

Article

Area–Power–Performance Trade-Offs in Lightweight Non-Pipelined and Pipelined NTT Accelerators for CRYSTALS-Dilithium

Omar S. Sonbul , Muhammad Rashid * , Khalid M. Khayyat  and Hamza H. Sulimani 

Computer and Network Engineering Department, Umm Al-Qura University, Makkah 24382, Saudi Arabia; ossonbul@uqu.edu.sa (O.S.S.); kmkhayyat@uqu.edu.sa (K.M.K.); hhhsulimani@uqu.edu.sa (H.H.S.)

* Correspondence: mfelahi@uqu.edu.sa

Abstract

Number Theoretic Transform (NTT)-based polynomial multiplication is a computationally intensive operation in lattice-based post-quantum cryptography (PQC) schemes such as CRYSTALS-Dilithium. Existing hardware accelerators optimize area and timing performance, without focusing on evaluating trade-offs among hardware utilization, execution latency, operating frequency, and power consumption. This article investigates such trade-offs through two lightweight field-programmable gate array (FPGA) implementations of an iterative NTT-based polynomial multiplication accelerator, namely non-pipelined and 4-stage pipelined architectures. Both implementations employ a single butterfly unit based on Cooley–Tukey and Gentleman–Sande configurations to compute the forward NTT (FNTT), inverse NTT (INTT), and coefficient-wise multiplication (CWM). The 4-stage pipelined architecture employs pipeline registers in the modular multiplication and Barrett reduction datapaths to maximize the operating frequency. Both architectures are implemented on an Artix-7 FPGA and evaluated across operating frequencies ranging from 10 MHz to 280 MHz. The results show that the non-pipelined architecture provides reduced hardware overhead and lower power consumption, whereas the pipelined architecture improves timing scalability and successfully operates at 280 MHz. At the maximum operating frequency, the pipelined implementation utilizes 1115 slices and achieves execution times of 4.58 μ s, 0.93 μ s, and 4.58 μ s for FNTT, CWM, and INTT computations, respectively, with an average power consumption of 133 mW. The Area–Time Product (ATP) and Energy–Delay Product (EDP) evaluations demonstrate that the pipelined architecture achieves improved overall efficiency within the proposed lightweight single-butterfly-based polynomial multiplication architecture at higher operating frequencies, obtaining an ATP of 7.74×10^3 Slice- μ s and EDP of 1501.52 nJ- μ s.



Academic Editors: Shuwen Deng and Kun Yang

Received: 30 May 2026

Revised: 6 July 2026

Accepted: 10 July 2026

Published: 13 July 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.

Keywords: Number Theoretic Transform (NTT); Post-Quantum Cryptography (PQC); polynomial multiplication; FNTT; INTT; CWM; FPGA

1. Introduction

Lattice-based post-quantum cryptography (PQC) has emerged as a leading direction for securing digital systems against the threat posed by quantum computers. The National Institute of Standards and Technology (NIST) has standardized lattice-based schemes such as CRYSTALS-Kyber [1] and CRYSTALS-Dilithium [2], while additional digital signature candidates are being evaluated through the ongoing NIST process [3]. Beyond PQC, lattice-based constructions are also widely used in homomorphic encryption (HE) schemes [4].

A common computational unit in these cryptographic systems is polynomial multiplication, which is typically accelerated using the Number Theoretic Transform (NTT) [5]. Compared with conventional schoolbook polynomial multiplication, which has quadratic complexity $O(n^2)$, NTT-based multiplication reduces the computational complexity to $O(n \log n)$, making it highly suitable for efficient hardware implementation [6].

The NTT can be regarded as the finite-field counterpart of the Fast Fourier Transform (FFT). Unlike the FFT, which relies on complex arithmetic and floating-point operations, the NTT performs all computations over a finite field or polynomial ring by replacing complex roots of unity with integer roots of unity. This property makes the NTT particularly attractive for hardware accelerators, where modular arithmetic can be implemented efficiently. In lattice-based cryptographic algorithms, NTT-based polynomial multiplication is often one of the most computationally demanding operations. Therefore, designing efficient NTT accelerators is essential for improving the performance and practicality of PQC implementations.

NTT-based polynomial multiplication consists of three main operations: forward NTT (FNTT), coefficient-wise multiplication (CWM), and inverse NTT (INTT). The butterfly operation is the fundamental computational unit used to perform FNTT and INTT transformations. Among the commonly used butterfly structures, the Cooley–Tukey (CT) and Gentleman–Sande (GS) configurations are widely adopted for efficient FNTT and INTT computation, respectively [6,7].

1.1. Related NTT-Based Hardware Accelerators

Numerous NTT-based polynomial multiplication accelerators have been reported for implementation on field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) [7–31]. The works reported in [7–11] investigate NTT implementations for distinct PQC algorithms, whereas NTT architectures targeting homomorphic encryption applications are explored in [12–14]. The designs in [15–20] focus on CRYSTALS-Kyber, while dedicated accelerators for CRYSTALS-Dilithium are presented in [21–27]. In contrast, unified architectures supporting both CRYSTALS-Kyber and CRYSTALS-Dilithium are described in [28–31]. Beyond NTT accelerators, supporting arithmetic frameworks have also been explored. An open-source generator for large-integer polynomial multipliers targeting FPGA and ASIC implementations is presented in [32], with extended architectural evaluations reported in [33]. Moreover, a multiplierless constant multiplication framework is introduced in [34], which can potentially be employed to optimize twiddle-factor multiplications in NTT architectures.

A low-complexity NTT accelerator targeting the NewHope key-exchange scheme was presented in [7], emphasizing efficient transform computation for lattice-based cryptography. The work in [8] explored ultra-high-throughput implementations by comparing conventional schoolbook polynomial multiplication with NTT-based approaches. To support a wide range of application scenarios, from resource-constrained IoT devices to high-performance computing platforms, an open-source compile-time configurable NTT generator was proposed in [9], enabling evaluation across multiple polynomial lengths. A pipelined NTT architecture utilizing the Negative Wrapped Convolution (NWC) method was employed in [10] to improve computational throughput. Furthermore, a lightweight coprocessor based on the Ring Learning-With-Errors (R-LWE) cryptosystem was reported in [11], where noise approximation techniques were exploited to reduce hardware complexity for IoT-oriented applications.

For homomorphic encryption applications, an NTT accelerator employing a zero-padding technique was implemented on a Kintex UltraScale+ FPGA in [12]. To facilitate the automated generation of efficient NTT architectures for fully homomorphic encryption

(FHE), the NTTGen framework was proposed in [13], enabling the synthesis of low-latency FPGA-based NTT implementations. More recently, a lightweight and hardware-efficient FPGA accelerator tailored for FHE applications was presented in [14], where a novel twiddle-factor generation method for Radix-2 multi-path delay commutator (MDC) architectures was introduced to improve throughput while maintaining low implementation overhead.

For CRYSTALS-Kyber, several lightweight and high-performance NTT accelerators have recently been reported. The AT-NTT architecture proposed in [15] introduced a fully lookup-table (LUT)-based modular reduction approach to efficiently implement modular multiplication. In addition, the design employed a novel negative-biased table in the final reduction stage to simplify addition and subtraction operations. An ultra-lightweight and memory-free NTT core targeting CRYSTALS-Kyber parameters was presented in [16], focusing on minimizing hardware overhead. For the same parameter set, the importance of unifying CT and GS butterfly structures was experimentally investigated in [17], demonstrating the potential benefits of resource sharing in NTT implementations. More recently, the unified parallel pipelined NTT (UPP-NTT) accelerator was introduced in [18], where four parallel pipelined radix-2 butterfly functional units (BFUs) were integrated into a reconfigurable datapath supporting both FNTT and INTT computations. Their architecture further incorporated a shift-and-add-based Barrett reduction scheme to reduce dependence on DSP resources, while a conflict-free single-memory organization and dedicated address generation logic enabled deterministic access to polynomial coefficients and twiddle factors. The work in [19] presents a RISC-V system-on-chip (SoC)-integrated NTT accelerator for CRYSTALS-Kyber and evaluates architectural trade-offs among parallelism, memory organization, area, and performance. Recently, the work in [20] presented a unified SRAM-based ASIC accelerator for Crystals-Kyber that supports all NIST security levels while eliminating FIFOs and intermediate buffers to achieve high area–time efficiency, low power consumption, and low energy per operation.

Lightweight hardware accelerators supporting CRYSTALS-Dilithium, including dedicated NTT computation, have been proposed in [21,22], emphasizing a reduction in hardware costs while maintaining acceptable performance. The study reported in [23] investigated the area, power, and performance trade-offs of NTT implementations using CRYSTALS-Dilithium parameters, highlighting the impact of architectural choices on hardware efficiency. In [24], the EMINEM framework introduced efficient FPGA implementations of mixed-radix NTT accelerators targeting multiple lattice-based schemes, including Falcon, CRYSTALS-Dilithium, and HAWK. More recently, an optimized hardware architecture for Dilithium across all security levels was presented in [25]. Their design incorporated parallel modular multiplication units together with an enhanced scaling method to reduce operand bit width and calibration overhead. Furthermore, an improved radix-2 Multipath Delay Commutator NTT (MDC-NTT) architecture and pipelined parallelization using FIFO and BRAM-based buffers were employed to increase the achievable operating frequency. The architecture in [26] proposes a resource-efficient FPGA implementation of Dilithium-NTT using an optimized modular multiplier and dual ping-pong memory scheme to improve area–time efficiency and operating frequency. In [27], a lightweight Crystals-Dilithium accelerator is presented, employing an iterative dual-Keccak SHA-3 module, optimized rejection sampling, and NTT stage elimination via lookup tables to improve area efficiency and latency.

A memory-unified NTT-based accelerator named CRYPTOR was introduced in [28] to support CRYSTALS PQC schemes. The architecture integrates dedicated arithmetic and control modules to accelerate FNTT, INTT, CWM, and additional polynomial operations within a shared hardware framework. High-speed NTT implementations based

on radix-2 and radix-4 butterfly structures were further investigated in [29], where the transform latency was reduced through optimized butterfly scheduling. In [30], a unified accelerator targeting both SABER and CRYSTALS-Dilithium was proposed by exploiting the modular arithmetic property $2^{23} - 2^{13} + 1$ to simplify modular reduction. Similarly, the KALI architecture presented in [31] adopted resource sharing and pipelining techniques to improve hardware efficiency and timing performance for unified CRYSTALS-based cryptographic operations.

1.2. Limitation(s) of Existing Accelerators

Existing NTT hardware accelerators discussed in Section 1.1 have primarily focused on improving a single design objective, such as maximizing throughput through parallel butterfly units and pipelining [15,18,25], or minimizing hardware utilization using iterative computation and resource sharing [16,21,22]. Although several studies report area and timing results, comprehensive evaluations considering the trade-offs among hardware cost, operating frequency, execution latency, and power consumption remain comparatively limited. Furthermore, only a few studies investigate energy-related metrics despite their importance for lightweight cryptographic implementations [17,23].

Such trade-offs are particularly critical for resource-constrained platforms, including Internet-of-Things (IoT) devices, radio-frequency identification (RFID) systems, and embedded secure processors, where compact hardware and low power consumption are often prioritized over maximum throughput. In contrast, high-performance edge and cloud-based systems require improved timing closure and higher operating frequencies. Consequently, lightweight NTT accelerators should ideally provide flexibility to operate across different timing constraints while maintaining acceptable area and energy efficiency. Moreover, most existing studies report implementation results at a fixed operating frequency, providing limited insight into the impact of timing constraints and pipelining on hardware utilization, power consumption, and overall performance. Therefore, evaluating lightweight NTT architectures across multiple operating frequencies becomes important to understand the trade-offs between compactness, energy efficiency, and timing scalability.

1.3. Proposed Work

Before describing the contributions, we highlight that our previous conference work published in [23] investigated only the non-pipelined architecture concerning area, power, and performance characteristics of lightweight NTT accelerators targeting CRYSTALS-Dilithium parameters. This article extends the work of [23] by performing a comprehensive evaluation of two architectural variants, namely non-pipelined (although the non-pipelined architecture considered in this work employs an identical datapath to that reported in [23], slight differences in the reported hardware utilization and power consumption are observed because of updated synthesis settings, implementation constraints, and post-place-and-route (PPnR) optimization strategies adopted in the present work) and 4-stage pipelined implementations, under different operating frequencies to analyze timing scalability, hardware utilization, execution latency, power consumption, and energy efficiency. Specifically, the primary objective of this work is to investigate the area–power–performance trade-offs between lightweight non-pipelined and pipelined NTT architectures rather than to perform a comprehensive pipeline-depth design-space exploration. The main contributions of this work are summarized as follows:

- **Non-pipelined Architecture:** A lightweight non-pipelined NTT-based polynomial multiplication accelerator is implemented using a single reusable butterfly unit supporting both CT and GS butterfly configurations. The same arithmetic datapath is iteratively reused to perform FNTT, INTT, and CWM, thereby minimizing hardware utilization

and power consumption at the expense of increased clock cycles and execution latency compared with highly parallel NTT accelerators.

- **Pipelined Architecture:** A 4-stage pipelined variant of the NTT architecture is implemented by inserting registers within the arithmetic datapath to improve timing closure and increase the achievable operating frequency. Compared with the non-pipelined implementation, the pipelined architecture incurs a moderate increase in hardware utilization and execution latency due to additional pipeline stages, while enabling improved timing scalability and reduced execution time at higher operating frequencies.
- **Comprehensive Results Evaluation:** The implemented non-pipelined and pipelined architectures are implemented on an Artix-7 FPGA and analyzed across operating frequencies ranging from 10 MHz to 280 MHz. The resulting hardware utilization, computation time, and power consumption are examined to investigate timing scalability and area–power–performance trade-offs under different implementation constraints. Furthermore, additional performance assessments are conducted using Area–Time Product (ATP) and Energy–Delay Product (EDP) metrics to quantify the combined effects of hardware cost, execution latency, and energy consumption. The obtained results are further compared with existing lightweight and high-performance NTT accelerators to evaluate the proposed architectures in terms of compactness, energy efficiency, and overall performance.

We want to highlight that this work is an extended version of work previously published in [23], which focused exclusively on a non-pipelined NTT architecture based on a single reusable butterfly unit. The present work extends the design by introducing a 4-stage pipelined architecture and a comprehensive evaluation framework. While the underlying butterfly architecture and functionality remain unchanged, the pipelined design improves the maximum operating frequency from 250 MHz to 280 MHz and achieves timing closure at higher operating frequencies. Furthermore, this work provides a detailed analysis of the trade-offs between non-pipelined and pipelined implementations in terms of hardware utilization, computation time, power consumption, energy consumption, ATP and EDP over a wide operating-frequency range. These additions significantly broaden the scope of the previous work of [23] and provide deeper insights into the impact of pipelining on lightweight CRYSTALS-Dilithium NTT accelerators.

The remainder of this article is organized as follows. Section 2 presents the mathematical background of NTT-based polynomial multiplication. Section 3 describes the proposed non-pipelined and pipelined architectures. Section 4 discusses FPGA implementation results and evaluates the proposed architectures in terms of area, power, timing, ATP, and EDP metrics. Finally, Section 5 concludes the article and outlines future research directions.

2. Background

This section introduces the mathematical foundations and implementation principles relevant to the proposed NTT-based polynomial multiplication architecture. Section 2.1 defines the notation and algebraic parameters associated with the CRYSTALS-Dilithium scheme. Section 2.2 describes the mathematical flow of NTT-based polynomial multiplication, including the forward NTT, coefficient-wise multiplication, and inverse NTT operations. Finally, Section 2.3 reviews commonly used implementation approaches for NTT computation, with emphasis on the Cooley–Tukey and Gentleman–Sande (GS) butterfly structures adopted in this work.

2.1. Notation and Mathematical Preliminaries

The CRYSTALS-Dilithium signature scheme [2] operates over the polynomial ring $R_{n,q} = \mathbb{Z}_q[x]/(x^n + 1)$, where the polynomial degree is fixed at $n = 256$, and the modulus

is defined as $q = 2^{23} - 2^{13} + 1$. Since q is prime, it enables the existence of suitable primitive roots of unity required for efficient NTT computation. In particular, the selected modulus supports a primitive 512-th root of unity, which allows the decomposition of a 256-point transform into $\log_2(256) = 8$ computational stages.

Throughout this work, the stage index associated with the iterative transform process is denoted by s_i , where $i \in \{1, 2, \dots, 8\}$. The input polynomials in the standard algebraic domain are represented by $a(x)$ and $b(x)$, whereas their transformed NTT-domain representations are denoted by $\tilde{a}(x)$ and $\tilde{b}(x)$, respectively. The symbol ω represents the twiddle factor corresponding to the primitive root of unity used during butterfly computation. Furthermore, n^{-1} denotes the modular multiplicative inverse of n modulo q , and the operator $\circ$ indicates coefficient-wise multiplication.

2.2. NTT-Based Polynomial Multiplication

Polynomial multiplication in the algebraic domain can be expressed as $c(x) = a(x) \cdot b(x) \bmod (x^n + 1, q)$, where the resulting polynomial coefficients are reduced modulo q , and the polynomial degree remains less than n . Conventional multiplication approaches such as schoolbook multiplication, Karatsuba multiplication, and Toom–Cook multiplication may be used to compute this operation [33]. However, these methods become increasingly expensive as the polynomial degree grows. To reduce computational complexity, lattice-based cryptographic systems commonly employ NTT-domain multiplication. In this approach, the input polynomials are first transformed into the spectral domain using the forward Number Theoretic Transform (i.e., FNTT). For a polynomial $a(x)$, the transformation is mathematically defined as

$$\tilde{a}_i = \sum_{j=0}^{n-1} a_j \omega^{ij} \bmod q, \quad i \in [0, n-1]. \quad (1)$$

After both input polynomials are transformed into the NTT domain, multiplication is performed independently on the corresponding coefficients. This operation is commonly referred to as coefficient-wise multiplication (i.e., CWM) and is represented as

$$\tilde{c}(x) = \tilde{a}(x) \circ \tilde{b}(x). \quad (2)$$

Finally, the inverse Number Theoretic Transform (i.e., INTT) is applied to map the resultant polynomial back to the algebraic domain. The inverse transformation is expressed as

$$c_i = n^{-1} \sum_{j=0}^{n-1} \tilde{c}_j \omega^{-ij} \bmod q, \quad i \in [0, n-1]. \quad (3)$$

Compared with direct polynomial multiplication methods, the NTT-based approach significantly lowers arithmetic complexity and, therefore, provides substantial advantages for hardware acceleration in lattice-based cryptographic systems.

2.3. Architectural Approaches for NTT Computation

Several implementation techniques have been proposed in the literature to realize the transformations given in Equations (1) and (3). One conventional approach employs zero-padding to support negacyclic polynomial multiplication [12]. Although functionally straightforward, this method introduces additional computational overhead due to the enlarged transform size.

An alternative technique is the Negative Wrapped Convolution (NWC) method, which eliminates explicit zero-padding and improves computational efficiency [10]. Nevertheless,

NWC-based architectures require additional pre-processing and post-processing operations, increasing design complexity. In contrast, the Cooley–Tukey and Gentleman–Sande butterfly formulations inherently support efficient NTT computation without requiring separate NWC processing stages [6,7].

In this work, the CT butterfly structure is employed for forward transformation, while the GS butterfly structure is utilized for inverse transformation. The CT butterfly accepts inputs in normal order and generates outputs in bit-reversed order. Conversely, the GS butterfly operates on bit-reversed inputs and restores the outputs to normal order. This complementary behavior enables efficient realization of iterative FNTT and INTT computation without introducing additional permutation hardware. Therefore, the fundamental arithmetic operations associated with CT and GS butterflies are illustrated in Figure 1. These butterfly units form the core computational component of the proposed architecture and are repeatedly reused during iterative transform execution.

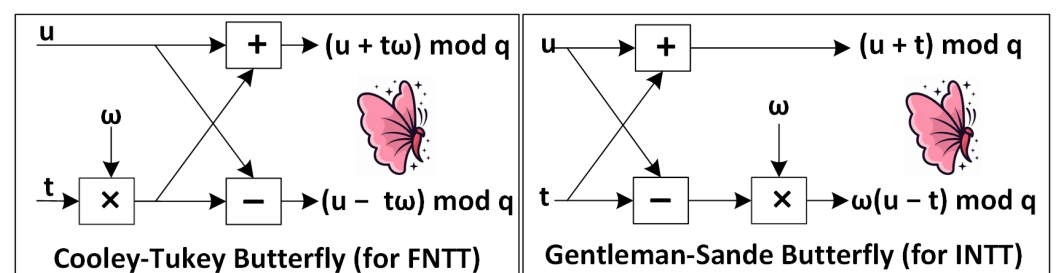


Figure 1. Basic computational structures of the Cooley–Tukey (CT) and Gentleman–Sande (GS) butterfly units.

3. Proposed NTT-Based Polynomial Multiplication Architectures

The implemented hardware accelerator architecture is illustrated in Figure 2. It consists of three units: a unified butterfly unit (BU), a memory subsystem, and a finite-state machine (FSM)-based control circuit. Together, these modules execute the forward NTT, coefficient-wise multiplication, and inverse NTT operations required for polynomial multiplication in CRYSTALS-Dilithium. To investigate the trade-off between hardware efficiency and processing speed, two variants of the proposed architecture are developed in this work: a non-pipelined implementation and a 4-stage pipelined implementation. Both architectures employ the same iterative transform strategy and unified butterfly datapath. However, they differ in the organization of the modular multiplication and Barrett reduction stages. The non-pipelined design prioritizes reduced switching activity and lower hardware overhead, making it suitable for lightweight and low-power cryptographic applications. In contrast, the 4-stage pipelined architecture introduces pipeline registers within the modular multiplication and reduction datapaths to shorten the combinational critical path and improve timing performance at higher operating frequencies. This pipelining strategy increases latency slightly due to pipeline filling and draining overheads but improves throughput.

The BU serves as the primary computational engine of both architectures. It supports CT and GS butterfly configurations for FNTT and INTT transformations while also reusing the same datapath for coefficient-wise multiplication. The detailed organization of the butterfly architecture, including the pipelined and non-pipelined datapaths, is discussed in Section 3.1. Moreover, the memory subsystem stores the input, intermediate, and output polynomial coefficients during transform execution. Two dual-port block-RAMs (BRAMs) are utilized for iterative coefficient processing, while a dedicated read-only memory (ROM) stores the twiddle factors required during butterfly computation. Additional details regarding the memory organization are provided in Section 3.2. Finally, the FSM-based controller

coordinates all computational activities within the non-pipelined and pipelined accelerators. The controller generates memory addresses, butterfly-selection signals, routing controls, and transform-stage synchronization signals required for iterative execution. It additionally manages pipeline synchronization and arithmetic scheduling. A detailed description of the control mechanism and computational cost analysis is presented in Section 3.3.

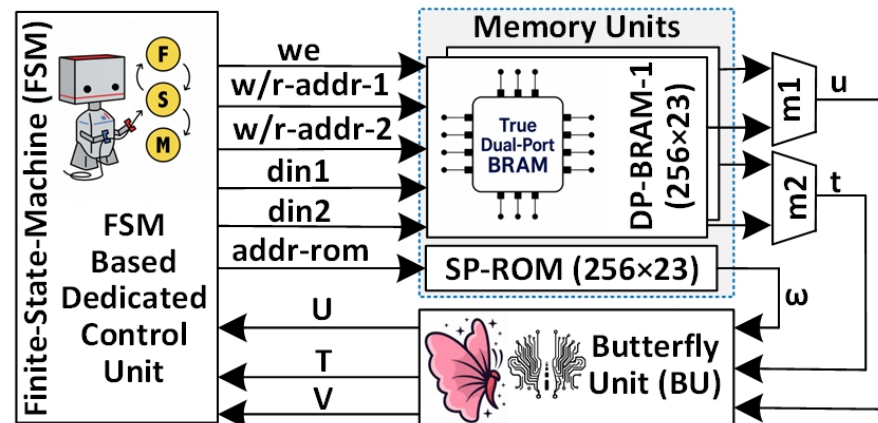


Figure 2. Top-level architecture of the implemented lightweight NTT accelerator supporting both non-pipelined and pipelined implementations. Internal data-loading and storage operations are omitted for clarity and are managed by the FSM-based controller.

3.1. Four-Stage Pipelined, Unified Butterfly Unit (BU)

The BU forms the main computational engine of the proposed NTT accelerator, as illustrated in Figure 3. The architecture adopts a unified datapath that supports both CT and GS butterfly configurations while also enabling coefficient-wise multiplication using the same arithmetic resources. The design consists of two adders, two subtractors, a modular multiplication unit, and several routing multiplexers. By dynamically reconfiguring the datapath through control signals, the same hardware structure is reused for FNTT, INTT, and CWM operations, resulting in a compact and resource-efficient implementation. Moreover, unlike many existing NTT accelerators that heavily rely on deep pipelining to maximize throughput [7,10,28,30,31], the proposed BU architecture adopts a lightweight pipelining strategy. Only selected pipeline registers are inserted within the modular multiplication datapath to support reliable operations while avoiding excessive switching activity and register overhead. This design choice maintains low power consumption while still enabling practical operating frequencies on FPGA devices. We have provided more detailed insights into the BU architectures in Sections 3.1.1–3.1.3.

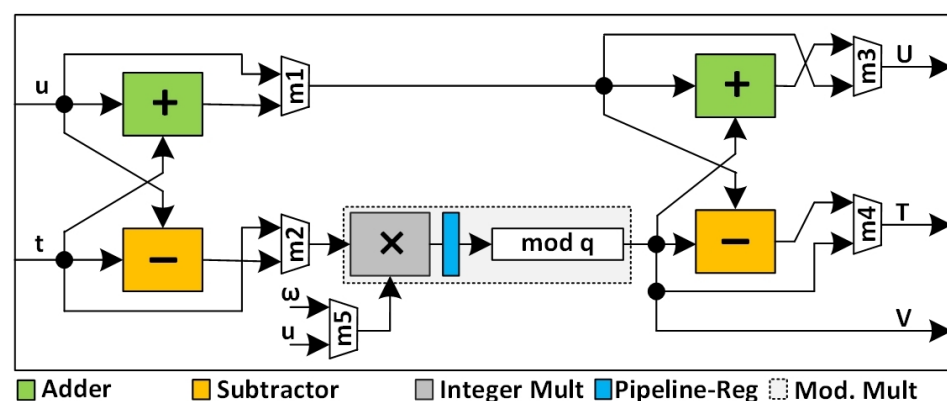


Figure 3. A 4-stage pipelined, unified butterfly architecture supporting CT and GS operations for FNTT and INTT computation. The same datapath is reused for coefficient-wise multiplication.

3.1.1. FNTT and INTT Computation

As illustrated in Figure 3, the butterfly unit receives three 23-bit inputs denoted by u , t , and ω . The operands u and t correspond to polynomial coefficients retrieved from memory, whereas ω represents the associated twiddle factor supplied from the SP-ROM. The outputs of the butterfly structure are represented by U , T , and V , each having a width of 23 bits. The signals U and T correspond to transform outputs generated during CT or GS operation, while V is reserved for coefficient-wise multiplication.

The arithmetic datapath combines two addition units and two subtraction units with a shared modular multiplication block. Since CRYSTALS-Dilithium operates using 23-bit coefficients, conventional arithmetic operators are sufficient for implementing addition and subtraction efficiently on FPGA fabric. Therefore, no specialized arithmetic optimization is required for these operations. In contrast, modular multiplication introduces substantially higher computational complexity because the multiplication result must be reduced modulo q .

Data movement within the butterfly structure is controlled through multiplexers $m1$, $m2$, $m3$, and $m4$. These multiplexers are driven by a one-bit control signal referred to as *mode*. Depending on the value of this signal, the datapath dynamically switches between CT and GS butterfly configurations. When $mode = 0$, the architecture performs the CT butterfly operation according to $U = (u + t\omega) \bmod q$, $V = (u - t\omega) \bmod q$. Conversely, when $mode = 1$, the same datapath executes the GS butterfly operation as $U = (u + t) \bmod q$, $V = \omega(u - t) \bmod q$. Through this unified arithmetic structure, both forward and inverse transforms are executed using the same hardware resources, significantly reducing implementation overhead compared with separate CT and GS realizations.

3.1.2. Modular Multiplication and CWM Computation

The modular multiplication section of the butterfly architecture is highlighted in Figure 3 using a dotted boundary filled with light gray color. This block consists of an integer multiplier followed by a Barrett-based modular reduction stage (i.e., $\bmod q$). The same modular multiplication hardware is reused for both transform operations (i.e., FNTT and INTT) and coefficient-wise multiplication.

The integer multiplier accepts two 23-bit operands. One operand is selected through multiplexer $m2$, while the second operand is chosen using multiplexer $m5$. For transform computation, the multiplier receives the twiddle factor ω together with either the coefficient t or the subtraction result generated by the butterfly datapath. During coefficient-wise multiplication, the routing logic reconfigures the multiplier input so that polynomial coefficients are multiplied directly.

The signal controlling multiplexer $m5$ is denoted as *flag-cwm*. When $flag-cwm = 0$, the multiplier operates in transform mode for either FNTT or INTT computation and receives the twiddle factor ω . Alternatively, when $flag-cwm = 1$, the multiplier accepts the coefficient u to perform coefficient-wise multiplication. This resource-sharing strategy enables a single multiplier architecture to support multiple computational modes, thereby reducing DSP utilization and logic duplication.

The output of the integer multiplier produces a 46-bit intermediate result that is subsequently forwarded to the modular reduction unit. Since all polynomial coefficients must remain within the finite field $\mathbb{Z}_q$, modular reduction is required before the result can be stored or reused by subsequent transform stages.

3.1.3. Barrett-Based Modular Reduction

The proposed modular reduction architecture is based on the Barrett reduction algorithm, which avoids expensive division operations by replacing them with multiplication

and shift operations [35]. Since the output of integer multiplication exceeds the coefficient width defined by the modulus q , modular reduction is required to map the intermediate result back into the finite field $\mathbb{Z}_q$.

For an intermediate multiplication result a , the Barrett reduction operation is approximated as

$$z \approx a - \underbrace{\left(\underbrace{(a \cdot q_{inv})}_{a1} \gg \text{len}(a) \right)}_{a2} \cdot q \quad (4)$$

where z denotes the reduced modular result, $\text{len}(a)$ represents the bit-width of a , and q_{inv} is a precomputed reciprocal approximation given by

$$q_{inv} = \frac{2^{\text{len}(a)}}{q}.$$

The hardware architecture corresponding to Equation (4) is illustrated in Figure 4. The datapath consists of two integer multipliers (IM1 and IM2, with gray-filled boxes), two subtraction units (orange-filled boxes), lightweight pipeline registers (blue vertical strips), and a 2×1 routing multiplexer $m6$. The architecture is structured into multiple stages to reduce combinational delay and improve timing stability. Initially, the 46-bit multiplication result a enters the Barrett reduction block and is first stored in a pipeline register. This register isolates the incoming multiplication stage from the remaining modular reduction circuitry, thereby shortening the critical path and improving signal stability. The registered value of a is then multiplied by the constant q_{inv} using the first integer multiplier (IM1). The multiplication result is subsequently shifted right by $\text{len}(a)$ bits to generate the intermediate approximation term $a1$. A second pipeline register is inserted after this stage to separate the shift operation from the subsequent multiplication stage.

Next, the intermediate signal $a1$ is multiplied by the modulus constant q using the second integer multiplier (IM2), producing the approximation value $a2$. To further reduce timing overhead, another pipeline register is placed after this multiplication stage before the subtraction operations are performed.

The modular correction stage begins by subtracting $a2$ from the delayed version of the original multiplication result a , producing the following intermediate result:

$$z' = a - a2$$

Due to the approximate nature of Barrett reduction, the value of z' may occasionally exceed the modulus q . Therefore, a final modular correction step is required to ensure that the output remains within the finite field $\mathbb{Z}_q$. The correction operation is performed by a second subtraction unit, which computes $z' - q$. Subsequently, multiplexer $m6$ selects either z' or $z' - q$ according to the following condition:

$$z = \begin{cases} z' & \text{if } z' < q \\ z' - q & \text{if } z' \geq q \end{cases} \quad (5)$$

Consequently, the final output z is guaranteed to lie within the valid modular range $[0, q - 1]$. This conditional subtraction preserves the correctness of the Barrett reduction while requiring only a small amount of additional hardware. The proposed modular reduction was functionally verified over the expected 46-bit multiplication range in the CRYSTALS-Dilithium datapath.

For the targeted CRYSTALS-Dilithium parameter set, the modulus and reciprocal constants are fixed as $q = 8380417$ and $q_{inv} = 8396807$. Previous studies have explored shift-add realizations and arithmetic optimizations to simplify Barrett reduction hardware [10,21,30]. In contrast, the proposed architecture employs dedicated integer multipliers together with lightweight pipelining registers. Although this approach increases DSP utilization, it simplifies arithmetic routing, improves timing robustness, maintains moderate hardware complexity, and reduces switching activity.

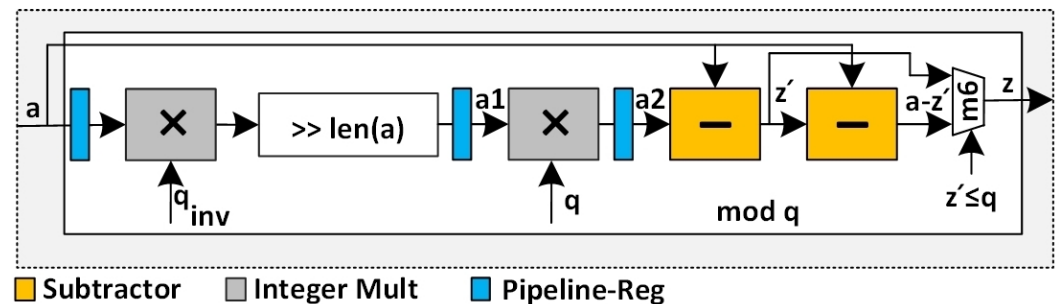


Figure 4. Proposed 3-stage pipelined Barrett-based modular reduction architecture.

3.2. Memory Subsystem

To sustain iterative executions of the FNTT, CWM, and INTT, the design incorporates two dual-port block RAMs, denoted as DP-BRAM-1 and DP-BRAM-2. Since butterfly computation requires simultaneous access to two polynomial coefficients during each processing cycle, dual-port memory organization is adopted to support parallel read and write operations efficiently. Each memory block is configured with a depth of 256 locations and a word width of 23 bits, corresponding to the polynomial degree and modulus representation used in CRYSTALS-Dilithium. In addition to the coefficient memories, a dedicated single-port ROM (SP-ROM) of size 23×256 stores the precomputed twiddle factors ω required during transform computation.

The FNTT and INTT procedures are executed using a ping-pong memory mechanism between DP-BRAM-1 and DP-BRAM-2. During computation, one memory block supplies input coefficients to the butterfly unit, while the transformed outputs are simultaneously written to the alternate memory block. This alternating memory-access strategy avoids data conflicts and supports iterative stage-wise processing. For coefficient-wise multiplication, the transformed polynomials $\tilde{a}(x)$ and $\tilde{b}(x)$ are loaded into the two DP-BRAMs, and the generated multiplication results are written back into DP-BRAM-1.

3.3. Control Unit and Computational Cost of the Proposed Architecture

The proposed NTT accelerator employs a dedicated FSM-based control unit to coordinate the overall execution of the FNTT, CWM, and INTT. The controller manages the read/write operations of the utilized memories and generates the control signals required for the routing multiplexers shown in Figures 3 and 4. In addition, the FSM controls the operational sequencing of the unified butterfly unit during FNTT, INTT, and CWM computations. Since the architecture utilizes a single butterfly unit, all transform operations are executed iteratively. Consequently, the controller is responsible for stage scheduling, butterfly indexing, memory synchronization, coefficient routing, and twiddle-factor addressing throughout the computation process. The FSM further controls the *mode* and *flag-cwm* signals to dynamically switch the butterfly datapath between CT, GS, and coefficient-wise multiplication modes.

To illustrate the iterative execution mechanism, Figure 5 presents the data-flow organization of a 16-point transform. The forward transform progresses sequentially from stage-1 to stage-4, whereas the inverse transform follows the reverse order from stage-4 back to

stage-1. Each filled block in the figure represents one butterfly operation, while the highlighted green and orange blocks indicate the iterative scheduling pattern adopted by the controller. Therefore, for a 16-point transform, the execution requires a total of 15 butterfly iterations distributed across four stages. The same computation strategy directly extends to the CRYSTALS-Dilithium parameter set with $N = 256$, where the transform requires eight stages. In this case, the iterative architecture performs 255 butterfly iterations to complete one full FNTT or INTT computation. The FSM controller sequentially updates butterfly addresses and twiddle-factor indices at every stage to ensure correct coefficient pairing throughout the transform process.

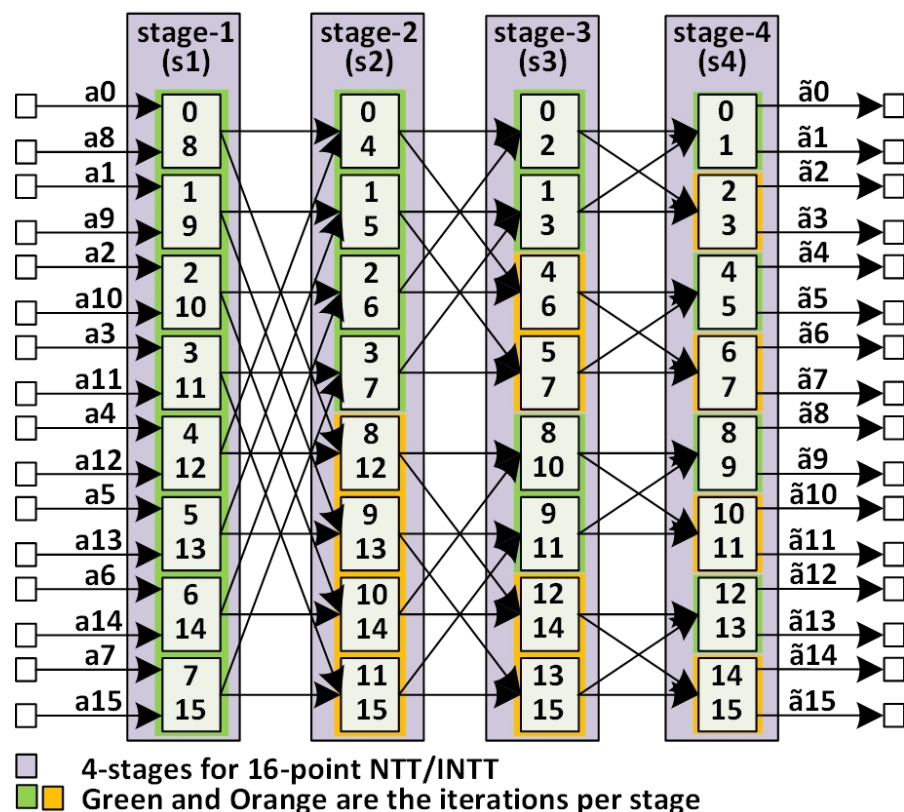


Figure 5. Data-flow organization of the iterative butterfly execution for a 16-point FNTT/INTT transform. The highlighted green and orange blocks indicate the butterfly iterations scheduled within each transform stage.

3.3.1. Clock-Cycle Analysis of the Non-Pipelined Architecture

For an iterative NTT architecture employing a single butterfly unit without pipelining, the theoretical computation cost of one transform is given by $N \times \log_2(N)$, where N denotes the transform length. For CRYSTALS-Dilithium with $N = 256$, the ideal transform cost therefore becomes $256 \times 8 = 1024$ clock cycles. However, practical implementation overheads associated with memory access increase the overall execution latency. In the proposed architecture, coefficient operands are fetched from dual-port BRAMs, where valid data become available after memory-access synchronization. Consequently, each butterfly iteration incurs an additional clock cycle before arithmetic processing can begin.

As illustrated in Figure 5, the iterative scheduling of a 16-point transform requires 15 butterfly iterations. Extending the same scheduling strategy to $N = 256$ results in 255 iterations. Therefore, the overall transform latency becomes $1024 + 256 = 1280$ clock cycles, where the additional 256 cycles originate from: (1) one initial memory synchronization cycle, and (2) 255 iteration-dependent operand-fetch cycles. Thus, the proposed iterative architecture requires 1280 clock cycles to complete one FNTT or INTT operation.

The coefficient-wise multiplication stage follows a different execution pattern because no multi-stage transform scheduling is required. The CWM operation processes one coefficient pair per iteration across all 256 coefficients. Accordingly, the total latency for CWM computation becomes $256 + 1 + 1 = 258$ clock cycles, where: (1) 256 cycles correspond to coefficient multiplications, (2) one cycle is required for the first valid memory read, and (3) one additional cycle is used to assert the completion signal.

3.3.2. Clock-Cycle Analysis of the Pipelined Architecture

The proposed design additionally incorporates lightweight pipelining registers inside the modular multiplication and Barrett reduction datapaths, as illustrated previously in Figures 3 and 4. In the butterfly architecture, one pipeline register is inserted after the integer multiplication stage. Furthermore, the Barrett reduction unit contains three additional pipeline registers distributed across the reduction datapath.

Consequently, the modular multiplication path introduces a total pipeline depth of four stages. Since the butterfly outputs depend on the completion of modular multiplication, the arithmetic results experience a four-cycle latency before becoming available for subsequent operations. Moreover, for the pipelined architecture, the transform computation still performs the same number of butterfly iterations as the non-pipelined design. However, additional latency is introduced during pipeline filling and draining. Therefore, the total clock-cycle cost for one FNTT or INTT operation becomes $1024 + 256 + 4 = 1284$ clock cycles, where the extra four cycles correspond to the modular multiplication pipeline depth. Similarly, the total clock-cycle cost for coefficient-wise multiplication becomes $256 + 1 + 1 + 4 = 262$ clock cycles.

Although the pipelined organization introduces a small latency overhead, it significantly shortens the combinational critical path by separating multiplication and reduction stages across multiple cycles. As a result, the architecture can operate at higher clock frequencies while maintaining stable timing behavior.

4. Implementation Results and Comparisons

The proposed NTT-based polynomial multiplication architectures were implemented in Verilog HDL and synthesized using Vivado 2023.2, targeting the Artix-7 FPGA device (XC7A100TCSG324-3). For benchmarking and performance evaluation, two operating-frequency ranges are considered: (i) 10–40 MHz with an increment of 10 MHz, and (ii) 100–250 MHz with an increment of 50 MHz. The functionality of the implemented architectures was verified using 1000 test cases. The final FNTT, INTT, and CWM outputs generated by the RTL implementation were compared against a software-based Python (v 3.13) golden model implementing the CRYSTALS-Dilithium reference implementation. The implementation results, including hardware utilization, computation time, and power consumption, are presented in Section 4.1, whereas the performance evaluation using Area–Time Product and Energy–Delay Product metrics is discussed in Section 4.2.

4.1. Hardware Area–Time and Power Results

Table 1 summarizes the implementation results for both the non-pipelined and 4-stage pipelined architectures using the CRYSTALS-Dilithium parameters $N = 256$ and $q = 8380417$. The reported metrics include hardware utilization, execution latency in clock cycles, computation time, and average power consumption for the FNTT, INTT, and CWM operations, where all results are obtained after post-place-and-route (PPnR) analysis. Specifically, the reported power values were obtained using the Vivado Power Analyzer after PPnR implementation of the proposed architectures. Power estimation was performed using switching activity captured from Voltage Change Dump (VCD) files generated during

simulation and imported into the Vivado Power Analyzer. The analysis was conducted using the default Vivado power-analysis settings under a typical operating condition of 25 °C. Moreover, the pipelined architecture achieves a maximum operating frequency of 280 MHz while satisfying the specified timing constraints, whereas the non-pipelined architecture fails to meet timing closure at this frequency due to its longer critical-path delay, resulting in negative timing slack. For all reported implementation results that satisfy timing closure, the Worst Negative Slack (WNS) remains non-negative and is typically below 0.5 ns. The implementation results, including hardware area, clock cycles, computation time, and power consumption, for the non-pipelined and pipelined architectures are discussed in Section 4.1.1 and Section 4.1.2, respectively.

Table 1. Implementation results on Artix-7 FPGA for $N = 256$ and $q = 8380417$. The reported FNTT, INTT, and CWM clock cycles and execution times correspond to individual transform operations, whereas Poly-Mul denotes the end-to-end polynomial multiplication cost computed as $2T_{\text{FNTT}} + T_{\text{CWM}} + T_{\text{INTT}}$.

Design	Freq. (MHz)	Slices	LUTs	FFs	DSPs	BRAMs	Execution Latency (Clock Cycles) (FNTT/INTT/CWM/Poly-Mul)	Time (μ s) (FNTT/INTT/CWM/Poly-Mul)	Average Power
Non-pipelined	10	860	3071	602	6	3	1280/1280/258/4098	128/128/25.80/409.80	3 mW
	20	870	3073	602	6	3	1280/1280/258/4098	64/64/12.90/204.90	6 mW
	30	891	3076	602	6	3	1280/1280/258/4098	42.66/42.66/8.60/136.60	10 mW
	40	903	3119	619	6	3	1280/1280/258/4098	32/32/6.45/102.45	19 mW
	100	935	3180	619	6	3	1280/1280/258/4098	12.80/12.80/2.58/40.98	41 mW
	150	972	3248	619	6	3	1280/1280/258/4098	8.53/8.53/1.72/27.32	63 mW
	200	1014	3316	619	6	3	1280/1280/258/4098	6.40/6.40/1.29/20.49	87 mW
	250	1062	3395	619	6	3	1280/1280/258/4098	5.12/5.12/1.03/16.39	112 mW
	280	Timing constraints are not met							
4-stage pipelined	10	883	3071	694	6	3	1284/1284/262/4114	128.40/128.40/26.20/411.40	3 mW
	20	893	3073	694	6	3	1284/1284/262/4114	64.20/64.20/13.10/205.70	6 mW
	30	914	3076	694	6	3	1284/1284/262/4114	42.80/42.80/8.73/137.13	11 mW
	40	926	3119	711	6	3	1284/1284/262/4114	32.10/32.10/6.55/102.85	20 mW
	100	958	3180	711	6	3	1284/1284/262/4114	12.84/12.84/2.62/41.14	43 mW
	150	995	3248	711	6	3	1284/1284/262/4114	8.56/8.56/1.75/27.42	66 mW
	200	1037	3316	711	6	3	1284/1284/262/4114	6.42/6.42/1.31/20.57	91 mW
	250	1085	3395	711	6	3	1284/1284/262/4114	5.14/5.14/1.05/16.45	118 mW
	280	1115	3438	711	6	3	1284/1284/262/4114	4.58/4.58/0.93/14.69	133 mW

Note: Post-place-and-route timing analysis reports a WNS of +0.30 ns for the non-pipelined architecture at 250 MHz, −0.47 ns for the non-pipelined architecture at 280 MHz (timing not met), and +0.12 ns for the 4-stage pipelined architecture at 280 MHz, confirming that the inserted pipeline registers enable timing closure at 280 MHz.

4.1.1. Non-Pipelined Architecture

The non-pipelined architecture is designed primarily to minimize switching activity and hardware overhead. Since no intermediate pipeline registers are inserted within the datapath, the architecture maintains relatively low flip-flop utilization and reduced dynamic power consumption. Furthermore, the iterative reuse of a single butterfly unit enables compact hardware implementation for the FNTT, INTT, and CWM operations.

As shown in Table 1, the non-pipelined implementation utilizes 860 slices, 3071 LUTs, and 602 FFs at 10 MHz. Within the low-frequency operating range from 10 MHz to 40 MHz, the hardware utilization increases gradually to 903 slices, 3119 LUTs, and 619 FFs. This increase is mainly attributed to synthesis optimizations and routing adjustments required to satisfy tighter timing constraints at higher frequencies. A similar trend is also observed in the high-frequency operating range from 100 MHz to 250 MHz, where the slice utilization increases from 935 to 1062 and the LUT count rises from 3180 to 3395. In contrast, the FF utilization remains nearly constant because the architecture does not employ pipelining

registers within the datapath. Furthermore, the number of DSP blocks and BRAMs remains fixed at six and three, respectively, across all operating frequencies since the underlying arithmetic and memory resources are unchanged.

In terms of computational latency, the non-pipelined architecture consistently requires 1280 clock cycles for both the FNTT and INTT operations and 258 clock cycles for the CWM operation across all operating frequencies. Within the low-frequency operating range from 10 MHz to 40 MHz, the execution time decreases from 128 μ s to 32 μ s for the FNTT and INTT operations, while the CWM execution time decreases from 25.80 μ s to 6.45 μ s. Over the same frequency range, the average power consumption increases gradually from 3 mW to 19 mW. A similar trend is observed in the high-frequency operating range from 100 MHz to 250 MHz, where the execution time for the FNTT and INTT operations further decreases from 12.80 μ s to 5.12 μ s, and the CWM execution time decreases from 2.58 μ s to 1.03 μ s. However, this improvement in performance is accompanied by increased power consumption, which rises from 41 mW at 100 MHz to 112 mW at 250 MHz. Furthermore, Table 1 indicates that the non-pipelined architecture can operate successfully up to 250 MHz. Beyond this frequency, timing closure could not be achieved because of the long combinational delay associated with the modular multiplication and Barrett-based modular reduction stages, resulting in negative timing slack.

Since a complete NTT-based polynomial multiplication consists of two FNTT operations, one CWM operation, and one INTT operation, the corresponding end-to-end computational cost is also reported in Table 1 as Poly-Mul. For the non-pipelined architecture, the total latency remains constant at 4098 clock cycles across all operating frequencies. Consequently, the Poly-Mul execution time decreases from 409.80 μ s at 10 MHz to 102.45 μ s at 40 MHz and further to 16.39 μ s at 250 MHz, demonstrating the expected reduction in execution time with increasing operating frequency while preserving the same computational cost in clock cycles.

4.1.2. 4-Stage Pipelined Architecture

To improve timing performance and operating frequency, a 4-stage pipelined version of the proposed architecture was implemented. The pipelined design introduces four additional 23-bit registers within the butterfly and Barrett-based modular reduction datapaths, thereby shortening the critical combinational paths and improving timing closure. As shown in Table 1, the pipelined architecture utilizes 883 slices, 3071 LUTs, and 694 FFs at 10 MHz. Within the low-frequency operating range from 10 MHz to 40 MHz, the hardware utilization gradually increases to 926 slices, 3119 LUTs, and 711 FFs because of additional routing and synthesis optimizations required at higher operating frequencies. A similar trend is observed in the high-frequency operating range from 100 MHz to 280 MHz, where the slice utilization increases from 958 to 1115 and the LUT count rises from 3180 to 3438. The increase in FF utilization compared to the non-pipelined architecture is mainly caused by the inserted pipeline registers. However, the DSP and BRAM utilization remains constant at six and three, respectively, since the arithmetic and memory resources are reused without duplication.

In terms of computational latency, the pipelined architecture requires 1284 clock cycles for both the FNTT and INTT operations and 262 clock cycles for the CWM operation. The additional clock cycles originate from the latency introduced by the four pipeline stages. Within the low-frequency operating range from 10 MHz to 40 MHz, the execution time decreases from 128.40 μ s to 32.10 μ s for the FNTT and INTT operations, while the execution time for the CWM operation decreases from 26.20 μ s to 6.55 μ s. Over the same range, the average power consumption increases from 3 mW to 20 mW. Similarly, in the high-frequency operating range from 100 MHz to 280 MHz, the execution time for the FNTT and

INTT operations further decreases from 12.84 μs to 4.58 μs , whereas the execution time for the CWM operation decreases from 2.62 μs to 0.93 μs . However, the improved performance is accompanied by increased switching activity, causing the power consumption to rise from 43 mW at 100 MHz to 133 mW at 280 MHz.

For the pipelined architecture, the Poly-Mul operation requires 4114 clock cycles across all operating frequencies. Within the low-frequency operating range from 10 MHz to 40 MHz, the Poly-Mul execution time decreases from 411.40 μs to 102.85 μs . Similarly, in the high-frequency operating range from 100 MHz to 280 MHz, the execution time further decreases from 41.14 μs to 14.69 μs .

4.2. Area–Time Product (ATP) and Energy–Delay Product (EDP) Evaluations

To further analyze the proposed non-pipelined and 4-stage pipelined architectures, the Area–Time Product and Energy–Delay Product are evaluated in Table 2. The ATP metric reflects the combined effect of hardware cost and computation time, whereas the EDP metric captures the joint impact of energy consumption and execution delay. The ATP is calculated using Equation (6), where the slice-equivalent cost (SEC) is used as the normalized area metric and is calculated using Equation (7). The average computation time is calculated using the mean execution times of the FNTT, INTT, and CWM operations and is expressed mathematically in Equation (8). Since the SEC is represented in slices and the execution time is measured in μs , the resulting ATP unit is expressed as slice- μs .

$$\text{ATP} = \text{SEC} \times T_{\text{avg}} \quad (6)$$

$$\text{SEC} = \text{Slices} + (100 \times \text{DSPs}) + (196 \times \text{BRAMs}) \quad (7)$$

$$T_{\text{avg}} = \frac{T_{\text{FNTT}} + T_{\text{INTT}} + T_{\text{CWM}}}{3} \quad (8)$$

Similarly, for EDP, the average consumed energy is initially computed using Equation (9), where the average power consumption obtained from Table 1 is multiplied by the average computation time T_{avg} . Since the average power is reported in mW and the execution time is expressed in μs , the resulting energy is represented in nJ. Thereafter, the EDP is computed using Equation (10). Because the EDP is obtained from the product of energy (nJ) and time (μs), its unit is expressed as nJ- μs .

$$E_{\text{avg}} = P_{\text{avg}} \times T_{\text{avg}} \quad (9)$$

$$\text{EDP} = E_{\text{avg}} \times T_{\text{avg}} \quad (10)$$

Table 2 summarizes the ATP and EDP evaluations for both the non-pipelined and 4-stage pipelined architectures across different operating frequencies. For both ATP and EDP metrics, lower values indicate higher hardware efficiency because they represent reduced area–time and energy–delay overheads, respectively. The results demonstrate that increasing the operating frequency significantly reduces the average computation time, thereby improving the ATP performance for both architectures. For the non-pipelined design, the ATP decreases from 192.37×10^3 Slice- μs at 10 MHz to 8.46×10^3 Slice- μs at 250 MHz. Similarly, the pipelined architecture achieves a reduction in ATP from 195.36×10^3 Slice- μs at 10 MHz to 7.74×10^3 Slice- μs at 280 MHz. A comparable trend is observed for the EDP metric, where the higher-frequency implementations achieve lower energy–delay costs despite the increase in average power consumption. Although the pipelined architecture introduces additional registers and slightly increases the SEC and switching activity, it provides superior timing scalability and achieves the best overall ATP and EDP results at 280 MHz. In contrast, the non-pipelined implementation re-

mains more attractive for lightweight and low-power cryptographic systems operating at moderate frequencies.

Table 2. ATP & EDP evaluation of our non-pipelined and 4-stage pipelined architectures.

Design	Freq. (MHz)	SEC	T _{avg} (μs)	E _{avg} (nJ)	ATP (×10 ³ Slice-μs)	EDP (nJ-μs)
Non-pipelined	10	2048	93.93	281.79	192.37	26,469.43
	20	2058	46.97	281.82	96.66	13,236.07
	30	2079	31.31	313.10	65.09	9801.16
	40	2091	23.48	446.12	49.10	10,475.30
	100	2123	9.39	384.99	19.94	3615.06
	150	2160	6.26	394.38	13.52	2468.82
	200	2202	4.70	408.90	10.35	1921.83
	250	2250	3.76	421.12	8.46	1582.02
4-stage pipelined	10	2071	94.33	282.99	195.36	26,690.25
	20	2081	47.17	283.02	98.15	13,348.25
	30	2102	31.44	345.84	66.09	10,874.89
	40	2114	23.58	471.60	49.85	11,120.33
	100	2146	9.43	405.49	20.24	3823.85
	150	2183	6.29	415.14	13.73	2611.23
	200	2225	4.72	429.52	10.50	2027.29
	250	2273	3.78	446.04	8.59	1684.53
	280	2303	3.36	446.88	7.74	1501.52

Overall trade-offs: In general, the implementation results reveal a clear trade-off between hardware compactness, power efficiency, and timing performance for the proposed non-pipelined and 4-stage pipelined architectures. The non-pipelined design achieves lower hardware utilization and reduced switching activity by eliminating intermediate pipeline registers, resulting in improved area efficiency and lower power consumption across all operating frequencies. In contrast, the 4-stage pipelined architecture introduces additional registers within the butterfly and Barrett reduction datapaths, slightly increasing the slice-equivalent cost and power consumption. However, the shorter critical path enables significantly improved timing scalability and allows the architecture to operate successfully up to 280 MHz. The ATP and EDP evaluations further demonstrate this trade-off. Although the pipelined design incurs additional hardware and energy overheads at lower frequencies, it achieves superior ATP and EDP values at higher operating frequencies because the reduction in computation time outweighs the increase in area and power consumption. Therefore, the non-pipelined architecture provides a favorable solution for low-power IoT and RFID-oriented cryptographic applications, whereas the pipelined architecture is better suited for high-throughput systems requiring enhanced timing performance and computational efficiency.

4.3. Comparison with State-of-the-Art Accelerators

Table 3 compares the proposed pipelined NTT accelerator with closely related CRYSTALS-Dilithium NTT implementations on the Artix-7 FPGA using $N = 256$ and $q = 8380417$. For a fair area comparison, the SEC is calculated using Equation (7). For studies where slices are not directly reported, the slice count is converted from LUTs and FFs following the method used in [36]. This states that $1 \text{ slice} = (\frac{\text{LUTs}}{4}) + (\frac{\text{FFs}}{8})$ is for the Artix-7 FPGA. The ATP is calculated using Equation (6). When CWM timing is not reported in a reference design, the average computation time (T_{avg}) is calculated using only the available

FNTT and INTT execution times. When FNTT, INTT, and CWM timings are all available, all three values are included in the average computation time. It is important to note that a direct comparison of power- and energy-related metrics with state-of-the-art NTT accelerators is not presented because power consumption results are unavailable in many published reference works. Therefore, the power and energy measurements reported in Sections 4.1 and 4.2 are intended to provide additional insight into the trade-offs among area, performance, and energy consumption of the proposed architectures. These measurements are used to evaluate and compare the energy efficiency of the proposed non-pipelined and pipelined architectures rather than to claim energy superiority over prior implementations.

Table 3. Comparison with state-of-the-art CRYSTALS-Dilithium NTT designs on Artix-7 FPGA. All designs use $N = 256$ and $q = 8380417$. BU denotes butterfly units, BRM denotes BRAMs, and $^+$ indicates estimated slices computed as $\frac{\text{LUTs}}{4} + \frac{\text{FFs}}{8}$. TW indicates this work. Lower ATP indicates better area–time efficiency.

Ref.	BU	Slices	LUTs	FFs	DSP	BRM	Clock Cycles FNTT/INTT/CWM	Freq. (MHz)	Time (μ s) FNTT/INTT/CWM	SEC	ATP ($\times 10^3$)
Dedicated CRYSTALS-Dilithium implementations											
[21]	2	944 ⁺	2759	2037	4	7	606/614/–	163	3.71/3.76/–	2716	10.14
[22]	2	226 ⁺	524	759	17	1	533/536/–	311	1.71/1.72/–	2122	3.64
[23]	1	942	3101	655	6	3	1280/1280/258	250	5.12/5.12/1.03	2130	8.01
[24]	1	534	1557	1652	6	5	271/271/–	158	1.71/1.71/–	2114	3.61
[25]	8	1595 ⁺	4187	4387	18	0	286/286/–	297	0.96/0.96/–	3395	3.26
Unified CRYSTALS-Kyber and CRYSTALS-Dilithium implementations											
[28]	1	397 ⁺	1302	571	9	3	1074/1074/264	131	8.19/8.19/2.01	1885	11.56
[29] ^(R2)	8	2456	7451	5275	0	0	128/128/–	180	0.71/0.71/–	2456	1.74
[29] ^(R4)	16	4522	13,804	11,019	0	0	64/64/–	163	0.39/0.39/–	4522	1.76
TW	1	1115	3438	711	6	3	1284/1284/262	280	4.58/4.58/0.93	2303	7.74

^(R2) and ^(R4) for reference [29] denote that the architectures use radix-2 and radix-4 butterfly units, respectively.

4.3.1. Comparison with Dedicated CRYSTALS-Dilithium Implementations [21–25] Lightweight Architectures Supporting Only Single Butterfly Unit [23,24]

The non-pipelined implementation reported in [23] employs a single butterfly unit without pipelining, resulting in a slightly lower SEC compared with the proposed work. In contrast, the proposed architecture incorporates four pipeline stages to improve timing performance while maintaining the same computational functionality. Consequently, the maximum operating frequency increases from 250 MHz in [23] to 280 MHz in the proposed design, while the ATP improves from 8.01 to 7.74. These results indicate that the introduced pipelining enhances timing closure and overall area–time efficiency with only a marginal increase in hardware cost. The NTT accelerator reported in [24] achieves a lower ATP of 3.61 owing to reduced execution latency and optimized transform scheduling. However, the proposed architecture maintains comparable SEC while additionally supporting unified computation of FNTT, INTT, and CWM operations within a single reusable butterfly datapath, whereas the reference design does not provide support for CWM computation.

NTT Designs Using Two Butterfly Units [21,22]

Compared with the accelerator in [21], the proposed architecture reduces the SEC from 2716 to 2303, corresponding to approximately a $1.18\times$ lower hardware cost. This reduction is mainly achieved by using a single unified butterfly unit and extensive resource sharing across arithmetic operations. Although the design in [21] achieves lower execution latency by employing two butterfly units and parallel processing, the proposed architecture offers

greater area compactness while supporting FNTT, INTT, and CWM computations on the same datapath. In contrast, the reference design focuses only on FNTT and INTT operations and does not provide support for CWM computation. Moreover, despite utilizing fewer hardware resources and additional functionality, the proposed design improves the ATP from 10.14 to 7.74, achieving approximately $1.31\times$ better ATP efficiency, which indicates a more favorable trade-off between area and computation time.

In comparison with [22], the proposed architecture exhibits a slightly higher SEC (2303 versus 2122). This difference arises primarily because the proposed architecture incorporates dedicated support for FNTT, INTT, and CWM computations within a unified datapath, whereas the reference design focuses only on FNTT and INTT operations. Although the design in [22] attains lower execution latency and higher operating frequency by employing two butterfly units at 311 MHz, the proposed architecture maintains support for FNTT, INTT, and CWM computations while requiring only a single butterfly unit. Furthermore, the reference design does not support CWM computation, resulting in overall ATP efficiency compared to the proposed work.

NTT Accelerator Using Eight Butterfly Units for Parallelism [25]

The architecture presented in [25] achieves a lower ATP of 3.26, which is approximately $2.37\times$ better than the proposed work. This improvement is due to the use of eight butterfly units in the reference design, while the proposed architecture employs only one butterfly unit. Nevertheless, in terms of SEC alone, the proposed architecture demonstrates improved hardware efficiency and a substantially smaller implementation cost.

4.3.2. Comparison with Unified CRYSTALS-Kyber and CRYSTALS-Dilithium Designs Accelerator Using Single Butterfly Unit [25]

Table 3 further compares the proposed architecture with unified accelerators supporting both the CRYSTALS-Kyber and CRYSTALS-Dilithium schemes. Compared with CRYPTOR [28], the proposed architecture exhibits a higher SEC (2303 versus 1885) while improving ATP from 11.56 to 7.74, achieving nearly $1.49\times$ better ATP efficiency. The lower ATP is achieved through the higher operating frequency and improved timing scalability of the proposed pipelined architecture. These improvements are expected since the proposed design is dedicated to CRYSTALS-Dilithium, whereas the reference architecture supports both CRYSTALS PQC algorithms.

Design Supporting Multiple Butterfly Units (Eight and Sixteen [29])

The radix-2 (R_2) and radix-4 (R_4) architectures reported in [29] achieve lower execution latency and ATP values by employing 8 and 16 butterfly units, respectively, thereby exploiting parallelism to maximize throughput. However, such parallel implementations incur increased hardware complexity and are primarily optimized for high-performance applications. Similarly, the unified accelerators proposed in [30,31], although not included in Table 3, emphasize maximum throughput while supporting multiple PQC algorithms. The work in [30] targets SABER and CRYSTALS-Dilithium NTT implementations, whereas [31] supports both CRYSTALS-Kyber and CRYSTALS-Dilithium. These architectures focus on resource reuse across multiple PQC schemes through pipelining and architectural parallelism. In contrast, the proposed architecture prioritizes lightweight implementation, competitive SEC, and improved ATP while maintaining support for FNTT, INTT, and CWM computations within a single reusable datapath. Consequently, the proposed design provides a favorable trade-off for resource-constrained FPGA platforms, where limited hardware resource utilization is more critical than maximum throughput.

It is important to note that the primary objective of this work was to investigate the area–power–performance trade-offs between lightweight non-pipelined and pipelined

NTT architectures. Consequently, the proposed architectures do not incorporate dedicated side-channel countermeasures, and no side-channel security evaluation (e.g., TVLA, CPA, DPA, or electromagnetic analysis) was performed. Nevertheless, for a fixed transform size, both the non-pipelined and pipelined architectures execute using predetermined operation schedules and fixed clock-cycle counts, thereby avoiding data-dependent timing variations.

5. Conclusions and Future Directions

This article investigates area–power–performance trade-offs in lightweight NTT-based polynomial multiplication accelerators targeting CRYSTALS-Dilithium parameters through two architectural variants: non-pipelined and 4-stage pipelined implementations. Both architectures employ a single butterfly unit supporting Cooley–Tukey and Gentleman–Sande configurations for FNTT, INTT, and CWM computations using shared arithmetic resources. The non-pipelined architecture prioritizes reduced hardware utilization and switching activity, whereas pipelining the modular multiplication and Barrett reduction datapaths improves operating frequency. Implementation and evaluation on an Artix-7 FPGA across operating frequencies ranging from 10 MHz to 280 MHz reveal a clear trade-off between hardware compactness and timing performance. The non-pipelined architecture is more suitable for lightweight and energy-constrained applications, while the pipelined architecture incurs moderate increases in hardware utilization and power consumption but significantly improves timing scalability, enabling successful operation at a maximum of 280 MHz. ATP and EDP evaluations further show that the reduction in execution time at higher frequencies outweighs the additional hardware costs introduced by pipelining, resulting in improved overall efficiency within the proposed lightweight single-butterfly architecture. Comparisons with state-of-the-art CRYSTALS-Dilithium accelerators indicate that the proposed pipelined architecture provides a favorable trade-off between hardware efficiency and computational performance while maintaining support for FNTT, INTT, and CWM computations within a single reusable butterfly datapath. Although several reference designs achieve lower latency and superior ATP through increased hardware parallelism and the use of multiple butterfly units, the proposed architecture remains attractive for resource-constrained FPGA platforms due to its lightweight implementation and low hardware cost. Future work will investigate alternative modular reduction approaches, including Montgomery and Plantard reduction schemes, as well as CRYSTALS-Dilithium-specific arithmetic optimizations based on the property $2^{23} \equiv 2^{13} - 1$, to further improve area and energy efficiency.

Author Contributions: Conceptualization, O.S.S. and M.R.; Methodology, O.S.S., M.R., K.M.K. and H.H.S.; Software, O.S.S. and M.R.; Validation, M.R., K.M.K. and H.H.S.; Formal analysis, O.S.S. and M.R.; Resources, K.M.K.; Data curation, K.M.K. and H.H.S.; Writing—original draft, O.S.S. and M.R.; Writing—review & editing, M.R.; Visualization, O.S.S. and M.R.; Supervision, M.R. and K.M.K.; Project administration, O.S.S. and M.R.; Funding acquisition, O.S.S. and M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Umm Al-Qura University, Saudi Arabia under grant number: 26UQU4320199GSSR02.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors extend their appreciation to Umm Al-Qura University, Saudi Arabia for funding this research work through grant number: 26UQU4320199GSSR02.

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

References

1. Avanzi, R.; Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (Version 3.01). Available online: <https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf> (accessed on 11 July 2024).
2. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation. Available online: <https://pq-crystals.org/dilithium/data/dilithium-specification.pdf> (accessed on 11 July 2024).
3. NIST. Post-Quantum Cryptography (Round-1): Additional Digital Signature Schemes. Available online: <https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures> (accessed on 26 June 2024).
4. Sonbul, O.S.; Rashid, M.; Jaffar, A.Y.; Alghamdi, M.J.; Masud, M.I.; Aman, M. Optimizing Arithmetic Resource Allocation in Configurable NTT Hardware for Area-Constrained Homomorphic Encryption. *Results Eng.* **2026**, *31*, 111270. [\[CrossRef\]](#)
5. Sonbul, O.S.; Rashid, M.; Jaffar, A.Y. Accelerating CRYSTALS-Kyber: High-Speed NTT Design with Optimized Pipelining and Modular Reduction. *Electronics* **2025**, *14*, 2122. [\[CrossRef\]](#)
6. Satriawan, A.; Mareta, R.; Lee, H. A Complete Beginner Guide to the Number Theoretic Transform (NTT). *IEEE Access* **2024**, *11*, 70288–70316. [\[CrossRef\]](#)
7. Zhang, N.; Yang, B.; Chen, C.; Yin, S.; Wei, S.; Liu, L. Highly Efficient Architecture of NewHope-NIST on FPGA using Low-Complexity NTT/INTT. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 49–72. [\[CrossRef\]](#)
8. Kundi, D.E.S.; Zhang, Y.; Wang, C.; Khalid, A.; O'Neill, M.; Liu, W. Ultra High-Speed Polynomial Multiplications for Lattice-Based Cryptography on FPGAs. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 1993–2005. [\[CrossRef\]](#)
9. Derya, K.; Mert, A.C.; Öztürk, E.; Savaş, E. CoHA-NTT: A Configurable Hardware Accelerator for NTT-based Polynomial Multiplication. *Microprocess. Microsyst.* **2022**, *89*, 104451. [\[CrossRef\]](#)
10. Tan, W.; Wang, A.; Lao, Y.; Zhang, X.; Parhi, K.K. Pipelined High-Throughput NTT Architecture for Lattice-Based Cryptography. In *Proceedings of the 2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*; IEEE: New York, NY, USA, 2021; pp. 1–4. [\[CrossRef\]](#)
11. Kundi, D.-E.-S.; Khalid, A.; Bian, S.; Wang, C.; O'Neill, M.; Liu, W. AxRLWE: A multilevel approximate ring-LWE co-processor for lightweight IoT applications. *IEEE Internet Things J.* **2021**, *9*, 10492–10501.
12. Hua, S.; Zhang, H.; Zhang, J.; Wang, S. Optimization and implementation of the number theoretic transform butterfly unit for large integer multiplication. *J. Inf. Secur. Appl.* **2021**, *59*, 102857. [\[CrossRef\]](#)
13. Yang, Y.; Kuppannagari, S.R.; Kannan, R.; Prasanna, V.K. NTTGen: A framework for generating low latency NTT implementations on FPGA. In *Proceedings of the 19th ACM International Conference on Computing Frontiers; CF '22*; ACM: New York, NY, USA, 2022; pp. 30–39. [\[CrossRef\]](#)
14. Zaman, M.A.; Bilal Khan, M.; Ahmad, W.; Khalid, A.; O'Neill, M. A Lightweight and Hardware-Efficient NTT FPGA Accelerator for FHE Applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2026**, *73*, 78–82. [\[CrossRef\]](#)
15. Javeed, K.; Gregg, D. AT-NTT: Area-time efficient polynomial multiplication architecture for CRYSTALS-Kyber. *Comput. Electr. Eng.* **2026**, *133*, 111069. [\[CrossRef\]](#)
16. Ni, Z.; Khalid, A.; Liu, W.; O'Neill, M. Towards a lightweight CRYSTALS-Kyber in FPGAs: An ultra-lightweight BRAM-free NTT core. In *Proceedings of the 2023 IEEE International Symposium on Circuits and Systems (ISCAS)*; IEEE: New York, NY, USA, 2023; pp. 1–5. [\[CrossRef\]](#)
17. Imran, M.; Khan, S.; Khalid, A.; Rafferty, C.; Shah, Y.A.; Pagliarini, S.; Rashid, M.; O'Neill, M. Evaluating NTT/INTT Implementation Styles for Post-Quantum Cryptography. *IEEE Embed. Syst. Lett.* **2024**, *16*, 485–488. [\[CrossRef\]](#)
18. 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. [\[CrossRef\]](#)
19. Matteo, S.D.; Valea, E. Evaluating Communication and Architectural Overheads in NTT Accelerators for ML-KEM. In *Proceedings of the 2026 IEEE 29th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*; IEEE: New York, NY, USA, 2026; pp. 1–6. [\[CrossRef\]](#)
20. Matteo, S.D.; Sarno, I.; Valea, E.; Saponara, S. KEM-22: An Efficient Post-Quantum ML-KEM Hardware Accelerator on 22-nm ASIC. *IEEE Internet Things J.* **2026**, *13*, 4715–4726. [\[CrossRef\]](#)
21. Gupta, N.; Jati, A.; Chattopadhyay, A.; Jha, G. Lightweight Hardware Accelerator for Post-Quantum Digital Signature CRYSTALS-Dilithium. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 3234–3243. [\[CrossRef\]](#)
22. Land, G.; Sasdrich, P.; Güneysu, T. A Hard Crystal—Implementing Dilithium on Reconfigurable Hardware. In *Proceedings of the Smart Card Research and Advanced Applications; Grosso, V., Pöppelmann, T., Eds.; Springer: Cham, Switzerland, 2022; pp. 210–230.*

23. Imran, M.; Khalid, A.; Rafferty, C.; Rashid, M.; O'Neill, M. Evaluating Area-Time-Power Trade-offs in NTT Accelerators for Post-Quantum Cryptography. In *Proceedings of the 2025 IEEE 38th International System-on-Chip Conference (SOCC)*; IEEE: New York, NY, USA, 2025; pp. 1–6. [\[CrossRef\]](#)
24. Tu, Y.; Xie, J. EMINEM: Efficient FPGA Implementation of Mixed-Radix NTT Hardware Accelerators for NIST Post-Quantum Cryptography Falcon, Dilithium, and HAWK. *ACM Trans. Reconfig. Technol. Syst.* **2025**, *18*, 1–25. [\[CrossRef\]](#)
25. 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. [\[CrossRef\]](#)
26. Gupta, H.; Goyal, A.; Dhadwal, P.; Gandhi, J.; Shekhawat, D.; Santosh, M.; Pandey, J.G. A Low-overhead Dilithium-NTT Architecture using Accelerated K-RED Modular Reduction Unit. In *Proceedings of the 2026 27th International Symposium on Quality Electronic Design (ISQED)*; IEEE: New York, NY, USA, 2026; pp. 1–6. [\[CrossRef\]](#)
27. Ni, Z.; Khalid, A.; Zhang, Z.; Cui, Y.; Liu, W.; O'Neill, M. LightHD: A Lightweight and High-Performance Hardware Accelerator of CRYSTALS-Dilithium. *IEEE Trans. Comput.* **2026**, *75*, 2007–2019. [\[CrossRef\]](#)
28. Matteo, S.D.; Sarno, I.; Saponara, S. CRYPTOR: A Memory-Unified NTT-Based Hardware Accelerator for Post-Quantum CRYSTALS Algorithms. *IEEE Access* **2024**, *12*, 25501–25511. [\[CrossRef\]](#)
29. 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. [\[CrossRef\]](#)
30. Aikata, A.; Mert, A.C.; Jacquemin, D.; Das, A.; Matthews, D.; Ghosh, S.; Roy, S.S. A Unified Cryptoprocessor for Lattice-Based Signature and Key-Exchange. *IEEE Trans. Comput.* **2023**, *72*, 1568–1580. [\[CrossRef\]](#)
31. Aikata, A.; Mert, A.C.; Imran, M.; Pagliarini, S.; Roy, S.S. KaLi: A Crystal for Post-Quantum Security Using Kyber and Dilithium. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 747–758. [\[CrossRef\]](#)
32. Imran, M.; Abideen, Z.U.; Pagliarini, S. An Open-source Library of Large Integer Polynomial Multipliers. In *Proceedings of the 2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*; IEEE: New York, NY, USA, 2021; pp. 145–150. [\[CrossRef\]](#)
33. Imran, M.; Abideen, Z.U.; Pagliarini, S. A Versatile and Flexible Multiplier Generator for Large Integer Polynomials. *J. Hardw. Syst. Secur.* **2023**, *7*, 55–71. [\[CrossRef\]](#)
34. Aksoy, L.; Roy, D.B.; Imran, M.; Karl, P.; Pagliarini, S. Multiplierless Design of Very Large Constant Multiplications in Cryptography. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 4503–4507. [\[CrossRef\]](#)
35. 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. [\[CrossRef\]](#)
36. Ni, Z.; Khalid, A.; Kundi, D.E.S.; O'Neill, M.; Liu, W. HPKA: A High-Performance CRYSTALS-Kyber Accelerator Exploring Efficient Pipelining. *IEEE Trans. Comput.* **2023**, *72*, 3340–3353. [\[CrossRef\]](#)

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.