9-27-2023

Area-Efficient Finite Field Multiplication Using Hybrid SET-MOS Technology

Chen Zhang
University of Windsor

Area-Efficient Finite Field Multiplication Using Hybrid SET-MOS Technology

by

Chen Zhang

A Dissertation

Submitted to the Faculty of Graduate Studies

through the Department of Electrical and Computer Engineering

in Partial Fulfilment of the Requirements for

the Degree of Doctor of Philosophy at the

University of Windsor

Windsor, Ontario, Canada

Area-Efficient Finite Field Multiplication Using Hybrid SET-MOS Technology

by

Chen Zhang

APPROVED BY:

E. Atoofian, External Examiner

Lakehead University

X. Yuan

School of Computer Science

J. Wu

Department of Electrical and Computer Engineering

S. Chowdhury

Department of Electrical and Computer Engineering

C. Chen, Co-Advisor

Department of Electrical and Computer Engineering

H. Wu, Co-Advisor

Department of Electrical and Computer Engineering

September 5, 2023

DECLARATION OF

CO-AUTHORSHIP/PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows: Chapters 2-6 of the thesis were developed under the supervision of my co-advisors, Dr. Huapeng Wu and Dr. Chunhong Chen. My advisors provided the initial ideas for Chapters 4, 5. The logic gate optimization work for Chapter 5 was joint research in collaboration with my advisors. In all cases, the author performed the key ideas, primary contributions, logic gate design and simulation, multiplication architecture modifications, interpretation, result comparisons, and writing. The contribution of my advisors was primarily through content checking and comments on the thesis organization, providing feedback on logic gate simulations and performance evaluations, and editing the manuscript.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes parts from three original papers that have been published and submitted for publication in conferences and journals, as follows:

Thesis Chapter	Publication title/full citation	Publication status
Parts of Chapter 2, 3, 4	Chen Zhang, Chunhong Chen, and Huapeng Wu. "Area-Efficient Finite Field Multiplication in GF(2^n) Using Single-Electron Transistors." 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS), pp. 25-28, IEEE, 2021.	Published
Parts of Chapter 2, 3, 4	Chen Zhang, Huapeng Wu, and Chunhong Chen. "Area-Efficient Finite Field Multiplication Using Hybrid SET-MOS Technology." IEEE Transactions on Circuits and Systems I:Regular Papers 69.11 (2022): 4358-4366.	Published
Parts of Chapter 2, 3, 5	Chen Zhang, Huapeng Wu, and Chunhong Chen. "Area-Efficient Finite Field Multiplication Using Combinational SET-MOS Logic gates" To be submitted to IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2023	Under preparation

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's 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.

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

This work studies area-efficient implementation of bit-parallel binary polynomial multiplication and Karatsuba algorithm (KA), which have important applications in error control coding and network security, especially elliptic curve cryptography, among many other areas. All existing works in this respect are based on traditional CMOS technology. Research shows that new nanodevices such as single-electron transistors (SETs) can offer the great potential at device- and technology-level to further improve area cost in many digital applications (especially multiplication), thanks to their unique Coulomb oscillation characteristics.

This thesis first proposes hybrid SET-MOS technology to design the XOR networks that are extensively used in polynomial multiplication architectures, with the goal of reducing the total area cost for their implementation. This is done by using a single SET-MOS gate to realize a multi-input XOR operation which would traditionally require multiple CMOS gates. The existing KA architectures are modified and implemented accordingly using SET-MOS XOR gates. Results show that the proposed multiplier can provide around 37% savings in gate count (with a moderate increase in latency) for the multiplier size ranging from 256 to 2048, compared to the best of CMOS-based multipliers.

This thesis also presents novel combinational gates using SET-MOS technology in order to obtain combined AND/XOR operations within a single logic gate. This leads to area-efficient implementation for both AND and XOR operations in multiplication architectures, and produces further reduction in gate count. In comparison with the best existing CMOS work, the proposed 2-way and 4-way KA multipliers save around 46% in gate count. While the latency of proposed SET-MOS work is longer than the existing CMOS counterpart, the area-latency product of the former still shows 4% less than that for the latter. The proposed approaches in this thesis not only promise high area efficiency, but also open up new opportunities for general digital

applications involving extensive multiplications. This could potentially and significantly improve the way current multiplier architectures and algorithms are designed and implemented.

DEDICATION

To my loving Family:

Father and Mother

For their selfless contribution to my life

ACKNOWLEDGEMENTS

I want to express my profound appreciation and gratitude to all those who have contributed to the completion of this thesis and the journey of my studies. Without their unwavering support and invaluable input, this achievement would not have been possible.

First and foremost, I extend my heartfelt thanks to Dr. Huapeng Wu and Dr. Chunhong Chen, my supervisors, for their guidance, encouragement, and expertise throughout this endeavour. Their wisdom and mentorship have been instrumental in shaping the direction and quality of this work. Thanks to their patience and meticulous instruction, and diligent guidance, I have become a mature person ready to take on more challenging tasks in my future life.

I want to thank Ms. Andria Ballo, the graduate secretary at the Electrical and Computer Engineering Department, for her valuable time and unwavering support.

Additionally, I want to thank my parents and friends for their support, understanding, and encouragement during this demanding journey. Their belief in me has been a constant source of motivation and inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ASIC Application Specific Integrated Circuit

CM Component Multiplication in block recombination

CMOS Complementary Metal-Oxide-Semiconductor

CPF Component Polynomial Formation in block recombination

ECC Elliptic Curve Cryptography

ECDLP Elliptic Curve Discrete Logarithm Problem

FPGA Field Programmable Gate Array

GPDK45 $45nm$ generic process design kits

IoT Internet of Things

KA Karatsuba algorithm

PD Propagation delay

PTL Pass-Transistor Logic

RC Reconstruction in block recombination

RFID Radio Frequency Identification

RSA Rivest-Shamir-Adleman

SBM Schoolbook Method

SBM-KA Hybrid multiplication architecture of SBM and KA

SET Single Electron Transistor

SET-MOS Hybrid SET and CMOS

XOR Exclusive-OR

1 Introduction

1.1 Research Motivation

Network security refers to the practices and technologies to secure computer networks from unauthorized access. In recent years, threat situations have significantly evolved, with cyber-attacks becoming more frequent and sophisticated. These attacks can result in data breaches, financial losses, property damage, and even injuries to life in some cases. Therefore, deploying resilient network security and cryptographic measures to safeguard against these potential risks is crucial. Cryptography is a fundamental technology used to ensure the security and privacy of digital communication. It can provide confidentiality, integrity, and authenticity by ensuring that data transmission over networks is secure and tamper-proof [1, 2].

Public key cryptography provides more security services than symmetric key cryptography, transforming how we secure digital communication. Unlike symmetric key cryptography, which relies on a single shared secret key to encrypt data, public key cryptography uses two mathematically related keys, a public key and a private key, to encrypt and decrypt information [3]. Public key cryptography has many applications, including secure email communication, secure online transactions, and digital signature verification. It is also a critical technology used to implement secure network protocols, such as SSL/TLS, SSH, and IPsec. Additionally, public key cryptography establishes trust between two parties who have never met before, enabling secure communication and collaboration [4].

One of the most significant challenges of public key cryptography is that it is computationally intensive and time-consuming [5]. This is because public key cryptography involves complex mathematical computations that require considerable arithmetic power and resources, making them slow and difficult to establish [6]. A dedicated hardware implementation, such as an accelerator or co-processor, can provide

enough computing power and bandwidth to support public key cryptography computations, allowing for more security strength or time savings [7]. As a result, efficient hardware implementation on public key cryptography with both low latency and small area has become a popular research topic in past decades.

The elliptic curve cryptosystem, independently proposed by N. Koblitz [8] and V. Miller [9] in the mid-1980s, is a widely used public key cryptosystem known for its security and efficiency. The security of ECC relies on the mathematical difficulty of solving the elliptic curve discrete logarithm problem (ECDLP) over a finite field [2], while RSA [10], which is the other commonly used public key cryptography, establishes its security in factoring large numbers. In recent years, elliptic curve cryptosystem has become preferable to RSA, especially in applications requiring smaller key sizes and better performance with equivalent security levels [11].

In elliptic curve cryptosystems, point multiplication (scalar multiplication) is the primary operation based on finite field operations. The efficiency of ECC implementation largely depends on the speed and performance of computing the point multiplications [12]. The primary operations in point multiplication are finite field multiplications and finite field inversions (multiplicative inverse). These two finite field operations have a higher degree of complexity than the other two operations (additions and subtractions), which can affect the overall efficiency of ECC applications. Therefore, an efficient implementation of the finite field multiplier is crucial, especially for applications where resources are constrained and fast execution time is also required. *i.e.*, smart cards, Radio Frequency Identification (RFID) tags, IoT devices, and cell phones [13]. Besides cryptography, finite field multipliers also have applications in coding theory, and other mathematics and computer science areas where finite fields are used [14].

There are several main types of binary field polynomial multiplication architectures. A bit-parallel polynomial multiplier computes all the bits of the polynomial

product in parallel, which are fast but require a large area [15]. A bit-serial polynomial multiplier computes the polynomial product one bit at a time, which is slow but requires a small area [16]. A digit-serial polynomial multiplier is a compromise between bit-parallel and bit-serial multiplication [17].

Bit-parallel multiplication is often preferred in real-time systems because it offers substantially higher processing speed than the other two. However, many applications have strict requirements for speed meanwhile also requiring a small area. The schoolbook implementation of bit-parallel multiplier of size n requires $\mathrm{O}(n^2)$ area complexity, which can be inefficient regarding area for large polynomial size n [18].

Karatsuba algorithm (KA) is a divide-and-conquer algorithm that reduces the number of operations required by recursively breaking down the multiplications into smaller sub-multiplications [19]. KA multiplication can compute the product of two n-bit operands by three sub-multiplications of $\frac{n}{2}$-bit operands along with a few additions, resulting in an overall area complexity of $O(n^{\log_2 3})$, which is more area-efficient than schoolbook multiplication, especially for large size multiplier.

Much work has been proposed to improve this algorithm since the KA has been extended from integer multiplication to finite field multiplications [20]. The reconstructed KA [21, 22] improved the area complexity of the KA with similar latency complexity (critical path delay). Overlap-free KA [23] improved the latency complexity and kept the same area complexity compared to the original KA. The block recombination method [24] further improved the area complexity and kept the same latency compared to reconstructed KA under the 2-way splits. The optimization techniques discussed are primarily focused on improving a specific process of the Karatsuba algorithm, namely the reconstruction process. In addition, other methods (such as those described in references [25, 26, 27, 28, 29, 30]) aim to further improve the area or time complexity of the algorithm by using more splits to generate fewer sub-multiplications.

The above improvements of the multiplier are from the algorithm-to-architecture perspective. When we move to the gate-to-device perspective, bit-parallel multipliers are implemented using Application Specific Integrated Circuits (ASIC) and Field Programmable Gate Arrays (FPGA). Both approaches employ Complementary Metal-Oxide-Semiconductor (CMOS) technology. The Single Electron Transistor (SET) is a fresh nanoelectronics device that operates on the principle of controlling the flow of individual electrons through a small circuit [31]. It consists of two tunnel junctions constructed by ultrathin dielectrics and a conductor between the tunnel junctions as an island. A thick dielectric coupled to the island serves as gates, each of which controls the flow of electrons through the SET. Applying voltages to the gates(inputs) of SET makes it possible to occur a unique characteristic, the Coulomb-blockade oscillation [32]. The oscillation characteristic makes SET appealing for various applications, particularly those that involve operations with multiple inputs. The finite field multiplier necessitates a plentiful presence of this type of logic gate that accepts multiple inputs.

1.2 Problem Statement

Bit-parallel multipliers are indispensable in elliptic curve cryptosystems to provide high-speed computation. It needs to address the problem of enormous area requirements. The KA has significantly reduced the area requirement of bit-parallel polynomial multiplications at the cost of a slight latency increase. However, existing work focusing on improvements to KA has reached a bottleneck, where the area reduction is getting less and less. More research efforts, probable gate- or device-level investigations, are needed to break the research bottleneck and further enhance the area performance of bit-parallel multipliers.

1.3 Research Contributions

This thesis covers a comprehensive research scope on finite field polynomial multiplications over binary extension field, including architecture- and algorithmic-level designs, as well as gate- and device-level implementations. We used Cadence software to design and simulate logic gates, and integrated knowledge from both algorithmic and digital circuit designs to perform a reasonable scheme for evaluating the multiplier's area and latency performance within our research area.

1. Our research involves the design of 4-input XOR gates using SET technology. We conducted a comprehensive study on existing parallel polynomial multiplication architectures, with a focus on identifying structures and building blocks where the 4-input XOR gates can be effectively applied. The utilization of SET technology in implementing KA multiplication has resulted in a notable improvement of approximately 30% in terms of area performance when compared to the existing best result achieved with CMOS technology. This aspect of the work was published in the 2021 IEEE Asia Pacific Conference on Circuits and Systems.

2. Our research presents a hybrid multiplication architecture that is specifically designed to well-suit SET-based XOR gates for further area savings. The hybrid multiplication architecture has shown a significant improvement of around 9.8% in terms of area performance compared to our previous work, and a remarkable 37% improvement when compared to the best results obtained using CMOS technology. We also enhance the latency of hybrid multiplication architecture. This aspect of the work was published in IEEE Transactions on Circuits and Systems I: Regular Papers in November 2022.

3. Our research proposes novel logic gates with SET technology that can achieve $(a \wedge b) \oplus c$ and $(a \wedge b) \oplus c \oplus d$ logic operations. We implement these operations

directly by observing the truth tables and designing the mapping using the oscillation characteristic of SET-MOS. With the proposed combinational gates, the proposed 2-way hybrid multiplier outperforms the best of CMOS-based existing work by saving 46% in gate count and 4% in the area-and-latency-product. The proposed modified 4-way hybrid multiplier outperforms its CMOS counterpart by saving 46% in gate count and 6% in the area-and-latency-product. If compared to our other work, the proposed multiplier with combinational gates achieves a further 14% improvement in gate count and 2.7%–5.7% improvement in area-and-latency product. This aspect of the work is going to be submitted to IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

1.4 Organization of Thesis

The rest of the thesis is organized as follows. Chapter 2 introduces the math background of finite fields and the architectures of polynomial multiplication. Chapter 3 introduces the implementation of bit-parallel multiplication, followed by an introduction to SET technology. Chapter 4 designs the multiple-input XOR gate using SET technology and applies these XOR gates to KA multiplication and its extensions. It also presents a hybrid KA multiplication architecture with better area performance. Chapter 5 develops the combinational gates using SET technology and applies the combinational gates in multiplications. The most area-efficient bit-parallel multipliers are proposed in this chapter. Chapter 6 concludes the thesis and suggests future research opportunities.

2 Math Background

This chapter covers the basics of the finite field and its arithmetic, particularly the multiplication operation. The latter part of the chapter delves into a discussion on architectures of finite field multiplication.

2.1 Finite Fields

Definition 2.1. A field, denoted as F, is a set of elements together with two binary operations, $(+)$ and $(*)$, on F, which satisfies the following properties [33]:

- $(+)$ operation in F is commutative, which means $a + b = b + a$; $(*)$ operation in F is commutative, which means $a * b = b * a$.
- $(+)$ operation is associative. For all $a, b, c \in F$, the expression $(a + b) + c = a + (b + c)$ holds; $(*)$ operation is associative. For all $a, b, c \in F$, the expression $(a * b) * c = a * (b * c)$ holds.
- There exists an element $0 \in F$ such that $0 + a = a + 0 = a$ for all $a \in F$; There exists an element $1 \in F$ such that for all $a \in F$, $1 * a = a * 1 = a$.
- For every $a \neq 0$, $a \in F$, there exists an element $a^{-1} \in F$ such that $a^{-1} + a = a + a^{-1} = 0$; for every $a \neq 0$, $a \in F$, there exists an element $a^{-1} \in F$ such that $a^{-1} * a = a * a^{-1} = 1$.
- $(*)$ operation is distributive over $(+)$ operation: For any a, b, c in F, $a*(b+c) = a * b + a * c$, and $(b + c) * a = b * a + c * a$.

There are two types of fields, namely, infinite fields and finite fields. An infinite field is a field that has infinite number of element. The most well-known examples of infinite fields are the field of real numbers and the field of complex numbers. A finite field or Galois field is a field that contains a finite number of elements, denoted as $GF(q)$ or $\mathbb{F}_q$. The general form for finite fields is $GF(q) = \{0, 1, 2, ..., q - 1\}$.

A finite field can only exist when q equals p^n, where p is a prime number (known as the characteristic of the finite field) and n is a positive integer. If $n = 1$, the finite field is known as the prime field $GF(p)$. If $n > 1$, then it is known as an extension finite field. When $p = 2$, the finite field is called the binary extension finite field, denoted as $GF(2^n)$. The arithmetic operations in $GF(2^n)$ binary extension fields are considerably simpler than those in $GF(p)$ prime fields [34]. This thesis concentrates on the binary extension field $GF(2^n)$.

In binary fields, an irreducible polynomial is a polynomial that cannot be factored into two non-constant polynomials of lower degrees over the field (detailed refer to [35]). Let $f(x)$ be an irreducible polynomial of degree n over $GF(2^n)$. Then a polynomial basis for $GF(2^n)$ over $GF(2)$ can be formed as,

$$\{1,\ x,\ x^2,\ x^3,\ ...,\ x^{n-1}\},$$

where x is a root of $f(x)$. Any elements, such as A, in $GF(2^n)$ with respect to the polynomial basis can be represented as[1],

$$A(x) = \sum_{i=0}^{n-1} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1},$$

where the coefficients $a_i \in \{0, 1\}$. It can also be denoted as a binary vector like

$$A = (a_{n-1}, a_{n-2}, \cdots, a_2, a_1, a_0).$$

When representing $GF(2^n)$ field elements, there are commonly used bases: polynomial basis, normal basis, dual basis, and redundant basis [36]. The most popular bases in modern applications are polynomial basis and a specific type of normal ba-

[1]In this thesis, + denotes modular two addition when operands are signals in a finite field; and + denotes the addition of real numbers when operands are real numbers, such as those calculations of area and latency complexities. It depends on the properties of the operands.

sis known as the optimal normal basis [34]. Different representation basis has pros and cons. Like normal basis can handle squaring operations over at no cost but just shifting, but it lacks efficiency in hardware implementation and is hard to use in handling multiplication and inversion [37]. The polynomial basis is more efficient for multiplication, and it is ideal for hardware implementation [38]. This thesis will only focus on finite field arithmetic with a polynomial basis. In addition, trinomials, pentanomials, equally spaced polynomials, and all one polynomials are the major types of irreducible polynomials that are commonly used in literature for implementing finite field arithmetic. The selection of irreducible polynomial is flexible, and the complexity of implementation is related to which kind of polynomial is selected.

2.2 Arithmetic in Binary Extension Field

The major arithmetic in $GF(2^n)$ are addition, subtraction, multiplication, inversion, and squaring. With polynomial basis, these operations are described as follows.

- **Addition** and **subtraction** are the same. Let $A(x)$ and $B(x)$ be two elements in $GF(2^n)$, we have:

$$A(x) + B(x) = \sum_{i=0}^{n-1}(a_i + b_i)x^i \ \ mod\, 2. \tag{2.1}$$

The addition of A and B under modular 2 is equivalent to the bit-wise XOR operation of A and B.

- **Multiplication** is the most frequent and fundamental. Let $A(x)$ and $B(x)$ be two elements in $GF(2^n)$, the straightforward multiplication (also called schoolbook method or classical method) can be shown as follow,

$$A(x)B(x) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j x^{i+j} \ \ mod\, f(x) \ \ mod\, 2. \tag{2.2}$$

Noting that the multiplication involves a polynomial multiplication followed by a modular reduction.

- **Squaring** can be seen as a special case of multiplication for two identical operands. For example, $A(x)$ in $GF(2^n)$, the squaring of A(x) can be represented as,

$$A^2(x) = \sum_{i=0}^{n-1} a_i x^{2i} \ mod\ f(x). \tag{2.3}$$

- **Inversion** can be achieved by serial multiplications. Let $A(x)$ be an element in $GF(2^n)$. There always exists an element $A^{-1}(x)$, the inverse of $A(x)$, because the irreducible polynomial $f(x)$ cannot be factorized and the great common divisor of $A(x)$ and $f(x)$ is 1.

It is seen that in $GF(2^n)$, addition, subtraction, and squaring are easy to implement, while inversion can be computed by multiplications. Thus, the importance of efficient multiplication is evident. Next, three major architectures of finite field multipliers will be presented.

2.3 Architectures of Finite Field Polynomial Multiplication

Hardware implementation of multiplications in $GF(2^n)$ fields can be roughly categorized into these major types: bit-serial, digit-serial, and bit-parallel.

- **Bit-serial** multiplication performs one bit at one clock cycle, with each bit of the product generated sequentially. The bit-serial multiplication design can be relatively easy to implement and efficient in terms of hardware resources (*i.e.*, area requirement) [16, 39]. However, its sequential structure can lead to longer latency and circuit depth in the computation process.

- In **bit-parallel** multiplications, all the bits of the operands are processed simultaneously, typically using parallel logic gates and networks consisting of logic

gates [15, 18]. This method can be the fastest than the other two, but it can also have the greatest demand on area, as it requires a large number of logic gates to achieve its parallel computation.

- **Digit-serial** multiplication is a trade-off between the other two types of multiplication. This type of multiplier defines a digit size, then divides the two inputs (any two elements in $GF(2)$) in terms of digits according to the digit size. Each clock cycle computes one digit, and the resulting bits are accumulated to form the final sequence [17, 40]. The complexity of this type of multiplication depends on the size of the digit.

The asymptotic space and time complexity to evaluate the performance of different types of multiplication is summarized in Table 2.1, where the area complexity is of operand count and clock cycle.

Table 2.1: Complexities of different types of multiplication.

Type	Area complexity	Clock cycle
Bit-serial	$O(n)$	$O(n)$
Digit-serial	$O(dn)^{\dagger}$	$O(d)$
Bit-parallel	$O(n^{1+\epsilon})$, $0 < \epsilon \leqslant 1$	$O(1)$

† d denotes the digit size.

It is worth noting that the complexity of digit-serial multiplication is influenced by the size selection of digits. For example, in Table 2.1, as the digit size d increases from 1 to n, its area complexity can increases from $O(n)$ to $O(n^2)$, while its time complexity can decrease from $O(n)$ down to $O(1)$[17]. In addition, bit-parallel multiplication is the fastest of all, and its space complexity, can be affected by different multiplication architectures. The quadratic area complexity $O(n^2)$ can be reduced to sub-quadratic area complexity $O(n^{1+\epsilon})$, $0 < \epsilon < 1$, by improvement from the multiplication algorithm or architecture perspective [15]. And this is the main focus of this thesis.

3 Research Overview

This chapter is an overview of the existing work, including multiplication architectures with quadratic area complexity $O(n^2)$ and sub-quadratic area complexity $O(n^{\log_2 3})$.

3.1 Quadratic Area Complexity Methods

This section introduces the schoolbook method (SBM) represented by polynomial basis.

3.1.1 Schoolbook Method

Let binary extension field $GF(2^n)$ be generated with the irreducible polynomial $f(x)$ of degree n. Then $f(x)$ introduces a polynomial basis, $\{1, x, x^2, x^3, \ldots, x^{n-1}\}$, in $GF(2^n)$. Suppose A and B are two elements in $GF(2^n)$ represented with polynomial basis $A(x) = \sum_{i=0}^{n-1} a_i x^i$ and $B(x) = \sum_{i=0}^{n-1} b_i x^i$, where $a_i, b_i \in GF(2)$, $0 \leq i \leq n-1$. A bit-parallel polynomial multiplication of A and B can be given as,

$$A(x)B(x) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_i b_j x^{i+j} \triangleq \sum_{k=0}^{2n-2} c_k x^k = C(x),$$

where,

$$c_k = \sum_{\substack{i+j=k \\ 0\leq i,j\leq n-1}} a_i b_j, \ k = 0, 1, \ldots, 2n-2. \tag{3.1}$$

A following-up step of polynomial reduction modulo $f(x)$ would complete finite field multiplication. In this work we focus on the polynomial multiplication to obtain c_k for given A and B. In the sequel, the polynomial multiplication using (3.1) will be denoted as the SBM.

There are only two kinds of ground field operations in (3.1), and they can be conveniently implemented by 2-input XOR and 2-input AND gates. We denote the gate counts of 2-input AND and 2-input XOR as $G_\otimes$ and $G_\oplus$, respectively. Latency,

denoted as L, is usually evaluated by the number of gates on the longest path (critical path/circuit depth) from input to output of the multiplier [41, 42, 43]. The gate delay of 2-input AND and 2-input XOR are denoted as $D_{\otimes}$ and $D_{\oplus}$. Clearly from (3.1) we see the area complexity of SBM is $O(n^2)$ respecting to multiplier size n.

From (3.1), the area complexity of SBM can be summarized as,

$$\begin{cases} G_{\otimes}(n) = n^2, \\ G_{\oplus}(n) = n^2 - 2n + 1. \end{cases} \tag{3.2}$$

The critical path length occurs during the processing of $c_{n-1} = a_0 b_{n-1} + \cdots + a_{n-1} b_0$ from (3.1). Therefore, the latency of SBM is,

$$L(n) = \lceil \log_2 (n) \rceil D_{\oplus} + D_{\otimes}.$$

The SBM is the fastest among all binary field polynomial multipliers, but it lack of application scenarios due to the immense area complexity. The area issue of schoolbook becomes even more pronounced when dealing with larger multiplier sizes. Therefore, in bit-parallel multiplication, the main objective is to achieve optimal space complexity while maintaining an acceptable critical path delay.

When it comes to improving the area complexity of the bit-parallel multiplier, there are two classes that need to be talked about. One aims to linearly reduce the quadratic area complexity $O(n^2)$ with modifications on multiplication architecture while keeping the latency no larger than $\log_2 n$, such as the Chinese remainder theorem method [44]. The other class exponentially reduces the area complexity to sub-quadratic $O(n^{\log_2 3})$ at the cost of doubling to tripling the latency, which will be discussed in the following section.

3.1.2 Mastrovito Method

One quadratic method , the Mastrovito method [45], has an area complexity of $O(n^2)$. It differs from SBM, which focuses on polynomial multiplication, as it involves both modular reduction and polynomial multiplication. Extensions of this method, such as [46, 47], improve the Mastrovito multiplier, but their area complexities still remain quadratic.

3.2 Sub-Quadratic Area Complexity Methods

The following section is dedicated to discussing KA and its extensions, specifically in regard to 2-way and 4-way splits. These sub-quadratic methods can help reduce the excessive area complexity of bit-parallel multiplication, albeit at the expense of a slight increase in latency.

3.2.1 Original 2-Way KA

In order to reduce the quadratic area complexity, KA has been intensively researched and applied for polynomial multiplication in finite fields [15]. Research has shown that KA multipliers are with sub-quadratic area complexity, *i.e.*, [19]. The original KA is described in detail to facilitate an understanding of the complexity calculation.

Let $A(x)$ and $B(x)$ be two elements in the binary extension field $GF(2^n)$, where n is a power of 2. They can be divided into two parts of equal length,

$$A(x) = \sum_{i=0}^{n-1} a_i x^i = A_1(x)x^{\frac{n}{2}} + A_0(x), \quad B(x) = \sum_{i=0}^{n-1} b_i x^i = B_1(x)x^{\frac{n}{2}} + B_0(x). \quad (3.3)$$

It can be seen that all $A_j(x)$ and $B_j(x)$ are polynomials of degree $\frac{n}{2} - 1$ with x, for $j \in \{0, 1\}$. Then, the product C can be calculated as,

$$C = AB$$

$$
\begin{aligned}
&= A_0(x)B_0(x) + (A_0(x)B_1(x) + A_1(x)B_0(x))x^{\frac{n}{2}} + A_1(x)B_1(x)x^n \\
&= P_0 + (P_2 + P_1 + P_0)x^{\frac{n}{2}} + P_1x^n, \qquad (3.4)
\end{aligned}
$$

where

$$
\begin{cases}
P_0 = A_0(x)B_0(x), \;\; P_1 = A_1(x)B_1(x), \\
P_2 = (A_0(x) + A_1(x))(B_0(x) + B_1(x)).
\end{cases} \qquad (3.5)
$$

The partial product, P_i, are polynomial of degree $n-1$, for $j \in [0, 2]$, which can be seen as the products of sub-polynomial multiplication with operand size $\frac{n}{2}$. Therefore, the computation process of $C = AB$ can be divided into,

- Component process: Pre-processing the input operands $A_0(x) + A_1(x)$ and $B_0(x) + B_1(x)$ for P_2, which can be implemented by XOR gates.
- Three sub-multiplications: Calculating three multiplications of size $\frac{n}{2}$.
- Reconstruction process: Reconstructing the partial products generated by sub-multiplications. For example, a part of reconstruction is to compute $P_0+P_1+P_2$, which XOR gates or networks can implement.

Clearly, KA multiplication recursively uses the "divide and conquer" approach. Based on (3.4) and (3.5), we can calculate the recursive XOR and AND gate count complexity, denoted by $G_{\otimes}(n)$ and $G_{\oplus}(n)$, as shown in Table 3.1.

Table 3.1: Recursive gate count of 2-way KA

Process	$G_{\oplus}(n)$	$G_{\otimes}(n)$
Component	n	0
Sub-multiplication	$3G_{\oplus}(\frac{n}{2})$	$3G_{\otimes}(\frac{n}{2})$
Reconstruction	$3n-4$	0

Reconstruction for 2-way split KA can be computed in the following two steps.

- Step 1: $R = P_0 + P_1 + P_2$. Since each P_i has $n-1$ bits, R can be processed by $2(n-1) = 2n-2$ 2-input XOR gates.

- Step 2: $AB = P_0 + Rx^{\frac{n}{2}} + P_1 x^n$. The overlap can be processed by $2(\frac{n}{2}-1) = n-2$ 2-input XOR gates.

Consequently, the recursive area complexity of 2-way original KA can be summarized as,

$$\begin{cases} G_{\otimes}(n) = 3G_{\otimes}(\frac{n}{2}), \\ G_{\oplus}(n) = 3G_{\oplus}(\frac{n}{2}) + 4n - 4. \end{cases} \tag{3.6}$$

Non-recursive area complexity can be solved in many ways, here we directly use the Lemma given in [48].

Lemma 1 ([48])**.** *Let a, b, k be positive integers. Let $n = b^k$ and assume $a \neq b$, $a \neq 1$. The solution to the recurrence relations*

$$\begin{cases} R_1 = e, \\ R_n = aR_{n/b} + cn + d. \end{cases}$$

is shown as follows

$$R_n = (e + \frac{bc}{a-b} + \frac{d}{a-1})n^{\log_b a} + \frac{-bc}{a-b}n + \frac{-d}{a-1}. \tag{3.7}$$

The initial gate counts are $G_{\oplus}(1) = 0$ and $G_{\otimes}(n) = 1$. By using (3.7), non-recursive area complexity of 2-way original KA can be obtained as,

$$\begin{cases} G_{\otimes}(n) = n^{\log_2 3}, \\ G_{\oplus}(n) = 6n^{\log_2 3} - 8n + 2. \end{cases} \tag{3.8}$$

Table 3.2 shows an example to find recursive latency complexity. We calculate the latency complexity of the original KA in 5 stages. From the table we can see the total delay is one sub-multiplication's delay plus 3 XOR gates delay.

Table 3.2: Recursive latency complexity accumulation for 2-way original KA

Stage	Inputs A and B			Accumulated Latency
I	Sub-multip. for P_0	$A_0(x)+A_1(x)$; $B_0(x)+B_1(x)$	Sub-multip. for P_1	$D_\oplus$
II	Sub-multiplication for P_2		P_1+P_0	$D_\oplus + L(n/2)$
III	$R=((P_0+P_1)+P_2)$			$2D_\oplus + L(n/2)$
V	Output $C=P_0+Rx^{n/2}+P_1x^n$			$3D_\oplus + L(n/2)$

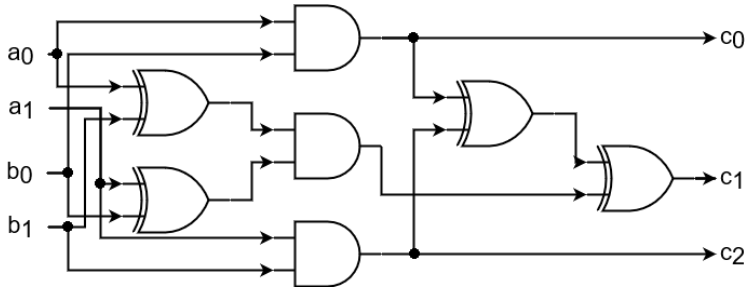


Fig. 3.1: A schematic of 2-bit KA multiplication.

The recursive latency complexity of original KA can be summarized as,

$$L(n) = 3D_\oplus + L(n/2).$$

The initial latency is $L(1) = D_\otimes$. Noting that the 2-bit KA has no overlap as shown in Figure 3.1, so that $L(2) = 2D_\oplus + D_\otimes$. For $n \leq 4$, each KA recursion will add $3D_\oplus$ to the total latency. Hence, the non-recursive latency complexity can be derived as,

$$L(n) = (3\log_2 n - 1)D_\oplus + D_\otimes$$

3.2.2 2-Way KA's Extensions

This section outlines 2-way methods extended from the original KA, optimized for either area or latency complexity.

3.2.2.1 2-Way Reconstructed KA

The reconstructed KA method suggested in [21, 22] can reduce XOR gate count in polynomial multiplication compared with original KA. Let $GF(2^n)$ and elements A, B be defined in the same way as original KA. The product C is,

$$\begin{aligned} C =& AB \\ =& A_0(x)B_0(x) + (A_0(x)B_1(x) + A_1(x)B_0(x))x^{\frac{n}{2}} + A_1(x)B_1(x)x^n \\ =& (P_0 + P_1x^{\frac{n}{2}})(x^{\frac{n}{2}} + 1) + P_2x^{\frac{n}{2}}, \end{aligned} \tag{3.9}$$

where the notation for partial products is the same with (3.4).

The component process and sub-multiplications require the same gate counts as original KA, which is shown in Table 3.1. The totally gate count to implement reconstruction process can be obtained as $\frac{5n}{2} - 3$ XOR gates. Adding the component process and sub-multiplications shown in Table 3.1, recursive area complexity of reconstructed KA can be summarized. For latency complexity this method exhibits the same latency complexity as the original KA, which is 3 additional $D_\oplus$ for each recursion. One $D_\oplus$ can be subtracted due to no overlap in $L(2)$, as shown in Figure 3.1. Therefore, non-recursive complexities of reconstructed KA method can be summarized as,

$$\begin{cases} G_\otimes(n) = n^{\log_2 3}, \\ G_\oplus(n) = \frac{11}{2}n^{\log_2 3} - 7n + \frac{3}{2}, \\ L(n) = (3\log_2 n - 1)D_\oplus + D_\otimes. \end{cases} \tag{3.10}$$

3.2.2.2 2-Way Overlap-Free KA

An overlap-free method was proposed to speed up KA multiplication by dividing the input operands into odd and even terms [23]. The overlap-free method doesn't decrease the area complexity to the original KA, but it reduces latency. Let $GF(2^n)$

be defined in the same way as original KA. Let A and B are two elements in $GF(2^n)$ with degree $n-1$. In overlap-free method, A and B can be expressed as,

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

where

$$A_0(x^2) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i} x^{2i}, \qquad A_1(x^2) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1} x^{2i},$$
$$B_0(x^2) = \sum_{i=0}^{\frac{n}{2}-1} b_{2i} x^{2i}, \qquad B_1(x^2) = \sum_{i=0}^{\frac{n}{2}-1} b_{2i+1} x^{2i}.$$

$A_j(x^2)$ and $B_j(x^2)$ for $j \in 0, 1$ can be seen as equal-length polynomials with degree $\frac{n}{2}-1$. A_0 and B_0 contains all the terms of even degree, while A_1 and B_1 contains all the terms of odd degree. Let $y = x^2$, the produce of A and B can be expressed as,

$$\begin{aligned} C =& AB \\ =& A_0(y)B_0(y) + (A_0(y)B_1(y) + A_1(y)B_0(y))x + A_1(y)B_1(y)y \\ =& P_0 + (P_2 + P_1 + P_0)x + P_1 y, \end{aligned} \tag{3.11}$$

$$\begin{cases} P_0 = A_0(y)B_0(y), \;\; P_1 = A_1(y)B_1(y), \\ P_2 = (A_0(y) + A_1(y))(B_0(y) + B_1(y)). \end{cases} \tag{3.12}$$

Noting that a polynomial multiplied by x or x^2 in hardware implementation can be easily done by shifting the polynomial's coefficients by 1 or 2 bits, which means no additional gates are required or additional latency happened. In reconstruction process of (3.11) one $D_{\oplus}$ can be eliminated by this method when compared with original KA.

The multiplier's XOR gate count is $4n-4$ plus 3 sub-multiplications; its AND

gate count consists of 3 sub-multiplications. Latency is 2 XOR gate delay and 1 sub-multiplication. It can be seen that for each recursion, this method has one less XOR gate delay than the original KA. Its non-recursive formulas can be summarized as,

$$\begin{cases} G_{\otimes}(n) = n^{\log_2 3}, \\ G_{\oplus}(n) = 6n^{\log_2 3} - 8n + 2, \\ L(n) = 2\log_2 n D_{\oplus} + D_{\otimes}. \end{cases}$$

It is faster than other 2-way KA's extensions, but still much slower than the SBM.

3.2.2.3 2-Way Block Recombination Method

Block recombination method was first presented for Toeplitz matrix-vector multiplications in [49], and later [24] applied and advanced this method to polynomial multiplications. This method decomposes KA multiplications into three blocks and calculates the block's complexity separately.

From input to output, block recombination method decomposes the KA into the following three blocks.

- Component polynomial formation (CPF): For any element A in $GF(2^n)$, $CPF_n(A) = \hat{A}$, where $\hat{A}$ performs an array with $\log_2 3$ bits.
- Component multiplication (CM): This block consists of parallel AND gates, which can process bit-wise multiplications. $CM(\hat{A}, \hat{B}) = \hat{A} \otimes \hat{B} = \hat{C}$
- Reconstruction (RC): RC involves the reconstruction process of KA multiplications. It recursively using reconstruction process to compute $\hat{C}$ with $\log_2 3$ bits down to C with $2n-2$ bits. $RC_n(\hat{C}) = C$.

Among these blocks, the CM is the simplest to construct, *i.e.*, $n^{\log_2 3}$ AND gates in parallel. The detailed description with proofs for the functions of CPF, CM and

RC are in [24]. The complexities of CPF and RC can be simply represented by computing non-recursive formulas for the component process and the reconstruction process coupled with the sub-multiplications for a certain KA method. For example, if original KA is applied, following steps illustrate computation of the area complexity to $G_{\oplus}^{CPF}(n)$, $G_{\otimes}^{CM}(n)$ and $G_{\oplus}^{RC}(n)$, and latency complexity to $L^{CPF}(n)$, $L^{CM}(n)$ and $L^{RC}(n)$.

For example, if reconstructed KA is applied, the complexity of these block can be shown as follows,

$$\begin{cases} G_{\oplus}^{CPF}(n) = n^{\log_2 3} - n, \ G_{\otimes}^{CM}(n) = 6n^{\log_2 3}, \\ G_{\oplus}^{RC}(n) = \frac{7}{2}n^{\log_2 3} - 5n + \frac{3}{2}, \\ L^{CPF}(n) + L^{CM}(n) + L^{RC}(n) = 3(\log_2 n - 1)D_{\oplus} + D_{\otimes}. \end{cases} \tag{3.13}$$

The block recombination method requires one more piece of content to be fully executed, which is the two-multiplication-and-add structure. Let A and B be two elements in $GF(2^n)$, $A = A_0 + A_1 x^{\frac{n}{2}}$, $B = B_0 + B_1 x^{\frac{n}{2}}$, where A_i and B_i are equal length with degree $\frac{n}{2} - 1$ for $i \in \{0, 1\}$. 2-way block recombination expand the $C = AB$ as

$$AB = A_0 B_0 + (A_0 B_1 + A_1 B_0) x^{\frac{n}{2}} + A_1 B_1 x^n.$$

It can be performed by 4 $\frac{n}{2}$-bit KA multiplications as $C_{i,j} = A_i B_j$, $i, j \in \{0, 1\}$. For $C_{0,1} + C_{1,0}$, one RC block can be omitted by (3.14). A $2n - 1$ bit-wise addition is replaced by a component addition (CA) block consisting of XOR gates.

Lemma 2 ([24]). *Let $\hat{C}$ and $\hat{C}'$ be two arrays of $n^{\log_2 3}$ bits. Let RC_n be the reconstruction function, we have:*

$$RC_n(\hat{C}) + RC_n(\hat{C}') = RC_n(\hat{C} + \hat{C}'). \tag{3.14}$$

According to the multiplication architecture after the recombination, we can summarize the following The area and latency complexity to different blocks are summarized below.

$$\begin{cases} G_{\otimes}(n) = 4G_{\otimes}^{CM}(\frac{n}{2}), \\ G_{\oplus}(n) = 4G_{\oplus}^{CPF}(\frac{n}{2}) + 3G_{\oplus}^{RC}(\frac{n}{2}) + G_{\oplus}^{CA}(\frac{n}{2}) + n - 2, \\ L(n) = L^{CPF}(\frac{n}{2}) + L^{CM}(\frac{n}{2}) + L^{RC}(\frac{n}{2}) + 2D_{\oplus}. \end{cases}$$

The complexities of blocks are shown in (3.13). Therefore, the area and latency complexities of 2-way block recombination can be summarized as,

$$\begin{cases} G_{\otimes}(n) = \frac{4}{3}n^{\log_2 3}; \\ G_{\oplus}(n) = \frac{31}{6}n^{\log_2 3} - \frac{17}{2}n + \frac{5}{2}; \\ L(n) = (3\log_2 n - 1)D_{\oplus} + D_{\otimes}. \end{cases}$$

The 2-way block recombination method saves XOR gate count and increases AND gate count compared to reconstructed KA, with a minor reduction in total gate count.

3.2.3 4-Way KA and Its Extensions

4-way KA has a more complex reconstruction process that provides more room for further improvements, resulting in better area and latency complexity than 2-way KA.

3.2.3.1 4-Way KA

A(x) and $B(x)$ with degree $n-1$ can be divided into four equal-length polynomials with degree $\frac{n}{4} - 1$ shown as follows,

$$A(x) = \sum_{i=0}^{n-1} a_i x^i = \underbrace{\sum_{i=0}^{\frac{n}{4}-1} a_i x^i}_{A_0} + x^{\frac{n}{4}} \underbrace{\sum_{i=0}^{\frac{n}{4}-1} a_{i+\frac{n}{4}} x^i}_{A_1} + x^{\frac{n}{2}} \underbrace{\sum_{i=0}^{\frac{n}{4}-1} a_{i+\frac{n}{2}} x^i}_{A_2} + x^{\frac{3n}{4}} \underbrace{\sum_{i=0}^{\frac{n}{4}-1} a_{i+\frac{3n}{4}} x^i}_{A_3},$$

$$B(x) = \sum_{i=0}^{n-1} b_i x^i = \underbrace{\sum_{i=0}^{\frac{n}{4}-1} b_i x^i}_{B_0} + x^{\frac{n}{4}} \underbrace{\sum_{i=0}^{\frac{n}{4}-1} b_{i+\frac{n}{4}} x^i}_{B_1} + x^{\frac{n}{2}} \underbrace{\sum_{i=0}^{\frac{n}{4}-1} b_{i+\frac{n}{2}} x^i}_{B_2} + x^{\frac{3n}{4}} \underbrace{\sum_{i=0}^{\frac{n}{4}-1} b_{i+\frac{3n}{4}} x^i}_{B_3}.$$

which can be shortened as

$$A = A_0 + A_1 x^{\frac{n}{4}} + A_2 x^{\frac{n}{2}} + A_3 x^{\frac{3n}{4}}, \text{ and } B = B_0 + B_1 x^{\frac{n}{4}} + B_2 x^{\frac{n}{2}} + B_3 x^{\frac{3n}{4}}. \tag{3.15}$$

The product of A and B can be performed as,

$$\begin{aligned} AB =& P_0 + (P_0 + P_1 + P_2) x^{\frac{n}{4}} + (P_0 + P_1 + P_3 + P_6) x^{\frac{n}{2}} \\ &+ (P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8) x^{\frac{3n}{4}} \\ &+ (P_1 + P_3 + P_4 + P_7) x^n + (P_3 + P_4 + P_5) x^{\frac{5n}{4}} + P_4 x^{\frac{3n}{2}} \end{aligned} \tag{3.16}$$

where the partial products $P_i, i \in [0, 8]$ can be expressed as,

$$\begin{cases} P_0 = A_0 B_0, \quad P_1 = A_1 B_1, \\ P_2 = (A_0 + A_1)(B_0 + B_1), \\ P_3 = A_2 B_2, \quad P_4 = A_3 B_3, \\ P_5 = (A_2 + A_3)(B_2 + B_3), \\ P_6 = (A_0 + A_2)(B_0 + B_2), \\ P_7 = (A_1 + A_3)(B_1 + B_3), \\ P_8 = (A_0 + A_1 + A_2 + A_3)(B_0 + B_1 + B_2 + B_3). \end{cases} \tag{3.17}$$

Table 3.3: Complexity of 4-way KA without reconstruction process.

Computation	$G_{\oplus}$	$G_{\otimes}$	L
P_0, P_1, P_3, P_4	$4G_{\oplus}(\frac{n}{4})$	$4G_{\otimes}(\frac{n}{4})$	$L(\frac{n}{4})$
P_2, P_5, P_6, P_7	$4(G_{\oplus}(\frac{n}{4}) + \frac{n}{2})$	$4G_{\otimes}(\frac{n}{4})$	$L(\frac{n}{4}) + D_{\oplus}$
P_8	$G_{\oplus}(\frac{n}{4}) + \frac{n}{2}$	$G_{\otimes}(\frac{n}{4})$	$L(\frac{n}{4}) + 2D_{\oplus}$

The gate counts of components process and sub-multiplication implemented with (3.17) can be calculated in Table 3.3. For example, to carry out the partial products, P_0, P_1, P_3, and P_4, only 4 sub-multiplications are needed. Both XOR gate and AND gate counts come from these sub-multiplications, and the latency is one sub-multiplication's latency. For P_2, P_5, P_6, and P_7 ,the XOR gate count comes from 4 sub-multiplications and $\frac{n}{2}$ XOR gates used to calculate components such as $A_0 + A_1$. Their AND gate count also comes from 4 sub-multiplications. The latency of P_2, P_5, P_6, and P_7 can be determined by one sub-multiplication with $\frac{n}{2}$ XOR gate delay. For P_8, the XOR gate count arises from 1 sub-multiplication, and one XOR to compute the component such as $A_0 + A_1 + A_2 + A_3$. Note that the components like $A_0 + A_1$ and $A_2 + A_3$ have been carried out by computation of P_2, P_5, P_6, and P_7, therefore, only one parallel layer of XOR is needed. The AND gate count arises from one sub-multiplication. The latency to produce P_8 one sub-multiplication's latency plus two XOR gate delay.

Typically, previous research enhance the formula (3.16) before utilizing it to a reconstruction process. This formula is inefficient to implement with CMOS based gates because it involves multiple overlaps.

3.2.3.2 4-Way Reconstructed Method

It is shown that the reconstruction process in (3.16) can be optimized following these steps [21].

$$R_0 = P_0 + P_1 x^{\frac{n}{4}} + P_3 x^{\frac{n}{2}} + P_4 x^{\frac{3n}{4}},$$

$$R_1 = R_0 + R_0 x^{\frac{n}{4}},\ R_2 = R_1 + P_2 x^{\frac{n}{4}} + P_5 x^{\frac{3n}{4}}$$
$$R_3 = R_2 + R_2 x^{\frac{n}{2}},\ R_4 = P_6 + P_7 x^{\frac{n}{4}}$$
$$R_5 = R_4 + R_4 x^{\frac{n}{4}} + P_8 x^{\frac{n}{4}}. \tag{3.18}$$

The calculation of area complexity is the same as 2-way KA, hence it is skipped here. About latency complexity, we see from $(P_0, P_1, P_3, P_4) \rightarrow R_0 \rightarrow R_1 \rightarrow R_2 \rightarrow R_3$ the latency in reconstruction process involves 4 XOR gates delay. The latency from $(P_2, P_5), \rightarrow R_2 \rightarrow R_3$ also includes 3 XOR gates delay, which is the same as $(P_6, P_7) \rightarrow R_4 \rightarrow R_5$. From $P_8 \rightarrow R_5$ the latency has 2 XOR gates delay. In conjunction with the latency of component and sub-multiplication process in Table 3.3, we can conclude that each round of KA produces 5 XOR gates delay. In short, the complexity of reconstructed KA is summarized as,

$$\begin{cases} G_{\otimes}(n) = n^{\log_2 3}, \\ G_{\oplus}(n) = \frac{217}{40} n^{\log_2 3} - \frac{34}{5} n + \frac{11}{8}, \\ L(n) = \frac{5}{2} \log_2 n D_{\oplus} + D_{\otimes}. \end{cases}$$

One can notice that a KA multiplier reconstructed using 4-way block recombination is superior to its 2-way version regarding both gate count and latency, when n is a power of 4. In the following paragraphs, we will provide a brief overview of the 4-way block recombination method.

3.2.3.3 4-Way Block Recombination Method

4-way block recombination is evolved from of 2-way block recombination, and it exhibits reduced area complexity compared to the 4-way reconstructed KA method [24]. Its reconstruction blocks can be implemented using either an area-saving or a latency-saving method, resulting in optimal complexities in two different orientations. The

n-bit operands $A(x)$ and $B(x)$ can be divided into four equal-length polynomials with degree $\frac{n}{4}-1$, shown as (3.15). Then their product can be expressed as,

$$AB = C_1 + C_1 x^{\frac{n}{4}} + C_2 x^{\frac{n}{2}} + C_3 x^{\frac{3n}{4}} + C_4 x^{n} + C_5 x^{\frac{5n}{4}} + C_6 x^{\frac{3n}{2}}, \tag{3.19}$$

where

$$\begin{aligned} C_0 &= A_0 B_0, \;\; C_1 = A_0 B_1 + A_1 B_0, \\ C_2 &= A_0 B_2 + A_1 B_1 + A_2 B_0, \\ C_3 &= A_0 B_3 + A_1 B_2 + A_2 B_1 + A_3 B_0, \\ C_4 &= A_1 B_3 + A_2 B_2 + A_3 B_1, \\ C_5 &= A_2 B_3 + A_3 B_2, \;\; C_6 = A_3 B_3. \end{aligned} \tag{3.20}$$

The blocks and extra XOR gates requirement can be calculated from (3.19) and (3.20), and the area and latency complexities can be summarized as follows,

$$\begin{cases} G_{\otimes}(n) = 16 G_{\otimes}^{CM}(n), \\ G_{\oplus}(n) = 8 G_{\oplus}^{CPF}(\frac{n}{4}) + 7 G_{\oplus}^{RC}(\frac{n}{4}) + 9 G_{\oplus}^{CA}(\frac{n}{4}) + \frac{3n}{2} - 6; \\ L(n) = L^{CPF}(\frac{n}{4}) + L^{CM}(\frac{n}{4}) + L^{RC}(\frac{n}{4}) + 3D_{\oplus}. \end{cases} \tag{3.21}$$

Noting that the latency of these three blocks $L^{CPF}(\frac{n}{4})+L^{CM}(\frac{n}{4})+L^{RC}(\frac{n}{4})$ is equivalent to the delay of a 4-way KA of size $\frac{n}{4}$.

Table 3.4: Complexity of blocks in the 4-way block recombination

	$G_{\oplus}^{CPF}(n)$	$G_{\oplus}^{RC}(n)$	$G_{\otimes}^{CM}(n)$	$L^{CPF}(n)+L^{CM}(n)+L^{RC}(n)$
Area	$n^{\log_2 3} - n$	$\frac{137}{40} n^{\log_2 3} - \frac{24}{5} n + \frac{11}{8}$	$n^{\log_2 3}$	$\frac{5}{2}\log_2 n D_{\oplus} + D_{\otimes}$
Latency	$n^{\log_2 3} - n$	$\frac{31}{8} n^{\log_2 3} - 6n + \frac{17}{8}$	$n^{\log_2 3}$	$2\log_2 n D_{\oplus} + D_{\otimes}$

The area and latency complexity regarding reconstruction blocks is summarized in Table 3.4. The left column displays the two designs for the block recombination

method, area-oriented (area) and latency-oriented (latency). The latency-oriented method is the 4-way KA method proposed in [24], which has lower latency complexity. The middle three columns are the area complexities of CPF, RC and CM blocks with respect to the input size. The right column is the latency required for these three blocks. By combining Table 3.4 and equation (3.21), we can determine the complexities of both the area-oriented and latency-oriented 4-way block recombination methods.

For the area-oriented method:

$$\begin{cases} G_{\otimes}(n) = \frac{16}{9} n^{\log_2 3}, \\ G_{\oplus}(n) = \frac{1639}{360} n^{\log_2 3} - \frac{89}{10} n + \frac{29}{8}, \\ L(n) = (\frac{5}{2} \log_2 n - 2) D_{\oplus} + D_{\otimes}. \end{cases} \tag{3.22}$$

For the latency-oriented method:

$$\begin{cases} G_{\otimes}(n) = \frac{16}{9} n^{\log_2 3}, \\ G_{\oplus}(n) = \frac{353}{72} n^{\log_2 3} - 11n + \frac{71}{8}, \\ L(n) = (2 \log_2 n - 1) D_{\oplus} + D_{\otimes}. \end{cases} \tag{3.23}$$

It becomes apparent that the two methods are the most efficient among all the existing methods mentioned in this chapter, in terms of area complexity and latency complexity, respectively, when n is an integer power of 4.

3.2.4 Mixed Algorithm

In [50], a "mixed" algorithm that combined the SBM with KA was used to search for the best space complexity for $3 \leq n \leq 128$. This method was also used in [51] to design multipliers of several sizes in the range of $113 \leq n \leq 283$ for FPGA implementation.

3.2.5 K-way Split KA, $k \neq 2, 4$

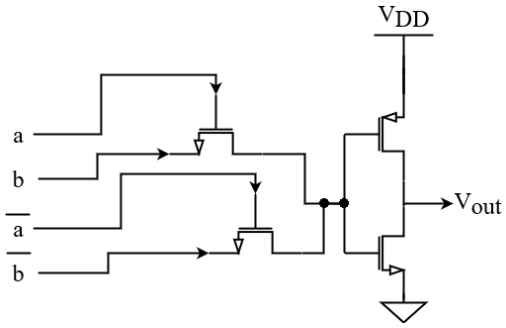
There are KA architectures with other splits, for example, the 3-way KA can use 6 sub-multiplications to achieve an area complexity of $n^{\log_3 6} \approx n^{1.63}$[52]. Montgomery presented presented 5, 6, and 7 split KA architectures and listed the bounds with repect to the exponent of n [30]. Research [53, 28, 26, 54] further improved multi-split KA and obtained better bounds than [30].

3.3 Logic Gate Implementation

The finite field multiplier described earlier employs the 2-input AND and the 2-input XOR gate as its fundamental logic gates.

3.3.1 CMOS XOR Gates

If we consider area-saving as a priority, Pass-Transistor Logic (PTL) is a better choice [55, 56] than conventional CMOS logic. A schematic of 2-input XOR gate designed with PTL is shown in Figure 3.2.



a	b	V_{out}
0	0	0
1	0	1
0	1	1
1	1	0

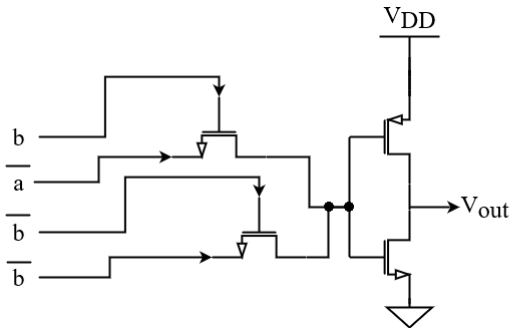
Fig. 3.2: 2-input XOR gate designed using PTL and its truth table

The XOR gate is designed using only 4 transistors, 3 n-MOS and 1 p-MOS. It operates in the following manner. If both input signals (a and b) are logic "low", the n-MOS connected to $\overline{a}$ and $\overline{b}$ generates a logic "low" signal, which is then inverted to produce a logic "high" output. (The output is preceded by an inverter consisting

of an n-MOS and a p-MOS; the other n-MOS is off) If both input signals are logic "high", the n-MOS linked to them generates a logic "low" signal (the other n-MOS is off), which is also inverted to produce a logic "high" output. Finally, if two inputs, one is logic "high" and the other one is logic "low", the two n-MOS on the left both deliver logic "low", causing the output to be logic "high".

3.3.2 CMOS AND Gates

A schematic of 2-input AND gate designed with PTL is shown in Figure 3.3.



a	b	V_{out}
0	0	0
1	0	0
0	1	0
1	1	1

Fig. 3.3: 2-input AND gate designed using PTL and its truth table

From the schematic on the left in Figure 3.3, it can be seen that when b is logic "low", the upper left n-MOS is off, and lower left n-MOS performs logic "high" signal. Passing the inverter, and output shows logic "low". When both b and a are logic "high", the lower left n-MOS is off, the upper left n-MOS performs logic "high", and output would be logic "low". When b is logic "high" and a is logic "low", both the left 2 n-MOS generate logic-low signals, and output should be logic "high".

3.4 Single Electron Technology

One of the most potential candidates for the next generation of nanoelectronics is the single electron technology [57], which enables the manipulation and control of either one or a small group of electrons. It is believed that the initial detection of Coulomb blockade, which led to the discovery of single electron technology, was likely

first performed by Gorter [58] in 1951. The research on single-electron device physics did not see significant activity until the mid-1980s, when Averin and Likharev [59, 60] introduced the single-electron transfer oscillation and single electron transistor, which sparked a renewed interest in the area of research.

3.4.1 Single Electron Transistor

The single electron transistor (SET) is an emerging nanoelectronics device that operates on the principle of controlling the flow of individual electrons through a small circuit. It consists of two tunnel junctions constructed by ultrathin dielectrics, and a conductor between the tunnel junctions as island. Thick dielectrics coupled to the island serve as gates, each of which controls the flow of electrons through the SET, as shown 3.4.

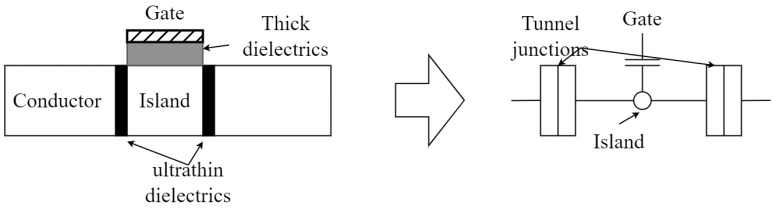


Fig. 3.4: Schematic and simple layout of a single electron transistor

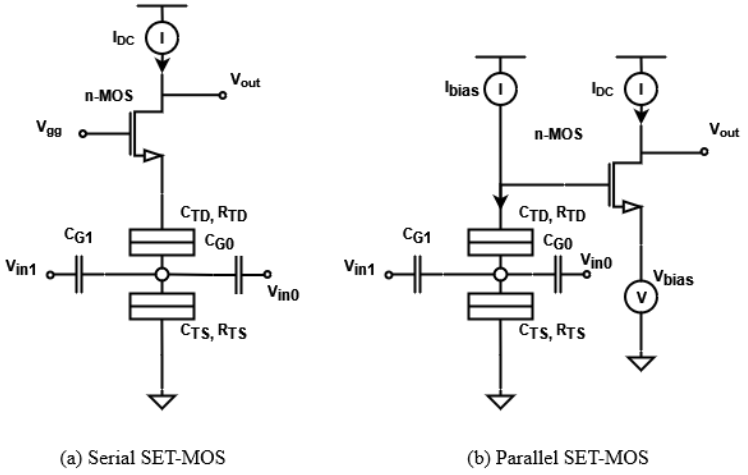
SET is distinguished by its exceptionally small size, minimal power consumption, and distinctive Coulomb blockade oscillation characteristics. By leveraging the unique Coulomb blockade oscillation of the SET, one can achieve novel functionalities with fewer transistor count. However, pure SET has its shortcomings, such as low current drive capability, lack of room temperature operations, and small voltage gain.

3.4.2 Hybrid SET-MOS Transistors

Research has shown that hybrid SET and CMOS (SET-MOS) transistor is an excellent solution to pure SET's intrinsic drawbacks [61]. The unique Coulomb blockade oscillations of SET provide it with advantages in low power consumption and new

functionality, while CMOS compensates for the intrinsic drawbacks of SET with its high-speed driving and voltage gain [32].

We design logic gates based on SET-MOS, and when it comes to simulation, the Monte Carlo technique is regarded as the most precise approach to simulate SET, using probability calculations. Nevertheless, this technique becomes time-consuming when simulating large circuits and is not suitable for co-simulation with CMOS gates. The compact model of SET known as MIB (initials of the author's name [62]) offers fast simulation and is attractive for hybrid CMOS and SET co-simulation. The model is developed using the Master Equation technique and has been verified with perfect match to the Monte Carlo result. The Verilog-A version of MIB model can be readily integrated into Cadence using the Verilog-A interface [63]. In this thesis, the co-simulations of SET-MOS gates are performed using Cadence Spector simulator integrated the MIB compact model, along with CMOS technology of $45nm$ generic process design kits(GPDK45).



(a) Serial SET-MOS (b) Parallel SET-MOS

Fig. 3.5: Two popular SET-MOS structures

Figure 3.5 shows two commonly utilized SET-MOS structures. The serial SET-MOS structure was first put forward in [64] with many applications such as random-access memory [64] and analog-to-digit converter [64, 65]. Our designs are based on parallel SET-MOS structure, which was first launched in [66] to increase the current

drivability. To achieve a high voltage gain, the n-MOS followed by SET in Figure 3.5(b) is biased in the sub-threshold region. Specifically, V_{in0} is used to adjust the initial phase of voltage oscillation at the SET's drain terminal, while V_{bias} compensates for the gate-to-source voltage of the n-MOS transistor. The n-MOS transistor is biased with a constant current, allowing the oscillation to be transferred to the output node with an amplified amplitude.

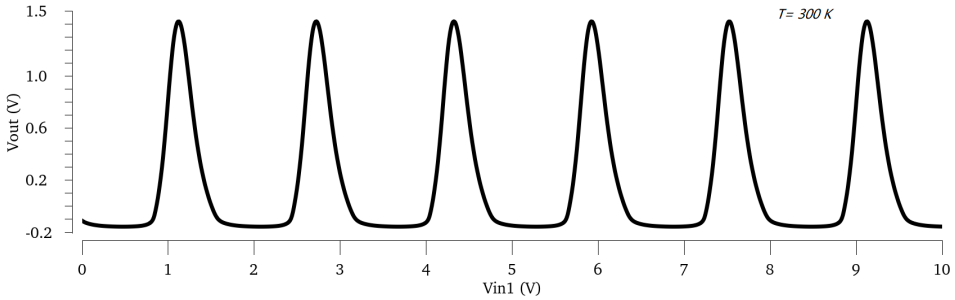


Fig. 3.6: A typical voltage transfer characteristics (VTC) of Figure 3.5(b).

When the voltage of V_{in1} increases linearly, V_{out} is shown in Figure 3.6 with parameters of $C_{TS} = C_{TD} = C_{G1} = C_{G2} = 0.1aF$, $I_{bias} = 80nA$, $V_{in0} = 0, V_{bias} = -180mV$, $I_{DC} = 1uA$, $R_{TS} = R_{TD} = 1M\Omega$. It can be seen that SET-MOS can produce an oscillating waveform at room temperature thanks to the Coulomb blocking effect of SET and the compatibility with CMOS devices. By adjusting the parameters of the circuit, such as increasing or decreasing the bias current and voltage, input capacitance, periodicity of oscillation, initial phase, amplitude, the peak and valley of the waveform can be changed. In short, SET-MOS provides great flexibility, but designing SET-MOS circuits with appropriate parameters to generate stable and suitable oscillation waveform can be a challenging task.

4 Proposed SET-MOS XOR Gates and Hybrid SBM-KA Multiplier

4.1 Introduction

Bit-parallel multipliers are crucial for achieving high-speed computation and ensuring the efficient operation of the elliptic curve cryptosystem. However, implementing SBM (the schoolbook/classical method for bit-parallel multiplier) requires a significant area (gate count). Fortunately, the introduction of KA (Karatsuba algorithm for bit-parallel multiplier) has substantially reduced the required area for implementing bit-parallel multipliers. Much of recent work on parallel finite field multipliers is based on or related to KA. Notably, there are reconstructed KA [21], overlap-free KA [23], block recombination method [24], and "mixed" methods [50, 51].

Noting that most research progress in this regard has been made using traditional CMOS technologies. Further performance improvement may be possible from an algorithmic and architectural perspective, but the room seems limited. Research shows that new nanodevices, such as SET devices, can play an essential role in this challenge from a device-level point of view [67]. With their unique oscillation characteristics and the compatibility with CMOS, SET-MOS can provide high flexibility and area savings in implementing logic operations, especially multiple-input XOR logic.

4.2 Multiple-Input XOR Gates Using SET-MOS

In this section, multiple-input XOR gates are designed using SET-MOS, simulated, and compared with CMOS counterparts.

4.2.1 Multiple-Input XOR Operation

A multiple-input XOR has the same logic as a 2-input XOR shown in Figure 3.2. It reveals a pattern that shows when the sum of all inputs that equal to logic "1" is odd, and the output is logic "1"; if the sum of all inputs that match logic "1" is even, the output is logic "0". When it comes to a 4-input XOR operation, the truth table can be shown as follows.

Table 4.1: Truth table of 4-input XOR operation

Input "a"	Input "b"	Input "c"	Input "d"	Sum of inputs	Output
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	0	1	1
1	0	0	0	1	1
0	0	1	1	2	0
0	1	1	0	2	0
1	1	0	0	2	0
0	1	0	1	2	0
1	0	0	1	2	0
1	0	1	0	2	0
0	1	1	0	2	0
0	1	1	1	3	1
1	0	1	1	3	1
1	1	0	1	3	1
1	1	1	0	3	1
1	1	1	1	4	0

We have added a column "Sum of input" to Table 4.1. By comparing the "Sum of input" and output columns, it can be seen that the output is 1 for odd "Sum of input" and 0 for even "Sum of input." Furthermore, this correspondence between input and output perfectly aligns with the oscillation characteristic of SET. CMOS technology typically constructs multi-input XOR gates using multiple 2-input XOR gates, where the input number is directly proportional to the transistor count (an indicator of the area performance of the multiplier in this thesis). Conversely, SET devices can incorporate multiple inputs with a single gate, as demonstrated by the fabrication

of multiple-input SET devices [68], which have been used in various applications [69, 70]. By adjusting the parameters of a multiple-input SET-MOS, we can design a multiple-input XOR gate with fewer transistors than its CMOS counterpart.

4.2.2 4-Input SET-MOS XOR Gates

A 4-input XOR gate can be developed by adding inputs and a CMOS inverter to the parallel MIB structure shown in Figure 3.5(b). The expected VTC of SET for the XOR operation from Table 4.1 can be sketched in Figure 4.1. Then we need to adjust the parameters to make the SET-MOS gate follow with this VTC.

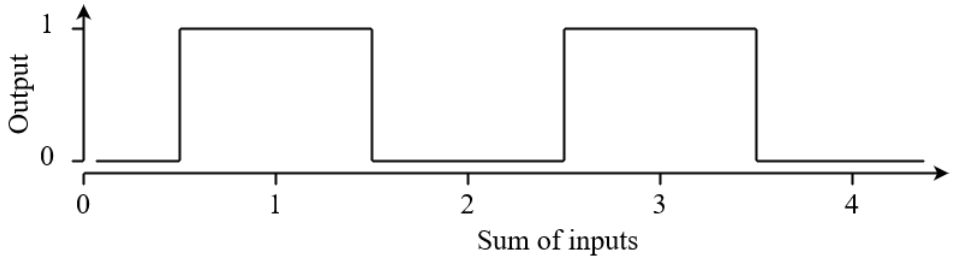


Fig. 4.1: Expected VTC of SET's input and output to a 4-input XOR gate.

Regarding the selection of parameters, such as logic high/low values, and oscillation periodicity. A fundamental assumption can be made that the minimum value of capacitance is $0.1aF$, and let C_{TD}, C_{TS}, and C_G be equal to this value in this thesis. The periodicity of SET's VTC can be obtained by $\frac{e}{C_G}$ [66], where e is the electronic charge ($1.6 \times 10^{-19}C$), and C_G is the gate capacitance ($0.1aF$ in this case)—the oscillation periodicity can be concluded as $1.6V$. The logic high value of the input and output can be set to $0.8V$, *i.e.*, one-half periodicity of the VTC, and the logic low value to 0. To determine the structure of a 4-input XOR, three identical inputs can be added to the SET (with all C_G values being $0.1aF$) followed by a CMOS inverter to boost its driving capability. Figure 4.2 displays a multiple-input SET-MOS structure that comprises a 4-input SET-MOS XOR gate.

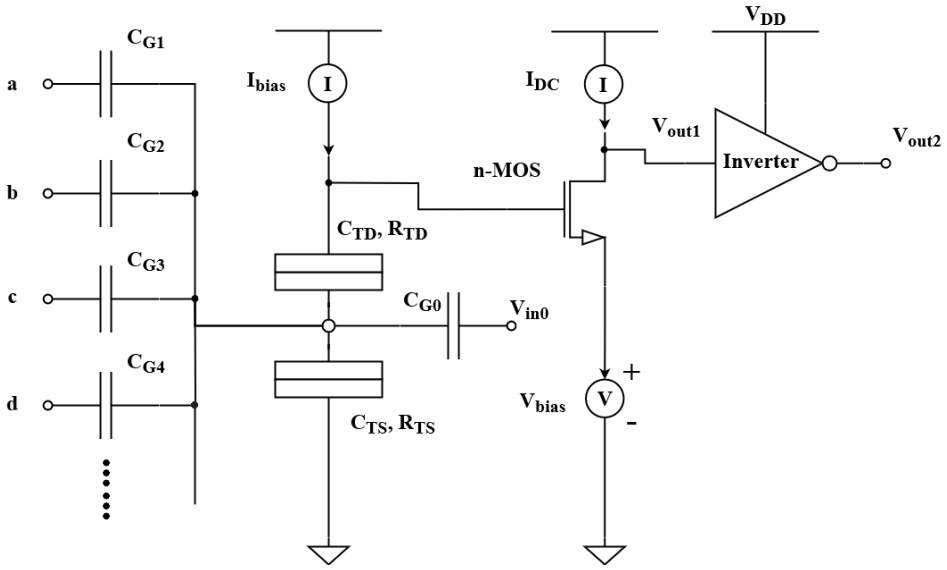


Fig. 4.2: A generic SET-MOS structure for multi-input logic gate, such as XOR.

To figure out the remaining parameters, some simulations on VTC of SET-MOS (V_{in-i} to V_{out1}, excluding the CMOS inverter) are shown in Figure 4.3 for ranging I_{bias} from $50nA$ to $90nA$ and V_{bias} from $-180mV$ to $350mV$, while keeping $C_{TS} = C_{TD} = C_{G1} = C_{G2} = 0.1aF$, $V_in0 = -100mV$, $I_{DC} = 1uA$, $R_{TS} = R_{TD} = 1M\Omega$. We aim to make the peak and valley values of the oscillation waveform closer to $0.8V$ and $0V$, respectively. It seems that the parameters of $I_{bias} = 65nA$ and $V_{bias} = -265mV$ are closer to our target. In addition, when V_{out1} goes through the inverter and outputs V_{out2}, whether it can reach 50% duty cycle is also a key consideration.

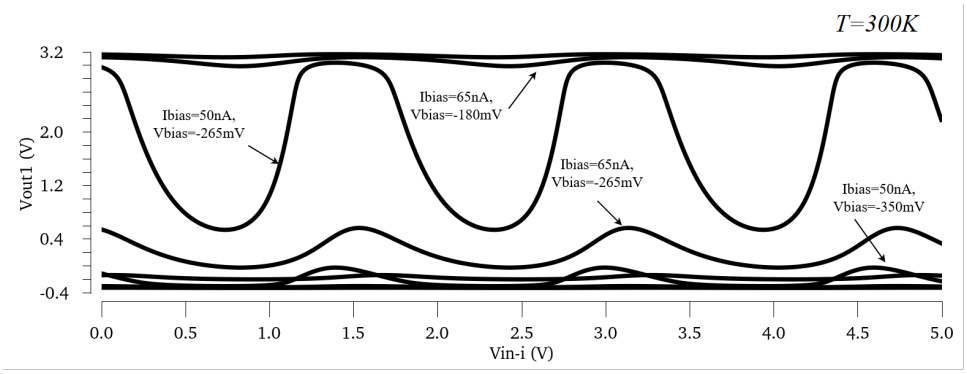


Fig. 4.3: VTCs of SET-MOS via different bias currents and voltages.

After several attempts, the results of Figure 4.4 were obtained with the parameters

shown in Table 4.2.

Table 4.2: Parameters for a 4-input SET-MOS XOR gate implemented in Fig.4.2

Device \ Variable	Parameter	Value
SET	C_{TD}, C_{TS}, C_{Gi}	$0.1aF$
	R_{TD}, R_{TS}	$1M\Omega$
Bias current	I_{DC}	$1uA$
	I_{bias}	$65nA$
Bias voltage	V_{in0}	$-100mV$
	V_{bias}	$-250mV$
	V_{DD}	$0.8V$
p-MOS	V_{th}	$-0.56V$
	W	$360nm$
	L	$45nm$
n-MOS	V_{th}	$0.59V$
	W	$120nm$
	L	$45nm$
Temperature	T	$300K$

It is worth mentioning that the width of the p-MOS to about three times that of the n-MOS where the width of n-MOS transistors is $120nm$ and the width of p-MOS transistors is $360nm$. It improves the noise margin of the circuit as stated in reference [71]. It can be seen that the output V_{out2} is $0.8V$ (*i.e.*, logic "high") if and only if the sum of all input voltages is an odd multiple of the logic "high" value (each input can be either logic high of $0.8V$ or logic low of $0V$), implementing an XOR logic.

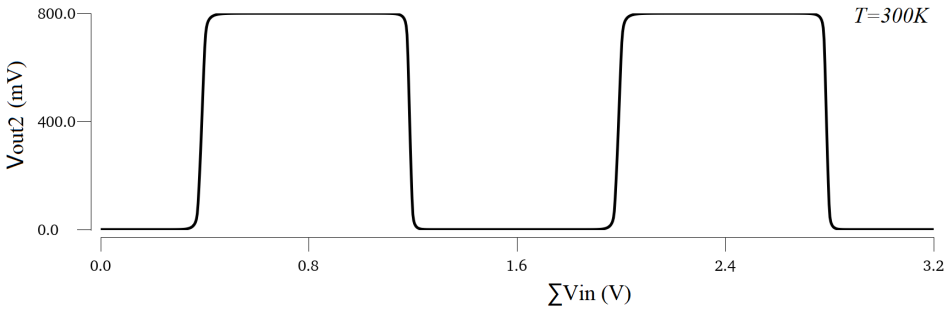


Fig. 4.4: Simulated VTC of a 4-input XOR gate from Figure 4.2

Notably, the horizontal coordinate of Figure 4.4 is ΣV_{in}, where $\Sigma V_{in} = V_{in1} + V_{in2} + V_{in3} + V_{in4}$. This is equivalent to using V_{in-i} as the horizontal coordinate. Because all gate capacitors are identical, each input has the same influence on the potential of the SET's island [32].

With slightly modified circuit parameters, Figure 4.2 can implement an XOR gate with a different number of inputs. For instance, if the bias voltages and currents of Table 4.2 change to$V_{in0} = -92.5mV$,$I_{bias} = 105nA$ and $V_{bias} = -125mV$ while keeping the same for the rest of parameters, one can implement a 2-input XOR gate. The 3-input XOR gates can also be alternatively implemented using a 4-input XOR gate with one input grounded. Theoretically, Figure 3.1 can be used to implement XOR gates with any number of inputs. However, gates with more inputs introduce larger capacitance which would generally require very low temperature for reliable operation. This is due to the fact that the maximum temperature is inversely proportional to the total capacitance with SET transistor [61]. Typically, with the total capacitance around $1aF$, hybrid SET-MOS circuits can operate reliably under the room temperature of $T = 300K$, which is used throughout this thesis.

4.2.3 SET-MOS Versus CMOS

It can be seen from Figure 4.2 that with SET-MOS structure, a 4-input XOR gate requires three MOS transistors, one SET transistor, and five capacitors. In contrast, implementing the 4-input XOR gate in CMOS technology generally requires three 2-input XOR gates. Assuming each 2-input XOR gate consists of four transistors on average using pass-transistor logic (PTL) plus an inverter being added at the output, the total number of transistors can be estimated as $3 \times 4 + 2 = 14$. In other words, the number of transistors increases linearly with the number of inputs for XOR gates in traditional CMOS technology. This area-saving feature makes the SET-MOS technology a promising candidate for area-efficient implementation of finite

field multiplications which involve a large number of multiple-input XOR operations.

Table 4.3: Area cost of logic gates using CMOS versus SET-MOS

Logic gate	2-input XOR		4-input XOR		2-input AND
Technology	SET-MOS	CMOS	SET-MOS	CMOS	CMOS
Transistor count	4	4	4	14	4
Capacitor count	3	-	5	-	-

It should be noted that the transistor count remains the same using hybrid SET-MOS technology for both 2-input and 4-input XOR gates with slightly different delay. As shown in Table 4.3, the hybrid SET-MOS technology saves nearly $\frac{2}{3}$ of the transistor count when implementing a 4-input XOR operation, thanks to the Coulomb oscillation characteristics with SET. We can disregard the capacitor in SET when estimating the circuit area since it is too small to have an impact.

In short, the Coulomb Blockade empowers SET-MOS with the convenience of designing multi-input logic and makes it a promising choice for efficiently implementing finite field multiplications that need many multiple-input XOR networks while conserving space.

4.3 Multiplications Improved by SET-MOS XOR Gates

Implementation of polynomial multiplication in $GF(2^n)$ would require both XOR and AND gates. This section proposes SET-MOS based methods to implement several bit-parallel polynomial multiplication architectures over $GF(2^n)$ where AND gate is implemented with CMOS PTL and XOR networks are realized with SET-MOS XOR_4. In the following of this thesis, we will use XOR_2, XOR_4, AND_2 to denote 2-input and 4-input XOR gates, and 2-input AND gates, respectively. Before discussing the multiplier implementation, let us introduce the scope of this research and how the area performance of multipliers be evaluated.

4.3.1 Area Performance Evaluation

Existing research on area efficiency of multiplication methods at the architecture point of view typically uses asymptotic area complexity as a metric for comparison, as demonstrated in [24, 23, 72, 29, 73]. Other studies [49, 24, 74, 75] consider specific ranges of n and the area cost of XOR/AND gates to compare the area efficiency of different multiplication architectures.

However, emerging SET devices and utilization of PTL design push the scope of this research to the device level, which requires a more reasonable approach to evaluating the area efficiency of multiplication architectures. We use the total transistor count of a multiplier implementation as a metric to compare area performance. As shown in Table 4.3, all gates involved in this work consist of 4 transistors. Therefore, we can also use gate count to estimate the area cost of a multiplier. Although a more accurate area estimation would require physical layout design, gate count/transistor count can serve as a reasonable measure of area cost.

4.3.2 SBM Using SET-MOS XOR Gates

The utilization of SET-MOS XOR_4 can effectively reduce the area complexity of the numerous XOR networks present in the SBM. Suppose A and B are two elements in $GF(2^n)$ represented with polynomial basis of degree $n-1$, the product of A and B can be given in (3.1). The general process of SBM can be understood as follows:

- First, the input polynomials A and B are fed into a parallel layer of 2-input AND gates, generating $a_i b_j$s, $0 \leq i, j \leq n-1$.
- These signals, $a_i b_j$s, are then passed through XOR networks of size k for $k < n$ (or $2n-2-k$ for $k \geq n$), resulting in output c_ks.

Implementing an XOR network of size n in CMOS technology requires $n-1$ 2-input XOR gates. However, calculating the XOR gate count becomes complicated when

dealing with multiple input XORs. We have made a program to calculate the XOR gate count required for an n-bit schoolbook multiplication when m-input XOR gates are available, as shown in algorithm 4.1. An m-input XOR can also implement XOR with less than inputs by simply grounding the excess inputs. For example, algorithm 4.1 can be used to determine the XOR gate count for a 256-bit SBM with 4-input XOR by setting the input parameters m to 4 and n to 256.

Algorithm 4.1 Counting m-input SET-MOS XOR gates in the n-bit SBM

Input: m, n; #m is input number of XOR gate, n is multiplication size;
Output: $G_{\oplus}$; # $G_{\oplus}$ is XOR gate count;
Initialization of variables: $G_{\oplus}, i, j \leftarrow 0$;

1: **for** $i := 2$ to n **do**
2: $j \leftarrow i$;
3: **while** $j < m$ **do**
4: $G_{\oplus} \leftarrow G_{\oplus} + Div(j, m)$; # $Div(a, b)$ returns integer of a divided by b
5: $j \leftarrow Div(j, m) + Mod(j, m)$; # $Mod(a, b)$ returns reminder of a divided by b
6: **end while**
7: $G_{\oplus} = G_{\oplus} + 1$;
8: **end for**
9: **for** $i := n - 1$ downto 2 **do**
10: $j \leftarrow i$;
11: **while** $j < m$ **do**
12: $G_{\oplus} \leftarrow G_{\oplus} + Div(j, m)$;
13: $j \leftarrow Div(j, m) + Mod(j, m)$;
14: **end while**
15: $G_{\oplus} = G_{\oplus} + 1$;
16: **end for**
17: **return** $G_{\oplus}$;

In addition to using computer programs, we have also derived mathematical formulas to calculate the XOR gate count in the SBM. With SBM, item c_k in (3.1) is a sum of multiple $a_i b_j$s for $k = 1, 2, \ldots, 2n - 3$. Note that c_k has $k + 1$ terms of $a_i b_j$s when $1 \leq k \leq n - 1$, and $(2n - k - 1)$ terms of $a_i b_j$s when $n \leq k \leq 2n - 3$. The gate count

of XOR_4 for SBM can be estimated as,

$$G_{\oplus}(n) = (2n-3)\lceil\frac{n-1}{3}\rceil - 3\lfloor\frac{n-2}{3}\rfloor\lceil\frac{n-1}{3}\rceil \leqslant \frac{n^2}{3} + \frac{n}{3}. \tag{4.1}$$

Noting that a 2-input or 3-input XOR operation can be implemented using a SET-MOS XOR_4 with the redundant inputs grounded. The gate count of AND_2 is n^2, and the critical path delay is $L(n) = (\log_4 n)D_{X4} + D_{\otimes}$, where D_{X4} denotes the latency of one SET-MOS XOR_4. The area and latency complexity of SBM implemented by CMOS AND_2 and SET-MOS XOR_4 can be summarized as,

$$\begin{cases} G_{\otimes}(n) = n^2, \\ G_{\oplus}(n) \approx 0.33n^2 + 0.33n, \\ L(n) = \frac{1}{2}\log_2 nD_{X4} + D_{\otimes}. \end{cases} \tag{4.2}$$

Compared with the CMOS-based SBM, as shown in (3.2), SBM improved by SET-MOS XOR_4 saves approximately 67% in terms of XOR gate count, and about 33% in terms of total gate count.

4.3.3 Original 2-Way KA Using SET-MOS XOR Gates

Using SET-MOS XOR_4 can significantly reduce the XOR gate count of the original KA. Let A and B be two polynomials over $GF(2^n)$ with degree $n-1$, which can be divided into two equal-length polynomials with degree $\frac{n}{2}-1$, shown as $A = A_0 + A_1 x^{\frac{n}{2}}$ and $B = B_0 + B_1 x^{\frac{n}{2}}$. Original KA method can be explained in (3.4) and (3.5). We rewrite (3.4) here for convenience, and the computation involved in this equation can

be illustrated in Table 4.4.

$$\begin{cases} P_0 = A_0B_0, \ \ P_1 = A_1B_1, \\ P_2 = (A_0 + A_1)(B_0 + B_1), \\ AB = P_0 + (P_2 + P_1 + P_0)x^{\frac{n}{2}} + P_1x^n. \end{cases} \tag{4.3}$$

Table 4.4: (4.3) implemented by SET-MOS XOR_4

Degree range	Operation	SET-MOS implementation
$[x^0, x^{\frac{n}{2}-1}]$	P_0	–
$[x^{\frac{n}{2}}, x^{n-2}]$	$P_0x^{-\frac{n}{2}} + (P_0 + P_1 + P_2)$	$(\frac{n}{2} - 1)$ XOR_4s
x^{n-1}	$P_0 + P_1 + P_2$	1 XOR_4
$[x^n, x^{\frac{3}{2}n-2}]$	$(P_0 + P_1 + P_2)x^{-\frac{n}{2}} + P_1$	$(\frac{n}{2} - 1)$ XOR_4s
$[x^{\frac{3}{2}n-1}, x^{2n-2}]$	P_1	–
$[x^0, x^{2n-2}]$	$P_0 + (P_0 + P_1 + P_2)x^{\frac{n}{2}} + P_1x^n$	$(n - 1)$ XOR_4s

It can be seen from Table 4.4 that only $n - 1$ SET-MOS XOR_4s are used to implement (4.3). Note that the gate count in the component process for implementing P_2 is n. Therefore, one round of KA requires $(n-1)+n = 2n-1$ XOR_4 gates along with three half-sized multipliers. The latency incurred is $2D_{X4}$. The complexity for one round of 2-way KA using SET-MOS XOR gates can be given as

$$\begin{cases} G_{\otimes}(n) = 3G_{\otimes}(\frac{n}{2}), \\ G_{\oplus}(n) = 3G_{\oplus}(\frac{n}{2}) + 2n - 1, \\ L(n) = L(\frac{n}{2}) + 2D_{X4}. \end{cases} \tag{4.4}$$

Compared to the CMOS-based method to implement KA (3.6), the gate count saved in (4.4) is $(4n-4)-(2n-1) = 2n-3$. The initial 1-bit multiplier has $G_{\oplus}(1) = 0$, $G_{\otimes}(n) = 1$ and $L(1) = D_{\otimes}$. When applying KA recursively to n-bit multiplication in

$GF(2^n)$, the gate count and the critical path delay can be estimated using (3.7) as

$$\begin{cases} G_{\otimes}(n) = n^{1.58}, \\ G_{\oplus}(n) = 3.5n^{1.58} - 4n + 0.5, \\ L(n) = 2\log_2(n)D_{X4} + D_{\otimes}. \end{cases} \tag{4.5}$$

Since $n^{\log_2 3}$ is approximately equal to 1.58, we use 1.58 to replace $n^{\log_2 3}$ in the area complexity in the subsequent this thesis.

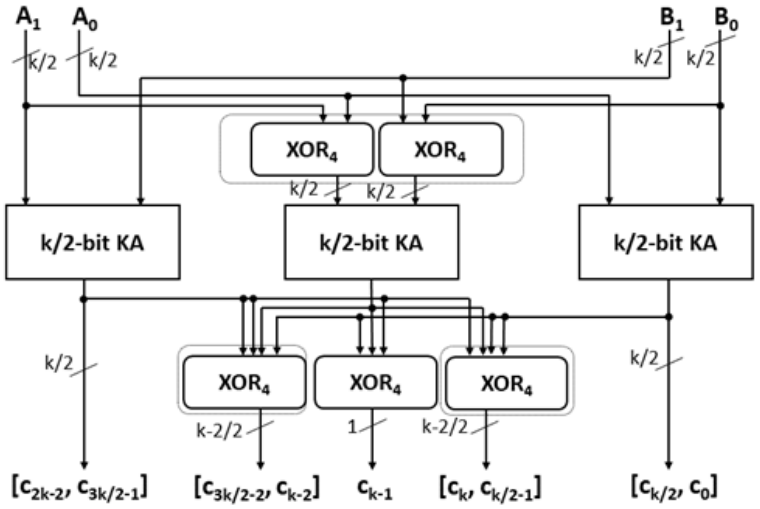


Fig. 4.5: Data flow of original 2-way KA using SET-MOS XOR_4

Figure 4.5 shows the data flow of one round of KA using SET-MOS XOR_4. Its architecture is straightforward compared to pure CMOS work, and each round of KA only adds two layers of parallel SET-MOS XOR_4 before and after the sub-multiplications.

Comparing (4.5) and (3.8), it can be found that when $n = 256$, the XOR reduction of KA using SET-MOS XOR_4 compared to CMOS counterpart can reach $(37320 - 21940)/37320 = 41\%$. However, the gate count of AND_2 in the CMOS KA multiplier is not reduced. When considering the total gate count, which includes both AND and XOR gates, the drop is about 35% for the CMOS-based 2-way KA.

4.3.4 2-Way KA's Extensions Using SET-MOS XOR Gates

This subsection will investigate the application of SET-MOS XOR gates to several popular existing KA's extensions to improve their area performance.

4.3.4.1 2-Way Reconstructed KA Using SET-MOS XOR Gates

The application of SET-MOS XOR_4 to the reconstructed KA leads to a reduction in its area complexity. Consider A_i and B_i, $i \in \{0, 1\}$, discussed in (3.3) and P_0, P_1, and P_2 in (4.3). The multiplication AB with three steps reconstructions from (3.9) can be summarized as,

1) $R_0 = P_0 + P_1 x^{\frac{n}{2}}$.

2) $R_1 = R_0 + R_0 x^{\frac{n}{2}}$.

3) $AB = R_1 + P_2 x^{\frac{n}{2}}$.

The SET-MOS XOR_4 has the capability to process four inputs simultaneously, thus allowing for the condensation of the KA method's three steps into just two, as shown below.

I) $R = P_0 + P_1 x^{\frac{n}{2}}$: Both P_0 and P_1 are polynomials with $n-1$ bits (of degree $n-2$). After shifting $\frac{n}{2}$ bits, from degree $\frac{n}{2}$ to $n-2$, each degree has two bits overlap. Thus, XOR requirement in this step is $\frac{n}{2} - 1$.

II) $AB = R + Rx^{\frac{n}{2}} + P_1 x^{\frac{n}{2}}$: From degree $\frac{n}{2}$ to $\frac{3n}{2} - 2$, each degree has 3 bits in overlap. Therefore, $n-1$ XOR_4s are needed with one input grounded.

This reconstruction process requires $1.5n - 2$ XOR_4 in total, besides additional $n-1$ XOR_4 in the component process. The recursive XOR gate count for one round of reconstructed KA can be given as

$$G_{\oplus}(n) = 3G(\frac{n}{2}) + 2.5n - 2, \tag{4.6}$$

It is clear that the AND gate has the same complexity as (3.10). The latency complexity is increased by three D_{X4}s per round of recursion. The area and latency complexity of reconstructed KA improved by SET-MOS XOR_4 can be obtained as,

$$\begin{cases} G_{\otimes}(n) = n^{1.58}, \\ G_{\oplus}(n) = 4n^{1.58} - 5n + 1, \\ L(n) = (3\log_2 n - 1)D_{X4} + D_{\otimes}. \end{cases} \tag{4.7}$$

Comparing (4.7) and (3.10), it can be found that when $n = 256$, the XOR reduction of reconstructed KA using SET-MOS XOR_4 compared to its CMOS counterpart reaches $(34259 - 24965)/34259 = 27\%$. Total gate count drop is about 23%.

4.3.4.2 2-Way Overlap-Free KA Using SET-MOS XOR Gates

After utilizing SET-MOS XOR_4, the overlap-free method, like other KA methods that have applied SET-MOS XOR_4, exhibits a decrease in area complexity. Let A and B be two elements in $GF(2^n)$ with degree $n-1$. In the overlap-free method, A and B can be expressed as $A = A_0 + A_1 x$ and $B = B_0 + B_1 x$, respectively. A_0 and B_0 involves all terms of the even degree of A and B, while A_1 and B_1 involve all terms of the odd degree of A and B. With SET-MOS XOR_4, the reconstruction function of the overlap-free method can be rewritten as follows,

$$\begin{cases} P_0 = A_0 B_0, \;\; P_1 = A_1 B_1, \\ P_2 = (A_0 + A_1)(B_0 + B_1), \\ AB = P_0 + (P_2 + P_1 + P_0)x + P_1 x^2. \end{cases} \tag{4.8}$$

In each round, the added XOR gate count and latency can be analyzed as follows,

- Components process: Pre-process the input operands $A_0 + A_1$ and $B_0 + B_1$ for

P_2, n XOR gates are required.

- Sub-multiplications: Three multiplications of size $\frac{n}{2}$.
- Reconstruction process: $P_0 + P_1 + P_2$, each partial product is of degree $n-2$, which can be implemented by $n-1$ XOR_4 with one input grounded. $P_0 + P_2$ needs $n-2$ XOR_4.

Therefore, the recursive area and latency complexities of the overlap-free method improved by SET-MOS XOR_4 can be summarized as,

$$\begin{cases} G_{\otimes}(n) = 3G_{\otimes}(\frac{n}{2}), \\ G_{\oplus}(n) = 3G_{\oplus}(n) + 3n - 3, \\ L(n) = L(\frac{n}{2}) + 2D_{\oplus}. \end{cases} \tag{4.9}$$

And the non-recursive complexities of the overlap-free method improved by SET-MOS XOR_4 are shown as follows,

$$\begin{cases} G_{\otimes}(n) = n^{1.58}, \\ G_{\oplus}(n) = 4.5n^{1.58} - 6n + 1.5, \\ L(n) = 2\log_2 nD_{X4} + D_{\otimes}. \end{cases} \tag{4.10}$$

For the overlap-free method when $n = 256$, the XOR gate and total gate count reduction by using SET-MOS XOR_4 are about 25% and 16%, respectively.

4.3.4.3 2-Way Block Recombination Method Using SET-MOS XOR Gates

The data flow for 2-way block recombination using SET-MOS XOR_4 can be described as follows. After operands A and B be divided into two equal-length parts, A_0, A_1 and B_0, B_1, they go through $4CPF$ blocks and generate $(\hat{A}_i)$ and $(\hat{B}_i)$ for $i \in \{0, 1\}$ with $n^{\log_2 3}$ bits. These vectors (or arrays) are grouped two by two. Two groups

$\hat{A}_0\hat{B}_0$ and $\hat{A}_1\hat{B}_1$ perform two CM blocks, the others, $\hat{A}_0\hat{B}_1$ and $\hat{A}_1\hat{B}_0$, through two CA and one CA blocks. After passing three RC blocks, the results of RC shift to the most significant bit, then perform a bit-wise addition consisting of $n-2$ parallel XOR gates. (This is because each $C_{i,j}, i, j \in \{0,1\}$ has $n-1$ bits. After shifting, the overlap is $2(\frac{n}{2}-1) = n-2$ bits). To calculate the area complexity, we see there are four CPF blocks, four CM blocks, three RC blocks and one CA with $n-2$ XOR gates for the final addition. For latency complexity, one of the critical paths is from B_0 to C, which is composed of the latency of one CPF, one CM, one CA, one RC and one XOR gate delay. The multi-input XOR operations can only be found in the RC blocks where SET-MOS XOR_4 can reduce their gate count. For a lower area complexity, the reconstructed KA improved by SET-MOS XOR_4 is considered to implement the RC blocks, where its area complexity can be shown as follows,

$$G_{\oplus}^{RC}(n) = 2n^{\log_2 3} - 3n + 1. \tag{4.11}$$

Bring $\frac{n}{2}$ into the RC formulas in (3.13) and (4.11), multiplying by 3 and subtracting. As a result, $(\frac{3}{2}n^{\log_2 3} - 3n + \frac{3}{2})$ XOR gates can be saved by applying SET-MOS XOR_4. And therefore, The area and latency complexities of 2-way block recombination improved by SET-MOS XOR_4 are summarized as,

$$\begin{cases} G_{\otimes}(n) = \frac{4}{3}n^{1.58}, \\ G_{\oplus}(n) = \frac{11}{3}n^{1.58} - \frac{11}{2}n + 1, \\ L(n) = (3\log_2 n - 1)D_{X4} + D_{\otimes}. \end{cases} \tag{4.12}$$

Applying SET-MOS XOR_4 reduces approximately 28% of the XOR gate count and 23% of the total gate count for the block recombination method when $n = 256$.

4.3.5 Improved Methods Versus Existing Ones

Figure 4.6 compares the total gate counts between the SET-MOS XOR_4 improved methods and CMOS-based methods presented in the previous section for n equal to $[2^8, 2^{10}]$. Comparing the hatched and blank bars in the figures indicates that the SBM [18] in 4.6(a) and the original KA [19] in 4.6(b) get significant gate count drop after using SET-MOS XOR_4.

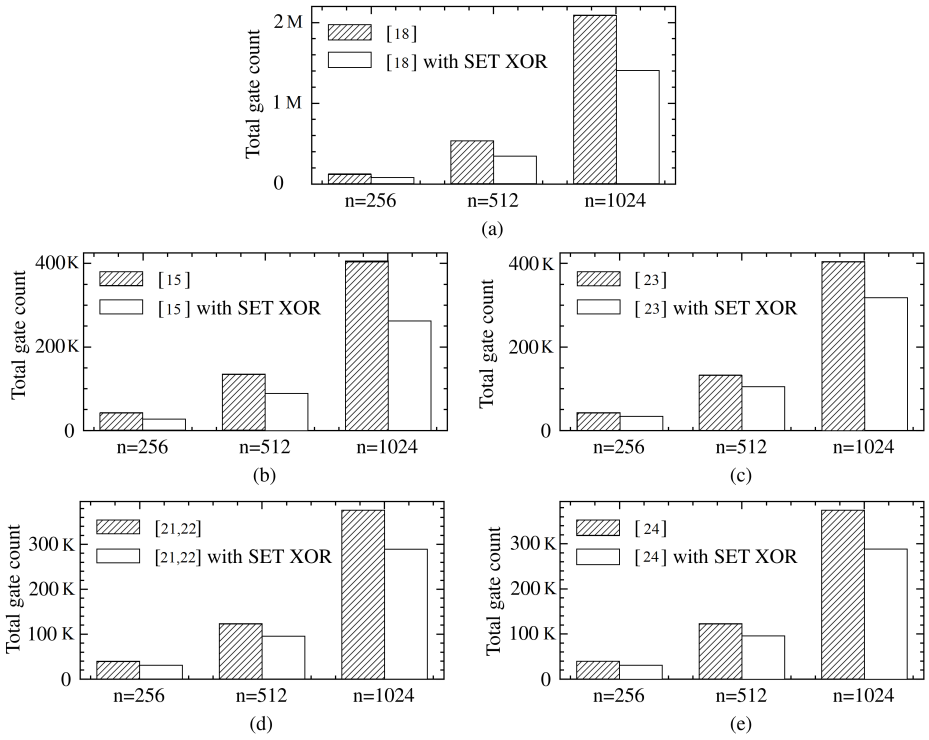


Fig. 4.6: Multipliers using SET-MOS XOR_4 versus their CMOS counterparts

The other three 2-way KA's extensions shown in 4.6(c), 4.6(d), 4.6(e) are not as well adapted to SET-MOS XOR_4 as 4.6(a) and 4.6(b). This is because the capability of SET-MOS, which processes multiple inputs with less area cost, has changed the way for optimization of the reconstruction process in KA. In other words, these extension methods based on CMOS XOR gates only partially exploit the capability of multi-input processing when integrated with SET-MOS XOR gates. It can also be seen that the original KA with SET-MOS XOR_4 has the lowest total gate count for the

same n value. (*i.e.*, for $n = 1024$, we can see original KA with SET-MOS XOR_4 in Figure 4.6(b) reaches a total gate count of around $280K$, while Figure 4.6(c) is over $300K$, and both Figure 4.6(d) and Figure 4.6(e) exceed $290K$.)

Worth noting that, when compared with the most efficient existing method of block recombination [24], a reduction of $(6.5 - 4.5)/6.5 = 30.7\%$ in area complexity has been achieved asymptotically for $n \gg 1$ with the proposed SET-MOS XOR improved KA. When we bring $n = [2^5, 2^{10}]$ into the area complexity formulas, it can be concluded that the proposed SET-MOS XOR's improved KA has a $26 - 30\%$ improvement in area performance compared with the most area-efficient existing method.

4.4 SBM-KA Multiplication Architecture

In this section, we advance the idea in [51] by providing a general formula for calculating the area complexity of hybrid architecture using SET-MOS XOR gates when the multiplier sizes are $n = 2^k$, where k is greater than or equal to 2.

4.4.1 Base Multiplier

Consider the proposed SBM with the complexity shown in (4.2) and the complexity of the 2-way KA improved by SET-MOS XOR_4 given in (4.4). Clearly, compared to KA multiplier with sub-quadratic complexity, SBM requires the higher gate count due to its quadratic complexity for large n, the multiplier size. However, SBM could require fewer gates than KA method when n is sufficiently small. In Table 4.5, we list the gate count of SBM and KA multiplier improved by XOR_4 in $GF(2^n)$, $n = 2^k$ for small values of k.

Table 4.5: Gate counts of SBM and 2-way KA using SET-MOS XOR_4 for $n = 2^k$

$\#AND_2 + \#XOR_4$					
k	**1**	**2**	**3**	4	5
SBM	$4+1$ $= \mathbf{5}$	$16+5$ $= \mathbf{21}$	$64+21$ $= \mathbf{85}$	$256+85$ $= 341$	$1024+341$ $= 1365$
KA	$3+3$ $= \mathbf{6}$	$9+16$ $= \mathbf{25}$	$27+63$ $= \mathbf{90}$	$81+220$ $= 301$	$243+723$ $= 966$

The complexity of the base multiplier and the reconstruction process of KA are both impacted by the value of k. It is imperative that the value of k is carefully examined and considered to make the multiplier more area-efficient. To determine the optimal value of k that minimizes the total number of gates required for the entire multiplication process, we initially computed the gate count for n = 256, 512, and 1024 with varying values of k (1, 2, and 3) using a program.

Algorithm 4.2 Total gate count in 2-way SBM-KA multiplication architecture

Input: m, n; #m is base multiplier size, n is entire multiplier size.
Output: G; # G is XOR gate count plus AND gate count, $G = G_{\oplus} + G_{\otimes}$.
Initialization of variables:
$G, G_{\oplus}, G_{\otimes} \leftarrow 0$; # Entire multiplier's gate counts.
$G^B_{\oplus} = Algorithm$; $4.1(m, 4)$; $G^B_{\otimes} = m^2$ #Base multiplier's gate count.
$k \leftarrow m$; #label for sub-multiplication size.

```
1: while k ≤ m do
2:   if k = m then
3:     G⊕ ← G⊕^B;
4:     G⊗ ← G⊗^B;
5:     k ← 2k;            # Base multiplication completed;
6:   else
7:     G⊕ ← 3G⊕ + 2k − 1   # k-bit KA improved by SET-MOS XOR4 in (4.4);
8:     G⊗ ← 3G⊗;
9:     k ← 2k;            # Next round of KA multiplication;
10:  end if
11: end while
12: G = G⊕ + G⊗;
13: return G;
```

The program used to calculate the total number of gates for hybrid multiplication is

presented in Algorithm 4.2. The input parameter m represents the size of the base multiplier, which is a positive integer power of 2, $m = 2^{k1}$; and n represents the size of the entire multiplication, which is also a positive integer power of 2 for $n = 2^k$, $0 < k1 < k$.

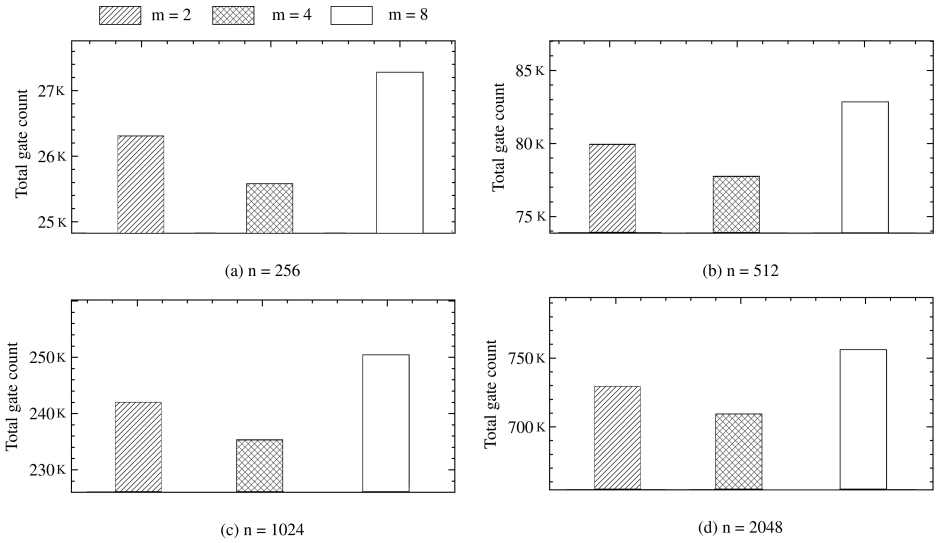


Fig. 4.7: Total gate counts of n-bit SBM-KA constructed from different m-bit base multipliers.

Total gate count of SBM-KA multiplications for $n = 2^k$, $k \in \{8, 9, 10, 11\}$, where the base multiplier size m ranges from 2^j, $j \in \{1, 2, 3\}$ are shown in Figure 4.7. Based on the four graphs Figure 4.7(a)-(d), it can be concluded that the optimal size for the base multiplier is 4, which results in the minimum number of gates for the multiplier when $n \in \{256, 512, 1024\}$.

In addition to the program calculation, we also derived a generic formula to calculate the asymptotic area complexity of SBM-KA multiplication, shown in (4.14).

Lemma 3. *Let a, b, k, $k1$ be positive integers. Let $m = n^{k1}$, $n = b^k$ and assume $a \neq b$, $a \neq 1$. The solution to the recurrence relations*

$$\begin{cases} R_m = e; \\ R_n = aR_{n/b} + cn + d. \end{cases} \tag{4.13}$$

is shown as follows

$$R_n = (e + \frac{bc}{a-b}m + \frac{d}{a-1})(\frac{n}{m})^{\log_b a} + \frac{-bc}{a-b}n + \frac{-d}{a-1}. \tag{4.14}$$

Proof is given in Appendix B. By bringing the formulas of (4.2) and (4.4) for XOR and AND gate counts into (4.14), the area complexity of hybrid multiplication can be expressed as follows,

$$G(n) \approx (1.33m^2 + 4.33m - 0.5)(\frac{m}{n})^{1.58} - 4n + 0.5, \tag{4.15}$$

where $G(n)$ denotes the total gate count and can be calculated as $G(n) = G_{\oplus}(n) + G_{\otimes}(n)$. Since Table 4.5 shows the value of $m = 2^k$ for $k = 1, 2, 3$ would make the area complexity of hybrid multiplication minimum, we used the SBM improved by SET-MOS XOR_4 of size $m = 2, 4$, and 8 as the bases of the recursive KA and took them into (4.15). The results show that the total gate count is asymptotically lowest when $m = 4$. The derivation of the area and latency complexities of this 2-way SBM-KA multiplication for $m = 4$ will be explained in the following subsection.

4.4.2 2-Way SBM-KA Using SET-MOS XOR Gates

By setting the base to be a 4-bit SBM with XOR_4, the recursive KA method can be modified to build a polynomial multiplier in $G(2^n)$.

$$\begin{cases} G_{\otimes}(4) = 16, \\ G_{\oplus}(n) = 5, \\ L(4) = D_{X4} + D_{\otimes}, \\ G_{\otimes}(n) = 3G_{\otimes}(\frac{n}{2}), \\ G_{\oplus}(n) = 3G_{\oplus}(\frac{n}{2}) + 2n - 1, \\ L(n) = L(\frac{n}{2}) + 2D_{X4}. \end{cases} \tag{4.16}$$

Note that the 4-bit SBM multiplier requires 16 AND_2 and 5 XOR_4 gates and has a critical path delay of D_{X4}+$D_{\otimes}$. Thus, the recursive complexity formulas of the proposed 2-way hybrid SBM-KA can be shown in (4.16) recursively for $n \geq 8$.

Non-recursive area complexity expressions for the proposed 2-way SBM-KA multiplier using XOR_4 can be obtained from their recursive relation (4.16) as

$$\begin{cases} G_{\otimes}(n) = 1.78n^{1.58}, \\ G_{\oplus}(n) = 2.28n^{1.58} - \frac{11}{2}n + 1, \\ L(n) = 2\log_2(\frac{n}{4})D_{X4} + D_{X4} + D_{\otimes}. \end{cases} \tag{4.17}$$

The total gate count for the proposed 2-way hybrid multiplication architecture, SBM-KA multiplier, can be obtained as,

$$G(n) = G_{\otimes}(n) + G_{\oplus}(n) = 4.06n^{1.58} - 4n + 0.5. \tag{4.18}$$

This area complexity of the 2-way SBM-KA multiplier, when compared with (4.5) for

the original KA, suggests that the number of XOR gates is substantially reduced. In contrast, the AND gate count increases moderately. Consequently, the sum of XOR and AND gates required for the hybrid SBM-KA method using SET-MOS XOR_4 is significantly lower than the original KA improved by SET-MOS XOR_4.

4.4.3 Latency Complexity Reduction

By inspecting the structure of the SBM-KA multiplier using SET-MOS XOR_4, we found that certain SET-MOS XOR_4 on the critical path are used to implement 2-input XOR operations. For example, the implementation of the P_2 in component process in each round of KA, as shown in (3.4), would contribute one D_{X4} to the multiplier's latency. If CMOS XOR gates are utilized instead in implementing P_2, then the latency complexity can be reduced by the difference of $D_{X4} - D_{\oplus}$ for each round of KA, while the total gate count remains unchanged. Generally speaking, D_{X4} is larger than $D_{\oplus}$ due to the SET's characteristics. Particular gate delays will be discussed in the next chapter.

The proposed SBM-KA method with latency reduction can be described as follows. SET-MOS XOR_4 improves 4-bit SBM and will be used as the 2-way KA recursion base reconstructed by SET-MOS XOR_4. For each round of KA architecture, the operations in (3.4) are implemented with CMOS-based XOR gates, while the computations in Table 4.4 are implemented with SET-MOS XOR_4s. The reduced latency for one round of KA would be $D_{X4} - D_{\oplus}$. Thus, the recursive representation for the reduced latency complexity is given as ($n \geq 8$).

$$\begin{cases} L(4) = D_{X4} + D_{\otimes}, \\ L(n) = L(\frac{n}{2}) + D_{X4} + D_{\oplus}. \end{cases} \tag{4.19}$$

$L(n)$ in (4.17) can be replaced with the shortened recursive latency complexity given

above. Therefore, the non-recursive latency complexity for the 2-way SBM-KA multiplier with latency reduction can be shown as,

$$\begin{aligned} L(n) &= (\log_2(\frac{n}{4}))(D_{X4} + D_{\oplus}) + D_{X4} + D_{\otimes}, \\ &= (\log_2(n) - 1)D_{X4} + (\log_2(n) - 2)D_{\oplus} + D_{\otimes}, \end{aligned} \tag{4.20}$$

while its AND and XOR gate counts remain the same as in (4.17).

4.5 Complexity Comparison

This section contains two comparisons. The first sub-section compares the area complexity of all methods introduced in the last chapter and proposed in this chapter using SET-MOS XOR_4. The second sub-section compares the total gate count of the proposed area-efficient method with existing methods.

4.5.1 Asymptotic Complexitiy Comparison

A comparison between the existing work and the proposed methods, including the SET-MOS XOR_4 improved methods and hybrid SBM-KA, is shown in Table 4.6. It can be seen that there are three sections in the table, where the top and middle sections list the existing work and their SET-MOS XOR_4 improved counterparts, respectively. The bottom section in Table 4.6 shows the proposed SBM-KA with SET-MOS XOR_4.

From the top two sections in the table, it can be seen that the proposed SET-MOS XOR_4 improved SBM, KA, reconstructed KA, overlap-free KA, and block recombination KA require less gate count than their CMOS-based architectures. Among them, the SET-MOS_4's improved KA has the lowest area complexity than any of the reconstruction KA, overlap-free KA and block recombination KA, even though these KA extensions are more efficient than the original KA in area and/or latency complexity

when implemented with CMOS-based gates.

The asymptotic area complexity savings for SET-MOS XOR_4 improved methods, including KA, the reconstruction KA, overlap-free KA and block recombination KA, are respectively 30.8%, 23.1%, 15.4% and 23.1%, compared to the best result of the existing method. In that regard, the KA with SET-MOS XOR_4 has also achieved the largest reduction, with asymptotic area savings of 30.8%.

Note that the asymptotic saving of 30.8% is calculated with $(6.5 - 4.5)/6.5 = 30.8\%$, where the number 4.5 is the coefficient of the highest order term ($n^{1.58}$) in total gate count for the KA multiplier improved by SET-MOS XOR_4 and 6.5 is coefficient of the highest order term in total gate count among all the existing works of sub-quadratic area complexity in comparison.

The proposed hybrid SBM-KA method with SET-MOS XOR_4, shown at the bottom of the table, has a significantly lower gate count than all the existing methods. It can be seen from the table that the total gate count for the proposed 2-way SBM-KA method achieves 37.5% asymptotic savings compared to the best existing method. Additionally, when compared to 2-way KA improved by SET-MOS XOR_4 [76], it can be seen that the asymptotic area complexity saving of 2-way SBM-KA proposed in this section is $(4.5 - 4.06)/4.5 = 9.8\%$.

4.5.2 Total Gate Count Comparison

To further demonstrate the clear advantage of our method in area efficiency, a comparison of total gate count has been made between the proposed 2-way SBM-KA method with SET-MOS XOR_4 and the existing CMOS methods for multiplier size of $n = 32, 64, 128, 256, 512$, and 1024, as shown in Table 4.7. It can be seen that our proposed 2-way SBM-KA with SET-MOS XOR_4 improves the area performance by 34% - 37% compared to the most area-efficient existing method.

Table 4.6: Comparison between the existing methods and the proposed SET-MOS XOR's improved methods

Existing methods					
Methods	#XOR	#AND	#XOR + #AND		Latency
			Gate count	Asym. savings	
SBM[18]	$n^2 - 2n + 1$	n^2	$2n^2 - 2n + 1$	–	$\log_2(n)D_{\oplus} + D_{\otimes}$
KA (2-way)[15]	$6n^{1.58} - 8n + 2$	$n^{1.58}$	$7n^{1.58} - 8n + 2$	–	$((3\log_2 n) - 1)D_{\oplus} + D_{\otimes}$
Reconst. KA (2-way)[21, 22]	$5.5n^{1.58} - 7n + 1.5$	$n^{1.58}$	$6.5n^{1.58} - 7n + 1.5$	–	$((3\log_2 n) - 1)D_{\oplus} + D_{\otimes}$
Overlap-F. (2-way)[23]	$6n^{1.58} - 8n + 2$	$n^{1.58}$	$7n^{1.58} - 8n + 2$	–	$(2\log_2 n)D_{\oplus} + D_{\otimes}$
Block recom. (2-way)[24]	$5.17n^{1.58} - 8.5n + 2.5$	$1.33n^{1.58}$	$6.5n^{1.58} - 8.5n + 2.5$	–	$((3\log_2 n) - 1)D_{\oplus} + D_{\otimes}$
Adapted existing architectures with SET-MOS XOR_4					
Methods	#XOR	#AND	#XOR + #AND		Latency
			Gate count	Asym. savings	
SBM [76]	$0.33n^2 + 0.33n$	n^2	$1.33n^2 + 0.33n$	33%†	$0.5\log_2(n)D_{X4} + D_{\otimes}$
KA (2-way) [76]	$3.5n^{1.58} - 4n + 0.5$	$n^{1.58}$	$4.5n^{1.58} - 4n + 0.5$	30.8%	$2\log_2(n)D_{X4} + D_{\otimes}$
Reconst. KA (2-way)	$4n^{1.58} - 5n + 1$	$n^{1.58}$	$5n^{1.58} - 5n + 1$	23.1%	$(3\log_2(n) - 1)D_{X4} + D_{\otimes}$
Overlap-F. (2-way)	$4.5n^{1.58} - 6n + 1.5$	$n^{1.58}$	$5.5n^{1.58} - 6n + 1.5$	15.4%	$2\log_2(n)D_{X4} + D_{\otimes}$
Block recom. (2-way)	$3.67n^{1.58} - 5.5n + 1$	$1.33n^{1.58}$	$5n^{1.58} - 5.5n + 1$	23.1%	$(3\log_2(n) - 1)D_{X4} + D_{\otimes}$
Proposed SBM-KA with SET-MOS XOR_4					
Methods	#XOR	#AND	#XOR + #AND		Latency
			Gate count	Asym. savings	
SBM-KA (2-way)	$2.28n^{1.58} - 4n + 0.5$	$1.78n^{1.58}$	$\mathbf{4.06n^{1.58} - 4n + 0.5}$	**37.5%**	$\log_2(n) - 1)D_{X4} + (\log_2(n) - 2)D_{\oplus} + D_{\otimes}$

† This is obtained by comparing to the *quadratic* result of the existing work, *i.e.,* $(2 - 1.33)/2 = 33.3\%$, while the other results in the column are obtained by comparing to the best *sub-quadratic* results among the existing work in comparison.

Table 4.7: Comparison of total gate count between the proposed and the existing methods

Gate count / Size of multiplication / Method	32	64	128	256	512	1024
SBM [18]	1985	8065	32513	130052	523265	2095105
Reconstructed KA (2-way) [21, 22]	1357	4292	13321	40856	124357	376652
Overlap-Free KA (2-way) [23]	1447	4593	14287	43881	133687	405153
Block recombination (2-way)[24]	1310	4197	13130	40473	123590	375117
Proposed SBM-KA (2-way)	858	2701	8358	25585	77778	235381
Achieved reduction	34.5%	35.7%	36.5%	36.8%	37.1%	37.3%

Note: Achieved reduction is obtained by comparing the proposed method to the best existing method.

4.6 Summary

The Coulomb oscillation of SET brings excellent flexibility to logic circuit design, particularly finite field multiplication. We designed SET-MOS XOR gates in this chapter and applied them to bit-parallel finite field polynomial multiplications. Compared with the existing sub-quadratic complexity architectures, the proposed method has a significant gate count and area efficiency advantage. Moreover, we proposed a hybrid multiplication architecture - a 2-way SBM-KA method with implementations for further improvement in area and latency. The results of the proposed methods show substantial savings in gate count compared with the existing methods. However, worth noting that the delay of SET is commonly more considerable than CMOS, which may make the proposed SBM-KA method's area superior at the cost of increased latency. The proposed work is expected to be favoured for applications where high-speed elliptic curve computation is required, and the compact area is also critical. With this in mind, the elliptic curve cryptosystems can be more area-efficient as single-electron technology develops and thus have broader applications in future network security.

The proposed approach promises high area efficiency and opens new opportunities for general finite field multiplications, which could change the way current multiplier architecture and algorithms are designed and implemented. More area savings would be likely if further work can be done to explore new architectures that allow for direct implementations of such multiplications without assuming AND/XOR gates as building blocks. Meanwhile, although the main objective of this thesis is to reduce the area of the bit-parallel multiplier, the latency performance still needs to be considered and explored. In other words, a reasonable evaluation between SET and CMOS gate delays is necessary to ensure that the area improvement achieved by the proposed method does not make the latency run out of control.

5 Proposed SET-MOS Combinational Gates and Area-Efficient Multiplier

5.1 Introduction

In this chapter, the logic gates that can achieve $(a \wedge b) \oplus c$ and $(a \wedge b) \oplus c \oplus d$ logic operations will be proposed thanks to the oscillation characteristics of SET technology. We refer them to as combinational gates that can process a 2-input AND operation followed by one or two 2-input XOR operations, achieving a specific type of logic operation. This design approach can leverage the flexibility of SET, which is distinct from widely used CMOS design. Furthermore, this chapter evaluates all the SET-MOS and CMOS gate delays in this thesis. Among 2-way KA and its extensions, where the conjunctions between the AND gate and XOR gates, there are some application scenarios of the SET-MOS combinational gate. Applying the SET-MOS combinational gates to last chapter's 2-way KA and its extensions, their area complexity can be further reduced. In addition, this chapter will improve the 4-way KA and its extensions based on SET-MOS XOR and combinational gates.

5.2 SET-MOS Combinational Gates

In this section, we will propose novel combinational gates using SET-MOS such that a single gate can implement multiple different two-input logical operations. (*i.e.*, $(a \wedge b) \oplus c$ and $(a \wedge b) \oplus c \oplus d$)

5.2.1 General Idea

The initial idea of combinational gates was to "directly use one SET-MOS gate to implement a specific output bit of the multiplier." For example, the output bit 'c_1', where $c_1 = a_0b_1 + a_1b_0$, in 2-bit multiplication. By adjusting the ratio between the

SET-MOS gate capacitance of inputs, it is possible to apply different weights to the input signals. Once the input is weighted and summed, it can be matched to the output to achieve the desired output bit. Unfortunately, after many attempts, this idea still needs more effort for $c_1 = a_0b_1 + a_1b_0$. No matter how to adjust the weighting between the inputs, there will always be several sets of inputs and outputs that do not correspond.

Nevertheless, in the effort of experimenting, we came up with a solution that takes a step back. Two SET-MOS gates are used to implement the logic $a_0b_1 + a_1b_0$, the first 4-input SET-MOS to correct those input combinations that are not corresponding, and the second 5-input SET for the output. Consequently, we successfully implemented the $a_0b_1 + a_1b_0$ logic using two SET-MOS. This reduced one gate (One-third reduction in total gate requirement) compared to the CMOS implementation.

After more discussions, one optimized solution was determined. We replaced two multi-input SET-MOS gates with one 2-input AND gate and one 3-input SET-MOS gate. The optimized solution mainly achieves an $(a \wedge b) \oplus c$ logic operation. Moreover, this change would decrease latency from two SET-MOS gates' delay to one SET-MOS and one CMOS gates' delay. Less input number of SET-MOS also provides a more stable functionality.

5.2.2 Combined AND/XOR Operation Using SET-MOS

In the previous chapter, a generic schematic diagram of a parallel SET-MOS gate with multiple inputs is shown in Figure 4.2. This gate can perform a 4-input XOR operation by adjusting its parameters. This gate can also be configured to operate as a combinational gate (*i.e.*, $(a \wedge b) \oplus c$ and $(a \wedge b) \oplus c \oplus d$) by changing the oscillation characteristics and parameters.

Table 5.1: Truth table of $(a \wedge b) \oplus c$ operation

Input "a"	Input "b"	Input "c"	Sum of inputs	Output
0	0	0	0	0
0	0	1	**1**	**1**
0	1	0	**1**	**0**
1	0	0	1	0
0	1	1	2	1
1	0	1	2	1
1	1	0	2	1
1	1	1	3	0

When designing the XOR operation using SET-MOS, we made adjustments to the periodicity and phase of the oscillation such that an odd sum of inputs produces an output of logic "1", and an even sum of inputs produces an output of logic "0". However, it is not possible to design the operation shown in Table 5.1 in the same manner as the XOR because the bolded cells (Last two cells of the 3^{rd} and 4^{th} rows) in the table indicate that when the sum of inputs is "1", the output can have two different possibilities. By researching, we have found a solution to this problem.

From our research, changing the gate capacitance of different inputs is the key to designing combinational gates. The on-off state of SET is controlled by the potential of its island [66], denoted as V_{island}. Assuming the background charge is negligible, the V_{island} of SET with three inputs (a, b and c) can be expressed in (5.1).

$$V_{island} = \frac{C_{TD}}{C_\Sigma} V_{DS} + \frac{C_{G1}}{C_\Sigma} V_{in-a} + \frac{C_{G2}}{C_\Sigma} V_{in-b} + \frac{C_{G3}}{C_\Sigma} V_{in-c} + \frac{C_{G0}}{C_\Sigma} V_{in0}. \tag{5.1}$$

Each input in (5.1), including V_{in-a}, V_{in-b}, and V_{in-c}, has the same logic-high and logic-low values. One possible solution for achieving such a combinational gate that follows Table 5.1 is to change the ratio of the different input's gate capacitance. (By changing the capacitance ratio, the effect of each input on Visland is scaled up or down depending on the size of the gate capacitance.) Moreover, by making a lot of attempts and simplifications, it is finally found that when the input capacitance of a

and b is half that of c, each input corresponds to a single output. The weighted truth table with an added column "Sum of weighted inputs" can be shown in Table 5.2.

Table 5.2: Weighted Truth table of $(a \wedge b) \oplus c$ operation

(1×)Input "a"	(1×)Input "b"	(2×)Input "c"	Sum of inputs	Output
0	0	0	0	0
0	0	2	2	1
0	1	0	1	0
1	0	0	1	0
0	1	2	3	1
1	0	2	3	1
1	1	0	2	1
1	1	2	4	0

To implement $(a \wedge b) \oplus c$ operations, Table 5.2's two right columns imply that we can develop a SET oscillation waveform that yields "0" output when V_{island} equals 0- or 1-times the high-level input voltage; and "1" output when V_{island} equals 2- or 3-times the high-level input voltage. Additionally, we know that the periodicity of SET oscillation can equal 4 times the high-level input voltage. The expected voltage transfer characteristic (VTC) of SET can be sketched in Figure 5.1. Noting that to get a better duty cycle, it is desirable to keep the rise of the oscillation as close to 1.5 units as possible and the fall as close to 3.5 units, as shown in Figure 5.1.

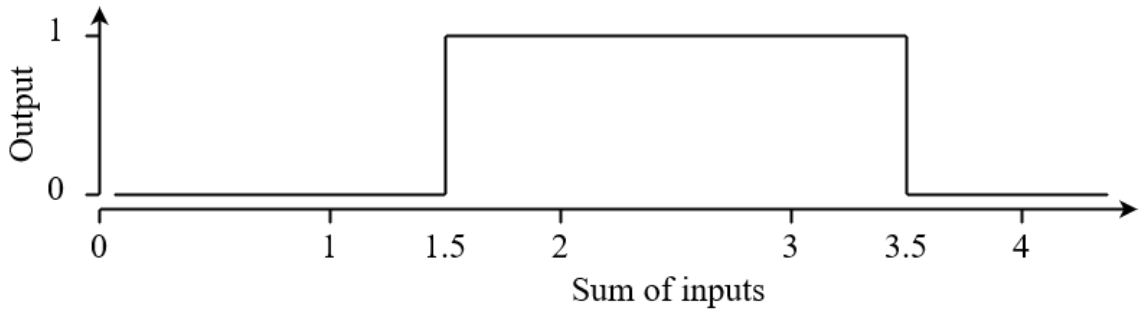


Fig. 5.1: Expected VTC of SET-MOS 3-input combinational gate.

Consistent with the basic assumption used to design SET-MOS XOR gates, we assume that the minimal gate capacitance at the input is $0.1aF$. Therefore, C_{G1} and

C_{G2}, which are the gate capacitance of input a and b, respectively, can be determined as $0.1aF$; the gate capacitance of input c, C_{G3}, can be attributed to $0.2aF$. Then VTC's oscillation periodicity, P, can be calculated as $1.6V$ from $P = \frac{e}{C_G}$, where e is the charge of a single electron. The high-level input voltage can be equal to a quarter of oscillation periodicity, which is $0.4V$. The high-level output voltage is equal to V_{DD}, which is $0.8V$. Table 5.3 shows the parameters of a 3-input SET-MOS that implements an $(a \wedge b) \oplus c$ operation, denoted as SET-MOS AndXor$_3$.

Table 5.3: Parameters for a 4-input SET-MOS AndXor$_3$ implemented in Figure 4.2

Device \ Variable	Parameter	Value
SET	C_{TD}, C_{TS}, C_{Gi}	$0.1aF$
	C_{G0}-C_{G2}	$0.1aF$
	C_{G3}	$0.2aF$
	R_{TD}, R_{TS}	$1M\Omega$
Bias current	I_{DC}	$1uA$
	I_{bias}	$65nA$
Bias voltage	V_{in0}	$-320mV$
	V_{bias}	$-250mV$
	V_{DD}	$0.8V$
p-MOS	V_{th}	$-0.56V$
	W	$360nm$
	L	$45nm$
n-MOS	V_{th}	$0.59V$
	W	$120nm$
	L	$45nm$
Temperature	T	$300K$

The simulation of a 3-input combinational gate by Cadence with GPDK45 is shown in Figure 5.2.

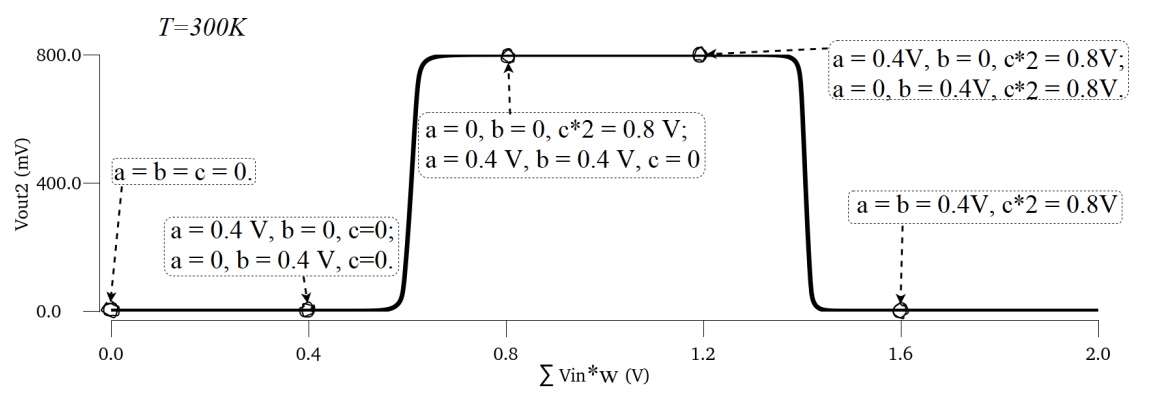


Fig. 5.2: Simulated VTC of SET-MOS 3-input combinational gate for $(a \wedge b) \oplus c$.

The horizontal axis is the summation of each input multiplied by its weight. The vertical axis is the final output of the gate. The dashed box in the figure contains the eight input combination cases from Table 5.2, and it can be seen that each set of inputs corresponds to the correct output.

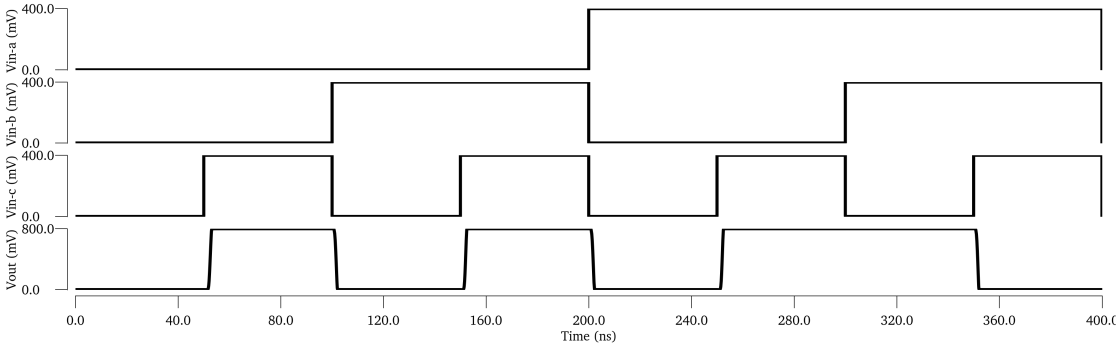


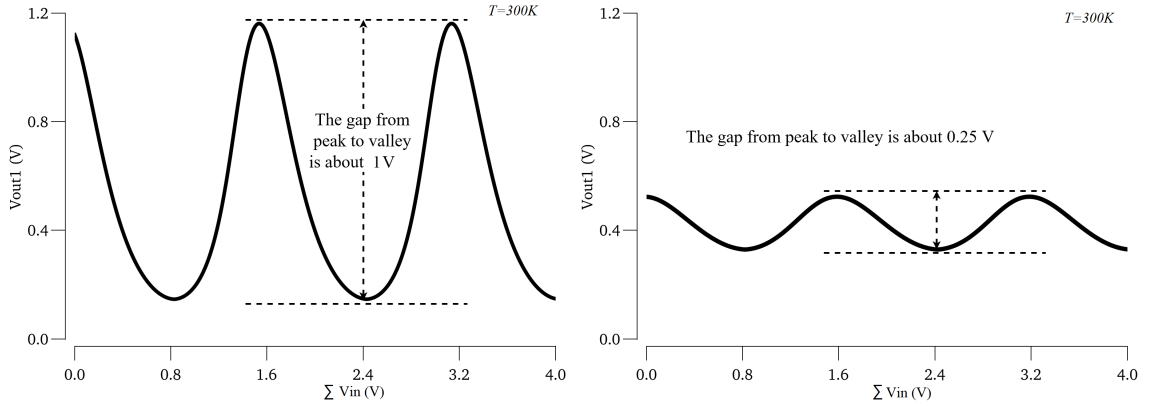
Fig. 5.3: Simulation of SET-MOS AndXor$_3$ with time as the horizontal axis. From top to bottom are the three inputs a, b, c, and the output V_{out}.

Figure 5.3 also shows the simulation result of the SET-MOS AndXor$_3$ with time as the horizontal axis. From top to bottom are input a, input b and input c and the final output (V_{out2} in Figure 4.2). The periodicity of the input signals from a to b and then to c are reduced to $\frac{1}{2}$ of the previous one to simulate all input cases. (The rise/fall time of the input signals is $1ps$.) All input combinations corresponding to Table 5.1 can be found in Figure 5.3 from various time slots, which confirms that the proposed SET-MOS AndXor$_3$ works appropriately.

Furthermore, one can implement an AndXor_4 using the SET-MOS by adding one more input (*i.e.*, input d) with a $0.2aF$ gate capacitance and modifying the value of V_{bias} and V_{in0} to $-279.5mV$ and $-150mV$ from Table 5.3 The simulation of SET-MOS AndXor_4's VTC is similar to that of SET AndXor_3 and can be explained as follows. Since the gate capacitance of V_{in-d} is $0.2aF$ (Twice V_{in-a} or V_{in-b}), $V_{in-d} * w$ is equal to $0.8V$, which is half of the oscillation period in Figure 5.2. The result of $(a \wedge b) \oplus c$ remains unchanged for $d = 0$; $(a \wedge b) \oplus c \oplus d = 1$ for $d = 1$ and $(a \wedge b) \oplus c = 0$; and $(a \wedge b) \oplus c \oplus d = 0$ for d=1 and$(a \wedge b) \oplus c = 1$. It means that increasing the oscillation periodicity by one-half corresponds exactly to the result of $(a \wedge b) \oplus c$ being reversed, resulting in correct $(a \wedge b) \oplus c \oplus d$. Therefore, SET-MOS AndXor_4's correctness can be easily explained by the simulation results of SET-MOS AndXor_3. Moreover, the 8-input XOR gate can also be implemented if 4 more inputs are added with $0.1aF$ gate capacitance and changing I_{bias} and V_{bias} in Table 5.3 to $45nA$ and $-360mV$, respectively.

5.2.3 Discussion on Input Number Limits

Logic operation implemented by SET-MOS gate can save more area than CMOS counterparts; however, there would be a limitation on the input number.



(a) SET-MOS XOR with 4 inputs

(a) SET-MOS XOR with 8 inputs

Fig. 5.4: Oscillation waveform for SET-MOS XOR with 4 or 8 inputs.

Figure 5.4 shows that when the number of inputs is increased from 4 to 8, the gap from the peak to the valley of the oscillation decreases significantly. When the gap is reduced, it lowers the stability of the SET-MOS gates. As per [61], it is crucial to note that adding more inputs to SET will result in a rise in total capacitance, which can have an adverse effect on its operation. Therefore, the maximal input number of SET-MOS gates in this thesis is that "For XOR gates, the number of inputs for XOR does not exceed 8; For combinational gates, the number of inputs does not exceed 4." The reason for this difference is that the input gate capacitance of combinational gates is generally twice the input gate capacitance of XOR gates.

All relevant logic gates in this thesis have been introduced. Next, we will discuss the evaluation of the multiplier latency and average gate delay.

5.3 Latency Performance Evaluation

In existing work, latency complexity is a formula that involves the critical path of a multiplier with respect to the multiplier size n. The speed of multipliers is roughly estimated by comparing the coefficients before $\log_2 n$ in latency complexity [72, 73, 49, 41]. Specifically, the latency performance can be represented as the product of the number of gates on the critical path and their corresponding gate delays [74, 77, 78, 79], with n being a given value.

In this thesis, latency performance of multipliers will be evaluated roughly in latency complexity by comparing the coefficients before $\log_2 n$ and also bringing the specific n and gate delays into latency complexity to find the exact latency of multipliers in ns.

5.3.1 Latency Complexity

In the previous chapter, latency complexity of different multiplication architectures was derived and summarized in Table 4.6. It can be seen that the SET-MOS XOR_4's

improved method does not differ much from CMOS in terms of latency complexity. The quadratic SET-MOS XOR_4 SBM is even better than its CMOS counterpart. However, D_{X4} is generally considered to be larger than $D_{\oplus}$ due to SET devices' lower current drive capability. The following section will introduce an evaluation scheme for the gate delays of SET-MOS and CMOS gates and present the simulation results.

5.3.2 Average Gate Delay

To evaluate gate delays, we begin by simulating the propagation delays generated by various input combinations and then calculate the average of these propagation delays, which is used as the gate delay for a certain logic gate.

The simulation results indicate that two key factors affecting the average are the rise/fall time of inputs and the circuit depth. As the circuit depth increases, it becomes harder to maintain a short rise/fall time for each gate. A longer rise/fall time can negatively impact gate delays, especially for gates that use CMOS technology. This sub-section starts by simulating the delays of individual gates under two different rise/fall times of inputs without loads and then observes the differences. Next, the total delay of multiple gates connected in series will be simulated, and the average delays are summarized based on circuit depth. Finally, a curve fit will be used to approximate the gate delays.

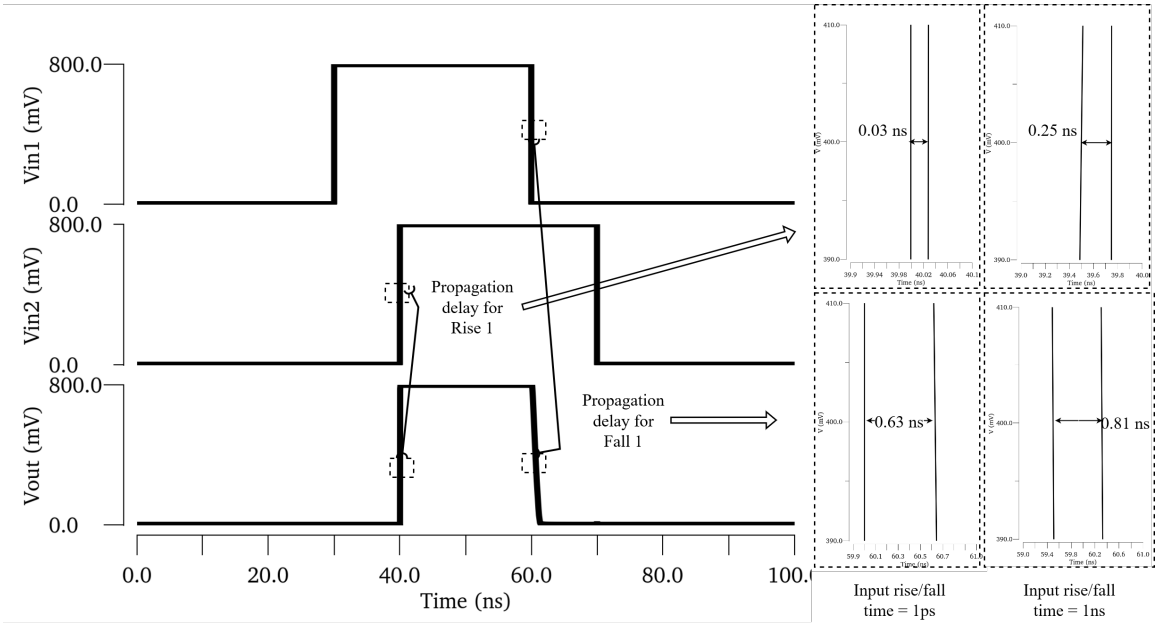


Fig. 5.5: Simulation for CMOS 2-input AND gate using PTL.

Figure 5.5 displays the delay simulation results for the AND_2 gate. The left-hand side of the figure shows three lines representing V_{in1}, V_{in2}, and V_{out}. The propagation delay occurs when V_{in1} transitions from low to high and V_{out} transitions from low to high in response. The propagation delays are displayed in an enlarged format on the right-hand side of the figure. In the lower right corner of the figure, it can be seen that enlarged views of the two output-falling-edge propagation delays for input rise/fall time are equivalent to $1ps$ or $1ns$. Similarly, there is another propagation delay when V_{in2} transits from high to low and V_out transits from high to low, shown in the top right corner. The right-hand side of Figure 5.5 shows that when the input's rise/fall time is $1ps$, the rising edge produces a propagation delay of $0.03ns$, while the falling edge produces a propagation delay of $0.63ns$. The average gate delay is $0.33ns$. In contrast, when the input's rise/fall time comes to $1ps$, the average gate delay increases to $0.53ns$, which is the average of $0.25ns$ and $0.81ns$.

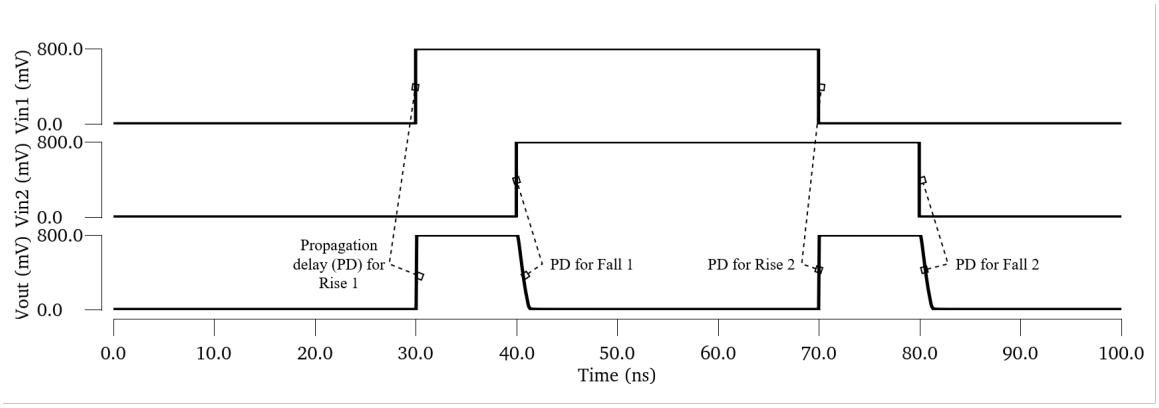


Fig. 5.6: Simulation for CMOS 2-input XOR gate using PTL.

The simulation for CMOS XOR_2 designed by PTL is shown in Figure 5.6. 4 propagation delays (PD) occur on the two rising and two falling edges of the output (indicated by Rise 1, 2 and Fall 1, 2). These PDs and the averaged Gate delays are shown in Table 5.4. Table 5.4 contains two sub-tables of gate delays, distinguished by the inputs' different rise/fall times.

Table 5.4: Gate delays of CMOS XOR_2 with different input's rise/fall times

Input rise/fall time = $1ns$			Input rise/fall time = $1ps$		
PD (ns)		Gate delay (ns)	PD (ns)		Gate delay (ns)
Rise 1	Rise 2		Rise 1	Rise 2	
0.37	0.31		0.07	0.04	
Fall 1	Fall 2	0.64	Fall 1	Fall 2	0.35
1.02	0.86		0.64	0.64	

The simulation for SET-MOS XOR_4 can be shown in Figure 5.7. When input's rise and fall time is $1ns$ or $1ps$, 10 PDs shown in Figure 5.7 and gate delays are summarized in Table 5.5.

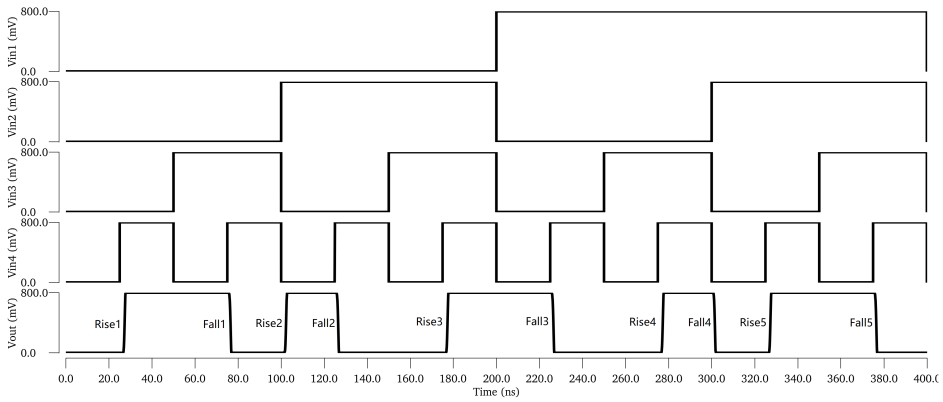


Fig. 5.7: Simulation for SET-MOS XOR_4.

Table 5.5: Gate delays of SET-MOS XOR_4 with different input's rise/fall times

Input rise/fall time = $1ns$					
PD (ns)					Gate delay (ns)
Rise 1	Rise 2	Rise 3	Rise 4	Rise 5	
2.25	2.37	2.25	2.25	2.25	
Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	1.9
1.52	1.52	1.52	1.49	1.52	
Input rise/fall time = $1ps$					
PD (ns)					Gate delay (ns)
Rise 1	Rise 2	Rise 3	Rise 4	Rise 5	
2.29	2.29	2.30	2.30	2.29	
Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	1.89
1.48	1.48	1.48	1.48	1.48	

The simulation of SET-MOS XOR_8 can be shown in Figure 5.8.

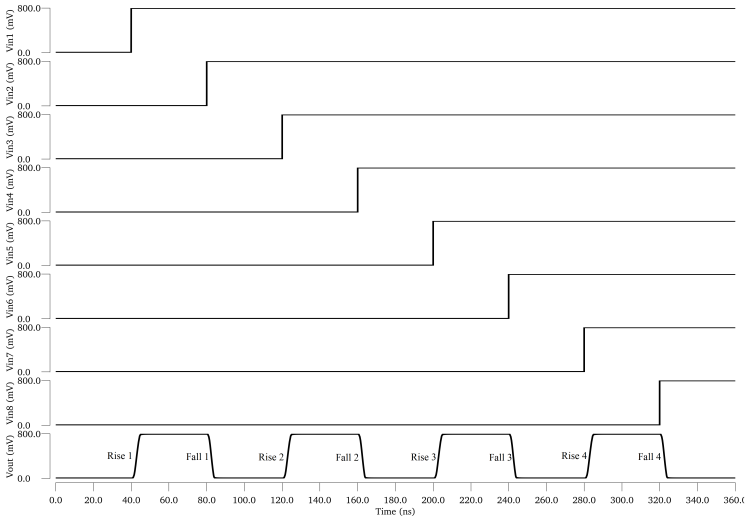


Fig. 5.8: Simulation for SET-MOS XOR_8.

The 8 PDs shown in Figure 5.8 and the gate delays of SET-MOS XOR_8 for different input rise/fall times are summarized in Table 5.6.

Table 5.6: Gate delays of SET-MOS XOR_8 with different input's rise/fall times

Input rise/fall time = $1ns$					Input rise/fall time = $1ps$				
PD (ns)				Gate	PD (ns)				Gate
Rise 1	Rise 2	Rise 3	Rise 4	delay	Rise 1	Rise 2	Rise 3	Rise 4	delay
2.92	2.93	2.94	2.94	(ns)	2.92	2.93	2.93	2.93	(ns)
Fall 1	Fall 2	Fall 3	Fall 4	2.7	Fall 1	Fall 2	Fall 3	Fall 4	2.68
2.47	2.47	2.47	2.49		2.44	2.44	2.44	2.44	

The simulation for SET $AndXor_3$ is shown in Figure 5.3, where PDs occur on 3 rising and falling edges of the V_{out}. The gate delays for $1ns$ and $1ps$ can be summarized in Table 5.7.

Table 5.7: Gate delays of SET-MOS AndXor$_3$ with different input's rise/fall times

Input rise/fall time = $1ns$				Input rise/fall time = $1ps$			
PD (ns)			Gate delay (ns)	PD (ns)			Gate delay (ns)
Rise 1	Rise 2	Rise 3		Rise 1	Rise 2	Rise 3	
2.72	1.67	1.67		2.49	2.00	2.01	
Fall 1	Fall 2	Fall 3	1.9	Fall 1	Fall 2	Fall 3	1.88
1.57	1.94	1.94		1.55	1.70	1.57	

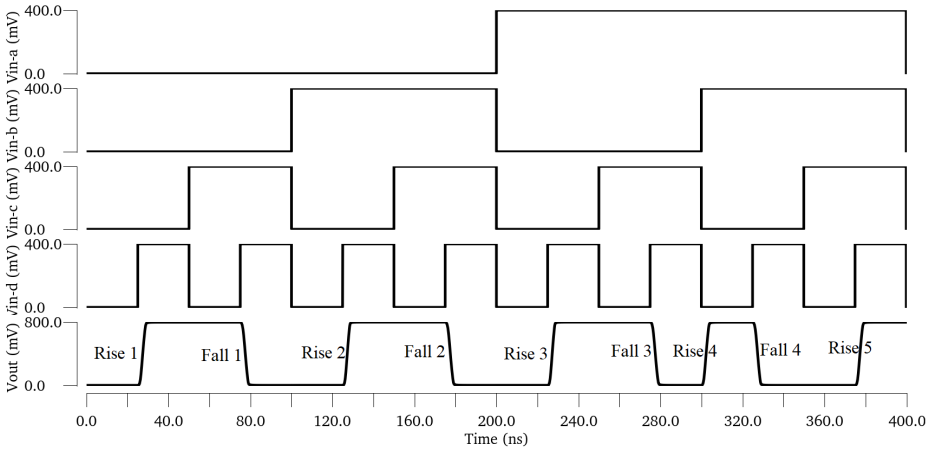


Fig. 5.9: Simulation for SET-MOS AndXor$_4$.

Figure 5.9 shows 9 PDs on 5 rising edges and 4 falling edges of V_{out}. The PDs and gate delays can be summarized in Table 5.9 for inputs' rise/fall times to be $1ns$ and $1ps$, respectively.

Table 5.8: Gate delays of SET-MOS AndXor_4 with different input's rise/fall times

Input rise/fall time = $1ns$					
PD (ns)					Gate delay (ns)
Rise 1	Rise 2	Rise 3	Rise 4	Rise 5	
2.7	1.91	1.91	2.33	2.69	
Fall 1	Fall 2	Fall 3	Fall 4	-	2.4
2.79	2.24	2.24	2.79		
Input rise/fall time = $1ps$					
PD (ns)					Gate delay (ns)
Rise 1	Rise 2	Rise 3	Rise 4	Rise 5	
2.41	2.22	2.22	2.34	2.40	
Fall 1	Fall 2	Fall 3	Fall 4	-	2.39
2.45	2.53	2.53	2.45		

An observation can be made comparing the gate delays of $1ns$ and $1ps$ using SET-MOS and CMOS technology presented in this sub-section. The inputs' rise and fall time barely affect the delays of SET-MOS gates, while the delays of CMOS gates are significantly affected. In the case of multiple gates connected, such as a critical path in a multiplier, where the input of the previous gate is the output of the next gate, it is difficult to maintain a very short rise/fall time of inputs for all the gates on the critical path.

Assuming a multiplier's circuit depth is d. Since the latency of the multipliers in this work generally ranges from $(2\log_2 n - 3)$ to $(3\log_2 n + 1)$, $d \in [9, 31]$ for n in the range of $[256, 2048]$ (n is a power of 2). Simulations were performed to measure the delay of XOR_2 and AND_2 with respect to circuit depth d. The average gate delays of CMOS XOR_2 and AND_2 are dotted in Figure 5.10, and the curved fitting lines, $D_{\otimes}(d)$ and $D_{\oplus}(d)$ as functions of d, can be expressed in the following equations. Noting that

the first gates are with very short input rise/fall time for $1ps$.

$$\begin{aligned} D_{\otimes}(d) &= -0.28/d + 0.61 = 0.33 + \left(\frac{d-1}{d}\right) \times 0.28\ ns. \\ D_{\oplus}(d) &= -0.31/d + 0.66 = 0.35 + \left(\frac{d-1}{d}\right) \times 0.31\ ns, \end{aligned} \tag{5.2}$$

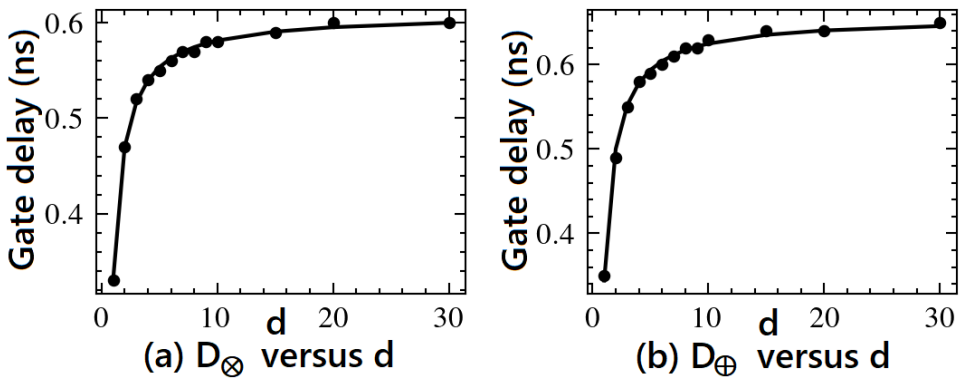


Fig. 5.10: Curve fitting to $D_{\otimes}$ and $D_{\oplus}$ with respect to the circuit depth d.

(5.2) shows that $D_{\otimes}(1) = 0.33ns$ and $D_{\oplus}(1) = 0.35ns$, which are the same values as the previous simulated gate delays shown in Figure 5.5 and Table 5.4. It is important to note that $D_{\otimes}(d) \to 0.61ns$ and $D_{\oplus}(d) \to 0.66ns$ when $d \gg 1$. For the range of circuit depth d of the practical interest, we approximate $D_{\otimes}$ and $D_{\oplus}$ with $0.6ns$ and $0.65ns$, respectively, as the average gate delay for XOR_2 and AND_2 in multiplier implementation. Simulations also showed that the gates using SET-MOS technology, including XOR_4, $AndXor_3$ and $AndXor_4$, are almost unaffected by the different input's rise/fall time and the loads caused by series connection. The average gate delays in this chapter are summarized in Table 5.9. Subsequently, we will re-examine the existing multiplier architectures to find possible application scenarios for all types of SET-MOS gates discussed in this section.

Table 5.9: Average gate delay in multiplier of practical sizes

Technology	CMOS		SET-MOS			
Gate name	AND_2	XOR_2	XOR_4	XOR_8	$AndXor_3$	$AndXor_4$
Notation for gate delay	$D_{\otimes}$	$D_{\oplus}$	D_{X4}	D_{X8}	D_{AX3}	D_{AX4}
Gate delay(ns)	0.6	0.65	1.89	2.68	1.88	2.39

5.4 Multiplications Using SET-MOS XOR and Combinational Gates

Like SET-MOS XOR_4 can be applied to some structures/building blocks that contain multiple XOR operations in the multiplier, SET-MOS combinational gates can be applied to structures combining AND and XOR operations, reducing the gate/transistor count required by multiplications.

In the previous chapter, we introduced the SET-MOS XOR_4 improvements to the existing SBM, 2-way KA and its extensions. This section will provide SET-MOS combinational gate improvements to these multiplier architectures. In addition, this section will apply the SET-MOS XOR_4 and combinational gates to the existing 4-way KA-like architectures to improve their area performance.

5.4.1 SBM Using SET-MOS Combinational Gates

The key to implementing SBM is to realize each of the c_ks in (3.1). SET-MOS $AndXor_3$s are ideally suited to implement this "AND followed by XOR operation" required by c_ks. For each c_k in (3.1), it is only necessary to implement the first $a_i b_j$ with one CMOS AND_2 and then use SET-MOS $AndXor_3$s to implement the subsequent "$+a_{i'}b_{j'}\ldots$". Therefore, the gate count required for each c_k is k, *i.e.*, one CMOS AND_2 and $k-1$ SET-MOS $AndXor_3$. The area complexity of an n-bit SBM

improved by SET-MOS AndXor_3 can be summarized as,

$$G(n) = \sum_{i=1}^{n} i + \sum_{j=1}^{n-1} j = n^2. \tag{5.3}$$

In this chapter, we no longer distinguish between $G_{\oplus}(n)$ and $G_{\otimes}(n)$ but use $G(n)$ to denote the multiplier's area complexity, where $G(n) = G_{\oplus}(n)+G_{\otimes}(n)$. This is because XOR gates have the same transistor count as AND gates in this work. Moreover, we consider each gate shown in Table 5.9 has the same impact on the multiplier's area complexity.

The latency of this implementation can be calculated intuitively, *i.e.*, the critical path delay of c_{n-1}, which contains $n-1$ SET-MOS AndXor_3 and one CMOS AND_2, shown below.

$$L(n) = (n-1)D_{AX3} + D_{\oplus}. \tag{5.4}$$

It is worth mentioning that the application SET-MOS AndXor_4 can optimize the latency of (5.4) at no additional delay cost. For example, various gate-level implementations of a 4-bit SBM are shown in Figure 5.11, where Figure 5.11(a) is a pure CMOS design, Figure 5.11(b) uses SET-MOS AndXor_3 gates, and Figure 5.11(c) utilizes both SET-MOS AndXor_3 and AndXor_4. It can be seen that a multiplier's gate count is reduced from 25 in Figure 5.11(a) to 16 in Figure 5.11(b)(c), which is equivalent to 36% gate count savings. Furthermore, the proposed implementation in Figure 5.11(c) keeps the same circuit depth compared to its CMOS counterpart in Figure 5.11(a). The optimized latency of SBM improved by SET-MOS combinational gates is,

$$L(n) \leq \lceil \frac{n-1}{2} \rceil D_{AX4} + D_{\otimes}, \tag{5.5}$$

where D_{AX3} and D_{AX4} are the gate delays of the SET-MOS AndXor_3 and AndXor_4, respectively. (5.5) represents the implementation with reduced latency using AndXor_4

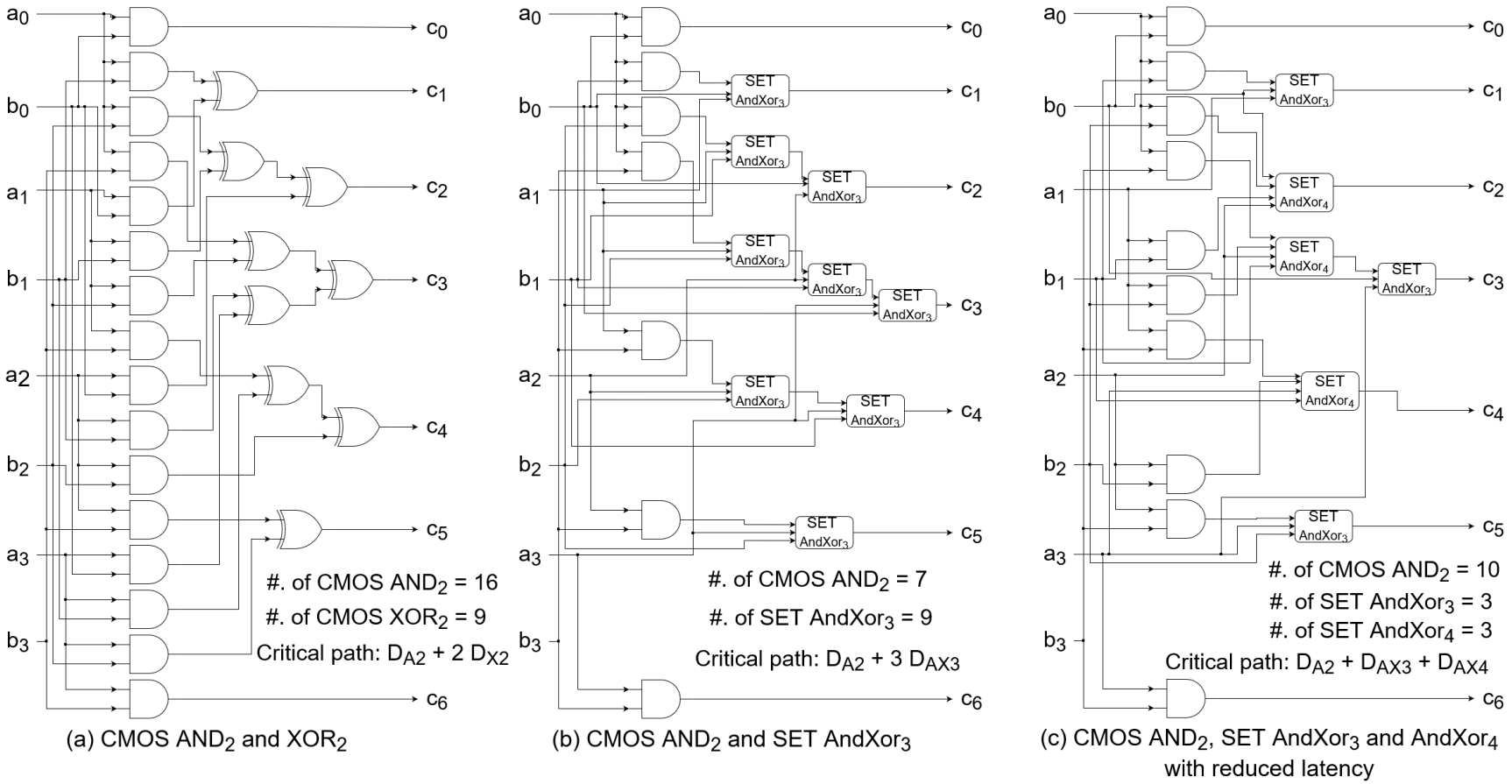


Fig. 5.11: Implementation of a 4-bit base multiplier via different technologies.

gates, as illustrated in Figure 5.11(c), where $L(n) = \frac{n-1}{2}D_{AX4} + D_{\otimes}$ if n is odd, and $L(n) = (\frac{n}{2} - 1)D_{AX4} + D_{AX3} + D_{\otimes}$ if n is even.

In the previous chapter, we introduced an SBM that used SET-MOS XOR and reduced the area complexity to $1.33n^2 + 0.33n$, compared to the conventional CMOS implementation that required $2n^2 - 2n + 1$. In this chapter, we have further improved the SBM using combinational gates, and the improved SBM only requires n^2 gates. This implementation is also comparable in circuit depth to the other two implementations, especially for a small multiplier size of $n \leq 4$.

5.4.2 2-Way KA and Its Extensions Using SET-MOS Gates

In this sub-section, we will continue to enhance the 2-way KA and its extensions which we improved in the previous chapter. The SET-MOS XOR_4, $AndXor_3$ and $AndXor_4$ will be deployed in suitable scenarios of these multiplication architectures.

5.4.2.1 Original 2-Way KA Using SET-MOS Gates

By applying the SET-MOS combinational gates, the area complexity of the 2-way KA using SET-MOS XOR_4 proposed in the previous chapter can be further reduced.

Consider two polynomials, A and B, over $GF(2^n)$, both with a degree of $n - 1$. Each of these polynomials can be split into two equal-length polynomials with a degree of $\frac{n}{2} - 1$. The original 2-way KA [19] multiplication architecture decomposed into three blocks of CPF, CM, and RC according to [49, 24], as shown in Figure 5.12.

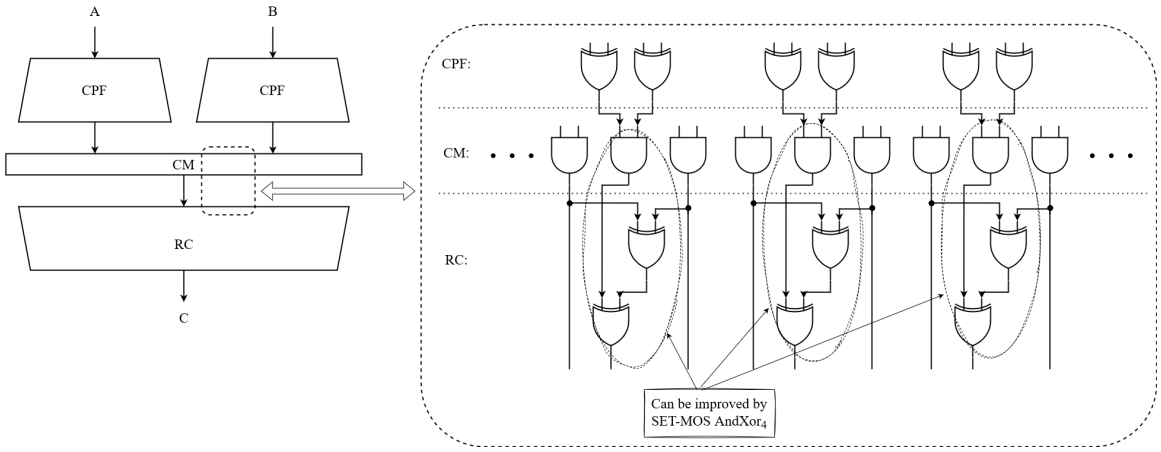


Fig. 5.12: Decomposition of 2-way KA and application scenarios of AndXor_4.

As can be seen from the enlarged illustration on the right-hand side of Figure 5.12, part of the AND_2 in CM block and the two XOR_2 in the first layer in RC block can be replaced by one AndXor_4. Each application of AndXor_4 can reduce two gates. In the multiplication implementation, the CPF blocks are created using CMOS XOR_2, while the CM and RC blocks are made with AND_2, AndXor_4 and XOR_4. It is essential to mention that the RC block primarily uses SET-MOS AndXor_4 for the first layer and SET-MOS XOR_4 for subsequent layers. The reconstruction process is the same as in Table 4.4 in the previous chapter. Therefore, the area complexity of the improved $CM + RC$ can be shown as follows:

- Recursive formula

$$\begin{cases} G^{CM+RC}(2) = 3, \\ G^{CM+RC}(n) = 3G^{CM+RC}(\frac{n}{2}) + n - 1, \end{cases} \tag{5.6}$$

- Non-recursive formula

$$G^{CM+RC}(n) = \frac{13}{6}n^{1.58} - 2n + \frac{1}{2}, \tag{5.7}$$

Note that $CM + RC$ is a combined block of CM and RC, which can be improved by SET-MOS AndXor$_4$, and G^{CM+RC} is the area complexity for that $CM + RC$ block. CPF blocks are the same as the existing method, which has an area complexity of $G^{CPF} = n^{1.58} - n$. Regarding latency, CMOS XOR$_2$s are used in CPF blocks. Thus, the gate delays of one CMOS XOR$_2$ and one SET-MOS XOR$_4$ are increased per round of KA. $L(2)$ can be observed from Figure 5.12 as one CMOS XOR$_2$ and one SET-MOS AndXor$_4$ (because Table 5.9 demonstrates that the delay of AndXor$_4$ is more significant than that of AND$_2$). As a result, the area and latency complexity of KA improved by AndXor$_4$ and XOR$_4$ can be summarized as,

$$\begin{cases} G(n) = 2G^{CPF}(n) + G^{CM+RC}(n) = 4.16n^{1.58} - 4n + 0.5, \\ L(n) = \log_2 n(D_{X4} + D_{\oplus}) - D_{X4} + D_{AX4}, \end{cases} \tag{5.8}$$

A brief comparison can be made between (5.8) and (4.5). When $n = 256$, the reduction of total gate counts ($G_{\otimes} + G_{\oplus}$) can be estimated by $(28501 - 26314)/28501 = 7.7\%$.

5.4.2.2 2-Way Reconstructed KA Using SET-MOS Gates

Let us decompose the 2-way reconstructed KA by three blocks. As with the original 2-way KA, the block between CM and RC can be improved with AndXor$_4$ to increase the area complexity further. From (5.8) and (4.5), we can see that the reduced AND gate count is $\frac{1}{3}n^{1.58}$. For latency, CPF blocks and $R = P_0 + P_1 x^n$ step of RC block can be implemented with CMOS XOR$_2$. In each round of reconstructed KA, the added gate delays are two CMOS XOR$_2$ and one SET-MOS XOR$_4$. $L(2)$ consists of one CMOS XOR$_2$ and one SET-MOS AndXor$_4$. In short, the area and latency complexities of 2-way reconstructed KA improved by AndXor$_4$ and XOR$_4$ can be

summarized as,

$$\begin{cases} G(n) = 4.77n^{1.58} - 5n + 1, \\ L(n) = 2\log_2 nD_{\oplus} + \log_2 nD_{X4} - D_{\oplus} - D_{X4} + D_{AX4}. \end{cases} \tag{5.9}$$

Taking a quick look at (5.9) and (4.7) and assuming $n = 256$, we can estimate a reduction in total gate counts of $(31782 - 29595)/31782$, amounting to 6.9%.

5.4.2.3 2-Way Overlap-Free KA Using SET-MOS Gates

Like the previous two KA-like architectures, applying SET-MOS AndXor$_4$ can contribute to a slight area improvement for 2-way overlap-free KA. A reduction of $\frac{1}{3}n^{1.58}$ can be achieved by constructing CM and RC blocks with the help of SET-MOS AndXor$_4$. Regarding latency, CPF blocks and the step $P0 + P2$ in the RC block can be implemented with CMOS XOR$_2$. CMOS XOR$_2$, used to process $P0 + P1$, and SET-MOS XOR$_4$, used to process $P0 + P1 + P2$, operate simultaneously. Since the gate delay of SET-MOS XOR$_4$is longer than that of CMOS XOR$_2$, the delays added to each round of overlap-free KA are one CMOS XOR$_2$ and one SET-MOS XOR$_4$. As a result, the area and latency complexities of overlap-free KA improved by SET-MOS AndXor$_4$ and XOR$_4$ are as follows.

$$\begin{cases} G(n) = 5.17n^{1.58} - 6n + 1.5, \\ L(n) = \log_2 n(D_{\oplus} + D_{X4}) - D_{X4} + D_{AX4}. \end{cases} \tag{5.10}$$

When the value of n is 256, comparing equations (4.10) and (5.10) shows that using SET-MOS AndXor$_4$ can further reduce the area complexity of overlap-free KA (Improved by SET-MOS XOR$_4$). The reduction is indicated by the equation $(35063 - 32876)/35063$, which results in a 6.3% area reduction.

5.4.2.4 2-Way Block Recombination Using SET-MOS Gates

Applying SET-MOS combinational and XOR gates can reduce the area complexity of 2-way block recombination method. Let A and B be two polynomials over $GF(2^n)$ with degree of $n-1$. These polynomials can be split into two equal-length polynomials, A_0, A_1 and B_0, B_1, with the degree of $\frac{n}{2}-1$. The data flow of $AB=C$ is displayed in Figure 5.13.

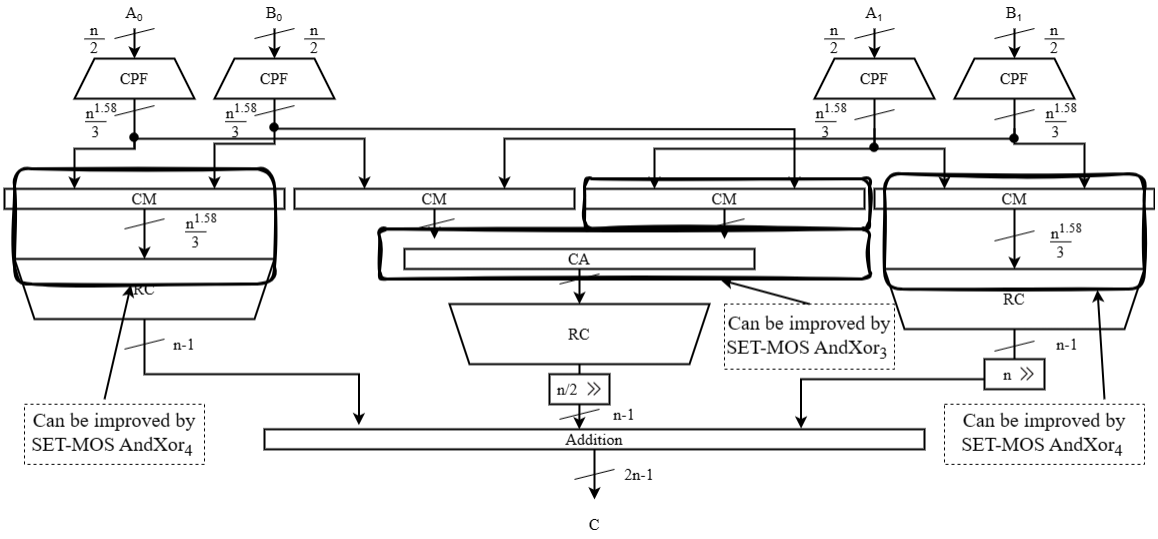


Fig. 5.13: Application scenarios of SET-MOS combinational gates in the 2-way block recombination.

As shown in Figure 5.13, SET-MOS AndXor$_4$ can be applied to two CM and RC combos; SET-MOS AndXor$_3$ can be applied to one CM and CA combo. If the original 2-way KA is used for the RC blocks and SET-MOS XOR$_4$ is applied. (This is different from existing work in [24] and proposed work in the previous chapter. The RC blocks in this chapter apply the original KA using XOR$_4$). We have,

$$\begin{cases} G^{CM}(n) = n^{\log_2 3}, \quad G^{CM+CA}(n) = n^{\log_2 3}. \\ G^{RC}(n) = \frac{3}{2}n^{\log_2 3} - 2n + \frac{1}{2}, \end{cases}$$

The area complexity can be calculated as follows,

$$\begin{cases} G(n) = & 4G^{CPF}(\frac{n}{2}) + G^{RC}(\frac{n}{2}) + G^{CM}(\frac{n}{2}) + 2G^{CM+RC}(\frac{n}{2}) + G^{CM+CA}(\frac{n}{2}) + n - 2, \\ G(n) = & \frac{71}{18}n^{\log_2 3} - 4n - \frac{1}{2}, \end{cases}$$

where $G^{CPF}(\frac{n}{2}) = \frac{4}{3}n^{\log_2 3} - \frac{n}{2}$ and $G^{CM+RC}(\frac{n}{2})$ can be calculated from (5.7). Note that $CM + CA$ is the combined block of CM and CA, which can be improved by AndXor_3. G^{CM+CA} denotes its area complexity. As for the latency, it can be seen from Figure 5.13 that one CMOS XOR on the critical path is replaced with an AndXor_3. The path of the left and right branches shown in Figure 5.13 and the path of the middle line are counted as follows.

$$\begin{cases} L_{l:h}(n) = \log_2 n D_{\oplus} + (\log_2 n - 2)D_{X4} + D_{AX4}, \\ L_m(n) = \log_2 n D_{\oplus} + (\log_2 n - 1)D_{X4} + D_{\otimes} + D_{AX3}, \end{cases}$$

where $L_{l:h}(n)$ shows the path from $A_i/B_i \rightarrow CM + RC \rightarrow \textit{Addition} \rightarrow$ to C, and $L_m(n)$ is from $A_i/B_i \rightarrow CM \rightarrow CM + CA \rightarrow RC \rightarrow \textit{Addition} \rightarrow$ to C. From Table 5.9, it can be seen that $D_{AX3} + D_{X4} + D_{\otimes}$ is longer than D_{AX4}. Thus, the middle line (L_m) is the critical path. The area and latency complexities of 2-way block recombination improved by SET-MOS gates are,

$$\begin{cases} G(n) = 3.94n^{1.58} - 4n - 0.5, \\ L_(n) = \log_2 n(D_{\oplus} + D_{X4}) - D_{X4} + D_{\otimes} + D_{AX3} \end{cases} \tag{5.11}$$

When the value of n is 256, comparing equations (5.11) and (4.12) shows that the new result of 2-way block recombination using all types of SET-MOS gates achieves a $(31398 - 24855)/31398 = 21\%$ area reduction to its predecessor in the previous chapter. It's important to note that this reduction is not solely due to the application

of the SET-MOS combinational gates, but also because in this chapter, *RC* applies the original KA using SET-MOS XOR_4.

5.4.3 4-Way KA and Its Extensions Using SET-MOS Gates

In this sub-section, we will use all types of the SET-MOS gates proposed in this thesis to enhance the 4-way KA, as well as the 4-way block recombination multiplication architecture.

5.4.3.1 4-Way KA Using SET-MOS Gates

4-way KA can be considered as two rounds of 2-way KA using 9 $\frac{n}{4}$-bit sub-multiplications. In contrast to 2-way KA, 4-way KA has more terms in each round of the reconstruction process than the 2-way KA, and this is the application scenario where SET-MOS XOR_8 can exploit its multi-input processing capability.

Considering A and B are two elements in $GF(2^n)$ represented to polynomial basis with degree $n-1$. Each of them can be divided into four equal-length polynomials with the degree $\frac{n}{4} - 1$ shown in (3.15). The multiplication $AB = C$ can be represented in (3.16) and (3.17). The component process is given in Table 3.3, where we can summarize the non-recursive formula for the CPF block in 4-way KA. $G^{CPF}(n)$ equals $n^{\log_2 3} - n$, equivalent to that in 2-way KA.

The reconstruction process of the 4-way KA with SET-MOS XOR_8 can be area-efficient as one gate can process up to 8 bits simultaneously. Dividing each partial product in (3.17), P_i, into two parts,

$$P_i = P_{i,h} x^{\frac{n}{4}} + P_{i,l}, \qquad i = 0, 1, \ldots, 8, \tag{5.12}$$

such that $P_{i,l}$ has $\frac{n}{4}$ bits and $P_{i,h}$ has $\frac{n}{4} - 1$ bits. We propose the reconstruction process implemented with SET-MOS XOR_8 as follows,

Step 1. Compute in parallel $R_i, i = 0, 1, 2, 3, 4$, using SET-MOS XOR_8:

$$\begin{cases} R_0 &= P_{0,h} + P_{0,l} + P_{1,l} + P_{2,l}, \\ R_1 &= P_{0,h} + P_{1,h} + P_{2,h} + P_{0,l} + P_{1,l} + P_{3,l} + P_{6,l}, \\ R_2 &= P_{0,h} + P_{1,h} + P_{3,h} + P_{6,h} + P_{1,l} + P_{2,l} + P_{4,l} + P_{7,l}, \\ R_3 &= P_{1,h} + P_{2,h} + P_{4,h} + P_{7,h} + P_{3,l} + P_{4,l} + P_{5,l}, \\ R_4 &= P_{3,h} + P_{4,h} + P_{5,h} + P_{4,l}. \end{cases}$$

Step 2. Compute in parallel $R'_i, i = 0, 1$, using SET-MOS XOR_8:

$$\begin{cases} R'_0 &= R_2 + P_{0,l} + P_{2,l} + P_{5,l} + P_{6,l} + P_{8,l}, \\ R'_1 &= R_2 + P_{3,h} + P_{4,h} + P_{5,h} + P_{7,h} + P_{8,h}. \end{cases}$$

The multiplication product can be obtained as

$$AB = P_{0,l} + R_0 x^{\frac{n}{4}} + R_1 x^{\frac{n}{2}} + R'_0 x^{\frac{3n}{4}} + R'_1 x^n + R_3 x^{\frac{5n}{4}} + R_4 x^{\frac{3n}{2}} + P_{4,h} x^{\frac{7n}{4}}.$$

There are $\frac{n}{4}$ SET-MOS XOR_8 required to implement each of R_0, R_1, R_2, and R_3, while only $\frac{n}{4} - 1$ XOR_8 gates are needed to realize R_4. As a result, $\frac{5n}{4} - 1$ XOR_8s are used to implement the Step 1. In the Step 2, R'_0 requires $\frac{n}{4}$ XOR_8 and R'_1 utilizes $\frac{n}{4} - 1$ XOR_8s. Obtaining the product AB does not require extra gates because all coefficients are below $\frac{n}{4}$-bit, where no overlap occurs. In total, there are $\frac{7n}{4} - 2$ SET-MOS XOR_8s required to implement the reconstruction process for each round of 4-way KA. It is worth mentioning that the connection between CM and RC can still be improved using SET-MOS AndXor_4 in the first round of KA, *i.e.*, a 4-bit 4-way KA multiplication (2 rounds of (5.6)). Then the area complexity of the $CM + RC$ blocks can be calculated as follows, $n \geq 4$.

- Recursive formula

$$\begin{cases} G^{CM+RC}(4) = 12, \\ G^{CM+RC}(n) = 9G^{CM+RC}(\frac{n}{4}) + \frac{7n}{4} - 2, \end{cases}$$

- Non-recursive formula

$$G^{CM+RC}(n) = \frac{347}{180}n^{1.58} - \frac{7}{5}n + \frac{1}{4},$$

About latency complexity, we see from $(P_0, P_1, \ldots, P_7) \rightarrow$ Step 1 $\rightarrow$ Step 2 involves two SET-MOS XOR_8's delay. The latency from $P_8 \rightarrow$ Step 2 involves one SET-MOS XOR_8's delay. Taking the latency of the component and sub-multiplication process in Table 3.3, the added latency of each round KA are one CMOS XOR_2 plus two SET-MOS XOR_8. Briefly, the area and latency complexities of KA improved by $AndXor_4$ and XOR_8 can be summarized as follows for $n \geq 4$.

$$\begin{cases} G(n) = 2G^{CPF}(n) + G^{CM+RC}(n) = 3.93n^{1.58} - 3.4n + 0.25, \\ L(n) = (\log_2 n - 2)(D_{X8} + 0.5D_{\oplus}) + 2D_{\oplus} + D_{AX4} + D_{X8}. \end{cases} \tag{5.13}$$

Considering that the high delay of SET-MOS XOR_8 may lead to unexpected costs in terms of multiplier's latency, we propose another version of 4-way KA that reconstructed by SET-MOS XOR_4. The only difference between XOR_4 version and the XOR_8 version presented above is the reconstruction process, which will be presented as follows.

Dividing each partial product, P_i, into two parts, which are the same as (5.12). We propose the reconstruction steps implemented with SET-MOS XOR_4 as follows,

Step 1. Compute in parallel $R_i, i = 0, 1, 2, \ldots, 5$, using SET XOR_4:

$$\begin{cases} R_0 &= P_{1,h} + P_{6,h} + P_{6,l} + P_{7,l}, \\ R_1 &= P_{6,h} + P_{7,h} + P_{3,l} + P_{7,l}, \\ R_2 &= P_{0,h} + P_{0,l} + P_{1,l} + P_{2,l}, \\ R_3 &= P_{0,h} + P_{1,h} + P_{2,h} + P_{1,l}, \\ R_4 &= P_{3,h} + P_{3,l} + P_{4,l} + P_{5,l}, \\ R_5 &= P_{3,h} + P_{4,h} + P_{5,h} + P_{4,l}. \end{cases} \tag{5.14}$$

Step 2. Compute in parallel $R'_i, i = 0, 1, 2, 3$, using SET XOR_4:

$$\begin{cases} R'_0 &= R_3 + P_{0,l} + P_{3,l} + P_{6,l}, \\ R'_1 &= R_0 + R_2 + R_4 + P_{8,l}, \\ R'_2 &= R_1 + R_3 + R_5 + P_{8,h}, \\ R'_3 &= R_4 + P_{1,h} + P_{4,h} + P_{7,h}. \end{cases} \tag{5.15}$$

The product is then obtained as

$$AB = P_{0,l} + R_2 x^{\frac{n}{4}} + R'_0 x^{\frac{n}{2}} + R'_1 x^{\frac{3n}{4}} + R'_2 x^n + R'_3 x^{\frac{5n}{4}} + R_5 x^{\frac{3n}{2}} + P_{4,h} x^{\frac{7n}{4}}. \tag{5.16}$$

It can be seen in Step 1 that there are $\frac{n}{4}$ XOR_4 required to implement R_0, R_1, R_2, and R_4 each, while only $\frac{n}{4} - 1$ XOR_4 are needed to realize R_3 and R_5. As a result, $\frac{3n}{2} - 2$ XOR_4 gates are used to implement Step 1. In Step 2, each R'_i requires $\frac{n}{4}$ XOR_4, except R'_3 utilizes only $\frac{n}{4} - 1$ XOR_4 gates. Note that there are no further gates needed in obtaining the product AB, as all the coefficients are of no more than $\frac{n}{4}$-bit.

In summary, there are $\frac{5n}{2} - 3$ XOR_4 gates required for reconstruction of each 4-way KA. The AndXor_4 can be applied to the same location as the XOR_8 version. Then

the area complexity of the $CM + RC$ blocks with SET-MOS AndXor$_4$ and XOR$_4$ can be obtained as follows for $n \geq 4$.

- Recursive formula

$$\begin{cases} G^{CM+RC}(4) = 12, \\ G^{CM+RC}(n) = 9G^{CM+RC}(\frac{n}{4}) + \frac{5n}{2} - 3, \end{cases}$$

- Non-recursive formula

$$G^{CM+RC}(n) = \frac{157}{72} n^{1.58} - 2n + \frac{3}{8}, \tag{5.17}$$

When considering the latency, this XOR$_4$ version is almost identical to the XOR$_8$ version, only replacing XOR$_8$ with XOR$_4$. The added latency is one CMOS XOR$_2$ and two SET-MOS XOR$_4$ for each round of 4-way KA. In short, the area and latency complexities for $n \geq 4$ can be shown as,

$$\begin{cases} G(n) = 4.18n^{1.58} - 4n + 0.375, \\ L(n) = (\log_2 n - 2)(D_{X4} + 0.5D_{\oplus}) + 2D_{\oplus} + D_{AX4} + D_{X4}. \end{cases} \tag{5.18}$$

The following evaluation will focus on 1) area, and 2) the product of area and latency. The XOR$_4$ version and XOR$_8$ version of 4-way KA introduced in this sub-section are compared as follows.

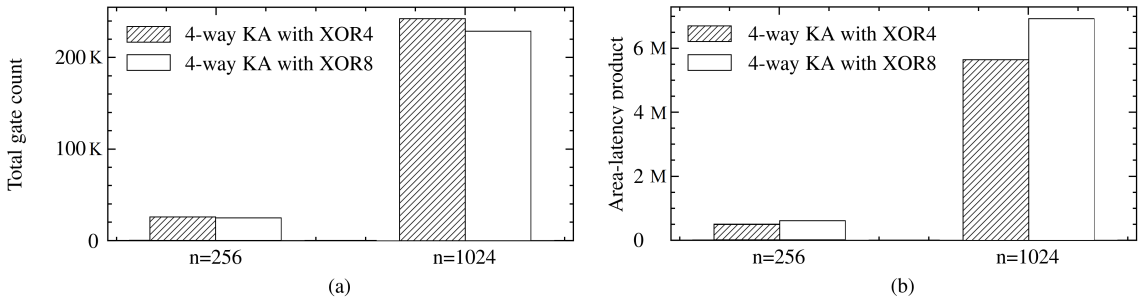


Fig. 5.14: Comparison of 4-way KA using XOR_4 versus XOR_8 counterparts respecting (a) total gate count and (b) area-latency product.

When $n \in (256, 1024)$, we bring n into $G(n)$, $L(n)$ in (5.13) and (5.18), and also bring the particular gate delays from Table 5.9 into $L(n)$, yields the specific gate count and latency. The area-latency product is the product of these two specific values, which is shown in Figure 5.14(b). From Figure 5.14(a), it can be seen that the XOR_8 version has a lower area, but from Figure 5.14(b), it can also be concluded that its area comes at the cost of more latency, resulting in a weaker area-latency product than the XOR_4 version. Because of the high cost in latency, the XOR_8 version of the 4-way KA will not be covered in subsequent comparisons.

5.4.3.2 4-Way Block Recombination Using SET-MOS Gates

Let A and B be two polynomials with degree $n-1$ in $GF(2^n)$. They can be represented as (3.15) in four equal-length polynomials with degree $\frac{n}{4} - 1$. Their product $AB = C$ can be expressed as (3.19) and (3.20). (5.19) gives the application scenario of SET-

MOS combinational gates in the 4-way block recombination.

$$
\begin{aligned}
&C_0 = \underbrace{A_0B_0}_{CM+RC}, \quad C_1 = \underbrace{\underbrace{A_0B_1}_{CM} + A_1B_0}_{\underbrace{CM+CA_{AndXor3}}_{RC}}, \quad C_2 = \underbrace{\underbrace{A_0B_2}_{CM} + \underbrace{A_1B_1}_{CM} + A_2B_0}_{\underbrace{CM+CA_{AndXor4}}_{RC}},\\
&C_3 = \underbrace{\underbrace{\underbrace{A_0B_3}_{CM} + \underbrace{A_1B_2}_{CM}}_{CM+CA_{AndXor4}} + A_2B_1 + A_3B_0}_{\underbrace{CM+CA_{AndXor3}}_{RC}},\\
&C_4 = \underbrace{\underbrace{A_1B_3}_{CM} + \underbrace{A_2B_2}_{CM} + A_3B_1}_{\underbrace{CM+CA_{AndXor4}}_{RC}}, \quad C_5 = \underbrace{\underbrace{A_2B_3}_{CM} + A_3B_2}_{\underbrace{CM+CA_{AndXor3}}_{RC}}, \quad C_6 = \underbrace{A_3B_3}_{CM+RC}.
\end{aligned}
\tag{5.19}
$$

Briefly, the blocks' area complexities in (5.19) can be summarized as follows.

$$
\begin{cases}
G^{CPF}(\frac{n}{4}) & = \frac{1}{9}n^{\log_2 3} - \frac{n}{4},\\
G^{CM}(\frac{n}{4}) & = G^{CM+CA}(\frac{n}{4}) = \frac{1}{9}n^{\log_2 3},\\
G^{CM+RC}(\frac{n}{4}) & = \frac{157}{648}n^{\log_2 3} - \frac{n}{2} + \frac{3}{8},\\
G^{RC}(\frac{n}{4}) & = \frac{13}{72}n^{\log_2 3} - \frac{n}{2} + \frac{3}{8}
\end{cases}
$$

And the area complexity of (5.19) is,

$$
\begin{cases}
G(n) = & 8G^{CPF}(\frac{n}{4}) + 8G^{CM}(\frac{n}{4}) + 6G^{CM+CA}(\frac{n}{4}) +\\
& 2G^{CM+RC}(\frac{n}{4}) + 5G^{RC}(\frac{n}{4}) + \frac{3n}{2} - 6,\\
G(n) = & \frac{2483}{648}n^{\log_2 3} - \frac{11}{2}n + \frac{21}{8}.
\end{cases}
$$

When considering the latency complexity, from (5.19) we can see four different paths:

1. $A|B \xrightarrow{CPF} \hat{A}_0|\hat{B}_0, \hat{A}_3|\hat{B}_3 \xrightarrow{CM+RC} C_0, C_6 \xrightarrow{Addition} C.$

2. $A|B \xrightarrow{CPF} \hat{A}_i|\hat{B}_j, i,j \in [0,3] \xrightarrow{CM} \hat{A}_0\hat{B}_1, \hat{A}_2\hat{B}_3 \xrightarrow{CM+CA_{AndXor3}} \hat{A}_0\hat{B}_1+\hat{A}_1\hat{B}_0, \hat{A}_2\hat{B}_3+\hat{A}_3\hat{B}_2 \xrightarrow{RC} C_1, C_5 \xrightarrow{Addition} C.$

3. $A|B \xrightarrow{CPF} \hat{A}_i|\hat{B}_j, i,j \in [0,3] \xrightarrow{CM} \hat{A}_0\hat{B}_2|\hat{A}_1\hat{B}_1, \hat{A}_1\hat{B}_3|\hat{A}_2\hat{B}_2 \xrightarrow{CM+CA_{AndXor4}} \hat{A}_0\hat{B}_2+\hat{A}_1\hat{B}_1+\hat{A}_2\hat{B}_0, \hat{A}_1\hat{B}_3+\hat{A}_2\hat{B}_2+\hat{A}_3\hat{B}_1 \xrightarrow{RC} C_2, C_4 \xrightarrow{Addition} C.$

4. $A|B \xrightarrow{CPF} \hat{A}_i|\hat{B}_j, i,j \in [0,3] \xrightarrow{CM} \hat{A}_0\hat{B}_3|\hat{A}_1\hat{B}_2 \xrightarrow{CM+CA_{AndXor4}} \hat{A}_0\hat{B}_3+\hat{A}_1\hat{B}_2+\hat{A}_2\hat{B}_1 \xrightarrow{CM+CA_{AndXor3}} \hat{A}_0\hat{B}_3+\hat{A}_1\hat{B}_2+\hat{A}_2\hat{B}_1+\hat{A}_3\hat{B}_0 \xrightarrow{RC} C_3 \xrightarrow{Addition} C.$

It is readily apparent that the path through C_3 is critical path.

$$L(n) = L^{CPF}(\frac{n}{4}) + D_{\otimes} + D_{AX4} + D_{AX3} + L^{RC}(\frac{n}{4}) + D_{\oplus}.$$

Consequently, the area and delay complexity of 4-way block recombination using SET-MOS gates can be summarized as,

$$\begin{cases} G(n) = 3.83n^{1.58} - 5.5n + 2.625, \\ L(n) = (\log_2 n - 2)(D_{X4} + 0.5D_{\oplus}) + D_{AX4} + D_{AX3} + D_{\otimes} + D_{\oplus}. \end{cases} \tag{5.20}$$

When n equals 256, 4-way block recombination using SET-MOS gates of (5.20) shows an $(39260 - 23735)/39260 = 40\%$ advantage in area compared with the area-oriented 4-way block recombination method in [24]. Latency of this SET-MOS method is longer than the CMOS counterpart, while the area-delay product of the SET-MOS method also achieves a reduction of $\frac{39260\times12.3-23735\times18.81}{39260\times12.3} = 7.5\%$.

5.5 Area-Efficient Hybrid Multipliers

It was reported that a hybrid KA architecture that includes both base multipliers and KA recursions is more efficient than a multiplier of pure KA recursion [51, 80]. The base multipliers can take up to 57% of the whole KA multiplier in terms of gate count

[80]. In this section, hybrid 2-way and 4-way SBM-KA architectures are proposed using SET-MOS gates to achieve the maximal gate count reduction.

5.5.1 2-Way SBM-KA Using SET-MOS Gates

In a hybrid KA multiplier, base multipliers with SBM are conventionally implemented with both AND and XOR logic gates. We propose to utilize the new SET-MOS combinational gates for replacing XOR gates and most of AND gates such that a significant reduction in gate count can be achieved for the base multipliers. The area and latency complexities of base multiplier is shown in (5.3), (5.4) and (5.5).

The base multiplier size can affect the gate count requirement of the whole SBM-KA multiplier. An optimal size of base multipliers can lead to better area performance of the hybrid SBM-KA multiplier. The previous chapter gave two approaches to finding the optimal value. Here we assume the base multiplier size is m; the component process of the whole SBM-KA multiplier is from (4.3) and then optimized with (4.19), where the multiplier's reconstruction process follows Table 4.3. The recursive area and latency complexities with initial conditions can be given as follows, for $0 < m < n$, m and n are positive powers of 2.

$$\begin{cases} G(m) = m^2, \\ L(m) = (m-1)D_{AX3} + D_{\otimes}, \\ G(n) = 3G(\frac{n}{2}) + 2n - 1, \\ L(n) = L(\frac{n}{2}) + D_{X4} + D_{\oplus}. \end{cases} \tag{5.21}$$

The non-recursive area and latency complexities for an n-bit hybrid SBM-KA multiplier using the CMOS AND_2, XOR_2, SET-MOS XOR_4 and $AndXor_3$ can be given in

(5.22).

$$\begin{cases} G(n) & = (m^2 + 4m - 0.5)(\frac{n}{m})^{1.58} - 4n + 0.5, \\ L(n) & = \log_2 \frac{n}{m}(2D_{X4} + D_{\oplus}) + D_{\otimes} + (m-1)D_{AX3}. \end{cases} \tag{5.22}$$

It can be seen that the coefficient of the highest order term in the area complexity $G(n)$ shown in (5.22) can be extracted separately. We denote the coefficient of $n^{1.58}$ in (5.22) by $K(m) = (m^2+4m-0.5)/m^{1.58}$, then $GF(n)$ can be asymptotically lowest when $K(m)$ reaches the minimum value. Figure 5.15 shows the curve of $K(m)$ with m.

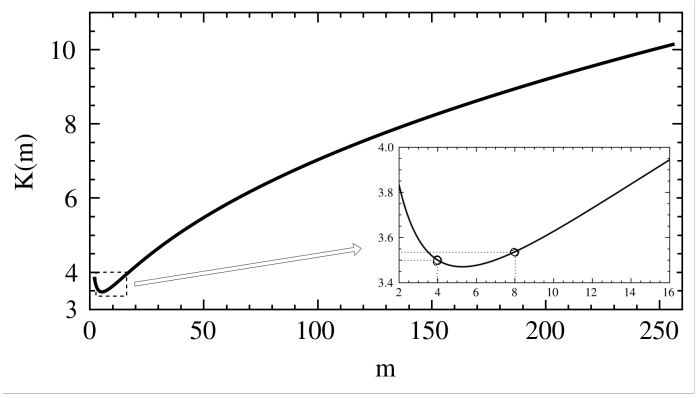


Fig. 5.15: Sub-quadratic term's coefficient $K(m)$ with respect to base multiplier's size m.

Since m should be a positive power of 2, it can be found in Figure 5.15 that the asymptotically lowest $G(n)$ can be obtained when m equals 4. Considering the latency complexity, it consists of two parts- the base multipllier and the KA recursion. The structure base multiplier is shown in Figure 5.11(c). The latency of component and reconstruction processes are the same with 2-way KA using XOR_4. As a result, for the proposed 2-way SBM-KA, we obtain the recursive and non-recursive complexity

of gate count and latency in (5.21) and (5.22), respectively.

$$\begin{cases} G(n) & = 3G(\frac{n}{2}) + 2n - 1, \\ G(4) & = 16, \\ L(n) & = L(\frac{n}{2}) + D_{X4} + D_{\oplus}, \\ L(4) & = D_{AX4} + D_{AX3} + D_{\otimes}. \end{cases} \tag{5.23}$$

$$\begin{cases} G(n) & = 3.5n^{1.58} - 4n + 0.5, \\ L(n) & = (\log_2(n) - 2)(D_{X4} + D_{\oplus}) + D_{AX4} + D_{AX3} + D_{\otimes}. \end{cases} \tag{5.24}$$

As can be seen from (5.24), 2-way SBM-KA using SET-MOS gates achieves the lowest area complexity in this thesis. Its area performance and area-delay product will be compared in the next section.

5.5.2 4-Way SBM-KA Using SET-MOS Gates

In the 4-way SBM-KA architecture, we still use the SBM as base multipliers presented in (5.3) and (5.5); the component process and sub-multiplications can be implemented by the methods described in (3.17) and Table 3.3; and the reconstruction process is implemented by the methods described in (5.12), (5.14), (5.15), and (5.16). Here are the area and latency complexities with initial conditions presented recursively, where m and n are positive powers of 2 and $0 < m < n$.

$$\begin{cases} G(m) = m^2, \\ L(m) \leq \lceil \frac{m-1}{2} \rceil D_{AX4} + D_{\otimes}, \\ G(n) = 9G(\frac{n}{4}) + 5n - 3, \\ L(n) = L(\frac{n}{4}) + 2D_{X4} + D_{\oplus}. \end{cases}$$

Bringing the m-bit base multipliers to the 4-way SBM-KA recursions improved by SET-MOS gates, the non-recursive $G(n)$ and $L(n)$ can be evaluated as,

$$\begin{cases} G(n) & = (m^2 + 4m - 0.375)(\frac{n}{m})^{1.58} - 4n + 0.375, \\ L(n) & \leq \log_4 \frac{n}{m}(2D_{X4} + D_{\oplus}) + D_{\otimes} + \lceil \frac{m-1}{2} \rceil D_{AX4}. \end{cases} \tag{5.25}$$

We also investigate $K(m)$ in order to minimize the area complexity $G(n)$—noting that m, in this case, is a positive integer even power of 2. The trend of $K(m)$ vs. m, in this 4-way case, is similar to Figure 5.15. It is found that the optimal value of m for the asymptotically lowest $G(n)$ can be obtained as $m = 4$. Therefore, the complexity expressions for the 4-way modified KA can be given as follows.

$$\begin{cases} G(n) & = 3.51n^{1.58} - 4n + 0.375, \\ L(n) & = (\log_2(n) - 2)(D_{X4} + 0.5D_{\oplus}) + D_{AX4} + D_{AX3} + D_{\otimes}. \end{cases} \tag{5.26}$$

Comparing (5.24) and (5.26), it can be seen that the 4-way SBM-KA has a minor increase in area complexity compared to the 2-way SBM-KA, and in exchange for a latency reduction (50% of the XOR_2 gate delays). In more detail, this delay reduction can be explained as follows. Usually, 4-way SBM-KA can be directly derived from two rounds of 2-way SBM-KA, while we design a new 2-Step reconstruction in (5.12), (5.14), (5.15), (5.16) so that the output of P_8 is directly connected to Step 2. This makes the original critical path in each round of 4-way recursion from $A_iB_j, i, j \in [0, 3] \xrightarrow{2D_{\oplus}}$ Sub-multiplication $\xrightarrow{\Delta} P_8 \xrightarrow{2D_{X4}} Output$ become $A_iB_j, i, j \in [0, 3] \xrightarrow{2D_{\oplus}}$ Sub-multiplication $\xrightarrow{\Delta} P_8 \xrightarrow{D_{X4}} Output$, while the new critical path becomes $A_iB_j, i, j \in [0, 3] \xrightarrow{D_{\oplus}}$ Sub-multiplication $\xrightarrow{\Delta} P_2, P_5, P_6, P_7 \xrightarrow{2D_{X4}} Output$. This means for each round of 4-way SBM-KA, one $D_{\oplus}$ can be saved. This exchange allows the 4-way SBM-KA to guarantee an outstanding area performance while the area-latency product is comparable to other multiplication

architectures.

5.6 Complexity Comparison

A complexity comparison between the existing and proposed work is first made regarding the asymptotic savings of area complexities. And then, the multiplier's particular gate counts and latency in time units are compared between the proposed work and the others for the multiplier size ranging from 256 to 2048. As seen from Table 5.9, the gate delays of SET-MOS are higher than that of CMOS. In order to make sure that our proposed work does not cost too much latency while improving area performance, we also compare the area-delay product.

5.6.1 Asymptotic Complexity Comparison

This sub-section summarizes the area and latency complexities of the proposed methods using SET-MOS gates along with existing methods from the architecture perspective. The column of relative savings in Table 5.10 is obtained by comparing the coefficient of the sub-quadratic term, $n^{1.58}$, which asymptotically indicates the area performance of a method (lower is better). The proposed SET-MOS 2-way and 4-way work introduced in Sections 5.4.2 and 5.4.3 shows noticeable advantages in area complexity compared to the existing counterparts, as shown in Table 5.10. The proposed SET-MOS work on existing methods can achieve $109\% - 148\%$ relative saving in asymptotic area complexity, which is better than the CMOS works with that of $186\% - 205\%$, at some cost in latency. However, even with the SET-MOS improvement, these proposed SET-MOS work are still not as efficient as the final work, *i.e.*, the 2-way and 4-way SBM-KA using SET-MOS gates proposed in Section 5.5.

Table 5.10: Complexity comparison of sub-quadratic methods

2-way methods				
Method	Technology	Area complexity	Relative saving†	Latency complexity
[15]	CMOS	$7n^{1.58} - 8n + 2$	200%	$3\log_2(n)D_{\oplus} - D_{\oplus} + D_{\otimes}$
[23]		$7n^{1.58} - 8n + 2$	200%	$2\log_2(n)D_{\oplus} + D_{\otimes}$
[21, 22]		$6.5n^{1.58} - 7n + 1.5$	186%	$3\log_2(n)D_{\oplus} - D_{\oplus} + D_{\otimes}$
[24]		$6.5n^{1.58} - 8.5n + 2.5$	186%	$3\log_2(n)D_{\oplus} - D_{\oplus} + D_{\otimes}$
[80]	SET-MOS	$4.06n^{1.58} - 4n + 0.5$	113%	$\log_2(n)(D_{X4} + D_{\oplus}) - D_{X4} - 2D_{\oplus} + D_{\otimes}$
Proposed work on [15]		$4.16n^{1.58} - 4n + 0.5$	119%	$\log_2(n)(D_{X4} + D_{\oplus}) - D_{X4} + D_{AX4}$
Proposed work on [23]		$5.17n^{1.58} - 6n + 1.5$	148%	$\log_2(n)(D_{X4} + D_{\oplus}) - D_{X4} + D_{AX4}$
Proposed work on [21, 22]		$4.77n^{1.58} - 5n + 1$	136%	$2\log_2(n)D_{\oplus} + \log_2(n)D_{X4} - D_{\oplus} - D_{X4} + D_{AX4}$
Proposed work on [24]		$3.94n^{1.58} - 4n - 0.5$	113%	$\log_2(n)(D_{X4} + D_{\oplus}) - D_{X4} + D_{\otimes} + D_{AX3}$
Proposed final work		$3.5n^{1.58} - 4n + 0.5$	100%	$\log_2(n)(D_{X4} + D_{\oplus}) - 2D_{X4}$ $-2D_{\oplus} + D_{AX3} + D_{AX4} + D_{\otimes}$
4-way methods				
Method	Technology	Total gate count	Relative saving†	Latency
[21]	CMOS	$6.43n^{1.58} - 6.8n + 1.38$	183%	$2.5\log_2(n)D_{\oplus} + D_{\otimes}$
[24](latency-oriented)		$6.69n^{1.58} - 11n + 8.88$	191%	$2\log_2(n)D_{\oplus} - D_{\oplus} + D_{\otimes}$
[24](area-oriented)		$6.34n^{1.58} - 8.9n + 3.63$	181%	$2.5\log_2(n)D_{\oplus} - 2D_{\oplus} + D_{\otimes}$
Proposed work on [19]	SET-MOS	$4.18n^{1.58} - 4n + 0.375$	119%	$\log_2(n)(D_{X4} + 0.5D_{\oplus}) - D_{X4} + D_{\oplus} + D_{AX4}$
Proposed work on [24]		$3.83n^{1.58} - 5.5n - 2.625$	109%	$\log_2(n)(D_{X4} + 0.5D_{\oplus}) - 2D_{X4} + D_{\otimes} + D_{AX3} + D_{AX4}$
Proposed final work		$3.51n^{1.58} - 4n + 0.375$	100%	$\log_2(n)(D_{X4} + 0.5D_{\oplus}) - 2D_{X4}$ $-D_{\oplus} + D_{AX3} + D_{AX4} + D_{\otimes}$

† Asymptotic relative gate count savings. Obtained by comparing the coefficient of the sub-quadratic term in gate count.

It can be seen that the area saving of the proposed final work in 2-way methods saves $(113\% - 100\%)/113\% = 11.5\%$, compared to the existing work using only SET-MOS XOR_4 and not combinational gates [80]. The area saving of the proposed method shows $(186\% - 100\%)/186\% = 46.2\%$ when compared to the best existing work using CMOS technology. Among the multipliers based on 4-way methods shown in Table 5.10, the proposed final work has clear advantages over all the existing work in area. Compared to the best of existing work with CMOS gates [24] (area-oriented), the proposed final work utilizes nearly half the gate counts, as the asymptotic savings in gate count are $(181\% - 100\%)/181\% = 44.8\%$.

Table 5.10 shows that the circuit depth is proportional to $O(\log_2(n))$ for all the work in comparison. As discussed in Section 5.3, the propagation delays of SET-MOS gates are usually more extensive than that of CMOS gates. While the proposed methods require substantially lower gate counts, the proposed methods are expected to have higher latency because of the application of SET-MOS gates. The following subsections will discuss the gate-level comparison of multiplier implementations with the simulated gate delays in Table 5.9.

5.6.2 Gate-Level Comparison

The gate count, latency in time units and area-latency product are calculated for the proposed final work and several best existing work for multiplier size of practical interest, as shown in Table 5.11.

Table 5.11: Complexity comparison of cryptographic sizes

2-way methods											
Methods	Technology	$n = 256$					$n = 512$				
		Total gate count (G)		Latency (L, *ns*)		G×L(%)	Total gate count (G)		Latency (L, *ns*)		G×L(%)
[21, 22]	CMOS	40856	186.2%	15.55	77.3%	144.0%	124357	186.0%	17.5	77.3%	143.7%
[23]		43881	200.0%	11.0	54.7%	109.4%	133687	200.0%	12.3	54.3%	108.6%
[24]		40473	184.5%	15.5	77.3%	142.6%	123590	184.9%	17.5	77.3%	142.9%
[80]	SET-MOS	25585	116.6%	17.73	88.2%	102.8%	77778	116.4%	20.27	89.5%	104.1%
Proposed final work		**21940**	**100%**	**20.11**	**100%**	**100%**	**66843**	**100%**	**22.65**	**100%**	**100%**
Methods	Technology	$n = 1024$					$n = 2048$				
		Total gate count (G)		Latency (L, *ns*)		G×L(%)	Total gate count (G)		Latency (L, *ns*)		G×L(%)
[21, 22]	CMOS	376652	185.9%	19.45	77.2%	143.6%	1137121	185.9%	21.4	77.2%	143.4%
[23]		405153	200.0%	13.6	54.0%	108.0%	1223647	200.0%	14.9	53.7%	107.5%
[24]		375117	185.2%	19.45	77.2%	143.0%	1134050	185.4%	21.4	77.2%	143.0%
[80]	SET-MOS	235381	116.2%	22.81	90.6%	105.2%	710238	116.1%	25.35	91.4%	106.1%
Proposed final work		**202576**	**100%**	**25.19**	**100%**	**100%**	**611823**	**100%**	**27.73**	**100%**	**100%**

4-way methods											
Methods	Technology	$n = 256$					$n = 1024$				
		Total gate count (G)		Latency (L, *ns*)		G×L(%)	Total gate count (G)		Latency (L, *ns*)		G×L(%)
[21]	CMOS	40415	183.4%	13.6	74.9%	157.1%	372428	183.1 %	16.85	74.6 %	157.7%
[24] (area-oriented)		39260	178.2%	12.3	67.7%	120.7%	364703	179.3%	15.55	68.8%	123.4%
[24] (latency-oriented)		41024	186.2%	10.35	57.0%	106.1%	383225	188.4%	12.95	57.3%	108.0%
Proposed final work	SET-MOS	**22031**	**100%**	**18.16**	**100%**	**100%**	**203396**	**100%**	**22.59**	**100%**	**100%**

Among 2-way methods, the proposed final work outperforms all the other work in gate count and the product of gate count and latency. The gate counts savings of the proposed final work are at least 14%[2], compared to a recent work using similar SET-MOS technology [80]. The improvement in the area-latency product of the proposed contribution ranges from 2.7% for $n = 256$ to 5.7%[3] for $n = 2048$. Suppose a comparison is made with the existing CMOS results. In that case, the proposed 4-way SBM-KA architecture achieves around 46% gate count reduction with the area-latency product improvement of 7.0% - 8.6% for the different multiplier sizes.

The proposed 4-way SBM-KA using SET-MOS technology has also shown clear advantages on both gate count and area-delay product over all the existing CMOS work in comparison. The gate count improvements are from 46.3% to 46.9%, and the area-latency product improvements range from 5.7% to 7.4% for multiplier sizes $n = 256$ and 1024.

5.7 Summary

In this chapter, we proposed two multiple-input SET-MOS gates to implement combined AND/XOR logic operations. The SET-MOS combinational gates have a similar area cost as a two-input CMOS gate because they all use the same number of transistors. The proposed combinational gates' delays are moderately higher than their CMOS counterpart when implementing a multiplier. Furthermore, we applied these SET-MOS combinational gates to bit-parallel finite field multipliers to improve the area performance. Compared to the previous chapter's work [80], the proposed 2-way and 4-way SBM-KA architectures achieve considerable area reduction by using combinational gates. The comparison on area-latency product ensures that the cost of latency is within a reasonable range.

[2] $(116\% - 100\%)/116\% \approx 14\%$

[3] $(102.8\% - 100\%)/102.8\% = 2.7\%$ and $(106.1\% - 100\%)/106.1\% = 5.7\%$.

6 Conclusion and Future Work

6.1 Conclusion

Bit-parallel finite field multipliers are crucial for achieving high-performance cryptographic systems, as they highly affect the speed of computations. However, bit-parallel multipliers suffer from problems in large area requirements. Researchers and engineers continuously strive to develop algorithms and architectures to implement finite field multipliers efficiently. However, improvements made solely from the algorithm or architecture perspective, such as the KA's extensions, yield diminishing returns.

The algorithms and architectures are closely related to the device characteristics. The findings of this thesis can be summarized in three aspects. Firstly, we design 4-input XOR gates using the emerging SET-MOS technology. We also analyze application scenarios in these existing multiplier architectures where 4-input XOR can be effectively applied. Modifications are made to the existing multiplication architectures, including the quadratic method, SBM, and sub-quadratic methods, such as KA, to suit the SET-MOS XOR better. The SET-MOS XOR improved KA multiplier can result in a substantial 25%-30% enhancement in the bit-parallel multiplier's area performance, surpassing the most area-efficient multiplier that employs CMOS technology.

Secondly, we develop a hybrid architecture of SBM and KA implemented with SET-MOS XOR. The base multipliers of this hybrid architecture affect the complete multiplier's area performance. This thesis shows two methods for finding the suitable size of base multipliers, programmatically and mathematically. The proposed hybrid multiplier using SET-MOS XOR results in a 10% area improvement compared to the first aspect of work and a remarkable 37% improvement compared to the most area-efficient existing work.

Lastly, this thesis proposes novel combinational gates using SET-MOS technology. Specifically, it proposes new logic gates with SET technology that can perform $(a \wedge b) \oplus c$ and $(a \wedge b) \oplus c \oplus d$ logic operations. These operations are obtained by adjusting the oscillation characteristic of SET-MOS and observing the truth tables. This design approach for combinational gates is very different from CMOS designs. Combinational gates' application scenarios are explored for constructing area-efficient multipliers with complexity comparisons. The result shows that the 2-way SBM-KA using combinational gates achieves a 46% reduction in area and a 4% decrease in the area-latency product compared to the most area-efficient CMOS multiplier. The proposed 4-way SBM-KA outperforms other 4-way multiplication using CMOS by 46% in area and 6% in the area-latency product. Moreover, this thesis discusses the latency cost of using SET-MOS gates and compares the delays of SET-MOS XOR and combinational gates with CMOS-based solutions. The lower area-latency product validates that the area-efficient multiplier proposed in this thesis substantially reduces the area while its increase in latency is within acceptable bounds.

Overall, our work breaks the bottleneck of bit-parallel multipliers for further area improvements. It promotes high-speed cryptosystems to be implemented on smaller chips, which can provide fast and secure network communications. The proposed work is expected to be favoured for applications where high-speed elliptic curve computation is required, and the compact area is also of critical importance, such as satellite phones, wearable equipment and smart homes.

6.2 Future Work

The research work proposed in this thesis integrates the essence of promising SET technology from the device point of view with the recent research efforts on sub-quadratic multiplication architectures. It proposes innovative logic gates and modifications on architecture. It breaks the bottleneck encountered in the improvements of

finite field multipliers and provides a new perspective to make further improvements.

The finite field multipliers can be more efficient by further unleashing the capabilities of SET-MOS. There is significant potential for advancements in the two perspectives that can be further explored and promoted.

- From a gate-to-device point of view, this thesis presents combinational gates for $(a \wedge b) \oplus c$ and $(a \wedge b) \oplus c \oplus d$ by observing their input and output patterns. We have not yet developed a comprehensive methodology for the design of SET-MOS combinational gates. Therefore, a methodology is very likely to be sketched out in future works on how to design a combinational gate with arbitrary combinations of XOR and AND operations with 4-inputs or 3-inputs. With the help of the new methodology, it is possible to continue exploring the feasibility of $(a \wedge b) \oplus (c \wedge d)$, which this thesis has not successfully designed.

- An 8-input SET-MOS XOR gate was also designed when I was working on the thesis. However, its application has not been fully investigated. For the 2-way and 4-way hybrid multiplication covered in this thesis, the 4-input SET-MOS XOR is sufficient. (The 8-input XOR cannot play any area saving than the 4-input XOR in the 2-way hybrid multiplication; 8-input XOR in a 4-way KA brings a slight area improvement but high costs in latency.) Among the existing bit-parallel multiplication architectures, research is dedicated to k-way multiplication [54, 28, 53, 26]. Theoretically, a higher number of splits will lead to more partial products in the reconstruction process of multiplication architecture, giving more room for 8-input XOR to perform its multiple-input handling capability. Therefore, the 8-input XOR gate may find a more suitable application scenario in k-way KA-like multiplications.

Investigating the power consumption of SET-MOS multipliers can be another possible future research direction. The estimation of power consumption is usually

divided into two parts: dynamic power consumption and static power consumption.

- **Dynamic power consumption**: The power consumption in CMOS occurs when the transistor states switch. If the frequency of the switching, or we say the clock frequency, gets higher, the dynamic power consumption will increase in CMOS based work [81]. In SET-MOS based work, on the other hand, there is always a small current flowing in the SET part. Its power consumption is barely affected by frequency, and its dynamic power consumption is apparently lower than CMOS work for high clock frequency.

- **Static power consumption**: Static power consumption in CMOS based work is also known as leakage power, which arises from the leakage current that flows through transistors even when they are in the off state. As the feature size decreases, this part of the power consumption gradually becomes not negligible. In SET-MOS, the power consumption is mainly static power consumption because there is always current flowing in its SET part. Since the current is always present, the "leakage current" in SET-MOS is different from that of CMOS, which is temperature-dependent [82].

Detailed comparisons would require reasonable assumptions, such as clock frequency, temperature, supply voltage, technology, and so on, which needs more research efforts.

In addition, the issue of SET-MOS circuit reliability can be addressed in the future, with comparison to CMOS circuit. For example, transient faults are a concern when it comes to CMOS technology. With decreasing feature sizes, CMOS-based circuits become vulnerable to soft errors. The SET-based circuit, on the other hand, is more affected by background charge and temperature. These factors could impact the accuracy and reliability of these circuits, which can be discussed as future work and possibly to find methods to increase reliability.

REFERENCES

[1] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2020.

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

[3] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 2018.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[5] W. Diffie, "The first ten years of public-key cryptography," *Proceedings of the IEEE*, vol. 76, no. 5, pp. 560–577, 1988.

[6] L. Batina, S. B. Örs, B. Preneel, and J. Vandewalle, "Hardware architectures for public key cryptography," *Integration*, vol. 34, no. 1-2, pp. 1–64, 2003.

[7] G. M. De Dormale and J.-J. Quisquater, "High-speed hardware implementations of elliptic curve cryptography: A survey," *Journal of systems architecture*, vol. 53, no. 2-3, pp. 72–84, 2007.

[8] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[9] V. S. Miller, *Use of elliptic curves in cryptography*. Springer, 1986.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[11] I. Blake, G. Seroussi, and N. Smart, *Elliptic curves in cryptography*. Cambridge university press, 1999, vol. 265.

[12] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 449–460, 2003.

[13] B. Rashidi, "A survey on hardware implementations of elliptic curve cryptosystems," *arXiv preprint arXiv:1710.08336*, 2017.

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

[15] C. Paar, "A new architecture for a parallel finite field multiplier with low complexity based on composite fields," *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 856–861, 1996.

[16] M. Wang and I. F. Blake, "Bit serial multiplication in finite fields," *SIAM Journal on Discrete Mathematics*, vol. 3, no. 1, pp. 140–148, 1990.

[17] L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 19, pp. 149–166, 1998.

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

[19] A. A. Karatsuba and Y. P. Ofman, "Multiplication of many-digital numbers by automatic computers," in *Doklady Akademii Nauk*, vol. 145, no. 2. Russian Academy of Sciences, 1962, pp. 293–294.

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

[21] D. J. Bernstein, "Batch binary edwards," in *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Springer, 2009, pp. 317–336.

[22] 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.

[23] 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.

[24] 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, 2013.

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

[26] M. Cenk and F. Ozbudak, "Improved polynomial multiplication formulas over $\mathbb{F}_2$ using chinese remainder theorem," *IEEE Transactions on computers*, vol. 58, no. 4, pp. 572–576, 2008.

[27] 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.

[28] I. Oseledets, "Improved n-term Karatsuba-like formulas in $GF(2^n)$," *IEEE Transactions on Computers*, vol. 60, no. 8, pp. 1212–1216, 2010.

[29] Z. Dyka, P. Langendoerfer, and F. Vater, "Combining multiplication methods with optimized processing sequence for polynomial multiplier in $GF(2^k)$," in *Research in Cryptology: 4th Western European Workshop, WEWoRC 2011, Weimar, Germany, July 20-22, 2011, Revised Selected Papers 4*. Springer, 2012, pp. 137–150.

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

[31] M. A. Kastner, "The single-electron transistor," *Reviews of modern physics*, vol. 64, no. 3, p. 849, 1992.

[32] S. Mahapatra and A. M. Ionescu, *Hybrid CMOS single-electron-transistor device and circuit design*. Artech House, Inc., 2006.

[33] X. Zhang, *VLSI architectures for modern error-correcting codes*. CRC Press, 2017.

[34] N. Sklavos and X. Zhang, *Wireless security and cryptography: specifications and implementations*. CRC press, 2017.

[35] J. A. Beachy and W. D. Blair, *Abstract algebra*. Waveland Press, 2019.

[36] W. Geiselmann and H. Lukhaub, "Redundant representation of finite fields," in *Public Key Cryptography*. Springer, 2001, pp. 339–352.

[37] B. Sunar and C. K. Koç, "An efficient optimal normal basis type ii multiplier," *IEEE Transactions on Computers*, vol. 50, no. 1, pp. 83–87, 2001.

[38] H. Wu, "Bit-parallel polynomial basis multiplier for new classes of finite fields," *IEEE Transactions on Computers*, vol. 57, no. 8, pp. 1023–1031, 2008.

[39] H. Wu, M. A. Hasan, I. F. Blake, and S. Gao, "Finite field multiplier using redundant representation," *IEEE Transactions on Computers*, vol. 51, no. 11, pp. 1306–1316, 2002.

[40] S. H. Namin, H. Wu, and M. Ahmadi, "Low-power design for a digit-serial polynomial basis finite field multiplier using factoring technique," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 441–449, 2016.

[41] N. Petra, D. De Caro, and A. G. Strollo, "A novel architecture for galois fields $GF(2^m)$ multipliers based on Mastrovito scheme," *IEEE Transactions on Computers*, vol. 56, no. 11, pp. 1470–1483, 2007.

[42] H. Wu and M. A. Hasan, "Low complexity bit-parallel multipliers for a class of finite fields," *IEEE Transactions on Computers*, vol. 47, no. 8, pp. 883–887, 1998.

[43] C. K. Koc and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields," *IEEE Transactions on Computers*, vol. 47, no. 3, pp. 353–356, 1998.

[44] H. Fan, "A Chinese Remainder Theorem approach to bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 343–352, 2015.

[45] E. D. Mastrovito, "VLSI designs for multiplication over finite fields $GF(2^m)$," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes: 6th International Conference, AAECC-6 Rome, Italy, July 4–8, 1988 Proceedings 6*. Springer, 1989, pp. 297–309.

[46] B. Sunar and C. K. Koc, "Mastrovito multiplier for all trinomials," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 522–527, 1999.

[47] 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.

[48] 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.

[49] 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, 2010.

[50] J. von zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba multipliers for polynomials over $\mathbb{F}_2$," in *Selected Areas in Cryptography: 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers 12*. Springer, 2006, pp. 359–369.

[51] G. Zhou, H. Michalik, and L. Hinsenkamp, "Complexity analysis and efficient implementations of bit parallel finite field multipliers based on Karatsuba-Ofman algorithm on FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 7, pp. 1057–1066, 2009.

[52] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba algorithm for efficient implementations," *Cryptology ePrint Archive*, 2006.

[53] 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.

[54] M. Cenk and M. A. Hasan, "Some new results on binary polynomial multiplication," *Journal of Cryptographic Engineering*, vol. 5, pp. 289–303, 2015.

[55] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE journal of solid-state circuits*, vol. 32, no. 7, pp. 1079–1090, 1997.

[56] Y. Ono, H. Inokawa, and Y. Takahashi, "Binary adders of multigate single-electron transistors: Specific design using pass-transistor logic," *IEEE Transactions on nanotechnology*, vol. 1, no. 2, pp. 93–99, 2002.

[57] K. Likharev and A. Zorin, "Theory of the Bloch-wave oscillations in small Josephson junctions," *Journal of low temperature physics*, vol. 59, no. 3-4, pp. 347–382, 1985.

[58] C. Gorter, "A possible explanation of the increase of the electrical resistance of thin metal films at low temperatures and small field strengths," *Physica*, vol. 17, no. 8, pp. 777–780, 1951.

[59] D. Averin and K. Likharev, "Probable coherent oscillations at single-electron tunneling," *SQUID*, vol. 85, p. 197, 1985.

[60] K. Likharev, "Single-electron transistors: Electrostatic analogs of the DC SQUIDS," *IEEE transactions on magnetics*, vol. 23, no. 2, pp. 1142–1145, 1987.

[61] A. M. Ionescu, M. J. Declercq, S. Mahapatra, K. Banerjee, and J. Gautier, "Few electron devices: towards hybrid CMOS-SET integrated circuits," in *Proceedings of the 39th annual Design Automation Conference*, 2002, pp. 88–93.

[62] S. Mahapatra, A. M. Ionescu, and K. Banerjee, "A quasi-analytical SET model for few electron circuit simulation," *IEEE Electron device letters*, vol. 23, no. 6, pp. 366–368, 2002.

[63] S. Mahapatra, K. Banerjee, F. Pegeon, and A. M. Ionescu, "A CAD framework for co-design and analysis of CMOS-SET hybrid integrated circuits," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No. 03CH37486)*. IEEE, 2003, pp. 497–502.

[64] H. Inokawa, A. Fujiwara, and Y. Takahashi, "A multiple-valued logic with merged single-electron and mos transistors," in *International Electron Devices Meeting. Technical Digest (Cat. No.01CH37224)*. IEEE, 2001, pp. 7.2.1–7.2.4.

[65] X. Ou and N.-J. Wu, "Analog-digital and digital-analog converters using single-electron and MOS transistors," *IEEE transactions on nanotechnology*, vol. 4, no. 6, pp. 722–729, 2005.

[66] S. Mahapatra, V. Vaish, C. Wasshuber, K. Banerjee, and A. M. Ionescu, "Analytical modeling of single electron transistor for hybrid CMOS-SET analog IC design," *IEEE Transactions on Electron Devices*, vol. 51, no. 11, pp. 1772–1782, 2004.

[67] M. Bounouar, F. Calmon, A. Beaumont, M. Guilmain, W. Xuan, S. Ecoffey, and D. Drouin, "Single electron transistor analytical model for hybrid circuit design," in *2011 IEEE 9th International New Circuits and systems conference*. IEEE, 2011, pp. 506–509.

[68] Y. Takahashi, A. Fujiwara, Y. Ono, and K. Murase, "Silicon single-electron devices and their applications," in *Proceedings 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000)*. IEEE, 2000, pp. 411–420.

[69] T. Oya, T. Asai, T. Fukui, and Y. Amemiya, "A majority-logic device using an irreversible single-electron box," *IEEE Transactions on Nanotechnology*, vol. 2, no. 1, pp. 15–22, 2003.

[70] G. Deng and C. Chen, "Binary multiplication using hybrid MOS and multi-gate single-electron transistors," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 21, no. 9, pp. 1573–1582, 2012.

[71] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital integrated circuits*. Prentice hall Englewood Cliffs, 2002, vol. 2.

[72] C. Negre, "Efficient binary polynomial multiplication based on optimized Karatsuba reconstruction," *Journal of Cryptographic Engineering*, vol. 4, pp. 91–106, 2014.

[73] 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.

[74] S.-H. Choi and K.-J. Lee, "Efficient systolic modular multiplier/squarer for fast exponentiation over $GF(2^m)$," *IEICE Electronics Express*, vol. 12, no. 11, pp. 1–6, 2015.

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

[76] C. Zhang, C. Chen, and H. Wu, "Area-efficient finite field multiplication in $GF(2^n)$ using single-electron transistors," in *2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS).* IEEE, 2021, pp. 25–28.

[77] S. R. Pillutla and L. Boppana, "Low-latency area-efficient systolic bit-parallel $GF(2^m)$ multiplier for a narrow class of trinomials," *Microelectronics Journal*, vol. 117, pp. 1–9 (105 275), 2021.

[78] S. E. Mathe and L. Boppana, "Design and implementation of a sequential polynomial basis multiplier over $GF(2^m)$," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 11, no. 5, pp. 2680–2700, 2017.

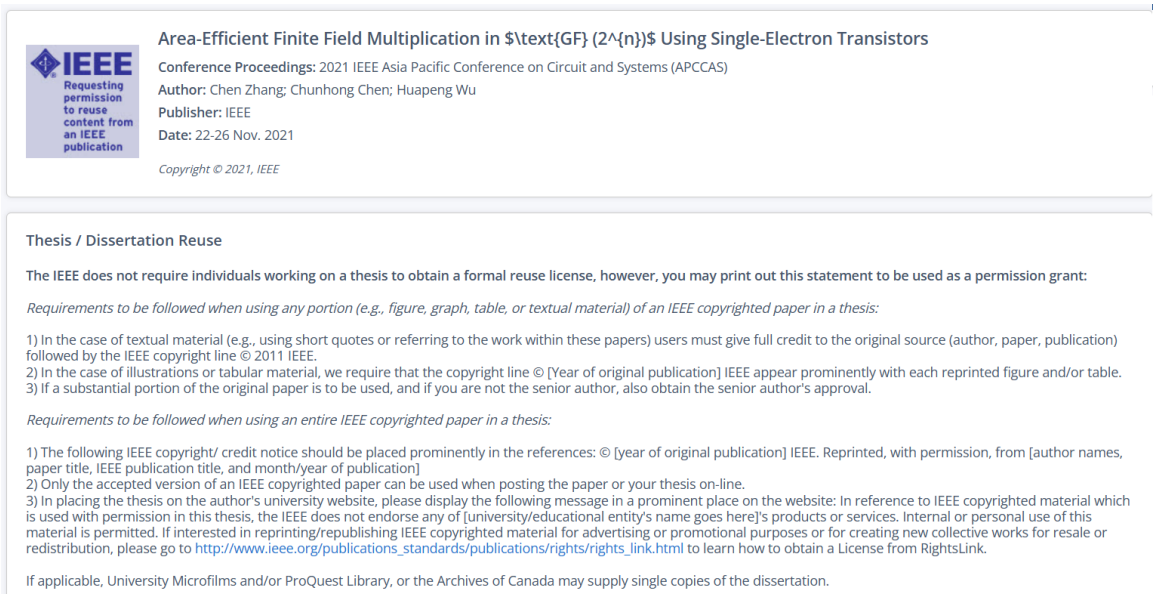
[79] W.-T. Huang, C.-H. Chang, C. W. Chiou, and F.-H. Chou, "Concurrent error detection and correction in a polynomial basis multiplier over $GF(2^n)$," *IET information security*, vol. 4, no. 3, pp. 111–124, 2010.

[80] C. Zhang, H. Wu, and C. Chen, "Area-efficient finite field multiplication using hybrid SET-MOS technology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 11, pp. 4358–4366, 2022.

[81] A. Wiltgen, K. A. Escobar, A. I. Reis, and R. P. Ribas, "Power consumption analysis in static CMOS gates," in *2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI)*. IEEE, 2013, pp. 1–6.

[82] S. Mahapatra, A. M. Ionescu, K. Banerjee, and M. J. Declercq, "Modelling and analysis of power dissipation in single electron logic," in *Digest. International Electron Devices Meeting,*. IEEE, 2002, pp. 323–326.

APPENDIX A: Permission for IEEE Publications



Area-Efficient Finite Field Multiplication in $\text{GF} (2^{n})$ Using Single-Electron Transistors

Conference Proceedings: 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS)
Author: Chen Zhang; Chunhong Chen; Huapeng Wu
Publisher: IEEE
Date: 22-26 Nov. 2021

Copyright © 2021, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

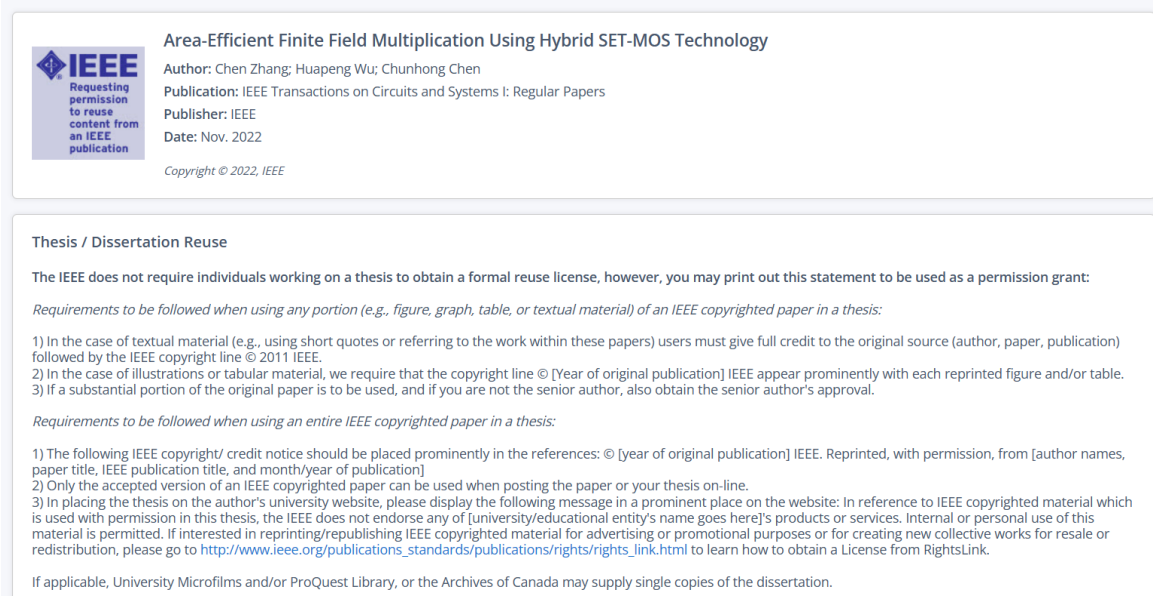
Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.



Area-Efficient Finite Field Multiplication Using Hybrid SET-MOS Technology

Author: Chen Zhang; Huapeng Wu; Chunhong Chen
Publication: IEEE Transactions on Circuits and Systems I: Regular Papers
Publisher: IEEE
Date: Nov. 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

APPENDIX B: Lemmas and Proofs

The proof of Lemma 3 is given as follows.

Lemma 3. *Let a, b, k, $k1$ be positive integers. Let $m = n^{k1}$, $n = b^k$ and assume $a \neq b$, $a \neq 1$. The solution to the recurrence relations*

$$\begin{cases} R_m = e; \\ R_n = aR_{n/b} + cn + d. \end{cases} \tag{B.1}$$

is shown as follows

$$R_n = (e + \frac{bc}{a-b}m + \frac{d}{a-1})(\frac{n}{m})^{\log_b a} + \frac{-bc}{a-b}n + \frac{-d}{a-1}. \tag{B.2}$$

Proof. Expanding (B.1) shows,

$$\begin{cases} R_m = e, \\ R_{mb} = aR_m + cmb + d, \\ R_{mb^2} = aR_{mb} + cmb^2 + d, \\ \quad \vdots \\ R_{n/b} = aR_{n/b^2} + c(n/b) + d, \\ R_n = aR_{n/b} + cn + d. \end{cases} \tag{B.3}$$

Taking $n = b^k$, $m = b^{k1}$ which gives

$$\begin{cases} R_{b^{k1}} = e, \\ R_{b^{k1+1}} = aR_{b^{k1}} + cb^{k1+1} + d, \\ R_{b^{k1+2}} = aR_{b^{k1+1}} + cb^{k1+2} + d, \\ \quad \vdots \\ R_{b^{k-1}} = aR_{b^{k-2}} + cb^{k-1} + d, \\ R_{b^k} = aR_{b^{k-1}} + cb^k + d. \end{cases} \tag{B.4}$$

Multiplying both sides by a^{k-k1-i} for $0 \leq i \leq k - k1$.

$$\begin{cases} a^{k-k1}R_{b^{k1}} = a^{k-k1}e, \\ a^{k-k1-1}R_{b^{k1+1}} = a^{k-k1}R_{b^{k1}} + a^{k-k1-1}cb^{k1+1} + a^{k-k1-1}d, \\ a^{k-k1-2}R_{b^{k1+2}} = a^{k-k1-1}R_{b^{k1+1}} + a^{k-k1-2}cb^{k1+2} + a^{k-k1-2}d, \\ \quad \vdots \\ aR_{b^{k-1}} = a^2R_{b^{k-2}} + acb^{k-1} + ad, \\ R_{b^k} = aR_{b^{k-1}} + cb^k + d. \end{cases} \tag{B.5}$$

Summing up both sides separately, we have,

$$R_{b^k} + \sum_{i=1}^{k-k1} a^i R_{b^{k-i}} = \sum_{i=1}^{k-k1} a^i R_{b^{k-i}} + c \sum_{i=0}^{k-k1-1} a^i b^{k-i} + d \sum_{i=0}^{k-k1-1} a^i + ea^{k-k1}. \tag{B.6}$$

Both sides have $\sum_{i=1}^{k-k1} a^i R_{b^{k-i}}$, which can be cancelled. Then summing the geometric

series,

$$
\begin{aligned}
c\sum_{i=0}^{k-k1-1} a^i b^{k-i} &= \frac{b^k((\frac{a}{b})^{k-k1}-1)}{\frac{a}{b}-1}c \\
&= \frac{b^k((\frac{a^k b^{k1}}{a^{k1}b^k})-1)}{\frac{a}{b}-1}c \\
&= \frac{(a^k b^{k1}-a^{k1}b^k)bc}{a^{k1}(a-b)} \\
&= a^{k-k1}b^{k1}\frac{bc}{(a-b)} - b^k\frac{bc}{(a-b)}.
\end{aligned} \tag{B.7}
$$

$$
\begin{aligned}
d\sum_{i=0}^{k-k1-1} a^i &= \frac{(a^{k-k1}-1)}{(a-1)}d \\
&= a^{k-k1}\frac{d}{(a-1)} - \frac{d}{(a-1)}.
\end{aligned} \tag{B.8}
$$

Now we have,

$$
\begin{aligned}
R_{b^k} &= a^{k-k1}b^{k1}\frac{bc}{(a-b)} - b^k\frac{bc}{(a-b)} + a^{k-k1}\frac{d}{(a-1)} - \frac{d}{(a-1)} + ea^{k-k1} \\
R_{b^k} &= ea^{k-k1} + a^{k-k1}b^{k1}\frac{bc}{(a-b)} + a^{k-k1}\frac{d}{(a-1)} - b^k\frac{bc}{(a-b)} - \frac{d}{(a-1)} \\
R_{b^k} &= (e + b^{k1}\frac{bc}{(a-b)} + \frac{d}{(a-1)})a^{k-k1} - b^k\frac{bc}{(a-b)} - \frac{d}{(a-1)}
\end{aligned} \tag{B.9}
$$

Making $b^{k1} = m$,$b^k = n$, and $a^{k-k1} = \frac{a^k}{a^{k1}} = \frac{a^{\log_b n}}{a^{\log_b m}} = (\frac{n}{m})^{\log_b a}$.

$$
R_n = (e + m\frac{bc}{a-b} + \frac{d}{a-1})(\frac{n}{m})^{\log_b a} + \frac{-bc}{a-b}n + \frac{-d}{a-1} \tag{B.10}
$$

■

VITA AUCTORIS

NAME: Chen Zhang

PLACE OF BIRTH: Hebei, China

YEAR OF BIRTH: 1992

EDUCATION: Zhuhai College of Science and Technology, Zhuhai, China
Bachelor of Computer Science and Technology, 2011-2015

University of Windsor, Windsor, ON, Canada
Master of Engineering, Electrical and Computer Engineering, 2016-2018

University of Windsor, Windsor, ON, Canada
Master of Applied Science, Electrical and Computer Engineering, 2018-2019

University of Windsor, Windsor, ON, Canada
Doctor of Philosophy, Electrical and Computer Engineering, 2019-2023