# Optimised Schoolbook Polynomial Multiplication for Compact Lattice-based Cryptography on FPGA

Liu, W., Fan, S., Khalid, A., Rafferty, C., & O'Neill, M. (2019). Optimised Schoolbook Polynomial Multiplication for Compact Lattice-based Cryptography on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1-5. https://doi.org/10.1109/TVLSI.2019.2922999

**Published in:**
IEEE Transactions on Very Large Scale Integration (VLSI) Systems

**Document Version:**
Peer reviewed version

**Queen's University Belfast - Research Portal:**
Link to publication record in Queen's University Belfast Research Portal

**Publisher rights**
Copyright 2019 IEEE. This work is made available online in accordance with the publisher's policies. Please refer to any applicable terms of use of the publisher.

**General rights**
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

**Take down policy**
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.

Download date:28. Aug. 2022

# Optimised Schoolbook Polynomial Multiplication for Compact Lattice-based Cryptography on FPGA

Weiqiang Liu, *Senior Member, IEEE,* Sailong Fan, Ayesha Khalid, *Member, IEEE,* Ciara Rafferty, *Member, IEEE,* Máire O'Neill, *Senior Member, IEEE*

***Abstract*—Lattice-based cryptography (LBC) is one of the most promising classes of post-quantum cryptography (PQC) that is being considered for standardisation. This paper proposes an optimised schoolbook polynomial multiplication for compact LBC. We exploit the symmetric nature of Gaussian noise for bit reduction. Additionally, a single FPGA DSP block is used for two parallel multiplication operations per clock cycle. These optimisations 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 schoolbook polynomial multiplication, as well as contributing to a more compact LBC system as compared to previously reported designs. The results targeting the FPGA platform show that the proposed design can achieve both high hardware efficiency with reduced hardware area costs.**

***Index Terms*—Lattice-based cryptography (LBC); polynomial multiplication; FPGA**

## 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. Amongst the potential PQC algorithms to be standardised, lattice-based cryptography (LBC) is one of the most promising types. Almost half of the PQC candidates in Round 2 of the PQC standardisation 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, 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]. [4] proposes adaptable and extensible hardware implementations of both NTT and SPM methods supporting various operand sizes. Working towards 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 optimisation techniques for SPM hardware. Moreover, [7] employed SPM in LBC for digital signatures on FPGAs. Until now, little research has been conducted in terms of a thorough trade-off 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 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 research proposes a compact and efficient hardware design for R-LWE encryption/decryption based on schoolbook polynomial multiplication. We exploit the distribution symmetry of Gaussian noise to achieve a *reduced bit-width* and *full utilisation of DSP blocks*. A compact SPM is designed with approximately 2× speedup without additional hardware resource consumption. A comparison with existing R-LWE designs is provided, which highlights the efficiency of our proposed design. The proposed design optimisations can also be undertaken for other LBC schemes and other FPGA families.

## 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 resource requirements for R-LWE encryption and decryption are mainly due to the SPM. $D_\sigma$ is the Gaussian distribution with standard deviation $\sigma$, and $U$ is the uniform distribution. Polynomials $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are the ciphertext results. The decryption procedure also performs polynomial

This work is supported by grants from the NSFC (61871216), the Fundamental Research Funds for the Central Universities China (No. NE2019102) and the Six Talent Peaks Project in Jiangsu Province (2018-XYDXX-009).

W. Liu, and S. Fan are with College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: {liuweiqiang, 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, UK. E-mail: {a.khalid, c.m.rafferty}@qub.ac.uk, m.oneill@ecit.qub.ac.uk.

multiplication and addition, and *DECODE*s 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.

TABLE I
THE 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 $\boldsymbol{c_1} = \boldsymbol{ae_1} + \boldsymbol{e_2}$, $\boldsymbol{c_2} = \boldsymbol{pe_1} + \boldsymbol{e_3} + \bar{\boldsymbol{m}}$. |
| $Dec(\boldsymbol{c_1}, \boldsymbol{c_2})$ | $m' = DECODE(\boldsymbol{c} = \boldsymbol{c_1 r_2} + \boldsymbol{c_2})$. |

The typical SPM algorithm used in R-LWE can be expressed as Eq. (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 this method has $O(n^2)$ complexity.

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

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 (denoted as PMA) on FPGA as shown in Alg. 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 ($n = 256, q = 7681, s = 11.31$) 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 centred discrete Gaussian distribution (like $\boldsymbol{r_2}$) 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 cryptosystem 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.

## III. THE PROPOSED OPTIMISED POLYNOMIAL MULTIPLICATION

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

**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** $\boldsymbol{d}$

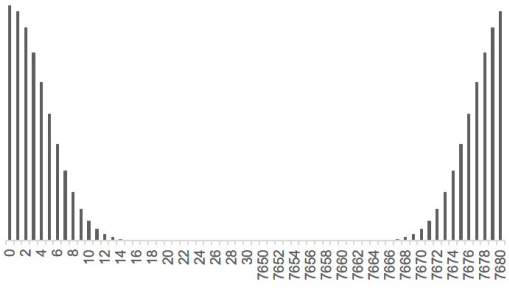


Fig. 1. Discrete Gaussian function distribution ($q = 7681$ and $\sigma = 4.51$).

### 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, 1 sign bit and 5 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.

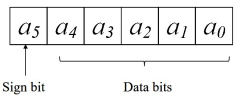


Fig. 2. The sign bit and data bits for reduced bit width representation.

The *reduced signed representation* reduces the data for multiplication as well as memory accesses, and the mod-

ular operation described in Alg. 2 is also simplified. The multiplication product width is reduced from 26 bits (13-bit $\times$ 13-bit) to 17 bits (13-bit $\times$ 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 Alg. 1. Compared to the original modular reduction in [11], it saves one addition, one multiplexer and one subtraction. In line 2 of Alg. 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 the next subsection.

**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** $y$

### B. Full Utilisation of FPGA DSP Blocks

In Xilinx 7 series FPGAs, a single DSP block can support a $25 \times 18$ bit multiplication. For R-LWE, due to the reduced size of the multiplication required ($13 \times 5$) we can efficiently pack two multiplicands to perform two multiplications using one DSP block on the FPGA. This bit packing is elaborated in Alg. 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 $tmp\_ab$. The $tmp\_ab$ 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 $23 \times 13$ multiplication is carried out. In lines 3-4, 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. This trick can be extended to newer Xilinx FPGA families (including Ultrascale and Ultrascale+), that come with DSP multiplier slices of similar or wider dimensions, e.g., $27 \times 18$ multiplier in Ultrascale+.

The optimised schoolbook PMA, presented in Alg. 4, uses both optimisation techniques. Firstly it offers *reduced bit-width* representation to save hardware resources which simplifies the modular reduction and reduces the critical path delay. In

**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** $m, n$

Alg. 4, the polynomial $\boldsymbol{b}$ elements are the discrete Gaussian distribution samples, each of length 5-bits. Secondly, by employing *full utilisation of DSP blocks*, the system carries out two multiplications, boosting performance without extra DSP resources. In line 14 and line 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 optimised 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 splits 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 the results $d1$ and $d2$ are written to the BRAMs. BRAMs are controlled by the Control Address Unit.

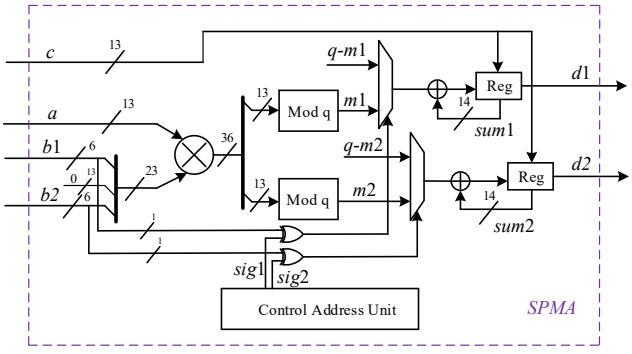


Fig. 4. Hardware structure of the optimised SPMA.

### 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 a naïve design with no optimisations as

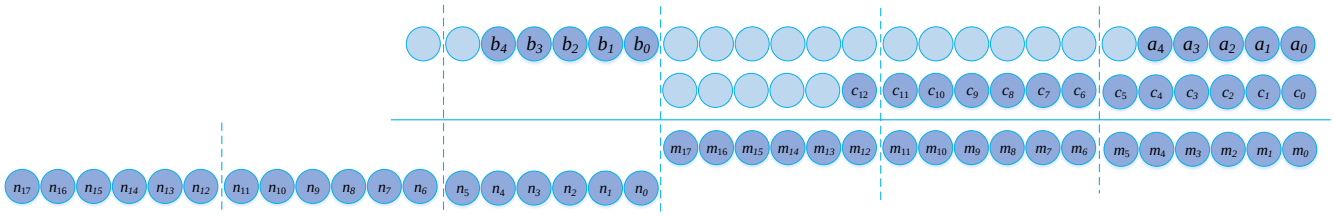


Fig. 3. The block diagram of two multiplications in DSP block.

**Algorithm 4** The Optimised 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 $\mathbb{Z}_q[x]/(x^n+1)$);

**for** $i = 0:2:n-2$ **do**
  $sum1 \leftarrow c[i]$;
  $sum2 \leftarrow c[i+1]$;
  **for** $j = 0:n-1$ **do**
    $tmp\_a \leftarrow a[j]$;
    $tmp\_b1 \leftarrow b[(i-j)\ mod\ n]$;
    $tmp\_b2 \leftarrow b[(i-j+1)\ mod\ n]$;
    $tmp\_b \leftarrow (tmp\_b1 << 18) + tmp\_b2$;
    $ab \leftarrow tmp\_a \times tmp\_b$;
    $ab\_m1 \leftarrow ab[17:0]\ mod\ q$;
    $ab\_m2 \leftarrow ab[35:18]\ mod\ q$;
    $sig1 \leftarrow (i < j)\ ?\ 1 : 0$;
    $sig2 \leftarrow (i+1 < j)\ ?\ 1 : 0$;
    **if** $sig1 \oplus tmp\_b1.sign()$ **then**
      $ab\_m1 \leftarrow q - ab\_m1$;
    **end if**
    **if** $sig2 \oplus tmp\_b2.sign()$ **then**
      $ab\_m2 \leftarrow q - ab\_m2$;
    **end if**
    $tmp1 \leftarrow sum1 + ab\_m1$;
    $tmp2 \leftarrow sum2 + ab\_m2$;
    $sum1 \leftarrow tmp1\ mod\ q$;
    $sum2 \leftarrow tmp2\ mod\ q$;
  **end for**
  $d[i] \leftarrow sum1$;
  $d[i+1] \leftarrow sum2$;
**end for**
**return** $\boldsymbol{d}$

described in Alg. 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% less FPGA resources and achieves a higher operating frequency. The SPMA-3 design achieves the highest throughput due to two reasons: firstly, due to a reduction in the critical path, enabling the highest operating frequency, and secondly, SPMA-3 almost halves the computation cycles required and consequently achieves twice the speedup compared to SPMA-1. The efficiency (denoted as throughput per slice) of SPMA-3 is 2.28× compared with SPMA-1.

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 |

### C. R-LWE Cryptography Implementation Results

TABLE III
EQUIVALENT NUMBER OF SLICES (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 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 3 Gaussian samplers, 2 SPMA-3 and one polynomial addition. The implementation of the Gaussian sampler is based on 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 presented in Fig. 5.

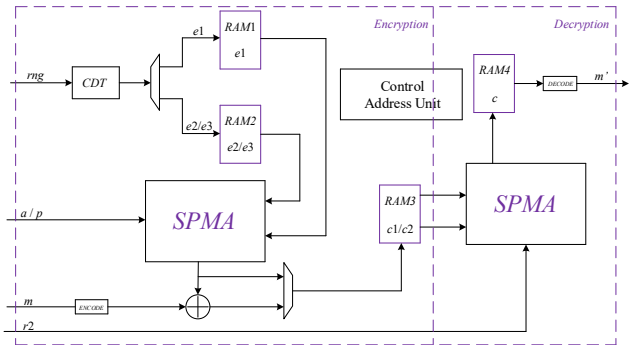


Fig. 5. Overall hardware structure of R-LWE scheme (The details of SPMA is provided in Fig. 4).

To ensure a fair comparison of our optimised SPMA-3 based R-LWE design with earlier reported FPGA implementations (on Spartan and Virtex families), the following method of equivalence conversion is proposed for design evaluation. We

TABLE IV
COMPARISONS WITH OTHER R-LWE DESIGNS

| Implementation | Device | Type | LUT/FF/Slice | DSP | BRAM(18K)/(8K) | MHz | Cycle | Time($\mu$s) | 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 |

first convert the DSP blocks and BRAMs used in a given design into an equivalent number of slices. For Xilinx 7 series, a single DSP block can be replaced by $128$ slices for a $25\times18$ multiplier using the built-in IP core. But not every design fully uses the DSP block, so weight of $0.8$ is assigned to the DSP block (1 DSP block $= 128 \times 0.8 = 102.4$) slices. Similarly, each BRAM(18K) can be substituted for $116.2$ ($166 \times 0.7$) slices and BRAM(8K) can be replaced by $56$ ($70 \times 0.8$) slices. The BRAMs are reconstituted by the slice memory using two dual-ported RAM mode. Table III shows the detailed equivalent number of slices (defined as ENS) on FPGA.

Table IV compares the proposed design with previous R-LWE implementations, using the same parameter set ($n = 256, q = 7681, s = 11.31$) 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, 8.35) which is considered to be less secure compared with other designs in the table. 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. [8] proposes a fast R-LWE cryptographic processor at the cost of substantially more resource. The most efficient design [9] only use NTT (without inverse NTT) for encryption and only inverse NTT for decryption. Meanwhile, NTT computation requires computation of twiddle factors, which requires the RAM storage when precomputed. However, these RAMs have not been included for comparison. As mentioned in the introduction section, 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 optimised SPMA, we achieve 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. CONCLUSIONS

This research proposes novel optimisations for the most computationally intensive part of lattice-based cryptography 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 utilisation of the DSP block* to gain a $2\times$ speedup.

## REFERENCES


[1] NIST, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[2] NIST, "PQC round 2." https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions. Last accessed Feb. 2019.

[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 Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pp. 273–277, 2017.

[4] T. Pöppelmann and T. Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware," in *Proc. Int. Conf. Cryptology and Information Security in Latin America*, pp. 139–158, 2012.

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

[6] T. Poppelmann and T. Guneysu, "Area optimization of lightweight lattice-based encryption on reconfigurable hardware," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, pp. 2796–2799, 2014.

[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 and Systems (ISCAS)*, pp. 1–4, 2017.

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

[9] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-lwe cryptoprocessor," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 371–391, 2014.

[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 and Systems I: Regular Papers*, pp. 1–10, 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. Cryptographic Hardware and Embedded Systems (CHES)*, pp. 663–682, 2015.

[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-Programmable Technology (FPT)*, pp. 241–244, 2016.