applied sciences

MDPI

Article

Area-Time Efficient Hardware Architecture for CRYSTALS-Kyber

Tuy Tan Nguyen †, Sungjae Kim †, Yongjun Eom and Hanho Lee *

Department of Information and Communication Engineering, Inha University, Incheon 22212, Korea; tuynguyen@inha.ac.kr (T.T.N.); ksj808080@naver.com (S.K.); 22211260@inha.edu (Y.E.)
* Correspondence: hhlee@inha.ac.kr; Tel.: +82-32-860-7449
† These authors contributed equally to this work.

Abstract: This paper presents a novel area-time efficient hardware architecture of the lattice-based CRYSTALS-Kyber, which has entered the third round of the post-quantum cryptography standardization competition hosted by the National Institute of Standards and Technology. By developing a dual-path delay feedback number theoretic transform multiplier dedicating for Kyber parameter set and deploying this multiplier in the Kyber architecture, the key generation, encryption, and decryption operations are accelerated substantially. Furthermore, the proposed architecture offers the best value of area-time product in comparison with existing approaches. The implementation results on Xilinx Vivado targeted for Virtex-7 FPGA board demonstrate that the proposed Kyber cryptoprocessor completes encryption and decryption operations in approximately 57.5 µs at the highest frequency of 226 MHz. Furthermore, the area-time product value when using the proposed Kyber architecture is improved by at least twofold compared with existing architectures.

Keywords: CRYSTALS-Kyber; decryption; encryption; number theoretic transform (NTT); polynomial multiplier; post-quantum cryptography

Citation: Nguyen, T.T.; Kim, S.; Eom, Y.; Lee, H. Area-Time Efficient Hardware Architecture for CRYSTALS-Kyber. *Appl. Sci.* **2022**, *12*, 5305. https://doi.org/10.3390/app12115305

Academic Editors: Howon Kim and Thi-Thu-Huong Le

Received: 31 March 2022
Accepted: 21 May 2022
Published: 24 May 2022

1. Introduction

Existing public key cryptography schemes such as Rivest-Shamir-Adleman and elliptic curve cryptography can be broken by powerful quantum computers running Shor's algorithm [1]. Consequently, post-quantum cryptography (PQC) has attracted immense attention from the research community [2–6] during the last few years. The National Institute of Standards and Technology (NIST) began promoting standardization of PQC [7] in 2016 through a competition. This competition attracted a large number of submissions, demonstrating the keen interest that has been engendered. There were initially 69 submissions, but only 25 and 16, respectively, remained after the first and second rounds. The results of round three are expected to be announced in 2022. Among the finalists of round three, CRYSTALS-Kyber [8,9] is considered a very promising candidate.

Implementations of Kyber have been described in recent works in pure software design [8,10,11], software-hardware codesign [11], and pure hardware design [11–14]. Authors in [8] presented the implementation of Kyber on Intel and ARM Cortex-M4 CPUs. In [11], authors reported the software implementation of Kyber using C language on ARM Cotex-A53. Authors in [10] introduced highly parallel algorithms implemented on a GPU to accelerate the operations of Kyber. Authors in [11] also introduced a software-hardware codesign of Kyber that helped improve the encapsulation and decapsulation times compared with those of the pure software design. Pure hardware implementations were presented in [11–14] to reduce the hardware complexity and speed up operations in Kyber. In hardware design of Kyber, many authors used the number theoretic transform (NTT)-based multiplier to execute the polynomial multiplication. However, the existing NTT-based multipliers which process data in serial limit the performance of operations in Kyber.

In this study, we focus on designing an area-time efficient hardware architecture for Kyber. Specifically, we introduce a dual-path delay feedback (DDF) number theoretic transform (NTT) designed for a specific Kyber's parameter set. Next, we implement the proposed NTT/INTT architecture in the key generation, encryption, and decryption modules. We also present the architecture design of Kyber's sub-modules in detail. The implementation results on Xilinx Vivado targeted for Virtex-7 FPGA board show that the proposed Kyber architecture offers the best value of area-time product compared with existing approaches. The contributions of this paper are itemized as follows:

1. We introduce a DDF-based NTT architecture designed for Kyber. Specifically, the coefficients of the input polynomial are grouped into odd and even groups to increase the speed of the NTT operation.
2. We deploy the proposed DDF-based NTT in the key generation, encryption, and decryption architectures of Kyber to accelerate these operations.
3. We introduce a new architecture design for the main modules used in Kyber such as key generation, encryption, decryption, polynomial multiplication, compression, decompression, and modulo reduction in detail.
4. We synthesize and implement the proposed Kyber architecture using Xilinx Vivado targeted for a Virtex-7 FPGA board. The implementation results show that the proposed Kyber architecture outperforms its predecessors in terms of area-time efficiency.

The remainder of this paper is structured as follows. In Section 2, we briefly discuss the preliminaries. In Section 3, we present the architecture design of the main modules in Kyber. The implementation results and comparison are discussed in Section 4. Finally, we conclude the paper in Section 5.

2. CRYSTALS-Kyber Algorithms

CRYSTALS-Kyber [8] is a lattice-based CCA-secure key encapsulation mechanism based on the problem of module learning with errors. Kyber introduces three parameter sets corresponding to three security levels of NIST, as summarized in Table 1. Specifically, polynomials are of the same degree $n = 256$, and their coefficients are members of the base prime field Z_q, where $q = 3329$ for all security levels. However, different numbers of polynomials are required for each security level. These polynomials are treated as a vector whose size is specified using the parameter k, where k is 2, 3, and 4, corresponding to three security levels, 1, 3, and 5, respectively. Secret noise polynomials are sampled from a centered binomial distribution.

Table 1. Kyber parameter sets [15].

Algorithm	NIST Security Level	Parameters		
		n	k	q
Kyber-512	1	256	2	3329
Kyber-768	3	256	3	3329
Kyber-1024	5	256	4	3329

Kyber is categorized in the public-key encryption and key-establishment algorithms of NIST round three finalists. Functions in the Kyber public-key encryption algorithm include key generation, encryption, and decryption, which are described as follows:

1. KeyGen(): Key generation creates a public key pk used in the encryption process and a secret key sk used in the decryption processes. Noise vectors $\mathbf{s}$ and $\mathbf{e}$ are sampled from a centered binomial distribution. The public matrix $\hat{\mathbf{A}}$ is sampled from a rejection sampler. The public key pk and private key sk are obtained from formulas $pk = (\rho, \hat{\mathbf{t}})$ and $sk = \hat{\mathbf{s}}$, where $\hat{\mathbf{t}} = \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$.
2. Enc(pk, m, r): Encryption constructs ciphertexts $c = (c_1, c_2)$ from the input message m, public key pk, and random coins $r \in \mathcal{B}^{32}$. $\hat{\mathbf{A}}^T$ is sampled from a uniform distribution,

and r, $\mathbf{e}_1$, e_2 are obtained from a binomial sampler. Ciphertext c is computed as $c = (\text{Compress}_q(\mathbf{u}, d_u), \text{Compress}_v(v, d_v))$, where $\mathbf{u} = \text{INTT}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ and $v = \text{INTT}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2 + m$.

3. Dec(sk, c): Decryption recovers the original message m from ciphertext c using secret key sk. The value of m is calculated as $m = \text{Compress}(v - \text{INTT}(\hat{\mathbf{s}}^T \circ \hat{\mathbf{u}})$, where $\mathbf{u}$ and v are obtained from c.

3. Proposed Area-Time Efficient Hardware Architecture for Kyber

In this section, we present the proposed DDF-based NTT/INTT architecture designed for Kyber. Next, we deploy the proposed NTT/INTT architecture in key generation, encryption, and decryption operations to enhance these operations. We also describe the architecture design of main modules used in Kyber in detail.

3.1. Proposed Dual-Path Delay Feedback NTT Architecture for Kyber

NTT is a variant of the conventional fast Fourier transform (FFT) [16,17] with arithmetic operations being performed in a finite field. NTT uses the n-th primitive root of unity ω_n in the ring $\mathbb{Z}_q$. NTT multiplication is selected as a part of the Kyber scheme. Consider a polynomial a in R_q as follows:

$$a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{255} x^{255} \tag{1}$$

NTT(a) is defined as

$$\hat{a}(x) = \hat{a}_0 + \hat{a}_1 X + \hat{a}_2 X^2 + \cdots + + \hat{a}_{255} X^{255} \tag{2}$$

In Equation (2), $\hat{a}_{2i} = \sum_{j=0}^{127} a_{2j} \zeta^{(2\text{br}_7(i)+1)j)}$ and $\hat{a}_{2i+1} = \sum_{j=0}^{127} a_{2j+1} \zeta^{(2\text{br}_7(i)+1)j)}$, where $\zeta = 17$ is the first primitive 256th root of unity modulo q, and $\text{br}_7(i)$ is the bit reversal of the unsigned 7-bit integer i. The inverse NTT (INTT) operation is similar to NTT, which converts $\hat{a}(x)$ back to $a(x)$, where ω_n is replaced by ω_n^{-1}, and the resulting coefficients of $a(x)$ are divided by n [14]. We proposed a DDF-based NTT/INTT architecture to enhance the polynomial multiplication in Kyber. A polynomial a defined in Equation (1) can be rewritten as

$$a(x) = \sum_{i=0}^{127} a_{2i} x^{2i} + \sum_{i=0}^{127} a_{2i+1} x^{2i+1} \tag{3}$$

In order to accelerate NTT/INTT operations, the coefficients of input polynomial are divided into two parts to be processed in parallel. The proposed NTT/INTT architecture for Kyber is presented in Figure 1. As can be seen from Figure 1a, the coefficients of input polynomials are divided into two groups, odd and even coefficients, so that they can be handled simultaneously. Each group will be processed through seven stages which include two modulo reduction modules, butterfly units, first-in-first-out (FIFO) registers, and pipelined stages. Values of ω_n and ω_n^{-1} are loaded from memory (MEM). The output of the NTT module includes two parts, $A_0, A_2, \ldots, A_{254}$ and $A_1, A_3, \ldots, A_{255}$, corresponding to the coefficients $a_0, a_2, \ldots, a_{254}$ and $a_1, a_3, \ldots, a_{255}$ of the input polynomial. In the proposed NTT architecture, modulo reduction modules M1 and M2 are Barret reduction [18] and Montgomery reduction [19], respectively. Architectures of Montgomery reduction and Barret reduction are introduced in Figure 2a,b, respectively.

The INTT architecture in Figure 1b converts $A_0, A_1, \ldots, A_{255}$ back to $a_0, a_1, \ldots, a_{255}$. These coefficients are also divided into two groups to process simultaneously.

$$A = \sum_{i=0}^{127} A_{2i} X^{2i} + \sum_{i=0}^{127} A_{2i+1} X^{2i+1} \tag{4}$$

The remaining parts of INNT architecture are similar to NTT architecture, with ω_n being replaced by ω_n^{-1}, as shown in Figure 1b. Montgomery reduction and Barret reduction are also used in INTT architecture.

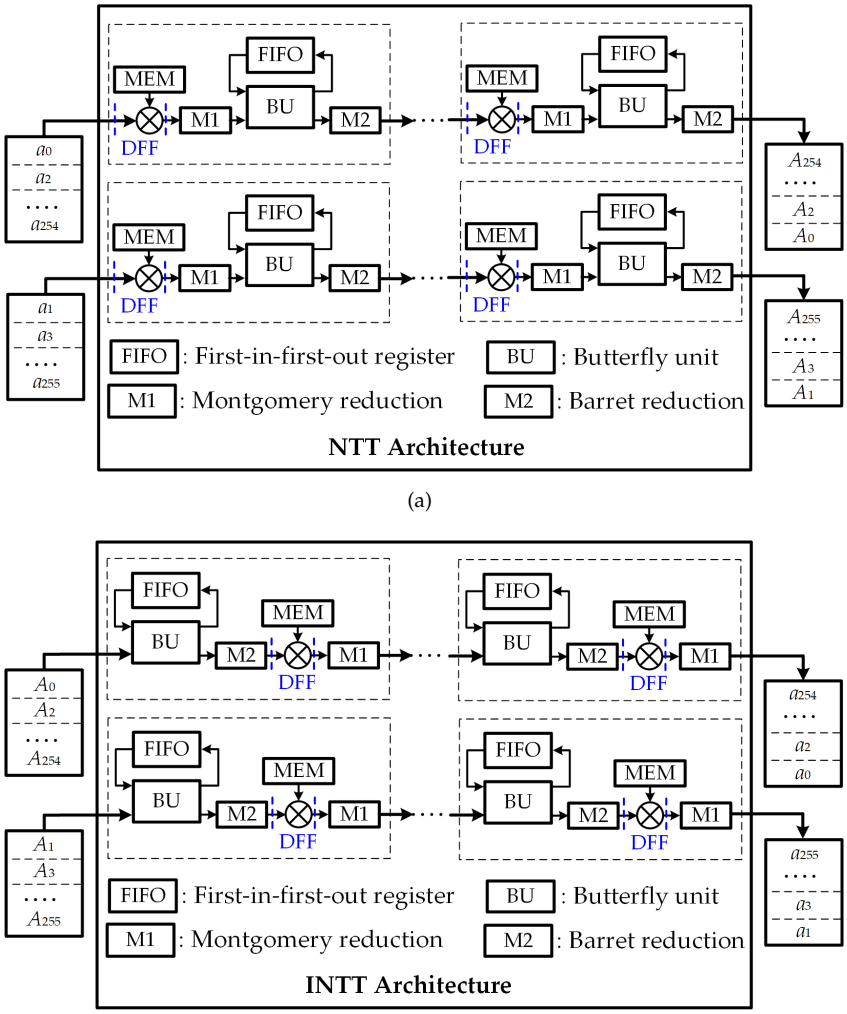

Figure 1. Proposed DDF-based architecture for Kyber (**a**) NTT architecture and (**b**) INTT architecture.

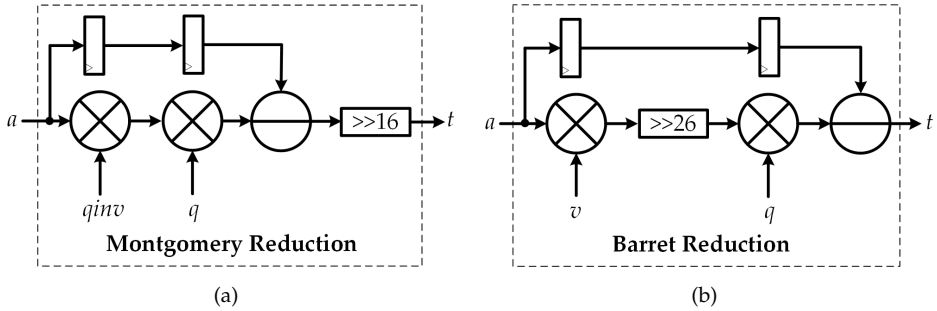

Figure 2. Modulo reduction architectures: (**a**) Montgomery reduction and (**b**) Barret reduction.

3.2. Proposed Kyber Key Generation Architecture

The proposed DDF NTT multiplier-based key generation architecture is introduced in Figure 3. Specifically, Figure 3a presents the key generation architecture to construct public key *pk* and secret key *sk* from input noise vectors. The PRNG core, a Keccak-based architecture [20,21] used to generate random values from the input seed, is presented in

Figure 3b. Noise vectors **s** and **e** are generated from a binomial sampler. In addition, $\hat{\mathbf{A}}$ is chosen from a uniform distribution. The architecture of the binomial sampler and rejection sampler is introduced in Figure 3c.

To accelerate the NTT operation, we deploy the proposed DDF-based NTT architecture in the key generation architecture to obtain the values of $\hat{\mathbf{s}}$ = NTT(**s**) and $\hat{\mathbf{e}}$ = NTT(**e**). Specifically, coefficients of each noise vector **s** and **e** are divided into two groups to be processed in parallel in order to reduce processing time. The values of NTT(**s**) and NTT(**e**) are then stored in BRAM. In addition, the value of $\hat{\mathbf{t}}$ is calculated by adding the point-wise multiplication result $\hat{\mathbf{A}} \circ \hat{\mathbf{s}}$ to $\hat{\mathbf{e}}$. Finally, a key pair (pk, sk) is generated using two encoders $pk = \text{Encode}(\hat{\mathbf{t}} \bmod^{+} q)||\rho$ and $sk = \text{Encode}(\hat{\mathbf{s}} \bmod^{+} q)$. The public key pk and secret key sk then participate in encryption and decryption processes, respectively.

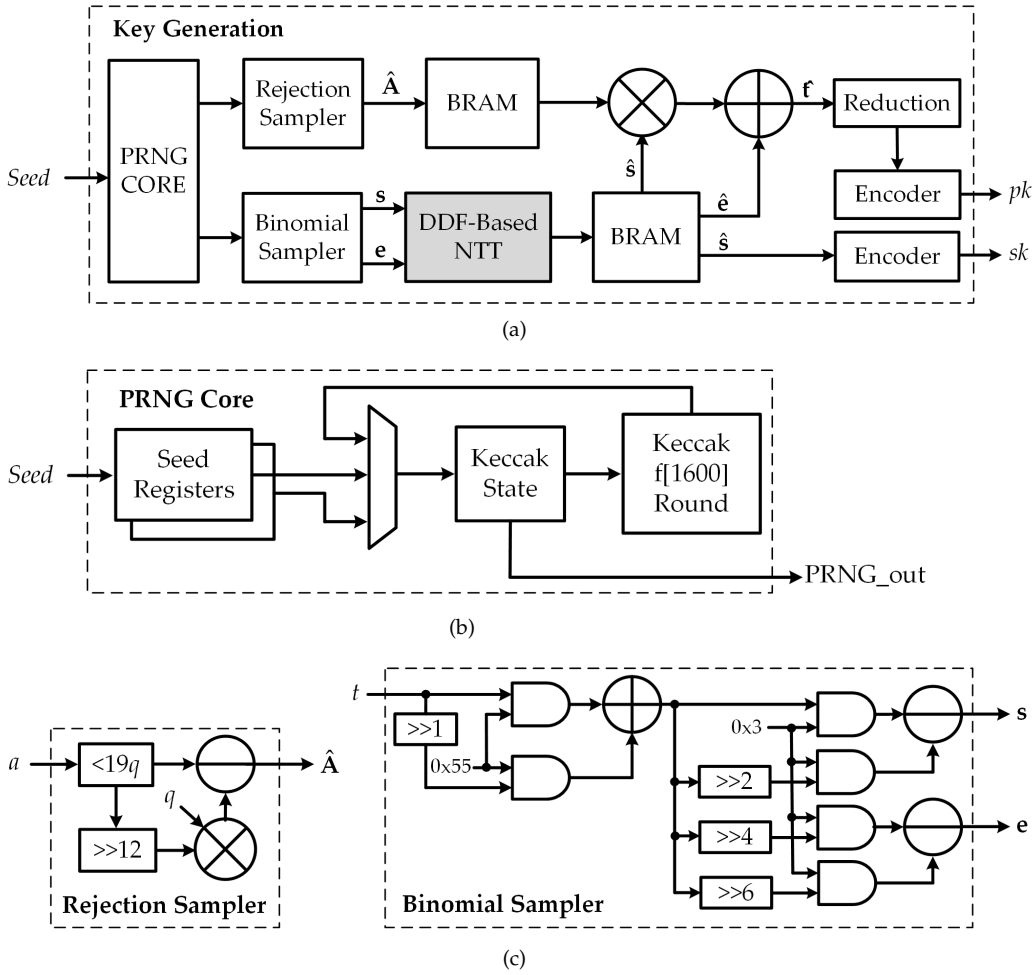

Figure 3. Proposed DDF NTT-based key generation architecture. (**a**) Top level of key generation architecture, (**b**) PRNG core architecture, and (**c**) rejection and binomial samplers.

3.3. Proposed Kyber Encryption Architecture

The proposed encryption architecture constructs the ciphertext $c = (c_1, c_2)$ from input message m using public key pk. The main modules in the proposed encryption architecture include samplers, DDF-based NTT, DDF-based INTTs (INTT1 and INTT2), decoder, encoder, compression, decompression, point-wise multiplication, and addition modules, as shown in Figure 4.

The value of ρ is obtained using the formula $\rho = pk + 12 \cdot k \cdot n/8$. The rejection sampler generates $\hat{\mathbf{A}}^{T}$ from the seed ρ, while the binomial sampler generates error vectors $\mathbf{e}_1$ and e_2 from a random coin r. $\hat{\mathbf{r}}$ is obtained by performing NTT(r) using the DDF-based NTT module in Figure 4. A part of ciphertext, c, c_1, is constructed by encoding the compressed

message $\mathbf{u}$, where $\mathbf{u} = \mathrm{INTT}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$. $\mathrm{INTT}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}})$ is performed by the proposed DDF-based INTT1 module in Figure 4.

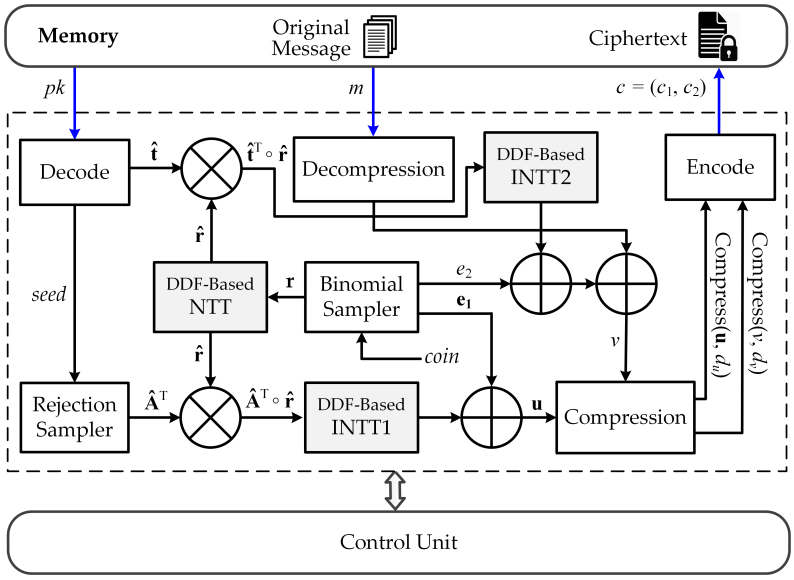

Figure 4. Proposed encryption architecture.

The architecture design of the compression block is presented in Figure 5. The compression elements (CE) in Figure 5a are detailed in Figure 5b.

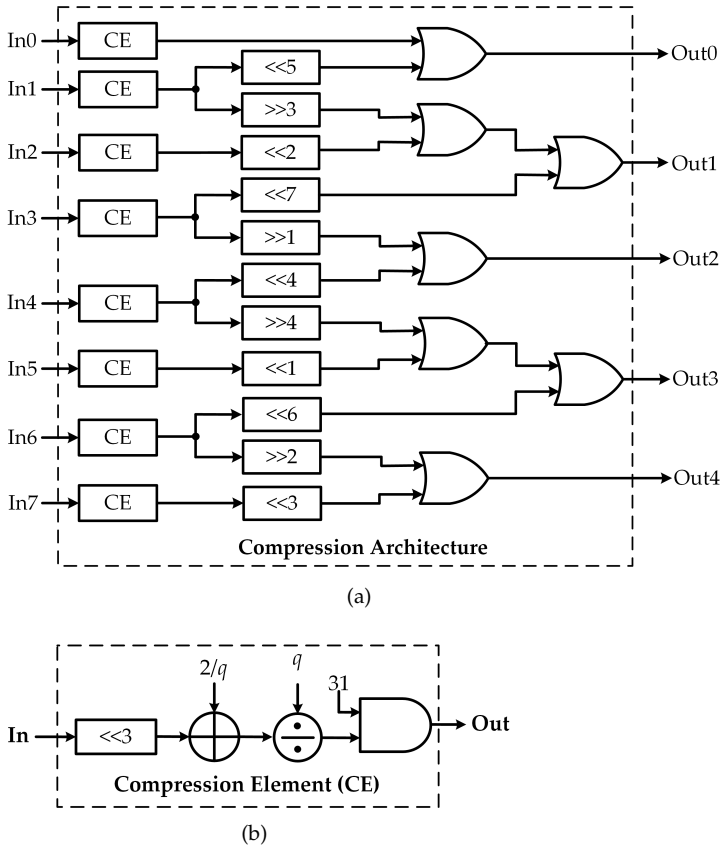

Figure 5. Compression block: (**a**) compression architecture, and (**b**) compression element.

The remaining part of ciphertext c, c_2 is constructed from the public key pk, error polynomial e_2, input coin r, and input message m. The output $\hat{\mathbf{t}}$ of the decode module

in Figure 4 is multiplied with the transform of sample $\mathbf{r}$, $\hat{\mathbf{r}} = \text{NTT}(\mathbf{r})$ using point-wise multiplication. This result is transformed using the DDF-based INTT2 module in Figure 4, and then the error polynomial e_2 is added. The result of this addition, $\text{INTT}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2$, is added with the decompression of input message m to generate $v = \text{INTT}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2 + \text{Decompress}(m)$. The architecture design of the decompression block is presented in Figure 6. The decompression elements (DE) in Figure 6a are detailed in Figure 6b. Ciphertext c_2 is generated by encoding the compression of v, $c_2 = \text{Encode}(\text{Compress}(v, d_v))$ using compression and encoder modules. Finally, ciphertext $c = (c_1, c_2)$ is constructed.

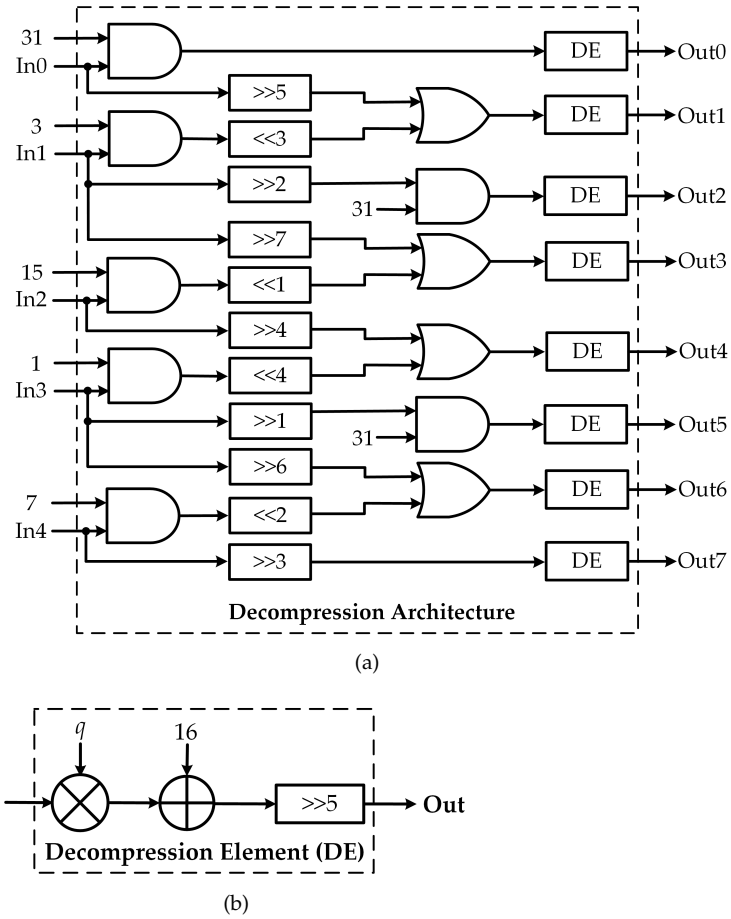

Figure 6. Decompression block: (**a**) decompression architecture, and (**b**) decompression element.

3.4. Proposed Decryption Architecture Design

The proposed Kyber decryption architecture to recover the input message m from ciphertext $c = (c_1, c_2)$ and secret key sk is presented in Figure 7.

Secret key sk and ciphertext c are decoded and then decompressed to obtained the values of $\hat{\mathbf{s}}$ and $\mathbf{u}$, respectively. In particular, $\mathbf{u} = \text{Decompress}_q(\text{Decode}_{d_u}(c), d_u)$, and $\hat{\mathbf{s}} = \text{Decode}(sk)$. In addition, the value of $v = \text{Decompress}_q(\text{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$ can be reconstructed from ciphertext c through the decompression and decoding operations. The obtained value of $\mathbf{u}$ is transformed using the proposed DDF-based NTT, and then multiplied by $\hat{\mathbf{s}}^T$. The product $\hat{\mathbf{s}}^T \circ \text{NTT}(\mathbf{u})$ is transformed using DDF-based INTT module, and then subtracted by v to obtain the value $v - \text{INTT}(\hat{\mathbf{s}}^T \circ \text{NTT}(\mathbf{u}))$. This result is compressed using the compression module in Figure 5, and encoded to obtain the value $m = \text{Encode}(\text{Compress}(v - \text{INTT}(\hat{\mathbf{s}}^T \circ \text{NTT}(\mathbf{u})), 1))$. The original message m is finally recovered.

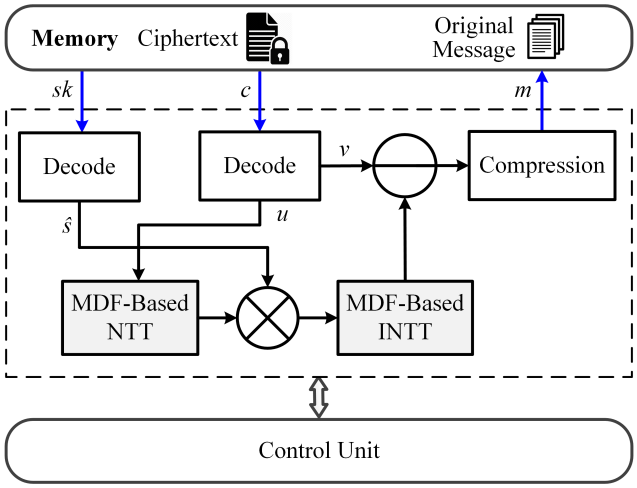

Figure 7. Proposed decryption architecture.

4. Implementation Results

The proposed architectures for Kyber are modeled using Verilog HDL and synthesized using the Xilinx Vivado 2020.1. The implementation results are the placed-and-routed data on Xilinx Virtex-7. The results for key generation and encryption/decryption are presented in Tables 2 and 3, respectively.

From Table 2, the proposed key generation architecture requires 22K LUTs, 12K FFs, 21 DSPs, and 4.5 BRAMs. As can be seen, the proposed key generation architecture can operate at a maximum frequency of 217 MHz. Public key *pk* and secret key *sk* are successfully generated after only 39 µs. As a result, the proposed key generation architecture achieves the area-to-time value of 0.86.

Table 2. Implementation results of Kyber key generation architectures.

Parameters	This Work
Devices	Virtex-7
LUTs	22K
FFs	12K
DSPs	21
BRAMs	4.5
Frequency (MHz)	217
Time (µs)	**39**
Area × Time (LUTs × *s*)	**0.86**

Table 3 describes a comparison between the proposed Kyber architecture with the most recent works [12–14,22]. The reported values of the proposed work in Table 3 are for encryption and decryption operations. Generally, the proposed architecture offers the best value of area-time product. The proposed architecture requires 29,718 LUTs, 22,556 FFs, 71 DSPs, and 11 BRAMs to perform Kyber encryption and decryption operations. These hardware resource requirements are substantially lower than those of the architecture in [13]. The hardware cost of the proposed architecture is slightly higher than that of the architectures in [12,22]. However, the proposed architecture outperforms those architectures in terms of processing time and area-time product. Specifically, the total encryption and decryption time of the proposed architecture is approximately 57.5 µs, which is approximately 3.6 times and 2.7 times faster than that of architectures in [12,14], respectively. Noticeably, the proposed architecture is much better than that of architectures in [13,22] in terms of latency. Furthermore, the area-time product value of the proposed architecture

is 1.68, which is approximately half of that of the architectures in [12,14]. Specifically, this area-time product value is approximately 189 times and 306 times smaller than that of the designs in [13,22]. Furthermore, the proposed architecture can operate at the highest frequency of 226 MHz, which is higher than the frequency at which architectures in [12–14] operate.

Table 3. Performance comparison with existing schemes in NIST security level 5.

Parameters	This Work	[14]	[12]	[13]	[22]
Devices	Virtex-7	Virtex-7	UltraScale+	Virtex-7	Virtex-7
Protocol	Kyber-1024	Kyber-1024	Kyber-1024	Kyber-1024	SIKEp751
LUTs	29,718	16,000	23,868	252,107	20,207
FFs	22,556	6000	9805	335,125	39,339
DSPs	71	12	0	584	452
BRAMs	11	17	2	402.5	41.5
Frequency (MHz)	226	156	150	192	233
Time (µs)	57.5	205	153	1264	25,500
Area × Time (LUTs× s)	**1.68**	3.28	3.62	318.66	515.28
Norm. Area × Time	**1.00**	1.95	2.15	189.68	306.71

5. Conclusions

An area-time efficient hardware architecture for CRYSTALS-Kyber is introduced in this paper. By implementing the proposed dual-path delay feedback NTT and INTT architectures in key generation, encryption, and decryption, these operations are noticeably improved in terms of processing time and area-time efficiency. Specifically, the key generation operation in the proposed architecture is accelerated by twofold compared with existing works. In addition, the proposed architecture offers the best value of area-time product among the state-of-the-art Kyber architectures. Therefore, the proposed architecture can be applied in designs where the high balance between hardware complexity and processing time is strictly considered.

Author Contributions: T.T.N., S.K. and Y.E. conceptualized the idea of this research, conducted experiments, collected data, and prepared the original version. T.T.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 Ministry of Science and ICT (MSIT) under the ITRC support program (IITP-2021-0-02052) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP), in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1011232), and in part by the IITP grant funded by the Korea government (MSIT) (No. 20210007790012003).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
2. Nejatollahi, H.; Dutt, N.; Ray, S.; Regazzoni, F.; Banerjee, I.; Cammarota, R. Post-Quantum Lattice-Based Cryptography Implementations: A Survey. *ACM Comput. Surv.* **2019**, *51*, 1–41. [CrossRef]

3. Nguyen Tan, T.; Lee, H. High-Performance Ring-LWE Cryptography Scheme for Biometric Data Security. *IEIE Trans. Smart Process. Comput.* **2018**, *7*, 97106. [CrossRef]
4. Bos, J.W.; Costello, C.; Naehrig, M.; Stebila, D. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 18–20 May 2015; pp. 553–570. [CrossRef]
5. Moody, D.; Alagic, G.; Apon, D.; Cooper, D.; Dang, Q.; Kelsey, J.; Liu, Y.; Miller, C.; Peralta, R.; Perlner, R.; et al. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. In Proceedings of the NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, USA, 22 July 2020. [CrossRef]
6. Duong-Ngoc, P.; Nguyen Tan, T.; Lee, H. Efficient NewHope Cryptography Based Facial Security System on a GPU. *IEEE Access* **2020**, *8*, 108158–108168. [CrossRef]
7. Moody, D. Post Quantum Cryptography Standardization: Announcement and outline of NIST's Call for Submissions. In Proceedings of the PQCrypto 2016, Fukuoka, Japan, 24–26 February 2016. Available online: https://csrc.nist.gov/Presentations/2016/Announcement-and-outline-of-NIST-s-Call-for-Submis (accessed on 27 March 2022).
8. Avanzi, R.; Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation. Available online: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf (accessed on 27 March 2022).
9. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Kyber: A CCA-Secure Module-Lattice-Based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 353–367. [CrossRef]
10. Gupta, N.; Jati, A.; Chauhan, A.K.; Chattopadhyay, A. PQC Acceleration Using GPUs: FrodoKEM, NewHope, and Kyber. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 575–586. [CrossRef]
11. Nguyen, D.T.; Dang, V.B.; Gaj, K. A High-Level Synthesis Approach to the Software/Hardware Codesign of NTT-Based Post-Quantum Cryptography Algorithms. In Proceedings of the 2019 International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, 9–13 December 2019; pp. 371–374. [CrossRef]
12. Roy, S.S.; Basso, A. High-Speed Instruction-Set Coprocessor for Lattice-Based Key Encapsulation Mechanism: Saber in Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *4*, 443–466. [CrossRef]
13. Huang, Y.; Huang, M.; Lei, Z.; Wu, J. A Pure Hardware Implementation of CRYSTALS-KYBER PQC Algorithm through Resource Reuse. *IEICE Electron. Express* **2020**, *17*, 20200234. [CrossRef]
14. Bisheh-Niasar, M.; Azarderakhsh, R.; Mozaffari-Kermani, M. Instruction-Set Accelerated Implementation of CRYSTALS-Kyber. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 4648–4659. [CrossRef]
15. Nguyen, T.T.; Nguyen, T.T.B.; Lee, H. An Analysis of Hardware Design of MLWE-Based Public-Key Encryption and Key-Establishment Algorithms. *Electronics* **2022**, *11*, 891. [CrossRef]
16. He, S.; Torkelson, M. Designing Pipeline FFT Processor for OFDM (De)modulation. In Proceedings of the 1998 URSI International Symposium on Signals, Systems, and Electronics, Pisa, Italy, 2 October 1998; pp. 257–262. [CrossRef]
17. Nguyen, T.T.B.; Lee, H. High-throughput low-complexity mixed-radix FFT processor using a dual-path shared complex constant multiplier. *J. Semicond. Technol. Sci.* **2017**, *17*, 101–109. [CrossRef]
18. Barrett, P. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. *Adv. Cryptol.–CRYPTO'86* **1987**, *263*, 311–323.
19. Montgomery, P.L. Modular Multiplication without Trial Division. *Math. Comput.* **1985**, *44*, 519–521. [CrossRef]
20. Bertoni, G.; Daemen, J.; Peeters, M.; Assche, G.V. Keccak and the SHA-3 Standardization. NIST, Gaithersburg, MD, USA, 6 February 2013. Available online: https://csrc.nist.gov/csrc/media/projects/hash-functions/documents/keccak-slides-at-nist.pdf (accessed on 27 March 2022).
21. *FIPS PUB 202–SHA3 Standard*; Permutation-Based Hash and Extendable-Output Function. Federal Information Processing Standards Publication: Gaithersburg, MD, USA, August 2015. [CrossRef]
22. Elkhatib, R.; Azarderakhsh, R.; Mozaffari-Kermani, M. Highly Optimized Montgomery Multiplier for SIKE Primes on FPGA. In Proceedings of the 2020 IEEE 27th Symposium on Computer Arithmetic (ARITH), Portland, OR, USA, 7–10 June 2020; pp. 64–71. [CrossRef]