JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL. 22, NO. 6, DECEMBER, 2022 ISSN(Print) 1598-1657
https://doi.org/10.5573/JSTS.2022.22.6.459 ISSN(Online) 2233-4866

Efficient Partially-parallel NTT Processor for Lattice-based Post-quantum Cryptography

Soyeon Choi[1], Yerin Shin[2], Kiho Lim[3], and Hoyoung Yoo[1,*]

***Abstract*—This paper presents a partially-parallel number theoretic transform (NTT) processor design for polynomial multipliers, which is a key component of a lattice-based cryptography. Since the data flow of NTT is similar to that of FFT, studies have been conducted to apply the FFT structure to fit the NTT structure. However, the previous architectures suffer from high hardware complexity and low throughput. Thus, we propose a new partially-parallel design that models the data reordering process and derives a generalized data reordering circuit. The proposed partially parallel design solved the problem of the previous architectures. Moreover, it provides improved performance through efficient data reordering. Synthesis results shows that the proposed 8-parallel 512-point NTT processor achieves 15% to 76% improvements in terms of hardware efficiency compared to the previous architectures. As a result, the proposed NTT processor is a good solution in a more diversified lattice-based crypto-processor with constrained usage conditions.**

***Index Terms*—Lattice-based cryptography, number theoretic transform, polynomial multiplier, post-quantum crypto-processer**

Manuscript received Jul. 20, 2022; reviewed Oct. 31, 2022; accepted Nov. 13, 2022
[1]Department of Electronics Engineering, Chungnam National University, Daejeon, 34134, Korea
[2]LX Semicon, Daejeon, 34027, Korea
[3]William Paterson University of New Jersey, 07470, US
E-mail : hyyoo@cnu.ac.kr

I. INTRODUCTION

Encryption for reliable security is one of the most important elements in modern communication systems [1]. However, the classic encryption techniques face a crisis of collapse due to the rapidly developing quantum computing technology [2]. It has been proven that modern cryptographic algorithms such as Ribest, Shamir and Adelman (RSA) and Elliptic Curve Cryptography (ECC) can be nullified within a short time [3] by the Shor quantum algorithm [4]. Mathematic problems such as prime factorization, which take a long time to calculate with current computing performance, are no longer valid as cryptographic algorithms in quantum computers. Therefore, studies on post-quantum cryptography (PQC) have been actively conducted centered on the National Institute of Standards and Technology (NIST), and standardization work has been carried out over three rounds since 2016 [5]. The PQC standardization work is selecting the final candidate based on five basic technologies: lattice-based cryptography [6], code-based cryptography [7], multi-variate-polynomial-based crypto-graphy [8], isogeny-based cryptography [9], and hash-based cryptography [10]. Among them, the lattice-based crypto-system is most frequently selected as the final candidate [11] because of its high cryptographic efficiency, affordable complexity, and fully homomorphic encryption (FHE) applicability [12, 13] and is perceived as the candidate with the most potential for the NIST standard.

In step with the PQC algorithm standardization work, studies on hardware development have been actively conducted [14-18]. The operation that requires the longest computation time in lattice-based cryptography is

polynomial multiplication [19], so studies to improve performance such as processing speed and hardware utilization efficiency have been intensively conducted centered on polynomial multiplication [20, 21]. Recently, many researchers have studied NTT-based polynomial multiplication algorithms and hardware [22, 23]. Whereas Fast Fourier Transform (FFT) carries out operations on a complex plane, Number Theoretic Transform (NTT) performs operations of values on a finite ring rather than complex numbers based on roots of unity. Because the hardware architecture of FFT has been studied for more than 50 years, a stably studied FFT architecture can be applied to NTT [24, 25]. Similar to the fully-parallel architecture of FFT, the fully-parallel architecture that maps the dataflow of NTT directly to hardware is intuitive and simple. However, identically to the fully-parallel architecture of the FFT [24], the fully-parallel architecture of the NTT also has a problem that hardware complexity increases linearly as the length of the NTT increases [26]. To solve this problem, a design that provides the partially parallel architecture of NTT through a feedback type and a feedforward type was proposed in a similar method applied in FFT [26-29]. Since the partially-parallel architecture of the NTT provides the desired timing as a delay element of the feedback loop, single-path delay feedback (SDF) [26] and multi-path delay feedback (MDF) [27] have been proposed as structures to support the partially-parallel architecture. In addition, a single-path delay commutator (SDC) [28] and multi-path delay commutator (MDC) [29] that provide the desired timing through an additional reordering circuit have been proposed. However, since SDF [26] and SDC [28] have low throughput due to serial operation, there are restrictions in real encryption hardware implementation, and in the case of MDF [27] and MDC [29], low hardware usage efficiency has been raised as a problem. In this paper, the data reordering sequences between processing elements by step are analyzed to propose an efficient partially-parallel NTT processor structure that can be applied to various parallel factors. The main contributions of this paper are summarized as follows:

1) The proposed design technique concentrates on the data rearrangement process, which provides matrix representation and draws a reordering circuit to present a comprehensive implementation method.
2) Since the proposed structure can be applied to any parallel factor, a hardware structure optimized to fit the given design constraints can be provided.
3) The proposed structure is a partially parallel architecture that solves the problem of parallelization restriction by the high radix of the MDC design [29] and the problem of huge hardware complexity of the MDF design [27].
4) The proposed structure achieves 100% hardware utilization efficiency to provide the best performance in terms of hardware complexity and throughput.

The rest of the paper is organized as follows. In the second chapter, the theoretical background necessary for this study is identified. Contents regarding NTT-based polynomial multiplication algorithms and general NTT processors are included. In the third chapter, the previous partially-parallel NTT processor architecture is examined, and its problems are analyzed. In the fourth chapter, a design technique for a new partially parallel architecture is proposed, and then in the fifth chapter, the performance of the NTT processor designed by applying the relevant technique is evaluated. Finally, in the last chapter, the conclusion of this study is presented.

II. BACKGROUND

1. NTT Polynomial Multiplication

Lattice-based cryptosystems carry out the encryption and decryption of problems such as Ring-Learning with Error (Ring-LWE) [30, 31] and Module Learning with Error (Module-LWE) [32] by performing polynomial addition, multiplication, and modulo operations on a finite ring. Because polynomial multiplication has higher complexity compared to other arithmetic operations, it is essential to implement efficient polynomial multiplication for lattice-based cryptographic hardware. The finite ring is defined as $R_q = \mathbb{Z}_q[x]/f(x)$, where $q \equiv 1 (\mathrm{mod}\ 2N)$ is a prime number and $f(x) = x^N + 1$ is an irreducible polynomial of degree N [33]. Any two polynomials $a(x)$, $b(x)$ on the finite ring R_q can be expressed as

$$a(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1}$$
$$b(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{N-1} x^{N-1}. \quad (1)$$

A polynomial $c(x)$ of length $2N-1$ can be obtained by multiplying two polynomials of length N, and the coefficient of $c(x)$ can be calculated with (2).

$$c_i = \sum_{n=0}^{i} a_n b_{i-n}\ , \quad (2)$$

Since (2) representing polynomial multiplication is the same as the operation to obtain the discrete convolution $c(x) = a(x) * b(x)$, polynomial multiplication can be calculated with discrete convolution. However, the convolution operation in (2) has a large operation quantity of $O(N^2)$, so it is generally processed as multiplication through domain transformation. The coefficient of the polynomial can be changed into the frequency domain using Discrete Fourier Transform (DFT). Instead of the time domain convolution operation, pointwise multiplications for coefficients of $a(x)$ and $b(x)$ are carried out, in the frequency domain. Finally, inverse DFT (IDFT) of $c(x)$ in the frequency domain is performed to obtain the $c(x)$ of the time domain.

$$c(x) = a(x) * b(x)$$
$$= IDFT(DFT(a(x)) \odot DFT(b(x)))\ , \quad (3)$$

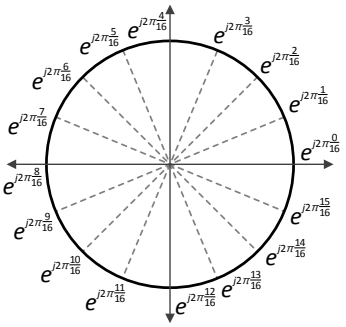
In this case, the lengths of $a(x)$ and $b(x)$ must be adjusted to $2N-1$, which is the length of $c(x)$, for DFT and IDFT, and zero-padding is necessary [23].

However, the DFT and IDFT also have an operation quantity of $O(N^2)$. The FFT algorithm, well known as a method to transform the domain quickly by reducing the operation quantity, is applied. The DFT of (4) can reduce the operation quantity to $O(N \log N)$ as shown in (5) through the divide-and-conquer method by using the periodic and conjugate properties of W_N^{ij} .

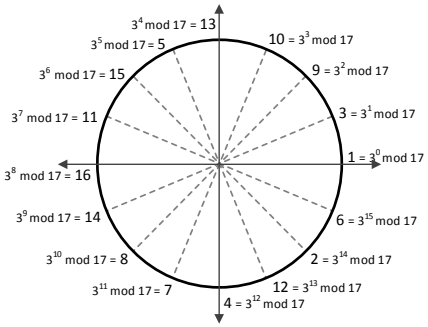
$$A_i = \sum_{j=0}^{N-1} a_j W_N^{ij}\ , \quad (4)$$

$$A_i = \sum_{j=0}^{(N/2)-1} a_{2j} W_{N/2}^{ij} + W_N^i \sum_{j=0}^{(N/2)-1} a_{2j+1} W_{N/2}^{ij}\ , \quad (5)$$

Furthermore, a complex number-based operation for FFT can be replaced with a integer number-based operation by employing NTT while maintaining the properties of FFT [22, 23]. As shown in Fig. 1(a), in FFT, $W_N = e^{j2\pi/N}$ is defined as a primitive root, and as shown in Fig. 1(b), in NTT, $W_N \bmod q = 1$ is defined as a primitive root [34]. NTT and inverse NTT (INTT) are expressed with the following (6) and (7).



(a) FFT for $N = 16$



(b) NTT for $N = 16$

Fig. 1. Primitive root for $N = 16$.

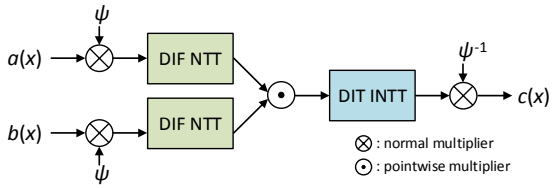


Fig. 2. Overall structure for NTT polynomial multiplication.

$$A_i = \sum_{j=0}^{N-1} a_j W^{ij} \bmod q\ , \quad (6)$$

$$a_i = N^{-1} \sum_{j=0}^{N-1} A_j W^{-ij} \bmod q\ , \quad (7)$$

Accordingly, (3) representing the DFT-based polynomial multiplication operation [35] can be expressed as (8) representing NTT-based polynomial multiplication.

$$c(x) = a(x) * b(x)$$
$$= INTT(NTT(a(x)) \odot NTT(b(x)))\ , \quad (8)$$

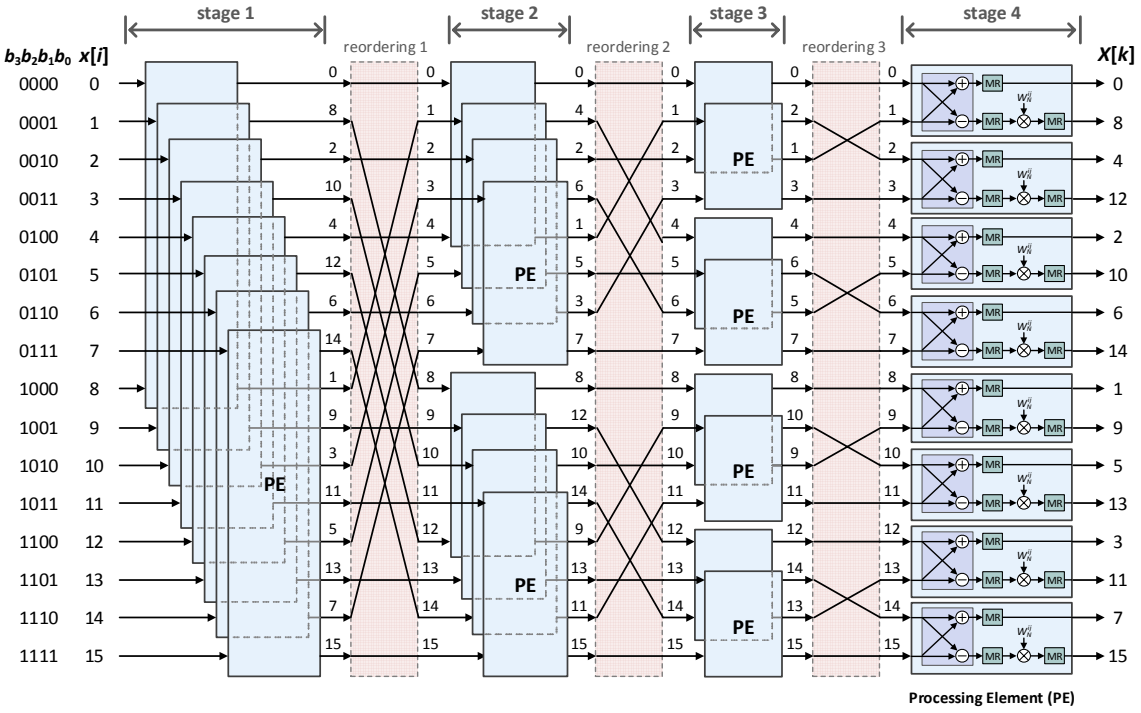


Fig. 3. Fully-parallel architecture for $N = 16$.

Finally, a negative wrapped convolution that efficiently removes the zero-padding that occurs in the NTT and INTT processes of (8) was proposed [36]. According to [36], the zero-padding process that may occur during NTT-based polynomial multiplication can be omitted by multiplying the coefficients of the two polynomials by ψ that satisfies $\psi \equiv W \bmod q$. The two polynomial coefficients multiplied by ψ are expressed as $a'(x) = a_0 + \psi a_1 x + \cdots + \psi^{n-1} a_{n-1} x^{n-1}$ and $b'(x) = b_0 + \psi b_1 x + \cdots + \psi^{n-1} b_{n-1} x^{n-1}$. As a result, (8) can be expressed

$$c(x) = \psi^{-1} INTT(NTT(a'(x)) \odot NTT(b'(x))) . \quad (9)$$

In conclusion, polynomial multiplication using NTT can be performed reducing the computational complexity from $O(N^2)$ to $O(N \log N)$ through integer operation instead of complex numbers.

2. General NTT Processor

Fig. 2 shows a typical polynomial multiplier architecture that implements (9), where DIF and DIT represent decimation-in-frequency (DIF) and decimation-in-time (DIT), respectively [35]. If the same decimation is used for all NTTs and INTTs, it is inevitable to add a bit-reversal circuit to every NTT or INTT [35]. In order to completely remove the added bit-reversal circuit, the DIF structure, in which the output is bit-reversal converted when the inputs are sequential, is generally applied to the front NTT, and the DIT structure, in which bit-reversal inputs are sequentially converted and output, is applied to the rear INTT as shown in Fig. 2. Since the DIT and DIF algorithms are structurally symmetric [35], and one of the NTT and INTT structures can be easily derived by applying the other structure, this paper will mainly explain the structure of the DIF NTT processor.

A. Fully-parallel Architecture

Fig. 3 shows the DIF NTT processor of a fully-parallel architecture with $N = 16$, $q = 17$, and $W_N = 3$. NTT for $N = 16$ points is performed over n steps, and $N/2$ processing elements (PEs) are required for each step, where n is $\log_2 N$. In order to perform the NTT operation in (6), PE consists of one radix-2 butterfly unit, three modular reduction units (MR), and one multiplier. The basic PE is similar to the structure of the FFT, but in accordance with the characteristics of the lattice-based cryptoprocessor that must be operated on R_q, modular reduction is additionally required in the transformation

process over n steps. In this paper, among various implementations including Barrett and Montgomery modular reduction [37-39], [39] is used to limit the result of each operation to $\log_2 N = 4$ bits.

As shown in Fig. 3, PE operation is performed at every s stage ($1 \le s \le n$) and the PE of the next stage is performed after rearranging the data. According to [24], data reordering performs rearrangement while satisfying the following properties.

NTT pair property: pairs of data processed simultaneously in the PEs at stage s differ in b_{n-s} for any N-point NTT with $n = \log_2 N$ stages.

For instance, the PEs begin by operating on pairs of data with indexes (0,8), (1,9), (2,10), (3,11), and so forth after the first stage. By converting these indices to binary, we obtain (0000,1000), (0001,1001), (0010,1010), and (0011,1011). Comparing the indices in each pair reveals that they are identical except for the most significant bit, $b_3 = b_{4-1}$. This is true for all the pairs of indices at stage 1. By repeating the analysis for stage 2 and 3, it is clear that the indices of pairs of data processed by the PEs differ only in b_2 and b_1, respectively. As a result, we can conclude that the NTT architecture is correct only when the data pairs input during the same clock period in all PEs differ in the index bits b_{n-s}. This will serve as the foundation for the following explanations of the other structures.

B. Partially-parallel Architecture

As can be observed in Fig. 3, since the fully-parallel architecture simultaneously computes N data through a total of $(N/2)\log_2 N$ PEs and reordering networks by stage, it has high throughput of N points per clock cycle. However, in terms of hardware complexity, it has a severe disadvantage that the required number of PEs increases linearly as N increases. To alleviate the high hardware complexity of the fully-parallel architecture, a partially- parallel architecture that simultaneously processes $P = 2^p$ data according to parallel factor P has emerged. Fig. 4 shows the general structure of the partially-parallel NTT. The partially-parallel structure is composed of n stages connected in series with data flowing from stage 1 to stage n. Since each stage contains P inputs and P outputs, N data points are

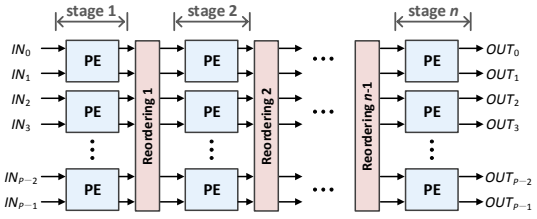
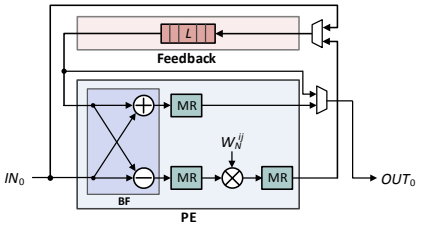
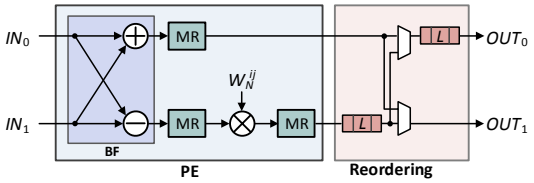


Fig. 4. General partially-parallel architecture.



(a) Feedback architecture



(b) Feedforward architecture

Fig. 5. Feedback and feedforward architecture.

Table 1. The previous architectures.

Type	Architecture	Parallelism
Feedback	SDF	Serial
	MDF	Partially-parallel
Feedforward	SDC	Serial
	MDC	Partially-parallel

subdivided into N / P number of P points and thus the throughput becomes P data per clock cycle. Note that each stage s of the architecture performs all calculations associated with one stage of the NTT algorithm.

To convert the fully-parallel architecture shown in Fig. 3 to the partially-parallel architecture shown in Fig. 4, data rearrangement must be considered. The output pair with the natural order of s stage is rearranged while satisfying the pair property in the next s + 1 stage. The partially-parallel architecture is largely divided into a feedback type [26, 27] and a feedforward type [28, 29] according to the data rearrangement processing method. First, the feedback type shown in Fig. 5(a) proceeds with data reordering using a feedback loop. In this case, some output of butterfly unit (BF) is fed back as the delay element at the same stage through the feedback loop, so

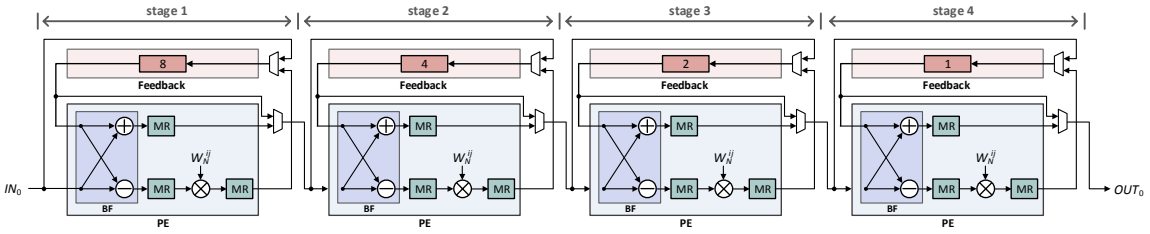


Fig. 6. The SDF architecture for $N = 16$.

the operation of the input pair where a difference of b_{n-s} occurs in the next stage. On the other hand, in the feedforward type shown in Fig. 5(b), the data processed from the PE of each stage uses a separate reordering circuit to match the input pair with a difference of b_{n-s} in the next stage.

III. PREVIOUS WORKS

The previous NTT architectures are classified in Table 1. The table denotes feedback and feedforward types [26-29]. First, the feedback type satisfies the NTT pair property by providing the desired timing with the delay elements via the feedback loop. They are classified as single-path delay feedback (SDF) [26] or multi-path delay feedback (MDF) [27] based on their serial or parallel processing configurations. Second, the feedforward type accomplishes the NTT pair property with an additional reordering circuit following PE. They are classified as single-path delay commutator (SDC) [28] or multi-path delay commutator (MDC) [29], based on their serial or parallel processing capabilities.

1. Single-path Delay Feedback (SDF) Design

Fig. 6 shows the $N = 16$ point SDF design, which is a representative design based on feedback. The SDF design has serial data flow because it uses a single path and satisfies the NTT pair property using the delay element of the feedback loop [26]. SDF receives data at each stage in the natural order of the index from 0 to 15. For serial data flow, pairs of data that differ by b_{n-s} arrive with a difference of 2^{n-s} clock cycles according to the natural order. At each stage, a buffer of length $L = 2^{n-s}$ is used to store these pairs of data concurrently. Thus, the buffer's output is calculated simultaneously with the stage's input in the PE. Following that, one of the butterfly's outputs is sent to the multiplier, while the other is placed in the buffer. As illustrated in Fig. 6, each stage consists of one BF, one multiplier, three MRs, and a FIFO with a length of $L = 2^{n-s}$. When all hardware resources are added for all stage $s = \{1, \cdots, n\}$, $\log_2 N$ BFs, $\log_2 N$ multipliers, $3\log_2 N$ MRs, and $N-1$ FIFO are required in total. Compared to the fully-parallel architecture shown in Fig. 3, the SDF design [26] can overwhelmingly improve the hardware complexity problem because it uses a single path. However, it has the problem that the hardware throughput is processing one data per clock due to serial data flow, and hardware utilization is reduced to 50% due to the feedback loop.

2. Multi-path Delay Feedback (MDF) Design

The MDF design [27] is the parallel version of the SDF design [26]. Fig. 7 shows a 16-point 4-parallel MDF design, which consists of the first independent serial PEs and the second combining parallel PEs. Let us compare the SDF in Fig. 6 and MDF in Fig. 7. Whereas data in the SDF [26] is processed in series in natural order from stage 1 to n = 4, MDF processes P data flow independently through the first serial PEs and combines P data flow through the second parallel PEs [27]. The first serial PEs are the same as the original SDF [26] except for the fact that the length of the buffers in MDF [27] is divided by P with respect to those in the SDF [26]. Since each independent serial part provides parallel streams that differ in the bit b_{n-s}, the remaining parallel part simply combines those parallel streams with shuffling circuits. The general P-parallel MDF [27] uses $P\log_2(N/P) + (P/2)\log_2 P$ BFs, $P\log_2(N/P) + (P/2)\log_2 P$ multi-pliers, $3(P\log_2(N/P) + (P/2)\log_2 P)$ MRs, and $N-P$ FIFO. Consequently, the MDF design [27] can parallelize the SDF design [26] by $P = 2^p$ to improve the throughput by P times. However, it uses

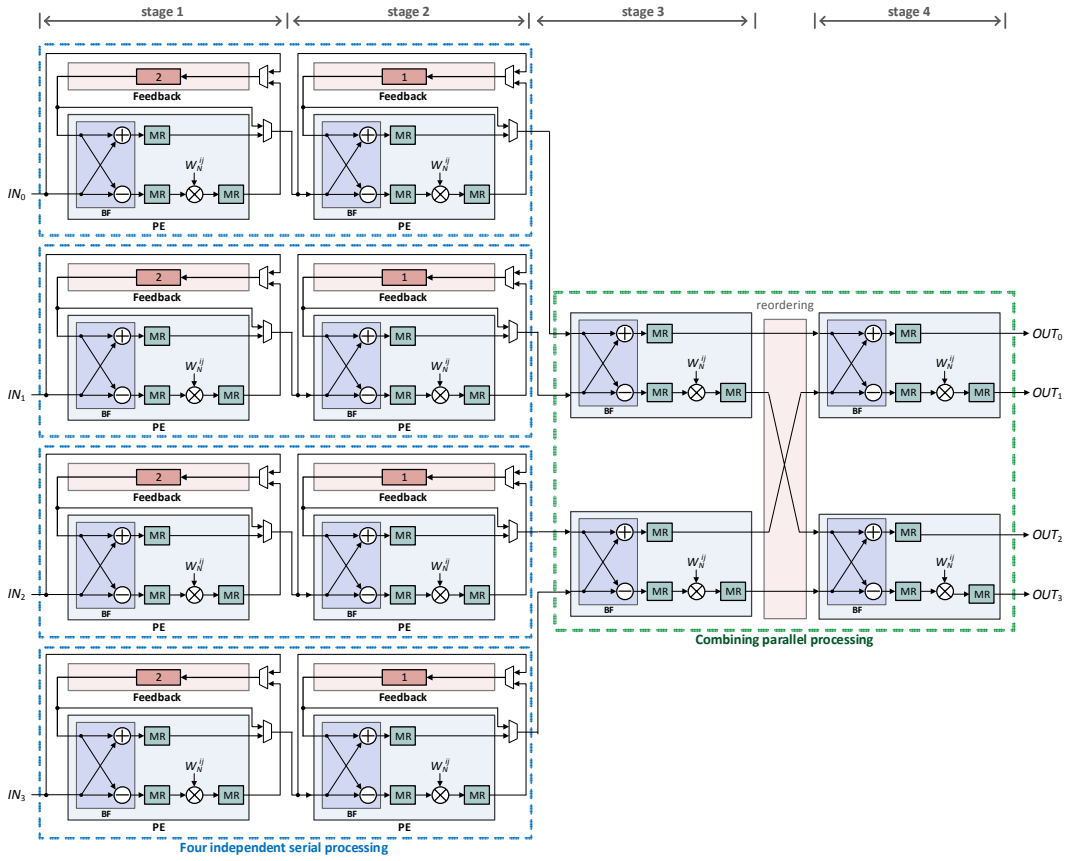


Fig. 7. The MDF architecture for $N = 16$ and $P = 4$.

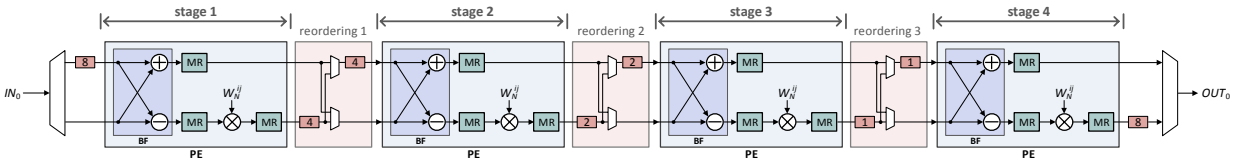


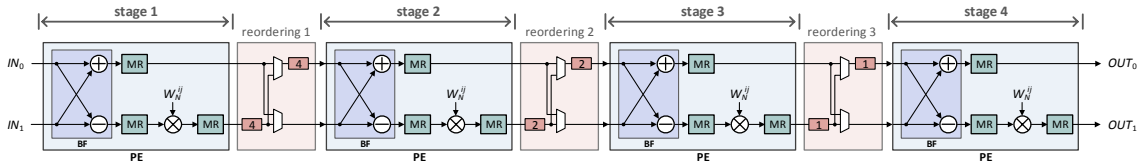
Fig. 8. The SDC architecture for $N = 16$.

approximately P times more hardware than the SDF design [26], and like the SDF design [26], it fails to achieve full hardware utilization.

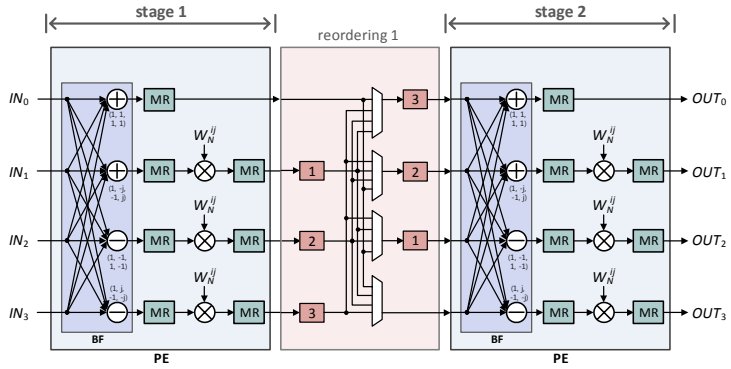
3. Single-path Delay Commutator (SDC) Design

In comparison to feedback designs [26, 27], feedforward designs [28, 29] incorporate additional reordering circuitry to ensure that the NTT pair property is satisfied. Fig. 8 illustrates a 16-point SDC for serial data processing. Despite the fact that the SDC processes serial data [28], it is based on the 2-parallel MDC depicted in Fig. 9(a). Let us begin with the 2-parallel MDC [29]. In 2-parallel MDC [29], it is assumed that input data differentiated in bit arrive, but the data order is not consistent between stages, necessitating the participation of shuffling circuits to reorder the data as shown in Fig. 9(a). The reordering circuits consist of upper and lower paths, each of which demands buffers and multiplexers to satisfy the NTT pair property. First, the lower path is delayed using the buffers, and then both upper and lower sets of data are swapped using multiplexers. Lastly, the upper path is delayed using buffers, resulting in an aligned data pair.

The SDC in Fig. 8 receives input data in series. The only difference between SDC [28] and 2-parallel MDC [29] is in the data input and output. As shown in Fig. 8, the first half of the data is routed through the input buffer, while the second half is connected to the SDC's lower input [28]. Finally, the output is serialized once again.



(a) The 2-parallel MDC architecture



(b) The 4-parallel MDC architecture.

Fig. 9. The MDC architecture for $N = 16$ and $P = 2$ and 4.

Due to data parallelization, SDC [28] is only operational 50% of the time, with the remaining 50% used to receive and produce data. The general SDC employs a total of $\log_2 N$ BFs, $\log_2 N$ multipliers, $3\log_2 N$ MRs, and $2(N-1)$ FIFO in terms of hardware resources.

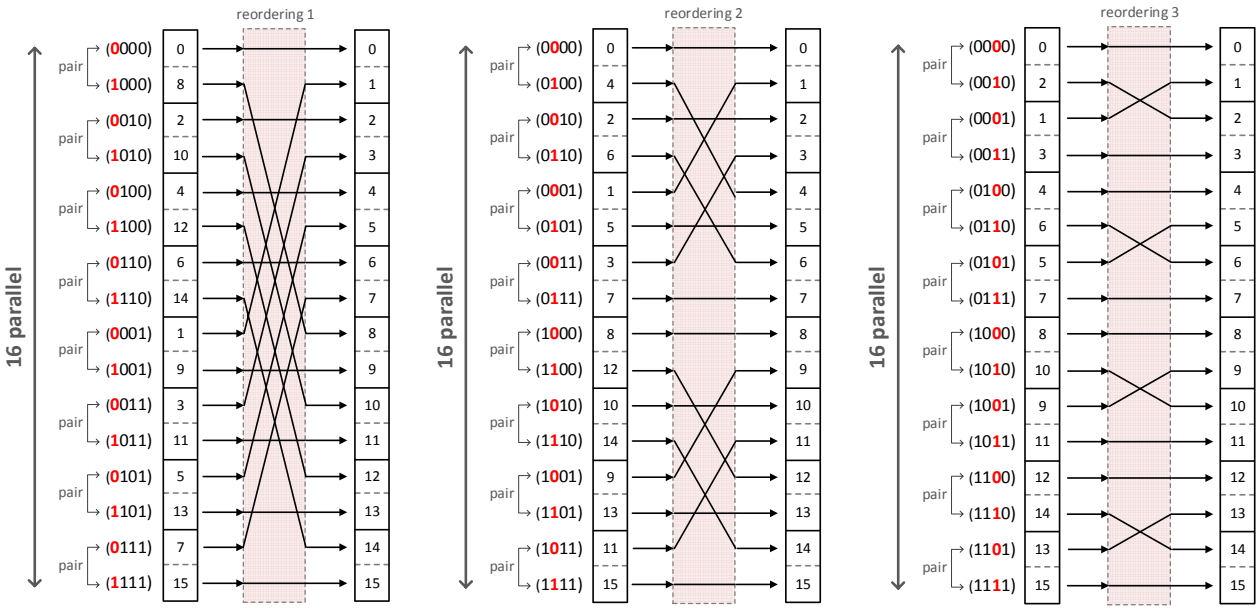
4. Multi-path Delay Commutator (MDC) Design

The MDC design has a parallel design with a higher parallel factor because it uses high-radix for radix-2 based 2-parallel MDC [29]. Fig. 9(a) and (b) show the 2-parallel MDC and 4-parallel MDC designs, respectively. The 2-parallel MDC [29] has the same structure as the SDC design [28] after removing additional input/output circuits required for serial processing. Fig. 9(a) processes two data per one PE and processes two data per clock based on radix-2, and one stage of the 2-parallel MDC shown in Fig. 9(a) carries out the operation assigned to one stage of the fully-parallel architecture [26]. On the other hand, Fig. 9(b) processes four data per one PE and four data per clock based on radix-4. One stage of the 4-parallel MDC in Fig. 9(b) performs the operations assigned to two successive stages of the 2-parallel MDC. A typical P-parallel MDC design [29] requires $P\log_p N$ BFs, $(P-1)\log_p N$ multipliers, $2(P-1)\log_p N$ MRs, and $N-P$ FIFO.

In conclusion, the P-parallel MDC design [29] achieves a throughput improvement by P times by changing the 2-radix structure of the 2-parallel MDC design to the P-radix structure. Due to the limit on the use of the trivial multiplier, the high radix application of the NTT does not significantly improve performance. In addition, it has the disadvantage that hardware utilization is limited by P times due to the input/output delay circuit.

IV. Proposed Partially-Parallel NTT Design

Although the MDF design [27] and the MDC design [29] succeeded in achieving high throughput by converting the SDF design [26] and the SDC design [28] into a parallel architecture, respectively, the resultant increase in hardware complexity and decrease in hardware utilization are among the problems that must be solved. Therefore, we propose a new partially-parallel NTT architecture that can improve hardware performance by combining the advantages of MDF [27] and MDC design [29]. Similar to the overall structure of MDF [27], the proposed partially-parallel NTT architecture composes many independent serial structures on the front, and a parallel structure combining them on the rear. For the technique to design a partially-parallel NTT, the reordering structure that satisfies the NTT pair



(a) Reordering 1 (b) Reordering 2 (c) Reordering 3

Fig. 10. Reordering analysis of (a) reordering 1; (b) reordering 2; (c) reordering 3.

property is mainly analyzed and a partially-parallel NTT structure is presented based on the analysis. In this paper, the 4-parallel structure for 16-point NTT is used as an example without loss of generality. The advantages of the proposed partially-parallel NTT design are emphasized through direct comparison with 4-parallel MDF and MDC. Finally, the structures of parallel factors 4, 8, and 16 are presented for the 512-point NTT structure to provide a practical example and show that it can be easily applied to various parallel factors.

1. Proposed Partially-parallel NTT Algorithm

The proposed new partially-parallel NTT design begins with analyzing the data rearrangement processes to transform them to fit the given parallel factors. Fig. 10 shows the rearrangement processes extracted from the 16-point fully-parallel architecture in Fig. 3. The input of the reordering circuit has a pair that differs only in b_{n-s} bits at stage s and satisfies the pair property, and the output of the reordering circuit are described in natural order. For example, in the fully-parallel architecture for $N = 16$ points in Fig. 3, the NTT algorithm is performed from stage 1 to 4, in reordering 1 shown in Fig. 10(a), there is a difference in $b_{4-1} = b_3$ in each pair, and in reordering 2 shown in Fig. 10(b), there is a difference in the $b_{4-2} = b_2$ bit of each pair. Similarly, in reordering 3 shown in Fig. 10(c), it can be seen that b_1 of the pair are different. In the previous NTT structure including SDF [26], MDF [27], SDC [28], and MDC [29], it can be seen that the NTT pair property is satisfied for all data pairs in the NTT processing process regardless of serial and parallel data flow, feedback and feedforward types [26-29].

In order to derive the proposed partially-parallel architecture based on Fig. 10, a new scheduling task should be performed that transforms each of $N\times1$ matrices into $P\times(N/P)$ matrices. Fig. 10 and 11 represent 16×1 matrices and 4×4 matrices for reordering, respectively. For matrix representation, the column of the matrix means one clock cycle and the row means the number of data processed per one clock cycle. Note that P row determines the number of PEs as $P/2$ to process P data in parallel. The $P\times(N/P)$ matrix represents P data in one clock for N/P clock cycles to process the entire N data. Whereas the 16×1 matrices shown in Fig. 10 rearranges sixteen data at a single clock in a fully-parallel manner, the 4×4 matrices shown in Fig. 11 processes four data for four clock cycles in a partially-parallel manner. As with the transformed 16×1 matrix where the NTT pair property is maintained, the NTT pair property is also maintained in all 4×4 matrices.

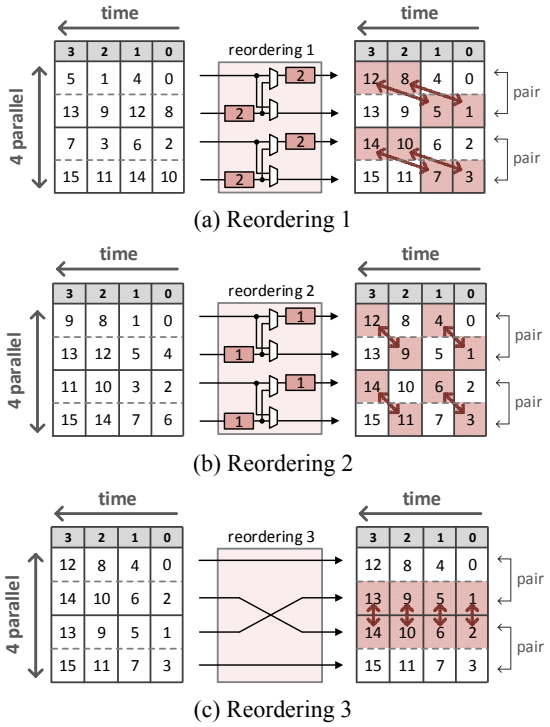


Fig. 11. 4×4 matrix representation for (a) reordering 1; (b) reordering 2; (c) reordering 3.

More importantly, through transformation into 4×4 matrices, the data pair that needs swapping can be identified out of the entire 16 data. The pairs that require swapping are highlighted in Fig. 11 and summarized as follows:

Stage 1: (1,8), (3,10), (5,12), (7,14)
Stage 2: (1,4), (3,6), (9,12), (11,14)
Stage 3: (1,2), (5,6), (9,10), (13,14)

As shown in Fig. 11, the swapping pairs in stages 1 and 2 require delay factors as much as 2 and 1, and the swapping pairs in stage 3 does not require any delay.

In order to satisfy the NTT pair property by changing the 16×1 matrix of the fully-parallel architecture to the 4×4 matrix of the partially-parallel architecture, the design of a reordering circuit that performs swapping to an appropriate clock is carefully designed. In this case, the required hardware structure varies depending on whether there is a delay between swapping data and inter/intra-PE. To explain in more detail, the reordering circuit used in SDC [28] and MDC [29] is suitable in cases where delay is required, as with stage 1 and stage 2, and swapping occurs in intra-PE. Upper and lower paths of reordering circuits can be exchanged with the desired delay difference through the FIFO and multiplexers of each path. Additionally, as with stage 3, when swapping occurs between inter-PEs without requiring delay, it is easy to implement simple hardwiring similar to the last combining circuitry of MDF [27]. Since the replacement data pair already occurs in the same clock, data rearrangement can be completed with a simple interconnection without delay.

2. Proposed Partially-parallel NTT Structure

With the matrix transformation, analysis, and derivation of a rearrangement circuit, the finally proposed 4-parallel structure for 16-point NTT is as shown in Fig. 12. Basically, since P data are processed in parallel per clock cycle, each stage consists of $P/2$ PEs, and a rearrangement circuit depicted in Fig. 11 was added between each stage to operate to fit the matrix operation. Between stage 1 and stage 2, a feedforward type reordering circuit was built with delay times 2 and 1, and in stage 3, they were interconnected with hardwiring. In conclusion, two independent MDC designs [29] operate in 2-parallel in the front part of the proposed partially-parallel NTT architecture, and parallel structures similar to MDF design [27] combine independent streams in the rear part. In other words, the overall structure is similar to the MDF design in Fig. 7, but the internal structure is composed of the 2-parallel MDC design in Fig. 9, not the feedback-based SDF in Fig. 6. To sum up, the proposed NTT processor requires $(P/2)\log_2 N$ BFs, $(P/2)\log_2 N$ multipliers, $3(P/2)\log_2 N$ MRs, and $N-P$ FIFO for partially-parallel P. The proposed partially-parallel architecture can achieve 100% hardware utilization because it does not internally include the feedback loop that MDF [27] has and minimizes the waste of the delay element of the reordering circuit due to the high radix of MDC [29]. In conclusion, it is possible to provide higher throughput compared to SDF [26] and SDC [28] while absorbing all the advantages of MDF [27] and MDC [29]. Consequently, the design method for the proposed partially-parallel architecture can be described as follows:

1) Create $n-1$ pieces of $N\times1$ matrices by reordering for a fully-parallel architecture

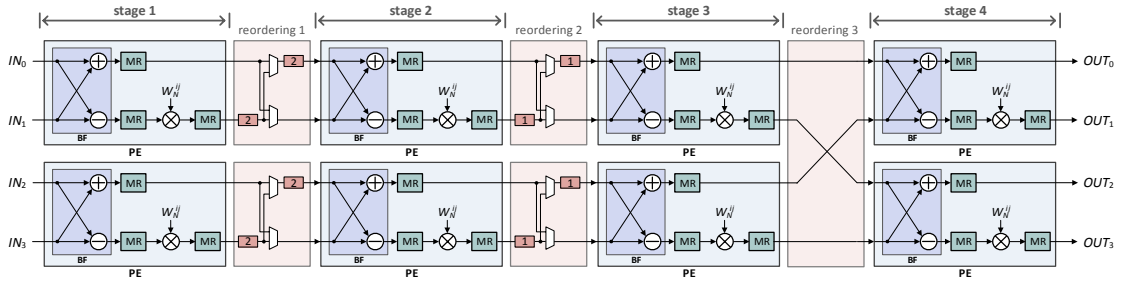


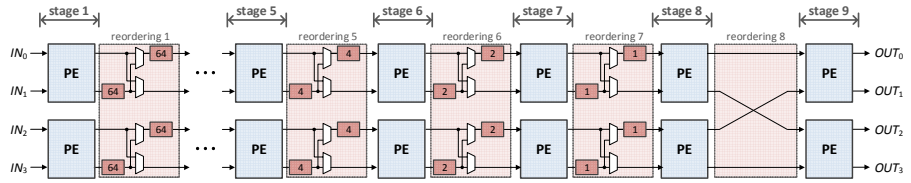
Fig. 12. The proposed partially-parallel design for $N = 16$ and $P = 4$.

2) Transform the $N \times 1$ matrices into $n-1$ pieces of $P \times (N/P)$ matrices by stage for a partially-parallel architecture
3) Check swapping data by stage and configure the reordering circuit
4) Place $P/2$ PEs for each stage and interconnect them with the reordering circuit

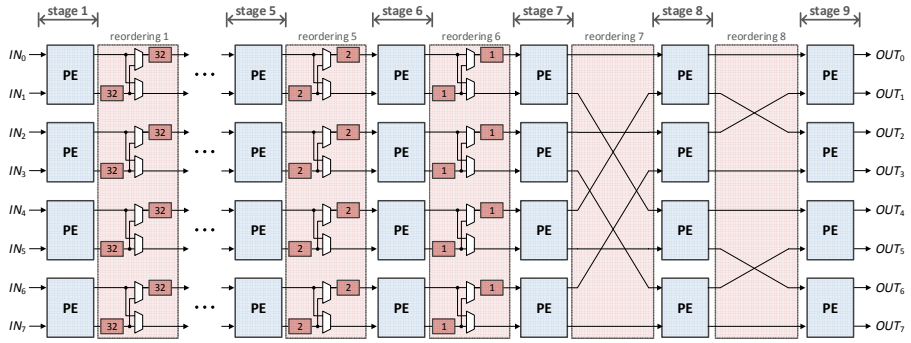
Fig. 13 shows practical examples of N = 512, q = 12,289, W = 3 using the design method above. While changing the parallel factor to 4, 8, and 16, it can be seen that the above systematic design method is applied regardless of the NTT length and parallel factor. Note that if the parallel factor P is 2, the proposed partially-parallel structure will become a 2-parallel MDC [29], and if the parallel factor P is equal to the NTT length N, the proposed partially-parallel architecture will become a fully-parallel architecture [26]. It is important to note that since the proposed partially-parallel architecture proposes a comprehensive structure and can be applied to any P, it is possible to implement the most efficient NTT processor under the constraints by selecting a P suitable for the design environment.

V. Experimental Results

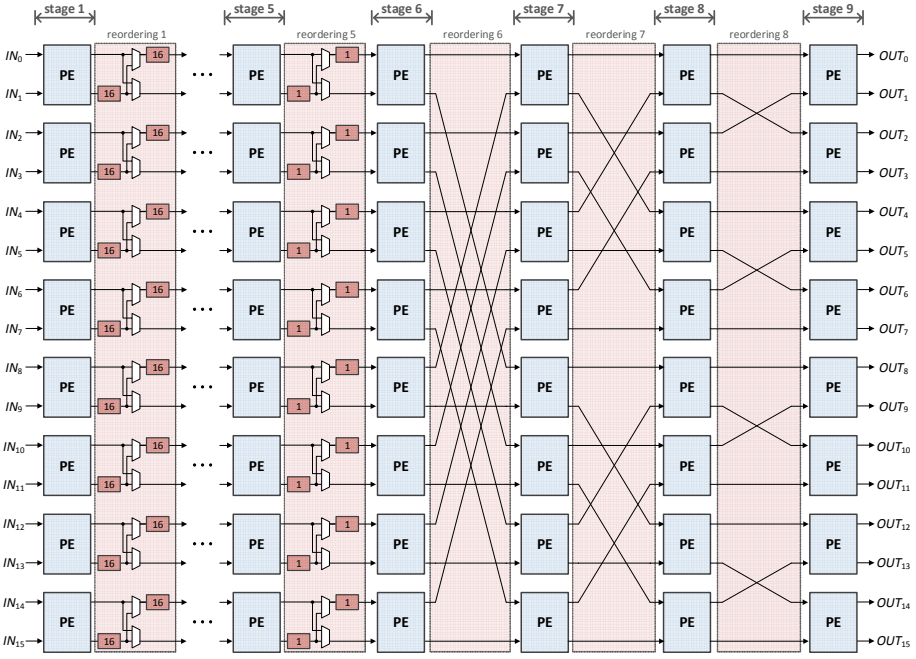
Table 2 shows a quantitative comparison between the proposed partially parallel NTT processor structure and the previous structures in terms of hardware resources, latency, throughput, and utilization. Note that utilization is defined as the ratio of the number of clocks during which PE conducts valid and effective operations to the total number of running clocks in order to demonstrate the efficiency of the hardware in addition to the typical hardware performance metrics. Representative hardware performance metrics were compared for the proposed NTT processor and the previous fully parallel architecture, serial architectures, and partially parallel architectures. In this case, N represents the length of the NTT and P represents a parallel factor. As shown in Fig. 3, the fully parallel architecture is intuitive and can support high throughput by processing N data per clock [26]. However, it has a problem that hardware complexity increases linearly along with the increase in the NTT length, and in fact a high throughput of processing N data per clock cycle is not required in a realistic system. Next, the SDF [26] in Fig. 6 and SDC [28] in Fig. 8, which are serial structures, satisfy the NTT pair property through the feedback type and feedforward type, respectively, and can serialize a fully-parallel architecture. Although they can reduce hardware including large amounts of butterflies, multipliers, modular reductions, and memories compared to the fully-parallel architecture [26], they provide a low throughput of processing one data per clock. Finally, the MDF [27] in Fig. 7 and MDC [29] in Fig. 9 designs can partially parallelize SDF [26] and SDC [28] to provide affordable hardware complexity while providing realistic throughput. However, they waste delays due to the inefficient data rearrangement circuit or impose limitations on the use of high radix structures, resulting in low level of hardware utilization. The proposed partially-parallel NTT design in Fig. 12 solves the problem by utilizing the advantages of both designs. Due to the efficient reordering circuit, the proposed partially-parallel design has the lowest hardware complexity quantitatively among partially-parallel structures regardless of NTT length N and parallel factor P. Moreover, Table 3 provides a numerical comparison by substituting the length N for 512 and the parallel factor P for 8. Through Table 2 and 3, the advantages of each architecture including fully-parallel, serial, and partially-



(a) Proposed partially-parallel design for $N = 512$ and $P = 4$



(b) Proposed partially-parallel design for $N = 512$ and $P = 8$



(c) Proposed partially-parallel design for $N = 512$ and $P = 16$

Fig. 13. The proposed partially-parallel design for $N = 512$ and $P = 4$, 8, and 16.

parallel architectures can be clearly confirmed. Moreover, it can also be confirmed that the proposed partially-parallel design always has low hardware complexity, short latency, and high hardware utilization among partially-parallel architectures.

Finally, to bring a practical comparison, Table 4 compares the synthesis results for all NTT designs in Verilog HDL with 200 MHz operating frequency using a CMOS 180 nm process. Without loss of generality, all NTT processors were maintained at $N = 512$, $q = 12{,}289$, $W = 3$, and $P = 8$. The hardware complexity, latency, throughput, and hardware efficiency information representing the performance of the entire structure were indicated. The hardware complexity was defined as the number of 2-input NAND gates, and the efficiency obtained by dividing the throughput by the complexity

Table 2. Comparison on hardware complexity.

Architecture	Fully parallel	Serial		Partially Parallel (P)		
Design	Fully parallel [26]	SDF [26]	SDC [28]	MDF [27]	MDC [29]	Proposed
Butterflies	$(N/2)(\log_2 N)$	$\log_2 N$	$\log_2 N$	$P\log_2(N/P)$ $+(P/2)(\log_2 P)$	$P^2 \log_p N$	$(P/2)(\log_2 N)$
Multipliers	$(N/2)(\log_2 N)$	$\log_2 N$	$\log_2 N$	$P\log_2(N/P)$ $+(P/2)(\log_2 P)$	$(P-1)\log_p N$	$(P/2)(\log_2 N)$
Modular reduction	$3(N/2)(\log_2 N)$	$3(\log_2 N)$	$3(\log_2 N)$	$3P\log_2(N/P)$ $+3(P/2)(\log_2 P)$	$2(P-1)\log_p N+1$	$3(P/2)(\log_2 N)$
Delays	-	$N-1$	$2(N-1)$	$N-P$	$N-P$	$N-P$
Latency	1	$N-1$	$N-1$	$(N/P)-1$	$2N/P-1$	$(N/P)-1$
Throughput	N	1	1	P	P	P
Utilization	100 %	50 %	50 %	$50(1+\log_N P)$ %	100 %	100 %

Table 3. Comparison on hardware complexity for $N = 512$

Architecture	Fully parallel	Serial		Partially Parallel ($P = 8$)		
Design	Fully parallel [26]	SDF [26]	SDC [28]	MDF [27]	MDC [29]	Proposed
Butterflies	2,304	9	9	60	192	36
Multipliers	2,304	9	9	60	27	36
Modular reduction	6,912	27	27	180	55	108
Delays	-	511	1,022	504	504	504
Latency	1	511	511	63	127	63
Throughput	512	1	1	8	8	8
Utilization	100 %	50 %	50 %	66 %	100 %	100 %

Table 4. Synthesis result for $N = 512$

Architecture	Fully parallel	Serial		Partially Parallel ($P = 8$)		
Design	Fully parallel [26]	SDF [26]	SDC [28]	MDF [27]	MDC [29]	Proposed
Gate count [#NAND]	39,789K	196K	257K	966K	955K	553K
Latency [ns]	5	2,555	2,555	315	635	315
Throughput [Gbps]	102.4	0.23	0.23	1.62	0.81	1.62
Efficiency [Kbps/#NAND]	2.57	1.02	0.78	1.68	0.84	2.94

was additionally used to make it easier to understand the results of performance improvement of the proposed structure. Consequently, it can be seen in the synthesis results in Table 4 that the proposed structure shows the best efficiency by efficiently implementing the rearrangement circuit. In more detail, the proposed partially-parallel architecture achieved 15% efficiency improvement thanks to the smaller hardware usage compared to the fully-parallel architecture [26], and achieved efficiency improvements of 66% and 74% thanks to the higher throughput compared to the serial architectures of SDF [26] and SDC [28], respectively. With the improvement of the rearrangement structure compared to MDF [27] and MDC [29], which are partially parallel architectures, efficiency improvements of 43% and 76%, respectively, were achieved. Although Table 4 presents the synthesis results only for $P = 8$, it can be inferred that it has the highest efficiency for any P according to Table 2. Therefore, the proposed NTT design can respond flexibly to the lattice-based post-quantum cryptosystem that has various design requirements.

VI. CONCLUSION

In this paper, a new partially parallel NTT processor architecture applicable to polynomial multiplication was proposed to improve the cryptographic processor efficiency of lattice-based post-quantum cryptography. The proposed structure analyzes the data rearrangement processes as matrices derives and a data rearrangement circuit and the entire structure using the generalized data rearrangement scheduling process. As a result, the problems of the high complexity of the previous fully-parallel architecture [26] and low throughput of the serial architecture SDF [26] and SDC [28] designs were solved. It provides improved performance in terms of hardware complexity and utilization through efficient data rearrangement compared to partially parallel MDF [27] and MDC [29]. Consequently, the proposed structure enables the design of a partially parallel *N*-point NTT processor that is easily optimized even with changes in NTT length *N* and parallel factor *P*. It can be applied to a lattice-based post-quantum crypto-processor for a wider variety of devices with a limited use environment.

ACKNOWLEDGMENTS

This work was supported by research fund of Chungnam National University.

REFERENCES

[1] T. N. Tan and H. Lee, "High-Secure Low-Latency Ring-LWE Cryptography Scheme for Biomedical Images Storing and Transmitting," in Proc. 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1-4.

[2] D. -e. -S. Kundi, A. Khalid, S. Bian, C. Wang, M. O'Neill and W. Liu, "AxRLWE: A Multi-level Approximate Ring-LWE Co-processor for Lightweight IoT Applications," in IEEE Internet of Things Journal, vol. 9, no. 13, pp. 10492-10501.

[3] T. Shimada and M. Ikeda, "High-Throughput Polynomial Multiplier Architecture for Lattice-Based Cryptography," in Proc. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5.

[4] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124-134.

[5] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta et al., "Status report on the second round of the nist post-quantum cryptography standardization process," US Department of Commerce, National Institute of Standards and Technology, 2020.

[6] D. Micciancio, and O. Regev. "Lattice-based cryptography," in Post-quantum cryptography. Springer, Berlin, Heidelberg, 2009, pp. 147-191.

[7] R. Overbeck, and N. Sendrier. "Code-based cryptography," in Post-quantum cryptography. Springer, Berlin, Heidelberg, 2009, pp. 95-145.

[8] J. Ding, and B.Y. Yang. "Multivariate public key cryptography," in Post-quantum cryptography. Springer, Berlin, Heidelberg, 2009, pp. 193-241.

[9] A. Faz-Hernández, J. López, E. Ochoa-Jiménez and F. Rodríguez-Henríquez, "A Faster Software Implementation of the Supersingular Isogeny Diffie-Hellman Key Exchange Protocol," in IEEE Transactions on Computers, vol. 67, no. 11, pp. 1622-1636, 1 Nov. 2018.

[10] M. Mozaffari-Kermani and R. Azarderakhsh, "Reliable hash trees for post-quantum stateless cryptographic hash-based signatures," in Proc. 2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), 2015, pp. 103-108.

[11] Y. Zhu et al., "LWRpro: An Energy-Efficient Configurable Crypto-Processor for Module-LWR," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 3, pp. 1146-1159, March 2021.

[12] N. Zhang et al., "NTTU: An Area-Efficient Low-Power NTT-Uncoupled Architecture for NTT-Based Multiplication," in IEEE Transactions on Computers, vol. 69, no. 4, pp. 520-533, 1 April 2020.

[13] P. Martins, L. Sousa, and A. Mariano. "A Survey on Fully Homomorphic Encryption: An Engineering Perspective," ACM Comput. Surv., vol. 50, no. 6, pp. 1-33, Nov. 2018.

[14] B. Koziel, R. Azarderakhsh and M. M. Kermani, "A High-Performance and Scalable Hardware

Architecture for Isogeny-Based Cryptography," in IEEE Transactions on Computers, vol. 67, no. 11, pp. 1594-1609, 1 Nov. 2018.

[15] S. Heyse, and T. Güneysu. "Towards one cycle per bit asymmetric encryption: Code-based cryptography on reconfigurable hardware," in Proc. International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2012.

[16] T. Oder, T. Güneysu, F. Valencia, A. Khalid, M. O'Neill and F. Regazzoni, "Lattice-based cryptography: From reconfigurable hardware to ASIC," in Proc. 2016 International Symposium on Integrated Circuits (ISIC), 2016, pp. 1-4.

[17] J. Xie, K. Basu, K. Gaj and U. Guin, "Special Session: The Recent Advance in Hardware Implementation of Post-Quantum Cryptography," in Proc. 2020 IEEE 38th VLSI Test Symposium (VTS), 2020, pp. 1-10.

[18] P. Duong-Ngoc and H. Lee, "Configurable Mixed-Radix Number Theoretic Transform Architecture for Lattice-Based Cryptography," in IEEE Access, vol. 10, pp. 12732-12741, 2022.

[19] L. Ducas, and A. Durmus. "Ring-LWE in polynomial rings," in Proc. International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2012.

[20] K. Yao, D. -E. -S. Kundi, C. Wang, M. O'Neill and W. Liu, "Towards CRYSTALS-Kyber: A M-LWE Cryptoprocessor with Area-Time Trade-Off," in Proc. 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5.

[21] T. N. Tan, Y. Hyun, J. Kim, D. Choi and H. Lee, "Ring-LWE Based Face Encryption and Decryption System on a GPU," in Proc. 2019 International SoC Design Conference (ISOCC), 2019, pp. 15-16.

[22] C. Du and G. Bai, "Towards efficient polynomial multiplication for lattice-based cryptography," in Proc. 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 1178-1181.

[23] S. S. Roy, et al., "Compact ring-LWE cryptoprocessor," in Proc. International workshop on cryptographic hardware and embedded systems. Springer, Berlin, Heidelberg, 2014.

[24] M. Garrido, "A survey on pipelined FFT hardware architectures," Journal of Signal Processing Systems, 2021, pp. 1-20.

[25] D. Harvey, "Faster arithmetic for number-theoretic transforms," Journal of Symbolic Computation, vol. 60, pp. 113-119, 2014.

[26] C. P. Rentería-Mejía and J. Velasco-Medina, "Hardware design of an NTT-based polynomial multiplier," in Proc. 2014 IX Southern Conference on Programmable Logic (SPL), 2014, pp. 1-5.

[27] T. Nguyen Tan, T. Thi Bao Nguyen, and H. Lee, "High Efficiency Ring-LWE Cryptoprocessor Using Shared Arithmetic Components," MDPI Electronics vol. 9, no. 7, pp. 1-12, 2020.

[28] C. P. Rentería-Mejía and J. Velasco-Medina, "High-Throughput Ring-LWE Cryptoprocessors," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 8, pp. 2332-2345, Aug. 2017.

[29] S. Jumde, R. N. Mandavgane, and D. M. Khatri. "Review of parallel polynomial multiplier based on FFT using Indian Vedic mathematics," International Journal of Computer Applications, vol. 111, no. 17, pp. 10-13, 2015.

[30] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in Proc. Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2010.

[31] J. W. Bos, C. Costello, M. Naehrig and D. Stebila, "Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem," in Proc. 2015 IEEE Symposium on Security and Privacy, 2015, pp. 553-570.

[32] A. Langlois, and D. Stehlé, "Worst-case to average-case reductions for module lattices," Designs, Codes and Cryptography, vol. 75, no. 3, pp. 565-599, 2015.

[33] M. Abdalla, F. Benhamouda, and D. Pointcheval, "Public-key encryption indistinguishable under plaintext-checkable attacks," IET Information Security, vol. 10, no. 6, pp. 288-303, 2016.

[34] P. Longa, and M. Naehrig. "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," In Proc. International Conference on Cryptology and Network Security. Springer, Cham, 2016.

[35] M. Garrido, et al. "Hardware architectures for the

fast Fourier transform," Handbook of signal processing systems. Springer, Cham, 2019, pp. 613-647.

[36] F. Winkler, “Polynomial algorithms in computer algebra,” Springer Science & Business Media, 1996.

[37] D. D. Chen et al., "High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 62, no. 1, pp. 157-166, Jan. 2015.

[38] Z. Cao, and X. Wu, "An improvement of the Barrett modular reduction algorithm," International Journal of Computer Mathematics, vol. 91, no. 9, 2014, pp. 1874-1879.

[39] Z. Liu, et al., "Efficient Ring-LWE encryption on 8-bit AVR processors," In Proc. International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2015.

Soyeon Choi received B.S. degree in electronics engineering from Chungnam National University, Daejeon, South Korea, in 2018, where she is currently working toward the Unified Master and Ph.D. degree. Her main interests are VLSI for error correction codes, embedded system, and FPGA reverse engineering.

Yerin Shin received the B.S., and M.S. degrees in electronics engineering from Chungnam National University, Daejeon, South Korea, in 2016 and 2022, respectively. Her research interests include VLSI for error correction codes, and SoC for embedded system. Since 2022, she has been with the Department of TV/Commercial, LX Semicon, Daejeon, where she is currently a Researcher.

Kiho Lim received his M.S. and Ph.D. degrees in Computer Science from the University of Kentucky in 2012 and 2016, respectively. Currently, he is an Assistant Professor in the Department of Computer Science at William Paterson University of New Jersey, USA. Prior to joining WPU, he as an assistant professor with the Department of Computer Science at the University of South Dakota from 2016 to 2019. His research interests include cybersecurity, network security, vehicular networks, wireless communications, fog/edge computing, and IoT. He is currently the director of Cybersecurity Lab at WPU.

Hoyoung Yoo received the B.S. degree in electrical and electronics engineering from Yonsei University, Seoul, South Korea, in 2010, and the M.S., and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2012 and 2016, respectively. Since 2016, he has been with the Department of Electronics Engineering, Chungnam National University (CNU), Daejeon, where he is currently an Associate Professor. Prior to joining CNU, in 2016, he was with Samsung Electronics, Hwasung, South Korea, where he was involved in the research of nonbinary LDPC decoders for NAND flash memories. His current research interests include algorithms and architectures for errorcorrecting codes, FPGA reverse engineering, GNSS communication, and 5G communication systems.