ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

HARDWARE DESIGN OF K^2RED MODULAR MULTIPLICATION ALGORITHM USED IN NUMBER THEORETIC TRANSFORM FOR POST QUANTUM CRYPTOGRAPHY AND HOMOMORPHIC ENCRYPTION

M.Sc. THESIS

Furkan CAN

Department of Electronics and Communication Engineering

Electronics Engineering Programme

FEBRUARY 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

HARDWARE DESIGN OF K^2RED MODULAR MULTIPLICATION ALGORITHM USED IN NUMBER THEORETIC TRANSFORM FOR POST QUANTUM CRYPTOGRAPHY AND HOMOMORPHIC ENCRYPTION

M.Sc. THESIS

Furkan CAN
(504211209)

Department of Electronics and Communication Engineering

Electronics Engineering Programme

Thesis Advisor: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

FEBRUARY 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

POST KUANTUM KRİPTOGRAFİ VE HOMOMORFİK ŞİFRELEME İÇİN SAYI TEORİK DÖNÜŞÜMÜNDE KULLANILAN K^2RED MODÜLER ÇARPMA ALGORİTMASININ DONANIM TASARIMI

YÜKSEK LİSANS TEZİ

Furkan CAN
(504211209)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Sıddıka Berna ÖRS YALÇIN

ŞUBAT 2024

Furkan CAN, a M.Sc. student of ITU Graduate School student ID 504211209 successfully defended the thesis entitled "HARDWARE DESIGN OF K^2RED MODULAR MULTIPLICATION ALGORITHM USED IN NUMBER THEORETIC TRANSFORM FOR POST QUANTUM CRYPTOGRAPHY AND HOMOMORPHIC ENCRYPTION", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Sıddıka Berna ÖRS YALÇIN** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Sıddıka Berna ÖRS YALÇIN** Istanbul Technical University	.....
	Assis. Prof. Dr. Mehmet Tahir SANDIKKAYA Istanbul Technical University	.....
	Assoc. Prof. Dr. Özgü CAN Ege University	.....

Date of Submission : **5 January 2024**
Date of Defense : **22 February 2024**

To my family,

FOREWORD

First of all, I would like to thank my supervisor, Prof. Dr. Berna Örs Yalçın, who guided and supported this thesis study with her extensive knowledge and experience and for her guidance, advice, and help were endless throughout my MSc studies. I want to thank my family for their support. I always felt the moral and motivational support they gave me in moments of despair. Lastly, I want to thank my colleagues and managers at TÜBİTAK BİLGEM and for their valuable supports.

February 2024

Furkan CAN
(Electronics and Communication Engineer)

TABLE OF CONTENTS

ABBREVIATIONS

RSA	**:** Rivest-Shamir-Adleman
ECC	**:** Elliptic Curve Cryptography
NIST	**:** National Institute of Standards and Technology
PQC	**:** Post-Quantum Cryptography
NTT	**:** Number Theoretical Theorem
FPGA	**:** Field-Programmable Gate Arrays
CRYSTALS	**:** Cryptographic Suite for Algebraic Lattices
LWE	**:** Learning with Error
MLWE	**:** Module Learning with Error
PKE	**:** Public Key Encryption

SYMBOLS

mod	:	Modulus of any number
A^T	:	Transpose of A matrix
$k \times l$	:	Matrix multiplication
B_η	:	Binomial distribution
$U(x)$	:	Uniform distribution
$Z_q[X]/(X^n+1)$	:	Ring polynomial
pk	:	Public Key
sk	:	Secret Key
$GF(q)$	:	Galois Field
$>>$	:	Shift Right
$<<$	:	Shift Left

LIST OF TABLES

Page

LIST OF FIGURES

Page

HARDWARE DESIGN OF K^2RED MODULAR MULTIPLICATION ALGORITHM USED IN NUMBER THEORETIC TRANSFORM FOR POST QUANTUM CRYPTOGRAPHY AND HOMOMORPHIC ENCRYPTION

SUMMARY

In light of the rapid advancements in quantum computing, security researchers delve into the field of post-quantum cryptography, aiming to address the imminent threat posed to conventional cryptographic systems. As quantum machines approach practical realization, the security underpinnings of current cryptographic protocols are at risk of compromise. By evaluating the strengths and weaknesses of these emerging cryptographic techniques, the goal is to contribute to the development of robust, quantum-resistant cryptographic hardware systems that ensure the ongoing security and privacy of digital communications in the post-quantum era. Hence, the National Institute of Standards and Technology (NIST) has undertaken the Post-Quantum Cryptography Standardization Process with the objective of devising algorithms resilient to quantum attacks. Currently, this process has progressed to its concluding stage, featuring four key encapsulation mechanisms and three digital signature methods.

If we were to discuss the motivation for starting the study; it is aimed at implementing cryptographic algorithms proposed to prevent security vulnerabilities arising from the high processing capacity of quantum computers, which are expected to be increasingly used in the future and in the digital realm, on current hardware to ensure digital security. In line with this objective, work has begun to implement the algorithms identified as a result of the call opened by NIST on Field Programmable Gate Array (FPGA) hardware, and subsequently, it focuses on ensuring the seamless integration of selected post-quantum encryption algorithms into modern devices by recognizing the increasing efficiency needs in modular arithmetic operations in PQC and Homomorphic Encryption algorithms.

The prevailing techniques in modular arithmetic predominantly rely on Barrett's and Montgomery's reduction methods, complemented by a few proposed alternatives aiming to supplant conventional reduction approaches. These modular arithmetic algorithms find application in cryptographic protocols like Rivest-Shamir-Adleman (RSA), Elliptic Curve Cryptography (ECC), among others. However, the advent of Quantum Computing techniques has instilled concern among researchers regarding data privacy.

Given the escalating data privacy challenges, especially with the proliferation of network-connected devices, the imperative is to integrate post-quantum cryptography into every available device. The modular reduction modules (Barrett, Montgomery, Plantard, K^2RED) employed in PQC are also required to utilize hardware resources with optimal efficiency.

In the studies carried out in the thesis, a literature research was conducted on the algorithms determined in the field of PQC. During these researches, algorithms based on the lattice-based cryptography method were implemented. These algorithms must be implemented on hardware according to security needs.

It was considered to be done on FPGA because of its ability to perform parallel processing, and the design processes are software-based. At the same time, literature studies suggest that it is more suitable to implement lattice-based algorithms on FPGA. The implementation of the lattice-based algorithms selected by NIST and also the lattice-based encryption method used in Homomorphic encryption has begun. At this stage, a common step was encountered while following the steps of the algorithms. This step is the Number Theoretic Theorem (NTT) step, a type of Fast Fourier Transform made in Galois Field, and they share commonalities in terms of mathematical definition. Since the NTT process has the highest processing load among these algorithms, optimizations in this step (such as acceleration, area reduction) play a key role in increasing the overall performance of the system.

When we look at the basis of this process, it is seen that there are basic mathematical operations such as multiplication, addition, subtraction, and modular reduction. Studies on this type of processes reveal that the modular reduction methods used today and innovative studies have been carried out. One of these methods is the K^2RED algorithm. When other modular reduction algorithm studies are examined, different circuit designs for the K^2RED algorithm have been obtained according to various circuit needs. For example, it has been observed that a different circuit can be designed for applications where the maximum clock frequency of the circuit is crucial, while a different circuit can be used for another application where space utilization is important. In the last part, the values of the K^2RED algorithm were tested by adapting it to the most optimal design within the scope of this thesis, using an NTT kernel found in the literature. The results obtained are presented in the thesis content.

This research conducts a comparative analysis by implementing existing methods of modular arithmetic algorithms on FPGA. Simultaneously, it compares the results obtained with a different modular reduction found in the literature through this modular reduction method, and it provides circuit solutions that meet different requirements based on parameters such as speed, area, and delay. The K^2RED algorithm, a reduction method presented in the literature, is transformed into multiplication operations using DSP blocks and is utilized for CRYSTAL-Dilithium and Homomorphic encryption processes, presenting an original study within the scope of this thesis. Following the improvements obtained from these results, this reduction method is compared with a different modular reduction method in NTT operation. According to the results of the study, it is observed that the performance result is better in terms of area usage, and as a result, the algorithm comparison obtained aims to guide future applications of encryption algorithms to increase data privacy and security across various devices.

POST KUANTUM KRİPTOGRAFİ VE HOMOMORFİK ŞİFRELEME İÇİN SAYI TEORİK DÖNÜŞÜMÜNDE KULLANILAN K^2RED MODÜLER ÇARPMA ALGORİTMASININ DONANIM TASARIMI

ÖZET

Kuantum bilgisayar teknolojisinin hızlı gelişmeleri ışığında, donanımsal kriptografi araştırmacıları geleneksel şifreleme sistemlerine yönelik oluşan yakın tehdidi ele almak amacıyla post-kuantum kriptografi alanında çalışmalar yapmaktadırlar. Kuantum bilgisayarlar pratik uygulamaya yaklaştıkça, mevcut şifreleme protokollerinin güvenlik temelleri tehlike altında kalacaktır. Bu yeni şifreleme tekniklerinin güçlü ve zayıf yönlerini değerlendirerek, amacımız, devam eden dijital iletişimin post-kuantum dönemindeki güvenliğini ve gizliliğini sağlayan donanımda gerçeklenebilir, kuantum dirençli şifreleme donanımlarının geliştirilmesine katkıda bulunmaktır. Bu nedenle, Ulusal Standartlar ve Teknoloji Enstitüsü (NIST), kuantum saldırılarına dayanıklı algoritmalar tasarlamayı amaçlayan Post-Kuantum Şifreleme Standartlaştırma Sürecini başlatmıştır. Bu çağrının yapılmasının ardından ilk etapta 69 farklı algoritma sunulmuştur. Bu başvurular NIST'in belirlediği komitenin değerlendirmesine sunulmuştur. Bu komite güvenlik, donanımda gerçeklenebilme, kuantum dirençli olması bakımından algoritmaları değerlendirmiştir. Şu anda, bu süreç dört anahtar kapsülleme mekanizması ve üç dijital imza yöntemini içeren sona erme aşamasına ilerlemiştir. NIST'in bu çağrıyı açmasının önemli bir diğer nedeni ise bu alanda yapılacak çalışmalarla kuantum çağının güvenliği konusunda bir farkındalık oluşturmaktır.

Çalışmaya başlama motivasyonundan bahsedecek olursak; gelecekte kullanımı artacağı ve dijital ortamda kullanılacağı düşünülen kuantum bilgisayarların yüksek işleme kapasitesiyle oluşacak güvenlik zaafiyetinin önlenmesi için ortaya konulan kriptografik algoritmaların günümüz donanımlarında gerçeklenmesi ve bu sayede dijital güvenliğin sağlanmasıdır. Bu amaçla NIST'in açtığı çağrı sonucunda belirlediği algoritmaların FPGA donanımı üzerinde gerçeklenmesi için çalışmaya başlanılmıştır ve devamında PQC ve Homomorfik Şifreleme algoritmalarındaki modüler aritmetik işlemlerinde artan verimlilik ihtiyacını tanıyarak, seçilen post-kuantum şifreleme algoritmalarının modern cihazlara sorunsuz entegrasyonunu sağlamak üzerine odaklanmaktadır.

Modüler aritmetikte yaygın olarak kullanılan teknikler genellikle Barrett ve Montgomery'nin indirgeme yöntemlerine dayanmakta, bunlara ek olarak geleneksel indirgeme yöntemlerini değiştirmeyi amaçlayan önerilen birkaç alternatif bulunmaktadır. Bu modüler indirgeme algoritmaları, Rivest-Shamir-Adleman (RSA), Eliptik Eğri Şifreleme (ECC) gibi şifreleme protokollerinde de kullanılmaktadır. Ancak, Kuantum Bilgisayar tekniklerinin ortaya çıkması, araştırmacılarda veri gizliliği konusunda endişeye yol açmıştır. Bu sebeple araştırmacılar yeni ve etkin modüler indirgeme yöntemleri geliştirmek üzere çalışmalar yapmaktadır.

Artan veri gizliliği zorlukları, özellikle ağa bağlı cihazların yaygınlaşmasıyla birlikte, Post-Kuantum Kriptografiyi mevcut tüm dijital aygıtlara entegre etmenin zorunlu hale gelmesine yol açmaktadır. Aksi takdirde kuantum bilgisayarların kullanımının yaygınlaşmasıyla üçüncü şahıslar mevcut güvenlik algoritmalarını çok kısa zaman periyotlarında aşabilecektir. Post-Kuantum Kriptografi'de ve diğer kriptografik algoritmalarda kullanılan modüler indirgeme modülleri (Barrett, Montgomery, Plantard, K^2RED) aynı zamanda donanım kaynaklarını en etkili şekilde kullanmalıdır. Bunun nedeni ise günümüzde ağa bağlanan cihazların hayatımızda geniş yer edinmesidir. Bu cihazlar hem maliyet bakımından hem de alan bakımından fazla yer kaplamamaktadır. Bu sebeple kuantuma dirençli algoritmaların olabilecek en hızlı, en az donanım kullanan veya en az alan kaplayan ve en maliyet etkin bir biçimde tasarlanması gerekmektedir. Kuantum bilgisayarların ortaya çıkmasıyla bu cihazların güvenliği sağlanması için bahsedilen bu kriterler mümkün olduğunca sağlanmalıdır. Aksi durumda veri güvenliği ağa bağlı tüm cihazlarda sürekli risk altında olacaktır.

Tezde yapılan çalışmalarda, ilk olarak, Post-Kuantum Kriptografi alanında belirlenen algoritmalar üzerine bir literatür araştırması yapılmıştır. Bu araştırmalar sırasında, gerek NIST'in başlattığı çağrıya dönüş olarak yapılan çalışmalarda çoğunlukla tercih edilmesi gerekse NIST'in finale kalan son 7 algoritmanın 5'inin kafes tabanlı şifreleme yöntemine dayalı algoritmaların seçilmesi bu yöntemin geliştirmeye açık olduğu görülmüştür. Bu algoritmaların güvenlik ihtiyaçlarına göre donanımda uygulanması gerekmektedir. Bu sebeple donanımda gerçekleme çalışmalarının yapılarak bu alanda ortaya çıkan ihtiyaçları ve darboğazların görülmesi ve bunlara karşı çözüm yollarının geliştirilmesi amaçlanmıştır.

Bu işlemi FPGA üzerinde gerçekleştirmek düşünülmüştür çünkü paralel işleme yapma yeteneğine sahiptir ve donanım tasarım süreçleri yazılım tabanlıdır. Aynı zamanda, literatür çalışmalarında, kafes tabanlı algoritmaların FPGA üzerinde gerçeklenmesinin yaygın olduğu görülmüştür. NIST tarafından seçilen kafes tabanlı algoritmaların ve homomorfik şifrelemede kullanılan kafes tabanlı şifreleme yönteminin uygulanması başlamıştır. Bu aşamada, algoritmaların adımlarını takip ederken ortak bir adımla karşılaşılmıştır. Bu adım Sayı Teorik Teorem (NTT) adı verilen bir adımdır. Bu, Galois Alanında yapılan bir tür Hızlı Fourier Dönüşümdür ve matematiksel tanım açısından ortaklıklara sahiptir çünkü NTT işlemi, bu algoritmalarda en çok kullanılan birimlerdendir ve bu adımda yapılacak optimizasyonlar (hızlandırma, alan azaltma gibi) sistem genelinde performansı arttırmada kilit bir rol oynamaktadır.

Bu işlemin temeline inildiğinde ise çarpma, toplama, çıkarma ve modüler indirgeme gibi temel matematiksel işlemler olduğu görülmüştür. Bu tip işlemlerle ilgili yapılan çalışmalara bakıldığında günümüzde kullanılan modüler indirgeme yöntemlerinin ve yenilikçi çalışmaların yapıldığı görülmüştür. Bu yöntemlerden biri de K^2RED algoritmasıdır. Diğer modüler indirgeme algoritma çalışmaları incelenip bakıldığında çeşitli devre ihtiyaçlarına göre farklı devre tasarımları K^2RED algoritması elde edilmiştir. Örneğin devrenin maksimum saat frekansının çok önemli olduğu uygulamalar için farklı bir devre tasarımı yapılırken alan kullanımının önemli olduğu bir başka uygulama için ise farklı bir devre kullanılabileceği gözlenmiştir. Son kısımda ise literatürde bulunan bir NTT çekirdeği kullanarak K^2RED algoritmasının değerleri

bu tez kapsamında tasarlanan en optimum tasarım ile adapte edilerek test edilmiştir. Elde edilen sonuçlar tez içeriğinde verilmiştir.

Bu araştırma, modüler aritmetik algoritmalarındaki mevcut yöntemlerin FPGA üzerinde uygulanmasıyla karşılaştırmalı bir analiz yapar. Aynı zamanda bu modüler indirgeme yöntemiyle literatürde bulunan farklı bir modüler indirgemeyle elde edilen sonuçları karşılaştırır ve hız, alan ve gecikme gibi parametrelere göre farklı ihityaçları karşılayan devre çözümleri sunmaktadır. Literatürde sunulan bir indirgeme yöntemi olan K^2RED algoritması DSP blokları kullanılarak çarpma işlemine dönüştürülüp CRYSTAL-Dilithium ve Homomorfik şifreleme işlemi için de kullanılarak özgün bir çalışma yapılarak bu tez kapsamında sunulmuştur. Bu sonuçlardan elde edilen geliştirmelerden sonra NTT işleminde bu indirgeme yöntemi kullanılarak farklı bir modüler indirgeme yöntemiyle karşılaştırılmıştır. Sonuçlara göre yapılan çalışmayla birlikte alan kullanımına göre performans sonucunun daha iyi olduğu görülmüştür ve sonuç olarak elde edilen algoritma karşılaştırması, şifreleme algoritmalarının sonraki uygulamalarına rehberlik etmek amacıyla, veri gizliliğini ve güvenliği çeşitli cihazlar üzerinde kullanımı artırmayı hedefler.

1. INTRODUCTION

The growing prevalence of social networks and the heightened connectivity through network channels in today's world have underscored the increasing significance of data privacy [9]. The potential advent of quantum computers and the rising demand for data have given rise to concerns related to privacy and security issues.

To address this challenge, research in cryptography has witnessed a surge in recent years. The anticipated arrival of quantum computers has amplified the demand for more robust cryptographic algorithms compared to contemporary ones such as RSA, ECC, etc. [10,11]. Algorithms that are resistant to quantum computers and classical computers need to be developed and the development of new algorithms in the realm of post-quantum cryptography is currently being shaped through a competition organized by the National Institute of Standards and Technology(NIST). As part of this contest, selected algorithms are subsequently published [12].

Among the chosen algorithms, this thesis places emphasis on CRYSTALS-Kyber [13], CRYSTALS-Dilithium [14] and Homomorphic Encryption (HE) [15]. While CRYSTALS-Kyber and CRYSTALS-Dilithium algorithms are examined under the Post-Quantum Cryptography (PQC) subheading, Homomorphic encryption is a different encryption method generally used in cloud storage processes. The common point is the Number Theoretical Theorem (NTT) process used in PQC and HE algorithms [16,17]. Modular reduction methods are used to optimize this process.

In the realm of modular arithmetic, the NTT stands out as a crucial operation [16]. NTT serves as an efficient method for conducting polynomial multiplications, and modular arithmetic plays a pivotal role within NTT, particularly in executing multiplication operations [18].

The modular reduction algorithm serves as a fundamental component for modular multiplication and modular exponentiation. These operations hold significant

importance in various asymmetric encryption schemes, including the extensively employed RSA algorithm [19] and the CRYSTALS-Kyber algorithm. The efficiency of these modular reduction algorithms is critical in handling the expansion of numbers during computations, ensuring that computations remain both manageable and efficient while upholding security standards [16].

Proposed by Niasar et al. in their work cited as [20], the K^2RED algorithm performs modular multiplication operations employing prime numbers identified as Proth numbers [21]. They used the K^2RED reduction method in the CRYSTALS-Kyber algorithm and made their transactions only for this algorithm. While performing these operations, they designed the circuit using the shift and addition method. In this thesis, the K^2RED reduction method was extended for CRYSTALS-Dilithium and HE, and the implementations were designed using DSP blocks. Additionally, different alternatives are offered according to different circuit requirements (area, time, delay). The results obtained were compared with the Plantard reduction method and a performance comparison was made.

If we look at NTT, there are many NTT implementations in the literature. While there are different NTT implementations, the NTT used in this thesis is a completely parametric NTT. In the reference NTT implementation, an NTT core with an input length of 32 bits and used for HE was used. The number of butterfly units used and many parameters can be changed. This NTT core from Mert uses the Word-Level Montgomery algorithm as a modular reduction method [8]. With the results of this implementation, new results were obtained by replacing our own design, K^2RED, with Word-Level Montgomery. In this way, K^2RED was used for the first time for the NTT core used in HE and the results were compared with the Word-Level Montgomery algorithm. According to the results obtained, the same performance could be achieved by using less area at the same clock frequency.

If we delve into the motivation behind initiating this study, it's centered on implementing cryptographic algorithms aimed at mitigating security risks posed by the anticipated proliferation of quantum computers in the future digital landscape. These algorithms are being adapted for current hardware to ensure digital security.

To achieve this goal, efforts have commenced to deploy the algorithms specified by NIST on FPGA hardware. Subsequently, upon commencement of the study, attention was drawn to optimizing the Number Theoretic Transform (NTT) operation, which is widely utilized in these algorithms. To this end, an analysis of existing profiling studies in the literature was conducted. Notably, research indicates that the NTT operation ranks as the second most utilized operation in the CRYSTALS-Kyber algorithm [22]. Thus, improvements in this operation would directly impact system performance. Investigations were carried out on modular multiplication, addition, subtraction, and reduction operations to optimize the NTT operation. As part of this effort, implementation outcomes were attained by employing the K^2RED modular reduction operation across various PQC algorithms and HE schemes.

This project involves the analysis and implementation of a NTT and modular reduction algorithms on an Field-Programmable Gate Arrays (FPGA). Comparison of area, time and delay parameters of hardware implementations of different modular reduction modules and NTT implementation with this modular reduction method are explained. To execute this implementation, Xilinx Vivado and HLS tools are employed [23].

1.1 Purpose of Thesis

The aim of this thesis is to contribute to the hardware implementation of new algorithm implementation studies aimed at making currently used information security systems dysfunctional thanks to the data processing speed of quantum computers or quantum computing tools that are expected to come into our lives in the future. In addition, HE, another encryption system used in cloud systems today, is also within the scope of this thesis. The focus is on accelerating the NTT process used in HE and algorithms determined by NIST to be resistant to the data processing power of quantum computers. In this context, comparison studies were made on the performance of the hardware implementations of the modular reduction operations in the NTT process in terms of area, time and latency.

1.2 Organization of Thesis

Figure 1.1 : IBM Quantum Computer at CES 2020 [3]

In the first chapter, there is an introduction section about the thesis. In this section, information is given about the studies carried out and the general template of the thesis. Then, the studies carried out and the purpose of the thesis are given. Then, the theoretical and mathematical background of PQC and HE and the algorithms used are given. Then, theoretical information about the NTT process was given. Sub-blocks and schemes related to this process are explained. In the next chapter, the studies and results related to modular reduction are shared. In the next chapter, NTT implementation and a comparison with an implementation in the literature are made. Algorithms and diagrams related to the operations performed are given. In the last section, information about the results and future work is given.

2. MATHEMATICAL BACKGROUND

In this chapter, we briefly introduce the basics of PQC and HE, the main topic of the thesis. Section 2.1 explains PQC background and section 2.2 explains HE. In the following sections, essential background information about PQC algorithms and examine the existing alternative solutions.

2.1 Post-Quantum Cryptography

In the dynamic landscape of modern cryptography, the advent of quantum computing casts a shadow over conventional cryptographic systems. Quantum machines, with their exponentially enhanced computational power, threaten to unravel the very fabric of security protocols that have long been the basis of digital communication. As we stand at the precipice of this quantum revolution, PQC emerges as a critical field, not merely as a conservative response but as a proactive effort to fortify our digital fortresses against the impending quantum attacks.

Traditional cryptographic systems, anchored in the computational complexity of mathematical problems such as integer factorization and discrete logarithms, face a shift in their security landscape. Quantum algorithms, notably Shor's algorithm, loom large as potential disruptors, challenging the robustness of our current cryptographic infrastructure [10,11]. In this context, this section undertakes a deep dive into the mathematical foundations of PQC, seeking not only to adapt to the quantum challenge but to redefine the very essence of cryptographic resilience.

The NIST PQC project, initiated in 2016, involves a global effort to standardize cryptographic algorithms resilient to both classical and quantum threats [12]. Through a rigorous evaluation process and public competition, diverse candidate algorithms spanning lattice-based, hash-based, code-based, and multivariate polynomial-based approaches have been submitted [24]–[27]. The ongoing project, marked by iterative rounds of analysis and feedback, aims to select and standardize a small

set of robust PQC algorithms for various cryptographic applications. While the finalization of standards is pending, the initiative underscores the importance of preparing cryptographic systems for the advent of quantum computers, with the implications extending to public-key encryption, digital signatures, and key exchange in a post-quantum era. As of the current stage, the NIST Post-Quantum Cryptography (PQC) project is in its concluding phase, with four Key Exchange Mechanisms (KEM) and three digital signature methods still under consideration. Notably, five of the remaining seven candidates are rooted in Lattice-Based Cryptography (LBC), highlighting the prominence of this approach among various techniques. The finalization of these standards signifies a critical step in fortifying cryptographic systems against the evolving landscape of quantum threats, with implications for securing key exchange and digital signatures in a post-quantum era. In the next subsections, some fundamental cryptographic methods are given a quick look that, as far as we know, are still safe from quantum attacks. These methods are quite complex, and since we're not covering them in detail here, we will just touch on them briefly. They serve as the basic building blocks for understanding how we can protect information, laying the groundwork for exploring more advanced cryptographic techniques within the limits of this thesis.

2.1.1 Lattice-based cryptography

A lattice is formally described as the linear combination of integer multiples of linearly independent base vectors, with these vectors being termed the lattice's basis. The security foundations of lattice-based encryption schemes rest upon the computational hardness of lattice problems. Presently, there exists no discerned quantum or classical algorithm that can efficiently solve these intricate problems. The security of this method relies on the difficulty of solving certain math problems involving these dot patterns, such as finding the shortest path between points or dealing with errors in the data. These are named with Shortest Vector Problem (SVP) and Learning with Errors (LWE). Cryptographic algorithms constructed upon lattice structures exhibit notable efficiency, demonstrating rapid execution and a commendable suitability for parallel

computation. The combination of these attributes cements LBC as the foremost approach within the realm of PQC.

2.1.2 Code-based cryptography

Code-based cryptography involves safeguarding information through the utilization of unique codes designed to rectify errors. Comparable to tools employed in digital communication, these codes ensure the accurate transmission of information, even in the presence of errors. The essence of code-based cryptography lies in the challenge of comprehending or "decoding" these particular codes, and it is the complexity of this task that strengthens its effectiveness in ensuring the security of information. The most well-known code-based cryptography method is McEliece, introduced in 1978. Its security remains intact against both quantum and classical computer attacks.

2.1.3 Hash-based cryptography

In the domain of digital signatures, hash-based schemes are implemented, capitalizing on the security derived from the formidable challenge of reconstructing the original input string from an output generated by a hash function. Originally proposed by Lamport in 1975, his scheme encountered a notable limitation—specifically, a constrained capacity for securely generating multiple signatures with the same keys. To overcome this restriction, Merkle introduced his method in 1979 [83], substantially augmenting the permissible number of trusted key usages. However, this enhancement came at the expense of an increased key and signature size. Despite the augmented size, Merkle's method succeeded in significantly expanding the practicality of hash-based digital signature schemes.

2.1.4 Multivariate polynomial cryptography

Multivariate cryptography is a type of public-key cryptography that is based on the difficulty of solving systems of multivariate polynomial equations over finite fields. Unlike traditional public-key cryptosystems such as RSA or ECC, multivariate cryptography relies on the complexity of solving systems of equations in multiple variables rather than number-theoretic problems. Multivariate cryptography is

commonly employed in the formulation of public-key encryption schemes. In this context, the public key comprises a system of multivariate polynomial equations, while the private key consists of the solutions to these equations. The process of encrypting a message entails solving a system of equations, whereas decryption necessitates possession of the private key, representing the solution set. The Rainbow algorithm, founded on multivariate equations, stands out as one of the finalists in the PQC standardization process for digital signature methods.

2.2 Homomorphic Encryption

In today's expansive digital landscape, where safeguarding data privacy is paramount, homomorphic encryption emerges as a revolutionary cryptographic concept. It tackles the fundamental challenge of conducting computations on encrypted data without the need for decryption, a departure from traditional methods that expose data vulnerabilities during this process. Unlike conventional approaches, homomorphic encryption allows mathematical operations to be performed directly on encrypted data, ensuring the confidentiality of sensitive information throughout the entire processing pipeline.

Essentially, homomorphic encryption maintains data confidentiality by executing mathematical operations on ciphertexts, generating results that remain encrypted. Craig Gentry formalized this groundbreaking concept in 2009, leading to the development of various homomorphic encryption schemes [28,29]. These range from partially homomorphic systems, supporting specific operations, to fully homomorphic encryption (FHE), enabling arbitrary computations.

Homomorphic encryption's strength lies in achieving a delicate balance between data usability and security. By facilitating computations on encrypted data, it empowers organizations and individuals to harness the capabilities of cloud computing and data analytics without compromising sensitive information privacy. This makes homomorphic encryption particularly relevant in scenarios where outsourcing data processing to untrusted servers, such as in cloud computing, is a necessity.

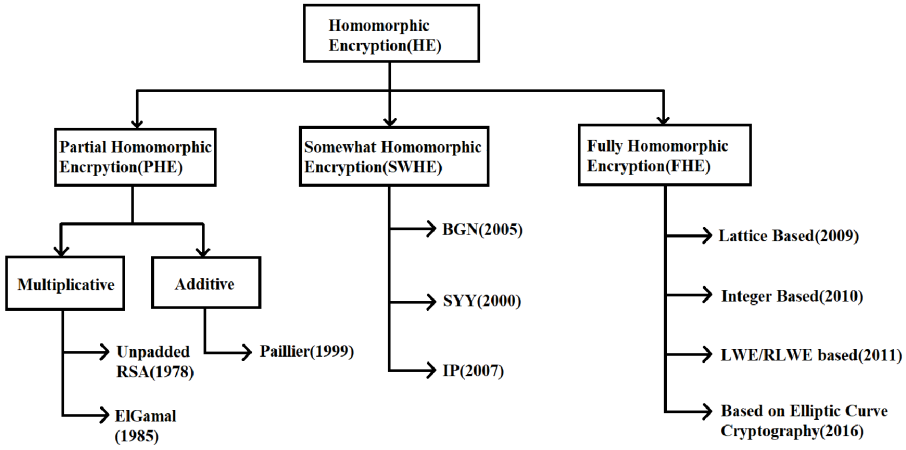


Figure 2.1 : Homomorphic Encryption subsection figure [4]

Despite the progress in FHE schemes, homomorphic computation remains challenging for many cloud applications. Ongoing research is actively working to enhance the performance of the fundamental arithmetic components within existing FHE schemes. Various implementations and constructions have been developed to introduce practical hardware and software realizations of FHE schemes, including HElib [30] , NFLlib [31], and cuHe [32]. In the pursuit of making FHE more practical, the focus is on improving the performance of the most time-consuming arithmetic component found in many FHE schemes: large degree polynomial multiplication and modular reduction of the multiplications.

We will explore FHE, distinguishing it from other encryption methods based on its unique capability to perform unlimited types of operations on encrypted data. Unlike other encryption approaches, FHE stands out by allowing a broad range of operations to be carried out on encrypted data without limitations in frequency.

If we make a mathematical representation of homomorphic encryption [33]:

$$Encryption(a) = A, Encryption(b) = B \tag{2.1}$$

$$Encryption(a) + Encryption(b) = A + B = Encryption(a + b) \tag{2.2}$$

Homomorphic encryption excels in performing computations on encrypted data without the need to disclose the private key. Typically, when a client wants to execute operations, sharing the secret key with the service provider becomes a potential threat to data accessibility. Homomorphic encryption addresses this concern by enabling computation on cloud data without the necessity of decryption. Consequently, the secret key remains secure, and only the client retains the capability to decode the data, ensuring exclusive access to sensitive information [8].

In Lattice-Based Fully Homomorphic Encryption schemes, a key high-level operation involves multiplying two polynomials with exceptionally large degrees. Recent literature has seen numerous publications focusing on the multiplication of these large-degree polynomials, particularly emphasizing NTT-based multiplication schemes. These schemes are recognized for providing highly efficient algorithms tailored for the multiplication of polynomials with large degrees, contributing to the advancement of lattice-based homomorphic encryption.

In this thesis, the parameters of the NTT process employed in lattice-based homomorphic encryption, such as area, time, and delay, will be explored using various modular reduction methods on FPGA hardware. This investigation aims to assess and compare the efficiency of different modular reduction techniques in facilitating HE operations through an optimized NTT process. The subsequent section will provide a detailed explanation of the NTT process.

2.3 CRYSTALS-Kyber Algorithm

Kyber, a component of the Cryptographic Suite for Algebraic Lattices (CRYSTALS), stands as a post-quantum key exchange mechanism relying on the security of the Module-LWE (MLWE) problem [34,35]. Additionally, the Kyber algorithm has progressed to the third round of evaluation in the PQC algorithm assessment conducted by the NIST. This recognition underscores the algorithm's standing and potential for adoption as a secure post-quantum cryptographic solution. Its approach involves a conventional construction method wherein it initially establishes an IND-CPA public key encryption (PKE) scheme. Subsequently, Kyber transforms this scheme into an IND-CCA KEM using the modified Fujisaki-Okamoto transform [36,37]. This two-step process ensures the integrity and security of the key exchange mechanism within the framework of post-quantum cryptography [38].

The security foundation of Kyber relies on the Module-LWE problem, formally defined as follows:

$$(A, b = A^T s + e) \tag{2.3}$$

In this context, a hidden vector, denoted as s, is sampled from a binomial distribution centered around $B_\eta(R_q{}^{kx1})$; in this case, k represents the dimension of the fundamental Module-LWE problem. The public matrix A is randomly sampled from a uniform distribution denoted by $U(R_q{}^{kxk})$. The decisional Module-LWE problem asserts that indistinguishable (A, b) from a uniformly random sample [39].

CRYSTALS-Kyber utilizes the polynomial ring $R_q = Z_q[X]/(X^n + 1)$ with q = 3329 and n = 256 across all parameter sets where $n = 2^{n'-1}$ such that $X^n + 1$ is the $2^{n'-1}$-th cyclotomic polynomial. Each coefficient of the polynomials in Rq can be represented as a 16-bit integer, enabling 16-bit NTT polynomial multiplication. CRYSTALS-Kyber employs this structure to build a Chosen-Ciphertext Attack (CCA)-secure KEM from a Chosen-Plaintext Attack (CPA)-secure public key encryption (PKE). The CRYSTALS-Kyber PKE (IND-CPA) comprises Key Generation Algorithm (1), Encryption Algorithm (2) and Decryption Algorithm (3)

Table 2.1 : Parameter sets for CRYSTALS-Kyber.

	n	k	q	η_1	η_2	d_u, d_v
Kyber512	256	2	3329	3	2	10,4
Kyber768	256	3	3329	3	2	10,4
Kyber1024	256	4	3329	2	2	11,5

and parameter given in 2.1. Matrix-vector multiplication, specifically $\hat{A} \circ NTT(s)$ and $A^T \circ \hat{t}$, where each element of the matrix and vector is a polynomial, forms the core operation in key generation and encryption for CRYSTALS-Kyber. For a more comprehensive understanding, readers are directed to CRYSTALS-Kyber's specification [40].

Before encryption or decryption in the CRYSTALS-Kyber algorithm, data undergoes preprocessing to achieve a regular form, involving compression, decompression, sampling, rejection, encoding, and decoding. The compression function transforms an element data $X \in Z_q$ to an integer within $(d < log_2(q))$, and the decompression function performs the reverse operation. In CRYSTALS-Kyber, the superimposed error matrix is sampled from a centered binomial distribution (CBD) [41], ensuring each coefficient from the polynomial $f \in R_q$ is appropriately sampled. Subsequently, these coefficients undergo modulo operation by a rejection function. The encoding process involves serializing polynomials into byte arrays, and decoding translates byte streams back into polynomial vectors.

Algorithm 1 Kyber.PKE Key Generation [40]

Output: $pk = (\hat{b}, \rho), sk = \hat{s}$

1: $seed \leftarrow (0, ..., 255)^{32}$
2: $\rho, \sigma \leftarrow SHAKE256(64, seed)$
3: $\hat{A} \leftarrow GenMatrixA(\rho)$
4: $s \leftarrow SampleVec(\sigma, 0)$
5: $e \leftarrow SampleVec(\sigma, 1)$
6: $\hat{b} \leftarrow \hat{A} \circ NTT(s) + NTT(e)$
7: **return** $pk = (\hat{b}, \rho), sk = \hat{s}$

Algorithm 2 Kyber.PKE Encryption [40]

Input: $pk = (\hat{b}, \rho), message \ \ \mu \in R_q, seedcoin \in (0,1,...,255)^{32}$

Output: $Ciphertext(u', h)$

1: $\hat{A} \leftarrow GenMatrixA(\rho)$
2: $s' \leftarrow SampleVec(coin, 0)$
3: $e' \leftarrow SampleVec(coin, 1)$
4: $e'' \leftarrow SampleVec(coin, 2)$
5: $\hat{t} \leftarrow NTT(s')$
6: $u \leftarrow NTT^{-1}(\hat{A}^T \circ \hat{t}) + e'$
7: $v' \leftarrow NTT^{-1}(\hat{b}^T \circ \hat{t}) + e'' + \mu$
8: **return** $(u' = Compress(u), h = Compress(v'))$

Algorithm 3 Kyber.PKE Decryption [40]

Input: $Ciphertext \ c = (u', h), secret \ key \ sk = \hat{s}$

Output: $Message \ \mu \in R_q$

1: $u \leftarrow Decompress(u')$
2: $v' \leftarrow Decompress(h)$
3: **return** $\mu = v' - NTT^{-1}(\hat{s}^T \circ NTT(u))$

KEM, which provides CCA security, is also implemented in the Kyber algorithm. Key generation is the same as CPA. The only difference is that sk contains $pk = (t, \rho)$ and the secret contains 256 bits of random value. Let $G : \{0:1\}^* \rightarrow \{0,1\}^{2\times 256}$ and $H : \{0:1\}^* \rightarrow \{0,1\}^{256}$ be hash functions. Algorithm 1, 4 and 5 are shows CCA secure KEM in CRYSTALS-Kyber.

Algorithm 4 Kyber.Encaps$(pk = (t, \rho))$ [40]

1: $m \leftarrow \{0,1\}^{256}$
2: $(\hat{K}', r') \leftarrow G(H(pk), m)$
3: $(u, v) \leftarrow Kyber.CPA.Enc((t, \rho), m; r)$
4: $c \leftarrow (u, v)$
5: $K \leftarrow H(\hat{K}, H(c))$
6: **return** (c, K)

Algorithm 5 Kyber.Decaps$(sk = (s,z,t,\rho), c = (u,v))$ [40]

$m' \leftarrow Kyber.CPA.Dec(s,(u,v))$
$(\hat{K}', r') \leftarrow G(H(pk), m')$
$(u', v') \leftarrow Kyber.CPA.Enc((t,\rho), m'; r')$
if $(u', v') = (u, v)$ **then**
  return $K \leftarrow H(\hat{K}', H(c))$
else
  return $K \leftarrow H(z, H(c))$
end if

As seen in the CRYSTALS-Kyber algorithm, NTT operation is used for polynomial multiplication. The NTT process will be explained in detail in the next chapter. In this section, the CRYSTALS-Kyber algorithm is given in general terms.

2.4 CRYSTALS-Dilithium Algorithm

The CRYSTALS-Dilithium signature is a component of the CRYSTALS, which includes both a KEM called Kyber and a signature scheme known as CRYSTALS Dilithium [14]. The security of both protocols hinges on the computational difficulty of the Module variant of the Learning With Error (MLWE) problem [34,35]. In the MLWE problem for CRYSTALS-Dilithium, A is a matrix of polynomials with dimensions $k \times l$, while s and e become vectors of dimensions l and k, respectively. Informally, the MLWE problem can be conceptualized as the Ring-LWE problem, where the single ring elements (a and s) are substituted with module elements over the same ring.

CRYSTALS-Dilithium employs the MLWE problem with fixed values for n and q. The security level of this signature scheme varies by simply adjusting the dimensions of the matrix A, denoted as k and l. Since $\mathfrak{R}_q$ remains the same for all security levels, it becomes feasible to optimize operations in $\mathfrak{R}_q$ to enhance the efficiency across all security levels. This flexibility facilitates easy adjustment of security levels. CRYSTALS-Dilithium defines four parameter sets: weak, medium, recommended, and very high, corresponding to matrix dimensions (k,l) = (3, 2), (4, 3), (5, 4), and (6, 5), respectively [42].

Dilithium comprises three algorithms: Key Generation, Signing, and Verification, outlined in Algorithms 6, 7, and 8, respectively. Each algorithm utilizes a set of internal functions that are employed multiple times and/or across various phases.

Algorithm 6 outlines the Key Generation process, consisting of two main parts: (1) the expansion of two random *seeds*, ρ and K, into A and $(s1, s2)$, respectively, using the extendable output function $SHAKE-128$, and (2) the calculation of the remaining components for both public and secret keys. $SHAKE-256$ serves as the collision-resistant hash (CRH) in all algorithms. It's worth noting that A is directly presented in the NTT domain representation, wherein its elements are polynomials represented as vectors [1].

Algorithm 6 Key Generation [1]

1: $\rho, K \leftarrow (0,1)^{256}$
2: $(s_1, s_2) \in {S_\eta}^l \; x \; {S_\eta}^k := ExpandMask(K)$
3: $A \in {R_q}^{kxl} := ExpandA(\rho)$
4: $t := As_1 + s_2$
5: $(t_1, t_0) := Power2Round(t, d)$
6: $tr \in (0,1)^{384} := CRH(\rho || t_1)$
7: **return** $pk = (\rho, t_1), sk = (\rho, K, tr, s_1, s_2, t_0)$

In Algorithm 7, during the Signing process, we create a bunch of special numbers called y, then do some math with them (Ay). We focus on certain parts of those numbers (w_1, the "high-order" bits) and mix them with the message M. We use a bunch of helper functions like $Power2Round_q$, $Decompose_q$, $HighBits_q$, and $LowBits_q$ to pick the right parts (w_1) correctly. Other functions, $MakeHint_q$ and $UseHint_q$, help us put together the missing bits for later checking in the Verification step. This clever method lets us make the public key smaller by about 2.5 times, but it adds a bit more data to the signature. We use $SHAKE-256$ for creating these special numbers (y) through the *ExpandMask* function and in another step called H.

Finally, Algorithm 8 illustrates the Verification step. The verifier calculates w_1' and approves the signature z if it meets the criteria of being small enough (refer to the original article [1] for more details).

Algorithm 7 Signing Sign(sk, M) [1]

$A \in R_q^{kxl} := ExpandA(\rho)$
$\mu \in (0,1)^{384} := CRH(tr||M)$
$\kappa := 0, (z,h) := \perp$
while $(z,h) := \perp$ **do**
  $y \in S^l_{\gamma_1 - 1} := ExpandMask(K||\mu||\kappa)$
  $w := Ay$
  $w_1 := HighBits_q(w, 2\gamma_2)$
  $c \in B_{60} := H(\mu||w_1)$
  $z := y + cs_1$
  $(r_0, r_1) := Decompose_q(w - cs_2, 2\gamma_2)$
 if $\|z\|_\infty \geq \gamma_1 - \beta$ *or* $\|r_0\|_\infty \geq \gamma_2 - \beta$ *or* $r_1 \neq w_1$ **then**
  $(z,h) := \perp$
else
$h := MakeHint_q(-ct_0, w - cs_2 - ct_0, 2\gamma_2)$
if $\|ct_0\|_\infty \geq \gamma_2$ *or the number of 1's in h is greater than w* **then**
$(z,h) := \perp$
end if
end if
$\kappa = \kappa + 1$
end while
return $\sigma = (z,h,c)$

Algorithm 8 Verification Verify [1]

$A \in R_q^{kxl} := ExpandA(\rho)$
$\mu \in (0,1)^{384} := CRH(CRH(\rho||t_1)||M)$
$w_1' := UseHint_q(h, Az - ct_1 \cdot 2^d, 2\gamma_2)$
return $\|z\|_\infty < \gamma_1 - \beta$ *and* $c := H(\mu||w_1')$ *and number of 1's in h is* $\leq w$

Table 2.2 : Dilithium Parameters. [1]

Scheme	NIST Level	(k,l)	η	τ	γ_1	γ_2
Dilithium2	2	(4,4)	2	39	2^{17}	$(q-1)/88$
Dilithium3	3	(6,5)	4	49	2^{19}	$(q-1)/32$
Dilithium5	5	(8,7)	2	60	2^{19}	$(q-1)/32$

3. NUMBER THEORETIC TRANSFORM

PQC aims to forge cryptographic algorithms impervious to the computational might of quantum computers [17]. A leading contender in this realm is lattice-based cryptography, leveraging the complexity of lattice problems to fortify security [24]. 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 [13]:

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 $f\ \varepsilon\ R_q$ 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

$$X^{256} + 1 = \prod_{i=0}^{127}(X^2 - \zeta^{2i+1}) = \prod_{i=0}^{127}(X^2 - \zeta^{2br_7(i)+1}) \tag{3.1}$$

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

$$(fmodX^2 - \zeta^{2br_7(0)+1}, ..., fmodX^2 - \zeta^{2br_7(127)+1}) \tag{3.2}$$

$$NTT(f) = \hat{f} = \hat{f}_0 + \hat{f}_1 X + ... + \hat{f}_{255} X^{255} \tag{3.3}$$

with

$$\hat{f}_{2i} = \sum_{j=0}^{127} f_{2j} \zeta^{(2br_7(i)+1)j} \tag{3.4}$$

$$\hat{f}_{2i+1} = \sum_{j=0}^{127} f_{2j+1} \zeta^{(2br_7(i)+1)j} \tag{3.5}$$

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 (3.2), utilizing the definitions for $\hat{f}_i$ provided in (3.4) and (3.5). That is,

$$NTT(f) = \hat{f} = (\hat{f}_0 + \hat{f}_1 X, ..., \hat{f}_{254} + \hat{f}_{255} X^{255}) \tag{3.6}$$

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.

$$\hat{h}_{2i} + \hat{h}_{2i+1} X = (\hat{f}_{2i} + \hat{f}_{2i+1} X(\hat{g}_{2i} + \hat{g}_{2i+1} X) mod X^2 - \zeta^{2br_7(i)+1} \tag{3.7}$$

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 [43] and Gentleman-Sande [44] 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 [5].

3.1 Number Theoretic Transform Algorithm

Modular reduction algorithms are used during NTT process in CRYSTALS-Kyber, CRYSTALS-Dilithium and Homomorphic encryption algorithms. NTT operation is a mathematical method that reduces the computational load when multiplying very large polynomials with each other. Modular reduction algorithms are also used in the NTT process. The NTT operation is a two-way operation, and the polynomials are first passed into the NTT domain. Bit lengths that increase as a result of multiplication in the pass phase are converted to lower bit lengths by modular reduction algorithms. Then, the INTT(Inverse Number Theoretical Transform) operation is applied and the

number domain of the input polynomials is returned again. For this reason, it is important to use and examine modular reduction algorithms for the NTT process.

Number Theoretic Transform is a powerful operation for computing polynomial multiplication and convolution operations [45]. It is an appliance of Fast Fourier Transform (FFT) [46]. NTT is more specifically FFT in Galois Field.

Aforementioned NTT is mathematical operation to apply polynomial modular multiplication with a selected prime number. It has significant applications in post quantum cryptography, signal processing, etc. [47].

Algorithm 9 Number Theoretic Transform (NTT)

Require: Polynomial $A(x) \in Z_P[x], A(x) = a[k-1]x^{k-1} + a[k-2]x^{k-2} + \cdots + a[0]$
Require: Primitive k-th root of unity $\omega \in Z_P$
Require: $q \equiv 1(\mod 2k)$,
Ensure: $NTT(A(x))$

$A \leftarrow \text{BitReverse}(A)$
for $s = 1$ **to** $\log_2(k)$ **do**
  $m \leftarrow 2^s$
  $\omega_m \leftarrow \omega^{\frac{k}{m}}$
  for $k = 0$ **to** $k-1$ **step** m **do**
    $\omega_k \leftarrow 1$
    for $j = 0$ **to** $\frac{m}{2} - 1$ **do**
      $t \leftarrow \omega_k \cdot A[k + j + \frac{m}{2}] \mod q$
      $u \leftarrow A[k + j]$
      $A[k + j] \leftarrow u + t \mod q$
      $A[k + j + \frac{m}{2}] \leftarrow u - t \mod q$
      $\omega_k \leftarrow \omega_k \cdot \omega_m \mod q$

NTT takes polynomials as normal order but returns it in reversed order. For example (0,1,2,3) to (0,2,1,3). To avoid that, bit reversal applied in the beginning of Algorithm 9. As in FFT, NTT can be implemented using Cooley-Tukey structure [43] and Gentleman-Sande [44] butterflies inside. Using these butterflies, NTT can be implemented by a "divide-and-conquer" method which reduces the time complexity of polynomial multiplication from $O(n^2)$ to $O(n \cdot \log n)$. The NTT operation can be seen in Fig. 3.1.

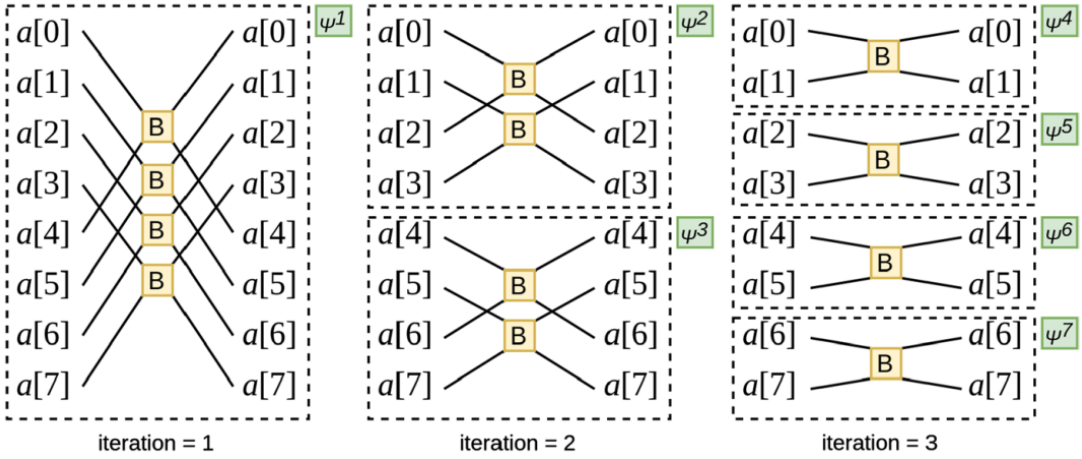


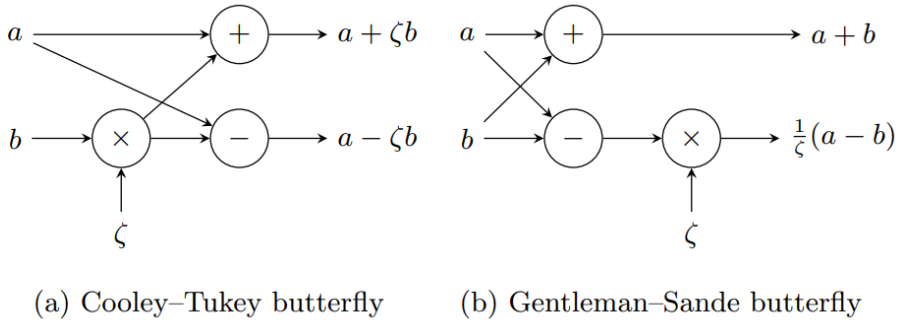
Figure 3.1 : Basic NTT Operation for $k = 8$ [5]

In the Figure 3.1, basic operation that is done by NTT is shown. NTT multiplies input in every stage with ω_n which are precalculated coefficients that is explained in the algorithm.

As seen in the Algorithm 9, involved concepts in the algorithm includes Cooley-Tukey and Gentleman-Sande butterfly operations, ω and arithmetic in finite field.

In the Algorithm 9, between the steps 8-12 needs of modulo operation can be seen. Given pseudo-code of NTT, since number in line 11 and 12 cannot be bigger than $2 \cdot q$, in this part we will focus on line number 9. If line 9 is inspected, $\omega_k \cdot A$ can be between 0 and q^2. Therefore, the algorithm needs a fast and reliable reduction method to reduce the number to $GF(q)$.

As seen in NTT Algorithm 9, ω_m what is called "twiddle factor" is constant that can be precomputed when modulo algorithm is applied. That twiddle factor is crucial for correctness of algorithm. In the algorithm, polynomials are represented as sequence of coefficients. Also, algorithm needs modular arithmetic to be used when multiplying a number with a constant. Therefore in this study, change this precomputed twiddle factor with number that is needed by used modular arithmetic algorithm.



(a) Cooley–Tukey butterfly (b) Gentleman–Sande butterfly

Figure 3.2 : Cooley-Tukey and Gentleman-Sande butterfly units. [6]

3.2 Butterfly Unit

The essential operations in NTT and INTT are carried out in the butterfly unit. There are typically two types of butterfly structures: the Cooley-Tukey (CT) butterfly and the Gentleman-Sande (GS) butterfly. In Kyber, a common approach is to use the CT butterfly in NTT and the GS butterfly in INTT to prevent coefficients from flipping. However, recent reports suggest that employing the CT butterfly for both NTT and INTT in LBC schemes could result in faster code [6,48].

The structure shown in Figure 3.3 shows addition, subtraction and multiplication operations. The structure encountered in the literature is given in figure 3.2. Here, the value given by ξ is the constant values, while q is the prime value used. Other inputs are external values to be encrypted.The values resulting from these processes are then subjected to the modular reduction process. While the Cooley Tukey form is used for the NTT transaction, the Gentleman-Sande form is used for the INTT transaction, which is the opposite of the NTT transaction. The reason is that the Gentleman-Sande form gives the terms reversely in its output. In the NTT process, the process can be accelerated by using the butterfly unit structure in parallel, but it can only be used a certain number of times due to space problems in hardware implementations. Acceleration of the algorithm can be achieved with multiplier modules and modular reduction modules.

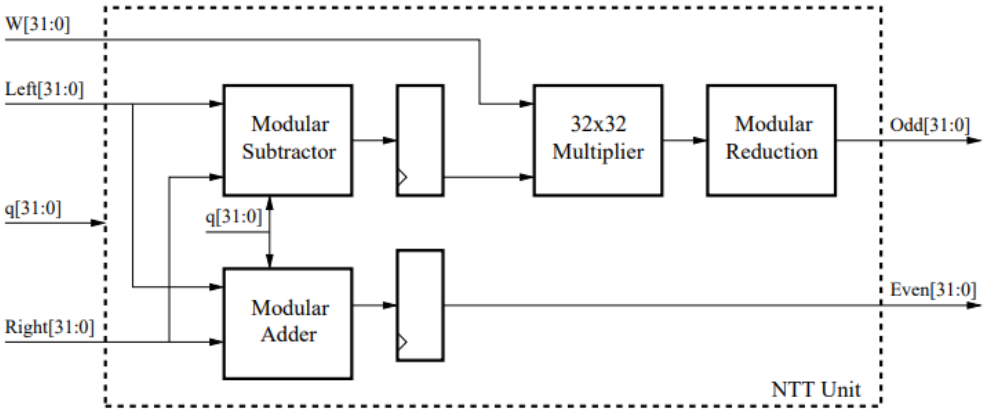


Figure 3.3 : 32 bit NTT Butterfly Unit. [7]

3.3 Modular Multiplication Algorithms

Following a 32-bit multiplication operation, the outcome must be reduced to match the bit-length of the modulus. To enhance speed and efficiency in a scalable architecture, we have made modifications to butterfly units or more specifically modular reduction and modular multiplication operations, tailoring it to perform a rapid and effective modular reduction operation.

For this reason, with the modular multiplication blocks we present in this section, the NTT process used in LBC algorithms (CRYSTALS-Kyber, CRYSTALS-Dilithium, Lattice Based Homomorphic Encryptions etc.) can be used according to different circuit requirements. Specifically, the K^{2}RED algorithm will be explained in detail in the next chapter.

3.3.1 K^{2}RED Modular Multiplication Algorithm

Consider two integers a and b, both ranging from $[0,q)$, where q is a special kind of number defined as $k^2 \cdot 2^m + k \cdot 2(m+1) + 1$. Now, let C be the product of a and b, an integer. The objective is to reduce C modulo q by leveraging the unique structure of q, specifically the property that $k^2 \cdot 2^m \equiv -1(modq)$. Express C as the sum of C_0 and $2^m \cdot C_1$, where $0 \leq C_0 < 2^m$. Then, $0 \leq C_1 = (C - C0)/2^m < k^2 \cdot 2^m + 2k + 1/(2^m) =$

$kq+k+1/(2^m)$. By considering these bounds, it follows that $kC \equiv kC_0 - C_1 (mod q)$, and the absolute value of $kC_0 - C_1$ is bounded by $|kC_0 - C_1| < (k+1/(2^m))q$. Given that k is a small integer, the value $kC_0 - C_1$ can be adjusted into the range $[0, q)$ by adding or subtracting a small multiple of q. The maximum value for C is $(q-1)^2 = k^2 \cdot 2^m$, in which case $C_0 = 0$ and $C_1 = k \cdot 2^m = k(q-1)$. This implies that $(k-1)q$ must be added to $kC_0 - C_1$ to complete the reduction.

Algorithm 10 K-RED Modular Reduction Algorithm [49]

Input: C, *24 bit integer*

Output: $kC_0 - C_1$ *,12 bit integer*

1: $C_0 \leftarrow C mod 2^m$
2: $C_1 \leftarrow C/2^m$
3: **return** $kC_0 - C_1$

Algorithm 11 K-RED-2x Modular Reduction Algorithm [49]

Input: C, *24 bit integer*

Output: $k^2C_0 - kC_1 + C_2$ *,12 bit integer*

1: $C_0 \leftarrow C mod 2^m$
2: $C_1 \leftarrow C/2^m mod 2^m$
3: $C_2 \leftarrow C/2^{2m}$
4: **return** $k^2C_0 - kC_1 + C_2$

In contrast to alternative modular multiplication techniques, K^{2}RED stands out as a method exclusively applicable to specific prime numbers [20]. Introduced by Niasar et al. in [20], K^{2}RED executes modular multiplication operations using prime numbers recognized as Proth numbers [21]. 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.

The K^{2}RED algorithm, a variant of the Montgomery Modular Multiplication Algorithm [50] introduced in [49], builds upon the K-RED algorithm outlined in [49]. In the K^{2}RED algorithm, the K-RED algorithm [49] is applied twice and executed through an adder and a shifter. In Algorithm 10 and 11 constitute the mathematical infrastructure of the K^{2}RED Algorithm. The values obtained as a result of the NTT operation are reduced with these reduction algorithms, thus avoiding operations with larger bit lengths.

Initially K^{2}RED modular reduction algorithm was proposed for CRYSTALS-Kyber PQC algorithm. In this work we added multiplication in order to extend K^{2}RED reduction algorithm to K^{2}RED modular multiplication algorithm and also we applied it to CRYSTALS-Dilithium PQC algorithm for the first time.

Algorithm 12 K^{2}RED 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: $C' \leftarrow kC_l - C_h$

Algorithm 12 shows the steps of K^{2}RED modular reduction algorithm. In CRYSTALS-Kyber algorithm, $P = 3329$, $m = 8$ and $k = 13$. $P = 8380417$ is given in CRYSTALS-Dilithium [14], we calculated the parameters of K^{2}RED 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 K^{2}RED by k^{-2}. However, since twiddle factors ω_i in the NTT process given as [7] are fixed values, these constants are multiplied by k^{-2} in advance.

3.3.2 Plantard Modular Multiplication Algorithm

Thomas Plantard proposed Plantard reduction algorithm for modulo operations where prime number is smaller than it is used in RSA, ECC etc. in 2021 [51].

Algorithm 13 Plantard Modular Reduction Algorithm.

Require: $q, \Phi = \frac{1+\sqrt{5}}{2}, q < \frac{2^n}{\Phi}, R = q^{-1} \bmod 2^{2n}, A = X \cdot Y \cdot R \bmod 2^{2n}$
Ensure: C with $0 \leq C < q, C = X \cdot Y(-2^{-2n}) \bmod q$
1: $C \leftarrow (\lfloor \frac{A}{2^n} \rfloor) + 1$
2: $C \leftarrow \lfloor \frac{C \cdot q}{2^n} \rfloor$
3: **if** $C = q$ **then**
4: $\quad C \leftarrow 0$

In the Plantard modular reduction algorithm, the algorithm needs the input integer A, modulus q, and inverse modulus $q^{-1} \bmod 2^{2n}$. Plantard modular reduction algorithm that as every algorithm has its advantages for specific primes, most of the algorithms study for bigger primes ($n >> 64$). With this method, as it is seen in Algorithm 13 is able to compute modulo operation with smaller primes with one multiply-add operation. Here one little thing is that n is not equal bit size of prime number but it is $q < \frac{2^n}{\Phi}$ shown in Algorithm 13.

This algorithm output is $C = A \cdot (-2^{-2n}) \mod q$, also multiplied by factor -2^{-2n}. Therefore, algorithm output should be multiplied with 2^{2n} to ensure the output $C = A \mod q$ at the end.

4. IMPLEMENTATION OF K^{2}RED MODULAR MULTIPLICATION

In this study, hardware architectures are designed for K^{2}RED modular multiplication algorithms. Plantard algorithm is chosen as a baseline to compare with proposed K^{2}RED modular multiplication algorithms for performance results. Block diagrams of the proposed hardware architectures are presented in the Fig. 4.1, 4.7.

Previous studies are focused on implementing modular arithmetic on long bits sizes ($n \gg 64$), this study focuses of optimized modular multiplication algorithms for lower bit sizes ($n \ll 64$).

Xilinx xc7a100tcsg324-2L Field programmable Gate Array (FPGA) is used in order to implement modular reduction and/or multiplication algorithms. In the section 4.1, when creating the resulting performance tables, this study uses the Eq. 4.2 and Eq. 4.1.

Here, in Eq. 4.1, P is power and t is time and in Equation 4.2, f is frequency, n is bit size of output. To convert result table in bits to bytes, resultant is divided by 8 as 8 bits to 1 byte. The energy calculation is obtained by performing a post implementation functional simulation in the Xilinx Vivado program by giving 1000 random inputs for all algorithms. In this way, all algorithms run for equal periods of time and energy values are calculated at equal intervals.

$$E = P \cdot t \tag{4.1}$$

$$Throughput = f \cdot n \tag{4.2}$$

Algorithm	T(ns)	Pipeline	E(nJ)	Throughput(GB/s)
CRYSTALS-Kyber	2.4	6	0.296	0.75
CRYSTALS-Dilithium	1.9	6	0.277	1.17
Homomorphic	2.8	6	0.323	1.52

Table 4.1 : Implementation Performance Results of the Design for K^{2}RED Modular Multiplication Algorithm.

4.1 Design and Verification of K^{2}RED

Block diagram of the design is given in Fig. 4.1. As seen in the figure, the data input of the system has been registered 2 times. Since the $A*B-C$ operation will be performed, the output of the multiplication result takes one clock, while the A_h corresponding to the A input has to be delayed by one clock cycle. Hardware module for K^{2}RED is implemented using Xilinx DSP48 Macro IP, which enables the FPGA to use DSP resources. The same steps are applied again to the result obtained in the first step. The bit length of the prime value is masked from the result in second step. This value will be the output of the module [23].

The number of FPGA resources used as a result of the implementation of the K^{2}RED algorithm are given in the Table-4.2. The first thing to note in the table is that the DSP numbers are the same even though the bit lengths of all three 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 64. The inputs of the DSP sources in the FPGA are 24 and 18 bits. The bit lengths used in these three algorithms make it possible to use DSP for this reason. When we look at other sources, the number of LUT(Look Up Table) increases proportionally according to the bit lengths. Likewise, FF(Flip-Flop) numbers also increase according to the order of bit lengths, but there is no exact proportion.

As shown in Fig. 4.1, since the same circuit structure is used for all 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, 23 and 32 for CRYSTALS-Kyber, CRYSTALS-Dilithium and Homomorphic, and the throughput values are calculated with the Eq. 4.2 and using the given bit lengths.

Algorithm	# of LUTs	# of FFs	# of DSPs
CRYSTALS-Kyber	-	70	3
CRYSTALS-Dilithium	-	104	4
Homomorphic	47	190	6

Table 4.2 : Implementation Utilization Results of the Design for K^{2}RED Modular Reduction Algorithm

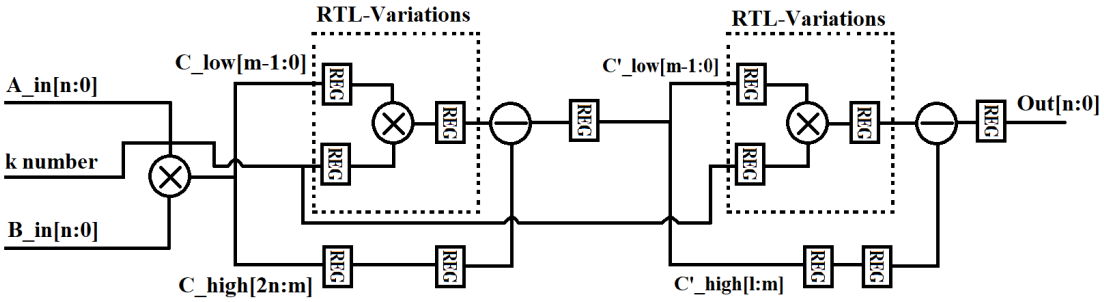


Figure 4.1 : K^{2}RED Modular Multiplication Register Transfer Level(RTL) Diagram.

```
72  begin
73
74  kyber_1: dsp_first_step
75    PORT MAP (
76      CLK => CLK,
77      A => '0' & mult_out(7 downto 0),
78      B => '0' & k,
79      C => '0' & mult_out(23 downto 8),
80      P => c_prime
81    );
82
83  kyber_2 : dsp_second_step
84      PORT MAP (
85        CLK => clk,
86        A => '0' & c_prime(7 downto 0),
87        B => '0' & k,
88        C => c_prime(17 downto 8),
89        P => c2_prime
90      );
91
92  process(clk)
93      begin
94      if(rising_edge(clk)) then
95          mult_out     <= std_logic_vector(unsigned(A_in) * unsigned(Zeta_in));
96  --        data_in_reg <=data_in;
97          data_out     <= c2_prime(11 downto 0);
98      end if;
99      end process;
100 end Behavioral;
```

Figure 4.2 : K^{2}RED Modular Multiplication Register Transfer Level(RTL) Code with Using DSP for CRYSTALS-Kyber Algorithm.

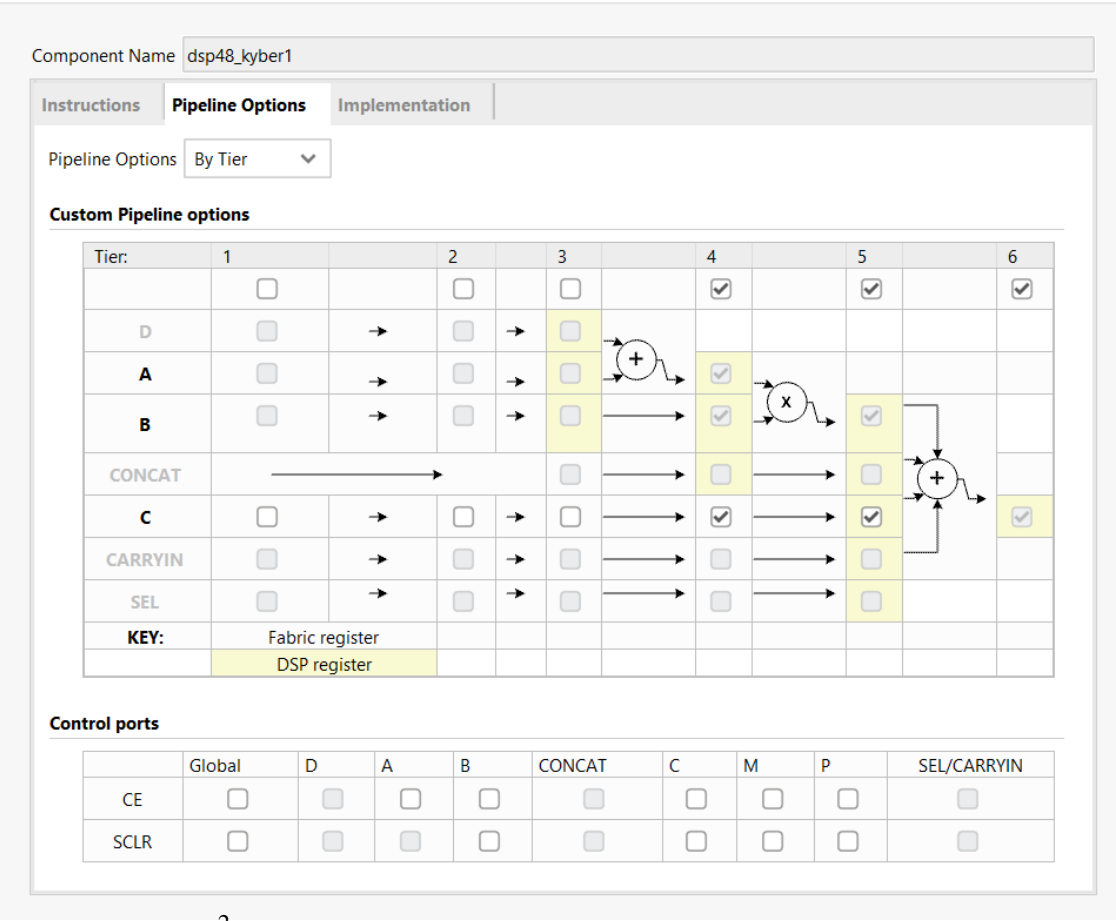


Figure 4.3 : K^2RED Modular Multiplication Register Transfer Level(RTL) Code DSP Configuration for Stage-1.

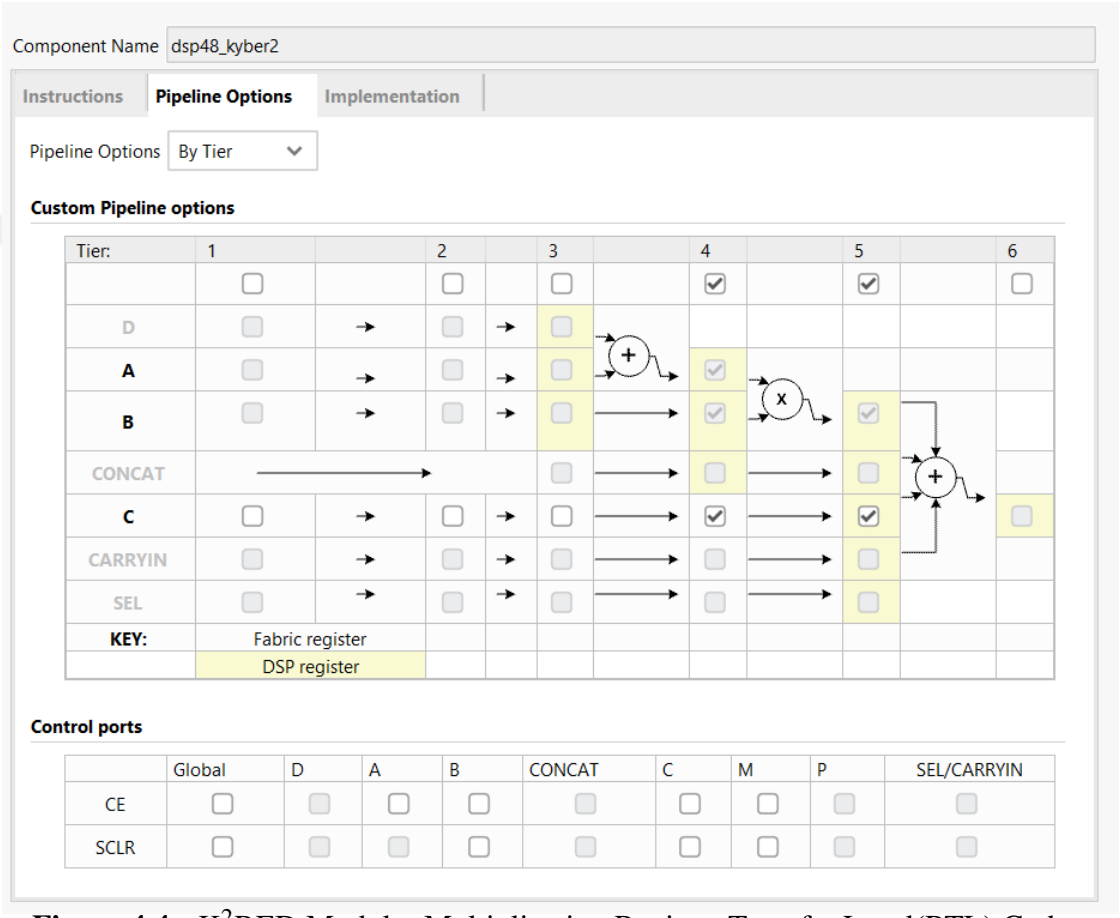


Figure 4.4 : K^2RED Modular Multiplication Register Transfer Level(RTL) Code DSP Configuration for Stage-2.

If we look at the energy comparison, according to the values implemented in Xilinx Vivado, the highest energy consuming circuit is the hardware module designed for Homomorphic encryption, middle is the module for CRYSTALS-Kyber and the lowest is the module for CRYSTALS-Dilithium. Their values are 0.323 nJ, 0.296 and 0.277, respectively.

Power and speed performance values of the K^2RED algorithm is shown in the Table 4.1. Since, input and output bit lengths are less than other algorithms, the hardware module for K^2RED used in CRYSTALS-Kyber algorithm reached the highest operating frequency. It has been observed that it can operate at a clock signal of 2.0 nanoseconds, in other words 500 MHz, due to the smaller footprint and thus the shorter path length.

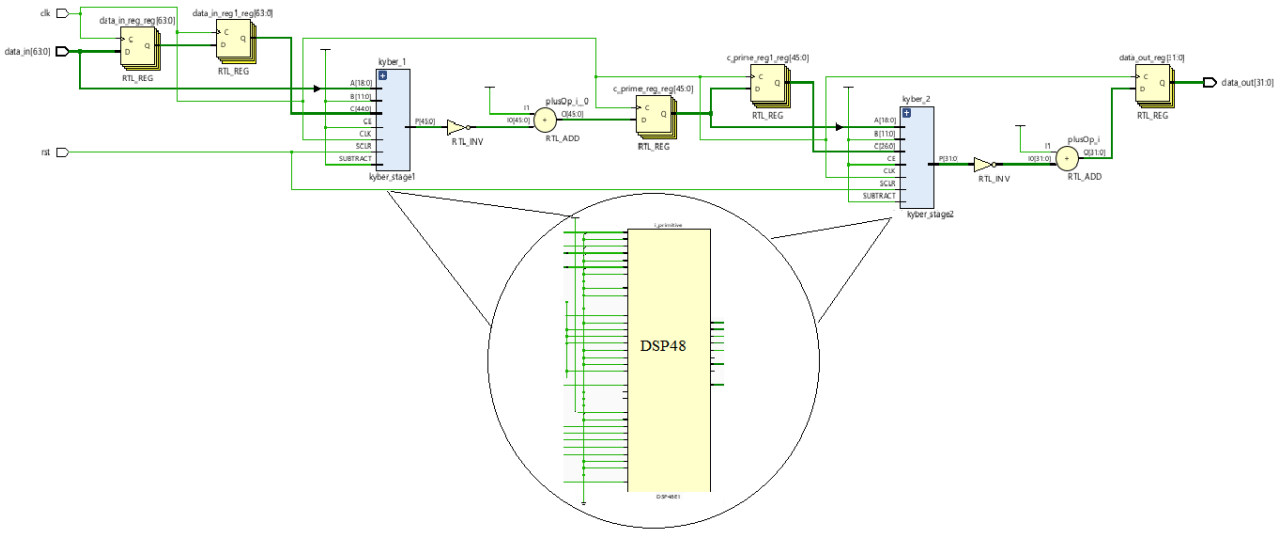


Figure 4.5 : RTL Schematic Diagram of the Design for K^2RED Modular Reduction Algorithm.

The verification of the designed hardware modules was made with simulations. By giving random inputs to each algorithm separately, the pipeline values, the control of whether the result is obtained according to the given input and the boundary conditions are checked. In Fig. 4.6, the results obtained for CRYSTALS-Kyber, CRYSTALS-Dilithium and Homomorphic are given. In all three simulations, it was seen that the data came after a delay of 6 clocks. Then, the input value given is expressed with A, while the result is shown with r. r_expected shows the theoretically expected result. The result of this algorithm should be $C' = k^2 \cdot A \mod P$.

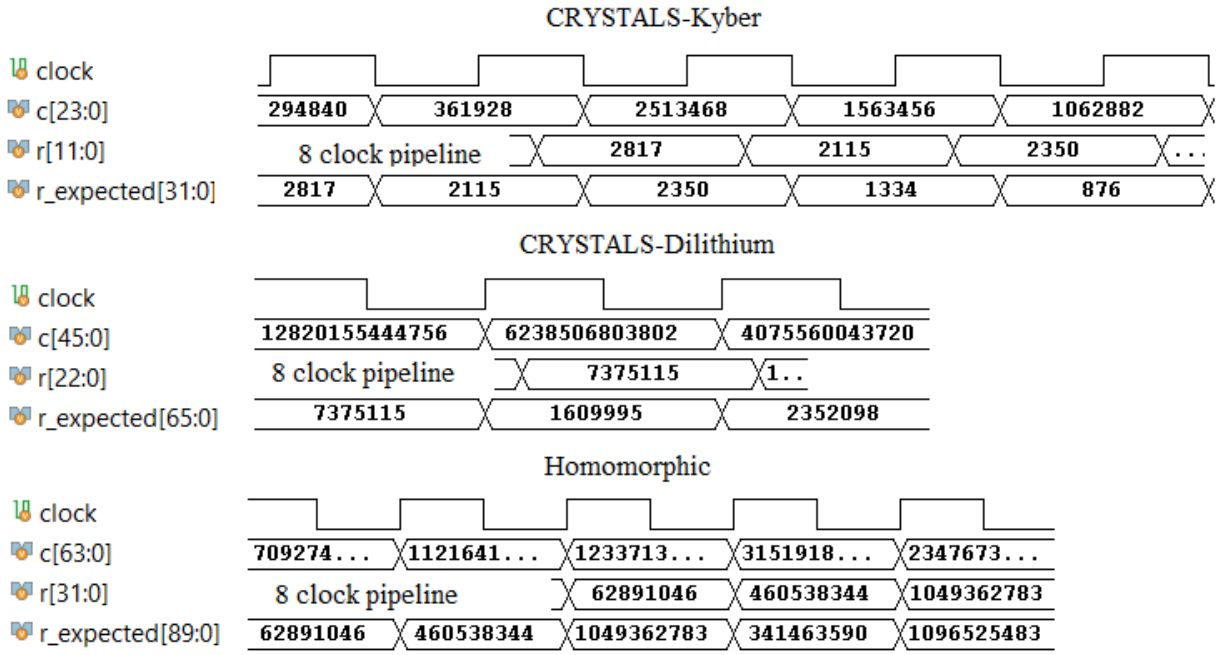


Figure 4.6 : Behaviour Simulation Results of the RTL Design for K^2RED Modular Reduction Algorithm.

??

4.2 RTL Variations and Comparisons

Different implementations of K^2RED modular multiplication are dependent with the design of upper blocks and requirements of the design. In this study, proposed K^2RED modular multiplication is compared and results of this comparison has given in the Table 4.3 and Figure 4.10. These modular multiplication module that are designed is a part of bigger design which can be NTT design, frequencies of different designs of K^2RED and Plantard modular multiplication [2] is highly dependent with the pipeline number of design. Different pipeline numbers given in the Table 4.3 named as 'area', 'opt' and 'performance' which means optimization is taken care of these terms.

In Table 4.3 and Figure 4.10 comparison between K^2RED and Plantard modular multiplication is given. From the Table 4.3, 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 4.10, 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.

There is three different effective RTL Designs. The first of these is the design given in the previous section(Fig. 4.1). The second design shown in Fig. 4.7. In another given design, with the aim of minimizing the delay, the registers in the RTL Variations section given in fig. 4.1 were removed and the multiplication operations were performed by

shifting and adding. With these three different RTL designs, the required design can be selected according to the graphics given in fig. 4.10 For example, when a small FPGA is used as a logic cell, the design to be used will be different from the design to be used in applications where the maximum clock frequency is desired to be as high as possible. As a result, the K^2RED algorithm can be adapted to different purposes and RTL designs can be used.

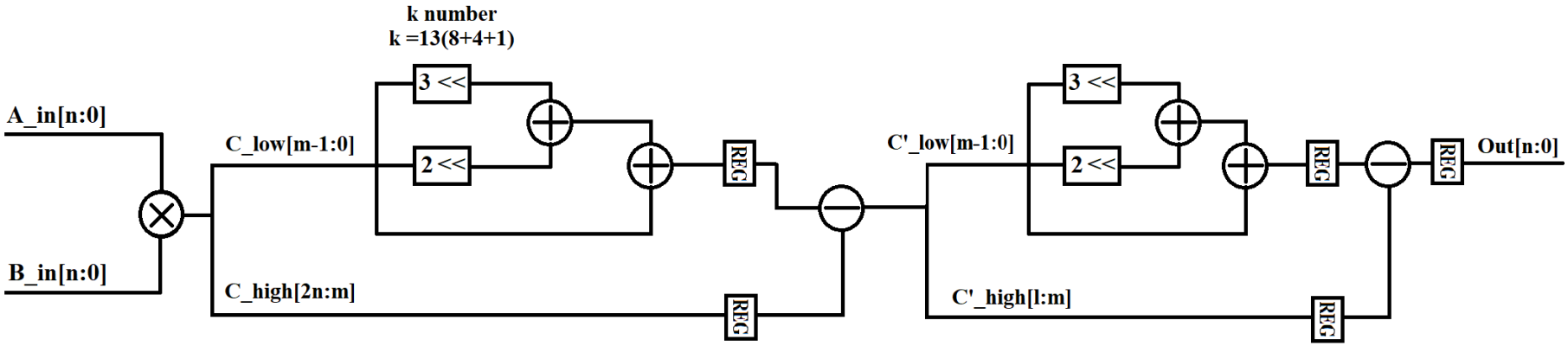


Figure 4.7 : K^2RED Modular Multiplication with Shifting Register Transfer Level(RTL) Diagram.

```
141         data_in_reg2              <= mult_out(7 downto 0) & "000";
142         data_in_reg3              <= mult_out(7 downto 0) & "00";
143         c_prime_reg1(10 downto 0) <= c_prime(7 downto 0) & "000";
144         c_prime_reg2(9 downto 0)  <= c_prime(7 downto 0) & "00";
145
146 --      c_prime_q    <= std_logic_vector( k * unsigned(data_in_reg(7 downto 0)));
147 --      c_second_q   <= std_logic_vector( unsigned(k) * unsigned(c_prime(7 downto 0)));
148
149 process(clk) begin
150       if rising_edge(clk) then
151           mult_out     <= std_logic_vector(unsigned(A_in) * unsigned(Zeta_in));
152           data_out       <= c_second(11 downto 0);
153       end if;
154 end process;
155 end Behavioral;
```

Figure 4.8 : K^2RED Modular Multiplication Register Transfer Level(RTL) Code with Using Shift and Add for CRYSTALS-Kyber Algorithm.

```
101  kyber_step1 : kyber_subtracter1
102      PORT MAP (
103      A => c_prime_q,
104      B => mult_out(23 downto 8),
105      CLK => clk,
106      S => c_prime
107      );
108
109  kyber_step2 : kyber_subtract2
110    PORT MAP (
111      A => c_second_q,
112      B => c_prime(16 downto 8),
113      CLK => clk,
114      S => c_second
115   );
116   adder_step_1 : kyber_adder1_step1
117     PORT MAP (
118       A => data_in_reg2,
119       B => data_in_reg3,
120       S => c_prime_q_reg
121     );
122   adder_step_2 : kyber_adder1_step2
123       PORT MAP (
124         A => c_prime_q_reg,
125         B => mult_out(7 downto 0),
126         S => c_prime_q
127       );
128
129    adder_step_3 : kyber_adder1_step1
130         PORT MAP (
131           A => c_prime_reg1,
132           B => c_prime_reg2,
133           S => c_second_q_reg
134         );
135    adder_step_4 : kyber_adder1_step2
136         PORT MAP (
137           A => c_second_q_reg,
138           B => c_prime(7 downto 0),
139           S => c_second_q
140             );
```

Figure 4.9 : K^2RED Modular Multiplication Register Transfer Level(RTL) Intellectual Property(IP) Instantiation Code with Using Shift and Add for CRYSTALS-Kyber Algorithm.

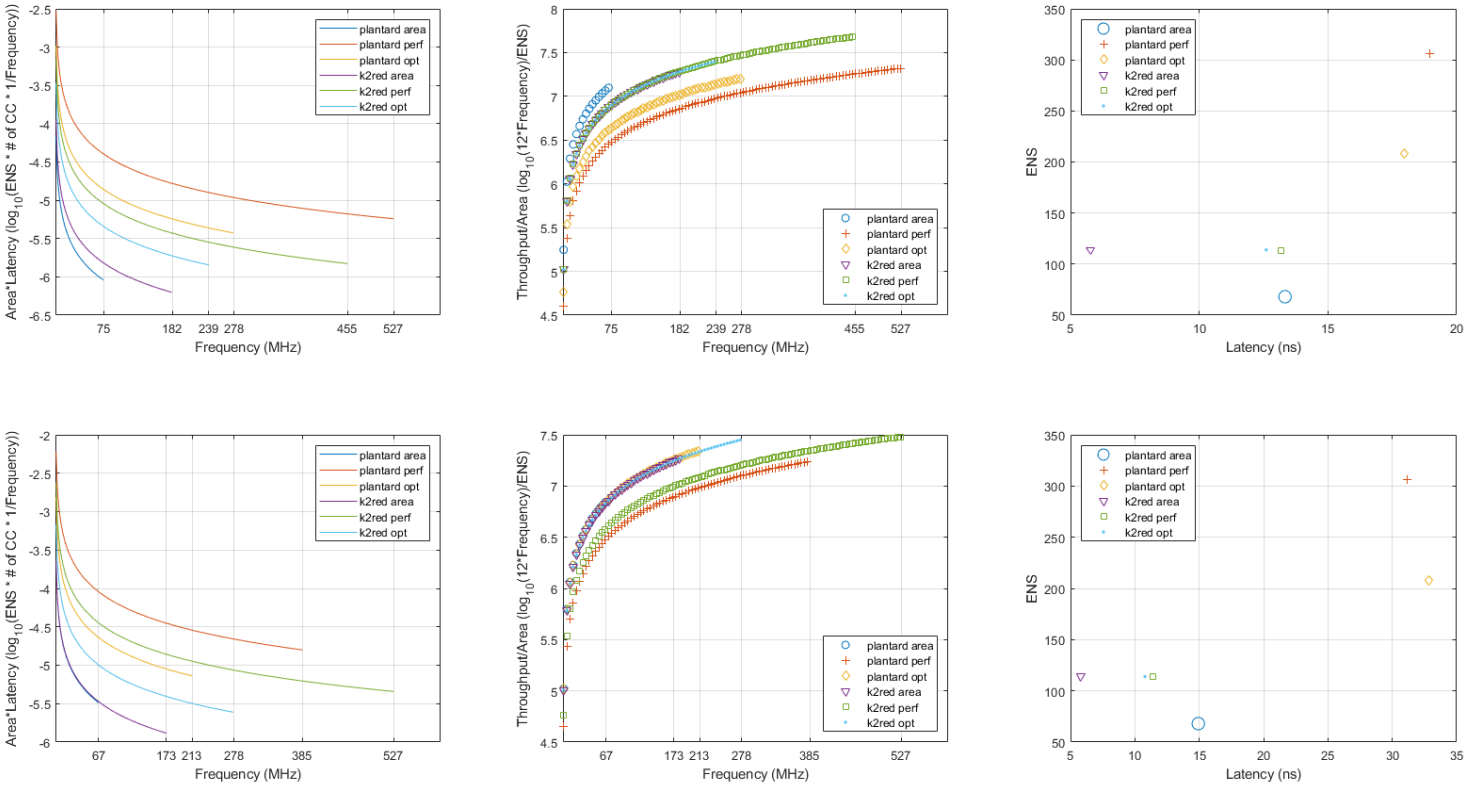


Figure 4.10 : Comparisons of K^2RED and Plantard Modular multiplication.

Table 4.3 : Utilization of different methods used in K^2RED and Plantard modular multiplication algorithm for CRYSTALS-Kyber and CRYSTALS-Dilithium PQC Schemes. [2]

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^2RED	CRYSTALS-Kyber	114	14	54	36	1	5.5	2181.82	1
K^2RED	CRYSTALS-Kyber	113	13	-	70	3	2.4	5454.48	6
K^2RED	CRYSTALS-Kyber	114	14	55	65	1	4.2	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^2RED	CRYSTALS-Dilithium	226	26	149	40	2	8.16	3965.2	1
K^2RED	CRYSTALS-Dilithium	400	0	0	104	4	1.9	12104.9	6
K^2RED	CRYSTALS-Dilithium	226	26	103	130	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 [52].
** ENS (Eqivalant Number of Slices) = Slices + DSPs× 100 + BRAMs× 196 [53].
*** Critical Path Delay (CPD)

5. IMPLEMENTATION OF NUMBER THEORETIC THEOREM

In this section, the parametric NTT core shared by [54] with open source is used for comparison. The number of butterfly units used in this core can be given as a parameter to the system and experiments can be carried out. Here, when the number of butterfly units that will work in parallel is selected as 32 or more, it cannot be synthesized on the FPGA model being studied. For this reason, the results obtained using 16,4,1 are shared.

5.1 Benchmarking Setup

In this section, the performance comparison will be made by replacing the Word-level Montgomery modular multiplication algorithm used in the NTT algorithm with the K^2RED algorithm. NTT, whose source codes are given in [54], is written parametrically. In other words, a comparison can be made between modular multiplication algorithms by changing the number of butterfly units that can work in parallel. Used NTT code, which is compared, operates via 32-bit data. In this work [8], Mert et al. opted to implement a polynomial multiplication architecture with a degree of 1024, intended to achieve a 128-bit security level. Throughout this section, the variable n represents the degree of the polynomial, and q signifies the prime used as the modulus. Instead of fixing the modulus size and the modulus, a scalable architecture supporting modulus lengths between 22 and 32 bits was implemented but using with comparison we use 32 bit prime number.

5.1.1 Core Multiplier

In used structure, modular multiplier architecture is optimized for modulus lengths between 22 and 32 bits. A 32-bit multiplier was designed, as illustrated in fig. 5.1. Focus on an designed FPGA architecture, 16-bit core multipliers were utilized due to limitations in DSP size. On our Artix-7 Architectures, DSP slices comprise

Algorithm 14 Word-Level Montgomery Modular Reduction Algorithm [8]

Input: $C = A \cdot B$, *a 2K-bit positive integer*
Input: M, *a K-bit positive integer*
Input: mu, *w-bit positive integer* $mu = -M^{-1} \mod 2^w$, $w \leq K$
Output: $Res = C \cdot R^{-1} \mod M$ *where* $R = 2^{K+w} \mod M$

1: $L \leftarrow \lceil K/w \rceil$
2: $T1 \leftarrow C$
3: **for** *i from 0 to L - 1* **do**
4: $\quad T2 \leftarrow T1 \mod 2^w$
5: $\quad T3 \leftarrow (T2 \cdot mu) \mod 2^w$
6: $\quad T1 \leftarrow \lfloor (T1 + (T3 \cdot M))/2^w$
7: **end for**
8: $T4 \leftarrow T1 - M$
9: **if** $(T4 < 0)$**then**
10: $\quad Res \leftarrow T1$
11: **else**
12: $\quad Res \leftarrow T4$
13: **end if**

18x25-bit signed multipliers, while on Virtex-7 Architectures in open source design platform, DSP slices include 18x25-bit signed multipliers [55]. To align with existing literature, multiplier was opted to implement for both architectures, necessitating a core multiplier length of 16 bits.

NTT architecture is fully pipelined, making the pipeline registers shown in fig. 5.1 inconsequential to the overall architecture's throughput in terms of clock cycles. This refinement significantly improves the overall performance in terms of seconds.

5.1.2 Modular Reduction

Following a 32-bit multiplication operation, it is necessary to reduce the result back to the bit-length of the modulus. In the pursuit of a scalable architecture, we adapted Algorithm 14 to accomplish a rapid and effective modular reduction operation. For a more optimized design, the following features were used:

$$q \equiv 1 \mod 2n \tag{5.1}$$

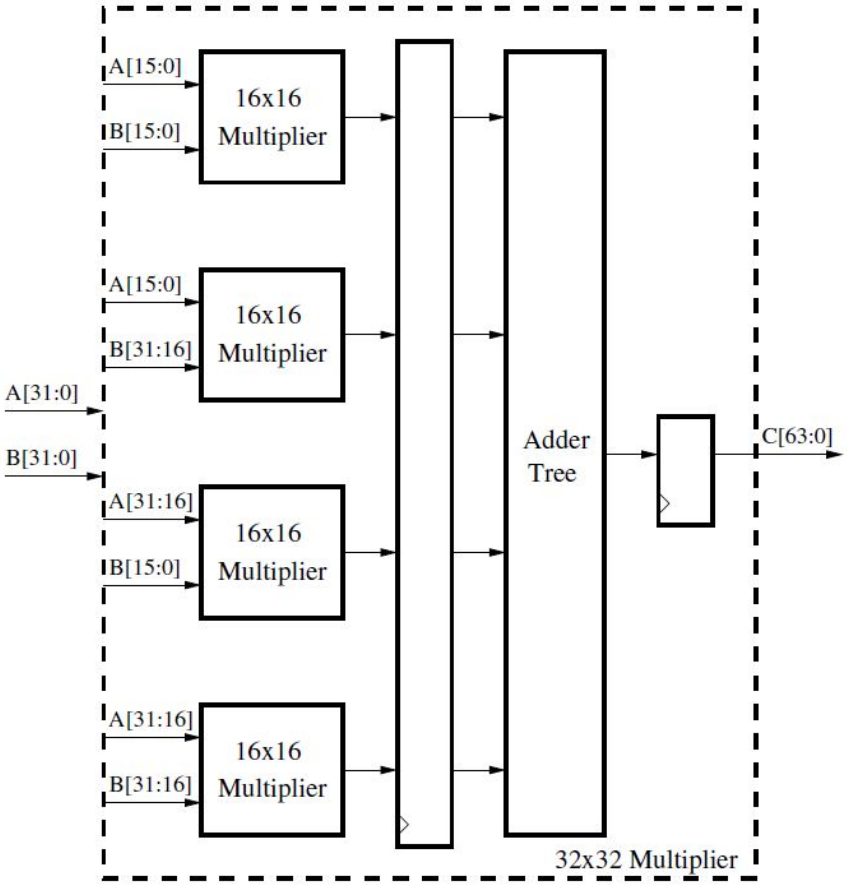


Figure 5.1 : 32-bit Multiplier [8]

Any prime number q suitable for the NTT with this characteristic can be expressed as:

$$q = qH \cdot 2^{log_2 2n} + 1 \tag{5.2}$$

In the employed design, $n = 1024$ and $log_2 2n = 11$, resulting in:

$$q = qH \cdot 2^{11} + 1 \tag{5.3}$$

For Montgomery Reduction operation, they select word size w = 11,

$$mu = -q^{-1} \mod 2^1 1 \equiv -1 \mod 2^{11} \tag{5.4}$$

Utilizing this property, they rewrote Montgomery Reduction as shown in Algorithm 15.

Algorithm 15 Word-Level Montgomery Modular Reduction Algorithm modified for NTT friendly primes [8]

Input: $C = A \cdot B$, *a 2K-bit positive integer,* $44 \leq K \leq 64$
Input: q *a K-bit positive integer,* $q = qH \cdot 2^{11} + 1$
Output: $Res = C \cdot R^{-1} \mod M$ *where* $R = 2^{33} \mod M$

1: $T1 \leftarrow C$
2: **for** *i from 0 to 2* **do**
3: $T1H \leftarrow T1 >> 11$
4: $T1L \leftarrow T1 \mod 2^{11}$
5: $T2 \leftarrow$ *2's complement of T1L*
6: $carry \leftarrow T2[10] ORT1L[10]$
7: $T1 \leftarrow T1H + (qH \cdot T2) + carry$
8: **end for**
9: $T4 \leftarrow T1 - q$
10: **if** $(T4 < 0)$**then**
11: $Res \leftarrow T1$
12: **else**
13: $Res \leftarrow T4$
14: **end if**

To ensure that a single subtraction at the conclusion of Algorithm 15 is sufficient, it is required that $K < (3 * 11)$ be satisfied. If $K < (2 * 11)$ then 2 iterations are necessary instead of 3. The algorithm can be readily adapted to accommodate other values of

n. For instance, with $n = 2048$, $w = 12$, and a modulus length within the range $(4 \times 12) < K < (5 \times 12)$ 5 iterations are needed. Similarly, for $n = 4096$, $w = 13$, and a modulus length within $(4 \times 13) < K < (5 \times 13)$, 5 iterations are required. Hardware design for Algorithm 5 is shown in Fig 5.2.

5.1.3 Number Theoretic Transform Unit

The iterative NTT algorithm presented in Algorithm 1 was modified to achieve optimized performance for NTT computations, with the optimizations being influenced by the adjustments described in [21]. In the source from which the code was taken, some adjustments were made to the NTT kernel and thus it was optimized. You can refer to the source for detailed information. [8]

The resulting coefficients of the polynomial are not in the correct order. To obtain an accurate NTT result, a permutation operation is required. However, given that now we are in the NTT domain and every operand within this domain follows the same scrambled order, it is possible to retain the result of this operation without executing the permutation.

In the case of polynomial multiplication, both polynomials are transformed into the NTT domain, and their inner multiplication is computed. This process results in an outcome that maintains the same scrambled order. Subsequently, after the inner multiplication of operands in NTT form, we employ the inverse NTT operation to return the operand to its original polynomial domain. Through slight adjustments to the inverse NTT operations, we successfully reverse the scrambling of NTT operands without the need for additional permutation operations. This optimization contributes to a reduced latency for NTT operations [8].

5.2 Comparison of Modular Multiplications

In this section, the results obtained from the study will be evaluated. First of all, the 32-bit multiplier structure used in the design is the same as the version implemented with K2RED modular reduction. The RTL I use in the modular multiplication process uses code designed entirely using DSP. The reason for this is that the maximum clock

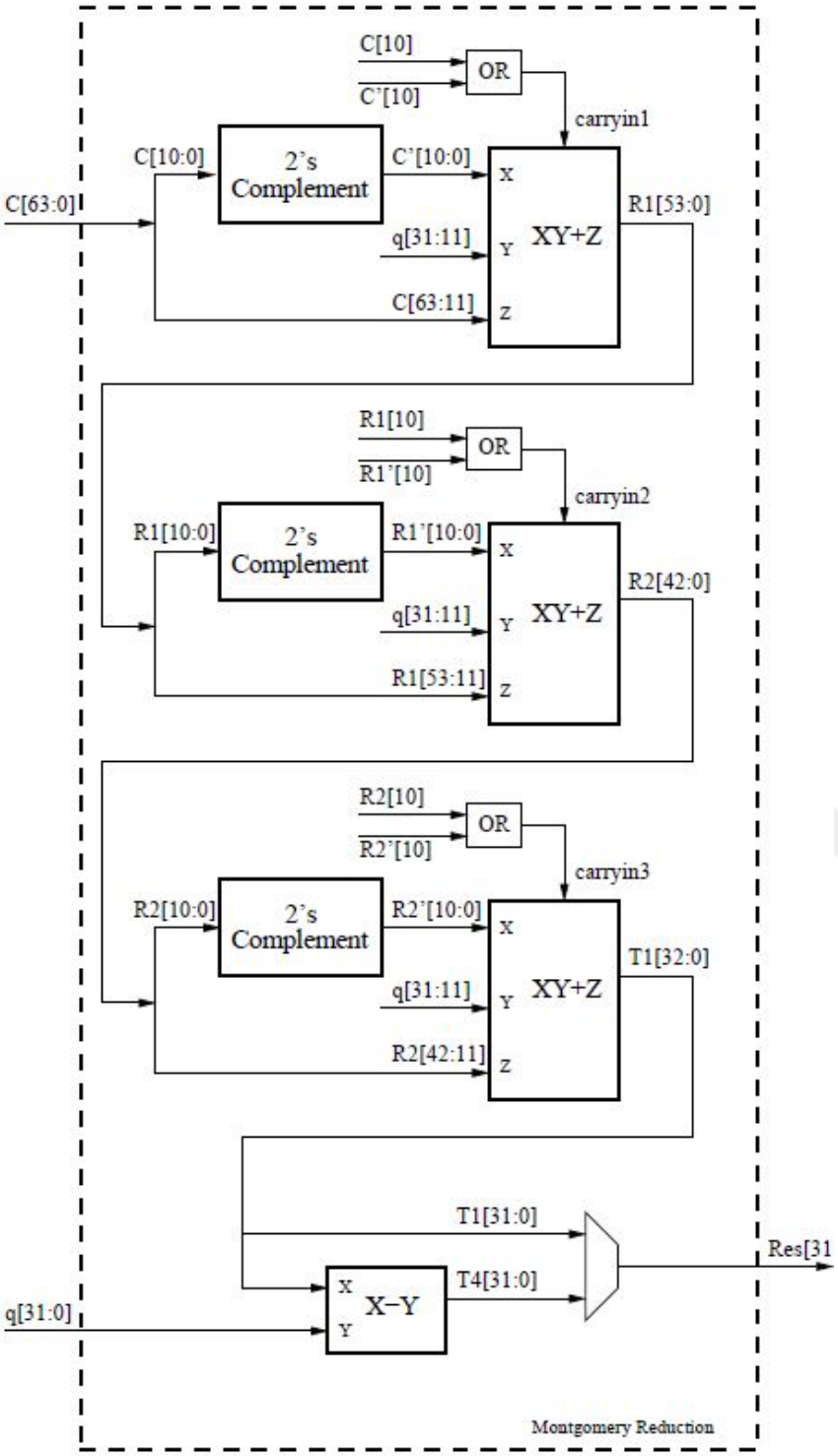


Figure 5.2 : Word-Level Montgomery Reduction algorithm modified for NTT-friendly primes [8]

Table 5.1 : Comparison of 32-bit NTT Implementation K^2RED and Word-Level Montgomery Algorithms.

Algorithm	BU	MCF(MHz)	Area			
			LUT	FF	DSP	BRAM
WLM	1	131.5	1888	1059	12	3
K^2RED	1	131.5	1774	1096	10	3
WLM	2	131.5	3035	1582	24	4
K^2RED	2	126.5	2828	1712	20	4
WLM	4	135.1	4917	2620	48	6
K^2RED	4	128.5	4552	2936	40	6
WLM	8	126.5	10929	4776	96	12
K^2RED	8	121.9	10180	5418	80	12
WLM	16	113.6	21905	9039	192	24
K^2RED	16	113.6	18211	10486	160	24

BU: Butterfly Unit
MCF: Maximum Clock Frequency
LUT: Look Up Table
FF: Flip-Flop
DSP: Digital Signal Processor
BRAM: Block Random Access Memory

frequency is higher and the area consumption and DSP usage are less than some major algorithms.

When Table 5.1 is examined, the columns provide, in order, the algorithm name, the number of butterfly units operating in parallel, the maximum frequency at which the circuit can operate after the place and routing process, the number of used LUTs (Look-Up Tables), FFs (Flip-Flops), DSPs (Digital Signal Processors), and finally, the number of BRAMs (Block RAMs).

First of all, if we talk about the number of butterfly units, this unit is the unit that performs the basic operations in the NTT process. Operations such as nodular addition, subtraction and multiplication are performed here. In order to perform an NTT operation, this block is called many times using a single butterfly unit, or values are given in multiples of 2 of these units in parallel to the circuit. The limiting effect here is the size of the FPGA used.

When circuits are synthesized with different numbers of butterfly units, the first thing that stands out is that as the number of butterfly units increases, the maximum clock

frequency at which the circuit can operate gradually decreases. In K^2RED and WLM algorithms, when the number of butterfly units is 1 and 16, MCF values are the same. In other values, there is a difference of approximately 5-10 MHz. When comparing the circuit in terms of area, when using WLM, the area increases as the number of DSP butterfly units increases. While there was a significant decrease in the number of LUTs and DSPs, there was a certain increase in the number of FFs.

6. CONCLUSIONS AND FUTURE WORKS

This research extensively examines modular multiplication algorithms within the realm of PQC and investigates their practical implementations on FPGAs. The analysis prioritizes time-area trade-offs to guide the selection of these algorithms. Envisioning applications across diverse domains like information, computation, and communication, ongoing studies on algorithm implementations actively address security concerns in the post-quantum era of informatics.

The study specifically focuses on modular reduction algorithms, critical for accelerating NTT and optimizing fields. Looking forward, these designed algorithms show potential for concluding the NTT process, highlighting the necessity of designing an efficient multiplier circuit. Moreover, there is an opportunity for a comprehensive comparison of modular multiplication algorithms, taking into account factors such as energy consumption, resilience against side-channel attacks, and overall performance criteria. Such comparative studies significantly contribute to a nuanced understanding and enhancement of the overall cryptographic system performance.

When compared with studies in the literature, the place and route results of WLM modular reduction algorithm and K^2RED algorithm in different numbers of butterfly units are shown in Table 5.1 in article [8]. As can be seen from Table 5.1, as the number of Butterfly units increases, the difference between the number of DSP usage widens. In NTT operations performed with WLM at butterfly unit numbers 2,4,8, the maximum clock frequency value is higher in the range of 5-10 MHz.

If we look at the contribution of this thesis to the literature: CRYSTALS-Kyber is the first time that the process called modular reduction, together with modular multiplication or multiplication, which is a sub-module of the NTT algorithm used in CRYSTALS-Dilithium and Homomorphic encryption, has been implemented in hardware using DSP blocks. In addition, different circuit designs have been presented and different circuit topologies have been implemented to suit different needs in terms

of area, time and delay. In the last part of the thesis, a comparison was made using the K^2RED algorithm instead of the Word-Level Montgomery algorithm in an NTT IP core used in homomorphic encryption and the results were shared. According to these results, the K^2RED algorithm is more efficient than the Word-Level Montgomery algorithm at the same frequency and in terms of the number of FPGAs it uses per unit process.

As a result of this study, the NTT process can be used more effectively in FPGA hardware by using the K^2RED modular reduction algorithm in the NTT process. As a result, Post Quantum Cryptography algorithms and Homomorphic encryption algorithms can be implemented on faster or less costly hardware. However, with the introduction of Quantum computers, which are rapidly working today in areas such as IoT(Internet of Things), Cloud operations and Secure Communications, into our lives, a significant gap will be closed in the coming years. The first thing to do in the continuation of this study is to choose one of the CRYSTALS-Kyber, CRYSTALS-Dilithium or Homomorphic encryption and implement it using the appropriate algorithm with the results obtained from this thesis. After the implementations are completed and some security tests and verification studies are carried out, it will be a work that can be used on classical computers as well.

REFERENCES

[1] (Last accessed 25 December 2023). *NIST - COMPUTER SECURITY RESOURCE CENTER (CSRC): Post-Quantum Cryptography - Round 3 Submissions*, https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[2] **Üstün A., Can, F. and Örs, B.** (2024). Comparison of FPGA Implementations of Modular Reduction Algorithms for Small Prime Numbers.

[3] (2020). *IBM Newsroom*, https://newsroom.ibm.com/media-quantum-innovation?keywords=quantum&l=100, accessed on December 12, 2023.

[4] **Chaudhary, P.**, **Gupta, R.**, **Singh, A. and Majumder, P.** (2019). Analysis and Comparison of Various Fully Homomorphic Encryption Techniques, *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*, pp.58–62.

[5] **Özerk, O.**, **Elgezen, C.**, **Mert, A.C.**, **Öztürk, E. and Savaş, E.** (2022). Efficient Number Theoretic Transform Implementation on GPU for Homomorphic Encryption, *J. Supercomput.*, *78*(2), 2840–2872, https://doi.org/10.1007/s11227-021-03980-5.

[6] **Abdulrahman, A.**, **Hwang, V.**, **Kannwischer, M.J. and Sprenkels, A.** (2022). *Faster Kyber and Dilithium on the Cortex-M4*, Cryptology ePrint Archive, Paper 2022/112, https://eprint.iacr.org/2022/112, https://eprint.iacr.org/2022/112.

[7] **Mert, A.C.**, **Öztürk, E. and Savaş, E.** (2019). Design and Implementation of a Fast and Scalable NTT-Based Polynomial Multiplier Architecture, *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pp.253–260.

[8] **Mert, A.C.**, **Öztürk, E. and Savaş, E.** (2019). Design and Implementation of a Fast and Scalable NTT-Based Polynomial Multiplier Architecture, *2019 22nd Euromicro Conference on Digital System Design (DSD)*, pp.253–260.

[9] **Nannipieri, P.**, **Di Matteo, S.**, **Zulberti, L.**, **Albicocchi, F.**, **Saponara, S. and Fanucci, L.** (2021). A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms, *IEEE Access*, *9*, 150798–150808.

[10] **Shor, P.W.** (1994). Algorithms for Quantum Computation: Discrete Logarithms and Factoring, *Proceedings of the 35th Annual Symposium on Foundations*

of Computer Science, SFCS '94, IEEE Computer Society, USA, p.124–134, https://doi.org/10.1109/SFCS.1994.365700.

[11] **Grover, L.K.** (1996). *A fast quantum mechanical algorithm for database search*, quant-ph/9605043.

[12] *Submission Requirements and Evaluation Criteria for the Post-quantum Cryptography Standardization Process*, https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[13] **Bos, J.**, **Ducas, L.**, **Kiltz, E.**, **Lepoint, T.**, **Lyubashevsky, V.**, **Schanck, J.**, **Schwabe, P.**, **Seiler, G. and Stehlé, D.** *CRYSTALS -Kyber: a CCA-secure module-lattice-based KEM*, https://eprint.iacr.org/2017/634.pdf.

[14] **Ducas, L.**, **Kiltz, E.**, **Lepoint, T.**, **Lyubashevsky, V.**, **Schwabe, P.**, **Seiler, G. and Stehlé, D.** (2018). CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 238–268.

[15] **Levina, A.B.**, **Kadykov, V.Y. and Kaplun, D.I.** (2021). New direction in Cryptography: Homomorphic Encryption, *2021 International Conference Automatics and Informatics (ICAI)*, pp.234–237.

[16] **Zhang, C.**, **Liu, D.**, **Liu, X.**, **Zou, X.**, **Niu, G.**, **Liu, B. and Jiang, Q.** (2021). Towards Efficient Hardware Implementation of NTT for Kyber on FPGAs, *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.1–5.

[17] **Nejatollahi, H.**, **Dutt, N.**, **Ray, S.**, **Regazzoni, F.**, **Banerjee, I. and Cammarota, R.** (2019). Post-Quantum Lattice-Based Cryptography Implementations: A Survey, *ACM Comput. Surv.*, *51*(6), https://doi.org/10.1145/3292548.

[18] **Liang, Z.** (2022). *Number Theoretic Transform and Its Applications in Lattice-based Cryptosystems: A Survey*, https://arxiv.org/pdf/2211.13546.pdf.

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

[20] **Bisheh Niasar, M.**, **Azarderakhsh, R. and Mozaffari Kermani, M.** (2021). High-Speed NTT-based Polynomial Multiplication Accelerator for Post-Quantum Cryptography, pp.94–101.

[21] **Longa, P. and Naehrig, M.** (2016). Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography, pp.124–139.

[22] **Celik, A.**, **Yilmaz, F.**, **Korkmaz, M.A. and Ors, B.** (2023). Implementation of CRYSTALS-Kyber Post-Quantum Algorithm Using RISC-V Processor, *2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp.1–4.

[23] (2023). `https://www.xilinx.com/products/design-tools/vivado.html`.

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

[25] **Sendrier, N.** (2017). Code-Based Cryptography: State of the Art and Perspectives, *IEEE Security and Privacy*, *15*(4), 44–50.

[26] **Kuang, R. and Barbeau, M.** (2021). Performance Analysis of the Quantum Safe Multivariate Polynomial Public Key Algorithm, *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp.351–358.

[27] **Ananda Mohan, P.V.** (2023). Hash-based Digital Signatures- A tutorial review, *2023 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*, pp.1–8.

[28] **Gentry, C. and Halevi, S.** (2010). Implementing Gentry's Fully-Homomorphic Encryption Scheme., *IACR Cryptology ePrint Archive*, *2010*, 520.

[29] **Gentry, C.** (2009). Fully Homomorphic Encryption Using Ideal Lattices, *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, Association for Computing Machinery, New York, NY, USA, p.169–178, `https://doi.org/10.1145/1536414.1536440`.

[30] **Halevi, S. and Shoup, V.** (2014). Algorithms in HElib.

[31] **Aguilar-Melchor, C.**, **Barrier, J.**, **Guelton, S.**, **Guinet, A.**, **Killijian, M.O. and Lepoint, T.** (2016). NFLlib: NTT-Based Fast Lattice Library, pp.341–356.

[32] **Dai, W. and Sunar, B.** (2016). cuHE: A Homomorphic Encryption Accelerator Library, volume9540, pp.169–186.

[33] **Kumar, A.V.**, **Bhavana, K. and Yamini, C.** (2022). Fully Homomorphic Encryption for Data Security Over Cloud, *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, pp.782–787.

[34] **Regev, O.** (2009). On Lattices, Learning with Errors, Random Linear Codes, and Cryptography, *J. ACM*, *56*(6), `https://doi.org/10.1145/1568318.1568324`.

[35] **Lyubashevsky, V.**, **Peikert, C. and Regev, O.** (2010). On Ideal Lattices and Learning with Errors over Rings, volume 60, pp.1–23.

[36] **Fujisaki, E. and Okamoto, T.** (1999). Secure Integration of Asymmetric and Symmetric Encryption Schemes, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume1666 of *Lecture Notes in Computer Science*, Springer, pp.537–554.

[37] **Hofheinz, D.**, **Hövelmanns, K. and Kiltz, E.** (2017). *A Modular Analysis of the Fujisaki-Okamoto Transformation*, Cryptology ePrint Archive, Paper 2017/604, `https://eprint.iacr.org/2017/604`, `https://eprint.iacr.org/2017/604`.

[38] **Guo, W.**, **Li, S. and Kong, L.** (2022). An Efficient Implementation of KYBER, *IEEE Transactions on Circuits and Systems II: Express Briefs*, *69*(3), 1562–1566.

[39] **Huang, J.**, **Zhang, J.**, **Zhao, H.**, **Liu, Z.**, **Cheung, R.C.C.**, **Çetin Kaya Koç and Chen, D.** (2022). *Improved Plantard Arithmetic for Lattice-based Cryptography*, Cryptology ePrint Archive, Paper 2022/956, `https://eprint.iacr.org/2022/956`, `https://eprint.iacr.org/2022/956`.

[40] **Avanzi, R.**, **Bos, J.**, **Ducas, L.**, **Kiltz, E.**, **Lepoint, T.**, **Lyubashevsky, V.**, **Schanck, J.M.**, **Schwabe, P.**, **Seiler, G. and Stehlé, D.** (2017). CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation, `https://api.semanticscholar.org/CorpusID:198992527`.

[41] **Brakerski, Z.**, **Langlois, A.**, **Peikert, C.**, **Regev, O. and Stehlé, D.** (2013). Classical Hardness of Learning with Errors, *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, Association for Computing Machinery, New York, NY, USA, p.575–584, `https://doi.org/10.1145/2488608.2488680`.

[42] **Ricci, S.**, **Malina, L.**, **Jedlicka, P.**, **Smekal, D.**, **Hajny, J.**, **Cibik, P. and Dobias, P.** (2021). *Implementing CRYSTALS-Dilithium Signature Scheme on FPGAs*, Cryptology ePrint Archive, Paper 2021/108, `https://eprint.iacr.org/2021/108`, `https://eprint.iacr.org/2021/108`.

[43] **Cooley, J.W. and Tukey, J.W.** (1965). An algorithm for the machine calculation of complex Fourier series, *Mathematics of Computation*, *19*(90), 297–297.

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

[45] **Kim, S.**, **Lee, K.**, **Cho, W.**, **Nam, Y.**, **Cheon, J.H. and Rutenbar, R.A.** (2020). Hardware Architecture of a Number Theoretic Transform for a Bootstrappable RNS-based Homomorphic Encryption Scheme, *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp.56–64.

[46] **Heideman, M.**, **Johnson, D. and Burrus, C.** (1984). Gauss and the history of the fast fourier transform, *IEEE ASSP Magazine*, *1*(4), 14–21.

[47] **Fürer, M.** (2007). Faster Integer Multiplication, *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, Association for Computing Machinery, New York, NY, USA, p.57–66, https://doi.org/10.1145/1250790.1250800.

[48] **Chung, C.M.M.**, **Hwang, V.**, **Kannwischer, M.J.**, **Seiler, G.**, **Shih, C.J. and Yang, B.Y.** (2021). NTT Multiplication for NTT-unfriendly Rings: New Speed Records for Saber and NTRU on Cortex-M4 and AVX2, *IACR Transactions on Cryptographic Hardware and Embedded Systems*, *2021*(2), 159–188, https://tches.iacr.org/index.php/TCHES/article/view/8791.

[49] **Longa, P. and Naehrig, M.** (2016). *Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography*, Cryptology ePrint Archive, Paper 2016/504, https://eprint.iacr.org/2016/504, https://eprint.iacr.org/2016/504.

[50] **Montgomery, P.L.** Modular Multiplication Without Trial Division, *44*, https://www.ams.org/journals/mcom/1985-44-170/S0025-5718-1985-0777282-X/S0025-5718-1985-0777282-X.pdf.

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

[52] **Xilinx** (2023). *7 series FPGAs configurable logic block: User guide*, https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB.

[53] **e Shahwar Kundi, D.**, **Zhang, Y.**, **Wang, C.**, **Khalid, A.**, **O'Neill, M. and Liu, W.** (2022). Ultra High-Speed Polynomial Multiplications for Lattice-Based Cryptography on FPGAs, *IEEE Transactions on Emerging Topics in Computing*, *10*(4), 1993–2005, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9693303.

[54] *Ahmet Can Mert Github Repo*, https://github.com/acmert/parametric-ntt, accessed: 2023-12-25.

[55] (2023). https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html#documentation, accessed: 2023-12-29.

CURRICULUM VITAE

Furkan CAN

EDUCATION:

- **B.Sc.:** 2021, Istanbul Technical University, Electric Electronics Faculty, Electronics and Communication Engineering Department
- **M.Sc.:** 2024, Istanbul Technical University, Electric Electronics Faculty, Electronics Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2020-2021 TÜBİTAK BİLGEM UEKAE Student Researcher.
- 2021 - Now TÜBİTAK BİLGEM BTE Digital Design Engineer

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Can F.**, Üstün A., Örs B., Alaybeyoğlu, E. , Savaş E. (2024). Hardware Implementation of K^2RED and Plantard Modular Multiplication Algorithms in Post-Quantum Cryptography. *Panhellenic Conference on Electronics and Communications 2024*, March 28-29, 2024 Thessaloniki, Greece. (Presentation Instance)