

Article

A Configurable and Highly Efficient NTT/INTT/PWM Unit for CRYSTALS-Kyber and CRYSTALS-Dilithium

Yuchen Wang ^{1,2,†}, Xiaoke Wang ^{3,†}, Chaoxing You ^{1,2}, Shuang Xia ⁴, Kainan Ma ^{1,*}  and Ming Liu ^{1,2,*}

¹ Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China; wangyuchen245@mailsucas.ac.cn (Y.W.); youchaoxing23@mailsucas.ac.cn (C.Y.)

² University of Chinese Academy of Sciences, Beijing 100049, China

³ College of Mechanical Engineering, University of Hong Kong, Hong Kong, China; u3636546@connect.hku.hk

⁴ School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China; x_shuang@cumt.edu.cn

* Correspondence: makainan@semi.ac.cn (K.M.); liuming@semi.ac.cn (M.L.)

† These authors contributed equally to this work.

Abstract

CRYSTALS-Kyber and CRYSTALS-Dilithium are representative lattice-based post-quantum cryptographic schemes, where number theoretic transform (NTT), inverse NTT (INTT), and point-wise multiplication (PWM) dominate polynomial arithmetic. Existing hardware accelerators are typically optimized for a single scheme or operation, resulting in limited scalability and redundant hardware resources. This paper presents a unified and scalable NTT/INTT/PWM architecture for both Kyber and Dilithium. A configurable two-dimensional processing-element (PE) array combined with an EVEN/ODD ping-pong memory organization enables conflict-free memory access and partial inter-stage pipelining. A unified K-RED-based modular multiplier supports either one Dilithium multiplication or two parallel Kyber multiplications using the same DSP resources, while PWM is mapped onto the existing PE array without requiring a separate complete PWM arithmetic array. Experimental results on a Xilinx Artix-7 FPGA show that the proposed modular multiplier reduces LUT utilization by 39.8% compared with the previous unified Kyber/Dilithium design. Under matched PE configurations, the proposed architecture achieves ATP reductions of up to 79.5% for Dilithium NTT/INTT and 73.9% for Kyber NTT/INTT, while the Kyber/Dilithium PWM mode achieves an ATP reduction of up to 78.3%. These results demonstrate that the proposed architecture provides an efficient, scalable, and highly reusable hardware solution for lattice-based post-quantum cryptographic accelerators.

Keywords: CRYSTALS-Dilithium; CRYSTALS-Kyber; NTT; configurable hardware architecture; point-wise multiplication; modular multiplication



Academic Editors: Esteban Tielo-Cuautle, Alexander Barkalov, Lobna A. Said, Georgia Tsirimokou and Costas Psychalinos

Received: 4 July 2026

Revised: 31 July 2026

Accepted: 3 August 2026

Published: 7 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the rapid development of quantum computing, conventional public-key cryptographic schemes based on integer factorization and discrete logarithms are facing potential security threats. Post-quantum cryptography (PQC) has therefore become an essential direction for future secure communication systems [1,2]. Among various PQC candidates, lattice-based cryptography has attracted significant attention due to its strong security foundation, efficient arithmetic structure, and suitability for hardware implementation. In particular, CRYSTALS-Kyber and CRYSTALS-Dilithium are representative lattice-based schemes for key encapsulation and digital signature applications, respectively. Both schemes rely

heavily on polynomial arithmetic over finite rings, where the Number Theoretic Transform (NTT), point-wise multiplication (PWM), and inverse NTT (INTT) are widely adopted to accelerate polynomial multiplication [3,4].

Although NTT/INTT significantly reduces the computational complexity of polynomial multiplication, it remains one of the major performance bottlenecks in hardware implementations of Kyber and Dilithium. For Dilithium, existing works have optimized butterfly units, modular multipliers, memory-access patterns, and NTT architectures to improve throughput and reduce arithmetic delay [4–21]. For Kyber, polynomial multiplication and matrix–vector multiplication are also critical operations, and representative accelerators optimize NTT-based computation, memory scheduling, and Datapath utilization [22–29]. Moreover, recent PQC hardware frameworks and configurable NTT accelerators further demonstrate the importance of adaptable and reusable architectures for lattice-based cryptographic workloads [30,31].

However, existing accelerators still exhibit insufficient resource reuse across the complete NTT/PWM/INTT computation flow. Gao et al. [27] unified the NTT/INTT butterfly Datapath but did not implement PWM, so supporting the complete polynomial-multiplication flow would require additional coefficient-multiplication hardware. Tsai et al. [25] supported NTT, PWM, and modular accumulation, but employed a dedicated multiplier array for PWM alongside the butterfly and adder arrays. The RNS-based design in [12] shares modular multiplication resources across operations, but its decomposition, recombination, routing, and butterfly-specific logic cannot be fully reused during PWM. Similarly, ref. [21] supports multiple operations through configurable BFUs, but additional multiplexers, pipeline registers, and partially underutilized arithmetic paths remain. Therefore, arithmetic-unit and Datapath reuse between NTT/INTT and PWM is still incomplete, resulting in redundant or underutilized hardware. In contrast, the proposed architecture maps PWM directly onto the existing NTT/INTT PE array and extensively reuses the dominant arithmetic, memory, pipeline, and control resources, thereby minimizing incremental hardware and reducing redundancy.

Another challenge arises from the parameter differences between Kyber and Dilithium. Although both schemes are based on polynomial arithmetic, they employ different moduli, reduction methods, coefficient widths, and transform schedules. A design optimized for only one scheme may achieve high performance for a specific parameter set, but it often lacks flexibility when extended to another scheme. Therefore, a configurable arithmetic Datapath capable of supporting different cryptographic parameters is highly desirable.

Existing NTT accelerators can generally be classified into in-place, ping-pong, and pipelined architectures. In-place architectures [9–13] reduce memory requirements by overwriting intermediate coefficients in the same memory space. However, increasing the number of parallel PEs may cause multiple butterfly operations to access the same memory banks or address regions simultaneously, resulting in severe data conflicts and requiring additional address mapping, data reordering, and conflict-resolution logic. Ping-pong architectures [14–16] alleviate memory conflicts by alternately storing intermediate coefficients in different memory banks, but introduce additional BRAM resources and memory-bandwidth overhead. Pipelined architectures [7,17,18] improve throughput through deep pipeline structures, but usually require dedicated pipeline resources and are optimized for specific parallel configurations. Furthermore, ref. [7] indicates that although arithmetic units and BRAMs increase proportionally with the number of PEs, the complexity of the associated control logic increases exponentially with PE parallelism, limiting the scalability and area efficiency of highly parallel NTT architectures. Therefore, achieving high throughput while maintaining low resource overhead and scalability remains a major challenge.

To address these limitations, this paper proposes a scalable and highly efficient unified NTT/INTT/PWM architecture for lattice-based cryptography. The proposed design supports both Kyber and Dilithium by integrating a unified modular multiplication Datapath. Instead of constructing independent arithmetic units for different schemes, the proposed modular multiplier reuses a shared computation structure and adapts to different modulus requirements. For Dilithium, an optimized K-RED-based modular multiplication method is adopted to reduce the resource cost of conditional correction and shorten the critical path [32]. For Kyber, two coefficients are packed and processed in parallel within the same modular multiplication framework, enabling two Kyber modular multiplications to be executed using the same DSP resources.

Furthermore, this work extends the NTT/INTT Datapath to support efficient PWM computation without introducing a separate complete PWM or accumulation array. The proposed PWM mode reuses the processing elements and modular arithmetic resources originally designed for NTT/INTT butterfly computation. By reorganizing the dataflow and control schedule, the same arithmetic units perform coefficient-wise multiplication and accumulation in the NTT domain. Dilithium PWM requires no additional arithmetic unit, while Kyber base multiplication introduces only one auxiliary 12-bit modular multiplier per PE group. This reuse strategy avoids duplicated multiplier and adder arrays, improves hardware utilization, and reduces unnecessary DSP and logic overhead. As a result, the proposed architecture supports the complete polynomial multiplication flow, including NTT, PWM, and INTT, rather than accelerating only the transform stages.

In terms of architecture, the proposed design employs a configurable two-dimensional processing-element array and a conflict-free memory-access pattern. Intermediate data are alternately stored in EVEN and ODD memory banks, and adjacent stages can be partially overlapped through carefully scheduled read and write operations. Unlike conventional in-place architectures, the proposed design avoids complex data-reordering and conflict-resolution logic. Compared with ping-pong architectures, the proposed memory organization reduces the storage overhead associated with parallel processing. In addition, the number of PE pairs can be configured to support different parallelism levels, allowing the same architecture to scale across low-area, balanced, and high-throughput design points.

The main contributions of this paper are summarized as follows.

1. A unified and scalable NTT/INTT/PWM architecture is proposed for both Kyber and Dilithium. The architecture supports multiple parallelism levels and can be adapted to different resource and performance requirements.
2. A unified modular multiplication Datapath is designed for Kyber and Dilithium. For Dilithium, a K-RED-based hardware optimization flow with LUT-assisted high-bit compression is introduced to reduce conditional-correction overhead and improve resource efficiency, while the standard K-RED formulation remains unchanged. For Kyber, two modular multiplications are executed in parallel within the same arithmetic framework to improve DSP utilization and hardware reuse.
3. An efficient PWM computation mode is integrated into the proposed architecture. By reusing the existing NTT/INTT processing elements and modular arithmetic resources, the design supports point-wise multiplication for Kyber and Dilithium without introducing a separate complete PWM arithmetic array; only one auxiliary 12-bit modular multiplier is added for Kyber base multiplication.
4. A conflict-free two-dimensional processing-element architecture with EVEN/ODD memory banks is developed. The proposed scheduling method enables inter-stage overlap, reduces memory-access conflicts and control complexity, and improves throughput under different PE configurations.

Implementation results on Xilinx Artix-7 demonstrate the efficiency and scalability of the proposed architecture. The proposed unified Kyber/Dilithium modular multiplication unit reduces LUT utilization by 39.8% compared with the previous unified multiplier while maintaining the same DSP and BRAM consumption. Under iso-PE comparisons, the proposed design achieves ATP reductions of up to 79.5% for Dilithium NTT/INTT and 73.9% for Kyber NTT/INTT. For the $PE = 1 \times 2$ configuration, the Kyber/Dilithium PWM mode reduces ATP by 57.0% to 78.3% compared with prior designs. These results demonstrate that the proposed architecture provides a scalable and resource-efficient hardware framework for unified NTT/INTT, PWM, and modular multiplication operations across Kyber and Dilithium.

The remainder of this paper is organized as follows. Section 2 introduces the background of CRYSTALS-Kyber, CRYSTALS-Dilithium, NTT/INTT algorithms, point-wise multiplication, and K-RED modular reduction. Section 3 presents the proposed unified PE, Kyber/Dilithium modular multiplier, NTT/INTT/PWM architecture, and memory scheduling strategy. Section 4 provides FPGA implementation results and comparisons with previous works. Finally, Section 5 concludes this paper.

2. Background

2.1. CRYSTALS-Kyber and CRYSTALS-Dilithium

CRYSTALS-Kyber and CRYSTALS-Dilithium, standardized by NIST as representative lattice-based post-quantum cryptographic schemes, provide key encapsulation and digital signature functions, respectively. Both rely on structured polynomial arithmetic over finite rings rather than costly operations such as Gaussian sampling, making them suitable for efficient hardware implementation. Polynomial multiplication is a dominant computation in both schemes and is commonly accelerated using the Number Theoretic Transform (NTT), inverse NTT (INTT), and point-wise multiplication (PWM).

2.2. Number Theoretic Transform

NTT is an efficient algorithm for accelerating polynomial multiplication. It reduces the time complexity from $O(n^2)$ to $O(n \log n)$, making it significantly faster for large-scale computations. Both Kyber and Dilithium adopt NTT/INTT-based polynomial multiplication, but their arithmetic parameters are different. Kyber operates with the modulus ($q = 3329$), while Dilithium uses a much larger modulus ($q = 8,380,417$). Therefore, the modular multiplication bit-width and reduction complexity of Dilithium are significantly higher than those of Kyber. In addition, although both schemes use degree-256 polynomial arithmetic, their NTT structures are not identical. Kyber adopts a 7-stage NTT/INTT structure, whereas Dilithium adopts an 8-stage NTT/INTT structure. These differences make it necessary to design a unified but configurable NTT/INTT Datapath that can support different moduli, arithmetic bit-widths, and transform schedules.

$$N_k = \sum_{j=0}^{n-1} n_j \omega^{jk} \bmod q \quad (1)$$

In (1), the input is the coefficient representation of a polynomial of degree at most $n - 1$. The NTT employs a primitive n -th root of unity in the finite field modulo q , mapping the polynomial to its point-value form. The operation of the INTT is shown in Equation (2).

$$n_k = \frac{1}{N} \sum_{j=0}^{n-1} N_j \omega^{-jk} \bmod q \quad (2)$$

The inverse NTT (INTT) is the reverse of the NTT, mapping from the point domain back to the coefficient domain.

2.3. K-RED Modular Reduction

Modular reduction is a fundamental operation in NTT/INTT and PWM, since the intermediate results of modular multiplication must be reduced back to the finite field. In this work, both Kyber and Dilithium are supported within a unified arithmetic framework.

The standard K-RED procedure used in this work is summarized in Algorithm 1. In Kyber and Dilithium, compared with Barrett [33] and Montgomery [34] reduction, the K-RED algorithm [32] provides an efficient reduction method for moduli of the form $q = k \times 2^m + 1$. Kyber satisfies $q = 3329 = 13 \times 2^8 + 1$, while Dilithium satisfies $q = 8,380,417 = 1023 \times 2^{13} + 1$.

Algorithm 1. K-RED algorithm

- 1: **Input:** $x[n:0] = a \cdot b$, $q = k \cdot 2^m + 1$.
Parameter setting:
 - 2: Kyber: $q = 3329 = 13 \cdot 2^8 + 1$, $k = 13$, $m = 8$;
Dilithium: $q = 8,380,417 = 1023 \cdot 2^{13} + 1$, $k = 1023$, $m = 13$.
 - 3: **Output:** $KRED(x) = x \cdot k \bmod q$.
 - 4: $r_0 = x[m - 1:0]$.
 - 5: $r_1 = x[n:m]$.
 - 6: $y = k \cdot r_0 - r_1$.
 - 7: **return** y .
-

This reduction mainly uses shift, addition, and subtraction operations, avoiding complex division or large-width modular correction. Therefore, it is well suited for hardware implementation, especially in highly parallel NTT/INTT architectures where the modular reduction unit is repeatedly instantiated inside processing elements. In this work, Kyber and Dilithium are supported through a unified K-RED-based modular reduction Datapath with configurable parameters k and m . By changing these parameters, the same reduction structure can adapt to the different moduli and bit-width requirements of Kyber and Dilithium, thereby improving hardware reuse and reducing redundant arithmetic resources.

2.4. Point-Wise Multiplication

Point-wise multiplication (PWM) is an essential operation after NTT transformation for polynomial multiplication in lattice-based cryptography. Although Kyber and Dilithium both employ PWM in the NTT domain, their arithmetic structures are different due to the different polynomial representations and modulus parameters.

For Dilithium, the PWM operation is implemented as a straightforward coefficient-wise modular multiplication. Given two transformed polynomials A and B , each output coefficient is independently calculated as:

$$C_i = A_i \times B_i \bmod q \quad (3)$$

where the Dilithium modulus is $q = 8,380,417$. Since each coefficient has a width of approximately 24 bits, Dilithium PWM mainly requires a 24-bit modular multiplier followed by modular reduction. Therefore, the hardware implementation of Dilithium PWM is dominated by the efficiency of the modular multiplication unit.

Different from Dilithium, Kyber PWM adopts a more complex base multiplication structure. Due to the negacyclic polynomial multiplication property and the NTT decomposition, two adjacent coefficients are grouped into a degree-one polynomial:

$$(a_0 + a_1x)(b_0 + b_1x) \quad (4)$$

where $x^2 = \zeta$, and ζ is the corresponding root of unity. The multiplication can be expressed as:

$$r_0 = (a_0b_0 + \zeta a_1b_1) \bmod q \quad (5)$$

$$r_1 = (a_0b_1 + a_1b_0) \bmod q \quad (6)$$

where all operations are performed modulo $q = 3329$. Therefore, one Kyber base multiplication requires multiple modular multiplications, a multiplication with the twiddle factor, and modular additions. Compared with Dilithium PWM, Kyber PWM introduces additional multiplication and accumulation operations, resulting in higher arithmetic complexity and stricter requirements on the modular multiplication Datapath.

Therefore, an efficient unified PWM architecture should not only support the simple coefficient-wise multiplication required by Dilithium, but also efficiently execute the base multiplication operation required by Kyber. This motivates the proposed unified modular multiplication Datapath and reusable processing-element architecture for Kyber and Dilithium.

3. NTT Unit

3.1. Modular Multiplication

Modular multiplication is the critical bottleneck in NTT/INTT and PWM computations. Dilithium adopts a 23-bit modulus $q = 8,380,417$, where $q = 2^{23} - 2^{13} + 1$. In [4], a 23-bit $\times$ 23-bit multiplication is implemented using two serial DSPs, resulting in a long critical path. To overcome this limitation, ref. [7] employs two parallel DSPs by splitting one operand into 12-bit and 11-bit parts; however, the required folding operations introduce additional add/subtract overhead. In contrast, ref. [17] applies K-RED to compress and recombine the partial products into a 37-bit value, followed by one additional K-RED and conditional correction, achieving lower resource consumption.

However, to support scalable NTT/INTT architectures with different PE configurations, further improvement in hardware efficiency is required because resource overhead increases with the number of PEs. The standard K-RED formulation itself is not modified in this work. Instead, a K-RED-based hardware optimization flow is introduced to reduce the correction overhead after modular reduction. In [17], the intermediate result lies in the range $[-q, 2q)$, requiring two boundary checks to normalize it into $[0, q)$: q is added when the result is negative, and q is subtracted when the result is greater than or equal to q . These correction paths increase both hardware cost and latency. Therefore, as shown in Algorithm 2, a LUT-assisted high-bit compression step is inserted before the final K-RED operation to reduce the intermediate range and simplify the final conditional correction.

On this basis, this work proposes a unified Kyber/Dilithium modular multiplier, as shown in Figure 1. The design adopts a unified 24-bit input format with two modes. In Dilithium mode, the same two DSPs perform one modular multiplication on a 24-bit coefficient. In Kyber mode, each input contains two packed 12-bit coefficients, allowing the two DSPs to execute two independent modular multiplications in parallel. Thus, the multiplier supports either one Dilithium multiplication or two Kyber multiplications without additional DSP resources.

Algorithm 2. Proposed K-RED-based modular multiplication flow**Input:** a, b (23 bit), $a, b \in [0, q]$, $q = 8,380,417$;**Result:** $r = k^2 \cdot a \cdot b \bmod q$.

- 1: $\text{mult}_0 = a[22:12] \cdot b$
- 2: $\text{mult}_1 = a[11:0] \cdot b$
- 3: $k_red_0 = \text{K-RED}(\text{mult}_0)$
- 4: $k_red_1 = \text{K-RED}(\text{mult}_1)$
- 5: $p = (k_red_0 \ll 12) + k_red_1$
- 6: $\text{LUT} = \text{LUT}(p[36:33]) = (p[36:33] \cdot 2^{33} \cdot k) \bmod q$
- 7: $\text{compress} = \text{LUT} + p[32:0]$
- 8: $k_red_2 = \text{K-RED}(\text{compress})$
- 9: $r = (k_red_2 > 0) ? k_red_2 : k_red_2 + q$
- 10: **return** $r[22:0]$

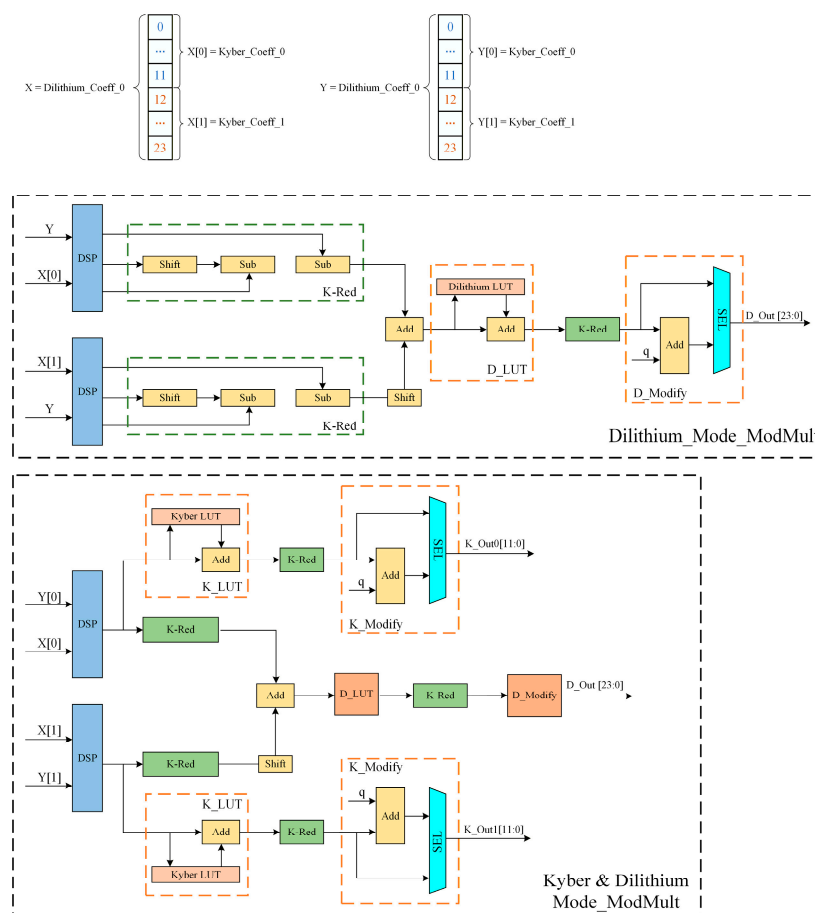


Figure 1. Unified modular-multiplication (ModMult) structure. Blue blocks are DSP multipliers; yellow blocks are shift/add/subtract units; green blocks are K-RED units; orange blocks are LUT-assisted correction or modification units; cyan blocks are selectors; dashed boxes group functional submodules and operating modes.

In the Dilithium mode, the upper Datapath in Figure 1 follows the computation flow of Algorithm 2. The two DSPs generate partial products, which are first processed by K-RED modules. The Shift and Add units then align and recombine the partial results. After recombination, the D_LUT block provides a precomputed correction according to the upper bits of the intermediate result, thereby compressing the intermediate value before the final reduction. This LUT-assisted compression reduces the range and complexity of the

following correction stage, so that only one final conditional correction is required. Finally, the K-RED and D_Modify blocks complete the final reduction and conditional correction to generate the Dilithium result.

After the LUT compression step, the high-order contribution of the intermediate value p is folded into the low-order bits, and the compressed result is reduced to a 34-bit representation. For the final K-RED operation, according to Algorithm 1, the input is decomposed into r_1 and r_0 , where r_1 is a 21-bit value and r_0 is a fixed 13-bit value. Since the K-RED constant is $k = 1023$, the intermediate result can be expressed as:

$$K\text{-RED}(\text{compress}) = k \times r_0 - r_1 \quad (7)$$

The maximum value of $k \times r_0$ is bounded by the modulus q , while r_1 is also bounded by q . Therefore, the output range of the final K-RED operation is limited to:

$$-q < K\text{-RED}(\text{compress}) < q \quad (8)$$

Consequently, only one conditional correction is required. If the output is negative, adding q maps the result into the standard modular range $[0, q)$; otherwise, the result can be directly used. This guarantees the correctness of the modular multiplication while eliminating multiple conditional correction operations.

In the Kyber mode, the lower Datapath in Figure 1 is used. Since Kyber has a smaller modulus, each 24-bit input can be divided into two 12-bit coefficients. The two DSPs then operate as two parallel multiplication lanes. Similarly to the Dilithium mode, the Kyber multiplier also adopts a K-RED and LUT-assisted reduction method, as described in Algorithm 1. The K-RED block compresses the DSP output, the K_LUT block provides the Kyber-specific correction value, and the K_Modify and SEL blocks complete the final correction and output selection. $q = 13 \times 2^8 + 1$.

The main operators in Figure 1 are shared between the two modes. DSPs perform multiplication, while Shift and Add/Sub units handle partial-product alignment, recombination, and K-RED-related operations. K-RED provides modular compression, and K_LUT/D_LUT with K_Modify/D_Modify complete modulus-specific correction.

Overall, the proposed modular multiplier exploits the 24-bit input format to unify Kyber and Dilithium arithmetic. The same two DSPs perform one Dilithium multiplication or two parallel Kyber multiplications, while the shared K-RED and LUT-assisted reduction logic improves resource utilization and avoids redundant arithmetic units, providing an efficient basis for the unified NTT/INTT/PWM architecture.

3.2. Unified PE

Figure 2 illustrates the proposed unified CT/GS butterfly and Kyber/Dilithium PWM processing-element architecture. The upper part shows the unified CT/GS butterfly within a single PE, while the lower part presents the PWM architecture implemented by reusing the arithmetic resources of two PEs.

Each PE contains one shared KD_ModMult, one KD_ModAdd, and one KD_ModSub. By controlling the internal multiplexers, the same PE can switch between CT-based NTT and GS-based INTT operations. In CT mode, twiddle-factor multiplication is followed by modular addition and subtraction. In GS mode, modular addition and subtraction are performed first, and the subtraction result is then forwarded to the modular multiplier. Thus, the same arithmetic resources support both NTT and INTT without separate CT and GS processing units.

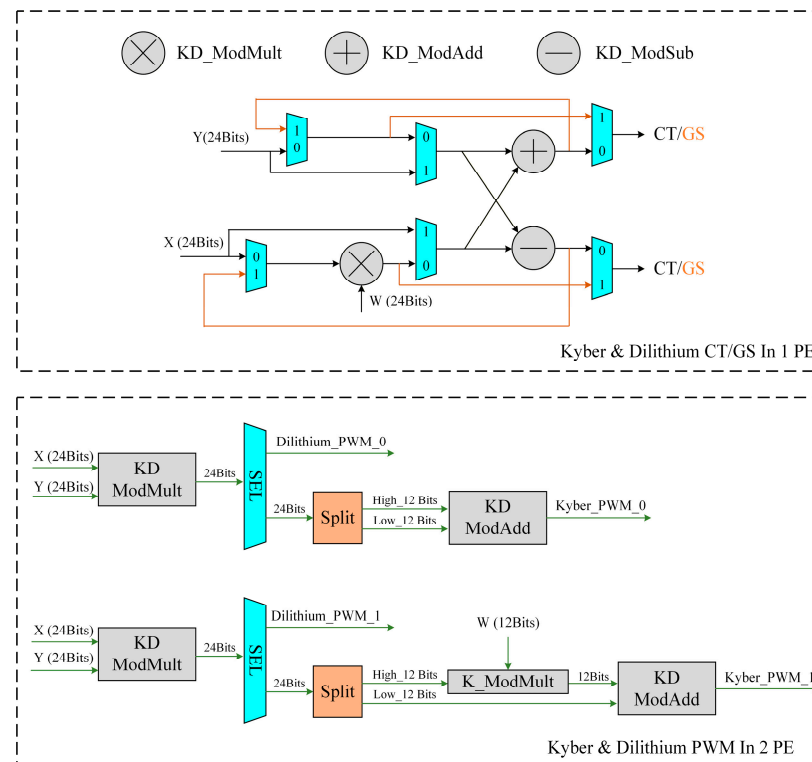


Figure 2. Unified CT/GS butterfly and Kyber/Dilithium PWM architecture based on PEs. Gray blocks are modular arithmetic units, cyan blocks are selectors, orange paths indicate GS-mode feedback and routing, and green paths indicate PWM dataflow.

The lower part of Figure 2 shows how PWM is mapped onto a two-PE group. In Dilithium mode, PWM consists of coefficient-wise modular multiplication. The KD_ModMult units in the two PEs independently perform two 24-bit modular multiplications and directly generate two PWM outputs. Therefore, Dilithium PWM fully reuses the existing multiplication resources without additional arithmetic units.

In Kyber mode, PWM is implemented as a base multiplication that generates two output coefficients, (r_0) and (r_1) , from four input coefficients. The computation of (r_0) requires three modular multiplications and one modular addition, whereas (r_1) requires two modular multiplications and one modular addition. A two-PE group already contains two KD_ModMult units and two KD_ModAdd units. Since each KD_ModMult performs two parallel 12-bit Kyber multiplications, one KD_ModMult generates two products required by (r_0) , while an additional 12-bit K_ModMult performs the remaining twiddle-factor-related multiplication. The existing KD_ModAdd then completes the accumulation for (r_0) . Meanwhile, the other KD_ModMult generates the two products required by (r_1) , and its KD_ModAdd completes the corresponding modular addition.

Each KD_ModMult produces a packed 24-bit output containing two 12-bit multiplication results. The Split blocks in Figure 2 are implemented by static bit slicing and therefore introduce neither arithmetic logic nor an additional pipeline stage. The auxiliary K_ModMult consists of K_LUT, K-RED, and K_Modify blocks and is used only for the additional small-width multiplication required by (r_0) .

With this mapping, a two-PE group produces either two Dilithium PWM outputs or the two outputs of one Kyber base multiplication. Dilithium PWM completely reuses the existing KD_ModMult units, while Kyber PWM reuses the multiplication and addition Datapaths and adds only one 12-bit K_ModMult together with lightweight multiplexing and control logic. KD_ModSub remains inactive during PWM execution. Consequently,

the proposed architecture avoids a separate PWM arithmetic array and minimizes the additional hardware required for the unified NTT/PWM/INTT Datapath.

3.3. The Hardware Architecture of NTT/INTT/PWM

Figure 3 illustrates the proposed unified NTT/INTT/PWM hardware architecture. The architecture consists of an address generation and control unit, two alternating data memory banks (EVEN BRAM and ODD BRAM), a dedicated Twiddle Factor BRAM, two Mux_Groups for data routing, and a configurable 2D PE array. The address generation and control unit generates read/write addresses, selects the corresponding twiddle factors, and controls the dataflow among memory banks, multiplexers, and processing elements.

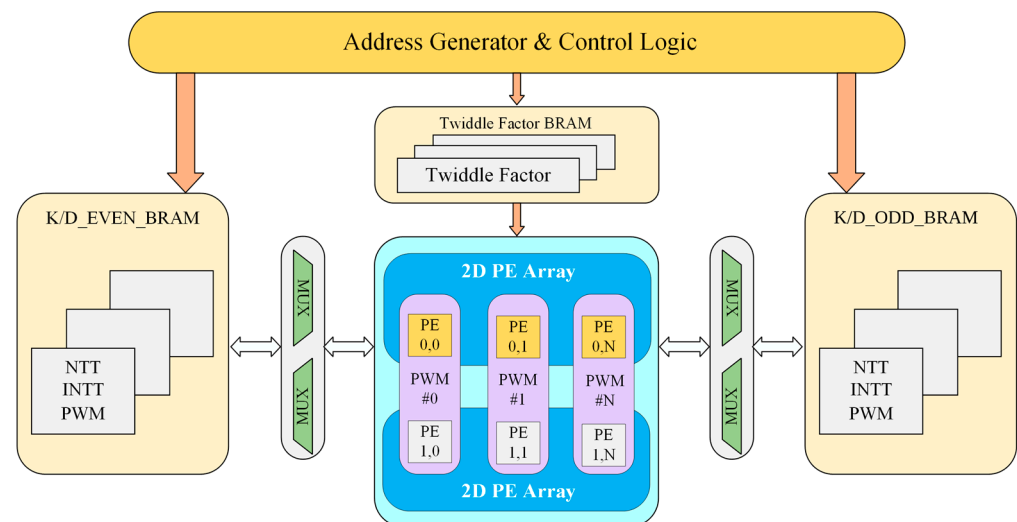


Figure 3. Proposed NTT/INTT/PWM hardware architecture. Yellow blocks denote control and memory modules, blue blocks denote the 2D PE array, purple columns denote PE pairs/PWM lanes, and green blocks denote multiplexers. The symbols #0, #1, and #N identify PWM lane indices.

The proposed architecture adopts an EVEN/ODD ping-pong memory organization to store intermediate data during NTT/INTT computation. The source and destination BRAMs are determined according to the parity of the current stage. For even stages, data are read from EVEN BRAM, processed by the even-stage PE group, and written into ODD BRAM. For odd stages, the dataflow direction is reversed, where data are read from ODD BRAM and written back into EVEN BRAM. This alternating memory mapping enables adjacent stages to execute in an overlapped manner when intermediate results become available.

The 2D PE array is organized into multiple configurable PE pairs. Each PE pair consists of one PE assigned to even-stage computation and one PE assigned to odd-stage computation. The PE-pair structure enables adjacent transform stages to be processed by different PE groups, forming an inter-stage pipeline. The number of PE pairs can be flexibly configured according to different area and throughput requirements. Each PE integrates a unified butterfly Datapath with configurable modular arithmetic units, supporting CT-based NTT and GS-based INTT operations. In addition, the same PE array reuses modular multiplication resources to execute PWM operations without introducing a separate complete PWM Datapath.

Figure 4 illustrates the detailed memory mapping strategy under the configuration of $PE = 2 \times 2$ and $N = 16$. Four PEs are divided into two PE groups: PE(0, 0) and PE(0, 1) are responsible for even-stage computation, while PE(1, 0) and PE(1, 1) are responsible for odd-stage computation. These two PE groups are not independent duplicated structures but cooperate through stage alternation.

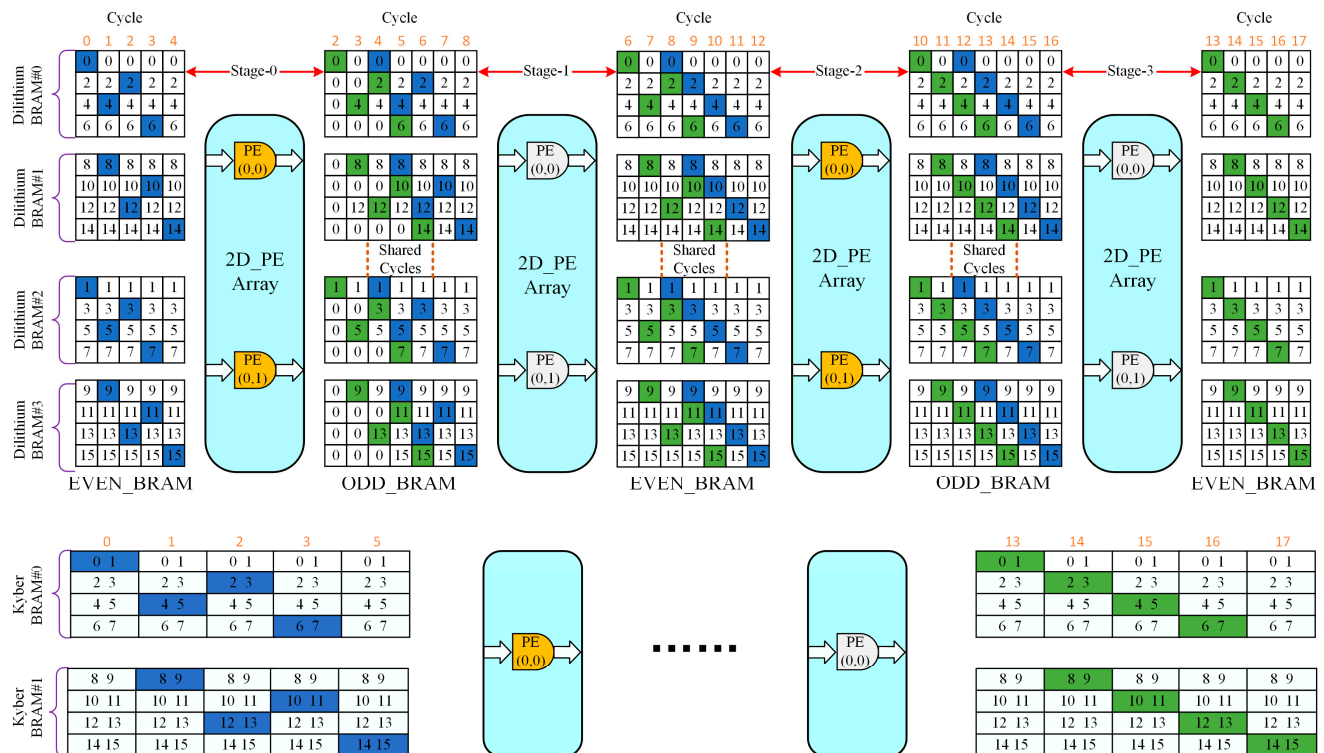


Figure 4. Memory-access pattern of Dilithium/Kyber NTT ($N = 16$, $PE = 2 \times 2/1 \times 2$). Blue blocks denote read operations, green blocks denote write operations, red arrows indicate stage transitions, orange numbers indicate clock cycles, dashed orange brackets identify shared cycles, yellow and gray PE blocks denote active and inactive PE groups, respectively, and # identifies the BRAM-bank index.

During Stage 0, PE(0, 0) and PE(0, 1) read butterfly operands from EVEN_BRAM and write the generated results into ODD_BRAM. After the required intermediate results become available in ODD_BRAM, PE(1, 0) and PE(1, 1) start Stage 1 computation from ODD_BRAM and write the results back into EVEN_BRAM. The same scheduling rule is applied to subsequent stages: even stages use the dataflow direction from EVEN_BRAM to ODD_BRAM, while odd stages use the reverse direction. Therefore, adjacent stages can partially overlap during execution, as illustrated by the overlapped cycles in Figure 4.

Because adjacent stages use different butterfly access orders, the following stage is launched only after the operands required by its early accesses have been written into the opposite BRAM. The start offset is precomputed from the fixed read schedules and the PE read-to-write latency, rather than assumed to be a constant $N/4$ for every configuration. This dependency-aware schedule enables partial inter-stage overlap while preserving a regular access pattern. The proposed memory mapping strategy also simplifies address generation. The controller only needs to select the source and destination BRAMs according to stage parity and activate the corresponding PE group according to the stage index. Due to the regularity of the access pattern, the same control logic can be reused for different PE configurations and transform sizes. Moreover, the structural symmetry between CT-based NTT and GS-based INTT allows the same scheduling mechanism to be applied to both operations.

For the proposed four-PE parallel architecture, the 16-point NTT dataflow is illustrated in Figure 5. In this configuration, PE denotes the total number of processing elements. The four PEs execute butterfly operations in parallel, and each stage therefore requires $\left(\frac{N}{4} + 1\right)$ read operations. The blue blocks in Figure 5 represent data read operations, while the green blocks indicate the corresponding write-back operations after butterfly computation. The

interval between data reading and writing is determined by the internal PE processing latency, denoted as x .

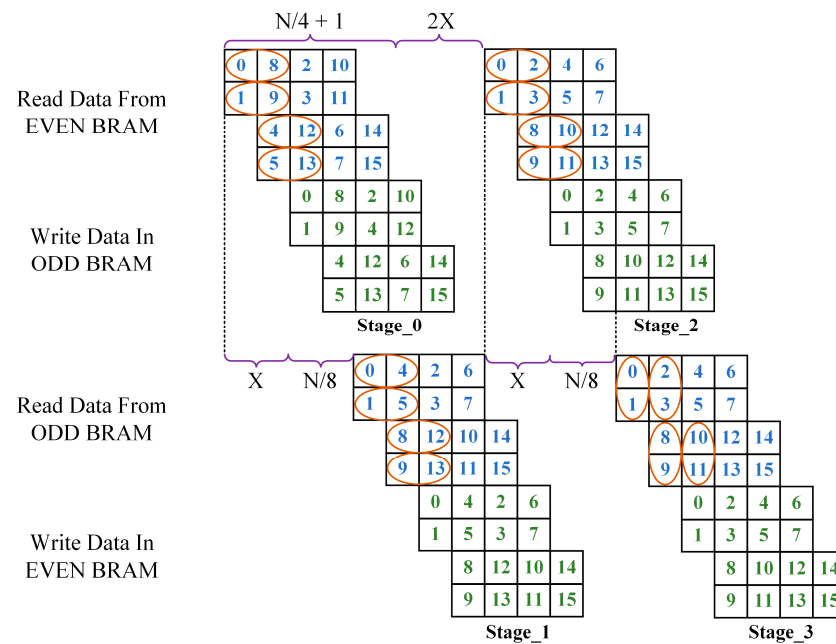


Figure 5. Dataflow diagram of a 16-point NTT with four PEs. Blue values denote read operands, green values denote write-back results, orange circles identify butterfly operand pairs, and braces indicate access-interval or inter-stage-delay terms.

Although the proposed EVEN/ODD ping-pong memory organization separates the data storage of adjacent stages, the next stage cannot start immediately after the current stage begins, because it depends on the intermediate results generated by the previous stage. Therefore, an appropriate inter-stage delay is required to ensure that the required operands have already completed the read-compute-write process and have been stored in the target BRAM before being accessed by the following stage.

Taking the transition from Stage-1 to Stage-2 as an example, the coefficient stored at address 4 is selected as the critical case. In Stage-1, this coefficient is accessed as the $\frac{N}{8}$ -th read element, while in Stage-2, the same coefficient is accessed as the $\left(\frac{N}{4} - 1\right)$ -th read element. Therefore, the minimum interval between the start of Stage-1 and Stage-2 can be obtained as:

$$\left(\frac{N}{4} - 1\right) - \frac{N}{8} = \frac{N}{8} - 1 \quad (9)$$

cycles.

To guarantee that Stage-2 reads the corresponding operand only after it has been completely generated and written back by Stage-1, the inter-stage delay should satisfy:

$$Delay_{min} > \frac{N}{8} - 1 \quad (10)$$

cycles.

With this delay constraint, Stage-2 starts accessing its required butterfly operands only after the corresponding data generated by Stage-1 have completed the read-compute-write process and have been stored into the opposite BRAM. Meanwhile, the remaining butterfly operations of Stage-1 access different coefficient locations, which do not overlap with the read operations of Stage-2. Therefore, the proposed scheduling mechanism avoids read-after-write (RAW) conflicts while enabling partial overlap between adjacent stages.

The same scheduling principle can be generalized to different PE configurations. For an architecture with a total of PE processing elements, the number of read operations required by each stage is reduced to:

$$Latency_{stage} = \frac{N}{2 \times PE} \quad (11)$$

cycles.

Therefore, the minimum inter-stage delay can be expressed as:

$$Delay_{min} > \frac{N}{2 \times PE} - 1 \quad (12)$$

cycles.

By applying this delay constraint, the following stage always accesses valid intermediate results after they have been completely generated and stored by the previous stage. Meanwhile, the coefficient ordering remains unchanged among different NTT stages, and the regular memory access pattern is preserved. Consequently, the proposed EVEN/ODD ping-pong scheduling combined with PE-pair execution enables conflict-free inter-stage pipelining without additional conflict detection or dynamic memory scheduling logic.

Therefore, increasing PE parallelism changes the stage issue time and the precomputed inter-stage offset, but not the coefficient dependency relation or access order. Within the stated memory-port limit, the EVEN/ODD scheduling and PE-pair organization maintain conflict-free inter-stage pipelining without additional data-reordering or dynamic conflict-resolution logic.

The proposed memory organization supports both Kyber and Dilithium operations through a unified coefficient mapping strategy. As described in Section 3.2, the proposed PE Datapath can be configured to process 24-bit input data under different arithmetic modes. Therefore, the same memory interface can support both Dilithium and Kyber computations without introducing additional data conversion units or extra storage resources.

Each RAM address has a fixed width of 24 bits and stores either one Dilithium coefficient or two concatenated 12-bit Kyber coefficients. In Dilithium mode, the word is interpreted as one coefficient. In Kyber mode, the upper and lower 12-bit fields are directly connected to the two Kyber arithmetic lanes. The corresponding 24-bit output is written back in the same packed format. These operations are static bus partitioning and concatenation rather than run-time arithmetic packing or unpacking.

The same mapping is applied to twiddle-factor storage. Algorithm selection changes only the interpretation of the 24-bit word and the existing PE input multiplexers; it does not introduce an independent unpacking network, an extra pipeline stage, or additional BRAM. The Split block used in Kyber PWM is likewise a fixed bit slice of the packed result bus.

As illustrated in Figure 4, Kyber and Dilithium therefore share the same BRAM interface, address generator, and stage scheduler. The incremental routing consists only of fixed 12-bit field connections and the mode-selection paths already required by the configurable PE, avoiding a separate memory Datapath. Because one 24-bit word stores two Kyber coefficients but only one Dilithium coefficient, the effective memory depth for a Kyber transform is one half of that for Dilithium under the same mapping. Consequently, a Kyber configuration requires only half of the active PE parallelism to obtain a comparable transform latency. This packed mapping allows the architecture to scale between the two schemes without changing the memory organization.

Similarly, the proposed architecture also supports both Dilithium and Kyber PWM operations by reusing the same PE resources and memory interface. Different from NTT/INTT computation, PWM operations do not require stage-based data scheduling. The required

polynomial coefficients are pre-stored in RAM and are sequentially read according to their addresses. A pair of PEs is used to perform point-wise multiplication, where each PE receives the corresponding coefficient pair and completes the required modular multiplication operation. For Dilithium PWM, the PE pair directly processes the coefficient pairs according to the standard point-wise multiplication operation. For Kyber PWM, the stored coefficients are read in the same sequential manner, and the PE Datapath switches to the Kyber-specific multiplication mode to complete the corresponding point-wise multiplication. Therefore, Kyber and Dilithium PWM operations share the same RAM access pattern and PE organization, while only the internal arithmetic operation mode is configured according to the selected algorithm.

By reusing the existing PE resources and sequential coefficient-access mechanism, the proposed architecture avoids introducing a separate complete PWM Datapath or independent multiplier array, further enhancing the resource efficiency of the unified NTT/INTT/PWM architecture.

4. Experimental Results and Comparisons

The proposed unified NTT/INTT/PWM architecture was implemented on a Xilinx Artix-7 FPGA device using Vivado 2020.2. The experimental evaluation was conducted in terms of resource utilization, execution latency, area-time product (ATP), and power consumption. Three configurations, $PE = 1 \times 2$, $PE = 2 \times 2$, and $PE = 4 \times 2$, corresponding to 2, 4, and 8 total PEs, respectively, were evaluated to investigate scalability and performance trade-offs. The area metric is calculated using the commonly adopted FPGA resource normalization method:

$$Area = LUT + DSP \times 100 + BRAM \times 300 \quad (13)$$

where DSP and BRAM resources are converted into LUT-equivalent values according to previous FPGA accelerator designs [7,16]. To keep the tabulated data internally reproducible, ATP values in Tables 2–4 are calculated from the displayed normalized area and displayed latency and rounded to the nearest integer. To ensure fair evaluation in Sections 4.2 and 4.3, all percentage improvements are calculated only against implementations with the same total PE count. Entries with unmatched or unspecified PE counts are retained as supplementary reference points but are not used to derive the reported improvement percentages. When no exact PE-matched baseline is available, the corresponding result is used only to demonstrate the scalability of the proposed architecture.

4.1. Modular Multiplication Evaluation

Table 1 compares the proposed modular multiplication unit with previous Kyber- and Dilithium-oriented implementations. The proposed unit uses 222 LUTs, 266 FFs, and 2 DSPs without BRAM. Through the unified 24-bit Datapath, the same two DSPs perform either two parallel Kyber modular multiplications or one Dilithium modular multiplication, providing cross-scheme compatibility and higher Kyber parallelism within one configurable arithmetic structure.

The Dilithium-specific designs in [7,13] use fewer LUTs, but they support only one Dilithium multiplication and cannot directly execute Kyber arithmetic. Conversely, the Kyber-specific designs in [22–24] use one DSP and fewer LUTs because they are optimized for one modulus and one multiplication lane; they support neither Dilithium nor two parallel Kyber modular multiplications. Their lower resource use is therefore associated with narrower functional coverage and lower arithmetic parallelism.

Among the compared implementations, ref. [19] provides the same functional capability as the proposed design: two parallel Kyber multiplications or one Dilithium

multiplication using 2 DSPs and no BRAM. However, ref. [19] requires 369 LUTs, whereas the proposed unit uses 222 LUTs, corresponding to a 39.8% LUT reduction. The improvement results from sharing the DSP, shift, addition/subtraction, K-RED, and correction resources across the two arithmetic modes. The FF count of [19] is not reported, so an FF comparison is not made.

Table 1. Comparison of modular multiplication results with prior works. A check mark (✓) indicates supported functionality, a cross (×) indicates unsupported functionality, and $\times 1/\times 2$ denotes the number of parallel multiplication lanes.

Design	LUT	FF	DSP	BRAM	Kyber	Dilithium
[7]	196	146	2	0	×	✓ (×1)
[13]	152	170	2	0	×	✓ (×1)
[22]	135	96	1	0	✓ (×1)	×
[23]	127	0	1	0	✓ (×1)	×
[24]	175	82	1	0	✓ (×1)	×
[19]	369	Not reported	2	0	✓ (×2)	✓ (×1)
Our	222	266	2	0	✓ (×2)	✓ (×1)

Although the proposed unit uses more resources than dedicated single-scheme multipliers, it achieves a more favorable balance among hardware cost, cross-scheme compatibility, and arithmetic parallelism. In particular, its resource advantage is evaluated against [19], which offers the same Kyber $\times 2$ /Dilithium $\times 1$ capability, rather than against functionally narrower designs.

The same configurable modular multiplier is directly instantiated in the NTT/INTT/PWM PE array, avoiding separate Kyber- and Dilithium-specific multiplier structures and providing an efficient arithmetic foundation for the unified architecture.

4.2. Dilithium NTT/INTT Evaluation

Table 2 presents the comparison results of the proposed Dilithium NTT/INTT architecture with previous works. The results are discussed in iso-PE groups, where PE = 1×2 , PE = 2×2 , and PE = 4×2 of the proposed architecture are compared with baselines using 2, 4, and 8 PEs, respectively. This grouping separates architectural efficiency from the performance gain obtained solely by increasing the number of PEs.

Table 2. Comparison of Dilithium NTT/INTT with prior works.

Work	Device	PE	LUT	FF	DSP	BRAM	Freq. (MHz)	Cycles	Latency (μs)	Area ^b	ATP ^c
[17]	Artix-7	–	755	844	2	0	237	1058	4.46	955	4260
[15] ^a	Virtex-7	1	1.2K	640	4	0	246	1034	4.20	1600	6720
[11] ^a	Virtex-7	1	691	451	5	3	187	1031	5.51	2091	11,521
[19] ^a	Artix-7	2	3.1K	2.4K	4	4.5	200	512	2.56	4850	12,416
Our ^a	Artix-7	1×2	1.1K	1K	5	2.5	336	640	1.91	2350	4489
[19] ^a	Artix-7	2×2	6.2K	3.6K	8	5.5	165	256	1.55	8650	13,407
[10] ^a	Artix-7	4	3.1K	2.5K	28	8	125	328	2.62	8300	21,746
[20]	Artix-7	4	4.5K	3.1K	8	1	250	300	1.20	5600	6720
Our ^a	Artix-7	2×2	1.9K	1.7K	10	4.5	336	352	1.05	4250	4463
[6] ^a	Artix-7	8	4.2K	4.4K	18	0	297	286	0.96	6000	5760
[16] ^a	Virtex-7	8	>2.8K	–	>24	20	329	151	0.46	>11,200	>5152
Our ^a	Artix-7	4×2	2.8K	2.8K	20	10	336	208	0.62	7800	4836

a: This work supports multiple degrees of parallel NTT operations (Dilithium). b: Area = #LUTs + #DSPs $\times$ 100 + #BRAMs $\times$ 300 [7,16]. c: The area-time product (ATP) is calculated as area $\times$ latency. Note: Percentage improvements in Section 4.2 are calculated only against implementations with the same total PE count; entries with unmatched or unspecified PE counts are included for reference only.

For the 2-PE comparison, the proposed PE = 1×2 configuration operates at 336 MHz and completes Dilithium NTT/INTT in 640 cycles, corresponding to a latency of 1.91 μ s, a normalized area of 2350, and an ATP of 4489. Compared with the 2-PE implementation in [19], the proposed design reduces latency, normalized area, and ATP by 25.4%, 51.5%, and 63.8%, respectively. The PE count in [17] is not specified, while [11,15] use one PE; these entries are therefore retained only as supplementary references and are excluded from the iso-PE percentage comparison.

For the 4-PE comparison, the proposed PE = 2×2 configuration achieves a latency of 1.05 μ s and an ATP of 4463. Compared with the 4-PE implementations in [10,19,20], the ATP is reduced by 66.7%, 79.5%, and 33.6%, respectively. For the 8-PE comparison, the proposed PE = 4×2 configuration reduces the latency to 0.62 μ s with an ATP of 4836. Compared with the 8-PE implementation in [6], the ATP is reduced by 16.0%; it is also at least 6.1% lower than the lower-bound ATP reported for the 8-PE implementation in [16].

These iso-PE results show that the reported improvements are not obtained by comparing the highest-parallelism configuration with lower-PE baselines. Instead, they arise from the EVEN/ODD ping-pong schedule, PE-pair organization, and unified arithmetic Datapath. Increasing the configuration from 2 to 4 and 8 PEs progressively reduces latency while preserving the same memory organization and control framework, demonstrating a scalable area-throughput trade-off.

4.3. Kyber NTT/INTT Evaluation

Table 3 summarizes the comparison results of the proposed Kyber NTT/INTT architecture. Different from Dilithium, Kyber uses 12-bit coefficients. The proposed architecture reuses the same 24-bit memory organization by storing two Kyber coefficients in one memory word. Therefore, Kyber computation directly reuses the existing address generation, memory access, and PE scheduling mechanisms without introducing an independent memory structure. As in Table 2, all quantitative improvement claims are made only between designs with the same total PE count.

Table 3. Comparison of Kyber NTT/INTT with prior works.

Work	Device	PE	LUT	FF	DSP	BRAM	Freq. (MHz)	Cycles	Latency (μ s)	Area ^b	ATP ^c
[22]	Artix-7	2	1.7K	1.1K	2	3	161	448	2.8	2800	7840
[24]	Artix-7	1	503	545	1	2	200	1029/1285	5.14	1203	6183
[19] ^a	Artix-7	2	3.1K	2.4K	4	4.5	200	224	1.12	4850	5432
Our ^a	Artix-7	1×2	1.1K	1K	5	2.5	336	292	0.87	2350	2045
[23] ^a	Virtex-7	4	2.5K	1.2K	4	0	166	235	1.4	2925	4095
[19] ^a	Artix-7	2×2	6.2K	3.5K	8	5.5	165	112	0.68	8650	5882
Our ^a	Artix-7	2×2	1.9K	1.7K	10	4.5	336	164	0.49	4250	2083
[26] ^a	Artix-7	7	4.8K	4.6K	0	1	250	277	1.11	5100	5661
Our ^a	Artix-7	4×2	2.8K	2.8K	20	10	336	100	0.30	7800	2340

a: This work supports multiple degrees of parallel NTT operations (Kyber). b: Area = #LUTs + #DSPs $\times$ 100 + #BRAMs $\times$ 300 [7,16]. c: The area-time product (ATP) is calculated as area $\times$ latency. Note: Percentage improvements in Section 4.3 are calculated only against implementations with the same total PE count; entries with unmatched PE counts are included for reference only.

For the 2-PE comparison, the proposed PE = 1×2 configuration operates at 336 MHz and completes NTT/INTT in 292 cycles, corresponding to a latency of 0.87 μ s and an ATP of 2045. Compared with the 2-PE implementations in [22] and [19], the latency is reduced by 68.9% and 22.3%, respectively, while the ATP is reduced by 73.9% and 62.4%. The single-PE implementation in [24] is retained only as a supplementary reference and is not included in the iso-PE percentage comparison.

Within the matched 2-PE group, ref. [19] achieves fewer execution cycles, but its lower operating frequency and larger normalized area result in a higher ATP. The proposed design therefore achieves improved area-time efficiency through both high-frequency implementation and efficient memory organization.

For the 4-PE comparison, the proposed PE = 2×2 configuration achieves a latency of 0.49 μs and an ATP of 2083. Compared with the 4-PE implementations in [23] and [19], the latency is reduced by 65.0% and 27.9%, respectively, while the ATP is reduced by 49.1% and 64.6%. The PE = 4×2 configuration further reduces latency to 0.30 μs with an ATP of 2340. Because Table 3 contains no exact 8-PE baseline, this result is used only to demonstrate scalability and is not included in the percentage comparison; the 7-PE result in [26] is retained as a supplementary reference. Across the matched 2-PE and 4-PE comparisons, the proposed architecture achieves ATP reductions ranging from 49.1% to 73.9%.

4.4. PWM Evaluation

Table 4 compares the proposed PWM architecture with previous implementations. For PE = 1×2 , the proposed architecture achieves a normalized area of 2350, a latency of 0.41 μs , and an ATP of 964. Compared with [22], the normalized area and latency are reduced by 16.1% and 74.2%, respectively, resulting in a 78.3% ATP reduction. Compared with the high-speed polynomial multiplication accelerator in [12], the proposed architecture achieves a 53.9% reduction in normalized area and an 18.0% reduction in latency, resulting in a 62.2% reduction in ATP. Across the other implementations listed in Table 4, the ATP reduction ranges from 57.0% to 78.3%.

Table 4. Comparison of Kyber/Dilithium PWM with prior works.

Work	Device	PE	Cycles	Mode	Freq. (MHz)	Latency (μs)	Area ^b	ATP ^c
[22]	Artix-7	2	256	K	161	1.59	2800	4452
[12]	Artix-7	2×2	169	K	334	0.50	5102	2551
[28]	Artix-7	4	67	D	100	0.67	3350	2244
[21]	Artix-7	1×2	135	D	100	1.35	3004	4055
[19]	Artix-7	2×2	64	K/D	165	0.39	8650	3373
[19]	Artix-7	2	128	K/D	200	0.64	4850	3104
Our	Artix-7	1×2	137	K/D	336	0.41	2350	964
Our	Artix-7	2×2	73	K/D	336	0.22	4250	935
Our	Artix-7	4×2	41	K/D	336	0.12	7800	936

b: Area = #LUTs + #DSPs $\times$ 100 + #BRAMs $\times$ 300 [7,16]. c: The area-time product (ATP) is calculated as area $\times$ latency.

Although [19,22] support PWM through configurable BFUs, Kyber base multiplication is divided into PWM0 and PWM1 stages, so different subsets of the multiplier and adder resources are active in different cycles and additional operand-selection and control paths are required. Increasing throughput generally replicates complete BFUs, including arithmetic, registers, and routing resources that are not all required by PWM. In [12], the dual-butterfly structure reuses the RNS modular multiplier, but butterfly-specific units and part of the routing and pipeline remain inactive during PWM; replicating the complete unit to raise throughput also replicates these resources. In [21], PWM processes two coefficients per cycle whereas NTT/INTT processes four, and multi-mode support requires additional multiplexers and pipeline-balancing registers. Thus, these designs provide functional reuse, but the effective utilization of the complete PE Datapath remains limited.

In the proposed architecture, PWM is not implemented by constructing another complete arithmetic Datapath. Instead, the PWM operation is directly mapped onto the existing NTT/INTT PE array. The KD_ModMult and KD_ModAdd units originally employed by

the CT and GS butterflies are reused as the main arithmetic operators for PWM, while KD_ModSub remains inactive during PWM execution. The existing coefficient-memory organization, PE input/output interfaces, pipeline structure, and control framework are also retained. Therefore, most of the hardware required for PWM is already present in the NTT/INTT Datapath.

For Dilithium, PWM only requires coefficient-wise modular multiplication. Hence, it is performed entirely by the existing KD_ModMult units, without adding any PWM-specific arithmetic unit. For Kyber, the incomplete NTT results in a more complex base multiplication that cannot be completed using only the original butterfly multiplier. Consequently, only one auxiliary 12-bit modular multiplier is introduced in each PE group. The existing modular multiplier, modular adders, modular subtractors, reduction logic, memory system, and control path remain shared. Thus, the proposed design minimizes the incremental hardware required to extend an NTT/INTT PE into an NTT/INTT/PWM PE.

This extensive PE-resource reuse explains why the proposed PE = 1×2 configuration simultaneously achieves a smaller area and a competitive latency. In contrast to architectures that incorporate a more complete PWM-oriented dataflow into each processing unit, the proposed architecture retains almost the entire original NTT/INTT PE and introduces only the arithmetic resource that is strictly required by Kyber base multiplication. As a result, the proposed architecture reduces the normalized area while preserving the parallel execution of the dominant PWM and addition operations, thereby achieving the substantial ATP improvement reported in Table 4.

Increasing the configuration from PE = 1×2 to PE = 2×2 and PE = 4×2 reduces the PWM latency from 0.41 μ s to 0.22 μ s and 0.12 μ s, respectively, while the ATP remains nearly constant at 964, 935, and 936. This scalability is achieved because every additional PE is reused by NTT, INTT, and PWM rather than being added exclusively for PWM. Therefore, the increased area is proportionally converted into higher PWM throughput without duplicating a separate PWM accelerator.

Overall, the advantage of the proposed architecture lies in the high reuse ratio of the existing PE resources. Except for the auxiliary 12-bit modular multiplier required by Kyber base multiplication, PWM reuses the dominant arithmetic, storage, pipeline, and control resources of the NTT/INTT PE array. This minimizes the area increment required to support PWM while preserving parallel execution, thereby improving both area efficiency and ATP.

4.5. Power Consumption Analysis

In addition to area and performance, Table 5 reports the estimated total on-chip power of the proposed unified NTT/INTT/PWM architecture for different PE configurations. The values were obtained using Vivado power analysis for the target Artix-7 device at 336 MHz and include both static and dynamic contributions. Separate component values were not retained in the archived reports; therefore, only the directly available total-power estimates are presented, and no unsupported static/dynamic split is inferred.

Table 5. Power consumption of the proposed unified NTT/INTT/PWM architecture under different PE configurations.

Work	Device	PE	Freq. (MHz)	Total Power
Our	Artix-7	1×2	336	92 mW
Our	Artix-7	2×2	336	104 mW
Our	Artix-7	4×2	336	141 mW

The estimated total power is 92 mW, 104 mW, and 141 mW for $PE = 1 \times 2$, $PE = 2 \times 2$, and $PE = 4 \times 2$, respectively. Relative to $PE = 1 \times 2$, total power increases by 13.0% and 53.3% for the two higher-parallelism configurations.

The measured trend is consistent with the larger number of active processing, routing, and memory resources at higher parallelism. Increasing the PE count reduces execution latency but raises total power, providing a configurable throughput–power trade-off. The table is therefore used to compare the proposed configurations with one another rather than to claim a component-level power breakdown or a cross-paper power advantage.

5. Conclusions

This paper presents a unified and scalable NTT/INTT/PWM hardware architecture for CRYSTALS-Kyber and CRYSTALS-Dilithium. To overcome the insufficient hardware reuse among different parameter sets and polynomial operations in existing post-quantum cryptographic accelerators, a configurable two-dimensional PE array combined with an EVEN/ODD ping-pong memory organization is proposed. By exploiting deterministic coefficient access patterns and precomputed inter-stage delay constraints, the proposed memory scheduling mechanism achieves conflict-free data access under different PE configurations and enables partial overlap between adjacent NTT/INTT computation stages without additional data-reordering units, dynamic conflict detection logic, or complicated memory control schemes.

To support both CRYSTALS-Kyber and CRYSTALS-Dilithium, this work further proposes a unified coefficient mapping strategy and shared modular arithmetic datapath. With the proposed 24-bit RAM organization, each memory address can store either one Dilithium coefficient or two packed Kyber coefficients, allowing both schemes to share the same memory interface, address generation logic, and computation datapath without additional conversion and storage overhead. Based on the K-RED-based hardware optimization flow and LUT-assisted correction mechanism, the proposed unified modular multiplier uses the same two DSP resources to perform one wide-word Dilithium modular multiplication or two parallel Kyber modular multiplications. Experimental results demonstrate that the proposed multiplier requires only 222 LUTs, 266 FFs, and 2 DSPs, reducing LUT utilization by 39.8% compared with previous unified $Kyber \times 2$ /Dilithium $\times 1$ modular multiplier designs.

Furthermore, this work introduces a unified PWM computation approach without an additional arithmetic array. Unlike conventional designs that employ an independent PWM datapath, the proposed architecture primarily reuses the existing KD_ModMult and KD_ModAdd units inside the NTT/INTT PE, while KD_ModSub remains inactive during PWM execution. For Dilithium PWM, the existing modular multiplier directly performs coefficient-wise multiplication. For Kyber PWM, the multiplication and addition datapaths of two PEs are reused, and only one auxiliary 12-bit K_ModMult is introduced to complete the base multiplication. Therefore, the proposed design supports unified NTT, INTT, and PWM operations while minimizing additional hardware overhead.

The proposed architecture was implemented on a Xilinx Artix-7 FPGA platform using the Vivado 2020.2 design flow. Three configurations, $PE = 1 \times 2$, $PE = 2 \times 2$, and $PE = 4 \times 2$, corresponding to 2, 4, and 8 PEs, respectively, were evaluated to verify scalability. Under matched PE configurations, the proposed architecture achieves maximum ATP reductions of 79.5% for Dilithium NTT/INTT and 73.9% for Kyber NTT/INTT; the Kyber/Dilithium PWM mode achieves an ATP reduction of up to 78.3%. For Dilithium NTT/INTT, increasing the PE count from 2 to 4 and 8 reduces latency from 1.91 μ s to 1.05 μ s and 0.62 μ s, with ATP values of 4489, 4463, and 4836. For Kyber NTT/INTT, the corresponding latency decreases from 0.87 μ s to 0.49 μ s and 0.30 μ s, with ATP values of 2045, 2083, and 2340. For PWM,

latency decreases from 0.41 μ s to 0.22 μ s and 0.12 μ s, while ATP remains nearly constant at 964, 935, and 936, demonstrating that additional PE resources are effectively converted into throughput improvement.

The power analysis further verifies the configurable performance–power trade-off of the proposed architecture. As PE parallelism increases, the computation latency decreases, while the total power consumption increases from 92 mW to 141 mW. Therefore, different PE configurations provide flexible design points for applications requiring different throughput, area, and power constraints.

To further evaluate the cross-platform scalability of the proposed architecture, future work will conduct synthesis and implementation experiments on additional advanced FPGA platforms, such as Xilinx Artix UltraScale+ (e.g., XCAU10P), Kintex UltraScale (e.g., XCKU040), and Virtex UltraScale+ (e.g., XCVU3P) devices. By comparing LUT/FF/BRAM/DSP utilization, maximum operating frequency, power consumption, and PE scaling trends across different FPGA platforms, the proposed architecture can be further validated as a device-independent design. The scalability mainly benefits from the regular PE array organization, unified modular arithmetic datapath, and deterministic memory access mechanism. Such cross-platform evaluations will further demonstrate the portability and deployment potential of the proposed architecture for future high-performance post-quantum cryptographic accelerators.

Overall, this work provides a highly reusable, configurable, and scalable hardware acceleration framework for lattice-based post-quantum cryptography. Through conflict-free EVEN/ODD memory scheduling, two-dimensional PE parallelism, unified Kyber/Dilithium modular multiplication, and extensive NTT/INTT/PWM resource sharing, the proposed architecture efficiently supports multiple polynomial arithmetic operations while achieving competitive area-time efficiency, configurable throughput–power trade-offs, and promising cross-FPGA scalability. Additional supporting information is provided in Supplementary File S1.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/electronics15163495/s1>, Supplementary File S1: Additional supporting information for the proposed NTT/INTT/PWM architecture.

Author Contributions: Conceptualization, Y.W. and C.Y.; methodology, Y.W. and C.Y.; architecture design, Y.W.; validation, Y.W. and C.Y.; formal analysis, Y.W., C.Y. and X.W.; investigation, Y.W., C.Y. and S.X.; data curation, Y.W. and X.W.; writing—original draft preparation, Y.W. and K.M.; writing—review and editing, Y.W., C.Y., X.W., S.X., K.M. and M.L.; visualization, Y.W.; supervision, K.M. and M.L.; project administration, K.M. and M.L.; resources, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 52474270. The article processing charge (APC) was funded by the Institute of Semiconductors, Chinese Academy of Sciences.

Data Availability Statement: The data supporting the findings of this study are contained within the article and its Supplementary Materials.

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

References

1. Bos, J.W.; Costello, C.; Naehrig, M.; Stebila, D. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 553–570. [CrossRef]

2. Chib, S.; Devarajan, H.R.; Chundru, S.; Pulivarthy, P.; Isaac, R.A.; Oku, K.M. Standardized Post-Quantum Cryptography and Recent Developments in Quantum Computers. In Proceedings of the 2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT), Bhimtal, Nainital, India, 21–22 February 2025; pp. 1018–1023. [\[CrossRef\]](#)
3. Bisheh-Niasar, M.; Azarderakhsh, R.; Mozaffari-Kermani, M. Instruction-Set Accelerated Implementation of CRYSTALS-Kyber. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 4648–4659. [\[CrossRef\]](#)
4. Wang, T.; Zhang, C.; Cao, P.; Gu, D. Efficient Implementation of Dilithium Signature Scheme on FPGA SoC Platform. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 1158–1171. [\[CrossRef\]](#)
5. Javeed, K.; Gregg, D. CUBA: A Configurable Unified Butterfly Architecture for Lattice-Based Cryptosystems. *IEEE Embed. Syst. Lett.* **2026**, *18*, 128–131. [\[CrossRef\]](#)
6. Cui, Y.; Zhong, J.; Wang, B.; Xu, T.; Wang, C.; Liu, W. High-Performance Hardware Implementation of CRYSTALS-Dilithium Based on Improved MDC-NTT. *IEEE Trans. Comput.* **2025**, *74*, 2896–2908. [\[CrossRef\]](#)
7. Ye, Z.; Cheung, R.C.C.; Huang, K. PipeNTT: A Pipelined Number Theoretic Transform Architecture. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 4068–4072. [\[CrossRef\]](#)
8. Li, X.; Lu, J.; Liu, D.; Li, A.; Yang, S.; Huang, T. A High Speed Post-Quantum Crypto-Processor for Crystals-Dilithium. *IEEE Trans. Circuits Syst. II Express Briefs* **2024**, *71*, 435–439. [\[CrossRef\]](#)
9. Li, M.; Tian, J.; Hu, X.; Wang, Z. Reconfigurable and High-Efficiency Polynomial Multiplication Accelerator for CRYSTALS-Kyber. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 2540–2551. [\[CrossRef\]](#)
10. Mert, A.C.; Karabulut, E.; Öztürk, E.; Savaş, E.; Aysu, A. An Extensive Study of Flexible Design Methods for the Number Theoretic Transform. *IEEE Trans. Comput.* **2022**, *71*, 2829–2843. [\[CrossRef\]](#)
11. Mu, J.; Ren, Y.; Wang, W.; Hu, Y.; Chen, S.; Chang, C.-H.; Fan, J.; Ye, J.; Cao, Y.; Li, H.; et al. Scalable and Conflict-Free NTT Hardware Accelerator Design: Methodology, Proof, and Implementation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 1504–1517. [\[CrossRef\]](#)
12. Alhassani, A.; Benaissa, M. High-Speed Polynomials Multiplication HW Accelerator for CRYSTALS-Kyber. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2024**, *71*, 6105–6113. [\[CrossRef\]](#)
13. Syafalni, I.; Teffandi, N.; Ibrahim, F.; Pramudita, I.; Sutisna, N.; Adiono, T. Reconfigurable Radix-2/4/8 of Unified 2^N Points NTT-INTT for Homomorphic Encryption. In Proceedings of the 2025 IEEE International Symposium on Circuits and Systems (ISCAS), London, UK, 25–28 May 2025; pp. 1–5. [\[CrossRef\]](#)
14. Su, Y.; Yang, B.-L.; Yang, C.; Yang, Z.-P.; Liu, Y.-W. A Highly Unified Reconfigurable Multicore Architecture to Speed Up NTT/INTT for Homomorphic Polynomial Multiplication. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 993–1006. [\[CrossRef\]](#)
15. Geng, Y.; Hu, X.; Li, M.; Wang, Z. Rethinking Parallel Memory Access Pattern in Number Theoretic Transform Design. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 1689–1693. [\[CrossRef\]](#)
16. Liu, S.-H.; Kuo, C.-Y.; Mo, Y.-N.; Su, T. An Area-Efficient, Conflict-Free, and Configurable Architecture for Accelerating NTT/INTT. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2024**, *32*, 519–529. [\[CrossRef\]](#)
17. Zhong, J.; Wang, B.; Zhu, Z.; Liu, W.; Cui, Y. A Lightweight and Efficient BRAM-Free NTT Unit for Crystals-Dilithium. In Proceedings of the 2025 IEEE International Symposium on Circuits and Systems (ISCAS), London, UK, 25–28 May 2025; pp. 1–5. [\[CrossRef\]](#)
18. Nguyen, T.-H.; Binh, N.T.; Nghi, H.P.; Pham, C.-K.; Hoang, T.-T. Unified-Pipelined NTT Architecture for Polynomial Multiplication in Lattice-Based Cryptosystems. In Proceedings of the 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, 19–22 May 2024; pp. 1–5. [\[CrossRef\]](#)
19. Mandal, S.; Basu Roy, D. KiD: A Hardware Design Framework Targeting Unified NTT Multiplication for CRYSTALS-Kyber and CRYSTALS-Dilithium on FPGA. In Proceedings of the 2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID), Kolkata, India, 6–10 January 2024; pp. 455–460. [\[CrossRef\]](#)
20. Beckwith, L.; Nguyen, D.T.; Gaj, K. High-Performance Hardware Implementation of CRYSTALS-Dilithium. In Proceedings of the 2021 International Conference on Field-Programmable Technology (ICFPT), Auckland, New Zealand, 6–10 December 2021; pp. 1–10. [\[CrossRef\]](#)
21. Kamel, D.; Standaert, F.-X. Area Efficient Polynomial Arithmetic Accelerator for Post-Quantum Digital Signatures and KEMs. In *Constructive Approaches for Security Analysis and Design of Embedded Systems*; Rivain, M., Sasdrich, P., Eds.; Springer: Cham, Switzerland, 2025; Volume 15952, pp. 159–184. [\[CrossRef\]](#)
22. Xing, Y.; Li, S. A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-Kyber on FPGA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 328–356. [\[CrossRef\]](#)
23. Ramyavani, P.; Reddy, R.T.; Casheekar, A.; Panda, A.K.; Geethakumari, G.; Sahoo, S.K. UPP-NTT: A Unified Parallel Pipelined NTT Accelerator for CRYSTALS-Kyber Post-Quantum Cryptography. *IEEE Access* **2026**, *14*, 51093–51107. [\[CrossRef\]](#)

24. Sonbul, O.S.; Rashid, M.; Masud, M.I.; Aman, M.; Jaffar, A.Y. Deeply Pipelined NTT Accelerator with Ping-Pong Memory and LUT-Only Barrett Reduction for Post-Quantum Cryptography. *Electronics* **2026**, *15*, 513. [\[CrossRef\]](#)
25. Tsai, Y.-C.; Lin, Y.-H.; Hwang, W.-J. An Open Hardware ML-KEM Polynomial Ring Accelerator on Chipyard RISC-V SoC: System-Level Integration and Evaluation. *Electronics* **2026**, *15*, 2511. [\[CrossRef\]](#)
26. Nguyen, T.-H.; Kieu-Do-Nguyen, B.; Pham, C.-K.; Hoang, T.-T. High-Speed NTT Accelerator for CRYSTAL-Kyber and CRYSTAL-Dilithium. *IEEE Access* **2024**, *12*, 34918–34930. [\[CrossRef\]](#)
27. Gao, X.; Li, Y.; Li, T.; Li, X.; Wang, J. A High-Throughput, BRAM-Efficient NTT/INTT Accelerator for ML-KEM. *Electronics* **2025**, *14*, 4868. [\[CrossRef\]](#)
28. Wang, T.; Zhang, C.; Zhang, X.; Gu, D.; Cao, P. Optimized Hardware–Software Co-Design for Kyber and Dilithium on RISC-V SoC FPGA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2024**, *2024*, 99–135. [\[CrossRef\]](#)
29. Tan, W.; Lao, Y.; Parhi, K.K. KyberMat: Efficient Accelerator for Matrix–Vector Polynomial Multiplication in CRYSTALS-Kyber Scheme via NTT and Polyphase Decomposition. In Proceedings of the 2023 42nd IEEE/ACM International Conference on Computer Aided Design (ICCAD), San Francisco, CA, USA, 28 October–2 November 2023; pp. 1–9. [\[CrossRef\]](#)
30. Sun, J.; Bai, X. A High-Speed Hardware Architecture of an NTT Accelerator for CRYSTALS-Kyber. *Integr. Circuits Syst.* **2024**, *1*, 92–102. [\[CrossRef\]](#)
31. Hong, J.; Zhang, B.; Mao, G.; Hung, P.S.Y.; Cheung, R.C.C. Systematic HLS Co-Design: Achieving Scalable and Fully-Pipelined NTT Acceleration on FPGAs. *Electronics* **2025**, *14*, 3922. [\[CrossRef\]](#)
32. Longa, P.; Naehrig, M. Speeding Up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography. In *Cryptology and Network Security*; Foresti, S., Persiano, G., Eds.; Springer: Cham, Switzerland, 2016; Volume 10052, pp. 124–139. [\[CrossRef\]](#)
33. Barrett, P. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In *Advances in Cryptology—CRYPTO ’86*; Odlyzko, A.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1987; Volume 263, pp. 311–323. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Montgomery, P.L. Modular Multiplication without Trial Division. *Math. Comput.* **1985**, *44*, 519–521. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.