An Efficient Number Theoretic Transform Implementation for FIPS-203 on FPGA

Sherif Elewa
Department of Electrical and Computer Engineering
The Ohio State University
elewa.3@osu.edu

Eslam Tawfik
Department of Electrical and Computer Engineering
The Ohio State University
tawfik.10@osu.edu

2025 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-5683-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISCAS56072.2025.11043425

***Abstract*—The Number Theoretic Transform (NTT) is a key component in modern Post-Quantum Cryptography (PQC) systems, known for its efficient polynomial multiplication capabilities. This paper presents an innovative conflict-free NTT memory architecture that features a mathematically optimized Barrett-based modular reduction with bit correction, tailored for FIPS-203. The primary goal is to significantly reduce hardware resource usage while maintaining high performance and superior efficiency (frequency of operation/area) compared to prior work. Our NTT design, implemented on a Xilinx FPGA Virtex-7, achieves a clock speed of 300 MHz, to the best of our knowledge, the highest reported in the field, outperforming the best-known implementation by 15%. Additionally, it utilizes a single RAMB, one DSP block, 374 LUTs, and 270 registers, making it the most resource-efficient design in the literature, with less than half the hardware usage of prior work. Compared to the leading designs, the proposed architecture reduces LUT utilization by 38%, register usage by 58%, DSP utilization by 50%, and RAMB usage by 75%. As a result, the overall efficiency measured by the area-time product (ATP) surpasses the best design in the literature by a factor of 1.38. These results establish our approach as an optimal solution for applications that require minimal resources and low power consumption without compromising performance.**

***Index Terms*—Post Quantum Cryptography, CRYSTALS-Kyber, FIPS-203, NTT, Modular Multiplication, Barret Modular Reduction.**

I. Introduction

The rapidly evolving landscape of quantum computing introduces a threat to classic cryptography, particularly after Shor's algorithm [1]. Therefore, post-quantum cryptography (PQC) has become critical to protect data security from future quantum threats. Recently, CRYSTALS-Kyber has been chosen to be the standard Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) under standard FIPS-203. Since FIPS-203 is built over lattice-based cryptography (LBC), it involves extensive polynomial multiplication which poses significant challenges for computational performance and hardware design. To address these complexities, the Number Theoretic Transform (NTT) is widely used in PQC systems for its efficiency in reducing the computational burden of polynomial multiplication. Although NTT helps reduce polynomial multiplication complexity, it still acts as the primary performance bottleneck in the system, particularly, due to the iterative nature of the NTT algorithm that introduces challenges with memory access conflict. Therefore, numerous studies have investigated this issue, yet all attempted to architect it using a common approach, which is using multiple memory instances to resolve the conflict, that ended up utilizing more hardware resources [2]–[13]. Moreover, in PQC, all operations are modular arithmetic which adds a level of complexity to hardware design. Specifically, for modular multiplication, the most commonly used algorithms are Montogomory [14] and Barrett [15]. This work will focus on Barrett modular multiplication. Multiple investigations have been conducted to reduce the complexity of the modular multiplier utilized for PQC [3], [16]–[23]. However, the hardware complexity remains a prominent concern, leaving a room for further improvements.

This work will present an efficient yet area-optimized implementation for NTT/inverse-NTT (iNTT), especially for FIPS-203. The two key features of the proposed architecture are 1) efficient memory architecture with conflict-free addressing utilizing a single memory module while maintaining optimal performance., and 2) a low-complexity hardware design for optimized Barrett modular reduction.

The rest of the paper is organized as follows. Section II explains the NTT theory and reviews the related work done for NTT, focusing on FIPS-203. Section III presents the proposed NTT/iNTT architecture for FIPS-203 PQC. Section IV discusses the experimental results of the implemented hardware on Xilinx FPGA.

II. Theory and Related Work

A. Number Theoretic Transform

FIPS-203 operations are performed over an integer ring of the polynomial modulo q represented by $R_q = \mathbb{Z}_q[x]/x^n + 1$, where $\mathbb{Z}_q$ is finite integer field modulo q and $x^n + 1$ is the n-th cyclotomic polynomial. The multiplication operation of polynomials poses a significant challenge in hardware design compared to other operations as the cost increases exponentially. Within the context of FIPS-203, the cost of multiplying two polynomials of degree $n = 256$ using the conventional polynomial multiplication (without NTT) will be $O(n^2)$ ($256^2 = 65536$ multiplication operation), which is a very complex and costly operation. Therefore, mathematicians have put some effort into developing algorithms that help speed up the polynomial multiplication operation. One of the prominent algorithms is the NTT which reduces the cost of multiplying two polynomials to $O((3 * n \log n) + n)$ as will be explained shortly. NTT serves as an extension

of the well-known Fast Fourier Transform (FFT), with all arithmetic operations conducted within a finite field instead of complex numbers [24]. Specifically, the NTT takes a set of n-coefficient polynomials as input and generates an n-element vector representing the input in the NTT domain (this costs $n \log n$ multiplication operation for each polynomial). Polynomial multiplication can then be performed through point-wise multiplication within the NTT domain (this costs n multiplication operation). Finally, the inverse NTT is applied to the final result to restore the coefficients from the NTT domain (this costs $n \log n$ multiplication operation). Consequently, the multiplication of two polynomials, "a" and "b" of degree n using NTT can be accomplished as follows

$$c = a \cdot b = INTT_n(NTT_n(a) \odot NTT_n(b)) \tag{1}$$

where $\odot$ denotes the element-wise multiplication.

From mathematics point of view, a polynomial $a(x)$ with coefficients $a(x) = (a_0, a_1, a_2, \ldots, a_{n-1})$ has the NTT representation $\hat{a}(x) = (\hat{a}_0, \hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{n-1})$ as the following equation

$$\hat{a}_i = \sum_{j=0}^{n-1} a_j \omega_n^{ij} \; mod \, q, \quad i \in [0, n-1] \tag{2}$$

where ω_n is the n^{th} root of unity such that $\omega_n^n = 1 \; mod \, q$. On the other side, the INTT operation converts $\hat{a}(x)$ back to $a(x)$ as follows

$$a_i = (1/n) \sum_{j=0}^{n-1} \hat{a}_j \omega_n^{-ij} \; mod \, q, \quad i \in [0, n-1] \tag{3}$$

The NTT operation is described as Algorithm 1 where $br()$ denotes the bit reverse operation.

Algorithm 1 FIPS-203 NTT Algorithm

Input: Polynomial coefficients $a = a_0, a_1, ..., a_{n-1}$, $n = 256$
Input: modulus $q = 3329$, n^{th} root of unity ω_n
Output: $\hat{a} = NTT(a)$

$step = 1$
$len = 128$
for $len \geq 2$ **do**
  $start = 0$
  for $start < 256$ **do**
    $tf = \omega_n^{br(step)} \; mod \, q$
    $j = start$
    for $j < (start + len)$ **do**
      $t = (a_{j+len} * tf) \; mod \, q$
      $a_{j+len} = (a_j - t) \; mod \, q$
      $a_j = (a_j + t) \; mod \, q$
      $j = j + 1$
    end for
    $start = j + len$
    $step = step + 1$
  end for
  $len = len/2$
end for
return $\hat{a} = a$

B. Hardware Implementation Comprehensive Review

This section will present a comprehensive literature review of various NTT implementations focusing on FIPS-203. A configurable crypto-processor is presented in [12] with configurable prime modulus q allowing their architecture to be suitable for various PQC algorithms including FIPS-203. Another hardware implementation for NTT in FIPS-203 is presented in [2] which utilizes a single BRAM. However, the authors mentioned that it takes 2 cycles to read/write the data which degraded the performance to half compared to read/write in a single cycle. In [5], the authors introduce three different hardware architectures for NTT based on the modular reduction algorithm presented in [25]. Despite the variety of implementations, The primary performance bottleneck is the recursive nature of the modular reduction algorithm. A modified Barrett modular multiplier is presented in [3] by calculating an approximate quotient, and then correcting it using a series of comparisons to identify the correct reminder. Due to the long series of consecutive comparisons and switches, which create a critical path, the frequency of this method is as low as 161 MHz. A configurable hardware accelerator for NTT in PQC is presented in [6] using configurable Montgomery modular reduction. The authors claim double improvement in the performance compared to [12]. The works in [13], [23] introduce high-speed NTT designs, however, they utilize multiple DSP blocks to achieve this performance. The design in [4] presents a new modular reduction for FIPS-203, however, consecutive summations and comparisons create a critical path that ultimately degrades the performance. The authors in [11] introduce a scalable NTT architecture that prioritizes flexibility by utilizing additional hardware resources.

III. NTT Architecture

A. Overall NTT Architecture

The architecture for the NTT/iNTT engine, as shown in Fig.1, consists of data memory which stores the values of the polynomial coefficients, a Butterfly unit (BFU) which is the modular arithmetic calculation engine, ROM that stores the Twiddle factor (TF) constant values, and a control unit that has a state machine controlling the whole engine. Starting with the control path flow, the control unit takes the "mode" signal as input which indicates the mode of operation (NTT/iNTT) and the "enable" signal that marks the start of the operation. Once the operation starts the control unit generates the addresses and the enable signals for the data memory and the TF ROM based on the NTT/iNTT stage. Finally, the controller generates a "done" signal marking the end of the operation after passing through all NTT/iNTT stages. For the data path flow, every clock cycle the polynomial coefficients are read from the data memory, and the TF value is read from the ROM. These values are then fed to the BFU to perform the calculations, shuffle the data result, and store it in the data memory. The purpose

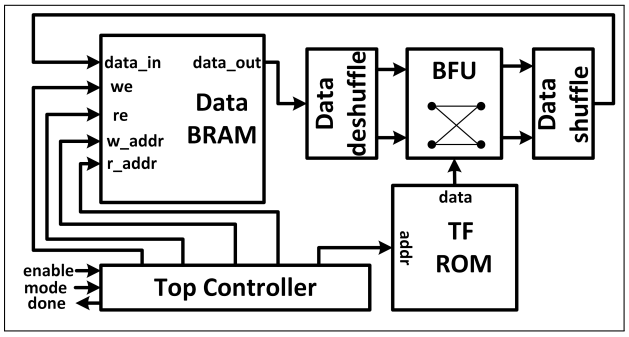

Fig. 1. NTT/iNTT Top Architecture Block Diagram

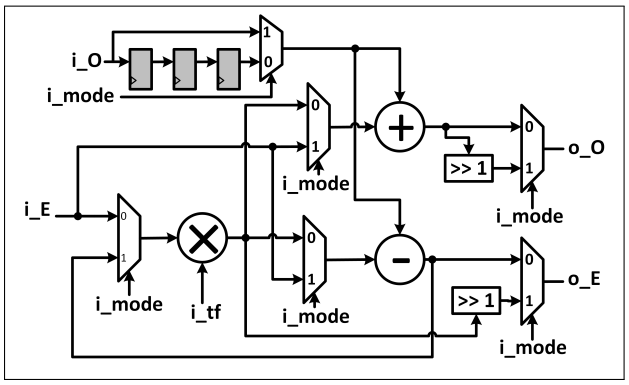

Fig. 2. Butterfly Unit Architecture Block Diagram

of the shuffling/de-shuffling is to resolve the memory conflict, which will be further explained in III-C.

B. Butterfly Unit Design

1) Overall Butterfly Unit Architecture: As shown in Fig.2 The BFU consists of a modular adder, a modular subtractor, and a modular multiplier all performing modulo $q = 3329$ arithmetic operations for FIPS-203. Based on the "mode" signal, the BFU will perform NTT (Cooley-Tukey algorithm) or iNTT (Gentleman-Sande algorithm). The modular multiplier multiplies the TF read from the TF-ROM with the input even polynomial coefficient in the case of NTT, and multiplies it with the output of the modular subtractor in the case of iNTT. The design of the modular adder and subtractor is simply an adder, subtractor, and comparator as shown in Fig.3. Additionally, from (3) there is a division by n at the end of iNTT operation. To perform this, a division by 2 (shift right by 1) is performed on every iNTT calculation (every stage). The design is pipelined to relax the timing on the critical path and achieve high performance.

2) Optimized Modular Multiplier: This research focuses on Barrett arithmetic modular multiplier customized for FIPS-203 ($q = 3329$). In this research, a modified version of the Barrett algorithm is invented by applying mathematical derivations to convert the division to a simple shift operation and then perform a bit-correction mechanism to correct the residue error as shown in Algorithm 2. The main idea is to convert the division by q into a division by 2^{12}, and then

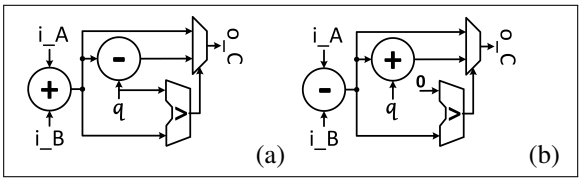

Fig. 3. Modular (a) Adder and (b) Subtractor Block Design

Algorithm 2 The proposed modular reduction with bit-correction for FIPS-203.

Input: c, 24-bits unsigned integer; $q = 3329$ modulus
Output: x 12-bits unsigned, $x = c \; mod \; q$

1: $\hat{m} = \lceil \frac{c}{q} \rceil = \frac{c}{4096}(1 + \frac{1}{4} - \frac{1}{64} - \frac{1}{256}) = (c \gg 12) + (c \gg 14) - (c \gg 18) - (c \gg 20)$
2: $correct = round((c[11:9] + c[13:11] - c[17:15] - c[19:17]) \gg 3)$
3: $m = \hat{m} + correct$
4: $x = c - (q \times m)$
5: **if** $x < 0$ **then**
6: $\quad x = x + q$
7: **end if**
8: **return** x

approximate the multiplication by the constant $2^{12}/3329$ into division by power-of-two. Consequently, there is a small error in the calculated quotient $\hat{m}$ due to the truncation of some bits during the approximation of $2^{12}/3329$. Therefore, a bit correction operation is performed using the most significant 3 truncated bits to calculate their accurate weight. Finally, the computed quotient is corrected to be m and the residue is calculated by subtracting $(q * m)$ from the input number to get the modulo. Finally, the computed residue is checked to add q if negative.

The architecture of the proposed modular multiplier is shown in Fig.4. It consists of 3 pipeline stages to balance the critical path and achieve a frequency of 476 MHz, making it the highest reported in the literature. The first stage computes the quotient $\hat{m}$ and the correction in parallel. The second stage multiplies the final quotient m by constant $q = 3329$. In the final stage, the calculated $q * m$ is subtracted from the input c to calculate the modulus. Then, a final check for the output to add q if negative.

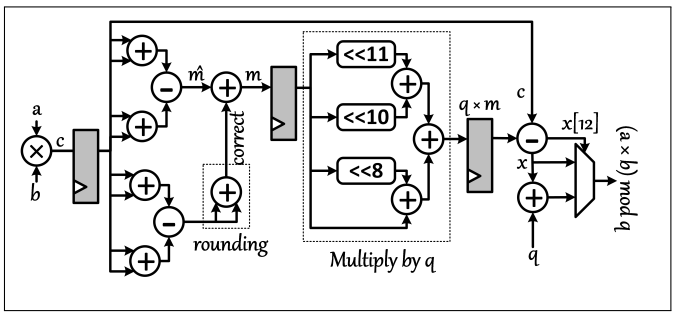

Fig. 4. Modified Barrett Modular Multiplier Block Diagram

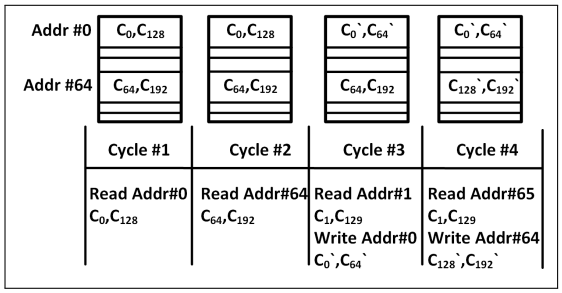

Fig. 5. Memory Scheduling For NTT Operation (First 4 Cycles)

C. Memory Scheduling Optimization

In this research, a conflict-free memory addressing technique is proposed which allows utilizing a single memory (FPGA BRAM) resource while avoiding conflict during read/write operations as shown in Fig.5. The main idea starts with writing the polynomial coefficients C inside the memory in a special way that eases the operation. This way is simply by concatenating the two coefficients that need to be provided simultaneously as inputs to the BFU and storing them in the memory. Consequently, reading a single location from the memory (in a single clock cycle) means the data is ready for BFU calculation. Still, there is a challenge in the data writing because the two outputs from BFU $C^{'}$ will not be used together in the next NTT/iNTT stage. Therefore, to overcome this challenge, the output from BFU is registered for one clock cycle waiting for the next input data to be read from the memory. On the other side, the top controller generates the proper address of this input data so that we read from memory the data that will be used with the registered data in the next stage. Once BFU finishes the calculations the shuffling unit takes care of data shuffling by concatenating the coefficients that will be used together in the next NTT stage. On the control path flow, the top controller generates the write address together with the enable signals, based on the stage, in a fully pipelined fashion as depicted in Fig.5.

IV. Experimental Results and Analysis

A. Modular Multiplier Comparison

The proposed modular multiplier algorithm is synthesized on Xilinx Artix-7 FPGA. As shown in Table I, our design achieved a 2x performance improvement using the same logic compared to the best case reported in [16]. This could be achieved thanks to the careful pipelined design together with the mathematical simplification of the algorithm. Consequently, the efficiency is doubled to 48.8 Kbps/A compared to 23 Kbps/A reported in [16]. More importantly, the proposed algorithm achieved 100% accuracy (0 errors).

B. NTT Architecture Review and Comparison

The proposed top design is synthesized on Xilinx Virtex-7 FPGA. Table II shows the comparison with related work targeting FIPS-203. From a logic utilization viewpoint, our implementation is designed with a single BFU utilizing only one DSP block for the multiplier. Furthermore, a single RAMB36 memory is utilized thanks to the conflict-free memory address architecture. Our NTT engine could achieve a frequency of up to 300 MHz with a total latency of $3\mu s$ for NTT/iNTT operation. In comparison with the related work, our design stands as the lowest hardware utilization with around 50% area reduction compared to [13], [23]. The area-time product (ATP) is calculated as a measure of design efficiency, where a lower ATP indicates higher efficiency. Our design achieves a total ATP of 7.93, outperforming state-of-the-art designs, which report a total ATP value of 10.94 in [23] and 13.78 in [13]. This demonstrates that our architecture achieves higher efficiency considering various trade-offs.

TABLE I
Modular Multiplier Implementation Results and Comparison

Work	Freq (MHz)	LUT/FF	Eff (Kbps/A)[a]	Error
This Work	**476**	**66/51**	**48.8**	**0**
[16]	239	75/50	23.0	8%
[22], [16]	188	83/67	15.0	$> 25\%$
[23], [16]	232	59/70	21.6	-
[3]	161	135/96	8.4	0
[4]	159	142/79	8.6	-

a. Efficiency (Eff.) = (Frequency × No. of bits)/(LUTs + FFs).

TABLE II
NTT Implementation Results and Comparison

Work	LUT/FF/DSP/BRAM	Freq.	Latency	ATP[a]
TW[b]	**374/270/1/1**	**300MHz**	$3\mu s$	**1.12/0.81/3/3**
[5]	948/352/1/2.5	190MHz	$4.76\mu s$	4.5/1.7/4.8/11.9
[6]	2128/1144/8/3	174MHz	$5.3\mu s$	11.3/6/42.4/15.9
[11]	1914/2249/3/3	275MHz	$3.83\mu s$	7.3/8.6/11.5/11.5
[3]	1737/1167/2/3	161MHz	$3.18\mu s$	5.5/3.7/6.4/9.5
[23]	801/717/4/2	222MHz	$1.46\mu s$	1.17/1.05/5.8/2.92
[13]	609/640/2/4	257MHz	$1.9\mu s$	1.16/1.22/3.8/7.6

a. ATP: The area-time product for LUT/FF/DSP/BRAM respectively. A lower ATP value corresponds to higher design efficiency.
b. TW: This Work.

V. Conclusion

In this paper, an efficient architecture was devised for NTT/iNTT targeting FIPS-203. Our innovative memory scheduling technique streamlined data read/write processes to utilize just a single RAMB efficiently. Furthermore, the introduction of a novel modular reduction algorithm has led to a 2x increase in efficiency compared to the state-of-the-art reported in [16]. Our design achieves superior performance when benchmarked against published designs, exhibiting the lowest hardware utilization with almost 50% area reduction compared to existing designs in the literature. Considering both area and latency through the area-time product as a measure of overall efficiency, our architecture scores a 27.5% improvement over [23] and a 42.5% improvement over [13]. This feature makes our architecture well-suited for applications requiring minimal hardware in low-power environments while maintaining an optimal balance between area and performance for maximum efficiency.

REFERENCES

[1] P. W. Shor *et al.*, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. los alamos physics preprint archive," 1995.

[2] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of crystals-kyber pqc algorithm through resource reuse," *IEICE Electronics Express*, vol. 17, no. 17, pp. 20 200 234–20 200 234, 2020.

[3] Y. Xing and S. Li, "A compact hardware implementation of cca-secure key exchange mechanism crystals-kyber on fpga," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 328–356, 2021.

[4] W. Guo, S. Li, and L. Kong, "An efficient implementation of kyber," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1562–1566, 2021.

[5] F. Yaman, A. C. Mert, E. Öztürk, and E. Savaş, "A hardware accelerator for polynomial multiplication operation of crystals-kyber pqc scheme," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1020–1025.

[6] K. Derya, A. C. Mert, E. Öztürk, and E. Savaş, "Coha-ntt: A configurable hardware accelerator for ntt-based polynomial multiplication," *Microprocessors and Microsystems*, vol. 89, p. 104451, 2022.

[7] A. C. Mert, E. Karabulut, E. Öztürk, E. Savaş, and A. Aysu, "An extensive study of flexible design methods for the number theoretic transform," *IEEE Transactions on Computers*, vol. 71, no. 11, pp. 2829–2843, 2022.

[8] W. Guo and S. Li, "Highly-efficient hardware architecture for crystals-kyber with a novel conflict-free memory access pattern," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[9] M. Bisheh-Niasar, D. Lo, A. Parthasarathy, B. Pelton, B. Pillilli, and B. Kelly, "Pqc cloudization: Rapid prototyping of scalable ntt/intt architecture to accelerate kyber," in *2023 IEEE Physical Assurance and Inspection of Electronics (PAINE)*. IEEE, 2023, pp. 1–7.

[10] Y. Zhu, W. Zhu, Y. Ouyang, J. Sun, M. Zhu, Q. Zhao, J. Yang, C. Chen, Q. Tao, G. Yang *et al.*, "16.2 a 28nm 69.4 kops 4.4 μj/op versatile post-quantum crypto-processor across multiple mathematical problems," in *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 298–300.

[11] B. Li, Y. Yan, Y. Wei, and H. Han, "Scalable and parallel optimization of the number theoretic transform based on fpga," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2024.

[12] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan, "Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 4, p. 17–61, Aug. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8344

[13] C. Zhang, D. Liu, X. Liu, X. Zou, G. Niu, B. Liu, and Q. Jiang, "Towards efficient hardware implementation of ntt for kyber on fpgas," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[14] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[15] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," in *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 311–323.

[16] T.-H. Nguyen, C.-K. Pham, and T.-T. Hoang, "A high-speed barrett-based modular multiplication with bit-correction for the crystal-kyber cryptosystem," in *International Conference on Intelligence of Things*. Springer, 2023, pp. 191–199.

[17] T. X. Pham, P. Duong-Ngoc, and H. Lee, "An efficient unified polynomial arithmetic unit for crystals-dilithium," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[18] R. Müller, W. Meier, and C. F. Wildfeuer, "Area efficient modular reduction in hardware for arbitrary static moduli," *arXiv preprint arXiv:2308.15079*, 2023.

[19] P. Duong-Ngoc and H. Lee, "Configurable mixed-radix number theoretic transform architecture for lattice-based cryptography," *IEEE Access*, vol. 10, pp. 12 732–12 741, 2022.

[20] M. Knezevic, F. Vercauteren, and I. Verbauwhede, "Faster interleaved modular multiplication based on barrett and montgomery reduction methods," *IEEE Transactions on Computers*, vol. 59, no. 12, pp. 1715–1721, 2010.

[21] D. I. Maulana, W. Jung *et al.*, "Kyber accelerator on fpga using energy-efficient lut-based barrett reduction," in *2022 19th International SoC Design Conference (ISOCC)*. IEEE, 2022, pp. 83–84.

[22] Z. Liu, H. Seo, S. Sinha Roy, J. Großschädl, H. Kim, and I. Verbauwhede, "Efficient ring-lwe encryption on 8-bit avr processors," in *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*. Springer, 2015, pp. 663–682.

[23] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "High-speed ntt-based polynomial multiplication accelerator for post-quantum cryptography," in *2021 IEEE 28th symposium on computer arithmetic (ARITH)*. IEEE, 2021, pp. 94–101.

[24] R. T. Moenck, "Practical fast polynomial multiplication," in *Proceedings of the third ACM symposium on Symbolic and algebraic computation*, 1976, pp. 136–148.

[25] N. Zhang, B. Yang, C. Chen, S. Yin, S. Wei, and L. Liu, "Highly efficient architecture of newhope-nist on fpga using low-complexity ntt/intt," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 49–72, 2020.