electronics

MDPI

Article

High Efficiency Ring-LWE Cryptoprocessor Using Shared Arithmetic Components

Tuy Nguyen Tan [1], Tram Thi Bao Nguyen [2] and Hanho Lee [2,*]

[1] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City 758307, Vietnam; nguyentantuy@tdtu.edu.vn

[2] Department of Information and Communication Engineering, Inha University, Incheon 22212, Korea; baotram137@gmail.com

* Correspondence: hhlee@inha.ac.kr; Tel.: +82-32-860-7449

Received: 17 May 2020; Accepted: 28 June 2020; Published: 30 June 2020

Abstract: A high efficiency architecture for ring learning with errors (ring-LWE) cryptoprocessor using shared arithmetic components is presented in this paper. By applying a novel approach for sharing number theoretic transform (NTT) polynomial multiplier and polynomial adder in encryption and decryption operations, the total number of polynomial multipliers and polynomial adders used in the proposed ring-LWE cryptoprocessor are reduced. In addition, the processing time of NTT polynomial multiplier is speeded up by employing multiple-path delay feedback (MDF) architecture and deploying pipelined technique between all stages of NTT processes. As a result, the proposed architecture offers a great reduction in terms of the hardware complexity and computation latency compared with existing works. The implementation result for the proposed ring-LWE cryptoprocessor on Virtex-7 FPGA board using Xilinx VIVADO shows a significant decrease in the number of slices and LUTs compared with previous works. Moreover, the proposed ring-LWE cryptoprocessor offers higher throughput and efficiency than its predecessors.

Keywords: cryptoprocessor; pipelined; multiple-path delay feedback; ring-LWE; shared arithmetic components

1. Introduction

Cryptographic algorithms are grouped into two categories named symmetric algorithms and public key (or asymmetric) algorithms. The former uses a single key between two parties to enable a secure communication, where the key is kept private from all other parties. Symmetric algorithms are widely used because of its simplicity. However, a symmetric key algorithm requires an agreement between sender and receiver on the secret key. Asymmetric, or public key cryptosystems use two different keys called private keys and public keys. Whereas the private key is kept secret for the decryption process, the public key is used for encryption and can be revealed to all other parties. The encryption operation is conducted using a public key, and the encrypted message can only be decrypted using the corresponding private key. The security of Rivest, Shamir, and Adleman (RSA) cryptosystems [1] and elliptic curve cryptosystems (ECC) [2,3] are based on the difficulty of solving some number theoretic problems and the difficulty of solving the elliptic curve discrete logarithm problems, respectively. However, these problems can be solved by the algorithm proposed by Shor [4] in polynomial time with quantum computers. Therefore, stronger security systems or post-quantum cryptosystems have been proposed, and the National Institute of Standards and Technology (NIST) is standardizing them. Among lattice-based cryptosystems for the post-quantum era, ring learning with errors (LWE) is a promising candidate because its security proofs are based on the worst-case hardness of lattice problems that there is no known quantum algorithm can efficiently solve [5]. A typical block diagram of a typical

ring-LWE cryptoprocessor is shown in Figure 1. Input message m is encrypted into ciphertext (c_1, c_2) using arithmetic computation on public key (a, p) and error polynomials e_1, e_2, and e_3. Original message m can be recovered from ciphertext (c_1, c_2) and private key r_2 using the decryption operation.

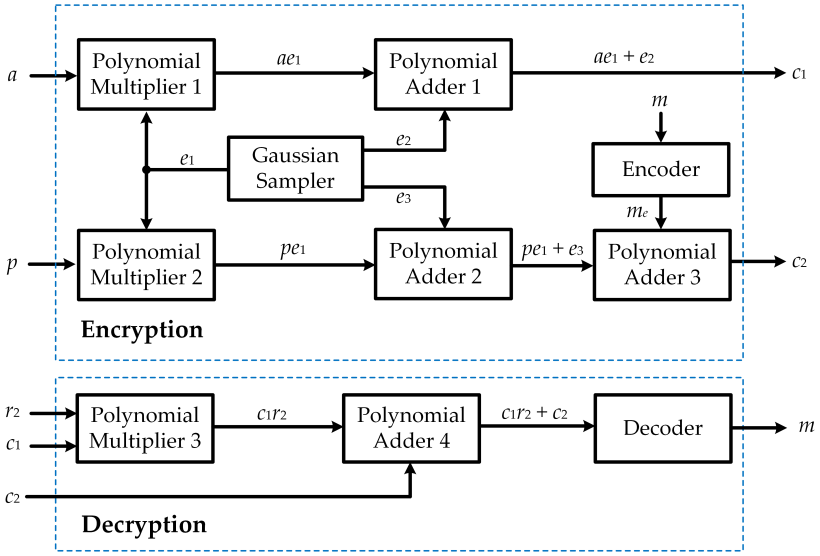

Figure 1. Block diagram of a typical ring-LWE cryptoprocessor [6].

The ring-LWE problem has been discussed in recent studies, both in software and hardware [5,7–9]. In Reference [5], high-throughput ring-LWE cryptoprocessors are designed to perform ring-LWE encryption and decryption operations. In Reference [7], an approach of integrating ring-LWE cryptography into existing fingerprint authentication systems to fully protect the fingerprint data are introduced. Authors in [9] present the implementation of ring-LWE encryption on IoT processors.

In the ring-LWE cryptosystem, lattices with an algebraic structure like polynomial multiplication and addition are performed over a polynomial ring, typically $R_q = \mathbb{Z}_q[x]/f(x)$. Among these operations, polynomial multiplication is the most complex one that can be efficiently performed using number theoretic transform (NTT)-based polynomial multiplication [5]. NTT multiplier is a modified version of fast Fourier transform (FFT) to work in a finite field without inaccurate floating point or complex arithmetic to compute polynomial multiplication efficiently [10]. There are several NTT multiplier architectures that deploy single-path delay feedback (SDF) or multiple path delay commutator (MDC) structures in literature. For example, a high throughput multiplier using NTT cores with radix-2 SDF architecture is presented in [11]. In Reference [5], authors introduce radix-2 and radix-8 MDC architecture-based NTT cores for ring-LWE cryptoprocessors to obtain the encryption throughput of gigabits per seconds and decryption throughput of megabits per second. However, these architectures require large hardware resources and high computation time since the NTT polynomial multipliers work separately and NTT operations are not fully optimized.

In this paper, a novel approach to efficiently use arithmetic components in ring-LWE cryptoprocessors to achieve a high efficiency is presented. Specifically, the polynomial multiplier and polynomial adder that participate in the ring-LWE encryption operation are reused in decryption phase to reduce hardware complexity. Additionally, the NTT polynomial multiplier is designed using MDF architecture and deploying pipeline technique among all stages of NTT and INTT transforms to mitigate hardware complexity and speed up multiplication operations. Our contributions of this article are summarized as follows:

- We propose a ring-LWE cryptoprocessor architecture in which the same arithmetic components, including one polynomial multiplier and one polynomial adder, are used in both encryption and decryption operations to reduce hardware complexity. As a result, the proposed ring-LWE cryptoprocessor requires less hardware resource than existing architectures to perform encryption and decryption operation.

- We deploy the polynomial multiplier using NTT multiplier with parallel based MDF architecture to enhance the polynomial multiplication. Furthermore, the pipeline technique is applied in the proposed design to reduce the system latency.
- We implement the proposed ring-LWE cryptoprocessor architecture on Xilinx Virtex-7 FPGA board and compare the obtained results with its predecessors. Performance evaluation results show that the proposed architecture offers a higher throughput and a better efficiency than others.

The remaining of this paper is structured as follows. In Section 2, brief discussions on the ring-LWE cryptosystems are carried out. Section 3 presents the proposed algorithm and architecture for ring-LWE cryptoprocessor. Section 4 provides the results of implementation and comparison, and Section 5 includes conclusions.

2. Background

2.1. Ring-LWE Cryptosystem

The ring-LWE problem introduced by Regev [12] in 2005 is a machine learning problem that is equivalent to the worst-case lattice problems. The ring-LWE cryptosystem operates over a ring $R_q = \mathbb{Z}_q[x]/f(x)$, where $f(x)$ is the irreducible polynomial of degree n, n is a power-of-two number, and q is a prime number such that $q \equiv 1 \bmod 2n$. The common case of irreducible polynomial is $f(x) = x^n + 1$ that is presented in [11].

Polynomial multiplication and polynomial addition are employed to carry out the cryptographic primitives of the ring-LWE cryptosystem. The procedures for key generation, encryption, and decryption of ring-LWE cryptosystem are described as follows:

Key generation: This process generates the public key (a, p) and the private key r_2. Polynomial a is chosen uniformly and two polynomials r_1 and r_2 are selected from the discrete Gaussian distribution χ_σ to compute the public key polynomial:

$$p \leftarrow r_1 - a \times r_2 \tag{1}$$

Encryption: The input message m is encoded to get the polynomial m_e. If the ith coefficient of m is 1, it is mapped to $(q-1)/2$; otherwise, it is converted to 0. The cipher-text c_1 and c_2 are computed from the given polynomials and three error polynomials e_1, e_2, and e_3 that sampled from the Gaussian distribution:

$$\begin{aligned} c_1 &\leftarrow a \times e_1 + e_2 \\ c_2 &\leftarrow p \times e_1 + e_3 + m_e \end{aligned} \tag{2}$$

Decryption: The input message m is recovered from the pre-decoded polynomial m_d:

$$m_d \leftarrow c_1 \times r_2 + c_2 \tag{3}$$

Depending on the value of each coefficient of the message m_d, the decoder maps it to a corresponding binary bit to recover the original message m.

2.2. Arithmetic Operations over Ring

The arithmetic operations over the ring $R_q = \mathbb{Z}_q[x]/f(x)$ include modulo reduction, polynomial addition, and polynomial multiplication. Given coefficients a_i and b_i in R_q, two polynomials $a(x)$ and $b(x)$ over the ring can be defined as follows:

$$\begin{aligned} a(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} \\ b(x) &= b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1} x^{n-1} \end{aligned} \tag{4}$$

As mentioned previously, the polynomial multiplication requires the most processing time, and the NTT-based algorithm is an efficient algorithm for performing this multiplication. Assume that ω is a primitive nth root of unity, the NTT of each coefficient of $a(x)$ is defined as:

$$A_i = \sum_{j=0}^{n-1} a_j \omega^{ij} \mod p \tag{5}$$

The inverse NTT (INTT) is calculated as:

$$a_i = n^{-1} \sum_{j=0}^{n-1} A_j \omega^{-ij} \mod p \tag{6}$$

Let α and β be extended vectors of $a(x)$ and $b(x)$ by filling n zero elements. The multiplication of two polynomials $a(x)$ and $b(x)$ can be computed based on NTT and INTT:

$$\begin{aligned} c(x) &= a(x) \times b(x) \\ &= INTT_{\omega}^{2n}(NTT_{\omega}^{2n}(\alpha) \odot NTT_{\omega}^{2n}(\beta)) \end{aligned} \tag{7}$$

where $\odot$ is the point-wise multiplication.

To avoid zero padding in NTT polynomial multiplication, the negative wrapped convolution is used. Assume that c is the negative convolution of $a(x)$ and $b(x)$, the negative wrapped convolution can be expressed as:

$$c_i = \sum_{j=0}^{i} a_j b_{i-j} - \sum_{j=i+1}^{n-1} a_j b_{n+i-j} \tag{8}$$

Define $a' = (a_0, \psi a_1, ..., \psi a_{n-1})$, $b' = (b_0, \psi b_1, ..., \psi b_{n-1})$, and $c' = (c_0, \psi c_1, ..., \psi c_{n-1})$, where $\psi^2 \equiv \omega \mod p$, the NTT polynomial multiplication becomes

$$c' = a' \times b' = INTT_{\omega}^{n}(NTT_{\omega}^{n}(a') \odot NTT_{\omega}^{n}(b')) \tag{9}$$

By using the negative wrapped convolution, the NTT multiplication can be calculated using only n-coefficients.

The polynomial addition of two polynomials $a(x)$ and $b(x)$ is simply adding corresponding coefficients of two polynomials and then doing modulo reduction (MR). In MR operation, the coefficients of the resulting polynomial are reduced by modulus q. To execute this operation, a few MR algorithms are presented in [5,13]. Since the parameters used in this paper are $n = 512$ and $q = 12,289$, the SAMS2 algorithm for $q = 12,289$ that is presented in [5] is selected.

2.3. Discrete Gaussian Sampler

In ring-LWE cryptography, error polynomials sampled from a discrete Gaussian distribution χ_σ with a standard deviation σ are required. This distribution uses the probability function described as follows:

$$Pr(E = i) = \frac{1}{S} e^{-\frac{i^2}{2\sigma^2}} \tag{10}$$

where E is the random variable on $\mathbb{Z}$, S is the normalization factor, and i is an integer. By approximating S to the probability density function can also be described using

$$Pr(E = i) = \frac{1}{S} e^{-\frac{\pi i^2}{S^2}} \tag{11}$$

In References [5,14,15], authors present several methods for performing discrete Gaussian sampling. Among these methods, rejection sampling and inversion sampling are the popular ones [14].

In practice, rejection sampling for a discrete Gaussian distribution is slow due to the high rejection rate for the sampled values, which are far from the center of the distribution. The inversion method first generates a random probability and then selects a sample value such that the cumulative distribution up to that sample point is just larger than the randomly generated probability. Since the random probability should be of high precision, this method also requires a large number of random bits. The Knuth–Yao algorithm [14,15] uses a random walk model for sampling from any non-uniform distribution. However, the output of Knuth–Yao algorithm is generated in an unpredictable time; then, it is not a reliable sampler [5]. In this work, we deploy the linear feedback shift registers (LFSRs) approach that was proposed in [16]. This approach offers a low-complexity with an approximated uniform pseudo-random distribution; hence, it can be exploited to generate an accurate approximation of a Gaussian distribution with low maximum auto-correlation.

3. Proposed Ring-LWE Cryptoprocessor Architecture Using Shared Arithmetic Components

3.1. Proposed Algorithm for the Ring-LWE Cryptoprocessor

The proposed shared arithmetic components based ring-LWE cryptography algorithm is described in Algorithm 1. Two additional parameters *enc* and *dec* are used to control the encryption and decryption operations. Encryption operation is enabled when $enc = 1$, and the multiplier Mult2 and adder Add3 participate in encryption phase. Input message m is encoded to get polynomial m_e. This encoded message is then added with error polynomial e_3 and stored in c_{21}. Polynomials' multiplications $c_{10} = a \times e_1$ and $c_{20} = p \times e_1$ are calculated by Mult1 and Mult2, respectively. Ciphertext c_1 is computed by adding two polynomials c_{10} and e_2 using polynomial addition function Add2, while $c_2 = c_{20} + c_{21}$ is conducted by addition function Add3. Ciphertext (c_1, c_2) is then successfully carried out.

Algorithm 1: Proposed ring-LWE cryptography algorithm using shared arithmetic components

Input : $a, p \in \mathbb{Z}_q^n, m \in \{0,1\}^n, \omega \in \mathbb{Z}_q$
Output: Ciphertext $(c_1, c_2) \in \mathbb{Z}_q^n$, or original messsage m
$r_1, r_2 \leftarrow$ Gaussian sampler(s, n);
$e_1, e_2, e_3 \leftarrow$ Gaussian sampler(s, n);
while $enc = 1$ **do**
  for $i = 0$ **to** $n - 1$ **do**
    if $m[i] = 1$ **then**
      $m_e[i] \leftarrow \lfloor \frac{q}{2} \rfloor \times m[i]$;
    else
      $m_e[i] \leftarrow 0$;
    end if;
  end for;
  $c_{10} \leftarrow$ Mult1(a, e_1); $c_{20} \leftarrow$ Mult2(p, e_1);
  $c_{21} \leftarrow$ Add1(m_e, e_3); $c_1 \leftarrow$ Add2(c_{10}, e_2);
  $c_2 \leftarrow$ Add3(c_{20}, c_{21});
end while;
Return(c_1, c_2)
while $dec = 1$ **do**
  $m_{d1} \leftarrow$ Mult2(c_1, r_2); $m_d \leftarrow$ Add3(m_{d1}, c_2);
  for $i = 0$ **to** $n - 1$ **do**
    if $(\lfloor \frac{q}{4} \rfloor \leq m_d[i] \leq 3 \times \lfloor \frac{q}{4} \rfloor)$ **then**
      $m[i] = 1$;
    else
      $m[i] = 0$;
    end if;
  end for;
end while;
Return(m)

The decryption operation is enabled by the signal d_en. In this phase, multiplier Mult2 and adder Add3 that participate in the encryption phase are reconfigured to perform operations over the ring. Specifically, multiplier Mult2 calculates the multiplication between the cipher-text c_1 and the polynomial r_2. The result of this multiplication is then transferred to adder Add3, where it is added to the cipher-text polynomial c_2 to return the pre-decoded polynomial $m_d = c_1 \times r_2 + c_2$. Finally, the original message m is recovered from the pre-decoded message m_d using a decoder. The original message m is recovered from the pre-decoded polynomial m_d using a decoder. If the i-th coefficient of m_d satisfies the condition $q/4 \leq m_d[i] \leq 3q/4$, the corresponding i-coefficient of message m ($m[i]$) is decoded to 1; otherwise, it is converted to 0. The detailed timing diagram of the proposed ring-LWE crytoprocessors is shown in Figure 2.

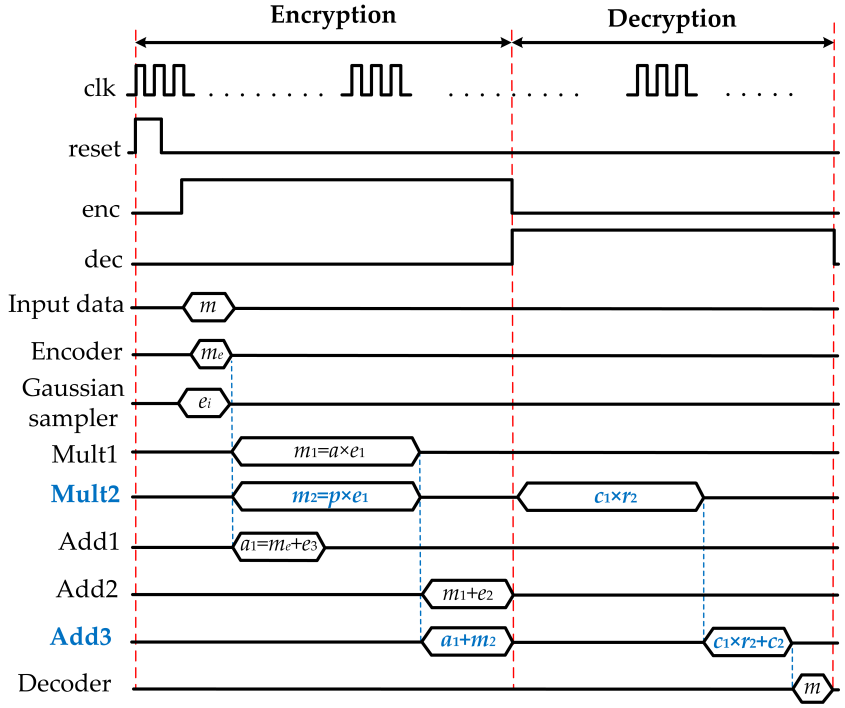

Figure 2. Timing diagram of the proposed ring-LWE cryptoprocessor architecture.

When $dec = 1$, polynomial multiplication $m_{d1} = c_1 \times r_2$ is computed using the same multiplication resource Mult2 in encryption phase. Similarly, polynominal addition Add3 is reused to compute polynomial addition m_d between polynomial m_{d1} and ciphertex c_2. Finally, the original message m is retrieved by decoding message m_d.

3.2. Ring-LWE Cryptoprocessor Architecture Using Shared Arithmetic Components

The proposed ring-LWE cryptoprocessor architecture using shared NTT polynomial multiplier and polynomial adder is illustrated in Figure 3, which consists of an encoder, a Gaussian sampler, polynomial adders, polynomial multipliers, a decoder, and a controller. As can be seen, Multiplier 2 and Adder 3 are deployed to participate in both encryption and decryption operations. The detailed architecture is described in Figure 4.

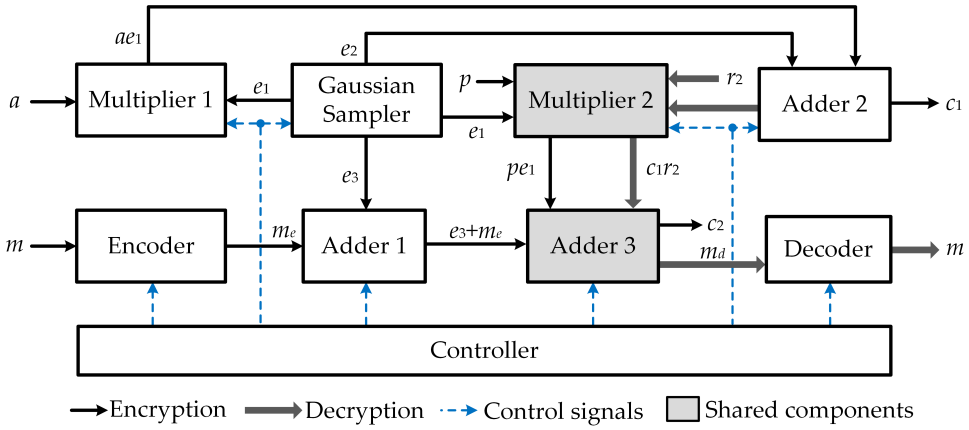

Figure 3. Proposed top-level ring-LWE cryptoprocessor architecture.

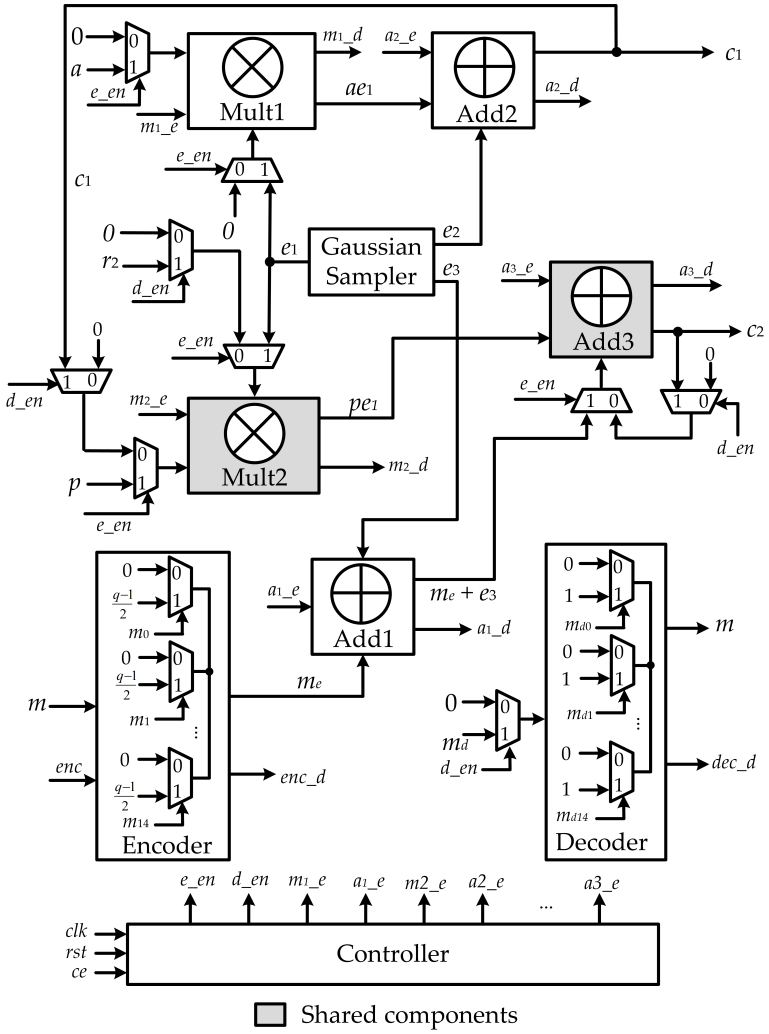

Figure 4. Proposed ring-LWE cryptoprocessor architecture using shared arithmetic components.

The encryption operation computes the cipher-text (c_1, c_2). This operation is enabled by the control signal *enc*. Initially, the input information m is encoded using an encoder. Each bit of the message m works as the control signal of the corresponding MUX, where its inputs are 0 or $(q-1)/2$. The encoded message m_e is constructed by the outputs of these MUXs. The encoded message is then added with the error polynomial e_3 using the adder Add1, controlled by signals a_1_e and a_1_d, to get

the value $(m_e + e_3)$. Multiplier Mult1 computes the product of the polynomial a and the error vector e_1, while multiplier Mult2 calculates the multiplication of the public key p and e_1. These multipliers are triggered by the control signals m_1_e and m_2_e, respectively. In the proposed architecture, we use different architectures of NTT multiplier, which are discussed in the following part. Two control signals m_1_d and m_2_d are assigned to 1 indicating that the multiplications at multipliers Mult1 and Mult2 are completely executed. The output of multiplier Mult1 becomes the input of adder Add 2, where it is added to the error vector e_2 to form the ciphertext c_1. Concurrently, the output of multiplier Mult2 is added with the polynomial $(e_3 + m_e)$ to generate ciphertext c_2, performed by adder Add3. Upon control signals, a_2_d and a_3_d are equal to 1, and the encryption operation is fully accomplished. The ciphertext (c_1, c_2) is carried out.

3.3. Proposed NTT Polynomial Multiplier Using MDF Architecture

To speed up the computation time and reduce the complexity of ring-LWE cryptoprocessors, a novel NTT polynomial multiplier using MDF architecture is proposed. The MDF architecture can provide a higher throughput rate with minimal hardware cost by combining the features of MDC and SDF. In MDF architecture, the SDF architecture is extended by using a multi-path approach. In order to achieve higher throughput rate, the number of data-paths can be increased to eight or even sixteen.

Theoretically, a NTT-based polynomial multiplier consists of three bit-reverse processes, two NTT processes, one point-wise multiplication, and one INTT process. By using the reverse Cooley–Tukey algorithm [17] in the NTT-based polynomial multiplication operation, three bit-reverse operations are eliminated, as described in Figure 5a. Therefore, the computation time and hardware complexity are greatly reduced. In addition, two NTT calculations for input polynomials are executed concomitantly to mitigate the multiplication latency. The pipeline technique is also applied between all stages of NTT multipliers to decrease critical path delay. In this work, the 8-parallel MDF architecture-based NTT multiplier is deployed. This multiplier is employed in the ring-LWE cryptoprocessors to conduct the encryption and decryption operations. The detail architecture of NTT core using 8-parallel MDF architecture is illustrated in Figure 5b. In Figure 5b, 512-coefficients of input polynomial are divided into eight parallel paths, each path consists of 64 coefficients. For example, path a_1 of polynomial $a(x)$ in Figure 5b consists of coefficients a_1, a_9, a_{17}, and so on. Two input polynomials $a(x)$ and $b(x)$ are processed using NTT transform architecture. After obtaining the NTT transform of two polynomials $a(x)$ and $b(x)$, the point-wise multiplication operation is calculated, followed by the INTT transform to return the value of the multiplication operation. The architecture of INTT core is similar with NTT core architecture, which consists of processing elements PE1 and PE2, as presented in Figure 5b. The proposed PE1 and PE2 architectures for the NTT core are detailed in Figure 6.

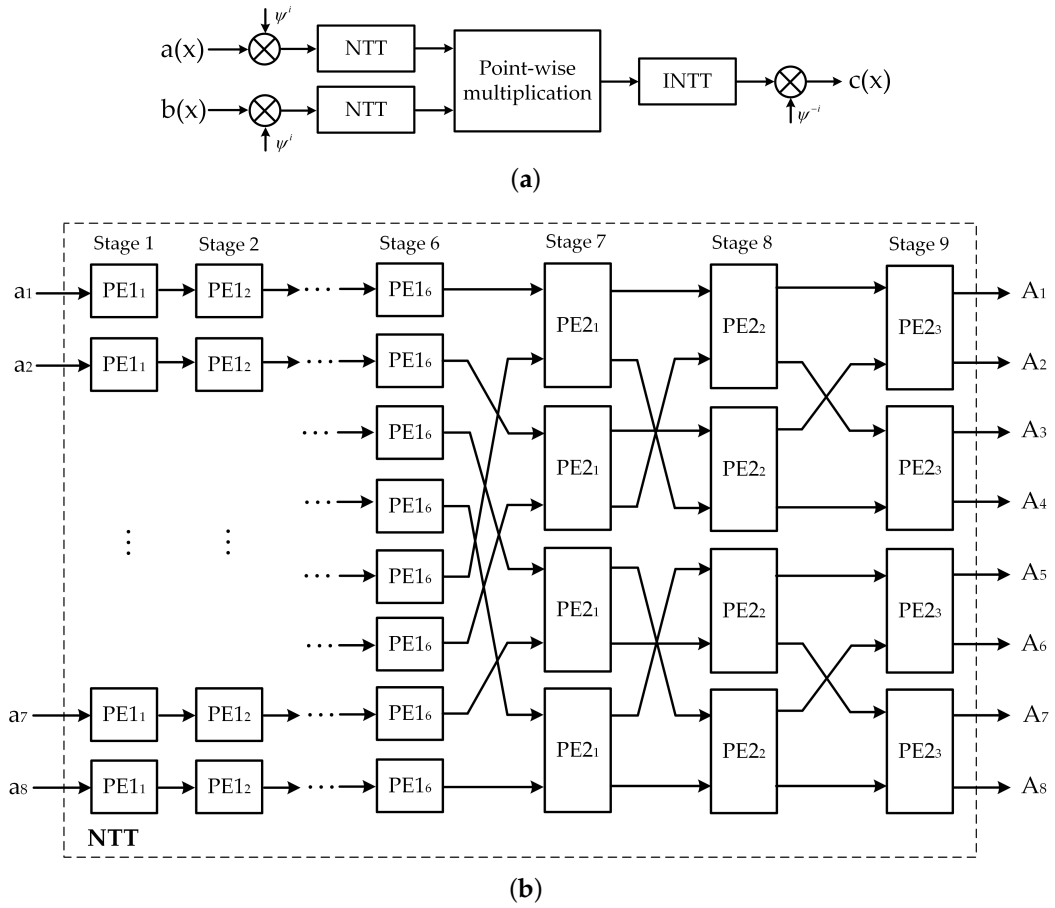

Figure 5. (**a**) NTT multiplier, and (**b**) proposed number theoretic transform (NTT) core architecture.

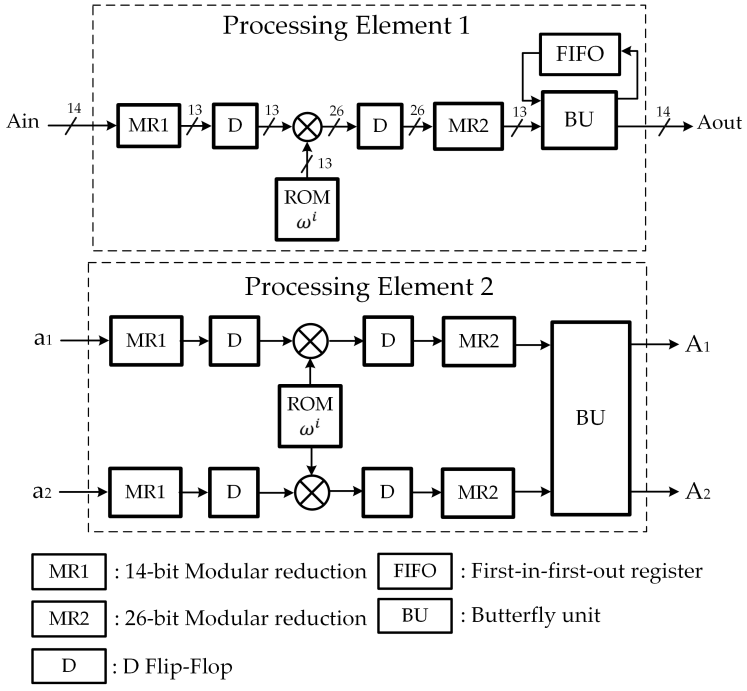

Figure 6. Proposed PE1 and PE2 architectures for the NTT core.

4. Implementation Results and Comparison

The proposed architecture for ring-LWE cryptoprocessor is modeled in Verilog HDL, synthesized, and implemented using Xilinx VIVADO 2017.4 on a Virtex-7 FPGA platform. The implementation results of ring-LWE cryptoprocessor are summarized in Table 1.

Table 1. Performance comparison of different ring-LWE cryptoprocessors.

	Proposed	**R8M [5]**	**[6]**	**[18]**	**[19]**
Devices	Virtex-7	Stratix IV	Virtex-7	Virtex-6	Virtex-6
LUTs (enc./dec.)	61,258/–	62,994/27,313	61,514/25160	5595/5595	1536/1536
Slices (enc./dec.)	23,707/–	56,435/32,019	42,374/23,495	4760/4760	953/953
Frequency (enc./dec.) (MHz)	284/330	226/216	269/315	250/251	277/276
Clock cycles (enc./dec.)	242/235	391/225	240/224	13,769/8883	13,300/5800
Time (enc./dec.) (µs)	0.85/0.71	1.73/1.04	0.89/0.71	54.86/35.39	47.90/21.00
Throughput[a] (enc./dec.) (Mbps)	8432/721	4465/492	8054/720	130/14	150/24
Efficiency[b] (Kbps/LUT)	137	66	130	23	95

[a] Throughput = (Frequency × Number of bits)/Number of clock cycles. [b] Efficiency = Throughput/Number of LUTs.

As can be seen from Table 1, the proposed architecture requires less hardware resources, calculated in number of slices and LUTs, to conduct a completed ring-LWE encryption–decryption operation compared with similar parallel multiplier-based ring-LWE architectures presented in [5,6]. Specifically, to perform ring-LWE encryption and decryption operations, the proposed architecture uses only 23,707 slices and 61,258 LUTs, which is about 42% and 67.83% of that in [5], respectively. Additionally, the encryption and decryption throughput of the proposed architecture are higher than that of architecture in [6] and R8M architecture in [5]. The architectures in [18,19] require a small number of slices and LUTs to perform encryption and decryption; however, these architectures require high latency to complete ring-LWE encryption and decryption operations. Therefore, the values of throughput offered by these architectures are smaller than 150 Mbps, as described in Table 1.

Efficiency presented in [20] can be used as a parameter to evaluate the performance of different designs on various FPGA platforms. In detail, the efficiency parameter represents the throughput value that one hardware unit (LUT) of an architecture can offer. As can be seen from Table 1, the proposed ring-LWE cryptoprocessor architecture can offer a better value of efficiency compared with that of other architectures. Specifically, the obtained efficiency value of the proposed architecture is about two and seven times larger than that of architectures in [5,18], respectively. Comparison in encryption time, decryption time, and efficiency is described in Figure 7.

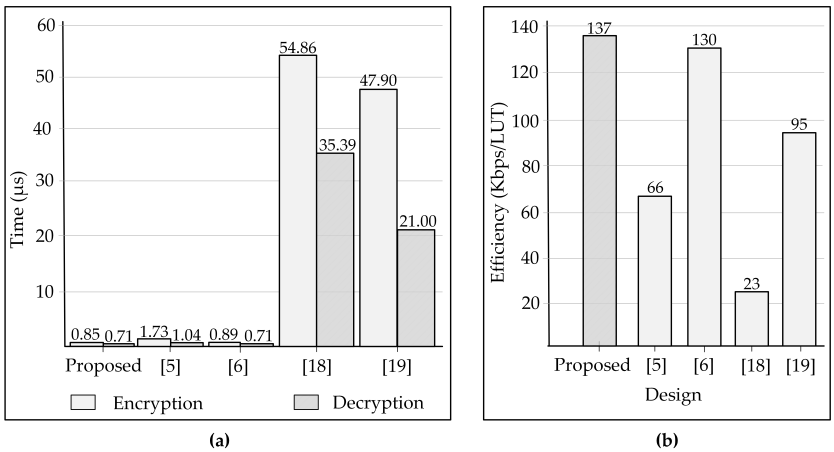

Figure 7. (**a**) comparison in encryption and decryption time, and (**b**) comparison in system efficiency.

5. Conclusions

A novel approach to improve encryption and decryption operation of ring-LWE cryptoprocessors has been presented in this paper. By sharing the hardware resources, including polynomial multiplier and polynomial adder, in both encryption and decryption phases, the hardware complexity of the proposed architecture can be significantly reduced. Furthermore, the polynomial multiplication is greatly enhanced by deploying an efficient NTT multipliers using MDF architecture and the pipeline technique between all stages of NTT multipliers. As a result, the proposed ring-LWE cryptoprocessor offers higher throughput and efficiency compared with that of existing works. Therefore, it can be applied in hardware resource-limited systems that require high throughput and low latency.

Author Contributions: T.N.T. conceptualized the idea of this research, conducted experiments, collected data, and prepared the original version. T.T.B.N. reviewed, analyzed data, and updated the manuscript. H.L. supervised, validated, reviewed, and supported the research with funding. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Basic Science Research Program (2019R1F1A1061926) through the NRF funded by the MSIT and by Inha University, Incheon, Korea.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ochoa-Jiménez, E.; Rivera-Zamarripa, L.; Cruz-Cortés, N.; Rodríguez-Henríquez F. Implementation of RSA Signatures on GPU and CPU Architectures, *IEEE Access* **2020**, *8*, 9928–9941. [CrossRef]
2. Nguyen, T.T.; Lee, H. Efficient Algorithm and Architecture for Elliptic Curve Cryptographic Processor. *J. Semicond. Technol. Sci.* **2016**, *16*, 118–125. [CrossRef]
3. Basu Roy, D.; Mukhopadhyay, D. High-Speed Implementation of ECC Scalar Multiplication in GF(p) for Generic Montgomery Curves. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 1587–1600. [CrossRef]
4. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
5. Rentería-Mejía, C.P.; Velasco-Medina, J. High-Throughput Ring-LWE Cryptoprocessors. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2332–2345. [CrossRef]
6. Nguyen Tan, T.; Lee, H. Efficient-Scheduling Parallel Multiplier-Based Ring-LWE Cryptoprocessors. *Electronics*, **2019**, *8*, 413(1–13). [CrossRef]
7. Nguyen Tan, T.; Lee, H. High-Secure Fingerprint Authentication System Using Ring-LWE Cryptography. *IEEE Access* **2019**, *7*, 23379–23387. [CrossRef]
8. Nguyen Tan, T.; Lee, H. High-Performance Ring-LWE Cryptography Scheme for Biometric Data Security. *IEIE Trans. Smart Process. Comput.* **2018**, *7*, 97–106. [CrossRef]
9. Liu, Z.; Azarderakhsh, R.; Kim, H.; Seo, H. Efficient Software Implementation of Ring-LWE Encryption on IoT Processors. *IEEE Trans. Comput.* **2017**. [CrossRef]
10. Liu, D.; Zhang, C.; Lin, H.; Chen, Y.; Zhang, M. Resource-Efficient and Side-Channel Secure Hardware Implementation of Ring-LWE Cryptographic Processor. *IEEE Trans. Circuits and Syst. I, Reg. Papers* **2019**, *66*, 1474–1483. [CrossRef]
11. Rentería-Mejía, C.P.; Velasco-Medina, J. Hardware Design of an NTT-Based Polynomial Multiplier. In Proceeding of 2014 Southern Conference on Programmable Logic (SPL), Buenos Aires, Argentina, 5–7 November 2014; pp. 1–5.
12. Regev, O. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In Proceeding of ACM symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005; pp. 84–93.
13. Cao, Z.; Wu, X. An Improvement of the Barrett Modular Reduction Algorithm. *Int. J. Computer Mathematics* **2014**, *91*, 1874–1879. [CrossRef]
14. Roy, S.S.; Vercauteren, F.; Verbauwhede, I. High Precision Discrete Gaussian Sampling on FPGAs. In Proceeding of Selected Areas in Cryptography, Burnaby, BC, Canada, 14–16 August 2013; pp. 383–401.
15. Du, C.; Bai, G. Towards Efficient Discrete Gaussian Sampling for Lattice-Based Cryptography. In Proceeding of 2015 International Conference on Field Programmable Logic and Applications (FPL), London, UK, 2–4 September 2015; pp. 1–6.

16. Condo, C.; Gross, W.J. Pseudo-Random Gaussian Distribution Through Optimised LFSR Permutations. *Electron. Lett.* **2015**, *51*, 2098–2100. [CrossRef]
17. Cooley, J.W.; Tukey, J.W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation* **1965**, *19*, 297–301. [CrossRef]
18. Roy, S.S.; Vercauteren, F.; Mentens, N.; Chen, D.D.; Verbauwhede, I. Compact Ring-LWE Cryptoprocessor. In Proceeding of International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Busan, South Korea, 23–26 September 2014; pp. 371–391.
19. Pöppelmann, T.; Güneysu, T. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. In Proceeding of International Conference on Selected Areas in Cryptography, Burnaby, BC, Canada, 14–16 August 2013; pp. 68–85.
20. Mahdizadeh, H.; Masoumi, M. Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor over GF(2^{163}). *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *21*, 2330–2333. [CrossRef]