2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) | 979-8-3503-1205-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/FCCM57271.2023.00021

Efficient Implementation of Ring-Binary-LWE-based Lightweight PQC Accelerator on the FPGA Platform

Pengzhou He[1], Tianyou Bao[1], Yazheng Tu, and Jiafeng Xie[2]
Department of Electrical and Computer Engineering, Villanova University, Villanova PA 19087
Email: {phe,tbao,ytu1,jiafeng.xie}@villanova.edu; [1]: Equal contribution. [2]: Corresponding author.

***Abstract*—Post-quantum cryptography (PQC) has gained substantial attention from various communities recently. Along with the ongoing National Institute of Standards and Technology (NIST) PQC standardization process that targets the general-purpose PQC algorithms, the research community is also looking for efficient lightweight PQC schemes. Among this direction of efforts, Ring-Binary-Learning-with-Errors (RBLWE)-based encryption scheme (RBLWE-ENC) is regarded as a promising lightweight PQC fitting Internet-of-Things (IoT) and edge computing applications. As hardware implementation for PQC algorithms has become one of the major advances in the field, in this paper, we follow this trend to present an efficient implementation of RBLWE-ENC lightweight accelerator on the field-programmable gate array (FPGA) platform. Overall, we have demonstrated three coherent interdependent stages of efforts: (i) we have presented detailed derivation processes to formulate the proposed algorithmic operation; (ii) we have then implemented the proposed algorithm into a desired hardware accelerator; and (iii) we provided thorough complexity analysis and comparison to showcase the superior performance of the proposed accelerator over the state-of-the-art designs, e.g., the proposed accelerator with $v = 8$ has at least 66.67% less area-time complexities than the existing ones (Virtex-7 FPGA). We hope the outcome of this work can facilitate lightweight PQC development.**

***Index Terms*—Efficient implementation, FPGA, lightweight PQC accelerator, Ring-Binary-Learning-with-Errors.**

I. Introduction

It is proven that the existing cryptosystems such as Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC) can be solved by a well-established quantum computer [1]–[3]. To address this challenge, the National Institute of Standards and Technology (NIST) has already started the post-quantum cryptography (PQC) standardization process for general-purpose usage [4]. Besides, the research community is also looking for efficient lightweight PQC fitting specific applications such as Internet-of-Things (IoT) and edge devices [5]–[8], which was confirmed by the recent National Science Foundation (NSF) Secure and Trustworthy Cyberspace Principal Investigators' Meeting 2022 (SaTC PI Meeting'22) [9].

Many of the lattice-based PQC are based on the learning-with-errors (LWE) problem [10]. Binary-LWE (BLWE), a binary variant of LWE, was introduced later with smaller computational complexity than LWE [11]–[20]. Nevertheless, the BLWE-based scheme has restrictions on the number of samples and hence is not that ideal for lightweight applications [7]. Fortunately, a structured variant of BLWE was introduced later, i.e., the Ring-Binary-LWE (RBLWE) [5], which is based on the arithmetic operation over ring to obtain small key sizes and low resource usage. Security analyses have been carried out that the RBLWE-based encryption scheme (RBLWE-ENC) is secure and promising for lightweight applications [5], [21].

Existing Works. Since the introduction of RBLWE-ENC [5], a number of works have been released on the efficient implementation of this scheme on different platforms. The software-implemented RBLWE-ENC was reported in [5] along with rigorous security analysis. Then, an efficient RBLWE-ENC hardware structure was released in [7]. Two structures for RBLWE-ENC were proposed in [8]. A high-speed structure was reported in [12], but the design was not complete. Compact architectures were proposed in [22], [23], respectively. A lookup-table (LUT) method-based hardware design was released in [24]. A new high-speed structure for RBLWE-ENC was presented in [25]. Recently, several RBLWE-ENC hardware accelerators were also reported in [26]–[30], respectively.

Existing Challenges. Despite the amounts of existing designs, these structures suffer from three main limitations. First, the existing designs are mostly based on the schoolbook polynomial multiplication algorithm in a traditional format, e.g., see [26], [27]. Second, the existing hardware structures mostly cover only one or two processing types (i.e., offering limited speed/throughput choices), and hence their application potentialities are restricted. For example, the compact accelerator of [26] has a latency of n^2 cycles (decryption), which is a little bit slow for practical usage. Third, most of these structures' setups do not consider the practical application environment thoroughly. For instance, the output/input of the structure is assumed as 1-bit/8-bit per cycle [8], [22], [25], while these data typically are stored to (come from) memories of the modern processors, which can be 64/32-bit per cycle.

Meanwhile, we also consider that the recent trend in the PQC field has switched to more on the efficient hardware implementations. While for RBLWE-ENC, we also consider resource-constrained applications like IoT and edge devices. For this type of application, the field-programmable gate array (FPGA)-based implementation is preferred as not only the modern FPGA devices offer a variety of application environments but also the finalized architecture can be easily extended for further specific integrated circuits design.

Major Contributions. Based on the above discussions, it is clear that an efficient hardware-implemented accelerator for

2576-2621/23/$31.00 ©2023 IEEE
DOI 10.1109/FCCM57271.2023.00021

TABLE I
NOTATIONS

a	public parameter (integer polynomial)
r_1, r_2	binary polynomials (r_2: secret key)
e_1, e_2, e_3	binary errors (binary polynomials)
m	message
$f(x)$	ring polynomial ($f(x) = x^n + 1$)
B, H	binary polynomials (algorithmic derivation)
T, G, D, W	polynomials with coefficients of $\log_2 q$-bit (derivation)
u, v	integers according to $n = uv$

RBLWE-ENC with a practical input/output setup is needed. Following this direction, the contributions of the paper are:

- We have derived the arithmetic operation of RBLWE-ENC to obtain the proposed **group decomposition initiated processing** algorithm with efficient computation.
- We have designed a novel hardware accelerator based on optimized algorithm-to-architecture mapping techniques with practical input and output processing setup.
- We have implemented the proposed accelerator and have compared it with the competing ones to demonstrate the superior performance of the proposed designs.

The rest of the paper is organized as follows. Section II gives the preliminaries. The proposed algorithmic operation is presented in Section III. The proposed accelerator is introduced in Section IV. Complexity and comparison are provided in Section V. Finally, the conclusion is given in Section VI.

II. PRELIMINARIES

Notations. We use three major notations: (i) n is the security level of RBLWE-ENC (polynomial size); (ii) RBLWE-ENC relies on the operations over ring $\mathbb{Z}_q/(x^n + 1)$; (iii) q is the modulus. The others are listed in Table I (see Fig. 1 as well).

- **Key generation**. a is a global parameter shared by Alice and Bob. r_1 and r_2 are random binary polynomials and r_2 is the secret key. After $p = r_1 - a \cdot r_2$, p (public key) is sent to Bob (p is $n\log_2 q$-bit) and r_1 is discarded.
- **Encryption**. Bob uses three errors (binary polynomials) e_1, e_2, and e_3 to generate c_1 and c_2, where $\tilde{m}$ is obtained by multiplying each coefficient of m (binary polynomial) by $q/2$. c_1 and c_2 (both $n\log_2 q$-bit) are sent back to Alice.
- **Decryption**. Alice uses r_2 to recover the message m. One final decoder is needed, which returns '1' if the coefficient falls into the range of $(q/4, 3q/4)$ and '0' otherwise.

An inverted RBLWE-ENC was proposed in [8], where the integer coefficients are represented in the inverted range $(-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor - 1)$ (the three phases of Fig. 1 remain the same except minor changes on the encoding and output decoder). Under this denotation, no extra modular reductions are needed for the involved arithmetic operations under two's complement format. We also adopt this strategy in this paper.

Security. RBLWE is the structured variant of BLWE [5]. As BLWE retains the worst-cast hardness of the lattice problem, RBLWE-ENC is based on the average-case hardness [5].

Security analyses and attacks on BLWE can be seen in quite a number of reports [31], [32]; while the security analyses on RBLWE-ENC were firstly performed in [5] and then in [21]. It is shown that RBLWE-ENC achieves quantum/classic security of 73/84-bits and 140/190-bits, respectively, for parameters of

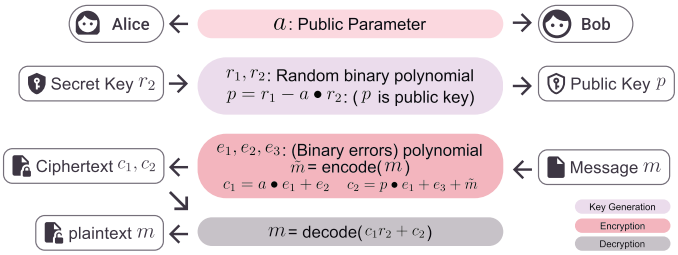

Fig. 1. Three major phases of RBLWE-ENC.

$(n, q) = (256, 256)$ and $(n, q) = (512, 256)$ [21]. Decryption failure rates and possible solutions were covered in [5]. These efforts confirm that RBLWE-ENC is a secure variant of BLWE that fits well with emerging lightweight applications [7].

III. ALGORITHMIC FORMULATION

As shown in Fig. 1, the most complicated and common operation within each phase is the polynomial multiplication followed by a polynomial addition (under two's complement representation, the polynomial subtraction in the key generation phase can also be realized by the polynomial addition), i.e., $W = GB + D + H \bmod f(x)$ where H is an extra binary polynomial needed for the encryption phase (producing c_2).

Consideration and Motivation. As polynomial multiplication is more complicated than polynomial addition, we can focus on the polynomial multiplication first. Meanwhile, we consider the fact that: (i) the parameter settings of RBLWE-ENC are not in favor of deploying widely used Number Theoretic Transform [33]; (ii) other fast algorithms such as Karatsuba algorithm (KA) may only be suitable for super high-performance applications as the implemented design involves very large area [29]; (iii) most of the existing designs are based on the traditional schoolbook polynomial multiplication algorithm in a general format (e.g., [7], [28]). Finally, following the discussion of **Existing Challenges** in Section I, a breakthrough in algorithmic derivation is seriously needed.

Proposed Derivation Strategy. We propose to use the schoolbook method (but with a novel algorithmic operation) to obtain the proposed algorithmic operation for RBLWE-ENC. A recent paper of [34] has proposed to use small-size coefficients oriented modulo operation to obtain a resource usage efficient computation for high-performance systolic accelerator design. We follow this style to use the binary polynomial (small-size coefficient) oriented derivation to obtain a novel **group decomposition initiated processing** algorithm that the major arithmetic operation can be executed through a timing-flexible format with minimized implementation complexity.

Proposed Algorithm. Define polynomial multiplication of RBLWE-ENC as $T = GB \bmod f(x)$, where $f(x) = x^n + 1$, $T = \sum_{i=0}^{n-1} t_i x^i$, $G = \sum_{i=0}^{n-1} g_i x^i$, and $B = \sum_{i=0}^{n-1} b_i x^i$ ($g_i \in \{0, 1\}$ and t_i and b_i are integers in $\mathbb{Z}_q$).

Then, we have

$$\begin{aligned} T =& Gb_0 \bmod f(x) + \cdots + Gb_{n-1}x^{n-1} \bmod f(x) \\ =& Gb_0 + (g_0 x + g_1 x^2 + \cdots - g_{n-1})b_1 \\ &+ \cdots \quad \cdots \quad \cdots \\ &+ (g_0 x^{n-1} - g_1 + \cdots - g_{n-1}x^{n-2})b_{n-1}, \end{aligned} \tag{1}$$

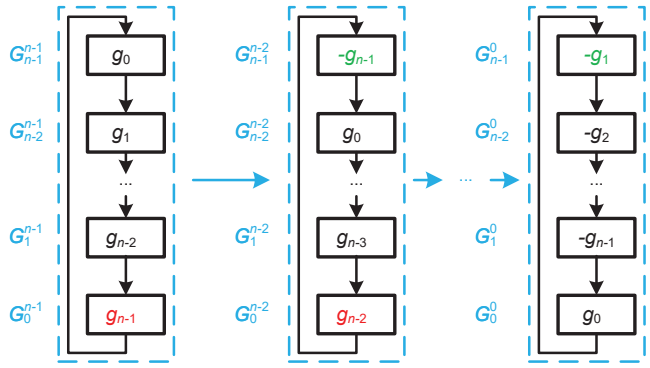

Fig. 2. The procedure to obtain $G^{(i-1)}$ from $G^{(i)}$ (from $i = n-1$ to 1).

where we have substituted x^n with $x^n \equiv -1$ (as $f(x) = x^n + 1$), such that $t_0 = g_0 b_0 + (-g_{n-1})b_1 + \cdots + (-g_1)b_{n-1}$, $\cdots$, $t_{n-2} = g_{n-2}b_0 + g_{n-3}b_1 + \cdots + (-g_{n-1})b_{n-1}$, $t_{n-1} = g_{n-1}b_0 + g_{n-2}b_1 + \cdots + g_0 b_{n-1}$ (we have compared both sides of (1) to list the corresponding coefficients with the same order of x). Then, we can define $G^{(0)} = g_0 + (-g_{n-1})x + \cdots + (-g_1)x^{n-1}$, $\cdots$, $G^{(n-1)} = g_{n-1} + (g_{n-2})x + \cdots + g_0 x^{n-1}$. We also define $G^{(i)}_j$ as the (j+1)th coefficient of $G^{(i)}$, for $0 \le i, j \le n-1$, e.g., $G^{(0)}_0 = g_0, \cdots, G^{(0)}_{n-1} = -g_1$.

Follow the **Proposed Derivation Strategy**, we first define $n = uv$, where u and v are integers (the selection of u or v depends on the actual applications, see Section V). Then, we can decompose the coefficients of $G^{(n-1)}$ into u groups. For simplicity of discussion, $G^{(n-1)}$ is used as an example first:

$$\begin{aligned} &\{g_{n-1}, g_{n-2}, \cdots, g_{n-v}\} \text{ (1st group)},\\ &\{g_{n-v-1}, g_{n-v-2}, \cdots, g_{n-2v}\} \text{ (2nd group)},\\ &\cdots \quad \cdots \quad \cdots,\\ &\{g_{v-1}, g_{v-2}, \cdots, g_0\} \text{ (uth group)}, \end{aligned} \tag{2}$$

which can be summarized as:

$$\begin{aligned} \overline{G^{(n-1)}_0} &= \{g_{n-1}, \cdots, g_{n-v}\},\\ \cdots \quad &\cdots \quad \cdots\\ \overline{G^{(n-1)}_{u-1}} &= \{g_{v-1}, \cdots, g_0\}, \end{aligned} \tag{3}$$

where $\overline{G^{(n-1)}_{0,0}} = g_{n-1}, \cdots, \overline{G^{(n-1)}_{n-1,0}} = g_{v-1}, \cdots, \overline{G^{(n-1)}_{n-1,v-1}} = g_0$. Other $G^{(i)}$ ($0 \le i \le n-2$) has similar decomposition.

Likewise, we can also have

$$\begin{aligned} \overline{B_0} &= \{b_0, b_1, \ldots, b_{v-1}\},\\ \cdots \quad &\cdots \quad \cdots\\ \overline{B_{u-1}} &= \{b_{n-v}, b_{n-v+1}, \ldots, b_{n-1}\}, \end{aligned} \tag{4}$$

for $\overline{B_{0,0}} = b_0, \cdots, \overline{B_{u-1,0}} = b_{n-v}, \cdots, \overline{B_{u-1,v-1}} = b_{n-1}$.

Thus, we can re-express (1) as

$$T = \sum_{i=0}^{n-1} t_i x^i = \sum_{i=0}^{n-1} \sum_{h=0}^{u-1} \sum_{k=0}^{v-1} \overline{G^{(i)}_{h,k}}\,\overline{B_{h,k}} x^i, \tag{5}$$

where one can calculate related point-to-point multiplications $\overline{G^{(i)}_{h,k}}\,\overline{B_{h,k}}$ by group processing. For instance, for t_{n-1} of (5), one can calculate the v number of coefficient-wise multiplications of $\{\overline{G^{(n-1)}_{0,0}}\,\overline{B_{0,0}}, \overline{G^{(n-1)}_{0,1}}\,\overline{B_{0,1}}, \cdots, \overline{G^{(n-1)}_{0,v-1}}\,\overline{B_{0,v-1}}\}$ first (namely $\overline{G^{(n-1)}_0}\,\overline{B_0}$), and then $\{\overline{G^{(n-1)}_{1,0}}\,\overline{B_{1,0}}, \overline{G^{(n-1)}_{1,1}}\,\overline{B_{1,1}}, \cdots, \overline{G^{(n-1)}_{1,n-1}}\,\overline{B_{1,n-1}}\}$, $\cdots$, and finally $\{\overline{G^{(n-1)}_{u-1,0}}\,\overline{B_{u-1,0}}, \cdots, \overline{G^{(n-1)}_{u-1,v-1}}\,\overline{B_{u-1,v-1}}\}$, where these point-wise multiplications are added within each group and then the results of these u groups can be accumulated together to produce t_{n-1}. Similar operation applies to other t_i (from $i = n-2$ down to 0).

Based on the derivation of (2)-(5), we can have the proposed algorithm for RBLWE-ENC as follows (includes D and H).

Algorithm 1: Proposed algorithm for RBLWE-ENC

Input : W, D, and B are polynomials with integer coefficients ($\log_2 q$-bit); G and H are binary polynomials; // $D = \sum_{i=0}^{n-1} d_i x^i$, $H = \sum_{i=0}^{n-1} h_i x^i$

Output: $W = GB \bmod (x^n + 1) + D + H$;

Initialization step

1 Decompose B as $\{\overline{B_0}, \overline{B_1}, \cdots, \overline{B_{u-1}}\}$; // see (4)

2 Decompose $G^{(n-1)}$ as $\{\overline{G^{(n-1)}_0}, \overline{G^{(n-1)}_1}, \cdots, \overline{G^{(n-1)}_{u-1}}\}$; // (3)

3 $\bar{t}_i = 0$;

Main step

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

5 **for** $j = 0$ *to* $u-1$ **do**

6 **for** $l = 0$ *to* $v-1$ **do**

7 $\bar{t}_i = \bar{t}_i + \overline{G^{(i)}_{j,l}}\,\overline{B_{j,l}}$; // follow (5)

8 **end**

9 **end**

10 $t_i = \bar{t}_i$;

11 $w_i = t_i + d_i + h_i$;

12 Obtain $G^{(i-1)}$ from $G^{(i)}$. // until $G^{(0)}$ is derived

13 **end**

Final step

14 Obtain output W from serially delivered w_i;

where Line 12 is processed according to the procedure below:

$$\begin{aligned} G^{(n-2)}_{n-1} &= -G^{(n-1)}_0, \quad G^{(n-2)}_j = G^{(n-1)}_{j-1},\\ G^{(n-3)}_{n-1} &= -G^{(n-2)}_0, \quad G^{(n-3)}_j = G^{(n-2)}_{j-1},\\ &\cdots \quad \cdots \quad \cdots,\\ G^{(0)}_{n-1} &= -G^{(1)}_0, \quad G^{(0)}_j = G^{(1)}_{j-1}, \end{aligned} \tag{6}$$

which can be denoted by the operations in Fig. 2 ($1 \le j \le n-1$): (i) all the coefficients of the $G^{(i)}$ are circularly shifted by one position; and (ii) the first coefficient (from the bottom, red color) of the $G^{(i)}$ becomes negative, as the nth coefficient of the $G^{(i-1)}$ (green color). Note that the processing sequence starts from $G^{(n-1)}$, where none of the coefficients is inverted.

Brief discussion of the unique advantage of the proposed algorithm. Algorithm 1 fulfills the principle of the proposed derivation strategy, i.e., group decomposition initiated processing (flexible speed) based on small-size coefficient oriented modulo derivation (small resource usage). This contribution

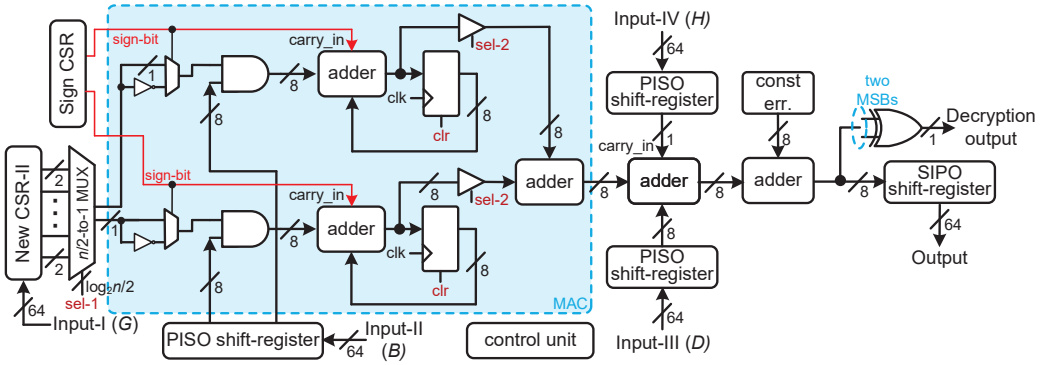

Fig. 3. The proposed accelerator for RBLWE-ENC (from Algorithm 1) based on the case of $v = 2$. PISO: parallel-in serial-out; SIPO: serial-in parallel-out.

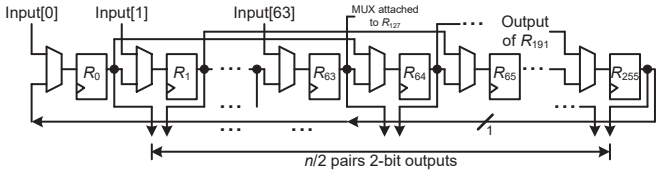

Fig. 4. Details of CSR-II ($v = 2$).

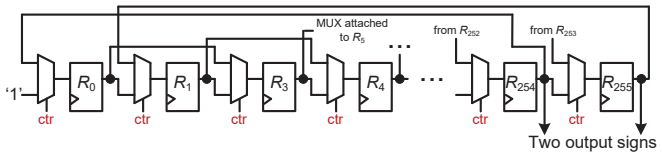

Fig. 5. Details of the sign-CSR ($v = 2$).

also leads to the unique advantage of the proposed algorithms over the existing ones [22]–[27] ([7], [8] do not provide the algorithmic operations). While compared to the very recent one of [34] (systolic computation, needs large resource usage), the proposed group decomposition allows us to have a fast computation (while maintaining efficient resource usage).

IV. Proposed Hardware Accelerator

Following the introduced algorithm, we present the proposed hardware accelerator for RBLWE-ENC in this section. Specifically, we consider: (i) optimized algorithm-to-architecture mapping techniques; (ii) delivering the final output in a serial format to obtain minimum resource usage; (iii) practical input/output processing setup, i.e., the values fed to (or delivered out) from the architectures are set as 64-bit (64-bit is the word-width of a modern memory).

Proposed Hardware Accelerator. We follow the algorithmic procedure of Algorithm 1 to design the proposed hardware accelerator. For simplicity of discussion, we present the accelerator based on the case of $v = 2$ first, see Fig. 3.

The accelerator in Fig. 3 consists of three main components, namely the input processing unit (for G and B), the multiplication-and-accumulation (MAC) unit, the output processing unit, and the control unit. The circular shift-register (CSR) for G (CSR-II) is shown in Fig. 4, where the shifting frequency has changed to once per u cycles after the initial values are loaded. Besides that, the two neighboring output bits of the CSR are combined to form $n/2$ pair of 2-bit outputs (see Fig. 4). These $n/2$ groups of outputs are then fed to an $n/2$-to-1 MUX to be delivered out according to Line 7 of Algorithm 1 with the explanation in (5). An additional sign-CSR is needed for the sign inverting related operations, as shown in Fig. 5. All the registers in the sign-CSR are initiated as '0' (coefficients of $G^{(n-1)}$ are all '0's), and then through the selection signal of the MUXes ("ctr", generated from the control unit) that '1' can be introduced into the sign-CSR, matching the specific output delivered from the $n/2$-to-1 MUX. The two sign-bits delivered out from the sign-CSR are then used to determine the output of the $n/2$-to-1 MUX, i.e., positive or negative based on the two's complement format. The PISO shift-register for B transforms the 64-input into 8-bit coefficients, and then delivers two neighboring coefficients to the MAC unit, matching the 2-bit output of the $n/2$-to-1 MUX. The following MAC unit executes the operation of Lines 4-9 of Algorithm 1. Finally, the output component delivers the desired result according to Lines 9-12 in Algorithm 1. A final adder is also needed for the addition with the constant error to produce the correct output (follow the suggestion in [28]).

Control unit. A control unit is needed for generating all the necessary control signals for the accelerator of Fig. 3. The control unit contains a finite state machine (FSM), which has six states: "reset", "load", "shift", "calculation", "output", and "done". In the reset state, the control unit resets all signals in the whole structure. The Load state in the compact accelerator only lasts for $n/8$ cycles and the shift state shifts the sign bit and G. The calculation state is n^2/v cycles long. The output state outputs one data every n/v cycles, and after that, the accelerator turns back to the calculation state.

Overall operation. In general, the proposed compact accelerator needs a total latency time of $(n/64 + un)$ cycles. The accelerator presented here can be easily extended for the designs based on other values of v. For example, the accelerator shown in Fig. 6 is based on the case of $v = 4$.

Summary. Overall, the proposed accelerator possesses unique features: (i) this accelerator involves low implementation complexity yet offers flexible processing speed; (ii) the proposed design has a practical input/output setup, which can easily be communicated with the modern processor.

V. Complexity and Comparison

Complexity Analysis. The proposed accelerator has v AND cells with $\log_2 q$-bit, v adders of $\log_2 q$-bit, v MUXes of 1-bit, v inverters of 1-bit, one tri-state gate of $\log_2 q$-bit, and

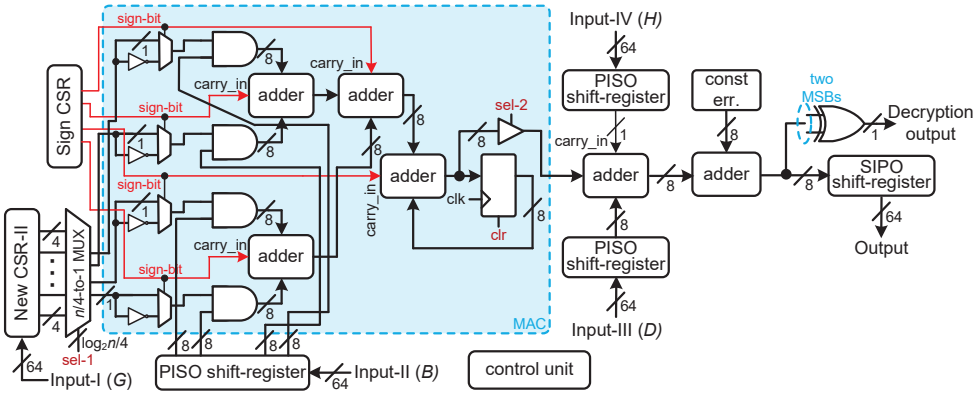

Fig. 6. Details of the proposed accelerator for the case of $v = 4$.

TABLE II
THEORETICAL AREA-TIME COMPLEXITIES OF VARIOUS HARDWARE ACCELERATORS/STRUCTURES FOR RBLWE-ENC

design	AND[1]	XOR[2]	adder[1]	MUX[3]	n-to-1 MUX[1,4]	register[1]	critical-path[5]	latency (Dec.)[6]	I/O setup[7]
[8]	1	1	1	2	2	1	$> T_A + T_{MUX_n} + T_{AD} + 2T_{MUX}$	$n^2 + 2n$	No
[22]	1	1	2	1	1	1	$> T_A + T_{MUX_n} + T_{AD}$	$n^2 + n$	No
[26]	1	1	1	2	1	1	$\geq T_A + T_{MUX_n} + T_{AD} + 2T_{MUX}$	$n^2 + 3n$	No
Proposed (general)	v	1	$v + 1$	0	0	1	$\geq T_A + T_{MUX_{n/v}} + (\log_2 v + 1)T_{AD}$	$n/64 + un$	Yes
Pro. Fig. 3	2	1	3	0	0	1	$\geq T_A + T_{MUX_{n/2}} + 2T_{AD}$	$n/4 + n(n/2 + 2)$	Yes

As the existing designs do not consider the practical setup for both input and output processing, we here only list the area-time complexities of the major structure (neither CSR nor different types of shift-registers are listed here), just for a fair comparison.
[1]: Refers to the one of $\log_2 q = 8$-bit. [2]: 1-bit.
[3]: The number of MUXes listed here mainly refers to the $\log_2 q = 8$-bit MUX (2-to-1), not including the ones of 1-bit (used in the proposed design).
[4]: The proposed accelerator has one (n/v)-to-1 MUX (v-bit).
[5]: T_A, T_{MUX}, T_{AD}, and T_{MUX_x}: delay of AND gate, $\log_2 q$-bit MUX, $\log_2 q$-bit adder, x-to-1 MUX, respectively.
[6]: The latency listed here is the total number of cycles, including the input loading time and the output delivery time, and we have used the decryption phase (Dec.) to calculate the latency cycles. Meanwhile, we have used $q = 256$ to calculate the latency cycles of the proposed accelerator.
[7]: Refers to the practical setup on the input and output processing components based on the regular word length in modern processors, e.g., 64-bit.

one register of $\log_2 q$-bit in the MAC. Besides that, one adder of $\log_2 q$-bit and eight XOR gates are needed for the output delivery. One (n/v)-to-1 MUX (v-bit) is required in the input component. Six different shift-registers and one control unit are also contained in the accelerator. The accelerator has a latency of $(n/64 + un)$ with a critical-path of $(\geq T_A + T_{MUX_{n/v}} + (\log_2 v + 1)T_{AD})$. Again, this proposed accelerator has a practical setup for input and output processing.

The theoretical area-time complexities of the proposed accelerator, in terms of the number of AND cells, XOR gates, adders, MUXes, and registers, as well as latency, are all listed in Table II along with the existing ones of [8], [22], [26]. Note that: the ones of [23], [24], [28] do not list their complexities; the designs of [25], [27], [29], [30], [34] targeted high-performance applications (we do not include them here).

The proposed accelerators have significantly better area-time complexities than the existing ones, based on the theoretical analysis shown in Table II. As the design of [26] has shown its efficiency over the other ones like [8], [22], we just discuss here the comparison with [26]. Compared with this recent compact architecture of [26], the proposed accelerator not only has the advantage in smaller time-complexity (also with smaller area usage), but also involves practical I/O setup. Overall, the proposed design's potential for application in practical environments is higher than the existing designs.

FPGA Implementation and Comparison. For a detailed comparison, we have also implemented the proposed accelerator on the FPGA platform and have obtained the implementation results for comparison with the existing designs.

Our experimental setup is described below. For parameters, we have followed the existing designs [7], [8], [22]–[27] to select $n = 256$ & $q = 256$ and $n = 512$ & $q = 256$; we have also selected $v = 2, 4, 8$ for the proposed accelerator. The proposed design is coded in VHDL, verified through ModelSim, and implemented on the Xilinx Virtex-7 (XC7V2000) and Intel Stratix-V (5SGXMA9N1F45C2) FPGA devices, respectively, through Vivado 2020.2 and Quartus Prime 20.1. All implementation results, including area usage, maximum frequency (Fmax, MHz), latency (#cycles), delay (delay=latency/Fmax), area-delay product (ADP), and ADP reduction (ADPR), are listed in Table III, along with the existing ones.

We can clearly see that the proposed accelerator has better performance than the state-of-the-art designs. The proposed accelerator offers a variety of efficiency in area-time complexities. For example, the proposed accelerator of $v = 8$ has at least 66.67% and 71.13% less ADP than the existing design of [26] for $n = 256$ and $n = 512$ on Virtex-7, respectively. When comparing with the design of [23], we have made specific

TABLE III
COMPARISONS OF THE FPGA IMPLEMENTATION RESULTS OF VARIOUS RBLWE-ENC PQC ACCELERATORS

design	n	phase[1]	device	LUT	FF	Slice/ALM	Fmax	CL[2]	PL[3]	delay[4]	ADP[5]	ADPR[6]
[22]*	256	Enc./Dec.	Stratix-V	-	-	1,864	317	65,536	65,792	207.6	386.87	-
Pro. ($v=2$)	256	Enc./Dec.	Stratix-V	-	-	3,009	283	33,279	33,343	117.82	354.52	8.36%
Pro. ($v=4$)	256	Enc./Dec.	Stratix-V	-	-	3,030	259	16,895	16,959	65.5	198.47	48.70%
Pro. ($v=8$)	256	Enc./Dec.	Stratix-V	-	-	2,573	213	8,703	8,767	41.16	105.90	72.63%
[26][7]	256	Enc/Dec	Virtex-7	737	2,691	657	288	66,048	66,304	229.91	151.05	-
[23]*,#	256	Enc./Dec.	Virtex-7	380	640	165	434	66k	66k+256	152.66	25.19#	-171.74%#
[28]#	256	Enc/Dec	Virtex-7	213	336	71	510	66,304	66,304+256	131.51	9.27#	-
Pro. ($v=8$)	256	Enc./Dec.	Virtex-7	141#	35#	56#	641#	8,448#	8,448#	13.18#	0.74#	92.02%#
				3,363	3,343	1,154	201	8,703	8,767	43.62	50.34	66.67%
[22]*	512	Enc./Dec.	Stratix-V	-	-	3,551	297	262,144	262,656	884.4	3,140.38	-
Pro. ($v=2$)	512	Enc./Dec.	Stratix-V	-	-	5,934	275	132,095	132,223	480.81	2,853.13	9.15%
Pro. ($v=4$)	512	Enc./Dec.	Stratix-V	-	-	5,980	245	66,559	66,687	272.19	1,627.70	48.17%
Pro. ($v=8$)	512	Enc./Dec.	Stratix-V	-	-	4,784	222	33,791	33,919	152.79	730.95	76.72%
[26][7]	512	Enc/Dec	Virtex-7	1,335	5,269	1,292	269	263,168	263,680	978.66	1264.43	-
Pro. ($v=8$)	512	Enc./Dec.	Virtex-7	6,362	6,192	2,088	194	33,791	33,919	174.84	365.07	71.13%

*: The existing designs' reported results do not include the full resource usage for practical input/output processing components such as CSRs (or other types of shift-registers). Hence, the **actual ADPR of the proposed accelerator is much larger than the listed one here**.
#: For a fair comparison with [23], [28], where almost all CSRs have not been included, we just listed the performance of the proposed accelerator core (without CSRs) as well.
[1]: Working phase of the hardware structure.
[2]: CL: Calculation latency (major computation of decryption phase). The existing designs are mostly based on the decryption phase, e.g., [23], [28].
[3] : PL: Practical latency (includes input loading and output delivery).
[4] : Delay=PL/Fmax μs.
[5]: ADP=#Slice(ALM)$\times$delay($\times 10^{-3}$).
[6]: ADPR: ADP reduction (based on the same FPGA device with the same n, and between the same type of designs).
[7]: The data is obtained based on the released open-access code of [26].

arrangements on the actual implementation since the design of [23] did not include the input and output resources (such as CSRs). As seen from Table III, one can see that the proposed accelerator has significantly better ADP than [23] at the same condition of input/output setup, i.e., 92.02%. A much more efficient situation applies to the comparison with [28] (Table III). Overall, one can conclude that the proposed accelerator obtains the best performance among all the listed designs.

Note that we have used the cycle-count-based method combined with FSMs to implement the proposed control unit. A major advantage of such setup is that no external resource is needed to manage data loading/storing operations. This design consideration can be regarded as shifting the workload from memory interaction into the structure, although it might affect the overall timing performance (and area usage), as shown in Table III. Nevertheless, the proposed setup makes the designed accelerator more practical for FPGA-based applications.

Discussions and Future Research. To the authors' best knowledge, the work of [34] also has practical input and output setup, but its resource usage on FPGA is too large. The proposed hardware design, however, has an efficient area occupation on different FPGAs while offering flexible processing speed and maintaining practical input and output setup. The proposed accelerator is hence desirable for FPGA-based lightweight applications. We also recommend choosing relatively small values of v since an accelerator with smaller area usage is more attractive in lightweight applications.

Overall, the proposed accelerators have constant time running, which is resistant to regular timing attacks [35]. While the major focus of this paper is to present a novel hardware-implemented accelerator for RBLWE-ENC on the FPGA platform, the study on side-channel analysis is out of the scope of this work. Nevertheless, we want to emphasize that the side-channel attacks and countermeasures have been explored in [7] and can be extended for the proposed accelerator (one of our future research). Meanwhile, there still exist two main challenges: (i) novel fast hardware-based algorithmic innovation for further complexity reduction on different FPGAs; (ii) further develop this scheme for actual application in FPGA-based platform. Our future efforts will focus on these areas.

Other Related Works. Other hardware implementation works for different PQC schemes can be seen at [13], [16], [17], [34], [36]–[41], not necessarily limited to pure hardware design and FPGA platforms. Due to the research focus differences, we do not list them for comparison. Nevertheless, we recognize these designs are important works in the field.

VI. CONCLUSION

This paper presents a novel hardware implementation for RBLWE-ENC PQC accelerator on the FPGA platform. We have proposed three layers of efforts, including algorithmic derivation, architectural innovation, and complexity & comparison, to conduct the proposed work. The proposed accelerator for RBLWE-ENC involves efficient resource usage and offers flexible speed choices. Our experimental results show that the proposed accelerator outperforms the existing designs. The proposed design strategy is highly efficient, and the proposed accelerator can be extended further for different FPGA-based lightweight applications. It is expected that the proposed work can stimulate further advancement in lightweight PQC.

VII. ACKNOWLEDGMENT

This work was supported by NSF SaTC-2020625 and NIST-60NANB20D203.

REFERENCES

[1] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-quantum cryptography*, pp. 1–14, Springer, 2009.

[2] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[3] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proc. 35th ann. sym. foundations of computer science*, pp. 124–134, Ieee, 1994.

[4] "PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates." https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4.

[5] J. Buchmann et al., "High-performance and lightweight lattice-based public-key encryption," in *ACM int. workshop on IoT privacy, trust, and security*, pp. 2–9, 2016.

[6] D. Micciancio and C. Peikert, "Hardness of sis and lwe with small parameters," in *Annual Cryptology Conference*, pp. 21–39, Springer, 2013.

[7] A. Aysu et al., "Binary Ring-LWE hardware with power side-channel countermeasures," in *DATE'2018*, pp. 1253–1258, IEEE, 2018.

[8] 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.

[9] "National science foundation (nsf) 2022 secure and trustworthy cyberspace principal investigators' meeting (satc pi meeting '22)–break out group reports/slides: Security in a post-quantum world," p. slides page 4, 2022.

[10] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.

[11] V. Lyubashevsky et al., "On ideal lattices and learning with errors over rings," in *EUROCRYPT'2010*, pp. 1–23, Springer, 2010.

[12] J. Xie et al., "Special session: The recent advance in hardware implementation of post-quantum cryptography," in *IEEE VTS'2020*, pp. 1–10, IEEE, 2020.

[13] J. Howe et al., "Lattice-based encryption over standard lattices in hardware," in *DAC'2016*, pp. 1–6, IEEE, 2016.

[14] S. S. Roy et al., "Compact ring-lwe cryptoprocessor," in *CHES*, pp. 371–391, 2014.

[15] S. Bian et al., "Filianore: better multiplier architectures for lwe-based post-quantum key exchange," in *DAC'2019*, pp. 1–6, 2019.

[16] H. Nejatollahi, S. Shahhosseini, R. Cammarota, and N. Dutt, "Exploring energy efficient quantum-resistant signal processing using array processors," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1539–1543, IEEE, 2020.

[17] H. Nejatollahi, F. Valencia, S. Banik, F. Regazzoni, R. Cammarota, and N. Dutt, "Synthesis of flexible accelerators for early adoption of ring-lwe post-quantum cryptography," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 2, pp. 1–17, 2020.

[18] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *International Conference on Selected Areas in Cryptography*, pp. 68–85, Springer, 2013.

[19] A. Aysu, C. Patterson, and P. Schaumont, "Low-cost and area-efficient fpga implementations of lattice-based cryptography," in *2013 IEEE international symposium on hardware-oriented security and trust (HOST)*, pp. 81–86, IEEE, 2013.

[20] T. Fritzmann and J. Sepúlveda, "Efficient and flexible low-power NTT for lattice-based cryptography," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 141–150, IEEE, 2019.

[21] F. Göpfert et al., "A hybrid lattice basis reduction and quantum search attack on LWE," in *PQCrypto*, pp. 184–202, Springer, 2017.

[22] P. He et al., "Novel low-complexity polynomial multiplication over hybrid fields for efficient implementation of binary ring-lwe post-quantum cryptography," *IEEE JETCAS*, vol. 11, no. 2, pp. 383–394, 2021.

[23] K. Shahbazi and S.-B. Ko, "Area and power efficient post-quantum cryptosystem for iot resource-constrained devices," *Microprocessors and Microsystems*, vol. 84, p. 104280, 2021.

[24] J. Xie et al., "Efficient implementation of finite field arithmetic for binary Ring-LWE post-quantum cryptography through a novel lookup-table-like method," in *DAC'2021*, vol. 21, pp. 1–6, 2021.

[25] J. Xie, P. He, X. Wang, and J. L. Imaña, "Efficient hardware implementation of finite field arithmetic $ab + c$ for binary ring-lwe based post-quantum cryptography," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1222–1228, 2022.

[26] B. J. Lucas and et al., "Lightweight hardware implementation of binary ring-lwe pqc accelerator," *IEEE Computer Architecture Letters*, 2022.

[27] J. L. Imaña, P. He, T. Bao, Y. Tu, and J. Xie, "Efficient hardware arithmetic for inverted binary ring-lwe based post-quantum cryptography," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[28] D. Xu, X. Wang, Y. Hao, Z. Zhang, Q. Hao, and Z. Zhou, "A more accurate and robust binary ring-lwe decryption scheme and its hardware implementation for iot devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 8, pp. 1007–1019, 2022.

[29] P. He, Y. Tu, J. Xie, and H. Jacinto, "Kina: Karatsuba initiated novel accelerator for ring-binary-lwe (rblwe)-based post-quantum cryptography," *TechRxiv*, pp. 1–12, 2022.

[30] T. Bao, J. L. Imaña, P. He, and J. Xie, "Work-in-progress: High-performance systolic hardware accelerator for rblwe-based post-quantum cryptography," in *2022 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 5–6, IEEE, 2022.

[31] J. Buchmann, F. Göpfert, R. Player, and T. Wunderer, "On the hardness of lwe with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack," in *International Conference on Cryptology in Africa*, pp. 24–43, Springer, 2016.

[32] D. Micciancio, "On the hardness of learning with errors with binary secrets," *Theory of Computing*, vol. 14, no. 1, pp. 1–17, 2018.

[33] J. M. Pollard, "The fast fourier transform in a finite field," *Mathematics of computation*, vol. 25, no. 114, pp. 365–374, 1971.

[34] T. Bao and et al., "Systolic acceleration of polynomial multiplication for kem saber and binary ring-lwe post-quantum cryptography," in *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, pp. 157–160, IEEE, 2022.

[35] T. Schneider, A. Moradi, and T. Güneysu, "ParTI–towards combined hardware countermeasures against side-channel and fault-injection attacks," in *Annual International Cryptology Conference*, pp. 302–332, Springer, 2016.

[36] J. Howe, T. Oder, M. Krausz, and T. Güneysu, "Standard lattice-based key encapsulation on embedded devices," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 372–393, 2018.

[37] D. Liu, C. Zhang, H. Lin, Y. Chen, and M. Zhang, "A resource-efficient and side-channel secure hardware implementation of ring-lwe cryptographic processor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1474–1483, 2018.

[38] 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 Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 157–166, 2014.

[39] V. B. Dang, F. Farahmand, M. Andrzejczak, K. Mohajerani, D. T. Nguyen, and K. Gaj, "Implementation and benchmarking of round 2 candidates in the nist post-quantum cryptography standardization process using hardware and software/hardware co-design approaches," *Cryptology ePrint Archive: Report 2020/795*, 2020.

[40] E. Camacho-Ruiz, S. Sánchez-Solano, P. Brox, and M. C. Martínez-Rodríguez, "Timing-optimized hardware implementation to accelerate polynomial multiplication in the ntru algorithm," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 3, pp. 1–16, 2021.

[41] K. Basu, D. Soni, M. Nabeel, and R. Karri, "Nist post-quantum cryptography-a hardware evaluation study," *Cryptology ePrint Archive*, 2019.