8-31-2018

Efficient Bit-parallel Multiplication with Subquadratic Space Complexity in Binary Extension Field

Xiaolin Duan
University of Windsor

Efficient Bit-parallel Multiplication with Subquadratic Space Complexity in Binary Extension Field

by

Xiaolin Duan

A Thesis

Submitted to the Faculty of Graduate Studies

through Electrical and Computer Engineering

in Partial Fulfilment of the Requirements for

the Degree of Master of Applied Science

at the University of Windsor

Windsor, Ontario, Canada

2018

Efficient Bit-parallel Multiplication with Subquadratic Space Complexity in Binary Extension Field

by

Xiaolin Duan

APPROVED BY:

J. Chen

School of Computer Science

M. Mirhassani

Department of Electrical and Computer Engineering

H. Wu, Advisor

Department of Electrical and Computer Engineering

July 26, 2018

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Bit-parallel multiplication in $GF(2^n)$ with subquadratic space complexity has been explored in recent years due to its lower area cost compared with traditional parallel multiplications. Based on 'divide and conquer' technique, several algorithms have been proposed to build subquadratic space complexity multipliers. Among them, Karatsuba algorithm and its generalizations are most often used to construct multiplication architectures with significantly improved efficiency. However, recursively using one type of Karatsuba formula may not result in an optimal structure for many finite fields. It has been shown that improvements on multiplier complexity can be achieved by using a combination of several methods.

After completion of a detailed study of existing subquadratic multipliers, this thesis has proposed a new algorithm to find the best combination of selected methods through comprehensive search for constructing polynomial multiplication over $GF(2^n)$. Using this algorithm, ameliorated architectures with shortened critical path or reduced gates cost will be obtained for the given value of n, where n is in the range of [126, 600] reflecting the key size for current cryptographic applications. With different input constraints the proposed algorithm can also yield subquadratic space multiplier architectures optimized for trade-offs between space and time.

Optimized multiplication architectures over NIST recommended fields generated from the proposed algorithm are presented and analyzed in detail. Compared with existing works with subquadratic space complexity, the proposed architectures are highly modular and have improved efficiency on space or time complexity. Finally generalization of the proposed algorithm to be suitable for much larger size of fields is discussed.

DEDICATION

To my loving Family:

Grandfather and Grandmother

Father and Mother

My wife

My aunt and her family

For their selfless contribution to my life

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Huapeng Wu, for the continuous support of my master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Without his detailed and constructive comments on my research, none of this thesis would be possible.

Besides my supervisor, I would like to show my gratitude to my committee members, Dr. Jessica Chen and Dr. Mitra Mirhassani, for their insightful comments and encouragement in my seminar and defense.

Additionally, I would like to appreciate my loving family members for providing me with unfailing support and continuous encouragement throughout the years of my study and especially during the process of doing research and writing this thesis. This accomplishment would not have been possible without them. Moreover, I wish to especially thank my wife for her understanding and patiently waiting during the time of completing this thesis.

Finally, I wish to extend my gratitude to the people at UWindsor's Faculty of Electrical and Computer Engineering for their help during my study in the MASc program, especially to the department secretary and my best friends.

Xiaolin Duan

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ACRONYMS

ECC Elliptic Curve Cryptography

ECDLP Elliptic Curve Discrete Logarithm Problem

XOR Exclusive-OR

PB Polynomial Basis

SPB Shifted Polynomial Basis

NB Normal Basis

DB Dual Basis

WDB Weakly Dual Basis

KA Karatsuba Algorithm

NIST National Institute of Standards and Technology

GNB Gaussian Normal Basis

ONB Optimal Normal Basis

ESP Equally-Spaced Polynomials

AOP All One Polynomials

CRT Chinese Reminder Theorem

CPF Component Polynomial Formation

CM Component Multiplication

R Reconstruction

GBR Generalized Bernstein's Reconstruction

TMVP Toeplitz Matrix-Vector Product

HDL Hardware Description Language

1 Introduction

Currently, the popularization of smart devices makes the world at people's fingertips. More and more daily activities, such as chatting and shopping, are involved with using the Internet. However, people enjoy the enormous convenience brought by the rapid development of information techniques, while their private information is threatened by potential attackers. In order to protect the data being transmitted over a high risk network, such as the Internet, cryptographic services have been widely used in government, military, culture, education, business, finance and many other fields.

Cryptography is considered as a core technology for network security since the network communication is inseparable from the transmission and storage of encrypted information. According to the keys types, the modern cryptosystems are usually categorized into symmetric key cryptosystems and public key cryptosystems (or called asymmetric key cryptosystems). Taking the advantage of low computational intensity and high throughput, symmetric key systems are often applied in the cryptographic services, such as confidentiality, authentication and data integrity. Since such systems complete the encryption and decryption process with a single key, one inherent problem faced by these systems is how to safely exchanged the key between the sender and receiver.

Public key technique was initially introduced to address the key issue. In public key cryptosystems, a pair of keys, including a public encryption key and a private decryption key, is owned by each user. Anyone who wants to communicate with the others in the system can encrypt the message with the receiver's public key. And the private key can be used by the receiver to decrypt the message. Practically both symmetrical and public key systems work together to provide the message confidentiality service. For example, when bulk confidential data streams, such as media streams and scientific data streams, are transmitted through the Internet, symmetric key systems are responsible for encrypting the data and the key exchange between two involved

parties is realized by public key technique. Moreover, public key cryptography not only can be used to transmit the symmetric keys, it also can independently provide above security mechanisms as well as two unique and indispensable services, digital signature and key management, in network security.

In addition, in such a fast-developing digital society, the speed of computing and network transmission is constantly increasing, and public key cryptography is bound to play an increasingly important role. As more business activities begin to penetrate into the Internet and the potential threat posed by quantum computers, it will be extended to provide reliable security services that covers people's social life. However, intensive computation required in public key cryptosystems is the main issue faced by the promotion of such systems. Therefore, fast algorithms and efficient implementations for public key cryptography have been extensively studied and researched in recent years.

1.1 Motivation

Elliptic Curve Cryptography (ECC), belonging to public key cryptography, has been proposed by N. Koblitz [1] and V. Miller [2] in 1984 and 1985, respectively. Similar to other public key cryptosystem, ECC is also based on a hard mathematic problem. Its security depends on the difficulty of solving elliptic curve discrete logarithm problem (ECDLP), while ECDLP is considered as a much more difficult problem than integer factorization. By far no efficient algorithm has been found to solve this problem. When the key size is large enough (more than 160 bits), it has been shown that ECC is secure against mathematic attacks in terms of current computing capabilities.

In an elliptic curve cryptosystem, encryption and decryption require point multiplications (or called scalar multiplications) on the elliptic curve. The lower level operations used to realize point multiplication are point addition and point doubling, and they can be further decomposed into finite field arithmetic operations. Finite

field arithmetic required in computing elliptic curve point operations contains addition, multiplication and inversion. Addition is straightforward and can be realized with bit-wise XOR operation between two input field elements, while solving inverse can usually be realized with several multiplications. Therefore, the cost of multiplications plays a decisive role in the time and space complexity of an ECC system.

There are two important families of finite field, prime fields and extension fields, both of which can be used to define ECC systems. Among the extension fields, there is one special class of fields called binary extension field and denoted as $GF(2^n)$. It is often used to define and implement ECC systems due to its carry-free arithmetic and suitability for hardware implementation.

Consequently, many works have concentrated on designing a finite field multiplier over $GF(2^n)$ with improved efficiency on area and speed. It is worth noting that finite field arithmetics can be applied not only in cryptography, but also in a variety of applications, such as error-correcting code and quantum error correction.

1.2 Objective

The architectures of finite field multiplication are generally categorized into three types: bit-serial, digit-serial and bit-parallel. Bit-serial multipliers are designed to be most compact, but suffer the longest latency. The bit-parallel structure usually has the smallest latency at expense of large space complexity. Digit-serial multipliers offer trade-offs between time and space complexities. The multipliers also can be subdivided based on the representation of elements. There are several representation bases that have been used for construction of finite field multiplier and reported in literatures, such as polynomial basis (PB), shift polynomial basis (SPB), normal basis (NB), dual bases (DB), weakly dual bases (WDB), and triangular bases. Among them, PB is probably the most commonly discussed and it has been utilized in many cryptographic applications, such as NIST standards for cryptography. Other bases

usually have advantages in certain applications, for example, the squaring operation in normal basis is cost free.

Parallel multiplication is often desired to be used in real-time systems due to its high processing speed. Many applications, such as chip cards, however, require immediacy as well as small area. An effective method to reduce the area of parallel multiplier, called subquadratic space complexity multiplier (called subquadratic multiplier in short), is considered to be most advisable and has been explored in many literatures. Compared with traditional parallel multiplier, which has a quadratic space complexity, multiplications built with subquadratic methods have a space complexity of $O(n^k)$ with $1 < k < 2$. In addition, both of these two types parallel multipliers have logarithm time complexity.

Since Karatsuba algorithm (KA), a 'divide and conquer' technique for efficient integer multiplication, was extended to finite field multiplication with subquadratic space complexity [3], many improvements to this method have been made during the past few years. Specifically, the improvements can be summarized into two subfields: one attempts to improve the architecture of KA with an optimized reconstruction process, and the other focuses on generalizing Karatsuba formulas with reduced number of sub-multiplications.

In [4] and [5], reconstructed KA is independently proposed with improved space complexity. At the same time, a time efficient KA is presented with an application of overlap-free approach [6]. Then inspired by four-way split KA recommended in [4] and improved in [7], an optimized structure of s layers two-way split KA is suggested in [8]. However, original KA is more efficient in the case that field size n is a power of 2. When field size n is not a power of two, Karatsuba-like formulas are introduced. Three-way split formula, considered as the first extension of KA, can be found as early as in [9]. Then more optimized Karatsuba-like are discussed later in [4], [10], [11], [12], [13], [14] and [15].

For current security requirements, several binary extension fields [16] are suggested by National Institute of Standards and Technology (NIST) of US. If an appropriate combination of above methods is applied to forming multiplication in these fields, significantly more improvements can be achieved than using a single method. Several multipliers [4], [15], [17], [18], [19], [20], [21], [22] have been constructed by using such combined methods with improved efficiency.

In this thesis, a comprehensive study and classification of the existing methods on building bit-parallel subquadratic multipliers is presented. Then a new algorithm is proposed to search for and find a combination of these methods to achieve $GF(2^n)$ multiplication with lowest subquadratic space complexity in the range of $160 \leq n \leq 600$. The proposed algorithm provides options of constraints on the inputs such that more architectures are obtained with a trade-off of time and space complexity. Additionally, detailed analysis for the multiplication architectures generated from the proposed algorithm over NIST recommended fields is presented. A comparison with the existing methods has shown that the proposed works are advantageous for different practical requirements that either space or latency is more prominent.

1.3 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 presents the mathematic preliminary of finite field and its arithmetics. A brief introduction of three types of multiplications is also demonstrated in this chapter. In the following chapter, a comprehensive review of existing works of subquadratic space complexity multiplication is introduced. Chapter 4 presents the new work and it starts with an analysis and summary of methods used in proposed algorithm. After introducing the new algorithm in section 2, detailed decomposition and complexity computation of multiplication architectures over NIST fields are discussed as well as a comparison to the existing works in this area. Finally, a summary of our main contributions is given in Chapter

5, and some future works at both of circuits and algorithm levels are suggested.

2 Preliminary

In this chapter, fundamental concepts of abstract algebra including finite fields are first introduced. Binary extension fields is reviewed as a special class of finite field. In the later section of the chapter, arithmetics in $GF(2^n)$, especially finite field multiplication, are discussed in detail with specific attention given to multiplication operation and different styles of multiplication architectures.

2.1 Fundamental Algebraic Concepts

This section briefly presents the definition of three important concepts in abstract algebra: groups, rings and fields.

Definition 2.1. A group, denoted as G, is an algebraic system comprising a set of elements together with a binary operator $(*)$ defined on it. A group must satisfy the following properties[23]:

- For any elements $a, b \in G$, the result of $a * b \in G$.
- For any elements $a, b, c \in G$, $(*)$ is associative: $a * (b * c) = (a * b) * c$.
- For any elements $a \in G$, there is an identity element $e \in G$ such that, $a * e = e * a = a$.
- For any elements $a \in G$, an inverse element $a^{-1} \in G$ exists, and $a * a^{-1} = a^{-1} * a = e$

If the binary operator $(*)$ is commutative and for any elements $a, b \in G$, $a*b = b*a$, the group G is called abelian group[23].

Definition 2.2. A ring, denoted as R, is an algebraic system comprising a set of elements together with two binary operator $(\cdot)$ and $(+)$ defined on it. A ring must satisfy the following properties[23]:

- R is an abelian group in term of $(+)$ operation.
- For any elements $a, b, c \in G$, $(\cdot)$ is associative: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
- For any elements $a, b, c \in G$, $(\cdot)$ and $(+)$ are distributive: $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a$.

Definition 2.3. A field, denoted as F, is an algebraic system comprising a set of elements together with two binary operator $(\cdot)$ and $(+)$ defined on it. A field must satisfy the following properties[23]:

- F is a ring in term of $(\cdot)$ and $(+)$ operation.
- For any elements $a, b \in F$, $(\cdot)$ is commutative: $a \cdot b = b \cdot a$.
- Nonzero elements of F respect to $(\cdot)$ operation form an abelian group.

2.2 Finite Fields

Finite field, also called Galois field, is a field consisting of finite number of elements. It is commonly denoted as $GF(q)$ or $\mathbb{F}_q$, where q is the number of elements in this field. The characteristic x of a finite field $GF(q)$ is defined as the least positive integer x and $ax = 0$ for any element $a \in GF(q)$[23]. There are two important class of finite field.

- Prime fields, denoted as $GF(p)$, is a set of $\{0, 1, 2, ..., p-1\}$, where p is a prime number. In $GF(p)$, the binary operator $(\cdot)$ is defined as *mod-p* multiplication and $(+)$ refers to *mod-p* addition.
- Finite extension fields, denoted as $GF(p^n)$, is a set of polynomials of degree up to n-1 with coefficients belonging to GF(p), and where the variable of these polynomials is a root of irreducible polynomial: $f(X) = \sum_{i=0}^{n} f_i X^i$, for $f_i \in GF(p)$. It is noted that p is prime number and n is a positive integer which

is greater than 1. In $GF(p^n)$, the binary operator $(\cdot)$ refers to *mod-*$f(x)$ and *mod-p* multiplication and $(+)$ is defined as *mod-p* addition.

An irreducible polynomial in finite field can be defined as a polynomial that can not factorized into two smaller polynomials in the same field. In the next section, the irreducible polynomial over the ground field $GF(2)$ will be discussed in detail.

2.3 Arithmetic in Binary Extension Fields

Binary extension field, denoted as $GF(2^n)$, is a special class of finite extension fields with characteristic 2. Elements in this fields can be generated with an irreducible polynomial $f(X)$ of degree n. If X is the root of $f(X)$, a polynomial base can be represented as:

$$\{1,\ X,\ X^2,\ ...,\ X^{n-1}\}$$

And any elements in $GF(2^n)$ can be represented using above basis, such that

$$A(X) = \sum_{i=0}^{n-1} a_i X^i = a_o + a_1 X + \cdots\cdots + a_{n-1} X^{n-1},$$

where $a_i \in [0, 1]$.

Excepting PB, there are many other bases that can be used to represent the elements in $GF(2^n)$, such as SPB, NB, DB, WDB, redundant basis and Dickson polynomial. And NB can be further categorized into several types, such as Gaussian normal basis (GNB) and optimal normal basis (ONB). Different representation methods will have a significant impact on the efficiency of the arithmetic in $GF(2^n)$ with specific applications. For example, NB is attractive because it is almost cost-free for implementing squaring operation in $GF(2^n)$. PB is probably the most popular one discussed and recommended by standard organizations, such as NIST [16]. This thesis will focus on efficient finite field arithmetic represented in PB.

For a given value of n, usually more than one irreducible polynomial exists in $GF(2^n)$. Among them, some special types of irreducible polynomials can be used to achieved a lower complexity of arithmetics in $GF(2^n)$. Irreducible trinomial is often discussed in the literatures since it only contains three non-zero coefficients, which can result in a low complexity modular operation. In some cases, no irreducible trinomial exists in the field, irreducible pentanomial is recommended as an alternative option. It is proved that at least one irreducible trinomial or pentanomial exists for $n \in [3, 10000]$[24]. There are two more classes of irreducible polynomials, equally-spaced polynomials (ESP) and all one polynomials (AOP), used with other algorithms to provide improved architectures. The general expressions of these four irreducible polynomials is shown in the following table.

Table 2.1: Expression of Typical Irreducible Polynomials

	Expression
Trinomial	$f(X) = X^n + X^k + 1, \quad 1 < k < n$
Pentanomial	$f(X) = X^n + X^{k_2} + X^{k_1} + X^{k_0} + 1, \quad 1 < k_0 < k_1 < k_2 < n$
ESP	$f(X) = X^n + X^{(m-1)k} + \cdots + X^k + 1, \quad n = mk, \ 1 < k < \frac{n}{2}$
AOP	$f(X) = X^n + X^{n-1} + \cdots + X + 1$

Since the arithmetic in $GF(2^n)$ is very suitable for hardware implementation, it is widely used in applications such as realizing ECC cryptosystem. In the following subsections, addition, multiplication and inversion will be discussed in $GF(2^n)$.

Let $A(X)$ and $B(X)$ be two elements in $GF(2^n)$, then

$$A(X) + B(X) = \sum_{i=0}^{n-1} (a_i + b_i) X^i \ \ mod\, 2$$

Since one bit modular 2 addition is equivalent to XOR operation, additions in $GF(2^n)$ can be defined as bit-wise XOR operation of the coefficients with same power. Additionally, the implementation of $GF(2^n)$ addition requires only n XOR gates.

There exists a multiplicative inverse of $A(X)\ mod\, f(X)$, denoted as $A^{-1}(X)$, in

this field and

$$A(X)A^{-1}(X) \equiv 1 \ mod\, f(X) \ mod\, 2.$$

One popular way to calculate inversion can be based on Fermat's little theorem [25] and only multiplication is involved in this methods.

The multiplication of $A(X)B(X)$ can be represented as

$$C(X) = A(X)B(X) = (\sum_{k=0}^{2n-2} \sum_{\substack{i+j=k \\ 0\leq i,j\leq n-1}} a_i b_j X^k) \ mod\, f(X) \ mod\, 2$$

The above finite field multiplication can be realized into two steps:

- Polynomial multiplication: the partial products of $a_i b_j$ with the same exponentiation are added together.
- Modular reduction: results from polynomial multiplication do the modular $f(X)$ and modular 2 operations.

As mentioned before, there are three classes of polynomial multiplications based on the input and output modes. An introduction and comparison of these multiplications will be briefly presented in the next section.

In[26], the complexity bound of modular reduction is $(n-1)(r-1)$, where r is the non-zero terms in irreducible polynomial $f(X)$. When irreducible trinomial or pentanomial is considered, this step will only cost at most $2(n-1)$ or $4(n-1)$ XOR gates, respectively.

Additionally, there is a special case of polynomial multiplication. Let $A(X)$ be an element in $GF(2^n)$ with irreducible polynomial f(X). The square of $A(X)$ can be given in the following expression.

$$A^2(X) = \sum_{i=0}^{n-1} a_i X^{2i} \ mod\, f(X)$$

Hence, only modular reduction is required in squaring operation and it can be realized with less gates compared with polynomial multiplication.

2.4 Multiplication and Its Architectures

Space and time complexities are often applied to measure the efficiency of $GF(2^n)$ multiplier. In $GF(2)$, polynomial addition can be realized by a 2-input XOR gate and polynomial multiplication can be implemented with a 2-input AND gate. So the space complexity of multiplier based on binary finite field can be represented by the total number of required AND gates and XOR gates. Let $S_{\otimes}$ and $S_{\oplus}$ denote the cost of AND gate and XOR gate, respectively. And the delays incurred by one 2-input AND gate and one 2-input XOR gate are represented with T_A and T_X, respectively. The symbol "D" is used to denote the critical path of the multiplier. These symbols will also be used in the rest of thesis.

In the three schemes of polynomial multiplication in $GF(2^n)$, the fully parallel and serial structures usually achieve a lowest time and space complexity, respectively; and the digital-serial architectures is a combination of serial and parallel methods and will result in trade-off between time and space.

2.4.1 Bit-Parallel Multiplication

Bit-parallel multipliers are usually recommended for applications with a requirement of high performances because of its large throughput and it can generate result within one clock cycle.

The classical method (or called school-book method) to compute polynomial multiplication is a typical parallel structure. In this approach, all inputs are entered and computed in parallel. The detailed computation steps of classical polynomial multiplication can be shown in the following table [26].

Table 2.2: Classical Method for Computing Polynomial Multiplication

Signal		$S_{\otimes}$	$S_{\oplus}$	D
c_0	a_0b_0	1	0	T_A
c_1	$a_0b_1 + a_1b_0$	2	1	$T_A + T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_{n-2}	$a_0b_{n-2} + \cdots + a_{n-2}b_0$	$n-1$	$n-2$	$T_A + \lceil \log_2 (n-1) \rceil T_X$
c_{n-1}	$a_0b_{n-1} + \cdots + a_{n-1}b_0$	n	$n-1$	$T_A + \lceil \log_2 n \rceil T_X$
c_n	$a_1b_{n-1} + \cdots + a_{n-1}b_1$	$n-1$	$n-2$	$T_A + \lceil \log_2 (n-1) \rceil T_X$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_{2n-3}	$a_{n-2}b_{n-1} + a_{n-1}b_{n-2}$	2	1	$T_A + T_X$
c_{2n-2}	$a_{n-1}b_{n-1}$	1	0	T_A
Total		n^2	$(n-1)^2$	$T_A + \lceil \log_2 n \rceil T_X$

Although the school-book method is the fastest structure among the $GF(2^n)$ multipliers, the applications are limited due to its large space complexity, especially for large fields. So main works in bit-parallel multiplication is to obtained a optimal space complexity with an acceptable critical path delay and it can be summarized into two subfields in terms of the space complexity. The first one focuses on quadratic space complexity multipliers which aims to reduce the space complexity with a slight increase in time complexity. It usually comes with methods such as non-recursive KA[27], Chinese reminder theorem (CRT) [28] and Mastrovito matrix [29]. Recently, a large number of parallel architectures has been proposed in the literature to construct subquadratic space complexity multipliers since it achieves a same asymtotic time complexity with a dramatic decrease in gate cost.

2.4.2 Bit-Serial Multiplication

In the bit-serial multiplication, although final results are obtained after 'n' clock cycle, the lowest area cost make it competitive in the applications with constrained resources. Efforts made in this field are to reduce the latency and maintain a linear space complexity. Based on the input and output sequence, there are four types of bit-level multiplication [30].

- BL-SISO: bit-level serial input and serial output
- BL-SIPO: bit-level serial input and parallel output
- BL-PISO: bit-level parallel input and serial input
- BL-PIPO: bit-level parallel input and parallel output

2.4.3 Digital-Serial Multiplication

In digital level architecture, one operand is separated into multiple digits with a same length. For each clock cycle, each digit is computed with another operand and the result bits are accumulated to form the final sequence. The complexity of this kind of multiplication depends on the size of digit. By choosing a different value of digit length, a wide range of applications can be covered with a consideration of both speed and area.

The following table will present a complexity comparison of these three types of polynomial multiplication.

Table 2.3: Asymtotic Complexity Comparison of Polynomial Multiplication with Different Architecture Styles

Architecture Style		$S(n)$	Latency
Bit-serial		$O(n)$	$O(n)$
Digit-serial		Between $O(n)$ and $O(n^2)$	Between 1 and $O(n)$
Bit-parallel	Quadratic Space Complexity	$O(n^2)$	1
	Subquadratic Space Complexity	$O(n^k)$, $1 < k < 2$	1

The research focus in this thesis is the optimization of bit-parallel binary polynomial multiplication with subquadratic space complexity.

2.5 Elliptic Curve Cryptography

Elliptic curve cryptosystem is a popular public key system as it uses a much shorter key compared to other public key techniques when providing the same level of security

strength. The following graph shows the methodology of designing ECC systems. An ECC system can be implemented with the point operations defined on it and the fundamental layer contains the three finite field arithmetic. In this section, elliptic curves defined over $GF(2^n)$ and point operations performs on the curves are introduced.

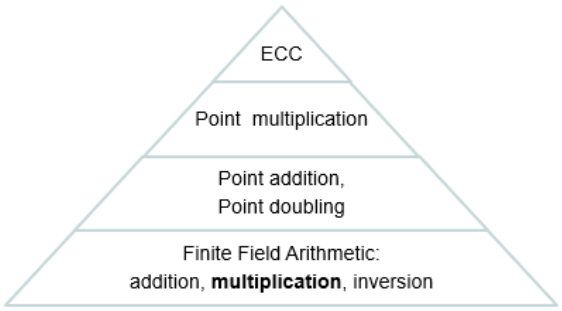

Fig. 2.1: ECC Design Methodology

There are many types of elliptic curve that can be defined in $GF(2^n)$. For cryptographic purposes, an elliptic curve E over $GF(2^n)$ can be simplified as [31]

$$y^2 + xy = x^3 + ax^2 + b$$

where a and b belong to $GF(2^n)$.

Consider the curve E and two points $P = (x_1, y_1)$ and $Q = (x_2, y_2) \in E$. Two basic point operations are defined on this curve.

- Point addition: $P + Q = (x_3, y_3)$ in case of $P \neq Q$;
- Point doubling: $P + Q = (x_4, y_4)$ in case of $P = Q$.

Then the result of point addition $((x_3, y_3))$ is computed with the following equations [31]:

$$\begin{cases} \lambda = \frac{y_2+y_1}{x_2+x_1}; \\ x_3 = \lambda^2 + \lambda + x_1 + x_2 + a; \\ y_3 = \lambda(x_1 + x_3) + x_3 + y_1. \end{cases}$$

And the result of point addition ((x_4, y_4)) can be defined as [31]

$$\begin{cases} \lambda = x_1 + \frac{y_1}{x_1}; \\ x_3 = \lambda^2 + \lambda + a; \\ y_3 = {x_1}^2 + \lambda x_3 + x_3. \end{cases}$$

The points on the curve E with point addition operation form an abelian group. The identity element in this group is ∞, called point at infinity and it is defined as $P+(-P) = P-P = \infty$, for any point $P \in E$. If $P = (x_1, y_1)$, then $-P = (x_1, x_1+y_1)$.

In figure 2.1, the second layer of an ECC system is point multiplication (also called scalar multiplication), which can be built with point addition and point doubling. Consider the curve E defined in above, an integer k and two different points P and Q on this curve. The point multiplication is defined as following:

$$Q = kP = \underbrace{P + \cdots\cdots + P}_{k \text{ times}};$$

The point multiplication is the major operation in elliptic curve based cryptographic protocols, such as elliptic curve Diffie-Hellman key exchange scheme, elliptic curve digital signature and elliptic curve key transport protocols. And the efficiency of the point multiplication is mainly determined by the cost of finite field multiplications. Therefore, algorithm and architecture with lower complexity are desired for finite field multiplication.

3 An Overview of Bit-Parallel Multiplication in $GF(2^n)$ for Composite n with Subquadratic Space Complexity

This chapter contains two sections according to the bases used to represent elements in $GF(2^n)$. Firstly, subquadratic space complexity $GF(2^n)$ multiplication using polynomial basis will be reviewed. In this section, Karatsuba method and its generalizations are firstly introduced when n is a power of small prime. A block recombination based structure is discussed later with further improvements on asymtotic multiplication complexity. Then subquadratic multipliers constructed with a mix of quadratic and subquadratic methods are reviewed with improved efficiency for general binary extension fields. In the second section, subquadratic multiplication algorithms using other bases will be briefly reported.

3.1 Subquadratic Space Complexity Binary $GF(2^n)$ Multiplication Using Polynomial Representation

The method of design subquadratic space complexity multiplication can be traced back to early 1960, when KA [32] was first discovered by Anatoly Karatsuba for fast integer multiplication. KA was later adapted to be applied to polynomial multiplication in early 1980s [9]. After ECC proposed with wide attentions, KA was extended to build $GF(2^n)$ multiplier for cryptographic applications [33]. The current works mainly focus on the design of efficient polynomial multiplication algorithms or structures using improved Karatsuba formulas.

3.1.1 Karatsuba Algorithm

Let A and B be two polynomials of degree $n-1$, where $n = 2^m (m \geqslant 1)$. In KA [3], the input operands A and B are split into two parts shown as:

$$A = \sum_{i=0}^{n-1} a_i X^i = A_1(X) X^{\frac{n}{2}} + A_0(X);$$
$$B = \sum_{i=0}^{n-1} b_i X^i = B_1(X) X^{\frac{n}{2}} + B_0(X),$$

where $A_i(X)$ and $B_i(X)$ are polynomials of degree $\frac{n}{2} - 1$ in X. Then $C = AB$ can be computed as

$$\begin{aligned} C =& AB \\ =& A_0(X)B_0(X) + A_1(X)B_1(X)X^n + (A_0(X)B_1(X) + A_1(X)B_0(X))X^{\frac{n}{2}} \quad (1) \\ =& P_0 + P_1 X^n + (P_2 + P_1 + P_0)X^{\frac{n}{2}}, \quad (2) \end{aligned}$$

where

$$\begin{cases} P_0 = A_0(X)B_0(X); \;\; P_1 = A_1(X)B_1(X); \\ P_2 = (A_0(X) + A_1(X))(B_0(X) + B_1(X)). \end{cases} \quad (3)$$

From (1), C can be computed from four sub-polynomial multiplications and two polynomial additions with half size. However, (2) shows that C can be obtained from three sub-polynomial multiplications of degree $\frac{n}{2} - 1$ and five polynomial additions because of reusing P_0 and P_1. Therefore, one sub-multiplication can be saved at expense of three more polynomial additions in above KA two-way splitting formula.

Specifically, supposing that one iteration of Karatsuba formula is applied to construct the polynomial multiplication and the sub-multiplications are computed with

school-book method, the gate cost can be given as:

$$S_{\otimes}(n) = 3(\frac{n}{2})^2; \qquad S_{\oplus}(n) = 3(\frac{n}{2} - 1)^2 + 4n - 4.$$

As mentioned in previous chapter, computing AB with equation (1) will consume n^2 AND gates and $(n-1)^2$ XOR gates. So, $(\frac{n}{2})^2$ AND gates and $(\frac{n}{2} - 1)^2$ XOR gates will be saved on computing one polynomial multiplication of degree $\frac{n}{2} - 1$. And the increased part of polynomial additions will only cost $2n - 1$ more XOR gates.

For the convenience of complexity analysis, Karatsuba-based multiplication shown in equation (2) can be decomposed into three separate blocks: CPF, CM, and R, which are given by

- CPF: $A_0(X) + A_1(X)$, $B_0(X) + B_1(X)$;
- CM: Computing P_i;
- R: Constructing C with $P_0 + P_1 X^n + (P_2 + P_1 + P_0)X^{\frac{n}{2}}$.

And a comprehensive view of KA is presented in the following diagram.

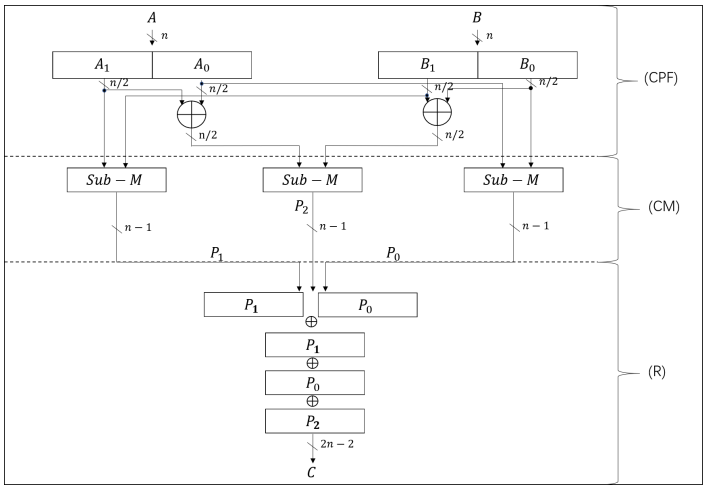

Fig. 3.1: Block Decomposition of Karatsuba Algorithm

It is noted that the computation of P_2 is separated into CPF and CM block. The

component addition is completed in CPF block. In CM block, only multiplication operation is considered.

Time complexity: From equation (2), in the first step, $P_0 + P_1$ and P_2 can be calculated concurrently within the delay of $D(\frac{n}{2})+T_X$, where $D(\frac{n}{2})$ represents the time delay of polynomial multiplication of degree $\frac{n}{2} - 1$. Then by reusing the result from P_0+P_1, one more T_X is required to compute the expressions in brackets in (2). Finally, one XOR gate delay is consumed to compute the overlap part of $(P_2 + P_1 + P_0)X^{\frac{n}{2}}$ and $P_0 + P_1X^n$. Hence, the time complexity of recursive Karatsuba Algorithm is $3T_X + D(\frac{n}{2})$ for $n \geq 4$.

Space complexity: If the space complexity of computing P_i in CM block is denoted as $S_\otimes(\frac{n}{2}) + S_\oplus(\frac{n}{2})$, then the total cost of CM block are 3 times of this value. The computation of CPF part in this iteration will cost n XOR gates, and $3n-4$ XOR gates are required in reconstruction part. It is noticed that the computation of multiplying of X only requires shift operation, which is cost-free in hardware design. So no logic gates are required to calculate $(P_2 + P_1 + P_0)X^{\frac{n}{2}}$ and P_1X^n. Furthermore, because the highest exponent of P_0 is smaller than the lowest exponent of P_1, the summation of $P_0 + P_1X^n$ doesn't cost XOR gates. Therefore, the space and time complexity of recursive Karatsuba Algorithm can be summarized as:

- Space Complexity

$$\begin{cases} S_\otimes(n) = 3S_\otimes(\frac{n}{2}); \\ S_\oplus(n) = 3S_\oplus(\frac{n}{2}) + 4n - 4. \end{cases} \tag{4}$$

- Time Complexity

$$\begin{cases} D_\otimes(n) = D_\otimes(\frac{n}{2}); \\ D_\oplus(n) = D_\oplus(\frac{n}{2}) + 3T_X. \end{cases} \tag{5}$$

If this kind of "Divide and Conquer" method is recursively applied into the computation of CM block until the length of sub-polynomial is 1, the $GF(2^n)$ multiplication

can be achieved with subquadratic space complexity. The non-recursive form of space and time complexity for iterative KA can be derived based on lemma 3 with the following initial values.(It is noticed that when $n = 2$, there is no overlaps occurred in this process.)

$$S_{\otimes}(2) = 3; \qquad D_{\otimes}(2) = 1T_A;$$

$$S_{\oplus}(2) = 4; \qquad D_{\oplus}(2) = 2T_X.$$

Hence, non-recursive form of equations (4) and (5) to solve the space and time complexity of KA based parallel polynomial multiplication can be represented as

$$\begin{cases} S_{\otimes}(n) = n^{\log_2 3}; \\ S_{\oplus}(n) = 6n^{\log_2 3} - 8n + 2; \\ D_{\otimes}(n) = 1; \\ D_{\oplus}(n) = 3\log_2 n - 1. \end{cases} \tag{6}$$

Let $S_{\oplus,CPF}(n)$, $S_{\otimes,CM}(n)$ and $S_{\oplus,R}(n)$ denote the total gates required in CPF, CM, and R blocks over all iterations, respectively. The following table summarizes the space complexity of these blocks in both recursive form and non-recursive form. (CPF part is considered as two parts, the formation of A and B.)

Table 3.1: Space Complexity of the Construction Blocks

	$S_{\oplus,CPF}(n)$	$S_{\otimes,CM}(n)$	$S_{\oplus,R}(n)$
Recursive formula	$\frac{n}{2} + 3S_{\oplus,CPF}(\frac{n}{2})$	$3S_{\otimes,CM}(\frac{n}{2})$	$3n - 4 + 3S_{\oplus,R}(\frac{n}{2})$
Non-recursive formula	$n^{\log_2(3)} - n$	$n^{\log_2(3)}$	$4n^{\log_2(3)} - 6n + 2$
Total gates	$2S_{\oplus,CPF}(n) + S_{\oplus,R}(n) + S_{\otimes,CM}(n) = 6n^{\log_2(3)} - 8n + 2$		

3.1.2 Reconstructed Karatsuba Algorithm

In 2009, Bernstein [4] and Zhou [5] have independently proposed a reconstructed Karatsuba formula with improved space complexity. Their proposed work can be

shown in the following expressions:

$$\begin{aligned} C =& AB \\ =& P_0 + P_1 X^n + (P_2 + P_1 + P_0) X^{\frac{n}{2}}, \\ =& (P_0 + P_1 X^{\frac{n}{2}})(X^{\frac{n}{2}} + 1) + P_2 X^{\frac{n}{2}}, \end{aligned} \tag{7}$$

where

$$P_0 = A_0(X)B_0(X); \quad P_1 = A_1(X)B_1(X);$$

$$P_2 = (A_0(X) + A_1(X))(B_0(X) + B_1(X)).$$

From (7), it is noticed that the difference between two formulas lies in the reconstruction part. The computations of the other two types of blocks (CPF and CM) remain unchanged. Then, one iteration of Karatsuba Algorithm based on reconstructed approach can be depicted in four steps:

1. CPF $\{A_0(X) + A_1(X), B_0(X) + B_1(X)\}$; CM $\{A_0(X)B_0(X), A_1(X)B_1(X)\}$;
2. CM $\{(A_0(X)+A_1(X))(B_0(X)+B_1(X))\}$; R $\{A_0(X)B_0(X)+A_1(X)B_1(X)X^{\frac{n}{2}};\}$;
3. R $\{(A_0(X)B_0(X) + A_1(X)B_1(X)X^{\frac{n}{2}})(1 + X^{\frac{n}{2}})\}$;
4. R $\{(A_0(X)B_0(X) + A_1(X)B_1(X)X^{\frac{n}{2}})(1 + X^{\frac{n}{2}}) + (A_0(X) + A_1(X))(B_0(X) + B_1(X))X^{\frac{n}{2}}\}$.

Time complexity: For each iteration, totally three XOR gate delay will be required excepting CM block. Therefore, the time complexity of the reconstructed formula is same with original KA.

$$D(n) = (3\log_2 n - 1)T_X + T_A$$

Space complexity: In this architecture, the computation of CPF and R blocks need

n and $\frac{5n}{2} - 3$ XOR gates, respectively. Then the space complexity can be specified recursively as

$$S_{\otimes}(n) = 3S_{\otimes}(\frac{n}{2});$$
$$S_{\oplus}(n) = 3S_{\oplus}(\frac{n}{2}) + \frac{7n}{2} - 3.$$

If this reconstructed algorithm is applied recursively, $\frac{n}{2} - 1$ gates can be saved in the first iteration. And the next one will save $3(\frac{n}{2^2} - 1)$. The general formula for computing reduced gates in ith iteration can be shown as

$$3^{i-1}(\frac{n}{2^i} - 1)$$

So, compared with KA, the total number of saved XOR gates for this reconstructed structure is $\frac{1}{2}n^{\log_2 3} - n + \frac{1}{2}$.

The detailed space complexity for each block in reconstructed KA is summarized in the following table.

Table 3.2: Space Complexity of the Construction Blocks in Reconstructed KA

	$S_{\oplus,CPF}(n)$	$S_{\otimes,CM}(n)$	$S_{\oplus,R}(n)$
Recursive formula	$\frac{n}{2} + 3S_{\oplus,CPF}(\frac{n}{2})$	$3S_{\otimes,CM}(\frac{n}{2})$	$\frac{5}{2}n - 3 + 3S_{\oplus,R}(\frac{n}{2})$
Non-recursive formula	$n^{\log_2(3)} - n$	$n^{\log_2(3)}$	$\frac{7}{2}n^{\log_2(3)} - 5n + \frac{3}{2}$
Total gates	$2S_{\oplus,CPF}(n) + S_{\oplus,R}(n) + S_{\otimes,CM}(n) = \frac{11}{2}n^{\log_2(3)} - 7n + \frac{3}{2}$		

3.1.3 Overlap-Free Karatsuba Algorithm

In [6], Fan has proposed a new KA based on overlap-free approach, which can reduce the time complexity of the KA based binary extension field multiplier without increasing its space complexity. In term of two-way split method, the time complexity of overlap-free KA is 33% better. This algorithm can eliminate the overlap addition required in the original KA with a new segmentation method in which the original version of Karatsuba formula divides the operands into the most significant part and the least significant part, and this new approach provides a way of splitting based on the parity of the X's exponent.

Let A and B be two polynomial of degree $n-1$ over $GF(2^n)$, where $n = 2^m (m > 1)$, operands A and B can be expressed as:

$$A = \sum_{i=0}^{n-1} a_i X^i = A_1(X)X + A_0(X);$$
$$B = \sum_{i=0}^{n-1} b_i X^i = B_1(X)X + B_0(X),$$

where

$$A_0 = \sum_{i=0}^{\frac{n}{2}-1} a_{2i} X^{2i}; \qquad A_1 = \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1} X^{2i}$$
$$B_0 = \sum_{i=0}^{\frac{n}{2}-1} b_{2i} X^{2i}; \qquad B_1 = \sum_{i=0}^{\frac{n}{2}-1} b_{2i+1} X^{2i}$$

Hence, A_0 and B_0 are $\frac{n}{2}$ terms polynomial with all the odd exponent elements in A. Similarly, A_1 and B_1 contains the rest $\frac{n}{2}$ even items. Let $Y = X^2$, then

$$\begin{aligned} C =& AB \\ =& A_0(Y)B_0(Y) + A_1(Y)B_1(Y)Y + (A_0(Y)B_1(Y) + A_1(Y)B_0(Y))X \\ =& P_0 + P_1 Y + (P_2 + P_1 + P_0)X \end{aligned} \tag{8}$$

where

$$P_0 = A_0(Y)B_0(Y); \quad P_1 = A_1(Y)B_1(Y);$$

$$P_2 = (A_0(Y) + A_1(Y))(B_0(Y) + B_1(Y)).$$

The overall structure of overlap-free KA is same with the original version, except the different way of splitting the inputs. There are four operations required in this algorithm – multiplication, addition, shift and insert, where shift and insert operations

are cost-free in hardware implementation. So the one iteration overlap-free KA can be described as:

1. CPF $\{A_0(Y) + A_1(Y), B_0(Y) + B_1(Y)\}$; CM $\{A_0(Y)B_0(Y), A_1(Y)B_1(Y)\}$;
2. CM $\{(A_0(Y) + A_1(Y))(B_0(Y) + B_1(Y))\}$; R $\{A_0(Y)B_0(Y) + A_1(Y)B_1(Y)Y$; $A_0(Y)B_0(Y) + A_1(Y)B_1(Y)\}$;
3. R $\{((A_0(Y) + A_1(Y))(B_0(Y) + B_1(Y)) + A_0(Y)B_0(Y) + A_1(Y)B_1(Y))X\}$;
4. Component Interleaving.

Space complexity: From above procedure, it it easy to remark that the cost of CPA and CM block in both original and overlap free KA are same because of the same length of operands. Moreover, the reconstruction parts also need the same amount of XOR gates. Instead of spending $n-2$ XOR gates on overlap part in original approach, overlap-free KA requires $n-2$ XOR gates to perform $A_0(Y)B_0(Y) + A_1(Y)B_1(Y)Y$. Therefore; the space complexity of overlap-free KA is exactly same with original one which is shown in section 3.1.1.

Time complexity: Compared with KA which requires $3T_X + D_{\oplus}(\frac{n}{2})$, overlap-free method only need $2T_X + D_{\oplus}(\frac{n}{2})$. Step 1 and 2 shown above can be finished in $T_X + D_{\oplus}(\frac{n}{2})$. Reconstruction in step 3 costs one XOR delay and the final step is cost-free. For each iteration, one T_x is saved in overlap-free method. So, the iterative formula of time complexity is expressed as

$$D_{\otimes}(n) = D_{\otimes}(\frac{n}{2});$$
$$D_{\oplus}(n) = D_{\oplus}(\frac{n}{2}) + 2T_X.$$

and the non-iterative form is

$$D_{\otimes}(n) = 1T_A;$$
$$D_{\oplus}(n) = 2\log_2 n.$$

3.1.4 Three-Way Split Formula

Another formula proposed to build subquadratic space complexity multiplication is based on three-way split technique in [9]. It is more appropriate when the key size is a power of 3. Consider two polynomials of degree $n-1$, where $n = 3^m (m \geqslant 1)$. The input operands A and B can be equally divided into three parts:

$$A = \sum_{i=0}^{n-1} a_i X^i = A_2(X) X^{\frac{2n}{3}} + A_1(X) X^{\frac{n}{3}} + A_0(X);$$
$$B = \sum_{i=0}^{n-1} b_i X^i = B_2(X) X^{\frac{2n}{3}} + B_1(X) X^{\frac{n}{3}} + B_0(X).$$

Then $C = AB$ can be expended with the following expression:

$$\begin{aligned} C =& AB \\ =& P_0 + R_1 X^{\frac{n}{3}} + R_2 X^{\frac{2n}{3}} + R_3 X^n + P_2 X^{\frac{4n}{3}}, \end{aligned} \tag{9}$$

where

$$\begin{cases} P_0 = A_0(X)B_0(X);\ P_3 = (A_1(X) + A_2(X))(B_1(X) + B_2(X)); \\ P_1 = A_1(X)B_1(X);\ P_4 = (A_0(X) + A_1(X))(B_0(X) + B_1(X)); \\ P_2 = A_2(X)B_2(X);\ P_5 = (A_0(X) + A_2(X))(B_0(X) + B_2(X)); \end{cases} \tag{10}$$

$$R_0 = P_0 + P_1; \quad R_2 = P_5 + P_2 + R_0;$$
$$R_1 = P_4 + R_0; \quad R_3 = P_3 + P_2 + P_1.$$

Totally six scalar multiplications are required in equation (9). And the recursively complexity can be represented as:

- Space Complexity

$$S_{\otimes}(n) = 6S_{\otimes}(\frac{n}{3});$$
$$S_{\oplus}(n) = 6S_{\oplus}(\frac{n}{3}) + \frac{22n}{3} - 10.$$

- Time Complexity

$$D_{\otimes}(n) = D_{\otimes}(\frac{n}{3});$$
$$D_{\oplus}(n) = D_{\oplus}(\frac{n}{3}) + 4T_X.$$

By reconstructing R block in three-way split formula, a lower space complexity is reached in [7] without changing its time complexity. The idea of the new reconstruction process is decomposing the results from sub-multiplications and eliminating redundant computations. Supposing P_0, P_1 and P_2 in equation (10) are split into two equal parts, where $P_i = P_{i,0} + P_{i,1}X^{\frac{n}{3}}$. The final result of C can be computed as

$$\begin{aligned}
&R_{0,1} = P_{0,1} + P_{1,0}; \quad R_{0,2} = P_{1,1} + P_{2,0};\\
&R_{1,1} = P_{0,0} + R_{0,1}; \quad R_{1,2} = R_{0,2} + R_{1,1};\\
&R_{1,4} = P_{2,1} + R_{0,2}; \quad R_{1,3} = R_{1,4} + R_{0,1};\\
&C = (P_{0,0} + R_{1,1}X^{\frac{n}{3}} + R_{1,2}X^{\frac{2n}{3}} + R_{1,3}X^{n} + R_{1,4}X^{\frac{4n}{3}} + P_{2,1}X^{\frac{5n}{3}})\\
&\qquad + P_3X^{\frac{n}{3}} + P_4X^{\frac{2n}{3}} + P_5X^{n}
\end{aligned}$$

This is a space optimized three-way split formula, and its recursively space complexity can be represented as:

$$S_{\otimes}(n) = 6S_{\otimes}(\frac{n}{3});$$
$$S_{\oplus}(n) = 6S_{\oplus}(\frac{n}{3}) + 6n - 6.$$

In [7], another time optimized three-way split formula is also suggested based on overlap-free approach. In this method, the operands is split as follows:

$$A = A_2(Y)X^2 + A_1(Y)X + A_0(Y),$$

where $Y = X^3$. Let the scalar multiplications P_i be defined similar to (10), then C can be computed as:

$$R_0 = P_0(Y) + XP_1(Y) + X^2P_2(Y); \qquad R_1 = R_0(1 + X + X^2)$$
$$C = R_1 + XP_3(Y) + X^2P_4(Y) + X^3P_5(Y)$$

It is observed that there are some redundant bit additions in computing R_1. By reforming the computation process of R_1, the following complexity results can be obtained for this time optimized three-way split formula.

- Space Complexity
$$S_{\otimes}(n) = 6S_{\otimes}(\frac{n}{3});$$
$$S_{\oplus}(n) = 6S_{\oplus}(\frac{n}{3}) + 7n - 9.$$
- Time Complexity
$$D_{\otimes}(n) = D_{\otimes}(\frac{n}{3});$$
$$D_{\oplus}(n) = D_{\oplus}(\frac{n}{3}) + 3T_X.$$

There is another three-way split formula with five scalar multiplication proposed in [4] and later it is improved in [34]. It achieves a lower asymptotic complexity compared with the above reviewed version. However, the large coefficients of the big (O) representation make it more competitive when n is a large number which is not considered in this thesis. Additionally, the higher time delay is another reason that this formula is not chosen in proposed method.

3.1.5 Four-Way Split Formula

Supposing that a polynomial multiplication is expanded by two layers of two-way split KA (Section 3.1.2), a further improvement can be done by optimizing the construction sequence. This kind of four-way split method is proposed in [4], and it can achieve a lower space and time complexity.

Consider two polynomials A and B, which can be equally split into four parts:

$$A = \sum_{i=0}^{n-1} a_i X^i = A_3(X)X^{\frac{3n}{4}} + A_2(X)X^{\frac{n}{2}} + A_1(X)X^{\frac{n}{4}} + A_0(X);$$
$$B = \sum_{i=0}^{n-1} b_i X^i = B_3(X)X^{\frac{3n}{4}} + B_2(X)X^{\frac{n}{2}} + B_1(X)X^{\frac{n}{4}} + B_0(X),$$

where A_i and B_i are sub-polynomials of size $\frac{n}{4}-1$. The regular four-way split method constructs the polynomial multiplication AB by applying two recursions of two-way split KA with 9 recursive polynomial products. The scalar multiplications can be represented as:

$$\begin{cases} P_0 = A_0(X)B_0(X); \quad P_1 = A_1(X)B_1(X); \\ P_2 = (A_0(X) + A_1(X))(B_0(X) + B_1(X)); \\ P_3 = A_2(X)B_2(X); \quad P_4 = A_3(X)B_3(X); \\ P_5 = (A_2(X) + A_3(X))(B_2(X) + B_3(X)); \\ P_6 = (A_0(X) + A_2(X))(B_0(X) + B_2(X)); \\ P_7 = (A_1(X) + A_3(X))(B_1(X) + B_3(X)); \\ P_8 = (A_0(X) + A_1(X) + A_2(X) + A_3(X))(B_0(X) + B_1(X) + B_2(X) + B_3(X)). \end{cases} \tag{11}$$

With the same CM and CPF block as conventional four-way split formula, Bernstein's new method constructs C based on the following expression.

$$\begin{cases} R_0 = P_0 + P_1X^{\frac{n}{4}} + P_3X^{\frac{n}{2}} + P_4X^{\frac{3n}{4}}; \\ R_1 = R_0(1 + X^{\frac{n}{4}}) + P_2X^{\frac{n}{4}} + P_5X^{\frac{3n}{4}}; \\ C = R_1(1 + X^{\frac{n}{2}}) + ((P_6 + P_7X^{\frac{n}{4}})(1 + X^{\frac{n}{4}}) + P_8X^{\frac{n}{4}})X^{\frac{n}{2}}. \end{cases} \tag{12}$$

From equation (11) and (12), the following recursive formulas can be derived for space

and time complexity.

- Space Complexity

$$S_{\otimes}(n) = 9S_{\otimes}(\frac{n}{4});$$
$$S_{\oplus}(n) = 9S_{\oplus}(\frac{n}{4}) + \frac{17n}{2} - 11.$$

- Time Complexity

$$D_{\otimes}(n) = D_{\otimes}(\frac{n}{4});$$
$$D_{\oplus}(n) = D_{\oplus}(\frac{n}{4}) + 5T_X.$$

Compared with two layers of KA , the Bernstein's four-way split method saves $\frac{n}{4} - 1$ XOR gates and one gate delay of XOR. This formula can be considered as a space optimized KA with four-way split.

Another alternative four-way split formula has been proposed in [7] and it is a time optimized method with reduced space complexity. The idea of this method is to apply the equation (12) into two recursions of overlap-free KA.

Let $Y = X^4$, then two n-terms polynomials A and B are decomposed into four blocks based on overlap-free split method, which can be shown as:

$$A = \sum_{i=0}^{n-1} a_i X^i = X^3 A_3(Y) + X^2 A_2(Y) + X A_1(Y) + A_0(Y);$$
$$B = \sum_{i=0}^{n-1} b_i X^i = X^3 B_3(Y) + X^2 B_2(Y) + X B_1(Y) + B_0(Y),$$

where $A_i(Y) = \sum_{j=0}^{\frac{n}{4}-1} a_{i+4j} Y^j$ and $B_i(Y) = \sum_{j=0}^{\frac{n}{4}-1} b_{i+4j} Y^j$. The CPF and CM blocks can be computed in a similar way shown in equation (11). After obtaining the 9 recursive polynomial multiplication products, $C = AB$ can be formed by the following

expression.

$$R_0 = P_0(Y) + P_1(Y)X + P_3(Y)X^2 + P_4(Y)X^3;$$
$$R_1 = R_0(1+X) + P_2(Y)X + P_5(Y)X^3;$$
$$C = R_1(1+X^2) + ((P_6(Y) + P_7(Y)X)(1+X) + P_8(Y)X)X^2.$$

By eliminating the overlap part, this time optimized four-way split KA reduces one more XOR gate delay compared with Bernstein's approach, but with an increase of $\frac{3n}{2}$ XOR gates.

- Space Complexity
$$S_{\otimes}(n) = 9S_{\otimes}(\frac{n}{4});$$
$$S_{\oplus}(n) = 9S_{\oplus}(\frac{n}{4}) + 10n - 17.$$

- Time Complexity
$$D_{\otimes}(n) = D_{\otimes}(\frac{n}{4});$$
$$D_{\oplus}(n) = D_{\oplus}(\frac{n}{4}) + 4T_X.$$

3.1.6 2^s-Way Split with Optimized Karatsuba Reconstruction

With Bernstein's reconstruction and Fan's overlap-free method, KA has been improved on space and time, respectively. Although it cannot concurrently apply these two methods together in original two-way Karatsuba formula, it can be expected that both of the space and time complexity can be improved together in an extension of two-way split KA. This is revealed by the above reviewed time optimized four-way split formula[7]. Moreover, inspired by Bernstein's idea on two-way and four-way Karatsuba formulas, an optimization of reconstruction process in s iterations of 2-term KA has been proposed by Negre [8] in 2014.

The recursive Karatsuba-based structure also can be viewed as three independent blocks as shown in figure 3.2. In recursive CPF of depth s, the operands in a higher

layer are split into two halves and then form three half size polynomials, which can be further separated later. Finally, $2 * 3^s$ terms of operands can be obtained during these recursions. Moreover, the intermediate value in each layer will be entered into CM block to generate the multiplication partial products used in R block.

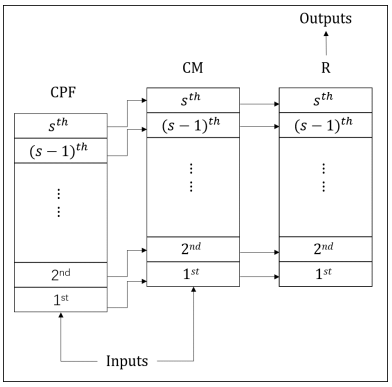

Fig. 3.2: Block Decomposition of Recursive KA

The s layers of R block is presented in a tree structure in [8]. The following graph shows the original construction tree based on reconstruction formula in two-way split KA. Each node in the higher layer can be extended into three lines. From left to right, these lines are corresponding to P_0, P_2 and P_1 in equation (3), respectively.

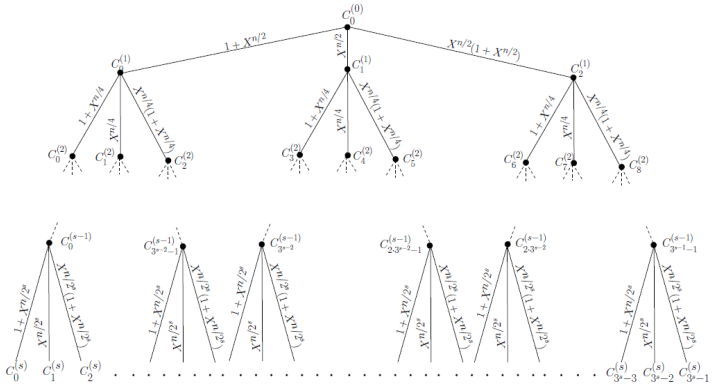

Fig. 3.3: Reconstruction Tree of Depth s Based Original Two-Way Split KA [8]

In [8], a modified reconstruction tree of depth s is proposed based on the generalization of Bernstein's four-way split reconstruction formula (12) and an algorithm,

named a generalized Bernstein's reconstruction(GBR_s), has been proposed to construct this three structure. The modified tree structure is shown in figure 3.4.

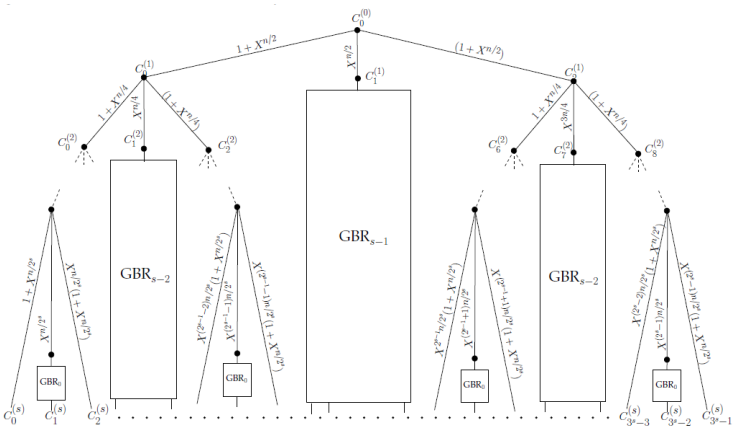

Fig. 3.4: Modified Reconstruction Tree of Depth s Based on GBR_s [8]

Based on above structure, the following recursive complexity formula can be derived.

- Space Complexity

$$S_{\otimes}(2^s n) = 3^s S_{\otimes}(n);$$
$$S_{\oplus}(2^s n) = 3^s S_{\oplus}(n) + (\frac{13 \cdot 3^s - 12 \cdot 2^s - 1}{2})n - \frac{5 \cdot 3^s - 5 + 2s}{4}.$$

- Time Complexity

$$D_{\otimes}(2^s n) = D_{\otimes}(n);$$
$$D_{\oplus}(2^s n) = D_{\oplus}(n) + (2s+1)T_X.$$

When $s = 1$ or 2, the above complexity is consistent with Bernstein's two-way and four-way split formula. With the increase of s, the space and time complexity will continuously reduce until $s = \log_2 n$ and the results are better than those of other existing 2-way split methods. Moreover, a combination of this work of $s = \log_2 n - 2$ depths and classical method for building 4-bit sub-multiplication will achieve a lowest complexity for $GF(2^{2^m})$ compared with existing works [4], [5] [6], [7], [35], [36], [37].

The following table shows the complexity achieved in [8].

Table 3.3: Complexity of 2^s-Way Split Structure with Optimized Karatsuba Reconstruction

	# AND	# XOR	Delay
$s = \log_2 n$	$n^{\log_2 3}$	$5.25n^{\log_2 3} - 6n^{\log_2 3} + 0.75$	$(2\log_2 n + 1)T_X + T_A$
$s = \log_2 n - 2$ with classical method	$1.78n^{\log_2 3}$	$3.75n^{\log_2 3} - 6n + 0.25 - 0.5\log_2 n$	$(2\log_2 n - 1)T_X + T_A$

3.1.7 k-term Karatsuba-Like Formula

Since KA and three-way split method are more appropriate to form subquadratic multiplication where n is a power of 2 or 3, an extension class of k-term KA has been discussed in many papers for practical cryptography applications, where $k = 5, 6, 7, \cdots$. The construction of subquadratic multiplications based on the recursive Karatsuba series formulas can also be summarized as three blocks..

1. CPF: Splitting n-term polynomials into k parts and forming components used in the scalar multiplication with corresponding parts;

2. CM: Computing the sub-multiplications recursively;

3. R: Constructing the final result with sub-multiplication products based on the formula;

Assuming that $M(k)$ represents the minimum scalar multiplication required for a CM block, many works focus on improving the upper bound of $M(k)$, which can significantly reduce the complexity. In [17], detailed analysis of classical KA are presented for polynomial of arbitrary size and a generalized formula has been proposed for computing two k-term polynomial multiplication by 'divide and conquer' technique. In this article, Weimerskirch and Paar also carefully studied the complexity of two n-term polynomial multiplication. A upper bound of $M(n)$ is generated up to polynomials of degree 127 using the following expression: if n is a composite number

which can be factorized into some small prime number.

$$M(n) \leqq M(k_0)M(k_1)\cdots M(k_i), \tag{13}$$

where $n = k_0k_1\cdots k_i$ and if $n = 2m + 1$, then

$$M(n) \leqq M(m) + 2M(m + 1). \tag{14}$$

In [38], Sunar has proposed a similar work of generalized subquadratic algorithm derived from Winograd short convolution algorithm, which is identical with KA but with improved efficiency in some aspects.

However, as n increases, optimizing the structure of the generalized formula becomes very complicated. Some later works focus on reducing scalar multiplications required in k-term Karatsuba formula, when k is a small integer. Then a lower bound of $M(n)$ for large number of n can be achieved by recursively using k-term Karatsuba formula. In [10], five, six and seven-term Karatsuba-like formulas has been presented by Montgomery with fewer multiplications. Montgomery's Karatsuba-like formula is the first time to reach the best result in term of $M(5), M(6), M(7)$. Although a lower $M(5)$ can be obtained by using three-way splitting with five scalar multiplications in [11], the large coefficients of $O(n)$ make it uncompetitive for the size of n considered in this thesis.

When $k \geqslant 7$, the upper bound of $M(K)$ are improved with CRT by Fan and Hasan [39]. Then a better upper bound for some $M(n)$ are presented by Cenk et al. [12]. Later, Fan et al. [13] presents more $4, 5, 6, 7, 8, 9$-term Karatsuba-like formulas based on CRT with the same number of scalar multiplications. Furthermore, these bounds for some $n \geqslant 7$ are reduced by exhaustive search method proposed by Oseledets [14]. Previous results of $M(n)$ are defined over the ground field $GF(2)$. In [11], a better upper bound of $M(n)$ over an arbitrary nontrivial ring are proposed for some n and

corresponding Karatsuba-like formula are derived based on the ring. The following table presents a summary of $M(k)$ over $GF(2)$, when $2 \leqslant k \leqslant 11$. More results of M(n) can be found in the reference shown in the table.

Table 3.4: Selected Upper Bound for M(k) Over $GF(2)$

k	M(k)	k	M(k)
2	3	7	22 [10, 39, 12, 14]
3	6	8	26 [39]
4	9	9	30 [12, 14]
5	13 [10, 12]	10	35 [39, 14]
6	17 [10]	11	39 [12, 14]

3.1.8 Subquadratic Multiplication Based on Block Recombination Approach

Another technique to reduce the asymptotic space complexity for subquadratic multiplier is block recombination. It was initially applied into TMVP based multiplier. Then Cenk et al. [7] extended this method to Karatsuba-based polynomial Multiplier. Block Recombination method is based on a structure called "Two Multiplications and Add". Specifically, it considers the problem of computing two polynomial multiplications with a same structure in parallel followed by an addition. So this method is independent of KA, TMVP or other subquadratic methods, and it can be used in any improvement on these algorithms

The "Two Multiplications and Add" architecture can be defined as

$$S = AB + A'B'.$$

A straightforward method to solve S in a parallel architecture is computing two multiplications by KA; and then add the two final multiplication products together. However, in this 'Two Multiplications and Add' structure, the computation of the reconstruction block and component addition block can be reversed. The new structure

performs the component addition of two Karatsuba expansion results first and then constructs the sum through Karatsuba formula. The following lemma is specified to reduce the space complexity of computing S by recombining two reconstruction blocks appeared in "Two Multiplications and Add" structure.

Lemma 1. *Let $R(\hat{C})$ and $R(\hat{C}')$ be the reconstruction function of two multiplications AB and $A'B'$, separately, where $\hat{C}$ and $\hat{C}'$ are vectors of $n^{\log_2 3}$ bits. Then*

$$R(\hat{C}) + R(\hat{C}') = R(\hat{C} + \hat{C}').$$

The proof of 1 is shown in appendix. Then the new architecture can be shown in the following graph.

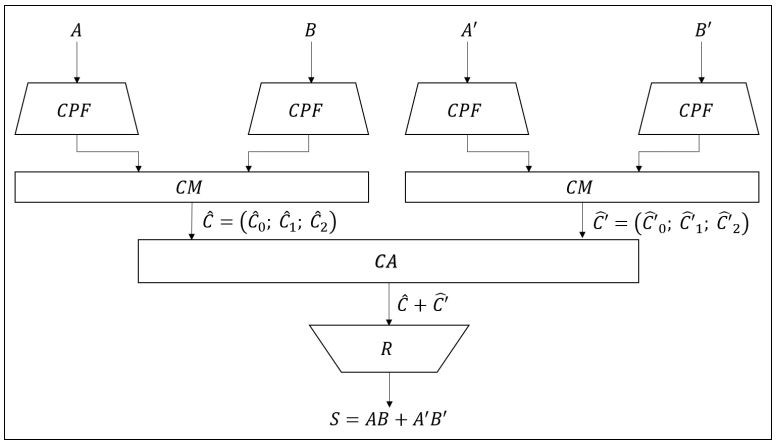

Fig. 3.5: Two Multiplications and Add Architecture Based on Block Recombination Approach

Compared to the architecture shown in figure 3.5 with straightforward method, the cost of XOR gates are increased in CA block. It is noted that CA part in the reconstructed structure cost $n^{\log_2 3}$ XOR gates to combine the vector $\hat{C} + \hat{C}'$ and the straightforward structure requires $2n - 1$ XOR gates in this block. However, one R block is saved in the new structure and the saved gates in this block is more than

the raised gates in CA block. According the complexity analysis in block decomposition, at least $\frac{5}{2}n^{\log_2 3} - 3n + \frac{1}{2}$ XOR gates will be saved in the recombined 'Two Multiplications and Add' architecture.

In order to apply the above recombined architecture into the Karatsuba multiplication, Cenk et al. [7] has expended AB with school-book method first, which is represented as following equation.

$$C = AB = A_0B_0 + A_1B_1X^n + (A_0B_1 + A_1B_0)X^{\frac{n}{2}}.$$

Obviously, $A_0B_1+A_1B_0$ can be implemented with the structure expressed in figure 3.5. The explicit non-recursive complexity formula of recombined Karatsuba Multiplier can be shown in the following table.

Table 3.5: Complexity of Recombined Karatsuba Multiplier

Recombined Formula	# AND	# XOR	Delay
KA	$\frac{4}{3}n^{\log_2 3}$	$\frac{17}{3}n^{\log_2 3} - 10n + 4$	$(3\log_2 n - 2)T_X + T_A$
Overlap-free KA	$\frac{4}{3}n^{\log_2 3}$	$\frac{17}{3}n^{\log_2 3} - 10n + 4$	$(2\log_2 n - 1)T_X + T_A$
Reconstructed KA	$\frac{4}{3}n^{\log_2 3}$	$\frac{31}{6}n^{\log_2 3} - \frac{17}{2}n + \frac{5}{2}$	$(3\log_2 n - 2)T_X + T_A$

Compared to the complexity before and after applying block recombination approach, it is noted that number of increased AND gates is less than that of the decreased XOR gates. Furthermore, Cenk [7] et al. indicated that one two-input XOR gate is twice as large as one two-input AND gate when the ASIC environment is considered. So the overall space complexity of recombined Karatsuba multiplier is reduced.

In [37], a new block recombination has been proposed for overlap-free KA and both of these two block recombination approaches can be applied together. Supposing that AB is expended by the school-book method accompanied with overlap-free split approach.

$$C = AB = A_0B_0 + A_1B_1X^2 + (A_0B_1 + A_1B_0)X.$$

Then the new block recombination is used to compute $A_0B_0 + A_1B_1X^2$ by further decomposing the results from CM blocks. And with the increase in the number of decomposition layers, more XOR gates will be saved. Let the number of decomposed recursions be denoted as t. The final complexity results can be represented as:

Table 3.6: Complexity of New Recombined Karatsuba Multiplier

# AND	# XOR	Delay
$\frac{4}{3}n^{\log_2 3}$	$\frac{14}{3}n^{\log_2 3} + \frac{2^t}{3^t}n^{\log_2 3} - 10n + 2^{t+1} + 2$	$(2\log_2 n + (t-1))T_X + T_A$

When $t = 1$ and $t = m - 1$, the space complexity results are better than the block recombination based overlap-free KA and reconstructed KA, respectively. The reviewed methods in this section also can be extended to three-way and four-way split formula to obtain better complexity results. A comprehensive complexity results for KA with block recombination can be found in [37].

3.1.9 Subquadratic Multipliers Using Mixed Methods

It has been shown that classical multiplications are more effective than subquadratic multiplications within a certain range. Then some new parallel multipliers (or called hybrid structure in some literatures) in $GF(2^n)$ have been proposed by combining KAs and classical method to gain an improved complexity.

In [18], a new Karatsuba-based multiplier was presented for $GF(2^{233})$. For computing polynomial multiplication with 233 bits, it pads seven zeros at the most significant bits of the operands and factorizes the 240 bits into $2 * 3 * 40$. This multiplier consists of three 80 bits sub-multipliers which is serially used twice to complete 6 scalar multiplication in three-way split KA. And the 80 bits multiplier is built with three traditional multiplier of size 40 by two-way split KA. In 2005, a similar 240 bits multiplier, factorized as $240 = 2 * 2 * 2 * 30$, was proposed [19]. It combines two-way split KA and school-book method. And in [20], two Karatsuba-based subquadratic multipliers have been proposed for $GF(2^n)$ in the case of $n = 2^i m$ and $n = 2^i + d$,

respectively, where i, d, m are integers.

In [17], a summary table has been presented for the cost of polynomial multiplication up to 128 bits by Weimerskirch and Paar. The results shown in this table contain the number of required multiplication and addition operations (in terms of binary extension fields, they are equivalent to XOR and AND operations, respectively), and the way of decomposition for different values of n. As mentioned in the section 3.1.7, a generalization of Karatsuba formula has been proposed in this paper. It also indicates factorizing n into small numbers can achieve improved efficiency, and for the large prime number, a decomposition can be made after padding appropriate number of zeros at the most significant bits of the operands. In addition, the authors also studied a better combination sequence when multiple Karatsuba formulas are used.

Later, a new method for constructing polynomial multiplier in $GF(2^n)$ has been proposed in [15] with reduced space complexity and power consumption. In order to achieve such results, two algorithms are introduced in this article. One is to examine Karatsuba-like formulas with optimized reconstruction sequence. Another algorithm focuses on the optimal order of combing the Karatsuba-like formulas used in the new architecture. Improvements of $5, 6, 7$-term Karatsuba-like formulas are presented in this paper with corresponding recursive space complexity formulas. New multipliers for NIST recommended fields are built by combining $2, 3, 4, 5, 6, 7$-term KAs and classical method with an optimal process sequence. Although the time complexity of these multipliers are not shown in the result table, their space complexity are better than the methods reviewed in above.

In [21], a new padding algorithm is proposed for subquadratic multiplications. With an application of this algorithm and Bernstein's reconstructed two-way split KA, efficient multiplication structures are raised for NIST fields. The space complexity of these multipliers are a little worse than the results shown in [15]. An updated

padding method used in our proposed work is derived from this paper.

To our best knowledge, the current best bounds (denoted as $S(n)$) of the bit operations (XOR and AND) required to multiply two n-term polynomials are kept in [4], [56] and [22]. Similar with other works reviewed in this section, combinational techniques are utilized by these two research groups to form subquadratic multipliers for $GF(2^n)$. In these papers, $S(n)$ has a significant improvement due to Bernstein's three-way split KA with five scalar multiplications [4], its improved version [22], and the improved 5-way split method [22]. The main contribution of [56] is the improved bound of small n, which provides an efficient foundation for building large multipliers. Most of the optimized results of $S(n)$ can be found in table 2 in [22] and some new results are presented in [56].

In this section, exiting works for constructing subquadratic multipliers with a combination of multiple methods have been reviewed. Although the asymptotic space complexity of these multipliers are not improved in these papers, their proposed multipliers are more efficient for current cryptographic systems.

3.2 Subquadratic Space Complexity Multiplication Based on Other Representations

The representation of elements in $GF(2^n)$ plays an important role in the design of multipliers. All of the above methods used to form subquadratic space complexity multiplication are referred to polynomial based multiplication. Although most works related to subquadratic multiplier are based on the polynomial representation due to its simplicity and flexibility, there are some other bases that can be applied with improved efficiency for some applications.

3.2.1 Toeplitz Matrix-Vector Product

In 2007, a novel parallel multiplication with subquadratic space complexity is proposed by Fan and Hasan [35]. This new multiplier takes the advantage of Toeplitz matrix-vector products which also can be accelerated by 'divide and conquer' scheme. And both of its asymptotic space and time complexity match the best results in above three version of two-way split KA. Moreover, this approach can be extended in dual or weakly dual or triangular bases, which is the first time to build subquadratic multiplier based on these representations.

In order to take the advantage of TMVP approach, a shifted polynomial basis and the coordinate transformation technique are applied to form the Toeplitz matrix. In [40], a Mastrovito multiplier for all trinomials is proposed based on SPB. In that paper, detailed derivations are proposed to form Mastrovito matrix based on SPB and it also provides the formula to complete basis conversion between PB and SPB.

The construction of the TMVP based subquadratic multiplier can be summarized into the following steps:

1. Form the SPB Mastrovito matrix based on corresponding irreducible trinomial
2. Convert the SPB Mastrovito matrix into Toeplitz Matrix
3. Recursively build the matrix-vector multiplication with TMVP approach
4. Convert the result from step 3 into the final result

In above procedure, step 2 and 4 are based on coordinate transformation technique which is cost-free. Step 1 only cost one XOR gate delay and at most $n-1$ XOR gates. The final complexity of subquadratic multiplier based on TMVP approach can

be summarized into the following expressions.

$$S_{\otimes}(n) = n^{\log_2 3};$$
$$S_{\oplus}(n) = 5.5n^{\log_2 3} - 5n - 0.5;$$
$$D_{\otimes}(n) = 1;$$
$$D_{\oplus}(n) = 2\log_2 n + 1.$$

The proposed multiplier is only applicable for irreducible trinomials and some special pentanomials; However, irreducible trinomials don't exist in all $GF(2^n)$, where $1 \leqslant n \leqslant 10000$ [24]. So a later work presented a TMVP based multiplier for all irreducible pentanomials [41, 42], where there is no irreducible trinomial exists. Moreover, in [43] and [44], a multiple way split formula and its improved version are presented based on TMVP approach, respectively.

Some further works also extend TVMP approach to other basis such as ONB [45, 46, 47, 48, 49], nearly all one polynomial [50], dual, weakly dual and triangular basis [51] and these multipliers provide more choices for different applications.

3.2.2 Subquadratic Multiplication Using Dickson Polynomial

Since there is almost no cost to constructing a squaring operation with finite field elements expressed on a NB, many works focus on providing effective polynomial multiplication based on this representation. When the multiplier with subquadratic space complexity has been proposed, it is desirable to design this kind of finite field multiplier over an NB. In order to further improve the efficiency, ONB (or GNB) are most considered in the design of subquadratic algorithms. However, this basis doesn't exist in some binary extension fields. In this case, Dickson polynomial is an alternative way to develop an subquadratic multiplier first proposed by Hasan and Negre [52]. And the complexity result of parallel multiplier for irreducible Dickson binomials or trinomials using TMVP approach is acceptable in today's ECC cryptographic system.

Later, a new GNB multiplier [53] are developed using Dickson-Karatsuba decomposition with improved space complexity. Moreover, there are two similar representations, called Charlier polynomials and Hermite polynomials, are presented in [54] and [55], receptively. The first one is proposed with Karatsuba formula and the second one builds subquadratic multiplier with TMVP method.

Since our work concentrates on polynomial representation, explicit complexity formulas for multipliers based on the ONB and Dickson polynomials are not shown in this paper due to their different features in practical applications. In case of comparing multipliers using these basis, the comparison should be established on the an entire ECC system.

4 Proposed Bit-Parallel Multiplication with Subquadratic Space Complexity in $GF(2^n)$

In previews chapter, multiplication algorithms with subquadratic space complexity have been reviewed as well as how to combine these methods to construct efficient subquadratic multipliers over $GF(2^n)$. Our works will continuously focus on investigating the optimized multiplication architectures with a combination of existing methods for current cryptographic purposes.

This chapter begins with an analysis of KA-based subquadratic multipliers and the general idea to construct such architecture with improved efficiency. Then a new algorithm is proposed to design optimized multiplication architectures in $GF(2^n)$, where $n \in [160, 600]$, with mixed methods. Examples of these new architectures are given later on NIST recommended fields and the corresponding complexity results are compared with current works. Finally, the idea of proposed algorithm is extended to construct subquadratic multiplier with a lager field size.

4.1 General Idea

Using Karatsuba algorithms, a multiplier with larger operands can be constructed with several smaller sub-multipliers. In a simple KA-based recursive structure, if the highest level is defined as the final results, the lowest layer should be degree one polynomial multiplications. Let A and B be polynomials with the size of n, where $n = k_1 k_2 \cdots k_m$, the following 'top-down' architecture demonstrates the construction of KA-based multiplier.

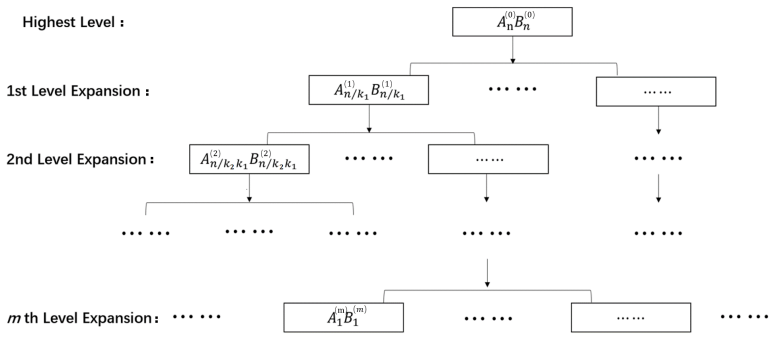

Fig. 4.1: 'Top-down' Architecture of KA

In picture 4.1, the subscript shown beside A and B is the length of the operands in that iteration and superscript means the number of layers. From the top to bottom, the multiplication of A and B is broken down into smaller multiplications and additions by k_i-term Karatsuba series formula layer by layer. In this structure, the complexity of modules in upper layer is determined by the gates required in lower layer modules and the reconstruction process.

In some case, padding algorithm is required in above KA-based structure and could result in an improved efficiency. The traditional padding strategy is to add enough zeros at the most significant bits and extend the operands to a desired length, for example: lengthen $GF(2^{2^m+k})$ to $GF(2^{2^{m+1}})$, where $k < 2^m$. Instead of supplementing all zeros at the first iteration, the padding methods proposed in [21] prolong the length of operands to a multiple of 2 at each layer. Through this approach, some redundant operations can be eliminated. Similarly, in the proposed multiplication, a suitable number of zeros will be padded at the most significant bits of the operands layer by layer.

The key idea of constructing efficient subquadratic space complexity multiplication is to find a better decomposition of n. For a given value of operands size n padding with appropriate zeros, different methods can be found to decompose the multiplication due to the variety of Karatsuba series formulas, and the independence and flexibility of the CM modules.

For example, 233 bits operands can be factorized into $2 * 3 * 3 * 13$ with padding one zero at the most significant bits. In this case, the multiplication can be built with a combination of $2, 3$-way split KAs and 13-bit classical multiplication modules. Or it can be decomposed as $2*2*2*(14+15)+1$. This architecture can be constructed with 232-bit subquadratic multiplication and classical method, where the 232-bit multiplier is built with 2-way split KA and optimized $14, 15$-bit multiplication modules.

Therefore, in terms of different Karatsuba series formulas and CM modules used to construct the multiplier, there are many KA-based architectures for a specific n. In order to find a desired complexity result among these multiplication architectures, existing Karatsuba series formulas and optimized fundamental multiplication modules are needed to be examined and selected.

4.1.1 Selection of KAs

In this section, existing Karatsuba formulas with the best complexity results will be selected to construct subquadratic multiplier in $GF(2^n)$ for composite n. Two cases are discussed when n cannot be directly factorized with selected KAs.

In chapter 3, several Karatsuba formulas are reviewed for polynomial multiplication and the current best bound of $M(n)$ for some n are mentioned in table 3.4. The asymtotic complexity of these formulas are shown in the following table.

Table 4.1: Asymtotic Complexity of $M(n)$

n	Asymtotic Complexity	n	Asymtotic Complexity
2	$O(n^{1.58})$	7	$O(n^{1.59})$
3	$O(n^{1.63})$	8	$O(n^{1.57})$
4	$O(n^{1.58})$	9	$O(n^{1.54})$
5	$O(n^{1.59})$	10	$O(n^{1.54})$
6	$O(n^{1.58})$	11	$O(n^{1.52})$

It has been shown that the asymptotic complexity will gradually decrease with an increase of n, and when n tends to infinity, the asymtotic complexity tends to linear

level [38]. Although lower asymtotic complexity means less multiplications required for multiplying two n-term polynomial, the corresponding construction process will be more complicated and result in the large coefficients of the $O(n)$, which also indicated large space complexity for practical cryptographic applications. In addition, as the value of n increasing in Karatsuba-like formulas, the structure of the formula is also intricate; concurrently, the critical path for reconstruction block will be extended.

In contrast, 2, 3, 4-way split formulas have been detailedly derived with improved efficiency. They are selected to construct $GF(2^n)$ subquadratic multipliers in proposed algorithm due to its simplicity and low complexity. A summary of these Karatsuba formulas is shown in the following table.

Table 4.2: Karatsuba Formulas Used in Proposed Work

Split	Recursive Space Complexity	Recursive Time Complexity
2 [4] & [5]	$S(2n) = 3S(n) + 7n - 3$	$D(2n) = D(n) + 3T_X$
2 [6]	$S(2n) = 3S(n) + 8n - 4$	$D(2n) = D(n) + 2T_X$
3 [7]	$S(3n) = 6S(n) + 18n - 6$	$D(3n) = D(n) + 4T_X$
3 [7]	$S(3n) = 6S(n) + 21n - 9$	$D(3n) = D(n) + 3T_X$
4 [4]	$S(4n) = 9S(n) + 34n - 11$	$D(4n) = D(n) + 5T_X$
4 [7]	$S(4n) = 9S(n) + 40n - 17$	$D(4n) = D(n) + 4T_X$
2^s [8]	$S(2^s n) = 3^s S(n) + (\frac{13 \cdot 3^s - 12 \cdot 2^s - 1}{2})n - \frac{5 \cdot 3^s - 5 + 2s}{4}$	$D(2^s n) = D(n) + (2s+1)T_X$

In the case n is not a multiple of $2, 3, 4$, the following two cases may be used to achieve a low complexity. Supposing A and B are two polynomials in the field of $GF(2^n)$, where $n = km \pm i$ and i is less than both k and m.

For the case $n = km + i$, A and B are split into $k + 1$ segments, with k terms in the size of m and one term owing i bits.

$$AB = A_1 B_1 X^{2km} + (A_0 B_1 + A_1 B_0) X^{km} + A_0 B_0 \tag{15}$$

In (15), $A_0 B_0$ can be expanded by k-way split Karatsuba-like formula and the rest parts are computed directly. Within suggested range of fields, i is selected as 1. And

the following complexity equations are given for $i = 1$.

$$S_{\otimes}(n) = S_{\otimes}(n-1) + (2n-1)$$
$$S_{\oplus}(n) = S_{\oplus}(n-1) + (2n-3)$$
$$D(n) = D(n-1) + T_X,$$

In the case of $n = km - i$, A and B are separated into k parts with one block size very close to m. Due to that the CM blocks in KA-based multiplier are independent of each other, the multiplication of the smaller block can be constructed using a different method than other m-term multiplications. Then $S(n)$ is equal to $(q-1)S(m) + S(m-i)$ plus the cost of construction part, where q is the number of scalar multiplications required in k-way split KA.

In term of the space optimized Karatsuba-series formulas used in the proposed work, exactly complexity formulas for the case $n = km - i$, where $k = 2, 3, 4$, can be shown in table 4.3. It's noted that $2i$ and $2(k-1)i$ redundant XOR operations will be removed in R block and CPF block, respectively. Additionally The results can be easily extended to time optimized KAs used in proposed algorithm.

Table 4.3: Complexity for $GF(2^{km-i})$

Case	#AND	# XOR
$n = 2m - 1$	$2S_{\otimes}(\frac{n+1}{2}) + S_{\otimes}(\frac{n-1}{2})$	$2S_{\oplus}(\frac{n+1}{2}) + S_{\oplus}(\frac{n-1}{2}) + \frac{7(n+1)}{2} - 7$
$n = 3m - i$	$5S_{\otimes}(\frac{n+i}{3}) + S_{\otimes}(\frac{n-2i}{3})$	$5S_{\oplus}(\frac{n+i}{3}) + S_{\oplus}(\frac{n-2i}{3}) + 6(n+i) - 6i - 6$
$n = 4m - i$	$8S_{\otimes}(\frac{n+i}{4}) + S_{\otimes}(\frac{n-3i}{4})$	$8S_{\oplus}(\frac{n+i}{4}) + S_{\oplus}(\frac{n-3i}{4}) + \frac{17(n+i)}{2} - 8i - 11$

4.1.2 Fundamental Multiplication Modules

Efficiency of sub-multiplications in the lowest recursion also dominate the complexity of the multiplications. For example, 4-bit classical multiplication, required 16 AND gates and 9 XOR gates, is more efficient than 4-bit Karatsuba multiplier, which costs 9 AND gates and 23 XOR gates. The following table shows a complexity comparison

of Bernstein's two-way split KA with KA Combined with Traditional Method.

Table 4.4: Comparison of Iterative KA (1) and KA Combined with Traditional Method (2)

Methods	#AND	# XOR	Time
(1) [4]	$n^{\log_2(3)}$	$5.5n^{\log_2(3)} - 7n + 1.5$	$D(n) = (3\log_2 n - 1)T_X + T_A$
(2)	$1.78n^{\log_2(3)}$	$3.94n^{\log_2(3)} - 7n + 1.5$	$D(n) = (3\log_2 n - 4)T_X + T_A$

It can be concluded that an efficient sub-multiplication module in the lowest layer can lead to an improved complexity in KA-based multipliers.

Table 4.5 presents the optimized multipliers in $GF(2^n)$, where $n \leq 15$. In the case of $n = 2, 3, 4, 5, 7$, results are achieved by school-book method. The complexity of $11, 12, 15$ and the corresponding architecture can be found in [56]. For the rest of n, the results can be easily obtained by a mixed method and the way of decomposition also shown in the table. These optimized multiplication modules are used in the proposed work to build subquadratic multiplications with improved complexity. Although more multiplication architectures can be found in [56] and [22] for larger n, they are not used in the proposed work due to its complicated structure and long critical path.

Table 4.5: Optimized Multiplier Modules in $GF(2^n)$, where $n \leq 15$

n	$S_\otimes(n)$	$S_\oplus(n)$	$D(n)$	Decomposition
2	4	1	$T_A + T_X$	2
3	9	4	$T_A + 2T_X$	3
4	16	9	$T_A + 2T_X$	4
5	25	16	$T_A + 3T_X$	5
6	27	30	$T_A + 5T_X$	$2 * 3$
7	49	36	$T_A + 3T_X$	7
8	48	52	$T_A + 6T_X$	$2 * 4$
9	54	72	$T_A + 6T_X$	$3 * 3$
10	75	80	$T_A + 6T_X$	$2 * 5$
11	78	108	$T_A + 7T_X$	11 [56]
12	81	126	$T_A + 7T_X$	12 [56]
13	106	149	$T_A + 8T_X$	$12 + 1$
14	147	154	$T_A + 6T_X$	$2 * 7$
15	140	172	$T_A + 9T_X$	15 [56]

4.2 Proposed Algorithm for Designing Subquadratic Multipliers

By combining above formulas and methods, a new algorithm 4.1 is proposed to explore an optimal combination of above approaches for constructing efficient multiplication architectures for $GF(2^n)$ where n is in the interval of $[16, 600]$.

Algorithm 4.1 The Proposed Algorithm to Design Efficient Multiplication Architectures in $GF(2^n)$

Input:

The set of required XOR gates for n shown in table 4.5: $S_{\oplus}(n)$;
The set of required AND gates for n shown in table 4.5: $S_{\otimes}(n)$;
The set of time delay for n shown in table 4.5: $D(n)$;
The set of $k(n)$ denotes the way of decomposition;

Output:

$S_{\oplus}(n)$, $S_{\otimes}(n)$, $D(n)$, $k(n)$ for $n \in [16, 600]$;

1: **for** $n = 16$ to $n = 600$ **do**
2: **for** $s = 9$ to $s = 3$ **do**
3: **if** $n = 2^s$ **then**
4: $Tem0\_S_{\oplus}(n) = 3.75n^{log_2 3} - 6n + 0.25 - 0.5log_2 n$;
5: $Tem0\_S_{\otimes}(n) = \frac{16}{9}n^{log_2 3}$;
6: $D0(n) = T_A + (2log_2 n - 1)T_X$;
7: $k0(n) = s.1$;
8: **else if** $n \ mod \ 2^s = 0$ **and** $n \neq 2^s$ **then**
9: $Tem0\_S_{\oplus}(n) = 3^s S_{\oplus}(\frac{n}{2^s}) + (\frac{13 \cdot 3^s - 12 \cdot 2^s - 1}{2})\frac{n}{2^s} - \frac{5 \cdot 3^s - 5 + 2s}{4}$;
10: $Tem0\_S_{\otimes}(n) = 3^s S_{\otimes}(\frac{n}{2^s})$;
11: $D0(n) = D(\frac{n}{2^s}) + (2s + 1)T_X$;
12: $k0(n) = s.\frac{n}{2^s}$;
13: Break;
14: **end if**
15: **end for**

16: $Tem1\_S_{\oplus}(n) = S_{\oplus}(n - 1) + (2n - 3)$;
17: $Tem1\_S_{\otimes}(n) = S_{\otimes}(n - 1) + (2n - 1)$;
18: $D1(n) = D(n - 1) + T_X$;
19: $k1(n) = 1.(n - 1)$;

20: **if** $n \ mod \ 2 = 0$ **then**
21: $Tem2\_S_{\oplus}(n) = 3S_{\oplus}(\frac{n}{2}) + \frac{7n}{2} - 3;$
22: $Tem2\_S_{\otimes}(n) = 3S_{\otimes}(\frac{n}{2});$
23: $D2(n) = D(\frac{n}{2}) + 3T_X;$
24: $k2(n) = 2.(\frac{n}{2});$
25: **else**
26: $Tem2\_S_{\oplus}(n) = 2S_{\oplus}(\frac{n+1}{2}) + S_{\oplus}(\frac{n-1}{2}) + \frac{7n}{2} - \frac{7}{2};$
27: $Tem2\_S_{\otimes}(n) = 2S_{\otimes}(\frac{n+1}{2}) + S_{\otimes}(\frac{n-1}{2});$
28: $D2(n) = \max\{D(\frac{n+1}{2}), D(\frac{n-1}{2})\} + 3T_X;$
29: $k2(n) = 2.(\frac{n+1}{2});$
30: **end if**

31: **if** $n \ mod \ 3 = 0$ **then**
32: $Tem3\_S_{\oplus}(n) = 6S_{\oplus}(\frac{n}{3}) + 6n - 6;$
33: $Tem3\_S_{\otimes}(n) = 6S_{\otimes}(\frac{n}{3});$
34: $D3(n) = D(\frac{n}{3}) + 4T_X;$
35: $k3(n) = 3.(\frac{n}{3});$
36: **else**
37: $i = 3\lfloor\frac{n}{3}\rfloor - n;$
38: $Tem3\_S_{\oplus}(n) = 5S_{\oplus}(\frac{n+i}{3}) + S_{\oplus}(\frac{n-2i}{3}) + 6n - 6;$
39: $Tem3\_S_{\otimes}(n) = 5S_{\otimes}(\frac{n+i}{3}) + S_{\otimes}(\frac{n-2i}{3});$
40: $D3(n) = \max\{D(\frac{n+i}{3}), D(\frac{n-2i}{3})\} + 4T_X;$
41: $k3(n) = 3.(\frac{n+i}{3});$
42: **end if**

43: **if** $n \ mod \ 4 = 0$ **then**
44: $Tem4\_S_{\oplus}(n) = 9S_{\oplus}(\frac{n}{4}) + \frac{17n}{2} - 11;$
45: $Tem4\_S_{\otimes}(n) = 9S_{\otimes}(\frac{n}{4});$
46: $D4(n) = D(\frac{n}{4}) + 5T_X;$
47: $k4(n) = 4.(\frac{n}{4});$
48: **else**
49: $i = 4\lfloor\frac{n}{4}\rfloor - n;$
50: $Tem4\_S_{\oplus}(n) = 8S_{\oplus}(\frac{n+i}{4}) + S_{\oplus}(\frac{n-3i}{4}) + \frac{17n}{2} + \frac{i}{2} - 11;$
51: $Tem4\_S_{\otimes}(n) = 8S_{\otimes}(\frac{n+i}{4}) + S_{\otimes}(\frac{n-3i}{4});$
52: $D4(n) = \max\{D(\frac{n+i}{4}), D(\frac{n-3i}{4})\} + 5T_X;$
53: $k4(n) = 4.(\frac{n+i}{4});$

54: **end if**

55: Compare above five sets of results based on overall space complexity, and Assign the best one to $S_{\oplus}(n)$, $S_{\otimes}(n)$, $D(n)$, $k(n)$;
56: **for** $l = (n-1)$ to 1 **do**
57: **if** The overall space complexity of $S(n)$ is smaller than $S(n-1)$ **then**
58: Assign the results corresponding with n to $S_{\oplus}(n-1)$, $S_{\otimes}(n-1)$, $D(n-1)$, $k(n-1)$;
59: **else**
60: Break;
61: **end if**
62: **end for**
63: **return** $S_{\oplus}(n)$, $S_{\otimes}(n)$, $D(n)$, $k(n)$;
64: **end for**

In order to obtain a KA-based multiplier with improved efficiency, multiplications in every intermediate layer should be optimized. Therefore, the algorithm should start with designing optimized 16-bit multiplication, and then build larger multiplier with existing multiplier with small operands. Depending on the value of n, different methods are compared to obtain a best result by applying some constraints. From line 2 to 15, the condition of $n = 2^s m$ are checked for some n with optimized 2^s-way Karatsuba reconstruction. Then the following four lines compute the first situation shown in section 4.1.1. From line 20 to 30, 31 to 42 and 43 to 54, reconstructed $2, 3, 4$-way are used to decompose n, and the results are calculated based on the above reviewed formulas. In the last part, all of results are compared based on overall space complexity in algorithm 4.1. Here, the overall space complexity is approximately estimated by the following expression: $1.5S_{\oplus} + S_{\otimes}$, where the size of 2-input XOR gate is considered as 1.5 times of 2-input AND gate. The coefficient 1.5 can be found in related article [48] on finite field multiplier and the corresponding data comes from NanGate's Library Creator [57].

It is noted that the KAs used in above algorithm is space optimized. Moreover, the Karatsuba formulas based on overlap-free approach also can be used in this algorithm

to achieve a time efficient structure.

4.3 Proposed Multipliers in NIST Recommended Fields

In this section, the proposed work is applied to build efficient polynomial multiplication over NIST recommended fields $GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$, $GF(2^{571})$ and detailed derivation for the complexity will also be presented. At the end, the complexity results will be compared with exiting works.

4.3.1 Multiplier in $GF(2^{163})$

Consider that $A(X)$ and $B(X)$ are polynomials with the degree of 162. By using the 'Dividing and Conquer' techniques, the multiplication of $A(X)B(X)$ can be decomposed as:

1. 163 bits multiplication with Bernstein's 4-way split formula: $163 = 40 + 41 + 41 + 41$;
2. 41 bits multiplication with the method shown in section(4.1.1): $41 = 1 + 40$;
3. 40 bits multiplication with optimized 2^n-way split KA: $40 = 8 * 5$;
4. 5 bits classical multiplication;

Then the above structure can be seen as the diagram 4.2. At the lowest layer of the multiplication, totally 243 5-bit classical multiplier modules are formed with corresponding coefficients based on the expansion of Karatsuba formula. These multiplication results are then reconstructed into nine 40-bit multipliers with 4-way split KA. In the next step, 8 of the 40-bit multipliers are extended to 41-bit multipliers. And these 81-bit outputs are used to construct final results with the rest one 40-bit multiplication.

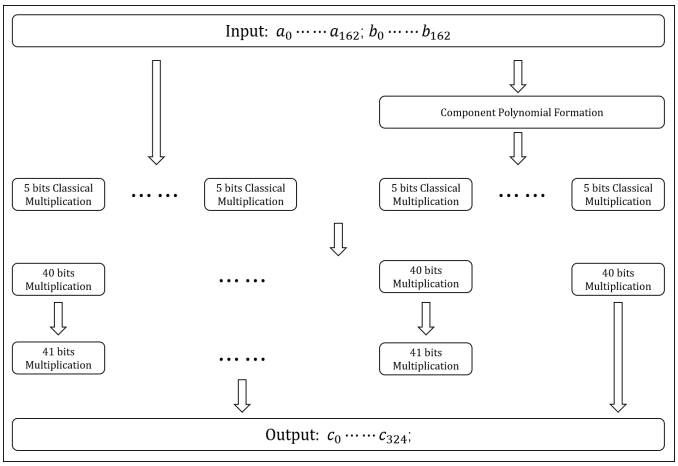

Fig. 4.2: Proposed Structure of $GF(2^{163})$ Multiplication Based on Combinational Methods

With the recursively complexity formulas shown in above sections, the detailed computation steps and final results can be presented as:

$$\begin{aligned}
S_{\oplus}(163) &= 8S_{\oplus}(41) + S_{\oplus}(40) + \frac{17 * 164}{2} - 8 * 1 - 11 \\
&= 8(S_{\oplus}(40) + 41 * 2 - 3) + S_{\oplus}(40) + 1375 \\
&= 9(27S_{\oplus}(5) + \frac{127 * 40}{8} - 34) + 2007 \\
&= 11304; \\
S_{\otimes}(163) &= 8S_{\otimes}(41) + S_{\otimes}(40) \\
&= 8(S_{\otimes}(40) + 41 * 2 - 1) + S_{\otimes}(40) \\
&= 243S_{\oplus}(5) + 648 = 6723; \\
D(163) &= D(41) + 5T_X = D(40) + 6T_X = D(5) + 13T_X = T_A + 16T_X;
\end{aligned}$$

4.3.2 Multiplier in $GF(2^{233})$

In this section, polynomial multiplication are consider over $GF(2^{233})$. Let $A(X)$ and $B(X)$ be polynomials with the degree of 232. By using proposed combinational techniques, firstly, the multiplication of $A(X)B(X)$ can be decomposed as:

1. 233 bits multiplication with Bernstein's 4-way split formula: $233 = 56 + 59 +$

$59 + 59$;

2. 59 bits multiplication with Bernstein's 4-way split formula: $59 = 14 + 15 + 15 + 15$;

3. 56 bits multiplication with optimized 2^n-way split KA: $56 = 8 * 7$;

4. 15 bits multiplication with the architecture shown in [56];

5. 14 bits multiplication with Bernstein's 2-way split formula: $14 = 2 * 7$;

6. 7 bits classical multiplication;

Then the above structure can be seen as the diagram 4.3. At the lowest layer of the multiplication, totally 51 7-bit classical multiplier modules are formed with corresponding coefficients and 64 15-bit multipliers are built with the structure presented in [56]. Then 27 of 7-bit multipliers are used to construct one 56-bit multiplier and the rest results are reconstructed into eight 14-bit multipliers, which can be combined with 15-bit multipliers to complete eight 59-bit multiplication. Finally, one 58-bit and eight 59-bit multiplications are utilized to construct the last output.

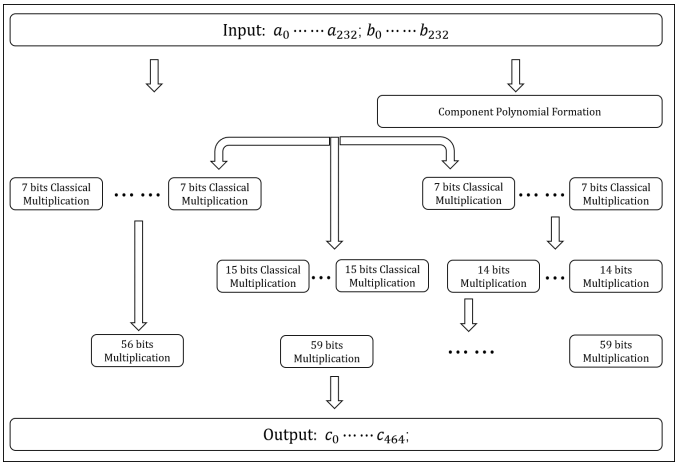

Fig. 4.3: Proposed Structure of $GF(2^{233})$ Multiplication Based on Combinational Methods

The detailed computation steps and final results is shown as following:

$$\begin{aligned}
S_{\oplus}(233) &= 8S_{\oplus}(59) + S_{\oplus}(56) + \frac{17*236}{2} - 8*3 - 11 \\
&= 8(8S_{\oplus}(15) + S_{\oplus}(14) + \frac{17*60}{2} - 8*1 - 11) + S_{\oplus}(56) + 1971 \\
&= 64S_{\oplus}(15) + 51S_{\oplus}(7) + 7122 \\
&= 19966; \\
S_{\otimes}(233) &= 8 * S_{\otimes}(59) + S_{\otimes}(56) = 64 * S_{\oplus}(15) + 51S_{\otimes}(7) = 11459; \\
D(233) &= D(59) + 5T_X = D(15) + 10T_X = D(15) + 10T_X = T_A + 19T_X;
\end{aligned}$$

4.3.3 Multiplier in $GF(2^{283})$

In this section, proposed $GF(2^{283})$ multiplier will be talked in detail. Let $A(X)$ and $B(X)$ be polynomials with the degree of 282. Firstly, the multiplication of $A(X)B(X)$ can be decomposed as:

1. 283 bits operands are extended to 288 bits, then with optimized 2^n-way split KA: $288 = 2^5 * 9$
2. 9 bits multiplication with 3-way split KA in [7]: $9 = 3 * 3$;
3. 3 bits classical multiplication;

The structure of proposed $GF(2^{283})$ multiplier is straightforward. At the lowest layer, totally 1458 3-bit classical multiplier modules are formed with corresponding coefficients based on the expansion of the Karatsuba formula. Then these multiplication results are used to construct 243 9-bit multipliers. After that, final results can be obtained based on the architecture of five layers of 2-way split KA with optimized reconstruction process. The structure of this multiplier can be represented in the diagram 4.4.

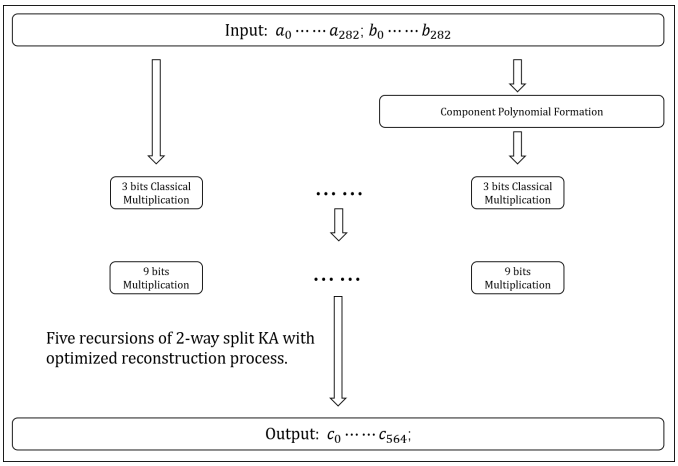

Fig. 4.4: Proposed Structure of $GF(2^{283})$ Multiplication Based on Combinational Methods

The detailed computation steps and final results is shown as following:

$$S_{\oplus}(283) = 243S_{\oplus}(9) + \frac{1387 * 288}{32} - 305 = 29674;$$
$$S_{\otimes}(283) = 243 * S_{\otimes}(9) = 13122;$$
$$D(233) = D(9) + 11T_X = T_A + 17T_X;$$

4.3.4 Multiplier in $GF(2^{409})$

This section will discuss the proposed $GF(2^{409})$ multiplier. Let $A(X)$ and $B(X)$ be polynomials with the degree of 408. Firstly, the multiplication of $A(X)B(X)$ can be decomposed as:

1. 409 bits multiplication with Bernstein's 2-way split formula: $409 = 204 + 205$;
2. 205 bits multiplication with Bernstein's 4-way split formula: $205 = 49 + 52 + 52 + 52$;
3. 204 bits multiplication with Bernstein's 4-way split formula: $204 = 51 + 51 + 51 + 51$;
4. 52 bits multiplication with Bernstein's 4-way split formula: $52 = 13 + 13 + 13 +$

13;

5. 51 bits multiplication with Bernstein's 4-way split formula: $51 = 12 + 13 + 13 + 13$;
6. 49 bits multiplication with the method shown in section(4.1.1): $49 = 48 + 1$;
7. 48 bits multiplication with optimized 2^n-way split KA: $48 = 16 * 3$;
8. 13 bits multiplication with the method shown in section(4.1.1): $13 = 12 + 1$;
9. 12 bits multiplication with the architecture shown in [56];
10. 3 bits classical multiplication;

In above structure, the lowest layer consists of two small multiplier modules. One is 3-bit multiplier which later is used to build as 48 bits multiplier. Another one is 12-bit multiplication formed with the architecture shown in [56]. Then 12-bit multiplication is extend to 13-bit and these two module are used to construct the results of 51 and 52-bit multiplication. At the same time, the results of 48-bit multiplication are also extended to the length of 49-bit with some input bits. In the next recursion, two 49-bit multipliers are reconstructed with corresponding 16 52-bits multiplier to obtained two 205 bits multiplier. And one 204-bit multiplication is built with nine 51-bit multiplication. Finally, these three results are used to construct the output of 409-bit multiplication by applying two-way split KA. The diagram of above process is shown in 4.5.

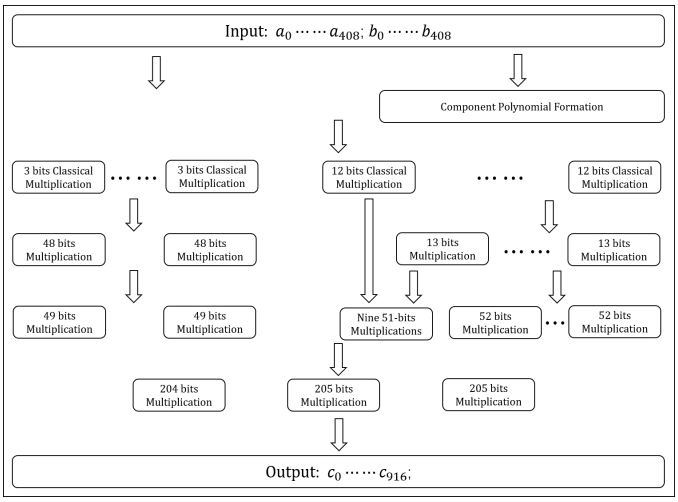

Fig. 4.5: Proposed Structure of $GF(2^{409})$ Multiplication Based on Combinational Methods

The detailed computation steps and final results is shown as following:

$$\begin{aligned}
S_{\oplus}(409) &= 2S_{\oplus}(205) + S_{\oplus}(204) + \frac{7*410}{2} - 4 - 3 \\
&= 2(8S_{\oplus}(52) + S_{\oplus}(49)) + 9S_{\oplus}(51) + 6617 \\
&= 216S \oplus (13) + 2S_{\oplus}(48)) + 9S_{\oplus}(12) + 17510 \\
&= 225S \oplus (12) + 162S_{\oplus}(3) + 24854 = 53852; \\
S_{\otimes}(409) &= 2 * S_{\otimes}(205) + S_{\otimes}(204) \\
&= 16S_{\oplus}(52) + 2S_{\oplus}(49) + 9S_{\oplus}(51); \\
&= 216S_{\oplus}(13) + 2S_{\oplus}(48) + 9S_{\oplus}(12) + 194; \\
&= 225S_{\oplus}(12) + 162S_{\oplus}(3) + 5594 = 25277; \\
D(409) &= D(205) + 3T_X = D(52) + 8T_X = D(13) + 13T_X \\
&= D(12) + 14T_X = T_A + 21T_X;
\end{aligned}$$

4.3.5 Multiplier in $GF(2^{571})$

The structure of proposed $GF(2^{571})$ multiplier is similar to the proposed multiplier over $GF(2^{283})$. Let $A(X)$ and $B(X)$ be polynomials with the degree of 570. Firstly,

the multiplication of $A(X)B(X)$ can be decomposed as:

1. 571 bits operands are extended to 576 bits, then with optimized 2^n-way split KA: $576 = 2^6 * 9$

2. 9 bits multiplication with 3-way split KA in [7]: $9 = 3 * 3$;

3. 3 bits classical multiplication;

Compared with the $GF(2^{283})$ multiplier discussed above, the proposed $GF(2^{571})$ multiplier has one more layer of two-way split KA. At the lowest layer, totally 4374 3-bit classical multiplier modules are formed with corresponding coefficients based on the expansion of the Karatsuba formula. Then 729 9-bit multipliers are constructed with these multiplication results. After that, in the light of the architecture of six layers of 2-way split KA with optimized reconstruction process, final outputs can be obtained. The structure of this multiplier is shown in the diagram 4.6.

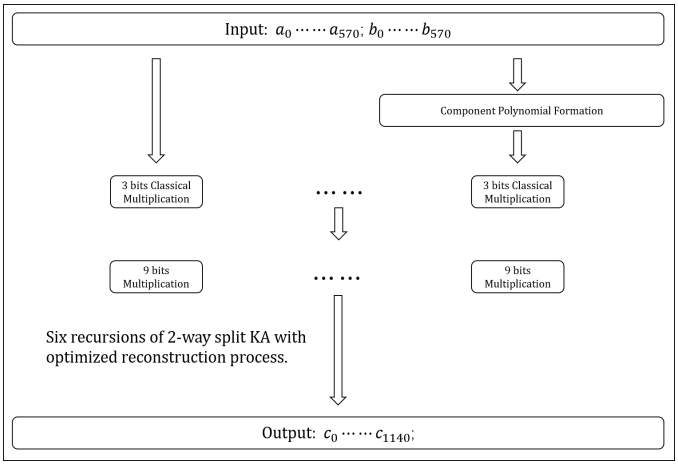

Fig. 4.6: Proposed Structure of $GF(2^{571})$ Multiplication Based on Combinational Methods

The detailed computation steps and final results is shown as following:

$$S_{\oplus}(571) = 729S_{\oplus}(9) + \frac{2177 * 576}{32} - 913 = 90761;$$
$$S_{\otimes}(571) = 729 * S_{\otimes}(9) = 39366;$$
$$D(571) = D(9) + 13T_X = T_A + 19T_X;$$

4.4 Time Efficient Multiplication Architectures for NIST Recommended fields

As mentioned before, if the overlap-free KA in table 4.1 is used in the proposed algorithm, another fast structures can be found with increased space complexity. In this section, the decomposition route will be introduced for these speed improved multiplications.

The following graph shows how to divide the 163-bit multiplication into sub-multiplications. First, the final result can be constructed with two 82-bit and one 81-bit multiplication. Then these multiplications can be further split with four-way split method. In the final recursion, 21-bit multiplier is built with six 7-bit classical multipliers. And 3, 4, 5-bit modules are used to form 19-bit and 18-bit multipliers.

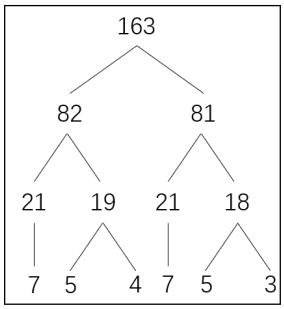

Fig. 4.7: Time Efficient Decomposition of $GF(2^{163})$ Multiplication

The complexity results of architecture in 4.7 can be computed by similar steps demonstrated in above sections, with replaced Karatsuba formulas based on overlap-free approach. In addition, the time efficient structure for the rest fields are presented

in the following graph and the complexity results will be presented in the next section.

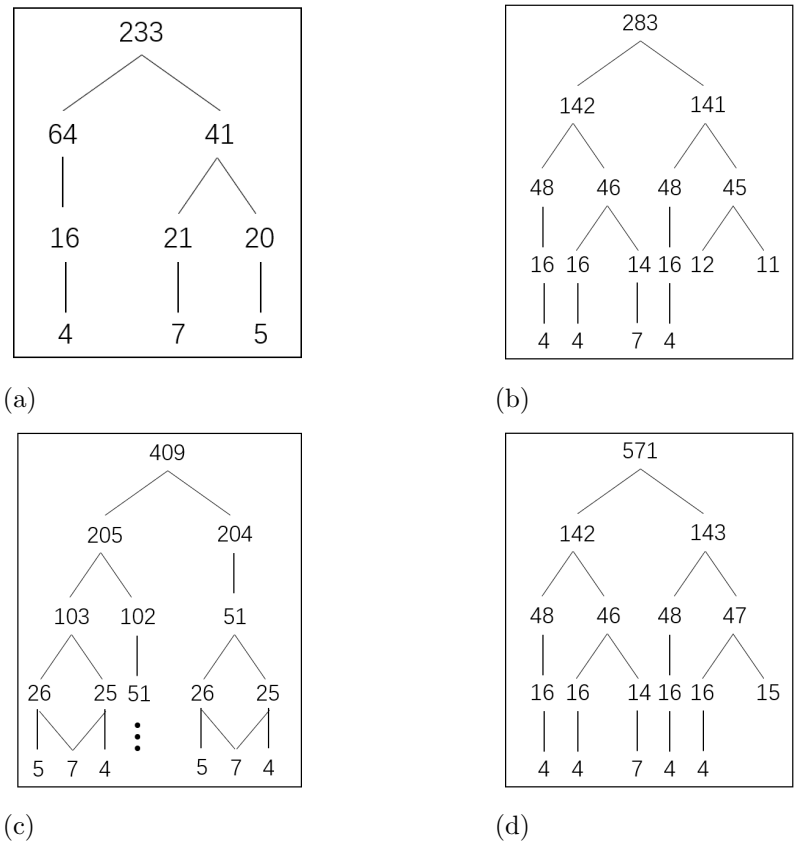

Fig. 4.8: Time Efficient Decomposition of $GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$, $GF(2^{571})$ Multiplication

4.5 Complexity Comparison

Table 4.6: Comparison of Existing Works on $GF(2^{163})$ Multiplier

Multiplier	$S_{\otimes}$ (#AND)	$S_{\oplus}$ (#XOR)	C $(1.5S_{\oplus}+S_{\otimes})$	C (%)	T (Latency)	T (%)	$C \times T$ (%)
Classical	26569	26244	65935	290 %	$T_A + 8T_X$	69 %	169 %
[21],2011	9801	11997	27796.5	122 %	$T_A + 13T_X$	108 %	111 %
[15],2012	7938	11751	25564.5	113 %	$T_A + 15T_X$	123 %	117 %
[22],2015	5052	11776	22716	100 %	$T_A + 194T_X$	1500 %	1264 %
[22],2015	5916	12003	23920.5	105 %	$T_A + 24T_X$	192 %	171 %
Proposed-1	6723	11304	23679	104 %	$T_A + 16T_X$	131 %	115 %
Proposed-2	7938	12676	26952	119 %	$T_A + 12T_X$	100 %	100 %

Table 4.7: Comparison of Existing Works on $GF(2^{233})$ Multiplier

Multiplier	$S_\otimes$ (#AND)	$S_\oplus$ (#XOR)	C $(1.5S_\oplus + S_\otimes)$	C (%)	T (Latency)	T (%)	$C \times T$ (%)
Classical	54289	53824	135025	344 %	$T_A + 8T_X$	60 %	159 %
[21],2011	11664	23589	47047.5	120 %	$T_A + 20T_X$	140 %	129 %
[15],2012	12150	21066	43749	111 %	$T_A + 17T_X$	127 %	103 %
[22],2015	8955	20201	39256.5	100 %	$T_A + 273T_X$	1827 %	1408 %
[22],2015	8804	22577	42669.5	105 %	$T_A + 42T_X$	287 %	240 %
Proposed-1	11459	19966	41408	105 %	$T_A + 19T_X$	133 %	108 %
Proposed-2	11667	26184	50943	130 %	$T_A + 14T_X$	100 %	100 %

Table 4.8: Comparison of Existing Works on $GF(2^{283})$ Multiplier

Multiplier	$S_\otimes$ (#AND)	$S_\oplus$ (#XOR)	C $(1.5S_\oplus + S_\otimes)$	C (%)	T (Latency)	T (%)	$C \times T$ (%)
Classical	80089	79524	199375	382 %	$T_A + 9T_X$	67 %	200 %
[21],2011	19683	28447	62353.5	119 %	$T_A + 19T_X$	133 %	125 %
[15],2012	13122	30091	58258.5	112 %	$T_A + 19T_X$	133 %	117 %
[22],2015	10809	27623	52243.5	100 %	$T_A + 413T_X$	2760 %	2170 %
[22],2015	12111	30357	57646.5	110 %	$T_A + 44T_X$	300 %	260 %
Proposed-1	13122	29674	57633	110 %	$T_A + 17T_X$	120 %	104 %
Proposed-2	15561	33922	66444	127 %	$T_A + 14T_X$	100 %	100 %

Table 4.9: Comparison of Existing Works on $GF(2^{409})$ Multiplier

Multiplier	$S_\otimes$ (#AND)	$S_\oplus$ (#XOR)	C $(1.5S_\oplus + S_\otimes)$	C (%)	T (Latency)	T (%)	$C \times T$ (%)
Classical	167281	166464	416977	453 %	$T_A + 9T_X$	63 %	217 %
[21],2011	35721	56218	120048	131 %	$T_A + 21T_X$	138 %	138 %
[15],2012	29700	54418	111327	121 %	$T_A + (12 + Q)T_X$	>163 %	>151 %
[22],2015	17958	49326	91947	100 %	$T_A + 584T_X$	1713 %	1313 %
[22],2015	22443	53776	103107	112 %	$T_A + 45T_X$	269 %	231 %
Proposed-1	25277	53852	106055	115 %	$T_A + 21T_X$	125 %	111 %
Proposed-2	33606	57519	119884.5	130 %	$T_A + 15T_X$	100 %	100 %

* $Q > 13$ refers to the time delay of the 7-term Karatsuba-like formula presented in [15].

Table 4.10: Comparison of Existing Works on $GF(2^{571})$ Multiplier

Multiplier	$S_\otimes$ (#AND)	$S_\oplus$ (#XOR)	C $(1.5S_\oplus + S_\otimes)$	C (%)	T (Latency)	T (%)	$C \times T$ (%)
Classical	326041	324900	813391	527 %	$T_A + 10T_X$	65 %	258 %
[21],2011	59049	87424	190185	123 %	$T_A + 22T_X$	135 %	126 %
[15],2012	37179	93383	177253.5	115 %	$T_A + (15 + E)T_X$	>165 %	>143 %
[22],2015	26148	85473	154357.5	100 %	$T_A + 869T_X$	5118 %	3876 %
[22],2015	33498	92563	172342.5	112 %	$T_A + 48T_X$	288 %	244 %
Proposed-1	39366	90761	175507.5	114 %	$T_A + 19T_X$	118 %	101 %
Proposed-2	46659	104778	203826	132 %	$T_A + 16T_X$	100 %	100 %

* $E > 13$ refers to the time delay of the 6-term Karatsuba-like formula presented in [15].

In table 4.6, 4.7, 4.8, 4.9, 4.10, the time and space complexity of proposed two works are presented with the comparison of classical multiplication and existing works on multiplication with subquadratic space complexity. The results shown in these tables contain the number of required AND and XOR gates, space complexity, Latency as well as the product of space complexity and latency. The space complexity is computed $rS_{\oplus} + S_{\otimes}$ and the coefficient r is determined by the area cost of 2-input AND gate and XOR gate and it can be varied in term of using different techniques on different platforms. It is chose as 1.5 in proposed works. Since the time clock of parallel multiplication is 1, the latency of proposed multiplier is equal to time complexity. In the last column, the product of space complexity and latency, denoted as $C \times T$ is used to estimate the overall performance of the multiplier by considering both of time and space.

Among the existing works, multipliers in [21] and [15] have a better overall performance by considering the product of $C \times T$. And two multiplication architectures are presented in [22] with reduced space complexity using optimized three-way split KA with five scalar multiplications, or called Bernstein's three-way split formula. Although one of them achieves the lowest space complexity in all of five NIST recommended fields, its linear time complexity results in a high product of $C \times T$. The other one architecture in [22] has obtained a logarithm time complexity with a increase of space complexity. Its product of $C \times T$ is still higher than the works in [21] and [15] due to the lager latency.

Compared with all existing works on subquadratic multipliers,

- In $GF(2^{163})$ and $GF(2^{233})$, the proposed multiplier-2 has the lowest latency and their products of space complexity and latency are 11 % and 3 % lower than existing best result, respectively.

- In $GF(2^{283})$, $GF(2^{409})$ and $GF(2^{571})$, the proposed architecture-1 achieves a lower product of space complexity and latency with a trade-off between space

and time complexity. And the proposed architecture-2 has the lowest latency and their products of space complexity and latency are 17%, 38% and 26% lower than the existing best result.

In addition, the proposed multiplier-1 has lower space complexity than the proposed multiplier-2 in all of the five NIST fields. In the field of $GF(2^{283})$, $GF(2^{409})$ and $GF(2^{571})$, it has a lower product of space complexity and latency than existing works as well as a lower space complexity than proposed architecture-2.

4.6 Generalized Procedure for Constructing Efficient Finite Field Multiplication

When a field is used in practical cryptography, its size is usually a large prime number. And with the increasing of n, especially not limited in above proposed fields, more decomposition paths can be found with different 'divide and conquer' technologies. In order to find a better combination for a specific field, a general procedure to form subquadratic multiplication over $GF(2^n)$ is summarized in this section.

Before extending the proposed algorithm to a general construction process for multiplication, it is important to figure out the features of different construction methods. For example, the first case shown in 4.1.1 is very suited for the condition when $n = k^m + 1$ and the second case is used when n is not a multiple of the number of split blocks.

Karatsuba-series formulas is the core component of the subquadratic multiplication, and there are three important parameters to evaluate its efficiency. The first one is the asymptotic space complexity, which is determined by the number of required scalar multiplications $M(n)$. When n is close to infinity, the value of $M(n)$ is the most effective factor. However, n is not usually large enough in current cryptographic demands, such as several hundred bits. Then the cost of XOR gates in construction

process will also be considered. The following recursive complexity equation of two six-way split methods will discuss the relation between these two parameters.

$$\begin{aligned} S_{\oplus}(6n) &= 18S_{\oplus}(n) + 75n - 21; \\ S_{\oplus}(6n) &= 17S_{\oplus}(n) + 96n - 34; \end{aligned} \tag{16}$$

In above equation, the first expression is a simple combination of two-way and three-way split formula. It requires one more sub-multiplication with less cost in reconstruction part. When the cost of $S_{\oplus}(n)$ starts to be greater than $21n$, the second algorithm [15] will have a better performance on area. The last parameter is the time complexity spent on constructing the formula. Calling for the table 4.1 and the conclusion presented in [38], although the relative asymtotic space complexity is tend to decrease from two-way split formula to k-way formulas, the construction process tends to be more complex, which means the time delay will grow in number. Example also can be given based on above two 6-way split methods. The first one will cost $7T_X$ for each iteration and the second will require more than this value. Therefore, for different sets of n and specific requirement in term of space and time, different formulas need to be examined in detail.

For constructing efficient finite field multiplication over $GF(2^n)$:

- Select construction methods for different sets of n;
- Replace the formulas or add more formulas in the proposed algorithm based on the requirements;
- Change the comparison constrains in proposed algorithm for specific time and space requirement as well as the consideration of the implementation environment;
- Use the modified proposed algorithm to obtain the expected combinational approaches.

- Start from the fundamental module, and form the multiplier layer by layer.

5 Conclusions

In this chapter, we summarize the main contributions in this thesis and propose possible future research works in the related areas.

5.1 Summary of Contributions

Bit-parallel multiplication with subquadratic space complexity has been investigated in this thesis when n is within the range of practical application of elliptic curve cryptography. Our main contributions are summarized as following.

With a suitable combination of some of the following works, namely, 2, 3, 4-way split Karatsuba algorithms along with Bernstein reconstruction and overlap-free approach, efficient padding algorithm, classical method and some optimal fundamental multiplication modules under 16 bits, a new algorithm is proposed to design an improved architecture of subquadratic multiplier for a specific $GF(2^n)$, where $n \in [160, 600]$. This algorithm takes advantage of the independence of CM blocks in KAs and the construction process starts with an optimized CM block in the lower layer of the architecture. A generalization of the proposed algorithm to be suitable for larger fields is also discussed.

As the results from the proposed algorithm, two multiplication architectures have been presented for each of NIST recommended fields optimized for area complexity and time complexity, respectively. Compared with all existing works on subquadratic multipliers,

- The proposed multiplier-2 has lower latency in all of the five NIST fields;
- The proposed multiplier-2 has lower product of space complexity and latency in all of the five NIST fields;
- The proposed multiplier-1 has lower product of space complexity and latency in the NIST fields (2^{283}), (2^{409}) and $(2^5 71)$;

- The proposed multiplier-1 has lower space complexity than the proposed work-2 in all of the five NIST fields.

The proposed works have applications in ECC cryptosystems and other security system requiring finite field based computations. Since multiplication is the most expensive and dominate arithmetic operation in $GF(2^n)$, it is expected the proposed work can significantly improved efficiency for the applied ECC and other related cryptosystems. Additionally, these structures also can be used as a submodule to construct multiplication with larger operands in some applications, such as homomorphic cryptosystem and post quantum cryptography.

5.2 Future Works

Subquadratic space complexity multiplication reflects the most recent research efforts on parallel finite field multiplication. It will dramatically reduce the required gates with an increase in critical path. The further optimization of subquadratic space complexity can be quested in the following areas and aims to continuously reduce the area cost within an acceptable time delay. In the next two sections, potential works will be discussed from the gate level implementation to architecture design.

5.2.1 Further Optimization on Circuits Modules

In term of the algorithm design for subquadratic multiplication, complexity is evaluated based on two basic components: 2-input AND and XOR gates. When an algorithm is proposed at the architecture level, it is assumed that the comparison is based on same modules of logical gates and synthesized in a same implementation environment. One most straightforward improvement that can be explored on the hardware is efficient and specific design of logic gates. For a customized cell library for finite field arithmetic, inputs constrains should be updated in the proposed algorithm to obtain more efficient multiplication structures.

Additionally, it has been shown that the efficiency of sub-multiplications is one governing factor to influence the performance of KA-based multiplication. For example, in section 4.3.5, totally 4374 3-bit classical multiplier modules are required to construct proposed $GF(2^{571})$ multiplication. Improvements on the 3-bit multiplication modules will have a significant impact on proposed $GF(2^{571})$ multiplier. Therefore, it is desirable to create basic multiplication modules with shortened latency, compact area and reduced power consumption.

5.2.2 K-way Karatsuba-like Formula with Optimized Construction Process

Since Karatsuba-based multiplier is formed with fundamental modules layer by layer, the construction process is another foremost block that guaranteed its efficiency. For the larger field, the Karatsuba formulas used in the proposed algorithm may not be efficient enough. In this case, it is expected to design improved multiplication architectures using improved k-term Karatsuba-like formulas, where $k > 4$.

Many existing works related to k-term Karatsuba-like formulas only focus on minimizing the upper bound of $M(n)$ without premeditating additions. These formulas cannot be used in the proposed algorithm with improved efficiency due to its unoptimized R block. Reducing the expanse of R block in these Karatsuba-like formulas and using them in the proposed work may obtain more efficient multiplication architectures for some binary extension field. Moreover, with an increase of k, the KA-like formulas becomes more complicated. Therefore, the improved R block may not only mean a decrease in space complexity, it also implies a simple and organized structure with fast speed.

In conclusion, the further work on $GF(2^n)$ multipliers with subquadratic space complexity can be concurrently searched in efficient 'divide and conquer' technologies and specific hardware design.

REFERENCES

[1] N. I. Koblitz, *Introduction to elliptic curves and modular forms*. Springer Science & Business Media, 2012, vol. 97.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.

[3] A. A. Karatsuba, "The complexity of computations," *Proceedings of the Steklov Institute of Mathematics-Interperiodica Translation*, vol. 211, pp. 169–183, 1995.

[4] D. J. Bernstein, "Batch binary Edwards," in *Advances in Cryptology - CRYPTO, 29th Annual International Cryptology Conference*, 2009, pp. 317–336.

[5] G. Zhou and H. Michalik, "Comments on" a new architecture for a parallel finite field multiplier with low complexity based on composite field"," *IEEE Transactions on Computers*, vol. 59, no. 7, pp. 1007–1008, 2010.

[6] H. Fan, J. Sun, M. Gu, and K.-Y. Lam, "Overlap-free Karatsuba–Ofman polynomial multiplication algorithms," *IET Information security*, vol. 4, no. 1, pp. 8–14, 2010.

[7] M. Cenk, M. A. Hasan, and C. Negre, "Efficient subquadratic space complexity binary polynomial multipliers based on block recombination," *IEEE Transactions on Computers*, vol. 63, no. 9, pp. 2273–2287, 2014.

[8] C. Negre, "Efficient binary polynomial multiplication based on optimized Karatsuba reconstruction," *Journal of Cryptographic Engineering*, vol. 4, no. 2, pp. 91–106, 2014.

[9] S. Winograd, *Arithmetic complexity of computations*. Siam, 1980, vol. 33.

[10] P. L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362–369, 2005.

[11] M. Cenk, C. K. Koç, and F. Ozbudak, "Polynomial multiplication over finite fields using field extensions and interpolation," in *Computer Arithmetic, 2009. ARITH 2009. 19th IEEE Symposium on.* IEEE, 2009, pp. 84–91.

[12] M. Cenk and F. Ozbudak, "Improved polynomial multiplication formulas over IF_2 using Chinese remainder theorem," *IEEE Transactions on computers*, vol. 58, no. 4, pp. 572–576, 2009.

[13] H. Fan, M. Gu, J. Sun, and K.-Y. Lam, "Obtaining more Karatsuba-like formulae over the binary field," *IET Information Security*, vol. 6, no. 1, pp. 14–19, 2012.

[14] I. Oseledets, "Improved n-term Karatsuba-like formulas in $GF(2)$," *IEEE Transactions on Computers*, vol. 60, no. 8, pp. 1212–1216, 2011.

[15] Z. Dyka, P. Langendoerfer, and F. Vater, "Combining multiplication methods with optimized processing sequence for polynomial multiplier in $GF(2^k)$," in *Western European Workshop on Research in Cryptology.* Springer, 2011, pp. 137–150.

[16] P. Gallagher, "Digital signature standard (DSS)," *Federal Information Processing Standards Publications, volume FIPS*, pp. 186–3, 2013.

[17] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba algorithm for efficient implementations." *IACR Cryptology ePrint Archive*, vol. 2006, p. 224, 2006.

[18] C. Grabbe, M. Bednara, J. Teich, J. V. Z. Gathen, and J. Shokrollahi, "FPGA designs of parallel high performance $GF(2^{233})$ multipliers [cryptographic applications," in *In Proc. International Symposium on Circuits and Systems (ISCAS*, 2003, pp. 268–271.

[19] J. von zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba multipliers for polynomials over $\mathbb{F}_2$," in *International Workshop on Selected Areas in Cryptography*. Springer, 2005, pp. 359–369.

[20] F. Rodríguez-Henríquez, "On fully parallel Karatsuba multipliers for $GF(2^m)$," in *Proc. International Conference on Computer Science and Technology-CST 2003, May*. Acta Press, 2003.

[21] Z. Ge, G. Shou, Y. Hu, and Z. Guo, "Design of low complexity $GF(2^m)$ multiplier based on Karatsuba algorithm," in *Communication Technology (ICCT), 2011 IEEE 13th International Conference on*. IEEE, 2011, pp. 1018–1022.

[22] M. Cenk and M. A. Hasan, "Some new results on binary polynomial multiplication," *Journal of Cryptographic Engineering*, vol. 5, no. 4, pp. 289–303, 2015.

[23] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications*. Cambridge university press, 1994.

[24] G. Seroussi, *Table of low-weight binary irreducible polynomials*. Hewlett-Packard Laboratories, 1998.

[25] D. M. Burton and D. M. Burton, "The history of mathematics: An introduction," 2007.

[26] H. Wu, "Bit-parallel finite field multiplier and squarer using polynomial basis," *IEEE Transactions on Computers*, vol. 51, no. 7, pp. 750–758, 2002.

[27] Y. Li, Y. Zhang, and X. Guo, "Efficient non-recursive bit-parallel Karatsuba multiplier for a special class of trinomials," *VLSI Design*, vol. 2018, 2018.

[28] H. Fan, "A Chinese remainder theorem approach to bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials," *IEEE Transactions on Computers*, no. 1, pp. 1–1, 2016.

[29] Y. Li, X. Ma, Y. Zhang, and C. Qi, "Mastrovito form of non-recursive Karatsuba multiplier for all trinomials," *IEEE Transactions on Computers*, vol. 66, no. 9, pp. 1573–1584, 2017.

[30] M. Imran and M. Rashid, "Architectural review of polynomial bases finite field multipliers over $GF(2^m)$," in *Communication, Computing and Digital Systems (C-CODE), International Conference on.* IEEE, 2017, pp. 331–336.

[31] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography.* Springer Science & Business Media, 2006.

[32] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," in *Doklady Akad. Nauk SSSR*, vol. 145, no. 293-294, 1962, p. 85.

[33] V. Afanasyev, "Complexity of VLSI implementation of finite field arithmetic," in *Proc.II.Intern.Workshop on Algebraic and Combinational Coding Theory*, USSR, 1990, pp. 6–7.

[34] M. Cenk, C. Negre, and M. A. Hasan, "Improved three-way split formulas for binary polynomial and Toeplitz matrix vector products," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1345–1361, 2013.

[35] H. Fan and M. A. Hasan, "A new approach to subquadratic space complexity parallel multipliers for extended binary fields," *IEEE Transactions on Computers*, vol. 56, no. 2, pp. 224–233, 2007.

[36] M. A. Hasan, N. Meloni, A. H. Namin, and C. Negre, "Block recombination approach for subquadratic space complexity binary field multiplication based on Toeplitz matrix-vector product," *IEEE Transactions on Computers*, vol. 61, no. 2, pp. 151–163, 2012.

[37] S.-M. Park, K.-Y. Chang, D. Hong, and C. Seo, "New block recombination for subquadratic space complexity polynomial multiplication based on overlap-free approach," *IEEE Transactions on Computers*, 2017.

[38] B. Sunar, "A generalized method for constructing subquadratic complexity gf $GF(2^k)$ multipliers," *IEEE Transactions on Computers*, vol. 53, no. 9, pp. 1097–1105, 2004.

[39] H. Fan and M. A. Hasan, "Comments on" five, six, and seven-term Karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 56, no. 5, pp. 716–717, 2007.

[40] H. Fan and Y. Dai, "Fast bit-parallel $GF(2^n)$ multiplier for all trinomials," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 485–490, 2005.

[41] J. Han and H. Fan, "$GF(2^n)$ shifted polynomial basis multipliers based on subquadratic Toeplitz matrix-vector product approach for all irreducible pentanomials," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 862–867, 2015.

[42] S.-M. Park, K.-Y. Chang, D. Hong, and C. Seo, "Explicit formulae for Mastrovito matrix and its corresponding Toeplitz matrix for all irreducible pentanomials using shifted polynomial basis," *Integration, the VLSI Journal*, vol. 53, pp. 27–38, 2016.

[43] M. A. Hasan and C. Negre, "Multiway splitting method for Toeplitz matrix vector product," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1467–1471, 2013.

[44] S.-M. Park, K.-Y. Chang, D. Hong, and C. Seo, "Comments on multiway splitting method for Toeplitz matrix vector product," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 332–333, 2016.

[45] H. Fan and M. A. Hasan, "Subquadratic computational complexity schemes for extended binary field multiplication using optimal normal bases," *IEEE Transactions on Computers*, vol. 56, no. 10, 2007.

[46] J. Adikari, A. Barsoum, M. A. Hasan, A. H. Namin, and C. Negre, "Improved area-time tradeoffs for field multiplication using optimal normal bases," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 193–199, 2013.

[47] C.-Y. Lee and P. K. Meher, "Area-efficient subquadratic space-complexity digit-serial multiplier for type-II optimal normal basis of $GF(2^n)$ using symmetric TMVP and block recombination techniques," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 12, pp. 2846–2855, 2015.

[48] C. W. Chiou, Y.-S. Sun, C.-M. Lee, J.-M. Lin, T.-P. Chuang, and C.-Y. Lee, "Gaussian normal basis multiplier over $GF(2^m)$ using hybrid subquadratic-and-quadratic TMVP approach for elliptic curve cryptography," *IET Circuits, Devices & Systems*, vol. 11, no. 6, pp. 579–588, 2017.

[49] C. Yang, J.-S. Pan, C.-Y. Lee, and L. Yan, "Reduction of space complexity based on symmetric TMVP," *Electronics Letters*, vol. 51, no. 9, pp. 697–699, 2015.

[50] M. A. Hasan and C. Negre, "Subquadratic space complexity multiplier for a class of binary fields using Toeplitz matrix approach," in *Computer Arithmetic, 2009. ARITH 2009. 19th IEEE Symposium on.* IEEE, 2009, pp. 67–75.

[51] H. Fan and M. A. Hasan, "A survey of some recent bit-parallel $GF(2^n)$ multipliers," *Finite Fields and Their Applications*, vol. 32, pp. 5–43, 2015.

[52] A. Hasan and C. Negre, "Low space complexity multiplication over binary fields with Dickson polynomial representation," *IEEE Transactions on Computers*, vol. 60, no. 4, pp. 602–607, 2011.

[53] J.-S. Pan, C.-Y. Lee, and Y. Li, "Subquadratic space complexity Gaussian normal basis multipliers over $GF(2^m)$ based on Dickson–Karatsuba decomposition," *IET Circuits, Devices & Systems*, vol. 9, no. 5, pp. 336–342, 2015.

[54] S. Akleylek, M. Cenk, and F. Özbudak, "Polynomial multiplication over binary fields using Charlier polynomial representation with low space complexity," in *International Conference on Cryptology in India*. Springer, 2010, pp. 227–237.

[55] F. Özbudak, S. Akleylek, and M. Cenk, "A new representation of elements of binary fields with subquadratic space complexity multiplication of polynomials," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 96, no. 10, pp. 2016–2024, 2013.

[56] Circuit minimization work, "A web page including explicit formulas for multiplication over the binary field by the circuit minimization team at the Yale university." [Online]. Available: http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html, 2013

[57] Nangate Standard Cell Library, "Nangate freepdk45 generic open cell library." [Online]. Available: http://www.si2.org/openeda.si2.org/projects/nangatelib, 2011.

APPENDIX A

Lemma 2. *The i-th term of a geometric sequence with initial value a and common ratio r is given by*

$$a_i = ar^{i-1}$$

Then a geometrics series, defined as the sum of numbers in a geometric progression, can be represented as

$$\begin{aligned} S &= a + ar + ar^2 + \cdots\cdots + ar^{i-1} \\ &= \frac{a(r^i - 1)}{r - 1} \end{aligned} \tag{17}$$

where $r \neq 1$.

The derivation of equation (17) *is very simple. Let* S *multiply the common ratio* r *and then minus* S. (17) *can be obtained by deforming the result of* $(r-1)S$.

Lemma 3. *Let* a, b, c, d, e *and* n *be positive integers, where* $a \neq b$, $a \neq 1$ *and* $n = b^m$. *There is a recurrence relations*

$$\begin{cases} R_1 = R(b) = e; \\ R_m = R(n) = aR(\frac{n}{b}) + cn + d. \end{cases} \tag{18}$$

The non-recursive form of R_m *is*

$$R_m = (\frac{e(a-1)+d}{a(a-1)} + \frac{cb^2}{a(a-b)})n^{\log_b a} - \frac{bc}{a-b}n - \frac{d}{a-1}. \tag{19}$$

Proof. The following part shows the derivation of lemma 3.

(18) is first-order non-homogeneous recurrence formulas with variable coefficients. The expansion of the recurrence relations shown in Equation (18) can be represented

as

$$\begin{cases} R_1 = e; \\ R_2 = aR_1 + cb^2 + d; \\ \quad \vdots \\ R_{m-1} = aR_{m-2} + cb^{m-1} + d; \\ R_m = aR_{m-1} + cb^m + d. \end{cases} \tag{20}$$

Let both sides of the equation (20) multiply a^{m-i} (i denotes the number of recursions.).

$$\begin{cases} a^{m-1}R_1 = a^{m-1}e; \\ a^{m-2}R_2 = a^{m-2}(aR_1 + cb^2 + d); \\ \quad \vdots \\ aR_{m-1} = a(aR_{m-2} + cb^{m-1} + d); \\ R_m = aR_{m-1} + cb^m + d. \end{cases} \tag{21}$$

Then the left and right sides of the formula in Equation (21) are added respectively.

$$R_m + \sum_{i=1}^{m-1} a^i R_{m-i} = \sum_{i=1}^{m-1} a^i R_{m-i} + c\sum_{i=0}^{m-2} a^i b_{m-i} + d\sum_{i=0}^{m-2} a^i + a^{m-1}e. \tag{22}$$

After both sides eliminate the term of $\sum_{i=1}^{m-1} a^i R_{m-i}$, R_m can be expressed as the summation of two geometric series and one exponential term.

Firstly, let

$$S_1 = c\sum_{i=0}^{m-2} a^i b_{m-i};$$
$$S_2 = d\sum_{i=0}^{m-2} a^i.$$

According to lemma 2,

$$S_1 = \frac{c(a^{m-1}b^2 - b^{m+1})}{a-b};$$
$$S_2 = \frac{d(a^{m-1}-1)}{a-1},$$

where $a \neq 1$ and $a \neq b$.

Replacing m with $\log_b n$, R_m can be given by

$$R_m = (\frac{e(a-1)+d}{a(a-1)} + \frac{cb^2}{a(a-b)})n^{\log_b a} - \frac{bc}{a-b}n - \frac{d}{a-1}.$$

■

APPENDIX B

Proof. The induction method is applied to prove lemma 1.

When $n = 1$, the proof of lemma 1 is simple. The polynomial multiplication $C = AB = R(\hat{C}) = \hat{C}$ for all C, which implies that $R(\hat{C}) + R(\hat{C}') = R(\hat{C} + \hat{C}')$.

When $n = 2^m$, it is assumed that the lemma is true for $n = 2^{m-1}$. It is noted from the graph 3.5 that $\hat{C}$ and $\hat{C}'$ can be decomposed into three parts, where

$$\hat{C} = \{\hat{C}_0;\ \hat{C}_1;\ \hat{C}_2\};$$
$$\hat{C}' = \{\hat{C}'_0;\ \hat{C}'_1;\ \hat{C}'_2\}.$$

According to the definition of R,

$$R(\hat{C}) = [R(\hat{C}_0),\ R(\hat{C}_1),\ R(\hat{C}_0) + R(\hat{C}_1) + R(\hat{C}_2)];$$
$$R(\hat{C}') = [R(\hat{C}'_0),\ R(\hat{C}'_1),\ R(\hat{C}'_0) + R(\hat{C}'_1) + R(\hat{C}'_2)].$$

Then based on the induction hypothesis and previous property, following derivation process is expressed.

$$\begin{aligned} R(\hat{C}) + R(\hat{C}') =& [R(\hat{C}_0) + R(\hat{C}'_0),\ R(\hat{C}_1) + R(\hat{C}'_1), \\ & R(\hat{C}_0) + R(\hat{C}'_0) + R(\hat{C}_1) + R(\hat{C}'_1) + R(\hat{C}_2) + R(\hat{C}'_2)]; \\ =& [R(\hat{C}_0 + \hat{C}'_0),\ R(\hat{C}_1 + \hat{C}'_1),\ R(\hat{C}_0 + \hat{C}'_0) + R(\hat{C}_1 + \hat{C}'_1) + R(\hat{C}_2 + \hat{C}'_2)]; \\ =& R(\hat{C} + \hat{C}'). \end{aligned}$$

■

VITA AUCTORIS

NAME:	Xiaolin Duan
PLACE OF BIRTH:	Anhui, China
YEAR OF BIRTH:	1991
EDUCATION:	Anhui University, Hefei, China Bachelor of Computer Science, 2009-2013
	University of Windsor, Windsor, ON, Canada Master of Applied Science, Electrical and Computer Engineering, 2015-2018