Designing Efficient and Flexible NTT Accelerators

Ahmet MALAL
Middle East Technical University
ASELSAN
Ankara, TURKEY
ahmetmalal@aselsan.com.tr

***Abstract*—The Number Theoretic Transform (NTT) is a powerful mathematical tool with a wide range of applications in various fields, including signal processing, cryptography, and error correction codes. In recent years, there has been a growing interest in efficiently implementing the NTT on hardware platforms for lattice-based cryptography within the context of NIST's Post-Quantum Cryptography (PQC) competition. The implementation of NTT in cryptography stands as a pivotal advancement, revolutionizing various security protocols. By enabling efficient arithmetic operations in polynomial rings, NTT significantly enhances the speed and security of lattice-based cryptographic schemes, contributing to the development of robust homomorphic encryption, key exchange, and digital signature systems.**

This article presents a new implementation of the Number Theoretic Transform for FPGA platforms. The focus of the implementation lies in achieving a flexible trade-off between resource usage and computation speed. By strategically adjusting the allocation of BRAM and DSP resources, the NTT computation can be optimized for either high-speed processing or resource conservation. The proposed implementation is specifically designed for polynomial multiplication with a degree of 256, accommodating coefficients of various bit sizes. Furthermore, the constant-geometry (Pease) method was utilized as an alternative to the Cooley-Tukey graph method, resulting in a notable simplification of BRAM addressing procedures. This adaptability renders it suitable for cryptographic algorithms like CRYSTALS-Dilithium and CRYSTALS-Kyber, which use 256-degree polynomials.

***Index Terms*—Number Theoretic Transform, Post-Quantum Cryptography, Hardware Cryptography, FPGA Implementation, Polynomial Multiplication**

I. Introduction

Quantum computers provide a severe threat to the information security industry's fast-evolving landscape. The security of sensitive data and communication is put at risk by the possibility of these computer systems breaking established encryption techniques. The necessity to develop new cryptographic techniques that can prevent both classical and quantum attacks has given rise to the area of post-quantum cryptography with the arrival of practical quantum computers.

The National Institute of Standards and Technology (NIST) has organized a comprehensive standardization process for post-quantum cryptography in response to the escalating significance of quantum-based security threats and their potential to compromise conventional cryptographic systems. Several algorithms are submitted from all over the world. These candidates include various mathematical techniques, such as lattice-based cryptography, code-based cryptography, multivariate polynomial cryptography, and more. Through an inclusive and detailed assessment, NIST seeks to identify cryptographic methods that exhibit robust resistance against potential quantum attacks. After an extensive standardization process, PQC competition's winners were announced in July 2022. Lattice-based cryptographic algorithms have become strong candidates in post-quantum cryptography competitions because they offer a good balance of security and speed [1], [2]. CRYSTALS-Dilithium was selected as a post-quantum secure digital signature algorithm, while CRYSTALS-Kyber was chosen as the public key encryption and key establishment algorithm.

Lattice-based cryptography employs complex mathematical structures known as lattices to build robust security measures for data protection [3]. Within this framework, polynomial multiplication is a fundamental operation that contributes to the effectiveness and efficiency of cryptographic algorithms. These cryptographic methods balance protecting sensitive information and maintaining computational performance by integrating polynomial multiplication with lattice-based techniques [1], [2].

The Number Theoretic Transform is a powerful polynomial multiplication technique, particularly in lattice-based cryptography and other areas of computer science. It leverages number theory principles to efficiently compute polynomial products by transforming the polynomial multiplication problem into a problem in the complex number domain. By converting the convolution operation into a point-wise multiplication operation in the transformed domain, NTT significantly reduces the computational complexity, making it an essential tool for applications requiring efficient polynomial multiplication. Its effectiveness in lattice-based cryptography and related fields highlights the critical role NTT plays in enhancing the performance of cryptographic algorithms and mathematical computations.

In this research, we introduce a new implementation architecture for FPGA platforms, centering on the NTT operation. We have developed a *run-time* configurable butterfly unit capable of supporting Cooley-Tukey, Gentleman-Sande, and pointwise multiplication operations simultaneously. This design feature provides us the flexibility to use the same butterfly unit for both forward and inverse NTT computations, as well as point-wise multiplications during *run-time*. The number of butterfly units used in the design is configurable and can be set at *compile-time*. This parameter directly impacts the utilization of Block RAM (BRAM) and Digital Signal

Processors (DSPs) within the design. Consequently, depending on specific requirements, it can be configured for either high-speed computation or resource conservative design. Using more butterfly units results in enhanced processing speed but costs a higher consumption of BRAM and DSP resources. We have used *run-time* configurable a Montgomery modular multiplier, capable of effectively multiplying coefficients of polynomials in R up to 31-bit in the butterfly units. Instead of using Cooley-Tukey or Gentleman-Sande methods, we preferred to use constant-geometry [4] method for easy to address BRAMs. Our design offers an adaptable and generic form for performing NTT operations on polynomials with a degree of 256, supporting coefficients up-to 31 bits.

II. Background and Related Work

Polynomial multiplication is a cornerstone operation in many computational domains, from classic computer algebra systems to cutting-edge cryptographic protocols. The significance of efficient polynomial multiplication extends to applications in signal processing, coding theory, and particularly emerging cryptographic schemes, such as lattice-based cryptography and homomorphic encryption as we mentioned in the introduction. Through leveraging the mathematical complexities of NTT, polynomial multiplication can be executed more rapidly, boosting the efficiency of applications that heavily depend on it. Thus, utilizing NTT becomes critical in advancing computational efficacy across several research fields.

Efforts to implement the NTT on FPGA platforms have gained momentum due to the demand for efficient and scalable solutions to secure digital communication in a post-quantum era. Researchers have explored various aspects of NTT FPGA implementation, from algorithmic considerations to optimization techniques and performance evaluations.

Lattice-based Cryptography

Lattice-based cryptography is an area of study in cryptographic research that builds security primitives upon the hardness of computational problems in lattices. These problems, often related to the difficulty of finding the shortest vector in a lattice or its closest variant, have been shown to be resistant against quantum attacks, made lattice-based schemes potential candidates for post-quantum cryptography. Both CRYSTALS-Dilithium and CRYSTALS-Kyber, the winners of PQC competition, are based on computational problems in lattices [5], [6].

A prominent problem in lattice-based cryptography is the Ring Learning With Error (Ring-LWE) problem. Unlike the standard Learning With Error (LWE) problem, Ring-LWE operates in polynomial rings and leverages the algebraic structure of these rings to achieve both efficiency and security. Mathematically, the Ring-LWE problem can be stated as follows: given random samples from a ring R, distinguish between the distributions $(a, a \cdot s + e)$ and (a, u), where s is a secret element, e is an error term, and u is a uniformly random element from the ring.

The Ring-LWE problem has found applications in various cryptographic constructs, including encryption schemes, digital signatures, and fully homomorphic encryption. Its hardness assumption, backed by worst-case hardness results in lattices, provides a strong foundation for building secure and efficient cryptographic protocols.

While lattice-based cryptography offers promising ways to secure data against quantum threats, ongoing research aims to optimize and further understand the mathematical foundations and practical implications of Ring-LWE and other related problems.

Number Theoretic Transform

Similar to the Fast Fourier Transform (FFT), NTT operates on sequences of numbers but focuses on modular arithmetic within a given number system. Let p be a prime number, and ω be a primitive n-th root of unity modulo p, where n is the length of the input sequence. The NTT converts a sequence $\{a_0, a_1, \ldots, a_{n-1}\}$ into another sequence $\{A_0, A_1, \ldots, A_{n-1}\}$, where

$$A_k = \sum_{j=0}^{n-1} a_j \cdot \omega^{jk} \pmod{p}$$

This transformation enables efficient operations in the frequency domain, allowing for the analysis and manipulation of data in a different representation. Lets assume that two polynomials a = $\{a_0, a_1, \ldots, a_{n-1}\}$ and b = $\{b_0, b_1, \ldots, b_{n-1}\}$ will be multiplied. The time complexity of polynomial multiplication is $\mathcal{O}(n^2)$ if schoolbook technique is used. Utilizing NTT technique necessitates the execution of the forward NTT operation on polynomial a, followed by its application on polynomial b. Subsequently, a point-wise multiplication is performed on the coefficients of polynomials a and b. Upon completion of these operations, the inverse NTT is employed. Consequently, the polynomial multiplication of a and b is effectively computed. NTT provides $\mathcal{O}(n \log n)$ time complexity, which is much more efficient than schoolbook technique.

Polynomial multiplication is generally based on two different parameters:

- **n**: the degree of the polynomial ring
- **k**: the bit length of the coefficient modulus

While homomorphic encryption schemes use large *n* and *k* parameters for polynomial multiplication, ranging from n = 1024 to n = 32768 and k = 14 to 60. Dilithium uses *(n,k)* = (256,23); Kyber uses *(n,k)* = (256,13).

Post-quantum cryptographic algorithms and homomorphic encryption schemes use NTT operation for the algorithm's efficiency. Therefore, several NTT implementations are designed and proposed with different implementation concerns for several platforms. *Run-time* reconfigurable NTT multiplication is proposed with supporting six different parameter sets [7] for FPGA platforms. Instead of applying the Cooley-Tukey algorithm, the constant-geometry-based NTT operation is implemented for ASIC platforms [8]. There also several

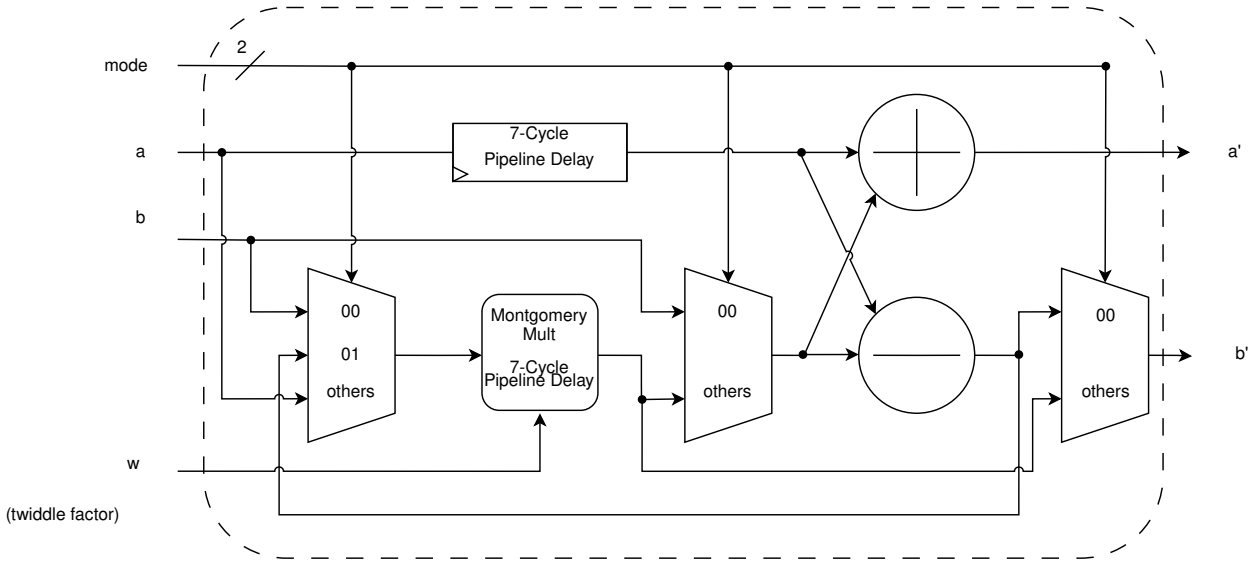

Fig. 1. The Proposed Butterfly Unit

optimized NTT implementations for software [9] and GPU [10] platforms.

Cooley-Tukey Algorithm

The Cooley-Tukey algorithm is a widely used method for efficiently computing the NTT and its inverse NTT. It employs a divide-and-conquer approach by recursively breaking down a NTT of any composite size $N = N_1 \times N_2$ into smaller NTTs of sizes N_1 and N_2. This iterative process continues until the base case of $N = 2$ is reached, at which point the NTT computation can be efficiently carried out using butterfly operations. These butterfly operations involve combining the results of the smaller NTTs to produce the final NTT result. The Cooley-Tukey algorithm significantly reduces the number of complex multiplications required compared to the straightforward computation of the NTT, making it an essential component in various applications involving signal processing.

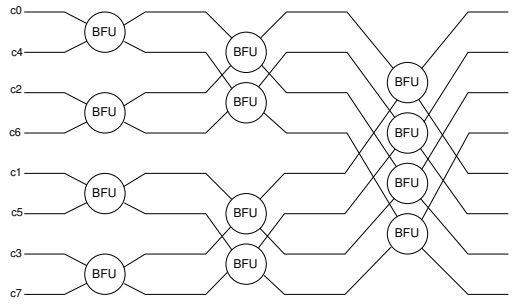

Fig. 2. Cooley-Tukey Graph based NTT operation for a polynomial degree 8.

III. The Proposed Architecture

Efficiently implementing NTT operation on FPGA platforms requires a careful consideration of memory utilization and computation speed. One key aspect in achieving optimal performance is to strike a balance between memory usage and computation speed, which is often referred to as the memory-speed trade-off. In this section, we present a proposed architecture that focuses on this trade-off to enhance the efficiency of NTT computations on FPGA.

Proposed Butterfly Unit

The butterfly unit is a fundamental building block in the NTT operation. It performs the essential complex multiplication and addition operations that contribute to the transformation of input data into NTT-domain coefficients. We have designed a butterfly unit such that it can be used both in forward and inverse NTT operation. Additionally, our proposed butterfly unit provides pointwise multiplication. Figure 1 shows the internal structure of the proposed butterfly unit. It receives four input parameters, denoted as a, b, w, $mode$, and produces two outputs, a' and b'. a and b are the coefficients of the polynomial, where w is the corresponding twiddle factor. $mode$ is used for encoding behaviour of the module. It is 2-bit, where "00" denotes forward NTT, "01" denotes inverse NTT, and "10" denotes point-wise multiplication. We used 7-stage pipeline delay in butterfly units to increase the clock frequency.

Montgomery Multiplication

Montgomery multiplication is a technique commonly used in modular arithmetic and cryptographic operations, particularly in modular exponentiation and elliptic curve cryptography. It aims to accelerate modular multiplications by reducing the number of expensive modular divisions. While several modular multiplication algorithms exist, including the Barrett and Karatsuba reduction methods, we selected the Montgomery algorithm due to its fulfillment with our design criteria. In Montgomery multiplication, the input operands are first transformed into a new representation known as the

Algorithm 1 Constant Geometry Based NTT Operation Algorithm

Require: Polynomial $a(x) \in \mathbb{R}_q$, η number of used butterfly unit and n-th primitive root of unity $\omega_n \in \mathbb{Z}_q$
Ensure: Polynomial $a'(x) \in \mathbb{R}_q$ such that $a'(x) = NTT(a(x))$

$x \leftarrow 0$
for i `in 0 to` $\eta - 1$ **do**
  for j `in 0 to` $256/\eta - 1$ **do**
    $RAM[i][j] \leftarrow a_{i*(256/\eta)+j}$ ▷ Fills the RAMs with coefficients
    $RAM[i+\eta][j] \leftarrow 0$ ▷ Generate other RAMs, Overall we need 2η RAMs
  end for
end for
for i `in 0 to 511` **do**
  $W[i] \leftarrow \omega_n^i 2^{\lceil log_2 q \rceil} \bmod q$ ▷ Calculates twiddle factors
end for ▷ NTT operation starts
for s `in 0 to 7` **do** ▷ Keeps the stage counter
  for j `in 0 to` $(256/2\eta) - 1$ **do**
    for i `in 0 to` $(\eta/2) - 1$ **do**
      $\omega_0 \leftarrow W[2^{7-s}(1 + 2(bitRev((i * 256/\eta) \bmod 2^s, s)))]$
      $\omega_1 \leftarrow W[2^{7-s}(1 + 2(bitRev((i * 256/\eta + 256/(2\eta)) \bmod 2^s, s)))]$
      $a_0 \leftarrow RAM[i][j]$, $b_0 \leftarrow RAM[i + \eta/2][j]$
      $a_1 \leftarrow RAM[i][j + 256/\eta]$, $b_1 \leftarrow RAM[i + \eta/2][j]$
      $c_0, d_0 \leftarrow BFU(a_0, b_0, w_0)$, $c_1, d_1 \leftarrow BFU(a_1, b_1, w_1)$ ▷ Butterfly Operation
      $RAM[2i + \eta][2x] \leftarrow c_0$
      $RAM[2i + \eta][2x + 1] \leftarrow d_0$
      $RAM[2i + 1 + \eta][2x] \leftarrow c_1$
      $RAM[2i + 1 + \eta][2x + 1] \leftarrow d_1$
      $x \leftarrow x + 1$
    end for
  end for
  for i `in 0 to` $\eta/2 - 1$ **do**
    $RAM[i] \leftarrow RAM[i + \eta]$
  end for
end for
for i `in 0 to` $\eta - 1$ **do**
  for j `in 0 to` $256/\eta - 1$ **do**
    $a'_{i*(256/\eta)+j} \leftarrow RAM[i][j]$ ▷ Constructs the output polynomial
  end for
end for

Montgomery domain, which simplifies the modular reduction step. The core of the algorithm involves a series of additions and bit-wise operations.

Since we work with signals in the FPGA, we can work on numbers of any bit length we want. While each coefficient is stored in a 32-bit register for Dilithium parameters in the software, we used 23-bit registers FPGA. This choice allowed us to set the Montgomery constant as 2^{23} instead of 2^{32}. As a result, when we transform a polynomial into the NTT domain, we obtain results that differ from the reference C-Code provided by the Dilithium designers [6]. However, when we perform the complete multiplication, we achieve the same result.

Constant-Geometry Based NTT

Instead of using the Cooley-Tukey method, we used constant-geometry based graph to perform NTT operation. The Cooley-Tukey method is a good practice for software platforms since it has a recursive design. However, implementing recursive methods commonly brings more cost to hardware platforms. In every stage, the butterfly unit input comes from different addresses of BRAM. We would need to construct an address arranger to rearrange the addresses, which also comes with resource costs. However, every stage rearrangements are the same in a constant-geometry based graph [4]. It also provides flexibility on number of butterfly units. Based on our choice of the number of butterfly units, the number of BRAMs increases correspondingly; however, this accelerates NTT operation. Algorithm 1 presents the pseudo-code we developed.

For the efficient utilization of these butterfly units, an enlargement in the number of Block RAMs (BRAMs) was necessitated, resulting in a twofold increase. We harnessed true-dual port BRAMs to accommodate this boost, which

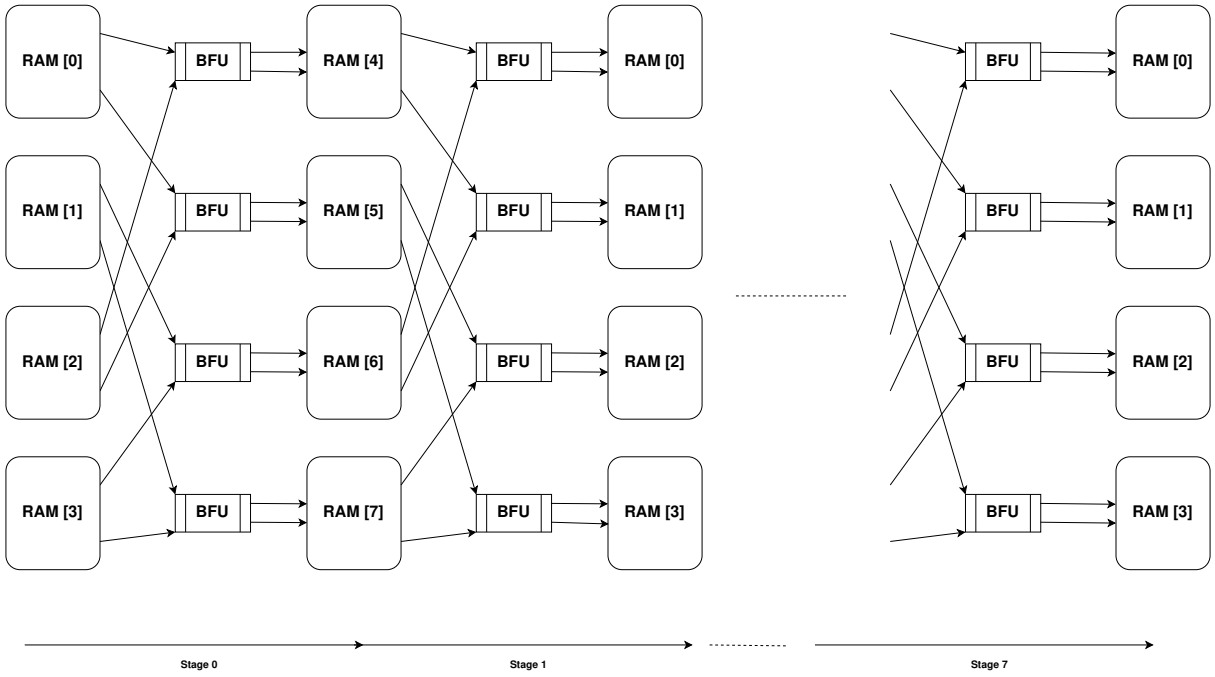

Fig. 3. Internal Structure of NTT Operation, where number of butterfly unit (η) is 4.

permits concurrent access to two distinct addresses within a single cycle. This allocation was strategized such that the coefficients of a given polynomial were uniformly distributed across the first half of the BRAMs. In each computational cycle, a coefficient was drawn sequentially from the upper section of the BRAM and another from the middle portion. These paired coefficients were then directed to their respective designated butterfly units for processing. The resulting outputs from these butterfly units were transferred directly to the latter half of the BRAMs.

In Figure 3, we showed the internal structure of the NTT operation stage by stage. Let us consider the scenario where the forward NTT operation of a polynomial comprising 256 coefficients is undertaken, employing a designated butterfly unit count (#BFU) of 4. The initial 64 coefficients will be allocated to the first BRAM, followed by the subsequent 64 coefficients to the second BRAM, and so forth, maintaining this pattern. The input of the first BFU is the first element of the first BRAM and the first element of the third BRAM. The inputs of the second BFU will be the 32^{nd} elements of the first BRAM and 32^{nd} elements of the third BRAM. The pattern is detailed in Figure 3. The results of BFUs will be directed to the corresponding BRAMs. For the first BFU, the results will be directed to the fifth BRAM. The output of the second BFU will be moved to the sixth BRAM. In the next stage, the inputs of the BFUs will come from the fifth, sixth, seventh, and eighth BRAMs. The BRAMs will be used vice-versa in each stage. After eight stages, the forward NTT operation of the given polynomial will be computed.

IV. Results and Conclusion

This section presents the results of our proposed memory-speed trade-off implementation of NTT operation within the CRYSTALS-Dilithium algorithm. The NTT operation is pivotal in various cryptographic primitives. It has significant attention due to its relevance in the context of the National Institute of Standards and Technology (NIST) Post-Quantum Cryptography (PQC) initiative.

Table II and Table III provide a comprehensive overview of the outcomes obtained from our implementations. In Table II, we present the results of the forward NTT implementation, focusing on memory consumption, execution time, and achieved throughput. Our proposed trade-off mechanism allows for a fine adjustment between memory utilization and processing speed, catering to diverse application requirements. The variation in memory-speed trade-off is systematically explored, highlighting the flexibility of our approach. Additionally, Table III illustrates the outcomes of the inverse NTT implementation, mirroring a similar analysis with a specific emphasis on the trade-off between memory utilization and computational efficiency.

To compare the effectiveness of our proposed approach, we provide a comparative analysis in Table I. This table compares our results with other prominent researchers in the field. By doing so, we gain valuable insights into the performance benchmarks of existing methodologies. The comparison covers aspects such as memory consumption, execution time, and balancing strategies, highlighting the improvements achieved by our proposed memory speed balancing implementation.

Overall, the results presented in this section underscore the significance of our contribution. Our memory-speed trade-off

TABLE I
COMPARATIVE FPGA RESOURCE AND PERFORMANCE TABLE, FOR DILITHIUM NTT PARAMETERS (N,K) = (256,23)

Design	LUT	FF	BRAM	DSP	Cycles	Frequency(MHz)	Platform
[11]	1919	1301	2	8	296	96.9	Artix-7
[12]	799	328	4.5	2	1405	172	Artix-7
[13]	2044	N/A	16	N/A	1170	216	Artix-7
[14]	524	759	17	1	533	311	Virtex-7
[15]	1899	2041	2	8	294	445	Zynq UltraScale+
[16]	1798	2532	3.5	48	502	637	Virtex UltraScale+
[17]	4509	3146	16	0	300	N/A	Virtex UltraScale+
This Work,**Our Smallest**	**1601**	699	5	10	603	142.8	Virtex UltraScale+
This Work,**Our Fastest**	11558	4912	40	80	**155**	142.8	Virtex UltraScale+

TABLE II
PERFORMANCE RESULTS OF THE PROPOSED DESIGN FOR FORWARD NTT OPERATION, WHERE (N,K) = (256,23) ON VIRTEX ULTRASCALE+

#BFU	LUT	FF	BRAM	DSP	Cycles
2	1601	699	5	10	512+91 = 603
4	3080	1279	10	20	256+91 = 347
8	5496	2463	20	40	128+91 = 219
16	11558	4912	40	80	64+91 = 155

TABLE III
PERFORMANCE RESULTS OF THE PROPOSED DESIGN FOR INVERSE NTT OPERATION, WHERE (N,K) = (256,23) ON VIRTEX ULTRASCALE+

#BFU	LUT	FF	BRAM	DSP	Cycles
2	2405	794	4	10	512+128+102=742
4	4734	1462	8	20	256+64+102=422
8	7864	2796	16	40	128+32+102=262
16	15885	5559	32	80	64+16+102=182

implementation of the NTT operation offers a customizable balance between resource utilization and computational swiftness. By configuring the number of butterfly units used in the design, we generated various of implementations. We compared our smallest and fastest results with the results of other researchers. The results we obtained show that our flexible design is comparable to other results. The comparative analysis shows our approach's competitiveness within the landscape of lattice-based cryptography, aligning with the objectives of the NIST PQC initiative. These findings collectively emphasize the potential of our methodology to pave the way for efficient and robust NTT solution.

As part of future work, we aim to expand the scope of our research in several directions. Our NTT implementation is designed to operate on polynomials with $n = 256$, supporting coefficients of up to 31-bit. In the context of this study, performance metrics were obtained for only Dilithium $(n, k) = (256, 23)$ parameters. First, we plan to obtain performance results for a wider range of NTT parameters, including those relevant to Kyber and other cryptographic algorithms. Additionally, we intend to delve deeper into the optimization of the Montgomery multiplier to enhance its frequency performance. Finally, for the purpose of comprehensive comparison and evaluation, we will consider performing our NTT implementation on different FPGA platforms. These future efforts will not only contribute to a more comprehensive understanding of the algorithm's flexibility and efficiency but also pave the way for its broader applicability in various computational contexts.

V. ACKNOWLEDGEMENT

The author would like to thank Erdem Alkım for the meaningful discussion and his comments during the preparation of this study.

REFERENCES

[1] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "Crystals - kyber: A cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 353–367, 2018.

[2] E. Alkim, P. S. L. M. Barreto, N. Bindel, J. Krämer, P. Longa, and J. E. Ricardini, "The lattice-based digital signature scheme qtesla," in *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part I* (M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, eds.), vol. 12146 of *Lecture Notes in Computer Science*, pp. 441–460, Springer, 2020.

[3] N. Bindel, J. Buchmann, and J. Krämer, "Lattice-based signature schemes and their sensitivity to fault attacks." Cryptology ePrint Archive, Paper 2016/415, 2016.

[4] M. C. Pease, "An adaptation of the fast fourier transform for parallel processing," *J. ACM*, vol. 15, p. 252–264, apr 1968.

[5] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals – kyber: a cca-secure module-lattice-based kem." Cryptology ePrint Archive, Paper 2017/634, 2017.

[6] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle, "Crystals – dilithium: Digital signatures from module lattices." Cryptology ePrint Archive, Paper 2017/633, 2017.

[7] A. C. Mert, E. Öztürk, and E. Savas, "FPGA implementation of a run-time configurable ntt-based polynomial multiplication hardware," *Microprocess. Microsystems*, vol. 78, p. 103219, 2020.

[8] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan, "Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 4, pp. 17–61, 2019.

[9] E. Alkim, V. Hwang, and B.-Y. Yang, "Multi-parameter support with ntts for ntru and ntru prime on cortex-m4," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, p. 349–371, Aug. 2022.

[10] O. Ozerk, C. Elgezen, A. C. Mert, E. Ozturk, and E. Savas, "Efficient number theoretic transform implementation on gpu for homomorphic encryption." Cryptology ePrint Archive, Paper 2021/124, 2021.

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

[12] G. MAO, D. CHEN, G. LI, W. DAI, A. SANKA, Ç. KOÇ, and R. CHEUNG, "High-performance and configurable sw/hw co-design of post-quantum signature crystals-dilithium," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, oct 2022. Research Unit(s) information for this publication is provided by the author(s) concerned.

[13] Z. Zhou, D. He, Z. Liu, M. Luo, and K. R. Choo, "A software/hardware co-design of crystals-dilithium signature scheme," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 14, no. 2, pp. 11:1–11:21, 2021.

[14] G. Land, P. Sasdrich, and T. Güneysu, "A hard crystal - implementing dilithium on reconfigurable hardware," in *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers* (V. Grosso and T. Pöppelmann, eds.), vol. 13173 of *Lecture Notes in Computer Science*, pp. 210–230, Springer, 2021.

[15] D. T. Nguyen, V. B. Dang, and K. Gaj, "A high-level synthesis approach to the software/hardware codesign of ntt-based post-quantum cryptography algorithms," in *International Conference on Field-Programmable Technology, FPT 2019, Tianjin, China, December 9-13, 2019*, pp. 371–374, IEEE, 2019.

[16] S. Ricci, L. Malina, P. Jedlicka, D. Smékal, J. Hajny, P. Cíbik, P. Dzurenda, and P. Dobias, "Implementing crystals-dilithium signature scheme on fpgas," in *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021* (D. Reinhardt and T. Müller, eds.), pp. 1:1–1:11, ACM, 2021.

[17] L. Beckwith, D. T. Nguyen, and K. Gaj, "High-performance hardware implementation of crystals-dilithium," in *International Conference on Field-Programmable Technology, (IC)FPT 2021, Auckland, New Zealand, December 6-10, 2021*, pp. 1–10, IEEE, 2021.