Crystals-Dilithium数字签名技术硬件实现综述

崔益军,李梦雪,王 辈,王成华,刘伟强
(南京航空航天大学集成电路学院,江苏南京 210016)

摘 要: 随着量子计算技术的不断发展,依赖传统公钥密码体制三大功能(密钥协商/数字签名/公钥加密)的各种应用系统将不再安全. 为应对量子威胁,以美国国家标准与技术研究院(National Institute of Standards and Technology,NIST)为首的国际标准组织积极征集与部署后量子密码(Post Quantum Cryptography,PQC)算法的标准化工作,致力于在真正实用型量子计算机问世之前,提前完成传统公钥密码算法到PQC算法的迁移过渡. Crystals-Dilithium是NIST-PQC标准中的一种基于格的数字签名算法,其安全性高,运算速度快,是实现抵抗量子攻击数字签名算法的重要路径之一. 本文从主流Crystals-Dilithium数字签名算法的理论基础出发,从底层关键组件的优化方法和整体硬件构架设计方法着手,围绕硬件资源优化和性能优化等现有方法和成果对比展开分析介绍,为研究者们后续研究探明方向,希望为设计性能与硬件资源均衡的后量子数字签名密码芯片提供有力参考.

关键词: 后量子密码;格密码;Crystals-Dilithium数字签名;硬件实现;优化方案
基金项目: 国家自然科学基金(No.62134002,No.62104107,No.62371226)
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2025)07-2558-21
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240505

A Survey of Hardware Implementation for Crystals-Dilithium Digital Signature in the Post Quantum Cryptography

CUI Yi-jun, LI Meng-xue, WANG Bei, WANG Cheng-hua, LIU Wei-qiang
(*College of Integrated Circuits*, *Nanjing University of Aeronautics and Astronautics*, *Nanjing*, *Jiangsu* 210016, *China*)

Abstract: With the continuous development of quantum computing technology, various application systems relying on the three major functions of traditional public key cryptography (key agreement/digital signatures/public key encryption) will no longer be secure. In response to the quantum threat, international standardization organizations led by the United States NIST (National Institute of Standards and Technology) are actively soliciting and deploying the standardization work of post-quantum cryptography (PQC) algorithms, aiming to complete the migration from traditional public key cryptography algorithms to PQC algorithms before truly practical quantum computers emerge. Crystals-Dilithium is one of the lattice-based digital signature algorithms in the NIST-PQC standard, which features high security and fast computation speed, making it an important path to implement digital signature algorithms resistant to quantum attacks. This paper commences with the theoretical foundations of the mainstream Crystals-Dilithium digital signature algorithm. It delves into optimization methods for underlying key components and overall hardware architecture design, focusing on hardware resource optimization and performance enhancement. The paper contrasts and analyzes existing methods and outcomes, aiming to clarify the direction for subsequent research. It aspires to provide a robust reference for the design of post-quantum digital signature cryptographic chips that balance performance with hardware resource efficiency.

Key words: post quantum cryptography; lattice-based cryptography; Crystals-Dilithium digital signature; hardware implementation; optimization scheme
Foundation Item(s): National Natural Science Foundation of China (No.62134002, No.62104107, No.62371226)

收稿日期:2024-06-02;录用日期:2025-04-23;责任编辑:宋甲英

1 引言

当今信息安全领域广泛使用的公钥密码体制主要是基于难以求解的数学困难问题而构造的. 例如:Rivest-Shamir-Adleman[1]是基于大整数分解问题,Diffie-Hellman[2]和ElGamal[3]是基于有限域上离散对数问题,椭圆曲线密码(Elliptic Curve Cryptography,ECC)算法[4]是基于椭圆曲线离散对数问题. 在经典的计算模型下,这些底层数学困难问题的攻击复杂度至少是亚指数级的[5]. 1994年,美国数学家Peter Shor提出Shor算法,能够在多项式时间内求解大整数分解问题和离散对数问题[6]. 这表明一旦真正实用型量子计算机问世,现有传统公钥密码体制的安全性将面临坍塌的风险. 为应对量子计算机对传统公钥密码体制带来的挑战,2016年12月,美国国家标准技术研究院(National Institute of Standards and Technology,NIST)开展后量子密码学(Post-Quantum Cryptography,PQC)算法标准征集工作[7],并计划通过几轮的标准化竞争过程评选出能够抵抗量子攻击的公钥密码体制. NIST的标准化评定对于算法的评估主要涵盖三个方向:一是评估候选密码算法的安全性,包括算法的抗攻击性、密码强度以及对各种安全威胁的防护能力;二是评估算法的运行效率,包括密钥生成速度、封装/解封装速度、签名/验签速度、内存消耗等指标;三是评估算法的实际应用性,包括算法的实现难度、对硬件和软件的适应性,以及算法的易用性和可扩展性. 经过多轮评审,NIST于2022年7月公布了被采纳使用且待标准化的PQC方案①,宣布Crystals-Kyber、Crystals-Dilithium、FALCON和SPHINCS+这4种PQC算法将进入待标准化进程中,其中基于格的PQC方案有Crystals-Kyber和Crystals-Dilithium,前者是实现密钥封装机制,而后者是实现数字签名协议. 2023年8月,NIST发布3份后量子密码算法标准草案,将Crystals-Kyber优化并更名为ML-KEM,Crystals-Dilithium优化并更名为ML-DSA(Modular Lattice-based Digital Signature Algorithm,ML-DSA),SPHINCS+优化并更名为SLH-DSA(StateLess Hash-based Digital Signature Algorithm). 2024年8月NIST正式确定此3个算法为标准后量子密码算法方案. 2024年11月,NIST发布了过渡到后量子密码学标准的初始公开草案《Transition to Post-Quantum Cryptography Standards》②,包含迁移的路线与时间表. NIST希望到2035年将政府机构的加密系统替换为后量子密码体制.

后量子密码方案所基于的数学原理主要包括以下五种:基于格(Lattice-based)[8]、基于哈希(Hash-based)[9]、基于编码(Code-based)[10]、基于多变量(Multivariate-based)[11]和基于超奇异同源(Isogeny-based)[12]. 由于不同的底层数学结构和设计门限方案,这些密码方案具有差异化的参数和性能. 表1从底层数学结构、优点、缺点和发展情况这四个方面介绍了主流后量子密码算法,可以看出基于格的PQC算法相较于其他算法具有更好的表现,在安全性、公私钥大小、运算速度上可达到较好的平衡,是目前最有应用前景的PQC算法之一.

在PQC征集过程中,一些PQC算法不断遭受挑战. 如2022年2月,晋级NIST的PQC决赛第三轮的Rainbow算法在其某个参数设置下被笔记本电脑成功破解[13];同年4月,3款基于LWE(Learning With Errors)的NIST后量子密码算法被曝再遭挑战[14];2022年7月31日,比利时鲁汶大学的研究人员[15]通过Magma计算软件,彻底破解了基于超奇异同源的密钥协商算法超奇异同源密钥封装(Supersingular Isogeny Key Encapsulation,SIKE). 截至目前,基于格的后量子密码算法依然被国内外研究学者和企业高度认可. 近些年,针对格密码算法(Crystals-Kyber和Crystals-Dilithium)软硬件实现的研究与应用越来越多,随之也面临迁移过程中的软硬件灵敏性和适配性等问题.

2021年12月,全球领先的半导体企业进阶精简指令集机器(Advanced RISC Machines,ARM)推出了关于后量子密码技术的白皮书③,从中可以看到国际芯片企业对未来网络空间安全基础技术的前瞻性布局. 事实上,包括Intel等国际芯片企业在内,均早已在关注未来芯片的抗量子计算的攻击. 2022年4月,美国众议院推动联邦政府信息技术系统向PQC算法过渡,我国也正在积极推动相关标准化制定和产业应用. 随着NIST对PQC算法标准化的推进,关于PQC算法的硬件实现研究也步入了快速发展的轨道,但是从当今的公钥密码体制向PQC算法迁移是一项既耗时又具有挑战性的工作. 最近的一些报道也表明,一些领先的科技公司正在加大对硬件PQC实现的研发和投入力度,这一举措被视为未来网络安全战略的重要组成部分. 因此,可以预见,硬件实现将成为未来PQC迁移的关键推动力量,为各行业提供更为高效、更为安全的解决方案,从而保障了数据的隐私和安全. 目前,在PQC算法的硬件实现方面相关研究较少. 在算法的硬件设计过程中,仍然需要在算法及计算架构层面,解决性能资源、速度和功耗不均衡,兼容性和安全性低等一系列的硬件实现问题.

考虑到PQC在从云到端的部署与应用中都离不开PQC芯片的支撑,因此设计出实用性、灵活性、高效性

①https://www.nist.gov/publications/status-report-second-round-nist-post-quantum-cryptography-standardization-process

②https://csrc.nist.gov/pubs/ir/8547/ipd

③https://community.arm.com/arm-research/m/resources/1002

表1 主流后量子密码算法

类别	安全性依赖的数学问题	优点	缺点	发展情况
基于格	依赖于格中困难问题	公私钥尺寸小、计算速度快,且能被用于构造多种密码学原语,因此更适用于实际应用环境	密钥封装的交互过程中,传输的密文尺寸较大	格密码算法被广泛认为是PQC算法征集中的主力军,目前已有两个格密码算法被标准化,未来Falcon也在标准化进程中
基于哈希	依赖于哈希算法的一些安全性质,例如单向性(抗原像攻击)、弱抗碰撞性(抗第二原像攻击)和伪随机性等	公钥很小、安全假设少、安全性强	签名尺寸较长,功能上只能构造签名	基于哈希的密码算法更适用于数字签名,但底层数学结构少,目前Sphincs+已被标准化
基于编码	使用错误纠正码对加入的随机性错误进行纠正和计算,对于随机线性码的解码问题是非确定性多项式时间 (Non-deterministic Polynomial time,NP)难	底层数学困难问题研究时间长,安全性更经得起考验	公钥大、密钥生成慢	后量子密码中相对具有竞争力的密码算法,但在实用化方面有待优化提升
基于多变量	求解有限域上非线性方程组NP难问题	签名验签速度快、消耗资源少	公钥大	NIST-PQC的附加数字签名方案第二轮中提交了多个基于多变量的密码算法,在多变量赛道大概率有算法被标准化
基于超奇异同源	寻找任意两条椭圆曲线之间的同源问题	公私钥或签名验签的尺寸较小	计算速度慢、代数结构复杂、缺乏可证明安全性	SIKE被攻破,不过交换超奇异同源迪菲-赫尔曼 (Commutative Supersingular Isogeny Diffie-Hellman,CSIDH)和超奇异二次同源签名 (Supersingular Quadratic Isogeny signatures,SQIsign算法仍未被攻破,且SQIsign已进入NIST-PQC的附加数字签名方案第二轮

与安全性有机统一的PQC芯片至关重要. 在当前网络信息互联时代,庞大的电子商务体系、银行和金融服务、电子数字签名与网址安全访问等应用广泛,其底层大量使用数字签名算法. 未来PQC数字签名体制必将应用到各行各业,而其签名算法硬件实现的资源和性能是衡量PQC数字签名算法应用落地的重要指标. 本文从主流Crystals-Dilithium数字签名算法的理论基础出发,从底层关键组件的优化方法和整体硬件构架设计方法着手,围绕硬件资源优化和性能优化等现有方法和成果对比展开综述,为数字签名芯片的设计实现与应用提供参考.

2 背景介绍

本章节将围绕Crystals-Dilithium数字签名算法的理论展开介绍. 首先,介绍本章节算法中出现的符号,接着介绍算法的详细实现流程,最后对算法中的环多项式乘法算法进行总结分析,为后续章节中Crystals-Dilithium数字签名算法的高效硬件实现与关键技术研究提供理论依据.

2.1 符号说明

本节主要用到的符号说明:设n和q是两个整数,n表示为2的幂,q表示为满足$q\equiv 1(\mathrm{mod}\ 2n)$的质数. 设$\mathbb{Z}$表示整数的集合,$R$表示环多项式环$\mathbb{Z}[x]/(x^n+1)$,其中$(x^n+1)$是$\mathbb{Z}[x]$的理想. 另外$\mathbb{Z}_q$定义为$\mathbb{Z}/q\mathbb{Z}$,$R_q$是由$\mathbb{Z}_q[x]/(x^n+1)$得到的商环. R(或R_q)中的元素为多项式,为了便于区分,这里用斜体化的大写字母表示,例如F,相反,向量用粗体的、斜体化的小写字母表示,如$\boldsymbol{x}$. 每个多项式元素,无论是$F\in R$还是$F\in R_q$,都可以只表示为$F=\sum_{i=0}^{n-1}F_iX^i$,其中F_i属于$i\in[0,n)$的$\mathbb{Z}$(或$\mathbb{Z}_q$). R(或R_q)中的矩阵用粗体的、斜体化的大写字母表示,例如$\boldsymbol{A}$. 对于多项式$F\in R$,它的无穷范数可以表示为$\|F\|_\infty=\max\{|F_i|\}$,同样的对于一个向量$\boldsymbol{x}=(\boldsymbol{x}_0,\boldsymbol{x}_1,\cdots,\boldsymbol{x}_{n-1})\in R^n$,它的无穷范数可以表示为$\|\boldsymbol{x}\|_\infty=\max\limits_{0\leqslant i\leqslant n-1}|\boldsymbol{x}_i|$.

2.2 Crystals-Dilithium数字签名算法

Crystals-Dilithium是一种基于格的数字签名算法,算法采用在环多项式$R_q=\mathbb{Z}_q[x]/(x^{256}+1)$上的Fiat-Shamir范式构造[16]. 其安全性基于模误差学习问题(Module-Learning With Errors,M-LWE)和模短整数解(Module-Short Integer Solution,M-SIS)问题[17]. 2023年8月,NIST发布后量子密码学标准草案,Crystals-Dilithium方案被修改和重新命名为ML-DSA,但其算法的整体架构与流程未发生改变,仅为了确保ML-DSA满

足特定的安全属性，如不可伪造性和非否认性，针对Crystals-Dilithium的某些参数进行了调整，例如增加了公钥tr的长度到512位，以及增加了$\tilde{c}$的长度到384位和512位等．为了保持算法的最新性，本文介绍ML-DSA的算法流程，包括密钥生成、签名及验签环节．

表2展示了三组ML-DSA参数集的算法参数．每个参数集都用于ML-DSA算法的密钥生成、签名和验证的所有参数分配值．表中第1行的参数集ML-DSA-44属于安全等级2，ML-DSA-65属于安全等级3，而ML-DSA-87属于安全等级5. 三种不同的参数集中模数(q)、采样系数(d)相同．其中，(k,l)代表着多项式矩阵$\boldsymbol{A}$的维度，通过改变(k,l)的值，可以调整签名的安全等级．η代表着私钥的范围，β是验证 if $c=\mathrm{H}\left(\rho,\boldsymbol{t}_1,\boldsymbol{w}_1,\mu\right)$ and $\|\boldsymbol{z}\|_\infty<\gamma_1-\beta$环节中的范数误差，$\omega$代表着提示$\boldsymbol{h}$中的最大值，Repetitions代表着签名算法主循环的预期重复次数．从表中可以看出，随着安全等级不断上升，范数误差的值在逐渐减小，签名信息也会更加准确．

表2　ML-DSA数字签名算法参数列表

序号	参数	ML-DSA-44	ML-DSA-65	ML-DSA-87
1	Claimed Security Level	2	3	5
2	(k,l)	(4,4)	(6,5)	(8,7)
3	q	8 380 417	8 380 417	8 380 417
4	d	13	13	13
5	τ	39	49	60
6	λ	128	192	256
7	γ_1	217	219	219
8	γ_2	$(q-1)/88$	$(q-1)/32$	$(q-1)/32$
9	η	2	4	2
10	$\beta=\tau\cdot\eta$	78	196	120
11	ω	80	55	75
12	$\log\binom{256}{\tau}+\tau$	192	225	257
13	Repetitions	4.25	5.1	3.85

密钥生成过程．密钥生成过程主要完成字节字符串公钥pk和私钥sk的生成，如算法1所示．首先，算法生成一个256的随机种子ξ，通过SHAKE-256哈希函数来产生其他随机值ρ、ρ'、K. 其中，公共的随机256位种子ρ经过一个伪随机采样函数ExpandA将其展开到多项式矩阵$\boldsymbol{A}\in R_q^{k\times l}$中并通过后续计算转化为数论变换(Number Theoretic Transform，NTT)的表达式$\hat{\boldsymbol{A}}$，其中$\hat{\boldsymbol{A}}$矩阵的项都是R_q环中的多项式．接着通过伪随机采样函数ExpandS展开另一个私有的随机256位种子ρ'，产生满足在$[-\eta,\eta]$之间具有均匀随机系数的密钥向量$\boldsymbol{s}_1$和$\boldsymbol{s}_2$. 紧接着，通过NTT矩阵多项式计算操作得到公共值$\boldsymbol{t}=\boldsymbol{A}\boldsymbol{s}_1+\boldsymbol{s}_2$，再通过Power 2Round函数将其分解为高低位，其中$\boldsymbol{t}$的低阶比特可以从少量签名中重建，因此为了提升运算的性能，算法将每个系数中d个最不重要的低阶比特丢弃从而产生压缩多项式系数向量$\boldsymbol{t}_1$. 最后算法通过字符串编码操作译为公钥pk和私钥sk.

算法1　ML-DSA.KeyGen()

输出：公钥$pk\in$ B,私钥$sk\in$ B

1. $\xi\leftarrow\{0,1\}^{256}$
2. $(\rho,\rho',K)\in\{0,1\}^{256}\times\{0,1\}^{512}\times\{0,1\}^{256}\leftarrow\mathrm{H}(\xi,1\,024)$
3. $\hat{\boldsymbol{A}}\leftarrow\mathrm{ExpandA}(\rho)$
4. $(\boldsymbol{s}_1,\boldsymbol{s}_2)\leftarrow\mathrm{ExpandS}(\rho')$
5. $\boldsymbol{t}\leftarrow\mathrm{NTT}^{-1}(\hat{\boldsymbol{A}}\circ\mathrm{NTT}(\boldsymbol{s}_1))+\boldsymbol{s}_2$
6. $(\boldsymbol{t}_1,\boldsymbol{t}_0)\leftarrow\mathrm{Power\,2Round}(\boldsymbol{t},d)S$
7. $pk\leftarrow\mathrm{pkEncode}(\rho,\boldsymbol{t}_1)$
8. $tr\leftarrow\mathrm{H}(\mathrm{BytesToBits}(pk),512)$
9. $sk\leftarrow\mathrm{skEncode}(\rho,K,tr,\boldsymbol{s}_1,\boldsymbol{s}_2,\boldsymbol{t}_0)$
10. RETURN(pk,sk)

签名过程．签名过程将被编码为字节串的私钥sk和位字符串消息M作为输入，为消息M生成一个签名σ，如算法2所示．首先签名者从私钥sk中通过skDecode函数提取所需要的ρ、K、tr、$\boldsymbol{s}_1$、$\boldsymbol{s}_2$、$\boldsymbol{T}_0$信息，其中ρ是256位的公共随机种子；K是256位的私有随机种子，tr是512位的公钥的哈希值；$\boldsymbol{s}_1$、$\boldsymbol{s}_2$是私钥的多项式向量；$\boldsymbol{t}_0$是多项式向量$\boldsymbol{t}$的每个系数d的最小有效位多项式向量．接着，签名者将$\boldsymbol{s}_1$、$\boldsymbol{s}_2$、$\boldsymbol{t}_0$通过计算转化为NTT的表达式$\hat{\boldsymbol{s}}_1$、$\hat{\boldsymbol{s}}_2$、$\hat{\boldsymbol{t}}_0$，展开ρ到与密钥生成中相同的矩阵$\boldsymbol{A}\in R_q^{k\times l}$并转化为NTT的表达式$\hat{\boldsymbol{A}}$. 在消息$M$被签名之前，签名者将它与公钥$tr$哈希连接，并使用哈希函数H生成512位消息代表$\mu$，接着计算私有随机种子$\rho'$. 签名算法的主要部分为生成签名的拒绝采样循环，如果签名可能泄露私钥信息，则判断为无效签名，算法将重复循环，直到产生有效签名，然后将其编码为字节字符串并输出．拒绝采样循环所涉及的主要步骤有：首先，算法签名者使用ExpandMask函数、种子ρ'和一个计数器κ，从多项式向量的子集中伪随机地采样系数在$[-\gamma_2,\gamma_2]$范围内的多项式向量$y\in R_q^l$. 在获得签名者样本$\boldsymbol{y}$后，签名者首先计算$\boldsymbol{w}=\boldsymbol{A}\boldsymbol{y}$，再根据公式$\boldsymbol{w}=2\gamma_2\cdot\boldsymbol{w}_1+\boldsymbol{w}_0$，通过使用HighBits函数，四舍五入到最接近的$2\gamma_2$的倍数来计算承诺$\boldsymbol{w}_1$；在计算完$\boldsymbol{w}_1$后，连接消息μ和编码过后的$\boldsymbol{w}_1$值，利用哈希函数创建其散列值并转化哈希形式$\tilde{c}$，再用$\tilde{c}_1$表示$\tilde{c}$的前256位．比特串$\tilde{c}_1$被用来伪随机地采样一个多项式$\boldsymbol{c}\in R_q$，其系数为−1、0或1，且非零系数的数量为τ. 在得到散列值$\boldsymbol{c}$后，计算签名$\boldsymbol{z}=\boldsymbol{y}+\boldsymbol{c}\boldsymbol{s}_1$，如果此时直接输出$\boldsymbol{z}$，将可能导致密钥被泄

露,那么签名方案将是不安全的. 因此,为了避免$\boldsymbol{z}$会泄露密钥的信息,算法使用拒绝抽样的方式检验前面是否有效. 如果检查通过,签名者可以计算一个提示多项式$\boldsymbol{h}$,这将允许验证者使用压缩的公钥(以及签名的其他组件)重构$\boldsymbol{w}_1$. 最后签名者输出最终的签名,即承诺$\tilde{c}$、响应$\boldsymbol{z}$和提示$\boldsymbol{h}$的字节编码σ.

验签过程. 验签过程将被编码为字节字符串的公钥pk、位字符串的消息M以及字节字符串的签名σ作为输入,验证消息M的签名σ是否正确,如算法3所示. 验证者首先从公钥pk中提取公共随机种子ρ和压缩多项式向量$\boldsymbol{t}_1$;接着从签名σ中提取签名者的承诺$\tilde{c}$、响应$\boldsymbol{z}$和提示$\boldsymbol{h}$. 如果验证者发现提示h未被正确地字节编码,则用符号"⊥"表示,验证算法立即返回false,表示签名无效. 假设签名成功地从其字节编码中提取出来,验证者通过伪随机方法从公共随机种子ρ中拓展多项式矩阵$\hat{\boldsymbol{A}}$,并通过对pk的哈希结果tr和消息pk的串联进行哈希运算来创建一个消息代表μ. 然后,验证者尝试根据公钥pk和签名σ重构签名者的多项式向量承诺$\boldsymbol{w}_1$,其中重构的值被称为$\boldsymbol{w}_1'$. 最后,验证者检查签名者的响应$\boldsymbol{z}$和签名者的提示$\boldsymbol{h}$是否有效以及重构的$\boldsymbol{w}_1'$与签名者的承诺哈希$\tilde{c}$是否一致,即检查:(1)所有系数是否足够小(即在范围$\left[-\gamma_2,\gamma_2\right]$内);(2)$\boldsymbol{h}$是否包含不超过$w$个非零系数;(3)$\tilde{c}$是否与消息代表$\mu$串联和表示为位字符串的$\boldsymbol{w}_1'$的哈希$\tilde{c}'$匹配. 如果所有这些检查都成功,则返回true,否则返回false.

算法2 ML-DSA.Sign(sk,M)

输入:私钥$sk\in$B,消息$M\in\{0,1\}^*$

输出:签名$\sigma\in$B

1. $(p,K,tr,\boldsymbol{s}_1,\boldsymbol{s}_2,\boldsymbol{t}_0)\leftarrow$skDecode($sk$)
2. $\hat{\boldsymbol{s}}_1\leftarrow$NTT($\boldsymbol{s}_1$)
3. $\hat{\boldsymbol{s}}_2\leftarrow$NTT($\boldsymbol{s}_2$)
4. $\hat{\boldsymbol{t}}_0\leftarrow$NTT($\boldsymbol{t}_0$)
5. $\hat{\boldsymbol{A}}\leftarrow$ExpandA$(\rho)$
6. $\mu\leftarrow$H(tr // M,512)
7. $rnd\leftarrow\{0,1\}^{256}$
8. $\rho'\leftarrow$H(K // rnd // μ,512)
9. $\kappa\leftarrow 0$
10. $(\boldsymbol{z},\boldsymbol{h})\leftarrow\perp$
11. WHILE $(\boldsymbol{z},\boldsymbol{h})=\perp$ DO
12. $\quad\boldsymbol{y}\leftarrow$ExpandMask$(\rho',\kappa)$
13. $\quad\boldsymbol{w}\leftarrow\mathrm{NTT}^{-1}(\hat{\boldsymbol{A}}\circ\mathrm{NTT}(\boldsymbol{y}))$
14. $\quad\boldsymbol{w}_1\leftarrow$HighBits($\boldsymbol{w}$)
15. $\quad\tilde{c}\in\{0,1\}^{2\lambda}\leftarrow$H($\mu$ // w1Encode($\boldsymbol{w}_1$),2λ)
16. $\quad(\tilde{c}_1,\tilde{c}_2)\in\{0,1\}^{256}\times\{0,1\}^{2\lambda-256}\leftarrow\tilde{c}s$
17. $\quad\boldsymbol{c}\leftarrow$SampleInBall($\tilde{c}_1$)
18. $\quad\hat{\boldsymbol{c}}\leftarrow$NTT($\boldsymbol{c}$)
19. $\quad\langle\langle\boldsymbol{cs}_1\rangle\rangle\leftarrow\mathrm{NTT}^{-1}(\hat{\boldsymbol{c}}\circ\hat{\boldsymbol{s}}_1)$
20. $\quad\langle\langle\boldsymbol{cs}_2\rangle\rangle\leftarrow\mathrm{NTT}^{-1}(\hat{\boldsymbol{c}}\circ\hat{\boldsymbol{s}}_2)$
21. $\quad\boldsymbol{z}\leftarrow\boldsymbol{y}+\langle\langle\boldsymbol{cs}_1\rangle\rangle$
22. $\quad\boldsymbol{r}_0\leftarrow$LowBits$(\boldsymbol{w}-\langle\langle\boldsymbol{cs}_2\rangle\rangle)$
23. $\quad$IF $\|\boldsymbol{z}\|_\infty\geqslant\gamma_1-\beta$ or $\|\boldsymbol{r}_0\|_\infty\geqslant\gamma_2-\beta$ THEN $(\boldsymbol{z},\boldsymbol{h})\leftarrow\perp$
24. $\quad$ELSE
25. $\quad\quad\langle\langle\boldsymbol{ct}_0\rangle\rangle\leftarrow\mathrm{NTT}^{-1}(\hat{\boldsymbol{c}}\circ\hat{\boldsymbol{t}}_0)$
26. $\boldsymbol{h}\leftarrow$MakeHint$\left(-\langle\langle\boldsymbol{ct}_0\rangle\rangle,\boldsymbol{w}-\langle\langle\boldsymbol{cs}_2\rangle\rangle+\langle\langle\boldsymbol{ct}_0\rangle\rangle\right)$
27. $\quad\quad$IF $\|\boldsymbol{ct}_0\|_\infty\geqslant\gamma_2$ or the number of 1's in h is greater than ω THEN $(\boldsymbol{z},\boldsymbol{h})\leftarrow\perp$
28. $\quad\quad$END IF
29. $\quad$END IF
30. $\quad\kappa\leftarrow\kappa+\ell$
31. END WHILE
32. $\sigma\leftarrow$sigEncode($\tilde{c},\boldsymbol{z}\bmod^{\pm}q,\boldsymbol{h}$)
33. RETUEN σ

算法3 ML-DSA.Verify(pk,M,σ)

输入:私钥$sk\in$B,消息$M\in\{0,1\}^*$,签名$\sigma\in$B

输出:Boolen.

1. $(\rho,\boldsymbol{t}_1)\leftarrow$pkDecode($pk$)
2. $(\tilde{c},\boldsymbol{z},\boldsymbol{h})\leftarrow$sigDecode($\sigma$)
3. IF $\boldsymbol{h}=\perp$ THEM RETURN FALSE
4. END IF
5. $\hat{\boldsymbol{A}}\leftarrow$ExpandA$(\rho)$
6. $tr\leftarrow$H(BytesToBits(pk),512)
7. $\mu\leftarrow$H(tr // M,512)
8. $(\tilde{c}_1,\tilde{c}_2)\in\{0,1\}^{256}\times\{0,1\}^{2\lambda-256}\leftarrow\tilde{c}$
9. $\boldsymbol{c}\leftarrow$SampleInBall($\tilde{c}_1$)
10. $\boldsymbol{W}'_{\mathbf{APPROX}}\leftarrow\mathrm{NTT}^{-1}(\hat{\boldsymbol{A}}\circ\mathrm{NTT}(\boldsymbol{z})-\mathrm{NTT}(\boldsymbol{c})\circ\mathrm{NTT}(\boldsymbol{t}_1\cdot 2^d))$
11. $\boldsymbol{W}_1'\leftarrow$UseHint($\boldsymbol{h},\boldsymbol{W}'_{\mathbf{APPROX}}$)
12. $\tilde{c}'\leftarrow$H(μ // w1Encode($\boldsymbol{W}_1'$),2λ)
13. RETURN$[\![\|\boldsymbol{z}\|_\infty<\gamma_1-\beta]\!]AND[\![\tilde{c}=\tilde{c}']\!]AND[\![$number of 1's in h is $\leqslant w]\!]$

表3展示了每个参数集相对应的密钥和签名的大小. 从此表中可以看到随着安全等级的不断上升,密钥和签名的规模也在逐渐增大. 在存储ML-DSA公钥和私钥时还可以采取不同的优化方式. 如果有较多的硬件存储资源,可以预先计算并存储多项式矩阵$\hat{\boldsymbol{A}}$,以加快签名和验证的速度. 如果希望减少私钥所需的空间,则只需要存储32字节的种子ξ,在签名生成过程中计算私钥的其他部分.

2.3 Crystals-Dilithium环多项式乘法算法

Crystals-Dilithium的多项式乘法在环多项式R_q上计算,值得注意的是:(1)多项式的最高项次数必须小于n;(2)多项式的系数必须在模q范围内. 若乘法的结

表3　ML-DSA数字签名密钥和签名大小　　单位:Byte

参数集	私钥	公钥	签名
ML-DSA-44	2 528	1 312	2 420
ML-DSA-65	4 000	1 952	3 293
ML-DSA-87	4 864	2 592	4 595

果就在多项式环上则不需要进行处理,否则需要对结果进行模操作,从而映射到多项式环中.

给定两个多项式 $A=\sum_{i=0}^{n-1}A_i x^i, B=\sum_{j=0}^{n-1}B_j x^j$,若用公式法计算多项式乘法,如式(1)所示:

$$\begin{aligned} C(x)=A(x)\times B(x)&=\left[\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}A_iB_jx^{i+j}\right]\bmod(x^n+1)\\ &=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(-1)^{\lfloor (i+j)/n\rfloor}A_iB_jx^{(i+j)\bmod n}\end{aligned}\tag{1}$$

该方式也被称为教科书(Schoolbook)算法,需要 $O(n^2)$ 次乘法. 在硬件实现时,如果采用单个乘法单元实现占用资源较少,但运算耗时较长[18];如果采用并行结构,虽然能够节省运算时间,但是会导致过多的资源消耗[19].

NTT算法需要 $O(n\log n)$ 次乘法,相比较传统的计算方式更加快速和高效,灵活性也更高[20]. NTT算法源于快速傅里叶变换(Fast Fourier Transform, FFT),NTT去除了FFT中复数部分复杂的浮点数和乘加计算,所有运算都在有限域上,且用整数原根代替了FFT中的复数形式的单位根(可统称为旋转因子),除了对模数有一定限制,NTT能够解决大多数格密码算法中的多项式乘法. NTT变换规则如式(2)所示:

$$\hat{A}_i=\sum_{j=0}^{n-1}A_jw^{ij}\bmod q,\ i=0,1,\cdots,n-1\tag{2}$$

将得到的NTT域上的数据再返回多项式环上称为NTT逆变换(Inverse Number Theoretic Transform, INTT),其变换规则如式(3)所示:

$$A_i=n^{-1}\sum_{j=0}^{n-1}\hat{A}_iw^{-ij}\bmod q,\ i=0,1,\cdots,n-1.\tag{3}$$

利用NTT计算有限域上的多项式乘法 $C(x)=A(x)\times B(x)$ 时,首先对 $A(x)$ 和 $B(x)$ 利用式(2)进行NTT变换获得NTT域上的表达式 $\hat{A}(x)$ 和 $\hat{B}(x)$,然后通过 $\hat{C}(x)=\hat{A}(x)\circ\hat{B}(x)$ 将乘法简化为逐点乘法,最后利用式(3)将结果 $\hat{C}(x)$ 恢复到多项式环运算内. 整个过程可以用以下公式概括:

$$A(x)\times B(x)=\mathrm{NTT}^{-1}(\mathrm{NTT}(A)\odot\mathrm{NTT}(B))\tag{4}$$

除了常用的NTT算法和Schoolbook算法,解决多项式乘法还可以使用Karatsuba算法[21],Karatsuba算法通过将较长的多项式拆分成多个较短的多项式,并重复利用较短多项式计算结果,减少了乘法次数,降低了计算复杂度,但在硬件实现时会带来较大的资源开销,影响系统的整体性能. NTT算法是目前所有环多项式乘法算法中计算复杂度最低的一类算法,并可将FFT的硬件结构拓展到NTT上,因此硬件结构灵活多变,可以根据不同的应用场景选择相应的NTT硬件结构,使得资源消耗和性能达到平衡.

3　Crystals-Dilithium底层关键组件的优化方案

针对Crystals-Dilithium数字签名不同的安全参数和不同的实现平台,例如现场可编程逻辑门阵列(Field Programmable Gate Array, FPGA)平台和专用集成电路(Application Specific Integrated Circuit, ASIC)平台,国内外已经有一些成果. 在Crystals-Dilithium算法中,哈希和伪随机数扩展、多项式乘法占据了整个算法超过90%的资源和时间消耗[22]. 为了提高硬件实现中的性能和效率,对这些底层关键组件进行优化是至关重要的. 本章节将会对这些关键组件的优化进行详细阐述和对比分析.

3.1　模乘的优化方案

多项式乘法占据了Crystals-Dilithium算法实现近50%的运行时间[23]. NTT和INTT作为多项式乘法的核心加速算法,其中模乘操作占据了主要的计算比重. 因此,对模乘操作的优化对于提升NTT实现的性能至关重要. 其中,优化模乘操作不仅涉及硬件实现结构的优化和资源的权衡,也包括算法层面的改进,通过改进多项式乘法模块中的模约简算法,主要的算法有Montgomery模乘算法[24]、Karatsuba分治算法[22]、Barrett模约简算法[25]和Plantard模约简算法[26],从而提升算法的执行效率和硬件实现的性能.

Land等人[27]采取了一种经典的模约简策略,利用 $2^{23}\equiv 2^{13}-1 \bmod q$ 的关系对模数进行约简,将一个23 bit的数 $r[22:0]$ 利用模数的特性约简为 $r[22:19]\cdot 2^9+r[18:0]$. 作为首个针对Xilinx Artix-7平台提出的设计,Land等人[27]通过使用大量数字信号处理器(Digital Signal Processors, DSPs)资源对模约简的结构进行优化设计,获得最快计算速度的优化方案. 具体来说,为了增加计算的吞吐量,作者巧妙地利用移位寄存器实例来延迟第三个DSP的输入,如图1所示. 这种设计在模乘法阶段仅使用了524个查找表(Look-Up-Tables, LUTs)以及759个触发器(Flip-Flops, FFs),但是在单个模乘法单元中仅使用了4个DSPs. 这样的做法虽然可以让整个模乘达到较高的工作频率,但是同时大量DSPs的使用,也会导致资源消耗的显著增加. Gupta等人[28]也参考这样的递归利用模数的性质进行约简的方式,不

同的是，为了降低资源的消耗，Gupta 等人[28]利用 LUTs 和 FFs 代替了其中 DSPs 的使用. 这种替代不仅减少了资源的占用，而且为低功耗和低成本的硬件实现提供了一种有效的思路.

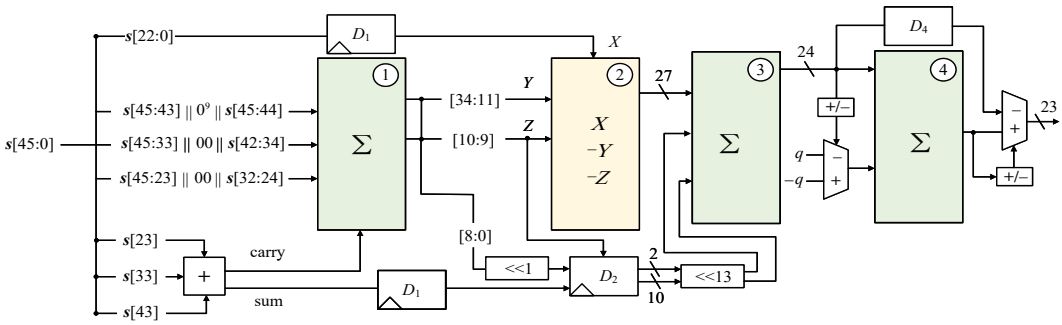

图 1 基于数个 DSP 的模约简单元[27]

Zhao 等人[22]则在该约简方法的基础上针对 Crystals-Dilithium 算法的三个模数进行了深入研究，并针对算法中涉及的模数提出了进一步的优化策略. 首先，他们将模数转换为一个标准的有符号数字(Canonical Signed Digit，CSD)表示法[29]，这一步骤利用了模数的固有属性来压缩需要约简的数的位宽. 通过重复压缩过程，直到结果小于模量的两倍，从而显著减少了数据的位宽需求. 在压缩步骤完成后，通过执行条件减法操作来获得最终的约简结果，如图 2 所示.

在模乘的算法实现中，除了从硬件结构和资源优化方面对模约简过程做优化之外，众多研究也从算法角度入手，致力于改进多项式乘法模块中的模约简算法，以减少所需的时钟周期并提升乘法计算的性能. Li 等人[25]将 Karatsuba 中的分治算法应用在 Crystals-Dilithium 的大数乘法中，实现了显著的性能提升. 由于 DSP 单元无法一次性处理 23 比特乘法，Li 等人[25]采用分治设计可以使所有 DSPs 单元的计算在一个时钟周期中完成，极大降低了关键路径中的乘法复杂度的同时，显著提高了模块并行度，如图 3 所示. 此外，该模乘法器架构中部署了三级流水线的设计，其中第一级完成 Karatsuba 算法的中间结果计算，第二级进行移位和加减运算并使用寄存器暂存中间结果，打断关键路径. 第三级完成最终的模约简. 采用该流水线的设计使得各级延迟均衡，吞吐率达到最大化. 整体电路在满足时序约束的同时实现了面积优化. 这种优化策略不仅减少了所需的时钟周期，而且通过并行处理提高了整体的计算吞吐量，但是这样的处理方式不可避免地造成了较大资源的消耗.

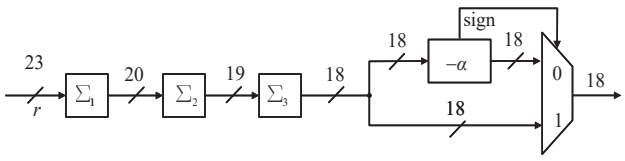

图 2 模数为 8380417 的模约简单元[22]

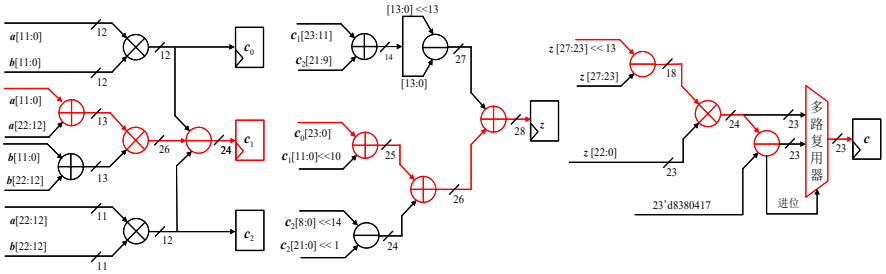

图 3 分治法模乘法器架构[25]

Hwang 等人[30]归纳了整数近似的概念并且利用 Barrett 算法的近似性质在 Cortex-M3 和 8 位 AVR 上实现 Crystals-Dilithium 的模乘. 在标准 Barrett 乘法中，为了计算 $ab \bmod q$，通常先计算一个近似商 $\tilde{q}=\mathrm{floor}\left(\frac{abR}{q}\right)$，其中 R 是 2 的幂，使得 $ab-q\tilde{q}$ 落在区间 $[-q,q)$ 内. 结合商的近似特性，Hwang 等人[30]提出可以通过放宽边界条件，即让 $ab-q\tilde{q}$ 落在一个更宽的区间 $[-B,B)$ 内，其中 $q \leqslant B \leqslant R$，降低商 $\tilde{q}$ 的计算成本，因此作者在计算 $\tilde{q}$ 时直接抛弃了部分低位，从而简化了高位乘法的模拟，用更少的乘法和移位操作来近似模拟高位乘法. 此外，作者证明了在 Cortex-M3 和 AVR 平台上，当乘法高指令和乘法长指令不可用或速度较慢时，只需 1 次高位乘法和 2 次低位乘法的 Barrett 乘法比需要 2 次高位乘法或长乘法

和1次低位乘法的Montgomery乘法更有优势.

在Cortex-M3上,基于Barrett乘法的NTT/INTT实现相比最新的基于Montgomery乘法的实现,常数时间版本快1.38~1.51倍,变量时间版本快1.10~1.21倍.在8位的AVR平台上,基于Barrett乘法的NTT/INTT实现相比基于Montgomery的NTT/INTT系统的C实现高出6.37~7.27倍.

Pham等人[31]同样基于Barret算法提出了一种新的模乘架构,如图4所示,该架构由2个DSP计算范元和一个模块化约简过程(MR)组成. 算法使用两个DSP对23位整数执行整数乘法,其中MR包括上半部分乘法器(UH-Mult)和下半部分乘法器(LH-Mult). UH-Mult单元执行将DSP计算单元输出的高24位V与预计算常数T的相乘,而LH-Mult单元执行UH-Mult单元输出右移24位后与模数相乘,两个乘法过程被优化为移位、加法和减法操作,优化后的Barrett模约简算法避免了使用乘法器进行大整数乘法,从而在硬件实现上显著降低了资源消耗和功耗. 为了进一步提升硬件效率,Pham等人[31]在架构中部署了四级流水线的设计. 第一和第二级位于DSP片内,在执行乘法和加法输出后,不消耗额外硬件资源,充分利用了DSP的内部结构,在不增加资源开销的情况下实现了流水线. 第三级放置在UH-Mult的输出处,通过将长关键路径分割成若干个较短的路径,从而降低关键路径延迟,提高电路的最高工作频率. 第四级位于模乘法器的输出处,进一步优化关键路径的时序表现,提高整个电路的工作频率. Pham等人[31]所设计的模乘法器架构能够以385 MHz的频率运行,整个模块仅消耗111个LUTs、71个FFs和2个DSPs,与Wang等人[32]的设计相比,硬件资源消耗减少到原来的1/149,显著降低了硬件资源的使用.

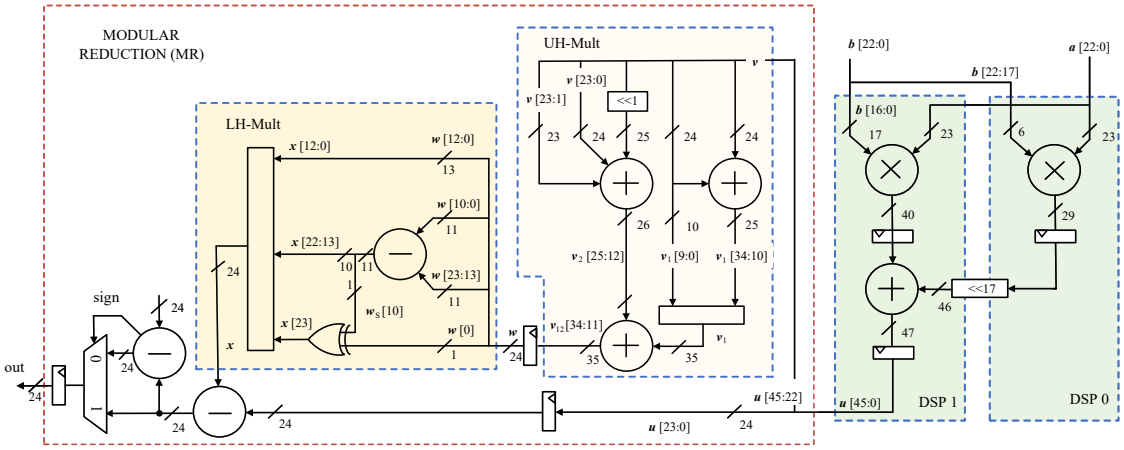

图4　基于Barret约简算法的模乘法器架构[31]

Malal等人[33]则选择了一种运行时可配置的Montgomery模约简算法,以增强设计的适应性和灵活性. 该算法能够让设计的蝶形单元适应不同大小和类型的多项式操作,为NTT和INTT计算以及运行时的逐点乘法提供了灵活性,在资源使用和计算速度之间实现灵活的权衡. 但是Montgomery模约简算法与其它算法相比,需要对被乘数进行预处理,以将它们转换到Montgomery域. 这一预处理步骤需要额外的计算资源,会增加算法的总体资源消耗.

Li等人[25]的设计基于Karatsuba算法简化了模乘法算法,设计的模乘法单元可以计算23比特和13比特乘法. 其设计基于Karatsuba算法可以减少关键路径的长度并提高系统的最大工作频率,进而提高了处理器的性能. 然而Karatsuba算法通过将大整数乘法分解成较小整数乘法的方式,虽然减少了单个乘法操作的复杂度,但是增加了乘法器的个数,并且引入了额外的加法操作,这会增加系统的复杂性和延迟. 此外,该设计存在DSP资源浪费的情况,性能还可以进一步提升. 在处理小规模数据时,Karatsuba算法的优势不明显,甚至导致性能下降. 因此,在实际应用中,需要综合考虑数据规模和系统需求,权衡Karatsuba算法的优点和缺点,以确定是否适合特定的硬件实现.

本节针对CRYSTALS-Dilithium算法中模乘操作,介绍了现有研究中的各种优化策略,这些策略涉及硬件结构、算法改进以及资源管理等多个方面. 其中Land等人[27]、Gupta等人[28]和Zhao等人[22]的设计均是采用经典模约简策略递归利用模数性质进行约简,Land等人[27]使用大量DSP资源,实现了快速计算,并且通过移位寄存器延迟DSP输入以增加吞吐量,但存在DSP资源浪费的问题.

Gupta等人[28]在此基础上使用LUTs和FFs代替DSPs,减少资源消耗. Zhao等人[22]进一步将模量转换为CSD表示法,压缩位宽,并通过多次的比较判别条件完成模约简,减少了数据位宽需求. Hwang等人[30]和

Pham 等人[31]的设计基于 Barrett 算法. Hwang 等人[30]通过结合整数近似的概念与 Barrett 算法,用于乘法计算之前对乘数进行约简来优化 NTT/INTT 的性能. Pham 等人[31]基于 Barret 算法提出了一种新的模乘架构可以在使用更少的硬件资源的同时实现相当的吞吐量. 该算法将模乘运算中的除法转化为乘法和移位操作,避免了直接执行昂贵的除法运算,从而降低了硬件复杂度和运算延迟,此外,Barrett 算法的各个步骤(如乘法、移位、减法)之间具有较好的并行性,可以通过流水线设计进一步提高运算效率和吞吐量. 但是 Barrett 算法在计算过程中需要预计算一个与模数相关的常数,这个预计算过程本身需要一定的计算量,当模数频繁变化时,预计算开销可能会抵消约简过程的性能优势,因此,对于某些特定的模数[如梅森(Mersenne)素数],基于 Barrett 算法的实现可能不如经典的 Montgomery 算法高效,例如 Malal 等人[33]为了实现更好的灵活性和适应性,采用运行可配置的 Montgomery 算法. 但是考虑到 Montgomery 算法需要对乘数进行预处理,这在模数频繁变化时可能会成为性能瓶颈. 因此,研究者需要根据具体应用场景权衡不同算法的性能和实现成本来选择最合适的模约简算法.

3.2 NTT 的优化方案

NTT 在 Crystals-Dilithium 算法中扮演着核心角色,其性能直接影响到整个密码系统的效率. 对于 NTT 计算,一般采用 CT(Cooley-Tukey)蝶形单元应用在 NTT 过程中,而 GS(Gentleman-Sande)蝶形单元应用在 INTT 的过程中. 针对 NTT 的优化主要集中在硬件实现的结构优化和前文提到的模乘算法的改进上. 硬件实现的典型结构是使用蝶形单元按照蝶形图进行逐层计算[34~36]. 目前,针对 NTT 的设计方案主要包括高速率设计[22,27,37]和轻量级设计[38].

针对 NTT 的高性能实现,基本采用并行设计结构,在同一层中同时计算多个蝶形单元[39~41]. 如图 5 所示,Ricci 等人[37]基于 FPGA 平台,提出了一种旨在充分利用硬件资源以实现高效 NTT 计算的设计方案. 他们设计了一个 2 × 2 NTT 蝶形计算单元的结构,并通过 4 个蝶形单元实现并行化和流水线化处理. 这样的设计合并了两个蝶形变换过程,有效减少了存储中间结果所需的块随机存取存储器(Block Random Access Memory, BRAM)数量,并减少了蝶形计算迭代间的间隔次数. 四个 BRAM 被用于存储输入值、中间结果以及输出值. 在未使用时,这些 BRAM 接口可以通过控制单元切换到 NTT 组件接口,以便进行数据的输入和输出. 在蝶形单元计算过程中,BRAM 接口被控制单元用于设置地址,同时连接到数据总线和 2 × 2 蝶形单元布局的计算结构中. 该设计方案同时考虑了并行计算和存储器访问的优化,使其能够在不同硬件资源的 FPGA 平台上灵活部署,具有较强的可扩展性. 然而,这种 2 × 2 蝶形单元设计也存在局限性. 它限制了蝶形单元的复用,并且由于内存访问顺序的复杂性,需要一个复杂的控制模块来管理,这在 LUTs 和 DSPs 资源的消耗上有显著增加.

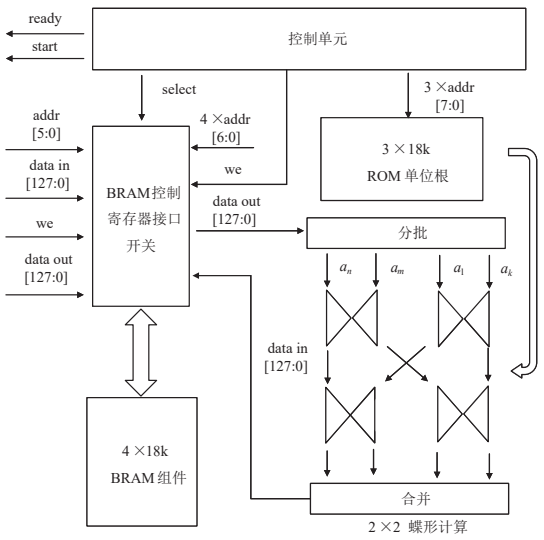

图 5 2 × 2 NTT 蝶形计算单元[37]

Land 等人[27]在 Ricci 等人[37]的 2 × 2 NTT 蝶形计算单元结构的基础上,选择使用 DSPs 来进行所有的算术操作,以实现低面积占用和高频率. 他们的设计特别利用了 FPGA 中 DSP 的不同操作模式,包括运行时重新配置、预添加以及单指令多数据(Single Instruction Multiple Data,SIMD)添加,这使得他们的 NTT 实现了 311 MHz 的最大频率,这一频率在当时基于 Xilinx Artix-7 平台的设计中是最高的,然而,这种高性能的实现也带来了非常高的 DSP 使用率. 此外,Land 等人[27]充分发挥了 BRAM 模块的双端口功能,使他们的设计能够在 NTT 的最底层同时读取两个旋转因子,并同时处理四个系数的计算. 这种设计提高了数据吞吐量,但同时也意味着每个蝶形单元在每个周期中需要读取和回写两个系数,导致所需的 BRAM 数量与蝶形单元的数量成正比. 具体来说,如果有 k 个蝶形单元,就需要 k 倍的 BRAM 资源. 虽然 Land 等人[27]的设计在频率上达到了显著优势,但也因为大量使用 DSP 资源和 BRAM 资源而造成了资源的利用率较低.

同样选择 2 × 2 NTT 蝶形计算单元进行 NTT 计算,Beckwith 等人[38]提出了一种由地址解码器和 4 位先入先出队列(First In First Out,FIFO)组成的无冲突内存调用方式,如图 6 所示. 其中,4xFIFO 组件用于转置 4 × 4 系数矩阵,以确保在 NTT 操作前和 INTT 操作后系数被正确排序. 地址解释器用于将 NTT 算法中的表示地址(RAddr)

转换为真实地址(TAddr),以适应NTT操作中的重新排序. 地址FIFO队列用于将真实地址延迟4个时钟周期,以匹配2×2蝶形单元流水线的深度. 通过将内存地址映射为一个6位二进制数,并将二进制数按照需要的计算顺序预先存储在内存块中,使得后续读取内存数据时不会产生逻辑冲突,提高并行度,同时使用FIFO减少存储中间数据所需寄存器的数量. 此外,为了解决DSP资源消耗大的问题,Beckwith等人[38]选择使用LUTs和FFs代替其中的常数乘法,与Ricci等人[37]的设计相比,在NTT和INTT阶段分别减少了40和80个DSPs的使用. 与Land等人[27]的设计相比,以19%的LUTs和100.5%的FFs的增加为代价,减少了64.6%的DSPs使用以及6.5%的BRAM的使用,并且获得了显著更好的性能. 然而,这种设计也存在挑战,即存在大量复杂的内存访问.

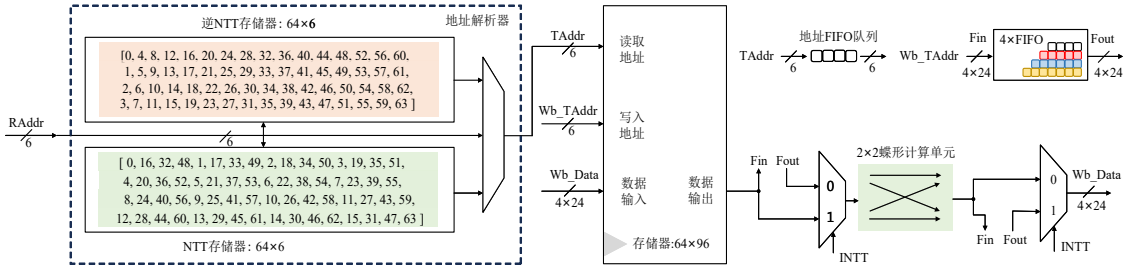

图6　无冲突内存调用2×2 NTT蝶形计算单元[38]

在此方案基础上,Zhao等人[22]提出了图7所示的分段流水式的NTT结构,该结构包含四个蝶形单元、一个用于存储旋转因子的BRAM,以及BRAM阵列的一些区域来代移位寄存器. 在2×2 NTT蝶形计算单元的实现中,每一层的输入系数相较于上一层的系数都需要进行一个移位操作. 为了优化这一过程,Zhao等人[22]提出用BRAM和计数器取代了大的移位寄存器,并通过使用BRAM阵列的空闲区域和端口来避免额外的BRAM消耗. 例如,在图7中,两个64 D和32 D的移位寄存器被一个BRAM替换,两个计数器的值交替用作BRAM的地址输入,一个计数器用于奇数周期,另一个用于偶数周期. 这样的设计涉及更少的内存访问,并且资源消耗更少,但是相对而言会增加整个NTT的关键路径.

本节针对CRYSTALS-Dilithium算法中的NTT操作,介绍了现有研究中的各种优化策略,这些策略涵盖了计算流程的优化、硬件结构的创新以及内存管理的改进,对应的硬件实现对比如表4所示. Ricci等人[37]提

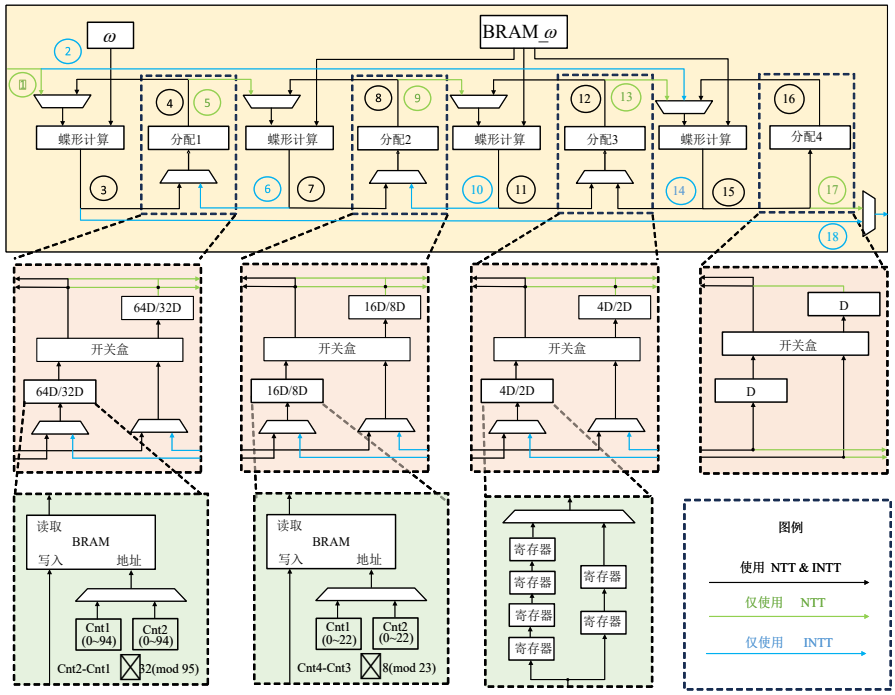

图7　分段流水式NTT蝶形计算单元[22]

出了一种2×2 NTT蝶形计算单元结构,通过并行化和流水线化处理减少了BRAM的使用,并降低了计算迭代间的间隔次数. 该设计优化了并行计算和存储器访问,提高了NTT计算的灵活性和可扩展性,但增加了LUTs和DSPs资源的消耗. Land等人[27]利用FPGA DSPs资源进行所有算术操作,实现了低面积占用和高频率,通过运行时DSPs的重新配置、预添加和SIMD添加等操作模式,达到了311 MHz的高频率,但DSPs使用率极高,在资源消耗上较大. Beckwith等人[38]提出了一种无冲突内存调用方式,通过地址解码器和FIFO组件优化了系数矩阵的转置和排序. 该设计使用LUTs和FFs代替常数乘法,减少了DSPs的使用,但增加了LUTs和FFs资源的消耗. Zhao等人[22]提出了分段流水式的NTT结构,用BRAM和计数器取代了大的移位寄存器,减少了内存访问和资源消耗. 该设计通过优化内存访问顺序和减少BRAM消耗,提高了计算效率,但增加了NTT的关键路径. Land等人[27]的设计虽然在频率上达到了优异的性能,但这种高性能的实现是以大量DSPs资源消耗为代价的. 相比之下,Beckwith等人[38]和Zhao等人[22]通过优化内存管理和减少DSPs使用,实现了资源的更高效利用. 可见,NTT的优化方案需要在性能提升、资源消耗和计算效率之间找到平衡,在追求高性能计算的同时如何平衡资源消耗将是未来研究开展的难点也是重点.

3.3 哈希和伪随机采样的优化方案

在CRYSTALS-Dilithium算法中,哈希函数和伪随机数生成器的作用是多方面的. 采样过程与伪随机数生成紧密相关,它依赖于伪随机数生成器来从特定的分布中选择元素,这些元素用于构建多项式、矩阵等. 优化这些操作可以显著提高算法的实现效率.

Crystals-Dilithium算法主要使用2015年被NIST标准化的第三代安全散列算法(Secure Hash Algorithm,SHA-3)(FIPS-202)[44]家族中的SHAKE-128和SHAKE-256可扩展输出函数(XOFs). 它们均是基于Keccak核设计的,统一使用1600-bit的状态(State)的Keccak_f[1 600]函数进行多轮f函数置换,每轮置换包括5种映射方式(θ、π、ρ、χ、ι),具体细节详见文献[42]. 作为SHA-3函数的核心,Keccak核排列占据了超过20%的资源消耗和处理周期[45],因此,Keccak核的优化设计是SHA-3函数优化的关键. 基于已有的研究成果,SHA-3函数的优化方案主要聚焦于Keccak核的轻量级或高速率设计. 针对SHA-3函数的低资源优化设计,使用单个Keccak核实现所有功能,牺牲性能换取低资源消耗[27,28]. 单个Keccak核的低资源优化方案主要采用折叠结构(Folded),其中折叠结构分为"通道式"(lane-wise)和"切片式"(slice-wise)两种折叠方式. Bertoni等人[46]首次提出基于通道式的折叠方式,每次轮函数迭代处理1个lane(25bit),这种设计方式进行每轮5种函数映射时需要调用大量的前轮函数的数据,导致高延迟且每轮需要215个时钟周期数. Jungk等人[47]提出了首个切片式的折叠方案,每次可以并行处理8个slice. 切片式折叠允许在一个时钟周期内完成对折叠的完整轮次函数处理,但需要重新调度轮次函数. 这种折叠方法是一种高度灵活的解决方案,关于并行处理的slice数量可以通过对控制逻辑进行小的修改而轻松调整. 随后,Sundal等人[48]和Jungk等人[47]将Keccak_f[1 600]函数按照同样的slice-wise方式,一次并行处理8个或者16个slice,即每次迭代处理200 bit或者400 bit进行多轮置换,如图8和图9所示,相比Jungk等人[47]的工作降低了延迟,达到了一定的硬件资源与计算效率的均衡.

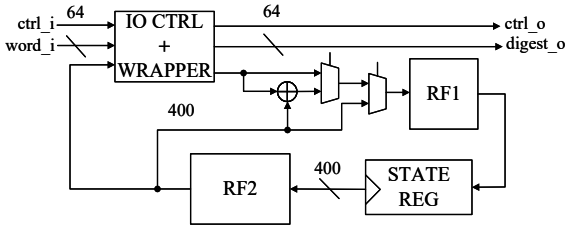

图8 每次处理400 bit的Keccak核折叠结构[48]

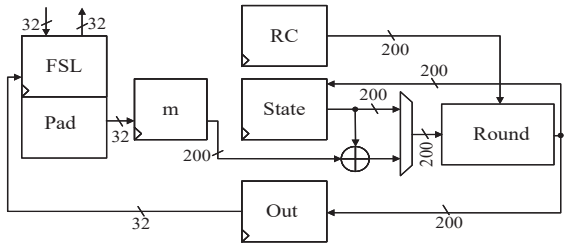

图9 每次处理200 bit的Keccak核折叠结构[47]

表4 NTT的硬件实现对比

设计	最大工作频率/MHz	LUTs	FFs	DSPs	BRAMs	测试平台
文献[42](仅NTT乘法模运算单元)	152	7 073	2 031	0	NR	Artix-7
文献[43](仅NTT乘法模运算单元)	216	2 044	NR	16	NR	Artix-7
文献[33](资源节约型)	142.8	1 601	699	10	5	UltraScale+
文献[33](高吞吐率)	142.8	11 558	4 912	80	40	UltraScale+

针对SHA-3函数的高性能优化设计,则是通过级联多个Keccak核实现高速的计算,但是缺点是消耗大量的面积资源[22,25,37,38]. 例如,Beckwith等人[38]在其设计方案中使用了三个Keccak核,其中两个主要用于$\boldsymbol{A}$矩阵的采样,而第三个用于其余的哈希和采样. 为了提高采样性能,Beckwith等人[38]还实现了多个系数的并行采样,以充分利用Keccak核,但是这也会导致较大的资源消耗. 除此之外,也有少数研究在基于格的密码方案中对SHA-3函数进行了高速高效的硬件实现和优化:Wong等人[49]首次提出具有2个IPR和2个OPR两次展开架构,并且将64比特轮常数(Simple Round Constants,SRC)简化为8比特的方法. 在此基础上,刘冬生等人[50]提出了一种可以适用于所有SHA-3函数的高效高速的硬件结构. 通过进一步压缩轮常数为7比特,并调整流水线寄存器的位置,进一步缩短关键路径,显著提升整个流水线架构的性能,提高了33.4%的吞吐量和28.2%的硬件效率,相关设计如图10所示.

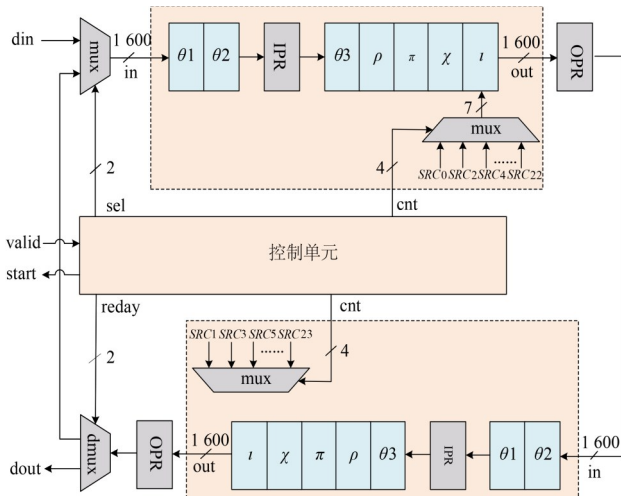

图10　SHA-3中Keccak核的高性能硬件结构[50]

但是这些采样算法都不支持在32位字节上工作. Land等人[27]设计了一个缓冲区来解决32位字与采样算法位宽不匹配的问题,实现了费舍尔-耶茨洗牌算法以优化挑战c的采样过程,并通过使用具有可变深度的移位寄存器来存储和处理非零系数的偏移量和符号位. 此外,作者还通过优化拒绝采样过程中的随机偏移量查找和替换操作,以及管理移位寄存器的深度来存储新的偏移量,提高了采样效率. 最终,处理后的多项式直接写入BRAM,简化了整个数据处理流程,从而提升了算法的硬件实现效率和资源利用率. 虽然作者这样的处理可以解决哈希算法输出与采样算法位宽不匹配的问题,但是会造成大量的寄存器的消耗,并且由于输出之后还需要格式转换,计算周期较长.

在此基础上,Zhao等人[22]提出了如图11所示的一种包含拒绝采样和基于双端口BRAM的SampleInBall采样模块,该模块使用BRAM阵列的空闲区域记录系数,代替了寄存器记录偏移量,从而避免了使用大量的寄存器. 其次,该模块的输出以标准多项式格式,可以直接进行NTT处理,无需额外的格式转换. 此外,直接的串行处理需要将每个有效样本进行一次读取操作和两次写入操作,对应于三个时钟周期,而提出的模块平均每个样本只需要一个周期. 其中黑色、蓝色和红色组件表示基于不同情况下的处理,实现存储和处理时间减少的同时,提高了系统的运行效率. 整体的设计达到低资源消耗的同时,减少了运算的周期.

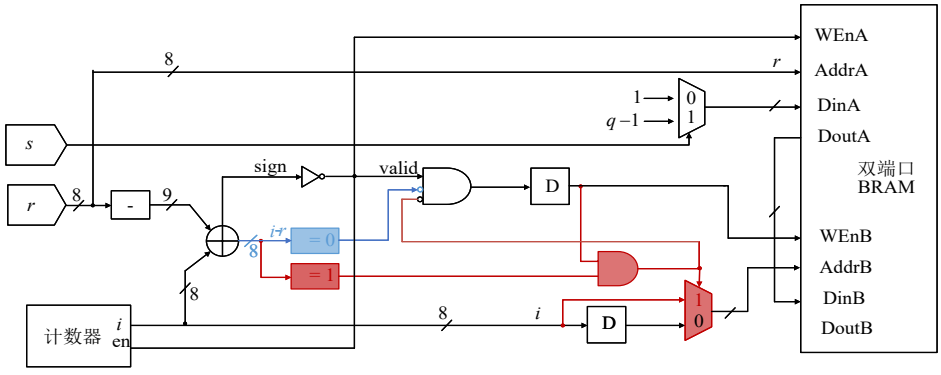

图11　基于BRAM的采样单元[22]

本节针对CRYSTALS-Dilithium算法中的哈希和采样操作的优化策略展开介绍,但是总的来说,目前对于哈希函数和采样还有巨大的优化空间. 尽管SHA-3函数的FPGA优化实现研究已经有了很多进展,但是只有少数基于CRYSTALS-Dilithium算法的文章使用相关的优化策略. 目前根据SHA-3函数结构的优化方式可以分为:折叠结构、流水线结构和展开结构. 折叠结构虽然占用的资源小,但是运行周期长并且需要较为复杂的控制逻辑;展开结构是在一个时钟周期内实现Keccak核的多轮迭代,缺点是相应的资源消耗也会成倍增

加,硬件效率低下;流水线结构则是可以在迭代轮内部增加流水线寄存器,也可以将其插入展开后的迭代轮之间,以提高运行效率;将三种技术根据应用场景结合,以实现低功耗、高速率或性能面积折中的高性能设计,是未来SHA-3函数优化的关键研究方向.这种优化不仅适用于CRYSTALS-Dilithium算法,也对许多后量子密码学(PQC)算法的硬件优化至关重要.

4 Crystals-Dilithium顶层系统架构的优化方案

在设计CRYSTALS-Dilithium硬件实现方案时,我们必须面对信息安全领域的多样化应用需求、开发通用性以及不同安全级别的多元化要求.这些需求对硬件实现方案的应用场景和性能有着显著影响.因此,在设计阶段,我们必须提前规划好不同资源的消耗,并综合考虑速度、资源消耗、安全性和功耗之间的折中关系.根据不同的设计要求,调整性能和资源之间的平衡至关重要.本节将总结CRYSTALS-Dilithium顶层系统架构的优化方案,这些方案可以归纳为三种设计的分类:高速率实现、轻量级实现和后量子密码算法的集成实现.这些分类方式有助于我们理解和区分不同实现的特点.尽管类别之间的界限并非绝对,设计过程中往往需要在性能与资源消耗之间进行权衡,但这些标签仍为我们提供了一个实用的框架,以识别和比较各种实现策略.通过这一框架,希望帮助研究者们能够更有效地评估不同方案的优缺点,从而选择最适合特定需求的实现方式.

4.1 高速率的硬件实现架构

在高速率应用场景中,设备面临着处理大量请求的挑战,同时还需保持能效和安全性.因此,在设计CRYSTALS-Dilithium的硬件实现时,我们优先考虑实现最大的处理速度,这通常通过优化系统架构的并行度[51]和改进环多项式乘法模块的运算性能[52]来满足设备快速处理的请求.

Ricci等人[37]首次提出了一种应用于FPGA平台的完全基于超大规模集成电路的硬件描述语言(Very High speed integrated circuit hardware Description Language,VHDL)的Crystals-Dilithium硬件实现.该设计选择UltraScale+作为FPGA平台,给出了每个安全级别的实验结果,并且通过FPGA验证了该方案的性能.如图12所示,Ricci等人[37]的签名算法在FPGA平台上的并行化硬件实现方式,充分利用了FPGA平台的并行计算能力和硬件资源,以实现高效的数字签名算法执行.具体来说,签名算法组件通过数据总线宽度为512位的输入和输出AXI流接口与环境进行通信.整体的签名算法被分为初始化部分和while循环两部分.由于初始化的各个部分都是相互独立的,可以直接进行并行化处理;而在while循环部分中,除了生成向量$\boldsymbol{y}$之外,其他所有操作都依赖于之前操作的结果.此外,为了实现并行化处理,作者将while循环的计算过程划分为11个单独模块,即使各个模块没有执行迭代,也还是会创建输出,从而顺利实现并行处理.总的来说,Ricci等人[37]将整体签名算法被分成18个并行运行的过程,与顺序方法相比,整体的算法需要更多的DSPs块,但可以实现低周期数和高频率.签名算法在UltraScale+平台上需要消耗68 461个LUTs和86 295个FFs;验证算法需要使用61 738个LUTs和34 963个FFs.该方案显著提高了Crystals-Dilithium整体系统在FPGA平台上的计算性能,能够实现每秒签署15 832条消息,验证10 524个签名,整体速度较Ducas等人[53]基于HLS实现的硬件设计提高约114倍.作为第一个在FPGA上完成Crystals-Dilithium整体系统的实现,该实现使用了许多资源直接将算法映射到硬件上,虽然实现了高性能,但是代价是花费大量的DSPs和BRAM,资源的利用效率较低.

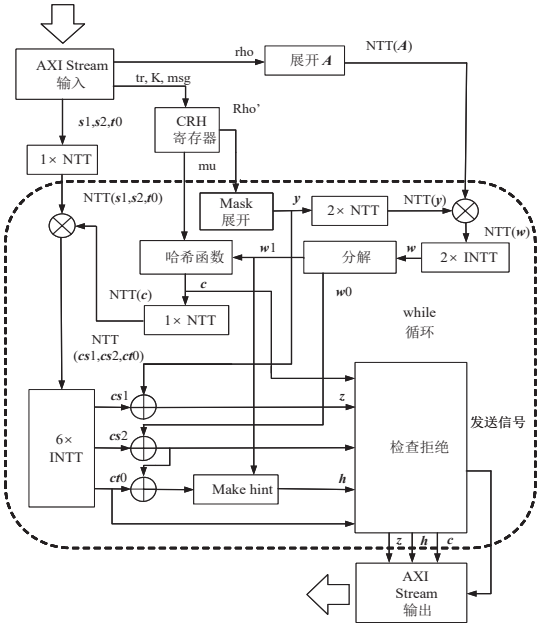

图12 基于VHDL的首个Crystals-Dilithium签名算法的硬件实现方案[37]

在此基础上,Beckwith等人[38]通过调整向量和矩阵的维度,设计了一种能够适应不同安全级别的统一架构,如图13所示.这种设计允许在三个算法操作中共享和复用大量资源,如Keccak核、多项式算术模块和BRAM.与分别实现三个算法流程相比,合并后的架构在LUTs使用上减少了近52%,显著降低了资源消耗.签名生成作为最复杂的操作之一,作者通过将其划分为两个模块组,并进一步细分为多个并行部分,最小化

了阶段间的依赖和数据传输,从而优化了性能和资源利用. 具体来说,签名过程被划分为预计算、阶段0和阶段1三个阶段. 预计算仅在签名生成开始时执行一次,涉及私钥的采样和解码,由第一组模块执行. 同时,另一组模块执行阶段0,包括$\boldsymbol{y}$的采样和矩阵乘法,以生成签名尝试所需的$\boldsymbol{w}$向量. 两个阶段完成后,最初用于预计算的模块组利用阶段0的结果尝试创建有效签名,即阶段1. 在执行阶段1的同时,另一组模块再次运行,以便在阶段1失败时立即进行新的尝试,形成了一个高效的两阶段流水线. 尽管这种设计提高了处理效率,但它仍然以牺牲资源消耗为代价.

参考以上的设计思路,Zhao等人[22]提出了一种分段流水线处理方法,在更短的操作周期和更少的存储需求下实现了高性能. 如图14所示,Zhao等人[22]将算法中的操作分为多个部分,不同的段被串行处理,段内的操作以流水线方式处理. 这种流水线方法减少了对中间结果的存储需求,并降低了内存访问次数;而分段处理则减少了算法所需的硬件资源,并允许更高效地利用BRAM模块为了进一步减少BRAM的存储消耗. 为了进一步降低BRAM的存储消耗,作者调整了预先计算和存储的矩阵$\boldsymbol{A}$,改为即时计算模式. 此外,考虑到NTT计算与脉冲宽度调制(Pulse Width Modulation, PWM)点乘计算的紧密相关性,他们设计了HEAD模块和TAIL模块,分别放置在流水线中的NTT模块之前

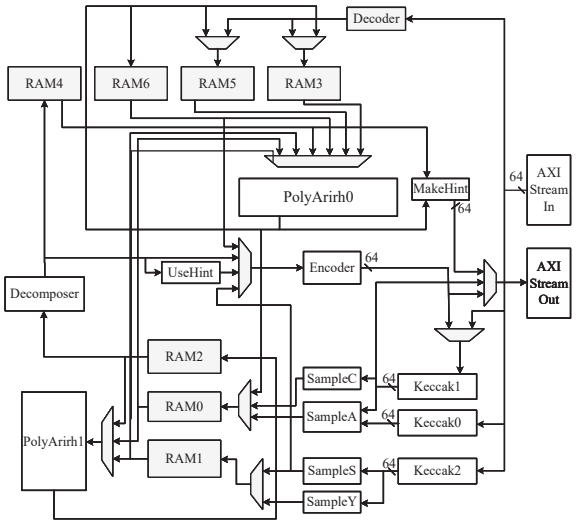

图13　CRYSTALS-Dilithium组合架构的块图[38]

和之后,以实现高速流水线的多项式乘法. HEAD模块集成了一个模乘法器和一个模加法器,而TAIL模块则包含比较器、模加法器、Decompose模块等. 这种设计使得整个多项式乘法过程无须存储中间结果,从而实现了较高的处理速率. 然而,这种设计的缺点在于无法复用NTT资源,这在一定程度上导致了显著的面积开销. 这主要是因为多路复用的增加和额外模块实例的需求.

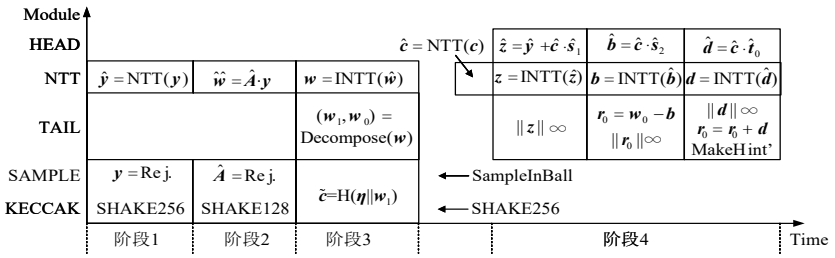

图14　签名算法循环部分的分段流水线处理[22]

本节介绍了CRYSTALS-Dilithium算法在追求高性能硬件实现时的不同策略和优化方案,这些策略涵盖了性能与资源消耗、并行化与流水线处理以及BRAM资源复用与优化. Ricci等人[37]的实现在性能上取得了显著提升,但资源消耗较高. Beckwith等人[38]和Zhao等人[22]通过优化设计减少了资源消耗,但这种优化在一定程度上牺牲了性能或增加了硬件复杂度. 其次,本节的研究都强调了并行化和流水线处理在提高性能中的重要性. Ricci等人[37]通过并行化处理提高了吞吐量,而Beckwith等人[38]和Zhao等人[22]通过并行化结合流水线处理减少了操作周期和存储需求. 此外,Zhao等人[22]的设计通过减少BRAM存储消耗和即时计算矩阵$\boldsymbol{A}$,提高了处理速率,但增加了硬件的复杂度和面积开销. 这种权衡在高性能硬件实现中也是常见的,尤其是在追求高吞吐量和快速响应的应用场景中.

4.2　低资源消耗的硬件实现架构

在面向资源受限的应用场景时,由于场景设备本身硬件资源十分有限,因此在设计的时候需要着重考虑低硬件资源消耗及低功耗进行轻量级的优化实现,如何在LUTs、DSPs、FFs三种不同的硬件资源中进行取舍从而获得最好的性能表现,并对核心模块进行复用,使得设计和实现的效率更高[54],一直是一个研究难点.

Land等人[27]的设计侧重于在尽可能少的LUTs和FFs资源利用下实现可能的最佳性能. 作为第一个被提出用于低端Xilinx Artix-7平台的设计,Land等人[27]在他们的体系结构中通过有效使用DSPs以减少资源消

耗,复用各种模块,大量使用DSPs代替LUTs和FFs的面积,在一定程度上减少了LUTs和FFs资源的使用,但该体系结构的并行度较低,导致其模块的利用率较低,并且在设计过程中忽略了DSPs对于资源的占用.其中该设计在采样模块使用单个共享实例来实现所有功能,也是为了节省资源而牺牲一些性能.

Gupta等人[28]的设计则选择使用LUTs和FFs代替DSPs从而减少资源消耗的优化方案,提出将协议算法中的Power2Round和Decompose两个分解运算模块如图15所示进行联合封装,共享两个模块的控制单元,并且在Decompose模块的第二个乘法中预先计算了16个可能出现的值,使用一个4位LUTs作为输入优化计算资源,从而节省对于DSPs的使用.其次,Gupta等人[28]还提出了基于Barrett算法的模约简算法以提高效率和减少资源使用,利用模约简操作有效地降低DSPs的数量.此外,在采样模块,Gupta等人[28]选择节省资源而牺牲一些性能,使用单个共享实例来实现所有功能.该设计在UltraScale+平台上达到了1.176 GHz的运行频率,并且在整体设计仅需要使用13 900个LUTs、6 800个FFs、4个DSPs以及35个BRAMs.

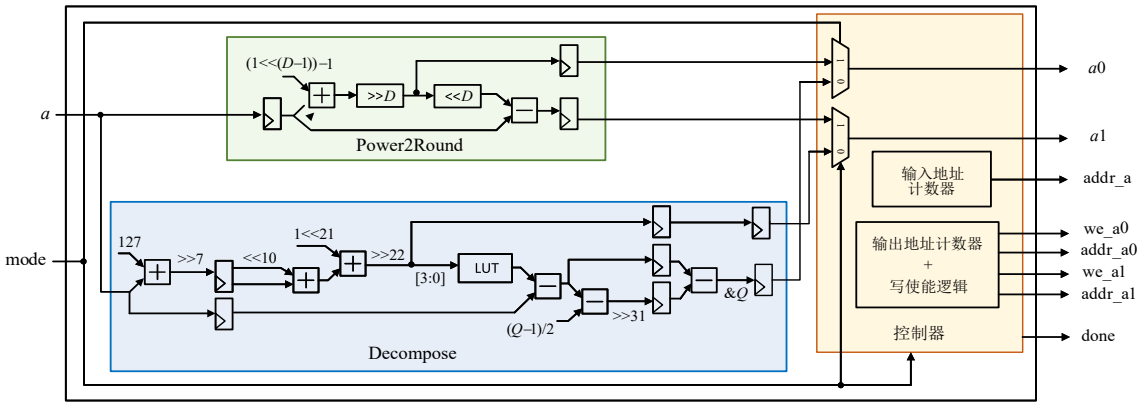

图15 Power2Round和Decompose联合封装模块[28]

本节介绍了CRYSTALS-Dilithium算法在追求低资源消耗硬件实现时的不同策略和优化方案.一个优化方案是用大量的DSPs单元进行大数乘法[28],可以显著提高系统的频率,降低LUTs和FFs的资源的消耗;另一个优化方法是通过更多地使用LUTs和FFs来替代DSPs的功能[28],以降低对昂贵或能耗较高的DSPs资源的依赖.选择哪种优化策略取决于具体的应用需求、成本预算、功耗限制以及性能要求.

4.3 后量子密码算法的联合实现

公钥加密、密钥协商(密钥封装)和数字签名技术作为公钥密码体制的三大重要组成部分,经常作为一个整体出现.由于分组密码加密速度比公钥加密速度快,通常在密码硬件设备中均会集成分组密码算法、哈希算法、密钥封装算法和数字签名算法,这些算法足以满足多种应用场景下的安全防护要求.在硬件实现过程中,一些密码算法会共用某些电路组件,如密码封装算法Crystals-Kyber和数字签名算法Crystals-Dilithium共用NTT模块等.所以,研究者们通过在密码算法实现中有效地复用的通用核心算子电路,提升整体资源利用率.具体的联合实现优化方法包含:Crystals-Kyber和Crystals-Dilithium的联合高性能硬件实现以及Saber和Crystals-Dilithium的联合高性能硬件实现.

Crystals-Dilithium和Crystals-Kyber均利用NTT用于高效地执行多项式乘法,二者的联合实现可以满足高安全性并适配多种场景.运用Crystals-Kyber对发送的信息进行公钥加密,同时运用Crystals-Dilithium对发送和接收端的身份信息进行数字签名加密.现阶段研究的难点主要有两点:(1)Crystals-Kyber中使用的是13比特多项式模乘,而Crystals-Dilithium使用的是23比特多项式模乘.二者不同位数的乘法使得在联合实现中,多项式模乘法单元的设计非常困难.同时,二者不同的模数也会带来不同的模约简要求.(2)Crystals-Kyber和Crystals-Dilithium整体实现流程不同,要求在设计的时候选择合适的内存调用方式、逻辑控制单元以及尽可能寻求更多的模块复用,才能达到性能和资源的平衡.

Aikata等人[55]首次针对Crystals-Dilithium和Crystals-Kyber提出了一种统一的体系结构KaLi,该体系设计了一个统一而灵活的多项式算术单元,可以为Crystals-Dilithium和Crystals-Kyber的所有安全级别执行密钥生成、封装、解封装、签名生成和签名验证,并且其处理Crystals-Kyber运算的速度是Crystals-Dilithium运算的两倍.Aikata等人[55]提出了一种用于Kali的算法,使为Crystals-Dilithium设计的整数乘法单元对Crystals-Kyber具有灵活性,它执行两次23比特乘12比特的整数乘法,如图16所示.这个算法在Crystals-

Dilithium的情况下给出了一个乘系数,在Crystals-Kyber的情况下给出了两个乘系数,如图17所示.

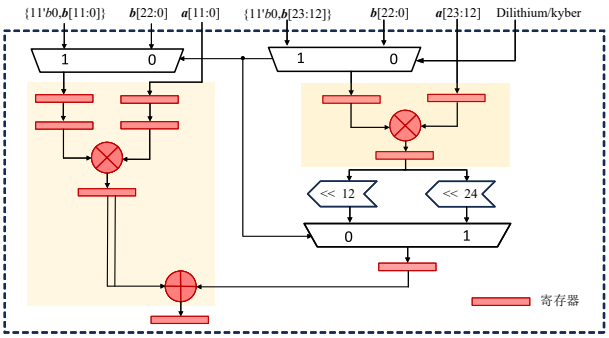

图16　适用于多模数的整数乘法器[55]

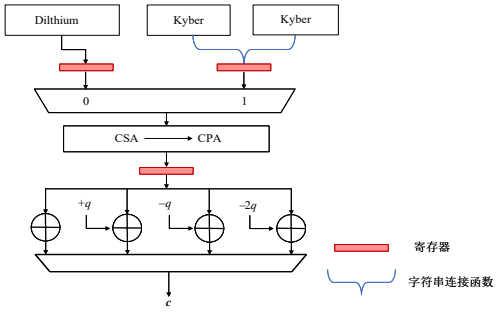

图17　Crystals-Kyber和Crystals-Dilithium统一的模约简单元[55]

KaLi设计在使用多个时钟域的ASIC平台上特别适用. 在ASIC的28 nm/65 nm技术上,该设计实现了2 GHz/560 MHz的时钟频率,可以用于存储单元的快速时钟. 在Zynq Ultrascale+平台上,其所提出的架构可以在270 MHz时钟频率下仅仅使用23 277个LUTs、9 758个FFs、4个DSPs和24个BRAMs. 经验证,KaLi方法所达到的性能比这两种方案中的任何一种的独立实现都要好.

Li等人[42]提出一种2KNTT多项式乘法结构,如图18所示,实现模乘法单元可以同时满足不同模数的操作,并且控制单元可以满足不同项数的要求,同时适用于Crystals-Dilithium和Crystals-Kyber算法. 在原始的NTT算法中,模数q需要满足$q=1\bmod 2n$的条件,而在2KNTT算法中,这种限制被削弱为只需要满足$q=1\bmod n$或$q=1\bmod n/2$,甚至更小的条件. 这种改进的算法使得在加密方案中选择更小的模数成为可能,从而减少了带宽需求和资源消耗. 在Artix-7平台以到152 MHz的频率运行,消耗7 073个LUTs、2 310个FFs和0个DSPs.

Li等人[42]改进的2KNTT算法的多项式乘法单元结构如图18所示,由于所选算法本身的时间复杂度较低,计算和存储体系结构采用流水线型的设计,实现了运算时间和面积的平衡,而整体的并行结构和统一的蝶形计算单元则保证了性能的提高. 与高度并行设计相比,它平衡了高并行条件下计算单元统一和存储单元调度的复杂问题,而更高的性能可以降低单位性能所需的资源量.

虽然Saber在第三轮中因为特定情况下的安全性难以被证明而被淘汰出局,但是Saber相对于其他格密码方案的实现较为容易,主要的原因是M-LWR所决定的采样方式和独特的基于2的幂次的模数,使得在设计硬件实现方案时结构较为简单,减少了随机性以及所需带宽. 因此Saber在联合实现方面仍然具有较高的研究价值. Basso等人[43]和Aikata等人[56]提出了可以同时实现于Crystals-Dilithium数字签名以及Saber密钥封装机制的联合设计.

除了Crystals-Kyber和Saber与Crystals-Dilithium的联合实现外,椭圆曲线数字签名算法(Elliptic Curve Digital Signature Algorithm,ECDSA)作为一种基于椭圆曲线密码学的数字签名算法,也有与Crystals-Dilithium混合的数字签名方案,Ghinea等人[57]提出了一种ECDSA和Crystals-Dilithium混合的数字签名方案软件实现,以在经典加密算法和后量子安全方案之间取得平衡. 这种方法为当前的安全密钥提供了一种过渡方案,既保持了经典加密算法的安全性,又具有抵抗量子攻击的能力,使其能够在量子计算时代仍然保持安全. Ghinea等人[57]采用了一种称为“Strong Nesting”的方法,首先使用X-EUF-CMA安全方案对给定的消息进行签名,得到$\sigma 1$,然后使用X-SUF-CMA安全方案对$(m,\sigma 1)$进行签名,得到$\sigma 2$,最终的混合签名为$\sigma=(\sigma 1,\sigma 2)$. 该方法保持了Crystals-Dilithium的强不可伪造性,同时也保持了ECDSA的存在性不可伪造性,即使一个方案被攻破,也可以保持另一个底层方案的安全性,在安全密钥身份验证的情境下具有强不可伪造性,因此对经典和量子攻击都具有抵抗力. 可惜该方案目前还停留在软件实现层面,在未来可以在硬件实现层面做深入研究.

5　Crystals-Dilithium硬件实现对比分析

本节将结合第3节和第4节的关键组件优化与整体架构优化方案,全面对比基于这些优化方案的Crystals-Dilithium硬件实现的性能. 通过文字描述和表格的形式,我们将展示不同优化策略对性能的影响,以便更好地理解各方案在实际应用中的效果.

表5汇总了已有的Crystals-Dilithium协议的硬件实现,从表中可以看出,大部分设计使用的DSP数量是非常少的,这是迭代实现的一个优点. 迭代实现的另外一个优点是能够节省很多LUTs和FFs的资源消耗. 由于迭代型模多项式乘法模块只有一个蝶形计算单元,所以需要大量的BRAM资源,一方面暂存中间计算数据,另一方面需要存储逻辑地址和计算地址来控制NTT计

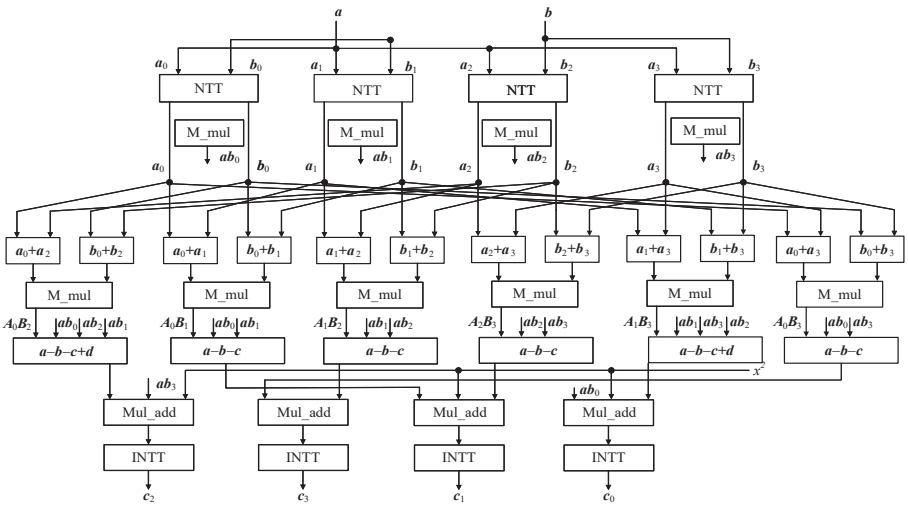

图18 2 KNTT多项式乘法结构[42]

算的顺序. Beckwith 等人[38]、Land 等人[27]、Aikata 等人[55,56]的设计方案BRAM数量都超过了20,其原因就是在于二者都使用了大量的BRAM进行NTT计算顺序的逻辑地址的控制. 其流水线型模多项式乘法模块虽然会消耗大量的DSP资源以及LUTs、FFs资源,但是在外围计算电路能够匹配乘法单元频率的前提下,流水线型设计仍能获得最好的运算性能表现,尤其是频率表现. 同时,部分设计方案采用流水线的设计方法,数据不需要在计算过程中间进行任何暂存和滞留,也不需要用逻辑地址控制NTT计算顺序,所以BRAM的使用量几乎为0.

在高速率的硬件实现方案中,Ricci 等人[37]设计了四个2×2的NTT蝶形计算单元实现并行化和流水线化的处理,使用消耗大量资源直接将算法映射到硬件上,虽然提高了硬件的性能,但是硬件资源的使用效率较低并且硬件消耗资源大. 在此基础上,Beckwith 等人[38]提出一个能够在所有安全级别执行三种操作的组合模块,也包括每个模块支持一种操作的单个模块. 其优点在于通过组合模块的调用,三种操作可以在同一硬件电路中实现,节约了近52%的硬件资源开销,并且通过

表5 硬件实现方案性能参数比较(安全级别2/3/5)

设计	CCs/k	Time/μs	Freq/MHz	LUT	FF	BRAM	DSP	ATP
文献[25](高速率)	4.4/30.1/5.0 7.3/49.5/8.0 10.9/55.2/12.7	33.9/231/38.5 56.1/381/61.6 83.8/425/97.6	130 (Artix-7)	32 320	9 734	14	24	4.01 6.59 8.02
文献[27](高速率)	NR	18.7/9.6/66.9 33.1/18.1/10.5 51.0/33.8/112.1	163 (Artix-7)	51 028	27 042	53	45	NR
文献[37](高速率)	12.6/18.4/10.5 18.2/21.0/15.3 22.9/22.3/20.2	27.8/17.7/15 19.2/15.5/10.5 15.2/14.2/7.8	350 (UltraScale+)	184 372	146 494	178	316	NR
文献[38](高速率)	4.8/6.5/10.9 8.2/9.7/16.7 14.1/14.6/24.4	42/57/94 71/84/139 121/126/210	116 (Artix-7)	53 187	28 318	29	16	NR
文献[22](低资源)	4.2/28.1/4.4 5.9/44.7/6.1 8.8/49.0/9.0	43.3/290/45.4 60.9/461/63.0 90.1/506/92.9	96.9 (Artix-7)	29 998	10 366	11	10	4.04 6.23 7.34
文献[28](低资源)	14.6/31.7/15.4 23.6/48.7/26.1 39.7/70.2/46.7	54.1/117/57.0 87.4/180/96.7 147/260/173	270 (UltraScale+)	23 347	9 798	24	4	1.83 2.92 4.65

FIFO的使用对系统内存模块做了优化,实现更低的延迟时间和更小的面积资源. 但因为该设计要兼容FALCON算法的电路需求,所以Crystals-Dilithium的本身的性能受到了限制. 参考这样的设计思路,Zhao等人[22]也提出了一个支持三个操作(密钥生成、签名和验签)以及所有安全级别的紧凑高性能硬件架构. 其优点在于:

(1)提出分段流水式的处理,减少时钟周期,实现数据独立执行的并行操作,这样的优化方式减少了大量操作的执行时间和中间结果的存储需求,解决了硬件资源利用率低的问题.

(2)设计了包括高速流水线NTT模块,基于BRAM的SampleInBall采样模块,紧凑的分解模块和三个优化的模约简模块,仅需要使用11个BRAMs单元,减少了硬件资源开销. 但是多路复用的增加、额外的模块实例等,会导致一定的面积资源消耗,同时受限于分段流水的结构以及架构中对DSP单元的利用率不高的原因,其最大工作频率为96.6 MHz,频率性能还没有达到最佳.

在低资源消耗的硬件实现方案中,Land等人[27]的整体实现方案中,该方案使用了45个DSPs使得整体频率能够达到205 MHz,但由于其结构单一,缺少对DSP单元的优化设计,造成了DSP资源的浪费,理论上该结构可以通过优化DSP单元的结构来获得更好的性能表现. Gupta等人[28]利用资源和控制逻辑共享、预计算的模块共享以及模块共享等多种优化策略,降低LUTs和FFs使用量的同时节省了对DSP单元的消耗. 与Beckwith等人[38]的设计相比,LUTs的资源消耗减少了73.7%,FFs的资源消耗减少了75.8%,DSPs的资源消耗减少了75%.

在联合实现的方案中,Li等人[42]和Aikata等人[55]提出了Crystals-Dilithium和Crystals-Kyber的联合实现. Li等人[42]提出了2KNTT算法的多项式乘法架构,该算法时间复杂度较低,计算和存储体系结构的流线型控制复杂度决定了时间和面积的平衡,整体并行结构和统一的蝶形计算单元大大提高电路性能,在Artix-7平台上,NTT模块的最大工作频率为152 MHz,电路消耗的资源为7 073个LUTs、2 310个FFs. Aikata等人[55]则提出了全新的Kali设计,设计了一种统一而灵活的多项式运算单元,联合实现了Crystals-Kyber和Crystals-Dilithium的所有计算过程,电路在DSPs的资源消耗上大大降低的同时,在性能表现上也较为优秀,电路消耗的资源为23 000个LUTs、9 700个FFs、4个DSPs和24个BRAMs.

Basso等人[43]和Mert等人[56]提出了Crystals-Dilithium和Saber的联合实现. 由于Saber和Crystals-Dilithium共享相同的模数,使得模乘法单元的设计变得简单. 但受限于Saber的性能,使得二者的联合设计只能用于特定的应用场景. Basso等人[43]的设计在Artix-7平台上实现了使用了333 MHz的运行频率,消耗2 012个LUTs、331个FFs以及6个DSPs. 而Aikata等人[56]的设计在Ultrascale+平台上实现了200 MHz的运行频率,消耗18 406个LUTs、9 323个FFs、4个DSPs以及24个BRAMs.

6　结束语

本文围绕PQC数字签名算法Crystals-Dilithium的国内外研究成果进行综述. 针对Crystals-Dilithium数字签名可以根据不同的设计要求调整速率和资源之间的平衡的特点,首先,本文就模乘、NTT、哈希和采样等底层关键技术的优化方案进行了分析,介绍了底层关键组件的优化方向和优化策略;随后,针对目前已有的Crystals-Dilithium算法的整体实现,将其分为高速率、低资源消耗和算法联合实现这三个场景,对不同场景下的硬件实现的优化方法和硬件实现的结果进行了归纳总结和对比分析,展现了不同设计思路和硬件架构的优劣.

当前PQC算法标准化正如火如荼地展开,未来PQC迁移将是全球安全战略中的重要一环. PQC算法的硬件实现相比软件实现具有更高的性能、更低的延迟、更高的并行性、更低的功耗、更高的安全性等优势,能够在各行各业中提供更高效、更安全的解决方案. 所以,在量子计算迅速发展的时代,PQC算法的硬件实现与优化设计研究愈发重要. 强烈建议未来对于后量子密码算法的硬件实现研究可以集中在以下四个方面:

(1)高速率硬件实现方案. 通过改进多项式模乘算法和模约简算法减少乘法计算的时钟周期,并且通过改进硬件实现方案,改善并行度和内存调用逻辑等方式,提高整体硬件实现速率.

(2)轻量级硬件实现方案,通过复用NTT、keccak核等多个核心组件,采用流水线的方式,减少存储需求和处理时间,降低硬件实现的资源消耗.

(3)多种后量子密码方案联合硬件实现,当然可以不局限于PQC算法之间,也可以将PQC算法与经典密码算法进行联合实现.

(4)高性能硬件实现方案,在速率和资源消耗之间寻找一种平衡,通过观察某些算法之间的组件关联度,对于通用的模块进行复用,展开电路设计与优化,减少不必要的硬件开销,为多应用场景提供全面的安全保障. 最后,希望本综述可以为研究者们在优化PQC算法的硬件实现方面提供思路和参考价值.

参考文献

[1] RIVEST R L, SHAMIR A, ADLEMAN L. A method for ob-

taining digital signatures and public-key cryptosystems[J]. Communications of the ACM, 1978, 21(2): 120-126.

[2] DAN B. The decision Diffie-Hellman problem[M]//Algorithmic Number Theory. Berlin, Heidelberg: Springer, 1998: 48-63.

[3] ELGAMAL T. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE Transactions on Information Theory, 1985, 31(4): 469-472.

[4] KOBLITZ N. Elliptic curve cryptosystems[J]. Mathematics of Computation, 1987, 48(177): 203-209.

[5] GALBRAITH S D. Mathematics of Public Key Cryptography[M]. Cambridge: Cambridge University Press, 2012.

[6] SHOR P W. Algorithms for quantum computation: Discrete logarithms and factoring[C]//Proceedings 35th Annual Symposium on Foundations of Computer Science. Piscataway: IEEE, 1994: 124-134.

[7] CHEN L, JORDAN S, et al. Report on Post-Quantum Cryptography[M]. Gaithersburg: US Department of Commerce, National Institute of Standards and Technology, 2016.

[8] MICCIANCIO D, REGEV O. Lattice-Based Cryptography[M]//Post-Quantum Cryptography. Berlin, Heidelberg: Springer, 2009: 147-191.

[9] BUCHMANN J, DAHMEN E, SZYDLO M. Hash-based digital signature schemes[M]//Post-Quantum Cryptography. Berlin, Heidelberg: Springer, 2009: 35-93.

[10] OVERBECK R, SENDRIER N. Code-based cryptography[M]//Post-Quantum Cryptography. Berlin, Heidelberg: Springer, 2009: 95-145.

[11] PETZOLDT A, BULYGIN S, BUCHMANN J. A multivariate based threshold ring signature scheme[J]. Applicable Algebra in Engineering, Communication and Computing, 2013, 24(3): 255-275.

[12] FEO L D. Mathematics of isogeny based cryptography[EB/OL]. (2017-11-11)[2025-03-19]. https://arxiv.org/abs/1711.04062.

[13] BEULLENS W. Breaking rainbow takes a weekend on a laptop[M]//Advances in Cryptology-CRYPTO 2022. Cham: Springer Nature Switzerland, 2022: 464-479.

[14] MAILLARD B. Implementing the Dual Approaches for Solving LWE[D]. Lund: Lund University, 2023.

[15] MAINO L, MARTINDALE C, PANNY L, et al. A Direct key recovery attack on SIDH[M]//Advances in Cryptology-EUROCRYPT 2023. Cham: Springer Nature Switzerland, 2023: 448-471.

[16] DUCAS L, KILTZ E, LEPOINT T, et al. CRYSTALS-dilithium: A lattice-based digital signature scheme[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018, 2018: 238-268.

[17] LANGLOIS A, STEHLÉ D. Worst-case to average-case reductions for module lattices[J]. Designs, Codes and Cryptography, 2015, 75(3): 565-599.

[18] 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.

[19] SINHA ROY S, BASSO A. High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020: 443-466.

[20] BERNSTEIN D J. Fast multiplication and its applications[J]. Algorithmic Number Theory, 2008, 44(1): 325-384.

[21] WEIMERSKIRCH A, PAAR C. Generalizations of the Karatsuba algorithm for efficient implementations [EB/OL]. (2006-07-03)[2025-03-19]. https://eprint.iacr.org/2006/224.

[22] ZHAO C K, ZHANG N, WANG H N, et al. A compact and high-performance hardware architecture for CRYSTALS-dilithium[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021: 270-295.

[23] 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.
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. (in Chinese)

[24] MCIVOR C, MCLOONE M, MCCANNY J V. Modified Montgomery modular multiplication and RSA exponentiation techniques[J]. IEE Proceedings - Computers and Digital Techniques, 2004, 151(6): 402-408.

[25] LI X, LU J H, LIU D S, et al. A high speed post-quantum crypto-processor for crystals-dilithium[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2024, 71(1): 435-439.

[26] PLANTARD T. Efficient word size modular arithmetic[J]. IEEE Transactions on Emerging Topics in Computing, 2021, 9(3): 1506-1518.

[27] LAND G, SASDRICH P, GÜNEYSU T. A hard crystal-implementing dilithium on reconfigurable hardware[M]//Smart

Card Research and Advanced Applications. Cham: Springer International Publishing, 2022: 210-230.

[28] GUPTA N, JATI A, CHATTOPADHYAY A, et al. Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-dilithium[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2023, 70(8): 3234-3243.

[29] HARTLEY R I. Subexpression sharing in filters using canonic signed digit multipliers[J]. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 1996, 43(10): 677-688.

[30] HWANG V, KIM Y, SEO S C. Barrett multiplication for dilithium on embedded devices[EB/OL]. (2023-12-25) [2025-03-19]. https://eprint.iacr.org/2023/1955.

[31] PHAM T X, DUONG-NGOC P, LEE H. An efficient unified polynomial arithmetic unit for CRYSTALS-dilithium[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2023, 70(12): 4854-4864.

[32] WANG T F, ZHANG C, CAO P, et al. Efficient implementation of dilithium signature scheme on FPGA SoC platform[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022, 30(9): 1158-1171.

[33] MALAL A. Designing efficient and flexible NTT accelerators [EB/OL]. (2024-09-15) [2025-03-19]. https://eprint. iacr. org/2023/1617.

[34] FRITZMANN T, SEPULVEDA J. Efficient and flexible low-power NTT for lattice-based cryptography[C]//2019 IEEE International Symposium on Hardware Oriented Security and Trust. Piscataway: IEEE, 2019: 141-150.

[35] JATI A, GUPTA N, CHATTOPADHYAY A, et al. SPQ-Cop: Side-channel protected post-quantum cryptoprocessor[C]//IACR Cryptology ePrint Archive. Trier: DBLP, 2019: 765.

[36] FRITZMANN T, SHARIF U, MÜLLER-GRITSCHNEDER D, et al. Towards reliable and secure post-quantum coprocessors based on RISC-V[C]//2019 Design, Automation & Test in Europe Conference & Exhibition. Piscataway: IEEE, 2019: 1148-1153.

[37] RICCI S, MALINA L, JEDLICKA P, et al. Implementing CRYSTALS-dilithium signature scheme on FPGAs[C]// Proceedings of the 16th International Conference on Availability, Reliability and Security. New York: ACM, 2021: 1-11.

[38] BECKWITH L, NGUYEN D T, GAJ K. High-performance hardware implementation of lattice-based digital signatures[EB/OL]. (2022-02-25) [2025-03-19]. https://eprint.iacr.org/2022/217.

[39] MERT A C, OZTURK E, SAVAS E. Design and implementation of a fast and scalable NTT-based polynomial multiplier architecture[C]//2019 22nd Euromicro Conference on Digital System Design. Piscataway: IEEE, 2019: 253-260.

[40] ZHANG N, YANG B H, CHEN C, et al. Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(2): 49-72.

[41] FRITZMANN T, SIGL G, SEPÚLVEDA J. RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(4): 239-280.

[42] LI H Q, CHEN T, WU A Q, et al. High efficient architecture of polynomial multiplier with variable parameter based on 2KNTT[C]//2022 IEEE Asia Pacific Conference on Circuits and Systems. Piscataway: IEEE, 2022: 606-610.

[43] BASSO A, AYDIN F, DINU D, et al. Where star wars meets star trek: Saber and dilithium on the same polynomial multiplier[EB/OL]. (2022-03-08) [2025-03-19]. https://eprint.iacr.org/2021/1697.

[44] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. SHA-3 Standard: Permutation-based hash and extendable-output functions[S/OL]. Gaithersburg: NIST, 2015(2015-08-04) [2025-03-19]. https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions?pub_id=919061.

[45] CARRIL X, KARDARIS C, RIBES-GONZÁLEZ J, et al. Hardware acceleration for high-volume operations of CRYSTALS-kyber and CRYSTALS-dilithium[J]. ACM Transactions on Reconfigurable Technology and Systems, 2024, 17(3): 1-26.

[46] BERTONI G, DAEMEN J, PEETERS M, et al. Keccak[C]// Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer, 2013: 313-314.

[47] JUNGK B, APFELBECK J. Area-efficient FPGA implementations of the SHA-3 finalists[C]//2011 International Conference on Reconfigurable Computing and FPGAs. Piscataway: IEEE, 2011: 235-241.

[48] SUNDAL M, CHAVES R. Efficient FPGA implementation of the SHA-3 hash function[C]//2017 IEEE Computer Society Annual Symposium on VLSI. Piscataway:

IEEE, 2017: 86-91.
[49] WONG M M, HAJ-YAHYA J, SAU S, et al. A new high throughput and area efficient SHA-3 implementation[C]// 2018 IEEE International Symposium on Circuits and Systems. Piscataway: IEEE, 2018: 1-5.
[50] 刘冬生, 陈勇, 熊思琦, 等. 应用于后量子密码的高速高效SHA-3硬件单元设计[J]. 信息安全学报, 2021, 6(6): 32-39.
LIU D S, CHEN Y, XIONG S Q, et al. Design of high-speed and high-efficiency SHA-3 hardware unit for post-quantum cryptography[J]. Journal of Cyber Security, 2021, 6(6): 32-39. (in Chinese)
[51] CONTI F, SCHILLING R, SCHIAVONE P D, et al. An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2017, 64(9): 2481-2494.
[52] BANERJEE U, PATHAK A, CHANDRAKASAN A P. 2.3 an energy-efficient configurable lattice cryptography processor for the quantum-secure Internet of Things[C]// 2019 IEEE International Solid-State Circuits Conference. Piscataway: IEEE, 2019: 46-48.
[53] DUCAS L, LEPOINT T, LYUBASHEVSKY V, et al. Crystals-dilithium: Digital signatures from module lattices[EB/OL]. (2018-12-01) [2023-12-10]. https://eprint. iacr. org/2017/633.
[54] GRECONICI D O C, KANNWISCHER M J, SPRENKELS D. Compact dilithium implementations on cortex-M3 and cortex-M4[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(1): 1-24.
[55] AIKATA A, MERT A C, IMRAN M, et al. KaLi: A crystal for post-quantum security using kyber and dilithium[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2023, 70(2): 747-758.
[56] AIKATA A, MERT A C, JACQUEMIN D, et al. A unified cryptoprocessor for lattice-based signature and key-exchange[J]. IEEE Transactions on Computers, 2022, 72(6): 1568-1580.
[57] GHINEA D, KACZMARCZYCK F, PULLMAN J, et al. Hybrid post-auantum signatures in hardware security keys[M]//Applied Cryptography and Network Security Workshops. Cham: Springer Nature Switzerland, 2023: 480-499.

作者简介

崔益军 男,1988年8月出生于江苏省海安市. 现为南京航空航天大学集成电路学院副院长、副研究员、博士生导师. 研究方向为集成电路设计,包括基于物理不可克隆函数的安全认证技术、面向未来信息系统的后量子密码安全通信技术、硬件木马、新兴计算等. 中国电子学会会员编号:E190036763M.
E-mail: yijun.cui@nuaa.edu.cn

李梦雪 女,2001年11月出生于浙江省温州市. 现为南京航空航天大学集成电路学院硕士研究生. 研究方向为后量子密码算法中核心算子的硬件设计与优化.
E-mail: limengxue@nuaa.edu.cn

王 辈 女,1991年2月出生于安徽省宿州市. 现为南京航空航天大学集成电路学院讲师. 研究方向为量子计算对公钥密码算法的安全性分析和后量子密码算法的软硬件加速研究. 发表SCI和EI论文10余篇.
E-mail: wangbei91@nuaa.edu.cn

王成华 男,1963年10月出生于江苏省扬中市. 现为南京航空航天大学集成电路学院教授、博士生导师. 研究方向为集成电路设计.
E-mail: chwang@nuaa.edu.cn

刘伟强 男,1983年3月出生于山东省东营市. 现为南京航空航天大学集成电路学院执行院长、教授、国家自然科学基金杰出青年基金项目入选者. 研究方向为高能效高安全新兴计算芯片. 中国电子学会会员编号:E1900111365.
E-mail: chwang@nuaa.edu.cn