AN EFFICIENT HARDWARE IMPLEMENTATION OF THE TATE PAIRING IN CHARACTERISTIC THREE

by
Giray Kömürcü
B.S., Microelectronics Engineering, Sabanci University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2008

AN EFFICIENT HARDWARE
IMPLEMENTATION OF THE TATE PAIRING
IN CHARACTERISTIC THREE

APPROVED BY:

Prof. Günhan Dündar ....
(Thesis Supervisor)

Assoc.Prof. Erkay Savaş ....
(Thesis Co-advisor)

Asst.Prof. İlker Hamzaoğlu ....

Asst.Prof. Şenol Mutlu ....

DATE OF APPROVAL : ....

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor Prof. Günhan Dündar for his valuable support and helps before and during my master thesis work. I also want to thank for the VLSI classes those made my thesis a lot easier during my MS studies.

I am also grateful to my Co Advisor Assoc. Prof. Erkay Savaş for his encouragement, support and helps on my master thesis. Everything would be harder without his guiding and experience on the specific subject.

Thanks to Asst. Prof. İlker Hamzaoğlu for being a member of my thesis committee and his outstanding VLSI classes during my undergraduate studies.

Thanks to Asst. Prof. Şenol Mutlu for being a member of my thesis committee.

I would also very much thank to the TUBITAK-UEKAE for its opportunities and contribution on my experience on Cryptography and VLSI design that let me complete my MS study.

I also want to thank Dr. Aziz Ulvi Çalışkan and Dr. Yaman Özelçi for their helps on my work and very valuable supports.

Also I want to thank my friends at UEKAE that have contributed to my thesis by their friendship and important support.

I also very much appreciate the motivation, help and encouragement of Elif Sezgin during my thesis work.

Lastly I want express my gratitude to my family, who has patiently supported me during my whole education life.

ABSTRACT

AN EFFICIENT HARDWARE IMPLEMENTATION OF THE TATE PAIRING IN CHARACTERISTIC THREE

Discrete Logarithm systems with bilinear structure recently became an important base for succesful cryptographic protocols such as identity-based encryption, short signatures and multiparty key exchange. Since the main computational task is the evaluation of the bilinear pairings over elliptic curves, which is known to be prohibitively expensive, efficient hardware or software implementations are required to render them applicable in real life scenarios. In this thesis, an efficient accelerator for computing the Tate Pairing in characteristic 3, based on the Modified Duursma Lee algorithm is presented. Accelerator implemented shows that it is possible to improve the area-time product by roughly 12 times on Field Programmable Gate Array (FPGA), compared to estimated values from one of the best known hardware architecture implemented on a same type of FPGA. Also the computation time is improved up to 16 times compared to software applications reported. In addition, the result of an ASIC implementation of the algorithm is presented, which is the first hitherto. Both implementation results show that pairing based cryptosystems can be used even on constrainted devices such as smartcards.

ÖZET

KARAKTERISTIK ÜÇ ÜZERINDE "TATE PAIRING" PROTOKOLUNU UYGULAYAN GÜÇLÜ BIR DONANIM TASARIMI

Çift yönlü lineer yapıdaki ayrık logaritma sistemleri güvenilir kriptografik protokoller için son dönemlerde önemli bir taban oluşturmaktadır. Kimlik Tabanlı Şifreleme (IBE), kısa anahtarlı yapılar, çoklu ortam anahtar değişimi gibi sistemler bu protokollere örnek gösterilebilir. Çift yönlü lineer ikililerin eliptik eğriler üzerinde hesaplanması çok fazla hesaplama gücü gerektirdiğinden, güçlü donanımsal ve yazılımsal uygulamaların geliştirilmesi uygulamaların gerçek hayatta kullanılabilmesi açısından kritik öneme sahiptir. Bu tezde "Modified Duursma-Lee" algoritması temel alınarak, karakteristik 3 üzerinde "Tate Pairing" protokolünü hesaplayan güçlü bir hızlandırıcı yapı sunulmaktadır. Bu hızlandırıcı yapıyla, Sahada Programlanabilir Kapı Dizisi (FPGA) üzerindeki bilinen en hızlı donanımsal yapılara göre zaman-alan çarpımına göre 12 kata kadar bir iyileşme sağlanabileceği görülmüştür. Bu iyileşmenin yazılımsal yapılarla karşılaştırmalarda 16 kat düzeyinde olduğu gözlemlenmiştir. Buna ek olarak algoritmanın ilk Uygulamaya Özel Entegre Devresine (ASIC) dair sonuçları da sunulmaktadır. Iki uygulamanın sonuçları da ikili tabanlı kriptosistemlerin akıllı kart gibi sınırlı araçlarda bile kullanılabileceğini göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
DES	Data Encryption Standard
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
FPGA	Field Programmable Gate Array
IBE	Identity Based Encryption
LSE	Least Significant Element First
MSE	Most Significant Element First
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PKG	Private Key Generator
SHA	Secure Hash Algorithm

1. INTRODUCTION

Cryptography is basically the technique of hiding information. It is generally referred as encryption and decryption. Encryption is the process of converting open information to a non-understandable format. Decryption is reversing the operation of encryption to make the hidden information available again. As shown in Figure 1.1 the term plaintext is used for open information and ciphertext is used for encrypted messages. Cipher generally refers to the algorithm used to apply encryption and decryption.

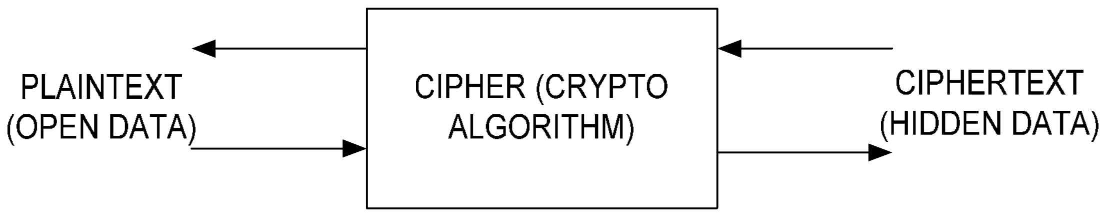

Figure 1.1. Plaintext, Cipher and Ciphertext

The origin of the word Cryptography comes from two Greek words, Kryptos (hidden) and Grafo (write). As understood from the origin, cryptography has a history of thousands of years that has started around 4000 years ago in Egypt with non-standard hieroglyphs which are carved into monuments. The first cryptos are thought to be aiming mystery or having a religious meaning rather than secret communication. After that, it is known that the Greeks of Classical times and Roman Empire used crypto for military purposes. With the development of cryptography, cryptanalysis efforts started to decrypt the messages encrypted, without knowing the actual keys.

In the 20th century World War II led to significant developments in cryptography. During this war, mechanical and electro mechanical crypto machines started to be used widely, and more and more people started to develop secure crypto schemes as well as breaking the codes of the enemy. It is assumed that the modern cryptography has started with Claude Shannon, who started mathematical cryptography in 1949 with his published work, Communication Theory of Secrecy [1]. In this work, cryptographic applications at that time have been divided into three sections; concealing a message into an innocent text, privacy systems those aiming to hide the existence of information, and true secrecy systems that encrypt messages using mathematical functions. Within a short time, the last section, true secrecy systems have become the core of cryptography.

In 1970's, the first encryption standard, Data Encryption Standard (DES), was developed by IBM as a response to the call of the National Institute of Standards and Technology (NIST) (called the National Bureau of Standards at the time). From that time on, many significant algorithms have been used for encryption purposes such as Advanced Encryption Standard (AES), RSA and Elliptic Curve Cryptography (ECC) that will be discussed very briefly in the next few pages [2].

In today's world, cryptography is used in many areas even by ordinary people. Primarily, it has been a must in military communication and data storage for a long time. It is also used in smart cards, financial services, wireless networks etc. Confidentiality, data integrity, authentication, authorization and non-repudiation are the main services that cryptography performs. Confidentiality is the secrecy of information to unauthorized people maintained by encryption. It should be implemented in such a way that unauthorized people can not recover the hidden data by any means. Data integrity is used to detect any changes in data done in an unauthorized manner. This may include insertion, deletion or substitution of data. Authentication is used to verify the origin of information. It is generally provided by digital signatures or message authentication codes. Authorization is providing an official permission to perform a security function or activity. It is generally done after authentication. If authentication is granted, the user is provided the required key or password for the secure application such as access to a database or a room. Non-repudiation provides assurance of the integrity and origin of data to be verified by a third party. By this way an entity can not deny an involvement to an action.

Approved cryptographic algorithms can be divided in to three main classes, hash functions, private key algorithms and public key algorithms. Hash functions can be seen as the simplest ones within these three. They are used to produce a digest of a large data that are hard to produce from a different data. These functions are mainly used for providing data authentication and integrity services, generating and verifying signatures from messages, generating deterministic random numbers and deriving keys. Secure Hash Algorithm (SHA) SHA1 is a well known example of hash functions [3, 4, 5]. SHA1 basically produces 160 bit output, called message digest, when a message of length less then 2^{64} bits is applied as input. The message digest can be used as an input to the Digital Signature Algorithm (DSA) which generates or verifies the signature for the message. This process is shown in Figure 1.2.

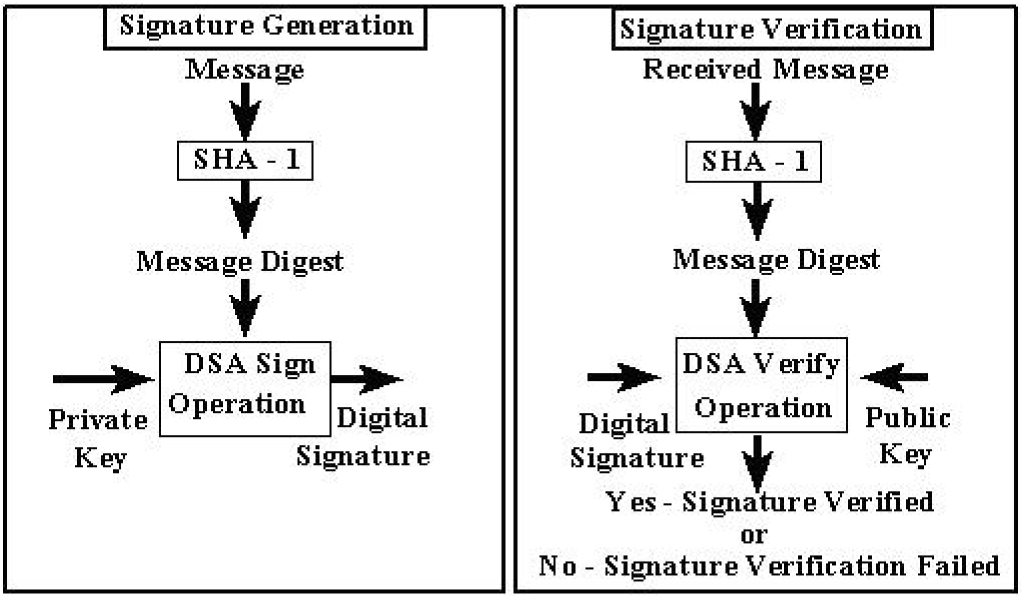

Figure 1.2. Signature generation and verification

Private key algorithms (known as symmetric or secret key algorithms also), encrypt data by using a secret key and it is hard to recover the plaintext without the knowledge of this key. Decryption is also done with the same key in this set of algorithms, hence they are called symmetric key algorithms. Structures of private key algorithms are shown in Figure 1.3. These algorithms lose their security if the keys are disclosed to unauthorized entities. The main functions of symmetric key algorithms are as follows: they are used to maintain data confidentiality, provide authentication and integrity, key establishment and generating deterministic random numbers. DES and AES are two examples of this type of algorithms that will be mentioned below [3, 6].

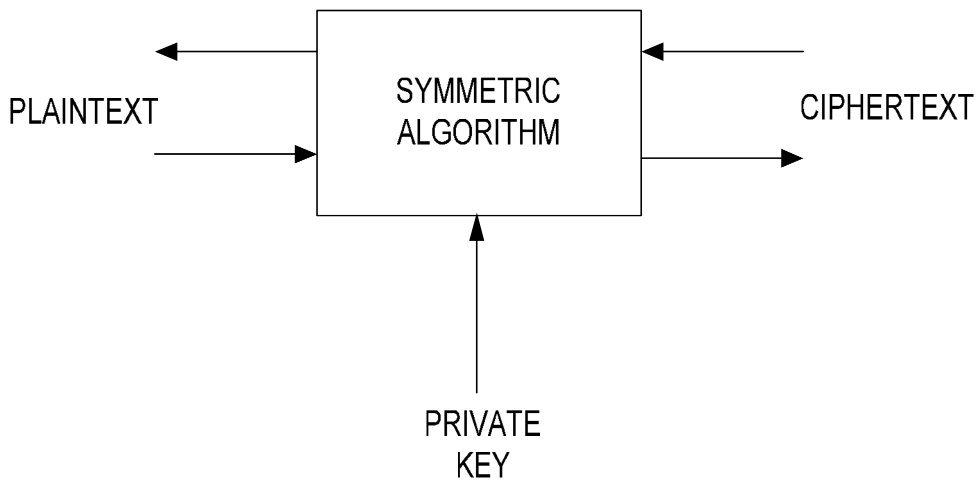

Figure 1.3. Symmetric Key Cryptography

Public key algorithms (known as asymmetric key algorithms also), use two different but mathematically related keys; one is public and one is private. Public key is published to everyone by an authority, but the private key should be known by just the owner of the key pair. In addition, the public key should not reveal information about the private key by any means. Public key algorithms are mainly used for computing digital signatures, generating random numbers and establishing cryptographic keying material. RSA and ECC are the well known examples of this class [3, 6]. The structures for this set of algorithms are shown in Figure 1.4.

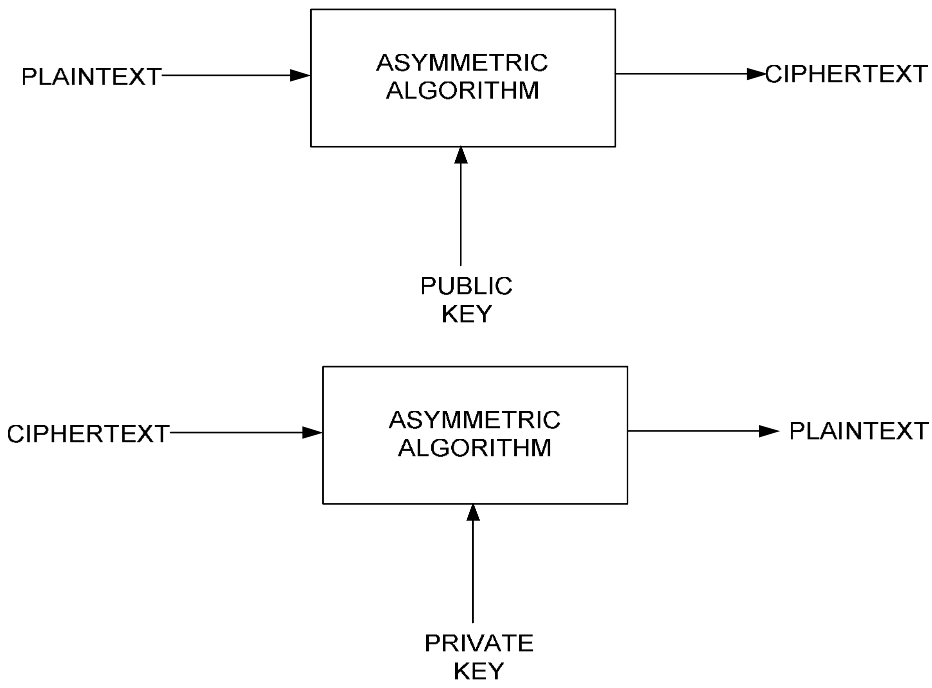

Figure 1.4. Asymmetric Key Cryptography

Public key and private key systems have advantages and disadvantages over each other that will be discussed before moving onto explain DES, AES and RSA algorithms in a more detailed way. Private key algorithms are faster than public key algorithms since they use shorter key lengths and utilize simpler operations, hence demand less computational power. They can generally be implemented in software in an efficient way which is generally much harder for public key systems. But the main problem of private key algorithms arises from key distribution. For example when an entity sends an encrypted message over the internet the same entity have to somehow send the secret key to the entity that will read the message. This is a real problem since the sender needs to use different keys for different set of recipients and these keys should be sent in a secure way each time. However, in public key algorithms, key owners publish public keys over a book or internet and save the private keys for just themselves. When sender wants to send a message it encrypts with the recipients publicly available key and sends the ciphertext, the recipient decrypts the message with its private key. This totally solves the key distribution problem [7,8].

Before moving onto the Tate Pairing and elliptic curve cryptography, the evolution of three modern crypto algorithms, DES, AES and RSA will be discussed briefly to have a better understanding on the subject. With the increasing volume, value and confidentiality of data used by governments, industry and other organizations in 1960's and early 1970's, the need for a strong cryptographic algorithm became significant. On May 15, 1973, NIST asked for public proposals for a very secure and cheap algorithm that would be available to the general public and could be used in a wide variety of applications to protect non-classified data. IBM submitted its algorithm called Lucifer in 1974, which was decided to be the best of the candidates after the evaluation was made with the help of National Security Agency (NSA). In 1976, it was adopted by NIST as a federal standard under the name of Data Encryption Standard (DES) and became the most widely used encryption algorithm in a very short time with the help of government. Unfortunately, with the increase of computational power the security of DES reduced significantly and super computers started to crack the codes within days and hours in 1990's. With the severe reduction in security of the algorithm, NIST planned the replacement of DES in 1997 with the Advanced Encryption Standard (AES). Today, DES is generally used with a modification and it is called Triple DES. Basically, the data is encrypted three times with two different keys by using standard DES in this version and it is still common in many areas such as financial services, smart cards and VPN services [7].

Technically speaking, DES maintains security by permutations and substitutions. In this algorithm, data is encrypted and decrypted in 64-bit blocks, using a 64-bit key. But the effective key length is 56 bits since the least significant bit in each byte is a parity bit and used to maintain odd number of 1's in that byte. The algorithm ignores these parity bits and most significant 7 bits are used resulting in an effective key length of 56 bits. The algorithm completes encryption and decryption in 16 rounds, basically repeating the same round function 16 times to produce the results. Here, each round increases the security level of the algorithm exponentially. As mentioned earlier, DES is a private key algorithm and encryption and decryption is done with the same key just reversing the order of sub keys used in the rounds [9].

Towards the middle of 1990's, the security of DES started to be questioned and it was recognized that a 56-bit key was not large enough for high security applications with the exponentially increasing computational power. As a result, NIST coordinated an open, public competition in 1997 to find its own replacement for DES. This time, the aim was to find a very secure and fast algorithm which will have a long life. The call for submission ended in 2000 and the winning algorithm Rijndael was announced as the Advanced Encryption Standard in November 2001 among five different algorithms [10]. In June 2003, the US Government announced that AES may be used for classifiying information, as well.

As it was the case with DES, AES is also a block cipher algorithm and it has an adjustable block size of. In addition to this, it can be used with three different key sizes, 128 bits, 192 bits and 256 bits. With 128 bit key size, the AES algorithm has on the order of 10^{21} times more AES 128-bit keys than DES 56-bit keys, making nearly impossible to crack it in the foreseeable future with brute force attacks. For the time being only, Side Channel Attacks, which attack the implementation rather than the algorithm, became successful against AES. These types of attacks are mainly based on the power consumption curves of the system. With a non-secure implementation, power consumption of the algorithms has a significant dependency on the private key. Different from the brute force attacks, this dependency allows cracking AES and DES with a low number power consumption curves. This weakness of the implementations can be overcome by inserting random numbers to the datapath, without changing the actual result. These secure implementations result in a cost of area and reduced clock frequency.

Another advantage of AES over DES is its suitability for software applications and small amount of memory requirement making it cheap for hardware implementations. With these features, today AES is used worldwide in many applications and by millions of people.

RSA is one of the first public key crypto algorithms developed and widely used in today's world. It was invented and named in 1977 by Ron Rivest, Adi Shamir and Len Adleman at MIT [11]. The letters RSA are the initials of their surnames. As explained above, public key crypto systems involve two keys, one is public and the other is private. In RSA systems, private key is as long as 512, 1024 or 2048 bits depending on the required security level. But public key is generally a short number such as 2^3-1, 2^5-1, 2^7-1, 2^{16}-1. The security of RSA is mainly based on two problems. First problem is factorization of large numbers, composed of two large primes and the second problem is taking *e*th roots modulo of a composite *m*. The main approach to crack RSA is to factor modulus n. By factoring the modulus n, the secret key can be easily recovered. Up to 2005, the known largest modulus that has been factored was 663 bits long with distributed computing. So, for high security applications it is required to use 1024 bit or higher modulus.

The main disadvantage of RSA over DES and AES is its speed. Since the key lengths are significantly larger than main private key algorithms, it needs much more computational power and it is nearly impossible to compute by software in a feasible time. Also in hardware it takes much more time to encrypt a message with RSA algorithm. In addition to this, with modulus lengths of 4096 or over, hardware became prohibitively large for many applications such as smart cards. This disadvantage of widely used public key crypto algorithm RSA is being overcome by a relatively new approach called Elliptic Curve Cryptography (ECC).

In this thesis, the implementation of the Tate Pairing based on the modified Duursma-Lee algorithm which is built on Elliptic Curve Cryptography is presented. ECC is based on the algebraic structure of elliptic curves over finite fields. In 1985 Neal Koblitz and Victor S. Miller first proposed the use of elliptic curves in cryptography independent from each other [12, 13]. Basically, ECC is based on operations on the points that compose elliptic curves. The hardness of the problem comes from the irreversibility of the operations without the knowledge of a parameter, the key, depending on the algorithm. Dominant operation in ECC is point multiplication which is a relatively easy and fast computation. However, the inverse (the elliptic curve discrete logarithm problem), being a very difficult problem makes the elliptic curves possible to use in cryptography. The main advantage of ECC over RSA is its greater security for the same key length. As seen from Table 1.1 below, there is a major difference between the relative key sizes of RSA and ECC, which affects the cost of the cryptosystem very closely. The main reason for this is its inverse operation gets harder than inverse operation of RSA with increasing key length [14].

Symmetric Key Size (bits)	**RSA and Diffie-Hellman Key Size (bits)**	**Elliptic Curve Key Size (bits)**
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1.1. NIST Recommended Key Sizes [14]

The smaller key sizes make more compact, power efficient and fast implementations possible for a given level of security, hence resulting in cheaper implementations of cryptography in the required applications. In addition to this, due to smaller area, it becomes easier to implement crypto protocols on constrained devices such as smart cards.

Recently pairings have become a new branch of public key cryptography. Simply, they operate on a pair of points defined in a group that contains all the points of an elliptic curve and the point at infinity. The main advantage of the pairings is that, they are the only cryptosystem for now that allow Identity Based Encryption (IBE) [15, 16, 17, 18]. With identity based encryption, any string can be used as the public key resolving the problem of public key validity. However, the computational power requirements of these pairings make their usage very limited. Among all pairings, Tate pairing is considered as the most convenient function in terms of computational cost [19, 20].

The use of Tate Pairing in cryptography first appeared with Victor Miller's unpublished paper [21] in 1986 and for a long time it was the most efficient way of computing the Tate Pairing. More recently, they are used to build cryptosystems with certain functionality. The concept became practical only with Boneh and Franklin in 2003 [16], who proposed the first IBE scheme. Until that time, a number of algorithms developed to calculate Tate Pairing and much progress have been achieved to make it more practical in applications. In this thesis, Tate Pairing using the modified Duursma-Lee algorithm in characteristic three is implemented.

The algorithm that is implemented in this work first appeared in [22] by Kwon. Even though it is possible to implement Tate pairing operations in software, it falls short of matching speed requirements of many pairing-based cryptography applications, especially in embedded systems. Therefore, despite the fact and necessity that designing dedicated hardware architectures gained significant importance, there is not much work on this subject in the literature. This modified Duursma-Lee algorithm was previously implemented as a full dedicated hardware partially. Aim in this thesis is to design an accelerator that reduces the computation time and area of Tate pairing in characteristic three, to make it practically more applicable on FPGA's and build the first ASIC implementation of Tate pairing.

This design was implemented using VHDL. Xilinx ISE was used as the design environment, while Modelsim was used as the simulation platform. For the ASIC, NEC's standard cell library built with 0.25µm technology was used.

The organization of this thesis is as follows. In chapter two, the basics of Elliptic Curve Cryptography and Tate Pairing are explained. In this context, finite fields, geometric approach to elliptic curves, discrete logarithm problem that enables elliptic curves to be used in cryptography, Identity Based Encryption, development of Modified Duursma-Lee algorithm implementing Tate Pairing, Tate Pairing calculation, related work on the subject and aim and contributions of the thesis are presented. In chapter three, arithmetic in characteristic three and design of sub-blocks are described. Representation of characteristic three, blocks on addition and subtraction, cubing and multiplication are discussed. In addition to these, the performances of blocks are compared to the previous works in the literature. In the next chapter, hardware implementation of Tate Pairing based on Modified Duursma Lee algorithm is presented. $GF(3^{6m})$ multiplication block, $GF(3^{6m})$ cubing block and our accelerator architecture is explained in detail. In chapter five the implementation aspects of the developed hardware is presented. FPGA implementation results are compared with similar work in the literature and the first ASIC implementation results are discussed. Conclusion is the last chapter of this thesis.

2. ELLIPTIC CURVE CRYPTOGRAPHY AND TATE PAIRING

2.1. Elliptic Curves

Elliptic curves have been studied for a very long time as mathematical entities. Based on these studies, it was seen that they have a potential for cryptographic applications. In 1985, cryptographic usage of Elliptic Curves was proposed by Koblitz and Miller independently from each other [12, 13]. Since then, many protocols were developed for cryptographic purposes, including the Tate pairing that is implemented in this thesis.

2.1.1. Finite Fields

Operations of ECC are based on finite fields. Fields consist of a set of elements with two basic operations, addition and multiplication. When the number of elements in these fields is finite, they are called finite fields. Finite fields are divided mainly into two sections; prime fields and extension fields. Prime fields are mainly represented as Galois Field (p), GF(p). Here p is a prime number and the points of the field are integers of modulo p. The other section is extension fields and represented as $GF(2^n)$ and $GF(p^n)$. In $GF(2^n)$ field elements can be represented as n-bit binary numbers. $GF(p^n)$ is also similar to $GF(2^n)$ but the base is a prime number rather than 2. In this field, prime numbers are generally chosen as small numbers such as 3, 7 etc. The modified Duursma-Lee Algorithm implementer here is also built in this extension field $GF(3^n)$ called characteristic three. Taxonomy of Finite Fields is presented in Figure 2.1.

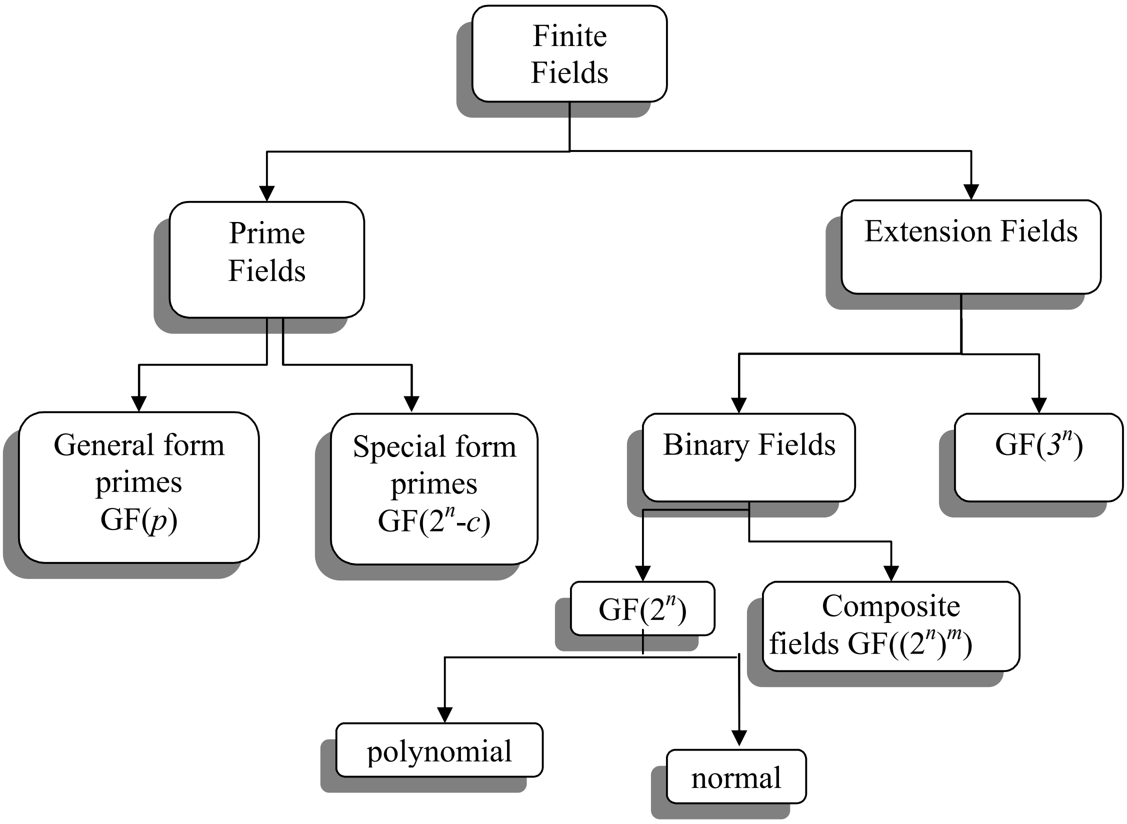

Figure 2.1. Finite Field Taxonomy

2.1.2. Geometric Approach

Points on an elliptic curve over finite fields form an additive group including the point at the infinity. The point at the infinity is defined as the identity element. These elliptic curve points are represented by two coordinates: P(x,y). Elliptic curves over real numbers (R) have a mathematical form of $y^2 = x^3 + a*x + b$ where x,y,a,b $\in$ R. Point addition and point doubling on elliptic curve in affine coordinates (The coordinates representing any point of an n-dimensional affine space A by an n-tuple of real numbers, thus establishing a one-to-one correspondence between A and $\mathbb{R}^n$.) are defined geometrically [23].

Point addition R = P + Q, P $\neq$ Q can be done geometrically as in Figure 2.2. In elliptic curves, the line connecting P & Q intersects the curve at the exact point of –R.

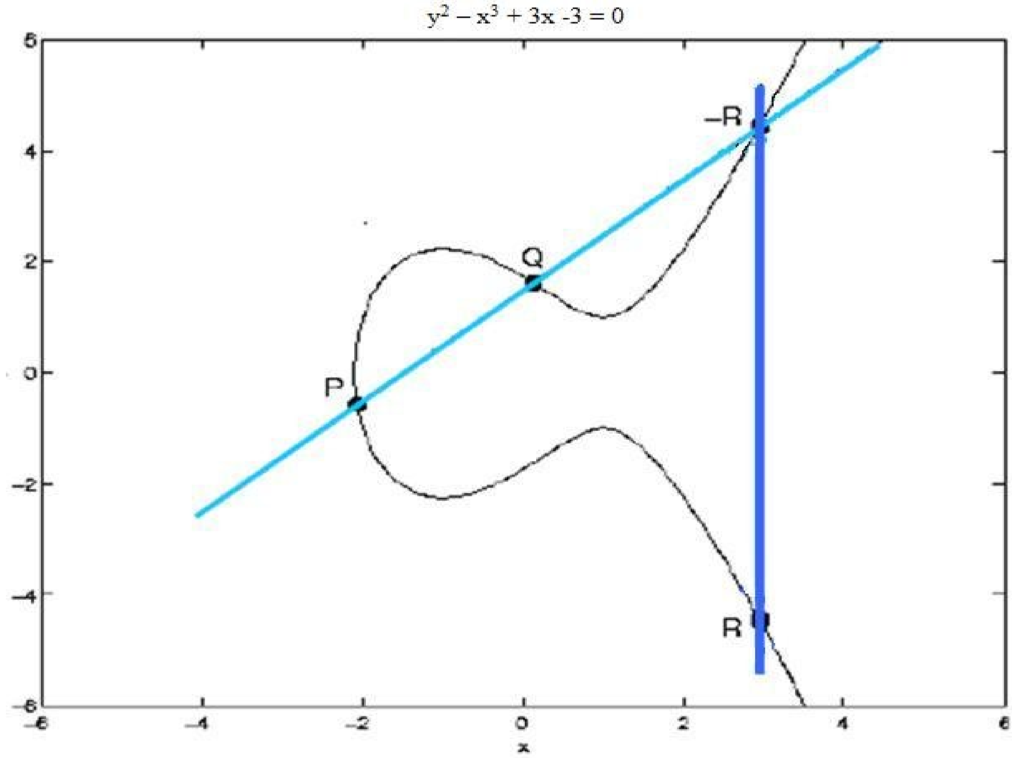

Figure 2.2. Point addition

Point Doubling R = P + Q = 2P can be done geometrically as in figure 2.3. The tangent to the point P intersects the curve on the exact point –R.

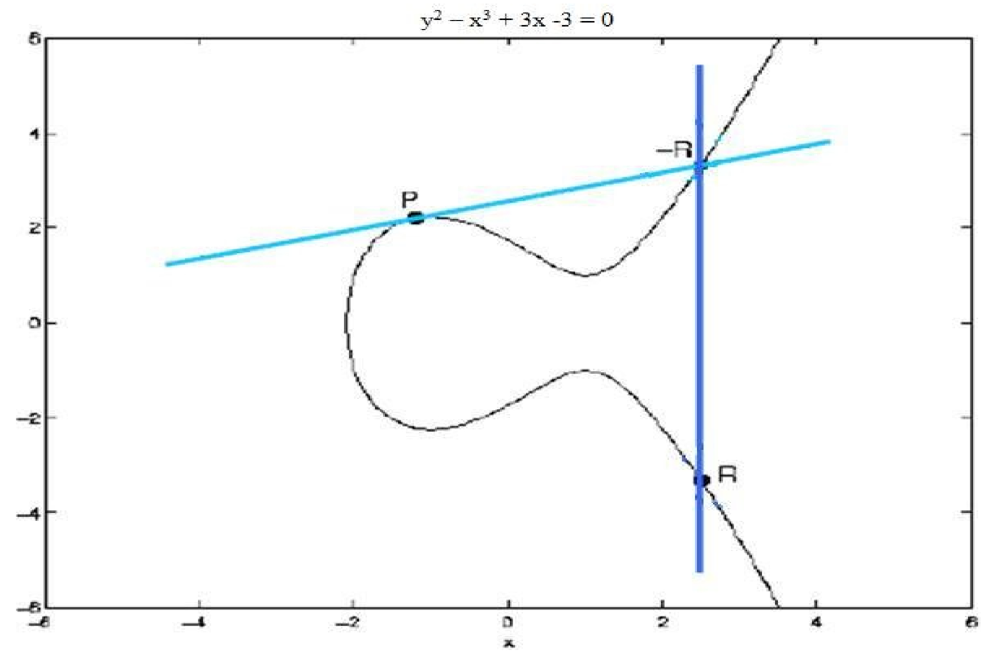

Figure 2.3. Point doubling

2.1.3. Elliptic Curve Discrete Logarithm Problem (DLP)

The property of Elliptic Curves that makes the elliptic curves possible to use in cryptography is the discrete logarithm problem. This problem can be defined as follows.

Scalar multiplication is a very time consuming operation on elliptic curves. $Q = k*P$, k is an integer and P and Q are points defined over the curves. The Elliptic Curve Discrete Logarithm Problem is the difficulty of calculating integer k when the points Q and P are known. Elliptic Curve Cryptosystem's security is directly proportional to the size of k. In commercial applications, the size of k is generally chosen more than 160 bits.

The advantage of ECC over RSA is also depends on the hardness of this problem, since the discrete logarithm problem over elliptic curves is a harder problem than the discrete logarithm problem over integers mod p. The main solving method of DLP over integers "Index Calculus Method" can not be applied to DLP over elliptic curves. Hence it is possible to achieve the same level of security with smaller key sizes than RSA [14].

2.2. Identity Based Encryption

Identity-based encryption (IBE), which is perhaps the most important application of pairing-based cryptography, is a public key cryptosystem that allows any arbitrary string to be used as a public key, such as recipients' email address. This vastly reduces the amount of work on behalf of the sender to set up an online lookup for public keys and presents novel functionalities especially useful in access control systems and maintaining privacy and anonymity. In these systems, trusted third parties called Private Key Generator (PKG) generate the private keys correspondingly. In this operation, a master private key is established. With this master private key, a private key corresponding to the identity can be computed, for instance by combining the email address and master key. By this way, messages can be encrypted without the distribution of keys between the individuals. This is definitely very useful under situations where the authenticated key distribution is not feasible. During the decryption phase, the appropriate key should be obtained from the PKG by the authorized user. In this system, the trustworthiness of PKG is highly important since it has the capability of generating users' private key and hence decrypting their messages [15, 19].

Shamir introduced the concept of identity-based cryptography in 1984 [15]. However, the concept became practical only with Boneh and Franklin in 2003 [16], who proposed the first Identity Based Encryption (IBE) scheme, by following the idea of Kasahara et. al. who used bilinear maps, or pairings over elliptic curves for their scheme [17]. Today many cryptographic protocols are based on pairings.

2.3. Development of Modified Duursma-Lee algorithm implementing Tate Pairing

Among different pairings, Tate Pairing, originally developed to attack the discrete logarithm problem of elliptic curves defined over finite fields by Frey and Rück [24], became popular, since it is efficiently computable and achieves its maximum security in characteristic three over super singular elliptic curves [25]. For quite a while, Miller's Algorithm [21] had been the most efficient way of Tate pairing until two different works [26,27] in 2002 improved the method by reducing the computational complexity. Later, in [27,28] tower fields of $GF(3^m)$, $GF(3^{6m})$ was proposed. In 2003 Duursma and Lee in [29] further improved the implementation of Tate Pairing presenting Duursma-Lee algorithm, and extending the computation to hyperelliptic curves. The Duursma-Lee algorithm calculating the Tate Pairing is described in Figure 2.4. The main difference of Duursma-Lee algorithm from the algorithm implemented in this thesis is the need for cube root operations, which results in a harder implementation.

Figure 2.4. The Duursma-Lee Algorithm (char 3)[29] calculating the Tate Pairing in characteristic three

---

input: $P = (x_p, y_p)$, $R = (x_r, y_r)$

output: $t = f_P, (\phi(Q)) \in F^{*}_{q}{}^{6} / F^{*}_{q}{}^{3}$

$f = 1$

for i in 0 to m-1 loop

$\quad x_1 = x_1^3$

$\quad y_1 = y_1^3$

$\quad \mu = x_1 + x_2 + b$

$\quad \gamma = -y_1 y_2\, \sigma$

$\quad g = \gamma - \mu p - p^2$

$\quad f = fg$

$\quad x2 = x_2^{1/3}$

$\quad y2 = y_2^{1/3}$

end loop

return: f

The algorithm that is implemented in this thesis first appeared in [22] by Kwon with further improvements and eliminating the cube root operation at the expense of two extra cubing operations. However, implementing pairing operations in software falls short of matching speed requirements of many pairing-based cryptography applications, especially in embedded systems. Therefore, designing dedicated hardware architectures gained significant importance and became a necessity, since there is not much work on this subject in the literature. This modified Duursma-Lee algorithm was previously implemented as a dedicated hardware partially only in [25] on FPGA. Aim in this thesis is to design an accelerator that reduces the computation time and area of Tate pairing in characteristic three, to make it practically more applicable on FPGA's and build the first ASIC implementation of Tate pairing.

2.4. Tate Pairing Calculation and Modified Duursma-Lee Algorithm

The Tate pairing is basically a transformation that takes two points on an elliptic curve $E_{\pm} : y^2 = x^3 - x \pm 1$ defined over ternary extension field GF(3^m) and outputs a nonzero element in GF(3^{6m}) [219]. The modified Duursma-Lee algorithm described in Figure 2.5 computes a nonzero element of GF(3^{6m}), which needs to be raised to the Tate power $\epsilon_1 = 3^{3m}$-1 in order to obtain the result of the pairing. The final exponentiation can be performed in the same circuitry used for computation of, the Modified Duursma-Lee Algorithm and takes comparably much less time[30].

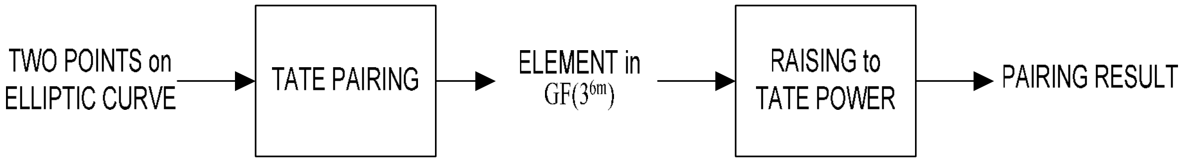

Figure 2.5. Tate Pairing and raiseing to Tate power scheme

Figure 2.6. The Modified Duursma-Lee Algorithm (char 3)[25]

---

input: $P = (x_p, y_p), R = (x_r, y_r) \in E_{\pm}[31](\mathrm{GF}(3^m))$

output: $t = e_3^{3m\text{-}1}(P, \phi(R)) \in \mathrm{GF}(3^{6m})^*$

initialize : $t = 1 \in \mathrm{GF}(3^{6m})$,

$\alpha = x_p, \beta = y_p, x = x_r^3, y = y_r^3, \mu = 0 \in \mathrm{GF}(3^m)$

$d = (\pm m) \bmod 3 \in \mathrm{GF}(3)$

for i in 0 to m-1 loop

$\alpha = \alpha^9, \beta = \beta^9$ (* arithmetic in GF(3^m) *)

$\mu = \alpha + x + \mathrm{d}$ (* arithmetic in GF(3^m) *)

$\gamma = (-\mu^2)\zeta^0 + (-\beta y)\zeta^1 + (-\mu)\zeta^2 + (0)\zeta^3 + (-1)\zeta^4 + (0)\zeta^5$ (*$\zeta = 3^m$*)

$t = t^3$ (* cubing in GF(3^{6m}) *)

$t = t\gamma$ (* multiplication in GF(3^{6m}) *)

$y = -y$ (* arithmetic in GF(3^m) *)

$d = (d \pm 1) \bmod 3$

end loop

return: t

Inputs to the modified Duursma-Lee algorithm are two points, $P = (x_p, y_p)$, $R = (x_r, y_r)$ on an elliptic curve constructed in GF(3^m). Here m is chosen as a prime number such as 97. As the value of m gets bigger, the security of the crypto algorithm increases as well. The output of the algorithm, t, is an element of GF(3^{6m}).

Modified Duursma-Lee algorithm consists of mainly two parts; initialization and computation loop as shown in Figure 2.6. In the initialization part, two elements of the first point, x_p, y_p, are directly assigned to the registers and the other two elements that represents the second point x_r, y_r are assigned after cubing in GF(3^m), also three values are assigned to internal elements t, μ and d.

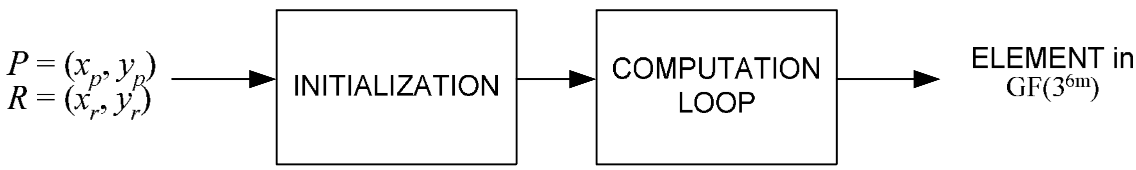

Figure 2.7. Tate Pairing calculation structure

The second part, loop section is executed m times to calculate the result. The first step in the loop is twice cubing of, x_p, y_p, in GF(3^m). In the next step, three elements of GF(3^m) are added. Then, one squaring, one multiplication and one negation is performed in GF(3^m). Next operation is cubing of the t variable in GF(3^{6m}). Then a multiplication is done in GF(3^{6m}). The last two operations are negation in GF(3^m) and addition in GF(3). At the end of this calculation t is returned as the result of the pairing. This structure of the calculation is shown in Figure 2.7.

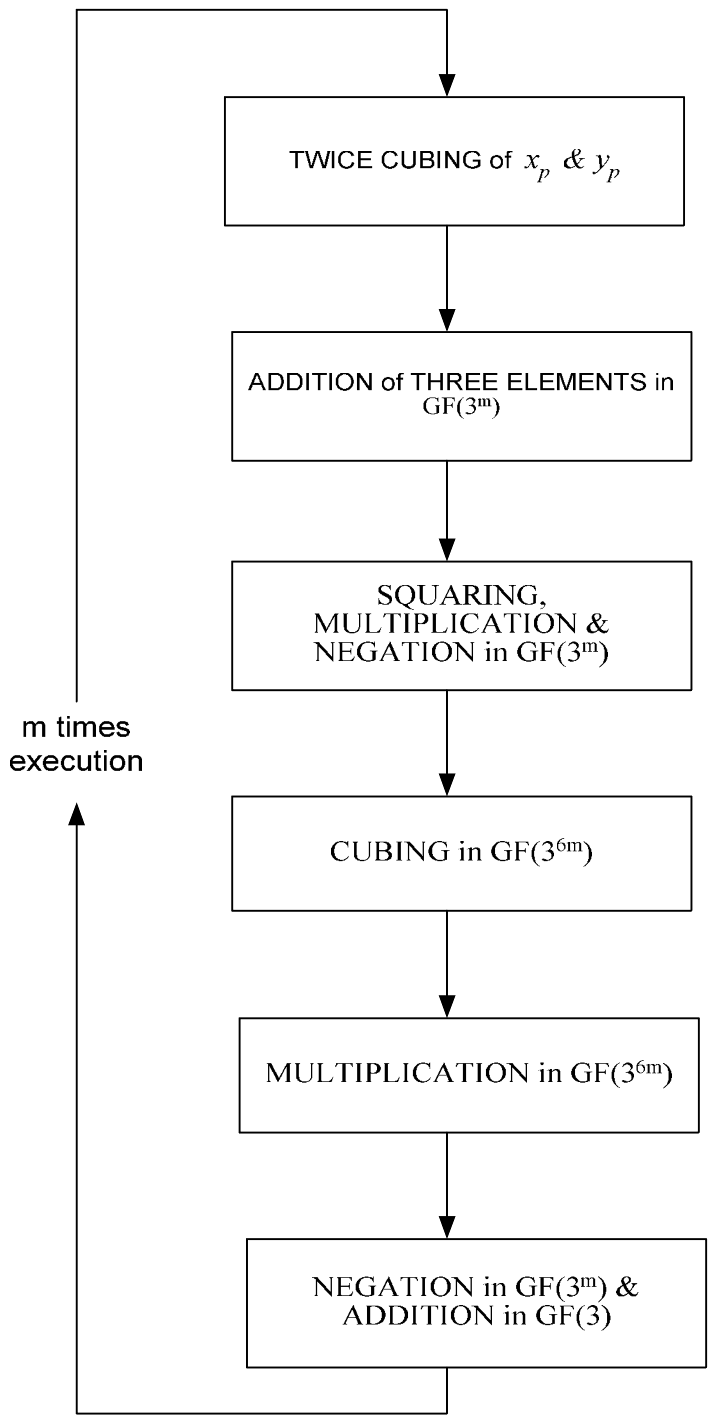

Figure 2.8. Computation loop structure

Arithmetic operations required to implement Modified Duusma-Lee Algorithm is addition, subtraction, cubing and multiplication in GF(3^m) and cubing and multiplication in the tower extension GF(3^{6m}). Cubing is relatively inexpensive operation in characteristic three similar to ease of squaring in binary fields and can be implemented by just using combinational gates in a single clock cycle. Therefore, multiplication in GF(3^{6m}) is the main complexity of the pairing[30].

Constructing the ternary extension field GF(3^{6m}) on the base field of GF(3^m) is suggested in [28,27] and described explicitly in [25]. Use of extension fields simplifies the arithmetic operations, allows parallelization for the cubing and multiplication operations, and finally renders the implementation suitable for hardware.

2.5. Related Work

With the advent of elliptic curve cryptography, GF(p) and GF(2^m) arithmetic attracted enormous attention and important amount of work appeared in the literature providing fast and efficient hardware accelerators. In contrast, very small interest in the arithmetic of more general extensions field of GF(p^m) stems from the fact that the need for it has recently appeared with pairings defined over extension fields of characteristic three. One of the earliest studies is by Page and Smart [32] which described GF(3^m) arithmetic architectures for cryptographic applications. They later, implemented Tate pairing with Duursma-Lee algorithm using an accelerator for arithmetic in GF(3^m) [31].

Another work by Kerins et. al [33] implements Miller's algorithm. They have also inversion blocks in characteristic three since Miller's algorithm requires multiplicative inversion operations. In addition, Bertoni et. al. [34] presented efficient GF(p^m) architectures for cryptographic applications that specifically focus on different multipliers with modulo reduction. They, also provided a case study for GF(3^m) multipliers.

Last work mentioned here is Kerins, Marnane, Popovici and Barreto's Tate pairing implementation [25] based on the modified Duursma-Lee algorithm. They presented parallel multiplication and cubing units to implement GF(3^{6m}) tower field arithmetic. By this approach, it is possible to multiply two ternary polynomials in GF(3^{6m}) using the same number of clock cycles as multiplying two GF(3^m) polynomials, at the expense of area overhead and reduced clock frequency. It is also possible to find intermediate solutions to reduce the hardware complexity with an increase in the number of clock cycles. For instance this can be achieved by scheduling the addition operations after the GF(3^m) multiplication blocks used in GF(3^{6m}) multiplication block. This reduced the number of adders used at this stage but increases the number of clock cycles required [30].

2.6. Aim and Contributions

This modified Duursma-Lee algorithm was previously implemented as a dedicated hardware partially only in [25] on FPGA. Aim of the thesis is to design an accelerator that reduces the computation time and area of Tate pairing in characteristic three, to make it practically more applicable on FPGA's and build the first ASIC implementation of Tate pairing based on the NEC's 0.25 µm standard cell technology with 5 metal routing layers.

Contributions can be summarized as follows. First, the work is the first full implementation of modified Duursma-Lee algorithm on both FPGA and ASIC. In this thesis, sub blocks, datapath and the control unit that calculates the Tate Pairing via modified Duursma-Lee algorithm have been built all in hardware. The VHDL codes of the design than mapped to proper FPGA devices and synthesized according to standard cell approach by using NEC's 0.25 µm cell library. Even though there are partial works on the subject, there is not a full hardware accelerator in the literature for FPGA or ASIC.

Second, it is demonstrated that subunits in the accelerator and the accelerator itself can be improved in terms of both area and time complexity compared to previous works in the literature by applying different design techniques. Improvements have been achieved in the least-significant-element-first (LSE) multiplier unit compared to the one in [25]. In addition to this, the cubing subunit of the accelerator has been improved significantly compared to the ones in [25] and [34].

Third, the actual implementation of modified Duursma-Lee algorithm is in fact faster and smaller than the estimated values given in the previous work. The calculation time of the algorithm has been improved up to 16 times compared to the best known software implementation [35] and more than 3 times compared to best known hardware implementation [25]. Also, time and area product has also improved 12 times with the suggested implementation [30].

3. ARITHMETIC IN CHARACTERISTIC THREE AND DESIGN OF THE SUB-BLOCKS

In this section, hardware architectures for addition, subtraction, multiplication and cubing in GF(3^m) are presented.

3.1. Characteristic Three Representation

Characteristic three arithmetic is slightly more complicated than characteristic two arithmetic since coefficients can take three values; {0, 1, 2}. Now, two bits are needed to represent each digit in GF(3). There are two common representations:

$$\{0, 1, 2\} = \{00, 01, 10\} \tag{3.1}$$

$$\{0, 1, 2\} = \{\{00, 01\}\ 10, 11\} \tag{3.2}$$

The advantage of the latter representation is that “check if zero operation” is implemented by only checking the most significant bit of the digit since both alternatives for representing digit {0} have 0 in the most significant position. The disadvantage, however, is that negation is performed by subtracting the digit from zero, which can be done by using the addition circuit again in one clock cycle. The negation, on the other hand, in the former representation is performed by just swapping the most and the least significant bits. This operation can be implemented just by wiring, without active area consumption. Also saving one clock cycle compared to the latter representation in negation is a critical aspect for achieving a high speed design. Since negation operation is used very often especially in performing GF(3^{6m}) multiplication, the former representation is more advantageous in this case by minimizing the operation time and active area usage.

For arithmetic operations, m bit elements are expressed as 2m bit arrays as follows:

$$A = (\{a^H{}_{m-1}, a^L{}_{m-1}\}, \ldots\ldots\ldots, \{a^H{}_1, a^L{}_1\}, \{a^H{}_0, a^L{}_0\}) \tag{3.3}$$

3.2. Addition and Subtraction

Addition and subtraction are performed component wise by using the Boolean expression in [31], i.e.

$$C_i = A_i + B_i, \text{ for } i = 0, 1, \ldots, m\text{-}1 \tag{3.4}$$

$$t = (A^L{}_i \vee B^H{}_i) \oplus (A^H{}_i \vee B^L{}_i) \tag{3.5}$$

$$C^H{}_i = (A^L{}_i \vee B^L{}_i) \oplus t \tag{3.6}$$

$$C^L{}_i = (A^H{}_i \vee B^H{}_i) \oplus t \tag{3.7}$$

where $\vee$ and $\oplus$ stands for logical OR and EXOR operations, respectively. In the representation, negation and multiplication of GF(3) element by two are equivalent operations and performed by swapping the most and least significant bits of the digit representing the element. Therefore, subtraction in GF(3^m) is equally efficient as the addition in the same field and thus the same adder block is used for both operations. If subtraction is needed, bits in each GF(3^m) element are individually swapped and connected to the adder block. Since this is achieved by only wiring no additional hardware resource is used.

$$\text{-A} = (\{a^L{}_{m\text{-}1}, a^H{}_{m\text{-}1}\}, \ldots\ldots, \{a^L{}_1, a^H{}_1\}, \{a^L{}_0, a^H{}_0\}) \tag{3.8}$$

$$2\text{*A} = (\{a^L{}_{m\text{-}1}, a^H{}_{m\text{-}1}\}, \ldots\ldots, \{a^L{}_1, a^H{}_1\}, \{a^L{}_0, a^H{}_0\}) \tag{3.9}$$

When implemented on FPGAs, for each GF(3) element addition, two 4-input "look-up tables" (LUTs) are used. Since one slice is composed of two LUTs, for *m*-bit long GF(3^m) additions, *m* slices are used. This result is almost the same in all papers implementing characteristic three addition such as [36]. The delay of the addition operation is 5,061 ns on Xilinx Virtex2p 100 device [37].

The VHDL code for this block is written in generic format for any *m* bit addition.

3.3. Cubing

For the Modified Duursma-Lee algorithm, cubing operation in GF(3^{6m}) is needed to calculate

$$t = t^3 \qquad (t \in GF(3^{6m})) \tag{3.10}$$

in the loop part of the algorithm. For this operation there are mainly two methods to build the hardware architecture. The first method is processing serially by using a GF(3^m) cubing block and a number of adder/subtracter blocks as shown in Figure 3.1. In this method, the area consumption of combinational circuits is relatively small since only one GF(3^m) cubing circuit is enough. However, a significant number of registers or RAM unit is needed to store the intermediate values in this approach. Also the clock count is much more than the parallel implementation.

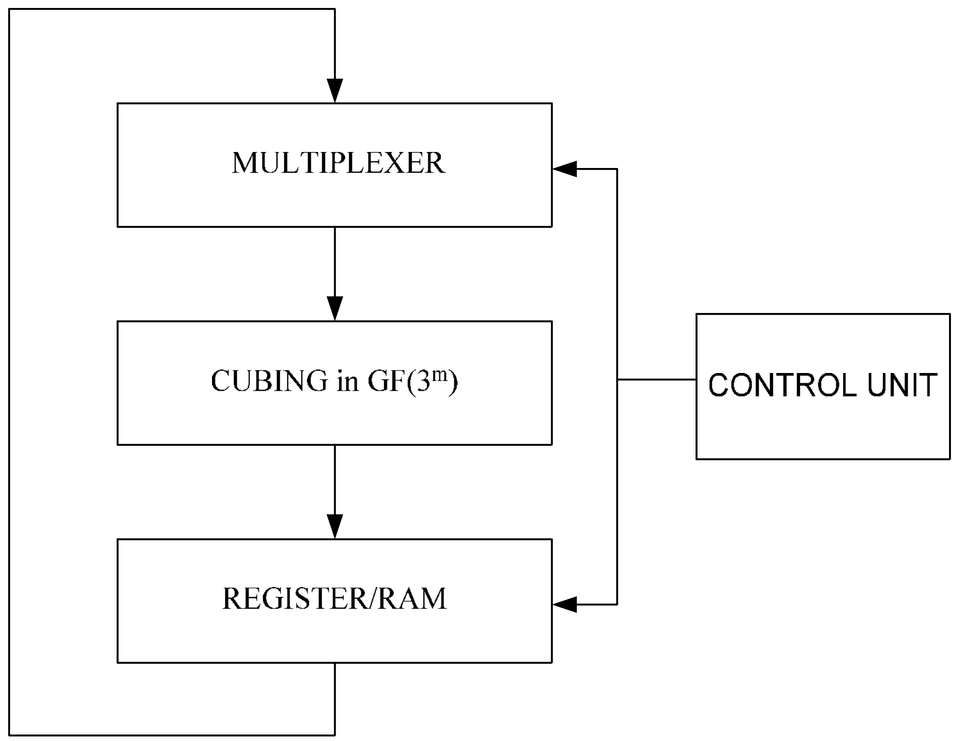

Figure 3.1. Serial calculation of Cubing in GF(3^{6m})

The second method is building a parallel GF(3^{6m}) cubing architecture by using GF(3^m) cubing blocks. In this method, the result is calculated in just one clock cycle and no register unit is needed to store any intermediate values. A total of 6 GF(3^m) cubing circuits and a number of adder/subtracter units are used in parallel in this structure. In this work, it is more advantageous to use the second method since clock count is much less and the control is also easier than the serial computation. The structure is explained in the next section for GF(3^{6m}) cubing computation.

In this section our aim is to build an optimum cubing circuit in GF(3^m). Cubing is a linear operation in characteristic three and the technique presented in [34] is adopted.

For characteristic three, the Frobenius map is written as follows:

$$A^3 \equiv \left(\sum_{i=0}^{m-1} a_i x^i\right)^3 \bmod p(x) = \sum_{i=0}^{m-1} a_i x^{3i} \bmod p(x) \tag{3.11}$$

This formula can be represented as follows:

$$A^3 \equiv \sum_{\substack{i=0 \\ i \equiv 0 \bmod 3}}^{3(m-1)} a_{i/3} x^i \bmod p(x) \equiv T + U + V \bmod p(x)$$

$$\equiv \left(\sum_{\substack{i=0 \\ i \equiv 0 \bmod 3}}^{m-1} a_{i/3} x^i\right) + \left(\sum_{\substack{i=m \\ i \equiv 0 \bmod 3}}^{2m-1} a_{i/3} x^i\right) + \left(\sum_{\substack{i=2m \\ i \equiv 0 \bmod 3}}^{3(m-1)} a_{i/3} x^i\right) \bmod p(x) \tag{3.12}$$

Here the degrees of the second term U and the third term V are bigger than m and need to be reduced. For $p(x) = x^m + p_t x^t + p_0$ and $t < m/3$, the terms can be represented as follows [34]:

$$U = \sum_{\substack{i=m \\ i\equiv 0 \bmod 3}}^{2m-1} a_{i/3}x^{i} \bmod p(x) = \sum_{\substack{i=m \\ i\equiv 0 \bmod 3}}^{2m-1} a_{i/3}x^{i-m}(-p_t\, x^{t} - p_0) \bmod p(x) \tag{3.13}$$

$$V = \sum_{\substack{i=2m \\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{i/3}x^{i} \bmod p(x) = \sum_{\substack{i=2m \\ i\equiv 0 \bmod 3}}^{3(m-1)} a_{i/3}x^{i-2m}(a^{2t} - p_t p_0 a^{t} + 1) \bmod p(x) \tag{3.14}$$

Reduction is basically done by additions. For irreducible polynomials, $p(x) = x^m + p_t x^t + p_0$, each x^m and x^{2m} are replaced with $(-p_t x^t - p_0)$ and $(a^{2t} - p_t p_0 a^t + 1)$, respectively. However, the terms with degrees equal to or bigger than m still remain after the first reduction step. This problem can be solved by performing reduction one more time. The result of the first reduction can be stored in a register and the second reduction can be performed in the next clock cycle. This naturally increases the maximum operating frequency of the block. However since the cubing circuit is not in the critical path, the second reduction step is implemented in the same clock cycle as the first reduction step. This structure enabled to complete the cubing operation in just one clock cycle and without using any registers.

Reduction is optimized for the well known polynomial $p(x) = x^{97} + x^{16} + 2$, [25] and the terms are calculated to be added in order to achieve reduction in the same clock cycle. This optimization for a specific polynomial results in a very efficient implementation. We used 111 GF(3) adders to complete the cubing operation. Critical path of the implemented system consists of three serially connected GF(3) adders. The general structure of our cubing block is presented in Figure 3.2. As seen from Table 3.1, implementation in this thesis is 2 to 5 times more efficient than the implementations reported in literature, namely [25] and [34]. Although the implementation details of the cubing circuits are not clear in [25] and [34], the improvement in the slice and LUT numbers should be due to register free design and doing the reduction for a fixed given polynomial [30].

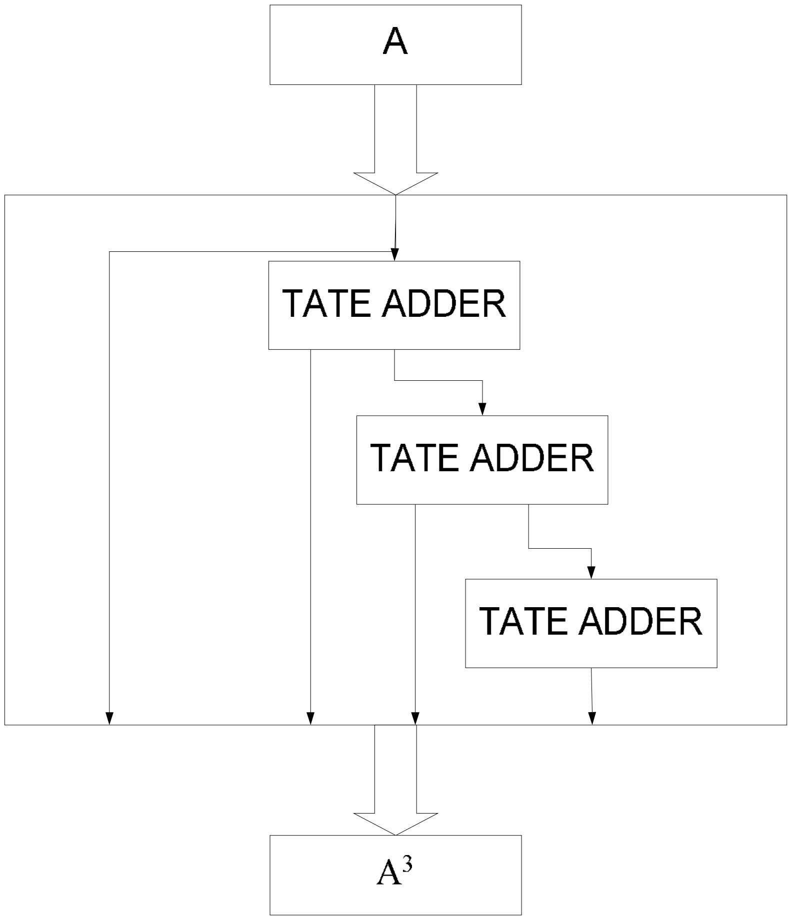

Figure 3.2. General structure of cubing circuit for fixed polynomial in GF(3^m)[30]

Table 3.1. Comparison of cubing circuits for GF(3^{97})

Cubing circuit	Proposed circuit	Circuit in [25]	Circuit in [34]
Number of Slices	116	514	
Number of LUTs	222		388*
Max. Frequency	144MHz		

*Estimation by authors of [34]; not the result of an actual implementation.

Another advantage of this cubing block is its suitability for full custom design. The reason for this lies in, building the whole block by just using a basic block, GF(3) tate adder. Due to this, a very regular structure can be maintained and total area of the block can be minimized. Also, optimizing the GF(3) tate adder block will further lead to faster and smaller circuits which is the main goal of full custom designs.

3.4. Multiplication

Multiplication is the most important operation for pairing implementations due to its complexity. Since the modified Duursma Lee algorithm requires GF(3^{6m}) multiplications, 18 GF(3^m) multipliers are needed in parallel, as explained in the next section. Therefore, designing an efficient GF(3^m) multiplier architecture is the key for an efficient hardware accelerator.

Hardware architectures proposed in the literature for GF(3^m) multiplication can be treated in three major classes: parallel, serial and digit multipliers [34]. First, parallel multipliers multiply two GF(3^m) elements in one clock cycle. Although parallel multipliers sustain a high throughput, they consume prohibitively large amount of area and reduce the maximum clock frequency due to very long critical path. Since area and time complexity are very critical parameters for the practical usage of pairings, parallel multipliers are not appropriate on constrained devices.

Second, serial multipliers process a single coefficient of the multiplier at each clock cycle. These types of multipliers require m clock cycles for each GF(3^m) multiplication, while their area consumption and critical path delay are relatively small compared to other types of multipliers.

Finally, digit multipliers are very similar to serial multipliers but they process n coefficients of the multiplier at each clock cycle rather than a single coefficient. Consequently, the operation is completed in $\lceil m/n \rceil$ cycles. The area consumption is more than the serial multipliers and increases with n. Since the area critical path delay also increases with n, choosing n is an important decision which is mainly influenced by area and time concerns. An algorithm for Digit Multiplier is presented in Figure 3.3.

Figure 3.3. Digit Multiplier [34]

Require: $A = \sum_{i=0}^{m-1} a_i\alpha^i$, where $a_i, \in GF(p)$, $B = \sum_{i=0}^{m/D-1} b_i\alpha^{Di}$, where $b_i = \sum_{j=0}^{D-1} b_{Di+j}\,\alpha^j$

Ensure: $C \equiv A \,.\, B = \sum_{i=0}^{m-1} c_i\alpha^i$, where $c_i \in GF(p)$

$C \leftarrow 0$

for $i = 0$ to $\lceil m/D \rceil - 1$ do

$C \leftarrow b_i A + C$

$A \leftarrow A\,\alpha^D \bmod p(\alpha)$

end for

Return (C)

Serial multipliers are preferred to be used in our implementation, which incur increased number of clock cycles, while providing a better solution in terms of area and frequency. Serial multipliers can also be treated in two classes:

i) least-significant-element-first (LSE)

ii) most-significant-element-first (MSE).

Although there is not much difference between the two types of multipliers the LSE Multiplier is implemented.

As illustrated in Figure 3.4 below, the reduction is performed in interleaved fashion.

Figure 3.4. LSE Multiplier [34]

Require: $A = \sum_{i=0}^{m-1} a_i\alpha^i$, $B = \sum_{i=0}^{m-1} b_i\alpha^i$, where a_i, $b_i \in$ GF(p)

Ensure: $C \equiv A \,.\, B = \sum_{i=0}^{m-1} c_i\alpha^i$, where $c_i \in$ GF(p)

$C \leftarrow 0$

for i = 0 to m - 1 do

$C \leftarrow b_iA + C$

$A \leftarrow A\ \alpha \bmod p(\alpha)$

end for

Return (C)

For interleaved reduction, we subtract $a_m(p_{m-1}x^{m-1} + \ldots + p_1x + p_0x)$ from the partial result C whenever $a_m \neq 0$ since $x^m = - p_{m-1}x^{m-1} - \ldots - p_1x - p_0$.

Two LSE multipliers are designed to examine the effect of fixed versus generic polynomials on time and space complexities. In the generic design, shown in Figure 3.5, polynomial is given as input to the block. The advantage of the generic design is that it can be used with any polynomial in characteristic three. This is an important flexibility for systems that may use more than one polynomial.

In case of fixed polynomial multiplier, shown in Figure 3.6, the coefficients of the polynomial can be hardcoded into the multiplier unit resulting in reduction of design complexity. For the fixed irreducible polynomial of $x^{97} + x^{16} + 2$, used in many pairing based cryptographic systems in literature, only one GF(3^m) additions are needed in each iteration of interleaved reduction. As illustrated in Table 3.2, the multiplier with hardcoded irreducible polynomial is 30% better than the generic multiplier in terms of area. Final architecture is synthesized with both multiplier blocks and the results are presented in the next section.

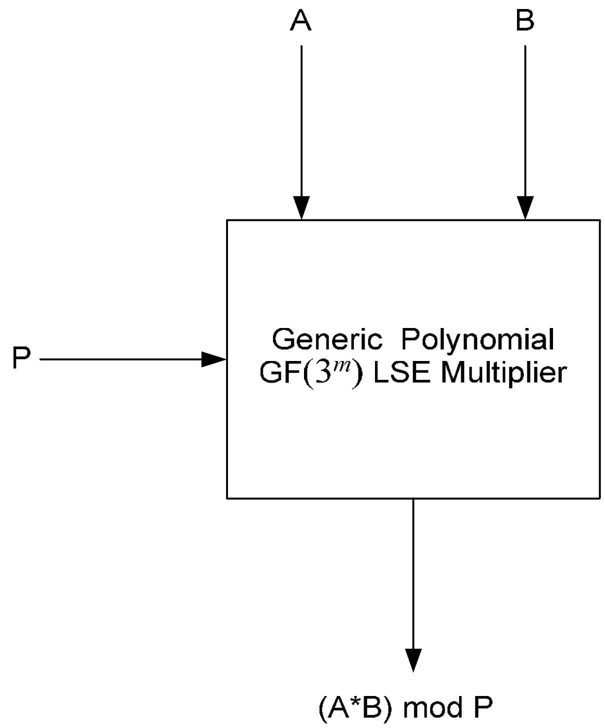

Figure 3.5. Generic Polynomial GF(3^m) LSE Multiplier

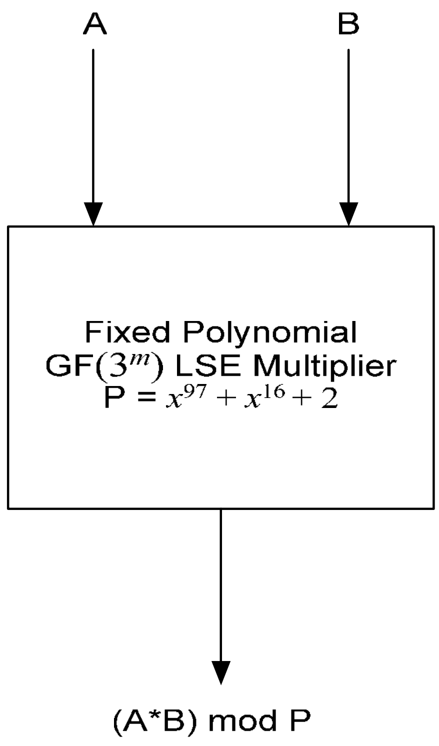

Figure 3.6. Fixed Polynomial GF(3^m) LSE Multiplier

Table 3.2. Comparison of LSE Multiplication circuits in GF(3^{97})

GF(3^{97}) Multiplier	Fixed LSE	Generic LSE	LSE in [25]	LSE in [34]
Number of Slices	389	599	1006	
Number of LUTs	727	1166		600*(LUT+FF)
Max. Frequency	161MHz	161MHz		
Total time (μs)	0,61	0,61		

*Estimation by paper's author; not the result of an actual implementation

The proposed GF(3^m) LSE multiplier architecture is shown in Figure 3.7.

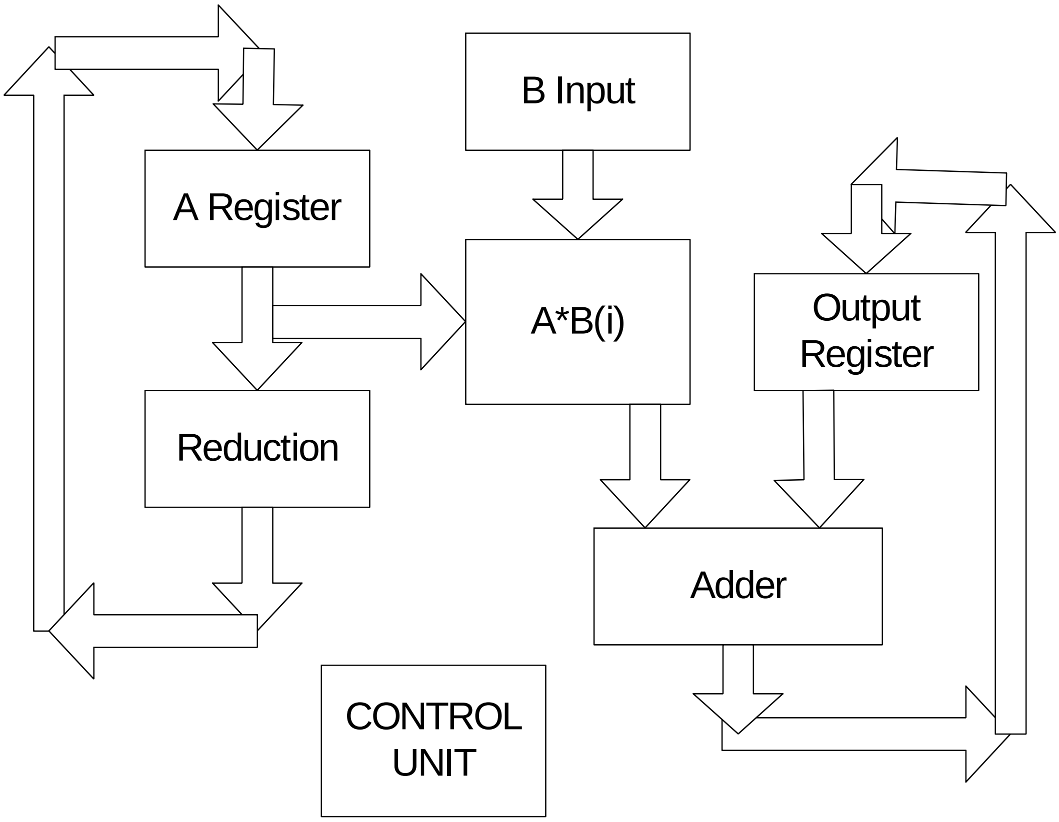

Figure 3.7. LSE multiplier architecture over GF(3^m)

The proposed multiplier is implemented for m = 97 on virtex2p-100 for comparison purposes since it is the same Xilinx device also used in [25]. As shown in Table 3.2, the fixed multiplier is nearly 2.5 times smaller than the architecture in [25] and the generic multiplier consumes around 60% of the area of the same architecture. Since the architecture in [25] is not described in detail, only informed guesses can be provided for the reasons of the improvement. One reason may be the fact that the multiplicand is not stored within the multiplier block and gets its coefficients from the inputs of the block, reducing the number of registers by 194 for $m = 97$. In Table 3.2, the estimation by the authors is included for the implementation of LSE multiplier in [34]. Although the conversion from LUTs and flip-flops to the number of slices cannot be done easily, it should consume at least 600 slices when the design is placed and routed on an FPGA. As a result the architecture presented is better than the architectures in the literature to the best of our knowledge.

4. HARDWARE IMPLEMENTATION OF TATE PAIRING BASED ON MODIFIED DUURSMA LEE ALGORITHM

4.1. GF(3^{6m}) Multiplication Block

As described in [25], GF(3^{6m}) can be considered as an extension field over GF(3^{2m}) with irreducible polynomial $z^3 - z \pm 1$. Also, as suggested again in the same work [25], the multiplication in GF(3^{6m}) can be done in two steps:

i) Karatsuba multiplication for polynomials with coefficients from GF(3^{2m}) [25], and

ii) Reduction with irreducible polynomial $z^3 - z \pm 1$.

Reader can profitably refer to [25] for further details.

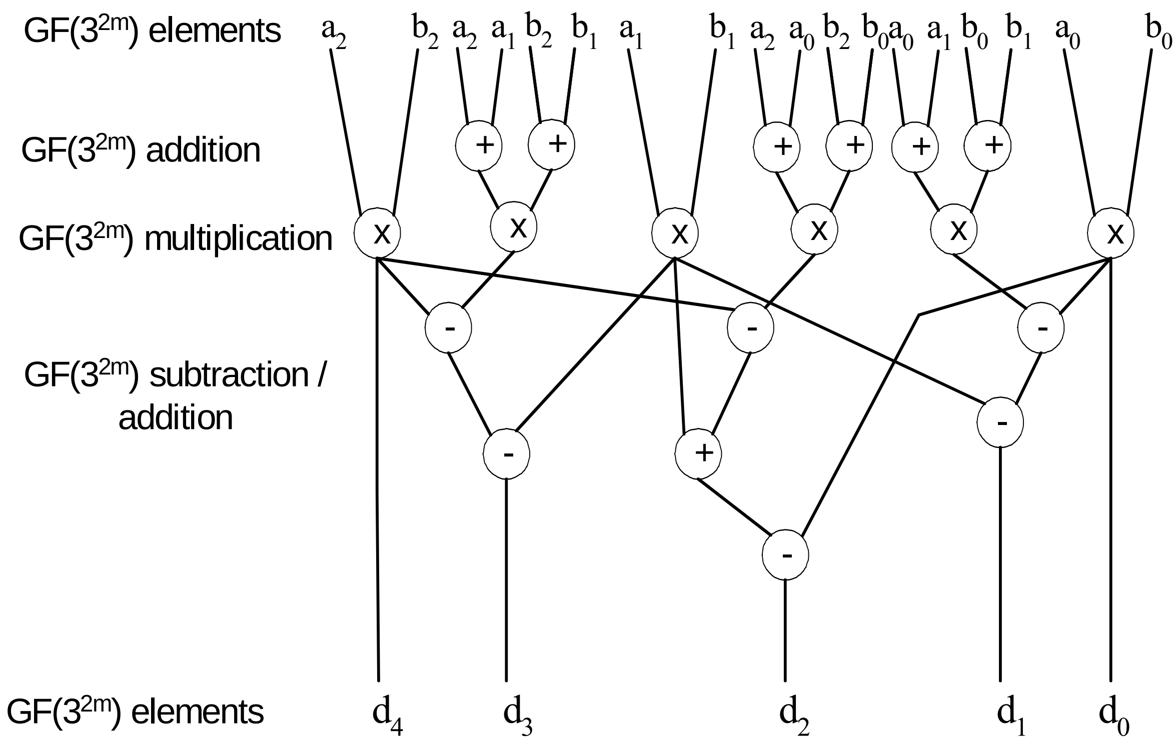

Figure 4.1. GF(3^{6m}) multiplier unit from [25]

In Figure 4.1 GF(3^{6m}) Karatsuba multiplier unit, as proposed in [25], is illustrated, where nodes represent the GF(3^{2m}) adders, subtractors, and multipliers. As seen from the Figure 4.1 this block uses 6 multipliers, 7 adders and 6 subtractors in total. Similarly, GF(3^{2m}) can also be seen as an extension field over GF(3^{m}) with irreducible polynomial $y^2 + 1$. Since the adder/subtracter units operate on the corresponding coefficients of the operand polynomials, their structure is the same as GF(3^{m}) adders. GF(3^{2m}) multiplier, however, consists of GF(3^{m}) adders, subtractors, and multipliers as seen in Figure 4.2. Each GF(3^{2m}) multiplier block uses 3 multipliers, 2 adders and 3 three subtractors in total to output two GF(3^{m}) elements, c_0 and c_1. These elements represent one GF(3^{2m}) element with c_0 the least significant part and c_1 the most significant part.

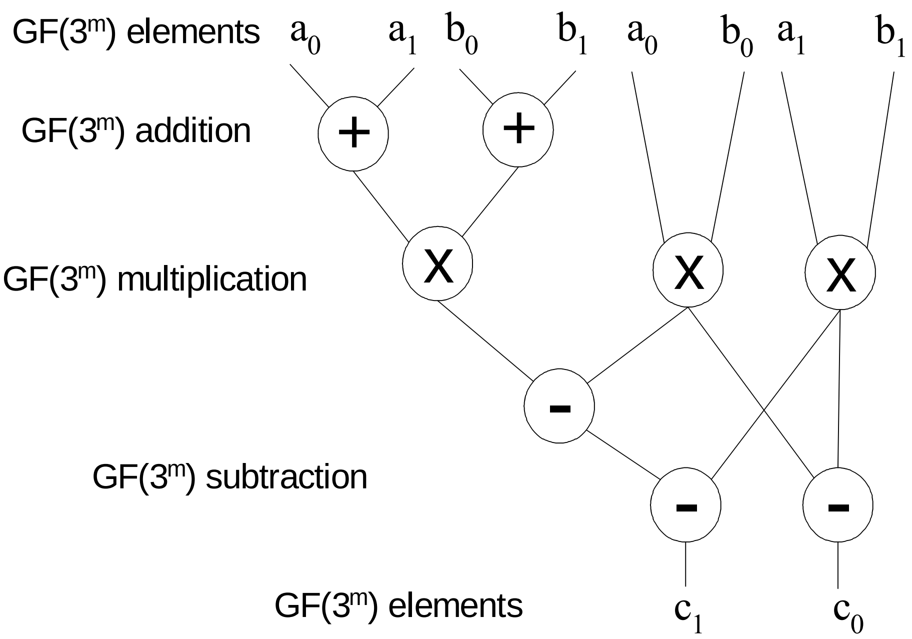

Figure 4.2. GF(3^{2m}) multiplier unit from [25]

As seen in Figure 4.1, GF(3^{6m}) Karatsuba multiplier has five GF(3^{2m}) elements as output. The result of the Karatsuba multiplier has the form

$\tilde{d}_4 z^4 + \tilde{d}_3 z^3 + \tilde{d}_2 z^2 + \tilde{d}_1 z + \tilde{d}_0$. Since $z^3 = z + 1$ from the irreducible polynomial, we have $(\tilde{d}_2 + \tilde{d}_4)z^2 + (\tilde{d}_1 + \tilde{d}_4 + \tilde{d}_3)z + (\tilde{d}_0 + \tilde{d}_4)$.

To summarize, 18 GF(3^m) multipliers and 52 GF(3^m) adders are used in one GF(3^{6m}) multiplier. The advantage of the proposed architecture is that multiplication is completed within m clock cycles as a GF(3^m) multiplication. On the other hand, a significant number of addition circuitry is needed that consume around 5200 slices for m = 97. In order to explore reduction strategies, two implementations are developed:

i) All the blocks are parallel
ii) Limited number of Adders after the multipliers

In the latter approach the number of adders is limited after the multipliers to four and the operations are scheduled. This approach increases the number of clock cycles by five (2.5% of all operations), but significantly reduces the amount of space consumed by adders. Similarly, scheduling approach tried to be used to decrease the number of multipliers. However, scheduling has not given successful results on FPGA implementation. This approach also increased the number of slices 5% approximately. The main reason for this increase in the hardware is the need for the multiplexers that are used to select the correct input to the adders at the correct clock cycle. In addition to this, wiring also gets harder with more connections to the same blocks.

The scheduling approach for ASIC implementations is left as the future work since it may save chip space in ASIC. Finally, for additions and subtractions the same adder block is used by just rewiring the inputs to swap the bits of the subtrahend since it negates the GF(3) elements in the employed representation.

4.2. GF(3^{6m}) Cubing Block

The second GF(3^{6m}) block is for performing cubing operation and as in the case of the multiplier, it is constructed using arithmetic units of the base field GF(3^{2m}) as proposed in [25]. As shown in Figure 4.3, GF(3^{6m}) cubing circuitry includes three adder/subtracter and three cubing blocks in GF(3^{2m}), while GF(3^{2m}) cubing circuit includes two GF(3^m) cubing circuit without any additional overhead but negation, as illustrated in Figure 4.4. Recall that $(a_2z^2 + a_1z + a_0)^3 = (a_2^3z^6 + a_1^3z^3 + a_0^3)$ and $z^3 = z + 1$ and $z^6 = z^2 - z - 1$. Thanks to the efficient GF(3^m) cubing blocks, implementing GF(3^{6m}) cubing block with parallel blocks does not consume much area and allows to finish the operation in one clock cycle. Another important aspect of GF(3^{6m}) cubing block is its register free design, which decreases the area consumed significantly.

In the accelerator, this parallel hardware architecture is used and optimized in terms of area and speed especially working on sub blocks. Cubing and multiplication units are optimized for specific irreducible polynomials used in the construction of ternary extension fields reducing the total area significantly. Additionally, an optimum algorithm and architecture is tried to be found to design a suitable Tate pairing accelerator for relatively constrained settings.

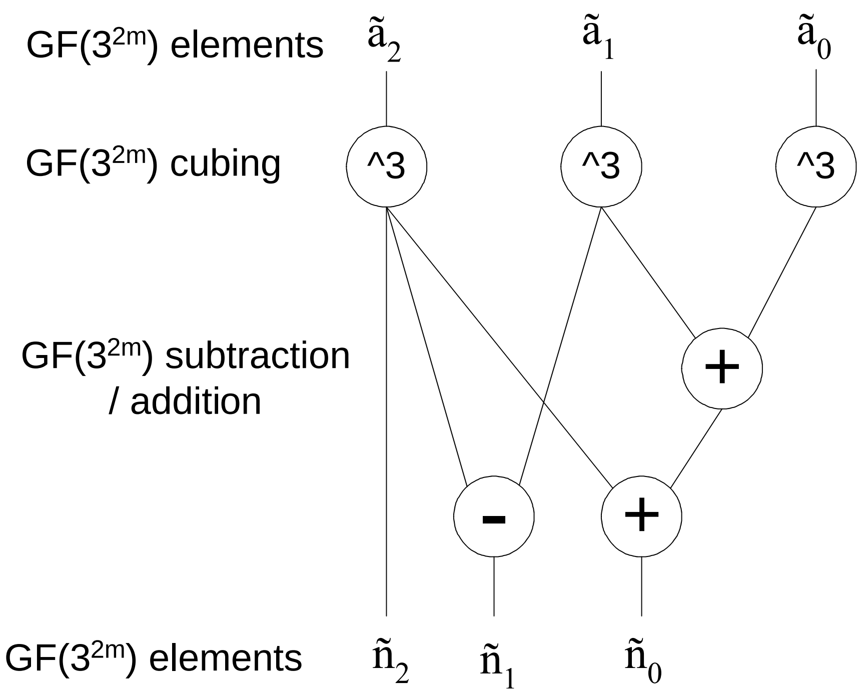

Figure 4.3. GF(3^{6m}) cubing unit from [25]

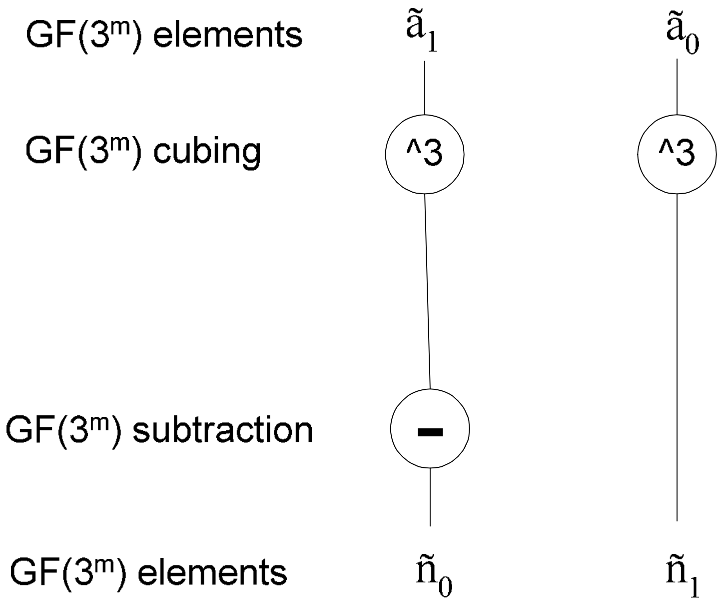

Figure 4.4. GF(3^{2m}) cubing unit from [25]

4.3. The Accelerator Architecture

After building the efficient blocks that are needed for the accelerator, a control unit and a datapath for the Tate Pairing operation is designed. The operation may be divided into two big phases as initialization and loop. In Table 4.1 operations are described in detail.

In the initialization phase, four GF(3^m) elements are input into the accelerator. For this part we use $2m$-bit long bus structure and connect it to all four related registers. With address selection and write signals, input data are written into the accelerator in four clock cycles. Cubing operations in steps 3 and 4 also take place during the initialization. The inputs that will be cubed will pass from the cubing circuitry and will be written to the related register. Since our cubing block is purely combinational, no extra clock cycles are used at these steps. The length of the databus can be adjusted depending on place-and-route and timing issues.

When the initialization is completed, accelerator starts operating in a loop. The control unit is composed of mainly two counters. First counter counts the loop's execution number to end the operation when completed. Second counter determines which step to be executed.

Table 4.1. Explanations for the number of clock cycles required for modified Duursma-Lee algorithm

	Step	Operation	Clock Cycles	total cycle for m = 97
initialization	1	$\alpha = x_p$	1	1
initialization	2	$ß = y_p$	1	1
initialization	3	$x = x_r^3$	1	1
initialization	4	$y = y_r^3$	1	1
Loop	5	$\alpha = \alpha^3$, $ß = ß^3$	1	97
Loop	6	$\alpha = \alpha^3$, $ß = ß^3$	1	97
Loop	7	$u = \alpha + x + d$	1	97
Loop	8	$\Upsilon = (-\mu^2)\zeta^0 + (-\beta y)\zeta^1 + (-\mu)\zeta^2 + (-1)\zeta^4$	97	97*97
Loop	9	$t = t^3$	1	97
Loop	10	$t = t*\Upsilon$, $y=-y$, $d=d-1 \bmod 3$	97	97*97
				19210

For the entire operation, only one $GF(3^{6m})$ multiplier is used for step 10, one $GF(3^{6m})$ cubing circuit for step 9, two $GF(3^m)$ cubing circuits for steps 5 and 6, two $GF(3^m)$ multipliers for step 8 and a number of adders. Each block starts working according to the counter 2. The operations that do not depend on each others' outputs are also overlapped to reduce the number of clock cycles. For instance, in step 10, three operations, $t = t*\Upsilon$, $y=-y$, $d=d-1 \bmod 3$, are done in the same clock cycle. The main advantage of the accelerator is that most of the operations are completed in single clock cycle. If the adder and cubing circuits were implemented with registers, clock count would increase around by 400 and registers would increase the area of the accelerator.

As seen from the table 4.1 the total clock cycle of one execution of Tate pairing is 19210 when the inputs are written in four clock cycles in total. Here it should be seen that 98% of the total execution time comes from two multiplication operations one in GF(3^{6m}) and one in GF(3^{m}). This mainly depends on the one clock cycle implementation of Cubing Circuit and the choice of serial LSE Multiplier. According to the requirements of the application by changing the serial multipliers with digit multipliers, the clock count may be reduced significantly with an area overhead.

5. IMPLEMENTATION ASPECTS

5.1. FPGA Implementation

In this work, behavioral simulations are done with Modelsim. Then, blocks and the whole accelerator have been mapped to Xilinx Virtex2Pro100 device using Xilinx 8.1 to make the comparison easier, since the previous works on the subject used the same device. The architecture is shown in Figure 5.1. As seen in Figure 5.1, the architecture is composed of 2 GF(3^m) Adders, 2 GF(3^m) Cubing Circuits, 2 GF(3^m) Multiplier Circuits, 1 GF(3^{6m}) Cubing Circuit, 1 GF(3^{6m}) Multiplier Circuit, Control Unit and a Register File that the inputs, intermediate results and the result is stored.

Two different versions of the hardware pairing accelerator are synthesized as seen from Table 5.1. First accelerator uses the GF(3^m) multiplier with fixed reduction polynomial of $p(x) = x^{97} + x^{16} + 2$. It occupies 14267 slices (32% of device) with an operating frequency of 77 MHz. In this case, total calculation time is about 251 μs. The second version of the accelerator is implemented with generic GF(3^m) multiplier. It occupies 16955 slices (38% of device) with an operating frequency of 69 MHz. In this case, total calculation time is around 278 μs.

The first implementation is 16% better in terms of area, 11% better in terms of calculation time and 25% better in terms of area time product. The only advantage of the latter implementation for this cost is its multiplier unit's flexibility for more than one polynomial. As indicated in section 3.4, fixed multiplication unit is 30% better than the generic one in terms of area. This difference is the main reason of the improvement between the two accelerators.

The verification of subblocks are done via modelsim simulations and the test vectors are generated manulally.

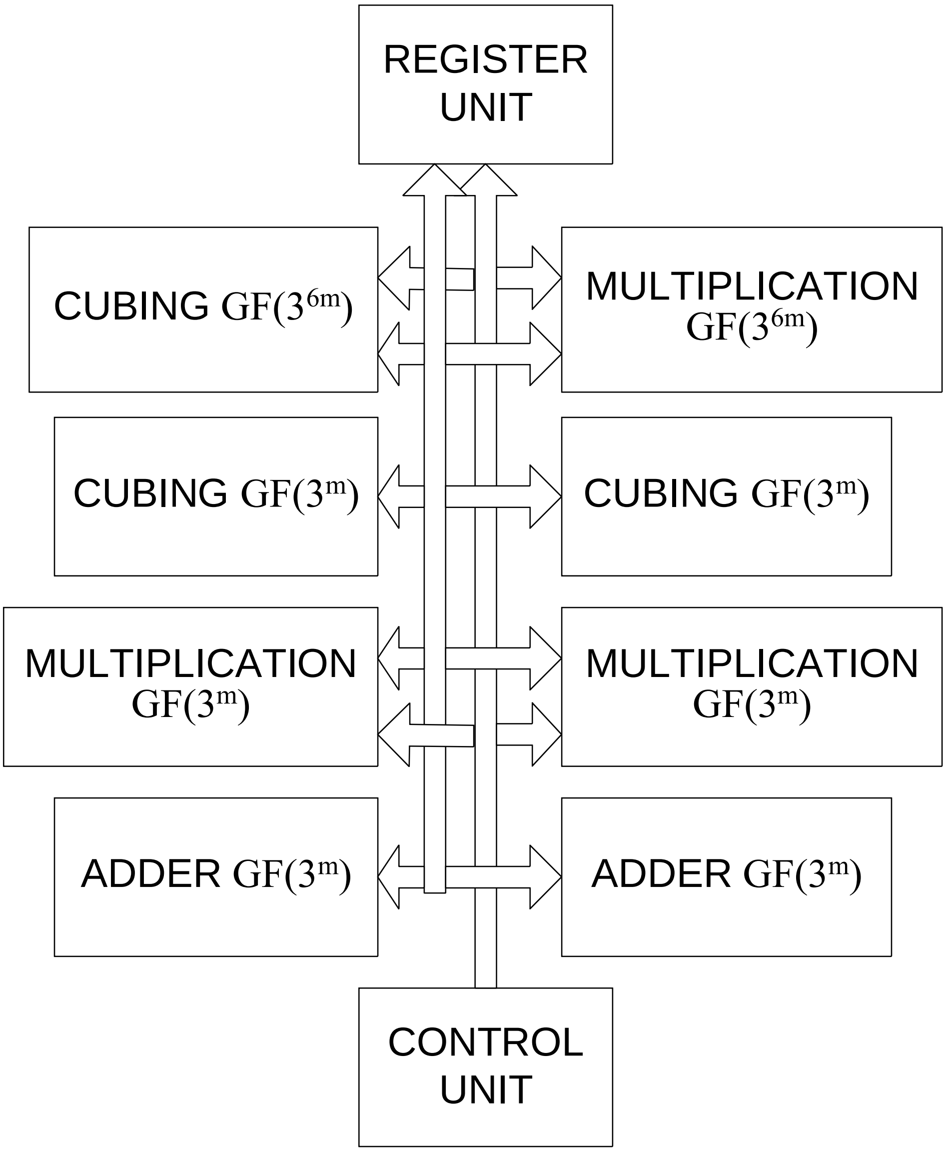

Figure 5.1. Tate Pairing accelerator architecture

As seen from Table 5.1, our implementation of Modified Duursma-Lee algorithm is almost three times (2.93) better than the previous implementation in the literature in terms of execution time and consumes nearly one-fourth of the estimated area in other implementations (namely [25]). In terms of area-time product, the Tate pairing accelerator with fixed multiplier is 12 times better than the one in [25] and the one with generic multiplier is 9 times better than the same implementation. In addition to these, hardware implementation shortens the calculation time nearly sixteen times compared to software implementation reported in [35].

Table 5.1. Comparison of the work with previous calculations of Modified Duursma-Lee Algorithm.

	Tate Pairing with fixed multiplier	Tate Pairing with generic multiplier	Tate pairing in [25]*	Tate Pairing in Software [37]
Slices	14267	16955	53406	
Op. Frequency	77,37 MHz	69,73 MHz	15 MHz	
Clock Count	19210	19210	12222	
Execution Time	250,72 μs	278,19 μs	0.815 ms	4,05 ms
Area*Time	1	1,33	12.2	

*Estimation by the authors of papers, not a complete implementation

5.2. ASIC Implementation

Vhdl codes of the algorithm are also synthesized for 0.25 μm CMOS technology using the Build Gates version 5.16 standard cell synthesizer. The total cell area is 4.3mm^2 excluding the buffers that are needed to satisfy the clock tree and static timing requirements, hold time and setup time. The constraints for this synthesis is 100 ns period, 15 maximum number of fanouts and 5 ns delay of external signals.

The implementation consumes around 10 mm^2 chip area after place and routing with 5 Metal technology using the Cadence's Place and Route tool First Encounter version 5.20. This tool also inserts the buffers for the clock tree and for satisfying the static timing requirements. Constraints for placement and routing are given as 100 ps maximum skew, 900 ps maximum clock delay and 700 ps minimum clock delay.

Area report generated by Build Gates after the synthesis is presented in Appendix B. In this report, cell area of the accelerator and sub-blocks are presented. As seen in the report the main part that consumes cell area is GF(3^{6m}) multiplier with 3.33 mm^2, 77 % of all area. Compared to GF(3^{6m}) multiplier, GF(3^{6m}) cubing circuit consumes much less area with 0.23mm^2, 5.3 % of all area.

ASIC implementation has reached the frequency of 78 MHz and completes the pairing in 250 µs. The computation time on Virtex-2 FPGA and the ASIC is nearly the same. We should note here that Virtex-2 devices are based on 90 nm CMOS technology with 9 metal routing layers. This shows that with better technology, ASIC implementation can become much more advantageous over FPGA implementation in terms of computation time, with a small chip area. The verification of the layout is done with the netlist generated after the synthesis with Build Gates.

The full layout of the Tate Pairing Accelerator is presented in Figure 5.2. As mentioned above the layout is routed with 5 metal layers. The fifth and forth metal layers are directly seen from the presented layout as the light blue and orange colors. In this picture the lower layers are not very obvious due to highly intense usage of metal layers. This property is a result of high wiring need of the blocks. Each bus between the blocks generally carries 97 elements which are connected by 194 wires. This high wiring affects the active area/total area ratio and lowers it to below 50%. A way of decreasing the area of the circuit may be decreasing the sizes of input and output buses down to 8 or 16 bits from 194 bits. This will make the wiring easier but the loading of the input parameters and the result will take 12 to 24 times more depending on the bus size. Since this portion of the calculation time is relatively short, this increase will not affect the performance significantly.

As seen from the Figure 5.2 the x/y proportion of the layout is 0.7 the reason for this proportion is 5 metal routing layers. The routing algorithm of First Encounter tool that we use for routing is based on selected layers for horizontal and vertical routing layers. Metal 1, Metal 3 and Metal 5 layers are used for vertical routing, Metal 2 and Metal 4 layers are used for horizontal routing. If the layout of the accelerator was designed on a square area horizontal routing would be problematic due to 2 metal layers rather than 3. This problem is overcome by changing the x/y ratio to 0.7, by this way horizontal and vertical routing is balanced.

In Figure 5.3 a portion of layout is presented. In this view, third metal layer is also seen obviously in green color. The areas close to the power lines are routed lightly compared to the inner parts as expected.

In Figure 5.4 a closer view of layout is presented. In this picture cells of the layout and power lines are seen. Red lines are routed in Metal 1 and white lines are routed in Metal 2.

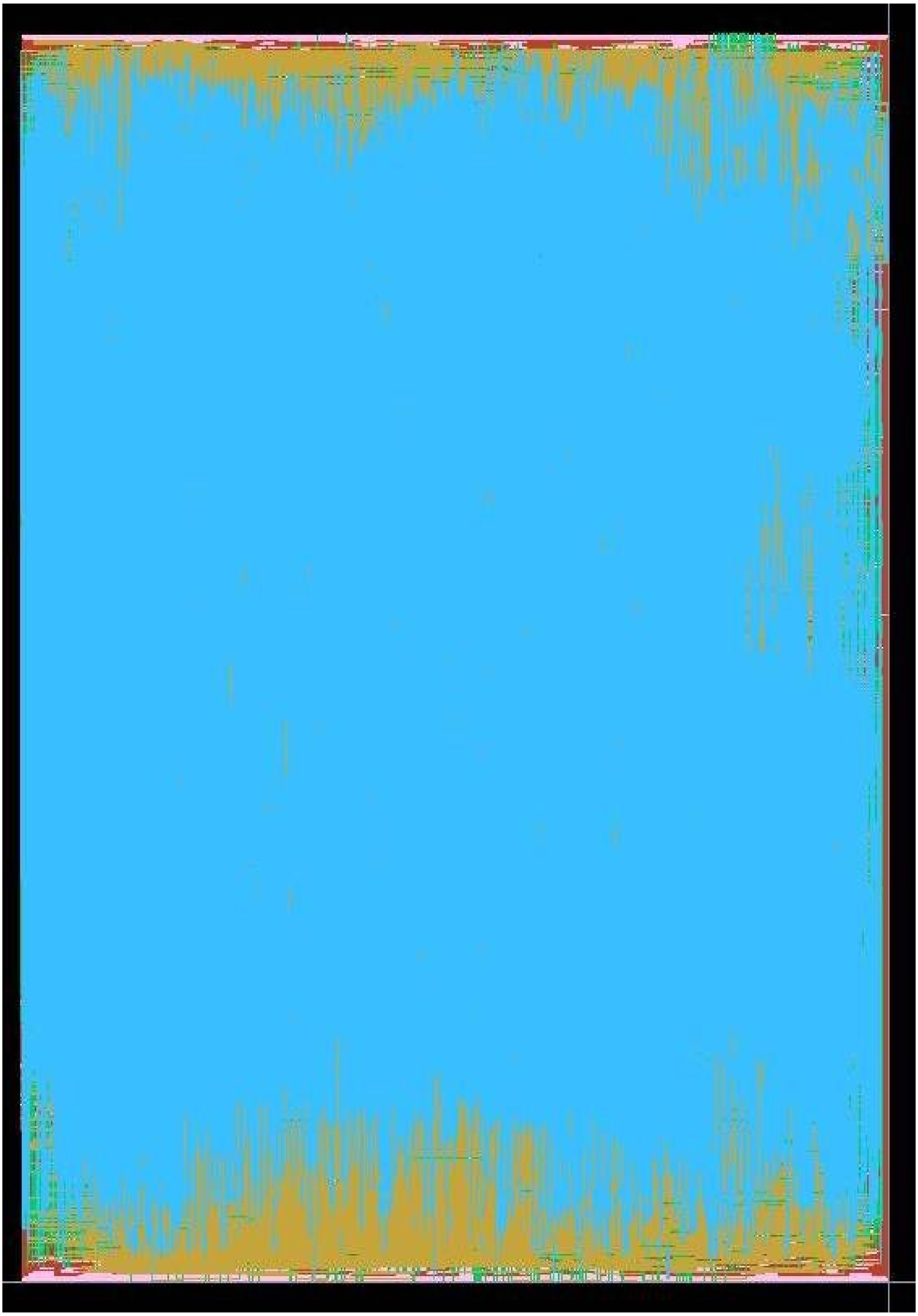

Figure 5.2. Full layout of Tate Pairing Accelerator

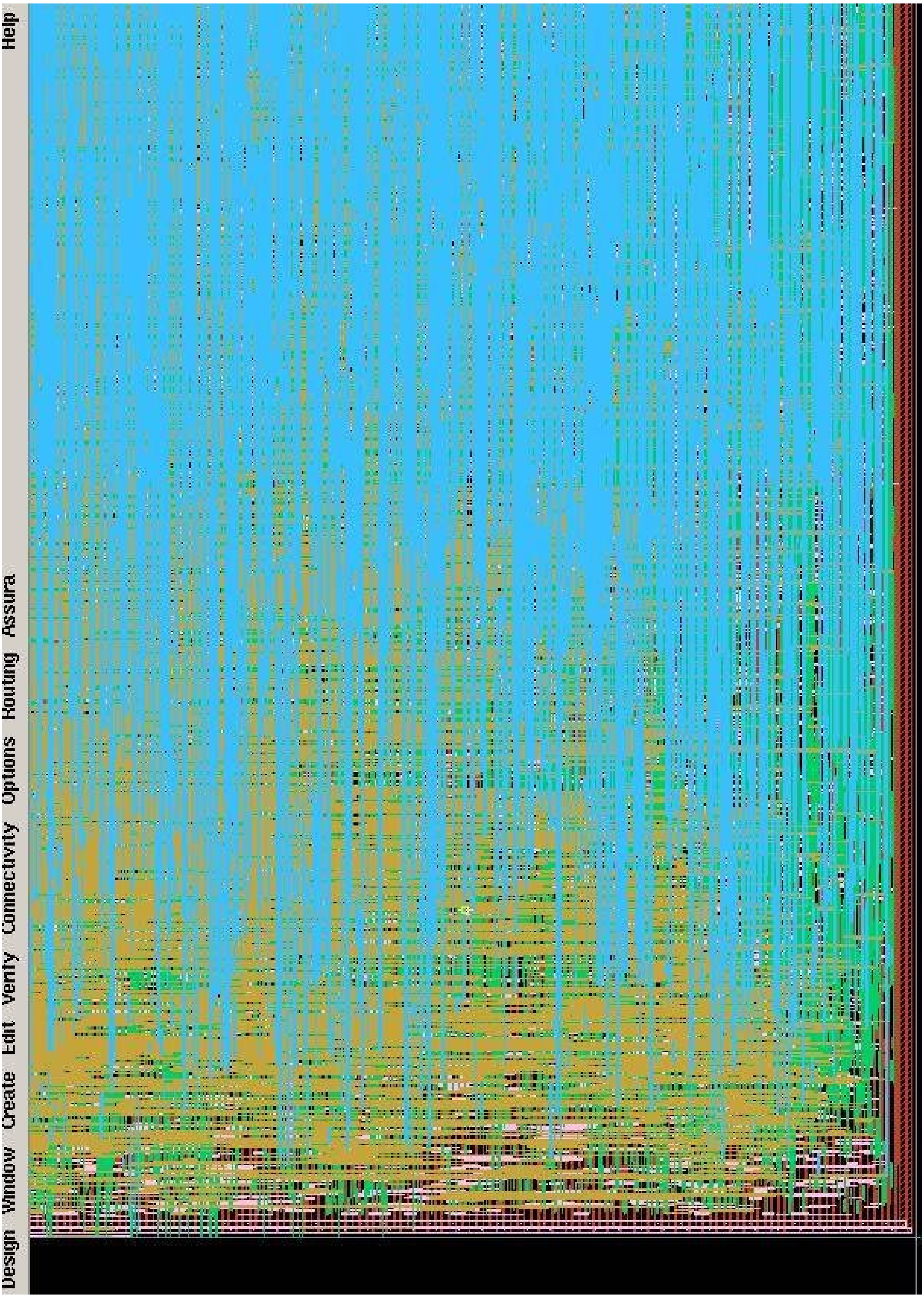

Figure 5.3. A portion of layout

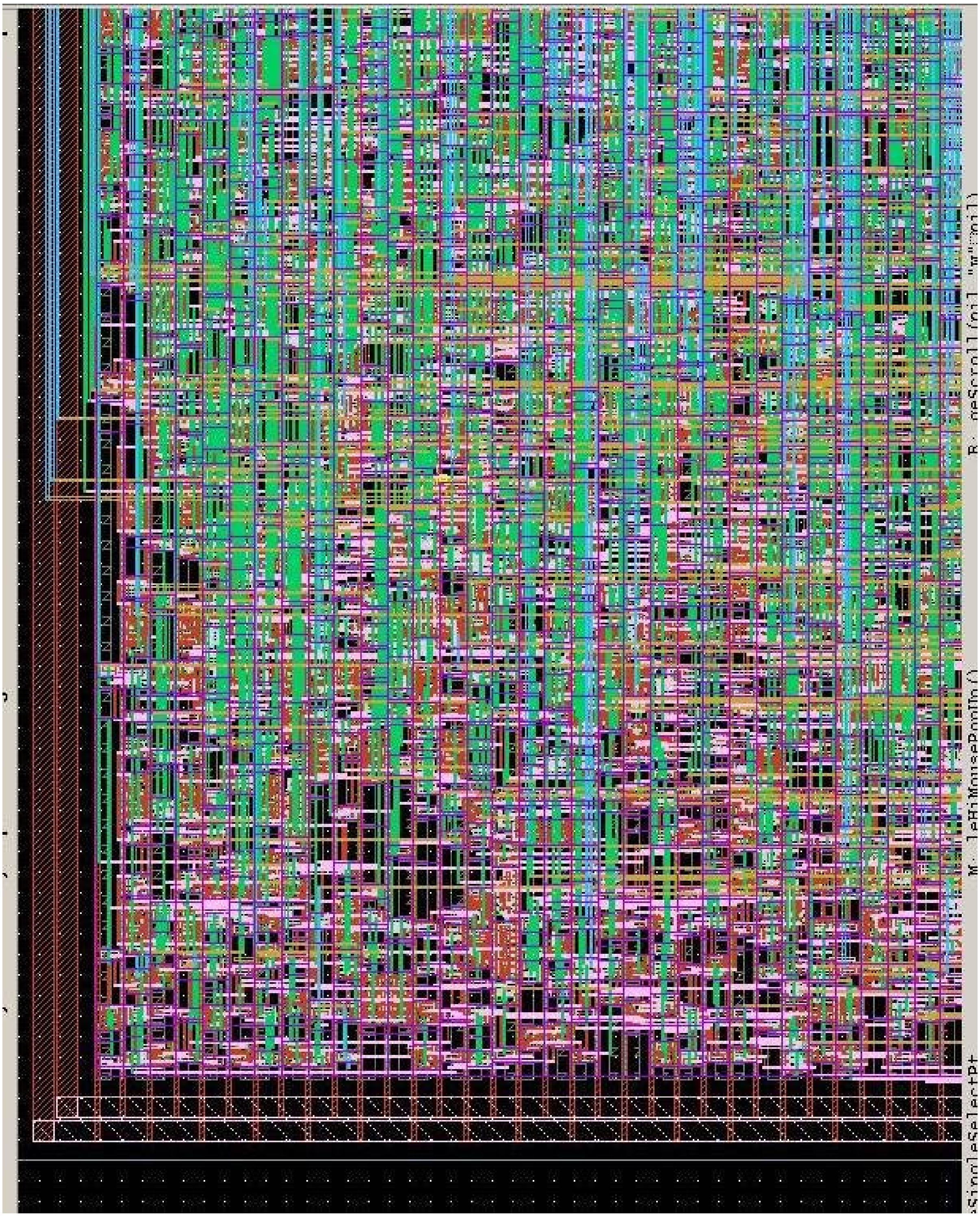

Figure 5.4. Closer view of layout

6. CONCLUSION

In this thesis, we have implemented an accelerator implementing the Tate Pairing in characteristic three based on the Modified Duursma Lee algorithm. We worked on the very hot field of cryptography, pairings, which allows the implementation of Identity Based Encryption. By using Identity Based Encryption, key distribution problem is resolved completely with its property of allowing any public string to be used as key. By this way, messages can be encrypted without the distribution of keys between the individuals. This is definitely very useful under situations where the authenticated key distribution is not feasible.

Our study is to mainly build on Elliptic Curve Cryptography which requires high computation power compared to commonly used algorithms such as DES and AES. Therefore, the performance and cost of our accelerator have great importance to make the usage of Tate Pairing feasible. Due to this, we have designed and optimized our sub-blocks in characteristic three to obtain a good performance.

Addition/Subtraction, Multiplication and Cubing blocks in $GF(3^m)$ are the required sub-blocks. Compared to the implementations in the literature significant improvements are achieved especially in the Multiplication and Cubing blocks. Then, we have built our architecture by using these sub-blocks.

With our parallel computation approach in $GF(3^{6m})$ multiplication and $GF(3^{6m})$ cubing block we have obtained a very fast and efficient implementation of algorithm compared to the previous works in the literature. The Tate Pairing calculation takes around 250µs in total with a Virtex-2 FPGA consuming a total of 14267 slices and working at 77,37 MHz. Lastly, our accelerator architecture is 12 times better than the known best implementation in the literature.

In addition to FPGA implementation, we have built the first ASIC implementation in the literature with 0.25µm CMOS technology. The total active area has been 4.3 mm^2

excluding the clock tree buffers. The layout area after the place and route with 5 metal layers has been 10 mm^2. with these results, the area of the circuit is suitable even for constrained devices such as smart cards.

APPENDIX A : AREA REPORT of ACCELERATOR from BUILD GATES

```
Summary Area Report
Source of Area : Timing Library
-------------------
+-----------------------------------------------------------------------
------------------+
|             Module |             Wireload |      Cell Area |    Net
Area |     Total Area |
|--------------------+----------------------+---------------+----------
--+---------------|
|                MDL | HHNEC25_Conservative | 4291561.43848 |
0.00000 | 4291561.43848 |
|    AWDP_INC_0133365 | HHNEC25_Conservative |    1841.66399 |
0.00000 |    1841.66399 |
|          AWDP_INC_1 | HHNEC25_Conservative |    1841.66399 |
0.00000 |    1841.66399 |
|       CUBIC_GF6_N97 | HHNEC25_Conservative |  234995.19873 |
0.00000 |  234995.19873 |
|         CUBIC_N97_0 | HHNEC25_Conservative |   20590.59189 |
0.00000 |   20590.59189 |
|         CUBIC_N97_1 | HHNEC25_Conservative |   20590.59189 |
0.00000 |   20590.59189 |
|      LSEMUL_GF6_N97 | HHNEC25_Conservative | 3335101.41304 |
0.00000 | 3335101.41304 |
|        LSEMUL_N97_0 | HHNEC25_Conservative |  127108.60661 |
0.00000 |  127108.60661 |
|        LSEMUL_N97_1 | HHNEC25_Conservative |  131073.53459 |
0.00000 |  131073.53459 |
|       TATE_ADDER_N1 | HHNEC25_Conservative |     185.85600 |
0.00000 |     185.85600 |
|    TATE_ADDER_N97_0 | HHNEC25_Conservative |   18028.03190 |
0.00000 |   18028.03190 |
|         CUBIC_N97_2 | HHNEC25_Conservative |   20691.96789 |
0.00000 |   20691.96789 |
|         CUBIC_N97_3 | HHNEC25_Conservative |   21215.74389 |
0.00000 |   21215.74389 |
|         CUBIC_N97_4 | HHNEC25_Conservative |   21215.74389 |
0.00000 |   21215.74389 |
|         CUBIC_N97_5 | HHNEC25_Conservative |   21215.74389 |
0.00000 |   21215.74389 |
|         CUBIC_N97_6 | HHNEC25_Conservative |   21243.90388 |
0.00000 |   21243.90388 |
|         CUBIC_N97_7 | HHNEC25_Conservative |   21243.90388 |
0.00000 |   21243.90388 |
|   TATE_ADDER_N194_0 | HHNEC25_Conservative |   36056.06380 |
0.00000 |   36056.06380 |
|   TATE_ADDER_N194_1 | HHNEC25_Conservative |   36056.06380 |
0.00000 |   36056.06380 |
```

REFERENCES

1. Shannon, C. E., “Communication Theory of Secrecy Systems”, *A Mathematical Theory of Cryptography*, 1949

2. Yazkurt. U., *Design and FPGA Implementation of an STM-1 Transceiver System Contataining the AES Algorithm*, M.S. Thesis, Boğaziçi University, 1991.

3. Barker, E., W. Barker, W. Burr, W. Polk, M. Smid, “Recommendation for Key Management”, NIST Special Publication 800-57, 2006.

4. Burrows, J. H., “Secure Hash Standard”, *Federal Information Processing Standards Publication 180-1*, April 1995.

5. Burrows, J. H., “Secure Hash Standard”, *Federal Information Processing Standards Publication 180*-2, August 2002.

6. Schneier, B., *Applied Cryptography*, John Wiley & Sons, 1996.

7. Burgess, J., E. Pattison, M. Goksel, “The History of Cryptography”, http://cse.stanford.edu/class/sophomore-college/projects-97/cryptography/history.html, 1997.

8. Kuhn, R. D., V. C. Hu, W. T. Polk and S. Chang, *Introduction to Public Key Technology and the Federal PKI Infrastructure,* Special Publication 800-32, February 2001.

9. Burrows, J. H., “Data Encryption Standard”, *Federal Information Processing*

10. "Advanced Encryption Standard", *Federal Information Processing Standards Publication 197*, November 2001.

11. Rivest, R.L., A. Shamir and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, v. 21, n. 2, pp. 120-126, 1978.

12. Koblitz, N., "Course In Number Theory and Cryptography", *Springer-Verlag* New York Inc.; 208p, 1987.

13. Miller, V. S., "Use of Elliptic Curves in Cryptography", *Advances in Cryptology, CRYPTO '85 Proceedings*, Volume 218/1986, 1985.

14. *The Case for Elliptic Curve Cryptography,* http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm

15. Shamir, A., "Identity-Based Cryptosystems and Signature Schemes", *In Advances in Cryptology (CRYPTO)*, Springer-Verlag LNCS 196, 47–53, 1985.

16. Boneh, D and M. Franklin, "Identity-Based Encryption from the Weil Pairing", *In SIAM Journal on Computing*, 32(3), 586–615, 2003.

17. Sakai, R., K. Ohgishi and M. Kasahara, "Cryptosystems Based on Pairings", *Symposium on Cryptography and Information Security (SCIS)*, 2000.

18. Shu, C., S. Kwon, K. Gaj, "FPGA Accelerated Tate Pairing Based Cryptosystem over Binary Fields", *IEEE International Conference on Field Programmable Technology*, 2006

19. Bertoni, G., L. Breveglieri, P. Fragneto, G. Pelosi, "Parallel Hardware Architectures for the Cryptographic Tate Pairing", *Third International Conference on Information Technology*, IEEE, 2006.

20. Frey, G., M. Muller and H. Rück, “The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems”, *IEEE Trans. Inform. Theory*, 1999.

21. Miller, V. S., “Short Programs for functions on curves”, Unpublished manuscript, http://crypto.stanford.edu/miller/miller.pdf, 1986.

22. Kwon, S., “Efficient Tate pairing computation for supersingular elliptic curves over binary fields”, *Cryptology ePrint Archive*, Report 2004/303, 2004, http://eprint.iacr.org/2004/303.

23. Barile, M., "Affine Coordinates.", *From MathWorld--A Wolfram Web Resource*, created by Eric W. Weisstein, http://mathworld.wolfram.com/AffineCoordinates.html.

24. Frey, G., and H. Rück, “A remark considering m-divisibility in the divisor class group of curves”, *Mathematics of Computation*, 62:865-874, 1994.

25. Kerins, T., W. P. Marnane, E. M. Popovici and P.S.L.M. Barreto, “Efficient Hardware for the Tate Pairing Calculation in Characteristic Three”, *International Association for Cryptologic Research*, CHES 2005, LNCS 3659, pp. 412–426, 2005.

26. Barreto, P. S.L.M., H.Y. Kim, B. Lynn, and M. Scott, “Efficient Algorithms for Pairing-Based Cryptosystems”, *In M. Yung, editor, Advances in Cryptology* CRYPTO 2002, volume LNCS 2442, pages 354–368, Springer-Verlag, 2002.

27. Galbraith, S., K. Harrison and D. Soldera, “Implementing the Tate pairing”, *In Algorithm Number Theory Symposium*, volume 2369 of Lecture Notes in Computer Science, pages 324-337, Springer-Verlag, 2002.

28. Barreto, P. S. L. M., H.Y. Kim, B. Lynn and M. Scott, "Efficient implementation of pairing based cryptosystems", *In Advances in Cryptology - CRYPTO 2002*, volume 2442 of Lecture Notes in Computer Science, pages 354-368, Springer-Verlag, 2002.

29. Duursma, I., and H.-S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x_p - x + d$", *In Advances in Cryptology - Asiacrypt 2003*, volume 2894 of Lecture Notes in Computer Science, pages 111-123, Springer-Verlag, 2003.

30. Kömürcü, G., E. Savaş, "An efficient hardware implementation of the Tate pairing in characteristic three" to be presented in ICONS'08.

31. Grabher, P and D. Page, "Hardware Acceleration of the Tate Pairing in Characteristic Three", *International Association for Cryptologic Research*, CHES 2005, LNCS 3659, pp. 398–411, 2005.

32. Page, D., and N.P. Smart, "Hardware implementation of finite fields of characteristic three", In B. S. Kaliski, Jr., C¸ . K. Koc, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems*, CHES 2002, volume LNCS. Springer-Verlag, 2002. 159, 160, 161, 170, 172.

33. Kerins, T., E. Popovici, and W. Marnane, "Algorithms and Architectures for Use in FPGA Implementations of Identity Based Encryption Schemes", *FPL 2004*, LNCS 3203, pp. 74–83, 2004, Springer-Verlag Berlin Heidelberg, 2004.

34. Bertoni, G., J. Guajardo, S. Kumar, G. Orlando, C. Paar, and T. Wollinger, "Efficient GF(p^m) Arithmetic Architectures for Cryptographic Applications", *CT-RSA 2003*, LNCS 2612, pp. 158–175, 2003, Springer-Verlag Berlin Heidelberg, 2003.

35. Barreto, P. S. L. M., "A note on efficient computation of cube roots in characteristic 3", *Cryptology ePrint Archieve*, Report 035/2004, http://eprint.iacr.org/2004/375, 2004.

36. Ronan, R., C. Eigeartaigh, C. Murphy, T. Kerins and M. Barreto "Hardware implementation of the ηT pairing in characteristic 3".

37. Xilinx Inc. Virtex-2 Platform FPGAs, *Complete Data Sheet*, Ds031 http://www.xilinx.com/bvdocs/publications/ds031.pdf, 2004.