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


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: 23. Aug. 2019
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 $2.2 \times$ speedup along with reduced resources for dimension $n = 256$. The overall efficiency (throughput per slice) is $1.28 \times$ 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 $2 \times$ 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. $D_S$ is the Gaussian distribution with standard deviation $\sigma$, and $U$ is the uniform distribution. Polynomials $c_1$ and $c_2$ are the ciphertext results. The decryption procedure also performs polynomial...
multiplication and addition, and \textit{DECODE} the polynomial to plaintext. Please refer to [3] 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 Eq. (1)\cite{4}. Considering the property of polynomial multiplication that \(x^n = -1\), 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 \((-1)^{(i+j)/n}\). 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.

\[
c = ab = \left[ \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j x^{i+j} \right] \mod (x^n + 1)
\]

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 \(d\) both polynomial multiplication and addition (denoted as PMA) and encryption/decryption performance of R-LWE, we implement 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.

#### TABLE I

<table>
<thead>
<tr>
<th>Polynomial (a \leftarrow U) and (p) are the public key, (m_2) is the secret key.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1, c_2, c_3 \leftarrow D_0). Let (m = ENCODE(m)). Then the cipher text is (c_1 = a_1 + c_1) and (c_2 = p_{c1} + c_3 + m).</td>
</tr>
<tr>
<td>(Dec(c_1, c_2)) (m' = DECODE(c = c_1 r_2 + c_2)).</td>
</tr>
</tbody>
</table>

The typical SPM algorithm used in R-LWE can be expressed as Eq. (1)\cite{4}. Considering the property of polynomial multiplication that \(x^n = -1\), 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 \((-1)^{(i+j)/n}\). 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.

\[
c = ab = \left[ \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j x^{i+j} \right] \mod (x^n + 1)
\]

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 \(d\) both polynomial multiplication and addition (denoted as PMA) and encryption/decryption performance of R-LWE, we implement the proposed methods in Section III can be extended to such distributions that have a bounded interval around 0.

### 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. 1. Discrete Gaussian function distribution (q = 7681 and σ = 4.51).](image)

![Fig. 2. The sign bit and data bits for reduced bit width representation.](image)
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. 2]. Compared to the original modular reduction in [1], 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.

\textbf{Algorithm 2} Novel Modular Reduction for \( q = 7681 \)

\textbf{Input:}
\begin{itemize}
  \item \( x \), 17-bit unsigned integer;
  \item \( q \), prime modular;
\end{itemize}

\textbf{Output:}
\begin{itemize}
  \item \( y \), 13-bit unsigned integer;
\end{itemize}

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

\textbf{Algorithm 3} Two Multiplications within One DSP Block

\textbf{Input:}
\begin{itemize}
  \item \( a \), 5-bit unsigned integer;
  \item \( b \), 5-bit unsigned integer;
  \item \( c \), 13-bit unsigned integer;
\end{itemize}

\textbf{Output:}
\begin{itemize}
  \item \( m \), 18-bit unsigned integer;
  \item \( n \), 18-bit unsigned integer;
\end{itemize}

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 \( m, n \)

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 \( a, b \) and \( c \), in which polynomial \( b \) 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.

\textbf{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 naive design with no optimisations as
Decryption

Algorithm 4 The Optimised Schoolbook PMA for Encryption or Decryption

Input:
- \(a, b, c\), (polynomial in \(\mathbb{Z}_q[x]/(x^n + 1)\));
- \(q\), prime modular;

Output:
- \(d\), (polynomial in \(\mathbb{Z}_q[x]/(x^n + 1)\));

1: for \(i = 0 : 2 : n - 2\) do
  2: \(\text{sum1} \leftarrow c[i];\)
  3: \(\text{sum2} \leftarrow c[i + 1];\)
  4: for \(j = 0 : n - 1\) do
    5: \(\text{tmp}_a \leftarrow a[j];\)
    6: \(\text{tmp}_b1 \leftarrow b[\lfloor i - j \rfloor \mod n];\)
    7: \(\text{tmp}_b2 \leftarrow b[\lfloor i - j + 1 \rfloor \mod n];\)
    8: \(\text{tmp}_b \leftarrow (\text{tmp}_b1 < 18) + \text{tmp}_b2;\)
    9: \(ab \leftarrow \text{tmp}_a \times \text{tmp}_b;\)
   10: \(\text{ab}_n1 \leftarrow ab[17 : 0] \mod q;\)
   11: \(\text{ab}_n2 \leftarrow ab[35 : 18] \mod q;\)
   12: \(\text{sig}1 \leftarrow \lfloor i < j \rfloor ? 1 : 0;\)
   13: \(\text{sig}2 \leftarrow \lfloor i + 1 < j \rfloor ? 1 : 0;\)
  14: if \(\text{sig}1 \oplus \text{tmp}_b1.\text{sign}()\) then
    15: \(\text{ab}_m1 \leftarrow q - \text{ab}_n1;\)
  16: end if
  17: if \(\text{sig}2 \oplus \text{tmp}_b2.\text{sign}()\) then
    18: \(\text{ab}_m2 \leftarrow q - \text{ab}_n2;\)
  19: end if
  20: \(\text{tmp}_1 \leftarrow \text{sum1} + \text{ab}_m1;\)
  21: \(\text{tmp}_2 \leftarrow \text{sum2} + \text{ab}_m2;\)
  22: \(\text{sum1} \leftarrow \text{tmp1} \mod q;\)
  23: \(\text{sum2} \leftarrow \text{tmp2} \mod q;\)
  24: end for
25: \(d[i] \leftarrow \text{sum1};\)
26: \(d[i + 1] \leftarrow \text{sum2};\)
27: end for
28: return \(d\)

The Optimised Schoolbook PMA for Encryption

C. R-LWE Cryptography Implementation Results

TABLE II

<table>
<thead>
<tr>
<th>Type</th>
<th>LUT/FF/Slice</th>
<th>Freq (MHz)</th>
<th>Cycle</th>
<th>Throughput (Kbps)</th>
<th>Throughput/Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPMA-1</td>
<td>277/266/112</td>
<td>390.61</td>
<td>102.4</td>
<td>65548</td>
<td>1135.0</td>
</tr>
<tr>
<td>SPMA-2</td>
<td>244/245/95</td>
<td>305.44</td>
<td>65548</td>
<td>1192.9</td>
<td>12.56</td>
</tr>
<tr>
<td>SPMA-3</td>
<td>317/198/108</td>
<td>333.00</td>
<td>34177</td>
<td>2494.3</td>
<td>23.10</td>
</tr>
</tbody>
</table>

TABLE III

<table>
<thead>
<tr>
<th>Type</th>
<th>Slice</th>
<th>DSP</th>
<th>BRAM(8K)</th>
<th>BRAM(16K)</th>
<th>BRAM(36K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Slice</td>
<td>1</td>
<td>128</td>
<td>70</td>
<td>102.4</td>
<td>116.2</td>
</tr>
<tr>
<td>Weight</td>
<td>1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>ENS</td>
<td>1</td>
<td>102.4</td>
<td>56</td>
<td>102.4</td>
<td>196.2</td>
</tr>
</tbody>
</table>

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.

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
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 × 18 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 × 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 IV shows the detailed equivalent number of slices (defined as ENS) on FPGA.

Table IV compares the proposed design with previous R-LWE 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× speedup.

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

REFERENCES