

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**ECC IMPLEMENTATION ON FPGA WITH FAST AND
SCHEDULED ARITHMETIC UNITS**



ŞÜKRÜ UZUN

**A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE
DEPARTMENT OF ELECTRONIC ENGINEERING**

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ECC IMPLEMENTATION ON FPGA
WITH FAST AND SCHEDULED
ARITHMETIC UNITS

ŞÜKRÜ UZUN
A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE
DEPARTMENT OF ELECTRONIC ENGINEERING

THESIS SUPERVISOR
ASSOC. PROF. DR. SERDAR SÜER ERDEM

GEBZE
2022

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HIZLI VE PLANLI ARİTMETİK
MODULLER İLE FPGA ÜZERİNDE ECC
GERÇEKLEME

ŞÜKRÜ UZUN
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DOÇ. DR. SERDAR SÜER ERDEM

GEBZE
2022



YÜKSEK LİSANS JÜRİ ONAY FORMU

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 10/02/2022 tarih ve 2022/10 sayılı kararıyla oluşturulan jüri tarafından 23/02/2022 tarihinde tez savunma sınavı yapılan Şükrü Uzun'un tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI)

: Doç Dr. Serdar Süer Erdem

ÜYE

: Dr. Öğr. Üyesi Köksal Hocaoglu

ÜYE

: Prof. Dr. Ali Tangel

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve/..... sayılı kararı.

İMZA/MÜHÜR

SUMMARY

Elliptic Curve Cryptosystems (ECC) have grown in prominence over the last decade as a result of their reduced bit sizes and ease of modular reduction compared to other public-key cryptosystems like RSA. Since the mid-1980s, ECC has been extensively studied in academics and investigated in industry as a viable public-key cryptography solution. To achieve the most compact and efficient architecture, many software and hardware implementations have been created. However, there is still room for advancement in FPGAs to speed up computations. Despite the fact that ASICs are the fastest way to implement any cryptographic algorithm in any sector, FPGAs are still commonly used in reality, especially for prototyping, due to their lower cost. In this paper, we provide an improved multiplication architecture with a pipelined ECC calculation schedule for scalar multiplication throughput. Despite the fact that there are two major prime fields of ECC, $GF(p)$ and $GF(2^m)$, our research focuses on $GF(p)$, which is the most prevalent and is also recommended by the National Institute of Standards and Technology (NIST). Because of its key size and security correlation, P256 (256-bit key length) is one of the most popular curves in the $GF(p)$ prime field.

Key Words: Elliptic Curve Cryptosystem, Big Integer Arithmetic, Fast Multiplication, FPGA, High-Performance.

ÖZET

Son yıllarda, Eliptik Eğri Kriptosistem (ECC), RSA gibi diğer açık anahtarlı şifreleme sistemlerine kıyasla daha küçük bit boyutları ve modüler indirgeme kolaylığı nedeniyle popüleritesi artıyor. 80'lerin ortalarından bu yana, ECC, akademide geniş çapta çalışıldı ve endüstride verimli bir açık anahtar şifreleme seçeneği olarak kullanılmak üzere araştırıldı. Günümüze kadar kompakt ve verimli tasarımı uygulamak için birçok yazılım ve donanım uygulaması geliştirilmiştir. Ancak, FPGA'larda hesaplamaları hızlandırmak için hala potansiyel iyileştirmeler mevcuttur. ASIC, herhangi bir alanda herhangi bir kriptografik algoritmayı uygulamanın en hızlı yolu olmasına rağmen, fiyatları nedeniyle, FPGA'lar özellikle prototipleme için pratikte hala yaygın tercih olarak görülmektedir. Bu çalışmada, skaler çarpmanın verimini artırmak için geliştirilmiş bir çarpma tasarımı ve ardışık düzende ECC hesaplama çizelgesi sunuyoruz. ECC'nin $GF(p)$ ve $GF(2^m)$ olmak üzere iki ana asal alanı olmasına rağmen, çalışmamız sadece en yaygın olanlarından biri olan ve National Institute of Standards tarafından önerilen $GF(p)$ 'yi kapsamaktadır. P256 (256 bit anahtar uzunluğu), anahtar boyutu ve güvenlik korelasyonu nedeniyle $GF(p)$ asal alanındaki en popüler eğrilerden biridir.

Key Words: Eliptik Eğri Kriptosistem, Büyük Integer Aritmetik, Hızlı Çarpma, FPGA, Yüksek Performans.

ACKNOWLEDGMENTS

First and foremost, I want to express my thanks to Assoc. Prof. Dr. Serdar Suer Erdem, for admitting me as a student. His classes and assignments were really instructive, and I learnt a great informations from him. I'd also like to express my gratitude to Dr. Furkan Cayci, my co-advisor. In overseeing, coaching, and mentoring me, he was really nice and helpful. He trusted me and set reasonable goals/deadlines for me. I appreciate his patience and encouragement, which allowed me to finish this thesis. Finally, I want to express my gratitude to the members of the thesis defense committee for their time and effort.

I'd also like to thank Procenne, my company, for allowing me to be a part of such a terrific team and projects, as well as for their interest in my continued education.

Finally, I want to express my gratitude to my family and friends for their unwavering support, companionship, love, and encouragement over the years.

TABLE of CONTENTS

	<u>Page</u>
SUMMARY	v
ÖZET	vi
ACKNOWLEDMENTS	vii
TABLE of CONTENTS	viii
LIST of ABBREVIATIONS and ACRONYMS	x
LIST of FIGURES	xi
LIST of TABLES	xii
1. INTRODUCTION	1
1.1. Cryptography	1
1.2. Symmetric Algorithms	3
1.3. Asymmetric Algorithms	4
1.4. Motivation	5
1.5. Contribution of Thesis	6
2. BACKGROUND	7
2.1. Digital Design Circuits	7
2.2. Field Programmable Gate Arrays	8
2.3. Metric Parameters	9
2.3.1. Area	10
2.3.2. Power Consumption	10
2.3.3. Throughput	11
2.3.4. Latency	11
3. ELLIPTIC CURVE CRYPTOGRAPHY	12
3.1. Mathematical Background	12
3.1.1 Coordinate Systems	14
3.1.2. Point Addition	14
3.1.3. Point Doubling	15
3.1.4 Discrete Logarithm Problem	17
3.2. Related Works	17
4. IMPLEMENTATIONS OF ELLIPTIC CURVE CRYPTOGRAPHY	19

4.1. Addition-Subtraction	20
4.2. Multiplication	21
4.3. Modular Reduction	25
4.4 Modular Inversion	26
4.5. Pipeline and Scheduling	28
4.6. Implementation	29
5. CONCLUSION	31
REFERENCES	32
BIOGRAPHY	35



LIST OF ABBREVIATIONS AND ACRONYMS

<u>Abbreviations</u>	<u>Explanations</u>
<u>and Acronyms</u>	
AES	: Advance Encryption Standard
ASIC	: Application Specific Integrated Circuit
CLB	: Configurable Logic Block
CPU	: Central Processing Unit
DES	: Data Encryption Standard
DLP	: Discrete Logarithm Problem
DSA	: Digital Signature Algorithm
DSP	: Digital Signal Processing
ECC	: Elliptic Curve Cryptography
FPGA	: Field Programmable Gate Array
Hz	: Hertz
LUT	: Look-up Table
RAM	: Random Access Memory
ROM	: Read Only Memory
RSA	: Ron-Rivest-Adleman
TDES	: Triple Data Encryption Standard
VLSI	: Very Large Scale Integration
W	: Watt

LIST OF FIGURES

<u>Figure No:</u>	<u>Page</u>
1.1 The picture of scytale.	2
1.2 Overview of the field of cryptology.	2
1.3 The illustration of symmetric key operations.	4
1.4 The illustration of asymmetric key operations.	5
2.1 Internal structure of FPGAs.	8
2.2 Parameter trade-off of hardware designs.	10
3.1 $y^2 = x^3 - 3x + 3$.	13
3.2 Point addition on an elliptic curve.	15
3.3 Point doubling on an elliptic curve.	16
4.1 Scalar Multiplication Algorithm.	19
4.2 Hardware Implementation of ECC Core.	20
4.3 Addition and Subtraction Module.	21
4.4 Accumulator of DSP slice.	22
4.5 Disordered example of multiplication.	22
4.6 Ordered example of multiplication.	23
4.7 Straightforward 256x256 bit multiplication with 24x17 bit DSP slices.	24
4.8 Proposed 256x256 bit multiplication by using 24x17 bit DSP slices.	24
4.9 Structure of P256 modular reduction.	25
4.10 Modular Reduction Module.	25
4.11 Modular Reduction specified for the NIST curve P256.	26
4.12 Modular Inversion Module.	27
4.13 Binary Extended Euclidean in Hardware.	27
4.14 Pipelined point addition schedule.	28
4.15 Pipelined point doubling schedule.	28

LIST OF TABLES

<u>Table No:</u>		<u>Page</u>
4.1	Comparison of scalar multiplication over P256.	29



1. INTRODUCTION

Information is one of the most important things in people's, even nations, lives. In time, People became interested in hiding or protecting information. After the invention of transistor, the development of the electronic systems and information technology in the last decades, has also increased the requirement for information security. In the last 50 years, various of algorithms and methods has been arisen for electronic systems. In this chapter, some information about the cryptography and its historical background.

1.1. Cryptography

Since the ancient times, secure and secret communication has been essential. People have attempted to conceal the core of crucial messages while adversaries had been wishing to be able to decipher the secret information contained in the ciphered messages.

Primitive means for securing military secrets were established in ancient times. During military battles, the Greeks utilized a tool called scytale which is shown in Fig. 1.1 [Web 2, 2020], which is shaped like a cylinder for removing a paper around it to write and recover hidden messages [Dinu, 2017]. The Caesar cipher is employed by Roman military troops for covert communication in the following years. It uses a straightforward letter replacement mechanism. A text can be decoded by substituting each letter in the message with a different letter and vice versa. The transition from classical encryption to modern cryptography began in the first part of the twentieth century. One of the most well-known military applications of contemporary cryptography is the Enigma machine. After World War II, both cryptography and cryptanalysis continued to evolve in parallel. The majority of early applications of contemporary cryptography were for military purposes. Cryptography has begun to play a critical part in our lives, as the heart of information security and communication. Modern cryptographic approaches are now employed in a wide range of industries, including banking, trade, and industry.



Figure 1.1: The picture of scytale.

Cryptology is the branch of science that examines the secure transmission and storage of information [Paar and Pelzl, 2009]. Cryptography and cryptanalysis are the two basic components of cryptology. The hierarchy is shown in Fig. 1.2 [Shawkat, 2016]. Algorithms and protocols in cryptography are mathematical strategies for securing information, such as data integrity, confidentiality, and authentication. The goal of cryptanalysis, on the other hand, is to use mathematical approaches and side channel techniques to uncover flaws in cryptographic algorithms and retrieve hidden information.

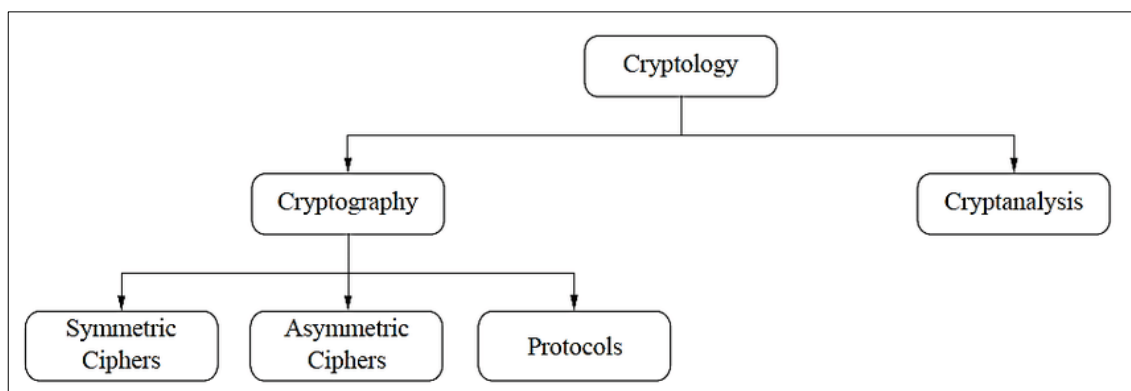


Figure 1.2: Overview of the field of cryptology.

Plain text is the information source in cryptography, and it is the one that needs to be secured. The occurrence text is referred to as cipher text once the plain text has been processed. Encryption is the process of employing a key to convert plain text

into cipher text, which is usually a meaningless form of the communication. Encryption's inverse operation is known as decryption. Using the key, decryption recovers the original form of the information (plain text) from encrypted text.

Symmetric and asymmetric ciphers are the two most used cipher methods. There are a few key distinctions between these algorithms. One of them is that symmetric algorithms use the same key for encryption and decryption, but asymmetric algorithms employ two keys, one public and one private, for encryption and decryption separately. The run times are the other major distinction. Asymmetric algorithms are substantially slower than symmetric algorithms. As with asymmetric keys, a symmetric key is also referred to as a private key. The private key is the confidential information that should only be known by authorized users. Finally, because symmetric key sizes are smaller than asymmetric key sizes, they are simple to use.

1.2. Symmetric Algorithms

Symmetric key algorithms encode and decrypt using the same cryptographic keys. The encrypted text is the same size as the plain text. Stream cipher or block cipher can be used for symmetric key encryption. Stream cipher encrypt the small pieces of message, like byte or letter, per round. RC4, ChaCha20 are the most popular algorithms among the stream ciphers. On the other hand, block ciphers take a word which contains a number of bits, as a single unit. Padding is an necessary operation in block cipher in order to fulfill the plain text so that it is multiple of the block size. AES, DES, TDES are the most popular among both block ciphers and symmetric algorithms. The relation between these terms is depicted in Fig. 1.3 [Web 3, 2020].

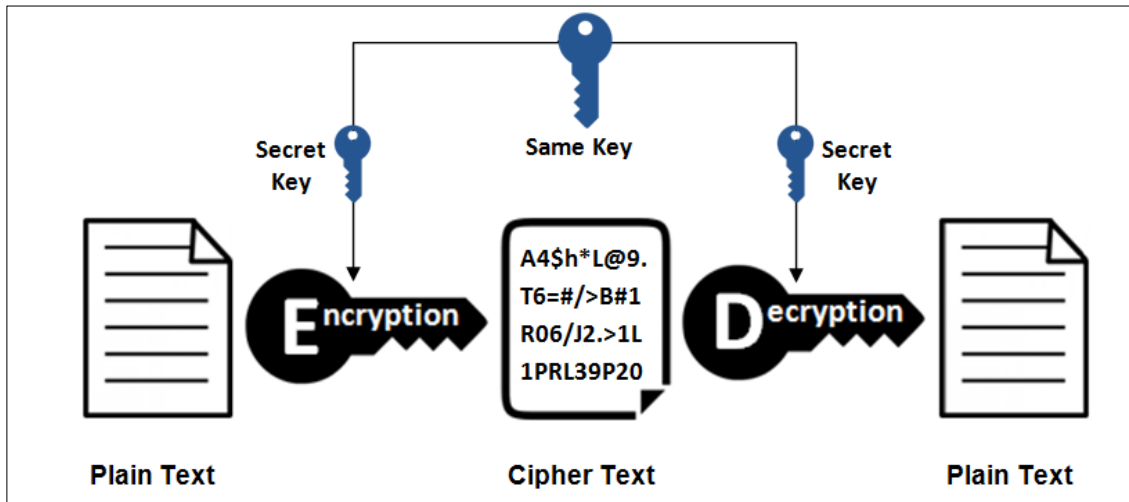


Figure 1.3: The illustration of symmetric key operations.

1.3. Asymmetric Algorithms

Public key cryptography is another name for the asymmetric key approach. To encrypt plain text, asymmetric algorithms use two keys as in Fig. 1.4 [Web 3, 2020]. Different key-exchange methods are used to exchange secret keys over a network. Anyone who wants to encrypt a message can get a public key for free. The second private key must remain hidden. A message that has been encrypted with the public key can only be decrypted with the private key, and vice versa.

Asymmetric key algorithms have a far better security of information comparing to symmetric ones. Asymmetric key algorithms are mostly used in communication channels, digital signature and verification over the Internet. Some popular asymmetric key algorithms are Elgamal, RSA, DSA and Elliptic Curve Cryptosystem.

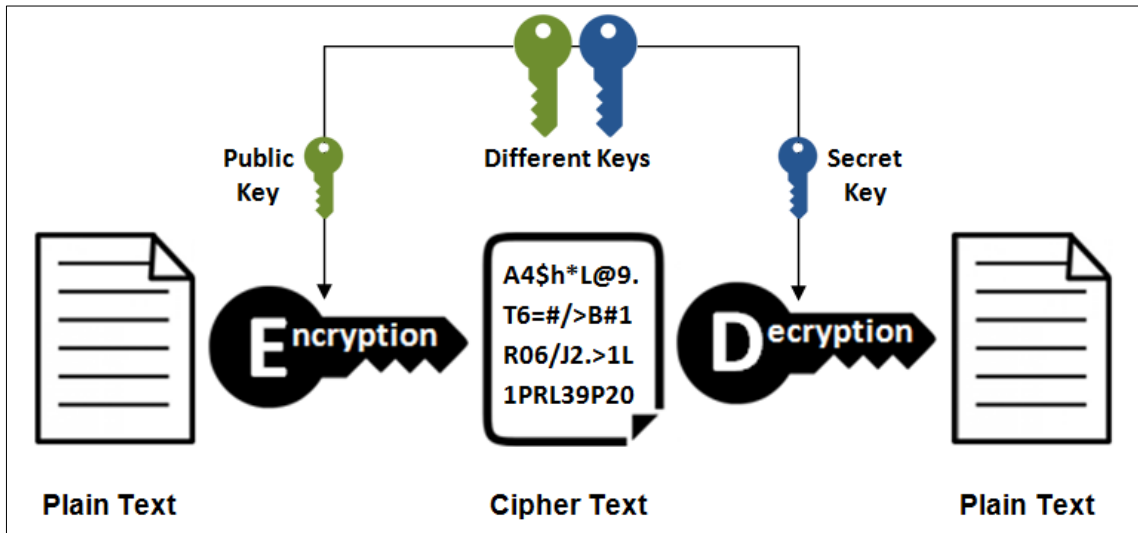


Figure 1.4: The illustration of asymmetric key operations.

1.4. Motivation

Whitfield Diffie and Martin Hellman, who published their equation in 1976, came up with the concept of an asymmetric key cryptosystem. They discovered a simple key-exchange system that they could use across public channels. Following this experiment, Massachusetts Institute of Technology's Ron Rivest, Adi Shamir, and Leonard Adleman attempted multiple times via secure data transmissions to develop a one-way function that was difficult to reverse. They published a concept called RSA in 1977. It was the first public-key cryptosystem, and it is still frequently used today.

Ever since the RSA algorithm has been invented, the fields of its usage have been increased in network and electronic devices. It has a widely usage in the industry. However, there is a big disadvantage of RSA which is computational challenge. It requires a lot resources and relatively slow. In the 80's, two different mathematicians Koblitz and Miller have suggested Elliptic Curve Cryptosystem. It has capability of doing anything that RSA can with less resources. It is also considered faster and more secure than RSA with smaller key sizes [Mahto and Yadav, 2017].

In the recent years, ECC has been increasing its popularity over RSA. Most of the users who used RSA in the past, have decided to adapt their systems to ECC. In the light of these data, ECC algorithm is worth to implement. They can also be used in different areas such as key exchange, digital signatures and encryption.

Implementing cryptographic methods while satisfying a set of restrictions based on use case scenarios can be difficult, and it necessitates understanding of various fields such as cryptography, computer science, and electronics engineering. The goal of this project is to create a fast and area efficient implementation of an ECC algorithm for digital designs.

1.5. Contributions of the Thesis

In modern cryptographic systems, ECC has been increasing its popularity over RSA. In recent times, ECC is more preferred because of its key length, implementation of simplicity compared to RSA and security aspects. Therefore, ECC cryptography has been implemented in this thesis.

ECC has several operations and all of these operations contain common arithmetic which are addition, subtraction or multiplication. So that, these operations should be enhanced to improve its performance. There are different aspects of improvements, in terms of area and performance, in digital design systems. For common usage, our goal is to make it balanced as far as possible.

In our system, multiplication, which is most used operation in ECC, has been focused to improve. Therefore, DSPs are used for this module. It is capable of multiplying 256 bit integers with 4 clocks at 90 MHz. The reason of utilization of inherent chain of DSPs are to hold the number of LUT at minimum and make multiplication fast. We also scratch a scheduling diagram to obtain a pipelined system.

2. BACKGROUND

In this chapter, some information will be given about digital design circuits such as FPGA or ASIC, and some metrics will be explained. Moreover, their advantages and disadvantages will be described.

2.1. Digital Design Circuits

Devices that contain electronic circuits, are everywhere in our lives and after the invention of transistor, they have been increasing their development process significantly. Integrated circuits, often known as IC, were invented in the early 1960s by embedding a few transistors onto silicon. The number of transistors into silicon were increasing rapidly. CPU, ROM, RAM, and other glue logic were merged into a single chip using Very Large Scale Integration (VLSI) technology, which combined thousands of transistors. Application-Specific Integrated Circuits (ASICs) are integrated circuits that have been built specifically for a certain application. Because its digital circuitry is permanently built by connected gates in silicon, this circuit is unchangeable after fabrication.

On the other hand, there is a requirement of relatively flexible digital circuits for development purposes. Xilinx was created in the 1980s and was the first company to provide a programmable logic device known as a Field Programmable Gate Array (FPGA). Thousands of Configurable Logic Blocks (CLBs) with customizable interconnections make up an FPGA. As a result, after production, an FPGA can be reprogrammed in the field and utilized as a development tool. It can also be used for a variety of purposes. Fig. 2.1 [Web 1, 2021] shows the primary structure and components of an FPGA.

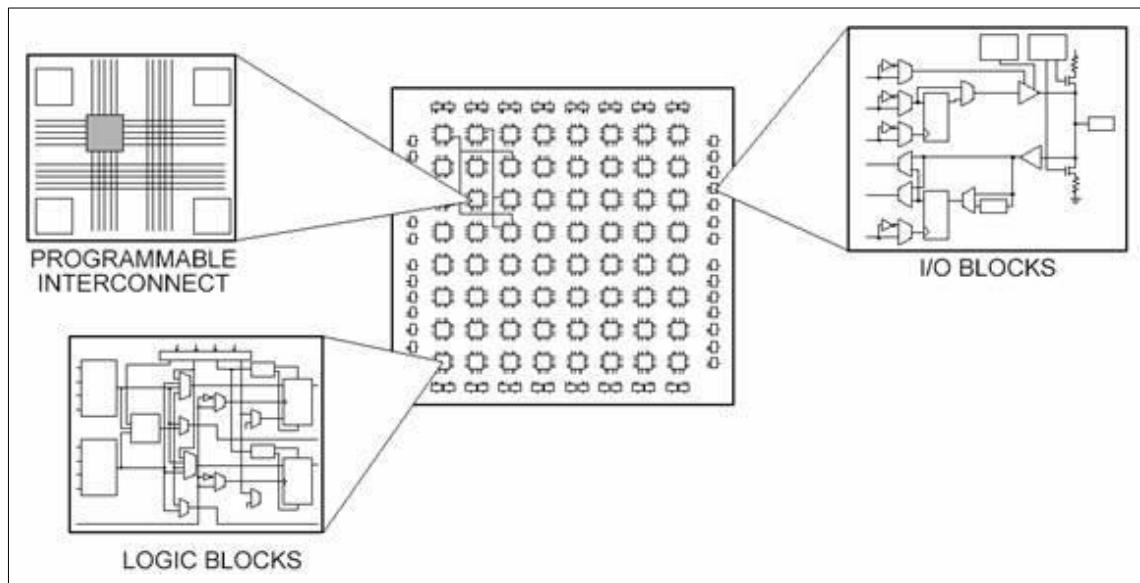


Figure 2.1: Internal structure of FPGAs.

In modern electronic technology, logic blocks configurations or interconnects of block can be described through hardware design flow. Hardware description languages (HDL) such as VHDL, Verilog, and others specify an FPGA's behavior. Working with huge structures is a good fit for HDL. Rather than drawing each piece by hand, they can be used to express high-level functional behavior. One of the primary differences between FPGA and ASIC is that FPGA may be reprogrammed for multiple applications, but ASIC can only be utilized for one. ASICs, on the other hand, are more energy efficient and perform better than FPGAs.

An FPGA is utilized as the target platform for implementations in this study since it is capable of prototyping various designs.

2.2. Field Programmable Gate Arrays

As shown in Fig. 2.1, a field programmable gate array (FPGA) is made up of an array of configurable logic blocks (CLBs) surrounded by programmable I/O blocks and coupled by programmable interconnections. An FPGA can include anything from 64 to tens of thousands of logic blocks and many more flip-flops. The logic blocks in most FPGAs do not have 100 percent connectivity. Instead, the logic is placed and routed on the device by related software.

FPGA architectures can be divided into two categories. Large logic blocks, frequently with four or six look-up tables and four flip-flops, make up coarse-grained

architectures. Fine-grained architectures are made up of a lot of small, simple logic components. The underlying process technology utilized to build the device is another variation between the architectures. The highest-density FPGAs are currently constructed utilizing SRAM technology, which is comparable to that used in microprocessors. Anti-fuses, which are one-time programmable, are another typical process technology.

The FPGA part used, virtex devices, has a flexible, regular architecture that includes CLBs which are surrounded by programmable input and output ports, all of which are coupled by a rich hierarchy of fast, varied routing resources. Virtex FPGAs have a coarse-grained design, are based on SRAM, and may be changed by putting configuration data into internal memory cells.

2.3. Metric Parameters

The following metrics should be considered as main constraint parameters for any applications. There is a common thought that, the design parameters have trade-off between them as shown in Fig. 2.2 [Ramsey, Ravi, William and Winsor, 2009]. By meaning that, if the designer wants to make a fast design then resource usage shall be much more than it could be. Therefore, these parameters should be balanced for a good design. An FPGA design developer should also compare his/her design by checking these parameters of others.

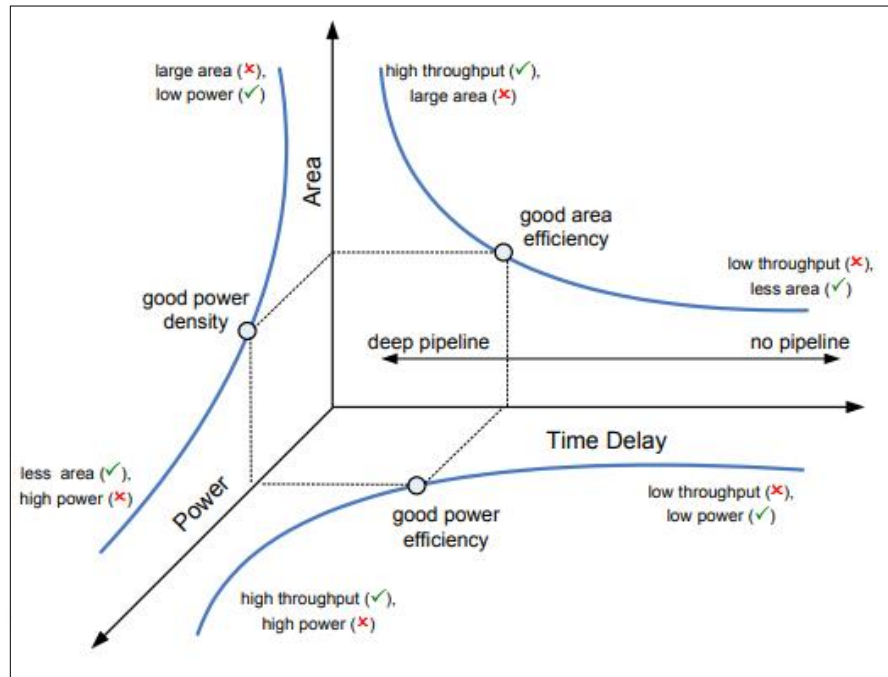


Figure 2.2: Parameter trade-off of hardware designs.

2.3.1. Area

FPGA chips on the development boards have different amount of resources. Main resource components are look-up tables (LUTs) and flip-flops (FF). These components have a big impact on the power consumption of the chips. In some FPGA boards, there are also some utilizations such as DSP and Block RAM. These components are designed for common usages like accumulation and storage. They also consume low energy.

2.3.2. Power Consumption

It is described as the energy consumption of a design. Power consumption is mostly represented mW. This metric has directly proportional with area. If resource usage is increased, then system would consume more power. More power also means more heat. Therefore, cooling system needs to stabilize the heat at nominal values. More power might cost more than balanced systems.

The total power usage over a period is equal to the energy consumption. For battery-powered devices, this metric is very important. The energy and power

consumption of an electronic circuit is highly related to the area usage and manufacturing technology.

2.3.3. Throughput

This metric is another indicator of the implementation's success. It's similar to the latency metric. However, it should be calculated in terms of second. In general, it is the rate of production. The expression is expressed in bits per second (bps).

2.3.4. Latency

From beginning to end, the length of time required to complete an operation. It's also known as the time between valid input and output for a particular process. It can be expressed in either clock cycles or seconds.

3. ELLIPTIC CURVE CRYPTOGRAPHY

In this section, mathematical background of ECC, previous works about ECC on FPGA and general view of this implementation will be described. First of all, we describe the hard problem behind this cryptosystems as well as the mathematical concepts of elliptic curves in order to understand how they work.

3.1. Mathematical Background

Koblitz [Koblitz, 1987] and Miller [Miller, 1985] were the first to suggest elliptic curves for use in public-key cryptography (PKC). To propose an elliptic curve-based public key cryptosystem, we first define the elliptic curve group and then the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is identical to the Discrete Logarithm Problem (DLP) for any cyclic group.

The mathematical background of ECC will be briefly given in the following part, which is crucial to this work. Elliptic Curves are multipurpose mathematical objects that can be used for encryption, decryption, digital signatures, and key exchange. All of these processes are crucial in cryptography, which is why it's called as cryptosystem. Note that only $GF(p)$ will be the focus of this research, as $GF(2^m)$ is deprecated by NIST due to its low usage [Chen, Moody, Regenscheid, Randall, 2019].

To ensure the system's security, ECC constructions rely on group theory. (x, y) coordinates are used to represent an elliptic curve over the prime field $GF(p)$. The following equation, known as the Weierstrass equation, should be satisfied by these coordinates.

$$y^2 = x^3 + ax^2 + b \quad (3.1)$$

where $a, b \in GF(p)$ and $4a^3 + 27b^2 \neq 0$ and p be a prime with $p > 3$. By doing some mathematical computations on the group of points defined on this curve, we can obtain (x, y) points. The addition $R = P + Q$ and the doubling $R = 2P$, $P = Q$ are the two types of arithmetic operations. Scalar multiplication ($Q = kP$) is the most basic

process in ECC, and it can be calculated using Algorithm 1 or by simply adding repeated points (i.e. $P + P + \dots + P$) for k -times.

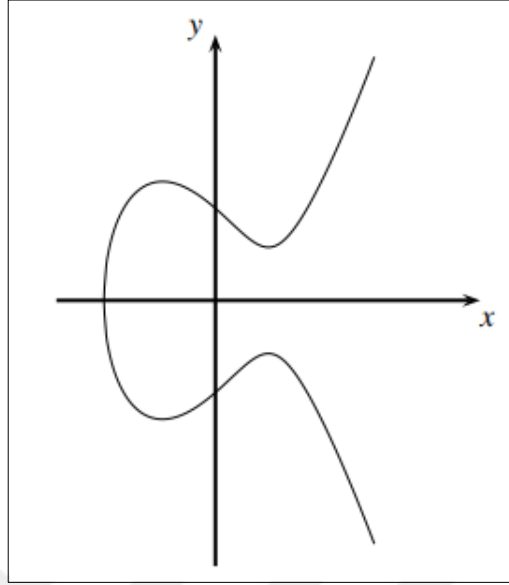


Figure 3.1: $y^2 = x^3 - 3x + 3$.

An example curve is shown in Fig. 3.1 [Paar and Pelzl, 2009]. These formulae only use the two basic algebraic operations which are addition and doubling. These operations are applicable to any field, not just the field of real numbers. We can use the curve equation from earlier, but this time it will be considered over prime fields $GF(p)$ instead of actual integers. The following analytical formulations for the group operation result from this.

$$\begin{aligned} x_3 &= s^2 - x_1 - x_2 \bmod p \\ y_3 &= s(x_1 - x_3) - y_1 \bmod p \end{aligned} \tag{3.2}$$

where

$$\begin{aligned} s &= (y_2 - y_1)/(x_2 - x_1) \bmod p; \text{ if } P \neq Q (\text{point addition}) \\ s &= (3x_1^2 + a)/(2y_1) \bmod p; \text{ if } P = Q (\text{point doubling}) \end{aligned} \tag{3.3}$$

The parameter s is the slope of the line through P and Q , or the slope of the tangent through P in the case of point doubling. And note that, these formulae

represented in affine coordinate system so that it requires division operation. In order to get rid of division operation, projective coordinate system has been selected.

3.1.1. Coordinate Systems

Because of the division operation, point addition and point doubling computations in affine coordinates are exceedingly rough. Modular inversion must be used after every addition or doubling, according to affine formulae. In projective coordinates, however, modular inversion is only performed once for every scalar multiplication. Additional add-sub and multiplication operations, on the other hand, are used every sub-steps. In projective coordinates, though, it is still considerably more worthwhile to compute. The Weierstrass equation has some projective coordinate shifts, which are represented in equation 4.

$$Y^2Z = X^3Z + a_4XZ^2 + a_6b \quad (3.4)$$

The point (X_1, Y_1, Z_1) on equation 2 corresponds to the affine point $(X_1/Z_1, Y_1/Z_1)$ when $Z_1 \neq 0$. They use a quadruple to represent the Jacobian coordinates internally (X, Y, Z, aZ^4) . The modified Jacobian coordinate system is the name for this representation. In the sections below, the algorithms for EC point addition and doubling are described.

3.1.2. Point Addition

The following is a representation of the construction process: Create a third point of intersection between the elliptic curve and the line by drawing a line across P and Q. Along the x-axis, mirror the third intersection point. By definition, this mirrored point is the point R. The point addition on an elliptic curve over the real numbers is shown in Fig. 3.2 [Paar and Pelzl, 2009].

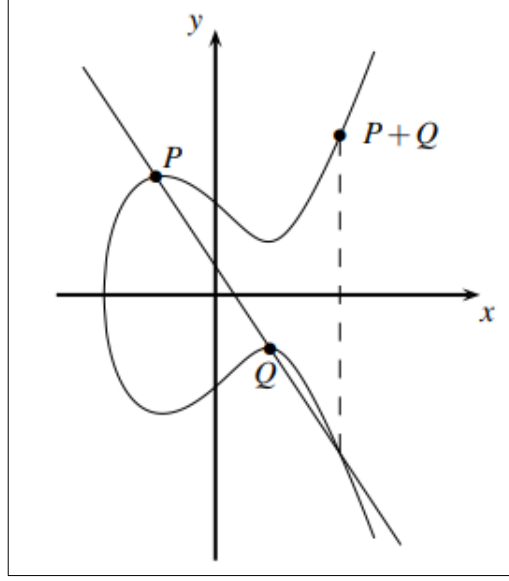


Figure 3.2: Point addition on an elliptic curve.

The point addition in projective coordinates is described in the formula below. The below formulas are represented in projective coordinates. And these parameters have dependencies between them in both addition and doubling. In order to compute point addition in affine coordinates, you'll need one modular inversion, two multiplications and six subtractions. In projective coordinates, on the other hand, fifteen multiplications and seven subtractions are required.

Assume that $P = (X_1, Y_1, Z_1, aZ_1^4)$, $Q = (X_2, Y_2, Z_2, aZ_2^4)$ and $P + Q = R = (X_3, Y_3, Z_3, aZ_3^4)$.

$$\begin{aligned}
 A &= Y_2 Z_1 - Y_1 Z_2, \\
 B &= X_2 Z_1 - X_1 Z_2, \\
 C &= A^2 Z_1 Z_2 - B^3 - 2B^2 X_1 Z_2, \\
 X_3 &= BC, Z_3 = B^3 Z_1 Z_2, \\
 Y_3 &= A(B^2 X_1 Z_2 - C) - B^3 Y_1 Z_2
 \end{aligned} \tag{3.5}$$

3.1.3. Point Doubling

When we compute $P + Q$, yet $P = Q$, we have a problem. As a result, $R = P + P = 2P$ can be written. Here, we'll need a somewhat different approach. We establish a second point of intersection between this line and the elliptic curve by drawing the tangent line through P . Along the x-axis, we mirror the position of the second

intersection. The doubling's outcome R is this mirrored point. Fig. 3.3 [Paar and Pelzl, 2009] depicts the real-number doubling of a point on an elliptic curve.

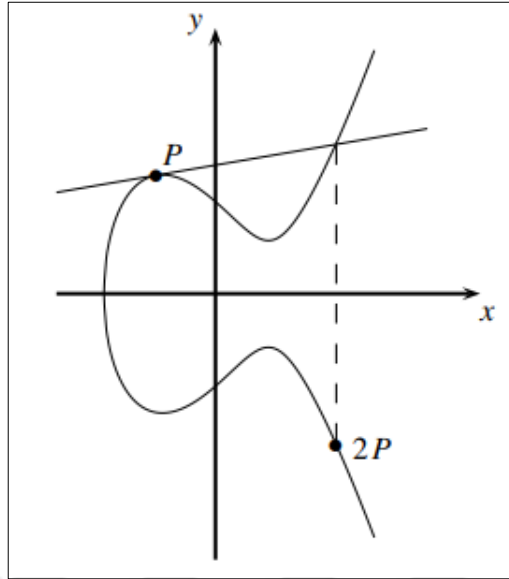


Figure 3.3: Point doubling on an elliptic curve.

The amount of multiplication operations in point doubling are more than in point addition, although addition and subtraction operations are lower. Point doubling should be done in affine coordinates by performing one modular inversion, six multiplication and four subtraction operations. For the other, it's only eighteen multiplications and four subtractions.

Assume that $P = (X_1, Y_1, Z_1, aZ_1^4)$ and $P + P = R = (X_3, Y_3, Z_3, aZ_3^4)$.

$$\begin{aligned}
 A &= a_4 Z_1^2 - 3X_1^2, \\
 B &= Y_1 Z_1, C = X_1 Y_1 B, \\
 D &= A^2 - 8C, \\
 X_3 &= 2BD, Z_3 = 8B^3, \\
 Y_3 &= A(4C - D) - 8Y_1^2 B^2
 \end{aligned} \tag{3.6}$$

Please note that, number of arithmetic operations, which mentioned above, are calculated with two input arithmetic units. Parameter multiplication by simple integers is also counted.

3.1.4. Discrete Logarithm Problem

The Elliptic Curve Discrete Logarithm Problem (ECDLP) is now described; the ECC protocols are based on its difficulty. Let G be any finite cycle group. Denote its group operation by multiplication and identity element by 1. Let b be any element of G . For any positive integer k , the expression b^k denotes the product of b with itself k times: $b^k = b.b.b \dots b$.

Similarly, let b^{-k} denote the product of b^{-1} with itself k times. For $k = 0$, the k_{th} power is the identity: $b^0 = 1$. Let a also be an element of G . An integer k that solves the equation $b^k = a$ is termed a discrete logarithm of a to the base b . One writes $k = \log_b a$.

3.2. Related Works

Elliptic Curve Cryptography (ECC) [Koblitz, 1987], [Miller, 1985] is a public-key cryptographic system which lets the user obtain equivalent security with less key sizes compared to a well known public-key crypto algorithm, RSA [Lenstra, Verheul, 2001]. Without compromising security aspects, acquiring high throughput, low resource usage and low power consumption have become of great importance. Since improving the performance of ECC is essential, several methods have been proposed to speed up the computation of the ECC. As advised by the National Institute of Standards and Technology (NIST), one of these ways is to use Mersenne primes, which make the modular reduction operation easier [National Institute of Standards and Technology, 2013]. However, we utilize the P256 curve, which is the most often used and is comparable to the RSA 3072 key size.

A crucial procedure in ECC is scalar multiplication ($Q = kP$), which comprises point additions and doublings. Point P is added to itself ($k-1$) times on the same curve to calculate point Q . Scalar k is a private scalar number, while points P and Q are public points. Because of the Discrete Logarithm Problem (DLP), obtaining the value of k given the values of P and Q is computationally hard [McCurley, 1990].

In order to accelerate these units various works have been studied. The basis of accelerating these operations depends on fast multiplication. In [Marzouqi, Al-Qutayri, Salah, Schinianakis, Stouraitis, 2016], an extensive

pipelining technique for Karatsuba- Ofman method is employed for high throughput. Another fast method for multiplication is Toom-Cook [Ding, Li, Gu, 2019]. This method is considerably fast but its resource usage is high. In [Mehrabi, Doche, Jolfaei, 2020], [Asif, Hossain, Kong, Abdul, 2018], [Matutino, Araujo, Sousa, Chaves, 2017], RNS based point multiplication has been studied to fasten common arithmetic operations for P256 prime. [Guillermin, 2010] is also implemented using RNS based arithmetic units, moreover it has been designed by using Montgomery ladder. Then parallelism and pipelining lead to high performance. In addition to these implementations there are some designs which contain Digital Signal Processing (DSP) units to enhance multiplication and addition as well. [Mcivor, Mcloone, Mccanny, 2006], [Guneysu, Paar, 2008], [Varchola, Guneysu, Mischke, Microecc, 2011], [Ananyi, Alrimeih, Rakhmatov, 2009], [Alrimeih, Rakhmatov, 2014] and [Wu, 2015] are example of using DSP in their designs. Most of them are committed to a single NIST prime field. In industrial applications, generally only one NIST prime field is used where a prime field of ECC is required.

In this paper, we describe the design and implementation of a GF (p) prime field of ECC. By simply replacing the modular reduction module, other curves with smaller key sizes can be accommodated as well. Multiplication, scheduling, and pipelining are the major areas of attention for us. Two multiplications, one addition, three modular reduction, and one modular inversion units were used to create an efficient schedule. The quantity of DSP units is the primary disadvantage, thus we've tried to optimize it to keep it portable. For targets with varying resource requirements, the schedule and pipelining can be changed.

4. IMPLEMENTATIONS OF ELLIPTIC CURVE CRYPTOGRAPHY

Fig. 4.2 illustrates the proposed circuit design used for ECC modular arithmetic operations. It has the following four units: Addition and subtraction, multiplication, modular reduction, and modular inversion.

Modular multiplication is performed with addition and subtraction, multiplication, and modular reduction units. Because ECC scalar point multiplication necessitates a large number of modular multiplications, the latencies of these three units are critical. Addition and subtraction are faster than multiplication. Moreover, it is even slower than the modular reduction for the P256 curve because the reduction is performed modulo the special prime $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. As a result, hardware resources are primarily employed to increase the multiplier unit's performance such that it has a similar latency to the other two units.

```
1: Input: P (X, Y ), A(X1, Y1), B(X2, Y2), int(k)
2: Output: Q = k.P, Q(0, 0)
3: for i = 0 : l - 1 do
4:   A = P + Q
5:   B = 2P
6:   if k(0) = 1 then
7:     Q = A
8:   end if
9:   P = B
10:  k = k >> 1
11: end for
```

Figure 4.1: Scalar Multiplication Algorithm.

The memory unit in Fig. 4.2 is made up of block RAMs and flip-flops. We tried to avoid multiplexers in order to save space and boost frequency, therefore block RAM is the reasonable approach in this case. Each arithmetic module has its own fixed-size block RAM cell. Fig. 4.14 and Fig. 4.15 show how their sizes are determined. Each module specifies how many distinct types of data will be used for each activity. The primary function of the address controller is to direct matching data to the arithmetic unit. Its sole function is to change the block RAM address. As a

result, we don't need a multiplexer for data, but we do for addresses with a considerably lower width than data.

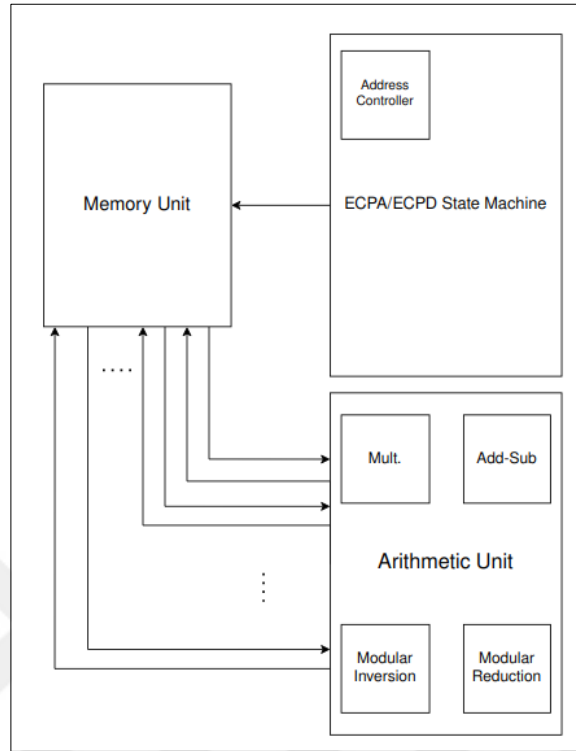


Figure 4.2: Hardware Implementation of ECC Core.

Each block of RAM has its own cells containing all of the parameters required for point addition and doubling. Because each module has two inputs, we employ dual port block RAM. Module inputs get data from these dual port inputs and where the result will be used in the next situation, it is written to the relevant block RAM of that module. For example; if first stage of the multiplication output will be used in the reduction module for the next stage, then the result is directed to regarding block RAM and for the next stage address controller will pull it where it is kept.

4.1. Addition and Subtraction

In comparison to other units, addition and subtraction are simple to implement. These two functions require less complexity thanks to summing operations and carry chain units in most advanced FPGAs. Both arithmetic operations are implemented in a single IP (Intellectual Property). As previously stated, the multiplication unit is mostly responsible for delay. As a result, rather than optimizing for time complexity,

addition and subtraction units are optimized for area complexity. It also has a comparable latency to the multiplication unit. This unit's architecture is scalable, allowing it to be used in a variety of prime areas.

In addition and subtraction module, all inputs should be fed properly in order to obtain correct result. In concurrent addition or subtraction operations, carry_i signal should be given as '0'. If the desired operation is addition, then twos_comp signal should be '0', otherwise '1'. Another important note for concurrent operations that, carry_o signal should be connected to carry_i signal in order to convey the carry chain. The last carry_o signal is also important for us because it is used in modular reduction module if the result exceeds 256 bit length. Our data out signal has 257 bit length because of signed operations.

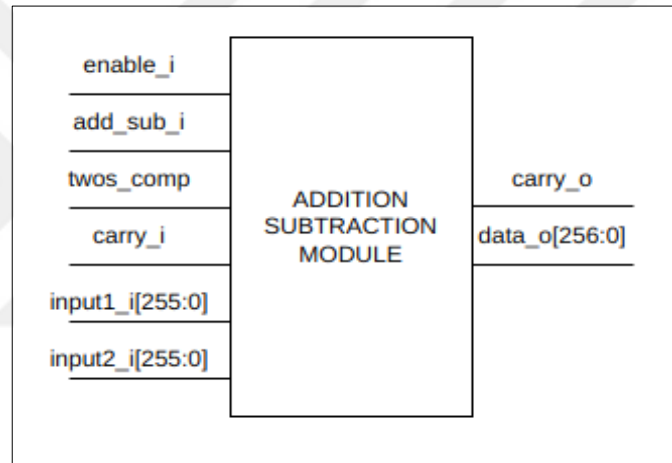


Figure 4.3: Addition and Subtraction Module.

4.2. Multiplication

In many cryptographic methods, big integer multiplication is a common operation. In embedded systems, performing this process is a computationally demanding effort. As previously indicated, certain multiplication designs for hardware platforms exist, however they are insufficiently performant for large-scale multiplication operations. DSP48E1 slices, which are available on Xilinx 7 Series FPGAs, are used in our design.

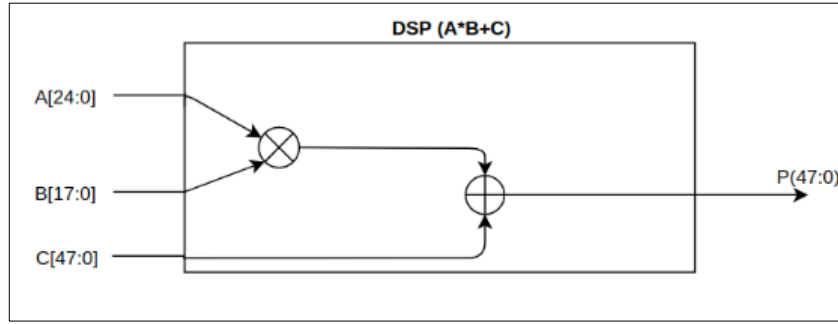


Figure 4.4: Accumulator of DSP slice.

We created a custom architecture to execute a 256x256 multiplication using these slices. The first input is divided into 24 bits, and the second input is divided into 17 bits in this design. The DSP slices then calculate the 24x17 bit partial products. The 48 bit C ports of the DSP slices are used to assemble the partial products.

$$\begin{aligned} A &= A(255:232) \dots A(47:24) \parallel A(23:0) \\ B &= B(255:239) \dots B(33:17) \parallel B(16:0) \end{aligned} \quad (4.1)$$

Fig. 4.7 shows how to create a 256x256 bit product from smaller 24x17 partial products in a simple way. DSP slices compute and merge the partial products. As can be seen, the partial products should be adjusted suitably before being combined.

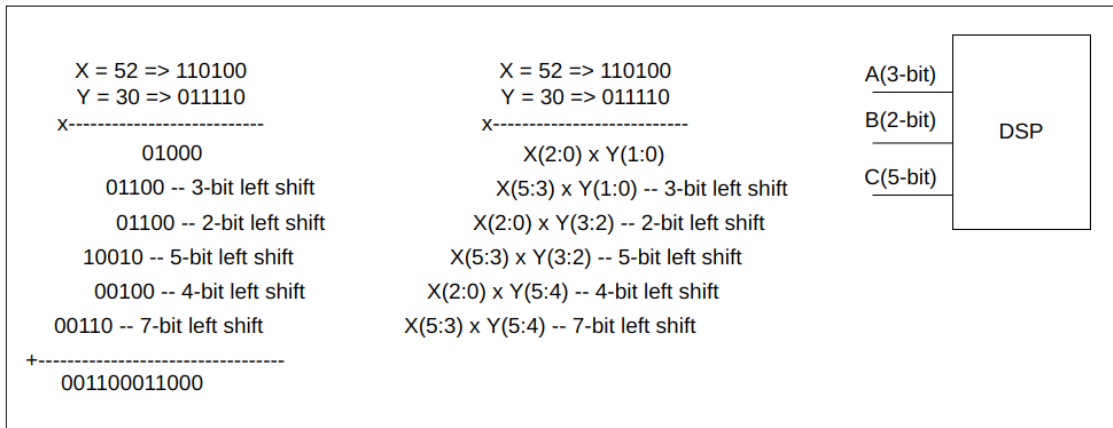


Figure 4.5: Disordered example of multiplication.

As an example of multiplication, we can consider 52(6-bit)x30(6-bit) operations. For simplicity, let's assume that the multiply inputs of the DSP are 3-bit and 2-bit, and the addition input is 5-bit. When calculating shift amounts, the next partial product is shifted as the shifted multiplier. For example, in $X(5:3) \times Y(1:0)$

multiplication, the more the multiplier is shifted, the more the partial product is shifted to the next line. If we look at the example this is 3-bit.

In Fig. 4.6, left swipes are sorted from smallest to largest. The order of them does not matter because it is the addition operation.

The C input comes into play here for the sum of the partial products. The striped intersections shown in the figure are summed up and written as the partial product of the sum, the non-intersecting parts are written directly into the result. This operation occurs 44 times in each state, and for a 256x256 bit operation, these 4 states are finished. Finally, the results of these binary addition operations are added in one go and the result of the multiplication operation is found.

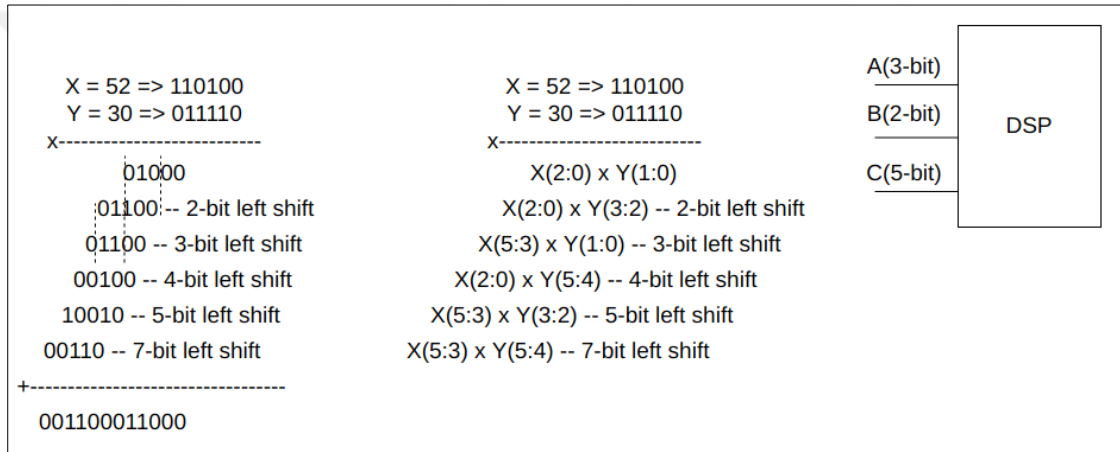


Figure 4.6: Ordered example of multiplication.

The C input comes into play here for the sum of the partial products. The striped intersections shown in the figure are summed up and written as the partial product of the sum, the non-intersecting parts are written directly into the result. This operation occurs 44 times in each state, and for a 256x256 bit operation, these 4 states are finished. Finally, the results of these binary addition operations are added in one go and the result of the multiplication operation is found.

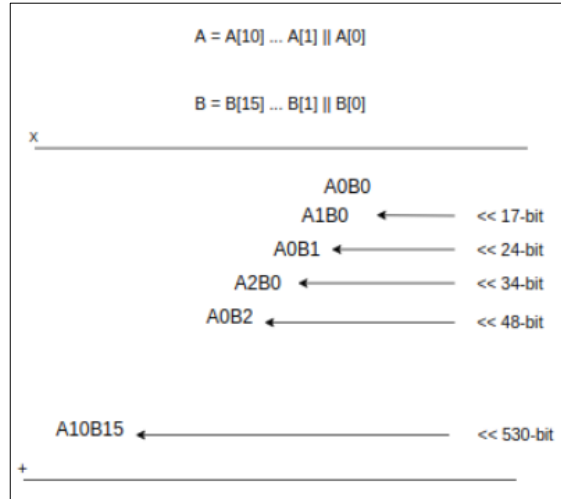


Figure 4.7: Straightforward 256x256 bit multiplication with 24x17 bit DSP slices.

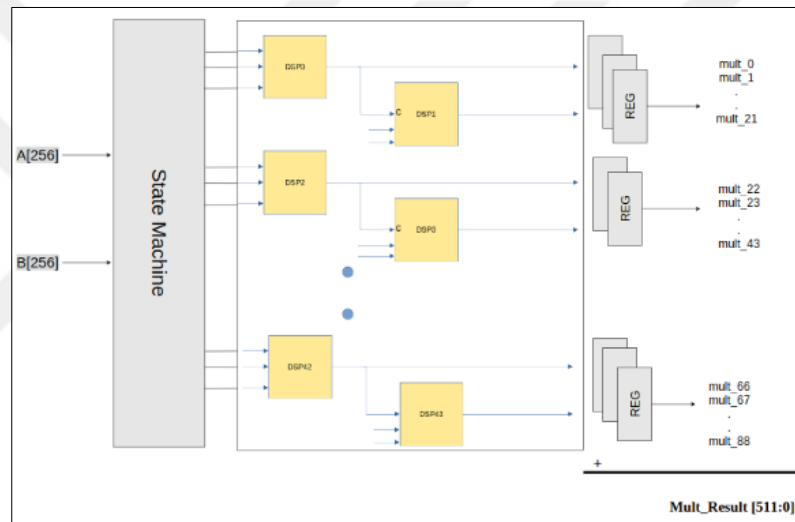


Figure 4.8: Proposed 256x256 bit multiplication by using 24x17 bit DSP slices.

Unfortunately, the straightforward implementation in Fig. 4.7 has a very large critical path. Thus, we propose the architecture in Fig. 4.8 to compute and accumulate the partial products of 256x256 bit multiplication. A state machine was written and a maximum of two DSP is connected at once. The intermediate results are registered and then they are added in 3 steps. The design uses 44 DSP slices and produces 256x256 bit multiplication in 5 clocks.

4.3. Modular Reduction

Algorithm 2 presents the modular reduction for the prime field used in the P256 curve. As seen, four addition, four subtraction and two scaling are required. The result should be in the range of $-4p < x < 5p$. x is defined in algorithm 2.

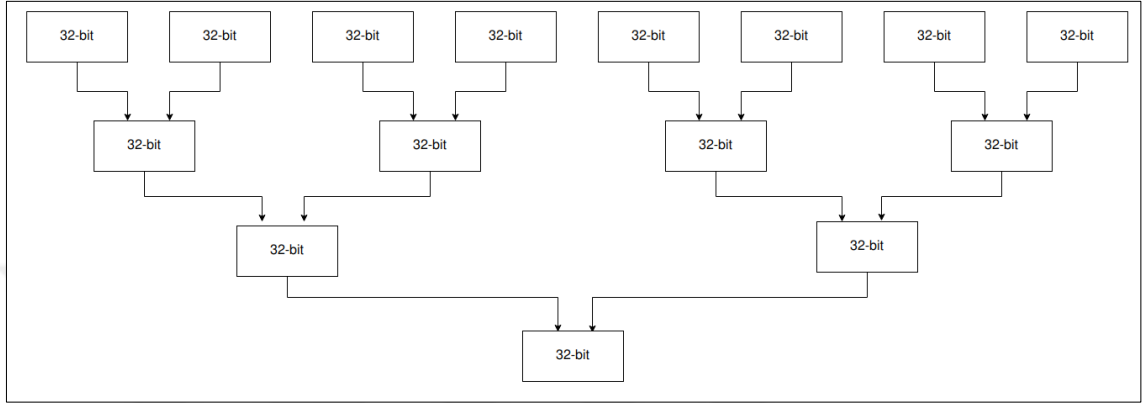


Figure 4.9: Structure of P256 modular reduction.

The modular reduction unit is likewise designed to have the same latency as the 256x256 multiplication unit. As a result, we were able to reduce resource consumption. The reduction unit is based on the Wallace tree, which is a well-known model. Fig. 4.9 depicts a modular reduction unit with the Wallace tree. This illustration depicts only one reduction module cell. There are eight cells in the P256 curve.

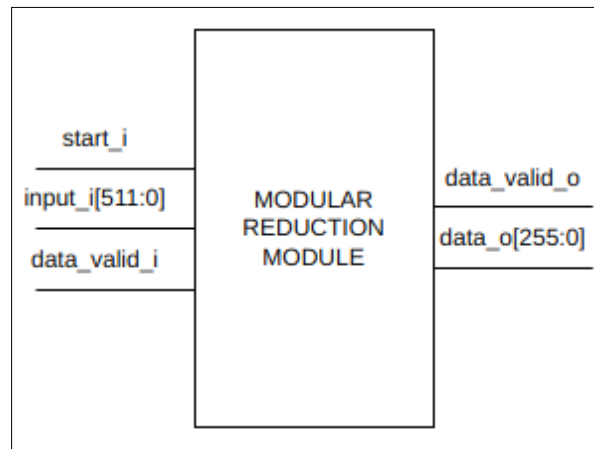


Figure 4.10: Modular Reduction Module.

```

1: Input: 512 bit nonnegative integer  $x = (x_{15}, \dots, x_2, x_1, x_0)$  in base  $2^{32}$ .
2: Output:  $x \bmod 2^{256} - 2^{224} + 2^{192} - 1$ 
3: Concatenate 32-bit partials to 256-bit integer
    $z_1 = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0),$ 
    $z_2 = (x_{15}, x_{14}, x_{13}, x_{12}, x_{11}, 0, 0, 0),$ 
    $z_3 = (0, x_{15}, x_{14}, x_{13}, x_{12}, 0, 0, 0),$ 
    $z_4 = (x_{15}, x_{14}, 0, 0, 0, x_{10}, x_9, x_8),$ 
    $z_5 = (x_8, x_{13}, x_{15}, x_{14}, x_{13}, x_{11}, x_{10}, x_9),$ 
    $z_6 = (x_{10}, x_8, 0, 0, 0, x_{13}, x_{12}, x_{11}),$ 
    $z_7 = (x_{11}, x_9, 0, 0, x_{15}, x_{14}, x_{13}, x_{12}),$ 
    $z_8 = (x_{12}, 0, x_{10}, x_9, x_8, x_{15}, x_{14}, x_{13}),$ 
    $z_9 = (x_{13}, 0, x_{11}, x_{10}, x_9, 0, x_{15}, x_{14})$ 
4: Compute  $x = z_1 + 2z_2 + 2z_3 + z_4 + z_5 - z_6 - z_7 - z_8 - z_9 \bmod 2^{256} - 2^{224} + 2^{192} - 1$ 

```

Figure 4.11: Modular reduction specified for the NIST curve P256.

In modular reduction module, our input length is 512 bit because of multiplication output length. Output of addition and subtraction module is padded with '0' before conveying to modular reduction module.

4.4. Modular Inversion

In all units, modular inverse has the longest latency. Fortunately, each scalar multiplication only requires one modular inverse. Despite the fact that it is only used once, its latency is critical to the system's throughput.

As mentioned before, modular inverse is needed for the conversion from projective coordinates (X_1, Y_1, Z_1) to affine coordinates (X, Y) as follows.

$$X = X_1(Z^{-1} \bmod P256), Y = Y_1(Z^{-1} \bmod P256) \quad (4.2)$$

To speed up modular inversion, the binary extended euclidean approach is used which is represented in Algorithm 3. Special studies [Hossain, Kong, 2015] have been conducted specifically to improve the performance of this unit. This paper is where we get it for our task [Savas, Koc, 2000]. The entire algorithm is available there. When compared to prior modular inversion studies, ours performs well in terms of resource utilization, latency, and other factors.

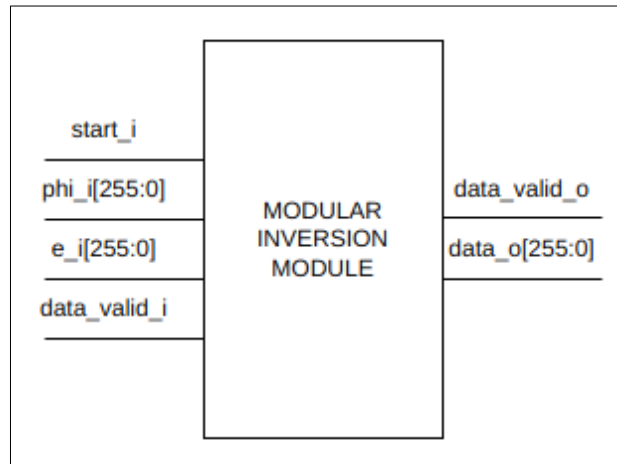


Figure 4.12: Modular Inversion Module.

```

1: Inputs: Integer numbers  $e$  and  $\varphi(N)$ 
2: Output:  $d = e^{-1} \bmod \varphi(N)$ 
3:  $u = e$ 
4:  $v = \varphi(N)$ 
5:  $A = 1$ 
6:  $B = 0$ 
7:  $C = 0$ 
8:  $D = 1$ 
9: if ( $u \neq 0$ )
10:   compare_u_and_v =  $u - v$ 
11:   while ( $u(1) = 0$ )
12:      $u = u \gg 1$ 
13:     if ( $A(0) = 0$  and  $B(0) = 0$ )
14:        $A = A \gg 1$ 
15:        $B = B \gg 1$ 
16:     else
17:        $A = (A + \varphi(N)) \gg 1$ 
18:        $B = (B - e) \gg 1$ 
19:   while ( $v(1) = 0$ )
20:     if ( $C(0) = 0$  and  $D(0) = 0$ )
21:        $C = C \gg 1$ 
22:        $D = D \gg 1$ 
23:     else
24:        $C = (C + \varphi(N)) \gg 1$ 
25:        $D = (D - e) \gg 1$ 
26:   if ( $A(0) = 0$  and  $B(0) = 0$ )
27:      $u = u - v$ 
28:      $A = A - C$ 
29:      $B = B - D$ 
30:   else
31:      $v = v - u$ 
32:      $C = C - A$ 
33:      $D = D - B$ 
34: return ( $d = C \bmod \varphi(N)$ )

```

Figure 4.13: Binary Extended Euclidean in Hardware.

4.5. Pipeline and Scheduling

Pipelining is used in our design to reduce latency. The pipelining of point addition is shown in Fig. 4.14, and the pipelining of point multiplication is shown in Fig. 4.15. The multiplication unit pipeline comprises two tiers of DSPs, as shown in Fig. 4.8.

In the arithmetic unit, we perform two multiplications, one addition, and three modular reductions. Because of the number of DSPs, the multiplication unit is our bottleneck, thus we determine other unit numbers in relation to it. For example, we could increase the number of addition and modular reduction units if we could employ four multiplications. It is entirely dependent on the hardware platform that will be used. We also opted to use a suitable number of DSPs on other platforms.

On the hardware side, state machines are used to implement and alter this pipeline mechanism. Although the number of states could be reduced, our investigation found that Fig. 4.14 and Fig. 4.15 are the best possibilities. Because the modular inversion unit is only used once for per-scalar multiplication, it is not required for intermediate procedures in projective coordinate.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1. Mult	M1=X2.Z1	M3=Y2.Z1	M5=Z1.Z2		M7=C.C	M9=A.A	M11=M7.C	M13=M5.M9	M15=M7.M8				X3=B.C		M16=S1.A			
2. Mult	M2=X1.Z2	M4=Y1.Z2	M6=Z.M2		M8=M4.C	M10=M5.C	M12=M6.M7	M14=M7.M2	Z3=M7.M10									
1. Add-Sub			C=M1-M2	A=M3-M4					A1=M11+M12		B=M13-A1		S1=M14-B				Y3=M16-M15	
1. Reduction		M1	M3	M5		M7	M9	M11	M13	M15				X3		M16		Y3
2. Reduction		M2	M4	M6		M8	M10	M12	M14	Z3								
3. Reduction				C	A					A1		B		S1				

Figure 4.14: Pipelined point addition schedule.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1. Mult	M1=X.X	B=Y.Z	M4=a.M2	C=M3.B	M7=Y.Y	M9=b.C	Z3=M8.M6	M12=M10.M7	M14=2.B		X3=M14.D		M15=A.S1			
2. Mult	M2=Z.Z	M3=X.Y	M5=3.M1	M6=B.B	M8=b.B	M10=b.M6	M11=A.A	M13=4.C								
1. Add-Sub					A=M4+M5				D=M11-M9		S1=M13-D				Y3=M15-M12	
1. Reduction		M1	B	M4	C	M7	M9	Z3	M12	M14		X3		M15		
2. Reduction		M2	M3	M5	M6	M8	M10	M11	M13							
3. Reduction						A				D		S1				Y3

Figure 4.15: Pipelined point doubling schedule.

4.6. Implementation

We focused on the area and time efficient hardware design of elliptic curve cryptosystems in this thesis. The efficiency of elliptic curve cryptosystems is determined by the efficiency of finite field operations, which are the fundamental operations. As a result, we initially studied at the most efficient hardware implementation of finite field operations over GF (p). The multiplication operation is the most essential of all the finite field operations. After that, we gave significant attention to structuring and implementing the most efficient structure possible.

Our method is implemented on the Virtex-7 FPGA platform from Xilinx. We also synthesize our design with the Xilinx Vivado 2018.3 software. The suggested design uses 7.2k slices, 44 DSP48 blocks (18x24-bit multipliers), and 12 block RAM units and runs at 90 MHz. Table 1 lists the majority of these values as well as a detailed comparison with other studies.

We used commercial software tools like Vivado to design, create, and simulate the circuits; however, real-world implementations would be preferable. We should have a prototype environment because there are numerous architectures we'd like to attempt, and we'd also like to test our circuits for debugging purposes before the final product is released. FPGAs are the ideal alternative for this requirement because of their reconfigurability.

Table 4.1. Comparison of scalar multiplication over P256.

Reference	Hardware	Area (Slice)	DSP	Freq (MHz)	Latency (ms)
Ours	Virtex-7	7.2k	44	90	0.2
[Guneyasu, Paar, 2008]	XCV4FX12	1.7k	32	490	0.495
[Wu, Wang, 2019]	XC7VX330t-3	8.4k	32	310	0.526
[Marzouqi, Qutayri, Salah, 2013]	XC5VLX50	10.2k	0	66.7	6.63
[Feng, Li, 2017]	Stratix II	14.2k ¹	64	86.9	0.16
[Mehrabi, Doche, Jolfaei, 2020]	Virtex-7	14k	560	125	0.25
[Choi, Lee, Kim, Kim, 2018]	Virtex-6	18.6k	0	121.6	0.3
[Ding, Li, Gu, 2019]	Virtex-6	7.6k*	48	291	0.054
[Awaludin, Larasati, Kim, 2021]	Virtex-7	7.3k	136	204	0.158

* Slices are estimated as LUT/4.

¹ In terms of ALM.

We, like the majority of others, focused on multiplications in our design. We concentrated on the P256 prime field, which is one of the most commonly used fields in industry, and optimized multiplication in our design. As a result, it can handle multiplications of up to 256×256 . Using modular multiplication or addition [Mehrabi, Doche, Jolfaei, 2020], [Alrimeih, Rakhmatov, 2014], [Awaludin, Larasati, Kim, 2021] is an option for these designs, however we decided to reduce the latency by scheduling. On the other hand, there are some studies [Guneysu, Paar, 2008] which do not use modular reduction. Instead reduce values while doing addition or multiplication. In order to do that, lots of block RAM blocks have been used to keep modular values. Most of these studies [Mcivor, Mcloone, Mccanny, 2006], [Guneysu, Paar, 2008], [Ananyi, Alrimeih, Rakhmatov, 2009], [Ding, Li, Gu, 2019], [Marzouqi, Al-Qutayri, Salah, 2013] present their design as a processor. Making a basic processor out of these primarily repeated actions might be the best idea of all. Another benefit of processors is that they usually handle a variety of prime fields. In essence, our architecture is a processor that has three instructions: point addition, point doubling, and modular inversion. However, it is significantly more rudimentary than the CPU architectures that have been presented.

To summarize, we examined the outcomes of our implementation and found that it has a significant place among previous studies. As previously stated, we used scheduling and pipelining to obtain the optimum scalar multiplication result using preset units. Our multiplication performance and scheduling tables set us apart from other tasks.

5. CONCLUSION

We introduced a new ECC solution for the P256 prime field and associated arithmetic units for expediting computationally expensive atomic blocks of modular operations across a limited set of prime fields $GF(p)$. We concentrate on multiplication acceleration in particular. In projective coordinates, there is a lot of multiplication due to point addition and doubling. Only the NIST-recommended P256 prime is supported by our architecture. Different prime fields, on the other hand, can be accommodated by constructing modular reduction units. A typical scalar multiplication takes 0.2 ms on our Xilinx Virtex-7 FPGA implementation, which runs at 90 MHz and consumes 7.2k-44/DSP-Slice.

In the future, we want to expand the suggested hardware architecture to incorporate more safe NIST prime fields [Web 4, 1999], with a particular focus on side-channel attacks and countermeasures.

REFERENCES

- Alrimeih H., Rakhmatov D., (2014), “Fast and flexible hardware support for ecc over multiple standard prime fields”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22(12), 2661–2674.
- Ananyi K., Alrimeih H., Rakhmatov D., (2009), “Flexible hardware processor for elliptic curve cryptography over nist prime fields”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 17(8), 1099–1112.
- Asif S., Hossain M. S., Kong Y., Abdul W., (2018), “A fully rns based ecc processor”, Integration, 61, 138–149.
- Awaludin A. M., Larasati H. T., Kim H., (2021), “High-speed and unified ecc processor for generic weierstrass curves over $gf(p)$ on fpga”, Sensors, 21(4), 1451.
- Chen L., Moody D., Regenscheid A., Randall K., (2019), “Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters”, Tech. Rep., National Institute of Standards and Technology.
- Choi P., Lee M. K., Kim J. H., Kim D. K., (2018), “Low-complexity elliptic curve cryptography processor based on configurable partial modular reduction over nist prime fields”, IEEE Transactions on Circuits and Systems II: Express Briefs, 65(11), 1703–1707.
- Dinu D. D., (2017), “Efficient and Secure Implementations of Lightweight Symmetric Cryptographic Primitives”, Doctoral Dissertation, Luxembourg University, Luxembourg.
- Ding J., Li S., Gu Z., (2019), “High-speed ecc processor over nist prime fields applied with toom–cook multiplication”, IEEE Transactions on Circuits and Systems I: Regular Papers, 66(3), 1003–1016.
- Feng X., Li S., (2017), “A high-speed and spa-resistant implementation of ecc point multiplication over $gf(p)$ ”, IEEE Trustcom/BigDataSE/ICSS, 255–260, Sydney, NSW, Australia, 1-4 August.
- Guillermin N., (2010), “A high speed coprocessor for elliptic curve scalar multiplications over fp ”, International Workshop on Cryptographic Hardware and Embedded Software, 48–64, Berlin, Heidelberg, 18 August.
- Guneysu T., Paar C., (2008), “Ultra high performance ecc over nist primes on commercial fpgas”, Cryptographic Hardware and Embedded Systems, CHES 5154.
- Hossain M. S., Kong Y., (2015), “High-performance fpga implementation of modular inversion over f_{256} for elliptic curve cryptography”, IEEE International

Conference on Data Science and Data Intensive Systems, 169–174, Sydney, NSW, Australia, 11-13 December.

Koblitz N., (1987), “Elliptic curve cryptosystems”, *Mathematics of Computation* 48(177), 203–209.

Lenstra, A., Verheul, E., (2001), “Selecting Cryptographic Key Sizes”, *Journal of Cryptology*, 14(4), 255–293.

Mahto D., Yadav D. K., (2017), “RSA and ECC: a comparative analysis”, *International Journal of Applied Engineering Research*, 12(19), 9053-9061.

Marzouqi H., Al-Qutayri M., Salah K., (2013), “An fpga implementation of nist 256 prime field ecc processor”, *IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, 493–496, Abu Dhabi, United Arab Emirates, 8-11 December.

Marzouqi H., Al-Qutayri M., Salah K., Schinianakis D., Stouraitis, T., (2016), “A high-speed FPGA implementation of an RSD-based ECC processor”, *IEEE Trans. VLSI Systems*, 21(1), 151–164.

Matutino P. M., Araujo J., Sousa L., Chaves R., (2017), “Pipelined fpga coprocessor for elliptic curve cryptography based on residue number system”, *2017 International Conference on Embedded, Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 261–268, Pythagorion, Greece, 17-20 July.

McCurley K. S., (1990), “The discrete logarithm problem”, *Proc. of Symp. in Applied Math*, 49–74. USA.

Mcivor C., Mcloone M., Mccanny J., (2006), “Hardware elliptic curve cryptographic processor over $gf(p)$ ”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(9), 1946–1957.

Mehrabi M. A., Doche C., Jolfaei A., (2020), “Elliptic curve cryptography point multiplication core for hardware security module”, *IEEE Transactions on Computers*, 69(11), 1707–1718.

Miller V., (1985), “Use of elliptic curves in cryptography”, in H. C. Williams, Editor, “*Lecture Notes in Computer Science*”, Springer-Verlag.

NIST, (2013), *FIPS 186-4–Digital Signature Standard (DSS)*, National Institute of Standards and Technology.

Paar C., Pelzl J., (2009), “*Understanding Cryptography: A Textbook for Students and Practitioners*”, Springer Science & Business Media.

Ramsey H., Ravi J., William D., Winser A., (2009), “Automated Design Space Exploration for DSP Applications”, *Signal Processing Systems*, 56, 199-216.

Savas E., Koc C. K., (2000), “The montgomery modular inverse-revisited”, IEEE transactions on computers, 49(7), 763–766.

Shawkat S. A. (2007). “Enhancing Steganography Techniques in Digital Images” Doctoral dissertation, Mansoura University.

Varchola M., Guneyusu T., Mischke O., (2011), “A lightweight reconfigurable elliptic curve crypto processor”, International Conference on Reconfigurable Computing and FPGAs, 204–210, Cancun, Mexico, November 30 - December 2.

Web 1, (2021), <https://www.ni.com/en-tr/innovations/white-papers/08/fpga-fundamentals.html>, Erişim Tarihi: 14/11/2021.

Web 2, (2020), <https://www.sibervatan.org/makale/scytale-sifreleme/19>, Erişim Tarihi: 16/11/2021.

Web 3, (2020), <https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences>, Erişim Tarihi: 17/11/2021.

Web 4, (1999), <https://safecurves.cr.yp.to/>, Erişim Tarihi: 22/11/2021.

Wu T., (2015), “Elliptic curve $gf(p)$ point multiplier by dual arithmetic cores”, IEEE 11th International Conference on ASIC (ASICON), 1–4, Chengdu, China, November.

Wu T., Wang R., (2019), “Fast unified elliptic curve point multiplication for nist prime curves on fpgas”, Journal of Cryptographic Engineering, 9(4), 401–410.

BIOGRAPHY

Sukru Uzun completed his Bachelor's in Electronics and Communication Engineering from Yildiz Technical University in 2018. After graduating, he started his master's education in Electronics Engineering at Gebze Technical University in 2019. He has also been working at Procenne as a digital design engineer since 2018, where he develops high-performance cryptographic cores, kernel-user space drivers and performs side-channel analysis on them.

