FPGA BASED CRYPTOGRAPHY COMPUTATION PLATFORM AND THE BASIS CONVERSION IN COMPOSITE FINITE FIELDS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUHAMMAD RIAZ SIAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

SEPTEMBER 2013

Approval of the thesis:

FPGA BASED CRYPTOGRAPHY COMPUTATION PLATFORM AND THE BASIS CONVERSION IN COMPOSITE FINITE FIELDS

submitted by **MUHAMMAD RIAZ SIAL** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Bülent Karasözen
Director, Graduate School of **Applied Mathematics** _____

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography** _____

Prof. Dr. Ersan Akyıldız
Supervisor, **Cryptography, METU** _____

Examining Committee Members:

Prof. Dr. Ferruh Özbudak
Cryptography, METU _____

Prof. Dr. Ersan Akyıldız
Cryptography, METU _____

Assoc.Prof.Dr. Ali Doğanaksoy
Cryptography, METU _____

Dr. Hamdi Murat Yıldırım
CTIS, Bilkent University _____

Dr. Oğuz Yayla
Cryptography, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MUHAMMAD RIAZ SIAL

Signature :

ABSTRACT

FPGA BASED CRYPTOGRAPHY COMPUTATION PLATFORM AND THE BASIS CONVERSION IN COMPOSITE FINITE FIELDS

Sial, Muhammad Riaz
Ph.D, Department of Cryptography
Supervisor : Prof. Dr. Ersan Akyıldız

September 2013, 42 pages

In the study of this thesis work we focused on the hardware based cryptographic algorithms computation platform, especially for elliptic-curve and hyper-elliptic curve based protocols. We worked for making the hyperelliptic curve based Tate Pairing computation efficient specially for hardware implementations. To achieve this one needs to make the underlying finite field arithmetic implementations efficient. For this we study the finite fields of type $\mathbb{F}_q, q = p^{2pn}$ from the efficient implementation point of view. We found that we can represent these fields with irreducible polynomials in the form $f(x) = x^p - x - a$ over $\mathbb{F}_{p^{2pn}}$. By using this representation we have found a way of constructing normal basis for the field, together with transmission matrix between normal basis and Polynomial Basis of $\mathbb{F}_q$ and vice versa. The key point is that this matrix and its inverse can be computed very efficiently without any memory requirement. Then we imply the techniques developed in this work on the Tate pairing computation algorithm proposed by I.Duursma, H.S.Lee in [20] and modified by S.Kwon [27] and hardware implementation scheme proposed in [2]. We found that by introduction of such efficient conversion of basis we can significantly reduce the pairing computation cost as well as the cost of other algorithms based on such composite finite field structures. In short we give a new efficient way of conversion from polynomial to normal basis and vice versa with zero memory complexity in the finite fields of type $\mathbb{F}_q, q = p^{2pn}$ for any prime and we reduce the Tate pairing computation cost

by 49.5% after applying such conversions.

Secondly as part of the FPGA based cryptography platform we have implemented cryptographic algorithms in FPGA, integrated it with other cores inside FPGA and accessories outside FPGA using Microblaze processor. We also give an efficient implementation of prime field multiplication for p = 7, which is 31% faster than the one in [2] and by using this multiplier Tate-pairing algorithm in [2] can be made 17% more efficient. We also implemented modular multiplication, addition, inversion and efficient squaring modules over binary fields needed to implement protocols based on NIST recommended elliptic curve K-163 and SHA-1 to use it for ECDSA and AES-128 for reference purpose to run over existing FPGA platform.

Keywords: Composite Finite Fields, basis conversion, Vandermonde matrix, Tate pairing, cryptography, FPGA

ÖZ

FPGA TABANLı KRIPTOGRAFI İŞLEM PLATFORMU VE BILEŞIK SONLU CISIMLERDE BAZ DÖNÜŞÜMÜ

Sial, Muhammad Riaz

Doktora, Kriptografi

Tez Yöneticisi : Prof. Dr. Ersan Akyıldız

Eylul 2013, 42 sayfa

Bu tezde eliptik ve hipereliptic eğri tabanlı protokoller başta olmak üzere donanım tabanlı kriptografik algoritmaların işlem platformlarına odaklandık. Hipereliptik eğri tabanlı Tate Pairing işlemlerinin özellikle donanımsal uygulamalarını verimli hale getirmeye Çalıştık. Bunun için $\mathbb{F}_q, q = p^{2pn}$ şeklindeki sonlu cisimlerini verimli uygulamaya çalıştık. Bu cisimleri $f(x) = x^p - x - a$ formundaki indirgenemez polinomlarla temsil edebileceğimizi gözlemledik. Bu temsil şeklini kullanarak cismin normal bazını oluşturmak, normal bazdan $\mathbb{F}_q$'ın polinom bazına geçiş matrisini ve de tersini oluştumak için bir yol bulduk. Burada önemli nokta bu matrisi ve tersini hiç bellek kullanmadan hızlı bir şekilde elde edilebiliyor olmasıdır. Daha sonra bu çalışmada geliştirdiğimiz teknikleri I.Duursma, H.S.Lee tarafından [20]'da önerilen algoritmaya, S.Kwon'un [27]'da değiştirdiği algoritmaya ve [2]'de önerilen donanım uygulaması şemasına uyguladık. Bu hızlı şekildeki baz değişimini kullanarak pairing işlem maliyetlerini azalttığımız gibi, bileşik sonlu cisim yapıları tabanlı diğer algoritmaların da maliyetlerini de önemli bir boyutta azaltabileceğimizi gördük. Kısacası, herhangi bir asal için $\mathbb{F}_q, q = p^{2pn}$ türündeki sonlu cisimlerde polinom bazından normal baza ve tersine geçişi hiç bellek kullanmadan hızlı bir şekilde yapmak için yeni bir yol gösterdik ve bu dönüşümleri yaparak Tate pairing işlem maliyetini %49.5 kadar azalttık.

İkinci olarak FPGA tabanlı kriptografinin bir parçası olarak kriptografik algoritmaları FPGA'lerde uyguladık, FPGA'lerin içindeki diğer çekirdeklerle ve Microb-

laze işlemci kullanarak FPGA dışarsındaki ek birimlerle bütünleştirdik. Ayrıca p=7 için [2]'dekinden %31 daha hızlı asal cisim çarpımını uyguladık ve bu çarpımı kullanarak [2]'deki Tate pairing algoritmasını %17 daha hızlı hale getirdik. Son olarak, NIST'in ECDSA ve AES-128'de kullanılmak üzere tavsiye ettiği K-163 eliptik eğrisi ve SHA-1 protokollerinde kullanılan ikili cisimler üzerinde modüler çarpma, toplama, tersini alma ve hızlı kare alma işlemlerini de referans olması için var olan FPGA platformunda uyguladık.

Anahtar Kelimeler: Kompozit Finite Field, temel dönüşüm, Vandermonde matris, Tate pairing, kriptografi, FPGA

To my family specialy my loving Mother (may her soul rest in peace) who suffered the most due to my absence in the last days of her life

ACKNOWLEDGMENTS

I thank ALLAH for the wisdom and perseverance that he has been bestowed upon me during my Ph.D research, and indeed throughout my life: "I can do everything through him who give me strength".

My first debt of gratitude and respect must go to my thesis supervisor Prof. Dr. Ersan Akyıldız for his patient guidance, enthusiastic encouragement and valuable advices whilst allowing me the room to work in my own way during the research and preparation of this thesis. I also thank Dr. Hamdi murat yıldırım for his valuable tips and guidance during this time.

I want to express my deeply felt thanks to Prof. Dr Feruh Özbudak for his patience to listen, warm encouragement and thoughtful guidance in the times of depression during my research in the cryptography lab in odd times.

My special thanks to Prof. Dr. Ali Doğanksoy my first teacher in the field of cryptography for his motivating and cheerful attitude at the start of my study in the field of cryptography.

I cant forget the help provided by Dr. Sedat Akleylek and Dr. Oğuz Yayla for helping in day to day problems faced during my study and their prompt response in case of any guidance.

Last but not the least I thank my family for their patience to bear me and thank my cute small angel HUDA RIAZ for her day and night prayers for her baba.

It has been a great privilege to spend the most memorable and demanding 06 years of my life in such a meritorious Institution like "Institute of Applied Mathematics" at Middle East Technical University Ankara, and its members will always remain dear to me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

There are primarily two parts of this thesis work. The first part deals with theoretical research over the efficient finite fields basis conversion and its application to Tate pairing and the second part is the Embedded FPGA based system Implementations.

Motivation of the first part started from [2] and [3], in which authors have proposed efficient ways to compute pairing-based cryptographic protocols on hyperelliptic curves of genus 3 over finite fields of characteristics 7 and 3 respectively. Both authors used the same composite structure of finite fields of type $F_{p^{2pn}}$ for Tate-pairing computation. In [2] it is shown that using this structure as the value of prime p grows the complexity of the algorithm is reduced. Specially using $p = 7$ is more efficient as compared to $p = 3$.

It is well known in the finite field arithmetic literature that multiplication in polynomial basis is much faster than in normal basis. On the other hand squaring in binary fields or exponentiation is much cheaper in the normal basis instead of polynomial basis as raising to the power p is just a shift operation. Normal basis are considered very attractive specially for hardware implementations due to the fact that shift operation is free to implement in hardware. To overcome the multiplication difficulty in normal basis many solutions are proposed to make it faster like Optimal Normal Basis(ONB) of type I and II have been introduced in many publications e.g [12] and [13]. So researchers had been always looking for the optimization of the algorithms using optimal choice of basis and algorithm. Conversion from one basis to other is a costly operation using matrix multiplication, generally requires $O(n^2)$ operations and $O(n^2)$ field elements storage [11]. Its hard specially for memory constrained devices, as one needs to store an (n-by-n) matrix for this operation.

In past there have been suggested solutions for this sake. In [12] Muchtadi-Alamsyah and F. Yuliawan focused on the basis conversion from polynomial to Optimal normal basis(ONB) and vice versa, in the finite fields of type $F_{2^{nm}}$, where $gcd(n, m) = 1$, $m+1$ must be prime and 2 must be primitive in Z^*_{m+1} or for ONB of type II, $2m + 1$ must be prime and 2 is a primitive in Z_{2m+1}, or $2m + 1 \equiv 3$ mod 4 and 2 generates the quadratic residues in Z_{2m+1}. They suggested a way to covert basis with $O(m)$ memory complexity and $O(m)$ time complexity.

In [10] Authors have suggested that it is more efficient to work in composite fields instead of binary fields and have devised algorithm to construct composite field of type $F_{2^{nm}}$ from F_{2^k} where $k = mn$. In [9] authors have suggested algorithms to convert basis in the finite field of type F_{2^m} with the $O(m)$ memory complexity and $O(m)$ time complexity.

In the literature researchers have also used the hybrid approach for basis to benefit from both basis advantages. One such example is from v.z.Gathen and Shokrollahi [8]. In this paper authors used hybrid basis system, say when multiplication is needed, two field elements are converted to polynomial basis and multiplication is carried out in polynomial basis, then the reduction is carried out in normal basis after converting back from polynomial basis. In [6] authors have demonstrated that using the hybrid type of basis representation for implementing elliptic curve protocol outperforms all existing FPGA set-up to break ECC-130 challenge.

In this work we focused on the efficient basis conversion between polynomial and normal basis to expedite the computation of such algorithms based on composite finite fields. Here we noticed that due to Frobenius in F_{q^p} over F_q the computing exponentiation can be much faster than the other methods when p is an odd prime. Moreover same can be extended to p^{2pn} type of field for all n co-prime to p. During this work luckily the basis conversion matrix happened to be a special type of Vandermonde matrix which can be constructed very efficiently. All the computations in matrix format are in the prime characteristic field which we show is very easy to compute as compared to parent field.

Hence we purpose a storage free basis conversion for the composite finite fields of type $F_{p^{2pn}}$ over $F_{p^{2n}}$ which efficiently computes the presentation of an element from one basis to other with little extra computation cost. We also analyse the Tate pairing algorithm described in [2] and give an estimate for the hardware implementation improvement for the Tate pairing computation using our approach.

Arrangement of this thesis is as below.
In chapter-1 we will discuss the preliminaries.
In chapter-2 we discuss the composite finite fields and basis conversions.
In chapter-3 we present the efficient way of Tate pairing computation.
In chapter-4 covers the FGPA implementations.
Where as the Appendix-A gives the guide for implementations and in Appendix-B we append the Verilog and C++ programming Code for FPGA implementations and embedded system .

1.1 PRELIMINARIES

In this part we will give an introduction to what are composite finite fields and what are the basis.

1.1.1 Finite Fields and their Presentation

In the study of abstract algebra, a finite field or Galois field is a field in which there are finite number of elements. Finite fields are of prime importance in number theory, cryptography, error correcting codes, algebraic geometry etc. Elements of the finite fields are presented using different basis system, a basis for $\mathbb{F}_{p^n}$ of size p^n elements over $\mathbb{F}_p$ is a set of n elements of $\mathbb{F}_{p^n}$ which are linearly independent. Once the type of basis is chosen then one needs to set the rules for field operations e.g multiplication, addition etc. There are mainly two types of basis known as polynomial and normal basis.

1.1.1.1 Polynomial Basis

If $\alpha \in \mathbb{F}_{p^n}$ is the root of an irreducible primitive polynomial over $\mathbb{F}_p$ then $\mathbb{F}_{p^n} = \mathbb{F}_p(\alpha)$ and $\{1, \alpha, \alpha^1, \alpha^2, ..., \alpha^{n-1}\}$ is a basis over $\mathbb{F}_p$ of the vector space of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ and is called polynomial basis (algebraic basis). An element $A \in \mathbb{F}_{p^n}$ can be written as

$$A = \sum_{i=0}^{n-1} a_i \alpha^i,$$

where $a_0, a_1, ..., a_{n-1} \in \mathbb{F}_p$ are the coefficients.

1.1.1.2 Normal Basis

Let $\mathbb{F}_{p^n}$ be the field extension of $\mathbb{F}_p$ then a basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ of the form $\{\beta, \beta^p, \beta^{p^2}, ..., \beta^{p^{n-1}}\}$, where β is a suitable element of $\mathbb{F}_{p^n}$ is called Normal basis of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$. Then an element $B \in \mathbb{F}_{p^n}$ can be written as

$$B = \sum_{i=0}^{n-1} b_i \beta^{p^i},$$

where $b_0, b_1, ..., b_{n-1} \in \mathbb{F}_p$ are the coefficients.

Both types of basis are very common in the theory of finite fields and cryptography. However the construction of polynomial basis is very easy as compared to normal basis. It is difficult to find normal basis however these are very popular due to their property that computing the exponentiation to power p is very easy. As we can see if we raise any element of the basis to p^{th} power the result is next

element of the basis. So when we do the same for for some B then its just the rotation of the elements b_i.

1.1.1.3 Composite Finite Fields

Composite Field can be defined as an extension field defined over a base field which is already an extension of some field. For instance if we want to construct a field of size p^{2pn}elements then there exist only one finite field of this size however its representations may vary depending upon the basis and structure of the field you choose. Now we may represent the same field using $\mathbb{F}_{p^m}$ structure, where $m = 2pn$ and the elements of the field can be represented as polynomials of degree $m-1$ whose coefficients belong to $\mathbb{F}_p$. The same field can be represented using composite field structure $\mathbb{F}_{p^{2pn}}$, then each element $A \in \mathbb{F}_{p^{2pn}}$ can be represented as polynomials of degree $p-1$ whose coefficients belong to $\mathbb{F}_{p^{2n}}$ provided p and n are co-prime. If $A \in \mathbb{F}_{P^{2pn}}$ then

$$A = \sum_{i=0}^{p-1} a_i \alpha^i,$$

where $a_0, a_1, ..., a_{n-1} \in \mathbb{F}_{p^{2n}}$ are the coefficients.

1.1.2 Basis Conversion

Basis conversion refers to change the presentation of a field element from one basis to other. For cost effective solutions, better performance and compatibility of different systems its necessary to convert from one type of basis presentation to other. In some situations where the raising power to a big number is more costly then hybrid basis approach is more appropriate. By hybrid we mean one uses the polynomial basis however when ever raising the power is needed converts to normal basis, power is computed and convert back to polynomial basis as multiplication operation is normally very costly in normal basis.

CHAPTER 2

BASIS CONVERSIONS OVER COMPOSITE FINITE FIELDS

In this chapter we will discuss the conversion from polynomial basis to normal basis and from normal basis to polynomial basis over the field $\mathbb{F}_{p^p}$. We present here all the theorems needed to prove the conversion matrix step by step.

Lemma 2.1. *$f(x) = x^p - x + 1$ is irreducible over any $\mathbb{F}_{p^n}$ where* $\gcd(p, n) = 1$.

Proof. In [1] theorem (3.78) says $f(x) = x^p - x - a$,$a \in \mathbb{F}_p^*$ is irreducible over $\mathbb{F}_q$ where $q = p^n$, if and only if it has no root $\in \mathbb{F}_q$. Now from [1] theorem (2.25) and [1] corollary (3.79) gives that $f(x) = x^p - x - a$ has a root in $\mathbb{F}_q$ if and only if absolute $Tr_{\mathbb{F}_q}(a) = 0$. Hence $f(x) = x^p - x - a$,$a \in \mathbb{F}_p^*$ is irreducible over $\mathbb{F}_q$ if $Tr_{\mathbb{F}_q}(a) \neq 0$. Since according to [1] theorem 2.23, abs $Tr_{\mathbb{F}_q}(a) = n * a$, so if $Tr_{\mathbb{F}_q}(a) = 0$ then either a is zero or n is divisible by p. It completes the proof. □

Lemma 2.2. *Let $\beta = \alpha^{-1}$, where α is the root of $f(x) = x^p - x + 1$ and $\mathbb{F}_{p^p} = \mathbb{F}_p[x]/ < f(x) = x^p - x + 1 >$ then β generates a normal basis namely $\{\beta, \beta^p, \beta^{p^2}, ..., \beta^{p^{p-1}}\}$ is a basis of $\mathbb{F}_{p^p}$ over $\mathbb{F}_p$.*

Proof. Since reciprocal of an irreducible polynomial is also irreducible so the reciprocal of $f(x) = x^p - x + 1$ (which is irreducible over prime p from lemma 2.1) is $x^p(f(x^{-1})) = x^p - x^{p-1} + 1$ is also irreducible. It follows from corollary (2.6) in [4] that Root of this polynomial generates normal basis. □

Theorem 2.3. *α be the root of irreducible polynomial $f(x) = x^p - x + 1$ over $\mathbb{F}_p$ and $\beta = \alpha^{-1}$ then $\beta^{p^i} = 1 - (\alpha - i)^{p-1}$, where $i = 0, 1, 2, \cdots, p - 1$.*

Proof. Since α is the root of $f(x) = x^p - x + 1$, we have

$\alpha^p = \alpha - 1$ and

$$\begin{aligned}
\beta &= f(\alpha) = a_0 + a_1\alpha + \cdots + a_k\alpha^{p-1} \\
&= \frac{1}{\alpha} = 1 - \alpha^{p-1} \\
\beta^p &= f(\alpha)^p = f(\alpha^p) = f(\alpha - 1) = 1 - (\alpha - 1)^{p-1} \\
\beta^{p^2} &= (\beta^p)^p = f(\alpha - 1)^p = f(\alpha^p - (1)^p) \\
&\quad \text{due to Frobenius} \\
&= f(\alpha - 2) = 1 - (\alpha - 2)^{p-1} \\
&\downarrow \\
&\downarrow \\
(\beta^{p^{i-1}})^p &= f(\alpha - (i-2))^p = f(\alpha^p - (i-2)) \\
&= f(\alpha - 1 - i + 2) = 1 - (\alpha - (i-1))^{p-1} \\
&\quad \text{Hence by induction} \\
\beta^{p^i} &= f(\alpha - (i-1))^p = f(\alpha - i) = 1 - (\alpha - i)^{p-1}
\end{aligned} \tag{2.1}$$

□

Lemma 2.4.

$$\binom{p-1}{k} \equiv \frac{(p-1)!}{k!(p-1-k)!} \equiv (-1)^k \mod p$$

where $k = 0, 1, 2, ..., p-1$.

Proof. Note that $(p-1)! = (-1) \mod p$ known as Wilson's Theorem.

$$
\begin{aligned}
\text{Let } S_k &\equiv (k!(p-1-k)!) \mod p, \\
&\quad \text{where } 0 \le k \le p-1 \\
S_0 &\equiv 0!(p-1)! \equiv (-1) \mod p \\
\text{since } S_{k+1} &= (k+1)!(p-1-k-1)! \\
&= \frac{(k+1)k!(p-1-k)!}{p-1-k} \text{ over Z then} \\
S_{k+1}(p-1-k) &\equiv S_k(k+1) \mod p \\
S_{k+1}(-1-k) &\equiv (-1)(S_k)(-k-1) \mod p \\
S_{k+1} &\equiv (-1)(S_k) \mod p \\
&\quad \text{From the above equality and} \\
\text{Since } S_0 &\equiv (-1) \mod p \\
&\quad \text{we can write} \\
S_k &\equiv (-1)^{k+1} \mod p \\
S_0 \frac{(p-1)!}{k!(p-1-k)!} &\equiv \frac{(p-1)!}{S_k} \mod p \\
&\equiv \frac{-1}{(-1)^{k+1}} \equiv (-1)^k \mod p
\end{aligned}
$$

□

Theorem 2.5. *The matrix*

$$
M = \begin{bmatrix}
-1 & 0 & \dots & 0 & 1 \\
-(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\
-(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
-(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0
\end{bmatrix}
$$

is transition matrix from polynomial basis $\{1, \alpha, \alpha^2, \cdots, \alpha^{p-1}\}$ *to normal basis* $\{\beta, \beta^p, \beta^{p^2}, \cdots, \beta^{p^{p-1}}\}$ *of the space* $\mathbb{F}_{p^p}$ *over* $\mathbb{F}_p$*, where* α *is the root of* $f(x) = x^p - x + 1$ *and* $\beta = \frac{1}{\alpha}$

Proof. Let

$$
\begin{aligned}
\mathbb{F}_{p^p} &= \mathbb{F}_p[x]/ < f(x) = x^p - x + 1 > \\
&= \mathbb{F}_p[\alpha]/ < f(\alpha) = \alpha^p - \alpha + 1 > \text{and let} \\
\beta &= \alpha^{-1}
\end{aligned}
\tag{2.2}
$$

From Theorem (2.3) we know that:

$$\beta^{p^i} = 1 - (\alpha - i)^{p-1}$$

where
$\beta^{p^i} for\ 0 \leq i \leq p-1 = \{\beta, \beta^p, \beta^{p^2}, \cdots, \beta^{p^{p-1}}\}$ are the normal basis.
From above equation using binomial expansion formulae we get

$$\begin{aligned} \beta^{p^i} &= 1 - (\alpha + (-i))^{p-1} \\ &= 1 - \sum_{k=0}^{p-1} \binom{p-1}{k} \alpha^{p-1-k}(-i)^k \end{aligned} \tag{2.3}$$

From Lemma (2.4) and above Eqn. we can write

$$\begin{aligned} (\alpha - i)^{p-1} &= \sum_{k=0}^{p-1} (-1)^k \alpha^{p-1-k}(-1)^k (i)^k \mod p \\ &= \sum_{k=0}^{p-1} (-1)^{2k} \alpha^{p-1-k} i^k \mod p \\ &= \sum_{k=0}^{p-1} \alpha^{(p-1-k)} i^k \mod p \end{aligned} \tag{2.4}$$

From above equations we can easily conclude that,

$\beta^{p^0} = 1 - \alpha^{p-1}$, as when $i = 0$, eqn:(2.4) $= \alpha^{p-1}$

$$\beta_i = \beta^{p^i} = -\sum_{k=0}^{p-2} \alpha^{(p-1-k)} i^k \mod p, \text{ for } 1 \leq i \leq p-1 \tag{2.5}$$

Now if we compute the above equation for all β_i in the tabular form we get our desired matrix and hence normal basis can be computed from polynomial basis as below:

$$\begin{bmatrix} \beta \\ \beta^p \\ \beta^{p^2} \\ \cdots \\ \beta^{p^{p-1}} \end{bmatrix} = \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 \\ -(1)^0 & -(1)^1 & \cdots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \cdots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \cdots & -(p-1)^{p-2} & 0 \end{bmatrix} * \begin{bmatrix} \alpha^{p-1} \\ \alpha^{p-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix}$$

As each row of the matrix is just raising the power of field element, so no need of memory to store the matrix. □

Lemma 2.6.

$$\sum_{m=0}^{p-2} k^m \mod p, k \in \mathbb{F}_{p^*} \equiv \begin{cases} -1 \mod p & \textit{if } k = 1, \\ 0 \mod p & \textit{otherwise.} \end{cases}$$

Proof. For k = 1 its straight forward sum of $p-1$ number of 1 = -1 mod p
For $k \neq 1$ lets have

$$\sum_{m=0}^{p-2} k^m \mod p \equiv k^0 + k^1 + k^2 + \cdots + k^{p-2} \mod p \tag{2.6}$$

Since this is a simple geometric series
$a + ar + ar^2 + ... + ar^{n-1}$, where
$a = 1$
$r = k$,
$n = p - 1$.
Then we have

$$\sum_{m=0}^{p-2} k^m \mod p \equiv \frac{a(1-r^{p-1})}{1-r} \mod p$$

Since $r^{p-1} \equiv 1 \mod p$, for $r = k, k \neq 1, k \in \mathbb{F}_{p^*}$ So

$$\sum_{m=0}^{p-2} k^m \mod p \equiv 0 \mod p$$

□

Theorem 2.7. *If the matrix*

$$M = \begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0 \end{bmatrix}$$

The inverse of the matrix M mod p *can be computed by simple transpose of the permuted rows of matrix M.*

Proof. If we analyse the matrix M of dim (p x p) contains a sub-matrix A of size (p-1) x (p-1) which is of special type known as Vandermonde matrix. This sub-matrix is shown below.

$$A = \begin{bmatrix} -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} \end{bmatrix}$$

This Vandermonde matrix is well known and is invertible.

Now if we look at each row then we observe that R_i is nothing but just powers of i from 0 to $p-2$. then we can write corresponding rows and the scalar product of two rows as under:

$$\begin{aligned}
-R_i &= i^0, i^1, i^2, ...i^{p-1} \\
-R_j &= j^0, j^1, j^2, ...j^{p-1} \\
R_i * R_j &= (i*j)^0 + (i*j)^1 + (i*j)^2 + ... + (i*j)^{p-2} \\
&= k^0 + k^1 + k^2 + \cdots + k^{p-2} \mod p, \text{ where k = i*j mod p} \\
&= \sum_{m=0}^{p-2} k^m \mod p, k \in \mathbb{F}_{p^*} \\
&= \begin{cases} -1 & \text{if } k = 1, \text{where } k = i*j \mod p \\ 0 & \text{otherwise: due to Lemma (2.6)} \end{cases}
\end{aligned}$$

Mathematically: if $R_i * R_i = 1$ and $R_i * R_j = 0$ where $i \neq j$ then such matrix is called **orthogonal matrix** and inverse of such matrix is just the transpose of it.

However in our case matrix A is not orthogonal but has the property that $R_i * R_j = -1$ only if $i * j \mod p = 1$ else 0. This property allows us to find the inverse A^{-1} of matrix A, which is accomplished by permuting the rows of A such that column(i) of A^{-1} = - $(Row(j)ofA)^t$, where $j = i^{-1} \mod p$. we can write as

$$C_i = -(R_{(i^{-1} \mod p)})^t \tag{2.7}$$

where R_j stands for the j^{th} row of the matrix A.

$$A = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_{p-1} \end{bmatrix}$$

and the column C_i stand for the i^{th} column of the matrix A^{-1}.

$$A^{-1} = \begin{bmatrix} C_1 & C_2 & \cdots & C_{p-1} \end{bmatrix}$$

□

2.0.3 Inverse of Matrix M

Inverse of the transition (p x p)matrix M is nothing but the permutation of the M which can be computed in 3 steps as described below:
1. Column 1 = p^{th} column of M but upside down.
2. Last row of the matrix is all 1.
3. Rest of the (p-1)x(p-1)matrix = A^{-1} as discussed above

This is very easily comprehensible in the example below, however the detailed algorithm 2.1.4 is given on the next page.

Example. Here is an example for the case p=7.

matrix M

6	0	0	0	0	0	1
6	6	6	6	6	6	0
6	5	3	6	5	3	0
6	4	5	1	3	2	0
6	3	5	6	3	5	0
6	2	3	1	5	4	0
6	1	6	1	6	1	0

inverse of M

0	1	1	1	1	1	1
0	1	4	5	2	3	6
0	1	2	4	4	2	1
0	1	1	6	1	6	6
0	1	4	2	2	4	1
0	1	2	3	4	5	6
1	1	1	1	1	1	1

M*M' mod 7

1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1

Algorithm for M to M^{-1}
1. C(1) = inverted C(p) of M.
2. Last row of the matrix is all 1.
3. For remaining(p-1)*(p-1)matrix
   $C_i = -(R_{(\frac{1}{i} \mod p)})^t$

Figure 2.1: Conversion matrix and Inverse matrix

2.0.4 Avoiding the Inversion in Computing M^{-1}

In the Above scheme it is shown that in the matrix M^{-1} we compute sub-matrix of dimension (p-1)-by-(p-1) as below,

$$C_i = -(R_{(i^{-1} \mod p)})^t \tag{2.8}$$

where C is the column of A^{-1} and R is row of matrix A so to find the row number of M to fill the Column of M^{-1} we need inversion for $a \in \mathbb{F}_p$. Here we explain how this inversion is avoided to to compute this permutation.

Each of row of matrix M is nothing but the powers of row number.
In sub-matrix A

$$R_i = (i^0, i^1, i^2, ..., i^{p-2}) \tag{2.9}$$

In A^{-1}

$$C_i = -([(i^{-1})^0, (i^{-1})^1, (i^{-1})^2, ..., (i^{-1})^{p-2}])^t \tag{2.10}$$

where as all computation is mod p.
Now since $(i^{-1})^{p-1} = 1$ then

$$(i^{-1})^{p-1} * i = i = (i^{-1})^{p-2}$$

and

$$(i^{-1})^{p-2} * i = i^2 = (i^{-1})^{p-3}$$

$$\vdots$$

$$(i^{-1})^2 * i = i^{p-2} = (i^{-1})^1$$

$$(i^{-1})^1 * i = i^{p-1} = (i^{-1})^0$$

from here we can easily conclude that 2.10 is nothing but the 2.9 with values in the reverse order. That can be written as 2.10 =¿

$$C_i = (i^{p-2}, i^{p-3}, ..., i^1, i^0)$$

which is nothing but 2.9 with values in the reverse order.

Note: This concludes that construction of matrix M is same as construction of matrix M^{-1} and both have equal complexity.

2.1 Basis Conversion

2.1.1 Polynomial to Normal Basis Conversion

Normal basis can be computed from polynomial basis using transition matrix as discussed in Theorem (2.5). Where as the transition matrix is constructed using the algorithm given in 2.1.3 with the time complexity of $\frac{p^2}{4}$. The equation is as shown below:

$$\begin{bmatrix} \beta \\ \beta^p \\ \beta^{p^2} \\ \dots \\ \beta^{p^{p-1}} \end{bmatrix} = \begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0 \end{bmatrix} * \begin{bmatrix} \alpha^{p-1} \\ \alpha^{p-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix}$$

2.1.2 Normal to Polynomial Conversion

As inverse of conversion matrix discussed in Lemma (2.2) above can be now used to compute polynomial basis from normal basis using the formulae below.

$$\begin{bmatrix} 0 & (1)^0 & \dots & (p-1)^0 \\ 0 & (1)^1 & \dots & (p-1)^1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & (1)^{p-2} & \dots & (p-1)^{p-2} \\ 1 & 1 & \dots & 1 \end{bmatrix} * \begin{bmatrix} \beta \\ \beta^p \\ \beta^{p^2} \\ \dots \\ \beta^{p^{p-1}} \end{bmatrix} = \begin{bmatrix} \alpha^{p-1} \\ \alpha^{p-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix}$$

Note: Its clear from the above conversion equations that there is no need of memory to store this matrix, as it can be computed with very little effort during basis computation.

2.1.3 Algorithm for PB-NB Conversion

Input : alpha($a_1, a_2, ...a_p$);
Output: beta ($b_1, b_2, ...b_p$);

Algorithm:
z = (p-1)/2 ;
beta[1] = (alpha[p] - alpha[1]) mod p;
for i = 1 to z do
    x = 1; m = 1;
    y1 = 0; y2 = 0; x1 = 0; x2 = 0;
    for j = 1 to z do
        y1 = (y1 - x * alpha[j]) mod p;
        y2 = (y2 - x * alpha[z+j]) mod p;
        x = x*i mod p;
        x1 = (x1 - m * alpha[j]) mod p;
        x2 = (x2 - m * alpha[z+j]) mod p;
        m = m*(-i) mod p;
    end for;
    beta[i+1] = (y1+ x*y2) mod p;
    beta[p-i+1] = (x1+ m*x2) mod p;
end for;
output = beta;

The above algorithm takes the coefficients of an element $A \in \mathbb{F}_{p^p}$ in the polynomial basis representation as input and computes its coefficients in the normal basis. The time complexity of the algorithm can be seen as $p^2/4$ without any storage requirement. Same algorithm can be extended for the coefficients α_i if they belong to $\mathbb{F}_{p^{2^n}}$.

Note: Normal to polynomial basis conversion can be done by using the algorithm as shown in Figure 2.1, which is given below in detail.

2.1.4 Algorithm for NB-PB Conversion

Input : beta($a_1, a_2, ...a_p$);
Output: alpha($b_1, b_2, ...b_p$);

Algorithm:
z = (p-1)/2 ;
alpha[p] = beta[p];
for i = 1 to z do
    alpha[p] = (alpha[p] + beta[i] + beta[p-i]) mod p;
    x = 1; m = 1; y =0;
    for j = 1 to p-1 do
        x = (x*i) mod p;
        m = m*(-i) mod p;
        y = alpha[p-j];
        alpha[p-j] = (y + m*beta[p-i+1] + x*beta[i+1]) mod p;
        end for;
    end for;
output = alpha;

The complexity of this algorithm is also same as above in algorithm 2.1.3.

2.1.5 Computation Complexity of the Algorithm 2.1.3 for $\alpha \in \mathbb{F}_{p^{2n}}$

To assess the complexity of an algorithm there are mainly two ways to compare. First we see how much time will it take to execute and the other one is how many base field operations are required to complete the task. As we can see from algorithm its time complexity is $\frac{p^2}{4}$ now we compute its binary complexity which can be described in terms of base field operations.

Let M denote the multiplication in $\mathbb{F}_p$ and A the Addition in $\mathbb{F}_p$.
No of rounds = $\frac{p^2}{4}$.
No of prime field multiplications = 2M
No of multiplications in the field of $\alpha \in \mathbb{F}_{p^{2n}}$ with $x \in \mathbb{F}_p is 4 = 8nM \in \mathbb{F}_p$
No of additions in the field of $\alpha \in \mathbb{F}_{p^{2n}} is 4 = 8nA \in \mathbb{F}_p$.
Cost of single round = $(2n + 2)nM + 8nA$
Total cost of the algorithm = $\frac{(8n+2)}{4} np^2M + 4p^2nA$
which can be approximately estimated to = $\mathbf{4np^2M + 4np^2A}$.

2.1.6 Cost Comparison

To the best of our knowledge in the literature there is no such basis conversion discussed over the composite extension fields of odd primes, Moreover our focus was to explore the mathematics behind such composite fields and to convert to normal basis for only exponentiation purposes and revert back to polynomial basis for other operations when ever required. Hence below we compare its cost to the multiplication in the same extension fields so that the comparison can be made for exponentiation computations.

Since there are some algorithm discussed in the literature for conversion to Optimal normal basis (ONB) of type I and II for efficient multiplication over some special Field extensions, Hence we compare our result to these suggestion in the table below.

Algorithm	Memory complexity	Time complexity	Finite Field
B.Sunar,E.Savas [10]	m+n	mn	2^{mn}
I.Muchtadi,F.Yuliawan[12]	m	m^2	2^{mn}

Table 2.1: Basis Conversion Cost for Binary Field

Algorithm	Memory complexity	Time complexity	Finite Field
Our Proposed	0	p^2	p^{2pn}

Table 2.2: Basis Conversion Cost for General odd Prime p

2.1.6.1 Cost of Multiplication in the Finite Field $\mathbb{F}_{p^{2pn}}$

In [2] authors have discussed the cost in such composite fields with detail at pg-58 where they have explored the Schonhage-Strassen and Karatsuba multiplications methods and have given the best cost for the case $p = 7$ as;

$$M(\mathbb{F}_{p^{2pn}}) = 39M(\mathbb{F}_{p^n}) + 728nM + 1079nA$$

If we take the values of p and n as p = 7 and n = 31, then the finite field will become of the size approximately 1300 bits. The security of the 1300 bit field size for pairing based crypto system can be regarded equal to crypto system based on exponentiation in the multiplicative group of same size. If we compare the cost of our conversion method with the multiplication then the above estimate can be regarded as

$$\geq 39(1004A + 92M) + (728 * 31)M + (1079 * 31)A$$

$1M = 76952$ operations in prime field keeping roughly the cost of addition = cost of multiplication
$1Conv = 8 * 31 * 49 = 6076$ operations.

The results for comparison are tabulated as below;

Operation	No of operations in the Prime Field
Multiplication as in [2]	76952
Proposed Basis Conversion	12152

Table 2.3: Single Conversion vs Single Multiplication

single Multiplication = 6.33 * cost of basis conversion
Hence for efficient implementations one can use the basis conversion whenever there is need to perform exponentiation to power p and then can convert back to polynomial basis for the efficient multiplication/inversion operations in polynomial basis with the cost of less than one third of that single multiplication.

Remarks:
1. It is important to highlight here that size of Finite field changes exponentially with the size of the prime p. For instance if we use the structure $\mathbb{F}_{p^{2pn}}$ and keeping the size of the n = 29 as in [2] constant then changing the size of the prime from 3 bit to 5 bit will result in changing the size of the finite field from 1000 to 9000 bits approximately. This shows that even at very high security level, the size of the p is not much so the conversion matrix cost will not be high as compared to multiplication in the field.
2. The matrix M and its inverse as discussed above are shown to prove the inversion, however for basis transition actually there is no matrix arithmetic involved because every row can be computed one by one for each basis element as shown in the algorithm 2.1.3
3. Since $f(x) = x^p - x - a$ is also irreducible over $\mathbb{F}_{p^2}$, this work can be easily extended to the fields $\mathbb{F}_{p^{2pn}}$. From this, one can use generic algorithms e.g Montgomery, Karatsuba, algorithms to have efficient computation of field operations on $\mathbb{F}_{p^{2pn}}$, for any n, where $\gcd(p, n) = 1$ as authors in [2] suggested in their algorithm.

CHAPTER 3

COMPUTING TATE PAIRING EFFICIENTLY OVER THE FINITE FIELDS OF CHAR 7

3.1 Introduction

The importance of Pairing based cryptography was fist realised by the introduction of ID-based encryption (IBE) by Adi Shamir in 1984. In which first time an Id(e.g: email address or any personnel id number) could be used as public key to encrypt the data or signature authentication. Since then much work has been done in this area and many applications have been devised based on pairings, e.g: ID-based key agreement, short signatures schemes, group signatures schemes, ring signatures, certificate-less encryption, hierarchical encryption, attribute-based encryption etc. There are two well knows type of pairings in use such as Weil Pairing and Tate Pairing.

In this part we study the Tate pairing computation over the Composite Finite Field of type $\mathbb{F}_{714n}$. Pairing computation over this composite field is only discussed in [2] till now, we will introduce some improvements in the Tate Pairing computation over characteristics 7 which can be achieved by using the mathematical improvements we suggested in the Chapter 2. The interest in implementation of pairings over elliptic and hyperelliptic curve in the Finite Field of odd characteristics 3 arose in [[15],[28]] and hence it resulted in the study of arithmetic in these fields. Previously arithmetic over odd primes p and its extension fields of type $\mathbb{F}_{p^{2pn}}$ was proposed in [[21],[25],[23]] for the case p = 3. Later on the authors in [23, 29] generalised the arithmetic in this type of composite fields for any prime $p \equiv 3 \mod 4$ and modified the Tate pairing computation over the hyperelliptic curve of type $C_b = Y^2 = X^p - X + b, b = \mp 1$ over $\mathbb{F}_{p^n}$. The authors in [16] and [17] implemented the same arithmetic over the same composite fields structure and show that char 3 always outperform the char 2 fields for pairing computation due to the advantage of faster cubing of the field elements.

In [2] the authors conclude that for large enough the value of n, the larger the prime p we use the lower the complexity of hardware implementation is achieved, and in this research over the case $p = 7$ he proves that instead of using $p = 3$ if we use $p = 7$ then the speed of Tate-pairing computation over such field is approximately six times faster.

3.2 Algorithm for Tate Pairing

The algorithm we are going to discuss is now known as Duursma-Lee-kwon (DLK) due to I.Duursma, H.S.Lee [20] and kwon [27]. These authors defined the same over the finite fields with char 3 however in [2] authors generalised the same for the case of characteristics 7. The Deatails of the algorithm are as under:

Let $C_b = Y^2 = X^p - X + b, b = \mp 1$ over $\mathbb{F}_{p^n}$ and $\rho \in \mathbb{F}_{p^p} \in F = \mathbb{F}_{p^{pn}}$ be a root of the polynomial $x^p - x + 2b$ over $\mathbb{F}_p$ *and* $\sigma \in \mathbb{F}_{p^2} \in K = \mathbb{F}_p^{2pn}$ be a root of the polynomial $x^2 + 1$. The polynomial $x^p - x + 2b$ is an irreducible polynomial over the field $\mathbb{F}_p$. Hence all its roots belong to the field $\mathbb{F}_{p^p}$. Simultaneously, the polynomial $x^p - x + 2b$ is also irreducible over the field $\mathbb{F}_{p^2}$ then,

Let the points $P = (\alpha, \beta) \in C_b(k)$ and
$Q = (\rho - x, \sigma y) \in F \times K, \quad (x, y) \in C_b(k)$, where
$\sigma^2 = -1, \quad \rho^p - \rho + 2b = 0, \quad k = \mathbb{F}_{p^n}, F = \mathbb{F}_{p^{pn}}), and\ K = \mathbb{F}_{p^{2pn}}$, be given.

Initialization:
$f = 1; \quad x = x^p; \quad y = y^p; \quad d = -(2n-1)b$

Calculation:
(1) for i = 1 to n,

$$\begin{aligned} \alpha &= (\alpha^p)^p \\ \beta &= (\beta^p)^p \\ g &= (\beta y\sigma - (\alpha + x + d - \rho)^{(p+1)/2} \\ f &= f^p g;\ y = -y;\ d = d + 2b \end{aligned} \tag{3.1}$$

(2) $\quad f = f^{p^{pn}-1}$
Return f.

In Algorithm above variables and their corresponding fields are as under.
$\alpha, \beta, x, y \in \mathbb{F}_{p^n}$, $\sigma \in \mathbb{F}_{p^2}$, $\rho \in \mathbb{F}_{p^p}$, and $d, b \in \mathbb{F}_p$.

In [2] authors considered the case prime $p = 7$ and used polynomial basis in the algorithm and estimated its cost to be 6 times faster than the same algorithm in the case $p = 3$ with the same security level. As it is shown in algorithm above that there is p^{th} power computation in the final exponentiation as well as for $f = f^p * g$. Where as value of $f \in \mathbb{F}_{p^{2pn}}$ which is the top field, so if we convert the polynomial basis element f to normal basis as suggested in Chapter 2 the cost of the algorithm can be reduced significantly which we will discuss in the next section.

3.3 Hardware Implementation of DLK algorithm

In [2] authors have devised different efficient schemes for arithmetic necessary for hardware implementation of DLK algorithm over composite field of characteristics 7, where as the same was done for characteristics 3 by [20],[15] etc. We suggest few changes by using the hybrid approach of finite field basis from the work in Chapter 2 and estimate the improvement in the implementation of the said algorithm.

3.3.1 Implementation of DLK Algorithm Using Hybrid Basis Approach

Here by hybrid basis we mean that for loop in the DLK algorithm we can use the polynomial basis as discussed in [2] and for final exponentiation we can convert to normal basis, compute the exponentiation and convert back to polynomial basis if required.

Before we discuss the computation cost we will give a short introduction to the involved arithmetic.

Lemma 3.1. *The multiplication of a constant $a \in \mathbb{F}_p$ with any element $\alpha \in \mathbb{F}_{p^n}$ can be carried out by the bits rotation only.*

Proof. Since $a \in \mathbb{F}_7$ is constant so it can be multiplied using bit rotations only, proof of the same follows from Lemma-4 in [2, pp-52] . □

Lemma 3.2. *Polynomial Basis to Normal Basis conversion and vice versa in $\mathbb{F}_{7^{14n}}$ over $\mathbb{F}_{7^{2n}}$ can be computed using 98n Additions in $\mathbb{F}_7$.*

Proof. Since multiplication in $\mathbb{F}_7$ or an element in $\mathbb{F}_p$ and $\alpha \in \mathbb{F}_{7^n}$ can be achieved by just a rotation of elements as in [2] and lemma 3.1.Now if $\alpha \in \mathbb{F}_{7^n}$ then it can be written as polynomial of degree $n-1$ e.g: $\alpha = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$. and if $\alpha \in \mathbb{F}_{7^{2n}}$ then

$$\alpha = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1}) + \sigma(b_0 + b_1x + \cdots + b_{n-1}x^{n-1})$$

$$= \alpha_l + \sigma\alpha_r$$

where $\sigma = \sqrt{-1}$; $a_i, b_i \in \mathbb{F}_7$ and $(\alpha_l, \alpha_r \in \mathbb{F}_{7^n})$

So if $(a \in \mathbb{F}_7)$, $(\alpha \in \mathbb{F}_{7^n})$ then their product can be performed free of cost, with negligible cost for shifting from 3.1 and same is true for $(\alpha_l, \alpha_r \in \mathbb{F}_{7^n})$.

Now by looking at Algorithm 2.1.3 we can count the number of mod p additions which is equal to 4 and there are $\frac{7^2}{4}$ rounds in the algorithm so there are total $2n * 7^2$ number of mod p additions = 98n. □

3.3.2 Normal Basis and Final Exponentiation

It follows from 3.2 that basis conversion from polynomial to normal and back can be computed using only 49 additions in prime field of characteristics 7. In the DLK algorithm we can see that the final exponentiation in the form of $f^{p^{pn}-1}$ is required to be computed. Let $q = p^{2n}$ and let $f \in \mathbb{F}_{q^p}$ then f can be represented in Normal basis generated by $\beta \in \mathbb{F}_{q^p}$ as below:

$$f = c_0\beta + c_1\beta^q + c_2\beta^{q^2} + ... + c_{p-1}\beta^{q^{p-1}}. \tag{3.2}$$

where $c_i \in \mathbb{F}_q$. Now we can write $f^{p^{pn}-1}$ as

$$f^{p^{pn}-1} = f^{p^{pn}} * f^{-1}. \tag{3.3}$$

on the other hand $f^{p^{pn}}$ can be written as

$$p^{pn} = p^{2n(\frac{p-1)}{2}+n} = p^n * (p^{2n})^{(\frac{p-1}{2})} = p^n * q^{(\frac{p-1}{2})}. \tag{3.4}$$

Therefore we can rewrite $f^{p^{pn}-1}$ as

$$f^{p^{pn}-1} = (f^{q^{(\frac{p-1}{2})}})^{p^n} * f^{-1}. \tag{3.5}$$

now let $m = \frac{p-1}{2}$ and let $f' = f^{q^m}$ then equation 3.5 can be rewritten as

$$f^{p^{pn}-1} = (f^{q^m})^{p^n} * f^{-1} = (f')^{p^n} * f^{-1}. \tag{3.6}$$

Since $f' = f^{q^m}$ can be computed free of cost from f using Normal basis presentation from equation 3.2 as below.

$$f = c_0\beta + c_1\beta^q + c_2\beta^{q^2} + ... + c_{p-1}\beta^{q^{p-1}}.$$

and

$$f^q = c_0\beta^q + c_1\beta^{q^2} + c_2\beta^{q^3} + ... + c_{p-1}\beta^{q^p}.$$

since $\beta^{q^p} = \beta$ therefore

$$f^q = c_{p-1}\beta + c_0\beta^q + c_1\beta^{q^2} + ... + c_{p-2}\beta^{q^{p-1}}$$

is just the right rotation of the coefficients of $f : c_0, c_1, c_2, ..., c_{p-1}$.

Then it is clear that f^{q^m} is just m times right rotation of the coefficients $c_0, c_1, c_2, ..., c_{p-1}$ in the normal basis representation of f. so we can write f^{q^m} as below:

$$f^{q^m} = c_{p-m}\beta + c_{p-m+1}\beta^q + c_{p-m+2}\beta^{q^2} + ... + c_{p-m-1}\beta^{q^{p-1}}$$

because $m < p$.

Now we can represent f' in polynomial basis and using equation 3.6 we can compute $f^{p^{pn}-1}$ by first computing $(f')^{p^n}$ then multiplying with f^{-1}.

3.4 Cost Estimation if p =7 , n=31

Cost of the final exponentiation $f^{p^{pn}}$ which is pn fold the cost of single exponentiation from [2, pp-65] is as under:

$$p(4(p-1)p + 2p^2 log(p))n^2 A(7) = 9765 * 7 * 31 = 3107874A(7)$$

From 3.3.2 we compute the f' with just $49 * 2n = 98nA(7)$ and from the above equation cost of computing f'^{p^n}

$$= \frac{3107874}{7} = 443982A(7)$$

So by using hybrid aproach of basis and using our proposed conversion total cost for computing $f^{p^{pn}} = 98 * 31 + 443982 = 447020A(7)$

These results in terms of A(7) which is the number of additions in prime field where p =7 can be tabulated as below:

Method	Final exp cost A(7)
using poly Basis & Scheme in [2]	3107874
Using Hybrid (proposed)	447020

Table 3.1: Exponentiation Complexity Comparison for p = 7 , n = 31

Total Cost of the DLK algorithm using optimized DFT multiplication as in [2, pp-66] is given as under :

$$43nM(\mathbb{F}_{7^n}) + 2040n^2 A(7) + 24nA(7) = 43n(2560)A(7) + 2040n^2 A(7) + 24nA(7)$$

As $1M(\mathbb{F}_{7^n})$ = 2560 A(7) from [2, pp-56] by assuming the cost of 1M(7) = 1A(7).then the total cost of DLK = 5373664 A(7).
Cost after using our proposed hybrid approach for basis conversion can be estimated by deducting the cost saving in terms of exponentiation using normal basis from the above total cost. Then we can write cost using our proposed technique equals

$$5373664 - 3107874 + 447020 = 2660854A(7)$$

Following table compares the cost of DLK algorithm:

Method	Complexity A(7)
using poly Basis & Scheme in [2]	5373664
Using Hybrid (proposed)	2660854

Table 3.2: Tate Pairing Complexity using Polynomial Basis vs Proposed Hybrid Basis

Hence by introducing the hybrid approach in the DLK algorithm we can make the system 1.98 times faster than the one proposed in [2], which was previously the fastest Tate pairing computation over the characteristics 7 composite field.

CHAPTER 4

FPGA IMPLEMENTATIONS

4.1 Introduction

Field Programmable Gates Array(FPGA) is a reconfigurable chip in which hardware circuit can be configured through software programming(Implementation). During the study of FPGA based Elliptic Curve Cryptography computation platform we started with Implementations of the necessary components of cryptography protocols. In this work we successfully implemented all the components required for Elliptic Curve based signature algorithm (ECDSA) over binary Koblitz curve using the latest improvements in the theory of such computations. In this section we will discuss the following work done during the study of this thesis work.

1. Improvement in case of multiplication mod Mersenne prime P=7.
2. Hardware implementation of multiplication mod 7.
3. Hardware implementation of AES and SHA-1.
4. Hardware implementation of multiplication and squaring using Barret reduction with out pre-computation.
5. Hardware implementation of Inversion and Addition operations in the Finite Field suitable for Koblitz Binary Curve 163 based Elliptic Curve operations.
6. Hardware implementation of Scalar Multiplication over Koblitz Binary Curve K-163.

4.2 Multiplication of two Integers mod Mersenne Prime P

In [2] authors have elaborated a smart scheme to implement arithmetic of multiplication in $\mathbb{F}_7$ and given an algorithm to compute 3 bit multiplier with 25 logic gates with the depth(latency) of 5 gates. Scheme works as under:
Let $(x_2x_1x_0)_2$ and $(y_2y_1y_0)_2$ be the two multiplicands then their product modulo 7 can be computed by introducing a sign bit and a low hamming weight 3 bit multiplier due to the fact that modulo 7 multiplication with 3 is equal to the multiplication with -4 so one can omit the multiplier 3 due to its binary hamming

weight which is 2, instead multiply with 4 which is of hamming weight 1 (only single bit is 1) and xor the result with sign bit which is 1 in this case.
Since for any Mersenne prime $2^m \equiv 1 \mod M$ Where M is the Mersenne Prime of length m bits. So in the above case multiplication with 4 is nothing but just the rotation of the bits of the other multiplier.e.g

$$(101)_2 * (011)_2 \equiv (101)_2 * (100)_2 \oplus m \equiv (110)_2 \oplus 1 \equiv (001)_2$$

$$5 * 3 \equiv 1 \mod 7$$

Where m is 1 when ever hamming weight of the multiplicand is ≥ 2 in case of p=7. This scheme is given in [2, pp-50].

Note: This idea is nothing but just the same as discussed in [31] for the bigger primes multiplication.

After analysing the referred algorithm we have noticed that if we use the same technique for both multiplicands then the hardware cost (gate count) can further be reduced, hence we present the new scheme as below.

Instead of finding the sign bit for single value we compute the sign bit for both multiplicands and reduce the 3 bit multiplier to two bit and a sign bit addition due to the same reason as above. This fact can be visualised from the table below as:

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	0
2	0	2	4	6	1	3	5	0
3	0	3	6	2	5	1	4	0
4	0	4	1	5	2	6	3	0
5	0	5	3	1	6	4	2	0
6	0	6	5	4	3	2	1	0
7	0	0	0	0	0	0	0	0

Figure 4.1: F7 Multiplier Results

As it is obvious that values in all 4 quadrants are same except the sign bits addition or the direction of the values so if we compute only 1st quadrant then rest all values can be derived from that by just managing the sign bit, and we have done it as below in the algorithm

4.2.1 Algorithm Code to Multiply two Elements in $\mathbb{F}_7$

```
module  gf7_mult  ( inp1 , inp2  , out ) ;
// Inputs
input   [2:0]  inp1,inp2   ;  // two 3 bit number input
// Outputs
output  [2:0]  out   ;  //  >out put is mod 7 mult result

// Internal nets
wire x0,x1,x2,y0,y1,y2,d0,d1,d2,m0,m1,m2,m3,z0,z1,z2,o0,o1,o2;
// Step 1
assign x0 = inp1[0] ^ inp1[2]; assign x1 = inp1[1] ^ inp1[2] ;
assign y0 = inp2[0] ^ inp2[2]; assign y1 = inp2[1] ^ inp2[2] ;
// Step 2
assign m0 = x0 | x1; assign m1 = y0 | y1 ;
assign m2 = inp1[2] ^ inp2[2] ;
// Step 3
assign m3 = m0 & m1 ;
assign d0 = x0 & y0 ; assign d1 = x1 & y1 ;
assign d2 = x0 & y1 ; assign d3 = x1 & y0 ;
// Step 4
assign m  = m2 & m3 ; assign z0 = d0 & ~d1;
assign z1 = d2 | d3 ; assign z2 = d1 & ~d0 ;
// Step 5
assign o0 = z0 ^ m ;  assign o1 = z1 ^ m ;
assign o2 = z2 ^ m ;

assign out = {o2,o1,o0};
endmodule
```

Figure 4.2: $\mathbb{F}_7$ Multiplier Algorithm

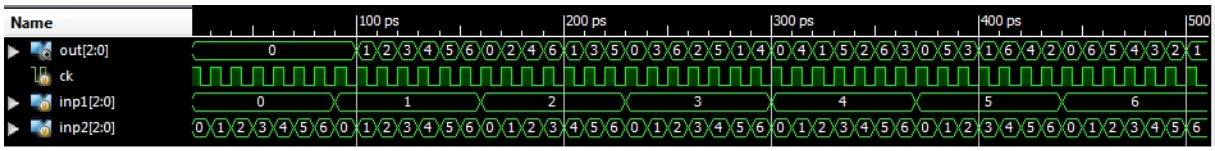

Figure 4.3: FPGA Implementation Simulation

4.2.2 Complexity Comparison

In the above algorithm there are 5 steps, where each step is computed at the same time so total gate latency = 5 and the gate count is quite evident its just 19 gates which is $\frac{25}{19} = 1.31$ times efficient than the one described in [2]. Output of the Implementation of the above code is as under.

Algorithm	# of Gates	Latency
as in [2]Algo	25	5
Our Proposed	19	5

Table 4.1: $\mathbb{F}_7$ Multiplication Comparison

Remark : In the Tate Pairing Algorithm 3.2 discussed in [2] if we analyse the complexity of the algorithm in terms of multiplications and addition [2, pp-67]. We can see it is in the form of $43m((m^2)M(7) + (m-1)^2A(7))$. We can compute the relation of multiplications to addition as at 1k bits security m = 26 then there are 52% multiplications. Now if we use this multiplication instead of the one used in [2], It will result in **17%** improvement in the hardware complexity of Tate Pairing Algorithm.

4.3 HARDWARE MODULES IMPLEMENTATION FOR CRYPTOGRAPHY PLATFORM

To establish a computation platform for verification of the hardware implementations or make the system practically work one needs to first understand how the embedded system works. By embedded system we mean a crypto core, its surrounding interfaces, a processor to control the flow of data, buses to manage the flow of signal and data. To achieve this there are many tools to be used for different tasks. The following picture will elaborate the use of tools for system development.

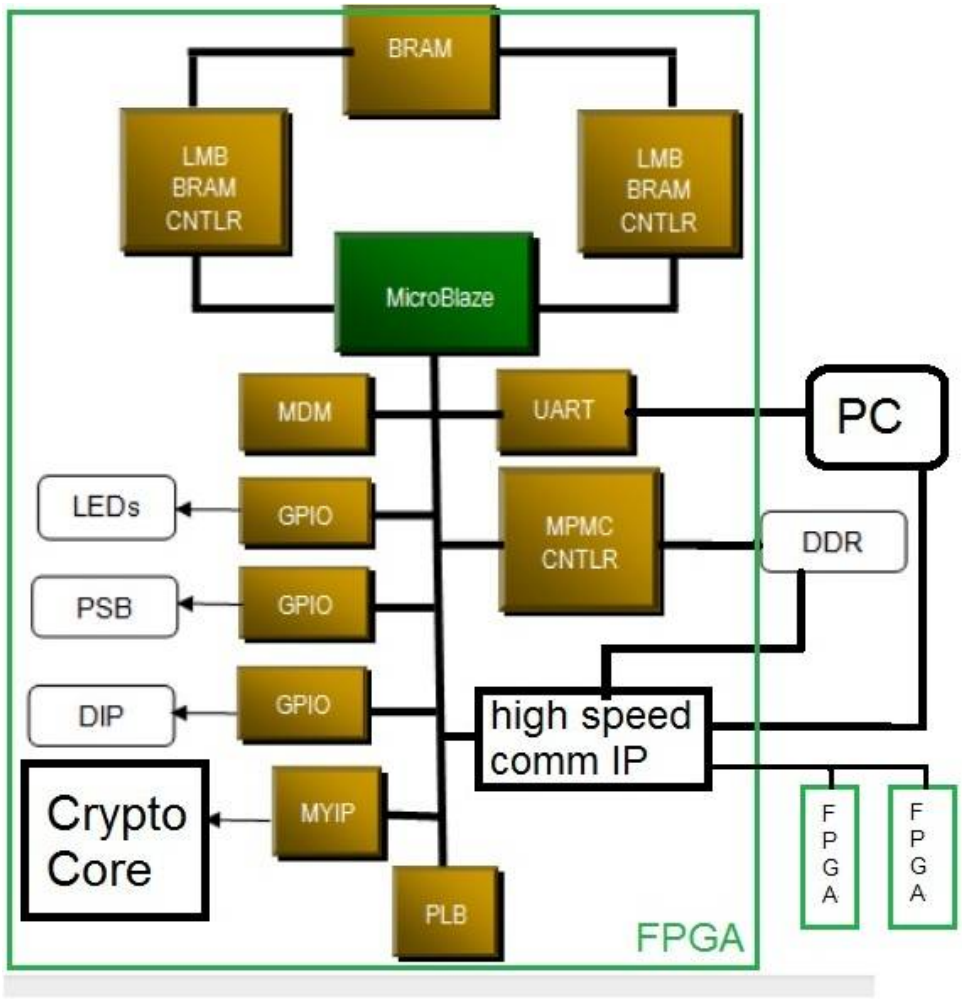

Figure 4.4: Embedded System Layout

In the above figure in a single FPGA, Microblaze and the Crypto Core are two primary cores and all other are supporting items. The Crypto Core handles the crypto codes where as Microblaze controls the interfaces and the flow of data from crypto core to the memory or the outer world of interaction.

In the following figure is the flow of tools which are used to implement the system.

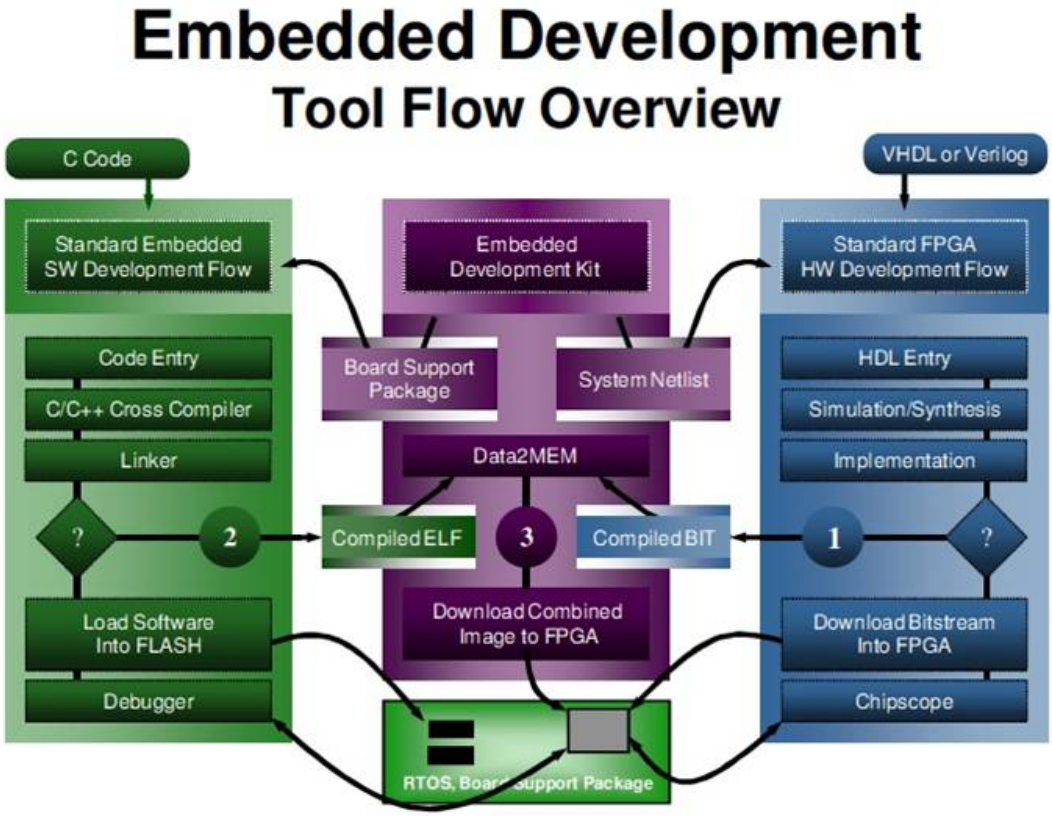

Figure 4.5: Tools Flow Chart

Above figure illustrates the 3 main parts of the Tools flow. 1. Hardware programming to configure the FPGA circuit e.g Crypt Core development. 2. Programming of the already built-in processor (MicroBlaze) for the system control. 3. Embeded development kit. Which physically programs the FPGA and gives the physical access to the out world

Following Modules were implemented in FPGA and were practically verified on the platform.

4.3.1 Hardware Implementation of Mult. over Finite Field 2^{163}

Montgomery Interlaced Multiplication over Finite Field 2^{163} was implemented following is the simulation of the same in 4.6, however the verilog code is appended in Appendix-A

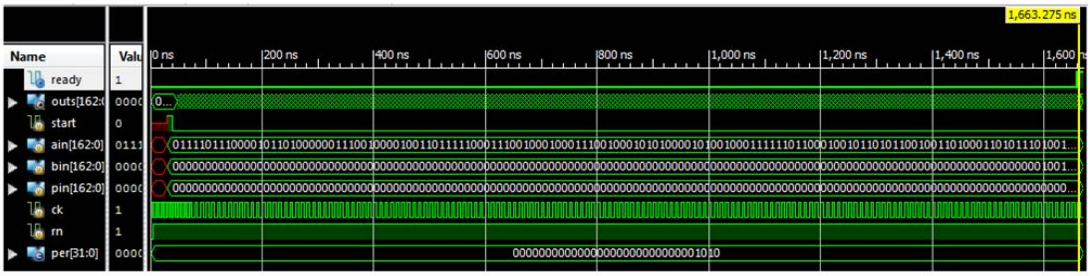

Figure 4.6: Finite Field Multiplier

4.3.2 Inversion Serial and Parallel Algorithm

Inversion in the Finite Field 2^{163} is implemented using both serial and parallel application of euclidean algorithm, so that it can be used as required. As in serial implementation it takes longer time (3.2 ms) but less hardware resources as compared to parallel which consumes more resources but computation time is very less, which only 30 μs

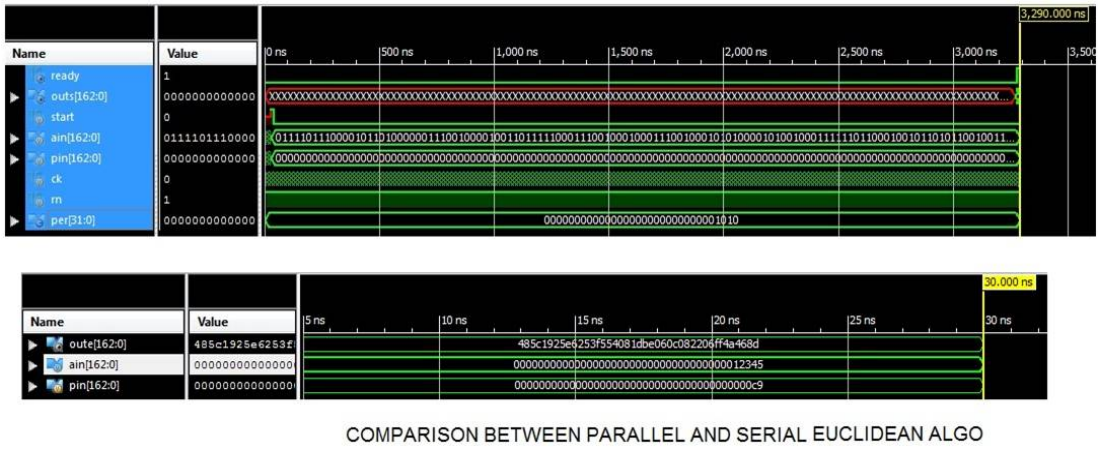

Figure 4.7: Inversion in the Finite Field

4.3.3 Squaring in Binary Fields

Squaring in the finite field was specially implemented to use the Barrett reduction with out pre-computation as in [34]. This is a special reduction algorithm in which there is no need of pre-computation for special moduli in which reduction polynomial has none of its co-efficient a_i greater than 0 for all $i \geq \frac{n}{2}$ except the leading co-efficient. In the following figure 4.8 simulation results are shown in which the hex values of the test vectors are verified.

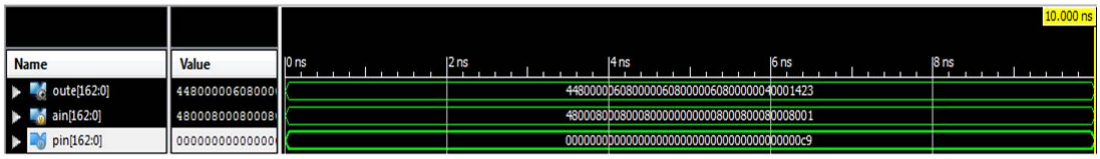

Figure 4.8: Finite Field Squaring with Barret Reduction

4.3.4 Hardware Implementation of AES

We implemented the AES-128 just for reference and verification of the embedded system, which uses all the components on board e.g external memory, Processor, communication with pc takes the inputs from pc and sends back the the encrypted output. It also shows the capabilites of the system for throughput which are shown in the picture 4.9 below.

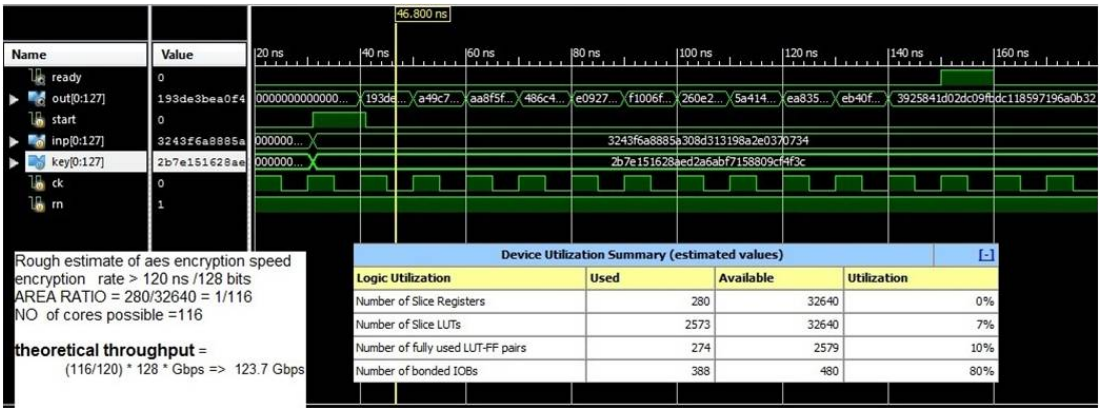

Figure 4.9: AES

CHAPTER 5

CONCLUSION

5.1 Part-1:

In the theoretical work we present new efficient way of basis conversion, which performs the conversion over composite fields of type $\mathbb{F}_{p^{2pn}}$ and characteristics p for any prime p. Main advantage is that conversion is possible with out any memory requirement as compared to such works for binary fields which at least require m elements of the field to be stored where m is the extension degree. We apply the same to Tate pairing computation DLK algorithm and it results in 1.98 times faster computation in hardware in terms of number of prime field operations.

5.2 Part-2:

We implemented the cryptographic algorithms for Binary Curves based cryptography e.g Koblitz-163, Finite Field Arithmetic implementations, AES and SHA-1 Implementations, Efficient Implementation of $\mathbb{F}_7$ multiplication, We also give the codes for micro blaze processor which is used as interface between crypto cores and the outer world and there interfaces. Our efficient implementation of multiplication modulo prime 7 results in 17% better complexity in terms of hardware.

Note: Besides a new basis conversion technique for any prime p and hardware implementations we suggest Tate pairing implementation which will be overall 2.389 times faster than the previous best known for characteristics p=7.

5.3 Future Work:

In the future we intend to implement the Tate pairing to see the practical results based on the previous research and the proposed enhancements in this work. we also study the application of the work discussed here to other algorithm based on such composite fields e.g Short signature Schemes etc

REFERENCES

[1] Rudolf Lidl and Harald Niederreiter, *Finite fields* 2^{nd} *edition*, 1997

[2] S. B. Gashkov, A. A. Bolotov, A. A. Burtsev, S. Yu. Zhebet, and A. B. Frolov *On hardware and software implementation of arithmetic in finite fields of characteristic 7 for calculation of pairings*,journal of Mathematical Sciences, Vol. 168, No. 1, 2010.

[3] Eunjeong Lee, Hyang-Sook Lee and Yoonjin LeeFast,*computation of Tate pairing on general divisors of genus 3 hyperelliptic curves*,2006.

[4] Chih-Hua Chien, Trieu-Kien Truong, Yaotsu Chang and Chih-Hsuan Chen, *A Fast Algorithm to Determine Normal Polynomial over Finite Fields* ,IMECS 2007

[5] Kieren MacMillan and Jonathan Sondow, *Proofs of Power Sum and Binomial Coefficient Congruences Via Pascal's Identity*,2010.

[6] Junfeng Fan , Daniel V. Bailey *Breaking Elliptic Curve Cryptosystems using Reconfigurable Hardware*,2010.

[7] Burton S. Kaliski Jr. and Moses Liskov, *Efficient Finite Field Basis Conversion Involving Dual Bases*,1999.

[8] J. v. z. Gathen, A. Shokrollahi, and J. Shokrollahi, *Efficient multiplication using type 2 optimal normal bases*,Lecture Notes in Computer Science,Springer, 2007.

[9] Burton S. Kaliski and yiqun Lisa Yin,1999 *storage efficient finite field basis conversion*, SAC-98, LNCS 156: 81-93.

[10] Berk Sunar,Erkay Savas, Çetin K.Koc, *Constructing Composite Field Representations for Efficient Conversion*,2003.

[11] Burt Kaliski, Moses Liskov and Yiqun Lisa Yin, *Efficient Finite Field Basis Conversion Techniques*,1999.

[12] I.Muchtadi-Alamsyah and F.Yuliawan, *Basis Conversion in Composite Field*, International Journal of Mathematics and Computation vol 16 Issue 2 (2013).

[13] Berk Sunar, Cetin K. Koc,*Constructing Composite Field Representations for Efficient Conversion*, IEEE Transactions on computers, vol X, No. X, month 2003.

[14] Shigeki Kobayashi, Yasuyuki Nogami, Tatsuo Sugimura, *A Relation between Self-Reciprocal Transformation and Normal Basis over OddCharacteristic Field*, 4th ICCIT conferance 2009.

[15] Paulo S.L.M.Barreto, Hae Y.Kim, Ben Lynn, and Michael Scott,*Efficient Algorithms for Pairing-Based Cryptosystems*,Crypto 2002.

[16] Beuchat,Detrey,Okamoto,*Fast Architectures for the nT Pairing over Small-Characteristic supersingular elliptic curves* ,IEEE Transactions on computers, VOL. 60, NO. 2, February 2011.

[17] Jean-luc Beuchat , Nicolas Brisebarre , Jeremie Detrey, et al, *A Comparison Between Hardware Accelerators for the Modified Tate Pairing over*$\mathbb{F}_{2^m}$ *and* $\mathbb{F}_{3^m}$, 2008.

[18] Junfeng Fan,Daniel V. Bailey,Lejla Batina,Tim Guneysu, Christof Paar and Ingrid Verbauwhede *Breaking Elliptic Curve Cryptosystems using Reconfigurable Hardware* 2009.

[19] R.Granger, D.Page, M.Stam,*Hardware and softeware normal basis Arithmetic for Pairing Based Cryptography in char 3*,2004.

[20] I.Duursma and H.S.Lee, *Tate pairing implementation for hyperelliptic* $y^2 = x^p - x + d$,Asiacrypt-2003.

[21] R.Granger,D.Page and M.Stam, *Hardware and Software Normal Basis Arithmetic for Pairing Based Cryptography in char three*,IEEE Trans.Comput,54 No.7,2005.

[22] *Fast computation of Tate pairing on general divisors of genus 3 hyperelliptic curves*

[23] *Hardware Implementation of Finite Fields of char 3*

[24] M.Scott and P. S. L. M. Barreto, *Compressed pairing*, Advances in Cryptology-CRYPTO 2004, Lect. Notes Comput. Sci., Vol. 3152, Springer, Berlin (2004), pp. 140-156.

[25] T. Kerins, W. P. Marname, E. M. Popovici, and P. S. L. M. Barreto, *Efficient hardware for Tate pairing calculation in characteristic three* ,Cryptographic Hardware and Embedded Systems-CHES 2005, Lect. Notes Comput. Sci., Vol. 3659, Springer, Berlin (2005), pp. 412-426.

[26] A. A. Bolotov, S. B. Gashkov, and A. B. Frolov, *Introductory Elliptic Curve Cryptography.* Protocols of Elliptic Curve Cryptography, URSS, Moscow (2006).

[27] S.Kwon, *Efficient Tate Pairing Computation for Supersingular Curves over binary Fields*,2004.

[28] Paulo S. L. M. Barreto, Steven Galbraith,Colm O hEigeartaigh, and Michael Scott,*Efficient Pairing Computation on Supersingular Abelian Varieties*,2004.

[29] Michael Scott and Barreto,*Compressed Pairing*, Crypto 2004.

[30] Masaaki Shirase1, Tsuyoshi Takagi and Eiji Okamoto*Some Efficient Algorithms for the Final Exponentiation of ηT Pairing*,eprint.iacr.org,2006.

[31] Jeffrey Hoffstein, Joseph H. Silverman, *Random Small Hamming Weight Products with Applications to Cryptography*, 2003.

[32] Simon Blake-Wilson, Hugh MacDonald, *GEC 2: Test Vectors for SEC 1 Certicom Research* , September, 1999

[33] Behrooz Parhami,*Efficient Hamming Weight Comparators for Binary Vectors Based on Accumulative and Up/Down Parallel Counters*.IEEE TRANS. Feb.2009.

[34] Tolga Acar and Dan Shumow,*Modular Reduction without Pre-Computation for Special Moduli*,2010.

APPENDIX A

Verilog Code For Hardware Implementations

Code is provided in DVD format

APPENDIX B

Implementation Guides

Steps to follow to complete an embedded project to run practically :

•Design the crypto core in Xilinx ISE Design Suite
•Simulate the design and achieve the results.
•Open the embedded system project in Xilinx Platform Studio
•Design processor with necessory peripherals and integrate the crypto core
•Export the hardware to SDK for its drivers and processor software programming in C
•In SDK implement the microblaze programming for all modules integration.
•Program the fpga and download the processor program
•Establish control platform at pc to use the embedded system

Changing the necessary files after running project wizard

1. Update user logic if required outer ports then add in user logic and define in top module too
2. Include all he files of vhdl or verilog(of hardware core) in the concern folders
3. Update .PAO file to include hdl files required.
4. Update ucf file if external ports are to be added.
5. Open the project in xps generate netlist and export to sdk.
6. Choose new folder if required for sdk-ws and add standalone bsp
7. Open c project and edit .c file for nessesory computation.
8. Save .c file it will automaticaly compile if selected so, program fpga also .elf file if required else one can run the software by run-as on hardware

Procedure to create and add ip core to xps system
This procedure illustrates how to integrate the developed core into the system

1. Hardware .. create and import ip core
2. Add IPIF (ip interface) registers e.g user logic software registers
3. Create template drivers for software interface
4. Update MPD file to include IP core PORT to connect the port in xps e.g

PORT lcd="", dir=o, VEC=[0:6].
5. Update ip-core.vhd(lcd-ip.vhd)
(a). add port under user ports (e.g: lcd :out std-logic-vector(0 to 6)
(b). map port under user ports mapped here. (e.g: lcd
6. Update user-logic.vhd and add port as above and then add user signal [signal lcd-i : std-logic-vector(0 to 6)]
7. Add the user logic implementation and save and close all files opened
8. Project . rescan user repositories to let changes take effect
9. Add ip-core .. make bus interface connection, make port as external if needed, generate addresses.
10. Update data
system.ucf file for external connections
11. Save close. click hardware generate bitstream .. project-export hardware design to sdk and open sdk
12. Sdk Select TestApp in the project view, right-click, and select Import..Expand General category and double-click on File System
13 Select lab3.c or you own c code file for microblaze and click Finish
14. In sdk .. xilinx tools .. program fpga (testApp.elf) .bmm, .bit

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Sial, Muhammad Riaz
Nationality: Pakistan
Date and Place of Birth: 14-06-1975, Pakistan
Marital Status: Married
Phone: 0090-534-2644248

EDUCATION

Degree	Institution	Year of Graduation
Ph.D	IAM (METU) ANKARA	2013
M.S.	IAM (METU) ANKARA	2009
B.E.	CAE (NUST) PAKISTAN	2001

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
4 years	Pakistan	Research Engineer
2 years	Pakistan	Communication Engineer

PUBLICATIONS

International Conference Publications

Storage Free Basis Conversion over Composite Finite Fields of odd characteristics, ISCTurkey, 2013.

Thesis submitted on sep 4, 2013.
Riaz Sial is with the Institute of Applied Mathematics, Middle East Technical University Ankara, Turkey, e-mail: riazsial @gmail,com
Advisor: Prof.Dr Ersan Akyildiz whs is the Dean of the Faculty of Arts and Sciences and is with the Institute of Applied Mathematics , Middle East Technical University Ankara, Turkey, e-mail: ersan@metu.edu.tr.