格密码关键运算模块的硬件实现优化与评估

陈朝晖[1,2]　马原[2]　荆继武[1,3,†]

1. 中国科学院大学计算机科学与技术学院，北京 100049；2. 中国科学院信息工程研究所信息安全国家重点实验室，北京 100093；3. 北京大学软件与微电子学院, 北京 100871

† 通信作者，E-mail: jwjing@ucas.ac.cn

摘　要　为提高格密码在实际应用中的运算效率，提出格密码中多项式乘法运算的优化实现技术。多项式系数采用乒乓结构存储提升存取带宽，通过消除预缩放运算减少了10.5%的模乘运算和16.7%的存储空间占用；采用移位寄存器和三输入加法器的结构，有效地减少了逻辑资源占用。设计了具有可选层级的流水线结构，使多项式乘法中的蝶形运算模块可满足不同密码硬件系统的时序要求。评估结果表明，采用优化技术的低面积、均衡型和高性能实现的蝶形运算模块最大工作频率可分别达到150MHz、250MHz和350MHz以上。与现有实现技术相比，优化的多项式乘法硬件实现能够以更小的电路面积实现更高的工作频率，使电路效率提升22.8%。

关键词　后量子密码; 多项式乘法; 数论变换; 蝶形运算; FPGA

中图分类号　TP391

Hardware Optimization and Evaluation for Crucial Module of Lattice-Based Cryptography

Zhaohui Chen[1], Yuan Ma[1,2], and Jiwu Jing[1,3,†]

1. School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049;
2. State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing 100093;
3. School of Software and Microelectronics, Peking University, Beijing, 100871;

† Corresponding author, E-mail: jwjing@ucas.ac.cn

Abstract To improve the efficiency of lattice-based cryptography in practical application, the optimization technology of polynomial multiplication in lattice-based cryptography is proposed. The polynomial coefficients are stored in a ping-pong structure to improve the bandwidth. By eliminating pre-scale operations, 10.5% of modular multiplication operations and 16.7% of storage space are saved. The structure based on look-up table shift register and three-input adder is adopted to reduce the logical resource occupation. The pipeline structure with optional stages is designed to make the butterfly module in polynomial multiplication meet the timing requirements of different cryptographic hardware systems. The evaluation results show that the maximum frequency of low-area, balanced and high-performance implementations of the optimized butterfly unit can reach 150MHz, 250MHz and 350MHz, respectively. Compared with the existing implementation technologies, the optimized hardware implementation can achieve higher operating frequency with a smaller circuit area, which improves the efficiency of polynomial multiplication module by 22.8%.

Key words post-quantum cryptography; number theoretic transformation; FPGA; butterfly operation; hardware evaluation

基金项目：国家自然科学基金项目 (61872357，61802396)，国家密码发展基金项目(MMJJ20170205，MMJJ20180113)，北京数字认证股份有限公司委托研究项目 BJCA2020-YF-0300

收稿日期：0000-00-00；修回日期：0000-00-00；网络出版时间：

网络出版地址：

量子计算技术给现有 RSA、椭圆曲线公钥密码算法的安全性带来了巨大的挑战。1994 年，Shor[1]提出了一种可快速破解大整数分解和离散对数问题的量子算法。Gidney 等人[2]的研究发现，只需要 2000 万个量子比特即可在 8 小时内破解的 RSA2048 算法。为征集合适的替代性公钥密码算法标准，美国国家标准与技术研究院(NIST)自 2016 年起开始征集具有后量子特性的公钥加密、密钥交换和数字签名方案。其中，基于格上困难问题的方案具有计算速度较快[3]，密钥和密文相对较小[4]等优势，因此受到了广泛关注，成为热门的后量子密码候选方案[5]。

Lyubashevsky 等人[6]提出的环上错误学习(ring learning with errors, RLWE)问题是许多格密码算法的理论基础。RLWE 密码算法中的关键运算是多项式乘法运算，该运算可采用基于蝶形运算的数论变换(number theoretic transform, NTT)快速实现[7]-[8]。NTT 要求密码算法使用特定的模数，以 NewHope 算法[9]为代表的密钥交换算法和以 BLISS 算法[10]为代表的数字签名算法均选择适用于 NTT 的参数以提高运算速度。NTT 多项式乘法硬件实现受到了学术界的广泛关注，其设计难点一方面在于 NTT 控制逻辑包含多层循环嵌套的原位运算结构，存储器中的多项式系数存取调度复杂且带宽要求较高；另一方面，蝶形运算在 NTT 计算过程中执行次数多，例如 512 维多项式每次 NTT 需要执行 2304 次蝶形运算，而 1024 维多项式每次 NTT 需要执行 5120 次蝶形运算，NTT 中循环调用的蝶形运算模块包含模约减运算，这往往是多项式乘法硬件实现的性能瓶颈，进而决定密码算法的整体效率和运算时延。

在现有的硬件平台中，FPGA 具有开发周期较短且通用性较强的优势，它由诸多可编程的逻辑资源组成，已广泛应用于密码算法的优化、实现和评估[11]。本文提出 NTT 多项式乘法优化方法和硬件架构，对优化的硬件实现和现有硬件实现进行评估，为格密码硬件设计提供参考。主要的贡献如下:

1)提出蝶形运算模块优化技术，以适用于 NewHope 密钥交换算法和 BLISS 数字签名算法的模数为例，采用查找表移位寄存器和三输入加法器减少蝶形运算模块特别是模约减的逻辑资源占用;

2)提出 NTT 多项式乘法优化技术，采用乒乓结构提升数据存取带宽，并消除了预缩放运算，从而减少了 10.5%的模乘运算，节省了 16.7%的 ROM 存储空间；

3)设计了蝶形运算的流水线结构，充分利用片上功能单元以减少流水线资源开销，针对不同应用场景下的时序要求，我们提供了可选的流水线层级，分别满足低面积、均衡型和高性能实现三种时序要求;

4)采用 FPGA 实现和评估了所提出的优化技术，与现有技术相比，我们的优化技术能够以更小的电路面积实现更高的工作频率，使多项式乘法模块得到 22.8%的效率提升。

1 相关工作

2014 年，Pöppelmann 等人[12]在 FPGA 上的 BLISS 数字签名实现达到了比 RSA 签名和 ECDSA 签名更大的吞吐且占用面积更小。2016 年，Bos 等人[4]的研究成果表明，使用 Newhope 密钥交换算法通过 TLS 提供 web 页面能够实现比 ECDHE 算法或 NTRU 算法更大的网络吞吐。Newhope 密钥交换算法和 BLISS 数字签名能够高效实现的重要原因是均设定模数为 12289，以使用基于蝶形运算的 NTT 将多项式乘法的时间复杂度从$\mathcal{O}(n^2)$降低到$\mathcal{O}(n\log n)$。

尽管 NTT 已经大幅减少了多项式乘法的复杂度，但 NTT 多项式乘法需要预缩放、后缩放等运算步骤，这造成了一定的运算时延；此外，NTT 的原位运算实现需要较大的数据存取带宽，一般的双端口 RAM 不能直接满足其数据存取需求。学者们提出了针对多项式乘法中预缩放运算和数据存取的优化技术。Roy 等人[13]通过整合 NTT 正变换和预缩放运算减少了模乘运算次数，该工作还提出了一种高位宽的数据存储结构，该方案虽然提高了存储器的吞吐，但需要消耗额外的时钟周期进行数据重排。Oder 等人[14]首次在 FPGA 上完整实现了 NewHope 密钥交换算法，但其模约减模块中包含延时较大的关键路径，并且数据存取带宽不足，需占用额外的时钟周期等待数据流，而等待期间运算单元处于空闲状态，因此其电路的效率并不高。

由于多项式系数在商环中运算，NTT 中蝶形运算采用的模约减算法是多项式乘法优化的一个重点。模约减算法与除法运算密切相关，2016 年，Poppelmann[15]提供了一种基于长除法的基本模约减(Generic)算法，并采用高层次综合的方法在 FPGA 上实现了该算法的流水线模块。基本模约减算法采用“比较器-减法器”逻辑链的结构，且逻辑链的长度与模数大小正相关，因此基本模约减算法往往需要较长的逻辑链。文献[15]提供的电路面积消耗较大，且该电路的性能优化过程需要插入的流水线级数较多，因此整个硬件模块效率不高。Barrett 约减[16]是一种基于商估计的算法，在蝶形运算中有较多的应用实例。2015 年，Liu 等人[17]提出 SAMS2 模约

减，该方法是一种以移位、加法、乘法、两次减法组成的模约减算法优化实现，适合在资源受限的微控制器上实现。2016 年，Poppelmann[15]实现了完全采用移位和加减法代替乘法的 FPGA 硬件实现，该实现的电路面积较基本模约减算法显著减小，但电路最大工作频率相对较低。 2019 年，Xing 等人[18] 在 FPGA 上实现了一种对加法树器优化的 Barrett 约减方案能够同时输出商和剩余，但由于乘数的汉明重量较大，该电路构建了较复杂的加法器树。2019 年，Banerjee 等人[19]提出了一种模数可控制的模算术电路，并给出了其 ASIC 实现。由于 Barrett 约减原理的限制，其硬件实现需要比较器-减法器电路以解决商估计值存在的误差。Montgomery 约减、模乘、模幂算法[20]一直广泛应用于密码优化实现和实现领域，但经典 Montgomery 约减实现中也需要比较器-减法器电路以得到最终结果。Alkim 等人[9]提出一种适用于蝶形运算的不完全模约减(incomplete reduction)算法，本文称该算法为 ADPS 算法。不完全模约减是指将输入约减到某个宽松的数值范围而不要求约减结果为非负最小剩余。ADPS 算法对加法结果采用不完全 Barrett 约减，对乘法结果采用不完全 Montgomery 约减。模数取 12289 时，ADPS 算法通过选取合适的寄存器宽度和运算参数可恰好使乘法结果约减到 14 比特以内的无符号整数，免去了 Barrett 约减和 Montgomery 约减的判断及减法操作。与传统 Barrett 或 Montgomery 相比，不完全约减能够节省比较器和减法器资源，因此非常适合软硬件实现。Jati 等人[21]在 FPGA 平台上实现了 ADPS 算法，由于纯组合电路的实现路径延时较大，该实现插入了 3 级流水线以提高电路的最大工作频率。

2016 年，Longa 和 Naehrig[22]提出了一种适用于模数 $q=k\cdot 2^p+1$ 的模约减算法，本文称为 LN 模约减，并将这种该算法应用到了 NTT 多项式乘法中。LN 模约减也是一种不完全约减算法，它包括用于加法结果约减的 K-RED 算法和用于乘法结果约减的 K-RED2x 算法，且运算过程中参数采用有符号整数表示，实验结果表明基于 LN 模约减的 NTT 比基于 ADPS 算法的 NTT 软件实现速度提高 86%以上。2017 年，Kuo 等人[23]提出 LN 模约减同样适用于硬件实现，为适配 FPGA 片上的数字信号处理器单元(DSP)并避免寄存器中数值溢出，Kuo 等人将原方案中参数的有符号数表示调整为无符号数表示；该文献实现了一种适用于 FPGA 的流水线电路结构，达到了较高的电路性能，但该实现中使用的加法器数量较多。

综上，设计 NTT 中高带宽的多项式系数存取方案，以及蝶形运算中模约减的高效实现方法，仍是多项式乘法硬件实现中困难的问题。本文提出基于 LN 模约减的 NTT 多项式乘法硬件实现优化方法，并提供不同层级的流水线优化实现，因此对格密码硬件实现具有很高的参考价值。

2 多项式乘法的高效算法

整数环用□表示，$□_q$ 为模整数 q 的商环，$R=\mathbb{Z}[X]/\left(X^n+1\right)$ 是模多项式 X^n+1 的整系数多项式环，其中 n 为多项式维度即多项式系数的个数，$R_q=\mathbb{Z}_q[X]/\left(X^n+1\right)$ 表示系数模 q 的整系数多项式环。对于 n 维多项式 $\boldsymbol{a}=\left(a_0,\ldots,a_{n-1}\right)$，$\boldsymbol{b}=\left(b_0,\ldots,b_{n-1}\right)\in R_q$，其中 $a_0,\ldots,a_{n-1},b_0,\ldots,b_{n-1}\in□_q$ 表示多项式系数，多项式乘法运算 $\boldsymbol{ab}$ 是一种卷积形式的运算，如果直接计算，R_q 上的多项式乘法需要 n^2 次模乘运算[7]。本节主要介绍整系数模 q 运算采用的高效模约减算法，以及采用 NTT 计算 R_q 上的多项式乘法。

2.1 高效模约减算法

LN 模约减适用于形如 $q=k\cdot 2^p+1$ 的模数，其中 k 为奇数且 $2^p>k$。对于 $a,b\in□_q$，以求乘积 $x=a\cdot b$ 对模数 q 的模运算为例，其思路是利用 $k\cdot 2^p\equiv -1 \bmod q$ 的特性，把 $x \bmod q$ 的计算转化为 $x\cdot k \bmod q$，再消去乘 k 的影响。特别的，q 取 12289 时，k 的值取 3，p 的值取 12。LN 模约减包含 K-RED 和 K-RED2x 两种算法形式，K-RED 算法将 x 表示为 $x=x_0+2^p x_1$，从而得到算法 1：

算法 1 K-RED 模约减算法

输入：正整数 $x\in\left\{0,(q-1)^2\right\}$；

输出：模约减结果 $r = x \cdot k \bmod q$。

1 $x_0 \leftarrow x \left(\bmod 2^p\right)$

2 $x_1 \leftarrow x / 2^p$

3 return $kx_0 - x_1$

K-RED2x 算法将 x 表示为 $x = x_0 + 2^p x_1 + 2^{2p} x_2$，从而得到算法 2：

算法 2 K-RED-2x 约减算法。

输入：正整数 $x \in \left\{0, (q-1)^2\right\}$；

输出：约减结果 $r = x \cdot k^2 \bmod q$。

1 $x_0 \leftarrow x \left(\bmod 2^p\right)$

2 $x_1 \leftarrow x / 2^p \left(\bmod 2^p\right)$

3 $x_2 \leftarrow x / 2^{2p}$

4 return $k^2 x_0 - k x_1 + x_2$

算法 1 和算法 2 的输出 r 均为有符号数，文献[23]采用了 LN 模约减，并对 r 加一个模数 q 的整数倍确保约减最终结果非负。其加法结果的约减采用 K-RED 算法，乘法结果的约减采用 K-RED-2x 算法，约减过程引入的 k 和 k^2 均可通过预计算等方式消除。

2.2 NTT 多项式乘法

NTT 的原理是将标准域的多项式乘法运算转换为 NTT 域上多项式系数对位点乘运算。对于一个满足维度 n 为 2 的正整数次幂，且模数 $q \equiv 1 \bmod 2n$ 的多项式 $\boldsymbol{a}$，NTT 正变换如算法 3 所示，其中 ω_n 是 $\square_q$ 中的本原 n 阶单位根，φ 是 ω_n 模 q 的平方根。

算法 3 NTT 正变换

输入：标准域多项式 $\boldsymbol{a}$，预计算 φ 的 i 次幂并按 i 的比特倒位序地址存储在 ROM 存储器中，其中 $i = 0,1,\ldots,n-1$；

输出：NTT 域多项式 $\tilde{\boldsymbol{a}}$=$\mathrm{NTT}_{\omega}^{n}(\boldsymbol{a})$。

1 $\boldsymbol{a} \leftarrow$ bit_reverse($\boldsymbol{a}$)　　多项式系数倒位序

2 $t = n$

3 for $m = 1$ to n by $m = 2m$ do

4 　　$t = t / 2$

5 　　for $i = 0$ to m do

6 　　$j_1 = 2 \cdot i \cdot t$

7 　　$j_2 = j_1 + t - 1$

8 　　$\omega = \mathrm{ROM}[i]$

9 　　for $j = j_1$ to j_2 do

10 　　$u = \text{K-RED}(\boldsymbol{a}[j])$

11 　　$v = \text{K-RED2x}(\boldsymbol{a}[j+t] \cdot \omega)$

12 　　$s = (\boldsymbol{a}[j+t] \cdot \omega)[11]$　　选取第 11 比特

13 　　$\boldsymbol{a}[j] = s\,?\,u + v : u + v + 2q$

14 　　$\boldsymbol{a}[j+t] = s\,?\,u + v + 4q : u + v + 2q$

15 end for

16 end for

17 end for

18 $\tilde{\boldsymbol{a}} \leftarrow \boldsymbol{a}$

19 return $\tilde{\boldsymbol{a}}$

由于多项式 $\boldsymbol{a}$ 的系数均由随机采样获得，可认为采样结果本身是倒位序的，从而将算法 3 的第 1 行省去[23]。算法 3 中的第 10-14 行即蝶形运算，蝶形运算处于 NTT 变换的最内层的循环，在串行实现的 NTT 变换中，蝶形运算作为一个算术逻辑单元被循环调用。值得注意的是，由于每次 K-RED 算法将结果缩放 k 倍，K-RED2x 算法将结果缩放 k^2 倍，若将算法 3 中每个 ω_n 在预计算时均乘 k^{-1}，则最终得到的 $\tilde{\boldsymbol{a}}$ 相当于每个多项式系数都乘以缩放系数 $k^{\log n}$。NTT 逆变换与算法 3 类似，区别在于其中预计算的 φ 幂替换为 φ^{-1} 幂，且最后需要给各系数乘缩放因子 n^{-1}。

在采用 NTT 计算多项式乘法时，首先采用基于蝶形运算的 NTT 正变换将标准域下的多项式转化为 NTT 域下的多项式，两个 NTT 域多项式将系数对位点乘，点乘结果再通过 NTT 逆变换转换到标准域，如算法 4 所示。算法 4 中，除了正反 NTT 变换和系数对位点乘运算外，还包括正 NTT 变换前乘 φ^i 的预缩放(第 2-3 行)和逆 NTT 变换后乘 φ^{-i} 的后缩放(第 12 行)，其中算法 4 的后缩放已经和乘 n^{-1} 运算合并。此外，为消除 LN 模约减引入的系数 k 幂，需要在运算中引入相应的逆系数。

算法 4 基于 LN 模约减的 NTT 多项式乘法

输入： $\boldsymbol{a},\boldsymbol{b} \in \mathbb{Z}_q[X]/\left(X^n+1\right)$； $k^{-1}\omega_n^i, k^{-1}\omega_n^{-i}$的预计算值,其中$i=0,1,\ldots,n/2-1$；

$\varphi^i k^{-(2+\log n)}, \varphi^{-i} k^{-(4+\log n)} n^{-1}$的预计算值,其中$i=0,1,\ldots,n-1$；

输出： $\boldsymbol{c} = \boldsymbol{a} * \boldsymbol{b} \in \mathbb{Z}_q[X]/\left(X^n+1\right)$。

1 for $i=0$ to $n-1$ do

2 $\quad \bar{\boldsymbol{a}}_i \leftarrow \text{K-RED2x}\left(\boldsymbol{a}_i\left(\varphi^i k^{-(2+\log n)}\right)\right)$

3 $\quad \bar{\boldsymbol{b}}_i \leftarrow \text{K-RED2x}\left(\boldsymbol{b}_i\left(\varphi^i k^{-(2+\log n)}\right)\right)$

4 end for

5 $\tilde{\boldsymbol{a}} \leftarrow \text{NTT}_{\omega}^{n}(\bar{\boldsymbol{a}})$

6 $\tilde{\boldsymbol{b}} \leftarrow \text{NTT}_{\omega}^{n}(\bar{\boldsymbol{b}})$

7 for $i=0$ to $n-1$ do

8 $\quad \tilde{\boldsymbol{c}}_i \leftarrow \text{K-RED2x}\left(\tilde{\boldsymbol{a}}_i \tilde{\boldsymbol{b}}_i\right)$

9 end for

10 $\bar{\boldsymbol{c}} \leftarrow \text{NTT}_{-\omega}^{n}(\tilde{\boldsymbol{c}})$

11 for $i=0$ to $n-1$ do

12 $\quad \boldsymbol{c}_i \leftarrow \text{K-RED2x}\left(\bar{\boldsymbol{c}}_i\left(\varphi^{-i} k^{-(4+\log n)} n^{-1}\right)\right)$

13 end for

14 return $\boldsymbol{c}$

3 面向硬件实现的优化

尽管现有模约减和多项式乘法已有较高效的算法，但其硬件实现技术还不完全成熟。本节以 NewHope 密钥交换算法和 BLISS 数字签名算法中通用的 R_q，$q=12289$ 为例，介绍多项式乘法算法的硬件优化方法。

3.1 基于 LN 模约减的蝶形运算硬件优化

LN 模约减包含由层级的加法器级联形成的加法树结构，文献[23]中的硬件实现广泛采用二输入加法器构建加法树结构。在主流 FPGA 架构中，六输入查找表(LUT)结构可以高效地例化为三输入加法器，其占用的逻辑资源量与二输入加法器相同[24]，因此三输入加法器的路径延时与普通加法器差异很小而集约程度更高。

基于优化 LN 模约减的蝶形运算结构如图 1 所示，优化的思路是尽可能采用三输入加法器构建硬件模块，这有助于节省 LUT 逻辑资源。图 1 中的 K-RED 算法使用了 1 个三输入加法器，K-RED2x 算法模块使用了 2 个三输入加法器，整个蝶形运算模块共使用了 5 个三输入加法器。文献[23]中提出的硬件结构使用了 10 个加法器，而本文采用三输入加法器合并加减法逻辑，可节省 50%的加法器数量。

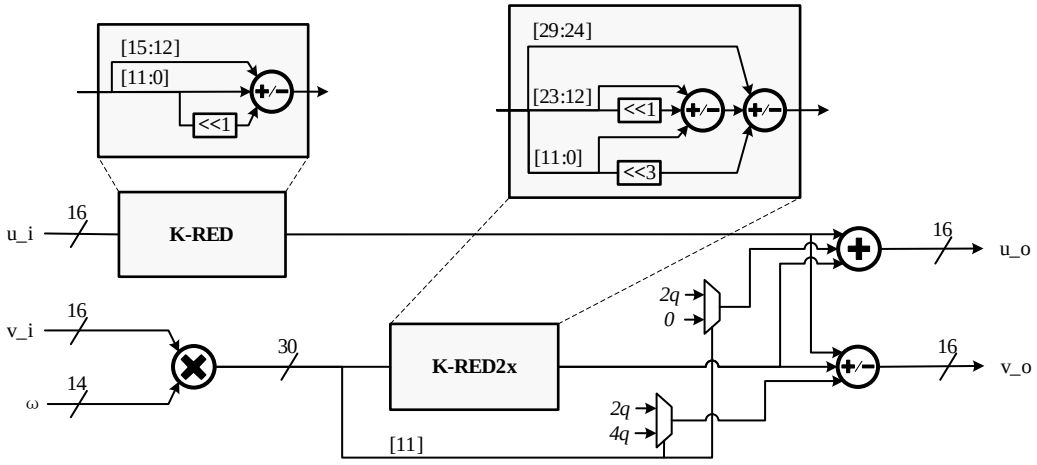

图 1 基于优化 LN 模约减的蝶形运算结构

Fig. 1 Butterfly operation structure based on optimized LN modular reduction.

3.2 多项式乘法的硬件优化

根据文献[15]提出的方法，算法 4 中第 2、3 行的预缩放运算可以整合在正 NTT 变换中。该优化只需要调整每次蝶形运算中 ω 的取值，即将 NTT 算法 3 中第 8 行改为 $\omega=\mathrm{ROM}[m+i]$。采用该优化得到的 NTT 正变换结果与带有预缩放的普通 NTT 运算结果一致，可减少算法 4 中 10.5%的模乘运算，得到算法 5。

算法 5 基于 LN 模约减的优化 NTT 多项式乘法

输入： $\boldsymbol{a},\boldsymbol{b}\in\mathbb{Z}_q[X]/\left(X^n+1\right)$；$k^{-1}\omega_n^{-i}$的预计算值,其中$i=0,1,\ldots,n/2-1$；

$k^{-1}\varphi^i,\varphi^{-i}k^{-(4+2\log n)}n^{-1}$的预计算值,其中$i=0,1,\ldots,n-1$；

输出： $\boldsymbol{c}=\boldsymbol{a}*\boldsymbol{b}\in\mathbb{Z}_q[X]/\left(X^n+1\right)$。

1 $\tilde{\boldsymbol{a}}\leftarrow \mathrm{NTT}^n_{\omega\text{优化}}(\boldsymbol{a})$

2 $\tilde{\boldsymbol{b}}\leftarrow \mathrm{NTT}^n_{\omega\text{优化}}(\boldsymbol{b})$

3 for $i=0$ to $n-1$ do

4 $\quad\tilde{\boldsymbol{c}}_i\leftarrow \mathrm{K\text{-}RED2x}\left(\tilde{\boldsymbol{a}}_i\tilde{\boldsymbol{b}}_i\right)$

5 end for

6 $\bar{\boldsymbol{c}}\leftarrow \mathrm{NTT}^n_{\omega^{-1}}(\tilde{\boldsymbol{c}})$

7 for $i = 0$ to $n-1$ do

8 $\quad \boldsymbol{c}_i \leftarrow \text{K-RED2x}\left(\bar{\boldsymbol{c}}_i \left(\varphi^{-i} k^{-(4+2\log n)} n^{-1}\right)\right)$

9 end for

10 return $\boldsymbol{c}$

分析算法 5，其相比算法 4 另一个优势是节省预计算参数对存储的占用，算法 4 需要 $3n$ 个预计算参数，而算法 5 需要 $2.5n$ 个预计算参数，从而可以减少 16.7%的只读存储器空间占用。算法 5 将 LN 模约减引入的系数 k 幂在后缩放阶段(第 8 行)消除。

观察算法 3，由于每次蝶形运算需要获取 2 个多项式系数并写入 2 个多项式系数，但每个双端口存储器每个时钟周期仅能读或写 2 个地址的数据，因此蝶形运算的直接原位运算实现需要额外的时钟周期执行数据读写，造成数据驻留[14][14]。为提高存储器的吞吐，本文采用乒乓结构的存储器存取方式。即例化两个同样大小的存储器 RAM0 和 RAM1，多项式系数的初始值存储在 RAM0 中，当算法 3 中 log$t$ 为奇数时，蝶形运算模块从 RAM0 读取多项式系数，RAM1 写入蝶形运算得到的新系数结果；反之 log$t$ 为偶数时，蝶形运算模块从 RAM1 读取多项式系数，RAM0 写入蝶形运算得到的新系数结果。对于 log$n$ 为偶数的情况，NTT 的最终结果将存储在 RAM0 中。

4 基于 FPGA 的多项式乘法运算实现

本节采用 FPGA 的硬件原语实现第三章中提出硬件实现优化方法，并提出高效的流水线实现方法。针对不同时序要求，提供可选的流水线层级。

4.1 基于 LN 模约减的蝶形运流水线设计

在 FPGA 平台上，由于多项式数据量较大，其系数存储一般采用块 RAM(block RAM, BRAM)实现，而较高位宽的乘法运算一般采用 DSP 实现。BRAM 和 DSP 都有丰富的逻辑资源，可采用流水线技术达到高工作频率。

基于优化 LN 模约减的蝶形运算流水线实现如图 2 所示，其中灰色的流水线层级作为基本流水线能够确保一般时钟频率要求，白色的流水线层级间仅有一层算术逻辑，可供更高性能要求的实现选择插入。在硬件实现方面我们采用了以下三个方面的设计优化：

1) 可选的流水线层级。由于图 1 所示的组合逻辑层级较多，其组合逻辑产生的路径延时较大。为改善路径延时，我们在u_o和v_o对应的路径插入相同级数的流水线，确保结果在同一时钟周期输出。流水线层级可通过条件编译选择，从而调整电路所需的面积和最大工作频率。

2) 充分利用移位寄存器资源。在 FPGA 中，部分 LUT 资源可以作为移位寄存器使用，从而节省 FPGA 的触发器(FF)资源。调用移位寄存器要求流水线层级间不存在组合逻辑，例如图 2 中减法器输出至u_o路径不是关键路径，其作为时延插入的流水线结构可采用移位寄存器实现。

3) 充分利用 DSP 单元资源。蝶形运算中的乘法运算需要占用一个 DSP 单元，除乘法器资源外，我们的流水线实现还使用了 DSP 单元的 3 层流水线寄存器资源。利用该 DSP 单元中的资源有利于减少 DSP 外部 FF 的占用。

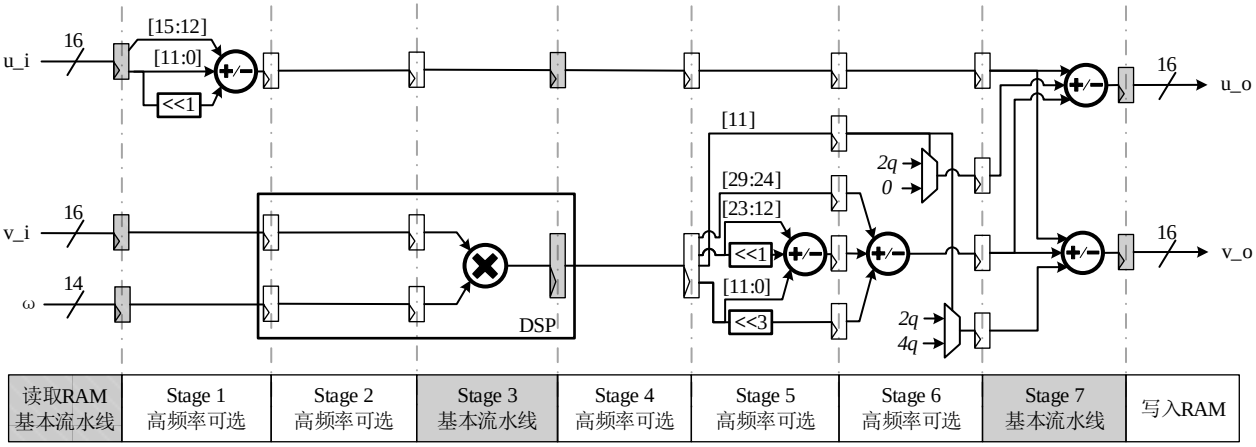

图 2 基于优化 LN 模约减的蝶形运算流水线实现

Fig. 2 Pipeline implementation of butterfly operation based on optimized LN modular reduction.

4.2 系统总体结构设计

多项式乘法处理器的硬件结构如图 3 所示，主要包含控制模块、蝶形运算模块和存储模块。其中控制模块负责控制蝶形运算模块的运算逻辑以及存储模块的读写地址、写使能以及乒乓转换；蝶形运算模块可以实现蝶形运算及模乘运算；存储模块包括 RAM 和 ROM，RAM 存储多项式系数及其运算中间值，ROM 存储多项式乘法运算过程中的预计算参数，在执行正 NTT 变换时从 ROM 中读取 $k^{-1}\varphi^{i}$ 区域对应参数，执行逆 NTT 变换时从 ROM 中读取 $k^{-1}\omega_n^{-i}$ 区域对应参数，执行后缩放运算时从 ROM 中读取 $\varphi^{-i}k^{-(4+2\log n)}n^{-1}$ 区域对应参数。

对于流水线层级为 ε 的蝶形运算模块，数据输出和读入之间有 ε 个时钟周期的延时，因此当一个存储器由读向写方向改变时，另一个存储器还未完成写操作，即在读写方向改变时会造成 ε 个时钟周期的流水线数据驻留。本文采用备份 RAM 作为缓冲区，确保数据存取时数据通路始终保持全流水。备份 RAM 的原理是，当原 RAM 写入没有完成时，控制模块选通备份 RAM 读出多项式系数，能够确保这部分数据与原 RAM 对应地址的数据相同。例如，起始时 ROM0 为读出方向，ROM1 和 RAM1 $_{备份}$同时写入，当 ROM0 读出完成时，控制模块选通 RAM1 $_{备份}$读出多项式系数， RAM1 继续完成 ε 个时钟周期的写入操作，RAM1 写入完成后，控制模块选通 RAM1 读出多项式系数，ROM0 和 RAM0 $_{备份}$同时写入。由于 ε 数值较小，下次乒乓方向转换时从 RAM0 $_{备份}$读出的数据与 ROM0 相同，且整个运算过程不会产生读写地址冲突，确保运算结果的正确性。

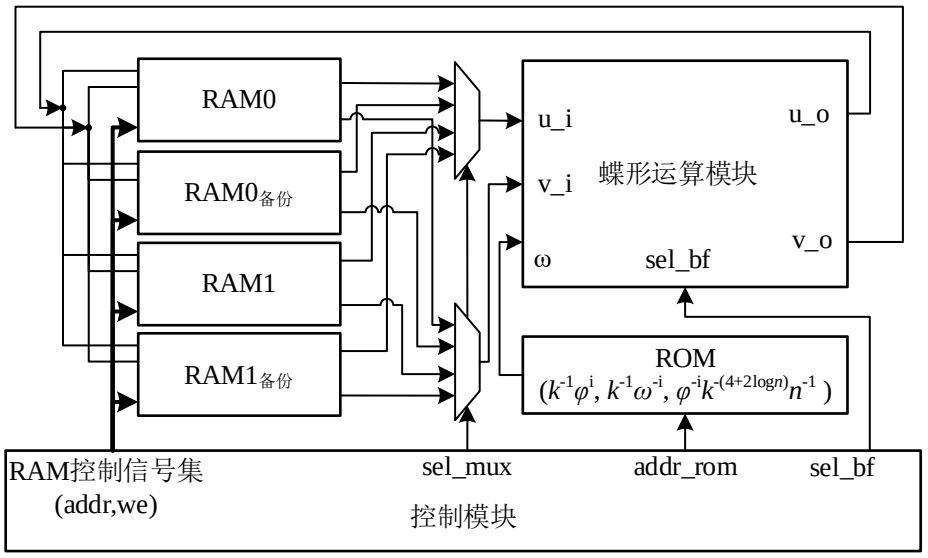

图 3 多项式乘法处理器总体架构图

Fig. 3 Overall architecture of polynomial multiplication processor.

5 实验结果与评估

为比较采用不同模约减算法的蝶形运算模块的面积和性能，我们将本文提出的硬件实现优化方法与现有实现方法进行对比评估。实验参数采用通用的模数 $q=12289$，多项式维度 $n=1024$。本文硬件实现的测试平台采用 Xilinx 公司的 XC7A200T 型号 FPGA。

5.1 蝶形运算硬件评估

我们分别给出了基于优化 LN 模约减的蝶形运算模块及其它现有蝶形运算模块在低面积、均衡型以及高速率三种频率设计要求下的实现数据，以充分比较各方案在不同时序要求下的性能。其中低面积表示占用逻辑资源较少，典型以 3 级流水线实现；均衡型表示兼顾资源占用量与运算速度，典型以 5 级流水线实现；高速率表示工作频率高，典型以 8 级以上流水线实现。评估结果如表 1 所示，表中 Montgomery、Barrett 和 Generic 算法所在的行表示该硬件实现的蝶形运算模块中采用对应的模约减算法。FPGA 中的资源类型较多，这里列举了占用基本逻辑资源(Slice)及其中 LUT、FF 资源的数量，此外表中所有硬件实现均占用 1 个 DSP 单元。

文献[23]仅提供了一种流水线设计，我们将基于优化 LN 模约减的均衡型实现与该方案对比。结果表明，由于三输入加法器的引入，本文的优化共减少了其 12.7%的 Slice，并减少了其 32.2%的 LUT 资源占用。基于优化 LN 模约减的低面积实现所需要的硬件资源最少，比基于优化 LN 模约减的均衡型实现进一步节省了 27.1%。基于优化 LN 模约减的高性能实现能达到最高的工作频率，且比基于 ADPS 模约减的高性能实现节省 21.3%的硬件资源。

在完全约减的实现方案中，基于 Montgomery 算法的蝶形运算模块资源占用较少且工作效率较高，也能够满足 250MHz 以上的工作频率要求，相比 Barrett 算法和 Generic 算法优势较为明显。文献[15]给出的基于高层次综合的实现结果虽然能够实现相对较高的工作频率，但占用的逻辑资源较多。相较而言，基于不完全约减的蝶形运算实现在面积和性能上均优于完全约减的蝶形运算，基于优化 LN 模约减的蝶形运算实现综合效果最优。

表 1 蝶形运算硬件实现评估

Table 1 Evaluation of butterfly operations on hardware

硬件实现	输出	器件	流水线	Slice	LUT	FF	最大频率 (MHz)
低面积实现							
优化 LN	$[0, 2^{16})$	Artix-7	3	**35**	**83**	93	**157**
ADPS [9]	$[0, 2^{14})$		3	56	114	**84**	154
Montgomery	[0, 12289)		3	58	130	**84**	120
Barrett	[0, 12289)		3	104	255	**84**	103
Generic	[0, 12289)		3	130	333	**84**	26
均衡型实现							
优化 LN	$[0, 2^{16})$	Artix-7	5	**48**	**80**	174	**250**
LN [23]	$[0, 2^{16})$		5	55	118	200	**250**
ADPS[9]	$[0, 2^{14})$		5	60	113	162	243
Montgomery	[0, 12289)		5	70	142	134	231
Barrett	[0, 12289)		5	99	264	**99**	179
Generic	[0, 12289)		5	123	363	113	73
高性能实现							
优化 LN	$[0, 2^{16})$	Artix-7	8	**59**	**89**	**189**	**359**
ADPS [9]	$[0, 2^{14})$		8	75	124	240	**359**
Montgomery	[0, 12289)		9	74	159	274	274
Barrett	[0, 12289)		9	100	229	235	208

Generic	[0, 12289)	Spartan-6	19	123	326	295	209
Barrett [15]	[0, 12289)		NA	111	362	235	208
Generic [15]	[0, 12289)		NA	245	782	473	298

说明：粗体标记表示最优结果。

5.2 多项式乘法运算硬件评估

我们采用表 1 中基于优化 LN 模约减的蝶形运算模块高性能实现完成了算法 5 的硬件实现，共占用 151 个 Slice。本文多项式乘法硬件实现与现有硬件实现的比较结果如表 2 所示，表中 AT 积(area time product)表示电路面积与运算时延的乘积，AT 积越小表示电路效率越高。为便于比较，表 2 中所有硬件实现的 AT 积均做归一化处理。

表 2 多项式乘法运算硬件实现评估

Table 2 Evaluation of polynomial multiplication module on hardware

硬件实现	器件	时延	LUT/AT 积	FF/AT 积	DSP/AT 积	BRAM/AT 积	最大频率(MHz)
本文	Artix-7	56.62	**389/1.00**	346/1.00	**1/1.00**	3.5/1.00	**307**
Kuo[23]	Artix-7	52.32	2382/5.66	1381/3.69	8/7.39	10/2.64	150
Oder[14]	Artix-7	1007.79	432/19.77	**278**/14.30	2/35.60	4/20.34	125
Xing[18]	Zynq-7000	**28.44**	2942/3.80	4881/7.09	8/4.02	**0/0.00**	153
Neng[25]	Zynq-7000	33.68	847/1.30	375/**0.64**	2/1.19	6/1.02	244
Jati[21]	Zynq-7000	88.45	524/2.10	823/3.72	3/4.69	3/1.34	251
Fritzmann[26]	Zynq-7000	239.72	886/9.64	618/7.56	26/110.08	1/1.21	NA

说明：粗体标记表示最优结果。

相比其它的多项式乘法硬件实现，本文硬件实现的工作频率最高，可达到 300MHz 以上，这主要得益于合理的流水线设计。多项式乘法模块的硬件实现要根据设计的目标工作频率确定蝶形运算模块的流水线级数。对于工作频率要求较低的硬件实现，蝶形运算模块的模约减运算一般为关键路径，可选择适当的流水线级数达到目标工作频率；对于工作频率要求较高的硬件实现，蝶形运算可以采用流水线技术进一步优化，但 NTT 控制逻辑的循环嵌套会产生较复杂的选通电路且其插入流水线的成本很高，因此控制逻辑更容易成为关键路径。

本文硬件实现的电路面积同样具有优势，其特点是具有均衡的 LUT 和 FF 数量，并且充分利用了 DSP、BRAM 中的逻辑资源，因此即使插入了更多级数的流水线，其电路面积并没有大幅增加。文献[23]、文献[18]和文献[25]通过例化 2 至 4 个并行的蝶形运算模块实现了更低的运算时延，相应的代价是电路面积明显大于本文硬件实现。文献[23]采用了基于 LN 模约减的蝶形运算，且 4 个并行的蝶形运算模块有效地提高了运算速度，但该实现没有消除预缩放运算，也需要更多 BRAM 存储预计算的参数值，且工作频率不高，因此各硬件资源的效率不高。文献[18]提出的 4 路蝶形运算模块并行实现的运算时延最小，且该硬件实现不占用 BRAM，其蝶形运算模块采用优化的 Barrett 约减，整体工作频率不高，且该硬件实现同样未消除预缩放运算，因此 LUT、FF、DSP 的效率均明显低于本文的硬件实现。文献[25]中 FF 的效率更高，但考虑到现有 FPGA 器件每个 Slice 结构中 FF 和 LUT 的数量比值一般为 2，而文献[25]中 LUT 和 FF 的利用很不均衡，最终 Slice 的占用会受数量较多的 LUT 决定，因此其基本逻辑资源的 AT 积约为本文硬件实现的 1.3 倍，其 DSP 和 BRAM 的 AT 积效率与本文硬件实现接近。文献[14]实现了较少的逻辑资源占用，但其单次多项式系数读取和模约减运算均需要大于 1 个时钟周期，造成流水线数据驻留，因此其各类硬件资源的效率不高。文献[21]的多项式乘法运算采用

了文献[9]中的模约减方法，相比本文硬件实现效率较低。表 2 中文献[26]的时延未计入系数对位乘法步骤，其工作频率按照等同本文硬件实现的工作频率计算，该硬件实现可作为 RISC-V 处理器的密码协处理器，尽管其相比软件实现具有更高的性能，但其需要额外的软硬件通信开销，因此相比纯硬件实现在效率上不占优势。

总体而言，本文提出的多项式乘法硬件实现技术至少能够提高 22.3%的工作频率，并提高 FPGA 逻辑资源 22.8%的效率。根据文献[18]的实现结果，在免去顺序倒置操作的情况下，多项式乘法运算约占 NewHope 算法全部密码运算时延的 80%，本文对多项式乘法运算的频率提升大约能够减少 NewHope 算法 15%的密码运算时延。

6 总结

本文提出格密码中多项式乘法的硬件实现优化方法，能够使用更小的电路面积实现更高的工作频率，对基于格的后量子密码实现具有很高的参考价值。本文给出的蝶形运算模块流水线实现能够满足不同的时序要求，包括低面积、均衡型和高性能实现。其中低面积实现可以达到 150MHz 以上的频率要求，且占用电路资源少；均衡型实现提供较好的性能与电路面积权衡，能够满足 250MHz 以上的中高频率要求；高性能实现能够提供 350MHz 以上的高频率要求。实验结果表明，相比现有的硬件实现，多项式乘法硬件优化技术能够得到 22.8%的效率提升。

参 考 文 献

[1] Shor P W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, 1997, 26(5): 1484-1509

[2] Gidney C, Ekera M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits [EB/OL]. (2019-05-23) [2020-02-21]. https://arxiv.org/abs/1905.09749.

[3] Nejatollahi H, Dutt N, Ray S, et al. Post-quantum lattice-based cryptography implementations: A survey. ACM Computing Surveys, 2019, 51(6): 1-41

[4] Bos J, Costello C, Ducas L, et al. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE // Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2016: 1006-1018

[5] 张平原, 蒋瀚, 蔡杰, et al. 格密码技术近期研究进展. 计算机研究与发展, 2017, 54(10): 2121-2129

[6] Lyubashevsky V, Peikert C, Regev O. On Ideal Lattices and Learning with Errors over Ring. Journal of the ACM, 2013, 60(6): 43

[7] Pöppelmann T, Güneysu T. Towards Efficient Arithmetic for Lattice-Based Cryptography on Reconfigurable Hardware // Proceedings of LATINCRYPT'12. Berlin Heidelberg: Springer, 2012: 139-158

[8] 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. 数据采集与处理, 2019, 34(4): 689-696

[9] Alkim E, Ducas L, Pöppelmann T, et al. Post-quantum key exchange—a new hope // Proceedings of the 25th USENIX Security'16. Austin, TX, USA: USENIX Association, 2016: 327-343

[10] Ducas L, Durmus A, Lepoint T, et al. Lattice Signatures and Bimodal Gaussians // Proceedings of CRYPTO'13. Berlin: Springer, 2013: 40-56

[11] Basu K, Soni D, Nabeel M, et al. NIST Post-Quantum Cryptography- A Hardware Evaluation Study [EB/OL]. (2019-1-17) [2020-03-15]. https://eprint.iacr.org/2019/047.pdf.

[12] Pöppelmann T, Ducas L, Güneysu T. Enhanced Lattice-Based Signatures on Reconfigurable Hardware // Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin Heidelberg: Springer, 2014: 353–370

[13] Roy S S, Vercauteren F, Mentens N, et al. Compact Ring-LWE Cryptoprocessor // Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin Heidelberg: Springer, 2014: 371-391.

[14] Oder T, Guneysu T. Implementing the NewHope-Simple Key Exchange on Low-Cost FPGAs // Lecture Notes in Computer Science, vol 11368. Cham.: Springer, 2017: 128-142.

[15] Poppelmann T. Efficient implementation of ideal lattice-based cryptography [D]. Bochum: Ruhr-University Bochum, 2017

[16] Barrett P. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor // Proceedings of CRYPTO'86. Berlin: Springer, 1987: 311-323

[17] Liu Z, Seo H, Roy S S, et al. Efficient Ring-LWE Encryption on 8-Bit AVR Processors // Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin Heidelberg: Springer, 2015: 663-682

[18] Xing Y, Li S. An Efficient Implementation of the NewHope Key Exchange on FPGAs. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(3): 866-878

[19] Banerjee U, Ukyab T S, Chandrakasan A P. Sapphire: A Configurable Crypto-Processor for Post-Quantum Lattice-based Protocols. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019, 2019(4): 17-61

[20] Montgomery P L. Modular Multiplication Without Trial Division. Mathematics of Computation, 1985, 44(170): 519-521

[21] Jati A, Gupta N, Chattopadhyay A, et al. SPQCop: Side-channel protected Post-Quantum Cryptoprocessor [EB/OL]. (2019-06-30) [2020-02-21]. https://eprint.iacr.org/2019/765.pdf.

[22] Longa P, Naehrig M. Speeding up the number theoretic transform for faster ideal lattice-based cryptography // Proceedings of International Conference on Cryptology and Network Security. Berlin: Springer, 2016: 124-139

[23] Kuo P-C, Li W-D, Chen Y-W, et al. High performance post-quantum key exchange on FPGAs [EB/OL]. (2017-06-12) [2020-02-21]. https://eprint.iacr.org/2017/690.pdf.

[24] Xilinx. UG949 - UltraFast Design Methodology Guide for the Vivado Design Suite (v2019.1) [EB/OL]. (2019-06-26) [2020-02-21]. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug949-vivado-design-methodology.pdf.

[25] Neng Z, Bohan Y, 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

[26] Fritzmann T, Sharif U, Mullergritschneder D, et al. Towards Reliable and Secure Post-Quantum Co-Processors based on RISC-V // Design, Automation, and Test in Europe. Piscataway, NJ: IEEE, 2019: 1148-1153.