格基密钥封装算法 OSKR/OKAI 硬件高效实现

胡 跃[1),2)] 赵旭阳[1)] 刘裕雄[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 100878)

摘 要 量子计算技术的快速发展为现有公钥密码体系(RSA、椭圆曲线密码等)带来了巨大的挑战,为了抵御量子计算的攻击,后量子密码技术受到了学术界和工业界的广泛研究.其中,格基密码方案具有良好的安全性与实现效率,成为后量子密码领域的主要研究方向之一.最近,美国标准与技术研究院公布了基于模格 MLWE 困难问题的 Kyber 算法作为密钥封装方案的标准,2019 年我国举行的后量子密码算法竞赛的一等奖获奖算法 Aigis 也是基于同类困难问题.基于非对称密钥共识机制、混合数论变换、封装 512 比特密钥长度等技术,我国学者进一步提出了 Kyber 和 Aigis 的优化算法:OSKR 和 OKAI.针对算法设计高效、统一的硬件架构对我国推进后量子密码的标准化进程具有重要的借鉴意义.本文基于 FPGA 平台设计实现 OSKR 和 OKAI 两种算法的专用电路结构,主要工作如下:设计了一种四并行的多项式运算模块,可实现多种模值参数(3329 和 7681)下的数论变换、多项式乘法、多项式压缩等运算过程,从而提升了算法的整体运行效率;在此基础上设计了多功能采样模块、编解码模块和存储模块等,充分利用 FPGA 平台并行性的特点研究核心运算模块的优化设计.考虑到在密码实际应用中,往往需要在一个硬件平台上同时实现国际标准和国家标准两套算法,本文的设计可同时满足两套算法共六组参数的运算需求.本文的设计方案在 Xilinx Artix-7 开发板上进行了实际的部署和运行,并且和已有的同类型工作进行了对比,包括纯硬件设计、软硬协同设计和纯软件优化三种类型.结果表明,与最新的工作相比,本文的设计使得解封装的效率提升了 30%左右;同时硬件资源消耗最高为 12765 个 LUT、11434 个 FF、4 个 DSP 和 12.5 个 BRAM,略多于最新的工作,但本文的硬件设计可实现更多的算法和参数,有效提升了硬件资源的复用率.

关键词 后量子密码;格基密码;密钥封装;FPGA 硬件实现

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2023.01156

Hardware Implementation of Lattice-Based Key Encapsulation Mechanism Algorithm OSKR/OKAI

HU Yue[1),2)] ZHAO Xu-Yang[1)] LIU Yu-Xiong[1)] ZHAO Yun-Lei[1),2)]

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

[2)](*State Key Laboratory of Cryptology, Beijing* 100878)

Abstract The rapid development of quantum computing technology has brought great challenges to the traditional public key cryptography (RSA, elliptic curve cryptography, etc), which will threaten the security of the existing encryption systems and applications. To defend against the possible attack of quantum computers, post-quantum cryptography (PQC) techniques have been widely studied by the academic community and industrial community in the past few years. Among which lattice-based cryptography scheme which has the excellent characteristic of strong security and highly effective implementation makes it an important research direction in the field

收稿日期:2022-07-25;在线发布日期:2023-01-18.本课题得到国家自然科学基金(61877011)、国家重点研发计划基金(2022YFB2701600)、上海科技创新行动计划技术标准项目(21DZ2200500)、山东省重点研发计划基金资助项目(2017CXG0701,2018CXGC0701)资助.

胡 跃,博士研究生,主要研究方向为后量子密码、密码工程.E-mail:20110240071@fudan.edu.cn.**赵旭阳**,博士研究生,主要研究方向为后量子密码、密码工程.**刘裕雄**,硕士研究生,主要研究方向为后量子密码、密码工程.**赵运磊**(通信作者),博士,复旦大学特聘教授,主要研究领域为后量子密码、密码协议、密码工程、计算理论等.E-mail:ylzhao@fudan.edu.cn.

of post-quantum cryptography. According to the newest statement, the National Institute of Standards and Technology (NIST) has published that the chosen ciphertext attack (CCA) secure CRYSTALS-KYBER will be the unique standard algorithm of lattice-based key encapsulation mechanism (KEM) protocol. Aigis is one of the KEM algorithms that won the first prize in the National Cryptographic Algorithm Design Competition which is held by the Chinese Association for Cryptologic Research in 2019. What's more, Kyber and Aigis are all KEM algorithms which are based the module learning-with-errors (MLWE). OSKR and OKAI are the optimization algorithms of Kyber and Aigis respectively which are proposed by Chinese researchers. These two algorithms introduce some optimized technologies on the foundation of Kyber and Aigis, including asymmetric key consensus mechanism (AKC), hybrid number-theoretic transform (HNTT), new method about how to encapsulate 512 bits and so on, which optimize the original two algorithms in terms of security and error rate. Considering the unique and highly efficient hardware design ways of these lattice-based key encapsulation algorithms, including OSKR and OKAI, will have important referential significance with the progressing of Chinese post-quantum cryptography algorithm standard. In this paper, we present a dedicated and specific hardware design of OSKR/OKAI algorithm based on the FPGA platform, our contribution include: Firstly, we design a new quadruple parallel polynomial operation module which can realize multiple operations just like Number Theoretic Transform (NTT), polynomial multiplication, polynomial compress, con/rec operation and so on, which significantly improve the calculation efficiency of the whole algorithm. Secondly, the polynomial operation module can realize the operations mentioned above with diffident module value (3329 and 7681). Thirdly, we design the other necessary modules including sampling module, encode module, storage module and so on, and we concentrate on the decent performance of each module through using the synchronizing characteristics of FPGA platform. Considering that the practical cryptography applications usually need to implement both international standard algorithm and national standard algorithm simultaneity, our design can realize the OSKR/OKAI algorithm simultaneously with six selected parameter sets. Our design has been implemented on the Xilinx Artix-7 device. We compare our simulation result with other work, including the pure hardware design, software-hardware co-design and pure software optimization, which shows that the decapsulation time performance of our design improve by about 30% compare with the corresponding work. On the other hand. Our design consumes 12765 LUTs, 11434 FFs, 4 DSPs and 12.5 BRAMs, which is slightly more than some corresponding work. However, our hardware implementation can realize two lattice-based key encapsulation mechanism protocols, which means that our design has better reusability of hardware resources.

Keywords　post-quantum cryptography; lattice cryptography; key encapsulation mechanism; hardware implementation of FPGA

1　引　言

现有的公钥密码系统的设计大多是基于两类数学难题：大整数因子分解问题(RSA)和椭圆曲线(Ellipese Curve Cryptography，ECC)上的离散对数问题.但随着量子计算的技术的发展，传统的公钥密码(例如 RSA 和 ECC)所依赖的数学难题将被运行 Shor 算法[1]的实用量子计算机在多项式时间内所破解，因此能够抵抗量子攻击的新型密码系统的研发得到了学术界和工业界的广泛关注.美国国家标准与技术研究院(National Institute of Standards and Technology，NIST)从 2016 年开始向全球征集后量子密码方案，包括密钥封装(Key Encapsulation

Mechanism,KEM)和数字签名(Digital Signature)两大类[2].经过三轮评选最终得到7个最终算法和8个候选算法,其中CRYSTALS-KYBER是基于格困难问题的密钥封装算法,具有很好的研究价值和应用前景.近日NIST官方又公布了最新的进展,确定将KYBER算法作为最终的KEM标准算法,另有三个数字签名算法将被标准化,并开始了第四轮的算法征集和评测工作[3].中国也在积极地推进后量子密码算法的研究[4],2019年中国密码学会举行的后量子密码算法设计竞赛,一等奖KEM获奖算法包括Aigis[5]和LAC等.其中Aigis方案是基于非对称模误差学习问题(A-MLWE)设计的后量子密码方案,包括密钥封装算法(Aigis-enc)和数字签名算法(Aigis-sig).与NIST第三轮的Kyber相比[6],Aigis-enc能够更有效地平衡错误率和安全性.在此基础上,基于非对称密钥共识机制(Asymmetric Key Consensus,AKC)[7-8]、混合数论变换(Hybrid Number Theoretic Transform,H-NTT)[9]等技术,我国学者进一步提出了密钥封装方案OSKR和OKAI[10],分别对应Kyber和Aigis的优化.

后量子格基密码算法的硬件高效实现和测试对我国推进后量子密码的标准化具有重要的借鉴意义.本文基于FPGA平台的并行性的特点,分析密钥生成、加密、解密等步骤的核心运算、调用关系、时序逻辑和接口设计,将完整的算法流程拆解为多个专用的模块,并研究核心运算模块的优化设计,最终实现了专用的格基密钥封装算法的电路结构.特别地,在密码算法的实际部署和应用时,往往需要同时满足国际标准和国家标准两种情况,因此本文的硬件设计可同时满足OSKR/OKAI两套算法共六组参数的运算需求.本文的主要工作包括:

(1)设计了一种多功能、可配置参数的并行多项式运算模块,可支持标准NTT(Number Theoretic Transform)运算、T-NTT(Truncated NTT)运算和H-NTT运算以及对应的模乘运算、模加运算、压缩运算,可满足多项式次数n为256和512两种情况下的多项式运算过程,并以此为基础给出密钥封装算法OSKR/OKAI的硬件设计,可满足两个算法共六组参数的运算需求;

(2)对多项式运算过程中的运算类型和数据位宽进行分析并确定存储模块的最小空间,包括BRAM资源和FIFO资源,可同时满足模值q为3329和7681两种情况下的多项式系数的存储需求;

(3)对较为耗时的哈希公钥和哈希密文操作进行合适的优化,进一步提高运行效率;

(4)在设计状态机时采用On-the-Fly的方法,通过分析各运算模块的时钟周期数,设计合理的顶层时序逻辑,利用FPGA并行性的特点实现最大化的时间复用.

2 相关工作

格基密码方案是基于格上的数学困难问题,例如格上的带误差学习问题(Learning With Error,LWE)[11],包括判定型LWE和搜索型LWE.LWE基本思想为已知矩阵$\boldsymbol{A}\in Z_q^{m\times n}$和向量$\boldsymbol{b}\in Z_q^m$满足公式$\boldsymbol{b}=\boldsymbol{A}\cdot\boldsymbol{s}+\boldsymbol{e}$,求解向量$\boldsymbol{s}\in Z_q^n$是困难的.由于LWE问题涉及到复杂的向量和矩阵的相乘,运算复杂度较高且消耗存储空间较大难以实际运用,因此提出了环上的带误差学习困难问题(Ring Learning With Error,RLWE)[12],基本思想为首先通过随机采样得到非零向量$\boldsymbol{a}=(a_0,a_1,\cdots,a_{n-1})$作为初始的代表元,通过对向量$\boldsymbol{a}$在多项式环$\mathbb{Z}_q/(x^n+1)$上的特定运算,例如负循环移位操作,得到其它向量集合并共同构成矩阵$\boldsymbol{A}$.在后量子密码体系中,多数情况下使用x^n+1作为多项式模.模格的带误差学习问题(Module Learning With Error,MLWE)[13],即矩阵$\boldsymbol{A}$和向量$\boldsymbol{b}$中的每一个元素均是一个n次多项式,且矩阵$\boldsymbol{A}$和向量$\boldsymbol{b}$的维度一般较小,目的是在性能和安全性中取得平衡.CRYSTALS-KYBER是基于MLWE问题构建的格基密钥封装方案[3,6,14],并且已被NIST官方选定为后量子密钥封装标准.Zhao等人[7-8]提出了“共识机制(AKC)”的密码学工具,由(Con,Rec)两个算法组成,核心思想为当两个算法输入参数较为接近时,Rec算法可恢复出Con算法中的共识值.Aigis-enc是我国自主提出的基于格上的非对称错误学习问题(A-MLWE)的密钥封装机制[5,15-16],与Kyber相比能够更好的平衡安全性和错误率.Shen等人从理论上比较了基于LWE类问题和基于AKC机制构造的格基密钥封装方案的异同点[10],给出了两种构造方案的参数选择和错误率之间的对应关系,并且证明了当参数相同时基于AKC机制构造的KEM方案具备更好的性能;在此基础上,文献[10]系统分析了基于MLWE的密钥封装机制并做出进一步优化,包括非

对称密钥共识机制(AKC)[7-8]、混合数论变换技术(H-NTT)[9]和扩展会话密钥为 512 比特三个方面，提出了新的密钥封装算法 OSKR 和 OKAI，分别对应于 Kyber 和 Aigis 的优化. 具体而言，相对于 Kyber 和 Aigis，OSKR 基于 AKC 机制可以在相同的参数下具有更优的解密效率和更低的错误率；相对于 Kyber，通过利用 H-NTT 技术，OSKR 在 1024 维度时可以高效封装 512 比特的密钥(而 Kyber 封装的密钥长度固定为 256 比特)；相对于 Aigis，基于 H-NTT 技术，OKAI 将三组参数的模数 q 统一为 7681(而 Aigis-1024 采用了不同的模数 $q=12289$).

基于 FPGA 的格基密码方案的硬件优化实现是 PQC 领域一个重要的研究方向，对于推动格基密码算法的实际应用具有重要的作用. 多项式模乘是格基密码方案中耗时最长的操作，通常采用数论变换技术(Number-Theoretic Transform，NTT)实现多项式运算. Mert 等人设计了专用的 NTT 运算模块和访问控制单元[17-19]，通过优化内存访问模式和时序逻辑设计防止数据冲突，并提高了 NTT 运算的并行化程度；模块的通用化设计也是一个研究重点[20-21]，其中 Xing 等人设计的多项式运算单元可实现 NTT 变换、乘法运算、模加运算、压缩运算等多种运算操作，有效地提升了硬件资源的利用率；Mert 等人设计了一种可实时配置且高度并行的多项式乘法器结构，可实现六种不同参数的 NTT 运算；Zhang 等人提出了约减模块的优化方案[22-23]，实现了恒定时间的模约减操作，提高了模运算的效率；Dang 等人基于 FPGA 平台设计并实现了完整的 CRYSTALS-KYBER 算法[17,20,24-25]，并给出了性能测试结果，包括时序测试和硬件资源占用情况；Xin 等人基于 RSIC-V 指令集设计实现 PQC 方案[26-29]，主要思想为将方案中的关键运算步骤通过硬件进行加速，并通过 RSIC-V 指令集设计的软核完成运算流程的控制，软核和硬件加速单元均使用 FPGA 的逻辑资源实现；文献[30]提出基于 HLS 等高层次硬件语言从算法层面设计 PQC 方案[30]，节约了开发时间，但缺陷是编译得出的 HDL 代码通常会存在大量冗余并消耗过多的硬件资源.

3　预备知识

3.1　算法介绍

OSKR 算法和 OKAI 算法是基于 MLWE 困难问题的抗量子密钥封装算法，其核心是首先构造满足 IND-CPA 安全的公钥加密方案，包括密钥生成算法 CPAPKE.KeyGen、加密算法 CPAPKE.Enc 和解密算法 CPAPKE.Dec 三个部分. 在此基础上通过 FO 转换(Fujisaki-Okamoto transformation)构造[31] CCA 安全的密钥封装方案，并且满足正确性和安全性需求.

如算法 1 所示，中心二项采样算法通过 CBD 函数实现，可生成私钥多项式向量 $\boldsymbol{s}$ 和噪声向量 $\boldsymbol{e}$ 等；拒绝采样算法通过 Parse 函数实现，生成公钥多项式矩阵 $\boldsymbol{A}$ 和转置矩阵 $\boldsymbol{A}^{\mathrm{T}}$. 在加密算法中，采样得到的多项式经过运算得到密文 $\boldsymbol{u}$ 和 v，加密信息 m 内嵌入 v 中，在经过压缩运算得到最终密文(c_1,c_2)，解密部分通过解压缩等运算得到明文 m. OSKR 方案和 OKAI 方案引入了文献[7-8]提出的密钥共识(AKC)的思想，在加密和解密时分别通过算法 Con 和算法 Rec 实现，两个算法的定义和实现原理分别见 3.3 节内容和式(7).

算法 1.　CPA-PKE.

KeyGen()：

1. $\sigma,\rho\leftarrow(0,1)^n$
2. $\boldsymbol{A}\in R_q^{l\times l}:=\mathrm{Parse}(\rho)$
3. $(\boldsymbol{s},\boldsymbol{e})\in\psi_{\eta_s}^{l\times l}\times\psi_{\eta_e}^{l\times l}:=\mathrm{CBD}(\sigma)$
4. $t:=compress_q(\boldsymbol{A}\circ\boldsymbol{s}+\boldsymbol{e},\boldsymbol{d}_t)$
5. $(pk:=(t,\rho)),sk:=\boldsymbol{s}$

Enc($\boldsymbol{pk},\boldsymbol{k},\boldsymbol{r}$)：

1. $\hat{\boldsymbol{A}}\in R_q^{l\times 1}:=\mathrm{Parse}(\rho)$
2. $(\boldsymbol{r},\boldsymbol{e}_1,\boldsymbol{e}_2)\in\psi_{\eta_{(\boldsymbol{r},\boldsymbol{e},\boldsymbol{e})}}^{l\times l}:=\mathrm{CBD}(\boldsymbol{r})$
3. $\boldsymbol{u}:=compress_q(\hat{\boldsymbol{A}}^{\mathrm{T}}\circ\boldsymbol{r}+\boldsymbol{e}_1,d_u)$
4. $\hat{\boldsymbol{t}}:=Decompress_q(t,d_t)$
5. $\sigma_2:=\hat{\boldsymbol{t}}^{\mathrm{T}}\circ\boldsymbol{r}+\boldsymbol{e}_2$
6. $v:=\mathrm{Con}(\sigma_2,m,parame)$
7. $ct:=(c_1:=\boldsymbol{u},c_2:=v)$

Dec($\boldsymbol{sk},\boldsymbol{ct}$)：

1. $\sigma_1:=\boldsymbol{s}^{\mathrm{T}}\circ Decompress_q(\boldsymbol{u},d_u)$
2. $m:=\mathrm{Rec}(\sigma_1,v,parame)$

为了提高密钥封装算法的安全性，需要将封装密钥的长度从 256 比特提升为 512 比特. 以 NIST 第三轮的 kyber-1024 算法对应的参数为例，为了实现这一目标共有三种可行的方案：(1)采用原始参数：$(n,l,m)=(256,4,2)$，完成两轮封装并合并结果；(2)参数 m 设置为 4，完成一次封装并生成 512 比特密钥；(3)扩大多项式次数，设参数 $n=512$，完成一次封装运算. 文献[10]对三种方案从错误率、安全强度和通信带宽三个维度进行严格的证明和分析，结论为在满足相同安全性和错误率的前提下，方案三具备更低的带宽消耗和更灵活的参数配置. 如表 1 所示，文献[10]给出了两个方案的最优参数

集,并与 kyber 第三轮的参数进行对比.由于 OSKR 和 OKAI 两个密钥协商算法的构造方式类似,仅在参数上有所区别,因此可采用同一个硬件电路结构进行实现,通过改变传入的片选信号实现不同的算法.

表 1 OSKR/OKAI 参数集

	n	q	m	l	η_s	η_e	d_k	d_v
				$N=512$				
OSKR	256	3329	2	2	3	2	10	4
OKAI	256	7681	2	2	1	4	8	4
				$N=768$				
OSKR	256	3329	2	3	2	2	10	4
OKAI	256	7681	2	3	1	4	9	4
				$N=1024$				
OSKR	512	3329	2	2	2	2	11	5
OKAI	512	7681	2	2	1	4	10	3

3.2 NTT 算法介绍

多项式模乘运算是格基密码方案中计算复杂度最高、耗时最长的操作,其实现效率关系着算法的整体性能,为了提高效率通常使用 NTT 算法实现多项式乘法运算.NTT 是一种在有限域 Z_q 上的特殊形式的 DFT 算法,其核心思想是将多项式由系数表达转化为点值表达,并通过在 $2n$ 次本原单位根 w 位置求插值,利用 w 的特殊性质降低乘法运算的计算复杂度.设环 $\mathbb{Z}_q[x]/(x^n+1)$ 上多项式 f 和 g,系数 $n=2^k$ 为多项式次数,系数 q 为模值,且严格满足公式 $q\equiv1(\bmod 2n)$,正向 NTT 运算和逆向 NTT 运算如式(1)和式(2)所示,其中 $m,k=0,1,\cdots,n-1$,w 是 $\mathbb{Z}_q$ 上的 $2n$ 次本原单位根,满足公式 $w^{2n}\equiv1(\bmod q)$,参数 $r=w^2 \bmod q$.

$$\mathrm{NTT}(f)=\hat{f}_m=\sum_{k=0}^{n-1}(f_k w^k)r^{mk} \bmod q \tag{1}$$

$$\mathrm{INTT}(\hat{f})=f_k=\frac{w^{-k}}{n}\sum_{k=0}^{n-1}\hat{f}_m r^{-mk} \bmod q \tag{2}$$

使用 FFT-trick 算法可实现快速的 NTT 运算,并将计算复杂度降为 $O(n\log n)$:由于单位根满足 $w^n\equiv-1(\bmod q)$,则多项式环 x^n+1 可分解为 $(x^{n/2}-w^{n/2})(x^{n/2}+w^{n/2})$,再次代入 w^n 可做进一步地分解,直到分解为 n 个一次多项式即可得到最终的运算结果,上述的正向 FFT-trick 操作通常使用 CT(Cooley-Tukey)蝴蝶变换来实现,输入正常序列,输出比特翻转序列.逆向 FFT-trick 是上述操作的逆过程,通常使用 GS(Gentleman-Sande)操作来实现,如图 1 所示.标准 FFT-trick 计算过程使用中国剩余定理(Chinese Remainder Theorem,CRT)可表示为

$$\mathbb{Z}_q[x]/(x^n+1)\cong\prod_{i=0}^{n-1}\mathbb{Z}_q[x]/(x-w^{2br(i)+1}) \tag{3}$$

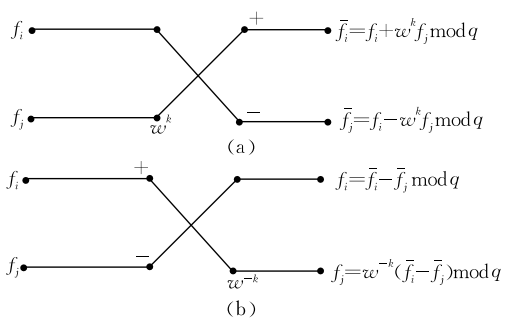

图 1 Cooley-Turkey 蝴蝶操作(a)和 Gentlemanm-Sande 蝴蝶操作(b)

其中 $br(i)$ 表示整数 i 的比特翻转值,结果对应 n 个一次多项式在 $2n$ 次本原单位根处求值.如果将 FFT-trick 计算过程只运行到倒数第二层,可得到 $n/2$ 个二次多项式.这种情况下只需满足 $\mathbb{Z}_q$ 中有 n 次本原单位根 $\{\xi,\xi^3,\xi^5,\cdots,\xi^{2n-1}\}$ 的存在即可完成对应点处求值,FFT-trick 最终运算结果为 $n/2$ 个一次多项式.这种使用底层裁剪技巧的 NTT 改进运算被称为 T-NTT(Truncated-NTT)[32-33] 使用 CRT 可表示为

$$\mathbb{Z}_q[x]/(x^n+1)\cong\prod_{i=0}^{n/2-1}\mathbb{Z}_q[x]/(x^2-w^{2br(i)+1}) \tag{4}$$

T-NTT 拓宽了标准 NTT 的使用范围,可满足 $q\equiv1(\bmod 2n/2^\beta)$ 条件下的多项式数论变换运算,其中 β 是底部删减的层数,当 $\beta=1$ 时计算复杂度为 $3/2n\log n+3/2n$,在 Kyber 第三轮[6] 已经使用到了这种方法.由于 T-NTT 的最终结果是 $n/2$ 个一次多项式,因此多项式的点乘运算将转化为一次域 $\mathbb{Z}_q$ 内的乘法运算,共涉及 5 次乘法运算,使用 Karatsuba 算法可将乘法次数减少为 4 次,如式(5)所示.

$$\begin{aligned}\hat{h}&=(\hat{f}_{2i}+x\hat{f}_{2i+1})(\hat{g}_{2i}+x\hat{g}_{2i+1}) \bmod (x^2-w^{2br(i)+1})\\&=\hat{f}_{2i}\hat{g}_{2i}+w^{2br(i)+1}\hat{f}_{2i+1}\hat{g}_{2i+1}+\\&\quad((\hat{f}_{2i}+\hat{f}_{2i+1})(\hat{g}_{2i}+\hat{g}_{2i+1})-\hat{f}_{2i}\hat{g}_{2i}-\hat{f}_{2i+1}\hat{g}_{2i+1})x\end{aligned} \tag{5}$$

在此基础上混和数论变换技术(Hybrid NTT,H-NTT)做了进一步的优化[9].定义 H-NTT(n,α,β),可满足 $q\equiv1(\bmod(n/2^{\alpha+\beta+1}))$ 条件下的多项式数论变换,H-NTT 技术的核心思想为将多项式从顶部分解为 2^α 个部分并分别进行 NTT 运算,同时对底层进行 β 层删减.使用 H-NTT 技术进行多项式乘法运算过程包括多项式分解(算法 2 中的 SPLIT 函数)、运算(TNTT 函数、POLY_POINTWISE 函数、INVTNTT 函数)和结合(COMBINE 函数)三个步骤.当系数 $\alpha=\beta=1$ 时使用 H-NTT 技术进行多项式乘法运算的计算复杂度为 $\frac{3}{2}n\log n+\frac{5}{4}n$,相比于

同等条件下的 T-NTT 计算复杂度$\frac{3}{2}n\log n+\frac{3}{2}n$更有优势，H-NTT 运算的伪代码如算法 2 所示.

算法 2.　H-NTT(n,α,β).

1. $(f_0,f_1)\leftarrow SPLIT(f)$
2. $(\boldsymbol{g}_0,\boldsymbol{g}_1)\leftarrow SPLIT(\boldsymbol{g})$
3. $\hat{\boldsymbol{f}}_0\leftarrow TNTT(\boldsymbol{f}_0)$
4. $\hat{\boldsymbol{f}}_1\leftarrow TNTT(\boldsymbol{f}_1)$
5. $\hat{\boldsymbol{g}}_0\leftarrow TNTT(\boldsymbol{g}_0)$
6. $\hat{\boldsymbol{g}}_1\leftarrow TNTT(\boldsymbol{g}_1)$
7. $\hat{\boldsymbol{p}}_0\leftarrow POLY\_POINTWISE(\hat{\boldsymbol{f}}_0,\hat{\boldsymbol{g}}_0)$
8. $\hat{\boldsymbol{p}}_1\leftarrow POLY\_POINTWISE(\hat{\boldsymbol{f}}_1,\hat{\boldsymbol{g}}_1)$
9. $\hat{\boldsymbol{p}}\leftarrow POLY\_POINTWISE(\hat{\boldsymbol{f}}_0+\hat{\boldsymbol{f}}_1,\hat{\boldsymbol{g}}_0+\hat{\boldsymbol{g}}_1)$
10. $\boldsymbol{h}_0\leftarrow INVTNTT(\hat{\boldsymbol{p}}_0+\hat{\boldsymbol{y}}\circ\hat{\boldsymbol{p}}_1)$
11. $\boldsymbol{h}_1\leftarrow INVTNTT(\hat{\boldsymbol{p}}-\hat{\boldsymbol{p}}_0-\hat{\boldsymbol{p}}_1)$
12. $\boldsymbol{h}\leftarrow COMBINE(\boldsymbol{h}_0,\boldsymbol{h}_1)$
13. return $\boldsymbol{h}$
14. End

表 1 所示的六组参数分别涉及到上述的三种 NTT 运算，其中(256,7681)对应标准 NTT，(256,3329)和(512,7681)对应标准 T-NTT，(512,3329)对应 H-NTT，在设计这部分的功能模块时需要同时满足三种情况.

3.3　多项式压缩运算

加解密过程中通常使用压缩函数在不影响安全性的情况下删去密文或公钥的低比特位数据，从而节约存储空间，压缩和解压缩函数定义如式(6)所示，其中 d 为 2 的幂次.

$$\begin{cases}\mathrm{Compress}_q(x,d)=\lfloor(2^d/q)\cdot x\rceil \bmod^+ 2^d\\ \mathrm{Decompress}_q(x,d)=\lfloor q/(2^d)\cdot x\rceil \bmod q\end{cases}\tag{6}$$

使用 AKC 机制可对加密过程进行优化：AKC 机制包含 Con 和 Rec 两个算法，可用于生成共识密钥，算法定义如式(7)所示，其中参数 $m=2^{d_m}$，$g=2^{d_v}$，$\sigma_1,\sigma_2\in\mathbb{Z}_q$均为多项式系数，$k_1,k_2\in\mathbb{Z}_m$，当满足公式$|\sigma_1-\sigma_2|\le d$时($d$ 是一个很小的值)共识机制可满足 $k_1=k_2$. 这意味着在加密过程中引入噪声多项式向量 $\boldsymbol{e}_2$在多项式 σ_2上加入一定的扰动，确保加密的安全性，只要扰动空间保持在一定范围，共识机制就可以正确的恢复出共享密钥 k_1. 与 Kyber 使用的压缩方法相比，基于 AKC 机制可以使解密过程效率更高且错误率更低.

进一步对比式(6)和式(7)，可以看出 Con 算法和 Rec 算法从形式上分别包含了压缩算法和解压缩算法的运算过程：Con 算法可解释为 $C_q(\sigma_2+D_q(k,d_m),d_v)$，Rec 算法可解释为 $C_q(D_q(v,d_v)-\sigma_1,d_m)$，其中 C_q和 D_q对分别为压缩和解压缩运算. 在进行硬件开发时，利用算法之间这种形式上的关系可设计一个统一的功能模块，有利于提高硬件资源的复用率.

$$\begin{cases}\mathrm{Con}(\sigma_2,k_1):=v=\lfloor g(\sigma_2+\lfloor k_1q/m\rceil)/q\rceil\rceil \bmod g\\ \mathrm{Rec}(v,\sigma_1):=k_2=\left\lfloor\frac{m}{q}(\lfloor q/g\cdot v_i\rceil-\sigma_1)\right\rceil \bmod m\end{cases}\tag{7}$$

3.4　约减运算

本文的约减模块借鉴了论文[20]的设计思路，使用 Barret 约减算法可同时得到余数和商，约减结果范围[0,q]. 为了便于计算通常将约减过程的除法运算用乘法运算代替，即乘以模数 q 的逆元得到，如式(8)所示.

$$r=a(\bmod q)=a-q\cdot\lfloor a\cdot q^{-1}\rfloor=a-q\cdot\left\lfloor\frac{a}{v}\cdot\frac{v}{q}\right\rfloor\tag{8}$$

其中系数 v 值通常为 2 的幂次，这样的取值可将除法运算转化为移位运算. 算法 3 所示步骤 1 中 v 的值可提前预计算得到，同时步骤 3 可将乘法运算通过移位操作实现，步骤 2 得到的结果 quo 是计算过程中的商值，最终的约减结果通过在步骤 4 中得到.

算法 3.　Barret Reduction.

输入：$-\frac{\beta}{2}\le a\le\frac{\beta}{2}$，$\beta=2^{16}$

1. $v=\frac{2^{2\log(q)-1}\beta}{q}$
2. $quo=\frac{av}{2^{2\log(q)-1}\beta}$
3. $t=q\cdot quo \bmod \beta$
4. $res=a-t$

4　方案设计

密钥封装算法 OSKR/OKAI 硬件实现的顶层架构如图 2 所示，主要包括顶层模块、多项式运算模块、采样模块、编解码模块以及存储模块. 其中顶层模块负责密钥生成、密钥封装和解封装三部分的时序逻辑设计，协调各底层模块通信的先后顺序，规定模块间的数据流通，最终输出密文 c 和密钥 K. 多项式运算模块由多功能蝴蝶变换模块、存储模块和对应的时序逻辑构成，可实现四并行的 NTT 运算、逆

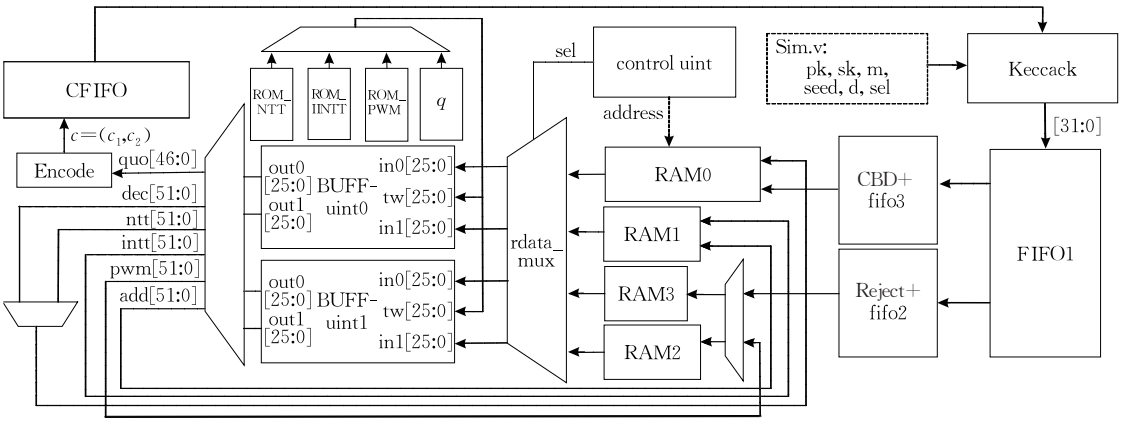

图 2 密钥封装算法硬件实现顶层结构框图

NTT 运算、模乘模加运算、压缩解压缩运算和 Con/Rec 运算，根据状态机的变化实现不同的功能，并最终完成密钥封装/解封装操作.

4.1 紧凑型多功能蝶形操作模块设计

多项式运算模块的各种运算均由蝶形操作模块完成，如图 3 所示蝶形模块包括加法器、减法器、乘法器、移位寄存器和约减单元等硬件资源组成，根据片选信号 sel 实现不同的运算过程. 多项式运算模块包含两个图 3 所示的蝶形模块，因此可实现四并行的多项式运算. 如 3.2 节所述，OSKR 算法(512,3329)这组参数的多项式运算需要通过 H-NTT 算法实现. 为了提高代码的兼容性，在进行硬件设计时可将 512 维的多项式分解为两个 256 维多项式分别进行 T-NTT 运算(即 $\alpha=\beta=1$)，状态机根据累加信号 ctr_NTT 判断是否完成 H-NTT 运算，从而实现整个多项式的正向和逆向 NTT 变换；对于参数为(256,3329)的多项式运算依然通过 T-NTT 算法完成，此时信号 ctr_NTT 为 0 即已完成运算. OKAI 算法的两组参数(512,7681)、(256,7681)分别与上述两种情况一一对应，不同点在于后者采用标准 NTT 算法实现多项式运算，在状态机设计时比 T-NTT 多进行一层运算即可.

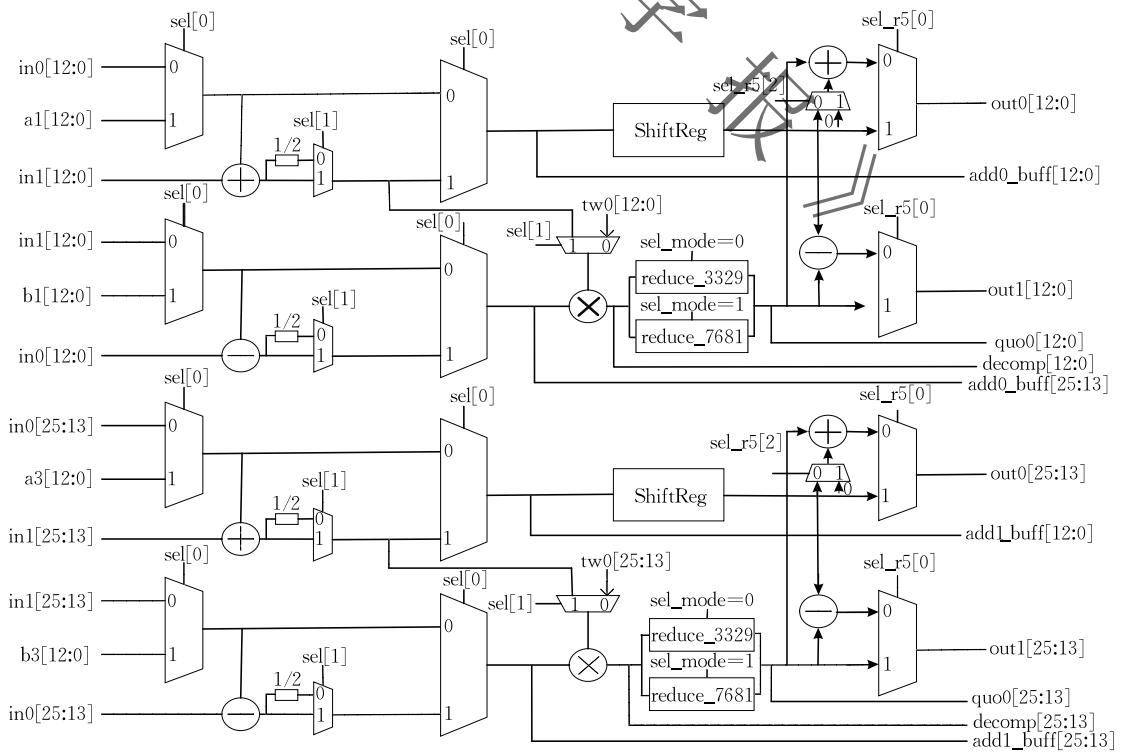

图 3 蝶形操作模块结构框图

表 2 给出了各种多项式运算对应的片选信号取值，当 sel=3′b000 时蝶形模块实现 CT 蝴蝶变换，以自然顺序排列的系数向量作为输入，输出比特翻转排列的系数向量，此时输出信号为 $A=a+b\cdot w(\mathrm{mod}\,q)$，$B=a-b\cdot w(\mathrm{mod}\,q)$；当 sel=3′b001 时蝶形模块实现 GS 蝴蝶变换，以比特翻转排列的向量作为输入，输出自然顺序排列的向量，此时输出信号 $A=(a+b)/2(\mathrm{mod}\,q)$，$B=(a-b)\cdot w/2(\mathrm{mod}\,q)$，其中 $x/2(\mathrm{mod}q)$ 操作可通过移位和加法运算实现[22]. 本文实现的多项式运算模块为四并行模式，在进行 NTT 运算时每个时钟周期分别输入输出四对(共八个)系数进行运算，因此需要设计合适的内存访问逻辑. 如图 5 所示，采样得到的多项式系数按照正常顺序存放在 BRAM 单元中，每层 NTT 运算时相邻两个时钟周期读取地址相差固定长度 i，如式(9)所示，确保每两个时钟周期读取的 16 个多项式系数可分成两组对应的操作数，再通过合理的延迟操作保证每个时钟周期均有四对正确的输入系数. 这样的设计模式可最大化的利用时钟周期和存储资源，保证了 NTT 运算流水线高效稳定地运行.

$$\begin{cases} raddr_0=k+1,\ k=0,1,2,3,\cdots,j \\ raddr_1=raddr_0+i,\ i=16,8,\cdots,1 \end{cases} \tag{9}$$

蝶形模块每个时钟周期可吸收进入 8 个多项式系数，完成一次正/逆向数论变换共需要 448 个时钟周期($n=512$).

表 2　片选信号取值

Operation	Sel[2:0]
NTT	3′b000
INTT	3′b001
PWM0	3′b011
PWM1/PWM2	3′b100
PWM3	3′b000
Compress	3′b001, 3′b011
Decompress	3′b000

在加解密阶段共涉及到 Compress、Decompress、Con 和 Rec 四个函数，本文 3.3 节已从理论方面详细分析了四个函数的运算过程和内在关系，而从硬件实现角度来看，函数 Compress 和 Con、函数 Decompress 和 Rec 分别具有类似的运算模式，因此四个函数可以通过设计一套硬件电路去实现. 本文参考了文献[20]的设计思路，使用蝴蝶操作模块中的加法器、乘法器和约减模块去实现上述四个函数，如算法 4 所示.

算法 4.　Con/Rec Algorithms.

Con：

1. $\sigma_2=\sigma_2+D_q(\boldsymbol{k},d_m)$
2. $\sigma_2=\sigma_2+e_2$
3. $\sigma_2=\sigma_2\cdot 2^{d_v}$
4. $quo=reduce(\sigma_2)$
5. $v=quo\gg d_v$

Rec：

6. $v=v\cdot q$
7. $v=reduce(v)$
8. $v=v\gg d_u$
9. $v=v-\sigma_1$
10. $\boldsymbol{k}=v/q$

sel=3′b001、3′b011 分别对应封装过程和解封装过程的 Compress 操作，运算过程分别为 $(2^{d_u}(\boldsymbol{u}_i+\boldsymbol{e}_i)/q)\bmod 2^{d_u}$、$(2^{d_v}(v-\sigma_1)/q)\bmod 2^{d_v}$，对应算法 4 的步骤 3 到步骤 5. 其中 2^{d_u} 和 2^{d_v} 为常数值，可通过预计算得到，x/q 结果对应约减模块的商值 quo，最后的模运算可通过在商值 quo 移位得到. 实现 Con 运算时需要多两个加法运算，对应算法 4 的步骤 1 和步骤 2；当 sel=3′b000 时可实现 Decompress 操作，运算过程为 $(ct/2^d)\cdot q$，使用乘法器计算密文多项式系数与模数 q 的乘积，除法操作 $x/2^d$ 结果通过对乘法器输出取对应比特位得到，对应算法 4 的步骤 6 到步骤 8. 实现 Rec 运算时额外计算步骤 9 和步骤 10. 这种设计不仅减少了时钟周期的消耗，且不需要设计额外的模块实现压缩和解压缩运算，提高了硬件资源的复用率，完成一次压缩/解压缩、Con/Rec 运算共需要 128 个时钟周期.

如 3.4 节所述，在设计约减功能模块时需要考虑模值 q 有 3329 和 7681 两种情况，对应的多项式系数的比特长度分别为 12 比特和 13 比特，因此需要将输入位宽设置为 26 比特，根据片选信号截取 v 的对应比特存入 $c_0\sim c_1$ 四个寄存器中得到正确的商值 quo. 对应算法 3 步骤 1 和 2；根据最低汉明重量原则将模数 q 进行分解，其中 $3329=2^{11}+2^{10}+2^8+1$，$7681=2^{13}-2^9+1$，将步骤 3 中的乘法运算 $t\cdot q$ 转化为移位操作和加减法操作；最后根据 res′的前三比特位对结果进行修正，得到正确的约减结果，模块设计如图 4 所示.

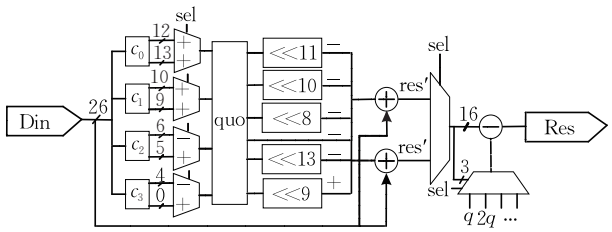

图 4　约减模块电路设计

4.2 改进型点乘运算方案

由于OSKR算法的参数(512,3329)、(256,3329)不满足公式$q\equiv1(\mathrm{mod}\,2n)$,因此多项式通过 H-NTT 算法和 T-NTT 算法进行数论变换后的结果实质上是 $n/2$ 个二项式的对应项系数,如 3.2 节所述两个二次项式 $\hat{f},\hat{g}$ 的乘法运算包括五次乘法运算和三次加法运算,使用 Karatsuba 方法[34]可将乘法运算次数降低为 4 次.由于本文设计的蝶形模块包括四个乘法器,每个时钟周期可输入 8 个操作数,通过对 5 次加法和 4 次乘法的顺序进行重新排布,结合四并行的多项式运算模块,可实现平均每周期完成一对系数的点乘运算,此过程包括四个步骤,对应的片选信号值 sel=3′b011,3′b100,3′b100,3′b000.以四组二次多项式点乘运算为例:

(1)计算$(f_0+f_1)\cdot(g_0+g_1)$,$(f_2+f_3)\cdot(g_2+g_3)$,$(f_4+f_5)\cdot(g_4+g_5)$,$(f_6+f_7)\cdot(g_6+g_7)$,结果存入 RAM3 模块,对应控制信号为 3′b011;

(2)计算 $f_0\cdot g_0$,$f_2\cdot g_2$,$f_4\cdot g_4$,$f_6\cdot g_6$,结果存入 RAM2 模块;同时计算$(f_0+f_1)\cdot(g_0+g_1)-f_0\cdot g_0$,$(f_2+f_3)\cdot(g_2+g_3)-f_2\cdot g_2$,$(f_4+f_5)\cdot(g_4+g_5)-f_4\cdot g_4$,$(f_6+f_7)\cdot(g_6+g_7)-f_6\cdot g_6$,结果存入 RAM3 模块,对应控制信号为 3′b100;

(3)计算 $f_1\cdot g_1$,$f_3\cdot g_3$,$f_5\cdot g_5$,$f_7\cdot g_7$,结果存入 RAM2 模块;同时计算$(f_0+f_1)\cdot(g_0+g_1)-f_0\cdot g_0-f_1\cdot g_1$,$(f_2+f_3)\cdot(g_2+g_3)-f_2\cdot g_2-f_3\cdot g_3$,$(f_4+f_5)\cdot(g_4+g_5)-f_4\cdot g_4-f_5\cdot g_5$,$(f_6+f_7)\cdot(g_6+g_7)-f_6\cdot g_6-f_7\cdot g_7$,结果存入 RAM3 模块,对应控制信号为 3′b100;

(4)计算 $f_0\cdot g_0+f_1\cdot g_1\cdot\omega_0$,$f_2\cdot g_2+f_3\cdot g_3\cdot\omega_1$,$f_4\cdot g_4+f_5\cdot g_5\cdot\omega_2$,$f_6\cdot g_6+f_7\cdot g_8\cdot\omega_3$,结果存入 RAM2 模块,对应控制信号 3′b000.

对于两个系数为(256,3329)的多项式乘法,步骤 4 中用到了128个不同的旋转因子$(w_0,w_1,\cdots,w_{127})$.根据中国剩余定理的可知,乘法运算中的 128 个旋转因子值与 T-NTT 正向变换最后一层的 64 个旋转因子值是相互对应的,即每两个乘法运算使用的旋转因子值对应一个数论变换最后一层使用到的旋转因子值,且数值刚好为一正一负,因此可首先预计算好旋转因子值并存入 ROM 模块.此外,系数为(256,7681)的多项式数论变换后结果为一次项式,可直接进行对应系数相乘,只需在蝶形模块的输入端传入不同的系数即可.完成一次多项式乘法运算共需要 256 个时钟周期(512,3329)或 128 个时钟周期(256,3329)周期或 64 个时钟周期(256,7681).

4.3 最小化存储模块设计

蝶形模块中使用到的操作数以及中间值需要保存到 BRAM 存储单元,为了提高 BRAM 硬件资源的利用率,需要对运算操作数的存储位置进行合适的安排.本文设计的四并行多项式运算模块每个时钟周期最多可输入输出 8 个系数,由于 OKAI 算法的模值为 7681,为了兼容两种算法,将每个系数位宽设为 13 比特,因此总长度为 104 比特.存储单元 RAM0 负责存储正向 NTT 运算时的操作数,包括多项式向量 $\boldsymbol{s}_i$,$\boldsymbol{r}_i$,$\boldsymbol{e}_i$,根据表 1 中的参数(n,l)的值可确定存储单元 RAM0 最多需存储四个 256 维多项式的系数(即两个 512 维多项式).为了存储数据的方便,存储单元 RAM0 分为两块面积相等的 bank,其位宽和深度分别为52和128.完成正向NTT变换后的多项式向量的系数将与采样模块传入的公钥多项式矩阵系数 $\hat{\boldsymbol{A}}_{i,j}$,$\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 进行乘法运算.存储单元 RAM2 和 RAM3 负责存储多项式乘法运算时的操作数.如 4.2 节所述,在进行多项式乘法运算时由于步骤 2 和步骤 3 计算得到的一次项系数需要存储其中间值,因此在$n=512$时存储单元 RAM3 每个 bank 的位宽和深度分别为 26 和 64,存储单元 RAM2 每个 bank 的位宽和深度分别为 52 和 64;当 $n=256$ 时两个存储单元的深度减半,即为了满足多项式乘法运算的需求,存储单元 RAM2 和 RAM3 共存放 1.5 个多项式的系数.乘法运算结果相加并存入到存储单元 RAM1 中,在完成 l 次加法运算后得到的多项式进行逆向 NTT 运算,结果存储在 RAM1 单元中,并进一步完成之后的 Compress/Con 运算.在$n=512$时存储单元 RAM1 每个 bank 位宽和深度分别为 52 和 64;当 $n=256$ 时存储单元的深度减半,即存储单元 RAM1 可存放一个多项式的系数.本文的硬件设计采用原位读写结构,使用双口 BRAM 模块保证蝴蝶操作数的吞吐需求.图 5 给出了 BRAM 单元存取逻辑设计,如 4.1 节所述,通过对于输入输出系数进行合适的延迟处理,可以确保在满足数据带宽的情况下实现原位存储的效果.

考虑到模块之间数据传输的同步问题,本文设计使用 FIFO 资源满足数据缓冲的需求,即采样模块输出的多项式系数首先存入 FIFO 单元,根据多项式运算模块的状态机信号输出存储数据存入到 BRAM 单元,或与 BRAM 单元的输出数据完成乘法运算.FIFO 资源包括两部分,暂时存储 $\hat{\boldsymbol{A}}_{i,j}$,$\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 系数

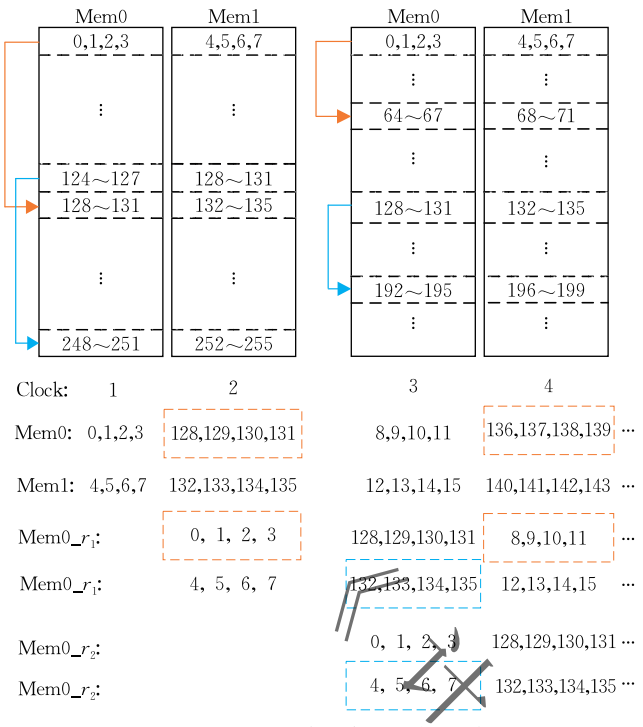

图 5　BRAM 单元存取逻辑设计

的 FIFO2 单元与暂时存储 $\boldsymbol{s}_i,\boldsymbol{r}_i,\boldsymbol{e}_i$ 系数的 FIFO3. 在 $n=256$ 时 FIFO2 深度为 128，位宽为 26 比特，FIFO3 深度 32，宽度为 32 比特，均可存放一个完整的多项式系数(FIFO3 中的暂存数据为 CBD 采样的中间结果，每 η_e/η_s 个比特代表一个系数)；当 $n=512$ 时 FIFO2 深度为 256，位宽为 26 比特，并根据合理的时序逻辑设计将采样得到的 $\boldsymbol{s}_i,\boldsymbol{r}_i,\boldsymbol{e}_i$ 多项式数直接存入到 BRAM 单元中，从而节约一个 FIFO 单元.

4.4　哈希模块与采样模块设计

采样得到多项式前首先需要哈希运算生成随机比特流，OSKR/OKAI 算法共使用到了 4 种 SHA3 实例，以参数(512,3329)为例：SHA128 用于生成公钥多项式矩阵 $\hat{\boldsymbol{A}}_{i,j},\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$，每 24 轮 Keccak 函数可输出 1344 比特数据，在拒绝采样的情况下生成 6144 比特数据至少需要 6 次哈希运算；SHA256 用于生成噪音多项式向量 $\boldsymbol{s}_i,\boldsymbol{r}_i,\boldsymbol{e}_i$，每 24 轮 Keccak 函数可输出 1088 比特数据，中心二项采样的条件下最多需要 1536 比特数据，两次完整的哈希运算可以完成；SHA3-512 和 SHA3-256 用于生成公钥和密文的摘要. 如图 6 所示本文设计了专用型哈希模块以满足哈希运算的需求，模块的工作模式包括吸收数据、轮函数运算和输出数据三个状态，由顶层模块的状态机控制状态的变化，hash_mode 信号负责确定哈希实例和填充数据 r，并判断是否需要进行迭代运算. 哈希模块的输入输出数据位宽设置为 32 比特，哈希生成 $\hat{\boldsymbol{A}}_{i,j}$，$\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 多项式的随机比特流时，输入数据 ρ 长度为 512 比特消耗 16 个时钟周期，六次输出 1344 比特消耗 252 个时钟周期，轮函数过程消耗 174 个时钟周期，共计 442 个时钟周期；哈希生成 $\boldsymbol{s}_i,\boldsymbol{r}_i,\boldsymbol{e}_i$ 多项式的随机比特流时输入数据 σ 长度为 512 比特消耗 16 个时钟周期，两次输出 1088 比特消耗 68 个时钟周期，轮函数过程消耗 58 个时钟周期，共计 142 个时钟周期.

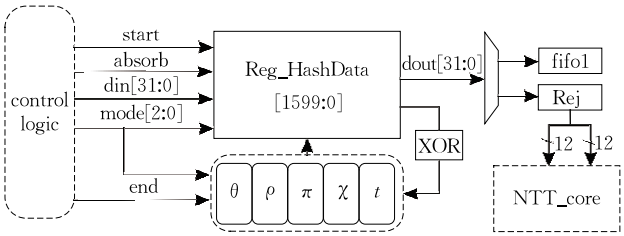

图 6　哈希核心模块结构

矩阵 $\hat{\boldsymbol{A}}_{i,j},\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 中的多项式系数通过拒绝采样模块得到，由于哈希输出位宽为 32 比特，在进行拒绝采样时最多可采样得到两个 12 比特或 13 比特的多项式系数，为了确保采样得到的数据按正确的顺序存入 FIFO2，拒绝采样算法的硬件设计逻辑如下：首先将 FIFO2 分为两个 bank 且输入位宽均为 13 比特，确保每个 bank 每周期最多存入一个数据；每时钟周期从随机比特流中截取两个多项式系数用于判断，当系数处于 Z_q 范围时将输出信号置高；模块内部设计一个计数器，用于确定当前周期采样得到的系数的索引值，根据索引值将奇数项存入到 bank0，偶数项存入到 bank1. CBD 模块负责实现中心二项采样生成噪声多项式，输入输出端位宽均为 32 比特，如表 1 所示 OSKR/OKAI 算法对应的 η_s，η_e 值包括(1,2,3,4)共四种情况，为了兼容四种采样结果本文 CBD 模块做如下设计：定义位宽为四比特的寄存器 $a_0[3:0]$ 和 $a_1[3:0]$ 用于存放哈希运算输出的随机比特流，当 $\eta=4$ 时分别存放四比特数据；当 $\eta=1,2,3$ 时寄存器的低位开始存入对应个比特的数据并将高位置零. 两个寄存器中比特数据的汉明重量相减可得到一个多项式系数，根据时序逻辑的需求 CBD 模块每个时钟周期输出四个系数.

4.5　移位寄存器的设计

由于不同模块的输入输出数据位宽和速率有所差异，因此需要设计编解码模块满足传输数据的规整化需求. 密文多项式 $(\boldsymbol{u},v)$ 的 Compress/Con 运算的输出结果位宽为 $4d_k/4d_v$，而哈希模块输入数

据位宽为 32 比特,需要在密钥封装时对 Compress/Con 运算结果进行编码操作生成 32 比特位宽的哈希输入数据,在解封装时对密文 c 进行解码操作进而生成多项式系数. 在进行编解码模块的硬件设计时通过移位寄存器结合 FIFO 单元实现数据速率匹配. 如图 7 所示,根据输入输出数据位宽确定移位寄存器的长度并作为输入数据的临时存放区域,每周期截取确定长度的输出数据,并使用 FIFO 单元作为输入输出数据的缓冲区.

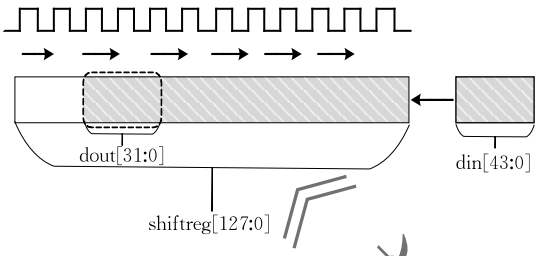

图 7 移位寄存器运行图

以(512,3329)这组参数为例,当处理密文 $\boldsymbol{u}/v$ 时编码模块的输入数据位宽为 44/20 比特,将模块的输出数据位宽设定为 32 比特可满足需求,移位寄存器 shiftreg 长度为 128. 每个时钟周期多项式运算模块的输出数据均进入 shiftreg 进行暂存,当 shiftreg 中的数据长度足够 32 比特时,当前时钟周期将输出一个数据并存入 CFIFO 模块. 由于编码模块的输出数据位宽是 32 的整数倍,可通过 CFIFO 模块将编码数据按照 32 比特位宽再送入到哈希模块.

4.6 顶层时序逻辑设计

时序逻辑设计是整个密码方案硬件实现中的关键部分,以 OSKR 算法(512,3329)这组参数的密钥封装运算过程为例,多项式运算模块的时序逻辑如下:采样模块生成的噪声多项式向量 $\boldsymbol{r}_i$ 的系数传入多项式运算模块并存储在 RAM0 单元,对多项式向量 $\boldsymbol{r}_i$ 进行正向 NTT 运算并将结果继续存储在原始位置;多项式运算模块请求公钥矩阵 $\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 并将系数存储在 FIFO2 单元,与向量 $\boldsymbol{r}_i$ 进行对应系数点乘操作,结果存入 RAM2 模块和 RAM3 模块,共需要四次点乘运算生成密文 $\boldsymbol{u}$;模块请求公钥多项式向量 $\hat{\boldsymbol{t}}_i^{\mathrm{T}}$ 并将系数存储在 FIFO2 单元,与向量 $\boldsymbol{r}_i$ 进行对应系数点乘操作,对运算结果进行 Con 运算生成密文 v;密文对($\boldsymbol{u}$,v)输出到外部的哈希模块,并计算出最终的密钥 $\boldsymbol{K}$. 参数集(256,3329)的实现与上述过程类似,如图 8 和图 9 所示.

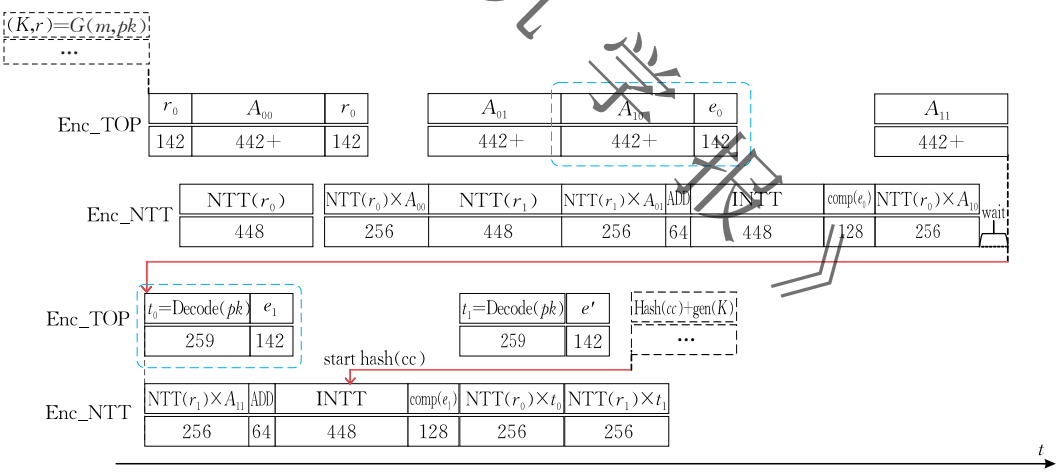

图 8 OSKR 协议 $l=2$,$n=512$ 时计算流程

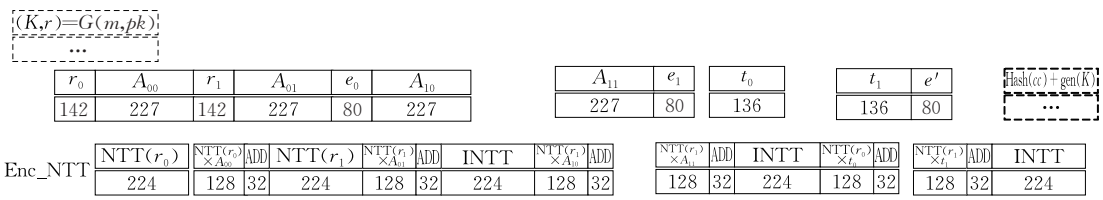

图 9 OSKR 协议 $l=2$,$n=256$ 时计算流程

根据各功能模块消耗的时钟周期数设计合理的计算流程,使得各个硬件模块不产生数据冲突,并且最大化的提高计算的并行度. 如表 3 所示参数为(512,3329)对应的时钟周期数有如下结论:(1)多项

式 $\hat{\boldsymbol{A}}_{i,j}$，$\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 采样和数论变换两种运算消耗的时钟周期数大致相同，且远大于多项式 $\boldsymbol{s}_i$，$\boldsymbol{r}_i$，$\boldsymbol{e}_i$ 采样和乘法

表 3　核心运算时钟周期数

Operation	OSKR(3329)		OKAI(7681)	
	256	512	256	512
NTT	224	448	256	512
INTT	224	448	256	512
PWM	128	256	64	256
Comp/Con	64	128	64	128
Rec	64	128	64	128
CBD	80	142	142	263
Parse	227	442	227	442

运算的时钟周期数；(2) 核心运算过程 CPAPKE.Enc 前后分别有一次哈希运算，即 Hash(pk) 和 Hash(c)，其中公钥 pk 长度为 1600 字节(12800 比特)，密文 c 长度为 1728 字节(13824 比特)，两次哈希运算需要消耗大量的时钟周期. 根据如上两个结论，本文在时序逻辑设计时做出如下改进：

(1) 如图 8 中蓝色部分所示，在采样生成噪音多项式 e' 前首先完成一次多项式 $\hat{\boldsymbol{A}}_{i,j}$，$\hat{\boldsymbol{A}}_{i,j}^{\mathrm{T}}$ 的采样过程，用于下一次的乘法运算. 上文提到拒绝采样模块与哈希模块同步运行，生成的随机比特流在 FIFO1 单元中暂存，后经拒绝采样传入到多项式运算模块的 FIFO2 单元；而噪声采样对应的随机比特流经过 CBD 模块后固定存入 FIFO1 单元，当多项式运算模块的请求信号置高时输出数据直接参与运算，即拒绝采样完成后 FIFO1 依然为空，而噪声采样后 FIFO1 单元中有数据存放. 同时在两次点乘运算之间还有加法运算、逆数论变换和压缩运算各一次，消耗时钟周期数远大于一次噪声采样运算. 因此提前进行拒绝采样并不会产生数据冲突，并且可充分利用时间资源.

(2) 两次哈希运算并非核心操作，但占据大量的时间，有必要在不降低算法安全性的前提下替换哈希过程，或者通过同步其他运算过程提高时间资源的利用率. 对于 Hash(pk) 运算本文采用了文献[31]所述的“prefix hashing”思想，取公钥中的不可预测部分进行哈希运算得到参数对 $(\bar{K}, r)$，文献证明了对公钥多项式向量 $\bar{\boldsymbol{t}}$ 编码结果的前 33 个字节进行哈希运算，与哈希整个公钥 pk 在算法安全性层面是等同的，硬件设计结果提升了 12 倍的运算效率；对于 Hash(c) 运算，由于密文 c 是由 $\boldsymbol{u}$ 和 v 两部分组成，并且根据时序逻辑可知，在对密文 v 进行 Con 操作之前已完成对密文 $\boldsymbol{u}$ 的压缩运算且结果已存入 CFIFO 单元，因此无需等到密文 v 运算完成即可提前开始哈希运算，只需确保 CFIFO 单元中最后 32 比特数据送入哈希模块之前对密文 v 运算操作已经开始，这一优化可节省大约 500 个时钟周期.

5　性能对比与分析

本文基于 Artix-7 系列芯片给出了 OSKR/OKAI 密钥封装算法的硬件实现，使用 Xilinx 官方提供的 vivado2017.4 工具进行的仿真、综合和布局布线，性能测试指标包括时序测试和硬件资源占用情况.

由表 4 可知，占用硬件资源最多的子模块是哈希模块，占据大约 50% 的 LUT 资源和 50% 的 FF 资源. 主要原因在于哈希运算的本质是根据五个轮函数的定义对输入数据进行 24 轮迭代变换，在具体的硬件实现时需要定义多个 1600 比特位宽的寄存器存储中间变量，这将消耗大量的硬件资源. 因此，本文设计的专用化哈希模块将输入输出位宽定义为 32 比特，在最大程度上降低了非必要的硬件资源消耗. 其次，多项式运算模块占用了大约 38% 的硬件资源，其中蝴蝶变换模块总共占用 2462 个 LUT 资源，用于存储数据的 RAM 模块、ROM 模块和 FIFO 模块共占用 10 个 BRAM 单元，四并行对应着四个 DSP 资源，其余资源硬件资源均消耗在外部逻辑上. 由于本文设计的多项式运算模块是四并行结构，并且蝶形模块可同时满足模值为 3329 和 7681 两种情况下的多项式运算，包括 NTT/INTT 变换、模乘/模加运算、压缩/解压缩运算和 Con/Rec 运算等，因此其蝶形模块和外部逻辑的设计更加复杂，这样的设计将增大模块的资源消耗，但同时提高了整体的硬件资源利用率. 从整体的硬件资源消耗数据来看，本文所实现的 OSKR 算法与文献[20]的工作类似，且该文献采用的是二并行多项式运算模块，但本文的整体资源消耗小于文献[20]的两倍.

表 4　子模块硬件资源消耗

Component	LUT	FF	BRAM	DSP
NTT_core	4856	3380	10	4
⌊butterfly	2462	1614	0	4
⌊fifo2	50	56	2	0
⌊fifo4	24	26	1	0
Hash_core	6426	5510	0	0
⌊Round	6272	5470	0	0
CBD	26	41	0	0
Parse	128	98	0	0
Encode	34	78	0	0
Decode	85	114	0	0
Fifo1	45	56	0	0
CFIFO	41	38	1	0

表 5 分别给出了硬件资源和实现效率两方面的整体性能对比，其中本文的($l=2,n=512$)这组参数可对应其他文献 $k=4$ 的情况. 文献[20,24-25]均给出了 CRYSTALS-KYBER 的 FPGA 硬件实现结果，与本文所实现 OSKR 算法功能类似. 文献[20]实现的是 NIST 第三轮的 CRYSTALS-KYBER 算法，参数(256,3329)，包括 $k=2,3,4$ 三种情况，论文对应的硬件实现代码包括 Client 端和 Server 端，其中 Client 端对应 KYBER 的密钥生成和密钥解封装两个过程，Server 端对应 KYBER 的密钥封装过程. 从整体的硬件资源消耗情况来看要优于本文，但是上文提到，本文工作是同时实现 OSKR/OKAI 两个密钥封装算法、共六组参数，功能上要比文献[20]更加丰富，同时由于多项式运算为四并行模式，比文献[20]的二并行模式速率更快，且硬件资源复用率更高. 根据表 5 可知，本文的设计在实现效率上要明显优于文献[20]. 文献[24]通过硬件实现了 NIST 第二轮的多种算法，其中 KYBER 算法消耗的 LUT 资源和 FF 资源与本文基本相等，不存在数量级的差别，同时本文设计的 $n=256$ 对应的四组参数均属于同一硬件电路，通过在仿真时改变片选信号的赋值进行选择，但文献[24]的三组参数是分别实现的，因此代码的兼容性较差. 文献[25]基于流水线结构设计并行 NTT 模块，但是由于没有设计专用的蝶形操作模块，在进行数论变换和乘法运算等多项式运算时需要插入大量的存储单元用于中间数据的存储和传递，使得电路设计结构较为复杂且消耗了大量的硬件资源，由表中数据可知其 BRAM 单元和 DSP 单元消耗数量分别是本文设计的 20 余倍和 50 余倍，与此同时本文的设计在效率方面也要优于文献[25]十倍左右.

表 5 时间性能运行效率对比(对比文献中的三行分别对应 $k=2/3/4$，时间性能中两列斜杠划分的三个部分对应密钥生成、密钥封装、密钥解封装；本文实现的 OSKR/和 OKAI 算法前两行和对比文献的 $k=2/3$ 比较，第三行 $l=2,n=512$ 和对比文献的 $k=4$ 比较)

Variant	平台	Time			Area			
		Freq/MHz	Cycles/($\times10^3$)	Time/μs	LUT	FF	RAM	DSP
本文								
OSKR	Artix-7	133.3	−/3.332/4.518	−/25.0/33.9	12765	11434	11	4
			−/5.091/6.745	−/38.2/50.6				
			−/5.530/7.886	−/41.5/59.2	12394	11101	12.5	4
OKAI		133.3	−/3.191/4.280	−/23.9/32.1	12765	11434	11	4
			−/4.861/6.354	−/36.5/47.7				
			−/5.925/8.398	−/44.5/63.0	12394	11101	12.5	4
相关工作								
20	Artix-7	161	3.8/5.1/6.7	23.4/30.5/41.3	7412	4644	3	2
			6.3/7.9/10.0	39.2/47.6/62.3				
			9.4/11.3/13.9	58.2/67.9/86.2				
24	Artix-7	210	−/3.0/4.4	−/14.3/20.9	11864	10348	15	8
			−/4.0/5.6	−/19.2/26.5	11884	10380		
			−/5.7/7.4	−/27.4/35.2	12183	12441		
25	Artix-7	155	−/49.0/68.8	−/316/444	88901	152875	202	354
			−/77.5/102.1	−/500/659	110260	167293		292
		192	−/107.1/135.6	−/558/706	132918	172489		548
27	RSIC-V	25	74.5/131.7/142.3	2980/5268/5692	14975	2539	14	11
			111.5/177.5/190.6	4461/7102/7623				
			148.5/223.5/241	5942/8939/9639				
28		−	150.1/193.1/204.8	−	24306	10867	32	18
			273.4/325.9/340.4					
			349.7/405.5/424.7					
29		59	710/971/870	12034/16458/14746	1842	1634	34	5
			−	−				
			2203/2619/2429	37339/44390/41169				
30	HLS	67	−/31669/43018	−475/645	1977896	194126		
15	Cortex-M4	168	920.8/1119.5/1095.9	5481.2/6663.7/6523.3	−			
14		24	−/634/597	26417/24875				
			−/1113/1059	46375/44125				
			−/1732/1653	72167/68875				

文献[27-29]使用的并非纯硬件开发方式，而是基于 RSIC-V 指令集的软硬件协同开发方案，其基本开发思路是算法中的关键且较为耗时的运算过程，例如多项式运算或者哈希运算，通过纯硬件的实现方式提升运算速度；而其余运算，主要是指逻辑运算部分，通过设计 RSIC-V 软核实现. 这种开发方式综合了硬件加速和软件灵活性两方面的特点，便于密码算法的快速实现. 从实现结果来看文献[27]和文献[28]的 LUT 资源分别为 14975 和 24306，与本文处于同一数量级，BRAM 单元和 DSP 单元的消耗略多于本文设计；而文献[29]消耗的资源数明显较少，原因在于硬件加速单元的设计差异：文献[27]中设计了专用硬件加速单元，包括 NTT 核和 Keccak 核，并且通过 RSIC-V 指令集将硬件单元整合并构建加密处理器. 文献[28]与之类似，也将 NTT、多项式乘法、Keccak 以及中心二项采样等多个部分通过硬件进行加速，因此两者消耗的硬件资源较多；而文献[29]未设计专用的硬件加速模块，主要是通过指令集实现算法功能，因此消耗的资源较少. 但从时间资源上看，与纯硬件开发方式相比，软硬结合的方式无论是频率还是时钟周期数均不占优势，最终耗时是本文设计的数十倍到数百倍，其原因在于硬件加速器与 RSIC-V 软核之间需要设计专用的接口用于数据交换，接口的带宽限制了数据传输速率导致频率无法提高.

文献[15]是基于 ARM Cortex-M4 平台的 Aigis 算法的软件实现，测试平台最高频率为 168 MHz，可对应本文的 OKAI 算法实现；文献[14]是基于 Cortex-M4 平台的 KYBER 算法的软件实现，频率为 24 MHz，可对应本文的 OSKR 算法实现. 软件的优势在于具体实现时主要考虑算法的逻辑，虽然使用指令集实现的向量并行运算可加速 NTT 等运算过程的效率，但从软件实现的整体层面来看依然是逻辑运算；而硬件需要兼顾时序逻辑、数据存储等底层逻辑，在灵活性上不如软件开发，但由于实现了整体上的并行运算，因此从表 5 所示的实现结果上可知硬件设计方案在实现效率要远优于软件方案. 另外由于指令集语言操作对象是 ARM 芯片上固定位宽的寄存器内的数据；而硬件设计是实现专用化的 RTL 级电路结构，通过设置不同位宽的 reg 和 wire 可直接操作每一比特的数据完成运算，因此在相同的条件下硬件实现理论上要比软件实现的效率高一个数量级.

6　总　结

本文基于 FPGA 平台给出了密钥封装算法 OSKR/OKAI 的硬件实现方案，包括多项式运算模块、采样模块、编解码模块等，可完整支持两个算法共六组参数的运算需求，并基于 Xilinx Artix-7 系列 FPGA 平台进行了实现. 此外，与现有工作进行了对比和分析，具体包括纯硬件实现、软件协同实现和纯软件实现三种，结果表明本方案在硬件资源和实现效率两个方面均有一定的优势，这对于推进格密码方案的工程化和实用化进程有较好的借鉴意义和参考价值.

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 Call for Proposals. 2017-01-03. https://Csrc. Nist. Gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization/Call-for-Proposals′

[3] Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. 2022-07. https://Doi. Org/10. 6028/Nist. Ir. 8413′

[4] Chinese Association for Cryptologic Research: Public key algorithms selected to the second round competition of national cryptographic algorithm competitions. 2019-09-27. http://Sfjs. Cacrnet. Org. Cn/Site/Term/List_77_1. Html′

[5] Zhang J, Yu Y, Fan S, et al. Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes//Proceedings of the IACR International Conference on Public-Key Cryptography. Cham: Springer, 2020: 37-65

[6] Bos J, Ducas L, Kiltz E, et al. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM//Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P). London, UK, 2018: 353-367

[7] Jin Z, Zhao Y. Generic and practical key establishment from lattice//Proceedings of the International Conference on Applied Cryptography and Network Security. Cham: Springer, 2019: 302-322

[8] Jin Z, Zhao Y. Optimal key consensus in presence of noise. arXiv preprint arXiv:1611. 06150, 2016

[9] Liang Z, Shen S, Shi Y, et al. Number theoretic transform: Generalization, optimization, concrete analysis and applications //Proceedings of the International Conference on Information Security and Cryptology. Cham: Springer, 2020: 415-432

[10] Shen S, He F, Liang Z, et al. OSKR/OKAI: Systematic optimization of key encapsulation mechanisms from module lattice. ACM Transactions on Privacy and Security, to appear. Full version available from: arXiv preprint arXiv: 2109.02893, 2021

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

[12] Lyubashevsky V, Peikert C, Regev O. On ideal lattices and learning with errors over rings//Proceedings of the Annual International Conference on the Theory and Applications of Cyptographic Techniques. Berlin, Germany: Springer, 2010: 1-23

[13] Langlois A, Stehlé D. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 2015, 75(3): 565-599

[14] Botros L, Kannwischer M J, Schwabe P. Memory-efficient high-speed implementation of Kyber on Cortex-M4//Proceedings of the International Conference on Cryptology in Africa. Cham: Springer, 2019: 209-228

[15] Shen Shi-Yu, He Feng, Zhao Yun-Lei. Multi-platform efficient implementation and optimization of Aigis-enc algorithm. Journal of Computer Research and Development, 2021, 58(10): 2238-2252(in Chinese)
(沈诗羽, 何峰, 赵运磊. Aigis 密钥封装算法多平台高效实现与优化. 计算机研究与发展, 2021, 58(10): 2238-2252)

[16] Zhang Jiang, Yu Yu, Fan Shuqin, et al. Supporting documentation of Aigis-enc. 2019-09-27. http://sfjs.cacrnet.org.cn/upload/5-db41c5175e9c.zip

[17] 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, 2020, 71(11): 2829-2843

[18] Roy S S, Vercauteren F, Mentens N, et al. Compact ring-LWE cryptoprocessor//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany: Springer, 2014: 371-391

[19] Roy S S, Turan F, Jarvinen K, et al. FPGA-based high-performance parallel architecture for homomorphic computing on encrypted data//Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). 2019: 387-398

[20] Xing Y, Li S. 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

[21] Mert A C, Öztürk E, Sava E. FPGA implementation of a run-time configurable NTT-based polynomial multiplication hardware. Microprocessors and Microsystems, 2020, 78: 103219

[22] 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, (1): 49-72

[23] Mert A C, Öztürk E, Sava E. Design and implementation of a fast and scalable NTT-based polynomial multiplier architecture//Proceedings of the 2019 22nd Euromicro Conference on Digital System Design(DSD). Kallithea, Greece, 2019: 253-260

[24] Dang V B, Farahmand F, Andrzejczak 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. Cryptology ePrint Archive, 2020: 795

[25] Huang Y, Huang M, Lei Z, et al. A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse. IEICE Electronics Express, 2020: 17. 20200234

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

[27] Banerjee U, Ukyab T S, Chandrakasan A P. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols. arXiv preprint arXiv:1910.07557, 2019

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

[29] Alkim E, Evkan H, Lahr N, et al. ISA extensions for finite field arithmetic: Accelerating kyber and NewHope on RISC-V. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, (3): 219-242

[30] Basu K, Soni D, Nabeel M, et al. NIST post-quantum cryptography — A hardware evaluation study. Cryptology ePrint Archive, 2019: 047

[31] Duman J, Hövelmanns K, Kiltz E, et al. Faster lattice-based KEMs via a generic Fujisaki-Okamoto transform using prefix Hashing//Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021: 2722-2737

[32] Zhu Y, Liu Z, Pan Y. When NTT meets Karatsuba: Preprocess-then-NTT technique revisited//Proceedings of the International Conference on Information and Communications Security. Cham: Springer, 2021: 249-264

[33] Zhou S, Xue H, Zhang D, et al. Preprocess-then-NTT technique and its applications to Kyber and New Hope//Proceedings of the International Conference on Information Security and Cryptology. Cham: Springer, 2018: 117-137

[34] Weimerskirch A, Paar C. Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, 2006: 224

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

ZHAO Xu-Yang, Ph. D. candidate. His main research interests include post-quantum cryptography and cryptographic engineering.

LIU Yu-Xiong, M. S. candidate. His main research interests include post-quantum cryptography and cryptographic engineering.

ZHAO Yun-Lei, Ph. D., distinguished professor. His main research interests include post-quantum cryptography, cryptographic protocols, cryptographic engineering, and theory of computing.

Background

This paper focuses on the compact hardware implementation of the lattice-based OSKR/OKAI KEM schemes based on the FPGA platform. This problem belongs to cryptographic engineering. The hardware implementation of the lattice-based cryptography scheme is very important for prompting the progress of PQC algorithm standard. Although the OSKR/OKAI schemes have AVX2 and ARM optimization, the hardware implementation of these two schemes is still blank. This paper presents a dedicated hardware design of OSKR/OKAI algorithm, the main contribution includes: (1) design a quadruple parallel polynomial operation module which can realize NTT, INTT, multiplication, compress, con/rec operation. (2) optimize the hash operation of public key based on "prefix hash" method which is nearly 12 times fast than the original way, and operate the hash operation of ciphertext which save hundreds clocks. (3) present a compact hardware implementation of OSKR/OKAI algorithms with decent performance. The previous research directions of our research group include software optimization of Aigis, Kyber, dilithium, ACKN-MLWE and so on.