Crystal-Kyber算法的FPGA高效并行优化

吕顺森[1,2],李 斌[1*],翟嘉琪[1,2],李松岐[1,2],周清雷[1]
(1. 郑州大学计算机与人工智能学院,河南郑州 450001;2. 数字工程与先进计算国家重点实验室,河南郑州 450001)

摘 要: 多项式乘法运算制约着基于格的后量子密码在现实中的应用. 为提高后量子密码Crystal_Kyber算法的性能效率,减少运行时间,降低多项式乘法的影响,本文设计了一种新的蝶形运算单元对素模q=3 329的Kyber方案进行优化. 首先,采用16路并行调度新型蝶形运算单元的方式执行算法,缩短了计算周期;其次,使用流水线技术以及改进的K^2RED算法,设计实现新型蝶形运算单元,用于降低资源消耗;最后,利用多RAM的方式存储数据,并且多通道优化RAM,允许数据交替存储在RAM中,提高资源复用率. 实验结果表明,本文优化后的数论变换(Number Theoretic Transform,NTT)、逆数论变换(Inverse NTT,INTT)、点对位相乘(Point-Wise Multiplication,PWM)的效率达到200 MHz,合并执行Kyber效率达到175 MHz,优于其他方案,具有良好的性能.

关键词: 后量子密码;Crystal-Kyber;K^2RED;蝶形运算;多项式乘法;硬件效率
基金项目: 国家自然科学基金(No.61702518)
中图分类号: TP313 **文献标识码:** A **文章编号:** 0372-2112(2024)05-1679-11
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20220523

FPGA Efficient Parallel Optimization of Crystal-Kyber

LÜ Shun-sen[1,2], LI Bin[1*], ZHAI Jia-qi[1,2], LI Song-qi[1,2], ZHOU Qing-lei[1]
(1. *School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, Henan* 450001, *China*;
2. *State Key Laboratory of Digital Engineering and Advanced Computing, Zhengzhou, Henan* 450001, *China*)

Abstract: Polynomial multiplication operations limit the practical applications of lattice-based post-quantum cryptography. In order to improve the performance and efficiency of post-quantum cryptography Crystal_Kyber algorithm, and reduce the running time and reduce the influence of polynomial multiplication,this paper designs a new butterfly operation unit to optimize the Kyber scheme with prime modulus $q=3329$. First of all,the algorithm is executed by 16-way parallel scheduling of the new butterfly operation unit, which shortens the calculation cycle. Secondly, using pipeline technology and improved K^2RED algorithm, the design and implementation of a new butterfly operation unit for reducing resource consumption. Ultimately, the data is stored in the way of multi-RAM, and the multi-channel RAM is optimized to allow data to be stored alternately in RAM and improve the resource reuse rate. The experimental results show that the optimized NTT (number theoretic transform), INTT (Inverse NTT), PWM (point-wise multiplication) efficiency reaches 200 MHz, and the combined execution Kyber efficiency reaches 175 MHz, which is superior to other schemes and has good performance.

Key words: post quantum cryptography; Crystal-Kyber; K^2RED; butterfly arithmetic; polynomial multiplication; hardware efficiency
Foundation Item(s): National Natural Science Foundation of China (No.61702518)

1 引言

随着量子计算机的高速发展,当前传统公钥密码密体制如AES和MD5等受到了严重的挑战. 比如Shor算法能够通过量子计算机在多项式时间内破解大整数分解问题和离散对数问题. 在此背景下,使用经典计算机抵挡量子计算机攻击的后量子密码得到了学术界和工业界的广泛研究. 在2016年,美国国家标准与技术研究院(National Institute of Standards and Technology,NIST)向全球发起了一场后量子密码方案竞赛,旨在在新的量子时代到来之前形成新的公钥密码标准[1]. 基

收稿日期:2022-05-09;修回日期:2022-07-02;责任编辑:王天慧
*通讯作者:李斌

于格的密码算法 Crystal-Kyber[2] 是一个用非对称加密的思想去封装密钥并进行密钥协商的算法，Kyber 支持的安全类别 1、3 和 5 分别相当于 AES-128，AES-192 和 AES-256 的强度，它是 NIST 在经过三轮筛选后得到的后量子密码方案中四个决赛方案之一，得到了学术界的广泛关注.

Kyber 算法由于存在大量的多项式乘法的运算，不仅严重消耗了执行时间和资源，而且也制约了 Kyber 算法本身的执行效率. 在现有的高效方法中，使用数论变换（Number Theoretic Transform，NTT）能够加速晶格密码系统中的核心算法运算. 可以利用硬件 FPGA 来实现基于蝶形操作的 NTT 从而达到降低多项式乘法的计算复杂度[3]的目的. 因此，如何利用 FPGA 对 NTT 进行优化，对 Kyber 算法的性能有着关键的作用.

受到硬件的执行效率和资源消耗的影响，为了达到运行速度和资源的平衡，同时保证算法本身的性能，本文利用 FPGA 采用 16 路并行的方式对 Kyber 算法执行优化操作，并利用改进的 K^2RED 算法对 NTT，逆数论变换（Inverse NTT，INTT），点对位相乘（Point-Wise Multiplication，PWM）中存在的蝶形运算操作进行优化，实现了对 Kyber 算法中存在的大量多项式乘法优化的目的.

本文的贡献如下：

（1）对 K^2RED 算法进行改进，实现一种新型蝶形运算单元，确保高频率的同时降低了硬件资源消耗，达到了功耗平衡的目的.

（2）采用 16 路并行方式对蝶形单元进行调度，提高性能的同时降低时延.

（3）对 PWM 调度进行设计，使新型蝶形运算单元在支持 NTT 和 INTT 运算的基础上，支持 PWM 运算.

（4）优化多 RAM，在有限的空间资源下完成对 Kyber 多项式乘法的多路并行运算，提高了空间资源利用率.

2 相关工作

本节介绍了贯穿全文的符号，简单介绍了 Kyber 算法，描绘了 NTT、INTT、PWM 算法的具体流程.

2.1 符号定义

整数环用 $\boldsymbol{Z}$ 表示，设置整数环模 q 的商环 $\{0,1,\cdots,q-1\}$ 使用 $\boldsymbol{Z}_q$ 表示. $\mathcal{R}_q=\dfrac{\boldsymbol{Z}_q[x]}{X_n+1}$ 是定义在 $\boldsymbol{Z}_q=\boldsymbol{Z}/q\boldsymbol{Z}$ 域中整系数多项式环. 其中，X_n+1 表示的最大维度数是 $n-1$.

Kyber 算法是基于错误学习问题（Learning With Error，LWE）的算法[4]，后量子安全等级见表 1. 在 NIST 第三轮公布的标准中，Kyber 的素模 q 的值由第一轮的 7 681 减少到了 3 329. 由于 Kyber 中大量的多项式乘法，如果采用公式法计算多项式，会导致运算时间的增加，所以在 Kyber 官方提供文档中采用了 NTT、INTT、PWM 变换算法来简化多项式计算.

表 1 Kyber 算法参数与安全等级

算法	NIST 等级	n	k	q
Kyber512	1	256	2	3 329
Kyber768	3	256	3	3 329
Kyber1024	5	256	4	3 329

NTT 是在快速数论变换（Fast Fourier Transform，FFT）数论基础上得到的一种能够对多项式乘法进行加速优化的算法，其是在整数环 $\boldsymbol{Z}_q$ 上进行的数论变换. 多项式的 NTT 运算是将一些特殊值代入多项式中，使多项式由系数表示法转换为点值表示法. 对多项式（1），a_i 在经过 NTT 变化后得到的新的多项式如式（2）所示：

$$A(x)=\sum_{i=0}^{n-1}a_i x^i,\ a_i\in\boldsymbol{Z}_q \tag{1}$$

$$\bar{a}_k=\sum_{i=0}^{n-1}a_i\omega^{ik}\ \bmod q \tag{2}$$

其中 ω 被称为旋转因子，是模 q 的 n 次本原单位根，且 $\omega\in\boldsymbol{Z}_q$，满足 $\omega^n\equiv 1\ (\bmod q),\omega^i\neq 1\ (\bmod q)\ \forall\, i<n$；素模 q 形式一般为 $q=\rho_{\text{odd}}\times 2^{\text{pow}}+1$，其中 $2^{\text{pow}}>\rho_{\text{odd}}$，满足条件 $q\equiv 1(\bmod 2n)$；INTT 的操作和 NTT 类似，但在 INTT 中，需要利用 ω^{-1} 来代替 ω 在 NTT 中的作用，而且还需乘以额外的 n^{-1}. INTT 将 $\bar{a}_k$ 转化为 a_i 如式（3）所示：

$$a_i=n^{-1}\sum_{k=0}^{n-1}\bar{a}_k\omega^{-ik}\ \bmod q \tag{3}$$

经过 NTT 之后，多项式 A 和 B 乘积表示为 $C=\text{INTT}((\text{NTT}(A)\,o\,(\text{NTT}(B))$，$o$ 表示多项式的点对位相乘 PWM.

算法 1、算法 2、算法 3 给出了 NTT、INTT、PWM 在 Kyber 算法中的具体迭代流程. ω=17 是素模 $q=3\,329$ 在 $n=256$ 时的第一个本原单位根，$\text{br}_{t-1}(f)$ 表示对 f 进行 $t-1$ 位的比特逆序操作.

在官方文档说明中，利用 NTT 和 INTT 可以将原有的 256 项多项式拆分为两个独立计算的 128 项多项式. 在 PWM 算法中，只需要进行 128 次二次多项式的乘法运算即可，相较于传统的二项式乘法计算所需的时间更短. 但在算法 1 和算法 2 中 NTT 和 INTT 存在着大量的模加、模减和模乘. 算法 3 中，PWM 的模加和模乘运算相较于 NTT 和 INTT 更为复杂. 在硬件设计时，大量的取模运算必然会出现繁杂的除法操作，不仅会加剧硬件资源消耗，而且会导致算法整体运算时间的上升. 为了提高算法效率，需要对取模操作进行化简优化，一般是采用其他方法替换该操作.

算法1　NTT

输入: $A(x)=\sum_{i=0}^{n-1}a_i x^i,\ a_i=\{a_0,a_1,\cdots,a_{n-1}\},\ \omega,\ q$

输出: $\bar{A}(x)=\sum_{i=0}^{n-1}\bar{a}_i x^i,\bar{a}_i=\{\bar{a}_0,\bar{a}_1,\cdots,\bar{a}_{n-1}\}$

1. FOR $i=\left((\log_2 n)-1\right)$ TO 1 do
2. $m=2^i\ r=0$
3. FOR $k=0$ TO $n-1$ BY m DO
4. $W=\left(\omega^{\mathrm{br}_{(\log_2 n)-1}(r)}\right) \bmod q$
5. FOR $j=0$ TO $m/2-1$ DO
6. $u=a_{j+k}\ v=a_{j+k+m/2}$
7. $t=(W\times v) \bmod q$
8. $e=(u+t) \bmod q \quad o=(v-t) \bmod q$
9. $a_{j+k}=e \qquad a_{j+k+m/2}=o$
10. END FOR
11. $r=r+1$
12. END FOR
13. END FOR
14. RETURN $\bar{A}(x)$

算法2　INTT

输入: $\bar{A}(x)=\sum_{i=0}^{n-1}\bar{a}_i x^i,\bar{a}_i=\{\bar{a}_0,\bar{a}_1,\cdots,\bar{a}_{n-1}\}$

输出: $A(x)=\sum_{i=0}^{n-1}a_i x^i,\ a_i=\{a_0,a_1,\cdots,a_{n-1}\}$

1　$k=0$
2. FOR $i=\left(\log 2^n-1\right)$ TO 1 DO
3. $m=2^{(\log_2 n)-i}$
4. FOR $s=0$ BY m TO 2^i DO
5. $u=\bar{a}_j,\ v=\bar{a}_{j+k},\ W=\omega^{\mathrm{br}_{(\log_2 n)-1}(k)+1} \bmod q$
6. $e=(u+v) \bmod q,\quad o=(u-v)W \bmod q$
7. $\bar{a}_j=(e/2) \bmod q,\quad \bar{a}_{j+k}=(o/2) \bmod q$
8. END FOR
9. $k=k+1$
10. END FOR
11. RETURN $A(x)$

2.2　国内外研究现状

Botros 等[5]研究了 Kyber算法的软件实现,提出了在 Cortex-M4 上对 Kyber算法进行了优化,但是采用软件方式实现 Kyber核心运算,不仅消耗时间,而且得到的频率较低,无法满足高性能的需求. 也有诸多学者提出使用硬件的方式来对 Kyber算法进行优化. Basu 等[6]基于 HLS的硬件设计方法将 Kyber算法映射到 FPGA 上进行实现,和软件实现相比,该方法性能有了很大的提升,但是 Basu 等人只是对算法进行比较,并没有对算法进行深度优化,使用的硬件资源过大,频率较低. Alkim

算法3　PWM

输入: $\bar{A}(x)=\sum_{i=0}^{n-1}\bar{a}_i x^i,\bar{a}_i=\{\overline{a}_0,\bar{a}_1,\cdots,\bar{a}_{n-1}\}$

$\bar{B}(x)=\sum_{i=0}^{n-1}\bar{b}_i x^i,\bar{b}_i=\left\{\bar{b}_0,\bar{b}_1,\cdots,\bar{b}_{n-1}\right\}$

输出: $\bar{C}(x)=\sum_{i=0}^{n-1}\bar{c}_i x^i,\ a_i=\left\{\bar{c}_0,\bar{c}_1\ldots,\bar{c}_{n-1}\right\}$

1. FOR i FROM 0 TO $2^{(\log_2 n)-1}$ DO
2. $W=\left(\omega^{1+\mathrm{br}_{\log_2 n-1}(i)}\right) \bmod q$
3. $a_0=\bar{a}[2i],a_1=\bar{a}[2i+1]$
4. $b_0=\bar{b}[2i],b_1=\bar{b}[2i+1]$
5. $\bar{c}[2i]=\left(a_0b_1+a_1b_0\right) \bmod q$
6. $\bar{c}[2i+1]=\left(a_1b_1W+a_0b_0\right) \bmod q$
7. END FOR
8. RETURN $\bar{C}(x)$

等[7]使用了集成了有限域指令集扩展的RISC-V处理器来加速Kyber轻量级架构中的多项式乘法,相较于软件实现提速85%,但是Alkim等使用处理器的方式优化算法,资源消耗很大,得到的频率较差,在实际应用中的场景受限. Huang等[8]、Dang等[9]、Xing等[10]提出了Kyber的纯硬件结构体系,但是Huang等的工作依赖组件之间的Block RAM原语来执行算术任务和存储中间结果,Xing等采用基于时域和频率的结合型蝶形单元来实现NTT操作,虽然在面积和时间上有了很好的提升,但是频率还可以进一步优化. Chen等[11]提出一种用于计算NTT和点对位乘法的Kyber处理器,达到了加快NTT计算频率目的,但并没有和算法性能达到平衡,消耗了过多资源. Roy等[12]利用传统的二次复杂度多项式乘法算法设计了一种灵活的硬件结构,但该硬件结构加重了资源负担,没有对Kyber算法起到很好的优化作用.

国内对后量子密码的的研究和国外相比较晚,但发展的很快. 周朕等[13]针对国内研发的Aigis-enc[14]设计了一种基于NTT算法实现环上多项式运算的紧凑硬件架构,该硬件架构采用Montgomery算法对关键模乘部分进行优化,算法整体性能得到了提高. 崔益军等[15]针对NTT优化设计了迭代型和流水线硬件结构,他们采用了文献[10]提出的修改的Barrett约减算法对NTT的关键部分进行优化,达到了提高硬件的执行效率的目的. Feng等利用Stockham FFT算法设计了一种多路径且高并行度的NTT算法[16],并且通过利用大量蝶形单元并行调度提高NTT模块的运行速度.

综上分析可知,利用FPGA对Kyber算法进行优化时,为了提高算法的性能. 需要对Kyber算法中的关键部分操作进行深度优化,化繁为简;而如果要提高算法的运行速度,增加并行度的同时提高各模块之间的数

据处理能力是一个很好的选择. 但大部分方案中在对NTT进行优化的时候,由于硬件资源问题,舍弃了高并行,导致NTT计算周期长. 而有些方案在采用高并行的方式加速算法时,也存在消耗过多的硬件资源的问题. 且部分方案为了进一步提高算法性能仅执行了NTT和INTT运算,舍弃了PWM运算. 因此如何做到满足算法性能的同时与硬件资源达到平衡,是当前学术界在格密码方案硬件实现的一大研究热点.

本文从NTT、INTT、PWM各计算阶段着手,设计实现一种新型蝶形单元,使得该蝶形单元支持NTT、INTT、PWM运算,实现了对Kyber算法关键部分的化简,并采用高度并行方式进行调度,提高计算速度的同时实现了功耗平衡.

3 Kyber优化设计

3.1 整体架构

为在提高Kyber算法运算速度的同时,增强硬件整体的灵活性和可扩展性,本文首先对NTT、PWM、INTT和Kyber多项式乘法进行16路并行执行,设置了16个蝶形单元匹配执行运算操作. 其次,采用多RAM的方式,对执行运算的多项式参数和每轮蝶形操作迭代后的结果进行存取;对于旋转因子ω,采用预训练的方式提前计算好之后保存在ROM中. 最后,对旋转因子ω的读取控制、RAM存取地址的索引控制和蝶形单元的输入输出都由控制逻辑来完成. 图1显示了FPGA整体结构.

如图1所示,16路蝶形运算操作并行执行,蝶形单元内部采用流水线技术进行实现;为了同步RAM的存取地址索引,在控制单元内将Kyber执行蝶形单元进行的轮数与循环次数和RAM的存储空间地址进行绑定,减少了程序的复杂性. 在PWM阶段,因为K^{2}RED算法的特殊性,蝶形单元输出的最终结果由T值代替了O值. 为了匹配16路蝶形运算,需要在每轮蝶形运算时从ROM中取出该轮蝶形运算对应的16个旋转因子. 硬件整体采用松耦合架构展开,可以根据输入的控制信号的不同,自主独立完成NTT、INTT、PWM运算.

3.2 流水线型蝶形单元硬件设计

16路蝶形单元块同步进行运算,虽然显著提升了运算速度,但也导致了硬件资源消耗过大. 为进一步节省资源,首先,在蝶形单元内部只设计了一个模加器和模减器,并采用组合逻辑的方法实现模加减. 其次,利用改进的K^{2}RED算法对Kyber中耗时的求模运算进行化简. 由于需要在使用改进的K^{2}RED算法前对传递进来的参数进行相乘,考虑到DSP块在最新的FPGA设备上大量可用,为了充分利用FPGA资源,选择使用DSP作为基础乘法器,从而实现快速高效的系数级乘法计算,进一步降低计算时间. 最后,因为NTT、INTT、PWM

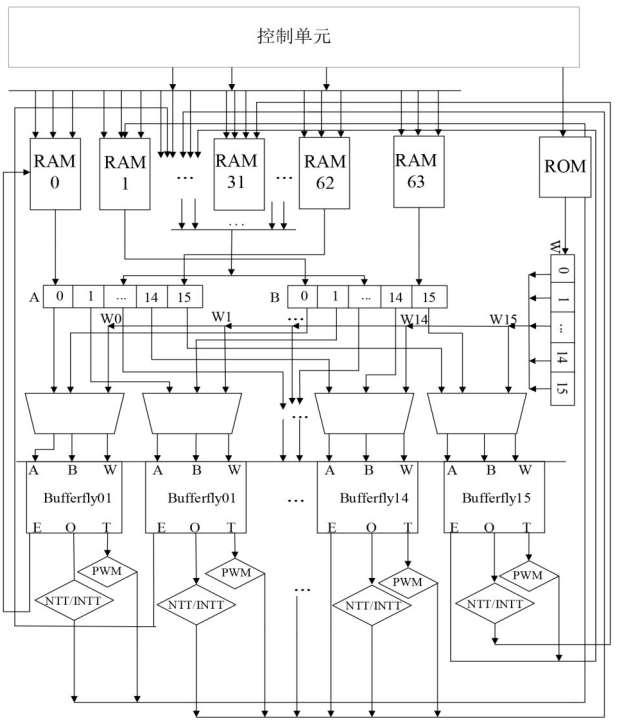


图1 FPGA整体结构

算法执行流程的不同,在不同的阶段内,保存在RAM块中的结果也不同,为了减少顶层逻辑的复杂度,以及进一步降低路径延时:第一,在蝶形运算单元内部设计了多个缓存器来缓存中间结果;第二,在顶层设置仲裁信号,根据不同的信号执行不同的流程. 该蝶形单元硬件结构和其各阶段流水线执行流程图如图2和图3所示.

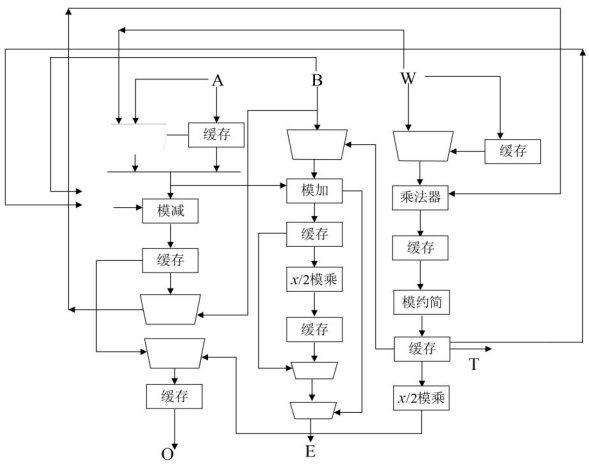


图2 蝶形运算单元硬件结构

在图2中,蝶形单元内部只设置了一个模加器、模减器和模约简器. 它还有用于同步输出系数的缓存寄存器和用于可重构的多路复用器. 该蝶形单元以A、B、W为输入进行运算操作,在对应的阶段内生成结果值E、$O(T)$.

如图3所示,在蝶形单元内部,NTT、INTT、PWM在不同的阶段内执行不同的流程. 首先,为了使算法执行流程和硬件时钟相匹配. 由于改进的K²RED算法的特殊性,需要对PWM调度进行改进. 其次,为了进一步节省资源,缩短硬件的计算周期,对模加减、$x/2 \bmod q$进行进一步的优化. 最后,为了匹配不同的流程,在时钟、RAM索引调度、控制单元上进行合理的设计,从而得到正确的结果.

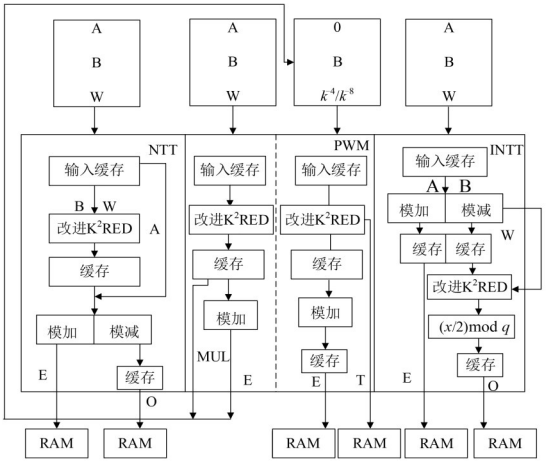


图3　流水线执行流程

本文采用下文所述方法对K²RED算法进行改进,使该算法能够支持NTT、INTT、PWM运算;然后对PWM调度进行设计,使其能够匹配改进的K²RED算法;在此基础上对模加减、$x/2 \bmod q$进行优化;最后,对多RAM存取进行设计,进一步增加空间资源复用.

3.2.1　改进的K²RED算法

针对蝶形单元运算中最耗时的求模运算操作,本文采用文献[3]推荐的K²RED算法对该操作进行优化. 之所以使用该算法,是因为K²RED算法对任意整数C的求模运算转化为了对C的部分bit拆分移位操作得到整数D,使$D=k^2C \bmod q$. 在该流程中,除法运算被舍弃,而且硬件中的bit移位操作实现简单且不消耗资源. 在Kyber算法中,得益于q=3 329,其形式可以表示为$q=k\times 2^m+1$,其中k=13,m=8,基于该性质,K²RED算法可以在蝶形单元中被使用对求模运算进行化简.

文献[3]设计的K²RED模块只支持NTT和INTT运算,且涉及了k的乘法操作. 本文对该算法进行改进和优化,舍弃K²RED算法中耗时的乘法操作,而只选择使用bit判断、移位、加法和减法,进一步提高了算法在硬件中的执行速度,降低了资源消耗,并使其在支持NTT和INTT运算的同时也支持PWM运算. 改进的K²RED算法的具体流程如算法4所示.

改进的K²RED算法利用输入数本身的bit移位操作代替高复杂度的除法来实现求模运算,在节省硬件资源的同时也缩短了计算周期.

算法4　改进的K²RED算法

输入: $C=A\times B$

输出: $R=k^2(A\times B) \bmod q$

1. $c_1=C[7:0]; c_2=C[23:8]$
2. $c_3=(c_1\ll 3)+(c_1\ll 2)+c_1-c_2$
3. $c_4=c_3[7:0]; c_5=c_3[15:8]$
4. $c_6=(c_3\ll 3)+(c_4\ll 2)+c_3-c_4$
5. $c_7=(c_5[7:5]==0)?\ \#c_6:c_6+256$
6. $c_8=c_7-3329$
7. $R=(c_7[12]==0)?\ \#((c_8[12]==0)?(c_8:c_7)):(c_7+3329)$
8. RETURN R

3.2.2　PWM调度设计

在设置PWM模块硬件结构时,由PWM算法可知,需要计算a_0b_1、a_1b_0、a_1b_1W、a_0b_0,共5次模乘和2次模加. 而且由算法3可知,PWM算法流程的第3~6步和NTT、INTT类似,所以都可以采用K²RED算法对求模操作进行化简. 但是由于K²RED算法返回值$D=k^2C \bmod q$,需要消除k^2所带来的影响后才能进行相乘操作,从而得到正确的$\bar{c}[2i]$和$\bar{c}[2i+1]$.

在对PWM调度进行设计的时候,我们先在前5个时钟完成PWM算法的第3~6步,得到被k^2影响的结果值,在后续5个时钟内消除k^2对结果值所带来的影响. 经过研究发现,需要在特定的时钟内将蝶形单元输出值与模逆值$k^{-4}=0x549$,$k^{-8}=0x8f5$进行结合后再次进行蝶形操作. 一共需要10个时钟,图4给出了PWM阶段具体调度流程.

如图4所示,在前5个时钟完成PWM算法的3~6的流程,然后在后5个时钟消除k^2的影响,其中在第9个时钟的时候,因为需要使用的ADD值是K²RED运算后经过模加后得到的,所以需要等待1个时钟.

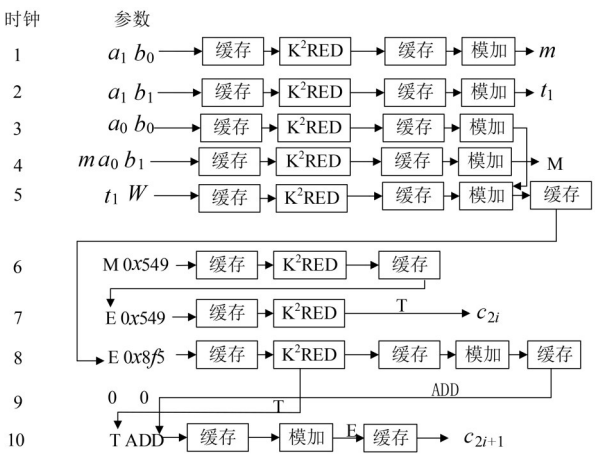


图4　PWM调度流程图

3.2.3 模加减与 $x/2$ 模乘

由图2可知，NTT、INTT、PWM都在执行阶段内使用到了模加减，蝶形单元结果值E、$O(T)$和模加减息息相关. 模加减对蝶形运算起到重要作用. 一个良好的模加减，不仅能加速算法的执行时间，而且能进一步降低蝶形运算单元的资源消耗. 本文对模加减采用组合逻辑实现来避免除法，具体流程是判断模加减的结果值和素数 q 的关系后进行处理. 其硬件结构如图5、图6所示.

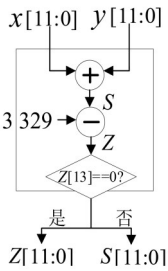


图5 模加硬件结构

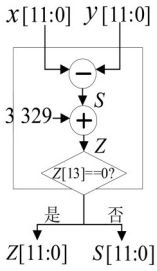


图6 模减硬件结构

对于$x/2 \bmod q$操作，因为涉及取模和乘1/2操作，本文采用了Zhang等[17]提出的方法对$x/2 \bmod q$进行优化. 具体是由于q为常数3 329且是奇素数的性质，对于$x/2 \bmod q$操作，可以通过移位、加法和逻辑与操作简化乘法和取模预算，进一步节省资源开销，其计算式如式(4)所示. 根据公式所设计的硬件结构如图7所示，G表示输入数x最低比特位$x[0]$的10倍扩展.

$$\frac{x}{2} \bmod q=(x \gg 1)+x[0]\&\left(\frac{q+1}{2}\right) \tag{4}$$

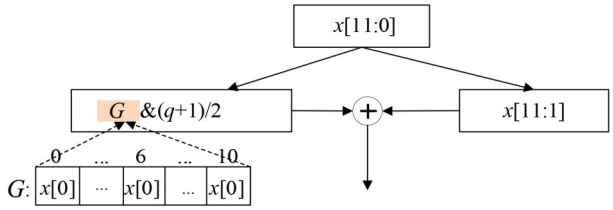


图7 $x/2 \bmod$优化结构图

3.3 多RAM存取设计

在Kyber算法流程中，需要将多项式A和B的NTT运算结果值执行PWM运算，然后对PWM结果执行INTT运算. 考虑到对于N个多项式点，需要执行$\log_{2^N}$的NTT和INTT，为了便于后续操作，需要将每轮蝶形单元运算得到的结果保存在RAM中. 在该过程中，因为涉及多个多项式的参数存取，而且需要防止执行蝶形操作的参数混乱，所以对保存结果的RAM设置唯一的存取地址.

图8所示的8点多项式蝶形运算，既要保证每个阶段内进行蝶形运算取得的多项式参数正确，也要确保蝶形运算中间结果必须存储在指定的索引位置上，从而防止索引地址混淆. 而在NIST第三轮的Kyber算法中，进行蝶形运算的每个多项式参数有256个，因为NTT和INTT的原因，可以将一个256项的多项式拆分为了两个独立计算的128次多项式进行蝶形运算，一共需要进行$\log_{2^N}=7$轮蝶形运算才能最终结果. 对多RAM的索引控制更为复杂，所需要的RAM块更多. 为了进一步优化FPGA的空间资源，减少资源消耗，以下给出本文对多RAM的优化方法.

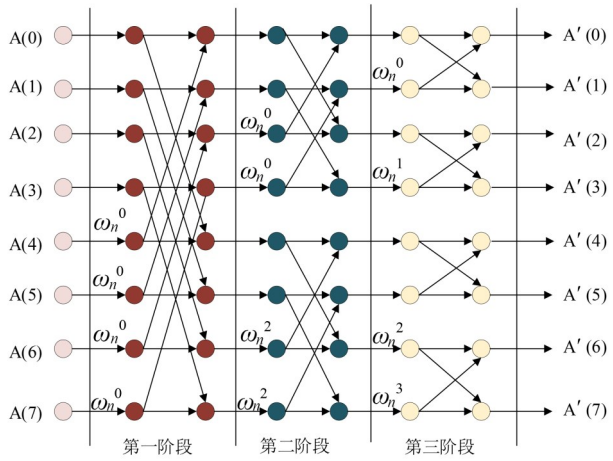


图8 8点多项式蝶形运算

3.3.1 多RAM优化

RAM是FPGA提供的紧凑的真正双端口内存库，方便存储大量数据. 在本文中为了在一个时钟内执行16次蝶形运算，一共设计了16×4个RAM块用来存储，单个RAM块需要有$(256\times2)/64=8$个存储空间来存储多项式参数. 前32个RAM块存储第一个多项式的参数，后32个RAM块存储第二个多项式参数. 在执行NTT和INTT的时候，在同一时钟内分别从各自32个RAM块中选择取出执行蝶形操作的第$2\times i$和$2\times i+1(0\leqslant i\leqslant 16)$的多项式参数进行16路并行执行NTT和INTT. 如此，16个蝶形运算单元可独立并行存取参数并执行. 图9给出了NTT阶段首轮初始化参数存储RAM示意图. 如图9所示，将256项多项式平分为上下128项，利用偶数组RAM存储上128项，奇数组RAM存储下128项. NTT首轮循环开始后，从第一部分偶数组中取出$a_0,a_1,\cdots,a_{15}$与第一部分奇数组$a_{128},a_{129},\cdots,a_{143}$分别组合进行蝶形运算. 随后，对第二部分按照相同方式组合进行蝶形运算，直到第一轮NTT结束. 同组相同部分之间多项式索引地址连续. 单RAM内相邻部分多项式索引地址相差16.

NTT、INTT、PWM合并执行Kyber多项式乘法运算

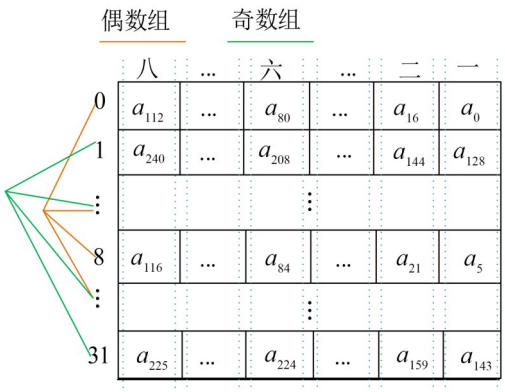


图 9　NTT 初始参数存储

时,单 RAM 存储示意图如图 10 所示. 对于多 RAM,将 64 个 RAM 块平分成两组,每组按照索引地址平分成集合 1 和集合 2. 将初始化要执行 Kyber 多项式乘法运算的多项式 $A(x)$ 和 $B(x)$ 分别保存在两个 RAM 分组的集合 1 中,保存完毕后从集合 1 取出多项式参数执行 NTT,将 NTT 运算结果保存在集合 2 中;在 PWM 阶段取出集合 2 中元素进行蝶形运算,并将结果保存在集合 1 中,等待 INTT 阶段的执行. 多 RAM 存储流程如图 11 所示.

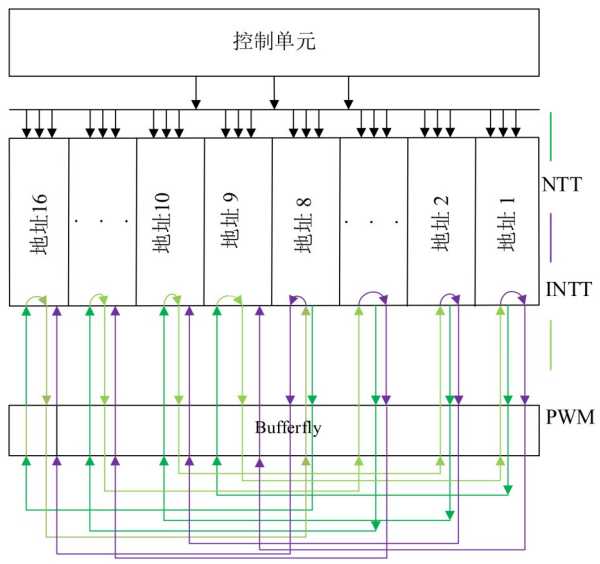


图 10　单 RAM 存储图

最后,由于 NTT、INTT、PWM 在执行相应轮数时对旋转因子 ω 值的参数固定,因此本文在对旋转因子 ω 进行预处理后,按照一定的顺序将其存放在 BROM 中. 具体来说,地址 0~38 存放执行 NTT 的旋转因子 ω. 地址 39~77 存放执行 INTT 的旋转因子 ω'. 地址 78~85 存放执行 PWM 的选择因子 ω^*. 在每个地址存放 16 个旋转因子,对应并行执行的 16 个蝶形运算单元. ROM 存储结构如图 12 所示.

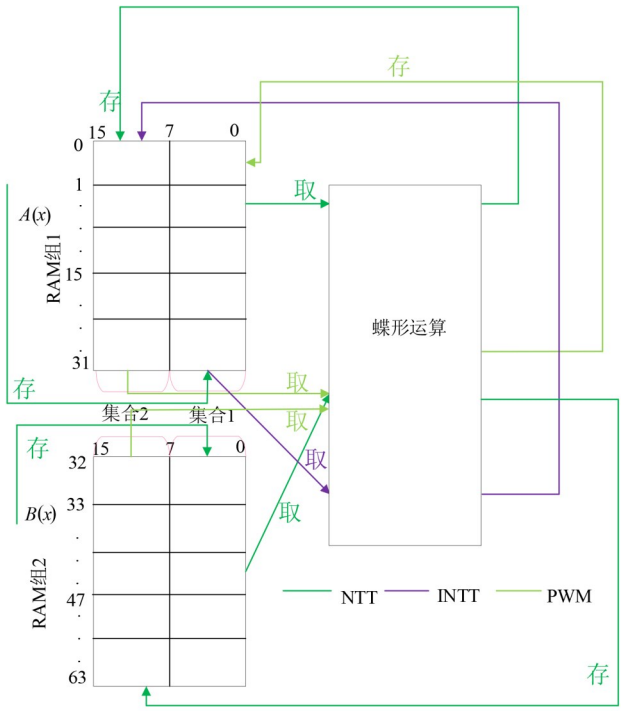


图 11　多 RAM 存储流程图

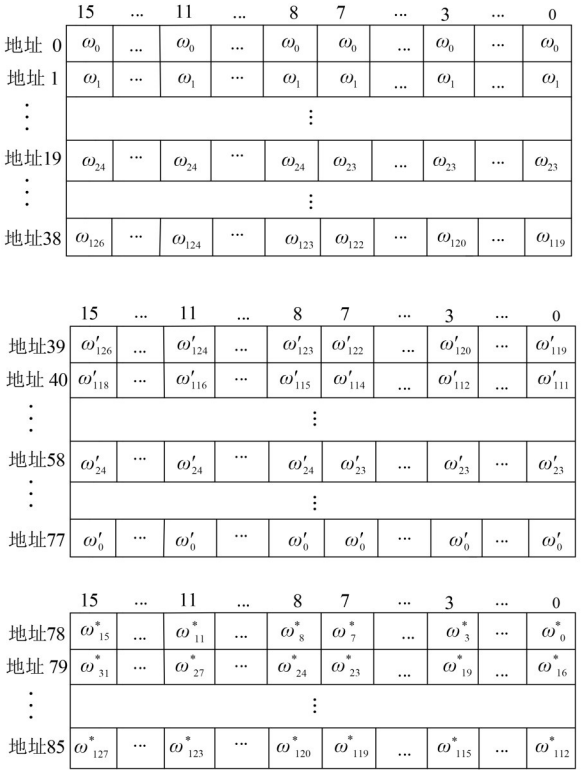


图 12　ROM 存储图

3.4　RAM 块索引地址控制

从第 3.3 节的分析可知,NTT、INTT、PWM 三个阶段中每轮进行蝶形运算的多项式参数不同,但存储在 RAM 块中的每个多项式参数地址唯一且固定,且需要

将每轮蝶形运算执行后得到的结果E、$O(T)$保存在RAM中指定索引位置上,以便下一个阶段读取,保证蝶形运算得到的结果值的正确性.

在RAM地址块设计时采用1个地址red对RAM块进行读取地址控制,两个地址wre和wro对E、$O(T)$进行存储地址控制. 在设置rae、wre、wro的参数时,考虑到NTT、INTT、PWM经过7轮蝶形运算完成存取,而且RAM块中存储空间最高最有16个,令rae、wre、wro的后三比特位由NTT、PWM、INTT的经历的轮数和循环次数决定. 另外,由于RAM中高8位和低8位的存取是循环进行的. 因此对rae、wre、wro的最高比特位按照每7轮进行一次交替取反. 算法5和算法6介绍了NTT情况下的rae、wre、wro的具体计算流程.

算法5 RAM读取地址算法

输入: 蝶形运算阶段stage[2:0], 循环次数loop[2:0]

输出: RAM 读地址 rae[2:0]

1. 初始化 rae[2:0] = 0

2. IF $(\text{stage} > 1)$

   rae[2:0] = loop

3. ELSE

   $\text{rae}[2:0]=\big((\text{loop}\gg 1)\&\big((1\ll(2-\text{stage}))-1\big)\big)+(\text{loop}[0]\ll(2-\text{stage}))+\big((\text{loop}\gg(3-\text{stage}))\ll(3-\text{stage})\big)$

4. RETURN rae[2:0]

如算法5所示,在步骤2中,当蝶形运算阶段大于1时,RAM的读地址直接由循环次数决定. 否则,如步骤3所示,读地址需要根据当前stage和loop进行额外计算,从而决定当前RAM的读地址.

算法6 RAM写入地址算法

输入: 蝶形运算阶段 stage[2:0],循环次数 loop[2:0]

输出: RAM 写入地址 wre[2:0],wro[2:0]

1. 初始化 wre[2:0] = 0; wro[2:0] = 0

2. IF $(\text{stage} > 2)$

   wre[2:0] = loop; wro[2:0] = loop

3. ELSE

   $\text{wre}[2:0] = (\text{loop}\gg 1) + \big((\text{loop}\gg(3-\text{stage}))\ll(2-\text{stage})\big);$

   $\text{wro}[2:0] = (\text{loop}\gg 1) + \big((\text{loop}\gg(3-\text{stage}))\ll(2-\text{stage})\big) + \big(1\ll(2-\text{stage})\big);$

4. RETURN wre[2:0]; wro[2:0]

如算法6中,因为RAM是先读后写,而且需要等待3个时钟. 所以在步骤2中,只有stage大于2的时候,wre和wro的值由loop决定. 而在其他情况也是根据当前stage和loop决定.

考虑到算法5和算法6在一定阶段的特殊性,在硬件设计时,针对RAM读写索引启动了延时缓存策略. 具体在NTT蝶形运算的前3轮,由于此时NTT的读写不是按序进行的,在每轮NTT蝶形结束运行时等待2个时钟,在4~7轮,此时RAM读写顺序一致,无需进行中间等待. 在INTT模块,在前三轮蝶形运算时每轮等待1个时钟. 对于PWM模块,由于PWM阶段每轮运算都需要经过多个状态,所以有足够的时钟使得读写地址匹配. 图13显示了NTT阶段的RAM读写索引控制.

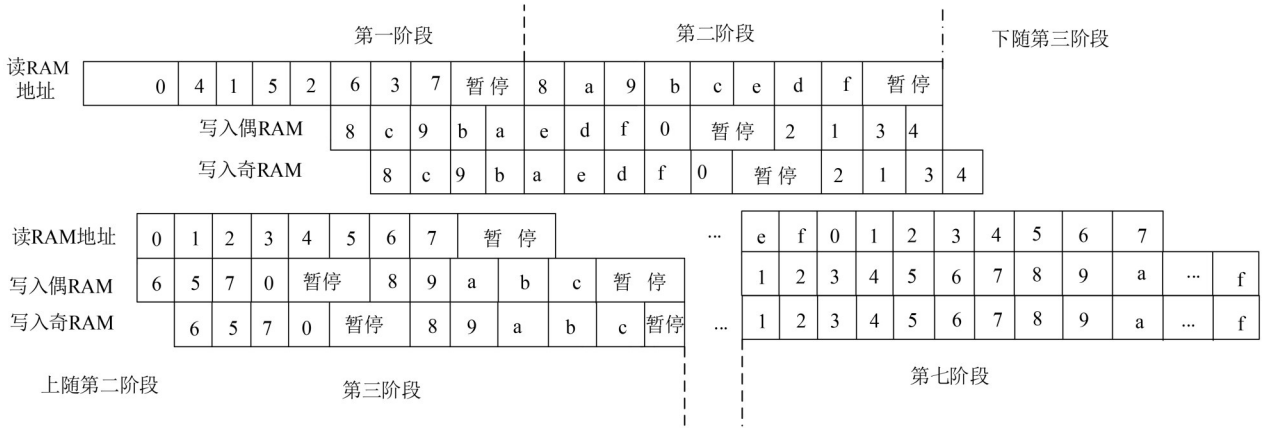


图13 RAM读写索引控制

为解决多蝶形单元并行计算地址冲突问题,每2个RAM对应1个蝶形运算,将32个RAM块分配给其对应的16个蝶形单元,各蝶形单元独立执行. 其对应的RAM块内按照算法5、算法6对蝶形运算系数和运算结果进行读取.

4 实验结果分析

4.1 实验结果与分析

为了验证本文所提出的优化方案,我们在ZYNQ-XC7Z035FFG676-2上实现了该程序. 表2展示了Kyber算法中各模块的资源使用情况. 表3给出了本方案中

NTT、INTT、PWM 和 Kyber 总体的吞吐量和资源使用情况.

表 2　Kyber 算法模块资源使用表

模块	实现方式	LUT	FF	BRAM	DSP
地址控制单元	流水线	3 351	134	0	0
蝶形运算单元	流水线	255	132	0	1
多RAM存储	并行	256	0	32	0
ROM存储块	串行	0	0	2.5	0
Kyber实现	串并混合	8 053	3 321	34.5	16

表 3　Kyber 性能表

模块	LUT	FF	BRAM	DSP	频率/MHz	功耗/W
NTT	4 981	3 063	18.5	16	200	0.690
INTT	4 418	2 797	18.5	16	200	0.661
PWM	4 470	2 790	18.5	16	200	0.670
Kyber	8 053	3 321	34.5	16	175	0.774

从表 2 分析可知,地址控制单元由于需要计算蝶形运算轮数、本轮迭代次数、RAM 和 ROM 的读写地址、控制信号以及读写等待信号等,所以消耗了更多的 LUT. 蝶形运算单元占用了较少的资源实现了模加、模减、模乘等运算,且仅需要 1 个 DSP 便可完成乘法运算,表明使用改进 K^2RED 优化蝶形运算单元的正确性. 64 个 RAM 块仅使用了很少的 LUT 和 BRAM 资源便完成了 Kyber 多项式的参数存储. BROM 块使用了少量的 BRAM 资源,便完成了对 Kyber 多项式乘法所需的旋转因子 ω 的存储. 所使用的 LUT 占用芯片 LUT 资源的 4.68%,FF 占用比例为 0.97%,BRAM 资源占用比例为 6.90%,DSP 资源占用比例为 1.78% 便完成了 Kyber 多项式乘法在 FPGA 的实现,达到了资源上的优化目的,证明了本文优化思想的正确性.

4.2　方案对比

为了更加方便的比较性能优劣,本文引入了一个新的计量值/参数/指标——单元性能(Unit Perfokmance,UP),客观反应了资源消耗和算法性能之间的关系. UP 值越高,表明在有限的硬件资源条件下,优化后的算法性能和资源利用率越好. UP 计算方式如下所示.

$$\mathrm{UP}=\frac{\text{频率}\times 10^3}{\mathrm{FF}+\mathrm{LUT}} \tag{5}$$

在表 4 中,将本文算法与其他 Kyber 算法经过 FPGA 优化后的资源消耗、频率和 UP 等进行了综合对比. 在表 5 中,将本文算法与其他 Kyber 算法在 NTT 阶段经过 FPGA 优化后的资源消耗、频率和 UP 等进行了综合对比.

表 4　FPGA 优化后 Kyber 总体综合对比

方案	设备	LUT	FF	BRAM	DSP	频率/MHz	时间/μs	UP
文献[9]	Artix-7	12 200	12 400	15	8	210	—	8.54
文献[18]	Alveo U250	166 100	46 000	16	138	—	1.97	—
文献[19]	Artix-7	9 508	2 684	35	16	172	0.40	14.1
本文	Zynq-7035	8 053	3 321	34.5	16	175	1.50	15.38

表 4 比较了不同方案优化后的 FPGA 实现的 Kyber 多项式乘法的优化性能. 本文通过改进的 K^2RED 算法优化蝶形操作,并且利用多路并行的方式优化后的 Kyber 算法在较高的频率下使用的资源相比其他文献使用的更少,得到的 UP 更好,资源利用率更好. 在表 4 中,文献[9]虽然得到的频率比本文方案多,但是其消耗的资源更多,而且从 UP 方面的比较来看,本文对资源使用和算法性能的平衡更好. 文献[18]采用 RISC-V 矢量扩展来加速 Kyber 多项式乘法,且更关注算法的执行时间而忽略了硬件资源消耗,增加了电路的复杂度. 在文献[19]中,采用 16 个蝶形运算单元的方式来实现 NTT,但是其采用的模约简方式相较于本文更复杂,导致消耗的资源更多.

在表 5 中,为了进一步比较优化后的算法性能,我们对 Kyber 算法在 FPGA 上实现 NTT 进行了性能比较. 文献[3]采用原版的 K^2RED 算法对 NTT 进行优化,但文献[3]采用了时间换取资源的策略,使用了一个 2×2 蝶型核搭配一个 K^2RED 模块按照完全流水线的方式执行 NTT,相较于本文的 16 个蝶形单元使用的资源必定很少,频率更高. 但其优化后得到的 NTT 吞吐量却和本文近似相同. 在计算周期上,本文可在 0.40 μs 内完成对 NTT 的计算,文献[3]却需要 1.46 μs,比本文所需的时间更长. 综合对比下,本文的性能更好. 文献[20]采用 K-RED 算法对求模运算进行化简并搭配 2×2 蝶型核对 NTT 进行优化,虽然降低了硬件资源消耗,但该方案的频率、执行时间都不如本方案. 文献[21]采用 Barrett 算法对蝶形单元的求模运算进行优化,并放置 32 个蝶形单元执行 NTT. 从表 5 的分析可知,本文利用改进的 K^2RED 算法优化后的蝶形单元的性能相比文献[21]的方案更好,UP 值更高. 文献[22]在 FPGA 内放置 4 个蝶形单元和 8 个乘法器并采用 SAMS2 算法[22]对求模运算进行化简实现 NTT,在综合性能和 UP 等方面都不如本文提出的方案. 文献[23]和文献[5]同样采用 Barrett 模约简来对蝶形运算单元进行优化,但其采用乓乓操作

表5 FPGA优化后NTT总体综合对比

方案	设备	LUT	FF	BRAM	DSP	频率/MHz	时间/μs	UP
文献[3]	Artix-7	801	717	2	4	222	1.46	146.24
文献[20]	Zynq-7000	2 832	1 381	10	8	150	17.44	35.6
文献[21]	Zynq-7035	8 428	3 979	11	32	175	0.25	14.1
文献[22]	Artix-7	4 823	2 901	10	8	153	—	19.8
文献[23]	Artix-7	18 000	5 000	15	6	115	14.8	5.00
本文	Zynq-7035	4 981	3 063	18.5	16	200	0.40	24.86

对系数存取进行优化,虽然提高了NTT的计算速度,但是整体频率偏低,消耗资源过高. 而本方案中实现的NTT在得到高频率的同时也兼顾了资源消耗.

对比Kyber和NTT的整体性能之后可知,本文提出的Kyber的16路并行优化方案在保证了较高的工作频率的同时,也充分利用了芯片的资源,效率更好,具有更优的使用价值.

5 结束语

随着计算机的快速发展,基于格的密码方案在即将到来的量子计算机时代具有优异的研究价值和实用场景. 本文利用改进的K^2RED算法对蝶形运算操作进行优化,并采用流水线技术在硬件上对优化后的蝶形单元进行实现,在此基础上通过16路并行方式完成了对NTT、INTT、PWM以及Kyber多项式的运算操作. 为了进一步优化空间资源的利用率,本文采用多RAM的方式对多项式进行了存取优化. 实验结果证明,本文的实验方案具有较高的实际应用和研究价值.

在未来工作中,将本文对Kyber算法的优化应用到其他格密码方案,是我们研究的重点之一. 并且我们将对PWM阶段继续进行优化,进一步缩短蝶形运算时钟周期.

参考文献

[1] NIST. Post-Quantum cryptography call for proposals[EB/OL].(2017-01-03)[2022-04-15]. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptographt-stan - dardization/Call-for-Proposals.

[2] AVANZI R, BOS J, DUCAS L, et al. Crystals-kyber[R]. Maryland: NIST, 2017.

[3] BISHEH-NIASAR M, AZARDERAKHSH R, MOZAFFARI-KERMANI M. High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography [C]//2021 IEEE 28th Symposium on Computer Arithmetic (ARITH). Piscataway:IEEE, 2021: 94-101.

[4] REGEV O. On lattices, learning with errors, random linear codes, and cryptography[J]. Journal of the ACM, 2009, 56 (6): 1-40.

[5] BOTROS L, KANNWISCHER M J, SCHWABE P. Memory-efficient high-speed implementation of Kyber on Cortex-M4[C]//International Conference on Cryptology in Africa. Springer: Cham, 2019: 209-228.

[6] BASU K, SONI D, NABEEL M, et al. Nist post-quantum cryptography-a hardware evaluation study[J]. IACR Crgptology Eprint Archive, 2019, 47: 1-16.

[7] ALKIM E, EVKAN H, LAHR N, et al. ISA extensions for finite field arithmetic: Accelerating Kyber and NewHope on RISC-V[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020: 219-242.

[8] HUANG Y M, HUANG M Q, LEI Z K, et al. A pure hardware implementation of Crystals-Kyber PQC algorithm through resource reuse[J]. IEICE Electronics Express, 2020, 17(17): 1-6.

[9] DANG V, FARAHMAND F, ANDRZEJ- CZAK M, et al. Implementation and benchmarking of round 2 candidates in the NIST post-quantum cryptography standardization process using hardware and software/hardware co-design approaches[EB/OL]. (2020) [2022]. https://eprint. iacr. org/2020/795.

[10] XING Y F, LI S G. A compact hardware implementation of CCA-secure key exchange mechanism Crystals-Kyber on FPGA[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021, 2021(2): 328-356.

[11] CHEN Z H, MA Y, CHEN T Y, et al. Towards efficient Kyber on FPGAs: A processor for vector of polynomials [C]//Proceedings of 2020 25th Asia and South Pacific Design Automation Conference(ASP-DAC). Piscataway: IEEE Press, 2020: 247-252.

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

[13] 周朕, 何德彪, 罗敏, 等. 紧凑的 Aigis-sig 数字签名方案软硬件协同实现方法[J]. 网络与信息安全学报, 2021, 7(2): 64-76.
ZHOU Z, HE D B, LUO M, et al. Compact software/hard-

ware co-design and implementation method of Aigis-sig digital signature scheme[J]. Chinese Journal of Network and Information Security, 2021, 7(2): 64-76. (in Chinese)

[14] ZHANG J, YU Y, FAN S, et al. Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes[C]//IACR International Conference on Public-Key Cryptography. Cham: Springer, 2020: 37-65.

[15] 崔益军, 姚衍, 倪子颖, 等. 基于 MLWE 的格密码高效硬件实现[J]. 信息安全学报, 2021, 6(6): 40-50.
CUI Y J, YAO K, NI Z Y, et al. Efficient hardware implementation of MLWE lattice based cryptography[J]. Journal of Cyber Security, 2021, 6(6): 40-50. (in Chinese)

[16] FENG X, LI S, XU S. RLWE-oriented high-speed polynomial multiplier utilizing multi-lane stockham NTT algorithm[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2019, 67(3): 556-559.

[17] ZHANG N, YANG B, 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, 2: 49-72.

[18] LI H M, MENTENS N, PICEK S. A scalable SIMD RISC-V based processor with customized vector extensions for CRYSTALS-kyber[C]//Proceedings of the 59th ACM/IEEE Design Automation Conference. New York: ACM, 2022: 733-738.

[19] YAMAN F, MERT A C, ÖZTÜRK E, et al. A hardware accelerator for polynomial multiplication operation of CRYSTALS-Kyber PQC scheme[C]//Proceedings of 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). Piscataway: IEEE Press, 2021: 1020-1025.

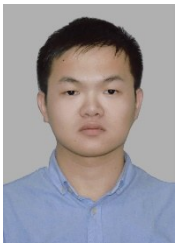
[20] KUO P C, CHEN Y W, HSU Y C, et al. High performance post-quantum key exchange on FPGAs[J]. Jouranl of Information Sciencie & Engineering, 2021, 37(5): 1211-1229.

[21] 李斌, 陈晓杰, 冯峰, 等. 后量子密码 CRYSTALS-KYBER 的 FPGA 多路并行优化实现[J]. 通信学报, 2022, 43(2): 196-207.
LI B, CHEN X J, FENG F, et al. FPGA multi-unit parallel optimization and implementation of post-quantum cryptography Crystals-Kyber[J]. Journal on Communications, 2022, 43(2): 196-207. (in Chinese)

[22] XING Y F, LI S G. An efficient implementation of the NewHope key exchange on FPGAs[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 67(3): 866-878.

[23] BISHEH-NIASAR M, AZARDERAKH- SH R, MOZAFFARI-KERMANI M. Instruction-set accelerated implementation of Crystals-Kyber[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2021, 68(11): 4648-4659.

作者简介



吕顺森　男,1999年出生,河南信阳人. 郑州大学硕士生. 主要研究方向为后量子密码、高性能计算.
E-mail: l_sh_sen@163.com

李　斌　男,1986年出生,河南郑州人. 博士. 郑州大学讲师. 主要研究方向为信息安全、可重构计算.
E-mail: iebinli@zzu.edu.cn

翟嘉琪　男,1998年出生,河南郑州人. 郑州大学硕士生. 主要研究方向为高性能计算、体系结构.
E-mail: zhaijiaqi777@163.com

李松岐　男,1998年出生,河南洛阳人. 郑州大学硕士生. 主要研究方向为高性能计算、体系结构.
E-mail: zzuqiezz@163.com

周清雷　男,1962年出生,河南新乡人. 博士. 郑州大学教授. 主要研究方向为信息安全、自动机理论和计算复杂性理论.
E-mail: ieqlzhou@zzu.edu.cn