Efficient Implementation of Finite Field Arithmetic for Binary Ring-LWE Post-Quantum Cryptography Through a Novel Lookup-Table-Like Method

Jiafeng Xie[1] (corresponding author), Pengzhou He[1], Wujie Wen[2]
[1]: Department of Electrical & Computer Engineering, Villanova University (jiafeng.xie;phe@villanova.edu);
[2]: Department of Electrical & Computer Engineering, Leigh University (wuw219@leigh.edu);

***Abstract*—The recent advance in the post-quantum cryptography (PQC) field has gradually shifted from the theory to the implementation of the cryptosystem, especially on the hardware platforms. Following this trend, in this paper, we aim to present efficient implementations of the finite field arithmetic (key component) for the binary Ring-Learning-with-Errors (Ring-LWE) PQC through a novel lookup-table (LUT)-like method. In total, we have carried out four stages of interdependent efforts: (i) an algorithm-hardware co-design driven derivation of the proposed LUT-like method is provided detailedly for the key arithmetic of the BRLWE scheme; (ii) the proposed hardware architecture is then presented along with the internal structural description; (iii) we have also presented a novel hybrid size structure suitable for flexible operation, which is the first report in the literature; (iv) the final implementation and comparison processes have also been given, demonstrating that our proposed structures deliver significant improved performance over the state-of-the-art solutions. The proposed designs are highly efficient and are expected to be employed in many emerging applications.**

***Index Terms*—BRLWE based scheme, finite field arithmetic, hybrid size structure, lookup table, post-quantum cryptography**

I. Introduction

Post-quantum cryptography (PQC) related research has reached an all time high due to the proof from the research community that the existing public-key cryptosystems are vulnerable to the quantum attacks [1], [2]. Many types of cryptosystems so far are under the consideration for PQC candidates [2]: the lattice-based cryptography, the code-based scheme, the isogeny-based cryptosystem, etc. Overall, the lattice-based cryptography has gained substantial attentions from research community due to its strong security proof and low computational complexity [3], [4].

Ring-Learning-with-Errors (Ring-LWE) based PQC is a very important variant of the LWE scheme (lattice-based), and is also currently under the consideration of the National Institute of Standards and Technology (NIST) PQC standardization process [4]–[6]. Apart from that, a new variant of the Ring-LWE scheme, i.e., binary Ring-LWE (BRLWE), where the binary errors are used to replace the regular Gaussian distributed errors, is proposed recently to achieve smaller implementation complexity for emerging applications [7], [8].

Since the original introduction of the BRLWE scheme in [7] with thorough security analysis, there are only a small number of reports released on the actual implementation of this scheme. In [9], the authors have introduced a novel side-channel attack resistant hardware architecture of the BRLWE scheme. After that, a pair of BRLWE cryptoprocessors are introduced in [10] to obtain low-complexity and high performance, respectively. These designs [9], [10], however, suffer from limited hardware efficiency as their architectures are directly built upon original algorithms and no dedicated algorithm-architecture co-optimization has been performed. Apart from that, the fault detection and fault resistant designs related to this scheme are recently released in [11], [12], where [11] uses the same structure of [10] and [12] is software based (due to their specific focus, we here do not include them as general hardware implementations).

Noticing the facts that: (i) the BRLWE scheme can be employed in lightweight applications and hence low resource occupation is desirable (even at the cost of extra timing a little bit); (ii) there is no hybrid size structure (suitable for flexible applications such as standard computation core and security level upgrading) ever reported in the literature, in this paper, we endeavor to propose an efficient implementation of the key arithmetic component of the BRLWE scheme through a novel lookup-table (LUT)-like method. Unlike the existing designs, the proposed work covers all the essential aspects related to the algorithm-architecture co-design. Overall, the main contributions of this paper can be summarized as:

- We have (for the first time) presented a novel algorithm-architecture co-design strategy to obtain the proposed LUT-like method as well as the corresponding algorithmic derivation process for the key arithmetic of the BRLWE scheme to match the proposed LUT-like approach.
- We have also introduced the algorithmic strategy for the hybrid size operation of the key arithmetic operation of the BRLWE scheme.
- We have introduced the corresponding hardware architecture with thorough internal structural description as well as the first hybrid size structure.
- We have given the final implementation and comparison to demonstrate the superior performance of the proposed designs over the existing ones.

The rest of the paper is organized as follows. Section II

978-1-6654-3274-0/21/$31.00 2021 IEEE

TABLE I
STAGES & OPERATIONS OF THE BRLWE SCHEME

stage	major operations
key generation	a: public parameter (Alice and Bob); r_1, r_2: polynomials with binary coefficients; Alice: $p = r_1 - a \cdot r_2 \rightarrow$ Bob; (p: public key; r_2: secret key)
encryption	e_1, e_2, e_3: three binary polynomials (errors); m: message; $\widetilde{m}$: encoded message; Bob: $c_1 = a \cdot e_1 + e_2 \rightarrow$ Alice; Bob: $c_2 = p \cdot e_1 + e_3 + \widetilde{m} \rightarrow$ Alice;
decryption	Alice: m =thresh-decoder$(c_1 \cdot r_2 + c_2)$;

TABLE II
ARITHMETIC OPERATIONS OF THE BRLWE SCHEME

stage	major arithmetic operations
key generation	PM→PA→p;
encryption	PM→PA→c_1; PM→PA→PA→c_2
decryption	PM→PA→output;

focuses on the preliminary knowledge. Section III presents the proposed method and the related algorithm. Section IV introduces the proposed structures. The comparison is provided in Section V and the conclusion is given in Section VI.

II. PRELIMINARY KNOWLEDGE

The BRLWE scheme is built on the finite field operations over the ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$. The major operations of the BRLWE scheme can be seen at Table I, where there are three stages of operations involved (key generation, encryption, and decryption) [7], [9], [10]. The detailed operations within each stage are described as follows:

- **Key generation**. In this stage, a is set as a public parameter (polynomial of integer coefficients) known by both Alice and Bob. Then, for the selected two binary polynomials r_1 and r_2, Alice does the operation of $p = r_1 - a \cdot r_2$ and then send it to Bob. Note that r_1 is discarded after this stage, r_2 is the secret key (n-bit), and the public key is p ($n\log_2 q$-bit).
- **Encryption**. In this stage, the targeted n-bit message m is firstly coded to $\widetilde{m}$ following: each coefficient of m (a binary polynomial) is multiplied with $q/2$ and then added with $(q + n/2 - 1 - i)$ (i is the coefficient degree of the related polynomial). Then, Bob uses three binary errors (polynomials) e_1, e_2, and e_3 to deliver the ciphertext ($2n\log_2 q$-bit) c_1 and c_2 to Alice.
- **Decryption**. This stage involves a threshold decoder function, as shown in Table I, where Alice obtains the desired message m through thresh-decoder$(c_1 r_2 + c_2)$. The decoder produces '1' if the coefficient of the $(c_1 r_2 + c_2)$ is in the range of $(q/4, 3q/4)$, otherwise it is '0'.

As summarized in Table II, one can see that the main arithmetic operation involved within each stage of the BRLWE scheme is actually the polynomial multiplication (PM) over ring (i.e., one polynomial has integer coefficients and the other is the binary polynomial) and a polynomial addition (PA) [9]. In this paper, we will focus on the efficient implementation of this finite field arithmetic for the BRLWE scheme.

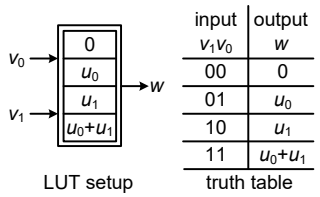

Fig. 1. The setup of the LUT and the truth table.

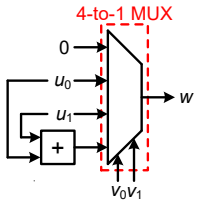

Fig. 2. The proposed method to realize (1), which functions exactly according to the truth table of Fig. 1.

Besides that, we will use the inverted range representation $(-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor - 1)$ to denote the coefficient of the integer polynomial such that the involved modular addition/subtraction can be performed without any reduction (under the two's complement representation). This strategy is originally proposed in [10], and following this denotation, all the major operations of Table II are still exactly the same though there are signs inverted on the encode and final decode operations [10].

III. PROPOSED METHOD & ALGORITHMIC OPERATION

Motivation & Consideration. It is noted the main operation of the BRLWE scheme, the PM (one integer polynomial and one binary polynomial) over ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, involves the operations of the accumulations of point-to-point multiplications (one point is integer and the other is binary value), we can thus use this property to derive efficient hardware structure specifically for this type of calculations.

Let us consider a small scale point-to-point multiplications:

$$w = u_0 v_0 + u_1 v_1, \tag{1}$$

where w and u_i ($0 \le i \le 1$) are $\log_2 q$-bit integers over $\mathbb{Z}_q$and $v_i \in \{0, 1\}$ ($0 \le i \le 1$). The calculation of (1) can actually be realized by a LUT, as seen in Fig. 1 (where the v_i is the input to the LUT and the content of the LUT covers four values as '0', u_0, u_1, and $(u_0 + u_1)$). It is clear that the 2-input LUT of Fig. 1 functions exactly as the operation in (1).

Proposed LUT-like method. The main feature of the LUT-based design (for equations (1) and (2)) is that the values of u_i (or u_j) are pre-calculated and then stored in the LUT to be read out when the matched input is fed. This strategy, however, only works when u_i (or u_j) are constant values. When u_i (or u_j) are uncertain inputs, an alternative solution is needed.

We thus use the structure of Fig. 2 to implement the operation of (1), where a 4-to-1 multiplexer (MUX) is used to replace the original 2-input LUT of Fig. 1 as well as an extra adder (to obtain $(u_0 + u_1)$ as the fourth value for the MUX). The key benefit of this type of design is that it allows the values (originally stored in the LUT) to be the variables/uncertain inputs, which facilitates the computation of (1).

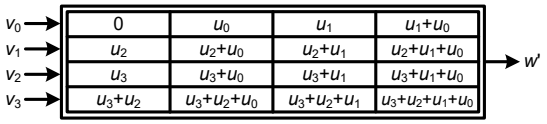

Fig. 3. The content of the 4-input LUT for (2).

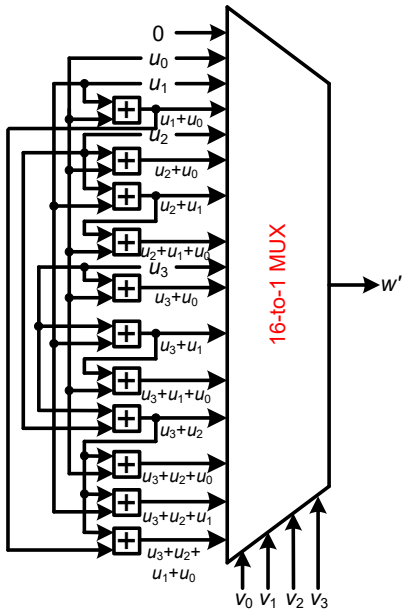

Fig. 4. The direct LUT-like method for Fig. 3 requires two many resources.

TABLE III
FPGA IMPLEMENTATION RESULTS ($\log_2 q = 8$-BIT) TO IMPLEMENT (2)

design style	#ALMs	area reduction
based on Fig. 4	94	-
equivalent form based on Fig. 2	28	70.2%

Remark. Note that the size of the LUT-like based design is limited. For example, if we extend the calculation of (1) to

$$w' = u_0v_0 + u_1v_1 + u_2v_2 + u_3v_3, \tag{2}$$

where w' and u_j $(0 \le j \le 3)$ are $\log_2 q$-bit integers over $\mathbb{Z}_q$ and $v_j \in \{0, 1\}$ $(0 \le j \le 3)$. We can then use a 4-input LUT to represent the corresponding function (Fig. 3).

Similarly, the 4-input LUT can be realized by the structure shown in Fig. 4, which consists of a 16-to-1 MUX and 11 adders to deliver the proper output, which requires significantly higher resource usage than the one of Fig. 2 (using the structure of Fig. 2 to realize Fig. 3 only requires two 4-to-1 MUXex, two related adders, and one final adder to add the two MUXes' outputs together).

Confirmation from the hardware implementation. We have also implemented the structures of Fig. 4 and also the equivalent form of that based on Fig. 2 (two 4-to-1 MUXex, two related adders, and one final adder) with VHDL ($\log_2 q = 8$, following [9], [10]) and obtained the number of adaptive logic modules (ALMs) usage through Intel Quartus Prime 17.0 based on the Stratix-V 5SGXMA9N1F45C2 Field-Programmable Gate Array (FPGA) device.

It is clear that (Table III) the proposed 2-input method of Fig. 2 achieves better performance than the equivalent form based on Fig. 4 for longer term computations. We thus set the design of Fig. 2 as our basic structure for further design.

Algorithmic process. Following the above strategy, we thus plan to derive the major arithmetic operation of the BRLWE scheme into a form to match the proposed LUT-like approach of Fig. 2. Without loss of generality, we can define the PM and PA of the BRLWE scheme (Table II) into a format of:

$$K = BD \bmod f(x) + G, \tag{3}$$

where $B = \sum_{i=0}^{n-1} b_i x^i$ $(b_i \in \{0, 1\})$, $D = \sum_{i=0}^{n-1} d_i x^i$, $G = \sum_{i=0}^{n-1} g_i x^i$, $K = \sum_{i=0}^{n-1} k_i x^i$ (d_i, g_i, and k_i are $\log_2 q$-bit integers over $\mathbb{Z}_q$), and $f(x) = x^n + 1$. Then, we have $K = b_0 D + b_1 x D + \cdots + b_{n-1} x^{n-1} D \bmod f(x) + G$ and

$$\begin{aligned} K =& b_0(d_0 + d_1 x + \cdots + d_{n-1}x^{n-1}) \bmod f(x) \\ &+ b_1 x(d_0 + d_1 x + \cdots + d_{n-1}x^{n-1}) \bmod f(x) \\ &+ \cdots \ \cdots \ \cdots \\ &+ b_{n-1}x^{n-1}(d_0 + \cdots + d_{n-1}x^{n-1}) \bmod f(x) + G, \end{aligned} \tag{4}$$

which can be substituted with $x^n \equiv -1$ $(x^n + 1 = 0)$ to have

$$\begin{aligned} K =& b_0d_0 + b_0d_1x + \cdots + b_0d_{n-1}x^{n-1} \\ &- b_1d_{n-1} + b_1d_0x + \cdots + b_1d_{n-2}x^{n-1} \\ &\cdots \ \cdots \ \cdots \\ &- b_{n-1}d_1 - b_{n-1}d_2x - \cdots + b_{n-1}d_0x^{n-1} + G, \end{aligned} \tag{5}$$

from which we can have

$$\begin{aligned} k_0 =& b_0d_0 + b_1(-d_{n-1}) + \cdots + b_{n-1}(-d_1) + g_0 \\ k_1 =& b_0d_1 + b_1d_0 + \cdots + b_{n-1}(-d_2) + g_1 \\ &\cdots \ \cdots \ \cdots \\ k_{n-2} =& b_0d_{n-2} + b_1d_{n-3} \cdots + b_{n-1}(-d_{n-1}) + g_{n-2}, \\ k_{n-1} =& b_0d_{n-1} + b_1d_{n-2} + \cdots + b_{n-1}d_0 + g_{n-1}, \end{aligned} \tag{6}$$

where one can find that for any neighboring k_i and k_{i+1}, the bits from input B stays the same while the values from input D are circularly shifted by one position with one value's sign changed (particularly, these values of D are circularly left-shifted for one position with the most left value's sign changed from k_{n-1} to k_{n-2}, ..., until k_0).

If $n = st$ (s and t are integers), we can have

$$\begin{aligned} k_0 =& \underbrace{b_0d_0 + \cdots + b_{s-1}(-d_{n-s+1})}_{\text{1st group}} \\ &+ \underbrace{b_s(-d_{n-s}) + \cdots + b_{2s-1}(-d_{n-2s+1})}_{\text{2nd group}} \\ &+ \cdots \ \cdots \ \cdots \\ &+ \underbrace{b_{n-s}(-d_s) + \cdots + b_{n-1}(-d_1)}_{t\text{th group}} + g_0 \\ =& k_{0,0} + k_{0,1} + \cdots + k_{0,t-1} + g_0, \end{aligned} \tag{7}$$

where $k_{0,0} = b_0d_0 + \cdots + b_{s-1}(-d_{n-s+1})$, ..., $k_{0,t-1} = b_{n-s}(-d_s) + \cdots + b_{n-1}(-d_1)$. When s is set as $s = 2$, then

each $k_{0,j}$ ($0 \le j \le t-1$) of (7) can be realized by the structure introduced in Fig. 2. Similarly, we can have

$$\begin{aligned} k_1 &= \sum_{j=0}^{t-1} k_{1,j} + g_1, \\ &\cdots \quad \cdots \quad \cdots \\ k_{n-1} &= \sum_{j=0}^{t-1} k_{n-1,j} + g_{n-1}, \end{aligned} \tag{8}$$

where each $k_{i,j}$ ($0 \le i \le n-1$ and $0 \le j \le t-1$) follows the same definition of (7), which greatly facilitates the employing of the structure of Fig. 2 as we set $s = 2$ and $t = n/2$.

We thus derive the proposed algorithmic operation as

Algorithm 1 Algorithmic process for $K = BD \bmod f(x) + G$

Inputs: B, D, and G from the BRLWE (D and G are polynomials with integer coefficients and B is a binary polynomial).
Outputs: $K = BD \bmod f(x) + G$.
1. Initialization step
1.1. get ready for the input operands B, D, and G.
2. Main step
2.1. for $i = 0$ to $n-1$.
2.2. $k_{n-1-i} = \sum_{j=0}^{t-1} k_{n-1-i,j} + g_{n-1-i}$. //each $k_{i,j}$ is realized by the proposed method of Fig. 2, for $s = 2$.
2.3. circularly left-shifting the coefficients of D by one position with one value's sign inverted. //following (6).
2.4. end for.
3. Final step
3.1. deliver K.

Note that in the proposed algorithmic operation, we start the calculation of k_{n-1}, and then k_{n-2}, ..., k_0, such that the n coefficients of D can directly used. Of course, we can begin the computation from k_0 to k_{n-1}, but there need preparing operations on the signs of the coefficients of D (see (6)).

Besides that, when connecting the proposed algorithmic process to the actual operations within the BRLWE scheme, especially the decryption stage (following the existing design of [10]), a final thresh-decoder is required (Table I), which can be easily realized by an XOR gate connecting with the two most significant bits (MSBs) of k_i [9].

Hybrid size operation. As indicated in (6), each k_i ($0 \le i \le n-1$) involves the multiplication of the identical bits of B with rotated coefficients of D (with sign changed once per k_i). Following this processing pattern, we can actually extend the proposed algorithm into a format of hybrid size operation.

Let us define a smaller size of the BRLWE scheme as n'. According to Algorithm 1, we can firstly calculate $k_{n'-1}$ as

$$k_{n'-1} = b_0 d_{n'-1} + b_1 d_{n'-2} + \cdots + b_{n'-1} d_0 + g_{n'-1}. \tag{9}$$

When comparing with the k_{n-1} of (6), we can define

$$D_H = d_0 + \cdots + d_{n'-1} x^{n'-1} + H(d_{n'} x^{n'} + \cdots + x^{n-1}), \tag{10}$$

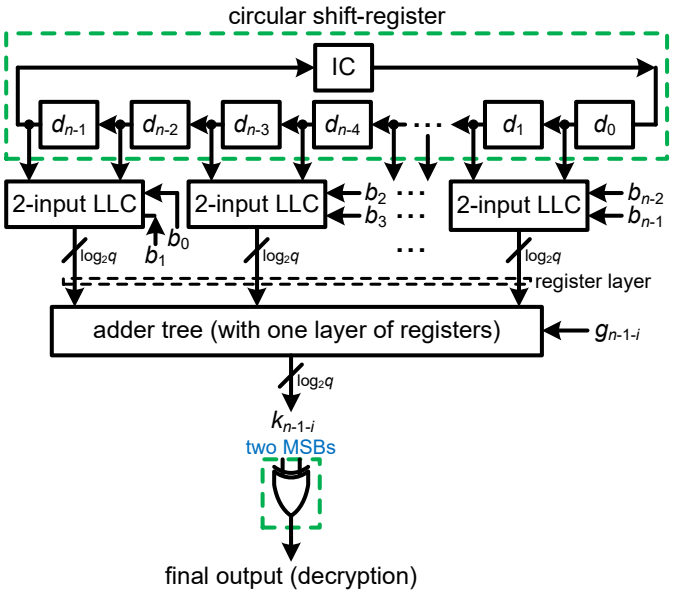

Fig. 5. The proposed BRLWE structure based on 2-input LLC ($0 \le i \le n-1$), where the values in the circular shift-register are the initial values. IC: inverter cell; LLC: LUT-like cell; MSB: most significant bit.

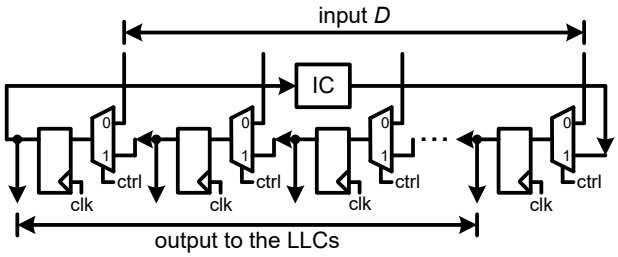

Fig. 6. The internal structure of the circular shift-register.

where D_H is the hybrid size representation of D, which can be switched from size n to n' as

$$H = \begin{cases} 1, & \text{size } n \\ 0, & \text{size } n' \end{cases} \tag{11}$$

Similarly, we can have the corresponding definitions of K_H, B_H, and G_H. It is clear that such setup facilitates the hybrid size operation: as seen from (6) and (9), we can use the value of H to determine the actual size for the BRLWE scheme, and then based on the selected size to compute $k_{n-1}/k_{n'-1}$, as well as the following $k_i/k_{i'}$ ($0 \le i \le n-1$ or $0 \le i' \le n'-1$) based on Algorithm 1. In the following section, we will also present the corresponding hardware to realize this hybrid size operation, especially on the realization of size switching.

IV. Proposed Structures for the BRLWE Scheme

Based on the algorithmic process we proposed, we can have the proposed BRLWE structure as shown in Fig. 5.

The structure of Fig. 5 mainly consists of three components, namely the circularly shift-register, the LUT-like cells (LLCs), and the adder tree (as well as an extra XOR gate to deliver the final output according to the decryption stage of Table I). The detailed functions of these components are listed below:

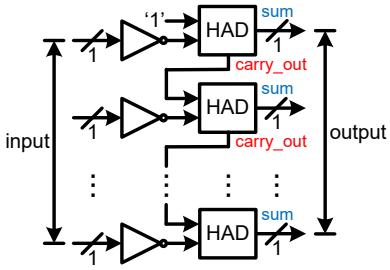

Fig. 7. The internal structure of the IC.

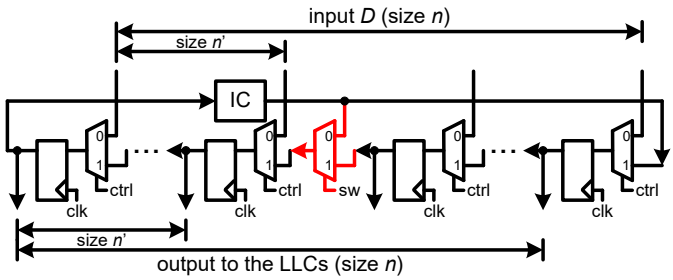

Fig. 8. Re-designed circular shift-register suitable for hybrid size operation.

Circular shift-register. As indicated in (6), the calculation of k_{n-1} involves all the bits of B multiplied with the corresponding bits from D, e.g., there are coefficients $\{d_{n-1}, d_{n-2}, \ldots, d_0\}$ involved within the calculation of k_{n-1} and then turn into $\{d_{n-2}, d_{n-3}, \ldots, d_0, -d_{n-1}\}$ in the computation of k_{n-2}, which requires one position circularly shifting plus the change of sign for one coefficient (d_{n-1}). The circular shift-register of Fig. 5 works exactly according to this signal flow pattern to deliver the required signals for the calculation of k_{n-1-i} ($0 \leq i \leq n-1$, based on (6)). As shown in Fig. 6, the initiation of values to the circular shift-register is carried out through the multiplexers (MUXes) attached to the registers such that all the input values of the operand D can be loaded into the registers through the switching of the control signal to the MUXes (ctrl) from '0' to '1'. After the registers in the circular shift-register are initiated with values of $\{d_{n-1}, d_{n-2}, \ldots, d_0\}$, respectively, as shown in Fig. 5, the values stored in each register can be circularly shifted by one position to deliver the desired output to the connected LLCs in the following cycles.

The inverter cell (IC) consists of $\log_2 q$ number of 1-bit inverters and half adders (HADs) to invert the sign of the related coefficient (based on the two's complement representation, an extra '1' is added as the carry-in), as shown in Fig. 7. All the sum bits from the HADs are used as the output of the IC.

LUT-like cell (LLC). We have used the 2-input LLC for the proposed structure, where each LLC has the same internal structure of Fig. 2 (the two input bits from B are used as the selection signals, while two output signals from the circular shift-register are used for the values attached to the MUX and the adder). There are in total $n/2$ LLCs involved (Fig. 5) to produce $n/2$ number of $\log_2 q$-bit signals to the adder tree.

Adder tree. The adder tree adds the $n/2$ input signals along with the corresponding g_i ($0 \leq i \leq n-1$) to produce the corresponding k_i according to (7) and (8). In total, there are $n/2$ number of adders involved within the adder tree. Following the decoding setup of Table I [10], an extra XOR gate can be attached to the two MSBs of k_i to produce the final output in a serial format. Note that g_i is delivered to the adder tree through a serial-in serial-out shift-register.

Pipelining setup. We have also insert the registers into the structure of Fig. 5 to facilitate the pipelined processing: (i) the signals from all the LLCs are pipelined through the registers, as indicated in Fig. 5; (ii) the adder tree is inserted with one layer of registers in the middle (or nearly) of the whole delay duration of the adder tree.

Hybrid size structure. Following the algorithmic discussion about the hybrid size operation of the finite field arithmetic for the BRLWE scheme ((9)-(11)), we can directly use the proposed structure of Fig. 5 for hybrid operation except a slight change in the circular shift-register.

As shown in Fig. 8, an extra MUX (red color) is inserted in the circular shift-register, where one input is connected with the output of the IC and another input is connected with the output of the $(n'+1)$th register from left (the output of the MUX is connected with the input to the n'th register) such that the original n-size circular shift-register turns into a n'-size circular shift-register when the selection signal of the MUX (sw) is set as '0' (the circular shift-register recovers to its original n-size when sw='1'), which functions exactly the same as the H of (11).

When used in the actual small size (n') operation (sw='0'), the rest part of the inputs (from length $n'+1$ to n, Fig. 8) will not be used, i.e., the related LLCs will only produce zero results and thus the following adder tree will deliver the accurate results according to the small size arithmetic operation of the BRLWE scheme.

V. Implementation and Comparison

Complexity. The proposed structure (Fig. 5) has a circular shift-register (n number of $\log_2 q$-bit registers and MUXes, an IC consists of $\log_2 q$ number of inverters and HADs), $n/2$ number of 2-input LLCs ($n/2$ number of 4-to-1 MUXes and $\log_2 q$-bit adders), an adder tree (n number of $\log_2 q$-bit adders), a final XOR gate, one layer of registers (n/2 number of $\log_2 q$-bit registers), and another layer of registers in the adder tree. While the proposed hybrid size design has the almost the same area-complexity as Fig. 5 except an extra MUX. Finally, both the two proposed structures have a latency of $(n+2)$ cycles.

We have then coded the proposed designs and then obtained the corresponding performance on the related FPGA device after place & route as well as the existing two designs of [10]. It is shown that [10] has higher efficiency than [9], we thus only compare our designs with the designs of [10] (while [11] uses the same hardware structure from [9] and [12] is a software based design, we do not compare with them here).

Overall experimental setup. The overall condition of the experimental setup is as follows: (i) we have coded both the proposed designs (with function verified) and the existing ones of [10] with VHDL and have used the Intel Quartus Prime

TABLE IV
COMPARISON OF THE AREA-TIME COMPLEXITIES FOR THE PROPOSED AND COMPETING DESIGNS (FPGA PLATFORM)

design	ALMs	Fmax	latency	delay	ADP*
$n = 256$					
[10]^1	3,472	201.25	65,792	326,917	1,135,055,824
[10]^2	5,734	369.14	257	696	3,990,834
Fig. 5	4,495	321.03	258	804	3,613,980
$n = 512$					
[10]^1	6,901	171.32	262,656	1,533,131	10,580,137,031
[10]^2	11,470	336.36	513	1,525	17,491,750
Fig. 5	9,038	317.06	514	1,621	14,650,598
Proposed hybrid size structure $n = 512$ and $n' = 256$					
Hybrid	8,944	319.9	514	1,607	14,373,008

Unit for Fmax: MHz. Unit for delay: ns. *: ADP=#ALM×delay.
[1]: low-complexity design (Fig. 5 of [10]). [2]: high-speed one (Fig.4 of [10]). Latency cycles do not include the shift-register's loading time.

17.0 to obtain the synthesized results based on the Stratix-V 5SGXMA9N1F45C2 device; (ii) we have selected $n = 256$, $n = 512$ and $q = 128$ for the proposed and competing designs, which follows the same parameter setting in [7], [9], [10] ($n = 512$ and $n = 256$ can provide equivalent 190/140 and 84/73 bits of class and quantum securities, respectively [8]); (iii) the adder used in the proposed and competing designs is the 8-bit ripple carry adder; (iv) we have used the MUXes in the circular shift-register to initiate the values stored in the shift-register (proposed designs); (v) the g_i is delivered to the proposed design through a serial-in serial-out shift-register (similarly for the design of [10]); (vi) we have used two layers of registers in the proposed structures; (vii) we have also coded and synthesized our hybrid size structure for $n = 512$ and $n' = 256$; (viii) The obtained area-time complexities of the proposed and the competing designs, namely the number of ALM, maximum frequency (MHz), latency cycles (not counting the loading time of the serial-in serial-out shift-register), delay (critical-path×latency cycles), and area-delay product (ADP), are all listed in Table IV.

As seen from Table IV, the proposed design has significantly better area-time complexities than the competing ones of [10]. The proposed design has 9.44% and 16.2% less ADP (21.6% and 21.2% smaller area occupation) than the existing high-speed one for the cases of $n = 256$ and $n = 512$, respectively, which is very desirable for lightweight applications. Besides that, the proposed structure has significantly smaller ADP than the existing low-complexity one of [10]. The proposed hybrid size structure has better mapping efficiency of the circuits on the FPGA device (due to the hybrid size design setup), and thus is similarly much more efficient than the existing designs.

Discussion. The proposed design (Fig. 5) is desirable for applications such as computing server or similar where demands relatively high-speed processing. While the hybrid size one can be used as a standard IP core in various environments (require different security levels) as well as for the applications where demand the upgrading of security level of the cryptosystem (from n' to n) without replacing the actual hardware device.

As the main focus of this paper is on the efficient implementation of arithmetic operation for the BRLWE scheme, we thus only compare our proposed designs with the same type of structure/scheme in the literature, for the sake of a fair comparison (we thus do not compare with the existing reports of [13-17] as they are based on Gaussian distributed errors). Besides, though the major purpose of this paper is to obtain efficient implementation for the key arithmetic of the BRLWE scheme, we still believe the techniques proposed in [9], [18] against the side-channel attacks are applicable to the structures proposed here for practical implementations & applications.

VI. CONCLUSION

This paper presents an efficient implementation of the key arithmetic operation for the BRLWE scheme based on a novel LUT-like method. We have firstly presented the proposed LUT-like method and the related arithmetic process to facilitate the using of the LUT-like technique. The proposed hardware structure is then detailed given along with a hybrid size design (the first report in the literature). The following implementation and comparison have fully demonstrated that the proposed designs have better performance over the competing ones.

ACKNOWLEDGMENT

J. Xie's work is supported by NSF CNS-2020625 and NIST 60NANB20D203. W. Jie is supported by NSF CNS-2011260.

REFERENCES

[1] W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *Symp. Founda. of Computer Science*, pp. 124-134, 1994.
[2] Post-Quantum Cryptography. https://en.wikipedia.org/wiki/Post-quantum_cryptography.
[3] D. Micciancio. Lattice-based cryptography. *Encyclopedia of Cryptography & Security*, 2011.
[4] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, 34, 2009.
[5] V. Lyubashevsky et al., "On ideal lattices and learning with errors over rings," *Int. Conf. Theory & Appl. of Crypto. Tech.*, pp. 1-23, 2010.
[6] Post-quantum cryptography round 3 submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions
[7] J. Buchmann et al., "High-performance and lightweight lattice-based public-key encryption," *ACM IoTPTS*, pp. 1-8, 2016.
[8] J. Buchmann et al., "On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack," *Int. Conf. on Cryptology in Africa*, pp. 24-43, 2016.
[9] A. Aysu et al., "Binary Ring-LWE hardware with power side-channel countermeasures," *DATE*, pp. 1253-1258, 2018.
[10] S. Ebrahimi et al., "Post-quantum cryptoprocessors optimized for edge and resource-constrained devices in IoT," *IEEE IoT-J*, vol. 6, no. 3, pp. 5500-5507, 2019.
[11] A. Sarker et al., "Fault detection architectures for inverted binary Ring-LWE construction benchmarked on FPGA," *IEEE TCAS-II*, accepted.
[12] S. Ebrahimi et al., "Lightweight and fault-resilient implementations of binary Ring-LWE for IoT devices," *IEEE IoT-J*, vol. 7, no. 8, pp. 6970-6978, 2020.
[13] T. Pöppelmann et al., "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," *ISCAS*, pp. 2796-2799, 2014.
[14] S.S. Roy et al., "Compact Ring-LWE cryptoprocessor," *CHES*, pp. 371-391, 2014.
[15] C. R.-Mejia and J. V.-Medina, "High-throughput Ring-LWE cryptoprocessors," *IEEE Trans. VLSI Syst.*, vol. 25, no. 8, pp. 2332-2345, 2017.
[16] J. Howe et al., "Lattice-based encryption over standard lattices in hardware," *DAC*, pp. 1-6, 2016.
[17] Y. Zhang et al., "An Efficient and Parallel R-LWE Cryptoprocessor," *in IEEE TCAS-II*, vol. 67, no. 5, pp. 886-890, 2020.
[18] T. Schneider et al., "Part I Towards combined hardware countermeasures against side-channel and fault-injection attacks," *Crypto*, pp. 302-332, 2016.