Hardware Implementation of K^{2}RED and Plantard Modular Multiplication Algorithms in Post-Quantum Cryptography

Furkan Can[1,2], Ali Ustun[1,2], Berna Ors[1], Ersin Alaybeyoglu[3,4], Erkay Savas[3]

[1]Istanbul Technical University, Turkey

[2]TUBITAK BILGEM, Turkey

[3]Sabancı University, Turkey

[4]Bartın University, Turkey

canf16, ustuna16, orssi @itu.edu.tr, ersin.alaybeyoglu, erkay.savas @sabanciuniv.edu

This manuscript has been accepted for publication in an IEEE conference.

The final published version is available in IEEE Xplore:

DOI: 10.1109/PACET60398.2024.10497033

Funding Acknowledgement:

This work has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement Number: 101079319.

Hardware Implementation of K^{2}RED and Plantard Modular Multiplication Algorithms in Post-Quantum Cryptography

Furkan Can[1,2], Ali Üstün[1,2], Berna Örs[1], Ersin Alaybeyoğlu[3,4], Erkay Savaş[3]
[1]Istanbul Technical University, Turkey
[2]TUBITAK BILGEM, Turkey
[3]Sabancı University, Turkey
[4]Bartın University, Turkey
canf16, ustuna16, orssi @itu.edu.tr, ersin.alaybeyoglu, erkay.savas @sabanciuniv.edu

Abstract

As the era of Post-Quantum Cryptography emerges, the demand for efficient and secure cryptographic algorithms has intensified. This conference paper navigates through the details of employing Number Theoretic Transform based modular multiplication algorithm which is K^{2}RED tailored for post-quantum cryptographic applications, with a specific emphasis on their Field Programmable Gate Array implementation. The study investigates in detail of these algorithms, considering their impact on security and performance in the face of quantum threats.

The FPGA implementation aspect of this research is a key highlight, aiming to bridge the gap between theoretical advancements and practical real-world applicability. We present a detailed analysis of the area, speed, and latency performance of the implemented modular multiplication algorithms on FPGA platforms. The FPGA implementation of NTT-based modular multiplication algorithms serves as a critical bridge between theoretical advancements and their real-world applicability in the context of PQC.

NTT is an operation of Lattice Based Cryptography and it converts the numbers to finite field and modular multiplication operation is carried out. So, modular multiplication is the essential operation of NTT and its efficiency is critical for NTT operation. Our findings not only contribute to the growing body of knowledge in PQC but also offer practical guidance for engineers and researchers seeking to implement robust and efficient cryptographic solutions on FPGA platforms. The presented performance metrics serve as a benchmark for evaluating the feasibility and scalability of modular multiplication algorithms in the context of PQC systems.

Index Terms

Modular Multiplication, K^{2}RED, Kyber, Dilithium, Number Theoretical Theorem(NTT), Post Quantum Cryptography(PQC), FPGA

I. Introduction

In the ever-evolving landscape of cryptography, the advent of quantum computing presents unprecedented challenges to the security of classical cryptographic algorithms. Quantum computers, harnessing the principles of quantum mechanics, have the potential to render widely-used encryption methods such as Rivest-Shamir-Adleman (RSA) [1] and Elliptic Curve Cryptography (ECC) [2] vulnerable, courtesy of algorithms like Shor's Algorithm [3]. As the cryptographic community deals with the looming threat of quantum adversaries, a new paradigm known as Post Quantum Cryptography (PQC) is emerging [4].

The National Institute of Standards and Technology (NIST) and other crypto research groups play a pivotal role in standardizing PQC algorithms [5]. Ongoing research and evaluation endeavors seek to establish a new cryptographic foundation that can withstand the capabilities of quantum computers. The CRYSTALS-Kyber [6] and CRYSTALS-Dilithium [7] are two candidates submitted to the NIST for consideration in the process of standardizing PQC algorithms. They are lattice-based cryptographic algorithms [8] and built upon learning with errors (LWE) problem [9].

The CRYSTALS-Kyber is designed for secure key exchange which is named as key encapsulation mechanism (KEM) [10]. The CRYSTALS-Dilithium is designed for digital signatures [11].

Within the nature of PQC, the Number Theoretic Transform (NTT) emerges as a pivotal mathematical operation [12]. NTT serves as a key point for applications such as lattice-based cryptography, offering an efficient framework for polynomial multiplication and convolution operations.

Nestled within the field of NTT-based approaches, K^{2}RED [13] and Plantard [14] modular reduction algorithms play significant role in optimizing modular multiplication —a vital component for maintaining finite field arithmetic in NTT operations. There is no previous implementation of Plantard modular multiplication algorithm, and there are few implementations of K^{2}RED reduction algorithm that are inspected in [15] and [16]. In case of Plantard modular multiplication algorithm, algorithm requires reduced multiplication number and simpler arithmetic than other modular multiplication algorithms and it is suitable for small sized primes as K^{2}RED modular multiplication algorithm.

This paper embarks on an exploration of the significance of modular multiplication, incorporating the K^{2}RED and Plantard algorithms, in the context of NTT-based PQC systems through an exhaustive examination of the theoretical foundations, practical FPGA implementations, and comprehensive performance evaluations. Our objective is to provide nuanced insights that seamlessly bridge theoretical advancements with real-world applications. This contribution is instrumental for designers in order to choose the unique modular multiplication architecture which meets their area, frequency, throughput and latency requirements.

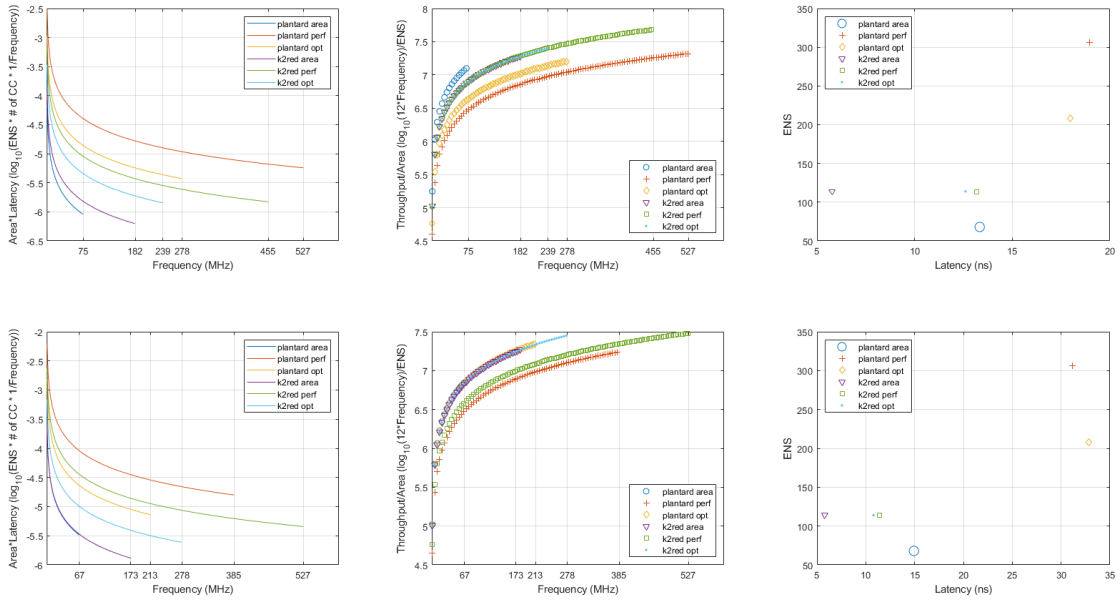

Fig. 1. Comparisons of K^{2}RED and Plantard modular multiplication

II. Mathematical Background

A. Lattice Based Post Quantum Cryptography

PQC aims to forge cryptographic algorithms impervious to the computational might of quantum computers [4]. A leading contender in this realm is lattice-based cryptography, leveraging the complexity of lattice problems to fortify security [8]. Unlike traditional public-key cryptography, lattice-based cryptography proffers resilience against quantum attacks. If we express mathematically the NTT process that can be used in Lattice based PQC processes for CRYSTALS-Kyber example [6]:

Let $q = 3329$ with $q - 1 = 2^8 13$, the base field $\mathbb{Z}_q$ contains primitive 256-th roots of unity. The polynomial $X^{256} + 1$ of R factors into 128 polynomials of degree 2 modulo q and the NTT of a polynomial $f \epsilon R_q$ is a vector of 128 polynomials of degree one. The NTT can be implemented simply and in-place without reordering outputs, resulting in polynomials presented in bit-reversed order. We define the NTT in this particular manner. For instance, let ζ = 17 be the first primitive 256-th root of unity modulo q, and ζ, ζ_3, ζ_5, . . . , ζ_{255} the series of all the 256-th roots of unity. The polynomial $X^{256} + 1$ can therefore be written as

$$X^{256} + 1 = \prod_{i=0}^{127}(X^2 - \zeta^{2i+1}) = \prod_{i=0}^{127}(X^2 - \zeta^{2br_7(i)+1}) \tag{1}$$

where $br_7(i)$ for $i = 0, 1, ...127$ is the bit reversal of the unsigned 7-bit integer. Then the NTT of $f \epsilon R_q$ is given by

$$(fmodX^2 - \zeta^{2br_7(0)+1}, ..., fmodX^2 - \zeta^{2br_7(127)+1}) \tag{2}$$

$$NTT(f) = \hat{f} = \hat{f}_0 + \hat{f}_1 X + ... + \hat{f}_{255} X^{255} \tag{3}$$

with

$$\hat{f}_{2i} = \sum_{j=0}^{127} f_{2j} \zeta^{(2br_7(i)+1)j} \tag{4}$$

$$\hat{f}_{2i+1} = \sum_{j=0}^{127} f_{2j+1}\zeta^{(2br_7(i)+1)j} \tag{5}$$

We want to emphasize that while we express $\hat{f}$ as a polynomial in R_q, it doesn't possess any inherent algebraic significance. The appropriate algebraic representation of $NTT(f) = \hat{f}$ is in the form of 128 polynomials of degree 1, as described in (2), utilizing the definitions for $\hat{f}_i$ provided in (4) and (5). That is,

$$NTT(f) = \hat{f} = (\hat{f}_0 + \hat{f}_1 X, ..., \hat{f}_{254} + \hat{f}_{255} X^{255}) \tag{6}$$

By employing the NTT and its inverse, NTT^{-1}, we can effectively compute the product $f \cdot g$ of two elements, f and g, in R_q. This computation is carried out efficiently as $NTT^{-1}(NTT(f) \circ NTT(g))$, where $NTT(f) \circ NTT(g) = \hat{f} \circ \hat{g} = \hat{h}$ represents the base case multiplication involving the 128 products.

$$\hat{h}_{2i} + \hat{h}_{2i+1} X = (\hat{f}_{2i} + \hat{f}_{2i+1} X(\hat{g}_{2i} + \hat{g}_{2i+1} X) mod X^2 - \zeta^{2br_7(i)+1} \tag{7}$$

The NTT operation serves as a mathematical technique designed to alleviate computational complexity when multiplying large polynomials.

NTT can be implemented using Cooley-Tukey structure [17] and Gentleman-Sande [18] butterflies inside. Using these butterflies, NTT can be implemented by a "divide-and-conquer´´ method which reduces the time complexity of the polynomial multiplications from $O(n^2)$ to $O(n \cdot \log n)$. The Cooley-Tukey structure for $k = 8$ can be seen in [19].

B. K²RED Modular Multiplication

In contrast to alternative modular multiplication techniques, K²RED stands out as a method exclusively applicable to specific prime numbers [13]. Introduced by Niasar et al. in [13], K²RED executes modular multiplication operations using prime numbers recognized as Proth numbers [20]. Proth numbers, characterized by a distinctive form as $q = k2^m + 1$, where $2^m > k$ and k is an odd number, serve as the prime numbers of choice for this specialized algorithm.

Algorithm 1 K²RED Modular Multiplication Algorithm

Require: A, B, P, n with $P < 2^n$, $P = k2^m + 1$, k is odd, $k < 2^m$, $0 \leq A$, $B < 2^n$, $t < 2n$

Ensure: $C' = k^2(AB) \mod P$

1: $R = AB$
2: $R_l \leftarrow (r_m, r_{m-1}, ..., r_1, r_0)_2$
3: $R_h \leftarrow (r_{2n-1}, ..., r_{m+1})_2$
4: $C \leftarrow kR_l - R_h$
5: $C_l \leftarrow (c_m, c_{m-1}, ..., c_1, c_0)_2$
6: $C_h \leftarrow (c_t, ..., c_{m+1})_2$
7: $C' \leftarrow kC_l - C_h$

The K²RED algorithm, a variant of the Montgomery Modular Multiplication Algorithm [21] introduced in [20], builds upon the KRED algorithm outlined in [13]. In the K²RED algorithm, the KRED algorithm is applied twice and executed through an adder and a shifter.

Initially K²RED modular reduction algorithm was proposed for CRYSTALS-Kyber PQC algorithm. In this work we added multiplication in order to extend K²RED reduction algorithm to K²RED modular multiplication algorithm and also we applied it to CRYSTALS-Dilithium PQC algorithm for the first time.

Algorithm 1 shows the steps of K²RED modular reduction algorithm. In CRYSTALS-Kyber algorithm, $P = 3329$, $m = 8$ and $k = 13$. $P = 8380417$ is given in CRYSTALS-Dilithium [7], we calculated the parameters of K²RED as $m = 13$ and $k = 1023$.

In step 6, the bit length is denoted as t, and it differs from the bit length generated in the preceding operation. It's important to note that these values are constants in the CRYSTALS-Kyber and CRYSTALS-Dilithium algorithms.

In conclusion, it is important to highlight that the resulting C' value does not directly correspond to the expected outcome of modular multiplication; rather, it is equivalent to the $k^2C \mod P$ value provided at the output. The actual result of the modular multiplication is achieved by multiplying the value obtained from K²RED by k^{-2}. However, since twiddle factors ω_i in the NTT process given as [22] are fixed values, these constants are multiplied by k^{-2} in advance.

Algorithm 2 Plantard Modular Multiplication Algorithm

Require: $A, B, P, R, n, \quad 0 \leq A, B < P,$
$R = P^{-1} \bmod 2^{2n}, \Phi = \frac{1+\sqrt{5}}{2}, P < \frac{2^n}{\Phi}$

Ensure: $C = A \cdot B \cdot (-2^{-2n}) \bmod P$

1: $R = A \cdot B \bmod 2^{2n}$
2: $C_1 \leftarrow (\lfloor \frac{R}{2^n} \rfloor) + 1$
3: $C_2 \leftarrow \lfloor \frac{C_1 \cdot P}{2^n} \rfloor$
4: **if** $C_2 = P$ **then**
5: $\quad C_2 \leftarrow 0$

C. Plantard Modular Multiplication

Thomas Plantard proposed Algorithm 2 in 2021 that is effective for smaller length prime numbers than RSA, ECC, etc [14].

Plantard modular multiplication algorithm needs n-bit A and B input which needs to be bigger than selected modulo size. In our case as selected CRYSTALS-Kyber and CRYSTALS-Dilithium, modulo sizes are 12-bits and 23-bits long respectively. According to limits that is shown in Algorithm 2 n should be selected as 13 and 24 bits respectively. As given in [22], twiddle factors must be precomputed with -2^{2n} since Plantard modular multiplication algorithm in NTT implementation

TABLE I
UTILIZATION OF DIFFERENT METHODS USED IN K^{2}RED AND PLANTARD MODULAR MULTIPLICATION ALGORITHM FOR CRYSTALS-KYBER AND CRYSTALS-DILITHIUM PQC SCHEMES

Algorithm	PQC Algorithm	ENS	Slices	LUT	FF	DSP	CPD (ns)	Throughput(Mbps)	Cycles
Plantard	CRYSTALS-Kyber	68	68	270	12	0	13.39	895.56	1
Plantard	CRYSTALS-Kyber	306	6	8	43	3	1.89	6319.44	10
Plantard	CRYSTALS-Kyber	208	8	29	28	2	3.6	3333.24	5
K^{2}RED	CRYSTALS-Kyber	114	14	54	36	1	5.5	2181.82	1
K^{2}RED	CRYSTALS-Kyber	113	13	52	90	1	13.2	5454.48	6
K^{2}RED	CRYSTALS-Kyber	114	14	55	65	1	12.6	2857.08	3
Plantard	CRYSTALS-Dilithium	219	219	876	23	0	15.1	1523.29	1
Plantard	CRYSTALS-Dilithium	508	8	32	96	5	2.59	8846.26	12
Plantard	CRYSTALS-Dilithium	221	221	881	671	0	4.69	4893.71	7
K^{2}RED	CRYSTALS-Dilithium	226	26	103	73	2	5.8	3965.2	1
K^{2}RED	CRYSTALS-Dilithium	400	0	0	104	4	1.9	12104.9	6
K^{2}RED	CRYSTALS-Dilithium	226	26	103	136	2	3.6	6387.1	3

* The slice of FPGA implementations is estimated by formula #LUTs/4 as one slice consists 4 LUTs in Artix-7 [23].
** ENS (Eqivalant Number of Slices) = Slices + DSPs× 100 + BRAMs× 196 [24].
*** Critical Path Delay (CPD)

III. HARDWARE DESIGN OF MODULAR MULTIPLICATION ALGORITHMS

In this study, hardware architectures have been developed for Plantard, and K^{2}RED modular multiplication algorithms. Block diagrams illustrating the proposed hardware architectures are depicted in Figures 2 and 3.

The implementation of Plantard and K^{2}RED modular multiplication algorithms is executed on a Xilinx xc7a100tcsg324-2L Field Programmable Gate Array (FPGA). This FPGA platform is chosen to realize the hardware implementation of the algorithms.

In the section III, when creating the resulting performance tables, this study uses the 8. In Equation 8, f is frequency, n is bit size of output.

$$Throughput = f \cdot n \tag{8}$$

A. K^{2}RED Modular Multiplication

We have drawn the block diagram shown in Fig. 2 that will be the basis for the hardware design of Algorithm 1.

The RTL schematic of the hardware module for K^{2}RED modular multiplication algorithm implemented on an FPGA is given in Fig. 2. The circuit RTL schematic of two algorithms is the same. The circuits can be synthesized according to the bit lengths of CRYSTALS-Kyber and CRYSTALS-Dilithium. Also, prime numbers are fixed in CRYSTALS-Kyber and CRYSTALS-Dilithium algorithms. The circuit outputs after 6 clock cycles in all three implementations. After 6 clock cycles, the circuit gives output at every clock. Therefore, the output bit lengths are 12 and 23 for CRYSTALS-Kyber and CRYSTALS-Dilithium. The throughput values are calculated with the Eq. 8 and using the given bit lengths.

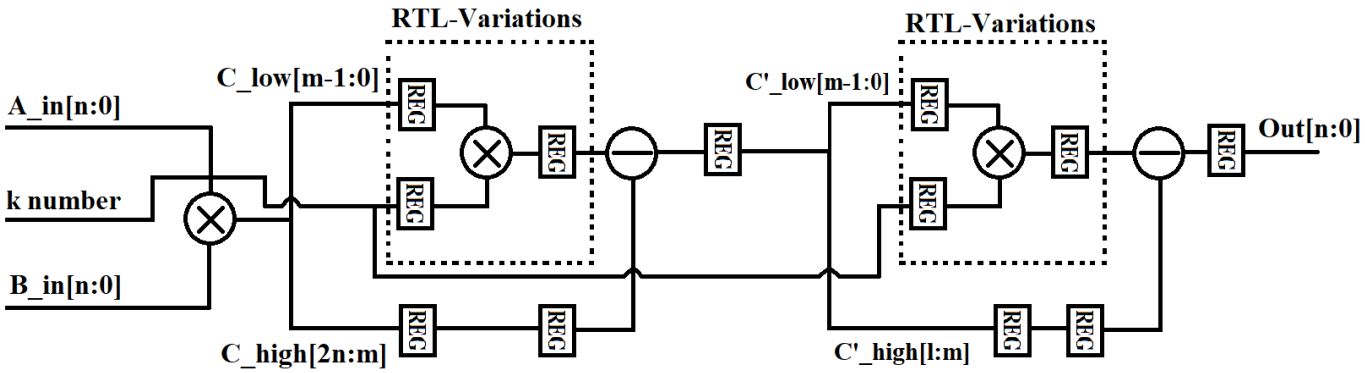

Fig. 2. K^{2}RED Modular Multiplication Register Transfer Level(RTL) Diagram.

The number of FPGA resources used as a result of the implementation of the K^{2}RED algorithm are given in the Table-I. The first thing to note in the table is that the Digital Signal Processor(DSP) numbers are the same even though the bit lengths of CRYSTALS-Kyber and CRYSTALS-Dilithium algorithms are different. The most important benefit of this algorithm is that it can be implemented with 2 DSPs if the input bit length is up to 32.

B. Plantard Modular Multiplication

In Fig. 3, Plantard modular multiplication algorithm can be seen and if we inspect the block diagram shown in Figure, we can deduct that 2 multiplication, 1 addition and 1 comparison is enough to realize the algorithm. First multiplication is $n * 2n$ and second multiplication is $n * n$ in Plantard modular multiplication algorithm. In the design of Plantard multiplication as seen in algorithm 2, modular side of algorithm needs C_1 which is middle n bits of AB calculation therefore in the design of a multiplier we can neglect the sums that will benefit to calculate AB multiplication but also when trying to implement this algorithm with DSP blocks ,that are provided by FPGA Vendor, we should take into account that DSPs provide 18x25 multipliers.

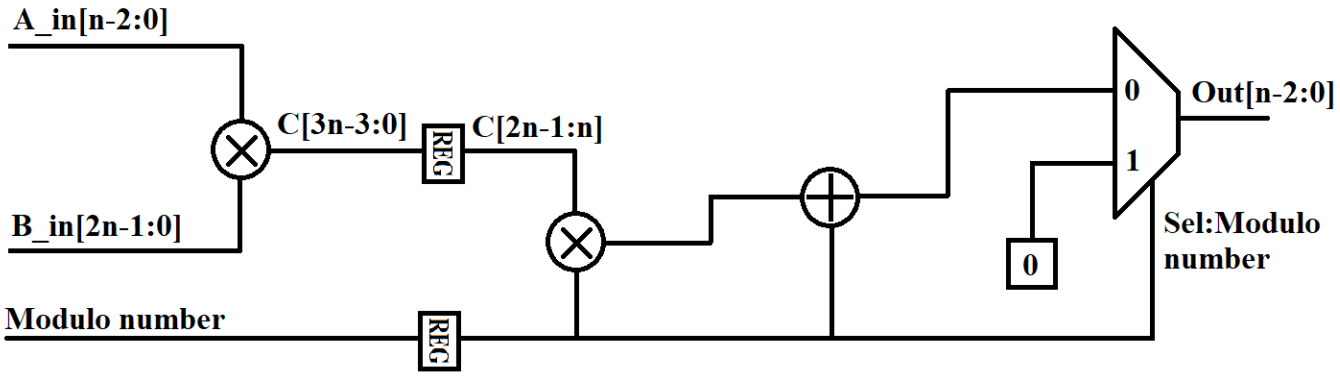

Fig. 3. Plantard Modular Multiplication RTL Diagram

For example, in the case of CRYSTALS-Kyber; as prime is 3329 which is 12-bit long, our word length n is minimum of 13 that means we need 13 by 26 bits multiplier which is not compatible with interface of DSPs therefore 2 DSPs are used to calculate 13- by 26-bits multiplication.

C. Comparison of Modular Multiplication Algorithms

Different implementations of K^{2}RED and Plantard modular multiplication are dependent with the design of upper blocks and requirements of the design. In this study, K^{2}RED and Plantard modular multiplication are compared and results of this comparison has given in the Table I and Figure 1. These modular multiplication module that are designed is a part of bigger design which can be NTT design, frequencies of different designs of K^{2}RED and Plantard modular multiplication is highly dependent with the pipeline number of design. Different pipeline numbers given in the Table I named as ‘area’, ‘opt’ and ‘performance’ which means optimization is taken care of these terms.

Here, in Table I and Figure 1 comparison between K^2RED and Plantard modular multiplication is given. From the Table I, we can deduct algorithms in their area and performance results. Looking at results, implementation selection between the modular multiplication algorithm highly depends on your requirements and looking to Figure 1, Plantard modular multiplication, when in CRYSTALS-Kyber PQC scheme, is ahead when design is combinational with no pipelines and it gets worse then other implementations eventough frequency and throughput of design gets greater than combinational designs. If we compare modular reduction algorithms in CRYSTALS-Dilithium PQC scheme, we can deduct that, between K^2RED and Plantard modular multiplication algorithms, K^2RED modular multiplication algorithm gets better in terms of Throughput/Area by Frequency and Area-Latency by frequency graphs. If your designs need lower than $\approx$ 70 MHz Plantard modular multiplication for CRYSTALS-Kyber PQC scheme can be selected as it has efficieny in terms of Throughput/Area and Area-Latency then other implementations but these result is until $\approx$ 70 MHz and if you have frequency greater than this, K^2RED modular multiplication gets lead against the Plantard modular multiplication algorithm.

IV. Conclusion

This study provides an extensive examination of modular multiplication algorithms within the context of PQC and explores their practical implementations on FPGAs. The analysis of time-area trade-offs serves as a key factor in the selection of these algorithms. Anticipating applications across diverse domains such as information, computation, and communication, ongoing algorithm implementation studies are actively addressing security concerns in the post-quantum era of informatics.

The research specifically delves into modular reduction algorithms, playing a crucial role in accelerating NTT and optimizing fields. Looking ahead, these designed algorithms hold promise for completing the NTT process, with an emphasis on the need for designing an efficient multiplier circuit. Furthermore, there is a potential avenue for a comprehensive comparison of modular multiplication algorithms, considering factors such as energy consumption, resilience against side-channel attacks, and overall performance criteria. Such comparative studies contribute significantly to a nuanced understanding and enhancement of the overall cryptographic system performance.

References

[1] Y. Li, Q. Liu, and T. Li, "Design and implementation of an improved rsa algorithm," in *2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT)*, vol. 1, pp. 390–393, 2010.

[2] M. Zeghid, H. Y. Ahmed, A. Chehri, and A. Sghaier, "Speed/area-efficient ecc processor implementation over gf(2m) on fpga via novel algorithm-architecture co-design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 8, pp. 1192–1203, 2023.

[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, (USA), p. 124–134, IEEE Computer Society, 1994.

[4] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Comput. Surv.*, vol. 51, jan 2019.

[5] C. S. R. Center, "Post-quantum cryptography," 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography [Accessed: (15.11.2023)].

[6] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, *CRYSTALS -Kyber: a CCA-secure module-lattice-based KEM*.

[7] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, p. 238–268, Feb 2018.

[8] A. Aysu, C. Patterson, and P. Schaumont, "Low-cost and area-efficient fpga implementations of lattice-based cryptography," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 81–86, 2013.

[9] O. Regev, "The learning with errors problem (invited survey)," in *2010 IEEE 25th Annual Conference on Computational Complexity*, pp. 191–204, 2010.

[10] W.-K. Lee and S. O. Hwang, "High throughput implementation of post-quantum key encapsulation and decapsulation on gpu for internet of things applications," *IEEE Transactions on Services Computing*, vol. 15, no. 6, pp. 3275–3288, 2022.

[11] Z. Chen, E. Karabulut, A. Aysu, Y. Ma, and J. Jing, "An efficient non-profiled side-channel attack on the crystals-dilithium post-quantum signature," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pp. 583–590, 2021.

[12] 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)*, pp. 1–5, 2021.

[13] M. Bisheh Niasar, R. Azarderakhsh, and M. Mozaffari Kermani, "High-speed ntt-based polynomial multiplication accelerator for post-quantum cryptography," pp. 94–101, 06 2021.

[14] T. Plantard, "Efficient word size modular arithmetic," in *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, IEEE, 2021.

[15] 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*, pp. 1–14, 2023.

[16] Z. Ni, A. Khalid, W. Liu, and M. O'Neill, "Towards a lightweight crystals-kyber in fpgas: an ultra-lightweight bram-free ntt core," *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2023.

[17] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, p. 297–297, May 1965.

[18] W. M. Gentleman and G. Sande, "Fast fourier transforms," *Proceedings of the November 7-10, 1966, fall joint computer conference on XX - AFIPS '66 (Fall)*, 1966.

[19] O. Özerk, C. Elgezen, A. C. Mert, E. Öztürk, and E. Savaş, "Efficient number theoretic transform implementation on gpu for homomorphic encryption," *J. Supercomput.*, vol. 78, p. 2840–2872, feb 2022.

[20] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," pp. 124–139, 10 2016.

[21] P. L. Montgomery, "Modular multiplication without trial division," vol. 44.

[22] A. C. Mert, E. Öztürk, and E. Savaş, "Design and implementation of a fast and scalable ntt-based polynomial multiplier architecture," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pp. 253–260, 2019.

[23] Xilinx, "7 series fpgas configurable logic block: User guide," 2023.

[24] D. e Shahwar Kundi, Y. Zhang, C. Wang, A. Khalid, M. O'Neill, and W. Liu, "Ultra high-speed polynomial multiplications for lattice-based cryptography on fpgas," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, p. 1993–2005, Oct 2022.