2017

**Novel Implementation of Finite Field Multipliers over GF(2^m) for Emerging Cryptographic Applications**

Nazeem Basha Shaik

*Wright State University*

Follow this and additional works at: [https://corescholar.libraries.wright.edu/etd_all](https://corescholar.libraries.wright.edu/etd_all)

Part of the [Electrical and Computer Engineering Commons](https://corescholar.libraries.wright.edu/etd_all)

Repository Citation

Shaik, Nazeem Basha, "Novel Implementation of Finite Field Multipliers over GF(2^m) for Emerging Cryptographic Applications" (2017). *Browse all Theses and Dissertations*. 1718. https://corescholar.libraries.wright.edu/etd_all/1718

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
Novel Implementation of Finite Field Multipliers over $GF(2^m)$ for Emerging Cryptographic Applications

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering

By

NAZEEM BASHA SHAIK
B.Tech, Rajiv Gandhi University of Knowledge Technologies, India, 2014

2017
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY
Nazeem basha Shaik ENTITLED NOVEL IMPLEMENTATION OF FINITE FIELD MULTIPLIERS
- OVER $GF(2^m)$ FOR EMERGING CRYPTOGRAPHIC APPLICATIONS BE ACCEPT IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Electrical
Engineering.

Jiafeng Xie, Ph.D.
Thesis Director

Brian Rigling, Ph.D.
Chair, Electrical Engineering

Committee on Final Examination

Jiafeng Xie, Ph.D.

Henry Chen, Ph.D.

Zhuang Yan, Ph.D.

Robert E.W. Fyffe, Ph.D.
Vice President for Research and
Dean of the Graduate School
Abstract

Shaik, Nazeem Basha. M.S.E.E., Department of Electrical Engineering, Wright State University, 2017. Novel Implementation of Finite Field Multipliers over $GF(2^m)$ for Emerging Cryptographic Applications.

Cryptographic and coding theory algorithms use arithmetic operations over finite fields. Finite field multiplications over $GF(2^m)$ are critical components for these systems. Well known irreducible polynomials are all-one-polynomials (AOP), equally spaced polynomial (ESP), trinomial and pentanomial. Due to its simple structure, AOP based multiplication is easy to implement and hence the AOP based multiplication of variable length can be used as a standard computation core. In this thesis, first of all, we employ low register complexity AOP based systolic multiplication core to propose multiplication over $GF(2^m)$ based on NIST recommended pentanomials. The proposed parallel and serial architectures use pre-computation (PC) modules to compute bits involve in multiplication and re-combination (RC) modules to combine computed bits from PC to form vectors which will reduce the multiplication complexity. The corresponding architecture based on the proposed algorithm is then synthesized by Xilinx ISE 14.1 on a Virtex 5 FPGA device and it is observed that the proposed structures has lower area-delay complexity than the best of existing designs. Second, we propose a novel obfuscation mechanism to equip multiplication over different irreducible polynomials and addition operations. Desired functionality of the proposed obfuscated structure is achieved through correct input sequence to controller (FSM). This is the first architecture proposed which can implement four types of polynomial multiplications and additions with obfuscated manner. The proposed architecture is synthesized and implemented in application specific integration circuits (ASIC) platform and have achieved excellent area-time performance.

**Key Words**- All one polynomials, finite field, hardware obfuscation, pentanomials, systolic architecture, trinomials.
Contents

1 Introduction ................................................................. 1
   1.1 Preliminary ......................................................... 1
   1.2 Report Outline .................................................. 2

2 Finite Field ............................................................... 4
   2.1 Introduction of Finite Field ................................. 4
   2.2 Binary Field ....................................................... 5

3 Polynomial Multiplication over $GF(2^m)$ .................. 6
   3.1 Polynomial Basis Representation over $GF(2^m)$ ....... 7
   3.2 Irreducible Polynomial ......................................... 7
      3.2.1 Definition of Irreducible Polynomial ............... 7
      3.2.2 Important Irreducible Polynomial based on Finite Field .... 7
   3.3 Polynomial Basis Multiplication Over $GF(2^m)$ ....... 8
      3.3.1 All-One Polynomial ................................. 9
      3.3.2 Trinomial ........................................ 10
      3.3.3 Pentanomial ........................................ 10
      3.3.4 Equally Spaced Polynomial ...................... 10

4 Low Register Complexity AOP based Bit Parallel and Bit Serial Systolic Multiplications over $GF(2^m)$ for General Pentanomials .... 12
   4.1 Proposed Bit Prallel-like AOP based Systolic Multiplier ........ 12
      4.1.1 Preliminaries ........................................ 12
      4.1.2 Proposed Architecture ............................... 13
      4.1.3 AOP Core ........................................ 20
      4.1.4 Irregular PE Core .................................. 23
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.5</td>
<td>Selective Addition Module</td>
<td>23</td>
</tr>
<tr>
<td>4.1.6</td>
<td>Pre-Computation and Re-Combination Modules</td>
<td>24</td>
</tr>
<tr>
<td>4.1.7</td>
<td>Example</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Proposed Bit Serial AOP based Systolic Multiplier</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Complexity Considerations</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>Addition and Multiplication of Polynomials over $GF(2^m)$</td>
<td>31</td>
</tr>
<tr>
<td>5.1</td>
<td>Polynomial Reduction to $GF(2^m)$ from $GF(2^{2m})$</td>
<td>31</td>
</tr>
<tr>
<td>5.1.1</td>
<td>All One Polynomial</td>
<td>31</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Trinomial</td>
<td>33</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Pentanomial</td>
<td>34</td>
</tr>
<tr>
<td>5.1.4</td>
<td>Equally Spaced Polynomial</td>
<td>35</td>
</tr>
<tr>
<td>5.2</td>
<td>Polynomial Multiplication</td>
<td>36</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Formation of Extended Basis $GF(2^{2m-1})$</td>
<td>37</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Reduction of $GF(2^{2m-1})$ to $GF(2^m)$</td>
<td>37</td>
</tr>
<tr>
<td>5.3</td>
<td>Novel Obfuscation Model</td>
<td>40</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Addition</td>
<td>40</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Multiplication</td>
<td>41</td>
</tr>
<tr>
<td>5.4</td>
<td>Hardware and Complexity</td>
<td>46</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>48</td>
</tr>
<tr>
<td>7</td>
<td>Future Research</td>
<td>49</td>
</tr>
<tr>
<td>7.1</td>
<td>Improving AOP Computation Core based Pentanomial Architecture</td>
<td>49</td>
</tr>
</tbody>
</table>
| 7.2     | Further Improvements and Applications of Addition/Multiplicaiton of Poly- 
|         | nomial Architecture | 50   |
| 8       | Publications | 51   |
| 9       | Reference | 52   |
List of Figures

4.1 Conventional Bit Systolic Pentanomial based Polynomial Multiplier Architecture where (a) Conventional $m$-bit Systolic Multiplier. (b) Internal Structure of Each Processing Element (PE). (c) Internal Structure of Shift adder (SA) Cell. ................................. 14
4.2 Bit Parallel Systolic Array Architecture. ................................. 21
4.3 Systolic AOP based Polynomial Multiplier. (a) Systolic array Architecture (b) Internal Structure of PE. ................................. 22
4.4 Bit Serial Systolic array Architecture ................................. 27

5.1 Polynomial Multiplication/Addition Architecture ................................. 36
5.2 $A$-rotate ................................. 44
5.3 Selective Buffer Block ................................. 45
5.4 $m$-bit Adder tree Block ................................. 45
5.5 $C$-rotate ................................. 46
## List of Tables

4.1 COMPARISON OF AREA-TIME COMPLEXITIES OF BIT-PARALLEL SYSTOLIC MULTIPLIERS ........................................... 28
4.2 COMPARISON OF AREA-TIME COMPLEXITIES OF SERIAL SYSTOLIC MULTIPLIERS ....................................................... 29
4.3 \( x^{163} + x^7 + x^6 + x^3 + 1 \) .................................................................................. 29
4.4 \( x^{283} + x^{12} + x^7 + x^5 + 1 \) .................................................................................. 29
4.5 \( x^{571} + x^{10} + x^5 + x^2 + 1 \) .................................................................................. 29
4.6 FPGA IMPLEMENTATION RESULTS OF BIT PARALLEL PENTANOMIALS FOR VARIOUS DEGREE \( m \) .......................... 30
4.7 SERIAL SYSTOLIC MULTIPLIERS: \( m = 163 \) ................................................................. 30
4.8 SERIAL SYSTOLIC MULTIPLIERS: \( m = 283 \) ................................................................. 30
4.9 SERIAL SYSTOLIC MULTIPLIERS: \( m = 571 \) ................................................................. 30
4.10 FPGA IMPLEMENTATION RESULTS OF BIT SERIAL PENTANOMIALS FOR VARIOUS DEGREE \( m \) ......................... 30

5.1 OBFUSCATED MODEL INPUT MODES .......................................................... 43
5.2 CONTROLLER OUTPUT SEQUENCES ................................................................. 43
5.3 COMPARISON OF AREA-TIME COMPLEXITIES OF POLYNOMIAL MULTIPLIERS ....................................................... 47
5.4 ASIC SYNTHESIS RESULTS OF POLYNOMIAL MULTIPLIER OVER DIFFERENT \( m \)-degree IRREDUCIBLE POLYNOMIALS .......................... 47
Chapter 1

Introduction

This chapter presents the outline of thesis. It presents basic ideas of the related work, previous proposed designs, and the contributions of this report.

1.1 Preliminary

In recent years, the finite field operations like addition, multiplication, squaring, and inversion have been used in various applications, such as cryptography, number theory and coding theory, etc.. Cryptosystem such as elliptic curve cryptography (ECC), which is used as a public key cryptography in various devices for secure data encryption and decryption (server and client connection), requires mainly finite field arithmetic operations.

There are basically three types of bases: dual basis [15-18], polynomial basis [2-10] and normal basis [11-14]. Normal basis works well in division and inverse operations but it requires extra basis conversion [13]. Polynomial basis does not require basis conversion and hence it is more suitable for very large scale integration (VLSI) applications.

Well known irreducible polynomials for field multiplications include all-one-polynomials (AOP), trinomials, and pentanomials. National Institute of Standards and Technology (NIST) has recommended two trinomials and three pentanomials for cryptosystems usage [2], while AOP based cryptosystems are not secure because of their simple and regular structures. Therefore, in this thesis, we focus more on the trinomial and pentanomial based multipliers.

There are basically two kinds of structures for finite field multipliers: systolic and non-systolic architectures. Non-systolic architectures mainly account for low ara complexity
which is scalable with the cost of low throughput. Systolic multipliers over \( GF(2^m) \) are preferred in high-performance applications because of their modularity and regularity \([4-5]\). They have high throughput however these structures possess high area-complexity, especially register complexity \([2]\). There are different architectures proposed for systolization such as bit serial, digit serial, and bit parallel. Many efforts have been reported to reduce the register-complexity in systolic multipliers \([4-7], [14-27]\).

Since pentanomials provide better security than the other irreducible polynomials, in this thesis, we will propose pentanomial based multiplication based on an AOP based multiplication core. AOP is a primitive irreducible polynomial which has huge applications in various fields. But as far as security concern, the AOP based design is likely to get crackable. The proposed architecture, however, will employ an AOP based computation core to achieve pentanomial based multiplication (including a regular AOP core will not affect its security level because of the overall multiplication complexity).

Besides that, many efforts reported to decrease area complexity of finite field multiplier in order to make suitable for the cryptosystems on FPGA and wearable devices \([1],[49],[52]\), e.g., ECC implementation in wearable devices require low area complexity rather than in latency and throughput. In chapter 4, we propose a multiplication algorithm which concentrate more on area constraint rather latency and throughput. In this architecture multiplication algorithm split into three sub components to complete a multiplication process. This provides opportunity to obfuscate the design for potential anti-counterfeit attacks. Detailed processes will be provided in following chapters.

### 1.2 Report Outline

The following parts of the thesis are organized as follows:

Chapter 2 shows the processes of mathematical formulation of polynomial basis multiplication over \( GF(2^m) \). Several classes of irreducible polynomials are shown in this chapter.

Chapter 3 presents the polynomial basis multiplication over \( GF(2^m) \). We propose a new algorithm which uses AOP-based computation core to implement bit serial and bit parallel pentanomial multipliers. This proposed structure lowers register complexity and also improves latency when compared with the existing ones.

Chapter 4 presents polynomial arithmetic (addition and multiplication) algorithms within
a single hardware with obfuscation strategy. The proposed design can compute multiplication over $GF(2^m)$ modulo four types of irreducible polynomials as well as addition of polynomials. This architecture is well suitable for area constraint devices like chip enabled secure systems and wearable devices.

Chapters 5 and 6 present the conclusion of the whole thesis as well as our future research plan.
Chapter 2

Finite Field

This chapter presents some basic mathematical background knowledge about finite field. In the following sections, we introduce the definition of finite field and algorithm operations in finite field. Besides this, we introduce a special finite field named Binary Field, as well as those field operations based on this field.

2.1 Introduction of Finite Field

Finite field also known as Galois field, is a field of elements which are finite. The total number of elements in finite field is called the order of finite field and the order is likely known to be root of that field. For example, a finite field with 8 elements is known as finite field of order 8 expressed as $f_8$ and the root of finite field is 8, integers modulo $8$. Two types of finite fields used in most of the Addition and multiplication are two operations which can be involved in field $F$. So for subtraction and division, we need to use addition and multiplication to define them. We define subtraction in terms of addition: $a - b = a + (-b)$, for $a, b \in F$, and $-b$ is the unique element in $F$ ($-b$ is the negative of $b$, $b + (-b) = 0$). Similarly, division is defined in terms of multiplication: $a/b = a \cdot b^{-1}$, for $a, b \in F$, $b \neq 0$, and $b^{-1}$ is a unique element in $F$ ($b^{-1}$ is the inverse of $b$, $b \cdot b^{-1} = 1$).

Here is an example of arithmetic operation in finite field.

The elements of $GF(5)$ are $\{0, 1, 2, 3, 4\}$, then the arithmetic operations in $GF(5)$ are:

(1). Addition: $3 + 4 = 7 \mod(5) = 2 \mod(5) = 2$.

(2). Subtraction: $2 - 4 = -2 \mod(5) = -2 +5 \mod(5) = 3 \mod(5) = 3$. 

4
(3). Multiplication: $3 \times 2 = 6 \mod(5) = 1$.

2.2 Binary Field

If the finite field with the order $2^m$ is called binary field, the integer $m$ is called the degree of the field. There are several ways to construct $GF(2^m)$, such as the binary field $GF(2^m)$ can be consisted by $2^m$ possible bit strings with length $m$. For example

$$GF(2) = \{0, 1\}$$

$$GF(2^3) = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

The addition and multiplication are as follows:

Addition:

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 1 = 0$$

The addition of two elements should be optimized by bitwise addition modulo 2.

Such as $(11000) + (10111) = (01111)$

Multiplication:

$$0 \times 0 = 0, \quad 0 \times 1 = 0, \quad 1 \times 1 = 1$$

Or we can use other two ways to construct $GF(2^m)$: one way is to use polynomial basis representation, the other way is to use normal basis representation. Here we just introduce polynomial basis representations in the next chapter.

$GF(2)$ is the simplest finite field, it is consisted only by two elements, 0 and 1. For addition and multiplication in $GF(2)$, the final result needs to be modulo 2. According to the results, one can see that the addition is equivalent to logical XOR operation and the multiplication is equivalent to logical AND process.
Chapter 3

Polynomial Multiplication over \( GF(2^m) \)

Assume there is a field \( F \), and the elements \( a_n, a_{n-1}, a_{n-2}, \ldots, a_1, a_0 \) belong to field \( F \). Then the expression with the form of \( f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0 \) is called a polynomial with degree \( n \) over \( F \). The element \( a_i \) is called the coefficient of \( x^i \) in \( f(x) \), and \( a_n \neq 0 \). Depending on the each power of \( X \) and corresponding coefficient we can justify if this two polynomials are equal or not.

Polynomial ring \( R[X] \) is a ring which can be formed by the set of polynomials in one or more variables (such as \( x \)) with coefficients in another ring \( R \) (or field). In \( X \) over a field \( P \) is defined as the set of expressions with the form \( f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0 \), where \( a_n, a_{n-1}, a_{n-2}, \ldots, a_1, a_0 \) are the coefficients which are the elements of field \( P \), then these coefficients can form a ring, called polynomial ring \( P[x] \).

Polynomials can be equipped with arithmetic operations. Two standard operations for polynomials are addition and multiplication. Here is an example: let \( A(x) = x^4 + 3x^2 + 2 \) and \( B(x) = 3x^2 + x + 4 \) be elements of polynomial ring \( R_4[x] \). The multiplications and addition of these two polynomials are:

\[
A(x) \cdot B(x) = 3x^6 + x^5 + x^4 + 3x^3 + 2x^2 + 2x + 4
\]

\[
A(x) + B(x) = x^4 + 2x^2 + x + 2
\]
3.1 Polynomial Basis Representation over $GF(2^m)$

The way to use polynomial basis representation to construct binary field $GF(2^m)$ is the elements of $GF(2^m)$ are binary polynomials with the degree at most $m - 1$. In this condition the polynomials’ coefficients are in the field of $GF(2)$. Such as, for finite field $GF(2^m)$, the elements in this field are the polynomials \{0, 1, x, x + 1, x^2, x^2 + 1, ..., x^{m-1} + x^{m-2} + ... + x + 1\}, where the $x$ is a root of an irreducible polynomial $f(\alpha)$ over $GF(2)$, and the polynomial coefficients are $GF(2) = \{0, 1\}$, where $f(x) = 0$.

We can use an example to show the exactly elements for finite field based on polynomials. The elements of finite field $GF(2^3)$ are as follows: (000), (001), ..., (111).

3.2 Irreducible Polynomial

3.2.1 Definition of Irreducible Polynomial

The irreducible polynomial means a polynomial ($f(x)$) which couldn’t be factored into the product of two non-constant polynomials over the same field ($f(x) \neq g(x)h(x)$). Whether the polynomial can be factored or not should depend on the field and ring to which the coefficients are considered belong to. For instance, the polynomial $f(x) = x^2 - 2$ is irreducible if the coefficients -1 and 2 are considered belong to integers. But if we consider the coefficients are belong to real numbers, then the polynomial $f(x)$ can be factored as $(x + \sqrt{2})(x - \sqrt{2})$.

3.2.2 Important Irreducible Polynomial based on Finite Field

All primitive polynomials are irreducible polynomials. The irreducible polynomial can be used to represent the elements of a finite field. For example, there is an irreducible polynomial over GF(2) with degree m $f(x) = x^m + f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + ... + f_1x + f_0$. Let $\alpha \in GF(2^m)$ be the root of this irreducible polynomial, then the set \{1, $\alpha$, $\alpha^2$, ..., $\alpha^{m-2}$, $\alpha^{m-1}\} can constitute the polynomial basis in $GF(2^m)$. So this polynomial basis can be used to represented the elements in $GF(2^m)$ with the most degree as $m - 1$, and the form is $GF(2^m) = \{a(x)|a(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_2x^2 + a_1x + a_0, a_j \in GF(2)\}$. There are several kinds of polynomial, and we usually choose all one polynomials (AOP), ESP, trinomials, and pentanomials to represent the elements of finite field.
All One Polynomials (AOP)

AOP means the polynomial in which all coefficients are one, and over the finite field $GF(2)$. AOP is an irreducible polynomial only if $m + 1$ is prime and $2$ is a primitive modulo $m + 1$. For example, the value of $m$ should be $m \in \{2, 4, 6, 10, 12, \ldots\}$, and the form of AOP can be expressed as

$$f(x) = \sum_{i=0}^{m} x^i = x^m + x^{m-1} + x^{m-2} + \ldots + x^2 + x + 1 \quad (3.1)$$

The degree is $m$ ($m + 1$ should be prime number). AOP has simple form, so usually it can be used to incorporate in other efficient algorithms and multiplications.

Trinomial

Trinomial is a polynomial consisting of three non-zero terms. We usually use trinomial in mathematics, such as

- $5x + y + 6z$, where $x, y, z$ are variables
- $m^2 + 4n + p$, where $m, n, p$ are variables
- $Ax^m + Bx^n + Cx^p$, where $x$ is variable, $m, n, p$ are nonnegative integers

Trinomials over finite field are widely applied in different areas, such as using trinomial to implement multiplier over finite field. And this kind of multiplier is the core component for ECC. The form of trinomial over $GF(2^m)$ is $f(x) = x^m + x^k + 1$.

The NIST [11] has recommended five binary finite fields for ECC implementation. There are two binary fields generate trinomials: $f(x) = x^{233} + x^{73} + 1$ and $f(x) = x^{409} + x^{87} + 1$.

3.3 Polynomial Basis Multiplication Over $GF(2^m)$

There is an irreducible polynomial over $GF(2^m)$: $f(x) = x^m + f_{m-1}x^{m-1} + f_{m-2}x^{m-2} + \ldots + f_2x^2 + f_1x + f_0$, $f_i \in GF(2)=\{0,1\}$.

The set $\{1, x, x^2, \ldots x^{m-2}, x^{m-1}\}$ is the polynomial basis over finite field $GF(2^m)$. Comparing characters of this set and the form of polynomial, we can use polynomial to represent
the elements of \( GF(2^m) \): that is, we can represent the elements of this set by defining \( f(x) \) as 
\[
\beta(x) = \beta_{m-1}x^{m-1} + \beta_{m-2}x^{m-2} + \ldots + \beta_1x + \beta_0, \text{ where } \beta_i \in GF(2).
\]
Let \( a(x) \) and \( b(x) \) be two field elements, and \( c(x) \) is the product of them, then we can have \( c(x) = a(x)b(x) \mod f(x) \).

Therefore, the polynomial basis multiplication is consisted by two steps: one is polynomial multiplication and another reduction modulo an irreducible polynomial. The product of \( d(x) = a(x)b(x) \) is a polynomial with degree \( 2m - 1 \), so after the modular reduction \( c(x) = a(x)b(x) \mod f(x) \), the degree \( 2m - 1 \) of polynomial \( d(x) \) is reduced by degree \( m \) irreducible polynomial \( f(x) \). The multiplication process can be represented by matrix as follows (fig. 3.1 and fig. 3.2),

The process of \( C = A \cdot B \mod f(x) \) can be represented as follows (fig. 3.3).

### 3.3.1 All-One Polynomial

Irreducible AOP for \( GF(2^m) \) is of the form \( f(x) = x^m + x^{m-1} + x^{m-2} + \ldots + x + 1 \) where \( (m + 1) \) is prime and 2 is primitive modulo \( (m + 1) \). Let the root of \( f(x) \) be \( \alpha \) then the set \( \{\alpha^{m-1} + \alpha^{m-2} + \ldots + \alpha + 1\} \) forms the polynomial basis

\[
f(\alpha) = \alpha^m + \alpha^{m-1} + \alpha^{m-2} + \ldots + \alpha + 1 = 0 \implies \alpha^m = \alpha^{m-1} + \alpha^{m-2} + \ldots + \alpha + 1
\]

Consider [24]

\[
f(\alpha) + \alpha f(\alpha) = \alpha^{m+1} + 1 = 0 \implies \alpha^{m+1} = 1
\]

Let us consider an extended polynomial basis \( \{\alpha^m + \alpha^{m-1} + \alpha^{m-2} + \ldots + \alpha + 1\} \) and choose a polynomial \( A \) in extended polynomial basis such that \( a_m = 0 \). Then
\[ \alpha \cdot A = \alpha \sum_{i=0}^{m} a_i \alpha^i = \alpha \left( a_0 + a_1 \alpha + a_2 \alpha^2 + \ldots + a_{m-1} \alpha^{m-1} + a_m \alpha^m \right) = a_0 \alpha + a_1 \alpha^2 + a_2 \alpha^3 + \ldots + a_{m-1} \alpha^m + a_m \alpha^{m+1} = a_m + a_0 \alpha + a_1 \alpha^2 + \ldots + a_{m-2} \alpha^{m-1} + a_{m-1} \alpha^m \quad (\text{from eq. (3.3)}) \]

In general,

\[ \alpha^k \cdot A = a_{m-k+1} + a_{m-k+2} \alpha + \ldots + a_{m-1} \alpha^{k-2} + a_m \alpha^{k-1} + a_0 \alpha^k + \ldots + a_{m-k} \alpha^m \quad (3.4) \]

### 3.3.2 Trinomial

A polynomial with three non-zero terms is called Trinomial. In other words, an irreducible polynomial which is of the form \( g(x) = x^{m-1} + x^n + 1 \) (where \( n \leq (m - 1)/2 \)). Let \( \beta \) be the root of \( g(x) \) then

\[ g(\beta) = \beta^m + \beta^n + 1 = 0 \implies \beta^m = \beta^n + 1 \quad (3.5) \]

### 3.3.3 Pentanomial

A polynomial with five non-zero terms is called Pentanomial. In other words, an irreducible polynomial which is of the form \( h(x) = x^{m-1} + x^{p_3} + x^{p_2} + x^{p_1} + 1 \) (where \( 1 < p_1 < p_2 < p_3 < m - 1 \)). Let \( \gamma \) be the root of \( h(x) \) then

\[ h(\gamma) = \gamma^m + \gamma^{p_3} + \gamma^{p_2} + \gamma^{p_1} + 1 = 0 \implies \gamma^m = \gamma^{p_3} + \gamma^{p_2} + \gamma^{p_1} + 1 \quad (3.6) \]

### 3.3.4 Equally Spaced Polynomial

A polynomial \( i(x) = x^{sm'} + x^{s(m'-1)} + x^{s(m'-2)} + \ldots + x^s + 1 = f(x^s) \) where \( f(x) \) is an AOP of degree \( m' \) is called s-equally spaced polynomial (s-ESP) of degree \( sm' \). The necessary and sufficient conditions for irreducible ESP \( i(x) \) are: (a) The AOP polynomial \( f(x) \) is irreducible.
(b) For some integer \( p \),

\[
s = (m' + 1)^{p-1} \text{ and } \quad 2^{m'(m'+1)^{p-2}} \neq 1 \pmod{(m'+1)^{p}}
\]

Consider an irreducible ESP \( i(x) \) of degree \( m = sm' \) and \( \omega \) be the root of \( i(x) \) then

\[
i(\omega) = \omega^m + \omega^{m-s} + \omega^{m-2s} + ... + \omega^s + 1 = 0 \tag{3.7}
\]

\[
\implies x^m = \omega^{m-s} + \omega^{m-2s} + ... + \omega^s + 1 \tag{3.8}
\]
Chapter 4

Low Register Complexity AOP based Bit Parallel and Bit Serial Systolic Multiplications over $GF(2^m)$ for General Pentanomials

In this section, we use an AOP computation core to propose general pentanomial based multiplication over $GF(2^m)$ (including NIST recommended pentanomials).

4.1 Proposed Bit Parallel-like AOP based Systolic Multiplier

4.1.1 Preliminaries

Let $f(x) = x^m + x^{p_3} + x^{p_2} + x^{p_1} + 1$ (where $m > p_3 > p_2 > p_1 > 1$) be an irreducible pentanomial over finite field $GF(2^m)$. Let $A$, $B$ and $C$ are the elements in this field defined by

$$A = \sum_{i=0}^{m-1} a_i x^i, \ B = \sum_{i=0}^{m-1} b_i x^i \text{ and } C = \sum_{i=0}^{m-1} c_i x^i \quad (4.1)$$
The product of the polynomials $A$ and $B$ is $C$ given by

$$ C = A \cdot B \mod f(x) $$

$$ = A \cdot \left\{ \sum_{i=0}^{m-1} b_i x^i \right\} \mod f(x) $$

$$ = \sum_{i=0}^{m-1} b_i \cdot \left\{ A x^i \mod f(x) \right\} $$

$$ = \sum_{i=0}^{m-1} b_i A^{(i)} $$ \hspace{1cm} (4.3)

where

$$ A^{(i)} = \left\{ \sum_{i=0}^{m-1} a_i x^i \right\} \cdot x^i \mod f(x) $$ \hspace{1cm} (4.4)

Now,

$$ A^{(i+1)} = x \cdot A^{(i)} \mod f(x) $$ \hspace{1cm} (4.5)

Since $f(x) = x^m + x^{p_3} + x^{p_2} + x^{p_1} + 1 = 0$

$$ \Rightarrow x^m = x^{p_3} + x^{p_2} + x^{p_1} + 1 $$

Implies,

$$ a_0^{(i+1)} = a_{m-1}^{(i)} $$ \hspace{1cm} (4.6)

$$ a_j^{(i+1)} = a_j^{(i)} \quad \text{for} \ j \neq p_3 \text{ or } p_2 \text{ or } p_1 $$ \hspace{1cm} (4.7)

$$ = a_{j-1}^{(i)} + a_{m-1}^{(i)} \quad \text{for} \ j = p_3 \text{ or } p_2 \text{ or } p_1 $$ \hspace{1cm} (4.8)

From the generalized equation above it is observed that for each `x` multiplied with $A$, one left circular shift (LCS) and three XOR operations required. The conventional $m$-bit pentanomial multiplier is as shown in Fig. 1.

### 4.1.2 Proposed Architecture

From the above equation, it is obvious that for each time $A$ is multiplied by $x$, where one shift and three XOR operations are required. The XOR operations are mentioned below.
Figure 4.1: Conventional Bit Systolic Pentanomial based Polynomial Multiplier Architecture where (a) Conventional \( m \)-bit Systolic Multiplier. (b) Internal Structure of Each Processing Element (PE). (c) Internal Structure of Shift adder (SA) Cell.

\[
\begin{align*}
    a_{p3}^{(i)} &= a_{p3-1}^{(i-1)} + a_{m-1}^{(i-1)} \\
    a_{p2}^{(i)} &= a_{p2-1}^{(i-1)} + a_{m-1}^{(i-1)} \\
    a_{p1}^{(i)} &= a_{p1-1}^{(i-1)} + a_{m-1}^{(i-1)}
\end{align*}
\]  

(4.9) \hspace{1cm} (4.10) \hspace{1cm} (4.11)

Let us represent the above addition operations in an ordered pair form, i.e.,

\[
\begin{align*}
    U(u_1 (1), u_2 (1)) &= (a_{p3-1}, a_{m-1}) \\
    V(v_1 (1), v_2 (1)) &= (a_{p2-1}, a_{m-1}) \\
    W(w_1 (1), w_2 (1)) &= (a_{p1-1}, a_{m-1})
\end{align*}
\]
where, \( a_{p3}^{(1)} = \sum U(1) = u_1(1) + u_2(1) \)
\( a_{p2}^{(1)} = \sum U(1) = v_1(1) + v_2(1) \)
\( a_{p1}^{(1)} = \sum U(1) = w_1(1) + w_2(1) \)

and

\[
U(u_1(2), u_2(2)) = (a_{p3-2}, a_{m-1})
\]
\[
V(v_1(2), v_2(2)) = (a_{p2-2}, a_{m-1})
\]
\[
W(w_1(2), w_2(2)) = (a_{p1-2}, a_{m-1})
\]

In general,

\[
U(d) = U(u_1(d), u_2(d)) = (a_{p3-d}, a_{m-d}) \quad (4.12)
\]
\[
V(d) = V(v_1(d), v_2(d)) = (a_{p2-d}, a_{m-d}) \quad (4.13)
\]
\[
W(d) = W(w_1(d), w_2(d)) = (a_{p1-d}, a_{m-d}) \quad (4.14)
\]

where \( 1 \leq d \leq k \)

Since the \( a_{p3-\alpha} = a_{p2} \) and \( a_{p3-\gamma} = a_{p1} \), to equate the terms shift set \( V \) by \( \alpha \) bits and shift set \( W \) by \( \gamma \) bits, we pad dummy bits (in this case ‘0’) to the remaining positions. Define \( \alpha = p_3 - p_2, \beta = p_2 - p_1, \) and \( \gamma = p_3 - p_1 \) then

\[
U'(d') = \begin{cases} 
U(1) \text{ to } U(k); & \text{for } 1 \leq d' \leq k \\
(0,0); & \text{for } k < d' \leq k + \gamma 
\end{cases} \quad (4.15)
\]

\[
V'(d') = \begin{cases} 
(0,0); & \text{for } 1 \leq d' \leq \alpha \\
V(1) \text{ to } V(k); & \text{for } \alpha < d' \leq k + \alpha \\
(0,0); & \text{for } k + \alpha < d' \leq k + \gamma 
\end{cases} \quad (4.16)
\]

\[
W'(d') = \begin{cases} 
(0,0); & \text{for } 1 \leq d' \leq \gamma \\
W(1) \text{ to } W(k); & \text{for } \gamma < d' \leq k + \gamma 
\end{cases} \quad (4.17)
\]

Define a set \( E' \) which is the union of the respective terms of \( U', V' \) and \( W' \) such that
\[ E'(d') = \{ u_1(d'), u_2(d') \} \cup \{ v_1(d'), v_2(d') \} \cup \{ w_1(d'), w_2(d') \} \quad (4.18) \]

Now, the terms which involves in addition operation for a bit width \( k \) are:

\[ E(d') = \sum_j E'(d')_j \text{ where } 1 \leq d' \leq k + \gamma \quad (4.19) \]

In this paper, we realize the computational process AOP core in addition to irregular PE core. A broadcasting signal is given to AOP core and a ‘\( k \)’ bit input is given to Irregular PE core. To process those two inputs with much less latency the vector \( E \) is divided into three sub vectors (namely \( X, Y \) and \( Z \)) which can be processed in parallel and hence the improvement in processing speed can be obtained.

The vector \( E'(d') \) (\( 1 \leq d \leq k + \gamma \)) has the maximum number of terms 4, and minimum number of terms 3. The 3\textsuperscript{rd} and 4\textsuperscript{th} terms of each element is realized as vector \( X \) where \( k \) bit length and the sequence of first two terms is divided into vector \( Y \) of \( k - p_1 \) bit length and \( Z \) of \( p_3 \) bit length. The sequence vectors \( X', Y' \) and \( Z' \) are formulated as mentioned below:

\[
X' = \begin{cases}
  U'(x)(u_2) & \alpha + 1 \leq x \leq \gamma \\
  \{ U'(x)(u_2), V'(x)(v_2) \} & \gamma + 1 \leq x \leq k \\
  \{ V'(x)(u_2), W'(x)(w_2) \} & k + 1 \leq x \leq k + \alpha
\end{cases}
\]

\[
Y' = W(y) \quad \text{for } \gamma + 1 \leq y \leq k + \gamma
\]

\[
Z' = \begin{cases}
  U'(z) & 1 \leq z \leq \alpha \\
  V'(z) & \alpha + 1 \leq z \leq \beta \\
  W'(z) & \beta + 1 \leq z \leq \gamma
\end{cases}
\]

Let \( k \) and \( l \) be two integers such that \( \lceil m \rceil = k \cdot l \). Then from the equation 4.3, the total number of vectors formed by \( A^{(i)}(0 \leq i \leq m - 1) \) are ‘\( m \)’. If we represent the total
number of vectors in a matrix form, the matrix is as follows:

\[ M_A = [A^{(0)} A^{(1)} A^{(2)} \ldots A^{(i)} A^{(i+1)} \ldots A^{(m-1)}] \]

The matrix \( M_A \) has \( m \) column vectors. Let us divide the matrix \( M_A \) into \( l \) sub matrices with each matrix having \( k \) column vectors ranging from \( A^{(kn)} \) to \( A^{((k+1)n-1)} \).

\[ M_A^{(n)} = [A^{(kn)} A^{(kn+1)} \ldots A^{((k+1)n-1)}] \quad (0 \leq n \leq l - 1) \]

The column vectors of \( M_A^{(n-1)} \) and the succeeding vector \( A^{(kn)} \) can be formed from the sets \( U, V \) and \( W \) of \( A^{(k(n-1))} \). So the first column vector of \( M_A^{(n)} \) is used to construct the remaining column vectors of \( M_A^{(n)} \) and initial vector of following sub matrix \( M_A^{(n+1)} \). The vector \( A^{((n+1)k)} \) formed from \( A^{nk} \) as follows:

Let \( A_p^{((n+1)k)} \) is a polynomial which is the result of \( E^{(nk)} \) i.e.,

\[
A_p^{((n+1)k)} = \sum_{i=p_1}^{k+p_2+\gamma-1} W^{(n)}(k - i + p_1)x^i \\
+ \sum_{i=p_2}^{k+p_3+\gamma-1} V^{(n)}(k - i + p_2)x^i \\
+ \sum_{i=p_3}^{k+p_3+\gamma-1} U^{(n)}(k - i + p_3)x^i
\]

\[
A^{((n+1)k)} = \sum_{i=0}^{p_1-1} a_{m-k+i}^{nk}x^i + A_p^{((n+1)k)} + \sum_{i=k+p_3}^{m-1} a_{i-k}x^i \quad (4.20)
\]

To reduce the number of addition operations, the set \( E^{(n)} \) is decomposed into three vectors \( X(n), Y(n) \) and \( Z(n) \). Below algorithm depicts the computation of the following vectors.

**Algorithm** Computation of \( X(n), Y(n), Z(n) \)

Input: \( A^{(nk)} \)

Outputs: \( X(n), Y(n), Z(n) \)

1. START
2. \(X(n): j = k - 1\)
   2. for \((i = 1; i \leq \beta; i = i + 1)\) loop
      2. \(X(n)[j] = a_{(n-\alpha-i)}^{(nk)}; j = j - 1;\) end loop
   2. for \((i = \beta + 1; i \leq k - \alpha; i = i + 1)\) loop
      2. \(X(n)[j] = a_{(m-i)}^{(nk)} + a_{(m-\beta-i)}^{(nk)}; j = j - 1;\) end loop
   2. for \((i = 1; i \leq \alpha; i = i + 1)\) loop
      2. \(X(n)[j] = a_{(m-k+\alpha-i)}^{(nk)}; j = j - 1;\) end loop
3. \(Y(n): j = (k - p_1) - 1\)
   3. for \((i = 1; i \leq (k - p_1); i = i + 1)\) loop
      3. \(Y(n)[j] = a_{(m-i)}^{(nk)} + a_{(m-\beta-i)}^{(nk)}; j = j - 1;\) end loop
4. \(Z(n): j = p_3 - 1\)
   4. for \((i = 1; i \leq \alpha; i = i + 1)\) loop
      4. \(Z(n)[j] = a_{(p_3-i)}^{(nk)} + a_{(m-i)}^{(nk)}; j = j - 1;\) end loop
   4. for \((i = 1; i \leq \beta; i = i + 1)\) loop
      4. \(Z(n)[j] = a_{(p_2-i)}^{(nk)} + a_{(m-i)}^{(nk)}; j = j - 1;\) end loop
   4. for \((i = 1; i \leq p_1; i = i + 1)\) loop
      4. \(Z(n)[j] = a_{(p_1-i)}^{(nk)} + a_{(m-i)}^{(nk)}; j = j - 1;\) end loop
5. END

The vectors \(X(n), Y(n)\) and \(Z(n)\) are computed in Pre-Computation module (PC). The computed vectors \(X(n), Y(n)\) along with \(A^{(nk)}\) form broadcasting vector \(P(n)\) of size \(m + k + \gamma - 1\) which will be the input of AOP core. The vector \(X(n)\) will be the input to irregular PE core. The structure and functionalities of AOP Core and irregular PE core are described in the following section.

From 4.1, we can write \(B\) as:

\[
B = \sum_{i=0}^{m-1} b_i x^i = \sum_{n=0}^{l-1} B_n x^{nk}
\]

where

\[
B_n = \sum_{i=nk}^{(n+1)k-1} b_i x^i
\]
Now from equations 4.3 & 4.1.2 we can derive,

\[ C = \sum_{n=0}^{l-1} M_A^{(n)} \cdot B_n^T \]  

(4.21)

Bit parallel architecture contains \( l \) arrays each array consists of PC block, RC block, AOP core and Irr. PE core. PC block computes \( M_A^{(n)} \) which undergoes multiplication with \( B_n \). The detailed bit parallel algorithm is mentioned below.

---

**Algorithm** Bit Parallel multiplication Algorithm.

Inputs: \( A \) and \( B \)

Output: \( C = A \cdot B \mod f(x) \).

1. START
2. \( P(n) = A^{(nk)}, X(n), Y(n), Z(n), C = 0 \)
3. For \( n = 0 \) to \( l - 1 \) loop
4. if \( n = 0 \):
5. \( PC0 \): Compute \( X(0), Y(0), Z(0) \) from \( A^{(0)} \)
6. \( RC0 \): Compute \( \omega(0) \) from \( A^{(0)} \)
7. else:
8. \( PC \): Compute \( X(n), Y(n), Z(n) \) from \( A^{((n-1)k)} \)
9. \( RC \): Compute \( A^{(nk)} \) and \( \omega(n) \) from \( A^{((n-1)k)} \)
10. \( AOP \) Core: Compute \( C_1 = P(n) \ast B_{nk} \mod f(x) \)
11. \( Irr. \) PE Core: Compute \( C_2 = X(n) \ast B_{nk} \mod f(x) \)
12. \( PE_{(k+1)} \): Compute \( C' = C_1 + C_2 \)
13. \( C = C + C' \)
4. END

There are five functions used in Algorithm 1. They are (a) PC0, (b) RC0, (c) PC1, (d) RC1, (e) 2\textsubscript{level}PE. 2\textsubscript{level}PE is a two level systolic array arrangement which consists of (1) AOP core (2) Irregular PE core and (3) Selective-bit-addition \( PE_{(k+1)} \).
The computed $X(n)$, $Y(n)$, $Z(n)$ and $A^{(nk)}$ from previous stage forms two sets of vectors. One vector $P(n)$ is a composition of $Y(n)$, $Z(n)$, and $A^{(nk)}$; and another vector being $X(n)$. From equation 4.21, we can write

$$C_1(n) = \sum_{n=0}^{n=l-1} P(n) \cdot B_n^T$$  \hspace{1cm} (4.22)$$

$$C_2(n) = \sum_{n=0}^{n=l-1} X(n) \cdot B_n^T$$  \hspace{1cm} (4.23)$$

The output of the $n^{th}$ stage of systolic multiplier is given by:

$$C_n = C_1(n) + C_2(n)$$  \hspace{1cm} (4.24)$$

The addition in equation 4.24 refers to addition of selective bits and it is achieved in $PE_{(k+1)}$.

### 4.1.3 AOP Core

Let us consider an irreducible AOP over finite field $GF(2^m)$ $f(x) = x^m + x^{m-1} + x^{m-2} + \ldots + x + 1$ ($x \in GF(2)$) where $m + 1$ is prime and 2 is the primitive modulo $m + 1$.

Define the extended polynomial basis [51] for AOP \{${x^m, x^{m-1}, \ldots, x, 1}$. Let $P$, $Q$, and $R \in GF(2^m)$, they can be represented as:

$$P = \sum_{i=0}^{m} p_i \cdot x^i, Q = \sum_{i=0}^{m} q_i \cdot x^i, R = \sum_{i=0}^{m} r_i \cdot x^i$$

where $p_i, q_i, r_i \in GF(2^m)$, for $0 \leq i \leq m - 1$, and $p_m = 0$, $q_m = 0$, and $r_m = 0$.

Define the product of $P$ and $Q$ is $R$:

$$R = P \cdot Q \mod f(x)$$  \hspace{1cm} (4.25)$$

$$= \sum_{i=0}^{m} q_i \cdot P^{(i)}$$  \hspace{1cm} (4.26)$$

$$= M \cdot Q$$  \hspace{1cm} (4.27)$$
where $M$ is a $(m + 1) \times (m + 1)$ matrix, given by

$$M = [P^{(0)}, P^{(1)}, \ldots, P^{(m)}]$$

$$= \begin{bmatrix}
p_0 & p_m & p_{m-1} & \cdots & \cdots & p_1 \\
p_1 & p_0 & p_m & \cdots & p_2 \\
p_2 & p_1 & p_0 & \cdots & p_3 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
p_m & p_{m-1} & p_{m-2} & \cdots & p_0
\end{bmatrix}$$
The systolic architecture shown in figure has \( m + 1 \) PEs and a broadcast signal \( P \) of \( (m + 1) \) bits. Each processing element \( PE_i \) (\( 0 \leq i \leq m \)) has three inputs: (1) \( (m + 1) \) - bit \( P^{(i)} \), (2) \( (m + 1) \) - bit \( R_{in} \) and (3) 1 - bit \( q_i \). \( PE_i \) consists of \( (m + 1) \) AND gates (inputs to the \( j^{th} \) AND gate: \( q_i \) and \( P^{(i)}(j) \)), \( (m + 1) \) XOR gates (inputs to the \( j^{th} \) XOR gate: \( AND_j \) which is the output of \( j^{th} \) AND gate and \( R_{in}(j) \)) and \( (m + 1) \) D-flip flops.

Proposed architecture requires AOP Core with:
(a) Broadcasting signal \( P_i = A^{(nk)} + X_i + Y_i + Z_i \).
(b) Total number of PE = \( m \). (Each PE has \( m \) AND gates, \( m \) XOR gates and \( m \) D-flip flops).
(c) Inputs to \( PE_i \): (1) \( m - \) bit \( A^{(nk+i)} \) (2) \( m - \) bit \( C_{2(n)m} \) (3) 1 - bit \( b_{(nk+i)} \).
(d) Inputs to \( j^{th} \) AND gate: (1) \( b_{nk+i} \) (2) \( A^{(nk+i)}(j) \).
(e) Inputs to \( j^{th} \) XOR gate: (1) \( C_{2(n)m} \) (2) \( AND_j \).
4.1.4 Irregular PE Core

The second systolic array in the 2-level PE core is irregular PE core. It is a set of serially connected PEs which are not regular like processing elements in AOP core. The architecture of each PE is different hence it is called irregular PE core. The aim of this core is to implement the multiplication of polynomial $B$ with $k$-bit vector $X$ which is obtained from $A$. The algorithm for the implementation of irregular PE is given below:

**Algorithm** Algorithm for irregular PE core.

Inputs: $B, X(n)$
Output: $C_{2(n)}$

1. Define: $IPE(i, j, B, X(n), C_2)$
2. for $(i = 0; i \leq j; i = i + 1)$ loop
   2. $C_2[i] = C_2[i] + B[j + \alpha] \cdot X(n)[k - 1 - j + i]$
   2. end loop
3. START
4. $C_2 = 0; j = 0$
5. for $(i = 0; i < k - 1; i = i + 1)$
   5. return $IPE(i, j, B, X(n), C_2)$
   5. $j = j + 1$
   5. end loop
6. END

4.1.5 Selective Addition Module

The $m$-bit output $C_{1(n)}$ from AOP core added with $k$-bit output of irregular core output $C_{2(n)}$ in respective bits (equation (8)). This operation is proceed in selective-bit adder PE, $PE_{(k+1)}$ described as:
\[ C_n = \begin{cases} 
C_{1(n)}[i]; & 0 \leq i \leq p_2 - 1 \\
C_{1(n)}[i - p_2] + C_{2(n)}[i]; & p_2 \leq i \leq p_2 + k - 1 \\
C_{1(n)}[i]; & p_2 + k \leq i \leq m - 1 
\end{cases} \]

4.1.6 Pre-Computation and Re-Combination Modules

Pre-computation module responsible for computing all possible additive bits resulting from \( A^{(nk)} \) to \( A^{((n+1)k)} \) which are \( X(n), Y(n) \) and \( Z(n) \). And RC module is responsible for computing \( A^{((n+1)k)} \). To avoid minimizing the latency of the architecture, the computations of \( A^{((n+1)k)} \) in RC block is shifted. i.e., the first RC block does not realize \( A^{(nk)} \) and will be realized in following RC block. This implies \( A^{(nk)} \) used in next PC block to compute \( A^{((n+2)k)} \) so there is a shift of \( k \) in the computation of \( X(n), Y(n) \) and \( Z(n) \) \((n > 0)\). This results in the change of \( Z(n) \) computation for \( n > 0 \) such that a new vector \( \omega(n) \) is computed in \( RC_{n-1} \) which involves in \( Z(n) \) computation in \( PC_n \) block.

\[ \omega_n(i) = a_{m-2k+i}^{(n-1)k} + a_{m-2k+i-\alpha}^{(n-1)k}, \quad 0 \leq i < k \]

4.1.7 Example

Consider an irreducible polynomial \( f(x) = x^{163} + x^7 + x^6 + x^3 + 1 \) (where \( m = 163 \), \( p_3 = 7 \), \( p_2 = 6 \), \( p_1 = 3 \)). For \( k = 12 \) the sets of potential additive bits from equations 4.15, 4.16 and 4.17 are as follows:
The above column vector $E$ due to 2 to 4 number of terms in each element, is divided into vectors $X$, $Y$, and $Z$ as shown below.
\[ X = \begin{bmatrix} a_{151} \\ a_{152} + a_{151} \\ a_{153} + a_{152} \\ a_{154} + a_{153} \\ a_{155} + a_{154} \\ a_{156} + a_{155} \\ a_{157} + a_{156} \\ a_{158} + a_{157} \\ a_{159} + a_{158} \\ a_{159} \\ a_{160} \\ a_{161} \end{bmatrix} \quad Y = \begin{bmatrix} a_{154} + a_{151} \\ a_{155} + a_{152} \\ a_{156} + a_{153} \\ a_{157} + a_{154} \\ a_{158} + a_{155} \\ a_{159} + a_{156} \\ a_{160} + a_{157} \\ a_{161} + a_{158} \\ a_{162} + a_{159} \end{bmatrix} \quad Z = \begin{bmatrix} a_{0} + a_{161} \\ a_{1} + a_{161} \\ a_{2} + a_{161} \\ a_{3} + a_{160} \\ a_{4} + a_{161} \\ a_{5} + a_{161} \\ a_{6} + a_{162} \end{bmatrix} \]

4.2 Proposed Bit Serial AOP based Systolic Multiplier

Let the input to block 1 (consisting PC0, RC0) is \( S_{in} \), output of block 1 is \( S_{out} \).
The input to block 2 (consisting PC1,RC1) is \( T_{in} \), output of block 2 is \( T_{out} \).
The inputs to 2\textit{level} PE core is \( P_n \), Q and output is 2\textit{level} PE\textit{out}. \( P_n \) is divided into \( P_{AOP(n)} \) & \( P_{irr(n)} \) and applied to AOP core and irregular PE core respectively. Each bit of \( Q \), \( q_i \) is applied to \( PE_{i(AOP)} \) and \( PE_{i(irr)} \).

In the first clock cycle, output of block 1 is applied to both 2\textit{level} PE core and block 2. From the second clock cycle onwards, the output of block 2 is applied to 2\textit{level} PE and block 2 itself. The algorithm below describes the bit serial systolic multiplication of pentanomials.

\textbf{Algorithm8} Bit Serial Multiplication Algorithm.

Inputs: \( A \) and \( B \)
Output: \( C = A \cdot B \mod f(x) \).

1. START
Figure 4.4: Bit Serial Systolic array Architecture

2. \( S_{in} = A; \ R = 0 \)
3. For \( n = 0 \) to \( l - 1 \) loop
   3. (a) \( Q = B[nk \ to \ (n+1)k - 1] \)
   3. (b) If \( MUX = 0 \) then
      3. (b) (i) \( P_n = S_{out} \)
      3. (b) (ii) \( T_{in} = S_{out} \)
   3. (c) Else
      3. (c) (i) \( P_n = T_{out} \)
      3. (c) (i) \( T_{in} = T_{out} \)
   3. (d) \( 2\text{levelPE}(P_n, Q) \implies 2\text{levelPE}_{out} \)
   3. (e) \( R = R + 2\text{levelPE}_{out} \)
3. end loop
4. \( C = R \)
5. END
Table 4.1: COMPARISON OF AREA-TIME COMPLEXITIES OF BIT-PARALLEL SYSTOLIC MULTIPLIERS

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37]</td>
<td>(m^2)</td>
<td>(m^2 + 2m - 1)</td>
<td>(2m^2 - 2m)</td>
<td>(T_A + T_X)</td>
<td>(mT_p)</td>
</tr>
<tr>
<td>[38]</td>
<td>(2m^2 + 2m)</td>
<td>(m^2 + m)</td>
<td>(3.5m^2 + 3.5m)</td>
<td>(T_A + 2T_X)</td>
<td>((m + 1)T_p)</td>
</tr>
<tr>
<td>[39]</td>
<td>(2m^2)</td>
<td>(2m^2)</td>
<td>(7m^2)</td>
<td>(T_A + T_X)</td>
<td>(2mT_p)</td>
</tr>
<tr>
<td>[39]</td>
<td>(2m^2)</td>
<td>(2m^2)</td>
<td>(7m^2)</td>
<td>(T_A + T_X)</td>
<td>(2mT_p)</td>
</tr>
<tr>
<td>4.2</td>
<td>((km + \sum_{i=0}^{k-1} i - 2 \sum_{i=0}^{p_1} i - p_1)i)</td>
<td>((km + 3(k - \gamma) + \sum_{i=0}^{k-1} i + \alpha)i)</td>
<td>((k + 3)m + 2(k - \gamma) + \sum_{i=0}^{k-1} i + \alpha + 1)i)</td>
<td>(T_A + T_X)</td>
<td>((k + 1) + l + 1)</td>
</tr>
</tbody>
</table>

4.3 Complexity Considerations

A. Complexity of Bit-Parallel-Like Systolic Architecture:

Proposed bit-parallel-like systolic architecture has \(l\) systolic arrays, where each array has \(k\) serially connected 2-level PEs. 2 level PE core contains three components. (1) AOP core has \(km\) AND gates, \((k - 1)m\) XOR gates and \(km\) flip flops. Irregular PE core requires \(\sum_{i=0}^{k-a-1} i\) AND gates. Since some bits of irregular PE core shared with AOP core, we can reduce the number of AND gates in irregular PE core to \(\sum_{i=0}^{k-a-1} i - [2 \cdot \sum_{i=0}^{p_1} i + p_1]\). Irregular PE core requires \(\sum_{i=0}^{k-2} i - [3(\gamma) - 2]\) XOR gates and \(\sum_{i=0}^{k-1} i - [3(\gamma) - 2]\) flip flops. (3) \(PE_{(k+1)}\) requires \((k - 1)\) XOR gates and \(m\) flip flops.

Each PC module requires \((2k - 1)\) XOR gates and \((2k + \gamma - 1)\) flip flops where as RC requires \(\alpha\) XOR gates and \((\alpha + m)\) flip flops. For the addition of the each array’s \((l_i)\) output with previous array \((l_{i-1})\) output, it requires \(m\) XOR gates and \(m\) flip flops.

Proposed structure yields output in \([k + 1] + l + 1\) clock cycles where critical path is \(T_A + T_X\) \((T_A: AND gate delay, T_X: XOR gate delay)\).

B. Complexity of Bit-Serial Systolic Architecture:

Proposed bit serial systolic architecture has one systolic array, two PCs, two RCs and one accumulator. Accumulator requires \(m\) XOR gates and \(m\) flip flops. The requirement of each PC and RC is same as the proposed bit parallel architecture and also there is no change in 2-level PE core. It has a latency of \([k + 1] + l + 1\] with critical path \(T_A + T_X\).

C. Comparison of Area and time complexities:
Table 4.2: COMPARISON OF AREA-TIME COMPLEXITIES OF SERIAL SYSTOLIC MULTIPLIERS

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[54]</td>
<td>$md + 2k_d - k_l + 2d - 1$</td>
<td>$md + 2k_d - k_l + d - 1$</td>
<td>$2m + d + k_l$</td>
<td>$T_A + (\log_2(d+1))T_c$</td>
<td>$\lceil \log_2(d+1) \rceil T_A$</td>
</tr>
<tr>
<td>[41]</td>
<td>$2md + m$</td>
<td>$2md$</td>
<td>$<a href="10d+1+9ed/2">m/d</a> + s$</td>
<td>$d(T_A + T_X + T_{MUX}/(s+1))$</td>
<td>$3m/dT_c$</td>
</tr>
<tr>
<td>4.4</td>
<td>$km + \sum_{i=0}^{\alpha-1} i - 2\sum_{i=0}^{\alpha-1} i - \alpha$</td>
<td>$km + \sum_{i=0}^{\alpha-1} i - \alpha - \gamma$</td>
<td>$(k+4)m + 4k + \sum_{i=0}^{\alpha-1} i + \alpha - \beta$</td>
<td>$T_A + T_X$</td>
<td>$(k+1) + l + 1$</td>
</tr>
</tbody>
</table>

Table 4.3: $x^{163} + x^7 + x^6 + x^3 + 1$

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[57]</td>
<td>20569</td>
<td>26244</td>
<td>52812</td>
<td>$T_A + T_X$</td>
<td>$mT_c$</td>
</tr>
<tr>
<td>[53]</td>
<td>51438</td>
<td>51438</td>
<td>185983</td>
<td>$T_A + T_X$</td>
<td>$2mT_c$</td>
</tr>
<tr>
<td>[39]</td>
<td>51343</td>
<td>51343</td>
<td>79707</td>
<td>$T_A + T_X$</td>
<td>$(m+1)T_c$</td>
</tr>
<tr>
<td>[38]</td>
<td>51343</td>
<td>51343</td>
<td>79707</td>
<td>$T_A + T_X$</td>
<td>$(m+1)T_c$</td>
</tr>
<tr>
<td>[55]</td>
<td>26569</td>
<td>32425</td>
<td>53138</td>
<td>$T_A + T_X$</td>
<td>$\sqrt{m + \log_2 m}$</td>
</tr>
<tr>
<td>[55]</td>
<td>26569</td>
<td>32425</td>
<td>53138</td>
<td>$T_A + T_X$</td>
<td>$\sqrt{m + \log_2 m}$</td>
</tr>
<tr>
<td>4.2</td>
<td>28028</td>
<td>28490</td>
<td>35406</td>
<td>$T_A + T_X$</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 4.4: $x^{283} + x^{12} + x^7 + x^5 + 1$

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[57]</td>
<td>80089</td>
<td>80654</td>
<td>159612</td>
<td>$T_A + T_X$</td>
<td>$mT_c$</td>
</tr>
<tr>
<td>[53]</td>
<td>160744</td>
<td>80372</td>
<td>281302</td>
<td>$T_A + 2T_X$</td>
<td>$(m+1)T_c$</td>
</tr>
<tr>
<td>[39]</td>
<td>160744</td>
<td>160178</td>
<td>560623</td>
<td>$T_A + T_X$</td>
<td>$2mT_c$</td>
</tr>
<tr>
<td>[38]</td>
<td>160744</td>
<td>160178</td>
<td>240267</td>
<td>$T_A + T_X$</td>
<td>$(m+1)T_c$</td>
</tr>
<tr>
<td>4.2</td>
<td>82484</td>
<td>84405</td>
<td>98974</td>
<td>$T_A + T_X$</td>
<td>36</td>
</tr>
</tbody>
</table>

Table 4.5: $x^{571} + x^{10} + x^5 + x^2 + 1$

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[57]</td>
<td>326041</td>
<td>327183</td>
<td>650940</td>
<td>$T_A + T_X$</td>
<td>$mT_c$</td>
</tr>
<tr>
<td>[53]</td>
<td>653224</td>
<td>326612</td>
<td>11143142</td>
<td>$T_A + 2T_X$</td>
<td>$(m+1)T_c$</td>
</tr>
<tr>
<td>[39]</td>
<td>653224</td>
<td>653208</td>
<td>2282287</td>
<td>$T_A + T_X$</td>
<td>$2mT_c$</td>
</tr>
<tr>
<td>[38]</td>
<td>653224</td>
<td>653208</td>
<td>978123</td>
<td>$T_A + T_X$</td>
<td>$(m+1)T_c$</td>
</tr>
<tr>
<td>4.2</td>
<td>332904</td>
<td>336216</td>
<td>336432</td>
<td>$T_A + T_X$</td>
<td>50</td>
</tr>
</tbody>
</table>
### Table 4.6: FPGA IMPLEMENTATION RESULTS OF BIT PARALLEL PENTANOMIALS FOR VARIOUS DEGREE $m$

<table>
<thead>
<tr>
<th>Polynomial($m$)</th>
<th>Area (#reg.)</th>
<th>Delay ($ns$)</th>
<th>Power (mW)</th>
<th>ADP</th>
<th>PDP ($pW$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>35406</td>
<td>1.695</td>
<td>34.157</td>
<td>60013.17</td>
<td>57.9</td>
</tr>
<tr>
<td>283</td>
<td>98974</td>
<td>1.70</td>
<td>63.649</td>
<td>168255.8</td>
<td>108.2</td>
</tr>
<tr>
<td>571</td>
<td>336432</td>
<td>1.732</td>
<td>136.10</td>
<td>582027.4</td>
<td>235.725</td>
</tr>
</tbody>
</table>

Unit for area: number of registers; Unit for delay: $ns$; Unit for power: W (Power is estimated at 100MHz).

1: Delay = Critical-Path in $ns$.

2: ADP: Area-delay product = Area $\times$ Delay in #registers $\cdot$ delay($ns$).


### Table 4.7: SERIAL SYSTOLIC MULTIPLIERS: $m = 163$

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[54]</td>
<td>$md + 2k_1d - k_1 + 2d - 1$</td>
<td>$md + 2k_1d - k_1 + d - 1$</td>
<td>$2m + d + k_1$</td>
<td>$T_A + \lceil \log_2(d + 1) \rceil T_x$</td>
<td>$3[m/d]T_{cp}$</td>
</tr>
<tr>
<td>[41]</td>
<td>4727</td>
<td>4564</td>
<td>2400</td>
<td>$d(T_A + T_X + T_{MUX}/(s + 1))$</td>
<td>$3[m/d]T_{cp}$</td>
</tr>
<tr>
<td>4.4</td>
<td>2002</td>
<td>2065</td>
<td>2720</td>
<td>$T_A + T_X$</td>
<td>28</td>
</tr>
</tbody>
</table>

### Table 4.8: SERIAL SYSTOLIC MULTIPLIERS: $m = 283$

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[54]</td>
<td>$md + 2k_1d - k_1 + 2d - 1$</td>
<td>$md + 2k_1d - k_1 + d - 1$</td>
<td>$2m + d + k_1$</td>
<td>$T_A + \lceil \log_2(d + 1) \rceil T_x$</td>
<td>$3[m/d]T_{cp}$</td>
</tr>
<tr>
<td>[41]</td>
<td>9905</td>
<td>9622</td>
<td>4301</td>
<td>$d(T_A + T_X + T_{MUX}/(s + 1))$</td>
<td>$3[m/d]T_{cp}$</td>
</tr>
<tr>
<td>4.4</td>
<td>4852</td>
<td>5004</td>
<td>6150</td>
<td>$T_A + T_X$</td>
<td>36</td>
</tr>
</tbody>
</table>

### Table 4.9: SERIAL SYSTOLIC MULTIPLIERS: $m = 571$

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>Critical-path delay</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[54]</td>
<td>$md + 2k_1d - k_1 + 2d - 1$</td>
<td>$md + 2k_1d - k_1 + d - 1$</td>
<td>$2m + d + k_1$</td>
<td>$T_A + \lceil \log_2(d + 1) \rceil T_x$</td>
<td>$3[m/d]T_{cp}$</td>
</tr>
<tr>
<td>[41]</td>
<td>27979</td>
<td>27408</td>
<td>8400</td>
<td>$d(T_A + T_X + T_{MUX}/(s + 1))$</td>
<td>$3[m/d]T_{cp}$</td>
</tr>
<tr>
<td>4.4</td>
<td>13871</td>
<td>14063</td>
<td>16362</td>
<td>$T_A + T_X$</td>
<td>50</td>
</tr>
</tbody>
</table>

### Table 4.10: FPGA IMPLEMENTATION RESULTS OF BIT SERIAL PENTANOMIALS FOR VARIOUS DEGREE $m$

<table>
<thead>
<tr>
<th>Polynomial($m$)</th>
<th>Area (#reg.)</th>
<th>Delay ($ns$)</th>
<th>ADP (#reg-ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>2720</td>
<td>2.07</td>
<td>5630.4</td>
</tr>
<tr>
<td>283</td>
<td>6150</td>
<td>2.07</td>
<td>10660.5</td>
</tr>
<tr>
<td>571</td>
<td>16362</td>
<td>2.11</td>
<td>34523.8</td>
</tr>
</tbody>
</table>
Chapter 5

Addition and Multiplication of Polynomials over $GF(2^m)$

In this chapter, we propose a hybrid architecture which can compute polynomial addition and polynomial multiplication for four different irreducible polynomials.

5.1 Polynomial Reduction to $GF(2^m)$ from $GF(2^{2m})$

Define a polynomial basis $B_{ext} = \{\delta^{2m-1}, \delta^{2m-2}, ..., \delta^2, \delta, 1\}$ over $GF(2^{2m})$ and consider a polynomial $P$ in $B_{ext}$ such that

$$P = \sum_{i=0}^{2m-1} p_i \cdot x^i \quad (5.1)$$

Now let us reduce polynomial $P$ to finite field $GF(2^m)$ using three irreducible polynomials discussed above.

5.1.1 All One Polynomial

Let us consider an AOP based polynomial basis $\{1 + \delta + \delta^2 + ... + \delta^{2m} + \delta^{2m+1} + \delta^{2m+2}\}$

Now From 3.2 and 3.3 we can write
\[ P(0) = p_0 \]
\[ P(1) = p_1 x^1 \]

\[ P(m) = p_m x^m \]
\[ P(m + 1) = p_{m+1} x^{m+1} = p_{m+1} \]
\[ P(m + 2) = p_{m+2} x^{m+2} = p_{m+2} \cdot x \]

\[ P(2m + 1) = p_{2m+1} x^{2m+1} = p_{2m+1} \cdot x^m \]

i.e.,

\[ \delta^i \implies \delta^i \text{ for } 0 \leq i \leq m \]
\[ \delta^i \implies \delta^{i-m+1} \text{ for } m + 1 \leq i \leq 2m + 1 \]

The GF(2^{(m+1)}) polynomial basis is reduced to GF(2^{m+1}) as

\[ \alpha^i = \delta^i + \delta^{i+m+1} \quad (0 \leq i \leq m) \quad (5.2) \]

Now the polynomial \( P \) can be reduced to \( m+1 \) degree polynomial (where \( p_m = 0 \)) and is given below

\[
\begin{bmatrix}
    p_0 + p_{m+1} \\
    p_1 + p_{m+2} \\
    p_2 + p_{m+3} \\
    \cdot \\
    p_{m-2} + p_{2m-1} \\
    p_{m-1} \\
    p_m
\end{bmatrix}
\begin{bmatrix}
    p_0 \\
    p_1 \\
    p_2 \\
    \cdot \\
    p_{m-1} \\
    p_m \\
    p_{m+1}
\end{bmatrix}
\]

\[ P_{\text{mod}} = P \mod f(x) = P_{\text{mod}} + \cdot \]

\[ \begin{bmatrix}
    p_0 \\
    p_1 \\
    p_2 \\
    \cdot \\
    p_{m-1} \\
    p_m \\
    p_{m+1}
\end{bmatrix}
\]

32
$= M_1 + M_2$ \hfill (5.3)

The given polynomial is divided into two vectors, vector $M_1$ which consists of elements from $p_0$ to $p_m$ and vector $M_2$ with $p_{m+1}$ to $p_{2m+1}$. The addition of these two vectors lead to the $m$-degree reduction of the $2m$-degree polynomial $P$ (Note that $p_m = p_{2m} = p_{2m+1} = 0$).

$M_1 = [p_0, p_1, ..., p_m]^T$

$M_2 = [p_{m+1}, p_{m+2}, ..., p_{2m-1}, p_{2m}, p_{2m+1}]^T$

### 5.1.2 Trinomial

The $m^{th}$ term of polynomial $P(m) = p_m \cdot x^m$

From equation(3.5),

$P(0) = p_0$

$P(1) = p_1x$

$\cdot$

$\cdot$

$\cdot$

$P(m) = p_mx^m = p_m \{1 + x^n\}$

$P(m + 1) = p_{m+1}x^{m+1} = p_{m+1} \{x + x^{n+1}\}$

$\cdot$

$\cdot$

$P(2m - 1) = p_{2m-1}x^{2m-1} = p_{2m-1} \{x^{m-1} + x^{n-1}\}$

i.e.,

$\delta^i \implies \delta^i 0 \leq i \leq m - 1$

$\delta^i \implies \delta^{i-m} + \delta^{i-m+n} m \leq i \leq 2m - 1$

33
The reduced polynomial basis is given by

$$\beta^i = \delta^i + \delta^{m+i} + \delta^{2m-n+i} (0 \leq i \leq m - 1) \quad (5.4)$$

Now the reduction of polynomial can be achieved as

$$P_t = P \mod g(x) = \begin{bmatrix} p_0 & p_m \end{bmatrix} + \begin{bmatrix} p_1 & p_{m+1} \end{bmatrix} + \begin{bmatrix} p_2 & \cdots \end{bmatrix} + \begin{bmatrix} \cdots \end{bmatrix} + \begin{bmatrix} p_{m-n} \end{bmatrix} + \begin{bmatrix} p_{m} \end{bmatrix} \quad (5.5)$$

The reduced $m$-degree polynomial $P$ is a combination of three vectors $M_1$, $M_2$ and $M_t$ where vectors $M_1$ and $M_2$ are same as the vectors for AOP based polynomial. It is observed that the vector $M_t$ is the $m$-bit shifted version of vector $M_2$.

### 5.1.3 Pentanomial

The polynomial $P$ reduced to $Degree - m$ pentanomial based polynomial basis as follows:

The $m^{th}$ term of polynomial $P(m) = p_m \cdot x^m$

From equation(3.6)

$$P(m) = p_m \cdot \{x^{p_3} + x^{p_2} + x^{p_1} + 1\}$$

$$= p_m + p_m x^{p_3} + p_m x^{p_2} + p_m x^{p_1} + 1$$
This implies

\[
\delta^i \implies \delta^i \quad 0 \leq i \leq m - 1
\]
\[
\delta^i \implies \delta^{i-m} + \delta^{i-m+p_1} + \delta^{i-m+p_2} + \delta^{i-m+p_3} \\
\quad m \leq i \leq 2m - 1
\]

The reduced polynomial basis is given by

\[
\alpha^i = \delta^i + \delta^{m+i} + \delta^{2m-p_1+i} + \delta^{2m-p_2+i} + \delta^{2m-p_3+i}
\] (5.6)

Now, the polynomial reduction modulo irreducible pentanomial is given as:

\[
P_p = P \mod h(x) = M_1 + M_2 + M_{p_3} + M_{p_2} + M_{p_1}
\]

where, \(M_{p_i}\) is \(p_i\) bit circular shift of \(M_2\).

If we combine column vector matrices \(M_{p_i}\) to form a single column vector matrix \(M_p\) such that,

\[
M_p = M_{p_3} + M_{p_2} + M_{p_1}
\]

Therefore,

\[
P_p = M_1 + M_2 + M_p
\] (5.7)

### 5.1.4 Equally Spaced Polynomial

The necessary and sufficient conditions for the ESP to be an irreducible polynomial is mentioned in introduction. The modular reduction of polynomial \(P\) with respect to degree—\(m\) irreducible ESP is given by:

\[
\delta^i \implies \delta^i \quad 0 \leq i \leq m - 1
\]
\[
\delta^i \implies \delta^{i-m} + \delta^{i-m+s} + \delta^{i-m+2s} + \ldots + \delta^{i-m+(m'-1)s} \\
\quad m \leq i \leq 2m - 1
\]

where, \(m' = \lceil m/s \rceil\).
Now,

\[ P_e = P \mod i(x) \]

\[ = M_1 + M_2 + \sum_{i=1}^{m'-1} M_{is} \]

where, \( M_{is} \) is the \( is \)th bit circular shift of \( M_2 \).

If we combine the column vector matrices \( M_{is} \) to form a new column vector matrix such that,

\[ M_e = \sum_{i=1}^{m'-1} M_{is} \]

Therefore,

\[ P_e = M_1 + M_2 + M_e \quad (5.8) \]

### 5.2 Polynomial Multiplication

Consider three polynomials \( A, B, \) and \( C \) over \( GF(2^m) \) where \( C \) is the product of \( A \) and \( B \), i.e.,

\[ C = A \cdot B \mod f(x) \quad (5.9) \]

The multiplication process (5.9) is divided into two parts. (1) Formation of extended
basis $GF(2^{2m-1})$ (2) Reduction of $GF(2^{2m-1})$ to $GF(2^m)$.

5.2.1 Formation of Extended Basis $GF(2^{2m-1})$

$$C = \{a_0 + a_1x + ... + a_{m-1}x^{m-1}\}.$$ (5.10)

$$C = \sum_{q=0}^{m-1} x^q \left\{ \sum_{i=0}^{q} a_i b_{q-i} \right\} + \sum_{q=m}^{2(m-1)} x^q \left\{ \sum_{i=q-m+1}^{m-1} a_i b_{m-1} \right\}$$ (5.11)

It is observed that basis for the resultant of multiplication is extended to $2m - 1$. Since the result of multiplication should lie in $GF(2^m)$, the next step is the conversion of $(2m - 1)$-degree polynomial into $m$-degree polynomial.

5.2.2 Reduction of $GF(2^{2m-1})$ to $GF(2^m)$

The resultant polynomial $C$ before applying modular reduction operation has a polynomial basis $\{x^{2m-2}, x^{2m-1}, ..., x, 1\}$. Modular reduction with irreducible polynomial $f(x)$ of degree $- m$ results in the product of $A$ and $B$ polynomials over a finite field $GF(2^m)$. $M_1$ and $M_2$ are common matrices in the equations (5.3), (5.5) and (5.7). equation(5.5) has an additional matrix $M_t$, equation(5.7) has an additional matrix $M_p$ and equation (5.8) has an additional matrix $M_e$. In our architecture, we first realize the sum of $M_1$ and $M_2$ and then select the proper matrix computation to realize reduction in terms of respective irreducible polynomial.

Computation of $M_1 + M_2$

From the equation(5.10), let us consider a vector $C^1 = AB \mod f(x)$ such that $C^1 = M_1 + M_2$. Let $a_i x^i \cdot b_j y^j = a_i b_j x^k$, then

For matrix $M_1$, $0 \leq k \leq m - 1$ for $c_i^1$ to $c_{m-1}^1$ and

For matrix $M_2$, $m \leq k \leq 2m - 1$ for $c_0^1$ to $c_{m-1}^1$

The elements of $C^1(c_0^1, c_1^1, ..., c_{m-1}^1)$ can be differentiated into three vectors:
\[ c_0^1 = a_0 \cdot b_0 \]
\[ + a_{m-1} \cdot b_1 + a_{m-2} \cdot b_2 + a_{m-3} \cdot b_3 + .... + a_1 \cdot b_{m-1} \]
\[ c_1^1 = a_0 \cdot b_1 \]
\[ + a_1 \cdot b_0 \]
\[ + a_{m-1} \cdot b_2 + a_{m-2} \cdot b_3 + .... + a_2 \cdot b_{m-1} \]
\[ c_2^1 = a_0 \cdot b_2 \]
\[ + a_2 \cdot b_0 + a_1 \cdot b_1 \]
\[ + a_{m-1} \cdot b_3 + a_{m-2} \cdot b_4 + .... + a_3 \cdot b_{m-1} \]

In general,

\[ c_k^1 = a_0 \cdot b_k \quad (5.12) \]
\[ = \sum_{i=1}^{k} a_i \cdot b_{k-i} \quad (5.13) \]
\[ = \sum_{i=k+1}^{m-1} a_i \cdot b_{m-1-(i-k)} \quad (\text{where} \, 0 \leq k \leq m-1) \quad (5.14) \]

From the above equations 5.12,5.13,5.14; for every \( k(0 \leq m - 1) \) three matrices \( T, U, V \) have been defined:

\[
T = \begin{bmatrix}
    a_0b_0 \\
    a_0b_1 \\
    a_0b_2 \\
    . \\
    . \\
    . \\
    a_0 \cdot b_{m-1}
\end{bmatrix}
\]

Matrix \( U \) :
It is an \((m - 1) \times (m - 1)\) lower triangular matrix.

\[
U = \begin{bmatrix}
    a_1 b_0 & 0 & 0 & 0 & \cdots & 0 \\
    a_2 b_0 & a_1 b_1 & 0 & 0 & \cdots & 0 \\
    a_3 b_0 & a_2 b_1 & a_1 b_2 & 0 & \cdots & 0 \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
    a_{m-1} b_0 & a_{m-2} b_1 & \cdots & \cdots & a_1 b_{m-1} \\
\end{bmatrix}
\]

Matrix \(V\):

It is an \((m - 1) \times (m - 1)\) upper triangular matrix.

\[
V = \begin{bmatrix}
    a_{m-1} b_1 & a_{m-2} b_2 & \cdots & a_2 b_{m-2} & a_1 b_{m-1} \\
    0 & a_{m-1} b_2 & \cdots & a_3 b_{m-2} & a_2 b_{m-1} \\
    0 & 0 & \cdots & a_4 b_{m-2} & a_3 b_{m-1} \\
    \vdots & \vdots & \vdots & \vdots & \ddots \\
    \vdots & \vdots & \cdots & a_{m-2} b_{m-2} & a_{m-3} b_{m-1} \\
    0 & 0 & 0 & \cdots & a_{m-1} b_{m-2} & a_{m-2} b_{m-1} \\
\end{bmatrix}
\]

From the matrices \(T, U\) and \(V\) it is clear that matrix \(M_1\) is the combination of matrices \(T \& U\) and \(M_2\) composed from the matrix \(V\).

\[
c_0^1 = T[0] + V[0] \\
c_k^1 = T[k] + U[k - 1] + V[k]; 1 \leq k \leq m - 2 \tag{5.15}
\]

\[
c_{m-1}^1 = T[m - 1] + U[m - 2]
\]

**Computation of \(M_t\), \(M_p\) and \(M_e\)**

Note that Matrix \(M_t\) is the result of \(n\)-bit circular shift of matrix \(M_2\). Similarly \(M_p\) is the summation of \(p_1\)-bit, \(p_2\)-bit and \(p_3\)-bit shifts of matrix \(M_2\). Hence it is evident that \(M_t\) and \(M_p\) can be computed from \(M_2\) (or) matrix \(V\).
Since the matrices $M_t, M_p, M_e$ are the circular shift results of $M_2$, these matrices can be realized from the matrix $V$. Hence the multiplication of two degree $-m$ polynomials $A, B$ over irreducible polynomial $i$ (AOP/trinomial/pentanomial/ESP) is given by,

$$C = C^1 + M_i; \quad i \in \{t, p, e\}$$

(5.16)

### 5.3 Novel Obfuscation Model

#### 5.3.1 Addition

Polynomial addition over $GF(2^m)$ is achieved by bit wise linear addition over $GF(2)$. Since polynomial coefficients defined over Finite Field $GF(2)$, the addition is nothing but bit XOR with $T_{xor}$ being the time delay. But in our obfuscated model, addition is processed in two clock cycles with delay being $T_{and} + T_{xor}$. There is not much difference between these two mentioned delays with respect to critical path. This is because each cycle in ECC implementation supposed to have delay greater than or equal to $T_{and} + T_{xor}$.

It is noticed that all the realizations in literature have critical path greater than or equal to $T_{and} + T_{xor}$. Since ECC implementation requires many additions to be performed, some of the addition operations perform in the proposed architecture. The correct triggering sequence enables the addition operation. Failing of giving the correct triggering sequence leads to the incorrect output data.

---

**Algorithm** Polynomial Addition.

---

Inputs: $A$ and $B$

Output: $C = A + B \mod f(x)$.

1. START
2. $C = 0$
3. if ($reset = 0$): $C = C + A$
4. else: $C = C + B$
5. END
5.3.2 Multiplication

From the equations (5.15) and (5.16), the polynomial multiplication is performed in such a way that, first the column vector \( T \) is realized with \( a_0 \cdot B \). We modify the input vector \( A_{in} \) which is input \( A \) but the \( x^0 \) coefficient of \( A_{in} \) is \( x^1 \) coefficient. i.e., \( A_{in} = "A(N - 1 : 1) \ & A(1)" \). The \( reset \) is at logic ‘1’ triggers rotation operation in \( A \)-rotation block \((fig.(5.5))\). The input \( A_{in} \) undergoes successive rotations for each clock when \( reset \) is ‘1’ and the result of rotation gets multiply with \( B_{in} \) in multiplication block. The input \( B_{in} \) is same as \( B \) and these inputs \( A_{in} \) and \( B_{in} \) processed to compute matrices \( U \) and \( V \). The computation of upper triangular and lower triangular matrices is achieved in selective buffer block \((fig.(5.3))\) where each row vector of \( V \) and \( U \) is computed in every clock cycle and sent to adder tree block \((fig.(5.3))\) to get the summation i.e., the result \( V_i \) and \( U_i \) \((0 \leq i \leq m - 2)\). The outputs from the two adder tree blocks form a vector \( UV \) to realize the equation (5.15) and the vector will be:

\[
UV(0) = \sum V_i \\
UV(1) = \sum U_i \\
UV(i) = \sum V_i \text{ for } 2 \leq i \leq m - 1
\]

In the first clock cycle, for \( reset \) is at logic ‘0’, the two inputs \( a_0 \) and \( B \) are given to post multiplication block. Let us take two inputs of post multiplication block are \( pm_1 \) & \( pm_2 \). The input \( a_0 \) is given to \( pm_2 \) and \( B \) is given to \( pm_1 \) through multiplexer. This multiplication at \( reset = '0' \) forms the column vector \( T \). For \( reset \) is at logic ‘1’, the irreducible polynomial vector \( z(x) \) is input to \( pm_2 \) and the outputs of adder tree blocks will be given to \( pm_1 \) through multiplexer. This product gives rise to the multiplications \( U \cdot B \) and \( V \cdot B \) and the circular shift column vector \( M_i \cdot B \).

\[
\begin{align*}
\text{pm}_1 &= B \text{ for } reset = 0 \\
\text{pm}_1 &= UV \text{ for } reset = 1
\end{align*}
\]

In \( C \)-rotation block \((fig.(5.5))\), the input from controller \( C_{in} \) is a zero vector. For \( reset \) at logic ‘0’, one input to the XOR gate is the output of post multiplication block and the
other input is the output of the multiplexer. For \( reset = 0 \), output of multiplexer is \( C_{in} \) and for \( reset = 1 \), output of multiplexer is the output of successor XOR gate triggering rotation.

The irreducible polynomial vector \( z(x) \) is a primitive vector in proposed polynomial multiplication. This vector input to post multiplication block and responsible for the multiplication of vector \( M_i \), \( i \in \{t, p, e\} \). The first two vector elements are logic ‘1’s so that it will enable the computation of \( V(i) \) and \( U(i) \). The remaining elements being logic ‘0’ except for respective irreducible polynomial terms. i.e., for trinomial, \( n^{th} \) element of \( z(x) \) will be one, for pentanomial \( p_3^{th}, p_2^{th}, p_1^{th} \) elements will be one. Vector \( z(x) \) for different irreducible polynomial vectors are listed below:

\[
z_1(x)_i = 1 \quad \text{for } i \in \{0, 1\} \\
z_1(x)_i = 0 \quad \text{other}
\]

For Trinomial:

\[
z_2(x)_i = 1 \quad \text{for } i \in \{0, 1, n\} \\
z_2(x)_i = 0 \quad \text{other}
\]

For Pentanomial:

\[
z_3(x)_i = 1 \quad \text{for } i \in \{0, 1, p_1, p_2, p_3\} \\
z_3(x)_i = 0 \quad \text{other}
\]

For ESP:

\[
z_4(x)_i = 1 \quad \text{for } i \in \{0, 1, s, 2s, ..., (m' - 1)s\} \\
z_4(x)_i = 0 \quad \text{other}
\]

In the obfuscation model, the vector \( z(x) \) has a total of 8 vectors from \( z_0(x) \) to \( z_7(x) \) among those one zero vector \( z_0(x) = 0 \) which is input \( z(x) \) to post multiplication block for addition and as input to \( C_{in} \) for multiplication. Only four vectors are correct for multiplication, will come to use and other vectors stored are false vectors. Selecting
Table 5.1: OBFUSCATED MODEL INPUT MODES

<table>
<thead>
<tr>
<th>Triggering Sequence</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Addition</td>
</tr>
<tr>
<td>001</td>
<td>AOP multiplication</td>
</tr>
<tr>
<td>010</td>
<td>Trinomial multiplication</td>
</tr>
<tr>
<td>011</td>
<td>Pentanomial multiplication</td>
</tr>
<tr>
<td>100</td>
<td>ESP multiplication</td>
</tr>
<tr>
<td>101</td>
<td>Incorrect output</td>
</tr>
<tr>
<td>110</td>
<td>Incorrect output</td>
</tr>
<tr>
<td>111</td>
<td>Incorrect output</td>
</tr>
</tbody>
</table>

Table 5.2: CONTROLLER OUTPUT SEQUENCES

<table>
<thead>
<tr>
<th>mode</th>
<th>reset = 0</th>
<th>reset = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$z(x)$</td>
<td>$C_{in}$</td>
</tr>
<tr>
<td>000</td>
<td>$z_0(x)$</td>
<td>$A$</td>
</tr>
<tr>
<td>001</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
<tr>
<td>010</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
<tr>
<td>011</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
<tr>
<td>100</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
<tr>
<td>101</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
<tr>
<td>110</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
<tr>
<td>111</td>
<td>$a_0$</td>
<td>$z_0(x)$</td>
</tr>
</tbody>
</table>

false vectors leads to incorrect output despite correct execution of architecture. All these vectors stored in ROM and accessed with correct input sequence. This address sequence is given from controller and the ROM output is given to post multiplication block/C-rotate block. The obfuscation model for possible input states are shown in table 5.1. From the table, for different input modes the modes of operation are different and it is clear that, the irreducible polynomial vectors $z_5(x)$ to $z_6(x)$ stored with incorrect sequences which lead to false outputs. Besides irreducible polynomial entries, the correct reset signal is required to ensure desire functionality of circuit. The change in reset alters the post multiplication block input ($z(x)$) and C-rotate block input ($C_{in}$). The controller outputs for different mode inputs and reset values is given in table 5.2.
Algorithm Polynomial Multiplication.

Inputs: $A$ and $B$

Output: $C = A \cdot B \mod f(x)$.

1. START
2. $C = 0$
3. if ($reset = 0$): $C = C + T$
4. else:
5. for $i = 1$ to $m-1$ loop
6. $A_{in}(i) = rot^1(A_{in}(i-1))$
7. $P = A_{in}(i) \cdot B$
8. $U, V = sel. \ buf^2 (P)$
9. $R = UV \cdot Z$
10. $C = rot(C)$
11. $C = C + R$ end loop
5. END

$^1 =$ Rotation; $^2 =$ Selective buffer.
The whole architecture is designed to compute the three matrix elements namely $U,V$ and $T$. Column vector matrix $T$ can be calculated in one time instance and computed in ost multiplication block. Matrices $U$ and $V$ had $(m - 1)$ row vectors so at every clock cycle, one row vector of $U$ and $V$ computed. For example at $i^{th}$ time instance $U(i)$ and $V(i)$ computed. The architecture is shown in fig(5.1). The architecture is composed of
seven hardware blocks and one controller block each block has different structure and functionality.

5.4 Hardware and Complexity

The obfuscated architecture with low area, optimal critical path (throughput) with desired latency is obtained. A controller block is used to initiate input sequences for different mode types. A reset signal is used which propagates from $A$-rotate block to $C$-rotate block. $A$-rotate block has $m - 2x1$ multiplexers and $m$ D flip flops. Multiplication block consists of $m$-AND gates and $m$ D flipflops. Selective buffer block has $3m$ - D flip flops with the time delay of two XOR gates, adder tree block contains \( \sum_{i=1}^{L} m/(4^i) \) (where \( L = \log_2 m/2 \)) flip flops and \( \sum_{i=1}^{2L} m/i \) XOR gates. Post multiplication block consists of $m - 2x1$ multiplexers, $m$ - AND gates and $m$ - D flip flops with a time delay of \( T_{\text{mux}} + T_A \). $C$-rotate contains $m - 2x1$ multiplexers, $m$ - XOR gates and $m$-D flip flops.

The total number of $2x1$ multiplexers = $3m$. 
Table 5.3: COMPARISON OF AREA-TIME COMPLEXITIES OF POLYNOMIAL MULTIPLIERS

<table>
<thead>
<tr>
<th>Design</th>
<th>AND</th>
<th>XOR</th>
<th>Register</th>
<th>MUX</th>
<th>Critical-path</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="AOP">24</a></td>
<td>((m + 1)^2)</td>
<td>((m + 1)^2)</td>
<td>((5/2 \times m^2) + (1/2 \times m) + 7)</td>
<td>(0)</td>
<td>(T_x)</td>
<td>((m/4) + 4)</td>
</tr>
<tr>
<td><a href="t">57</a></td>
<td>(T)</td>
<td>(T + 2)</td>
<td>(T [m/T])</td>
<td>(T)</td>
<td>(T_A + 2T_X)</td>
<td>(m[m/T])</td>
</tr>
<tr>
<td><a href="p">38</a></td>
<td>(2m^2)</td>
<td>(2m^2)</td>
<td>(3m^2)</td>
<td>(0)</td>
<td>(T_A + T_X)</td>
<td>((m + 1)T_{cp})</td>
</tr>
<tr>
<td>fig.[5.1]</td>
<td>(2m)</td>
<td>(m + 2 \cdot \sum_{i=1}^{2L} m/i)</td>
<td>(8m + 2 \cdot \sum_{i=1}^{L} m/(4^i) + 5 + L)</td>
<td>(3m)</td>
<td>(T_{max} + T_X)</td>
<td>(m + 5 + L)</td>
</tr>
</tbody>
</table>

Table 5.4: ASIC SYNTHESIS RESULTS OF POLYNOMIAL MULTIPLIER OVER DIFFERENT \(m\)-degree IRREDUCIBLE POLYNOMIALS

<table>
<thead>
<tr>
<th>(m)</th>
<th>Area</th>
<th>Delay</th>
<th>Power</th>
<th>ADP</th>
<th>PDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOP:162</td>
<td>12140</td>
<td>1.532</td>
<td>1.12</td>
<td>18598.5</td>
<td>1.716</td>
</tr>
<tr>
<td>Trinomial:233</td>
<td>22216.5</td>
<td>1.532</td>
<td>1.59</td>
<td>25619.4</td>
<td>2.436</td>
</tr>
<tr>
<td>Triomial:409</td>
<td>38998</td>
<td>1.532</td>
<td>2.63</td>
<td>59745</td>
<td>4.03</td>
</tr>
<tr>
<td>Pentanomial:571</td>
<td>64650</td>
<td>1.532</td>
<td>3.96</td>
<td>99044</td>
<td>6.07</td>
</tr>
</tbody>
</table>

1: Delay = Critical-Path.
2: ADP: Area-delay product = Area\times Delay.

The total number of XOR gates = \(m + 2 \cdot \sum_{i=1}^{2L} m/i\)

The total number of AND gates = \(2m\).

The total number of 1-bit D flip flops = \(7m + 2 \cdot \sum_{i=1}^{L} m/(4^i) + \text{latency}\).

The critical path of the proposed architecture is given by \(CP = \max \{T_{max} + T_A, T_{max} + T_X, 2T_X\}\) which is typically \(T_{max} + T_X\). Latencies of each block is given below:

- \(A\)-rotation block = 1;
- Multiplication block = 1;
- Selective Buffer block = 1;
- Adder tree block = \(L\);
- Post multiplication block = 1;
- and \(C\)-rotation block = 1.

The latency of the proposed architecture is the summation of \(m\) and latencies of individual blocks. Therefore the latency is \(m + 5 + L\) and the latency of polynomial addition is 2. As per the author’s knowledge there were no generalized polynomial multiplication and addition architecture is proposed. The hardware and time complexities of the proposed architecture is compared with preexisting AOP, trinomial, and pentanomial multiplication architectures in the table 5.3.
Chapter 6

Conclusion

The first design focuses on pentanomial multiplier over $GF(2^m)$ with AOP computation core to lower register-complexity and improve latency. There are many architectures proposed for pentanomial based multipliers, but in our multiplier we embedded AOP like regular and simple computation core as a component to implement pentanomial multiplier. This reduces the register complexity and hence overall area complexity. In the second design, we proposed obfuscated polynomial addition/multiplication model compatible for low area constraint hardwares. This architecture is useful in ECC implementation which requires finite field operations like point multiplication, addition, squaring and inversion. In devices like wearable devices, chip enabled credit cards and debit cards, etc. It is required that area of hardware should be relatively low. We have implemented addition within multiplication hence the need for extra $m$ XOR gates is nullified. So, resource sharing plays a prominent role in implementing low area cost architectures. The proposed architecture can be implemented in cryptographic circuits, error correcting codes, etc. The design yields good results in terms of area complexity with a cost of latency.
Chapter 7

Future Research

7.1 Improving AOP Computation Core based Pentanomial Architecture

It is in our interest to improve efficiency of the proposed architecture by optimizing the structure. The optimization has its limitations like to optimize area complexity would result in increase in latency and vice versa. The reduction in latency can be achieved by applying different strategies like Karatsuba algorithm or Montgomery algorithm which can change latency and we might get improvement in area.

Proposed architecture has an AOP computation core as a component. This decreases register complexity to some extent. We can further reduce area complexity and might get improvement in latency if we embedded low latency trinomial based computation core and derive an algorithm for the pentanomial multiplication with respect to trinomial multiplication.
7.2 Further Improvements and Applications of Addition/Multiplication of Polynomial Architecture

ECC requires addition, multiplication, squaring and inverse operation to be performed on message (data) bits. In our architecture, we have implemented addition and multiplication for various irreducible polynomials. The obligation of reducing area complexity is increasing in recent years. Resource sharing provides better alternative for the reduction of area complexity. So the proposed architecture is useful for low area constraint devices. The switching of irreducible polynomials and obfuscation structure aid these devices. In future we propose a hybrid architecture which can perform all arithmetic operations include addition, multiplication, squaring, inversion and division (note that division is multiplication of the data with its inverse) so that there is not going to be a need for multiple architectures for multiple operations. These operations share resources among themselves hence this can save a lot more area.

Future research includes, proposal of low area obfuscated ECC computation model. This model uses the above mentioned hybrid architecture to perform polynomial arithmetic over $GF(2^m)$ and in between two messages, we can change the irreducible polynomial and this results in improvement in security. If an adversary even break the first message, he has to invest the complete effort to break the second message because of change in irreducible polynomial. This will slow down the connection establishment and data encryption/decryption but we can ameliorate the robustness of hardware.
Chapter 8

Publications


2. S. N. Basha, J. Xie, “Low Register Complexity AOP based Bit Parallel and Bit Serial Systolic Multiplications over \( GF(2^m) \) for General Pentanomials,” IEEE TVLSI, prepared for submission.

3. S. N. Basha, J. Xie, “Unified hardware for polynomial addition and multiplication over \( GF(2^m) \),” IEEE Trans. Circuits & Systems-I, prepared for submission.
Chapter 9

Reference


2014.
