IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 February 2024, accepted 24 February 2024, date of publication 29 February 2024, date of current version 11 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3371581

RESEARCH ARTICLE

High-Speed NTT Accelerator for CRYSTAL-Kyber and CRYSTAL-Dilithium

TRONG-HUNG NGUYEN, (Graduate Student Member, IEEE), BINH KIEU-DO-NGUYEN, (Graduate Student Member, IEEE), CONG-KHA PHAM, (Senior Member, IEEE), AND TRONG-THUC HOANG, (Member, IEEE)

Department of Computer and Network Engineering, The University of Electro-Communications (UEC), Tokyo 182-8585, Japan

Corresponding author: Trong-Hung Nguyen (tronghung@vlsilab.ee.uec.ac.jp)

This work was supported by The University of Electro-Communications (UEC) via the Support for External Funds Acquisition by Early Career Scientists Program.

ABSTRACT The efficiency of polynomial multiplication execution majorly impacts the performance of lattice-based post-quantum cryptosystems. In this research, we propose a high-speed hardware architecture to accelerate polynomial multiplication based on the Number Theoretic Transform (NTT) in CRYSTAL-Kyber and CRYSTAL-Dilithium. We design a Digital Signal Processing (DSP) architecture for modular multiplication in butterfly and Point-Wise Multiplication (PWM) operations. Our method reduces the critical path delay of an n-bit multiplier to that of a $(2n\text{-}2)$-bit adder, optimizing both area and speed. These dedicated DSPs are employed in butterfly and PWM operations, completely eliminating the pre-process and post-process of NTT transforms. Furthermore, we introduce a novel unified pipelined architecture for the NTT and Inverse NTT (INTT) transformations of Kyber and Dilithium, with corresponding high-speed (Radix-2) and ultra-high-speed (Radix-4) versions. Lastly, we construct a complete hardware accelerator for polynomial matrix-vector multiplication in Kyber. The Field-Programmable Gate Array (FPGA) implementation results have proven that our designs have significantly improved execution time by 3.4×–9.6× for the NTT transforms in Dilithium and 1.36×–34.16× for Kyber polynomial multiplication, compared to previous studies reported to date. Additionally, the hardware footprint results indicate that our proposed architectures exhibit superior hardware performance in Area-Time-Product (ATP), corresponding to a 44%–96% improvement. The proposed architectures are efficient and well-suited for high-performance lattice-based cryptography systems.

INDEX TERMS Post-quantum cryptography, number theoretic transform, crystal-kyber, crystal-dilithium, unified-pipelined NTT, digital signal processing.

I. INTRODUCTION

Shor's algorithm is a quantum algorithm developed in 1994 to find the prime factors of an integer [1]. Shor's algorithm theoretically enables quantum computers to break the most widespread public key encryption systems like Rivest-Shamlr-Adleman (RSA) and elliptic curve cryptography (ECC). In 2016, the National Institute of Standards and Technology (NIST) initiated a standardization process for Post-Quantum Cryptography (PQC) public key algorithms. The aim is to enhance information security against potential threats from large-scale quantum computers. After three rounds of a six-year standardization process, in July 2022, NIST announced the selection of the first group of algorithms for standardization [2]. This group includes the CRYSTAL-Kyber Key Encapsulation Mechanism (KEM) [3] and three digital signature schemes: CRYSTAL-Dilithium [4], FALCON [5], and SPHINCS+ [6].

CRYSTAL-Kyber and CRYSTAL-Dilithium are two lattice-based algorithms recommended by NIST for implementation in most applications due to their strong security and excellent performance. The lattice-based algorithms

The associate editor coordinating the review of this manuscript and approving it for publication was Tianhua Xu.

are built on the mathematical foundation of the Learning With Errors (LWE) problem. Kyber and Dilithium are based on the Module-LWE problem, a variation of LWE. This variation replaces the matrix coefficients in LWE with noisy ring lattices. Matrix and vector multiplication operations can be efficiently computed using the Number Theoretic Transform (NTT). NTT speeds up polynomial multiplication with a quasilinear complexity of O($n \ \ log\, n$) compared to the naive schoolbook polynomial multiplication with a time complexity of O(n^2).

NTT designs include hardware implementations, software implementations, and hardware-software co-designs [7], [8]. The selection of design solutions depends on the application objectives. Software-based NTT implementations are more deployed for applications on IoT and constrained devices [9], [10], [11], [12]. For high-performance requirements like servers, Graphics Processing Units (GPUs) are prioritized for use. Parallelization techniques have been applied to accelerate NTT on GPUs efficiently [13], [14], [15], [16]. However, current design objectives require a combination of resource cost and execution time. Thus, pure hardware implementations might offer better optimization for NTT performance. NTT-based polynomial multiplication involves converting polynomials to the NTT domain, performing point-wise multiplications, and converting the product polynomial back to the normal domain. The transformations of polynomials into and out of the NTT domain are the primary performance challenge in implementing lattice-based cryptosystems. Existing optimization design strategies for NTT accelerators can be categorized into two distinct approaches.

The first approach involves executing NTT computations iterative for a sequence of butterfly operations on the input polynomial coefficients. This process includes loading pairs of coefficients from memory, performing butterfly operations, and storing the resulting output pairs back into memory. As a result, for a n-degree polynomial, $n/2$ butterfly operations are needed for each NTT/INTT stage. Parallel architecture with multiple butterfly cores can be used to accelerate the execution speed of the NTT transformations. However, implementing parallel execution with multiple butterfly cores leads to an exponential increase in temporary memory requirements. This approach also poses challenges in managing memory access patterns.

The other approach is to use a pipelined architecture for NTT transformations. This architecture uses a chain of butterfly cores, each performing the butterfly operation for a separate NTT/INTT stage. Re-order units are used between butterfly cores to rearrange the order of coefficients after each stage. Therefore, to execute the NTT of a n-degree polynomial, a chain of $n\,log\,n$ butterfly cores and $n\,log\,n$–1 re-order units are required. The pipelined architecture has the advantage of high speed, without temporary memory usage, and straightforward memory access patterns. However, the architecture's notable drawback is the large NTT size, particularly when using unified butterfly cores.

A. RELATED WORKS

Choosing an appropriate butterfly configuration is crucial for efficiently performing NTT and INTT transformations. In study [17], Banerjee et al. proposed a customizable unified butterfly unit architecture that supports the Cooley-Tukey (CT) and Gentleman-Sande (GS) algorithms, corresponding to NTT and INTT. This unified butterfly unit consists of two sets of a modular adder and a modular subtracter, along with a shared modular multiplier. As a result, the pre-process and post-process are merged into NTT and INTT calculations, eliminating the costly bit-reversal process. In study [18], Xing and Li proposed a compact hardware implementation that utilized a set of two unified butterfly cores. This design performs separate butterfly operations for even and odd coefficients of polynomials in Kyber. Their butterfly configuration also supported PWM mode with four modular multiplications for one PWM operation. Multi-butterfly cores can be used in parallel to accelerate iterative NTT architecture. In [19], Niasar et al. proposed a modular reduction algorithm, called K^2-RED, based on the K-RED algorithm [20], which helps reduce the hardware resources needed for modular multiplication. Therefore, the NTT architecture presented in [19] utilizes a 2×2 butterfly configuration and achieves a compact size with high speed. In [21], Dang et al. introduced an accelerator using k pairs of 2×1 butterfly cores, with k equal to 2, 3, and 4, corresponding to security levels 1, 3, and 5 of Kyber, respectively. Furthermore, in [22], Yaman et al. proposed three NTT hardware architectures for lightweight, balanced, and high-performance systems, corresponding to the number of parallel butterfly cores being 1, 4, and 16, respectively.

The Dilithium digital signature algorithm has a large prime parameter with $q = 8380417$, compared to the Kyber case with $q = 3329$. As a result, the Dilithium butterfly unit hardware becomes more costly, especially when using a unified configuration. Existing accelerators for Dilithium all employ an iterative NTT architecture, which helps save hardware resources. In studies [23], [24], a pair of butterfly cores are utilized for lightweight Dilithium implementations. [25] proposed a butterfly core architecture using DSPs for all arithmetic operations within the butterfly computation. The NTT accelerator architecture in [25] achieved low hardware resource usage and high clock frequency.

In [26], Pham et al. proposed a configurable computational unit for operations in polynomial multiplication based on NTT. In [27], Beckwith et al. designed a 2×2 butterfly core configuration for a high-speed Dilithium system. Furthermore, in study [28], a high-performance NTT architecture based on a pipelined design is proposed by Zhao et al. This design utilizes a segmented pipeline architecture to reduce hardware resource usage for the butterfly cores. Storage operations and coefficient reordering are handled by the re-order units. This results in simplified memory access patterns and reduced temporary memory requirements.

The most significant challenge in designing iterative NTT with multiple parallel butterfly cores is to propose an efficient memory access pattern. Studies [24], [29], [30], [31] have utilized the ping-pong memory access method. This approach easily avoids memory access and read-after-write conflicts. However, the memory usage in the NTT design is doubled. Studies [32], [33], [34], [35] have focused on proposing conflict-free memory access schemes on a single memory. Additionally, in [36], Ni et al. replaced the BRAM units with First-In-First-Outs (FIFOs) to rearrange the coefficient order after each NTT/INTT stage.

In contrast to the iterative architecture, pipelined NTT architecture has received less attention due to the trade-off between speed and area. In [37], Ni et al. introduced a fully pipelined Radix2-Multipath Delay Commutator NTT for Kyber. This design proposed a unified butterfly configuration for butterfly cores, requiring a significant hardware resource for the butterfly core chain. The difference in input order of NTT and INTT transformations also requires doubling the number of delay units. Additionally, Kyber's complex polynomial multiplication requires a separate PWM design. In [38], Ngoc and Lee proposed a configurable mixed-NTT structure. This design presented a minimalist uniform butterfly core architecture consisting of one modular addition, subtraction, and multiplication. However, this design has a low utilization rate when performing INTT transformations and requires considerable DSP usage. In [39], Ye et al. proposed a pipeline processing architecture with only one input coefficient on each clock cycle. Therefore, this design helps reduce hardware resources and memory consumption. However, the number of clock cycles required for NTT calculations increases significantly.

Based on our analysis of the current designs, we found that the high-speed pipelined NTT architecture is more suitable for large-scale PQC lattice-based cryptosystems. However, there needs to be more research on optimized implementations for this architecture, particularly for Dilithium. Therefore, it is essential to conduct further research on the pipeline-oriented approach.

B. OUR CONTRIBUTIONS

In this study, a fully unified pipelined NTT accelerator architecture is proposed for Kyber and Dilithium. We conduct comprehensive hardware optimization from modular multiplication and PWM operations to the pipelined NTT architecture for polynomial multiplication. As a result, the proposed designs achieve excellent hardware performance and can be deployed on FPGA and Application-Specific Integrated Circuit (ASIC) platforms with various customizable options. Our high-speed NTT accelerator is suitable for high-performance applications that require low latency and prioritize quantum-resistant security, such as cloud computing and servers. To the best of our knowledge, this is the first pipelined NTT design for Dilithium and the first hardware implementation of a fully unified NTT accelerator architecture for Kyber. The key contributions of this research can be summarized as follows:

1) We propose a novel DSP design for modular multiplying large coefficients and applying it in the case of Dilithium. The proposed DSP can perform a 23×23-bit multiplication and modular reduction $q = 8380417$. To achieve this, we first break down the 23×23-bit multiplication into 12×12-bit multiplications. We then optimize the Vedic multiplier to reduce the critical path delay from an n-bit multiplier to a $(2n-2)$-bit adder. Additionally, we directly apply the property of the prime number q, $2^{23} \equiv 2^{13} - 1 (mod\ q)$ to the 23×23-bit multiplication process. Finally, we can use the K-RED algorithm to perform modulus q on the product of multiplying two 23-bit coefficients. Furthermore, our DSP can be optimized to merge the multiplication of factor n^{-1} into the PWM calculations. As a result, the unified butterfly core architecture, which completely eliminates the pre-process and post-process, is designed.
2) We also propose a novel unified pipelined NTT architecture for Kyber and Dilithium. A chain of $\log n$ unified butterfly cores using our dedicated DSP is designed to minimize hardware footprint. Furthermore, our NTT architecture is optimized to enable reverse data flow control when performing NTT and INTT transformations. As a consequence, all re-order units achieve a 100% utilization rate. The number of re-order units is halved compared to other pipelined architectures. We implement two NTT versions, the high-speed (Radix-2) and ultra high-speed (Radix-4), for both Kyber and Dilithium cases. The number of clock cycles required for executing NTT of an n-degree polynomial corresponds to two versions, $n/2$ and $n/4$ clock cycles, respectively.
3) We utilize two proposed NTT architectures to create two complete accelerator versions for the polynomial matrix-vector multiplication in Kyber. The PWM architecture in our accelerators is also optimized for the data flow. Consequently, the INTT process can be performed closely in parallel with the PWM calculation, further enhancing the speed of the proposed accelerators.
4) Finally, we implement the proposed architectures on the baseline Xilinx Artix-7 FPGA platform and compare them to state-of-the-art works. Our results indicate that with bottom-up hardware design flow, the proposed architectures can achieve a speed-up ratio range of 3.4–9.6× for Dilithium NTT transformations and 1.36–34.16× for Kyber polynomial multiplication. The results on hardware utilization also show that our hardware efficiency is significantly improved in terms of area-time-product, corresponding to 52–85% for the Dilithium NTT design and 44–96% for the Kyber accelerator design.

TABLE 1. Specification of PQC CRYSTAL–Kyber and CRYSTAL–Dilithium.

Specifications	CRYSTAL-Kyber (Public-Key Encryption/KEMs)	CRYSTAL–Dilithium (Digital Signature Scheme)
Polynomial ring R_q	$\mathbb{Z}_q[x]/x^n+1$	$\mathbb{Z}_q[x]/x^n+1$
n	256	256
q	3329	8380417
Integer multiplication size	12×12	23×23
Module dimensions $(k \times l)$	$(2\times 2),(3\times 3),(4\times 4)$	$(4\times 4),(6\times 5),(8\times 7)$
Number of polynomial multiplication in Encapsulation/Sign	$(k^2+k)=(10,12,20)$	$(k\times l+2k+l)=(28,47,79)$
Number of polynomial multiplication in Decapsulation/Verify	$(k^2+2k)=(12,15,24)$	$(k\times l+k)=(20,36,64)$

The rest of this paper is organized as follows: Section II provides some background information about PQC algorithms CRYSTAL-Kyber, Dilithium, and NTT-based polynomial multiplication. Section III describes the major contributions of this paper. It includes the design methodology and implementation details. Section IV presents our implemented results and analysis on an FPGA device. Section V concludes the paper and suggests some future directions for our research.

II. BACKGROUND KNOWLEDGE

A. CRYSTAL-KYBER AND DILITHIUM

Kyber and Dilithium are two cryptographic primitives in the Cryptographic Suite for Algebraic Lattices (CRYSTAL) selected for standardization after the third round of the PQC standardization process by NIST. Kyber is a Key Encapsulation Mechanism (KEM) based on the hardness of the Module-Learning With Error (M-LWE) problem. Dilithium is a digital signature scheme based on the hardness of the M-LWE and Shortest Integer Solution (SIS) problems. Table 1 shows the specifications of Kyber and Dilithium in the third round. Both algorithms' polynomials and algebraic operations are on the polynomial ring $R_q = \mathbb{Z}_q[x]/x^n+1$. The polynomials have a degree of n, with $n = 256$ for Kyber and Dilithium. The polynomials in the matrix have coefficients in the range $[0, q)$, corresponding to $q = 3329 = 2^{12}-3\times 2^8+1$ for Kyber and $q = 8380417 = 2^{23}-2^{13}+1$ for Dilithium.

The basic operations of Kyber and Dilithium involve modular arithmetic on the prime number q, with the output being in the range $[0, q)$. Among all the operations, modular multiplication is considered the most complex, as it involves reducing the double bit-size result of integer multiplication. For Kyber, the integer multiplication size is 12 × 12-bits, while Dilithium is 23 × 23-bits.

A critical mathematical property of M-LWE is that the security level can be adjusted by changing the dimensions of the polynomial matrices $(k \times l)$ and polynomial vectors (k/l). When the security level changes, the size n of the cyclic polynomials remains constant. Therefore, the implementation resources are almost independent of the security level.

Specifically, Kyber uses polynomial matric with dimensions $k \times l = 2 \times 2$ for Kyber512 (NIST-security level 1, equivalent to AES-128), 3 × 3 for Kyber768 (NIST-security level 3, equivalent to AES-192), and 4 × 4 for Kyber1024 (NIST-security level 5, equivalent to AES-256). The number of polynomial multiplications for Kyber is calculated as k^2+k for the Encapsulation stage and k^2+2k for the Decapsulation stage. Dilithium, on the other hand, uses a polynomial matrix size of $k \times l = 4 \times 4$ for NIST-security level 2, 6 × 5 for NIST-security level 3, and 8 × 7 for NIST-security level 5. The number of polynomial multiplications for Dilithium is relatively large, with $k \times l + 2k + l$ for the Sign stage and $k \times l + k$ for the Verify stage. To implement security level 5, Dilithium requires executing up to 79 polynomial multiplications for Sign and 64 polynomial multiplications for Verify.

In summary, polynomial multiplication is crucial for both Kyber and Dilithium algorithms and needs to be optimized to improve implementation efficiency.

B. NTT-BASED POLYNOMIAL MULTIPLICATION

The Number Theoretic Transform (NTT) is an effective method for computing polynomial multiplication. It is a generalization of the fast Fourier transform in the finite field. NTT-based polynomial multiplication has a time complexity of $O(n\ log\ n)$, much faster than the schoolbook multiplication method with a complexity of $O(n^2)$. To utilize NTT, a primitive n-th root of unity ω_n exist in the ring $\mathbb{Z}_q$, which means that $q \equiv 1\ (mod\ n)$. Therefore, NTT-based polynomial multiplication is expressed as follows:

$$c = INTT(NTT(a) \circ NTT(b)) \tag{1}$$

where $a, b, c \in R_q = \mathbb{Z}_q[x]/x^n+1$, and $\circ$ denote point-wise multiplication in the NTT domain.

In [40], Lyubashevsky et al. proposed the Negative-Wrapped Convolution (NWC) to avoid zero-padding when performing NTT. The NWC method requires $q \equiv 1(mod\ 2n)$ so that a primitive $2n$-th root of unity exists, i.e., $\psi_{2n}^2 = \omega_n$. Thus, the NTT and INTT operations are expressed as follows:

$$NTT^{\psi}(a_j) = \sum_{i=0}^{n-1} a_i \psi_{2n}^i \omega_n^{ij}\ mod\ q,\ j = 0, \ldots, n-1. \tag{2}$$

$$INTT^{\psi^{-1}}(\hat{a}_i) = n^{-1}\psi_{2n}^{-i}\sum_{j=0}^{n-1} \hat{a}_j \omega_n^{-ij}\ mod\ q, i = 0, \ \ldots, n-1. \tag{3}$$

There are two widely used algorithms for fast Radix-2 NTT computation, namely Cooley-Tukey (CT) algorithm [41] and Gentleman-Sande (GS) algorithm [42]. Furthermore, in studies [43], [44], the authors proposed merging algorithms to eliminate the pre-process with CT configuration and the post-process with GS configuration, respectively. Finally, polynomial multiplication based on NWC-NTT can be performed as follows:

$$c = INTT_{GS}^{\psi^{-1}}(NTT_{CT}^{\psi}(a) \circ NTT_{CT}^{\psi}(b)) \quad (4)$$

III. PROPOSED ARCHITECTURES

A. MODULAR MULTIPLICATION

Modular multiplication is the critical path delay in NTT architecture. Previous studies have predominantly used DSP for integer coefficient multiplication and proposed optimal designs for modular reduction. In Xilinx FPGAs, the 7-series FPGAs containing the DSP48E1 block that supports 25×18 multiplication [45], and Ultrascale/Ultrascale+ FPGAs, which hold the DSP48E2 block supporting 27×12 multiplication [46], are commonly used. However, the 12×12 multiplication in Kyber and the 23×23 multiplication in Dilithium only partially utilize the hardware resources of the DSPs. Several efficient designs for Kyber modulus $q = 3329$ reduction have been listed in [47]. In the case of Dilithium with a large q, recent studies often utilize the property $2^{23} \equiv 2^{13} - 1(mod\ q)$ to design intricate combinational logic circuits for modulus $q = 8380417$ [23], [24], [25], [48], [49]. In [26], Pham et al. utilized a variation of the Barrett method combined with the $2^{23} \equiv 2^{13} - 1(mod\ q)$ property to execute the modular reduction for Dilithium efficiently. In [50], the K-RED and K-RED2x methods were employed for modulus q in the Dilithium NTT implementation. In [47], we proposed a novel DSP design method for modular multiplication to achieve even more area and speed-efficient designs. However, the method in [47] is suitable for small-integer multiplications. In this section, we present an optimized DSP design for large-integer multiplication in Dilithium.

The proposed multiplier design method is depicted in Figure 1, illustrating a simple example of multiplying two 4-bit numbers. Figure 1a represents the Vedic multiplier introduced in [47]. The vertical and crosswise technique is used to compute the partial products, and the product is carried out through a few layers of addition. However, the number of addition layers and partial products increases when applied to a larger multiplier. This results in an increase in the critical path delay and inefficiency compared to conventional multiplication methods.

To achieve a more optimized design regarding area and speed, we propose a design method for a large multiplier, as shown in Figure 1b. The partial products $(P0, P1, P2, P3, P4, P5, P6)$ are represented as a triplet of numbers (A, B, C). Consequently, computing the sum of these partial products through layers of addition is replaced by the conventional addition of three numbers: A, B, and C.

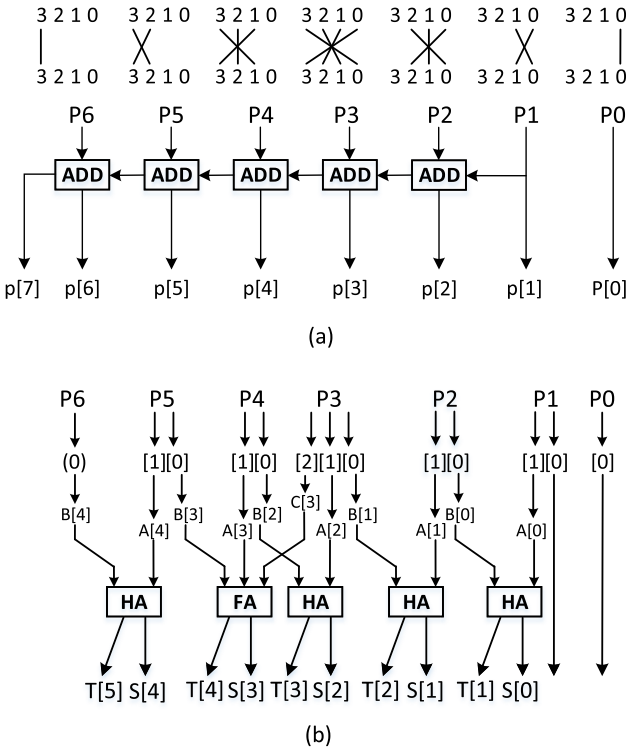

FIGURE 1. Architectures for binary multipliers: (a) A Vedic multiplier using carry-propagate adder for small integer in [47] and (b) The proposed effective multiplier using carry-save-adder for large integer.

Next, we utilize the Carry Save Adder (CSA) for adding k-bit integers A, B, and C to produce the integer pair (T, S). The CSA circuit consists of a parallel set of k full adders without horizontal connection. Thus, the addition of three integers to compute two integers requires a single full adder delay. Moreover, as the size of the multiplication increases, the number of integers representing the partial products also increases. We can easily expand the CSA addition by adding parallel layers of CSAs. The product p is finally computed from the $T + S$ addition result and concatenated with $P0[0]$ and $P1[0]$. As a result, an n-bit multiplier has a critical path delay equivalent to that of a $(2n\text{-}2)$-bit adder. The proposed DSP architecture performing modular multiplication for Dilithium is shown in Figure 2. The multiplication of two 23-bit coefficients, a and b, is separated into four parallel multiplications: $(a_L \times b_H), (a_H \times b_L), (a_L \times b_L)$, and $(a_H \times b_H)$, where $a = (a_L + 2^{12} \times a_H)$ and $b = (b_L + 2^{12} \times b_H)$. The partial multiplications are performed according to the proposed multiplier above. The result of the multiplication $(a_H \times b_H)$ is $(2^{23} \times a_H \times b_H)$, correspondingly. Therefore, we apply the property $2^{23} \equiv 2^{13} - 1(mod\ q)$ to reduce the higher-order bits of the product. Specifically, we obtain $(2^{23} \times a_H \times b_H) = 2a_H \times b_H \times (2^{13} - 1)$. Consequently, the final multiplication result is computed as the sum $S = P_H + P_M + P_L$, and S is 38-bits width. The parameter q of Dilithium can be formed as $q = k \times 2^m + 1$, and we apply the K-RED algorithm with $k = 2^{10} - 1$ and $m = 2^{13}$. To directly apply the K-RED reduction, the sum S is reduced

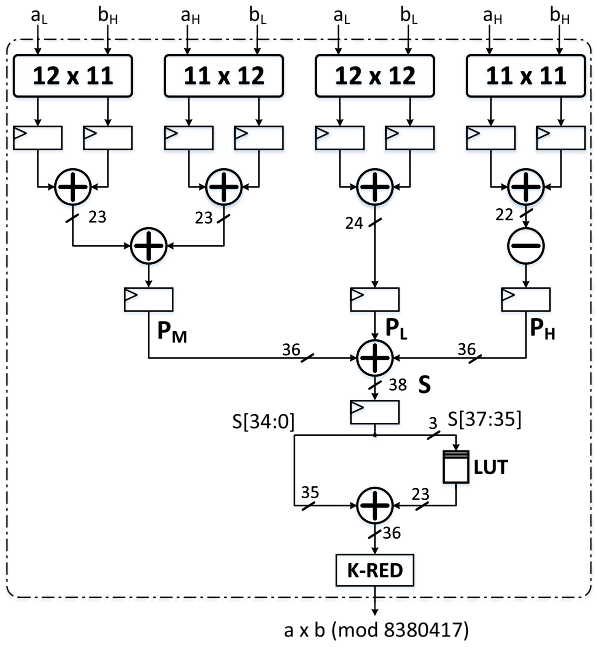

FIGURE 2. The proposed modular multiplication for Dilithium with the prime number $q = 8380417$.

to a width of 36 bits by using a Look-Up Table (LUT) for the operation $S[37:35] \times 2^{35} \pmod q$. The function K-RED takes any reduced integer S and returns an integer d, such that $d \equiv k \times S \bmod q$. The scale factor k can be eliminated by pre-multiplying twiddle factors with factor k^{-1}. Thus, K-RED(S) $= k \times d_0 - d_1$, with $d_0 = S \bmod 2^m$ and $d_1 = S/2^m$. For Dilithium, K-RED(S) $= (2^{10} - 1) \times S[12:0] - S[35:13] = (s[12:0] \ll 10) - (S[12:0] + S[35:13])$. Finally, at most, one addition is required to obtain the result of $a \times b \pmod{8380417}$. Our proposed DSPs take five Clock Cycles (CCs) for the modular multiplication of Kyber and Dilithium.

B. UNIFIED PIPELINED NTT ARCHITECTURE

The Multipath Delay Commutator (MDC) is a widely used pipelined architecture for performing Fast Fourier transform (FFT). It has been applied in several accelerated hardware designs for lattice-based post-quantum cryptography [37], [51], [52]. Compared to the iterative NTT architectures, MDC does not require temporary BRAM memory. It features a simple control pattern and proves highly effective in simultaneously processing multiple NTT/INTT. In this section, we propose two optimized MDC designs in terms of area and speed for executing NTT/INTT in the polynomial multiplications of Kyber and Dilithium. Our designs are based on Radix-2 MDC and Radix-4 MDC architectures, as illustrated in Figure 3 and Figure 4.

1) UNIFIED NTT CHAIN CONFIGURATION

The pipelined architecture uses one butterfly core for each NTT stage. To compute n-point NTTs in Radix-2 MDC architecture, a chain of $\log(n)$ butterfly cores is required, corresponding to seven butterfly cores for Kyber and eight butterfly cores for Dilithium. In the case of the Radix-4 MDC architecture, the number of butterfly units in the chain is doubled, resulting in 14 cores for Kyber and 16 cores for Dilithium. We use the unified butterfly core configuration from the study [47] for our chain of butterfly cores. This configuration supports CT operation for the NTT and GS operation for the INTT.

Moreover, the multiplication by the factor n^{-1} in the post-process can be eliminated by applying the property $k \times 2^m \equiv -1 \pmod q$ of the prime number q. Consequently, the unified NTT chain design achieves minimal area by completely removing both pre-process and post-process. The NTT and INTT computation also requires n powers of twiddle factor ω, equating to $n/2$ powers of ω^i and $n/2$ for ω^{-i}. However, the twiddle factor in NTT and INTT exhibits a symmetrical property. Thus, ω^{-i} can be calculated from ω^i and vice versa. Specifically, the twiddle factor conversion can be conducted as $\omega^{-i} = q - \omega^{n-i}$ in the same NTT layer. Hence, storing $n/2$ twiddle factors ω^i for NTT and INTT transformations saves half of the required Read-Only Memory (ROM) memory. Each of our butterfly units requires 7 CCs to perform CT or GS operations, with 5 CCs for modular multiplication and 2 CCs for modular addition/subtraction at the input/output of butterfly calculation.

2) UNIFIED DATAFLOW

The design for performing NTT-based polynomial multiplication must consider the order of input and output coefficients in the NTT accelerator. There are two NTT and INTT transformation configurations: Normal (No) and Bit-reversed (Br). The $No \rightarrow Br$ configuration requires input coefficients to be in normal order while producing output in bit-reversed order. Conversely, the $Br \rightarrow No$ configuration demands a bit-reversed order for inputs and a normal order for outputs. There are several variations of the CT and GS algorithms: $CT_{No \rightarrow Br}$, $CT_{Br \rightarrow No}$, $GS_{No \rightarrow Br}$, and $GS_{Br \rightarrow No}$. Reordering before or after NTT computation can demand significant computational resources. To entirely avoid the costly bit-reversal step, we utilize a combination of the $NTT^{CT}_{No \rightarrow Br}$ and $INTT^{GS}_{Br \rightarrow No}$ configurations.

However, this combination reverses the order in which the twiddle factors are loaded during NTT and INTT computations, adding complexity to the twiddle factor conversion technique in the previous subsection. Additionally, the coefficients are reordered after each butterfly computation through a Reordering Unit (RU). The RU design comprises two registers (D) and two controlled Multiplexer (MUX) units. The depth of register D corresponds to the difference order between input coefficients at each stage. In Kyber's NTT case, the D-depths are $(1, 2, 4, 8, 16, 32)$, leading to RU1, RU2, RU4, RU8, RU16, and RU32 unit sizes. For Dilithium, the RU unit sizes during NTT are RU1, RU2, RU4, RU8, RU16, RU32 and RU64. With the $NTT^{CT}_{No \rightarrow Br} \rightarrow INTT^{GS}_{Br \rightarrow No}$

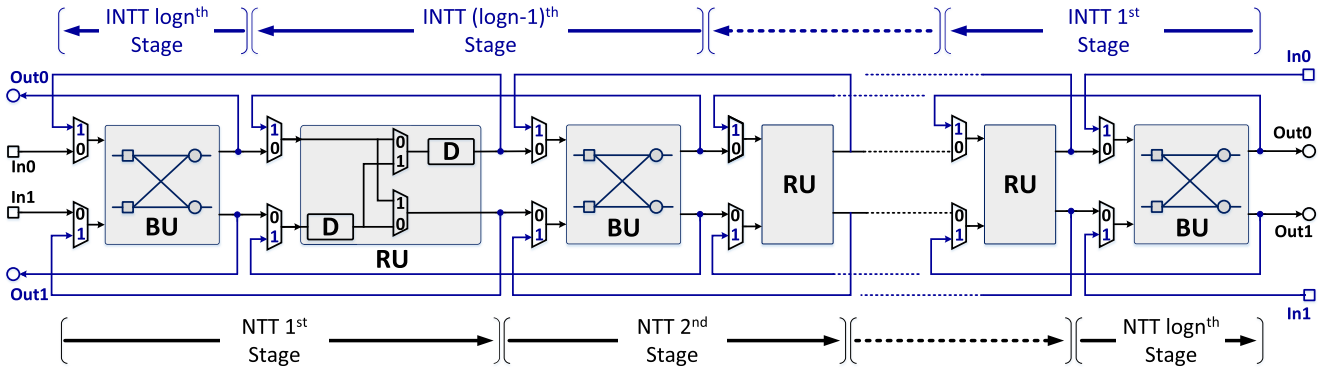

FIGURE 3. The proposed unified pipelined Radix-2 NTT MDC architecture.

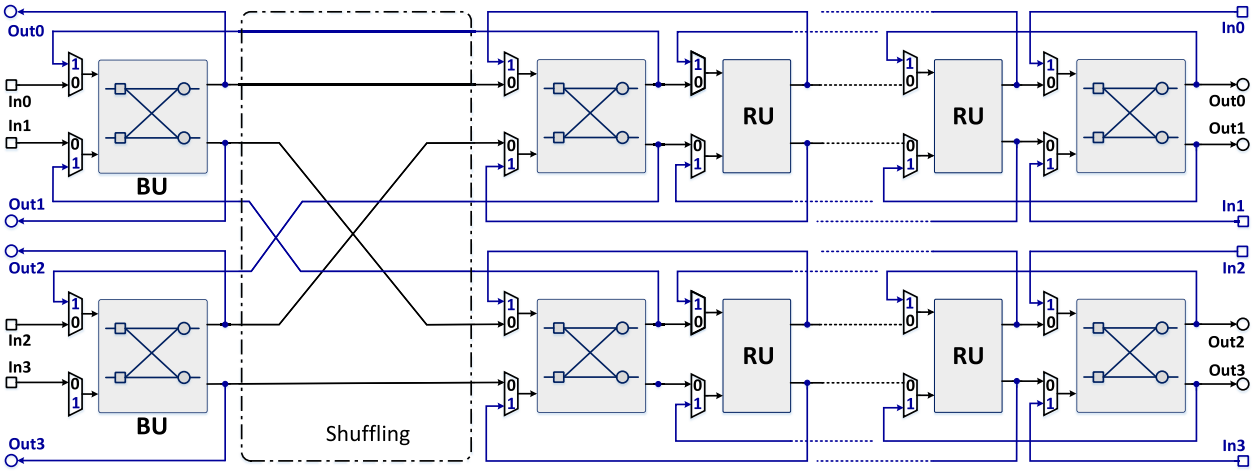

FIGURE 4. The proposed unified pipelined Radix-4 MDC NTT architecture.

configuration, the unit sizes for RU are reversed during NTT and INTT execution. Therefore, the number of RU units required doubles, achieving only a 50% utilization rate.

To eliminate the constraints of the $NTT^{CT}_{No\rightarrow Br}$ – $INTT^{GS}_{Br\rightarrow No}$ configuration, we reverse the data flow for NTT and INTT execution. Specifically, we utilize controlled MUXs for handling input data flow at the butterfly cores and the RUs, as illustrated in Figures 3 and 4. During NTT execution, the MUXs are set to mode 0, and the data flows from left to right, the black-lined path, respectively. Conversely, during INTT execution, the MUXs are set to mode 1, and the data flows from right to left, the blue-lined paths. With our unified NTT/INTT data flow, the $INTT^{GS}_{Br\rightarrow No}$ computation can be executed on the $NTT^{CT}_{No\rightarrow Br}$ architecture. Consequently, the RUs for the INTT architecture are eliminated, increasing the RUs utilization rate to 100%.

3) UNIFIED RADIX-2/4 MDC NTT ARCHITECTURE

The unified Radix-2 MDC architecture takes two coefficients per cycle as input and produces two outputs every cycle, as depicted in Figure 3. This architecture uses a single butterfly core to calculate and an RU unit to reorder data for the subsequent NTT stage. The Radix-2 MDC architecture can continuously perform NTT/INTT for multiple polynomials with an execution time of $\frac{n}{2} \times k + F$ clock cycles, where k represents the number of processed polynomials. F denotes the clock cycles needed to fulfill the pipeline. In the proposed Radix-2 MDC architecture, F is the sum of the delay across the butterfly chain, the pipeline registers for controlling NTT/INTT data flows, and the RU units. For Kyber, $F = (7 \times 7) + 7 + (32 + 16 + 8 + 4 + 2 + 1) = 119$ clock cycles. Similarly, for Dilithium, F is calculated as $(7\times 8)+8+(64+32+16+8+4+2+1) = 191$ clock cycles. In our architecture, F is constant when performing NTT and INTT transformations.

In the unified Radix-4 MDC architecture, four input coefficients are taken every cycle, producing four outputs per cycle, as illustrated in Figure 4. This design uses twice the number of butterfly cores compared to the unified Radix-2 MDC structure. Additionally, the Radix-4 MDC architecture replaces the RU unit with a shuffling circuit between the two initial stages of the NTT. Within our unified design, the shuffling circuit can also be deployed for the two final stages of INTT transformation. Thus, the data streams are

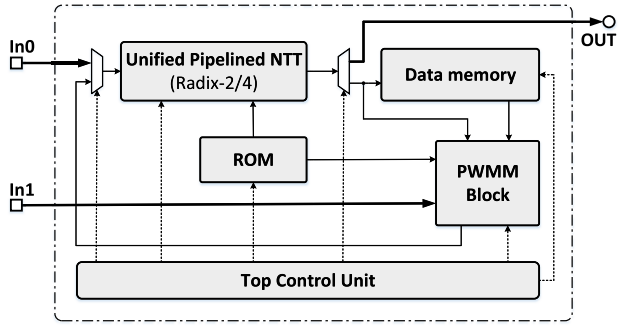

FIGURE 5. The proposed polynomial matrix-vector multiplication accelerator for CRYSTAL-Kyber.

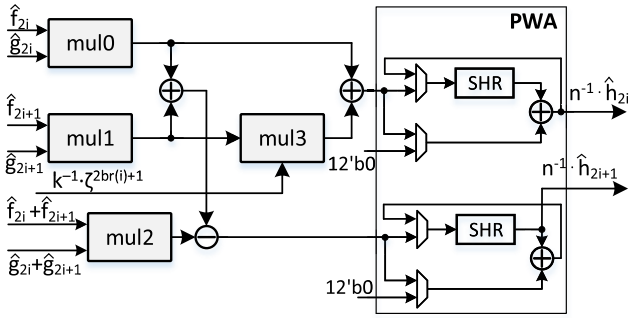

FIGURE 6. The architecture for a single PWMM operation.

performed directly without the need for reordering. As a result, the most delayed RU units are eliminated for both NTT and INTT, corresponding to RU32 for Kyber and RU64 for Dilithium. The unified Radix-4 MDC architecture can perform NTT/INTT for k polynomials with an execution time of $\frac{n}{4} \times k + F$ clock cycles. Consequently, the execution time for Kyber is 87 CCs, and for Dilithium is 127 CCs.

C. FULLY POLYNOMIAL MATRIX-VECTOR MULTIPLICATION ACCELERATOR

This section introduces a comprehensive accelerator for polynomial matrix-vector multiplication (PMVM) in CRYSTAL-Kyber. Our design employs two types of accelerators: a high-speed Radix-2 MDC and an ultra high-speed Radix-4 MDC, based on our proposed NTT architectures. The overall architecture of the accelerators is shown in Figure 5. The design includes a unified pipelined NTT unit that performs NTT/INTT transformations. After being transformed into the NTT domain, polynomials are stored in the Data Memory. The PWMM Block is designed to perform Point-Wise Multiplication (PWM) and Point-Wise Addition (PWA) operations. A ROM memory is used to store $n/2$ powers of the twiddle factor for NTT/INTT and $n/2$ factor ζ for PWM calculations. The Top Control Unit manages the enabling and disabling of functionalities of the units, switches the NTT/INTT mode, and generates data index for ROM output. Our accelerator takes polynomials to be NTT-executed at the input $In0$ and the polynomials for PWMM operation at the input $In1$. The result of the polynomial matrix-vector multiplication is obtained after performing INTT at the output, OUT.

1) THE PWMM ARCHITECTURE

The PWM operation in Kyber is carried out by multiplying two vectors of 128 degree-1 polynomials. In [18], Xing and Li proposed using the Karatsuba algorithm to optimize the PWM computation with four multiplications, such that $\hat{h} = \hat{h}_{2i} + \hat{h}_{2i+1}X = \hat{f} \circ \hat{g}$, where $\hat{h}_{2i} = \hat{f}_{2i}\hat{g}_{2i} + \hat{f}_{2i+1}\hat{g}_{2i+1}\zeta^{2br(i)+1}$ and $\hat{h}_{2i+1} = (\hat{f}_{2i} + \hat{f}_{2i+1})(\hat{g}_{2i} + \hat{g}_{2i+1}) - \hat{f}_{2i}\hat{g}_{2i} - \hat{f}_{2i+1}\hat{g}_{2i+1}$. We utilize the proposed architecture in [47] to calculate PWMM for a pair of even-odd coefficients, as illustrated in Figure 6. Four multiplications are executed using the proposed DSP, corresponding to $mul0$, $mul1$, $mul2$, $mul3$. In the PWA design, one T-depth shift register unit (SHR) with feedback and one modular addition is designed. The depth T precisely equals the number of clock cycles required for NTT/INTT execution, corresponding to 64 CCs for Radix-2 and 32 CCs for Radix-4. We use two PWMM units for the Radix-2 accelerator and four PWMM units for the Radix-4 accelerator. Therefore, the number of clock cycles needed to execute $\hat{h} = \hat{f} \times \hat{g}$ is 64 CCs and 32 CCs, matching the value of T. To optimize the data flow, we immediately extract the result of the PWMM computation for the even coefficients upon completing the PWA computations. Conversely, we extract the result of the odd coefficients after one feedback loop. This way, the even- and odd-polynomials undergo continuous INTT transformations without requiring temporary storage memory. Our PWMM design requires 15 CCs for pipeline registers.

2) POLYNOMIAL MATRIX-VECTOR MULTIPLICATION FLOW

In the proposed accelerator, the modules are designed with a tight constraint on the execution clock cycles, determined as T. Figure 7 illustrates an example of the data flow when performing the polynomial matrix-vector multiplication $t = \mathbf{A} \times s$ for Kyber768. The primary data flow involves NTT/INTT transformations, Random Access Memory (RAM) load/store operations, and PWMM calculations.

In the initial stage, the polynomial vector (s_0, s_1, s_2) undergoes the NTT module following the even–odd coefficient polynomial $(s_{0e} \rightarrow s_{0o} \rightarrow s_{1e} \rightarrow s_{1o} \rightarrow s_{2e} \rightarrow s_{2o})$. After F CCs, the NTT pipeline is fulfilled, and the polynomials $(\hat{s}_0, \hat{s}_1, \hat{s}_2)$ are sequentially stored in RAM after every T CCs. When $\hat{s}_{2e}$ appears at the NTT's output, the polynomial $\hat{s}_0(\hat{s}_{0e}, \hat{s}_{0o})$ is loaded, and the polynomial $\hat{\mathbf{A}}_{00}$ is fetched from the input $In1$. In the PWMM module, the multiplication $\hat{\mathbf{A}}_{00} \circ \hat{s}_0$ is performed in parallel with $NTT(s_{2e})$, $Wr(\hat{s}_{2e})$, and $Rd(\hat{s}_0)$. Next, the polynomials $\hat{s}_1$ and $\hat{\mathbf{A}}_{01}$ are loaded to compute $\hat{\mathbf{A}}_{00} \circ \hat{s}_0 + \hat{\mathbf{A}}_{01} \circ \hat{s}_0$. At $NTT(s_{2o})$ time, the PWMM module performs $\hat{t}_0 = \hat{\mathbf{A}}_{00} \circ \hat{s}_0 + \hat{\mathbf{A}}_{01} \circ \hat{s}_1 + \hat{\mathbf{A}}_{02} \circ \hat{s}_2$. Thus, when the $NTT(s_{2o})$ is completed, the result $\hat{t}_{0e}$ is ready to perform

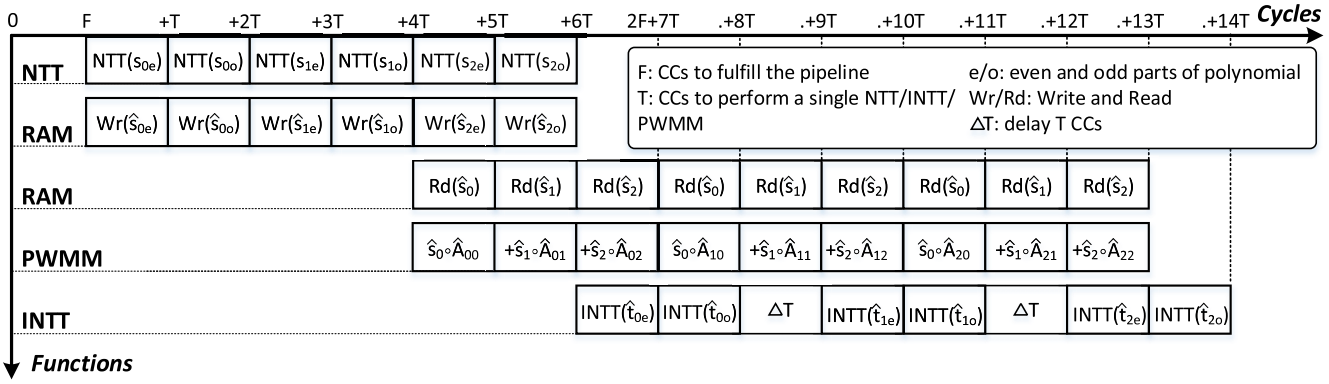

FIGURE 7. The data flow for performing matrix-vector multiplication in CRYSTAL-Kyber security level 3 using the proposed accelerator.

TABLE 2. Implementation results for the proposed unified pipelined NTT in Dilithium case compared with other studies.

Work	Device	Freq (MHz)	CCs	BFUs	LUTs	FFs	SLICEs	DSPs	BRAMs	Speed-up Ratio	Area-Time-Product (ATP)
This Work	**Artix-7**	**180**$^{(R2)}$	**128**	**8**	**7451**	**5275**	**2456**	**0**	**0**	**1.8**	**1753**
		163$^{(R4)}$	**64**	**16**	**13804**	**11019**	**4522**	**0**	**0**	**1.0**	**1764**
Gupta [24]	Artix-7	163$^{(I)}$	614	2	2759	2037	944	4	7	9.6	10336 (83%)
Zhao [28]	Artix-7	96.9$^{(I)}$	296	4	1919	1301	642	8	2	7.8	5626 (69%)
Land [25]	Artix-7	311$^{(I)}$	536	2	524	759	256	17	1	4.4	3716 (53%)
Beckwith [27]	Artix-7	116$^{(I)}$	300	2×2	9018	6292	3041	16	0	6.6	12002 (85%)
Chen [31]	Artix-7	202$^{(I)}$	270	4	2877	1800	944	8	5	3.4	3668 (52%)

$^{(R2),(R4)}$ The proposed Radix-2 and Radix-4 NTT; $^{(I)}$ Iterative NTT.

INTT immediately. After completing the NTT mode, the NTT module is switched to INTT mode. F CCs are needed to fulfill the pipeline. The polynomial vectors $(\hat{s}_0, \hat{s}_1, \hat{s}_2)$ and the matrix coefficient polynomial $\hat{\mathbf{A}}$ continue to be loaded to perform $\hat{t}_1 = \hat{\mathbf{A}}_{10} \circ \hat{s}_0 + \hat{\mathbf{A}}_{11} \circ \hat{s}_1 + \hat{\mathbf{A}}_{12} \circ \hat{s}_2$, $\hat{t}_2 = \hat{\mathbf{A}}_{20} \circ \hat{s}_0 + \hat{\mathbf{A}}_{21} \circ \hat{s}_1 + \hat{\mathbf{A}}_{22} \circ \hat{s}_2$. During the execution of INTT, there are time intervals ΔT where no polynomial is to be processed. However, as a bonus from the T constraint, the INTT results are unaffected before and after ΔT. Finally, the PWM execution time is absorbed by the NTT and INTT processes. Thus, our accelerator achieves comprehensive optimization of polynomial matrix-vector multiplication execution time.

IV. IMPLEMENTATION RESULTS

In order to demonstrate how efficient the proposed architectures are in terms of speed and hardware performance, they are modelized using Verilog-HDL on the Xilinx Vivado 2021.2 design tool. We select the baseline Xilinx Artix-7 FPGA platform (part number XC7A100tfgg676-3) to implement our proposed designs for a fair comparison with the studies reported to date. The implemented results are obtained after synthesis, place-and-route with default settings.

In this section, the Area-Time-Product (ATP) is reported for fair comparisons. A is the SLICES reported, plus the number of SLICEs converted from DSPs and BRAMs. As in the study [37], we use the conversion ratio of 1 DSP = 100 SLICEs and 1 BRAM = 196 SLICEs. In cases where studies do not report SLICEs, we perform the conversion from LUTs and Flipflops (FFs), corresponding to 1 SLICE = 4×LUTs + 8×FFs for Artix-7 FPGA platform. To compare the time execution of different studies, we report the Speed-up Ratio. Speed-up Ratio is calculated from *Latency* in time (μs), with *Latency* = *CCs* / *Freq*.

A. IMPLEMENTATION RESULTS OF DILITHIUM NTT ARCHITECTURES

Table 2 reports implementation results for the Dilithium unified pipelined NTT architecture, with two versions, Radix-2 MDC ($R2$) and Radix-4 MDC ($R4$). As you can see, these are the first implementations of the pipelined NTT architecture for Dilithium with $q = 8380417$, $n = 256$. The NTTs utilize the proposed DSP, eliminating the complete use of DSP and BRAM resources. The results show that our Radix-2 MDC NTT architecture achieves 180 MHz of frequency with an area of 7451 LUTs, 5275 FFs, and 2456 SLICEs. The Radix-4 MDC NTT architecture achieves a lower frequency of 162 MHz with 13804 LUTs, 11019 FFs, and 4522 SLICEs. The Radix-4 has a faster NTT execution time compared to Radix-2, with 64 CCs and 128 CCs, respectively. Our two NTT designs report the minimum ATP results, corresponding to 1753 and 1764.

In [24], the authors presented a lightweight hardware accelerator with a dual butterfly unit. They utilized two DSPs for coefficient multiplication and a modular reduction unit based on the property $2^{23} \equiv 2^{13} - 1 \ (mod\ q)$ of Dilithium.

TABLE 3. Implementation results for the proposed fully polynomial multiplication accelerator in Kyber compared with other studies.

Work	Device	Freq (MHz)	CCs	Latency (μs)	LUTs	FFs	SLICEs	DSPs	BRAMs	Speed-up Ratio	Area-Time-Product (ATP)
This Work	**Artix-7**	**250**$^{(R2)}$	**277**	**1.11**	**4834**	**4683**	**1794**	**0**	**1.0**	**1.85**	**2214**
		239$^{(R4)}$	**149**	**0.6**	**9511**	**9165**	**3523**	**0**	**1.0**	**1.0**	**2234**
Yaman [22]	Artix-7	172$^{(I)}$	256	1.49	9508	2684	2712	16	35.0	2.39	16646 (86%)
Bisheh [19]	Artix-7	115$^{(I)}$	2365	20.5	737	290	220	6	4.0	34.16	54792 (96%)
Xing [18]	Artix-7	161$^{(I)}$	1152	7.1	1579	1058	527	2	3.0	11.8	9422 (76%)
Dang [21]	Artix-7	229$^{(I)}$	1152	5.0	880	999	345	2	1.5	8.3	4225 (48%)
Duong [38]	Artix-7	265$^{(R2^k)}$	246	0.93	9211	9810	3529	60	1.0	1.5	9048 (75%)
Li [35]	Artix-7	273$^{(I)}$	224	0.82	4619	4166	1641	16	8	1.36	3970 (44%)

$^{(R2),(R4)}$ The proposed Radix-2 and Radix-4 NTT; $^{(R2^k)}$ Mixed-Radix 2^k NTT; $^{(I)}$ Iterative NTT.

Additionally, they used two memory units to support iterative NTT computation. On the other hand, our Radix-4 employs butterfly units 8.0× more than their design. However, it only requires 5.0× and 5.4× LUTs and FFs, respectively. Their NTT/INTT implementation requires 614 CCs with 4 DSPs and 7 BRAMs. Compared to their design, our Radix-4 is 9.6× faster and achieves an 83% improvement in ATP.

In [28], Zhao et al. proposed an NTT architecture based on Radix-2 MDC to reduce the number of memory access and simplify control logic. They used the folding transformation method to reduce the number of butterfly units from eight to four. Their design utilizes fewer than 7.2× LUTs and 8.5× FFs and requires 8 DSPs and 2 BRAMs compared to our Radix-4. Their NTT implementation requires 296 CCs with a low operating frequency of 96.9 MHz. As a result, our design achieves a 7.8× speed-up ratio and improves ATP by 69% compared to their implementation.

In [25], the authors proposed an optimized two-butterfly NTT architecture using DSPs. Specifically, modular addition and subtraction operations in the butterfly unit are designed to be performed by using DSPs. Therefore, their design achieved a maximum frequency of 311 MHz with a hardware footprint of 524 LUTs, 759 FFs, 17 DSPs, and 1 BRAM. However, this frequency is only achieved when synthesizing the NTT separately on the FPGA. Their design required 536 CCs for INTT processing. Compared to their results, our design achieves a speed-up ratio of 4.4× times and a 53% improvement in ATP.

In [27], Beckwith et al. presented a high-performance hardware design for Dilithium. They proposed a 2 × 2 butterfly configuration of parallel processing four pairs of coefficients. Thus, two layers of the NTT are processed on each memory access. However, due to constraints on the order of coefficients during NTT/INTT execution, they had to use the FIFOs for reordering coefficients. As a result, their design required 9018 LUTs, 6292 FFs, and 16 DSPs. This design achieved an operating frequency of 116 MHz and required 300 CCs for NTT execution. Our design has a significantly improved ATP of 85% and a speed-up ratio of 6.6× compared to their implementation results.

In [31], a Dilithium NTT/INTT hardware core and an efficient memory access pattern are proposed. They proposed a configuration of four butterfly cores for parallel computation. This design achieves an operating frequency of 202 MHz with a hardware footprint of 2877 LUTs, 1800 FFs, 8 DSPs, and 5 BRAMs. Our design accelerates approximately 3.4× and improves ATP by 52% compared to the reported implementation results.

B. IMPLEMENTATION RESULTS OF KYBER ACCELERATOR

Table 3 reports the implementation results for the Kyber polynomial matrix-vector multiplication accelerators and comparison with the state-of-the-art studies. Like the case of Dilithium, we propose two accelerator versions using Radix-2 MDC (R2) and Radix-4 MDC (R4). The Radix-2 accelerator achieves an operating frequency of 250 MHz with a hardware footprint of 4834 LUTs, 4683 FFs, 1794 SLICEs, 0 DSPs, and 1.0 BRAM. On the other hand, the Radix-4 accelerator has an operating frequency of 239MHz, 9511 LUTs, 9165 FFs, 0 DSPs, and 1.0 BRAM. The time constraint T enables parallel PWMM operations during NTT and INTT processes. Therefore, the number of clock cycles for one polynomial multiplication is calculated as the sum of the execution cycles of the NTT, INTT, the pipeline PWMM stages (15 CCs), and the pipeline registers between modules (6 CCs). The Radix-2 accelerator requires $277 = 128 + 128 + 15 + 6$ CCs, and the Radix-4 accelerator requires $149 = 64 + 64 + 15 + 6$ CCs. With the proposed DSP design and comprehensive optimizations for the modules, we achieved an excellent hardware efficiency of 2214 and 2234 ATPs.

In [22], Yaman et al. proposed a high-performance hardware architecture for Kyber NTT/INTT execution. Their design employed iterative NTT with 16 butterfly cores working in parallel. Four dual-port BRAMs are utilized to correct the order of polynomial coefficients and twiddle factors for each butterfly unit. Their architecture finishes one polynomial multiplication in 256 CCs. They reported hardware results with an operation frequency of 172 MHz and resource utilization of 9508 LUTs, 2684 FFs, 16 DSPs, and 35 BRAMs. Compared to their design, our Radix-4 hardware accelerator speeds up 2.39× and improves ATP by 85%.

The study [19] proposed a parallel dual-butterfly NTT architecture. The authors utilized two memory units operating on a ping-pong principle to achieve efficient memory access patterns. Their NTT architecture requires 2365 CCs for one polynomial multiplication, consisting of 474 CCs for NTT,

602 CCs for INTT, and 1289 CCs for PWM. Compared to their implementation, our design achieves a speed-up ratio of 34.16× and an ATP hardware efficiency improvement of up to 96%.

In [18], Xing and Li proposed an NTT architecture that performs two parallel butterfly operations for even and odd coefficients. Additionally, the authors utilized the Karatsuba algorithm to reduce the number of point-wise multiplications and execute a PWM operation in 2 CCs. This design finishes NTT in 448 CCs, INTT in 448 CCs, and PWM in 256 CCs. Our accelerator is 11.8× faster than theirs, improving ATP efficiency by 76%.

In [21], Dang et al. proposed a polynomial multiplication accelerator architecture using k double-butterflies and k memory banks, with k as the security level of Kyber. The architecture of each double-butterfly is optimized from the design in [18] to perform parallel NTT for even and odd coefficients. The authors proposed using two reordering units inspired by the MDC architecture to ensure the correct order of coefficients. Like NTT in [18], this architecture requires 1152 CCs for one polynomial multiplication. However, our accelerator improves ATP by 48% and is 8.3× faster than their design.

Study [38] introduced a configurable pipeline NTT architecture based on the multi-path delay feedback architecture. The authors presented an accelerator with two NTT cores to perform NTT and INTT transformations in $n/4$ CCs. However, this design only utilizes one NTT core when computing INTT, resulting in a 50% utilization rate. The reported implementation results show an operating frequency of 265 MHz with a corresponding hardware footprint of 9211 LUTs, 9810 FFs, 60 DSPs, and 1 BRAM. When comparing the implementation reports with theirs, our Radix-4 accelerator achieves a speed-up of 1.5× and improves hardware efficiency by 75% in terms of ATP.

In [35], Li et al. proposed an accelerator using four configurable butterfly cores to execute NTT, INTT, and PWM. Their design includes a binomial multiplier that accelerates PWM execution. This architecture achieves a high operating frequency of 273 MHz with a hardware footprint of 4619 LUTs, 4166 FFs, 16 DSPs, and 8 BRAMs. Compared to their reported results, our accelerator results in a 1.36× speed up and improves ATP efficiency by 44%.

Table 4 shows the FPGA resource breakdown of the proposed architectures in this study. The percentage value indicates the ratio of used hardware resources for the corresponding sub-module in architecture. For Kyber's PMVM, NTT and PWMM are the most hardware-intensive modules. The pipelined NTT architecture has the advantage of simple control, resulting in a tiny proportion of hardware consumption for data control and data memory. Moreover, the hardware consumption of sub-modules, such as MM units (modular multiplication), configured in Figure 2, and BUs (butterfly unit) for the Kyber parameter are reported. Besides, the hardware resources of the Dilithium NTT and its sub-modules (MM, BU) are also reported in Table 4. In the pipelined NTT architecture, BU units occupy most of the hardware resources. In contrast, the remaining resources are used for RU units, storage of twiddle factors, and data control.

TABLE 4. FPGA resource utilization of proposed implementations on the Artix-7 FPGA platform.

Module	LUTs	FFs	BRAMs
CRYSTAL-Kyber: $n = 256$; $q = 3329$			
PMVM$^{(R2)}$	4834 (100%)	4683 (100%)	1.0
⌊ NTT	2466 (51%)	2149 (46%)	0
⌊ BU	289	236	0
⌊ MM	213	176	0
⌊ PWMM	2297 (48%)	2081 (44%)	0
⌊ Control, Memory	69 (1%)	453 (10%)	1.0
PMVM$^{(R4)}$	9511 (100%)	9165 (100%)	1.0
⌊ NTT	4821 (51%)	4261 (46%)	0
⌊ PWM	4614 (48%)	4480 (49%)	0
⌊ Control, Memory	71 (1%)	424 (5%)	1.0
CRYSTAL-Dilithium: $n = 256$; $q = 8380417$			
NTT$^{(R2)}$	7451	5275	0
NTT$^{(R4)}$	13804 (100%)	11019 (100%)	0
⌊ BU (×16)	708 (82%)	526 (76%)	0
⌊ MM	465	289	0
⌊ RUs, Memory	2476 (18%)	2603 (24%)	0

TABLE 5. NTT implementation results for typical parameter sets (n, q) in LBC cryptosystem.

Parameter set (n, q)	Fmax (MHz)	LUTs	FFs	SLICEs
Radix-2 NTT architecture				
(256, 3329)	256	2466	2149	821
(256, 7681)	243	2938	2652	999
(512, 12289)	234	3669	3544	1127
(256, 8380417)	180	7451	5275	2456
Radix-4 NTT architecture				
(256, 3329)	245	4821	4261	1584
(256, 7681)	219	5735	5035	1982
(512, 12289)	208	7388	7151	2460
(256, 8380417)	163	13804	11019	4522

Table 5 presents the NTT implementation results for various parameters (n, q) in the LBC cryptosystems. We deploy both Radix-2 and Radix-4 versions to demonstrate the parameterizable capability of the proposed architecture. The idea behind the unified-pipelined NTT architecture is to design dedicated DSPs for modular multiplication, optimizing NTT transformations within polynomial multiplication. Thus, our designs achieve high efficiency in operating frequency and hardware footprint. Despite a significant reduction in operating frequency between the two NTT versions, our proposed NTT modules show potential for deployment on ASIC platforms.

V. CONCLUSION

This paper proposes optimized designs for core operations in implementing polynomial matrix-vector multiplication for Kyber and Dilithium. The approach begins with proposing a dedicated DSP architecture for efficient large modular multiplication in Dilithium. Subsequently, novel unified pipelined NTT architectures are presented for high-speed (radix-2) and ultra-high-speed (radix-4) NTT versions. Finally, the proposed designs are applied to construct two

accelerators for Kyber. The modules in each accelerator are crafted with execution time constraints to enhance parallel execution capabilities. Our accelerators demonstrate higher efficiency when performing continuous polynomial multiplications. The proposed designs have fast execution times with minimal hardware area.

The implementation results show that our Dilithium NTT accelerator speeds up from 3.4–9.6×, with hardware performance improvement in 52–85% ATP. Correspondingly, our Kyber accelerator speeds up from 1.36–34.16×, with 44–96% ATP improvement. Reports on our architecture may have a worse critical path due to increased net delay and suboptimal logic optimization caused by fixed DSPs on the FPGA platform. However, our design still has a high operating frequency compared to other designs. Therefore, the proposed designs in this paper can be more efficient on ASIC platforms for large-scale high-speed cryptosystem applications. Furthermore, our methods can be applied to implement other PQC algorithms where polynomial multiplication plays a crucial role.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Nov. 1994, pp. 124–134.

[2] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. S. Tone, "Status report on the third round of the NIST post-quantum cryptography standardization process," National Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST IR 8413, 2022, doi: 10.6028/NIST.IR.8413-upd1.

[3] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlè. (Jan. 2021). *CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (Version 3.01)*. [Online]. Available: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf

[4] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. (Feb. 2021). *CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (Version 3.01)*. [Online]. Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

[5] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. (Oct. 2020). *Falcon: Fast-Fourier Lattice-Based Compact Signatures Over NTRU (v1.2)*. [Online]. Available: https://falcon-sign.info/falcon.pdf

[6] J.-P Aumasson, D.-J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. Westerbaan. (Jun. 2022). *SPHINCS+—Submission To the 3rd Round of the NIST Post-quantum Project. V3.1*. [Online]. Available: https://sphincs.org/data/sphincs+-r3.1-specification.pdf

[7] D.-T. Dam, T.-H. Tran, V.-P. Hoang, C.-K. Pham, and T.-T. Hoang, "A survey of post-quantum cryptography: Start of a new race," *Cryptography*, vol. 7, no. 3, p. 40, Aug. 2023.

[8] A. C. Mert, F. Yaman, E. Karabulut, E. Öztürk, E. Savas, and A. Aysu, "A survey of software implementations for the number theoretic transform," in *Proc. Int. Conf. Embedded Comput. Syst.* Cham, Switzerland: Springer, 2023, pp. 328–344.

[9] D. O. C. Greconici, M. J. Kannwischer, and D. Sprenkels, "Compact Dilithium implementations on Cortex-M3 and Cortex-M4," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 1, pp. 1–24, Dec. 2020.

[10] Y. Kim, J. Song, and S. C. Seo, "Accelerating falcon on ARMv8," *IEEE Access*, vol. 10, pp. 44446–44460, 2022.

[11] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels, "Faster Kyber and Dilithium on the Cortex-M4," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, Jun. 2022, pp. 853–871.

[12] J. Huang, J. Zhang, H. Zhao, Z. Liu, R. C. C. Cheung, Ç. K. Koç, and D. Chen, "Improved plantard arithmetic for lattice-based cryptography," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 614–636, Aug. 2022.

[13] W.-K. Lee, S. Akleylek, W.-S. Yap, and B.-M. Goi, "Accelerating number theoretic transform in GPU platform for qTESLA scheme," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Cham, Switzerland: Springer, 2019, pp. 41–55.

[14] N. Gupta, A. Jati, A. K. Chauhan, and A. Chattopadhyay, "PQC acceleration using GPUs: FrodoKEM, NewHope, and Kyber," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 575–586, Mar. 2021.

[15] W.-K. Lee, S. Akleylek, D. C.-K. Wong, W.-S. Yap, B.-M. Goi, and S.-O. Hwang, "Parallel implementation of nussbaumer algorithm and number theoretic transform on a GPU platform: Application to qTESLA," *J. Supercomput.*, vol. 77, no. 4, pp. 3289–3314, Apr. 2021.

[16] Y. Gao, J. Xu, and H. Wang, "CuNH: Efficient GPU implementations of post-quantum KEM NewHope," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 551–568, Mar. 2022.

[17] U. Banerjee, A. Pathak, and A. P. Chandrakasan, "An energy-efficient configurable lattice cryptography processor for the quantum-secure Internet of Things," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 46–48.

[18] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-Kyber on FPGA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 2, pp. 328–356, Feb. 2021.

[19] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Instruction-set accelerated implementation of CRYSTALS-Kyber," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 11, pp. 4648–4659, Nov. 2021.

[20] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *Proc. Cryp. Netw. Secu., Int. Conf. (CANS)*, Nov. 2016, pp. 124–139.

[21] V. B. Dang, K. Mohajerani, and K. Gaj, "High-speed hardware architectures and FPGA benchmarking of CRYSTALS-Kyber, NTRU, and saber," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 306–320, Feb. 2023.

[22] F. Yaman, A. C. Mert, E. Öztürk, and E. Savas, "A hardware accelerator for polynomial multiplication operation of CRYSTALS-Kyber PQC scheme," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1020–1025.

[23] A. Aikata, A. C. Mert, M. Imran, S. Pagliarini, and S. S. Roy, "KaLi: A crystal for post-quantum security using Kyber and Dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 2, pp. 747–758, Feb. 2023.

[24] N. Gupta, A. Jati, A. Chattopadhyay, and G. Jha, "Lightweight hardware accelerator for post-quantum digital signature CRYSTALS-Dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 8, pp. 3234–3243, May 2023.

[25] G. Land, P. Sasdrich, and T. Güneysu, "A hard crystal-implementing Dilithium on reconfigurable hardware," in *Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS)*, Nov. 2021, pp. 210–230.

[26] T. X. Pham, P. Duong-Ngoc, and H. Lee, "An efficient unified polynomial arithmetic unit for CRYSTALS-Dilithium," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 12, pp. 4854–4864, Dec. 2023.

[27] L. Beckwith, D. T. Nguyen, and K. Gaj. (2022). *High-Performance Hardware Implementation of Lattice-Based Digital Signatures*. Cryptol. ePrint Arch. [Online]. Available: https://eprint.iacr.org/2022/217

[28] C. Zhao, N. Zhang, H. Wang, B. Yang, W. Zhu, Z. Li, M. Zhu, S. Yin, S. Wei, and L. Liu, "A compact and high-performance hardware architecture for CRYSTALS-Dilithium," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2022, no. 1, pp. 270–295, Nov. 2022.

[29] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "High-speed NTT-based polynomial multiplication accelerator for post-quantum cryptography," in *Proc. IEEE 28th Symp. Comput. Arithmetic (ARITH)*, Jun. 2021, pp. 94–101.

[30] C. Zhang, D. Liu, X. Liu, X. Zou, G. Niu, B. Liu, and Q. Jiang, "Towards efficient hardware implementation of NTT for kyber on FPGAs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[31] X. Chen, B. Yang, Y. Lu, S. Yin, S. Wei, and L. Liu, "Efficient access scheme for multi-bank based NTT architecture through conflict graph," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 91–96.

[32] Y. Geng, X. Hu, M. Li, and Z. Wang, "Rethinking parallel memory access pattern in number theoretic transform design," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 5, pp. 1689–1693, May 2023.
[33] J. Mu, Y. Ren, W. Wang, Y. Hu, S. Chen, C.-H. Chang, J. Fan, J. Ye, Y. Cao, H. Li, and X. Li, "Scalable and conflict-free NTT hardware accelerator design: Methodology, proof, and implementation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 5, pp. 1504–1517, May 2023.
[34] W. Guo and S. Li, "Highly-efficient hardware architecture for CRYSTALS-Kyber with a novel conflict-free memory access pattern," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 11, pp. 4505–4515, Nov. 2023.
[35] M. Li, J. Tian, X. Hu, and Z. Wang, "Reconfigurable and high-efficiency polynomial multiplication accelerator for CRYSTALS-Kyber," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 8, pp. 2540–2551, Dec. 2022.
[36] Z. Ni, A. Khalid, W. Liu, and M. O'Neill, "Towards a lightweight CRYSTALS-Kyber in FPGAs: An ultra-lightweight BRAM-free NTT core," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5.
[37] Z. Ni, A. Khalid, D.-E.-S. Kundi, M. O'Neill, and W. Liu, "HPKA: A high-performance CRYSTALS-Kyber accelerator exploring efficient pipelining," *IEEE Trans. Comput.*, vol. 72, no. 12, pp. 3340–3353, Dec. 2023.
[38] P. Duong-Ngoc and H. Lee, "Configurable mixed-radix number theoretic transform architecture for lattice-based cryptography," *IEEE Access*, vol. 10, pp. 12732–12741, 2022.
[39] Z. Ye, R. C. C. Cheung, and K. Huang, "PipeNTT: A pipelined number theoretic transform architecture," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 10, pp. 4068–4072, Oct. 2022.
[40] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen, "SWIFFT: A modest proposal for FFT hashing," in *Fast Software Encryption*. Berlin, Germany: Springer, Feb. 2008, pp. 54–72.
[41] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, Apr. 1965.
[42] W. M. Gentleman and G. Sande, "Fast Fourier transforms: For fun and profit," in *Proc. Fall Joint Comput. Conf. AFIPS (Fall)*, Nov. 1966, pp. 563–578.
[43] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-LWE cryptoprocessor," in *Cryptographic Hardware and Embedded Systems*. Berlin, Germany: Springer, Sep. 2014, pp. 371–391.
[44] T. Pöppelmann, T. Oder, and T. Güneysu, "High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers," in *Proc. Int. Conf. Cryptol. Inf. Secur. Latin Amer.*, Aug. 2015, pp. 346–365.
[45] *7 Series DSP48E1 Slice User Guide (UG479)*. Accessed: Feb. 7, 2024. [Online]. Available: https://docs.xilinx.com/v/u/en-U.S./ug479-7Series-DSP48E1
[46] *UltraScale Architecture DSP Slice User Guide (UG579)*. Accessed: Feb. 7, 2024. [Online]. Available: https://docs.xilinx.com/v/u/en-U.S./ug579-ultrascale-dsp
[47] T.-H. Nguyen, C.-K. Pham, and T.-T. Hoang, "A high-efficiency modular multiplication digital signal processing for lattice-based post-quantum cryptography," *Cryptography*, vol. 7, no. 4, p. 46, Sep. 2023.
[48] T. Wang, C. Zhang, P. Cao, and D. Gu, "Efficient implementation of Dilithium signature scheme on FPGA SoC platform," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 9, pp. 1158–1171, Sep. 2022.
[49] X. Hu, J. Tian, M. Li, and Z. Wang, "AC-PM: An area-efficient and configurable polynomial multiplier for lattice based cryptography," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 2, pp. 719–732, Feb. 2023.
[50] D. T. Nguyen, V. B. Dang, and K. Gaj, "A high-level synthesis approach to the software/hardware codesign of NTT-based post-quantum cryptography algorithms," in *Proc. Int. Conf. Field-Programmable Technol. (ICFPT)*, Dec. 2019, pp. 371–374.
[51] C. P. Rentería-Mejía and J. Velasco-Medina, "High-throughput ring-LWE cryptoprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2332–2345, Aug. 2017.
[52] D.-e.-S. Kundi, Y. Zhang, C. Wang, A. Khalid, M. O'Neill, and W. Liu, "Ultra high-speed polynomial multiplications for lattice-based cryptography on FPGAs," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1993–2005, Oct. 2022.

TRONG-HUNG NGUYEN (Graduate Student Member, IEEE) received the B.Sc. degree in control engineering and automation from Hanoi University of Science and Technology (HUST), Hanoi, Vietnam, in 2014, and the M.S. degree in electronic engineering from The University of Electro-Communications (UEC), Tokyo, Japan, in 2022, where he is currently pursuing the Ph.D. degree in information and network engineering. His research interests include digital signal processing, hardware accelerators, and post-quantum cybersecurity.

BINH KIEU-DO-NGUYEN (Graduate Student Member, IEEE) received the B.Sc. and M.S. degrees from the Department of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree in information and network engineering with The University of Electro-Communications (UEC), Tokyo, Japan. From 2017 to 2021, he was a Lecturer Assistant with HCMUT. Since 2022, he was a Research Assistant with UEC. His research interests include computer architecture, hardware security, and digital systems design.

CONG-KHA PHAM (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronics engineering from Sophia University, Tokyo, Japan. He is currently a Professor with the Department of Information and Network Engineering, The University of Electro-Communications (UEC), Tokyo. His research interests include hardware system design implementation by FPGA and integrated circuits. His recent projects include research on energy harvest power supply and low-power data-centric sensor network system utilizing it, development of long-distance transmission/miniaturization equipment of sensor network by low power wireless, super low-voltage device project, research on memory-based information detection systems, hardware implementation of hardware system by FPGA, and integrated circuit. He is teaching many undergraduate and postgraduate students and has received numerous awards for dissertations. The University of Electro-Communications Integrated Circuit Design Laboratory (Pham Laboratory) educates the design, implementation, evaluation of hardware systems and VLSI, aims to design a system-on-chip by integrating various information processing hardware, and develops a high-performance computational circuit realized with a small number of elements.

TRONG-THUC HOANG (Member, IEEE) received the B.Sc. and M.S. degrees in electronic engineering from Ho Chi Minh City University of Science (HCMUS), Ho Chi Minh City, Vietnam, in 2012 and 2017, respectively, and the Ph.D. degree in engineering from The University of Electro-Communications (UEC), Tokyo, Japan, in 2022. From 2012 to 2017, he was a Lecturer Assistant with HCMUS. From 2019 to 2020, he was a Research Assistant with UEC. From 2019 to 2022, he was a Research Assistant with the Cyber-Physical Security Research Center (CPSEC), National Institute of Advanced Industrial Science and Technology (AIST), Tokyo. Since April 2022, he has been an Assistant Professor with the Department of Computer and Network Engineering, UEC. His research interests include digital signal processing, computer architecture, cyber-security, and ultra-low power systems-on-a-chip.