NTT架构研究及其FPGA硬件优化实现

赵旭阳[1)] 梁志闯[1)] 胡 跃[1)] 耿合详[1)] 赵运磊[1),2)]

[1)](复旦大学计算机科学技术学院 上海 200433)

[2)](密码科学技术国家重点实验室 北京 100000)

摘 要 量子计算机的发展对现有公钥体系的影响是实质性的.在众多后量子密码流派中,格基密码方案因其安全性、高效性等优良特点而成为了主流技术路线.密码算法在各实现平台的运行效率是后量子算法评估进程中的重要指标.FPGA(Field Programmable Gate Array)具有并行性架构,是密码体系实际部署时的重要硬件平台.近年来,后量子算法的硬件优化实现研究吸引了越来越多的关注.针对格基密码算法中计算复杂度最高、耗时最长的操作——环上多项式乘法计算,本文对主流的加速技术——数论变换技术(Number-Theoretic Transform,NTT)进行了系统研究,根据参数不同将其分为标准NTT(Standard-NTT,SNTT)、删减NTT(Truncated-NTT,TNTT)、混合NTT(Hybrid-NTT,HNTT).为了提高适用性,设计了一种统一型、可常数时间执行、支持多参数的硬件电路,支持三种NTT架构,结合Karatsuba技巧,将对应系数相乘中乘法次数减少20%.针对NTT的核心操作——蝴蝶变换,设计了一种紧凑型、低时延的电路结构,可实现正向NTT、对应系数相乘、逆向NTT功能,同时支持Cooley-Tukey和Gentleman-Sande两种结构;提出一种新的系数访存模式,采用"交叉存储型"结构,使用双口BRAM将计算后的系数"对角交叉"变换位置后存储在原地址,利用双Bank存储模式满足蝴蝶变换单元的数据吞吐量;对Barrett约减算法进行修改,提出一种适用于多模值的模约减硬件单元,用加法和移位代替其中的乘法操作,减少乘法器资源的消耗.本方案使用Verilog HDL语言在Xilinx公司Artix-7系列XC7A200TFBG484-2型号FPGA芯片上进行了纯硬件优化实现.与当前研究相比,本设计在面积上减少12%-62.6%,时间上提升5.5%-49.8%.此外,我们还对高吞吐量的并行设计进行了优化实现,优化后的二并行、四并行架构时间上分别加速两倍、四倍,面积消耗仅为单结构的1.45倍、2.58倍.

关键词 后量子密码;格基密码算法;多项式乘法;数论变换;FPGA;硬件实现

中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2023.02670

NTT Architecture Research and Its FPGA Hardware Optimization Implementation

ZHAO Xu-Yang[1)] LIANG Zhi-Chuang[1)] HU Yue[1)] GENG He-Xiang[1)] ZHAO Yun-Lei[1),2)]

[1)](*School of Computer Science, Fudan University, Shanghai* 200433)

[2)](*State Key Laboratory of Cryptology, Beijing* 100000)

Abstract The development of quantum computers has a substantial impact on the existing public key systems. The security of traditional public key encryption schemes, such as RSA and ECC algorithm, is based on the most difficult mathematical problems such as large integer factoring and discrete logarithm problems. In 1994, Shor proposed that these mathematical problems can be solved in polynomial time on a quantum computer. IBM and Microsoft engineers expect large-scale quantum computers to emerge in the next 15-20 years. It is urgent to research and design

收稿日期:2022-12-01;在线发布日期:2023-07-14. 本课题得到国家重点研发计划(2022YFB2701600,2017YFB0802000)、国家自然科学基金(61972094))资助. **赵旭阳**,博士研究生,主要研究领域为后量子密码、密码工程、软硬件协同设计.E-mail:xuyangzhao21@m.fudan.edu.cn. **梁志闯**,博士研究生,主要研究领域为后量子密码. **胡 跃**,博士研究生,主要研究领域为后量子密码和密码工程. **耿合详**,硕士研究生,主要研究领域为后量子密码和密码工程. **赵运磊**(通信作者),博士,教授,博士生导师,主要研究领域为后量子密码、密码协议与计算理论.E-mail:ylzhao@fudan.edu.cn.

quantum-resistant public key cryptographic algorithms and protocols. Post-quantum cryptography (PQC) refers to cryptography that can resist quantum computer attacks. Among many post-quantum cryptographic schemes, lattice-based cryptography has attracted extensive attention due to its security, efficiency and other advantages. In July 2022, the National Institute of Standards and Technology (NIST) announced four algorithms that will serve as the future post-quantum cryptography standard. Three of them are lattice-based cryptographic schemes. Among the winning algorithms in the Chinese national cryptographic algorithm competition, lattice-based algorithms account for the majority, such as Aigis, AKCN-MLWE, etc. FPGA (Field Programmable Gate Array) is an important hardware platform for practical deployment, with a parallel architecture. There has been widespread interest in researching and designing dedicated hardware unit for post-quantum cryptographic algorithms on FPGAs. The most complex and time-consuming operation in lattice-based cryptography is polynomial multiplication over the rings. Normally, most schemes use number theoretic transform (NTT) technology to reduce computational complexity and accelerate polynomial multiplication. In this paper, we studied NTT technologies used in the mainstream lattice-based cryptographic schemes and typed them as Standard-NTT (SNTT), Truncated-NTT (TNTT) and Hybrid-NTT (HNTT) according to different parameters. In order to improve its applicability, a unified, constant-time executable, multi-parameter supported hardware architecture was designed, which supports all three NTT architectures. Combining the Karatsuba technique, the number of multiplication times during point-wise multiplication was reduced by 20%. A compact and low latency unit has been designed for the core operation of NTT, butterfly transform, which can perform forward NTT, point-wise multiplication, and reverse NTT functions. It supports both Cooley Tukey and Gentleman Sande structures. To avoid data conflicts, we designed a unique cross-storage memory access mode, using two banks to store polynomial coefficients. After each layer of NTT, polynomial coefficients are cross-swapped before being stored in BRAM. The coefficients in the NTT calculation process are carried out in the specific domain, where the intermediate results require modular reduction. There are two popular algorithms in the cryptographic engineering field, Montgomery reduction and Barrett reduction. Both of them require two multiplications which are expensive for hardware design. For that, we have designed a dedicated reduction unit that replaces multiplications with addition/shift operations and is suitable for multiple moduli. The proposed hardware design code is written in Verilog HDL, synthesized, placed, and routed using Xilinx Vivado tools on the Xilinx Artix-7 FPGA XC7A200TFBG484-2 chip. Compared to the state-of-art, this design reduces area consumption by 12%-62.6%, and speeds up by 5.5%-49.8%. We also designed a high-performance parallel framework to increase data throughput. Compared to the single structure, the optimized two-parallel and four-parallel architectures speed up twice and four times respectively, with only 1.45 and 2.58 times area consumption.

Keywords post-quantum cryptography; lattice-based algorithm; polynomial multiplication; number theoretic transform; FPGA; hardware design

1 引 言

传统的公钥加密方案，如RSA、ECC算法等，其安全性基于大整数分解、离散对数求解等数学困难问题. 1994年Shor[1]提出在量子计算机上，上述数学问题可以在多项式时间内求解. 近些年，量子计算机工程领域不断取得新的进展. 谷歌于2019年建成53量子比特计算机Sycamore，200秒可完成当前最强大超级计算机约一万年的计算量. IBM于

2022年刚刚完成433量子比特计算机Osprey，速度为Sycamore的八倍. 为了抵抗量子计算机的攻击，NIST于2016年举办后量子公钥密码(Post-Quantum Cryptography，PQC)标准国际征集[2]. 2020年7月，NIST公布了第三轮评估结果[3]，包括7个候选算法和8个备选算法. 这两组中有7个基于格的密码，3个基于编码的密码，3个基于多变量的密码，2个基于哈希的密码以及1个同源密码. 2022年7月5日，NIST公布即将标准化的算法[4]并决定开启第四轮[5]进行后续研究. 其中，密钥封装方案的标准化算法为CRYSTALS-Kyber(下文简称Kyber)，数字签名方案的标准化算法为CRYSTALS-Dilithium(下文简称Dilithium)、FALCON、SPHINCS+，并特别推荐实施Dilithium算法. 这四个算法中，只有SPHINCS+是基于哈希的，其他三个算法都是格基密码方案. 我国于2019年举办了后量子密码竞赛，获奖算法中格基密码方案占大多数，密钥封装算法包括Aigis，LAC，AKCN-MLWE，Scloud等[6-7]. Shen等人[8]引入AKCN共识机制[9-10]，对Kyber、Aigis两个算法的各组参数做了优化设计，新方案有着更低的错误率，更快的计算速度，分别命名为OSKR、OKAI. 格基密码方案因其安全性、高效性等优良特点而受到了广泛的关注，在后续标准化进程中占据着突出地位.

格基密码方案通常基于(环、模)带误差学习(Learning With Errors，LWE)/带舍入学习(Learning With Rounding，LWR)问题，其中计算复杂度最高、耗时最长的操作为多项式环$\mathbb{R}_q=\mathbb{Z}_q/(\phi(x))$中的元素乘法，其中$\phi(x)$是次数为$n$的分圆多项式. 一般来说，取$\phi(x)=x^n+1$，其中，$n$是2的幂次. 教科书乘法算法计算该环上的多项式乘法的复杂度为$O(n^2)$，当n较大时，效率较低[11]. 因此，通常采用数论变换(Number Theoretic Transform，NTT)加速多项式相乘，从而将计算复杂度降为$O(n\log n)$. NIST公布的3个格基标准算法全部采用了NTT技术. 在格方案中，Dilithium[12]，Aigis[13]，OKAI-512/768/1024-1[8]，Kyber-v1[14]采用标准NTT，下文简称SNTT，其需要多项式环$\mathbb{R}_q$的维度n和模值q满足$n=2^m$，q为素数，且$q\equiv 1(mod\ 2n)$. 方案Kyber-v2/v3[15-16]，OSKR-512/768/1024-1，OKAI-1024-2[8]中n和q满足$q\equiv 1(mod\ n)$. 为了处理这种情况，文献[17]提出一种不彻底的NTT，从“底部删减”若干层. 我们将这种NTT称为删减NTT，下文简称TNTT. 方案OSKR-1024-2中n和q满足$q\equiv 1\left(mod\ \frac{n}{2}\right)$. 为此，Liang[11]等结合“底部删减”和“顶部分解”的思想，提出混合NTT，下文简称HNTT，先从顶部将多项式分为若干个多项式，然后分别进行TNTT计算.

目前针对NTT方法的优化实现主要有软件实现、软硬结合、纯硬件实现三种. 软件实现主要针对PC通用平台或ARM嵌入式平台，使用语言有C、AVX2等. 软硬结合实现形式有CPU与FPGA联合设计、HLS设计方式. Mert[18]使用PCIe接口将FPGA和CPU平台进行物理连接，针对高维度多项式NTT加速计算进行了实现，这种设计方式会带来额外的通讯开销. HLS高层次综合设计借助编译器可以将C/C++语言进行RTL级实现. 文献[19-22]采用HLS方式对NTT进行了硬件实现并与纯硬件设计进行了对比.

纯硬件优化加速实现基于FPGA平台，使用Verilog HDL、VHDL硬件描述语言进行硬件电路设计. 文献[23-25]采用Verilog语言对Dilithium算法参数进行了SNTT架构纯硬件实现. 其中，Bechwith[23-24]采用2×2架构，复用4个蝴蝶计算单元同时对NTT相邻两层做加速. Zhao[25]调整多路延迟换向结构，使用4个蝴蝶单元，通过复用器调整数据流方向对NTT过程进行四并行加速. Mert[26]采用Verilog对高纬度多项式不同模值的NTT结构进行了纯硬件实现. 文献[27-29]对Kyber-v3参数进行了TNTT硬件实现. Yaman[27]设计了简洁性蝴蝶单元，并提供了四并行、十六并行版本. Xing[28]采用原位存储结构，复用两个蝴蝶单元进行加速. Niasar[29]采用VHDL硬件设计语言，使用2×2架构进行了四并行加速.

目前硬件设计存在以下几种问题：

NTT结构固定. 绝大多数硬件加速方案只适一种算法，或特定的NTT结构，其(n,q)参数组固定，硬件加速模块对多算法平台兼容性较差.

蝴蝶单元模块计算功能单一. 正向NTT、逆向NTT的核心操作为蝴蝶计算，通常有Cooley-Tukey(CT)和Gentleman-Sande(GS)两种结构. 多数硬件方案只支持其中一种结构，使得数据在做NTT计算时需要预处理、后处理操作，将系数顺序重新排列. 这种操作会带来额外的时钟周期开销，降低计算速度.

约减模块对硬件设计不友好. 多项式环$\mathbb{R}_q$中元素乘法，其中间计算结果需约减到$\mathbb{Z}_q$内，常用的约减算法Montgomery约减[30]和Barrett约减[31]. Montgomery约减需将数值扩展到Montgomery域内计算，导致中间值数据位宽增大，消耗额外的硬件资源. 此外，两种算法都需要乘法计算，但乘法器在FPGA内部较为稀缺，不利于硬件电路优化.

数据存储开销较大. 蝴蝶操作一次计算需要读取两个系数，存储两个系数. 为了避免数据冲突，送入模块内的数据需要两两配对. 目前的访存模式会导致片内RAM资源的浪费，降低内存利用率，如“乒乓”读取模式.

目前，NIST初步将Kyber、Dilithium确立为下一代密码标准算法. 因受到专利等方面的制约，NIST宣称存在更改为其他算法的可能[32]. 我国后量子密码标准进程正在持续推进，最终的标准算法尚未确认. 密码行业正处于后量子迁移的过渡期. 在此阶段，为了在最终标准形成前抢占市场先机，以及在标准确立后兼容多种算法，进而达到更优的性能优势，硬件设计上往往需要兼容多种候选算法. 由于各种算法采用的(n,q)参数不同，分别设计不同的硬件加速单元会造成硬件资源的浪费. 为了改善方案可重构能力，提高模块对各种算法的适配性，我们提出了一种统一型电路结构，同时支持CT/GS结构，采用“交叉存储型”访存模式，分别对SNTT、TNTT、HNTT进行实现，可对主流格基密码算法提供加速方案. 针对主流格基密码算法，我们采用不同的NTT架构，其对应关系如表1所示.

表1　格基算法参数及其NTT结构

支持算法	参数$n/q/k=\lceil\log_2 q\rceil$	NTT结构
Dilithium	256/8380417/23	SNTT
Kyber	256/3329/12	TNTT
Aigis-512/768	256/7681/13	SNTT
Aigis-1024	512/12289/14	SNTT
OSKR-512/768	256/3329/12	TNTT
OSKR-1024	**512**/3329/12	**HNTT**
OKAI-512/768/1024-1	256/7681/13	SNTT
OKAI-1024-2	512/**7681**/13	**TNTT**

本文的贡献主要有以下3点：

(1)本文对格基密码方案中主流的数论变换算法进行总结，根据参数不同，将其分为SNTT、TNTT、HNTT. 针对Artix-7系列FPGA芯片，采用Verilog HDL语言进行纯硬件设计，对不同算法参数提供对应的加速方案，共包括三种NTT技术的完整硬件优化实现. 给出NTT计算两多项式相乘的完整设计，采用流水线方式，有效降低关键路径时延. 其中，TNTT和HNTT结合Karatsuba技术[33]进行了硬件实现，将对应系数相乘中乘法次数减少20%.

(2)针对NTT的核心操作——蝴蝶变换，设计紧凑型、低时延的电路结构，可实现正向NTT、对应系数相乘(Point-Wise Mulitiplication，PWM)、逆向NTT(Inverse-NTT，INTT)功能，同时支持CT/GS两种结构；提出新的系数访存模式，采用“交叉存储型”结构，将计算后的系数“对角交叉”变换位置后存储在原地址，利用双Bank存储模式满足蝴蝶变换单元的数据吞吐量；对Barrett约减算法[31]进行修改，提出适用于多模值的模约减硬件单元，用加法和移位代替其中的乘法操作，减少乘法器(Digital-Signal-Processing，DSP)资源的消耗.

(3)此外，对三种NTT高性能版本进行了研究实现. 通过复用蝴蝶单元，重新规划存储结构，优化数据存储逻辑，分别进行了二并行、四并行加速实现. 优化后的多并行架构提升了系统吞吐量，硬件消耗仅为单结构的1.46倍、2.58倍.

本文其余部分结构如下：第二节介绍预备知识，包括基本定义与符号表示、标准NTT、删减NTT以及混合NTT. 第三节介绍主要模块的硬件设计思路，包括蝴蝶单元、模约减单元、访存模式以及并行设计实现. 第四节列出设计结果并和当前公开发表成果进行对比，第五节为总结.

2　预备知识

本节主要介绍一些预备知识，包括：基本定义与符号表示、Karatsuba算法、标准NTT、删减NTT以及混合NTT.

2.1　基本定义与符号表示

记n和q为某些正整数，符号$\mathbb{Z}$表示整数集. 定义$\mathbb{Z}_q=\mathbb{Z}/q\mathbb{Z}\cong\{0,1,\cdots,q-1\}$. 本文约定$x'\equiv x\ (mod\ q)$表示$x'$与$x$对模$q$同余；$x'=x\ mod\ q$表示$x'$为$x$模$q$的结果. 本文使用多项式环$\mathbb{R}_q=\mathbb{Z}_q[x]/(x^n+1)$，其中$n$为2的幂次，$q$为素数. $\mathbb{R}_q$中每个元素是n维多项式且系数均在$\mathbb{Z}_q$中.

本文的多项式使用小写字母表示，如f，默认次

数为$n-1$. f可以表示为幂级数形式$f=\sum_{i=0}^{n-1} f_i x^i$，或者向量形式$f=(f_0,f_1,\cdots,f_{n-1})$，默认为行向量. 带“^”符号的字母如$\hat{f}$表示NTT域中的多项式. 符号“∘”表示多项式系数对应点乘(PWM).

定义1. 比特翻转. 记br_m为m位无符号数比特翻转操作. 具体而言，对于m位无符号数b，记其二进制展开为$b=(b_{m-1},\cdots,b_1,b_0)_2$，则$br_m(b)=br_m(b_{m-1}2^{m-1}+\cdots+b_1 2+b_0)=b_0 2^{m-1}+\cdots+b_{m-2}2+b_{m-1}$.

定义2. Karatsuba技巧[33]. 令a, b, c, d为四个常数或多项式. 为了计算$s_1=a\cdot c$, $s_2=a\cdot d+b\cdot c$, $s_3=b\cdot d$, Karatsuba技巧首先计算s_1,s_3，然后计算$s_2=(a+b)\cdot(c+d)-s_1-s_3$. 可以看出，直接计算$s_1$, s_2, s_3，一共需要1个加法和4个乘法操作. 使用Karatsuba技巧后，一共需要4个加法和3个乘法操作，减少1次乘法计算. 在硬件设计中，DSP资源相对稀缺，加法器资源则十分充足. 将Karatsuba技巧应用到硬件设计中，可以节省时钟周期消耗.

2.2 标准NTT

数论变换(Number-Theoretic Transform，NTT)是离散傅立叶变换(Discrete Fourier Transform，DFT)在有限域上的特殊情形. 标准的基于负折叠卷积[34-36]的n点NTT需要参数满足$q\equiv 1(mod\ 2n)$，其中n是2的幂次，q是素数. 正向变换为$\hat{f}=NTT(f)$，表示为

$$\hat{f}_j=\sum_{i=0}^{n-1} f_i\zeta^i\omega^{ij}\ mod\ q,\ j=0,1,\cdots,n-1,$$

其中，ζ是$\mathbb{Z}_q$中$2n$次本原单位根，即$\zeta^{2n}\equiv 1(mod\ q)$，且$\zeta^i\neq 1(mod\ q)$, $0<i<2n$，并取$\omega\equiv\zeta^2(mod\ q)$. 逆向变换为$f=INTT(\hat{f})$，表示为

$$f_i=n^{-1}\zeta^{-i}\sum_{j=0}^{n-1}\hat{f}_j\omega^{-ij}\ mod\ q,\ i=0,1,\cdots,n-1.$$

此时，$\mathbb{R}_q$上多项式乘法$h=f\cdot g\in\mathbb{R}_q$可通过下式计算得到：

$$h=INTT\big(NTT(f)\circ NTT(g)\big),$$

它的详细计算过程如算法1所示. 为方便起见，我们记上述满足n和q条件的NTT记为标准NTT(SNTT). 值得注意的是，方案Dilithium[12]，Aigis[13]，OKAI-512/768/1024-1[8]，Kyber-v1[14]均使用SNTT计算多项式乘法，因为它们的参数n和q均满足SNTT的条件.

算法1. 利用SNTT计算$\mathbb{R}_q$上多项式乘法.

输入：$f\in\mathbb{R}_q$

输入：$g\in\mathbb{R}_q$

输出：$h=f\cdot g\in\mathbb{R}_q$

1. $\hat{f}=NTT(f)$
2. $\hat{g}=NTT(g)$
3. $\hat{h}=\hat{f}\circ\hat{g}$
4. $h=INTT(\hat{h})$

FFT-trick[37]是计算NTT的快速算法，其复杂度为$O(nlogn)$，而直接计算NTT的复杂度为$O(n^2)$. 由于$\zeta^n=-1(mod\ q)$，则有$x^n+1=(x^{n/2}-\zeta^{n/2})(x^{n/2}+\zeta^{n/2})$. FFT-trick使用中国剩余定理[37]首先得到：$\mathbb{Z}_q[x]/(x^n-\zeta^n)\rightarrow\mathbb{Z}_q[x]/(x^{n/2}-\zeta^{n/2})\times\mathbb{Z}_q[x]/(x^{n/2}+\zeta^{n/2})$. 注意到，$x^{n/2}+\zeta^{n/2}=x^{n/2}-\zeta^{n+n/2}$，则FFT-trick能够一直持续下去直到得到$\mathbb{Z}_q[x]/(x-\zeta^i)$中的像. 完整的FFT-trick树形图表示见图1. 记$m=\lceil\log_2(n)\rceil$，则完整的FFT-trick过程可以表示为

$$\mathbb{Z}_q[x]/(x^n+1)\cong\prod_{i=0}^{n-1}\mathbb{Z}_q[x]/(x-\zeta^{2br_m(i)+1}).$$

利用FFT-trick计算SNTT的正向变换、对应系数点乘、逆向变换的迭代型伪代码[35]分别如算法

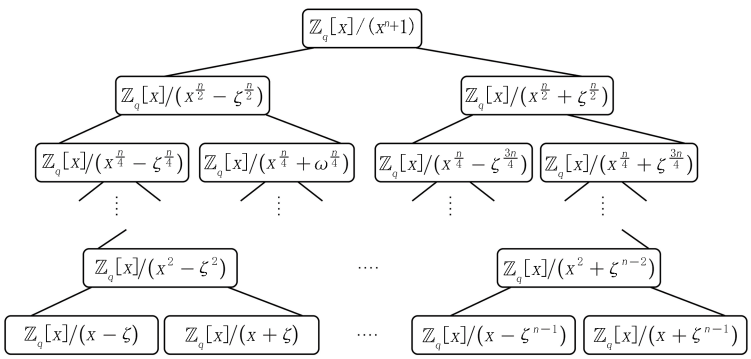

图1 完整的FFT-trick树形图

2、算法3和算法4所示．其中，正向变换使用了Cooley-Tukey（CT）蝴蝶结构[38]（见图2）的时域抽取法（Decimation-In-Time，DIT），逆向变换使用了Gentleman-Sande（GS）蝴蝶结构[39]（见图3）的频域抽取法（Decimation-In-Frequency，DIF）．这使得正向变换以自然顺序排列的多项式系数作为输入，计算输出比特翻转排列的多项式系数；逆向变换以比特翻转排列的多项式系数作为输入，计算输出自然顺序排列的多项式系数．

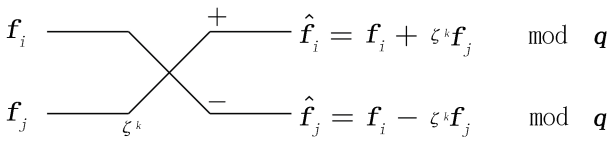

图2　Cooley-Tukey蝴蝶

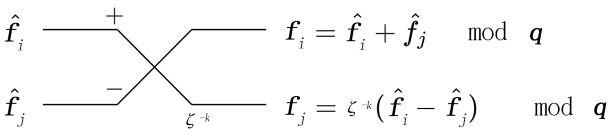

图3　Gentleman-Sande蝴蝶

算法2.　SNTT迭代型正向变换．

输入：自然顺序多项式$f=(f_0,f_1,\cdots,f_{n-1})$

输入：预计算得到的本原单位根$\zeta\in\mathbb{Z}_q$，$\zeta_k=\zeta^{br_m(k)}$，$0\leqslant k<n$，$m=\lceil\log_2(n)\rceil$

输出：比特翻转顺序多项式$\hat{f}=\left(\hat{f}_0,\hat{f}_{br_m(1)},\cdots,\hat{f}_{br_m(n-1)}\right)$

1. $k\leftarrow 1$
2. FOR $l\leftarrow n/2$; $l>0$; $l\leftarrow l/2$ DO
3.　FOR $s\leftarrow 0$; $s<n$; $s\leftarrow j+l$ DO
4.　FOR $j\leftarrow s$; $j<s+l$; $j\leftarrow j+1$ DO
5.　$t\leftarrow\zeta_k\cdot f_{j+l}$
6.　$\hat{f}_{j+l}\leftarrow f_j-t$
7.　$\hat{f}_j\leftarrow f_j+t$
8.　END FOR
9.　$k\leftarrow k+1$
10.　END FOR
11. END FOR

算法3.　SNTT对应点乘．

输入：NTT域多项式$\hat{f}=\left(\hat{f}_0,\hat{f}_{br_m(1)},\cdots,\hat{f}_{br_m(n-1)}\right)$

输入：NTT域多项式$\hat{g}=\left(\hat{g}_0,\hat{g}_{br_m(1)},\cdots,\hat{g}_{br_m(n-1)}\right)$

输出：NTT域多项式$\hat{h}=\hat{f}\cdot\hat{g}=\left(\hat{h}_0,\hat{h}_{br_m(1)},\cdots,\hat{h}_{br_m(n-1)}\right)$

1. FOR $k\leftarrow 0$; $k<n$; $k\leftarrow k+1$ DO
2.　$\hat{h}_k=\hat{f}_k\cdot\hat{g}_k$
3. END FOR

算法4.　SNTT迭代型逆向变换．

输入：比特翻转顺序多项式$\hat{f}=\left(\hat{f}_0,\hat{f}_{br_m(1)},\cdots,\hat{f}_{br_m(n-1)}\right)$

输入：预计算得到的本原单位根$\zeta\in\mathbb{Z}_q$，$\zeta_k=\zeta^{-(br_m(k)+1)}$，$0\leqslant k<n$，$m=\lceil\log_2(n)\rceil$

输出：自然顺序多项式$f=(f_0,f_1,\cdots,f_{n-1})$

1. $k\leftarrow 0$
2. FOR $l\leftarrow 1$; $l>n$; $l\leftarrow 2l$ DO
3.　FOR $s\leftarrow 0$; $s<n$; $s\leftarrow j+l$ DO
4.　FOR $j\leftarrow s$; $j<s+l$; $j\leftarrow j+1$ DO
5.　$t\leftarrow\hat{f}_j$
6.　$f_j\leftarrow t+\hat{f}_{j+l}$
7.　$f_{j+l}\leftarrow t-\hat{f}_{j+l}$
8.　$f_{j+l}\leftarrow\zeta_k\cdot f_{j+l}$
9.　END FOR
10.　$k\leftarrow k+1$
11.　END FOR
12. END FOR
13. FOR $j\leftarrow 0$; $j<n$; $j\leftarrow j+1$ DO
14.　$f_j\leftarrow f_j/n$
15. END FOR

2.3　删减NTT

在图1所示的FFT-trick树形图中，FFT-trick的计算过程并非必须持续到线性多项式．文献[17]注意到，FFT-trick树形图能够删减最后的β层，使得其计算过程只持续到$\mathbb{Z}_q[x]/\left(x^{2^\beta}-\zeta^i\right)$，即

$$\mathbb{Z}_q[x]/(x^n+1)\cong\prod_{i=0}^{n/2^\beta-1}\mathbb{Z}_q[x]/(x^{2^\beta}-\zeta^{2br_m(i)+1}),$$

其中，ζ是$\mathbb{Z}_q$中$\dfrac{2n}{2^\beta}$次本原单位根，β为小于$\lceil\log_2(n)\rceil$的非负整数，$m'=\lceil\log_2(n)\rceil-\beta$．正向变换得到$\dfrac{n}{2^\beta}$个次数为$2^\beta-1$的多项式．点乘包含对应$2^\beta-1$次多项式的乘法．此时多项式环的参数$n$和$q$只需要满足$q\equiv 1\left(mod\dfrac{2n}{2^\beta}\right)$．文献[11]称这种“底部删减层数”的NTT变体为删减NTT（Truncated-NTT，TNTT），并证明当$\beta=1$时，其复杂度最小．当$\beta=1$时，TNTT的FFT-trick树形图见图4所示，其中，ω是$\mathbb{Z}_q$中n次本原单位根．

方案Kyber-v2/v3[15-16]，OSKR-512/768/1024-1[8]，OKAI-1024-2[8]均使用了$\beta=1$的TNTT．其中，Kyber为了达到了更好的性能，从第二轮开始采用TNTT技术，在$n=256$保持不变时将模值从7681改为了更小的3329，此时q只满足$q\equiv 1(mod\,n)$．此

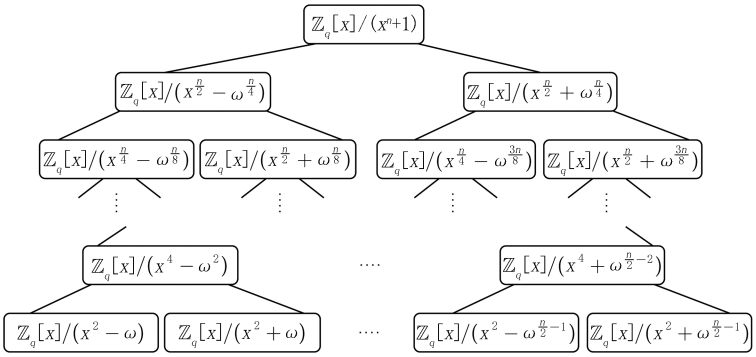

图4 $\beta=1$时TNTT的FFT-trick树形图

时，$\mathbb{R}_q$中多项式f经过正向变换后的结果$\hat{f}$即为

$$\hat{f}=\left(\hat{f}_0+\hat{f}_1x,\hat{f}_2+\hat{f}_3x,\cdots,\hat{f}_{254}+\hat{f}_{255}x\right)$$

在$\hat{f}$和$\hat{g}$的点乘运算中，对于$i=0,\cdots,\frac{n}{2}-1$有：

$$\hat{h}_{2i}+\hat{h}_{2i+1}x=\left(\hat{f}_{2i}+\hat{f}_{2i+1}x\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}x\right)\ mod\left(x^2-\zeta^{2br_7(i)+1}\right)$$

2.4 混合NTT

文献[40]提出基于“顶部分解”方法的预处理NTT(Preprocess-then-NTT，Pt-NTT). 随后文献[41]中结合Karatsuba技巧进一步改进Pt-NTT，减少了乘法计算量. 文献[8,11]对Pt-NTT、TNTT进行了系统分析，证明了两者计算等价性，并结合“顶部分解”和“底部删减”两种思想，提出了混合NTT(Hybrid-NTT，HNTT). 特别地，Pt-NTT、TNTT可以看作是HNTT的特殊形式.

给定某些非负整数α和β，HNTT计算$\mathbb{R}_q$上多项式乘法$h=f\cdot g\in\mathbb{R}_q$的过程具体为

第一步，顶部分解. 将原始多项式f, g从顶部分为2^α个子多项式：

$$f(x)=\sum_{i=0}^{2^\alpha-1}x^i\cdot\tilde{f}_i\left(x^{2^\alpha}\right)$$
$$g(x)=\sum_{i=0}^{2^\alpha-1}x^i\cdot\tilde{g}_i\left(x^{2^\alpha}\right)$$

其中，$\tilde{f}_i(y)=\sum_{j=0}^{\frac{n}{2^\alpha}-1}f_{2^\alpha j+i}y^j$，$\tilde{g}_i(y)=\sum_{j=0}^{\frac{n}{2^\alpha}-1}g_{2^\alpha j+i}y^j\in\mathbb{Z}_q[y]/\left(y^{\frac{n}{2^\alpha}}+1\right)$.

第二步，NTT计算. 对于$i=0,\cdots,2^\alpha-1$，在$\mathbb{Z}_q[y]/\left(y^{\frac{n}{2^\alpha}}+1\right)$中计算：

$$\tilde{h}_i=\sum_{l=0}^{i}\tilde{f}_l\cdot\tilde{g}_{i-l}+\sum_{l=i+1}^{2^\alpha-1}y\cdot\tilde{f}_l\cdot\tilde{g}_{2^\alpha+i-l}$$
$$=TNTT^{-1}\left[\sum_{l=0}^{i}TNTT\left(\tilde{f}_l\right)\circ TNTT\left(\tilde{g}_{i-l}\right)+\sum_{l=i+1}^{2^\alpha-1}TNTT(y)\circ TNTT\left(\widetilde{f}_l\right)\circ TNTT\left(\tilde{g}_{2^\alpha+i-l}\right)\right]$$

其中，$y=x^{2^\alpha}$，上述过程使用了在$\mathbb{Z}_q[y]/\left(y^{\frac{n}{2^\alpha}}+1\right)$上的从底部删减$\beta$层的$\frac{n}{2^\alpha}$点TNTT，并且$TNTT(y)$能够预先计算好并储存. 结合Karatsuba技巧，首先计算$TNTT\left(\tilde{f}_i\right)\circ TNTT\left(\tilde{g}_i\right),\forall i=j$，然后对于任意的$i\neq j$计算：

$$TNTT\left(\tilde{f}_i\right)\circ TNTT\left(\tilde{g}_j\right)+TNTT\left(\tilde{f}_j\right)\circ TNTT\left(\tilde{g}_i\right)$$
$$=\left(TNTT\left(\tilde{f}_i\right)+TNTT\left(\tilde{f}_j\right)\right)\circ\left(TNTT\left(\tilde{g}_i\right)+TNTT\left(\tilde{g}_j\right)\right)-TNTT\left(\tilde{f}_i\right)\circ TNTT\left(\tilde{g}_i\right)-TNTT\left(\tilde{f}_j\right)\circ TNTT\left(\tilde{g}_j\right)$$

同时在删减β层的$\frac{n}{2^\alpha}$点TNTT的对应点乘计算过程中使用Karatsuba技巧. 例如，计算$\left(\sum_{i=0}^{2^\beta-1}\hat{f}_ix^i\right)\left(\sum_{i=0}^{2^\beta-1}\hat{g}_ix^i\right)mod\,x^{2^\beta}-\zeta$时，首先计算$\hat{f}_i\cdot\hat{g}_i,\forall i=j$，然后对于任意的$i\neq j$计算：

$$\hat{f}_i\cdot\hat{g}_j+\hat{f}_j\cdot\hat{g}_i=\left(\hat{f}_i+\hat{f}_j\right)\left(\hat{g}_i+\hat{g}_j\right)-\hat{f}_i\cdot\hat{g}_i-\hat{f}_j\cdot\hat{g}_j$$

第三步，多项式重组. 计算：

$$h(x)=\sum_{i=0}^{2^\alpha-1}x^i\cdot\tilde{h}_i\left(x^{2^\alpha}\right)$$

HNTT对参数n和q的条件能够弱化到$q\equiv1\left(mod\,\frac{2n}{2^{\alpha+\beta}}\right)$. 文献[11]指出，当$\alpha=\beta=1$时，HNTT的复杂度最小. 注意到，文献[8]提出的OSKR-1024-2的参数n和q满足$q\equiv1\left(mod\,\frac{n}{2}\right)$，故它采用$\alpha=\beta=1$的HNTT加速计算多项式乘法.

3　硬件设计与实现

本节主要介绍NTT计算核心模块设计与实现，包括紧凑型蝴蝶操作单元，可重构的模约减单元，“交叉存储型”访存模式以及高性能并行优化设计.

3.1　紧凑型蝴蝶操作单元

NTT技术加速多项式乘法的计算过程包括正向NTT、对应系数点乘计算、逆向NTT，其核心是蝴蝶计算. SNTT正向、逆向共包含$2\times\log_2 n$层计算，每层$n/2$个蝴蝶计算；TNTT共包含$2\times(\log_2 n-\beta)$层，每层$n/2$个蝴蝶计算；HNTT共包含$2\times\left[2^{\alpha}\times(\log_2 n/2-\beta)\right]$层，每层$n/4$个蝴蝶计算. 蝴蝶单元(Butterfly Unit，BFU)的硬件设计关系到系统的时间性能及面积性能. 目前方案大多只支持一种CT/GS结构，正向NTT结束之后，需要花费额外的时钟周期调整系数存储位置. 为此，我们提出一种新的多功能、紧凑型电路结构，同时支持CT、GS结构以及对应点乘操作，通过1比特片选信号切换功能. 正向NTT采用CT结构，以自然顺序输入，比特翻转顺序输出. 逆向NTT采用GS结构，以比特翻转顺序输入，自然顺序输出.

蝴蝶单元硬件电路结构如图5所示. 图中a、b、w、sel为输入端口，A、B为输出端口，位宽为$\lceil\log_2 q\rceil$比特. 模块包含两个模加器、两个模减器、一个乘法器、一个约减单元. 为了缩短数据关键路径，分级隔绝计算步骤的中间数据，共插入两级寄存器. 加上DSP、约减模块，蝴蝶单元共消耗7个时钟周期. 当片选信号sel为0时，进行正向NTT计算，BRAM存储的多项式系数对从a、b端口输入，ROM存储的本原单位根ω从w端口输入. 此时，$A=a+b\cdot\omega \bmod q$，$B=a-b\cdot\omega \bmod q$；当片选信号sel为1时，进行逆向NTT计算，此时$A=(a+b)/2 \bmod q$，$B=(a+b)\cdot\omega/2 \bmod q$.

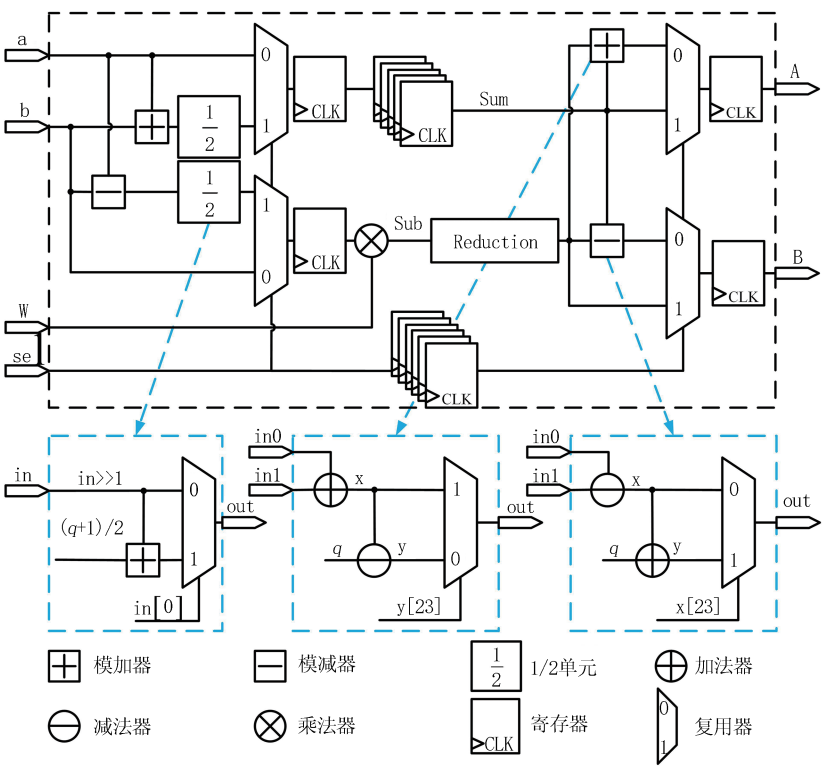

图5　紧凑型蝴蝶操作单元

为了消去逆向NTT算法4中13-15行除以n带来的额外时钟消耗，我们采用Zhang对Newhope的处理方式[42]，在每层计算时直接对系数进行乘$\frac{1}{2} \bmod q$操作，其电路结构如图5左下方所示. 该模乘计算在硬件设计时只需移位和模加操作即可实现. 素数q确定的情况下，$1/2 \bmod q$与$(q+1)/2$等价. 当处理数x为偶数时，$x/2 \bmod q$转换为$x\gg 1$；当x为奇数时，$x/2$则变为

$$\frac{x}{2}\equiv\left(2\left\lfloor\frac{x}{2}\right\rfloor+1\right)\frac{q+1}{2}\equiv\left\lfloor\frac{x}{2}\right\rfloor+\frac{q+1}{2}\ mod\ q,$$

其中，$\left\lfloor\frac{x}{2}\right\rfloor$转换为$x\gg1$，$(q+1)/2$为常数，可预先设置.

SNTT、TNTT、HNTT对应点乘计算操作数虽然不同，但都包含乘法计算，单独对其进行电路设计会带来额外的DSP资源开销. 为此，我们重新规划数据流向，使用蝴蝶单元的乘法器计算对应点系数乘法. SNTT对应点乘计算时，将两多项式系数f_i、g_i分别从b、w端口输入，片选信号sel设置为0，a端口输入固定值0，此时B端口输出计算结果$B=f_i\cdot g_i$. TNTT、HNTT对应点乘计算使用Karatsuba技巧，4次乘法复用蝴蝶单元中的乘法器，数据送入模块需要进行预处理. TNTT、HNTT对应点乘计算数据链路时序设计、存储逻辑在第3.3小节中详细阐述.

针对Xilinx公司28纳米工艺的Artix-7系列XC7A200TFBG484-2 FPGA芯片，优化后的蝴蝶单元共消耗261个查找表(Look-Up-Table，LUT)，108个触发器(Flip-Flop，FF)，165个片(Slice)，1个乘法器(Digital-Signal-Processing，DSP).

3.2 可重构的模约减单元

NTT计算过程中的系数是在域$\mathbb{Z}_q$中进行的，其中间乘法结果需要进行模约减，通常采用的模约减算法有Montgomery约减[30]和Barrett约减[31]. 两种算法都包含两次乘法计算，不同的是，Montgomery算法需要先将系数转变到Montgomery域内计算，这种操作会导致中间计算结果位宽的增加. 这在软件实现时对内存的影响较小，因为软件实现变量位宽固定，其位宽相对于中间结果甚至有冗余. 例如Kyber-v3中模值q为3329，占据12比特，而所创建的变量为int16_t类型，有16比特，存在4比特冗余. 扩展到Montgomery域内的中间计算结果不会造成过多内存浪费. 但硬件实现的数据位宽可自定义，中间结果导致的位宽扩展会占用较多的硬件资源. 此外，算法中的乘法计算在实现时可选择增加额外的DSP资源，或者将模约减单元与其他模块进行分时复用DSP资源. 前者增加了硬件资源的开支，后者增加了算法运行时间上的开支.

为此，针对主流格基密码方案，我们提出一种适用于多模值、可常数时间执行的模约减单元，对修改后的Barrett算法进行硬件电路设计，将算法中的乘法计算用加法和移位操作代替，减少了DSP资源的占用. 当模值q确定时(以$q=8\,380\,417=2^{23}-2^{13}+2^0$为例)，计算过程如算法5所示.

算法5. 改进后的Barrett约减算法.

输入：模值$q=8\,380\,417$，位宽为$\lceil\log_2 q\rceil=23$

输入：待约减数ω，最大位宽为$2\cdot\lceil\log_2 q\rceil=46$

输出：约减后的值$x\equiv\omega\ mod\ q$

1. $k=\lceil\log_2 q\rceil=23$
2. $\mu=\left\lfloor\frac{2^{2k}}{q}\right\rfloor=\left\lfloor\frac{2^{46}}{8\,380\,417}\right\rfloor=8\,396\,807=2^{23}+2^{13}+2^3-2^0$
3. $r\leftarrow\left\lfloor\left\lfloor\frac{\omega}{2^{k-1}}\right\rfloor\cdot\frac{\mu}{2^{k+1}}\right\rfloor=\omega\gg23+\omega\gg33+\omega\gg43$
4. $x\leftarrow\omega-r\cdot q=\omega-(r\ll23-r\ll13+r)$
5. WHILE $x<0$ DO
6. $\quad x\leftarrow x+q$
7. END WHILE
8. WHILE $x>q$ DO
9. $\quad x\leftarrow x-q$
10. END WHILE
11. RETURE x

算法5以Dilithium算法的模值为例，其中μ为确定值，乘法操作可以通过预先计算好的移位和加减替代. 本模块结构同样适用于其他模值，对于Kyber模数$q=3329=2^{11}+2^{10}+2^8+2^0$，其$\mu=\left\lfloor\frac{2^{24}}{3329}\right\rfloor=5040=2^{12}-2^{10}-2^6-2^4$；对于Aigis模数$q=7681=2^{13}-2^9+2^0$，其$\mu=\left\lfloor\frac{2^{26}}{7681}\right\rfloor=8736=2^{13}+2^9+2^5$. 调整算法后续的移位量，预设不同的值即可对不同模值进行约减. 模块从输入到输出端口的关键路径包含4个加法器，2个减法器，2个复用器，为了提升模块最大时钟频率，在其中插入4个寄存器，模约减操作共耗时4个时钟周期，优化后的电路结构如图6所示. 针对Xilinx公司28纳米工艺的Artix-7系列XC7A200TFBG484-2 FPGA芯片，优化后的约减模块共消耗硬件资源：113个LUT，47个FF，39个Slice，0个DSP.

3.3 “交叉存储型”访存模式

我们给出了采用NTT计算两多项式相乘的完整实现，多项式系数及中间计算结果暂存到片内BRAM. 计算开始之前，首先256/512维多项式系数暂存RAM0内. 然后，依次对多项式做正向NTT计算，结果分别返回RAM0. 接着，开始计算NTT域内对应点相乘操作，计算结果存入RAM0. 最后，从

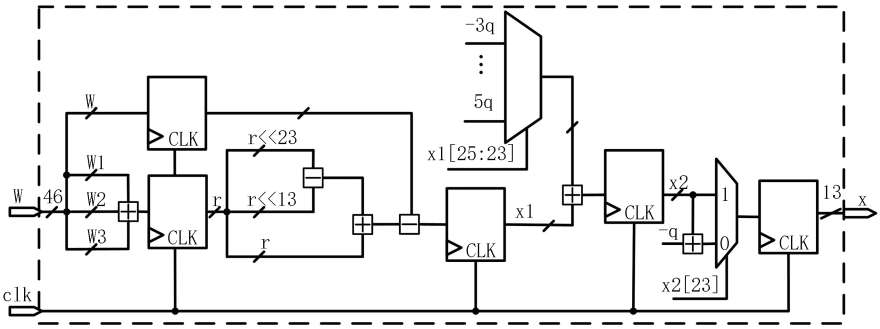

图6　可重构型模约减单元

RAM0内读取数据，计算逆向NTT，最终结果返回RAM0内.

数论变换常见的数据存储结构有原位读写结构[28]、几何不变结构[27]、多路延迟换向结构(Radix-2 Multipath Delay Commutator，R2MDC)[25]以及单路延迟反馈结构[43]. 原位存储结构多项式系数按照奇偶相邻存储，每一层运算之前从RAM内读取数据，经过蝴蝶变换操作后，数据存回RAM中原地址，各层之间的读取顺序不同. 为了保证送入蝴蝶单元的数据为相应数据对，需要对RAM读取出来的系数延迟打拍，进行错位匹配. 蝴蝶单元输出结果同样需要延迟后才能写入RAM，该结构数据读/写逻辑相对复杂. 几何不变结构采用“乒乓”读写逻辑，数据读写地址按顺序增加，每一层的读写逻辑相同. 该结构需要两倍的存储单元，RAM利用率只有50%. 多路延迟换向结构将各级的蝴蝶单元进行串联，通过复用选择器变换数据中间流向，保证数据正确配对. 单路延迟反馈结构与多路延迟换向结构同属于流水线型结构，但数据经过蝴蝶变换单元之后会反馈给输入位置. 这两种流水线型结构对多项式存储方式没有特定限制，只需要系数以串行方式输出给蝴蝶单元即可. 其数据读写逻辑简单，只需一次读取和一次写入，但硬件开销极大增加，RAM利用率只有50%且消耗额外的DSP资源.

蝴蝶操作一个时钟周期需要输入两个系数，输出两个计算结果，这要求存储单元一个周期内提供两点读取和两点存储操作. 为了满足蝴蝶操作数读写吞吐量，提高RAM复用率，我们提出一种新的内存访存模式——交叉存储结构，可同时支持三种NTT计算，内存利用率达100%. 该结构采用双口RAM，每个RAM分为Bank0、Bank1，两者读写地址保持一致，Bank内部每个单元存储一个系数. RAM设置为“简单双口模式”，其A端口支持读写操作，B端口仅支持读操作，在同一时钟周期内可以对该BRAM同时进行读写操作. 交叉存储结构多项式系数按照蝴蝶计算对进行存储，初始存储方式为Bank0存储0-127，Bank1存储128-255(以256维多项式为例). 在每一层正向/逆向NTT时，RAM内读取的两点数据直接输入蝴蝶单元，计算过后的结果先经过延迟打拍，相邻两点交叉调换后写入RAM. 这种交叉存储方式简化了模块读取逻辑，保证下一层读取的两点系数互为蝴蝶操作系数对. RAM的读取地址采用实时计算的方式生成，写入地址为读地址的延迟.

多项式维度$n=256$时，SNTT、TNTT做正向NTT计算时，各层计算后的系数存储方式如图7所示. 图中黄色部分和蓝色部分在计算后进行交叉存储，满足下一层计算数据配对需求. SNTT前7层做完之后都需要进行交叉存储操作，最后一层写入原地址，这是为了保证后续逆向NTT计算时送入蝴蝶单元的两点数据正好匹配. TNTT为SNTT从最后一层删减后的变体，因此其读写逻辑完全复用SNTT前7层. 多项式维度$n=512$时，HNTT首先从顶层将512维多项式分为前256维和后256维两个多项式，其访存模式可以复用TNTT，每个多项式系数存储方式与TNTT一致.

如算法3所示，SNTT对应系数点乘为一对一计算，从RAM0、RAM1中同时读取NTT域内的两个多项式对应系数，RAM0输出给蝴蝶单元的b输入口，RAM1输出给w端口，利用其内部的乘法单元做计算. 计算结果暂存到RAM0内，等待后续逆向NTT计算.

TNTT对应点乘与SNTT不同，为两点与两点做乘法计算. 根据2.1小节，利用Karatsuba技巧，可以将5次乘法减少为4次乘法计算过程如下：

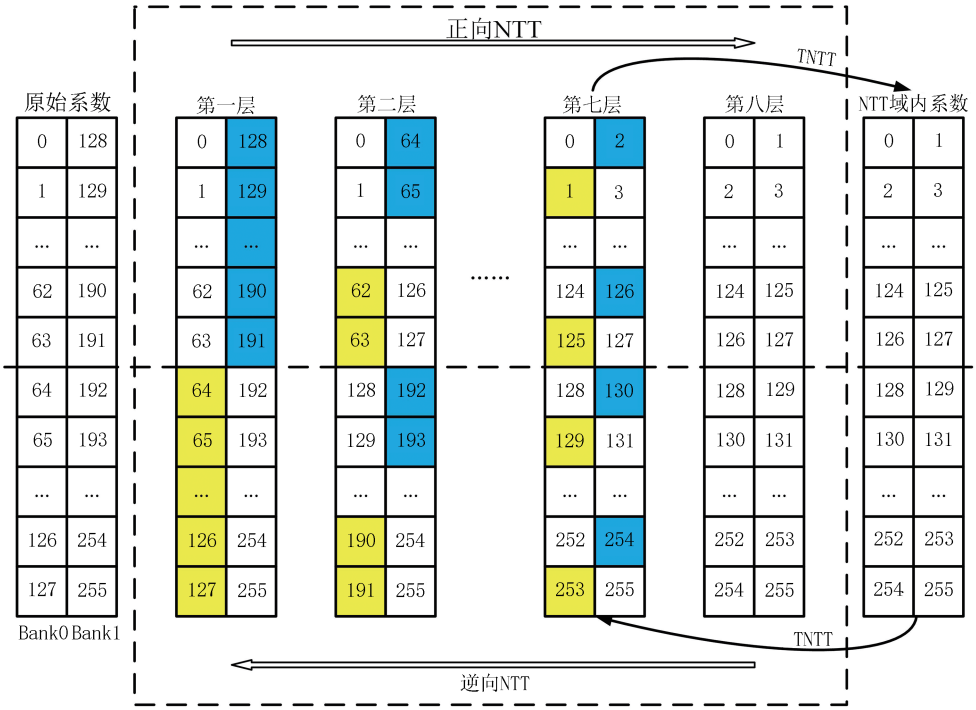

图7 “交叉存储型”访存模式

$$\left(\hat{f}_{2i}+\hat{f}_{2i+1}x\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}x\right)=\hat{f}_{2i}\hat{g}_{2i}+\hat{f}_{2i+1}\hat{g}_{2i+1}x^2+\left(\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)-\hat{f}_{2i}\hat{g}_{2i}-\hat{f}_{2i+1}\hat{g}_{2i+1}\right)x=\hat{f}_{2i}\hat{g}_{2i}+\hat{f}_{2i+1}\hat{g}_{2i+1}\zeta^{2br(i)+1}+\left(\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)-\hat{f}_{2i}\hat{g}_{2i}-\hat{f}_{2i+1}\hat{g}_{2i+1}\right)x$$

此时有，

$$\hat{h}_{2i}=\hat{f}_{2i}\hat{g}_{2i}+\hat{f}_{2i+1}\hat{g}_{2i+1}\zeta^{2br(i)+1}$$

$$\hat{h}_{2i+1}=\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)-\hat{f}_{2i}\hat{g}_{2i}-\hat{f}_{2i+1}\hat{g}_{2i+1}$$

通过复用蝴蝶单元乘法器，四个计算步骤时序安排如图8所示. 其中，红色字体表示“a”端口输入，蓝色字体表示“b”端口输入，绿色字体表示“w”端口输入. 为了避免数据覆盖、冲突，且最大化利用现有的RAM资源，我们只需扩展128点存储空间RAM2 Bank0，将其与RAM0复用即可存储对应点

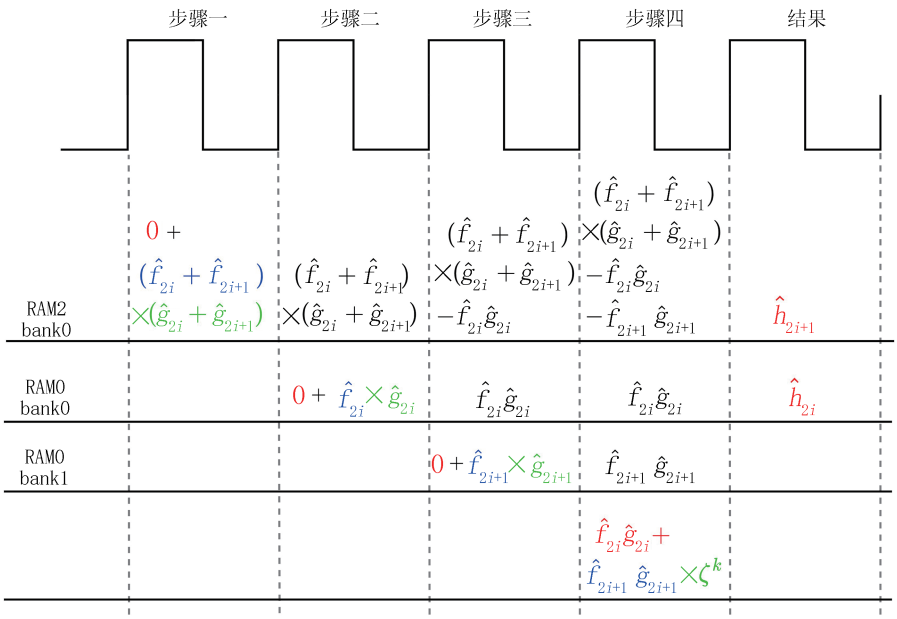

图8 结合Karatsuba技术的TNTT PWM时序设计

乘四个计算步骤的中间结果. 具体内存分配如下：

步骤一，计算$\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)$,$\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)$和$\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)$在顶层模块使用模加器进行预计算，然后输入到蝴蝶单元，输出结果存入RAM2 bank0；

步骤二，计算$\hat{f}_{2i}\times\hat{g}_{2i}$，结果存入RAM0 bank0，同时从RAM2 Bank0中读取数据计算$\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)-\hat{f}_{2i}\times\hat{g}_{2i}$，数据在顶层模块使用模减器进行后计算，结果继续返回写入RAM2 bank0；

步骤三，计算$\hat{f}_{2i+1}\times\hat{g}_{2i+1}$，结果存入RAM0 bank1，同时从RAM2 Bank0中读取数据计算$\left(\hat{f}_{2i}+\hat{f}_{2i+1}\right)\left(\hat{g}_{2i}+\hat{g}_{2i+1}\right)-\hat{f}_{2i}\times\hat{g}_{2i}-\hat{f}_{2i+1}\times\hat{g}_{2i+1}$，数据在顶层模块使用模减器进行后计算，结果继续返回写入RAM2 bank0，此时得到$\hat{h}_{2i+1}$；

步骤四，计算$\hat{f}_{2i}\times\hat{g}_{2i}+\hat{f}_{2i+1}\times\hat{g}_{2i+1}\times\zeta^{2br(i)+1}$，结果存入RAM0 bank0，此时得到$\hat{h}_{2i}$.

至此，TNTT对应点乘计算结束. 四个步骤中，前三步计算结果写入原地址，最后一步计算结果采用交叉存储方式写入RAM，以此保证后续逆向NTT第一层读取数据两两匹配. HNTT对应点乘内存分配与TNTT一致.

SNTT、TNTT、HNTT做逆向NTT计算时，各层计算后系数存储方式与正向NTT相反，具体如图7所示. SNTT前7层同样进行交叉存储操作，最后一层选择写入原地址，这样可以保证数据正序输出.

TNTT逆向NTT操作读写逻辑与SNTT后7层完全一致. HNTT逆向NTT过程同样复用TNTT.

对于256维多项式相乘，两个多项式SNTT架构共消耗256×2=512点RAM空间，Dilithium消耗512×23=11 776比特，Aigis-512/768消耗512×13=6656比特，OKAI消耗512×13=6656比特，Kyber-v1消耗512×13=6656比特；TNTT架构共消耗256×2+128=640点RAM空间，Kyber-v2/v3消耗640×12=7680比特，OSKR-512/768消耗640×12=7680比特.

对于512维的两个多项式相乘，采用SNTT架构共消耗512×2=1024点RAM空间，Aigis-1024消耗1024×14=14 336比特；采用TNTT架构共消耗512×2+256=1280点RAM空间，OKAI-1024-2消耗1280×13=16 640比特；采用HNTT架构共消耗512×2+128×2=1280点RAM空间，OSKR-1024消耗1280×12=15 360比特.

SNTT蝴蝶计算用到的ζ、TNTT、HNTT蝴蝶计算用到的ω的生成方式通常有在线计算和预计算存储两种. 在线计算方式预先将计算过程模块化，在蝴蝶计算时实时生成各层所需的ω，该方式需消耗大量计算资源. 为了追求简洁化设计，我们采用预计算存储方式，将预先计算得到的ω存储在片内ROM里，在蝶形计算时依次读取相应的值输入到蝴蝶单元. 此外，TNTT、HNTT对应点乘操作需要ω值参与计算，该值同样预存在ROM中.

对于256维多项式相乘，两个多项式SNTT架构共消耗256×2=512点ROM空间，Dilithium消耗512×23=11 776比特，Aigis消耗-512/768消耗512×13=6656比特，OKAI消耗512×13=6656比特，Kyber-v1消耗512×13=6656比特；TNTT架构共消耗128×2+128=384点ROM空间，Kyber-v2/v3消耗384×12=4608比特，OSKR-512/768消耗384×12=4608比特.

对于512维的两个多项式相乘，采用SNTT架构共消耗512×2=1024点ROM空间，Aigis-1024消耗1024×14=14 336比特；采用TNTT架构共消耗256×2+256=768点ROM空间，OKAI-1024-2消耗768×13=9984比特；采用HNTT架构共消耗256×2+128×2=768点RAM空间，OSKR-1024消耗768×12=9216比特.

3.4　高性能并行优化设计

上述小节介绍了环上多项式采用三种单结构NTT相乘的计算过程及硬件设计. 为了进一步提高系统吞吐量，我们通过复用蝴蝶单元、优化数据流向的方式，对三种NTT架构实现了二并行、四并行设计. 相较于直接复用全部模块，优化后的多并行设计在面积、性能上都有明显提升.

二并行设计采用两个蝴蝶单元BFU0、BFU1，将其用于层级内部，每层进行二并行加速. 其中BFU0用于处理前64对数据，BFU1用于处理后64对数据，两者读写逻辑一致. 为了保证二并行同一时钟周期4点输入、4点输出的数据吞吐量，我们将RAM0拆分成RAM0、RAM2. 多项式系数按照蝴蝶计算对存储，初始存储方式为RAM0 Bank0存储0-63，RAM2 Bank0存储64-127，RAM0 Bank1存储128-191，RAM2 Bank1存储192-255(以256维多项式为例). 对RAM中系数做二并行计算，每一层计算结果重新存入原Bank. 三种NTT架构的二并行

设计消耗时钟周期数减少一倍，综合、布局布线后的硬件资源仅为单结构的1.45倍. 四并行设计复用四个蝴蝶单元，数据流优化过程与二并行一致. 不同的是，四并行RAM存储结构分为8个Bank，每个Bank存储32点系数. 四并行设计消耗时钟周期数为单结构的1/4，硬件资源仅为单结构的2.58倍.

4 结果测试

本方案采用Verilog HDL语言进行了纯硬件实现，开发环境为Vivado 2017.4，针对28纳米工艺FPGA芯片Xilinx Artix-7系列XC7A200TFBG484-2型号做了仿真、综合、布局布线. 单结构、二并行、四并行整体性能及硬件消耗对比如表2所示. 根据我们的调研，目前没有同时支持三种NTT架构的设计，大多数只支持其中一种参数. 因此，在对实现结果进行分析时，我们只对相同NTT架构之间进行比较. 我们设计的紧凑型、低时延硬件架构支持多种算法参数，硬件资源消耗最多减少62.6%，计算速度最多提升49.8%.

表2 时间性能及硬件消耗对比

方案	蝴蝶单元	支持参数($n/q/k$)	NTT结构	NTT/INTT/PWM时钟数	最大时钟频率(MHz)	耗时(us)	LUT/FF/Slice/BRAM/DSP	设计语言	平台
本设计单结构	1	256/8 380 417/23	SNTT	1031/1031/256	216.5	4.76	655/213/235/1/1	Verilog	Artix-7
		256/7681/13							
		256/3329/12	TNTT	903/903/512		4.17			
		512/3329/12	HNTT	1799/1799/1024		8.31			
本设计二并行	2	256/8 380 417/23	SNTT	519/519/128	209.6	2.48	956/378/317/2/2	Verilog	Artix-7
		256/7681/13							
		256/3329/12	TNTT	455/455/256		2.17			
		512/3329/12	HNTT	903/903/512		4.31			
本设计四并行	4	256/8 380 417/23	SNTT	263/263/64	201.9	1.30	1688/672/606/4/4	Verilog	Artix-7
		256/7681/13							
		256/3329/12	TNTT	231/231/128		1.14			
		512/3329/12	HNTT	455/455/256		2.25			
文献[25]	4	256/8 380 417/23	SNTT	296/296/128	172	1.72	1919/1301/-/2/10	Verilog	Artix-7
文献[23-24]	2×2	256/8 380 417/23	SNTT	300/300/108	116	2.59	4509/3146/-/-/8	Verilog	Artix-7
文献[42]	2	512/12 289/14	SNTT	1289/-/-	245	5.3	741/330/-/5/2	Verilog	Artix-7
文献[28]	2	256/3329/12	TNTT	512/512/256	161	3.18	1737/1167/-/3/2	Verilog	Artix-7
文献[29]	2×2	256/3329/12	TNTT	324/324/128	222	1.46	801/717/-/2/4	VHDL	Artix-7
文献[27]	1	256/3329/12	TNTT	904/904/647	182	4.97	948/352/-/1.5/1	Verilog	Artix-7
	4			233/233/167	190	1.23	2543/792/-/9/4		
	16			71/71/47	182	0.39	9508/2684/-/35/16		
文献[44]	8	256/3329/12	TNTT	1052/1318/3688	117	8.99	2119/1058/-/3/8	Verilog	Artix-7 Virtex-7
	64			156/198/552	140	1.11	11K/5182/-/12/64		
	256			95/112/319	126	0.75	63K/18K/-48/256		
文献[22]	1	256/-/13	SNTT	919 570/-/-	-	-	893/-/-/3/2	Verilog	Virtex-7
	1			6147/-/-	-	-	979/-/-/2/3	HLS	
	8			3075/-/-	-	-	10487/-/-/16/24	HLS	

Zhao[25]对Dilithium参数进行了SNTT实现，采用修改后的R2MDC架构，使用4个蝴蝶单元，通过设置额外的转换器使得不同层级间的数据流可以复用相同的蝴蝶单元，最终蝴蝶单元利用率同样可以达到100%，冗余40个时钟周期. 蝴蝶单元电路内部计算需要消耗固定时钟周期，因此R2MDC冗余时钟周期与复用蝴蝶单元个数成正比. 此外，该设计删去R2MDC结构中最前端的延迟单元，将内存使用率提升至100%. 该电路结构仅支持正向/逆向NTT操作，因此需要额外消耗2个DSP来计算对应点乘操作. 其硬件资源消耗相对较少，但模块关键路径太长导致系统最大时钟频率太低. 与之相比，本设计硬件资源减少12%，速度提升了24.4%. Luke[23-24]同样对Dilithium多项式乘法进行了加速实

现，该方案采用2×2架构，同时对NTT相邻两层进行加速计算，每层使用2个蝴蝶单元，共计4个. 其系数按照每四个存入一个BRAM单元，读取地址采用预存方式，每层的读取地址存储在ROM内. 将ROM内数据直接输出给RAM的地址端口，即可满足蝴蝶单元计算需求. 这种架构有其自身的局限性，仅适用于含有偶数层的NTT参数($\log_2 n$为偶数)，对奇数层的算法并不友好. 与该方案相比，本设计硬件资源减少了62.6%，速度提升了49.8%. Zhang[42]对NewHope参数进行了简洁化设计，其蝴蝶单元只使用一个模加、一个模减和一个模乘，通过寄存器打拍和复用器改变数据流向，减少了硬件消耗. 值得一提的是，该方案约减模块针对模值进行固定的幂指数替代，电路架构与模值高度绑定，对不同模值的适用性较低. 我们SNTT架构可以用来计算512维多项式乘法，二并行折算后消耗1159个时钟周期，减少10.1%. 与采用HNTT计算512维多项式，本设计计算速度提升5.5%.

目前，一些学者对Kyber第三轮参数NTT加速实现进行了研究实现. Xing[28]对Kyber参数设计了多功能硬件电路，其方案支持正向/逆向NTT、对应点乘操作以及Kyber算法中压缩/解压缩计算. 内存读取逻辑采用原位存储架构，复用两个蝴蝶单元进行加速. 与之相比，本设计TNTT架构硬件资源减少45%，总耗时减少31.8%. Mojtaba[29]同样采用2×2电路架构，对Kyber进行了四并行加速. 该方案仅支持一组参数，消耗额外的BRAM资源减少了LUT的占用，但蝴蝶单元内部消耗始时钟周期数较多，导致总耗时增加. 与之相比，本方案在速度方面提升21.9%. Yaman[27]设计的蝴蝶单元仅包含一个模加、一个模减、一个模乘单元，通过片选单元和插入寄存器改变CT/GS架构的数据流向，可支持CT、GS两种架构. 其约减模块采用Zhang[42]的设计思路，针对Kyber模值3329重新设计了电路架构. 该方案对应点乘部分需要计算5次乘法，导致时钟周期数增加20%. Yaman分别设计了轻量型、均衡型、高性能型三种电路，分别对应单结构、四并行、十六并行. 本设计单结构、四并行硬件消耗分别减少了33.6%、25.8%，速度分别提升16.1%、7.3%. Mert对高纬度并行电路进行了研究[44]，随着复用蝴蝶单元的增多，整体时钟周期数明显减少，但优化设计呈现下降趋势. 此外，他采用HLS语言对多参数电路设计进行了实现[22]. HLS作为一种高层次硬件语言，在设计多参数电路时有着明显的灵活优势，但在时序设计上存在着不可避免的松弛度. 实验结果显示时钟周期数存在跨数量级增加，硬件资源消耗随着并行度增加急剧上升.

5 总 结

随着量子计算机的不断发展，后量子密码学得到了国际各组织机构的重点关注. 格基算法因其安全性和高效性的优点受到了各国学者的青睐，是下一代后量子密码标准的主流技术路线. 对于基于具有代数结构的格(如理想格或模格)的密码算法，一个基本且耗时的操作是多项式环上的元素乘法，采用NTT技术可降低计算复杂度，有效加速多项式计算. 根据多项式维度n和模值q满足条件不同，我们将其总结为三种NTT技术：SNTT、TNTT、HNTT，并进行了单结构、二并行、四并行可重构硬件优化实现. 针对其核心操作——蝴蝶计算，我们设计了紧凑型、低时延的电路结构，支持多种功能. 数据链路读写模式采用“交叉存储”方式，满足数据吞吐量的同时减少了存储资源的占用. 约减模块对Barrett算法进行修改，提出适用于多模值的硬件单元，使用加法器、移位器代替乘法器，减少了DSP资源的消耗. 实验结果表明，单结构在面积上最多减少62.6%，速度最大提升49.8%. 相较于单结构，优化后的二并行、四并行架构分别加速两倍、四倍，面积消耗仅为仅为其1.45倍、2.58倍. 本设计支持多种主流格基算法参数，如Kyber、Dilithium、Aigis、OSKR、OKAI等，可有效提升算法在硬件FPGA平台部署时的运行效率，为多算法协同设计提供一种解决方案.

参 考 文 献

[1] Shor P. W. Algorithms for quantum computation: discrete logarithms and factoring//Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, USA, 1994: 124-134

[2] NIST: Post-Quantum Cryptography Standardization, https://csrc. nist. gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization, 2023, 6, 29

[3] NIST: Post-Quantum Cryptography Round 3 Submissions, https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions 2023, 6, 29

[4] NIST: Post-Quantum Cryptography Selected Algorithms 2022 (2022), https://csrc.nist.gov/Projects/post-quantum-cryptography/

selected-algorithms-2022, 2023, 6, 29

[5] NIST: Post-Quantum Cryptography Round 4 Submissions (2022), https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions, 2023, 6, 29

[6] Chinese Association for Crypotologic Research, Public key algorithms selected to the second round competition of national cryptographic algorithm competitions, htmlhttps://sfjs.cacrnet.org.cn/site/term/list_77_1., 2019, 12, 28
(中国密码学会, 全国密码算法设计竞赛进入第二轮公钥算法, https://sfjs.cacrnet.org.cn/site/term/list_77_1.html)

[7] Chinese Association for Crypotologic Research, Announcement of the selection results of the national crypographic algorithm competitions, https://www. cacrnet. org. cn/site/content/854.html, 2020, 1, 2
(中国密码学会, 关于全国密码算法设计竞赛算法评选结果的公示, https://www.cacrnet.org.cn/site/content/854.html)

[8] Shen Shi-Yu, He Feng, Liang Zhi-Chuang, et al. OSKR/OKAI: Systematic Optimization of Key Encapsulation Mechanisms from Module Lattice, arXiv, 2021: 2109.02893

[9] Jin Zheng-Zhong, Zhao Yun-Lei. Optimal key consensus in presence of noise. IACR Cryptology ePrint Archive, 2017: 1058

[10] Jin Zheng-Zhong, Zhao Yun-Lei. Generic and Practical Key Establishment from Lattice//Proceedings of the Applied Cryptography and Network Security, Bogota, Colombia, 2019: 302-322

[11] Liang Zhi-Chuang, Shen, Shi-Yu, Shi Yuan-Tao, et al. Number Theoretic Transform: Generalization, Optimization, Concrete Analysis and Applications//Proceedings of the Information Security and Cryptology, Guangzhou, China, 2020: 415-432

[12] Shi Bai, DucasLéo, KiltzEike. CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation. NIST Round 3, https://csrc. nist. gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions, 2020, 10, 1

[13] Zhang Jiang, Yu Yu, Fan Shu-Qin, et al. Tweaking the Asymmetry of Asymmetric-Key Cryptography on Lattices: KEMs and Signatures of Smaller Sizes//Public-Key Cryptography, Edinburgh, UK, 2020: 37-65

[14] Roberto A., Joppe B., Léo D., et al. CRYSTALS-Kyber AlgorithmSpecifcations And SupportingDocumentation. NIST Round 1, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions, 2017, 11, 30

[15] AvanziRoberto, BosJoppe, DucasLéo, et al. CRYSTALS-Kyber Algorithm Specifcations And SupportingDocumentation (version 2.0), NIST Round 2, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions ,2019, 3, 30

[16] AvanziRoberto, BosJoppe, DucasLéo, et al. CRYSTALS-Kyber AlgorithmSpecifcations And SupportingDocumentation (version 3.0), NIST Round 3, https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions ,2020, 10, 1

[17] Moenck R. T. Practical fast polynomial multiplication// Proceedings of the third ACM symposium on Symbolic and Algebraic Computation, New York, USA, 1976: 136-148

[18] Mert A. C., Erdinç Ö., Erkay S. FPGA implementation of a run-time configurable NTT-based polynomial multiplication hardware, Microprocessors and Microsystems, 2020, 78:103219

[19] Ozcan E., Aysu A. High-Level Synthesis of Number-Theoretic Transform: A Case Study for Future Cryptosystems. IEEE Embedded Systems Letters, 2020, 12: 133-136

[20] Millar K., Łukowiak M., Radziszowski S. Design of a Flexible Schönhage-Strassen FFT Polynomial Multiplier with High-Level Synthesis to Accelerate HE in the Cloud// Proceedings of the International Conference on ReConFigurable Computing and FPGAs, Cancun, Mexico, 2019: 1-5

[21] Kawamura K., Yanagisawa M., Togawa N. A loop structure optimization targeting high-level synthesis of fast number theoretic transform//Proceedings of the 19th International Symposium on Quality Electronic Design, Santa Clara, USA, 2018: 106-111

[22] Mert A. C., Karabulut E., Öztürk E., et al. An Extensive Study of Flexible Design Methods for the Number Theoretic Transform, IEEE Transactions on Computers, 2022, 71(11): 2829-2843

[23] Beckwith L., Nguyen D. T., Gaj K. High-Performance Hardware Implementation of CRYSTALS-Dilithium// Proceedings of the 2021 International Conference on Field-Programmable Technology, Auckland, New Zealand, 2021: 1-10

[24] Beckwith L., et al. High-Performance Hardware Implementation of Lattice-Based Digital Signatures, IACR Cryptology ePrint Archive, 2022: 217

[25] Zhao C., Zhang N., Wang H. A Compact and High-Performance Hardware Architecture for CRYSTALS-Dilithium, IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022, 1: 270-295

[26] Mert A. C., Karabulut E., Öztürk E., et al. A Flexible and Scalable NTT Hardware: Applications from Homomorphically Encrypted Deep Learning to Post-Quantum Cryptography// Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 2020: 346-351

[27] Yaman F., Mert A. C., Öztürk E., et al. A Hardware Accelerator for Polynomial Multiplication Operation of CRYSTALS-KYBER PQC Scheme// Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 2021: 1020-1025

[28] Xing Yu-Fei, Li Shu-Guo. A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA, IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2: 328-356

[29] Mojtaba B., Reza A., Mehran M. High-Speed NTT-based Polynomial Multiplication Accelerator for CRYSTALS-Kyber Post-Quantum Cryptography, IACR Cryptology ePrint Archive, 2021: 563

[30] Montgomery P. L. Modular multiplication without trial

division, Mathematics of computation, 1985, 44 (170): 519-521
[31] Barrett P. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor//Proceedings of the Advances in Cryptology-CRYPTO' 86, Montreal, Canada, 1986: 311-326
[32] Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process, https://doi.org/10.6028/NIST.IR.8413-upd1 2022, 7, 5
[33] Weimerskirch A., Paar C. Generalizations of the karatsuba algorithm for efficient implementations, IACR Cryptology ePrint Archive, 2006:224
[34] Pöppelmann T., Güneysu T. Towards Efficient Arithmetic for Lattice-Based Cryptography on Reconfigurable Hardware// Progress in Cryptology-LATINCRYPT 2012, Santiago, Chile, 2012: 139-158
[35] Liang Zhi-Chuang, Zhao Yun-Lei. Number Theoretic Transform and Its Applications in Lattice-based Cryptosystems: A Survey, arXiv, 2022: 2211.13546
[36] Seiler G. Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography. IACR Cryptology ePrint Archive, 2018:39
[37] Bernstein D. J. Multidigit multiplication for mathematicians, https://cr.yp.to/papers/m3-20010811-retypeset-20220327.pdf 2001, 8, 11
[38] Cooley J. W., Tukey J. W. An algorithm for the machine calculation of complex Fourier series. Mathematics of computation, 1965, 19: 297-301
[39] Gentleman W. M., Sande G. Fast Fourier Transforms: for fun and profit//Proceedings of the Fall Joint Computer Conference, San Francisco, California, USA, 1966: 563-578
[40] Zhou Shuai, Xue Hai-Yang, Zhang Dao-De, et al. Preprocess-then-NTT Technique and Its Applications to KYBER and NEWHOPE//Proceedings of the Information Security and Cryptology, Nanjing, China, 2018: 117-137
[41] Zhu Yi-Ming, Liu Zhen, Pan Yan-Bin. When NTT Meets Karatsuba: Preprocess-then-NTT Technique Revisited// Proceedings of the Information and Communications Security, Chongqing, China, 2021: 249-264
[42] Zhang N., Yang B., Chen C., et al. Highly Efficient Architecture of NewHope-NIST on FPGA using Low-Complexity NTT/INTT, IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2: 49 - 72
[43] Groginsky H. L, Works G. A. A pipeline fast fourier transform, IEEE Transactions on Computers, 1970, 100(11): 1015-1019
[44] Derya K., Mert A. C., Öztürk E., Savas E.. CoHA-NTT: A Configurable Hardware Accelerator for NTT-based Polynomial Multiplication, IACR Cryptology ePrint Archive, 2021: 1527

ZHAO Xu-Yang, Ph. D. candidate. His research interests include post-quantum cryptography, cryptographic engineering and hardware/software co-design.

LIANG Zhi-Chuang, Ph. D. candidate. His research interests include post-quantum cryptography.

HU Yue, Ph. D. candidate. His research interests include post-quantum cryptography and cryptographic engineering.

GENG He-Xiang, M. S. candidate. His research interests include post-quantum cryptography and cryptographic engineering.

ZHAO Yun-Lei, Ph. D., professor. His research interests include post-quantum cryptography, cryptographic protocols and computational theory.

Background

This research belongs to the post-quantum cryptography and cryptographic engineering area. Traditional public key cryptography, such as RSA and ECC, rely on large integer factoring and discrete logarithm problems, which will be cracked in polynomial time by a practical quantum computer running the Shor algorithm. The quantum computers developed in the world today are dedicated computers, which have no obvious advantages over traditional electronic computers in the processing of general tasks. So, the traditional public key cryptography algorithm remains safe for now. But it can be predicted that a practical universal quantum computer will be built in the near future. IBM and Microsoft engineers expect large-scale quantum computers to emerge in the next 15-20 years. In particular, Google completed a 53-qubit computer named Sycamore in 2019, which can accomplish the computation of the most powerful supercomputer for about 10 000 years in 200 seconds. IBM has completed a 433 qubit computer named Osprey in 2022, which is eight times faster than Sycamore. It is urgent to research and design quantum-resistant public key cryptography algorithms and protocols. Among many post-quantum cryptographic schemes, lattice-based cryptography has attracted extensive attention due to its security, efficiency and other advantages.

Polynomial multiplication over the ring is the most complex and time-consuming operation in lattice-based cryptography. Normally, most schemes use number theoretic transform (NTT) technology to reduce computational complexity and accelerate polynomial multiplication. In this paper, we studied NTT technologies used in the mainstream lattice-based cryptographic schemes and typed them as Standard-NTT (SNTT), Truncated-NTT (TNTT) and Hybrid-NTT (HNTT) according to different parameters. Then, we designed a compact hardware structure that can be executed in constant time and supports multiple parameters using the Verilog HDL language. It was implemented and verified on an Artix-7 series FPGA chip. Our design can complete the whole polynomial multiplication process, including forward NTT, point-wise multiplication, and inverse NTT. The hardware design includes a compact butterfly unit with low time delay, a reduction unit suitable for multiple moduli and a unique cross-storage memory access mode. Compared to the state-of-art, this design reduces area consumption by 12%-62.6%, and speeds up by 5.5%-49.8%. We also designed a high-throughput parallel framework. Compared to the single structure, the optimized two-parallel and four-parallel architectures speed up twice and four times respectively, with only 1.45 and 2.58 times area consumption.