IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 June 2024, accepted 6 July 2024, date of publication 11 July 2024, date of current version 22 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3426990

RESEARCH ARTICLE

Multi-LFSR Architectures for BRLWE-Based Post Quantum Cryptography

SHAIK AHMADUNNISA AND SUDHA ELLISON MATHE, (Member, IEEE)
School of Electronics Engineering, VIT-AP University, Vijayawada 522241, India
Corresponding author: Sudha Ellison Mathe (ellison.ece@gmail.com)

ABSTRACT The advancement in quantum computing has led to a significant progress in the development of public-key cryptosystems, referred as Post Quantum Cryptography (PQC) which has robust security to withstand both classical and quantum attacks. Lattice-based cryptography, one of the most promising PQC candidate offers low complexity and has strong security proof relying on the hardness of Learning-with-errors (LWE) problem. A variant of LWE, Ring-learning-with-error (RLWE) performs arithmetic operations over a polynomial ring and has more efficient implementations compared to LWE. Recent works propose Binary-ring-learning-with-error (BRLWE), a new variant of RLWE which has less key size and more efficient implementations compared to both LWE and RLWE-based schemes. In this paper, an algorithm is developed for BRLWE-based scheme based on decomposing the arithmetic operation $H.L\ mod(x^n+1)+M$ into desired number of segments. The arithmetic operation includes polynomial multiplication and addition over the ring x^n+1 where H and M are two integer polynomials and L is a binary polynomial. We illustrate two efficient hardware architectures Dual-LFSR (DL) and Quad-LFSR (QL) to enable parallel execution of individual segments employing LFSR structures to have a significant reduction in latency compared to the existing works. Despite of having larger area, the reduction in latency leads to an improvement in other performance metrics such as delay, Area-Delay Product (ADP), Power-Delay Product (PDP), throughput and efficiency making the proposed structures well suitable for PQC schemes. Experimental results show that the proposed architectures when compared with the recently reported work has 23% and 25% ADP improvement with DL and QL structures respectively when $n = 256$.

INDEX TERMS Post quantum cryptography (PQC), lattice-based cryptography (LBC), polynomial multiplication, learning-with-error (LWE), ring-learning-with-error (RLWE), binary-ring-learning-with-error (BRLWE).

I. INTRODUCTION

The advent in quantum computing placed a significant demand for the development of public-key cryptosystems that can withstand security against quantum attacks. The accomplishment of quantum algorithms such as Shor's algorithm can factorize large numbers and break the widely used public-key cryptosystems namely Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC) [1], [2]. This leads to the field of Post Quantum Cryptography (PQC) to develop robust algorithms that could resist all possible classical and quantum attacks. In this view, the National Institute of Standards and Technology (NIST) has initiated the Post Quantum Cryptography (PQC) standardization process to resist both classical and quantum attacks. There are many PQC schemes such as lattice-based, code-based, hash-based, isogeny-based, and multivariate-quadratic cryptographic schemes [3]. Amongst all the PQC standardization processes, lattice-based cryptography (LBC) is considered as the most promising one due to its efficient implementations and strong security proof relying on the hardness of Learning with Error (LWE) [4], [5].

LWE, first introduced by Regev in 2005, is a computationally hard problem involving complex matrix-vector multiplication. Ring-learning-with-error (RLWE), a variant of LWE comprises less key size compared to original

The associate editor coordinating the review of this manuscript and approving it for publication was Christian Pilato.

LWE and performs arithmetic operations over a polynomial ring. A new variant of RLWE, Binary-ring-learning-with-error (BRLWE) comprises less key size and involves much more efficient implementations compared to both LWE and RLWE-based schemes. This scheme is similar in comparison to RLWE-based scheme, but it uses binary errors instead of Gaussian distributed errors in RLWE-based scheme [6]. Recently, some works have been proposed on the low-complexity implementation of BRLWE-based schemes [3], [6], [7], [8], [9], [10], [11], [12], [13], [14].

Some of the recent works comprising BRLWE-based schemes are as follows: BRLWE was first proposed in [7] with software implementation by replacing Gaussian noise distribution in RLWE-based schemes with a binary distribution. In 2018, a significant development is made by designing an efficient hardware implementation for BRLWE-based scheme in [8]. It ensures resistance to side-channel attacks by addressing certain aspects of DPA (Differential Power Analysis) and SPA (Simple Power Analysis) attacks. It has less area and high performance compared to the previous LWE-based architectures [15], [16] and RLWE-based architectures [17], [18], [19], [20]. In 2019, Ebrahimi et al. proposed a pair of hardware architectures: high speed and ultra-lightweight for IoT devices with limited resources and edge computing capabilities in [9]. They less area-time product compared to the best of previous RLWE implementations [17], [19], ECC implementations [21] and the other BRLWE implementation [8]. However, the work in [8] and [9] do not report their algorithms. In 2020, high-level synthesis (HLS)-based design-space exploration of PQC implementations was proposed in [10]. It comprises a serial processing structure with circular shift register but do not have proper sign control unit for performing polynomial multiplication. It has less Area-Delay Product (ADP) compared to [9]. In 2020, the work proposed in [22] demonstrates Differential Power Analysis (DPA) masking countermeasures for BRLWE-based schemes with a substantial improvement in speed and efficiency by eliminating PRNG module. Though, it consumes more resources due to the redundancy of operating modules which is required to offer more resistance to SPA and DPA attacks, it has a greater speed compared to the other works [17], [19] and an increased efficiency compared to [8]. In 2021, an implementation based on look-up-table structure was proposed by Xie and He in [11]. By precomputing and storing the results in tables, this technique simplifies finite field arithmetic operations and reduces the computational complexity involved in modular arithmetic computations. It has less ADP compared to [9]. The implementation proposed in [13] performs column-based multiplication with anti-circular rotation. Employing this method, multiplication is done column-wise, with one rotation executed per cycle. It is suitable for resource ultra-constrained applications and has less area-time product compared to [8]. The work proposed by He and Guin in [6] assumes the input is directly fed to the multiplexer. It is particularly suitable for resource ultra-constrained applications and has less ADP compared to [9]. The implementation proposed in [12] uses grouping of input coefficients to load the inputs in parallel to the processing blocks. Though this structure has an additional sign control unit, it has less ADP compared to the other BRLWE-based schemes [8], [9] and RLWE-based scheme [23]. In 2022, another pair of low-complexity and high-speed architectures are proposed in [14] suitable for lightweight applications and has less ADP compared to [6], [9], and [11]. The work proposed by Imana et al. in [3] comprises two efficient hardware architectures using Linear Feedback Shift Register (LFSR) structures. The first architecture increases processing speed whereas the second architecture effectively reduces area-complexity. These structures have less ADP compared to [8], [9], and [12]. A pair of hardware architectures, high-performance and lightweight are proposed in [24] for IoT terminal devices based on re-defined decryption function by analyzing the overflow and noise polynomial effect on the discriminant interval. The decryption accuracy is significantly increased compared to the other works [6], [8], [9], [13], [17], [22]. In 2023, a hardware implementation is proposed by Karim et al. in [25] comprising column-based multiplication technique where two consecutive coefficients are processed in each cycle to improve the computational efficiency. It has less ATP compared to the other works [6], [8], [13], [14]. The implementation proposed in [26] is suitable for resource-constrained applications where a two-level computation technique is employed for performing polynomial multiplication and it has less ADP compared to the other works [6], [13], [24].

Though significant progress has been made in BRLWE-based schemes, the ADP of these works can further be improved. In this paper, we aim to propose efficient architectures comprising multiple LFSR structures for performing arithmetic operation $HL+M$ over a polynomial ring x^n+1 where H and M are two integer polynomials, L is a binary polynomial and n is the highest degree of polynomial. The major contributions of this work are stated as follows:

- A desired algorithm is obtained by breaking down the required mathematical operations for performing the arithmetic operation $HL+M$ into smaller segments.
- Two efficient hardware architectures Dual-LFSR (DL) and Quad-LFSR (QL) are proposed for BRLWE-based scheme employing parallel processing technique to reduce the computational time 1.6× and 2.7× compared to [9].
- The reduction in latency of the proposed structures yields to a significant improvement in the other performance metrics such as delay, Area-Delay Product (ADP), Power-Delay Product (PDP), throughput and efficiency compared to the existing works [3], [9], [12].

Following the previous reports [3], [6], [7], [8], [9], [11], [13], we choose the set of parameters comprising (n, q) as $(256, 256)$ and $(512, 256)$ respectively.

The rest of the paper is organized as follows: Section II introduces preliminaries, Section III gives algorithm formulation, Section IV includes the proposed architecture, and Section V comprises results and discussions. Finally, we conclude the paper with conclusions given in Section VI.

II. PRELIMINARIES

In 2009, Regev et al. introduced a new lattice-based cryptosystem relying on the mathematical hardness of Learning with errors (LWE) problem [27]. The search version of the LWE problem is to determine the secret s from a linear combination $a.s+e$, where a is a known value and e is an error introduced according to certain distribution such as Gaussian or binary.

As it is computationally infeasible to determine s from the linear equation, a new variant of LWE scheme, named as Ring-LWE (RLWE) was introduced in 2010 [28]. It uses ideal lattices and the arithmetic operations such as addition and multiplication are performed over the polynomial ring R_q where $R_q = \mathbb{Z}_q/f(x)$ and $f(x)$ is an irreducible polynomial. Considering $f(x) = x^n + 1$ causes the elements within the ring to shift in anti-circular rotation. It requires less key size and is considered to be more efficient for both software and hardware implementations compared to LWE-based schemes. In RLWE-based schemes, the addition is a simple operation carried out element-by-element followed by modular reduction whereas multiplication is a complex operation. Though, there are different techniques to perform polynomial multiplication such as Toom Cook, Karatsuba, Schoolbook and Number Theoretic Transform (NTT), NTT-based polynomial multiplication is considered to be an efficient technique in RLWE-based schemes. Apart from the addition and multiplication modules, this scheme generates errors using discrete Gaussian distribution. As the implementation of this module is expensive, it is not suitable for resource constrained devices.

In 2016, a new variant of RLWE-based scheme named Binary-RLWE (BRLWE) has been proposed to overcome the drawback of RLWE-based schemes by replacing Gaussian errors with binary errors. Hence, this BRLWE-based scheme does not require Gaussian distribution and NTT-based polynomial multiplication resulting in a reduced key size and low complexity implementation compared to both LWE and RLWE-based schemes [6], [29]. It takes the advantage of representing the coefficients of integer polynomial in inverted range $(\lfloor\frac{-q}{2}\rfloor, \lfloor\frac{q}{2}\rfloor - 1)$, where the modular addition and subtraction operations could be performed efficiently without any extra reduction unit [12].

Security of BRLWE-Based Schemes: It relies on the basis of average case hardness and achieves 73 bits and 140 bits of quantum security for the set of parameters $(n, q) = (256, 256)$ and $(512, 256)$ respectively.

In this section, we have a brief discussion on the operational phases involved in BRLWE-based scheme as follows:

A. KEY GENERATION

This step comprises the calculation of the public key, $p = r_1 - a.r_2$, where r_1 and r_2 are random binary polynomials, r_2 is the secret key, a is a global parameter that is shared by the two parties [14].

B. ENCRYPTION

The input message comprising $0's$ and $1's$ is encoded into a polynomial $\widetilde{m}$ by multiplying each coefficient with $q/2$. In this phase ciphertexts c_1 and c_2 are generated by performing the operations: $c_1 = ae_1 + e_2$ and $c_2 = pe_1 + e_3 + \widetilde{m}$, where e_1, e_2 and e_3 are random binary polynomials and $\widetilde{m}$ is the encoded message.

C. DECRYPTION

The original message is decrypted back in this phase by computing $c = c_1 r_2 + c_2$ and applying the result to the threshold decoder function, which returns $'1'$ if the coefficients lie within the range of $(q/4, 3q/4)$, else $'0'$.

From all these operational phases (key generation, encryption and decryption) we identify the key arithmetic operation in each phase of BRLWE-based scheme involves polynomial polynomial multiplication and polynomial addition. In the next sections, we derive an algorithm and design efficient architectures for performing the key arithmetic operation (polynomial multiplication and addition) in cryptoprocessors.

III. ALGORITHM FORMULATION

In this section, we derive an algorithm to perform the arithmetic operation $H.L \ mod(x^n + 1) + M$ involved in BRLWE-based scheme. The conventional algorithm [3] is modified considerably by splitting the computations into smaller segments required for parallel executions. Let us consider the equation

$$W = HL \ mod(x^n + 1) + M \tag{1}$$

where H and M are integer polynomials and L is a binary polynomial given by $H = \sum_{i=0}^{n-1} h_i.x^i$, $L = \sum_{i=0}^{n-1} l_i.x^i$, $M = \sum_{i=0}^{n-1} m_i.x^i$, $W = \sum_{i=0}^{n-1} w_i.x^i$

From the integer polynomial H, we express the following terms $Hx, Hx^2, \ldots, Hx^{n-1}$ considering $x^n \equiv -1$.

$$\begin{aligned} HxY &= -h_{n-1} + h_0.x + h_1.x^2 + \ldots.. + h_{n-2}.x^{n-1}, \\ Hx^2 &= -h_{n-2} - h_{n-1}.x + h_0.x^2 + \ldots.. + h_{n-3}.x^{n-1}, \\ &\ldots.. \\ Hx^{n-1} &= -h_1 - h_2.x - h_3.x^2 - \ldots.. + h_0.x^{n-1} \end{aligned} \tag{2}$$

We consider $[Hx^i]_j$ as j^{th} coefficient of integer polynomial Hx^i (For example, $[Hx^0]_0 = h_0$, $[Hx^1]_0 = -h_{n-1}$).

On multiplying the two polynomials, H and L we obtain the resultant as shown below:

$$\begin{aligned} HL = h_0(l_0 + l_1.x + l_2.x^2 + \ldots.. + l_{n-1}.x^{n-1}) \\ + h_1.x(l_0 + l_1.x + l_2.x^2 + \ldots.. + l_{n-1}.x^{n-1}) \\ + h_2.x^2(l_0 + l_1.x + l_2.x^2 + \ldots.. + l_{n-1}.x^{n-1}) \end{aligned}$$

$$+\ldots\ldots h_{n-1}.x^{n-1}(l_0 + l_1.x + l_2.x^2 + \ldots.. + l_{n-1}.x^{n-1}) \tag{3}$$

The Eqn. 3 is the resultant of multiplication between two polynomials H and L with highest degree $2n-2$. Further, as we have ($x^n \equiv -1$), Eqn. 3 is simplified as follows:

$$\begin{aligned} &HL\ mod(x^n+1)\\ &= h_0(l_0 + l_1.x + l_2.x^2 + \ldots.. + l_{n-1}.x^{n-1})\\ &\quad + h_1(l_0.x + l_1.x^2 + l_2.x^3 + \ldots.. - l_{n-1})\\ &\quad + h_2(l_0.x^2 + l_1.x^3 + l_2.x^4 + \ldots.. - l_{n-2} - l_{n-1}.x)\\ &\quad + \ldots\ldots h_{n-1}.(l_0.x^{n-1} - l_1 - l_2.x + \ldots.. - l_{n-1}.x^{n-2}) \end{aligned} \tag{4}$$

It can be further simplified as follows:

$$\begin{aligned} &HL\ mod(x^n+1)\\ &= (h_0.l_0 - h_1.l_{n-1} - h_2.l_{n-2} - \ldots. - h_{n-1}.l_1)\\ &\quad x(h_0.l_1 + h_1.l_0 - h_2.l_{n-1} - \ldots. - h_{n-1}.l_2)\\ &\quad + x^2(h_0.l_2 + h_1.l_1 + h_2.l_0 - h_3.l_{n-1} - \ldots. - h_{n-1}.l_3)\\ &\quad + \ldots\ldots x^{n-1}(h_0.l_{n-1} + h_1.l_{n-2} + h_2.l_{n-3}\\ &\quad + \ldots.. + h_{n-1}.l_0) \end{aligned} \tag{5}$$

The obtained Eqn. 5 is the conventional equation derived as a result of multiplying the polynomials H and L over the polynomial ring x^n+1. The terms $Hx, Hx^2, \ldots.., Hx^{n-1}$ from Eqn. 2 are replaced in Eqn. 5 as follows:

$$\begin{aligned} &HL\ mod(x^n+1)\\ &= \sum_{i=0}^{n-1}[Hx^{n-1}]_i l_{n-1-i} + x(\sum_{i=0}^{n-1}[Hx^{n-2}]_i l_{n-1-i})\\ &\quad + x^2(\sum_{i=0}^{n-1}[Hx^{n-3}]_i l_{n-1-i}) + \ldots.. + x^{n-1}(\sum_{i=0}^{n-1}[H]_i l_{n-1-i}) \end{aligned} \tag{6}$$

In our work, we decompose the coefficients of Eqn. 6 into parallel segments such that each and every individual segment can be executed simultaneously.

For example, we decompose the coefficients of Eqn. 6 into two segments as follows:

$$\begin{aligned} &HL\ mod(x^n+1)\\ &= \sum_{i=0}^{n/2-1}[Hx^{n-1}]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[Hx^{n-1}]_i l_{n-1-i}\\ &\quad + x(\sum_{i=0}^{n/2-1}[Hx^{n-2}]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[Hx^{n-2}]_i l_{n-1-i})\\ &\quad + x^2(\sum_{i=0}^{n/2-1}[Hx^{n-3}]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[Hx^{n-3}]_i l_{n-1-i})\\ &\quad + \ldots.. + x^{n-1}(\sum_{i=0}^{n/2-1}[H]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[H]_i l_{n-1-i}) \end{aligned} \tag{7}$$

As shown in the Eqn. 7 the coefficients of polynomial multiplication are split into two groups to perform parallel executions. The obtained resultant is added to the coefficients of integer polynomial M to get the desired functionality as shown below:

$$\begin{aligned} &HL\ mod(x^n+1)+M\\ &= \sum_{i=0}^{n/2-1}[Hx^{n-1}]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[Hx^{n-1}]_i l_{n-1-i} + m_0\\ &\quad + x(\sum_{i=0}^{n/2-1}[Hx^{n-2}]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[Hx^{n-2}]_i l_{n-1-i} + m_1)\\ &\quad + x^2(\sum_{i=0}^{n/2-1}[Hx^{n-3}]_i l_{n-1-i} + \sum_{i=n/2}^{n-1}[Hx^{n-3}]_i l_{n-1-i} + m_2)\\ &\quad + \ldots\ldots + x^{n-1}(\sum_{i=0}^{n/2-1}[H]_i l_{n-1-i}\\ &\quad + \sum_{i=n/2}^{n-1}[H]_i l_{n-1-i} + m_{n-1}) \end{aligned} \tag{8}$$

Algorithm 1 Algorithmic Operation for Parallel Execution of $HL+M$ in Polynomial Ring $Z_q/(x^n+1)$

```
Input: $H, M$ and $W$ are integer polynomials.
(coefficients $\in Z_q$); $L$ is a binary polynomial.
$d$ determines the number of parallel processing segments ($d$ is an even number)
Output: $W = HL\ mod(x^n+1)+M$
Initialization step:
$\overline{W} = \sum_{i=0}^{n-1}\overline{w}_i x^i$;
Main step:
1: for $j = 0$ to $n-1$ do
2:     for $i = 0$ to $n/d-1$ do
3:         $\overline{w}_j = \overline{w}_j + \sum_{s=0}^{d-1}[Hx^{i+s.n/d}]_j.l_{i+s.n/d}$;
4:     end for
5: end for
$W = \overline{W} + M$;
Final step:
Deliver all the coefficients of $W$ serially.
```

The same concept can be extended by decomposing the coefficients of Eqn. 6 into four segments as follows:

$$\begin{aligned} &HL\ mod(x^n+1)+M\\ &= \sum_{i=0}^{n/4-1}[Hx^{n-1}]_i l_{n-1-i} + \sum_{i=n/4}^{n/2-1}[Hx^{n-1}]_i l_{n-1-i}\\ &\quad + \sum_{i=n/2}^{3n/4-1}[Hx^{n-1}]_i l_{n-1-i} + \sum_{i=3n/4}^{n-1}[Hx^{n-1}]_i l_{n-1-i} + m_0\\ &\quad + x(\sum_{i=0}^{n/4-1}[Hx^{n-2}]_i l_{n-1-i} + \sum_{i=n/4}^{n/2-1}[Hx^{n-2}]_i l_{n-1-i} \end{aligned}$$

$$+\sum_{i=n/2}^{3n/4-1}[Hx^{n-2}]_i l_{n-1-i}+\sum_{i=3n/4}^{n-1}[Hx^{n-2}]_i l_{n-1-i}+m_1)$$
$$+\ldots.+x^{n-1}(\sum_{i=0}^{n/4-1}[H]_i l_{n-1-i}+\sum_{i=n/4}^{n/2-1}[H]_i l_{n-1-i}$$
$$+\sum_{i=n/2}^{3n/4-1}[H]_i l_{n-1-i}+\sum_{i=3n/4}^{n-1}[H]_i l_{n-1-i}+m_{n-1}) \qquad (9)$$

The proposed Eqns. 8 and 9 are obtained by splitting the coefficients of conventional Eqn. 6 into two and four segments respectively. However, the coefficients of Eqn. 6 can be decomposed to any number of even segments as shown in Algorithm 1. The number of parallel processing segments are determined by variable d where d is an even number as shown in line 3 of Algorithm 1. In our work, we consider $d = 2$ and $d = 4$ for splitting the coefficients of polynomial multiplication into two and four segments as illustrated by Eqns. 8 and 9 respectively.

To illustrate the functionality for performing the arithmetic operation $W = HL\ mod(x^n + 1) + M$, we consider a sample value of $n = 8$. Based on the conventional Eqn. 5, we obtain Eqn.10 by replacing n with 8 as follows:

$$HL\ mod(x^8+1) = (h_0.l_0 - h_1.l_7 - h_2.l_6 - \ldots. - h_7.l_1)$$
$$+x(h_0.l_1 + h_1.l_0 - h_2.l_7 - \ldots. - h_7.l_2)$$
$$+x^2(h_0.l_2 + h_1.l_1 + h_2.l_0 - h_3.l_7 - \ldots. - h_7.l_3)$$
$$+\ldots\ldots x^7(h_0.l_7 + h_1.l_6 + h_2.l_5 + \ldots.. + h_7.l_0) \quad (10)$$

The coefficients of polynomial M are added to the corresponding coefficients of Eqn. 10 to get the desired polynomial W. The coefficients of polynomial W ($w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7$) are represented in Table 1.

The coefficients in Eqn. 10 are splitted into two segments considering $d = 2$ in line 3 of Algorithm 1. The coefficient w_0 is splitted into two segments following Eqn. 8 for performing polynomial multiplication and later the coefficient m_0 is added as shown below:

$$w_0 = \sum_{i=0}^{3}[Hx^7]_i l_{7-i} + \sum_{i=4}^{7}[Hx^7]_i l_{7-i} + m_0 \qquad (11)$$

As per Eqn. 11 the first segment is $(h_0.l_0 - h_7.l_1 - h_6.l_2 - h_5.l_3)$ and the second segment is $(-h_4.l_4 - h_3.l_5 - h_2.l_6 - h_1.l_7)$. These two segments are executed in four parallel computations given as follows: $h_0.l_0$, $-h_4.l_4$ in the first computation, $-h_7.l_1$, $-h_3.l_5$ in the second computation, $-h_6.l_2$, $-h_2.l_6$ in the third computation and $-h_5.l_3$, $-h_1.l_7$ in the fourth computation. The result of all the four computations is added to the coefficient m_0. Similarly, the remaining coefficients of W are obtained as shown in Fig. 1.

The same concept can be extended to split the coefficient w_0 into four segments following Eqn. 9 as shown below:

$$w_0 = \sum_{i=0}^{1}[Hx^7]_i l_{7-i} + \sum_{i=2}^{3}[Hx^7]_i l_{7-i}$$

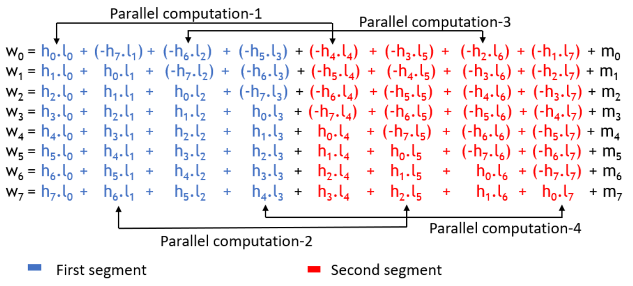
Parallel computation-1
Parallel computation-3
w0 = h0.l0 + (-h7.l1) + (-h6.l2) + (-h5.l3) + (-h4.l4) + (-h3.l5) + (-h2.l6) + (-h1.l7) + m0
w1 = h1.l0 + h0.l1 + (-h7.l2) + (-h6.l3) + (-h5.l4) + (-h4.l5) + (-h3.l6) + (-h2.l7) + m1
w2 = h2.l0 + h1.l1 + h0.l2 + (-h7.l3) + (-h6.l4) + (-h5.l5) + (-h4.l6) + (-h3.l7) + m2
w3 = h3.l0 + h2.l1 + h1.l2 + h0.l3 + (-h7.l4) + (-h6.l5) + (-h5.l6) + (-h4.l7) + m3
w4 = h4.l0 + h3.l1 + h2.l2 + h1.l3 + h0.l4 + (-h7.l5) + (-h6.l6) + (-h5.l7) + m4
w5 = h5.l0 + h4.l1 + h3.l2 + h2.l3 + h1.l4 + h0.l5 + (-h7.l6) + (-h6.l7) + m5
w6 = h6.l0 + h5.l1 + h4.l2 + h3.l3 + h2.l4 + h1.l5 + h0.l6 + (-h7.l7) + m6
w7 = h7.l0 + h6.l1 + h5.l2 + h4.l3 + h3.l4 + h2.l5 + h1.l6 + h0.l7 + m7
Parallel computation-2
Parallel computation-4
First segment
Second segment

FIGURE 1. Representation of polynomial multiplication equation into two segments while considering a sample value of $n = 8$.

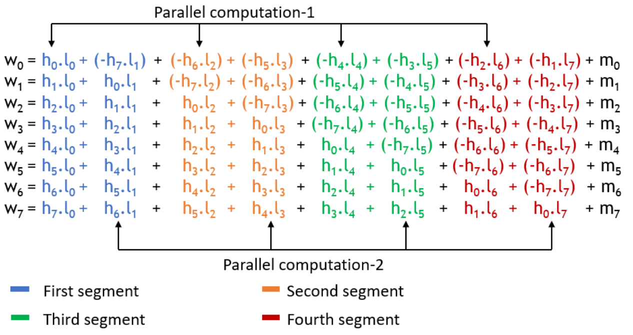
Parallel computation-1
w0 = h0.l0 + (-h7.l1) + (-h6.l2) + (-h5.l3) + (-h4.l4) + (-h3.l5) + (-h2.l6) + (-h1.l7) + m0
w1 = h1.l0 + h0.l1 + (-h7.l2) + (-h6.l3) + (-h5.l4) + (-h4.l5) + (-h3.l6) + (-h2.l7) + m1
w2 = h2.l0 + h1.l1 + h0.l2 + (-h7.l3) + (-h6.l4) + (-h5.l5) + (-h4.l6) + (-h3.l7) + m2
w3 = h3.l0 + h2.l1 + h1.l2 + h0.l3 + (-h7.l4) + (-h6.l5) + (-h5.l6) + (-h4.l7) + m3
w4 = h4.l0 + h3.l1 + h2.l2 + h1.l3 + h0.l4 + (-h7.l5) + (-h6.l6) + (-h5.l7) + m4
w5 = h5.l0 + h4.l1 + h3.l2 + h2.l3 + h1.l4 + h0.l5 + (-h7.l6) + (-h6.l7) + m5
w6 = h6.l0 + h5.l1 + h4.l2 + h3.l3 + h2.l4 + h1.l5 + h0.l6 + (-h7.l7) + m6
w7 = h7.l0 + h6.l1 + h5.l2 + h4.l3 + h3.l4 + h2.l5 + h1.l6 + h0.l7 + m7
Parallel computation-2
First segment
Second segment
Third segment
Fourth segment

FIGURE 2. Representation of polynomial multiplication equation into four segments while considering a sample value of $n = 8$.

$$+\sum_{i=4}^{5}[Hx^7]_i l_{7-i} + \sum_{i=6}^{7}[Hx^7]_i l_{7-i} + m_0 \qquad (12)$$

As per Eqn. 12 the four segments are $(h_0.l_0 - h_7.l_1)$, $(-h_6.l_2 - h_5.l_3)$, $(-h_4.l_4 - h_3.l_5)$ and $(-h_2.l_6 - h_1.l_7)$. These four segments are executed in two parallel computations given as follows: $h_0.l_0$, $-h_6.l_2$, $-h_4.l_4$, $-h_2.l_6$ in the first computation and $-h_7.l_1$, $-h_5.l_3$, $-h_3.l_5$, $-h_1.l_7$ in the second computation. The result of these computations is added to the coefficient m_0. Similarly, the remaining coefficients of W are obtained as shown in Fig. 2. In Eqns. 11 and 12, we have considered decomposing polynomial multiplication into two and four segments considering a sample value of $n = 8$ to understand the computations clearly. The same concept can be extended to any value of n and any level of parallelism.

Based on the splitting of polynomial equation into two and four segments, we propose two efficient hardware architectures namely Dual-LFSR (DL) and Quad-LFSR (QL) structures respectively. We follow the set of parameters $(n, q) = (256, 256)$ and $(512, 256)$ in our work following the existing works. These architectures will be discussed in Section - IV.

IV. PROPOSED ARCHITECTURES

The algorithmic formulation for performing arithmetic operation $HL + M$ over a polynomial ring $x^n + 1$ is described in Section III. In this section, we use these computational details to propose efficient architectures employing Dual-LFSR (DL) and Quad-LFSR (QL) structures used in BRLWE-based scheme. The proposed architectures

TABLE 1. Computation of $HL\ mod(x^8+1)+M$.

w_0	$h_0.l_0 - h_7.l_1 - h_6.l_2 - h_5.l_3 - h_4.l_4 - h_3.l_5 - h_2.l_6 - h_1.l_7 + m_0$
w_1	$h_1.l_0 + h_0.l_1 - h_7.l_2 - h_6.l_3 - h_5.l_4 - h_4.l_5 - h_3.l_6 - h_2.l_7 + m_1$
w_2	$h_2.l_0 + h_1.l_1 + h_0.l_2 - h_7.l_3 - h_6.l_4 - h_5.l_5 - h_4.l_6 - h_3.l_7 + m_2$
w_3	$h_3.l_0 + h_2.l_1 + h_1.l_2 + h_0.l_3 - h_7.l_4 - h_6.l_5 - h_5.l_6 - h_4.l_7 + m_3$
w_4	$h_4.l_0 + h_3.l_1 + h_2.l_2 + h_1.l_3 + h_0.l_4 - h_7.l_5 - h_6.l_6 - h_5.l_7 + m_4$
w_5	$h_5.l_0 + h_4.l_1 + h_3.l_2 + h_2.l_3 + h_1.l_4 + h_0.l_5 - h_7.l_6 - h_6.l_7 + m_5$
w_6	$h_6.l_0 + h_5.l_1 + h_4.l_2 + h_3.l_3 + h_2.l_4 + h_1.l_5 + h_0.l_6 - h_7.l_7 + m_6$
w_7	$h_7.l_0 + h_6.l_1 + h_5.l_2 + h_4.l_3 + h_3.l_4 + h_2.l_5 + h_1.l_6 + h_0.l_7 + m_7$

Computational values for a sample value of n = 8

undergo parallel execution of LFSR structures to reduce the computational time. Architecture I of [3] is modified considerably by providing coefficients of integer polynomial as parallel inputs and coefficients of binary polynomial as serial inputs to Dual and Quad LFSR structures. However, due to doubling and quadrupling the LFSR structures, the input and output processing setup differs in our work. In addition, we eliminate multiplexers in our architecture and the coefficients of integer polynomial are added at the later stage with carry save adder. We do not require any precomputation for performing $2's$ complement of input coefficients or additional clock cycles required for loading these precomputed $2's$ complement input coefficients for addition operation as compared to [3]. Though, this work yields to an increase in the hardware resources due to doubling and quadrupling the structures, it has less computational time compared to the other works. The reduction in computational time improves other performance metrics such as delay, Area-Delay Product (ADP), Power-Delay Product (PDP), throughput and efficiency compared to the existing works.

The modular addition in the proposed structure does not require any extra modular reduction unit. This can be illustrated as follows: We consider the parameter for finite ring modulo $q = 256$ and bit width $k = log_2 q = 8$. While performing modular addition operation, the maximum value of result is $2^k - 1 = 2^8 - 1 = 255$, and any value beyond 255 would be automatically truncated as it exceeds the bit size which is 8. Hence, though there is no reduction unit, the result is obtained within the finite ring modulo $q = 256$.

A finite state machine (FSM) controller is used for the proposed structures to generate control signals required for performing the input loading, processing and output delivery of the coefficients as shown in Fig. 3. It has three states s_0, s_1 and s_2 with internally generated controlled signals *load*, *start* and *done*. The state s_0 is an initial state with no input coefficients. When the control signal *load* is enabled, it moves to state s_1 where the coefficients of polynomials H are loaded. As long as it is in state s_1, the control signal *done* $= 0$. After the input coefficients of polynomial H are loaded, the *start* signal is enabled, and it moves to the next state s_2. During this state, the coefficients of polynomial L are loaded and the entire computations for polynomial multiplication are performed. Once, the results of polynomial multiplication are obtained, *done* signal is enabled and moves to the next state s_0. By this process, the results are carefully retrieved by eliminating unwanted operations. Further, the results obtained from polynomial multiplication are added to the coefficients of polynomial M and decryption operation is performed.

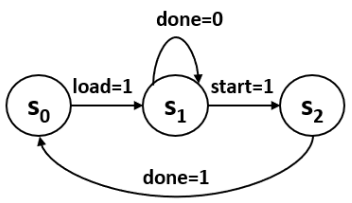

FIGURE 3. FSM controller.

The functionality of the DL and QL structures would be discussed in this section.

A. DUAL LFSR (DL) STRUCTURE

In this structure we perform the functionality of $HL\ mod(x^n + 1) + M$ using two LFSR structures. The computations can be decomposed into two segments as shown in Fig. 1 while considering a sample value of $n = 8$. These two segments can be executed simultaneously by using two LFSR structures and added with the coefficients of integer polynomial M. A sample value of $n = 8$ has been considered to understand the computations clearly and the same procedure can be extended for any value of n.

Following these computations, we propose an efficient hardware architecture employing a dual linear feedback structure (DL structure) which performs parallel executions in the two LFSR structures to reduce the computational time. The top-level design and the detailed architecture are as shown in Figs. 4 and 5 respectively [4]. The architecture comprises:

(i) a serial-in-parallel-out shift register where the coefficients of integer polynomial H are loaded serially into the shift register and delivered parallelly to LFSR structure-1 and LFSR structure-2 respectively.

(ii) two serial-in-serial-out shift registers to load the binary polynomial L. The coefficients of binary polynomial L are split into two halves where the first half of the coefficients $l_0, l_1, l_2, \ldots, l_{n/2-1}$ are loaded through a shift register and the remaining half of the coefficients $l_{n/2}, l_{n/2+1}, l_{n/2+2}, \ldots .l_{n-1}$ are loaded through another shift register.

(iii) two LFSR structures where each LFSR structure comprise a series of AND cells, registers, and adders. The functionality for performing the arithmetic operation $H.L + M$ is obtained by decomposing the polynomial multiplication operation into two segments. These segments are executed parallelly by LFSR structure-1 and LFSR structure-2 respectively.

(iv) a sign control unit consisting of a multiplexer, incrementer and inverter in the feedback section of LFSR structures. As we consider the polynomial ring $x^n + 1$, polynomial multiplication equations are obtained by considering

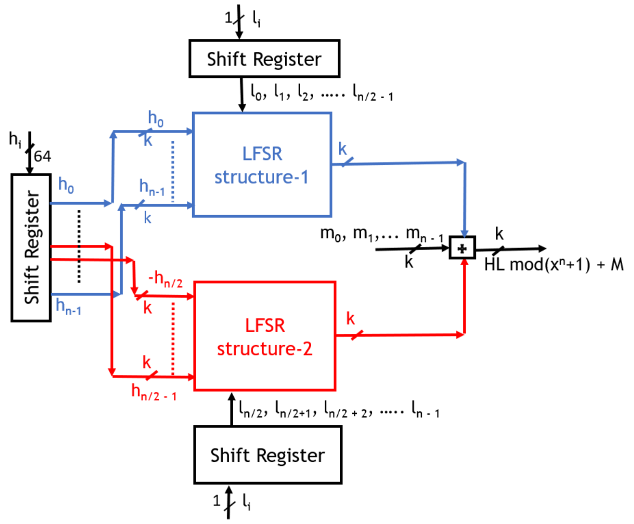

FIGURE 4. **Top level description for the proposed Dual-LFSR (DL) structure performing the functionality $H.L\ mod(x^n + 1) + M$.**

$x^n \equiv -1$. This results in a negative sign as observed from the Eqn. 8 and is obtained by performing $2's$ complement of that particular number and selected by multiplexers Mux 1 and Mux 2 respectively.

(v) a shift register to load the coefficients of integer polynomial M required for addition and

(vi) a carry save adder (CSA) to add the resultant of LFSR structures to the coefficients of integer polynomial M.

The working procedure is discussed in the following steps:

1) LOADING COEFFICIENTS OF INTEGER POLYNOMIAL H

The coefficients of integer polynomial H are loaded to the two LFSR structures for enabling parallel computations. As per line 3 of Algorithm 1, we obtain the coefficients of H in two segments considering $d = 2$. The first segment contains H in the order $h_0, h_1, \ldots., h_{n-1}$, whereas the second segment contains the order $-h_{n/2}, -h_{1+n/2}, \ldots, h_{n/2-1}$. Based on this grouping, we provide parallel inputs of H as shown in Fig. 5. The input coefficients $h_0, h_1, \ldots., h_{n-1}$ are fed to LFSR structure-1 and $-h_{n/2}, -h_{1+n/2}, \ldots, h_{n/2-1}$ are fed to LFSR structure-2 respectively. The negative values of the coefficients h_i (*i.e.*, $-h_{n/2}, -h_{1+n/2}, \ldots, -h_{n-1}$) are obtained by inverting the values and adding 1 to it. During this process of loading parallel coefficients, we intend to load only the coefficients of polynomial H by making the other input of AND gate 0 for preventing unwanted operations.

2) PERFORMING COMPUTATION HL+M

During this process, the coefficients of binary polynomial L are loaded into the two LFSR structures. We obtain the coefficients of L in two segments such as $l_0, l_1, l_2, \ldots.., l_{n/2-1}$ and $l_{n/2}, l_{n/2+1}, l_{n/2+2}, \ldots.., l_{n-1}$ as per line 3 of Algorithm 1. The coefficients $l_0, l_1, l_2, \ldots.., l_{n/2-1}$ are loaded into LFSR structure-1 and $l_{n/2}, l_{n/2+1}, l_{n/2+2}, \ldots.., l_{n-1}$ are loaded into LFSR structure-2 serially. As the two LFSR structures are identical, we have the same functionality in these structures. However, though the structures are identical, they are loaded with different set of inputs. Hence, we find the computational details for each LFSR structure separately.

The computational details for LFSR structure-1 and LFSR structure-2 are as shown in Table 2 and Table 3 respectively considering a sample value of $n = 8$. Since $n = 8$, we consider only eight registers $r_0, r_1, r_2, r_3, r_4, r_5, r_6$ and r_7 in LFSR structure-1, and registers $rr_0, rr_1, rr_2, rr_3, rr_4, rr_5, rr_6$ and rr_7 in LFSR structure-2. Following line 3 of Algorithm 1, we give the inputs $h_0, h_1, h_2, h_3, h_4, h_5, h_6$ and h_7 having the same sign to LFSR structure-1. These coefficients are multiplied with the coefficients of binary polynomials l_3, l_2, l_1, l_0 at each clock cycle. At clock cycle t_1, we have multiplication of the coefficients given by $h_0.l_3, h_1.l_3, h_2.l_3, h_3.l_3, h_4.l_3, h_5.l_3, h_6.l_3, h_7.l_3$ held at registers $r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7$ respectively as shown in Table 2. From the next clock cycle i.e., t_2 the output of last register is fed back to the left most register in $2's$ complement form (for sign reversal) following Eqn. 11. This is indicated by $\widehat{h_7.l_3}$ at the clock cycle t_2, $\widehat{h_6.l_3} + \widehat{h_7.l_2}$ at clock cycle t_3, and $\widehat{h_5.l_3} + \widehat{h_6.l_2} + \widehat{h_7.l_1}$ at clock cycle t_4. Considering $n = 8$, we require $n/2 = 4$ clock cycles for performing our computation by loading l_3, l_2, l_1, l_0 at each clock cycle. Since the two LFSR structures are identical the same functionality is observed in LFSR structure-2. Here, we give the inputs $-h_4, -h_5, -h_6, -h_7, h_0, h_1, h_2$ and h_3 consisting of both positive and negative signs. We represent negative sign coefficients with $2's$ complement form of that particular coefficients as shown in Table 3. So, we give the inputs $\widehat{h_4}, \widehat{h_5}, \widehat{h_6}, \widehat{h_7}, h_0, h_1, h_2$ and h_3 to LFSR structure-2. Now, we have the multiplication of these coefficients with the coefficients of binary polynomials l_7, l_6, l_5, l_4. Similar to the operation in LFSR structure-1, we have multiplication and addition of the coefficients as shown in Table 3. At the end of four clock cycles, the required computations are performed in both LFSR structures. After four clock cycles, the values in the registers of both LFSR structures are added along with the coefficients of integer polynomial M to get the result $HL\ mod(x^8 + 1) + M$.

Though the above explanation is carried for $n = 8$ to understand the computational process, the same process holds for any value of n. The parallel executions in the two LFSR structures reduce the computational time to $n/2$ clock cycles. After performing the executions in the two LFSR structures simultaneously, the result can be added with the coefficients of integer polynomial M. The proposed architecture comprises both 2-input and 3-input adders as shown in Fig. 5. We use conventional ripple carry adders and carry save adders for 2 input adders and 3 input adders respectively for implementing our design.

3) RETRIEVAL OF OUTPUT POLYNOMIAL

In this step, the output of polynomial multiplication is carefully retrieved by preventing unwanted operations. It is performed by passing $0's$ through output of AND gates and selecting $'0'$ input in mux-1 and mux-2 for both

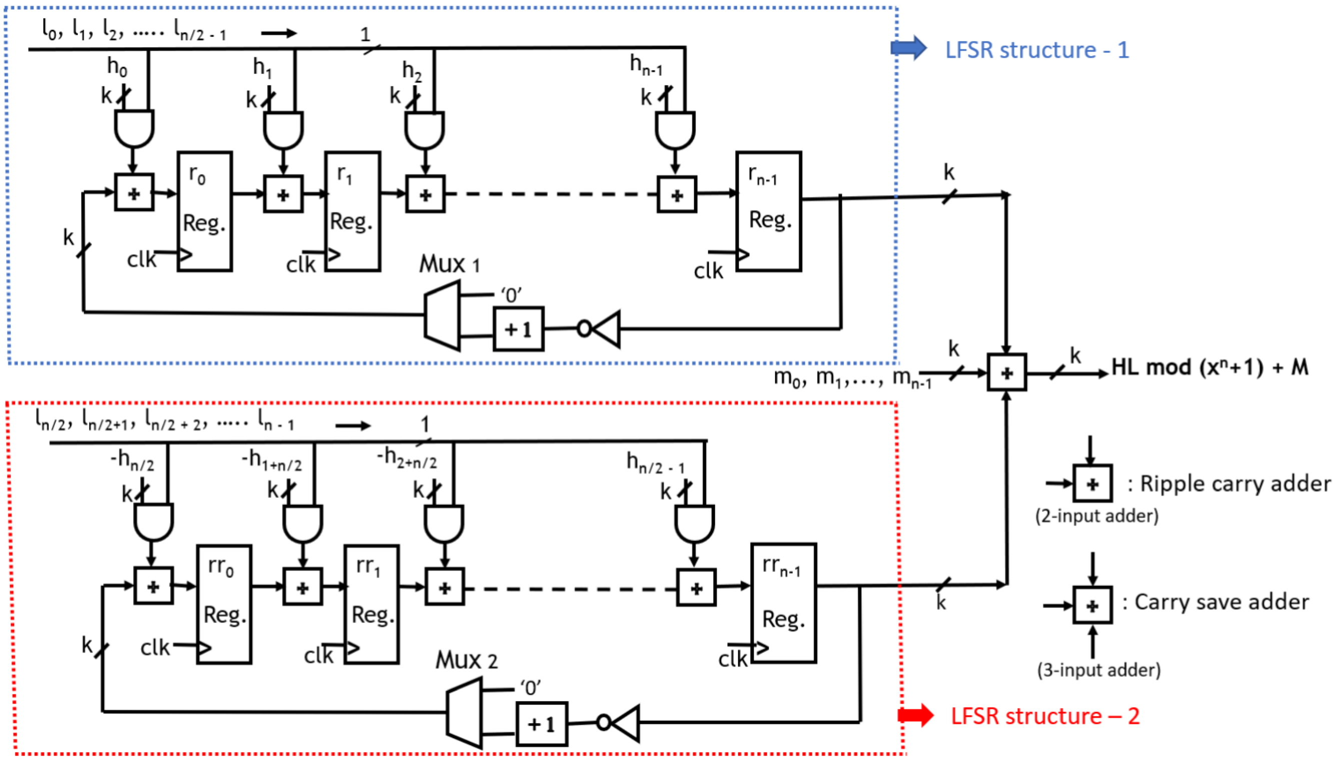

FIGURE 5. Hardware architecture for arithmetic operation $W = H.L\ mod(x^n + 1) + M$ based on BRLWE scheme using parallel executions in Dual LFSR structure to deliver the output coefficients ($w_{n-1}, w_{n-2}, \ldots, w_0$) serially.

LFSR structures. Thereby only $0's$ would be passed through registers and unwanted operations can be prevented. The previously obtained values are retrieved serially for obtaining the functionality $HL\ \ mod(x^{n+1}) + M$. We perform the decryption operation by tying the two MSB bits to an XOR gate following the works [8], [9], [12], and [14].

Example: The proposed Dual LFSR structure can be explained with a suitable example by considering $n = 8$. The value of $n = 8$ is considered to ease the understanding of architecture. Consider an integer polynomial

$$H = 2 + x + 3.x^2 + x^3 + 7.x^4 + x^5 + 6.x^6 + 2.x^7$$

The coefficients of polynomial H are represented as follows

$$H = \{h_0, h_1, h_2, h_3, h_4, h_5, h_6, h_7\} = \{2, 1, 3, 1, 7, 1, 6, 2\}$$

Let us consider a binary polynomial

$$L = 1 + x^2 + x^3 + x^5 + x^6 + x^7$$

The coefficients of polynomial L are represented as follows

$$L = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7\} = \{1, 0, 1, 1, 0, 1, 1, 1\}$$

Let us consider an integer polynomial

$$M = 1 + 6.x + 7.x^2 + 2.x^3 + 3.x^4 + 4.x^5 + 5.x^6 + 6.x^7$$

The coefficients of polynomial M are represented as follows

$$\begin{aligned} M &= \{m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7\} \\ &= \{1, 6, 7, 2, 3, 4, 5, 6\} \end{aligned}$$

Initially the coefficients of binary polynomial L are split into two groups {1,0,1,1} and {0,1,1,1}. These coefficients are provided as serial inputs to the two LFSR structures LFSR structure-1 and LFSR structure-2 respectively. The coefficients of integer polynomial H are given in parallel to the two LFSR structures following Fig. 5. So, LFSR structure-1 is loaded with coefficients {2,1,3,1,7,1,6,2} and LFSR structure-2 is loaded with coefficients {−7,−1,−6,−2,2,1,3,1} i.e., {1,7,2,6,2,1,3,1} parallelly. All the register values are initially set to 0.

At clock cycle-1, the parallel coefficients of LFSR structure-1 {2,1,3,1,7,1,6,2} are multiplied with serial input $l_3 = 1$ through AND gates to produce the register values $r_0, r_1, r_2, r_3, r_4, r_5, r_6$ and r_7 as shown in Table 4. At clock cycle-2, the value of the last register r_7 is fedback in $2's$ complement form to the first register and added to the result of the AND gate (which is the multiplication of parallel coefficients with serial input $l_2 = 1$). The same process continues in clock cycles-3 and 4 with serial input $l_1 = 0$ at $l_0 = 1$ respectively. At the end of clock cycle-4, the expected result from LFSR structure-1 is obtained and we prevent further operations by passing 0 through mux-1.

The same sequence of steps is carried in LFSR structure-2 also as both the LFSR structures are identical. At clock cycle-1, the parallel coefficients of LFSR structure-2 {1,7,2,6,2,1,3,1} are multiplied with serial input $l_7 = 1$ and produce the register values $rr_0, rr_1, rr_2, rr_3, rr_4, rr_5, rr_6$ and rr_7. At the clock cycle-2 the register value rr_7 is fed back in $2's$ complement form to the first register and added to the

TABLE 2. Computational details for LFSR structure - 1.

Clk.	Inp.	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7
t_1	l_3	$h_0.l_3$	$h_1.l_3$	$h_2.l_3$	$h_3.l_3$	$h_4.l_3$	$h_5.l_3$	$h_6.l_3$	$h_7.l_3$
t_2	l_2	$\widehat{h_7.l_3}+ h_0.l_2$	$h_0.l_3 + h_1.l_2$	$h_1.l_3 + h_2.l_2$	$h_2.l_3 + h_3.l_2$	$h_3.l_3 + h_4.l_2$	$h_4.l_3 + h_5.l_2$	$h_5.l_3 + h_6.l_2$	$h_6.l_3 + h_7.l_2$
t_3	l_1	$\widehat{h_6.l_3}+ \widehat{h_7.l_2}$ $+ h_0.l_1$	$\widehat{h_7.l_3}+ h_0.l_2$ $+ h_1.l_1$	$h_0.l_3+ h_1.l_2$ $+ h_2.l_1$	$h_1.l_3+ h_2.l_2$ $+ h_3.l_1$	$h_2.l_3+ h_3.l_2$ $+ h_4.l_1$	$h_3.l_3+ h_4.l_2$ $+ h_5.l_1$	$h_4.l_3+ h_5.l_2$ $+ h_6.l_1$	$h_5.l_3+ h_6.l_2$ $+ h_7.l_1$
t_4	l_0	$\widehat{h_5.l_3}+ \widehat{h_6.l_2}$ $+ \widehat{h_7.l_1}$ $+h_0.l_0$	$\widehat{h_6.l_3}+ \widehat{h_7.l_2}$ $+ h_0.l_1$ $+h_1.l_0$	$\widehat{h_7.l_3}+ h_0.l_2$ $+ h_1.l_1$ $+ h_2.l_0$	$h_0.l_3+ h_1.l_2$ $+ h_2.l_1$ $+ h_3.l_0$	$h_1.l_3+ h_2.l_2$ $+ h_3.l_1$ $+ h_4.l_0$	$h_2.l_3+ h_3.l_2$ $+ h_4.l_1$ $+ h_5.l_0$	$h_3.l_3+ h_4.l_2$ $+ h_5.l_1$ $+ h_6.l_0$	$h_4.l_3+ h_5.l_2$ $+ h_6.l_1$ $+ h_7.l_0$

Register values for LFSR structure 1 for a sample value of $n = 8$.

TABLE 3. Computational details for LFSR structure - 2.

Clk.	Inp.	rr_0	rr_1	rr_2	rr_3	rr_4	rr_5	rr_6	rr_7
t_1	l_7	$\widehat{h_4.l_7}$	$\widehat{h_5.l_7}$	$\widehat{h_6.l_7}$	$\widehat{h_7.l_7}$	$h_0.l_7$	$h_1.l_7$	$h_2.l_7$	$h_3.l_7$
t_2	l_6	$\widehat{h_3.l_7} + \widehat{h_4.l_6}$	$\widehat{h_4.l_7} + \widehat{h_5.l_6}$	$\widehat{h_5.l_7} + \widehat{h_6.l_6}$	$\widehat{h_6.l_7} + \widehat{h_7.l_6}$	$\widehat{h_7.l_7} + h_0.l_6$	$h_0.l_7 + h_1.l_6$	$h_1.l_7 + h_2.l_6$	$h_2.l_7 + h_3.l_6$
t_3	l_5	$\widehat{h_2.l_7}+ \widehat{h_3.l_6}$ $+ \widehat{h_4.l_5}$	$\widehat{h_3.l_7} + \widehat{h_4.l_6}$ $+ \widehat{h_5.l_5}$	$\widehat{h_4.l_7} + \widehat{h_5.l_6}$ $+ \widehat{h_6.l_5}$	$\widehat{h_5.l_7}+ \widehat{h_6.l_6}$ $+ \widehat{h_7.l_5}$	$\widehat{h_6.l_7} + \widehat{h_7.l_6}$ $+ h_0.l_5$	$\widehat{h_7.l_7}+ h_0.l_6$ $+ h_1.l_5$	$h_0.l_7+ h_1.l_6$ $+ h_2.l_5$	$h_1.l_7+ h_2.l_6$ $+ h_3.l_5$
t_4	l_4	$\widehat{h_1.l_7}+ \widehat{h_2.l_6}$ $+ \widehat{h_3.l_5}$ $+ \widehat{h_4.l_4}$	$\widehat{h_2.l_7}+ \widehat{h_3.l_6}$ $+\widehat{h_4.l_5}$ $+ \widehat{h_5.l_4}$	$\widehat{h_3.l_7} + \widehat{h_4.l_6}$ $+ \widehat{h_5.l_5}$ $+ \widehat{h_6.l_4}$	$\widehat{h_4.l_7} + \widehat{h_5.l_6}$ $+ \widehat{h_6.l_5}$ $+ \widehat{h_7.l_4}$	$\widehat{h_5.l_7} + \widehat{h_6.l_6}$ $+ \widehat{h_7.l_5}$ $+ h_0.l_4$	$\widehat{h_6.l_7}+ \widehat{h_7.l_6}$ $+ h_0.l_5$ $+ h_1.l_4$	$\widehat{h_7.l_7}+ h_0.l_6$ $+ h_1.l_5$ $+ h_2.l_4$	$h_0.l_7+ h_1.l_6$ $+ h_2.l_5$ $+ h_3.l_4$

Register values for LFSR structure 2 for a sample value of $n = 8$.

result of AND gate (which is the result of the multiplication of parallel coefficients to serial input $l_6 = 1$). The same operation continues in clock cycles-3 and 4 where we have the serial input $l_5 = 1$ and $l_4 = 0$ respectively. At the end of clock cycle-4, the expected result from LFSR structure-2 is obtained and we prevent further operations by passing 0 through mux-2. The register values at each clock cycle for LFSR structure-1 and LFSR structure-2 are computed and listed in Table 4.

At clock cycle-4 we add the results obtained in LFSR structure-1 {3,1,3,4,3,4,5,2} and LFSR structure-2 {3,5,7,2,7,2,1,6} to obtain the coefficients of polynomial multiplication {6,6,2,6,2,7,7,0}. This is further extended by adding coefficients of integer polynomial M {1,6,7,2,3,4,5,6} to get the desired result {7,4,1,0,5,3,4,6} for the arithmetic operation $H.L\ mod(x^8 + 1) + M$.

B. QUAD LFSR (QL) STRUCTURE

The same concept of DL structure is extended to QL structure by performing the parallel executions using four LFSR structures. The computations can be decomposed into four segments as shown in Fig. 2 while considering a sample value of $n = 8$. These four segments can be executed simultaneously by using four LFSR structures and added with the coefficients of integer polynomial M. A sample value of $n = 8$ has been considered to understand the computations clearly and the same procedure can be extended for any value of n. Due to these parallel executions, the computational time required to obtain the desired functionality is greatly reduced.

The top level architecture for the proposed QL structure is shown in Fig. 6. It uses four identical LFSR structures employing parallel executions to reduce latency. The architecture comprises (i) a serial-in-parallel-out shift register to load the coefficients of integer polynomial H parallelly, (ii) four serial-in-serial-out shift registers to load the binary polynomial L, (iii) four LFSR structures performing parallel executions, (iv) a sign control unit, (v) a shift register to load the coefficients of integer polynomial M and (vi) adders to add the resultant of LFSR structures and the coefficients of integer polynomial M. Similar to the DL structure, we do not have a separate modular reduction unit. We consider $q = 256$ and bit size $k = 8$ in our design. Any value exceeding $q - 1$ would exceed bit size and automatically be truncated. The detailed architecture for the proposed QL structure is as shown in Fig. 7.

The working procedure is similar to that of DL structure. The coefficients of integer polynomial H are given in parallel to the LFSR structures. As per line 3 of Algorithm 1, we obtain the coefficients of H in four segments considering $d = 4$. Based on this grouping, we provide the coefficients of H as parallel inputs to the four LFSR structures. The first LFSR structure constitutes the parallel coefficients $h_0, h_1, h_2, \ldots, h_{n-1}$, the second comprises $-h_{3n/4}, -h_{3n/4+1}, -h_{3n/4+2}, \ldots, h_{3n/4-1}$, the third has $-h_{n/2}, -h_{n/2+1}, -h_{n/2+2}, \ldots, h_{n/2-1}$ whereas the fourth structure has $-h_{n/4}, -h_{n/4+1}, -h_{n/4+2}, \ldots, h_{n/4-1}$ respectively. The coefficients of binary polynomial L are splitted into four segments as follows: $l_0, \ldots, l_{n/4-1}$ is the first group, $l_{n/4}, l_{n/4+1}, \ldots, l_{n/2-1}$ is the second group, $l_{n/2}, l_{n/2+1}, \ldots, l_{3n/4-1}$ is the third group and $l_{3n/4}, l_{3n/4+1}, \ldots, l_{n-1}$ is the fourth group. These coefficients are fed to the four LFSR structures serially. As all these LFSR structures are identical, the computation is performed similar to the DL structure. Due to the parallel executions in the four LFSR structures, the computational time is greatly reduced to $n/4$ clock cycles.

TABLE 4. Register values for polynomial multiplication in a Dual LFSR structure considering a sample value of $n = 8$.

Cycle	r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7
1	$2(1) = 2$	$1(1) = 1$	$3(1) = 3$	$1(1) = 1$	$7(1) = 7$	$1(1) = 1$	$6(1) = 6$	$2(1) = 2$
2	$C_2^{'}(2) + 2(1) = 0$	$2 + 1(1) = 3$	$1 + 3(1) = 4$	$3 + 1(1) = 4$	$1 + 7(1) = 0$	$7 + 1(1) = 0$	$1 + 6(1) = 7$	$6 + 2(1) = 0$
3	$C_2^{'}(0) + 2(0) = 0$	$0 + 1(0) = 0$	$3 + 3(0) = 3$	$4 + 1(0) = 4$	$4 + 7(0) = 4$	$0 + 1(0) = 0$	$0 + 6(0) = 0$	$7 + 2(0) = 7$
4	$C_2^{'}(7) + 2(1) = 3$	$0 + 1(1) = 1$	$0 + 3(1) = 3$	$3 + 1(1) = 4$	$4 + 7(1) = 3$	$4 + 1(1) = 5$	$0 + 6(1) = 6$	$0 + 2(1) = 2$
Cycle	rr_0	rr_1	rr_2	rr_3	rr_4	rr_5	rr_6	rr_7
1	$1(1) = 1$	$7(1) = 7$	$2(1) = 2$	$6(1) = 6$	$2(1) = 2$	$1(1) = 1$	$3(1) = 3$	$1(1) = 1$
2	$C_2^{'}(1) + 1(1) = 0$	$1 + 7(1) = 0$	$7 + 2(1) = 1$	$2 + 6(1) = 0$	$6 + 2(1) = 0$	$2 + 1(1) = 3$	$1 + 3(1) = 4$	$3 + 1(1) = 4$
3	$C_2^{'}(4) + 1(1) = 5$	$0 + 7(1) = 7$	$0 + 2(1) = 2$	$1 + 6(1) = 7$	$0 + 2(1) = 2$	$0 + 1(1) = 1$	$3 + 3(1) = 6$	$4 + 1(1) = 5$
4	$C_2^{'}(5) + 1(0) = 3$	$5 + 7(0) = 5$	$7 + 2(0) = 7$	$2 + 6(0) = 2$	$7 + 2(0) = 7$	$2 + 1(0) = 2$	$1 + 3(0) = 1$	$6 + 1(0) = 6$

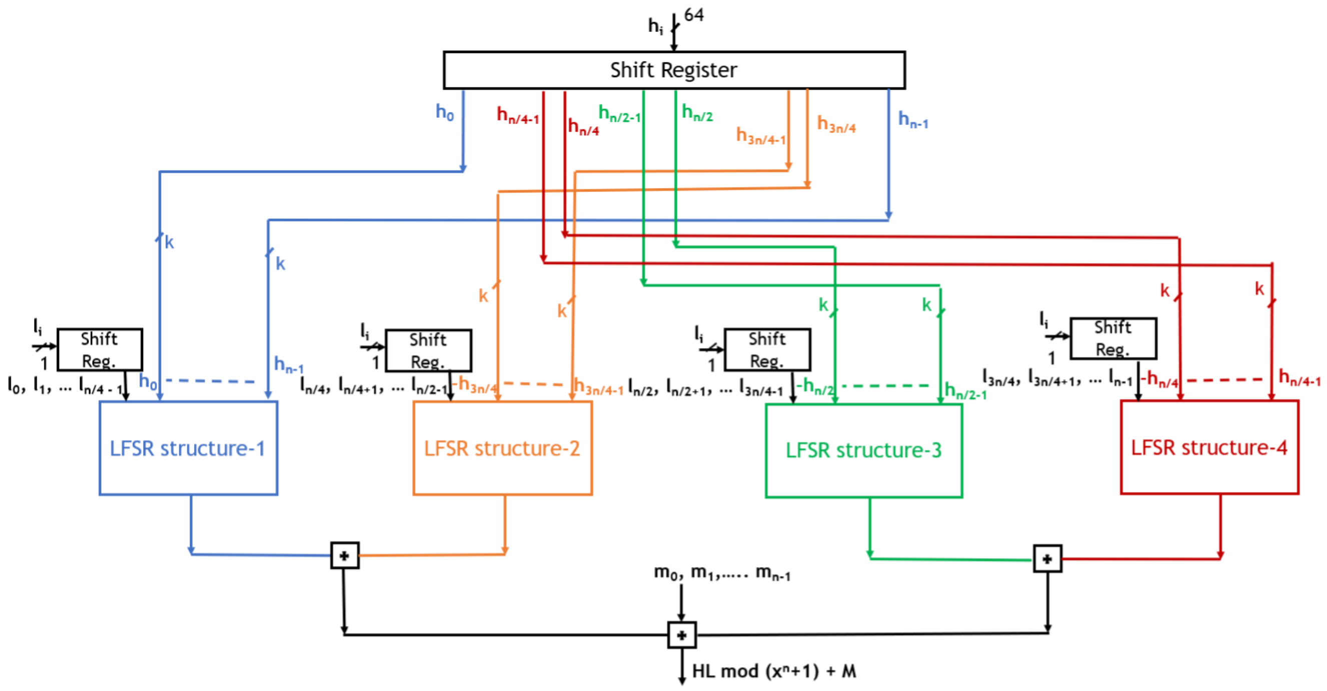

FIGURE 6. Top level description for the proposed Quad-LFSR (QL) structure performing the functionality $H.L \ mod(x^n + 1) + M$.

1) COMPLEXITY OF THE STRUCTURES

The proposed DL and QL structures requires parallel loading of n inputs with bitsize k. Following the existing works, we choose the parameters $n = 256$ and 512 respectively where $k = 8$ bits. So, the number of parallel inputs required when $n = 256$ and $n = 512$ are $256 \times 8 = 1024$ bits and $512 \times 8 = 2048$ bits respectively. Due to the practical limitation of I/O ports, we use a serial-in-parallel-out shift register to load 64 bits each cycle as shown in Fig. 4. This refers to the practical setup of input processing based on the 64-bit word length of the processors. Hence, we require an additional 32 and 64 clock cycles to load the parallel inputs when $n = 256$ and $n = 512$ respectively.

2) COMPUTATION OF LATENCY

The number of LFSR structures can be further increased to decrease the computational time (latency) by performing parallel executions. A mathematical equation is obtained to compute latency based on the number of LFSR structures. It is shown in Eqn. 13 as follows:

$$Latency = N + (n/u) - 1 \tag{13}$$

where N = number of clock cycles required for loading the coefficients of integer polynomial H, n = degree of polynomial, u = number of LFSR's used. We subtract with 1 as one clock cycle would be in common for both loading and computational process.

In our design, we require 32 and 64 clock cycles for loading the coefficients of polynomial H when $n = 256$ and $n = 512$ respectively. Let us consider a case when $n = 256$, if we use DL structure, the latency would be $32 + (256/2) - 1 = 159$ clock cycles. Similarly, with QL structure, the latency would be $32 + (256/4) - 1 = 95$ clock cycles. This shows the decrease in latency with increase in number of LFSR's. Based on these calculations, a graph is drawn as shown in

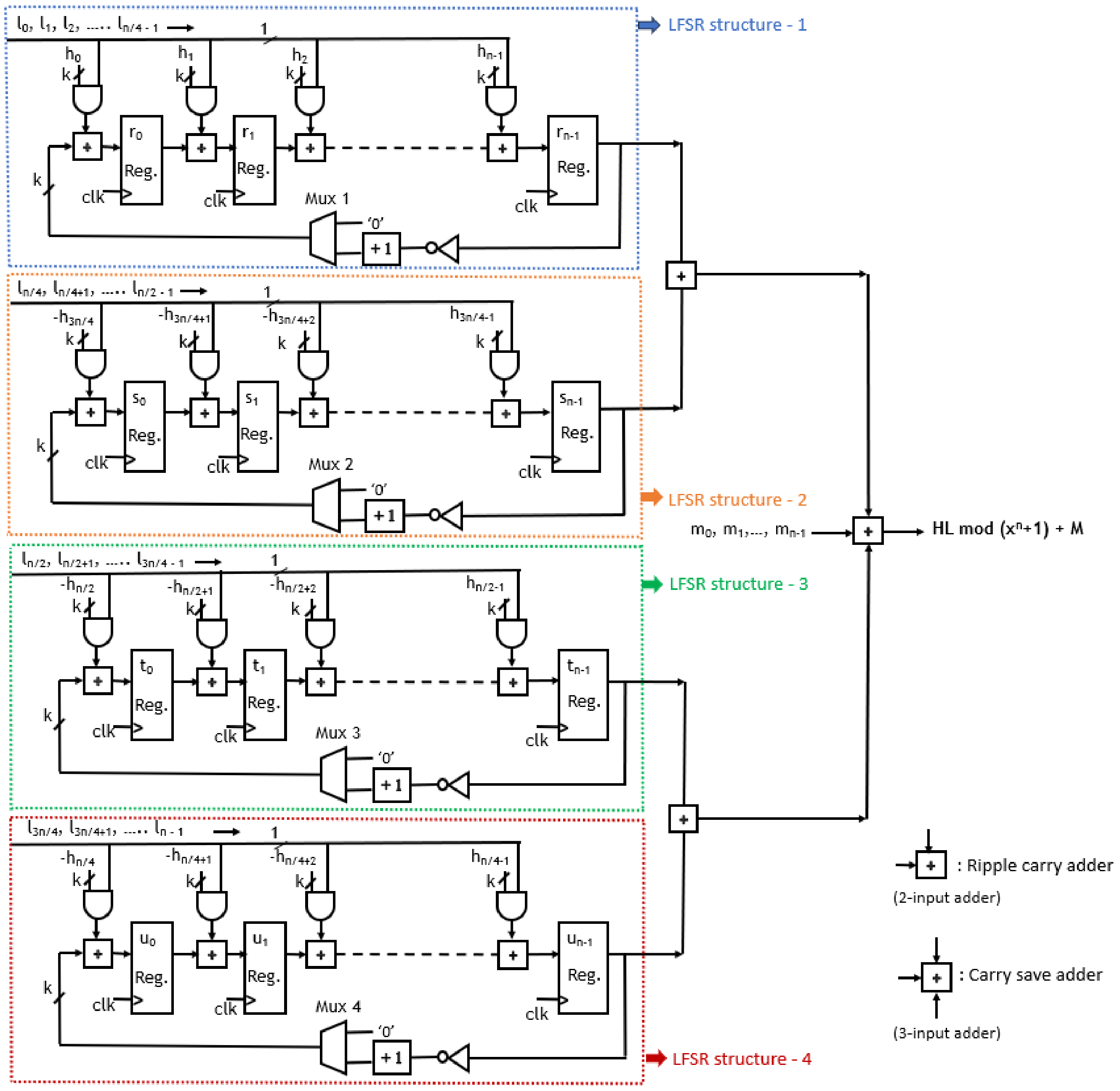

FIGURE 7. Hardware architecture for arithmetic operation $W = H.L \bmod (x^n + 1) + M$ based on BRLWE scheme using parallel executions in Quad LFSR structure to deliver the output coefficients ($w_{n-1}, w_{n-2}, \cdots, w_0$) serially.

Fig. 8 indicating the decrease in latency with an increase in the number of LFSR structures when $n = 256$ and $n = 512$.

3) COMPUTATION OF HARDWARE COMPLEXITY

There would be certainly an increase in hardware resources with an increase in the number of LFSR structures. The hardware complexity is evaluated theoretically based on CMOS logic [30], [31], [32]. Based on the conventional CMOS logic, the number of transistors for the proposed DL and QL structures are calculated considering 6 transistors for 2-input AND, XOR gates and 1-bit 2:1 MUX, 2 transistors for an inverter, 18 transistors for a 1-bit flipflop, 26 transistors for 2-input ripple carry adder. The components required for the proposed structures are listed in Table 5. Assuming a case when $n = 256$ and $k = 8$, the number of transistors required for the proposed DL-structure is calculated as follows: $2 \times 256 \times 8 \times 6 = 24,576$ transistors for AND gate, $2 \times 8 \times 2 = 32$ transistors for inverter, $2 \times 256 \times 8 \times 26 = 106,496$ transistors for 2-input adder, $510 \times 26 \times 8 + 12 \times 8 = 106,176$ transistors for carry-save adder (we require 510 full adders each comprising 26 transistors and 1 half adder comprising 12 transistors), $2 \times 8 \times 6 = 96$ transistors for multiplexer and $2 \times 256 \times 8 \times 18 = 73,728$ transistors for flipflops. We obtain the

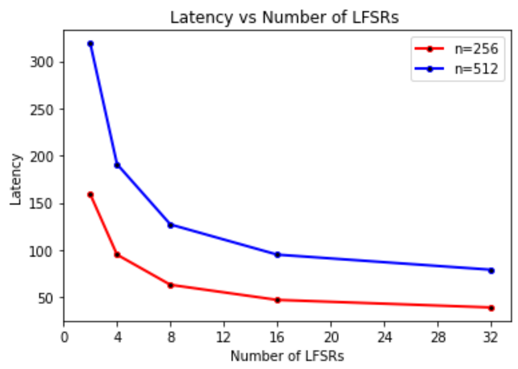

FIGURE 8. Graphical representation showing decrease in latency with increase in number of LFSR's when $n = 256$ and $n = 512$.

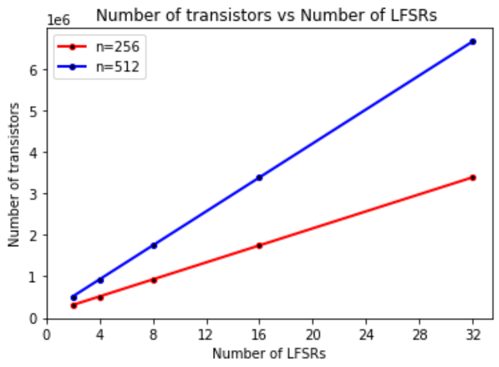

FIGURE 9. Graphical representation showing increase in the number of transistors with increase in number of LFSR's when $n = 256$ and $n = 512$.

transistors for the proposed DL-structure by the summation of transistors obtained through individual components leading to $311,104$ transistors ($24,576+32+106,496+106,176+96+73,728 = 311,104$). Similarly, the number of transistors for the proposed QL-structure can also be calculated as follows: $4 \times 256 \times 8 \times 6 = 49,152$ transistors for AND gate, $4 \times 8 \times 2 = 64$ transistors for inverter, $(4 \times 256 + 2) \times 8 \times 26 = 213,408$ transistors for 2-input adder, $510 \times 26 \times 8 + 12 \times 8 = 106,176$ transistors for carry-save adder(we require 510 full adders each comprising 26 transistors and 1 half adder comprising 12 transistors), $4 \times 8 \times 6 = 192$ transistors for multiplexer and $4 \times 256 \times 8 \times 18 = 147,456$ transistors for flipflops. By summing all the transistors from individual components, we obtain $516,448$ transistors ($49,152 + 64 + 213,408 + 106,176 + 192 + 147,456 = 516,448$). This shows an increase in the number of transistors from $311,104$ to $516,448$ when we increase the number of LFSR's from 2 to 4. On similar lines, we estimate the hardware complexity with an increase in the number of LFSR structures and represent graphically as shown in Fig. 9. Based on these estimates, a suitable number of LFSR structures can be chosen based on the design requirements.

V. RESULTS AND DISCUSSIONS

In this section, the results of the proposed structures are compared with the existing BRLWE/InvBRLWE-based schemes. We compare our work with similar works [3], [9], [12]. As we compare our work with [3] which has proven an improved ADP compared to the other works [8], [9], [11], [12], [14], we do not compare our work with [8], [11], and [14]. However, we compare our results with [9] and [12] as they have similar processing styles compared to our design. We do not compare our work with [6], [13], [25], and [26] as they are compact designs which are suitable for resource-constrained implementations and it would not be a fair comparison. The work in [10] does not have a proper sign control unit, and we have not included it in our comparison. Our work does not discuss anti-SPA countermeasures. Hence, we do not include the work in [24] in our comparisons as it requires additional hardware resources for anti-SPA countermeasures.

The theoretical area-time complexities for the proposed architectures are compared with existing works [3], [9], [12] as shown in Table 5. It includes major components of the architecture such as the number of 2-input AND gates (#AND), inverters (#INV), 2(k bit)-input adders operands, 3(k bit)-input adders operands, 2×1 (k-bit) multiplexers, 1 bit registers or flipflops (#FF), clock cycles required for computation (#Clk). The following illustration shows the number of components required for the proposed DL structure: There are two LFSR structures for the proposed DL structure as shown in Fig.5. There are nk AND gates in each LFSR structure contributing $2nk$ AND gates in the DL structure where n is the degree of polynomial and k is the bitsize. Similarly, there are n ripple carry adders (RCA) in each LFSR structure contributing a total of $2n$ ripple carry adders. A carry save adder (CSA) is required to add the resultant of LFSR structures with the coefficients of polynomial M. As there are k inverters in each LFSR structure, it contributes to $2k$ inverters and there are two k-bit multiplexers (Mux 1 and Mux 2) in the DL structure. This structure also requires double the number of flipflops compared to the works [3], [9], [12] as there is a parallel execution of two LFSR structures. Similarly, the number of components for the proposed QL structure is illustrated as follows: There are four LFSR structures for the proposed QL structure as shown in Fig. 7. There are nk AND gates in each LFSR structure contributing a total of $4nk$ AND gates. It has n ripple carry adders (RCA) in each LFSR structure and an additional two adders are required to perform the addition for the resultant of LFSR structures-1,2 and LFSR structures-3,4 respectively contributing a total of $4n+2$ adders. The resultant of LFSR structures-1,2 and LFSR structures-3,4 is added to the coefficients of polynomial M with a carry save adder (CSA). As there are k inverters and a k-bit multiplexer in each LFSR structure, it contributes a total of $4k$ inverters and four k-bit multiplexers respectively to the proposed QL structure. Further, as this structure comprises parallel execution of four LFSR structures, it requires four times the number of flipflops compared to the works [3], [9], [12]. Table 5 also includes the critical path or combinational path delay of the design. These calculations do not include shift register resources for loading the polynomial coefficients. Based on the theoretical

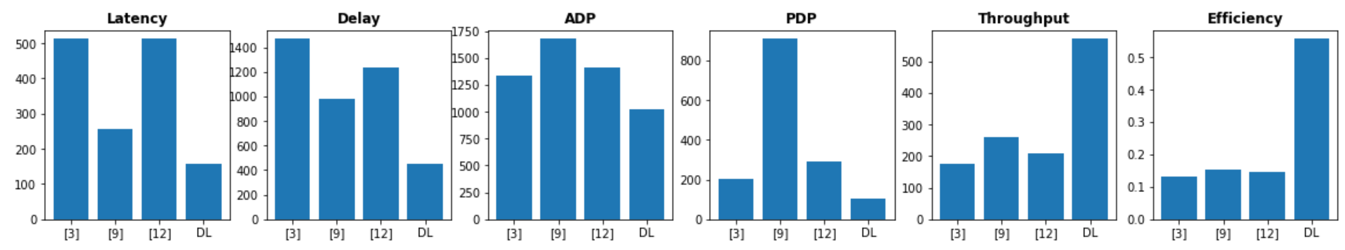

FIGURE 10. Performance comparison of FPGA implementation results for the proposed DL structure with similar structures when $n = 256, q = 256$.

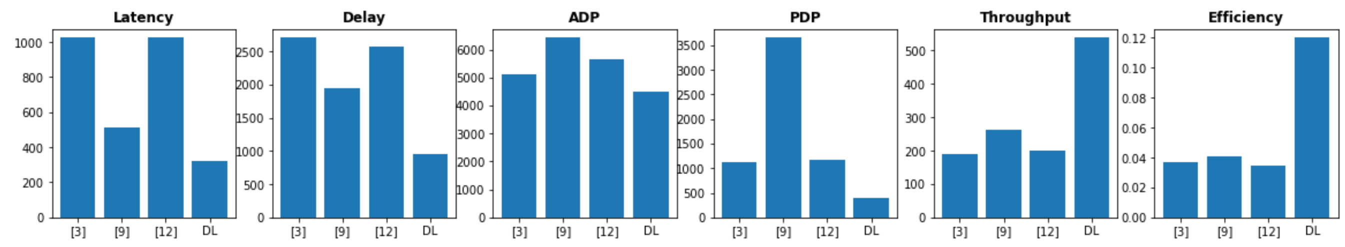

FIGURE 11. Performance comparison of FPGA implementation results for the proposed DL structure with similar structures when $n = 512, q = 256$.

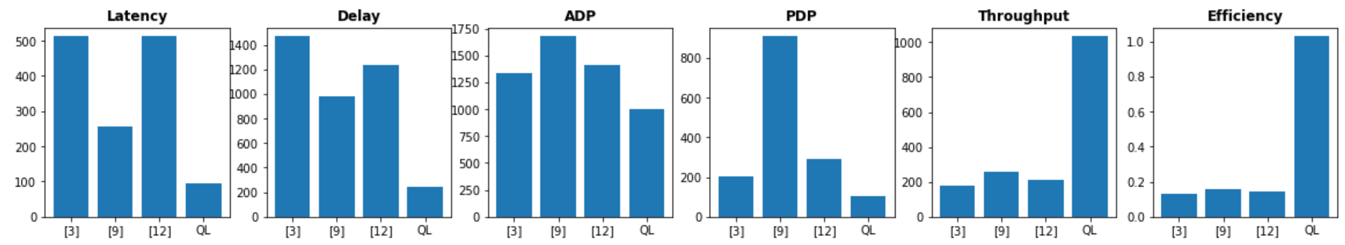

FIGURE 12. Performance comparison of FPGA implementation results for the proposed QL structure with similar structures when $n = 256, q = 256$.

TABLE 5. Theoretical area time complexities for different BRLWE-based architectures.

	#AND	#INV	#RCA	#CSA	#Mux(k-bit)	#Mux(2-bit)	#FF	#Clk	Critical-path*
[3] A-II	nk	k	n	-	1	-	nk	n	$T_{AND} + T_{ADD} + T_{MUX}$
[9]	nk	k+1	n	-	n+1	-	nk	n+1	$\geq T_{AND} + T_{ADD} + T_{MUX}$
[12] u = 1	(k+1)n	2n	n	-	1	n	nk	n	$\sim T_{AND} + T_{ADD} + T_{MUX}$
Prop. DL	2nk	2k	2n	1	2	-	2nk	n/2	$T_{AND} + T_{ADD} + T_{AD3} + T_{MUX}$
Prop. QL	4nk	4k	4n+2	1	4	-	4nk	n/4	$T_{AND} + T_{ADD} + T_{AD3} + T_{MUX}$

The listed area time complexities only include the major components. These do not include the shift register resources for loading the polynomials. Apart from the listed resources, we require incrementers in our proposed architecture. The actual complexities might differ from theoretical considerations.
*:There might be a variation in the actual critical-path as it does not include the input processing setup. This table indicates the theoretical critical-path where $T_{AND}, T_{ADD}, T_{AD3}$ and T_{MUX} refer to the delay time of an AND gate, ripple carry adder (RCA), carry save adder (CSA) and k-bit multiplexer respectively.

calculations, we find the computational time to be much less compared to the previous works i.e., $n/2$ clock cycles for DL structure and $n/4$ clock cycles for QL structure. Though theoretically our design comprises $n/2$ and $n/4$ clock cycles for DL and QL structures, there is a practical limitation in loading the inputs parallelly. Hence, we require an additional 32 and 64 clock cycles to load the input coefficients for $n = 256$ and $n = 512$ respectively as discussed in Section - IV. As per the theoretical calculations listed in Table 5, our design requires more hardware resources compared to the existing works. Though it has more hardware resources, due to the improvement in latency we have less delay and improved Area-Delay Product (ADP) in our implementations.

We have implemented our designs using Verilog HDL for the set of parameters $(n, q) = (256, 256)$ and $(512, 256)$ following the existing works [3], [6], [8], [9], [10], [11], [12], [13], [14]. The implementation is carried out on Xilinx FPGA Virtex - 7 (xc7v2000tfhg1761-2L), and Kintex - 7 (xc7k325tfbg676-2L) respectively on Vivado 2014.2 simulation tool. The practical values of the proposed

TABLE 6. Comparison of various performance metrics for n = 256 and n = 512 on FPGA for decryption phase.

design	n	device	LUT/FF/Slice	Max.freq.[1]	latency[2]	delay[3]	ADP[4]	power[5]	PDP[6]	EPC[7]	Thr.[8]	Efficiency [9]
[3] A-II	256	Virtex - 7	2,580/2,340/907	349	512	1,467	1,331	140	205	0.401	175	0.131
[9]	256	Virtex - 7	5,153/2,151/1,701	261	257	985	1,675	921	907	3.529	260	0.155
[12] u = 1	256	Virtex - 7	3,600/2,568/1,146	415	512	1,234	1,414	233	288	0.561	208	0.147
Prop. DL	256	Virtex - 7	7,470/6,363/2,294	355	159	448	1,027	235	105	0.662	571	0.556
Prop. QL	256	Virtex - 7	13,577/10,450/4,051	383	95	248	1,004	425	105	1.109	1032	1.027
[3] A-II	256	Kintex - 7	2,836/2,340/960	373	512	1,373	1,318	193	265	0.517	186	0.141
[12] u = 1	256	Kintex - 7	3,600/2,568/1,134	394	512	1,299	1,473	237	307	0.602	197	0.133
Prop. DL	256	Kintex - 7	7,479/6,363/2,338	350	159	454	1,061	233	106	0.666	564	0.532
Prop. QL	256	Kintex - 7	13,085/10,450/3,845	367	95	261	1,003	384	100	1.046	980	0.977
[3] A-II	512	Virtex - 7	5,650/4,656/1,894	379	1,024	2,702	5,118	420	1,134	1.108	190	0.037
[9]	512	Virtex - 7	10,285/4,249/3,289	263	513	1,951	6,417	1,871	3,650	7.114	262	0.041
[12] u = 1	512	Virtex - 7	7,184/5,128/2,208	399	1,024	2,566	5,666	456	1,170	1.143	200	0.035
Prop. DL	512	Virtex - 7	15,375/12,709/4,701	335	319	952	4,475	424	404	1.265	538	0.12
Prop. DL	512	Kintex - 7	15,433/12,709/4,613	333	319	958	4,419	419	401	1.258	534	0.12

The authors of [12] have re-implemented the work in [9] including all input processing shift registers for loading. We use these re-implemented values for our comparisons. All the calculations are performed in the decryption phase. Units of Max. freq [1]: MHz. Latency [2] refers to the computation time in clock cycles. Unit for delay[3]: ns. Delay = critical path × latency, where critical path = 1/(Max.freq.). Area-Delay Product (ADP) [4] = Number of slices × delay × 10^3. Power [5] represents dynamic power in mW. Power-Delay Product (PDP)[6] = Dynamic power × Delay. Energy per computation (EPC)[7] = dynamic power / (Max. freq. × number of output coefficients per cycle). Thr.[8] : Throughput = n/Delay × 10^3. Efficiency [9] = Throughput / ADP.

design are listed in Table 6. It includes the number of LUTs, Registers, Slices, maximum frequency, latency, delay, Area-Delay Product (ADP), power, Power-Delay Product (PDP), Energy per computation (EPC), Throughput, and Efficiency. The formulae for calculating various performance metrics are as shown in Table 6. As observed from the Table 6, the proposed DL and QL structures take the advantage of having the least latency of about 159 and 95 clock cycles respectively whereas the existing designs have a latency of 512, 257 and 512 clock cycles considering $n = 256$ [3], [9], [12]. Due to this reduction in latency, the proposed designs also have less delay based on the mathematical formula: $Delay = Critical{-}path \times Latency$ where $Critical - path = 1/Maximum\ \ frequency$. The proposed designs have an improvement in delay comprising $448ns$ and $248ns$ for DL and QL structures respectively whereas the existing designs have $1467ns$,$985ns$ and $1234ns$ respectively for $n = 256$. Though the hardware resources have increased, the decrease in latency and delay made a significant impact in the improvement of ADP. The DL structure has a considerable improvement in ADP of about 23%, 39% and 27% compared to the existing designs [3], [9], [12]. It also has a significant improvement in PDP of about 49%, 88%, and 63% compared to the existing designs [3], [9], [12]. Its Energy per computation (EPC) is 5.3× less compared to [9] indicating good performance of the design. It takes an advantage of having high throughput of about 3.3×, 2.2× and 2.7× compared with existing designs [3], [9], [12] to indicate its high performance of computation when $n = 256$. The efficiency is improved about 4.2×, 3.6× and 3.8× compared with the existing works. On similar lines, we have improvement in performance metrics for QL structure also as listed in Table 6. The QL structure has a considerable improvement in ADP of about 25%, 40% and 29% compared with existing works [3], [9], [12]. It has an improvement in PDP of about 49%, 88% and 63% among the existing works. It has a high throughput of about 5.9×,4× and 5× compared with existing designs. It takes the advantage of having high efficiency of about 7.8×, 6.6× and 7× compared with the existing works. The comparison of latency, delay, ADP, PDP, throughput and efficiency of our proposed DL structure with similar structures are given in the form of bar graphs as shown in Fig. 10 and Fig. 11 when $n = 256, q = 256$ and $n = 512, q = 256$ respectively and implemented in Virtex - 7 (xc7v2000tfhg1761-2L) target device. Similarly, the comparison of these performance metrics of our proposed QL structure with similar structures are given in the form of bar graph as shown in Fig. 12 for $n = 256, q = 256$ when implemented on the same target device. From these bar graphs we observe that the proposed designs has less latency, delay, ADP, PDP, high throughput and efficiency among the existing works. This work can be extended with any number of LFSR structures for obtaining low latency which subsequently results in less delay. As the increase in number of LFSR structures further require more hardware resources, a compromise can be made between the number of LFSR structures and latency based on the design requirement.

VI. CONCLUSION

In this paper, two efficient hardware architectures: Dual-LFSR (DL) and Quad-LFSR (QL) structures are proposed to perform the arithmetic operation used in BRLWE-based PQC schemes. These structures perform the arithmetic operation $H.L + M$ employing parallel LFSR structures to reduce the computational time compared to the existing works. Despite of consuming more hardware resources due to doubling and quadrupling the LFSR structures, it has a significant improvement in other performance metrics such as delay, Area-Delay Product (ADP), Power-Delay Product (PDP), throughput and efficiency. In particular the proposed DL structure has 23%

and QL structure has 25% improvement in ADP compared to recently reported work. The future directions of our work include to develop an efficient hardware architecture for implementing BRLWE-based cryptoprocessors using multi-LFSR structures employing parallel processing technique considering a balance between area and delay.

REFERENCES

[1] F. Özdemir and C. K. Koç, "Development of cryptography since Shannon," *Cryptol. ePrint Arch.*, 2022.

[2] V. Bhatia and K. R. Ramkumar, "An efficient quantum computing technique for cracking RSA using Shor's algorithm," in *Proc. IEEE 5th Int. Conf. Comput. Commun. Autom. (ICCCA)*, Oct. 2020, pp. 89–94.

[3] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 8, pp. 3297–3307, Aug. 2022.

[4] E. Carter, P. He, and J. Xie, "High-performance polynomial multiplication hardware accelerators for kem saber and ntru," *Cryptol. ePrint Arch.*, 2022.

[5] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Des., Codes Cryptogr.*, vol. 75, no. 3, pp. 565–599, Jun. 2015.

[6] P. He, U. Guin, and J. Xie, "Novel low-complexity polynomial multiplication over hybrid fields for efficient implementation of binary ring-LWE post-quantum cryptography," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 2, pp. 383–394, Jun. 2021.

[7] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder, and T. Pöppelmann, "High-performance and lightweight lattice-based public-key encryption," in *Proc. 2nd ACM Int. Workshop IoT Privacy, Trust, Secur.*, May 2016, pp. 2–9.

[8] A. Aysu, M. Orshansky, and M. Tiwari, "Binary ring-LWE hardware with power side-channel countermeasures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1253–1258.

[9] S. Ebrahimi, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "Post-quantum cryptoprocessors optimized for edge and resource-constrained devices in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5500–5507, Jun. 2019.

[10] J. Xie, K. Basu, K. Gaj, and U. Guin, "Special session: The recent advance in hardware implementation of post-quantum cryptography," in *Proc. IEEE 38th VLSI Test Symp. (VTS)*, Apr. 2020, pp. 1–10.

[11] J. Xie, P. He, and W. Wen, "Efficient implementation of finite field arithmetic for binary ring-LWE post-quantum cryptography through a novel lookup-table-like method," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 1279–1284.

[12] J. Xie, P. He, X. Wang, and J. L. Imaña, "Efficient hardware implementation of finite field arithmetic AB+CAB+C for binary ring-LWE based post-quantum cryptography," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 1222–1228, Apr. 2022.

[13] K. Shahbazi and S.-B. Ko, "Area and power efficient post-quantum cryptosystem for IoT resource-constrained devices," *Microprocess. Microsyst.*, vol. 84, Jul. 2021, Art. no. 104280.

[14] B. J. Lucas, A. Alwan, M. Murzello, Y. Tu, P. He, A. J. Schwartz, D. Guevara, U. Guin, K. Juretus, and J. Xie, "Lightweight hardware implementation of binary ring-LWE PQC accelerator," *IEEE Comput. Archit. Lett.*, vol. 21, no. 1, pp. 17–20, Jan. 2022.

[15] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden, "Lattice-based encryption over standard lattices in hardware," in *Proc. 53rd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2016, pp. 1–6.

[16] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss, "On the design of hardware building blocks for modern lattice-based encryption schemes," in *Cryptographic Hardware and Embedded Systems—CHES*. Leuven, Belgium: Springer, 2012, pp. 512–529.

[17] T. Pöppelmann and T. Güneysu, "Towards practical lattice-based public-key encryption on reconfigurable hardware," in *Selected Areas in Cryptography—SAC*. Burnaby, BC, Canada: Springer, 2014, pp. 68–85.

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

[19] S. Roy, F. Vercauteren, N. Mentens, D. Chen, and I. Verbauwhede, "Compact ring-lwe cryptoprocessor," in *Cryptographic Hardware and Embedded Systems—CHES*. Busan, South Korea: Springer, 2014, pp. 371–391.

[20] O. Reparaz, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "A masked ring-lwe implementation," in *Cryptographic Hardware and Embedded Systems—CHES*. Saint-Malo, France: Springer, 2015, pp. 683–702.

[21] R. Salarifard, S. Bayat-Sarmadi, and H. Mosanaei-Boorani, "A low-latency and low-complexity point-multiplication in ECC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 9, pp. 2869–2877, Sep. 2018.

[22] S. Ebrahimi and S. Bayat-Sarmadi, "Lightweight and DPA-resistant post-quantum cryptoprocessor based on binary ring-LWE," in *Proc. 20th Int. Symp. Comput. Archit. Digit. Syst. (CADS)*, Aug. 2020, pp. 1–6.

[23] Y. Zhang, C. Wang, D. E. S. Kundi, A. Khalid, M. O'Neill, and W. Liu, "An efficient and parallel R-LWE cryptoprocessor," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 5, pp. 886–890, May 2020.

[24] 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 Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 8, pp. 1007–1019, Aug. 2022.

[25] K. Shahbazi and S.-B. Ko, "An optimized hardware implementation of modular multiplication of binary ring LWE," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 33, pp. 817–821, Jul./Sep. 2023.

[26] P. He, T. Bao, J. Xie, and M. Amin, "FPGA implementation of compact hardware accelerators for ring-binary-LWE-based post-quantum cryptography," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 16, no. 3, pp. 1–23, Sep. 2023.

[27] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, Sep. 2009.

[28] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. 29th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 1–23.

[29] P. He, Y. Tu, J. Xie, and H. S. Jacinto, "KINA: Karatsuba initiated novel accelerator for ring-binary-LWE (RBLWE)-based post-quantum cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 31, no. 10, pp. 1551–1564, Oct. 2023.

[30] S. E. Mathe and L. Boppana, "Design and implementation of a sequential polynomial basis multiplier over GF(2^{m})," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 5, pp. 2680–2700, 2017.

[31] S. E. Mathe and L. Boppana, "Low-power and low-hardware bit-parallel polynomial basis systolic multiplier over GF(2^m) for irreducible polynomials," *ETRI J.*, vol. 39, no. 4, pp. 570–581, Aug. 2017.

[32] S. E. Mathe and L. Boppana, "Design and implementation of a novel bit-parallel systolic multiplier over GF(2^m) for irreducible pentanomials," *J. Circuits, Syst. Comput.*, vol. 27, no. 14, Dec. 2018, Art. no. 1850228.

SHAIK AHMADUNNISA received the B.Tech. degree in electronics and communication engineering from Andhra University, in 2013, and the M.Tech. degree in VLSI design from JNTU, Kakinada, in 2015. She is currently pursuing the Ph.D. degree with the School of Electronics Engineering, VIT-AP University, India. Her research interests include VLSI architectures and post quantum cryptography.

SUDHA ELLISON MATHE (Member, IEEE) received the B.Tech. degree in electronics and communications engineering from Acharya Nagarjuna University, Guntur, India, in 2011, the M.E. degree in embedded systems from the Birla Institute of Technology and Science, Pilani, India, in 2013, and the Ph.D. degree in VLSI architectures with cryptographic applications from the National Institute of Technology, Warangal, India, in 2018. He has been an Associate Professor with Vellore Institute of Technology (VIT-AP) University, Amaravati, India. His research interests include VLSI architectures, post quantum cryptography, finite field arithmetic, the IoT, embedded systems, and FPGA and ASIC design.