Journal of Information and Intelligence 2 (2024) 167–176

Contents lists available at ScienceDirect

Journal of Information and Intelligence

journal homepage: www.journals.elsevier.com/journal-of-information-and-intelligence

A lightweight hardware implementation of CRYSTALS-Kyber

Shiyang He [a], Hui Li [a,*], Fenghua Li [b], Ruhui Ma [a]

[a] School of Cyber Engineering, Xidian University, Xi'an 710126, China
[b] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

ARTICLE INFO

Keywords:
CRYSTALS-Kyber
Learning with errors (LWE)
Post-quantum cryptography (PQC)
Application specific integrated circuit (ASIC)
Field-programmable gate array (FPGA)

ABSTRACT

The security of cryptographic algorithms based on integer factorization and discrete logarithm will be threatened by quantum computers in future. Since December 2016, the National Institute of Standards and Technology (NIST) has begun to solicit post-quantum cryptographic (PQC) algorithms worldwide. CRYSTALS-Kyber was selected as the standard of PQC algorithm after 3 rounds of evaluation. Meanwhile considering the large resource consumption of current implementation, this paper presents a lightweight architecture for ASICs and its implementation on FPGAs for prototyping. In this implementation, a novel compact modular multiplication unit (MMU) and compression/decompression module is proposed to save hardware resources. We put forward a specially optimized schoolbook polynomial multiplication (SPM) instead of number theoretic transform (NTT) core for polynomial multiplication, which can reduce about 74% SLICE cost. We also use signed number representation to save memory resources. In addition, we optimize the hardware implementation of the Hash module, which cuts off about 48% of FF consumption by register reuse technology. Our design can be implemented on Kintex-7 (XC7K325T-2FFG900I) FPGA for prototyping, which occupations of 4777/4993 LUTs, 2661/2765 FFs, 1395/1452 SLICEs, 2.5/2.5 BRAMs, and 0/0 DSP respective of client/server side. The maximum clock frequency can reach at 244 MHz. As far as we know, our design consumes the least resources compared with other existing designs, which is very friendly to resource-constrained devices.

1. Introduction

With the introduction of Shor's algorithm [1], the security of traditional cryptosystems based on mathematical hard problem integer factorization and discrete logarithm is facing huge threats and challenges [2]. Under this severe situation, post-quantum cryptography (PQC) research which can resist quantum computers, is extremely urgent and important.

Lattice-based cryptography as one of the most promising algorithms among the PQC has attracted a lot of attention. Ajtai et al. [3,4] proved that the shortest vector problem (SVP) is NP-hard under random reduction, and then the closest vector problem (CVP) is also proved to be NP-hard. This work laid the foundation for the research of lattice-based PQC schemes. In 2005, Regev et al. [5] proposed the learning-with-errors (LWE) problem and reduced the difficulty of solving the problem to the worst-case lattice problem. Meanwhile,a public-key encryption scheme based on the LWE problem was proposed. Subsequently, many scholars proposed improved schemes [6,7], which further improved the efficiency and security of the lattice-based public-key encryption scheme. Due to excessive key and ciphertext expansion (up to megabytes) caused by LWE-problem-based design, the encryption scheme has relatively low efficiency, which is hard to be applied in practice. Inspired by the idea of ideal lattice, Lindner and Peikert [8] proposed the LWE problem

* Corresponding author.
E-mail address: lihui@mail.xidian.edu.cn (H. Li).

https://doi.org/10.1016/j.jiixd.2024.02.004
Received 22 September 2023; Received in revised form 23 January 2024; Accepted 20 February 2024
Available online 24 February 2024

on the ring, and gave a lattice-based public-key encryption scheme based on the ring-learning-with-errors (RLWE) problem. The key and ciphertext size of the scheme is n times smaller than those based on the LWE problem, greatly reducing the resource occupation in the implementation, while its algebraic structure may be vulnerable to attacks [9]. In 2015, Langlois et al. [10] proposed module-learning-with-errors (MLWE) hardness assumption. By introducing a scalable polynomial vector structure, the scheme can be a trade-off between safety and efficiency. In 2018, Bos et al. [11] introduced a CCA-secure key encapsulation mechanism (KEM), namely CRYSTALS-Kyber, based on the hardness assumptions of MLWE.

Since December 2016, the National Institute of Standards and Technology (NIST) has began to solicit post-quantum cryptographic (PQC) algorithms worldwide, including public-key cryptography algorithms, KEMs, digital signatures [12]. As of December 2017, a total of 76 applications for PQC algorithms had been submitted. In December 2017, January 2019, and July 2020, after 3 rounds of comprehensive comparison and evaluation, 4 KEMs including CRYSTALS-Kyber entered the final recommendation list. Compared with other lattice-based PQC algorithms, Kyber has the advantages of higher security (CCA-secure), smaller key and ciphertext size, and ability to trade-off between security and efficiency. In July 2022, CRYSTALS-Kyber was selected by NIST as one of the standards of PQC algorithms [13].

In Kyber or other lattice-based cryptography, the most resource consuming step is polynomial multiplication, which can be typically implemented by schoolbook or NTT algorithm. Both methods are widely used in practice. NTT is a faster choice, but it requires selected parameters, complex operations, and high hardware resource costs. Schoolbook is a naive method, with low resources costs, simple implementation, and regular operation, which is very attractive in lightweight scenarios and resource-constrained applications. Recently, various papers in Kyber described the optimization of the NTT module. Chen et al. [14] improved the efficiency of the NTT module by optimizing the GS butterfly structure, adopting dual-column sequential storage and bit-reversed address accessing schemes. Zhang et al. [15] also optimized the NTT module in Kyber, proposing a pipelined butterfly arithmetic unit and adopting the doubled bandwidth ping-pong memory access scheme to eliminate the bottleneck of RAM access. At the same time, little research is focused on the trade-off between performance and resource cost for SPM. Pöppelmann et al. [16] designed a row-wise SPM with two counters and a modular multiplication core. The row-wise SPM can be implemented on Xilinx Spartan-6 FPGA but the throughput is not very high. Liu et al. [17] proposed an optimized SPM by performing two multiplications using one DSP block on FPGA. The optimized SPM can get $2\times$ speedup without additional hardware resource consumption. A highly efficient design for SPM was proposed in [18], the optimized SPM can reduce its time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^2/4)$. However, there is not much research on Kyber scheme hardware implementation, especially the latest version of Kyber (version 3.01). [19] and [20] are two of the few papers that proposed complete hardware implementation of CRYSTALS-Kyber scheme, but they cost too much resources.

This brief proposes a lightweight hardware design for CRYSTALS-Kyber KEM protocol on ASIC and implement on Xilinx Kintex-7 FPGA for prototyping. The contributions of this paper are as follows:

We propose a novel modular multiplication unit (MMU) which consists of multiplier part and modular reduction part. A signed number representation is used to represent the coefficients from binomial sampling with 3 bits. Compared with the naive 12-bit representation, it can save considerable hardware resources. At present, a large number of designs are based on NTT architecture, but due to the limitations of NTT architecture itself, the resource consumption is large. We put forward a specially optimized SPM without DSP instead of NTT core for polynomial multiplication, which can reduce about 74% resource cost. Meanwhile, true-dual-port BRAMs are used to meet the bandwidth requirements of SPM. We implement SHA3-256, SHA3-512, SHAKE-128, and SHAKE-256 in an optimized Hash module, which cut off about 48% FF consumption by register reuse technology. In addition, we exploit a compact and efficient structure for the compression/decompression module to avoid division or multiplication operations. As far as we know, our design consumes the least resources compared with other existing designs, which is very friendly to resource-constrained devices. For example, in embedded systems and Internet of Things (IoT) devices, they often have limited computational resources and energy supply. Our optimized implementation can be used to ensure that these devices maintain energy efficiency while also ensuring a high level of security. In satellite communication applications, both satellite bandwidth and energy are extremely precious. The optimized implementation can help reduce energy consumption on the communication link and provide secure data transmission. In mobile communication scenarios, the security of device communication is often limited by cost and energy. The optimized implementation can provide secure communication protocols for devices such as smartphones and tablets, while reducing costs.

Our paper is organized as follows. The preliminaries of CRYSTALS-Kyber KEM and SPM are introduced in Section 2. Section 3 explains the design rationale of each module and describes the entire structure in the implementation of Kyber scheme. Section 4 presents the hardware implementation results of the proposed design and comparison with other related works. Section 5 draws the conclusions.

2. Preliminary

2.1. Notation

Let R denote the ring $Z[X]/(X^n+1)$, where $n = 2^{n'-1}$ such that X^n+1 is the $2^{n'}$th cyclotomic polynomial. Let R_q denote the ring $Z_q[X]/(X^n+1)$, where each coefficient of the n-degree polynomial is reduced modulo q. The elements in R_q are all expressed in regular font letters, polynomial vectors are expressed in bold lowercase letters, and polynomial matrixes are expressed by bold uppercase letters. For a vector $\mathbf{v}$ or matrix $\mathbf{A}$, the ith entry within a polynomial vector $\mathbf{v}$ is denoted by $\mathbf{v}[i]$, and the entry lies in ith row, jth column in polynomial matrix $\mathbf{A}$ is denoted by $\mathbf{A}[i][\mathrm{j}]$.

2.2. CRYSTALS-Kyber KEM

CRYSTALS-Kyber (part of CRYSTALS cryptographic suite for algebraic lattices) is an IND-CCA-secure KEM, which has first been introduced in [11]. The security of Kyber is based on the hardness of MLWE [10]. The construction of Kyber follows two steps: First introduce a CPA-secure public-key encryption scheme, then turn it into a CCA-secure KEM through the tweaked Fujisaki-Okamoto transform. From the 1st to 3rd round of evaluation by NIST, Kyber has been updated several times. Parameter sets of the latest 3.01 version is shown in Table 1.

2.3. SPM

Schoolbook algorithm can be written as shown in equation (1), where $\mathbf{a}, \mathbf{b} \in Z_q[X] / (X^n + 1)$. The product $\mathbf{a} \cdot \mathbf{b}$ can be computed by the special rule that $X^n \equiv -1$, and each coefficient in polynomials is implicitly reduced module q. The operation of module n can be performed efficiently by bit shifting because of n is a power of 2, and the value of $(-1)^{\lfloor (i+j)/n \rfloor}$ can be performed by comparison of $i+j$ and n. SPM has two types of hardware implementation approaches. The first approach is row-wise multiplication, in which one coefficient of polynomial $\mathbf{a}$ is fixed and it is multiplied by all the coefficients of the polynomial $\mathbf{b}$. This operation is performed in sequence until all the multiplication calculations are completed. Then, accumulation operation is performed to get final result. The second approach is column-wise multiplication, where a full column of the result is computed before moving to the next column. SPM is a naive method, with a complexity of $\mathcal{O}(n^2)$ in Z_q and requires n^2 modular multiplication and $(n-1)^2$ addition or subtraction operations. Although slower compared with NTT, it offers regular operations, low resources cost, and simple implementation. SPM is widely used in lattice-based cryptography [16,17].

$$\mathbf{a} \cdot \mathbf{b} = \left[\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \mathbf{a}[i]\mathbf{b}[j]X^{i+j} \right] \bmod < X^n + 1 >= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (-1)^{\lfloor (i+j)/n \rfloor} \mathbf{a}[i]\mathbf{b}[j]X^{(i+j) \bmod n}. \tag{1}$$

3. Hardware implementation

In this section, we give details of key modules used in our implementation, where the modules contain MMU, SPM, Hash, compression/decompression and others. Later, we present the lightweight architecture for overall implementation of Kyber.

3.1. Modular multiplication unit

In Kyber, the polynomial coefficients of **sk** and **r** are derived from binomial sampling, where the sampling value is small integers in the range [−2,2] or [−3,3] depending on the value of $\eta = 2$ or $\eta = 3$ respectively. As a result, different from other lattice-based cryptography hardware implementations using DSP block to complete multiplication, we propose a novel implementation method which can save about 80% resource consumption compared with DSP block. Taking $\eta = 3$ as an example, the structure of MMU, as shown in Fig. 1, consists of two parts, the multiplier and the modular reduction.

In the multiplier part, we adopt the signed number to represent the coefficients with 3 bits, where 1 bit is the sign bit and 2 bits is the data bit. Compared with the naive 12-bit representation, it can save a lot of memory resources. The value of data bits is limited to [0,3], so we can use bit shifting, adder, and multiplexer instead of DSP to complete the multiplication.

Modular reduction is carried out in two steps. The first step is an unsigned modular reduction which can be reduced into [0,3329) by at a maximum of two subtractions of q. The second step is a signed modular reduction which is shown in equation (2):

$$-a \bmod q = q - (a \bmod q). \tag{2}$$

In Kyber, the MMU circuit will be more concise and efficient if the parameter $\eta = 2$, because the value of data bits will be limited to [0,2].

3.2. SPM in Kyber

Polynomial multiplication is the most critical and costly part of the overall implementation. [19,20] used NTT to implement polynomial multiplication, because it can reduce the computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. However, it also brings many complex operations such as pre-computation, array reordering, post-computation and so on. In Kyber (version 3.01), $n = 256$ and $q = 3329$ are chosen such that Z_q contains nth primitive roots of unity, but not $2n$th primitive ones. As a consequence, more complicated

Table 1
Parameter sets for Kyber.

	n	k	q	η_1	η_2	(d_u,d_v)	δ
KYBER512	256	2	3329	3	2	(10,4)	2^{-139}
KYBER768	256	3	3329	2	2	(10,4)	2^{-164}
KYBER1024	256	4	3329	2	2	(11,5)	2^{-174}

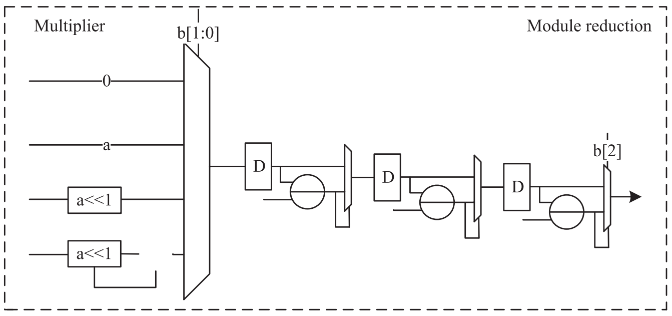


Fig. 1. Hardware structure of the MMU.

operations are needed to divide polynomial into two 128-term polynomials based on the parity of the indexes. In addition, schoolbook, as a naive method, has low resources cost, simple implementation, and regular operation, which is very attractive in lightweight scenarios and resource-constrained applications. In Kyber, schoolbook algorithm can not only avoid the complex transformation in NTT, but also use the MMU mentioned above to complete the multiplication operation, which can save a lot of hardware resources. Since NTT is bijective, polynomial coefficients sampled from NTT domain are equivalent to those sampled from Z_q [11]. Based on the above reasons, we propose to use SPM with MMU to achieve polynomial multiplication.

Combined with the work of Zhang et al. [18], we further optimize SPM for Kyber. The optimized SPM can reduce time complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^2/4)$. The pseudo code of the SPM algorithm is shown in Algorithm 1. The details are described as the following 4 steps:

Algorithm 1: Optimized SPM With MMU

```
Input: $\mathbf{a}(x)$, $\mathbf{b}(x)$, (polynomial in $Z_q[x]/(x^n + 1)$); $q$, (prime modular).
Output: $c0$, $c1$, $c2$, $c3$, ([0, $q$)).
1: for $i = 0 : 4 : n - 4$ do
2:     $c0 \leftarrow 0$.
3:     $c1 \leftarrow 0$.
4:     $c2 \leftarrow 0$.
5:     $c3 \leftarrow 0$.
6:     for $j = 0 : n - 1$ do
7:         $a0 \leftarrow \mathbf{a}[j]$.
8:         $a1 \leftarrow \mathbf{a}[(j + 2) \bmod n]$.
9:         $b0 \leftarrow \mathbf{b}[(i - j) \bmod n]$.
10:        $b1 \leftarrow \mathbf{b}[(i - j + 1) \bmod n]$.
11:        temp0 $\leftarrow$ MMU($a0$, $b0$).
12:        temp1 $\leftarrow$ MMU($a0$, $b1$).
13:        temp2 $\leftarrow$ MMU($a1$, $b0$).
14:        temp3 $\leftarrow$ MMU($a1$, $b1$).
15:        $c0 \leftarrow (c0 + \text{temp0}) \bmod q$.
16:        $c1 \leftarrow (c1 + \text{temp1}) \bmod q$.
17:        $c2 \leftarrow (c2 + \text{temp2}) \bmod q$.
18:        $c3 \leftarrow (c3 + \text{temp3}) \bmod q$.
19:        if $j == n - 1$ then
20:            return $c0$, $c1$, $c2$, $c3$.
21:        end if
22:    end for
23: end for
```

(1) Clearing register $c0$, $c1$, $c2$, $c3$ (lines 2~5).
(2) Reading coefficients $\mathbf{a}[j]$, $\mathbf{a}[(j + 2) \bmod n]$ from RAMA and $\mathbf{b}[(i - j) \bmod n]$, $\mathbf{b}[(i - j + 1) \bmod n]$ from RAMB in one clock cycle (lines 7~10).
(3) Calculating the product of $\mathbf{a}[j]$, $\mathbf{a}[(j + 2) \bmod n]$ and $\mathbf{b}[(i - j) \bmod n]$, $\mathbf{b}[(i - j + 1) \bmod n]$ in one clock cycle using MMU (lines 11~14).
(4) Accumulating the calculation results (mod q). if $j = n - 1$, return $c0$, $c1$, $c2$, $c3$. Else, continue to next loop iteration (lines 15~23).

An overall structure of SPM is shown in Fig. 2. We use two true-dual-port BRAMs (RAMA and RAMB) to store the public key **pk**, matrix **A** (12 bits in width) and vector **s**, **e**, **r** (3 bits in width), respectively. The calculation output adopts the register structure, which not only facilitates the data processing of the next module, but also saves one BRAM. For the multiplier, we use 4 MMUs to complete 4 multiplications in one clock cycle. For further performance optimization, we can also increase the memory width and store two or more coefficients in the same memory address. By equipping the SPM with 8 or more MMUs, we can make SPM 2 times or faster.

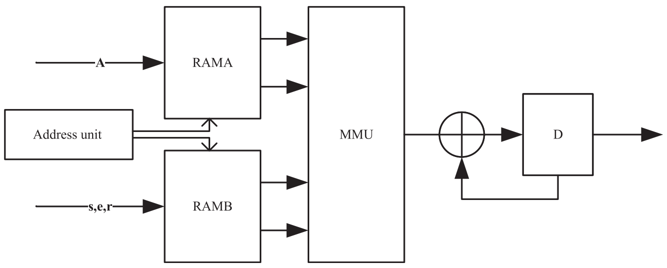


Fig. 2. Hardware structure of the optimized SPM.

3.3. Hash module

Secure Hash Algorithm 3 (SHA3) as the latest member of the Secure Hash Algorithm family of standards, was released by NIST on August 5, 2015. SHA3 consists of four cryptographic hash functions and two extendable-output functions (XOFs). The four cryptographic hash functions are SHA3-224, SHA3-256, SHA3-384, and SHA3-512, and the two XOFs are SHAKE-128 and SHAKE-256 [21].

As shown in Fig. 3, the sponge construction, a framework for SHA3, contains three components including an underlying function on fixed-length strings (denoted by *f*), a parameter called the rate (denoted by *r*), and a padding rule (denoted by pad). The rate *r* is a positive integer depending on the cryptographic hash functions or XOFs. The capacity, denoted by *c*, is the positive integer. Pad is a function that produces padding. In simple terms, it makes the length of output string positive multiple of *r*.

Each of the cryptographic hash functions and XOFs employs the same underlying function Keccak-*f*[1600], but with different rate *r* and different message suffix appended for domain separation. Keccak-*f*[1600] uses a permutation as a building block, where the permutation contains 24 rounds and each round consists of a sequence of five transformations (θ, ρ, π, χ, τ).

In Kyber, several Hash functions including SHA3-256, SHA3-512, SHAKE-128 and SHAKE-256 are involved. In order to save resource consumption, we implement SHA3-256, SHA3-512, SHAKE-128, and SHAKE-256 in an optimized Hash module. The bases of these hash functions and implementations are the Keccak algorithm and some other open-source codes [21,22]. The architecture of the Hash module which uses a permutation calculation as the bottom module is illustrated in Fig. 4.

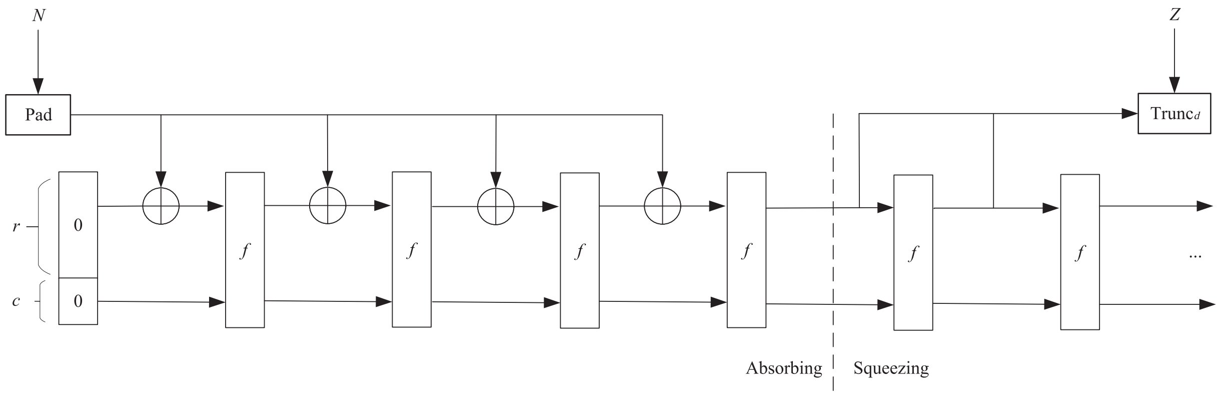


Fig. 3. The sponge construction.

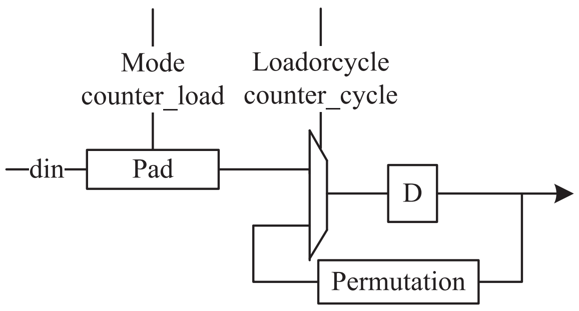


Fig. 4. Hardware structure of the Hash.

In the Hash module, signal loadorcycle is used to select the data padding or data permutation steps which are operated in sequence. The signal model is used to distinguish different SHA3 functions. In the padding part, the width of the data input is set to 64 bits per clock cycle. A counter (counter_load) and some simple logic circuits are used to complete data padding, which consume 9~21 clock cycles depending on SHA3 functions. If the padding is completed, the signal loadorcycle will be high, and then permutation calculation part will work. We use combinational logic to implement the permutation process, making a counter (counter_cycle) select the round constant, and 1600 registers store the output. A complete permutation requires 24 rounds, and consumes 24 clock cycles. Due to sequential operation, the padding part and permutation part reuse the same 1600 registers, which can save about 48% of the register consumption compared with [22].

3.4. Uniform sampling and binomial sampling

In Kyber, we use the reject sampling method to sample in R_q that are statistically close to a uniformly random distribution. We can easily implement reject sample module with a comparator and some combinational logic circuits. The noise polynomials $\mathbf{s}$, $\mathbf{e}$, $\mathbf{r}$, $\mathbf{e}'$, and element e'' are sampled from a centered binomial distribution with parameter $\eta = 2$ or $\eta = 3$, while in Kyber each sample consumes 4 or 6 pseudo random bits depending on η. For example, if $\eta = 2$, we use $b[3{:}0]$ to represent 4 random bits, and then the output sample value would be $b[0] + b[1] - b[2] - b[3]$, lying within $[-2, 2] \in Z$. The sample would be represented in signed number representation as described above with 3 bits. Both uniform sampling and binomial sampling are easy to implement in circuits with constant time.

3.5. Compression and decompression module

In Kyber, we define the function $\text{compress}_q(x, d)$ and $\text{decompress}_q(x, d)$ as follows, where $x \in Z_q$, d is an integer and $d < \lceil \log_2(q) \rceil$. The two functions are used to discard some low-order bits in the ciphertext which do not have much effect on decryption, thus reducing the size of ciphertexts. In addition, both functions can perform the usual LWE error correction during encryption and decryption:

$$\text{compress}_q(x, d) = \left\lceil \left(2^d / q\right) \cdot x \right\rfloor \bmod^{+} 2^d, \tag{3}$$

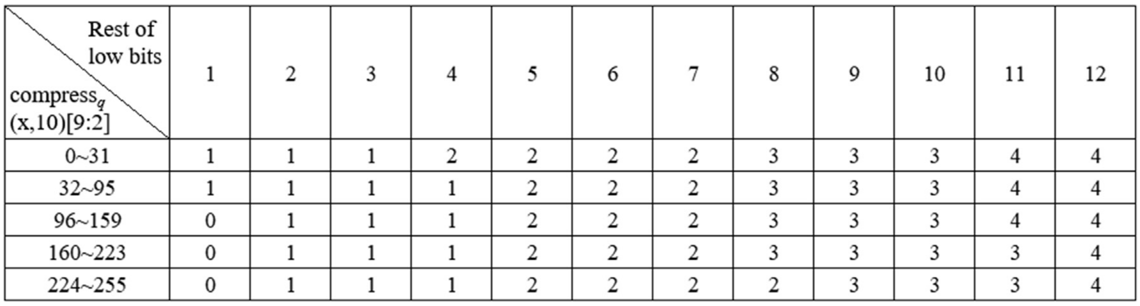
$$\text{decompress}_q(x, d) = \left\lceil \left(q / 2^d\right) \cdot x \right\rfloor. \tag{4}$$

In FPGA, the division structure is very complicated, and most related schemes use DSPs to complete function calculations. We propose novel techniques for the compression/decompression module to avoid division operations and DSPs.

In the compression module, taking $d = 10$ as an example, we find out the mapping between data x and $\text{compress}_q(x, 10)$ using equation $3328 = 2^{10}/4 \times 13$. According to the mapping relationship, we can use the 8-time dichotomy to determine the first 8 bits of $\text{compress}_q(x, 10)$. The 2 low bits of $\text{compress}_q(x, 10)$ can be determined by the first 8 bits and the rest after the dichotomy shown in Fig. 5, which can be easily implemented by a multiplexer. The circuit architecture of the compression module is shown in Fig. 6. Some registers are inserted into the circuit in order to reduce the critical path.

In the decompression module, the situation is exactly the opposite of that in the compression module. Taking $d = 10$ as an example, we divide $[0, 1024) \in Z$ into 256 data sets in ascending order. It is not difficult to find that the mapping relationship between each data set and decompress, $_q(x, 10)$ is fixed (except when $x = 512$), and four numbers in each data set also have fixed mapping with decompress $_q(x, 10)$ (except when $x = 257$ or $x = 771$). We make some logical adjustments for the special location, so decompress, $_q(x, 10)$ can be expressed as follows, in the equation (5), ! symbol represents the bitwise NOT operation, and ⊙ symbol represents the bitwise XOR operation:

$$\begin{aligned} &x[9](13 \times 128 + 1) + x[8](13 \times 64) + x[7](13 \times 32) + x[6](13 \times 16) + \\ &\quad x[5](13 \times 8) + x[4](13 \times 4) + x[3](13 \times 2) + x[2](13 \times 1) + \\ &\quad x[1](7 - x[9]) + x[0](3 + !x[9]x[8]!x[1] + x[9]x[8]x[1]) \end{aligned} \tag{5}$$



Rest of low bits / compress_q (x,10)[9:2]	1	2	3	4	5	6	7	8	9	10	11	12
0~31	1	1	1	2	2	2	2	3	3	3	4	4
32~95	1	1	1	1	2	2	2	3	3	3	4	4
96~159	0	1	1	1	2	2	2	3	3	3	4	4
160~223	0	1	1	1	2	2	2	3	3	3	3	4
224~255	0	1	1	1	2	2	2	2	3	3	3	4

Fig. 5. The relationship between $\text{compress}_q(x, 10)$ and the rest of low bits.

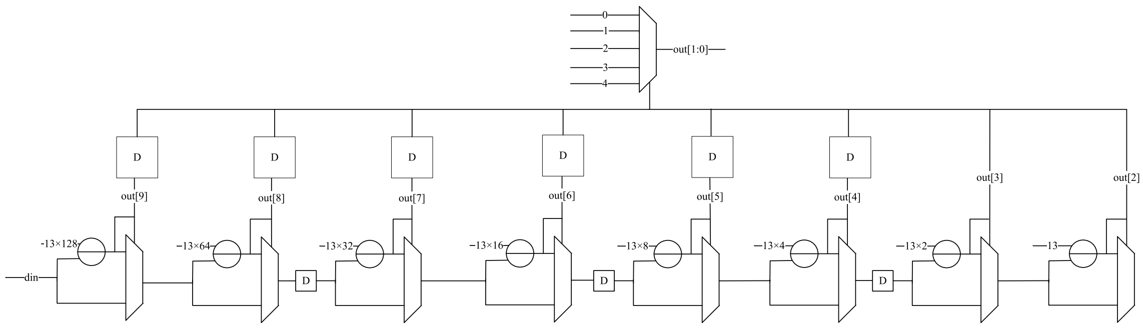


Fig. 6. Hardware structure of the compression.

Equation (5) can be further simplified as:

$$\begin{aligned} &x[9](1665) + x[8](832) + x[7](416) + x[6](208) + x[5](104) + x[4](52) + \\ &x[3](26) + x[2](13) + x[1](7) + x[0](3) + x[8]x[0](x[9] \odot x[1]) - x[9]x[1] \end{aligned} \tag{6}$$

3.6. Encode and decode module

Encode module is used to pack polynomials into byte arrays (32 bits), while the decode module is just the opposite. Polynomial coefficients in Kyber (for example 12 bits, d_u, d_v) can not always be divided by 32. Taking $k = 4$, $d_u = 11$ as an example, a long buffer ($11 \times 32 = 352$ bits) is used to deal with length mismatch between coefficients in polynomial and byte array in [23], and the structure is simple and easy to implement, while large in memory consumption. We refer to the work of Xing et al. [19], and use the shift register to implement encode/decode operation with a 16 to 1 multiplexer and 52 register bits.

3.7. Overall architecture of CRYSTALS-Kyber

The hardware architecture is designed for CCA-seure KEM CRYSTALS-Kyber with key generation, encryption, and decryption. The overall architecture is illustrated in Fig. 7. The CRYSTALS-Kyber's client side and server side share the same architecture. The Hash module is used to implement the SHA3 algorithm including SHA3-256, SHA3-512, SHAKE-128, SHAKE-256, which could complete Hash calculations and generate random bits. Reject sampler and binomial sampler perform sampling matrix **A**, sk **s**, noise **e** and **r** from random bits. Encode/decode modules are used to perform message encoding/decoding respectively, and compression/decompression modules are used to perform ciphertext compression/decompression respectively. The structure and design rationale of these modules have been detailed above. We use three fifos to implement data cache. The fifoa and fifob are located behind the reject sampler and binomial sampler respectively, the size of fifoa and fifob are same as RAMA and RAMB in SPM module. The fifoc is located behind the

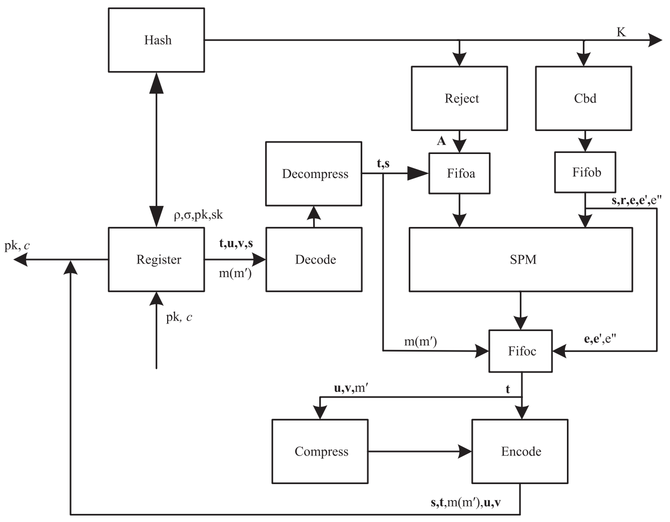


Fig. 7. Overall hardware structure of CRYSTALS-Kyber.

Table 2
Resource consumption of client/server side.

Module	LUT	FF	SLICE	BRAM	DSP
SPM	443	351	180	1	0
Hash	2826	1629	770	0	0
Sampling	57	53	21	0	0
Compress	47/56	67/75	30/32	0	0
Decompress	0/19	0/39	0/14	0	0
Encode	139/146	44/52	42/48	0	0
Decode	23/210	49/53	18/77	0	0

Table 3
Comparisons with other designs.

	LUT	FF	SLICE	BRAM	DSP	Freq. (MHz)	Time (μs)	Platform
[20]	88,901	152,875	–	202	354	155	-/316/444	ARTIX-7
[24]	14,975	2539	4173	14	11	25	2980/5268/5692	40 nm CMOS
[23]	24,950	10,720	–	2	0	150	18.4/26.9/33.6	XCZU9EG
[19]	7412	4644	2126	3	2	161	23.4/30.5/41.3	ARTIX-7
This work(client/server)	4777/4993	2661/2765	1395/1452	2.5	0	244	278/416/552	KINTEX-7

SPM module and arranged as 12 bits in width, 256 in piece-depth, it can save polynomial operation results like **t**, **u**, **v**. The register module is used to store some data for subsequent calculations, for example, the seed d, the coin r, etc.

4. Results and comparison

The proposed design has been described by Verilog HDL and simulated, synthesized, implemented on a Kintex-7 (XC7K325T-2FFG900I) FPGA using Xilinx Vivado 2019.1 for prototyping, with all building blocks implemented in hardware. In this implementation, a parameter set for Kyber ($n = 256$, $k = 2$, $q = 3329$, $\eta_1 = 3$, $\eta_2 = 2$, $d_u = 10$, $d_v = 4$) is selected.

Table 2 lists the detailed hardware resource consumption of some important modules on the client/server side after synthesis and implementation. It is not difficult to see that the Hash module consumes the most resources. It occupies 2826 LUTs, 1629 FFs, 770 SLICEs, and accounts for more than 56% LUT and 58% FF of total consumption. In the SPM module, RAMA (12 bits $\times$ 256) and RAMB (3 bits $\times$ 256) consume 1 BRAM (18 K). The fifoa, fifob and fifoc consume 1.5 BRAM (18 K).

In Table 3, the results show that our design (client/server) occupies 4777/4993 LUTs, 2661/2765 FFs, 1395/1452 SLICEs, 2.5/2.5 BRAMs, and 0/0 DSP. The consumption of DSP in both client and server is 0 because of the application of novel designed MMU, SPM, and compression/decompression module in the hardware implementation. The maximum clock frequency that the client and server can reach is 244 MHz. Key generation, encapsulation and decapsulation phase which is separated by slashes can be completed in 278/416/552 μs on our design.

The comparison results with related works, in terms of resource utilization, key generation, encapsulation and decapsulation process timing and performance, etc. is demonstrated in Table 3.

Huang et al. [20] presented a pure hardware implementation of CRYSTALS-Kyber algorithm on FPGA, and they accelerate polynomial multiplication by using the NTT algorithm. Some BRAMs are inserted between functional modules for communication purpose. They also reusing most of the functional modules to achieve maximizing resource utilization. With similar performance, The resource consumption corresponding to LUT and FF of our design is more than 17 times and 86 times smaller.

Banerjee et al. [24] proposed a lattice cryptography processor with configurable parameters called Sapphire, which is coupled with a low-power RISC-V. Cryptography processor was fabricated in TSMC 40 nm low-power CMOS process. They use a low-power modular arithmetic core to accelerate polynomial arithmetic operations, a single-port SRAM-based NTT memory architecture to save resource and other novel technology to further improve performance. Comparing with our design, even though Sapphire has improved performance, it consumes more than 3 times of LUTs and 5 times of BRAMs. Besides, it uses up to 11 DSPs, while our design uses zero.

Roy et al. [23] presented an instruction set coprocessor architecture for lattice-based cryptography which implements Saber as a case. Saber is another lattice-based KEM protocol, basing its security on module learning-with-rounding problem. Saber and Kyber have the same dimension of polynomial ring parameter $n = 256$. In order to enhance the performance, a parallel polynomial multiplier architecture is proposed that can accomplish a full multiplication in 256 cycles, However, this architecture is optimized for performance, it consumes a lot of resources. Compared with our proposed design, the utilization of LUTs and FFs is more than 5 times and 4 times larger respectively.

Xing et al. [19] put forward a scheme with an outstanding comprehensive performance. NTT core designed by Xing et al., can achieve decent performances with limited resources consumption. Compared with their NTT core, our SPM is more lightweight (consume 25% LUT, 30% FF), does not require additional BRAMs to store twiddle factors ω and select dimension parameters of the polynomial ring. Although our design has no performance advantage compare with Xing et al.'s schemes, it has a simple structure and less resource consumption. Meanwhile, our design consumes 67% LUT, 59% FF and can be implemented without DSPs. In ASIC implementation, considering the area of DSP, our design will use fewer gates.

5. Conclusion

In this work, we propose a pure hardware implementation of the PQC algorithm CRYSTALS-Kyber. We propose a novel MMU without DSP. A signed number representation is used to represent the coefficients which cut off 75% memory consumption compared with naive 12-bit representation. We put forward a specially optimized SPM instead of NTT core for polynomial multiplication, which can reduce about 74% SLICE cost. Meanwhile, true-dual-port BRAMs are used to meet the bandwidth requirements of SPM. We implement SHA3-256, SHA3-512, SHAKE-128, and SHAKE-256 in an optimized Hash module, which cut off about 48% FF consumption. In addition, we exploit a compact and efficient structure for the compression/decompression module to avoid division or multiplication operations. An lightweight overall architecture is implemented on FPGA for prototyping. Compared with other designs, we have obvious advantages in resource consumption.

Acknowledgments

This work was supported in part by the Shaanxi Province Key R&D Program (2019ZDLGY12-09), in part by the Higher Education Discipline Innovation 111 project (B16037), in part by the Shaanxi innovation team project (2018TD-007), in part by the China National Natural Science Foundation (62102298). The review of this article was coordinated by Dr. Jianbing Ni.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P.W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in: Proceedings of the IEEE Annual Symposium on Foundations of Computer Science (FOCS), IEEE, Piscataway, 1994, pp. 124–134.

[2] J. Buchmann, A. May, U. Vollmer, Perspectives for cryptographic long-term security, Communications of the ACM (CACM) 49 (9) (2006) 50–55.

[3] M. Ajtai, Generating hard instances of lattice problems, in: Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (STOC), 1996, pp. 99–108.

[4] M. Ajtai, C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence, in: Proceedings of the Twenty-ninth Annual ACM Symposium on the Theory of Computing (STOC), ACM, New York, 1997, pp. 284–293.

[5] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, in: Proceedings of the Thirty-seventh Annual ACM Symposium on the Theory of Computing (STOC), ACM, New York, 2005, pp. 84–93.

[6] B. Applebaum, D. Cash, C. Peikert, A. Sahai, Fast cryptographic primitives and circular-secure encryption based on hard learning problems, in: Proceedings of the Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference (CRYPTO), Springer, Berlin, 2009, pp. 595–618.

[7] V. Lyubashevsky, C. Peikert, O. Regev, On ideal lattices and learning with errors over rings, in: Proceedings of the Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT), Springer, Berlin, 2010, pp. 1–23.

[8] R. Lindner, C. Peikert, Better key sizes (and attacks) for LWE-based encryption, in: Proceedings of the Topics in Cryptology–CT-RSA 2011: The Cryptographer's Track at RSA Conference (CT-RSA), Springer, Berlin, 2011, pp. 319–339.

[9] C. Peikert, How (not) to instantiate ring-LWE, in: Proceedings of the International Conference on Security and Cryptography for Networks (SCN), Springer, 2016, pp. 411–430.

[10] A. Langlois, D. Stehlé, Worst-case to average-case reductions for module lattices, IACR Cryptology ePrint Archive, 2012, p. 90.

[11] J.W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler, D. Stehle, CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM, in: Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, Piscataway, 2018, pp. 353–367.

[12] National Institute of Standards and Technology, Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms (Online), December 2016. https://federalregister.gov/d/2016-30615.

[13] National Institute of Standards and Technology, NIST to Announce PQC Algorithm Selections, on July 5, 2022 (Online). https://quantumcomputingreport.com/nist-to-announce-pqcalgorithm-selections-on-july-5-2022/.

[14] Z. Chen, Y. Ma, T. Chen, J. Lin, J. Jing, Towards efficient Kyber on FPGAs: A processor for vector of polynomials, in: Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, Piscataway, 2020, pp. 247–252.

[15] C. Zhang, D. Liu, X. Liu, X. Zou, G. Niu, B. Liu, Q. Jiang, Towards efficient hardware implementation of NTT for Kyber on FPGAs, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, Piscataway, 2021, pp. 1–5.

[16] T. Pöppelmann, T. Güneysu, Area optimization of lightweight lattice-based encryption on reconfigurable hardware, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, Piscataway, 2014, pp. 2796–2799.

[17] W. Liu, S. Fan, A. Khalid, C. Rafferty, M. O'Neill, Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on FPGA, IEEE Transactions on Very Large Scale Integration Systems (VLSI) 27 (10) (2019) 2459–2463.

[18] Y. Zhang, C. Wang, D.E.S. Kundi, A. Khalid, M. O'Neill, W. Liu, An efficient and parallel R-LWE cryptoprocessor, IEEE Transactions on Circuits and Systems II: Express Briefs 67 (5) (2020) 886–890.

[19] Y. Xing, S. Li, A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA, IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) 2 (2021) 328–356.

[20] Y. Huang, M. Huang, Z. Lei, J. Wu, A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse, IEICE Electronic Express 17 (17) (2020) 234–238.

[21] National Institute of Standards and Technology, FIPS PUB 202 - SHA3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2015. https://csrc.nist.gov/pubs/fips/202/final.

[22] T. Oder, T. Güneysu, Implementing the NewHope-Simple key exchange on low-cost FPGAs, in: Proceedings of the Progress in Cryptology–LATINCRYPT 2017: 5th International Conference on Cryptology and Information Security in Latin America (LATINCRYPT), Springer, 2017, pp. 128–142.

[23] S.S. Roy, A. Basso, High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware, IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) (2020) 443–466.

[24] U. Banerjee, T.S. Ukyab, A.P. Chandrakasan, Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols, IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) (2019) 17–61.

Shiyang He received the M.S. degree in Telecommunications Engineering from Xidian University, China, in 2016. He is currently working toward the Ph.D. degree at the school of Cyber Engineering, Xidian University, Xian Shaanxi, China. His research interests include cryptographic algorithm, hardware speedup and field programmable gate array architectures and applications.

Hui Li received B.S. degree from Fudan University in 1990, M.A.S. and Ph.D. degrees from Xidian University in 1993 and 1998. Since June 2005, he has been a professor in the School of Cyber Engineering, Xidian University, Xian Shaanxi, China. His research interests are in the areas of cryptography, wireless network security, information theory, hardware security, and network coding. He is a chair of ACM SIGSAC CHINA. He served as the technique committee chair or co-chair of several conferences. He has published more than 170 international academic research papers on information security and privacy preservation.

Fenghua Li received the B.S. degree in computer software, the M.S. and Ph.D. degrees in computer systems architecture from Xidian University, Xian, China, in 1987, 1990, and 2009, respectively. He is currently a Professor and a Doctoral Supervisor with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. He is also a Doctoral Supervisor with the Xidian University. His research interests include network security, system security, privacy computing, and cryptographic processors.

Ruhui Ma received the Ph.D. degree in Cyber Security from Xidian University, China, in 2020. She is currently a lecturer at Xidian University, China. Her research interests include wireless communication, LTE/LTE-A/5G/6G networks, and cryptography security.