Novel Low-Complexity Polynomial Multiplication over Hybrid Fields for Efficient Implementation of Binary Ring-LWE Post-Quantum Cryptography

Pengzhou He, Ujjwal Guin, *Member, IEEE,* and Jiafeng Xie, *Senior Member, IEEE*

***Abstract*—Post-quantum cryptography (PQC) refers to the cryptosystem that can resist the attacks launched from mature quantum computers in the not far future and has recently gained intensive attention from the research community as most of the existing public-key cryptosystems are vulnerable to attacks from quantum computers. Ring-Learning-with-Errors (Ring-LWE)-based scheme is an essential type of the lattice-based PQC due to its strong security proof and ease of implementation. As the latest variant of the Ring-LWE, the binary Ring-LWE (BRLWE)-based scheme possesses even smaller computational complexity and thus is more suitable for resource-constrained applications. However, the existing works have not well covered various aspects related to this new scheme, especially on the low-complexity hardware implementation. In this paper, we aim to present a novel implementation of the BRLWE-based scheme on the hardware platform with very low-complexity with this point of view. To carry out the specified work in a successful manner, we have proposed mainly four layers of coherent inter-dependent efforts: (i) we have provided the necessary algorithmic derivation process in detail to formulate the desired algorithm for the polynomial multiplication over hybrid fields, which is the major arithmetic component of the BRLWE scheme; (ii) we have presented the corresponding hardware architecture in a thorough format with sufficient description of the internal structures; (iii) we have also provided the complexity analysis and implementation-based comparison to demonstrate the superior performance of the proposed polynomial multiplication over the state-of-the-art design; (iv) finally, we have extended the proposed low-complexity polynomial multiplication to the major operational phase of the BRLWE scheme. We have shown that the proposed BRLWE structure involves significantly lower area-time complexities over the existing design, e.g., the proposed design has at least 66.01% less area-delay product (ADP) than the newly reported [18] (Straix V device). Overall, the proposed design and implementation strategies are highly efficient, and the proposed BRLWE structure is desirable for many emerging applications.**

***Index Terms*—Binary Ring-Learning-with-Errors (BRLWE) scheme, field-programmable gate array (FPGA), hardware platform, low-complexity, polynomial multiplication, post-quantum cryptography (PQC).**

I. Introduction

IT has been shown that the existing widely used public key cryptosystems such as Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC) can be broken by the Shor's algorithm operated on a well-established quantum computer in the polynomial time [1], [2]. Hence, alternative solutions need to be proposed for this urgent challenge as it is predicted that the mature quantum computers will be available in the next 15-20 years (or even shorter) [3]. To address this severe concerns over security, several types of post-quantum cryptography (PQC) have been proposed recently. These schemes includes lattice-based PQC, code-based cryptography, isogeny-based cryptography, and hash-based cryptosystem, where lattice-based PQC is regarded as the most promising candidate due to its relatively implementational ease, and strong security proof [4]–[6]. In fact, the National Institute of Standards and Technology (NIST) has already started the PQC standardization process, and the recently released third round PQC candidates [7] have fully demonstrated the great potentiality of lattice-based cryptography for possible applications in the near future.

Most of the lattice-based schemes are based on the learning-with-errors (LWE) and Ring-LWE (a variant of LWE) problems [5], [6]. The standard LWE scheme involves complex matrix-vector multiplication, and hence it is impractical to deploy for practical implementations [5]. In comparison, the Ring-LWE scheme simplifies the main operation into the polynomial multiplication in the ring $\mathbb{Z}_q/(x^n+1)$, which involves less computational complexity than the standard LWE [6]. Thus, quite a number of works have been proposed on the efficient implementation of the Ring-LWE scheme [8]–[14].

Following this direction, a new variant of the Ring-LWE scheme, namely the binary Ring-LWE (BRLWE) [15], has been proposed recently specifically to target resource-constrained applications [15], [16]. The BRLWE scheme uses binary errors (instead of Gaussian distributed errors in the general Ring-LWE scheme [6]) to achieve low-complexity implementation. Note that the parameters for BRLWE are smaller than the start-of-the-art Ring-LWE settings. However, not many reports have been made to achieve efficient implementation of the BRLWE scheme, especially on the hardware platform, since the initial introduction of the BRLWE scheme. The first BRLWE scheme, which is based on software implementation, is proposed in [15] after extensive security analysis and mathematical derivation. Then, the first hardware structure is presented in [17]. Very recently, a pair of high-speed and low-complexity BRLWE structures are reported in [18]. A high-speed BRLWE structure is then proposed in [3]. Other reports also include the fault detection scheme based [19]

Manuscript received December 7, 2020, revised February 20, 2021, and accepted April 16, 2021. Corresponding author: Jiafeng Xie.

P. He and J. Xie are with the Department of Electrical and Computer Engineering, Villanova University, Villanova, PA, 19085 USA (e-mail: phe@villanova.edu; jiafeng.xie@villanova.edu).

U. Guin is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849, USA (e-mail: ujjwal.guin@auburn.edu).

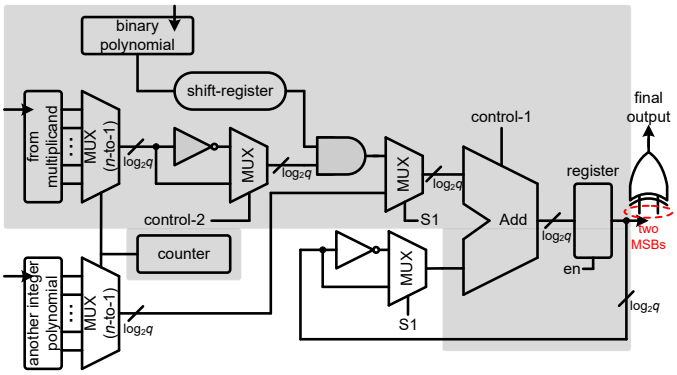


Fig. 1: The existing low-complexity BRLWE structure (Figure 5 of [18]), where the gray area represents the involved polynomial multiplication over hybrid fields.

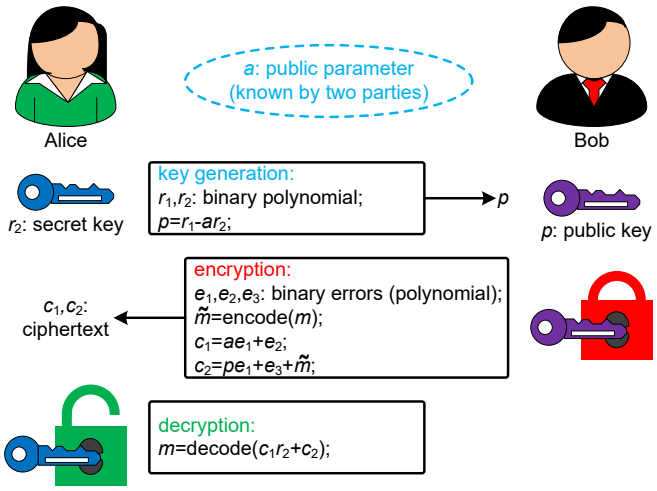


Fig. 2: Three operational phases of the BRLWE scheme.

(using the same structure of [18]) and fault resistance system (software-based [20]).

On one side, these existing designs represent the major works in the fields; on the other side, however, they still need to be improved for wide applications. Specifically for resource-constrained applications, the BRLWE scheme needs to have a low-complexity BRLWE structure. While the very recently proposed low-complexity design (see Figure 5 of [18]) needs significant improvements (see Figure 1). First, the design presented in [18] is based on the general algorithm of the key arithmetic component of the BRLWE scheme and no dedicated algorithm-to-architecture co-implementation technique has been provided. Second, the finalized structure has not designed the input resources in a comprehensive format, e.g., one of the input operands actually needs to be circularly shifted every cycle, which does require external resource usage (e.g., delivered by the software computed memory). Finally, the designed hardware architecture has not been optimized enough for potential application environments, e.g., that structure uses two n-to-1 multiplexers (MUXes of $\log_2 q$-bit, where $q = 256$ [18]) which increases the area usage and hence is not ideal for resource-constrained applications.

In this paper, we aim at proposing a novel implementation strategy for the BRLWE scheme specifically on the low-complexity aspect based on the above considerations. Noticing that the polynomial multiplication is the key arithmetic component of the BRLWE scheme, we thus propose four layers of interdependent efforts (**main contributions**) as follows:

- We have conducted the necessary mathematical derivation process to obtain the desired algorithm for the polynomial multiplication over hybrid fields, which is the primary arithmetic component of the BRLWE scheme.
- We have then presented the corresponding hardware architecture based on the proposed algorithm with a detailed explanation of the internal structures.
- We have also provided a thorough complexity analysis along with a comparison based on implementation results to validate the proposed polynomial multiplication has superior performance over the state-of-the-art solution.
- Finally, we have extended the proposed polynomial multiplication to obtain a low-complexity implementation of the BRLWE scheme and demonstrated that the proposed BRLWE structure has significantly smaller area-time complexities than the competing designs.

The rest of the paper is organized as follows. Section II introduces the preliminaries. Section III presents the proposed algorithmic derivation process. The corresponding hardware structure is described in Section IV, along with the following complexity analysis and comparison processes. The extension to the proposed BRLWE structure is then given in Section V, accompanied by comparison and discussion. Finally, we conclude the paper in Section VI.

II. Preliminaries

A. Binary Ring-Learning-with-Errors (BRLWE) Scheme

The BRLWE is a new variant of the Ring-LWE scheme, which is released recently in [15] with rigorous security analysis and proof that it still retains the hardness of the lattice problem. Though the BRLWE scheme based PQC is currently not a NIST candidate, it has excellent potential to be standardized in the future for lightweight applications. The BRLWE scheme uses binary errors (instead of Gaussian distributed errors for LWE scheme) to reduce the key size, and the corresponding area-complexity [15]. Consequently, the BRLWE involves mainly operations over the ring $\mathbb{Z}_q/f(x)$ $(f(x) = x^n + 1)$ that the polynomial of degree $(n-1)$ with integer coefficients modulo q is regarded as one typical element in the ring ($n = 512$ and $n = 256$ can provide the BRLWE scheme with equivalent 190/140 and 84/73 bits of class and quantum securities, respectively [15], [16]).

In general, a complete BRLWE scheme based PQC has three main operational phases [15], which include *key generation*, *encryption*, and *decryption*, as shown in Figure 2 and summarized as follows:

- *Key generation*: We define a as a public parameter (polynomial of integer coefficients) shared between two parties (Alice and Bob). Define that r_1 and r_2 are two binary polynomials with randomly selected coefficients and r_2 is the secret key. The main operation involved within this phase is $p = r_1 - a \cdot r_2$, where p is the public key sent to Bob. Then, r_1 is discarded after this operation.

Thus, the secret and public keys involve n and $n\log_2 q$ bits, respectively.

- *Encryption*: The n-bit message m (binary polynomial) is firstly encoded into a unique polynomial $\widetilde{m}$ according to (1). Then, Bob uses three errors (binary polynomials) e_1, e_2, and e_3 to generate the ciphertext c_1 and c_2 (a pair of polynomials and the length of the ciphertext is $2n\log_2 q$ bits), as shown in Figure 2. The ciphertext is then sent to Alice.

$$\begin{aligned} &m = m_0 + m_1 x + \cdots + m_{n-1}x^{n-1} \\ &\text{encode} : (m_0, \ldots, m_{n-1}) \rightarrow \sum_{i=0}^{n-1} m_i (\frac{q}{2}) x^i \ (\widetilde{m}). \end{aligned} \tag{1}$$

- *Decryption*: Alice uses the secret key r_2 to recover the original message m. Note that there is a threshold decoder function involved that returns a binary value of '1' if the coefficient of the calculated polynomial is in the range of $(q/4, 3q/4)$, otherwise, it will return a '0'.

Remark. For the sake of hardware implementation, the authors of [18] have proposed to use the coefficients of the polynomials in the ring from the inverted range (compared to the original one), where each coefficient of every element is represented in $(-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor - 1)$ (which exactly matches the two's complement representation range). In this case, all the necessary modular addition/subtractions of two coefficients are performed automatically without any reduction. Note that the three main phases of Figure 2 under this notation have exactly the same operations (parameters) as those of the original one. The encode function of (1) can be updated to

$$\text{encode} : (m_0, \ldots, m_{n-1}) \rightarrow \sum_{i=0}^{n-1} m_i (-\frac{q}{2}) x^i \ (\widetilde{m}), \tag{2}$$

and similar update is applied to the final decode (to be opposite of the original decode function) [18]. Note that we will also use this representation in the proposed structure.

B. Polynomial Multiplication over Hybrid Fields

The arithmetic operation involved with each phase of the BRLWE scheme, as shown in Figure 2, can be summarized as follows:

$$\begin{aligned} &\text{key generation} : \text{PM and PA} \rightarrow p, \\ &\text{encryption} : \text{PM and PA} \rightarrow c_1, \\ &\text{encryption} : \text{PM and PA and PA} \rightarrow c_2, \\ &\text{decryption} : \text{PM and PA} \rightarrow \text{output}, \end{aligned} \tag{3}$$

where, PM refers to the polynomial multiplication and *PA* denotes the polynomial addition. For example, the operation of ae_1 in the encryption phase of Figure 2 is a polynomial multiplication, similar to the other operations in other phases.

It is interesting to observe that these polynomial multiplications share one common feature: one polynomial has integer coefficients while the other has merely binary coefficients. Though polynomial multiplication based on integer coefficients or binary ones have been explored in the literature [21]–[24], this type of polynomial multiplication (let us just define as the polynomial multiplication over hybrid fields), however, has not been well covered. As the major arithmetic complexity of each phase of the BRLWE scheme lies mainly on this specific type of polynomial multiplication (polynomial additions are just point-wise operations), considerable efforts need to be made in this area (particularly on the implementation aspect).

III. Mathematical Formulation & Arithmetic Derivation

Without loss of generality and for simplicity of discussion, one can use a general form to represent all the polynomial multiplications for the three phases of the BRLWE scheme. In this paper, we focus on the low-complexity style.

A. Mathematical Formulation

Definition 1. Define a general polynomial multiplication over hybrid fields as:

$$T = DB \bmod f(x), \tag{4}$$

where $D = \sum_{i=0}^{n-1} d_i x^i$ (d_i are $\log_2 q$-bit integers in $\mathbb{Z}_q$), $B = \sum_{i=0}^{n-1} b_i x^i$ ($b_i \in \{0, 1\}$), and $f(x) = x^n + 1$. T is the multiplication product and $T = \sum_{i=0}^{n-1} t_i x^i$ and t_i are also $\log_2 q$-bit integers in $\mathbb{Z}_q$.

T of (4) can then be rewritten as

$$T = D\sum_{i=0}^{n-1} b_i x^i \bmod f(x) = \sum_{i=0}^{n-1} b_i (Dx^i \bmod f(x)). \tag{5}$$

Definition 2. Let us define again $D^{[0]} = Dx^0 \bmod f(x) = D$, $D^{[1]} = Dx^1 \bmod f(x)$, ..., $D^{[j]} = Dx^j \bmod f(x)$, ..., $D^{[n-1]} = Dx^{n-1} \bmod f(x)$.

From Definition 2, one can also find that

$$\begin{aligned} D^{[1]} =& Dx^1 \bmod f(x) = D^{[0]} x \bmod f(x) \\ &\cdots \ \cdots \ \cdots \\ D^{[j]} =& Dx^j \bmod f(x) = D^{[j-1]} x \bmod f(x), \\ &\cdots \ \cdots \ \cdots \\ D^{[n-1]} =& Dx^{n-1} \bmod f(x) = D^{[n-2]} x \bmod f(x), \end{aligned} \tag{6}$$

Thus, we can have

$$\begin{aligned} D^{[0]} =& d_0 + d_1 x + d_2 x^2 + \cdots + d_{n-1} x^{n-1}, \\ D^{[1]} =& D^{[0]} x \bmod f(x) \\ =& d_0 x + d_1 x^2 + \cdots + d_{n-1} x^n \bmod f(x) \\ =& -d_{n-1} + d_0 x + d_1 x^2 + \cdots + d_{n-2} x^{n-1}, \\ D^{[2]} =& D^{[1]} x \bmod f(x) \\ =& d_{n-1} x + d_0 x^2 + \cdots + d_{n-2} x^n \bmod f(x) \\ =& -d_{n-2} - d_{n-1} x + d_0 x^2 + \cdots + d_{n-3} x^{n-1}, \\ &\cdots \ \cdots \ \cdots \\ D^{[j]} =& D^{[j-1]} x \bmod f(x) \\ =& d_{n-1} x + d_0 x^2 + \cdots + d_{n-2} x^n \bmod f(x) \\ =& -d_{n-j} - \cdots + d_0 x^j + \cdots + d_{n-j-1} x^{n-1}, \\ &\cdots \ \cdots \ \cdots \\ D^{[n-1]} =& D^{[n-2]} x \bmod f(x) \\ =& d_2 x + d_3 x^2 + d_4 x^3 + \cdots + d_1 x^n \bmod f(x) \\ =& -d_1 - d_2 x - d_3 x^2 - d_4 x^3 + \cdots + d_0 x^{n-1}, \end{aligned} \tag{7}$$

where we have substituted $x^n \equiv -1$ (since $f(x) = x^n + 1$ and $x^n + 1 \equiv 0$ in integer field).

Thus, (5) can be transferred into

$$T = \sum_{i=0}^{n-1} D^{[i]} b_i = \sum_{j=0}^{n-1} t_j x^j, \tag{8}$$

which can be written as the form of

$$\begin{aligned} t_0 &= \sum_{i=0}^{n-1} D_0^{[i]} b_i, \\ t_1 &= \sum_{i=0}^{n-1} D_1^{[i]} b_i, \\ &\cdots \ \cdots \ \cdots \\ t_{n-1} &= \sum_{i=0}^{n-1} D_{n-1}^{[i]} b_i, \end{aligned} \tag{9}$$

where $D_j^{[i]}$ denotes the jth coefficient according to the order of x^j (for $0 \le j \le n-1$). For instance, connecting with (8), we have

$$\begin{aligned} D_0^{[0]} &= d_0, \ D_1^{[0]} = d_1, \cdots, D_{n-1}^{[0]} = d_{n-1}, \\ D_0^{[1]} &= -d_{n-1}, \ D_1^{[0]} = d_0, \cdots, D_{n-1}^{[0]} = d_{n-2}, \\ &\cdots \ \cdots \ \cdots \\ D_0^{[n-1]} &= -d_1, \ D_1^{[0]} = -d_2, \cdots, D_{n-1}^{[0]} = d_0. \end{aligned} \tag{10}$$

B. Algorithmic Derivation

TABLE I: Coefficients (without sign) of $D_j^{[i]}$ in (9)

	$D_0^{[i]}$	$D_1^{[i]}$	$D_2^{[i]}$	...	$D_{n-3}^{[i]}$	$D_{n-2}^{[i]}$	$D_{n-1}^{[i]}$
$D_j^{[0]}$	d_0	d_1	d_2	...	d_{n-3}	d_{n-2}	d_{n-1}
$D_j^{[1]}$	d_{n-1}	d_0	d_1	...	d_{n-4}	d_{n-3}	d_{n-2}
$D_j^{[2]}$	d_{n-2}	d_{n-1}	d_0	...	d_{n-5}	d_{n-4}	d_{n-3}
...	...	...	...	...	...	...	...
$D_j^{[n-3]}$	d_3	d_4	d_5	...	d_0	d_1	d_2
$D_j^{[n-2]}$	d_2	d_3	d_4	...	d_{n-1}	d_0	d_1
$D_j^{[n-1]}$	d_1	d_2	d_3	...	d_{n-2}	d_{n-1}	d_0

For low-complexity implementation of the polynomial multiplication over hybrid fields, we can actually compute the coefficients of T in a serial format. Connecting with (10) and (9), one can see that for computing each t_i ($0 \le i \le n-1$), the coefficients of b_i are the same while the corresponding $D_j^{[i]}$ varies from each other (also with the changes on the signs). For a more detailed information, we can list all the actual coefficients (without sign) of $D_j^{[i]}$ of (9) and the related signs, respectively, in Tables I and II, respectively.

It is clear that, from Table I, the jth coefficients of all the $D^{[i]}$ are actually circularly shifted (for $0 \le i, j \le n-1$), e.g., the $(n-1)$th coefficients of all the $D^{[i]}$ ($0 \le i \le n-1$) are $\{d_{n-1}, d_{n-2}, \ldots, d_2, d_1, d_0\}$ can be circularly shifted to have the $(n-2)$th coefficients of all the $D^{[i]}$ ($0 \le i \le n-1$) as $\{d_{n-2}, d_{n-3}, \ldots, d_1, d_0, d_{n-1}\}$ (similar to others).

TABLE II: Corresponding Signs for the Coefficients of $D_j^{[i]}$ in (9)

	$D_0^{[i]}$	$D_1^{[i]}$	$D_2^{[i]}$	...	$D_{n-3}^{[i]}$	$D_{n-2}^{[i]}$	$D_{n-1}^{[i]}$
$D_j^{[0]}$	+	+	+	⋯	+	+	+
$D_j^{[1]}$	−	+	+	⋯	+	+	+
$D_j^{[2]}$	−	−	+	⋯	+	+	+
...	...	...	...	...	...	...	...
$D_j^{[n-3]}$	−	−	−	⋯	+	+	+
$D_j^{[n-2]}$	−	−	−	⋯	−	+	+
$D_j^{[n-1]}$	−	−	−	⋯	−	−	+

Meanwhile, from Table II, one can observe that there is also an sign inverted between the jth coefficients of all the $D^{[i]}$ with the $(j-1)$th coefficients of all the $D^{[i]}$ (for $0 \le i, j \le n-1$). For instance, the signs of all $(n-1)$th coefficients of all the $D^{[i]}$ ($0 \le i \le n-1$) are $\{+, +, \ldots, +, +, +\}$, which becomes $\{+, +, \ldots, +, +, -\}$ for the signs of all $(n-2)$th coefficients of all the $D^{[i]}$.

Definition 3. Based on the above observations from Tables I and II, we can further define that the value (without sign) of the $D_j^{[i]}$ as $\gamma(D_j^{[i]})$ and the corresponding sign as $\varsigma(D_j^{[i]})$, e.g.,

$$\begin{aligned} \gamma(D_0^{[1]}) &= d_{n-1}, \\ \varsigma(D_0^{[1]}) &= -, \end{aligned} \tag{11}$$

which matches exactly with the same coefficient from (10).

Based on the above definitions and equations, we can thus have the proposed algorithm for low-complexity implementation of the polynomial multiplication over hybrid fields.

Algorithm 1: Proposed algorithm for the polynomial multiplication over hybrid fields.

Input : D, B, and T are polynomials over hybrid fields. D and T are the polynomials with $\log_2 q$-bit integer coefficients over ring and B is a binary polynomial.

Output: $T = DB \bmod f(x)$ ($f(x) = x^n + 1$).

Initialization step

$\overline{T} = 0$;

Main step

for $j = n-1$ *to* 0 **do**

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

    $\overline{T} = \overline{T} + \varsigma(D_j^{[i]})\gamma(D_j^{[i]})b_i$. // following (8)-(10)

  end

  $t_j = \overline{T}$;

end

Final step

Obtain the output T from serially delivered t_j ;

Note that the actual execution of the proposed Algorithm 1 also involves two unique features (contributions):

- All the values of $\gamma(D_j^{[i]})$ are processed based on the original input D and hence no external resources are required for value-shifting.

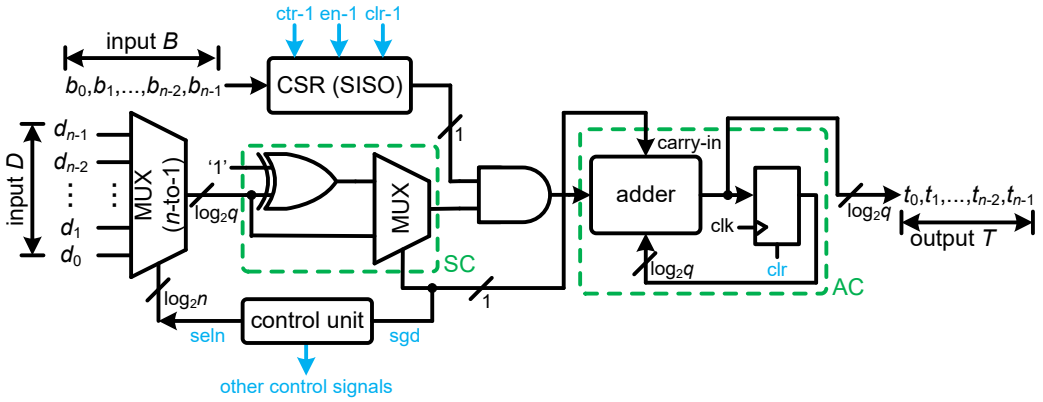


Fig. 3: The proposed structure for the polynomial multiplication over hybrid fields. CSR: circular shift-register; SISO: serial-in serial-out; SC: sign control; AC: accumulation cell.

- All the related signs of $\varsigma(D_j^{[i]})$ are controlled by the control unit to match the corresponding values of $\gamma(D_j^{[i]})$.

Besides that, the delivering of t_j is in the sequence of starting from t_{n-1}, t_{n-2}, ... to t_0, simply because all the coefficients of $D_{n-1}^{[i]}$ are positive values and no extra values are required to pre-process the input with negative signs. Similarly, the coefficients of B are processed from b_{n-1}, b_{n-2}, ... to b_0.

IV. Proposed Structure, Complexity, and Comparison

With the help of Algorithm 1, we have presented the proposed structure for polynomial multiplication over hybrid fields along with the corresponding complexity analysis and comparison with the existing one of [18] using their implementation on field-programmable gate array (FPGA) platform.

Parameter Setting. Connecting with the preliminary of Section II and the proposed algorithm in Section III, we specify the corresponding parameter setting: (i) n is the security level of the BRLWE scheme, also the size of the involved polynomial; (ii) the integer polynomial has n $\log_2 q$-bit coefficients and the binary polynomial has n binary values.

A. Proposed Structure

The proposed polynomial multiplier based Algorithm 1 is shown in Figure 3, where the structure consists of five main components, i.e., the input component, the sign control (SC) component, the AND gate cell, the accumulation cell (AC) component, and the control unit component. The details of these components are given as follows:

Input Component. The input component includes the input processing cells for polynomial D and B, as shown in Figure 3. The functions and internal structures of these cells are introduced as follows.

All the coefficients of the input D, namely $\{d_0, \ldots, d_{n-1}\}$, are connected to a n-to-1 MUX, where the corresponding $\log_2 n$-bit selection signals are generated from the control unit. The operation of the n-to-1 MUX is following the pattern shown in Table III, where the selection signals ($\log_2 n$-bit) start with "00...00" and end with "11...11" for the first n cycles such that the output of the MUX follows the sequence of d_{n-1} (last) $\rightarrow d_{n-2} \rightarrow \ldots \rightarrow d_1 \rightarrow d_0$ (first); while the selection signals start with "00...01" to "11...11" and end with "00...00" for the second n cycles (similar to the following cycles). This operation exactly executes the process of delivering $\gamma(D_j^{[i]})$ in the main steps (Lines 2-7) of Algorithm 1 (also following Table I), which does not require any external resources. Note that the details of the generating of selection signals (*seln*) for the n-to-1 MUX is described in the control unit component.

TABLE III: Operation Details of the MUX for the Input D

cycles	signal output sequence from the MUX*
the first n cycles	$d_{n-1} \rightarrow d_{n-2} \rightarrow \ldots \rightarrow d_1 \rightarrow d_0$
the second n cycles	$d_{n-2} \rightarrow d_{n-3} \rightarrow \ldots \rightarrow d_0 \rightarrow d_{n-1}$
the third n cycles	$d_{n-3} \rightarrow d_{n-4} \rightarrow \ldots \rightarrow d_{n-1} \rightarrow d_{n-2}$
...	... ...
the $(n-1)$th n cycles	$d_1 \rightarrow d_0 \rightarrow \ldots \rightarrow d_3 \rightarrow d_2$
the nth n cycles	$d_0 \rightarrow d_{n-1} \rightarrow \ldots \rightarrow d_2 \rightarrow d_1$

*: The signal on the far right is delivered out from the MUX first, while the signal on the far left is the last one to be delivered out from the MUX.

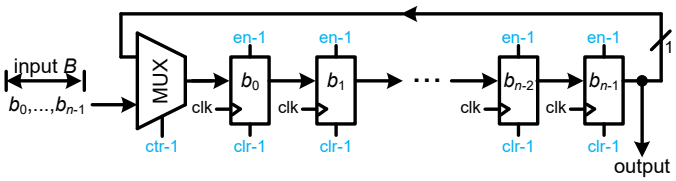


Fig. 4: The internal structure of the CSR (SISO) for B, where the values in each register are the initial loading values.

Besides that, the corresponding input B is fed to a serial-in serial-out (SISO) circular shift-register (CSR) to produce the desired output to the AND gate cell according to the sequence specified in Line 4 of Algorithm 1 (also see (9)) as: $\{b_{n-1}, b_{n-2}, \ldots, b_1, b_0\}$ (these n output values will be repeated with the same order again after n cycles). The internal structure of the CSR is shown in Figure 4. After the initial values are loaded in all the n registers (Figures 3 and 4), the MUX helps these related registers to function in a circularly-shifted format to produce the output to the AND cell.

Sign Control (SC) Component. As shown in Figure 3, the SC component consists of one XOR gate and a 2-to-1 MUX. The SC component functions according to the operation of

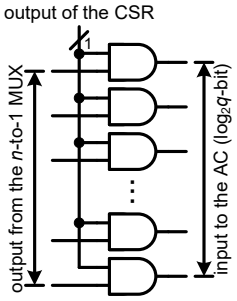


Fig. 5: The internal structure of the AND gate cell.

TABLE IV: Control Signals Generated by the Control Unit

name	description	width
seln	select D for the n-to-1 MUX during each cycle	$\log_2 n$
clr	clear the register in the AC after each n cycles	1
sgd	to obtain $\varsigma(D_j^{[i]})$ based on Definition 3	1
rr	t_j read ready signal	1
ctr-1	circularly shifting (CSR) for input B	1
en-1	enable the operation of the registers in the CSR (B)	1
clr-1	clear the content of the registers in the CSR (B)	1

$\varsigma(D_j^{[i]})$ in Line 4 of Algorithm 1 (also see Table II). One input of the XOR gate is set as '1' to help with the sign inverting for the output from the n-to-1 (Table II): (i) the selection signal (sgd, for the 2-to-1 MUX) switches to deliver the output from the XOR gate as the output (the XOR gate inverts the output of the n-to-1 MUX); (ii) the carry-in to the adder turns from '0' to '1' (meet the requirement of the two's complement representation). When the related sign is positive, the output of the n-to-1 MUX is just directly delivered to the AND cell (meanwhile the carry-in to the adder is set as '0').

AND Gate Cell. The following AND cell functions to obtain the multiplication of the $D_j^{[i]}b_i$ in (9) (also Line 4 of Algorithm 1) to be accumulated in the following AC, and its detail can be seen in Figure 5. All the output bits of the AND gates are then connected to the following AC.

Accumulation Cell (AC) Component. The AC component, as shown in Figure 3, consists of a $\log_2 q$-bit adder and a $\log_2 q$-bit register. The output of the adder is used as the input to the register, while the register's output is connected back as another input of the adder such that the desired result is produced in the output of the adder after n cycles, according to the Line 4 of Algorithm 1 (the accumulation operation is directed by the register clear signal (*clr*)). Every new output is available in every n cycles according to the sequence of t_{n-1},..., to t_0, which takes in total $n \times n$ cycles to obtain all the coefficients of T.

Control Unit Component. The control unit is in charge of all the related control signals within the proposed structure, including the selection signal for the n-to-1 MUX, clear signal (*clr*) for the AC, the sign of each $D_j^{[i]}$, the control signals for the CSR, and the ready-to-read signal for the output of the multiplier. Details of these signals are shown in Table IV.

After the releasing of the reset signal to the overall structure, the control unit works in two states in a successive order ("reset" is also set as a state, see Figure 6(a)), namely "load" and "compute". The state of "load" refers to the reading of

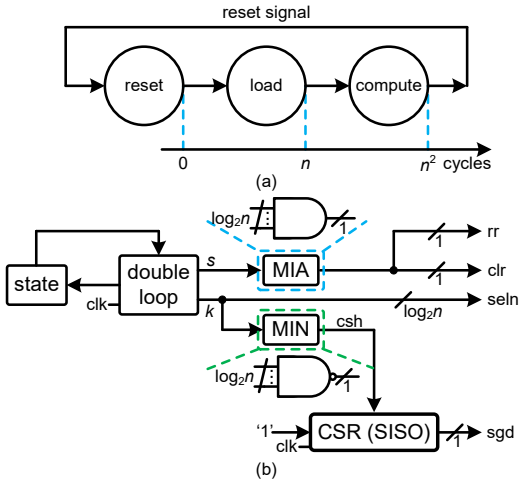


Fig. 6: (a) The main three states involved within the control unit. (b) The top-level structure of the control unit. MIA: multi-input AND gate. MIN: multi-input NAND gate.

b_j into the CSR for B (n clock cycles). Then, the control unit enters into the "compute" state (another n^2 cycles). Under the "compute" state, the control unit is responsible for organizing and controlling all the computational components in the structure of Figure 3 to get the correct output T.

To achieve this goal, the top-level structure of the control unit is shown in Figure 6(b), where the core is a specially designed double loop module. The double loop module produces two $\log_2 n$-bit signals (loop indicators) s and k, and from these two signals we can obtain all the other control signals required during the "compute" state. The synchronous clear signal *clr* for the register in the AC functions when s = "111...111" ($\log_2 n$-bit). Meanwhile, the output read-ready signal *rr* is released and activated. As shown in Figure 6, both signals (*clr* and *rr*) are produced by a multi-input AND gate (MIA), i.e., a $\log_2 n$-input AND gate. The selection signal to the n-to-1 MUX *seln* can be directly obtained from k to fulfill the corresponding requirement based on Table IV. The sign control signal to the MUX in the SC component and the carry-in in the AC component (*sgd*) is produced by controlling a SISO CSR, where the output (*csh*) of the multi-input NAND gate (MIN) is attached to the 2-to-1 MUX in the 1-bit CSR (SISO). When k = "000...000" ($\log_2 n$-bit), the CSR reads one '1' in (otherwise, it just delivers the proper output according to its circularly shifting nature). Note that the initial content of the registers in the CSR are set as zero. Besides that, *ctr*-1 (for the CSR of input B) is set as '0' in the "load" state and turns into '1' in the "compute" state, while *en*-1 is set as '1' and *clr*-1 as '0' all the time (the registers are enabled with initial values as zero).

The internal structure of the double loop module is shown in Figure 7. The loop indicator s, consists of $\log_2 n$-bits as $\{s_0, s_1, \ldots, s_{h-1}\}$, are generated by $\log_2 n$ number of loop units (paired 1-bit register and 1-bit full adder (AD) connected in a loop format. Finally, all the output bits of the 1-bit registers are combined together to form the loop indicator s. The $\log_2 n$-bits of the loop indicator s are then fed to a $\log_2 n$-input NAND gate to produce a selection signal (*sel*) for all the

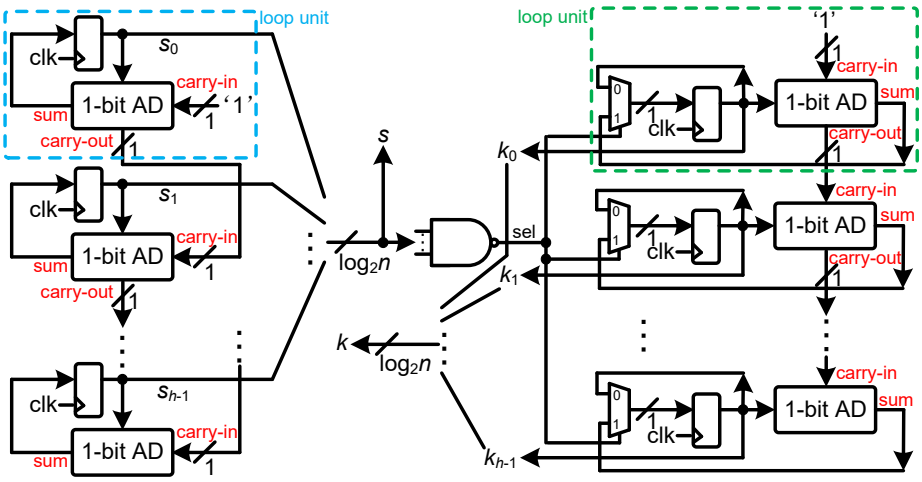


Fig. 7: The inner structure of the double loop module, where $h = \log_2 n$ and 1-bit AD refers to 1-bit full adder. The loop indicators s and k have $\log_2 n$-bits as $\{s_0, s_1, \ldots, s_{h-1}\}$ and $\{k_0, k_1, \ldots, k_{h-1}\}$, respectively.

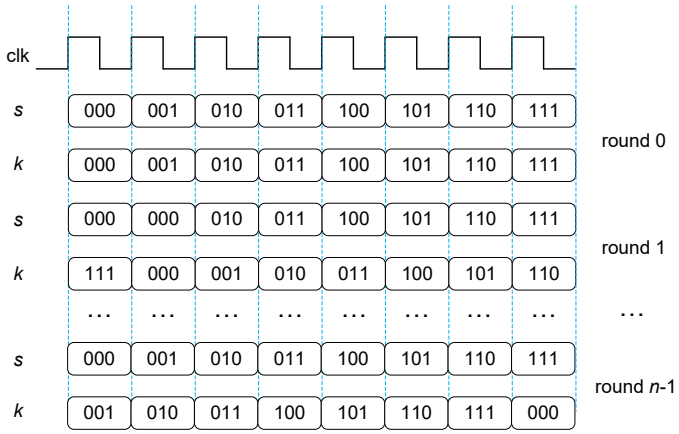


Fig. 8: The transition of both loop indicators s and k through an example of $n = 8$, where one round refers to the clock cycles required to deliver one t_j.

2-to-1 MUXes in the $\log_2 n$ number of loop units for producing another loop indicator k. Note that the first 1-bit AD's carry-in is set as '1' and its carry-out is used for the carry-in of the neighboring AD (similar to other loop units). Eventually, all the output bits of the 1-bit registers are used to form the loop indicator k.

Example. To illustrate the detailed operation of the double loop module, we can have the following explanation (with a case study example). As shown in Table II, there are $(j-1)$ negative signs in j-th row. Thus, we can add one more negative sign into the CSR in each round (one round refers to n cycles to produce one t_j), at a specific cycle determined by k. To get these specific cycles, we can define the n rounds' clock sequence for the "compute" state as $\{[1, 2, ..., n], [n+1, ..., 2n], ..., [n(n-1)+1, ..., n^2]\}$. Then, the clocks of which (the signs for D_j^i is negative) should be $\{[n+1], [2n+1, 2n+2], [3n+1, 3n+2, 3n+3], ..., [(n-1)n+1, (n-1)n+2, ..., (n-1)n+n-1]\}$. Note that through using a CSR of length n, the output of the CSR can be delayed for n cycles. In this case, the clock sequence becomes $\{[1], [n+1, n+2], [2n+1, 2n+2, 2n+3], ..., [(n-2)+1, (n-2)n+2, ..., (n-2)n+n-1]\}$. It is easy for us to get such a sequence from the output of the CSR by feeding '1' (negative sign) at cycles of $\{1, n+2, 2n+3, ..., (n-2)n+(n-1)\}$, which is exactly when the inner loop indicator $k = 0$. The rest of the negative signs in a round are inherited or circularly shifted back from the previous round. We have also shown an example of the actual value changes of the indicators s and k under the case of $n = 8$ (Figure 8), where one can notice that k is delayed for $(j-1)$ cycles compared to s for the jth round operation. Overall, the proposed control unit functions in a very effective way and also maintains very low area consumption.

B. Complexity Analysis

Overall, the area-time complexities of the proposed structure for the polynomial multiplication over hybrid fields (Figure 3) are listed as follows:

The input component of the proposed polynomial multiplication architecture has one $\log_2 q$-bit n-to-1 MUX (for input D) and one CSR for input B, where the CSR contains one 1-bit 2-to-1 MUX and n number of 1-bit registers.

The SC component has a $\log_2 q$-bit XOR gate and a $\log_2 q$-bit 2-to-1 MUX. While the following AND gate cell has $\log_2 q$ number of 1-bit AND gates.

The following AC component has one $\log_2 q$-bit adder and a $\log_2 q$-bit register. Finally, a control unit is also needed to direct the proper signal flow of the overall structure.

The proposed structure of Figure 3 delivers out output t_j (j from $n-1$ to 0) in every n cycles after all the necessary values are loaded in the proper status (e.g., all the initial values of B are stored in the CSR) and it takes in total $n \times n$ cycles of calculation to obtain the complete output of T.

Theoretical Comparison. Compared with the existing design of [18], as shown by the gray area in Figure 1, the proposed structure for the polynomial multiplication over hybrid fields (BRLWE scheme) undoubtedly has several unique advantages:

- The proposed structure does not require any external resources to shift the coefficients of input D in a circular format, which is required in the existing design (circularly shifted in every cycle). In fact, the input D is directly fed

to the proposed structure and no specific operation on the input operand is further needed. This undoubtedly saves potential resource usage, especially when considering the resource-constrained application environments.

- The proposed structure involves smaller critical-path than the existing one since the existing design of [18] requires one n-to-1 MUX, two 2-to-1 MUXes, and one adder (the AND cell and inverter involve very small delay) as the main critical-path of the design, while the proposed design in Figure 3 needs only one n-to-1 MUX, one 2-to-1 MUX, and one adder as the major critical-path delay. Hence, it is expected that the proposed design has higher maximum frequency than the existing one.
- The binary input B is fed to a SISO CSR (in the proposed design), which can produce the correct output to the main computation components of the structure in a repeated format (every n clock cycles). While in the existing design of [18], only a regular shift-register is used without detailed explanation of the internal structure, which potentially increases the difficulty of control signal generation as well as the corresponding area-complexity.
- The proposed structure of Figure 3 uses a novel control unit to coordinate the sign inversion (with the help of a SISO CSR, see Figure 6) for the required processing values as well as the related operations in the other components (with very small critical-path). While the existing design uses a counter to control the input signal processing, which potentially has larger time-complexity.

C. Comparison (FPGA Implementation Based)

For a detailed comparison, we have also coded and obtained the related implementational performance of the proposed structure on the FPGA platform along with the existing one of [18] (gray area of Figure 1).

For a fair comparison, the overall **experimental setups** are as follows:

(i) We have used VHDL to code the proposed design of Figure 3 and have verified its function through ModelSim. Meanwhile, we have also re-coded and re-implemented the existing lightweight structure of [18] (following the gray area of Figure 1). We have then obtained the results on the two designs through the Intel Quartus Prime 17.0 on the Stratix V (5SGXMA9N1F45C2), Arria V GZ (5AGZME7K3F40I4), and Cyclone V (5CSXFC6D6F31I7ES) devices (after implementation), respectively.

(ii) We have followed the parameter selections in the previous reports of [15], [17], [18], i.e., $n = 256$ (and $n = 512$) and $q = 256$. We have also used the same type of adder in the coding, namely the ripple carry adder.

(iii) As the internal structure of the shift-register in the existing design is unclear, we just use the proposed SISO CSR for the existing design, which facilitates the control signal setup in the existing design and potential complexity reduction.

(iv) We do not add any external resources in the existing design for the shifting of input D, but just assume that operation is provided by the external system already.

(v) The final area-time complexities of the proposed and the existing designs, in terms of the number of adaptive logic module (ALM), maximum frequency (short for Fmax, unit is MHz), latency cycles (computation cycles for delivering the output signals), delay (critical-path×latency cycles, where critical-path=1/Fmax), and area-delay product (ADP), are listed and calculated in Table V.

TABLE V: Comparison of the Area-Time Complexities for the Proposed and Competing designs on the FPGA Platform

design	ALMs	Fmax	latency	delay	ADP[1]	ER
$n = 256$ (Straix V)						
Fig. 1* [18]	1,776	201.05	65,536	326	578,976	Y
This work	1,793	318.47	65,536	206	369,358	N
$n = 512$ (Straix V)						
Fig. 1* [18]	3,478	186.39	262,144	1,406	4,890,068	Y
This work	3,491	288.77	262,144	908	3,169,828	N
$n = 256$ (Arria V)						
Fig. 1* [18]	1,777	176.24	65,536	372	661,044	Y
This work	1,793	280.03	65,536	234	419,562	N
$n = 512$ (Arria V)						
Fig. 1* [18]	3,479	172.56	262,144	1,519	5,284,601	Y
This work	3,490	249.0	262,144	1,053	3,674,970	N
$n = 256$ (Cyclone V)						
Fig. 1* [18]	1,808	81	65,536	809	1,462,672	Y
This work	1,828	143.08	65,536	458	837,224	N
$n = 512$ (Cyclone V)						
Fig. 1* [18]	3,550	76	262,144	3,449	12,243,950	Y
This work	3,564	134.57	262,144	1,948	6,942,672	N

ER: external resources? Y: Yes; N: No;
Unit for Fmax: MHz. Unit for delay: μs. [1]: ADP=#ALM×delay.
*: The gray area of the existing design, see Figure 1.

It is clear that the proposed design significantly outperforms the existing structure. As seen from Table V, the proposed structure not only has higher maximum frequency (benefited from the proposed structural design & control unit), but also has smaller ADP than the existing design. For instance, the proposed design has 36.2% and 35.2% less ADP than the existing one, for the cases of $n = 256$ and $n = 512$, respectively (on the Straix V device, similar on other devices). Besides that, the existing structure also requires external resources to deliver in the shifted input multiplicand every cycle, while the proposed design does not need this type of external assistance.

V. Extension to the Proposed BRLWE Structure

In this section, we will extend the proposed polynomial multiplication to the BRLWE structure, starting from the algorithmic process to the architectural extension and finally the implementation based comparison to confirm the superior performance of the proposed BRLWE structure over the state-of-the-art solution of [18].

A. Algorithmic Process

As seen from Figure 2 and equation (3), the typical BRLWE arithmetic operation also involves another two polynomials (one integer polynomial and one binary polynomial), especially at the stage of producing ciphertext c_2 (while the other phases require just the addition with one integer polynomial). Hence, to cover all the basic operations of the BRLWE scheme, it is expected to involve these two polynomials also.

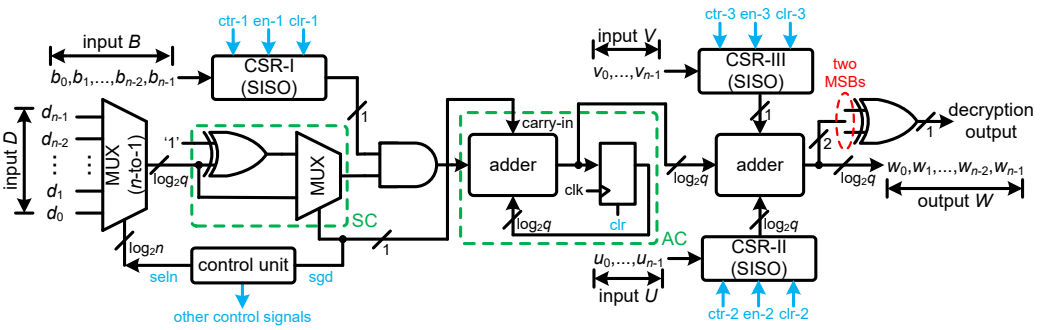


Fig. 9: The proposed structure for the BRLWE scheme.

Definition 4. Define polynomials as: $W = \sum_{i=0}^{n-1} w_i x^i$, $U = \sum_{i=0}^{n-1} u_i x^i$, and $V = \sum_{i=0}^{n-1} v_i x^i$ (w_i and u_i are $\log_2 q$-bit integers in $\mathbb{Z}_q$, and v_i is binary value). We define also

$$\begin{aligned} W =& DB \bmod f(x) + U + V, \\ =& T + U + V, \end{aligned} \tag{12}$$

where $f(x) = x^n + 1$ and the addition of $T+U+V$ is relatively easy since it is simply point-wise operation. The equation of (12) can thus be used as a universal form for the operations involved within each phase of Figure 2 (all the coefficients of V can be set as '0' if there is no addition with a binary polynomial involved for that specific phase).

Correspondingly, Algorithm 1 in Section III is updated, i.e., adding one step after Step 2.5 as $w_j = t_j + u_j + v_j$, which is serially delivered to obtain the final output W. Note that for the decryption phase, the final step actually requires an 1-bit XOR gate (following [17], [18]) connecting with the two most-significant bits (MSBs) of the w_j to obtain the desired decoded message m, according to the description in Figure 2.

B. Corresponding Structure

Following the algorithmic operation of Algorithm 2 and the design style of Figure 3, we can thn have the proposed BRLWE structure as shown in Figure 9, where it is updated with an extra adder and two CSRs (SISO style for inputs U and V, respectively) when compared with the design in Figure 3. The newly added adder is the same as that in the AC component, while the CSR-II and CSR-III have the same internal structure as that in Figure 4 except that the bit-width of the MUX and registers in CSR-II have been increased to $\log_2 q$-bit, as determined by the input polynomial U. The control signals to the CSR-II and CSR-III are still generated by the control unit, following the previous design strategy in Figure 3. Moreover, an extra XOR gate is used for the decryption phase operation following the setup in the existing design of [17]. The rest parts of the structure of Figure 9 are the same as those in Figure 3 (the area-complexity of Figure 9 is almost the same as that of Figure 3 except an XOR gate, an adder, and two CSRs as well as slight update on the control unit). Again, the final output w_j (j from $n-1$ to 0) is available in every n cycles and the total computation latency is $n \times n$ cycles for the obtaining of a complete output W (actually, it is $n \times n$ cycles for the decryption phase but $2n \times n$ cycles for the encryption phase).

C. FPGA based Implementation and Comparison

We have again coded the proposed design (Figure 9) and the existing lightweight structure (Figure 5 in [18]) with VHDL and have obtained their corresponding performance through the Intel Quartus Prime 17.0 on the Stratix V (5SGXMA9N1F45C2), Arria V GZ (5AGZME7K3F40I4), and Cyclone V (5CSXFC6D6F31I7ES) devices after implementation. We have also followed the same experimental setup in Section IV-C to carry out the FPGA based implementation. Note that we have again selected the parameters according to the setup in the existing design of [17], [18], where n is chosen as 256 and 512 ($n = 512$ and $n = 256$ can provide the BRLWE scheme with equivalent 190/140 and 84/73 bits of class and quantum securities, respectively [15]) and $q = 256$. Besides that, the report of [18] mainly shows the structure for the decryption phase of the BRLWE scheme, we hence just re-coded the design (Figure 5 in [18]) for the decryption phase for the existing design. While the proposed structure of Figure 9 fits both the encryption and decryption phases. The corresponding area-time complexities of the proposed and existing designs are then shown in Table VI. Note that the low-complexity design of [18] is so far the most recent BRLWE structure available in the literature (the authors of [18] shown their high-speed design outperforms the one in [17]), while the other designs in [3], [17] are either high-speed architectures (suitable for resource abundant applications) or other styles (such as [19], [20]). We thus compare our BRLWE structure only with the same processing style BRLWE design in the literature (Figure 5 of [18]).

Again, it is clear that the proposed BRLWE structure has significantly better performance than the existing BRLWE design (on the decryption phase), i.e., the proposed design has not only has smaller area-complexity than the existing one, but also involves higher maximum frequency. As a result, e.g., the proposed structure has 66.01% and 70.33% less ADP than that of the competing design for the cases of $n = 256$ and $n = 512$, respectively (on the Stratix V device, similar situation happens on other two devices also). In fact, even when operating in the encryption phase, the proposed structure still involves significantly smaller ADP over the existing structure (running in the decryption phase, which has only half of the latency cycles compared with the encryption phase).

The superior performance of the proposed structure is due to the following facts: (i) the proposed structure has employed one n-to-1 $\log_2 q$-bit MUX while the existing structure uses

TABLE VI: Comparison of the Area-Time Complexities for the Proposed and Competing BRLWE Structures.

design	device	ALMs	Fmax	latency	delay	ADP*	ADP reduction	external resource?
$n = 256$ (decryption phase)								
[18][1]	Straix V	3,472	201.25	65,792	327	1,135,344	-	Y
	Arria V	3,470	178.67	65,792	368	1,276,960	-	Y
	Cyclone V	3,556	89.67	65,792	734	2,610,104	-	Y
This work	Straix V	1,864	316.96	65,536	207	385,848	66.01%	N
	Arria V	1,864	268.17	65,536	244	454,816	64.38%	N
	Cyclone V	1,878	142.15	65,536	461	865,758	66.83%	N
$n = 512$ (decryption phase)								
[18][1]	Straix V	6,901	171.32	262,656	1,533	10,579,233	-	Y
	Arria V	6,900	155.76	262,656	1,686	11,633,400	-	Y
	Cyclone V	7,066	75.52	262,656	3,478	24,575,548	-	Y
This work	Straix V	3,551	296.65	262,144	884	3,139,084	70.33%	N
	Arria V	3,554	243.49	262,144	1,077	3,827,658	67.10%	N
	Cyclone V	3,614	129.22	262,144	2,029	7,332,806	70.16%	N
$n = 256$ (encryption phase)								
This work	Straix V	1,864	316.96	131,072	414	771,696	-	N
	Arria V	1,864	268.17	131,072	489	911,496	-	N
	Cyclone V	1,878	142.15	131,072	922	1,731,516	-	N
$n = 512$ (encryption phase)								
This work	Straix V	3,551	296.65	524,288	1767	6,274,617	-	N
	Arria V	3,554	243.49	524,288	2,153	7,651,762	-	N
	Cyclone V	3,614	129.22	524,288	4,057	14,661,998	-	N

The existing design does not provide the details for encryption phase and hence we do not put the implementation results here.
$^{\#}$ reduction of the ADP of the proposed work compared with the existing design on the same FPGA device with the same parameter setting.
Y: Yes; N: No; Unit for Fmax: MHz. Unit for delay: μs. *: ADP=#ALM$\times$delay. ^{1}: Refers to the structure of Figure 5 in [18].

two, which significantly increases the area usage; (ii) the proposed BRLWE scheme uses the output of the adder (in the AC) to be connected with the following components to maintain the latency cycles as Figure 3, while the existing structure has to use extra n cycles to calculate the addition with another integer polynomial; (iii) the proposed design uses a novel control unit which largely improves the timing efficiency while the existing structure has no such component for timing optimization; (iv) lastly, the proposed BRLWE design does not require external resource usage for the signal processing while the existing BRLWE structure requires the delivering of shifted input multiplicand in every cycle.

Overall, the proposed BRLWE cryptoprocessor has the features of: (i) low-complexity; (ii) efficient timing performance (indicated by the high maximum frequency); and (iii) independent operation (as it does not need external resource assistance on data computation), and thus is more desirable for resource-constrained application environments. Note that the existing design of [18] actually has not provided the details for extension to the encryption phase. Hence, the proposed structure is designed in a more complete and comprehensive way (covers both encryption and decryption phases).

Discussion. For a fair comparison, we have only compared the proposed BRLWE design with the same type of the design available in the literature (lightweight PQC suitable for resource-constrained applications). However, we want to mention that the authors of [17], [18] have shown sufficient comparison that the BRLWE schemes have better ADP over these regular LWE/Ring-LWE of [8], [9], [25]. Further efforts can be made to extend the proposed BRLWE cryptographic hardware in the actual application environment to be compared with various PQC. Besides that, the employing of side-channel attack resistance on the proposed BRLWE scheme, as covered in [17], [26], is also applicable to the proposed design, which can be seen as one of our future research directions.

Moreover, one has to note that the focus of our design is to obtain ultra low-complexity implementation of BRLWE scheme operating at point-to-point (coefficient-to-coefficient) level, we thus do not employ fast algorithms such as Karatsuba or Toom-Cook methods in our work (these algorithms typically works better in the parallel processing/high-performance based applications since the strategy of this type of fast algorithm is to decompose one original polynomial multiplication into smaller-size polynomial multiplications and hence has larger area usage than the low-complexity one [27]–[29] though involves better time-complexity). But in our following research work like the high-performance designs, we will endeavor to develop novel fast algorithms for the BRLWE scheme.

VI. Conclusion

This paper presents an efficient implementation of the polynomial multiplication as well as its application for the BRLWE cryptographic hardware. We have firstly presented a detailed mathematical derivation process to obtain the proposed algorithm for the polynomial multiplication of the BRLWE. Then, an efficient hardware structure is detailed presented along with following complexity analysis and FPGA based comparison to confirm its efficiency. The proposed polynomial multiplication is finally extended to obtain a low-complexity BRLWE cryptographic structure, and the implementation and comparison have fully confirmed the superior performance of the proposed design over the existing structure. The proposed design is highly efficient and can be extended further for actual cryptosystem implementation.

Acknowledgment

This work was supported in part by the National Science Foundation under grant numbers CNS-2020625, and CNS-1755733 as well as NIST-60NANB20D203. Any opinions,

findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF and NIST.

REFERENCES

[1] Post-Quantum Cryptography. https://en.wikipedia.org/wiki/Post-quantum_cryptography.

[2] W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *Symp. Founda. of Computer Science*, pp. 124-134, 1994.

[3] J. Xie, K. Basu, Kris, M. Gaj, and U. Guin, "Special Session: The recent advance of hardware implementation of post-quantum cryptography," *IEEE VLSI Testing Symposium (VTS)*, pp. 1-10, 2020.

[4] D. Micciancio. Lattice-based cryptography. *Encyclopedia of Cryptography & Security*, 2011.

[5] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, 34, 2009.

[6] V. Lyubashevsky et al., "On ideal lattices and learning with errors over rings," *Int. Conf. Theory & Appl. of Crypto. Tech.*, pp. 1-23, 2010.

[7] Post-quantum cryptography round 3 submissions. https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

[8] T. Pöppelmann and T. Güneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," *2014 IEEE Int. Symp. Circuits Syst. (ISCAS)*, Melbourne VIC, 2014, pp. 2796-2799.

[9] S.S. Roy, F. Vercauteren, N. Mentens, D.D. Chen, and I. Verbauwhede, "Compact Ring-LWE cryptoprocessor," *International Worshop on Cryptographic Hardware and Embedded Systems*, pp. 371-391, 2014.

[10] C. P. R.-Mejia and J. V.-Medina, "High-throughput Ring-LWE cryptoprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2332-2345, Aug. 2017.

[11] W. Liu et al., "Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 10, pp. 2459-2463, Oct. 2019.

[12] A. Khalid et al., "Compact, scalable, and efficient discrete Gaussian samplers for latticebased cryptography," *in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 15, 2018.

[13] Y. Zhang, C. Wang, D. E. S. Kundi, A. Khalid, M. O'Neill and W. Liu, "An Efficient and Parallel R-LWE Cryptoprocessor," *in IEEE Trans. Circuits and Systems II*, vol. 67, no. 5, pp. 886-890, May 2020.

[14] D. E. S. Kundi et al., "AxMM: Area and Power Efficient Approximate Modular Multiplier for R-LWE Cryptosystem," *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 1-5, 2020.

[15] J. Buchmann et al., "High-performance and lightweight lattice-based public-key encryption," *ACM IoTPTS*, pp. 1-8, 2016.

[16] 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.

[17] A. Aysu, M. Orshansky, and M. Tiwari, "Binary Ring-LWE hardware with power side-channel countermeasures," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1253-1258, 2018.

[18] S. Ebrahimi, S. Sarmadi, and H. Boorani, "Post-quantum cryptoprocessors optimized for edge and resource-constrained devices in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5500-5507, 2019.

[19] A. Sarker, M. M. Kermani and R. Azarderakhsh, "Fault detection architectures for inverted binary Ring-LWE construction benchmarked on FPGA," *in IEEE Trans. Circuits and Systems II*, early access.

[20] S. Ebrahimi and S. Bayat-Sarmadi, "Lightweight and fault-resilient implementations of binary Ring-LWE for IoT devices," *in IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6970-6978, Aug. 2020.

[21] D.D. Chen, N. Mentens, F. Vercauteren, S.S. Roy, R.C. Cheung, D. Pao, and I. Verbauwhede, "High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 157-166, 2014.

[22] J. Xie et al., "Novel bit-parallel and digit-serial systolic finite field multipliers over $GF(2^m)$ based on reordered normal basis," *IEEE Trans. VLSI Systems*, vol. 27, no. 9, pp. 2119-2130, 2019.

[23] P. Meher and X. Lou, "Low-latency, low-area, and scalable systolic-like modular multipliers for $GF(2^m)$ based on irreducible all-one polynomials," *IEEE TCAS-I* vol. 64, no. 2, pp. 399-408, 2017.

[24] J. L. Imaña, "LFSR-based bit-serial $GF(2^m)$ multipliers using irreducible trinomials" *IEEE Trans. Computers*, 2020 (early access).

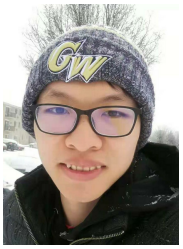
[25] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice based encryption over standard lattices in hardware," *Proceedings of the 53rd Design Automation Conference (DAC)*, 162, 2016.

[26] T. Schneider et al., "Part I Towards combined hardware countermeasures against side-channel and fault-injection attacks," *Proc. Annu. Cryptol. Conf.*, pp. 302-332, 2016.

[27] J. Xie et al., "Low register-complexity systolic digit-serial multiplier over $GF(2^m)$ based on trinomials," *IEEE Trans. Multiscale Computing Systems*, vol. 4, no. 4, pp. 773-783, 2018.

[28] J. Xie et al., "Efficient FPGA implementation of low-complexity systolic Karatsuba multiplier over $GF(2^m)$ based on NIST polynomials," *IEEE Trans. Circuits & Systems-I*, vol. 64, no. 7, pp, 1815-1825, 2017.

[29] G. Zhou et al., "Complexity analysis and efficient implementations of bit parallel finite field multipliers based on Karatsuba-Ofman algorithm on FPGAs," *IEEE TVLSI Systems*, vol. 18, no. 7, pp. 1057-1066, 2010.



Pengzhou He received his BS degree in Intelligence Science and Technology from University of Science and Technology Beijing in 2019. He is currently working towards a Ph.D. degree in Computer Engineering at Villanova University. His research interests include post-quantum cryptography, hardware security, high-performance IoT devices, and application of machine learning in security systems.

Ujjwal Guin (S'10–M'16) received his PhD degree from the Electrical and Computer Engineering Department, University of Connecticut, in 2016. He is currently an Assistant Professor in the Electrical and Computer Engineering Department of Auburn University, Auburn, AL, USA. He received his B.E. degree from the Department of Electronics and Telecommunication Engineering, Bengal Engineering and Science University, Howrah, India, in 2004 and his M.S. degree from the Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA, USA, in 2010. Dr. Guin has developed several on-chip structures and techniques to improve the security, trustworthiness, and reliability of integrated circuits. His current research interests include Hardware Security & Trust, Blockchain, and VLSI Design & Test. He has authored several journal articles and refereed conference papers. He serves on the organizing committees of HOST, VTS, and PAINE. He has been serving on the technical program committees in several reputed conferences, such as DAC, HOST, VTS, PAINE, VLSID, GLSVLSI, ISVLSI, and Blockchain. He is an active participant in the SAE International G-19A Test Laboratory Standards Development Committee and G-32 Cyber-Physical Systems Security Committee. He is a member of both the IEEE and ACM.

Jiafeng (Harvest) Xie (SM'20) received the M.E. and Ph.D. from Central South University and University of Pittsburgh, in 2010 and 2014, respectively.

He is currently an Assistant Professor in the Department of Electrical & Computer Engineering, Villanova University, Villanova, PA. His research interests include cryptographic engineering, hardware security, post-quantum cryptography, and VLSI implementation of neural network systems.

Dr. Xie has served as technical committee member for many reputed conferences such as HOST, ICCD, and ISVLSI. He is also currently serving as Associate Editor for Microelectronics Journal and IEEE Access. He was serving as Associate Editor for IEEE Transactions on Circuits and Systems-II: Express Briefs. He received the IEEE Access Outstanding Associate Editor for the year of 2019. He also received the Best Paper Award from HOST'19.