3-10-2021

FPGA Implementation of Post-Quantum Cryptography Recommended by NIST

Xi Gao
University of Windsor

FPGA Implementation of Post-Quantum Cryptography Recommended by NIST

by

Xi Gao

A Thesis

Submitted to the Faculty of Graduate Studies

through Electrical and Computer Engineering

in Partial Fulfilment of the Requirements for

the Degree of Master of Applied Science at the

University of Windsor

Windsor, Ontario, Canada

2021

FPGA Implementation of Post-Quantum Cryptography Recommended by NIST

by

Xi Gao

APPROVED BY:

X. Nie

Department of Mechanical, Automotive, Materials Engineering

A. Emadi

Department of Electrical and Computer Engineering

H. Wu, Advisor

Department of Electrical and Computer Engineering

January, 05, 2021

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In the next 10 to 50 years, the quantum computer is expected to be available and quantum computing has the potential to defeat RSA (Rivest-Shamir-Adleman Cryptosystem) and ECC (Elliptic Curve Cryptosystem). Therefore there is an urgent need to do research on post-quantum cryptography and its implementation.

In this thesis, four new Truncated Polynomial Multipliers (TPM), namely, TPM-I, TPM-II, TPM-III, and TPM-IV for NTRU Prime system are proposed. To the best of our knowledge, this is the first time to focus on time-efficient hardware architectures and implementation of NTRU Prime with FPGA.

TPM-I uses a modified linear feedback shift register (LFSR) based architecture for NTRU prime system. TPM-II makes use of x^2-net structure for NTRU Prime system, which scans two consecutive coefficients in the control input polynomial $r(x)$ in one clock cycle. In TPM-III and TPM-IV, three consecutive zeros and consecutive zeros in the control input polynomial $r(x)$ are scanned during one clock cycle, respectively.

FPGA implementation results are obtained for the four proposed polynomial multiplication architectures and a comparison between the proposed multiplier FPGA results for NTRU Prime system and the existing work on NTRUEncrypt is shown.

Regarding space complexity, TPM-I can reduce the area consumption with the least logical elements, although it takes more latency time among the four proposed multipliers and NTRUEncrypt work [12]. TPM-II has the best performance of latency with parameter sets $ees401ep1$, $ees449ep1$, $ees677ep1$ in security levels: 112-bit, 128-bit, and 192-bit, respectively. TPM-IV uses the smallest latency time with the parameter set $ees1087ep2$ in security level 256, compared to the other three latency time of proposed multipliers. Both TPM-II and TPM-IV have a lower latency time compared to NTRUEncrypt work [12] in different security levels. Note that NTRU Prime has enhanced security in comparison with NTRUEncrypt due to the fact, the former uses a new truncated polynomial ring, which has a more secure structure.

DEDICATION

To my loving parents:

Father: Lingbo Gao

Mother: Liping Yu

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to everyone who helped me make this thesis possible. First of all, I am deeply indebted to my supervisor Dr. Huapeng Wu, who gave me a big help during my study for the master degree. As one of best teachers I have ever had, Professor Wu guided me throughout the writing of this thesis. His broad knowledge and logical thinking have been of great value. Without his detailed and constructive comments on my research, none of this thesis would be possible. In addition, I would like to appreciate my loving parents. Without their support and encouragement, it is impossible for me to achieve such accomplishment. I also feel grateful to my friends, Pintian Huang, Ting Huang, for their support and help to my research and thesis. Finally, I wish to extend my gratitude to everyone at UWindsor's Faculty of Electrical and Computer Engineering for their efforts during my study in the M.A.Sc. Program. Moreover, I gratefully thank for the financial support from University of Windsor and my supervisor Dr. Huapeng Wu.

Xi Gao

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ACRONYMS

ALM Adaptive logic module

ECC Elliptic Curve Cryptosystem

RLWR Ring Learning with Rounding

IND-CCA The notion of semantic security against adaptive chosen-ciphertext attacks

En-time Encryption time

De-time Decryption time

CRC Cyclic Redundancy Check

FMax Maximum operating frequency

TPM Truncated Polynomial Multiplier

AU Arithmetic Unit

LFSR Linear Feedback Shift Register

LUT Look-up table

LWE Learning with errors

NIST National Institute of Standards and Technology

NTRU Nth Degree Truncated Polynomial Ring Unit

1 INTRODUCTION

1.1 Motivation

Internet brings great convenience and wonderful services to people, like social media, online shopping, and online banking. At any given moment, millions of internet users are sharing their personal information and confidential data via the internet. Therefore, network security should never be neglected.

To safeguard confidential data stored in a computer or transmitted over the network from being modified by hackers with malicious purpose, certain cybersecurity measures must be adopted and implemented. Among a wide range of technologies to provide network security, cryptography is probably the most effective and popular one to ensure data confidentiality and integrity.

Modern cryptography technology can be divided into two types: symmetrical-key cryptosystem and asymmetrical-key cryptosystem. Smmetrical-key cryptosystem requires to encrypt and decrypt messages with the same shared keys. However, asymmetrical-key cryptosystem (also called public-key cryptosystem) requires two separate keys: a public key to encrypt messages, which is known to everyone, and a private key to decrypt messages only known to the receiver. RSA and elliptic curve cryptosystem (ECC) are currently two frequently used public-key cryptosystems.

A quantum computer is a machine that can execute mathematical operations, store and process information at a much faster speed, which will transcend all electronics based computers. It is shown that the security of RSA and ECC can be compromised by Shor's algorithm running on a quantum computer [4]. As quantum computers are expected to be practically available in the next 10 to 50 years, it is necessary for us to study cryptographic technologies that will still be secure under the attacks launched from quantum computers. This area of research is called post-quantum cryptography [5].

Currently, post-quantum cryptography research focuses on several approaches and there are code-based cryptography, hash-based cryptography, multivariate cryptography, and lattice-based cryptography. One of the most popular research areas of post-quantum cryptography is lattice-based cryptography, which relies on the worst-case hardness of lattice problems. One popular example of lattice-based systems is NTRU, which is Nth Degree Truncated Polynomial Ring Unit. It is probably the most practical and widely researched one among all the existing post-quantum cryptosystems [1].

NTRU includes two algorithms: NTRUEncrypt, which is used for encryption, and NTRUSign, which is used for digital signature. NTRUEncrypt is the encryption scheme of NTRU system, which was firstly developed by Hoffstein, Pipher, and Silverman from Brown University in 1996 [6]. NTRU has been adopted by IEEE P1363.1 standards under the specifications for lattice-based public-key cryptography since 2009 [1]. The security of NTRU encryption has been reasonably-well understood and scrutinized for decades. In the area of post-quantum cryptography, NTRU and its efficient implementations have become a well-known research topic.

To facilitate the development of new quantum-safe and practical schemes, the National Institute of Standards and Technology (NIST) has initiated a contest for a standard with the aim to replace current schemes that are vulnerable to attack by quantum computers. After the first 2 rounds of the contest, two NTRU based schemes were among the finalists, which includes NTRUEncrypt system and NTRU prime system [21]. NTRU Prime system was first proposed by D. J. Bernstein in 2017, which was similar to NTRUEncrypt system [5]. It is a variant and security-enhanced version of the NTRUEncrypt cryptosystem and comes with a couple of tweaks to minimize the attack surface.

There are two public-key cryptosystems included in NTRU Prime, one is streamlined NTRU Prime, the other is NTRU LPRime. Streamlined NTRU Prime is a

lattice-based cryptosystem and considered to be resistant to quantum attacks. NTRU LPRime can be considered as Ring Learning with Rounding (RLWR) which has only one secret random variable generated in key generation and encapsulation steps. Both of them are designed for the IND-CCA2 (cipher chosen attack) security standard, which is regarded as the strongest as well as the best security level against a range of real-world attacks on the internet [7].

1.2 A Summary of Contributions

In this thesis, several time-efficient multiplication architectures for Streamlined NTRU Prime system are proposed, aiming to reduce latency time and speed up the system. Their FPGA simulations are performed and its FPGA implementation results are given. Note that efficiency is based on time-domain and best works are related to the latency comparisons between four proposed works. To the best of our knowledge, it is the first time that NTRU Prime system has been implemented in FPGA.

The main contribution includes four time-efficient multiplication architectures for NTRU Prime and their FPGA implementation results. The four proposed multipliers are named TPM-I, TPM-II, TPM-III, and TPM-IV. The details can be summarized as follows:

• TPM-I is a linear feedback shift register (LFSR) based architecture for NTRU prime system. TPM-II uses a high-speed arithmetic unit to perform modular polynomial multiplication required in NTRU prime system. TPM-III takes advantage of three consecutive zeros in polynomial coefficients from the input polynomial $r(x)$. TPM-IV takes advantage of consecutive zeros in polynomial coefficients by recoding the polynomial input $r(x)$. Compared to the four proposed architectures with each other, it can be seen that TPM-II and TPM-IV have the best performance of latency. While TPM-I uses the least area resources with the fewer logical modules, which includes the number of ALMs and registers, at the expense of longer latency.

• When compared to the existing work on FPGA implementations of NTRUEncrypt systems, our proposed FPGA implementations have certain advantages on latency over the best existing latency work on NTRUEncrypt. For example, the proposed TPM-II has latency by 1.19 %, 5.4% and 3.5%, when compared to the existing FPGA implementation results on NTRUEncrypt, which has the lowest latency time with parameter sets $ees401ep1$, $ees449ep1$ and $ees677ep1$, respectively [12]. Note that the parameter sets $ees401ep1$, $ees449ep1$ and $ees677ep1$ are corresponding to security level 112, 128 and 192, respectively [1]. The proposed TPM-IV has the smallest latency compared to the existing result on NTRUEncrypt system with the parameter set $ees1087ep2$ (in security level 256). In fact, for $ees1087ep2$, TPM-IV takes advantage of the smaller latency time than NTRUEncrypt work [12] by 21.4%. In addition, TPM-I has a smaller number of ALMs and registers but higher latency than all existing NTRUEncrypt works,

Note that more comparisons of FPGA results of proposed NTRU Prime multipliers with existing works on NTRUEncrypt are provided in Appendix A.

1.3 Thesis Organization

An organization of the rest of the thesis is shown as follows. In Chapter 2, mathematical background and review fundamentals of NTRU Prime system are introduced. The mathematical basics, such like truncated polynomial ring and modular arithmetic of the ring are covered. In addition, recommended parameter sets of NTRU Prime scheme are explained. In Chapter 3, a brief overview of existing works is given. Chapter 4 presents one new hardware architecture based on LFSR for NTRU prime with FPGA implementation results. Chapter 5 shows three new multiplier architectures and compares their FPGA results with similar existing works. In Chapter 6, conclusions are given and possible future works are discussed.

2 MATHEMATICAL PRIMITIVES

In this chapter, the mathematical background of NTRU based system is introduced firstly, then the definition of truncated polynomial ring and NTRU system are shown, which includes the key generation, encryption, and decryption algorithms. NTRU prime system, as well as the parameter selection for NTRU prime cryptosystem, are covered. Then it shows the mathematical background of NTRU prime cryptosystem, which includes key generation, encryption, decryption and parameter selection.

2.1 Group

Group and ring are two important abstract algebraic concepts to understand the cryptosystem, like NTRU and NTRU prime. Firstly, a brief introduction to group will be given.

In mathematics, a group is a set equipped with a binary operation that combines any two elements to form a third element in such a way that four conditions called group axioms are satisfied, namely closure, associativity, identity, and invertibility. The concept of a group arose from the study of polynomial equations, starting with Evariste Galois in the 1830s, who introduced the term of group for the symmetry group of the roots of an equation, now called a Galois group [16].

Defination 2.1.1 *A group is a set G together with a binary operation $*$ on G such that:*

1. Binary operator is closure, for all $a, b \in G$,

$$a * b \in G.$$

2. Binary operator is associative, for all $a, b, c \in G$,

$$(a * b) * c = a * (b * c).$$

3. There is an identity element, for all $a \in G$,

$$e * a = a * e = a.$$

4. For all $a \in G$, there exists an inverse element $a^{-1} \in G$ such that,

$$a * a^{-1} = a^{-1} * a = e.$$

A group is called abelian or commutative if it satisfies for all $a, b \in G, a * b = b * a$, which are named after early 19th-century mathematician Niels Henrik Abel.

In mathematics, a semigroup is an algebraic structure consisting of a set together with a binary operation, which only satisfies conditions closure and associative, but not identity or inverse. Note that a group must be a semigroup, but a semigroup is not necessarily a group [16].

2.2 Ring

In mathematics, a ring is one of the fundamental algebraic structures used in abstract algebra. Two binary operations are defined in a ring, which is named as addition and multiplication. Through this generalization, theorems from arithmetic are extended to non-numerical objects, like polynomials, series, matrices, and functions. The conceptualization of rings began in the 1870s and was completed in the 1920s. Key contributors include Dedekind, Hilbert, Fraenkel, and Noether [17].

Defination 2.2.1 *A ring is a set R together with two binary operations $+$ and $\cdot$ that satisfy the following properties:*

Defination 2.2.2 *R is an abelian group under addition $+$, which satisfies the following properities:*

1. Operation $+$ is associative, for all $a, b, c \in R$,

$$(a+b)+c=a+(b+c);$$

2. Operation $+$ is commutative, for all $a, b \in R$,

$$a+b=b+a;$$

3. There is an identity element $0 \in R$, for all $a \in R$,

$$0+a=a+0=a.$$

Defination 2.2.3 *R is a monoid under multiplication $\cdot$, meaning that:*

1. Operation $\cdot$ is associative, for all $a, b, c \in R$,

$$(a \cdot b) \cdot c=a \cdot (b \cdot c);$$

2. There is a multiplicative identity element $1 \in R$, for all $a \in R$,

$$1 \cdot a=a \cdot 1=a.$$

Defination 2.2.4 *Multiplication $\cdot$ is distributive with respect to addition $+$, which satisfies the following properities:*

1. Operation is left distributivity, for all $a, b, c \in R$,

$$a \cdot (b+c)=a \cdot b+a \cdot c.$$

2. Operation is left distributivity, for all $a, b, c \in R$,

$$(b+c) \cdot a=b \cdot a+c \cdot a$$

2.3 Truncated Polynomial Ring

The orginal NTRU is a lattice-based cryptosystem, which is called Nth Degree Truncated Polynomial Ring Unit and it uses a ring denoted by $R = Z_q[x]/(x^n - 1)$, where n is a prime number [3]. $Z_q[x]$ is the set of polynomials modulo $x^n - 1$ with integer coefficients and taken modulo q. Arithmetic operations on $R = Z_q[x]/(x^n - 1)$ includes addition and multiplication, which are discussed in the following two subsections.

2.3.1 Addition

Let $a(x)$, $b(x) \in R$ be given as

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 \tag{1}$$

$$b(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_0 \tag{2}$$

Note that $a_i \in \{0, 1, \ldots, q-1\}$ and $b_i \in \{0, 1, \ldots, q-1\}$.

Addition operation $a(x)$ and $b(x)$ can be performed by simply adding their coefficients.

$$a(x) + b(x) \triangleq c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \tag{3}$$

where

$$c_i = a_i + b_i \ mod \ q, i = 0, 1, \cdots, n-1 \tag{4}$$

2.3.2 Multiplication

Let $a(x)$, $b(x) \in R$ be given as in (1), (2):

$$a(x) \cdot b(x) \triangleq d(x) = d_{n-1}x^{n-1} + d_{n-2}x^{n-2} + \cdots + d_0 \tag{5}$$

where

$$d_k = \sum_{\substack{i,j=0,1,\ldots,n-1 \\ i+j \ mod \ n=k}} a_i b_j, k = 0, 1, \ldots, n-1 \tag{6}$$

The operations in NTRU based system works in the truncated polynomial field R modulo q, which means that all polynomial coefficients should be from integer set $\{0, 1, \ldots, q-1\}$. Every coefficient in the polynomial should modulo q, which is denoted by $R_q = Z_q[x]/(x^n - 1)$.

2.3.3 An example of truncated polynomial ring arithmetic

Let $n = 3$, $q = 4$ in $R_q = Z_q[x]/(x^n - 1)$ and let $a(x), b(x) \in R_q$, be given as follows:

$$a(x) = x^2 + 3 \tag{7}$$

$$b(x) = x^2 - 3x \tag{8}$$

Then, addition and multiplication operations between $a(x)$ and $b(x)$ can be shown as:

$$\begin{aligned} a(x) + b(x) &= x^2 + 3 + (x^2 - 3x) \ mod \ q \\ &= 2x^2 - 3x + 3 \ mod \ 4 \\ &= 2x^2 + x + 3 \end{aligned} \tag{9}$$

$$\begin{aligned} a(x) \times b(x) &= a(x) \times b(x) = (x^2 + 3) \times (x^2 - 3x) \ mod \ q \ mod \ x^n - 1 \\ &= x^4 - 3x^3 + 3x^2 - 9x \ mod \ 4 \ mod \ x^3 - 1 \\ &= 3x^2 + 1 \end{aligned} \tag{10}$$

2.4 NTRUEncrypt Cryptosystem

An overview of NTRUEncrypt cryptosystem is given in this section.The cryptosystem includes three parts: parameter selection, key generation, encryption and decryption operation.

2.4.1 Parameter Selection

Primarily, the NTRUEncrypt system is made up of three integer parameters n, p and q. n is a prime number to define the degree of the truncated polynomials in R. p and q must be relatively prime where $q >> p$. In addition, three integer parameters d_f, d_g and d_r are selected to determine three sets L_f, L_g and L_r, which are $n-1$ degree [3]. In order to describe all three polynomial sets, we uses $L(d_1, d_2)$, which means the polynomial in $L(d_1, d_2)$ has d_1 coefficients equal to 1, d_2 coefficients equal to -1 and the rest coefficients equal to 0. To go a step further, we get

$$L_f = L(d_f, d_f - 1), L_g = L(d_g, d_g), L_r = L(d_r, d_r) \tag{11}$$

A parameter generation algorithm for NTRU system was proposed in [3], which is shown in Algorithm 2.1. It uses security level k as the input, which represents security strength, and then outputs a parameter set of n and d_f. There are two functions that are used in the algorithm. One is $hybridSecurityEstimate\ (n, d_f)$, in order to estimate the minimal security over all attack strategies on the parameter sets, another one is used to estimate the chance of decryption failure on the parameter sets, which called $decryptionFailureProb\ (n, d_f)$.

Algorithm 2.1 Parameter Generation Function for NTRUEncrypt [3]

Input: Security Level, k

Output: Parameter Set (n, d_f)

$i \leftarrow 1$ {The variable i is used to index the set of acceptable primes $\mathcal{P}$}
$i* \leftarrow 0$ {This will become the first index which can achieve the required security}
repeat
$n \leftarrow P_i$
$d_f \leftarrow [n/3]$ {We will try each d_f from $[n/3]$ down to 1}
repeat
$k_1 \leftarrow hybridSecurityEstimate\ (n,\ d_f)$
$k_2 \leftarrow log_2\ (decryptionFailureProb\ (n,\ d_f))$
if ($k_1 \geq k$ and $k_2 < -k$) **then**
    $(i^*,\ d_f^*) \leftarrow (i,\ d_f)$ {Record the first acceptable index i and the value of d_f}
end if
$d_f \leftarrow d_f - 1$
until $i^* > \mathbf{0}$ **or** $d_f < \mathbf{1}$
$i \leftarrow i + 1$
until $i^* > \mathbf{0}$
$c^* \leftarrow cost(P_{i^*},\ d_f)$
while an increase in N can potentially lower the cost do **do**
    $n \leftarrow \mathcal{P}_\rangle$
    $d_f \leftarrow d_f^*$ {Note that when n increases the cost must be worse for all $d_f \geq d_f^*$, and that the decryption failure probability is decreased both by an increase in n and a decrease in d_f }
    repeat
    $k_1 \leftarrow \text{hybridSecurityEstimate}(n, d_f)$
    $c \leftarrow \text{cost}(n,\ d_f)$
    if ($k_1 \geq k$ and $c < c^*$) **then**
        $(c^*, i^*, d_f) \leftarrow (c, i, d_f)$ {Record the cost improvement, corresponding i, d_f }
    end if
    $d_f \leftarrow d_f - 1$
    until $d_f \leftarrow d_f < 0$
    $i \leftarrow i + 1$
    return $(P_{i^*},\ d_f^*)$
end while

Table 2.1 shows recommended parameter sets for NTRUEncrypt system, which is provided by [4]. It can be seen that there are four security levels and each of them has three parameter sets. The parameter q is set to 2048, which is in the form of 2^n aims to achieve a higher security level. Because two parameters p and q are relatively prime and $p << q$, 3 is selected to be the smallest odd prime number as p. Note that d_f and d_r have the same value. Also, n is the parameter that can define the security level of this system [3]. Then, the recommended parameter sets from Table 2.1 will be utilized in the four proposed works' computation.

Table 2.1: Recommended Parameters for NTRUEncrypt [1]

Security Level	Parameter sets	n	p	q	d_f	d_g	d_r
112	$ees401ep1$	401	3	2048	113	133	113
	$ees541ep1$	541	3	2048	49	180	49
	$ees659ep1$	659	3	2048	38	219	38
128	$ees449ep1$	449	3	2048	134	149	134
	$ees613ep1$	613	3	2048	55	204	55
	$ees761ep1$	761	3	2048	42	253	42
192	$ees677ep1$	677	3	2048	157	225	157
	$ees887ep1$	887	3	2048	81	259	81
	$ees1087ep1$	1087	3	2048	63	362	63
256	$ees1087ep2$	1087	3	2048	120	367	120
	$ees1171ep1$	1171	3	2048	106	390	106
	$ees1499ep1$	1499	3	2048	79	499	79

2.4.2 Key Generation

The key generation process is given below:

Step 1: Randomly choose a polynomial $f(x)$ from the polynomial set L_f, which is invertible in R_q and R_p.

Step 2: Randomly choose a polynomial $g(x)$ from the polynomial set L_g.

Step 3: Calculate f_q and f_p which are the inverse of polynomial $f(x)$ mod q and $f(x)$ mod p.

Step 4: Compute $h(x) = p \cdot f_q(x) \times g(x) \bmod q$.

Then the public key $h(x)$ and the corresponding private key pair ($f(x)$, $f_p(x)$) of the system can be obtained.

2.4.3 Encryption

Step 1: Encode the message m into a polynomial $m(x)$ with coefficients from $\{-1, 0, 1\}$.

Step 2: Randomly choose a polynomials $r(x)$ from the polynomial set L_r.

Step 3: Encrypt message by performing $e(x) = h(x) \times r(x) + m(x) \bmod q \bmod x^n - 1$.

Note that $e(x)$ is the ciphertext sent to the receiver.

2.4.4 Decryption

After receiving the encrypted message $e(x)$ from the sender.

Step 1: Compute $a(x) = f(x) \times e(x) \bmod q$.

Step 2: Shift coefficients of $a(x)$ from $[0, q-1]$to the range $[-\frac{q}{2}, \frac{q}{2}]$.

Step 3: Compute $m(x) = f_p(x) \times a(x) \bmod p$

Note that $m(x)$ is the plaintext sent from the sender.

2.5 Streamlined NTRU prime system

Streamlined NTRU prime system is defined over a special algebraic structure, which is denoted by $R = Z_q[x]/(x^n - x - 1)$, where n and q are prime numbers. In this notation, $Z_q[x]$ denotes the set of polynomials modulo $x^n - x - 1$ with integer coefficients and taken modulo q. It can be seen that the set of R contains all the polynomials with integer coefficients of degree up to $n-1$. Arithmetic operations on $R_q = Z_q[x]/(x^n - x - 1)$ includes addition and multiplication, which are discussed in the following two subsections.

2.5.1 Addition

Let $a(x)$, $b(x) \in R$ be given as

$$a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 \tag{12}$$

$$b(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_0 \tag{13}$$

Note that $a_i \in \{0, 1, \ldots, q-1\}$ and $b_i \in \{0, 1, \ldots, q-1\}$.

Addition operation $a(x)$ and $b(x)$ can be performed by simply adding their coefficients.

$$a(x) + b(x) \triangleq c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \tag{14}$$

where

$$c_i = a_i + b_i \ mod \ q, i = 0, 1, \cdots, n-1 \tag{15}$$

2.5.2 Multiplication

Let $a(x)$, $b(x) \in R$ be given as in (12), (13):

$$a(x) \cdot b(x) \triangleq d(x) = d_{n-1}x^{n-1} + d_{n-2}x^{n-2} + \cdots + d_0 \tag{16}$$

where

$$d_k = \sum_{\substack{i,j=0,1,\ldots,n-1 \\ i+j \ mod \ n=k}} a_i b_j, k = 0, 1, \ldots, n-1 \tag{17}$$

The operations in NTRU based system works in the truncated polynomial field R modulo q, which means that all polynomial coefficients should be from integer set $\{0, 1, \ldots, q-1\}$. Every coefficient in the polynomial should modulo q, which is denoted by $R_q = Z[x]/(x^n - x - 1)$.

2.5.3 An example of truncated polynomial ring arithmetic

Let $n = 3$, $q = 5$ in $R_q = Z_q[x]/(x^n - x - 1)$ and let $a(x), b(x) \in R_q$, be given as follows:

$$a(x) = 3x^2 + 2 \tag{18}$$

$$b(x) = x^2 - x \tag{19}$$

Then, addition and multiplication operations between $a(x)$ and $b(x)$ can be shown as:

$$\begin{aligned} a(x) + b(x) &= 3x^2 + 2 + (x^2 - x) \ mod \ q \\ &= 4x^2 - x + 2 \ mod \ 5 \\ &= 4x^2 + 4x + 2 \end{aligned} \tag{20}$$

$$\begin{aligned} a(x) \times b(x) &= a(x) \times b(x) = (3x^2 + 2) \times (x^2 - x) \ mod \ q \ mod \ x^n - x - 1 \\ &= 3x^4 - 3x^3 + 2x^2 - 2x \ mod \ 5 \ mod \ x^3 - x - 1 \\ &= 3x + 2 \end{aligned} \tag{21}$$

In addition, Streamlined NTRU prime cryptosystem includes totally four parts: parameter selection, key generation, encryption, and decryption. Each part is discussed in this section in detail.

2.5.4 Parameter Selection

Basically, Streamlined NTRU prime encryption is a parameterized cryptosystem where the truncated polynomial ring R is determined by three integer parameters

n, q and t. n is a prime number which defines the degree of truncated polynomials in R. Furthermore, q is a prime number and t is a positive integer, there are some ranges between these parameters $t \geq 1$, $n \geq 3t$, $q \geq 32t + 1$.

Table 2.2 shows a subset of recommended parameter sets for streamlined NTRU prime encryption system, which can get from [8]. The multiple parameter sets with three security levels for streamlined NTRU prime encryption systems are considered to be secure to resist attacks. Note that q is a prime number to allow highly efficient computation of the modular operation.

Table 2.2: Recommended Parameters for NTRU Prime [8]

Security Level	Parameter sets	n	q	t
112	$ees541ep1$	541	2297	71
	$ees659ep1$	659	2137	66
128	$ees613ep1$	613	1459	45
	$ees761ep1$	761	1619	50
192	$ees677ep1$	677	3251	101
	$ees887ep1$	887	13007	295

2.5.5 Key generation

The steps of key generation are shown as follows:

Step 1: Generate a uniform randon small element $g(x) = g_{p-1}x^{p-1} + \cdots + g_0$ with $g_i \in \{-1,0,1\}$, $g \in \frac{R}{3}$.

Step 2: Generate a uniform random t-small element $f(x)$.

Step 3: Compute public key $h(x) = \frac{g}{3f}$ in $\frac{R}{q}$. (By assumption q is a prime $>> 3$, so 3 is invertible in $\frac{R}{q}$.)

Step 4: Private key is $f(x)$ in R and $\frac{1}{g} \in \frac{R}{3}$.

Note that each element of Z_q is traditionally encoded as $[log_2 q]$ bits.

2.5.6 Encryption

Step 1: Decode the public key $h(x)$ and obtain $h(x) \in R_q$.

Step 2: Pick a random t-small polynomial $r(x) \in R_q$ and hash r to obtain the session key K.

Step 3: Obtain $e(x) \in R$ by rounding each coefficient of $(h(x) \times r(x)$ in $R_q)$ to the nearest multiple of 3, and set $e(x)$ as the ciphertext.

2.5.7 Decryption

Step 1: Obtain $a(x) = 3 \times f(x) \times e(x) \bmod q$.

Step 2: Hash $a(x)$ to obtain the session key K.

3 AN OVERVIEW OF EXISTING WORKS

Since NTRUEncrypt cryptosystem was first proposed by J. Hoffstein et al. in 1996 [6], it has become one of the most important and effective researched areas in the past twenty years compared with other popular public-key cryptosystems, like RSA and ECC. Streamlined NTRU prime cryptosystem was firstly invented in 2017 by Daniel J. Bernstein et al. [8], which has not been a popularly researched area in the past twenty years. Also, Streamlined NTRU prime is NTRU-based cryptosystem and there is a limited amount of literature published to provide both software and hardware implementations. In this chapter, we will review some similar existing works on efficient implementation for both NTRU system and NTRU prime system. In order to speed up time-efficient and area resources, several optimizations have been made on NTRU cryptosystem in [9] [12] [13]. Meanwhile, the software implementation is shown in [15].

The earliest published hardware implementation of NTRU system was in 2001 from D. Bailey and five others [9]. In this work, NTRUEncrypt system is implemented in software running on several different constrained devices, which contains which are C language, Palm Computing Platforms, Advanced RISC Machines ARM7TDMI, et al. and in hardware, FPGA, which applied on Xilinx's Virtex 1000EFG860 FPGA. The system used the parameter set for NTRUEncrypt, which was $(n,\ p,\ q) = (251, X + 2, 128)$. Two binary polynomials $r(x)$ and $m(x)$ can simplify NTRUEncrypt's encryption procedures in hardware. In addition, the efficiency of the system was improved due to its parallel architecture with hardware implementation. While the steps in decryption are difficult to implement in hardware and the system works only for light-weight security. Note that Table 3.1, 3.2, and 3.3 show implementation of FPGA results, C language, and Palm Assembly Language respectively.

Table 3.1: FPGA Implementation Results [9]

Encryption Cycles	259
Clock Period	19.975 ns
Clock Frequency	50.063 MHz
Encryption Time	5.174 μs
Encryption Throughput	48.52 Mbps
Slices Used	6373
Logic Resource Utilization	51 %
Approximate Gate Count	60,000
Approximate Register Gate Count	40,000
I/O Used	506
I/O Utilization	77 %

Table 3.2: NTRU Based on C Performance Results [9]

Operation	MC68EX328	Intel 80386	37MHz ARM7
Key Generation	1130 $msec$	858 $msec$	80.6 $msec$
Encryption	47 $msec$	39 $msec$	3.25 $msec$
Decryption	89 $msec$	72 $msec$	6.75 $msec$

Table 3.3: NTRU Palm Assembly Language Performance Improvements [9]

Operation	Palm C	Palm Assembly/C
Key Generation	1130 $msec$	630 $msec$
Encryption	47 $msec$	33 $msec$
Decryption	89 $msec$	60 $msec$

B. Liu proposed an efficient hardware architecture and FPGA implementation for NTRUEncrypt based on truncated polynomial ring unit [13]. In [13], the architecture uses LFSR structure due to its compact circuitry and high speed. Then a new architecture based on extended LFSR has been presented. During one clock cycle, the number of clock cycles can be reduced if two consecutive zero coefficients in the input polynomial $r(x)$ can be processed. In order to optimize the system, a new modular arithmetic unit is designed, which used the redundant state from polynomial $r(x)$. Then, an LFSR and extended LFSR based NTRUEncrypt structure can be obtained. After that, the FPGA implementation results based on LFSR shows the architecture

has the best performance in term of area-latency product compared with existing works. The FPGA implementation results based on extended LFSR shows that the product of area and latency of this architecture is the lowest compared with LFSR based architectures. But the design based on extended LFSR uses a slightly larger area resources compared with the implementations based on LFSR.

R. Dong proposed four new truncated multiplier architectures and used FPGA to simulate [12]. All new multiplication architectures are based on LFSR structure. Firstly, Multiplier I use x^2-net structure based on LFSR, which scans two consecutive coefficients in control input polynomial $r(x)$ during one clock cycle. In Multiplier II, three consecutive zeros in the control input polynomial $r(x)$ can be scanned aims to reduce the number of clock cycles. Multiplier III makes use of consecutive zeros in control input polynomial $r(x)$. Multiplier IV can resist certain side-channel attacks by controlling the operations for each clock cycle. Then, the FPGA complexity comparison among four multipliers with the existing works shows that the performance of Multiplier I is the best; Multiplier II has better area-latency-product, but the speed is the second-best; Multiplier III has a faster speed and Multiplier IV has the advantage to resist side-channel attacks. But Multiplier IV can be applied to only three parameter sets due to some restrictions on the control input polynomial $r(x)$.

D. J. Bernstein proposed streamlined NTRU Prime subject [8] to achieve the standard goal of IND-CCA2 security suggested by NIST. Using parameter set $n = 761, q = 4591$ and $t = 143$, the author shows a non-constant time implementation using software Sage computer-algebra system. It provides a large security margin beyond the target security level and the advantage is that cost of multiplication in NTRUEncrypt system is more expensive than streamlined NTRU Prime. However, the proposed system based on streamlined NTRU prime system has the bigger public (private) key size, and ciphertext than NTRUEncrypt cryptosystem.

An efficient Hardware/Software based on NTRUEncrypt system was proposed

in 2018 by T. Fritzmann et al. [15]. He proposed a complete NTRUEncrypt hardware and software co-design implementations on the Xilinx Zynq UltraScale+MPSoC ZCU102 platform. Moreover, he chose the single coefficient implementation as a reference used for NTRUEncrypt HW and SW. Then the full hardware and software implementations are presented. The encryption and decryption operations share common hardware modules in order to keep a low area cost and the required number of used logical gates and registers is lower based on HW/SW solution compared with only full hardware design. While for the parameter set $ees401ep1$, the padding scheme takes a longer time, which is nearly 90% of the encryption time.

A constant time software and hardware implementation of the NIST round 2 post-quantum cryptographic algorithms based on streamlined NTRU Prime was presented in 2020 by A. Marotzke [18]. The author implements the entire KEM algorithm, which includes key generation, encapsulation, and decapsulation using FPGA. Then focusing on the utility resources and comparing them to existing implementations based on streamlined NTRU prime systems, he found that his proposed design uses fewer resources but the encryption and decryption parts spend more time. Table 3.4 shows a full comparison of all metrics with parameter set $p = 761$, $q = 4591$ and $w = 246$. Note that Design A represents A. Marotzke's design; Design B is A. Marotzke's design without key generation or decoding; Design C is existing work without key generation or decoding.

Table 3.4: Comparison of Design A, B and Existing Implementations [18]

Design	Design A	Design B	Design C
Slices	1841	1261	10319
LUT	9538	6240	70066
Flip-flop	57803	6223	38144
BRAM	14	9	9
DSP	19	3	0
FMax	271 MHz	279 MHz	263 MHz
En-time	524 μs	483 μs	56.3 μs
De-time	958 μs	901 μs	53.3 μs

4 PROPOSED LFSR BASED ARCHITECTURE FOR NTRU PRIME

In this chapter, an architecture for NTRU Prime algorithm and its FPGA simulation results are presented. First of all, the linear feedback shift register (LFSR) is briefly introduced. Secondly, a proposed truncated polynomial ring multiplier for NTRU Prime, which is called TPM-I, and an LFSR based architecture is presented. Finally, the FPGA implementation results with different parameter sets corresponding to each security level are given.

4.1 Linear Feedback Shift Register

An LFSR is a shift register whose feedback value is a linear function of its previous state. In addition, It has well-known applications in (CRC) cyclic redundancy check, cryptography and the maximal-length of an LFSR is $2^n - 1$, where n is the size of registers [19]. An LFSR shown in Fig.4.1 is determined with its characteristic polynomial $f(x)$, which is in the form of

$$f(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1 x + 1, \tag{22}$$

where $f_i, i = 1, 2, \cdots, n-1$ is either 0 or 1.

In Fig.4.1, $\oplus$ represents an adder, $\otimes$ refers to a multiplier and $\square$ refers to a register. The registers are loaded with the coefficients of polynomial $A(x) = (a_{n-1}, \ldots, a_0)$. Following a shift-to-right operation, the LFSR calculates $A(x) \times x \bmod f(x)$ and the results are stored in the registers, where x is the root of its characteristic polynomial $f(x)$.

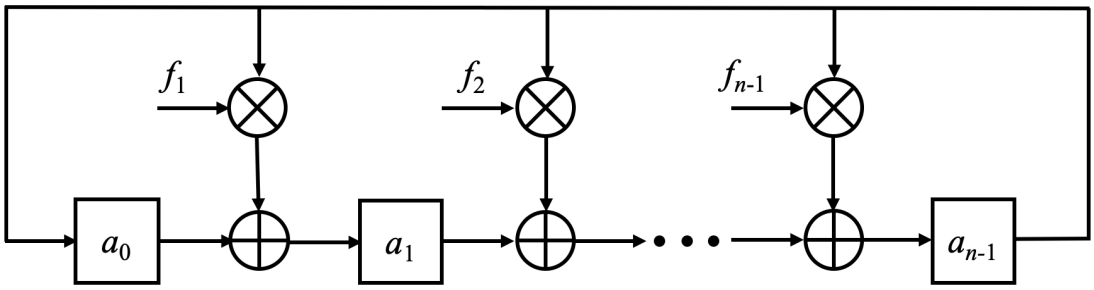


Fig. 4.1: Linear Feedback Shift Register [19]

4.2 Proposed Truncated Polynomial Ring Multiplier (TPM-I)

Since the polynomial in the truncated polynomial ring used for NTRU Prime requires to take modulo $x^n - x - 1$, LFSR can be configurated accordingly and used to implement polynomial operations in this truncated polynomial ring. The content of registers $e = (e_0, e_1, \ldots, e_{n-1})$ at clock cycle j be denoted by $e^{(j)} = (e_0^j, e_1^j, \ldots, e_{n-1}^j)$. Algorithm 4.1. shows the steps to perform truncated polynomial ring multiplication.

Algorithm 4.1 Multiplication in Truncated Polynomial Ring

Input: $h = h_{n-1}, \ldots, h_0, r = r_{n-1}, \ldots, r_0$
Output: $e = hr = e_{n-1}, \ldots, e_0$

1: $e_{(0)} := 0$
2: **for** $j := 1$ to n **do**
3: **for** $i := 0$ to $n - 1$ **do**
4: $e(j) := e_{i+1 \, mod \, n}^{(j-1)} + h_{i+1 \, mod \, n} \times r_{j-1} \bmod q$
5: **end for**
6: **end for**
7: **return** $e := e^{(n)}$

Then, an LFSR based multiplication architecture in the truncated polynomial ring is proposed and shown in Fig.4.2. It consists of n multipliers, n adders and n registers. Each register can store $\lceil log_2 q \rceil$ bits.

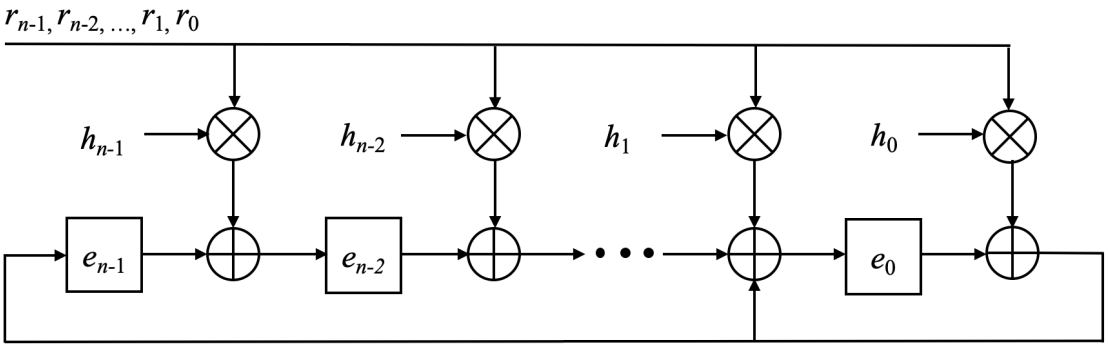


Fig. 4.2: Proposed Truncated Polynomial Ring Multiplier

In this proposed architecture, two inputs are polynomial $h(x)$ and $r(x)$, each with n coefficients and can be shown as $h(x) = (h_0, h_1, \ldots, h_{n-1})$ and $r(x) = (r_0, r_1, \ldots, r_{n-1})$. The output is the result polynomial $e(x)$ with n coefficients, which can be given as $e(x) = (e_0, e_1, \ldots, e_{n-1})$. The registers $e = (e_{n-1}, \cdots, e_1, e_0)$ are initialized with 0. After n clock cycles, the registers will store the product $h(x) \times r(x) \bmod (x^n - x - 1)$.

4.3 Proposed Arithmetic Unit

During clock cycle j, the coefficient r_j of the input polynomial $r(x)$ multiplies with the coefficients of polynomial $h(x)$. Since r_j picks up a value from $\{-1, 0, 1\}$, this operation can be evaluated without any integer multiplications, which means these operations can be calculated with only addition operation with modular arithmetic. In the proposed modular arithmetic unit, coefficient of $r(x)$ is encoded in 2 bits as $r_{(1)}r_{(0)}$. Coefficients of $h(x)$ and $e(x)$, h and e, are encoded in $m = \lceil log_2 q \rceil$ bits. Two input lines r, '11', '00' and '01' are used to represent the value of '-1', '0' and '1' for $r(x)$ coefficient, respectively. More specifically, if $r_j = -1$, subtraction operation $e_i = e_{i+1} - h_i \bmod q$ can be evaluated by

$$e_i^{(j+1)} = e_{i+1}^{(j)} + \overline{h_i} + 1 \tag{23}$$

Furthermore, Proposed Arithmetic Unit (AU) and its operations table are de-

signed for our design, which are shown as Table 4.1 and Fig.4.3.

Table 4.1: Operations with the Modular Arithmetic Unit

Input $r(r_{(1)}r_{(0)})$	Output s
01	$e + h$
11	$e + \overline{h} + 1$
00	e

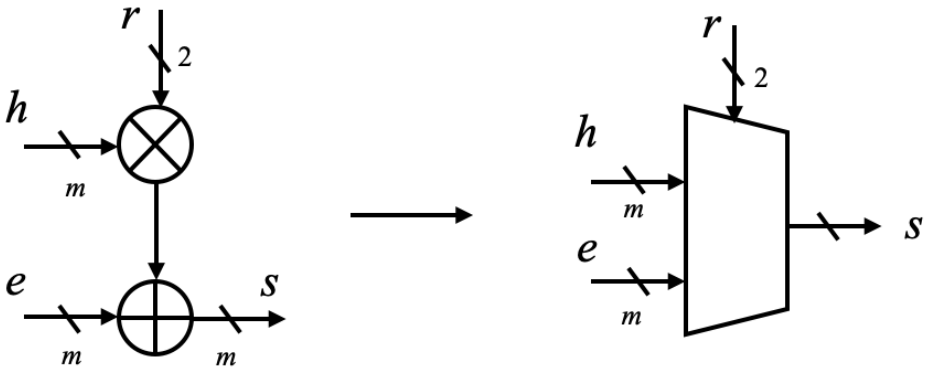


Fig. 4.3: Proposed Arithmetic Unit Architecture

Algorithm 4.2 shows several steps to perform operations in hardware and the Proposed AU architecture is shown in Fig.4.4.

Algorithm 4.2 Proposed Arithmetic Unit

Input: $e = e_{m-1}, \ldots, e_0, h = h_{m-1}, \ldots, h_0, r = r_1, r_0$
Output: $s_{m-1}, \ldots, s_0$
1: **if** $r_{(0)} = 0$ **then**
2: $(s_{(m-1)}, \ldots, s_{(1)}, s_{(0)}) = (e_{(m-1)}, \ldots, e_{(1)}, e_{(0)})$
3: **else**
4: $(h_{(m-1)}, \ldots, h_{(1)}, h_{(0)}) = (h_{(m-1)} \oplus r_{(1)}), \ldots, (h_{(1)} \oplus r_{(1)}), (h_{(0)} \oplus r_{(1)})$;
5: $(s_{(m-1)}, \ldots, s_{(1)}, s_{(0)}) = (e_{(m-1)}, \ldots, e_{(1)}, e_{(0)}) + (h_{m-1)}, \ldots, h_{(1)}, h_{(0)}) + r_{(1)}$;
6: **end if**
7: **return** $s = (s_{(m-1)}, \ldots, s_{(1)}, s_{(0)})$

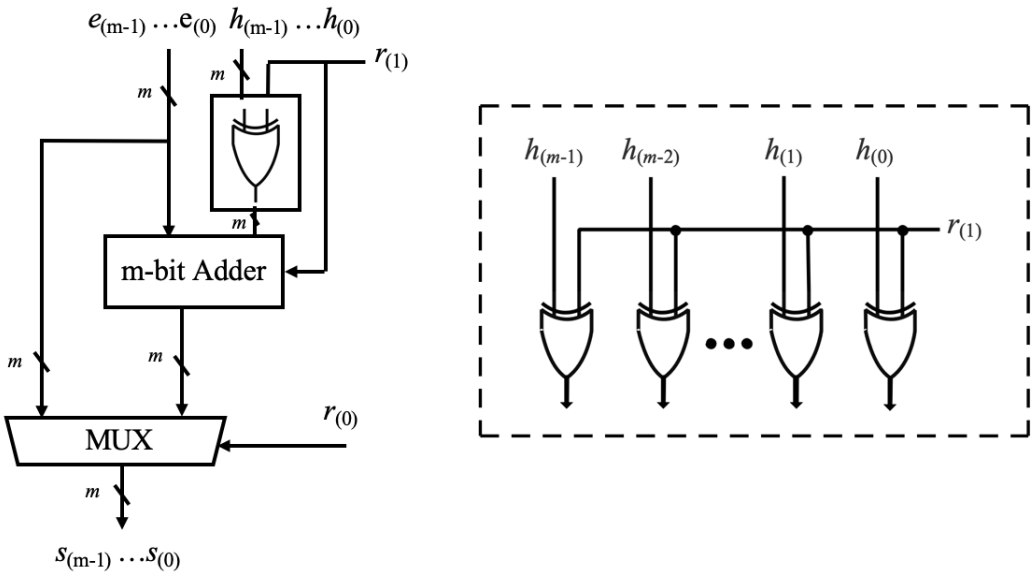


Fig. 4.4: Proposed Arithmetic Unit Circuit

4.4 TPM-I for NTRU Prime Encryption

Based on the proposed arithmetic unit, a multiplier for encryption in streamlined NTRU prime system is shown. Operations in encryption part is $e(x) = h(x) \times r(x) + m(x) \bmod q \bmod x^n - x - 1$. $h(x)$ is the public key, $r(x)$ is a randomly chosen polynomial and $m(x)$ is the message polynomial. Note that the coefficients of $r(x)$ and $m(x)$ are from $\{-1, 0, 1\}$. Then a detailed algorithm for TPM-I based encryption is shown in Algorithm 4.3, Fig.4.5 shows the architecture for encryption, which contains n registers and n AUs.

Algorithm 4.3 TPM-I: LFSR Based multiplication for NTRU Prime

Input: $h = h_{n-1}, \ldots, h_0, r = r_{n-1}, \ldots, r_0, m = m_{n-1}, \ldots, m_0$
Output: $e = e_{n-1}, \ldots, e_0$

1: $e_{(0)} = m$
2: **for** $j = 1$ to n **do**
3: **for** $i = 0$ to $n - 1$ **do**
4: $e(j) = e^{(j-1)}_{i+1 \, mod \, n} + h_{i+1 \, mod \, n} \times r_{j-1} \bmod q \bmod x^n - x - 1$
5: **end for**
6: **end for**
7: **return** $e = e^{(b)}$

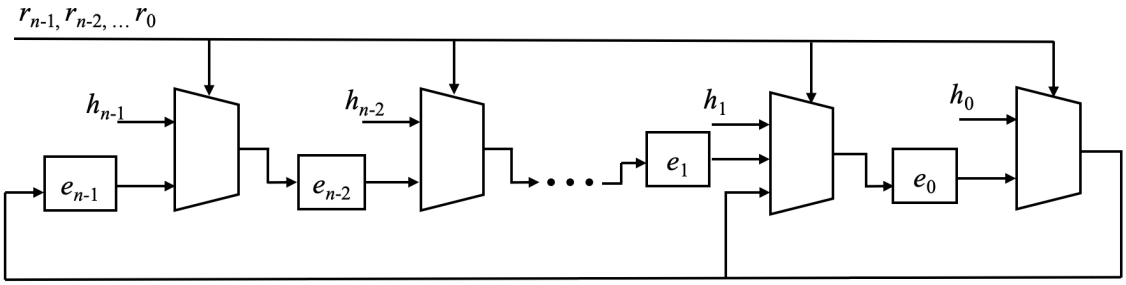


Fig. 4.5: TPM-I for NTRU Prime Encryption

This architecture contains n registers and n arithmetic units in total. The registers $e = (e_{n-1}, \ldots e_1, e_0)$ are initially loaded with $m = (m_{n-1}, \ldots, m_1, m_0)$. After n clock cycles, the architecture generates $e(x) = h(x) \times r(x) + m(x) \bmod q \bmod x^n - x - 1$. The content of registers at cycle j, j = 0,1, …, n, is given in Table 4.2.

Table 4.2: Register contents when TPM-I performing NTRU Prime encryption

Cycle j	Input r_j	$e_{n-1}^{(j)}$	$e_{n-2}^{(j)}$	$\ldots$	$e_0^{(j)}$
0	–	m_4	m_3	$\ldots$	m_0
1	r_0	$m_0 + h_0 r_0$	$m_4 + h_4 r_0$	$\ldots$	$m_1 + h_1 r_0$
2	r_1	$m_1 + h_1 r_0 + h_0 r_1$	$m_0 + h_0 r_0 + h_4 r_1$	$\ldots$	$m_2 + h_2 r_0 + h_1 r_1$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
n	r_{n-1}	$m_{n-1} + h_{n-1} r_0 + \ldots + h_{n-1} r_{n-1}$	$m_{n-2} + h_{n-2} r_0 + \ldots + h_{n-1} r_{n-1}$	$\ldots$	$m_0 + h_0 r_0 + \ldots + h_0 r_{n-1}$
$e = m + r * h$		$= e_{n-1}^{(n)}$	$= e_{n-2}^{(n)}$	$\ldots$	$= e_0^{(n)}$

4.5 FPGA Implementation

FPGA is an integrated circuit that can be reconfigured by a customer or a designer. In this way, FPGA can be used as a fast hardware implementation tool and has wide applications. In this subsection, FPGA implementations of NTRU Prime encryption architecture based on LFSR will be presented. FPGA results are also summarized.

FPGAs are designed in a higher description language called HDL, which is a modular programming code. Using HDL code, which includes VHDL or Verilog makes the design process extremely fast and efficient. In our implementation, we use Verilog HDL as our design language.

More specifically, the following tools are used for our implementation.

- Quartus II v14.1 (64-bit) Software
- ModelSim-Altera Software

Arria V 5AGXFB3H4F35I3 was chosen as the target device to provide implementation, which offers the highest bandwidth and delivers the lowest total power for midrange applications, such as remote radio units and broadcast studio equipment.

Implementation Results of TPM-I

The simulation results for different parameter sets are shown in Table 4.3.

Table 4.3: Simulation Results for Different Parameter Sets

Security Level	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
112	$ees401ep1$	6817	8826	402	274.80 MHz	1.46 μs
	$ees541ep1$	9197	11906	542	283.93 MHz	1.91 μs
	$ees659ep1$	11203	14502	660	282.17 MHz	2.33 μs
128	$ees449ep1$	7633	9882	450	280.35 MHz	1.60 μs
	$ees613ep1$	10421	13490	614	284.50 MHz	2.15 μs
	$ees761ep1$	12937	16746	762	281.85 MHz	2.70 μs
192	$ees677ep1$	11509	14678	678	274.73 MHz	2.46 μs
	$ees887ep1$	15079	19518	888	263.99 MHz	3.36 μs
	$ees1087ep1$	18479	23918	1088	284.26 MHz	3.82 μs
256	$ees1087ep2$	18479	23918	1088	284.26 MHz	3.82 μs
	$ees1171ep1$	19907	25766	1172	241.20 MHz	4.85 μs
	$ees1499ep1$	25483	32982	1500	225.99 MHz	6.63 μs

For each parameter set, we mainly focus on five aspects of our proposed architecture.

- #ALMs, the number of used logic elements by system.
- #Register, the number of registers.
- #Cycles, the number of clocks cycles.
- FMax, the maximum operating frequency of the system.
- Latency, required encryption time.

In our simulation, the number of registers defines how many registers will be required in the architecture. In addition, the combination of the number of ALMs and the number of registers will decide the area consumption. The number of clock cycles shows how many clock cycles are required for the computation. FMax means the maximum frequency that can be achieved during the operation, which usually depends on the FPGA chip that we use. Latency is calculated by the number of clock cycles divided by FMax, which reflects the time consumption for the encryption.

To the best of our knowledge, this is the first time that NTRU Prime has been implemented in FPGA. In the next chapter, we will present more efficient architectures by making significant modifications to LFSR and utilizing novel encoding techniques.

5 THREE PROPOSED EFFICIENT MULTIPLIERS FOR NTRU PRIME

In this chapter, three time-efficient multiplication architectures, called TPM-II, TPM-III, and TPM-IV, are designed for NTRU Prime system. Their FPGA simulation results are obtained. Note that the three proposed works are based on TPM-I, each of them is the optimized version of the previous one.

5.1 Proposed Efficient Multiplication and its FPGA Implementation (TPM-II)

5.1.1 Proposed x^2-net Architecture

x-net architecture is the core architecture in truncated polynomial ring multiplier, which is shown in Fig.5.1 [12]. In x-net architecture, r_i represents ith coefficient of the polynomial $r(x)$, which is the control input during clock cycle i+1. A register content $e_k(k = 0, \ldots, n-1)$ at clock cycle j is defined by $e_k^{(j)}$ and (h_{n-1}, ..., h_0) are the corresponding coefficients of the polynomial $h(x)$. One arithmetic unit includes one adder, one multiplier and there are totally n arithmetic units in x-net architecture. Note that each arithmetic unit has three inputs and one output.

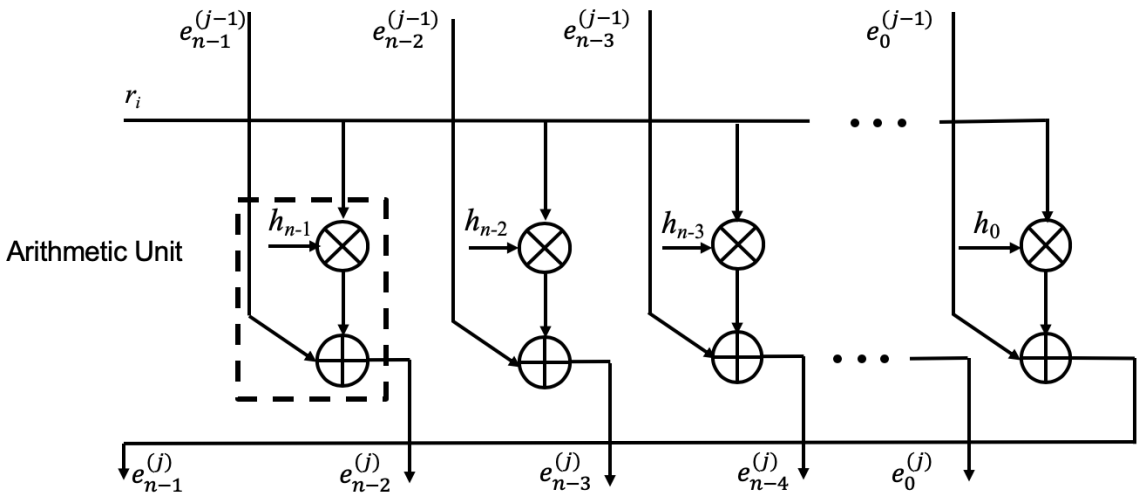


Fig. 5.1: x-net Architecture [12]

Then Fig.5.2 shows a new design of x^2-net architecture which is an extension to x-net architecture. In x^2-net architecture, r_i is a control input, which represents ith coefficient of the polynomial $r(x)$ and a new input r_{i-1} represents $(i-1)$th coefficient of the polynomial $r(x)$. $(h_{n-1}, \ldots, h_0)$ are the corresponding coefficients of the polynomial $h(x)$ and a register content $e_k (k = 0, \ldots, n-1)$ is defined by $e_k^{(j)}$ during clock cycle j. In addition, we can see that one arithmetic unit doubles two adders, two multipliers and there are totally n arithmetic units required in x^2-net architecture. Note that each arithmetic unit has five inputs and then we modify shift-to-right operation. We change x-net architecture multiplies with x to x^2-net architecture multiplies with x^2 in one shift-to-right operation. In another word, there are two coefficients of the polynomial $r(x)$ can be processed during one clock cycle, which means that half number of clock cycles can be saved.

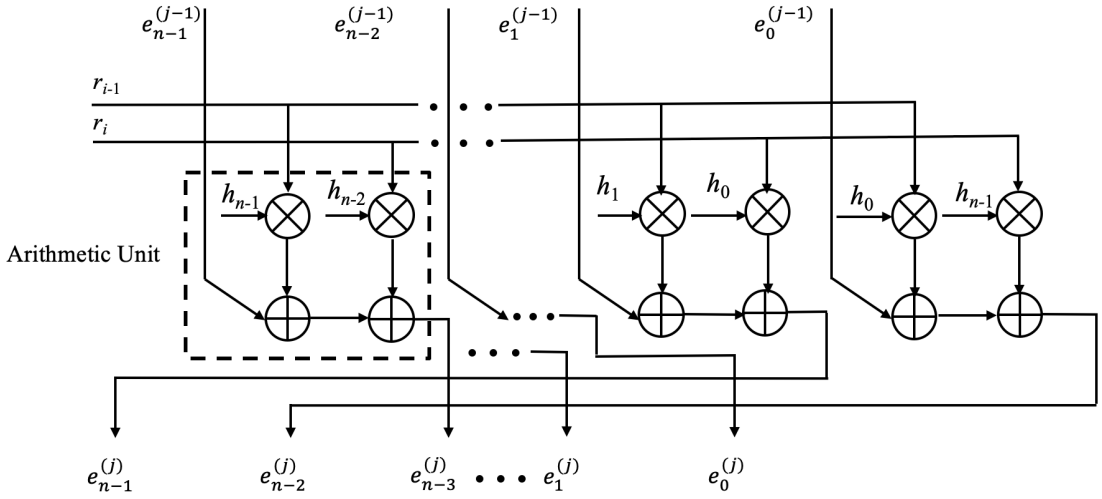


Fig. 5.2: x^2-net Architecture

5.1.2 Proposed Arithmetic Unit

For NTRU Prime encryption, the coefficients of input polynomial $r(x)$ are chosen from $\{-1, 0, 1\}$, so the multiplication $h(x) \times r(x)$ can be evaluated without any integer multiplication, which means addition and subtraction operations are performed instead of the multiplication operations. In addition, each register can store $m = \lceil log_2 q \rceil$ bits, which means that the parameter q is in the form of 2^n. Fig.5.3 shows an optimized arithmetic unit in x^2-net architecture.

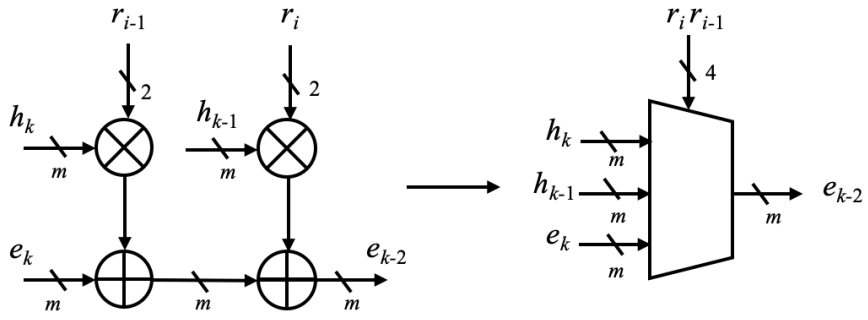


Fig. 5.3: Proposed Arithmetic Unit

From new design of arithmetic unit, the number of inputs is increased to four. h_i, h_{i-1} and e_i are encoded in $m = [log_2 q]$ bits and the control input r_i and r_{i-1} are

combined in order to get a new input $r_i r_{i-1}$, which is encoded in four bits. Output e_{i-2} is encoded in $m = [log_2 q]$ bits. In addition, binary representation '11', '00' and '01' are used to represent the coefficients '-1', '0' and '1' in $r(x)$, respectively. Table 5.1 shows the operations supported with new arithmetic unit.

Table 5.1: Operations Performed at the Arithmetic Unit

Input $r_i r_{i-1}(r_1^{(i)} r_0^{(i)} r_1^{(i-1)} r_0^{(i-1)})$	Output e_{k-2}
0000	e_k
0001	$e_k + h_k \bmod q$
0011	$e_k - h_k \bmod q$
0100	$e_k + h_{k-1} \bmod q$
0101	$e_k + h_k + h_{k-1} \bmod q$
0111	$e_k - h_k + h_{k-1} \bmod q$
1100	$e_k - h_{k-1} \bmod q$
1101	$e_k + h_k - h_{k-1} \bmod q$
1111	$e_k - h_k - h_{k-1} \bmod q$

Algorithm 5.1 performs steps of this arithmetic unit.

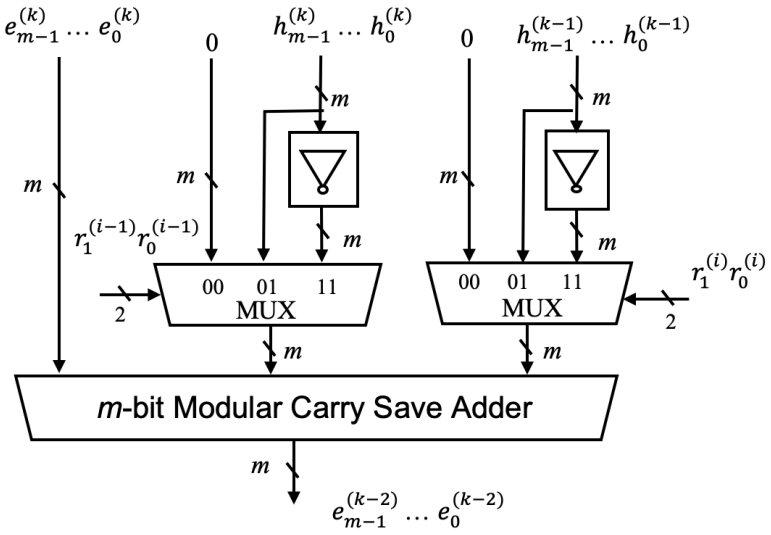
Algorithm 5.1 Proposed Algorithm for Arithmetic Unit

Input: $e = (e_{m-1}^{(k)}, \ldots, e_0^{(k)})_2, r_1 r_{i-1} = (r_1^{(i)} r_0^{(i)} r_1^{(i-1)} r_0^{(i-1)})_2$, $h_k = (h_{m-1}^{(k)}, \ldots, h_0^{(k)})_2$, $h_{k-1} = (h_{m-1}^{(k-1)}, \ldots, h_0^{(k-1)})_2$

Output: $e_{k-2} = (e_{m-1}^{(k-2)}, \ldots, e_0^{(k-2)})_2$

1: **if** $r_1^{(i-1)} r_0^{(i-1)} = 00$ **then**
2: $\quad (h_{m-1}^{(k)} \ldots h_0^{(k)}) := 0$
3: **else if** $r_1^{(i-1)} r_0^{(i-1)} = 11$ **then**
4: $\quad (h_{m-1}^{(k)} \ldots h_0^{(k)}) := -(h_{m-1}^{(k)} \ldots h_0^{(k)})$
5: **else if** $r_1^{(i)} r_0^{(i)} = 00$ **then**
6: $\quad (h_{m-1}^{(k-1)} \ldots h_0^{(k-1)}) := 0$
7: **else if** $r_1^{(i)} r_0^{(i)} = 11$ **then**
8: $\quad (h_{m-1}^{(k-1)} \ldots h_0^{(k-1)}) := -(h_{m-1}^{(k-1)} \ldots h_0^{(k-1)})$
9: **end if**
10: $(e_{m-1}^{(k-2)}, \ldots, e_0^{(k-2)}) := (e_{m-1}^{(k)}, \ldots, e_0^{(k)}) + (h_{m-1}^{(k)}, \ldots, h_0^{(k)}) + (h_{m-1}^{(k-1)}, \ldots, h_0^{(k-1)})$

Then the architecture for the proposed arithmetic unit is shown in Fig.5.4.



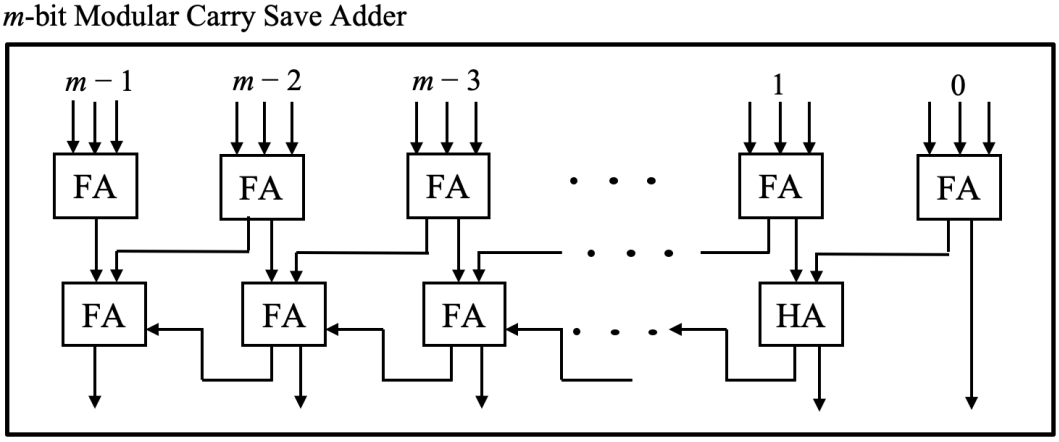


Fig. 5.4: Proposed Arithmetic Unit Architecture

5.1.3 TPM-II for Encryption

During NTRU Prime encryption, the ciphertext is computed by $e(x) = h(x) \times r(x) + m(x) \bmod q \bmod x^n - x - 1$. The coefficients of input $r(x)$ and message $m(x)$ are randomly choosen from $\{-1, 0, 1\}$. A new multiplier computing $e(x)$ with inputs $h(x)$, $r(x)$, and $m(x)$ based on the proposed arithmetic unit is presented.

Algorithm 5.2 shows a detailed algorithm for TPM-II encryption. Note that inputs $h(x), r(x), m(x)$ and output $e(x)$ are the polynomials with n coefficients.

Algorithm 5.2 Proposed multiplication for NTRU Prime (TPM-II)

Input: $m = m_{n-1}, \ldots, m_0, r = r_{n-1}, \ldots, r_0, h = h_{n-1}, \ldots, h_0$
Output: $e = e_{n-1}, \ldots, e_0 = hr + m \bmod q \bmod x^n - x - 1$

$e^{(0)} := m$
for $j := 1$ to $(\frac{n+1}{2})$ **do**
  for $i := 0$ to $n-1$ **do**
    if $r_{2j-1} = 00$ **then**
      if $r_{2j-2} = 00$ **then**
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)}$
      else if $r_{2j-2} = 01$ **then**
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} + h_{i+2 \bmod n} \bmod q$
      else
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} - h_{i+2 \bmod n} \bmod q$
      end if
    else if $r_{2j-1} = 01$ **then**
      if $r_{2j-2} = 00$ **then**
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} + h_{i+1 \bmod n} \bmod q$
      else if $r_{2j-2} = 01$ **then**
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} + h_{i+2 \bmod n} + h_{i+1 \bmod n} \bmod q$
      else
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} - h_{i+2 \bmod n} + h_{i+1 \bmod n} \bmod q$
      end if
    else
      if $r_{2j-2} = 00$ **then**
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} - h_{i+1 \bmod n} \bmod q$
      else if $r_{2j-2} = 01$ **then**
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} + h_{i+2 \bmod n} - h_{i+1 \bmod n} \bmod q$
      else
        $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} - h_{i+2 \bmod n} - h_{i+1 \bmod n} \bmod q$
      end if
    end if
  end for
end for
return $e := e^{(\frac{n+1}{2})}$

The proposed architecture for TPM-II is also given in Fig.5.5.

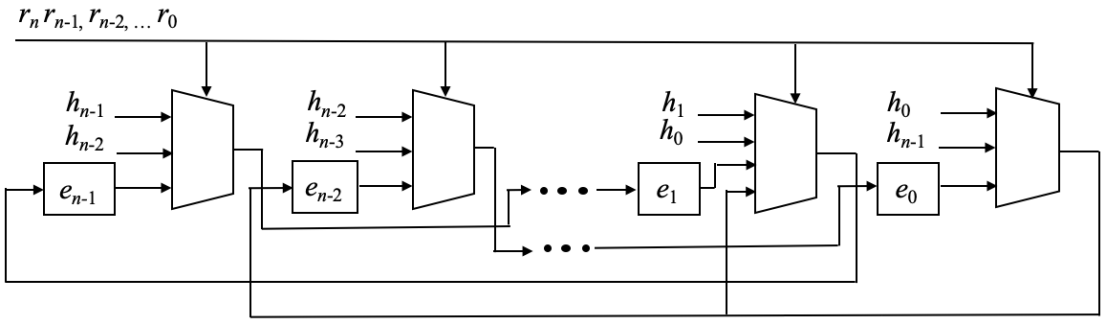


Fig. 5.5: TPM-II Architecture

The architecture contains n registers and n arithmetic units in total. When the operation starts, during one clock cycle, two consecutive coefficients of the input polynomial $r(x)$ are scanned. As a result, the polynomial $r(x)$ has n coefficients, we assume $r_n = 0$ and encode it as '00'. After $\frac{N+1}{2}$ clock cycles, the encryption result will be stored in the registers $e = (e_{n-2}, \cdots, e_0, e_{n-1})$.

5.1.4 Implementation of TPM-II

The following tools are used for our implementation.

- Quartus II v14.1 (64-bit) Software
- ModelSim-Altera Software

Arria V 5AGXFB3H4F35I3 was chosen as the target device in provided implementation and we use Verilog HDL as design language.

Implementation Results

The implementation results are shown in Table 5.2.

Table 5.2: FPGA Results for Different Parameter Sets

Security Level	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
112	$ees401ep1$	11904	8826	201	239.69 MHz	0.83 μs
	$ees541ep1$	16025	11906	271	240.91 MHz	1.12 μs
	$ees659ep1$	19539	14502	330	242.19 MHz	1.36 μs
128	$ees449ep1$	13327	9882	225	256.21 MHz	0.87 μs
	$ees613ep1$	18200	13490	307	246.97 MHz	1.24 μs
	$ees761ep1$	22558	16746	381	234.58 MHz	1.62 μs
192	$ees677ep1$	19737	14678	339	248.82 MHz	1.36 μs
	$ees887ep1$	26353	19518	444	239.01 MHz	1.85 μs
	$ees1087ep1$	32241	23918	544	222.82 MHz	2.44 μs
256	$ees1087ep2$	32241	23918	544	222.82 MHz	2.44 μs
	$ees1171ep1$	34848	25766	586	220.70 MHz	2.65 μs
	$ees1499ep1$	48723	32982	750	209.86 MHz	3.57 μs

5.2 Proposed Multiplication and its FPGA Implementation (TPM-III)

In this section, a time-efficient multiplication architecture is designed for NTRU Prime system. Its FPGA simulation results are obtained. Firstly, some background of our design is introduced. Secondly, a new arithmetic unit and a multiplier architecture, called TPM-III are shown. Finally, FPGA implementation results with different parameter sets in each security levels are obtained.

5.2.1 Basic Idea

From this architecture, the control input r_i is encoded in two bits. Thus, it has three states, '00', '10', and '11'. The state '10' is considered as a redundant state and we assumed that the architecture will be more efficient if using the redundant state.

According to the parameter selection, $r(x) \in L(d_r, d_r)$ and d_r is much smaller than the parameter n. Then, there has a large number of coefficients equal to 0 in $r(x)$ and its results can be implemented in the proposed architecture [13].

Especially, the proposed architecture is based on counting how many '0, 0, 0' coefficient pairs in $r(x)$ can be processed during one clock cycle. Then it is considered that the reduction in the number of clock cycles can be achieved. Therefore, finding out how many '0, 0, 0' coefficient pairs appear consecutively in $r(x)$ [12] is required.

The number of '1' and '-1' coefficients in $r(x)$ is defined by

$$n_1 = n_{-1} = d_r \tag{24}$$

The number of '0' coefficients in $r(x)$ can be calculated as

$$n_0 = n - 2d_r \tag{25}$$

The minimum number of '0, 0, 0' coefficient pairs in $r(x)$ can be calculated as

$$n_{000max} = n - 2d_r/3 \tag{26}$$

The maximum number of '0, 0, 0' coefficient pairs in $r(x)$ can be calculated as

$$n_{000max} = \begin{cases} 0 & n_1 + n_{-1} > \dfrac{n_0}{2} - 1 \\ n - 6d_r/3 & n_1 + n_{-1} < \dfrac{n_0}{2} - 1 \end{cases} \tag{27}$$

As we can see, the number of '0, 0, 0' coefficient pairs in different parameter sets is shown in Table 5.3. For each parameter set, first of all, the column # '0' is listed in the table after calculating, which means the total number of '0' coefficients. In the column # '0, 0, 0', there are three separated sub-columns Min, Max, and Avg, which represent the minimum number of '0, 0, 0' coefficient pairs, the maximum number of '0, 0, 0' coefficient pairs and the average number of '0, 0, 0' coefficient pairs in $r(x)$ respectively.

Table 5.3: Number of '0, 0, 0' Pairs in Different Parameter Sets [12]

Security level	Parameter set	n	#"0"	#'0,0,0'		
				Min	Max	Avg
112	$ees401ep1$	401	175	0	58	20.30
	$ees541ep1$	541	443	82	147	119.03
	$ees659ep1$	659	583	143	194	170.76
128	$ees449ep1$	449	181	0	60	18.62
	$ees613ep1$	613	503	94	167	135.51
	$ees761ep1$	761	677	169	225	199.54
192	$ees677ep1$	677	363	0	121	57.01
	$ees887ep1$	887	725	133	241	194.59
	$ees1087ep1$	1087	961	236	320	281.47
256	$ees1087ep2$	1087	847	122	282	215.22
	$ees1171ep1$	1171	959	178	319	258.05
	$ees1499ep1$	1499	1341	341	447	397.53

Then the average number of clock cycles for different parameter sets are given in Table 5.4.

Table 5.4: Average Number of Cycles for Different Parameter Sets [12]

Security level	Parameter set	n	p	#Cycles
112	$ees401ep1$	401	3	381
	$ees541ep1$	541	3	422
	$ees659ep1$	659	3	489
128	$ees449ep1$	449	3	431
	$ees613ep1$	613	3	478
	$ees761ep1$	761	3	562
192	$ees677ep1$	677	3	620
	$ees887ep1$	887	3	693
	$ees1087ep1$	1087	3	806
256	$ees1087ep2$	1087	3	872
	$ees1171ep1$	1171	3	913
	$ees1499ep1$	1499	3	1102

5.2.2 Proposed Arithmetic Unit

This arithmetic unit has three parts: inputs h_i, e_i and output e_{i-1}, which are encoded in $m = \lceil log_2 q \rceil$ bits. Then, t_i is considered as a control input which is encoded in two

bits. Next e_{k+2} is added as a new input in our arithmetic unit, which is also encoded in $m = \lceil log_2 q \rceil$ bits. Fig.5.6 shows the proposed arithmetic unit.

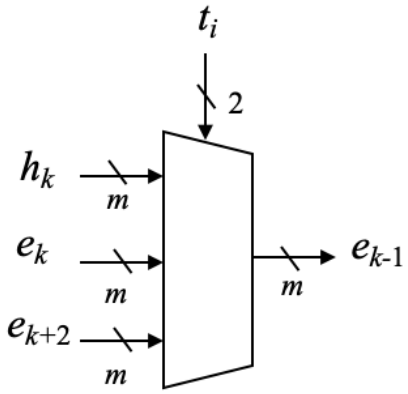


Fig. 5.6: Proposed Arithmetic Unit

The operation with arithmetic unit is shown below as Table 5.5.

Table 5.5: Operations performed at the Arithmetic Unit

r_j, r_{j+1}, r_{j+2}	Input $t_i(t_1^i t_0^i)$	Output e_{k-1}
0, $\times$, $\times$	00	e_k
1, $\times$, $\times$	01	$e_k + h_k \bmod q$
0, 0, 0	10	e_{k+2}
-1, $\times$, $\times$	11	$e_k - h_k \bmod q$

Algorithm 5.3 shows an algorithm that performs each step of this arithmetic unit and Fig.5.7 shows the architecture for proposed arithmetic unit.

Algorithm 5.3 Proposed Algorithm for Arithmetic Unit

Input: $e_k = (e_{m-1}^{(k)}, \ldots, e_0^{(k)})_2; e_{k+2} = (e_{m-1}^{(k+2)}, \ldots, e_0^{(k+2)})_2; h_k = (h_{m-1}^{(k)}, \ldots, h_0^{(k)})_2; t_i = (t_1^{(i)} t_0^{(i)})_2$

Output: $e_{k-1} = (e_{m-1}^{(k-1)}, \ldots, e_0^{(k-1)})_2$

1: **if** $t_i = t_1^{(i)} t_0^{(i)} = 00$ **then**
2: $\quad (e_{m-1}^{(k-1)} \ldots e_0^{(k-1)}) := (e_{m-1}^{(k)}, \ldots, e_0^{(k)})$
3: **else if** $t_i = t_1^{(i)} t_0^{(i)} = 01$ **then**
4: $\quad (e_{m-1}^{(k-1)} \ldots e_0^{(k-1)}) := (e_{m-1}^{(k)}, \ldots, e_0^{(k)}) + (h_{m-1}^{(k)}, \ldots, h_0^{(k)})$
5: **else if** $t_i = t_1^{(i)} t_0^{(i)} = 10$ **then**
6: $\quad (e_{m-1}^{(k-1)} \ldots e_0^{(k-1)}) := (e_{m-1}^{(k+2)}, \ldots, e_0^{(k+2)})$
7: **else**
8: $\quad (e_{m-1}^{(k-1)} \ldots e_0^{(k-1)}) := (e_{m-1}^{(k)}, \ldots, e_0^{(k)}) - (h_{m-1}^{(k)}, \ldots, h_0^{(k)})$
9: **end if**

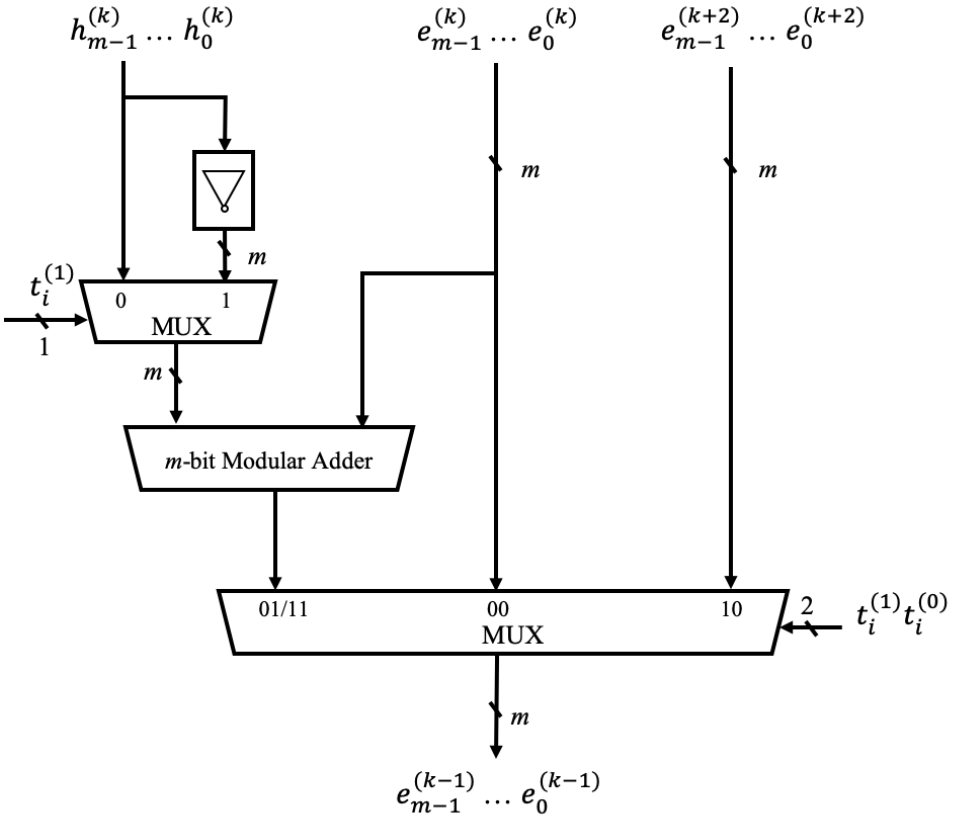


Fig. 5.7: Proposed Arithmetic Unit Architecture

5.2.3 TPM-III for Encryption

In this chapter, a time-efficient architecture for NTRU Prime system based on a new arithmetic unit are shown. The new architecture process three consecutive zero coefficients in $r(x)$ during one clock cycle. First of all, encoding $r(x) = (r_{n-1}, \ldots, r_0)$ to obtain a new control input sequence $(t_{u-1}, \ldots, t_0)$, where each t_i has two bits. If three consecutive zero coefficients occur in $r(x)$, '10' will be encoded to define this set and for other coefficients '-1', '0', '1', we encode them as '11', '00' and '01' respectively

Algorithm 5.4 and Fig.5.8 show the proposed architecture for encryption. In the architecture, there are three inputs, which are public key $h(x)$, message $m(x)$, the generated sequence $(t_{u-1}, \ldots, t_0)$, $e(x)$ with n coefficients and one output $e(x)$.

Algorithm 5.4 Proposed Multiplication for NTRU Prime (TPM-III)

Input: $m = m_{n-1}, \ldots, m_0; h = h_{n-1}, \ldots, h_0; (t_{u-1}, \ldots, t_0)$
Output: $e = e_{n-1}, \ldots, e_0 = hr + m \bmod q \bmod x^n - x - 1$

$e^{(0)} := m$
for $j := 1$ to u **do**
  for $i := 0$ to $n-1$ **do**
    if $t_{j-i} = 00$ **then**
      $e_i^{(j)} := e_{i+1 \, mod \, n}^{(j-1)}$
    else if $t_{j-i} = 10$ **then**
      $e_i^{(j)} := e_{i+3 \, mod \, n}^{(j-1)}$
    else if $t_{j-i} = 01$ **then**
      $e_i^{(j)} := e_{i+1 \, mod \, n}^{(j-1)} + h_{i+1 \, mod \, n} \bmod q$
    else
      $e_i^{(j)} := e_{i+1 \, mod \, n}^{(j-1)} - h_{i+1 \, mod \, n} \bmod q$
    end if
  end for
end for
return $e := e^{(u)}$

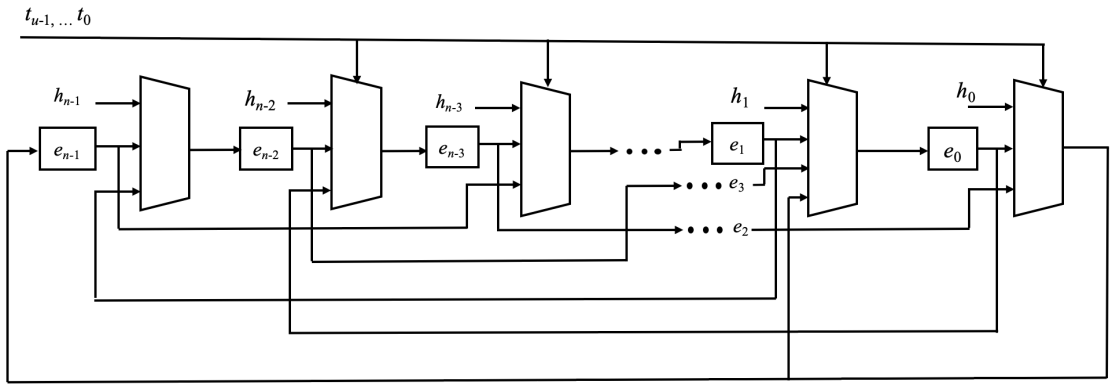


Fig. 5.8: TPM-III Architecture

The architecture contains n registers and n AUs. The registers $e = (e_{n-1}, \ldots, e_0)$ are initially loaded with $m = (m_{n-1}, \ldots, m_0)$. During one clock cycle, each t_i is scanned. After u clock cycles, the resulst will be stored in the registers $e = (e_{n-1}, \ldots, e_0)$.

5.2.4 Implementation of TPM-III

- Quartus II v14.1 (64-bit) Software
- ModelSim-Altera Software

Arria V 5AGXFB3H4F35I3 was chosen as the target device in provided implementation and we use Verilog HDL as design language.

Implementation Results

The implementation results are shown in Table 5.6.

Table 5.6: FPGA Results for Different Parameter Sets

Security Level	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
112	$ees401ep1$	6888	8826	381	303.12 MHz	1.25 μs
	$ees541ep1$	9256	11906	422	304.41 MHz	1.38 μs
	$ees659ep1$	11267	14502	489	284.74 MHZ	1.71 μs
128	$ees449ep1$	7731	9882	431	300.30 MHz	1.43 μs
	$ees613ep1$	10508	13490	478	289.86 MHz	1.65 μs
	$ees761ep1$	13023	16746	562	271.74 MHz	2.07 μs
192	$ees677ep1$	11566	14678	620	267.52 MHz	2.32 μs
	$ees887ep1$	15172	19518	693	264.27 MHz	2.62 μs
	$ees1087ep1$	18613	23918	806	257.53 MHz	3.13 μs
256	$ees1087ep2$	18613	23918	872	257.53 MHz	3.38 μs
	$ees1171ep1$	19972	25766	913	232.94 MHz	3.92 μs
	$ees1499ep1$	25595	32982	1102	245.58 MHz	4.48 μs

5.3 Proposed Multiplier and its FPGA Implementation (TPM-IV)

In this section, a time-efficient multiplication architecture is designed for NTRU Prime system. Firstly, our design is basically introduced. Secondly, an arithmetic unit and a multiplier architecture, which is called TPM-IV are proposed. Finally, FPGA implementation results with different parameter sets in each security level are shown.

5.3.1 Basic Idea

In TPM-III, the control input t_i is encoded as two bits and four encoding states of two bits are fulled used. However, In the new design, the control input t_i is increased from two bits to three bits in order to get eight encoding states.

Our proposed architecture is based on the idea that if different coefficient pairs in $r(x)$ can be processed during one clock cycle, aim to get if the number of clock cycles can be reducted. Hence, we require to find out how many coefficient pairs meet a certain requirement.

Table 5.8 shows the detailed encoding of t_i and r_j represents jth coefficient in $r(x)$, when we encode t_i, we scan each coefficient in $r(x)$. Such encoding does not contain one zero or two consecutive zeros coefficients in $r(x)$, we focus on a phase-shift value during the last scan. Firstly, the phase-shift value will be one, if the last scan is one zero and there is no clock cycles that can be added. Secondly, if the last scan is two consecutive zeros, the phase-shift value will be two and there is no clock cycle. For the rest, the phase-shift value will be zero and the number of clock cycles will be added by one.

The number of clock cycles for each parameter set can be calculated and shown in Table 5.8. Several steps need to be followed. First of all, one hundred sets of polynomial $r(x)$ for each parameter set are randomly generated. Secondly, starting to encode polynomial $r(x) = (r_{n-1}, \ldots, r_0)$ to obtain $(t_{u-1}, \ldots, t_0)$. At last, counting the number of t_i for each polynomial set, which is the number of clock cycles we are looking for.

From Table 5.7, the number of clock cycles for all parameter sets can be get from [12].

Table 5.7: Average Number of Cycles for Different Parameter Sets

Security level	Parameter set	n	p	#Cycles
112	$ees401ep1$	401	3	246
	$ees541ep1$	541	3	196
	$ees659ep1$	659	3	213
128	$ees449ep1$	449	3	286
	$ees613ep1$	613	3	222
	$ees761ep1$	761	3	243
192	$ees677ep1$	677	3	367
	$ees887ep1$	887	3	323
	$ees1087ep1$	1087	3	350
256	$ees1087ep2$	1087	3	420
	$ees1171ep1$	1171	3	424
	$ees1499ep1$	1499	3	473

5.3.2 Proposed Arithmetic Unit

The new design of arithmetic unit has five inputs $h_{k+1}, e_{k+1}, e_{k+2}, e_{k+3}, e_{k+4}$ and one output e_k, which are encoded in $m = \lceil log_2 q \rceil$ bits. t_i is control input, which is encoded in three bits. Note that there is no redundant state in the design, eight states of three bits are fully used. Fig.5.9 and Table 5.8 show the proposed arithmetic unit and the operation respectively.

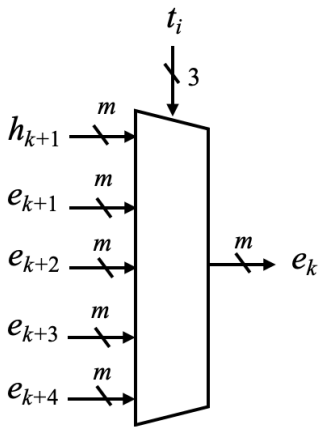


Fig. 5.9: Proposed Arithmetic Unit

Table 5.8: Operations performed at the Arithmetic Unit

$r_j, r_{j+1}, r_{j+2}, r_{j+3}$	Input $t_i(t_2^{(i)}t_1^{(i)}t_0^{(i)})$	Output e_k
0, 0, 0, 0	000	e_{k+4}
0, 0, 0, ± 1	001	e_{k+3}
0, 0, 1, $\times$	010	$e_{k+3} + h_{k+1} \bmod q$
0, 0, -1, $\times$	011	$e_{k+3} - h_{k+1} \bmod q$
0, 1, $\times$, $\times$	100	$e_{k+2} + h_{k+1} \bmod q$
0, -1, $\times$, $\times$	101	$e_{k+2} - h_{k+1} \bmod q$
1, $\times$, $\times$, $\times$	110	$e_{k+1} + h_{k+1} \bmod q$
-1, $\times$, $\times$, $\times$	111	$e_{k+1} - h_{k+1} \bmod q$

Algorithm 5.5 shows the algorithm for this arithmetic unit and Fig.5.10 shows the architecture for this arithmetic unit.

Algorithm 5.5 Proposed Algorithm for Arithmetic Unit

Input: $e_z = (e_{m-1}^{(z)} \dots e_0^{(z)})_2 (z = k+1 \dots k+4); h_{k+1} = (h_{m-1}^{(k+1)}, \dots, h_0^{(k+1)})_2; t_i = (t_2^{(i)} t_1^{(i)} t_0^{(i)})_2$

Output: $e_k = (e_{m-1}^{(k)}, \dots, e_0^{(k)})_2$;

if $t_2^{(i)} t_1^{(i)} = 00$ **then**
  if $t_0^{(i)} = 0$ **then**
    $(e_{m-1}^{(k)}, \dots, e_0^{(k)}) := (e_{m-1}^{(k+4)}, \dots, e_0^{(k+4)})$
  else
    $(e_{m-1}^{(k)}, \dots, e_0^{(k)}) := (e_{m-1}^{(k+3)}, \dots, e_0^{(k+3)})$
  end if
else if $t_0^{(i)} = 1$ **then**
  $(h_{m-1}^{(k+1)}, \dots, h_0^{(k+1)}) := -(h_{m-1}^{(k+1)}, \dots, h_0^{(k+1)})$
end if
if $t_2^{(i)} t_1^{(i)} = 01$ **then**
  $(e_{m-1}^{(k)}, \dots, e_0^{(k)}) := (e_{m-1}^{(k+3)}, \dots, e_0^{(k+3)}) + (h_{m-1}^{(k+1)}, \dots, h_0^{(k+1)})$
else if $t_2^{(i)} t_1^{(i)} = 11$ **then**
  $(e_{m-1}^{(k)}, \dots, e_0^{(k)}) := (e_{m-1}^{(k+1)}, \dots, e_0^{(k+1)}) + (h_{m-1}^{(k+1)}, \dots, h_0^{(k+1)})$
else
  $(e_{m-1}^{(k)}, \dots, e_0^{(k)}) := (e_{m-1}^{(k+2)}, \dots, e_0^{(k+2)}) + (h_{m-1}^{(k+1)}, \dots, h_0^{(k+1)})$
end if

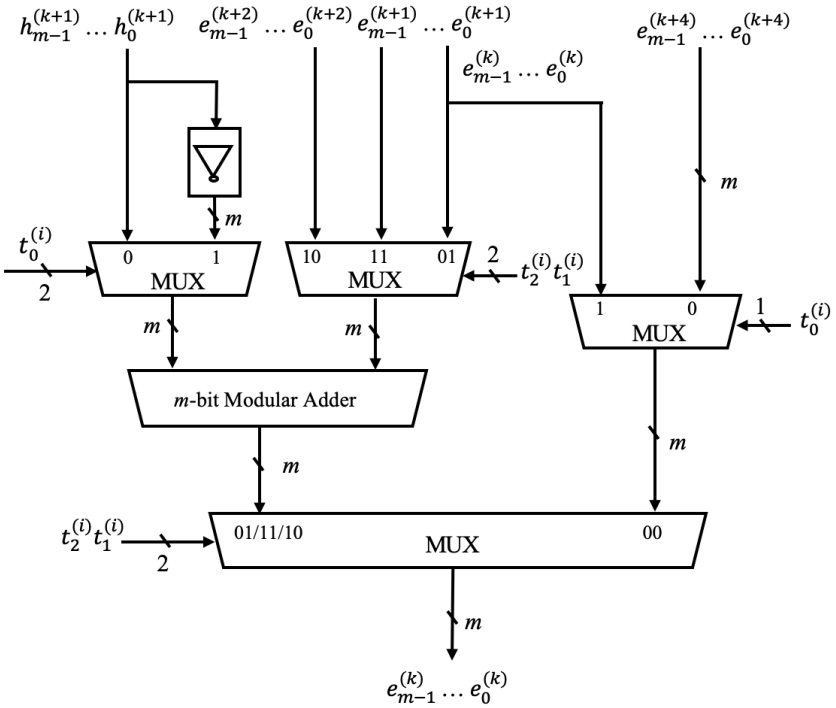


Fig. 5.10: Proposed Arithmetic Unit Architecture

5.3.3 TPM-IV for Encryption

A time-efficient architecture for NTRU Prime system based on an arithmetic unit is proposed. In this section, the control input $r(x) = (r_{n-1}, \ldots, r_0)$ is encoded aim to obtain $t(i) = (t_{u-1}, \ldots, t_0)$, each t_i has three bits.

Algorithm 5.6 shows the encryption part for proposed architecture. In addition, there are three inputs and one output in this architecture, which are public key $h(x)$, message $m(x)$, the generated sequence $(t_{u-1}, \ldots, t_0)$ and $e(x)$ with n coefficients.

Algorithm 5.6 Proposed Multiplication for NTRU Prime (TPM-IV)

Input: $m = m_{n-1}, \ldots, m_0; h = h_{n-1}, \ldots, h_0; (t_{u-1}, \ldots, t_0)$
Output: $e = e_{n-1}, \ldots, e_0 = hr + m \bmod q \bmod x^n - x - 1$

1: $e^{(0)} := m$
2: **for** $j := 1$ to u **do**
3: **for** $i := 0$ to $n-1$ **do**
4: **if** $t_{j-i} = 000$ **then**
5: $e_i^{(j)} := e_{i+4 \bmod n}^{(j-1)}$
6: **else if** $t_{j-i} = 001$ **then**
7: $e_i^{(j)} := e_{i+3 \bmod n}^{(j-1)}$
8: **else if** $t_{j-i} = 010$ **then**
9: $e_i^{(j)} := e_{i+3 \bmod n}^{(j-1)} + h_{i+1 \bmod n} \bmod q$
10: **else if** $t_{j-i} = 011$ **then**
11: $e_i^{(j)} := e_{i+3 \bmod n}^{(j-1)} - h_{i+1 \bmod n} \bmod q$
12: **else if** $t_{j-i} = 100$ **then**
13: $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} + h_{i+1 \bmod n} \bmod q$
14: **else if** $t_{j-i} = 101$ **then**
15: $e_i^{(j)} := e_{i+2 \bmod n}^{(j-1)} - h_{i+1 \bmod n} \bmod q$
16: **else if** $t_{j-i} = 110$ **then**
17: $e_i^{(j)} := e_{i+1 \bmod n}^{(j-1)} + h_{i+1 \bmod n} \bmod q$
18: **else**
19: $e_i^{(j)} := e_{i+1 \bmod n}^{(j-1)} - h_{i+1 \bmod n} \bmod q$
20: **end if**
21: **end for**
22: **end for**
23: **return** $e := e^{(u)}$

The multiplier contains n registers and n AUs. The registers $e = (e_{n-1}, \ldots, e_0)$ are initially loaded with $m = (m_{n-1}, \ldots, m_0)$. During one clock cycle, each t_i is scanned.

After u clock cycles, if the phase-shift value is zero, the encryption result will be stored in the registers $e = (e_{n-1}, \ldots, e_0)$. If the phase-shift value is one, the encryption result will be stored in the registers $e = (e_0, e_{n-1}, \ldots, e_1)$. If the phase-shift value is two, the encryption result will be stored in the registers $e = (e_1, e_0, e_{n-1}, \ldots, e_2)$. The corresponding architecture is shown in Fig.5.11.

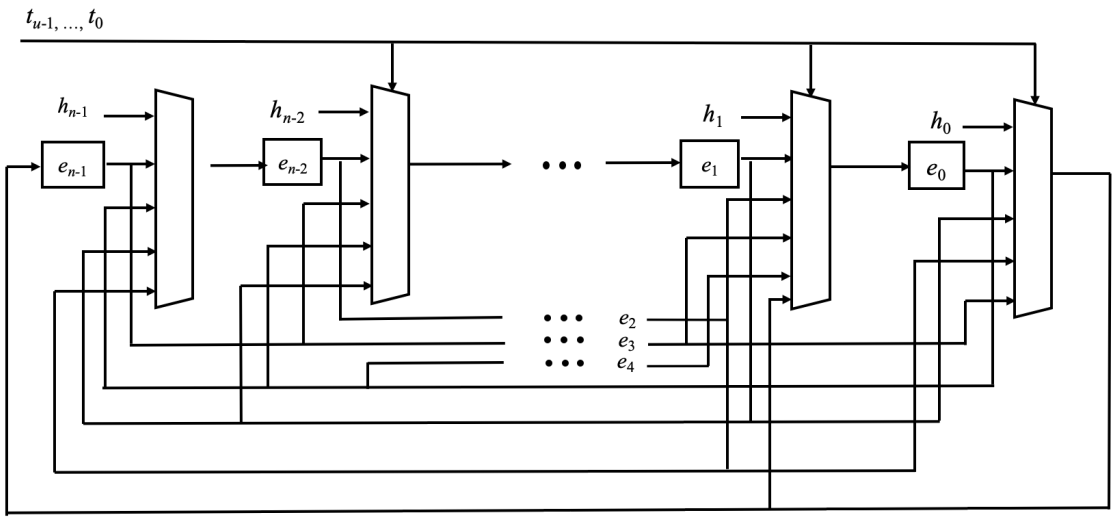


Fig. 5.11: TPM-IV Architecture

5.3.4 Implementation of TPM-IV

The following tools are used for our implementation.

- Quartus II v14.1 (64-bit) Software
- ModelSim-Altera Software

Arria V 5AGXFB3H4F35I3 was chosen as the target device in provided implementation and we use Verilog HDL as design language.

Implementation Results

The implementation results are shown in Table 5.9.

Table 5.9: FPGA Results for Different Parameter Sets

Security Level	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
112	*ees*401*ep*1	9032	8826	246	259.67 MHz	0.94 μs
	*ees*541*ep*1	12193	11906	196	248.14 MHz	0.78 μs
	*ees*659*ep*1	14880	14502	213	221.98 MHz	0.95 μs
128	*ees*449*ep*1	10155	9882	286	247.95 MHz	1.15 μs
	*ees*613*ep*1	13803	13490	222	253.87 MHz	0.87 μs
	*ees*761*ep*1	17130	16746	243	235.68 MHz	1.03 μs
192	*ees*677*ep*1	15343	14678	367	237.98 MHz	1.54 μs
	*ees*887*ep*1	20009	19518	323	228.05 MHz	1.41 μs
	*ees*1087*ep*1	24587	23918	350	223.41 MHz	1.56 μs
256	*ees*1087*ep*2	24587	23918	420	223.41 MHz	1.87 μs
	*ees*1171*ep*1	26393	25766	424	218.87 MHz	1.93 μs
	*ees*1499*ep*1	33820	32982	473	221.98 MHz	2.13 μs

5.4 FPGA Results Comparison

5.4.1 Comparison among Four Proposed works

In this section, TPM-I, II, III, and IV based on NTRU Prime are compared with each other. Please note that for each security level, only one parameter set is chosen for comparison.

The comparison results for security level 112 are shown in Table 5.10.

Table 5.10: Security Level 112

Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
TPM-I	*ees*401*ep*1	6817	8826	402	274.80 MHz	1.46 μs
TPM-II	*ees*401*ep*1	11904	8826	201	239.69 MHz	**0.83** μs
TPM-III	*ees*401*ep*1	6888	8826	381	303.12 MHz	1.25 μs
TPM-IV	*ees*401*ep*1	9032	8826	246	259.67 MHz	0.94 μs

It can be seen from the table that in security level 112, TPM-II has the lowest latency and it takes advantage of the smallest number of cycles. It saves 11.7% time compared to the second-best work, which is TPM-IV. It can be calculated by the subtraction of TPM-IV and TPM-II divided by TPM-IV. In addition, TPM-I uses

the least number of logical elements, which results in the smallest area consumption, although it has the largest latency among all the proposed works.

The comparison results for security level 128 are shown in Table 5.11.

Table 5.11: Security Level 128

Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
TPM-I	$ees449ep1$	7633	9882	450	280.35 MHz	1.60 μs
TPM-II	$ees449ep1$	13327	9882	225	256.21 MHz	**0.87** μs
TPM-III	$ees449ep1$	7731	9882	431	300.30 MHz	1.43 μs
TPM-IV	$ees449ep1$	10155	9882	286	247.95 MHz	1.15 μs

It can be seen from the table that in security level 128, TPM-II has the lowest latency and it uses the smallest number of cycles, which saves 24.3% time compared to the second-best work, which is TPM-IV. In addition, TPM-I takes advantage of the least number of logical elements, although it has the largest latency among all the proposed works.

The comparison results for security level 192 are shown in Table 5.12.

Table 5.12: Security Level 192

Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
TPM-I	$ees677ep1$	11509	14678	678	274.73 MHz	2.46 μs
TPM-II	$ees677ep1$	19737	14678	339	248.82 MHz	**1.36** μs
TPM-III	$ees677ep1$	11566	14678	620	267.52 MHz	2.32 μs
TPM-IV	$ees677ep1$	15343	14678	367	237.98 MHz	1.54 μs

It can be seen from the table that in security level 192, TPM-II has the lowest latency and it uses the smallest number of cycles and it saves 11.6% time compared to the second-best work, which is TPM-IV. In addition, TPM-I takes advantage of the least number of logical elements, although it has the largest latency among all the proposed works.

The comparison results for security level 256 are shown in Table 5.13.

Table 5.13: Security Level 256

Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
TPM-I	$ees1087ep2$	18479	23918	1088	284.26 MHz	3.82 μs
TPM-II	$ees1087ep2$	32241	23918	544	222.82 MHz	2.44 μs
TPM-III	$ees1087ep2$	18613	23918	872	257.53 MHz	3.38 μs
TPM-IV	$ees1087ep2$	24587	23918	420	223.41 MHz	**1.87** μs

It can be seen from the table that in security level 256, TPM-IV has the lowest latency and it uses the smallest number of cycles. We found that It saves 23.3% time compared to the second best work, which is TPM-II. It can be calculated by the subtraction TPM-II and TPM-IV divided by TPM-II. In addition, TPM-I takes advantage of the least number of logical elements, which includes the number of logical gates and registers, although it takes the largest latency among all the proposed works.

5.4.2 Compare Proposed works with NTRUEncrypt

In this section, the FPGA results of proposed works are compared to those of existing works on NTRUEncrypt. For each security level, we choose one parameter set for comparison.

The comparison results for security level 112 are shown in Table 5.14.

Table 5.14: Security Level 112

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	$ees401ep1$	4052	9638	401	62.95 MHz	6.37 μs
NTRU	[10]	$ees401ep1$	837	1165	3617	67.69 MHz	53.43 μs
NTRU	[11]	$ees401ep1$	15662	8838	227	55.00 MHz	4.13 μs
NTRU	[13]	$ees401ep1$	4636	8826	401	121.62 MHz	3.30 μs
NTRU	[15]	$ees401ep1$	9044	8826	349	113.67 MHz	3.07 μs
NTRU	[12]	$ees401ep1$	11871	8826	201	236.69 MHz	0.84 μs
NTRU Prime	TPM-II	$ees401ep1$	11904	8826	201	239.69 MHz	0.83 μs
NTRU Prime	TPM-I	$ees401ep1$	6817	8826	402	274.80 MHz	1.46 μs

For parameter set $ees401ep1$ in security level 112, it can be seen that the number

of clock cycles of TPM-II and [12] are the least, compared to all the existing works based on NTRUEncrypt. In addition, TPM-II for NTRU Prime system saves 1.19% of latency time, which can be calculated by the subtraction of [12] and TPM-II divided by [12] while using the same number of registers and a slightly higher number of ALMs, compared to the fastest existing work for NTRUEncrypt system. Moreover, although TPM-II and [12] performs better than TPM-I in terms of latency, TPM-I still takes advantage of the least number of ALM and registers.

The comparison results for security level 128 are shown in Table 5.15.

Table 5.15: Security Level 128

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*449*ep*1	4523	10793	449	56.99 MHz	7.88 μs
NTRU	[10]	*ees*449*ep*1	837	1165	4049	67.69 MHz	59.82 μs
NTRU	[11]	*ees*449*ep*1	17527	9884	269	53.95 MHz	4.99 μs
NTRU	[13]	*ees*449*ep*1	5188	9882	449	121.69 MHz	3.69 μs
NTRU	[15]	*ees*449*ep*1	10124	9882	398	112.90 MHz	3.53 μs
NTRU	[12]	*ees*449*ep*1	13301	9882	225	244.44 MHz	0.92 μs
NTRU Prime	TPM-II	*ees*449*ep*1	13327	9882	225	256.21 MHz	0.87 μs
NTRU Prime	TPM-I	*ees*449*ep*1	7633	9882	450	280.35 MHz	1.60 μs

It can be seen that the number of clock cycles of TPM-II and [12] are the least, compared to all the existing works on NTRUEncrypt. In addition, TPM-II based on NTRU Prime system has the smallest latency, it can save 5.4% of latency time and uses about the same number of registers and ALMs, compared to the fastest existing work for NTRUEncrypt system with the parameter set *ees*449*ep*1 in security level 128. In addition, although TPM-II and [12] performs better than TPM-I in terms of latency, TPM-I still uses the least number of logical elements, which includes ALMs and registers.

The comparison results for security level 192 are shown in Table 5.16.

Table 5.16: Security Level 192

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*677*ep*1	6740	16266	677	60.24 MHz	11.24 μs
NTRU	[10]	*ees*677*ep*1	837	1165	9486	67.69 MHz	140.14 μs
NTRU	[11]	*ees*677*ep*1	26423	14900	269	49.74 MHz	6.37 μs
NTRU	[13]	*ees*677*ep*1	7810	14898	677	120.00 MHz	5.64 μs
NTRU	[15]	*ees*677*ep*1	15254	14898	551	106.30 MHz	5.18 μs
NTRU	[12]	*ees*677*ep*1	13301	14898	339	239.12 MHz	1.41 μs
NTRU Prime	TPM-II	*ees*677*ep*1	19737	14678	339	248.82 MHz	1.36 μs
NTRU Prime	TPM-I	*ees*677*ep*1	11509	14678	678	274.73 MHz	2.46 μs

It can be seen that the number of clock cycles of TPM-II and [12] are the second least, compared to all the existing works based on NTRUEncrypt system. In addition, TPM-II for NTRU Prime system has the smallest latency, it can save 3.5% of latency time, while uses fewer registers and more ALMs, compared to the fastest existing work on NTRUEncrypt system with the parameter set *ees*677*ep*1 in security level 192. In addition, TPM-I uses the least number of ALMs and registers, although TPM-II and [12] performs better than TPM-I in terms of latency.

The comparison results for security level 256 are shown in Table 5.17.

Table 5.17: Security Level 256

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*1087*ep*2	10748	26105	1087	55.53 MHz	19.85 μs
NTRU	[10]	*ees*1087*ep*2	837	1165	23922	67.69 MHz	353.41 μs
NTRU	[11]	*ees*1087*ep*2	42427	23930	276	45.75 MHz	6.03 μs
NTRU	[13]	*ees*1087*ep*2	12526	23918	1087	104.06 MHz	10.45 μs
NTRU	[15]	*ees*1087*ep*2	24480	23918	717	94.07 MHz	7.62 μs
NTRU	[12]	*ees*1087*ep*2	32236	23918	544	228.15 MHz	2.38 μs
NTRU Prime	TPM-IV	*ees*1087*ep*2	24587	23918	420	223.41 MHz	1.87 μs
NTRU Prime	TPM-I	*ees*1087*ep*2	18479	23918	1088	284.26 MHz	3.82 μs

It can be seen that TPM-IV on NTRU Prime system has the smallest latency,

which can save 21.4% of latency time, compared to the fastest existing work on NTRUEncrypt system [12] with the parameter set $ees1087ep2$ in security level 256. The latency time can be calculated by the subtraction of [12] and TPM-IV divided by [12]. In addition, TPM-I uses the least number of ALMs and registers, although TPM-IV and [12] performs better than TPM-I in terms of latency. Note that TPM-II also uses much less ALMs compared to [12].

6 CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

NTRU Prime system and NTRUEncrypt system are probably the two most promising and actively researched ones among existing post-quantum cryptosystems. In this thesis, several time-efficient multiplication architectures are proposed for NTRU Prime system. Their FPGA implementations are also presented. To the best of our knowledge, this is the first time that NTRU Prime system has been implemented in FPGA.

TPM-I is architecture on NTRU Prime system and a new arithmetic unit is also presented. The new proposed architecture takes advantage of LFSR based structure for its compact circuitry. Our FPGA implementation has shown that the TPM-I based on NTRU Prime makes use of the least number of logical elements, including logical gates and registers, which means it consumes the smallest area among the other three proposed works.

Then, TPM-II aims to implement NTRU Prime based system, which takes advantage of x^2-net architecture. Multiplier scans two consecutive coefficients in the same control input polynomial $r(x)$ in one clock cycle and the number of clock cycles is half in TPM-II. The FPGA simulation results show that TPM-II has the best performance in terms of latency, compared to the other three proposed works. The latency time of TPM-II can be saved 11.7%, 24.3% and 11.6%, compare to the second lowest latency work, TPM-IV, for different parameter sets in different security levels: $ees401ep1$ in 112-bit, $ees449ep1$ in 128-bit, $ees677ep1$ in 192-bit.

Next, TPM-III based on NTRU Prime has been presented, which takes advantage of three consecutive zeros in polynomial coefficients, then recode the polynomial $r(x)$. Then, TPM-IV is obtained, which uses consecutive zeros in polynomial coefficients, then recode the polynomial $r(x)$. The FPGA implementation results show that with

the parameter set $ees1087ep2$ at security level 256-bit, TPM-IV has the lowest latency compare to the second best latency work, TPM-II.

Finally, comparing the proposed work with the best of existing work on NTRUEncrypt system. NTRU Prime system is different from NTRUEncrypt system in that they use different polynomial rings. With a slightly more complicated truncated polynomial ring, NTRU Prime system can increase the security strength. Moreover. the proposed TPM-II can save 1.19%, 5.4%, and 3.5% of time, compared to the best existing work published for NTRUEncrypt system in security level 112, 128 and 192, respectively. TPM-IV can save 21.4% latency time in security level 256. In addition, TPM-I uses the least area consumption, including the number of ALMs and registers, compared to the best NTRUEncrypt work.

A comparison list for all IEEE recommended parameter sets is given in Appendix A.

6.2 Future Works

Based on the research works on NTRU Prime system presented in this thesis, possible future work can be listed as follow:

- There are now both software and FPGA implementations available for NTRU Prime cryptosystem. It is expected that the implementation of this promising cryptosystem can be extended into ASIC design.

- There is one other lattice-based system, LPRime, and only its software realization is available in the literature. It should be interesting to see its FPGA implementation.

- There are not many published software implementation of other types of post-quantum cryptosystem, such like hash-based cryptography, code-based cryptography. etc. It is expected more research works be done on these systems in terms of architectures and hardware implementations.

REFERENCES

[1] *IEEE standard specification for public-key cryptographic techniques based on hard problems over lattices*, IEEE p1363.1, 2009.

[2] D. Cabarcas, P. Weiden, and J. Buchmann, "On the efficiency of provably secure NTRU," in *International Workshop on Post-Quantum Cryptography*. Springer International Publishing, 2014, pp. 22–39.

[3] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte, "Choosing NTRUEncrypt parameters in light of combined lattice reduction and mitm approaches," in *International Conference on Applied Cryptography and Network Security*. Springer Berlin Heidelberg, 2009, pp. 437–455.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999.

[5] D. J. Bernstein, J. Buchmann, and E. Dahmen, *Post-quantum cryptography*. Springer Science & Business Media, 2009.

[6] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*. Springer Berlin Heidelberg, 1998, pp. 267–288.

[7] D. Stehle and R. Steinfeld, "Making NTRU as secure as worst-case problems over ideal lattices," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer Berlin Heidelberg, 2011, pp. 27–47.

[8] D. J. Bernstein, C. Chuengsatiansup, T. Lange and C. V. Vredendaal, "NTRU Prime: reducing attack surface at low cost," in *International Conference on Selected Areas in Cryptography*. Springer, 2017, pp. 235-260.

[9] D. V. Bailey, D. Coffin, A. Elbirt, J. H. Silverman, and A. D. Woodbury, "NTRU in constrained devices," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer Berlin Heidelberg, 2001, pp. 262–272.

[10] C. M. O Rourke, "Efficient NTRU implementations," Master's thesis, Worcester Polytechnic Institute, 2002.

[11] A. A. Kamal and A. M. Youssef, "An fpga implementation of the NTRUEncrypt cryptosystem," in *2009 International Conference on Microelectronics-ICM*. IEEE, 2009, pp. 209–212.

[12] Ruiqing. Dong, "Efficient multiplication architectures for truncated polynomial ring," Master's thesis, University of Windsor, 2015.

[13] B. Liu, "Efficient architecture and implementation for NTRU based systems," Master's thesis, University of Windsor, 2015.

[14] B. Liu and H. Wu, "Efficient architecture and implementation for NTRUEncrypt system," in *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2015, pp. 1–4.

[15] T. Fritzmann, T. Schamberger, C. Frisch, K. Braun, G. Maringer, J. Sepúlveda, "Efficient hardware/software co-design for NTRU," in *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*. Springer, 2019, pp. 257-280.

[16] Dmitriy Rumynin, "Algebra-2: groups and rings," Lecture notes, University of Warwick, 2011.

[17] Math 152, "The very basics of groups, rings, and fields," in *http://www-users.math.umn.edu/ brubaker/docs/152/152groups.pdf*, 2006.

[18] Marotzke, Adrian, "Constant time full hardware implementation of Streamlined NTRU Prime," in *Published 2020, Computer Science, IACR Cryptol, ePrint Arch.* CARDIS, 2020.

[19] H Wu, "Linear Feedback Shift Register" Lecture notes, University of Windsor, 2019.

[20] C. Gentry et al., "Fully homomorphic encryption using ideal lattices," *STOC*, vol. 9, pp. 169–178, 2009.

[21] K. Rohloff, D. B. Cousins, "A scalable implementation of fully homomorphic encryption built on NTRU," in *Financial Cryptography and Data Security.* Springer, 2014, pp. 221-234.

[22] M. R. Albrecht, B. R. Curtis, "Estimate all the LWE, NTRU schemes," in *Security and Cryptography for Networks.* Springer, 2018, pp. 351-367.

[23] R. Gauravaram, H. Narumanchi, N. Emmadi, "Analytical study of implementation issues of NTRU," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI).* IEEE, 2014, pp. 700-707.

[24] Alagic, Gorjan, et al., "Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process," in *US Department of Commerce, National Institute of Standards and Technology*, 2019.

[25] A. Hulsing, J. Rijneveld, J. M. Schanck, P. Schwabe, "NTRU-HRSS-KEM Algorithm Specifications And Supporting Documentation," in *NIST submission 30*, 2017.

[26] A. Lopez-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings*

of the forty-fourth annual ACM symposium on Theory of computing. ACM, 2012, pp. 1219–1234.

[27] Q. Zeng, "An efficient architecture and fpga implementation of fully homomorphic encryption with cloud computing significance," Master's thesis, University of Windsor, 2018.

[28] D. Wichs, "CCA2 security for the cramer-shoup cryptosystem," in *https://www.ccs.neu.edu/home/wichs/class/crypto-fall17/lecture21.pdf*, 2017.

[29] Bernstein, Chuengsatiansup, Lange, V.Vredendaal, "NTRU Prime," *IACR Cryptol. ePrint Arch.*, vol. 2016, pp. 461, 2016.

[30] Cavallaro, Lorenzo, Gollmann and Dieter, "Information security theory and practice," *Security of Mobile and Cyber-Physical Systems*, vol. 7886, 2013.

[31] D. Slamanig, "Modern cryptography- public key encryption," Lecture notes, Austrian Institute of Technology, 2019.

[32] H. Cheng, D. Dinu, J. Groschadl, P. B. Ronne and P. Ryan, "A lightweight implementation of NTRU Prime for the post-quantum internet of things," in *IFIP International Conference on Information Security Theory and Practice*. Springer, 2020, pp. 103-119.

[33] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer Berlin Heidelberg, 1984, pp. 10–18.

[34] J. He, "Encryption and signature algorithms from NTRU," *Journal of Cyber Security*. vol. 41, no. 2, pp. 10–18, 2019.

[35] J. Buchmann, M. Doring, R. Lindner, "Efficiency Improvement for NTRU," in *German Ministry for Education and Research (BMBF)*. Gesellschaft fur Informatik e. V, 2008, pp. 164–178.

[36] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *Applied Cryptography and Network Security*. Springer, 2005, pp. 164–175.

APPENDIX A

Here shows existing works in order to compare results among our proposed works.

Table A.1: FPGA Results for $ees401ep1$, Security Level 112-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	$ees401ep1$	4052	9638	401	62.95 MHz	6.37 μs
NTRU	[10]	$ees401ep1$	837	1165	3617	67.69 MHz	53.43 μs
NTRU	[11]	$ees401ep1$	15662	8838	227	55.00 MHz	4.13 μs
NTRU	[13]	$ees401ep1$	4636	8826	401	121.62 MHz	3.30 μs
NTRU	[15]	$ees401ep1$	9044	8826	349	113.67 MHz	3.07 μs
NTRU	[12]	$ees401ep1$	11861	8826	201	103.01 MHz	1.95 μs
NTRU Prime	TPM-I	$ees401ep1$	6817	8826	402	274.80 MHz	1.46 μs
NTRU Prime	TPM-II	$ees401ep1$	11904	8826	201	239.69 MHz	0.83 μs
NTRU Prime	TPM-III	$ees401ep1$	6888	8826	381	303.12 MHz	1.25 μs
NTRU Prime	TPM-IV	$ees401ep1$	9032	8826	246	259.67 MHz	0.94 μs

Table A.2: FPGA Results for $ees541ep1$, Security Level 112-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	$ees541ep1$	5425	12995	541	61.50 MHz	8.80 μs
NTRU	[10]	$ees541ep1$	837	1165	5959	67.69 MHz	88.03 μs
NTRU	[11]	$ees541ep1$	21124	11918	121	52.87 MHz	2.29 μs
NTRU	[13]	$ees541ep1$	6246	11906	541	122.02 MHz	4.43 μs
NTRU	[15]	$ees541ep1$	12194	11906	349	116.14 MHz	2.94 μs
NTRU	[12]	$ees541ep1$	16091	11906	271	102.33 MHz	2.65 μs
NTRU Prime	TPM I	$ees541ep1$	9197	11906	542	283.93 MHz	1.91 μs
NTRU Prime	TPM-II	$ees541ep1$	16025	11906	271	240.91 MHz	1.12 μs
NTRU Prime	TPM-III	$ees541ep1$	9256	11906	422	304.41 MHz	1.38 μs
NTRU Prime	TPM-IV	$ees541ep1$	12193	11906	196	248.14 MHz	0.78 μs

Table A.3: FPGA Results for $ees659ep1$, Security Level 112-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	$ees659ep1$	6572	15832	659	59.43 MHz	11.09 μs
NTRU	[10]	$ees659ep1$	837	1165	9234	67.69 MHz	136.42 μs
NTRU	[11]	$ees659ep1$	25726	14514	107	51.08 MHz	2.09 μs
NTRU	[13]	$ees659ep1$	7603	14502	659	114.53 MHz	5.75 μs
NTRU	[15]	$ees659ep1$	14850	14502	386	106.92 MHz	3.61 μs
NTRU	[12]	$ees659ep1$	19573	14502	330	105.96 MHz	3.11 μs
NTRU Prime	TPM-I	$ees659ep1$	11203	14502	660	282.17 MHz	2.33 μs
NTRU Prime	TPM-II	$ees659ep1$	19539	14502	330	242.19 MHz	1.36 μs
NTRU Prime	TPM-III	$ees659ep1$	11267	14502	489	284.74 MHz	1.71 μs
NTRU Prime	TPM-IV	$ees659ep1$	14880	14502	213	221.98 MHz	0.95 μs

Table A.4: FPGA Results for *ees*449*ep*1, Security Level 128-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*449*ep*1	4523	10793	449	56.99 MHz	7.88 μs
NTRU	[10]	*ees*449*ep*1	837	1165	4049	67.69 MHz	59.82 μs
NTRU	[11]	*ees*449*ep*1	17527	9884	269	53.95 MHz	4.99 μs
NTRU	[13]	*ees*449*ep*1	5188	9882	449	121.69 MHz	3.69 μs
NTRU	[15]	*ees*449*ep*1	10124	9882	398	112.90 MHz	3.53 μs
NTRU	[12]	*ees*449*ep*1	13296	9882	225	105.24 MHz	2.14 μs
NTRU Prime	TPM-I	*ees*449*ep*1	7633	9882	450	280.35 MHz	1.60 μs
NTRU Prime	TPM-II	*ees*449*ep*1	13327	9882	225	256.21 MHz	0.87 μs
NTRU Prime	TPM-III	*ees*449*ep*1	7731	9882	431	300.30 MHz	1.43 μs
NTRU Prime	TPM-IV	*ees*449*ep*1	10155	9882	286	247.95 MHz	1.15 μs

Table A.5: FPGA Results for *ees*613*ep*1, Security Level 128-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*613*ep*1	6124	14730	613	61.62 MHz	9.95 μs
NTRU	[10]	*ees*613*ep*1	837	1165	7977	67.69 MHz	117.85 μs
NTRU	[11]	*ees*613*ep*1	23931	13502	135	50.92 MHz	2.65 μs
NTRU	[13]	*ees*613*ep*1	7075	13490	613	115.12 MHz	5.32 μs
NTRU	[15]	*ees*613*ep*1	13814	13490	387	111.08 MHz	3.48 μs
NTRU	[12]	*ees*613*ep*1	18201	13490	307	104.60 MHz	2.93 μs
NTRU Prime	TPM-I	*ees*613*ep*1	10421	13490	614	284.50 MHz	2.15 μs
NTRU Prime	TPM-II	*ees*613*ep*1	18170	13490	307	229.31 MHz	1.33 μs
NTRU Prime	TPM-III	*ees*613*ep*1	10508	13490	478	289.86 MHz	1.65 μs
NTRU Prime	TPM-IV	*ees*613*ep*1	13803	13490	222	253.87 MHz	0.87 μs

Table A.6: FPGA Results for *ees*761*ep*1, Security Level 128-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*761*ep*1	7571	18282	761	59.53 MHz	12.78 μs
NTRU	[10]	*ees*761*ep*1	837	1165	12184	67.69 MHz	180.0 μs
NTRU	[11]	*ees*761*ep*1	29707	16758	120	49.61 MHz	2.42 μs
NTRU	[13]	*ees*761*ep*1	8776	16746	761	110.92 MHz	6.86 μs
NTRU	[15]	*ees*761*ep*1	17144	16746	443	109.14 MHz	4.06 μs
NTRU	[12]	*ees*761*ep*1	22459	16746	381	103.33 MHz	3.69 μs
NTRU Prime	TPM-I	*ees*761*ep*1	12937	16746	762	281.85 MHz	2.70 μs
NTRU Prime	TPM-II	*ees*761*ep*1	22558	16746	381	234.58 MHz	1.62 μs
NTRU Prime	TPM-III	*ees*761*ep*1	13023	16746	562	271.74 MHz	2.07 μs
NTRU Prime	TPM-IV	*ees*761*ep*1	17130	16746	243	235.68 MHz	1.03 μs

Table A.7: FPGA Results for *ees*677*ep*1, Security Level 192-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*677*ep*1	6740	16266	677	60.24 MHz	11.24 μs
NTRU	[10]	*ees*677*ep*1	837	1165	9486	67.69 MHz	140.14 μs
NTRU	[11]	*ees*677*ep*1	26423	14900	269	49.74 MHz	6.37 μs
NTRU	[13]	*ees*677*ep*1	7810	14898	677	120.00 MHz	5.64 μs
NTRU	[15]	*ees*677*ep*1	15254	14898	551	106.30 MHz	5.18 μs
NTRU	[12]	*ees*677*ep*1	20042	14898	339	106.90 MHz	3.17 μs
NTRU Prime	TPM-I	*ees*677*ep*1	11509	14678	678	274.73 MHz	2.46 μs
NTRU Prime	TPM-II	*ees*677*ep*1	19737	14678	339	248.82 MHz	1.36 μs
NTRU Prime	TPM-III	*ees*677*ep*1	11566	14678	620	267.52 MHz	2.32 μs
NTRU Prime	TPM-IV	*ees*677*ep*1	15343	14678	367	237.98 MHz	1.54 μs

Table A.8: FPGA Results for *ees*887*ep*1, Security Level 192-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*887*ep*1	8788	21305	887	60.12 MHz	14.75 μs
NTRU	[10]	*ees*887*ep*1	837	1165	15974	67.69 MHz	235.99 μs
NTRU	[11]	*ees*887*ep*1	34622	19503	198	47.70 MHz	4.15 μs
NTRU	[13]	*ees*887*ep*1	10226	19518	887	106.24 MHz	8.35 μs
NTRU	[15]	*ees*887*ep*1	19980	19518	562	99.98 MHz	5.62 μs
NTRU	[12]	*ees*887*ep*1	26295	19518	444	103.68 MHz	4.28 μs
NTRU Prime	TPM-I	*ees*887*ep*1	15079	19518	888	263.99 MHz	3.36 μs
NTRU Prime	TPM-II	*ees*887*ep*1	26353	19518	444	239.01 MHz	1.85 μs
NTRU Prime	TPM-III	*ees*887*ep*1	15172	19518	693	264.27 MHz	2.62 μs
NTRU Prime	TPM-IV	*ees*887*ep*1	20009	19518	323	228.05 MHz	1.41 μs

Table A.9: FPGA Results for *ees*1087*ep*1, Security Level 192-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*1087*ep*1	10748	26105	1087	55.53 MHz	19.58 μs
NTRU	[10]	*ees*1087*ep*1	837	1165	23922	67.69 MHz	353.41 μs
NTRU	[11]	*ees*1087*ep*1	42427	23930	177	47.19 MHz	3.75 μs
NTRU	[13]	*ees*1087*ep*1	12526	23918	1087	104.06 MHz	10.45 μs
NTRU	[15]	*ees*1087*ep*1	24480	23918	637	94.07 MHz	6.77 μs
NTRU	[12]	*ees*1087*ep*1	32241	23918	544	96.43 MHz	5.64 μs
NTRU Prime	TPM-I	*ees*1087*ep*1	18479	23918	1088	284.26 MHz	3.82 μs
NTRU Prime	TPM-II	*ees*1087*ep*1	32241	23918	544	222.82 MHz	2.44 μs
NTRU Prime	TPM-III	*ees*1087*ep*1	18613	23918	806	257.53 MHz	3.13 μs
NTRU Prime	TPM-IV	*ees*1087*ep*1	24587	23918	350	223.41 MHz	1.56 μs

Table A.10: FPGA Results for *ees*1087*ep*2, Security Level 256-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*1087*ep*2	10748	26105	1087	55.53 MHz	19.58 μs
NTRU	[10]	*ees*1087*ep*2	837	1165	23922	67.69 MHz	353.41 μs
NTRU	[11]	*ees*1087*ep*2	42427	23930	276	45.75 MHz	6.03 μs
NTRU	[13]	*ees*1087*ep*2	12526	23918	1087	104.06 MHz	10.45 μs
NTRU	[15]	*ees*1087*ep*2	24480	23918	717	94.07 MHz	7.62 μs
NTRU	[12]	*ees*1087*ep*2	32241	23918	544	96.43 MHz	5.64 μs
NTRU Prime	TPM-I	*ees*1087*ep*2	18479	23918	1088	284.26 MHz	3.82 μs
NTRU Prime	TPM-II	*ees*1087*ep*2	32241	23918	544	222.82 MHz	2.44 μs
NTRU Prime	TPM-III	*ees*1087*ep*2	18613	23918	872	257.53 MHz	3.38 μs
NTRU Prime	TPM-IV	*ees*1087*ep*2	24587	23918	420	223.41 MHz	1.87 μs

Table A.11: FPGA Results for *ees*1171*ep*1, Security Level 256-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	*ees*1171*ep*1	11568	28121	1171	59.28 MHz	19.75 μs
NTRU	[10]	*ees*1171*ep*1	837	1165	28112	67.69 MHz	415.31 μs
NTRU	[11]	*ees*1171*ep*1	45703	25778	261	44.60 MHz	5.85 μs
NTRU	[13]	*ees*1171*ep*1	13491	25766	1171	108.31 MHz	10.81 μs
NTRU	[15]	*ees*1171*ep*1	26370	25766	740	93.48 MHz	7.92 μs
NTRU	[12]	*ees*1171*ep*1	34648	25766	586	99.50 MHz	5.89 μs
NTRU Prime	TPM-I	*ees*1171*ep*1	19907	25766	1172	241.20 MHz	4.85 μs
NTRU Prime	TPM-II	*ees*1171*ep*1	34848	25766	586	220.70 MHz	2.65 μs
NTRU Prime	TPM-III	*ees*1171*ep*1	19972	25766	913	232.94 MHz	3.92 μs
NTRU Prime	TPM-IV	*ees*1171*ep*1	26393	25766	424	218.87 MHz	1.93 μs

Table A.12: FPGA Results for $ees1499ep1$, Security Level 256-bit

Algorithm	Work	Parameter set	#ALM	#Register	#Cycles	FMax	Latency
NTRU	[9]	$ees1499ep1$	14755	35989	1499	51.87 MHz	28.90 μs
NTRU	[10]	$ees1499ep1$	837	1165	44978	67.69 MHz	664.47 μs
NTRU	[11]	$ees1499ep1$	58507	32994	228	42.95 MHz	5.31 μs
NTRU	[13]	$ees1499ep1$	17263	32982	1499	103.68 MHz	14.46 μs
NTRU	[15]	$ees1499ep1$	33750	32982	866	96.99 MHz	8.93 μs
NTRU	[12]	$ees1499ep1$	48719	32982	750	82.73 MHz	9.07 μs
NTRU Prime	TPM-I	$ees1499ep1$	25483	32982	1500	225.99 MHz	6.63 μs
NTRU Prime	TPM-II	$ees1499ep1$	48723	32982	750	209.86 MHz	3.57 μs
NTRU Prime	TPM-III	$ees1499ep1$	25595	32982	1102	245.58 MHz	4.48 μs
NTRU Prime	TPM-IV	$ees1499ep1$	33820	32982	473	221.98 MHz	2.13 μs

VITA AUCTORIS

NAME:	Xi Gao
PLACE OF BIRTH:	Daqing, China
YEAR OF BIRTH:	1996
EDUCATION:	Suihua University, Heilongjiang, China Bachelor of Engineering, Electrical & Engineering 2014-2018 University of Windsor, Windsor, ON, Canada Master of Applied Science, Electrical and Computer Engineering 2018-2021