Transactions Briefs

Optimized Schoolbook Polynomial Multiplication for Compact Lattice-Based Cryptography on FPGA

Weiqiang Liu, Sailong Fan, Ayesha Khalid, Ciara Rafferty, and Máire O'Neill

***Abstract*—Lattice-based cryptography (LBC) is one of the most promising classes of post-quantum cryptography (PQC) that is being considered for standardization. This brief proposes an optimized schoolbook polynomial multiplication (SPM) for compact LBC. We exploit the symmetric nature of Gaussian noise for bit reduction. Additionally, a single field-programmable gate array (FPGA) DSP block is used for two parallel multiplication operations per clock cycle. These optimizations enable a significant 2.2× speedup along with reduced resources for dimension $n = 256$. The overall efficiency (throughput per slice) is 1.28× higher than the conventional SPM, as well as contributing to a more compact LBC system compared to previously reported designs. The results targeting the FPGA platform show that the proposed design can achieve high hardware efficiency with reduced hardware area costs.**

***Index Terms*—Field-programmable gate array (FPGA), lattice-based cryptography (LBC), polynomial multiplication.**

I. Introduction

Traditional public key cryptography algorithms, including RSA and elliptic-curve cryptography (ECC), will no longer be secure in the near future, due to advancements in quantum computing. The National Institute of Standards and Technology (NIST) called for the proposal of post-quantum cryptographic (PQC) algorithms [1] and received 70 PQC algorithm submissions. Among the potential PQC algorithms to be standardized, lattice-based cryptography (LBC) is one of the most promising types. Almost half of the PQC candidates in round 2 of the PQC standardization process are lattice-based [2]. LBC algorithms are based on the hard problem of finding the shortest (or closest) vector (SVP or CVP) in a lattice. These problems are believed to be hard for both classical and quantum computers.

Polynomial multiplication plays a critical role in LBC and is typically carried out by schoolbook or number theoretic transform (NTT) multiplication. Schoolbook polynomial multiplication (SPM) is a naive method, requiring direct multiplication and subsequent accumulation of results. Although it is slow, it offers simple implementation and low hardware resource cost. NTT is a much faster alternative that comes with additional hardware resource costs and complexity in terms of operations (pre-computation, array reordering, and post-computation) apart from the NTT/inverse-NTT butterflies. Both methods have been widely used for different scenarios; a relevant survey discusses various butterfly architectures and techniques [3]. Pöppelmann and Güneysu [4] proposed adaptable and extensible hardware implementations of both NTT and SPM methods supporting various operand sizes. Working toward efficient design, Du and Bai [5] reduced the required clock cycles and saved storage by exploring the characteristics of twiddle factors. In the case of SPM, limited research exists: [6] suggested area optimization techniques for SPM hardware. Moreover, [7] employed SPM in LBC for digital signatures on field-programmable gate array (FPGA)s. Until now, little research has been conducted in terms of a thorough tradeoff between performance and hardware consumption for SPM. Furthermore, in round 2 of the NIST PQC initiative, half of the 12 LBC contestants, including FRODO-KEM, Round5, Saber, Threebears, and NTRUPrime, do not use NTT [2]. NTT is suitable for the parameters on the specific modulo ring, while schoolbook multiplication is a more generic approach. Therefore, it is important to explore how to efficiently implement schoolbook multiplication.

Ring-learning with errors (R-LWE) is a widely investigated algorithm that is based on a hard lattice problem. The most critical operation in the R-LWE schemes is polynomial multiplication on the ring. It operates on the ring $\mathbb{Z}_q[x]/(x^n+1)$, where q is the modular prime.

This brief proposes a compact and efficient hardware design for R-LWE encryption/decryption based on SPM. We exploit the distribution symmetry of Gaussian noise to achieve a reduced bit width and full utilization of DSP blocks. A compact SPM is designed with approximately 2× speedup without additional hardware resource consumption. A comparison with the existing R-LWE designs is provided, which highlights the efficiency of our proposed design. The proposed design optimizations can also be undertaken for other LBC schemes and other FPGA families.

TABLE I
R-LWE Encryption Scheme

	Polynomial $\boldsymbol{a} \leftarrow U$ and $\boldsymbol{p}$ are the public key, $\boldsymbol{r_2} \leftarrow D_\sigma$ is the secret key.
$Enc(\boldsymbol{a}, \boldsymbol{p}, m)$	Generate $\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3} \leftarrow D_\sigma$. Let $\bar{\boldsymbol{m}} = ENCODE(m)$. Then the cipher text is $\boldsymbol{c_1} = \boldsymbol{a}\boldsymbol{e_1} + \boldsymbol{e_2}$, $\boldsymbol{c_2} = \boldsymbol{p}\boldsymbol{e_1} + \boldsymbol{e_3} + \bar{\boldsymbol{m}}$.
$Dec(\boldsymbol{c_1}, \boldsymbol{c_2})$	$m' = DECODE(\boldsymbol{c} = \boldsymbol{c_1}\boldsymbol{r_2} + \boldsymbol{c_2})$.

II. Preliminary Background

Table I details the R-LWE-based public key encryption (PKE) scheme. We focus on encryption and decryption, assuming that key generation can be carried out infrequently and offline. The hardware

Manuscript received February 23, 2019; revised May 24, 2019; accepted June 11, 2019. Date of publication June 28, 2019; date of current version September 25, 2019. This work was supported by National Natural Science Foundation of China (NSFC) under Grant 61871216, the Fundamental Research Funds for the Central Universities China under Grant NE2019102, and Six Talent Peaks Project in Jiangsu Province under Grant 2018-XYDXX-009. *(Corresponding author: Weiqiang Liu.)*

W. Liu and S. Fan are with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China (e-mail: liuweiqiang@nuaa.edu.cn; fansl4g@nuaa.edu.cn).

A. Khalid, C. Rafferty, and M. O'Neill are with the Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast BT7 1NN, U.K. (e-mail: a.khalid@qub.ac.uk; c.m.rafferty@qub.ac.uk; m.oneill@ecit.qub.ac.uk).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2922999

Algorithm 1 Schoolbook PMA for Encryption or Decryption

```
Input:
    $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, (polynomial in $\mathbb{Z}_q[x]/(x^n+1)$);
    $q$, prime modular;
Output:
    $\boldsymbol{d}$, (polynomial in $\mathbb{Z}_q[x]/(x^n+1)$);
1:  for $i = 0 : n-1$ do
2:      $sum \leftarrow c[i]$;
3:      for $j = 0 : n-1$ do
4:          $ab \leftarrow a[j] \times b[(i-j)\ mod\ n]$;
5:          $ab\_m \leftarrow ab\ mod\ q$;
6:          $sig \leftarrow (i < j)\ ?\ 1 : 0$;
7:          if sig == 1 then
8:              $ab\_m \leftarrow q - ab\_m$;
9:          end if
10:         $tmp \leftarrow sum + ab\_m$;
11:         $sum \leftarrow tmp\ mod\ q$;
12:     end for
13:     $d[i] \leftarrow sum$;
14: end for
15: return $\boldsymbol{d}$
```

resource requirements for R-LWE encryption and decryption are mainly due to the SPM. D_σ is the Gaussian distribution with standard deviation σ, and U is the uniform distribution. Polynomials $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are the ciphertext results. The decryption procedure also performs polynomial multiplication and addition and decodes the polynomial to plaintext. Please refer to [8] for more details on the R-LWE scheme. It is evident that polynomial multiplication is the most computationally intensive part of the cryptographic scheme.

The typical SPM algorithm used in R-LWE can be expressed as in Table 1 [4]. Considering the property of polynomial multiplication that $x^n \equiv -1$, note that the product $c(x) = a(x) \times b(x)$ is not a normal circumferential convolution. It has a sign bit in the accumulation, namely, $(-1)^{\lfloor (i+j)/n \rfloor}$. This sign bit is 1 if $i + j < n$ and -1, otherwise. The dimension is denoted as n, which means that this method has $O(n^2)$ complexity

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

Polynomial addition is an ordered sequential addition, well suited for a low-cost hardware implementation. To evaluate the encryption/decryption performance of R-LWE, we implement both polynomial multiplication and addition (PMA) on FPGA, as shown in Algorithm 1. This algorithm calculates $\boldsymbol{d} = \boldsymbol{a} * \boldsymbol{b} + \boldsymbol{c}$. First, the elements a multiplied b are calculated, then the Barrett reduction algorithm is used to perform the modular operation with the prime (q), and finally, the result is assigned to $\boldsymbol{d}$. The modular reduction in line 11 only requires a multiplexer due to the small bit width.

In most of the reported hardware designs [8]–[10] with a medium security level, the R-LWE parameter set ($n = 256, q = 7681$, and $s = 11.31$) is used. For modular q reduction, [11] introduces an algorithm that uses shift, add, and subtract operations to accomplish the modular reduction. For the noise, a zero-centered discrete Gaussian distribution (such as $\boldsymbol{r_2}$) with a standard deviation of $s/\sqrt{2\pi} = 4.51$ is considered. On the modular ring, Gaussian distribution is shown in Fig. 1. As most of the lattice-based cryptosystems in NIST PQC Round 2 candidates require Gaussian or binomial distribution, the proposed methods in Section III can be extended to such distributions that have a bounded interval around 0.

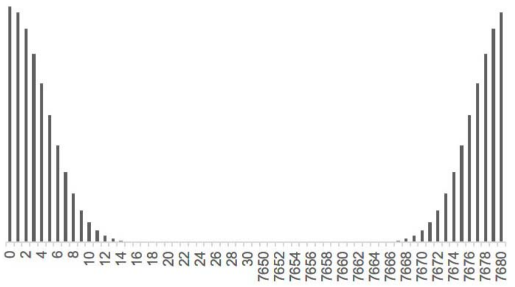


Fig. 1. Discrete Gaussian function distribution ($q = 7681$ and $\sigma = 4.51$).

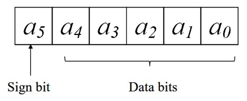


Fig. 2. Sign bit and data bits for reduced bit-width representation.

Algorithm 2 Novel Modular Reduction for $q = 7681$

```
Input:
    $x$, 17-bit unsigned integer;
    $q$, prime modular;
Output:
    $y$, 13-bit unsigned integer;
1:  $t \leftarrow x[17] + x[17:13]$;
2:  $tq \leftarrow (t << 13) + t - (t << 9)$;
3:  $y \leftarrow x - tq$;
4:  if $y \geq q$ then
5:      $y \leftarrow y - q$;
6:  end if
7:  if $y \geq q$ then
8:      $y \leftarrow y - q$;
9:  end if
10: return $y$
```

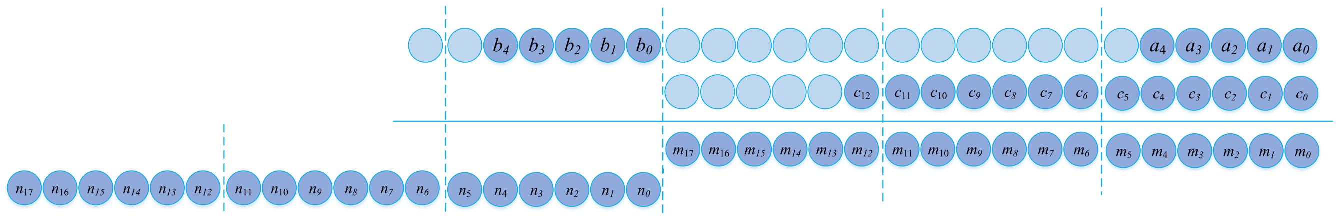
III. Proposed Optimized Polynomial Multiplication

This section proposes two novel techniques for efficient hardware implementation of R-LWE encryption/decryption modules.

A. Reduced Bit Width Due to the Noise Distribution Symmetry

The discrete Gaussian noise distribution, as shown in Fig. 1, is naturally symmetrical between $[0, q-1]$. Without loss of generality, for $\sigma = 4.51$, the number range is limited to $[0, 31]$ and $[-31, -1]$ (i.e., $[7650, 7680]$ on the modular integer ring if presented as an unsigned number). Opting for signed number representation instead of naïve 13-bit representation can save hardware resources. For the required number range, one sign bit and five data bits (6 bits in total) are enough to represent the input data instead of a 13-bit unsigned representation, as shown in Fig. 2. This proposed reduced bit-width technique can be applied to all polynomial multiplications in various R-LWE-based PKE schemes.

The reduced signed representation reduces the data for multiplication as well as memory accesses, and the modular operation described



b_4 b_3 b_2 b_1 b_0
a_4 a_3 a_2 a_1 a_0
c_{12} c_{11} c_{10} c_9 c_8 c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0
m_{17} m_{16} m_{15} m_{14} m_{13} m_{12} m_{11} m_{10} m_9 m_8 m_7 m_6 m_5 m_4 m_3 m_2 m_1 m_0
n_{17} n_{16} n_{15} n_{14} n_{13} n_{12} n_{11} n_{10} n_9 n_8 n_7 n_6 n_5 n_4 n_3 n_2 n_1 n_0

Fig. 3. Block diagram of two multiplications in the DSP block.

Algorithm 3 Two Multiplications Within One DSP Block

```
Input:
    $a$, 5-bit unsigned integer;
    $b$, 5-bit unsigned integer;
    $c$, 13-bit unsigned integer;
Output:
    $m$, 18-bit unsigned integer;
    $n$, 18-bit unsigned integer;
1: $tmp\_ab \leftarrow \{b, 13'd0, a\}$;
2: $tmp \leftarrow tmp\_ab \times c$;
3: $m \leftarrow tmp[17:0]$;
4: $n \leftarrow tmp[35:18]$;
5: return $m, n$
```

TABLE II
HARDWARE IMPLEMENTATION RESULTS OF DIFFERENT SPMA DESIGNS

Type	LUT/FF/Slice /BRAM/DSP	Freq (MHz)	Cycle	Throughput (Kbps)	Throughput /Slice
SPMA-1	277/266/112/0/1	290.61	65548	1135.0	10.13
SPMA-2	244/245/95/0/1	305.44	65548	1192.9	12.56
SPMA-3	317/198/108/0/1	333.00	34177	2494.3	23.10

TABLE III
ENS ON FPGA

Type	Slice	DSP	BRAM(8K)	BRAM(18K)	BRAM(36K)
#Slice	1	128	70	166	327
Weight	1.0	0.8	0.8	0.7	0.6
ENS.	1	102.4	56	116.2	196.2

in Algorithm 2 is also simplified. The multiplication product width is reduced from 26 bits (13 bit × 13 bit) to 17 bits (13 bit × 5 bit), as the sign bit is not used during the modular multiplication. The sign bit is used for number inversion, as shown in line 7 of Algorithm 1. Compared to the original modular reduction in [11], it saves one addition, one multiplexer, and one subtraction. In line 2 of Algorithm 2, tq is the product of t multiplied q. In line 3, y is an approximate modular result, which requires extra subtractions. Furthermore, the reduced bit-width multiplication makes it possible to perform two multiplications on a single DSP block in FPGA, which will be further discussed in Section III-B.

B. Full Utilization of FPGA DSP Blocks

In Xilinx 7 series FPGAs, a single DSP block can support a 25 × 18 bit multiplication. For R-LWE, due to the reduced size

Algorithm 4 Optimized Schoolbook PMA for Encryption or Decryption

```
Input:
    $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, (polynomial in $\mathbb{Z}_q[x]/(x^n+1)$);
    $q$, prime modular;
Output:
    $\boldsymbol{d}$, (polynomial in $\mathbb{Z}_q[x]/(x^n+1)$);
1: for $i = 0:2:n-2$ do
2:     $sum1 \leftarrow c[i]$;
3:     $sum2 \leftarrow c[i+1]$;
4:     for $j = 0:n-1$ do
5:         $tmp\_a \leftarrow a[j]$;
6:         $tmp\_b1 \leftarrow b[(i-j)\ mod\ n]$;
7:         $tmp\_b2 \leftarrow b[(i-j+1)\ mod\ n]$;
8:         $tmp\_b \leftarrow (tmp\_b1 << 18) + tmp\_b2$;
9:         $ab \leftarrow tmp\_a \times tmp\_b$;
10:        $ab\_m1 \leftarrow ab[17:0]\ mod\ q$;
11:        $ab\_m2 \leftarrow ab[35:18]\ mod\ q$;
12:        $sig1 \leftarrow (i < j)\ ?\ 1 : 0$;
13:        $sig2 \leftarrow (i+1 < j)\ ?\ 1 : 0$;
14:        if $sig1 \oplus tmp\_b1.sign()$ then
15:            $ab\_m1 \leftarrow q - ab\_m1$;
16:        end if
17:        if $sig2 \oplus tmp\_b2.sign()$ then
18:            $ab\_m2 \leftarrow q - ab\_m2$;
19:        end if
20:        $tmp1 \leftarrow sum1 + ab\_m1$;
21:        $tmp2 \leftarrow sum2 + ab\_m2$;
22:        $sum1 \leftarrow tmp1\ mod\ q$;
23:        $sum2 \leftarrow tmp2\ mod\ q$;
24:    end for
25:    $d[i] \leftarrow sum1$;
26:    $d[i+1] \leftarrow sum2$;
27: end for
28: return $\boldsymbol{d}$
```

of the multiplication required (13 × 5), we can efficiently pack two multiplicands to perform two multiplications using one DSP block on the FPGA. This bit packing is elaborated in Algorithm 3, where two multiplications $m = a \times c$ and $n = b \times c$ are depicted. First, in line 1, a and b are concatenated with 13 inserted zeros in the middle to form a new multiplicand tmp_ab. tmp_ab is 23 bits in size, where the first 5 bits are b, the last 5 bits are a, and with 13 zeros in the middle. Then, in line 2, a 23×13 multiplication is carried out. In lines 3 and 4, the results m and n are separated out for two parallel multiplications. The whole process is presented in detail in Fig. 3. The product of $a \times c$ is an 18-bit result, unrelated to the product of $b \times c$. This packing enables two simultaneous multiplications via one DSP slice per cycle.

TABLE IV
COMPARISONS WITH OTHER R-LWE DESIGNS

Implementation	Device	Type	LUT/FF/Slice	DSP	BRAM(18K)/(8K)	MHz	Cycle	Time(μs)	Throughput(Kbps)	ENS	Efficiency
This Work (Schoolbook)	Kintex-7	Enc	898/815/303	1	0/3	304.69	69654	229	1119.84	573.4	1.95
		Dec	635/190/194	1	0/1	303.40	34436	114	2255.49	352.4	6.40
[6] (Schoolbook)	Spartan-6	Enc	360/290/114	1	0/2	128	136986	1070	239.21	328.4	0.73
		Dec	162/136/51	1	0/1	179	66304	370	691.12	209.4	3.30
[8] (NTT)	Virtex-6	Enc	4549/3624/1506	1	12/0	262	6861	26	9775.83	3002.8	3.26
		Dec	4549/3624/1506	1	12/0	262	4404	17	15229.79	3002.8	5.07
[9] (NTT)	Virtex-6	Enc	1349/860/410	1	2/0	313	6300	20	12718.73	744.8	17.08
		Dec	1349/860/410	1	2/0	313	2800	9	28617.14	744.8	38.42
[10] (NTT)	Spartan-6	Enc	1307/889/406	0	1/3	80	360.5k	4500	56.81	690.2	0.08
		Dec	1307/889/406	0	0/1	80	72.0k	900	284.44	462.0	0.62

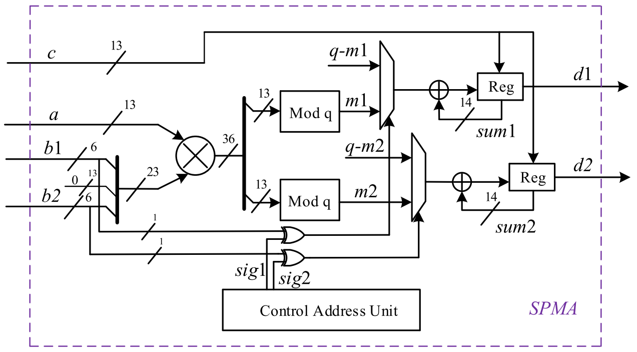


Fig. 4. Hardware structure of the optimized SPMA.

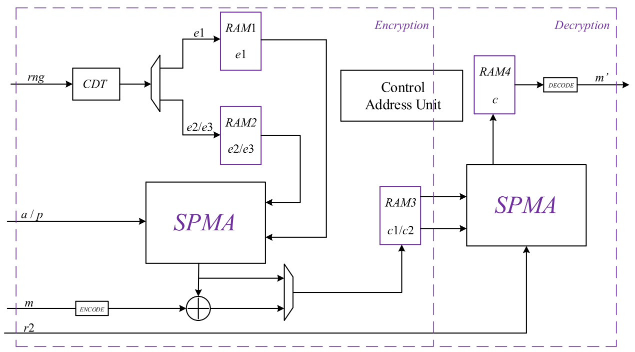


Fig. 5. Overall hardware structure of R-LWE scheme (the details of SPMA is provided in Fig. 4).

This trick can be extended to newer Xilinx FPGA families (including Ultrascale and Ultrascale+), which come with DSP multiplier slices of similar or wider dimensions, e.g., 27×18 multiplier in Ultrascale+.

The optimized schoolbook PMA, presented in Algorithm 4, uses both optimization techniques. First, it offers reduced bit-width representation to save hardware resources, which simplifies the modular reduction and reduces the critical path delay. In Algorithm 4, the polynomial $\boldsymbol{b}$ elements are the discrete Gaussian distribution samples, each of length 5 bits. Second, by employing full utilization of DSP blocks, the system carries out two multiplications, boosting performance without extra DSP resources. In lines 14 and 17, sign() denotes the MSB of the signed number (sig1/sig2 $=1$ for negative number). For negative numbers, result ab_m should be subtracted from the modulus q.

IV. HARDWARE IMPLEMENTATION RESULTS

A. Hardware Design Structure

The high-level hardware block diagram of the optimized SPMA is shown in Fig. 4. There are four input data from the BRAMs for storing the three polynomials $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$, in which polynomial $\boldsymbol{b}$ allows two parallel accesses per clock cycle. Right after the multiplication, the data split into two parallel pipelined parts. Then, modular reduction operations follow next. The control signals "sig1" XOR "$b1$.sign()" and "sig2" XOR "$b2$.sign()" are used to determine the sign of accumulated data. Finally, results $d1$ and $d2$ are written to the BRAMs. BRAMs are controlled by the control address unit.

B. SPMA Performance Results

The designs are synthesized and implemented using Xilinx Vivado 2016.4 targeting a Kintex-7 FPGA (KC705) and post-place and route results are presented in Table II. SPMA-1 refers to the naïve design with no optimizations, as described in Algorithm 1. SPMA-2 exploits the reduced bit-width technique, while SPMA-3 additionally uses the DSP bit packing technique. The SPMA-2 design requires around 15.2% fewer FPGA resources and achieves a higher operating frequency. The SPMA-3 design achieves the highest throughput due to two reasons: first, due to a reduction in the critical path, enabling the highest operating frequency, and second, SPMA-3 almost halves the computation cycles required and consequently achieves twice the speedup compared with SPMA-1. The efficiency (denoted as throughput per slice) of SPMA-3 is $2.28\times$ compared with SPMA-1.

C. R-LWE Cryptography Implementation Results

In the context of R-LWE-based PKE, SPMA-3 can be used in all three modules: key generation, encryption, and decryption. The encryption module consists of three Gaussian samplers, two SPMA-3, and one polynomial addition. The implementation of the Gaussian sampler is based on the cumulative distribution table (CDT) sampling design, which resists the threat of timing attacks by inherently running in constant time [12]. The overall R-LWE cryptosystem hardware block diagram is shown in Fig. 5.

To ensure a fair comparison of our optimized SPMA-3-based R-LWE design with the earlier reported FPGA implementations (on

Spartan and Virtex families), the following method of equivalence conversion is proposed for design evaluation. We first convert the DSP blocks and BRAMs used in a given design into an equivalent number of slices (ENS). For Xilinx 7 series, a single DSP block can be replaced by 128 slices for a 25×18 multiplier using the built-in IP core. But not every design fully uses the DSP block, therefore, the weight of 0.8 is assigned to the DSP block (one DSP block $= 128 \times 0.8 = 102.4$) slices. Similarly, each BRAM (18k) can be substituted for 116.2 (166×0.7) slices and BRAM (8k) can be replaced by 56 (70×0.8) slices. The BRAMs are reconstituted by the slice memory using two dual-ported RAM modes. Table III shows the detailed ENS on FPGA.

Table IV compares the proposed design with the previous R-LWE implementations using the same parameter set ($n = 256, q = 7681$, and $s = 11.31$) except [6]. The design in [6] also uses SPMAs, but it has lower frequency and throughput, and its efficiency is much lower than the proposed design. Furthermore, it uses a parameter set of (256, 4093, and 8.35), which is considered to be less secure compared with other designs in Table IV. The latest design [10] claims resistance against timing attack due to the usage of CDT-based noise sampling. However, it is much slower than other hardware designs. Pöppelmann and Güneysu [8] proposed a fast R-LWE cryptographic processor at the cost of substantially more resource. The most efficient designs [9] only use NTT (without inverse NTT) for encryption and only inverse NTT for decryption. Meanwhile, NTT computation requires the computation of twiddle factors, which requires the RAM storage when precomputed. However, these RAMs have not been included for comparison. As mentioned in Section I, half of the 12 LBC contestants in round 2 of the NIST PQC initiative do not use NTT.

Due to the two proposed novel techniques for optimized SPMA, we achieve an efficient design for R-LWE encryption and decryption. Our design only requires 69 654 clock cycles (0.229 ms) for encryption and 34 436 clock cycles (0.114 ms) for decryption, which makes our proposed design the optimal choice for resource-constrained devices, achieving both high hardware efficiency and performance.

V. Conclusion

This brief proposes novel optimizations for the most computationally intensive part of LBC constructions, i.e., the polynomial multiplier, targeting the high-speed FPGA platform. We exploit the noise distribution symmetry to reduce the dynamic range and reduced bit width of the discrete Gaussian data samples. This simplification also leads to smart packing of data and the full utilization of the DSP block to gain a $2\times$ speedup.

References

[1] L. Chen *et al.*, “Post-quantum cryptography,” U.S. Dept. Commerce, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NISTIR 8105, 2016.

[2] NIST. *PQC Round 2*. Accessed: Feb. 2019. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

[3] F. Valencia, A. Khalid, E. O’Sullivan, and F. Regazzoni, “The design space of the number theoretic transform: A survey,” in *Proc. Int. Conf. Embedded Comput. Syst. Archit., Modeling, Simul. (SAMOS)*, Jul. 2017, pp. 273–277.

[4] T. Pöppelmann and T. Güneysu, “Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware,” in *Proc. Int. Conf. Cryptol. Inf. Secur. Latin America*, 2012, pp. 139–158.

[5] C. Du and G. Bai, “Towards efficient polynomial multiplication for lattice-based cryptography,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1178–1181.

[6] 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.

[7] J. Howe, C. Rafferty, A. Khalid, and M. O’Neill, “Compact and provably secure lattice-based signatures in hardware,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[8] T. Pöppelmann and T. Güneysu, “Towards practical lattice-based public-key encryption on reconfigurable hardware,” in *Proc. Int. Conf. Sel. Areas Cryptogr.*, 2013, pp. 68–85.

[9] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, “Compact ring-LWE cryptoprocessor,” in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2014, pp. 371–391.

[10] D. Liu, C. Zhang, H. Lin, Y. Chen, and M. Zhang, “A resource-efficient and side-channel secure hardware implementation of ring-LWE cryptographic processor,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 4, pp. 1474–1483, Apr. 2018.

[11] 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. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2015, pp. 663–682.

[12] A. Khalid, J. Howe, C. Rafferty, and M. O’Neill, “Time-independent discrete Gaussian sampling for post-quantum cryptography,” in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2016, pp. 241–244.