格基后量子密码的可重构NTT运算单元与高效调度算法研究

付秋兴,李 伟*,别梦妮,陈 韬,南龙梅
(信息工程大学,河南郑州 450001)

摘 要: 为进一步提高格基后量子密码算法中多项式乘法的运算速率,同时考虑到不同格基密码中多项式乘法参数各异的现状,本文提出了一种面向高速的可重构数论变换(Number Theoretic Transforms,NTT)运算单元,并提出了相应的数据调度方案解决时序冲突和空间冲突问题. 本文首先分析了不同格基后量子密码算法中NTT算法的运算特征,提出一款4×4的可重构运算单元,满足不同位宽的基2/3/4-NTT运算需求. 其次,基于上述硬件设计提出了一种针对基4-NTT算法的数据调度方案,解决了高并行多流水级设计下的时序冲突问题. 最后,提出了基于m-着色算法的多Bank数据存储方案,解决数据访问冲突的问题. 实验结果表明,本文设计的硬件结构具备实现基2/3/4-NTT及其逆运算功能,能够支持Kyber、Dilithium在内的多种格基后量子密码算法,硬件支持最大并行度为4. 为进一步验证本文硬件设计的优越性,在Xilinx Virtex-7器件上进行实验验证,工作频率达169 MHz,可在0.40 μs内完成NTT算法功能,ATP降低约42%;在40 nm CMOS工艺节点进行综合实现,与现有的设计相比,本文的硬件设计AT积降低18%~90%.

关键词: 后量子密码;NTT;可重构;并行化;高速
基金项目: 高层次创新人才工程项目(No.6843255642JZ2301LZ)
中图分类号: TN402;TP309 **文献标识码:** A **文章编号:** 0372-2112(2025)04-1182-10
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240788

Research on Reconfigurable NTT Arithmetic Unit and Efficient Scheduling Algorithm for Lattice Post-Quantum Cryptography

FU Qiu-xing, LI Wei*, BIE Meng-ni, CHEN Tao, NAN Long-mei
(*University of Information Engineering*, *Zhengzhou*, *Henan* 450001, *China*)

Abstract: In order to further improve the rate of polynomial multiplication in lattices post-quantum cryptography, and considering the different parameters of polynomial multiplication in different lattices, a high-speed reconfigurable number theory transformation (NTT) arithmetic unit is proposed in this paper, and the corresponding data scheduling scheme is proposed to solve the problem of time sequence conflict and space conflict. In this paper, we first analyze the operation characteristics of NTT algorithm in different lattice-based post-quantum cryptography algorithms, and propose a 4×4 reconfigurable operating unit to meet the needs of 2/3/4-NTT operation in different bit widths. Secondly, based on the above hardware design, a data scheduling scheme based on the basic 4-NTT algorithm is proposed to solve the timing conflict problem in the highly parallel multi-pipeline-level design. Finally, a multi-bank data storage scheme based on m-coloring algorithm is proposed to solve the problem of data access conflict. Experimental results show that the hardware structure designed in this paper is capable of implementing base 2/3/4-NTT and its inverse operation functions, and can support a variety of latry-based post-quantum cryptography algorithms including Kyber and Dilithium. The maximum parallelism degree supported by the hardware is 4. In order to further verify the superiority of the hardware design in this paper, Xilinx Virtex-7 device is used for experimental verification. The working frequency is up to 169 MHz, and the NTT algorithm function can be completed within 0.40 μs, and ATP is reduced by about 42%. Integrated implementation on 40 nm CMOS process nodes results in a 18%~90% reduction in the AT volume of the hardware design compared with existing designs.

收稿日期:2024-08-28;修回日期:2025-03-10;责任编辑:孙瑶
*通讯作者:李伟

Key words: post-quantum cryptography; number theoretic transforms; reconfigurable; parallelization; high speed
Foundation Item(s): High-Level Innovative Talent Project (No.6843255642JZ2301LZ)

1 引言

为应对量子计算机带来的安全威胁,后量子密码已成为信息安全领域的新兴研究方向. 美国国家标准技术研究所(National Institute of Standardsand Technology, NIST)在2012年正式启动相关研究工作,并于2022年7月公布了首批进入标准化阶段的后量子密码算法. 由于格基密码具有最坏情况到平均情况的归约且能同时兼顾公钥加密和数字签名设计的优势,从后量子密码算法征集到标准公布过程中,格基后量子密码算法一直占据较大的比重. 相比于传统密码算法,格基后量子密码数据运算量大,硬件实现复杂,运算时间更长,其中,多项式乘法操作消耗的硬件资源大,运算的时间最长. 通过数论变换(Number Theoretic Transform,NTT)实现多项式乘法具有更低的运算复杂度,为提高硬件部署后量子密码算法时的整体速率和适应性,设计一种能支持不同的格基后量子密码的NTT算法运算单元具有重要意义.

不同的格基后量子密码算法之间参数有较大差异,为提高硬件设计的适应性,NTT单元的重构设计一直是研究的热点问题. 华中科技大学的刘冬生团队[1]针对不同格基密码算法中数论变换参数的多样性,提出一种基于随机存取存储器(Random Access Memory, RAM)的可重构多通道数论变换单元,支持256~2 048的不同参数n和13~32位模q的可重构配置,工作频率最高可达232 MHz,但消耗的内存量高达32×4×28 bit.

目前,高速实现NTT算法的硬件设计主要分为两种:(1)采用流水线结构,即每个时钟周期完成一个NTT运算步骤,通过提高运算频率以减少硬件完成整个算法的时延时间;(2)并行化设计,从以面积换时间的角度对运算单元展开并行化设计,减少NTT运算的周期数.

在针对NTT算法硬件实现的流水线结构研究中,文献[2]针对Saber算法,对固定参数的模乘单元进行分析,基于此设计了一款具有五级流水结构的蝶形单元,在性能方面表现出色. 文献[3]提出了一种用于多项式系数乘法的三级流水线Karatsuba乘法器对NTT蝶形单元中的乘法运算进行拆分,缩短了关键路径延时,针对特定参数优化的单元设计在性能方面具有明显的优势,但不适用于多参数的重构设计. 文献[4]在流水级的划分提出了另一种思路,根据NTT运算网络具备层次性的特点,串行放置多个蝶形单元BFU,每个单元承担网络中一个层级的运算任务,设计并在上述结构中插入了基于BRAM的重排序单元,确保运算顺序正确,多个BFU整体形成流水结构,各个BFU的控制相互独立. 由于NTT运算网络前后数据的相关性,该结构的硬件资源利用率不够高,BFU具有不同程度的闲置时间. 并行化设计普遍应用在高速需求的场景下,Xin等人[5]设计了一款面向后量子密码算法的向量处理器VPQC,在RISC-V架构下对NTT运算单元进行了矢量化设计,提高了整体吞吐率. 文献[6]提出了一种面向任意长度和模数的NTT运算单元的设计方法,可以根据需求设置不同的并行度生成对应的NTT硬件加速器,同时给出了存储访问无冲突的形式化证明,但未给出统一的硬件架构.

在实现NTT硬件设计时,需要综合考虑单元的重构功能、面积开销以及硬件利用效率等多个问题. 为了实现一种支持多参数的NTT硬件单元,具有相对较低的硬件开销,需要分析现有的后量子密码算法的运算特点并对硬件进行细粒度的重构设计;为了提高整体的吞吐率,需要利用流水线结构和并行设计,但NTT运算网络中不同stage之间数据存在关联性,这导致硬件单元在流水线深度和并行度之间存在限制关系,传统NTT算法的数据调用顺序会导致流水中断等问题,严重限制了硬件单元的整体效率,因此,迫切需要一套高效的数据调度方案和存储策略以满足NTT单元的运算数据需求.

基于上述问题,本文面向格基后量子密码算法中NTT单元的设计和数据调度展开研究,主要工作包括以下三点:(1)针对格基后量子密码算法中NTT运算的参数进行细致分析,提出了一款支持多位宽混合基的可重构NTT运算单元,支持基2/3/4-NTT算法及其逆运算;(2)基于上述硬件设计,提出了一种避免流水中断的数据调度方案;(3)针对NTT算法的数据存储和调用特点,提出了一种基于m-着色算法的数据存储方案,并基于该方案设计了一款解决数据访问冲突的多Bank存储单元.

2 算法背景

2.1 NTT算法介绍

与快速傅里叶变换的思想相同,快速数论变换通过特殊点带入求值将多项式从系数表示转换为点值表示. 大多数格基后量子密码算法中多项式乘法在多项式环$Z_q[x]/(x^n+1)$上进行,n为2的幂次,q为素数. 当在多项式环$Z_q[x]/(x^n-1)$上通过NTT算法实现多项式乘法时,有:$a\times b=\mathrm{INTT}_n\left(\mathrm{NTT}_n(a)\odot\mathrm{NTT}_n(b)\right)$,其中,

a、b 为环上多项式. 通过NTT算法在多项式环 $Z_q[x]/(x^n+1)$ 上对多项式乘法进行加速时首先要进行补零操作:对多项式系数向量 $\boldsymbol{a}=\{a_0,a_1,\cdots,a_{n-1}\}$、$\boldsymbol{b}=\{b_0,b_1,\cdots,b_{n-1}\}$进行补零后可得$\tilde{\boldsymbol{a}}=\{a_0,a_1,a_2,\cdots,a_{n-1},0,\cdots,0\}$、$\tilde{\boldsymbol{b}}=\{b_0,b_1,b_2,\cdots,b_{n-1},0,\cdots,0\}$,而后进行NTT和INTT操作:$\tilde{\boldsymbol{a}}\times\tilde{\boldsymbol{b}}=\mathrm{INTT}_{2n}\left(\mathrm{NTT}_{2n}(\tilde{\boldsymbol{a}})\odot\mathrm{NTT}_{2n}(\tilde{\boldsymbol{b}})\right)$.

该方案会导致NTT处理规模翻倍的问题,为避免该情况发生,可以通过使用负包裹卷积(NWC)消除补零操作,首先需要对数据进行预处理 $\bar{\boldsymbol{a}}=\{a_0,ga_1,\cdots,g^{n-1}a_{n-1}\}$,$\bar{\boldsymbol{b}}=\{b_0,gb_1,\cdots,g^{n-1}b_{n-1}\}$,而后直接使用NTT和INTT算法进行:$\bar{\boldsymbol{c}}=\bar{\boldsymbol{a}}\times\bar{\boldsymbol{b}}=\mathrm{INTT}_n\left(\mathrm{NTT}_n(\bar{\boldsymbol{a}})\odot\mathrm{NTT}_n(\bar{\boldsymbol{b}})\right)$. 其中 g 为 $2n$ 次单位根,最后将 $\bar{\boldsymbol{c}}$ 中元素乘以 $g^{-i}w^{-i}$ 即可得出多项式乘法的最终结果. 可通过改变NTT算法中旋转因子的调用顺序避免上述预处理和后处理以减少运算复杂度,具体NTT算法如算法1所示,其中,w 为 $2n$ 次单位根,即 $w^{2n}=1\bmod q$,且 $w^n=-1\bmod q$.

算法1 NTT算法

$a=a_0+a_1x+\cdots+a_{n-1}x^{n-1}\in Z_q[x]/(x^n+1)$

预计算:$w_k=w^{brv(k)}$

输入:$\boldsymbol{a}=\{a_0,a_1,\cdots,a_{n-1}\}$

输出:$\boldsymbol{a}=\{a_0,a_1,\cdots,a_{n-1}\}$

```
k=1
for l=n/2; l>0; l=l/2 do
  for s=0; s<n; s=j+1 do
    for j=s; j<s; j=j+1 do
      t=w_k·a_{j+l}
      a_{j+l}=a_j−t
      a_j=a_j+t
    end for
    k=k+1
  end for
end for
```

2.2 算法需求分析

目前我国后量子密码标准尚未公布,针对某个具体后量子密码算法的NTT运算单元进行硬件实现具有较大的局限性,应对未来后量子密码算法种类和安全等级更加多样化的情况,NTT单元应当立足重构设计. 为使得所设计的硬件单元具有更好的适应性,本文设计的硬件单元在支持实现NIST公布的后量子密码算法标准中格基后量子密码算法的同时,支持实现参与后量子密码标准筛选的NTRU、Saber和Falcon等算法. 表1列举了上述格基后量子密码算法中多项式乘法的项数 n、模数 q 和模多项式 $f(x)$ 三个运算特征.

现有的格基后量子密码算法中,kyber、Dilithium、Saber以及Aigis等算法中多项式项数均为 n=256,部分算法需求 n 为512或1 024. 其中,出现次数最多的项数为256,且Kyber、Dilithium已通过遴选并成为标准算法,具有较高的参考价值. NewHope等算法也更推荐使用项数 n=1 024的密码方案,上述项数均为4的幂次;NTRU算法在参数扩展后[7],项数分解后包含2、3、4的幂次,因此,所设计的单元需要立足实现基2、基3和基4三种蝶形运算. 在模数 q 的重构设计中,除Dilithium算法数据在16~32 bit的范围内,其他多数后量子密码算法的模数在16 bit以下.

表1 格基后量子密码多项式乘法参数分析

算法	n	q	$f(x)$
Kyber	256	3 329	x^n+1
Saber	256	2^{13}	x^n+1
Dilithium	256	8 380 417	x^n+1
Falcon	512/1 024	12 289	x^n+1
LAC	512/1 024	251	x^n+1
NewHope	512/1 024	11 289	x^n+1
NTRU	509/677/821/701	2 048/4 096/8 192	$(x^n-1)/(x-1)x^n-1$
Aigis(国内)	256/512	7 681/11 289	x^n+1

大多数格基后量子密码算法中多项式乘法在多项式环 $Z_q[x]/(x^n+1)$ 上进行,其中 n 为2的幂次,q 为素数,NTRU等算法则在 $Z_q[x]/(x^n-1)$ 上进行,上述两个多项式环上的NTT算法仅在旋转因子的生成和调用顺序上存在不同,对运算单元的设计不产生影响. 基于上述分析,对NTT单元的重构需求体现在基数和模数两个方面,单元需要具备支持基数为2/3/4,模数在32 bit内的NTT蝶形运算的功能.

3 可重构NTT硬件单元研究与设计

在位宽的重构方面,相比于逻辑中的模加、减单元,模乘单元的设计复杂,路径延时高,硬件开销更大,本文首先对模乘单元进行分析和设计. 在目前针对NTT的模乘单元实现方案中,常用的有三种:LN模乘、Montgomery模乘以及Barrett模乘. 其中,LN模乘算法对模数具有特定限制,要求模数 $q=k\cdot2^m+1$,k 为奇数,且 $2^m>k$;Montgomery模乘算法对模数和位数的限制较小,但需要对数据进行蒙域上的变换;相比于Montgomery模乘算法,硬件实现Barrett模乘算法时不需要额外的预处理和后处理电路,虽然多1次乘法运算,但Barrett模乘更适应可重构NTT单元的设计.

在使用Barrett模乘实现模乘操作/NTT单元的研究当中[5,8],一般是通过固定Barrett模乘的最大位宽(如32 bit),直接支持32 bit以下所有的模乘运算. 这种设计导致的问题是单元在低位宽(如Kyber算法12 bit位

宽)的应用时,80%以上的乘法资源处于闲置状态.

算法2　16 bit模乘Barrett算法

输入:$x,y\in Z_q,k=2\cdot\lceil\log(q)\rceil,m=\lfloor 2^k/q\rfloor$

输出:$z=x\cdot y \bmod q$

1. $z=x\times y$
2. $z_1=z[31:16],z_2=z[15:0]$
3. $t=(z_1\cdot m<<16+z_2\cdot m)>>k$
4. $z=z-(t\cdot q)$
5. if $z\geqslant q$ then
6. $z\geqslant q$
7. end if
8. return z

在第2节的需求分析中,本文设计的NTT单元需要支持模数在0~32 bit的运算,大多后量子密码算法模数在16 bit以内,为避免上述乘法资源闲置情况的发生,所设计的单元需要能够并行实现16 bit内的NTT算法,通过单元连接能实现17~32 bit的NTT运算,如算法2所示.考虑到4组16 bit乘法通过移位和加法完成32 bit乘法操作,基于Barrett模乘所设计的PE单元在位宽的扩展上具有好的适应性.基于上述分析,本文以16 bit乘法为基本运算单元,对16 bit模乘单元进行重构设计,通过配置可以完成32 bit乘法操作,为后续单元实现32 bit内算法核心运算提供硬件支撑.图1中展示了32 bit模乘运算的硬件实现方案,左图中32 bit乘法通过16 bit模乘单元通过功能上的重构实现,右图为重构单元的具体硬件结构,包含4组16 bit乘法和移位以及部分加法单元.在16 bit的NTT运算中,重构单元将作为独立的模乘模块应用在蝶形单元中.

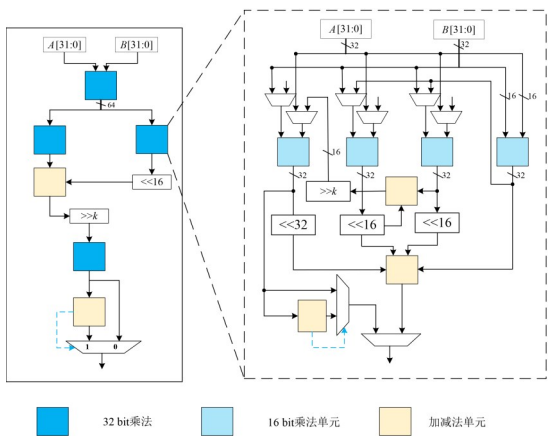

图1　32 bit Barrett模乘算法实现

通过第2节的需求分析,为了支持不同基数的后量子密码算法,可重构NTT的单元设计还需要对基2/3/4-NTT的核心运算进行拆解,提取公共逻辑部分来进行紧凑硬件设计,如算法3~算法5所示.

算法3　基2-NTT算法核心运算	**提取基2-NTT算法核心运算逻辑**
$a[k]=A[x]+w^kB[x]$ $a[k+n/2]=A[x]-w^kB[x]$	$\left.\begin{aligned}T_0&=A[x]+B[x]w^k\\T_1&=A[x]-B[x]w^k\end{aligned}\right\}u\pm vw$
算法4　基3-NTT算法核心运算	**提取基3-NTT算法核心运算逻辑**
$a[k]=A[x]+w^kB[x]+w^{2k}C[x]$ $a[k+n/3]=A[x]+\mu w^kB[x]+\mu^2w^{2k}C[x]$ $=A[x]-w^kB[x]-w^{2k}C[x]-(\mu^2w^kB[x]+\mu w^{2k}C[x])$ $a[k+2n/3]=A[x]+\mu^2w^kB[x]+\mu w^{2k}C[x]$	$\left.\begin{aligned}T_0&=A[x]+B[x]w^k\\T_1&=A[x]-B[x]w^k\end{aligned}\right\}u\pm vw$ $T_2=B[x]\mu^2w^k+C[x]\mu w^{2k}\}uw_1+vw_2$ $\left.\begin{aligned}a[j]&=T_0+C[x]w^{2k}\\T_3&=T_1-C[x]w^{2k}\end{aligned}\right\}u\pm vw$ $a[j+l]=T_3-T_2$ $a[j+2l]=A[x]+T_2$
算法5　基4-NTT算法核心运算	**提取基4-NTT算法核心运算逻辑**
$a[k]=A[x]+w^kB[x]+w^{2k}C[x]+w^{3k}D[x]$ $a[k+n/4]=A[x]+\mu w^kB[x]+(\mu w^k)^2C[x]+(\mu w^k)^3D[x]$ $=A[x]-w^{2k}C[x]+\mu(w^kB[x]-w^{3k}D[x])$ $a[k+2n/4]=A[x]+\mu^2w^kB[x]+(\mu^2w^k)^2C[x]+(\mu^2w^k)^3D[x]$ $=A[x]+w^{2k}C[x]-(w^kB[x]+w^{3k}D[x])$ $a[k+3n/4]=A[x]+\mu^3w^kB[x]+(\mu^3w^k)^2C[x]+(\mu^3w^k)^3D[x]$ $=A[x]-w^{2k}C[x]-\mu(w^kB[x]+w^{3k}D[x])$	$\left.\begin{aligned}T_0&=A[x]+C[x]w^k\\T_1&=A[x]-C[x]w^k\end{aligned}\right\}u\pm vw$ $\left.\begin{aligned}T_2&=B[x]w^k+D[x]w^{3k}\\T_3&=B[x]w^k-D[x]w^{3k}\end{aligned}\right\}uw_1\pm vw_2$ $a[j]=T_0+T_2$ $a[j+2l]=T_0-T_2$ $\left.\begin{aligned}a[j+l]&=T_1+T_3\mu\\a[j+3l]&=T_1-T_3\mu\end{aligned}\right\}u\pm vw$

通过对算法的核心运算进行逻辑提取,能够拆解出两种基本的运算逻辑形式:$u\pm vw$、$uw_1\pm vw_2$,配合少量的加减运算逻辑,可以支持完成上述3种核心运算.对上述逻辑形式进行如$u\pm vw\rightarrow u+v$、$(u-v)w$形式的简单变形,配合加减运算单元和模除运算逻辑可以支持相应的INTT核心运算.

在乘法单元设计的基础上，对上述分析出的基本运算逻辑形式 $u+vw$、$uw_1 \pm vw_2$ 进行硬件实现．流水线设计是提高硬件资源利用率和提升计算效率的常用设计方法．通过在硬件单元中合理位置插入寄存器，能够有效减小关键路径延时．图2展示了对逻辑 $u \pm vw$ 以及逻辑 $u+v$、$(u-v)w$ 的PE实现方案，通过Barrett算法实现16位模乘，共调用了4个16位乘法运算，其中第2个和第3个乘法运算完全并行，为满足高速的设计需求，本文将关键路径限制在16位乘法运算中，因此，该PE单元至少花费3个周期进行运算，PE单元内除模乘运算外，还包含模加减、模除等运算，若与16位乘法在同一流水级内执行则该路径会成为关键路径，因此该单元设计划分为4个流水级，关键路径为16位乘法和一次加法，蓝框为16位模乘单元，红色虚线框为NTT和INTT共用的模加减单元，在NTT和INTT的运算中，模加减单元的运算位置会发生变化，为支持INTT运算，加入了模除单元．

上述硬件设计同样适应于逻辑功能 uw_1+vw_2，对

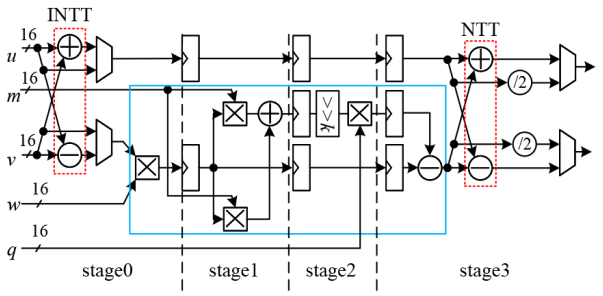

图2 $u \pm vw$ 逻辑及其逆变换单元设计

算法2～算法4提取出的运算逻辑分别进行实现，最终设计出图3所示2×2可重构蝶形运算单元，包含4个模乘单元、9个模加/减单元，通过2×2可重构蝶形运算单元内的PE互联以及PE内的配置，配合少量模除单元，该单元具备实现16 bit内基2/3/4-NTT及其INTT的运算功能．图3①~③分别为实现基2\3\4-NTT核心运算时单元使用和连接情况，图3④为4×4可重构运算单元，图3⑤为17~32 bit基4-NTT核心运算的单元启用情况．

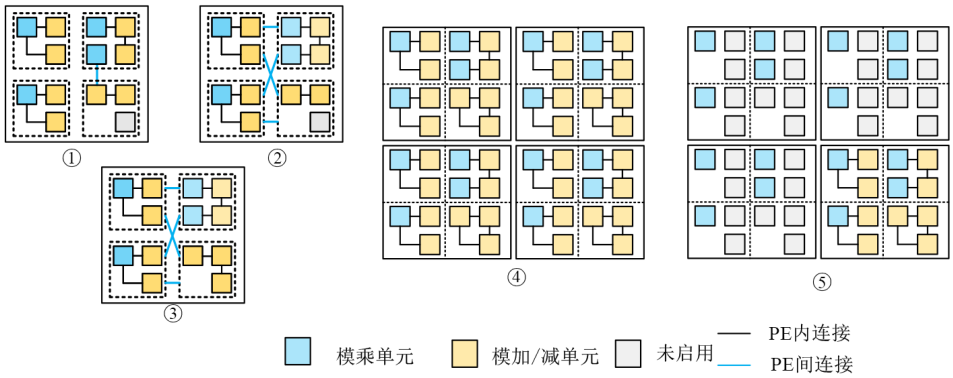

图3 2×2可重构蝶形运算单元

根据图1所示32 bit模乘算法实现方案，需要4组32 bit乘法（16 bit模乘）单元以及加法、移位单元实现其32 bit模乘功能，因此需要将上述2×2可重构蝶形运算单元扩展至图3④所示4×4可重构蝶形运算单元，包含16个16 bit模乘单元、36个模加减单元以及少量加法、移位和模除单元，整体可以实现17～32 bit的基2/3/4-NTT及INTT的核心运算，如图3⑤所示在16 bit位宽运算向32 bit位宽运算扩展时，该设计方案的乘法单元均不发生闲置情况，达到了较高的资源利用率．4×4可重构核心运算单元整体形成8级流水结构，支持16 bit基4-NTT运算的最大并行度为4．为了达到高吞吐率的设计目标，尽可能地提升单位面积的利用率，在调用该单元实现基4-NTT运算时会使用最大的并行度，因此，NTT运算数据的调度和存储需要在每个时钟周期向单元内输入4组正确的运算数据（4×4×16 bit），保证硬件整体的工作效率．

4 基于可重构蝶形单元的数据调度方案

基于第2节设计的4×4可重构蝶形单元，在进行基4-NTT运算时，单元整体形成8级流水结构，在最大并行度条件下对项数为256的多项式进行基4-NTT变换．正如前文所述，NTT运算网络中不同stage之间数据存在关联性，按照传统NTT算法的三层循环结构对数据进行调度，则会出现如图4所示的时序冲突问题，即stage-1数据还未更新进存储器，而stage-2需要调用该数据．上述时序冲突所涉及数据约占运算总量的1/8，若通过中断流水线来解决上述问题需要暂停8个时钟周期，占运算总时钟周期1/9，且需要额外地判断电路对冲突数据进行检测并控制流水中断．为达到高速设计的目的，同时避免增加判断、控制电路，需要设计满足无时序冲突的数据调度方案．在第3节硬件设计的基础上，要求数据调度方案满足在8级流水线内不发生时序冲突，为了省去存储器读写时对WAR冲突的判断电路，保证每组数据输入

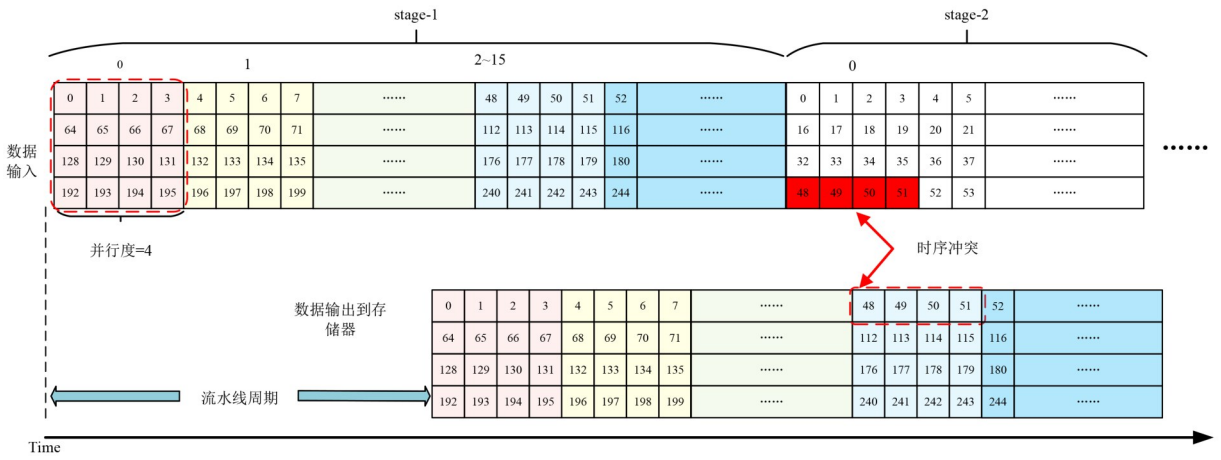

图4　时序冲突问题

后在9个时钟周期内不再访问该数据. 为达到上述目的,需要首先从存储器的设计入手,对数据的存储方案进行分析.

在进行16 bit基4-NTT运算时,每个周期需要从数据存储器中读取16组16 bit数据,一般的RAM存储器难以满足需求,需要进行特殊设计. 数据存储器的设计应当遵循以下两个原则:(1)访问数据干净、完整,单次数据访问能够获取每次运算所需的全部数据且尽可能不存在数据冗余;(2)逻辑电路简洁,地址生成逻辑简单、生成地址数尽可能少以便控制逻辑实现. 因此,存储器的位宽选择十分重要,128 bit、256 bit或更大的位宽选择会导致数据访问存在大量冗余,若将位宽设置为16 bit,每次需要生成16个读写地址对数据进行访问,对地址生成电路和存储器的设计是十分不利的. 因此本文将存储器数据位宽设置为64 bit,通过控制器生成4组地址对存储器进行访问,满足单元的运算需求.

在项数 $n=256$ 的基4-NTT运算网络中,共有 $\log(n)=4$ 个stage,在4个stage中,每组运算输入数据编号之间步长分别为64、16、4、1. 在上述存储器设计中,对于输入多项式的系数向量 $\boldsymbol{a}=\{a_0,a_1,\cdots,a_{n-1}\}$,元素 a_i 为16 bit数据,存储时按照顺序进行拼接和存储,即存储器存储数据 $\text{ram\_data}=\{a_i,a_{i+1},a_{i+2},a_{i+3}\}$. 每个时钟通过4个访存地址从存储器中读取运算所需的16组16 bit数据. 对于上述提出数据输入后在9个时钟周期内不发生时序冲突要求,按照上述存储方案,图5展示出从stage-4的数据输入情况,推理其他stage必须满足的数据输入条件.

图5左图展示了stage间的推理过程,在stage-3中,每个单元的输入数据编号之间step=4,因此需要从存储器访问4组数据 $\{a_i,a_{i+1},a_{i+2},a_{i+3}\}$, $\{a_{i+4},a_{i+5},a_{i+6},a_{i+7}\}$, $\{a_{i+8},a_{i+9},a_{i+10},a_{i+11}\}$, $\{a_{i+12},a_{i+13},a_{i+14},a_{i+15}\}$ 并进行简单的转换逻辑才能进行stage-3的运算,而在stage-4中,单次基4-NTT蝶形运算的输入数据编号之间步长为1,为一个访问地址存储的4组连续数据 $\{a_i,a_{i+1},a_{i+2},a_{i+3}\}$,stage-3中任意的计算顺序均不影响stage-4启动运算. 当stage-3在进行运算时,以运算数据 $\{a_0,a_1,a_2,\cdots,a_{15}\}$ 为例,stage-2单元输入数据编号之间步长为16,根据9个周期无时序冲突的需求以及每个stage需要消耗16个cycle,完成所有数据的输入,推理出stage-2必须在开始运算的前7个cycle完成对包含数据 $\{a_0,a_1,a_2,\cdots,a_{15}\}$ 在内的输入,即完成 $\{a_0,a_1,a_2,\cdots,a_{63}\}$ 的输入. 同理,推理出stage-1在前14个cycle需要完成 $\{a_0,a_1,a_2,\cdots,a_{255}\}$ 所有数据的输入,与前提条件相冲突.

上述问题的根本原因在于按照系数顺序拼接存储,数据之间的绑定关系,在由stage-2推理stage-1时出现数据关联限制,该限制体现在算法调度顺序和数据存储方式两个方面,在存储时,不能按照多项式系数顺序进行简单的拼接,存储方案应当使数据在访存和计算时具备更强的灵活性才能够避免上述问题的发生.

考虑到NTT运算包含的4个stage中的单元输入特点,存储的数据之间编号步长应当尽可能从1、4、16、64中选择,其中,数据之间编号步长设置为1或64均会发生上述问题,步长设置4时,图5右图展示了由stage-4最终推理到stage-1需要在21个cycle内完成数据输入,且stage-2、stage-3同样能够满足启动条件,步长选择4或16效果相同,本文选择存储数据编号步长为4的存储方案. 据此,存储数据按照 $\text{ram\_data}=\{a_i,a_{i+4},a_{i+8},a_{i+12}\}$ 的形式进行拼接和存储,其中 $i\%16\leqslant 3$,在该存储方案下,能够搜索出满足无时序冲突的数据调度方案.

在上述存储方案设计以及stage间的数据限制关系的基础上,本文设计数据调度方案如图6所示:stage-1、stage-2以及stage-4的数据调度具有较高的相似度,均采用双层for循环嵌套实现,最大限度地降低

控制器的实现成本，stage-3采用阶梯型的调用顺序进行调度，在实现时分为前4层运算和后3层嵌套运算．在数据输入阶段，除stage-3外，从存储器中读取到的16×16 bit数据均需要进行串并转换后输入到运算单元中，stage-3根据存储器读取顺序直接将数据输入至运算单元．

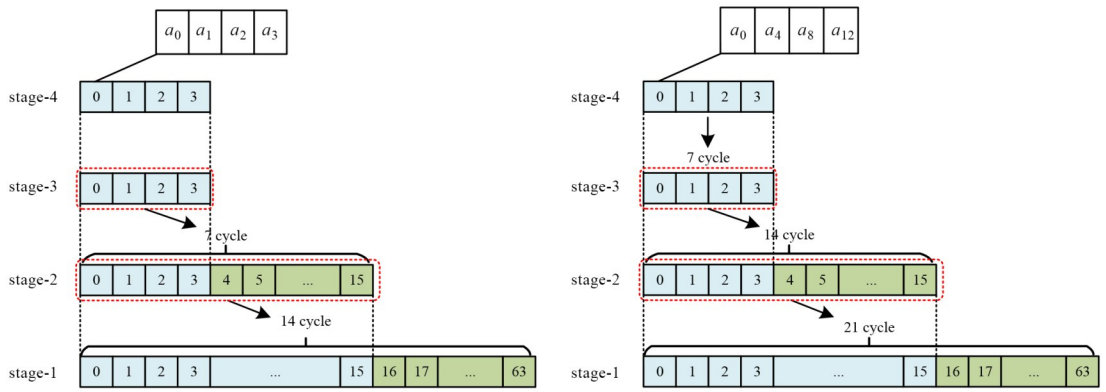

图5 不同数据拼接存储方案

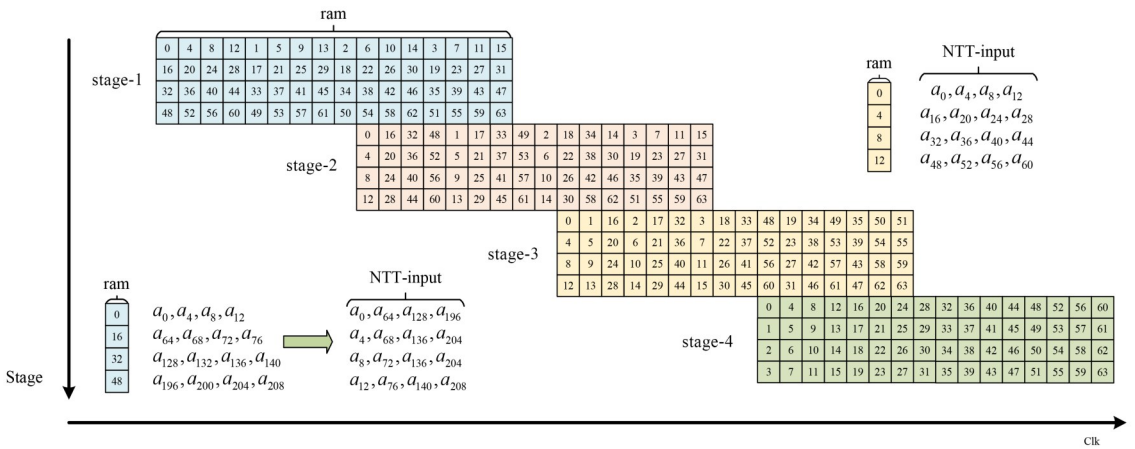

图6 避免流水中断的数据调度方案

相较于传统并行NTT的3层for循环嵌套结构，本文设计的调度方案可以最大限度地支持深度流水的单元设计进行无流水中断的运算，同时，stage-1、stage-2和stage-4的控制逻辑结构相同，在控制器实现时仅需要额外增加stage-3的判断和控制逻辑．在数据的存储方面，存储器的位宽为64 bit，数据在存储器中按照ram_data=$\{a_i, a_{i+4}, a_{i+8}, a_{i+12}\}$的排列顺序进行填充和存储．然而，硬件要求存储器在一个周期内同时读取4组有效数据，双端口RAM显然无法做到，为了支持本节提出的数据调度方案，需要对存储器的设计展开进一步研究．

5 基于着色算法的多Bank存储方案

在第3节的设计中，存储器的位宽为64 bit，硬件设计的最大并行度为4，数据在存储器中按照ram_data=$\{a_i, a_{i+4}, a_{i+8}, a_{i+12}\}$的排列顺序进行填充和存储，每个周期从64 bit位宽的存储器同时读取4组数据，通过复制4组相同的RAM在一个周期内进行读写，会增加单元的存储面积和控制复杂度．为此，本文在不增加额外的存储开销的前提下，根据第4节中数据调用规律，将RAM分为多个Bank对数据进行分块存储，在进行运算的每次读写当中，Bank之间不存在数据访问冲突问题，同时，给出地址的转换逻辑，通过该地址转换逻辑来生成将外部地址唯一地映射到内部存储地址上．

依据第3节设计，以存储器64 bit的存储数据为单位，在NTT运算的4个stage内，同一时钟周期访问存储器数据不能存在在同一Bank之中，否则会导致数据访问冲突，因此Bank的数量应当不少于某个临界值，同时过多的Bank设计显然不合理，为此需要找出无数据访问冲突的最少的Bank数量，对数据进行合理分配，使地址的转换逻辑电路更加简单．目前已知存储数据量大小为64×64 bit，64组存储数据当中，每组数据均与其余数据中的9组数据冲突，不能划分到同一Bank中．为合理划分Bank，以上述64组存储数据为顶点，相互的冲

突关系形成关联矩阵,形成一张无向连通图,对Bank的划分即对顶点进行着色,着色后相同颜色的顶点表示数据会被划分到同一Bank中,相邻顶点间不能着同一颜色. 根据上述问题映射,能够将存储数据的存储划分问题转化为无向图的着色问题,如图7所示.

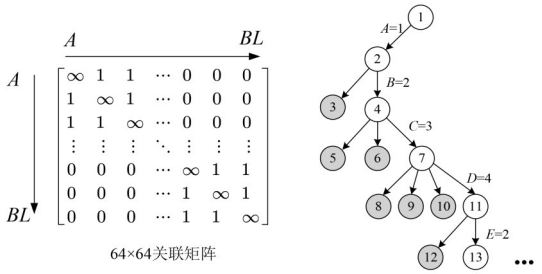

图7　NTT算法数据关联矩阵及着色算法解空间树

在着色问题当中,首先将所有顶点的颜色初始化为0,然后依次为每个顶点着色. 在上述问题的解空间树中,如果从根结点到当前结点对应一个部分解,也就是所有颜色指派都没有冲突,则在当前结点处选择第一棵子树继续搜索,也就是为下一个顶点着颜色1,否则,对当前子树的兄弟子树继续搜索,也就是为当前顶点着下一个颜色,如果所有m种颜色都已尝试过并且都发生冲突,则回溯到当前结点的父结点处,上一个顶点的颜色被改变,依此类推,算法6展示了无向图顶点着色算法的伪代码.

通过算法6搜寻最少的Bank数量为4,数据存储划分情况如图8所示,其中,方框内数字为存储号num,存储数据为ram_data$=\{a_i, a_{i+4}, a_{i+8}, a_{i+12}\}$,num$=i+i\gg 2$,且$i\%16\leqslant 3$.

算法6　无向图顶点着色算法

输入:无向连通图$G\boldsymbol{a}=a_0, a_1, \cdots, a_{n-1}$

输出:$\boldsymbol{a}=a_0, a_1, \cdots, a_{n-1}$着色方案color

```
color=0; k=1; m=1;
while(1)
    while(k≥0)
        color[k]=color[k]+1
        while(color[k]≤m)
            if(conflict(G, color)) break
            else color[k]=color[k]+1
        if(color[k]≤m&k=n−1)
            return m
        if(color[k]≤m&k<n−1)
            k=k+1
        else color[k−−]=0
    k=0 m=m+1
```

为降低控制器设计的复杂性,控制器向存储器发送存储号num作为外部访问地址,通过地址转化逻辑生成Bank编号Bank_num和Bank的内部地址Bank_addr基于上述分块存储,有

$$\text{Bank\_addr}=\text{num}\gg 2,$$
$$\text{Bank\_num}[1:0]=\{\text{num}[5]\oplus\text{num}[3]\oplus\text{num}[1],$$
$$\text{num}[4]\oplus\text{num}[2]\oplus\text{num}[0]\}.$$

通过4组地址访问4块Bank,在存储器的设计中加入Butterfly网络,通过Bank_num对该网络进行配置实现数据访问,并加入串并转换模块,在不同stage会对4×64 bit数据进行适时地串并转换,最终输出数据可直接送入运算单元中进行运算,写回的数据与写地址一同通过Butterfly网络访问存储Bank,最终存储器设计如图9所示.

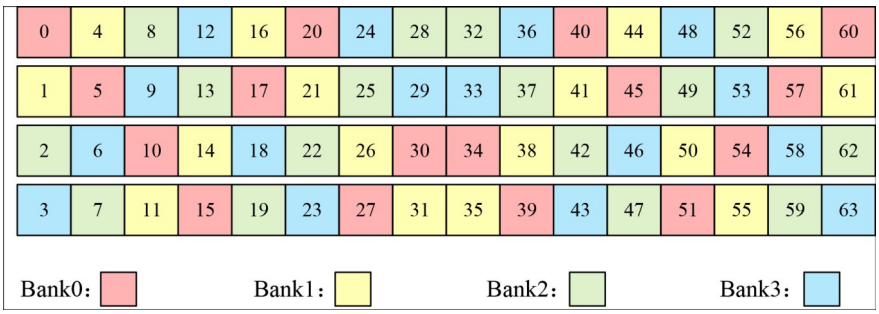

图8　多Bank数据存储方案

6　实验结果与分析

为进一步说明硬件设计的优越性,本文分别在FPGA和40 nm CMOS工艺下进行实验,并与近几年NTT算法的相关硬件实现进行了对比和分析. 本文首先通过Vivado2019.1软件进行仿真综合,选取芯片型号为Virtix-7中的xc7vx485tffg1158-3,最高工作频率达169 MHz. 表2具体展示了本文设计在FPGA上对比不同文献中硬件架构对同种参数的NTT算法硬件表现. 表3则列出本文设计架构在ASIC条件下的性能参数与相关研究成果的比较.

在可重构功能和具体实现方面,本文设计的NTT算法硬件架构具备实现混合基的NTT算法,在实现参数上能够支持16 bit以下、16~32 bit以内不同的NTT算法,

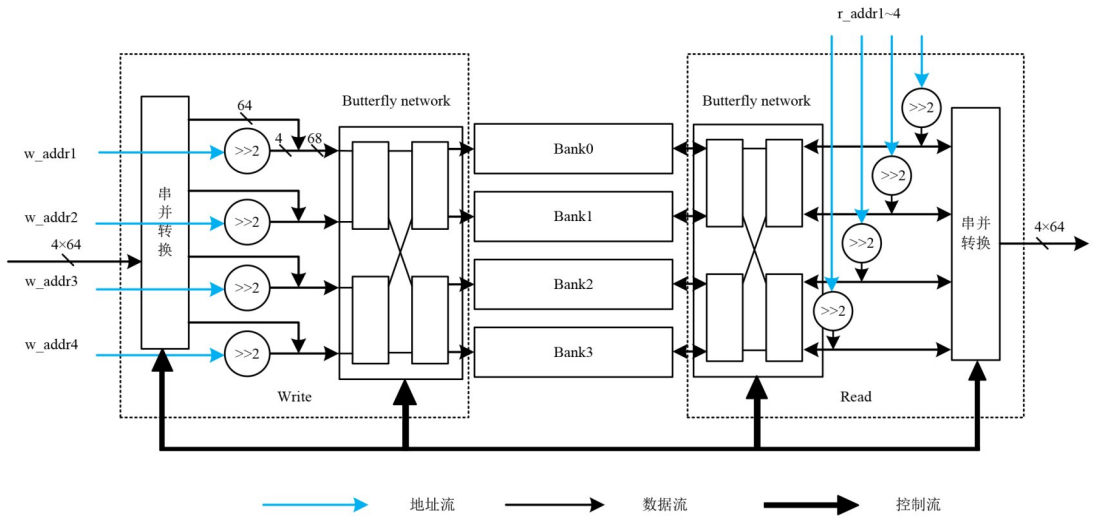

图9 避免数据冲突的多Bank存储器结构

表2 FPGA实验结果对比

对比文献	工艺	n, $\log_2(q)$	Freq/MHz	Cycles	LUT/FF/DSP/BRAM	时延/μs	ATP (LUT+FF)×Time
文献[4]	Virtex-7	25 616	228	556	2 294/1 645/7/3	2.44	9.6
文献[9]	Virtex-7	25 616	147	160	7 400/5 000/24/24	1.09	13.5
文献[10]	Virtex-7	25 613	186	156	11 000/5 182/64/12	0.84	13.6
文献[5]	Virtex-7	51 214	154	151	28 616/4 211/24/48	1.00	32
本文	Virtex-7	25 616 51 214	169	68 148	18 120/4 000/64/1	0.40 1.56	7.41 18.85

表3 ASIC实验结果对比

文献	工艺	n, $\log_2(q)$	Freq/MHz	Cycles	时延/μs	门数/kGe	AT积/(s*Ge)
文献[5]	28 nm CMOS	25 612	300	41	0.137	521	0.071
文献[8]	40 nm CMOS	25 613	72	1 289	17.903	19	0.34
文献[11]	65 nm CMOS	51 223	970	—	0.544	571	0.311
文献[10]	28 nm CMOS	25 613	540	32	—	581	0.034
本文	40 nm CMOS	25 613 51 223	684	72 580	0.105 0.848	301(292)	0.032(0.031) 0.255

且具备较高的资源利用率，相较于文献[9]、文献[10]，本文所设计硬件ATP降低约45.2%，文献[4]中提出一种pipeNTT硬件结构，使用了log(n)个PE单元串行排列，每个PE单独负责NTT算法中的一个stage，同时为各个PE分配了存储资源，本文相较该文献，ATP有效降低22.8%，并实现了硬件功能上的重构. 文献[5]设计了高性能的NTT加速引擎，设计了无访问冲突的存储方案并给出相应的形式化证明，相比该文献，本文的硬件设计ATP降低了41%，存储器的Bank数量明显减少，更有利于地址逻辑的生成.

在ASIC设计的实验当中，由于各个研究中使用的工艺节点不相同，因此在性能对比时以延时和资源消耗的乘积(AT积)作为主要指标. 文献[5]设计了一款VPQC向量处理器，对NTT运算单元进行了矢量化处理，硬件的并行度为32，虽然执行NTT运算操作的周期数较少，由于工作频率的限制导致整体AT积明显低于本文设计. 文献[8]中的运算单元与本文设计比较来看，都是将具体NTT运算与INTT运算进行了统一化设计，但由于没有将INTT最终的模除运算统一在每一轮的蝶形运算中，因此花费了大量时钟周期，由于使用单端口RAM的存储架构以及并未在单元上进行流水线设计，导致整体性能表现不佳，本文则设计固定的模除单元并使用双端口RAM，在单元上进行了流水线设计，因此在AT积方面相较该文献提升了一个数量级. 文献[10]采用32路并行设计，最高支持24 bit的NTT运算，本文设计能够在AT积相当的条件下，将算法支持位宽扩展至32 bit. 相比于文献[11]中NTT单元的硬件设计，本文AT积有效降低约18%.

整体来看,本文设计的可重构运算单元能够在增加少量资源消耗的条件下,支持实现混合基多位宽的NTT运算需求,通过ASIC仿真,延时、AT积优于表中所列出的设计. 在数据调度方面,为NTT运算的并行设计提供了合理的数据调度方案,并给出了解决NTT运算过程中数据访问冲突问题的解决思路与电路设计.

7 结论

本文通过对格基后量子密码算法中多项式乘法参数需求的分析,设计并实现了一款面向高速的可重构NTT运算单元,并解决了时序冲突和数据访问冲突两类问题,能够满足多款后量子密码算法中NTT算法的高速实现需求,同时也为NTT运算单元的设计和数据调用方案提供合理的设计建议.

参考文献

[1] LIU D S, ZHAO W, LIU Z, et al. Reconfigurable hardware design of multi-lanes number theoretic transform for lattice-based cryptography[J]. Electronics and Informatics, 2022, 44(2): 566-572.

[2] KUANG H L, ZHAO Y F, HAN J. A high-speed NTT-based polynomial multiplication accelerator with vector extension of RISC-V for saber algorithm[C]//2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). Piscataway: IEEE, 2022: 592-595.

[3] MONDAL S, PATKAR S, PAL T K. A configurable and efficient implementation of Number Theoretic Transform (NTT) for lattice based Post-Quantum-Cryptography[C]// 2022 IEEE 7th International Conference for Convergence in Technology (I2CT). Piscataway: IEEE, 2022: 1-6.

[4] YE Z W, CHEUNG R C C, HUANG K J. PipeNTT: A pipelined number theoretic transform architecture[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 69(10): 4068-4072.

[5] XIN G Z, HAN J, YIN T Y, et al. VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-V architecture[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(8): 2672-2684.

[6] MU J N, REN Y, WANG W, et al. Scalable and conflict-free NTT hardware accelerator design: Methodology, proof, and implementation[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023, 42(5): 1504-1517.

[7] CHUNG C M, HWANG V, KANNWISCHER M J, et al. NTT multiplication for NTT-unfriendly rings[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021: 159-188.

[8] BANERJEE U, UKYAB T S, CHANDRAKASAN A P. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols[EB/OL]. (2019-10-05) [2025-04-27]. https://arxiv.org/abs/1910.07557v2.

[9] CHEN X R, YANG B H, YIN S Y, et al. CFNTT: Scalable radix-2/4 NTT multiplication architecture with an efficient conflict-free memory mapping scheme[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 1: 94-126.

[10] ZHAO Y F, XIE R Q, XIN G Z, et al. A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2022, 69(7): 2871-2884.

[11] SHIMADA T, IKEDA M. High-throughput polynomial multiplier architecture for lattice-based cryptography[C]// 2021 IEEE International Symposium on Circuits and Systems (ISCAS). Piscataway: IEEE, 2021: 1-5.

作者简介

付秋兴　男,1999年8月出生于河南省商丘市. 现为信息工程大学网络空间安全专业博士研究生. 主要研究方向为全同态密码处理器设计.

E-mail: 1415505333@qq.com

李　伟　男,1983年11月出生于天津市. 现为信息工程大学教授. 主要研究方向为体系结构、安全芯片设计、集成电路技术.

E-mail: try_1118@163.com