Elliptic Curve Cryptography on FPGA

By

Sunny Kumar Pandit (959)

Sunny Kumar (958)

Saurav Roy (953)

Bachelor Thesis submitted to

Indian Institute of Information Technology Kalyani

for the partial fulfillment of the degree of

Bachelor of Technology

in

Computer Science and Engineering

Certificate

This is to certify that the thesis entitled “Elliptic Curve Cryptography on FPGA” is being submitted by Sunny Kumar Pandit (CSE/22105/959), Sunny Kumar (CSE/22104/958) and Saurav Roy (CSE/22099/953) undergraduate students, in Computer Science and Engineering, Indian Institute of Information Technology Kalyani for the award of Bachelors of Technology in Computer Science Engineering, is an original research work carried by them under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulation of the IIIT Kalyani and in my opinion, has reached the standards needed for submission. The works, techniques, and results presented have not been submitted to any other university or Institute for the award of any other degree or diploma.

Dr. Soumen Pandit

Assistant Professor

Indian Institute of Information Technology Kalyani

Kalyani, W.B.-741235, India

Declaration

We hereby declare that the work presented in this thesis, entitled *"*Elliptic Curve Cryptography on FPGA*"*, is submitted to the Indian Institute of Information Technology Kalyani in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering**.**

This thesis does not contain any classified information. Furthermore, it does not include any AI-generated text or material. The work was carried out during the period January 2025 to May 2025 under the supervision of Dr. Soumen Pandit, Indian Institute of Information Technology Kalyani, West Bengal – 741235, India.

Sunny Kumar Pandit (CSE/22105/959)

Sunny Kumar (CSE/22104/958)

Sourav Roy (CSE/22099/953)

Computer Science and Engineering

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.....

Dr. Soumen Pandit

Assistant Professor

Indian Institute of Information Technology Kalyani

Kalyani, W.B.-741235, India

Place: Kalyani

Date: 23/05/2025

Acknowledgments

First of all, we would like to take this opportunity to thank our supervisor Dr. Soumen Pandit, without whose supervision, this thesis would not have been possible. We are so grateful to him for working tirelessly after us, answering our doubts whenever and wherever possible. We are most grateful to the IIIT Kalyani, India, for providing us with this wonderful opportunity to complete our bachelor thesis. And last but not least, we want to thank each other for always having the backs of others and giving their best in completing the thesis.

Sunny Kumar Pandit (CSE/22105/959)

Sunny Kumar (CSE/22104/958)

Sourav Roy (CSE/22099/953)

Computer Science and Engineering

Indian Institute of Information Technology Kalyani

Kalyani, W.B.-741235, India

Place: Kalyani

Date: 23/05/2025

Contents

Abstract

Elliptic Curve Cryptography (ECC) is a modern public key cryptosystem that offers strong security with smaller key sizes compared to traditional methods such as RSA. This makes it highly suitable for applications requiring efficiency, such as embedded systems and IoT devices. However, implementing ECC in software can be computationally intensive and power-hungry.

This thesis presents the hardware implementation of ECC using a Field-Programmable Gate Array (FPGA), which offers significant advantages in terms of parallel processing, speed, and energy efficiency. The design is modeled in VHDL/Verilog and simulated using ModelSim, followed by synthesis and deployment on a Xilinx FPGA board. Key components such as finite field arithmetic, scalar multiplication, and point addition are optimized for hardware execution.

The results demonstrate that FPGA-based ECC significantly reduces computational latency while maintaining accuracy and resource efficiency. This makes the design a viable option for secure, high-performance embedded cryptographic applications.

Chapter 1

Introduction

In this moment, the protection of sensitive information is very crucial and its importance is heightened especially in today's digital world. Sensitive data such as financial interactions and personal correspondence are protected by cryptography. Elliptic Curve Cryptography (ECC) is one of the most remarkable and efficacious advancements in the cryptography field.

ECC falls under the category of public-key cryptography. It is based on the algebraic structure of elliptic curves over finite fields. It is particularly alluring because it offers a high level of security with greatly reduced key sizes. A perfect example would be that a 256-bit ECC key is as powerful as a 3072-bit RSA key. It is most effective in mobile devices, embedded systems, and Internet of Things (IoT) devices due to its faster computations, lesser power consumption, and lower storage needs which is ideal in resource-limited scenarios.

ECC's foundation is determined by the mathematical difficulties captivating elliptic curves, especially the challenge of solving the ECDLP. Unlike the standard cryptographic algorithms, which tend to require higher levels of calculations, ECC enables secure communication whilst minimizing the computational burden.

This dissertation focuses on the implementation of ECC using field program gate matric (FPGAS). FPGAs provide a flexible and effective hardware platforms that enable cryptographic high -speed cryptographic operations through parallel processing. By taking advantage of hardware acceleration, ECC can be used in a way that is both safe and customized to perform, which is especially important in real time and built -in applications.

The upcoming chapters will explore the underlying principle of ECC, review the related work and describe the design and implementation of ECC on an FPGA platform. The goal is to demonstrate how the ECC can feel effective in hardware to meet the requirements for modern safe communication systems.

1.1 Symmetric Key Cryptography

Symmetric key cryptography, also known as secret key cryptography, is one of the earliest and most widely used methods of securing information.

In this method, the identical key is employed for both the processes of encryption and decryption. This necessitates that both the sender and the receiver must have access to this one, shared secret key that remains confidential.

Symmetric key cryptography's main advantage is its speed and efficiency, which makes it ideal for the encryption of large amounts of data. Among the most common algorithms used are AES (Advanced Encryption Standard), DES (Data Encryption Standard), and Blowfish. One big drawback, though, is the problem of securely distributing the secret key to the parties having the communication, especially when the communication channels are themselves insecure. If the key gets intercepted or compromised, then the breakability of the encryption guarantees that the communication itself is at risk.

1.2 Asymmetric Key Cryptography

Asymmetric key cryptography, also called public key cryptography, addresses the key distribution problem found in symmetric methods.

It employs a set of two keys that are mathematically related—a public key and a private key. The public key is shared freely and used for encrypting messages. The private key is known only to the owner and is used for decrypting messages.

Secure communication can happen even when there is no forehanded sharing of private keys. We have at our disposal a number of asymmetric algorithms that work well and have been reliable over the years. These include RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC). As a group, they tend to be slower and require more computational power than symmetric key methods. Nonetheless, the payoff in features that enhance security and make it easier to achieve secure communication renders them a good investment. Among the features I speak of are digital signatures and key exchange.

1.3 Foundational Security Objectives

Confidentiality ensures that information is accessible only to those authorized to access it. It prevents eavesdropping and data leakage by transforming readable data (plaintext) into an unreadable format (ciphertext) using encryption algorithms. Even if attackers intercept the message, they cannot understand its contents without the corresponding decryption key. ECC contributes to confidentiality by enabling secure encryption with relatively smaller key sizes compared to other public-key systems like RSA, while maintaining a similar level of security.

Integrity guarantees that the information has not been altered in transit, either accidentally or maliciously. It assures the recipient that the data received is exactly what was sent. Cryptographic hash functions, digital signatures, and message authentication codes (MACs) are commonly used to detect unauthorized changes. ECC-based digital signatures, such as ECDSA (Elliptic Curve Digital Signature Algorithm), provide a lightweight yet secure way to verify message integrity.

Authentication is the process of verifying the identity of a user or system. It ensures that the party communicating is who they claim to be. Public-key cryptography, including ECC, supports authentication by using asymmetric key pairs. When a message is signed using a private key, anyone with the corresponding public key can verify the sender's identity, thus ensuring authenticity.

Chapter 2

Groups and Elliptic Curves

2.1 Elliptic Curves in Cryptography and Singularities

Elliptic curves used in cryptography are carefully selected to ensure both mathematical soundness and security. Not every elliptic curve is suitable for cryptographic applications. The curves must be non-singular and defined over a finite field, typically of the form $\mathbb{F}_p$ (a prime field) or $\mathbb{F}_{2^n}$ (a binary field).

In cryptography, the most commonly used form of an elliptic curve over a prime field $\mathbb{F}_p$ is the short Weierstrass form:

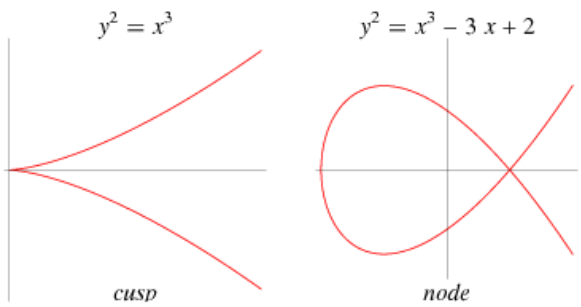
$$E : y^2 \equiv x^3 + ax + b \mod p$$

where:

- a and b are constants in the field $\mathbb{F}_p$,
- p is a large prime number, and
- the curve must satisfy the non-singularity condition:

$$4a^3 + 27b^2 \not\equiv 0 \mod p$$

This condition ensures that the curve does not have any singularities—points where the curve fails to be smooth or where it in tersects itself.



2.2 Group Structure in Elliptic Curves

Elliptic Curve Cryptography (ECC) relies heavily on the mathematical concept of groups, particularly abelian groups, formed by points on an elliptic curve. A group in mathematics is a set equipped with a binary operation that satisfies four fundamental properties: closure, associativity, identity, and inverses.

In the context of ECC, we work with elliptic curves over finite fields (typically denoted $\mathbb{F}_p$ where p is a prime). The set of points on an elliptic curve $E(\mathbb{F}_p)$, together with a special point at infinity denoted ∞, forms a finite abelian group under a well-defined addition operation.

Group Properties on Elliptic Curves :

1. **Closure**:
   If P and Q are two points on the curve, then their sum $P + Q$ is also a point on the curve.
2. **Associativity**:
   The addition of points is associative: $(P + Q) + R = P + (Q + R)$.
3. **Identity Element**:
   There exists a point ∞ (point at infinity) such that for any point P,
   $P + \infty = P$.
4. **Inverse Element**:
   For every point P, there exists a point $-P$ such that $P + (-P) = \infty$.
5. **Commutativity (Abelian Group)**:
   The addition operation is commutative: $P + Q = Q + P$.

The group structure ensures that operations like scalar multiplication are mathematically sound and behave predictably. The difficulty of reversing scalar multiplication—i.e., computing d from P and $Q = d \cdot P$—forms the basis of ECC's security. This problem is known as the Elliptic Curve Discrete Logarithm Problem (ECDLP), and it is computationally hard to solve, even with powerful algorithms.

2.3 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$, where *p* is a prime number. Suppose P is a point on $E(\mathbb{F}_p)$ with prime order n. Then, the subgroup generated by P is cyclic, and it consists of the following points:

$\langle P \rangle = \{\infty, P, 2P, 3P, ..., (n-1)P\}$

Here, ∞ denotes the point at infinity, which serves as the identity element of the group.

The following parameters are publicly known:

- The prime p
- The equation of the elliptic curve E
- The point P on the curve
- The order n of the point P

To generate a key pair:

- A private key is chosen as an integer d, randomly and uniformly selected from the range $[1, n-1]$
- The corresponding public key is then computed as $Q = d \cdot P$

The security of elliptic curve cryptography relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP)—that is, given P and $Q = d \cdot P$, it is computationally hard to determine d.

2.4 Operations on Elliptic Curves

Elliptic Curve Cryptography (ECC) relies heavily on arithmetic operations defined over the points on an elliptic curve. These operations form the basis of ECC algorithms such as encryption, decryption, key exchange, and digital signatures.

2.4.1 Point Addition

Point addition refers to the operation of adding two distinct points P and Q on the elliptic curve to obtain a third point R = P + Q, also on the curve. Geometrically, this involves drawing a straight line through P and Q. This line will intersect the curve at a third point, say −R, and reflecting this point over the x-axis gives R.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, and $P \neq Q$. Then, the slope m is given by:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \mod p$$

Then the resulting point $R = (x_3, y_3)$ is calculated as:

$$x_3 = m^2 - x_1 - x_2 \mod p$$

$$y_3 = m(x_1 - x_3) - y_1 \mod p$$

Why do we refelect third point ? Let's say we have three points A, B and C on a line that passes thorugh the curve. We can write

$$\mathrm{A} + \mathrm{B} = \mathrm{C}$$

$$\mathrm{B} + \mathrm{C} = \mathrm{A}$$

$$\mathrm{A} + \mathrm{C} = \mathrm{B}$$

Now with little algebra we can derive a contradiction

$$(\mathrm{B} + \mathrm{C}) + \mathrm{B} = \mathrm{C}$$

$$\mathrm{B} = \mathrm{inv}(\mathrm{B})$$

This says B is equal to its inverse but this should be only true for Identity element in the group. But clearly we know that B is not the identity element and adding it to other points will not give the same element back. Hence to fix this we take reflection of the third point.

2.4.2 Point Doubling

Point doubling is the special case of point addition where both points are the same, i.e., P = Q. Geometrically, this means drawing a tangent to the curve at point P, which intersects the curve at another point. Reflecting that point over the x-axis gives the result R = 2P.

Let $P = (x_1, y_1)$. The slope m for point doubling is:

$$m = \frac{3x_1^2 + a}{2y_1} \mod p$$

Then:

$$x_3 = m^2 - 2x_1 \mod p$$

$$y_3 = m(x_1 - x_3) - y_1 \mod p$$

2.4.3 Point at Infinity (Identity Element)

Elliptic curves include a special point known as the point at infinity, denoted $\mathcal{O}$ or sometimes ∞. It serves as the identity element for the elliptic curve group. This means:

$$P + \mathcal{O} = P \quad \text{for any point } P$$

Geometrically, it can be thought of as the point where vertical lines intersect the curve "at infinity."

2.4.4 Scalar Multiplication

Scalar multiplication is the most important operation in ECC. It refers to multiplying a point P on the curve by a scalar integer k, resulting in another point Q = kP.

This is done by performing repeated point additions and doublings:

- Efficient algorithms such as double-and-add, Montgomery ladder, or windowed methods are used to compute this operation securely and efficiently.
- Scalar multiplication underlies ECC key generation, digital signatures, and Diffie-Hellman key exchange.

For example : kP = P+P+⋯+P (k times)

2.5 Importnace of Operations

Among these operations, scalar multiplication is the backbone of ECC security. It is computationally easy to perform, but very difficult to reverse (i.e., given Q = kP, finding k is infeasible). This difficulty is known as the Elliptic Curve Discrete Logarithm Problem (ECDLP), and it forms the core hardness assumption in ECC-based systems.

Chapter 3

ElGamal with ECC

3.1 Classical ElGamal Overview

The classical ElGamal encryption scheme operates over multiplicative groups Z*p, where its security is based on the hardness of the Discrete Logarithm Problem (DLP). The scheme involves key generation, encryption, and decryption using modular exponentiation.

When transferred to ECC, modular arithmetic is replaced with elliptic curve point arithmetic, and the security depends on the Elliptic Curve Discrete Logarithm Problem (ECDLP).

3.2 System Parameters and Key Generation

The Elliptic Curve ElGamal encryption scheme begins with the selection of publicly agreed parameters. This includes a large prime number p, over which an elliptic curve E is defined, typically in the Weierstrass form. The curve E is constructed over the finite field Fp, and a base point G is chosen on this curve. This point G is a generator of a cyclic subgroup of the elliptic curve group and is selected such that it has a large prime order n. The parameters (E,p,G,n) are made public and are used by all participants in the cryptographic scheme.

Each user generates their key pair by first selecting a private key d, which is a randomly chosen integer in the range $1 \leq d \leq n-1$. The corresponding public key is then computed as $Q=dG$, where the operation denotes scalar multiplication of the base point G by the scalar d on the elliptic curve. The private key d is kept secret, while the public key Q is shared openly. The security of this key generation process relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP), which ensures that even with knowledge of Q and G, it is computationally infeasible to derive the private key d.

3.3 Encryption

To encrypt a message M, the message must first be encoded as a point $P_m \in E(F_p)$. This is often done using deterministic or probabilistic point encoding schemes.

The sender:

1. Selects a random integer $k \in [1,n-1]$
2. Computes:

$$C_1 = kG$$

$$C_2 = P_m + kQ$$

The ciphertext is the pair (C_1, C_2).

3.4 Decryption

The recipient uses the private key d to compute:

$P_m = C_2 – dC_1$

$C_2 – dC_1 = P_m + kQ – d(kG) = P_m + k(dG) – d(kG) = P_m$

This retrieves the original point P_m, which can then be decoded to the message M.

3.5 Message Encoding

ECC operates on points, not raw binary data. Therefore, encoding the plaintext into a point P_m is necessary. Some common methods include:

- **Koblitz encoding**: Attempts to map a message block to a curve point using a fixed function and retries until a valid point is found.
- **Hybrid encryption**: Encrypt a symmetric session key using ECC-ElGamal, and use symmetric encryption (e.g., AES) for the actual message.

3.6 Advantages of ECC-ElGamal

One of the primary advantages of Elliptic Curve Cryptography (ECC) lies in its ability to provide strong security with significantly shorter key lengths compared to traditional public-key systems such as RSA. For instance, a 256-bit key in ECC offers a comparable level of security to a 3072-bit RSA key, resulting in more compact cryptographic keys and reduced storage and transmission requirements. In addition to key size efficiency, ECC operations are computationally less intensive, making the scheme particularly well-suited for resource-constrained environments such as mobile devices, embedded systems, and Internet of Things (IoT) applications. Furthermore, the scalability of ECC enables seamless integration into modern cryptographic frameworks, including Transport Layer Security (TLS), blockchain technologies, and various cryptocurrency protocols, where efficient and secure public-key operations are essential.

3.7 Conclusion

Elliptic Curve ElGamal encryption provides a robust, efficient, and secure public-key cryptographic scheme. It inherits the security foundation of the classical ElGamal encryption while taking advantage of the mathematical elegance and efficiency of elliptic curve groups. Despite its strengths, practical implementations must account for secure message encoding, resistance to side-channel attacks, and authenticated encryption techniques.

Chapter 4

ECC on FPGA

4.1 Overview of the Boolean Board

The implementation of Elliptic Curve Cryptography (ECC) in hardware requires a reconfigurable and resource-efficient platform. For this purpose, the Boolean Board equipped with a **Spartan-7 FPGA** was selected. This development board is tailored for digital logic design, embedded systems education, and prototyping of cryptographic algorithms due to its accessibility, moderate resource capacity, and support for Xilinx's Vivado Design Suite.

The Boolean Board integrates the **Spartan-7 XC7S50CSGA324-1** Field-Programmable Gate Array (FPGA), which belongs to the **Xilinx Spartan-7 product family**. The Spartan-7 series is known for offering an optimal balance between performance, power efficiency, and cost, making it suitable for implementing moderately complex cryptographic operations such as ECC scalar multiplication and modular arithmetic.

4.2 FPGA Device Specification

The specific FPGA device used in this project is the **XC7S50CSGA324-1**, characterized by the following key features:

- **Logic Cells**: Approximately 52,160 logic cells, suitable for implementing parallelized datapaths and finite field arithmetic.
- **LUTs and Flip-Flops**: Provides 32,600 look-up tables (LUTs) and 65,200 flip-flops, supporting custom datapath design for ECC point operations.
- **Block RAM**: Up to 2.7 Mb of block RAM (BRAM) for storing intermediate results, lookup tables, or key material during ECC computations.
- **DSP Slices**: Contains 120 DSP slices, useful for accelerating multiplication and arithmetic operations in finite fields.
- **I/O Pins**: Supports up to 200 general-purpose I/O pins, useful for interfacing with external devices or debugging modules.

- **Package**: The device is housed in a CSGA324 BGA package, ensuring compact form factor and reliable connectivity.
- **Speed Grade**: The speed grade "-1" denotes standard timing performance within the Spartan-7 lineup.

4.3 Modular Addition on FPGA

4.3.1 Modular Adder Based on Repeated Subtraction

The following Verilog module implements a modular addition circuit using the method of repeated subtraction. Modular addition is a fundamental operation in elliptic curve cryptography (ECC), particularly in point addition and doubling where arithmetic operations must be performed within a finite field defined by a modulus m.

This module, named modular_adder_repeated_subtract, takes three WIDTH-bit inputs: operands a, b, and the modulus m. The output is the value of (a+b) mod m, computed through iterative subtraction, which is well-suited for hardware implementation where resource constraints limit the use of more complex arithmetic units.

The design is clocked and synchronous with a reset (rst) and start signal (start). Internally, a sum register (one bit wider than the operands) temporarily holds the result of a+b. The subtraction loop continues until sum becomes less than m, at which point the final modular result is assigned to result. A done signal indicates the completion of the operation, while a busy flag controls the processing state.

This approach ensures compatibility with simple FPGA architectures and supports parameterization via the WIDTH parameter, allowing flexible operand sizes.

Key Design Features:

- **Parameterized Width**: The bit-width of inputs and outputs is configurable via the WIDTH parameter, enhancing reusability.
- **Overflow Protection**: The sum register is declared with an extra bit to safely accommodate the addition without overflow.

- **Control Logic**: A simple FSM-like control is implemented using busy and done flags for start-complete coordination.
- **Synchronous Reset**: Ensures all state variables are properly initialized when the reset signal is asserted.

```
module modular_adder_repeated_subtract #(
   parameter WIDTH = 4
)(
   input  wire clk,
   input  wire rst,
   input  wire start,
   input  wire [WIDTH-1:0] a,
   input  wire [WIDTH-1:0] b,
   input  wire [WIDTH-1:0] m,
   output reg  [WIDTH-1:0] result,
   output reg  done
);
   reg [WIDTH:0] sum;
   reg busy;

   always @(posedge clk or posedge rst) begin
      if (rst) begin
         result <= 0;
         sum <= 0;
         done <= 0;
         busy <= 0;
      end else begin
         if (start && !busy) begin
            sum <= a + b;
            busy <= 1;
            done <= 0;
         end else if (busy) begin
            if (sum >= m) begin
               sum <= sum - m;
            end else begin
               result <= sum[WIDTH-1:0];
               done <= 1;
               busy <= 0;
            end
         end
      end
   end
endmodule
```

4.3.2 Modular Reduction Using Shift-and-Subtract (Long Division) Method

The Shift_Registers module implements a more efficient modular reduction algorithm inspired by binary long division. It is designed to compute the result of (r1_in + r2_in) mod m, where the inputs r1_in, r2_in, and m are all 4-bit values. The final output is stored in the 5-bit register out.

The key advantage of this design is that the number of subtraction operations is limited by the number of bits in the input sum, not the numerical value of the inputs. Since both r1_in and r2_in are 4-bit values, their maximum possible sum is 15+15=30, which fits in 5 bits. Thus, only 5 iterations are required to fully reduce the result — one iteration per bit of the sum.

Although the loop's counter checks for termination at 4, the operations (shifting and conditional subtraction) occur before the count is checked, making this a post-check loop. As a result, the module effectively performs exactly 5 iterations, processing all 5 bits of the sum from most significant to least significant.

Operation Summary

1. The module calculates the initial sum r3 = r1_in + r2_in, stored in a 5-bit register.
2. The 5-bit register out is initialized to zero.
3. Across five clock cycles, the following operations occur:

    - The out register is left-shifted and the MSB of r3 is shifted in.
    - If the resulting temporary value is greater than or equal to mmm, a subtraction is performed: temp=temp−mtemp = temp - mtemp=temp−m.
    - The r3 register is left-shifted, effectively removing the bit that was just processed.
    - A counter tracks the number of iterations and halts the process after the fifth cycle.

```
module Shift_Registers (
    input wire clk,
    input wire rst,
    input wire start,
    input wire [3:0] r1_in,
    input wire [3:0] r2_in,
    input wire [3:0] m,
    output reg [4:0] out,
    output reg busy
);
```

```
    reg [4:0] r3;
    reg [4:0] temp;
    reg [2:0] count;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            r3 <= 0;
            out <= 0;
            temp <= 0;
            busy <= 0;
            count <= 0;
        end else begin
            if (start && !busy) begin
                r3 <= r1_in + r2_in;
                out <= 5'b00000;
                busy <= 1;
                count <= 0;
            end else if (busy) begin
                temp = {out[3:0], r3[4]};
                if (temp >= m) begin
                    temp = temp - m;
                end

                out <= temp;
                r3 <= {r3[3:0], 1'b0};
                count <= count + 1;

                if (count == 4) begin
                    busy <= 0;
                end
            end
        end
    end
endmodule
```

You can find the pin mapping I/O constrain file for both of this code on my github profile.

github.com/Sunny1619/ECC_on_FPGA

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis presented the design and implementation of Elliptic Curve Cryptography (ECC) on an FPGA platform, with a particular focus on the ElGamal encryption scheme. We explored the mathematical foundations of ECC, including group structures, point operations, and scalar multiplication, and demonstrated how these can be mapped onto hardware for efficient cryptographic processing.

The Spartan-7-based Boolean Board was chosen for its balanced combination of logic resources, DSP slices, and RAM blocks, making it suitable for implementing modular arithmetic and ECC primitives. As a case study, we developed and compared two approaches to modular addition: a basic repeated subtraction method and a more optimized shift-and-subtract method inspired by long division. Our results show that hardware-oriented algorithms can significantly reduce the number of operations and improve timing predictability — a crucial requirement for real-time cryptographic applications.

Through simulation, synthesis, and testing, we verified the functional correctness of the modules and demonstrated the feasibility of FPGA-based ECC implementations for secure, high-performance embedded systems.

5.2 Future Work

While this work has laid a strong foundation by implementing and optimizing modular addition, ECC on FPGA encompasses several additional arithmetic and control operations critical for complete ElGamal encryption and decryption. In future work, we plan to continue identifying and implementing all remaining core operations required for the full hardware realization of ElGamal on FPGA. This includes efficient designs for modular multiplication, modular inversion, scalar multiplication, and point addition and doubling under various coordinate systems (e.g., affine and projective).

The primary goal will be to explore hardware-optimized algorithms for each of these operations — with the same design philosophy applied in the modular adder — emphasizing performance, area efficiency, and constant-time execution to mitigate timing side-channel attacks. Additionally, we intend to integrate these individual modules into a complete pipeline for end-to-end ECC-ElGamal encryption and decryption, followed by performance evaluation in terms of resource usage, latency, and throughput.

By the end of this extended work, we aim to develop a fully functional, scalable, and resource-efficient ECC-based cryptosystem implemented entirely in FPGA hardware — a significant step toward deploying secure cryptographic engines in constrained and real-time environments.

References

[1] “Elliptic-curve cryptography,” *Wikipedia*, [Online]. Available: https://en.wikipedia.org/wiki/Elliptic-curve_cryptography. [Accessed: 16-May-2025].

[2] R. Saied, “Elliptic curve point addition explained,” *RareSkills Blog*, Feb. 7, 2023. [Online]. Available: https://www.rareskills.io/post/elliptic-curve-addition. [Accessed: 16-May-2025].

[3] Riverninj4, “Elliptic Curve Cryptography (ECC) Explained,” *YouTube*, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=XmygBPb7DPM&list=PL8nBmR5eGh37N_BFFj3y35KIBzpSFXmNG. [Accessed: 16-May-2025].