A Novel High Speed GF (2^{173}) Elliptic Curve Crypto-processor

Leung Pak Keung

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Electronic Engineering

Supervised by

Prof. Oliver, C. S. Choy

Abstract

Elliptic Curve Cryptography is a subset of the public key cryptography. It is becoming popular in recent decades due to its highest security strength per bit, less memory requirement and low processing power. These advantages make it attractive to be applied in energy constrained applications such as contact-less smartcard system and portable cellular phone.

In this project, a 173-bit (m = 173) Type II Optimal Normal Basis (ONBII) representation is chosen in the implementation of the Galois Field GF (2^m) arithmetic logic unit over the projective coordinates system. An efficient Elliptic Curve Crypto-processor which is optimized for low power consumption is presented in this thesis.

Firstly, the mathematical equations are simplified. Compared with the original one, the proposed expression can save 7.7% curve addition operations, which is the major operation in the algorithm.

Secondly, the architecture of the finite field multiplier is modified to reduce the data transferred between the flip flops and logic elements. By using a 3-way parallel multiplier, the processor can maintain the speed performance while achieve the low power consumption.

Furthermore, in the design of curve operations, the field operations are integrated into one single instruction. This effectively reduces the number of memory units which are used to store the temporary variables during the computation.

The test chip was fabricated with a 0.35um CMOS technology. At

27℃ and under a 3V supply voltage, this chip can operate at 18MHz and the time for a key generation is about 7.56 ms.

摘要

橢圓曲線密碼學是公鑰密碼學的一種，而此種密碼學在近十多年間正逐漸普及起來，這是因爲它能以相對較少的公鑰位元數（key length）來提供一個較高的安全系數，從而減少記憶體的使用量和減低運算時所消耗的能量。正因爲這些有利條件，使此項技術能應用於一些能量供應緊拙的器材上，如無線智能咭系統和手提無線電話便是其中一些例子。

本篇論文的研究課題是製作一個 173 位元的橢圓曲線密碼系統，它的運算邏輯單元（ALU）裡採用了有限場域（finite field）中第二類最優化正規基底（Type II Optimal Normal Basis）表示，再與投影幾何坐標系統（projective coordinate system）結合，並加上有效的設計方法來達至低耗電量的目的。

首先，此項新的設計是透過簡化數學公式來減少座標點相加（curve addition）的運算次數。此項簡化相比原有的座標點相加公式能有效節省約 7.7 個百分點的運算負荷。

其次，此設計在有限場域乘法器中採用了三路並行方法，這有效地減少了觸發器（flip flop）及邏輯單元之間的數據傳輸次數，從而獲得速度保證及減低耗電量。

最後在座標點運算層上，我們把所有的場域運算整合起來成爲單一運算指令，此舉能減少用來儲存臨時數據的記憶空間。

此測試晶片使用 0.35 微米互補金屬氧化半導體(CMOS)技術製作。在 27℃及 3 伏電壓環境下，此晶片能夠在 18 兆赫的頻率運作，而產生一個密鑰所需要的時間大概只是 7.56 毫秒。

Acknowledgements

I would like to express my deepest gratitude to many people who provided me with sincere assistance in this research.

I would firstly like to thank my supervisor, Professor Oliver, Choy Chiu Sing, for his invaluable support and guidance throughout the research. Beyond these, he gave me a chance to study the master degree, which has changed my life a lot. Furthermore, I would like to thank Professor Chan Cheong Fat and Professor Pun Kong Pang, who taught me the fundamental knowledge about the ASIC design.

I would also like to thank Professor Philip, Leong Heng Wai from the computer engineering department, for he has pointed out the research direction for me and given me many invaluable advices about the research.

Thank my colleagues, Cheng Wang Chi, Hon Kwok Wai, Han Wei, Chan Wing Kin, Yu Chun Pong, Yeung Wing Ki, Shen Junhua, Chan Chi Hong, Cheng Wang Tung, Kwok Yan Lun, Chan Pak Kei and Tang Siu Kei who always hold a group discussion to share the research experiences. And thank our ASIC Laboratory technician Mr. Yeung Wing Yee for his supporting the computer facilities and solving my computer related problems.

At last, I would like to give thanks to my parents for their understanding and sacrifices. Without their support, I could not finish my study successfully.

Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Introduction to Elliptic Curve Crypto-processor

Nowadays, contact-less smartcard systems play an important role in the market and are widely used in data communication such as electronic money, identification, etc. As many of these applications communicate through an insecure channel, the smart card manufacturers add the public key scheme in order to keep the data in secrete. The most well known public key algorithm is RSA and ECC (Elliptic Curve Cryptography). Although RSA is mature and widely used today, its long key length and huge computational requirement make it not suitable in the exponential growing market. However, the ECC proposed by Neal Koblitz [1] and Victor Miller [2] is more appropriate do this task.

In this thesis, a hardware GF(2^{173}) Elliptic Curve Crypto-processor is proposed. This cryptosystem is implemented by using 173-bit type II Optimal Normal Basis (ONBII) representation. ONB was considered as the fastest hardware implementation method in ECC over polynomial basis representation and prime number representation. The basic operations in ONB are addition, squaring, multiplication, and inversion, all of which are simple in hardware implementation. The following chapters will describe how to improve the performance of the ECC processor in details.

1.2 Aims

Power saving is one of the most crucial criteria in the contact-less smart card systems. The objective of this research is to develop a low power public key crypto-processor which can be embedded into these systems.

Power efficient architecture can be achieved by:

- Using efficient public key cryptography algorithm like ECC. Since ECC provides the highest security bit per length and minimum computational and memory requirements among all the algorithms, its operating time is very short and thus the power consumption is small.
- Suitable choosing of the power efficient design methodology like Optimal Normal Basis (ONB) representation.
- Reducing the switching activities of the logic gates, since the power consumption is directly proportional to the switching activities of the logic gates.

1.3 Contributions

The proposed ECC processor has several features, and some of them even have not been presented by all the previous designs yet. They are:

- Properly choosing the key size of the cryptosystem. Consider the trade-off between the security strength and power concern, this research uses 173-bit as the key length. First of all, 173 is a prime number and the fastest method to crack a 173-bit cryptosystem is the discrete logarithm problem, which is

infeasible to crack in short time. Secondly, the security level is higher than 1024-bit RSA (a commercial standard using nowadays). And finally, as the Optimal Normal Basis (ONB) representation is employed in the design, we have to choose some specific numbers which can be represented by ONBII and the 173-bit number is one of them.

- The mathematical expressions of the elliptic curve addition have been modified. By doing this, one less multiplication is involved in every curve addition operation.
- Combining the instructions of the finite field operations into one single instruction. This has effectively reduced the power consumption and the propagation delay.
- The ECC processor was designed by using a 0.35um standard cell library. The interconnections between the logic gates were minimized by the layout tool "Silicon Ensemble".

1.4 Thesis Outline

The idea of cryptography is introduced in Chapter 2.

In Chapter 3, the number theory about finite field is presented. Some examples are given to explain the theory.

Chapter 4 is an introduction to the elliptic curve theory. The relationship between the curve and cryptography is shown here.

Chapter 5 describes the design methodology of this ECC processor, the implementation of the crypto-processor will be dissected part by part.

Chapter 6 shows the specification, pin assignment and the

communication protocol used in this integrated circuit.

Chapter 7 shows the performance of this cryptosystem. The comparison between the new processor and the previous work also stated here.

At last, Chapter 8 is the conclusion of this thesis and some of the recommendations of the future research are given.

Chapter 2 Cryptography

2.1 Introduction to Cryptography

Cryptography is the science of using mathematics to encrypt and decrypt data. It enables you to store secret information or transmit it across insecure media such as internet, so that it cannot be interpreted by anyone especially cryptanalysts.

To avoid the cryptanalysts retrieve secret information easily, the security level of cryptography should be strong enough to against the attack. Time and resources required to recover the original information are essential indicators to measure the cryptosystem is strong or not.

Moreover, properly choosing the cryptographic algorithm is also necessary. For example, Triple DES and RC5 are suitable in symmetric key encryption (secret key encryption), while ECC and RSA are suitable in asymmetric key encryption (public key encryption).

A cryptographic algorithm is a mathematical function used in the encryption and decryption process. It works in combination with a key to encrypt the plaintext, and the same plaintext encrypts to different cipher text with different keys. The security of encrypted data is entirely dependent on the strength of the algorithm chosen and the size of the key.

In this chapter, two major classes of cryptosystems - secret key cryptography and public key cryptography are introduced. Then a brief overview of discrete logarithm problem is given. Finally, two major public

key encryption standards, ECC and RSA are compared here.

2.2 Public-key Cryptosystems

Public key cryptosystems are based on mathematical functions about number theory. It involves the use of two separate keys, so we call it asymmetric.

Public key algorithms rely on one key for encryption and a different but related key for decryption. There are two important characteristics in these algorithms.

- It is computationally infeasible to determine the decryption key except when the cryptographic algorithm and the encryption key are known.
- Either of the two related keys can be used for encryption, with the other is used for decryption.

Public key cryptosystems can be classified into three main categories, they are:

Encryption/Decryption: The sender encrypts a message with the recipient's public key. Figure 2.1 illustrates this application. Suppose Alice wants to send a secret message M to Bob through an insecure channel, she needs to encrypt the message. In the first step, Bob should generate public key pair (Kb_{pri}, Kb_{pub}), and gives his public key (Kb_{pub}) to Alice. While Alice gets Bob's public key, she can use this key to encrypt her message, Kb_{pub} (M), and send the ciphertext, C, to Bob safely. When Bob receives this ciphertext, he can use his private key, Kb_{pri}, to decrypt

the information and get the original plaintext, M.

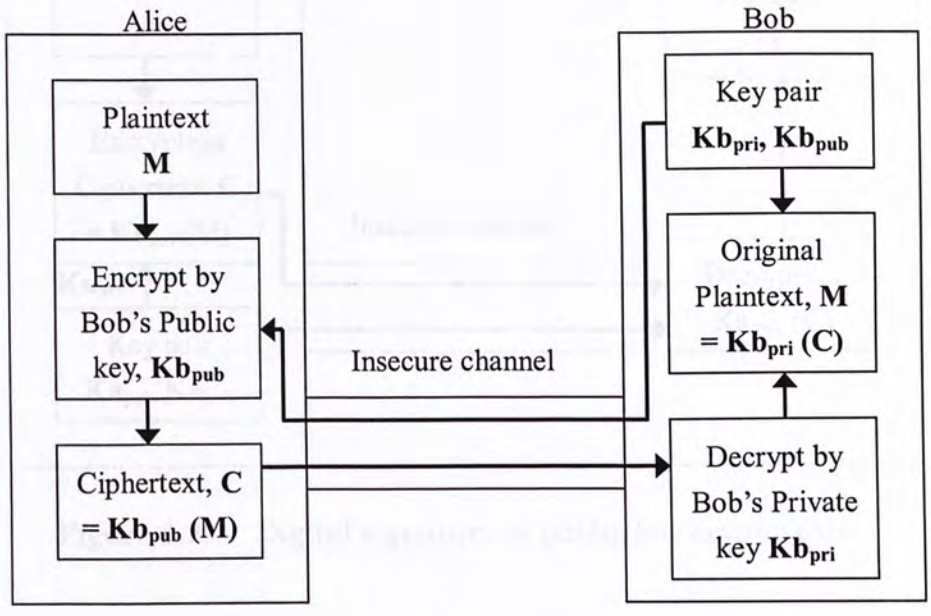


Figure 2.1. Encryption/decryption using public key cryptosystem

Digital signature: The sender signs a message with its own private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message. Figure 2.2 illustrates this application. In this case, Alice prepares a message to Bob and encrypts it using her private key before transmitting it. Then Bob can decrypt the ciphertext using Alice's public key. Because the ciphertext is signed by Alice's private key, any other public keys cannot decrypt it except her public key, so we called this digital signature.

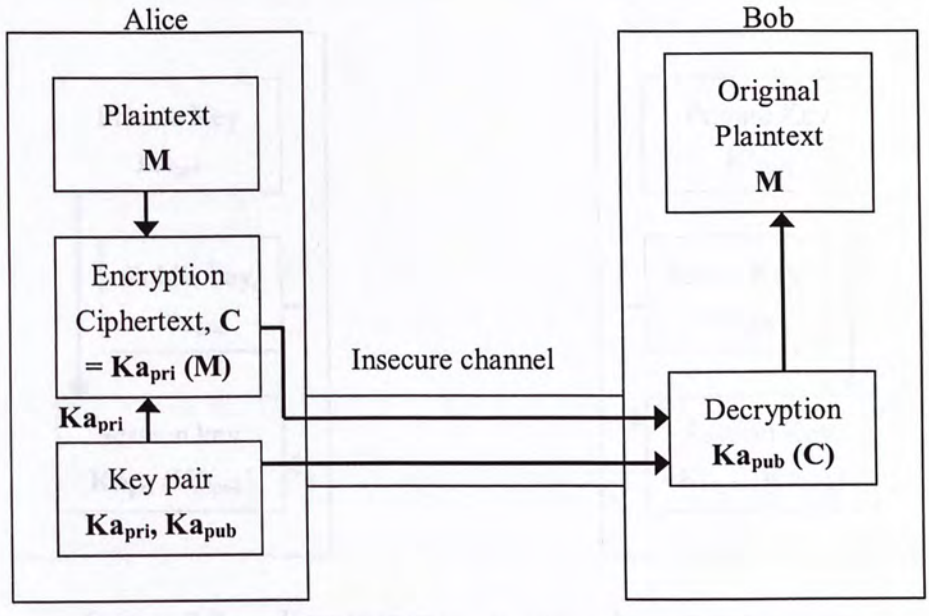


Figure 2.2. Digital signature in public key cryptosystem

Key exchange: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key of one or both parties. Figure 2.3 illustrate this application. Suppose Alice and Bob want to encrypt/decrypt their message using symmetric key encryption, they need to get a session key first. Key exchange can let them get the session key safely through an insecure channel. At the beginning, Alice and Bob exchange their public key to the other, and using their private key to encrypt the other's public key, the keys generated by this method are identical, which is called session key.

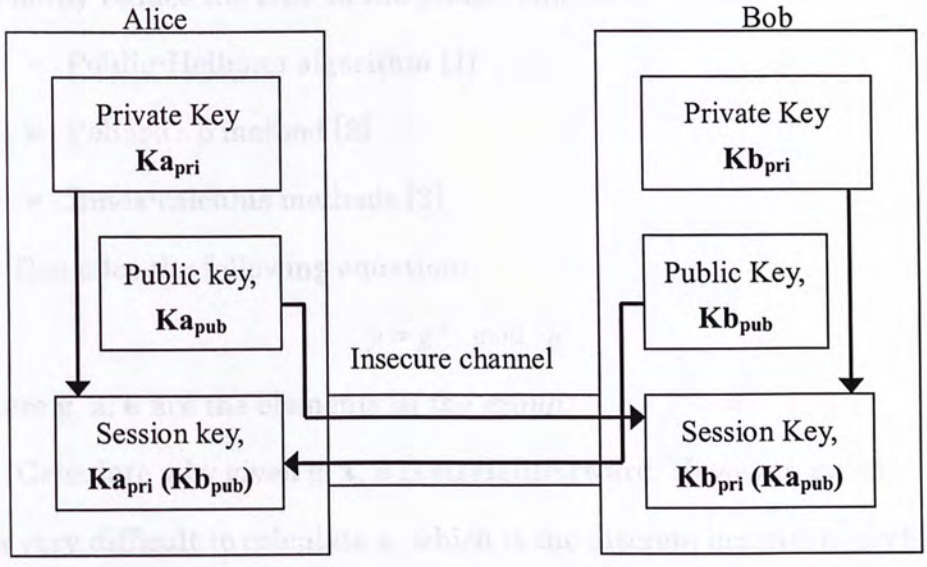


Figure 2.3. Key exchange in public key cryptosystem

2.3 Secret-key Cryptosystems

Secret key cryptosystems also called symmetric key cryptosystems. It was because the keys used in encryption and decryption are the same.

$$E(M) = C$$

$$D(C) = M$$

$$D(E(M)) = M$$

Where E ( ) for encryption function, D ( ) for decryption function, M is plaintext and C is ciphertext.

2.4 Discrete Logarithm Problem

Discrete Logarithm Problem (DLP) is fundamental to a number of public key algorithms, such as Diffie-Hellman key exchange and the digital signature algorithm. There are some efficient algorithms that can

efficiently reduce the DLP in the *group*. They are:

- Pohlig-Hellman algorithm [1]
- Pollard's ρ method [2]
- Index-calculus methods [3]

Consider the following equation:

$$y = g^x \mod n$$

Where g, x, n are the elements in the *group*.

Calculate y by given g, x, n is straightforward. However, given y, g, n, it is very difficult to calculate x, which is the discrete logarithm problem. For example in elliptic curve DLP, Pollard's ρ method has an expected running time of $\sqrt{\pi n/2}$ elliptic curve operations, which is the best general purpose algorithm known.

2.5 Comparison between ECC and RSA

In the public key cryptosystems, there are two main encryption algorithm, they are RSA and ECC.

RSA was published by Ron Rivest, Adi Shamir, and Len Adleman at MIT in 1978 [4]. This algorithm is a block cipher in which the plaintext and ciphertext are integers between 0 and n – 1 for some n. In the generation of the key pair, it involves two prime numbers, which is sufficiently large to prevent the discovery of the keys by exhaustive search method. Nowadays, the commercial systems use 1024-bit RSA for encryption applications.

ECC was proposed by Neal Koblitz [6] and Victor Miller [7] in 1985. This algorithm based on the "addition" properties of the elliptic curve.

Details of the elliptic curve will be introduced in Chapter 4. The key size used in ECC is much shorter than RSA with approximately the same security level. The amount of work required to solve 128-bit ECC was about fifty times that required to solve the 512-bit RSA. Therefore, ECC will become popular in public key cryptosystems.

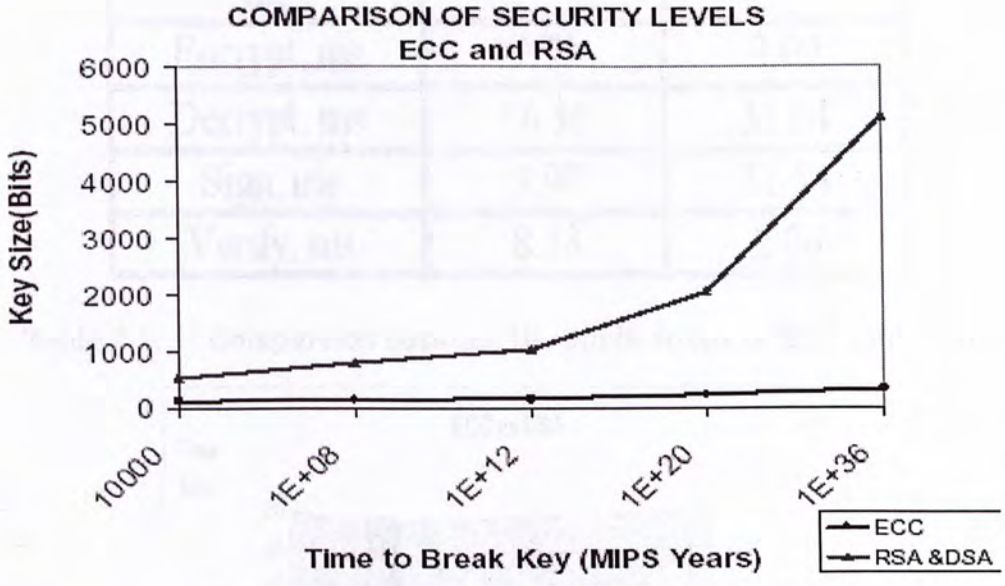


Figure 2.4. Comparison of security levels

(Source from www.certicom.com)

Figure 2.4 shown above is the security strength of ECC and RSA against the time to break. In the graph, we can see that ECC has a potential to replace RSA in the coming future due to its high security strength per bit. Table 2.1 and Figure 2.5 show the time used for different applications of ECC and RSA. The overall performance of ECC is better than RSA due to its shorter key length, especially in the key generation. In the conclusion, RSA is a mature algorithm and commonly

used in the commercial systems. However, ECC is a new mathematical algorithm, which has a large improvement in the research area.

API Function	ECC Engine (163 bits)	RSA Engine (1024 bits)
Key generation, ms	3.19	681.63
Encrypt, ms	9.72	2.09
Decrypt, ms	16.36	31.64
Sign, ms	3.97	31.54
Verify, ms	8.33	2.06

Table 2.1. Comparison between the applications of ECC and RSA

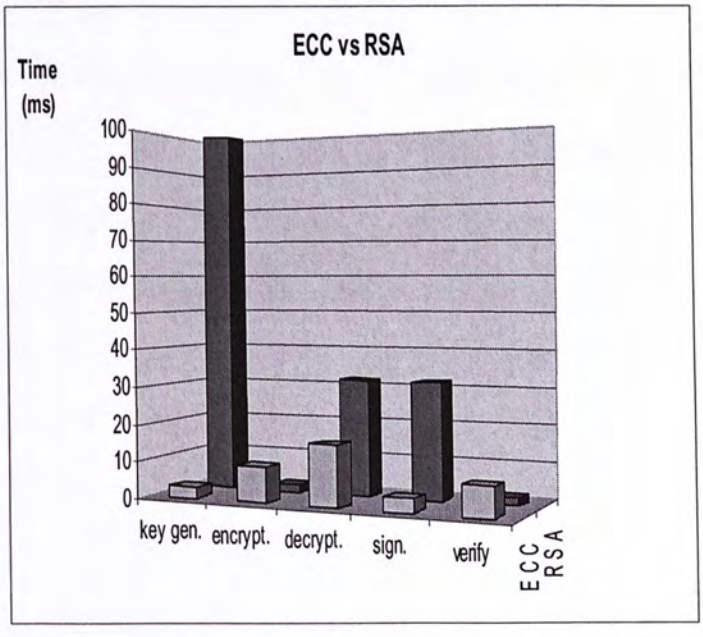


Figure 2.5. The plot of the applications of ECC and RSA

2.6 Summary

This chapter has given an introduction about the topics of cryptography, the different between symmetric and asymmetric key cryptography, the applications of the public key cryptography, and the discrete logarithm problem about the cryptography. At last, the comparison between two major public key cryptography algorithms was presented.

Chapter 3 Mathematical Background in Number Systems

3.1 Introduction to Number Systems

Elliptic curve mathematics is based on abstract algebra in particular finite fields. In this chapter, some basic concepts of Groups, Rings and Fields are introduced. Then follow with the modular arithmetic used in Finite Fields. Finally, the arithmetic operations in the optimal normal basis mathematics are discussed in details.

3.2 Groups, Rings and Fields

Group theory is concerned with systems in which always have a unique solution. It requires only that a mathematical system obey a few simple rules. The theory then seeks to find out properties common to all systems that obey these few rules.

The axioms for a group are:

- Closure: If a and b are in the group then $a \cdot b$ is also in the group.
- Associativity: If a, b and c are in the group then
  - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- Identity: There is an element e of the group such that for any element a of the group $a \cdot e = e \cdot a = a$.
- Inverse: For any element a of the group there is an element a^{-1} such that

- $a \cdot a^{-1} = e$, and
- $a^{-1} \cdot a = e$

The theory does not concern itself with what a and b actually are nor with what the operation symbolized by "$\cdot$" actually is. For example, if the group operation is replaced by multiplication "$\times$", then the group is said to be multiplicative group.

The group is called abelian if the operation is commutative, i.e.

$$a \cdot b = b \cdot a$$

A ring $(R,+,\times)$ consists of a set R, addition operation "+", and multiplication operation "$\times$" on R.

The axioms of a ring are:

- $(R,+,\times, 0)$ is and abelian group.
- The operation $\times$ is associative. i.e. $(a \times b) \times c = a \times (b \times c)$.
- There exists an identity 1 such that $1 \times a = a \times 1 = a$.
- The operation $\times$ is distributive over +. i.e.
  - $a \times (b + c) = (a \times b) + (a \times c)$
  - $(b + c) \times a = (b \times a) + (c \times a)$

A field is a commutative ring in which every non-zero element has a multiplicative inverse. i.e. $a^{-1} \times a = a \times a^{-1} = 1$ for $a \neq 0$.

3.3 Finite Fields

A finite field is a field which contains a finite number of elements with several properties:

- Every element in the finite field has an inverse.
- The finite field contains an additive identity element in the rules of addition.
- The finite field contains a multiplicative identity element in the rules of multiplication.

3.4 Modular Arithmetic

In integer field, if b is any integer and n is a positive integer, we write b mod n for the remainder in {0, ..., n-1} that occurs if b is divided by n. Suppose a is a integer in {0, ..., n-1}, we call a, b congruent modulo n, written as a = b (mod n) if and only if one of the following equivalent conditions holds:

- their difference is divisible by n.
- they leave the same remainder when divided by n.
- a – b = kn for some integer k.
- a – b in the ideal of all integers divisible by n.

3.5 Optimal Normal Basis

Optimal normal basis were considered the fastest implementation method in the elliptic curve cryptosystems. Only AND, XOR and rotation operations are needed, which are very efficient in hardware. In the following sub-sections, the theory of normal basis mathematics will be described first, and then discuss the theory behind the normal basis multiplication and inversion.

3.5.1 What is a Normal Basis?

A normal basis representation can be described with algebraic polynomials. Suppose β is an element in the field $F_p{}^m$, the polynomial representation is:

$$\beta = a_n x^n + \ldots + a_1 x + a_0$$

Where n < m.

A normal basis can be formed using the set:

$$\left\{\beta^{p^{m-1}}, \cdots, \beta^{p^2}, \beta^p, \beta\right\}$$

For the ease of hardware implementation, we always use characteristic 2 in the finite field, (i.e. set p = 2, $F_2{}^m$).

Any element e in the finite field $F_2{}^m$ can be represented in a normal basis format:

$$e = e_{m-1}\beta^{2^{m-1}} + \cdots + e_2\beta^{2^2} + e_1\beta^{2^1} + e_0\beta^{2^0} = \sum_{i=0}^{m-1} e_i\beta^{2^i}$$

3.5.2 Addition

The addition in finite field is simple, different from the addition over real number field, only involve XOR logic, no carry path is needed in the operation.

For

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$$

$$B = \sum_{i=0}^{m-1} b_i \beta^{2^i}$$

$$A + B = \sum_{i=0}^{m-1} a_i \beta^{2^i} + \sum_{i=0}^{m-1} b_i \beta^{2^i}$$

$$= \sum_{i=0}^{m-1} (a_i + b_i) \beta^{2^i}$$

3.5.3 Squaring

Squaring is the special case of multiplication. It only involves rotating one bit from least significant bit to most significant bit.

There are two reasons for this:

- $\left(\beta^{2^i}\right)^2 = \beta^{2^{i+1}}$
  - Proof: $\left(\beta^{2^i}\right)^2 = \beta^{2(2^i)} = \beta^{2^{i+1}}$
- $\beta^{2^m} = \beta$
  - Proof: $\beta^{2^m} = \beta^{2^m} \left(\text{mod } F_{\beta^{2^m}} \right) = \beta$

3.5.4 Multiplication

Multiplication over F_{2^m} is defined as follows.

For

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$$

$$B = \sum_{i=0}^{m-1} b_i \beta^{2^i}$$

$$C = A \times B = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i} \beta^{2^j}$$

$$= \sum_{k=0}^{m-1} c_k \beta^{2^k}$$

by rearranging this term:

$$\beta^{2^i} \beta^{2^j} = \sum_{k=0}^{m-1} \lambda_{ijk} \beta^{2^k} = \lambda_{ijk} \sum_{k=0}^{m-1} \beta^{2^k}$$

where λ_{ijk} is called the "lambda matrix", and

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \lambda_{ijk}$$

The mathematicians proved that the above equation can be further simplified to:

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{ij0}$$

3.5.5 Optimal Normal Basis

The lambda matrix described in the previous sections is a m × m multiplication table. The more the zero terms in the table, the more the efficient of the multiplier. We called it an "Optimal Normal Basis (ONB)" if it has the minimum number of nonzero terms.

There are two types of optimal normal basis over F_{2^m}. They are called Type I ONB and Type II ONB. Different types of ONB have different

rules, as shown in follows.

- Type I Optimal Normal Basis over F_{2^m}
  - m + 1 must be prime.
  - 2 must be primitive in Z_{m+1}.
- Type II Optimal Normal Basis over F_{2^m}
  - 2m + 1 is prime
  - and either two of the following
    - 2 is primitive in Z_{2m+1}.
    - $2m + 1 \equiv 3 \bmod 4$ and 2 generates the quadratic residues in Z_{2m+1}.

where Z_m is a set of integers in the range 1 ... m.

Although the theory behind ONB seems very difficult, the generation of the lambda matrix is petty easy. Next section will show you an example on how to generate this matrix over F_{2^m}.

3.5.6 Generation of the Lambda Matrix

Suppose using the field F_{2^4} with m = 4 in ONB representation.

F_{2^4} is a Type I ONB because,

- 4 + 1 = 5 is prime
- 2 is primitive in Z_5.

Then define the irreducible polynomial for F_{2^4}, i.e.

$$f(x) = x^4 + x^3 + x^2 + x + 1$$

and setup the polynomial for normal basis of F_{2^4} over F_2.

$$\left(x, x^2, x^{2^2}, x^{2^3}\right)$$

Next, construct a 4×4 matrix A as follows:

Row 0: $x \mod f(x) = 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 0 \cdot x^0 = (0\ 0\ 1\ 0)$

Row 1: $x^2 \mod f(x) = 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0 = (0\ 1\ 0\ 0)$

Row 2: $x^4 \mod f(x) = 1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0 = (1\ 1\ 1\ 1)$

Row 3: $x^8 \mod f(x) = 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0 = (1\ 0\ 0\ 0)$

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Find A^{-1},

$$A^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Construct another 4×4 matrix T^t as follows:

Row 0: $x \cdot x \mod f(x) = 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0 = (0\ 1\ 0\ 0)$

Row 1: $x \cdot x^2 \mod f(x) = 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0 = (1\ 0\ 0\ 0)$

Row 2: $x \cdot x^4 \mod f(x) = 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = (0\ 0\ 0\ 1)$

Row 3: $x \cdot x^8 \mod f(x) = 1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0 = (1\ 1\ 1\ 1)$

$$T^t = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

and calculate, $T = T^t \times A^{-1}$

$$T = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The lambda matrix can be found by:

$$\lambda = \begin{pmatrix} T(0,0) & T(3,3) & T(2,2) & T(1,1) \\ T(1,0) & T(0,3) & T(3,2) & T(2,1) \\ T(2,0) & T(1,3) & T(0,2) & T(3,1) \\ T(3,0) & T(2,3) & T(1,2) & T(0,1) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Actually, this lambda matrix is a table used for wiring the circuit like that, suppose $A=\{a_0,a_1,a_2,a_3\}, B=\{b_0,b_1,b_2,b_3\}$ and $C=\{c_0,c_1,c_2,c_3\}$

$$\begin{array}{c} \\ a_0 \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{cccc} b_0 & b_1 & b_2 & b_3 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

$$c_0 = a_0b_2 \oplus a_1(b_2 \oplus b_3) \oplus a_2(b_0 \oplus b_1) \oplus a_3(b_1 \oplus b_3)$$

$$c_1 = a_1b_3 \oplus a_2(b_3 \oplus b_0) \oplus a_3(b_1 \oplus b_2) \oplus a_0(b_2 \oplus b_0)$$

$$c_2 = a_2b_0 \oplus a_3(b_0 \oplus b_1) \oplus a_0(b_2 \oplus b_3) \oplus a_1(b_3 \oplus b_1)$$

$$c_3 = a_3b_1 \oplus a_0(b_1 \oplus b_2) \oplus a_1(b_3 \oplus b_0) \oplus a_2(b_0 \oplus b_2)$$

The example shown above is the method used to generate the ONB multiplication table. This matrix only give the information in the connection between the input A, input B and the output C, which can be done using Matlab simulation.

3.5.7 Inversion

Suppose a is an element over F_{2^m}, the inversion of a is denoted by a^{-1} and defined as:

$$a \times a^{-1} \equiv 1$$

By Fermat's Theorem [5]:

$$a^{-1} = a^{2^m-2} = \left(a^{2^{m-1}-1}\right)^2$$

If m is odd,

By considering the exponent, it can be factorized to:

$$2^{m-1}-1=\left(2^{(m-1)/2}-1\right)\left(2^{(m-1)/2}+1\right)$$

$$\therefore a^{2^{m-1}-1}=a^{\left(2^{(m-1)/2}-1\right)\left(2^{(m-1)/2}+1\right)} \qquad \text{Eq. 3.1}$$

Recursively decompose term $a^{2^{m-1}-1}$ until m = 2 (i.e. $a^{2^{2-1}-1}=a$), the algorithm can be solved. In this case, we only need to compute $a^{2^{(m-1)/2}+1}$, and computing this term is easy, just rotate it by (m-1)/2 bit to left.

It is little complicated when m is even, the exponent can be factorized to:

$$2^{m-1}-1=2\left(2^{(m-1)/2}-1\right)\left(2^{(m-1)/2}+1\right)+1$$

substitute it back to a, the equation will become:

$$a^{2^{m-1}-1}=\left(a^{\left(2^{(m-2)/2}-1\right)\left(2^{(m-2)/2}+1\right)}\right)^2\cdot a \qquad \text{Eq. 3.2}$$

this means, an extra multiplication and shifting operations are needed to compute the result.

Since this research is to implement a 173-bit Elliptic Curve Crypto-processor, let's pick m = 173 as an example to describe this inversion algorithm in practice.

Since

$a^{-1}=a^{2^{173}-2}$

$a^{2^{173}-2}=a^{2\cdot\left(2^{86}-1\right)\left(2^{86}+1\right)}$ apply Eq. 3.1 ∵ 173 is odd

$a^{2^{86}-1}=a^{\left(2^{43}-1\right)\left(2^{43}+1\right)}$ apply Eq. 3.2 ∵ 86 is even

$a^{2^{43}-1}=a^{2\cdot\left(2^{21}-1\right)\left(2^{21}+1\right)+1}$ apply Eq. 3.1 ∵ 43 is odd

$a^{2^{21}-1}=a^{2\cdot\left(2^{10}-1\right)\left(2^{10}+1\right)+1}$ apply Eq. 3.1 ∵ 21 is odd

$a^{2^{10}-1} = a^{(2^5-1)(2^5+1)}$ apply Eq. 3.2 ∵ 10 is even

$a^{2^5-1} = a^{2\cdot(2^2-1)(2^1+1)+1}$ apply Eq. 3.1 ∵ 5 is odd

$a^{2^2-1} = a^{(2^1-1)(2^1+1)} = a^{(2+1)}$ apply Eq. 3.2 ∵ 2 is even

For m=173 bit, it takes 10 multiplication operations.

The pseudo-code is shown below:

Given a in F_{2^m}, find a^{-1}.

Convert m into binary format

s = (number of bits in m) - 1

Set temp1 = a

For i = s downto 0

Set temp2 = temp1

Set shift = shift m to right by s bit(s)

Rotate temp2 to left "truncate (shift)" bit(s)

temp1 = temp1 × temp2

If shift is odd

Rotate temp1 to left 1 bit

temp1 = temp1 × a

End

End

Set a^{-1} = Rotate temp1 to left 1 bit

3.6 Summary

This chapter has discussed about the theory of the ONB finite field arithmetic. It was considered the fastest implementation method in the

elliptic curve cryptography at the present time. Furthermore, the efficient algorithm in ONB arithmetic also provides the benefit for low power consumption and small area design. This research has taken an advantage of these to implement a 173-bit Type II ONB elliptic curve cryptosystem.

Chapter 4 Introduction to Elliptic Curve Mathematics

4.1 Introduction

The Elliptic Curve theory has been well studied by mathematicians for many years, and it has yielded some significant results in an application of the cryptographic field in the later 20th century. Actually, these curves are the simple functions that contains x and y coordinates. Besides real number system, these curves can also be applied into finite field arithmetic over polynomial basis and optimal normal basis number systems.

In this chapter, the elliptic curves over real number system will be introduced first as it is easier to interpret. Afterwards, the elliptic curves over finite field are detailed. For an efficient hardware implementation, projective coordinates system will be introduced. Finally, the applications in cryptography using the elliptic curves are shown.

4.2 Mathematical Background of Elliptic Curves

The following general equation is the "Weierstrass" form of the elliptic curves, where a_1, a_2, a_3, a_4 and a_6 are the coefficients, x and y cover a plane, this plane can be real, complex, integer, polynomial basis, optimal normal basis and any other kind of field element.

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

In 1985, Koblitz [6] and Miller [7] proposed using elliptic curves for cryptographic purposed. They found that when taking any two points on the same elliptic curve, add them together, the resulting point also lie on the same curve. This is difficult to figure out which two points adding together to get the result, and this is exponential to the key length. Therefore, this discrete logarithm problem on the elliptic curves is an attractive one way function because there is no sub-exponential attack known to solve this problem.

4.3 Elliptic Curve over Real Number System

This section starts with the most familiar elliptic curve over the real number system on a real plane. The following equation is the simple form of elliptic curve, which will use in this section:

$$y^2 = x^3 + a_4 x + a_6$$

By putting $a_4 = -8$ and $a_6 = 8$. Figure 4.1 shows the plot of this curve.

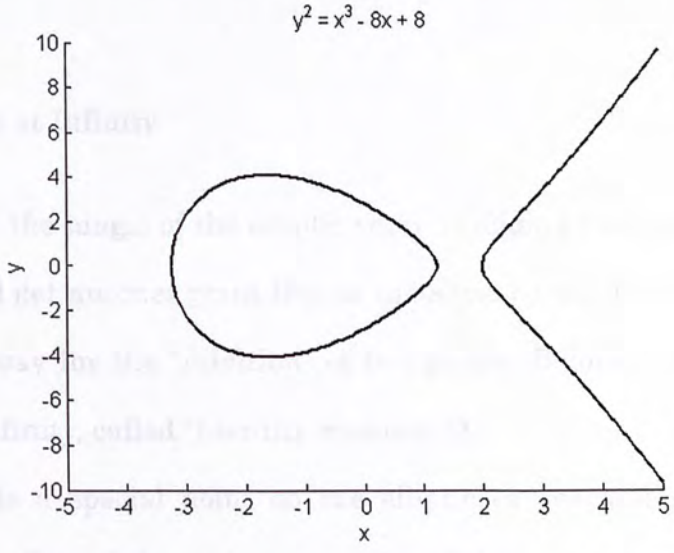


Figure 4.1. Plot of Elliptic Curve: $y^2 = x^3 - 8x + 8$

4.3.1 Order of the Elliptic Curves

The term order of the elliptic curves is defined by the number of elements lie on the elliptic curve over a finite field, and this must satisfy "Hasse's Theorem".

$$|N-(q+1)| \leq 2\sqrt{q}$$

Where N is the order of the curve and q is the field size.

4.3.2 Negation of Point P

In the elliptic curves, for every solvable value of x, there exist two values of y.

$$\pm y = \sqrt{x^3 - 8x + 8}$$

Therefore, the relationship between P and –P is:

If $P = (x, y)$, then $-P = (x, -y)$.

4.3.3 Point at Infinity

Since, the magic of the elliptic curve is adding two point lies on this curve, and get another point lies on the same curve. Therefore, we need to find a way for the "Addition" of two points. Before that, we define a point at infinity, called "Identity element, O_∞".

This is a special point on the elliptic curves and has two major properties. One of the major properties of this point is when adding O_∞ with other points on the curve would give the same point back. i.e.

$$(P) + (O_\infty) = (P)$$

The other property is when adding two points, P and –P, the resulting point will come to the point at infinity, O_∞. See Figure 4.2.

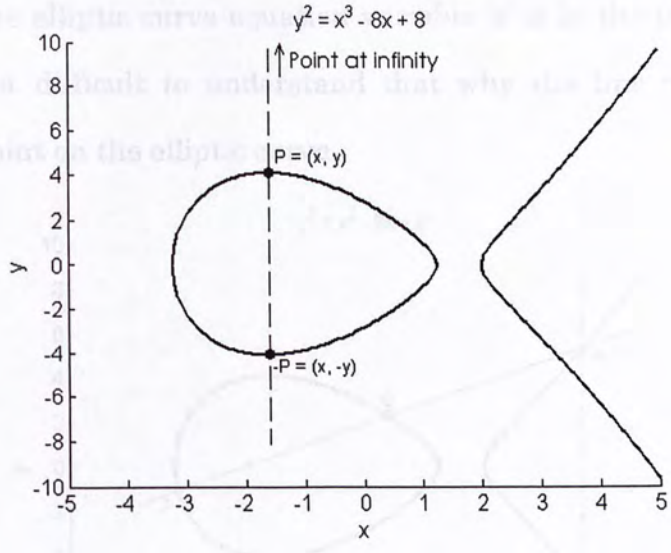


Figure 4.2. Adding P and –P to get the point at infinity

The point at infinity is the special case of the elliptic curves that defined by the mathematicians who want to fulfill all the cases. Under normal conditions, the engineers would avoid the occurrence of this special case.

4.3.4 Elliptic Curve Addition

Elliptic Curve Addition is known as the point addition. In mathematical representation, it is R = P + Q.

Figure 4.3 shows the geometrical relationship of the curve addition. Suppose there are two distinct points P and Q lie on the curve and connecting these two points by a line, it must passing through another point which lies on the same curve –R, by negating the y-coordinate of –R,

the solution R is obtained.

In mathematical approach, by solving the simultaneous equations on the elliptic curve and the straight line, three solutions we can solved because the elliptic curve equation variable 'x' is in the power of three. This is not difficult to understand that why the line must intersect another point on the elliptic curve.

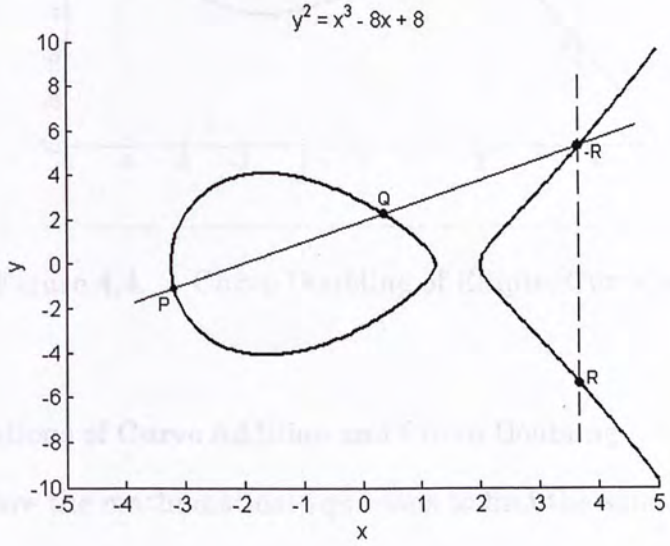


Figure 4.3. Curve Addition of Elliptic Curve points

4.3.5 Elliptic Curve Doubling

Elliptic Curve Doubling is a special case in Curve Addition. The line is tangent to the curve at point P and then intersecting at another point R. This is shown in Figure 4.4.

In mathematical approach, firstly, by differentiating the elliptic curve, and finding the slope of the straight line, i.e. the gradient at point P (x_1, y_1). The coordinates -R can be solved by finding the intersecting point of the elliptic curve and that line.

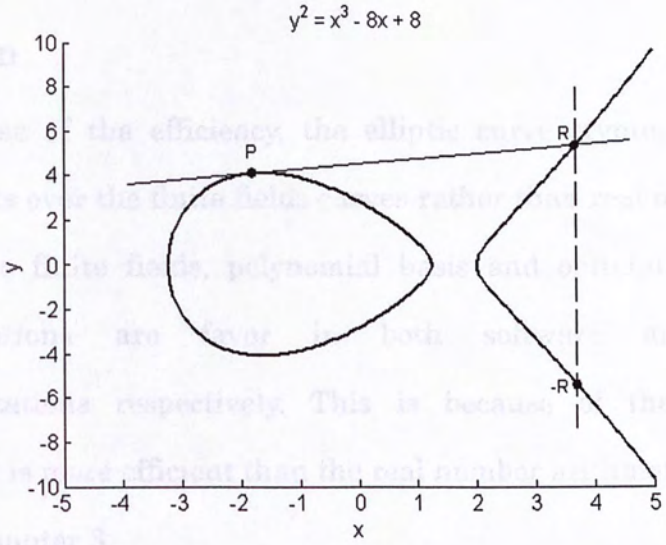


Figure 4.4. Curve Doubling of Elliptic Curve point

4.3.6 Equations of Curve Addition and Curve Doubling

Here are the mathematical equations to find the solution of R:

Let

$$P = (x_1, y_1)$$
$$Q = (x_2, y_2)$$
$$R = (x_3, y_3) = P + Q$$

$$x_3 = \theta^2 - x_1 - x_2$$
$$y_3 = \theta(x_1 + x_3) - y_1$$

where

$$\theta = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{if} \quad P \neq Q \quad (\textit{Curve Addition})$$

or

$$\theta = \frac{3x_1^2 + a_4}{2y_1} \quad \text{if} \quad P = Q \quad (\textit{Curve Doubling})$$

4.4 Elliptic Curve over Finite Fields Number System

Because of the efficiency, the elliptic curve cryptography always implements over the finite fields curves rather than real number curves. Among the finite fields, polynomial basis and optimal normal basis representations are favor in both software and hardware implementations respectively. This is because of the finite fields arithmetic is more efficient than the real number arithmetic. Details can refer to Chapter 3.

Since the arithmetic operations in the finite fields are different from real number arithmetic, there has another set of equations different from the one stated in Chapter 4.3. Moreover, to avoid the cryptanalyst attack the system easily, the mathematicians have proved that using a "non-supersingular" elliptic curve could provide a maximum benefit of security.

Here is the general form of "non-supersingular" elliptic curve.

$$y^2 + xy = x^3 + a_2x^2 + a_6$$

Where a_2, a_6 are finite field number and $a_6 \neq 0$.

4.4.1 Elliptic Curve Operations in Optimal Normal Basis Number System

This section shows the set of equations of elliptic curves using in the optimal normal basis arithmetic.

Let

$$P = (x_1, y_1)$$
$$Q = (x_2, y_2)$$
$$R = (x_3, y_3)$$

For curve addition, $P \neq Q$; $R = P + Q$

$$\theta = \frac{y_2 - y_1}{x_2 - x_1}$$
$$x_3 = \theta^2 + \theta + x_1 + x_2 + a_2$$
$$y_3 = \theta(x_1 + x_3) + x_3 - y_1$$

For curve doubling, $P = Q$; $R = 2Q$

$$\theta = x_1 + \frac{y}{x_1}$$
$$x_3 = \theta^2 + \theta + a_2$$
$$y_3 = x_1^2 + (\theta + 1)x_3$$

The number of finite field operations involved in the curve operations is shown in Table 4.1.

4.4.2 Elliptic Curve Operations in Projective Coordinates

In the previous section, the set of curve operation equations are in affine coordinates (i.e. only x, y axis are involved). In affine coordinates, both curve addition and doubling consist of one field inversion operation. As we have discussed in Chapter 3, field inversion process is the most time consuming process in field operation, to eliminate this field inversion in every curve operations, a projective coordinates system is suggested in the implementation of elliptic curve.

Projective coordinates system consists of x, y, z axis, every point in affine coordinates (x, y) can be converted to projective coordinates (x, y, z) and vice versa. The conversion methods are given as follows:

Affine coordinates to Projective coordinates

$$P(x,y) \rightarrow P'(x,y,1)$$

Projective coordinates to Affine coordinates

$$Q'(x,y,z) \rightarrow Q'\left(\frac{x}{z},\frac{y}{z},1\right) \rightarrow Q(x,y)$$

where P, Q and P', Q' are in affine coordinates and projective coordinates respectively.

The conversion from affine coordinates to projective coordinates is easy, just assigning 1 to z-coordinate is enough, no extra operation required. However, when converting projective coordinates to affine coordinates, the z-axis is needed to convert to 1 first by finding the inverse of z, and then multiply the inverse of z to x and y coordinates respectively as shown above.

4.4.3 Elliptic Curve Equations in Projective Coordinates

This section will show the equations of the curve operations in the projective coordinates.

Let

$$P = (x_1, y_1, z_1)$$
$$Q = (x_2, y_2, z_2)$$
$$R = (x_3, y_3, z_3)$$

For curve addition, $P \neq Q$; $R = P + Q$

$$A = x_2 z_1 + x_1$$
$$B = y_2 z_1 + y_1$$
$$C = A + B$$
$$D = A^2(A + az_1) + z_1 BC$$

$$x_3 = AD$$
$$y_3 = CD + A^2(Bx_1 + Ay_1)$$
$$z_3 = A^3 z_1$$

For curve doubling, $P = Q$; $R = 2Q$

$$A = x_2 z_2$$
$$B = a_6 z_2^4 + x_2^4$$
$$x_3 = AB$$
$$y_3 = x_2^4 A + B(x_2^2 + y_2 z_2 + A)$$
$$z_3 = A^3$$

	Affine Coordinates		Projective Coordinates	
	Addition	Doubling	Addition	Doubling
Field Addition	9	5	7	4
Field Squaring	1	2	1	6
Field Multiplication	2	2	13	7
Field Inversion	1	1	0	0

Table 4.1. The relationship between finite field operation and curve operation in different coordinates system

In the above table, we can conclude that using projective coordinates should have a better performance that affine coordinates. It was because in implementing 173 bit elliptic curve as an example, one field inversion operation consists of ten field multiplication operations, although the number of multiplications in projective coordinates increased, the overall

performance can be increased.

4.5 Curve Multiplication

The Elliptic Curve Multiplication is defined by repeating the Elliptic Curve Addition. The rule of this multiplication is described as follows.

Let

$$P = (x_1, y_1)$$
$$Q = (x_2, y_2)$$

For

$$Q = c \times P$$
$$Q = P + P + \cdots + P \qquad (c \ \ times)$$

Where P, Q are the points lie on the elliptic curve over affine coordinates, and c is an integer.

This is the definition of the curve multiplication. In the computation algorithm, we can implement this by curve addition and curve doubling operations. The algorithm can be divided into 4 steps and stated below.

Step 1: convert the integer c into binary form.

Step 2: count down the integer c bit by bit and start from the second MSB to LSB.

Step 3: if the bit is '1', then compute curve doubling and then curve addition; if the bit is '0', then compute curve doubling only.

Step 4: repeat Step 2 and Step 3 until the LSB is computed.

For example, suppose we want to compute 21 P.

Step 1: integer c = 21 = 10101_{ii}

Step 2: ∵ 2nd MSB = 0, then compute 2P by doubling.

Step 3: ∵ 3rd MSB = 1, then compute 4P and 5P by doubling and

addition respectively.

Step 4: ∵ 4th MSB = 0, then compute 10P by doubling.

Step 5: ∵ LSB = 1, then compute 20P and 21P by doubling and addition respectively. And the result 21P is obtained.

When implementing the curve multiplication over projective coordinates, there have a conversion from projective coordinates back to affine coordinates once the multiplication process is done.

4.6 Elliptic Curve Discrete Logarithm Problem

Let $P(x,y)$ be a point lies on elliptic curve of order n. The Elliptic Curve Discrete Logarithm Problem (ECDLP) is to find the integer k, for $0 \leq k \leq n-1$, such that $Q = kP$.

There is no index-calculus method known for solving the ECDLP that has a sub-exponential running time by given Q and P to compute k. However, the most efficient algorithm known for solving the ECDLP is using Pollard's ρ method [8]. This algorithm was improved by Gallant, Lambert and Vanstone [9] and Wiener and Zuccherato [10] which has an expected running time of $\sqrt{\pi n}/2$ elliptic curve operations. Therefore, the known methods for computing ECDLP are still exponential in n and less efficient than those for factoring and discrete logarithm problem used in other cryptography attacking approach. As a result, Elliptic Curve Cryptography could be a better cryptosystem over the others with the same key sizes.

4.7 Public-key Cryptography in Elliptic Curve Cryptosystem

The elliptic curve cryptosystem can be used as a public key cryptosystem. When the parties want to send or receive the private information through the insecure channel using ECC, they need to know the sender or receiver's public key. Therefore, ECC key pairs are generated before the communication.

In the elliptic curve cryptosystems, the key pair generation method is defined as follows.

Step 1: define a set of elliptic curve domain parameters, include elliptic curve equation, (i.e. a_2 and a_6.) and generator point G (x, y), (i.e. the initial point which lies on the specified elliptic curve). And these parameters are set public to everyone.

Step 2: select a random number k where $1 \leqq k \leqq n - 1$ (n: order of the curve).

Step 3: Compute Q = kG. (Q is set public to everyone.)

Then, k and Q are the private key and public key respectively.

4.8 Diffie-Hellman Key Exchange in Elliptic Curve Cryptosystem

Diffie-Hellman Key Exchange scheme is the first public key algorithm invented by Diffie and Hellman. The purpose of this algorithm is to enable two users to exchange a session key (secret key) that can be

used for subsequent encryption of the messages. The generation method of the session key is defined as follows.

Step 1: suppose there are two users Alice and Bob who have already known the elliptic curve domain parameters.

Step 2: then, both of them generate their own public key pair first. (i.e. Alice: k_a, Q_a; Bob: k_b, Q_b).

Step 3: they can send their own public key (Q_a and Q_b) to the other.

Step 4: Once they exchange their public key, they can compute the session key by multiplying their own private key with the other's public key like that:

$$session\ \ key = k_a \times Q_b = k_b \times Q_a$$

4.9 Summary

In this chapter, the idea of elliptic curve over real number and finite field coordinates systems were introduced. And then follow with the comparison between the number of operations in the affine coordinates and the projective coordinates over the finite fields. We can conclude that using projective coordinates system is more efficient than affine coordinates system. Afterwards, two of the elliptic curve applications, public key cryptography and Diffie-Hellman key exchange, were described. And these applications still have no known sub-exponential time algorithm to solve.

Chapter 5 Design Architecture

5.1 Introduction

In this chapter, a new implementation of the 173-bit elliptic curve over type II ONB is presented. This new architecture is based on the optimization of the mathematical equations to reduce the number of multiplication used. Furthermore, the combination of the field operations into single instruction would reduce the frequently fetch and retrieved data between ALU and register unit. The following sections will dissect the research part by part.

5.2 Criteria for the Low Power System Design

One of the objectives of this research is low power consumption. It has many different ways to achieve this goal. They are summarized as follows:

Use of efficient algorithm:

In this research, Type II ONB is chosen rather than using polynomial basis representation or even prime field. It was because the multiplication operation is the most efficient one than the others, and the inversion operation can be avoid by using projective coordinates. Moreover, the newly proposed idea has simplified the mathematical equations to save one multiplication operation in curve addition. These improvements directly reduce the number of operations in the

computation of the results, so that, the overall power consumption can be reduced.

Using standard cell library:

This research is using 0.35um technology standard cell library provided by Austria Micro System (AMS). Compared with the other hardware design approach such as FPGA, the standard cell library provided a more power efficient solution by minimizing the routing distance of the wires between the logic gates which would reduce the capacitive loading in the metal lines. Furthermore, the standard cell library provided by the manufacture is compact, especially the complex gates. This is also a benefit to high speed low power design.

Reduce the switching activities of the logic:

For digital design, the power consumption can be expressed by this formula:

$$P = afCV^2$$

where a is the switching activities of the logic gates, f is the operating frequency of the system, C is the overall parasitic capacitance, and V is the operating voltage.

Since the switching activities of the logic is directly proportional to the power consumption, minimize the variable a is a possible solution.

5.3 Simplification in ONB Curve Addition Equations over Projective Coordinates

Recall the elliptic curve mathematical equation over projective coordinates described in Chapter 4.4.3.

Curve Addition: P ≠ Q

$$A = x_2 z_1 + x_1 \quad \text{Eq. 5.1}$$
$$B = y_2 z_1 + y_1 \quad \text{Eq. 5.2}$$
$$C = A + B \quad \text{Eq. 5.3}$$
$$D = A^2(A + az_1) + z_1 BC \quad \text{Eq. 5.4}$$
$$x_3 = AD$$
$$y_3 = CD + A^2(Bx_1 + Ay_1) \quad \text{Eq. 5.5}$$
$$z_3 = A^3 z_1$$

The mathematical expression shown above is commonly used equations in the implementation of ONB elliptic curve cryptosystems nowadays. This operation involves 13 multiplications, 7 additions and 1 squaring.

The equations shown below are the new expression of the curve addition operation.

$$A = x_2 z_1 + x_1 \quad \text{Eq. 5.6}$$
$$B = y_2 z_1 + y_1 \quad \text{Eq. 5.7}$$
$$D = A^2(A + az_1) + z_1(AB + B^2) \quad \text{Eq. 5.8}$$
$$x_3 = AD$$
$$y_3 = AD + B(A^2 x_1 + D) + A^3 y_1 \quad \text{Eq. 5.9}$$
$$z_3 = A^3 z_1$$

This operation involves 12 multiplications, 8 additions and 2 squaring, which can save 1 field multiplication operation. The proof is based on the original expression as shown below.

Put (Eq. 5.3) into (Eq. 5.4), we get (Eq. 5.8)

Put (Eq. 5.3) into (Eq. 5.5) to compute y_3, we have:

$$\begin{aligned} y_3 &= CD + A^2(Bx_1 + Ay_1) \\ &= (A + B)D + A^2(Bx_1 + Ay_1) \\ &= AD + BD + A^2 Bx_1 + A^3 y_1 \\ &= AD + B(A^2 x_1 + D) + A^3 y_1 \end{aligned}$$

which is the same as (Eq. 5.9). Since the term AD and A^3 in y_3, can be calculated in x_3 and z_3 respectively, 1 field multiplication is saved. In conclusion, over 90% of the curve operations involve the field multiplication, reduction of the field multiplication would greatly enhance the overall performance of the system.

5.4 Finite Field Adder Architecture

The finite field operations consist of four major operators. They are addition, squaring, multiplication and inversion. Since, the inversion algorithm is derived by several multiplications and squaring. Therefore, only the other three operators can be implemented on the ALU.

Adding two finite field numbers is simple. The result is just computed by applying bitwise exclusive-OR on two input numbers.

Figure 5.1. The circuit design of the finite field addition

In the figure, 173 XOR gates are used to compute the addition in parallel, $c_i = a_i \oplus b_i$ for i = 0 to 172.

5.5 Finite Field Squaring Architecture

Squaring of the finite field number is a special case of the

multiplication. It can be implemented by using multiplexer to rotate the input 1 bit to left as shown below. In Figure 5.2, 173 multiplexers are used in parallel to select the input a to the output. When the "Ctrl" signal is reset, the output c store the value of a, otherwise, the output c store the value of a^2 which is rotating 1 bit to left.

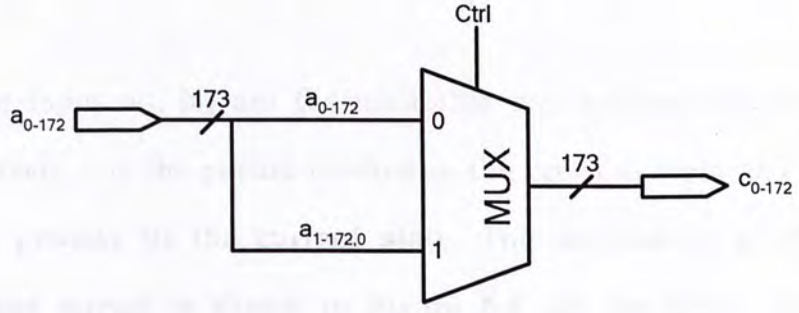


Figure 5.2. The circuit diagram of the finite field squaring

5.6 Finite Field Multiplier Architecture

Since we have discussed about the theory behind the field multiplier in Chapter 3, in this section, the optimized low power consumption GF (2^{173}) parallel multiplication architecture is proposed. The idea of this architecture is efficiently use of Latches in FIFO pipelines by reducing the switching activities in the register unit. When latching the data in the registers, a significant portion of power is consumed in the latches and the clock tree rather than the combinatorial logics, therefore, decreasing the number of switching in flip flops would greatly reduce the overall power consumption. Figure 5.3 shows the combinatorial logic used in the multiplier element. It is a 3-level logic, 3 gate delays are needed to wait for the result. The input of bit pattern $a_{i,j}$ and $b_{i,j}$ to the

multiplier element is based on the lambda matrix (see Appendix).

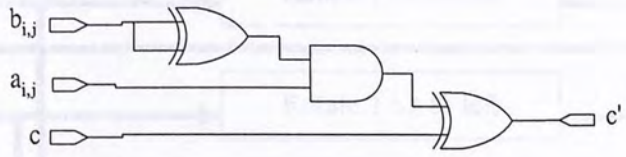


Figure 5.3. The combinatorial logics used in multiplier element

The input $a_{i,j}$, $b_{i,j}$ are the multiplier and multiplicand over F_2^{173} respectively, c is the partial product in the previous state and c' is the partial product in the current state. The mechanism of this field multiplier circuit is shown in Figure 5.4. In the figure, block ME represents "multiplier element" and block FF is the flip flop to hold data. At the beginning, data C is initialized to 0. In the first clock cycle, data A and data B is input into the circuit, the bit pattern of data A and data B then rearrange inside the "wiring" block, and pass into ME and FF. The "wiring" block does not consist any logic cell, it is only a rewiring of the connections based on the lambda matrix described in Chapter 3.5.6. The data stored in the FF is the partial product C' which is ready to feedback in the next cycle. Both A, B and C are needed to rotate 1 bit to left in each cycle as shown in the figure. Since, this is a 173-bit field multiplier, the final product will be stored in C after 173 clock cycles.

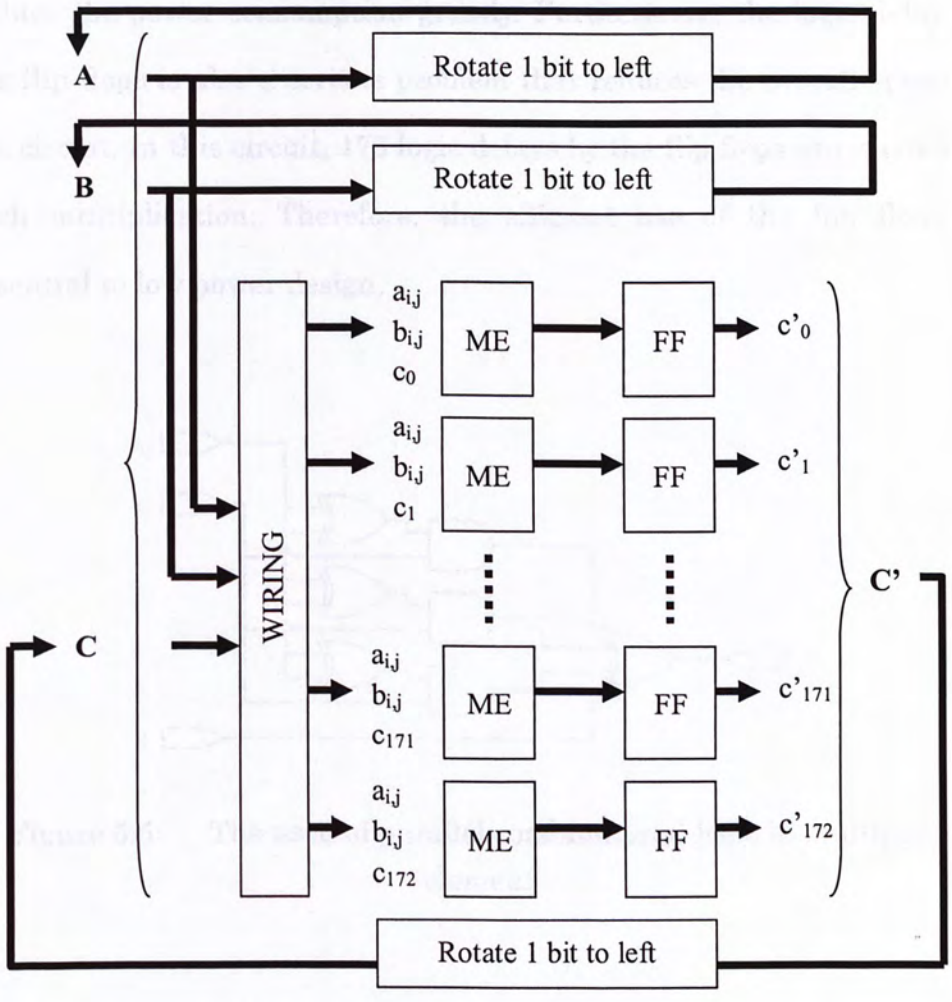


Figure 5.4. The finite field multiplier circuit over F_2^{173}

5.7 3-way Parallel Finite Field Multiplier

Previous section shown is the finite field multiplier proposed by G. B. Agnew [11]. This multiplier has some disadvantages that are not suitable in low power design.

In each multiplication, the flip flops used in the circuit are operated frequently. Since the flip flop is one of the most high power consumption

logic cells, the reduction of the switching activity of the flip flop would reduce the power consumption greatly. Furthermore, the logic delay in the flip flops is also a serious problem that reduces the overall speed of the circuit. In this circuit, 173 logic delays by the flip flops are wasted in each multiplication. Therefore, the efficient use of the flip flops is essential to low power design.

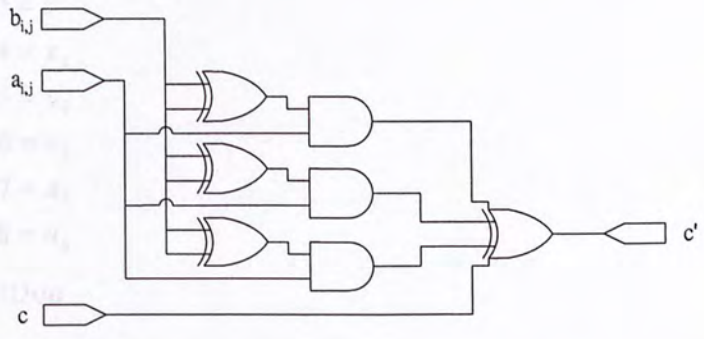


Figure 5.5. The used of parallel combinatorial logic in multiplier element

To overcome these problems, the multiplier element can be modified to compute 3 cycles at a time as shown in Figure 5.5. Simply replace the multiplier element by this one, only 44 clock cycles are used to complete each multiplication. Compare with the original one using 173 clock cycles, over 70% power and time consumption inside flip flops can be saved.

5.8 Finite Field Arithmetic Logic Unit

In this section, a newly proposed finite field ALU is presented. This ALU combines all the field operations into single instruction, so that, the

data transfer between ALU and the registers are greatly reduced. As a result, the power consumption and the propagation delay due to the registers can be minimized.

Refer to the mathematical equations over projective coordinates, they can be arranged into the form: $Q = A \times B + C$

Let

$$R1 = x_1$$
$$R2 = y_1$$
$$R3 = z_1$$
$$R4 = x_2$$
$$R5 = y_2$$
$$R6 = z_2$$
$$R7 = a_2$$
$$R8 = a_6$$

Curve Addition

$$R9 = R1 \times R6 + R4$$
$$R10 = R2 \times R6 + R5$$
$$R11 = R7 \times R6 + R9$$
$$R12 = R9 \times R10 + R10^2$$
$$R12 = R12 \times R6 + 0$$
$$R12 = R9^2 \times R11 + R12$$
$$R11 = R9 \times R12 + 0$$
$$R12 = R9^2 \times R4 + R12$$
$$R12 = R10 \times R12 + R11$$
$$R9 = R9^2 \times R9 + 0$$
$$R5 = R9 \times R5 + R12$$
$$R6 = R9 \times R6 + 0$$
$$R4 = R11$$

Curve Doubling

$$R9 = R4 \times R6 + 0$$
$$R10 = R6^4 \times R8 + R4^4$$
$$R11 = R9 \times R10 + 0$$
$$R12 = R5 \times R6 + R4^2$$
$$R12 = R10 \times R12 + R11$$
$$R5 = R4^4 \times R9 + R12$$
$$R6 = R9^2 \times R9 + 0$$
$$R4 = R11$$

The code shown above is the computation of the curve operations. It can be seen that implementing the function: $Q = A \times B + C$ is enough for the curve operations. Combining these operations would not increase the hardware usage, and the execution time would be decreased due to the "squaring" operation is integrated into the proposed architecture, so that an extra clock cycle for the squaring process is omitted. Figure 5.6 shows the circuit diagram of the ALU.

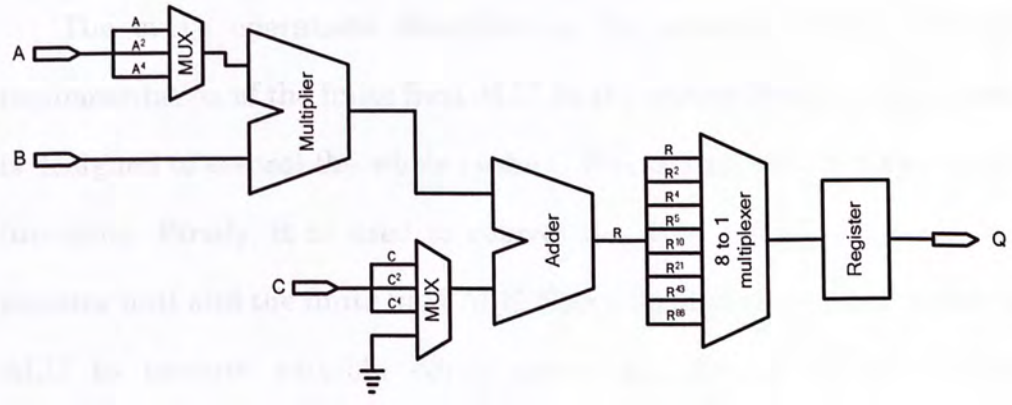


Figure 5.6. The finite field arithmetic logic unit

The function of the 8 to 1 multiplexer is designed for the inversion operation to avoid the continuous rotating process. The input R, R^2, R^4,

R^5, R^{10}, R^{21}, R^{43} and R^{86} represents the number of bit need to rotate (i.e. R^2 : rotate 2 bit to left, R^4 : rotate 4 bit to left and so on). Since R is a 173 databus, in databus representation, the input of the 8 to 1 multiplexer are:

For R = <0 : 172>

R^2 = <2 : 172, 0 : 1>

R^4 = <4 : 172, 0 : 3>

R^5 = <5 : 172, 0 : 4>

R^{10} = <10 : 172, 0 : 9>

R^{21} = <21 : 172, 0 : 20>

R^{43} = <43 : 172, 0 : 42>

R^{86} = <86 : 172, 0 : 85>

5.9 Elliptic Curve Crypto-processor Control Unit

The curve operations described in the previous section are the implementation of the finite field ALU. In the system level, a control unit is designed to control the whole system. This control unit has two major functions. Firstly, it is used to control the data transfer between the register unit and the finite field ALU. Secondly, it controls the finite field ALU to execute suitable curve operations. Figure 5.7 shows the architecture of the elliptic curve crypto-processor control unit.

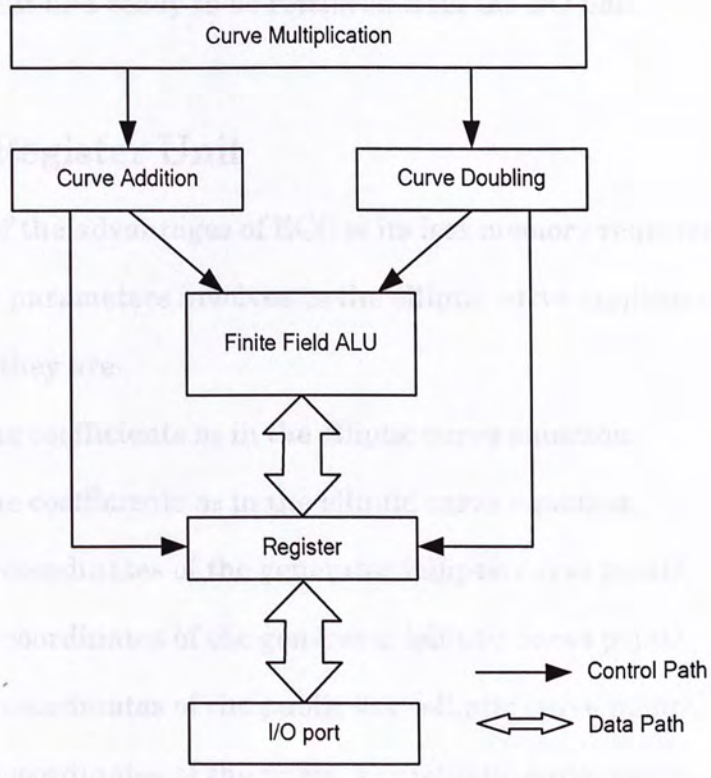


Figure 5.7. The architecture of Elliptic Curve Crypto-processor control unit

First of all, when a start signal received, the processor performs the Curve Multiplication (CM) based on the private key stored in the register unit. When the current bit of the private key is set, CM will send a start signal to Curve Doubling (CD) block and then Curve Addition (CA) block after CD is done. When the current bit of the private key is reset, CM will only send a start signal to CD block to compute curve doubling, no curve addition is needed.

Both CD and CA blocks are the control units to control the relative data fetch and retrieved from the register unit and also control the computation inside the finite field ALU.

When the calculation is complete, the results will store in the

register unit and ready to be retrieved from the I/O pad.

5.10 Register Unit

One of the advantages of ECC is its less memory requirements, there are only 8 parameters involves in the elliptic curve cryptosystems in this research, they are:

- the coefficients a_2 in the elliptic curve equation,
- the coefficients a_6 in the elliptic curve equation,
- x-coordinates of the generator (elliptic curve point),
- y-coordinates of the generator (elliptic curve point),
- x-coordinates of the public key (elliptic curve point),
- y-coordinates of the public key (elliptic curve point),
- z-coordinates of the public key (elliptic curve point), and
- the private key which is a 173 bit random number.

Including 4 extra variables to store the temporary results during the curve operations, totally only 173×12=2076 bit memory is needed for 173-bit ECC. This is extremely low memory usage compared with 1024-RSA. Figure 5.8 shows the register allocation inside the crypto-processor.

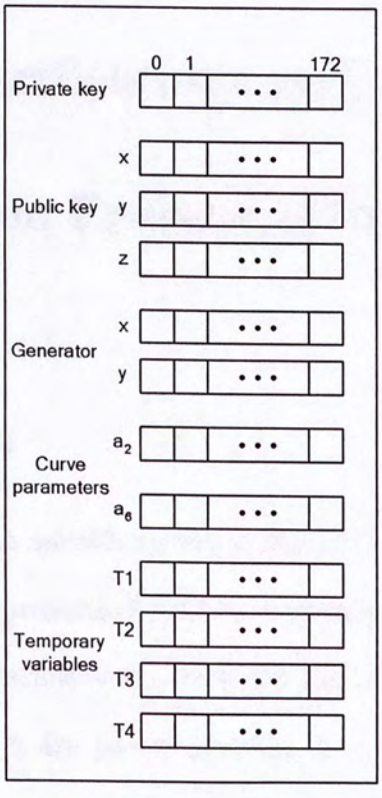


Figure 5.8. Configuration of the register unit

5.11 Summary

In this chapter, the idea of the design methodology is proposed. It begins with the criteria essential for the low power design, and then based on these criteria we improve the processor so that it can fulfill the requirements. By modifying the mathematical expressions in the elliptic curve equations and reducing the switching activities of the logic gates, the processor is a low power consumption one.

Chapter 6 Specifications and Communication Protocol of the IC

6.1 Introduction

In this chapter, the specifications of the elliptic curve cryptosystem integrated circuit are presented. This integrated circuit was fabricated using AMS 0.35um technology with 48-pin dual-in-line packaging. Among these 48 pins, 5 are power sources, 5 are grounds, 16 are data I/Os and the remaining 22 are the control signals. The communication protocol will be described after the specification.

6.2 Specifications

The system architecture of the design is shown in Figure 6.1 and Figure 6.2 shown is the pin assignment, the description of the pins can be found in Table 6.1.

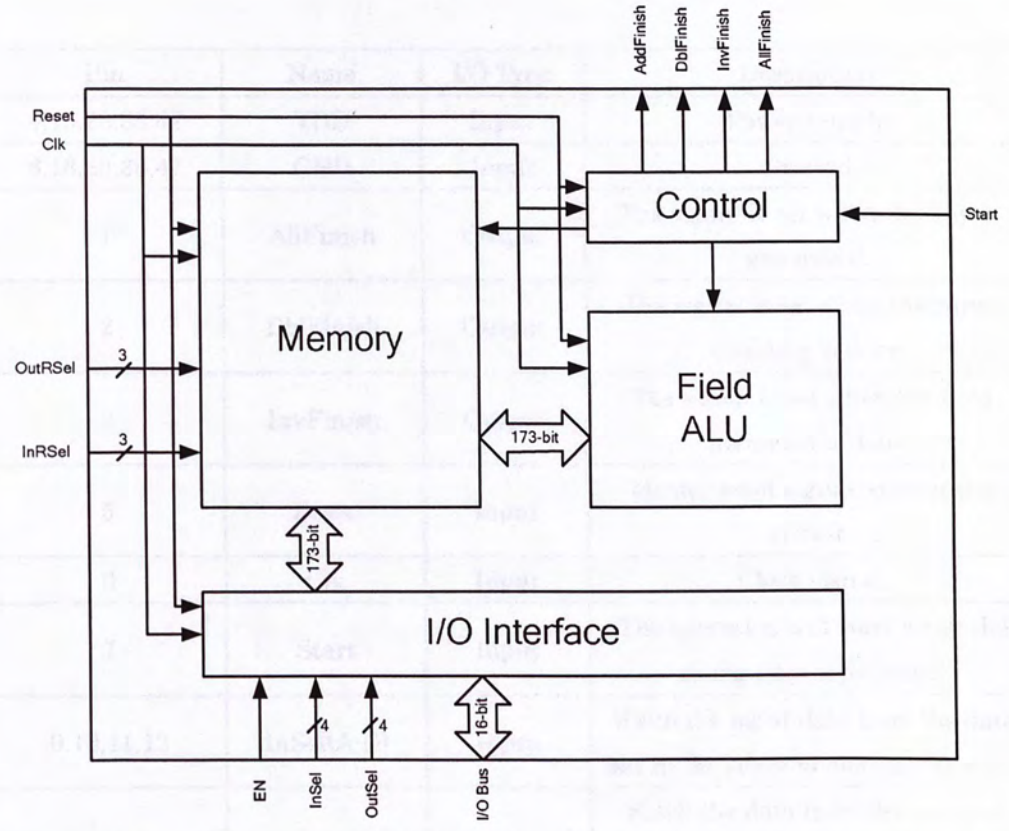


Figure 6.1. The system architecture of the Elliptic Curve cryptosystem

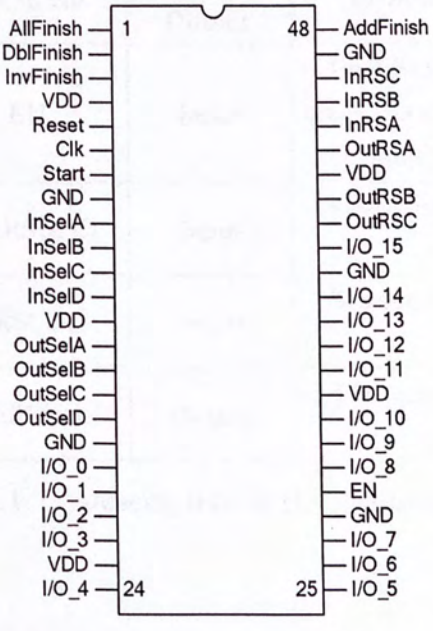


Figure 6.2. Pin assignment of the Elliptic Curve integrated circuit

Pin	Name	I/O Type	Description
4,13,23,33,42	VDD	Input	Power Supply
8,18,28,38,47	GND	Input	Ground
1	AllFinish	Output	The signal is set when the key is generated.
2	DblFinish	Output	The signal is set when the curve doubling is done.
3	InvFinish	Output	The signal is set when the field inversion is done.
5	Reset	Input	Master reset signal to reset the circuit.
6	Clk	Input	Clock signal.
7	Start	Input	The operation will start when the rising edge is detected.
9,10,11,12	InSel(A-D)	Input	Write the input data from the data bus to the selected memory location.
14,15,16,17	OutSel(A-D)	Input	Fetch the data from the selected memory location to the output data bus.
19-22,24-27, 30-32,34-37,39	I/O_(0-15)	Input/ Output	16-bit bidirectional data bus.
29	EN	Input	Data bus load data from the pin when the bit is set, write data to the pin when the bit is reset.
40,41,43	OutRS(A-C)	Input	Select the memory location to be output.
44,45,46	InRS(A-C)	Input	Select the memory location to be written.
48	AddFinish	Output	The signal is set when the curve addition is done.

Table 6.1. Description of the pin assignment

6.3 Communication Protocol

The operation of this integrated circuit is simple. Firstly, prepare all the input parameters, and then load all these data into the register unit and follow with a starting signal, the result will be computed automatically and an acknowledgement signal is given out once the result is complete.

This integrated circuit uses 16-bit databus as a communication path to the peripheral. Since, the data are 173 bit length, each data will be divided into 173/16=11 parts and input to the register unit part by part. Table 6.2 shows the relationship between the control signals and the bits of data to be written in.

InSel / OutSel				Data written
A	B	C	D	
0	0	0	0	Bit (15 ... 0)
0	0	0	1	Bit (31 ... 16)
0	0	1	0	Bit (47 ... 32)
0	0	1	1	Bit (63 ... 48)
0	1	0	0	Bit (79 ... 64)
0	1	0	1	Bit (95 ... 80)
0	1	1	0	Bit (111 ... 96)
0	1	1	1	Bit (127 ... 112)
1	0	0	0	Bit (143 ... 128)
1	0	0	1	Bit (159 ... 144)
1	0	1	0	Bit (172 ... 160)
1	0	1	1	Unused
1	1	X	X	Unused

Table 6.2. The relationship between the control and the mapping of data

Several parameters such as the generator, a_2, a_6 and the private key should be loaded into the register unit before the computation start. Table 6.3 shows how these parameters loaded into the register unit. When generating the public key pair, x, y-coordinates of the generator should load into the address 010 and 011 respectively, otherwise, x, y-coordinates of the public key should load into that address when computing the Diffie-Hellman key exchange.

InRS / Out RS			Parameter written into the register unit
A	B	C	
0	0	0	x-coordinate of the generator
0	0	1	y-coordinate of the generator
0	1	0	x-coordinate of the generator or the public key
0	1	1	y-coordinate of the generator or the public key
1	0	0	a_2
1	0	1	a_6
1	1	0	The private key
1	1	1	unused

Table 6.3. The relationship between the control and the parameters written into the register unit

After all the parameters have loaded into the circuit already, the operation can be activated by feeding a pulse into the "Start" signal. When the result is complete, the "AllFinish" signal will change to logic high.

Chapter 7 Results

7.1 Introduction

In this chapter, a post-layout simulation of the public key generation and the session key generation is presented. The simulation is conducted by Verilog-XL in Cadence under 3 V power supply at 27℃, the operating frequency is 20MHz. In the following sections, the private keys are randomly chosen and the numbers are represented in hexadecimal form with MSB on the LHS.

The measurement results will also be given in this chapter, including the comparison between the post-layout simulation result and the testing result, and the power consumption of this crypto-processor. Some discussions about this crypto-processor are given in the end of this chapter.

7.2 Results of the Public-key Cryptography

In this section, two sets of the public key pair are generated using the same curve parameters. These results will be forwarded to the next section for Diffie-Hellman key exchange.

Example I:

Input:

a_2 = 0000 00000000 00000000 00000000 00000000 00000003

a_6 = 0000 00000000 00000000 00000000 00000000 0000000A

Generator: P

x = 0000 00000000 00000000 00000000 00000000 00000003

y = 0000 00000000 00000000 00000000 00000000 00000004

Private Key A:

K_A = 07EA 1ECCE3FB A449B1B6 185262CB 4EA6A70B 2CFDAC21

Output:

Public Key (K_{pubA}):

x = 0400 45474113 80BAB85A 66F69A87 B4F0A4E2 64D58442

y = 0737 7026E7BD DC66B577 758A5426 2A85D9BA 282235A2

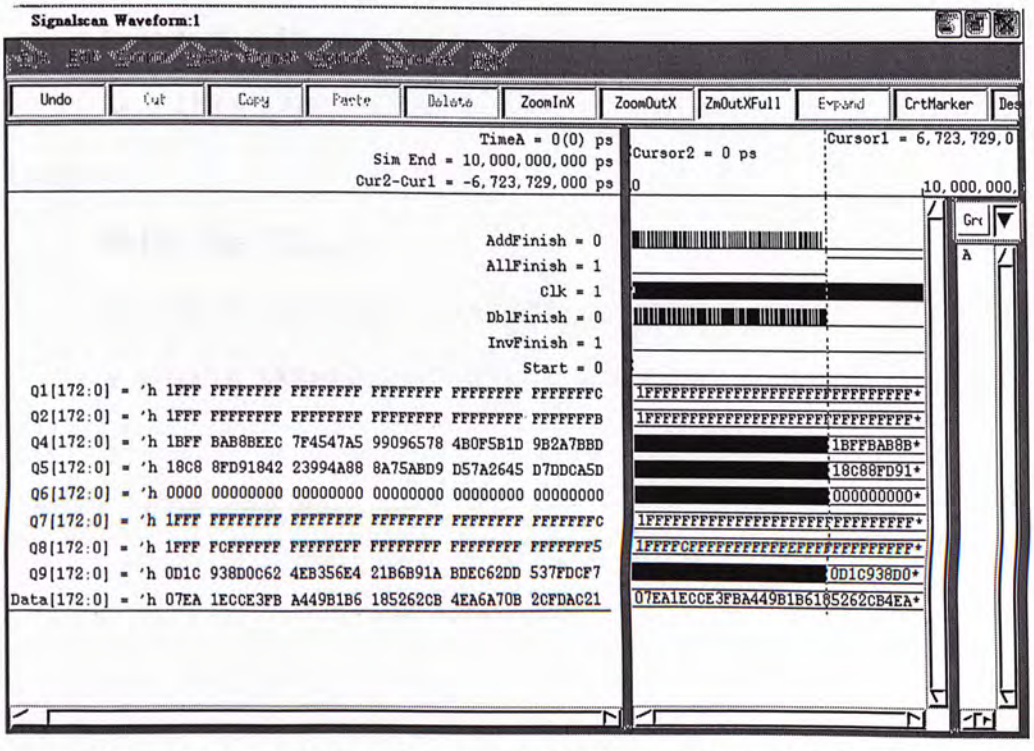


Figure 7.1. The postlayout simulation of the public key pair (Example 1)

In Figure 7.1, the public key pair is shown in the waveform "Q4" and "Q5", where Q4 and Q5 are the complement of x and y-coordinates of the public key respectively. And "Data" is the private key. The names of these waveforms are also applied to the following waveforms. In this example,

the computation time is about 6.72ms.

Example II:

Input:

a_2 = 0000 00000000 00000000 00000000 00000000 00000003

a_6 = 0000 00000000 00000000 00000000 00000000 0000000A

Generator: P

x = 0000 00000000 00000000 00000000 00000000 00000003

y = 0000 00000000 00000000 00000000 00000000 00000004

Private Key B:

K_B = 1FE9 12FE919F 52C74DE4 477E45E4 2367C21A E079D1E2

Output:

Public Key (K_{pubB}):

x = 0204 B2A9CBBF 448A84E0 3D324023 7CE20C0E F3512780

y = 03A9 5AEBEFC2 19E83E06 9149AC7F 278566BD D82BB463

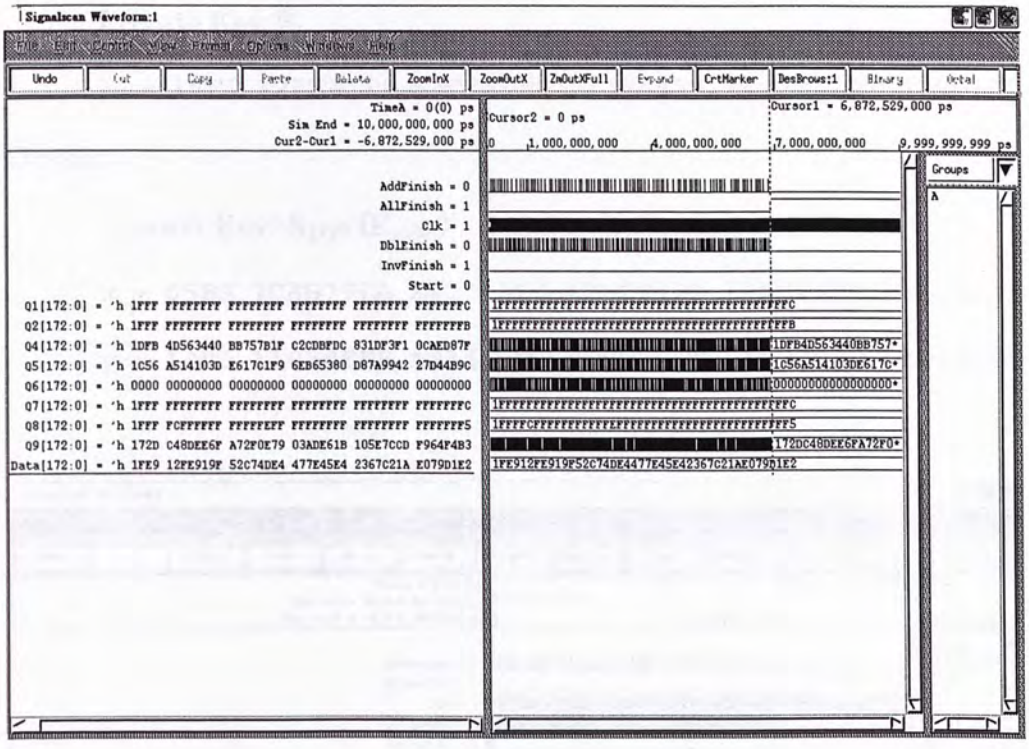


Figure 7.2. The postlayout simulation of the public key pair (Example 2)

The computation time in this example is about 6.87ms.

7.3 Results of the Session-key Cryptography

Example I:

Input:

a_2 = 0000 00000000 00000000 00000000 00000000 00000003

a_6 = 0000 00000000 00000000 00000000 00000000 0000000A

Public Key A:

x = 0400 45474113 80BAB85A 66F69A87 B4F0A4E2 64D58442

y = 0737 7026E7BD DC66B577 758A5426 2A85D9BA 282235A2

Private Key B:

K_B = 1FE9 12FE919F 52C74DE4 477E45E4 2367C21A E079D1E2

Output:

Session Key: K_{priB} (K_{pubA})

x = 05BE 7CBB790A 237588F0 40029129 13FCD581 A41CEA04

y = 1305 119305EB A93459EB 5D8CF3FF D15E7062 0DF18C18

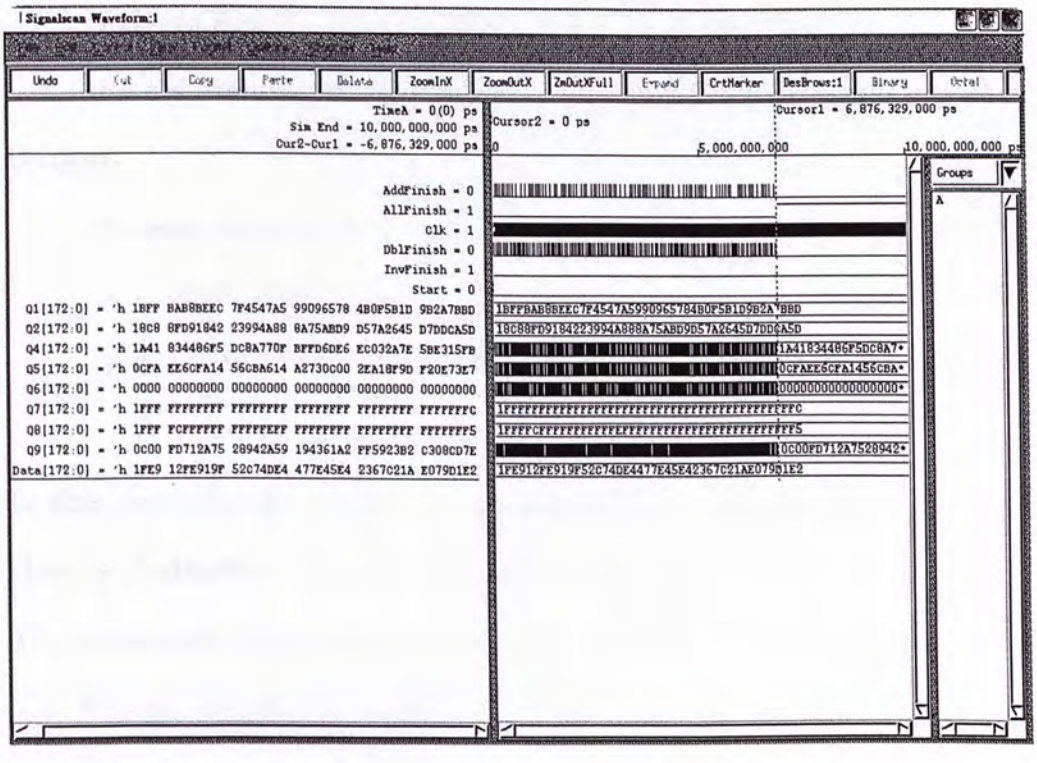


Figure 7.3. The postlayout simulation of the Diffie-Hellman key exchange (using A's public key to generate the session key)

In this example, the session key is generated by using A's public key, and then it is multiplied with the B's private key using curve multiplication. The computation time is about 6.88 ms.

Example II:

Input:

a_2 = 0000 00000000 00000000 00000000 00000000 00000003

a_6 = 0000 00000000 00000000 00000000 00000000 0000000A

Public Key B:

x = 0204 B2A9CBBF 448A84E0 3D324023 7CE20C0E F3512780

y = 03A9 5AEBEFC2 19E83E06 9149AC7F 278566BD D82BB463

Private Key A:

K_A = 07EA 1ECCE3FB A449B1B6 185262CB 4EA6A70B 2CFDAC21

Output:

Session Key: K_{pubB} (K_{priA})

x = 05BE 7CBB790A 237588F0 40029129 13FCD581 A41CEA04

y = 1305 119305EB A93459EB 5D8CF3FF D15E7062 0DF18C18

In this example, the session key is generated by using B's public key, and then is multiplied with the A's private key using curve multiplication. The computation time is about 6.73 ms. Compare Figure 7.3 and Figure 7.4, we can see that Q4 and Q5 are the same. This is the session key generation method in ECC.

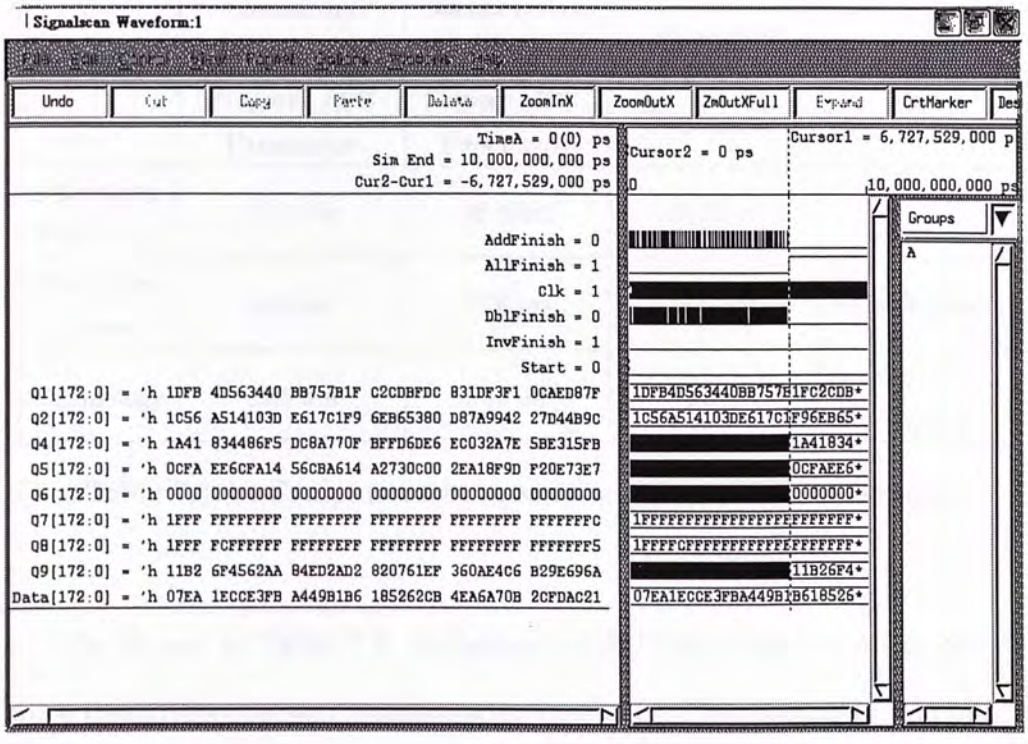


Figure 7.4. The postlayout simulation of the Diffie-Hellman key exchange (using B's public key to generate the session key)

7.4 Comparison with the Existing Crypto-processor

Since ECC can be implemented by different approaches, for example, polynomial basis vs optimal normal basis, affine coordinates system vs projective coordinates system. In 2001, a microcoded ECC processor is proposed [12]. This processor is implemented by FPGA using optimal normal basis representation over projective coordinates. The comparison between the proposed ECC processor and the FPGA one is shown in Table 7.1.

	Simulation Results of the Proposed ECC Processor	Measurement Results of the Proposed ECC Processor	Microcoded ECC Processor	Improvements
Operating Frequency	20 MHz	18 MHz	28 MHz	35.71% reduced
Execution Time	6.8 ms	7.56 ms	11.1 ms	31.89% faster
Technology	0.35 um	0.35 um	0.22 um	59.1% increased

Table 7.1. Comparison between the 173-bit ECC Processors

As shown in Table 7.1, the proposed ECC processor is much better than the ECC processor implemented by FPGA. This is mainly due to the modification of the mathematical expressions, the use of efficient finite field multiplier and the integration of the finite field instructions.

7.5 Power Consumption

In the public key pair generation, the maximum measured power consumption of the crypto-processor is 95mW when it operates at 18 MHz and 3 V. In the power saving mode, that is, the clock frequency is 1 MHz and the power supply is 1.6 V, the crypto-processor consumes at most 20mW in the public key pair generation. The power consumption of the I/O pads is included here. If we do not count the energy consumed by the I/O pads, the power consumption will be smaller. Table 7.2 shows the power consumption of this IC under different operating voltage for a single key pair generation.

Operating Voltage	Power Consumption
3.0 V	95 mW
2.5 V	54 mW
2.0 V	33 mW
1.6 V	20 mW

Table 7.2. The power consumption of ECC crypto-processor under different operating voltage

Chapter 8 Conclusion

A 173-bit type II Optimal Normal Basis Elliptic Curve Crypto-processor is implemented. This processor is a public key cryptosystem that can generate the key pair for public key cryptography and perform Diffie-Hellman key exchange using the elliptic curve.

The aim of this research is to optimize the design for low power consumption so that it can be applied in the contact-less smartcard applications. The advantages of this crypto-processor are summarized as follows:

- Reduce the number of field multiplications in curve addition by simplification of the mathematical expressions.
- Reduce the frequency the flip flops used in the field multiplication by using 3-way parallel multiplier architecture, so that both the power consumption and the propagation delay are minimized.
- Integrate the field addition, squaring and multiplication into one single instruction $Q = A \times B + C$ so that the number of the registers used is reduced.

Furthermore, many other advantages of ECC also make it favorable in the public key cryptography such as high security strength per bit, efficient hardware implementation over finite field systems, etc.

Since the Elliptic Curve Cryptography is not mature enough and there are still spaces for future improvement, we believe that it will become broad applied in the cryptographic field and will be dominated in the market in the near future.

Bibliography

[1] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance", IEEE Transactions on Information Theory 24 (1978), 106-110.

[2] J. Pollard, "Theorems on factorization and primality testing", Proceedings of the Cambridge Philosophical Society, Vol 76 (1974), 521-528.

[3] M. Hellman and J. Reyneri, "Fast computation of discrete logarithm in GF (q)", in CRYPTO '82 (1982) Plenum Publishing, 3-14.

[4] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", Communications of the ACM (1978), 21:120-126.

[5] A. J. Menezes, "Elliptic Curve Public Key Cryptosystems", Kluwer Academic Publishers, 1993.

[6] N. Koblitz, "Elliptic Curve Cryptosystems", Mathematics of Computation (1987), 203-209.

[7] V.S. Miller, "Use of Elliptic Curves in Cryptography", in CRYPTO '85, New York (1987) Springer-Verlag, 417-426.

[8] J. Pollard, "Monte Carlo methods for index computation (mod p)", Mathematics of Computation 32 (1978), 918-924.

[9] R. Gallant, R. Lambert and S. Vanstone, "Improving the parallelized Pollard lambda search on anomalous binary curves", Mathematics of

Computation 69 (2000), 1699-1705.

[10] M. Wiener and R. Zuccherato, "Faster attacks on elliptic curve cryptosystems", Selected Areas in Cryptography – SAC '98, Lecture Notes in Computer Science 1556 (1999), Springer-Verlag, 190-200.

[11] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S.A. Vanstone, "An Implementation for a fast public key cryptosystem", Journal of Cryptography (1991), 3:63-79.

[12] K.H. Leung, K.W. Ma, W.K. Wong, P.H.W. Leong, "FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor", IEEE Symposium on FPCCM (2000), 68-76.

Appendix

173-bit Type II ONB Multiplication Table

The following logic expressions are the pin assignment of the multiplier element used in the 173-bit field multiplier. These expressions are based on the information given in the 173-bit lambda matrix. "•" represents logic AND, "⊕" represents logic XOR.

c000 = a0 • (b1)
c001 = a2 • (b1 ⊕ b153)
c002 = a4 • (b106 ⊕ b154)
c003 = a6 • (b119 ⊕ b134)
c004 = a8 • (b14 ⊕ b87)
c005 = a10 • (b59 ⊕ b83)
c006 = a12 • (b80 ⊕ b2)
c007 = a14 • (b39 ⊕ b135)
c008 = a16 • (b71 ⊕ b101)
c009 = a18 • (b91 ⊕ b153)
c010 = a20 • (b14 ⊕ b142)
c011 = a22 • (b58 ⊕ b89)
c012 = a24 • (b62 ⊕ b82)
c013 = a26 • (b40 ⊕ b54)
c014 = a28 • (b161 ⊕ b1)
c015 = a30 • (b73 ⊕ b76)
c016 = a32 • (b108 ⊕ b172)
c017 = a34 • (b50 ⊕ b93)
c018 = a36 • (b71 ⊕ b127)
c019 = a38 • (b103 ⊕ b137)
c020 = a40 • (b144 ⊕ b156)
c021 = a42 • (b43 ⊕ b44)
c022 = a44 • (b43 ⊕ b147)

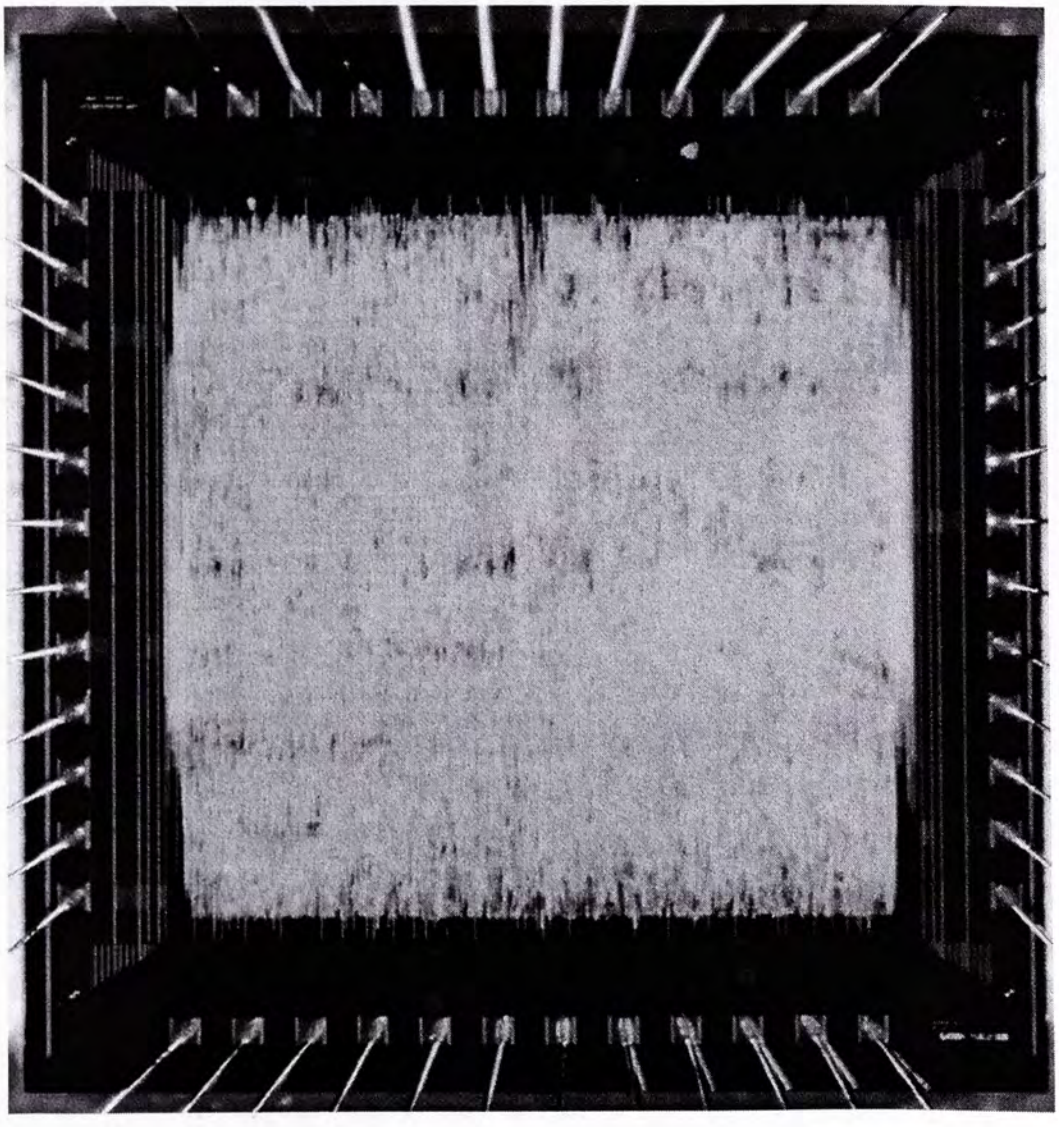
c023 = a46 • (b44 ⊕ b160)
c024 = a48 • (b144 ⊕ b149)
c025 = a50 • (b111 ⊕ b18)
c026 = a52 • (b66 ⊕ b94)
c027 = a54 • (b40 ⊕ b98)
c028 = a56 • (b162 ⊕ b15)
c029 = a58 • (b67 ⊕ b152)
c030 = a60 • (b141 ⊕ b149)
c031 = a62 • (b158 ⊕ b169)
c032 = a64 • (b39 ⊕ b107)
c033 = a66 • (b50 ⊕ b126)
c034 = a68 • (b124 ⊕ b143)
c035 = a70 • (b101 ⊕ b17)
c036 = a72 • (b95 ⊕ b25)
c037 = a74 • (b94 ⊕ b159)
c038 = a76 • (b67 ⊕ b26)
c039 = a78 • (b106 ⊕ b150)
c040 = a80 • (b66 ⊕ b127)
c041 = a82 • (b54 ⊕ b92)
c042 = a84 • (b87 ⊕ b16)
c043 = a86 • (b11 ⊕ b28)
c044 = a88 • (b151 ⊕ b17)
c045 = a90 • (b87 ⊕ b97)
c046 = a92 • (b123 ⊕ b31)
c047 = a94 • (b58 ⊕ b7)
c048 = a96 • (b118 ⊕ b120)
c049 = a98 • (b118 ⊕ b44)
c050 = a100 • (b62 ⊕ b129)
c051 = a102 • (b92 ⊕ b139)
c052 = a104 • (b97 ⊕ b160)
c053 = a106 • (b71 ⊕ b130)
c054 = a108 • (b59 ⊕ b138)
c055 = a110 • (b129 ⊕ b47)
c056 = a112 • (b4 ⊕ b22)
c057 = a114 • (b94 ⊕ b117)
c058 = a116 • (b73 ⊕ b46)
c059 = a118 • (b95 ⊕ b42)
c060 = a120 • (b117 ⊕ b27)

c061 = a122 • (b76 ⊕ b36)
c062 = a124 • (b154 ⊕ b163)
c063 = a126 • (b71 ⊕ b19)
c064 = a128 • (b146 ⊕ b162)
c065 = a130 • (b9 ⊕ b46)
c066 = a132 • (b101 ⊕ b3)
c067 = a134 • (b106 ⊕ b56)
c068 = a136 • (b94 ⊕ b62)
c069 = a138 • (b118 ⊕ b140)
c070 = a140 • (b82 ⊕ b118)
c071 = a142 • (b98 ⊕ b140)
c072 = a144 • (b120 ⊕ b33)
c073 = a146 • (b64 ⊕ b68)
c074 = a148 • (b80 ⊕ b129)
c075 = a150 • (b107 ⊕ b41)
c076 = a152 • (b93 ⊕ b60)
c077 = a154 • (b123 ⊕ b130)
c078 = a156 • (b83 ⊕ b89)
c079 = a158 • (b129 ⊕ b75)
c080 = a160 • (b168 ⊕ b20)
c081 = a162 • (b5 ⊕ b51)
c082 = a164 • (b91 ⊕ b146)
c083 = a166 • (b87 ⊕ b27)
c084 = a168 • (b103 ⊕ b138)
c085 = a170 • (b48 ⊕ b77)
c086 = a172 • (b111 ⊕ b39)
c087 = a1 • (b127 ⊕ b26)
c088 = a3 • (b139 ⊕ b168)
c089 = a5 • (b24 ⊕ b59)
c090 = a7 • (b124 ⊕ b11)
c091 = a9 • (b18 ⊕ b73)
c092 = a11 • (b108 ⊕ b154)
c093 = a13 • (b101 ⊕ b126)
c094 = a15 • (b11 ⊕ b65)
c095 = a17 • (b22 ⊕ b28)
c096 = a19 • (b65 ⊕ b72)
c097 = a21 • (b5 ⊕ b38)
c098 = a23 • (b162 ⊕ b55)

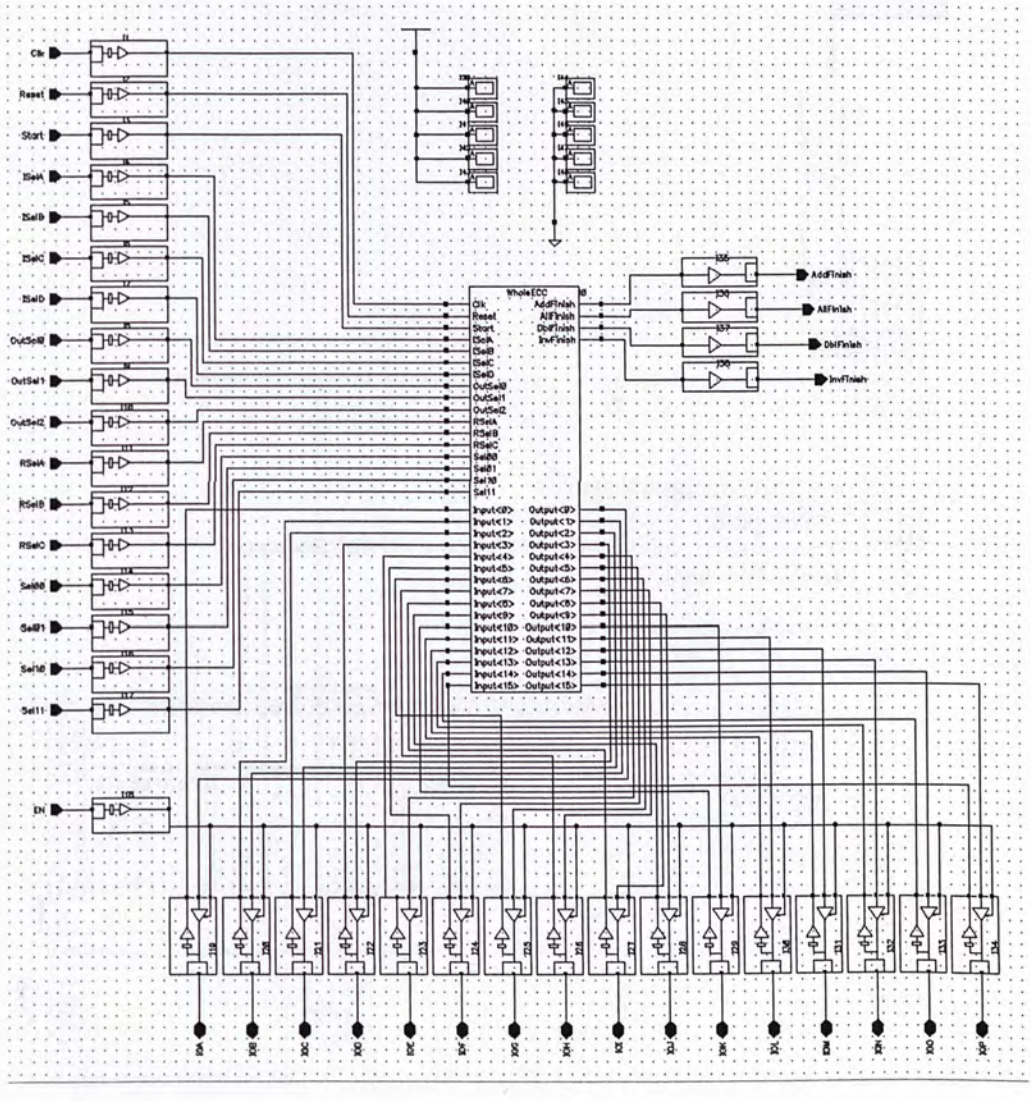
c099 = a25 • (b31 ⊕ b80)
c100 = a27 • (b18 ⊕ b22)
c101 = a29 • (b163 ⊕ b77)
c102 = a31 • (b58 ⊕ b100)
c103 = a33 • (b45 ⊕ b81)
c104 = a35 • (b106 ⊕ b84)
c105 = a37 • (b31 ⊕ b63)
c106 = a39 • (b28 ⊕ b78)
c107 = a41 • (b151 ⊕ b76)
c108 = a43 • (b160 ⊕ b24)
c109 = a45 • (b127 ⊕ b143)
c110 = a47 • (b3 ⊕ b55)
c111 = a49 • (b141 ⊕ b150)
c112 = a51 • (b26 ⊕ b66)
c113 = a53 • (b20 ⊕ b110)
c114 = a55 • (b38 ⊕ b91)
c115 = a57 • (b45 ⊕ b72)
c116 = a59 • (b119 ⊕ b96)
c117 = a61 • (b9 ⊕ b27)
c118 = a63 • (b137 ⊕ b55)
c119 = a65 • (b149 ⊕ b70)
c120 = a67 • (b144 ⊕ b85)
c121 = a69 • (b4 ⊕ b114)
c122 = a71 • (b159 ⊕ b112)
c123 = a73 • (b152 ⊕ b85)
c124 = a75 • (b144 ⊕ b70)
c125 = a77 • (b147 ⊕ b149)
c126 = a79 • (b39 ⊕ b90)
c127 = a81 • (b158 ⊕ b66)
c128 = a83 • (b135 ⊕ b125)
c129 = a85 • (b19 ⊕ b58)
c130 = a87 • (b55 ⊕ b72)
c131 = a89 • (b134 ⊕ b63)
c132 = a91 • (b142 ⊕ b104)
c133 = a93 • (b7 ⊕ b119)
c134 = a95 • (b162 ⊕ b33)
c135 = a97 • (b85 ⊕ b126)
c136 = a99 • (b156 ⊕ b48)

c137 = a101 • (b160 ⊕ b90)
c138 = a103 • (b169 ⊕ b85)
c139 = a105 • (b22 ⊕ b41)
c140 = a107 • (b27 ⊕ b124)
c141 = a109 • (b11 ⊕ b116)
c142 = a111 • (b65 ⊕ b76)
c143 = a113 • (b51 ⊕ b59)
c144 = a115 • (b153 ⊕ b65)
c145 = a117 • (b78 ⊕ b104)
c146 = a119 • (b17 ⊕ b132)
c147 = a121 • (b161 ⊕ b16)
c148 = a123 • (b36 ⊕ b116)
c149 = a125 • (b72 ⊕ b77)
c150 = a127 • (b91 ⊕ b148)
c151 = a129 • (b81 ⊕ b150)
c152 = a131 • (b153 ⊕ b154)
c153 = a133 • (b84 ⊕ b96)
c154 = a135 • (b46 ⊕ b80)
c155 = a137 • (b17 ⊕ b73)
c156 = a139 • (b172 ⊕ b42)
c157 = a141 • (b60 ⊕ b124)
c158 = a143 • (b28 ⊕ b31)
c159 = a145 • (b119 ⊕ b132)
c160 = a147 • (b1 ⊕ b15)
c161 = a149 • (b26 ⊕ b46)
c162 = a151 • (b25 ⊕ b56)
c163 = a153 • (b112 ⊕ b157)
c164 = a155 • (b64 ⊕ b126)
c165 = a157 • (b47 ⊕ b77)
c166 = a159 • (b18 ⊕ b114)
c167 = a161 • (b62 ⊕ b157)
c168 = a163 • (b44 ⊕ b68)
c169 = a165 • (b2 ⊕ b75)
c170 = a167 • (b110 ⊕ b125)
c171 = a169 • (b100 ⊕ b148)
c172 = a171 • (b150 ⊕ b171)

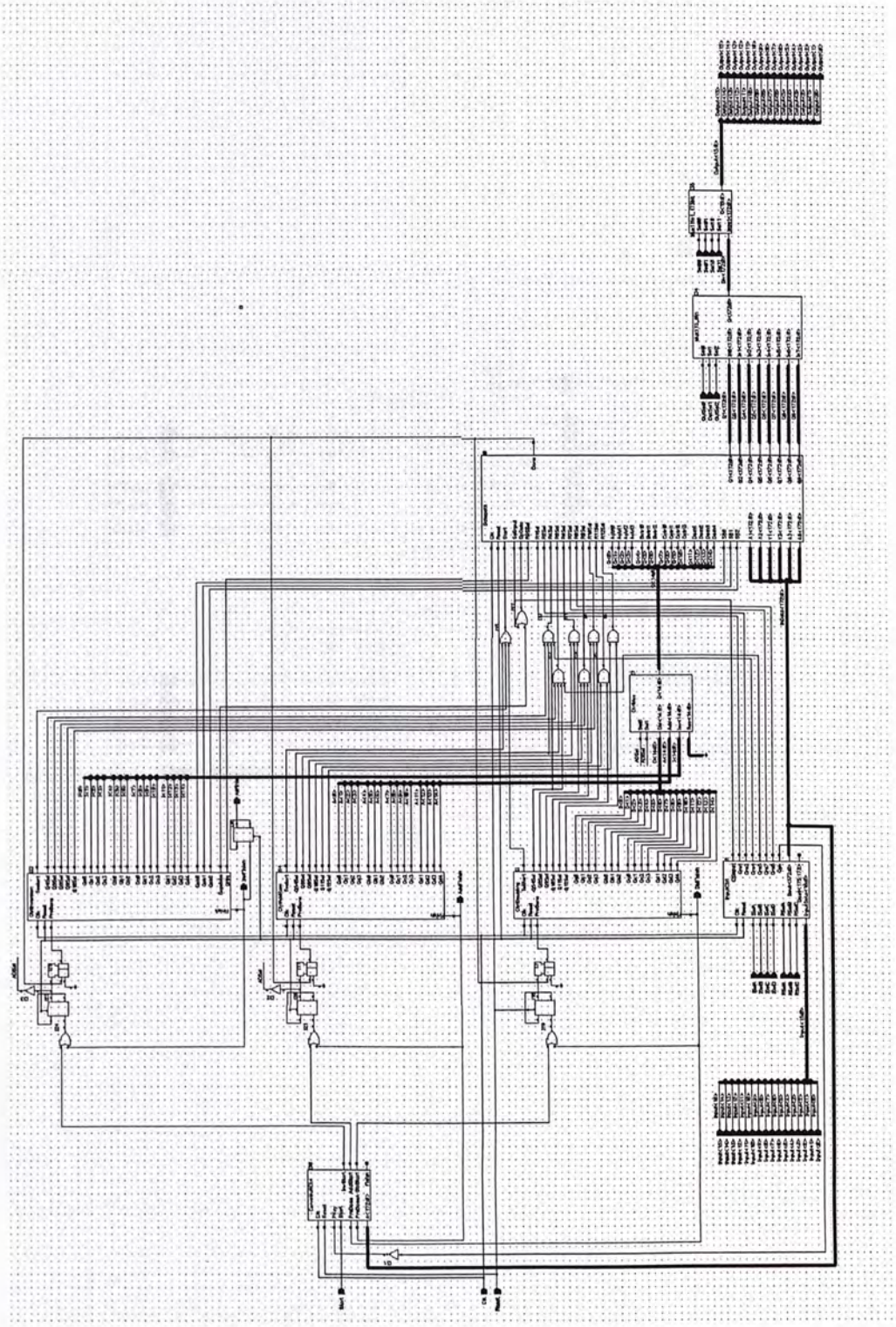
Layout View of the Elliptic Curve Crypto-processor



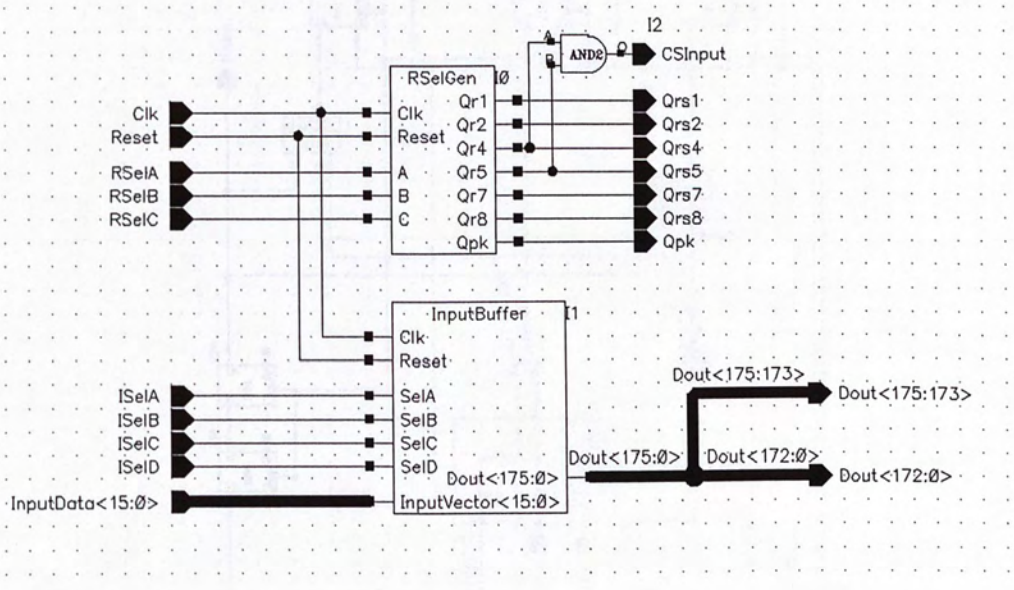
Schematics of the Elliptic Curve Crypto-processor



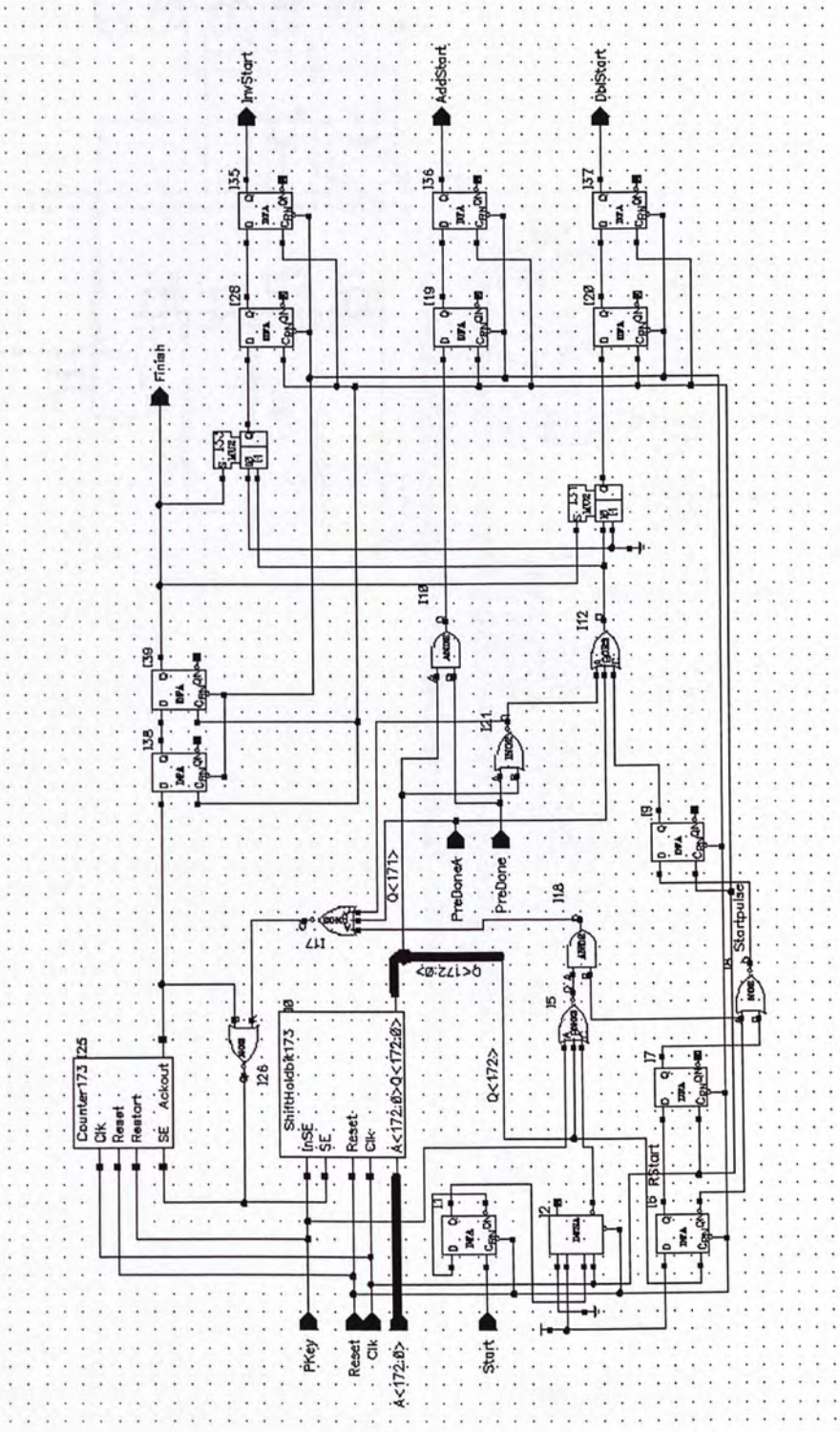
Schematics of the System Level Design



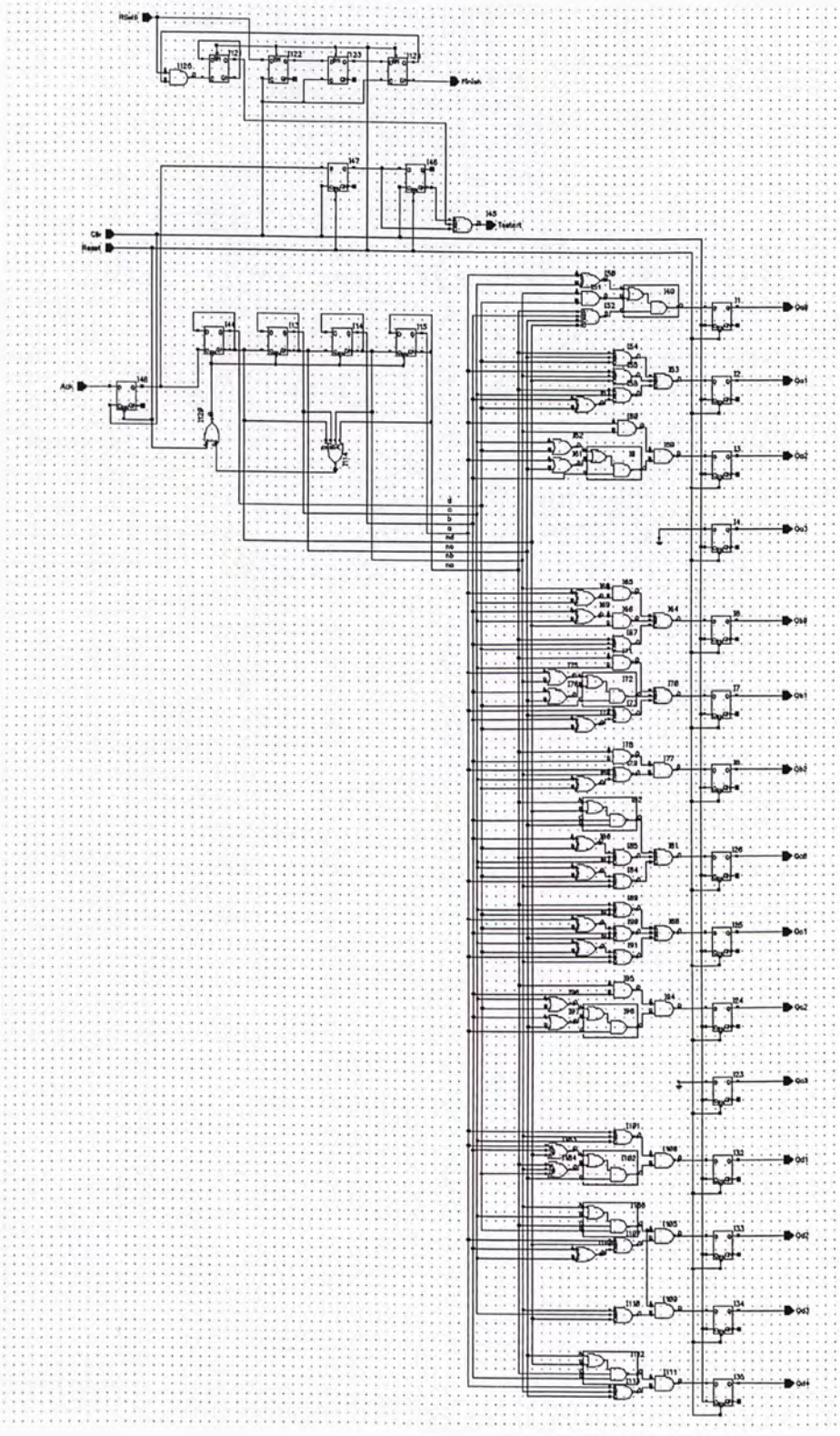
Schematics of the I/O Control Interface



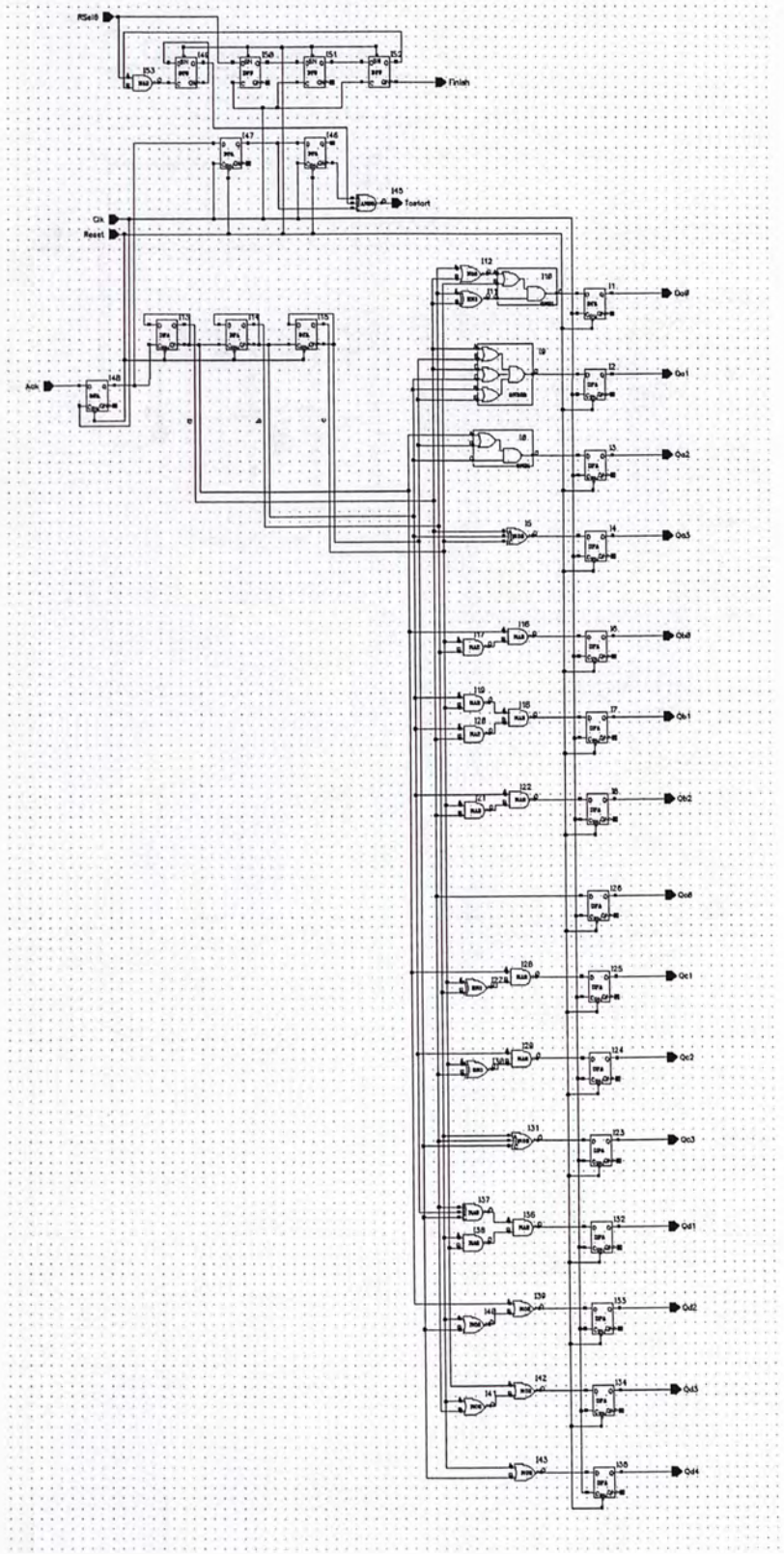
Schematics of the Curve Multiplication Module



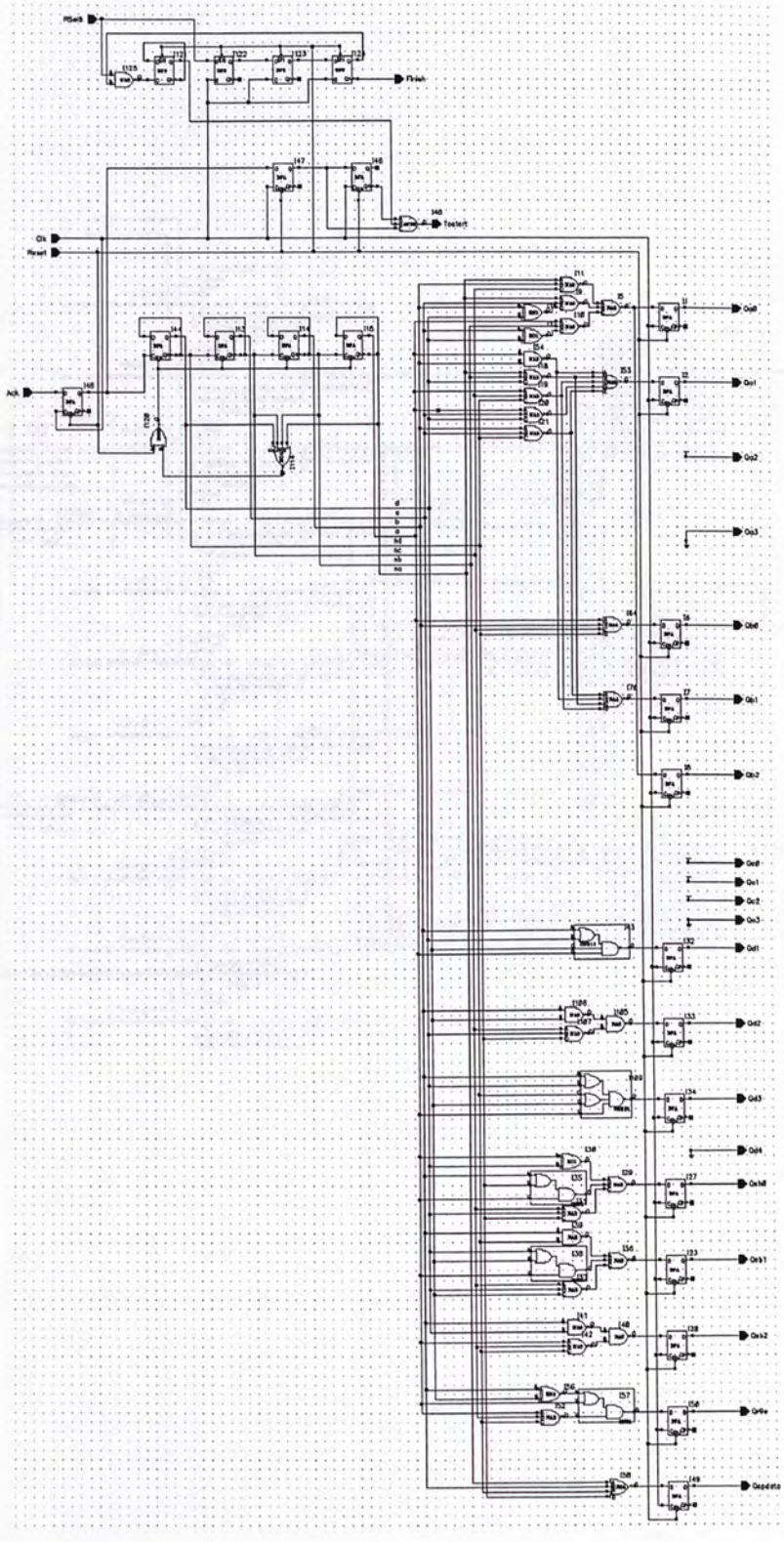
Schematics of the Curve Addition Module



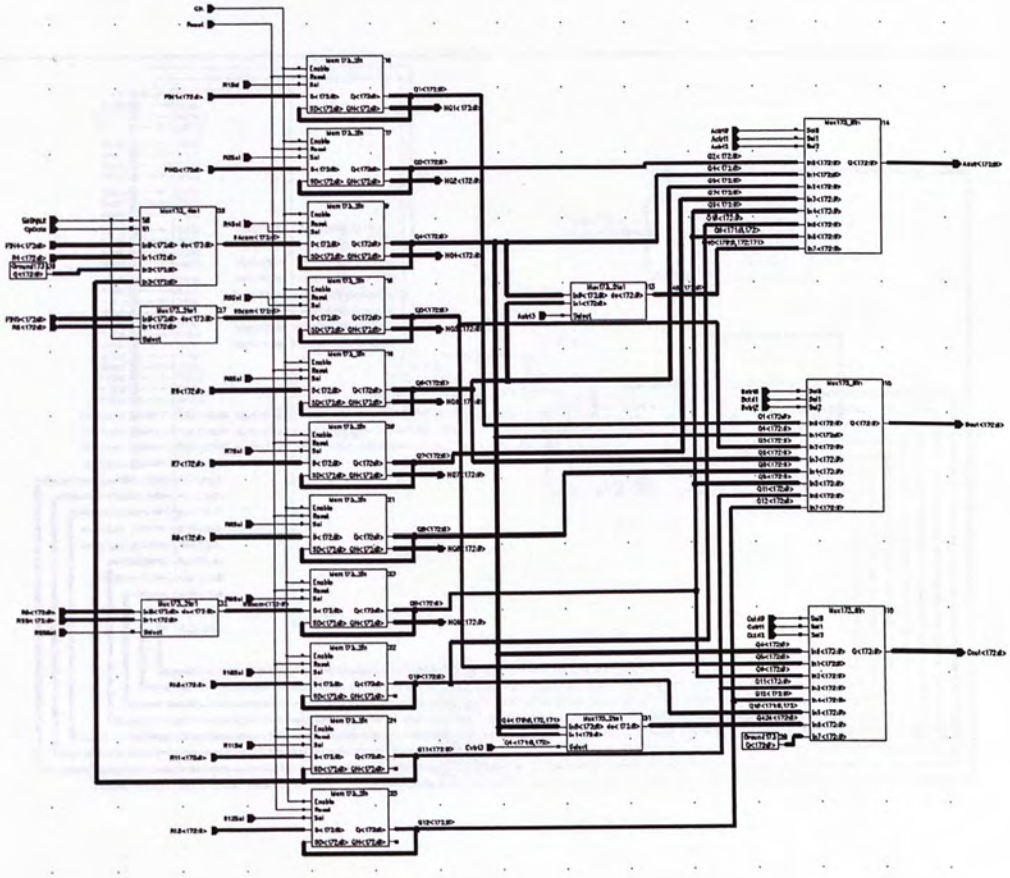
Schematics of the Curve Doubling Module



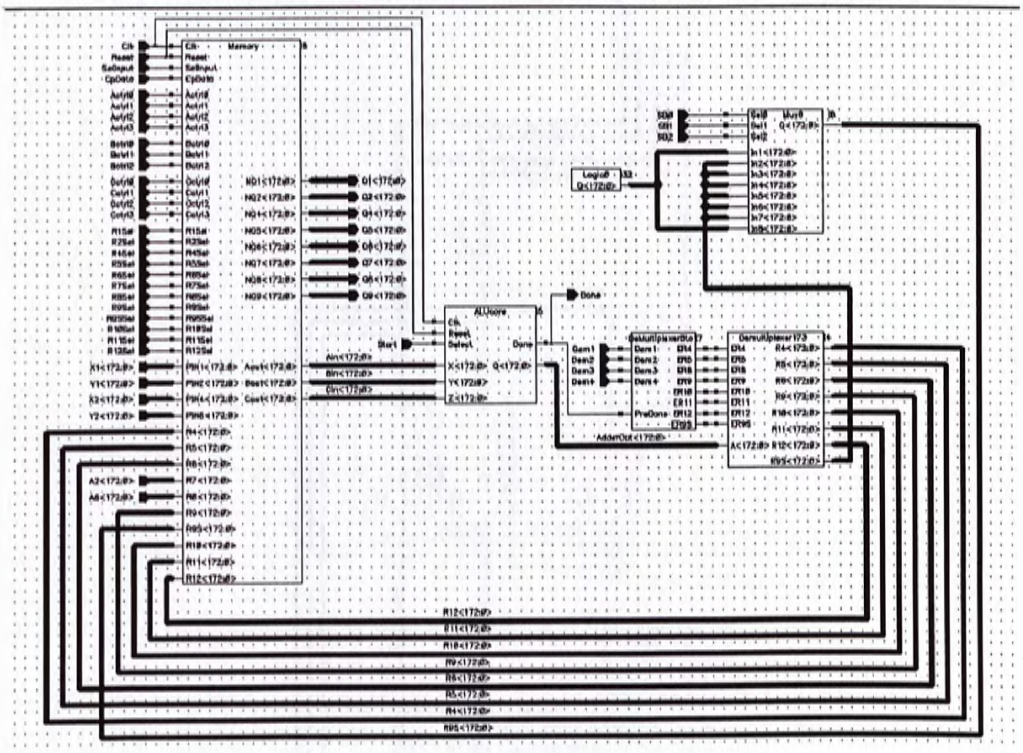
Schematics of the Field Inversion Module



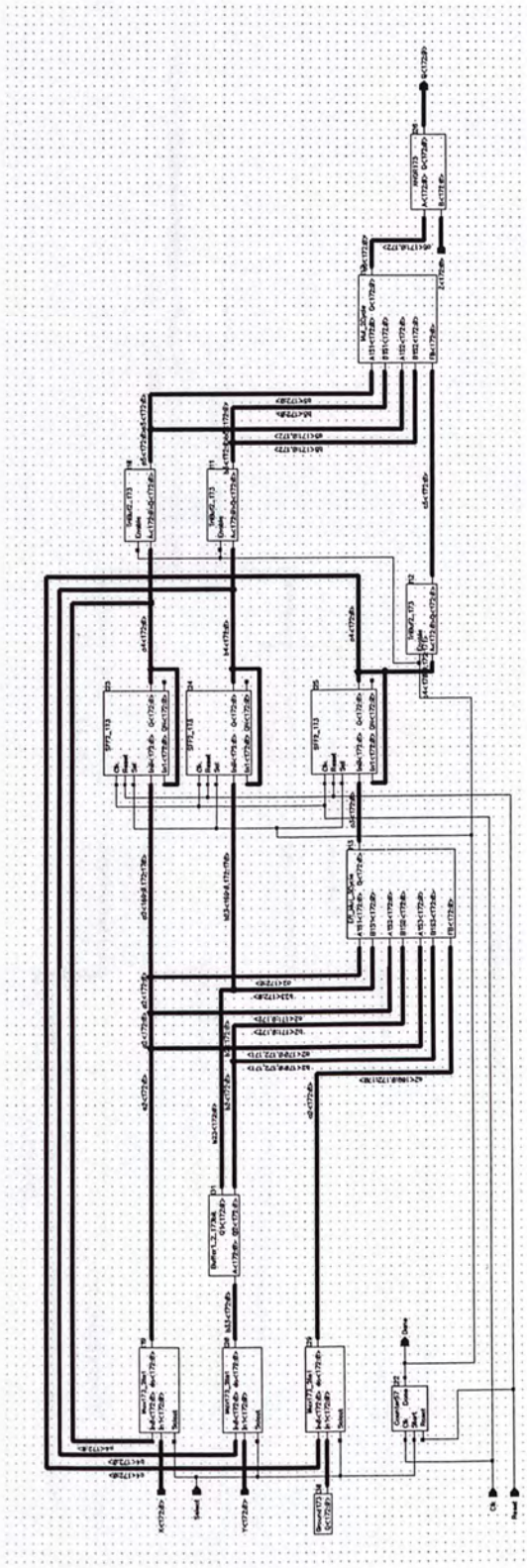
Schematics of the Register Unit



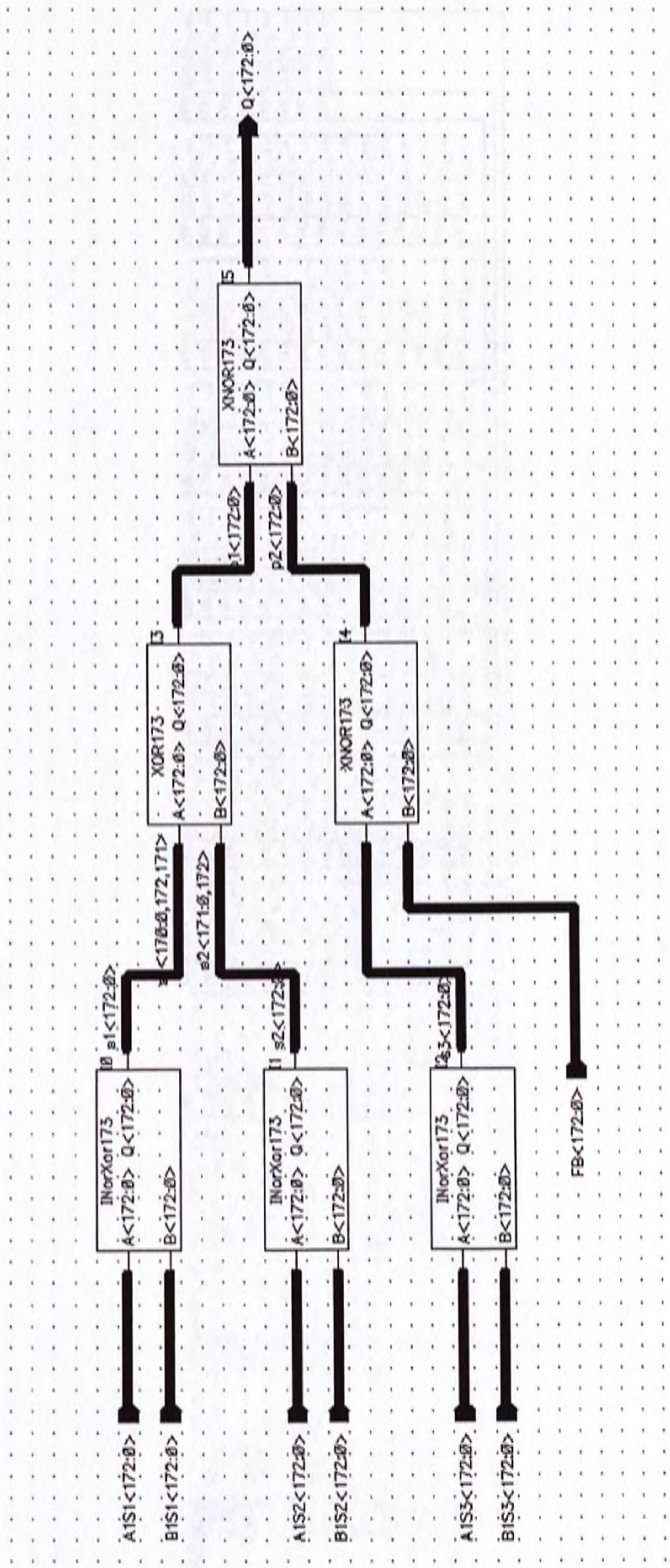
Schematics of the Datapath



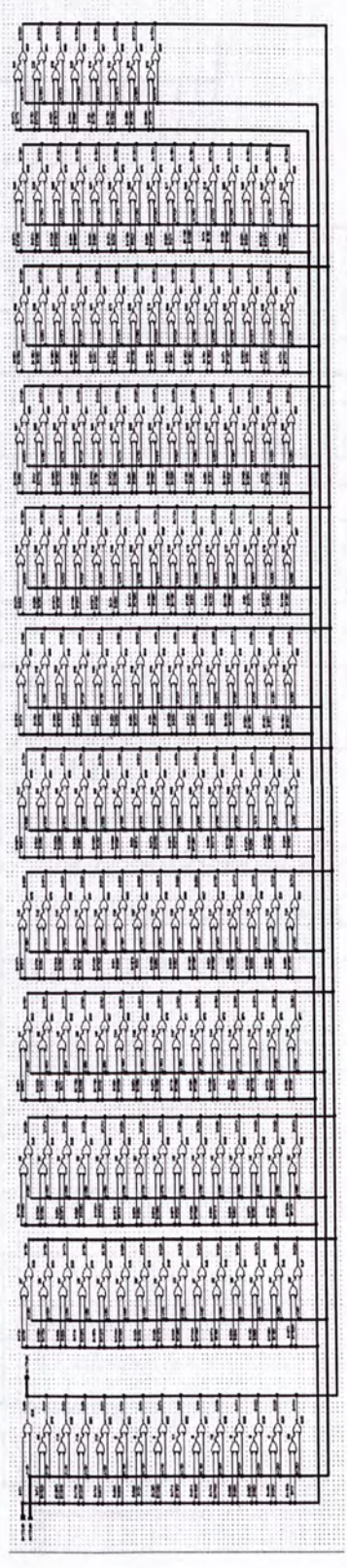
Schematics of the Finite Field ALU



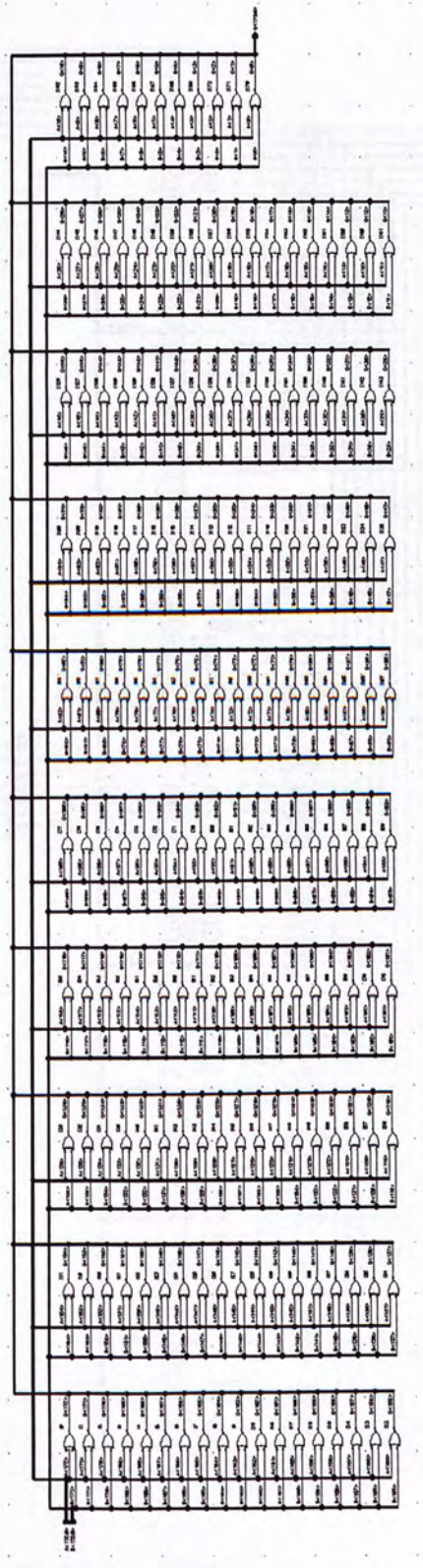
Schematics of the 3-way Parallel Multiplier



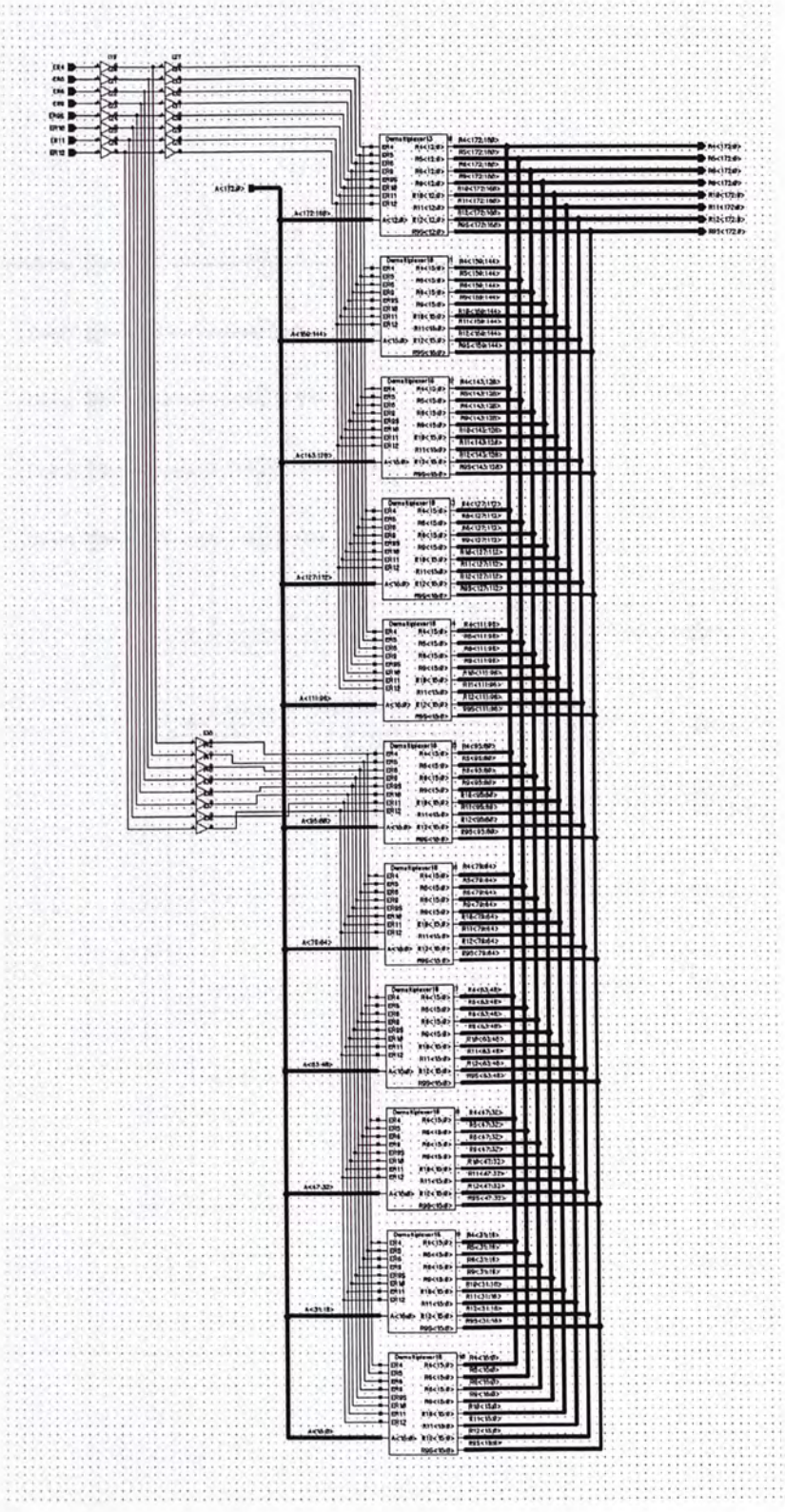
Schematics of the Multiplier Elements



Schematics of the Field Adder



Schematics of Demultiplexer



Schematics of the Control of the Demultiplexer