高的向量对应.

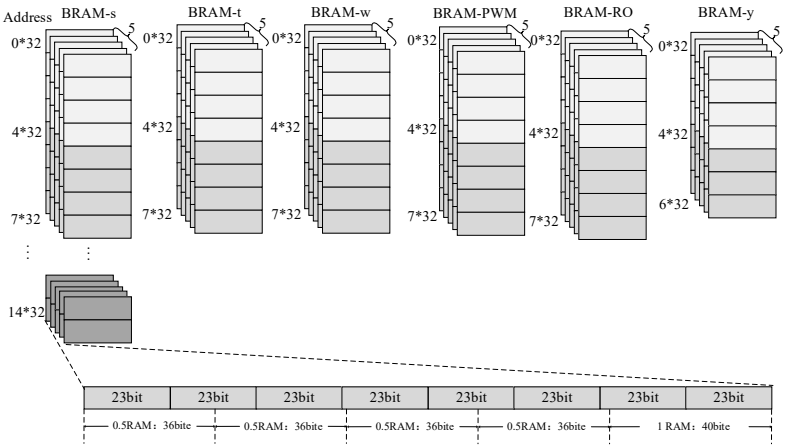

图 6 可重构存储单元阵列

3.6 读写地址映射逻辑设计

在设计 BRAM 阵列的访存逻辑时还需要考虑脉动阵列单元两方面的运行特点：多项式运算需要基于多功能重排序网络实现系数匹配；运算单元的输入端和输出端存在固定的延迟时间,这意味着从多项式向量的尺度上看,可能出现同一组系数的访存过程和运算过程的时钟复用.为了在避免时序冲突的前提下提升整体架构的运行效率,本文针对上述两种情况分别设计了四种地址映射规则和两组同步访存逻辑.

重排序网络将操作数的存储位置在多项式尺度层面进行了定向置换,旨在满足脉动阵列单元的运算需求,因此重排序网络的功能实现可具象化为多项式系数在存储单元阵列中的读写地址映射规则.具体地,在第一种重排序模式下,输入端的每组操作数分布在八个不相邻的地址位,初始读地址映射规则为(0,16,4,20,8,24,12,28),之后每组读数据位置在此基础上递增,最终遍历 32 个地址位上的存储系数.在第二种重排序模式下,每组操作数对应四个不同的地址位,其输出端的写地址初始映射规则为(0,8,16,24),此时存储单元中的系数呈现顺序排列的形式,从而使得下一步的乘法累加运算$\boldsymbol{Ay}$的读数据地址与时钟周期数同步.其余两种重排序模式遵循类似的设计逻辑,旨在满足INTT后的模加运算和NTT后的INTT运算规则.本文将四种专用的读写地址映射预存储在 160 比特的寄存器网络中,根据顶层的片选信号以及每周期的比特左移操作即可得到正确的读写地址位信息,如表 1 所示.

表 1 输入系数地址位与时钟周期偏移量映射关系

工作模式	Cycle1	Cycle2	Cycle3	Cycle4	Cycle5	Cycle6	Cycle7	Cycle8

采样后 NTT	0/1/2/3	16/17/18/19	4/5/6/7	20/21/22/23	8/9/10/11	24/25/26/27	12/13/14/15	28/29/30/31
NTT 后模乘	0/2/1/3	4/6/5/7	8/10/9/11	12/14/13/15	16/18/17/19	20/22/21/23	24/26/25/27	28/30/29/31
INTT 后模加	0/2/1/3	16/18/17/19	8/10/9/11	24/26/25/27	4/6/5/7	20/22/21/23	12/14/13/15	28/30/29/31
NTT 后 INTT	0/8/4/12	1/9/5/13	2/10/6/14	3/11/7/15	16/24/20/28	17/25/21/29	18/26/22/30	19/27/23/31

考虑到脉动阵列运算单元的输入输出端的时序差异,通过合理的功能设计可以实现读操作、重排序操作、写操作三个过程的同步运行,从而保持整体的流水线运算结构的统一性.本文针对存储单元阵列设计了两组并行的读写地址生成器：（1）针对核心的多项式运算过程的计数器组(ctr_k, ctr_{k_v2}),其中主计数器ctr_k负责确定输入过程的总时钟周期数,辅计数器ctr_{k_v2}根据运算类型、在主计数器的基础上延迟确定的时钟周期后即可生成写地址位.（2）重排序、采样数据存储等辅助运算过程使用计数器组$(raddr_{ro}, waddr_{ro})$生成读地址和写地址,与第一组计数器采用相同的设计逻辑.图 7 以核心运算过程$\boldsymbol{Ay}$为例给出了两组地址生成器同步运行的具体规则.首先,向量$\boldsymbol{y}$中的多项式系数根据第一种重排序模式实现数据存储,此过程共消耗$(l*32+4)$个时钟周期,容易看出,此时点乘运算和重排序操作存在时序复用的情况,根据上文分析,需要同时启动两组地址生成器分别完成两种运算需求.具体地,计数器$raddr_{ro}$的工作范围设定为$(0, l*32)$,可完整遍历 BRAM_y 单元的所有地址位,读取数据并进行对应的重排序操作,并基于表 1 给出的映射规则和时钟延迟生成 BRAM_RO 单元的写地址$waddr_{ro}$.另一方面,矩阵$\boldsymbol{A}$中的一个完整的多项式采样生成后,主计数器ctr_k启动并完成一次 32 时钟周期的循环遍历,用于生成 BRAM_RO 单元的读地址.在此基础上,辅计数器ctr_{k_v2}在延迟 8 个时钟周期后开始启动,并生成 32 个自然顺序排列的写地址数据,从而完成一次完整的点乘运算.需要注意的是,重排序操作持续运行并先于第一次点乘运算启动时间至少 32 个时钟周期,这样的设计可避免两个过程的时序冲突,确保运算结果的正确性.

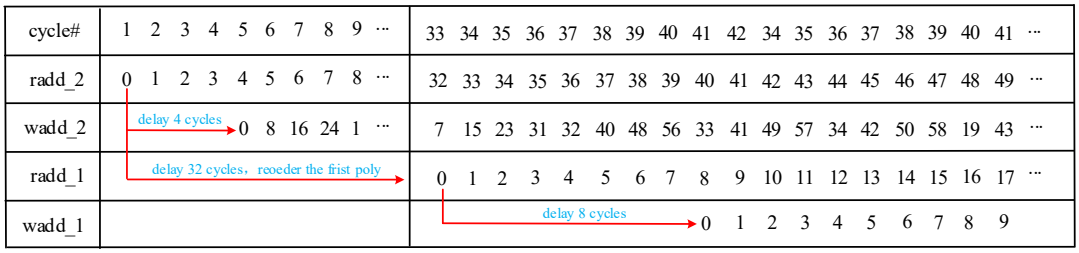

图 7　针对乘法累加运算的同步访存逻辑运算规则

3.7　分段式时序状态设计

根据 3.1 节关于脉动阵列单元的设计特点可知,在硬件延迟时间固定的前提下,增加单次参与运算的数据量可有效提升时钟周期资源的利用率.本文以脉动阵列单元的运算特点为基础,以向量而非多项式为运算的基本单位,并充分考虑签名运算的具体过程,提出一种分段式时序状态控制逻辑,其核心思路是将签名运算的核心计算流程进行细粒度的解构,根据运算类型的差异将其重组为多个专用运算组并统一参与运算,极大提高了核心运算单元的吞吐率,从而提升了硬件架构的整体运行效率.

将上述的时序控制设计思路在签名运算过程中进行具象化实现,以数据量最大的安全等级 5 为例.首先,将向量组$(\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{t_0})$中的多项式系数统一送入脉动阵列单元进行正向数论变换,输入端共消耗 732 个时钟周期,综合考虑 96 个时钟周期的固定延迟时间,此运算过程的有效时间利用率为 88%.其次,采样运算同步进行并生成向量$\boldsymbol{y}$的多项式系数,同样使用脉动阵列单元实现正向数论变换,此时输入端消耗 224 个时钟周期,有效时间利用率为 70%,可以看出向量组运算的有效时间利用率明显高于单个向量的对应数据.最后重点分析$(\boldsymbol{y}+\boldsymbol{cs_1}, \boldsymbol{w}-\boldsymbol{cs_2}, \boldsymbol{w}-\boldsymbol{cs_2}+\boldsymbol{ct_0})$三个核心的运算过程,其特点是均由乘法累加、逆向数论变换和模加运算三部分组成,基于细粒度的分解重组设计思想可获得三个类型统一的运算组(cs_1, cs_2, ct_0)、INTT(~) 和$(+,-,-,+)$.使用脉动阵列单元分别进行点乘运算、逆向数论变换和模加运算,分别对应 8、96、8 个时钟周期的延迟时间.可以看出,这样的时序设计最大化的减少了核心运算单元的启动次数及其附带的固定延迟时间,并且有效分解了单一系数尺度上的无效等待时间,在确保流水线运算正确性的前提下极大提升了整体的签名运算效率.

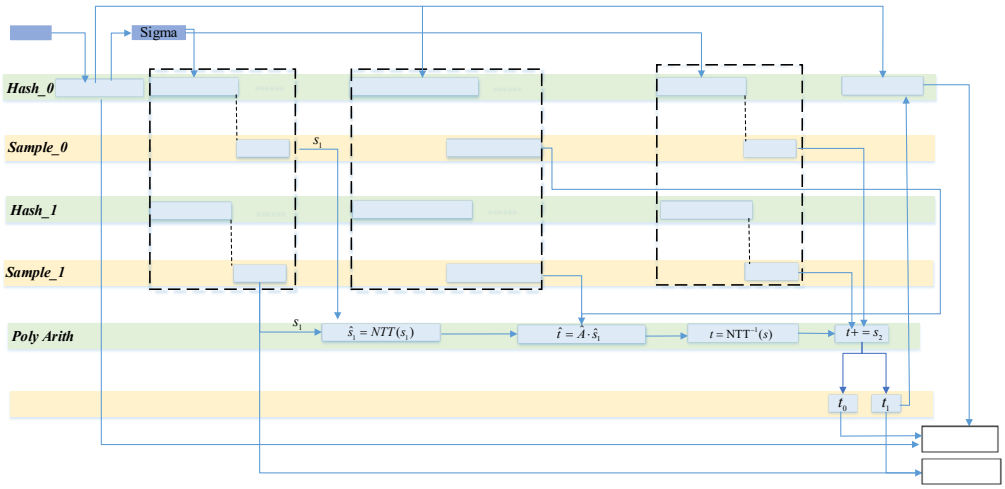

图 8 密钥生成算法时序控制逻辑

4 性能对比

本文使用 Verilog 硬件描述语言设计并实现统一型电路架构,可同时支持 Dilithium 算法三种安全等级下的数字签名运算,通过调整输入端口的片选信号值选择算法实现的参数集.基于 Xilinx Vivado 2017.4 开发套件对编写的 Verilog 代码进行仿真、综合和布局布线,并选择 Xilinx Artix-7 FPGA 平台进行实际的部署和测试,芯片具体型号为 xc7a200tfbg484-2,最终得到完整的性能测试报告,包括时钟周期数据、最高时钟频率和硬件资源消耗三个方面.其中,时钟周期数对应硬件电路生成正确签名结果情况下的时间,也就是最高速率.硬件资源主要关注四个通用的数据,分别是:用于实现基础逻辑功能的查找表资源(LUTs)、与数据存储和信号延迟相关的触发器资源(FFs)、用于大量数据存储与缓存相关 BRAM 资源以及用于乘法运算的 DSP 资源.

4.1 硬件资源消耗数据分析

表 2 给出了各子功能模块的硬件资源占用情况,其中以脉动阵列为核心的多项式运算模块占用了 51%的 LUT 资源和 54%的 FF 资源.由于乘法器单元以 18 比特为基准,在签名算法中 23 比特的多项式系数点乘对应两个 DSP 单元,因此图 1 所示的整个脉动阵列模块共消耗 64 个 DSP 单元.基于面积节约优先的 IP 核设置方式能够节约 50%的 DSP 单元,但同时会消耗更多的 LUT 资源,在实际的应用环境中可根据具体的需求选择更为合适的设计方式.基于本文使用的测试平台分析,Atrix-7 系列 FPGA 芯片包含的 DSP 单元数量可以满足脉动阵列的乘法器资源的消耗.并行采样单元在硬件层面共占用了 32%的 LUT 资源和 26%的 FF 资源.对比两个模块可知,采样型 Keccak 核占用的硬件资源数据明显小于通用型 Keccak 核,共节约了 16%的 LUT 资源和 25%的 FF 资源.文献[26]提出了一种单核级联式架构的哈希模块设计方案,同样可以实现采样过程速率加倍,共消耗 4577 个 FF 资源,与通用型 Keccak 核的对应数据处于同一数量级,而经过优化设计的采样型 Keccak 核仅消耗 3308 个 FF 资源,与之相比节约了 28%左右的寄存器资源.为了实现单时钟周期内遍历两次轮函数运算,级联式的设计架构需要实例化更多的组合逻辑电路.从整体上看,文献[26]设计的 Keccak 核共消耗 15784 个 LUT 资源,与之相比本文提出的双核结构节约了 23%左右的硬件资源.此外,在设计存储阵列的单元组合规则和数据访存逻辑过程中同样充分考虑了 IP 核的硬件特点,每组存储单元的单地址位上可容纳 8 个长度为 23 比特的多项式系数,仅占用 2.5 个 BRAM 资源,在满足可重构的基础上有效节约了存储资源的消耗.

表 2 功能模块硬件资源占用量

模块	硬件资源			
	LUT	FF	BRAM	DSP
Keccak_v1	6569	4414	0	0
Keccak_v2	5502	3308	0	0

Expand_s ×2	113/45	46/46	0	0
ExpandA ×2	179/116	367/367	0	0
ExpandMask ×2	336/366	264/265	0	0
Encode ×2	47/22	38/38	0	0
sampleInball	40	62	0	0
RAM_C_TAU	0	0	0.5	0
NTT_core	19664	16588	18	64
FIFO0	478	203	0	0
FIFO1	478	203	0	0
FIFO2	1647	154	0	0
Combine	37943	30268	18.5	64

4.2 核心运算模块对比分析

本文首先对核心的多项式运算模块进行了部署和测试,对比文献包括纯硬件实现和软硬结合实现两种类型,具体内容如表 3 所示,其中时间与面积乘积(area-time product, ATP)中的时间数据统一使用 NTT 运算与 PWM(point-wise multiplication,PWM)运算的时钟周期数之和,同时以本文的数据为基底,将对比文献的 ATP 数据进行等比例缩减,从而更加清晰地体现数据差异.从整体上看,本文设计的硬件模块在运算性能方面具有较大的优势,根据 ATP 结果可知,在 LUT 和 DSP 两种硬件资源的消耗量上也达到了较好的平衡.同时文献[35]中的 NTT 运算、PWM 运算、PWA 运算(piont-wise addition, PWA)分别对应三个不同的硬件单元,因此如果按照上述的统一规则来算,文献[35]设计的核心运算模块对应的真实 ATP 值要比表 3 中的数据更高.对于 BRAM 数据,不同的设计思想对应不同的存储资源消耗,本文的存储资源面向整个数字签名算法的所有运算过程,而软硬结合设计模式还可以利用软核中的存储空间,如文献[34]中提到的 L2 缓存.此外,本文使用了大量的寄存器资源实现多功能置换网络,在确保运算正确性的同时消耗了过多的 FF 资源,其他的设计方案中不存在这一过程.因此为了公平起见,本文主要列举了与多项式运算过程关系更加密切的 LUT 和 DSP 两种硬件资源的数据对比,并重点对最新的纯硬件实现方案[26]和性能最优的软硬结合方案[32]两项工作对比分析.

本文和文献[32]都设计了基于流水线思想的高性能脉动阵列单元,有效提升了多项式运算的速率,区别在于文献[32]针对 Kyber 和 Dilithium 两个算法设计核心运算单元,因此对前三列蝴蝶单元的寄存器网络和数据流通路径进行了针对性的设计,分别用以支持 TNTT 算法(Truncated-NTT)对应的三阶段 PWM 运算过程.根据 LUT 的 ATP 的数据对比可知,这种设计模式提升了硬件模块运算过程的可扩展性,但同时也增加了面积消耗,并在一定程度上降低了硬件资源的复用率,核心原因在于:如 3.2 节所述,由于 Dilthium 的参数集支持标准数论变换过程,在硬件层面的 PWM 运算和 NTT 运算满足同构设计的要求,而文献[32]中前三列专用化蝴蝶单元所额外消耗的硬件资源仅在 Kyber 算法的 PWM 运算过程中发挥作用;此外,从两个算法对应的 MLWE 实例化结构来看,Kyber 算法的运算量要远低于 Dilithium 算法,在采样速率比较接近的条件下,核心的多项式运算单元对于两个算法的性能提升幅度有较大的差异.这意味着4×8结构的脉动阵列单元已超过了 Kyber 算法的正常加速需求,更适合作为 Dilitium 算法的专用化运算模块.相比之下,文献[26]对应的 ATP 的数值最低,说明核心运算模块的综合性能更加均衡,主要原因在于作者针对 Dilthium 算法进行了专用化的设计,使用四个蝴蝶单元设计流水线结构的核心运算单元.其次,论文分析了运算模块和采样模块之间的数据依赖关系,设计了级联结构的 Keccak 核用于生成采样运算所需的随机比特流,使得多项式的采样模块和运算模块的输入输出速率大致处于相互匹配的状态,在状态机层面降低了核心运算的无效等待时间,这也意味着硬件资源在大多数时钟周期下都得到了有效利用.

进一步将上述两篇文献共同与本文进行对比分析,根据 RTL 基础设计原则之一“面积与速度互换”可知,文献[26]提出的纯硬件设计方案更好地平衡了面积和时间两个方面的资源消耗,具有更好的综合性能,在实际部署过程中具有一定的成本优势.文献[32]设计的核心运算模块具有更好的灵活性和可扩展性,适合更加多元化且资源不敏感的应用场景.相比之下,本文的设计方案在运算效率上具有明显的优势,可以更好地满足高性能运算的应用场景.

表 3 多项式运算模块性能对比

相关工作	测试平台	单元	Area Source	Cycle Count	Area × Time

		数量	LUT	BRAM	DSP	NTT/INTT	PWM	LUT	DSP
26	Artix-7	4	1919	2	8	256	64	0.49	0.63
25	Zynq-7000	1	2386	1	8	264	260	0.99	1.02
34	Zynq-7000	1	6639	3	16	265	73	1.78	1.32
24	Artix-7	2	524	1	17	533/536	85	0.26	2.59
35	Artix-7	1	799	4.5	2	1405	269	1.06	0.82
32	Artix-7	32	25674	6	64	32	32	1.31	1.00
本文	Artix-7	32	19664	18	64	32	32	1.00	1.00

4.3 综合性能对比分析

表 4 给出了本文实现方案与相关工作的对比,包括纯硬件实现和软硬结合实现两种类型，并分别给出详细的对比分析结果.为了更加公平和准确地体现本文方案的性能优势，将所有的纯硬件实现方案的性能数据进行了可视化处理,如图 9 所示,纵坐标单位为 us.如第 1 章所述,文献[25]虽然是基于 Zynq 平台设计的软硬结合方案,但签名算法的核心计算流程完全在 PL 端实现,PS 端仅负责初始的信号控制和最后的数据输出,因此文献[25]的测试数据也在图 9 中体现.此外,表 3 中提到的文献[32]基于 Artix-7 平台对核心的多项式运算模块进行部署和测试,但该文献提出的完整实现方案采用了基于 Rocket Core 的软硬双核设计,并且基于 TSMC 技术进行部署和测试，与其他的实现方案区别较大,因此本文不考虑在表 4 中对文献[32]进行对比分析.

文献[26]针对第三轮 Dilithium 算法的参数集设计了专用化的硬件电路架构,可支持三种安全等级下的数字签名运算.与本文类似,核心的多项式运算模块同样基于延迟转向结构进行设计,使用 BRAM 单元实现运算级之间的数据延迟,每个蝴蝶单元可完成相邻两级的数论变换运算,因此仅实例化四个蝴蝶运算单元.这样的电路结构在一定程度上节约了硬件资源、特别是寄存器资源的消耗,但同时也限制了多项式运算的效率：每个时钟周期仅支持输入一个多项式系数,一次完整的数论变换需要消耗 296 个时钟周期,其中包括 40 个时钟周期的固定延迟时间.本文采用了高并行度的脉动阵列的设计架构,极大提升了模块的数据吞吐量,完成单次数论变换仅需要 32 个时钟周期.同时,辅以分段式时序状态控制逻辑,将签名算法的核心运算步骤重构为点乘运算、INTT 运算和模加运算三个部分,并以向量组为基本单元统一参与运算,能够最大程度发挥脉动阵列运算单元的高吞吐率优势,从而提升硬件架构的运行效率.从整体上看,文献[26]提出的设计方案消耗的硬件资源为 29998LUTs、10366FFs、11BRAMs 和 10DSPs,在面积资源方面有一定的优势,但时钟周期消耗远高于本文的设计方案.此外,本文的最高时钟频率也优于文献[26],其主要原因在于采样模块的设计.如上文所述,虽然基于级联式设计思想的单核硬件架构同样可以提升采样运算的效率,但需要设计较为复杂的组合逻辑电路,并且在单时钟周期内连续完成两次轮函数迭代运算,这意味着时钟的建立时间T_{su}和保持时间T_h将被极大地延长,从而限制了时钟频率的提升.而本文采取双核模式的设计思路,并根据不同类型的采样运算的特点针对性地优化 Keccak 核,在满足并行采样需求的前提下实现了更高的时钟频率.与文献[26]相比,在三种安全等级下,本文的设计方案的整体运算效率分别提升了 1.4/7.4/0.9 倍、1.1/8.3/0.7 倍、0.8/5.6/0.6 倍.进一步分析性能对比结果,可以看出签名运算过程的效率提升最为明显,其次是密钥生成运算,而签名验证过程的性能提升效果最低,主要原因与运算类型和向量维度相关：参与签名运算过程的多项式数量最多,基于细粒度优化思想的分段式时序状态控制逻辑、结合高并行度的脉动阵列单元可有效提升整体的运算效率,因此相比于另外两个部分,签名运算获得了更高的性能提升表现.此外,相比于密钥生成运算,签名验证过程需要多进行一次$\mathrm{H}(\mu \parallel \boldsymbol{w}_1')$运算.由于本文的双核架构的哈希模块无法加速$\mathrm{H}(\mu \parallel \boldsymbol{w}_1')$运算过程,而级联式 Keccak 核在此种情况下依然可以实现并行运算,因此在哈希运算密度较高的签名验证运算过程中,性能提升效果低于另外两个运算部分.

表 4 硬件设计方案整体性能对比

文献	LUT	FF	BRAM	DSP	f[MHz]	KeyGen		Sign		Verify		安全等级
						t[us]	cycle	t[μs]	cycle	t[μs]	cycle	
本文 [a,1]	37943	30268	18.5	64	131	17.9	2348	34.4	4508	23.8	3123	2
						29.4	3864	49.2	6458	36	4718	3
						48.9	6401	76	9958	56.2	7763	5
[26] [a,1]	29998	10366	11	10	96.9	43.1	4172	289.9	28091	45.6	4422	2
						60.4	5851	461.3	44706	63.8	6181	3
						90.5	8765	505.6	48996	93.9	9039	5

[27][b,1]	53907	28435	29	16	256	19	4875	43	10945	26	6582	2
						32	8291	63	16178	39	9724	3
						55	14037	95	24358	57	13642	5
[24][a,1]	24320	9668	15	45	140	134	18761	478.3	66966	62.6	8770	2
	29987	11274	23	45	142	233.1	33102	740.3	105129	85.1	12084	3
	42860	14136	33	45	127	401.4	50982	883	112145	129.6	16462	5
[25][c,2]	18558	7432	17	10	159	49	7757	327	52038	48	7675	2
	19614	8466	21	10	159	82	12982	561	89213	71	11232	3
	20973	9677	28	10	159	127	20189	589	93708	100	15875	5
[30][2]	-	-	-	-	72	2325.5	167433	8816.2	634763	3187.2	229481	2
						3101	223272	11328.2	815636	3836.4	276221	3
[31][d,2]	2620	-	16	6	100	—	—	12600	—	9940	—	3
[34][c+d,2]	6639	1660	3	16	600	731	438698	2829	1697566	904	542212	2
						1330	798111	4685	2990703	1466	879854	3
						2085	1251278	5615	3369190	2298	1378741	5
[35][c,2]	13128	11556	14	4	150	1100		2300		1100		2
						1500	-	3100	-	1600	-	3
						2200		4500		2300		5

-字母上标 *a,b,c,d* 对应四种测试平台：*a:*Artix-7; *b:*VirtexUltraScale+; *c:*Zynq; *d:*PolarFire

-数字上标 1,2 对应两种设计方案：1 纯硬件设计方案; 2 软硬件结合设计方案

文献[27]实现了的2×2架构的多项式运算单元设计方案,利用 FIFO 的数据缓存功能实现操作数的匹配,同时基于迭代型访存架构实现 NTT 运算,每个时钟周期可同时输入四个多项式系数,每个蝴蝶单元对应数论变换的相邻两级运算,完成一次正向或者逆向 NTT 运算共消耗64×4个时钟周期.与本文的脉动阵列运算单元相比在硬件资源占用方面有一定的优势,但运行效率方面有较大的差距,对于核心的签名运算所消耗的时钟周期数是本文的两倍以上.考虑到文献[27]基于高性能的 Virtex UltraScale+平台进行方案的部署与测试,因此具有更高的时钟频率,但综合来看,本文的设计方案在运行效率方面依然有一定的优势,时间消耗分别为文献[27]的 92%/80%/91.5%、90.6%/78.1%/92%、87.5%/83%/98.5.由于文献[27]使用三个 Keccak 核实现不同类型的运算功能,并且没有提出类似于本文的模块优化技巧,因此消耗了较多的硬件资源.根据表 4 中给出的数据对比可知,LUT 资源和 BRAM 资源的消耗量分别是本文的 1.4 倍和 1.5 倍.

文献[24]同样提出了支持第三轮 Dilithium 算法的硬件设计方案,区别在于三组参数集对应三个不同的电路结构,与本文工作和文献[26]相比硬件资源的兼容性和复用率较低.虽然文献[24]专门设计了 NTT 模块和 MACC 模块分别用于多项式的数论变换和乘法运算,但不同的功能模块之间存在一定的时序差异,完成一次正向 NTT 运算、逆向 NTT 运算、点乘运算和模加运算所需的时钟周期数分别为 533、536、85、75.本文基于高性能的脉动阵列单元统一实现所有的运算过程,在提升运算效率的同时将不同运算类型的时钟周期数进行了规整化操作,从而为顶层的时序控制状态机的设计和优化提供了可行性.综合来看,对于 Dilithium 的三组参数集,在采用同样的硬件测试平台的前提下,本文提出的硬件设计方案在密钥生成、签名运算和验证运算三个过程中分别提升了 6.6/12.9/1.6 倍、7.0/14.0/1.3 倍、7.3/10.6/1.3 倍的运行效率.

文献[25,30,31,34,35]针对 Dilithium 算法提出了多种软硬结合的设计方案,核心思想是统一硬件设计的高效性和软件实现的灵活性,针对多项式乘法等复杂的运算过程设计专用的硬件加速器,同时在软件层面调用硬件功能模块并完成算法逻辑.文献[30]使用 RISC-V 指令集进行方案设计,可支持 NIST 第二轮的 Dilithium 算法的参数集运算.由于该方案仅实现了 NTT 和 Keccak 两个专用的功能模块,因此整体的硬件资源占用量较少.使用指令集语言调用硬件功能模块可有效加速部分运算过程的效率,但与纯硬件设计相比整体的并行程度较低,无法进一步提升算法的整体性能.文献[31]基于 Zynq-7020 平台设计了专用的多项式运算模块,完成一次数论变换需要 1170 个时钟周期,相比于本文的脉动阵列运算单元性能提升效果较低.

文献[25]和文献[34]是相同的作者设计的两种不同的软硬结合方案,文献[35]与文献[25]都基于 Zynq-7000 平台实现了完整的 Dilthium 算法，但采用了不同的设计思路.三篇文献的设计方案具有一定程度的关联性，因此统一进行对比分析.首先,文献[25]采用了全硬件开发的思路,所有的签名运算功能均在硬件层面进行实现,软件部分仅负责基本的使能信号和控制逻辑,这样的设计模式能够有效降低软件平台和硬件平台之间的交互频率,进而提升算法的整体运行效率.此外,论文设计了专用化的基 4 多项式运算模块,每次数论变换过程固定消耗

264 个时钟周期,而本文设计的脉动阵列单元在流水线运行的条件下,随着参与运算的数据量加大,单次运算的延迟时间将逐步减小,更加符合 Dilithium 算法的数据集特点.从整体上看本文的设计方案依然具有较大的性能优势,在三组参数集下的运算效率分别提升了 1.7/8.5/1.0 倍、1.8/10.4/0.9 倍、1.6/6.75/0.7 倍.相比之下,文献[34]的软硬件的解耦程度更高,其中硬件端主要负责加速多项式运算,考虑到通信开销和整体的可扩展性,将较为耗时的 SHA3 函数以及对应的采样运算在软件端实现,并使用指令集语言进行性能优化,最终完成了对于 Kyber 和 Dilithium 两个算法的设计与实现.此外,作者进行了三种类型的性能测试,第一种为纯软件单核运行,第二种为软硬结合单核运行,第三种为软硬结合多核运行,多核意味着采样运算速率倍增,表 4 中给出的是性能最高的多核条件下测试数据.与文献[25]的对比可知,尽管软硬结合多核方案能够支持多算法的运行,但整体性能存在大幅度的下降.最后,文献[35]同样采取了解耦合的思想设计软硬结合方案,在 Zynq-7000 平台的 PL 端设计多项式运算模块和 SHAKE 模块,PS 端负责功能调用和算法逻辑.如 4.2 节所述,文献[35]设计了三个硬件模块分别用于 NTT/INTT 运算、PWM 运算和 PWA 运算,没有采用文献[40]中提到的硬件分时复用的思想,因此与文献[34]相比消耗了更多的硬件资源. 将上述三个设计方案与本文进行综合对比分析可知，随着硬件化程度的提高，算法的整体运行效率也随之提升,其核心原因有两点:1)纯硬件设计的并行运算同时体现在功能模块和整体架构两个维度上，相较于其他的设计方式并行化程度更高;2)如文献[34]和文献[35]所述，软硬件之间的数据交互过程本身同样需要消耗大量的时钟周期,进而将影响最高频率.因此基于软硬结合方式的设计方案在整体性能上要低于纯硬件的设计方案.

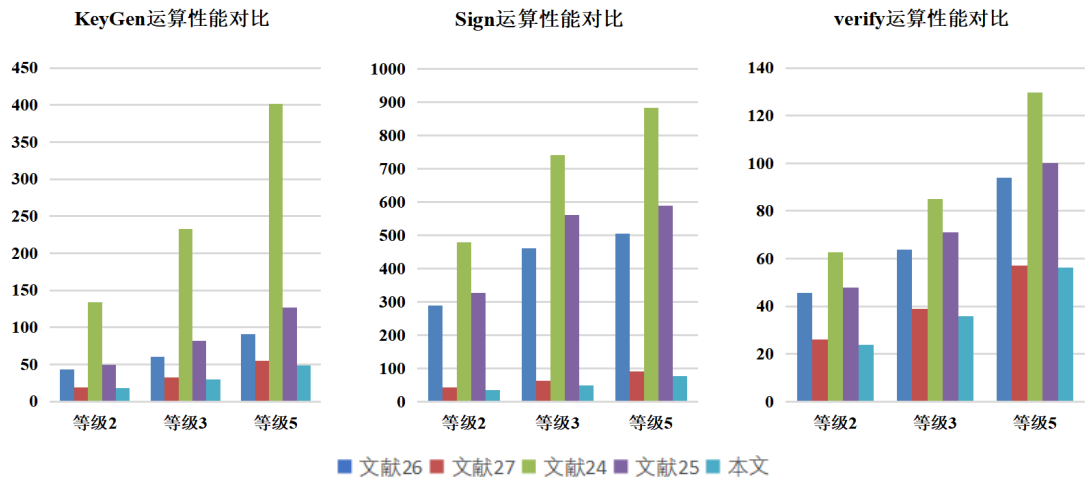

图 9 设计方案性能对比

5 总结

本文针对 NIST 第三轮的格基数字签名算法 Dilithium 提出了一种全新的纯硬件实现方案,基于可重构的思想设计并实现同时满足三组参数集条件下密钥生成运算、数字签名运算和签名验证运算的统一型电路结构.本文提出的细粒度并行分段式时序控制逻辑有效匹配了脉动阵列单元的高吞吐率优势,同时根据采样运算的特点给出全新的双 Keccak 核优化设计方案,使得最终的硬件设计架构同时具备了高性能和高资源复用率两方面的优势.对于推进格密码方案的工程化和实用化进程有较好的借鉴意义和参考价值.

References:

[1] Diffie W, Hellman ME. New directions in cryptograph. IEEE Transactions on Information Theory, 1976, 22(6): 644 – 654.[doi: 10.1109/TIT.1976.1055638]

[2] Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review, 1999,41(2): 303–332. [doi: 10.1137/S0036144598347011]

[3] Shor PW. Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994). Piscataway, NJ: IEEE, 1994: 124 – 134.[doi: 10.1109/SFCS.1994.365700]

[4] Grassl M, Langenberg B, Roetteler M, Steinwandt R. Applying Grover's algorithm to AES: quantum resource estimates. In: International Workshop on Post-Quantum Cryptography. Cham: Springer International Publishing, 2016: 29-43.[doi: 10.1007/978-3-319-29360-8_3]

[5] NIST. Post-Quantum Cryptography Call for Proposals. 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals.

[6] Chinese Association for Cryptologic Research. Public key algorithms selected to the second round competition of national cyptographic algorithm competitions. 2019. https://sfjs.cacrnet.org.cn/site/term/list_77_1.html

[7] NIST. Post-Quantum Cryptography Round 3 Submissions. 2020. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[8] Kumari S, Singh M, Singh R, Tewari H. Signature based Merkle Hash Multiplication algorithm to secure the communicat ion in IoT devices. Knowledge-Based Systems, 2022, 253: 109543.[doi: 10.1016/j.knosys.2022.109543]

[9] McEliece RJ. 1978. A public-key cryptosystem based on algebraic. Coding Thv 4244 (1978), 114–116.

[10] Courtois N, Klimov A, Patarin J, Shamir A. Efficient algorithms for solving overdefined systems of multivariate polynomi al equations. In: International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2000:392–407.[doi: 10.1007/3-540-45539-6_27]

[11] Peng C, Chen JH, Zeadally S, and He D. Isogeny-Based Cryptography: A Promising Post-Quantum Technique. IT Profess ional, 2019, 21(6), 27–32.[doi: 10.1109/MITP.2019.2943136]

[12] Nejatollahi H, Dutt N, Ray S, Regazzoni F, Banerjee I, Cammarota R. Post-quantum lattice-based cryptography implement ations: A survey. ACM Computing Surveys (CSUR), 2019, 51(6): 1-41.[doi: 10.1145/3292548]

[13] Avanzi R, Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, Schwabe P, Seiler G, Stehlé D. CRYSTA LS-Kyber: Algorithm specifications and supporting documentation (version 3.01). 2021. https://pq-crystals.org/kyber/data/kyb er-specificationround3-20210131.pdf

[14] Bai S, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, Stehlé D. CRYSTALS-Dilithium algorithm s pecifications and supporting documentation (version 3.1). 2021. https://pq-crystals.org/dilithium/data/dilithium-specification-ro und3-20210208.pdf

[15] Fouque PA, Hoffstein J, Kirchner P, Lyubashevsky V, Pornin T, Prest T, Ricosset T, Seiler G, Whyte W, Zhang ZF. Fal con: Fast-Fourier lattice-based compact signatures over NTRU (specification v1.2). 2020. https://falcon-sign.info/falc on.pdf

[16] NIST. PQC standardization process: Announcing four candidates to be standardized, plus fourth round candidates. 2022. ht tps://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4

[17] NIST. Module-Lattice-Based Digital Signature Standard. 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf

[18] Langlois A, Stehlé D. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 2015, 75(3) 565–599.[doi: 10.1007/s10623-014-9938-4]

[19] Kannwischer MJ, Rijneveld J, Schwabe P, Stoelen K. pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4. 2019. https://eprint.iacr.org/2019/844

[20] Kim Y, Song J, Youn TY, Seo SC. Crystals-Dilithium on ARMv8. Security and Communication Networks, 2022, 2022(1): 5226390.[doi: 10.1155/2022/5226390]

[21] Zheng JY, He F, Shen SY, Xue CX, Zhao YL. Parallel Small Polynomial Multiplication for Dilithium: A Faster Design and Implementation. In: Proceedings of the 38th Annual Computer Security Applications Conference. Austin: ACM,2022:3 04-317.[doi: 10.1145/3564625.3564629]

[22] Soni, D., Basu, K., Nabeel, M.M., & Karri, R. (2019). A Hardware Evaluation Study of NIST Post-Quantum Cryptographic Signature schemes. In: Second PQC Standardization Conference. NIST, 2019. https://api.semanticscholar.org/CorpusID:198939541

[23] Ricci S, Malina L, Jedlicka P, Smékal D, Hajny J, Cibik P, Dzurenda P, Dobias P. Implementing CRYSTALS-Dilithium Signature Scheme on FPGAs. In: Proceedings of the 16th International Conference on Availability, Reliability and Security. Austria: ACM, 2021: 1-11.[doi: 10.1145/3465481.3465756]

[24] Land G, Sasdrich P, Güneysu T. A hard crystal-implementing dilithium on reconfigurable hardware. In: International Conference on Smart Card Research and Advanced Applications. Germany: Springer, 2021: 210-230.[doi: 10.1007/978-3-030-97348-3_12]

[25] Wang T, Zhang C, Cao P, Gu Dawu. Efficient implementation of dilithium signature scheme on fpga soc platform. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022, 30(9):1158-1171. [doi: 10.1109/TVLSI.2022.3179459]

[26] Zhao C, Zhang N, Wang H, Yang B, Zhu W, Li Z, Zhu M, Yin S, Wei S, Liu L. A Compact and High-Performance Hardware Architecture for CRYSTALS-Dilithium. IACRTransactions on Cryptographic Hardware and Embedded Systems, 2022, 2022(1): 270-295.[doi: 10.46586/tches.v2022.i1.270-295]

[27] Beckwith L, Nguyen DT, Gaj K. High-Performance Hardware Implementation of CRYSTALS-Dilithium. In: 2021 International Conference on Field-Programmable Technology (ICFPT 2021). Auckland: IEEE, 2021: 1–10. [doi: 10.1109/ICFPT52863.2021.9609917]

[28] Barrett P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In: Odlyzko AM, ed. Proc. of the 1987 Conf. on the Theory and Application of Cryptographic Techniques (CRYPTO 1986). Berlin,Heidelberg: Springer, 1987. 311–323. [doi: 10.1007/3-540-47721-7_24]

[29] Montgomery PL. Modular multiplication without trial division. Mathematics of Computation, 1985, 44(170): 519–521.[doi: 10.1090/S0025-5718-1985-0777282-X]

[30] Banerjee U, Ukyab TS, Chandrakasan AP. Sapphire: A configurable cryptoprocessor for post-quantum lattice-based protocols (extended version). IACR Cryptology ePrint Archive,2019: 1140. [doi: 10.13154/tches.v2019.i4.17-61]

[31] Zhou Z, He D, Liu Z, Luo M, Choo KR. A Software/Hardware Co-Design of Crystals-Dilithium Signature Scheme. ACM Transactionson Reconfigurable Technology and Systems, 2021, 14(2): 1–21.[doi: 10.1145/3447812]

[32] Zhao Y, Xie R, Xin G, Han J. A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(7): 2871-2884.[doi: 10.1109/TCSI.2022.3162593]

[33] Xin G, Han J, Yin T, Zhou Y, Yang J, Cheng X, Zeng X. VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-Varchitecture. IEEE transactions on circuits and systems I: regular papers, 2020, 67(8): 2672-2684.[doi: 10.1109/TCSI.2020.2983185]

[34] Wang T, Zhang C, Zhang X, Zhang X, Gu D, Cao P. Optimized Hardware-Software Co-Design for Kyber and Dilithium on RISC-V SoC FPGA[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024, 2024(3): 99-135. [doi: 10.46586/tches.v2024.i3.99-135]

[35] Mao G, Chen D, Li G, Dai W, Sanka AI, Koc CK, Cheung RC. High-performance and configurable SW/HW co-design of Post-Quantum Signature CRYSTALS-Dilithium. ACM Transactions on Reconfigurable Technology and Systems, 2023, 16(3): 1-28.[doi: 10.1145/3569456]

[36] Aikata A, Mert AC, Jacquemin D, Das A, Matthews D, Ghosh S. A unified cryptoprocessor for lattice-based signature and key-exchange. IEEE Transactions on Computers, 2022, 72(6): 1568-1580.[doi: 10.1109/TC.2022.3215064]

[37] Mert A C, Karabulut E, Öztürk E, Savaş E, Aysu A. An extensive study of flexible design methods for the number theoretic transform. IEEE Transactions on Computers, 2020, 71(11): 2829-2843.[doi: 10.1109/TC.2020.3017930]

[38] Yaman F, Mert A C, Öztürk E, Savaş E. A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). Grenoble: IEEE, 2021,1020-1025.[doi: 10.23919/DATE51398.2021.9474139]

[39] Zhao XY, L ZC, Hu Y, Geng HX, Z YL. Research on NTT architecture and FPGA hardware optimization implementatio n. Journal of Software, 2023, 46(12): 2670-2686(in Chinese with English abstract).[doi: 10.11897/SP.J.1016.2023.02670]

[40] Xing Y, Li S. A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FP GA[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021: 328-356.[doi: 10.46586/tches.v2021.i 2.328-356]

[41] Hu Y, Zhao XY, L YX, Zhao YL. An efficient hardware implementation of lattice-based key encapsulation algorithms O SKR/OKAI. Chinese Journal of Computers, 2023, 46(6): 1156-1171(in Chinese with English abstract).[doi: 10.11897./SP.J. 1016.2023.01156]

[42] Mookherjee S, DeBrunner L, DeBrunner V. A low power radix-2 FFT accelerator for FPGA. In: 2015 49th Asilomar Co nference on Signals, Systems and Computers. Pacific Grove,CA: IEEE, 2015: 447-451.[doi: 10.1109/ACSSC.2015.7421167]

[43] Dang V B, Mohajerani K, Gaj K. High-speed hardware architectures and FPGA benchmarking of CRYSTALS-Kyber, NT RU, and Saber[J]. IEEE Transactions on Computers, 2022, 72(2): 306-320.[doi: 10.1109/TC.2022.3222954]

附中文参考文献:

[39] 赵旭阳, 梁志闯, 胡跃, 耿合详,赵运磊. NTT 架构研究及其 FPGA 硬件优化实现 [J]. 计算机学报, 2023(12):2670-2686.[do i: 10.11897/SP.J.1016.2023.02670]

[41] 胡跃,赵旭阳, 刘裕雄, 赵运磊. 格基密钥封装算法 OSKR/OKAI 硬件高效实现 [J]. 计算机学报, 2023, 46(6):1156-1171.[do i: 10.11897./SP.J.1016.2023.01156]