Hardware Implementation of Modular Arithmetic and Group Operation Over Prime Field for Elliptic Curve Cryptography Processor

Submitted By

Sakib Absar

Roll No.:1403078

Supervised By

Dr. Md. Selim Hossain

Associate Professor

Dept. of Electrical and Electronic Engineering

A thesis submitted in partial fulfillment for the requirements of the degree of

Bachelor of Science in Electrical & Electronic Engineering

February 2019

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Khulna-9203, Bangladesh

Declaration

I certify that the thesis work entitled **"Hardware Implementation of Modular Arithmetic and Group Operation Over Prime Field for Elliptic Curve Cryptography Processor"** has not been submitted anywhere for the part of the requirements for any degree or diploma.

I also certify that the thesis has been carried out by me, under the supervision of Dr. Md. Selim Hossain in the department of Electrical & Electronic Engineering at Khulna University of Engineering & Technology, Khulna-9203, Bangladesh.

In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Supervisor

Dr. Md. Selim Hossain

Associate Professor

Dept. of Electrical & Electronic Engineering

Khulna University of Engineering & Technology

Khulna-9203.

Signature of Student

Sakib Absar

Roll: 1403078

Acknowledgements

First of all express immense gratefulness and humbleness to almighty, omnipotent ALLAH for the successful completion of my thesis. I feel fortunate to be at Khulna University of Engineering & Technology (KUET), Khulna, Bangladesh to pursue my B. Sc. in a wonderful research environment.

I am extremely grateful and thankful to my honorable supervisor Dr. Md. Selim Hossain, Associate Professor, Department of EEE, KUET, for providing me inspiration and such a supportive research environment. The completion of this thesis would not have possible without his constructive guidance, constant inspiration and encouragement throughout the research work.

Then I would like to express my gratitude and deepest appreciation to Prof. Dr. Md. Rafiqul Islam, Head of the Department of EEE, KUET and all the faculty members of the department of EEE, KUET for helping in various ways in formulating this thesis.

I have to thank my family for being the ultimate source of strength and encouragement throughout my academic carrier and personal life.

February 2019 Author

ABSTRACT

The need for secure communication over the network has increased very quickly over the recent years and Elliptic Curve Cryptography (ECC) which is a Public-key Cryptography (PKC) plays an important role in moving secured information. In this work, a hardware implementation of modular arithmetic and group operation over the prime field for Elliptic Curve Cryptography Processor (ECP) for efficient security system has been described. These implementations are done based on the National Institute of Standards and Technology (NIST) recommended standard. Both modular arithmetic (addition, subtraction, multiplication/squaring) and group operations (point doubling and point addition) over the prime field of $\mathbb{F}_{\boldsymbol{P}}$-256 are optimized to improve the performance of Elliptic Curve Point Multiplication. The efficiency of elliptic curve cryptography processor depends a lot on the modular arithmetic. A high-performance modular addition, subtraction and multiplication for elliptic curves are implemented over a prime field using the NIST standard. A modular addition or subtraction operation takes only one clock cycle, and can speed up the overall calculation of point doubling and point addition (PDPA) and hence the ECP. Elliptic curve scalar (or point) multiplication (ECSM or ECPM) which is the most time and resource consuming operation, is developed for high efficiency performance. **.** An efficient modular multiplier is designed using interleaved modular multiplication method which requires 257 clock cycles. This modular multiplication needs 522 slices registers, 1489 slice LUTs and 514 LUT-FF pairs with a delay of 3.28 us. Separate point doubling (PD) and point addition (PA) architectures are implemented in Jacobian coordinate using efficient modular arithmetic to achieve high speed and low area consumption. This new architecture is simulated in Xilinx ISE 14.7. For the point doubling operation the number of required LUT-FF pair is 6568, number of required slice LUT is 20003 and number of required slice register is 6729 and the delay is 13.62 us. For the point addition operation the number of required LUT-FF pair is 9561, number of required slice LUT is 33297 and number of required slice register is 10141 and the delay is 20.43 us. After that, the architecture is implemented in the field-programmable gate array (FPGA) board. Proposed modular arithmetic and group operations can be utilized to design an Elliptic Curve Point Multiplication (ECPM).

Dedicated to

My beloved parents

And

Honorable Thesis Supervisor

Dr. Md. Selim Hossain

Table of Contents

List of Figures

List of Tables

Abbreviations

ECC	-	Elliptic Curve Cryptography
ECP	-	Elliptic Curve Cryptographic Processor
PKC	-	Public Key Cryptography
EC	-	Elliptic Curve
GF	-	Galois Field
FPGA	-	Field Programming Gate Array
ASIC	-	Application-Specific Integrated Circuit
ANSI	-	American National Standards Institute
AT	-	Area × Time
AES	-	Advance Encryption Standard
VLSI	-	Very Large Scale Integration
DES	-	Data Encryption Standard
3DES	-	Triple Data Encryption Standard
CC	-	Clock Cycle
DH	-	Diffie-Hellman
DLP	-	Discrete Logarithm Problem
DSP	-	Digital Signal Processing
ECPA	-	Elliptic Curve Point Addition
ECPD	-	Elliptic Curve Point Doubling
ECPM	-	Elliptic Curve Point Multiplication
ECSM	-	Elliptic Curve Scalar Multiplication
HDL	-	Hardware Descriptive Language

FFA	-	Finite Field Arithmetic
FFMA	-	Finite Field Modular Arithmetic
LUT	-	Look-Up Table
MUX	-	Multiplexer
NIST	-	National Institute of Standard and Technology
ONB	-	Optimal Normal Basis
PD	-	Point Doubling
PA	-	Point Addition
PDPA	-	Point Doubling and Point Addition
RSA	-	Rivest, Shamir and Adleman PKC Algorithm
SCA	-	Side Channel Attacks
SEC	-	Standards for Efficient Cryptography
VHDL	-	VHSIC Hardware Descriptive Language
VHSIC	-	Very High Speed Integrated Circuits

Chapter 1
Introduction

1.1 Overview

Data security is the protection of stored digital information to prevent unauthorized access to computers, personal database, and websites. Data security is critical for most businesses and even home computer users. Client information, payment information, personal files, bank account details - all of this information can be hard to replace and potentially dangerous if it falls into the wrong hands as losing it to hackers or a malware infection can have much greater consequences. Therefore the need for secure communications and transactions over the network has increased rapidly in recent times and efficient cryptography is needed for the growing amount of secured transactions over the network. Cryptography is a field of computer science and mathematics that focusses on techniques for secured communication between two parties while a third-party is present [1]. This is based on methods like encryption, decryption, signing, generating of pseudo-random numbers etc. Strong and efficient cryptography is necessary to ensure authentication, authorization, data confidentiality, and data integrity [2-5]. Cryptographic systems can be broadly divided into two kinds - private-key (or symmetric) cryptography and public-key (or asymmetric) cryptography (PKC) [4]. Private-key cryptography is computationally less expensive than PKC but it is inflexible with respect to key management because it requires pre-distribution of keys. On the other hand, PKC has a huge computational complexity but allows for flexible key management [2]. There are many types of public key cryptography and Elliptic Curve Cryptography is just one of them. Others algorithms include Rivest–Shamir–Adleman (RSA), Diffie-Helman, etc. ECC was first proposed in the mid-1980s by Koblitz [6] and Miller [7]. Other researchers have also implemented ECPs over prime fields. The security of ECC depends on the difficulty of the discrete logarithm problem. PKC of this types are based upon a one-way function; a function for which the output corresponding to a particular input is easy to calculate, but the input corresponding to a particular output is computationally infeasible to calculate. Different parameters can be selected for an ECC for the optimization of the performance. The National Institute of Standards and

Technology (NIST) recommends ten finite fields; five of which are for binary fields and other five are for prime fields, for using in the elliptic curve digital signature algorithm (ECDSA) [8]. IEEE [9] and ANSI [10] have presented the ECC parameters. ECC implementations have shown to be more efficient than other public-key cryptography algorithms. ECC can provide the same level of security as the other traditional cryptosystems with a shorter key. For example, a 256- bit ECC over a prime field provides the same level of security as a 3072-bit Rivest–Shamir–Adleman RSA [11]. It is also seen that, less memory and hardware resources are required to implement elliptic curve cryptosystems. This makes ECC very popular for devices such as smart cards, credit cards, pagers, personal digital assistants, and cellular phones.

1.2 Challenges and Objectives

The asymmetric nature of PKC allows it a sizable advantage over symmetric-key algorithms but asymmetric keys must be many times longer than keys in private cryptography in order to boost equivalent security. As PKC algorithms (e.g. ECC) are based on complicated mathematics, the computational complexity is very high. For applications with large quantities of encrypted data, the public key systems can be very slow because of computational overhead. So, the throughput rate is reduced and it gets computationally costly. Therefore implementation becomes difficult. However, a high-throughput elliptic curve cryptosystem, low computational cost and energy efficiency during computation are mandatory for modern applications as area, time and power are the three most important performance parameter for cryptographic processors. For increasing the speed of the processor, the subsequent time (cycle count) for multiplication has to be reduced. But for the reduction of time, the number of necessary logic gates has to be increased. A high-speed, low-area, and low-power-consumption ECP is suitable for practical cryptographic applications. So, there is a trade-off between time, area, and power in designing a cryptographic processor.

Cryptosystems can be implemented with either a hardware or software approach. In this research, the hardware approach is considered because software implementations are usually very slow or much too power consuming.

The efficiency of modular arithmetic, e.g. modular addition, subtraction, multiplication, squaring, and inversion has a huge impact when cryptosystem performs a huge number of encryption and decryption processes because a modular arithmetic unit is mandatory for elliptic curve group operations and elliptic curve point multiplication (ECPM), where ECPM is the core operation of an ECC processor (ECP) [2]. In this literature, the implementation of efficient modular arithmetic is done in the prime field. To date, several modular multiplication methods have been proposed such as Montgomery modular multiplication, Radix-4 Montgomery modular multiplication, Bipartite modular multiplication, Sum of Residues modular multiplication, Modular multiplication using core function etc. In this literature, Interleaved modular multiplication method has been used because of its efficiency and simplicity.

Elliptic curve group operation is required to design a full hierarchy of EPCM. Separate group operations such as point doubling (PDBL) and point addition (PADD) are proposed based on the prime field. Pre-computations and parallelization are used during group operations to increase the overall efficiency of the system.

In this literature, the prime focus is to explore high-performance implementations of ECP in hardware and evaluate their energy efficiency and performance. For doing this, several hardware architectures of modular arithmetic and group operations are designed and optimized in the prime field. Hardware implementations of ECP on a field-programmable gate array (FPGA) is the focus of this thesis. FPGA implementations are suitable for prototype design and offer higher flexibility or reprogrammable hardware design of ECP. Detailed performance analyses of the proposed architectures based on FPGA are discussed thoroughly in this thesis.

Chapter 2
Background

This chapter provides background knowledge regarding cryptography, finite field operations, and elliptic curve cryptography (ECC). Classification of cryptography is also described in this chapter. This chapter also introduces group operations, point doubling (PDBL) and point addition (PADD) for elliptic curve point multiplication (ECPM). The mathematics and the hardware implementation procedure behind architectures will be explained clearly here. Finally, this chapter provides an overall summary of designs proposed in this thesis.

2.1 Overview of Cryptography

There are many aspects to security depending upon applications, ranging from secure commerce and payments to private communications and protecting health care information. One essential aspect for secure communications is that of cryptography. The word 'cryptology' comes from the Greek and means hidden or secret. Cryptography is the study of secure communication techniques that allow only the sender and intended recipient of a message to view its contents. It is a type of a rule or a technique by which private or sensitive information is secured from the public or other members. It focuses on the confidential data, authentication, data integrity etc. Cryptography is based on mathematical theory and some Computer Science principle [12]. The goal of the cryptography is to protect private communication in the public world.

The science of secure communication is known as cryptology which is a more general term than cryptography. Cryptology consists of two main branches: cryptography and cryptanalysis. The purpose of cryptography is to hide the contents of messages by encrypting them in order to make them unrecognizable except by someone who has the decryption key. It is the mathematical techniques to encrypt and decrypt data and supporting information security, such as confidentiality, integrity, non-repudiation, and authentication [2]. On the other hand, the purpose of cryptanalysis is to defeat this security by finding ways to decrypt messages without being given the decryption key. It is regarded as the study of methods to examine the security of a cryptographic algorithm or to find the weakness of a cryptographic system for getting the plaintext of a message.

Cryptography requires two fundamental steps: encryption and decryption. The encryption process encrypts a plaintext and turn it into ciphertext by using a cipher. Decryption process, on the other hand, applies that same cipher to turn the ciphertext back into plaintext [23].

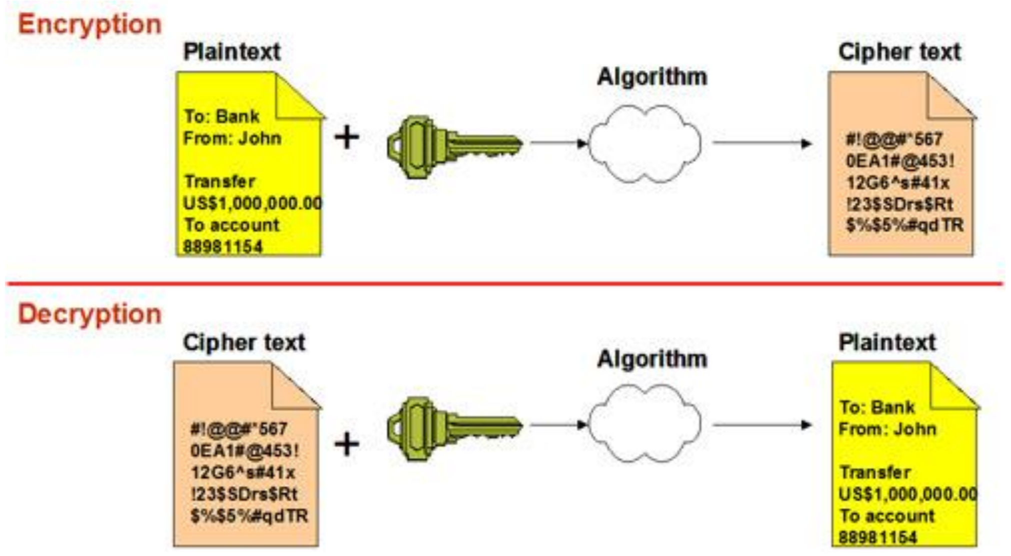


Figure-2-1: Encryption and decryption process of cryptography

2.2 Classification of Cryptography

Cryptographic systems can be divided into two categories. They are: 1) Private-Key Cryptography which uses a single key that both the sender and recipient have and 2) Public-Key Cryptography (PKC) which uses two keys, a public key known to everyone and a private key known to only the recipient of the message. They are described as below:

2.2.1 Private-key Cryptography

Private-key cryptography which is also known as symmetric or secret key cryptography can be defined as an encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. In private-key cryptography, the same key is shared between the sender and recipient to perform both the encryption and decryption. The process of encrypting and decrypting data with a shared key is illustrated in Fig. 2.2. To perform encryption, private-key cryptography is commonly used [22].

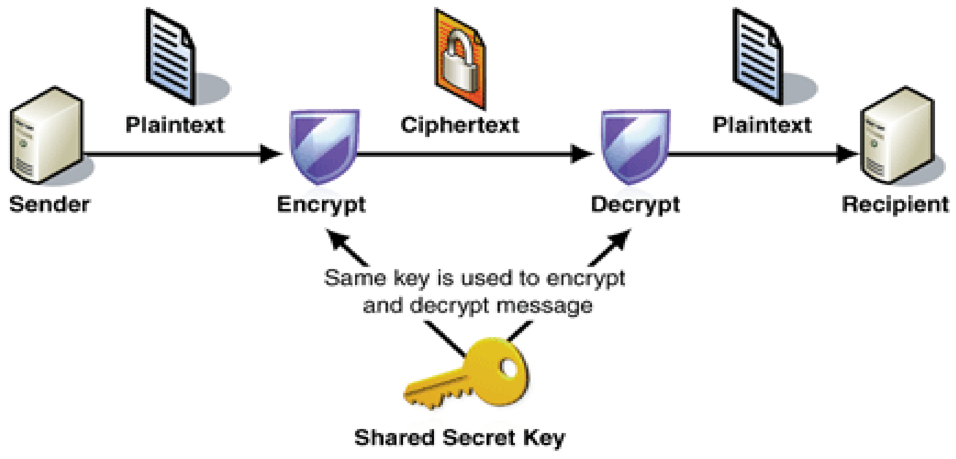


Figure-2-2: Private-key cryptography (or symmetric cryptography)

In private key cryptography, a sender sends a message in the form of plaintext as shown in figure-2.2. The plaintext is converted into ciphertext after encryption process by the secret key shared between the sender and recipient. The recipient decrypts the ciphertext by the same secret key and converts it back to the plaintext and reads the original message sent by the sender. Without sender and recipient, nobody can read the message because, he doesn’t know the secret key required for the encryption and decryption.

The major disadvantage of a private-key cryptosystem is related to the exchange of keys. Symmetric encryption is based on the exchange of a secret keys. The problem of key distribution therefore arises. Again, a user wanting to communicate with several people while ensuring separate confidentiality levels has to use as many private keys as there are people. For a group of *N* people using a secret-key cryptosystem, it is necessary to distribute a number of keys equal to N*(N-1)/2 [13].

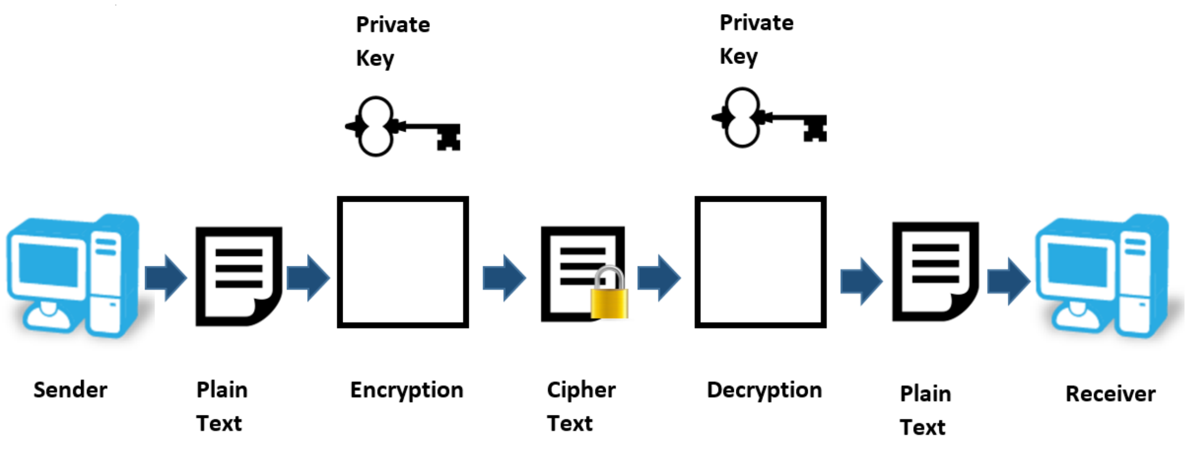


Figure-2-3: Private-key cryptography (or symmetric cryptography)

2.2.1 Public-key Cryptography

Public key cryptography, also known as asymmetric cryptography, uses public and private keys for encryption and decryption. The keys are usually large numbers that have been paired together but are not identical (asymmetric). One key in the pair can be shared with everyone; it is called the public key. The other key in the pair is kept secret; it is called the private key. Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption. Public key cryptography involves the use of a public and private key pair. A user may freely distribute the public key but must always keep the private key secret [14].

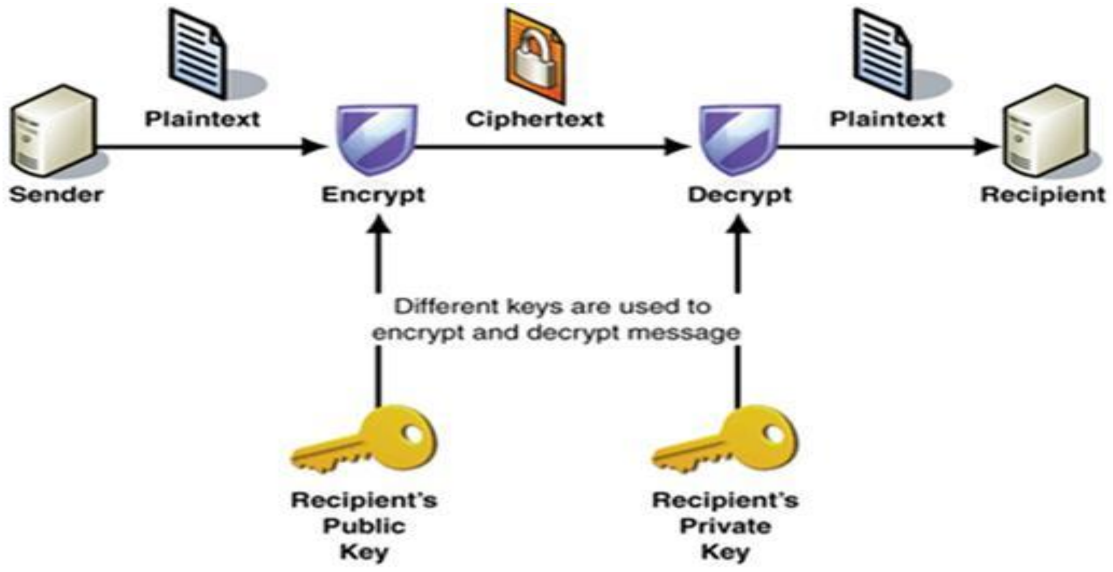


Figure-2-4: Public-key cryptography (or asymmetric cryptography)

Using this public-key cryptography method, the sender and receiver are able to authenticate one another as well as protect the secrecy of the message. Because pair of keys is required, this approach is also called asymmetric cryptography. Asymmetric encryption can be used for confidentiality, authentication, or both. The main objective of public-key encryption is to provide secrecy or confidentiality. The main advantage of public key cryptosystems is that there is no need to worry about key distribution problem. It is more flexible and easy to simply select larger keys. The disadvantage to public key cryptography is that it is necessarily slower and may also introduce added complexity as key length grows.

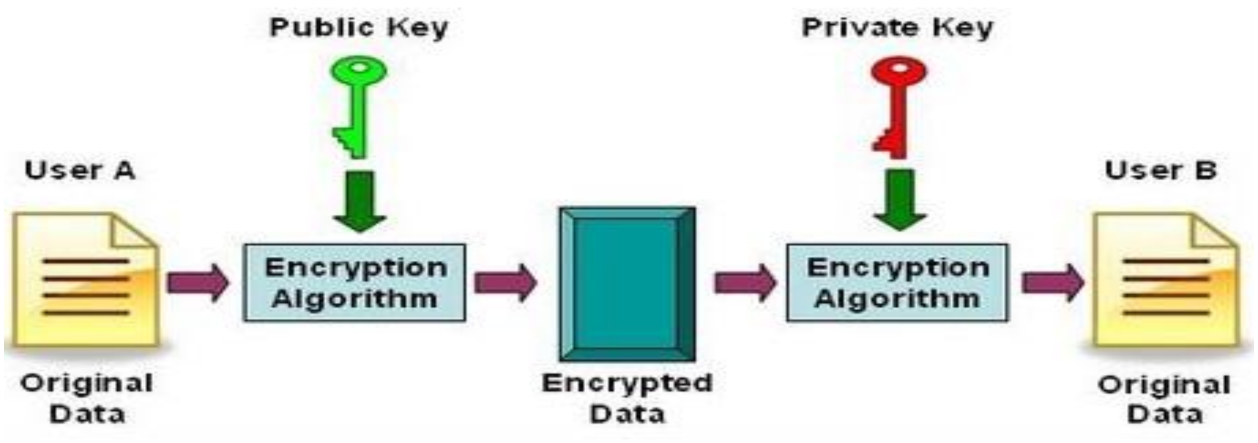


Figure-2-5: Public-key cryptography (or asymmetric cryptography)

2.3 Finite Field

Finite field, also known as Galois field, refers to a field in which there exists finitely many elements. It is usually useful while translating computer data as they are represented in binary forms. That is, computer data consist of combination of two numbers, 0 and 1, which are the components in Galois field whose number of elements is two [15]. A finite cyclic group is needed for a cryptosystem in which the group operations are efficiently calculable. A finite field is denoted as GF ($q = p^m$) where m is a positive integer and p is a prime number called the characteristic of field F. Three types of finite field exist in the literature: prime field, extension field, and binary field [4, 17].

Prime field: The field is said to be a prime field if m = 1. GF(p) = prime field of order p. GF (p) contains p elements, p – 1 units. At the basis of GF(p) related operations is integer modular arithmetic. Basic operations are:

- addition (GF add) : a + b mod p
- subtraction (GF sub) : a – b mod p
- multiplication (GF mul) : a x b mod p
- division (GF div) : a / b mod p
- inversion (GF inv) : 1 / b mod p

Extension field: The field is said to be a extension field if m is greater than 2.

Binary field: The field is said to be characteristic-two finite field or a binary field if $q = 2^m$. Basic operations are:

- addition (GF add) : A(x) + B(x)
- subtraction (GF sub) : A(x) – B(x)
- multiplication (GF mul) : A(x) x B(x) mod F(x)
- division (GF div) : A(x) / B(x) mod F(x)
- inversion (GF inv) : 1 / B(x) mod F(x)

Two main advantages regarding the Binary Finite Field math GF(2):

- the bit additions are performed mod 2 and hence represented in hardware by simple XOR gates => no carry chain is required
- the bit multiplications are represented in hardware by AND gates

2.4 Modular Arithmetic for Prime Field

The bottom level of an ECPM is made of modular arithmetic over prime field. All modular arithmetic operations over a prime field Fp (or GF(p)) are accomplished using modulo p, consisting of the integers between 0 and p-1. For any integer x, x mod p can be defined as the unique integer remainder q obtained upon dividing x by p; this operation is called reduction modulo p, which is required for cryptographic schemes. In cryptographic applications, e.g. an elliptic curve cryptosystem, x and p are large integers, such as 224 or 256 bits, and p must be a prime number [2].

Table-2.1: NIST recommended primes for prime field

Prime	Size (bits) m	Numeric Value
p192	192	$2^{192} - 2^{64} - 1$
p224	224	$2^{224} - 2^{96} + 1$
p256	256	$2^{256} - 2^{224} + 2^{192} + 2^{224} - 1$
p384	384	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
p521	521	$2^{521} - 1$

Table-2.2: Prime p for Koblitz curves over prime field

Prime	Size (bits) m	Numeric Value
p192	192	$2^{192} - 2^{32} - 2^{12} - 2^{8} - 2^{7} - 2^{6} - 2^{3} - 1$
p224	224	$2^{224} - 2^{32} - 2^{12} - 2^{11} - 2^{9} - 2^{7} - 2^{4} - 2 - 1$
p256	256	$2^{256} - 2^{32} - 2^{9} - 2^{8} - 2^{7} - 2^{6} - 2^{4} - 1$

The prime field for random curves recommended by NIST in the FIPS 186-2 standard for 192, 224, 256, 384 and 521 bits are presented in Table 2.1. In this thesis, prime p256 is used for Koblitz curve. 256 bits of security is used because it is enough for the modern day security devices.

2.4.1 Modular Adder over Prime Field

Modular addition adds two inputs, x, y, and subtracts the modulus p from the sum until the sum is less than the modulus p. The intermediate result (x + y) of modular addition could be greater than the modulus. To subtract the modulus p from the intermediate result, an adder is used that adds the intermediate result to the bitwise inverted modulus p with the carry-in set to 1, thus performing a two's-complement subtraction. The carry-out of the second adder checks whether the intermediate result is in the proper range or not. If the sum (x + y) is in the proper range, the result of the first adder is correct; otherwise, the second adder is right. The condition is checked by using a multiplexer to select whether (x + y) is greater than or equal to the modulus p [18].

$$Z = x + y \pmod{p}$$

The output Z of the modular addition is obtained by adding x and y and then subtracting the modulus p from the sum until the output is less than p. The modular reduction is usually very simple for an adder because the inputs x and y are in the range between 0 and p-1, hence the output Z must be $\leq 2p$

The formula can be written as: (**A + B**) **mod C = (A mod C + B mod C) mod C** [24]

Example:

Let **A=14, B=17, C=5**

Verify: **(A + B) mod C = (A mod C + B mod C) mod C**

LHS = Left Hand Side of the Equation

RHS = Right Hand Side of the Equation

LHS = (A + B) mod C

LHS = (**14 + 17**) mod 5

LHS = **31** mod 5

LHS = 1

RHS = (A mod C + B mod C) mod C

RHS = (**14 mod 5 + 17 mod 5**) mod 5

RHS = (**4 + 2**) mod 5

RHS = 1

LHS = RHS = 1

2.4.2 Modular Subtractor over Finite Field

Modular subtraction over the prime field is performed similarly to modular addition. For modular subtraction, y is bitwise inverted and added to x with a carry-in set to 1. If the result is negative or the carry-out of this adder is low, then the modulus p must be added to produce an output in the correct range between 0 and p - 1 inclusive.

$$\mathrm{Z} = \mathrm{x} - \mathrm{y}\ (\mathrm{mod}\ \mathrm{p})$$

The output Z can be reduced by simply subtracting p. Modular subtraction is very similar to modular addition. In a modular subtractor, if $x \geq y$ then it can be computed easily by simple subtraction or 2's-complement addition and the output Z will be in the range. Otherwise, if $x \leq y$ the modulus p should be added to the input x before the subtraction starts.

The formula can be written as: (**A - B**) **mod C = (A mod C - B mod C) mod C** [24]

Example:

Let **A=17, B=7, C=5**

Verify: **(A - B) mod C = (A mod C - B mod C) mod C**

LHS = Left Hand Side of the Equation

RHS = Right Hand Side of the Equation

LHS = (A - B) mod C

LHS = (**18-7**) mod 5

LHS = **11** mod 5

LHS = 1

RHS = (A mod C - B mod C) mod C

RHS = (**18 mod 5 - 7 mod 5**) mod 5

RHS = (**3 - 2**) mod 5

RHS = 1

LHS = RHS = 1

2.4.3 Modular Multiplier / Squarer over Finite Field

Modular multiplication, including squaring, is expensive and the most important operation over the prime field Fp. The basic modular multiplier operation of two elements is presented as

$$\boldsymbol{Z = (x \times y)\ (mod\ p)}$$

Example:

Consider x = 28; y = 20, and p = 29, then

$Z = (x \times y)\ (mod\ p)$= (28 × 20) (mod 29) = (19 × 29 + 9) (mod 29) = 9:

Here the reduction operation can be performed by divisions, but in a hardware implementation modular reduction using division is quite slow. Various methods are available to perform modular multiplication. Two efficient methods are Montgomery modular reduction [19] and Barrett modular reduction [20]. The Barrett algorithm consists of division operations which are usually slow, but it can be implemented efficiently by using right-shift operations. Montgomery modular multiplication is slightly faster than the Barrett algorithm.

An efficient method for modular multiplication is Interleaved modular multiplication. This is a very simple method as the first operand is multiplied bitwise with the second operand, then added to the intermediate result. The benefit of the interleaved method is that the intermediate result is only one or two bits larger than the operands because the intermediate result is always reduced by taking the modulus.

A modular squarer nothing but a modular multiplier except that only one input is required for a modular squarer rather than the two inputs for a modular multiplier; otherwise all other operations are the same as for modular multiplication [2].

2.5 Complexity Analysis of Modular Arithmetic over $\mathbb{F}_{\boldsymbol{P}}$

Modular addition, subtraction, and multiplication take a different number of clock cycles. Their complexity analyses in terms of clock cycles are reported in Table-3.3. As we can see from Table-3.1, the addition or subtraction takes one clock cycle to complete operation. Here, three different types of modular multipliers are implemented, where the radix-4 modular multiplier provides the best performance. It costs more area also.

Table-2.3 Complexity of GF(p) arithmetic operations in terms of clock cycles:

GF (p) operation	Complexity of operation in terms of clock cycles.
Addition / Subtraction	1
Combined addition and subtraction	2
Montgomery Multiplier	$m + 1$
Modified interleaved multiplier	$m + 1$
Radix-4 multiplier	$m/2 + 1$

Although Radix-4 multiplier has less complexity because of less number of required clock cycles Modified Interleaved multiplier avoids the costly modular inversion process. This is why, Modular Interleaved multiplier is used in this thesis.

2.6 Elliptic Curve Cryptography

Elliptical curve cryptography (ECC) is a public key encryption technique based on *elliptic curve theory* that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers. The technology can be used in conjunction with most public key encryption methods, such as RSA, and Diffie-Hellman. According to some researchers, ECC can yield a level of security with a 164-bit key that other systems require a 1,024-bit key to achieve. Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for mobile applications [6, 7].

ECC is based on properties of a particular type of equation created from the mathematical group derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that is very valuable for cryptography purposes: they are relatively easy to perform, and extremely difficult to reverse [2].

2.7 Overview of Elliptic Curve Cryptography

ECC can be implemented in either prime fields or binary fields; both provide almost the same level of security. An elliptic curve E over a field K, which is denoted E=K, is defined by the long form of the Weierstrass equation [4]

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where the coefficients a_1, a_2, a_3, a_4 and a_6 belong to the field K. To get the short form of the Weierstrass equation, there are two conditions: 1) if the underlying field K has characteristic = 2 or 3 and 2) if the underlying field K has the characteristic ≠ 2 and ≠ 3.

Based on the first condition, when the underlying field K has characteristic 2 and a_1= 0, the elliptic curve E becomes

$$y^2 + xy = x^3 + ax^2 + b$$

This equation is called the elliptic curve E over the binary field GF(2^m).

Based on the second condition and the elliptic curve E becomes

$$y^2 = x^3 + ax + b$$

where x, y, a, b ∈ GF(p) with

$$4a^3 + 27b^2 \neq 0$$

This equation is called the elliptic curve over GF(p).

First, the fact that the term in y is squared means that the curve is symmetrical about the x axis; i.e., for every positive value of y, there is a corresponding negative value with the same *x* coordinate. Also, the fact that the highest order term in x is x-cubed means that, for certain values of a and b, there is exactly one point of inflection in the line on each side of the x axis. The inflection points are located where the line crosses the y axis. A necessary consequence for curves with these two properties is that a straight line that crosses the curve in two places also crosses the curve in exactly one other place. The operation of adding two points involves finding the third point of intersection between the curve and the straight line passing through the two points that are being added, then reflecting the result in the x axis. Adding a point to itself is also known as doubling the point, and it involves finding a second point where the tangent at the first point intersects the curve, and then reflecting that point in the x-axis.

Let P = (x, y) be a point in an affine coordinate system; the projective coordinates P = (X, Y, Z) are given by the following:

$$X = x; \quad Y = y; \quad Z = 1$$

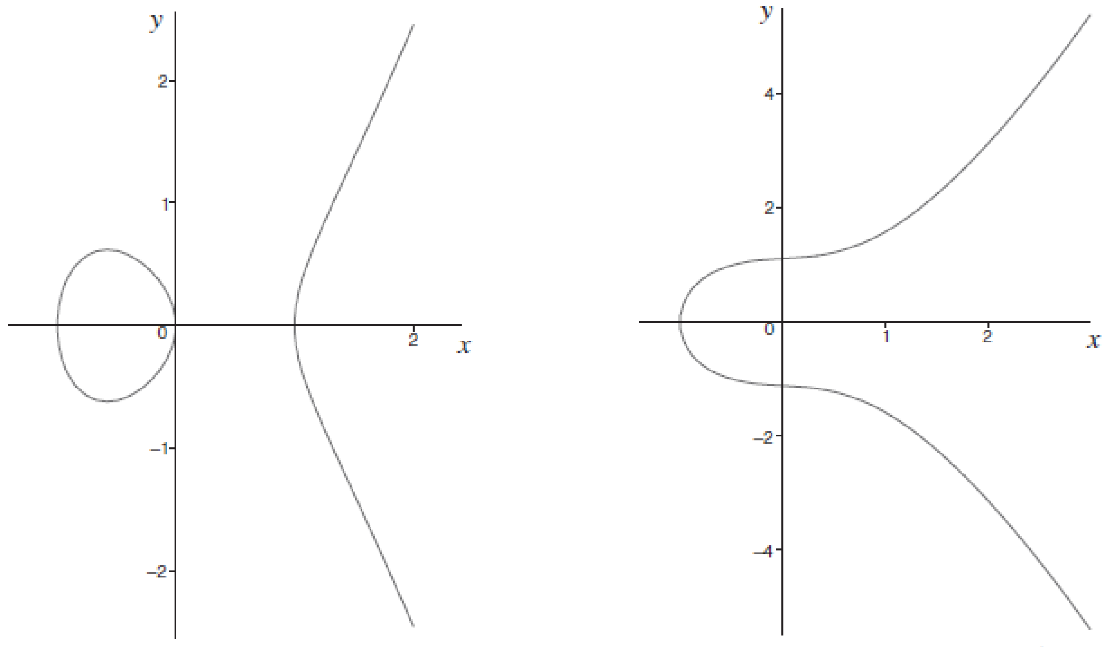
The projective point P = (X, Y, Z), Z ≠ 0 corresponding to the affine point (P = (x, y)) is given by

$$x = \frac{X}{Z^2}$$

$$y = \frac{Y}{Z^3}$$

The Weierstrass equation of the elliptic curve becomes

$$Y^2 = X^3 + aXZ^4 + bZ^6$$



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Figure-2-6: Elliptic curves according to equations [4]

2.8 Elliptic Curve Point Arithmetic (Group Operation)

The second level in the hierarchy of an elliptic curve cryptosystem is elliptic curve point arithmetic (or group operations), point doubling (PD) and point addition (PA). Given two distinct points P = (x1, y1) and Q = (x2, y2), then the PA of P and Q is the point R (R = P + Q) which is the line that crosses the points P and Q on the elliptic curve E, also the point R is the reflection of the point about the x-axis, as shown in Fig. 2.4(a). Similarly, the PD is the results of adding a point P itself which can be defined as R = 2P. First draw the tangent line to the elliptic curve at P then the projection over the x axis of the point is defined as R, The PD and PA can be computed both in the binary field and the prime field.

Several projective coordinates have been proposed to represent elliptic curve points.

In the binary field, the projective coordinates are
1) Standard,
2) Jacobian, and
3) Lopez-Dahab projective coordinates.

In the prime field Fp, the available projective coordinates are:
1) Standard projective coordinates,
2) Jacobian projective coordinates, and
3) Chudnovsky coordinates. In this dissertation,
Jacobian projective coordinates have been used to implement the PD and PA operations in this thesis.

2.9 Point doubling and point addition over in Jacobian Projective Coordinates

The Jacobian projective form of the Weierstrass equation for an elliptic curve over the prime field can be written as

$$\mathrm{Y}^2 = \mathrm{X}^3 + \mathrm{aXZ}^4 + \mathrm{bZ}^6$$

Where P = (X, Y, Z) is the point of projective coordinates

2.9.1 Point Doubling over $\mathbb{F}_{\boldsymbol{P}}$

Given a point P = (X1, Y1, Z1) on the elliptic curve, then the PD over GF(p) in Jacobian projective coordinates for Koblitz curve R = 2P = (X3, Y3, Z3) can be computed as follows [21]

$X_3 = (3X_1^2)^2 - 8X_1Y_1^2$

$Y_3 = (3X_1^2)(4X_1Y_1^2 - X_3) - 8Y_1^4$

$Z_3 = 2Y_1Z_1$

2.9.2 Point Addition over $\mathbb{F}_{\boldsymbol{P}}$

Given two points P = (X1, Y1, Z1) and Q = (X2, Y2, Z2) on the elliptic curve, then the PA over GF(p) in Jacobian coordinates R = P +Q = (X3, Y3, Z3) can be computed as follows

$X_3 = A^2 - B^3 - 2X_1Z_2^2B^2$

$Y_3 = A(X_1Z_2^2B^2 - X_3) - Y_1Z_2^3B^3$

$Z_3 = Z_1Z_2B$

Where,

$A = Y_2Z_1^3 - Y_1Z_2^3$

$B = X_2Z_1^2 - X_1Z_2^2$

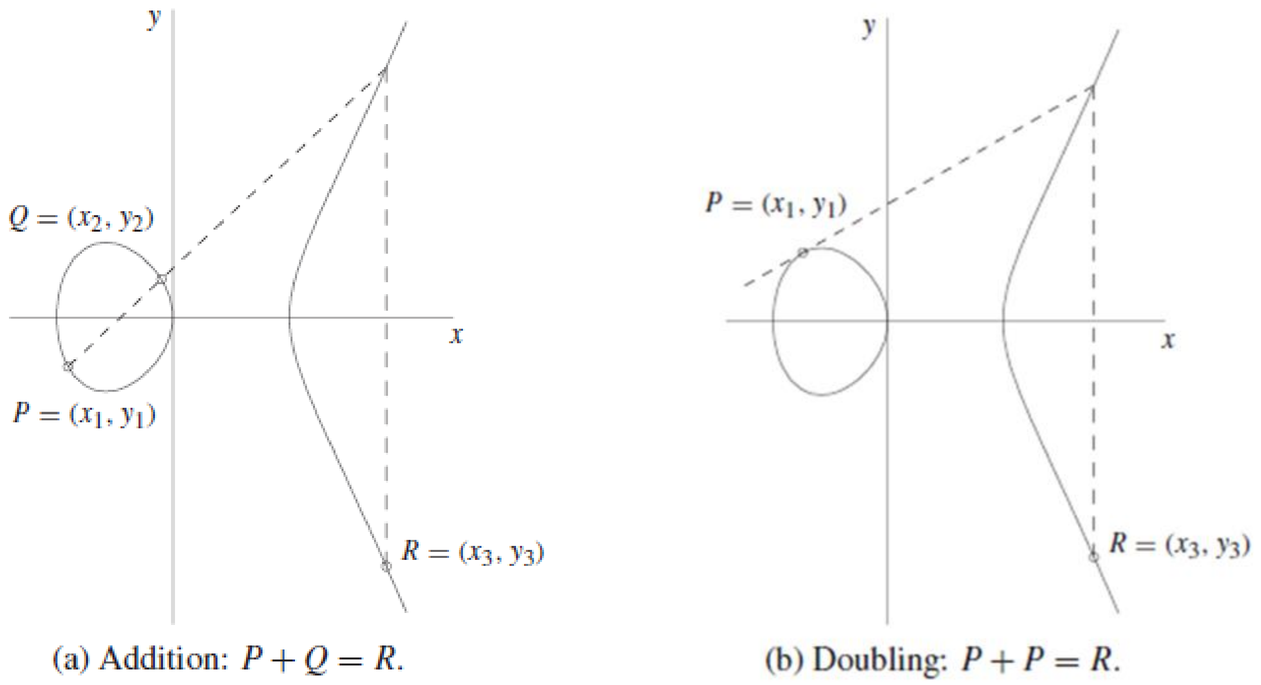


Figure-2-7: (a) Point addition; (b) Point doubling

2.10 Elliptic Curve Cryptography Processor (ECP)

The computation of an Elliptic curve cryptosystem over GF(p) consists of three levels: 1) modular arithmetic 2) group operations and 3) elliptic curve scaler multiplication (ECSM).

Modular Arithmetic
EC Group Operation
ECSM

The implementation hierarchy of the Elliptic Curve Cryptographic operations over a prime finite field is presented in Figure. ECC protocols are the building blocks of Elliptic curve scalar or point multiplication (ECPM), Elliptic curve group operations and finite field modular arithmetic. The top level of the ECC cryptosystem contains ECC protocol like ECDSA (Elliptic Curve Digital Signature Algorithm). The second level contains ECPM, which is the series of EC group operations like Elliptic curve point addition (ECPA) and Elliptic curve point doubling (ECPD). The third level comprises ECPA and ECPD, which are called Elliptic curve group operations. These are the series of the finite field modular arithmetic operations such as modular addition, modular subtraction, modular multiplication, modular squaring. The finite field modular arithmetic units are the bottom or fourth level in the hierarchy. For this finite field modular arithmetic, modular multiplication is the most crucial for the overall performance of the ECC processor because most of the clock latency depends upon the operation of this modular multiplication

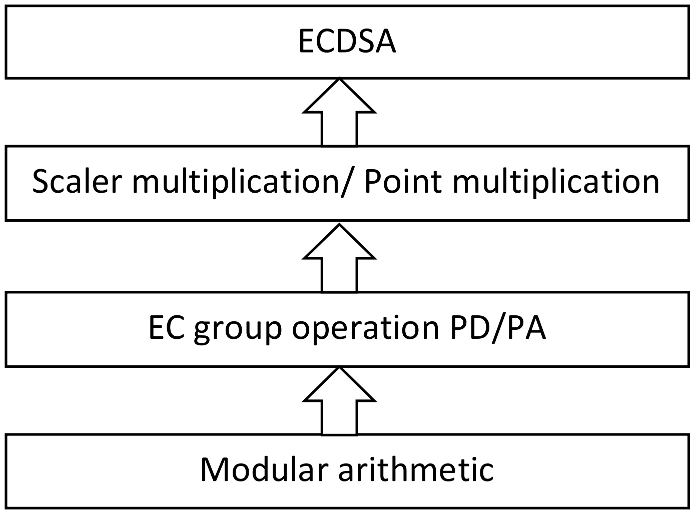


Figure-2-8: Implementation hierarchy of the ECP operation over GF(p).

2.11 Hardware Implementation on FPGA and ASIC

FPGA stands for Field Programmable Gate Array. An FPGA is a component that can be thought of as a giant ocean of digital components (gates, look-up-tables, flip-flops) that can be connected together by wires. The code that you write makes real physical connections with wires to perform the function that you need. What makes FPGAs and ASICs special is that they are very good at performing a large number of operations in parallel.

ASIC stands for Application Specific Integrated **C**ircuit. An ASIC is similar in theory to an FPGA, with the exception that it is fabricated as a custom circuit. This means that – unlike FPGAs – it is not reprogrammable, so you had better get it right the first time! Since ASICs are custom circuits, they are very fast and use less power than an FPGA.

Both FPGAs and ASICs are designed with a Hardware Description Language (HDL). In this thesis, FPGA is used as the hardware platform.

Figure-2-9: Virtex-5 FPGA board [25]

Chapter 3

Hardware Architecture

Hardware architecture refers to the identification of a system's physical components and their interrelationships. In this section, the proposed hardware architectures for the modular arithmetic operations and elliptic curve group operations have been demonstrated. In our thesis, five architectures have been designed in total for finite field arithmetic and group operation**.** Three architectures have been developed for finite field arithmetic i.e. modular addition, modular subtraction and modular multiplication; another two for group operations i.e. point doubling and point addition. For all the hardware architectures, National Institute of Standard and Technology (NIST) standard of $\mathbb{F}_{\mathbf{256}}$ has been used. According to the standards of NIST, the prime number used for 256-bit operation for Koblitz curves over prime field is

$$p = 2^{256} - 2^{32} - 2^{9} - 2^{8} - 2^{7} - 2^{6} - 2^{4} - 1$$

$$p = 115792089237316195423570985008687907853269984665640564039457584007908834671663$$

3.1 Hardware for Finite Field Arithmetic

3.1.1 Architecture for Modular Addition over $\mathbb{F}_{\boldsymbol{p}}$

Modular addition adds two inputs, x, y, and subtracts the modulus p from the sum until the sum is less than the modulus p. The intermediate result (x + y) of modular addition could be greater than the modulus. To subtract the modulus p from the intermediate result, an adder is used that adds the intermediate result to the bitwise inverted modulus p with the carry-in set to 1, thus performing a two's-complement subtraction. The carry-out of the second adder checks whether the intermediate result is in the proper range or not. If the sum (x + y) is in the proper range, the result of the first adder is correct; otherwise, the second adder is right. The condition is checked by using a multiplexer to select whether (x + y) is greater than or equal to the modulus p.

Here,

$$Z = x + y \ (\text{mod p})$$

If $(x + y) \leq p, Z = (x + y)$;

If $(x + y) > p, Z = (x + y) - p$;

Algorithm 3.1: Algorithm for modular addition in GF(p)

```
Input: Prime $p$ and $x, y \in [1, p - 1]$
Output: $Z = (x + y) \ mod \ p$
1.   $Z = 0$;
2.   $Z_a = x + y$;
3.   $Z_b = (x + y) - p$;
4.1  if $Z_a \leq p$ then $Z = Z_a$;
4.2  else if $Z_a > p$ then $Z = Z_b$;
5.   Return $Z$;
```

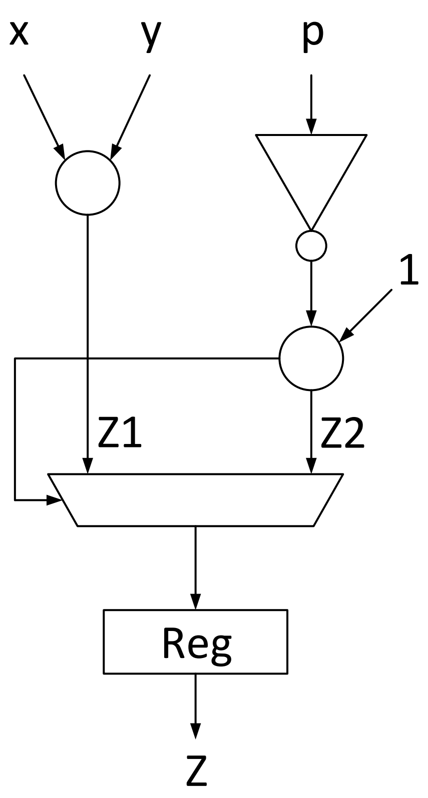


Figure-3.1: Hardware architecture of Modular Adder.

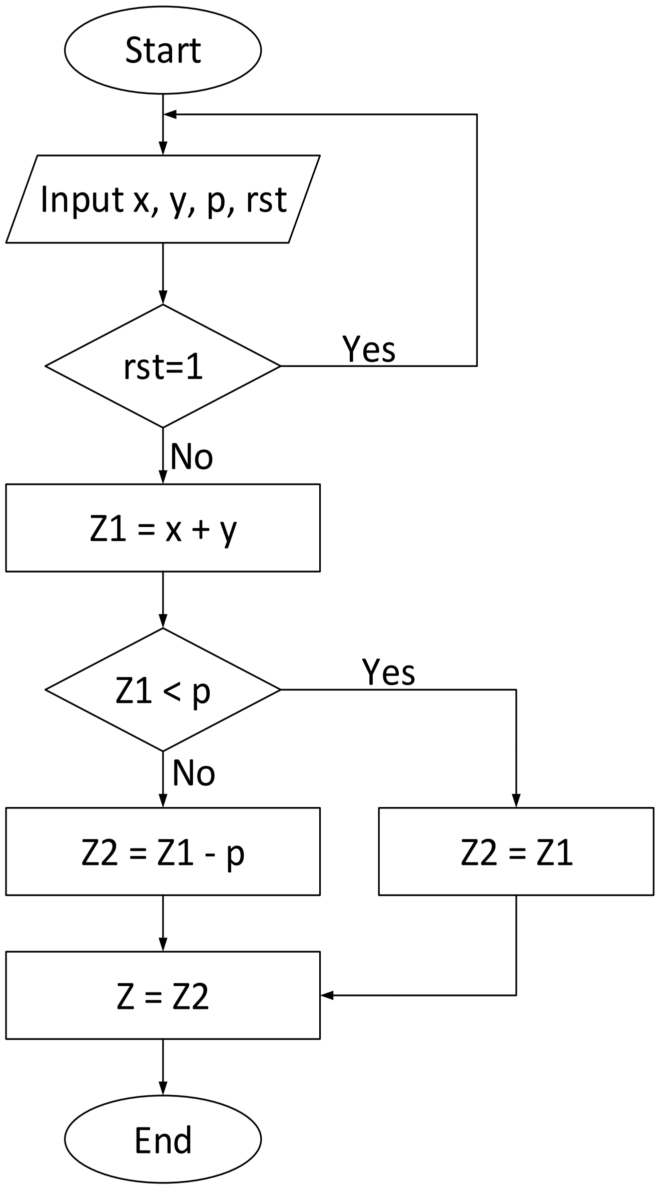


Figure-3.2: Flow chart of Modular Adder.

3.1.2 Architecture for Modular Subtraction over $\mathbb{F}_{\boldsymbol{p}}$

Modular subtraction over the prime field is performed similarly to modular addition. For modular subtraction, y is bitwise inverted and added to x with a carry-in set to 1. If the result is negative or the carry-out of this adder is low, then the modulus p must be added to produce an output in the correct range between 0 and p - 1 inclusive.

Here,

$$Z = (x - y)\ mod\ p$$

If $0 \leq (x - y) \leq p, Z = (x - y);$

If $-p \leq (x - y) < 0, Z = (x - y) + p;$

Algorithm 3.2: Algorithm for modular subtraction in GF(p)

```
Input: Prime $p$ and $x, y \in [1, p - 1]$
Output: $Z = (x - y)\ mod\ p$
1.   $Z = 0;$
2.   $Z_a = x - y;$
3.   $Z_b = (x - y) + p;$
4.1  if $x \geq y$ then $Z = Z_a;$
4.2  else if $x < y$ then $Z = Z_b;$
5.   Return $Z;$
```

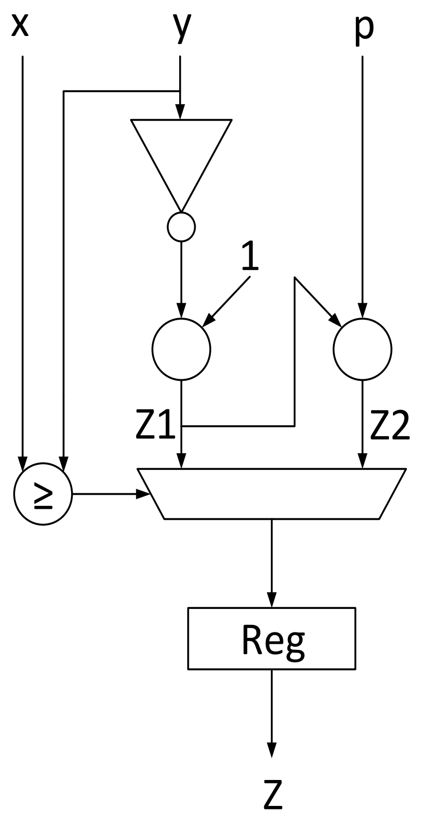


Figure-4.1: Hardware architecture of

Modular Subtractor.

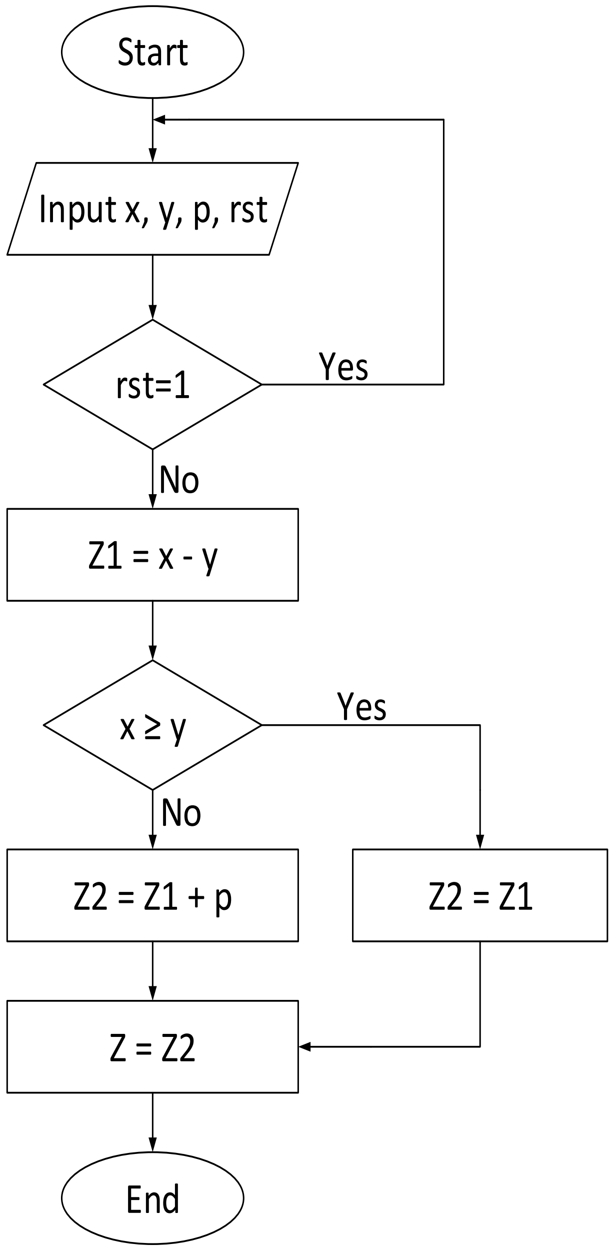


Figure-4.2: Flow chart of Modular Subtractor.

3.1.3 Architecture for Modular Multiplication over $\mathbb{F}_{\boldsymbol{p}}$

Modular multiplication is the backbone of the whole structure of Elliptic curve cryptographic processor. The performance of a cryptographic processor depends on the efficiency of modular multiplication operation at every stage. So, efficient modular multiplier is mandatory for Edward curve cryptography.

Modular multiplication is presented as,

$$Z = (x * y) mod\ p = (x\ mod\ p * y\ mod\ p)\ mod\ p \qquad 4.3$$

An efficient algorithm is proposed for modular multiplication/squaring as shown in Algorithm-3.3 which is based on the interleaved multiplication method.

Algorithm 3.3: Algorithm for modular multiplication in GF(p)

```
Input: Prime $p$ and $A, B \in [1, p-1]$
Output: $C$ =(A*B) mod p
1.    $C = 0; p_2 = 2 * p$ ( pre-computed);
2.1     for $i = m - 1$ down to 0 do
2.2   $c_1 = C; c_2 = 2 * c_1$ ( left-shift operation);
2.3   $i_1 = A[i] * B$ ( and-gate operation);
2.4   $c_3 = c_2 + i_{1s}; c_4 = c_3 - p; c_5 = c_3 - p_2\ (p_2 = 2p)$;
2.5   if $c_3 \geq p$ then $c_6 = c_4$;
2.6   else if $c_3 \geq p_2$ then $c_6 = c_5$; else $c_6 = c_3$; end if $C = c_6$
2.7   end for
3.    Return C;
```

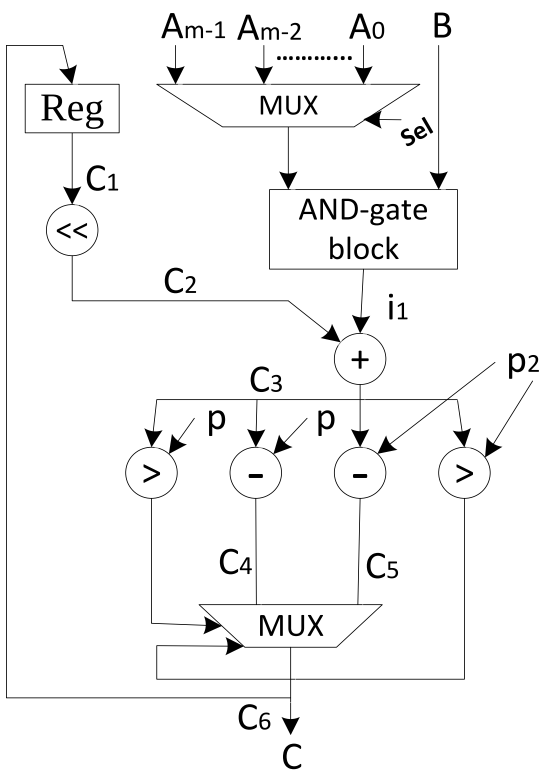


Figure-3.5: Hardware architecture of modular multiplier [2]

Figure-3.5 presents the hardware architecture of Algorithm 3.3 over prime field $\mathbb{F}_P$.The proposed modular multiplication algorithm is very efficient because the critical path consists of only gates, adder and subtractor. This approach requires (*m* +1) cycles to compute the final result of modular multiplication where *m* is the bit length of operands *A*, *B* or *p*.

In this method, one left shift operation, one simple addition, two comparators, two subtractions and two multiplexers operations are required. The first comparator is used to compare the value of C_3 with *P*. Another comparator is used to compare the value of C_3 with *2p*.

If $C_3 \leq p,$ the final output is *C3*.

If $P < C_3 < 2p,$ the final output is $C_3 - p$.

If $C_3 \geq 2p,$ the final output is$C_3 - 2p$.

The two subtractors are used for two subtractions. One for $C_3 - p$ and another for $C_3 - 2p$. Two of these subtractors, one is used depending on the value of C_3.

The first multiplexer is used to select a bit of the first operand A such as $A_s = A[i]$. The second multiplexer is used to select the final output.

When ‘select = 01’, the final output $C_6 = C_3$.

When ‘select = 10’, the final output $C_6 = C_4 = C_3 - p$.

When ‘select = 11’, the final output $C_6 = C_5 = C_3 - 2p$.

3.2 Hardware for Group Operation

3.2.1 Architecture for Point Doubling

Given a point P = (X1, Y1, Z1) on the elliptic curve, then the PD over GF(p) in Jacobian projective coordinates for Koblitz curve R = 2P = (X3, Y3, Z3) can be computed as follows [21]

$$X_3 = (3X_1^2)^2 - 8X_1Y_1^2$$

$$Y_3 = (3X_1^2)(4X_1Y_1^2 - X_3) - 8Y_1^4$$

$$Z_3 = 2Y_1Z_1$$

The proposed hardware architecture for point doubling is shown in fig. 3.6. Pre-computation, balancing of architecture and parallelization of operations have been applied to reduce the total clock cycle (CC), power consumption, longest path and to increase the speed of operation. For example, in level 1, two multiplications and one squaring are operated in parallel. So only 257 CCs are required instead of 3×257 CCs i.e. the CCs needed in level 1 is reduced to one-third of CCs if parallelization was not applied. Modular inversion has been avoided as it is the most costly operation. The overall latency required for PDBL is 4m+3 , where m is the latency of multiplication. The overall architecture requires 3 multipliers, 4 squarers, 9 adders and 3 subtractors. So the total cost of PDBL is 3MUL+4SQ+9ADD+3SUB. Only 7 levels are required to complete the total PD operation. Required total clock cycle is (257+257+257+1+1+257+1)=1031 CCs.

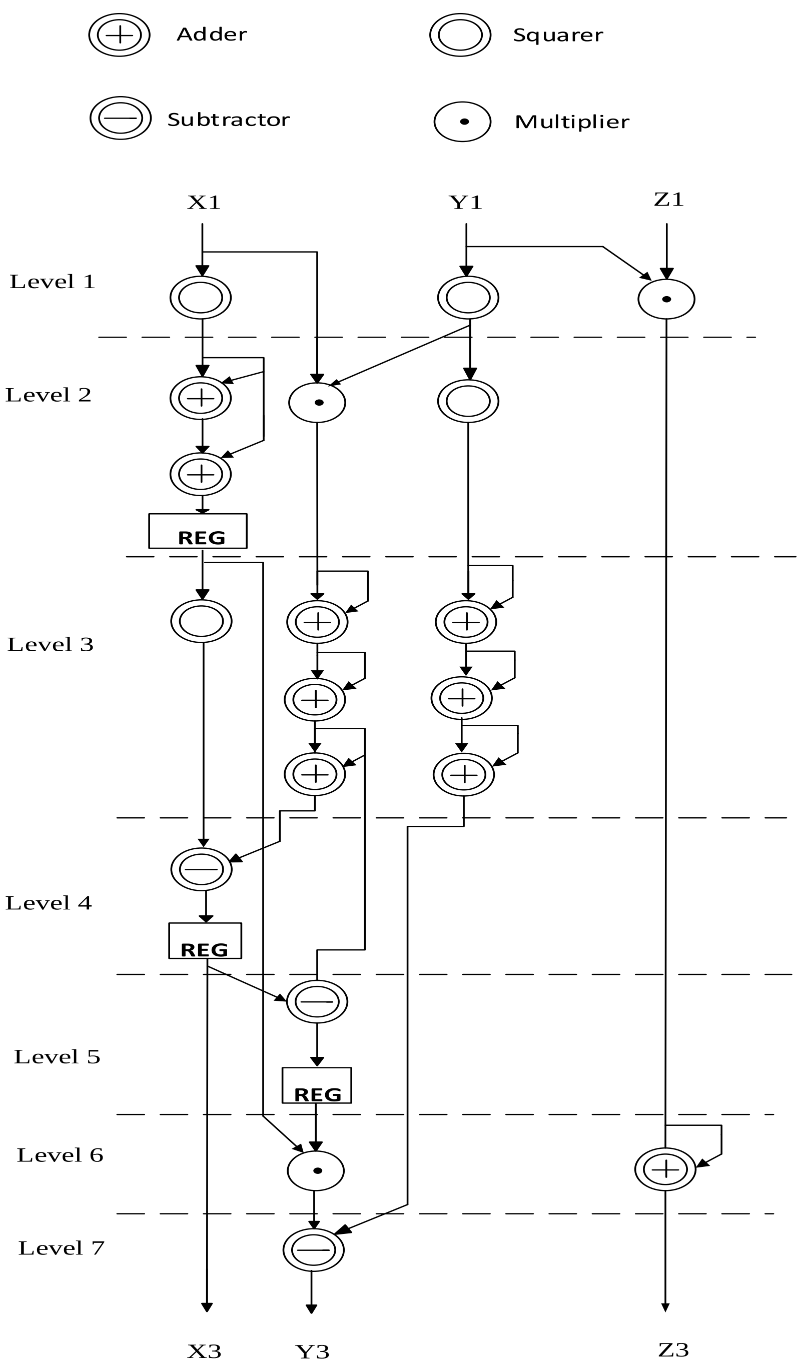


Figure-3.6: Hardware architecture of point doubling

3.2.2 Architecture for Point Addition

Given two points P = (X1, Y1, Z1) and Q = (X2, Y2, Z2) on the elliptic curve, then the PA over GF(p) in Jacobian coordinates R = P +Q = (X3, Y3, Z3) can be computed as follows

$$X_3 = A^2 - B^3 - 2X_1Z_2^2B^2$$

$$Y_3 = A(X_1Z_2^2B^2 - X_3) - Y_1Z_2^3B^3$$

$$Z_3 = Z_1Z_2B$$

Where,

$A = Y_2Z_1^3 - Y_1Z_2^3$

$B = X_2Z_1^2 - X_1Z_2^2$

The hardware architecture for point addition is proposed and it is shown in fig. 3.7. Pre-computation, balancing of architecture and parallelization of operations have also been used in case of point addition operation for achieving better performance. For example, five multiplications are operated in parallel in level 2. So only 257 CCs are required instead of 5×257 CCs i.e. the CCs needed in level 2 is reduced to one-fifth of CCs if parallelization was not applied. The overall latency required for PDBL is 6m+4 , where m is the latency of multiplication. The overall architecture requires 12 multipliers, 4 squarers, 1 adders and 6 subtractors. So the total cost of PDBL is 12MUL+4SQ+1ADD+6SUB. Only 10 levels are required to complete the total PD operation. Required total clock cycle is (257+257+257+257+257+1+1+1+257+1) =1546 CCs.

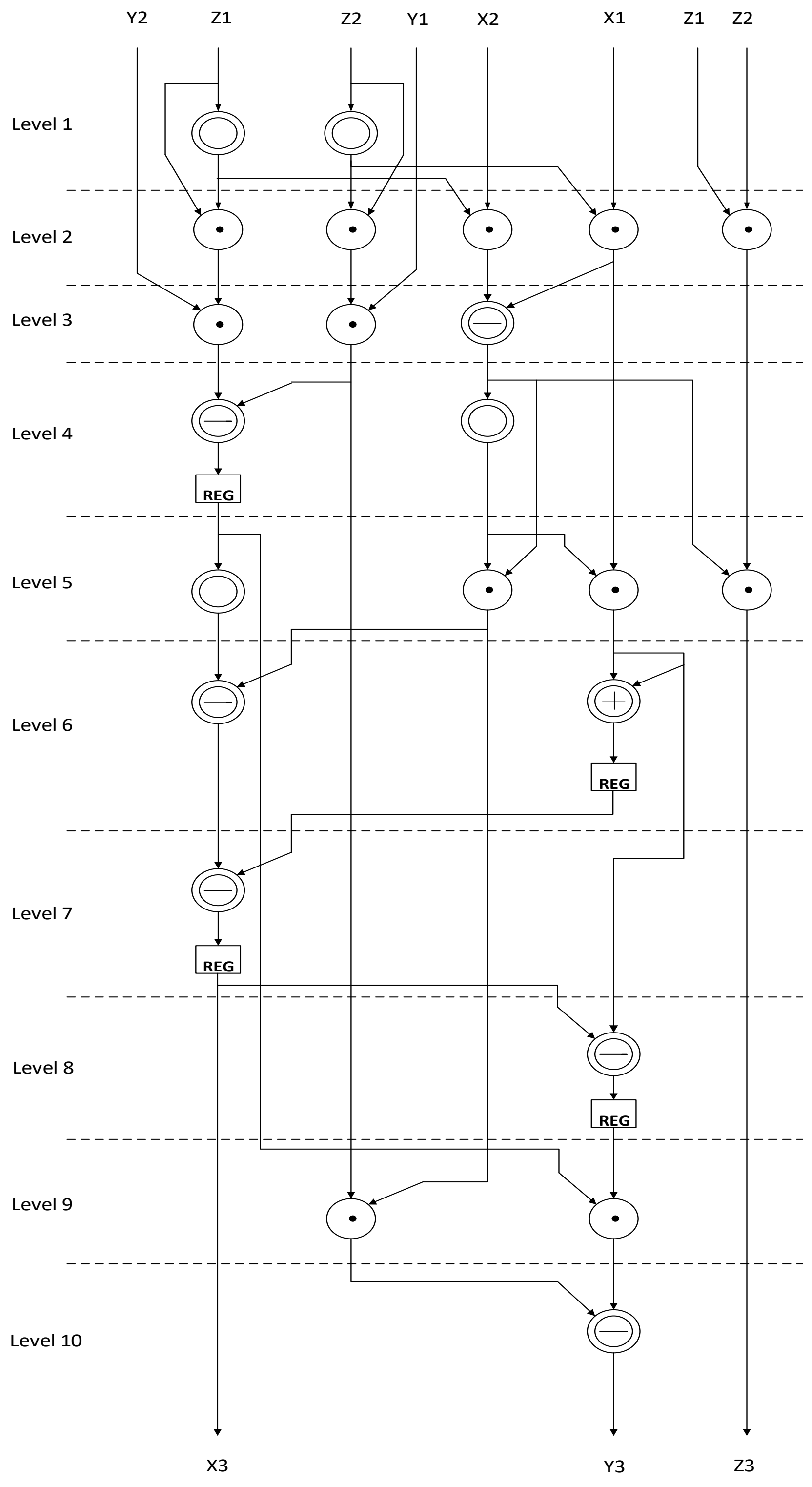


Figure-3.7: Hardware architecture of point addition

Chapter 4

Results and Discussion

This section presents the implementation results for the proposed design. The proposed design has been simulated using both ModelSim PE and ISim, and synthesized using Xilinx ISE 14.7 for an FPGA board of Virtex-5 family with device code XC5VLX50T with an optimized goal of 'Speed'. All simulation results are verified using high-level Maple software.

4.1 Implementation & Simulation Results of Modular Addition

4.1.1 Implementation of Modular addition

Device utilization summary (Estimated Values) of modular addition:

Table-4.1 Device utilization summary of modular addition

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	768	28800	2%
Number of fully used LUT-FF pairs	0	768	0%
Number of bonded IOBs	771	480	160%
Number of BUFG/BUFGCTRLs	1	32	3%

Timing Summary of modular addition:

Speed Grade: - 2

Minimum period = 5.655ns (Maximum Frequency: 176.835MHz)

Minimum time required = $5.655ns \times 1 = 5.655\ ns$

4.1.2 Simulation and Verification of Modular Addition

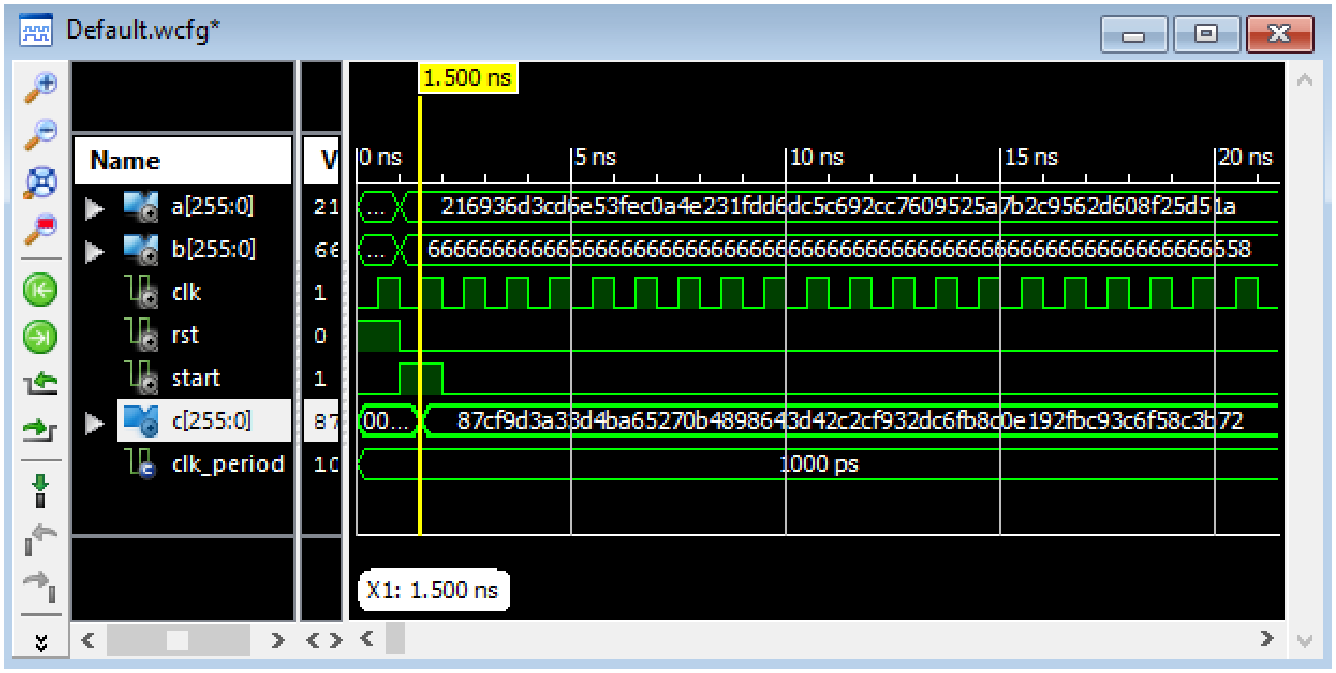
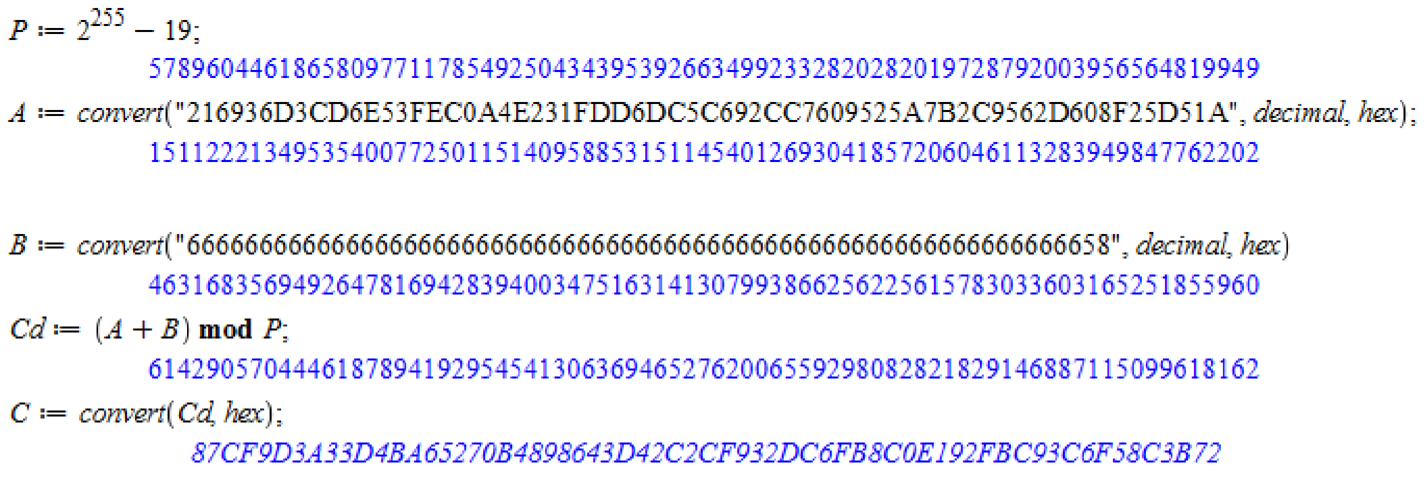


Figure-4.1: Simulation waveform of modular addition in virtex-5 FPGA

The modular adder requires only 1 clock cycle to be completed. We have taken 1 clock cycle = 1ns. So the addition operation requires 1ns to be finished.

The modular addition operation is also performed in Maple for verification as shown in figure-4.2. The results which were found from the simulation of modular addition (figure-4.1) are same as the results observed in Maple.



```
P := 2^255 − 19;
                57896044618658097711785492504343953926634992332820282019728792003956564819949
A := convert("216936D3CD6E53FEC0A4E231FDD6DC5C692CC7609525A7B2C9562D608F25D51A", decimal, hex);
                15112221349535400772501151409588531511454012693041857206046113283949847762202

B := convert("6666666666666666666666666666666666666666666666666666666666666658", decimal, hex)
                46316835694926478169428394003475163141307993866256225615783033603165251855960
Cd := (A + B) mod P;
                61429057044461878941929545413063694652762006559298082821829146887115099618162
C := convert(Cd, hex);
                  87CF9D3A33D4BA65270B4898643D42C2CF932DC6FB8C0E192FBC93C6F58C3B72
```

Figure-4.2: Results verification of modular addition in Maple

4.2 Implementation & Simulation Results of Modular Subtraction

4.2.1 Implementation of Modular Subtraction

Device utilization summary of modular subtraction:

Table-4.2 Device utilization summary of modular subtraction

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	1279	28800	4%
Number of fully used LUT-FF pairs	0	1279	0%
Number of bonded IOBs	771	480	160%
Number of BUFG/BUFGCTRLs	1	32	3%

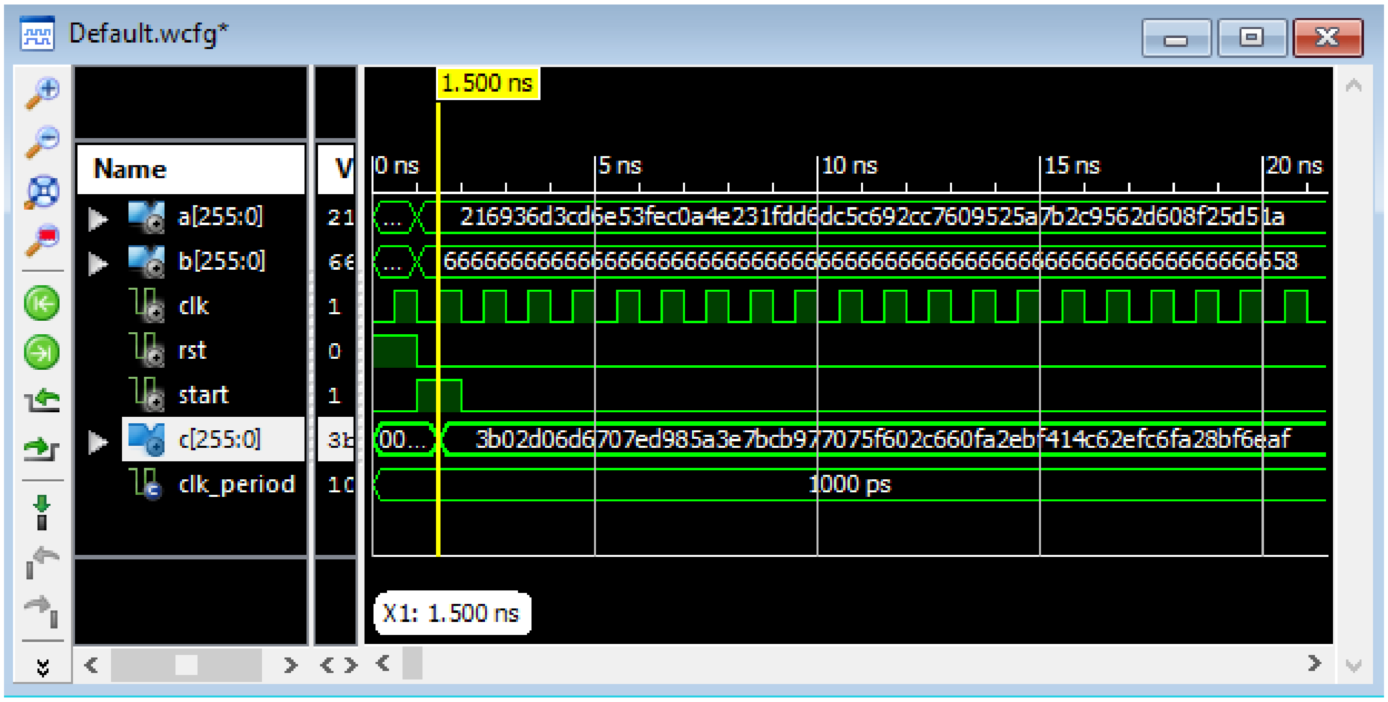
Timing Summary of modular subtraction:

Speed Grade: - 2

Minimum period = 5.667ns (Maximum Frequency: 176.460MHz)

Minimum time required = $5.667ns \times 1 = 5.667\ ns$

4.2.2 Simulation and Verification of Modular subtraction

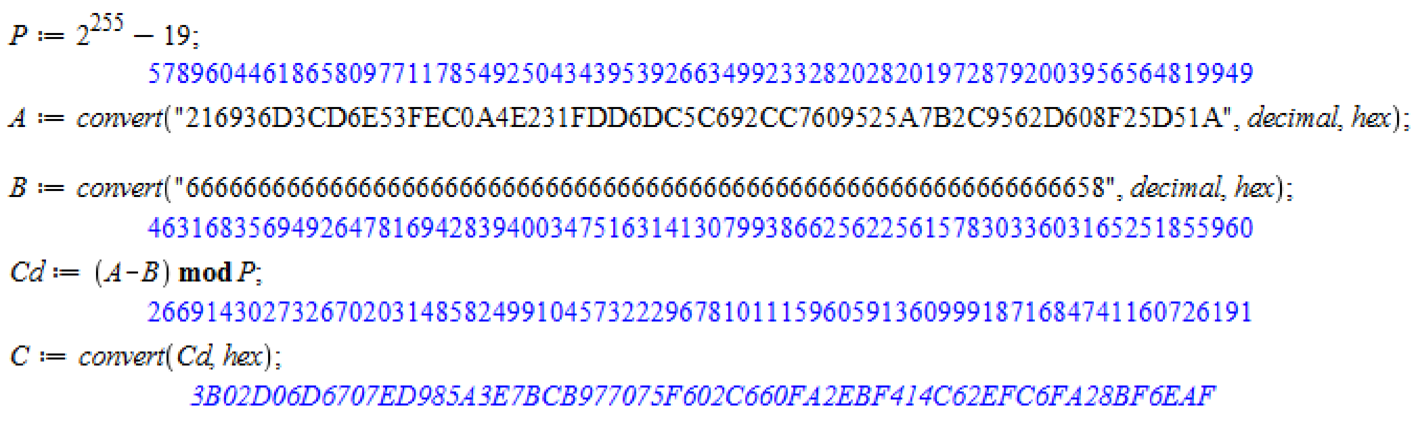


```
a[255:0]     21   216936d3cd6e53fec0a4e231fdd6dc5c692cc7609525a7b2c9562d608f25d51a
b[255:0]     66   6666666666666666666666666666666666666666666666666666666666666658
clk          1
rst          0
start        1
c[255:0]     3b   3b02d06d6707ed985a3e7bcb977075f602c660fa2ebf414c62efc6fa28bf6eaf
clk_period   10   1000 ps
```

Figure-4.3: Simulation waveform of modular subtraction in virtex-5 FPGA

The modular subtractor requires only 1 clock cycle to be completed. We have taken 1 clock cycle = 1ns. So the subtraction operation requires 1ns to be finished.

The modular subtraction operation is also performed in Maple for verification as shown in figure-4.4. The results which were found from the simulation of modular subtraction (figure-4.3) are same as the results observed in Maple.



```
P := 2^255 - 19;
            57896044618658097711785492504343953926634992332820282019728792003956564819949
A := convert("216936D3CD6E53FEC0A4E231FDD6DC5C692CC7609525A7B2C9562D608F25D51A", decimal, hex);

B := convert("6666666666666666666666666666666666666666666666666666666666666658", decimal, hex);
            46316835694926478169428394003475163141307993866256225615783033603165251855960
Cd := (A-B) mod P;
            26691430273267020314858249910457322296781011159605913609991871684741160726191
C := convert(Cd, hex);
                3B02D06D6707ED985A3E7BCB977075F602C660FA2EBF414C62EFC6FA28BF6EAF
```

Figure-4.4: Results verification of modular subtraction in Maple

4.3 Implementation & Simulation Results of Modular Multiplication

4.3.1 Implementation of Modular Multiplication

Device utilization summary of modular multiplication:

Table-4.3 Device utilization summary of modular multiplication

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	522	28800	1%
Number of Slice LUTs	1489	28800	5%
Number of fully used LUT-FF pairs	514	1497	34%
Number of bonded IOBs	772	480	160%
Number of BUFG/BUFGCTRLs	1	32	3%

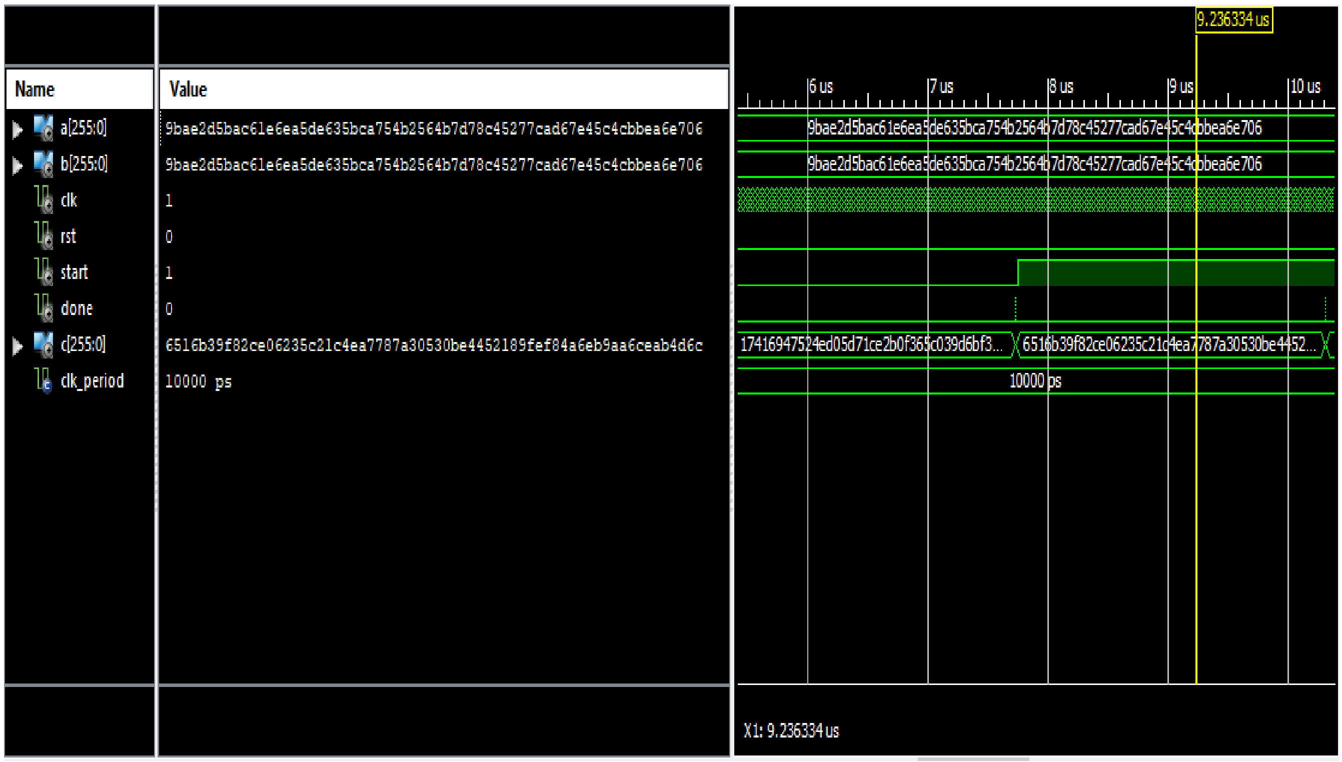
Timing Summary of modular multiplication:

Speed Grade: - 2

Minimum period = 12.784ns (Maximum Frequency: 78.222MHz)

Minimum time required = $12.784ns \times 257 = 3.28\mu s$

4.3.2 Simulation and Verification of Modular Multiplication



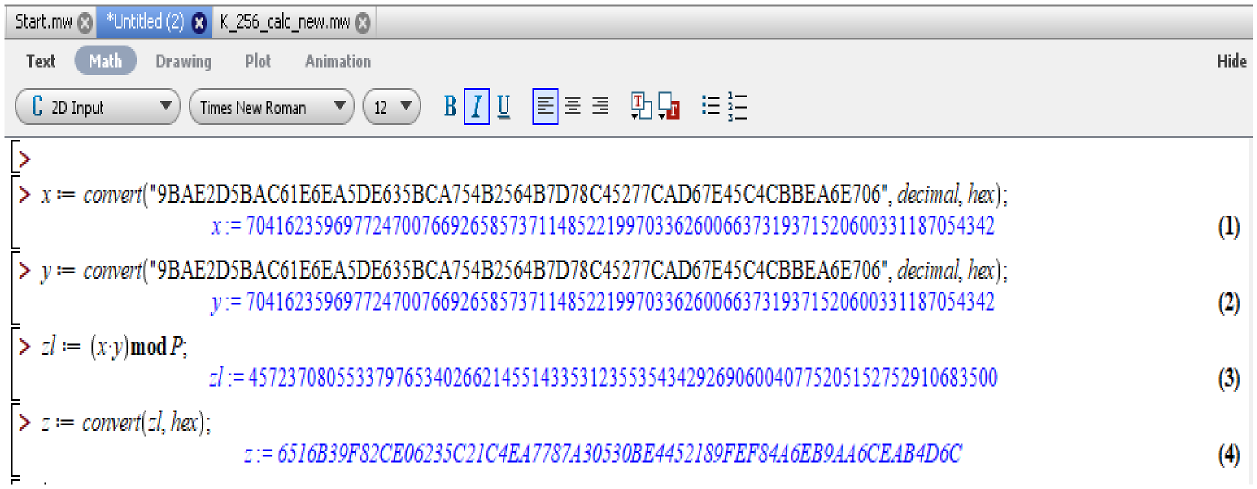
```
Name         Value
a[255:0]     9bae2d5bac61e6ea5de635bca754b2564b7d78c45277cad67e45c4cbbea6e706
b[255:0]     9bae2d5bac61e6ea5de635bca754b2564b7d78c45277cad67e45c4cbbea6e706
clk          1
rst          0
start        1
done         0
c[255:0]     6516b39f82ce06235c21c4ea7787a30530be4452189fef84a6eb9aa6ceab4d6c
clk_period   10000 ps
```

9.236334 us
6 us
7 us
8 us
9 us
10 us
9bae2d5bac61e6ea5de635bca754b2564b7d78c45277cad67e45c4cbbea6e706
9bae2d5bac61e6ea5de635bca754b2564b7d78c45277cad67e45c4cbbea6e706
17416947524ed05d71ce2b0f365c039d6bf3...
6516b39f82ce06235c21c4ea7787a30530be4452...
10000 ps
X1: 9.236334 us

Figure-4.5: Simulation waveform of modular multiplication in virtex-5 FPGA

For *n bit* operation, the modular multiplication requires $(n + 1)$ clock cycles to be completed. So for 256-bit operation, the modular multiplication requires 257 clock cycles. We have taken 1 clock cycle = 1ns. So the multiplication operation requires 257ns to be finished.

The modular multiplication operation is also performed in Maple for verification as shown in figure-4.6. The results which were found from the simulation of modular multiplication (figure-4.5) are same as the results observed in Maple.



```
> x := convert("9BAE2D5BAC61E6EA5DE635BCA754B2564B7D78C45277CAD67E45C4CBBEA6E706", decimal, hex);
        x := 70416235969772470076692658573711485221997033626006637319371520600331187054342    (1)
> y := convert("9BAE2D5BAC61E6EA5DE635BCA754B2564B7D78C45277CAD67E45C4CBBEA6E706", decimal, hex);
        y := 70416235969772470076692658573711485221997033626006637319371520600331187054342    (2)
> zl := (x·y) mod P;
        zl := 45723708055337976534026621455143353123553543429269060040775205152752910683500    (3)
> z := convert(zl, hex);
        z := 6516B39F82CE06235C21C4EA7787A30530BE4452189FEF84A6EB9AA6CEAB4D6C    (4)
```

Figure-4.6: Results verification of modular multiplication in Maple

4.4 Implementation & Simulation Results of Point Doubling

4.4.1 Implementation of Point Doubling

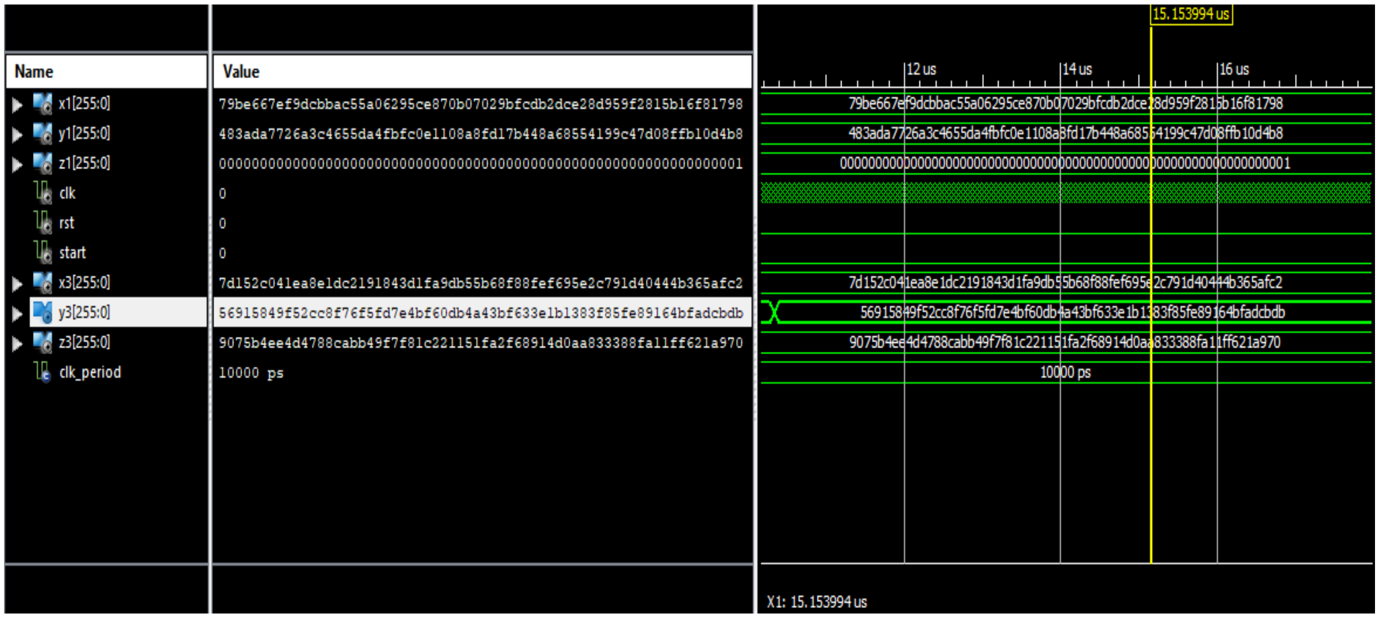
Device utilization summary of point doubling

From the table, the number of LUT-FF pair is 6548, number of slice LUT is 20003 and number of slice register is 6729. The total area is comprised of all the above three parameters.

Table-4.4 Device utilization summary of point doubling

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	6729	28800	23%
Number of Slice LUTs	20003	28800	69%
Number of fully used LUT-FF pairs	6548	20184	32%
Number of bonded IOBs	1540	480	320%
Number of BUFG/BUFGCTRLs	2	32	6%

4.4.2 Simulation of point Doubling



```
Name          Value
x1[255:0]     79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
y1[255:0]     483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
z1[255:0]     0000000000000000000000000000000000000000000000000000000000000001
clk           0
rst           0
start         0
x3[255:0]     7d152c041ea8e1dc2191843d1fa9db55b68f88fef695e2c791d40444b365afc2
y3[255:0]     56915849f52cc8f76f5fd7e4bf60db4a43bf633e1b1383f85fe89164bfadcbdb
z3[255:0]     9075b4ee4d4788cabb49f7f81c221151fa2f68914d0aa833388fa11ff621a970
clk_period    10000 ps
```

Figure-4.7: Simulation waveform of point doubling in virtex-5 FPGA

The result of point doubling is obtained at 1.034 us. Here, the clock cycle is taken as 1 ns. As a result total time is 1034 ns (1.034 us).

4.5 Implementation & Simulation Results of Point Addition

4.5.1 Implementation of Point Addition

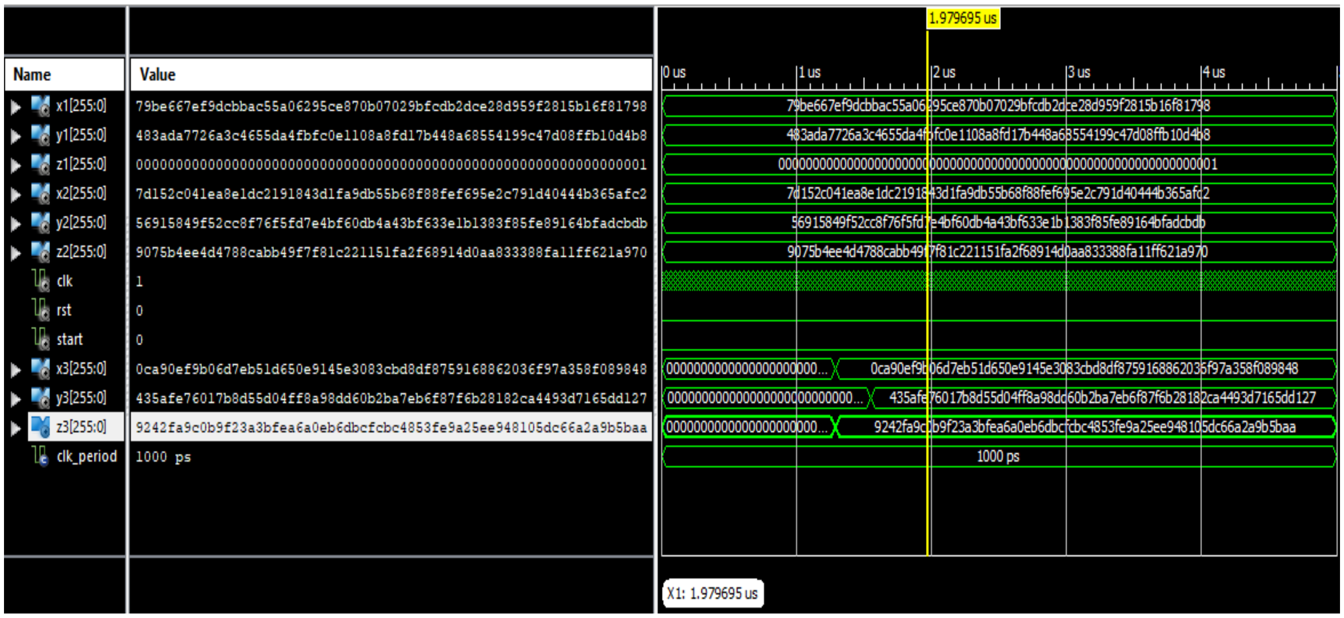
Device utilization summary of point addition

From the table, the number of LUT-FF pair is 9561, number of slice LUT is 33297 and number of slice register is 10141. The total area is comprised of all the above three parameters

Table-4.3 Device utilization summary of modular multiplication

Device Utilization Summary (Estimated Values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	10141	28800	35%
Number of Slice LUTs	33297	28800	115%
Number of fully used LUT-FF pairs	9561	33877	28%
Number of bonded IOBs	2308	480	480%
Number of BUFG/BUFGCTRLs	3	32	9%

4.5.2 Simulation of point Addition



```
Name          Value
x1[255:0]     79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
y1[255:0]     483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
z1[255:0]     0000000000000000000000000000000000000000000000000000000000000001
x2[255:0]     7d152c041ea8e1dc2191843d1fa9db55b68f88fef695e2c791d40444b365afc2
y2[255:0]     56915849f52cc8f76f5fd7e4bf60db4a43bf633e1b1383f85fe89164bfadcbdb
z2[255:0]     9075b4ee4d4788cabb49f7f81c221151fa2f68914d0aa833388fa11ff621a970
clk           1
rst           0
start         0
x3[255:0]     0ca90ef9b06d7eb51d650e9145e3083cbd8df8759168862036f97a358f089848
y3[255:0]     435afe76017b8d55d04ff8a98dd60b2ba7eb6f87f6b28182ca4493d7165dd127
z3[255:0]     9242fa9c0b9f23a3bfea6a0eb6dbcfcbc4853fe9a25ee948105dc66a2a9b5baa
clk_period    1000 ps
```

Figure-4.7: Simulation waveform of point addition in virtex-5 FPGA

4.6 Hardware Implementation Summary of Proposed Elliptic Curve Cryptosystem

Table-4.6: Hardware Implementation Summary of Proposed Elliptic Curve Cryptosystem

Implementation Summary

Module	Clock Cycle Required	Minimum Time Period (ns)	Maximum Frequency (MHz)	Total Time Required (ns)	Number of Slice Registers	Number of LUT-FF pairs	Number of LUTs
Modular Multiplication	257	12.784	78.222	3285.488	522	514	1489
Point Doubling	1031	13.220	75.641	13629.82	6729	6548	20003
Point Addition	1546	13.220	75.641	20438.12	10141	9561	33297

Total summary of the implementation of the proposed architectures are shown in table-4.7. The working frequency, delay and area of different units can be found from the table. Total area is comprised of the number of LUT-FF pair, number of slice LUT and number of slice register which are all presented in the table.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

Hardware implementation of efficient modular arithmetic and group operation for elliptic curve cryptosystem over prime field $\mathbb{F}_{256}$ was the prime focus of this thesis. For doing this high-performance hardware architectures of modular addition, subtraction, multiplication along with point doubling and point addition operations were designed. All the proposed architectures have been simulated using both ModelSim PE and ISim, and synthesized using Xilinx ISE 14.7 for an FPGA board of Virtex-5 family with device code xc7vx485t with an optimized goal of 'Speed'. All simulation results are verified using high-level Maple software.

NIST prime P-256 was used for all the hardware implementations of this work and the Jacobian coordinate system has been used to utilize them. The proposed modular adder and subtractor requires only 1 clock cycle each. An efficient modular multiplier is designed using interleaved modular multiplication method which requires 257 clock cycles. This modular multiplication needs 522 slices registers, 1489 slice LUTs and 514 LUT-FF pairs with a delay of 3.28 us. The group operations which include point doubling and point addition were also implemented using pre-computations and parallelization of operations for reducing the delay. For the point doubling operation the number of required LUT-FF pair is 6548, a number of required slice LUT is 20003 and number of required slice register is 6729 and the delay is 13.62 us. For the point addition operation, the number of required LUT-FF pair is 9561, a number of required slice LUT is 33297 and number of required slice register is 10141 and the delay is 20.43 us.

So, a high performance hardware architecture of modular arithmetic and group operation for elliptic curve cryptosystem over prime field $\mathbb{F}_{256}$ was implemented considering trade-off between area, time and power consumption which can be further used in elliptic curve scalar multiplication or elliptic curve point multiplication to build an overall high performance hardware architecture for elliptic curve cryptosystem over prime field $\mathbb{F}_{256}$.

5.2 Future Works

In the future, our thesis can be utilized and extended for the following aspects:

- ❑ Carry-save-adder can be used for improving the performance of modular multiplication.
- ❑ The group operations PD, PA (point doubling, point addition) can be rearranged to decrease the AT, hence increase the performance.
- ❑ Proposed modular arithmetic and group operations can be utilized to design an Elliptic Curve Point Multiplication (ECPM).
- ❑ The total architecture of Elliptic curve cryptographic processor (ECP) can be developed using the ECPM module to perform the encryption-decryption operation for security.
- ❑ The Elliptic Curve Digital Signature Algorithm (ECDSA) can be designed using the above implementations.

References

[1] Mohamed Barakat, Christian Eder, Timo Hanke, “An Introduction to Cryptography”, September 20, 2018.

[2] Md Selim Hossain, “High Performance Hardware Implementation of Elliptic Curve Cryptography”, Research Book, Doctor of Philosophy, Department of Engineering, Faculty of Science and Engineering, Macquarie University, Sydney, Australia, June 2017.

[3] N. Koblitz, A. Menezes, and S. Vanstone, “The state of elliptic curve cryptography,” Des. Codes Cryptography, vol. 19, no. 2, pp. 173–193, Mar. 2000.

[4] D. Hankerson, A. J. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[5] D. Pamula, “Arithmetic operators on GF(2m) for cryptographic applications: performance-power consumption - security tradeoffs,” Theses, Université Rennes 1, Dec. 2012.

[6] Koblitz, N.: ‘Elliptic curve cryptosystems’, Math. Comput., 1987, 48, pp. 203–209

[7] Miller, V.S.: ‘Use of elliptic curves in cryptography’. Proc. CRYPTO 1985,1986, pp. 417–426

[8] R. G. Kammer, “NIST- National Institute of Standards and Technology, Digital Signature Standard, FIPS Publication 186-2,” 2000.

[9] “IEEE standard specifications for public-key cryptography,” IEEE Std 1363-2000, pp. 1–228, Aug. 2000.

[10] “X 9.62 public key cryptography for the financial services industry: Elliptic curve digital signature algorithm (ECDSA),” american national standards institute,” 1999.

[11] Rivest, R.L., Shamir, A., Adleman, L.: ‘A method for obtaining digital signatures and public-key cryptosystems’, *Commun. ACM*, 1978, 21, (2), pp.120–126

[12] https://www.c-sharpcorner.com/article/overview-of-cryptography/

[13] https://ccm.net/contents/130-private-key-or-secret-key-cryptography

[14] https://searchsecurity.techtarget.com/definition/asymmetric-cryptography

[15] Christoforus Juan Benvenuto “Galois Field in Cryptography” May 31, 2012

[16] https://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography

[17] G. Sutter, J. Deschamps, and J. Imana, “Efficient elliptic curve point multiplication using digit-serial binary field operations,” IEEE Trans. Ind. Electron., vol. 60, no. 1, pp. 217–225, Jan. 2013.

[18] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, Handbook of Applied Cryptography, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[19] P. L. Montgomery, “Modular multiplication without trial division,” Mathematics of Computation, vol. 44, no. 170, pp. 519–521, 1985.

[20] P. Barrett, Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 311–323.

[21] G. Orlando and C. Paar, “A high-performance reconfigurable elliptic curve processor for GF(2m),” in Proc. CHES, 2000, pp. 41–56.

[22] http://immanueluccdbq.org/cipher-tutorial/cipher-tutorial-astonishing-javaquery-tutorial-with-code-symmetric-key-encryption/

[23] https://commons.wikimedia.org/wiki/File:Symmetric_key_encryption.svg

[24] https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic

[25] https://store.digilentinc.com/genesys-virtex-5-fpga-development-board-limited-time-see-genesys2/