An Efficient and Parallel R-LWE Cryptoprocessor

Yuqing Zhang, Chenghua Wang, Dur E Shahwar Kundi, *Member, IEEE*, Ayesha Khalid, *Member, IEEE*, Máire O'Neill, *Senior Member, IEEE*, and Weiqiang Liu, *Senior Member, IEEE*

***Abstract*—Lattice-based cryptography (LBC) is a promising and efficient public key cryptography scheme whose theoretical foundation usually lies in Learning with Error (LWE) problem and its variant such as Ring-LWE (R-LWE) is the most studied cryptosystem which allows for more efficient implementation while maintaining the hardness of an original problem. Polynomial multiplication is the bottleneck of R-LWE, that can either be done using Number Theoretic Transform (NTT) or schoolbook polynomial multiplication (SPM) algorithm. The use of SPM is wider and possible for all parameters of R-LWE schemes. This work proposes an efficient and parallel strategy for SPM in R-LWE; by successfully reducing its time complexity from n^2 to $n^2/4$ (making it 1.8× faster and 1.4× hardware efficient). Furthermore, by adjusting the bit width for the error terms, the polynomial multiplication and addition blocks are reused for both encryption and decryption modules resulting in 14% reduced area and 1.7× better throughput in comparison to state-of-art SPM based R-LWE designs.**

***Index Terms*—Lattice-based cryptography (LBC), Schoolbook polynomial multiplication (SPM), Ring Learning with Errors (R-LWE)**

I. Introduction

Due to the algorithm proposed by Shor [1], classic public key cryptography (PKC) algorithms such as elliptic curve cryptography (ECC) and RSA have been proved to be vulnerable to attacks performed by quantum computers. In the round 2 of National Institute of Standards and Technology (NIST) post-quantum cryptography (PQC) standardization process, lattice-based cryptosystems (LBC) are the most promising candidates and account for 53% of the total of public key encryption (PKE) schemes [2].

Polynomial multiplication is a core module in LBC and is the bottleneck of the cryptosystem, which can be typically implemented by schoolbook and NTT algorithm. To speed up the operation of polynomial multiplication, usually NTT algorithm is utilized. However, it can only be used with selected parameters of cryptography and needs complex operations, including pre-computation, array reordering and post-computation.

In fact, half of the round 2 LBC candidates such as LAC, FRODO-KEM, Round5, Saber, Threebears and NTRUPrime in the NIST PQC initiative [2] have not used NTT method. In spite of the time complexity of schoolbook algorithm being larger than that of NTT algorithm, its simple implementation and low resource consumption makes it attractive for resource-constraint applications. Various architectures for NTT and schoolbook algorithm are analyzed and implemented comprehensively in [3]. NTT based Compact R-LWE cryptoprocessor are proposed in [4], [5], whereas NTT based high-throughput R-LWE cryptoprocessor are proposed in [6], [7].

In schoolbook polynomial multiplication (SPM), very little research is focused on the trade-off between performance and resource consumption. A lightweight design for SPM was proposed in [8], but it does not optimize the iteration clock cycle, as a result the throughput is not very high. On the other hand, [9] proposed an optimized SPM that reduces half of the iterative clock cycle. However, since its Gaussian sampling is signed data, some modules in encryption and decryption cannot be shared.

In this paper, we proposed a highly efficient SPM based R-LWE architecture by exploiting the features of polynomial multiplication. In addition, the bit width of Gaussian sampled data is adjusted appropriately, so that the polynomial multiplication module can be reused. In this method, only one module of polynomial multiplication is needed for the full R-LWE cryptosystem, which ensures reduced consumption of resources.

The paper is organized as follows: Section II reviews the background for R-LWE public key scheme. Section III presents the design of schoolbook polynomial multiplier and describes the entire structure of R-LWE. Section IV presents the hardware implementation results of the proposed designs, and comparison with other works. Section V concludes this paper.

II. Background

The hardware implementation of R-LWE mainly includes a discrete Gaussian sampler and a polynomial multiplier. The entire process of encryption and decryption is shown in Alg. 1. In the algorithm, all operations are based on polynomials on the ring $\mathbb{Z}_q[x]/(x^n+1)$, where n is a power-of-two integer representing lattice dimension while q is prime modular [10]. U is a uniform distribution, and D_σ is a Gaussian distribution with $\mu = 0$ and a variance of σ. '×' and '+' represent polynomial multiplication and polynomial addition, respectively.

We use the Cumulative Distribution Table (CDT) based scheme for discrete Gaussian sampling. CDT sampling is easier to be implemented in hardware than other sampling methods, it promises low resource consumption and high-throughput [11]. In addition, CDT sampling has been proven to

This work is supported by grants from the National Natural Science Foundation of China (61871216 and 61771239), the Fundamental Research Funds for the Central Universities China (NE2019102), the Six Talent Peaks Project in Jiangsu Province (2018XYDXX-009). This brief is recommended by ISCAS-2020 Special Issue Guest Editors

(Corresponding author: Weiqiang Liu.)

Yuqing Zhang, Chenghua Wang, Dur E Shahwar Kundi, and Weiqiang Liu are with College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: {zhangyuqing, chwang, dureshahwar, liuweiqiang}@nuaa.edu.cn.

Ayesha Khalid, and Máire O'Neill are with the Centre for Secure Information Technologies (CSIT), Queen's University Belfast, UK. E-mail: a.khalid@qub.ac.uk, m.oneill@ecit.qub.ac.uk.

Algorithm 1 The Encryption and Decryption of R-LWE

```
Input: Public key: $\boldsymbol{a} \leftarrow U$ and $\boldsymbol{p}$; Secret key: $\boldsymbol{r_2} \leftarrow D_\sigma$;
    Plaintext: $\boldsymbol{m} \leftarrow \{0,1\}$.
Output: Ciphertext: $\boldsymbol{c_1}$, $\boldsymbol{c_2}$; Decrypted plaintext: $\boldsymbol{m'}$.
1: Sampling to generate $\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3} \leftarrow D_\sigma$;
2: Encoding plaintext $\boldsymbol{\bar{m}} = \boldsymbol{encode(m)}$;
3: Ciphertext is calculated $\boldsymbol{c_1} = \boldsymbol{a} \times \boldsymbol{e_1} + \boldsymbol{e_2}$, $\boldsymbol{c_2} = \boldsymbol{p} \times \boldsymbol{e_1} + \boldsymbol{e_3} + \boldsymbol{\bar{m}}$;
4: Decrypting $\boldsymbol{c} = \boldsymbol{c_1} \times \boldsymbol{r_2} + \boldsymbol{c_2}$;
5: Decoding plaintext $\boldsymbol{m'} = \boldsymbol{decode(c)}$;
6: return $\boldsymbol{c_1}$, $\boldsymbol{c_2}$, $\boldsymbol{m'}$.
```

be resistant to side-channel attack (SCA), scalable in nature [12], and adaptable to deter fault-attacks [13]. The product $c(x)$ of two integer polynomial $a(x)$ and $b(x)$ can be computed by SPM algorithm, as expressed by (1) [3]:

$$\begin{aligned} a(x) \cdot b(x) &= \left[\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j x^{i+j} \right] \bmod (x^n + 1) \\ &= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (-1)^{\lfloor (i+j)/n \rfloor} a_i b_j x^{(i+j) \bmod n} \end{aligned} \tag{1}$$

The sign bit of the coefficient in polynomial $c(x)$ is determined by $(-1)^{\lfloor (i+j)/n \rfloor}$ as follows :

$$sign \;\; bit = \begin{cases} 0 & i + j < n \\ 1 & n \le i + j \le 2n - 2 \end{cases} \tag{2}$$

The method of shift-addition-multiplication-subtraction-subtraction (SAMS2) [14] is taken up to accelerate the implementation of modular reduction operation in hardware. A well-optimization structure for SPM is proposed in [9]. For discrete Gaussian noise distribution, sampling with signed numbers is proposed for the first time, which reduces resources consumption. The formula of signed modular multiplication is expressed by (3):

$$a \times (-e) \bmod q = q - [\, a \times e \bmod q \,] \tag{3}$$

Where, a represents an unsigned number, e is a signed Gaussian noise and q is the prime modulus. Thus a 13-bit Gaussian sampled number can be written as a 6-bit signed number for q = 7681 and the original 13×13 bit modular multiplication becomes 13×5 bit modular multiplication and one subtraction. Furthermore, as a result of this reduction, a single digital signal processing slice (i.e. DSP48) in Xilinx 7 series FPGA is configured to accommodate 2 simultaneous multiplications [9], which in turn reduces the time complexity of SPM by half (from $\boldsymbol{n^2}$ to $\boldsymbol{n^2/2}$).

III. Proposed Optimized Structure of R-LWE

In this section, we present our efficient SPM design and later the hardware sharing structure for overall implementation of R-LWE hardware. In this work, a medium security level R-LWE parameter set (n = 256, q = 7681, s = 11.31) is selected [15].

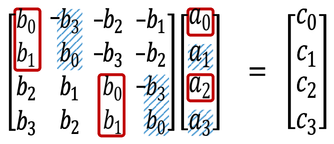

Fig. 1. Proposed memory access scheme.

A. An Efficient Design of SPM

Polynomial multiplication is the most critical part of overall implementation, determining the performance of R-LWE hardware design. It is a slow operation, since calculating one

Algorithm 2 High Efficiency SPM for R-LWE via Splitting

```
Input:
    $\boldsymbol{a(x)}$, $\boldsymbol{b(x)}$ (polynomial in $\mathbb{Z}_q[x]/(x^n+1)$);
    $q$ prime modular;
Output:
    $\boldsymbol{c(x)}$ (polynomial in $\mathbb{Z}_q[x]/(x^n+1)$);
1: for $i = 0 : 4 : n - 4$ do
2:   for $j = 0 : n - 1$ do
3:     $coeff\_a0 \leftarrow a[j]$;
4:     $coeff\_a1 \leftarrow a[(j+2) \bmod n]$;
5:     $coeff\_b0 \leftarrow b[(i-j) \bmod n]$;
6:     $coeff\_b1 \leftarrow b[(i-j+1) \bmod n]$;
7:     $me\_b \leftarrow \{coeff\_b0[4:0], 13'b0, coeff\_b1[4:0]\}$;
8:     $temp\_ab1 \leftarrow coeff\_a0 \times me\_b$;
9:     $temp\_ab2 \leftarrow coeff\_a1 \times me\_b$;
10:    $temp1 \leftarrow temp\_ab1[17:0] \bmod q$;
11:    $temp2 \leftarrow temp\_ab1[35:18] \bmod q$;
12:    $temp3 \leftarrow temp\_ab2[17:0] \bmod q$;
13:    $temp4 \leftarrow temp\_ab2[35:18] \bmod q$;
14:    $sign1 \leftarrow (i < j)\ ?\ 1 : 0$;
15:    $sign2 \leftarrow (i + 1 < j)\ ?\ 1 : 0$;
16:    $sign3 \leftarrow (i + 2 < (j+2) \bmod n)\ ?\ 1 : 0$;
17:    $sign4 \leftarrow (i + 3 < (j+2) \bmod n)\ ?\ 1 : 0$;
18:    if $sign1 \oplus coeff\_b0[5] == 1$ then
19:      $temp1 \leftarrow q - temp1$;
20:    end if
21:    if $sign2 \oplus coeff\_b1[5] == 1$ then
22:      $temp2 \leftarrow q - temp2$;
23:    end if
24:    if $sign3 \oplus coeff\_b0[5] == 1$ then
25:      $temp3 \leftarrow q - temp3$;
26:    end if
27:    if $sign4 \oplus coeff\_b1[5] == 1$ then
28:      $temp4 \leftarrow q - temp4$;
29:    end if
30:    $c[i] \leftarrow temp1 + c[i]$;
31:    $c[i+1] \leftarrow temp2 + c[i+1]$;
32:    $c[i+2] \leftarrow temp3 + c[i+2]$;
33:    $c[i+3] \leftarrow temp4 + c[i+3]$;
34:  end for
35: end for
36: return $\boldsymbol{c(x)}$.
```

polynomial multiplication with dimensions of $\boldsymbol{n}$ requires to compute $\boldsymbol{n^2}$ times modular multiplications and $\boldsymbol{(n-1)^2}$ times modular additions/ subtractions. Our strategy is to improve the parallelism in the design of SPM to reduce its time complexity and thereby improving its performance. As shown in Alg. 1, two operations of polynomial multiplication are needed in encryption, while one in decryption.

Public key, secret key and errors require 2 Block RAMs (BRAMs) for storage. Based on the operational features of the SPM, a scheme of efficient memory accessing is proposed. For simplicity, toy setting with $n = 4$ is considered as an example to illustrate the polynomial multiplication. In order to intuitively display the operation of $c(x) = a(x) \times b(x)$, it is represented in the form of a matrix as shown in Fig. 1. In the first clock cycle, the corresponding coefficients of $a(x)$ and $b(x)$ as highlighted by red box are accessed from the memory and after their retrieval, the strategy of full utilization of FPGA DSP Blocks in [9] is used to compute 4 partial products of $c(x)$ at once. Then, in second clock cycle the next coefficients of $a(x)$ and $b(x)$ as highlighted by blue shade are accessed to generate the next 4 partial products of $c(x)$, and so on. Hence, the iteration period of the polynomial multiplication is reduced from 16 to 4 for $n = 4$.

Based on the proposed memory access method in Fig. 1, a scheme of parallel SPM is proposed, which mainly includes 6 steps to determine $a(x) \times b(x)$, detailed outline in Alg. 2:

1) Reading coefficients $a[j]$, $a[(j + 2) \ mod \ n]$ from RAM_A that stores $a(x)$ and coefficients $b[(i - j) \ mod \ n]$, $b[(i - j + 1) \ mod \ n]$ from RAM_B that stores $b(x)$ in one clock cycle (line $3 - 6$).
2) Merging data bit of $b[(i - j) \ mod \ n]$ and $b[(i - j + 1) \ mod \ n]$ into a single 23-bit number (line 7).
3) Performing a modular multiplication operation of 23-bit $\times$ 13-bit (line 8−9).
4) Separating the 18-bit data after the 36-bit product (line 10−13).
5) Determining the sign of separated data to carry out in-place reduction as well as subtraction from modulus q based on (3) (line 14−28). The value of $sign1$ to $sign4$ represents the condition for in-place reduction (line 14−17) whereas MSB of $coeff\_b0[5]$ & $coeff\_b1[5]$ provides the coefficients sign of polynomial $b(x)$. Both are $XORed$, if equal to "1" then $temp1$ ... $temp4$ are to be subtracted from the modulus q (line 18−28).
6) Final accumulation of generated result (line 30−33).

B. Structure of Polynomial Multiplier (PM)

A high-performance and parallel structure of polynomial multiplier is proposed based on Alg. 2, is shown in Fig. 2. The polynomial multiplier has polynomials $a(x)$ and $b(x)$ as multiplicand and multiplier. Their values are accessed every clock cycle. After data bit of $coeff\_b0$ and $coeff\_b1$ are merged, operation of multiplication is performed. The proposed structure requires only 2 DSPs, while a single DSP block is supporting a 25 $\times$ 18 bit multiplication in Xilinx Kintex-7 FPGA. Resultant product is split into two halves and reduced via Barret Reduction (BR) module. The data is split into a total of four numbers, $temp1 \ldots temp4$. The most significant bit (MSB) of both coefficients $coeff\_b0$ and $coeff\_b1$ are input to the control unit before the multiplication operation, and sign of the final reduced data is determined by control unit. Finally, the generated 4 data are used as inputs to 4 modules of accumulation (Acc.) respectively. After completing all iterations, the 4 coefficients of polynomial $c(x)$ are output, which are $coeff\_c0$, $coeff\_c1$, $coeff\_c2$ and $coeff\_c3$.

C. Reuse of Polynomial Multiplication and Addition in both R-LWE Encryption/ Decryption.

Further to enhance the design efficiency, we exploit a smart and novel resource-sharing possibility for a combined R-LWE

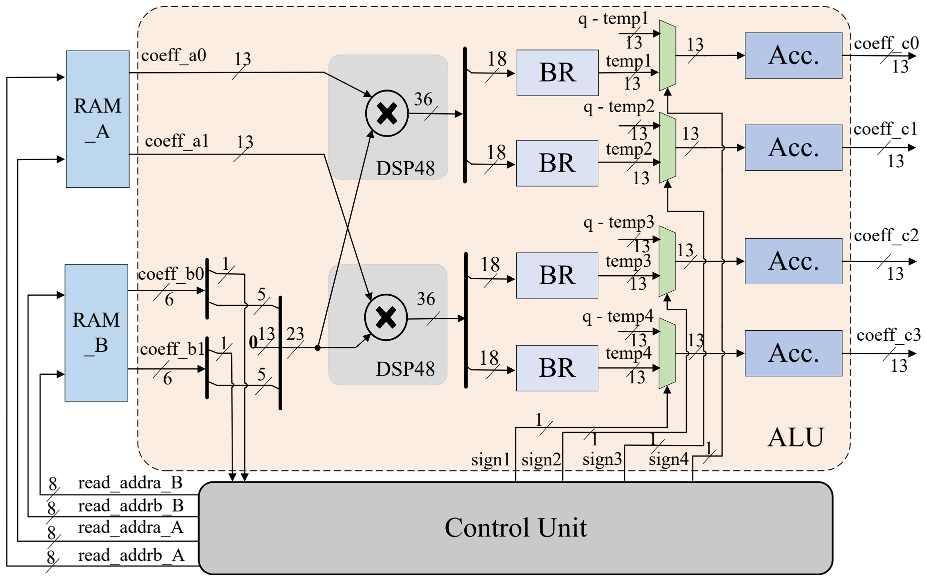

Fig. 2. The proposed structure of the efficient polynomial multiplier (PM).

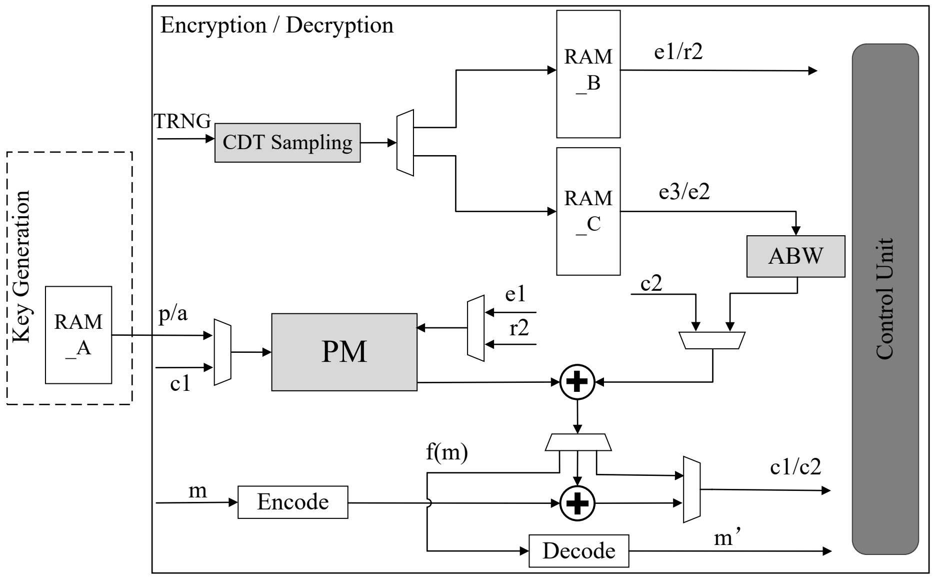

Fig. 3. Combined structure of encryption and decryption modules.

cryptoprocessor design. It stems from a simple observation that encryption and decryption operations have the same operation rules, both of which are $d(x) = a(x) \times b(x) + e(x)$, as can be seen in Alg. 1 (line 3-4). Hence decryption module can reuse the polynomial multiplication as well as the polynomial addition in encryption module. However, error items e_2 and e_3 are 6-bit signed samples from a narrow discrete Gaussian distribution, hence to enable the reuse of polynomial addition hardware, the bit width of e_2 and e_3 is changed from 6 bits to 13 bits as presented by Alg. 3. If $e(x)$ is negative (i.e. $e[5]$ = 1), the data bits (last 5 bits) of this number should be subtracted from the modulus q (line 4). Otherwise, MSB bits of $e(x)$ is padded by "0" to make it 13 bits (line 6).

Algorithm 3 Adjusting Bit Width (ABW) of Error Terms

```
Input: $e[N]$ (each element is a 6-bit signed integer)
    $q$ prime modular
Output: $f[N]$ (each element is a 13-bit unsigned integer)
1: for $i = 0 : N - 1$ do
2:     $reg\_temp \leftarrow e[i]$;
3:     if $reg\_temp.\,get\_bit(0) == 1$ then
4:         $f[i] \leftarrow q - reg\_temp$;
5:     else
6:         $f[i] \leftarrow reg\_temp \times (1 << 8) + reg\_temp$;
7:     end if
8: end for
9: return $f[N]$.
```

The proposed polynomial multiplier and adder can be utilized by the R-LWE cryptosystem, including key generation, encryption and decryption. The block diagram of the combined hardware-sharing structure is shown in Fig. 3, comprising of the discrete Gaussian sampler, Encryption and Decryption modules. As can be seen from this proposed structure, the polynomial multiplier is utilized by all the three operation of key-generation, encryption and decryption. The execution sequence is controlled by the control unit. Implementation of the Adjusting Bit Width (ABW) unit follows Alg. 3.

This thrifty reuse of schoolbook polynomial multiplier and the polynomial adder (PA) results in the reduction of area consumption of hardware resources, which has not been undertaken in the reported implementations.

IV. Hardware Implementation Results

The proposed structures are synthesized and implemented on a Kintex-7 (KC705) FPGA using Xilinx Vivado 2018.3. Table I lists the detailed resource consumption, performance results (frequency, execution time and throughput) and hardware efficiency of FPGA based designs in terms of Throughput per Slice (TPS) from the post place and route (post-PAR) results. TPS is a trade off between throughput and the utilized hardware resources to achieve that throughput, thus higher value of TPS represents a good balanced design. To calculate the TPS for each implementation, a Slice equivalency is taken in account for dedicated/embedded resources of FPGA such as BRAM and DSP [9]. For a fair comparison of our proposed R-LWE design, we selected SPM based hardware implementations only. To facilitate the comparison, first the encryption and decryption architectures of proposed R-LWE design incorporating efficient polynomial multiplier is provided separately and then in the combined one.

The R-LWE design proposed in [8] is a lightweight design, but takes more clock cycles hence results in very low throughput as well as TPS. The design is based on parameters set of (256, 4092/4093, 8.35) that not only provided low security but also the structure of modular multiplication consumed less area resources. Moreover it utilized Bernoulli approach for rejection sampling that is more vulnerable to SCA. The design presented in [16], is not only based on better parameters set (256, 7681, 11.31) to achieve medium security but also utilized the CDT Gaussian sampling which is more resistant to timing attacks. Whereas, an optimized R-LWE design in [9] made

TABLE I
FPGA Based Results Comparison (post-PAR) with Other Designs

Implementation	Device	Type	LUT/FF/Slice	DSP*	BRAM**	Freq. MHz	Cycle	Time (μs)	Throughput (Kbps)	TPS (Kbps/Slice)
[8] (Schoolbook)	Spartan-6	Enc	360/290/114	1	2	128	136986	1070	239.21	0.73
		Dec	162/136/51	1	1	179	66304	370	691.12	3.30
[9] (Schoolbook)	Kintex-7	Enc	898/815/303	1	3	304.69	69654	229	1119.84	1.95
		Dec	635/190/194	1	1	303.40	34436	114	2255.49	6.40
[16] (Schoolbook)	Kintex-7	Enc	1098/407/337	1	0	288	131604	457	560.9	1.27
		Dec	609/318/182	1	0	288	65802	228	1120.45	3.94
This Work	Kintex-7	Enc	1254/1046/402	2	2	280	35478	128	2020.4	2.81
		Dec	722/558/249	2	0	292	17732	61	4215.65	9.22
This Work	Kintex-7	Enc/Dec	1381/1179/479	2	2	275	35478/17732	129/64	1984.32/3970.22	2.56/4.99

*1 DSP equivalent to 102.4 Slices, **1 8K BRAM equivalent to 56 Slices [9]

use of signed bit sampling and full utilization of FPGA DSP Blocks and achieved the highest throughput of 1,119.84 Kbps and 2,255.49 Kbps for encryption and decryption, respectively.

According to the features of the SPM, we devised strategic memory accessing from the BRAM and used 2 DSPs with 100% utilization which enable us to perform 4 multiplication operations in one clock cycle. Hence, reduce the overall latency of the design by 4 and achieved highest throughput values. Compared with the encryption and decryption modules of [9], our proposed separate designs have 1.8$\times$ throughput and 1.4$\times$ better hardware efficiency (i.e. TPS) at a cost of just 25.35% increased hardware.

Further, adjusting the bit width for the error terms helps us to improve the area consumption by re-using the hardware of SPM and PA for both the encryption and decryption processes of R-LWE. Our proposed combine architecture resulted in reduced area consumption for the overall cryptosystem i.e. utilized 796 Equivalent Slices (479+2$\times$102.4+2$\times$56). In comparison to the combined area resources of both encryption and decryption designs in [9], we saved 14% area and still have better throughput figures i.e. 1.7$\times$ of the state-of-art fastest available SPM based R-LWE design [9].

V. Conclusion

This work proposes an efficient and parallel SPM structure for the most critical operation of polynomial multiplication in R-LWE. We exploit the operational features of polynomial multiplication to implement a parallel design, which reduced iteration clock cycle by factor of 4 and results in 1.8$\times$ speedup and 1.4$\times$ TPS. Furthermore, an idea of re-using the most critical parts of the R-LWE encryption and decryption hardware is presented in this work. The combined R-LWE encryption/decryption architecture have 14% reduced area with 1.7$\times$ throughput.

References

[1] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in *Proc. 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[2] “National Institute of Science & Technology (NIST) PQC Round 2 was last accessed Feb. 2019,” https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[3] T. Pöppelmann and T. Güneysu, “Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware,” in *Proc. International Conference on Cryptology and Information Security in Latin America*, 2012, pp. 139–158.

[4] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, “Compact Ring-LWE cryptoprocessor,” in *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, 2014, pp. 371–391.

[5] A. Aysu, C. Patterson, and P. Schaumont, “Low-cost and area-efficient fpga implementations of lattice-based cryptography,” in *Proc. IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 81–86.

[6] C. P. Rentería-Mejía and J. Velasco-Medina, “High-throughput Ring-LWE cryptoprocessors,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, pp. 2332–2345, 2017.

[7] T. N. Tan and H. Lee, “Efficient-scheduling parallel multiplier-based Ring-LWE cryptoprocessors,” *Electronics*, vol. 8, no. 4, p. 413, 2019.

[8] T. Pöppelmann and T. Güneysu, “Area optimization of lightweight lattice-based encryption on reconfigurable hardware,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 2796–2799.

[9] W. Liu, S. Fan, A. Khalid, C. Rafferty, and M. O’Neill, “Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on FPGA,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2459–2463, 2019.

[10] V. Lyubashevsky, C. Peikert, and O. Regev, “On ideal lattices and Learning with Errors over rings,” in *Proc. Advances in Cryptology (EUROCRYPT)*, 2010, pp. 1–23.

[11] J. Howe, A. Khalid, C. Rafferty, F. Regazzoni, and M. O’Neill, “On practical discrete Gaussian samplers for lattice-based cryptography,” *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 322–334, 2016.

[12] A. Khalid, J. Howe, C. Rafferty, F. Regazzoni, and M. O’Neill, “Compact, scalable, and efficient discrete gaussian samplers for lattice-based cryptography,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[13] J. Howe, A. Khalid, M. Martinoli, F. Regazzoni, and E. Oswald, “Fault attack countermeasures for error samplers in lattice-based cryptography,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[14] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede, “Efficient Ring-LWE encryption on 8-bit AVR processors,” in *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, 2015, pp. 663–682.

[15] 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 *Proc. Cryptographic Hardware and Embedded Systems*, 2012, pp. 512–529.

[16] S. Fan, W. Liu, J. Howe, A. Khalid, and M. O’Neill, “Lightweight hardware implementation of R-LWE lattice-based cryptography,” in *Proc. IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2018, pp. 403–406.