Low-Power and Area-Efficient Finite Field Multiplication Architectures for Cryptographic Applications

Submitted in partial fulfilment of the requirements

for the award of the degree of

Doctor of Philosophy

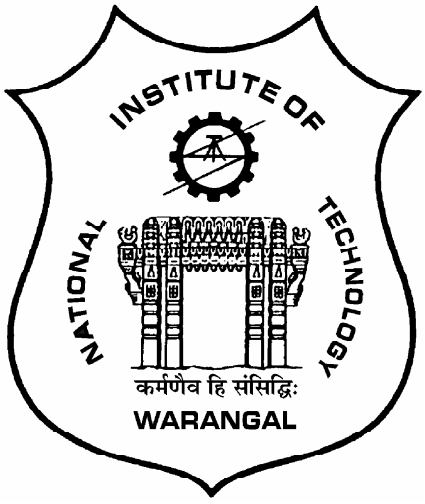
by

Sudha Ellison Mathe

(Roll No: 701353)

Under the supervision of

Dr. B. Lakshmi



Department of Electronics & Communication Engineering

National Institute of Technology Warangal

Telangana, India - 506004

2018

Dedicated

To

My Family,

Teachers & Friends

Approval Sheet

This thesis entitled **Low-Power and Area-Efficient Finite Field Multiplication Architectures for Cryptographic Applications** by **Sudha Ellison Mathe** is approved for the degree of **Doctor of Philosophy**.

Examiners

Research Supervisor

Dr. B. Lakshmi
Department of ECE
NIT Warangal, India-506004

Chairman & Head

Prof. N. Bheema Rao
Department of ECE
NIT Warangal, India-506004

Place:

Date:

Declaration

This is to certify that the work presented in this thesis entitled **Low-Power and Area-Efficient Finite Field Multiplication Architectures for Cryptographic Applications** is a bonafied work done by me under the supervision of **Dr. B. Lakshmi** and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my own ideas and even considered others ideas which are adequately cited and further referenced the original sources. I understand that any violation of the above will cause disciplinary action by the institute and can also evoke panel action from the sources or from whom proper permission has not been taken when needed. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea or data or fact or source in my submission.

Place:

Date:

Sudha Ellison Mathe
Research Scholar
Roll No.: 701353

NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL, INDIA-506004

Department of Electronics & Communication Engineering

CERTIFICATE

This is to certify that the thesis work entitled **Low-Power and Area-Efficient Finite Field Multiplication Architectures for Cryptographic Applications** is a bonafide record of work carried out by **Sudha Ellison Mathe** submitted to the faculty of **Electronics & Communication Engineering** department, in partial fulfilment of the requirements for the award of the degree of **Doctor of Philosophy** in **Electronics and Communication Engineering, National Institute of Technology Warangal, India-506004**. The contributions embodied in this thesis have not been submitted to any other university or institute for the award of any degree.

Dr. B. Lakshmi

Research Supervisor

Associate Professor

Department of ECE

NIT Warangal, India-506 004.

Place:

Date:

Acknowledgements

First, I would like to express my sincere gratitude to my supervisor Dr. B. Lakshmi for her continuous support, patience and motivation during the entire duration of my Ph.D study. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better supervisor and mentor for my Ph.D.

Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. N. Bheema Rao, Prof. C. B. Rama Rao, Dr. P. Srihari Rao, and Prof. N. Subrahmanyam, for their insightful comments and encouragement that incited me to widen my research from various perspectives. I would also like to extend my gratitude to all the faculty of E.C.E Department who have been supportive, friendly and always helpful.

The role of supportive and welcoming friends and colleagues in the life of a researcher is undeniable. To this end, I would like to thank all my friends who have helped and supported me. Especially, I would like to express my deepest gratitude to my closest friends Mr. Shravan, Mr. Kiran and Mr. Ranjith who have always been there for me during my entire research period and have immensely supported and encouraged me through my difficult times. I would also like to thank my seniors and co-scholars Dr. Keerthi, Dr. Siva Prasad, Mr. Sandeep, Mr. Suman, Mr. Rahul, Mr. Shashank, Mr. Ashok, Mr. Sivaram and Mr. Pradeep for supporting me.

I would like to thank my Father, Mother, Sister and all my near and dear for their immense support and encouragement. Last and most importantly, I would like to thank and praise God for protecting, guiding and helping me, not only during my research period, but also during my entire life.

Sudha Ellison Mathe

Abstract

Cryptography in the current digital world is a prime necessity for secure communication of sensitive data over an insecure channel. In the present day, modern cryptography provides such security using computer algorithms which encrypt and decrypt information at the sender and receiver terminals, respectively. In our study of the cryptographic algorithms, it is observed that the multiplication in finite fields is the most extensively used and also the most compute-intensive operation. In order to optimize this finite field multiplication, several techniques to perform efficient finite field multiplications have been proposed in the literature to reduce the computational complexity.

The cryptographic algorithms can also be realized in hardware to achieve enhanced security and high speed compared to software implementations. Therefore, several optimized hardware architectures to compute finite field multiplications have been proposed in the literature over the years. Optimizations in hardware architectures are achieved with respect to three performance parameters: area complexity, time delay and power consumption. In the literature, these optimizations are achieved using several techniques. Among such techniques, it is observed that the interleaved multiplication technique provides low computational complexity and low area complexity.

In this work, efficient hardware architectures for finite field multiplications are realized by employing interleaved multiplication algorithms derived from a conventional interleaved multiplication algorithm. The efficiency of these hardware architectures are verified by employing them in realizing cryptographic applications such as the Advanced Encryption Standard (AES) and Twofish. The HDL models of these AES and Twofish algorithms are implemented using Xilinx Field Programmable Gate Array (FPGA) prototype board and also synthesized using Synopsys Design Vision compiler which is an Application Specific Integrated Circuit (ASIC) tool.

In this research, some interleaved multiplication algorithms are derived from a conventional interleaved multiplication algorithm available in the literature. Subsequently, efficient multiplier architectures for finite field multiplications over $GF(2^m)$ are realized for the proposed algorithms. Firstly, a sequential multiplier architecture over $GF(2^m)$ for irreducible polynomials is proposed. It can perform multiplications over any field of order m and for any irreducible polynomial of that field. In addition, a sequential multiplier architecture over $GF(2^8)$ for irreducible polynomials is derived from the proposed sequential architecture over $GF(2^m)$. Two cryptographic algorithms, Advanced Encryption Standard (AES) and Twofish, are developed employing this proposed architecture over $GF(2^8)$. Secondly, a systolic multiplier architecture over $GF(2^m)$ for irreducible polynomials is proposed that can also perform multiplications over any field of order m and for any irreducible polynomial of that field. Moreover, a systolic multiplier architecture over $GF(2^8)$ for irreducible polynomials is derived from the proposed systolic architecture over $GF(2^m)$. Two cryptographic algorithms, AES and Twofish, are developed employing this proposed architecture over $GF(2^8)$. Thirdly, a systolic multiplier architecture over $GF(2^m)$ for irreducible trinomials is proposed. This architecture can perform multiplications over any field of order m. However, it may be noted that irreducible trinomials should exist for that field. Lastly, a systolic multiplier architecture over $GF(2^m)$ for irreducible pentanomials is proposed. This architecture can also perform multiplications over any field of order m with the similar condition that irreducible pentanomials should exist for the field considered. The performance of these proposed architectures are verified analytically and also by implementing them using ASIC and FPGA technologies by computing the area complexity, power consumption, area-delay product and power-delay product. These results are compared with the performance of the existing architectures available in the literature. It is observed from these performance comparisons that the proposed architectures outperform the existing architectures.

Contents

List of Figures

List of Tables

List of Abbreviations

DES	Data Encryption Standard
AES	Advanced Encryption Standard
ECC	Elliptic Curve Cryptography
RSA	Rivest Shamir Adleman
FFT	Fast Fourier Transform
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
ADP	Area-Delay Product
PDP	Power-Delay Product
MUX	Multiplexer
DMUX	De-Multiplexer
MSB	Most Significant Bit
LSB	Least Significant Bit
NIST	National Institute of Standards and Technology
CMOS	Complementary Metal Oxide Semiconductor

Chapter 1

Introduction

Cryptography is the study of hiding sensitive information and sending it on any channel/medium to avoid any kind of unauthorized access by third parties. It involves the process of converting an intelligible message into an unintelligible one by the sender. This unintelligible message needs to be converted back by the receiver to retrieve the original intelligible message. This ensures that only the sender and receiver have access to the original intelligible message. Integrity, Confidentiality, Authentication and Non-Repudiation are four pillars of cryptography, which collectively form an envelope of information security to guard the sensitive data communicated between the sender and receiver. Cryptography can be broadly classified into classical cryptography and modern cryptography. Classical cryptography dates back to around 1900 B.C, where sensitive messages were required to be communicated secretly using techniques such as substitution, transposition, pictorial representation, puzzles etc. Tangible information was stored or transported secretly using mechanical devices such as cryptex, rotor machines etc.

With the advent of computers in mid 1900s, modern cryptography came into existence to provide security with the help of computers. Modern cryptography involves two stages, namely, encryption and decryption. Encryption is performed on the original message at the sender side using a secret key to obtain an unreadable cipher message that can be sent over a communication channel. The receiver performs decryption of the received cipher message to attain the original message using a secret key. Based on the key sharing strategy, the techniques in modern cryptography can be divided into symmetric-key cryptography and asymmetric-key cryptography [1]. The encryption and decryption performed

using the same key is known as symmetric-key cryptography, whereas the encryption and decryption performed using different keys is known as asymmetric-key cryptography. Few techniques developed using the symmetric-key principle are Data Encryption Standard (DES) [2] and Advanced Encryption Standard (AES) [3]. Few techniques developed using the asymmetric-key principle are Elliptic Curve Cryptography (ECC) [4, 5] and Rivest-Shamir-Adleman (RSA) [6]. The encryption and decryption processes are performed using computer algorithms designed based on the concepts of mathematics and computer science. These cryptographic algorithms are impervious to almost any kind of external attacks by third parties who try to steal or modify sensitive information being communicated.

In our study, it is observed that these cryptographic algorithms involve the following operations such as multiplication in finite fields, modular addition, logical XOR, logical AND, rotate/shift, exponentiation, modular reduction, Elliptic curve arithmetic etc. Among these operations, the multiplication in finite fields is the most complex and compute intensive operation, whereas other operations are very simple and straightforward. Moreover, multiplication operation is used extensively in almost all the cryptographic algorithms due to its mathematical properties that are useful for the encryption and decryption processes.

In abstract algebra [7], a field is a non-zero commutative ring that contains a multiplicative inverse for every non-zero element, or equivalently a ring whose non-zero elements form an abelian group under multiplication. As such, it is an algebraic structure with notions of addition, subtraction, multiplication, and division satisfying the appropriate abelian group equations and distributive law. The properties of fields are closure of field under addition and multiplication, associative property of addition and multiplication, commutative property of addition and multiplication, existence of additive and multiplicative identity elements, existence of additive inverses and multiplicative inverses, distributive property of multiplication over addition.

A finite field is a field that contains finite number of elements. A finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and it possesses the above properties. The most common example of finite fields is the 'integers *mod* m', where m is a prime number. The number of elements of a finite field is called its

order. A finite field of order q exists if and only if the order q is a prime power p^k (where p is a prime number and k is a positive integer). In a field of order p^k, adding p copies of any element always results in zero i.e. the characteristic of the field is p. The finite field with p^m elements is denoted as $GF(p^m)$ and is also called as the Galois Field, in honour of the founder of finite field theory, Évariste Galois. $GF(p)$ is simply a ring of integers modulo p and is called a prime field of order p, where p is a prime number. That is, one can perform operations (addition, subtraction, multiplication) using the usual operation on integers, followed by reduction modulo p on this prime field. An m-dimensional vector space, called the basis, allows for an extension field $GF(2^m)$ to exist over $GF(2)$, where the elements of the field can be represented using polynomials whose coefficients belong to $GF(2)$. The basis of a field is given by the set $\{1, \alpha, \alpha^2, \alpha^3, \cdots, \alpha^m\}$, where α is the root of an irreducible polynomial of that field. Here, the irreducible polynomial of a finite field can be defined as the polynomial that cannot be factored into the product of two or more polynomials belonging to the said finite field.

Finite field operations can be performed using three basis representations, namely, Polynomial, Normal and Dual Basis [7]. The multiplication operation in these bases are performed as follows: both multiplier and multiplicand uses normal basis representation in the realization of a normal basis multiplier; the multiplicand uses dual basis representation and the multiplier uses polynomial basis representation in the realization of dual basis multiplier; both the multiplier and multiplicand use polynomial basis representation in the realization of a polynomial basis multiplier. The polynomial basis multiplier does not require basis conversion and can be readily matched to any system, whereas basis conversions are necessary for normal and dual basis multipliers. In polynomial basis representation, elements of $GF(2^m)$ can be represented as polynomials of degree strictly less than m over $GF(2)$. All finite field operations involve the modulo reduction by R, where R is an irreducible polynomial of degree m over $GF(2)$. The addition of two polynomials P and Q is performed by usual polynomial addition. Multiplication can be performed as follows: $W = P.Q$ is computed by usual polynomial multiplication followed by modulo reduction of resultant polynomial W by the irreducible polynomial R using polynomial long division. The remainder polynomial attained as a result of the long division operation represents the final result of the finite field multiplication operation.

Several techniques to realize these finite field multiplications have been proposed in the literature. Karatsuba et al. [8] proposed the Karatsuba-Ofman multiplication method in 1962. The hardware realizations of this method employ parallel architectures resulting in high speed at the expense of more area complexity. Montgomery [9] proposed the Montgomery multiplication method in 1985. This method is preferred for realization of high speed multipliers as it simplifies modulo reduction. However, this method employs parallel architectures resulting in high area complexity. Mastrovito [10] proposed the Mastrovito multiplication method in 1988. This method uses matrix computations to achieve fast multiplications. However, this method requires high area complexity due to the 2-dimensional multiplication characteristic of matrix computations. In 1989, Cantor [11] proposed a multiplication method based on the fast fourier transform (FFT) method. This method also achieves fast multiplications but requires high area complexity due to the extra hardware required in realizing the FFT functions. The idea of interleaving the modular reduction with the polynomial multiplication was introduced by Blakely [12] in 1983. This method is known as the interleaved modular multiplication, or simply interleaved multiplication. Since the two steps involved in finite field multiplications are interleaved, the computational complexity is significantly reduced. Hence, several multiplier architectures are proposed in the literature for realizing this interleaved multiplication method to achieve low area complexity.

1.1 Motivation and Objective

Recent advances in technology have enabled vast usage of portable devices in several applications. Security, size of the portable device and power consumption are of major concern in such devices. Moreover, there is a continuous demand for increase in security, reduction in size and power consumption of these portable devices. Security can be increased by implementing the cryptographic algorithms in hardware and by increasing its bit-width. The performance metrics of hardware implementation of any security algorithm are area complexity and power consumption i.e. the size of the device depends on the area complexity; power consumption of the device also depends on the area complexity and computational complexity. However, increasing the bit-width increases the

computational complexity which in turn leads to increase in area complexity and power consumption. Hence, there is a need to reduce the computational complexity of the cryptographic algorithm to reduce area complexity and power consumption while maintaining the same security. Since major portion of computational complexity of a cryptographic algorithm is due to the finite field multiplications, reduction in area complexity and power consumption of finite field multiplier architectures are imperative in order to reduce the overall size and power consumption of the portable devices.

In this research, polynomial basis finite field multiplications are considered as they are less complex and allow low hardware structures compared to the other two basis representations. In order to achieve further reduction in the computational complexity of finite field multiplications, several efficient techniques and algorithms were reported in the literature. These algorithms are used to realize hardware architectures that require low area complexity and/or low delay and/or low power consumption. In this work, we have attempted to propose low-power and area-efficient sequential and systolic architectures for realizing the finite field multiplications.

The objectives of this research are summarized as follows:

- Reduction in area complexity is imperative for reduction in device size and also to achieve subsequent reduction in power consumption. Hence, an interleaved multiplication algorithm is derived from a conventional interleaved algorithm. A sequential multiplier architecture is designed based on the derived algorithm to reduce the area complexity with minimum increase in delay. This sequential multiplier is implemented using ASIC and FPGA technologies to compute area complexity, power consumption and delay performance and compared with the existing works.

- Systolic architectures play an important role in high-speed circuits but they have the drawback of high area overheads. Hence, an interleaved multiplication algorithm is derived to design a systolic multiplier architecture which results in low area complexity. This systolic architecture is implemented in ASIC and FPGA technologies to compute its area, power consumption and delay performance and compared with the existing works.

- Multiplier architectures based on special classes of polynomials, namely trinomials

and pentanomials, are designed to achieve further reduction in area complexity compared to systolic multiplier for irreducible polynomials. Hence, a novel Pre-Computation technique is introduced to design systolic multiplier architectures for trinomials and penatanomials. These architectures are implemented in ASIC and FPGA technologies to compute area, power consumption and delay performance and compared with the existing works.

1.2 Thesis Contributions

The contributions of the thesis are summarized as follows:

- **Low-Power and Area-Efficient Sequential Multipliers over Polynomial Basis.** A sequential multiplier architecture over polynomial basis that performs multiplication of any two random finite field elements for any irreducible polynomial is proposed. The performance of this proposed architecture is evaluated through theoretical analysis and practical hardware implementations. The contributions of this work are briefly described as:
  - **Proposed Interleaved Multiplication Algorithm:** A modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm to perform finite field multiplications of any two arbitrary elements over $GF(2^m)$. This algorithm allows realization of a sequential multiplier architecture that achieves low area complexity compared to the existing works.
  - **Proposed Sequential Multiplier Architecture over $GF(2^m)$ for Irreducible Polynomials**: A sequential multiplier architecture is developed based on the proposed interleaved multiplication algorithm. Area complexity, delay and area-delay product (ADP) analysis of the proposed architecture is performed. The proposed architecture achieves a minimum reduction of about 28% in ADP compared to the previous works for the field of order $m = 163$. The ASIC and FPGA implementations of the proposed architecture indicate a minimum reduction of about 49% in area, 16% in power consumption, 13% in

ADP and 45% in power-delay product (PDP) compared to the existing works.

- **Proposed Sequential Multiplier Architecture over $GF(2^8)$ for Irreducible Polynomials:** A sequential multiplier architecture over $GF(2^8)$ is derived from the proposed sequential multiplier architecture over $GF(2^m)$. Area complexity, delay and ADP analysis of the proposed architecture is performed. The proposed architecture achieves a minimum reduction of about 29% in ADP compared to the previous works. The FPGA implementation of two cryptographic algorithms employing the proposed architecture achieves a minimum reduction of 22% in area, 42% in power consumption, 34% in ADP and 41% in PDP compared to the existing works.

- **Low-Power and Area-Efficient Systolic Multipliers over Polynomial Basis.** A systolic multiplier architecture over polynomial basis that performs multiplication of any two random finite field elements for any irreducible polynomial is proposed. The performance of this proposed architecture is evaluated through theoretical analysis and practical hardware implementations. The contributions of this work are briefly described as:

  - **Proposed Interleaved Multiplication Algorithm:** A modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm to perform finite field multiplications of any two arbitrary elements over $GF(2^m)$. This algorithm allows realization of a systolic multiplier architecture that achieves low area complexity compared to the existing works.

  - **Proposed Systolic Multiplier Architecture over $GF(2^m)$ for Irreducible Polynomials:** A systolic multiplier architecture is developed based on the proposed interleaved multiplication algorithm. Area complexity, delay and ADP analysis of the proposed architecture is performed. The proposed multiplier achieves a minimum reduction of about 20% in area complexity compared to previous works for the field of order $m = 163$. The ASIC and FPGA implementations of the proposed multiplier indicate a minimum reduction of about 35% in area, 43% in power consumption, 73% in ADP and 76% in PDP compared to the existing works.

  - **Proposed Systolic Multiplier Architecture over $GF(2^8)$ for Irreducible**

Polynomials: A systolic multiplier architecture is developed based on the proposed interleaved multiplication algorithm. Area complexity, delay and ADP analysis of the proposed architecture is performed. The proposed architecture achieves a minimum reduction of about 21% in area complexity compared to previous works. The FPGA implementation of two cryptographic algorithms employing the proposed architecture achieves a minimum reduction of 24% in area, 46% in power consumption, 33% in ADP and 41% in PDP compared to the existing works.

- **Low-Power and Area-Efficient Systolic Multipliers for Special Classes of Irreducible Polynomials.** Two systolic multiplier architectures over polynomial basis for any irreducible trinomial or pentanomial are proposed to perform multiplication of any two random finite field elements. The performance of this proposed architectures is evaluated through theoretical analysis and practical hardware implementations. The contributions of this work are briefly described as:
  - **Proposed Interleaved Multiplication Algorithm for Irreducible Trinomials and Pentanomials:** A modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm based on a novel Pre-Computation technique. This algorithm performs finite field multiplications of any two arbitrary elements over $GF(2^m)$ and allows realization of two systolic multiplier architectures that achieve low area complexity compared to the existing works.
  - **Proposed Systolic Multiplier Architecture over $GF(2^m)$ for Irreducible Trinomials:** A systolic multiplier architecture for irreducible trinomials is developed based on the proposed interleaved multiplication algorithm. Area complexity, delay and ADP analysis of the proposed architecture is performed. The proposed architecture achieves a minimum reduction of about 28% in area complexity and about 17% in ADP compared to previous works for the field of order $m = 233$. The ASIC and FPGA implementations of the proposed architecture indicates a minimum reduction of 40% in area, 31% in power consumption, 44% in ADP and 36% in PDP compared to the existing works.
  - **Proposed Systolic Multiplier Architecture over $GF(2^m)$ for Irreducible**

Pentanomials: A systolic multiplier architecture for irreducible pentanomials is developed based on the proposed interleaved multiplication algorithm. Area complexity, delay and ADP analysis of the proposed architecture is performed. The proposed architecture achieves a minimum reduction of about 28% in area complexity compared to previous works for the field of order m = 283. The ASIC and FPGA implementations of the proposed architecture indicates a minimum reduction of 40% in area and 59% in power consumption compared to the existing works.

1.3 Thesis Organization

The rest of the thesis is structured as follows:

Chapter 2 presents a brief overview of the fundamental concepts of finite field theory. First, the basic concepts of Groups, Rings and Fields are presented followed by the concept of polynomial Rings. The construction of finite fields over $GF(p^m)$ is presented followed by the various basis representations over which a field can be constructed. Finally, the multiplication operation is presented along with some algorithms that describe the operation in detail.

Chapter 3 presents the finite field multiplication architectures proposed in the literature. The chapter begins with sequential multiplier architectures for irreducible polynomials followed by systolic multiplier architectures for irreducible polynomials and systolic multiplier architectures for irreducible trinomials and pentanomials. The performance of these architectures, i.e. area complexity, latency and critical path delay, are also discussed in detail.

Chapter 4 presents a sequential multiplier architecture over $GF(2^m)$ for irreducible polynomials along with its analysis and hardware implementations and the results are compared with existing works. It also presents the design of a sequential multiplier architecture over $GF(2^8)$ for irreducible polynomials derived from the proposed multiplier over $GF(2^m)$. Two cryptographic algorithms employing the proposed architecture are implemented on FPGA and the results are compared with existing works.

Chapter 5 presents a systolic multiplier architecture over $GF(2^m)$ for irreducible polyno-

mials along with its analysis and hardware implementations and the results are compared with existing works. It also presents the design of a systolic multiplier architecture over $GF(2^8)$ for irreducible polynomials derived from the proposed multiplier over $GF(2^m)$. Two cryptographic algorithms employing the proposed architecture are implemented on FPGA and the results are compared with existing works.

Chapter 6 presents a systolic multiplier architecture over $GF(2^m)$ for irreducible trinomials along with its analysis and hardware implementations and the results are compared with existing works. It also presents a systolic multiplier architecture over $GF(2^m)$ for irreducible pentanomials along with its analysis and hardware implementations and the results are compared with existing works.

Chapter 7 concludes the thesis and presents some possible directions for future work.

1.4 Conclusion

In this chapter, a brief overview of the entire research work along with the motivation behind this research and objectives are presented. The next chapter presents an overview of the mathematical concepts of finite field theory.

Chapter 2

Finite Field Theory

This chapter presents the fundamental concepts of finite field theory [7] required for implementing finite field multiplications. Firstly, the mathematical preliminaries of a Group and its properties along with some examples are presented, followed by the concept of Rings and Fields along with some examples. Secondly, different bases for finite fields and their representations are presented followed by the concept of polynomials in a finite field and its properties. Thirdly, the construction of finite fields over $GF(p^m)$ using the polynomial basis is presented. Finally, the multiplication process over polynomial basis is presented along with some examples to describe the operation in detail.

2.1 Groups, Rings and Fields

This section presents the definitions, properties and examples of Groups, Rings and Fields.

2.1.1 Groups

Definition 2.1. A set S is said to be a group if there exists a binary operation $*$ on the set satisfying the following properties:
(i) The operation $*$ obeys associative law

$$a * (b * c) = (a * b) * c \tag{2.1}$$

for all $a, b, c \in S$

(ii) An identity element $e \in S$ exists such that

$$e * a = a * e = a \tag{2.2}$$

for all $a \in S$

(iii) For all $a \in S$, there is an element $a^{-1} \in S$, such that

$$a * a^{-1} = a^{-1} * a = e \tag{2.3}$$

where, a^{-1} is known as the inverse of a.

A group is said be an abelian group if it also satisfies the commutative property in addition to the above i.e. $a * b = b * a$ for all $a, b \in S$. It can be noted that the group notation used for the group operation is multiplicative in nature. Additive group notation can also be used for the group operation where the identity element is often associated with a zero (0) element and $-a$ is the inverse of the element a.

Example 2.1. (a) An example of a group under the addition operation with identity element 0 and under the multiplication operation with identity element 1 is the set of real numbers R.

(b) Another example of an additive group with identity element 0 is the set of integers Z.

(c) An example of a group under addition modulo m with identity element 0 is the set of integers modulo m, Z_m. However, the group Z_m does not have multiplicative inverses for all its elements and hence is not a group under multiplication modulo m operation.

Corollary 2.1. A group S can have finite number of elements in it and its order is denoted as $|S|$.

Definition 2.2. A group S is cyclic if an element $a \in S$ exists such that there exists an integer j and $b = a^j$ for each $b \in S$, then the group S is said to be cyclic and the element a is called generator of S represented as $S =< a >$ i.e. a must generate all the elements in S. The order of $b \in S$ can be defined as the least positive integer l such that $b^l = e$, where e is an identity element in S. Here, the order of an element $b \in S$ is represented as $ord(\beta)$.

Example 2.2. A group of integers modulo 5, $Z_5^* = \{1, 2, 3, 4\}$, is a cyclic group with generator 2 i.e. all the elements in the group can be generated under the multiplication

modulo 5 operation using the generator 2. It can be shown as follows: $2 \equiv 2 \, mod \, 5$, $22 \equiv 4 \, mod \, 5$, $23 \equiv 8 \equiv 3 \, mod \, 5$, $24 \equiv 16 \equiv 1 \, mod \, 5$.

Corollary 2.2. In addition, if $a \in Z_m^*$ is a generator element, then b is also a generator element where $b \equiv a^j \, mod \, m$ and $gcd(j, f(m)) = 1$.

2.1.2 Rings and Fields

Definition 2.3. A set R together with two binary operations $+$ and $*$ on R is said to be a Ring if it satisfies the following properties:
(i) $(R, +)$ must be an abelian group under the additive operation $+$ having the identity element 0.
(ii) The operation $*$ obeys associative law

$$a * (b * c) = (a * b) * c \tag{2.4}$$

for all $a, b, c \in R$
(iii) There is a multiplicative identity element 1 such that

$$a * 1 = 1 * a = a \tag{2.5}$$

where, $1 \neq 0$ and $a \in R$.
(iv) The operation $*$ obeys distributive law over $+$ operation

$$\begin{aligned} a * (b + c) &= (a * b) + (a * c) \\ (b + c) * a &= (b * a) + (c * a) \end{aligned} \tag{2.6}$$

for all $a, b, c \in R$

Example 2.3. (a) A commutative ring is the set of integers modulo m, Z_m, under the addition and multiplication modulo m operations.
(b) Another example is the set of integers Z along under the usual addition and multiplication operations can be considered as a commutative ring.
(c) Other examples of commutative rings are set of all rational numbers Q, set of all real numbers R, and set of all complex numbers C under the usual addition and multiplication operations.

Corollary 2.3. A ring is said to be a 'commutative ring' if the operation $*$ obeys commutative law i.e., $a * b = b * a$.

Definition 2.4. A commutative ring in which all non-zero elements have multiplicative inverses is said to be a field F. A field G can be termed as the subfield of F if G is a subset of F and G is also a field with respect to the operations in F. Here, F is the extension field of G.

Example 2.4. (a) Some examples of fields are the set of all real numbers R, set of all complex numbers C, and the set of all rational numbers Q.
(b) Another example of a field is the set of all integers modulo m, Z_m, under the $+$ and $*$ operations, where m is prime.

Definition 2.5. For any value of $m \geq 1$, if $1 + 1 + \cdots + 1$ (m times) is never equal to 0, then 0 is termed as the characteristic of the field. On the contrary, if $\sum_{i=0}^{m} 1 = 0$ then the least positive integer m is termed as the characteristic of the field.
It can be observed that $Z_2, Z_3, Z_5, \cdots, Z_p$ are fields having characteristic p, where p is a prime. These fields have finite number of elements and hence they are termed as finite fields, where the number of elements in the field is its order. It is also known as Galois fields named after Évariste Galois who introduced the concept of finite fields.

Example 2.5. (a) The inverse of any integer $a \, mod \, p$ can be denoted as 'a^{p2}' where $gcd(a, p) = 1$ and $a^{p1} = 1 \, mod \, p$, where p is a prime and $a < p$.
(b) The inverse of 3 modulo 7 i.e. $3^{-1} \, mod \, 7$, can be found as $3^5 = 243 \equiv 5 \, mod \, 7$. Hence, 5 is the inverse of 3 modulo 7 i.e. $3.5 = 15 \equiv 1 \, mod \, 7$.
(c) Consider two integers v and w such that $a.w + p.v = h = gcd(a, p)$. The inverse of $a \, mod \, p$ can be computed as $a.w + p.v = 1 \Rightarrow a.w \equiv 1 \, mod \, p \Rightarrow a^{-1} \equiv w \, mod \, p$, if $gcd(a, p) = 1$. This is the extended Euclidean algorithm for finding inverse of an element.

Some basic properties of finite fields are as follows:
(i) (Existence and uniqueness of finite fields) If F is a finite field then F contains p^m elements for some prime p and positive integer $m \geq 1$. For every prime power p^m, there is a unique finite field of order p^m. Informally speaking, two finite fields are isomorphic if they are structurally the same, although the representation of their field elements may be different.

(ii) If $GF(q)$ is a finite field of order $q = p^m$, p a prime, then the characteristic of $GF(q)$ is p. In addition, $GF(q)$ contains a copy of $GF(p)$ as a subfield. Hence, $GF(q)$ can be viewed as an extension of $GF(p)$ of degree m.
(iii) Let $GF(q)$ a finite field of order $q = p^m$, then every subfield of $GF(q)$ has order p^n for some positive divisor n of m. Conversely, if n is a positive divisor of m, then there is exactly one subfield of GF(q) of order p^n. An element $A \in GF(q)$ is in the subfield $GF(p^n)$ if and only if $A^{p^n} \equiv A$. The non-zero elements of $GF(q)$ form a group under multiplication called the multiplicative group of $GF(q)$, denoted $GF(q)^*$. In fact $GF(q)^*$ is a cyclic group of order $q-1$. Thus, $A^q = A$ for all $A \in GF(q)$.
(iv) Let $A \in GF(q)$, with $q = p^m$, then the multiplicative inverse of A can be computed as $A^{-1} \equiv A^{q-2}$. Alternatively, one can use the extended Euclidean algorithm for polynomials to find $S(\alpha)$ and $T(\alpha)$ such that $S(\alpha)A(\alpha) + T(\alpha)P(\alpha) = 1$, where $P(x)$ is an irreducible polynomial of degree m over $GF(p)$. Then, the multiplicative inverse $A^{-1} = S(\alpha)$.
(v) If $A, B \in GF(q)$, with $GF(q)$ a finite field of characteristic p, then

$$(A + B)^{p^t} = A^{p^t} + B^{p^t} \tag{2.7}$$

for all $t \geq 0$.

2.2 Polynomial Rings

Definition 2.6. Let R be a commutative ring, then a polynomial over R can be expressed as: $P(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_2 x^2 + p_1 x + p_0$ where each $p_i \in R$, x is an indeterminate value and $n \geq 0$. The element p_i is the coefficient of x^i in $P(x)$ and the degree of $P(x)$ is the largest n for which the leading coefficient $p_n \neq 0$ and is denoted by $deg(P(x))$. $P(x)$ is known as monic polynomial if $p_n = 1$. $P(x)$ is known as constant polynomial if there exists only one constant term in the polynomial i.e. a_0, where degree the of the polynomial is zero. $P(x)$ is known as zero polynomial if all the coefficients of the polynomial are zero.

Example 2.6. (i) The addition of two polynomials can be computed using the expression

$$A(x) + B(x) = \sum_{i=0}^{n} (a_i + b_i)\, x^i \tag{2.8}$$

(ii) The product of two polynomials $A(x) = \sum a_i x^i$ and $B(x) = \sum b_i x^i$ over R is defined

as

$$C(x) = A(x).B(x) = \sum_{k=0}^{n+m} c_k x^k \tag{2.9}$$

Here, c_k is computed using the expression $c_k = \sum a_i b_j$ where $i + j = k$, $0 \leq i \leq n$, $0 \leq j \leq m$ and the addition and multiplication of coefficients is performed in R. Together with the operations of addition and multiplication defined as above it is easily seen that the set of polynomials over R forms a ring.

Corollary 2.4. Consider a commutative ring R. Then the polynomial ring $R[x]$ represents the set of polynomials over R with addition and multiplication of polynomials (see Example 2.6).

Definition 2.7. Consider a polynomial $T(x) \in F[x]$ having a positive degree and $T(x) = A(x).B(x)$, where $A(x)$ or $B(x)$ are constant polynomials. Then $T(x)$ can be termed as an irreducible polynomial over F. If either $A(x)$ or $B(x)$ is not a constant polynomial, then $T(x)$ is a reducible polynomial.

Corollary 2.5. Consider a polynomial $P(x) \in F[x]$ and if $P(\alpha) = 0$ then α is known as the root of $P(x)$, where $\alpha \in F$.

2.3 Construction of Finite Fields GF(p^m)

Definition 2.8. Let m be a positive integer and $P(x)$ be an irreducible polynomial of degree m over $GF(p)$. Moreover, let α be a root of $P(x)$, i.e., $P(\alpha) = 0$. Then, the Galois field of order p^m and characteristic p, denoted $GF(p^m)$ or F_{p^m}, is the set of polynomials $a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \cdots + a_2\alpha^2 + a_1\alpha + a_0$, with $a_i \in GF(p)$ together with the addition and multiplication operations defined as follows. Let $A(\alpha), B(\alpha), C(\alpha) \in GF(p^m)$, with $A(\alpha) = \sum a_i\alpha^i, B(\alpha) = \sum b_i\alpha^i$, and $C(\alpha) = \sum c_i\alpha^i$, where $a_i, b_i, c_i \in GF(p)$ then:
(i) Addition: $C(\alpha) = A(\alpha) + B(\alpha) = \sum (a_i + b_i)\,\alpha^i$
(ii) Multiplication: Let $C(\alpha)$ to be the result of multiplying $A(\alpha)$ by $B(\alpha)$ via standard polynomial multiplication as described in Example 2.6. Thus, $C(\alpha)$ is a polynomial with $deg(C(\alpha)) \leq 2m - 1$. Then, $D(\alpha)$ is computed using the expression $C(\alpha)$ modulo $P(\alpha)$, i.e., $D(\alpha) \equiv C(\alpha)\, mod\, P(\alpha)$. This modulo operation can be computed if $C(\alpha)$ can be written as $C(\alpha) = P(\alpha)Q(\alpha) + D(\alpha)$, where $Q(\alpha) \in GF(p^m)$ and $deg(D(\alpha)) < m$. Here,

$D(\alpha)$ is the final result of the multiplication operation.

Example 2.7. Let $p = 2$ and $P(x) = x^4 + x + 1$. Then, $P(x)$ is irreducible over $GF(2)$. Let α be a root of $P(x)$, i.e., $P(\alpha) = 0$, then the Galois field $GF(2^4)$ is defined by $GF(2^4) = \{a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 | a_i \in GF(2)\}$ together with addition and multiplication as defined in Definition 2.8. The field $GF(2^4)$ is of characteristic 2 and it has order $2^4 = 16$, in other words, it has 16 elements. The elements of $GF(2^4)$ can be written as shown in Table 2.1.

Table 2.1: Representation of $GF(2^4)$ elements.

As a 4-tuple	As a polynomial	As a power of α
0000	0	0
0001	1	$\alpha^{15} \equiv 1$
0010	α	α
0011	$\alpha + 1$	α^4
0100	α^2	α^2
0101	$\alpha^2 + 1$	α^8
0110	$\alpha^2 + \alpha$	α^5
0111	$\alpha^2 + \alpha + 1$	α^{10}
1000	α^3	α^3
1001	$\alpha^3 + 1$	α^{14}
1010	$\alpha^3 + \alpha$	α^9
1011	$\alpha^3 + \alpha + 1$	α^7
1100	$\alpha^3 + \alpha^2$	α^6
1101	$\alpha^3 + \alpha^2 + 1$	α^{13}
1110	$\alpha^3 + \alpha^2 + \alpha$	α^{11}
1111	$\alpha^3 + \alpha^2 + \alpha + 1$	α^{12}

To add $\alpha^3 + 1$ and $\alpha^3 + \alpha^2 + 1$ we simply perform polynomial addition and reduce the coefficients of the resulting polynomial modulo 2. Thus, $(\alpha^3 + 1) + (\alpha^3 + \alpha^2 + 1) \equiv \alpha^2$. Similarly, $\alpha^3 + 1$ multiplied by $(\alpha^3 + \alpha^2 + 1)$ is obtained as

$$(\alpha^3 + 1).(\alpha^3 + \alpha^2 + 1) = \alpha^6 + \alpha^5 + \alpha^3 + \alpha^3 + \alpha^2 + 1 \tag{2.10}$$

Example 2.8. Let $p = 3$. Then $P(x) = x^3+2x+2$ is irreducible over $GF(3)$. Let β be a root of $P(x)$. Then, the elements of $GF(3^3)$ can be written as polynomials $a_2\beta^2 + a_1\beta + a_0$ with $a_i \in GF(3)$. The order of $GF(3^3)$ is $3^3 = 27$ and the elements of $GF(3^3)$ are $0, 1, 2, \beta, 2\beta, \beta + 1, \beta + 2, 2\beta + 1, 2\beta + 2, \beta^2, \beta^2 + 1, \beta^2 + 2, \beta^2 + \beta, \beta^2 + 2\beta, \beta^2 + \beta + 1, \beta^2 + \beta + 2, \beta^2 + 2\beta + 1, \beta^2 + 2\beta + 2, 2\beta^2, 2\beta^2 + 1, 2\beta^2 + 2, 2\beta^2 + \beta, 2\beta^2 + 2\beta, 2\beta^2 + \beta + 1, 2\beta^2 + \beta + 2, 2\beta^2 + 2\beta + 1, 2\beta^2 + 2\beta + 2$.

2.4 Basis Representations

It can be observed from Table 2.1 that two different representations of the elements of $GF(2^4)$ are shown. In one case, the elements of $GF(2^4)$ are represented as polynomials, in the other case the elements are represented as powers of a suitable element, say a primitive element. In this sub-section, we describe different types of bases that can be used to represent the elements of a finite field $GF(q^m)$.

Definition 2.9. Let $GF(q^m)$ be an extension of $GF(q)$ and let $\alpha \in GF(q^m)$. Then the elements $\alpha^q, \alpha^{q^2}, \cdots, \alpha^{q^{m-1}}$ are called the conjugates of α with respect to $GF(q)$. Different basis can be used to represent the elements of a finite field as evident from Example 2.8. In particular, the two different representations from Table 2.1 lead to the ideas of polynomial basis and normal basis.

Definition 2.10. Let $E = GF(q^m)$ and $F = GF(q)$ be two fields. Then a basis of E over F of the form $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-2}, \alpha^{m-1}\}$ is called a polynomial basis, where $\alpha \in GF(q^m)$ and it is often taken to be a primitive element. Similarly, a basis of E over F of the form $\left\{\alpha, \alpha^q, \alpha^{q^2}, \cdots, \alpha^{q^{m-1}}\right\}$ receives the name of a normal basis for a suitable element $\alpha \in GF(q^m)$. It can be shown that for any field $GF(q)$ and any extension field $GF(q^m)$, there exists always a normal basis of $GF(q^m)$ over $GF(q)$ (see Theorem 2.35 in Ref. [13]). A lot of research is carried out on finding normal bases that are optimal to perform arithmetic operations. Such normal bases have received the name of optimal normal bases [14] because they allow efficient implementations of arithmetic operations in fields $GF(q^m)$. It may be observed that although there exist always a normal basis for every field, the same is not true in the case of optimal normal bases. Another type of basis which has received attention in the literature is the dual basis.

Definition 2.11. $E = GF(q^m)$ and $F = GF(q)$. Then two bases $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ and $\{\beta_0, \beta_1, \cdots, \beta_{m-1}\}$ of E over F are said to be dual or complementary bases if for $0 \leq i, j \leq m-1$ we have

$$Tr_{E/F}(\alpha_i \beta_j) = \begin{cases} 0 & \text{for } i \neq j \\ 1 & \text{for } i = j \end{cases} \tag{2.11}$$

Here, $Tr_{E/F}(\alpha) = \alpha + \alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^{m-1}}$.

References [15, 16] define the concept of a weakly dual basis as follows:

Definition 2.12. Let E and F be defined as in Definition 2.11. Then two bases $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ and $\{\beta_0, \beta_1, \cdots, \beta_{m-1}\}$ of E over F are said to be weakly dual to each other if for $0 \leq i,\ j \leq m-1$ we have

$$Tr_{E/F}(\gamma \alpha_i \beta_j) = \begin{cases} 0 & \text{for } i \neq j \\ 1 & \text{for } i = j \end{cases} \tag{2.12}$$

for $\gamma \in E \setminus \{0\}$. Reference [17] used weakly dual basis to build finite field multipliers for fields $GF(q^m)$, where q is an odd prime power. Finally, it is important to point out that given a basis $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ of $GF(q^m)$ over $GF(q)$, one can always represent an element $\beta \in GF(q^m)$ as

$$\beta = b_0\alpha_0 + b_1\alpha_1 + \cdots + b_{m-1}\alpha_{m-1} \tag{2.13}$$

where, $b_i \in GF(q)$.

2.5 Multiplication using Polynomial Basis

Multiplication in finite fields can be performed in different ways by viewing finite fields as vector spaces over sub-fields. In order to specify a multiplication rule, it is necessary to choose a basis. Polynomial basis is a better choice than normal basis for software implementations. Hence, some algorithms available in the literature for multiplication in finite fields using polynomial basis are presented in the following sub-sections.

2.5.1 Standard Field Multiplication

Let $a, b \in GF(2^n)$ be two polynomials represented as $a(x) = \sum_{i=0}^{n-1} a_i x^i$ and $b(x) = \sum_{i=0}^{n-1} b_i x^i$, where $a_i, b_i \in GF(2)$ or, equivalently, as vectors $a = (a_{n-1}, \cdots, a_1, a_0)$ and $b = (b_{n-1}, \cdots, b_1, b_0)$. Let $p(x) = x^n + r(x)$ be an irreducible polynomial of degree n over $GF(2)$. The simplest algorithm for field multiplication using polynomial representation is the shift-and-add method. This method is based on the observation that $a(x)b(x) = a_{n-1}x^{n-1}b(x) + \cdots + a_1 x b(x) + a_0 b(x)$. Thus, $x^i b(x)\, mod\, p(x)$ can be successively computed

Algorithm 2.1: Right-to-Left Shift-and-Add Field Multiplication

1 Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, a_i, b_i \in GF(2)$
2 Output: $c(x) = a(x)b(x)(mod\ p(x)) = \sum_{i=0}^{n-1} c_i x^i, c_i \in GF(2)$
3 IF $a_0 = 1$ THEN
4 $\quad c(x) \leftarrow b(x)$
5 ELSE
6 $\quad c(x) \leftarrow 0$
7 END IF
8 FOR $i = 0 \cdots n-1$ DO
9 $\quad b(x) \leftarrow b(x)x\ (mod\ (p(x))$
10 $\quad$ IF $a_i = 1$ THEN
11 $\quad\quad c(x) \leftarrow b(x) + c(x)$
12 $\quad$ END IF
13 END FOR
14 return $c(x)$

for all $1 \leq i \leq n-1$ and all the results are added for which $a_i = 1$. If $b(x) = b_{n-1}x^{n-1} + \cdots + b_2x^2 + b_1x + b_0$, then

$$\begin{aligned} b(x)x &= b_{n-1}x^n + \cdots + b_2x^3 + b_1x^2 + b_0x \\ &\equiv b_{n-1}r(x) + \left(b_{n-2}x^{n-1} + \cdots + b_2x^3 + b_1x^2 + b_0x\right)(mod\, p(x)) \end{aligned} \tag{2.14}$$

Therefore, $b(x)x(mod\, p(x))$ can be computed by a left-shift of the vector representation of $b(x)$, followed by the addition of $r(x)$ to $b(x)$, if the most significant bit b_{n-1} is 1. This algorithm is presented as Algorithm 2.1. The shift-and-add method is not particularly suitable for software implementations as the bit-wise shifts are costly to implement for processor architectures that are based on fixed-length words.

2.5.2 Polynomial Multiplication

This sub-section presents two fast algorithms available in the literature for performing finite field multiplications by multiplying the two polynomials of the finite field followed by modulo reduction using irreducible polynomial $P(x)$. The representation of polynomials in software is given as follows: Let w be the word-length of the processor

Algorithm 2.2: Right-to-Left Comb Method for Polynomial Multiplication

Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, a_i, b_i \in GF(2)$
Output: $c(x) = a(x)b(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in GF(2)$
$C \leftarrow 0$
FOR $i = 0 \cdots w-1$ DO
    FOR $j = 0 \cdots t-1$ DO
        IF $A[j][i] = 1$ THEN
            $C\{j\} = C\{j\} + B$
        END IF
    END FOR
    IF $i \neq w-1$ THEN
        $B \leftarrow Bx$
    END IF
END FOR
return C

(usually w is a multiple of 8) and $t = \lfloor (n/w) \rfloor$, where n is the degree of the polynomial. Thus, the vector $a = (a_{n-1}, \cdots, a_1, a_0)$ may be stored in an array of t w-bit words as shown

$$A = (A[t-1], \cdots, A[1], A[0]) \tag{2.15}$$

where, the rightmost bit is a_0 and the leftmost $(wt - n)$ bits are set to 0. The i^{th} bit of the j^{th} word is denoted by $A[j][i]$.

An efficient method for polynomial multiplication is presented i.e. the comb method. Here, multiplication is implemented in two separate steps: polynomial multiplication is performed to obtain a $2n$ bit-length polynomial followed by reducing it with the reduction polynomial. The right-to-left comb method for polynomial multiplication is based on the observation that if $b(x)x^i$ has been computed for some $i \in \{0, \cdots, w-1\}$, then $b(x)x^{wj+i}$ can be easily computed by appending j zero words to the right of the vector representation of $b(x)x^i$. Algorithm 2.2 processes the bits of the words of A from right to left.

Here, $C\{j\} = (C[n], \cdots, C[j+1], C[j])$. It can be observed that Algorithm 2.2 can also be improved. The left-to right comb method which processes the bits of a from left

Algorithm 2.3: Left-to-Right Comb Method for Polynomial Multiplication with Windows of Width w'

Input: $a(x) = \sum_{i=0}^{n-1} a_i x^i, b(x) = \sum_{i=0}^{n-1} b_i x^i, a_i, b_i \in GF(2)$
Output: $c(x) = a(x)b(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in GF(2)$
Precompute $B_u = u(x)b(x)$ for all polynomials $u(x)$ of degree at most $w' - 1$
FOR ALL $i = \frac{w}{w'} \cdots 0$ DO
    FOR ALL $j = 0 \cdots t - 1$ DO
        $u \leftarrow (u_{w'-1}, \cdots, u_1, u_0)$ where $u_k = A[j][w'i + k]$
        $C\{j\} \leftarrow C\{j\} + B_u$
    END FOR
    IF $i \neq 0$ THEN
        $C \leftarrow Cx^{w'}$
    END IF
END FOR
return C

to right is one such method given as follows:

$$a(x)b(x) = (\cdots((a_{n-1}b(x)x + (a_{n-2}b(x))x + (a_{n-3}b(x))x + \cdots + a_1 b(x))x + a_0 b(x) \quad (2.16)$$

It may be noted that the left-to right comb method can be considerably accelerated at the expense of some storage overhead by first computing $u(x)b(x)$ for all polynomials $u(x)$ of degree less than some fixed w', and then processing the bits of $A[j]$ one at a time. This modified method is called the left-to-right comb method with windows of width w'. The corresponding algorithm is presented as Algorithm 2.3.

The maximal degree of the output polynomial $c(x)$ is $(2n - 2)$. In some cases, the modular reduction required for field multiplication is done separately. In other cases, the irreducible polynomial $p(x)$ is included as an input to the algorithm and reduction is done mid-step. Such multiplications are also known as interleaved multiplication method. For example, Algorithm 2.3 can be modified to calculate $(u(x)b(x)(mod\, p(x)))$ using optimization, and will be presented in the following sub-section.

Algorithm 2.4: Bit-Level Modular Reduction

Input: $c(x) = \sum_{i=0}^{2n-2} c_i x^i, c_i \in GF(2), p(x) = x^n + r(x)$
Output: $c(x)\,(mod\ p(x))$
Precompute $u_k(x) = x^k r(x)$ for all $x^n r(x)$, $0 \leq k \leq w-1$
FOR ALL $i = (2n-2) \cdots n$ DO
    IF $c_i = 1$ THEN
        $j \leftarrow \frac{i-n}{w}$
        $k \leftarrow (i-n) - wj$
        $C\{j\} \leftarrow C\{j\} + u_k(x)$
    END IF
END FOR
return $(C[t-1], \cdots, C[0])$

2.5.3 Field Reduction

In this section, efficient reduction of polynomials of degree $(2n-2)$ is presented in Algorithm 2.4. Let $p(x) = x^n + r(x)$ be an irreducible polynomial of degree n over $GF(2)$ and $c(x) = \sum_{i=0}^{2n-2} c_i x^i$, where $c_i \in GF(2)$. The algorithm 2.4 is based on the observation that

$$\begin{aligned} x^n &\equiv r(x)\ (mod\, p(x)), \\ x^{n+k} &\equiv r(x)x^k\ (mod\, p(x)) \end{aligned} \tag{2.17}$$

and thus $c(x)$ can be computed as

$$\begin{aligned} c(x) &= c_{2n-2}x^{2n-2} + \cdots + c_1 x + c_0 \\ &\equiv (c_{2n-2}x^{2n-2} + c_n)r(x) + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0\ (mod\, p(x)) \end{aligned} \tag{2.18}$$

The reduction modulo $p(x)$ is done one bit at a time, starting from the leftmost bit as shown in Algorithm 2.4. In order to accelerate reduction, polynomials $x^k r(x)$ are precomputed, where $0 \leq k \leq w-1$. Moreover, it may be noted that Algorithm 2.4 involves operations performed at bit-level.

Multiplication in a finite field is simply the product of two field polynomials, modulo

reduced by $p(x)$, but there are some polynomials whose modulo reduction is more efficient than other polynomials.

Specifically, the reduction of polynomials modulo $p(x)$ is particularly efficient if $p(x)$ has a small number of terms. The irreducible polynomials with the least number of terms are the trinomials given by the expression $x^n + x^a + 1$. Thus, it is common practice to choose a trinomial for the field polynomial, provided that one exists. For example, Algorithm 2.4 can be accelerated even more when using trinomials, because the space requirements will be smaller and the additions involving $x^k r(x)$ become faster.

If an irreducible trinomial of degree n does not exist, then the next best polynomials are the pentanomials $x^n + x^a + x^b + x^c + 1$. In binary fields, for every n up to 1000, there exists either an irreducible trinomial or pentanomial of degree n. Such polynomials are widely recommended in all major standards, such as the IEEE Standard Specifications and National Institute of Standards and Technology (NIST).

2.6 Conclusion

In this chapter, a brief overview of the fundamental concepts of Groups, Rings, Fields, polynomial Rings, construction of finite fields, representation of finite fields using basis and the description of multiplication operation are presented. The next chapter presents the review of finite field multiplication architectures over $GF(2^m)$ available in the literature.

Chapter 3

Finite Field Multiplication Architectures over GF(2^m)

This chapter presents the details of different architectures proposed in the literature for finite field multiplications. Firstly, the sequential multipliers proposed in the literature over $GF(2^m)$ for irreducible polynomials are presented. Secondly, the systolic multipliers proposed in the literature over $GF(2^m)$ for irreducible polynomials are presented. Finally, the systolic multipliers proposed in the literature over $GF(2^m)$ for irreducible trinomials and pentanomials are presented. In addition, the performance improvements achieved by these multipliers in terms of area complexity, latency and critical path delay are also presented.

3.1 Sequential Multipliers over GF(2^m) for Irreducible Polynomials

Several sequential multipliers proposed in the literature for the finite field multiplications over $GF(2^m)$ for irreducible polynomials are reviewed and the performance of these multipliers are presented in Table 3.1. Hasan et al. [18] proposed a bit-serial sequential multiplier in 1998 having a maximum field dimension M to support multiplication over $GF(2^m)$ for any irreducible polynomial of degree m, where $1 < m \leq M$. The multiplier uses triangular basis representation in addition to polynomial basis and the input or output can be represented with respect to any of these two bases. The architecture proposed

Table 3.1: Performance comparison of sequential multipliers available in the literature.

Multipliers	#XOR	#AND	#MUX	#Registers	Latency	Critical Path Delay
[18]	$2m$	$3m$	m	m^2	$3m$	$(T_A + T_N + T_O)log_2m + T_X$
[19]	m	$2m$	$(m-1)^a + m^b$	$3m$	m	$T_X + 2T_A + T_N + (m+1)T_O$
[20]	m	m	$2m+1$	$3m$	$2m$	$T_X + T_A$
[21]	$2m$	$4m$	$m^a + m^c$	$3m$	m	$T_X + T_A$
[22]	$6m+18$	0	$14m+26$	$6m+7$	$m/4$	$4T_X + 2T_M$

[a]OR gates; [b]1-to-2 DMUX; [c]Inverter.

for this pipelined multiplier requires an area complexity of $2m$ exclusive-OR (XOR) gates, $3m$ AND gates, m 2-to-1 Multiplexers (MUX) and m^2 1-bit registers. The critical path delay of this architecture is given by the expression $((T_A + T_N + T_O)log_2m + T_X)$ with latency of $3m$ clock cycles, where T_A, T_N, T_O, T_X, T_M and T_{4M} represents the propagation delays of a 2-input AND gate, inverter, 2-input OR gate, 2-input XOR gate, 2-to-1 MUX and 4-to-1 MUX, respectively. These notations are used to compute the delays of all architectures presented in this thesis. This multiplier achieves reduction in area complexity compared to the sequential multipliers proposed in the literature. However, this multiplier requires the operands to be transformed from polynomial basis to triangular basis or vice-versa. Moreover, to support variable field size, serial-in serial-out registers,m number of m:1 multiplexers are used resulting in an increase of the clock cycles and area complexity required for performing multiplication.

A sequential polynomial basis multiplier over $GF(2^m)$ was proposed in 2003 by Kitsos et al. [19] based on an MSB-first technique having a maximum field dimension M, where $1 < m \leq M$. This multiplier requires an area complexity of m XOR gates, $2m$ AND gates, $(m-1)$ OR gates, m 1-to-2 De-Multiplexers (DMUX) and $3m$ 1-bit registers. The critical path delay is given by the expression $(T_X + 2T_A + T_N + (m+1)T_O)$ with latency of m clock cycles. The multiplier achieves reduction in area complexity compared to the sequential multiplier [18] due to MSB-first multiplication method and achieves low power consumption due to the gated clock technique. However, it's speed is reduced compared to the previous multipliers due to its sequential architecture.

A sequential multiplier designed based on a modified Montgomery multiplication method was proposed by Fournaris et al. [20] in 2008. This multiplier algorithm is derived

from a Montgomery multiplication algorithm available in the literature. This multiplier requires m XOR gates, m AND gates, $(2m+1)$ MUX and $3m$ 1-bit registers. The critical path delay is given by the expression (T_X+T_A) with latency of $2m$ clock cycles. This multiplier achieves reduction in area complexity compared to the sequential multipliers [18,19]. The latency of this multiplier is double than that of the multiplier [19], while requiring low latency compared to the sequential multiplier [18]. However, the Montgomery multipliers require additional computations to transform operands to Montgomery domain and vice-versa which increases it's area complexity compared to previous multipliers.

Zakerolhosseini et al. [21] proposed a bit-serial sequential multiplier in 2013 based on an efficient MSB-first method for different operand lengths with the condition $1 < m \leq M$, where M is the maximum order of the field. This architecture requires $2m$ XOR gates, $4m$ AND gates, m OR gates, m inverters and $3m$ 1-bit registers. The critical path delay is given by the expression (T_X+T_A) with latency of m clock cycles. This multiplier achieves reduction in area complexity compared to the sequential multiplier [18]. However, it has high area complexity compared to the sequential multiplier [19, 20]. In addition, this architecture also achieves improvement in latency compared to the sequential multipliers [18, 20], while requiring same latency as that of the sequential multiplier [19].

A sequential multiplier was proposed by Ho [22] in 2014 based on the condition $m \geq k_t + 4$, where k_t is the degree of the second leading term of the irreducible polynomial. This architecture requires $(6m + 18)$ XOR gates, $(14m + 26)$ Multiplexers and $(6m + 7)$ 1-bit registers. The critical path delay is given by the expression $(4T_X + 2T_M)$ with latency of $(m/4)$ clock cycles. This multiplier achieves reduction in latency compared to the sequential multiplier [18–21]. However, this architecture requires high area complexity compared to the sequential multipliers [19–21], while requiring low area complexity compared to the sequential multiplier [18]. Although this multiplier has achieved low latency, the area complexity is high which is not useful for low power and area-efficient applications.

3.2 Systolic Multipliers over GF(2^m) for Irreducible Polynomials

Several systolic multipliers proposed in the literature for finite field multiplication over $GF(2^m)$ based on irreducible polynomials are reviewed and the performance of these multipliers are presented in Table 3.2. In 1984, Yeh et.al. designed the first systolic multiplier [23] using a bit-parallel 2-dimensional design to achieve simple control, regular interconnection pattern, modular structure and concurrency of operations. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $7m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $3m$ clock cycles. However, this multiplier utilizes contraflowing datastreams in it's systolic structure which degrades the chip's cascadability and fault tolerance.

Wang et al. [24] realized a bit-parallel systolic structure in 1991 having unidirectional dataflow. This architecture requires $2m^2$ XOR gates, $2m^2$ AND gates and $7m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_{3X})$ with latency of $3m$ clock cycles. This multiplier achieves better chip cascadability and fault tolerance compared to the systolic multiplier [23], while requiring same area and delay complexities. Moreover, it requires higher critical path delay than the the systolic multiplier [23].

Wu et al. [25] realized a systolic multiplier in 1995 based on an MSB-first algorithm. This architecture requires $2m^2$ XOR gates, $(2m^2 - m)$ AND gates and $(8m^2 - 7m)$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(2m - 1)$ clock cycles. This multiplier achieves reduction in latency compared to the systolic multipliers [23, 24], while requiring approximately m^2 additional 1-bit registers. However, this multiplier is unable to reduce the critical path delay to a major extent.

A pipelined parallel-in/parallel-out structure was proposed by Jain et al. [26] in 1995. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $3m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(m + 1)$ clock cycles. This multiplier achieves low hardware complexity and low latency compared to the systolic multipliers [23–25]. Although this multiplier achieved reduction in the number of registers and latency compared to previous multipliers, it has not reduced the number of XOR gates and AND gates. Such reductions in the number of XOR gates and AND

Table 3.2: Performance comparison of systolic multipliers for irreducible polynomials available in the literature.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[23]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[24]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_{3X}$
[25]	$2m^2 - m$	$2m^2$	0	$8m^2 - 7m$	$2m - 1$	$T_A + T_X$
[26]	$2m^2$	$2m^2$	0	$3m^2$	$m + 1$	$T_A + T_X$
[27]	$2m^2$	$2m^2$	0	$3m^2$	$m + 1$	$T_A + T_X$
[28]	$2m^2$	$2m^2$	0	$4m^2$	$2m$	$T_A + T_X$
[29]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[30]	m	$2m^2 + 2m$	$(m^2/2)^a$	$6m^2 + 8m$	$3m/2$	$T_{4M} + T_X$
[31]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[32]	$2m^2 + 3m$	$(m^2 + m)^b$	0	$3m^2 + 4m$	$m + 1$	$T_A + T_{3X}$
[33]a	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	$3m$	$T_A + T_X$
[33]b	m^2	m^2	0	$5m^2$	$4m$	$T_A + T_X$
[34]	$2m^2$	$2m^2$	0	$3m^2$	m	$T_A + T_X$
[35]a	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	$3m$	$T_A + T_X$
[35]b	m^2	m^2	0	$5m^2$	$4m$	$T_A + T_X$
[36]	m	$2m + (m^2/2)^b$	$(m^2 + m/2)^a$	$7m^2$	$3m/2$	$T_{4M} + T_{3X}$
[20]	$m^2 - m + 1$	$m^2 - 1$	$2m^2 + m - 3$	$2m^2 - m$	$2m$	$T_A + T_X$
[37]	$2m^2$	$2m^2$	$2m^2 + m - 3$	$7m^2$	$3m$	$T_A + T_X$
[38]	$2m^2 + 2m$	$2m^2 + 3m$	0	$3m^2 + 4m$	$\lfloor m/2 \rfloor + 1$	$T_A + T_X$

[a]4-to-1 MUX; [b]3-input XOR gate.

gates can be observed in the multipliers proposed later.

In 1998, Jain et.al. [27] developed a pipelined semi-systolic structure based on an LSB (Least Significant Bit)-first algorithm. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $3m^2$ 1-bit registers. The critical path delay is given by the expression (T_A+T_X) with latency of $(m+1)$ clock cycles. This multiplier achieves low area complexity and low latency compared to the systolic multipliers [23–25], while requiring same area complexity and same latency compared to the systolic multiplier [26]. It can be observed that this multiplier has not achieved any change in the area complexity or delay compared to the previous multiplier [26].

In 1998, Koć et al. [28] introduced a Montgomery based multiplication method. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $4m^2$ 1-bit registers. The critical

path delay is given by the expression $(T_A + T_X)$ with latency of $2m$ clock cycles. This architecture achieves reduction in area complexity compared to the systolic multipliers [23–25], while requiring more area complexity compared to the systolic multipliers [26,27]. Moreover, it achieves reduction in latency compared to the systolic multipliers [23, 24], while requiring same latency compared to the systolic multiplier [25] and high latency compared to the systolic multipliers [26, 27].

A systolic design based on LSB-first method was proposed by Kwon et.al. [29] in 2003. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $7m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $3m$ clock cycles. This multiplier requires same area complexity and latency as that of the systolic multipliers [23, 24]. Moreover, it requires high area complexity and latency compared to the systolic multipliers [26–28]. However, it achieves low area complexity and high latency compared to the systolic multiplier [25]. It can be observed that this multiplier does not show significant improvement in area complexity or delay than the previous multipliers.

In 2006, Lee et.al. [30] proposed a systolic multiplexer-based structure using iterative arrays based on modified Booth's algorithm. This multiplier requires $(m^2 + 2m)$ XOR gates, m AND gates, $(m^2/2)$ 4:1 Multiplexers and $(6m^2 + 8m)$ 1-bit registers. The critical path delay is given by the expression $(T_{4M} + T_X)$ with latency of $(3m/2)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multipliers [23–25, 29], while requiring low latency compared to the systolic multipliers [23–25, 28, 29]. However, it requires high area complexity due to extra multiplexers and also requires high latency compared to the systolic multipliers [26, 27].

Kwon et.al. [31] developed a two-dimensional systolic multiplier in 2006 based on an LSB-first algorithm. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $7m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $3m$ clock cycles. This multiplier achieves same area complexity and latency as that of the systolic multipliers [23, 24, 29]. Moreover, it requires more area complexity and higher latency compared to the systolic multipliers [26–28, 30]. However, it achieves low area complexity while requiring high latency compared to the systolic multiplier [25]. It can be observed that this multiplier does not show significant improvement in area complexity or delay than previous multipliers.

A systolic multiplier was realized by Chiou et.al. [32] in 2006 based on a time-independent Montgomery multiplication algorithm. This multiplier requires $(m^2 + m)$ 3-input XOR gates, $(2m^2 + 3m)$ AND gates and $(3m^2 + 4m)$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_{3X})$ with latency of $(m + 1)$ clock cycles. This multiplier achieves reduction in area complexity and latency compared to the systolic multipliers [23–25, 28–31]. Moreover, it requires same area complexity and latency as that of the systolic multipliers [26, 27] with high critical path delay which effectively reduces its frequency of operation.

Two multipliers were proposed by Lee et.al. [33] in 2006 using MSB-first time-independent and time-dependent algorithms based on a conventional interleaved multiplication and a folded technique. The time-independent multiplier requires $(m^2 + 2m)$ XOR gates, m^2 AND gates and $(4m^2 + 3m)$ 1-bit registers and the time-dependent multiplier requires m^2 XOR gates, m^2 AND gates and $5m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ for both the designs with latency of $3m$ clock cycles for the time-independent multiplier and latency of $4m$ clock cycles for the time-dependent multiplier. The time-independent multiplier achieves reduction in area complexity compared to the systolic multipliers [23–32]. However, it requires higher latency compared to the systolic multipliers [25–28, 30, 32], while requiring same latency as that of the systolic multipliers [23, 24, 29, 31]. The time-dependent multiplier achieves reduction in area complexity compared to the systolic multipliers [23–25, 29–31]. However, it requires high latency compared to the systolic multipliers [23–32].

In 2007, a linear two-dimensional systolic structure based on a linear feedback shift register (LFSR) was proposed by Chiou et.al. [34]. This multiplier requires $2m^2$ XOR gates, $2m^2$ AND gates and $3m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of m clock cycles. This multiplier achieves reduction in area complexity compared to the systolic multipliers [23–25, 28–33], while requiring same area complexity as that of the systolic multipliers [26, 27] and requiring high area complexity compared to the systolic multipliers [33]. Moreover, it achieves low latency compared to the systolic multipliers [23, 24, 26–33]. However, this multiplier does not show significant improvement in area complexity compared to the previous multipliers.

Lee [35] proposed a time-independent and time-dependent algorithm in 2008 and

realized two bit-parallel systolic multipliers by employing these algorithms. The time-independent multiplier requires (m^2+2m) XOR gates, m^2 AND gates and ($4m^2$+3m) 1-bit registers and the time-dependent multiplier required m^2 XOR gates, m^2 AND gates and $5m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ for both the designs with latency of $3m$ clock cycles for the time-independent multiplier and latency of $4m$ clock cycles for the time-dependent multiplier. This time-independent multiplier achieves reduction in area complexity compared to the systolic multipliers [23–34] and requires same area complexity as that of the systolic multiplier [33]. However, it requires high latency compared to the systolic multipliers [25–28, 30, 32, 34], while requiring same latency as that of the systolic multipliers [23, 24, 29, 31, 33] and low latency compared to the systolic multiplier [33]. The time-dependent multiplier achieves reduction in area complexity compared to the systolic multipliers [23–25, 29–31, 33], while requiring high area complexity compared to the systolic multipliers [26–28, 32–34]. However, it requires high latency compared to the systolic multipliers [23–34].

In 2008, Lee [36] developed a multiplexer-based systolic architecture using an algorithm that employs cut-set systolization and modified Booths recoding technique. This multiplier requires $2m$ XOR gates, m AND gates, $(m^2/2)$ 3-input XOR gates, $(m^2 + m/2)$ 4:1 Multiplexer and $(6m^2 + 8m)$ 1-bit registers. The critical path delay is given by the expression $(T_{4M} + T_{3X})$ with latency of $(3m/2)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multipliers [23–25, 29, 31] while requiring high area complexity compared to the systolic multipliers [26–28, 30, 32–35]. It achieves low latency compared to the systolic multipliers [23–25, 28, 29, 31, 33, 35], while requiring high latency compared to the systolic multipliers [26, 27, 32, 34] and requires same latency as that of the previous multiplier [30]. Although, this multiplier reduces the AND gate count, 4:1 multiplexers and 3-input XOR gates were used which increased the area complexity compared to previous multipliers.

Fournaris et al. [20] proposed a systolic Montgomery multiplier from an optimized Montgomery multiplication algorithm. This multiplier requires $(m^2 - 1)$ XOR gates, $(m^2 - m + 1)$ AND gates, $(2m^2 + m - 3)$ Multiplexers and $(2m^2 - m)$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $2m$ clock cycles. This multiplier achieves reduction in area complexity compared to the systolic

multipliers [23–32, 36]. It also achieves low latency compared to the systolic multipliers [23, 24, 29, 31, 33, 35], while requiring high latency compared to the systolic multipliers [26, 27, 30, 32, 34, 36] and same latency as that of the systolic multipliers [25, 28]. This multiplier has high latency when compared to previous multipliers but achieved significant improvement in area complexity.

A systolic multiplier is proposed by Kwon et.al. [37] in 2009 based on an LSB-first algorithm. This multiplier require $2m^2$ XOR gates, $2m^2$ AND gates, $(2m^2 + m - 3)$ Multiplexers and $7m^2$ 1-bit registers. The critical path delay is given by the expression (T_A+T_X) with latency of $3m$ clock cycles. This multiplier requires same area complexity as that of the systolic multipliers [23–25, 29, 31] and requires high area complexity compared to the systolic multipliers [20, 26–28, 30, 32–36]. Moreover, it has same latency as that of the systolic multipliers [23, 24, 29, 31, 33, 35] and requires high latency compared to the systolic multipliers [20, 25–28, 30, 32, 34, 36]. On the whole, this multiplier has very high area complexity and also more delay compared to previous multipliers.

In 2014, Kim et al. [38] presented a cellular array multiplier based on Montgomery multiplication using a qualified Montgomery factor to divide the algorithm into two parts in order to reduce the delay. This multiplier requires $(2m^2+3m)$ XOR gates, $(2m^2+2m)$ AND gates and $(3m^2 + 4m)$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $((m/2) + 1)$ clock cycles. This multiplier achieves reduction in area complexity compared to the systolic multipliers [23–25, 28–31, 33, 35–37], while requiring high area complexity compared to the systolic multipliers [20, 26, 27, 33–35] and same area complexity as that of the systolic multiplier [32]. Even though this multiplier shows significant improvement in latency, the area complexity is high which needs to be reduced further in order to be used in area and power constrained applications.

3.3 Systolic Multipliers over GF(2^m) for Irreducible Trinomials

Irreducible polynomials can be classified into equally spaced polynomials, all-one polynomials, trinomials, and pentanomials. The equally spaced polynomials and all-one polynomials are not widely used due to their scarcity. However, up to 5148 irreducible trinomials were identified for field orders of $m \leq 10,000$ [39] which is approximately

Table 3.3: Performance comparison of systolic multipliers for irreducible trinomials available in the literature.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[40]	m^2	m^2+lm	0	$4m^2+2lm$	$m+l+1$	T_A+T_X
[41]	m^2	m^2+m-1	0	$3m^2+2m-2$	$2m-1$	T_A+T_{3X}
[42]	m^2+N_V	m^2+m	0	$4m^2+m$	$m+1$	T_A+T_X
[43]	m^2	m^2+m-1	0	$2m^2$	$2m-1$	T_A+T_X
[44]	m^2	m^2+m	m	$3m^2+m$	$m+n$	T_A+T_X
[45]	m^2	m^2+m-1	0	$2m^2$	$2m-1$	T_A+T_X
[46]	$(m^2)^a$	m^2-1	$(m^2-2m)^b$	$2m^2-m$	m	$T_{NA}+T_X$
[47]	m^2	m^2+m	m^2	$3.5m^2+3m$	$m+2$	T_M+T_X
[48]	m^2	m^2+m	m	$2m^2+3m$	$m+1$	$T_A+T_X+T_M$
[49]	$(m^2)^a$	$1.5m^2+0.5m$	$(1.5m^2-2.5m+3)^b$	$1.5m^2+2m-1$	$m+2$	$T_{NA}+T_X$

[a]NAND gates; [b]Inverter.

half of the m values. Moreover, trinomials are recommended by the National Institute of Standards and Technology (NIST) for use in cryptographic applications since they allow realisation of efficient hardware structures with low area complexity and low power consumption. Therefore, several systolic multipliers are proposed in the literature over $GF(2^m)$ for irreducible trinomials and the performance of these multipliers is presented in Table 3.3.

Lee [40] presented a systolic multiplier for trinomials in 2003 based on the condition $gcd(m,n)=1$, where n is the degree of the second term of the trinomial. This multiplier requires (m^2+lm) XOR gates, m^2 AND gates and $(4m^2+2lm)$ 1-bit registers. The critical path delay is given by the expression (T_A+T_X) with latency of $(m+l+1)$ clock cycles, where l is the number of reductions that can be applied on the polynomial of degree $(2m-2)$ which is obtained after the polynomial multiplication step and is given by the expression $l=\left\lfloor\frac{m-2}{m-n}\right\rfloor+1$. This multiplier achieves low area complexity and low latency compared to the generic polynomial multipliers proposed in the literature. However, this multiplier had high area complexity and more latency compared to the multipliers for trinomials and significant reduction in area complexity and delay were necessary.

In 2003, Lee [41] introduced the importance of trinomials in finite field multiplications and proposed a low-complexity systolic multiplier for irreducible trinomials. This multiplier requires (m^2+m-1) XOR gates, m^2 AND gates and $(3m^2+2m-2)$ 1-bit

registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(2m-1)$ clock cycles. This multiplier achieves reduction in area complexity but has high latency compared to the systolic multiplier [40].

A compact systolic multiplier for trinomials was designed by Lee et.al. [42] in 2005 based on Montgomery multiplication algorithm employing a Hankel matrix-vector representation. This multiplier requires $(m^2 + m)$ XOR gates, $(m^2 + N_V)$ AND gates and $(4m^2 + m)$ 1-bit registers, where N_V is given by the expression $\frac{(m-n)(m-n-1)+n(n+1)}{2}$ and n represents n-term Hankel matrix-vector representation. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(m+1)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multipliers [40, 41]. However, it has high latency compared to the systolic multiplier [40], while requiring same latency as that of the systolic multiplier [41].

A systolic multiplier for trinomials based on a transformation method applied on conventional Montgomery multiplication was proposed by Lee et.al. [43] in 2006. This multiplier requires $(m^2 + m - 1)$ XOR gates, m^2 AND gates and $2m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(2m - 1)$ clock cycles. This multiplier requires similar latency compared to the systolic multiplier [40], while requiring same latency as that of the systolic multipliers [41,42]. However, it requires significantly high area complexity compared to the systolic multipliers [40–42] .

In 2007, Lee et.al. [44] proposed a systolic multiplier for trinomials by employing Hankel matrix-vector multiplications. This multiplier requires $(m^2 + m)$ XOR gates, m^2 AND gates, m Multiplexers and $(3m^2 + m)$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(m + n)$ clock cycles, where n represents n-term Hankel matrix-vector representation. This architecture achieves reduction in area complexity compared to the systolic multipliers [40, 43], while requiring same area complexity as that of the systolic multiplier [41]. However, it requires high area complexity compared to the systolic multiplier [42]. In addition, this multiplier achieves low latency compared to the systolic multipliers [41–43], while requiring same latency as that of the systolic multiplier [40]. Overall, this multiplier has low lantecy but significantly high area complexity than the previous multipliers.

Lee et.al. [45] proposed a scalable and systolic multiplier for trinomials in 2007.

This is designed employing the Hankel matrix-vector multiplications on the Montgomery multiplication algorithm. This multiplier requires $(m^2 + m - 1)$ XOR gates, m^2 AND gates and $2m^2$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X)$ with latency of $(2m - 1)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multipliers [40, 41, 43, 44], while requiring same area complexity as that of the systolic multiplier [42]. It requires high latency compared to the systolic multipliers [40, 44], while requiring same latency as that of the systolic multipliers [41–43].

In 2008, Meher [46] proposed a pipelined systolic multiplier for trinomials employing a suitable cut-set retiming technique. This multiplier requires $(m^2 - 1)$ XOR gates, m^2 NAND gates, $(m^2 - 2m)$ Inverters and $(2m^2 - m)$ 1-bit registers. The critical path delay is given by the expression $(T_{NA} + T_X)$ with latency of m clock cycles. This multiplier achieves low area complexity and low latency compared to the systolic multipliers [40–45]. The latency of this multiplier is low but the area complexity is moderate and can be improved further.

A systolic architecture designed by employing the Toeplitz matrix-vector representation on Montgomery multiplication was proposed by Lee [47] in 2008. This multiplier requires $(m^2 + m)$ XOR gates, m^2 AND gates, m^2 Multiplexers and $(3.5m^2 + 3m)$ 1-bit registers. The critical path delay is given by the expression $(T_M + T_X)$ with latency of $(m + 2)$ clock cycles. This multiplier achieves reduction in area complexity compared to the systolic multipliers [40, 43] while requiring high area complexity compared to the systolic multipliers [41, 42, 44–46]. Moreover, it achieves low latency compared to the systolic multipliers [41–43, 45] while requiring same latency as that of the systolic multipliers [40, 44, 46]. On the whole, this multiplier utilizes extra multiplexers which increased it's area complexity significantly compared to the previous multipliers.

Chiou et.al. [48] proposed a systolic multiplier for trinomials in 2011, based on Mastrovito multiplication method. This multiplier requires $(m^2 + m)$ XOR gates, m^2 AND gates, m Multiplexers and $(2m^2 + 3m)$ 1-bit registers. The critical path delay is given by the expression $(T_A + T_X + T_M)$ with latency of $(m+1)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multipliers [40, 41, 43, 44, 47], while requiring same area complexity as that of the systolic multiplier [42, 45, 46]. Moreover, it achieves low latency compared to the systolic multipliers [41–43, 45], while requiring

same latency as that of the systolic multipliers [40, 44, 46, 47]. This multiplier had high critical path delay and the area complexity is also moderately high compared to previous multipliers.

Bayat-Sarmadi and Farmani [49] proposed a systolic multiplier for trinomials in 2015 based on Montgomery multiplication algorithm. This multiplier requires $(1.5m^2 + 0.5m)$ XOR gates, m^2 NAND gates, $(1.5m^2 - 2.5m + 3)$ Inverters and $(1.5m^2 + 2m - 1)$ 1-bit registers. The critical path delay is given by the expression $(T_{NA} + T_X)$ with latency of $(m + 2)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multipliers [40, 41, 43, 44, 47, 48], while requiring same area complexity as that of the systolic multiplier [42, 45, 46]. Moreover, it achieves low latency compared to the systolic multipliers [41–43, 45], while requiring same latency as that of the systolic multipliers [40, 44, 46–48]. This multiplier has moderate area complexity compared to previous multipliers which needs to be reduced further in order to be used in area and power constrained applications.

3.4 Systolic Multipliers over GF(2^m) for Irreducible Pentanomials

Pentanomials are special class of irreducible polynomials that are recommended by NIST to be used for defining a binary field when trinomials in that field are absent. Along with trinomials, pentanomials also have a vital role in the design of some cryptographic algorithms because efficient hardware structures can be realized with low area complexity and low power consumption. Hence, some systolic multipliers are proposed in the literature over $GF(2^m)$ for irreducible pentanomials and the performance of these multipliers are presented in Table 3.4.

A systolic multiplier for pentanomials was presented by Lee [47] in 2008, based on Montgomery multiplication method using Toeplitz matrix-vector representation. This multiplier requires (m^2+m) XOR gates, m^2 AND gates, m^2 Multiplexers and $(3.5m^2+3m)$ 1-bit registers. The critical path delay is given by the expression $(T_M + T_X)$ with latency of $(m + 2)$ clock cycles. This multiplier achieves reduction in area complexity compared

Table 3.4: Performance comparison of systolic multipliers for irreducible pentanomials available in the literature.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[47]	m^2	m^2+3m+2	m	$3.5m^2+7m$	$m+4$	T_X
[50]	$(m^2)^a$	m^2+2m-1	$(m^2)^b$	$2m^2-2m$	m	$T_{NA}+T_X$
[51]	m^2	m^2+2m	$(m^2)^b$	$2m^2$	$\frac{m}{2}+2$	$2T_X$
[52]	$(m^2)^a$	$2m+2lm$ $+2l+2$	$(m^2)^c$	$3m^2-2m-2lm$ $-2l-2$	$\frac{m}{2l+2}+1$ $+log_2(2l+2)$	$T_{NA}+T_{XN}$

[a]NAND gates; [b]Inverter; [c]XNOR gate.

to multiplier architectures for generic irreducible polynomials. However, it had very high area complexity compared to the multipliers for pentanomials.

In 2009, Meher [50] proposed a systolic multiplier for pentanomials using an efficient modular reduction algorithm to realize efficient Reed-Solomon codecs. This multiplier requires (m^2+2m-1) XOR gates, m^2 NAND gates, m^2 inverters and $(2m^2-2m)$ 1-bit registers. The critical path delay is given by the expression $(T_{NA}+T_X)$ with latency of m clock cycles. This multiplier achieves reduction in area complexity and reduction in latency of 4 clock cycles compared to the systolic multiplier [47]. However, the reduction in area complexity is not much significant and further reduction can be made to the area complexity.

A digit-serial systolic multiplier for pentanomials was designed by Xie et.al. [51] in 2012, based on a novel decomposition technique to divide the multiplier into two parallel units. This multiplier requires (m^2+2m) XOR gates, m^2 AND gates, m^2 Inverters and $2m^2$ 1-bit registers. The critical path delay is given by the expression $2T_X$ with latency of $((m/2)+2)$ clock cycles. This multiplier achieves low area complexity compared to the systolic multiplier [47], while requiring slightly high area complexity compared to the systolic multiplier [50]. Moreover, it also achieves about 50% reduction in latency compared to the systolic multipliers [47, 50]. The improvement in delay obtained due to the digit-serial design causes a penalty on area complexity, which is not suitable for area and power constrained applications.

A digit-serial systolic Montgomery multiplier for pentanomials by employing a novel pre-computation technique was proposed by Xie et.al. [52] in 2013. This multiplier re-

quires $(2m + 2lm + 2l + 2)$ XOR gates, m^2 XNOR gates, m^2 NAND gates and $(3m^2 - 2m - 2lm - 2l - 2)$ 1-bit registers, where l is the decomposition factor given by the expression $l = min\{m - k_1, k_1 - k_2, k_2 - k_3, k_3\}$ for pentanomials of the form $(x^m + x^{k_1} + x^{k_2} + x^{k_3} + 1)$. The critical path delay is given by the expression $(T_{NA} + T_{XN})$ with latency of $((m/(2l + 2)) + 1 + log_2(2l + 2))$ clock cycles. This multiplier achieves low area complexity compared to the systolic multiplier [47], while requiring high area complexity compared to the systolic multipliers [50, 51]. Moreover, it also achieves reduction in latency compared to the systolic multipliers [47, 50, 51]. Although it achieves reduction in delay due to latency improvement, the area complexity is high for this digit-serial design.

3.5 Conclusion

In this chapter, a survey of different architectures of finite field multipliers available in the literature and the improvements achieved is presented. The next chapter presents the design of the proposed sequential multiplier for irreducible polynomials.

Chapter 4

Low-power and Area-Efficient Sequential Multipliers over Polynomial Basis

This chapter presents an interleaved multiplication algorithm derived from a conventional interleaved multiplication algorithm available in the literature. A sequential multiplier architecture over $GF(2^m)$ for irreducible polynomials is designed based on the proposed algorithm. The performance of the proposed sequential multiplier architecture is computed analytically and compared with the multiplier architectures available in the literature. In addition, the analytical results are also verified by implementing the proposed architecture on Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) technologies and the results are compared with the existing architectures available in the literature. Moreover, the Verilog models of two cryptographic algorithms, AES and Twofish, are developed employing the proposed sequential multiplier and the multipliers available in the literature. These Verilog models are implemented on FPGA to compute the performance improvement achieved by the proposed multiplier compared to the sequential multipliers available in the literature.

4.1 Introduction

Finite field arithmetic over $GF(2^m)$ is used in a variety of applications such as cryptography, coding theory and computer algebra. Finite field multiplication is an important and complex operation mainly used in cryptographic applications. Many algorithms and

architectures are proposed in the literature to realize efficient multiplication operation in both hardware and software and the implementation of these Cryptographic algorithms in hardware achieves better security and low resource utilizations. In addition, the design of finite field multipliers using sequential architecture allows further reduction in area complexity and power consumption as the computations are carried out iteratively using the same hardware. In spite of these sequential multipliers being slow, they are used in several applications such as medical implants, wireless sensors, Internet of Things etc., due to their extremely low area and low power requirements. All the sequential architectures proposed in the literature [18–22] were aimed at reducing the area complexity or delay when compared to the previous multipliers. This reduction in area complexity & power consumption or the delay is essential to fulfil the ever increasing demand to miniaturize the VLSI device, to reduce it's power consumption and to increase it's speed. Hence, there is a never ending need to reduce the area complexity & power consumption and also to increase the speed of any VLSI hardware device.

In this work, a modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm to realize the proposed sequential multiplier. Subsequently, an efficient sequential polynomial basis multiplier, that supports multiplication of any two arbitrary finite field elements over $GF(2^m)$ for generic irreducible polynomials, is designed based on the proposed algorithm. The area complexity and delay of the proposed sequential multiplier over $GF(2^m)$ is estimated and its performance is compared with existing sequential multipliers [18–22]. It is observed that the proposed sequential multiplier achieves reduction in area complexity and area-delay product (ADP) over the existing sequential multipliers verified for a field of order $m = 163$. The proposed multiplier and some existing multipliers are implemented using ASIC and FPGA technologies and the implementation results shows that the proposed sequential multiplier achieves reduction in area complexity, power consumption, ADP and power-delay product (PDP) over existing multipliers.

In addition, a sequential multiplier architecture over $GF(2^8)$ is derived from the proposed sequential multiplier architecture over $GF(2^m)$ as an example. The area complexity and delay of the proposed multiplier are estimated and performance comparison with existing sequential multipliers [18–22] is also presented. The proposed architecture

achieves reduction in area complexity and ADP over the best of existing multipliers for $m = 8$. In order to evaluate the performance of the proposed multiplier in a cryptographic application, the Verilog models of Advanced Encryption Standard (AES) and Twofish algorithms are developed employing the proposed sequential multiplier and other multipliers available in the literature. These models are implemented on FPGA device and the device utilization summary shows that the proposed multiplier achieves low area complexity, low power consumption, less ADP and PDP compared to the existing multipliers.

4.2 Proposed Interleaved Multiplication Algorithm

$GF(2^m)$ is an extension field of $GF(2)$ having an m-dimensional vector space over it, where $GF(2)$ is a binary field having only two elements $\{0, 1\}$ [53]. The addition and subtraction operations can be performed by the logical exclusive-OR (XOR) operation and the multiplication operation can be performed by the logical AND operation over $GF(2)$. However, the multiplication over $GF(2^m)$ is performed by multiplying the two polynomials and modular reduction of the result using the irreducible polynomial.

Definition 4.1. Let $T(x)$ be the irreducible polynomial of degree m over $GF(2)$ which defines the field $GF(2^m)$. Then,

$$T(x) = x^m + t_{m-1}x^{m-1} + \cdots + t_1x + t_0 \tag{4.1}$$

where, $t_0, t_1, \cdots, t_{m-1} \in GF(2)$ are the coefficients of the irreducible polynomial $T(x)$.

Definition 4.2. Let $\alpha \in GF(2^m)$ be a root of $T(x)$. Then the following set constitutes the polynomial basis in $GF(2^m)$

$$\omega = \left\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\right\} \tag{4.2}$$

Definition 4.3. In the polynomial basis ω, the elements of $GF(2^m)$ are polynomials of degree $\leq m - 1$ over $GF(2)$ and the set of all polynomial elements over $GF(2^m)$ is represented as

$$GF(2^m) = \{f(x) \mid f(x) = a_{m-1}x^{m-1} + \cdots + a_2x^2 + a_1x + a_0\} \tag{4.3}$$

where, $a_i \in GF(2)$; for $i = 0, 1, 2, \cdots, m-1$.

Definition 4.4. Let $f(x) = a_{m-1}x^{m-1} + \cdots + a_2x^2 + a_1x + a_0$ be a polynomial in $GF(2^m)$, then the binary representation of this polynomial is

$$a = (a_{m-1}, \cdots, a_2, a_1, a_0) \tag{4.4}$$

where, $a_i \in GF(2)$ and a_{m-1} is the most significant bit (MSB) and a_0 is the least significant bit (LSB); for $i = 0, 1, 2, \cdots, m-1$. Let the polynomials $A(x), B(x)$ and $T(x)$ be two polynomials and the irreducible polynomial, respectively and $D(x)$ be the final product polynomial. Then,

$$D(x) = (A(x) \times B(x)) \; mod \; T(x) \tag{4.5}$$

Polynomial multiplication: The product of $A(x)$ and $B(x)$, each of degree at most $m-1$, results in an intermediate polynomial given by

$$\begin{aligned} C(x) &= A(x) \times B(x) \\ &= (a_0 + a_1x + \cdots + a_{m-1}x^{m-1}) \times (b_0 + b_1x + \cdots + b_{m-1}x^{m-1}) \\ &= c_0 + c_1x + \cdots + c_{2m-2}x^{2m-2} \end{aligned} \tag{4.6}$$

Modular reduction: The intermediate polynomial $C(x)$ of degree at most $(2m-2)$ is modular reduced by a degree m irreducible polynomial $T(x)$ resulting in the polynomial $D(x)$ of degree at most $(m-1)$, which is the final result of the multiplication operation.

$$\begin{aligned} D(x) &= C(x) \times T(x) \\ &= (c_0 + c_1x + \cdots + c_{2m-2}x^{2m-2}) \times (t_0 + t_1x + \cdots + t_{m-1}x^{m-1} + x^m) \\ &= d_0 + d_1x + \cdots + d_{m-1}x^{m-1} \end{aligned} \tag{4.7}$$

Thus the multiplication of two polynomials of degree $(m-1)$ results in a polynomial of degree $(m-1)$ such that the resultant polynomial resides in the given field $GF(2^m)$.

Let $a = (a_{m-1}, \cdots, a_1, a_0)$ and $b = (b_{m-1}, \cdots, b_1, b_0)$ be the binary representations of the two elements, $A(x)$ and $B(x)$, over $GF(2^m)$, respectively. Let $t = (t_{m-1}, \cdots, t_1, t_0)$ be the binary representation of the field defining irreducible polynomial $T(x)$ of degree at most m, and let $p = (p_{m-1}, \cdots, p_1, p_0)$ be the accumulator of the intermediate calculations.

The derivation of the proposed algorithm from a conventional interleaved multiplication algorithm [54] is given as follows:
The two arbitrary elements $A(x)$ and $B(x)$ in $GF(2^m)$ can be expressed as

$$A = \sum_{i=0}^{m} a_i x^i \qquad B = \sum_{i=0}^{m} b_i x^i \tag{4.8}$$

Let $C(x) \in GF(2^m)$ be the product polynomial of the two elements $A(x)$ and $B(x)$.

$$\begin{aligned} C(x) &= A(x) \times B(x) \\ &= A(x) \times \sum_{i=0}^{m} b_i x^i \\ &= b_0 A(x) + b_1 x A(x) + b_2 x^2 A(x) + \cdots + b_{m-1} x^{m-1} A(x) \end{aligned} \tag{4.9}$$

It may be observed from Eqn. (4.9) that $C(x)$ is the summation of the multiplication result of b_i and $A(x)x^i$; for all $i = 0, 1, \cdots, m-1$ i.e. the entire summation can be carried out in m iterations. $A(x)x^i$ is calculated by the modular reduction step which is then multiplied with b_i using AND operation; for all $i = 0, 1, \cdots, m-1$. Contrary to the generic case of summation by addition, the exclusive-OR (XOR) operation is considered for the summation of each $b_i A(x)x^i$; for all $i = 0, 1, \cdots, m-1$, since the addition is simply an XOR operation over $GF(2)$. Hence, the calculation of $C(x)$ in Eqn. (4.9) is transformed as Steps 3, 4, 8 in Algorithm 4.1. Here, $p = (p_{m-1}, \cdots, p_1, p_0)$ acts as the accumulator of $A(x)x^i$ and is initialized to zero at the beginning of each multiplication operation.

The modular reduction of the conventional interleaved multiplication algorithm [54] is performed as shown

$$A(x) = \left(A(x) \times x^i\right) mod\ T(x) \tag{4.10}$$

Eqn. (4.10) is evaluated for each i as follows
For $i = 0$:

$$\begin{aligned} C(x) &= A(x)\ mod\ T(x) \\ &= \left(a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}\right)\ mod\ (t_0 + t_1 x + \cdots + t_{m-1} x^{m-1} + x^m) \end{aligned} \tag{4.11}$$

A degree m polynomial cannot modulo divide a degree $(m-1)$ polynomial. Hence, this step can be skipped.

For $i = 1$:

$$\begin{aligned} C(x) &= (A(x) \times x) \ mod\ T(x) \\ &= \left(a_0 + a_1x^2 + \cdots + a_{m-1}x^m\right) \ mod\ (t_0 + t_1x + \cdots + t_{m-1}x^{m-1} + x^m) \\ &= a_{m-1}t_0 + (a_{m-1}t_1 + a_0)x + (a_{m-1}t_2 + a_1)x^2 + \cdots + (a_{m-1}t_{m-1} + a_{m-2})x^{m-1} \end{aligned} \tag{4.12}$$

It is revealed from Eqn. (4.12) that the modular reduction is reduced to the summation of $a_{m-1}T(x)$ and $A(x)x^i$. The $A(x)x^i$ is computed by left shifting $A(x)$ by i times; for all $i = 0, 1, 2, \cdots, m-1$. The $a_{m-1}T(x)$ is computed by bit-wise AND operation of a_{m-1} with the binary representation of $T(x)$ i.e. $(t_{m-1}, \cdots, t_1, t_0)$. The summation is carried out in m iterations using the XOR operation. Therefore, the modular reduction step can be transformed as Steps 5, 6, 7 as shown in Algorithm 4.1.

Both the polynomial multiplication and modular reduction steps occur simultaneously resulting in an interleaved algorithm. The two modified equations, Eqn. (4.9) and (4.12), results in the derivation of Algorithm 4.1.

Algorithm 4.1: Proposed interleaved multiplication algorithm over $GF(2^m)$

1 Initialization: $p = 0, counter = 0$
2 FOR $counter = 0$ TO 7 DO
3 $\quad a = a\ \&\ b_0$
4 $\quad p = p \oplus a$
5 $\quad t = t\ \&\ a_{m-1}$
6 $\quad a = a \ll 1$
7 $\quad a = a \oplus t$
8 $\quad b = b \gg 1$
9 END FOR

4.3 Proposed Sequential Multiplier Architecture over GF(2^m) for Irreducible Polynomials

This sub-section presents the design of the proposed sequential multiplier architecture over $GF(2^m)$ for irreducible polynomials. The estimations of area complexity

and delay of this architecture are computed analytically and compared with the existing multipliers available in the literature. The functionality of the proposed architecture is implemented using FPGA and ASIC technologies. These analytical and implementation results of the proposed architectures and the architectures available in the literature are also presented in the following sub-sections.

4.3.1 Design of Proposed Sequential Multiplier Architecture over GF(2^m) for Irreducible Polynomials

The design of the proposed sequential multiplier over finite fields of an arbitrary field order m is presented in this sub-section. Fig. 4.1 shows the block diagram of the proposed sequential polynomial basis multiplier over $GF(2^m)$ realized using the proposed algorithm (Section 4.2). This architecture consists of two main modules A and B and three m-bit registers. The multiplier takes one m-bit input t; where, t denotes the binary representation of the irreducible polynomial. Module A computes the polynomial multiplication and module B computes the modular reduction. The logic diagrams of the modules A and B are shown in Fig. 4.2(a) and (b) respectively.

The inputs of module A are a, b_0, p and output is p_{out}. In module A, an AND operation is performed on a with the LSB of b i.e. b_0, to attain an m-bit result aa. The m-bit signal aa is bit-wise XORed with the m-bit signal p resulting in the output p_{out}. The inputs of module B are a, b, t and outputs are a_{new} and b_{new}. In module B, b is right shifted by one bit resulting in the output b_{new}. a is left shifted by one bit to obtain an m-bit result al. Logical AND operation is performed on t with a_{m-1} to obtain an m-bit result tt. This m-bit signal tt is XORed with al resulting in the output a_{new}. Before the multiplication operation begins, the registers $Reg2$ and $Reg3$ are initialized with the multiplicands b and a, respectively, and $Reg1$ is cleared. For every clock cycle, module A updates p value given by p_{out} and module B updates the a and b values given by a_{new} and b_{new} respectively. The final multiplication result is given by res after m clock cycles.

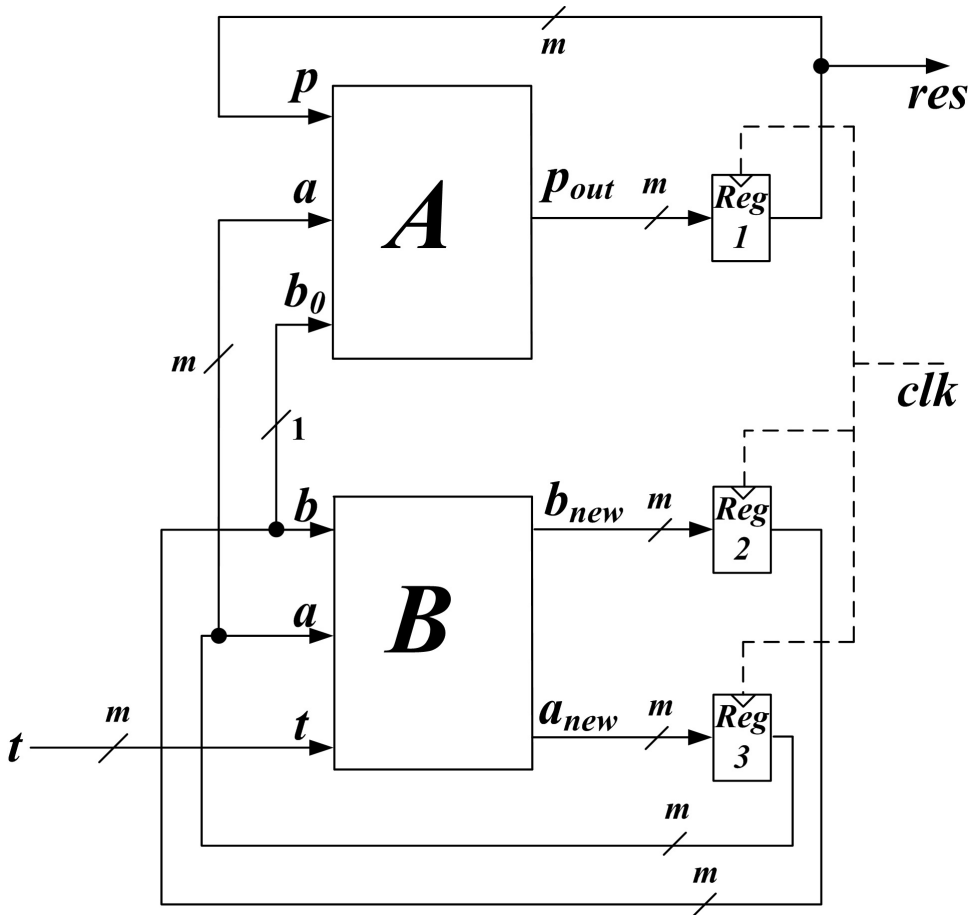
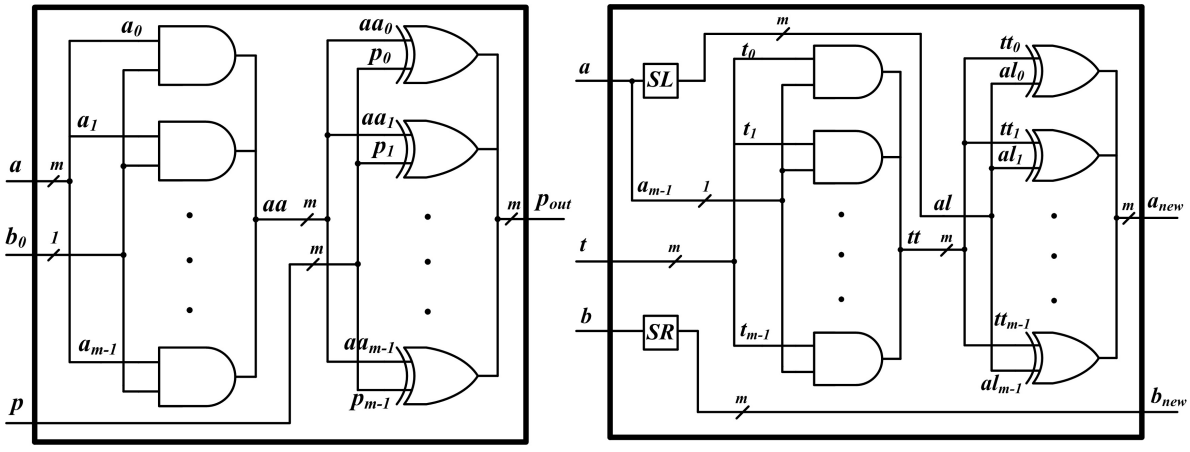


Figure 4.1: Block diagram of the proposed sequential multiplier architecture.



(a) Module A (b) Module B

Figure 4.2: Internal circuit details of the proposed architecture.

4.3.2 Analytical Results

As discussed in previous section, the proposed sequential multiplier consists of three m-bit registers, one module A and one module B. Each of the modules A and B consists of

m 2-input XOR gates and m 2-input AND gates each. An m-bit register can be realized using m 1-bit registers. Since the proposed architecture requires three m-bit registers, the total 1-bit registers required are $3m$. The shifting blocks, SL and SR, are consists of only re-wiring and hence do not contribute to any complexity. Therefore, the total area complexity of the proposed architecture is $2m$ XOR gates, $2m$ AND gates and $3m$ 1-bit registers. The critical path delay of the proposed multiplier is the maximum of delays of either module A or module B. It can be observed from the architecture that the delays of module A and module B are equal and is given by the expression $(T_X + T_A)$, where T_X and T_A are the delays of the XOR gate and AND gate, respectively. Hence, the critical path delay of the multiplier is computed as $(T_X + T_A)$. As established by the proposed algorithm, the multiplication of two m-bit elements is computed over m iterations. Hence, the resultant latency is m clock cycles.

Table 4.1: Area complexity and delay comparison of the proposed architecture with existing architectures over $GF(2^m)$.

Multipliers	#XOR	#AND	#MUX	#Registers	Latency	Critical Path Delay
[18]	$2m$	$3m$	m	m^2	$3m$	$(T_A + T_N + T_O)log_2 m + T_X$
[19]	m	$2m$	$(m-1)^a + m^b$	$3m$	m	$T_X + 2T_A + T_N + (m+1)T_O$
[20]	m	m	$2m+1$	$3m$	$2m$	$T_X + T_A$
[21]	$2m$	$4m$	$m^a + m^c$	$3m$	m	$T_X + T_A$
[22]	$6m+18$	0	$14m+26$	$6m+7$	$m/4$	$4T_X + 2T_M$
Proposed	$2m$	$2m$	0	$3m$	m	$T_X + T_A$

[a]OR gates; [b]1-to-2 DMUX; [c]Inverter.

Table 4.1 presents the comparison of area complexity (in terms of gate count), latency (#clock cycles) and critical path delay of the proposed architecture with other sequential architectures [18–22] available in the literature. It may be noted that T_N, T_O and T_M denote the delays of an inverter, OR gate and 2:1 MUX, respectively. In order to highlight the differences among various multiplier designs, the irreducible polynomial with field order $m = 163$, i.e. $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ recommended by National Institute of Standards and Technology (NIST), is considered as an example. Since the comparison of area complexity expressed in terms of gate count cannot provide a clear difference among the multipliers considered. A better area complexity comparison can be achieved using the transistor count parameter. Moreover, latency alone cannot achieve a

fair comparison of computation time; total delay as a product of latency and critical path must be considered.

Table 4.2: Comparison of transistor count, latency, critical path delay, total delay, area-delay product, % reduction in area and % reduction in ADP of the proposed architecture with existing architectures over $GF(2^{163})$.

Multipliers	#Transistors	Latency	CP (ns)	Total Delay (ns)	ADP ($\times 10^6$)	%Reduction in Area	%Reduction in ADP
[18]	217442	489	174	85086	18501	96%	99%
[19]	8796	163	1344	219072	1927	11%	98%
[20]	7830	326	18	5868	46	0.08%	50%
[21]	11084	163	18	2934	32	29%	28%
[22]	27704	41	60	2460	68	71%	66%
Proposed	7824	163	18	2934	23	-	-

Table 4.2 provides the comparison of area complexity (in terms of total transistor count), total delay (latency × critical path) and area-delay product (#transistors × ns). In order to estimate the transistor count of individual gates, traditional CMOS logic transistor counts [55] are used: six transistors for a 2-input XOR gate, six for a 1-bit 2:1 MUX, six for a 1-bit 1:2 DMUX, six for a 2-input OR gate, six for a 2-input AND gate and eight for a 1-bit register. The critical path delay of the multipliers are estimated using the real time circuits from STMicroelectronics [56]. The typical values of the propagation delays (t_{PD}) is considered for all the gates to ensure fair comparison. The circuits used are M74HC86 (XOR gate, t_{PD} = 12ns), M74HC257 (MUX, t_{PD} = 11ns), M74HC08 (AND gate, t_{PD} = 6ns), M74HC32 (OR gate, t_{PD} = 8ns) and M74HC04 (INVERTER, t_{PD} = 8ns). It can be observed from Table 4.2 that the proposed architecture requires low area complexity and low area-delay product compared to the sequential architectures available in the literature. This table also provides the percentage reduction in area complexity and area-delay product achieved by the proposed architecture compared to the architectures available in the literature. It can be noted that the proposed architecture achieves significant reduction in area complexity of about 96% and 71% and reduction in ADP of about 99% and 66% compared to the architectures [18,22], respectively. Moreover, the proposed architecture achieves about 11% and 29% reduction in area complexity while

achieving 98% and 28% reduction in ADP compared to the architectures [19, 21]. The proposed architecture requires the same area complexity as that of the architecture [20]. However, the ADP of the proposed architecture achieves 50% reduction in ADP compared to the sequential architecture [20].

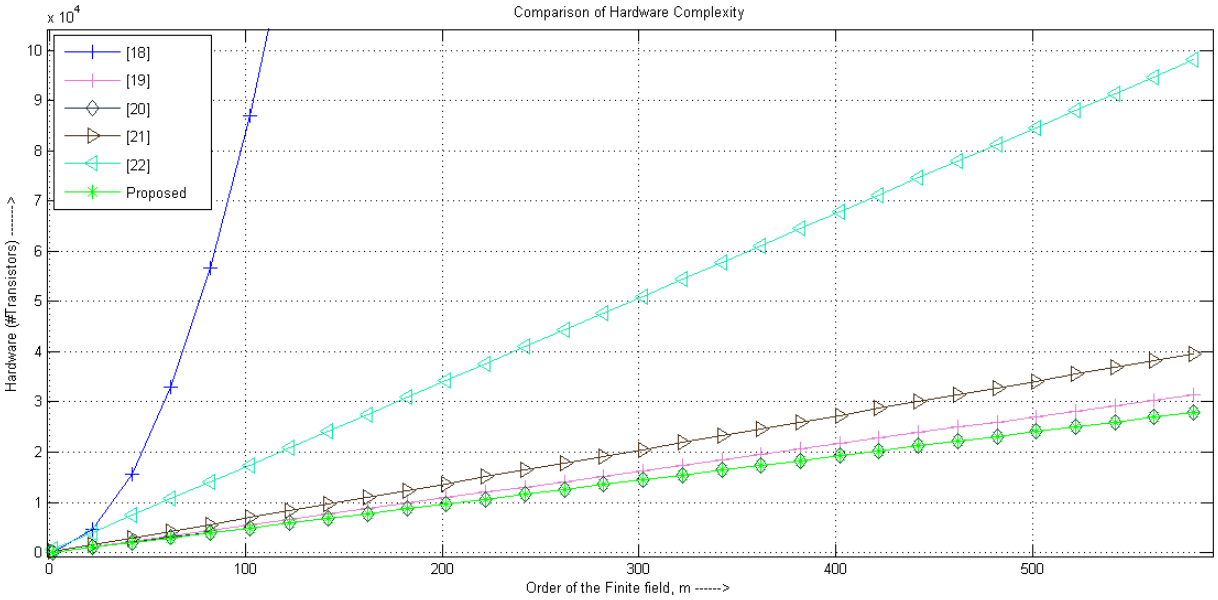


Figure 4.3: Area complexity comparison of sequential multipliers.

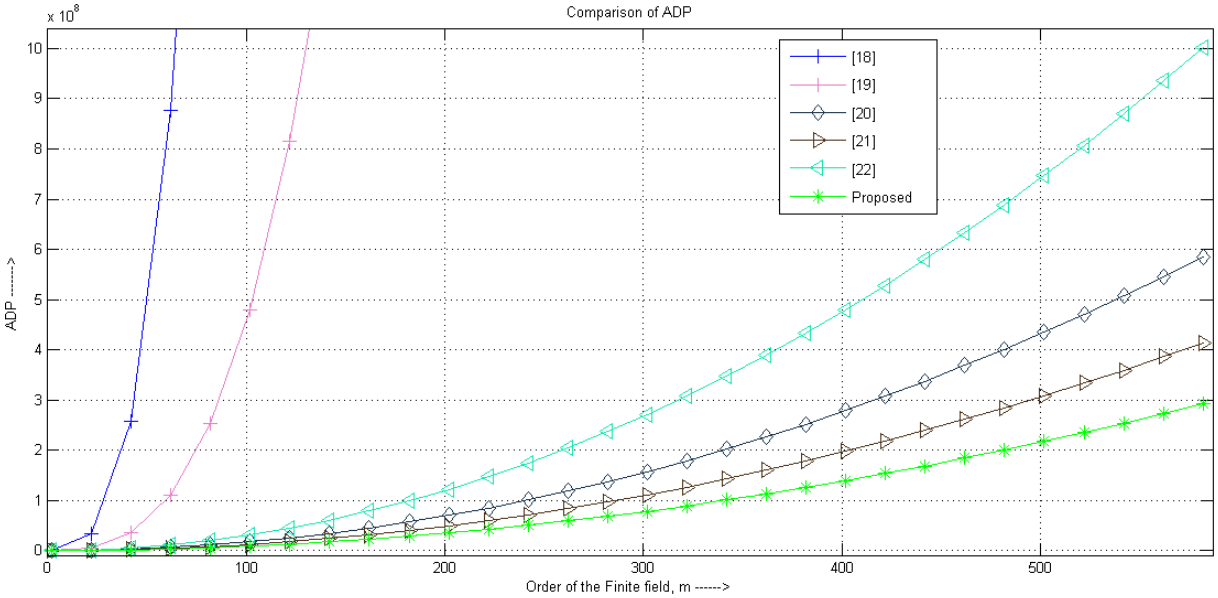


Figure 4.4: Area-Delay Product comparison of sequential multipliers.

Fig. 4.3 and 4.4 shows the area complexity and ADP comparison of the sequential

multipliers over a range of m values. As the order of the finite field increases from $m = 2$ to $m = 600$ in Fig. 4.3, it can be observed that the proposed architecture achieves low area complexity compared to the sequential architectures [18, 19, 21, 22], while requiring equal area complexity compared to the sequential architecture [20]. Furthermore, Fig. 4.4 shows that the proposed architecture achieves low ADP compared to the sequential architectures [18–22]. Moreover, it can also be observed that the delay of the proposed architecture is comparable to other architectures.

4.3.3 Implementation Results

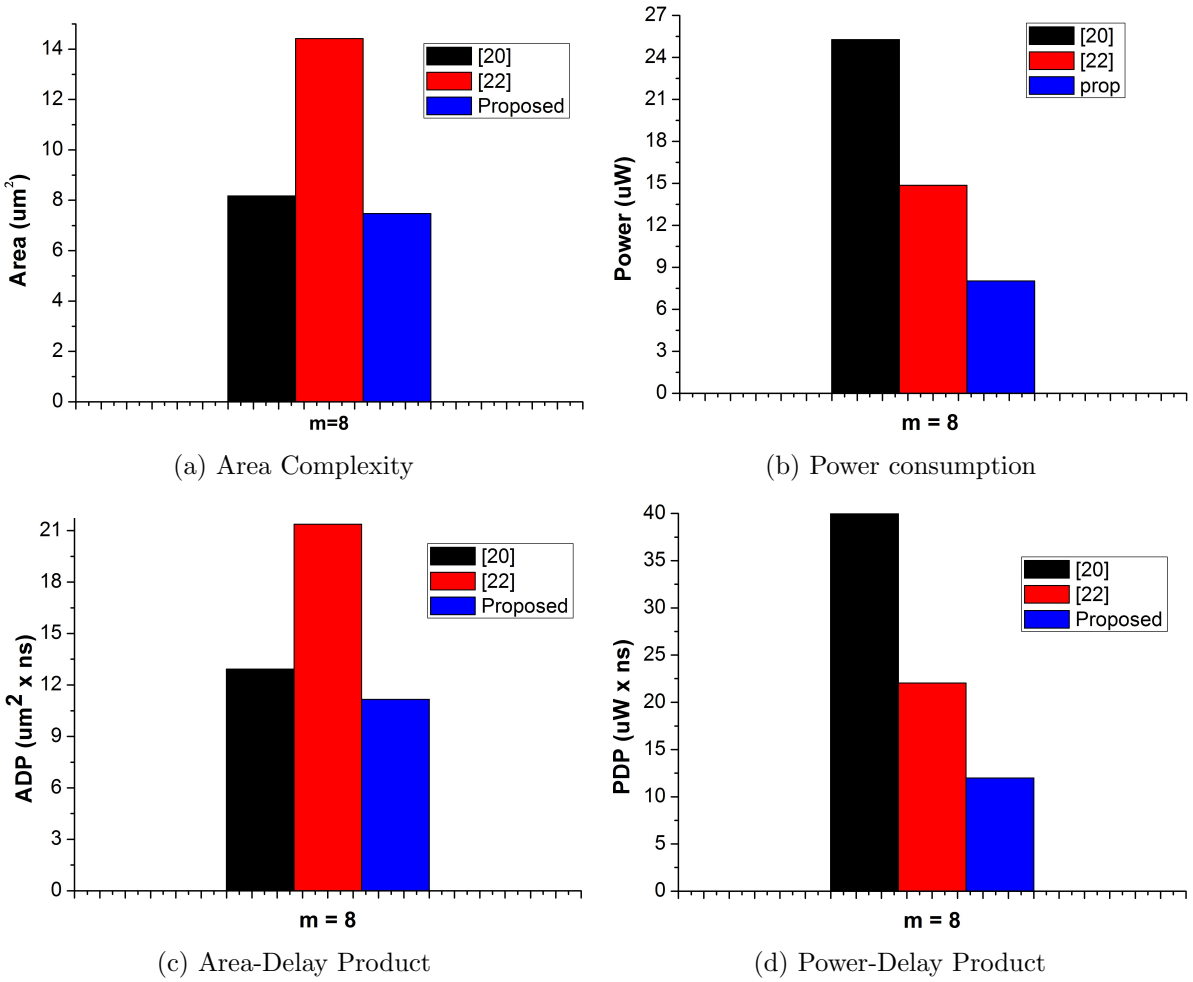
The performance of the proposed sequential multiplier architecture and the sequential multiplier architectures available in the literature are verified by implementing them on ASIC and FPGA platforms. The implementation results of these architectures are presented in the following sub-sections.

4.3.3.1 ASIC Implementation Results

Table 4.3: ASIC implementation results of sequential multipliers.

	Multipliers / Metrics	[20]	[22]	Proposed
m=8	Total Delay (ns)	1.582	1.482	1.493
	Area (μm^2)	8.17	14.418	7.48
	Power (μW)	25.282	14.866	8.035
	ADP ($\mu m^2 \times ns$)	12.925	21.368	11.168
	PDP ($\mu W \times ns$)	39.99	22.031	11.996
m=163	Total Delay (ns)	1.717	1.731	1.615
	Area (μm^2)	177.594	372.276	87.48
	Power (μW)	811.769	404.839	95.282
	ADP ($\mu m^2 \times ns$)	304.929	644.410	141.28
	PDP ($\mu W \times ns$)	1393.807	700.776	153.88

The sequential multipliers [20,22] are considered for hardwarea implementation and



(a) Area Complexity

(b) Power consumption

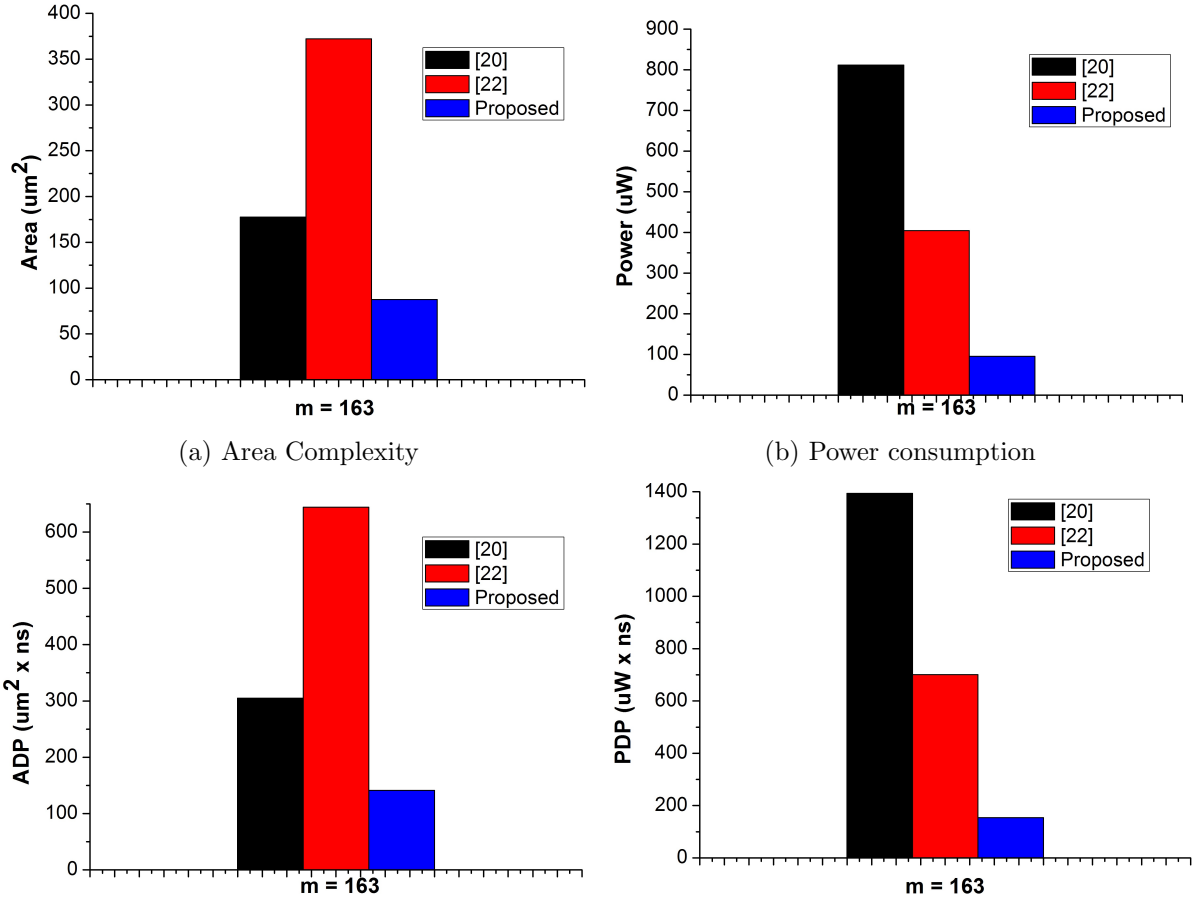
(c) Area-Delay Product

(d) Power-Delay Product

Figure 4.5: ASIC implementation results of sequential multipliers for $m = 8$.

comparison since the sequential multiplier [20] requires same area complexity as the proposed multiplier and the sequential multiplier [22] requires low ADP among existing multipliers. Therefore, the proposed multiplier along with the multipliers in [20, 22] are modelled in Verilog and synthesized with Cadence Encounter RTL Compiler Tool which uses UMC $0.18\mu m$ technology for $m = 8$ and $m = 163$. The field orders $m = 8$ and $m = 163$ are considered for implementation as $m = 8$ is used in AES and $m = 163$ is recommended by NIST. The delay, area complexity, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 4.3). The area complexity, power consumption, ADP and PDP results are also plotted for $m = 8$ and $m = 163$ as shown in Fig. 4.5(a)-(d) and Fig. 4.6(a)-(d), respectively.

It is clear from the histogram (see Fig. 4.5 and Fig. 4.6) that the proposed multiplier



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 4.6: ASIC implementation results of sequential multipliers for $m = 163$.

requires the least area complexity, power consumption, ADP and PDP among existing multipliers. More specifically, it can be observed that the proposed multiplier achieves reduction of about 8.5%, 68%, 13.6% & 70% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [20] for $m = 8$. Similarly, the proposed multiplier achieves reduction of about 48%, 45%, 47% & 45% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [22] for $m = 8$. The proposed multiplier achieves reduction of about 50%, 88%, 53% & 88% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [20] for $m = 163$. Similarly, the proposed multiplier achieves reduction of about 76%, 76%, 78% & 78% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [22] for $m = 163$. Moreover, the proposed multiplier also achieves reduction in delay compared to the multipliers [20,22] for $m = 163$. These improvements in area complexity, power consumption, ADP and PDP achieved by

the proposed multiplier indicate an efficient design in terms of both area complexity and power consumption without much increase in delay.

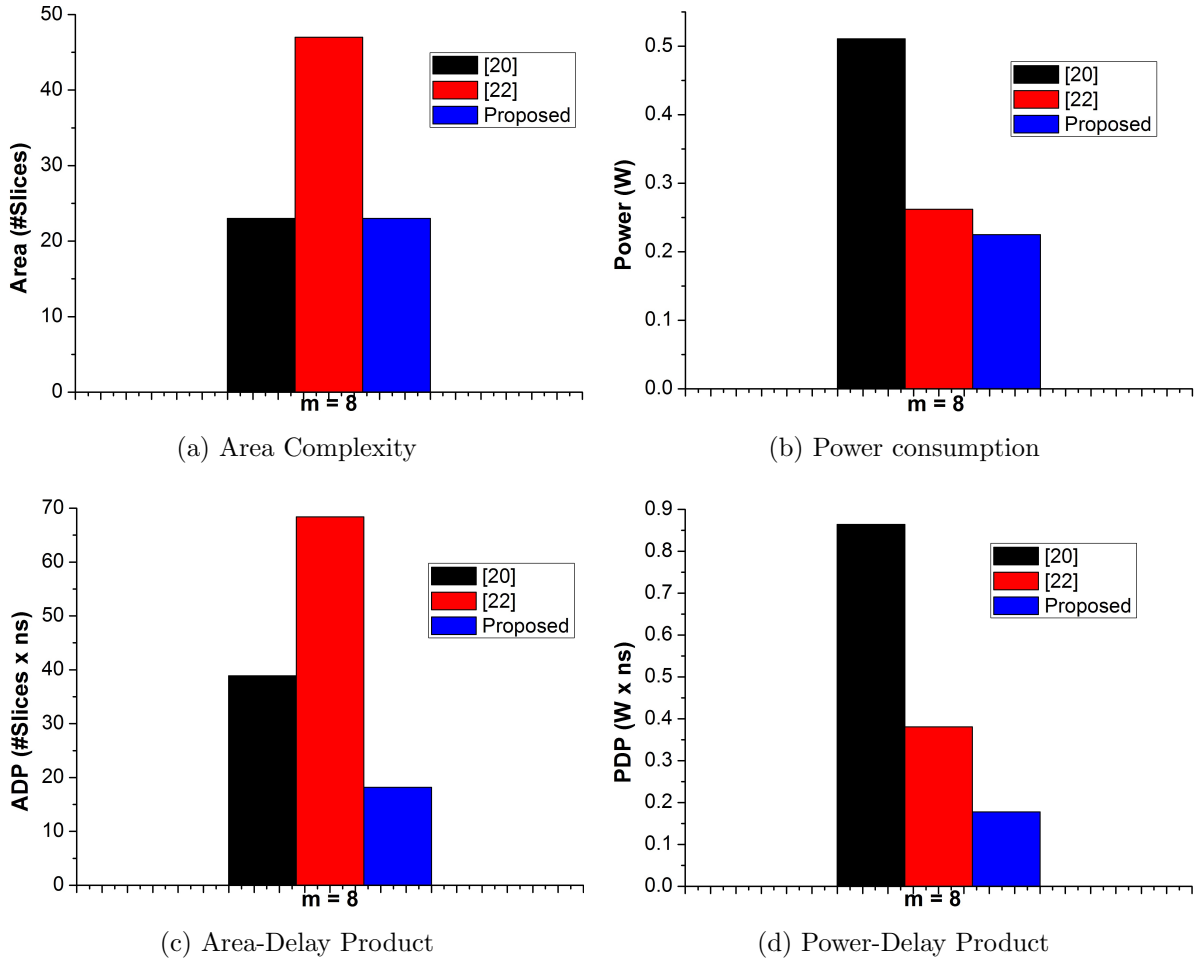
4.3.3.2 FPGA Implementation Results

Table 4.4: FPGA implementation results of sequential multipliers.

	Metrics \ Multipliers	[20]	[22]	Proposed
m=8	Total Delay (ns)	1.691	1.455	0.791
	Area ($\#Slices$)	23	47	23
	Power (W)	0.511	0.262	0.225
	ADP ($\#Slices \times ns$)	38.893	68.385	18.193
	PDP ($W \times ns$)	0.864	0.381	0.178
m=163	Total Delay (ns)	9.131	11.686	4.105
	Area ($\#Slices$)	326	850	165
	Power (W)	0.923	0.713	0.599
	ADP ($\#Slices \times ns$)	2976.706	9933.1	677.325
	PDP ($W \times ns$)	8.428	8.33	2.459

In addition to the ASIC implementation, the functionality of the proposed multiplier are also verified by implementing the Verilog models on FPGA platform. The Verilog models of the proposed multiplier and the multipliers [20,22] are simulated and synthesized using Xilinx Vivado 2014.2 tool. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The delay, area, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 4.4). The area complexity, power consumption, ADP and PDP results are also plotted for $m = 8$ and $m = 163$ as shown in Fig. 4.7(a)-(d) and Fig. 4.8(a)-(d), respectively.

It is clear from the histogram (see Fig. 4.7 and Fig. 4.8) that the proposed multiplier requires the least area complexity, power consumption, ADP and PDP among existing multipliers. More specifically, it can be observed that the proposed multiplier achieves



(a) Area Complexity

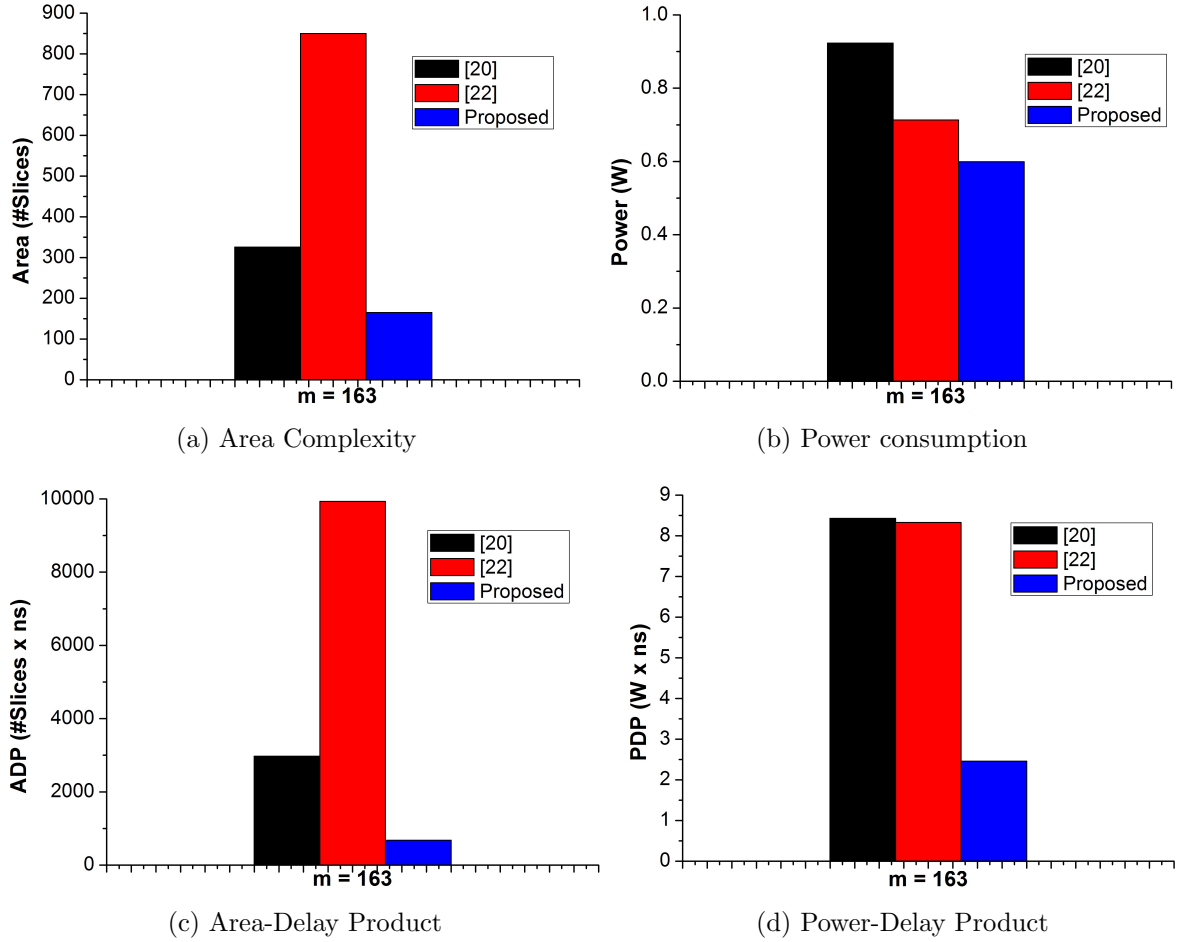
(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 4.7: FPGA implementation results of sequential multipliers for $m = 8$.

same area complexity and reduction of about 56%, 53% & 79% in power consumption, ADP and PDP, respectively, when compared to the multiplier [20] for $m = 8$. Similarly, the proposed multiplier achieves reduction of about 51%, 14%, 73% & 53% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [22] for $m = 8$. For $m = 163$, the proposed multiplier achieves reduction of about 49%, 35%, 77% & 70% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [20]. Similarly, the proposed multiplier achieves reduction of about 80%, 16%, 93% & 70% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [22] for $m = 163$. Moreover, the proposed multiplier also achieves reduction in delay compared to the multipliers [20, 22] for $m = 163$. In addition, the proposed multiplier achieves reduction in delay compared to the multipliers [20, 22]. These improvements in area complexity, power consumption, ADP and PDP achieved by the proposed multiplier indicate an efficient design in terms



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 4.8: FPGA implementation results of sequential multipliers for $m = 163$.

of both area and power without much increase in delay.

4.4 Proposed Sequential Multiplier Architecture over GF(2^8) for Irreducible Polynomials

This sub-section presents the design of the proposed sequential multiplier architecture over $GF(2^8)$ for irreducible polynomials. AES and Twofish encryption algorithms are realized using the proposed sequential multiplier architecture and implemented on an FPGA platform. These implementation results of the proposed architecture are compared with the results achieved by the multiplier architectures available in the literature.

4.4.1 Design of Proposed Sequential Multiplier Architecture over GF(2^8) for Irreducible Polynomials

Fig. 4.9 shows the block diagram of the sequential multiplier architecture over $GF(2^8)$ derived from the proposed sequential multiplier architecture over $GF(2^m)$ (section 4.3). The block diagram consists of two modules A and B and three 8-bit registers. The multiplier takes one 8-bit input t, where t denotes the binary representation of the irreducible polynomial over $GF(2^8)$. Module A computes the polynomial multiplication and module B computes the modular reduction. The logic diagrams of the modules A and B are shown in Fig. 4.10(a) and (b), respectively. The inputs of module A are a, b_0, p and output is p_{out}. In module A, an AND operation is performed on a with the LSB of b i.e. b_0 to attain 8-bit result aa. The 8-bit signal aa is bit-wise XORed with the input p resulting in the output p_{out}. The inputs of module B are a, b, t and outputs are a_{new} and b_{new}. In module B, b is right shifted by one bit to attain the output b_{new}. a is left shifted by one bit to attain an 8-bit result al. An AND operation is performed on t with a_7 to attain an 8-bit result tt. The 8-bit signal tt is XORed with al to obtain a_{new}. Before the multiplication operation begins, the registers $Reg2$ and $Reg3$ are initialized with the multiplicands b and a respectively and $Reg1$ is cleared. For every clock cycle, module A computes the new p value given by p_{out} and module B computes the new a and b values given by a_{new} and b_{new} respectively. The final multiplication result is given by res after 8 clock cycles.

4.4.2 Analytical Results

As presented in previous section, the proposed multiplier requires three 8-bit registers, one module A and one module B. Both module A and module B consists of eight 2-input XOR gates and eight 2-input AND gates. An 8-bit register can be realized using eight 1-bit registers. Since the proposed architecture requires three 8-bit registers, the total 1-bit registers required is twenty-four. The shifting blocks, SL and SR, are comprised of only re-wiring and hence do not contribute to any complexity. Hence, the total area complexity of the proposed architecture is sixteen 2-input XOR gates, sixteen 2-input AND gates and twenty-four 1-bit registers. The critical path delay of the proposed multi-

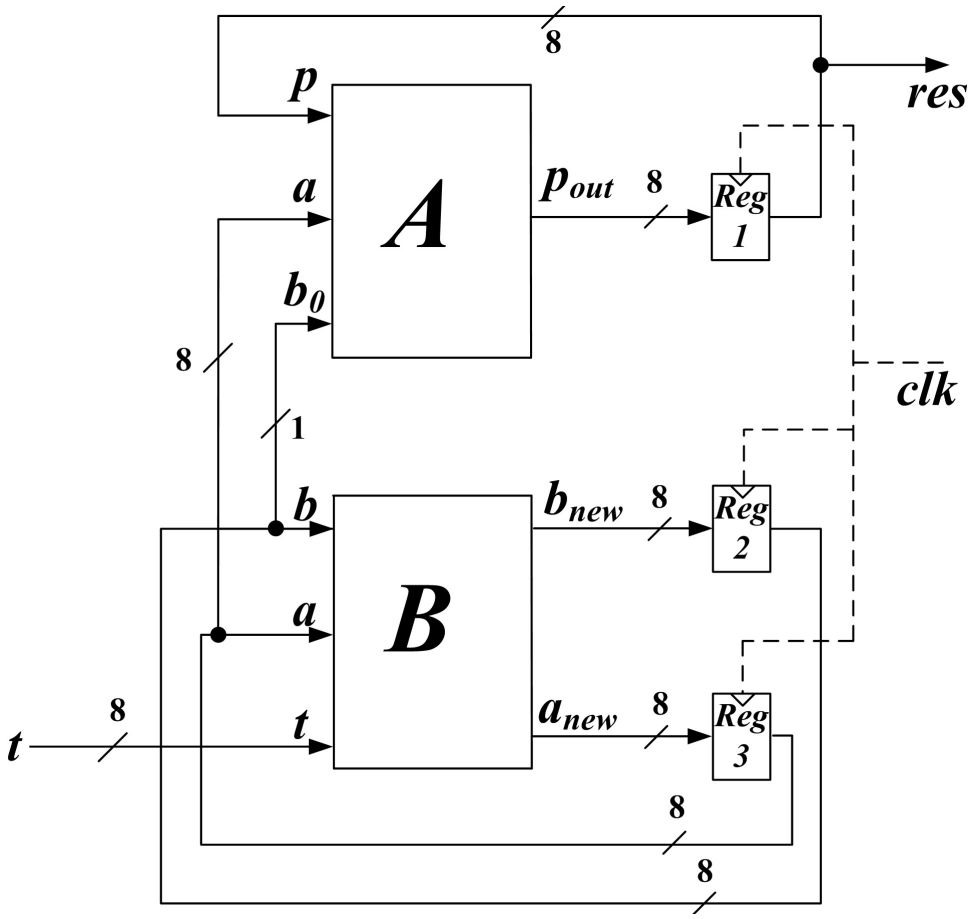
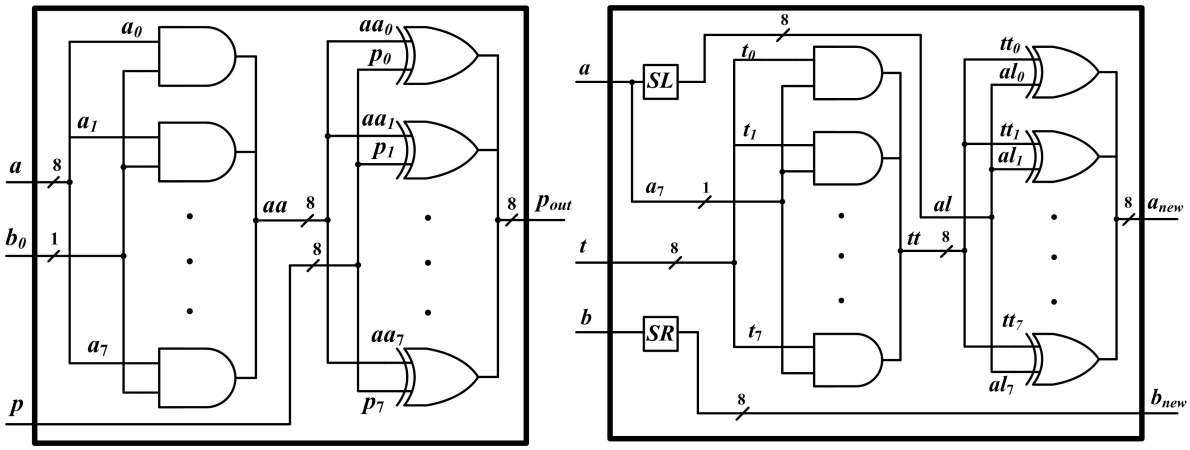


Figure 4.9: Block diagram of the proposed sequential multiplier architecture over $GF(2^8)$.



(a) Module A (b) Module B

Figure 4.10: Internal circuit details of the proposed architecture.

plier is the maximum of delays of either module A or module B. It can be observed from the architecture that the delays of module A and module B are equal i.e. $(T_X + T_A)$, where T_X and T_A are the delays of a 2-input XOR gate and a 2-input AND gate, respec-

Table 4.5: Area complexity and delay comparison of the proposed architecture with existing architectures over $GF(2^8)$.

Multipliers	#XOR	#AND	#MUX	#Registers	Latency	Critical Path Delay
[18]	16	24	8	64	24	$(T_A + T_N + T_O)3 + T_X$
[19]	8	16	$7^a + 8^b$	24	8	$T_X + 2T_A + T_N + 9T_O$
[20]	8	8	17	24	16	$T_X + T_A$
[21]	16	32	$8^a + 8^c$	24	8	$T_X + T_A$
[22]	66	0	138	55	2	$4T_X + 2T_M$
Proposed	16	16	0	24	8	$T_X + T_A$

[a]OR gates; [b]1-to-2 DMUX; [c]Inverter.

tively. Hence, the critical path delay of the multiplier is computed as $(T_X + T_A)$. Since the multiplication of two 8-bit elements can be computed over eight iterations, the resultant latency is eight clock cycles.

Table 4.5 presents the comparison of number of gates, latency and critical path of the proposed architecture compared to other multiplier architectures [18–22] available in the literature. It may be noted that T_N, T_O and T_M denotes the delay of an inverter, 2-input OR gate and 2:1 MUX, respectively. The comparison of area complexity in terms of gate count cannot provide a clear difference among the multipliers considered. A better area-complexity comparison can be achieved using the transistor count parameter. Since latency alone cannot achieve a fair comparison of computation time, total delay as a product of latency and critical path is considered.

Table 4.6 provides the comparison of area complexity(number of transistors), total delay (latency × critical path) and ADP (#Transistors × ns). In order to estimate the transistor count of individual gates, traditional CMOS logic transistor counts [55] are used: six transistors for a 2-input XOR gate, six for a 1-bit 2:1 MUX, six for a 1-bit 1:2 DMUX, six for a 2-input OR gate, six for a 2-input AND gate and eight for a 1-bit register. Some real time circuits from STMicroelectronics [56] are considered to estimate the critical path delay of the multipliers. The typical propagation delay (t_{PD}) of the respective gates is considered to ensure fair comparison. The circuits used are M74HC86 (XOR gate, t_{PD} = 12ns), M74HC257 (MUX, t_{PD} = 11ns), M74HC08 (AND gate, t_{PD} =

Table 4.6: Comparison of transistor count, latency, critical path delay, total delay, area-delay product, % reduction in area and % reduction in ADP of the proposed architecture with existing architectures over $GF(2^8)$.

Multipliers	#Transistors	Latency	CP (ns)	Total Delay (ns)	ADP ($\times 10^3$)	%Reduction in Area	%Reduction in ADP
[18]	800	24	78	1872	1497.6	52%	96%
[19]	384	8	104	832	319.488	Equal	82%
[20]	390	16	18	288	112.32	Equal	50%
[21]	544	8	18	144	78.336	29%	29%
[22]	1664	2	60	118	196.352	77%	71%
Proposed	384	8	18	144	55.296	-	-

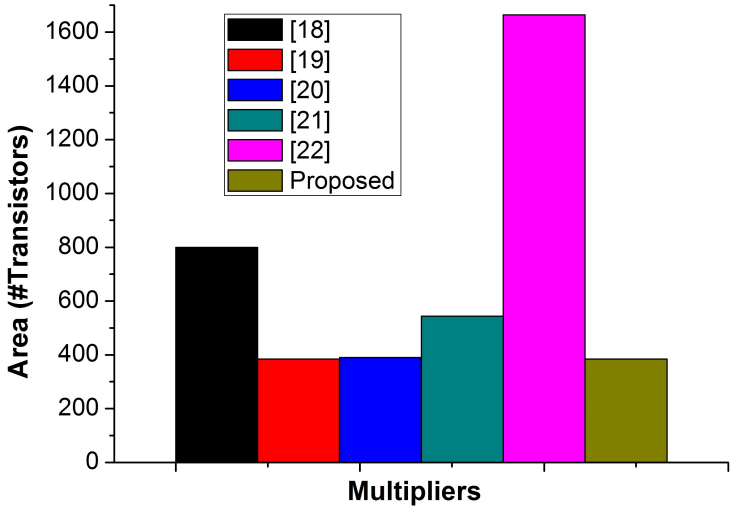


Figure 4.11: Area complexity comparison of the proposed architecture with existing architectures over $GF(2^8)$.

6ns), M74HC32 (OR gate, t_{PD} = 8ns) and M74HC04 (INVERTER, t_{PD} = 8ns).

Fig. 4.11 and Fig. 4.12 illustrates the histograms plotted for area complexity and ADP, respectively, of the proposed multiplier and various multipliers reported in the literature. It can be observed from the results that the proposed multiplier achieves low ADP compared to existing sequential multipliers available in the literature. It may also be noted that the area complexity is low for the proposed architecture compared to the multipliers [18, 21, 22] while its same as that of the multipliers [19, 20]. However, the proposed multiplier achieves low ADP compared to all the multipliers [18–22]. Moreover,

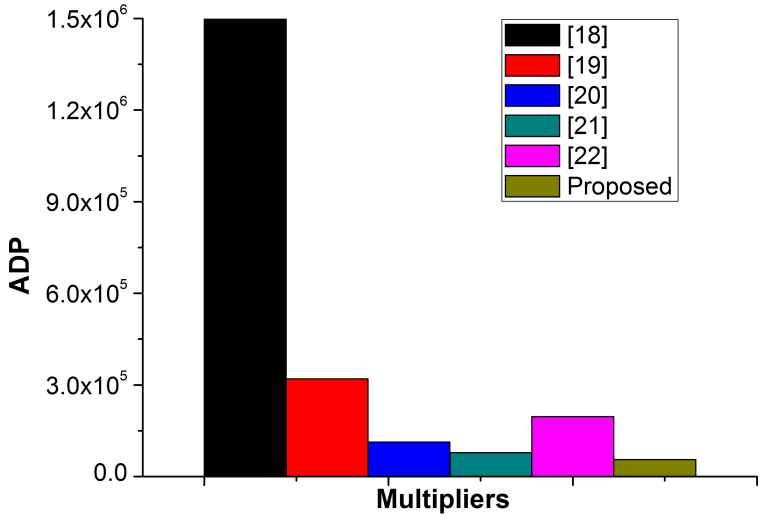


Figure 4.12: Area-Delay Product comparison of the proposed architecture with existing architectures over $GF(2^8)$.

the delay of the proposed multiplier is low compared to the sequential multipliers [18–21] while requiring slightly more delay compared to the sequential multiplier [22]. The critical path delay of the proposed design is also low compared to multipliers [18,19,22] indicating that it can operate at higher frequencies.

4.4.3 Implementation Results

The performance of the proposed sequential multiplier architecture is verified by designing AES and Twofish cryptographic algorithms and implementing them on FPGA platform. The implementation results of these two algorithms are presented in the following sub-sections.

4.4.3.1 FPGA implementation of AES

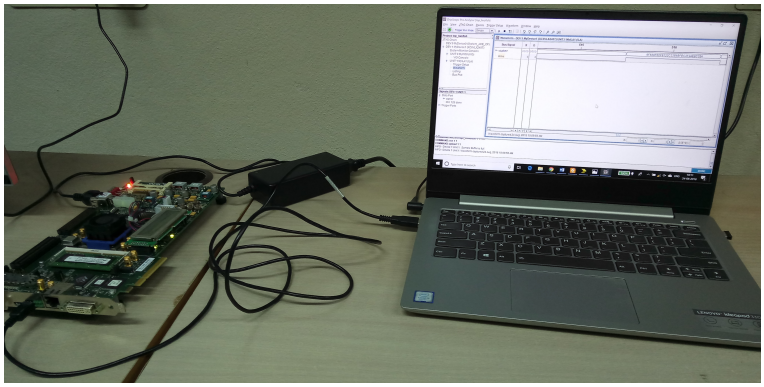
The proposed sequential multiplier architecture over $GF(2^8)$ is employed to realize the AES algorithm and is implemented on an FPGA platform. The AES is a symmetric-key cryptographic algorithm developed based on a substitution-permutation structure using block-cipher technique. The block size of the plaintext to be encrypted is 128 bits with the key size options of 128, 192 and 256 bits. In this work, the key size of 128 bits is used for the hardware implementation on the FPGA device [57] and hence the

Table 4.7: FPGA implementation results of AES.

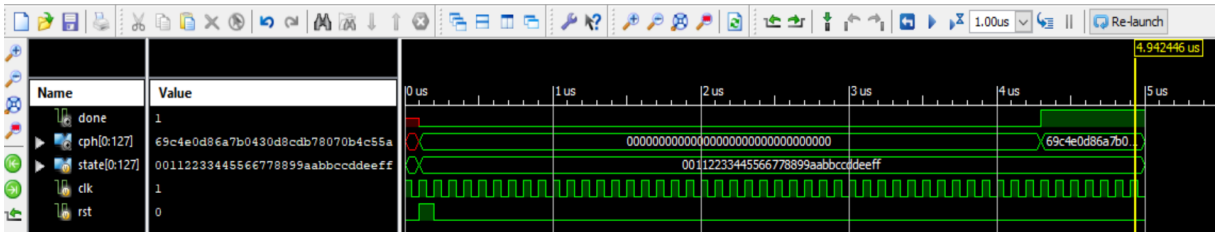
Metrics \ Multipliers	[20]	[22]	Proposed
Total Delay (ns)	78.371	58.693	60.327
Area ($\#Slices$)	12981	15793	10097
Power (W)	0.917	0.459	0.263
ADP ($\#Slices \times ns$)($\times 10^3$)	1017.334	926.939	609.122
PDP ($W \times ns$)	71.87	26.94	15.87

algorithm is performed for 10 rounds. The Verilog models for AES encryptor-decryptor are developed employing the proposed sequential multiplier and the sequential multipliers [20, 22] available in the literature to perform finite field multiplications. These Verilog models are simulated and synthesized using Xilinx Vivado 2014.2 software tool to verify their functionality. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The experimental setup of the FPGA implementation of AES is shown in Fig. 4.13(a). The AES encryption and decryption is performed with Plaintext and Key values as 0x00112233445566778899AABBCCDDEEFF and 0x000102030405060708090A0B0C0D0E0F and the Ciphertext obtained is 0x69C4E 0D86A7B0430D8CDB78070B4C55A and the simulation waveforms are shown in Fig. 4.13(b) and (c).

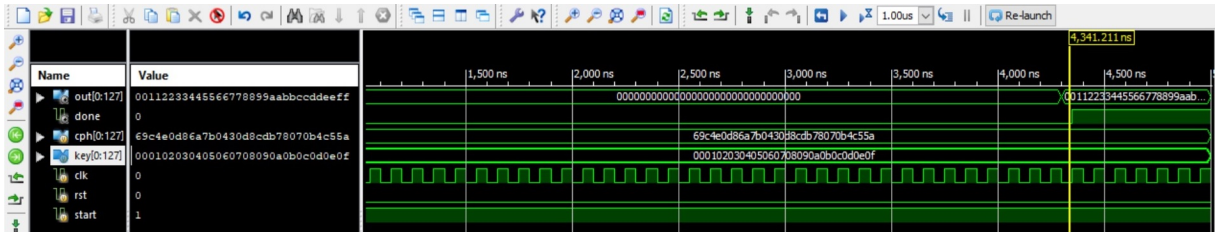
The delay, area, power consumption, ADP and PDP results are computed using the device utilization summary and presented in Table 4.7. The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 4.14(a)-(d), respectively. It is clear from the histogram (see Fig. 4.14) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, the proposed multiplier achieves reduction of about 22%, 71%, 40% & 77% in area complexity, power consumption, ADP and PDP compared to the multiplier [20]. Similarly, the proposed multiplier also achieves reduction of about 36%, 42%, 34% & 41% in area complexity, power consumption, ADP and PDP compared to the multiplier [22]. These improvements achieved by the proposed multiplier in area complexity, power consumption, ADP and PDP indicates an efficient design in both area



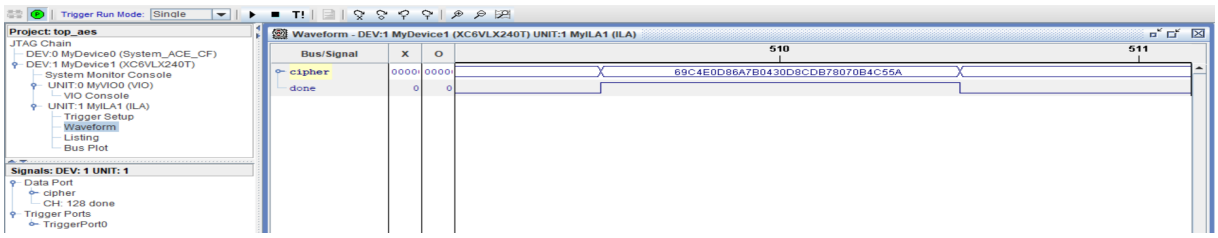
(a) Experimental setup



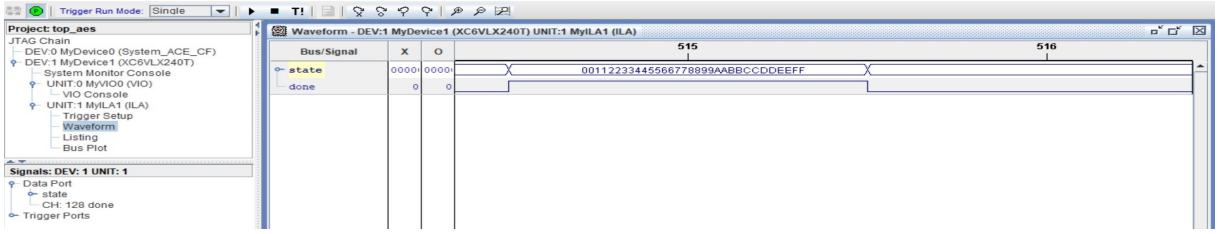
(b) FPGA simulation waveform of AES encryption



(c) FPGA simulation waveform of AES decryption

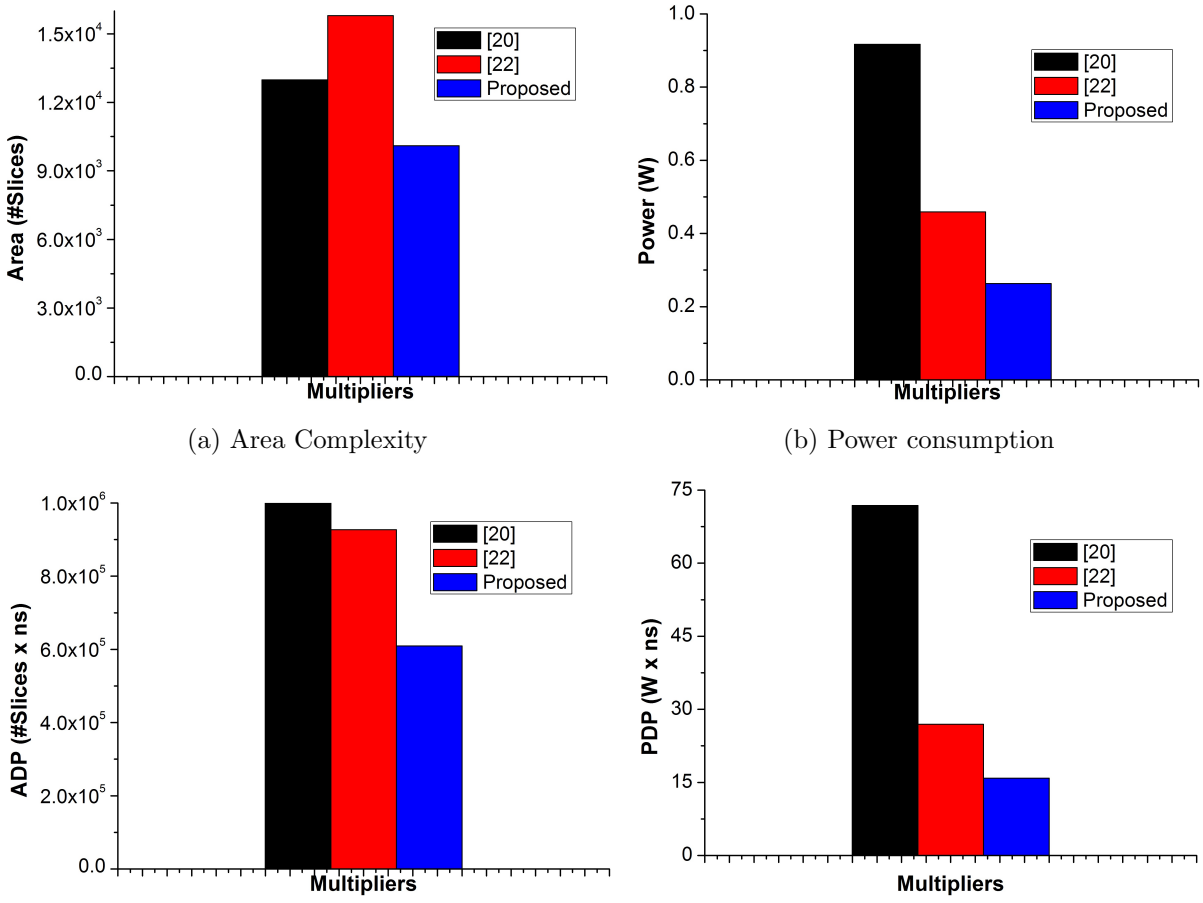


(d) FPGA ChipscopePro implementation waveform of AES encryption



(e) FPGA ChipscopePro implementation waveform of AES decryption

Figure 4.13: Experimental setup and simulation of AES.



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

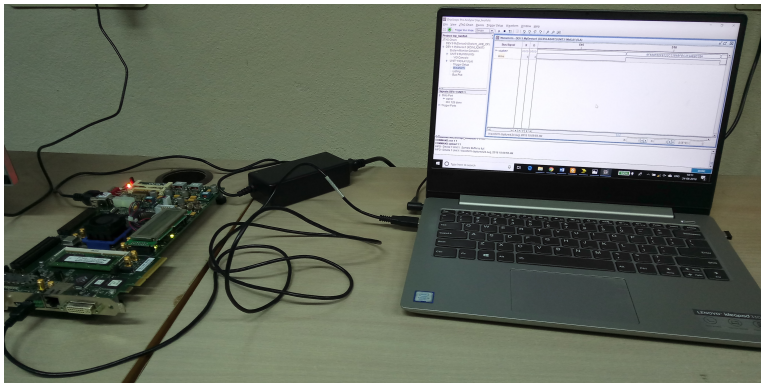
Figure 4.14: FPGA implementation results of AES.

and power without much increase in delay.

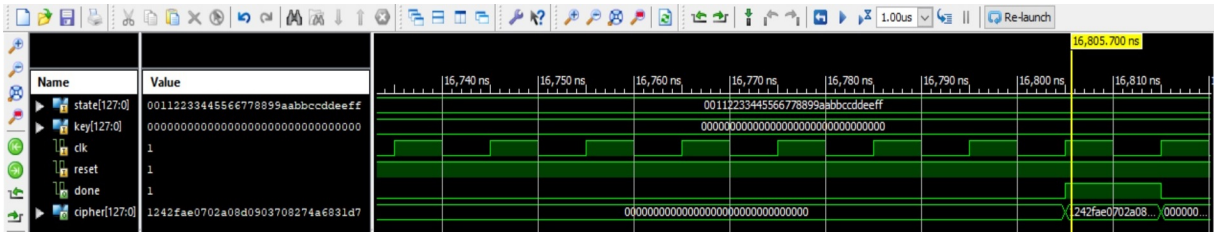
4.4.3.2 FPGA implementation of Twofish

Table 4.8: FPGA implementation results of Twofish.

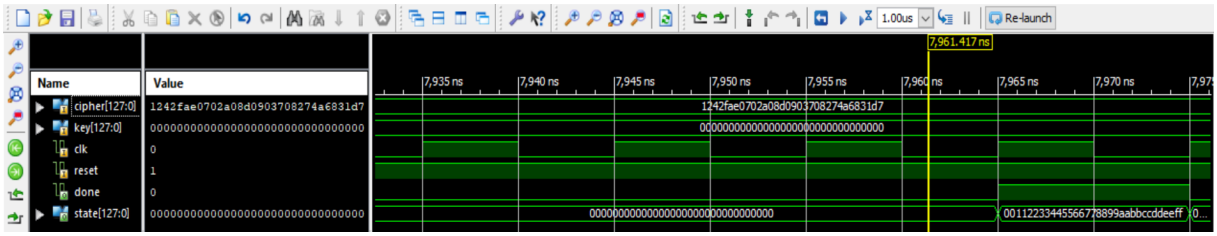
Metrics \ Multipliers	[20]	[22]	Proposed
Total Delay (ns)	106.009	85.359	91.870
Area $(\#Slices)$	15805	19805	15148
Power (W)	0.654	0.274	0.171
ADP $(\#Slices \times ns)(\times 10^3)$	1675.472	1690.535	1391.647
PDP $(W \times ns)$	69.33	23.39	15.71



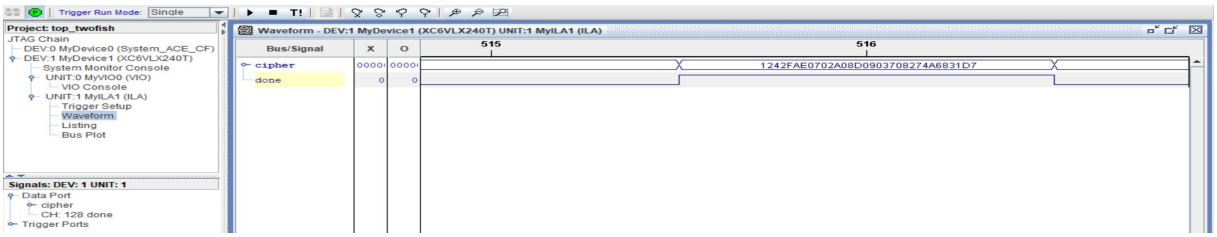
(a) Experimental setup



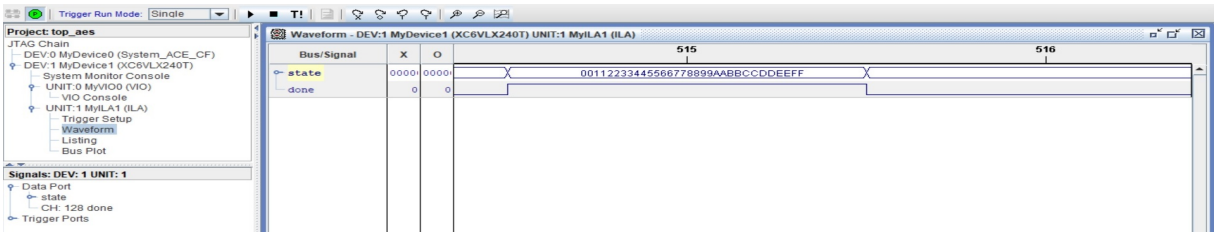
(b) FPGA simulation waveform of Twofish encryption



(c) FPGA simulation waveform of Twofish decryption

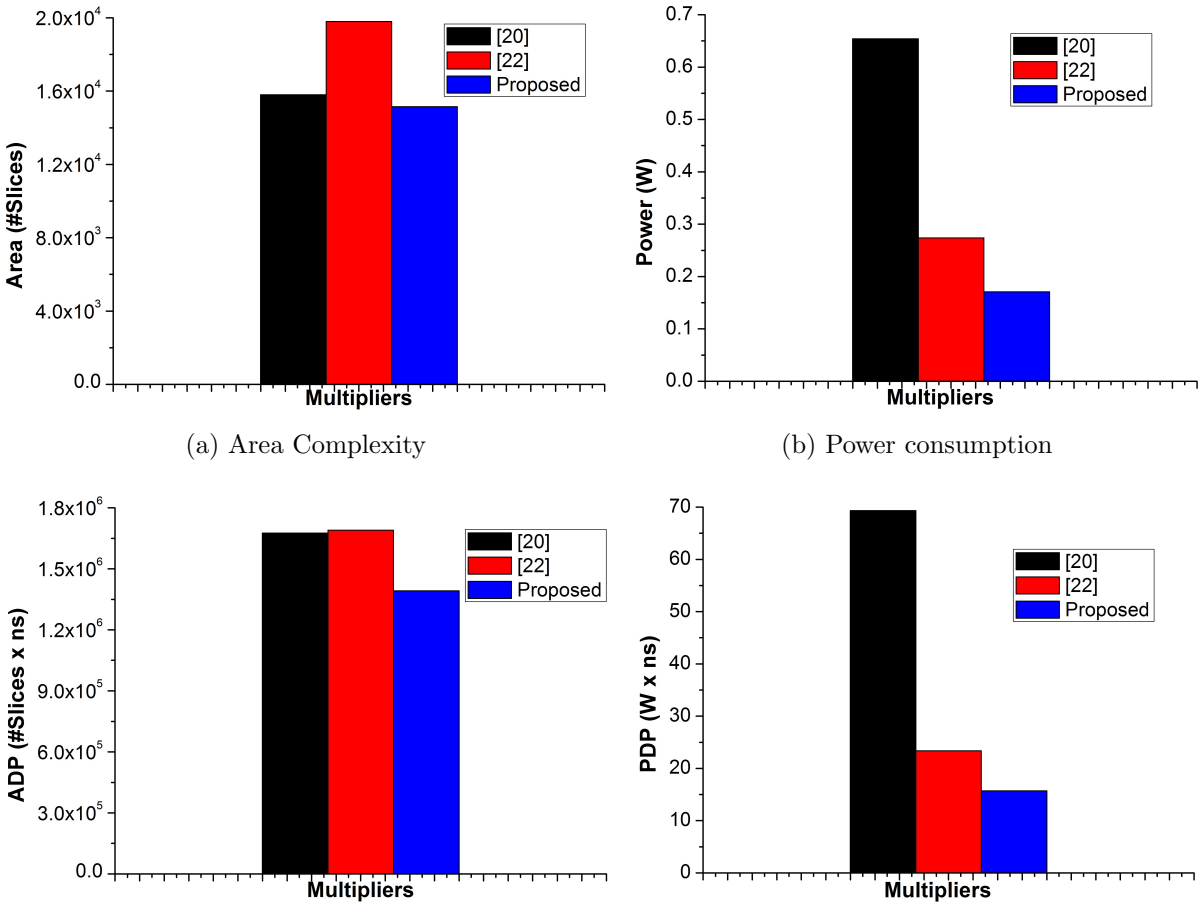


(d) FPGA ChipscopePro implementation waveform of Twofish encryption



(e) FPGA ChipscopePro implementation waveform of Twofish decryption

Figure 4.15: Experimental setup and simulation of Twofish.



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 4.16: FPGA implementation results of Twofish.

The Twofish cryptographic algorithm is developed based on Fiestel structure and the block cipher encryption technique. The block size of the plaintext to be encrypted is 128 bits with the key size options of 128, 192 and 256 bits. In this work, the key size of 128 bits is used for the hardware implementation on the FPGA device [58]. The Verilog models for Twofish encryptor-decryptor are developed employing the proposed sequential multiplier and the sequential multipliers [20, 22] available in the literature to perform finite field multiplications. These Verilog models are simulated and synthesized using Xilinx Vivado 2014.2 software tool to verify their functionality. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The experimental setup of the FPGA implementation of Twofish algorithm is shown in Fig. 4.15(a). The Twofish encryption and decryption is performed with Plaintext and Key values as 0x00112233445566778899AABBCCDDEEFF and 0x00000000000000000000000000000000 and the Ciphertext obtained is 0x1242FAE0702A08D0903708274A6831D7 and the

simulation waveforms are shown in Fig. 4.15(b) and (c).

The delay, area, power consumption, ADP and PDP results are computed using the device utilization summary and presented in Table 4.8. The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 4.16(a)-(d), respectively. It is clear from the histogram (see Fig. 4.16) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, the proposed multiplier achieves reduction of about 4.2%, 73%, 16% & 77% in area complexity, power consumption, ADP and PDP compared to the multiplier [20]. Similarly, the proposed multiplier also achieves reduction of about 23%, 37%, 17% & 32% in area complexity, power consumption, ADP and PDP compared to the multiplier [22]. These improvements achieved by the proposed multiplier in area complexity, power consumption, ADP and PDP indicates an efficient design in both area and power without much increase in delay.

4.5 Conclusion

In this chapter, a sequential polynomial basis multiplier architecture over $GF(2^m)$ is realized for the proposed algorithm. The area complexity and delay of the proposed multiplier are estimated and performance is compared with other sequential multipliers available in the literature. It may be concluded from the comparisons of the estimated results that the proposed multiplier achieves low area complexity for generic irreducible polynomials of degree m. The resultant area-delay product of the proposed multiplier is also low when compared to other multipliers, indicating an efficient multiplier design in terms of both area and delay. From the ASIC and FPGA synthesis results of the multipliers, it can be concluded that the proposed sequential multiplier achieves low area complexity, power consumption, area-delay product and power-delay product compared to the existing multipliers. In addition, a sequential multiplier architecture over $GF(2^8)$ is derived from the sequential multiplier architecture over $GF(2^m)$. The area complexities and delay of the proposed multiplier is estimated and performance is compared with other sequential multipliers available in the literature for $m = 8$. The Verilog models of two cryptographic algorithms, AES and Twofish, are developed employing the proposed

multiplier and the multipliers available in the literature. From the FPGA synthesis results of AES and Twofish algorithms, it can be concluded that the proposed multiplier achieves low area complexity, power consumption, area-delay product and power-delay product compared to the existing multipliers. The next chapter presents the design of the proposed systolic multiplier for irreducible polynomials.

Chapter 5

Low-power and Area-Efficient Systolic Multipliers over Polynomial Basis

This chapter presents an interleaved multiplication algorithm derived from a conventional interleaved multiplication algorithm available in the literature. A systolic multiplier architecture over $GF(2^m)$ for irreducible polynomials is designed based on the proposed algorithm. The performance of the proposed systolic multiplier architecture is computed analytically and compared with the multiplier architectures available in the literature. In addition, the analytical results are also verified by implementing the proposed architecture on Field Programmable Gate Array (FPGA) and Application Specific Integrated Circuit (ASIC) technologies and the results are compared with the existing architectures available in the literature. Moreover, the Verilog models of two cryptographic algorithms, AES and Twofish, are developed employing the proposed systolic multiplier and the multipliers available in the literature. These verilog models are implemented on FPGA to compute the performance improvement achieved by the proposed multiplier compared to the systolic multipliers available in the literature.

5.1 Introduction

The design of finite field multipliers employing systolic architectures tend to achieve high speeds since same hardware blocks are replicated to obtain a parallel structure. Hence, these architectures are preferred for applications with strict speed constraints

such as defence, network servers, etc. However, the area complexity of these systolic architectures is high and there is a need to reduce the area complexity.

In this work, a modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm to realize systolic multipliers. Subsequently, an efficient systolic polynomial basis multiplier, that supports multiplication of any two arbitrary finite field elements over $GF(2^m)$ for irreducible polynomials, is designed based on the proposed algorithm. The area complexity and delay of the proposed systolic multiplier over $GF(2^m)$ is estimated and its performance is compared with existing systolic multipliers [20, 23–38]. It is observed that the proposed systolic multiplier achieves reduction in area complexity and area-delay product (ADP) over the existing systolic multipliers for a field of order $m = 163$. The proposed multiplier and some existing multipliers are implemented using ASIC and FPGA technologies and the implementation results shows that the proposed systolic multiplier achieves reduction in area complexity, power consumption, ADP and power-delay product (PDP) over existing multipliers.

In addition, a systolic multiplier architecture over $GF(2^8)$ is designed for irreducible polynomials as an example. The area complexity and delay of the proposed multiplier are estimated and performance comparison with the existing systolic multipliers [20, 23–38] is also presented. The proposed architecture achieves reduction in area complexity and ADP over the best of existing multipliers for $m = 8$. In order to evaluate the performance of the proposed multiplier in a cryptographic application, the Verilog models of Advanced Encryption Standard (AES) and Twofish algorithms are developed employing the proposed systolic multiplier and other multipliers available in the literature. These models are implemented on FPGA device and the device utilization summary shows that the proposed multiplier achieves low area complexity, low power consumption, less ADP and PDP over existing multipliers.

5.2 Proposed Interleaved Multiplication Algorithm

Let $A(x)$ and $B(x)$ be two arbitrary elements of $GF(2^m)$ expressed as

$$A = \sum_{i=0}^{m} a_i x^i \qquad B = \sum_{i=0}^{m} b_i x^i \tag{5.1}$$

Let $C(x) \in GF(2^m)$ be the product polynomial of the two elements $A(x)$ and $B(x)$.

$$\begin{aligned} C(x) &= A(x) \times B(x) \\ &= A(x) \times \sum_{i=0}^{m} b_i x^i \\ &= b_0 A(x) + b_1 x A(x) + b_2 x^2 A(x) + \cdots + b_{m-1} x^{m-1} A(x) \end{aligned} \tag{5.2}$$

It may be observed from Eqn. (5.2) that $C(x)$ is the summation of the multiplication result of b_i and $A(x)x^i$; for all $i = 0, 1, \cdots, m-1$ and the entire summation can be carried out in m iterations. $A(x)x^i$ is calculated in the modular reduction step which is then accumulated in each iteration if $b_i = 1$. On the contrary, $A(x)x^i$ is not considered for the summation if $b_i = 0$. Here, the summation is performed using the exclusive-OR (XOR) operation of each $b_i A(x)x^i$; for all $i = 0, 1, \cdots, m-1$, since the addition is simply an XOR operation over $GF(2)$. Hence, the calculation of $C(x)$ in Eqn. (5.2) can be transformed into Steps 3, 4, 5 & 6 in Algorithm 5.1. Here, $p = (p_{m-1}, \cdots, p_1, p_0)$ acts as the accumulator of $A(x)x^i$ and is initialized to zero at the beginning of each multiplication operation.

The modular reduction of the conventional interleaved multiplication algorithm [54] is performed as shown

$$A(x) = \left(A(x) \times x^i\right) \, mod \, T(x) \tag{5.3}$$

Eqn. (5.3) is evaluated for each i as follows

For $i = 0$:

$$\begin{aligned} C(x) &= A(x) \; mod \; T(x) \\ &= \left(a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}\right) \; mod \; (t_0 + t_1 x + \cdots + t_{m-1} x^{m-1} + x^m) \end{aligned} \tag{5.4}$$

A degree m polynomial cannot modulo divide a degree $(m-1)$ polynomial. Hence, this step can be skipped.

For $i = 1$:

$$\begin{aligned} C(x) &= (A(x) \times x) \; mod \; T(x) \\ &= \left(a_0 + a_1 x^2 + \cdots + a_{m-1} x^m\right) \; mod \; (t_0 + t_1 x + \cdots + t_{m-1} x^{m-1} + x^m) \\ &= a_{m-1} t_0 + (a_{m-1} t_1 + a_0) x + (a_{m-1} t_2 + a_1) x^2 + \cdots + (a_{m-1} t_{m-1} + a_{m-2}) x^{m-1} \end{aligned} \tag{5.5}$$

It can be observed that the summation in Eqn. (5.5) can be performed over m iterations. In each iteration, $T(x)$ and $A(x)x^i$ are XORed and accumulated to the previous result if

$a_{m-1} = 1$; for all $i = 0, 1, \cdots, m-1$. On the contrary, $T(x)$ is not considered in the above accumulation if $a_{m-1} = 0$. Here, $A(x)x^i$ is computed by left shifting $A(x)$ by i times; for all $i = 0, 1, 2, \cdots, m-1$. This result is also used in the polynomial multiplication step given above. Therefore, the modular reduction step can be transformed as Steps 7, 8, 9 & 10 in Algorithm 5.1.

Both the polynomial multiplication and modular reduction steps occur simultaneously resulting in an interleaved algorithm which is presented in Algorithm 5.1.

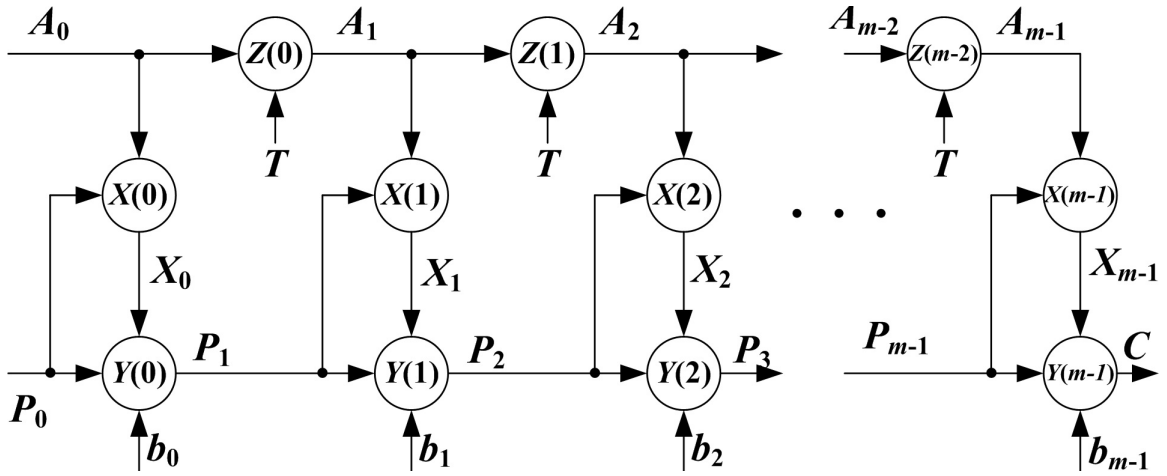
Algorithm 5.1: Proposed interleaved multiplication algorithm over $GF(2^m)$

1 Initialization: $p = 0, counter = 0$
2 FOR $counter = 0$ TO $m - 1$ DO
3 IF($b_0 == 1$)
4 $p = p \oplus a$
5 END IF
6 $b = b \gg 1$
7 $a_{msb} = a_{m-1}$
8 $a = a \ll 1$
9 IF($a_{msb} == 1$)
10 $a = a \oplus t$
11 END IF
12 END FOR

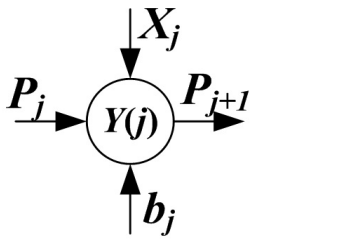
5.3 Proposed Systolic Multiplier Architecture over GF(2^m) for Irreducible Polynomials

This sub-section presents the design of the proposed systolic multiplier architecture over $GF(2^m)$ for irreducible polynomials. The estimations of area complexity and delay of this architecture are computed analytically and compared with the existing multipliers available in the literature. The functionality of the proposed architecture is implemented using FPGA and ASIC technologies. These analytical and implementation results of the proposed architectures and multipliers available in the literature are also presented in the following sub-sections.

5.3.1 Design of Proposed Systolic Multiplier Architecture over GF(2^m) for Irreducible Polynomials

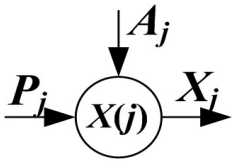


(a) The Signal Flow Graph (SFG)



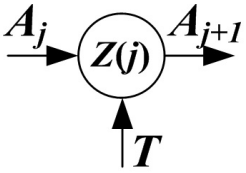
$P_{j+1} = \overline{b}_j \,.\, P_j + b_j \,.\, X_j$

(b) Logic function of $Y(j)$ node



$X_j = P_j + A_j$

(c) Logic function of $X(j)$ node

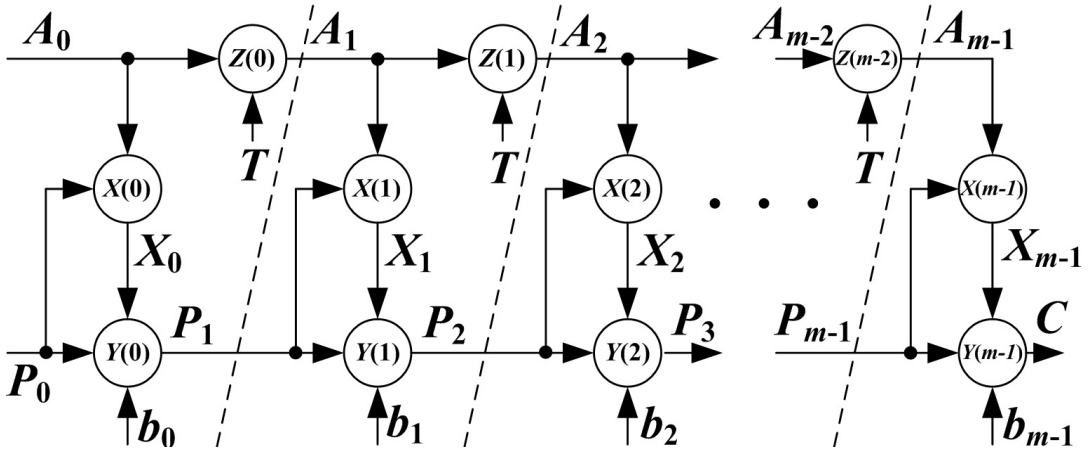


$A_{j+1} = A_j \bmod T$

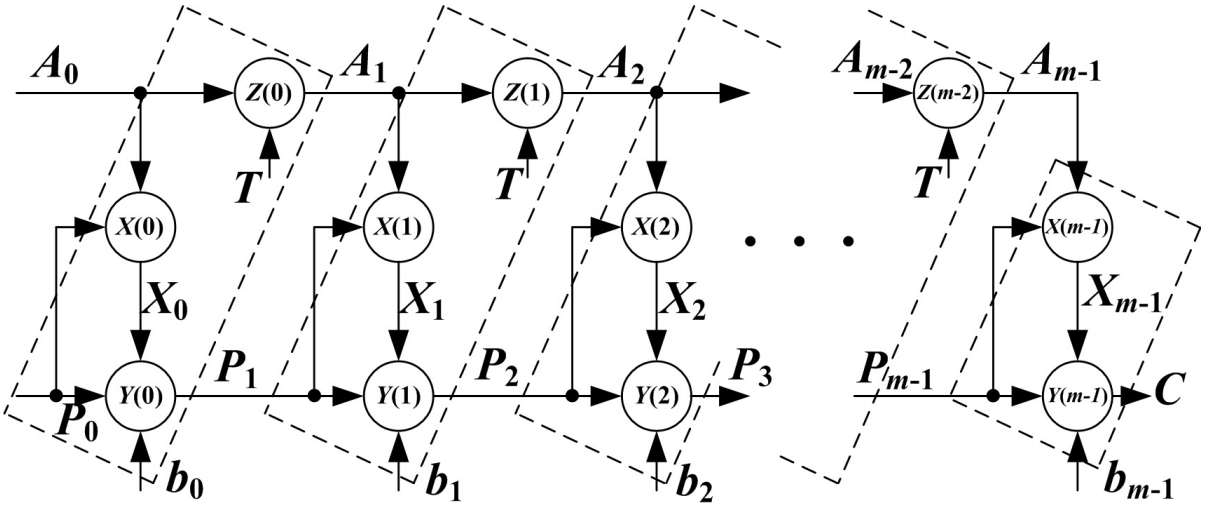
(d) Logic function of $Z(j)$ node

Figure 5.1: SFG derived from the proposed algorithm.

A signal flow graph (SFG) is realized from Algorithm 5.1, as shown in Fig. 5.1(a). The SFG consists of m addition nodes $X(j)$, m decision nodes $Y(j)$, and $(m-1)$ modular reduction nodes $Z(j)$. The logic functionality of these nodes are shown in Fig. 5.1(b)-(d). Here, A_0 is the binary representation of $A(x)$, A_{j+1} is the result of the modular reduction of A_j for the j^{th} iteration, P_{j+1} is the result of the decision node for the j^{th} iteration, X_j is the result of the addition node $X(j)$ for the j^{th} iteration, T is the binary representation of the irreducible polynomial $T(x)$, b_i is the i^{th} coefficient of $B(x)$, and C is the binary representation of the final product $C(x)$. Node $X(j)$ performs a bit-addition operation on the partial results P_j and A_j using the XOR operation. Node $Y(j)$ performs the decision (or selection) operation where it selects between the partial results P_j and X_j using the



(a) Proposed cut-set Retiming

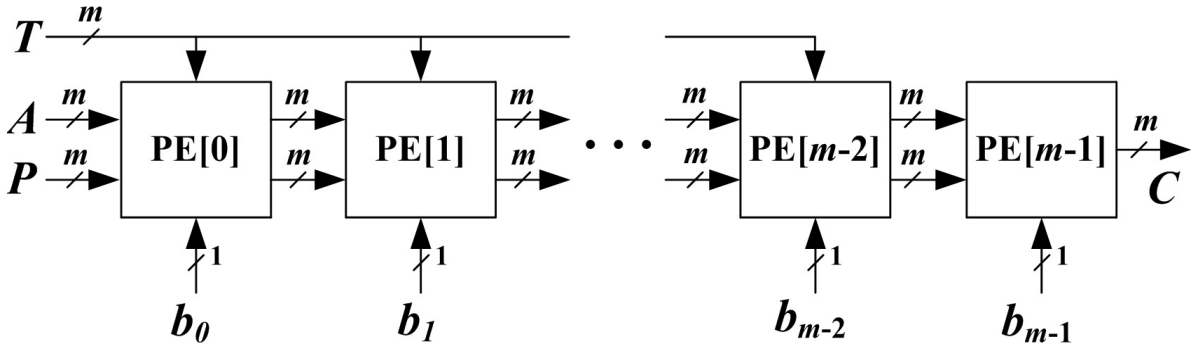


(b) Realization of PE

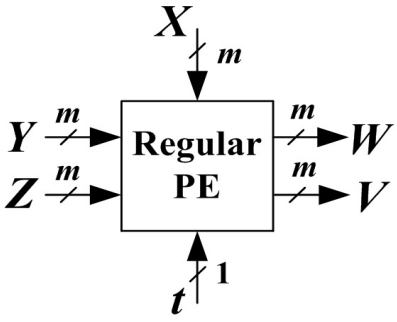
Figure 5.2: Cut-set retiming of the SFG.

selection input b_i. Node $Z(j)$ performs the modular reduction of the degree of A_j by one, and A_{j+1} gives the result. Here, i denotes the index of the coefficient of the polynomial under consideration, and j denotes the iteration count.

Figure 5.2(a) shows the proposed cut-set retiming [59] of the SFG, which is performed to obtain a pipelined structure with reduced critical path delay. The proposed cut-set retiming allows one addition node, one decision node, and one modular reduction node in each iteration of the cut-set, thus enabling a reduced critical path. It also eliminates the data dependency between the addition node and the modular reduction node by performing them in a single iteration. The critical path after the proposed cut-set retiming amounts to $max\{T_{XN}, T_{YN}, T_{ZN}\}$, where T_{XN}, T_{YN}, and T_{ZN} are the computation times of the addition node, decision node, and modular reduction node, respectively.



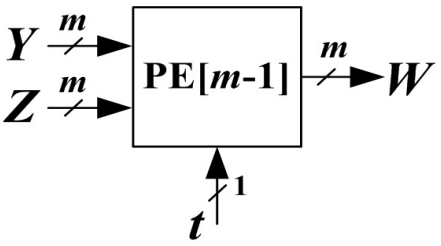
(a) Proposed systolic multiplier



$$W = Z \,.\, \overline{t} + (Y \oplus Z) \,.\, t$$

$$V = Y \,.\, \overline{Y}_{m-1} + (Y \oplus X) \,.\, Y_{m-1}$$

(b) Logic of the regular PE (PE_0 to PE_{m-2})



$$W = Z \,.\, \overline{t} + (Y \oplus Z) \,.\, t$$

(c) Logic of PE_{m-1}

Figure 5.3: Proposed systolic multiplier using PEs realized from the SFG.

Other variations of the cut-set retiming have been considered, and the cut-set retiming proposed in Fig. 5.2(a) is found to provide a good trade-off between the critical path and the latency. Further, the achieved trade-off is found to be comparable to, or better than similar structures that are reported in the literature. Each iteration of the proposed cut-set is wrapped into a single entity called a processing element (PE). Figure 5.2(b) shows the grouping of the nodes of the retimed SFG into PEs based on the proposed cut-set. It can be observed that the PEs obtained from such a grouping of the nodes enables the realization of a regular and modular design consisting of one addition node, one decision node, and one modular reduction node in each PE. The addition node is realized using one XOR operation, the decision node is realized using one 2:1 MUX operation, and the modular reduction node is realized using one XOR operation and one 2:1 MUX operation.

Figure 5.3(a) shows the systolic design consisting of m PEs realized from the proposed cut-set retimed SFG given in Fig. 5.2(b). The $(m-1)$ regular PEs (i.e. PE_0 to

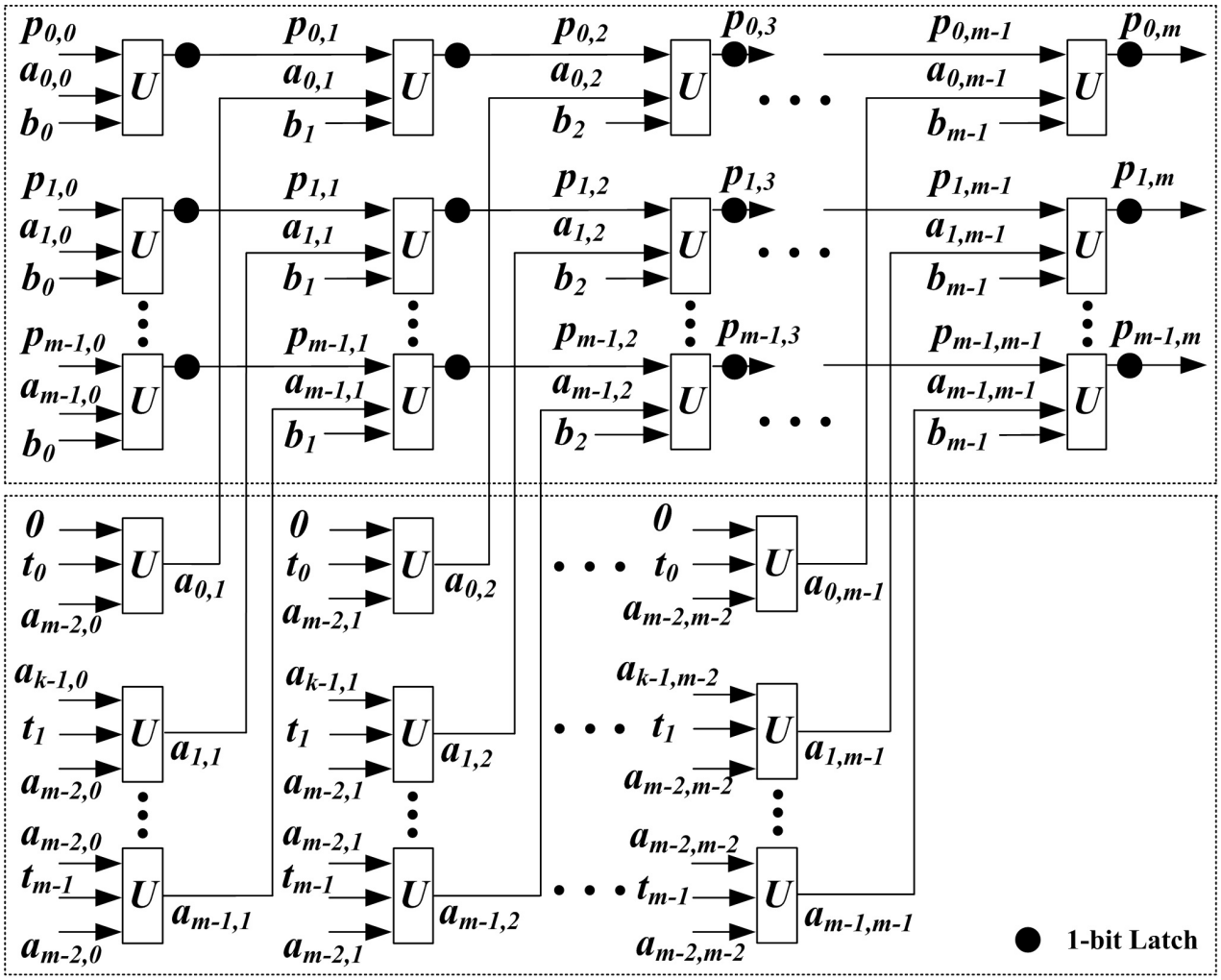


Figure 5.4: Proposed systolic multiplier design using U-cells over $GF(2^m)$ for irreducible polynomials.

PE_{m-2}) perform the addition, decision, and modular reduction operations concurrently, whereas the m^{th} PE (i.e. PE_{m-1}) performs only the addition and decision operations concurrently, which is in accordance with the proposed cut-set retimed SFG. The functions of these two types of PEs are shown in Fig. 5.3(b) & (c). Each regular PE receives A_j, P_j, b_i and T as inputs, and computes the new A_{j+1} and P_{j+1} values for the next iteration. The m^{th} PE receives A_{m-1}, P_{m-1}, and b_{m-1} from the $(m-1)^{\text{th}}$ regular PE, and produces the final result of the finite-field multiplication C. The regular PE and PE_{m-1} are further decomposed into $2m$ U-cells and m U-cells, respectively, to derive a regular, scalable structure, and is much simpler for implementation and optimization. The decomposed systolic structure realized using the U-cells is shown in Fig. 5.4.

The first set of m U-cells (corresponding to the addition and decision nodes) of a regular PE are represented column-wise in the upper block, and the second set of m U-cells (corresponding to the modular reduction nodes) of the same PE are represented column-wise in the lower block. The m U-cells (corresponding to the addition and decision nodes) of the m^{th} PE are represented in the upper block in the rightmost column. The first set of m^2 U-cells perform the polynomial multiplication operation corresponding to Eqn. (5.2), and the second set of $(m^2 - m)$ U-cells perform the modular reduction operation corresponding to Eqn. (5.5). The inputs to each cell are $p_{i,j}, a_{i,j}, b_i$, and $a_{i,j}, t_i, a_{m-2,j}$ for the upper and lower blocks, respectively. The values $p_{i,j}, a_{i,j}$ and t_i are the i^{th} bit values of the m-bit P_j, A_j and T, respectively, and $a_{m-2,j}$ is the $(m-2)^{\text{th}}$ bit value of A_j. Here, i denotes the index of the coefficient of the polynomial under consideration, and j denotes the iteration count. It may be noted that the $a_{i,j}$ in the modular reduction block is left shifted by one bit according to Eqn. (5.5).

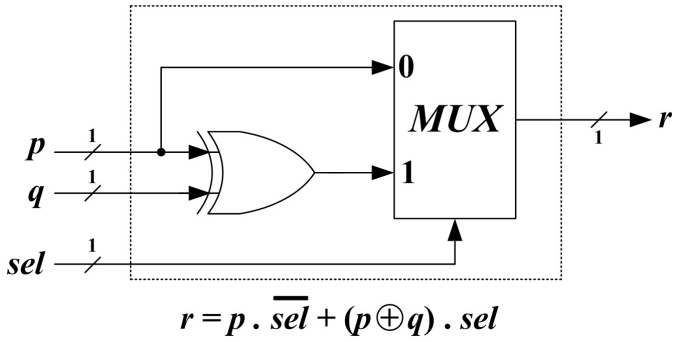


Figure 5.5: Internal circuit detail and logic functionality of U-cell.

The details of the circuit and the function of a U-cell are shown in Fig. 5.5. Each U-cell consists of one XOR gate and one 2:1 MUX. According to Eqn. (5.2), the XOR and MUX in the U-cell for the upper block are derived from the addition node and the decision node, respectively. According to Eqn. (5.5), the XOR and MUX in the U-cell for the lower block are derived from the modular reduction node. It can be observed that the pipelining registers applied for the systolic structure in Fig. 5.4 enable concurrent operations such that the critical path is minimized to $(T_X + T_M)$, where T_X and T_M are the delays of an XOR gate and a 2:1 MUX, respectively. The gate count of the structure is $(2m^2 - m)$ XOR gates, $(2m^2 - m)$ MUX gates, and m^2 1-bit registers. The multiplier produces the first output with an initial latency of m clock cycles followed by one output for every clock cycle. Hence, the throughput is one output per clock cycle, with an initial

latency of m clock cycles.

5.3.2 Analytical Results

The proposed systolic multiplier requires $(2m^2 - m)$ XOR gates, $(2m^2 - m)$ MUXs, and m^2 1-bit registers. The critical path and latency of the proposed systolic multiplier are $(T_X + T_M)$ and m clock cycles, respectively. It may be noted that the proposed multiplier gives one output in every clock cycle, with an initial latency of m clock cycles.

Table 5.1: Area complexity and delay comparison of the systolic multipliers over $GF(2^m)$.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[23]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[24]	$(m^2)^b$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_{3X}$
[25]	$2m^2 - m$	$2m^2$	0	$8m^2 - 7m$	$2m - 1$	$T_A + T_X$
[26]	$2m^2$	$2m^2$	0	$3m^2$	$m + 1$	$T_A + T_X$
[27]	$2m^2$	$2m^2$	0	$3m^2$	$m + 1$	$T_A + T_X$
[28]	$2m^2$	$2m^2$	0	$4m^2$	$2m$	$T_A + T_X$
[29]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[30]	m	$2m^2 + 2m$	$(m^2/2)^a$	$6m^2 + 8m$	$3m/2$	$T_{4M} + T_X$
[31]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[32]	$2m^2 + 3m$	$(m^2 + m)^b$	0	$3m^2 + 4m$	$m + 1$	$T_A + T_{3X}$
[33]a	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	$3m$	$T_A + T_X$
[33]b	m^2	m^2	0	$5m^2$	$4m$	$T_A + T_X$
[34]	$2m^2$	$2m^2$	0	$3m^2$	m	$T_A + T_X$
[35]a	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	$3m$	$T_A + T_X$
[35]b	m^2	m^2	0	$5m^2$	$4m$	$T_A + T_X$
[36]	m	$2m + (m^2/2)^b$	$(m^2 + m/2)^a$	$7m^2$	$3m/2$	$T_{4M} + T_{3X}$
[20]	$m^2 - m + 1$	$m^2 - 1$	$2m^2 + m - 3$	$2m^2 - m$	$2m$	$T_A + T_X$
[37]	$2m^2$	$2m^2$	$2m^2 + m - 3$	$7m^2$	$3m$	$T_A + T_X$
[38]	$2m^2 + 2m$	$2m^2 + 3m$	0	$3m^2 + 4m$	$\lfloor m/2 \rfloor + 1$	$T_A + T_X$
Proposed	0	$2m^2 - m$	$2m^2 - m$	m^2	m	$T_M + T_X$

[a]4-to-1 MUX; [b]3-input XOR gate.

Table 5.1 shows a comparison of the hardware complexity, latency, and critical path of the proposed systolic multiplier with existing systolic multipliers [20, 23–38] available in the literature. Here, T_A, T_X, T_M, T_{3X}, T_{4M} denote the delays of a 2-input AND gate, 2-input XOR gate, 2:1 MUX, 3-input XOR gate, and 4:1 MUX, respectively. The

Table 5.2: Comparison of total transistor count, number of clock cycles, total delay, % reduction in area and % reduction in ADP of the proposed systolic multiplier with existing multipliers over $GF(2^{163})$.

Multipliers	#Transistors	Latency	CP (ns)	Total Delay (ns)	ADP ($\times 10^9$)	%Reduction in Area	%Reduction in ADP
[23]	2125520	489	19	9291	19.75	60	83
[24]	2125520	489	31	15159	32.22	60	90
[25]	2326988	325	19	6175	14.37	63	77
[26]	1275312	164	19	3116	3.97	33	19
[27]	1275312	164	19	3116	3.97	33	19
[28]	1487864	326	19	6194	9.22	42	65
[29]	2125520	489	19	9291	19.75	60	83
[30]	1660644	245	28	6860	11.39	48	71
[31]	2125520	489	19	9291	19.75	60	83
[32]	1285418	164	31	5084	6.54	34	50
[33]a	1175882	489	19	9291	10.93	27	70
[33]b	1541002	489	19	9291	19.09	44	77
[34]	1275312	163	19	3097	3.95	33	19
[35]a	1175882	489	19	9291	10.93	27	70
[35]b	1382566	652	19	12388	17.13	38	81
[36]	2076620	245	40	9800	20.35	59	84
[20]	1061438	326	19	6194	6.58	20	51
[37]	2445308	489	19	9291	22.72	65	85
[38]	1284114	82	19	1558	2	33	37*
Proposed	848252	163	23	3749	3.22	-	-

*%Increase in ADP; CP is the Critical Path Delay.

polynomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ (recommended by NIST) is taken as an example to compare the area complexity and delay of the systolic multipliers available in the literature. Traditional CMOS logic is used to estimate the hardware complexity, wherein the transistor counts are six transistors for a 2-input XOR gate, 2-input AND gate, and a 1-bit 2:1 MUX, 16 transistors for a 1-bit 4:1 MUX, and eight transistors for a 1-bit register [55]. To estimate the delay, real-time circuits from STMicroelectronics [56] are considered, where the typical propagation delays of gates used in the various designs are:

2-input XOR gate (M74HC86) t_{PD} = 12ns, 2-input AND gate (M74HC08) t_{PD} = 7ns, 2:1 MUX (M74HC257) t_{PD} = 11ns, and 4:1 MUX (M74HC153) t_{PD} = 16ns. A 3-input XOR gate is realized using two 2-input XOR gates. Therefore, the hardware complexity and propagation delay of a 3-input XOR gate are estimated as twelve transistors and t_{PD} = 24 ns, respectively.

Table 5.2 shows the hardware complexity (total transistor count), latency (number of clock cycles), critical path delay (CP), total delay, and percentage reduction in the hardware complexity of all the multipliers considered. Here, the total delay is obtained as the product of the latency and critical path delay. From Table 5.2, it may be observed that the proposed multiplier achieves low hardware complexity when compared to existing systolic structures that are available in the literature. Specifically, it achieves reduction in area complexity of about 60%, 60%, 63%, 33%, 33%, 42%, 60%, 48%, 60%, 34%, 27%, 44%, 33%, 27%, 38%, 59%, 20%, 65%, and 33% for $m = 163$ compared to existing multipliers [20, 23–38], respectively. Similarly, the proposed multiplier achieves about 83%, 90%, 77%, 19%, 19%, 65%, 83%, 71%, 83%, 50%, 70%, 77%, 19%, 70%, 81%,84%, 51% and 85% compared to the systolic multipliers [20,23–37]. Moreover, the systolic multiplier [38] requires 37% less ADP compared to the proposed multiplier due to the savings achieved in number of clock cycles. These savings are due to the appropriate Montgomery factor chosen by the authors to attain the lowest possible delay. However, it may be observed that the delay of the systolic multiplier [38] increases for other Montgomery factors and hence the delay comparison with the proposed multiplier is not totally fair. Moreover, the proposed multiplier achieves 33% reduction in area complexity compared to the systolic multiplier [38].

Figs. 5.6 and 5.7 shows the comparison of the area complexity and ADP, respectively, of the proposed systolic multiplier with existing systolic multipliers for field orders $m = 2$ to $m = 600$. From Fig. 5.6, it is observed that the proposed multiplier achieves low area complexity compared to the existing multipliers [20, 23–38]. It can also be observed that the difference in the area complexities between the proposed multiplier and the existing multipliers increases as the order of the finite field increases. Hence, the proposed systolic multiplier achieves better area complexity for higher-order finite fields. Moreover, the ADP comparison depicted in Fig. 5.7 indicates that the proposed multiplier achieves

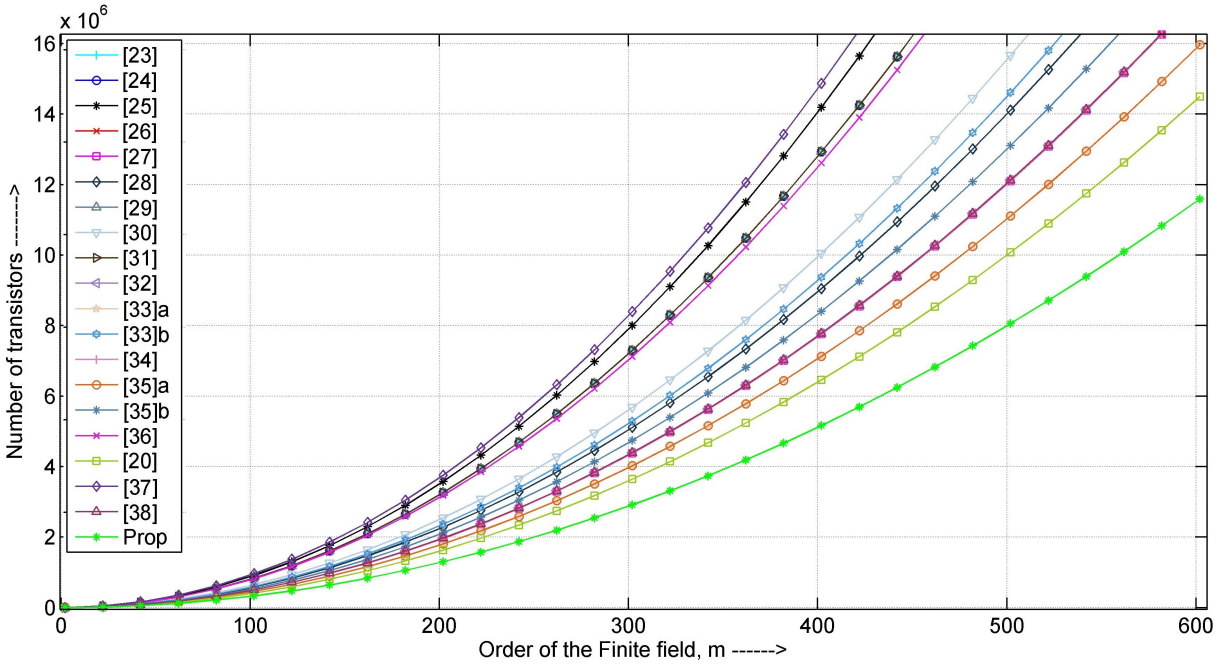


Figure 5.6: Area complexity comparison of sequential multipliers.

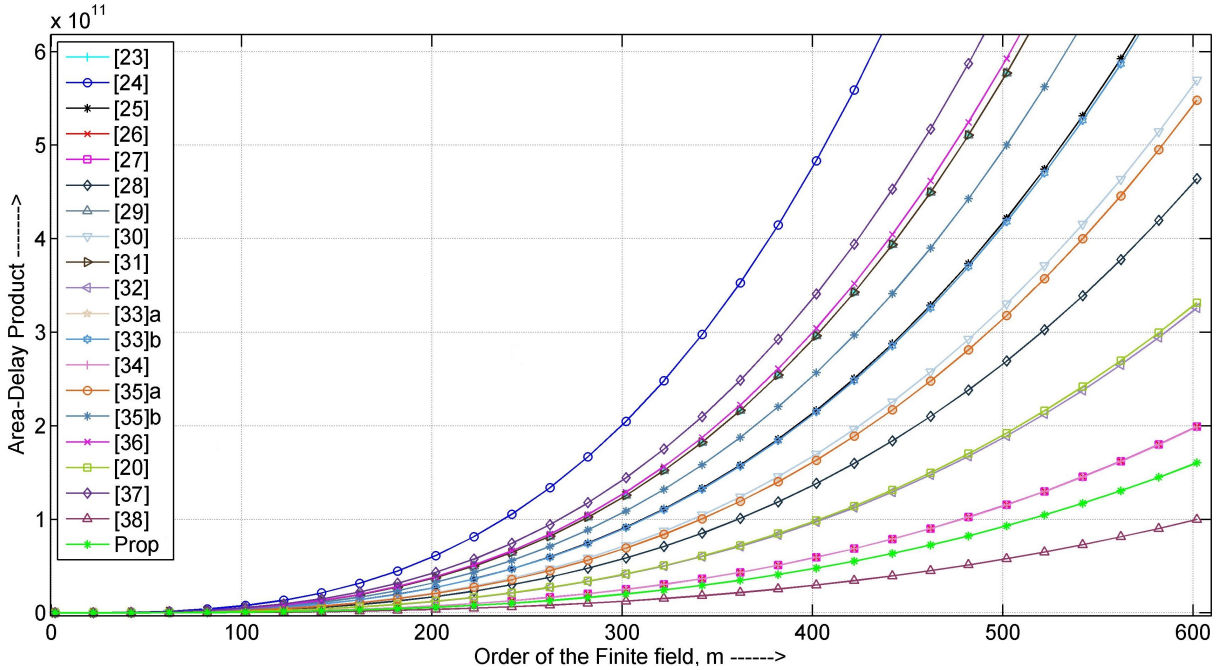


Figure 5.7: Area-Delay Product comparison of sequential multipliers.

reduction in area complexity without much increase in delay.

5.3.3 Implementation Results

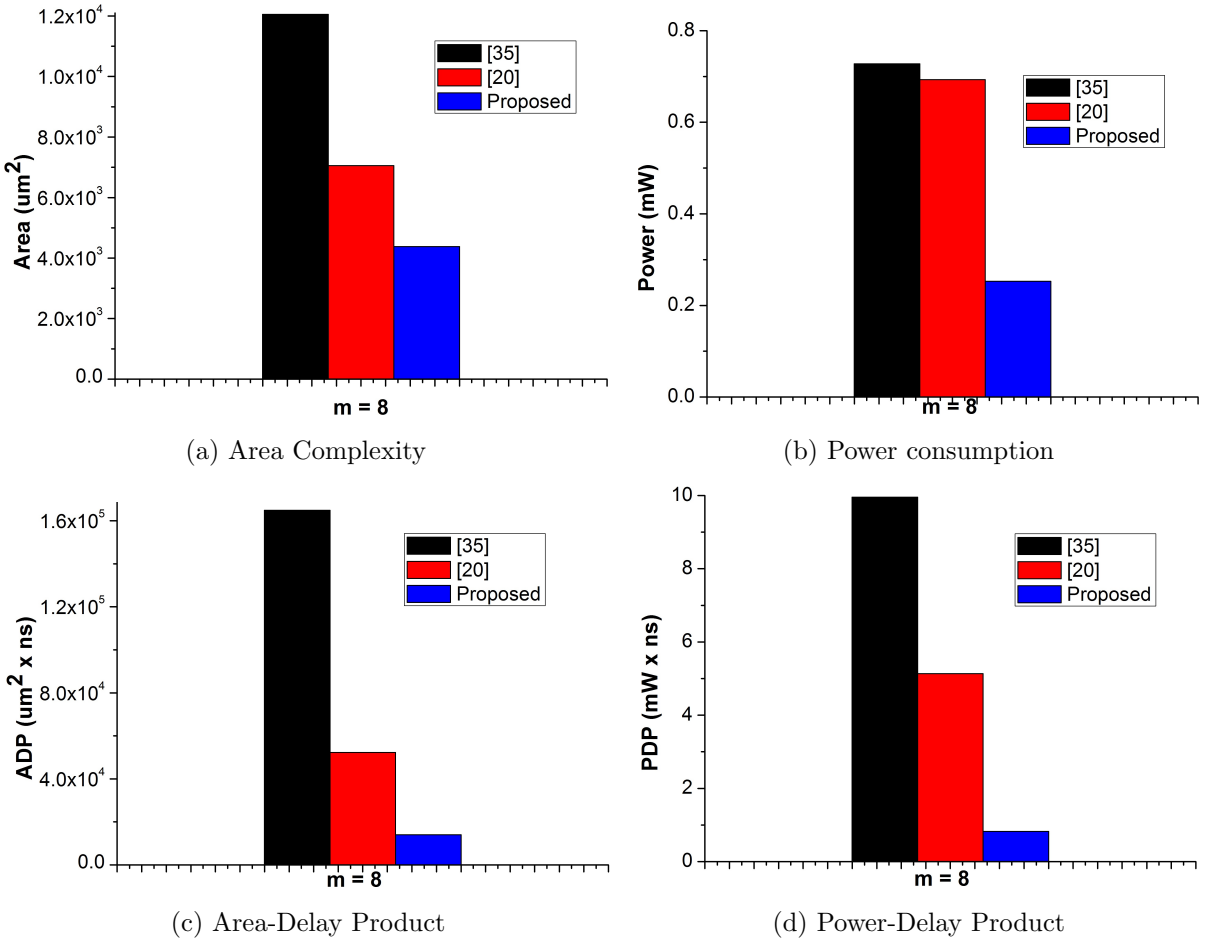
The performance of the proposed systolic multiplier architecture and the systolic multiplier architectures available in the literature are verified by implementing them on ASIC and FPGA platforms. The implementation results of these architectures are presented in the following sub-sections.

5.3.3.1 ASIC Implementation Results

Table 5.3: ASIC implementation results of systolic multipliers.

	Metrics \ Multipliers	[35]	[20]	Proposed
m=8	Total Delay (ns)	13.68	7.41	3.2
	Area (μm^2)($\times 10^3$)	12.055	7.057	4.381
	Power (mW)	0.7277	0.6931	0.2527
	ADP ($\mu m^2 \times ns$)($\times 10^3$)	164.912	52.298	14.019
	PDP ($mW \times ns$)	9.955	5.136	0.823
m=163	Total Delay (ns)	303.51	157.78	65.2
	Area (μm^2)($\times 10^3$)	4953.948	2985.735	1917.372
	Power (mW)	175.5645	167.3967	95.1454
	ADP ($\mu m^2 \times ms$)($\times 10^6$)	1503.573	471.089	125.013
	PDP ($mW \times ns$)($\times 10^3$)	53.286	26.412	6.204

The proposed systolic multiplier and the systolic multipliers [20, 35] are considered for hardware implementations since they require low area complexity compared to the existing systolic multipliers. These multipliers are modelled in Verilog for $m = 8$ and $m = 163$ and synthesized using Synopsys Design Vision Compiler and Synopsys $90nm$ Generic Library. The delay, area complexity, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 5.3). The area complexity, power consumption, ADP and PDP results are also plotted for $m = 8$ and $m = 163$ as shown in Fig. 5.8(a)-(d) and Fig. 5.9(a)-(d), respectively.



(a) Area Complexity

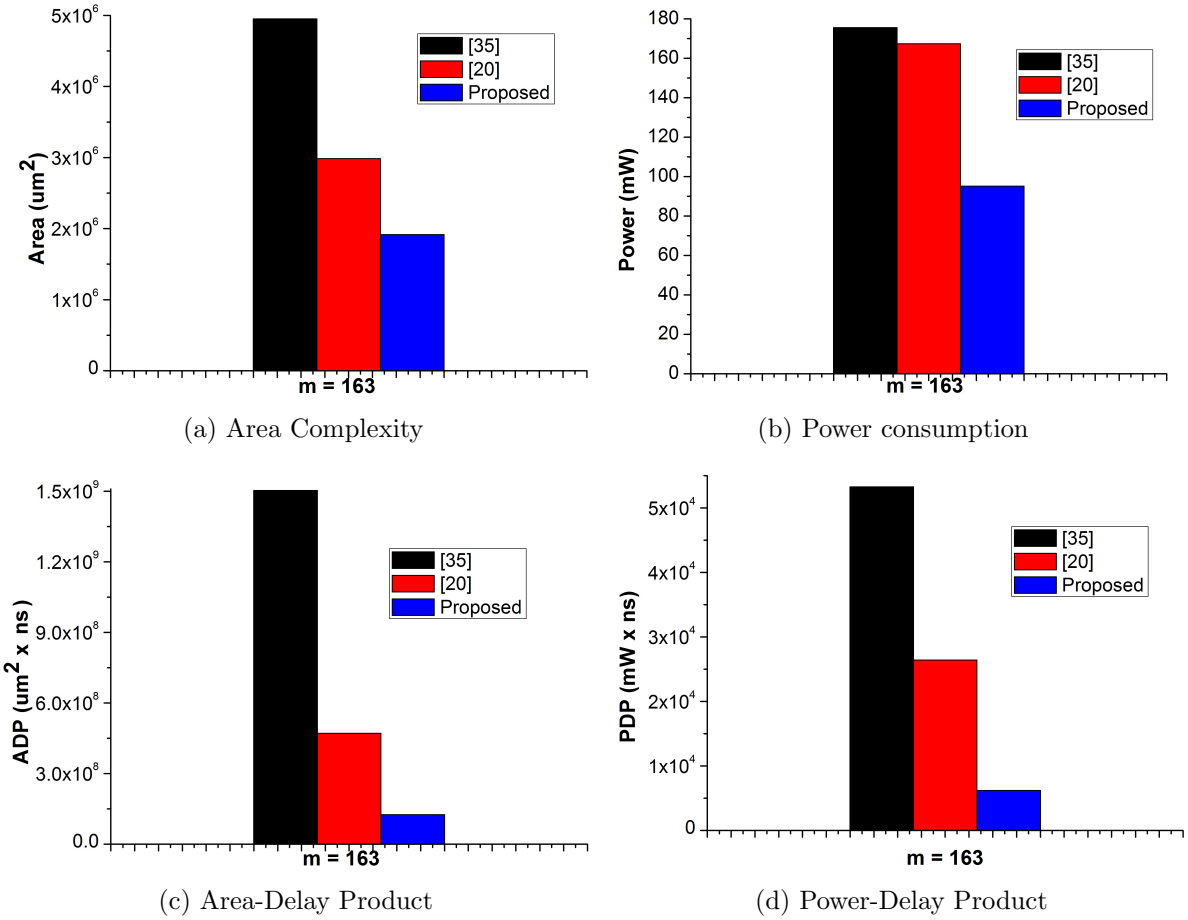
(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 5.8: ASIC implementation results of systolic multipliers for $m = 8$.

It is clear from the histogram (see Fig. 5.8 and Fig. 5.9) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, it can be observed that the proposed multiplier achieves reduction of about 63%, 65%, 91% & 91% in area complexity, power consumption, ADP and PDP, respectively, when compared to the systolic multiplier [35] for $m = 8$. Similarly, the proposed multiplier achieves reduction of about 37%, 62%, 73% & 83% in area complexity, power consumption, ADP and PDP, respectively, when compared to the systolic multiplier [20] for $m = 8$. The proposed multiplier achieves reduction of about 61%, 45%, 91% & 88% in area complexity, power consumption, ADP and PDP, respectively, when compared to the systolic multiplier [35] for $m = 163$. Similarly, the proposed multiplier achieves reduction of about 35%, 43%, 73% & 76% in area complexity, power consumption, ADP and PDP, respectively, when compared to the systolic multiplier [20] for $m = 163$. Moreover, the proposed multiplier also achieves reduction in delay com-



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 5.9: ASIC implementation results of systolic multipliers for $m = 163$.

pared to the systolic multipliers [20, 35] for $m = 163$. These improvements achieved by the proposed multiplier in area complexity, power consumption, ADP and PDP indicates an efficient design in both area and power without much increase in delay.

5.3.3.2 FPGA Implementation Results

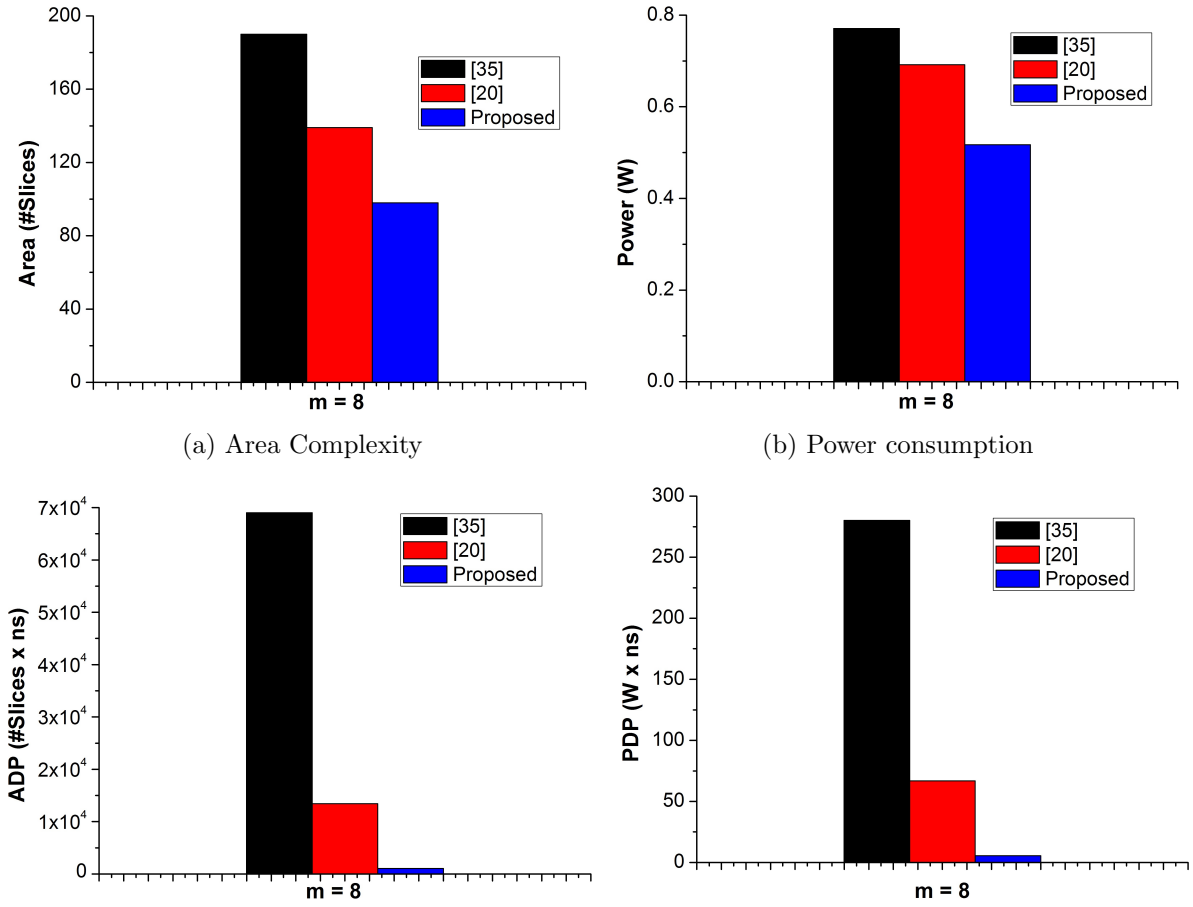
In addition to the ASIC implementation, the functionality of the proposed multiplier is also verified by implementing the Verilog models on FPGA platform. The Verilog models of the proposed multiplier and the multipliers [20,35] are simulated and synthesized using Xilinx Vivado 2014.2 tool. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The delay, area, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 5.4). The area complexity, power consumption, ADP and PDP results are also plotted for $m = 8$ and $m = 163$ as

Table 5.4: FPGA implementation results of systolic multipliers.

	Metrics \ Multipliers	[35]	[20]	Proposed
m=8	Total Delay (ns)	363.36	96.672	10.552
	Area ($\#Slices$)	190	139	98
	Power (W)	0.771	0.692	0.517
	ADP ($\#Slices \times ns$)($\times 10^3$)	69.038	13.437	1.034
	PDP ($W \times ns$)	280.151	66.897	5.455
m=163	Total Delay (ms)	7.403	1.970	0.215
	Area ($\#Slices$)	154635	105787	66434
	Power (W)	3.6	6.187	2.848
	ADP ($\#Slices \times ns$)($\times 10^3$)	1144.763	208.4	14.283
	PDP ($W \times ms$)	26.651	12.188	0.612

shown in Fig. 5.10(a)-(d) and Fig. 5.11(a)-(d), respectively.

It is clear from the histogram (see Fig. 5.10 and Fig. 5.11) that the proposed multiplier achieves low area complexity, power consumption, ADP and PDP among existing multipliers. More specifically, it can be observed that the proposed multiplier achieves reduction of about 48%, 32%, 98% & 98% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [35] for $m = 8$. Similarly, the proposed multiplier achieves reduction of about 29%, 25%, 92% & 91% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [20] for $m = 8$. For $m = 163$, the proposed multiplier achieves reduction of about 57%, 48%, 98% & 97% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [35]. Similarly, the proposed multiplier achieves reduction of about 37%, 53%, 93% & 94% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [20] for $m = 163$. Moreover, the proposed multiplier also achieves reduction in delay compared to the multipliers [20, 35] for $m = 163$. These improvements achieved by the proposed multiplier in area complexity, power consumption, ADP and PDP indicates an efficient design in both area and power without much increase in delay.



(a) Area Complexity

(b) Power consumption

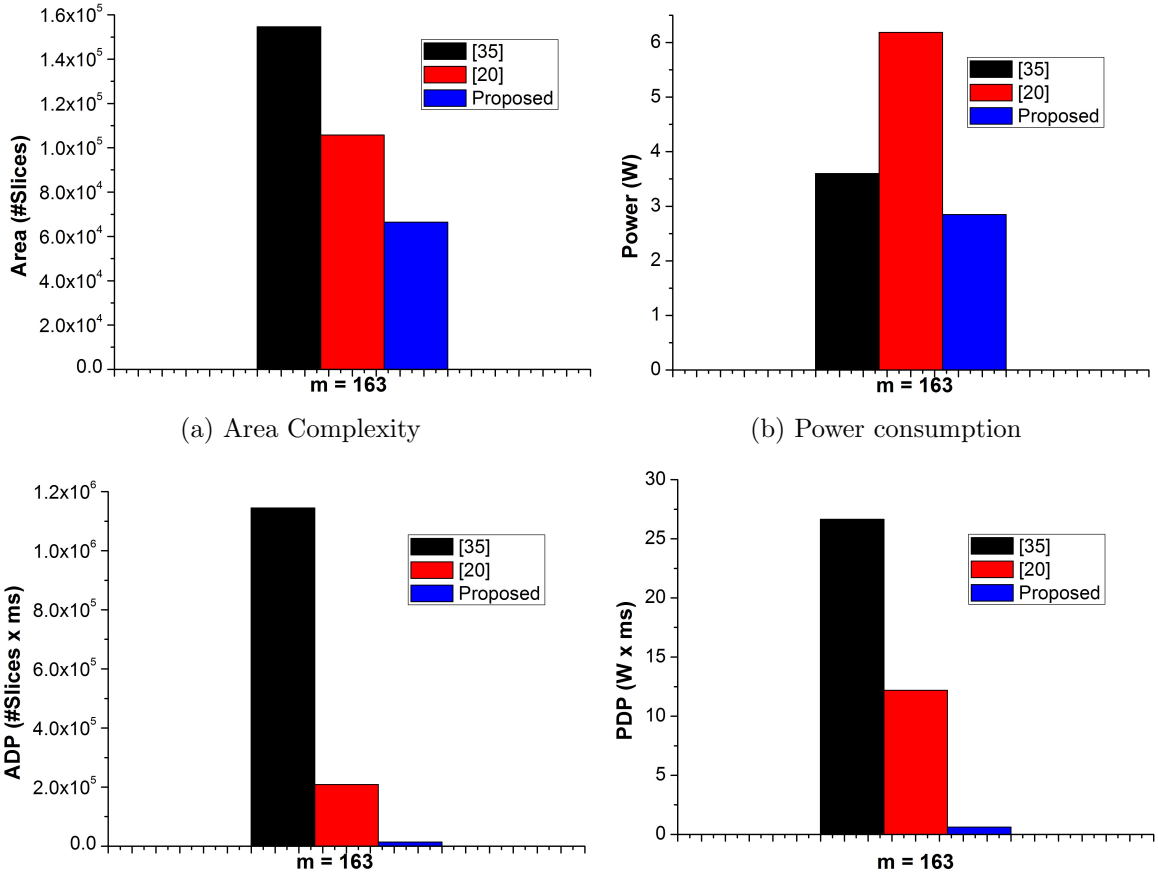
(c) Area-Delay Product

(d) Power-Delay Product

Figure 5.10: FPGA implementation results of systolic multipliers for $m = 8$.

5.4 Proposed Systolic Multiplier Architecture over GF(2^8) for Irreducible Polynomials

This sub-section presents the design of the proposed systolic multiplier architecture over $GF(2^8)$ for irreducible polynomials. AES and Twofish encryption algorithms are realized using the proposed systolic multiplier architecture and implemented on an FPGA platform. These implementation results of the proposed architecture are compared with the results achieved by the multiplier architectures available in the literature.



(a) Area Complexity

(b) Power consumption

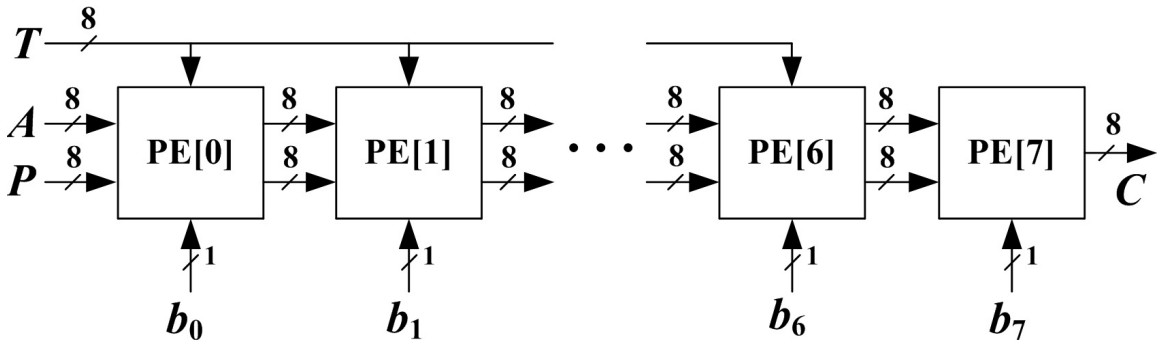
(c) Area-Delay Product

(d) Power-Delay Product

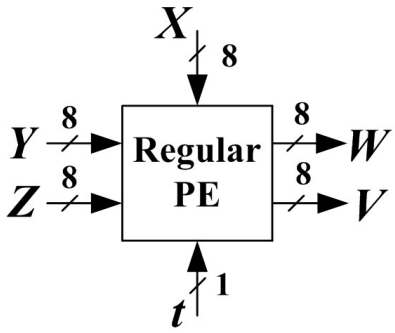
Figure 5.11: FPGA implementation results of systolic multipliers for $m = 163$.

5.4.1 Design of Proposed Systolic Multiplier Architecture over GF(2^8) for Irreducible Polynomials

Figure 5.12(a) shows the proposed systolic multiplier architecture over $GF(2^8)$ derived from the proposed systolic multiplier architecture over $GF(2^m)$ (section 5.3). It consists of 8 PEs, where seven of them are regular PEs (i.e. PE_0 to PE_6) that perform polynomial multiplication and modular reduction operations concurrently whereas the 8^{th} PE (PE_7) performs only the polynomial multiplication operation. The logic functionality of these two types of PEs are shown in Fig. 5.12(b)-(c). PE_7 and the regular PEs are decomposed into 8 V-cells and 16 V-cells, respectively, to derive a more simple, scalable architecture. The proposed systolic multiplier realized using these V-cells is shown in Figure 5.13. Figure 5.14 shows the internal circuit detail and logic function of a V-cell. The first set of 64 V-cells performs the polynomial multiplication operation and the second set of 56 V-cells performs the modular reduction operation. The inputs to each cell are



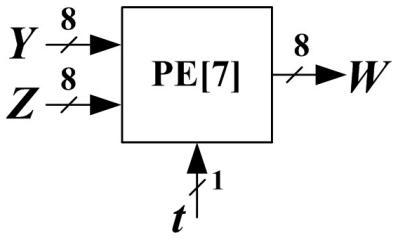
(a) Proposed systolic multiplier



$$W = Z \,.\, \overline{t} + (Y \oplus Z) \,.\, t$$

$$V = Y \,.\, \overline{Y}_{m-1} + (Y \oplus X) \,.\, Y_{m-1}$$

(b) Circuit detail and function of Regular PE (PE_0 to PE_6)



$$W = Z \,.\, \overline{t} + (Y \oplus Z) \,.\, t$$

(c) Circuit detail and function of PE_7

Figure 5.12: Proposed systolic multiplier realized using PEs.

$p_{i,j}, a_{i,j}, b_i$ and $a_{i,j}, t_i, a_{6,j}$, respectively; where i denotes the index of the coefficient of the polynomial under consideration and j denotes iteration count.

5.4.2 Analytical Results

Since the V-cell consists of an Exclusive-OR (XOR) gate and a 2:1 Multiplexer (MUX), the gate count of the entire proposed structure can be computed as 120 XOR gates, 120 MUX gates and 64 registers. The critical path is given by the expression $(T_X + T_M)$, where T_X and T_M are the delays of the XOR gate and the 2:1 MUX respectively. The total clock cycles required by the multiplier to give the first output is 8 and thereafter gives outputs for every clock cycle. Therefore, the throughput is 1 with an initial latency of 8 clock cycles.

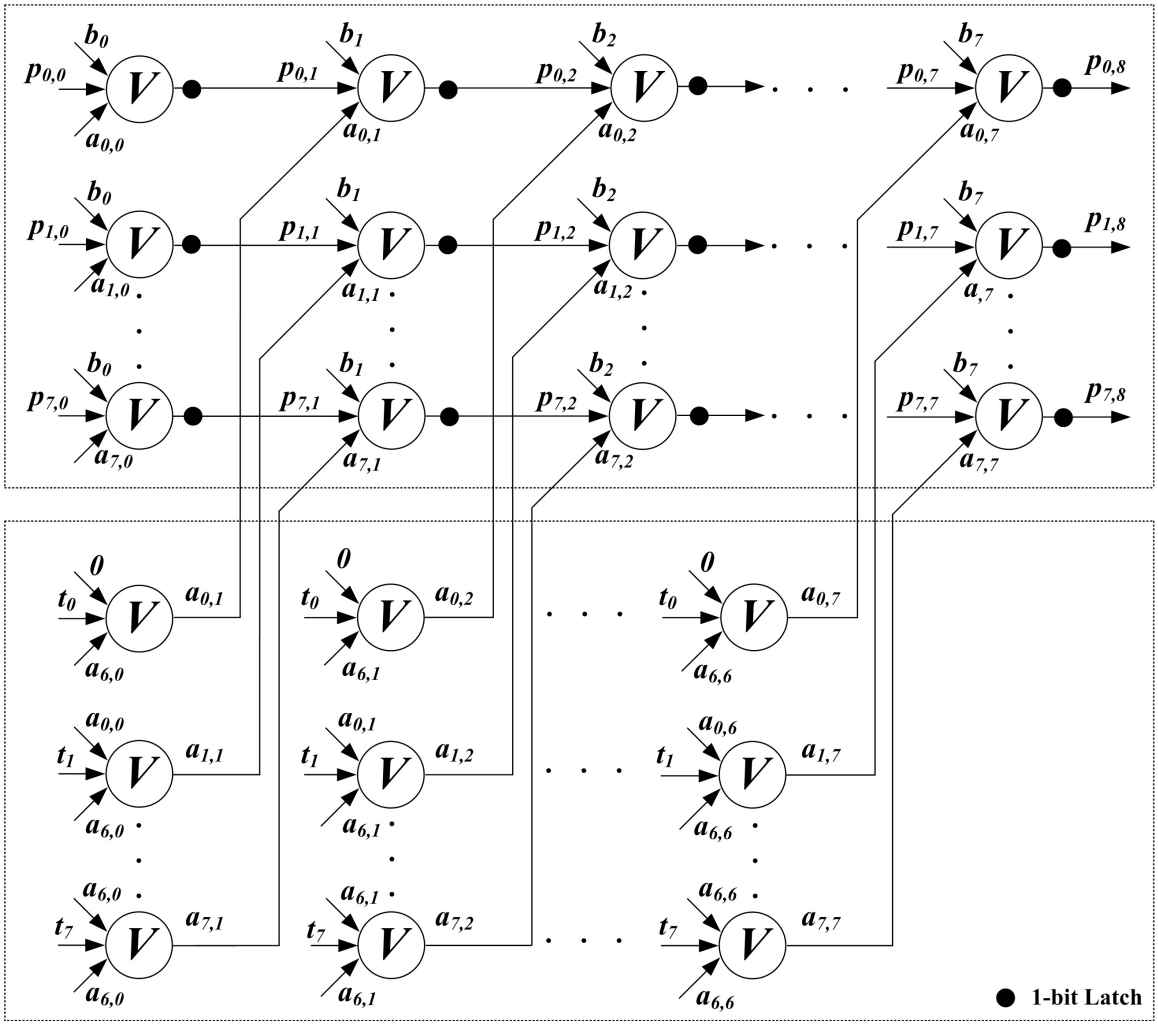


Figure 5.13: Proposed systolic multiplier design using V-cells over $GF(2^8)$ for irreducible polynomials.

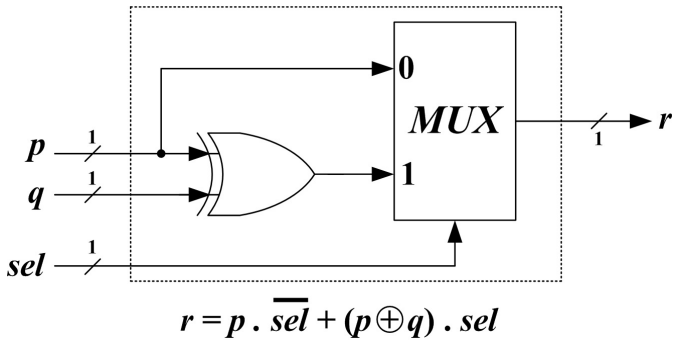


Figure 5.14: Internal circuit detail and logic functionality of V-cell.

Table 5.5 presents the comparison of number of gates, latency and critical path of the proposed architecture with other systolic multiplier architectures [20, 23–38] available in the literature. It may be noted that T_A, T_{3X} and T_{4M} denotes the delay of a 2-input

Table 5.5: Area complexity and delay comparison of the systolic multipliers over $GF(2^8)$.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[23]	128	128	0	448	24	$T_A + T_X$
[24]	$(64)^b$	128	0	448	24	$T_A + T_{3X}$
[25]	120	128	0	456	15	$T_A + T_X$
[26]	128	128	0	192	9	$T_A + T_X$
[27]	128	128	0	192	9	$T_A + T_X$
[28]	128	128	0	256	16	$T_A + T_X$
[29]	128	128	0	448	24	$T_A + T_X$
[30]	8	144	$(32)^a$	448	12	$T_{4M} + T_X$
[31]	128	128	0	448	24	$T_A + T_X$
[32]	152	$(72)^b$	0	224	9	$T_A + T_{3X}$
[33]a	64	80	0	280	24	$T_A + T_X$
[33]b	64	64	0	320	32	$T_A + T_X$
[34]	128	128	0	192	8	$T_A + T_X$
[35]a	64	80	0	280	24	$T_A + T_X$
[35]b	64	64	0	320	32	$T_A + T_X$
[36]	8	$16 + (32)^b$	$(68)^a$	448	12	$T_{4M} + T_{3X}$
[20]	57	63	133	120	16	$T_A + T_X$
[37]	128	128	133	448	24	$T_A + T_X$
[38]	144	152	0	224	5	$T_A + T_X$
Proposed	0	120	120	64	8	$T_M + T_X$

[a]4-to-1 MUX; [b]3-input XOR gate.

AND gate, 3-input XOR gate and 4:1 multiplexer (MUX), respectively. The comparison of area complexity in terms of gate count cannot provide a clear difference among the multipliers considered. A better hardware complexity comparison can be achieved using the transistor count parameter. Moreover, latency alone cannot achieve a fair comparison of computation time; total delay as a product of latency and critical path must be considered. In order to estimate the area complexity, traditional CMOS logic is used wherein the transistor counts are six transistors for 2-input XOR gate, 2-input AND gate, 1-bit 2:1 MUX, sixteen transistors for 1-bit 4:1 MUX, eight transistors for a 1-bit register.

Table 5.6: Comparison of total transistor count, number of clock cycles, total delay, % reduction in area and % reduction in ADP of the proposed systolic multiplier with existing multipliers over $GF(2^8)$.

Multipliers	#Transistors	Latency	CP (ns)	Total Delay (ns)	ADP ($\times 10^6$)	%Reduction in Area	%Reduction in ADP
[23]	5120	24	19	456	2.33	61	84
[24]	5632	24	31	744	4.19	65	91
[25]	5088	15	19	285	1.45	61	75
[26]	3072	9	19	171	0.53	36	32
[27]	3072	9	19	171	0.53	36	32
[28]	3584	16	19	304	1.09	45	67
[29]	5120	24	19	456	2.33	61	84
[30]	4624	12	28	336	1.55	57	76
[31]	5120	24	19	456	2.33	61	84
[32]	4144	9	31	279	1.16	52	69
[33]a	3152	24	19	456	1.44	38	75
[33]b	3376	32	19	608	2.05	42	82
[34]	3072	8	19	152	0.47	36	23
[35]a	3152	24	19	456	1.44	38	75
[35]b	3376	32	19	608	2.05	42	82
[36]	5456	12	40	480	2.62	64	86
[20]	2478	16	19	304	0.75	21	52
[37]	5918	24	19	456	2.7	67	86
[38]	3568	5	19	95	0.34	45	5*
Proposed	1952	8	23	184	0.36	-	-

*%Increase in ADP; CP is the Critical Path Delay.

Real-time circuits from STMicroelectronics are considered to estimate the delay using the typical propagation delays of gates; XOR gate t_{PD} = 12ns (M74HC86), AND gate t_{PD} = 7ns (M74HC08), 2:1 MUX t_{PD} = 11ns (M74HC257), 4:1 MUX t_{PD} = 16ns (M74HC153). The 3-input XOR gate can be realized by two 2-input XOR gates. Hence, the propagation delay is computed as t_{PD} = 24ns and the transistor count is twelve. Table 5.6 shows the area complexity, latency, critical path delay, total delay and ADP of the proposed systolic multiplier compared to the systolic multipliers [20, 23–38] available in

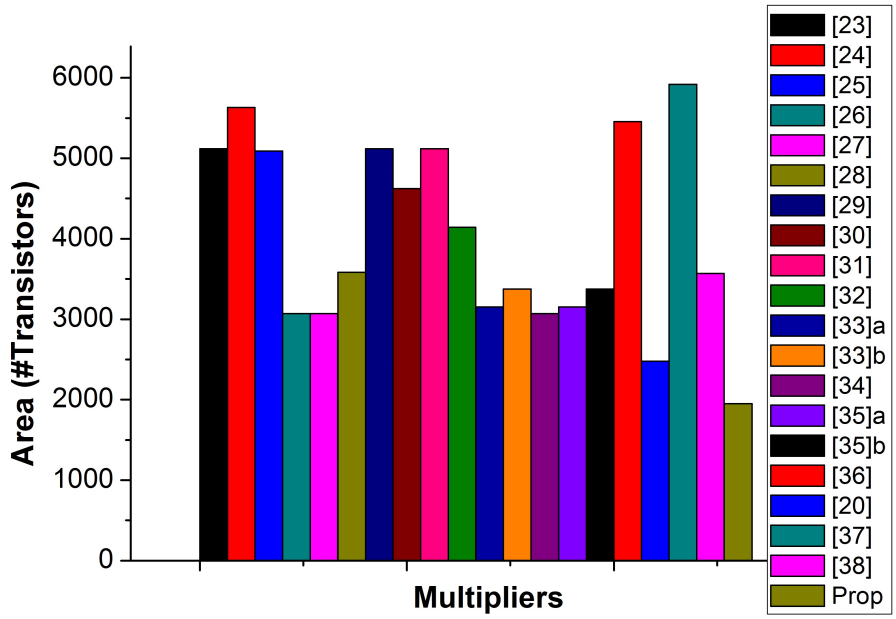


Figure 5.15: Area complexity comparison of systolic multipliers over $GF(2^8)$

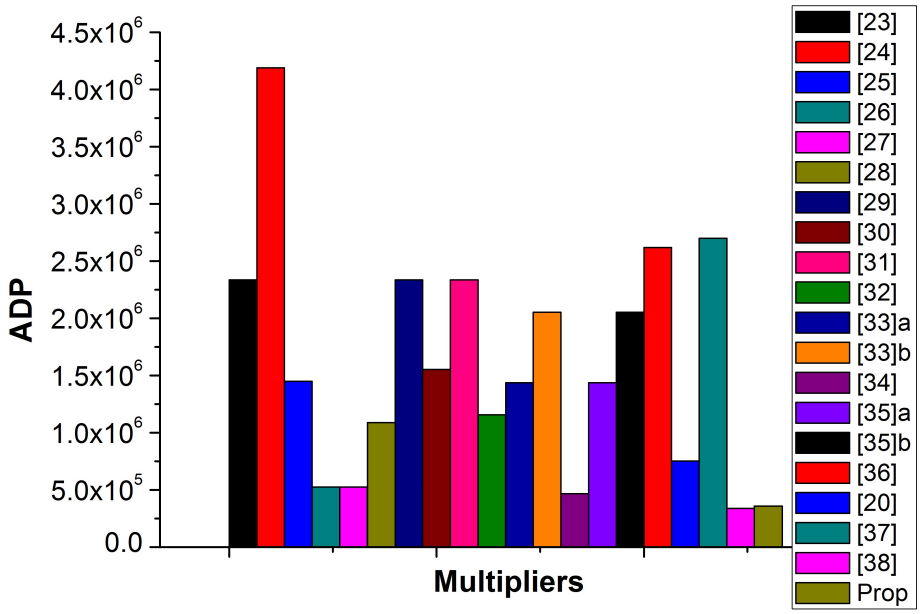


Figure 5.16: Area-Delay Product comparison of systolic multipliers over $GF(2^8)$

the literature for $m = 8$.

Fig. 5.15 and Fig. 5.16 illustrates the histograms plotted for area complexity and ADP, respectively, of the proposed multiplier and other systolic multipliers available in the literature. From Table 5.6, it can be observed that the proposed multiplier achieves low area complexity compared to other systolic designs. Specifically, it achieves about 61%, 65%, 61%, 36%, 36%, 45%, 61%, 57%, 61%, 52%, 38%, 42%, 36%, 38%, 42%,

64%, 21%, 67%, and 45% reduction in hardware complexity when compared with existing multipliers [20,23–38], respectively. Similarly, the proposed multiplier achieves about 84%, 91%, 75%, 32%, 32%, 67%, 84%, 76%, 84%, 69%, 75%, 82%, 23%, 75%, 82%, 86%, 52% and 86% compared to the multipliers [20, 23–37]. Moreover, the multiplier [38] achieves 5% less ADP compared to the proposed multiplier due to the savings achieved in number of clock cycles. However, the decrease in number clock cycles puts extra requirement of hardware and hence the proposed multiplier achieves very low area complexity compared to the multiplier [38]. The improvement achieved in area complexity and also in ADP indicates that the proposed multiplier is an efficient design in terms of area and without much increase in delay.

5.4.3 Implementation Results

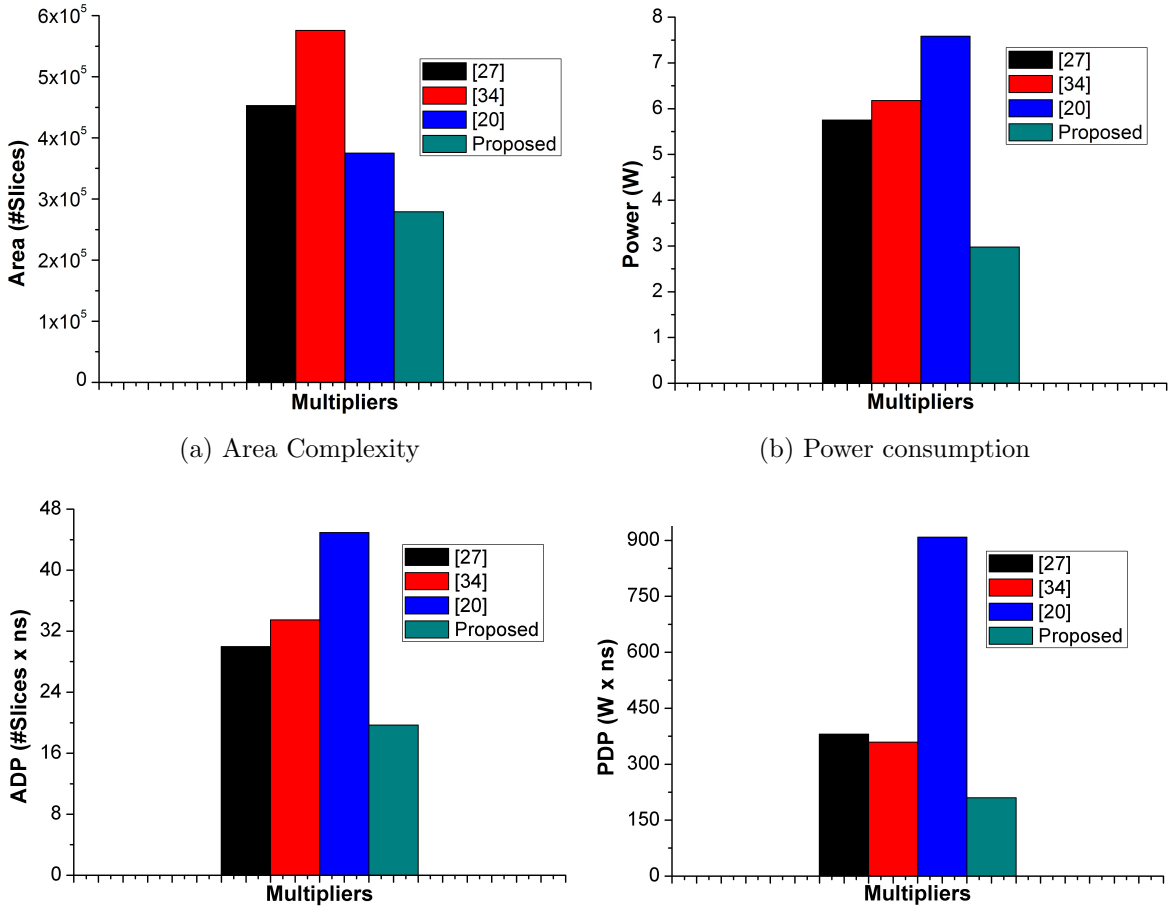
The performance of the proposed systolic multiplier architecture is verified by designing AES and Twofish cryptographic algorithms and implementing them on FPGA platform. The implementation results of these two algorithms are presented in the following sub-sections.

5.4.3.1 FPGA implementation of AES

Table 5.7: FPGA implementation results of AES.

Metrics \ Multipliers	[27]	[34]	[20]	Proposed
Total Delay (ns)	66.175	58.159	119.856	70.557
Area ($\#Slices$)	452871	575882	374933	279055
Power (W)	5.749	6.177	7.583	2.976
ADP ($\#Slices \times ns$)($\times 10^6$)	29.97	33.49	44.94	19.69
PDP ($W \times ns$)	380.44	359.25	908.87	209.98

The proposed sequential multiplier architecture over $GF(2^8)$ is employed to realize the AES algorithm and implemented on FPGA platform. The AES is a symmetric-key cryptographic algorithm developed based on a substitution-permutation structure using



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 5.17: FPGA implementation results of AES.

block-cipher technique. The block size of the plaintext to be encrypted is 128 bits with the key size options of 128, 192 and 256 bits. In this work, the key size of 128 bits is used for the hardware implementation on the FPGA device [57] and hence the algorithm is performed for 10 rounds. The Verilog models for AES encryptor-decryptor are developed employing the proposed sequential multiplier and the sequential multipliers [20, 27, 34] available in the literature to perform finite field multiplications. These Verilog models are simulated and synthesized using Xilinx Vivado 2014.2 software tool to verify their functionality. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The experimental setup of the FPGA implementation of AES is same as described in the previous chapter. The AES encryption and decryption is performed with Plaintext and Key values as 0x00112233445566778899AABBCCDDEEFF and 0x000102030405060708090A0B0C0D0E0F and the Ciphertext obtained is 0x69C4E 0D86A7B0430D8CDB78070B4C55A.

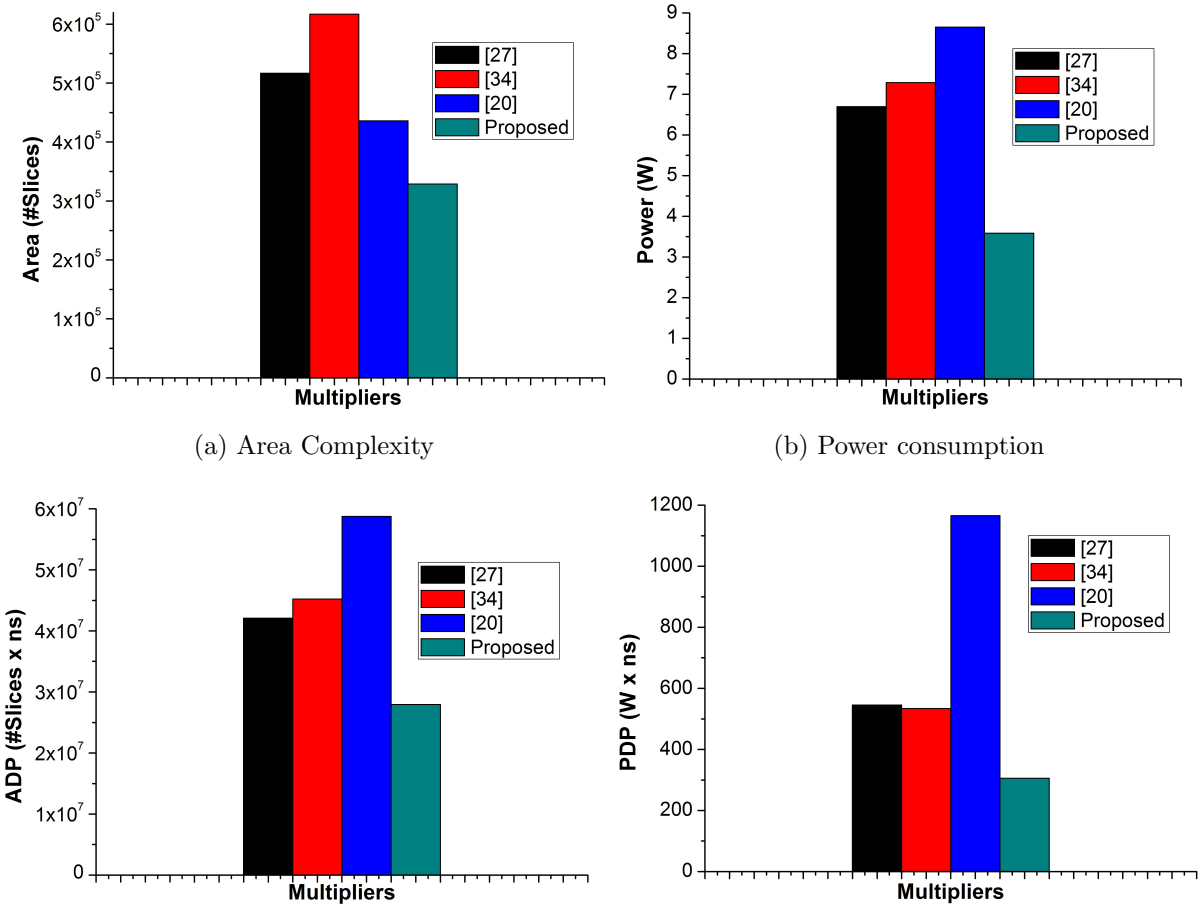
The delay, area, power consumption, ADP and PDP results are computed using the device utilization summary and presented in Table 5.7. The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 5.17(a)-(d), respectively. It is clear from the histogram (see Fig. 5.17) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, it can be observed that the proposed systolic multiplier achieves about 48%, 51% & 60% less power and about 38%, 51% & 25% less area when compared to the systolic multipliers [20,27,34], respectively. Moreover, the proposed multiplier achieves about 34%, 41% & 56% less ADP and about 44%, 41% & 76% less PDP when compared to the systolic multipliers [20, 27, 34], respectively. These improvements achieved by the proposed multiplier in area complexity, power consumption, ADP and PDP indicates an efficient design in both area and power without much increase in delay.

5.4.3.2 FPGA implementation of Twofish

Table 5.8: FPGA implementation results of Twofish.

Metrics \ Multipliers	[27]	[34]	[20]	Proposed
Total Delay (ns)	81.471	73.311	134.758	85.117
Area ($\#Slices$)	516712	616823	435979	328553
Power (W)	6.699	7.286	8.651	3.589
ADP ($\#Slices \times ns$)($\times 10^6$)	42.1	45.22	58.75	27.97
PDP ($W \times ns$)	545.77	534.14	1165.79	305.49

The Twofish cryptographic algorithm is developed based on Fiestel structure and the block cipher encryption technique. The block size of the plaintext to be encrypted is 128 bits with the key size options of 128, 192 and 256 bits. In this work, the key size of 128 bits is used for the hardware implementation on the FPGA device [58]. The Verilog models for Twofish encryptor-decryptor are developed employing the proposed sequential multiplier and the sequential multipliers [20, 27, 34] available in the literature to perform finite field multiplications. These Verilog models are simulated and synthesized using Xilinx Vivado 2014.2 software tool to verify their functionality. The syn-



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 5.18: FPGA implementation results of Twofish.

thesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The experimental setup of the FPGA implementation of Twofish is same as described in the previous chapter. The Twofish encryption and decryption is performed with Plaintext and Key values as 0x00112233445566778899AABBCCDDEEFF and 0x000102030405060708090A0B0C0D0E0F and the Ciphertext obtained is 0x1242FAE 0702A08D0903708274A6831D7.

The delay, area, power consumption, ADP and PDP results are computed using the device utilization summary and presented in Table 5.8. The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 5.18(a)-(d), respectively. It is clear from the histogram (see Fig. 5.18) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, it can be observed that the proposed systolic multiplier achieves about 46%, 50% & 58% less power and about 36%, 46% & 24% less area when compared

to the systolic multipliers [20, 27, 34], respectively. Moreover, the proposed multiplier achieves about 33%, 38% & 52% less ADP and about 44%, 42% & 73% less PDP when compared to systolic multipliers [20,27,34], respectively. These improvements achieved by the proposed multiplier in area complexity, power consumption, ADP and PDP indicates an efficient design in both area and power without much increase in delay.

5.5 Conclusion

In this chapter, a systolic polynomial basis multiplier architecture over $GF(2^m)$ is realized for the proposed algorithm. The area complexities and delay of the proposed multiplier are estimated and performance is compared with other systolic polynomial basis multipliers available in the literature. It may be concluded from the comparisons of the estimated results that the proposed architecture over $GF(2^m)$ for irreducible polynomials achieves low area complexity compared to the existing multipliers. The area-delay product of the proposed multiplier is also low when compared to other multipliers, indicating an efficient multiplier design in terms of both area and delay. From the ASIC and FPGA synthesis results of the multipliers, it can be concluded that the proposed systolic multiplier achieves low area complexity, power consumption, area-delay product and power-delay product compared to the existing multipliers. A systolic multiplier architecture over $GF(2^8)$ is derived from the sequential multiplier architecture over $GF(2^m)$. The area complexity and delay of the proposed multiplier are estimated and performance is compared with other systolic multipliers available in the literature. The Verilog models of two cryptographic algorithms, AES and Twofish, are developed employing the proposed multiplier and the multipliers available in the literature. From the FPGA synthesis results of AES and Twofish algorithms realized using the proposed multiplier, it can be concluded that the proposed multiplier achieves low area complexity, power consumption, area-delay product and power-delay product compared to the existing multipliers. The next chapter presents the design of the proposed systolic multipliers for special classes of polynomials, i.e. trinomials and pentanomials.

Chapter 6

Low-power and Area-Efficient Systolic Multipliers for Special Classes of Irreducible Polynomials

This chapter presents the Pre-Computation (PC) technique proposed to reduce the computational complexity of the interleaved multiplication algorithm presented in chapter 5. The proposed PC technique can be applied for any special classes of irreducible polynomials such as trinomials, pentanomials, all-one polynomials and equally-spaced polynomials. However, since trinomials and pentanomials are widely used for real-time applications, systolic multiplier architectures for these special polynomials are developed. Two systolic multiplier architectures over $GF(2^m)$ for irreducible trinomials and pentanomials, respectively, are designed based on the proposed PC technique. The performance of these proposed systolic multiplier architectures are computed analytically and compared with their respective multiplier architectures available in the literature. In addition, the functionality of these proposed architectures are verified by implementing on FPGA prototype board using Xilinx Vivado Design Suite. The proposed architectures are also synthesized on ASIC using Synopsys Design Vision Compiler for $90nm$ technology library. These implementation results are compared with the results obtained for the systolic architectures available in the literature.

6.1 Introduction

In polynomial basis representation, extended binary fields $GF(2^m)$ are generated using irreducible polynomials. An irreducible polynomial in the field $GF(2^m)$ is defined as a polynomial that cannot be factored into two or more polynomials. These irreducible polynomials can be classified into equally spaced polynomials, all-one polynomials, trinomials and pentanomials. The equally spaced polynomials and all-one polynomials are not widely used due to their scarcity. However, up to 5148 irreducible trinomials were identified for field orders of $m \leq 10,000$ [39] which is approximately half of the m values and pentanomials exist for all the fields in which trinomials are absent. Moreover, efficient realization of hardware structures is possible when employing trinomials and pentanomials. Hence, multipliers designed for trinomials and pentanomials are recommended by NIST for use in cryptographic applications and hence there is a need to design efficient multiplier architectures for these special classes of polynomials.

In the previous chapter, a modified interleaved multiplication algorithm is derived for performing multiplications over $GF(2^m)$ for irreducible polynomials. The computational complexity of this multiplication algorithm for irreducible trinomials and pentanomials can be further reduced by employing a novel pre-computation (PC) technique proposed in this chapter. Subsequently, two systolic multiplier architectures over $GF(2^m)$ for irreducible trinomials and pentanomials are developed for the proposed algorithm. The area complexity and delay of the two proposed systolic multipliers is estimated and their performance is compared with systolic multipliers available in the literature for trinomials and pentanomials. It is observed that the proposed systolic multiplier for trinomials achieves reduction in area complexity and ADP compared to the systolic multipliers [40–49] for the trinomial $x^{233} + x^{74} + 1$. Similarly, the proposed systolic multiplier for pentanomials achieves reduction in area complexity compared to the systolic multipliers [47, 50–52] for the pentanomial $x^{283} + x^{12} + x^7 + x^5 + 1$. These trinomials and pentanomials are recommended by NIST to be used in cryptographic applications. In order to verify the functionality, the proposed multipliers and some of the existing multipliers are implemented on ASIC and FPGA technologies.

6.2 Proposed Interleaved Multiplication Algorithm for Irreducible Trinomials and Pentanomials

In this sub-section, the modified interleaved multiplication algorithm employing the proposed Pre-Computation (PC) technique is presented. The generic expressions of trinomials and pentanomials that are used for the derivation of the proposed algorithm are $f_1(x) = x^m + x^k + 1$ and $f_2(x) = x^m + x^{k_1} + x^{k_2} + x^{k_3} + 1$.

Algorithm 6.1: Interleaved multiplication algorithm over $GF(2^m)$ for irreducible polynomials

1 Inputs: A_j, B_j, T
2 Result: P_m
3 Initialization: $P_0 = \{0, 0, 0, \cdots, 0\}$
4 FOR $j = 0$ TO $m - 1$ DO
5 IF($b_j == 1$)
6 $P_{j+1} = P_j \oplus A_j$
7 ELSE
8 $P_{j+1} = P_j$
9 END IF
10 $\hat{A}_j = A_j \ll 1$
11 IF($a_{m-1,j} == 1$)
12 $A_{j+1} = \hat{A}_j \oplus T$
13 ELSE
14 $A_{j+1} = \hat{A}_j$
15 END IF
16 END FOR

Algorithm 5.1 derived in the previous chapter is rewritten as shown in Algorithm 6.1. The modifications proposed for Algorithm 6.1 for the iterations $j = 0, j = 1$ to $m-2$ and $j = m - 1$ are described as follows.

For $j = 0$:

Consider the Steps 3-7 of Algorithm 6.1 given as

$$\begin{aligned} &IF\ b_0 = 1 \\ &\quad P_1 = P_0 \oplus A_0 \\ &ELSE \\ &\quad P_1 = P_0 \\ &ENDIF \end{aligned} \tag{6.1}$$

P is initialized to zero at the beginning of the algorithm i.e. $P_0 = (0, 0, 0, \cdots, 0)$. The IF-ELSE condition in Eqn. (6.1) can be realized using MUX operation having $P_0 = (0, 0, 0, \cdots, 0)$ and $A_0 = (a_{0,0}, a_{1,0}, a_{2,0}, \cdots, a_{m-1,0})$ as inputs with b_0 as the selection input given as

$$\begin{aligned} P_1 &= \left(\overline{b}_0 \,\&\, P_0\right) OR \left(b_0 \,\&\, (P_0 \oplus A_0)\right) \\ &= \left(\overline{b}_0 \,\&\, (0, 0, 0, \cdots, 0)\right) OR \left(b_0 \,\&\, ((0, 0, 0, \cdots, 0) \oplus A_0)\right) \\ &= (0, 0, 0, \cdots, 0)\, OR \left(b_0 \,\&\, A_0\right) \\ &= b_0 \,\&\, A_0 \end{aligned} \tag{6.2}$$

In Eqn. (6.2), the logical AND operation of b_0 with A_0 is performed in a bitwise manner i.e. logical AND operation of b_0 is performed with each $a_{i,0}$, for $i = 0, 1, 2, \cdots, m-1$.
In Step 8 of Algorithm 6.1, A_0 is shifted left by one bit to obtain $\hat{A}_0 = (\hat{a}_{0,0}, \hat{a}_{1,0}, \hat{a}_{2,0}, \cdots, \hat{a}_{k-1,0}, \hat{a}_{k,0}, \hat{a}_{k+1,0}, \cdots, \hat{a}_{m-2,0}, \hat{a}_{m-1,0}) = (0, a_{0,0}, a_{1,0}, \cdots, a_{k-2,0}, a_{k-1,0}, a_{k,0}, \cdots, a_{m-3,0}, a_{m-2,0})$. Here, it may be noted that k can be the middle term of the trinomial $f_1(x)$ or it can be k_1, k_2, k_3 of the pentanomial $f_2(x)$.
Consider the Steps 9-13 of Algorithm 6.1 given as

$$\begin{aligned} &IF\ a_{m-1,0} = 1 \\ &\quad A_1 = \hat{A}_0 \oplus T \\ &ELSE \\ &\quad A_1 = \hat{A}_0 \\ &ENDIF \end{aligned} \tag{6.3}$$

In Eqn. (6.3), T can be an irreducible trinomial or pentanomial which is represented as $T = (t_0, 0, 0, \cdots, t_k, 0, 0, \cdots, t_m)$, where $t_0 = t_k = t_m = 1$. The IF-ELSE condition in

Eqn. (6.3) is evaluated for three cases as illustrated below.

Case – I: For $i = 0$ (i.e. computation of the x^0 coefficient), the IF-ELSE condition in Eqn. (6.3) can be computed using MUX operation having $\hat{a}_{0,0} = 0$ and $t_0 = 1$ as inputs with $a_{m-1,0}$ as the selection input given as

$$\begin{aligned} a_{0,1} &= (\overline{a}_{m-1,0} \,\&\, \hat{a}_{0,0})\, OR\, (a_{m-1,0} \,\&\, (\hat{a}_{0,0} \oplus t_0)) \\ &= (\overline{a}_{m-1,0} \,\&\, 0)\, OR\, (a_{m-1,0} \,\&\, (0 \oplus 1)) \\ &= (0)\, OR\, (a_{m-1,0} \,\&\, 1) \\ &= a_{m-1,0} \end{aligned} \tag{6.4}$$

Case – II: For $i = k$, the IF-ELSE condition in Eqn. (6.3) can be computed using MUX operation having $\hat{a}_{k,0} = a_{k-1,0}$ and $t_k = 1$ as inputs with $a_{m-1,0}$ as the selection input given as

$$\begin{aligned} a_{0,1} &= (\overline{a}_{m-1,0} \,\&\, \hat{a}_{k,0})\, OR\, (a_{m-1,0} \,\&\, (\hat{a}_{k,0} \oplus t_k)) \\ &= (\overline{a}_{m-1,0} \,\&\, a_{k-1,0})\, OR\, (a_{m-1,0} \,\&\, (a_{k-1,0} \oplus 1)) \\ &= (\overline{a}_{m-1,0} \,\&\, a_{k-1,0})\, OR\, (a_{m-1,0} \,\&\, \overline{a}_{k-1,0}) \\ &= a_{m-1,0} \oplus a_{k-1,0} \end{aligned} \tag{6.5}$$

Case – III: For all i, where $i \neq 0$, $i \neq k$ and $i \neq m$ (i.e. computation of all coefficients except the x^0 coefficient, x^k coefficient and x^m coefficient), the IF-ELSE condition in Eqn. (6.3) can be computed using MUX operation having $\hat{a}_{i,0} = a_{i-1,0}$ and $t_i = 0$ as inputs with $a_{m-1,0}$ as the selection input given as

$$\begin{aligned} a_{0,1} &= (\overline{a}_{m-1,0} \,\&\, \hat{a}_{i,0})\, OR\, (a_{m-1,0} \,\&\, (\hat{a}_{i,0} \oplus t_i)) \\ &= (\overline{a}_{m-1,0} \,\&\, a_{i-1,0})\, OR\, (a_{m-1,0} \,\&\, (a_{i-1,0} \oplus 0)) \\ &= (\overline{a}_{m-1,0} \,\&\, a_{i-1,0})\, OR\, (a_{m-1,0} \,\&\, a_{i-1,0}) \\ &= a_{i-1,0} \end{aligned} \tag{6.6}$$

The computations for $i = m$ coefficient of T are ommitted since there are no computations for this coefficient in Algorithm 6.1.

For j = 1, 2, 3, $\cdots$, m-2:

Consider the Steps 3-7 of Algorithm 6.1 given as

$$\begin{aligned} &IF\ b_j = 1 \\ &\quad P_{j+1} = P_j \oplus A_j \\ &ELSE \\ &\quad P_{j+1} = P_j \\ &ENDIF \end{aligned} \tag{6.7}$$

The IF-ELSE condition in Eqn. (6.7) can be realized using the MUX operation having $P_j = (p_{0,j}, p_{1,j}, p_{2,j}, \cdots, p_{m-1,j})$ and $A_j = (a_{0,j}, a_{1,j}, a_{2,j}, \cdots, a_{m-1,j})$ as inputs with b_j as the selection input given as

$$P_{j+1} = \left(\overline{b}_j \,\&\, P_j\right) OR \left(b_j \,\&\, (P_j \oplus A_j)\right) \tag{6.8}$$

In Eqn. (6.8), the logical AND operation of $\overline{b}_j$ with P_j and b_j with $(P_j \oplus A_j)$ is performed in a bitwise manner i.e. logical AND operation of $\overline{b}_j$ is performed with each $p_{i,j}$, for all $i = 0, 1, 2, \cdots, m-1$ and logical AND operation of b_j is performed with each $(p_{i,j} \oplus a_{i,j})$, for all $i = 0, 1, 2, \cdots, m-1$.

In Step 8, A_0 is shifted left by one bit to obtain $\hat{A}_j = (\hat{a}_{0,j}, \hat{a}_{1,j}, \hat{a}_{2,j}, \cdots, \hat{a}_{k-1,0}, \hat{a}_{k,0}, \hat{a}_{k+1,0}, \cdots, \hat{a}_{m-2,0}, \hat{a}_{m-1,0}) = (0, a_{0,0}, a_{1,0}, \cdots, a_{k-2,0}, a_{k-1,0}, a_{k,0}, \cdots, a_{m-3,0}, a_{m-2,0})$.

Consider the Steps 9-13 of Algorithm 6.1 given as

$$\begin{aligned} &IF\ a_{m-1,j} = 1 \\ &\quad A_{j+1} = \hat{A}_j \oplus T \\ &ELSE \\ &\quad A_{j+1} = \hat{A}_j \\ &ENDIF \end{aligned} \tag{6.9}$$

The IF-ELSE condition in Eqn. (6.9) is evaluated for three cases, i.e. $i = 0$, $i = k$ and for remaining i values, in a similar manner as derived for the $j = 0$ case.

Therefore, for $i = 0$ (i.e. computation of the x^0 coefficient), the resultant expression can be written as

$$a_{0,j+1} = a_{m-1,j} \tag{6.10}$$

For $i = k$, the resultant expression can be written as

$$a_{k,j+1} = a_{m-1,j} \oplus a_{k-1,j} \tag{6.11}$$

For all i, where $i \neq 0$, $i \neq k$ and $i \neq m$ (i.e. computation of all coefficients except the x^0 coefficient, x^k coefficient and x^m coefficient), the resultant expression can be written as

$$a_{i,j+1} = a_{i-1,j} \tag{6.12}$$

For $j = m$-1:

Consider the Steps 3-7 of Algorithm 6.1 given as

$$\begin{aligned} &IF\ b_{m-1} = 1 \\ &\quad P_m = P_{m-1} \oplus A_{m-1} \\ &ELSE \\ &\quad P_m = P_{m-1} \\ &ENDIF \end{aligned} \tag{6.13}$$

where, $P_m = (p_{0,m}, p_{1,m}, p_{2,m}, \cdots, p_{m-1,m})$ is the final result of the finite field multiplication operation. The IF-ELSE condition of Eqn. (6.13) can be realized using the MUX operation having $P_{m-1} = (p_{0,m-1}, p_{1,m-1}, p_{2,m-1}, \cdots, p_{m-1,m-1})$ and $A_{m-1} = (a_{0,m-1}, a_{1,m-1}, a_{2,m-1}, \cdots, a_{m-1,m-1})$ as inputs with b_{m-1} as the selection input given as

$$P_m = \left(\overline{b}_{m-1}\,\&\,P_{m-1}\right) OR (b_{m-1}\,\&\,(P_{m-1} \oplus A_{m-1})) \tag{6.14}$$

In Eq. (6.14), the logical AND operation of $\overline{b}_{m-1}$ with P_{m-1} and the logical AND operation of b_{m-1} with $(P_{m-1} \oplus A_{m-1})$ is performed in a bitwise manner i.e. logical AND operation of $\overline{b}_{m-1}$ is performed with each $p_{i,m-1}$, for all $i = 0, 1, 2, \cdots, m-1$ and logical AND operation of b_{m-1} is performed with each $(p_{i,m-1} \oplus a_{i,m-1})$, for all $i = 0, 1, 2, \cdots, m-1$.

The modified interleaved multiplication algorithm over GF(2^m) for trinomials and pentanomials is proposed based on the above derivations as shown in Algorithm 6.2. It can be observed from Eqn. (6.2), Eqn. (6.4), Eqn. (6.5), Eqn. (6.6), Eqn. (6.10), Eqn. (6.11) and Eqn. (6.12) that they require either a single logical AND operation or a single XOR operation or a simple rewiring as opposed to requiring one logical XOR operation, two logical AND operations and one logical OR operation as evident in Algorithm 6.1. This indicates reduction in computational complexity achieved by the proposed algorithm.

Algorithm 6.2: Proposed interleaved multiplication algorithm over $GF(2^m)$ for irreducible trinomials and pentanomials

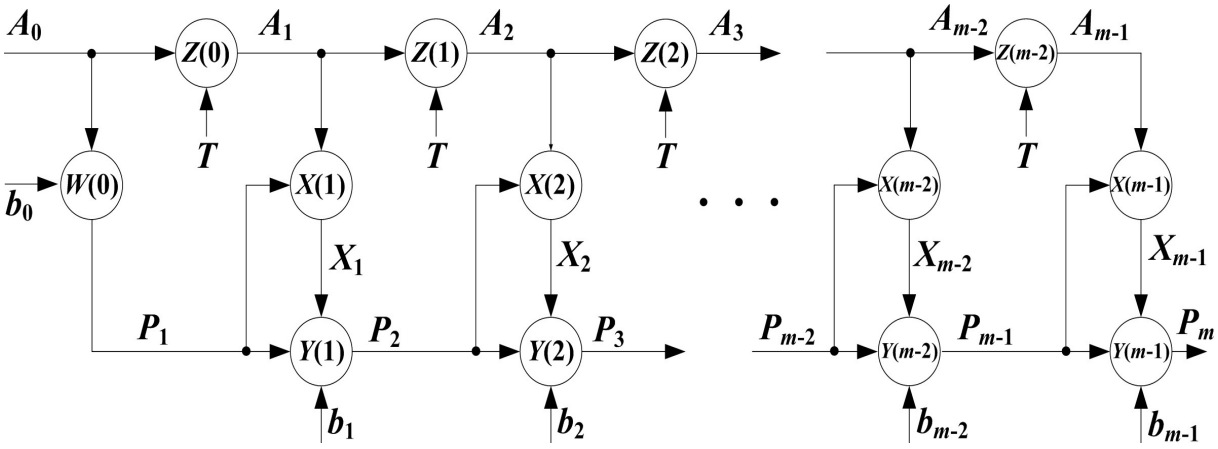
Inputs: A_j, B_j, T
Result: P_m
Initialization: $P_0 = \{0, 0, 0, \cdots, 0\}$
FOR $j = 0$ TO $m - 1$ DO
    IF($j == 0$)
        $p_{i,1} = a_{i,0} \& b_0$
        $\hat{A}_0 = A_0 \ll 1$
        $a_{0,1} = a_{m-1,0}$
        $a_{k,1} = a_{m-1,0} \oplus \hat{a}_{k-1,0}$
        $a_{i,1} = \hat{a}_{i,0}$
    ELSE IF($j == m - 1$)
        $p_{i,m-1} = \left(\overline{b}_{m-1} \,\&\, p_{i,m-1}\right) \; OR \; (b_{m-1} \& (p_{i,m-1} \oplus a_{i,m-1}))$
    ELSE
        $p_{i,j+1} = \left(\overline{b}_j \,\&\, p_{i,j}\right) OR \, (b_j \& \, (p_{i,j} \oplus a_{i,j}))$
        $\hat{A}_j = A_j \ll 1$
        $a_{0,j+1} = a_{m-1,j}$
        $a_{k,j+1} = a_{m-1,j} \oplus \hat{a}_{k-1,j}$
        $a_{i,j+1} = \hat{a}_{i,j}$
END FOR

6.3 Proposed Systolic Multiplier Architecture over GF(2^m) for Irreducible Trinomials

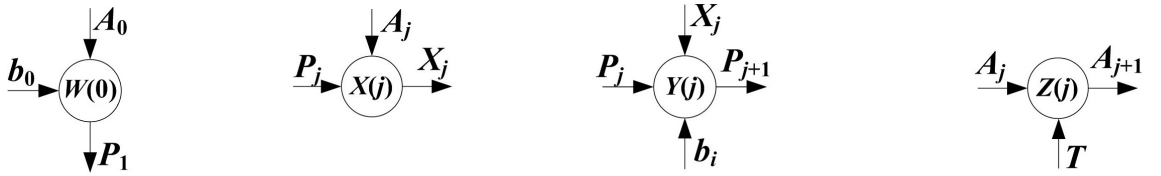
This sub-section presents the proposed architecture for systolic multiplier over $GF(2^m)$ for irreducible trinomials. The area complexity and delay of this architecture are estimated analytically and compared with the existing multipliers available in the literature. The functionality of the proposed architecture is implemented using ASIC and FPGA technologies. These analytical and implementation results of the proposed architecture are compared with the results of the existing multipliers that are presented in the following

sub-sections.

6.3.1 Design of Proposed Systolic Multiplier Architecture over GF(2^m) for Irreducible Trinomials



(a) The Signal Flow Graph (SFG)



(b) Logic function of $W(0)$ node

(c) Logic function of $X(j)$ node

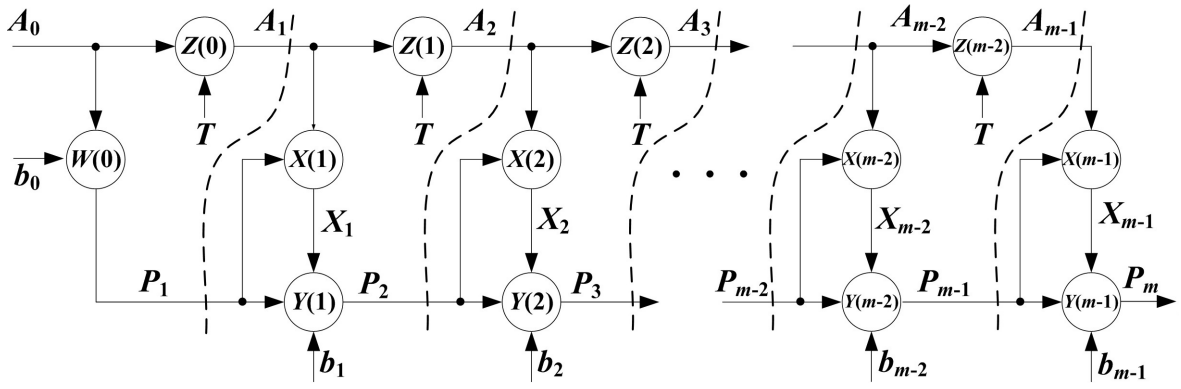
(d) Logic function of $Y(j)$ node

(e) Logic function of $Z(j)$ node

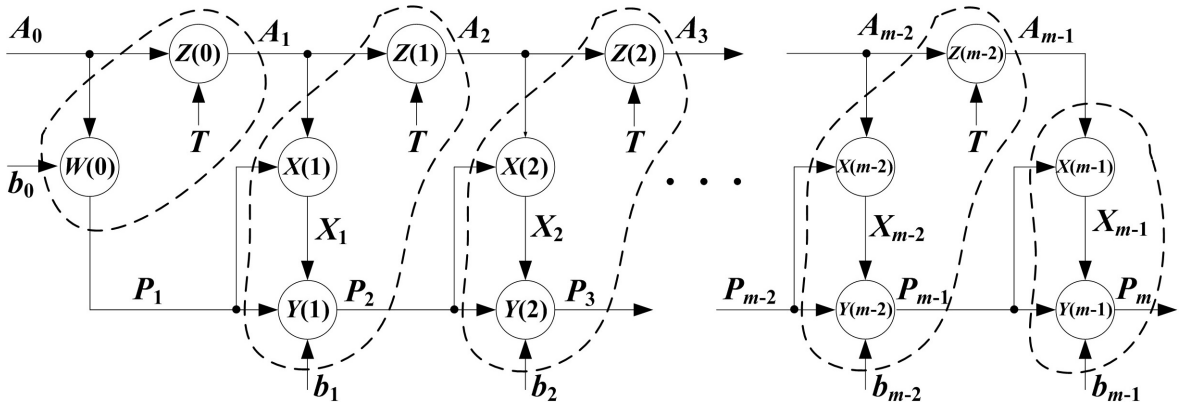
Figure 6.1: SFG derived from the proposed algorithm.

The computations in Algorithm 6.2 are represented by a Signal Flow Graph (SFG) shown in Fig. 6.1(a). The computations for $j = 0$ are represented using the nodes, $W(0)$ and $Z(0)$, the computations for $j = 1, 2, 3, \cdots, m-2$ are represented using the nodes, $X(j)$, $Y(j)$ and $Z(j)$ and the computations for $j = m-1$ is represented using the nodes, $X(m-1)$ and $Y(m-1)$. Here, $W(j)$ is the multiplication node wherein the multiplication operation is performed by a logical AND operation, $X(j)$ is the addition node wherein the addition operation is performed by a logical XOR operation, $Y(j)$ is the decision node wherein the decision operation is performed by a MUX operation and $Z(j)$ is the reduction node wherein the modular reduction operation is performed by a logical XOR and MUX operations. The logical functionality of these nodes are shown in Fig. 6.1(b)-

(e). Here, A_0 is the binary representation of $A(x)$, b_0 is the LSB of the $B(x)$, P_1 is the result of the multiplication node $W(0)$ which performs the multiplication of b_0 and A_0, P_j is the binary representation of $P(x)$ in the j^{th} iteration, A_j is the binary representation of $A(x)$ in the j^{th} iteration, X_j is the result of the addition node $X(j)$ for the j^{th} iteration which performs the addition of P_j and A_j , b_i is the i^{th} coefficient of $B(x)$, P_{j+1} is the result of the decision node for the j^{th} iteration which performs decision between the P_j and X_j using b_i as the selection input, T is the binary representation of the irreducible polynomial $T(x)$, A_{j+1} is the result of the modular reduction of A_j for the j^{th} iteration which performs the reduction of A_j by T, and C is the binary representation of the final product $C(x)$.



(a) Proposed cut-set retimed SFG

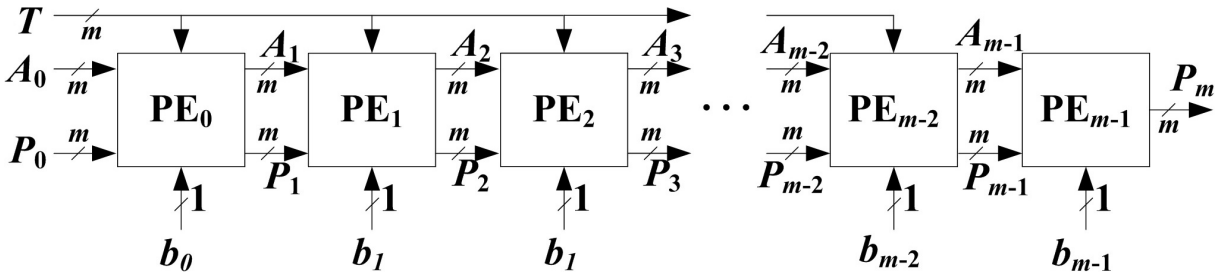


(b) SFG showing the formation of PEs

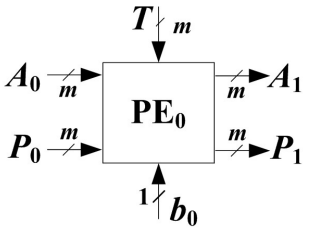
Figure 6.2: Cut-set retiming of the SFG.

The critical path can be reduced by employing appropriate cut-set retiming technique [59] on the SFG as shown in Fig. 6.2(a). Based on the cut-set retiming, the nodes of the SFG are grouped together to form processing elements as shown in Fig. 6.2(b). The nodes, $W(0)$ and $Z(0)$, are grouped together to form the processing element PE_0.

The nodes, $X(j), Y(j)$ and $Z(j)$, are grouped together to form a regular PE (PE_1 to PE_{m-2}), where $j = 1, 2, 3, \cdots, m-2$. The nodes, $X(m-1)$ and $Y(m-1)$, are grouped together to form PE_{m-1}.



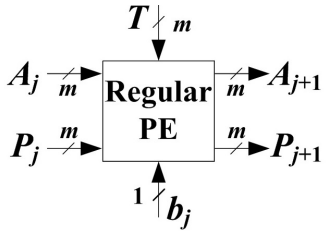
(a) Systolic structure



$$P_1 = P_0 \,.\, \bar{b}_0 + (A_0 \oplus P_0) \,.\, b_0$$

$$A_1 = A_0 \,.\, \bar{a}_{m-1,0} + (A_0 \oplus T) \,.\, a_{m-1,0}$$

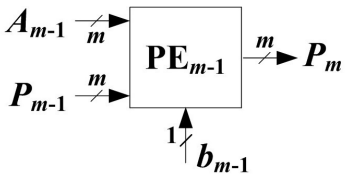
(b) Logic function of PE_0



$$P_{j+1} = P_j \,.\, \bar{b}_j + (A_j \oplus P_j) \,.\, b_j$$

$$A_{j+1} = A_j \,.\, \bar{a}_{m-1,j} + (A_j \oplus T) \,.\, a_{m-1,j}$$

(c) Logic function of Regular PE



$$P_m = P_{m-1} \,.\, \bar{b}_{m-1} + (A_{m-1} \oplus P_{m-1}) \,.\, b_{m-1}$$

(d) Logic function of PE_{m-1}

Figure 6.3: Proposed systolic multiplier using PEs.

The systolic multiplier architecture using processing elements derived from the cut-set retimed SFG is shown in Fig. 6.3(a). In $PE_0, W(0)$ is realized using m Y-cells and $Z(0)$ is realized using one V-cell, one W-cell and $(m-2)$ Z-cells. Here, each Y-cell uses one AND gate, each V-cell uses rewiring to forward the selection input of the decision node $a_{m-1,0}$ to the output, each W-cell uses one XOR gate and each Z-cell uses rewiring to forward an input of the decision node $a_{i-1,0}$ to the output. In regular PE, $X(j)$ and $Y(j)$ are realized using m U-cells and $Z(j)$ is realized using one V-cell, one W-cell and $(m-2)$ Z-cells. Here, each U-cell uses one XOR gate and one MUX, each V-cell uses rewiring to

forward the selection input of the decision node $a_{m-1,j}$ to the output, each W-cell uses one XOR gate and each Z-cell uses rewiring to forward an input of the decision node $a_{i-1,j}$ to the output. In PE_{m-1}, $X(m-1)$ and $Y(m-1)$ are together realized using m U-cells, where each U-cell uses one XOR gate and one MUX. The logical functionality of each PE is shown in Fig. 6.3(b)-(d).

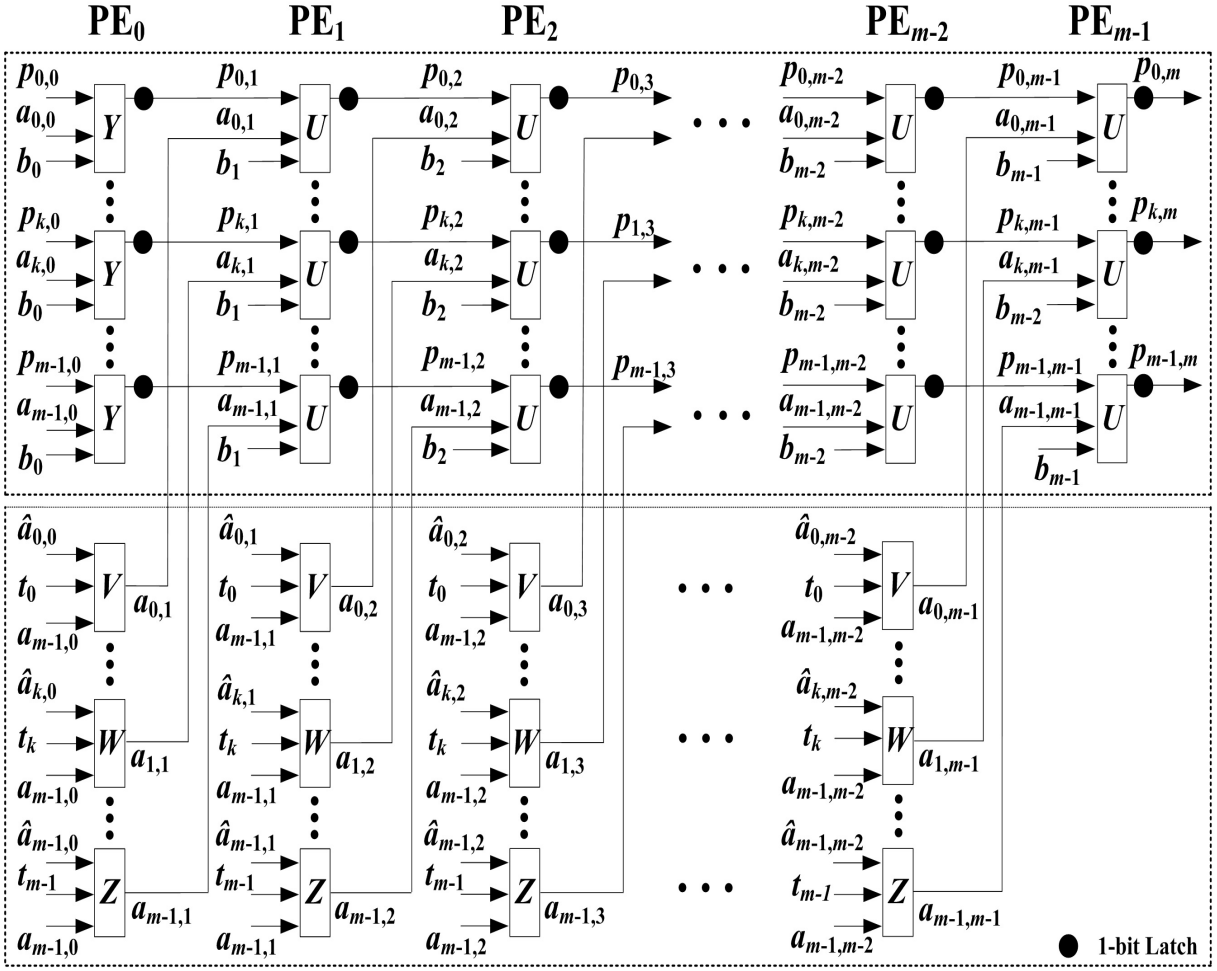
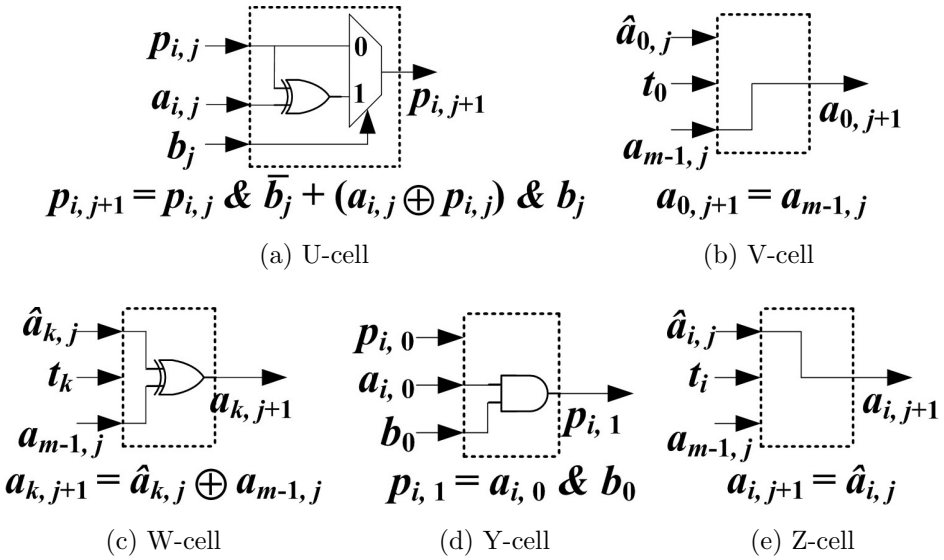


Figure 6.4: Proposed systolic multiplier architecture for trinomials using fundamental cells

This architecture is decomposed into a scalable, regular and simple structure realized using fundamental cells as shown in Fig. 6.4. The cells in each PE is represented column-wise. The upper block computes the polynomial multiplication and the lower block computes the modular reduction. The inputs to each cell are $p_{i,j}, a_{i,j}, b_i$ and $a_{i,j}, t_i, a_{m-2,j}$ for upper and lower blocks, respectively, where i denotes the index of the coefficient of the polynomial under consideration and j denotes the iteration count. The internal circuit detail and logical functionality of U-cell, V-cell, W-cell, Y-cell and Z-cell are shown in



(a) U-cell (b) V-cell

(c) W-cell (d) Y-cell (e) Z-cell

Figure 6.5: Internal circuit detail and logic functionality of fundamental cells.

Fig. 6.5(a)-(e). The optimizations achieved in the proposed algorithm for trinomials and pentanomials are transformed into logic gates i.e. Eqn. (6.2) is transformed into Y-cell, Eqn. (6.4) is transformed into V-cell, Eqn. (6.5) is transformed into W-cell, Eqn. (6.6) is transformed into Z-cell, Eqn. (6.8) is transformed into U-cell. These optimized cells replace the U-cells in the lower modular reduction block of the systolic structure. These cells translate the algorithmic optimizations achieved in the previous sub-section into architectural hardware optimizations. As a result, from Fig. 6.4 and Fig. 6.5, it can be observed that the proposed architecture requires a total of (m^2-1) XOR gates, (m^2-m) MUXs, m AND gates and m^2 latches. The critical path of the proposed architecture is given by the expression $max\{T_X + T_M, T_X, T_A\}$ with latency of m clock cycles.

6.3.2 Analytical Results

The proposed architecture requires (m^2-1) XOR gates, (m^2-m) MUXs, m AND gates and m^2 latches. The critical path of the proposed architecture is $(T_X + T_M)$ and the latency is m clock cycles as illustrated in previous section.

Table 6.1 presents the area complexity, latency and critical path of the proposed architecture and the existing multiplier architectures [40–49] available in the literature. Here, $T_A, T_X, T_M, T_{3X}, T_{NA}$ denote the delays of 2-input AND gate, 2-input XOR gate, 2:1 MUX, 3-input XOR gate and 2-input NAND gate, respectively. The trinomial $f(x) =$

Table 6.1: Area complexity and delay comparison of the systolic multipliers for trinomials over $GF(2^m)$.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[40]	m^2	m^2+lm	0	$4m^2+2lm$	$m+l+1$	T_A+T_X
[41]	m^2	m^2+m-1	0	$3m^2+2m-2$	$2m-1$	T_A+T_{3X}
[42]	m^2+N_v	m^2+m	0	$4m^2$+m	m+1	T_A+T_X
[43]	m^2	m^2+m-1	0	$2m^2$	$2m$-1	T_A+T_X
[44]	m^2	m^2+m	m	$3m^2$+m	m+n	T_A+T_X
[45]	m^2	m^2+m-1	0	$2m^2$	$2m$-1	T_A+T_X
[46]	$(m^2)^*$	m^2-1	$(m^2$-$2m)^\dagger$	$2m^2$-m	m	$T_{NA}+T_X$
[47]	m^2	m^2+m	m^2	$3.5m^2$+$3m$	m+2	T_M+T_X
[48]	m^2	m^2+m	m	$2m^2$+$3m$	m+1	$T_A+T_X+T_M$
[49]	$(m^2)^*$	$1.5m^2$+$0.5m$	$(1.5m^2$-$2.5m$+$3)^\dagger$	$1.5m^2$+$2m$-1	m+2	$T_{NA}+T_X$
Proposed	m	m^2-1	m^2-m	m^2	m	T_M+T_X

*NAND gates; †Inverter; $l=\left\lfloor\frac{(m-2)}{(m-k)}\right\rfloor+1$; $N_V=\frac{(m-k)(m-k-1)+k(k+1)}{2}$; n = n-term Hankel matrix representation.

Table 6.2: Comparison of area complexity, latency, critical path delay, total delay, ADP and % reduction in area of the proposed systolic multiplier for trinomials with existing systolic multipliers over $GF(2^{233})$.

Multipliers	#Transistors	Latency	CP (ns)	Delay (ns)	ADP ($\times10^{10}$)	%Reduction in hardware	%Reduction in ADP
[40]	2,398,968	236	19	4,484	1.0757	54	45
[41]	1,959,508	465	19	8,835	1.7312	44	66
[42]	1,521,484	465	19	8,835	1.3442	28	56
[43]	2,483,994	234	19	4,446	1.1044	56	47
[44]	1,959,064	236	19	4,484	0.8784	44	33
[45]	1,521,484	465	19	8,835	1.3442	28	56
[46]	1,517,290	233	20	4,660	0.7071	28	17
[47]	2,179,952	235	23	5,405	1.1783	50	50
[48]	1,528,480	234	30	7,020	1.073	28	45
[49]	1,523,356	235	20	4,700	0.716	28	18
Proposed	1,085,774	233	23	5,359	0.5819	-	-

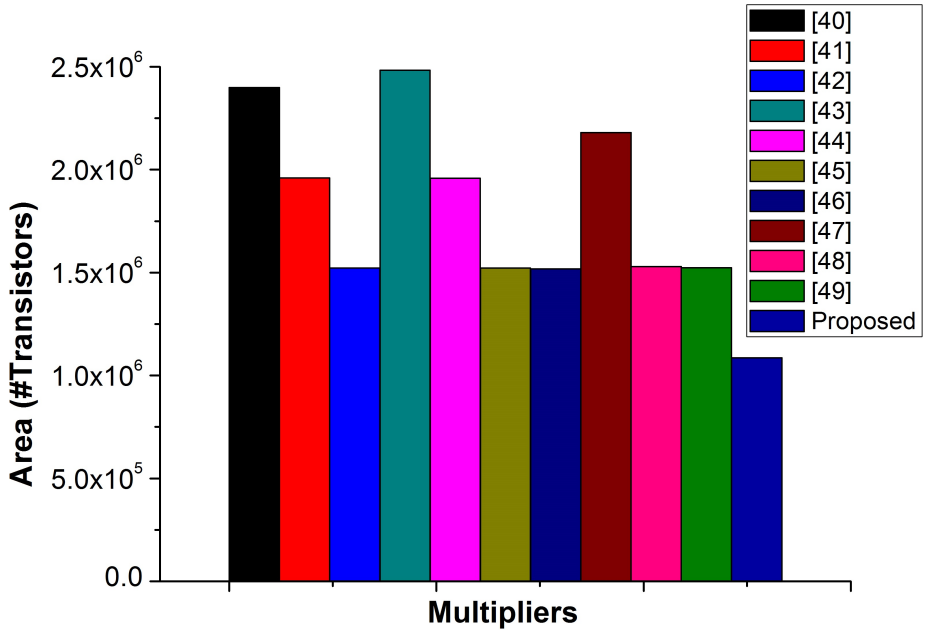
$x^{233}+x^{74}+1$ recommended by NIST is considered as an example to compare the area complexity and delay of the proposed architecture with the existing multipliers. The area

complexity is estimated using traditional CMOS logic transistor counts [55]: six transistors for 2-input XOR gate, 2-input AND gate, 1-bit 2:1 MUX, four transistors for a 2-input NAND gate, two transistors for an inverter, eight transistors for a 1-bit register. Some real-time circuits from STMicroelectronics [56] are considered to estimate the critical path delay and their propagation delays are: t_{PD} = 12ns [2-input XOR gate (M74HC86)], t_{PD} = 7ns [2-input AND gate (M74HC08)], t_{PD} = 11ns [2:1 MUX (M74HC257)], t_{PD} = 8ns [2-input NAND gate (M74HC00)].

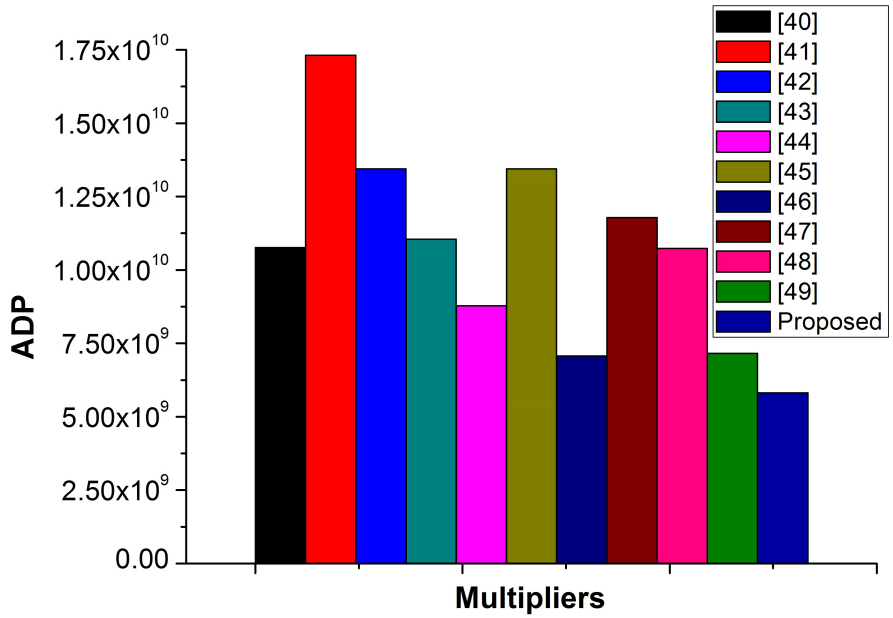
Based on these estimations, the area complexity (# Transistors), latency (#clock cycles), critical path delay (ns), total delay (ns), ADP, % reduction in area and % reduction in ADP are computed for $m = 233$ as shown in Table 6.2. The area complexity and ADP estimations are also plotted as shown in Fig. 6.6(a)-(b), respectively. It can be observed from Fig. 6.6(a) that the proposed architecture achieves low area complexity compared to the existing multipliers available in the literature. Specifically, it achieves up to 54%, 44%, 28%, 56%, 44%, 28%, 28%, 50%, 28% and 28% reduction in area complexity for $m = 233$ compared to the existing multipliers [40–49], respectively. Moreover, the proposed architecture also achieves low ADP as evident from Fig. 6.6(b). It achieves about 45%, 66%, 56%, 47%, 33%, 56%, 17%, 50%, 45% and 18% reduction in ADP compared to the existing multipliers available in the literature. The reduction in ADP metric shows that the proposed architecture achieves reduction in area complexity without much increase in delay compared to the existing multipliers and hence is suitable for low-hardware cryptographic applications.

6.3.3 Implementation Results

The performance of the proposed multiplier architecture for trinomials and the systolic multipliers available in the literature are verified by implementing them on ASIC and FPGA platforms. The implementation results of these architectures are presented in the following sub-sections.



(a) Area complexity



(b) Area-Delay Product

Figure 6.6: Comparison of systolic multipliers over $GF(2^{233})$ for irreducible trinomials.

6.3.3.1 ASIC Implementation Results

The proposed systolic multiplier and the systolic multipliers [46, 49] are considered for hardware implementations since they require low ADP compared to the existing systolic multipliers. These multipliers are modelled in Verilog for $m = 233$ and synthesized using Synopsys Design Vision Compiler and Synopsys $90nm$ Generic Library. The de-

Table 6.3: ASIC implementation results of systolic multipliers over $GF(2^{233})$ for irreducible trinomials.

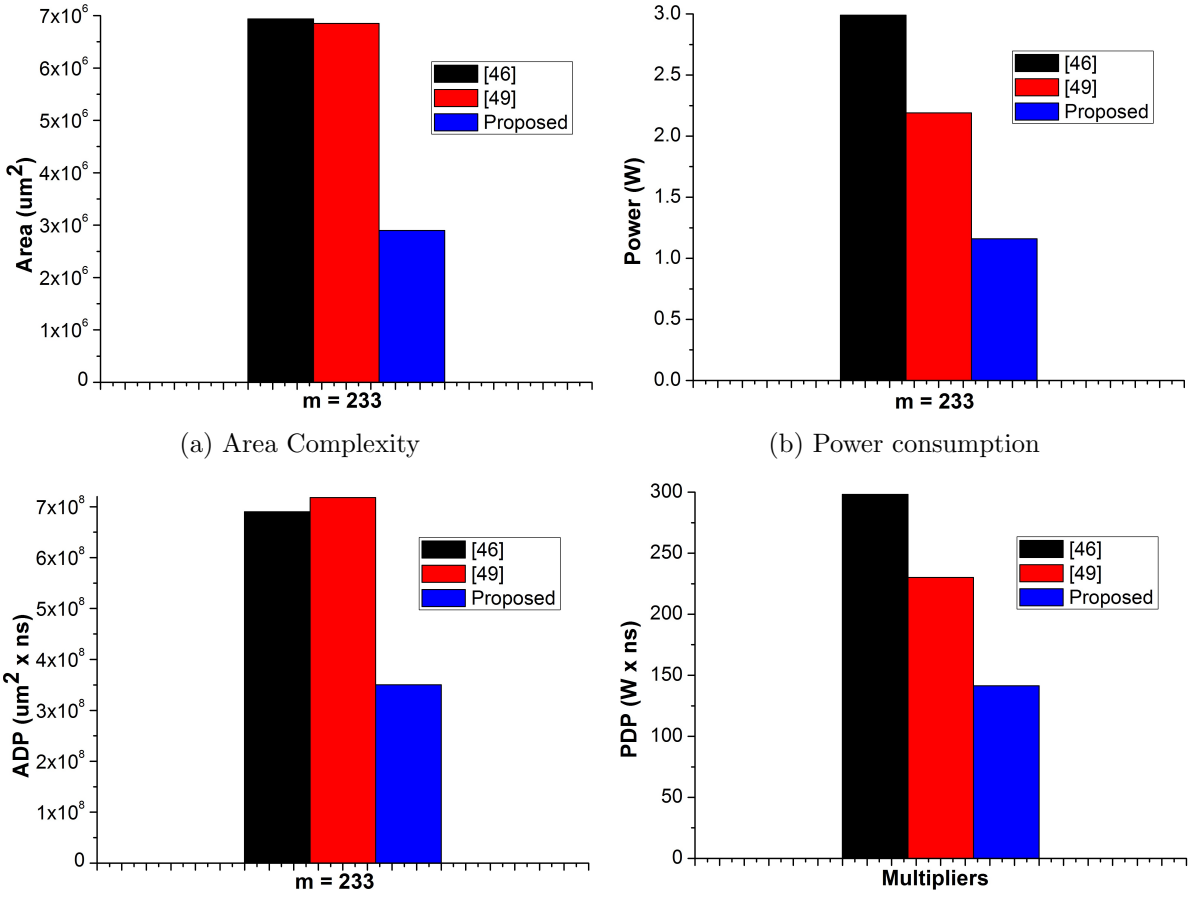
Metrics \ Multipliers	[46]	[49]	Proposed
Total Delay (ns)	99.491	104.81	120.927
Area $(\mu m^2)(\times 10^3)$	6939.995	6850.603	2899.179
Power (W)	2.99	2.19	1.16
ADP $(\mu m^2 \times ns)(\times 10^6)$	690.169	718.011	350.589
PDP $(W \times ns)$	298.214	230.121	141.412

lay, area complexity, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 6.3).

The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 6.7(a)-(d), respectively. It is clear from the histogram (see Fig. 6.7) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, it can be observed that the proposed multiplier achieves reduction of about 58%, 61%, 49% & 52% in area complexity, power consumption, ADP and PDP, respectively, compared to the systolic multiplier [46]. Similarly, the proposed multiplier achieves reduction of about 57%, 51%, 47% & 38% in area complexity, power consumption, ADP and PDP, respectively, when compared to the systolic multiplier [49]. Moreover, it can also be observed that the delay of the proposed architecture is slightly higher than existing designs but it has lower ADP and PDP which indicates that the proposed architecture achieve improvement in area complexity and power consumption without much increase in delay.

6.3.3.2 FPGA Implementation Results

In addition to the ASIC implementation, the functionality of the proposed multiplier is also verified by implementing the Verilog models on FPGA platform. The Verilog models of the proposed multiplier and the multipliers [46,49] are simulated and synthesized



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

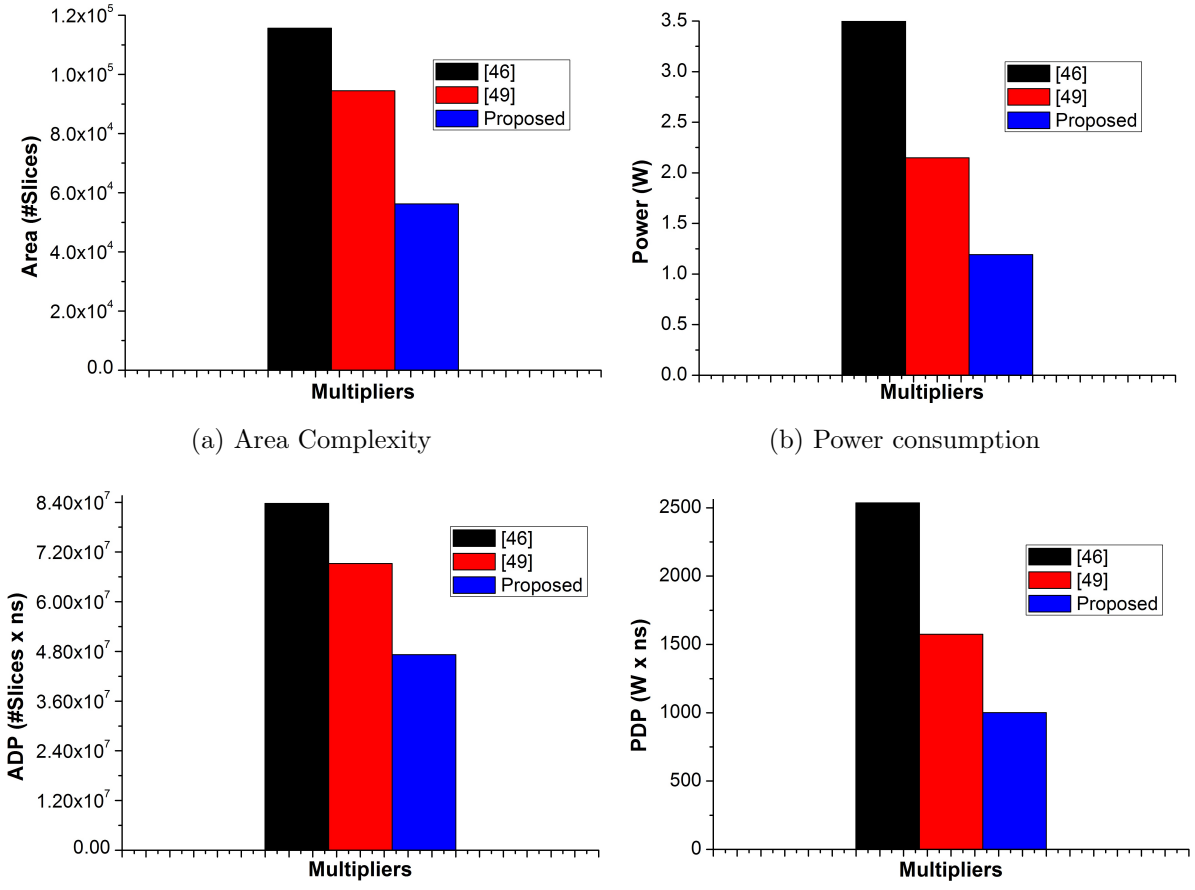
(d) Power-Delay Product

Figure 6.7: ASIC implementation results of systolic multipliers over $GF(2^{233})$ for irreducible trinomials.

Table 6.4: FPGA implementation results of systolic multipliers over $GF(2^{233})$ for irreducible trinomials.

Metrics \ Multipliers	[46]	[49]	Proposed
Total Delay (ns)	724.397	732.73	839.732
Area $(\#Slices)(\times 10^3)$	115.606	94.498	56.233
Power (W)	3.501	2.148	1.192
ADP $(\#Slices \times ns)(\times 10^6)$	83.745	69.242	47.221
PDP $(W \times ns)(\times 10^3)$	2.536	1.574	1.001

using Xilinx Vivado 2014.2 tool. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA prototype board. The delay, area, power



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 6.8: FPGA implementation results of systolic multipliers over $GF(2^{233})$ for irreducible trinomials.

consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 6.4).

The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 6.8(a)-(d), respectively. It is clear from the histogram (see Fig. 6.8) that the proposed multiplier achieves low area complexity, power consumption, ADP and PDP among existing multipliers. More specifically, it can be observed that the proposed multiplier achieves reduction of about 51%, 43%, 65% & 60% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [46]. Similarly, the proposed multiplier achieves reduction of about 40%, 31%, 44% & 36% in area complexity, power consumption, ADP and PDP, respectively, when compared to the multiplier [49]. Moreover, it can also be observed that the delay of the proposed architecture is slightly higher than existing designs but it has lower ADP and PDP which

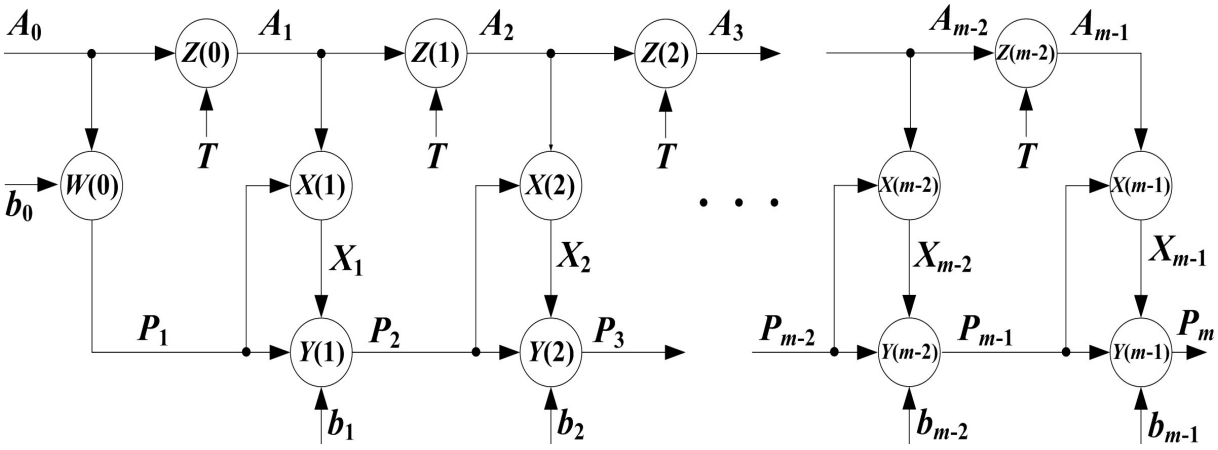
indicates that the proposed architecture achieve improvement in area complexity and power consumption without much increase in delay.

6.4 Proposed Systolic Multiplier Architecture over GF(2^m) for Irreducible Pentanomials

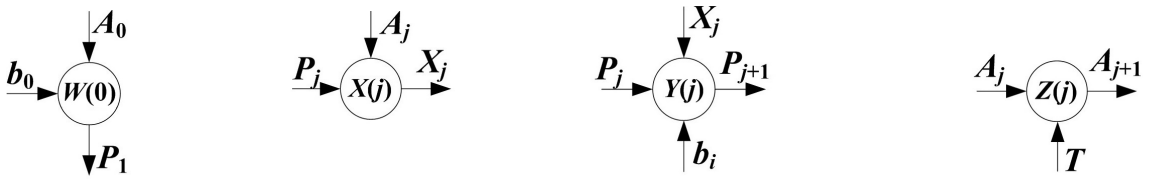
This sub-section presents the proposed arch for systolic multiplier over $GF(2^m)$ for irreducible pentanomials. The area complexity and delay of this architecture are estimated analytically and compared with the existing multipliers available in the literature. The functionality of the proposed architecture is implemented using ASIC and FPGA technologies. These analytical and implementation results of the proposed architecture and the multipliers available in the literature are also presented in the following sub-sections.

6.4.1 Design of Proposed Systolic Multiplier Architecture over GF(2^m) for Irreducible Pentanomials

The computations in Algorithm 6.2 are represented by a Signal Flow Graph (SFG) shown in Fig. 6.9(a). The computations for $j = 0$ are represented using the nodes, $W(0)$ and $Z(0)$, the computations for $j = 1, 2, 3, \cdots, m-2$ are represented using the nodes, $X(j)$, $Y(j)$ and $Z(j)$ and the computations for $j = m-1$ is represented using the nodes, $X(m-1)$ and $Y(m-1)$. Here, $W(j)$ is the multiplication node wherein the multiplication operation is performed by a logical AND operation, $X(j)$ is the addition node wherein the addition operation is performed by a logical XOR operation, $Y(j)$ is the decision node wherein the decision operation is performed by a MUX operation and $Z(j)$ is the reduction node wherein the modular reduction operation is performed by a logical XOR and MUX operations. The logical functionality of these nodes are shown in Fig. 6.9(b)-(e). Here, A_0 is the binary representation of $A(x)$, b_0 is the LSB of the $B(x)$, P_1 is the result of the multiplication node $W(0)$ which performs the multiplication of b_0 and A_0, P_j is the binary representation of $P(x)$ in the j^{th} iteration, A_j is the binary representation of $A(x)$ in the j^{th} iteration, X_j is the result of the addition node $X(j)$ for the j^{th} iteration which performs the addition of P_j and A_j , b_i is the i^{th} coefficient of



(a) The Signal Flow Graph (SFG)



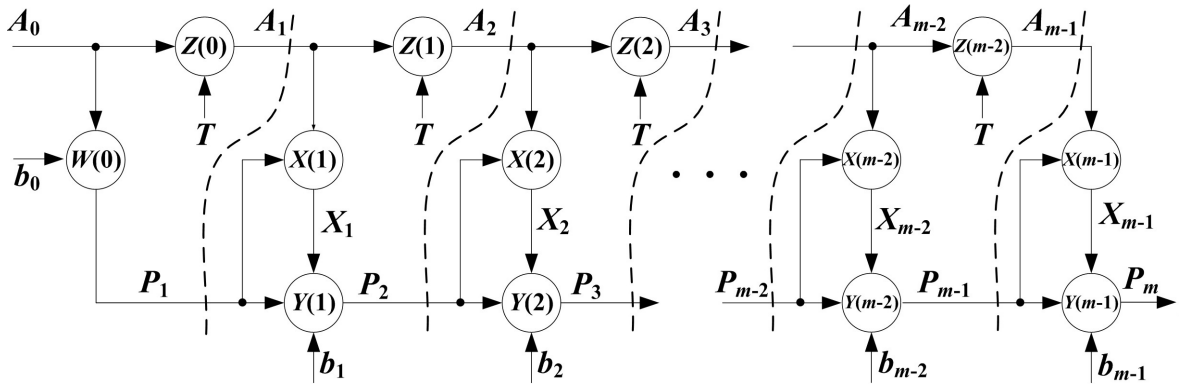
$P_1 = b_0 \,.\, A_0$ $X_j = P_j + A_j$ $P_{j+1} = \overline{b}_i \,.\, P_j + b_i \,.\, X_j$ $A_{j+1} = A_j \bmod T$

(b) Logic function of $W(0)$ node (c) Logic function of $X(i)$ node (d) Logic function of $Y(i)$ node (e) Logic function of $Z(i)$ node

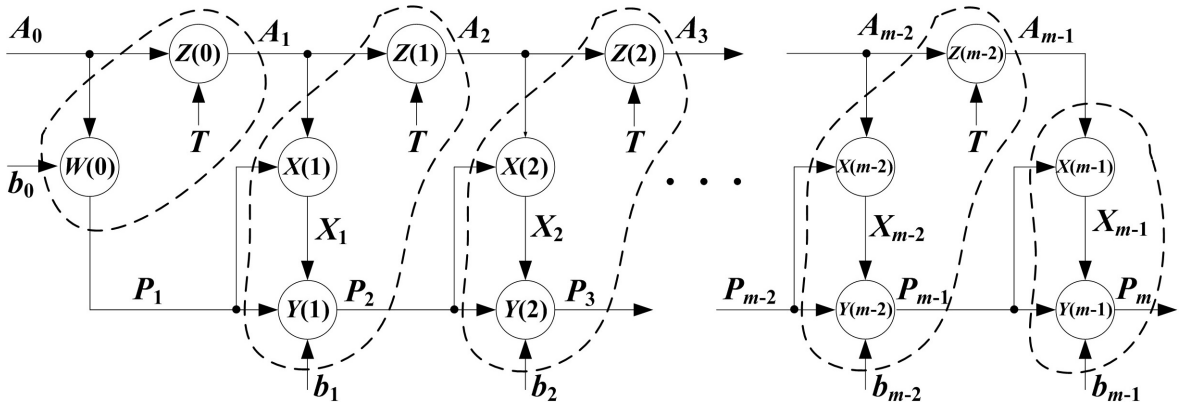
Figure 6.9: SFG derived from the proposed algorithm.

$B(x)$, P_{j+1} is the result of the decision node for the j^{th} iteration which performs decision between the P_j and X_j using b_i as the selection input, T is the binary representation of the irreducible polynomial $T(x)$, A_{j+1} is the result of the modular reduction of A_j for the j^{th} iteration which performs the reduction of A_j by T, and C is the binary representation of the final product $C(x)$. The critical path can be reduced by employing appropriate cut-set retiming technique [59] on the SFG as shown in Fig. 6.10(a). Based on the cut-set retiming, the nodes of the SFG are grouped together to form processing elements as shown in Fig. 6.10(b). The nodes, $W(0)$ and $Z(0)$, are grouped together to form the processing element PE_0. The nodes, $X(j), Y(j)$ and $Z(j)$, are grouped together to form a regular PE (PE_1 to PE_{m-2}), where $j = 1, 2, 3, \cdots, m-2$. The nodes, $X(m-1)$ and $Y(m-1)$, are grouped together to form PE_{m-1}.

The systolic multiplier architecture using processing elements derived from the cut-set retimed SFG is shown in Fig. 6.11(a). In $PE_0, W(0)$ is realized using m Y-cells and $Z(0)$ is realized using one V-cell, one W-cell and $(m-2)$ Z-cells. Here, each Y-cell uses one AND gate, each V-cell uses rewiring to forward the selection input of the decision



(a) Proposed cut-set retimed SFG

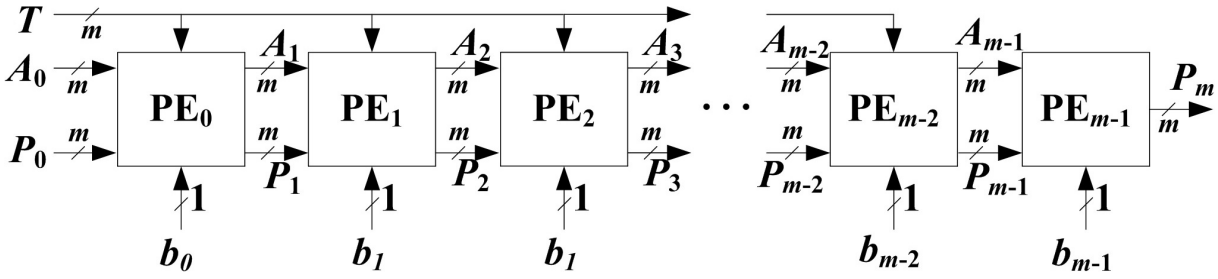


(b) SFG showing the formation of PEs

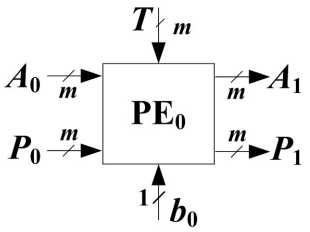
Figure 6.10: Cut-set retiming of the SFG.

node $a_{m-1,0}$ to the output, each W-cell uses one XOR gate and each Z-cell uses rewiring to forward an input of the decision node $a_{i-1,0}$ to the output. In regular PE, $X(j)$ and $Y(j)$ are realized using m U-cells and $Z(j)$ is realized using one V-cell, one W-cell and $(m-2)$ Z-cells. Here, each U-cell uses one XOR gate and one MUX, each V-cell uses rewiring to forward the selection input of the decision node $a_{m-1,j}$ to the output, each W-cell uses one XOR gate and each Z-cell uses rewiring to forward an input of the decision node $a_{i-1,j}$ to the output. In PE_{m-1}, $X(m-1)$ and $Y(m-1)$ are together realized using m U-cells, where each U-cell uses one XOR gate and one MUX. The logical functionality of each PE is shown in Fig. 6.11(b)-(d).

This architecture is decomposed into a scalable, regular and simple structure realized using fundamental cells as shown in Fig. 6.12. The cells in each PE is represented column-wise. The upper block computes the polynomial multiplication and the lower block computes the modular reduction. The inputs to each cell are $p_{i,j}, a_{i,j}, b_i$ and



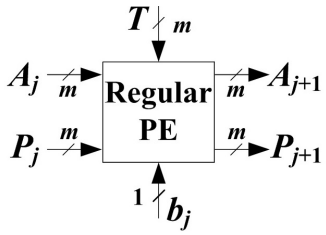
(a) Systolic structure



$$P_1 = P_0 \,.\, \overline{b}_0 + (A_0 \oplus P_0) \,.\, b_0$$

$$A_1 = A_0 \,.\, \overline{a}_{m\text{-}1,0} + (A_0 \oplus T) \,.\, a_{m\text{-}1,0}$$

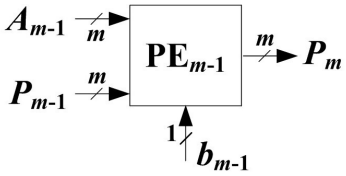
(b) Logic function of PE_0



$$P_{j+1} = P_j \,.\, \overline{b}_j + (A_j \oplus P_j) \,.\, b_j$$

$$A_{j+1} = A_j \,.\, \overline{a}_{m\text{-}1,j} + (A_j \oplus T) \,.\, a_{m\text{-}1,j}$$

(c) Logic function of Regular PE



$$P_m = P_{m\text{-}1} \,.\, \overline{b}_{m\text{-}1} + (A_{m\text{-}1} \oplus P_{m\text{-}1}) \,.\, b_{m\text{-}1}$$

(d) Logic function of PE_{m-1}

Figure 6.11: Proposed systolic structure using PEs.

$a_{i,j}, t_i, a_{m-2,j}$ for upper and lower blocks, respectively, where i denotes the index of the coefficient of the polynomial under consideration and j denotes the iteration count. The internal circuit detail and logical functionality of U-cell, V-cell, W-cell, Y-cell and Z-cell are shown in Fig. 6.13(a)-(e). These cells are realized in the same manner as in the case of trinomials described in the previous section. From Fig. 6.12 and Fig. 6.13, it can be observed that the proposed architecture requires a total of $(m^2 + 2m - 3)$ XOR gates, $(m^2 - m)$ MUXs, m AND gates and m^2 latches. The critical path of the proposed architecture is given by the expression $max\{T_X + T_M, T_X, T_A\}$ with latency of m clock cycles.

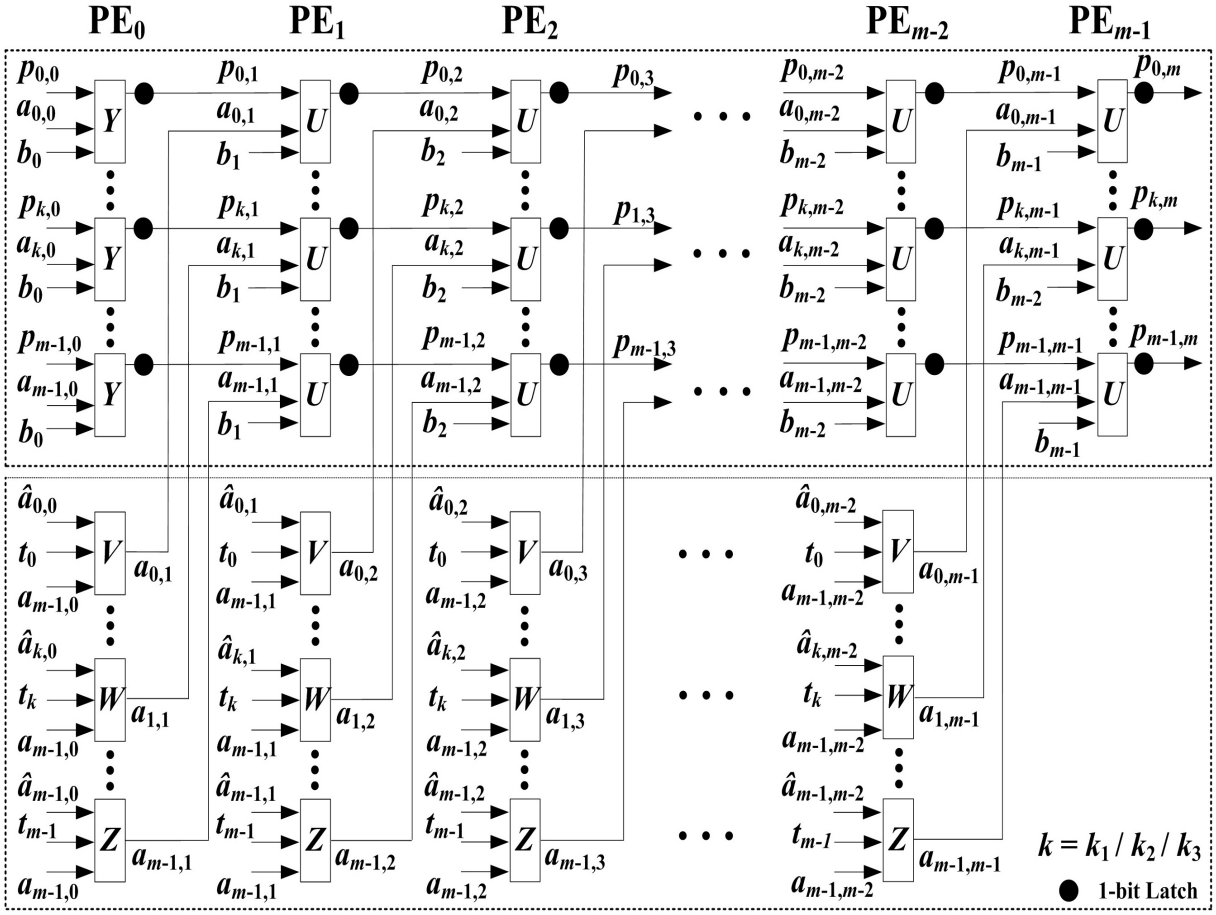
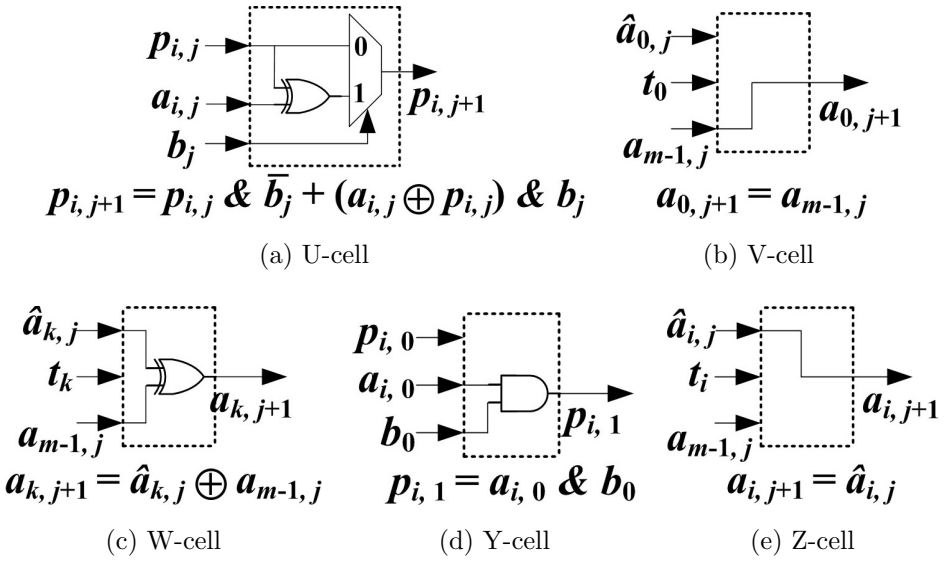


Figure 6.12: Proposed systolic multiplier architecture for pentanomials using fundamental cells.



(a) U-cell (b) V-cell (c) W-cell (d) Y-cell (e) Z-cell

Figure 6.13: Internal circuit detail and logic functionality of fundamental cells.

Table 6.5: Area complexity and delay comparison of the systolic multipliers for pentanomials over $GF(2^m)$.

Multipliers	#AND	#XOR	#MUX	#Registers	Latency	Critical Path Delay
[47]	m^2	m^2+3m+2	m	$3.5m^2+7m$	$m+4$	T_X
[50]	$(m^2)^a$	m^2+2m-1	$(m^2)^b$	$2m^2-2m$	m	$T_{NA}+T_X$
[51]	m^2	m^2+2m	$(m^2)^b$	$2m^2$	$\frac{m}{2}+2$	$2T_X$
[52]	$(m^2)^a$	$2m+2lm$ $+2l+2$	$(m^2)^c$	$3m^2-2m-2lm$ $-2l-2$	$\frac{m}{2l+2}+1$ $+log_2(2l+2)$	$T_{NA}+T_{XN}$

[a]NAND gates; [b]Inverter; [c]XNOR gate.

6.4.2 Analytical Results

The proposed architecture requires (m^2+2m-3) XOR gates, (m^2-m) MUXs, m AND gates and m^2 latches. The critical path of the proposed architecture is (T_X+T_M) and the latency is m clock cycles as illustrated in previous section.

Table 6.5 presents the area complexity, latency and critical path of the proposed architecture and the existing multiplier architectures [47,50–52] available in the literature. Here, $T_A, T_X, T_M, T_{3X}, T_{NA}$ denote the delays of 2-input AND gate, 2-input XOR gate, 2:1 MUX, 3-input XOR gate and 2-input NAND gate, respectively. The pentanomial $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$ recommended by NIST is considered as an example to compare the area complexity and delay of the proposed architecture with the existing multipliers. The area complexity is estimated using traditional CMOS logic transistor counts [55]: six transistors for 2-input XOR gate, 2-input AND gate, 1-bit 2:1 MUX, four transistors for a 2-input NAND gate, two transistors for an inverter, eight transistors for a 1-bit register. Some real-time circuits from STMicroelectronics [56] are considered to estimate the critical path delay and their propagation delays are: t_{PD} = 12ns [2-input XOR gate (M74HC86)], t_{PD} = 7ns [2-input AND gate (M74HC08)], t_{PD} = 11ns [2:1 MUX (M74HC257)], t_{PD} = 8ns [2-input NAND gate (M74HC00)], t_{PD} = 10ns [2-input XNOR gate (M74HC266)].

Based on these estimations, the area complexity and delay are computed for $m = 283$ as shown in Table 6.6. The area complexity and ADP estimations are also plotted as shown in Fig. 6.14(a) and (b), respectively. It can be observed from this figure that the proposed

Table 6.6: Comparison of area complexity, latency, critical path delay, total delay, ADP and % reduction in area of the proposed systolic multiplier for pentanomials with existing systolic multipliers over $GF(2^{283})$.

Multipliers	#Transistors	Latency	CP (ns)	Delay (ns)	ADP ($\times 10^{10}$)	%Reduction in hardware
[47]	3,226,212	287	23	6,601	21.296	50
[50]	2,241,354	283	18	5,094	11.417	28
[51]	2,406,066	144	24	3,456	8.315	33
[52]	2,879,796	51	18	918	2.644	44
Proposed	1,605,158	283	23	6,509	10.448	-

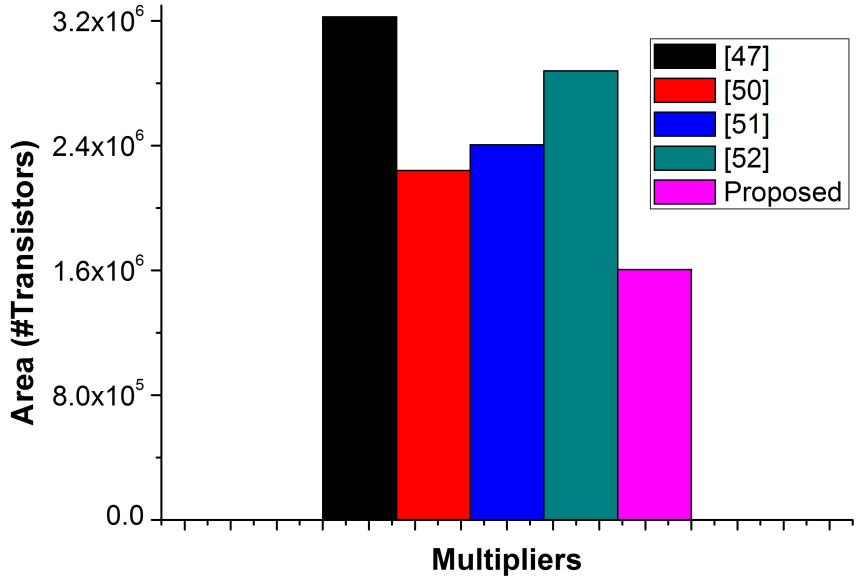
architecture achieves low area complexity compared to the existing multipliers available in the literature. Specifically, it achieves up to 50%, 28%, 33% and 44% reduction in hardware for $m = 283$ when compared to similar multipliers [47, 50–52], respectively, and hence suitable for low-hardware cryptographic applications. It can also be observed that the multipliers [51, 52] require low ADP and the multipliers [47, 50] require high ADP compared to the proposed architecture.

6.4.3 Implementation Results

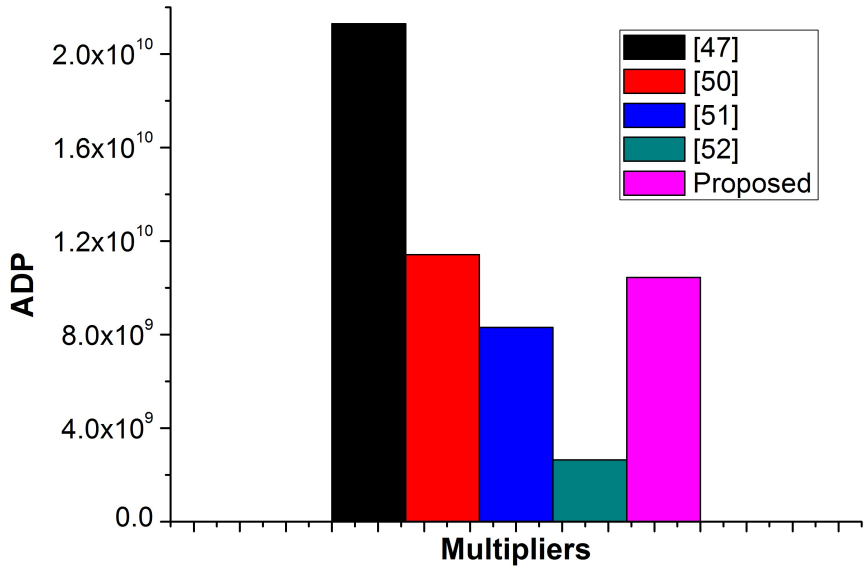
The performance of the proposed systolic multiplier architecture and the systolic multiplier architectures available in the literature are verified by implementing them on ASIC and FPGA platforms. The implementation results of these architectures are presented in the following sub-sections.

6.4.3.1 ASIC Implementation Results

The proposed systolic multiplier and the systolic multipliers [51, 52] are considered for hardware implementations since they require low ADP compared to the existing systolic multipliers. These multipliers are modelled in Verilog for $m = 283$ and synthesized using Synopsys Design Vision Compiler and Synopsys $90nm$ Generic Library. The de-



(a) Area Complexity



(b) Area-Delay Product

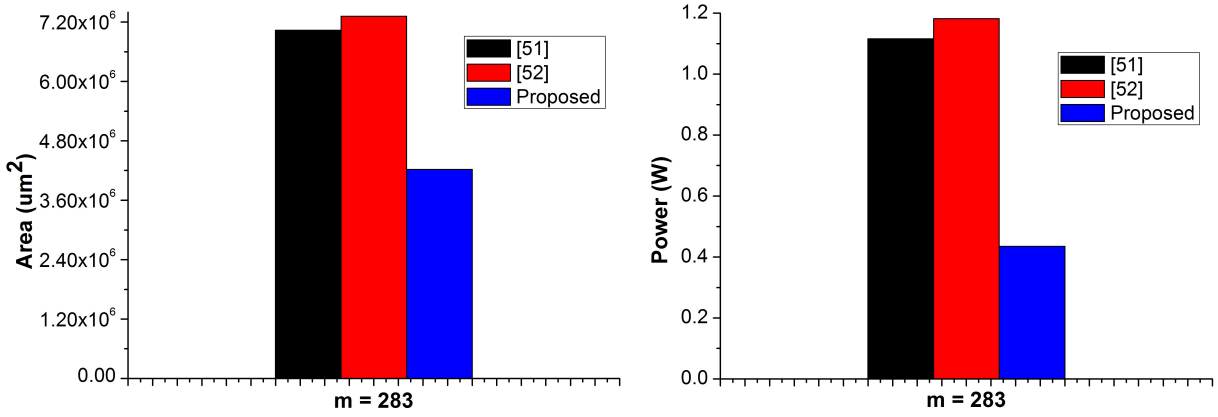
Figure 6.14: Comparison of systolic multipliers over $GF(2^{283})$ for irreducible pentanomials.

lay, area complexity, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 6.7).

The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 6.15(a)-(d), respectively. It is clear from the histogram (see Fig. 6.15) that the proposed multiplier requires low area complexity, power consumption, ADP and

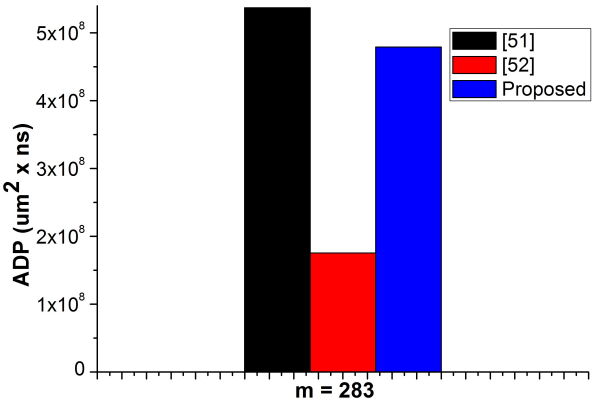
Table 6.7: ASIC implementation results of systolic multipliers over $GF(2^{283})$ for irreducible pentanomials.

Metrics \ Multipliers	[51]	[52]	Proposed
Total Delay (ns)	76.32	23.97	113.2
Area $(\mu m^2)(\times 10^6)$	7.035	7.319	4.233
Power (W)	1.116	1.182	0.435
ADP $(\mu m^2 \times ns)(\times 10^6)$	536.911	175.436	479.176
PDP $(W \times ns)$	85.173	28.333	49.242

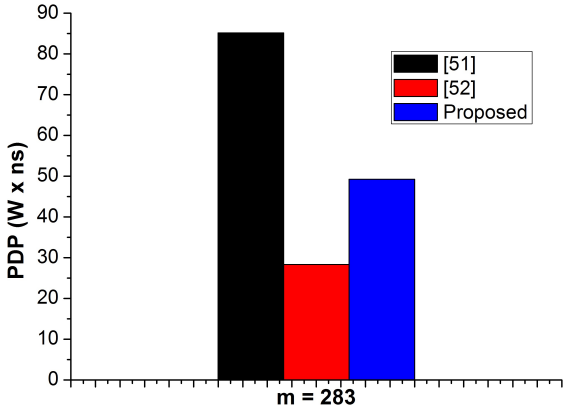


(a) Area Complexity

(b) Power consumption



(c) Area-Delay Product



(d) Power-Delay Product

Figure 6.15: ASIC implementation results of systolic multipliers over $GF(2^{283})$ for irreducible pentanomials.

PDP compared to the existing multipliers. More specifically, it can be observed that the proposed multiplier achieves reduction of about 39%, 61%, 10% & 42% in area complexity

and power consumption, respectively, compared to the systolic multiplier [51]. Similarly, the proposed multiplier achieves reduction of about 42% & 63% in area complexity and power consumption, respectively, when compared to the systolic multiplier [52]. It can be observed that the proposed multiplier requires more ADP and PDP compared to the systolic multiplier [52]. However, the proposed multiplier achieves substantial decrease in area complexity and power consumption than the systolic multiplier [52] and hence is suitable for applications with strict area and power constraints.

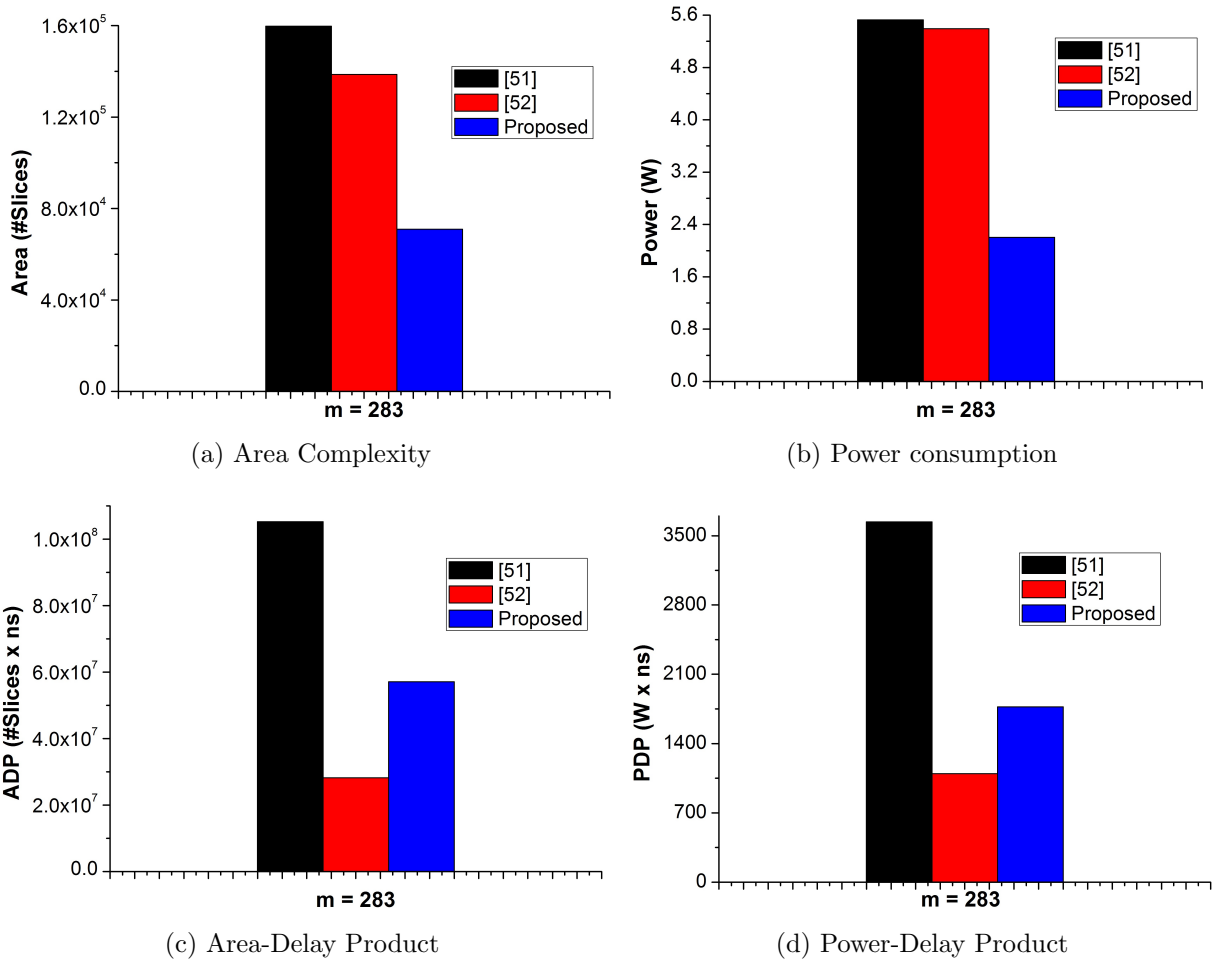
6.4.3.2 FPGA Implementation Results

Table 6.8: FPGA implementation results of systolic multipliers over $GF(2^{283})$ for irreducible pentanomials.

Metrics \ Multipliers	[51]	[52]	Proposed
Total Delay (ns)	658.8	203.388	803.72
Area ($\#Slices$)	159,754	138,676	70,988
Power (W)	5.527	5.392	2.204
ADP ($\#Slices \times ns$)($\times 10^6$)	105.246	28.205	57.054
PDP ($W \times ns$)($\times 10^3$)	3.641	1.097	1.771

The proposed systolic multiplier and the systolic multipliers [51, 52] are considered for hardware implementations since they require low ADP compared to the existing systolic multipliers. These multipliers are modelled in Verilog for $m = 283$ and synthesized using Synopsys Design Vision Compiler and Synopsys 90nm Generic Library. The delay, area complexity, power consumption, ADP and PDP of the all the architectures are computed from the device utilization summary generated by the synthesis tool (see Table 6.8).

The area complexity, power consumption, ADP and PDP results are also plotted as shown in Fig. 6.16(a)-(d), respectively. It is clear from the histogram (see Fig. 6.16) that the proposed multiplier requires low area complexity, power consumption, ADP and PDP compared to the existing multipliers. More specifically, it can be observed



(a) Area Complexity

(b) Power consumption

(c) Area-Delay Product

(d) Power-Delay Product

Figure 6.16: FPGA implementation results of systolic multipliers over $GF(2^{283})$ for irreducible pentanomials.

that the proposed multiplier achieves reduction of about 49%, 60%, 45% & 51% in area complexity, power consumption, ADP and PDP respectively, compared to the systolic multiplier [51]. Similarly, the proposed multiplier achieves reduction of about 55% & 59% in area complexity and power consumption, respectively, when compared to the systolic multiplier [52]. Similar to the ASIC implementation results, it can be observed that the proposed multiplier requires more ADP and PDP compared to the systolic multiplier [52] but the proposed multiplier is suitable for applications with strict area and power constraints due to low area and power requirements.

6.5 Conclusion

In this chapter, systolic multiplier over $GF(2^m)$ for irreducible trinomials and pentanomials are realized for the proposed algorithm. The area complexity and delay of the proposed architectures are estimated and performance is compared with other systolic polynomial basis multipliers available in the literature. It may be concluded from the comparisons of the estimated results that the proposed architecture for trinomials achieves low area complexity compared to other systolic multipliers for trinomials available in the literature. Moreover, the area-delay product of the proposed architecture is also low when compared to other multipliers, indicating an efficient multiplier design in terms of both area and delay. From the ASIC and FPGA synthesis results of the multipliers realized using trinomials, it can be concluded that the proposed architecture achieves low area complexity, power consumption, ADP and PDP compared to the existing multipliers. Similarly, it is observed from the comparisons of the estimated results that the proposed architecture for pentanomials achieves low area complexity compared to other systolic multipliers for pentanomials available in the literature. From the ASIC and FPGA synthesis results of the multipliers realized using pentanomials, it can be concluded that the proposed architecture achieves low area complexity and power consumption compared to the existing multipliers. The next chapter presents the conclusions of this thesis and some possible directions for future work.

Chapter 7

Conclusions and Future Scope

This chapter concludes the thesis by underlining the main contributions. It also discusses the possible directions of future work.

7.1 Conclusions

Cryptography is a prime necessity for secure communication of sensitive data over an insecure channel. Several algorithms are available in the literature to provide security for such insecure data communications. In our study of these cryptographic algorithms, it is observed that the multiplication in finite fields is the most extensively used and also the most compute-intensive operation. Several techniques to perform efficient finite field multiplications have been proposed in the literature to reduce the computational complexity of these finite field multiplications. Among these techniques, it is observed that the interleaved multiplication technique provides low computational complexity. In this work, modified interleaved multiplication algorithms are proposed to reduce the computational complexity.

The finite field multiplications can also be realized in hardware to achieve enhanced security and high speed compared to software implementations. Hence, several architectures are proposed in the literature to implement these finite field multiplications. The performance of these architectures can be improved with respect to area complexity, time delay and power consumption using optimized interleaved multiplication algorithms.

In this work, efficient sequential and systolic architectures are proposed for the implementation of finite field multiplications employing the proposed interleaved multiplication algorithms. The efficiency of these architectures are verified by employing them in realizing cryptographic algorithms such as the Advanced Encryption Standard (AES) and Twofish. The HDL models of these AES and Twofish algorithms are implemented using Xilinx Field Programmable Gate Array (FPGA) prototype board and also synthesized using Synopsys Design Vision compiler tool.

A modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm to perform finite field multiplications of any two arbitrary elements over $GF(2^m)$. This algorithm allows realization of a sequential multiplier architecture that achieves low area complexity compared to previous works. A sequential multiplier architecture over $GF(2^m)$ is developed based on the proposed interleaved multiplication algorithm. The performance of this architecture is verified by computing area complexity, delay and ADP for the field of order $m = 163$. It is observed that this architecture achieves a minimum reduction of about 28% in ADP compared to the existing works. The ASIC and FPGA implementations of the proposed architecture indicate a minimum reduction of about 49% in area, 16% in power consumption, 13% in ADP and 45% in PDP compared to the existing works. Since the AES and Twofish cryptographic algorithms use finite field multiplications of order $m = 8$, a sequential multiplier architecture over $GF(2^8)$ is developed for the proposed algorithm. The performance of this architecture is verified by computing area complexity, delay and ADP for the field of order $m = 8$. The proposed architecture achieves a minimum reduction of about 29% in ADP compared to the existing works. The FPGA implementation of two cryptographic algorithms, AES and Twofish, employing the proposed architecture achieves a minimum reduction of 22% in area, 42% in power consumption, 34% in ADP and 41% in PDP compared to the existing works.

A modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm to perform finite field multiplications of any two arbitrary elements over $GF(2^m)$. This algorithm is derived to realize a systolic multiplier architecture that achieves low area complexity compared to the existing works. A systolic multiplier architecture over $GF(2^m)$ is developed based on the proposed interleaved mul-

tiplication algorithm. The performance of this architecture is verified by computing area complexity, delay and ADP for the field of order $m = 163$. It is observed that this architecture achieves a minimum reduction of about 20% in area complexity compared to the existing works. The ASIC and FPGA implementations of the proposed multiplier indicate a minimum reduction of about 35% in area, 43% in power consumption, 73% in ADP and 76% in PDP compared to the existing works. Since the AES and Twofish cryptographic algorithms use finite field multiplications of order $m = 8$, a systolic multiplier architecture over $GF(2^8)$ is developed for the proposed algorithm. The performance of this architecture is verified by computing area complexity, delay and ADP for the field of order $m = 8$. The proposed architecture achieves a minimum reduction of about 21% in area complexity compared to the existing works. The FPGA implementation of the two cryptographic algorithms, AES and Twofish, employing the proposed architecture achieves a minimum reduction of 24% in area, 46% in power consumption, 33% in ADP and 41% in PDP compared to the existing works.

A modified interleaved multiplication algorithm is derived from a conventional interleaved multiplication algorithm based on a novel Pre-Computation technique. This algorithm performs finite field multiplications of any two arbitrary elements over $GF(2^m)$ and allows realization of two systolic multiplier architectures to achieve low area complexity compared to the existing works. A systolic multiplier architecture over $GF(2^m)$ for irreducible trinomials is developed based on the proposed interleaved multiplication algorithm. The performance of this architecture is verified by computing area complexity, delay and ADP for the field of order $m = 233$. The proposed architecture achieves a minimum reduction of about 28% in area complexity and about 17% in ADP compared to the existing works. The ASIC and FPGA implementations of this proposed architecture indicate a minimum reduction of 40% in area, 31% in power consumption, 44% in ADP and 36% in PDP compared to the existing works. Moreover, a systolic multiplier architecture over $GF(2^m)$ for irreducible pentanomials is developed based on the proposed interleaved multiplication algorithm. The performance of this architecture is verified by computing area complexity, delay and ADP for the field of order $m = 283$. This architecture achieves a minimum reduction of about 28% in area complexity compared to the existing works. The ASIC and FPGA implementations of the proposed architecture indicate a minimum reduction of 40% in area and 59% in power consumption compared to the existing works.

7.2 Future Scope

The work proposed in this thesis can be extended for future research. Some of the possible directions in which the problems can be further pursued are:

- The multipliers in this thesis are developed using bit-level approach. In future, a digit-serial approach can be used to design these architectures. The digit-serial approach can be easily explained as: when the digit size is 1, then the design becomes a bit-level design whereas the design changes into a bit-parallel design when digit size is equal to the field order m. Hence, this approach provides flexibility to the designer to chose an appropriate digit size for a particular application. The bit-level designs offer very low area complexity with low speeds while the bit-parallel designs offer high speeds with high area complexity. However, a digit-serial design can achieve optimal trade-off between area complexity and delay due to the provision of variable bit-width.

- The proposed systolic multipliers can be further optimized and tested for applications where irreducible polynomials are already known in advance using the MUX-based operations developed in this thesis for systolic multipliers. In addition, digit-serial approach can be used to design these multipliers to further achieve improvements in area complexity and power consumption.

- The architectures proposed in this thesis achieve reduction in power consumption using algorithmic and architectural optimizations. Hence, an important future extension would be to implement low-power techniques into the designs to achieve further reduction in power.

Publications

List of International Journals:

1. Sudha Ellison Mathe and Lakshmi Boppana, "Efficient Bit-parallel Systolic Polynomial Basis Multiplier over GF(2^8) based on Irreducible Polynomials," *Indian Journal of Science and Technology*, vol. 9, no. S1, December, 2016. **(Scopus-Indexed)** https://doi.org/10.17485/ijst/2016/v9iS1/107827

2. Sudha Ellison Mathe and Lakshmi Boppana, "Design and Implementation of a Sequential Polynomial Basis Multiplier over GF(2^m)," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 5, pp. 2680-2700, 2017. **(SCI-Indexed)** https://doi.org/10.3837/tiis.2017.05.021

3. Sudha Ellison Mathe and Lakshmi Boppana, "Low-Power and Low-Hardware Bit-parallel Polynomial Basis Systolic Multiplier over GF(2^m) for Irreducible Polynomials," *ETRI Journal*, vol. 39, no. 4, pp. 570-581, 2017. **(SCI-Indexed)** https://doi.org/10.4218/etrij.17.0116.0770

4. Sudha Ellison Mathe and Lakshmi Boppana, "Bit-Parallel Systolic Multiplier over GF(2^m) for Irreducible Trinomials with ASIC and FPGA Implementations," *IET Circuits, Devices and Systems*. (Published online) **(SCI-Indexed)** https://doi.org/10.1049/iet-cds.2017.0426

5. Sudha Ellison Mathe and Lakshmi Boppana, "Design and Implementation of a Novel Bit-Parallel Systolic Multiplier over GF(2^m) for Irreducible pentanomials," *Journal of Circuits, Systems and Computers*. (Published online) **(SCI-Indexed)** https://doi.org/10.1142/S0218126618502286

List of International Conferences:

1. Sudha Ellison Mathe and Lakshmi Boppana, "Efficient Sequential Polynomial Basis Multiplier over GF(2^8)," *Second International Conference on Computers and Management (ICCM'16)*, Kota, Rajasthan, 2016.

Bibliography

[1] R. L. Rivest, "Cryptography," in *Handbook of Theoretical Computer Science*, J. V. Leeuwen, Ed. Cambridge, USA: Elsevier, 1990, ch. 13, pp. 717–750.

[2] "Data encryption standard," Federal Information Processing Standard, Fairfax, USA, Tech. Rep., 10 1999.

[3] "Advanced encryption standard," Federal Information Processing Standard, Fairfax, USA, Tech. Rep., 10 1999.

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

[5] V. S. Miller, *Use of elliptic curves in cryptography.* London, UK: Springer-Verlag, 1986, vol. 218.

[6] 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.

[7] S. Roman, *Field theory*, 2nd ed. New York, USA: Springer-Verlag, 2006, vol. 158.

[8] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," *Doklady Akademii Nauk SSSR*, vol. 145, no. 2, p. 293294, 1962.

[9] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, p. 293294, 1985.

[10] E. D. Mastrovito, "VLSI designs for multiplication over finite fields $GF(2^m)$," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, T. Mora, Ed. Berlin, Heidelberg: Springer, 1988, vol. 357, pp. 201–213.

[11] D. G. Cantor, "On arithmetical algorithms over finite fields," *Journal of Combinatorial Theory*, vol. 50, no. 2, pp. 285–300, 1989.

[12] G. R. Blakely, "A computer algorithm for the product AB modulo M," *IEEE Transactions on Computers*, vol. 32, no. 5, pp. 497–500, 1983.

[13] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications.* New York, USA: Cambridge University Press, 1986.

[14] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson, "Optimal normal bases in GF(p^n)," *Discrete Applied Mathematics*, vol. 22, no. 2, pp. 149–161, 1989.

[15] M. Morii, M. Kasahara, and D. Whiting, "Efficient bit-serial multiplication and discrete time Wiener-Hoph equation over finite fields," *IEEE Transactions on Information Theory*, vol. 35, no. 6, pp. 1177–1183, 1989.

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

[17] H. Wu, M. A. Hasan, and I. F. Blake, "Low complexity parallel multiplier in F_{q^n} over F_q," *IEEE Transactions on Circuits and Systems I*, vol. 49, no. 7, pp. 1009–1013, 2002.

[18] M. A. Hasan and M. Ebtedaei, "Efficient architectures for computations over variable dimensional galois fields," *IEEE Transactions on Circuits and Systems I*, vol. 45, no. 11, pp. 1205–1210, 1998.

[19] P. Kitsos, G. Theodoridis, and O. Koufopavlou, "An efficient reconfigurable multiplier architecture for galois field GF(2^m)," *Microelectronics Journal*, vol. 34, no. 10, pp. 975–980, 2003.

[20] A. P. Fournaris and O. Koufopavlou, "Versatile multiplier architectures in GF(2^k) fields using the montgomery multiplication algorithm," *Integration, the VLSI journal*, vol. 41, no. 3, pp. 371–384, 2008.

[21] A. Zakerolhosseini and M. Nikooghadam, "Low-power and high-speed design of a versatile bit-serial multiplier in finite fields GF(2^m)," *Integration, the VLSI journal*, vol. 46, no. 2, pp. 211–217, 2013.

[22] H. Ho, "Design and implementation of a polynomial basis multiplier architecture over GF(2^m)," *Journal of Signal Processing Systems*, vol. 75, no. 3, pp. 203–208, 2014.

[23] C. S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multipliers for finite fields GF(2^m)," *IEEE Transactions on Computers*, no. 4, pp. 357–360, 1984.

[24] C. L. Wang and J. L. Lin, "Systolic array implementation of multipliers for finite fields GF(2^m)," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 7, pp. 796–800, 1991.

[25] C. W. Wu and M. K. Chang, "Bit-level systolic arrays for finite-field multiplications," *Journal of VLSI Signal Processing Systems*, vol. 10, no. 1, pp. 85–92, 1995.

[26] S. K. Jain and K. K. Parhi, "Low latency standard basis GF(2^m) multiplier and squarer architectures," International Conference on Acoustics, Speech, Signal Processing. Detroit, USA: IEEE, 5 1995, pp. 2747–2750.

[27] S. K. Jain, L. Song, and K. K. Parhi, "Efficient semi-systolic architectures for finite-field arithmetic," *IEEE Transactions on VLSI Systems*, vol. 6, no. 1, pp. 101–113, 1998.

[28] C. K. Koc and T. Acar, "Montgomery multiplication in GF(2^k)," *Design, Codes and Cryptography*, vol. 14, no. 1, pp. 57–69, 1998.

[29] S. Kwon, C. H. Kim, and C. P. Hong, "A systolic multiplier with LSB-first algorithm over GF(2^m) which is as efficient as the one with MSB-first algorithm," International Symposium on Circuits and Systems. Bangkok, Thailand: IEEE, 5 2003, pp. 633–636.

[30] C. Y. Lee, Y. H. Chiu, and C. W. Chiou, "New bit-parallel systolic multiplier over GF(2^m) using the modified Booth's algorithm," Asia Pacific Conference on Circuits and Systems. Singapore: IEEE, 12 2006, pp. 610–613.

[31] S. Kwon, C. H. Kim, and C. P. Hong, "Unidirectional two dimensional systolic array for multiplication in GF(2^m) using LSB-first algorithm," International Workshop on Fuzzy Logic Applications. Crema, Italy: Springer-Verlag, 9 2005, pp. 420–426.

[32] C. W. Chiou, C. Y. Lee, A. W. Deng, and J. M. Lin, "Efficient VLSI implementation for Montgomery multiplication in GF(2^m)," *Tamkang Journal of Science and Engineering*, vol. 9, no. 4, pp. 365–372, 2006.

[33] C. Y. Lee, C. C. Chen, Y. H. Chen, and E. H. Lu, "Low-complexity bit-parallel systolic multipliers over GF(2^m)," International Conference on System, Man and Cybernetics. Taipei, Taiwan: IEEE, 10 2006, pp. 1–6.

[34] C. W. Chiou, C. Y. Lee, and J. M. Lin, "Finite field polynomial multiplier with linear feedback shift register," *Tamkang Journal of Science and Engineering*, vol. 10, no. 3, pp. 253–264, 2007.

[35] C. Y. Lee, "Low-complexity bit-parallel systolic multipliers over GF(2^m)," *Integration, The VLSI Journal*, vol. 41, no. 1, pp. 106–112, 2008.

[36] ——, "Multiplexer-based bit-parallel systolic multipliers over GF(2^m)," *Computers & Electrical Engineering*, vol. 34, no. 5, pp. 392–405, 2008.

[37] S. Kwon, C. H. Kim, and C. P. Hong, "More efficient systolic arrays for multiplication in GF(2^m) using LSB first algorithm with irreducible polynomials and trinomials," *Computers & Electrical Engineering*, vol. 35, no. 1, pp. 159–167, 2009.

[38] K. W. Kim and J. C. Jeon, "Polynomial basis multiplier using cellular systolic architecture," *IETE Journal of Research*, vol. 60, no. 2, pp. 194–199, 2014.

[39] G. Seroussi, "Table of low-weight binary irreducible polynomials," Hewlett-Packard, Palo Alto, USA, Tech. Rep., 08 1998.

[40] C. Y. Lee, "Low-latency bit-parallel systolic multiplier for irreducible $x^m + x^n + 1$ with $gcd(m, n) = 1$," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 55, no. 3, pp. 828–837, 2003.

[41] ——, "Low complexity bit-parallel systolic multiplier over GF(2^m) using irreducible trinomials," *IEE Proceedings on Computers and Digital Techniques*, vol. 150, no. 1, pp. 39–42, 2003.

[42] C. Y. Lee, J. S. Horng, I. C. Jou, and E. H. Lu, "Low-complexity bit-parallel systolic montgomery multipliers for special classes of GF(2^m)," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1061–1070, 2005.

[43] C. Y. Lee, C. C. Chen, and E. H. Lu, "Compact bit-parallel systolic montgomery multiplication over GF(2^m) generated by trinomials," TENCON. Hong Kong, China: IEEE, 1 2006, pp. 1–4.

[44] C. Y. Lee, Y. H. Chen, C. W. Chiou, and J.-M. Lin, "Unified parallel systolic multiplier over GF(2^m)," *Journal of Computer Science and Technology*, vol. 22, no. 1, pp. 28–38, 2007.

[45] C. Y. Lee, C. W. Chiou, J. M. Lin, and C.-C. Chang, "Scalable and systolic montgomery multiplier over GF(2^m) generated by trinomials," *IET Circuits, Devices, and Systems*, vol. 1, no. 6, pp. 477–484, 2007.

[46] P. K. Meher, "Systolic and super-systolic multipliers for finite field GF(2^m) based on irreducible trinomials," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 4, pp. 1031–1040, 2008.

[47] C. Y. Lee, "Low-complexity parallel systolic montgomery multipliers over GF(2^m) using Toeplitz matrix-vector representation," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 91, no. 6, pp. 1470–1477, 2008.

[48] C. W. Chiou, J. M. Lin, C. Y. Lee, and C.-T. Ma, "Novel Mastrovito multiplier over GF(2^m) using trinomial," International Conference on Genetic and Evolutionary Computing. Kitakyushu, Japan: Sringer, 09 2011, pp. 237–242.

[49] S. Bayat-Sarmadi and M. Farmani, "High-throughput low-complexity systolic Montgomery multiplication over GF(2^m) based on trinomials," *IEEE Transactions on Circuits and Systems II*, vol. 62, no. 4, pp. 377–381, 2015.

[50] P. K. Meher, "Systolic and non-systolic scalable modular designs of finite field multipliers for Reed-Solomon codec," *IEEE Transactions on VLSI Systems*, vol. 17, no. 6, pp. 747–757, 2009.

[51] J. F. Xie, J. J. He, and W. H. Gui, "Low latency systolic multipliers for finite field GF(2^m) based on irreducible polynomials," *Journal of Central South University*, vol. 19, no. 5, pp. 1283–1289, 2012.

[52] J. F. Xie, J. J. He, and P. K. Meher, "Low latency systolic Montgomery multiplier for finite field GF(2^m) based on pentanomials," *IEEE Transactions on VLSI Systems*, vol. 21, no. 2, pp. 385–389, 2013.

[53] S. S. Erdem, T. Yanik, and C. K. Koc, "Polynomial basis multiplication over GF(2^m)," *Acta Applicandae Mathematicae*, vol. 93, no. 1, pp. 33–55, 2006.

[54] F. Rodriguez-Henriquez, A. D. Perez, N. A. Saqib, and C. K. Koc, "Binary finite field arithmetic," in *Cryptographic Algorithms on Reconfigurable Hardware*. New York, USA: Springer, 2007, ch. 6, pp. 139–188.

[55] J. L. Imana, "Low latency GF(2^m) polynomial basis multiplier," *IEEE Transactions on Circuits and Systems I*, vol. 58, no. 5, pp. 935–946, 2011.

[56] ST microelectronics. [Online]. Available: `http://www.st.com/`

[57] A. Brokalakis, A. Kakarountas, and C. Goutis, "A high-throughput area efficient FPGA implementation of AES-128 encryption," Workshop on Signal Processing Systems Design and Implementation. Athens, Greece: IEEE, 11 2005, pp. 116–121.

[58] P. Chodowiec and K. Gaj, "Implementation of the Twofish cipher using FPGA devices," George Mason University, Fairfax, USA, Tech. Rep., 07 1999.

[59] K. K. Parhi, *VLSI digital signal processing systems: Design and Implementation*, 1st ed. London, UK: John Wiley and Sons, 2007.